
MM M.Morris sss
MM Mano ooooo
BB Bilgisayar rrr
SS Sistemleri iiii
MM Mimarisi iiiii

BİLGİSAYAR SİSTEMLERİ MİMARİSİ

BİLGİSAYAR SİSTEMLERİ MİMARİSİ

M. Morris Mano

Çevirenler:
Abdüssamet Marşoğlu
Nurşen Suçsuz

Literatür Yayınları: 89

BİLGİSAYAR SİSTEMLERİ MİMARİSİ
M. Morris Mano

Çevirenler: Abdüssamet Marşoğlu, Nurşen Suçsuz

Birinci Basım: Ekim 2002
Beşinci Basım: Eylül 2015
Altıncı Basım: Mart 2018

Genel Yayın Yönetmeni: Kenan Kocatürk
Kapak Tasarım: Yetkin Başarır
Baskı ve Cilt: Pasifik Ofset
Cihangir Mah. Güvercin Cad. Baha İşmerkezi
A Blok No: 3/1 Kat: 2 Avcılar - İstanbul
Sertifika No: 12027
Tel: (0212) 412 17 77

ISBN: **978-975-8431-31-1**

© 1993, 1982, 1976 by Prentice-Hall, Inc.
A Simon & Schuster Company
Englewood Cliffs, New Jersey 07632

The author and Publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and Publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

© Copyright 2001, **Literatür:** Yayıncılık

Bu kitabın yayın hakları Literatür Yayıncılık Dağıtım Pazarlama San.
ve Tic. Ltd. Şti.'ne aittir. Kitabın tamamı veya bir bölümü hiçbir biçimde
fotokopi yoluyla çoğaltılamaz, dağıtılamaz, yeniden elde edilmek üzere saklanamaz.

Sertifika No: 10843

Literatür: Yayıncılık, Dağıtım, Pazarlama, San. ve Tic. Ltd. Şti.
İstiklal Caddesi, İpek Sokak, No: 10 Kat: 2
Beyoğlu 34433 İstanbul
T (0212) 292 4120
F (0212) 245 5987
E literatur@literatur.com.tr
www.literatur.com.tr

İçindekiler

Önsöz **xv**

BÖLÜM BİR
Sayısal Mantık Devreleri **1**

1.1 Sayısal Bilgisayarlar 1
1.2 Mantık Kapıları 4
1.3 Boole Cebri 7
Bir fonksiyonun Tümleyeni 10
1.4 Devre Sadeleştirme 11
Toplamların Çarpımı Basitleştirmesi 14
Dikkate Almama 17
1.5 Birleşik Devreler 18
Yar-Toplayıcı 19
Tam-Toplayıcı 20
1.6 Yaz-Bozlar 22
SR Tipi Yaz-boz 22
D Tipi Yaz-boz 23
JK Tipi Yaz-boz 24
T Tipi Yaz-boz 24
Kenar Tetiklemeli Yaz-bozlar 25
Yükseltgeme Çizelgeleri 26
1.7 Sıralı Devreler 27
Yaz-boz Giriş Fonksiyonları 28
Durum Çizelgesi 29
Durum Şeması 31
Çizim Örneği 32
Tasarım İşlemi 34
Sorular 35
Kaynaklar 37

BÖLÜM İKİ

Sayısal Elemanlar **39**

2.1 Birleştirilmiş Devreler 39
2.2 Kod Çözücüler 41
VE-DEĞİL Kapılı Kod çözücü 43
Kod Çözücülerin Genişletilmesi 43
Kodlayıcılar 44
2.3 Seçiciler 45
2.4 Yazaçlar 48
Paralel Yüklemeli Yazaçlar 49
2.5 Kaydırma Yazaçları 49
Paralel Yüklemeli Çift Yönlü Kaydırma Yazaçları 51
2.6 İkili Sayıcılar 53
Paralel Yüklemeli İkili Sayıcı 55
2.7 Bellek Birimi 57
Adreslenebilir Bellek 57
Yalnız Okunabilir Bellek 59
ROM Çeşitleri 60
Sorular 61
Kaynaklar 63

BÖLÜM ÜÇ

Veri Gösterimi **65**

3.1 Veri Tipleri 65
Sayı Sistemleri 66
Sekizlik ve Onaltılık Sayılar 67
Ondalık Gösterim 69
Alfasayısal Gösterim 71
3.2 Tümleyenler 72
($r-1$)'e Göre Tümleyen 72
r'ye Göre Tümleyen 73
İşaretsiz sayılarda çıkarma 73
3.3 Sabit Nokta Gösterimi 75
Tamsayı Gösterim 75
Aritmetik Toplama 76
Aritmetik Çıkarma 77
Taşma 77
Ondalık Sabit Nokta Gösterimi 78

3.4 Kayan Noktalı Gösterim 79
3.5 Diğer İkili Kodlar 80
Gray kodu 80
Diğer Ondalık Kodlar 81
Diğer alfasayısal kodlar 82
3.6 Hata Bulma Kodları 83
Sorular 84
Kaynaklar 86

BÖLÜM DÖRT
Yazaç Aktarımı ve Mikro İşlemler 89
4.1 Yazaç Aktarım Dili 89
4.2 Yazaçlar Arası Aktarım 90
4.3 Veri Yolu ve Bellek Aktarımı 93
Üç Durumlu Veri Yolu Buffer ları 95
Bellek Aktarımı 96
4.4 Aritmetik Mikro İşlemler 97
İkili Toplayıcı 98
İkili Toplayıcı-çıkartıcı 99
İkili Arttırıcı 100
Aritmetik Devre 101
4.5 Mantıksal Mikro İşlemler 103
Mantıksal Mikro İşlemlerin Listesi 104
Donanımın Kurulumu 105
Bazı Uygulamalar 106
4.6 Kaydırma Mikro İşlemleri 108
Donanım Kurulumu 110
4.7 Aritmetik Mantık Kaydırma Birimi 111
Sorular 113
Kaynaklar 116

BÖLÜM BEŞ
Temel Bilgisayar Yapısı ve Tasarımı 117

5.1 Buyruk Kodu 117
Saklanmış Program Yapısı 119
Dolaylı Adres 120

5.2 Bilgisayar Yazaçları 121
Ortak Veri Yolu Sistemi 123
5.3 Bilgisayar Buyrukları 126
Buyruk Kümesinin Tamamlılığı 128
5.4 Zamanlama ve Denetim 129
5.5 Buyruk Süreci 133
Al-getir ve Kodunu-çöz 133
Buyruk Tipinin Belirlenmesi 135
Yazaç Adreslemeli Buyruklar 137
5.6 Bellek-Adreslemeli Buyruklar 138
AND: AC yi VE le 139
ADD: AC ye Ekle 139
LDA: AC ye Yükle 140
STA: AC yi Belleğe Aktar 140
BUN: Şartsız Dallanma 140
BSA: Dallan ve Geri Dönüş Adresini Sakla 141
ISZ: Arttır ve Eğer Sıfır İse Atla 142
Denetim Akış Şeması 142
5.7 Giriş-Çıkış Buyrukları ve Kesmeler 144
Giriş/Çıkış Düzeni 144
Giriş/Çıkış Buyrukları 145
Program Kesme 146
Kesme Süreci 148
5.8 Bilgisayarın Tamamlanmış Tanımı 149
5.9 Temel Bilgisayarın Tanımı 150
Denetim Mantık Kapıları 150
Yazacın ve Belleğin Denetimi 153
Tek Yaz-bozların Denetimi 154
Ortak Veri Yolunun Denetimi 155
5.10 İşlemci Yazacı Mantık Tasarımı 156
AC Yazacının Denetimi 157
Toplayıcı ve Mantık Devre 157
Sorular 159
Kaynaklar 163

BÖLÜM ALTI
Temel Bilgisayarın Programlanması 165

6.1 Giriş 165
6.2 Makine Dili 167

6.3 Birleştirici Dil 170
Dilin Kuralları 170
Bir Örnek 172
İkili'ye Çevrilme 172
6.4 Derleyici 174
Sembolik Programın Bellekteki Görünüşü 174
İlk Geçiş 176
İkinci Geçiş 177
6.5 Program Döngüleri 180
6.6 Aritmetik ve Mantıksal İşlemlerin Programlanması 182
Çarpma Programı 182
Çift Duyarlıklı Toplama 184
Mantıksal İşlemler 187
Kaydırma İşlemleri 187
6.7 Alt Programlar 188
Alt Program Parametreleri ve Veri Bağlantıları 190
6.8 Giriş / Çıkışın Programlanması 192
Karakter İşleme 192
Program Kesmeleri 194
Sorular 196
Kaynaklar 199

BÖLÜM YEDİ
Mikro Programlanmış Denetim 201
7.1 Denetim Belleği 201
7.2 Adres Sıralama 203
Şartlı Dallanma 206
Buyruğun Haritalanması 206
Alt Programlar 207
7.3 Mikro Program Örneği 208
Bilgisayar Konfigürasyonu 208
Mikro Buyruk Biçimi 210
Sembolik Mikro Buyruk 213
Al-getir Yordamı 213
Sembolik Mikro Program 214
İkili Mikro Program 216
7.4 Denetim Birimi Tasarımı 217
Mikro Program Sıralayıcı 219
Sorular 221
Kaynaklar 224

BÖLÜM SEKİZ

Merkezi İşlemci Birimi **227**

8.1 Giriş 227

8.2 Genel Yazaç Kurulumu 228
Denetim Kelimesi 230
Mikro İşlem Örnekleri 231

8.3 Yığıt Kurulumu 232
Yazaç Yığıtı 233
Bellek Yığıtı 234
Zıt Polonyalı Gösterimi 236
Aritmetik İfadelerin Değerlendirilmesi 238

8.4 Buyruk Biçimleri 239
Üç Adresli Buyruklar 242
İki Adresli Buyruklar 242
Bir Adresli Buyruklar 243
Sıfır Adresli Buyruklar 243
RISC Buyrukları 244

8.5 Adresleme Kipleri 244
Sayısal Örnek 247

8.6 Veri Aktarımı ve İşleme 249
Veri Aktarım Buyrukları 249
Veri İşleme Buyrukları 251
Aritmetik Buyruklar 251
Mantıksal ve Bit İşleme Buyrukları 253
Kaydırma Buyrukları 254

8.7 Program Denetimi 255
Durum Bit Şartları 256
Şartlı Dallanma Buyrukları 257
Alt Program Çağırma ve Geri Dönüş 260
Program Kesme 261
Kesme Çeşitleri 263

8.8 Azaltılmış Buyruk Kümeli Bilgisayar (RISC) 264
CISC Özellikleri 264
RISC Özellikleri 265
Üst Üste Binen Yazaç Pencereleri 266
Berkeley RISC I 269
Sorular 272
Kaynaklar 278

BÖLÜM DOKUZ

Boru Hattı ve Vektör İşlemleri 281

9.1 Paralel İşleme 281

9.2 Boru Hattı İşlemleri 284

Genel Düşünceler 286

9.3 Aritmetik İşlem Boru Hattı 289

9.4 Buyruk Boru Hattı 291

Örnek: Dört Kesimli Buyruk Boru Hattı 292

Veri Beklentisi 295

Dallanma İşlemlerinin İdaresi 295

9.5 RISC Boru Hattı 297

Örnek: Üç Kesimli Boru Hattı 297

Gecikmeli Yükleme 298

Gecikmiş Dallanma 300

9.6 Vektör İşlemleri 301

Vektör İşlemleri 302

Matris Çarpımı 303

Bellek Ayrıştırma 305

Süper Bilgisayarlar 306

9.7 Dizi İşlemcileri 307

Eklenmiş Dizi işlemci 307

SIMD Dizi İşlemcisi 308

Sorular 309

Kaynaklar 311

BÖLÜM ON

Bilgisayar Aritmetiği 313

10.1 Giriş 313

10.2 Toplama ve Çıkarma 314

İşaretli Sayılarda Toplama ve Çıkarma 314

Donanımın Kurulması 315

Donanım Algoritması 317

İşaretli 2 ye Göre Tümleyen ile Toplama Çıkarma 318

10.3 Çarpma Algoritmaları 319

İşaretli Sayılar İçin Donanımın Kurulması 320

Donanım Algoritması 321

Booth Çarpım Algoritması 321

Dizi Çarpım 326

10.4 Bölme Algoritmaları 328
İşaretli Sayılar İçin Donanım Kurulması 328
Bölüm Taşması 330
Donanım Algoritması 331
Diğer Algoritmalar 332
10.5 Kayan Noktalı Aritmetik İşlemler 333
Temel Düşünceler 333
Yazaç Yapılanması 335
Toplama ve Çıkarma 336
Çarpma 338
Bölme 340
10.6 Ondalık Aritmetik Birim 342
BCD Toplayıcı 343
BCD Çıkarma 345
10.7 Ondalık Aritmetik İşlemler 347
Toplama ve Çıkarma 348
Çarpma 350
Bölme 351
Kayan Noktalı İşlemler 353
Sorular 353
Kaynaklar 357

BÖLÜM ON BİR
Giriş / Çıkış Organizasyonu 359
11.1 Çevre Aletleri 359
ASCII Alfasayısal Karakterler 361
11.2 Giriş / Çıkış Ara Yüzeyi 362
G/Ç Veri Yolu ve Ara yüzey Modülleri 363
G/Ç a Karşı Bellek Veri Yolu 365
Yalıtılmışa Karşı Bellek-Haritalanmış G/Ç 365
G/Ç Ara Yüzey Örneği 366
11.3 Eş Zamanlı Olmayan Veri Aktarımı 368
Başla Denetimi 369
Onaylama 370
Eş Zamanlı Olmayan Seri Aktarım 373
Eş Zamanlı Olmayan Haberleşme Ara yüzeyi 375
FİFO İlk Giren İlk Çıkar Buffer 377
11.4 Aktarma Kipleri 379
Programlanmış G/Ç Örneği 380
Kesme Başlatmalı G/Ç 381
Yazılım İncelenmesi 383

11.5 Kesme İstemi Önceliği 383
Papatya Zinciri Önceliği 384
Paralel Öncelikli Kesme 386
Öncelik Kodlayıcı 388
Kesme Çevrimi 389
Yazılım Yordamları 389
Başlangıç ve Sonuç İşlemleri 391
11.6 Doğrudan Bellek Erişimi (DMA) 392
DMA Denetim Birimi 393
DMA Aktarımı 395
11.7 Giriş / Çıkış İşlemcisi (GÇİ) 396
MİB-GÇİ Haberleşmesi 398
IBM 370 G/Ç Kanalı 400
Intel 8089 GÇİ 402
11.8 Seri Haberleşme 406
Karakter Esaslı Tutanak 408
Aktarma Örneği 409
Veri Saydamlığı 411
Bit Esaslı Tutanak 412
Sorular 414
Kaynaklar 418

BÖLÜM ON İKİ
Bellek Tasarımı **419**
12.1 Bellek Hiyerarşisi 419
12.2 Ana Bellek 422
RAM ve ROM Yongaları 422
Bellek Adres Haritası 424
Bellekle MİB Bağlantısı 425
12.3 Yardımcı Bellek 427
Manyetik Disk 428
Manyetik Teyp 429
12.4 İçerik Adreslemeli Bellek 429
Donanım Tasarımı 430
Çakıştırma Mantığı 432
Okuma İşlemi 433
Yazma İşlemi 434

12.5 Ön Bellek 434
İçerik Adreslemeli Haritalama 436
Doğrudan Haritalama 437
Kümelenmiş İçerik Adreslemeli Haritalama 439
Ön Belleğe Yazma 440
Ön Bellek İlklendirmesi 441
12.6 Sanal Bellek 441
Adres Uzayı ve Bellek Uzayı 441
Sayfa Kullanarak Adres Haritalama 443
İçerik Adreslemeli Bellek Sayfa Çizelgesi 445
Sayfa Değiştirme 446
12.7 Bellek Yönetim Donanımı 447
Kesimlenmiş-Sayfa Haritalaması 448
Sayısal Örnek 450
Bellek Koruması 453
Sorular 454
Kaynaklar 457

BÖLÜM ON ÜÇ
Çoklu İşlemciler 459
13.1 Çoklu İşlemcilerin Karakteristikleri 459
13.2 İç Bağlantı Yapıları 461
Zaman Paylaşımlı Ortak Veri Yolu 461
Çok Kapılı Bellek 463
Çapraz Bağlantı Anahtarı 464
Çok Seviyeli Anahtarlamalı Ağ 466
Hiper Küp İç Bağlantısı 468
13.3 İşlemciler Arası Seçim 470
Sistem Veri Yolu 470
Seri Veri Yolu Seçim Kararı 472
Paralel Veri Yolu Seçim Kararı 474
Dinamik Seçim Kararı Algoritması 475
13. 4 İşlemciler Arası İletişim ve Eş Zamanlılık 476
İşlemciler Arası Zaman Uyumluluğu 477
Semafor ile Karşılıklı Dışlamalar 478
13.5 Ön Bellek Ardışıklığı 479
Ardışıksızlık Koşulları 479
Ön Bellek Uyumluluğu Sorununa Çözümler 481
Sorular 483
Kaynaklar 484

Önsöz

Bu kitap bilgisayar kurulumu ve tasarımıyla olduğu kadar mimarisiyle de uğraşmaktadır. Bilgisayar mimarisi, bilgisayarın çeşitli işlevsel kısımlarının yapı ve davranışlarıyla ve bunların etkileşiminin kullanıcının işlemsel gereksinimlerini nasıl karşıladığıyla ilgilidir. Bilgisayar kurulumu, donanım bileşenlerinin bilgisayarı oluşturmak üzere birbirlerine bağlanma yöntemiyle ilgilidir. Bilgisayar tasarımı, bazı özellikleri sağlayan donanımın geliştirilmesiyle ilgilidir.

Bu kitap, sayısal bilgisayarlarda donanım işlemlerini anlamak için gerekli temel bilgileri sunmakta ve bilgisayar donanımıyla ilgili üç konuyu kapsamaktadır. 1'den 4'e kadar olan bölümlerde, bilgisayarın kurulum ve tasarımında kullanılan sayısal bileşenler anlatılmaktadır. 5'ten 7'ye kadar olan bölümlerde, basit bir temel bilgisayar tasarlamak isteyen tasarımcının atması gereken adımlar ayrıntılı olarak verilmektedir. 8'den 10'a kadar olan bölümler, merkezi işlemci biriminin kurulum ve mimarisiyle ilgilidir. Bölüm 11 ve 12, giriş-çıkışın ve belleğin kurulum ve mimarisini anlatmaktadır. Bölüm 13 çoklu işlem kavramını ele almaktadır. Bu kitabın düzeni, basit konuları önce ele alıp ileri kavramları sonra sunmak biçimindedir. Dolayısıyla ilk yedi bölüm, basit bir sayısal bilgisayarın kurulum, tasarım ve programlanmasının anlaşılması için gerekli temel bilgileri içermektedir. Son altı bölüm, ayrık işlevsel birimlerin kurulum ve mimarisini, daha gelişmiş konuları vurgulayarak anlatmaktadır.

Üçüncü baskıda bilgiler, ikinci baskıyla aynı tarzda kurulmuştur ve konuların çoğu aynı kalmıştır. Buna rağmen, üçüncü baskıda ikinci baskıya göre birçok gelişmiş kısım vardır. İkisi (6 ve 10) hariç tüm bölümler baştan sona gözden geçirilip düzeltilerek bilgilerin güncelleşmesi ve sunuşun açıklık kazanması sağlanmıştır. İki yeni bölüm eklenmiştir. Bölüm 9 boru hattı ve vektör işlemleri üzerinedir. Bölüm 13 çoklu işlemcileri anlatmaktadır. İki bölüm, azaltılmış buyruk kümeli bilgisayarlar (RISC) ile ilgilidir. Bölüm 5 tümüyle değiştirilip yenilenmiş, böylece temel bilgisayarın tasarımı basitleştirilmiştir. On üç bölümün on birine yeni sorular konmuştur.

Bir bilgisayarın yazaçlarla birlikte fiziksel kurulumu, veri akışı, mikro işlemler ve denetim fonksiyonları, sembolik olarak bir donanım tanımlama diliyle anlatılabilir. Bu kitapta biz basit bir yazaç aktarım dili geliştirdik ve bunu birçok bilgisayar işleminde özlü ve duyarlı biçimde kullandık. Yazaç aktarım dilinin donanım kurulumu ve tasarımıyla bağlantısı bütünüyle açıklandı.

Kitap, bilgisayar donanımı için ön bilgi gerektirmemektedir. Verilen bilgiler başka ilk bilgiler olmaksızın anlaşılabilir. Fakat birleştirici dilde biraz deneyim sahibi olmak, anlamayı kolaylaştırır. Bölüm 1'den 3'e kadar olan kısım, okuyucu sayısal mantık tasarımını biliyorsa atlanabilir.

Aşağıda, her bölümün konularının, yapılan düzeltmelerin özellikle vurgulandığı kısa tanıtımlarını bulacaksınız.

Bölüm 1 kapılar ve yaz-bozlardan oluşan sayısal sistemler hakkında temel bilgileri içermektedir. Bölüm, Boole cebri, birleşik devreler ve sıralı devreleri kapsamaktadır. Bu sayede, sunulan sayısal devrelerin anlaşılması için gerekli temel sağlanmaktadır.

Bölüm 2 en yaygın standart sayısal bileşenlerin mantıksal işleyişini ayrıntılı bir biçimde anlatmaktadır. Bileşenler içinde kod çözücüler, seçiciler, yazaçlar, sayaçlar ve bellekler bulunmaktadır. Bu sayısal bileşenler bundan sonraki bölümlerde daha büyük birimlerin tasarımında yapıtaşları olarak kullanılmaktadır.

Bölüm 3 sayısal bilgisayarlarda çeşitli veri tiplerinin bilgisayar yazaçları içinde ikili düzende nasıl temsil edildiğini göstermektedir. Ağırlıklı olarak üzerinde durulan konu, aritmetik işlemlerde kullanılan sayıların gösterimi ve veri işlemede kullanılan sembollerin ikili kodlamasıdır.

Bölüm 4'te yazaç aktarım dili tanımlanmakta ve bu dilin mikro işlemleri sembolik biçime dönüştürmekte nasıl kullanıldığı anlatılmaktadır. Semboller aritmetik, mantık ve kaydırma mikroişlemleri için tanımlanmıştır. Çok kullanılan mikro işlemlerin donanımsal tasarımını göstermek için, birleştirilmiş bir aritmetik, mantık ve kaydırma birimi geliştirilmiştir.

Bölüm 5 temel sayısal bilgisayarın kurulum ve tasarımını vermektedir. Bu temel bilgisayar ticari bilgisayarlara göre her ne kadar basit olsa da, genel amaçlı, yerleştirilmiş programlı bir bilgisayarın işlevsel yeteneklerini göstermeye yeterlidir. Yazaç aktarım dili, bilgisayarın iç işleyişini tanımlamak ve tasarımındaki gereksinimleri belirlemekte kullanılmıştır. Temel bilgisayar, ikinci baskıdaki temel bilgisayarla aynı buyruk kümesine sahip olmakla birlikte, donanım kurulum tasarımı tümüyle değiştirilmiştir. Bu bölümde anlatılan tasarım adımlarını ayrıntılarıyla öğrenen öğrenciler sayısal bilgisayarın iç işleyişini kolayca anlayacaklardır.

Bölüm 6 temel bilgisayarın yirmi beş buyruğunu kullanarak birleştirici dil programlamasında kullanılan yöntemleri uygulamalı biçimde göstermektedir. Birkaç veri işleme gereksinimi için programlama örnekleri verilmiştir. Sembolik kodla ikili düzendeki programlar arasındaki bağlantılar örneklerle açıklanmıştır. Derleyicinin temel işlemleri, sembolik koddan ikili düzende eşdeğer programlara geçişi gösterilerek anlatılmıştır.

Bölüm 7 mikro programlama kavramını getirmektedir. Tipik bir buyruk kümesi için mikro kodların nasıl yazılacağını göstermek için mikro programlanmış bir denetim birimi geliştirilmiştir. Denetim biriminin tasarımı, mikro program sıralayıcının donanımsal yapısını da içeren ayrıntılarla yerine getirilmiştir.

Bölüm 8 merkezi işlem birimi (MİB) ile uğraşmaktadır. Merkezi işlem biriminin genel yazaç kurulumu ortak veri yolu olan bir icra birimi ve bir aritmetik mantık birimi geliştirilerek gösterilmiştir. Bellek yığıtının işlemesi, bazı uygulamaları verilerek açıklanmıştır. Çeşitli buyruk biçimleri birçok adresleme kipiyle birlikte gösterilmiştir. Bilgisayarda ortak olan buyruklar, yaptıkları işi anlatacak biçimde uygulamalı olarak anlatılmış ve uygulanmıştır. Son kısım, azaltılmış buyruk kümesi (RISC) bilgisayarlarını tanıtmakta, bunların yararlarını ve özelliklerini tartışmaktadır.

Bölüm 9 boru hattı ve vektör işlemleri konusunu ele almaktadır ve üçüncü baskıda yeni bir bölümdür. (İkinci baskıdaki aritmetik işlemler konusunu işleyen kısımlar onuncu bölüme kaydırılmıştır.) Boru hattı kavramı ve bunun hızlandırmayı nasıl sağladığı çeşitli örneklerle gösterilmiştir. Hem aritmetik hem buyruk boru hattı ele alınmıştır. RISC işlemcilerin, gecikmeli yükleme ve gecikmeli dallanma yöntemleriyle buyruk boru hattını etkili biçimde kullanarak tek çevrimli buyruk icrasını nasıl yaptıkları gösterilmiştir. Vektör işlemleri ele alınmış, boru hattı yöntemlerini kullanan kayan noktalı sayı işlemleri örneklerle verilmiştir.

Bölüm 10 toplama, çıkarma, çarpma ve bölme için aritmetik algoritmaları sunmakta, bunların sayısal donanımla nasıl uygulanır hale getirildiğini göstermektedir. İşaretli büyüklük ve 2'ye göre tümleyen gösteriminde sabit noktalı sayılar için, ikili gösterimde kayan noktalı sayılar için ve ikili kodlanmış ondalık sayılar için yöntemler geliştirilmiştir. Algoritmalar, akış şemaları yoluyla sunulmuştur. Akış şemaları, mikro işlemlerin sırasını ve bunların kurulup çalışması için gerekli denetim yargılarını belirlemek için yazaç aktarım dilini kullanmaktadır.

Bölüm 11 bilgisayarların giriş ve çıkış aygıtlarıyla haberleşmek için kullandıkları teknikleri tartışmaktadır. Ara yüz birimleri, işlemcinin çevre birimlerle etkileşim yolunu gösterir. Paralel ve seri veriler için zaman uyumsuz aktarım işlemleri açıklanmıştır. Dört çeşit aktarım tartışılmıştır:

programlanmış giriş/çıkış, kesme istemli aktarım, doğrudan bellek erişimi ve giriş/çıkış işlemcileri kullanmak. Seri veri aktarımı, belirli örnekler yoluyla açıklanmıştır.

Bölüm 12 bellek sıra düzeni kavramını tanımlamaktadır. Bellekler ön bellek, ana bellek ve sabit disk gibi yardımcı belleklerden oluşur. İçerik adreslemeli belleklerin kurulum ve işleyişleri ayrıntılı olarak açıklanmıştır. Bellek yönetimi kavramı, ön bellek ve sanal bellek sistemi için bellek gereksinimlerinin incelenmesi sırasında ele alınmıştır.

Bölüm 13 çoklu işlemcilerin temel özelliklerini ele almaktadır. Çeşitli iç bağlantı yapıları gösterilmiştir. İşlemciler arası karar verme, iletişim ve eşzamanlılık tartışılmaktadır. Ön bellek ardışıklığı sorunu bazı olanaklı çözümlerle birlikte açıklanmıştır.

Her bir bölümde sorular ve kaynakça vardır. Bazı sorular bölümde anlatılan konular için alıştırmadır. Diğerleri daha ileri seviyededir ve bilgisayarın mimarisi ve tasarımıyla ilgili sorunların çözümü amaçlanmıştır. Çözüm el kitabı yayıncı tarafından sunulmaktadır.

Bu kitap, bilgisayar donanım sistemleri dersi için uygundur. Elektrik mühendisliği, bilgisayar mühendisliği ve bilgisayar bilimleri bölümlerine okutulabilir. Kitap kısım kısım da kullanılabilir. 7'nci Bölüme kadar olan kısım bilgisayar donanımı konusunda ilk ders olabilir. Bir "Bilgisayar Tasarım ve Kurulumu" dersi de 4'üncü Bölümün özetiyle 5'ten 13'üncü Bölüme kadar olan kısmı kapsayabilir. Bu ders sayısal bilgisayarların beş işlemsel kısmını, yani denetim (Bölüm 7), işlemci (Bölüm 8 ve 9), aritmetik işlemler (Bölüm 10), giriş-çıkış (Bölüm 11) ve bellek (Bölüm 12) kapsar. Kitap, bilgisayar mimarisiyle ilgili temel bilgileri edinme ihtiyacı duyan mühendis ve bilim adamlarının kendi kendine öğrenmesi için de yararlı olacaktır.

Teşekkür

Metni gözden geçiren kişilere, özellikle Iowa Üniversitesi'nden Profesör Thomas L. Casavant'a; George Mason Üniversitesi'nden Profesör Murray R. Berkowitz'e; Brooklyn Politeknik Üniversitesi'nden Profesör Cem Ersoy'a; Kansas City, Missouri Üniversitesi'nden Profesör Upkar Varshney'e; Texas A&M Üniversitesi'nden Profesör Karan Watson'a ve Virginia Politeknik Enstitüsü'nden Profesör Scott F. Midkiff'e teşekkür ederim.

M. Morris Mano

BÖLÜM 1

Sayısal Mantık Devreleri

BU BÖLÜMDE

1-1 Sayısal Bilgisayarlar
1-2 Mantık Kapıları
1-3 Boole Cebri
1-4 Devre Sadeleştirme
1-5 Birleşik Devreler
1-6 Yaz-Bozlar
1-7 Sıralı Devreler

1-1 Sayısal Bilgisayarlar

Sayısal

Sayısal bir bilgisayar sayısal bir sistem olup çeşitli hesaplama görevlerini yapar. *Sayısal* sözcüğü bilgisayarda bilginin değişkenlerle temsil edildiğini ve bu değişkenlerin sonlu sayıda ayrık değerler aldığını anlatır. Bu değerler, bilgisayarın içinde, sonlu sayıda ayrık koruyabilen bileşenler tarafından işlenir. 0, 1, 2,..., 9 ondalık rakamları on tane ayrık değer belirtir. İlk sayısal bilgisayarlar 1940 yılının sonunda geliştirilmiş olup temel olarak sayısal hesaplamalar için kullanılmıştır. Bu durum için ayrık elemanlar rakamlardır. Bu uygulamanın sonucunda *sayısal bilgisayar* terimi ortaya çıktı. Uygulamada sayısal bilgisayarlar, sadece iki durum kullanılması halinde daha güvenli işlemektedir. Bileşenlerin fiziksel kısıtlamalarından dolayı, ayrıca insan mantığı ikili düşünmeye (yani doğru-yanlış ya da evet-hayır ifadeleri gibi) daha yatkın olduğundan, sayısal bileşenler, sayısal değerler alacak biçimde kısıtlanmanın ötesinde, sadece iki değer alacak biçimde de kısıtlanmıştır ve bu nedenle de *ikili* diye anılırlar.

Bit

Sayısal bilgisayarlar ikili sayı sistemini kullanır, bunun iki tane rakamı vardır: 0 ve 1. Bu ikili rakama *bit* denir. Sayısal bilgisayarlarda bilgi

bit grupları şeklinde gösterilir. Çeşitli kodlama yöntemleri kullanılarak bit gruplarının yalnızca ikili sayıları değil diğer sembolleri de, örneğin ondalık sayıları veya alfabenin harflerini göstermesi sağlanır. Çeşitli kodlama yöntemlerinin ve ikili düzenlemelerin akıllıca ve mantıklı kullanılmasıyla, bit grupları, çeşitli işlemlerin yapılması için gerekli buyruk kümeleri oluşturabilir.

10 tabanlı sistemi kullanan olağan ondalık sayıların aksine, ikili sayılar 2 tabanlı sistemi kullanır ve bunun iki rakamı vardır: 0 ve 1. İkili bir sayının ondalık eşdeğerini bulmak için, onu 2'nin kuvvetleri şeklinde kuvvet serisine açmak gerekir. Örneğin, 1001011 ikili sayısının temsil ettiği nicelik, aşağıdaki gibi, her biti 2'nin tamsayı kuvvetleriyle çarparak ondalık sayıya dönüştürülebilir:

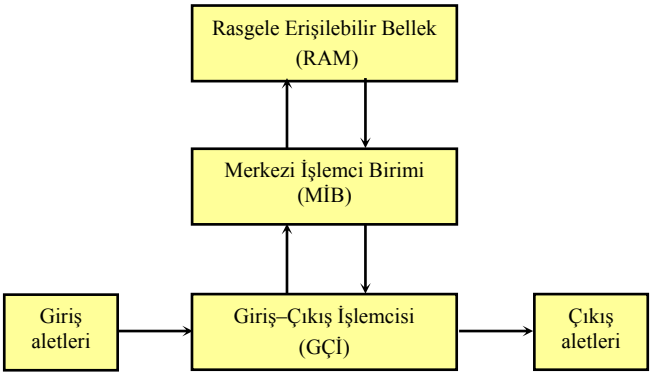
$$2 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 75$$

7 bitlik 1001011 ikili sayısı, ondalık eşdeğeri 75 olan sayıyı gösterir. Buna karşılık aynı bit grubu, alfabenin harflerini gösteren kod olarak kullanılırsa "K" harfini gösterir. Aynı bit grubu, bir karar mantığı içinde, bu mantığı kullanan bilgisayarda denetim kodu olarak görev yapar. Yani, bit grupları, sayısal bir bilgisayarda birçok farklı şeyi temsil edebilir. Bu tıpkı, aynı alfabe harflerinin kullanılarak örneğin İngilizce ve Fransızca gibi farklı diller oluşturulmasına benzer.

Bir bilgisayar sistemi bazen iki işlevsel alt kısma ayrılır. Bunlar donanım ve yazılımdır. Bilgisayar donanımı, bilgisayarın fiziksel kısmını oluşturan tüm elektronik bileşenlerden ve elektromekanik aletlerden oluşur. Bilgisayar yazılımı, bilgisayarın çeşitli veri işleme görevlerini yerine getirmek için kullandığı buyruklardan ve verilerden oluşur. Bir dizi buyruğa *program* adı verilir. Program tarafından işlenen veriler *veri tabanı*nı oluşturur.

Program

Bir bilgisayar sistemi, donanım ile donanımın kullanacağı yazılımdan oluşur. Bilgisayarın sistem yazılımı, bilgisayarın en etkili biçimde kullanılmasını amaçlayan ve sağlayan programlar topluluğundan oluşur. Sistem yazılımı paketinde bulunan programlara topluca *işletim sistemi* denmektedir. Bunlar kullanıcı tarafından bazı özel problemlerin çözümü amacıyla yazılan uygulama programlarından ayrılır. Örneğin, kullanıcı tarafından özel bir veri işlem gereksinimini karşılamak için yüksek seviyeli bir dille yazılan bir program bir uygulama programıdır. Oysa, yüksek seviyeli dildeki programı makine diline çeviren derleyici, bir sistem programıdır. Bir bilgisayar sistemi satın alan bir müşteri, donanımın yanısıra, bilgisayarın etkili çalışmasını sağlayacak bir yazılıma da gerek duyar. Sistem yazılımı, bilgisayar sisteminin ayrılmaz bir parçasıdır. Bu sayede kullanıcı istekleriyle donanımın yetenekleri arasındaki farklar giderilmiş olur.



Şekil 1.1 Sayısal bilgisayarın blok şeması.

Bilgisayar Donanımı

Bilgisayar donanımı, Şekil 1.1'de gösterildiği gibi üç temel kısımdan oluşur. Merkezi işlem birimi (MİB) içerisinde, verileri işlemek için bir aritmetik ve mantık birimi, verileri saklamak için bir dizi yazaç, ayrıca buyrukları alıp getiren ve icra eden denetim devreleri bulunur. Bilgisayarın belleği buyrukları ve verileri saklamakta kullanılır. Buna RAM (Rasgele Erişilebilir Bellek) adı verilir. Çünkü MİB, belleğin herhangi bir adresine herhangi bir anda erişebilir ve sonlu bir zaman içinde oradaki ikili bilgiyi alabilir. Giriş ve çıkış işlemcisi (GÇİ), iletişimi sağlayan ve bilgisayarla dış dünya arasındaki bilgi akışını denetleyen elektronik devreler içerir. Bilgisayara bağlanan giriş-çıkış aletleri ise klavye, yazıcılar, uçlar, manyetik disk sürücüsü ve diğer iletişim aletlerinden oluşur.

Bu kitap bilgisayar sistemlerinde donanım işlemlerini anlamak için gerekli temel bilgileri vermektedir. Konu, okuyucunun ilgisine bağlı olarak, üç farklı yönden ele alınabilir. Bilgisayar donanımından söz ederken, bilgisayar tasarımı, bilgisayar kurulumu ve bilgisayar mimarisinin farklı şeyler olduğuna dikkat etmek gerekir.

Bilgisayar Kurulumu

Bilgisayar kurulumu, donanım bileşenlerinin çalışması ve bunların bilgisayar sistemini oluşturmak üzere birleştirilmeleriyle ilgilenir. Çeşitli bileşenlerin yerlerine takılı oldukları varsayılır ve yapılacak iş, bileşenlerin istenen biçimde çalışıp çalışmadıklarını doğrulamak için kurulum yapısının incelenmesidir.

Bilgisayar Tasarımı

Bilgisayar tasarımı, bilgisayarın donanım tasarımıyla ilgilenir. Bilgisayarın özellikleri belirlendikten sonra, tasarımcının görevi donanımı tasarlayıp geliştirmektir. Tasarımda, hangi donanımın kullanılacağı ve parçaların birbirine nasıl bağlanacağıyla uğraşılır. Donanımın bu yönüne bazen *bilgisayar kurgulaması* da denir.

Bilgisayar Mimarisi

Bilgisayar mimarisi, kullanıcının anladığı biçimde yapı ve işlevsellikle ilgilenir. Bilgi biçimlerini, buyruk kümesini ve bellek adresleme

yöntemlerini kapsar. Bir bilgisayar sisteminin mimari tasarımı, işlemciler ve bellekler gibi çeşitli işlemsel kısımların özellikleriyle ve bunların bilgisayar sistemi içinde yapılanmasıyla ilgilenir.

Bu kitap, bilgisayar donanımıyla ilgili olarak sözü edilen her üç konuyla da ilgilenmektedir. 1'den 4'e kadar bölümlerde, bilgisayar sisteminin kurulum ve tasarımında kullanılan çeşitli sayısal bileşenler sunulmuştur. 5'ten 7'ye kadar olan bölümlerde, basit bir bilgisayarın tasarımı ve programlanması için bir tasarımcının geçmesi gereken adımlar yer almaktadır. 8 ve 9'uncu bölümler merkezi işlem biriminin mimarisini anlatmaktadır. 11 ve 12'nci bölümlerde giriş-çıkış işlemcisiyle belleğin yapısı ve mimarisi sunulmuştur.

1-2 Mantık Kapıları

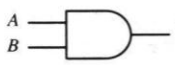
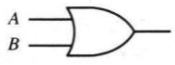
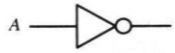
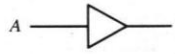
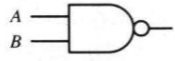
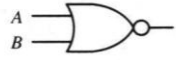
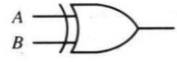
İkili bilgi sayısal bilgisayarlarda *işaret* diye adlandırılan fiziksel büyüklüklerle belirtilir. Gerilim gibi elektriksel işaretler, tüm bilgisayar içinde, tanınabilir iki durumdan birinde bulunur. Bu iki durum, ikili (iki değer alabilen) bir değişkeni gösterir ve bu değişken 1 ya da 0'a eşit olabilir. Örneğin, sayısal bir bilgisayar ikili düzende 1'i temsil etmek için 3 voltluk bir işaret, ikili düzende 0'ı temsil etmek için de 0.5 voltluk bir işaret kullanabilir. Sayısal devrelerin giriş uçları 3 ve 0.5 voltluk işaretleri kabul ederler ve devreler çıkış uçlarında 3 voltluk veya 0.5 voltluk işaretler vererek sırasıyla 1 ve 0'a karşılık gelen ikili giriş ve çıkışı gösterirler.

İkili mantık, ikili değişkenlerle ve mantıksal anlamı olan işlemlerle ilgilenir. İkili bilginin işlenmesini ve yönetimini cetvel halinde veya cebirsel formda tanımlamak için kullanılır. İkili bilgilerin işlenmesi *kapı* adı verilen mantık devreleriyle yapılır. Kapılar donanım blokları olup giriş mantığı gerekleri yerine getirildiğinde ikili düzende 1 ya da 0 çıktısı verirler. Sayısal bilgisayarlarda çok sayıda farklı kapı kullanılır. Her kapının farklı bir grafik sembolü vardır ve yaptığı işlem cebirsel bir fonksiyonla ifade edilir. Her kapı için ikili değişkenlerin giriş-çıkış bağıntıları cetvel halinde *doğruluk çizelgesi* ile gösterilir.

kapı

Şekil 1.2'de sekiz tane mantık kapısının adları, grafik sembolleri, cebirsel fonksiyonları ve doğruluk çizelgeleri verilmiştir. Her kapının bir veya iki girişi —bunlar *A* ve *B* ile gösterilmiştir—, bir tane de çıkışı vardır, bu da *x* ile gösterilmiştir. VE kapısı mantıksal VE fonksiyonu üretir. Bu fonksiyona göre, *A* ve *B*'nin ikisi de 1 ise çıkış 1'dir, aksi durumda çıkış 0'dır. Bu koşullar VE kapısı için verilen doğruluk çizelgesinde de belirtilmiştir. Tablodan görüldüğü gibi *x* çıkışı, yalnızca hem *A* hem *B* girişleri 1 iken 1 değerini almaktadır. VE fonksiyonunun cebirsel işlem sembolü aritmetikteki çarpma sembolüyle aynıdır. Yani değişkenler ara-

VE

İSİM	Grafik Sembolü	Cebirsel Fonksiyon	Doğruluk Çizelgesi
VE		$x = A \cdot B$ veya $x = AB$	A B | x 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
VEYA		$x = A + B$	A B | x 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
TERSLEYEN		$x = A'$	A | x 0 | 1 1 | 0
BUFFER		$x = A$	A | x 0 | 0 1 | 1
VE-DEĞİL		$x = (AB)'$	A B | x 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
VEYA-DEĞİL		$x = (A + B)'$	A B | x 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
YADA		$x = A \oplus B$ veya $x = A'B + AB'$	A B | x 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
YADA DEĞİL		$x = (A \oplus B)'$ veya $x = A'B' + AB$	A B | x 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

Şekil 1.2 Sayısal mantık kapıları

sına "·" konabilir veya hiçbir şey konmadan iki değişken ardarda yazılabilir. VE kapısının ikiden fazla girişi varsa, çıkışın 1 olması için tüm girişlerin 1 olması gerekir.

VEYA VEYA kapısı kapsayıcı VEYA fonksiyonunu üretir; yani VEYA kapısının çıkışının 1 olması için ya *A* girişinin ya da *B* girişinin ya da her ikisinin birden 1 olması gerekir. Aksi durumda çıkış 0'dır. VEYA fonksiyonu cebirsel toplamaya karşılık gelen + sembolüyle gösterilir. VEYA kapısının ikiden fazla girişi varsa, herhangi bir giriş 1 olursa çıkış 1 olur.

tersleyen Tersleyen devre, ikili işareti tersine çevirir. DEĞİL veya tümleyen görevi yapar. Tümleyenin cebirsel sembolü ya değişken sembolü üzerine bir çizgi ya da sağ üst köşeye üssü işaretidir. Bu kitapta, ikili bir değişkenin mantıksal tümleyeni için üssü sembolünü, 4. bölümde tanımlanan tümleyen mikro işlemini belirtmek içinse harf üzerine çizgi sembolünü kullanacağız.

Grafik sembolün çıkış ucuna konan küçük bir daire mantıksal tümleyen anlamına gelir. Tersleyen sembolündeki üçgen ise tek başına bir buffer devresini gösterir. Buffer devresi herhangi bir mantıksal fonksiyon üretmez. Yani çıkışın ikili değeri girişin aynıdır. Bu devre sadece güç yükseltimi için kullanılır. Örneğin, ikili 1 değeri için 3 volt kullanan bir buffer, girişi 3 volt olduğunda çıkışında da 3 volt üretir. Ama buffer girişinde gereken elektrik gücü miktarı bufferin çıkışında üretilen güçten çok küçüktür. Bufferin asıl görevi, daha büyük güç gerektiren kapılara elektrik sağlamaktır.

VE-DEĞİL ***VEYA-DEĞİL*** VE-DEĞİL kapısı VE kapısının tümleyenidir. Grafik sembolü VE'nin grafik sembolünün ucuna eklenen küçük bir daireyle oluşturulur. VEYA-DEĞİL kapısı VEYA'nın tümleyeni olup, sembolü VEYA sembolünün ucuna küçük bir daire eklenerek elde edilir. Her iki kapının da ikiden fazla girişi olabilir. Çıkış ise her zaman, sırasıyla VE ya da VEYA fonksiyonunun tümleyenidir.

YADA YADA kapısı sembol olarak VEYA'yı andırır, tek farkı giriş tarafında bir tane daha daire yayı olmasıdır. Eğer girişlerden bir tanesi 1 ise bu kapının çıkışı 1 olur, ama kombinasyon her iki girişin de 1 olmasına izin vermez. YADA fonksiyonun kendi cebirsel sembolü vardır, ayrıca Şekil 1.2'de görüldüğü gibi VE, VEYA ve tümleyen işlemleri cinsinden de ifade edilebilir. YADA-DEĞİL kapısı YADA'nın tümleyenidir ve grafik sembolde uçtaki küçük daireyle belirtilir. Bu kapıda ancak her iki giriş de 1 veya her iki giriş de 0 ise çıkış 1 olur. YADA işlemine "tek fonksiyon" demek aslında daha uygun olurdu; yani çıkış 1 olacaksa tek sayıda giriş 1 olmalıdır. Dolayısıyla üç girişli bir YADA (tek) fonksiyonunda çıkışın 1 olması için sadece bir giriş 1 olmalı veya üç girişin üçü de 1 olmalıdır. YADA ve YADA-DEĞİL kapıları çoğunlukla iki girişlidir, üç veya daha fazla girişlisi çok nadirdir.

1.3 Boole Cebri

Boole cebri ikili değişkenler ve mantık işlemlerinin birlikte kullanıldığı bir cebirdir. Değişkenler *A*, *B*, *x*, *y* gibi harflerle belirtilmektedir. Boole cebrindeki üç temel mantık işlemi VE, VEYA ve TÜMLEYEN'dir. Bir Boole fonksiyonu ikili değişkenler, mantık işlemi sembolleri, parantezler ve eşitlik işareti kullanılarak ifade edilebilir. Verilen değişken değerlerine göre Boole fonksiyonu ya 1 ya da 0 değerini alır. Sözgelimi aşağıdaki Boole fonksiyonunu gözönüne alalım:

Boole fonksiyonu

$$F = x + y' z$$

F fonksiyonu, $x = 1$ ise veya hem y' hem de *z*, 1'e eşitse 1 değerini alır; diğer tüm durumlarda *F*, 0'a eşittir. Fakat y', *y*'nin tümleyeni olduğundan, $y' = 1$ demek $y = 0$ demekle eşdeğerdir. Dolayısıyla, $x = 1$ iken veya $yz = 01$ iken *F* fonksiyonu 1'e eşittir diyebiliriz. Fonksiyonla ikili değişkenler arasındaki ilişki *doğruluk çizelgesiyle* gösterilir. Bir fonksiyonu doğruluk çizelgesinde göstermek için, *n* adet ikili değişkenin 2^n kombinasyonunu içeren bir listeye gerek duyarız. Şekil 1.3(a)'da görüldüğü gibi, üç değişken *x*, *y* ve *z*'ye atanan bitlerin sekiz farklı olası kombinasyonu vardır. *F* fonksiyonu, $x = 1$ veya $yz = 01$ olan kombinasyonlar için 1'e eşit, diğer tüm kombinasyonlar içinse 0'a eşittir.

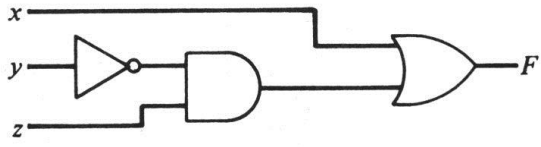
doğruluk çizelgesi

Bir Boole fonksiyonu cebirsel ifade yerine VE, VEYA ve TERSLEYEN kapılarından oluşan bir mantık şemasıyla da gösterilebilir. *F* fonksiyonunun mantık şeması Şekil 1.3(b)'de görülmektedir. *y* girişinde, tümleyeni y''yü üretmek için bir tersleyen vardır. $y'z$ terimi için bir VE kapısı bulunur ve iki terimi birleştirmek için bir VEYA kapısı kullanılır. Bir mantık şemasında, fonksiyonun değişkenleri devrenin girişleri olarak, fonksiyonun değişken sembolü (*F*) ise devrenin çıkışı olarak alınır.

Mantık Şeması

x	*y*	*z*	*F*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

a) Doğruluk çizelgesi



b) Mantık şeması

Şekil 1.3 $F = x + y'z$ fonksiyonun doğruluk çizelgesi ve devresi.

Boole cebrinin amacı sayısal devrelerin tasarım ve analizini kolaylaştırmaktır.

1. Bir doğruluk çizelgesinde cebirsel biçimdeki ifade ile ikili değişkenler arasında ilişki vardır.
2. Cebirsel biçimdeki ifade mantık şemasındaki giriş-çıkış ile ilişkilidir.
3. Bazı fonksiyonların devrelerini çizmek basittir.

Boole İfade

Bir Boole fonksiyonu doğruluk çizelgesi kullanılarak açık bir şekilde çok farklı yollardan cebirsel olarak ifade edilebilir. Bir Boole ifade Boole cebir kurallarına göre düzenlenip daha az kapı gerektiren basit bir ifade elde edilir.

Çizelge 1.1 Boole cebrinin temel tanımlarını göstermektedir. Çizelgedeki tanımların tümü doğruluk çizelgeleri kullanılarak kanıtlanabilir. İlk sekiz denklem tek değişken ve kendisi arasındaki ilişkiyi göstermektedir. 9-13 denklemleri normal matematiğe benzer. 14**.** denklem normal matematikte kullanılmaz fakat Boole ifadeler açısından çok faydalıdır. 15-16 denklemleri De Morgan teoremleri olarak adlandırılır. 17**.** denklem bir değişkenin 2 kez tümleyeninin kendisine eşit olduğunu gösterir.

Çizelge 1.1 Boole cebrinin temel tanımları

(1) $x + 0 = x$	(2) $x \cdot 0 = 0$
(3) $x + 1 = 1$	(4) $x \cdot 1 = x$
(5) $x + x = x$	(6) $x \cdot x = x$
(7) $x + x' = 1$	(8) $x \cdot x' = 0$
(9) $x + y = y + x$	(10) $x \cdot y = y \cdot x$
(11) $x + (y + z) = (x + y) + z$	(12) $x\,(yz) = (xy)z$
(13) $x\,(y + z) = xy + xz$	(14) $x + yz = (x + y)(x + z)$
(15) $(x + y)' = x'y'$	(16) $(x\,y)' = x' + y'$
(17) $(x')' = x$	

Çizelge da listelenen bu tanımlar tek değişkenlere veya ikili değişkenlerin terimleriyle ifade edilen Boole fonksiyonlarına da uygulanabilir. Örnek olarak:

$$AB' + C'D + AB' + C'D$$

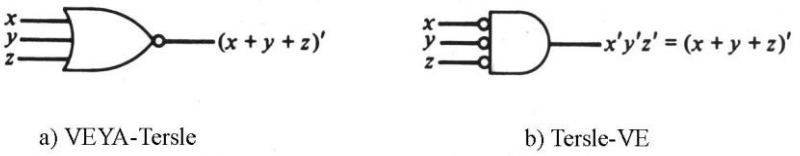
ifadesinde $x = AB' + C'D$ yazıldığında üstteki denklem $x + x$ olarak yazılır. Beşinci ifadeden $x + x = x$ ten

$$AB' + C'D + AB' + C'D = AB' + C'D$$

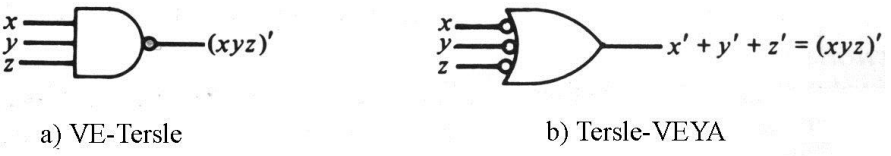
olur.

DeMorgan Teoremi

DeMorgan teoremi VEYA ve VE-DEĞİL kapıları için çok uygun ve anlaşılabilirdir. VEYA kapısında $(x + y)'$ yerine eşdeğeri olan x′y′ yazılabildiği gibi VE-DEĞİL fonksiyonunda da $(xy)'$ veya $(x' + y')$ den birisi kullanılabilir. VEYA ve VE-DEĞİL kapılarının iki farklı grafik sembolü vardır. Şekil 1.4, şekil 1.5. VEYA-DEĞİL kapısı için VEYA kapısı ve bir daire kullanmak yerine, tüm girişlere daire kullanarak VE kapısıyla yapılabilir. Benzer olarak VE-DEĞİL kapısı içinde iki farklı sembol kullanılır.



Şekil 1.4 VEYA kapısı için iki grafik sembol



Şekil 1.5 VE-DEĞİL kapısı için iki grafik sembol

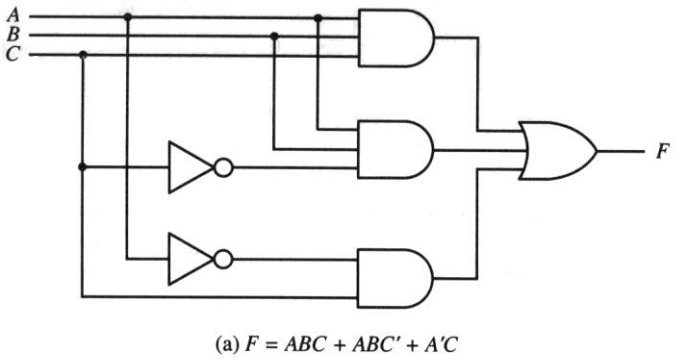
Şekil 1.6a daki mantık şeması basit sayısal devrelerden oluşmuş bir Boole fonksiyonuna aittir. Devrenin çıkışı cebirsel olarak Boole cebrinin temel açılımlarını kullanarak

$$F = BAC + BAC' + A'C = AB(C + C') + A'C$$

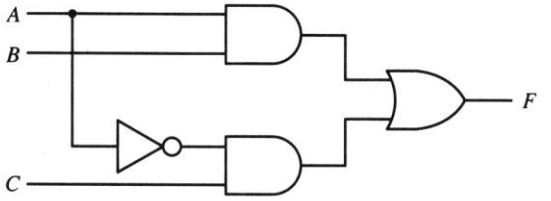
$$F = AB + A'C$$

olur (7 tanımı) $(C + C') = 1$ ve $AB \cdot 1 = AB$ den (4 tanımı). Bu da 1.5b de verilmiştir.

Mantık şemasının basit hali şekil 1.6(b) de gösterilmektedir. Şekil 1.6(a) da 6 kapı kullanılırken şekil 1.6(b9 sadece 4 kapıya ihtiyaç duyar. İki devre denktir ve doğruluk çizelgesindeki A, B, C girişleriyle F çıkışındaki ilişki aynıdır.



(a) $F = ABC + ABC' + A'C$



(B) $F = AB + A'C$

Şekil 1.6 Aynı Boole fonksiyonu için iki mantık şema

Bir Fonksiyonun Tümleyeni

Doğruluk çizelgesi kullanarak F fonksiyonunun ifadesi elde edildiğinde tümleyeni, F in doğruluk çizelgesindeki 1 ve 0 değerlerinin birbirlerinin yerlerine yazılmasıyla elde edilir. Fonksiyon cebirsel biçimde ifade edildiğinde tümleyeni DeMorgan teoremleriyle üretilir.

$$(x_1 + x_2 + x_3 + \ldots + x_n)' = x_1'\, x_2'\, x_3' \ldots x_n'$$

$$(x_1\, x_2\, x_3 \ldots x_n)' = x'_1 + x'_2 + x'_3 + \ldots + x'_n$$

Genel DeMorgan teoremlerinden cebirsel ifadenin tümleyenini bulmak için basit bir yöntem de tümleyeni alınacak fonksiyondaki tüm VE ler yerine VEYA, tüm VEYA lar yerinede VE işlemlerinin yazılması ve tüm harflerin her birinin tümleyeninin alınmasıyla yapılır.

Örneğin:

$$F = AB + C'D' + B'D$$

$$F' = (A' + B')\,(C + D)\,(B + D')$$

Tümleme ifadesi VE ve Veya işlemlerinin yerlerinin değiştirilmesi ve her bir terimin tümleyeninin alınmasıyla yapılır. C' in tümleyeninin C olduğu not edilmektedir.

1.4 Devre Sadeleştirme

Mantık şeması karışık olabilir. Bu yerleştirilen devreyi ifade eden cebirsel fonksiyonun karmaşıklığından gelir.Bir fonksiyonu ifade eden doğruluk çizelgesi tektir. Fakat bunu ifade eden cebirsel fonksiyon çok farklı biçimlerde kurulabilir. Boole cebrinin kuralları kullanılarak bu bağıntılar sadeleştirilebilir. Fakat bu işlem bazen zordur. Çünkü işlemlerin hangi sırada yapılması gerektiğini Boole cebri söylemez. Çizelge yöntemi, Boole bağıntılarını sadeleştirmede doğrudan sonuca götüren bir yöntem verir. Bu yöntem aslında doğruluk çizelgesinin şekilsel bir düzenlemesidir. Fonksiyonu cebirsel olarak ifade etmek için gerekli en az sayıda terimin seçilmesini sağlar. En az sayıda terim, en sade fonksiyondur. Bu yönteme bazen Karnough haritası veya K-haritası adı da verilir.

Minterm

Doğruluk çizelgesinde, değişkenlerin her birleşimine *minterm* denir. Örneğin şekil 1.3 teki doğruluk çizelgesi 8 minterm içerir. Bir doğruluk çizelgesinde n tane değişken için 2^n tane minterm vardır. Yani n bitten 2^n tane ikili sayı elde edildiği gibi. Bazı mintermler için Boole fonksiyonu 1, bazıları için 0 dır. Bir doğruluk çizelgesindeki bilgileri, çok kısa biçimde göstermek mümkündür. Sadece Boole fonksiyonu 1 olan mintermlerin numaraları bir araya getirilir. Örneğin şekil 1.3 teki doğruluk çizelgesi bu yolla

$$F(x, y, z) = \Sigma(1, 4, 5, 6, 7)$$

olarak ifade edilebilir. Parantez içindeki harfler, doğruluk çizelgesindeki ikili değişkenleri, çizelgedeki sıra ile ifade eder. Σ sembolü parantez içinde numarası yazılı minterm'lerin toplanacağını gösterir. Parantez içindeki minterm numaraları ondalık sistemde verilmiştir. Yazılmayanların Boole fonksiyonu 0 dır.

Çizelge, küçük karelerden oluşur. Her kare bir mintermi ifade eder. Değeri 1 olan mintermlere karşılık gelen karelere 1 yazılır. Diğerlerine ya 0 yazılır veya boş bırakılır. Çeşitli dizinleri saptamak ve 1 olanları birleştirmek için farklı cebirsel ifade türetmek olasıdır. Bunlar arasında en uygun olan seçilir.

Şekil 2.7 de iki, üç ve dört değişkenli çizelgeler görülüyor. Çizelge içindeki karelerin sayısı n değişken için 2^n tanedir. 2^n minterm kolay

anlaşılması için ondalık sayıyla numaralandırılmıştır. Numaralar, sadece 1 değişkenin değeri değişecek şekilde soldan sağa ve yukarıdan aşağıya büyür. Değişken adları köşegen çizginin iki yanına yazılıdır. Çizelgenin dışındaki sayılar o satır veya sütundaki değişkenlerin alacağı değerleri gösterir. Parantezlerin yanındaki harfler, o değişkenin üssüz olarak bulunduğu yerlerdir. Üslü olan değişkenler (yani tümleyenler) karelerin diğer yarısında bulunur.

Bir kare tarafından temsil edilen minterm, çizelgenin üst ve sol tarafındaki kenarlara yazılmış olan değişken değerleri yardımıyla hesaplanır. Örneğin, 3 değişkenli çizelgedeki 5 numaralı minterm, ikili olarak 101 dir. İkinci satırın sol tarafındaki 1 ile üstteki 01 in yan yana getirilmesiyle elde edilir. Bu minterm *A*, *B* ve *C* ikili değişkenlerinin aldığı değerleri gösterir. *A* ve *C* değişkenleri üssüz, *B* üslüdür (yani $AB'C$). Öte yandan dört değişkenli çizelgede 5 numaralı minterm, dört değişkeni ifade eder. Bunların ikili değerleri 0101 olup, $A'BC'D$ dir.

Bitişik Kareler

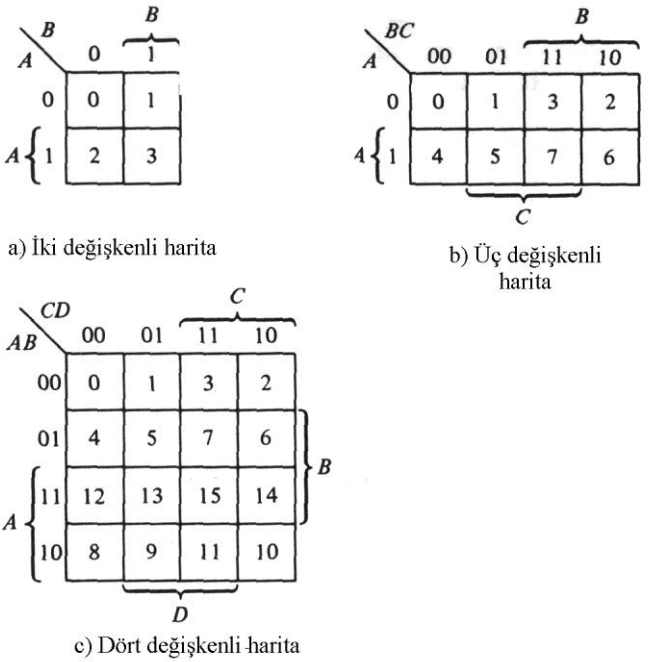
Bitişik kareler tek bir değişken hariç özdeştir. Bu tek değişken karenin birinde üslü, diğerinde üssüz olarak yer alır. Yani birinde kendisi, komşu karede tümleyeni yer alır. Bu tanıma göre bir kenardaki bir kare ile onun karşısındaki kenardaki bir kare sadece bir tek değişkenin kendisi ve tümleyeni olup, diğer değişkenler özdeştir. Dolayısıyla bunlar bitişiklik koşulunu sağlarlar. Aynı düşünce alt ve üst kenarlar için de doğrudur. Yani komşudurlar. Dolayısıyla çizelgenin dört köşesindeki dört kare bitişiktir.

Doğruluk çizelgesi tarafından gösterilen bir fonksiyon, sadeleştirme çizelgesine fonksiyonun değerlerinin 1 olduğu mintermlerin karesine 1 konarak ifade edilir. Eğer bitişik karelerde de 1 varsa, bunlar gruplandırılır. Komşu karelerin grup olması için, 2 kare veya 2 nin kuvvetleri sayısında kare olması gerekir. (Örneğin 2, 4, 8, 16, 32, ... gibi). Bu şekilde gruplanan bitişik kare grupları bir veya fazla sayıda kareyi paylaşabilir. Her grup cebirsel bir terimdir. Bu terimler VEYA ile bağlanarak sadeleşmiş fonksiyon elde edilir. Aşağıdaki örnek Boole fonksiyonunun sadeleştirilmesini gösteriyor.

İlk örnek olarak

$$F(A, B, C) = \Sigma(3, 4, 6, 7)$$

fonksiyonunu sadeleştirelim. Bu fonksiyonun 3 değişkenli çizelgesi şekil 1.8 de gösterilmiştir. 1 değeri almış 4 kare var. Çünkü bu karelerin gösterdiği minterm'ler fonksiyon için 1 değeri veriyor. 1 olan kareler (yani minterm'ler) 3, 4, 6 ve 7 numaralı olanlardır. (Şekil 1.7(b)). Üçüncü sütundaki bitişik iki kare bir gruptur. Bu sütun hem *B* hem *C* nin ortak sütunudur. Dolayısıyla bu grup *BC* terimini oluşturur. Kalan iki tane kare

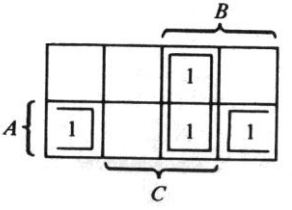


Şekil 1.7 iki üç dört değişkenli fonksiyonlar için haritalar

köşelerdedir. Bu iki kare A ve C' e aittir. Dolayısıyla bu grup AC' terimini oluşturur. İki terim VEYA lanarak

$$F = BC + AC'$$

Elde edilir.



Şekil 1.8 *F(A, B, C) = Σ(3, 4, 6, 7)* için sadeleştirme çizelgesi

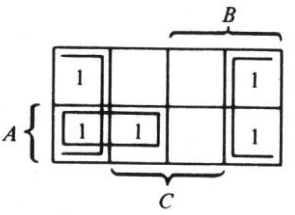
İkinci örnek aşağıdaki Boole fonksiyonunu sadeleştirir.

$$F(A, B, C) = \Sigma(0, 2, 4, 5, 6)$$

Bu fonksiyonun çizelgesi şekil 1.9 da gösterilmiştir. Birinci ve dördüncü sütundaki 4 kare bitişiktir. Bunların grubu C' terimini verir. 1 olan diğer

iki kare 4 ve 5 inci mintermlere aittir. Bu da AB' terimini verir. Basitleştirilmiş fonksiyon şöyledir:

$$F = C' + AB'$$



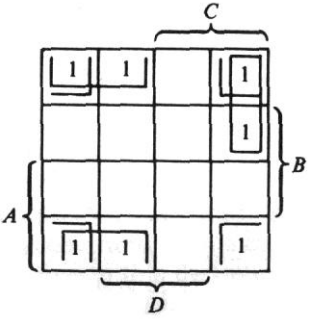
Şekil 1.9 *F(A, B, C) = Σ(0, 2, 4, 5, 6,)* için sadeleştirme çizelgesi

Üçüncü örnek dört değişkenli çizelge gerektirir.

$$F(A, B, C, D) = \Sigma(0, 1, 2, 6, 8, 9, 10)$$

Fonksiyonunun çizelgesi şekil 1.10 da verilmiştir. Fonksiyon 4 köşede 1 içerir. Bu grup $B'D'$ terimini verir. Bu mümkündür. Çünkü alt ve üst kenarlar ve sağ ve sol kenarlar bitişik varsayılmaktadır. Yukarıdaki kenarın soldaki iki karesi, aşağıdaki kenarın soldaki iki karesiyle bitişiktir. Bu $B'C'$ terimini verir. 6 numaralı kare, 2 numaralı kare ile bir grup oluşturur ve $A'CD'$ yü verir. Basitleştirilmiş fonksiyon şöyledir:

$$F = B'D' + B'C' + A'CD'$$



Şekil 1.10 F(A, B, C, D) = Σ(0, 1, 2, 6, 8, 9, 10) fonksiyonu için harita

Toplamların Çarpımının Basitleştirmesi

Önceki kısımdaki örneklerde elde edilen Boole bağıntıları çarpımların toplamı biçiminde idi. Çarpımdan kastedilen VE işlemi ve toplamdan kastedilen VEYA işlemidir. Bazen cebirsel bağıntının toplamların

çarpımı şeklinde elde edilmesi daha uygun olur. Toplamlar VEYA lanmış terimler, çarpım ise bunların VE lenmesidir. Küçük bir değişiklikle toplamların çarpımı çizelgeden elde edilebilir.

Toplamların çarpımı biçimindeki bağıntı, Boole cebrinin temel özelliklerinden elde edilebilir. Çizelgedeki 1 ler, fonksiyon için 1 üreten mintermleri gösterir. Fonksiyon için 0 üreten mintermler 1 ile gösterilmez. Boş kareler 0 (sıfır) ile gösterilir ve bunlar bitişik kareler şeklinde gruplanırsa, fonksiyonun tümleyeni olan F' elde edilir. F in değeri toplamların çarpımı biçiminde elde edilir. Bu en iyi bir örnekle gösterilebilir.

Aşağıdaki Boole fonksiyonu hem toplamların çarpımı hem de çarpımların toplamı biçiminde sadeleştirilsin.

$$F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$$

Fonksiyon için 1 olan mintermler şekil 1.11 de gösterilmiştir. F içinde bulunmayan yani 0 olan karelerde 0 ile donatılmıştır. Bunlar F in tümleyenini verir. 1 olan kareler uygun şekilde bir araya getirilerek sadeleştirilmiş fonksiyon çarpımların toplamı biçiminde elde edilir.

$$F = B'D' + B'C' + A'C'D$$

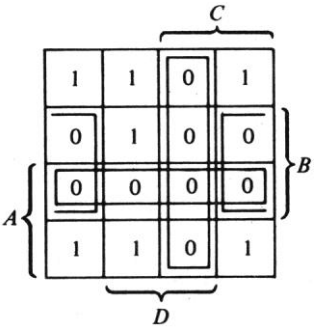
Eğer 0 olan kareler bir araya getirilirse (şekildeki gibi) sadeleştirilmiş tümleyen fonksiyon elde edilir.

$$F' = AB + CD + BD'$$

F' nün tümleyeni alınarak

$$F = (A' + B')\,(C' + D')\,(B' + D)$$

toplamların çarpımı biçiminde elde edilir.

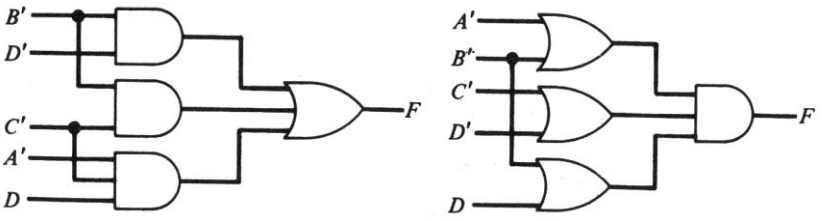


Şekil 1.11 $F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$ fonksiyonu için sadeleştirme çizelgesi

Bu iki sadeleştirilmiş fonksiyonun mantık şeması Şekil 1.12'de gösterilmiştir. Çarpımların toplamı biçimindeki bağıntı 1.12(a)'da verilmiştir. Her bir VE'li terim için VE kapılarından oluşur. VE kapılarının çıkışları tek bir VEYA kapısıyla birleştirilmiştir. Aynı fonksiyon 1.12(b)'de verilmiştir. Bu toplamların çarpımı biçiminde olup bir grup VEYA kapısıyla bunları birleştiren bir VE kapısından oluşur. Her durumda tümleyenlerin mevcut olduğu varsayılarak tersleyenler çizilmemiştir. Şekil 1.12'deki biçim her hangi bir Boole fonksiyonu için hangi yöntemle elde edilirse edilsin kullanılan bir yöntemdir. Çarpımların toplamında VE kapıları tek bir VEYA kapısına bağlanır. Toplamların çarpımında ise VEYA kapıları tek bir VE kapısına bağlanır.

VE-DEĞİL Uygulaması, VEYA-DEĞİL Uygulaması

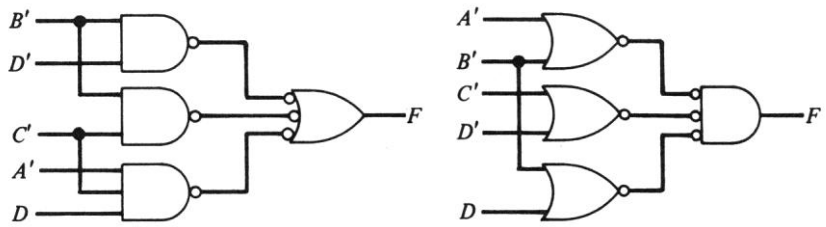
Çarpımların toplamı biçiminde bir bağıntı VE-DEĞİL kapılarıyla da kurulabilir. Şekil 1.13(a). Bu şekildeki VE-DEĞİL kapıları Şekil 1.5(b)' deki grafik sembol ile çizilmiştir. Şekilde, her iki ucunda küçük daireler bulunan 3 tane çizgi vardır. Aynı çizgide 2 tane küçük daire çift tümlemeyi gösterir. Yani $(x')' = x$ olur. İki daire kaldırılabilir ve şekil 1.12(a) dakinin eşdeğeri olur. Benzer şekilde toplamların çarpımı VEYA-DEĞİL kapılarıyla çizilebilir. Şekil 1.13(b). VE-DEĞİL kapısı, Şekil 1.4(b)'deki grafik sembol ile çizilmiştir. Benzer şekilde her iki uçtaki daireler kaldırılırsa elde edilen Şekil 1.12(b)'dekinin aynı olur.



a) Toplam sonuç:
F=B'D'+B'C'+A'C'D'

b) Toplam sonuç:
F=(A'+B')(C'+D')(B'+D')

Şekil 1.12 VE ve VEYA kapıları için mantık şeması



a) VE-DEĞİL Kapılarıyla

b) VEYA-DEĞİL Kapılarıyla

Şekil 1.13 VE-DEĞİL ve VEYA kapıları için mantık şeması

Dikkate Almama

Dikkate Almama

Çizelgedeki 1 ve 0 lar, minterm'leri temsil eder. Minterm'ler fonksiyonu 1 veya 0 yapar. Bazen minterm'i oluşturan fonksiyonun 1 veya 0 olması fark etmez. Yani öyle de olsa olur, tersi de olsa olur. Bu durumda minterm'i oluşturan fonksiyonun değerini dikkate alma deriz. Dolayısıyla minterm'i 0 veya 1 yapan koşullar oluşunca buna dikkate almama koşulları deriz ve çizelge de bunu *x* ile gösteririz. Koşullar cebirsel bağıntının daha fazla sadeleştirilebilmesinde kullanılabilir.

Çizelgede bitişik kareleri seçerken, *x* lerin değeri 0 veya 1 olarak seçilebilir. Hangi değer daha fazla sadeleştirme sağlıyorsa o seçilir. Bunun ötesinde eğer *x* in değeri sadeleştirmeye katkıda bulunmuyorsa bu değer kullanılmayabilir. Örnek olarak, aşağıdaki Boole fonksiyonlarını dikkate almama minterm'leri ile beraber ele alalım.

$$F(A, B, C) = \Sigma(0, 2, 6)$$

$$d(A, B, C) = \Sigma(1, 3, 5)$$

F içindeki minterm'ler fonksiyon için 1 üretir. *d* içinde yer alan minterm'ler fonksiyon için 0 veya 1 üretir. Geri kalan minterm'ler 4 ve 7 fonksiyon için 0 üretir. Çizelge şekil 1.14 te gösterilmiştir. *F*'nin minterm'leri 1 ile, *d*'ninkiler *x* ile ve kalanlar 0 ile işaretlenmiştir. 1 ler ve *x* ler öyle bir araya getirilebilir ki en fazla sayıda bitişik kareler oluşsun. Bütün *x* leri almak zorunlu değildir. Fakat minterm'lerini ve birinci sıradaki 1 leri alarak A' terimini elde ederiz. Minterm 6 daki 1, minterm 2 deki 1 ile bir araya getirilerek BC' terimi elde edilir. Sadeleştirilmiş bağıntı

$$F = A' + BC'$$

olur. 5 numaralı minterm dikkate almama olup bağıntının daha fazla sadeleşmesine katkıda bulunmadığı için dahil edilmemiştir. Eğer 1 ve 3 teki dikkate almama minterm'leri dahil edilmeseydi *F* bağıntısı

$$F = A'C' + BC'$$

olacaktı. Bu durumda, daha önce elde edilenden fazla olarak bir VE ve bir TERSLEYEN kapı kullanılacaktır.

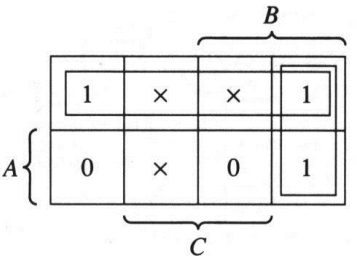
Fonksiyon 1 ve 0 değerleriyle beraber *x* değerleri de çizelgeye yazılınca

$$F = A' + BC'$$

olarak bulunur. Bu ifade

$$F(A, B, C) = \Sigma(0, 1, 2, 3, 6)$$

Boole fonksiyonunu ifade eder. Bu fonksiyonda 0, 2 ve 6 minterm'lerinin yanı sıra 1 ve 3 dikkate almama minterm'leri vardır. 5 nolu minterm fonksiyona dahil edilmemiştir. Aslında 1, 3 ve 5 nolu minterm'ler dikkate almama koşulları olarak belirlenmişse de burada 1 ve 3 nolu minterm'ler 1 olarak ve 5 nolu minterm 0 olarak alındı. Çünkü bu en sade Boole bağıntısını veriyordu.



Şekil 1.14 Dikkate almama örneği

1.5 Birleşik Devreler

Blok Şema

Bir birleşik devrede mantık kapıları vardır ve bunlar birbirine bağlanmıştır. Ayrıca giriş ve çıkışlar vardır. Herhangi bir anda çıkışların ikili değerleri, girişlerin ikili değerlerinin bir fonksiyonudur. Bir birleşik devrenin blok şeması Şekil 1.15'te gösterilmiştir. *n* tane ikili giriş değişkeni dış bir kaynaktan gelir. *m* tane ikili çıkış değişkeni dış bir hedefe gider. Bunların arasında mantık kapılarının bağlantıları vardır. Birleşik bir devre verilen bir ikili giriş bilgisini istenen ikili biçime çevirerek çıkışa yollar. Birleşik devreler, sayısal bilgisayarlarda ikili denetim kararları vermek ve bilgi işlem için gereken sayısal bileşenleri üretmek için kullanılır.

Bir birleşik devre *n* tane giriş değişkeni ve *m* tane çıkış değişkeni arasındaki ikili bağları gösteren doğruluk çizelgesi ile tanımlanabilir.



Şekil 1.15 Birleşik devrenin blok şeması

Doğruluk çizelgesinde her bir 2^n elemanlı giriş kümesine karşılık gelen ikili çıkış değerleri kümesi bulunur. Bir birleşik devre, her biri bir çıkış değişkenine karşılık gelen *m* tane Boole fonksiyonu ile de tanımlanabilir. Her bir çıkış fonksiyonu, *n* tane giriş değişkeninden oluşan terimlerden meydana gelir.

Analiz

Bir birleşik devrenin analizi verilen bir mantık devre şeması ile başlar ve bir Boole fonksiyonları kümesi veya doğruluk çizelgesi ile sona erer. Eğer sayısal devrenin yanında yaptığı işe ait yazılı veya sözlü açıklama varsa, Boole fonksiyonları veya doğruluk çizelgesi ispat için yeterlidir. Devre halen araştırılıyor ise, yaptığı işi yani fonksiyonunu, türetilen Boole fonksiyonları veya doğruluk çizelgesi ile açıklamak gerekir. Böyle bir araştırmanın başarısı, araştırmayı yapan kişinin deneyimi ve sayısal devrelerle yakınlığı ile artar. Bilgi-işlem görevinde bir doğruluk çizelgesi veya Boole fonksiyonları kümesi kurabilmek deneyimle kazanılan bir sanattır.

Tasarım

Birleşik devrenin tasarımı, sorunun özetlenmesiyle başlar, mantık devre şeması ile son bulur. Yapılan işlem şu adımlardan oluşur.

1. Soru ortaya konur.
2. Giriş ve çıkış değişkenlerine harf ile ad verilir.
3. Girişler ve çıkışlar arasındaki ilişkiyi gösteren doğruluk çizelgesi hazırlanır.
4. Her çıkış için sadeleştirilmiş Boole fonksiyonları türetilir.
5. Mantık şeması çizilir.

Birleşik devrelerin tasarım sürecini göstermek için basit aritmetik devrelerden oluşan iki örnek görelim. Bu basit devreler, çok daha karışık aritmetik devrelerin kurulumunda temel yapı taşı görevi yaparlar.

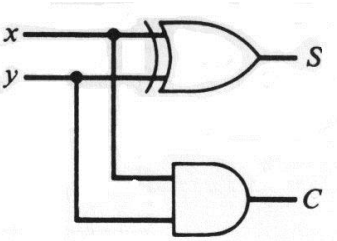
Yarı-Toplayıcı

En temel sayısal aritmetik devre, iki tane ikili rakamı toplayan devredir. İki bitin aritmetik toplamını yapan birleşik devreye yarı toplayıcı adı verilir. Üç bitin (2 önemli bit ve bir önceki elde) toplamasını yapan devreye tam-toplayıcı denir. İlk devrenin adı bir tam toplayıcıyı kurmak için iki yarı toplayıcı gerekmesinden gelir.

Bir yarı toplayıcının giriş değişkenleri toplayan ve toplanan bitlerdir. Çıkış değişkenleri toplam ve elde dir. İki çıkış değişkeni vermek gerekir. Çünkü ikili sistemde $1 + 1 = 10$ dır. Yani iki rakam vardır. İki giriş değişkenine *x* ve *y* ve toplama *S*, elde ye de *C* adını veriyoruz. Yarı toplayıcı için doğruluk çizelgesi şekil 1.16(a) da gösterilmiştir. Her iki giriş 1 olana kadar *C* çıkışı 0 dır. *S* çıkışı toplamın en önemsiz bitini gösterir. Her iki çıkış için Boole fonksiyonları doğruluk çizelgesinden doğrudan elde edilebilir.

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

a) Doğruluk çizelgesi



b) Mantık şeması

Şekil 1.16 Yarı toplayıcı devresi

$$S = x'y + xy' = x \oplus y$$

$$C = xy$$

Mantık şeması Şekil 1.16(b)'e gösterilmiştir. Şemada bir ÖZEL-VEYA ve bir VE kapısı var.

Tam-Toplayıcı

Tam toplayıcı, üç giriş bitinin aritmetik toplamını bulan bir birleşik devredir. Üç girişi, iki çıkışı vardır. Giriş değişkenlerinden ikisi x ve y ile gösterilir ve toplanacak iki önemli biti gösterir. Üçüncü giriş z, bir önceki önemsiz bitin elde sidir. İki çıkış gereklidir. Çünkü 3 tane ikili rakamın toplamı 0 dan 3 e kadar değer alabilir. İkili düzende 2 ve 3 iki rakamlık yer ister. İki çıkış S (toplam) ve C (elde) ile gösterilir. İkili değişken S, toplamın önemsiz bitini gösterir. İkili değişken C, çıkış elde sidir. Tam toplayıcının doğruluk çizelgesi çizelge 1.2 de gösterilmiştir. Giriş değişkenlerinin altındaki 8 satır, ikili değişkenlerin alabileceği tüm mümkün değerleri gösterir. Çıkış değişkenlerinin değerleri, giriş bitlerinin aritmetik toplamından bulunur. Tüm giriş bitleri 0 olunca çıkış 0 dır. S çıkışı ya yalnız bir giriş 1 ise veya tüm üçü 1 ise 1 olur. C çıkışı, iki yada üç giriş 1 ise 1 olur.

Şekil 1.17'deki çizelge iki çıkış değişkeni için cebirsel bağıntıları bulmakta kullanıldı. S ve C sütunlarındaki 1 ler, doğruluk çizelgesindeki minterm'lerden doğrudan bulundu. S çıkışı içindeki 1 ler bitişik kareler ile guruplanamadı. Fakat tek sayıda girişlerin çıkışı 1 olduğundan S bir "tek" fonksiyondur. Dolayısıyla ÖZEL-VEYA bağıntısı vardır. (Bölüm 1.2 sonundaki tartışmaya bakın). C nin kareleri farklı bir çok biçimde guruplandırılabilir. Mümkün bir yol

$$C = xy + (x'y + xy')z$$

$$S = x \oplus y \oplus z$$

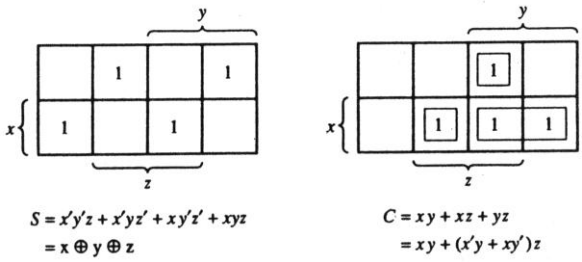
Çizelge 1.2 tam toplayıcı için doğruluk çizelgesi

GİRİŞLER			ÇIKIŞLAR	
x	*y*	*z*	*C*	*S*
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

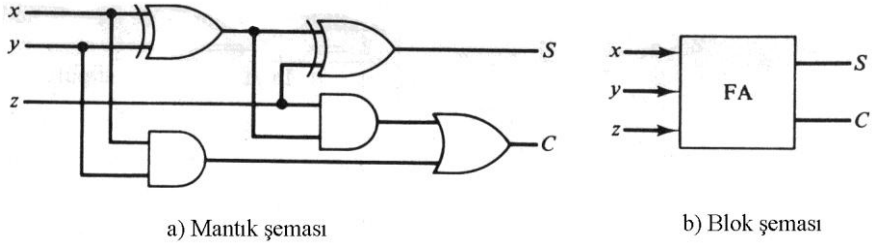
olabilir. $x'\,y + x\,y' = x \oplus y$ olduğu düşünülerek

$$C = xy + (x \oplus y)\,z$$

Boole bağıntıları elde edilir. Tam toplayıcı için mantık şeması şekil 1.18 de çizilmiştir. Tam toplayıcı, iki yarı toplayıcı ve bunları bağlayan bir VEYA kapısından oluşur. İlerki bölümlerde tam toplayıcı, Şekil 1.18(b)' de gösterilen blok şema şeklinde kullanılacaktır.



Şekil 1.17 Tam toplayıcı fonksiyonları



Şekil 1.18 Tam toplayıcı devresi

1.6 Yaz-Bozlar

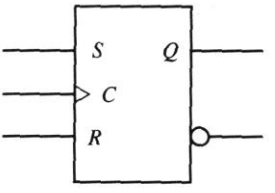
Şu ana kadar, sayısal devrelerin birleşik devreler olduğu varsayıldı. Yani herhangi bir andaki çıkışlar bütünüyle o andaki girişlere bağlıdır. Her ne kadar her sayısal sistemde bir birleşik devre bulunursa da, gerçekte kullanılan çoğu sistem saklama öğeleri de içerir. Bu sistemlere sıralı devreler denir. En yaygın sıralı devre eş zamanlı devrelerdir. Eş zamanlı sıralı devreler, saklama öğelerine ancak zamanın ayrık anlarında etkileyen sinyaller yollayarak kullanırlar. Eş zamanlılığa, bir zaman aleti (yani saat) kullanarak erişilebilir. Zaman aleti (saat) bir darbe üretecidir ve darbe dizileri üretir. Saat darbeleri tüm sisteme yayılarak saklama öğelerinin ancak eşzamanlılık darbesinin gelişi ile işlemesi sağlanır. Saatli eş zamanlı sıralı devreler, en çok kullanılan devrelerdir. Çok nadiren dengesizlik olayı olur ve bunların zamanlaması kolayca bağımsız ayrık adımlara ayrılabilirler. Her bir adım ayrı bir zaman olarak düşünülebilir.

Saatli Eşzamanlı Devre

Saatli eşzamanlı devrelerde kullanılan saklama öğelerine yaz-boz denir. Bir yaz-boz hücresi, bir bit bilgi saklayabilir. İki çıkışı vardır. Biri öğenin içindeki değeri, diğeri bunun tümleyenini verir. Bir yaz-boz içindeki ikili değeri, durumu değiştirmesi için bir saat darbesi gelene kadar korur. Bir çok farklı yaz-boz mevcuttur. Bunların bazılarının giriş sayıları farklıdır. Bazılarında ise farklı girişlere göre durum değişir. Çok kullanılan bazı yaz-bozlar aşağıda gösterilmiştir.

SR Tipi Yaz-boz

SR yaz-bozunun grafik sembolü Şekil 1.19(a)'da gösterilmektedir. Burada 3 giriş ve iki çıkış vardır. Girişler *S* (kurma) *R* (silme) ve *C* (saat) olarak isimlendirilmektedir. Çıkış *Q* dur diğer çıkış bu çıkışın tümleyeni olan *Q'* dir. Saat girişi küçük bir üçgenle belirlenmiştir. Üçgen sembolüne *devingen gösterge* denir ve devrenin 0 dan 1 e geçiş halinde çalışacağını belirtir. yaz-boz çıkışı *Q* dur. *Q'* çıkışındaki daire *kutupsal gösterge* olup çıkışın tümleyeni olduğunu gösterir.



a) Grafik sembolü

S	*R*	*Q* (*t* + 1)	
0	0	*Q* (*t*)	Değişiklik yok
0	1	0	Sıfır yapma
1	0	1	Bir yapma
1	1	?	Tanımsız

b) Karakteristik tablosu

Şekil 1.19 *SR* tipi yaz-boz

SR yaz-bozunun çalışması saat vuruşu 0 da kaldığı sürece olmaz ancak saat vuruşunun 0 dan 1 e geçişi esnasında olur. Saat vuruşu 0'dan 1'e geçerken yaz-boz çıkışları girişteki *S* ve *R* değerlerine göre değişmektedir. Bu değişiklikler karakteristik çizelgesinde belirtilmektedir. Eğer $S = 1$ ve $R = 0$ ise ve saat 0 dan 1 e geçiyorsa *Q* çıkışı 1 olur. Eğer girişler $S = 0$ ve $R = 1$ ise ve saat 0 dan 1 e geçiyorsa *Q* çıkışı 0 ile silinir. Girişlerin her ikisi de 1 ise çıkışta tahmin edilmeyen bir durum oluşur.

SR yaz-boz karakteristik çizelgesi şekil 1.19(b) de gösterilmektedir. *S* ve *R* girişlerinin alabileceği bütün ikili değerler çizelgenin sol tarafında sağ tarafında ise bu girişlere göre yaz-boz çıkışının bir sonraki saat vuruşunda ne olacağı belirtilmektedir. $Q(t)$ ikili değeri saat işaretinden önceki çıkışı, $Q(t + 1)$ ikili değeri de saat işaretinden sonraki çıkışı göstermektedir. Saat = 1 iken $S = 1$ ve $R = 0$ ise kurma durumu elde edilir. Saat = 1 iken $S = 0$ ve $R = 1$ iken silme durumu elde edilir. Saat = 1 $S = 0$ ve $R = 0$ yaz-bozun durumunu değiştirmez. ($Q(t + 1) = Q(t)$), Saat = 1 iken *S* ve *R* ikisi birden 1 olamaz, çünkü yaz-bozun bundan sonraki durumu bilinemez. Bu kararsızlık durumu yüzünden *SR* tipi yaz-bozlar pratikte nadir kullanılırlar.

D Tipi Yaz-boz

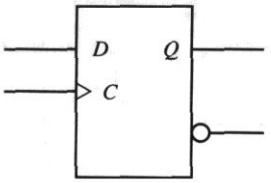
RS in biraz değişiğidir. *S* ve *R* girişi arasına bir TERSLEYEN konur. Girişe *D(veri)* denir. Bu tek bir giriştir. Saatin 0 dan 1 e geçişi sırasında *D* girişindeki değeri alır. Eğer *D* girişi 1 ise çıkış 1, *D* girişi 0 ise çıkış 0 olur.

D yaz-bozunun grafik sembolü ve karakteristik tablosu Şekil 1.20'de gösterilmektedir. Karakteristik tablodan anlaşılabileceği gibi gelecek durum $Q(t + 1)$, *D* girişi tarafından belirlenir. Aşağıdaki karakteristik denklem ikisi arasındaki ilişkiyi ifade etmektedir:

$$Q(t + 1) = D$$

Bu demektir ki saat sinyalinin 0 dan 1 e her geçişinde yaz-bozun *Q* çıkışı *D* girişinin değeri alır.

Dikkat edilecek olursa *D* yaz-bozunun durumunun değişmeden kalmasını sağlayacak herhangi bir giriş koşulu bulunmamaktadır. *D* yaz-bozunun sadece 1 girişe sahip olma avantajı olmasına rağmen (*C* hariç) karakteristik tablosunda "değişiklik yok" koşulunun bulunmaması ($Q(t + 1) = Q(t)$) onun için bir dezavantajdır. "Değişiklik yok" koşulu ya saat sinyalini iptal ederek yada çıkışı vererek elde edilebilir. Böylelikle saat vuruşları yaz-bozun durumunu değiştirmez.



a) Grafik sembolü

D	Q (t + 1)	
0	0	Sıfır yapma
1	1	Bir yapma

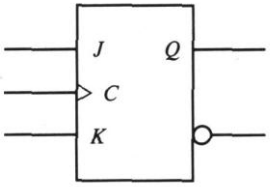
b) Karakteristik tablosu

Şekil 1.20 *D* tipi yaz-boz

JK Tipi Yaz-boz

JK türü yaz-boz, *RS* türünün belirsiz durumunun düzeltilmişidir. *J* ve *K* girişleri, *S* ve *R* girişleri gibi yaz-bozu 1 ve 0 yapacak şekilde davranırlar. *J* ve *K* girişlerinin ikisi de 1 olduğunda saat geçişi anında yaz-bozun çıkışları tümlenir.

JK tipi yaz-bozun grafik sembolü ve karakteristik tablosu şekil 1.21 de gösterilmektedir. *J* girişi *SR* tip yaz-bozun *S* girişine, *K* girişi de *R* girişine denktir. Sadece $J = K = 1$ halinde belirsizlik yerine $Q(t + 1) = Q'(t)$ olur. Sayaç ve zamanlayıcıların tasarımında kullanılır.



a) Grafik sembolü

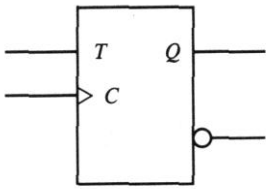
J	K	Q (t + 1)	
0	0	Q (t)	Değişiklik yok
0	1	0	Sıfır yapma
1	0	1	Bir yapma
1	1	Q′ (t)	Tümleyen alma

b) Karakteristik tablosu

Şekil 1.21 *JK* tipi yaz-boz

T Tipi Yaz-boz

Başka bir yaz-boz türü *T*(tümleme) tipi yaz-bozdur. Bu yaz-boz şekil 1.22 de gösterildiği gibi *JK* girişleri tek bir giriş olacak şekilde birbirine bağlanarak elde edilir. Bu tek girişe *T* adı verilir. *T* yaz-bozunun sadece iki durumu vardır. $T = 0$ ($J = K = 0$) olduğunda saat geçişi yaz-bozun



a) Grafik sembolü

T	Q (t + 1)	
0	Q (t)	Değişiklik yok
1	Q′ (t)	Tümleyen alma

b) Karakteristik tablosu

Şekil 1.22 *T* tipi yaz-boz

durumunu değiştirmez. $T = 1$ $(J = K = 1)$ olduğunda saat geçişi yaz-bozun durumunun tümlenmesine yol açar. Bu çalışma şekli,

$$Q(t + 1) = Q(t) \oplus T$$

biçimindeki karakteristik denklem ile gösterilebilir.

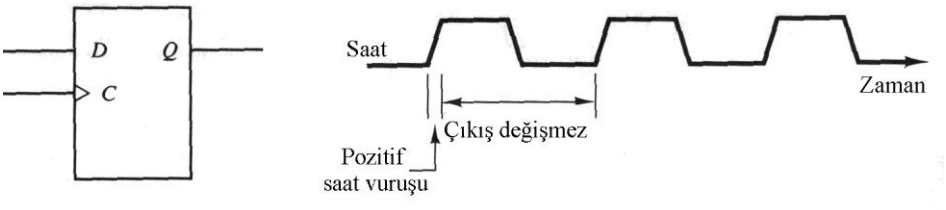
Kenar Tetiklemeli Yaz-bozlar

Kenar tetiklemeli yaz-bozlar durumunu saatin değişmesi sırasında değiştiren ve çok yaygın kullanılan bir diğer yaz-boz çeşididir. Bu tür yaz-bozda çıkış saatin belli bir düzeyinde olur. Saat vuruşu bu eşik değerini geçince girişler kilitlenir. Böylece yaz-boz girişlerde bundan sonra olacak değişikliklere duyarlı olmaz. Bu durum saat = 0 olup yeni vuruş başlayıncaya kadar devam eder. Bazı kenar tetiklemeli yaz-bozlar değişme işlemini saatin yükselen kenarında (pozitif-uç geçişi), bazıları ise inen kenarında (negatif-uç geçişi) yaparlar.

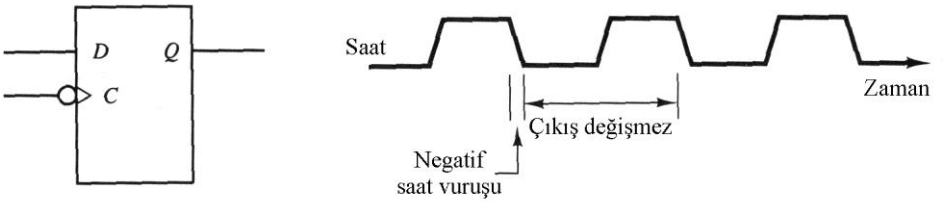
Saat Vuruşları

Şekil 1.23(a) pozitif kenar tetiklemeli *D* tipi yaz-bozun saat vuruş işaretini göstermektedir. Saat bir pozitif geçiş yaptığında yaz-bozun *D* giriş değeri *Q* çıkışına gider. Saatin 1 veya 0 seviyesinde yada 1 den 0 a geçişi esnasında çıkış değer değiştirmez. Etkin pozitif saat geçişi, geçişten önce *D* girişinin sabit bir değerde kalmasının gerektiği ve *kurma zamanı* olarak adlandırılan bir minimum zaman ile pozitif geçişten sonra *D* girişinin değişmemesinin gerektiği ve *tutma zamanı* olarak adlandırılan bir zamanı içerir. Etkin saat geçişi, saat darbesinin toplam periyodunun genellikle çok küçük bir kısmıdır.

Şekil 1.23(b)'de negatif kenar tetiklemeli bir *D* yaz-bozunun sembolü ve zamanlama şeması verilmiştir. Grafik sembol *C* girişindeki dinamik göstericinin önünde küçük bir daire içermektedir.Bu durum negatif kenar tetiklemeli çalışmayı anlatır. Yani bu yaz-boz saat işaretinin 1 seviyesinden 0 seviyesine geçişinde çalışmaktadır.



a) Pozitif-uç-tetiklemeli D tipi yaz-boz



b) Negatif-uç-tetiklemeli D tipi yaz-boz

Şekil 1.23 kenar tetiklemeli yaz-bozlar

Ana-Uydu Yaz-Bozu

Bazı sistemlerde kullanılan yaz-bozların başka bir tipi de ana-uydu (efendi-köle) yaz-bozdur. Bu tür devre yapısı iki ayrı yaz-boz dan oluşur. İlki ana saatin yükselen kenarında ikincisi uydu saatin düşen kenarında çalışmaktadır. Sonuç olarak saat işaretinin 1 den 0 a değişmesi boyunca çıkış değişmemektedir. Eğilim bu yaz-bozların kullanımından uzaklaşıp kenar tetiklemeli yaz-bozlara yönelmek şeklindedir.

IC paketleri içinde olan yaz-bozlarda bazen özel giriş uçları bulunabilir. Bu özel giriş uçları, yaz-bozların saatten bağımsız olarak değerlerinin 1 yada 0 yapılması için kullanılmaktadır. Bu uçlara "ön yükleme" veya "silme" uçları denir. Bu uçlar saat vuruşu olmaksızın yaz-bozu etkilerler. Böylece saatten önce yaz-bozların başlangıç durumuna getirilmesi sağlanmış olur.

Yükseltgeme Çizelgeleri

Yaz-bozların doğruluk çizelgeleri şu andaki durumları belli olduğunda bir sonraki zaman için durumlarını verir. Tasarım aşamasında genellikle şu anki durumla gelecek zamandaki durumun ne olması gerektiği bilinir. Bize gereken bu geçiş için yaz-boz un giriş koşullarını bilmemizdir. Bu durumda öyle bir çizelgeye ihtiyacımız olur ki verilen bir durum değişmesi için hangi giriş koşullarının gerektiğini bize versin. Bunlara yaz-boz yükseltgeme çizelgeleri denir.

Çizelge 1.3 dört tip yaz-boz için bu çizelgeleri veriyor. Her çizelgede $Q(t)$ ve $Q(t + 1)$ zamanlarında alabilecekleri durumlarla bunlara karşılık gelen giriş koşulları bulunuyor. $Q(t)$ ile $Q(t + 1)$ geçişlerinde 4 olasılık vardır. Bunlar doğruluk çizelgelerinden elde edilir. × ler dikkate almama durumlarını gösterir. Burada yaz-bozun girişinin 0 yada 1 olması önemli değildir.

Çizelge 1.3 Dört yaz-boz için yükseltgeme çizelgeleri

SR yaz-bozu

$Q(t)$	$Q(t+1)$	S	R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

D yaz-bozu

$Q(t)$	$Q(t+1)$	D
0	0	0
0	1	1
1	0	0
1	1	1

JK yaz-bozu

$Q(t)$	$Q(t+1)$	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

T yaz-bozu

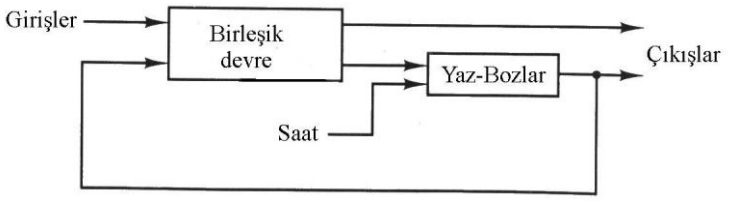
$Q(t)$	$Q(t+1)$	T
0	0	0
0	1	1
1	0	1
1	1	0

Buradaki dikkate almama durumlarının varolması, aynı sonuca iki farklı yoldan erişebileceğimizi göstermesi bakımından önemlidir. Örneğin *JK* yaz-bozunda saat vuruşunun gelmesi sırasında J ve K yı 0 yaparak değişmeme durumunu elde edebiliriz. Veya aynı sonucu $J = 0$, $K = 1$ ile de elde edebiliriz. Yani durumunu koru ile $0 \rightarrow 0$ geçişi elde edebiliriz. Veya silme ile $0 \rightarrow 0$ geçişi elde edebiliriz. O halde K nın 0 veya 1 olması durumu bozmaz (dikkate almama durumu). Böylece tasarımcı duruma göre iki durumdan uygun geleni kullanır.

1.7 Sıralı Devreler

Sıralı bir devre kapıların ve yaz-bozların oluşturduğu bir devredir. Kapılar bir birleşik devre oluşturur. Saatli bir sıralı devrenin blok şeması şekil 1.24 te gösterilmektedir. Bu devrede bir birleşik devre ve bir çok sayıda saatli yaz-boz bulunmaktadır. Genel olarak istenen tip ve sayıda yaz-boz devreye eklenebilir. Birleşik devre kısmı dışarıdan gelen ikili işaretleri ve yaz-bozlar dan çıkan ikili işaretleri alır ve benzer şekilde dış

devrelere veya yaz-bozlara ikili işaretleri gönderir. Birleşik devredeki kapılar dışarıya gönderilecek ikili işaretin değeri ile yaz-bozlarda saklanacak ikili değerleri belirler. Bu iş her zaman saat vuruşundan sonra gerçekleşir. yaz-bozların çıkışları birleşik devrenin girişlerine uygulanarak devrenin davranışlarını belirler. Bu işlerin açıkça ortaya koyduğu şey sıralı bir devrenin çıktısının, dış girdilerle yaz-bozların o anki durumlarının sonucu olduğudur. yaz-bozların bir sonraki durumları ise bulundukları o anki durumları ile dış girdilerin sonucudur. O halde sıralı bir devre, dış girdiler, dışarı çıktılar ve içeride yaz-bozların ikili durumları ile bellidir.



Şekil 1.24 saatli sıralı devrenin blok şeması

Yaz-boz Giriş Fonksiyonları

Giriş Fonksiyonu

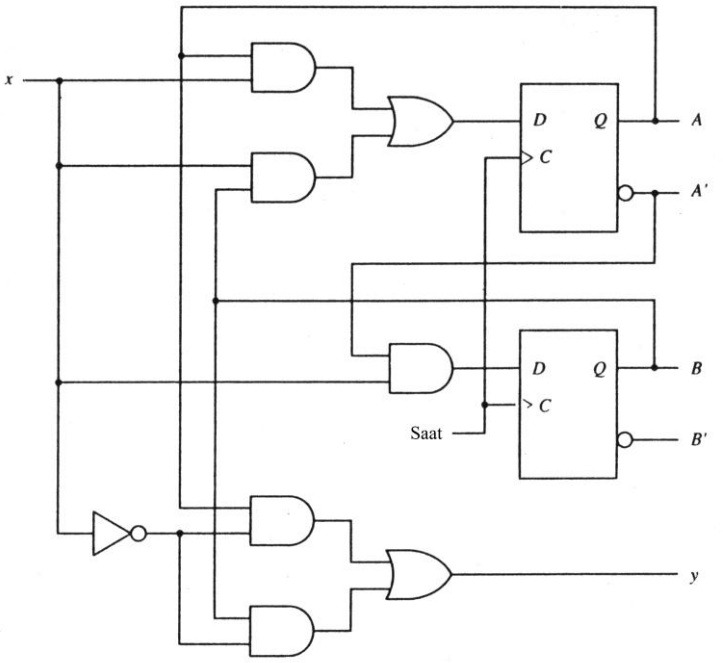
Sıralı bir devre örneği şekil 1.25 te verilmiştir. Bir *x* girişi, bir *y* çıkışı ve iki tane saatli *D* yaz-bozu vardır. VE kapıları, VEYA kapıları ve TERSLEYEN birleşik devre kısmını oluşturur. Kapılar arasındaki bağlantılar bir gurup Boole fonksiyonu ile belirlenebilir. Yaz-bozların girişlerini oluşturan devreyi belirleyen Boole fonksiyonunu *yaz-boz giriş fonksiyonları* denir. Giriş fonksiyonunu belirlemek için iki harfli değişkenler kullanılır. Birinci harf yaz-bozun tipini belirler ikinci harf yaz-bozun adıdır. Şekil 1.25 te D_A *ve* D_B olmak üzere iki giriş fonksiyonu vardır. *D* yaz-bozun girişini, *A* ve *B* ise yaz-boz un adını gösterir.

Giriş fonksiyonları yaz-boz giriş değişkenlerinin Boole fonksiyonlarıdır. Devrenin incelenmesiyle elde edilir. Birinci giriş denklemi *A* ve *B* çıkışları ile *x* girişinin VE lenmesinden oluşan

$$D_A = Ax + Bx$$

şeklindedir. Burada *A* ve *B* iki yaz bozun çıkışını *x* te harici girişi göstermektedir. İkinci giriş *D* yaz-bozunun *B* girişine tek bir VE kapısından gelmektedir:

$$D_B = A'x$$



Şekil 1.25 Sıralı devreye bir örnek

şeklinde yazılır. Burada A ve B iki yaz-bozun çıkışını ve x te girişi gösterir.

Sıralı devre yaz-bozların durumu ile harici girişin fonksiyonu olan tek bir çıkışa sahiptir. Bu çıkış cebirsel olarak

$$y = Ax' + Bx'$$

ile verilir.

Bu örnekten görüldüğü gibi yaz-bozlar birleşik devre için bir giriş oluşturmaktadır. Altta tanımlanan değişkenler birleşik devrenin çıkışının ikili değişken ismidir. Bu çıkış daima bir yaz-boz girişi ile ilişkilidir.

Durum Çizelgesi

Sıralı bir devrenin davranışı, girişlerden, yaz-bozların çıktıları ve bulundukları durumlardan elde edilir. Çıktılar ve bir sonraki durum şimdiki durumun ve girdilerin bir sonucudur. Sıralı bir devre durum çizelgesi ile bellidir. Saatli sıralı devrelerde bir sonraki duruma geçiş saat vuruş işaretinin varlığı ile veya gelmesi ile meydana çıkar.

Şimdiki Durum, Sonraki Durum

Şekil 1.25 teki devrenin durum çizelgesi 1.4 te verilmiştir. Bu çizelge *şimdiki durum, giriş, sonraki durum,* çıkış olmak üzere dört bölüm içermektedir. Şimdiki durum kısmı *A* ve *B* yaz-bozlarının herhangi bir *t* zamandaki durumunu göstermektedir. *x* ile isimlendirilen harici giriş her zaman şimdiki durumu göstermektedir. Sonraki durum kısmındaki yaz-bozlar bir sonraki *t* + 1 zamanındaki durumu göstermektedir. Çıkış kısmı *y* şimdiki durum ile giriş koşullarına bağlı olarak oluşmaktadır.

Çizelge 1.4 Şekil 1.24 için durum çizelgesi

Şimdiki Durum		Giriş	Sonraki Durum		Çıkış
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

Şimdiki durum ve girişin tüm ikili mümkün durumları yazılarak durum çizelgesi türetilir. Bu durumda 000 dan 111 e kadar sekiz durum elde ederiz. Sonraki durum değerleri giriş fonksiyonlarına bağlı olarak mantıksal şemadan elde edilir. A yaz-bozu için giriş denklemi

$$D_A = Ax + Bx$$

dir. Her bir yaz-boz için sonraki durum şimdiki durumda *D* yaz-bozunun girişine gelen değerlere bağlı olarak elde edilir. Saat işaretinin gelmesi ile şimdiki durumdan sonraki duruma geçilir. Bu yüzden sonraki durumda *A* = 1 olması için şimdiki durumda giriş koşulları $Ax = 1$ veya $Bx = 1$ olması gerekir. Bu da D_A yı 1 yapar.Bu durum durum çizelgesinde *A* nın sonraki durumunda üç tane 1 ile gösterilmektedir. Aynı şekilde *B* yaz-bozu için giriş denklemi

$$D_B = A'x$$

B girişinin de bir sonraki durumda 1 olması için şimdiki durumda $A = 0$ ve $x = 1$ olmalıdır. Çıkış sütunu

$$y = Ax' + Bx'$$

çıkış denkleminden elde edilir.

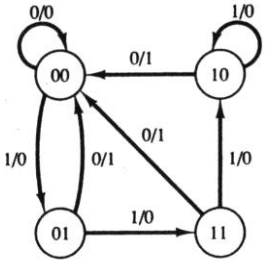
Durum Çizelgesi

Herhangi bir sıralı devrenin durum çizelgesini elde etmek için: Genel olarak *m* tane yaz-bozu olan ve *n* girişi bulunan bir sıralı devre varsa ve bunun *p* tane çıkışı varsa şimdiki durum için *m* tane sütun ve girişler için *n* sütun içeren bir çizelge olmalıdır. Gelecek an için *m* sütün, çıkışlar için *p* sütun olmalıdır. Şimdiki hal ve girişlerin 2 tane ikili kombinezonu alınır. Gelecek an ve çıktılar doğrudan devreden hesaplanır.

Durum Şeması

Durum Şeması

Durum çizelgesindeki bilgiler durum şeması denen bir şekille gösterilebilir. Bu şemada her durum bir daire ile gösterilir. Durumlar arası geçişler daireleri birleştiren doğrularla belirtilir. Şekil 1.25 in şeması 1.26 da gösterilmektedir. Her dairenin içindeki ikili sayılar yaz-bozların durumunu göstermektedir. Durumları bağlayan doğrular slaş la ayrılmış iki adet ikili sayı ile gösterilir. Durum geçişine neden olan giriş değeri önce yazılır. bulunulan durumdaki çıktı ise ikinci değerdir. Örneğin 00 durumundan 01 durumuna giden doğru 1/0 olarak işaretlenmiştir. Bunu anlamı sıralı devre şu anda 00 durumundadır. Giriş1 ve çıkış 0 dır. Saat vuruşundan sonra devre 01 durumuna geçer. Eğer giriş 0 a değişirse çıkış 1 olur, fakat eğer giriş 1 de kalırsa çıkış 0 olur. Durum şemasından bu bilgi 01 durumunu gösteren daireden çıkan iki çizgiden elde edilir. Bir daireyi kendisine bağlayan çizgi durumda değişme olmadığını gösterir. Bu şema durum çizelgesinden elde edilir.



Şekil 1.26 Sıralı devrenin durum şeması

Durum şeması ile Durum çizelgesi arasında hiçbir fark yoktur. Sadece şemanın anlaşılması daha kolaydır. Ayrıca çoğunlukla sıralı bir devrenin ilk çizimi için kullanılır. Örneğin, Şekil 1.26 daki devrenin durum şemasından giriş 0 çıkış 1 olduğu sürece durumun 00 da kaldığı, giriş 1 çıkış 0 da devrenin değer değiştirerek başlangıç 00 durumuna gittiği görülmektedir.

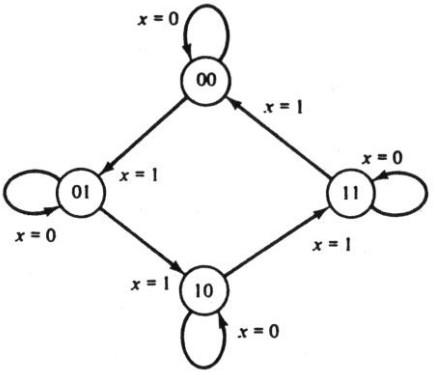
Çizim Örneği

Sıralı devrenin tasarım işlemi belirli bir örnek ile gösterilebilir. Tasarım işlemi devre özelliklerinin bir durum şemasına aktarılmasıyla başlar. Sonra bu şema durum çizelgesine çevrilir. Buradan da devre çizilir.

İkili Sayıcı

Şimdi saatli bir sıralı devre çizmek istiyoruz. Durumlar 00, 01, 10 ve 11 dir. Harici giriş $x = 1$ olunca durum değişsin. $x = 0$ ise durum değişmesin. Böyle devreye 2 bit ikili sayıcı denir. x girişi denetim değişkeni olarak sayma işleminde kullanılır.

İkili sayıcının 2 yaz-bozu olmalıdır. Çünkü her durum için 2 bit gerekmektedir. Devrenin durum şeması şekil 1.27 de gösterilmiştir. Bu devre $x = 1$ ise çalışır. 11 durumunu 00 durumu takip etmektedir. Böylece sayma devam eder. Devrenin çıktısı olmadığından şemada yazılmadı.



Şekil 1.27 İkili sayıcının durum şeması

Çizelge 1.5 İkili sayıcısının yükseltgeme çizelgesi

Şimdiki Durum		Giriş	Sonraki Durum		Yaz-boz Girişleri			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	×	0	×
0	0	1	0	1	0	×	1	×
0	1	0	0	1	0	×	×	0
0	1	1	1	0	1	×	×	1
1	0	0	1	0	×	0	0	×
1	0	1	1	1	×	0	1	×
1	1	0	1	1	×	0	×	0
1	1	1	0	0	×	1	×	1

Biz her zaman x i giriş değişkenini göstermek üzere kullanacağız. İki yaz-boz çıkışı A ve B sembolü ile gösterilmekte. Çizelge 1.5'in ilk 5 sütunu durum çizelgesinden elde edilir. Şema yardımıyla çizelge oluşturulur.

Yükseltgeme Çizelgesi

Sıralı devrenin yükseltgeme çizelgesi durum tablosunun genişletilmişidir. Bu genişleme genişletilmiş yaz-boz girişlerinin bir listesini içerir. Bu bize durum geçişlerini bulmamızı sağlar. Yaz-boz giriş durumları kullanılan yaz-bozun tipine bağlıdır. *JK* yaz-bozları için her yaz-bozun J ve K girişleri için sütunlara ihtiyacımız vardır. A yaz-boz girişlerini J_A ve K_A, B yaz-boz girişlerini J_B ve K_B olarak tanımlarız.

Çizelge 1.3 deki *JK* yaz-bozu için öngörülen genişletilmiş tablo, sıralı devrenin genişletilmiş tablosunun türetilmesi için kullanılır. Örnek olarak tablo 1.5 deki ilk sıra A yaz-bozu için şimdiki 0 durumundan diğer durum 0 a geçişe sahiptir. Tablo 1.3 te $Q(t) = 0$ dan $Q(t + 1) = 0$ durumların bir geçişini buluruz. *JK* yaz-bozu $J = 0$ ve $K = \times$ gerektirir. Dolayısıyla 0 ve X J_B ve K_B altındaki satıra kopyalanır. İlk sütun B yaz-bozunun şimdiki 0 durumundan sonraki 0 durumuna geçişini gösterir. 0 ve X J_B ve K_B altındaki ilk sütuna kopyalanır. Çizelge 1.5 in ikinci sütunu B yaz-bozunun şimdiki 0 durumundan sonraki 1 durumuna geçişini gösterir. Çizelge 1.3 ten $Q(t) = 0$ dan $Q(t + 1) = 1$ geçişi bulunur ve bu geçiş $J = 1$ ve $K = \times$ tir. Dolayısıyla 1 ve X J_B ve K_B altındaki 2. sütuna kopyalanır. Bu işlem çizelgenin her satırı ve her yaz-boz için yapılır.

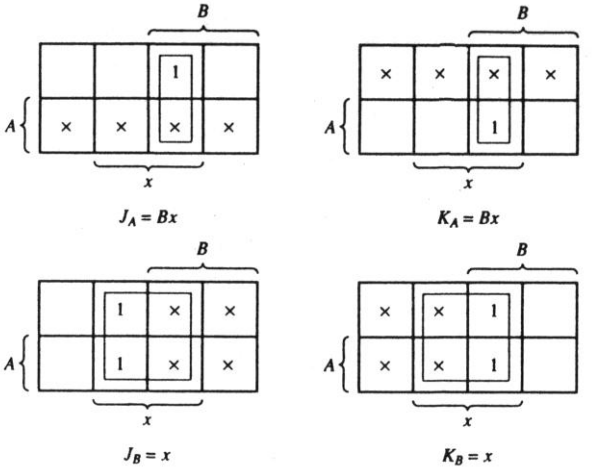
Örnek olarak çizelge 1.5 te, yükseltgeme çizelgesinde bulunan bilgiler değerlendirilirse, buradan bir sıralı devrenin sahip olduğu yaz-bozlar ve birleşik devre öğrenilebilir. 1.24 teki blok şemadan birleşik devrenin çıkışları dört yaz-boz girişlerine gitmelidir. (J_A, J_B, K_A, K_B). Birleşik devrenin girişleri harici giriş x ile yaz-bozların şu anki durumlarıdır. Bir birleşik devreyi belirten Boole fonksiyonları doğruluk çizelgesinden de elde edilir. Listenin tümündeki birleşik devre girişleri şu anki durum ve harici giriş sütunu ile belirtilir. Birleşik devre çıkışları yaz-boz girişleri sütununda belirtilir. Böylece bir yükseltgeme çizelgesi durum diyagramının doğruluk çizelgesine çevrilmesini sağlar ki buda sıralı devrenin parçası olan birleşik devrenin tasarımında gereklidir.

Birleşik devredeki Boole fonksiyonları basitçe türetilebilir. A, B, x değişkenleri girişlerdir. J_A, K_A, J_B ve K_B değişkenleri de çıkışlardır. Şekil 1.28 deki haritalara yükseltgeme çizelgesinden bilgiler aktarılır. Bu tablodan türetilen 4 yaz-boz giriş denklemi

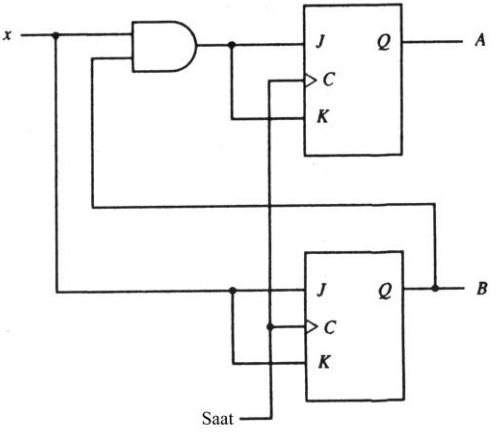
$$J_A = Bx \qquad K_A = Bx$$

$$J_B = x \qquad K_B = x$$

bulunur. Mantıksal şema Şekil 1.29'da çizilmiştir. İki *JK* yaz-bozu ve bir tane VE kapısı içerir. Saat geçişi sağlandığında J ve K girişleri sayıcının



Şekil.1.28 Sayıcı birleşik devresi için haritalar



Şekil 1.29 2 bit ikili sayıcının mantık şeması

sonraki durumunu belirler. Eğer $J = K = 0$ ise saat sinyalinin etkisi olmaz yani yaz-bozların durumu değişmez. Böylece $x = 0$ olduğunda tüm yaz-bozların dört girişi 0 a eşittir ve saat vuruşu devamlı uygulansa dahi yaz-bozların durumları değişmez.

Tasarım İşlemi

Sıralı devrelerin tasarımı yukarıdaki örnekte özetlendiği gibidir. Devrenin önce durum şeması çizilmiştir. Bu şemadaki dairelerin içindeki bitlerden gerekli yaz-bozların sayısı belirlenmiştir. Devrenin girişlerinin sayısı

daireler arasındaki bağlardan bulunur. Bundan sonra yaz-bozlar ve giriş değişkenleri adlandırılır ve sonra durum çizelgesi elde edilir.

m tane yaz-boz ve *n* girişle durum çizelgesi *m* sütunla şimdiki durumu *n* sütunla girişleri ve *m* sütunla gelecek durumu gösterir. Satır sayısı 2^{m+n} dir. Bir sütun şimdiki durum ve harici girişlerin tüm ikili kombinezonlarını gösterir. Durum çizelgesinde tüm sonraki durumları belirtmeliyiz. Bundan sonra yaz-boz tipi seçilir ve adlandırılır. Adlandırma iki harftir. Yaz-boz tipinin yükseltgeme çizelgesi (çizelge 1.3) kullanılarak yaz-bozların şimdiki ve gelecek durumlarından yaz-bozların giriş koşulları elde edilir.

Sıralı devrenin birleşik devre kısmının durum tablosu yükseltgeme çizelgeleri için uygundur. Şimdiki durum birleşik devrenin girişlerini oluşturur. Dolayısıyla birleşik devrenin doğruluk çizelgesindeki girişlerdir. yaz-bozların giriş koşulları ise birleşik devrenin çıkışlarıdır. Bunun anlamı sadeleştirmeyi kullanarak birleşik devre için yaz-bozların giriş fonksiyonları kümesini bulabiliriz.Her bir yaz-boz giriş fonksiyonu bir mantıksal şemadan bulunur. Birleşik devre ve yaz-bozlar sıralı devreyi oluşturmaktadır.

Yaz-bozların çıkışları genelde sıralı devrenin çıkışları ile ilişkilidir. Buna rağmen birleşik devre harici çıkışlara sahip olabilir. Bu durumda harici çıkışların Boole ifadeleri birleşik devre tasarım yöntemleri kullanılarak durum çizelgesinden elde edilir.

Yaz-boz giriş fonksiyonları kümesi sıralı devrenin cebirsel formudur. Sadeleştirme ile Boole fonksiyonları elde edilir. Bunlar yaz-bozların giriş fonksiyonları olarak belirlenir. Önce yaz-bozlar çizilerek tüm giriş ve çıkışları etiketlendirilir. Yaz-boz giriş ve çıkış fonksiyonlarına göre birleşik devre çizilir. Sonuçta, yaz-bozların çıkışları birleşik devrenin girişlerine ve birleşik devrenin çıkışları yaz-bozların girişlerine bağlanır.

SORULAR

1-1. Üç değişken için De Morgan teoreminin geçerliliğini doğruluk çizelgesi ile gösterin: $(ABC)' = A' + B' + C'$

1-2. Üç değişkenli ÖZEL-VEYA fonksiyonunu doğruluk çizelgesinde gösterin: $x = A \oplus B \oplus C$

1-3. Boole ifadelerini kullanarak aşağıdaki ifadeleri basitleştirin.

a. $A + AB'$

b. $AB + AB'$

c. $A'BC + AC$

d. $A'B + ABC' + ABC$

1-4. Boole ifadelerini kullanarak aşağıdaki ifadeleri basitleştirin.
- **a.** $AB + A(CD + CD')$
- **b.** $(BC' + A'D)(AB' + CD')$

1-5. De Morgan teoremini kullanarak aşağıdaki ifadelerin doğruluğunu gösterin.
- **a.** $(A + B)'(A' + B')' = 0$
- **b.** $A + A'B + A'B' = 1$

1-6. $F = x'y + xyz'$ Boole ifadesi veriliyor
- **a.** F' tümleyeni için matematiksel bir ifade yazınız
- **b.** $F.F' = 0$ olduğunu gösteriniz
- **c.** $F + F' = 1$ olduğunu gösteriniz

1-7. $F = xy'z + x'y'z + xyz$ Boole ifadesi veriliyor
- **a.** Fonksiyonun doğruluk çizelgesini gösterin.
- **b.** Boole ifadelerini kullanarak mantık şemasını çizin
- **c.** Boole ifadelerini kullanarak verilen fonksiyonu basitleştirin
- **d.** Basitleştirilen fonksiyonun doğruluk çizelgesini çizin ve ilk ifade ile karşılaştırın.
- **e.** Basitleştirilen ifadenin mantık şemasını çizin

1-8. Üç değişkenli haritaları kullanarak aşağıdaki ifadeleri basitleştirin.
- **a.** $F(x, y, z) = \Sigma(0, 1, 5, 7)$
- **b.** $F(x, y, z) = \Sigma(1, 2, 3, 6, 7)$
- **c.** $F(x, y, z) = \Sigma(3, 5, 6, 7)$
- **d.** $F(A, B, C) = \Sigma(0, 2, 3, 4, 6)$

1-9. Dört değişkinli haritaları kullanarak aşağıdaki ifadeleri basitleştirin
- **a.** $F(A, B, C, D) = \Sigma(4, 6, 7, 15)$
- **b.** $F(A, B, C, D) = \Sigma(3, 7, 11, 13, 14, 15)$
- **c.** $F(A, B, C, D) = \Sigma(0, 1, 2, 4, 5, 7, 11, 15)$
- **d.** $F(A, B, C, D) = \Sigma(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$

1-10. Aşağıdaki ifadeleri (1) çarpımların toplamı (2) toplamların çarpımı şeklinde basitleştirin.
- **a.** $x'z' + y'z' + yz' + xy$
- **b.** $AC' + B'D + A'CD + ABCD$

1-11. Aşağıdaki Boole ifadesini dört değişkenli haritaları kullanarak toplamların-çarpımı şeklinde basitleştirin. (a) VE-VEYA kapılarını kullanarak, (b) VE-TERSLEYEN kapılarını kullanarak devresini çizin.

$$F = (A, B, C, D) = \Sigma(0, 2, 8, 9, 10, 11, 14, 15)$$

1-12. Aşağıdaki Boole ifadesini dört değişkenli haritaları kullanarak toplamların-çarpımı şeklinde basitleştirin. (a) VEYA-VE kapılarını kullanarak, (b) VEYA-TERSLEYEN kapılarını kullanarak devresini çizin.

$$F = (w,x,y,z) = \Sigma(2,3,4,5,6,7,11,14,15)$$

1-13. Dikkate almama durumu *d* ifadesi ile birlikte *F* ifadesini (1) çarpımların toplamı (2) toplamların çarpımı şeklinde basitleştirin.

$$F(w, x, y, z) = \Sigma(0, 1, 2, 3, 7, 8, 10)$$
$$d(w, x, y, z) = \Sigma(5, 6, 11, 15)$$

1-14. Çizelge 1.2. yi kullanarak tam toplayıcının *S* çıkışı için çarpımların toplamı şeklinde Boole ifade elde edin. Üç değişkenli ÖZEL-VEYA kapıları ile $S = x \oplus y \oplus z$ şeklinde gösteriniz.

1-15. Üç değişkenli bir birleşik devrede giriş değişkenlerinde 1 ler 0 lardan daha fazla ise çıkış 1 diğer durumlarda çıkış 0 olsun. Doğruluk çizelgesi ve mantık şemasını çizin.

1-16. Giriş değişkenleri *x*, *y*,*z* ve çıkış değişkenleri *A*, *B*, *C* olan bir birleşik devre tasarlansın. İkili giriş 0, 1, 2, 3 olduğunda çıkış girişten bir fazla, diğer durumda 1 az olsun. Doğruluk çizelgesi ve mantık şemasını çizin.

1-17. *JK* yaz-bozunun girişleri arasına bir tersleyen ekleyerek *D* yaz-bozunun elde edilmesini gösterin.

1-18. *JK* yaz-bozunun karakteristik çizelgesini kullanarak *JK* yaz-bozunun yükseltgeme çizelgesini elde edin. Elde edilen çizelgeyi Çizelge 1.3 teki ile karşılaştırın.

1-19. Harici (*x*, *y*) iki girişli ve bir (*z*) çıkışlı bir sıralı devre tasarlayın. *A* ve *B* gibi iki tane *D* tipi yaz-boz içersin. Yaz-boz giriş fonksiyonları ve devre çıkış fonksiyonu aşağıdaki gibi olsun.

$$D_A = x\acute{y} + xA$$
$$D_B = x\acute{B} + xA$$
$$z = B$$

a. Devrenin mantıksal şemasını çizin.
b. Durum tablosunu oluşturun.

1-20. 2 bitlik ikili sayıcı tasarlayın. Bu sıralı devre iki yaz-boz ve *x* harici girişinden oluşsun. $x = 0$ olduğunda ikili sayıcı değer değiştirmesin. $x = 1$ olduğunda 11, 10, 01, 00, 11 sırasını izlesin yani 1 azalsın.

1-21. *JK* tipi iki yaz-boza ve *E*, *x* harici girişlerine sahip bir sıralı devre tasarlayın. *x* ne olursa olsun $E = 0$ olduğunda devre eski değerlerini korusun. E = 1, $x = 1$ de 00,01,10,11 ve 00 a gitsin. Yani 1 artsın $E = 1$, $x = 0$ durumunda ise 00 dan 11, 10, 01, 00 a gitsin yani 1 azalsın.

KAYNAKLAR

1. Hill, F.j., and G.R. Peterson, *Introduction to Switching Theory and Logical Desing*, 3rd ed. New York: John Wiley , 1981.
2. Mano, M.M., *Digital Desing*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1991.

3. Roth, C.H., *Fundamentals of Logic Desing*, 3rd ed. St. Paul, MN: West Publishing, 1985.
4. Sandige , R.S., *Modern Digital Desing*. New York: McGraw-Hill,1990.
5. Shiva, S. G., *Introduction to Logic Desing*. Glenview, IL: Scott, Foresman, 1988.
6. Wakerly, J.F., *Digital Desing Principles and Practices*. Englewood Cliffs, NJ: Prentice Hall, 1990.
7. Ward. S. A., and R.H. Halstead, Jr., *Computation Structures*. Cambridge, MA: MIT Press, 1990.

B Ö L Ü M 2

Sayısal Elemanlar

BU BÖLÜMDE

2-1 Birleştirilmiş Devreler
2-2 Kod Çözücüler
2-3 Seçiciler
2-4 Yazaçlar
2-5 Kaydırma Yazaçları
2-6 İkili Sayıcılar
2-7 Bellek Birimi

2.1 Birleştirilmiş Devreler

IC

Sayısal devreler, birleştirilmiş devrelerden yapılmıştır. Bir IC küçük silikon, yarı iletken kristaldir ve yonga diye adlandırılır. IC ler sayısal kapılar için elektronik elemanları içerirler. Bunlar transistörler, diyodlar, dirençler ve kapasitörlerdir. Yonga içindeki çeşitli kapılar devreye gerektiği biçimde birleştirilmiştir. Yonga, seramik ya da plastik kap içine monte edilmiştir ve dış bağlantıları ancak devreye ince altın tellerle kaynaklı ayaklar ile sağlanır. Küçük IC paketinde 14, daha büyük pakette 100 veya daha fazla ayak bulunur. Her bir IC paketinin ön yüzüne, tanımlanabilmesi için sayısal değeri yazılmaktadır. Her IC satıcısı bütün IC lerin imalatı hakkında bilgi ve kesin tanımları içeren veri kitapları ya da kataloglar yayınlarlar.

IC nin gelişim teknolojisine göre tek bir yonga içerisine konmuş kapıların sayısı hayli arttırılmış olabilir. Bazı yongalar birkaç iç kapıya sahiptir, bazılar ise yüzlerce hatta binlerce iç kapıya sahiptirler. Bunlar küçük, orta ya da büyük ölçekli birleştirilmiş aletlerdir.

SSI

Küçük Ölçekli Birleşik Aletler (SSI): Tek bir pakette birkaç bağımsız kapıdan meydana gelmiştirler. Kapıların giriş ve çıkışları doğrudan pake-

tin uçlarına birleştirilmiştir. IC de mevcut uçların sayısı sınırlıdır ve genellikle 10 dan küçüktür.

Orta Ölçekli Birleşik Aletler (MSI): Tek pakette yaklaşık olarak 10 ile 200 arası karmaşık kapıya sahiptirler. Genellikle kod çözücü, toplayıcı ve yazaç gibi temel sayısal fonksiyonları kapsarlar.

MSI

Büyük Ölçekli Birleşik Aletler (LSI): Tek pakette 200 ile birkaç 1000 arasında kapıya sahiptirler. Bunlar işlemciler, hafıza yongaları ve programlanabilir modüller gibi sayısal sistemleri kapsarlar.

LSI

Çok Büyük Ölçekli Birleşik Aletler (VLSI): Tek pakette binlerce kapıya sahiptirler. Örnek olarak büyük hafıza dizileri ve karmaşık mikrobilgisayar yongaları söylenebilir. Küçük boyutlara ve düşük fiyatlara sahiptirler. VLSI aletleri bilgisayar sistem tasarım teknolojisini tamamen değiştirmiştir.

VLSI

Sayısal birleştirilmiş devreler mantıksal işlevlere göre değil aynı zamanda kendine özgü teknolojik işlevlerine göre de sınıflandırılmıştır. Her bir sayısal mantık ailesi, temel elektronik devreler üzerinde kendine has daha karmaşık devrelere ve gelişmiş fonksiyonlara sahiptir. Temel devredeki her bir teknoloji VE-DEĞİL, VEYA-DEĞİL ya da TERSLEYEN kapısıdır. Elektronik bileşenler temel devrelerin yapımında kullanılmıştır. En çok kullanılanlar şunlardır.

TTL	Transistor-transistor mantık
ECL	Yayıcı-bağlanmış mantık
MOS	Metal-oksit yarı iletken
CMOS	Tümleyen metal-oksit yarı iletken

TTL Uzun süreden beri mantık ailesinde kullanılmaktadır ve standartlara uygundur. ECL sistemin ihtiyaç duyduğu yüksek hızlı işlerde üstünlüklere sahiptir. MOS devrenin ihtiyaç duyduğu yüksek yoğunluklu parça için elverişlidir ve CMOS sistemin ihtiyaç duyduğu düşük güçte parçada tercih edilir.

TTL mantık ailesi, temel VE-DEĞİL kapısı için eski teknolojide kullanılan diyot ve transistörlere göre gelişmiştir. Eski teknoloji DTL diye adlandırılmaktadır. Daha önceki diyodlar transistörlerle yer değiştirmiştir. TTL ailesi pek çok gelişme göstermiştir. Hızlı TTL ler, düşük güçlü TTL ler, Schottky TTL, düşük güçlü Schottky TTL ve gelişmiş Schottky TTL. TTL devresi için güç kaynağı 5 volttur ve iki mantıksal seviyesi yaklaşık 0 ve 3.5 volttur.

TTL

Yayıcı Bağlanmış Mantık, önceki yüksek hızlı sayısal devrelerin birleştirilmiş halidir. ECL, süper bilgisayar sistemlerinde yüksek hızda işaret işlemleri için kullanıldı. ECL kapısı işlemlerindeki transistörler doymamış durumdadır. Yaymada müsaade edilen durum 1 veya 2 nano saniye gecikmedir.

ECL

MOS, CMOS

Metal Oksit Yarı İletken, tek kutuplu transistörler de elektriği yalnızca tek tipte taşırlar. Bu durumda çift kutuplu transistörlerde TTL ve ECL kapıları kullanılır. Normal işlemlerde her iki taşıyıcı da vardır. *P*-kanal MOS a PMOS ve *n*-kanal MOS a NMOS denmektedir. NMOS transistör sadece bir MOS tipinde transistörün bir devrede müşterek olarak kullanılmasıdır. Tamamlayıcı MOS CMOS teknolojisinde birleştirilmiş PMOS ve NMOS transistörler bütün devrelerde tamamlayıcı kullanılır. CMOS un en önemli avantajı devrelerin küçük boyutlu oluşu, basit bir fabrikasyon tekniği gerektirmesi ve düşük güç tüketiminden dolayı ekonomik olmasıdır.

Bunların pek çok avantajları vardır. Özel kullanımlarda bilgisayar sistemlerinin tasarımında, tümleşik devreler farklı sayısal bileşenlere ihtiyaç duyulmasını sağlar. Sayısal bilgisayarların tasarımında bilinenler ile bileşik devrelerde karşılaşılan farklı parçalar çok önemlidir. Bu parçalar sayısal bilgisayarların dizaynında kullanılır.

2.2 Kod Çözücüler

Sayısal bilgisayarlarda, bilgilerin farklı değerleri ikili kodlarla belirlenmiştir. Çözücü devre ile birlikte kodluk ikili girişlerden 2^n lik farklı çıkışlar elde edilir. Eğer n bit kod bilgisi, bit kombinezonu ile kullanılmasaydı çıkış 2^n den daha az olabilirdi. Çözücülerin amaçları n girişli değişkenler ile 2^n lik ikili birleşimler meydana getirmektir.

Kod Çözücü

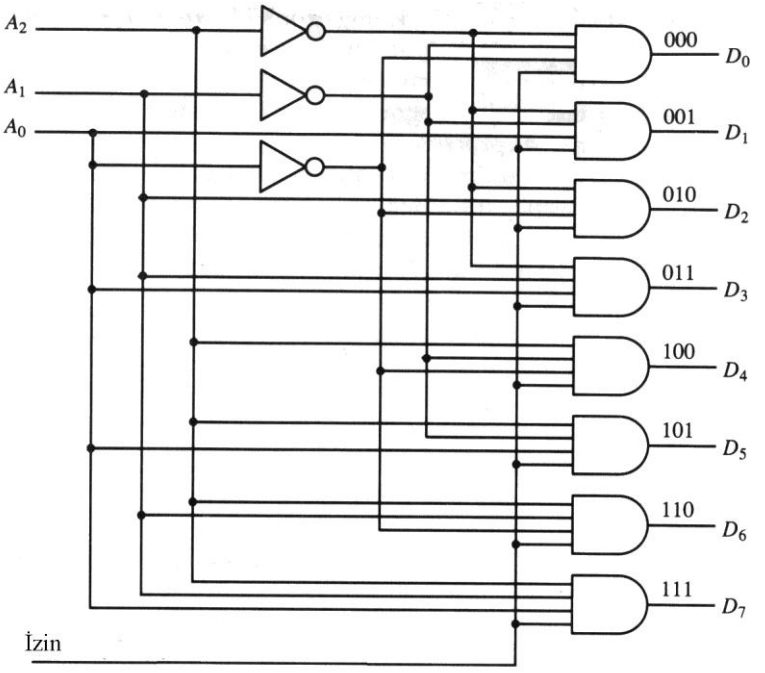
$m \leq 2^n$ olduğunda kod çözücüye $n \times m$ hatlı kod çözücü denir. Bu kod çözücünün amacı n farklı girişten 2^n ikili çıkış kombinezonu elde etmektir. Burada n giriş ve m çıkış vardır.

Şekil 2.1 de 3×8 hatlı kod çözücünün mantık şeması gösterilmektedir. A_0, A_1 ve A_2 olan 3 veri girişinden sekiz çıkış kodlanmaktadır. 3 tane TERSLEYEN, girişlerin tümleyenini almaktadır. 8 VE kapısının her biri ikili kombinezonlardan birini üretir. Bu kod çözücünün en temel uygulaması, ikiliden sekizliye çevirmedir. Giriş değişkenleri ikilik değer alır ve çıkış olarak sekizlik sayı sisteminin 8 rakamından biri alınır. Bununla birlikte 3×8 lik kod çözücü herhangi bir üç bit koddan 8 farklı çıkış elde etmek için kullanılır.

İzin Girişi

Ticari olarak üretilen kod çözücüler devrenin çalışmasını denetleyen bir veya daha fazla izin girişine sahiptir. Şekil 2.1 deki devrede bir izin girişi vardır (E). $E = 1$ olduğunda kod çözücü çalışır, $E = 0$ olduğunda çalışmaz.

Kod çözücünün çalışması Çizelge 2.1 deki doğruluk çizelgesi kullanılarak açıklanabilir. İzin girişi sıfır olduğunda ($E = 0$) 3 veri girişinin değerine bakılmaksızın tüm çıkışlar 0 olur. $E = 1$ olduğunda kod

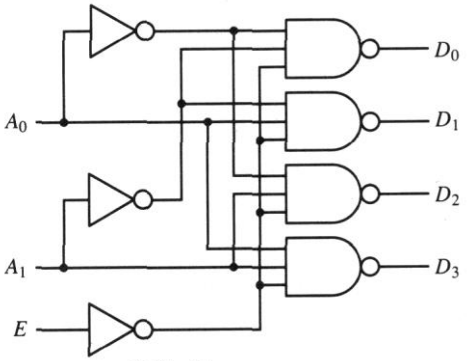


Şekil 2.1 3 × 8 Hatlı kod çözücü

Çizelge 2.1 3 × 8 lik kod çözücünün doğruluk çizelgesi

Enable	Inputs			Outputs							
E	A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	×	×	×	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

çözücü normal kipte çalışır. Her bir giriş için çıkışlardan sadece bir tanesi 1, diğer tümü 0 dır. Çıkış değişkeninin değeri 1 olduğunda giriş hattına bağlı olarak ikili sayının sekizlik karşılığı üretilir.



a) Mantık şeması

E	A_1	A_0	D_0	D_1	D_2	D_3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	×	×	1	1	1	1

b) Doğruluk çizelgesi

Şekil 2.2 VE-DEĞİL kapıları ile 2 x 4 hatlı kod çözücü

VE-DEĞİL Kapılı Kod çözücü

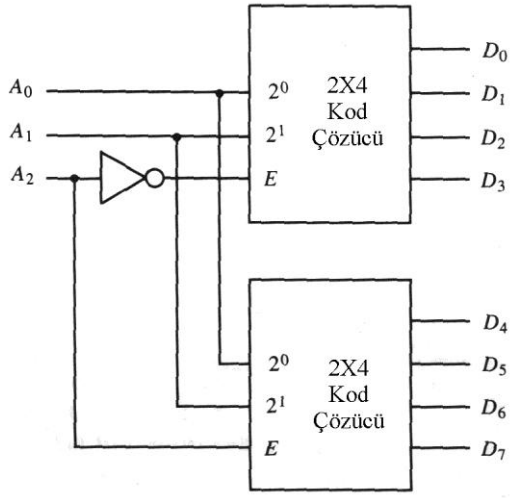
VE kapılarının çıkışlarının terslenmesi ile oluşturulan VE-DEĞİL kapıları kullanılmasına karşılık, aynı işi gerçekleştiren kod çözücüler daha ekonomik bir hale gelmiştir. 2 × 4 hatlı kod çözücü VE-DEĞİL kapıları ile izin girişleri gerçekleştirilmiş şekil 2.2 de gösterilmiştir.

E girişi sıfıra eşit olduğu zaman kod çözücü seçilir kılınmıştır. Doğruluk çizelgesinde görüldüğü gibi her zaman yalnızca bir çıkış 0 diğerleri 1 dir. Değerleri 0 olan çıkışlar A_0 ve A_1 girişlerindeki ikili sayıların karşılığı olarak gösterilirler. Devre seçilir kılınmadığında, yani $E = 1$ olduğunda diğer iki girişin (A_0, A_1) değerlerine bakılmaz, hiçbir çıkış seçilmez, bütün çıkışlar zaten 1 e eşittir. Genelde kod çözücü tamamlanmış veya tamamlanmamış çıkışlarla işlemin yapılmasına imkan verir. İzin girişi 0 ve 1 sinyal seviyelerinde de aktif hale geçer. Devreyi yeteri kadar seçilir kılmak kaydıyla, bazı kod çözücüler iki veya daha fazla izin girişine sahip olabilirler.

Kod Çözücülerin Genişletilmesi

İhtiyaç duyulan kod çözücü elde bulunan kod çözücülerden çok daha büyük olabilir. Böyle bir durum oluştuğunda, elde bulunan iki veya daha fazla kod çözücüyü birlikte kullanarak daha büyük ve kullanıma uygun kod çözücüler elde edebiliriz. Örneğin; eğer 6 x 64 hatlı bir kod çözücüye ihtiyaç varsa bu, 4 tane 4 × 16 hatlı kod çözücü birleştirilerek elde edilebilir.

Şekil 2.3 te izin girişleri birleştirilerek büyütülmüş bir kod çözücünün nasıl oluşturulduğu görülmektedir. İki adet 2 × 4 hatlı kod çözücü



Şekil 2.3 İki tane 2 × 4 kod çözücü kullanılarak oluşturulmuş bir tane 3 × 8 kod çözücü

birleştirilerek 3 × 8 hatlı bir kod çözücü meydana getirilmiştir. En az önemli olan iki bit, her iki kod çözücüye bağlanır. En önemli bit ise kod çözücülerden birine direk, diğerine ise terslenerek izin girişi olarak bağlanır. *E* girişi 1 olduğunda her kod çözücünün seçilir kılındığı kabul edilir. *E* girişi 0 olduğunda kod çözücü seçilir kılınmamıştır ve tüm çıkış seviyeleri sıfırdır. $A_2 = 0$ olduğunda üstteki kod çözücü seçilir kılınmış alttaki kod çözücü seçilir kılınmamıştır. Bu durumda alttaki kod çözücünün çıkışları pasif duruma geçmiştir ve bütün çıkış seviyeleri sıfırdır. $A_2 = 0$ olduğunda üstteki kod çözücü A_0 ve A_1 giriş değerlerine bağlı olarak D_0 dan D_3 e kadar olan çıkışları üretir. $A_2 = 1$ olduğunda üstteki kod çözücü seçilir kılınmazken alttaki kod çözücü seçilir kılınır. A_2 girişi 1 durumunda olduğunda alttaki kod çözücü D_4 ten D_7 ye kadar olan çıkışların ikili karşılığını üretir.

Örnekte, kod çözücüler ya da diğer birlikte çalışan mantıksal parçalarda izin girişinin yararlı bir şekilde nasıl kullanıldığı gösterilmektedir. İzin girişleri, iki veya daha fazla devrenin fonksiyonlarını değiştirmeden daha fazla giriş ve çıkış yollarına sahip olmak amacı ile birbirine bağlanarak genişletilmesine elverişlidir.

Kodlayıcılar

Kodlayıcılar, kod çözücülerin tam ters işlemini yapan sayısal devrelerdir. 2^n veya daha az giriş hattına ve *n* tane çıkış hattına sahiptirler. Çıkış hatları giriş değerlerine bağlı olarak uygun ikili kodlar şeklinde üretilmektedir.

Çizelge 2.2 İkiliden sekizliye kodlayıcı için doğruluk çizelgesi

Girişler								Çıkışlar		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Çizelge 2.2 de kodlayıcı örneğinin sekizlikten ikiliye dönüşümünün doğruluk çizelgesi görülmektedir. 8 giriş hattının her birine verilen sekizlik rakam ikilik sayı karşılığına dönüşerek üç çıkış hattına gider. Herhangi bir zamanda girişlerden sadece bir tanesi 1 değerini alabilir, diğer durumda devre anlamsızdır, çalışmaz.

Kodlayıcı, çizelgede görüldüğü gibi direk şekilde VEYA kapıları ile kurulmuş kararlı girişleri uygulamaya koyabilir. Giriş sekizlik sayı 1, 3, 5 veya 7 ise çıkış $A_0 = 1$ dir. Diğer iki çıkış için de benzer durumlar geçerlidir. Bu durumlar Boole fonksiyonları kullanılarak ifade edilir.

$$A_0 = D_1 + D_3 + D_5 + D_7$$

$$A_1 = D_2 + D_3 + D_6 + D_7$$

$$A_2 = D_4 + D_5 + D_6 + D_7$$

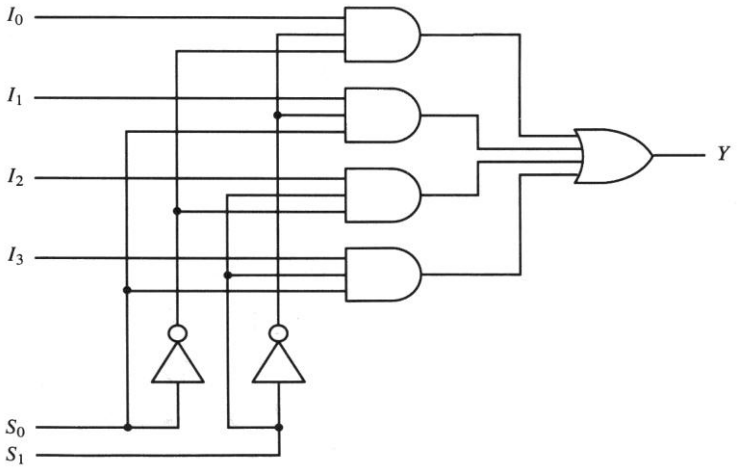
Kodlayıcı üç adet VEYA kapısı ile oluşturulur.

2.3 Seçiciler

Seçiciler

Bir seçici 2^n tane giriş hattının bir tanesinden aldığı ikili bilgiyi, tek bir çıkışa yollayan birleşik bir devredir. Seçim girişlerinin ayarlanması ile belirli bir veri giriş hattının değeri tek çıkışa gönderilir. 2^n den 1 e seçici 2^n giriş hattına sahiptir, n seçim giriş hatları hangi giriş datasının çıkışa yollanacağını belirtir.

4 × 1 seçici Şekil 2.4 te gösterilmiştir. 4 veri girişinden her biri I_0 dan I_3 e kadar VE kapılarının girişlerine uygulanır. VE kapısının çıkışları bir VEYA kapısına uygulanarak tek bir çıkış elde edilir. S_0 ve S_1 seçim girişleri hangi girişin çıkışa yönlendirileceğini belirler.



Şekil 2.4 4 × 1 Hatlı seçici

Devrenin çalışmasını görmek için S_1S_0 = 10 olduğu durum göz önüne alınsın. I_2 girişinin bağlı olduğu VE kapısının diğer girişleri 1 dir. 3 giriş kapısı I_2 ye bağlanmıştır. Diğer 3 VE kapısından en az bir tanesinin girişi diğerlerinin çıkışını 0 yapacak şekilde 0 a eşit olur. Böylece VEYA kapısının çıkışı I_2 değerine eşittir. Seçilen girişten çıkışa doğru bir yol elde edilir. 4 × 1 seçici 6 giriş ve 1 çıkışa sahiptir. Doğruluk çizelgesinin yardımıyla 2^n den 6 giriş değişkeninde 64 devreye ihtiyaç vardır. Bu son derece uzun bir çizelgedir. Seçicinin çalışmasını daha kısa ifade edebilmek için fonksiyon çizelgeleri kullanılır. Seçici için fonksiyon çizelgesi çizelge 2.3 te gösterilmiştir.

Veri Seçici

Çizelge 2.3, 4 veri girişi arasındaki bağlantı ve çıkış hattını, seçilmiş S_0 ve S_1 giriş hatlarına bağlı olarak göstermektedir. Seçim girişleri 00 iken Y çıkışı I_0 girişine eşittir. Seçim girişleri 01 iken Y çıkışı I_1 girişine eşittir, diğerleri ikinin kombinezonları şeklindedir. Seçici aynı zamanda *veri seçici* olarak adlandırılır. Çok sayıda veri girişinden biri seçilerek ikili biçimde çıkışa yönlendirilir.

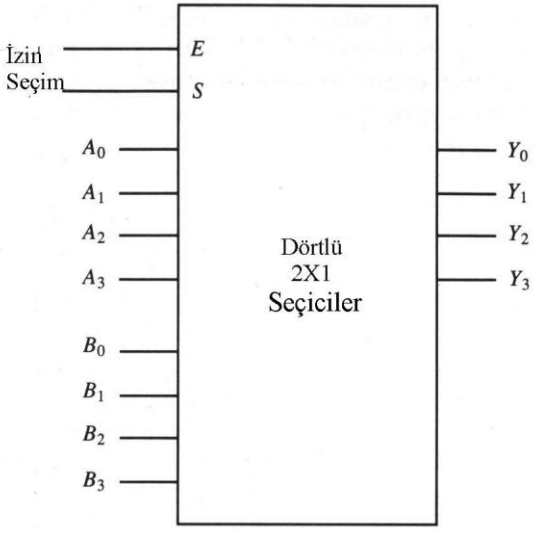
Çizelge 2.3 4 × 1 Seçici için fonksiyon çizelgesi

Seçim		Çıkış
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

VE kapıları ve TERSLEYEN ler bir kod çözücü devre oluştururlar ve gerçekte seçilmiş giriş hatlarını çözerler. Genelde bir 2^n den 1 e seçicinin boyutu 2^n sayıda veri girişi ve tek çıkış hattı ile belirtilir. Seçiciler MUX olarak isimlendirilir.

Bir seçici bir $n \times 2^n$ kod çözücü devresine benzer. Tek farkı kod çözücüye tek bir çıkış yapılmış olmasıdır. Büyüklüğü aynı kod çözücüler gibi giriş ucu sayısı ile ölçülür. Seçicide izin girişleri bir veya daha fazla bulunabilir. Bu girişler birden fazla seçicinin IC şeklinde yapılmasını sağlarlar.

Bazı durumlarda iki veya daha fazla MUX bir entegre devrede birleştirilir. Çoklu birim yapılarında MUX ların hepsi için seçim veya izin girişleri genelde ortaktır. 2 den 1 e 4 lü MUX şekil 2.5 te blok şema olarak gösterilmiştir. Devre 4 MUX a sahiptir ve bunlardan her biri iki giriş hattından birini seçebilir. Y_0 çıkışı A_0 veya B_0 girişinin herhangi birinden olabilir. Benzer şekilde Y_1 çıkışı A_1 veya B_1 girişinin herhangi birinin değerini alabilir ve böylece devam edilir. Bir seçilmiş S giriş hattı 4 MUX tan herhangi birinin bir hattını seçer. E izin girişi normal çalışmada aktif olmak zorundadır. Her ne kadar devre 4 MUX a sahipse de bu 4 bitlik veri hattından birini seçen devre olarak düşünülebilir. Fonksiyon çizelgesinde gösterildiği gibi $E = 1$ olduğunda birim aktif hale geçer.



E	S	Y
0	×	Hepsi 0
1	0	A
1	1	B

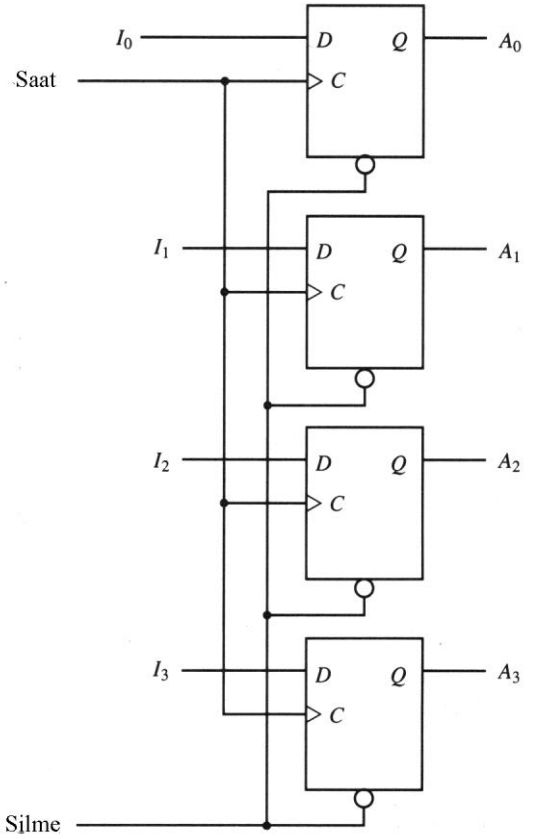
(a) Blok diyagram

Şekil 2.5 Dörtlü 2 × 1 seçiciler

Daha sonra $S = 0$ ise A nın 4 girişi 4 çıkışa yönlendirilir. Diğer taraftan $S = 1$ ise 4 B girişi 4 çıkışa yönlendirilir. S nin değeri ne olursa olsun $E = 0$ ise çıkışlar 0 olur.

2.4 Yazaçlar

Bir yazaç bir bitlik bilgiyi saklama kapasitesine sahip yaz-bozlardan oluşan yaz-bozlar grubudur. n bit yazaç, n adet yaz-boza ve n bitteki herhangi ikili bilgiyi saklama kapasitesine sahiptir. Yaz-bozlara ilave olarak, bir yazaç belirli verileri işleme amaçları oluşturmak için kullanılan birleşik kapılara sahip olabilir. Daha geniş olarak bir yazaç kapılar ve yaz-bozlardan oluşur. Yaz-bozlar ikili bilgiyi tutar ve kapılar yeni bilginin yazaç içerisine aktarılıp aktarılmadığını denetlerler.



Şekil 2.6 4 Bit yazaç

Yazaçların çeşitli tipleri vardır. En basit yazaç yaz-bozlardan oluşur, harici kapıya sahip değildir. Şekil 2.6 da 4 tane *D* yaz-bozundan oluşan bir yazaç gösterilmektedir. Tüm yaz-bozların saat girişleri tetiklemenin çıkan kenarında bulunur ve 4 girişteki ikili veriler 4 bitlik yazaca aktarılır. 4 çıkışta, yazaçtaki ikili bilgiyi herhangi bir zamanda saklamak içindir. Silme girişi her bir yaz-bozda özel bir uçtur. Bu giriş 0 olduğu zaman tüm yaz-bozlar saatten bağımsız olarak sıfırlanır. Silme girişi saatli işlemler için tüm yazaçlara 0 aktarmada kullanılır. Silme girişi normal çalışma esnasında 1 de tutulmalıdır. Saat sinyali D girişini denetler fakat silme girişi saatten bağımsızdır.

Yazaca Yükleme

Yeni bilginin bir yazaca aktarımına, yazaca yükleme denir. Eğer tüm yazaç bitleri ortak saat vuruşu geçişinde aynı anda yükleniyorsa, yükleme paralel olmaktadır. Şekil 2.6 daki yazacın *C* girişlerine bir saat geçişi uygulanarak I_0 dan I_3 e kadar girişlerin tümü paralel olarak yüklenir.

Paralel Yüklemeli Yazaçlar

Çoğu sayısal sistemler devamlı darbeler zinciri oluşturmak için ana saat üretecine sahiptir. Tüm yaz-bozlara ve sistemdeki yazaçlara saat darbeleri uygulanır. Ana saat, tüm sistem parçalarına sabit vuruş sağlamak için çalışan bir pompa olarak düşünülebilir. Paralel denetleme girişli 4 bit yazaç kapılar ile *D* girişlerine yönlendirilmiştir (Şekil 2.7). *C* girişleri saat darbelerini her zaman alır. Saat girişindeki buffer kapısı, saat üretecinden alınan güç gereksinimini alır. Saat sadece bir girişe bağlandığı zaman, 4 girişe bağlı olduğuna nazaran daha az güç çeker.

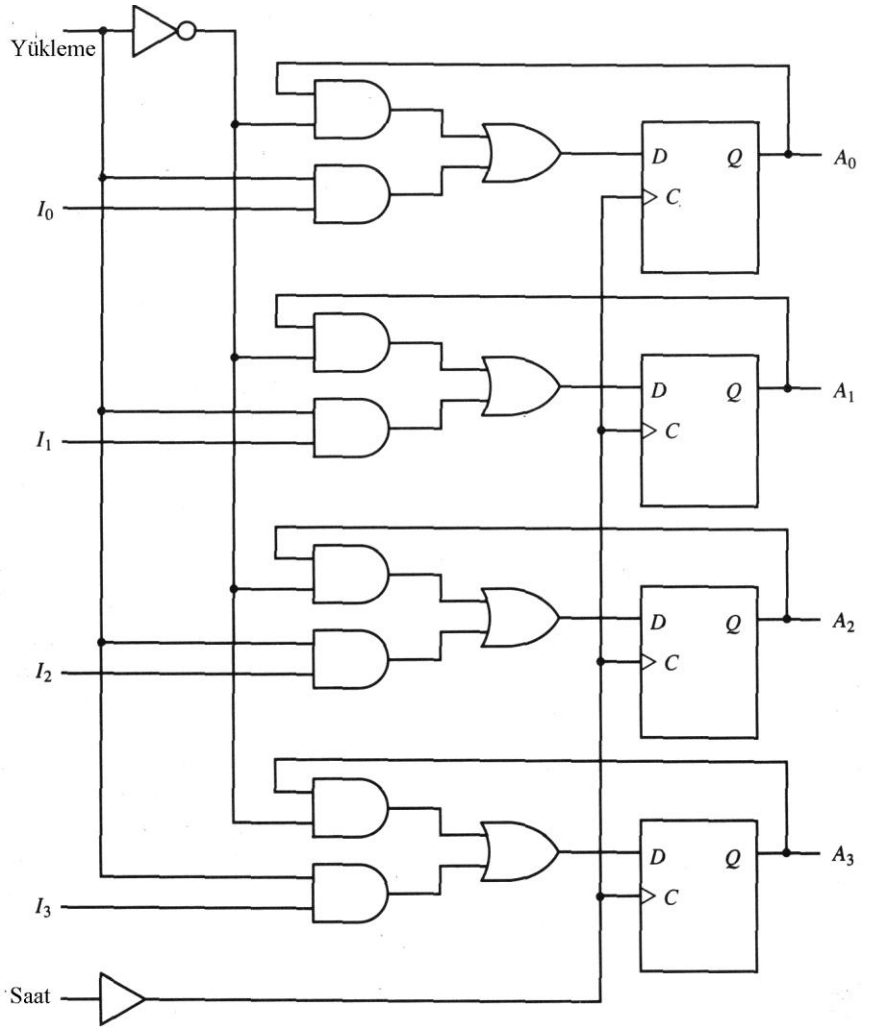
Yükleme Girişi

Yazaçtaki yükleme girişi her bir saat vuruşunun eylemini belirler. Yükleme girişi 1 olduğunda, 4 girişteki veri bir sonraki saat vuruşunun yukarı çıkışında yazaçlara aktarılır. Yükleme girişi 0 olduğunda veri girişleri ayrılır ve yaz-bozların *D* girişleri yazaç çıkışlarına bağlanır. Çünkü *D* yaz-bozu durum değiştirme özelliğine sahip değildir. Her bir saat vuruşu ile *D* girişleri, çıkışların bir sonraki konumunu belirler.

Saat girişleri her zaman *C* girişlerine uygulanmaktadır. Yükleme girişleri bilginin alınıp alınmadığını veya bir sonraki darbede kabul edilip edilmeyeceğini belirler. Girişlerden yazaçlara bilgi aktarımı tek bir saat işareti girişi esnasında tüm bitler için aynı anda yapılır.

2.5 Kaydırma Yazaçları

İkili bilgiyi bir veya daha fazla yöne kaydıran yazaçlara kaydırma yazacı denir. Bir kaydırma yazacının mantık düzeni kademeli olarak bağlı yaz-bozlardan oluşur. Bir yaz-bozun çıkışı bir sonraki yaz-bozun girişine bağlıdır. Tüm yaz-bozlar ortak saat vuruşu alırlar.

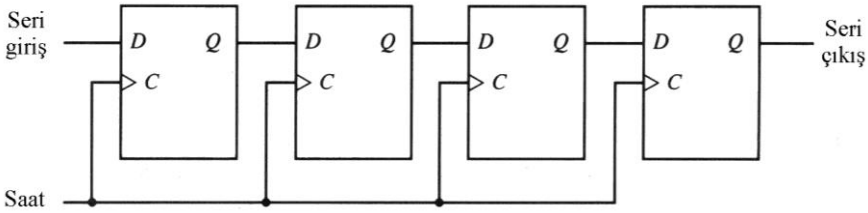


Şekil 2.7 4 Bit paralel yüklemeli yazaç

Seri Giriş

En basit kaydırma yazacı şekil 2.8 de gösterildiği gibi sadece yaz-bozları kullanır. Bir yaz-bozun çıkışı diğer *D* yaz-bozunun girişine bağlanmıştır. Saat tüm yaz-bozlarda aynıdır. *Seri giriş* en soldaki yaz-boza uygulanır. *Seri çıkış* sağa doğru olacak şekilde en sağdaki yaz-bozdan alınır.

Hangi saat darbelerinde olduğunu anlamak için bazen kaydırmayı denetlemek gerekir. Böylece kaydırma yapmak istenmediği zaman saat işareti yazaç girişlerinden ayrılır. Şekil 2.8 deki kaydırma yazacında, kaydırma, saat girişi bir VE kapısının ilk girişine bağlanarak denetlenebilir ve VE kapısının ikinci girişi saati yetkilendirmek için kullanılır.



Şekil 2.8 4 bitlik kaydırma yazacı

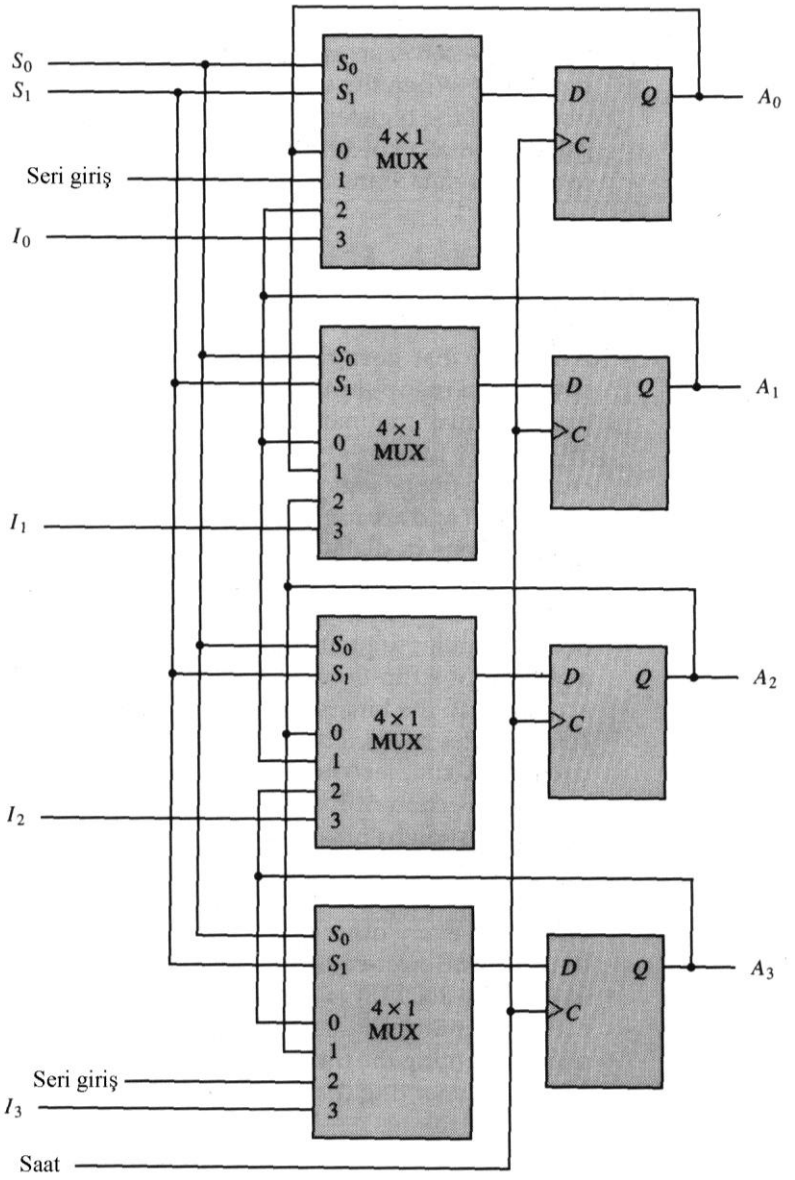
Bununla beraber yaz-bozların *D* girişlerine kaydırma işlemlerini denetlemek için harici devreler yapılabilir.

Paralel Yüklemeli Çift Yönlü Kaydırma Yazaçları

Yalnızca tek bir yönde kaydırma özelliğine sahip yazaçlara tek yönlü, her iki yönde de kaydırma özelliğine sahip yazaçlara da çift yönlü kaydırma yazaçları denir. Bazı kaydırma yazaçları paralel aktarımlar için gerekli giriş ve çıkış bağlantılarını sağlarlar. Genel kaydırma yazaçlarının çoğu aşağıda belirtilen özelliklerin tümüne, diğerleri de en az bir tanesine sahiptir:

1. Tüm işlemlerde uyumluluğu sağlamak için ortak bir saat girişi
2. Sağa kaydırma işlemi ve seri giriş hattı sağ tarafta bağlanmıştır.
3. Sola kaydırma işlemi ve seri giriş hattı sol tarafta bağlanmıştır.
4. Paralel yükleme işlemi, *n* adet giriş hattına paralel aktarımla gerçekleştirilir.
5. *n* adet paralel çıkış hattı.
6. Devamlı saat darbeleri uygulandığında yazacın değişip değişmediğini denetleyen bir denetleme durumu.

Dört bitlik çift yönlü paralel yüklemeli kaydırma yazacı şekil 2.9 da gösterilmektedir. Her bir kat, bir *D* yaz-bozundan ve 4 × 1 lik seçiciden oluşur. Seçilmiş S_0 ve S_1 girişlerinin her biri *D* yaz-bozları için veri girişlerinden birini seçerler. Seçim hatları çizelge 2.4 teki fonksiyon çizelgesine uygun olarak yazacın çalışma kipini denetler. Kip denetimi $S_1S_0 = 00$ olduğunda her MUX un veri girişi 0 seçilir. Bu durumda her bir yaz-boz un çıkışından aynı yaz-boz un girişine bir yol oluşturulur, bir sonraki saat geçişinde ikili değer her bir yaz-boza aktarılır. $S_1S_0 = 01$ olduğunda her bir MUX ta 1 olan giriş, ona karşılık gelen yaz-bozun veri girişine bağlanır.



Şekil 2.9 Paralel yüklemeli çift yönlü kaydırma yazacı

Bu sağa kaydırma işlemine neden olur. Seri girişteki veri yaz-boz A_0 a transfer edilir ve her bir A_{i-1} yaz-bozunun içeriği A_i yaz-bozuna aktarılır. (i = 0, 1, 2) S_1S_0 = 10 olduğunda sola kaydırma işlemi olur. Seri girişteki veri A_3 yaz-bozuna gider ve her bir A_{i+1} yaz-bozunun içeriği A_i yaz-

Çizelge 2.4 Şekil 2.9 daki yazaç için fonksiyon çizelgesi

Kip denetimi		
S_1	S_0	Yazaç işlemi
0	0	Değişiklik yok
0	1	Sağa kaydırma (aşağı)
1	0	Sola kaydırma (yukarı)
1	1	Paralel yükleme

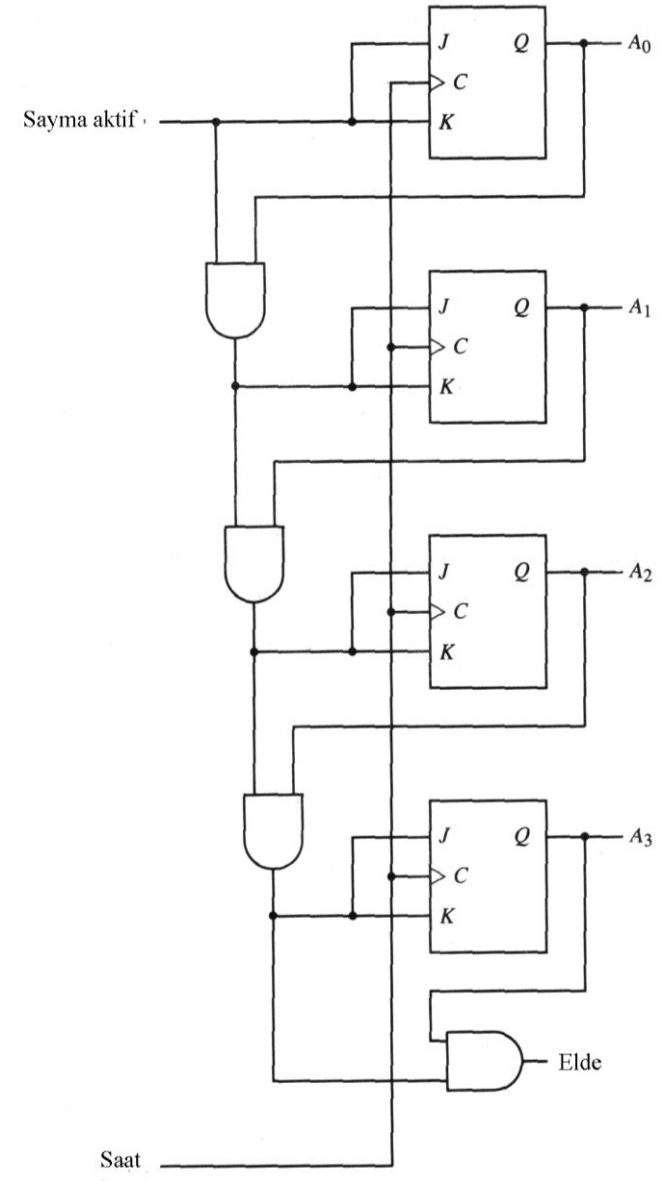
bozuna aktarılır. (i = 0, 1, 2) S_1S_0 = 11 olduğunda I_0 dan I_3 e kadar girişlerdeki bilgi, paralel işlem sonucunda ona karşı gelen yaz-boza aktarılır. Şemanın çizilişine dikkat edilirse sağa kaydırma işlemi yazacın içeriğini aşağı doğru, sola kaydırma işlemi yazacın içeriğini yukarı doğru kaydırır.

Kaydırma yazaçları sık sık sayısal sistemleri birbirine bağlamak için kullanılır. Örneğin n sayıdaki bit bilgiyi aktarmak için, eğer kaynakla gidilecek yer arasındaki mesafe uzaksa, n sayıdaki hattı kullanmak pahalı olur. Tek bir hat kullanılarak bilgi seri olarak aktarılır ve bu yol daha ekonomiktir. Göndericiler n sayıda biti paralel olarak kaydırma yazacına yükler ve sonra seri çıkış hattından veriyi iletirler. Alıcılar veriyi seri olarak kaydırma yazacı içine alır, n sayıda bit girildiğinde bilgiler toplanarak yazaçların çıkışlarından paralel olarak alınır. Böylece gönderici paralelden seriye veri çevirmesi yapar ve alıcı gelen seri veriyi tekrar paralele çevirerek aktarır.

2.6 İkili Sayıcılar

Giriş vuruşlarının uygulanmasıyla önceden belirlenmiş değerler alan yazaçlara sayıcı denir. Giriş vuruşları saat vuruşları olabileceği gibi başka dış girişler de olabilir. Bunlar belli zamanlarda ve aralıklarda olabileceği gibi rasgele de olabilir. Sayıcılar sayısal mantık bulunan hemen hemen tüm cihazlarda bulunur. Bunlar bir olayın kaç defa tekrar ettiğini saymakta kullanılır ve zaman sinyalleri üretirler. Böylece sayısal bilgisayarlarda işlemlerin sırasını denetlerler.

Bir sayaç birçok farklı sırayı izleyebilir. En basit olanları en düz sırayı izleyenlerdir. İkili sayım sırasını izleyen bir sayaca ikili sayıcı denir. n bit ikili sayıcı n bit yazaçtır, n tane yaz-boz u vardır ve uygun birleşik devresi ile ikili sayım yapar. n bit için sayım 0 dan 2^n–1 e kadar sürer. İkili sayıcıların tasarım çizimi Bölüm 1.7 de görüldüğü gibi yapılır. Basit bir alternatif tasarım, durum sıralarının incelenmesi ile bulunabilir.



Şekil 2.10 4 bit saatli ikili sayıcı

İkili sayıların sayma sırası 0000, 0001, 0010, 0011 vb gibi olduğunda, düşük mertebeli bitlerin her sayma işleminden sonra tümleyeni alınır. Diğer bitlerin ise, bir sayımdan diğerine ve düşük mertebedekilerin hepsi 1 ise tümleyenleri alınır. Örneğin 0111 (7) den 1000 (8) elde etmek için; (a) düşük mertebedeki bitlerin tümleyeni alınır, (b) ikinci sıradaki düşük

mertebeli bitin tümleyeni alınır. 0111 in ilk biti 1 dir, (c) üçüncü sıradaki düşük mertebeli bitin tümleyeni alınır. 1, 2 ve 3. Bitler 1 olduğundan tümleyeni 0 dır, (d) dördüncü sıradaki bit 0 olduğundan tümleyeni 1 dir.

Bir sayıcı devrede genellikle tümleyen alabilme özelliğine sahip yaz-bozlar kullanılır. *JK* ve *T* tipi yaz-bozların her ikisi de bu özelliğe sahiptir. *JK* yaz-bozun *J* ve *K* girişlerinin her ikisi de 1 ise ve saat işareti pozitif geçişe sahipse tümleyen alınır. $J = K = 0$ ise yaz-bozun çıkışı değişmez. Bununla birlikte sayıcı yaz-bozlar saat sinyali kaldırılmaksızın izin girişi ile aktif ya da pasif yapılabilir.

Saate bağlı ikili sayıcılar düzenli görüntüye sahiptir. Şekil 2.10 da 4 bitlik ikili sayıcı gösterilmektedir. Tüm yaz-bozların *C* girişleri ortak saat işareti alır. Eğer sayma aktif 0 ise tüm *JK* girişleri 0 a yönlendirilir ve sayıcının çıkışı değişmez. A_0 ın birinci aşaması sayma aktif 1 olduğunda ve saat bir pozitif geçişe geçtiğinde tamamlanmış olur. Diğer üç yaz-bozun her biri kendilerinden önceki en küçük önemli yaz-bozun çıkışı 1 olduğunda ve sayma aktif 1 olduğunda değer değiştirir. VE kapısının bağlanması *J* ve *K* girişleri için gerekli mantığı üretir. Çıkış elde si sayıcıyı bir adım daha genişletmek için kullanılabilir. Her bir adım ilave bir yaz-boz ve VE kapısı içerir.

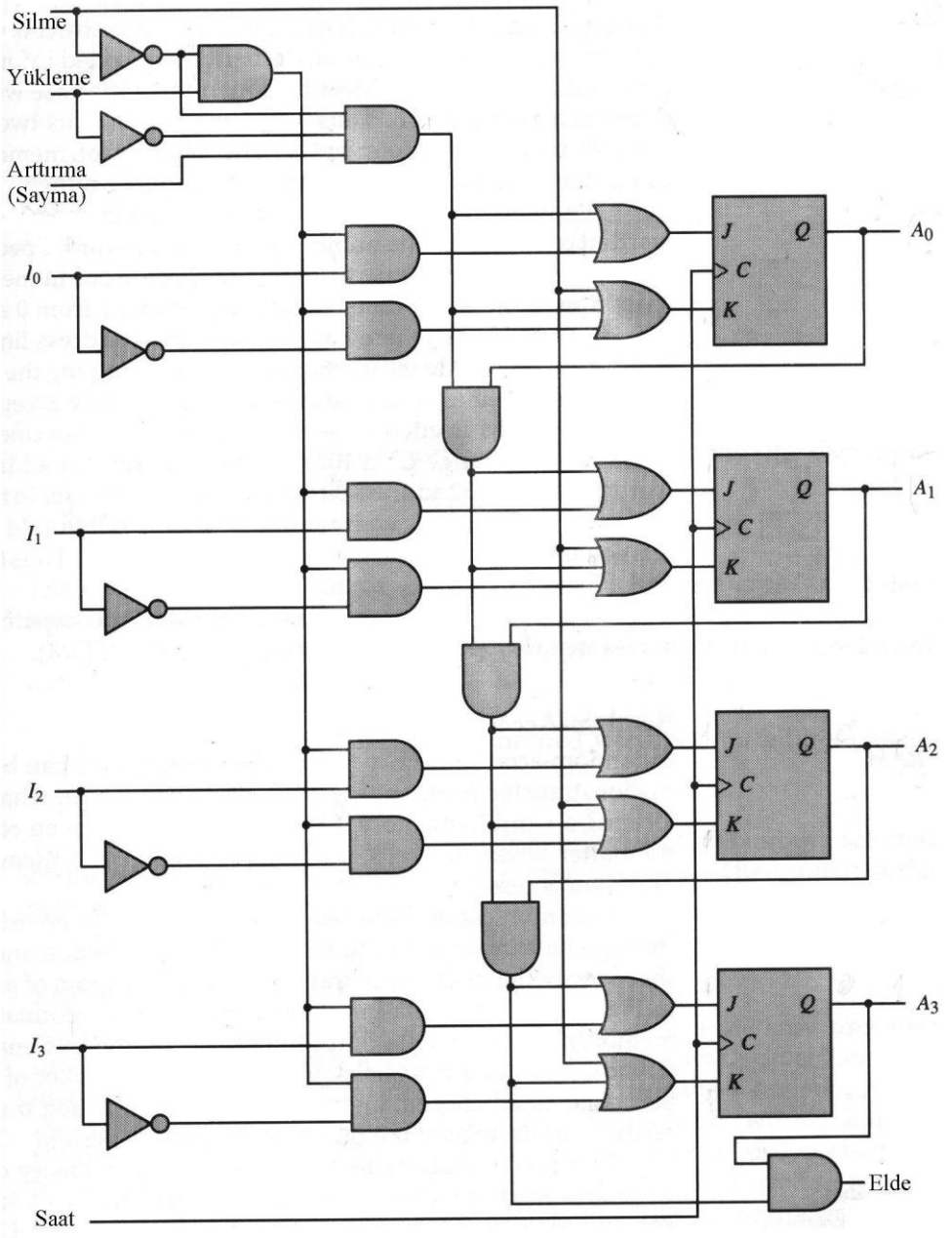
Paralel Yüklemeli İkili Sayıcı

Sayısal sistemlerde sayma işleminden önce bir başlangıç değerini elde etmek için paralel yükleme işlemi yapabilen sayıcılara oldukça sık ihtiyaç duyulur. Şekil 2.11 ikili sayıcının mantık şemasını göstermektedir. Bu ikili sayıcı, bir paralel yükleme kapasitesine ve aynı zamanda saate bağlı olarak silinebilme kapasitesine sahiptir. Sonucun 1 e eşit olması için silme girişi ile tüm *K* girişleri 1 olmalıdır. Böylece bütün yaz-bozlar sonraki saat vuruşunda silinirler (silme girişi 0 olduğu zaman). Yükleme girişi 1 olduğu zaman sayma işlemi çalışmaz ve 4 paralel girişten veri dört yaz-boz içine yüklenir.

Silme, yükleme girişleri 0 ve sayma girişi 1 olduğunda devre bir ikili sayıcı olarak çalışmaya başlar. Devrenin işlemleri Çizelge 2.5'te özetlenmektedir. Vuruşlar *C* ucuna uygulandığı zaman dahi silme, arttırma ve yükleme girişlerinin tümü 0 olduğunda çıkışta değişiklik olmaz, devre çalışmaz. Eğer silme ve yükleme girişleri mantıksal 0 da kalırsa, sayıcının arttırma giriş denetleme işlemi ve çıkışları, saatin her bir pozitif geçişinde, bir sonraki ikili sayıyı değiştirir.

Arttırma

Paralel yüklemeli sayıcılar sayısal bilgisayarların tasarımına çok uygundur. Arttırma işlemi yazacın içeriğini 1 arttırır. Bir saat süresi boyunca giriş sayısı belirlenerek yazacın içeriği 1 arttırılabilir.



Şekil 2.11 Paralel yüklemeli saatli silmeli 4 bit ikili sayıcı

Çizelge 2.5 Şekil 2.11 deki yazaçların fonksiyon çizelgesi

Saat	Silme	Yükleme	Arttırma	İşlem
↑	0	0	0	Değişiklik Yok
↑	0	0	1	Sayma 1 artar
↑	0	1	×	Yükleme Girişleri $I_0...I_3$ arasıdır.
↑	1	×	×	Çıkışlar 0 la silinir.

2.7 Bellek Birimi

Kelime, Byte

Bir bellek birimi saklama yapabilen yazaçlardan oluşmuş bir topluluktur. Bellek, *kelime* diye tanımlanan ikili bilgileri bit grupları halinde saklar. Bellek kelimesi 1 lerin ve 0 ların oluşturduğu bir gruptur ve bir sayıyı, işlem kodunu, bir veya daha fazla alfasayısal karakteri veya diğer ikili kodları içerir. 8 bitten oluşan gruplara *byte* denir. Bir çok bilgisayar belleğinde kelimeler kullanılır ki, bitlerin 8 katı byte tır. Böylece 16 bit kelime 2 byte, 32 bit kelime 4 byte içerir. Ticari bilgisayarların hafızasında saklayabileceği byte sayısı genellikle bellidir.

Bellek biriminin iç yapısını kelimelerin sayısı belirtir ve her kelime bir çok bit içerir. Özel giriş hatları bir tek kelime seçen adres yolları olarak adlandırılır. Bellekte her kelime belirtilen bir sayıya atanır ve adres diye isimlendirilir. 0 dan başlayıp bir bir artarak 2^k-1 e kadar devam eder. k adres hatların sayısıdır. Belirtilen kelimenin seçimi, bellek içinde adres çizgileriyle k bitlik ikili adrese başvurularak yapılır. Bellek içinde bir kod çözücü bu adresi kabul eder ve belirtilen kelimenin bitlerini seçerek yolu açmasını sağlar. Bilgisayar hafızası 1024 kelime olarak düzenlenebilir. Bu 10 bit adres gerektirir. 2^{32} kelime 32 bit adres gerektirir. Kelimelerin (veya byte ların) sayısı hafızada K (Kilo), M (Mega) veya G (Giga) harflerinden biri ile işaretlidir. K, 2^{10} a; M, 2^{20} ye ve G, 2^{30} a eşittir. Böylece 64 K = 2^{16}, 2M = 2^{21} ve 4G = 2^{32} olur.

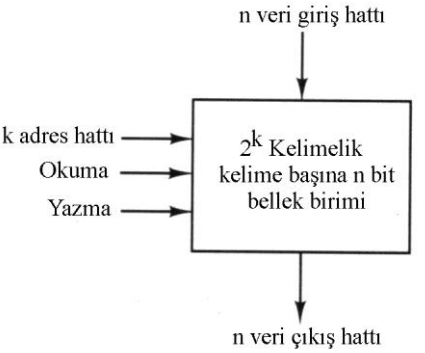
Bilgisayar sistemlerinde hafıza, RAM (adreslenebilir bellek) ve ROM (yalnız okunabilir bellek) şeklindedir.

Adreslenebilir Bellek (Rasgele Erişilebilir Bellek)

RAM

Rasgele erişilebilir bellekte bellek hücrelerine bilgi aktarımı için hangi adreste olurlarsa olsunlar erişilebilirler. Yani bellekte bir kelimeye erişim süreci, hücreler belleğin neresinde olursa olsun aynı zaman sürecini gerektirir. Dolayısıyla buna "rasgele erişilebilir" adı verilir.

Bellekle çevresi arasındaki iletişim giriş ve çıkış hatları ile olur. Ayrıca adres seçim hatlarına ve aktarımın yönünün bilinmesi için denetim hatlarına gerek vardır. RAM biriminin blok şeması şekil 2.12 de



Şekil 2.12 RAM ın blok şeması

gösterilmiştir. *n* tane veri giriş hattı bilgilerin belleğe saklanmasını, *n* tane veri çıkış hattı, bellekten istenen bilgilerin alınmasını sağlar. *k* tane adres hattı, *k* bitlik ikili bir sayı oluşturur. Bu bellekte mevcut 2^k tane bitlerin *k* tanesinden oluşan bir kelimedir. Aktarımın yönünü belirlemek için iki denetim girişi gerekir.

Yazma ve Okuma İşlemleri

Rasgele erişilebilir veya adreslenebilir belleğe uygulanabilecek iki işlem oku veya yaz işlemleridir. Yaz işlemi belleğe girişi, oku işlemi ise bellekten dışarı aktarımı gösterir. Bu denetim sinyallerinden birini kabul ederek, belleğin içindeki devrelerin istenen işlemi yapması sağlanır. Belleğe yeni bir kelime aktarıp, belleğe yazma işlemini yapması için aşağıdaki adımlar gerçekleştirilmelidir.

1. İstenen kelimenin ikili adresini adres yoluna yaz.
2. Belleğe yazılmasını istediğin veri bitlerini veri giriş yoluna yaz.
3. Yaz girişini çalıştır.

Bellek birimi, veri giriş hatlarında halen bulunmakta olan veri bitlerini alır ve adres hattı tarafından belirlenen adrese yazar.

Halen bellekte bulunan bir kelimenin bellekten dışarı taşınması için gerekli adımlar şunlardır.

1. İstenen kelimenin adresini adres yoluna yaz.
2. Oku girişini çalıştır.

Bellek birimi adres tarafından belirlenen kelimenin bitlerini alır ve veri çıkış yoluna aktarır. Okumadan sonra okunan kelimenin içeriği değişmez.

Yalnız Okunabilir Bellek

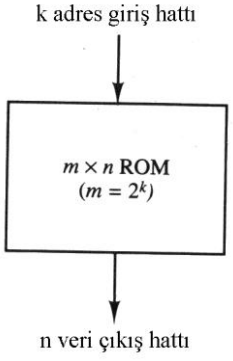
ROM

Adından anlaşılacağı gibi yalnız okunabilir bellek, sadece okuma işlemi yapan bellek birimidir. Yazma yeteneği yoktur. Bundan şu anlaşılır ki, ROM içindeki ikili bilgiler, birimin üretimi sırasında sabit olarak yazılır ve farklı kelimeler yazarak değiştirilemez. Halbuki RAM, genel amaçlı bir birim olup, bilgisayar işlemleri sırasında içeriği değiştirilebilir. ROM ise kısıtlı bir birim olup sabit olarak yerleştirilmiş bulunan kelimeleri okumaya yarar. ROM'un içine yazılacak ikili bilgiler, tasarımcı tarafından belirlenir. Sonra birimin içine, istenen iç bağlantı biçimini gerçekleştirecek şekilde yerleştirilir. ROM'lar içlerinde bazı elektronik sigortalarla yapılırlar. Böylece özel bir iş için programlanabilirler. Bir kere program yapılınca, bu program bilgisayar tekrar tekrar kapatılsa ve yeniden açılsa bile silinmez.

$m \times n$ bir ROM, m kelimeden oluşur. Her kelime n bittir. Şekil 2.13 te görüldüğü gibi ROM k bitle adreslenir. Yani $2^k = m$ kelimelik bir bellek olup n tane çıkışı vardır. Tümleşik devre şeklinde bir ROM'un bir veya daha fazla izin girişi vardır. Böylece birkaç tane ROM bir araya getirilebilir ve daha büyük bellekler elde edilebilir.

Bir ROM'da, oku denetimine gerek yoktur. Çünkü herhangi bir anda, adres tarafından seçilen kelimenin n biti çıkışa gider. ROM'da tüm çıkışlar girişlerin (burada k tane adres biti) fonksiyonu olduğundan tanım olarak ROM birleşik devredir. Gerçekte, ROM kod çözücüler ve VEYA kapılarından oluşur. Daha büyük saklama hacmine gerek yoktur. Çünkü RAM'ın aksine ROM daki kelimelerin hepsi değişmez (sabit) değerlerdir.

ROM'lar sayısal sistemlerin tasarımında geniş bir biçimde kullanılırlar. Temel olarak ROM doğruluk çizelgesi ile belirlenen bir giriş-çıkış bağıntısı üretir. Bunun anlamı, k girişi n çıkışı olan herhangi bir birleşik



Şekil 2.13 ROM un bellek yapısı

devre kurulabilir. Bilgisayarda bellek birimi olarak kullanıldığında, değişmeyen programlar veya sabitler çizelgesi gibi işler için kullanılır. ROM bilgisayarlarda denetim biriminin tasarım ve kurulumunda da kullanılır. Böylece, bilgisayarda muhtelif işlerin yapılması ve denetimi için gerekli, iç denetim değişkenlerinin sırasını ve değerlerini saklamakta kullanılır. Denetim biriminde, ikili denetim bilgilerinin saklanmasında ROM kullanılan bir bilgisayarın denetim birimine mikro programlanmış denetim denir. Bu konu ayrıntılı olarak 7. bölümde ele alınacaktır.

ROM Çeşitleri

ROM lar üç farklı şekilde programlanabilir. Bunlardan ilki, son üretim aşamasında yarı iletkenlik şirketi tarafından gerçekleştirilen maske programlanmasıdır. ROM un üretimi ile ilgili işlemler, onu isteyen müşterinin doldurduğu doğruluk çizelgesine göre yapılır. Doğruluk çizelgesi, satıcı firma tarafından, özel bir şekil veya özel bir biçim içeren bir bilgisayar çıkışı olarak verilebilir. Üretici müşterinin doğruluk çizelgesine göre 1 ler ve 0 lardan oluşan düzenek için istenilen maskeyi yapar. Bu işlem hayli pahalıdır, çünkü satıcı firma belirli bir ROM için yapılan maskeye özel bir ücret talebinde bulunur. Bu yüzden aynı ROM şekline sahip ROM lardan büyük sayıda sipariş edildiğinde daha ucuza mal olabilir.

PROM

Ucuza mal etmek ve az miktarda siparişler için ikinci çeşit ROM olan programlanabilir sadece okunabilen hafıza veya PROM isimli ROM kullanılabilir. PROM içerisindeki sigortalar her adresin çıkış terminaline, doğru akım dalgaları uygulaması ile attırılabilir. Atık sigorta ikili sistemde 0 durumunu, normal sigorta da 1 durumunu verir. Bu durum kullanıcıların depolanan kelimeler ile girilen adresler arasında istenilen ilişkiyi kurmak için kendi laboratuarlarında PROM ları programlamalarına izin vermektedir. Bu programlamayı kolaylaştırmak için PROM programlayıcıları olarak bilinen aletler ticari olarak satılmaktadır. Her ne durumda olursa olsun ROM ların programlanmasında gerekli olan işlemlerde her ne kadar "programlama" terimi kullanılsa da bu bir donanım işlemidir.

EEPROM

ROM ların veya PROM ların programlanması için kullanılan donanım işlemleri geri dönüşümsüzdür ve bir kez programlandığında belirtilen kalıp kalıcıdır ve değiştirilemez. Bir kez kısa kelime parçalarından oluşan kalıp kurulduğunda; eğer bu kalıp değiştirilmek isteniyorsa kalıbın bulunduğu tüm aksamın atılması gerekir. Üçüncü tip ROM ise silinebilen programlanabilir

ROM veya EPROM olarak adlandırılır. EPROM sigortaları daha önce atmış olsa bile tekrar başlangıç değeri verilebilir. EPROM sigortaları özel ultra-viyole ışık altında belli bir süre bırakılırsa, kısa dalga radyasyonu sigorta olarak kullanılan iş kapılarını devre dışı bırakır. Silinme işlemi bittikten sonra, EPROM başlangıç durumuna döner ve kelime gruplarına yeniden programlanabilir. Bazı PROM lar ultraviyole ışık yerine elektrik sinyalleri ile de silinebilir. Böyle PROM lar elektriksel olarak silinebilen EPROM veya kısaca EEPROM olarak adlandırılır.

SORULAR

2-1. TTL SSI entegreleri genelde 14 iğneli IC paketlerinde gelirler. 2 iğne güç bağlantısı için ayrılmıştır. Diğer iğneler giriş, çıkış uçları için kullanılmaktadırlar. Aşağıdaki devrelerden bir pakette kaç devre bulunmaktadır.

a. Tersleyen
b. 2 girişli ÖZEL-VEYA
c. 3 girişli VE-DEĞİL
d. 4 girişli VE
e. 5 girişli VEYA-DEĞİL
f. 8 girişli VE-DEĞİL
g. Saatten bağımsız silmeli, saatli *JK* yaz-bozu

2-2. MSI yongaları sayıcılar, kod çözücüler, seçiciler, yazaçlar gibi belirli basit sayısal fonksiyonları yerine getirirler. Bu fonksiyonları yerine getiren aşağıdaki devreler TTL tipi entegre devrelerdir. Bunların tanımlarını kataloglardan bularak bu bölümlerdekiler ile karşılaştırın.

a. IC 74155 ikili 2 × 4 kod çözücü
b. IC 74157 4 lü 2 × 1 seçici
c. IC 74194 4 bitlik paralel yüklemeli çift yönlü kaydırma yazacı
d. IC 74163 4 bitlik paralel yüklemeli saatli silmeli ikili sayıcı

2-3. Bir tane 2 × 4 , dört tane 3 x 8 lik kod çözücü kullanarak 5 × 32 lik kod çözücü oluşturun.

2-4. Sadece VEYA-DEĞİL kapılarını kullanarak izin girişi bulanan 2 × 4 lük kod çözücü yapın.

2-5. Şekil 2.2. deki kod çözücüye $E = 1$ olduğunda aktif $E = 0$ olduğunda pasif yapacak şekilde değiştirin. Doğruluk çizelgesini elde edin.

2-6. Doğruluk çizelgesi 2.2 de verilen 8 girişli 3 çıkışlı kodlayıcıyı çizin. Tüm girişler 0 olduğunda çıkışlar ne olur. Sadece D_0 girişi sıfır olduğunda çıkış ne olur belirtin.

2-7. 2 tane 8 × 1 lik, 1 tane 2 × 1 lik seçici kullanarak 16 × 1 lik seçici yapınız. Tüm seçiciler blok şema şeklinde olsun.

2-8. 2 tane 4 × 1 lik seçicinin blok şemasını çizip fonksiyonu çizelgesini oluşturun.

2-9. Şekil 2.6 daki yazaç devresinin saatine bir VE kapısının çıkışını bağlayın. VE kapısının bir girişi saatten diğer girişi de yükleme denetiminden gelsin.

2-10. Şekil 2.7 deki yazacın saat girişindeki buffer kapısının görevi nedir?

2-11. Şekil 2.7 deki paralel yüklemeli yazaca saatli silme özelliği ekleyin.

2-12. Başlangıç olarak 4 bitlik yazacın içeriği 1101 dir. Yazaç sağa doğru seri giriş 101101 alarak 6 kez kaydırılsın. Her kaydırmadan sonra yazacın içeriği nedir?

2-13. Seri ve paralel geçiş arasındaki fark nedir? Paralel yüklemeli kaydırma yazacı kullanarak seri giriş bilgisinin paralel çıkışa ve paralel giriş bilgisinin seri çıkışa nasıl dönüştürüleceğini açıklayınız.

2-14. Bir halka sayıcı şekil 2.8 deki gibi seri çıkışı seri girişe bağlayan bir kaydırma yazacıdır. 1000 başlangıç durumundan her kaydırmadan sonra 4 yaz-bozun durumlarının sırasını listeleyin.

2-15. Şekil 2.9 da gösterilen 4 bitlik çift yönlü paralel yüklemeli kaydırma yazacı bir IC paketine kapatılır.

a. IC paketinin bütün giriş ve çıkışlarını gösteren blok şemasını çizin. 2 tane güç ihtiyacını karşılayan uç içersin.

b. 8 bitlik çift yönlü paralel yüklemeli kaydırma yazacı oluşturmak için 2 IC kullanarak blok şemasını çizin.

2-16. 10 bitlik ikili sayıcıda (a) 1001100111 (b) 001111111 dan bir sonraki saymada kaç tane yaz-boz terslenir.

2-17. Şekil 2.11 deki 4 bitlik ikili sayıcıları kullanarak 16 bitlik paralel yüklemeli ikili sayıcı elde ediniz. 4 bitlik ikili sayıcılar için blok şema kullanınız.

2-18. Paralel yüklemeli bir ikili sayıcı nasıl *N* ile bölünen sayıcı olarak iş görür gösteriniz. (0000 dan *N* e sayan ve geri 0000 olan sayıcıdır) Özel olarak şekil 2.11 deki sayıcıyı ve bir harici VE kapısını kullanarak 10 ile bölünün sayıcı devresini gösteriniz.

2-19. Aşağıdaki bellek üniteleri kelime sayısı ve her kelimedeki bit sayısı olarak verilmektedir. Her bir durumda ne kadar adres ve veri hattı gerekmektedir (a) 2K × 16, (b) 64K × 8, (c) 16M × 32, (d) 4G × 64

2-20. Soru 2.9 için bellekte ne kadar byte depolanır belirleyiniz.

2-21. 4096×16 bellek kapasitesi oluşturmak için kaç tane 128×8 bellek yongasına ihtiyaç vardır.

2-22. İzin girişli 32×8 ROM yongası veriliyor. 4 yonga ve bir kod çözücü ile 128×8 ROM oluşturmak için gerekli dış bağlantıları gösteriniz.

KAYNAKLAR

1. Hill, F. J., and G.R. Peterson, *Introduction to Switching Theory and Logical Desing*, 3rd ed. New York: John Wiley , 1981.
2. Mano, M. M., *Digital Desing*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1991.
3. Roth, C. H., *Fundamentals of Logic Desing*, 3rd ed. St. Paul, MN: West Publishing, 1985.
4. Sandige , R. S., *Modern Digital Desing*. New York: McGraw-Hill, 1990.
5. Shiva, S. G., *Introduction to Logic Desing*. Glenview, II: Scott, Foresman, 1988.
6. Wakerly, J. F., *Digital Desing Principles and Practices*. Englewood Cliffs, NJ: Prentice Hall, 1990.
7. Ward. S. A., and R.H. Halstead, Jr., *Computer Structures*. Cambridge, MA: MIT Press, 1990.

BÖLÜM 3

Veri Gösterimi

BU BÖLÜMDE

3-1 Veri Tipleri
3-2 Tümleyenler
3-3 Sabit Nokta Gösterimi
3-4 Kayan Noktalı Gösterim
3-5 Diğer İkili Kodlar
3-6 Hata Bulma Kodları

3.1 Veri Tipleri

İkili bilgiler bilgisayar içinde bellek veya işlemci yazaçlarında saklanırlar. Yazaçlar veri ve denetim bilgileri içerirler. Denetim bilgisi bir bit veya bir grup bittir. Buyruk işaretlerinin sırasını belirler. Bunlar diğer veri yazaçlarındaki verilerin işlenmesinde kullanılırlar. Veriler sayılar veya diğer ikili olarak kodlanmış bilgilerdir. Bunlar işlenerek istenen sonuçlara ulaşılır. Bu bölüm sayısal bilgisayarlarda en çok ortak olan veri tiplerini ve muhtelif veri tiplerinin nasıl ikili kodlanmış biçimde saklandıklarını anlatacaktır.

Sayısal bilgisayarlarda bulunan veri tipleri aşağıdaki sınıflardan birine girer. (1) Aritmetik işlemlerde kullanılan sayılar, (2) veri işlemlerinde kullanılan alfabenin harfleri ve (3) özel amaçlarla kullanılan diğer ayrık semboller. İkili sayılar hariç tüm veri tipleri bilgisayar yazaçlarında ikili kodlanmış olarak bulunurlar. Çünkü yazaçlar yaz-bozlardan yapılmıştır ve yaz-bozlar ise 2 durumlu birimler olup yalnız 1 ve 0 saklayabilirler. Buna karşılık ikili sayılar en doğal sistemdir. Fakat bazen farklı sayı sistemleri özellikle ondalık sistem kullanmak gerekebilir. Çünkü aritmetik işlemler ondalık sistemle yapılır.

Sayı Sistemleri

Taban, Ondalık

Bir sayı sisteminin *tabanı r* dir. Farklı *r* tane rakam kullanır. Sayılar rakamların bir sırası ile ifade edilir. Her bir rakam *r* nin bir tamsayı üssü ile çarpılır. Bu da bulunduğu basamağın sayısıdır. Her zaman kullanılan 10 tabanının sembolleri 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 dur. 724,5 in büyüklük olarak gösterimi

$$7 \times 10^2 + 2 \times 10^1 + 4 \times 10^0 + 5 \times 10^{-1}$$

şeklindedir. Her zaman ondalık sayı bu şekilde gösterilebilir.

İkili

İkili sayı sistemleri 2 tabanını kullanırlar. İki rakamlı 0 ve 1 olmak üzere

$$(101101)_2 = (1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0) = (45)_{10}$$

Farklı tabandaki sayıları birbirinden ayırmak için rakamlar parantez işaretleri arasına yazılmakta ve parantezin sağ alt köşesine taban değeri yazılmaktadır. Örneğin 45 sayısı ikili ve ondalık sistemde $(101101)_2 = (45)_{10}$ olur.

Sekizlik, Onaltılık

Ondalık ve ikili sayı sistemlerinden başka sekizlik ve onaltılık sayı sistemleri de sayısal bilgisayarların çalışmalarında önemlidir. Sekizlik sistemlerin 8 sembolü 0, 1, 2, 3, 4, 5, 6, 7, dir. Onaltılık sistemlerin 16 sembolü 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F dir. Bunlardan son 6 sembol harftir. 10, 11, 12, 13, 14, 15 sayıları sırasıyla A, B, C, D, E, F harfleriyle gösterilmektedir. Sekizlik 736.4 sayısı ondalık sayıya dönüştürülmek istenirse

$$(736.4)_8 = 7 \times 8^2 + 3 \times 8^1 + 6 \times 8^0 + 4 \times 8^{-1}$$

$$= 7 \times 64 + 3 \times 8 + 6 \times 1 + 4 / 8 = (478.5)_{10}$$

Onaltılık F3 sayısı da ondalığa aynı şekilde dönüştürülebilir.

$$(F3)_{16} = F \times 16 + 3 = 15 \times 16 + 3 = (243)_{10}$$

Dönüşüm

Ondalık sayı 0 a denk herhangi bir *r* tabanındaki bir sayıya dönüştürülürken tam sayılar ve gerçel sayılar farklı şekilde dönüştürülür. Tamsayılarda veya gerçel sayıların tam kısmında taban dönüşümü aynı şekilde yapılır. Ondalık sayı taban değerine bölünerek (bölme işleminde yuvarlama ve ondalık bölme yapılmaz) çıkan değer tekrar taban değerine bölünür. Bu bölme işlemi sonuç değer taban değerinden küçük oluncaya kadar devam eder. Şekil 3.1 de bu aşamalar gösterilmektedir.

Ondalık 41.6875 sayısı ikiliye dönüştürülürken tam kısmı 41 ve ondalık kısmı .6875 ayrılır. Tam kısım 41 sayısı $r = 2$ ye bölünerek bölüm 20 ve kalan 1 elde edilir. Bölüm tekrar 2 ye bölünerek yeni bölüm

Şekil 3.1 Ondalık 41.6875 sayısının ikiliye dönüşümü

Tam kısım = 41		Ondalık kısım = 0.6875
41		0.6875
20	1	x 2
10	0	1.3750
5	0	x 2
2	1	0.7500
1	0	x 2
0	1	1.5000
		x 2
		1.0000

(41)10 = (101001)2 (0.6875)10 = (1011) 2

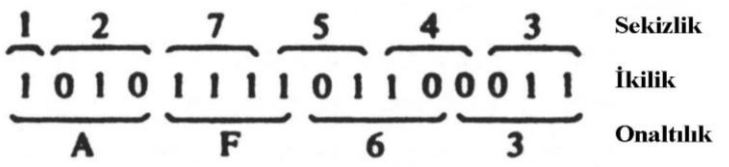
(41.6875) 10 = (101001.1011) 2

ve kalan elde edilir. Bu işlemler bölüm 0 kalana kadar tekrar edilir. Ondalık sayıların ikili karşılıkları ilk kalan ikili sayının en önemsiz bitine yazılarak elde edilir.

Ondalık kısımda taban değişikliği olduğunda ondalık kısım alınır. Taban ile çarpılır ve bu çarpma işleminde tam kısımdaki değer taban değerinden büyükse bu sayıdan taban değeri çıkarılarak yazılır. Bu işlem kaç adım yapılırsa duyarlılık o kadar artar. Tam kısım 1 kaldığında ondalık sayının ikili karşılığı elde edilmiş olur ve buradaki tam kısımdaki değerler yazılır.

Sekizlik ve Onaltılık Sayılar

Sekizlik ve onaltılık gösterimden ikiliye, ikiliden sekizlik ve onaltılığa dönüşüm sayısal bilgisayarların önemli bir bölümünü içerir. $2^3 = 8$ ve $2^4 = 16$ dan her bir sekizlik rakam 3 ikili rakamla, her bir onaltılık rakamda 4 ikili rakamla gösterilebilir. İkili sayıların sekizlik sayılara dönüşümü kolayca her bir üç bit bir gruba alınarak yapılır. Bitlerin her bir grubu bir sekizlik sayıyı göstermektedir. Böylece ikili sayıların sekizlik denklikleri elde edilmiş olur. Örnek olarak 16 bitlik bir yazaç göz önüne alalım. Fiziksel olarak bu yazaç 16 adet 0 yada 1 değerlerini tutan depo hücrelerinden oluşmakta. Bit konfigürasyonları yazaçlara şekil 3.2 deki gibi yüklü olsun. 16 bit yazaç 0 ve 1 ikili sayılarından oluşan 0 dan 2^{16}-1 kadar farklı sayıyı depolayabilir. Örnekte ikili olarak verilen sayı ondalık 44899 sayısına denktir. Yazaçtaki sayıyı guruplara ayırırken en düşük bitten başlanarak guruplanır. İkili sayı sekizliğe çevrilirken üçerli bit gurupları alınır (16.bit tek kalmıştır). Bu gurupların her birinin sekizlik karşılıkları yazılır.



Şekil 3.2 İkili, sekizli ve onaltılık dönüşüm

İkiliden onaltılığa dönüşümde de bitlerin (en düşük mertebedeki bitten başlayarak) dörderli gurupları alınır ve her bit gurubunun onaltılık karşılıkları yazılarak yapılır. İkili sayıların sekizlik karşılıkları ilk sekiz giriş olarak çizelge 3.1 de, ikili sayıların onaltılık karşılıkları ilk 16 giriş olarak çizelge 3.2 de verilmektedir.

Çizelge 3.1 ikili kodlanmış sekizlik sayıların yazaçlardaki gösterimini listelemektedir. İkili kodların elde edilmesi işlemi yukarıda açıklanmıştı. Benzer şekilde çizelge 3.2 de birkaç onaltılık sayıyı ve bu sayıların yazaçlardaki ikili kodlanmış onaltılık halini göstermektedir.

Sayıların ikili denkliği ile ikili kodlanmış sekizlik ve ikili kodlanmış onaltılık sayılar karşılaştırıldığında bit kümeleri için 3 gösterim şekli olduğu kolayca anlaşılabilir. Örneğin ondalık 99 sayısı ikiliye dönüştürüldüğünde 1100011 elde edilir. 99 sayısının ikili kodlanmış sekizlik karşılığı 001 100 011 ve 99 ondalık sayısının ikili kodlanmış onaltılık karşılığı da 0110 0011 dir. Üç ikili gösterimdeki en sondaki sıfırlar ihmal edilirse, üç ikili gösteriminde birbirine denk oldukları görülür. Bir dizgi

Çizelge 3.1 İkili kodlanmış sekizlik sayılar

Sekizlik sayı	İkili kodlanmış sekizlik	Ondalık denkliği	
0	000	0	↑
1	001	1	
2	010	2	Bir
3	011	3	sekizlik
4	100	4	rakam için
5	101	5	kod
6	110	6	
7	111	7	↓
10	001 000	8	
11	001 001	9	
12	001 010	10	
24	010 100	20	
62	110 010	50	
143	001 100 011	99	
370	011 111 000	248	

Çizelge 3.2 İkili kodlanmış onaltılık sayılar

Onaltılık sayı	İkili kodlanmış onaltılık	Ondalık denkliği	
0	0000	0	
1	0001	1	
2	0010	2	
3	0011	3	
4	0100	4	
5	0101	5	
6	0110	6	
7	0111	7	Onaltılık bir rakam için kod
8	1000	8	
9	1001	9	
A	1010	10	
B	1011	11	
C	1100	12	
D	1101	13	
E	1110	14	
F	1111	15	
14	0001 0100	20	
32	0011 0010	50	
63	0110 0011	99	
F8	1111 1000	248	

yazaçlarda 1 ve 0 lardan oluşan ikili gösterimde depolanır. Bu dizgi üçerli guruplara ayrıldığında ikili kodlanmış sekizlik, dörderli guruplara ayrıldığında ikili kodlanmış onaltılık şeklinde saklanmış olur.

Bir sayısal bilgisayardaki yazaçlar pek çok bit içerir. Yazaçların içeriklerinin ikili değer ile belirtilmesi uzun bir ikili dizgi oluşturur. Yazaç içeriğinin sekizlik yada onaltılık olarak belirtilmesi daha uygun olmaktadır. Örneğin 1111 1111 1111 ikili sayısı 12 rakama sahiptir. Sekizliğe dönüştürüldüğünde 7777 olur ve dört rakam yer kaplar, onaltılık de ise FFF olur ve 3 rakam yer kaplar.

Ondalık Gösterim

İkili sayı sistemleri sayısal bilgisayarlarda kullanılan sayı sistemleridir. Fakat insanlar ondalık sistemlere alışıktır. Bu karışıklığın giderilmesi için bir yol ondalık sayıların ikili sayılara dönüşümüdür. Bilgisayarda ikili sayılarla tüm matematiksel işlemler yapılabilir ve sonra ikili sonuçlar

ondalık sayılara dönüştürülür. Ondalık sayılar yazaçlarda kendi biçimlerinde saklandıklarında direkt olarak ondalık sayılarla matematiksel işlem yapılabilir. Ondalık sayılar bilgisayarlara BCD ler gibi alfasayısal karakterler olarak aktarılabilir. Bu kodlar harflerle gösterilmekte ve her bir ondalık rakam için 6-8 bit kullanılmakta. Ondalık sayı dahili matematiksel işlemlerde kullanıldığında her bir rakam 4 bitlik ikili koda dönüştürülmektedir.

İkili Kod, BCD

Bir ikili kod *n* bitlik bir guruptur. Kümenin elemanları 0 ve 1 lerin birleşiminden oluşan *n* farklı değer içerir. Örneğin 4 elemanlı bir küme (00, 01, 10, 11) 2 bitten oluşur. 8 elemanlı bir küme için 3 bit koda, 16 elemanlı bir küme için 4 bit koda ihtiyaç vardır. Ve bu böyle devam eder. Bir küme içindeki elemanların sayısı 2 nin kuvvetine denk değilse o gurupta bazı kullanılmayan bit birleşimleri vardır. 10 ondalık rakamın bulunduğu bir küme alındığında bu küme içindeki birbirinden farklı on eleman en az 4 bit ile gösterilir. 6 adet 4 bitlik birleşim dışarıda kalmıştır. 0 dan 9 a kadar ondalık sayıların ikili karşılıklarına ikili-kodlanmış-ondalık veya BCD denir. Diğer ondalık kodlar bölüm 3.5 te anlatılmaktadır.

İkili kodların ondalık sayılara, ondalık sayıların ikili koda dönüşümü çok önemlidir. Örneğin 99 ondalık sayısı bir ikili sayıya dönüştürüldüğünde 1100011 olur. Fakat BCD'ye dönüştürüldüğünde 1001 1001 elde edilir. Çizelge 3.3 te bir kaç ondalık sayı ve onların BCD'deki karşılıkları gösterilmektedir.

Çizelge 3.3 BCD sayılar

Ondalık sayı	BCD sayı			
0			0000	↑
1			0001	
2			0010	
3			0011	
4			0100	Bir ondalık
5			0101	rakam için
6			0110	kod
7			0111	
8			1000	
9			1001	↓
10		0001	0000	
20		0010	0000	
50		0101	0000	
99		1001	1001	
248	0010	0100	1000	

Alfasayısal Gösterim

Karakter, ASCII

Sayısal bilgisayar sistemlerinin bir çok uygulamalarında sayıların, harflerin yanında özel karakterlere de ihtiyaç vardır. Alfasayısal küme 10 ondalık rakamı, 26 harfi, $,+,= gibi özel karakterleri içermektedir. Bir küme sadece büyük harfleri içerdiğinde 32,...,64 arasında eleman içermekte, yada küçük veya büyük harfleri içerdiğinde de 64,...,128 arasında eleman içermektedir, birinci durumda 6 bite, ikinci durumda 7 bite ihtiyaç vardır. Standard alfasayısal ikili kodlar yani ASCII (American Standard Code for Information Interchange) 128 karakter kod 7 bitle belirlenmektedir. Çizelge 3.4 te büyük harfler, ondalık rakamlar bir kaç özel karakter gösterilmektedir. Not edilirse ondalık rakamlar ASCII ye BCD den dönüştürülürken en önemli 3 bit 011 alınmaktadır.

Çizelge 3.4 ASCII karakterleri

Karakter	İkili Kod	Karakter	İkili kod
A	100 0001	0	011 0000
B	100 0010	1	011 0001
C	100 0011	2	011 0010
D	100 0100	3	011 0011
E	100 0101	4	011 0100
F	100 0110	5	011 0101
G	100 0111	6	011 0110
H	100 1000	7	011 0111
I	100 1001	8	011 1000
J	100 1010	9	011 1001
K	100 1011		
L	100 1100	Space	010 0000
M	100 1101	.	010 1110
N	100 1110	(	010 1000
O	100 1111	+	010 1011
P	101 0000	$	010 0100
Q	101 0001	*	010 1010
R	101 0010	)	010 1001
S	101 0011	-	010 1101
T	101 0100	/	010 1111
U	101 0101	,	010 1100
V	101 0110	=	011 1101
W	101 0111		
X	101 1000		
Y	101 1001		
Z	101 1010		

İkili kodlar sayısal bilgisayar işlemlerinin en önemli bölümünü tutar. Çünkü yazaçlar sadece ikili bilgileri tutabilmektedir. İkili kodların, sembollerin simgeledikleri ayrık elemanların anlamını değil, yalnızca sembollerin kendisini değiştirdiği anlaşılmalıdır. Sayısal bilgisayarlar için tanımlanmış işlemler yazaçlarda depolanmış bitlerin anlamlarını dikkate almak zorundadırlar. Böylelikle işlemlerin aynı tipte işlenenler üzerinde yapılması sağlanır. Bir bilgisayar yazacının bitlerini rasgele gözden geçirirken, yazacın bir ikili sayıyı değil belirli bir tipte kodlanmış bilgiyi içerdiğini görmek olasıdır.

İkili kodlar müzik notaları, satranç taşları ve satranç tahtası üzerindeki konumları gibi herhangi bir ayrık elemanlar kümesi için formüle edilebilir. İkili kodlar bilgisayar için denetim bilgisini tanımlayan buyrukları ifade etmek için de kullanılır. Bu bölüm veri gösterimi ile ilgili olacaktır. Buyruk kodları bölüm 5 te anlatılacaktır.

3.2 Tümleyenler

Tümleyenler sayısal bilgisayarlarda çıkarma işlemleri ve mantıksal işlemlerin yapılmasını basitleştirir *r* tabanına göre iki tür tümleyen vardır. *r* ye göre tümleyen ve (*r-1*) e göre tümleyen olmak üzere taban *r* ile gösterildiğinde ikili sayılarda *r ye göre tümleyen* 2 ye göre tümleyen, *r-1* e göre tümleyen 1 e göre tümleyen olarak isimlendirilir. Ondalık sayılarda *r ye göre tümleyen* 10 un tümleyeni *r-1* e göre tümleyen 9 un tümleyeni olarak isimlendirilir.

(r-1)'e Göre Tümleyen

9'a Göre Tümleyen

r tabanında *n* rakamlık bir *N* sayısının (*r – 1*) e göre tümleyeni alınırken $(r^n - 1) - N$ den hesaplanır. Örneğin $n = 4$ alındığında $(r^n - 1) = (10^4 - 1) = 10000 - 1 = 9999$ bulunur. Burada *n* sayıda 9 bulunmaktadır. Bir ondalık sayının 9 a göre tümleyeni her bir terim 9 dan çıkarılarak elde edilir. Örneğin 546700 ın tümleyeni 999999 – 546700 = 453299 ve 12389 un 9 a göre tümleyeni 99999 – 12389 = 87610 dur.

1'e Göre Tümleyen

İkili sayılarda $r = 2$ için $r - 1 = 1$ dir. *N* sayısı için 1 e göre tümleyen $(2^n - 1) - N$ dir. 2^n *n* adet 0 ve bu 0 ları takip eden 1 sayısı ile gösterilir. *N* adet 1 lerden oluşan ikili sayı $2^n - 1$ in gösterimidir. Örneğin $n = 4$ ile $2^4 = (10000)_2$ ve $2^4 - 1 = (1111)_2$ yazılır. İkili sayılarda 1 e göre tümleyen 1 den her bir rakamın çıkarılması ile elde edilebilir. Örneğin 1011001 sayısının 1 e göre tümleyeni 0100110 dır. 0001111 inde 1110000 dır.

(*r* – 1) e göre tümleyen sekizlik sayılarda 7 den çıkarılarak onaltılık sayılarda F ten (ondalık 15 ten) çıkarılarak elde edilir.

r' ye Göre Tümleyen

10'a Göre Tümleyen

r tabanındaki bir *N* rakamlık *N* sayısının *r* ye göre tümleyeni $N \neq 0$ olduğunda $r^n - N$ ile tanımlıdır. *r* ye göre tümleyen $(r - 1)$ e göre tümleyene 1 eklenerek bulunur. Çünkü $r^n - N = [(r^n - 1) - N] + 1$ dir. 2389 ondalık sayısının 10 a göre tümleyeni 9 a göre tümleyenine 1 eklenerek bulunur. 7610 + 1 = 7611 dir. İkili sayılarda 101100 ın 2 ye göre tümleyeni 010011 + 1 = 010100 dır. Buda 1 e göre tümleyene 1 eklenerek bulunur.

2'ye Göre Tümleyen

N sayısının 10 a göre tümleyeni ($10^n - N$ olduğundan) 10^n den dolayı *n* adet 0 ve en sola 1 yazılarak bu sayıdan *N* sayısı çıkarılarak bulunur. Bir sayının 10 a göre tümleyeni sağdan başlanarak 0 lar varsa aynen 0 olmayan birinci en önemli rakam 10 dan ve ondan sonraki tüm yüksek önemli rakamlar 9 dan çıkarılarak bulunur. Benzer olarak ikili sayılarda bir sayının 2 ye göre tümleyeni alınırken en düşük önemli bitten başlayarak sıfırlar aynen alınır 1 değeri de aynen yazılır. 1101100 ın 2 ye göre tümleyeni 0010100 dır ve en düşük önemli 2 0 bit ve 1 (0 ların yanındaki) bit aynen diğer bitlerde de 0 lar yerine 1,1 ler yerine 0 yazılır.

Yukarıdaki tanımlarda sayıların bir taban noktasına sahip olmadıkları varsayılmıştır. Eğer orijinal *N* sayısı bir taban noktası içeriyorsa *r* yada *r* –1 e göre tümleyenini oluşturmak için geçici olarak yok edilmelidir. Taban noktası, daha sonra tekrardan tümlenen sayıya aynı göreceli konumda eklenir. Bir sayının tümleyeninin tümleyeni sayının kendisini vermektedir. *N* in *r*'ye göre tümleyeni $r^n - N$ dir. Tümleyeninin tümleyeni, ilk sayıyı geri veren, $r^n - (r^n - N) = N$ dir.

İşaretsiz sayılarda çıkarma

İlkokullarda öğretilen doğrudan çıkarma yöntemi ödünç alma kavramını kullanır. Bu yöntemde, kendisinden çıkarma yapılacak rakam çıkarılacak rakamdan küçük olduğu zaman bir yüksek basamaktan bir "1" ödünç olarak alınır. Bu, insanlar çıkarma işlemini kağıt ve kalem ile yaptıkları zaman en kolay yöntem gibi görünmektedir. Çıkarma işlemi sayısal donanım ile gerçekleştirildiği zaman, bu yöntemin tümleme işlemini kullanan yöntemden daha az verimli olduğu görülmektedir.

Çıkarma

Bilgisayarda *r* tabanındaki *n* rakamlık iki işaretsiz sayının çıkarma işlemi $M - N$ $(N \neq 0)$ şeklinde gösterilir. Aşağıdaki yolla yapılır.

1. Eksiltilen *M* ile çıkartılan *N* in *r* ye göre tümleyeni toplanır. Bu işlem $M + (r^n - N) = M - N + r^n$ dir.
2. Eğer $M \geq N$ ise elde toplamdan *r* çıkarılarak çıkarma sonucu $M - N$ bulunur.

3. Eğer $M < N$ ise toplam üzerinde son elde ile ilgili işlem yapılmaz. Çıkan toplamın r ye göre tümleyeni alınır ve sonuç bunun negatif değerine eşittir.

Bir örnek olarak 72532 – 13250 = 59282 dir. Bunu tümleyen ile yaparsak 13250 nin 10'a göre tümleyeni 86750 dir. Bu yüzden;

M =	72532
N in 10'a göre tümleyeni =	+ 86750
Toplam =	159282
10^5 son elde atmak için =	– 100000
Cevap =	59282

bulunur.

Şimdi $M < N$ olan bir örnek alalım normalde 13250 – 72532 işleminin sonucu negatif 59282 dir. Bunu tümleyen işlemini kullanarak yaparsak

M =	13250
N in 10'a göre tümleyeni =	+ 27468
Toplam =	40718

son elde yoktur. Cevap negatif 59282 = 40718 in 10'a göre tümleyenidir.

İşaretsiz sayılar ile uğraşıldığında, ikinci örnek için gerçekten işaretsiz bir sonuca ulaşmak mümkün değildir. Kalem ve kağıt ile çalışırken, sonucun işaretli işaretli bir negatif sayıya dönüştürülme zorunluluğu anlaşılmaktadır. Tümleme ile çıkarma işlemi yaparken, negatif sonuç son elde nin olmamasından ve tümlenen sonuçtan elde edilmektedir.

Aynı yöntem ikili sayılardaki çıkarma işleminde de kullanılır. Örneğin iki ikili sayı X = 1010100 ve Y = 1000011 alınsın. 2 ye göre tümleyen kullanarak $X - Y$ ve $Y - X$ bulunursa

X =	1010100
Y nin 2'ye göre tümleyeni =	+ 0111101
Toplam =	10010001
2^7 son elde atıldığında =	– 10000000
Cevap: $X - Y$ =	– 10000000

Y =	1000011
X in 2'ye göre tümleyeni =	+ 0101100
Toplam =	1101111

son elde yoktur. Toplamın 2'ye göre tümleyeninin negatifi alınırsa Cevap 0010001 bulunur.

3.3 Sabit Nokta Gösterimi

Sıfır da dahil olmak üzere pozitif tamsayılar işaretsiz sayılar gibi yazılabilir. Buna rağmen negatif sayıları göstermek için gösterime ihtiyaç vardır. Normal matematikte pozitif sayılar + negatif sayılar – ile belirtilir. Bilgisayarda her şey 0 ve 1 lerle gösterildiğinden işarette sayı ile belirtilmektedir. Bu yüzden en soldaki bit işaret biti olarak alınır. Bu bit 0 ise sayı pozitif 1 ise sayı negatiftir.

İkili nokta

İkili veya ondalık noktalı sayılar işaretleri ile toplanır. Tamsayıların, kesirli sayıların, tam veya kesirli karışık sayıların gösterilmesinde ikili noktanın yerine ihtiyaç vardır. Bir yazaçtaki ikili noktanın yerini kesinlikle göstermek için iki yol vardır. Bir sabit pozisyon verilerek yada bir kayan nokta gösterimi kullanarak sabit nokta yönteminde ikili noktanın her zaman sadece bir sabit pozisyonun olduğu varsayılır. Nokta ya (1) en sola veya (2) en sağa konur. Böylece yeri sabit olur. Aslında nokta yoktur. Sadece var olduğu varsayılır. Kayan noktalı sayıları saklamak için iki yazaç kullanılmaktadır. Ondalık noktanın ilk yazaçta olduğu kabul edilir. Kayan noktalı sayılar daha sonraki bölümde anlatılacaktır.

Tamsayı Gösterim

İşaretli Sayılar

Bir ikili tamsayı pozitif olduğunda işareti 0 negatif olduğunda ise işareti 1 ile gösterilir. Bir sayının 3 farklı gösterimi olabilir.

1. İşaretli normal ikili
2. İşaretli 1 e göre tümleyen
3. İşaretli 2 ye göre tümleyen

Negatif sayıların işaretli normal gösterimi bir eksi işareti ve sayının büyüklüğünden oluşur. Pozitif sayılar sadece 1 inci şekilde negatif sayılar her üç şekilde de gösterilebilir. Örneğin 8 bit yazaçta 14 depolanmak istendiğinde +14 en soldaki işaret biti 0 olacak şekilde 00001110 şeklinde bulunur. 8 inci bitin sayının büyüklüğü üzerine hiç bir etkisi yoktur. Sadece işareti gösterir. -14 sayısı 8 bitte 3 farklı şekilde gösterilebilir.

İşaretli normal ikili	1 0001110
İşaretli 1 e göre tümleyen	1 1110001
İşaretli 2 ye göre tümleyen	1 1110010

İşaretli gösterimde –14 değeri +14 değerinin sadece işaretinin tümleyeni alınarak elde edilir. –14 sayısının işaretli 1'e göre tümleyeninde, işaret dahil tüm bitlerin 1'e göre tümleyeni veya işaretli 2'ye göre tümleyende

işaret dahil tüm bitlerin 2 ye göre tümleyeni alınarak +14 sayısı elde edilir.

Normal matematikte negatif sayılar için işaretli büyüklük gösterimi uygundur. Fakat bilgisayar aritmetiğine uygulandığında kullanılması zordur. 1'e göre tümleyende 0, +0 ve –0 olmak üzere iki şekilde gösterilmektedir. Bu yüzden 1'e göre tümleyen bazı eski bilgisayarlarda ikili aritmetik işlemlerde kullanılmıştır. Bilgisayarlarda ikili aritmetik işlemlerde negatif sayıları göstermek için 2'ye göre tümleyen kullanılır.

Aritmetik Toplama

2'ye Göre Tümleyen İle Toplama

İşaretli iki sayının toplamı ile ilgili kurallar normal matematikte verilmektedir. İki sayının işareti aynı ise sayılar toplanarak çıkan sonuca ortak işaret konur. Eğer işaretler farklı ise değer olarak daha büyük olandan küçük çıkarılıp sonuca büyüğün işareti verilir. Örneğin (+25) + (–37) = –(37 – 25) = 12 olur. Yani iki sayıdan biri negatif ise mutlak değerce büyük olandan küçük olanın çıkarılması ve büyük olanın işaretinin verilmesi gerekir. Dolayısıyla işaret karşılaştırması gerekir. Eğer işaretler aynı ise değerler toplanır eğer işaretler farklı ise yukarıdaki işlem yapılır ve sonucun işaretinin de belirlenmesi gerekir. Bu durum karışık devreleri ve karışık denetim işlemlerini gerektirir. Bu olay 2'ye göre tümleyen ile çok basittir. Bu durumda kıyaslama ve çıkarma işlemine gerek olmayıp sadece toplama ve tümleyen alma işlemine gerek vardır. Bu işlem işaret bitleriyle birlikte iki sayının toplanması ve eğer en önemli bit (işaret biti) ten sonra bir elde varsa atılır. Toplama işlemi için verilen örneklerde negatif sayılar ve çıkan negatif sonuçlar 2'ye göre tümleyen şeklindedir.

+6	00000110	–6	11111010
+13	00001101	+13	00001101
+19	11110011	+7	00000111
+6	00000110	–6	11111010
–13	11110011	–13	11110011
–7	11111001	–19	11101101

4 durumun her birinde işlem toplama işlemidir. Negatif sayılar 2'ye göre tümleyeni olarak tanımlandığında onu pozitif sayıya dönüştürmek mümkündür. Örneğin 11111001 sayısı ilk bit 1 olduğundan negatiftir. Bu sayı 00000111 in 2'ye göre tümleyenidir. Bu sayı +7 olduğundan negatif sayı –7 dir.

Aritmetik Çıkarma

2'ye Göre Tümleyen İle Çıkarma

İşaretli iki sayı eğer işaret bitleriyle birlikte 2'ye göre tümleyeni olarak verilmişlerse çıkarılması kuralı: çıkanın, 2'ye göre tümleyeni alıp (işaret biti dahil) bunu işaret biti dahil çıkarılana eklemek. Toplamda işaret bitinin solunda elde oluşuyorsa atılır.

Bu işlemin toplamadan farkı yoktur. Yani

$$(\pm A) - (+ B) = (\pm A) + (- B)$$
$$(\pm A) - (- B) = (\pm A) + (+B)$$

olur. Bu işaretli toplamadır. Bir pozitif sayı bir negatif sayıya 2'ye göre tümleyen kullanılarak kolayca dönüştürülebilir. Bu dönüşüm tamamen doğrudur. Çünkü bir negatif sayının tümleyeni pozitif dengini vermektedir. Örneğin (–6) – (–13) = +7 İkili 8 bit kullanarak bu sayılar yazılırsa 11111010 – 11110011 olur. (–13) ün 2 ye göre tümleyeni (+13) olmaktadır. Çıkanın 2'ye göre tümleyeni alınıp çıkarılana eklendiğinde 11111010 + 00001101 = 100000111 işaret bitinden sonraki bit atıldığında doğru cevap 00000111 (+7) bulunur.

Taşma

Taşma

n rakamlı iki sayının toplamı $n + 1$ rakamlıdır. Bu durumda "taşma" meydana gelir. Bu durum işaretli veya işaretsiz tüm sayılarda ortaya çıkar. Elle yapılan işlemlerde taşma sorun olmaz. Fakat sayısal bilgisayarlarda taşma sorun olur. Çünkü yazaçlar sınırlıdır. Yazacın uzunluğu n bitlik ise bunun içine $n + 1$ bit bilgi aktarılamaz. Bundan dolayı bir çok bilgisayar taşma olup olmadığına bakar, eğer olursa bir taşma yaz-bozu devreye girer.

Taşma farklı işaretli sayılarda meydana gelmez, aynı işaretli sayılarda meydana gelir. Sayının biri pozitif diğeri negatif olduğunda taşma olmaz sonuç daima mutlak değerce büyük sayıdan daha küçüktür. Her iki sayı pozitif yada her iki sayı negatif olduğunda taşma olur. Örneğin iki işaretli sayı +70 ve +80 8 bit yazaçta saklansın. Her bir yazaca +127 den –128 e kadar ikili değerler aktarılabilir. İki sayının toplamı +150 dir. 8 bit yazacın kapasitesini aşmıştır.

elde: 0	1	elde: 1	0
+70	0 1000110	–70	1 0111010
+80	0 1010000	–80	1 0110000
+150	1 0010110	–150	1 1101010

Birinci işlemin sonucu pozitif olduğu halde toplamada taşmadan dolayı negatif işaret biti var. Bu 8 bit göz önüne alındığında böyledir. 9 bit göz önüne alınsaydı sonuç doğru olacaktı. Fakat 9 bit bu sistem içinde

saklanamaz. O halde 8 bit sonuç yanlıştır. Aynı şey negatif sayılar içinde doğrudur.

Taşmanın Tespiti

Taşmanın varlığı işaret biti ile bundan taşan elde nin aynı olup olmamasından anlaşılır. Buda ÖZEL-VEYA kapısıyla ortaya çıkar. İki elde aynı değil ise taşma vardır. ÖZEL-VEYA = 1 olur.

Ondalık Sabit Nokta Gösterimi

İkili sayıların yazaçlarda gösterimi ondalık rakamlarla belirtilen ikili kodların bir fonksiyonudur. Bir 4 bitlik ondalık kodun her bir ondalık rakamı için 4 yaz-boza ihtiyaç vardır. 4385 in BCD gösterimi için her bir rakam 4 yaz-bozda olacak şekilde 16 yaz-boza ihtiyaç vardır. Bu sayı 16 yaz-bozlu yazaçta şöyle gösterilir:

0100 0011 1000 0101

İkili kodda bir ondalık sayıyı saklamak için gerekli bit sayısı, bu sayının eşdeğer ikili gösterimi için gerekli bit sayısından daha fazla olduğundan sayıları ondalıkta göstermekle dikkate değer miktarda hesaplamalara ihtiyaç duyulur. Bu yüzden bazı bilgisayarlar ve elektronik hesap makineler, aritmetik işlemleri doğrudan ondalık verilerle yaparlar. Bazı bilgisayar sistemleri ikili ve ondalık verilerin her ikisi ile matematiksel işlem yapmak için gerekli donanıma sahiptir.

BCD deki işaretli ondalık sayıların gösterimi ikili deki işaretli sayıların gösterimine benzer. Bir ondalık sayının işareti genellikle ondalık rakamların 4 bit kodu ile uyum sağlaması için 4 bitle gösterilir. Genellikle + işareti 0000 ile – işareti de 9 un gösterimine denk 1001 ile verilir.

İşaretli büyüklük sisteminin bilgisayarlarda kullanımı zordur. İşaretli tümleyen sistemi 9'a veya 10'a göre tümleyen olabilir. En çok kullanılan 10'a göre tümleyendir. Bir BCD sayısının 10'a göre tümleyenini elde etmek için önce 9'a göre tümleyeni alınıp daha sonra en küçük rakama 1 eklenir. 9'a göre tümleyeni her bir rakamın 9 dan çıkarılmasıyla elde edilir.

Bu işlemler işaretli 2'ye göre tümleyeni almak için geliştirilir. Buda ondalık sayılarda işaretli 10'a göre tümleyene uygulanır. Toplama işaretli rakamlar içeren bütün rakamları toplamakla ve son elde yi atmakla yapılır. Ondalık sayılardan negatif sayılar işaretli 10'lu tümlenmiş şekildedir. İşaretli 10'a göre tümlenmiş sistemde (+375) + (–240) = +135 toplamının yapılması incelendiğinde

$$\begin{array}{rrl} 0 & 375 & (0000\ 0011\ 0111\ 0101)_{BCD} \\ +\ 9 & 760 & (1001\ 0111\ 0110\ 0000)_{BCD} \\ \hline 0 & 135 & (0000\ 0001\ 0011\ 0101)_{BCD} \end{array}$$

İkinci sayının solundaki 9 sayının negatif olduğunu gösterir. 9760, 0240 ın 10'a göre tümleyenidir. İki sayı toplanarak son elde atılıp +135 elde edilir. Bilgisayardaki ondalık sayılar işaret rakamlarını içeren BCD olmalıdır. Toplama BCD toplayıcısı ile yapılır (bak şekil 10.18).

İşaretsiz veya işaretli 10'a göre tümlenmiş sistemdeki ondalık sayıların çıkarılması ikili sayılarla aynıdır. Bir çok bilgisayarın BCD de ondalık sayılarla doğrudan aritmetik hesaplamalar yapması için gerekli donanımı vardır. Bilgisayarın kullanıcısı, ikiliye dönüştürücü olmaksızın ondalık sayılarla çalıştırılan aritmetik işlemleri, programlanmış buyruklarla yapabilir.

3.4 Kayan Noktalı Gösterim

Kesir, Üs

Bir sayının kayan-noktalı gösterimi iki bölümden oluşur. İlk bölüm işaretli bir sabit-noktalı sayıdır ve kesir adını alır. İkinci bölüm ondalık (veya ikili) noktanın yerini gösterir ve üs olarak adlandırılır. Sabit noktalı kesir bir kesir veya tamsayı olabilir. Örneğin +6132.789 ondalık sayısı kayan-noktalı bir kesir ve bir üs olarak şu şekilde ifade edilir.

kesir	*üs*
+ 0.6132789	+ 04

Burada üssün değeri, bize ondalık düzende noktanın gerçek yerinin kesirde gösterilenden 4 basamak sağda olacağını gösterir. Bu gösterim bilimsel ifadede $+0{,}6132789 \times 10^{+4}$ ile eşdeğerdir.

Kayan noktalı bir sayı daima

$$m \times r^e$$

biçiminde olmalıdır. Sadece kesir (m) ve üs (e) fiziksel olarak bir yazaçta saklanır. (İşaretleri ile birlikte). Taban (r) ve bu tabana göre kesrin noktasının yeri için ise bazı kabuller yapılır. Kayan-noktalı sayılarla işlem yapan devreler doğru hesaplama sonuçlarını elde etmek için bu iki konuda yapılan kabullerle uyuşmak zorundadır.

İkilik düzendeki bir kayan noktalı sayı da aynı şekilde gösterilir, fakat üssün tabanı 2 dir. Örneğin +1001.11 ikili sayısı 8 bit kesir ve 6 bit üs olarak şu şekilde gösterilir.

Kesir	*üs*
01001110	000100

Kesir

Kesrin pozitif olduğunu göstermek üzere en solda bir adet 0 kullanılmıştır. Kesrin ikili noktası işaret bitini izler fakat yazaçta gösterilmez. Üs, +4'e eşdeğer ikili sayıdır. Kayan noktalı sayı aşağıdakine eşittir:

$$m \times 2^e = +(.1001110)_2 \times 2^{+4}$$

Normalizasyon

Eğer bir kayan noktalı sayının kesrinin en önemli basamağı sıfırdan farklı ise bu sayı normalize edilmiştir. Örneğin 350 ondalık sayısı normalizedir. Fakat 00035 sayısı normalize değildir. Yani kesrinin en önemli basamağı sıfır olan bir sayı normalizasyona gerek duyar. Örneğin 8 bitlik 00011010 ikili sayısı normalize değildir, çünkü ilk 3 biti sıfırdır. Bu sayı 11010000 elde edilinceye kadar 3 bit sola kaydırılarak normalize edilir. Üç basamaklık kayma, sayıyı $2^3 = 8$ ile çarpma anlamına gelir. Kayan noktalı sayının aynı değeri koruyabilmesi için sayı üssün –3 üncü kuvveti ile çarpılmalıdır. Normalize sayılar, mümkün olan en yüksek duyarlılığı sağlar. Sıfır sayısı normalize hale getirilmez. Çünkü sıfırdan farklı bir basamağı yoktur.

Kayan noktalı sayılarla aritmetik işlemler, sabit noktalı sayılara göre daha zordur, gerçekleştirilmeleri daha uzun sürer ve daha karmaşık donanım gerektirir. Bununla beraber sabit-noktalı gösterimin içerdiği ölçeklendirme problemleri yüzünden bilimsel hesaplamalar için kayan-noktalı gösterimin kullanılması bir zorunluluktur. Çoğu bilgisayarlar ve elektronik hesap makineleri kayan-noktalı işlemleri gerçekleştirmek için gerekli donanıma sahiptir. Bu donanımı bulunmayan bilgisayarlarda kullanıcının bilimsel programları, kayan noktalı sayılarla gerçekleştirebilmesine yardım eden bir altprogram kümesi bulunur.

3.5 Diğer İkili Kodlar

Önceki bölümlerde sayısal bilgisayarlarda en çok kullanılan ikili kodlanmış veri tipleri tanıtıldı. Ondalık sayılar ve alfasayısal karakterler için bazen diğer ikili kodlar kullanılır. Sayısal bilgisayarlarda kullanılan diğer ikili kodlar:

Gray Kodu

Gray Kodu

Sayısal sistemler sadece ayrık biçimdeki veriyi işleyebilirler. Pek çok fiziksel sistem ise sürekli çıkış verir. Bu verinin bilgisayar tarafından işlenmeden önce sayısal hale getirilmesi gerekir. Analog veya sürekli analog-sayısal çeviriciler tarafından sayısal veriye çevrilir. Gray kodu bazen çevrilmiş sayısal veri için kullanılır. (Çizelge 3.5). Gray kodunun, basit ikili koda göre avantajı kendine karşı düşen dizi bir diğer terimine geçtiğinde gray kodunun bir biti değişikliğe uğramaktadır. Gray kodunun tipik bir uygulaması; sürekli verinin bir shaft milinin sürekli değişen pozisyonunun temsil ettiği durumdur. Mil bölümlendirilmiş ve her bir bölmeye bir sayı karşılık getirilmiştir. Eğer arka arkaya gelen bölmeler, ardışık gray kodlarına karşılıksa, milin pozisyonu doğarak yanlış algılama en alt düzeye indirilebilir.

Çizelge 3.5 4 bit Gray Kodu

İkili kod	Ondalık karşılığı	İkili kod	Ondalık karşılığı
0 0 0 0	0	1 1 0 0	8
0 0 0 1	1	1 1 0 1	9
0 0 1 1	2	1 1 1 1	10
0 0 1 0	3	1 1 1 0	11
0 1 1 0	4	1 0 1 0	12
0 1 1 1	5	1 0 1 1	13
0 1 0 1	6	1 0 0 1	14
0 1 0 0	7	1 0 0 0	15

Gray kodu sayıcıları, bazen sayısal sistemlerde işlemleri denetleyen zamanlama dizilerini sağlamak amacıylada kullanılmaktadır. Bir gray kodu sayıcısı, yaz-bozların durumları çizelge 3.5 te gösterilen şekilde değişen sayıcıdır. Gray kodu sayıcıları, sayıcının bir durumundan diğerine geçişi sırasında ortaya çıkabilecek hataları ortadan kaldırır.

Diğer Ondalık Kodlar

Ondalık rakamlar için ikili kodların kullanılması, en azından 4 bit gerektirir. 4 veya daha fazla biti düzenleyerek 10 mümkün birleşim için çeşitli kodlar tanımlanabilir. Çizelge 3.6 da bir kaç mümkün birleşim verilmiştir.

Kendini Tümleme

BCD önceden tanıtılmıştı. Bu kod doğrudan basamağın ikili karşılığını kullanır. BCD kodu kullanıldığında 6 kullanılmayan bit birleşimi anlamsızdır. BCD kullanmanın bir dezavantajı bir sayının 9'a göre tümleyeninin hesaplanması istenildiğinde karşılaşılan güçlüktür. Diğer yandan 2421 ve Excess-3 kodlarında bu tümleyen işlemi kolayca gerçekleştirilebilir. Bu iki kodun kendini-tümleme özellikleri vardır. Yani bir ondalık sayı, bu kodlarla ifade edilmişse 9'a tümleyeni 1 leri 0, 0 larıda 1 yaparak elde edilebilir.

Ağırlıklı Kod

2421 kodu ağırlıklı kodlara bir örnektir. Ağırlıklı bir kodda bitler verilen ağırlıklarla çarpıldığında ve elde edilen ağırlıklı bitler toplandığında sonuç ondalık basamağı verir. Örneğin 1101 bit birleşimi sıra ile 2421 ağırlıklarına karşı düşürüldüğünde $2 \times 1 + 4 \times 1 + 2 \times 0 + 1 \times 1 = 7$ Buda 2421 kodunda 7 nin karşılığıdır. BCD kodu içinde 8421 şeklinde ağırlıklar tanımlanabilir, bu yüzden BCD bazen 8421 kodu olarak ta isimlendirilir.

Excess-3 Kodu

Bu kod daha eski bilgisayarlarda kullanılan bir ondalık koddur. Ağırlıksızdır. Bir basamağın excess-3 kodu ona karşılık düşen BCD koduna 3 eklenerek elde edilir.

Çizelge 3.6 Ondalık rakamlar için 4 farklı ikili kod

Ondalık Rakam	BCD 8421	2421	Excess-3	Excess-3 gray
0	0000	0000	0011	0010
1	0001	0001	0100	0110
2	0010	0010	0101	0111
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1100
6	0110	1100	1001	1101
7	0111	1101	1010	1111
8	1000	1110	1011	1110
9	1001	1111	1100	1010
	1010	0101	0000	0000
	1011	0110	0001	0001
tanımlanmış	1100	0111	0010	0011
bit	1101	1000	1101	1000
birleşimi	1110	1001	1110	1001
	1111	1010	1111	1011

Çizelge 3.5 ten Gray kodunun (eğer ilk 10 sayıya bakılırsa) ondalık bir kod olarak oluşturulmadığı görülür. Çünkü 9 dan 0 a geçişte üç bitte değişiklik vardır. (1101 → 0000). Bu sorunu çözmek için üçüncüden (0010), onikinci ye (1010) kadar kodlar ele alınmaktadır. Burada kod 3 kaydırıldığı için Excess-3 Gray adını alır. Çizelge 3.6 da diğer ondalık kodlarla birlikte listelenmektedir.

Diğer Alfasayısal Kodlar

Çizelge 3.4 teki ASCII kodu ikili bilginin iletimi için genellikle kullanılan standart koddur. Her bir karakter 7 bitlik bir kod ile simgelenir. 8 inci bit ise genelde doğrulama (eşlik) işlemi içindir. Kod 128 karakteri kapsar. 95 karakter grafik semboldür. (Büyük-küçük harfler, 0...9 arası rakamlar, noktalama işaretleri ve özel semboller). 23 karakter ise biçim karakterleridir. Örneğin yeni satır, bir karakter geri gibi. Diğer 10 karakter ise veri iletişiminin akışını yönlendirme ve durumunu rapor etme amacıyla kullanılır.

EBCDIC

Bir başka alfasayısal kodda IBM lerde kullanılan EBCDIC dir. Bu kod her bir karakter için 8 bit ve buna ek olarak da 1 adet doğrulama biti kullanır. ASCII ile aynı karakter sembollerine sahiptir. Fakat bu sembollerin karşılıkları farklıdır.

Alfasayısal karakterler, bu bilgisayarın içerisinde veri işleme amacıyla kullanılıyorsa (iletim amacıyla değil), 64 karakteri temsil eden 6 bitlik bir kod kullanılması daha elverişlidir. 6 bitlik bir kod, 26 büyük

harfi, rakamları ve 28 özel karakteri tanımlayabilir. Kodlama için daha az bit kullanımı, büyük miktarda alfasayısal veri depolanması gerektiğinde bellek alanı ihtiyacı düşünüldüğünde avantajlıdır.

3.6 Hata Bulma Kodları

İkili bilgiler iletişim ortamlarında taşınırken, dışarıdan gelen (gürültülere) etkilere bağlı olarak bazı bitler 1 den 0 a, 0 dan 1'e karşılıklı olarak değişir. Bir hata bulma kodu ile bu hatalar taşınma esnasında bulunabilir. Fakat düzeltilemez. Hataların sıklığı nedeniyle bu işlemler genelde gözlenmektedir.

Eşlik Biti

Çoğu hata bulma kodları eşlik bitlerini kullanır. Bir eşlik bit, ikili mesaj ile birlikte bir ilave bit içerir. Eşlik bit ikili mesajdaki 1 lerin toplamına göre değer alır. 3 bitlik bir mesaj ve iki mümkün eşlik bit çizelge 3.7 de gösterilmiştir.

Eşlik Üreteci, Eşlik Denetimi

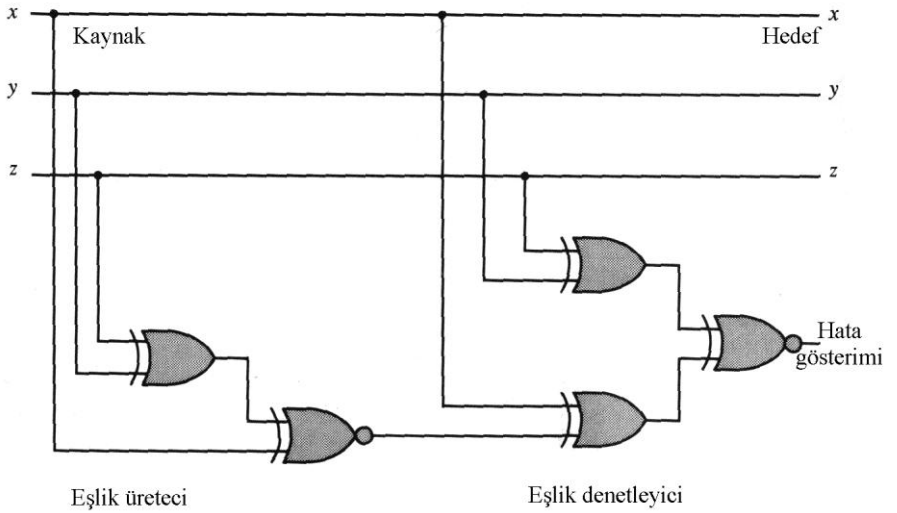
Bir bilginin bir yerden başka bir yere aktarımı boyunca eşlik biti aşağıdaki şekilde tutulur. Göndermenin sonunda mesaj (bu kez 3 bit) bir eşlik bit üretecine uygulanır. Eşlik bitini üretmeye ihtiyaç duyulduğunda, eşlik bitini içeren mesaj onun varacağı yere gönderilir. Alış bittiğinde tüm yeni bitler (burada 4 bit) eşlik denetimine uygulanır. Eşlik denetimi bitlerden 1, 3 veya daha fazla tek bit değiştiğinde hatayı sezer. Çift sayıdaki hatalar sezilemez.

Tek Fonksiyon

Eşlik üreteci ve denetleyici ağı ÖZEL-VEYA fonksiyonlarından oluşan bir mantık devresidir. Bölüm 1.2 de anlatılan ÖZEL-VEYA fonksiyonu üç veya daha fazla değişken için tanımlanan bir *tek* fonksiyondur. Tek fonksiyon bir mantıksal fonksiyondur. Tek sayıdaki değişkenlerin değeri 1 ise bu fonksiyonun değeri 1 dir. Bu tanımlamaya göre *P*(çift) fonksiyonu *x*, *y*, *z* değişkenlerinden herhangi ikisi 1 ise mantıksal 1 değerini alır (çizelge 3.7). *P*(tek) fonksiyonu *P*(çift) fonksiyonunun tümleyenidir.

Çizelge 3.7 Eşlik bit oluşumu

Mesaj xyz	*P*(tek)	*P*(çift)
000	1	0
001	0	1
010	0	1
011	1	0
100	0	1
101	1	0
110	1	0
111	0	1



Şekil 3.3 Tek eşlik biti ile hata bulma

Bir örnek olarak, 3 bitlik mesaj tek eşlik biti ile gönderilsin. Gönderme sonunda eşlik üreteci devresi ile tek-eşlik biti üretilir. Şekil 3.3 te olduğu gibi, bu devre bir ÖZEL-VEYA ve bir ÖZEL-VEYA-DEĞİL kapısına sahiptir. P(çift), x, y, z nin ÖZEL-VEYA lanması ve P(tek) in tümlenmesinden oluşur. Mesaj ve tek-eşlik biti varış yerinde alındığında eşlik denetimine uygulanır. Aktarım boyunca bir hata oluştuğunda, dört bitin eşlik biti çift olarak alınırsa, aktarılan ikili bilginin kendisi tektir. Dört giriş için ÖZEL-VEYA fonksiyonu bir tek fonksiyondur. Bizim bir ÖZEL-VEYA-DEĞİL kapısı kullanarak çıkışın tekrar tümleyenini almamız gerekir.

Dördüncü bit daima mantıksal 0 kaldığında eşlik üreteci eşlik denetleyicisi ile aynı devre olmaktadır. Bunun avantajı aynı devrenin hem eşlik üreteci hem de eşlik denetleyicisi olarak kullanılabilmesidir.

Bu örnekten de görüldüğü gibi çift-eşlik üreteci ve denetleyicileri, ÖZEL-VEYA fonksiyonları ile uygulanabilir. Tek-eşlik ağları, çıkış fonksiyonunun tümleyenini almak için bir ÖZEL-VEYA-DEĞİL kapısı kullanırlar.

SORULAR

3-1. İkilik tabandaki değerleri ondalık tabandaki değerlere dönüştürün: 101110; 1110101 ve 110110100.

3-2. Tabanlarına uygun olarak sayıları ondalık tabana dönüştürün: $(12121)_{3;}\,(4310)_{5;}\,(50)_{7;}\;\;(198)_{12}$

3-3. Ondalık tabandaki değerleri ikilik tabana dönüştürün: 1231; 673; 1998.

3-4. Ondalık tabandaki sayıları belirtilen tabanlara dönüştürün.
a. 7562 sekizlik tabana
b. 1938 onaltılık tabana
c. 175 ikilik tabana

3-5. F3A7C2 onaltılık tabandaki değeri ikilik ve sekizlik tabana çevirin.

3-6. $x^2 - 10x + 31 = 0$ denklem sisteminin çözümü $x = 5$ ve $x = 8$ ise denklem sisteminin tabanı nedir?

3-7. Onlu 215 sayısını (a) ikili, (b) ikili kodlanmış sekizli; (c) ikili kodlanmış onaltılık; (d) ikili kodlanmış onlu biçimde tutan 12 bitlik yazacın değerlerini bulunuz.

3-8. Onlu 295 sayısının eşdeğerini (a) ikili; (b) BCD; (c) 8 bit çift eşlikli ASCII olarak 24 bit yazaçtaki durumlarını gösteriniz.

3-9. İsminizi, 8 bitlik ASCII kodu ile yazın en soldaki bit 0 olsun. İsimlerinizin arasına boşluk koyun.

3-10. Aşağıdaki ASCII kodunu çözün

1001010 1001111 100100010011 10 0100000 1000100 1001111 1000101

3-11. Aşağıdaki 8 haneli ondalık sayıların 9'a göre tümleyenini alınız.
12349876 00980100 90009951 00000000

3-12. Aşağıdaki 6 haneli ondalık sayıların 10'a göre tümleyenini alınız.
123900 090657 100000 000000

3-13. Aşağıdaki 8 basamaklı ikili sayıların 1'e ve 2'ye göre tümleyenini alınız.
10101110 10000001 10000000 00000001 00000000

3-14. Aşağıdaki işaretsiz ondalık sayıları çıkarılanın 10'a göre tümleyenini alarak birbirinden çıkartınız.
a. 5250 – 1321
b. 1753 – 8640
c. 20 – 100
d. 1200 – 250

3-15. Aşağıdaki işaretsiz ikili sayıları 2 ye göre tümleyen alarak birbirinden çıkarınız.
a. 11010 – 10000
b. 10101 – 1101
c. 100 – 110000
d. 1010100 – 1010100

3-16. Aşağıdaki işlemleri negatif sayılar için işaretli 2 ye göre tümleyeni kullanarak yapınız.
(+42) + (–13) ve (–42) – (–13)

3-17. Aşağıdaki aritmetik işlemleri 2 ye göre tümleyeni kullanarak gerçekleştiriniz. Her bir sayıyı 8 bitlik yazaçta işareti ile birlikte saklayınız. Her iki durumda da taşma olduğunu görünüz.

(+70) + (+80) ve (–70) + (–80)

3-18. Aşağıdaki ondalık sayıları belirtilen işlemlere negatif sayılar için 9 a göre tümleyenini kullanarak yapınız.

a. (–638) + (+785)

b. (–638) + (+185)

3-19. 36 bit kayan noktalı bir sayı 8 bit üs 26 bit kesir içeriyor. Kesir normalize olduğuna göre sıfır dahil olmak üzere gösterebildiği en büyük ve en küçük sayılar nelerdir.

3-20. +46.5 sayısını 24 bit kayan noktalı ikili şekilde gösterin. Normalize kısım 16 bit mantis ve 8 bit üs içermektedir.

3-21. Gray kodu özelliklerini kullanarak

a. Çizelge 3.5 in devamı olarak 16 dan 31 e kadar olan gray kodlarını yazınız.

b. Çizelge 3.6 nın devamı olarak ondalık 10 ile 19 arasındaki sayıların excess-3 gray kodlarını yazınız.

3-22. Ondalık 8620 sayısını (a) BCD; (b) excess-3 kod; (c) 2421 kod; (d) ikili olarak yazınız.

3-23. 10 BCD sayıyı en soluna çift eşlenik bitini yazarak sıralayın. Aynı işlemi tek eşlenik biti ile tekrarlayın.

3-24. Ondalık 3984 sayısını 2421 koduna çevirin. Çevrilmiş sayıların tüm bitlerinin tümleyenini alın ve sonucun 2421 kodunda 3984 sayısının 9 a göre tümleyeni olduğunu gösterin.

3-25. $x=A \oplus B \oplus C \oplus D$ ÖZEL-VEYA fonksiyonun tek fonksiyon olduğunu gösteriniz. Bunu göstermek için bir yol doğruluk çizelgelerinden $y = A \oplus B$ ve $z = C \oplus D$ yi elde ediniz. Sonra $x = y \oplus z$ yi doğruluk çizelgesinden formüle ediniz.

3-26. 3 bitlik eşlik üretecini ve 4 bitlik eşlik denetleyicisini çift eşlenik bitini kullanarak oluşturun.

KAYNAKLAR

1. Hill, F. J., and G.R. Peterson, *Introduction to Switching Theory and Logical Desing*, 3rd ed. New York: John Wiley , 1981.
2. Langholz, G., J. Francioni, and A. Kandel, *Elements of Computer Organization*. Englewood Cliffs, NJ: Prentice Hall, 1989.
3. Lewin, M. H., *Logic Desing and Computer Organization*. Reading, MA: Addison-Wesley, 1983.
4. Mano, M. M., *Digital Desing*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1991.

5. Roth, C. H., *Fundamentals of Logic Desing*, 3rd ed. St. Paul, MN: West Publishing, 1985.
6. Sandige , R. S., *Modern Digital Desing*. New York: McGraw-Hill, 1990.
7. Shiva, S. G., *Introduction to Logic Desing*. Glenview, II: Scott, Foresman, 1988.
8. Tomek, I., *Introduction to Computer Organization*. Rockville, MD: Computer Science Press, 1981.
9. Wakerly, J. F., *Microcomputer Architecture and Programming*. New York: John Wiley, 1981.
10. Ward. S. A., and R. H. Halstead, Jr., *Computation Structures*. Cambridge, MA: MIT Press, 1990.

BÖLÜM 4

Yazaç Aktarımı ve Mikroişlemler

BU BÖLÜMDE

4-1 Yazaç Aktarım Dili
4-2 Yazaçlar Arası Aktarım
4-3 Veri Yolu ve Bellek Aktarımı
4-4 Aritmetik Mikro İşlemler
4-5 Mantıksal Mikro İşlemler
4-6 Kaydırma Mikro İşlemleri
4-7 Aritmetik Mantık Kaydırma Birimi

4-1 Yazaç Aktarım Dili

Sayısal bir sistem, sayısal donanım modüllerinin belli bir bilgi işleme olayını gerçekleştirmek üzere iç bağlantılarla bir araya gelmesinden oluşur. Sayısal sistemlerin büyüklükleri çok farklıdır. Ayrıca çok basit veya çok karışık olabilirler, birkaç basit IC den de oluşabilirler. Birbirleriyle bağlantılı ve etkileşimli bir kaç bilgisayardan da oluşabilirler. Sayısal bilgisayarların tasarımında modüler yaklaşım kullanılır. Bu modüller yazaçlar, kod çözücüler, aritmetik elemanlar ve denetim mantığından oluşur. Muhtelif modüller ortak veri ve denetim yollarıyla bir bilgisayarı oluşturacak şekilde iç bağlanmışlardır.

Mikro İşlem

Her bir sayısal modül bunları oluşturan yazaç ve yazaçlarda mevcut bilgiler üzerinde icra edilen işlemlerle belirlenir. Yazaçlarda yüklü veriler üzerinde icra edilen işlemlere mikro işlem denir. Basit bir işlemdir ve bir saat zamanı içinde yazaçtaki veri üzerinde icra edilir. Mikro işlemin sonucunda yazacın içindeki ikili bilgi değişebilir veya başka bir yazaca aktarılabilir. Mikro işlem örnekleri kaydırma, sayma, yükleme ve silmedir. 2. bölümdeki bazı sayısal IC fonksiyonları mikro işlemler içerirler.

Örneğin paralel yüklemeli bir ikili sayıcı yükleme ve arttırma mikro işlemlerini icra edebilir. İki yönlü bir kaydırma yazacı sağa ve sola kaydırma işlemlerini yapar.

Bir sayısal bilgisayara aşağıdaki bilgilerin verilmesiyle iyi bir biçimde tanımlanabilir.

1. İçerdiği yazaç kümesi ve bunların yapacağı işler.
2. Yazaç içine yazılı ikili bilgiler üzerine icra edilecek mikro işlemler.
3. Mikro işlemlerin icrasını sağlayacak (başlatacak) denetim fonksiyonları.

Bütün mikro işlemleri, her işlem için ayrı ayrı kelimelerle açıklamak mümkündür. Bu uzun işlemdir. Bunun yerine uygun semboller bulunması daha iyi olur. Bu semboller yazaçlar arası aktarmaları ve muhtelif aritmetik ve mantık mikro işlemlerini tanımlamalıdır.

Yazaç Aktarım Dili

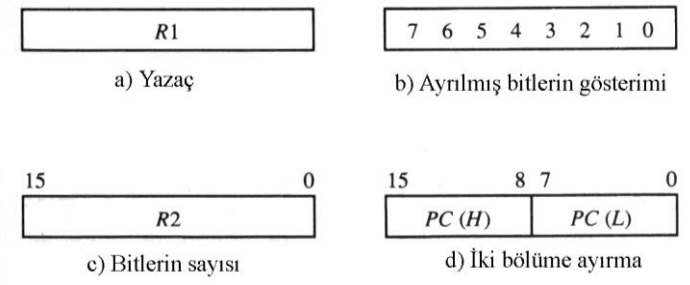
Yazaçlar arasındaki mikro işlem aktarma işlemlerini tanımlayan sembolik biçime, *Yazaç aktarım dili* adı verilir. Bunun anlamı istenen mikro işlemi yapacak donanımın mevcut, olduğu ve işlemin sonucunun aynı veya farklı yazaca aktarılabileceğidir. Yazaç aktarım dili, sayısal modüllerin yazaçları arasında meydana gelecek mikro işlemleri sembolik biçimde ifade eden bir sistemdir. Sayısal bilgisayarların iç yapılarını hassas ve tam bir biçimde ifade etmek için uygundur. Aynı zamanda sayısal sistem tasarımında kullanılabilir.

Ancak şu anda bu konuda birlik sağlanmıştır. Burada verilecek semboller en basit olanlar olarak seçilmiştir. Bunları anlatacak ve kullandıkları donanımları belirteceğiz. Yazaç aktarım dili, bağlı olduğu yazaçlar ve donanımdan ayrılamaz.

4-2 Yazaçlar Arası Aktarım

Yazaçlar

Bilgisayarlardaki yazaçlar büyük harflerle gösterilir. Bazen arkasına sayılarda gelebilir. Örneğin bellek birimin bir adresini tutan yazaç genelde bellek adres yazacı olarak isimlendirilir, kısaca *MAR* diye belirtilir. Yazaçların diğerleri *PC* (program sayıcı), *IR* (buyruk yazacı) ve *R*1 (işlemci yazacı). *n* bitlik bir yazaç içindeki yaz-boz lar 0 dan $n - 1$ e kadar numaralandırılır. Sayma sağdan veya soldan başlayabilir. Şekil 4.1 çeşitli yazaç blok şema biçimlerini göstermektedir. Yazaçların gösterilmesi için yaygın yöntem, dikdörtgen bir kutu içine yazacın ismi yazılmaktadır şekil 4.1(a). Bitler (b) deki gibi ayrı ayrı verilmektedir. (c) de 16 bitlik bir yazaç 0 dan 15 e kadar numaralandırılmaktadır. (d) de 16



Şekil 4.1 Yazaçların blok şeması

bitlik bir yazaç iki bölüm içermektedir. 0 dan 7 ye kadar düşük mertebeli bitler *L* harfi ile belirtilmekte, 8-15 arası yüksek seviyeli bitler *H* sembolü ile belirtilmektedir. Bu yazaç 16 bitlik bir *PC* dir. Düşük mertebeli bitler *PC*(0–) veya *PC*(*L*) ile yüksek mertebeli bitler *PC*(8–15) veya *PC*(*H*) ile gösterilir.

Yazaç Aktarımı

Bir yazaçtan diğerine bilgi aktarımı sembolik biçimde

$$R2 \leftarrow R1$$

şeklinde düzenlenmektedir. Burada *R*1 yazacının içeriği *R*2 yazacına aktarılmaktadır. Aktarma sonucunda kaynak yazaç *R*1 in içindeki değer değişmemektedir.

Yazaçlar arası veri aktarımı, mikro işlemlerle gerçekleşir. Bu işlemlerde bilgi bir yazaçtan diğerine doğrudan aktarılır. Yeni bilgiyi alan yazaç içindeki bilgi, kaynak yazacının aynı olur. Aktarımdan dolayı kaynak yazacın değeri değişmez. Bir yazaçtan diğerine bilgi aktarımı paralel yolla olduğunda aynı zamanda olur, bir saat zamanı içinde gerçekleşir. Normalde aktarmanın sadece önceden belirlenen bir koşul altında gerçekleşmesi istenir.

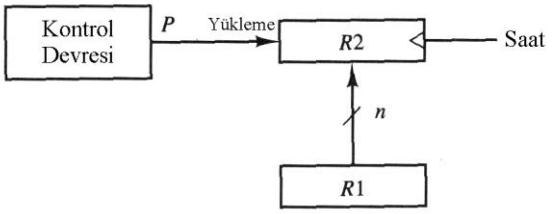
Denetim Fonksiyonu

Bir if - then koşulunda bu durum

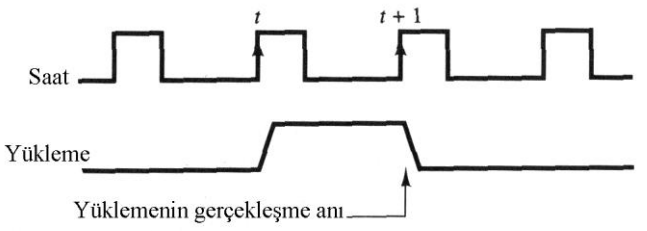
$$if\ (P = 1)\ then\ (R2 \leftarrow R1)$$

şeklinde gösterilir. Bu aktarımın olmasını sağlayan ve bunu gerçekleştiren ikili şarta *denetim fonksiyonu* denir. Bir denetim fonksiyonu 1 veya 0 değerini içeren Boole değişkendir. Denetim fonksiyonu $P = R2 \leftarrow R1$ şeklindeki bir durumu içermektedir.

Her bir durum bir yazaç aktarım gösterimi ile yazılabilir. Bunu yerine getirmek için donanım yapısı olmalıdır. Şekil 4.2 *R*1 yazacından *R*2 yazacına aktarımı göstermektedir. *R*1 yazacının *n* çıkışı *R*2 yazacının *n* girişine bağlanmaktadır. *n* harfi yazaçtaki bit sayısını gösterir. Yazacın genişliği bilindiğinde yazaç üzerine gerçek sayılar yerleştirilebilir. *P* denetim değişkeni aktif hale geçtiğinde *R*2 yazacına giriş değerleri yüklenmek-



a) Blok Şeması



b) Zamanlama Şeması

Şekil 4.2 $P = 1$ olduğunda $R1$ den $R2$ ye aktarım

tedir. Denetim değişkeninin saate bağlı olduğunu kabul edelim. $R2$ nin ayrıca saat vuruşlarını alacak bağlantısı mevcuttur. P denetim fonksiyonu, denetim devresi tarafından üretilir. Sonra yazacın yükleme devresine uygulanır. $P = 1$ fonksiyonu ile t zamanından $t + 1$ zamanına geçişte bilgiler paralel olarak $R2$ yazacına yüklenir ve sonra yine P nin değeri 0 yapılır.

Not edelim ki saat yazaç aktarım durumlarını içeren bir değişken değildir. Tüm aktarımların saatin yükselen kenarında yapıldığı kabul edilmektedir. P nin değeri t zamanında aktif olsa da gerçek aktarım saatin $t + 1$ zamanına geçişine kadar yapılmaz.

Çizelge 4.1 yazaç aktarım dilindeki temel sembolleri göstermektedir. Yazaçlar büyük harf ve harfleri takiben sayılarla gösterilir. Parantez içindeki rakamlar bitlerin başlangıç ve bitiş değerlerini veya yazaçların bölüm sembollerini gösterir. Oklar aktarım yönünü göstermektedir. "," aynı anda olan mikro işlemleri göstermektedir. Bu durum

$$T: R2 \leftarrow R1, R1 \leftarrow R2$$

$T = 1$ durumunda aynı saat vuruşunda $R1$ ve $R2$ yazaçlarının içlerinin değiştirilmesini göstermektedir.

Çizelge 4.1 Yazaç aktarımdaki temel semboller

Sembol	Tanımla	Örnekler
Harfler (ve rakamlar)	Bir yazaç gösterir	*MAR*, *R*2
Parantez ( )	yazacın bir bölümünü gösterir	*R*2(0–7), *R*2(*L*)
Ok ←	bilginin aktarma yönünü	*R*2 ← *R*1
Virgül	birden fazla mikro işlemi	*R*2 ← *R*1, *R*1 ← *R*2

4.3 Veri Yolu ve Bellek Aktarımı

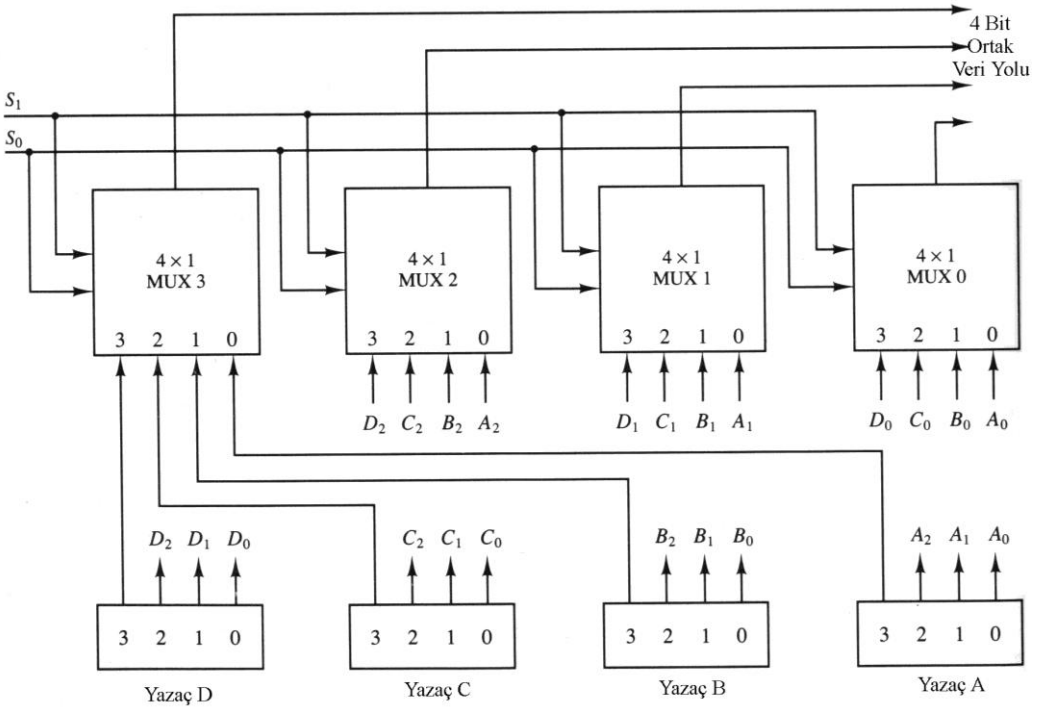
Ortak Veri Yolu

Genel bir sayısal bilgisayar bir çok yazaca sahiptir ve bir yol ile aktarım bilgilerini bir yazaçtan diğerine aktarmayı sağlamaktadır. Sistemdeki tüm yazaçlar arasında yollar kullanıldığından sayıları çoktur. Bir ortak veri yolu sistemi, çoklu yazaç konfigürasyonları için yazaçlar arasındaki bilgilerin bir çok etkin şekilde aktarılmasını sağlar. Bir veri yolu yapısı yazaçtaki her bir bit için ortak yolların bir kümesini oluşturmaktadır. Çoklu yazaç konfigürasyonlarında bilgilerin yazaçlar arasında aktarımı için daha etkin bir yöntem ortak veri yolu kullanmaktır.

Ortak veri yolu sistemi kurmanın bir yolu MUX lar kullanmaktır. Seçiciler kaynak yazacı seçmekte, bu yazaçların içinde veri yolu üzerinde olması gereken ikili bilgiler bulunmaktadır. Şekil 4.3 te dört yazaç için bir veri yolu sisteminin yapısı gösterilmektedir. Her bir yazaç 0 dan 3 e kadar numaralandırılan 4 bite sahiptir. 0...3 arası 4 veri girişine sahip, S_1 ve S_0 seçim girişleri bulunan 4 tane 4 × 1 lik MUX un çıkışları veri yoluna bağlanmıştır. Yazaç çıkışlarından MUX girişlerine gelen 16 yol sıralı değildir. Örneğin *A* yazacının çıkışı 1 olduğundan MUX1 in 0 girişine bu değer gelir, çünkü bu A_1 girişidir. Bu şema bitlerin bazı önemli yerlerini göstermektir. Her bir yazaç, bir MUX un veri girişi ile ve böylece veri yolunun bir yolu ile ilişkilidir. Böylece yazaçların tüm 0 ıncı bitleri MUX0 ile, tüm 1 inci bitleri MUX1 ile ilişkilidir.

Veri Yolu Seçimi

S_1 ve S_0 seçim telleri 4 MUX unda seçim girişlerini oluşturur. Seçim telleri 4 yazacın her birinden bir bit seçerek, 4 hatlık ortak veri yolunun içine aktarır. $S_1S_0 = 00$ olduğunda 4 MUX unda 0 girişleri seçilerek MUX ların çıkışına ve buradan veri yoluna uygulanır. Bu seçim veri yoluna *A* yazacındaki bilgilerin aktarılmasını sağlar. Benzer olarak $S_1S_0 = 01$ olduğunda *B* yazacı seçilir ve böyle devam edilir. Çizelge 4.2 seçim telleri vasıtasıyla ikili değer içeren 4 yazaçtan uygun olanın seçimini göstermektedir.



Şekil 4.3 Dört yazaç için veri yolu sistemi

Çizelge 4.2 Şekil 4.3 için fonksiyon çizelgesi

S_1	S_0	Seçilen yazaç
0	0	*A*
0	1	*B*
1	0	*C*
1	1	*D*

Genelde bir veri yolu sistemi *n* bitlik *k* tane yazaçtan ve *n* hatlı ortak veri yolundan oluşmaktadır. *n* hatlı veri yolunda MUX ların sayısı her bir yazacın bit sayısına eşit olur. *k* yazaç olduğundan, *k* veri girişli *k* × 1 MUX a gerek vardır. Örneğin 16 bitlik 8 yazaçta 16 MUX a gerek vardır ve ortak veri yolu 16 bitlik olur. Her bir MUX, 8 veri giriş ve 3 seçim hattına sahiptir.

Bir bilginin bir yazaçtan istenen bir çok yazaçtan birine aktarılması, bu yazaçların girişlerinin veri yolu çizgilerine bağlanması ve yükle denetiminin aktif hale getirilmesi ile sağlanır. Veri yolu aktarımı için kullanılan sembolik deyimlerde veri yolu belirtilmelidir veya veri yolunun varlı-

ğı deyim içinde ima edilmelidir. Veri yolunun varlığı deyimler içinde gösterildiği zaman yazaç aktarımı şu şekilde gösterilir:

$$VERİ\ YOLU \leftarrow C, \quad R1 \leftarrow VERİ\ YOLU$$

C yazacının içeriği *VERİ YOLU* na aktarılmakta ve *R*1 yazacının yükleme denetim girişi aktif hale geçtiğinde *VERİ YOLU* nun içeriği *R*1 yazacına aktarılmaktadır. Eğer *VERİ YOLU* nun sistem içinde olduğu biliniyorsa, sadece aktarımın gösterilmesi yeterlidir.

$$R1 \leftarrow C$$

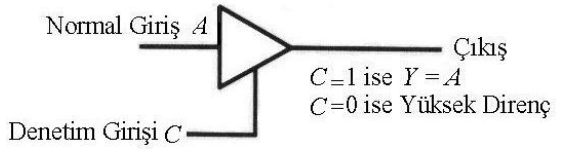
Buradan tasarımcı veri yolu ile ilgili aktarım işlemlerinde denetim sinyallerinin aktif halde olduğunu bilmektedir.

Üç Durumlu Veri Yolu Bufferları

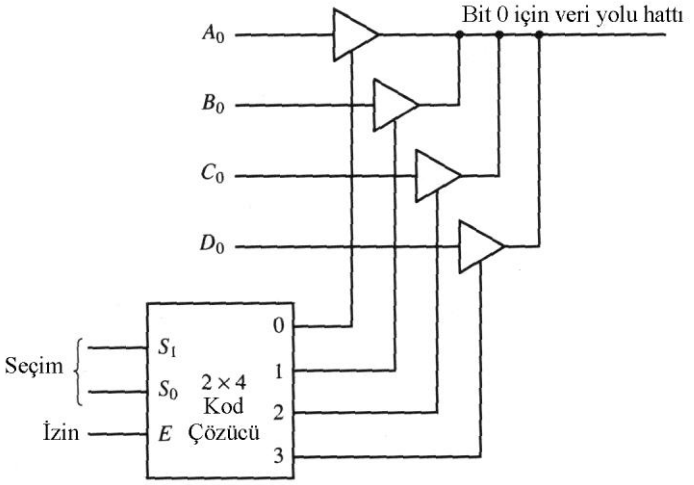
Üç Durumlu Kapı, Yüksek Direnç, Buffer

Bir veri yolu sistemi MUX lar yerine 3 durumlu kapılarla imal edilmişlerdir. Bir üç durumlu kapı, üç durumu gösteren sayısal devredir. Bu durumlardan ikisi diğer kapılar gibi 1 ve 0 sinyallerine denktir. Üçüncü durum yüksek direnç durumudur. Yüksek direnç durumu bir açık devre gibi davranmaktadır. Bunun anlamı, çıkışlar ayrılmış durumdadır ve bir mantıksal anlamı yoktur. VE, VEYA, VE-DEĞİL de olduğu gibi, üç durumlu kapı, herhangi bir normal mantıkta çalışabilir. Buna rağmen genelde en çok buffer kapılarından oluşan bir veri yolu sisteminin tasarımında kullanılır.

Üç durumlu buffer kapısının grafik sembolü şekil 4.4 te gösterilmektedir. Normal giriş ve denetim girişinin her ikisini de içerdiğinden, normal bir buffer dan ayrılmaktadır. Denetim girişi, çıkış durumunu tanımlamaktadır. Denetim girişi 1 olduğunda, normal bir buffer gibi işlem yapmakta ve normal girişi çıkışa aktarmaktadır. Denetim girişi 0 olduğunda çıkış devre dışıdır ve normal girişin değeri ne olursa olsun kapı yüksek direnç durumuna geçer. 3 durumlu kapının yüksek direnç durumu, diğer kapılarda var olmayan bir özel nitelik içermektedir.



Şekil 4.4 Üç durumlu buffer kapısı



Şekil 4.5 Üç durumlu buffer larla veri yolu

Veri Yolu Sistemi

Üç durumlu kapılarla yapılmış bir veri yolu sistemi şekil 4.5 te gösterilmektedir. 4 buffer ın çıkışları tek bir veri yolu hattına bağlanmıştır. Buffer ların denetim girişleri tanımlıdır. Ve 4 normal giriş, veri yolu hattı ile ilişkilidir. Herhangi bir zamanda sadece bir buffer aktif durumdadır.

Şekilde görüldüğü gibi, herhangi bir zamanda sadece bir denetim girişinin aktif hale geçmesi bir kod çözücü ile sağlanmaktadır. Kod çözücünün izin girişi 0 olduğunda tüm çıkışlar 0 olur ve tüm dört buffer devre dışı olduğundan veri yolu hattı yüksek direnç durumunda olur. İzin girişi aktif olduğunda kod çözücünün seçim girişlerinin ikili değerlerine göre üç durumlu buffer lardan biri aktif hale geçer.

Her bir üç durumlu buffer kullanarak *n* bitlik dört yazaç için bir ortak veri yolunun yapımında *n* devreye gerek duyulur. Bu devreler şekil 4.5 te gösterildiği gibi 4 buffer dan oluşmaktadır. Dört buffer lık her bir grup, dört yazaçtan gereken biti almaktadır. 4 yazaç arasında seçim için sadece bir kod çözücüye gerek vardır.

Bellek Aktarımı

Bellek biriminin çalışması bölüm 2.7 de gösterilmiştir. Bilginin bellekten dış çevreye aktarılmasına okuma işlemi, yeni bilgilerin belleğe aktarılmasına da yazma işlemi denir. Her iki işlemde de seçilen bellek kelimesi, adresi yardımıyla belirlenir. Bir bellek kelimesi *M* harfiyle gösterilir. Bellek kelimesi adres yardımıyla işlem sırasında belirlenir. Bellek aktarım işlemleri yazılırken *M* in adresinin de yazılması gerekir. Adres *M* harfini takiben köşeli parantez içine yazılır.

Bellekten Okuma

İçinde adres bilgisi bulunan bir yazaç adres yazacı olarak isimlendirilmekte ve *AR* ile gösterilmekte, verilerin bir başka yazaca aktarılabildiği yazaç, veri yazacı olarak isimlendirilmekte ve *DR* ile gösterilmektedir. Okuma işlemi aşağıdaki gibi ifade edilir.

$$Okuma: DR \leftarrow M[AR]$$

Bu işlem *AR* içindeki adres ile seçilen *M* bellek kelimesindeki bilgilerin *DR* içine aktarımını göstermektedir.

Belleğe Yazma

Yazma işlemi adres ile seçilen bir bellek kelimesine bir veri yazacının içeriğini yazmayı sağlar. Adresin *AR* de olduğunu ve giriş verilerinin de *R*1 yazacında olduğunu kabul edelim. Yazma işlemi sembolik olarak

$$Yazma: M[AR] \leftarrow R1$$

şeklinde ifade edilir. Bu işlem *AR* içindeki adres ile belirtilen bellek kelimesi *M* in içine, *R*1 yazacındaki bilgiyi aktarır.

4.4 Aritmetik Mikro İşlemler

Mikro işlem, yazaçlar içinde depolanan bilgi ile gerçekleştirilen en basit ve temel işlemdir. Sayısal bilgisayarlarda sıkça görülen mikro işlemler 4 kategoride sınıflandırılır.

1. Yazaç aktarım mikro işlemleri, ikili sisteme göre hazırlanmış bilgiyi bir kayıttan diğerine aktarır.
2. Aritmetik mikro işlem, yazaçlarda saklanan sayısal bilgilerin aritmetik işlemlerini gerçekleştirir.
3. Mantıksal mikro işlemler, yazaçlarda saklanan sayısal olmayan bilgi ile ilgili bit kullanım işlemlerini gerçekleştirir.
4. Kaydırma mikro işlemleri, yazaçlarda saklanan veriler üzerinde kaydırma işlemlerini gerçekleştirir.

Yazaç aktarım mikro işlemlerine bölüm 4.2 de bir giriş yapıldı. Bu tip mikro işlemlerde, ikili sisteme göre hazırlanmış bilgi, başlangıç kaydından hedef kaydına hareket ettiğinde değer değiştirmez. Diğer 3 çeşit mikro işlem, aktarım sırasında bilgi içeriğini değiştirir. Bu kısımda aritmetik mikro işlemler anlatılacak, sonraki kısımlarda ise mantık ve kaydırma mikro işlemleri hakkında bilgiler verilecek.

En temel aritmetik mikro işlem toplama, çıkarma, artma, azalma ve kaydırmadır. Aritmetik kaydırma ileride kaydırma mikro işlemleri kısmında açıklanacaktır. Aritmetik mikro işlem aşağıdaki biçim ile gösterilir.

$$R3 \leftarrow R1 + R2$$

Çizelge 4.3 Aritmetik mikro işlemler

Sembolik atama	Tanımlama
$R3 \leftarrow R1 + R2$	$R1$ ve $R2$ nin içeriği toplanarak $R3$ e aktarılıyor
R3 ← R1 - R2	$R1$ de $R2$ çıkarılarak $R3$ e aktarılıyor.
$R2 \leftarrow \overline{R2}$	$R2$ nin 1 e göre tümleyeni alınıyor
$R2 \leftarrow R2 + R1$	$R2$ nin içeriğinin 2 ye göre tümleyeni alınıyor.
$R3 \leftarrow R1 + \overline{R2} + 1$	$R1$ ile $R2$ nin 2 ye göre tümleyeni toplanıyor.
$R1 \leftarrow R1 + 1$	$R1$ in içeriği bir arttırılıyor
$R1 \leftarrow R1 - 1$	$R1$ in çeri bir azaltılıyor.

Toplama Mikro İşlemi

Bu biçim toplama mikro işlemini gösterir. Bu biçimde $R1$ yazacının içeriği $R2$ yazacının içindeki ile toplanır ve toplam $R3$ yazacına aktarılır. Bu toplamın bilgisayar donanımında gerçekleşebilmesi için 3 adet yazaca ve toplama işlemi için sayısal bir elemana gerek vardır. Diğer temel aritmetik mikro işlemler çizelge 4.3 te gösterilmiştir. Çıkarma büyük çoğunlukla tümleyenini alma ve toplama ilave etme yoluyla uygulanmaktadır. Çıkarma operatörü kullanmak yerine çıkarma işlemi aşağıdaki tanımlamaya göre yapılır.

$$R3 \leftarrow R1 + \overline{R2} + 1$$

Çıkarma Mikro İşlemi

$\overline{R2}$ sembolü, $R2$ nin 1'e göre tümleyenidir. 1'e göre tümleyene 1 ekleyerek 2'ye göre tümleyen elde edilir. $R1$ in içeriği ile $R2$ nin 2'ye göre tümleyeninin toplanması $R1 - R2$ ye denktir.

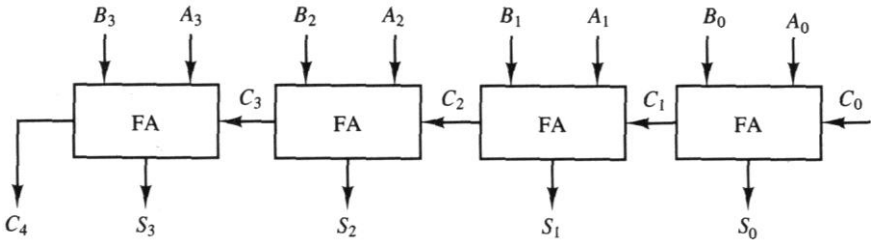
Artma ve azalma mikro işlemleri +1 ve –1 işlemleri olarak gösterilmekte. Bu mikro işlemler ikili sistemde bir birleşik devre ile veya aşağı-yukarı ikili sayıcı ile yapılır.

Çarpma ve bölme aritmetik mikro işlemleri çizelge 4.3 te yoktur. Bu iki işlem geçerli olmasına rağmen, temel gurup mikro işlemleri içinde değildir. Bu işlemlerin mikro işlem olarak tanımlanabileceği tek yer, gruplaşmış döngü yoluyla uygulanabilecek bir sayısal sistemdir. Bu durumda bu işlemleri gerçekleştiren sinyaller, kapılardan geçerek artar ve işlem sonucu çıkış sinyali birleşik devre yoluyla artar artmaz saat vuruşu ile hedef yazaç içerisine aktarılır. Bilgisayarların çoğunda çarpma işlemleri, toplama ve kaydırma mikro işlemlerinin belli bir sıra takip etmesi ile uygulanır. Bu gibi durumlarda, donanımı tanımlamak için toplama, çıkarma ve kaydırma gibi temel mikro işlemlerin kullanılmasını sağlayacak bilgi listesine gerek vardır (Bak bölüm 10 a).

İkili Toplayıcı

İkili Toplayıcı

Toplama mikro işlemini bilgisayar donanımına uygulamak için, bilgiyi tutmaya yarayan yazaçlara ve aritmetik toplamayı gerçekleştirecek sayısal elemanlara gerek vardır. İki bitin aritmetik toplamını yapan



Şekil 4.6 4 bit ikili toplayıcı

sayısal devre, ve daha önceki elde bitini içeren bir devre, tam toplayıcı olarak isimlendirilir (şekil 1.17). Herhangi bir uzunluktaki iki ikili sistemdeki rakamların aritmetik toplamını gerçekleştiren sayısal devre ikili toplayıcı olarak isimlendirilir. İkili toplayıcı bir tam toplayıcının çıkış elde sini bir sonraki tam toplayıcının giriş elde sine bağlanmasıyla oluşmaktadır. Şekil 4.6 4 tam toplayıcı kullanarak 4 bit ikili toplayıcının iç bağlantılarını göstermektedir. Elde ler tam toplayıcılar yoluyla oluşan zincire bağlanmıştır. İkili toplayıcıya giren elde C_0 ve çıkış elde si C_4 dür. Tam toplayıcının 5 çıkışı bitlerin gerek duyulan toplamını oluşturur.

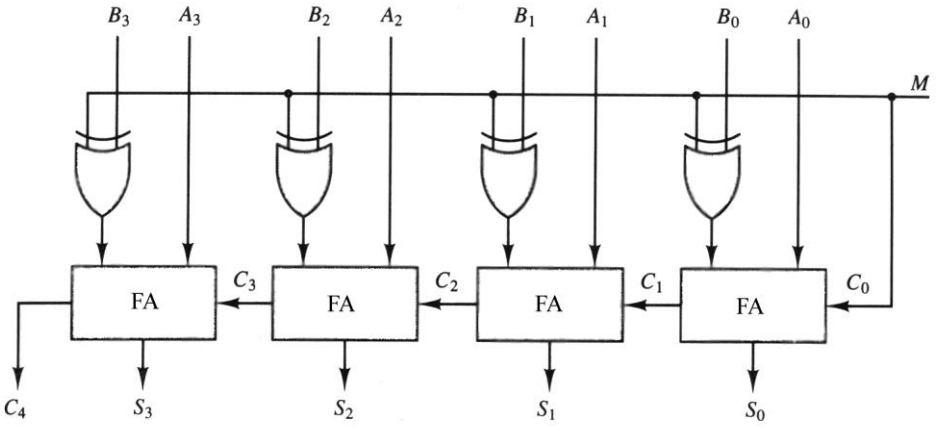
Bir *n*-bitlik ikili toplayıcı *n* tam toplayıcıya gerek duyar. Her tam toplayıcının çıkış elde si bir sonraki yüksek mertebede tam toplayıcının girişine bağlanır. *A* girişlerinin *n* veri biti bir yazaçtan gelir (*R*1 gibi) ve *B* girişlerinin *n* veri biti diğer bir yazaçtan gelir (*R*2 gibi). Bunların toplamı üçüncü bir yazaca veya başlangıç yazaçlarından (*R*1 veya *R*2) birine başlangıç değeri silinerek aktarılır.

İkili Toplayıcı-çıkartıcı

İkili sayılarda çıkarma işlemi bölüm 3.2 de tartışıldığı gibi en uygun şekilde tümleme işlemiyle yapılabilir. $A - B$ çıkarma işleminin B nin 2 ye göre tümleyeninin alınıp A ya eklenmesi ile yapıldığı bilinmektedir. 2 ye göre tümleyen 1 e göre tümleyeni alıp en önemsiz bit çiftine 1 ekleyerek elde edilebilir. 1 e göre tümleyen tersleyenler ile yapılır ve toplama giriş elde si aracılığı ile 1 eklenir.

Toplayıcı-çıkartıcı

Toplama ve çıkartma işlemleri her bir tam toplayıcı ile ÖZEL-VEYA kapısının bir genel devre içinde birleştirilmesi ile olur. Bir 4 bitlik toplayıcı-çıkartıcı devre şekil 4.7 de gösterilmiştir. M giriş kipi yapılacak işlemi denetler. $M = 0$ olduğunda devre toplayıcı, $M = 1$ olduğunda devre çıkartıcıdır. Her bir ÖZEL-VEYA kapısı giriş olarak M ve B değerlerini alır. $M = 0$ olduğunda $B \oplus 0 = B$ dir. Tam toplayıcılar B değerini alır. Giriş elde si 0 dır ve devre A, B toplamını gerçekleştirir. $M = 1$ olduğunda $B \oplus 1 = B'$ ve giriş elde si $C_0 = 1$ dır. B girişlerinin hepsinin tümleyeni



Şekil 4.7 4 bit toplayıcı-çıkartıcı devresi

alınır ve tek bir giriş elde si yoluyla tümleyene 1 eklenir. Devre *B* nin 2 ye göre tümleyenini alıp *A* ile toplama işlemini yapar. İşaretsiz sayılarda eğer $A \geq B$ ise $A - B$ yi $A < B$ ise $B - A$ yı 2 ye göre tümleyen olarak verir. İşaretli sayılarda *A* - *B* de taşma yoktur.

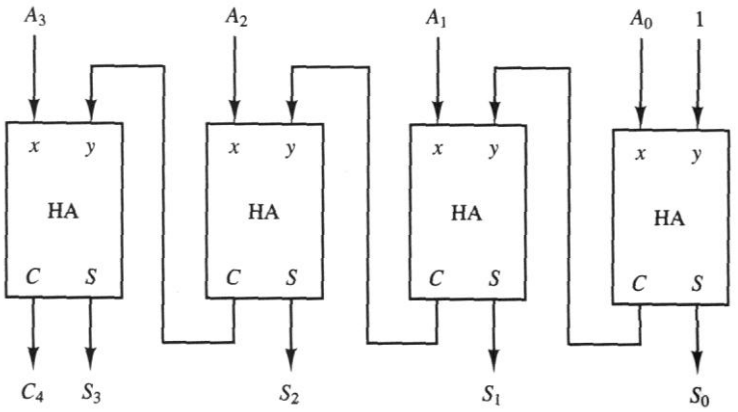
İkili Arttırıcı

Arttırma mikro işlemi yazaçtaki sayıya 1 ekler. Örneğin 4 bitlik bir yazaçtaki ikili değer 0110 ise artıştan sonra 0111 olur. Bu mikro işlem ikili sayıcı ile kolayca gerçekleştirilir (şekil 2.10). Her zaman sayma izin girişi aktiftir, saat vuruş geçişi yazacın içeriğini 1 arttırır. Arttırma mikro işleminin belirli bir yazaçtan bağımsız birleşik bir devre ile yapılmak zorunda olduğu durumlar olabilir. Kademeli olarak bağlanmış yarı toplayıcılar ile yapılabilir (bak şekil 1.16).

Arttırıcı

Şekil 4.8 dört bitlik birleşik arttırıcı devrenin şemasını göstermektedir. Yarı toplayıcının düşük giriş biti 1 ile ilişkilidir ve diğer giriş arttırma yapabilmek için en düşük önemsiz bit ile ilişkilidir. Bir yarı toplayıcının çıkış elde si komşu yüksek mertebeli yarı toplayıcının girişine bağlıdır. Devre A_0 dan A_3 e kadar 4 bit alır, buna bir ekler, S_0 dan S_3 e doğru çıkışta bir eklemiş olarak verir. Sadece ikili 1111 de çıkış elde si C_4 1 olur. Bu aynı zamanda S_0 dan S_3 e kadar çıkışların 0 değerini almasını sağlar.

Şekil 4.8 deki devre *n* yarı toplayıcıdan oluştuğunda *n* bitlik ikili sayıcıya dönüştürülebilir. *E* önemsiz bit girişi mantıksal-1 girişine bağlı olmalıdır. Diğer girişler ya önceki yarı toplayıcıdan çıkış eldesini alarak ya da arttırılarak sayısal değer alır.



Şekil 4.8 4 bit İkili arttırıcı

Aritmetik Devre

Aritmetik Devre

Çizelge 4.3 de belirtilen aritmetik mikro işlem bir karma aritmetik devre içinde uygulanabilir. Aritmetik devrenin temel elemanları paralel toplayıcılardır. Toplayıcının bilgi girişini denetleyerek farklı tiplerdeki aritmetik işlemleri elde etmek mümkündür.

Giriş Eldesi

4 bitlik aritmetik devrenin şeması şekil 4.9 da gösterilmektedir. Bu devre 4 tam toplayıcıdan oluşan, 4 bit toplayıcıdan ve farklı işlemlerin seçimi için 4 MUX tan oluşmaktadır. Burada 2 tane *A* ve *B* gibi 4 bitlik giriş bir tane 4 bitlik çıkış *D* vardır. 4 *A* girişi ikili toplayıcının *X* girişine doğrudan gider. *B* girişi her biri MUX ların bilgi girişi ile ilgilidir. MUX bilgi girişleri aynı zamanda *B* nin tümleyenini de alır. Diğer iki veri girişi mantıksal-0 ve mantıksal-1 dir. Mantıksal-0 sabitleştirilmiş voltaj değerine sahiptir. Mantıksal-1 sinyali, giriş 0 ın tümleyeni yardımıyla elde edilir. 4 MUX S_1 ve S_0 iki seçim girişi tarafından denetlenir. *C* giriş elde si daha az önemli yerdeki tam toplayıcının elde girişine bağlanır. Diğer elde ler bir tam toplayıcıdan diğerine bağlanır.

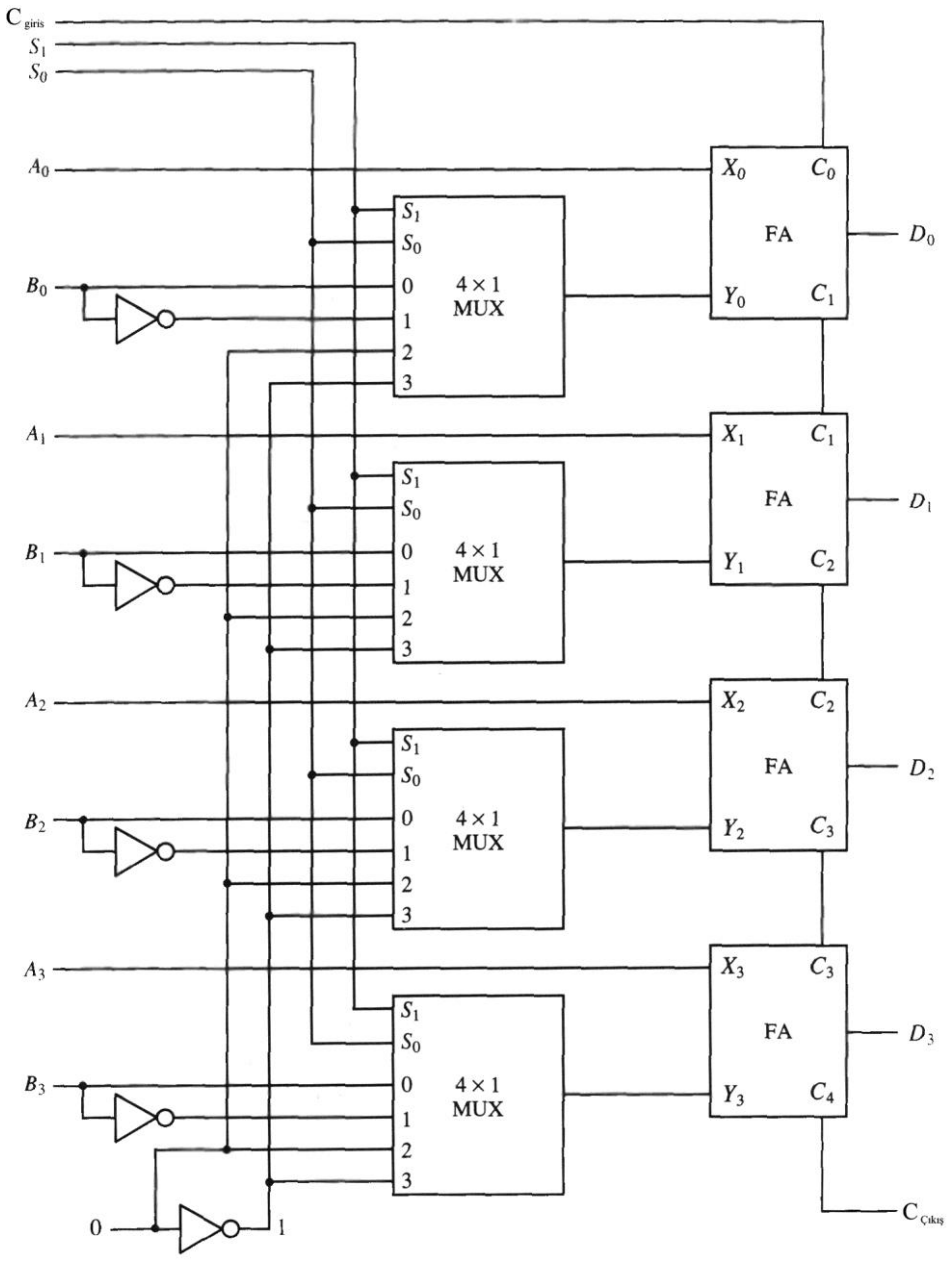
İkili toplayıcının çıkışı aşağıdaki aritmetik formülden hesaplanır.

$$D = A + Y + C_{\text{giriş}}$$

Burada *A* FA nın *X* girişindeki 4 bitlik ikili değeri ve *Y* ise FA nın *Y* girişindeki 4 bitlik ikili değeridir. $C_{\text{giriş}}$'in 0 veya 1 e eşit olması ile çizelge 4.4 teki 8 farklı mikro işlemi yapmak mümkündür.

Toplama

$S_1S_0 = 00$ olduğunda, *B* değeri toplayıcının *Y* girişine uygulanır. Eğer $C_{\text{giriş}} = 0$ ise çıkış $D = A + B$ dir. Eğer $C_{\text{giriş}} = 1$ ise çıkış $D = A + B + 1$ dir. Her iki durumda da giriş elde si ne olursa olsun toplama mikro işlemi gerçekleşir.



Şekil 4.9 4 bit aritmetik devre

Çizelge 4.4 Aritmetik devre fonksiyon çizelgesi

Seçim			Giriş	Çıkış	
S_1	S_0	$C_{\text{Giriş}}$	Y	$D = A + Y + C_{\text{giriş}}$	Mikro işlem
0	0	0	B	$D = A + B$	Toplama
0	0	1	B	$D = A + B + 1$	Elde ile toplama
0	1	0	$\overline{B}$	$D = A + \overline{B}$	Borç ile çıkarma
0	1	1	$\overline{B}$	$D = A + \overline{B} + 1$	Çıkarma
1	0	0	0	$D = A$	A nın aktarımı
1	0	1	0	$D = A + 1$	A nın değerini 1 arttırma
1	1	0	1	$D = A - 1$	A nın değerini 1 azaltma
1	1	1	1	$D = A$	A nın aktarımı

Çıkarma

$S_1S_0 = 01$ ise B nin tümleyeni toplayıcının Y girişine uygulanır. Eğer $C_{\text{giriş}} = 1$ ise $D = A + \overline{B} + 1$ olur. Bu da $A - B$ çıkarımına denktir. (2 ye göre tümleyen aldığından). $C_{\text{giriş}} = 0$ ise $D = A + \overline{B}$ dir. Buda $A - B - 1$ ile gösterilen borç ile çıkarmaya eşittir.

Arttırma

$S_1S_0 = 10$ ise B girişleri devre dışı kalarak tüm Y girişlerine 0 değeri aktarılır. Çıkış $D = A + 0 + C_{\text{giriş}}$ haline gelir. Bu $C_{\text{giriş}} = 0$ olduğunda $D = A$ yı $C_{\text{giriş}} = 1$ ise $D = A + 1$ i verir. İlk durumda A girişinden D çıkışına doğrudan aktarma vardır. İkinci durumda A nın değeri 1 artar.

Azaltma

$S_1S_0 = 11$ olduğunda tüm 1 ler Y nin içine aktarılır. $C_{\text{giriş}} = 0$ olduğunda $D = A - 1$ azaltma işlemi yapılır. $C_{\text{giriş}} = 1$ ise $D = A - 1 + 1 = A$ olur. Buda A girişinden D çıkışına doğrudan aktarımı göstermektedir. Görüldüğü gibi $D = A$ mikro işlemi 2 defa üretildiğinde aritmetik devrede 7 farklı mikro işlem vardır.

4.5 Mantıksal Mikro İşlemler

Mantıksal mikro işlemler yazaçlarda saklanan bit dizgileri için kesinlikle tanımlıdır. Bu işlemler ayrı ayrı yazaçların her bir bitini içermekte ve onlara ikili değişkenler gibi davranmaktadır. Örneğin ÖZEL-VEYA mikro işlemi $R1$ ve $R2$ yazaçlarının içerikleri ile sembolik olarak

$$P: R1 \leftarrow R1 \oplus R2$$

Olur. Denetim değişkeni $P = 1$ olduğunda bir mantıksal mikro işlem yazaçlardaki tek bitler üzerinde kesinlikle çalışır. Nümerik örnek verilirse, her bir yazacın 4 bit olduğu, $R1$ in içeriğinin 1010 ve $R2$ nin içeriğinin 1100 olduğu kabul edilsin. ÖZEL-VEYA mikro işlemi sembolik biçimin dışında mantıksal hesaplama ile ifade edilebilir.

1010 $R1$ in içeriği
1100 $R2$ nin içeriği
0110 $P = 1$ olduğunda $R1$ in içeriği

$R1$ in önceki değeri ile $R2$ nin bitlerine bit bit ÖZEL-VEYA fonksiyonu uygulandığında mikro işlemin çalışması sonucunda $R1$ in içeriği değişir. Mantıksal mikro işlemler bilimsel hesaplamalarda nadiren kullanılırlar, fakat mantıksal karar verme ve ikili bilgilerin idaresi için oldukça uygundurlar.

Özel Semboller

VEYA, VE ve TÜMLEYEN mantıksal mikro işlemleri için özel semboller kullanılır. Bu sembollerden uygun olanları Boole fonksiyon ifadelerinde kullanılır. $\vee$ sembolü VEYA mikro işlemini ve $\wedge$ sembolü de VE mikro işlemini göstermektedir. Tümleyen işlemi 1 in tümleyeninde olduğu gibi yazaç adının üzerine çizgi çizerek belirtilir. Farklı semboller kullanarak, mantıksal VEYA işlemi yerine aritmetik + işareti de kullanılmaktadır. + sembolünün iki anlamı vardır. Birincisi bir mikro işlemde + sembolü aritmetik toplamayı, bir denetim veya Boole fonksiyonunda bir VEYA işlemini tanımlamaktadır. Burada VEYA mikro işleminde + işareti kullanılmayacaktır. Örneğin,

$$P + Q : R1 \leftarrow R2 + R3, \quad R4 \leftarrow R5 \vee R6$$

durumunda $P + Q$ arasında + işareti iki ikili değişken arasındaki bir denetim fonksiyonunu göstermektedir. $R2$ ve $R3$ arasındaki + işareti toplama mikro işlemini tanımlamaktadır. $R5$ ve $R6$ yazaçları arasındaki $\vee$ sembolü VEYA mikro işlemini tanımlamaktadır.

Mantıksal Mikro İşlemlerin Listesi

İki ikili değişken ile çalıştırılan 16 farklı mikro işlem vardır. x, y gibi iki ikili değişkenle çizelge 4.5 te gösterilen doğruluk çizelgesi kullanılarak farklı tüm mümkün mantıksal durumları elde edilir. Bu çizelgede F_0 dan F_{15} e kadar 16 alan bulunmakta ve her bir sütun ayrı Boole fonksiyonunu göstermektedir.

Çizelge 4.5 iki değişkenin 16 fonksiyonu için doğruluk çizelgesi

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Çizelge 4.6 16 mantıksal mikro işlem

Boole fonksiyonu	Mikro işlem	İsmi
$F_0 = 0$	$F \leftarrow 0$	silme
$F_1 = xy$	$F \leftarrow B \wedge A$	VE
$F_2 = xy'$	$F \leftarrow A \wedge B$	
$F_3 = x$	$F \leftarrow A$	A nın aktarılması
$F_4 = x'y$	$F \leftarrow \overline{A} \wedge B$	
$F_5 = y$	$F \leftarrow B$	B nin aktarılması
$F_6 = x \oplus y$	$F \leftarrow A \oplus B$	ÖZEL-VEYA
$F_7 = x + y$	$F \leftarrow A \vee B$	VEYA
$F_8 = (x + y)'$	$F \leftarrow \overline{A \vee B}$	VEYA-DEĞİL
$F_9 = (x \oplus y)'$	$F \leftarrow \overline{A \oplus B}$	ÖZEL-VEYA-DEĞİL
$F_{10} = y'$	$F \leftarrow \overline{B}$	B nin tümleyeni
$F_{11} = x + y'$	$F \leftarrow A \vee \overline{B}$	
$F_{12} = x'$	$F \leftarrow \overline{A}$	A nın tümleyeni
$F_{13} = x' + y$	$F \leftarrow \overline{A} \vee B$	
$F_{14} = (x\ y)'$	$F \leftarrow \overline{A \wedge B}$	VE-DEĞİL
$F_{15} = 1$	$F \leftarrow 1$	Tüm bitlere 1 aktarılması

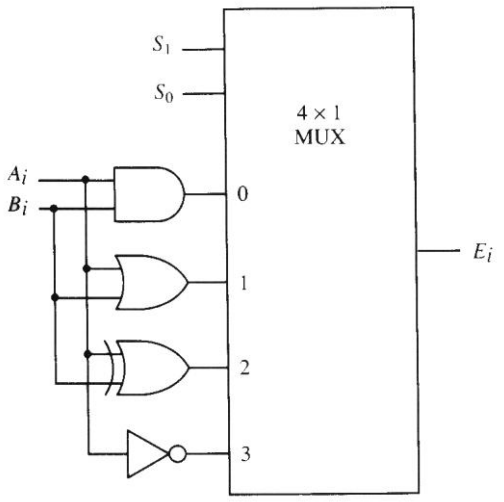
x ve y iki değişken 16 Boole fonksiyonu cebirsel biçimde çizelge 4.6 nın 1. sütununda ifade edilmektedir. 16 mantıksal mikro işlem x değişkeninin yerine A yazacının ikili içeriği, y değişkeninin yerine de B yazacının ikili içeriği kullanılarak tanımlanmaktadır. 2. sütunda A ve B yazaçları kullanılarak mantıksal mikro işlemlerin listesi verilmektedir. Yazacın her bir biti bir ikili değişken gibi işlem görmekte ve mikro işlemler yazaçlarda yüklü bitler üzerine uygulanmaktadır.

Donanımın Kurulumu

Mantıksal mikro işlemlerin donanım yapısı için mantıksal kapılara gerek vardır. 16 mantıksal mikro işlem olmasına rağmen, çoğu bilgisayarlarda sadece dördünden diğer tümü elde edilebilir Bu dördü VE, VEYA,ÖZEL-VEYA ve TERSLEYEN dir.

Mantıksal Devre

Şekil 4.10 devrenin bir bölümünü göstermektedir. Bu bölüm 4 temel mantıksal mikro işlemi yapmaktadır. 4 mantıksal işlemin her biri bir kapı vasıtasıyla yerine getirilir. Bu kapı, yapılacak işlemin mantıksal kapısıdır. Kapıların çıkışları bir MUX a veri girişi olarak uygulanmaktadır. S_1 ve S_0 seçim girişleri MUX un veri girişlerinden birini çıkışa doğrudan yönlendirir. n bitlik bir mantıksal devre için şekil $i = 0 \ldots n–1$ e kadar n kez tekrarlanmaktadır. Seçim telleri tüm kademelere uygulanır. Şekil 4.10(b) deki fonksiyon çizelgesi seçim tellerinin her bir birleşimi için elde edilen mantıksal mikro işlemlerin listesidir.



a) Mantık Şeması

S_1	S_0	Çıkış	İşlem
0	0	$E = A \wedge B$	VE
0	1	$E = A \vee B$	VEYA
1	0	$E = A \oplus B$	ÖZEL-VEYA
1	1	$E = \overline{A}$	TÜMLEYEN

b) Fonksiyon Çizelgesi

Şekil 4.10 Mantıksal devrenin bir kademesi

Bazı Uygulamalar

Mantıksal mikro işlemlerde yazaçlarda yüklü kelimenin bir bölümü veya tek bir bit işlem görmektedir. Mikro işlemler bit değerlerini değiştirmeyi, bir grup biti silmeyi veya bir yazacın içine yeni bir bit sokmayı sağlarlar. Aşağıdaki örnekler *A* yazacı ile *B* yazacı içindeki bilgilere mantıksal mikro işlemlerin uygulanmasını göstermektedir. Tipik bir uygulama, işlem yazacı *A* ve mantıksal veri içeren yazaç *B* (bellekten çıkarılan) nin bitleri kıyaslanmakta.

Seçimlilik-belirleme

Seçimlilik-belirleme işlemleri *A* yazacındaki bitleri 1 yapmak için *B* yazacında uygun 1 ler bulunmaktadır. *B* deki 0 lar bit değişimini etkilemez. Aşağıdaki örnek bu işlemi açıklamaktadır.

1010	önceki *A*
1100	*B* (mantıksal işlem)
1110	işlemden sonraki *A*

A nın uygun bitlerini 1 yapmak için *B* nin en soldaki iki biti 1 alınmıştır. Buradaki iki bitten birisi zaten 1 dir, diğeri 0 iken 1 oldu. *A* nın 2 biti *B* deki 0 lardan dolayı olduğu gibi kalmıştır. Doğruluk çizelgesinden *A* nın önceki değerleri ile *B* deki bitlerin mantıksal-VEYA lanması ile *A* nın işlemden sonraki yeni değerleri elde edilir.

Seçimlilik-tümleyen

Seçimlilik-tümleyen işleminde *B* deki uygun bitler ile *A* nın istenilen bitlerinin tümleyeni alınır. *B* deki bitler 0 olduğunda tümleyen almada hiç bir etkileri olmaz. Örneğin:

1010	önceki *A*
1100	*B* (mantıksal işlem)
0110	işlemden sonraki *A*

Burada *B* nin en soldaki iki biti 1 dir. Bunlar *A* nın iki bitinin tümleyenini almak için kullanılırlar. Seçimlilik-tümleyen işlemi bir ÖZEL-VEYA mikro işlemi gibidir.

Seçimlilik-silme

Seçimlilik-silme işlemi *B* deki uygun 1 ler vasıtasıyla *A* daki istenilen bitleri 0 yapar. Örneğin:

1010	önceki *A*
1100	*B* (mantıksal işlem)
0010	işlemden sonraki *A*

B nin en soldaki iki biti 1 olduğunda *A* nın istenilen bitleri sıfırlanmaktadır. Burada AB' Boole fonksiyonu çalışmaktadır. Uygun mantıksal mikro işlem $A \leftarrow A \wedge B$ dir.

Maskeleme

Maskeleme işlemi seçimlilik-silme işlemine benzer ancak *B* deki uygun 0 lar vasıtasıyla *A* daki bitler silinir (sıfırlanır). Maskeleme işlemi örnekte görüldüğü gibi:

1010	önceki *A*
1100	*B* (mantıksal işlem)
1000	işlemden sonraki *A*

Örneğin *B* nin en sağındaki 2 bit 0 olduğundan *A* nın en sağındaki uygun 2 bit silinir. *B* nin en solundaki iki bit 1 olduğu için *A* nın solundaki iki bitte değişiklik olmaz. Maskeleme işlemi seçimlilik-silme işleminden çok daha kullanışlıdır. Çoğu bilgisayarlarda bir AND buyruğu bulunmakta, ancak bir kaç bilgisayarda seçimlilik-silme mikro işlemini çalıştıracak buyruk bulunmaktadır.

Araya ekleme

Araya ekleme işlemi bir grup bitin içine yeni bir değer almayı sağlar. Önce değiştirilecek değerler maskeleme işlemi ile silinir, ve elde edilen değer ile gerek duyulan değer VEYA lanır. Örneğin *A* yazacının içeriği 8 bit ve 0110 1010 olsun. En soldaki 4 bit 1001 ile değiştirilmek istendiğinde, istenilmeyen 4 bit maskelenmektedir.

0110 1010	*A* önceki
0000 1111	*B* (maske)
0000 1010	*A* maskelemeden sonra

ve sonra yeni değer araya eklenir.

0000	1010	*A* önceki
1001	0000	*B* (maske)
1001	1010	*A* maskelemeden sonra

Maskeleme işlemi VE mikro işlemi, araya ekleme işlemi VEYA mikro işlemdir.

Silme işlemi *A* ve *B* kelimelerini kıyaslamakta ve iki değer eşit olduğunda sıfır değerini üretmektedir. Bu işlem aşağıda gösterildiği gibi bir ÖZEL-VEYA mikro işlemi ile yapılır.

1010	*A*
1010	*B*
0000	$A \leftarrow A \oplus B$

İki sayının tüm bitleri aynı olduğundan ÖZEL-VEYA sonucunda A nın tüm bitleri 0 değerini alır.

4.6 Kaydırma Mikro İşlemleri

Kaydırma mikro işlemleri verilerin seri olarak aktarılmasında kullanılır. Bunun yanında aritmetik, mantıksal ve diğer veri işleme işlemleriyle birleştirilerek de kullanılır. Bir yazacın içeriği sağa veya sola kaydırılabilir. Herhangi bir zamanda bitler kaydığında birinci yaz-boz seri girişte ikili bilgiyi alır. Seri girişlerle bilgi aktarımında kaydırmanın tipi tanımlıdır. Mantıksal, dairesel ve aritmetik olmak üzere 3 tip kaydırma vardır.

Mantıksal Kaydırma

Bir mantıksal kaydırma seri giriş vasıtasıyla 0 değerini alır. Mantıksal sola kaydırma ve sağa kaydırma işlemlerinde sembol olarak shl ve shr kullanılır. Örnekte

$$R1 \leftarrow \text{Shl } R1$$
$$R2 \leftarrow \text{Shr } R2$$

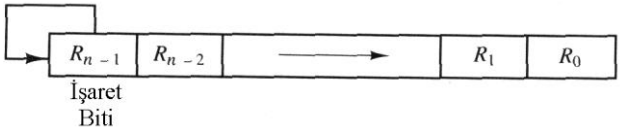
iki mikro işlem tanımlanmakta, birincisi *R*1 yazacının içeriğini bir bit sola, ikincisi *R*2 yazacının içeriğini bir bit sağa kaydırmaktadır. Yazaçların gösteriminde her ikisinde de okun yönü aynı taraftadır. Seri giriş vasıtasıyla bit aktarımında, mantıksal kaydırma boyunca bitiş yerindeki bitin değeri 0 sayılır.

Dairesel Kaydırma

Dairesel kaydırma (dönme işleri olarak ta bilinir), iki bitiş biti ile birlikte, bilgi kaybı olmadan, bilgi yazacın etrafında dairesel olarak döner. Dairesel kaydırmada, kaydırma yazacının seri çıkışı, bu yazacın seri girişine bağlanmaktadır. Dairesel sola kaydırmada cil dairesel sağa kaydırmada da cir sembolleri kullanılır. Kaydırma mikro işleminin sembolik gösterimi çizelge 4.7 de gösterilmiştir.

Çizelge 4.7 kaydırma mikro işlemleri

Sembolik gösterim	tanımlama
$R \leftarrow$ Shl R	R yazacında sola kaydırma
$R \leftarrow$ Shr R	R yazacında sağa kaydırma
$R \leftarrow$ cil R	R yazacında dairesel sola kaydırma
$R \leftarrow$ cir R	R yazacında dairesel sağa kaydırma
$R \leftarrow$ ashl R	R yazacında aritmetik sola kaydırma
$R \leftarrow$ ashr R	R yazacında aritmetik sağa kaydırma



Şekil 4.11 Aritmetik sağa kaydırma

Aritmetik Kaydırma

Bir aritmetik kaydırma işaretli ikili sayıyı sağa veya sola doğru kaydıran bir mikro işlemdir. Bir aritmetik sola kaydırma işaretli sayıyı 2 ile çarpar, sağa kaydırmada işaretli sayıyı 2 ye böler. Aritmetik kaydırmada işaret biti olduğu yerde kalır, değişmez. Çünkü normalde sayı 2 ile çarpıldığında veya bölündüğünde işareti değişmez. Yazacın en solundaki bit işaret bitini saklamaktadır. İşaret biti 0 olduğunda sayı pozitif, 1 olduğunda sayı negatiftir, negatif sayı 2 ye göre tümleyen biçimindedir. Şekil 4.11 n bitlik bir yazacı göstermektedir.

R_{n-1} inci bit en solda bulunmakta ve işaret bitini tutmaktadır. R_{n-2} sayının en anlamlı biti R_0 ise sayının en anlamsız bitidir. Aritmetik sağa kaydırma işaret bitine dokunmadan diğer bitleri sağa kaydırır, R_0 biti kaybolur.

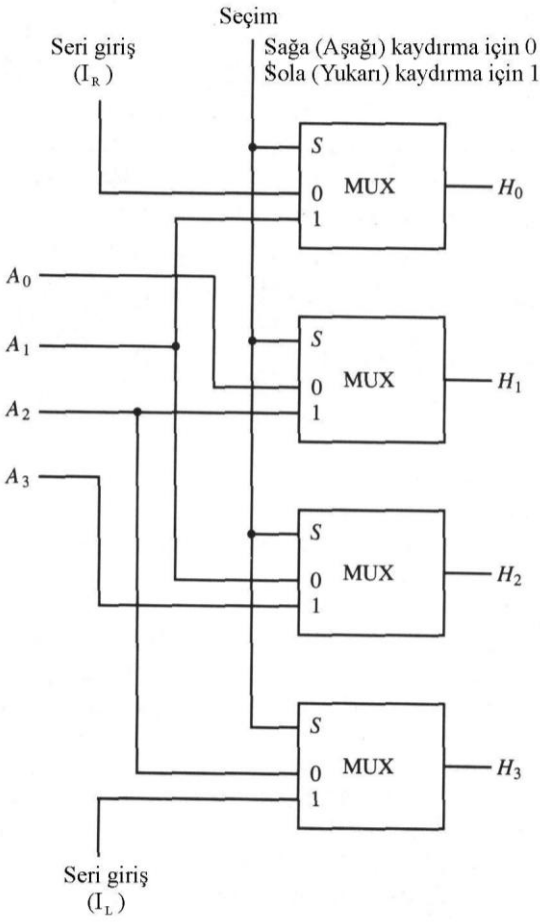
Aritmetik sola kaydırmada R_0 ın içine 0 değeri gelir, ve tüm bitler sola kaydırılır. Başlangıç biti R_{n-1} kaybolur ve R_{n-2}, R_{n-1} in yerine geçer. Kaydırma işlemi sonucunda R_{n-1} in değeri değişmiş ise tersine çevrilir. Eğer 2 ile çarpma işleminde taşma oluyorsa işaret değişir. Kaydırma işleminden önce R_{n-1} ve R_{n-2} bitlerinin değerleri eşit değilse sola kaydırma işleminden sonra taşma olur. Bir taşma yaz-bozu V_s kullanarak bir sola kayma işlemi sonunda taşma denetlenebilir.

$$V_s = R_{n-1} \oplus R_{n-2}$$

Eğer $V_s = 0$ ise taşma olmaz, $V_s = 1$ ise taşma vardır ve kayma işleminden sonra işaret biti ters çevrilir. Yazaçlarda kayma yapıldığında bazı saat vuruşlarında taşma V_s yaz-bozunun içine aktarılır.

Donanım Kurulumu

Kaydırma birimi için uygun bir seçim iki yönlü paralel yüklemeli kaydırma yazaçları kullanmaktadır. Bilgi yazaçlarda paralel olarak aktarılabilir. Ve sağa veya sola kaydırılabilir. Bu tip bir donanım verileri yazacın içine yüklemek için saat vuruşuna gerek duyar ve bir sonraki saat vuruşunda kaydırma işlemine başlar. Bir çok yazaca sahip bir işlemci, bir birleşik devre ile kaydırma işlemini etkin şekilde yerine getirmektedir. Bu yolla bir kez kaydırılmış olan yazacın içeriği ortak veri yoluna aktarılmakta ve sayı kaydırıldıktan sonra tekrar geriye yazacın içine yüklenmektedir. (Ortak veri yolu birleşik kaydırıcı ile bağlantılıdır).



Fonksiyon Çizelgesi

Seçim S	Çıkış H0	H1	H2	H3
0	I_R	A_0	A_1	A_2
1	A_1	A_2	A_3	I_L

Şekil 4.12 4 bit birleşik kaydırıcı devre

Kaydırıcı

MUX larla yapılan bir birleşik kaydırıcı devre şekil 4.12 de gösterilmektedir. 4 bit kaydırıcı A_0 dan A_3 e kadar 4 veri girişine ve H_0 dan H_3 e kadar 4 veri çıkışına sahiptir. I_L sola kaydırma ve I_R sağa kaydırma olmak üzere iki seri giriş vardır. Seçim girişi $S = 0$ olduğunda giriş verisi sağa (şekilde aşağı), $S = 1$ olduğunda giriş verisi sola (şekilde yukarı) doğru kaymaktadır. Çizelge 4.12 girişleri ve kaydırma sonucu çıkışları göstermektedir. n veri giriş ve n çıkışlı bir kaydırıcıda n MUX a gerek vardır.

4.7 Aritmetik Mantık Kaydırma Birimi

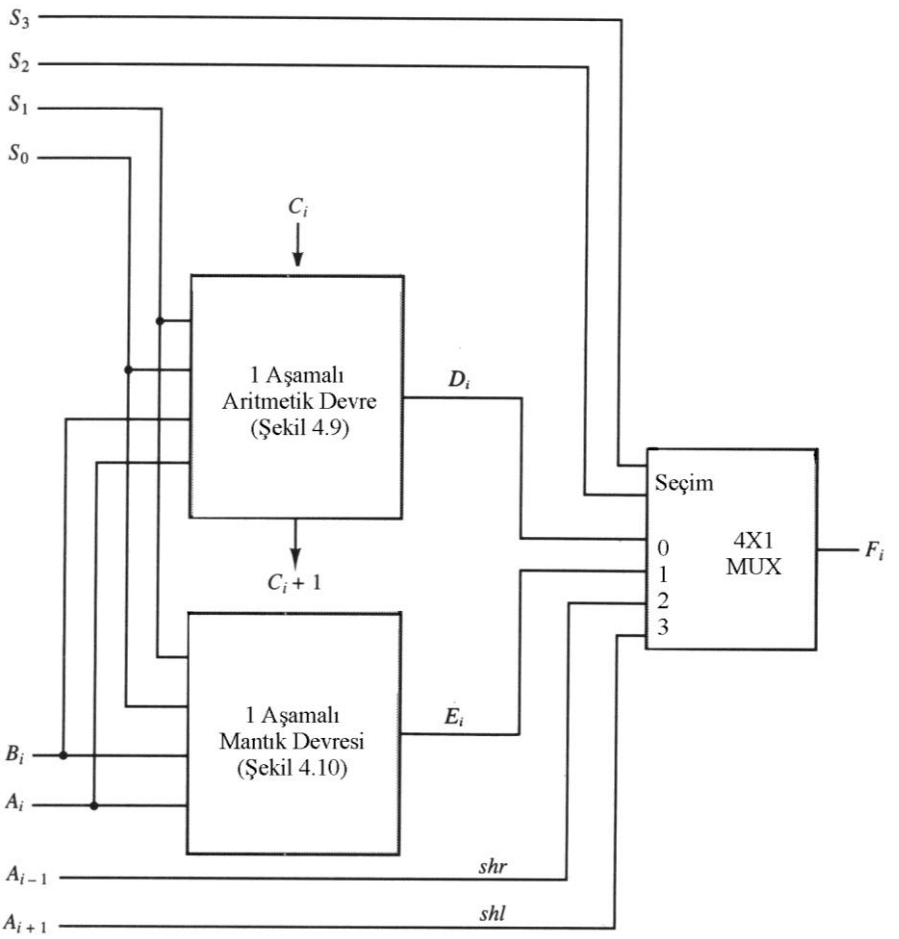
Mikro işlemleri doğrudan gerçekleştirmek için ayrılmış yazaçlar yerine, bilgisayar sistemleri, aritmetik mantık birimi (ALU) adı verilen genel işlemli birime bağlantılı bir dizi depolama yazaçları kullanır. Bir mikro işlemi gerçekleştirmek için belirli yazaçların içerikleri ALU nun girişine uygulanır, ALU işlemi gerçekleştirir ve sonucu bir sonuç (varış) yazacına aktarır. ALU birleşik bir devredir ve böylece bütün yazaç aktarım işlemleri bir saat vuruşu sürecinde gerçekleştirilebilir. Kaydırma mikro işlemleri çoğunlukla ayrı bir birimde gerçekleştirilir, fakat bazen kaydırma birimi ALU nun bir bölümü olarak yapılır.

Aritmetik, mantık ve kaydırma devreleri, (ortak genel) seçim değişkenleri ile bir ALU içinde bir araya getirilebilir. Aritmetik mantık kaydırma biriminin bir kademesi şekil 4.13 te gösterilmiştir, i indisi bir kademeyi tanımlamaktadır. A_i ve B_i girişleri hem aritmetik, hem de mantık birimlerinin girişlerine uygulanır.

S_1 ve S_0 seçim girişleri bir mikro işlem seçerler. Bir 4 x 1 MUX aritmetik çıkış D_i ve mantıksal çıkış E_i den birisini seçer. MUX ların veri girişleri S_3 ve S_2 girişleri ile seçilir. MUX un diğer 2 veri girişi sağa kaydırma işlemi için A_{i-1} girişini ve sola kaydırma işlemi için A_{i+1} girişini almaktadır. Bu şema sadece bir kademeyi göstermektedir. n bitlik bir ALU için şekil n kez tekrarlanmaktadır. Bir aritmetik kademenin C_{i+1} çıkış elde si, bir sonraki ardışık işlemlerin giriş elde si C_i ile ilişkilidir. Birinci kademenin giriş elde si $C_{\text{giriş}}$ tir. Bu aritmetik işlemler için bir seçim değişkeni içermektedir.

Şekil 4.13 te tanımlanan tek kademeli devre 18 aritmetik işlemi içerir, (5 mantıksal, 3 kaydırma işlemi olmak üzere). Her bir işlem, S_3, S_2, S_1, S_0 ve $C_{\text{giriş}}$ den oluşan 5 değişken ile seçilir. $C_{\text{giriş}}$ sadece bir aritmetik işlemi seçmek için kullanılır.

Çizelge 4.8 ALU daki 14 işlemin listesidir. İlk sekiz matematik işlem (çizelge 4.4) $S_3S_2 = 00$ olduğunda seçilir. $S_3S_2 = 01$ olduğunda sonraki 4 mantıksal işlem seçilir (şekil 4.10). Giriş eldesi x ile işaretlenmiştir ve mantıksal işlem boyunca bir etkiye sahip değildir. $S_3S_2 = 10$ olduğunda son iki kaydırma işlemi yapılır. Kaydırmada diğer 3 girişin bir etkisi yoktur.



Şekil 4.13 Aritmetik mantık kaydırma biriminin bir kademesi

Çizelge 4.8 Aritmetik mantıksal kaydırma biriminin fonksiyon çizelgesi

İşlem seçimi						
S_3	S_2	S_1	S_0	$C_{giriş}$	İşlem	Fonksiyonu
0	0	0	0	0	$F = A$	A F'e aktarılıyor
0	0	0	0	1	$F = A + 1$	A nın değeri 1 artıyor
0	0	0	1	0	$F = A + B$	Toplama
0	0	0	1	1	$F = A + B + 1$	Elde ile toplama
0	0	1	0	0	$F = A + \overline{B}$	Ödünç alma ile çıkarma
0	0	1	0	1	$F = A + \overline{B} + 1$	Çıkarma
0	0	1	1	0	$F = A - 1$	A nın değerini bir azaltma
0	0	1	1	1	$F = A$	A nın değerini F'e aktarma
0	1	0	0	×	$F = A \wedge B$	VE
0	1	0	1	×	$F = A \vee B$	VEYA
0	1	1	0	×	$F = A \oplus B$	ÖZEL-VEYA
0	1	1	1	×	$F = \overline{A}$	A nın tümleyeni
1	0	×	×	×	F = shr A	A sağa kaydırılarak F'e aktarılıyor
1	1	×	×	×	F = shl A	A sola kaydırılarak F'e aktarılıyor

SORULAR

4-1. Şekil 4.2(a) ya benzer şekilde aşağıdaki yazaç aktarım işlemlerini gerçekleştiren donanımın blok şemasını çiziniz.

$$yT_2: R2 \leftarrow R1, R1 \leftarrow R2$$

4-2. $R0$, $R1$, $R2$, $R3$ yazaçlarının çıkışları 4x1 lik seçicilere oradan da bir başka yazaç $R5$ e giriş olarak gelmektedir. Her bir yazaç 8 bit uzunluğundadır. T_0, T_1, T_2, T_3 zaman değişkenleri yazaç aktarım işlemlerini aşağıdaki gibi gerçekliyor.

$$T_0: R5 \leftarrow R0$$
$$T_1: R5 \leftarrow R1$$
$$T_2: R5 \leftarrow R2$$
$$T_3: R5 \leftarrow R3$$

Zaman değişkenlerinden biri 1 iken diğeri 0 olduğuna göre bu yazaç aktarım işlemlerini gerçekleştiren donanımın blok şemasını çizin. Çizimde T_0 dan T_3 e zaman değişkenleri seçicilerin seçim girişlerine ve $R5$ yazacının yükleme girişine uygulansın.

4-3. Aşağıdaki denetim işlemini iki yazaç aktarım işlemi yapan denetim fonksiyonları ile gerçekleyin.

if (P = 1) then ($R1 \leftarrow R2$) else *if* (Q = 1) then ($R1 \leftarrow R3$)

4-4. Şekil 4.3 teki veri yolu sisteminde bir yazaçtan diğerine veri aktarabilmek için ne yapılmalıdır? Özellikle *C* yazacının çıkışlarından *A* yazacının girişlerine bir yol bulan bağlantıları gösterin.

4-5. Şekil 4.3 teki veri yolu sistemine benzer fakat MUX lar yerine üç durumlu buffer kapıları ve kod çözücüler kullanarak hazırlanmış veri yolu sistemini çiziniz.

4-6. Bir sayısal bilgisayar sisteminde seçicilerle oluşturulmuş 32 bitlik 16 yazaçlı veri yolu sistemi bulunuyor.
a. Her seçici için kaç seçim girişi kullanılır?
b. Seçicilerin boyutu ne olmalıdır?
c. Veri yolunda kaç seçici bulunur?

4-7. Aşağıdaki aktarım işlemleri bellekte bulunmaktadır. Her birini ayrı ayrı açıklayınız.
a. $R2 \leftarrow M[AR]$
b. $M[AR] \leftarrow R3$
c. $R5 \leftarrow M[AR]$

4-8. Aşağıdaki işlemleri gerçekleştiren donanımın blok şemasını çiziniz.

$$x + yz: \quad AR \leftarrow AR + BR$$

Burada *AR* ve *BR* iki adet *n* bitlik yazaç ve *x*, *y*, *z* denetim değişkenleridir. (+işleminin denetim veya Boole işlemlerinde VEYA, mikro işlemlerde de toplamayı temsil ettiği unutulmamalıdır).

4-9. Aşağıdaki işlemleri gerçekleştiren donanımın blok şemasını çizin. Denetim fonksiyonu için mantıksal kapılar, ikili sayıcı için "sayma izin" girişi kullanınız.

$$xyT_0 + T_1 + y'T_2: \quad AR \leftarrow AR + 1$$

4-10. Aşağıdaki yazaç aktarım işlemlerini 4 bitlik $R1$ ve $R2$ yazaçları için inceleyiniz.

$$xT: \quad R1 \leftarrow R1 + R2$$
$$x'T: \quad R1 \leftarrow R2$$

T nin 1 e eşit olması durumunda $x = 1$ ise $R1$ ve $R2$ toplanarak sonuç $R1$ e aktarılacak $x = 0$ ise $R2$ nin içeriği doğrudan $R1$ e aktarılacaktır. İki işlemi de gerçekleştiren donanımın blok şemasını çiziniz. Çizimde 2 tane 4 bit yazaç, 1 tane 4 bit toplayıcı ve $R1$ girişlerini seçmek için dörtlü 2×1 lik MUX kullanınız. Şemada x ve T denetim değişkenlerinin MUX un girişlerini ve $R1$ yazacının yükleme girişini nasıl seçtiğini gösteriniz.

4-10. Şekil 2.11 deki gibi paralel yüklemeli 4 bit sayıcı ve şekil 4.6 daki gibi bir 4 bit toplayıcı kullanarak aşağıdaki işlemlerin nasıl gerçekleştiğini gösteren donanımın blok şemasını çizin. Burada $R1$ yazacı paralel yüklemeli sayıcı ve $R2$ de 4 bitlik yazaçtır.

x: $R1 \leftarrow R1 + R2$
$x'y$: $R1 \leftarrow R1 + 1$

4-11. Şekil 4.7 deki toplayıcı-çıkartıcı devre için işlem ayağı M ve veri girişleri A ve B aşağıdaki değerleri alıyor. Her durumda S_3, S_2, S_1 ve S_0 ve C_4 çıkışlarının alacağı değerleri gösteriniz.

	M	A	B
a.	0	0111	0110
b.	0	1000	1001
c.	1	1100	1000
d.	1	0101	1010
e.	1	0000	0001

4-13. 4 tane tam toplayıcı kullanarak 4 bitlik birleşik azaltıcı devre tasarlayın.

4-14. Şekil 4.9 daki aritmetik devrenin bir IC paketinde bulunduğunu kabul edin. Böyle 2 IC nin 8 bitlik bir aritmetik devre oluşturması için gerekli bağlantıları gösterin.

4-15. Bir S denetim girişli ve iki tane n-bitlik A ve B veri girişli aritmetik devre tasarlayın. Devre elde girişi $C_{\text{giriş}}$ de kullanılarak aşağıdaki 4 aritmetik işlemi yapmaktadır. Devrenin mantıksal şemasını çizin.

S	$C_{\text{giriş}}= 0$	$C_{\text{giriş}}= 1$
0	$D = A + B$ (toplama)	$D = A + 1$ (arttırma)
1	$D = A - 1$ (azalma)	$D = A + \overline{B} + 1$ (çıkarma)

4-16. Çizelge 4.5 te belirtilen 16 mantıksal fonksiyonu seçen ve uygulayan birleşik devreyi tasarlayın.

4-17. ÖZEL-VEYA, ÖZEL-VEYA-DEĞİL, VEYA-DEĞİL ve VE-DEĞİL işlemlerini yapan ve 2 seçim girişi kullanan bir sayısal devre tasarlayın.

4-18. A yazacı 8 bitlik 11011001 ikilik sayısını içeriyor. A yazacının içeriğinin aşağıdaki gibi değişmesi için kullanılacak B yazacının içeriğini ve kullanılacak mantıksal mikro işlemi bulunuz.
a. 01101101
b. 11111101

4-19. 8 bitlik AR, BR, CR ve DR yazaçları aşağıdaki değerleri içeriyor.

$AR = 11110010$
$BR = 11111111$
$CR = 10111001$
$DR = 11101010$

Aşağıdaki işlemler sonunda yazaçların içerikleri ne olur?

AR ← AR + BR	AR ile BR nin toplanması
CR ← CR ∧ DR, BR ← BR + 1	CR ile DR nin VE lenmesi, BR nin 1 artması
AR ← AR – CR	CR nin AR den çıkarılması

4-20. 8 bitlik bir yazaç 10011100 değerini içeriyor. Aritmetik sağa kaydırma işleminden sonra yazacın içeriği ne olur? Yazacın ilk içeriğine bir aritmetik sola kaydırma işlemi uygulanırsa bir taşma olup olmayacağını bulun.

4-21. R = 11011101 değerinden başlayarak sırasıyla; mantıksal sola kaydırma, dairesel sağa kaydırma, mantıksal sağa kaydırma ve dairesel sola kaydırma uygulanırsa R yazacının son içeriği ne olur?

4-22. Şekil 4.12 deki H nin değeri, A girişi 1001, $S = 1$, $I_R = 1$ ve $I_L = 0$ ise ne olur?

4-23. Aşağıdaki yazaç aktarım işlemlerinden hangisi yanlıştır.

a. xT: $AR \leftarrow \overline{AR}$, $AR \leftarrow 0$

b. yT: $R1 \leftarrow R2$, $R1 \leftarrow R3$

c. zT: $PC \leftarrow AR$, $PC \leftarrow PC + 1$

KAYNAKLAR

1. Bell, C. G., J. C. Mudge, and J.E. McNamara, *Computer Engineering*. Bedford, MA: Digital Press, 1980.
2. Booth, T. L., *Introduction to Computer Engineering*. 3rd ed. New York: John Wiley , 1984.
3. Hays, J. F., *Computer Architecture and Organization*. 2nd ed. New York: McGraw-Hill, 1988.
4. Hill, F. J., and G. R. Peterson, *Digital Systems:Hardwire Organization and Desing*, 3rd ed. New York: John Wiley, 1987.
5. Mano, M. M., *Computer Engineering: Hardwire Desing*, Englewood Cliffs, NJ: Prentice Hall, 1988.
6. Patterson, D. A., and J. L. Hennessy, *Computer Architecture :A Quantitative Approach*. San Mateo, CA: Morgan Kaufmann Publishers, 1990.
7. Prosser, F. P., and D. E. Winkel, The Art of Digital Desing, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1987.
8. Sandige , R. S., *Modern Digital Desing*. New York: McGraw-Hill, 1990.
9. Shiva, S. G., *Computer Desing and Architecture*. 2nd ed. New York: HarperCollins Publishers, 1991.
10. Tomek, I., *Introduction to Computer Organization*. Rockville, MD: Computer Science Press, 1981.
11. Ward. S. A., and R. H. Halstead, Jr., *Computer Structures*. Cambridge, MA: MIT Press, 1990.

BÖLÜM 5

Temel Bilgisayar Yapısı ve Devreleri

BU BÖLÜMDE

5-1 Buyruk Kodu
5-2 Bilgisayar Yazaçları
5-3 Bilgisayar Buyrukları
5-4 Zamanlama ve Denetim
5-5 Buyruk Süreci
5-6 Bellek-Adreslemeli Buyruklar
5-7 Giriş-Çıkış Buyrukları ve Kesmeler
5-8 Bilgisayarın Tamamlanmış Tanımı
5-9 Temel Bilgisayarın Tasarımı
5-10 İşlemci Yazacı Mantık Tasarımı

5.1 Buyruk Kodu

Bu bölümde temel bilgisayar tanıtılacak ve işleyişinin yazaç aktarım deyimleri ile nasıl belirlenebileceği gösterilecektir. Bilgisayarın tasarımı, iç yazaçları, zamanlama ve denetim yapısı ve kullandığı buyruk kümesi ile tanımlanmaktadır. Daha sonra bilgisayarın dizaynı detaylandırılacaktır. Bu bölümde tanıtılan temel bilgisayar, ticari bilgisayarlar ile kıyaslandığında, çok küçük olmasına rağmen çok fazla ayrıntıya girmeksizin tasarım sürecini anlatabileceğimiz derecede basit olma avantajına sahiptir.

Bilgisayarın yapısı iç yazaçlar, zamanlama ve denetim yapıları ile ve bunlarca kullanılan buyruk kümesi ile tanımlanır. Daha sonra tasarım ayrıntılarıyla ele alınacaktır. Burada tanımlanacak temel bilgisayar, ticari bilgisayarlara göre daha küçüktür. Fakat basittir ve kolay anlaşılabilir. Ayrıca tasarımı çok daha kolay yapılabilir.

Bir bilgisayarın iç yapısı, yazaçtaki veriler üzerine işlem yapacak mikro işlemler sırasıyla tanımlanmıştır. Genel amaçlı bir sayısal bilgisayar verilerle birçok mikro işlem icra etme yeteneğine sahip olduğu gibi, ayrıca bu mikro işlemlerin hangi sırada icra edileceği konusunda da önceden programlanma yeteneğine sahiptir. Bilgisayarın kullanıcısı, bilgisayarın işlemlerini bir program ile denetler. Program bir dizi buyruklar olup bunlar işlemleri belirtirler. Ayrıca verileri ve hangi işlemin sırasının geldiğini de belirtirler. Verilerin işlenmesi yeni bir programla değiştirilebilir veya aynı program farklı veriler için kullanılabilir.

Bir bilgisayar buyruğu ikili bir koddur. Bir dizi mikro işlemleri tanımlar. Buyruklar verilerle beraber bellekte bulunurlar. Bilgisayar her bir buyruğu bellekten okur bunu denetim yazacına yazar. Denetim bu kodu anlar ve onun gereğini mikro işlemlerin sırasıyla icra ederek yerine getirir. Her bilgisayarın kendi özel buyruk kümesi vardır. Buyrukların bellekte saklanabilmesi ve sonra sırasıyla icra edilmesi, genel amaçlı bilgisayarın en önemli özelliğidir.

Buyruk kodu, İşlem Kodu

Buyruk kodu bir grup bitten ibarettir. Bunlar bilgisayara belli bir işlemi yapmasını söylerler. Genelde buyruk birkaç kısma ayrılır. Buyruğun en önemli parçası işlem parçasıdır. Burada bir kısım bitler toplama, çıkarma, çarpma, kaydırma ve tümleme gibi bazı işlemleri tanımlarlar. Buyruklarının içinde OPR bitlerinin sayısı, bilgisayar içinde mevcut işlemlerin sayısına bağlıdır n tane bit ile 2^n (veya daha az) işlem gösterilebilir. 64 farklı işlem yapabilen bir bilgisayar düşünelim. Bunlardan biri toplama (ADD) işlemi olsun OPR kodu 6 bitten oluşur ve toplama için 110010 dır. Bu buyruğun denetim birimi tarafından kodu çözülünce bilgisayar veriyi bellekten okur ve bunu işlemci yazacında toplar.

Bu noktada bir bilgisayar işlemi ile bir mikro işlem arasındaki farka dikkat edelim. *Mikro işlem;* bilgisayar belleğinde bulunan buyruğun bir parçasıdır. *Buyruk;* Bir ikili kod olup, bilgisayara belli bir işlemi yapmasını söyler. Denetim birimi Buyruğu bellekten alır işlem kodunun bitlerini anlamlandırır. Bundan sonra bilgisayar yazaçlarında bazı mikro işlemleri başlatmak üzere bir dizi denetim sinyali üretir. Her işlem kodu için denetim bir dizi mikro işlemlerin yapılmasını sağlar. Bunlarla belirlenen işlem donanım tarafından yerine getirilir. Bu nedenle bazen işlem kodu mikro işlem olarak adlandırılır, çünkü bir dizi mikro işlem belirler.

Buyruk kodunun işlem kısmı (OPR) icra edilecek işlemi belirler. Bu işlem, yazaç içinde yerleştirilmiş olan veri üzerine yapılmalıdır. Dolayısıyla buyruk kodu sadece işlemi değil aynı zamanda verilerin bulunduğu yazaç ve bellek adresleri ile sonucun yazılacağı yazaç veya bellek adresini de içermelidir. Buyruk kodu içinde bellek kelimeleri adresleri ile bellidir. İşlem yazaçları 2^k tane yazaç için k bit ikili kod ile belirlenir. Buyruk

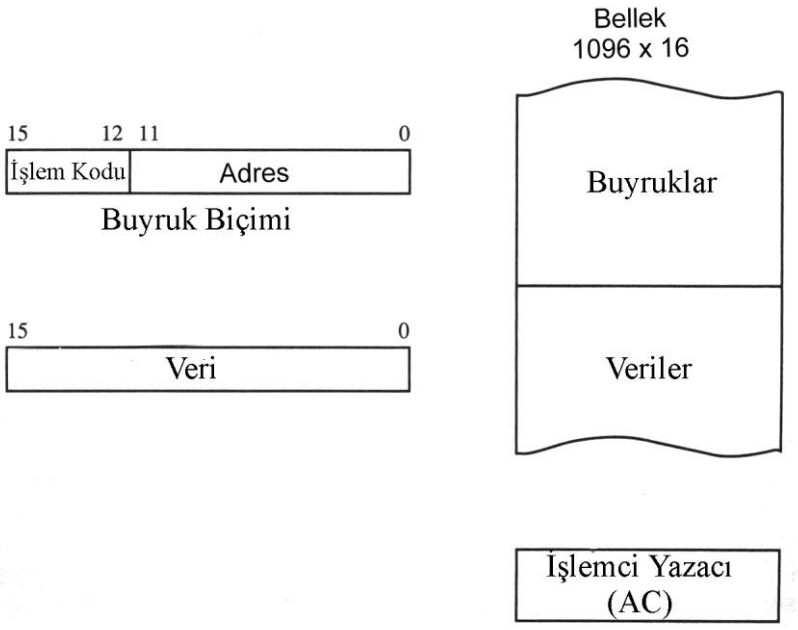
kodlarının oluşturulması için sayısız yöntem mevcuttur. Her bilgisayar kendi özel buyruk kod biçimine sahiptir. Bu biçimler işlemcinin tasarımını ve bilgisayarın mimarisini hazırlayanlar tarafından belirlenirler. Bu bölümde sayısal bilgisayarların tasarımı ve yapısını anlatmak üzere kendimize özel kod biçimi seçeceğiz.

Saklanmış Program Yapısı

Bir bilgisayarı organize etmenin en basit yolu bir işlemci yazacı almak ve iki parçalı bir buyruk biçimi seçmektir. Birinci parça icra edilecek işlemi, ikinci parça ise bir adresi belirler. Buradaki bellek adresi verinin adresini denetim birime bildirir. Bu veri bellekten okunur ve işlemci yazacı içinde bulunan veri ile beraber işleme tabi tutulur.

Şekil 5.1 bu tip bir yapıyı göstermektedir. Buyruklar belleğin bir kısmında, veriler ise başka bir kısmında bulunmaktalar. 4096 kelimelik bir bellek birimi için 12 bit adres uzunluğu gerekmektedir. Çünkü 2^{12} = 4096 dır. Eğer her bir buyruğu 16 bit bellek kelimesine yerleştirecek olursak işlem kodu için 4 bit yerimiz kalır. Böylece 2^4 = 16 buyruk tanımlanır. 12 bit ise verinin adresi için ayrılmıştır. Denetim 16 bit buyruğu, belleğin program kısmından okur. Buyruğun 12 bit adres kısmını kullanarak belleğin veri kısmından 16 bit veriyi okur. Sonra işlem kodu tarafından belirlenen işlemi icra eder.

İşlem Kodu



Şekil 5.1 saklanmış program yapısı

İşlemci Yazacı (AC)

Tek bir işlemci yazacı bulunan bilgisayarlarda bu işlemci yazacına birikeçte denir. İşlem bellekten alınan veri ile *AC* nin içeriği arasında gerçekleştirilir.

Eğer bir buyruk Kodu içindeki işlem, bellekten bir veriye gerek duymuyorsa, bu takdirde buyruk içindeki diğer bitler başka maksatlar için kullanılabilir. Örneğin, *AC* yi temizle, *AC* yi tümle, *AC* yi 1 arttır gibi buyruklar *AC* yazacı içinde mevcut veriye uygulanır. Dolayısıyla bellekten başka bir veriye gerek yoktur. Bu tip işlemler için buyruk kodunun diğer bitlerine (0 dan 11 e kadar) bir bellek adresini göstermesi için gerek yoktur. Bu bitler başka işlemler için kullanılabilir.

Dolaylı Adres

Bazen buyruk kodunun içindeki adres bitleri kısmına bir adres yerine gerçek veriyi yazmak daha uygun olur. Eğer bir buyruğun adres kısmında (ikinci parçası) bir veri varsa buna derhal, ani buyruk adı verilir. Eğer buyruğun ikinci parçasında bir verinin adresi varsa bu buyruk doğrudan adrese sahiptir denir. Bunun dışındaki ihtimal dolaylı adrestir. Bu buyruk içinde ise geçerli verinin adresi vardır. Buyruk Kodu içindeki 1 bit adresin doğrudan veya dolaylı olduğunu belirtir.

Ani Buyruk

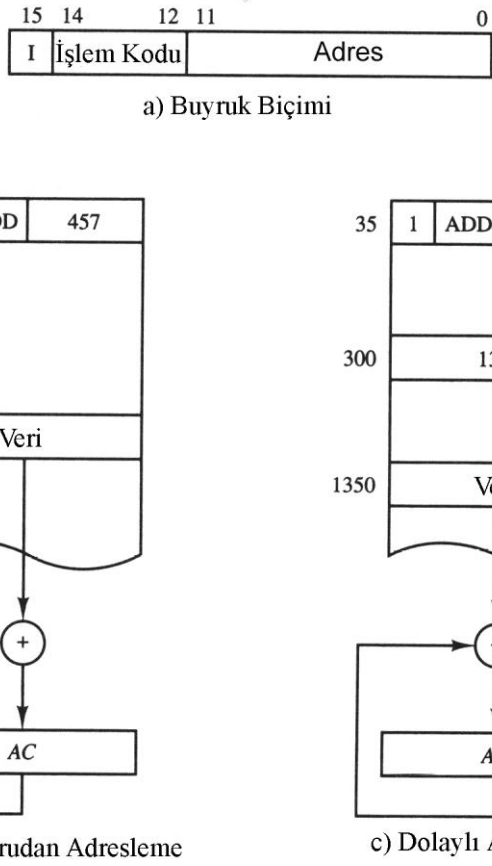
Bu durumun bir örneği olarak şekil 5-2 (a) da gösterilen buyruk kodu biçimini ele alalım. Bu biçimde 3 bit işlem kodu, 12 bit adres ve *I* ile gösterilen 1 bit dolaylı adres göstergesi vardır. Doğrudan adres için kip biti $I = 0$ dır. Dolaylı adres için $I = 1$ olur. Doğrudan adresli bir buyruk şekil 5-2 (b) de gösterilmektedir. Bu buyruk Bellekte 22 adresinde bulunuyor. *I* biti 0 dır. Bu yüzden buyruk doğrudan adresli bir buyruktur. İşlem kodu ADD toplama işlemi buyruğunu gösteriyor. Adres kısmı ise 457 nin ikili karşılığıdır. Denetim 457 adresinde bulunan veriyi *AC* nin içeriğine toplar. 35 adresinde görülen buyruk şekil 5-2 (c) de gösterilmektedir. Bunun kip biti $I = 1$ dir Dolayısıyla dolaylı adreslemeli bir buyruktur.

Adres kısmında 300 adresinin ikili karşılığı var. Denetim, 300 adresine giderek verinin adresini bulmaktadır. Bu durumda verinin gerçek adresi 1350 dir. 1350 adresinde bulunan veri *AC* nin içeriğine eklenir. Dolaylı adresli buyruğun veriyi alması için kaynak gerekmektedir. Birinci kaynak verinin adresinin bulunması, ikicisi ise verinin kendisidir.

Etkin adres

Hesaplama tipi bir buyrukta, verinin adresi, dallanma tipi bir buyrukta ise hedef adres olarak tanımlanır. Bu durumda şekil 5.2 (b) deki buyrukta etkin adres 457, şekil 5.2 (c) deki buyrukta ise 1350 olur.

Bu bölümde tanımlanan bilgisayarda doğrudan ve dolaylı adresleme kipleri kullanılmıştır. Bir bellek kelimesinde, dolaylı adreslemeli bir buy-

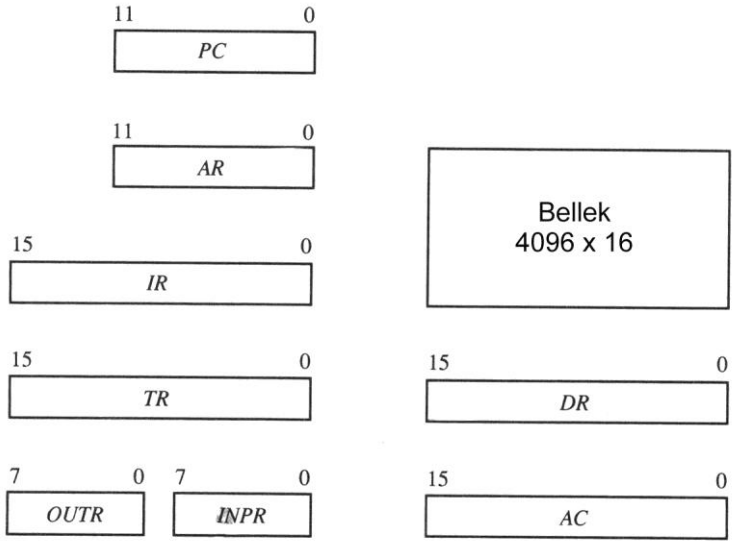


Şekil 5.2 Doğrudan ve dolaylı adresin gösterimi

ruğun verisinin adresinin tutulduğunu düşünelim. Bu bellek kelimesi bir veri dizisi için işaretleyici olarak kullanılır. İşaretleyici bir bellek kelimesinde tutulacağına, ticari bilgisayarlarda olduğu gibi bir işlemci yazacında tutulabilir.

5.2 Bilgisayar Yazaçları

Bilgisayar buyrukları normal olarak ardışık bellek adreslerinde tutulurlar ve bir zamanda sadece bir tanesi icra edilir. Denetim, belleğin belirgin bölgesinden bir buyruk okur ve bunu icra eder. Sonra bir sonraki buyruğu okur ve onu icra eder ve bu böyle devam eder. Böyle sırayla yapılan buyruk sıralamasında bir buyruğun icrasından sonra getirilecek ve icra edilecek buyrukların adresini hesaplamak için bir sayaca gerek vardır. Ayrıca



Şekil 5.3 Temel bilgisayar yazaçları ve bellek

Çizelge 5.1 Temel bilgisayar için yazaçların listesi

Yazaç Sembolü	bit sayıları	Yazaç ismi	Fonksiyonu
DR	16	Veri yazacı	bellek verisini tutar
AR	12	Adres yazacı	bellek için adres tutar
AC	16	Birikeç	İşlemci yazacı
IR	16	Buyruk yazacı	buyruğun kodunu tutar
PC	12	Program sayıcı	buyruğun adresini tutar
TR	16	Geçici yazaç	Geçici veriyi tutar
INPR	8	Giriş yazacı	Giriş karakterini tutar
OUTR	8	Çıkış yazacı	Çıkış karakterini tutar

denetim birim içinde bellekten okunan buyruğun saklanması için bir yazaç bulunması gerekir. Bilgisayarın veriyi işlemek için işlemci yazaçlarına ve bellek adreslerini tutmak için bir yazaca ihtiyacı vardır. Bu gereksinimler şekil 5.3 teki yazaç düzenini oluşturur. Bu yazaçlar çizelge 5.1 de kısa işlevleriyle verilmiştir. Ayrıca bit sayıları da verilmiştir.

Bellek birimi 4096 kelimeden oluşur. Her kelime 16 bittir. Bir buyruğun 12 bitlik kısmı verinin adresi için kullanılır. Üç bit işlem kodu için 1 bit ise doğrudan veya dolaylı adreslemeyi belirlemek için kullanılır. Veri yazacı (*DR*) bellekten okunan veriyi tutar. İşlemci yazacı (*AC*) ise

genel amaçlı bir yazaçtır. Bellekten okunan buyruk, buyruk yazacında (*IR*) tutulur. Geçici yazaç (*TR*) işlem sırasında verilerin geçici olarak tutulmasında kullanılır.

Program Sayıcı (PC)

Bellek adres yazacı (*AR*) 12 bitten oluşur. Çünkü bu Bilgisayarda adres uzunluğu 12 bittir. Program sayıcı (*PC*) de bir adres gösterdiği için 12 bit uzunluğundadır. Bu adres halen icra edilmekte olan buyruğun icrasından sonra bellekten okunacak buyruğun adresidir. *PC*, bir programın çalışması sırasında ardışık değerler alarak, yani sayarak belleğe daha önce yerleştirilmiş buyrukların sırasıyla okunmasını sağlar. Bir dallanma buyruğu, program içinde başka bir buyruk sırasına bir geçişi gösterir. Böyle bir buyruğun adres kısmı *PC* a aktarılır ve bir sonraki buyruğun adresini oluşturur. Başka bir durum olmadığı sürece program bu yeni adresten tekrar sırasıyla oku-icra et döngüsüne girer, *PC* 1 arttırılır ve devam eder.

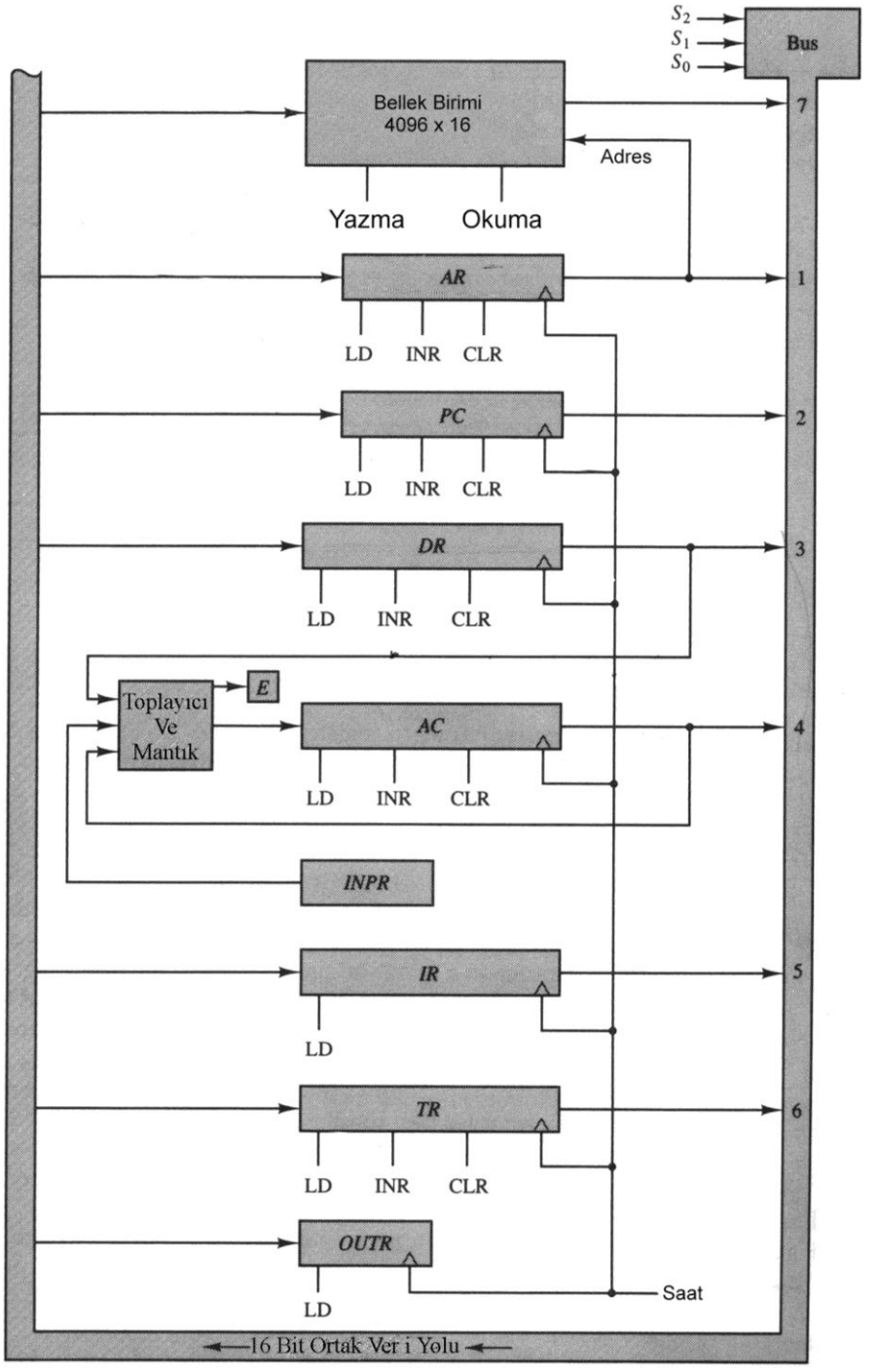
Giriş/çıkış için iki yazaç kullanılır. Bunlar *INPR* ve *OUTR* olup bir çevre biriminden 8 bit alıp saklar veya çevre birimine gönderilecek 8 bit veriyi tutar.

Ortak Veri Yolu Sistemi

Temel bilgisayarımızın 8 yazacı vardır. Bir bellek bulunmaktadır. Bir de burada gösterilmeyen (bölüm 5.4) denetim birimi vardır. Bizim yazaçlar arası ve yazaçlar ile bellek arasında verileri aktarabileceğimiz yollara ihtiyacımız vardır. Eğer bu yazaç çıkışlarını diğer yazaçların girişlerine bağlarsak karmakarışık bir teller ağı oluşur. Bunun daha kolay ve basit şekilde halledilmesi için ortak bir veri yolu kullanılmaktadır. Bölüm 4.3 te MUX lar veya 3 durumlu buffer kapıları kullanarak bir veri yolu sistemi tasarlanmıştı. Belleğin ve yazaçların ortak veri yolu sistemine bağlanmaları şekil 5.4 te gösterilmiştir.

Yükle (LD)

Yedi yazacın çıkışları ve bellek, ortak veri yoluna bağlanmışlardır. Belli, bir anda veri yolu için seçilecek çıkış S_2, S_1 ve S_0 seçim telleri yardımıyla belirlenir. Her yazacın çıkışındaki numara o yazacın seçilmesi için gerekli numaralardır. Örneğin veri yazacının çıkışındaki numara 3 tür. $S_2\ S_1\ S_0 = 011$ olduğunda *DR* nin 16 bit içeriği veri yoluna geçer. Ortak veri yolunun telleri her yazacın ve belleğin girişlerine bağlıdır. Bir yazacın veri yolundan bilgi alabilmesi için *LD* (yükle) girişinin izinlendirilmesi gerekir. Bir sonraki saat vuruşunda veriler veri yolundan yazaca gider. Bellek veri yolunun içeriğini yazma girişi aktiflendiği zaman alır. Belleğin 16 bit çıkışının veri yoluna aktarılması için oku girişinin aktiflenmesi gerekir. $S_2\ S_1\ S_0 = 111$ olmalıdır. 4 yazaç *DR*, *AC*, *IR* ve *TR* nin her biri 16 bitliktir.



Şekil 5.4 Temel bilgisayar yazaçlarının bir ortak veri yoluna bağlanması

AR ve *PC* yazaçları 12 bittir. Çünkü bellek adreslerini tutarlar. *AR* ve *PC* yazaçlarının içerikleri ortak veri yoluna aktarıldığında önemli 4 bit 0 olur. *AR* ve *PC* veri yolundan bilgi aldıklarında 12 önemsiz biti alırlar.

Giriş ve çıkış yazaçlarının (*INPR* ve *OUTR*) her biri 8 bittir. Veri yolunun 8 önemsiz biti yardımıyla haberleşirler. *INPR* veri yoluna veri verir. *OUTR* ise sadece veri yolundan veri alır. Çünkü *INPR* herhangi bir çevre biriminden tek bir karakter alır ve bunu *AC* ye aktarır. *OUTR* ise *AC* den bir karakter alır ve çıkış aletine verir. *OUTR* den, başka yazaçlara aktarım yoktur.

Ortak veri yolunun 16 hattı 6 yazaçtan ve bellekten bilgi alır. Veri yolu hatları 6 yazacın ve belleğin girişlerine bağlanmışlardır. 5 yazacın 3 denetim girişi vardır. Bunlar *LD* Yükle, *INC* arttır ve sil veya temizle *CLR* dir. Bu tip yazacın şekil 2.11 de görülene benzer paralel yüklemeli ve saatli silmeli ikili sayıcıya ihtiyacı vardır. Artma işlemi sayma girişinin izinlendirilmesi ile sağlanmaktadır. İki yazacın sadece *LD* girişi vardır. Bu tip yazaçlar şekil 2.7 de gösterilmiştir.

Bellek Adresleri

Belleğin veri giriş ve çıkışları ortak veri yoluna bağlıdır. Belleğin adresleri ise *AR* ye bağlıdır. Dolayısıyla bir bellek adresini belirlemede (okuma veya yazma için) daima *AR* yazacı kullanılır. Tek bir adres yazacı kullanarak ayrı bir adres yolu gereksiniminden kurtulmuş olduk. Herhangi bir yazacın içeriği bir yazma işlemi sırasında belleğe aktarılabilir. Benzer şekilde *AC* hariç herhangi bir yazaç okuma işlemi ile bellekten bilgi alabilir.

AC nin 16 girişi bir toplayıcı ve mantık biriminden gelir. 16 bitin bir başka gelişi de *AC* nin kendisindendir. Bunlar *AC* üzerinde yapılan işlemler için, örneğin *AC* yi tümle, *AC* yi kaydır gibi işlemler için gereklidir. *AC* ye bir başka veri girişi *DR* veri yazacındandır. *DR* ve *AC* den gelen veriler aritmetik ve mantık mikro işlemlerde kullanılırlar. Örneğin ADD, *DR* yi *AC*'ye ekle veya AND, *DR* ve *AC* yi VE le gibi. Toplamanın sonucu *AC* ye aktarılır. Toplamada taşma olursa, taşma *AC* nin uzantısı olan *E* bitine aktarılır. Üçüncü bir veri kümesi 8 bit olarak *INPR* den gelir. *INPR* ve *OUTR* nin çalışmaları bölüm 5.7 de anlatılacaktır.

Herhangi bir yazacın içeriği veri yoluna aktarılabilir ve toplayıcı ve mantık devrede aynı saat süreci içinde işlem görebilir. Saat süreci sonunda işlemin sonucu olarak veri yolunda bulunan değer belirlenen hedef yazaca aktarılır. Toplayıcı ve mantık devrenin çıkışları, *AC* ye yazılır. Örneğin iki mikro işlem

$$DR \leftarrow AC \quad \text{ve} \quad AC \leftarrow DR$$

işlemleri aynı zamanda icra edilebilir. Bunun yapılabilmesi için *AC* nin içeriğinin veri yoluna aktarılması için $S_2S_1S_0 = 100$ yapılması ve *DR* nin

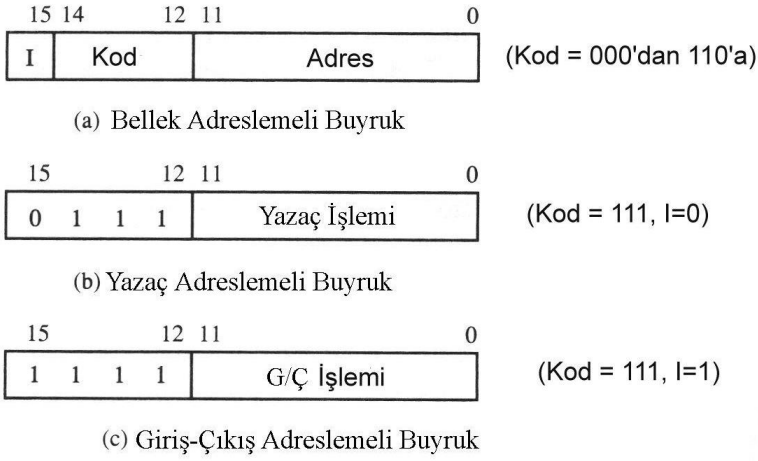
LD girişinin izinlendirilmesi gerekir. *DR* nin içeriğinin *AC* ye aktarılması toplayıcı ve mantık devre üzerinden olur. Bunun için *AC* nin *LD* girişini de izinlendirmek gerekir (aynı saat zamanında).

5.3 Bilgisayar Buyrukları

Buyruk Biçimi

Temel bilgisayarın üç buyruk kod biçimi var. Şekil 5.5. Her biçim 16 bittir. İşlem kodu kısmı üç bittir. Geriye 13 bit kalır. Bellek adreslemeli bir buyruk 12 biti adres için ve 1 biti adres kipi için kullanır. Doğrudan adresleme için *I* = 0 ve dolaylı adres için *I* = 1 dir. (şekil 5.2 ye bakınız). Yazaç adreslemeli buyruklar işlem kodunun 111 olmasıyla ayrılır. En sol bit 0 dır. (15 nolu bit). Yazaç adreslemeli buyruk *AC* üzerinde bir işlem veya *AC* nin test edilmesi işlemlerini yapar. Bellekten veriye gerek yoktur. Dolayısıyla kalan 12 bit yapılacak işlem veya test için kullanılabilir. Benzer şekilde G/Ç işlemi de bellekten bir veriye gerek duymaz. G/Ç işlemi işlem kodu 111 ve en sol bit 1 olmasıyla ayrılır. Kalan 12 bit G/Ç işleminin tipini belirtmek veya test işlemini belirtmek için kullanılabilir.

Buyruğun tipi bilgisayar denetimi tarafından 12'den 15'e kadar olan, yani önemli 4 bitlerin tanınmasıyla belli olur. Eğer buyruğu 12'den 14'e kadar olan 3 biti 111 değilse, buyruk bellek adreslemeli bir buyruktur. 15 nolu bit adresleme kipi olarak kullanılır. Eğer işlem kodu bitleri 111 ise denetim 15



Şekil 5.5 Temel bilgisayar buyruk biçimleri

Çizelge 5.2. Temel bilgisayar buyrukları

Sembol	Onaltılı kodu $I = 0$	$I = 1$	Tanımlama
AND	0XXX	8XXX	Bellekteki kelimeyi VE leyerek *AC* ye aktar
ADD	1XXX	9XXX	Bellek kelimesini toplayarak *AC* ye aktar
LDA	2XXX	AXXX	Bellek kelimesini *AC* ye yükle
STA	3XXX	BXXX	*AC* nin içeriğini belleğe depola
BUN	4XXX	CXXX	Şartsız dallan
BSA	5XXX	DXXX	Dallan ve geri dönüş adresini sakla
ISZ	6XXX	EXXX	Arttır ve eğer sıfır ise atla
CLA		7800	*AC* yi sil
CLE		7400	*E* yi sil
CMA		7200	*AC* nin tümleyenini al
CME		7100	*E* nin tümleyenini al
CIR		7080	*AC* ve E nin içeriğini sağa dairesel olarak kaydır
CIL		7040	*AC* ve E nin içeriğini sola dairesel olarak kaydır
INC		7020	*AC* nin değerini 1 arttır.
SPA		7010	*AC* pozitif ise bir sonraki buyruğu atla
SNA		7008	*AC* negatif ise bir sonraki buyruğu atla
SZA		7004	*AC* sıfır ise bir sonraki buyruğu atla
SZE		7002	*E* sıfır ise bir sonraki buyruğu atla
HLT		7001	Programı durdur
INP		F800	Giriş karakterini *AC* ye al
OUT		F400	*AC* den çıkış karakterini al
SKI		F200	giriş bayrağını atla
SKO		F100	çıkış bayrağını atla
ION		F080	Kesmeyi aktif yap
IOF		F040	Kesmeyi pasif yap

nolu bite (en önemli bit) bakar. Bu 0 ise buyruk yazaç adreslemeli tiptedir. Eğer bu bit 1 ise bu bir G/Ç buyruğudur. 15 nolu biti *I* harfiyle gösterdik. Ancak sadece bellek adreslemeli buyruklar için kullanılır ve kip biti adını alır.

İşlem kodu için sadece 3 bit kullanıldı. Dolayısıyla bu basit temel bilgisayarda sadece 8 işlem vardır. Fakat yazaç adreslemeli ve G/Ç buyrukları da kalan 12 bit buyrukların ve içindeki işlemlerin tanımlanmasında kullanıldığından, buyrukların toplam sayısı 8 den farklıdır. Gerçekte temel bilgisayar için seçilen buyrukların sayısı 25 tir.

Onaltılık Kod

Bilgisayarın buyrukları çizelge 5.2 de gösterilmiştir. 3 harfle gösterilen kısaltmalar programcı ve kullanıcılar için verilmiştir. Onaltılık kodlar eşdeğer

ikili kodları gösterir. Onaltılık kodların kullanılması ile 16 bit buyruk kodunu 4 basamağa indirmiş olduk. Her bir onaltılık basamağı 4 bit gösterir. Bellek adreslemeli buyrukların her birinde adres kısmı 12 bit uzunluğundadır. Adres kısmında 3 tane *x* konmuştur. Bunlar adrese karşılık gelen onaltılık sayılar olarak alınır. Buyruğun en soldaki sol biti *I* ile gösterilir ve *I* = 0 olunca soldaki 4 bit 0 ile 6 arasında bir onaltılık sayıya karşılık gelir. Son bit *I* = 1 olunca son 4 basamağa karşılık gelen onaltılık sayı 8 ile E arasındadır.

Yazaç adreslemeli buyruklar işlemi belirlemek için 16 bit kullanır. Soldaki 4 bit daima 0111 dir. Bu da onaltılık 7 ye karşılık gelir. Kalan 3 tane onaltılık basamak kalan 12 bitin onaltılık karşılığıdır. G/Ç buyrukları da 16 bit kullanırlar. Son 4 bit daima 111 olup onaltılık F e eşittir.

Buyruk Kümesinin Tamamlılığı

Buyrukların yapacağı işlemleri araştırmadan önce bir bilgisayarda bulunması gereken buyruk tiplerini tartışalım. Bir bilgisayarda bir buyruk kümesi mevcuttur. Bu buyruklar yardımıyla kullanıcı birleştirici dil programlar yaparak işlenebilir fonksiyonlar yazabilir. Bilgisayarın buyruk kümesinde aşağıdaki sınıflarda yeterince buyruk varsa küme TAM dır denir.

1. Aritmetik, mantık ve kaydırma buyrukları
2. Bilginin bellek ve yazaçlar arasında aktarılmasını sağlayacak buyruklar
3. Durumu belirleyen buyruklarla program denetim buyrukları
4. G/Ç buyrukları

Aritmetik, mantık ve kaydırma buyrukları kullanıcının, istediği tipte hesaplama ve işlem yapabilmesini sağlar. Sayısal bir bilgisayarda bilgilerin tamamı bellekte tutulur. Fakat bütün hesaplama ve işlemler işlemci yazaçlarında yapılır. Dolayısıyla kullanıcı bilginin bu iki birim arasında aktarılmasına gerek duyar. Sayısal bilgisayarın önemli bir özelliği karar verme kabiliyetine sahip olmasıdır. Örneğin iki sayı mukayese edilebilir ve büyük olan sayı ayrı işlem görebilir veya ikincisi büyükse ikinci ayrı işlem görebilir. Program denetim buyrukları, örneğin dallanma buyrukları programın akış yönünü değiştirebilirler. G/Ç buyrukları bilgisayar ile kullanıcı arasında haberleşme için gereklidir. Programlar ve veriler belleğe aktarılmalı ve sonuçlar kullanıcıya verilmelidir.

Çizelge 5.2 deki buyruklar yukarıda söylenen ve istenen özellikler için minimum bir küme oluşturur. Bir tek aritmetik buyruk ADD vardır. Buna

yardımcı iki buyruk *AC*(CMA) ve *AC*(INC) bulunur. Bu üç buyruk yardımıyla toplama ve çıkarma yapılabilir ve çıkarma işlemi için negatif sayılar 2 ye göre tümleyene çevrilebilir. İki adet dairesel kaydırma CIL ve CIR buyrukları aritmetik kaydırma veya mantıksal kaydırma yapabilirler. Çarpma ve bölme işlemleri, toplama, çıkarma ve kaydırma işlemleri yardımıyla yapılabilir. 3 tane mantık işlem mevcuttur. Bunlar AND, *AC* yi tümle (CMA) ve *AC* yi temizle (CLA) dır. AND ve tümle NAND işlemini yapar. Çizelge 4.6 da gösterilen diğer bütün mantık işlemler iki değişken ile yapılabilir. Bilginin bellekten *AC* ye gönderilmesi *AC* ye yükle (LDA) ile mümkündür. *AC* deki bilginin belleğe yazılması (STA) ile olur. Dallanma buyrukları BUN, BSA ve ISZ diğer 4 atlama buyrukları ile birlikte program denetim ve durum şartlarının belirlenmesi işlemlerini yaparlar. Giriş (INP) ve çıkış (OUT) buyrukları çevre birimlerle bilgisayar arasında haberleşmeyi sağlar.

Her ne kadar temel bilgisayarımız için seçtiğimiz buyruklar TAM ise de, bu bilgisayar çok etkin bir bilgisayar değildir. Çok sık kullanılan işlemler hızlı çalışmazlar. Daha hızlı bir bilgisayarda daha etkin çıkarma, çarpma, VEYA ve ÖZEL-VEYA işlemleri bulunmalıdır. Temel bilgisayarımızda olmayan bu işlemler programlar yardımıyla elde edilmelidir. Bunlar altıncı bölümde diğer örneklerle beraber verilmiştir. Sınırlı sayıda buyruklar kullanarak bilgisayarın mantıksal tasarımının ayrıntılarını göstermek mümkündür. Daha çok buyruklu bir buyruk kümesi bilgisayar tasarımını çok karışık hale getirir. 8. ci bölümde tam bir bilgisayarın buyruk listesi verilmektedir. Bunlar birçok ticari bilgisayarda kullanılmaktadır.

Çizelge 5.2 de verilen her bir fonksiyon ve bunların icrası için gerekli mikro işlemler ve bunların çalışması 5.5 ten 5.7 ye kadar anlatılacaktır. Bunları sonraya bırakmamızın nedeni önce denetim birimini anlatmak, çalışmasını açıklamak ve iç tasarımını göstermek gereğindendir.

5.4 Zamanlama ve Denetim

Saat Vuruşları

Temel bilgisayarda bütün yazaçların zamanı bir ana saat tarafından kontrol edilmektedir. Saat vuruşları sistemdeki tüm yaz-bozlara ve yazaçlara uygulanmaktadır.Saat vuruşu yazaçların içeriğini denetim sinyali tarafından izinlendirilmediği sürece değiştirmez. Diğer bir deyişle saat vuruşu sadece izin verilmiş yazaçları ve yaz-bozları değiştirir. Denetim sinyalleri, denetim birimi tarafından üretilir ve ortak veri yolundaki MUX ların girişlerine, işlemci yazaçlarına ve mikro işlemler için SC ye gönderilir.

Donanımsal Denetim, Mikro Programlanmış Denetim

İki tip denetim yapısı vardır. Donanımsal denetim ve mikro programlanmış denetim. Donanımsal denetim yapısının mantığı kapılar, yazbozlar, kod-çözücüler ve diğer sayısal devrelerle sağlanır. Oldukça hızlı çalışır. Mikro programlanmış denetimde değişiklik mikro programın güncellenmesi ile sağlanır. Bu kısımda buradaki temel bilgisayar için donanımsal denetim anlatılmıştır. Benzer bir bilgisayar için mikro programlanmış denetim bölüm 7 de anlatılacaktır.

Denetim Birimi

Denetim biriminin blok şeması şekil 5.6 da verilmiştir. İki kod çözücü vardır. Bir sıra sayıcı ve çeşitli sayılarda denetim mantık kapılarından oluşur. Bellekten okunan bir buyruk *IR* buyruk yazacında tutulur. Bu yazacın ortak veri yolu sistemi içindeki yeri şekil 5.4 te gösterilmiştir. *IR* şekil 5.6 da gösterilmiştir. Buyruk yazacı üç parçaya bölünmüştür. *I* biti, OPR kodu ve 0 dan 11'e kadar bitler. İşlem kodu bitleri 12 ile 14 arasındadır. Bu 3 bitteki OPR bir 3×8 kod çözücü tarafından çözülür. Kod çözücünün 8 çıkışı $D_0 - D_7$ ile gösterilir. Her bir indis numarası bir işleme karşılık gelir. Buyruğun 15 inci biti *I* ile gösterilir ve *I* yaz-bozuna aktarılır. 0 ile 11 arasındaki bitler mantık denetim kapılarına gönderilir. 4 bit sıra sayıcı ikili olarak 0 ile 15 arasında sayar.

Zaman Sinyalleri

Sıra sayıcının çıkışları 4×16 bir kod çözücüye gönderilir. Bu kod çözücünün çıkışları 16 saat sinyalini oluşturur. Sinyaller T_0 dan T_{15} e kadardır. Denetim kapılarının iç tasarımı bilgisayarın ayrıntılı tasarımdan sonra ele alınacaktır.

Sıra sayıcı arttırılabilir veya sıfırlanabilir. (şekil 2.11 deki sayıcıya bakın). Çoğu zaman sayıcı artırılarak 4×16 kod çözücüden zaman sinyalleri gönderilmesi sağlanabilir. Eğer sayıcı bir şekilde sıfırlanırsa buna karşılık gelen T_0 sinyali gönderilir. Örnek olarak, sıra sayıcı arttırılarak T_0, T_1, T_2, T_3, ve T_4 zaman sinyalleri göndermesi sağlanmış olur. T_4 zamanında eğer D_3 kod çözücü çıkışı aktifse $SC \rightarrow 0$ olur. Bu durum sembolik olarak

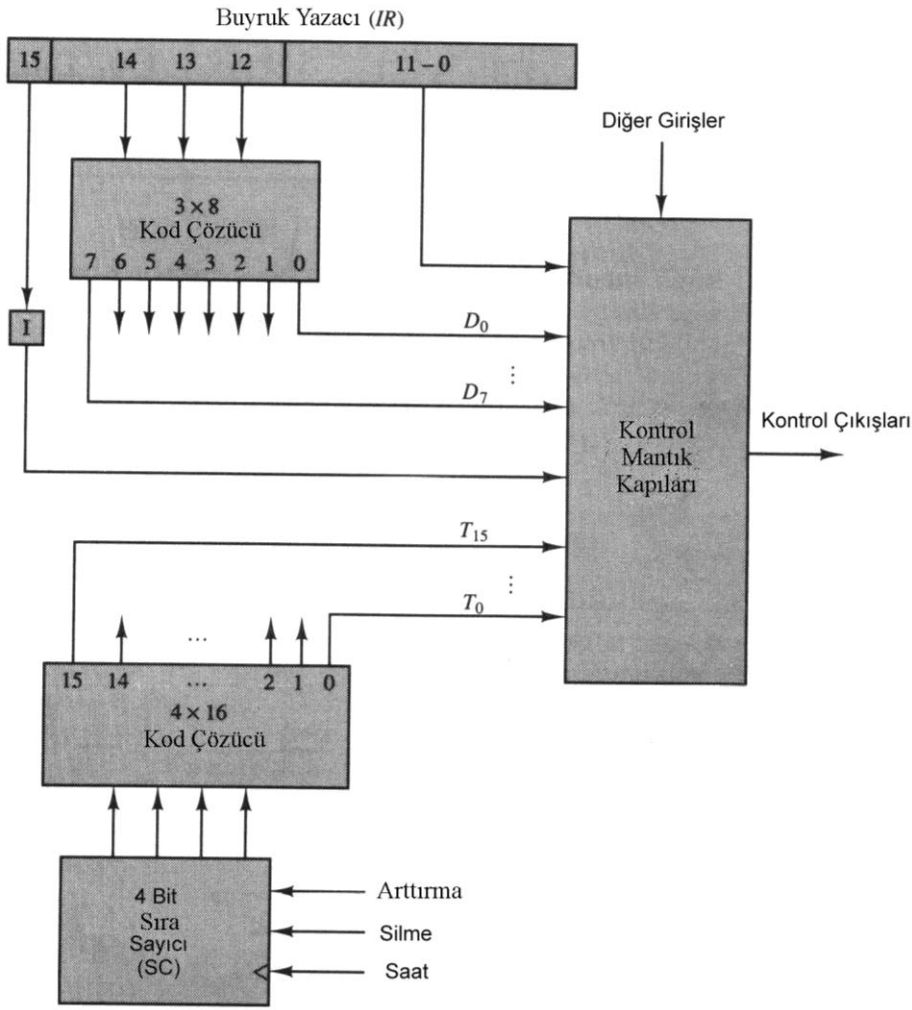
$$D_3\, T_4\text{:} \quad SC \leftarrow 0$$

ile gösterilir.

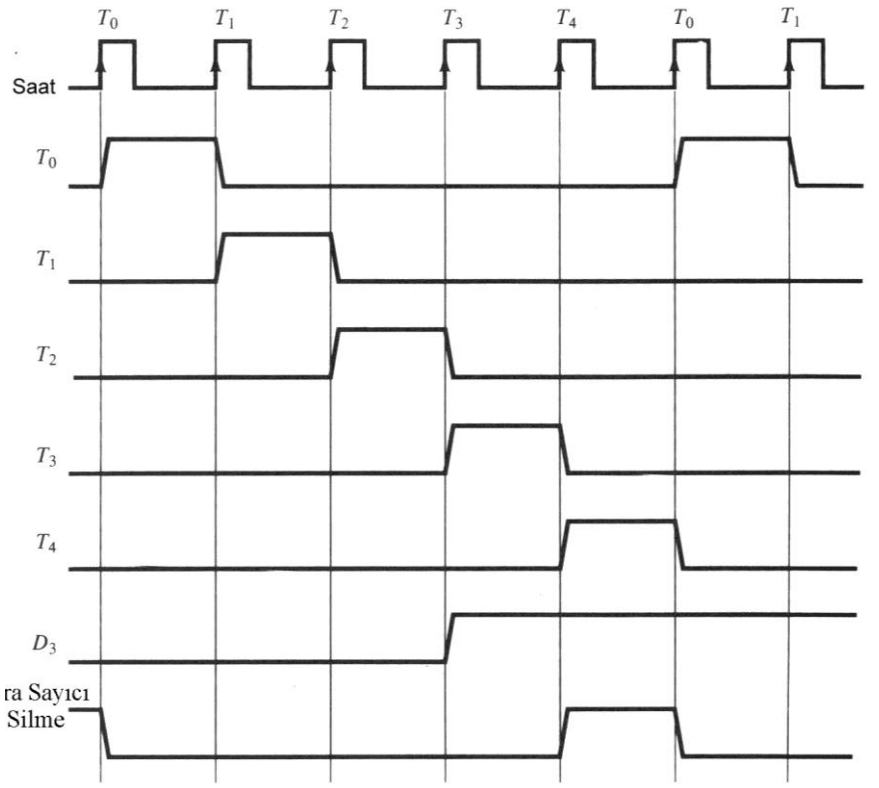
Şekil 5.7'deki zaman şeması denetim sinyallerinin zamanla bağıntısını gösteriyor.

Sıra sayıcı (*SC*) saatin yükselen koluna bağlı olarak sinyal gönderir. *SC* ın silme girişi aktiftir. Saatin ilk çıkış kolu *SC* yi 0 yapar. Bu da T_0 zaman sinyalini çalıştırır. T_0 zaman birimi süresince (yani saatin bir çıkışından başlayıp öbür çıkışının başında biten sürede) aktif kalır. T_0 ile gösterilen T_0 sinyali, sadece denetim girişlerinde T_0 sinyal teli bulunan yazaçları tetikler. *SC* her saat vuruşunun çıkış kısmında arttırılır, ancak

silme girişi aktif olursa artma durur. Böylece T_0, T_1, T_2,... zaman sinyalleri durmadan üretilir. Zaman sinyalleri ile saat vuruşu arasındaki ilişkiye dikkat edelim. Eğer *SC* sıfırlanmaz ise sinyaller T_0, T_1,…T_{15} olur ve T_0 dan devam eder.



Şekil 5.6 Temel bilgisayarın denetim ünitesi



Şekil 5.7 Denetim zaman sinyal örneği

Şekil 5.7 deki son 3 dalga biçimi *SC* nin nasıl sıfırlandığını gösteriyor. $D_3T_4 = 1$ olunca $SC \leftarrow 0$ oluyor. İşlem kod çözücünün D_3 çıkışı T_2 sinyalinin son bulmasıyla aktif hale gelir. T_4 zaman sinyali aktif hale gelince VE kapısının çıkışı D_3T_4 denetim fonksiyonunu aktif hale getirir. Sinyal *SC* nin silme girişine gider. Bir sonraki saat sürecinde (şemada T_4 ile gösterilen) sayaç sıfırlanır. Bundan sonra T_5 değil T_0 zaman sinyali ve diğer süreçler sırasıyla çalışır. Eğer bu iş olmasaydı T_5, T_6… süreçleri devam edecekti.

Bellek okuma süreci bir zaman sinyalinin yükselen koluyla başlatılır. Bellekten bilgi alma (veya yazma) sürecinin, zaman sinyali sürecinden kısa olduğu varsayılır. Bu varsayıma göre bir zaman sinyalinin çıkış koluyla başlatılan okuma veya yazma işi, sinyal kolunun inişinden önce tamamlanır. Saat süreci bundan sonra okunan bellek kelimesinin yazaca yazılması için kullanılır. Bu bağıntı birçok bilgisayarda geçerli değildir. Çünkü bellek okuma/yazma süreci zaman sürecinden daha uzundur. Böyle bir durumda işlemci için *bekle* süreçleri oluşturarak bellek kelimesinin

hazır olması sağlanır. Temel bilgisayarda bekle zamanının, zaman sürecinden küçük olduğu varsayılacaktır.

Bir bilgisayardaki çalışmayı tamamen ve iyi anlamak için zaman sinyalleri ile zaman süreçlerinin çok iyi anlaşılması gerekir. Örneğin

$$T_0: \quad AR \leftarrow PC$$

yazaç aktarım deyimine bakalım. Bu demektir ki *PC* nin içeriği *AR* ye T_0 süreci içerisinde aktarılır. T_0 bu süreçte devamlı aktiftir. Bu zaman süreci içinde *PC* ın içeriği veri yoluna yerleştirilir. ($S_2S_1S_0 = 010$). *AR* yazacının *LD* girişi aktiflenir. Gerçek aktarım ancak saatin tekrar yükselme koluna gelmesiyle tamamlanır. Bu kol ile *SC* 0000 dan 0001'e geçer. Bundan sonra T_1 aktiftir. T_0 ise aktif değildir.

5.5 Buyruk Süreci

Bellekte bulunan bir program bir dizi buyruktan oluşmaktadır. Bu programın icrası, her bir buyruk için bir sürecin sonunda mümkün olur. Her bir buyruk süreci bir takım alt parçalardan oluşur. Bunlara alt süreçler veya süreçler denir. Temel bilgisayarımızda her bir buyruk aşağıdaki süreçlerden oluşur.

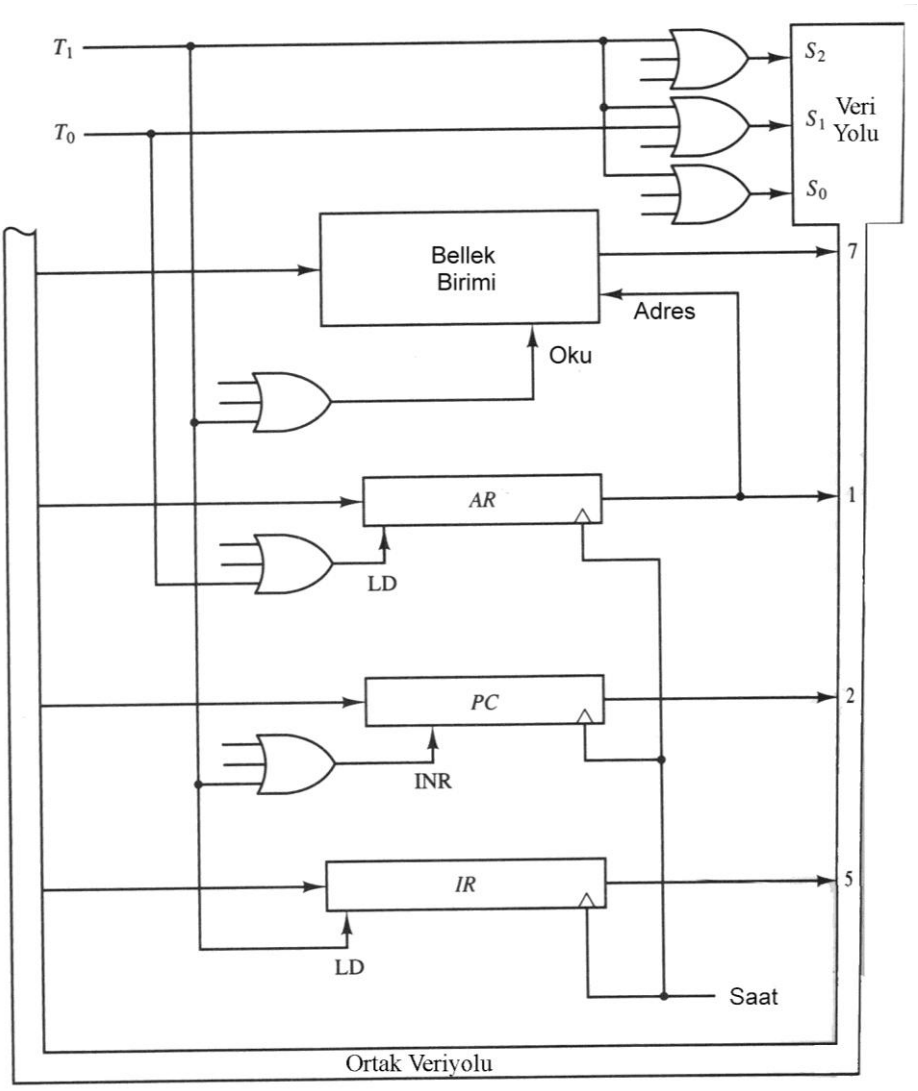
1. Bir buyruğun bellekten alınıp getirilmesi (FETCH).
2. Buyruk kodunun çözülmesi (DECODE).
3. Eğer buyruk dolaylı adreslemeli ise etkin adresin okunması.
4. Buyruğun icrası.

Dördüncü adımın tamamlanmasından sonra denetim birinci adım olan al-getir evresine döner. Bir sonraki buyruğu alır getirir, kodunu çözer, buyruğu icra eder. Bu işlem HALT buyruğuna kadar sürer.

Al-getir ve Kodunu Çöz

İlk olarak *PC* (program sayıcı) ilk buyruğun adresi ile yüklenir. Sıra sayıcı (*SC*) sıfırlanır. Böylece T_0 zaman sinyali elde edilir. Her bir saat vuruşundan sonra *SC* bir arttırılır. Böylece zaman sinyalleri T_0, T_1, T_2,… diye gider. Al-getir ve kod-çöz evrelerine ait mikro işlemler aşağıdaki yazaç aktarım buyruklarıyla belirlenir.

T_0: $AR \leftarrow PC$
T_1: $IR \leftarrow M[AR]$, $PC \leftarrow PC + 1$
T_2: $D_0, ..., D_7 \leftarrow$ kodu çöz $IR(12–14)$, $AR \leftarrow IR(0–11)$, $I \leftarrow IR(15)$



Şekil 5.8 Al-getir evresi için yazaç aktarımları

Belleğin adres girişlerine sadece *AR* bağlı olduğundan adresin önce *PC* den *AR* ye aktarılması gerekir. Bu iş T_0 zamanda olur. Bellekten okunan buyruk *IR* ye yerleştirilir. Bu iş T_1 zamanında olur. Aynı süreçte *PC* arttırılarak bir sonraki buyruk adresi hazırlanır. T_2 sürecinde *IR* içindeki işlem kodu çözülür. Dolaylı bit, *I* yaz-bozuna aktarılır. Buyruğun adres kısmı *AR* ye aktarılır. Sıra sayıcı her saat vuruşundan sonra arttırılarak T_0, T_1 ve T_2 süreçleri elde edilir.

Şekil 5.8 ilk iki yazaç aktarım deyimlerinin veri yolu sistemine nasıl bağlandığını gösteriyor. Veri yolunun *PC* den *AR* ye aktarımının sağlanması için T_0 sinyali uygulanmalıdır.

1. *PC* nin içeriğini, seçim tellerini $S_2S_1S_0 = 010$ seçerek veri yoluna yükler.
2. *AR* nin *LD* girişi izinlendirilerek veri yolunun içeriği *AR* ye yazılır.

Sonraki saat geçişinde *PC* den *AR* ye aktarım gerçekleşir. Çünkü $T_0 = 1$ dir. İkinci deyimi oluşturmak için

$$T_1: \quad IR \leftarrow M[AR],\ PC \leftarrow PC + 1$$

T_1 sinyalinin kullanılması gerekir. T_1 aşağıdaki bağlantıları sağlar.

1. Belleğin oku girişini izinlendir.
2. $S_2S_1S_0 = 111$ yaparak bellek içeriğini veri yoluna yerleştir.
3. *IR* in LD girişini izinlendirerek veri yolunun içeriğini *IR* ye aktar.
4. *PC* nin INR girişini izinlendirerek *PC* nin içeriğini 1 arttırır.

Bir sonraki saat vuruşu okuma ve arttırma işlemlerini başlatır.

Şekil 5.8 de veri yolu sisteminin bir kısmı tekrar çizilmiştir. Burada T_0 ve T_1 in yazaçların denetim girişlerine bağlantısı görülüyor. Aynı zamanda bellek ve seçim girişlerine bağlantısı da gösterilmiştir. Çoklu girişler için VEYA kapıları mevcuttur. Bunlar benzer işlemlerde kullanılır.

Buyruk Tipinin Belirlenmesi

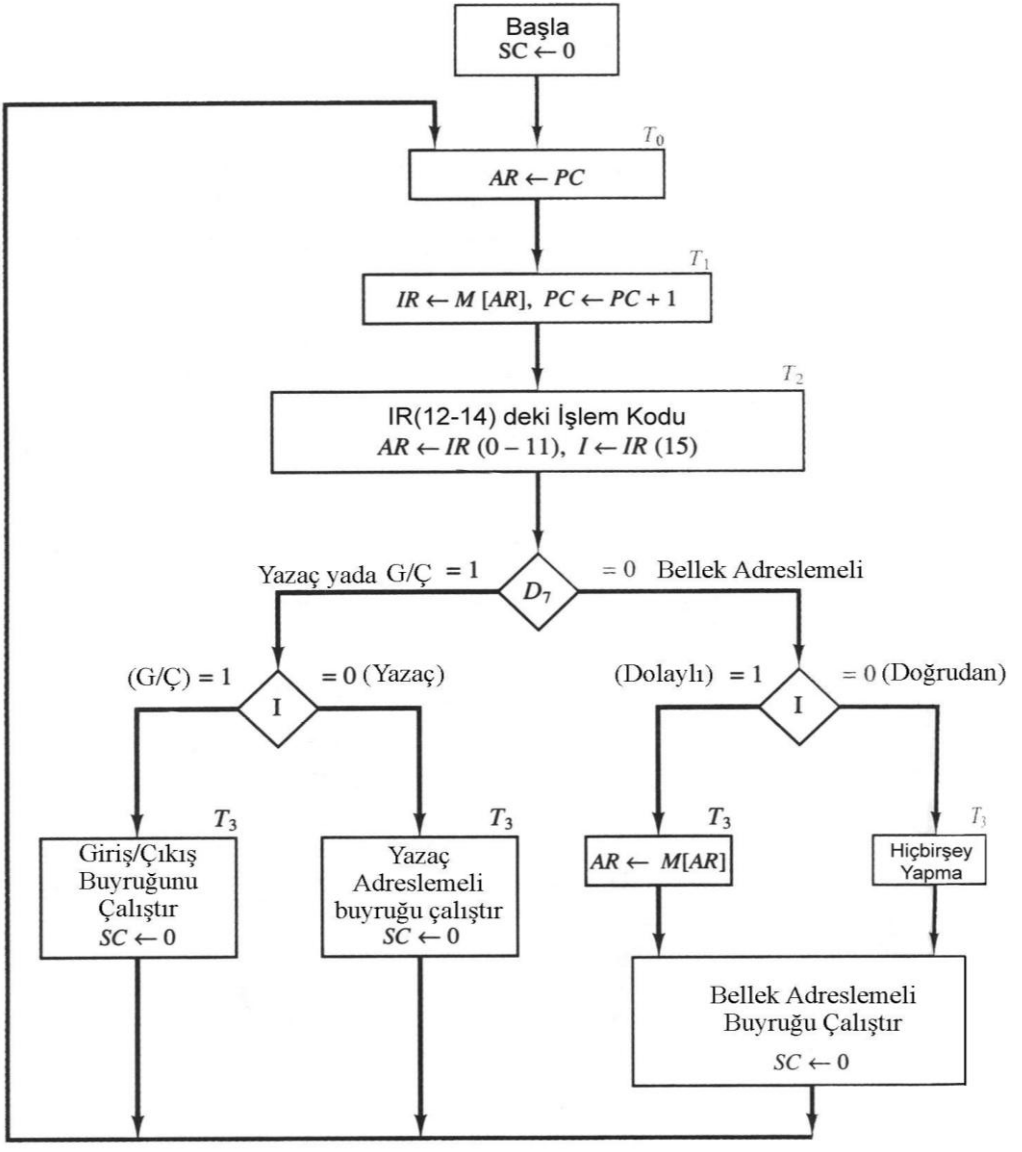
Kod-çözme olayından sonra aktif duruma geçen zaman T_3 tür. T_3 zamanında denetim birimi, bellekten henüz okunmuş olan buyruk tipini belirler. Şekil 5.9 daki akış şeması, buyruk sürecinin basit durumunu gösteriyor. Ayrıca denetimin, kodun çözülmesinden sonra buyruk tipini nasıl belirlediği de burada görülüyor. Temel bilgisayarda mevcut 3 tip buyruk şekil 5.5 te gösterilmişti.

Dolaylı Adres

Eğer OPR kodu 111 ise kod çözücünün D_7 çıkışı 1 dir. Şekil 5.5 ten eğer $D_7 = 1$ ise buyruğun yazaç adreslemeli veya G/Ç tipinde olduğu görülüyor. Eğer $D_7 = 0$ ise işlem kodu diğer 7 taneden biri (000 dan 110' a kadar) olabilir. Bunlar bellek adreslemeli buyruklardır. Denetim bundan sonra buyruğun ilk bitine bakar. Bu bit *I* da bulunmaktadır. Eğer $D_7 = 0$ ve $I = 1$ ise bu bir bellek adreslemeli buyruktur. Adresi dolaylı adreslemedir. Bundan sonra etkin adresin bellekten okunması gerekir. Dolaylı adres durumu için mikro işlem, yazaç aktarım diliyle

$$AR \leftarrow M[AR]$$

ile gösterilebilir. *AR* buyruğun adres kısmını tutmaktadır. Bu adres bellekten okuma sırasında kullanılır. *AR* tarafından verilen adresten okunan kelime bellekten okunduktan sonra ortak veri yoluna yerleştirilir. *AR* nin *LD* girişi izinlendirilerek bellek kelimesinin önemsiz 12 bitinde bulunan dolaylı adresi *AR* nin alması için hazır hale getirilir.



Şekil 5.9 Buyruk süreci için akış şeması

Üç buyruk tipi 4 alt bölüme ayrılmıştır. Seçilen işlem T_3 zaman sinyali ile çalışmaya başlar. Bu durum söyle gösterilebilir.

D'_7IT_3: $AR \leftarrow M[AR]$
$D'_7I'T_3$: Hiçbir şey
$D_7I'T_3$: Yazaç adreslemeli buyruğu icra et.
D_7IT_3: G/Ç buyruğunu icra et.

Bir bellek adreslemeli buyrukta $I = 0$ ise etkin adres halen AR de bulunduğundan bir şey yapmaya gerek yoktur. Nitekim $D'_7T_3 = 1$ ise sıra sayıcı arttırılır. Böylece bellek adreslemeli buyruğun T_4 zaman değişkeni ile icrasına devam edilir. Yazaç adreslemeli veya G/Ç buyruğu T_3 zaman sinyalinin verilmesiyle icra edilebilir. Buyruğun icrasından sonra sıra sayıcı sıfırlanır ve denetim al-getir evresine döner. $T_0 = 1$ olur.

SC (sıra sayıcı) her pozitif saat geçişinde ya arttırılır ya sıfırlanır. Eğer SC arttırılıyorsa $SC \leftarrow SC + 1$ yazılmayacak. Bunun yerine denetimin bir sonraki zaman sinyalini verdiği söylenecek. SC temizlenirse $SC \leftarrow 0$ gösterimi kullanılacaktır.

Yazaç adreslemeli buyruklarının icrası için gerekli yazaç aktarımları bu bölümde anlatılmaktadır. Bellek-adreslemeli buyruklar gelecek bölümde açıklanacaktır. Giriş/Çıkış buyrukları bölüm 5.7 dedir.

Yazaç Adreslemeli Buyruklar

Yazaç adreslemeli buyruklar $D_7 = 1$ olması ile tanınır. $I = 0$ dır. Bu buyruklar 0 dan 11'e kadar olan bitlerle 12 buyruk ifade ederler. Bu 12 bit $IR(0–11)$ de mevcuttur. Bunlar T_2 zaman sürecinde AR ye aktarılırlar.

Yazaç adreslemeli buyruğun denetim fonksiyonları ve mikro işlemleri çizelge 5.3 te verilmiştir. Bu buyruklar saatin T_3 geçişi sırasında icra edilir. Her denetim fonksiyonu $D_7I'T_3$ Boole bağıntısını içerir. Bu kolaylık için "r" ile gösterilecek. Denetim fonksiyonu $IR(0–11)$ bitlerinden birisi ile ayırt edilir. IR nin i inci biti B_i ile gösterilirse, bütün denetim fonksiyonları kısaca rB_i ile gösterilebilir. Örneğin CLA buyruğu 7800 onaltılık kodundadır. (Çizelge 5.2 de var). Bunun ikili karşılığı 0111 1000 0000 0000 dır. İlk bit 0 dır ve I ya eşittir. Sonraki 3 bit işlem kodunu oluşturur ve kod çözücüden D_7 çıkışı ile tanınır. IR nin 11 inci biti 1 dir ve B_{11} den anlaşılır. Bu buyruk için denetimin çalıştıracağı fonksiyon $D_7I'T_3B_{11} = rB_{11}$ dir. Yazaç adreslemeli buyruğun icrası T_3 zamanında tamamlanır. SC sıfırlanır. Denetim T_0 sinyali ile sonraki buyruğu alıp getirmeye gider.

İlk 7 yazaç adreslemeli buyruklar sil, tümleme, dairesel kaydırma ve arttırma mikro işlemlerini yaparlar. Bu işlemler AC veya E yazaçları üzerinde yapılır. Sonraki 4 buyruk bir sonraki buyruğun atlanması olayını gerçekleştirir. Bunun için belli bir şartın sağlanması gereklidir. Bir buy-

Çizelge 5.3 Yazaç-adreslemeli buyruklarının çalışması

$D_7I'T_3$ = *r (tüm yazaç adreslemeli buyruklarda ortak)*
$IR(i)$ = B_i *(işlemlerle belli olan IR*(0–11) *inci bitleri)*

	r:	$SC \leftarrow 0$	Sil *SC*
CLA	rB_{11}*:*	$AC \leftarrow 0$	Sil *AC*
CLE	rB_{10}*:*	$E \leftarrow 0$	Sil *E*
CMA	rB_9*:*	$AC \leftarrow \overline{AC}$	Tümleyen *AC*
CME	rB_8*:*	$E \leftarrow \overline{E}$	Tümleyen *E*
CIR	rB_7*:*	$AC \leftarrow$ shr AC, $AC(15) \leftarrow E$, $E \leftarrow AC(0)$	Sağa döndür
CIL	rB_6*:*	$AC \leftarrow$ shl AC, $AC(15) \leftarrow E$, $E \leftarrow AC(15)$	Sola döndür
INC	rB_5*:*	$AC \leftarrow AC + 1$	*AC* arttırma
SPA	rB_4*:*	Eğer ($AC(15) = 0$) ise ($PC \leftarrow PC + 1$)	Eğer pozitifse atla
SNA	rB_3*:*	E	eğer negatifse atla
SZA	rB_2*:*	*Eğer (*$AC = 0$) ise ($PC \leftarrow PC + 1$)	*AC* sıfır ise atla
SZE	rB_1*:*	Eğer ($E = 0$) ise ($PC \leftarrow PC + 1$)	*E* sıfır ise atla
HLT	rB_0*:*	S ← 0 (S bir başlangıç yaz-bozudur)	Bilgisayarı durdur

ruğun atlanması *PC* ın 1 arttırılması ile sağlanır. Bu artma, normal artmaya ek olarak yapılır. Şart denetim deyimleri koşulların bir parçası olarak tanınır. Eğer $AC(15)$ biti sıfır ise sayı pozitif; eğer $AC(15) = 1$ ise sayı negatiftir. Eğer $AC = 0$ ise tüm yaz-bozların değeri 0 dır. HLT buyruğu başlangıç yaz-bozu *S* yi 0 lar ve sıra sayıcıyı saydırmaz, durdurur. Bilgisayarın işleme devam etmesini sağlamak için başlangıç yaz-bozu elle 1 yapılmalıdır.

5.6 Bellek Adreslemeli Buyruklar

Her buyruğun icrası için gereken mikro işlemlerin belirlenmesi için, yapmaları beklenen fonksiyonun çok iyi tanımlanması gerekmektedir. Buyrukların listelendiği çizelge 5.2 ye bakınca bazı buyrukların çok anlamlı açıklandığı görülmektedir. Bunun nedeni bir buyruğun sözcüklerle açıklanmasının genelde çok uzun olması ve çizelgede böylesi uzun açıklamalar için yeterli yer bulunmamamsıdır. Şimdi bellek adreslemeli buyrukların fonksiyonunun yazaç aktarım dili ile çok kesin olarak tanımlanabileceği gösterilecektir.

Etkin Adres

Çizelge 5.4 te 7 adet bellek adreslemeli buyruk gösteriliyor. Kod-çözücünün her bir buyruk için çıkış değeri D_i ($i = 0, 1, 2, ..., 6$) çizelgede verilmiştir. Buyrukların etkin adresleri adres yazacı *AR* dedir ve T_2 zaman sinyali sürecinde $I = 0$ için veya $I = 1$ için T_3 zaman sinyalinde başlar. Her bir buyruğun sembolik tanımı çizelgede yazaç aktarım diliyle veril-

Çizelge 5.4 Bellek-adreslemeli buyruklar

Sembol	*kod-çözücü işlem*	*Sembolik tanımlama*
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR]$, $E \leftarrow C_{çıkış}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[\mathrm{AR}] \leftarrow PC$, $PC \leftarrow AR + 1$
ISZ	D_6	$M[AR] \leftarrow M[AR] + 1$
		Eğer $M[AR] + 1 = 0$ ise $PC \leftarrow PC + 1$

miştir. Buyruğun veri yolu sistemi içinde gerçek icrası bir dizi mikro işlemi gerektirir. Çünkü bellekte bulunan veriler doğrudan doğruya işlenemezler. Veriler bellekten okunmalı, işlemin yapılacağı yazaca getirilmeli ve mantık devreleriyle işlem yapılmalıdır. Şimdi her bir buyruk işlemini açıklayacak ve denetim fonksiyonlarını ve mikro işlemlerini vereceğiz. Bütün mikro işlemleri özetleyen bir akış şeması bölüm sonunda verilecektir.

AND: *AC* yi VE le

Bu buyruk *AC* de bulunan bitlerle etkin adresten alınan bellek kelimesinin bitlerini çift çift VE ler. İşlemin sonucu *AC* ye aktarılır. Bu buyruğu icra eden mikro işlemler

$$D_0T_4: \quad DR \leftarrow M[AR]$$
$$D_0T_5: \quad AC \leftarrow AC \wedge DR, \quad SC \leftarrow 0$$

Bu buyruk için denetim fonksiyonu D_0 işlem kod çözücüsünü kullanır. Çünkü ikili kod değeri 000 olan AND işlemi buyruk içinde varsa, AND işlemi aktif durumdadır. Buyruğun icrası için iki zaman sinyali gerekir. T_4 zaman sinyalini oluşturan saat vuruşuyla veri bellekten *DR* ine aktarılır. T_5 sinyalini çalıştıran saat vuruşu ile de *AC* nin içeriği ile *DR* nin içeriği VE lenir ve sonuç *AC* ye yazılır. Aynı saat vuruşu *SC* yi 0 yapar. Böylece zaman sinyali T_0 ile yeni bir buyruk sürecine başlar.

ADD: *AC* ye Ekle

Bu buyruk etkin adresi verilen bellek kelimesinin içeriğini *AC* nin değeriyle toplar. Toplam *AC* ye yazılır. *AC* nin çıkış elde si *E* ye aktarılır. Bu buyruğun icrası için gerekli mikro işlemler

$$D_1T_4: \quad DR \leftarrow M[AR]$$
$$D_1T_5: \quad AC \leftarrow AC + DR, E \leftarrow C_{\text{çıkış}}, \quad SC \leftarrow 0$$

Aynı T_4 ve T_5 zaman sinyalleri kullanılmıştır. Fakat işlem kod çözücüsü D_1 dir, D_0 değildir. Buyruk bellekten alınıp kodu çözüldükten sonra, işlem kod çözücünün sadece bir ucu aktiftir. Bu çıkış, denetimin buyruğun icrası sırasında uyacağı mikro işlemleri belirler.

LDA: *AC* ye Yükle

Bu buyruk etkin adresle verilen bellek kelimesinin içeriğini *AC* ye aktarır. Bu buyruğun icrası için gerekli mikro işlemler

$$D_2T_4: \quad DR \leftarrow M[AR]$$
$$D_2T_5: \quad AC \leftarrow DR, \quad SC \leftarrow 0$$

Şekil 5.4 e bakıldığında veri yolundan *AC* ye doğrudan yol olmadığı görülmektedir. Bu yüzden önce bellek kelimesinin *DR* ye okunması sonra *DR* nin içeriğinin *AC* ye aktarılması yapılır. Veri yolunun *AC* nin girişlerine bağlı olmamasının nedeni toplayıcı ve mantık birimlerindeki gecikmelerdir. Bellekten okuma, veri yolu üzerinde geçiş süresi ve kelimenin mantık ve toplama devrelerinden geçiş süresinin 1 zaman sürecinden uzun olduğu varsayılmıştır.

STA: *AC* yi Belleğe Aktar

Bu buyruk *AC* nin içeriğini etkin adresi verilen bellek kelimesine yazar. *AC* nin çıkışları, veri yoluna ve veri yoluda belleğin veri girişlerine bağlıdır. Dolayısıyla bu işlem bir tek mikro işlemle yapılabilir.

$$D_3T_4: \quad M[AR] \leftarrow AC, \quad SC \leftarrow 0$$

BUN: Şartsız Dallanma

Bu buyruk programı etkin adresle verilen adrese gönderir. *PC*, bir sonraki adımda bellekten okunarak buyruğun adresini tutar *PC*, T_1 zamanında arttırılarak bir sonraki buyruğun adresi hazırlanır. BUN buyruğu programcıya buyrukların sırasını bozabilme yetkisi kazandırır. Buna program dallanır veya atlar denir. Bu işi şartsız olarak yapar. Buyruk bir tek mikro işlemle icra edilebilir.

$$D_4T_4: \quad PC \leftarrow AR, \quad SC \leftarrow 0$$

Etkin adres AR den ortak veri yolu aracılığıyla *PC* a aktarılır. *SC* nin sıfır yapılması ile denetim T_0 zamanına geçer ve *PC* deki yeni adreste bulunan buyruk alınıp getirilir.

BSA: Dallan ve Geri Dönüş Adresini Sakla

Bu buyruk programın alt program adı verilen bir bölümüne dallanması için faydalıdır. BSA buyruğu icra edilince bir sonraki buyruğun adresini *PC* den alarak bellekte belirli bir adrese yazar. Alt programın başlangıç adresini *PC* a yazar ve alt programın ilk adresi olarak çalışır. Bu işlem çizelge 5.4 te aşağıdaki yazaç aktarım işlemleriyle verilmiştir.

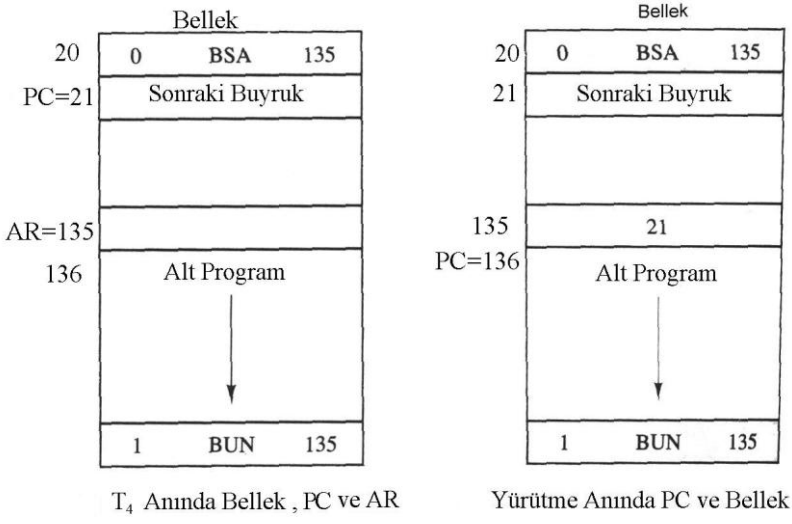
$$M[AR] \leftarrow PC, \quad PC \leftarrow AR + 1$$

Geri Dönüş Adresi

Bu buyruğun bir alt program da nasıl kullanıldığını gösteren sayısal bir örnek şekil 5.10 da verilmiştir. BSA buyruğunun bellek adresi 20 olsun. *I* biti 0 dır. Buyruğun adres kısmında 135 sayısının ikili eşdeğeri vardır. Al-getir ve kod-çöz adımlarından sonra *PC* da 21 vardır. Buda programdaki bir sonraki buyruğun adresidir. *AR* inde 135 adresi vardır. Bunlar (a) şeklinde gösteriliyor. BSA buyruğu aşağıdaki sayısal işlemi yapar.

$$M[135] \leftarrow 21, \quad PC \leftarrow 135 + 1 = 136$$

Bu işlemin sonucu şekilde (b) kısmında görülmektedir. Dönüş adresi 21 bellek adresi 135 te bulunmaktadır. Denetim programı işletmeye 136 adresinden devam eder. Alt programdan ana programa dönüş işlemi (21 adresi), dolaylı bir BUN buyruğu ile gerçekleşir.



Şekil 5.10 BSA buyruğunun icra örneği

BUN buyruğu BSA buyruğun son buyruğu olarak verilmiştir. Bu buyruk icra edilince denetim dolaylı sürece geçer, 135 adresinde bulunan etkin adresi okur. Burada daha önce saklanmış bulunan 21 adresi vardır. Bu adres *PC* a aktarılır. Bir sonraki buyruk süreci *PC* ı 21 adresiyle bulur. Dolayısıyla denetim programı dönüş adresinden itibaren icra eder.

Alt Program Çağrısı

BSA buyruğu genellikle alt program çağırma adı verilen fonksiyonu icra eder. Alt programın sonundaki dolaylı BUN buyruğu ise alt programdan dönüş adı verilen fonksiyonu icra eder. Ticari bilgisayarların çoğunda dönüş adresi ya bir işlemci yazacında veya belleğin yığıt adı verilen kısmında saklanır. Bu olay bölüm 8.7 de ayrıntılarıyla incelenecektir.

BSA buyruğunun 1 zaman sürecinde icrası mümkün değildir. Belleğin ve veri yolunun uygun biçimde kullanılmasıyla BSA buyruğu birbirini takip eden iki mikro işlemle 2 zamanda tamamlanır.

$$D_5T_4: \quad M[AR] \leftarrow PC, \quad AR \leftarrow AR + 1$$
$$D_5T_5: \quad PC \leftarrow AR, \quad SC \leftarrow 0$$

T_4 zaman sinyali belleğe yazma işlemini başlatır. *PC* nin içeriğini veri yoluna yazar ve *AR* nin INR girişini izinlendirir. Belleğe yazma ve *AR* nin arttırılması işlemi bitince sonraki saat geçişi olur. Veri yolu, *AR* nin *PC* ye aktarılması işini T_5 zamanında yapar.

ISZ: Arttır ve Eğer Sıfır İse Atla

Bu buyruk etkin adres ile verilen kelimenin değerini arttırır. Eğer arttırılan değer 0 ise. *PC* 1 arttırılır. Böylece bu buyruktan sonra gelen buyruk atlanmış olur. Çünkü zaten *PC* daha önce 1 arttırılmıştı. Programcı genellikle negatif bir sayıyı (2 nin tümleyeni olarak) bellek kelimesine yerleştirir. Negatif sayı her icrada 1 arttırılınca sonunda sıfır değerine ulaşır.

Bellekte bir kelimeni arttırılması mümkün olmadığından, bu kelimenin *DR* nin içine okunması şarttır. Bundan sonra toplama biriminde arttırılır, *DR* ye yazılır sonra tekrar belleğe yerine aktarılır. Bu iş aşağıdaki mikro işlemlerle yapılır.

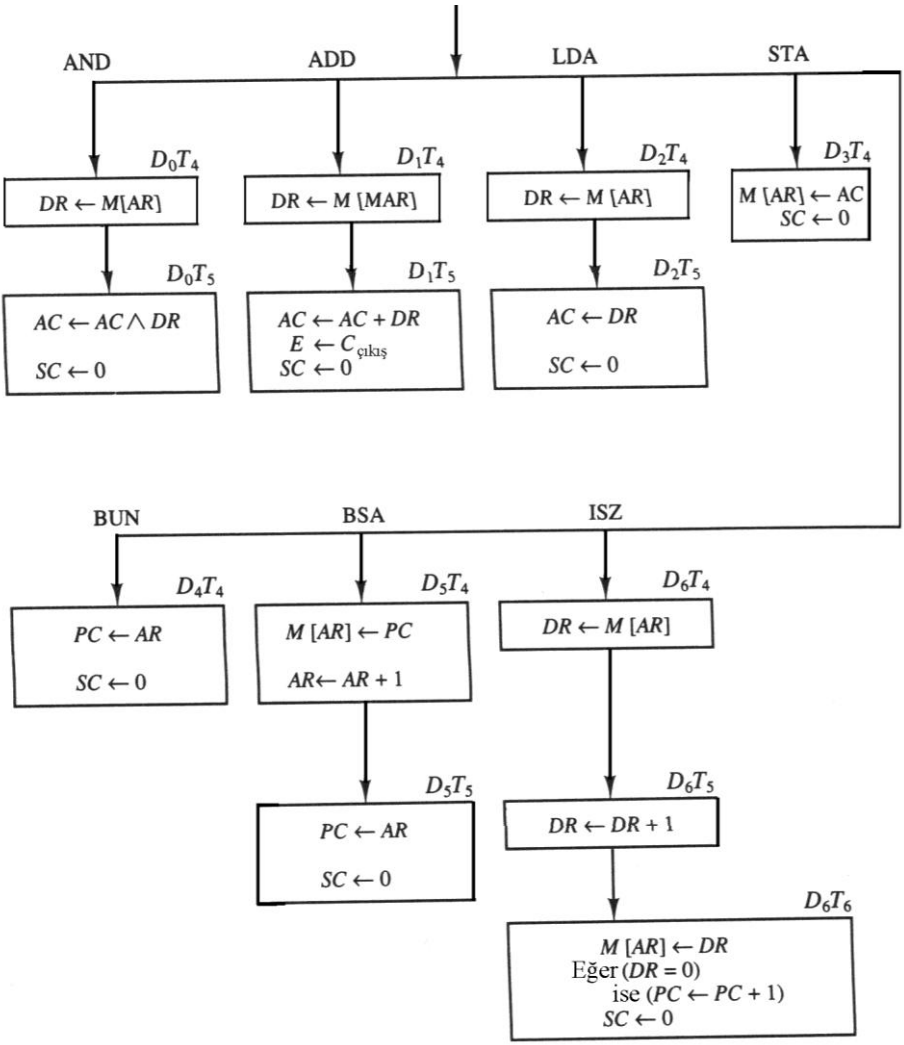
$$D_6T_4: \quad DR \leftarrow M[AR]$$
$$D_6T_5: \quad DR \leftarrow DR + 1$$
$$D_6T_6: \quad M[AR] \leftarrow DR, \text{ eğer } (DR = 0) \text{ ise } (PC \leftarrow PC + 1), \ SC \leftarrow 0$$

Denetim Akış Şeması

7 adet bellek adreslemeli buyruğun icrası için gerekli mikro işlemler şekil 5.11 de verilmektedir. Denetim fonksiyonları her kutunun üstünde belirtilmektedir. T_4, T_5 ve T_6 zamanlarında icra edilen mikro işlemler işlem

koduna (kodun değerine) bağlıdır. Bunlar akış şemasında 6 farklı yolla gösterilmiştir. Buyruk kodu çözüldükten sonra denetim bu yollardan birine geçer. Her durumda son zaman sinyalinden sonra sıra sayıcı sıfırlanır. Bir sonraki buyruk sürecinin başlaması için T_0 sinyaline geçilir.

En uzun buyruğu (ISZ) icra etmek için yedi zaman sinyaline ihtiyaç var. Bilgisayar 3 bit sıra sayıcı ile yapılabilir. Fakat sıra sayıcının 4 bit yapılması, problemlerle verilecek olan diğer buyruklar için gerekli ilave zamanları üretebilmesi içindir.



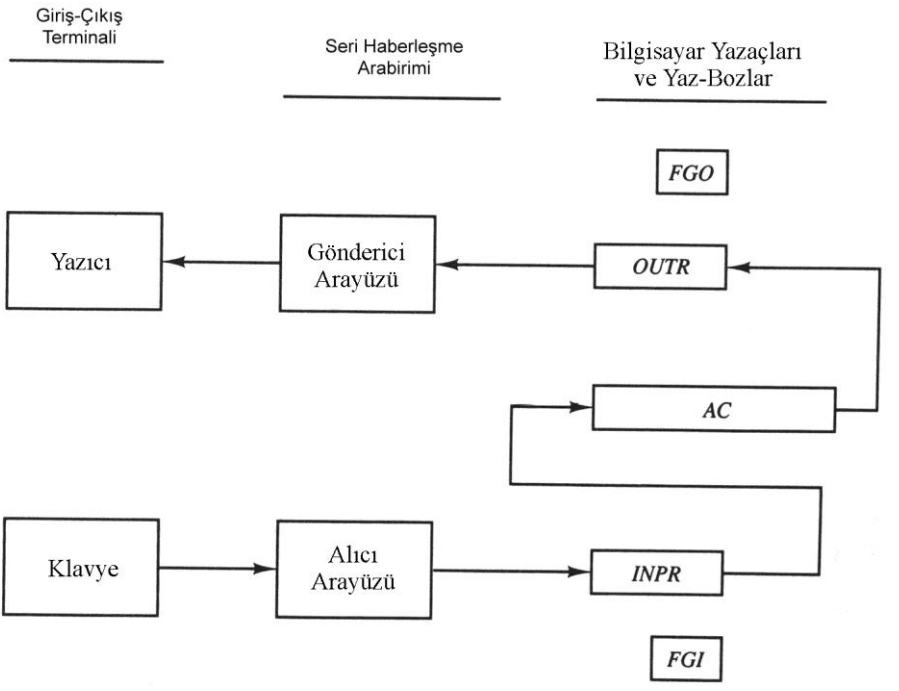
Şekil 5.11 Bellek adreslemeli buyruklar için akış şeması

5.7 Giriş-Çıkış Buyrukları ve Kesmeler

Bir bilgisayarın çalışmasında dış çevre birimleriyle haberleşmesi zorunludur. Buyruklar ve veriler bir giriş aletinden gelmelidir. İşlem sonuçları çıkış aletleri ile kullanıcıya aktarılmalıdır. Ticari bilgisayarların çok çeşitli giriş/çıkış aletleri vardır. Temel bilgisayarımızın en temel ihtiyaçlarını karşılamak üzere bir terminal, bir klavye ve bir yazıcıdan oluşan G/Ç aletleri olduğunu varsayacağız. G/Ç yapısı 11. Bölümde verilecektir.

Giriş/Çıkış Düzeni

Terminal seri bilgiler gönderir ve alır. Her bir bilgi birimi 8 bit alfasayısal koddan oluşur. Klavyeden gelen seri bilgiler INPR giriş yazacına aktarılır. Yazıcı için seri bilgiler OUTR çıkış yazacında tutulur. Bu iki yazaç seri haberleşme ara yüzeyi ile haberleşirler. Bunlar *AC* ile paralel olarak haberleşirler. G/Ç düzeni şekil 5.12 de görülmekte. Gönderici ara yüzeyi seri bilgileri klavyeden alır ve INPR yazacına aktarır. Alıcı ara yüzeyi bilgiyi OUTR yazacından alır ve yazıcıya seri olarak aktarır. Seri haberleşme ara yüzeyi bölüm 11.3 te açıklanacaktır.



Şekil 5.12 Giriş/Çıkış düzeni

Giriş Yazacı

Giriş yazacı *INPR* 8 bitten oluşur ve alfasayısal bilgileri tutar. 1 bitlik *FGI* giriş bayrağı bir denetim yaz-bozudur. Bayrak eğer giriş aletinde yeni bilgiler varsa 1 yapılır. Bilgiler bilgisayar tarafından kabul edilince bayrak 0 yapılır. Bayrak giriş aleti ile bilgisayar arasındaki zamanın uyumlu yapılması için gereklidir. Bilgi aktarım işlemi şöyledir: İlk olarak *FGI* giriş bayrağı 0 yapılır. Bir tuşa basıldığında 8 bit alfasayısal kod *INPR* yazacına aktarılır ve *FGI* sıfırlanır. Bayrak sıfırlanınca yeni bilgiler *INPR* yazacına kaydırılır.

Çıkış Yazacı

Çıkış yazacı *OUTR* benzer şekilde çalışır. Fakat bilginin yönü tersine olur. İlk önce çıkış bayrağı *FGO* 1 yapılır. Bilgisayar bayrak bitine bakar ve eğer 1 ise bilgi *AC* den *OUTR* yazacına paralel olarak aktarılır, sonra *FGO* sıfırlanır. Çıkış aleti bilgiyi kabul eder, istenen karakteri basar. İşlem tamamdır ve tekrar *FGO* = 1 yapılır. *FGO* 0 olduğu sürece bilgisayar *OUTR* yazacına yeni karakter yüklenir. Çünkü bu koşul yazıcının karakter basmakla meşgul olduğunu gösterir.

Giriş/Çıkış Buyruklar

G/Ç buyrukları bilginin *AC* nin içine ve *AC* den dışarıya aktarımı için gereklidir. Ayrıca bayrak bitleri de denetlenir. Ayrıca kesmelerde bu buyruklarla denetim yapılır. G/Ç buyruklarının işlem kodu 111 dir. ve $D_7 = 1$ ve $I = 1$ ile tanınır. Buyruğun kalan bitleri işlemin özel kodu için ayrılmıştır. G/Ç buyrukları için denetim fonksiyonları ile mikro işlemler çizelge 5.5 te verilmektedir.

Bu buyruklar T_3 zaman sinyalini oluşturan saat geçişi ile icra edilir. Her denetim fonksiyonu D_7IT_3 Boole bağıntısı ile çalışır. Bu fonksiyon kısalık için p ile gösterilecektir. Denetim fonksiyonu *IR* yazacı içindeki *IR*(6-11) bitlerinden biri ile belli olur. *IR* nin i inci biti B_i ile gösterilirse bütün denetim fonksiyonları pB_i şeklinde gösterilebilir. $I = 6$ ile 11 arasındadır. $p = D_7IT_3 = 1$ olunca *SC* sıfırlanır.

Çizelge 5.5 Giriş /çıkış buyrukları

$D_7IT_3 = p$ (tüm giriş-çıkış buyruklarında ortak kullanılır).
$IR(i)$ = (IR(6-11) bitleri bunlar buyruklarla belli)

	p:	$SC \leftarrow 0$	Sil *SC*
INP	pB_{11}:	$AC(0–) \leftarrow INPR$, $FGI \leftarrow 0$	Giriş karakteri
OUT	pB_{10}:	$OUTR \leftarrow AC(0–7)$, $FGO \leftarrow 0$	Çıkış karakteri
SKI	pB_9:	Eğer ($FGI = 1$) ise ($PC \leftarrow PC + 1$)	Giriş bayrağını atla
SKO	pB_8:	Eğer ($FGI = 0$) ise ($PC \leftarrow PC + 1$)	Çıkış bayrağını atla
ION	pB_7:	$IEN \leftarrow 1$	Kesme aktif olacak
IOF	pB_6:	$IEN \leftarrow 0$	Kesme pasif olacak

INP buyruğu bilgiyi *INPR* yazacından, *AC* nin düşük önemdeki 8 bitine aktarır. Ayrıca giriş bayrağını sıfırlar. OUT buyruğu ise *AC* nin en önemsiz 8 bitini çıkış yazacına (*OUTR*) taşır ve çıkış bayrağını sıfırlar. Çizelge 5.5 teki diğer iki buyruk durum bayraklarını kontrol eder ve eğer bayrak 1 ise bir sonraki buyruğu atlar. Atlanan buyruk normal olarak bir dallanma buyruğudur. Bu dallanma ile geri dönülür ve bayrak tekrar kontrol edilir. Eğer bayrak 0 ise dallanma buyruğu atlanmaz. Eğer bayrak 1 ise dallanma buyruğu atlanır ve giriş veya çıkış buyrukları icra edilir. (G/Ç programlarına örnekler 6.8 de verilmiştir.) Son iki buyruk *IEN* yaz-bozunu 0 yada 1 yapmaktadır. *IEN* yaz-bozunun amacı kesme işlemi ile beraber anlatılacaktır.

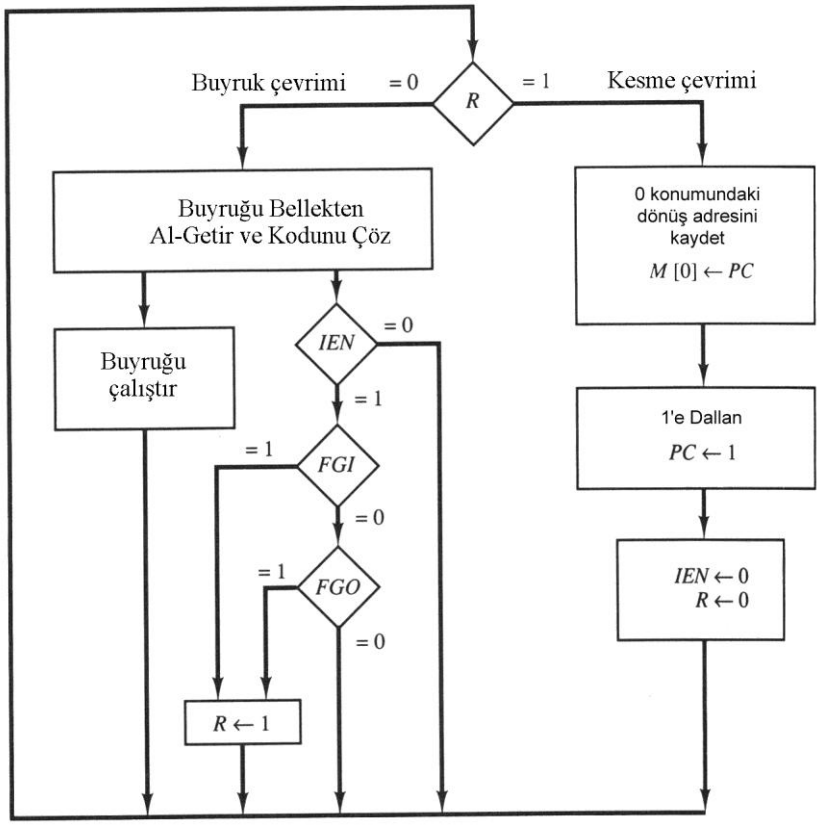
Program Kesme

Burada anlatılan haberleşme çeşidi, programla aktarım denetimi olarak adlandırılır. Bilgisayar, bayrak bitini kontrol etmeye devam eder, bitin 1 yapıldığını görünce bilgi aktarımına başlar. Bu tip bir aktarım yeterince etkin değildir. Özellikle çevre birimi ile bilgisayar arasındaki aktarma oranı çok düşüktür. Bunun nedeni incelenirse; buyruk sürecini 1 μs de bitiren bir bilgisayar düşünelim. Bir G/Ç aleti düşünelim. Saniyede maksimum 10 karakter aktarabilirsin. Bu 100.000 μs de bir karakter eder. Bilgisayar her bir aktarım arasında bayrağı 50.000 defa kontrol eder. Faydalı bir iş yapılacağına, bayrak kontrolü ile zaman kaybedilmektedir.

Programlama ile denetim için bir alternatif yol, dış aletin aktarıma hazır olduğunu bilgisayara haber vermesidir. Bu arada bilgisayar başka işlerle uğraşabilir. Bu tip aktarım kesme yöntemini kullanır. Bilgisayar bir programı çalıştırırken, bayrakları kontrol etmez. Bir bayrak 1 yapıldığında, icra etmekte olduğu programın çalışması durdurulur ve bayrağın 1 yapıldığı haber verilir. Bilgisayar yapmakta olduğu işi anında bırakır ve G/Ç aktarımı için gerekli önlemleri alır. G/Ç işini bitirince, kesme den önce çalıştırdığı programa döner ve kaldığı yerden devam eder.

Kesme izin yaz-bozu (*IEN*) iki buyruk ile 1 veya 0 yapılabilir. *IEN* silindiğinde yani 0 yapıldığında (IOF buyruğu ile), bilgisayar, kesme yapılabilir duruma geçer. Bu iki buyruk programcıya kesme yeteneğini kullanma veya kullanmama yetkisi verir.

Kesmelerin nasıl kullanıldığı 5.13 teki akış şemasında gösterilmiştir. *R* kesme yaz-bozu bilgisayara eklenmiştir. $R = 0$ iken bilgisayar buyruk sürecine gider. Buyruk sürecinin icra evresi boyunca *IEN* denetim birim tarafından gözlenir. Eğer 0 ise programcı kesmeleri kullanmak istemiyordur. Dolayısı ile denetim bir sonraki buyruk sürecine geçer. Eğer *IEN* = 1 ise denetim bayrak bitlerine bakar. Eğer her iki bayrakta 0 ise *G* (giriş) veya *Ç* (çıkış) yazaçlarının kullanım için hazır olmadıkları anlaşılır.



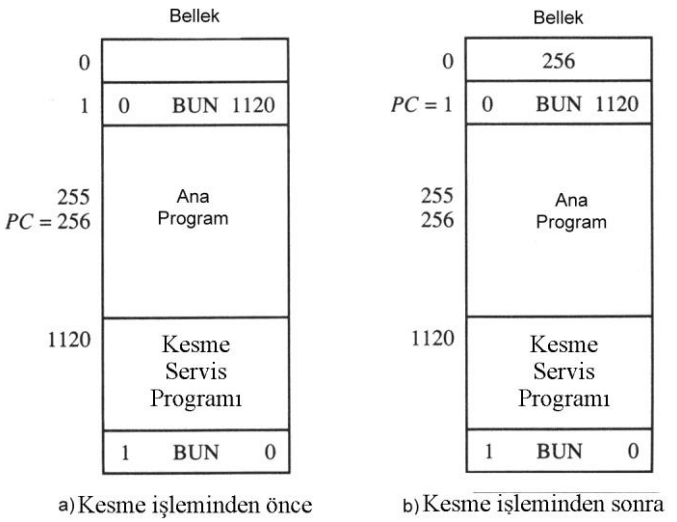
Şekil 5.13 Kesme süreci için akış şeması

Bu durumda denetim bir sonraki buyruk sürecine geçer. Eğer *IEN* = 1 iken bayraklardan biri 1 ise *R* yaz-bozu 1 yapılır. İcra evresinin sonunda denetim *R* nin değerine bakar. *R* = 1 ise buyruk sürecine geçmez, kesme sürecine geçer.

Kesme süreci

Bu süreç dallanma ve geri dönüş işleminin donanımla yapılmış bir biçimidir. *PC* içindeki dönüş adresi, daha sonra kolayca bulunabilecek bir adrese yerleştirilir. Program kesmenin geldiği andaki buyruğuna geri döner. Bu yer bir işlemci yazacı veya bellek yığıtı yada özel bir bellek adresi olabilir. Burada biz belleğin 0 adresini dönüş adresini yazmak için kullanıyoruz. Bundan sonra denetim *PC* a 1 adresini yazar, *R* yi *IEN* i 0 yapar. Böylece bir kesme isteği halledilene kadar başka kesmelerin olmasına izin verilmez.

Şekil 5.14'te kesme süreci boyunca neler olduğu gösterilmiştir. Varsayalım ki bir kesme isteği olsun ve *R* = 1 yapılsın. Bu anda *PC* içinde 256 adresi mevcuttur (bu dönüş adresidir). Programcı daha önceden kes-



Şekil 5.14 Kesme süreci gösterimi

meye karşılık bir hizmet programını 1120 bellek adresinden başlayarak yerleştirmiş olsun. 1 adresinde BUN 1120 buyruğu vardır. Şekil 5.14(a).

Denetim T_0 zaman sinyaline geldiğinde $R = 1$ durumundadır ve denetim kesme sürecine girer. *PC* deki 256 adresi belleğin 0 adresine yazılır. *PC* a 1 adresi yazılır, $R = 0$ yapılır. Bir sonraki buyruk sürecinin başında 1 bellek adresindeki buyruk, denetimi 1120 adresindeki G/Ç hizmet programına taşır. Bu program bayrakları kontrol eder. Hangi bayrağın set edildiğini bulur. Bayrağın gösterdiği G ve Ç işlemini yapar. Yani aktarmayı yapar. Bu yapıldıktan sonra ION buyruğunu icra edilerek $IEN = 1$ yapılır. Yani sonraki kesmeler izinlendirilir. Program kesmenin geldiği adrese döner. Şekil 5.14(b).

Bilgisayarı esas programda kaldığı yere geri döndüren buyruk adres kısmında 0 bulunan dolaylı bir dallanma buyruğudur. Bu buyruk kesme hizmet programının sonuna konmuştur. Bu buyruk bellekten okunur. $I = 1$ olduğu için denetim dolaylı evreye gider. Etkin adresi okur. Etkin adres 0 adresinde bulunmaktadır. Kesme süreci başında oraya yazılmıştır. Dolaylı BUN buyruk icrasıyla *PC* a 0 adresindeki dönüş adresi yazılır.

Kesme Süreci

Artık kesme süreci için gereken yazaç aktarım deyimleri yazılabilir. Kesme süreci, kesme yaz-bozu *R* nin 1 yapılmasından sonraki icra evresi bitince başlar. *R* yaz-bozu eğer $IEN = 1$ ve bayraklardan birisi aktiflendirilmişse 1 yapılır. *R* nin 1 yapılması herhangi bir zamanda yani her-

hangi bir T_i de olabilir Yalnız T_0, T_1 ve T_2 de olmaz. Bu durumda R nin 1 yapılması şartı:

$$T_0'T_1'T_2\ (IEN)\ (FGI + FGO): \quad R \leftarrow 1$$

Değiştirilmiş Al-getir Evresi; Şimdi buyruk sürecinin al-getir ve kod-çöz evreleri biraz değiştirilsin. Şekil 5.9 da görüldüğü biçimde sadece T_0, T_1 ve T_2 zaman sinyallerini kullanmak yerine 3 zaman sinyali R' ile VE lenmektedir. Böylece al-getir ve kod-çöz evreleri $R'T_0$, $R'T_1$ ve $R'T_2$ denetim fonksiyonlarıyla belli olur. Bunun nedeni buyruğun icrasından sonra $SC = 0$ yapılmasıdır. Denetim al-getir evresine yalnızca $R = 0$ ise gider. Eğer $R = 1$ ise denetim kesme evresine gider. Kesme süreci PC içindeki dönüş adresini bellek adresi 0 a yerleştirir, 1 bellek adresine dallanır. IEN, R ve SC yi temizler. Bunlar aşağıdaki mikro işlemlerle yapılabilir.

$$\begin{aligned} RT_0&: \quad AR \leftarrow 0, \quad TR \leftarrow PC \\ RT_1&: \quad M[AR] \leftarrow TR, \quad PC \leftarrow 0 \\ RT_2&: \quad PC \leftarrow PC + 1, \quad IEN \leftarrow 0, \quad R \leftarrow 0, \quad SC \leftarrow 0 \end{aligned}$$

İlk zaman sinyalinde AR sıfırlanır. PC nin içeriği TR yazacına aktarılır. İkinci zaman sinyalinde dönüş adresi 0 bellek adresine yerleştirilir. PC sıfırlanır. Üçüncü zaman sinyalinde PC 1 arttırılır. R ve IEN sıfırlanır, denetim tekrar T_0 zamanına döner. Çünkü SC sıfırlanmıştır. Bu buyruk sürecinin başında $R'T_0$ şartı vardır ve PC un içeriği 1 dir. Denetim buyruk sürecine girer. 1 adresindeki BUN buyruğunu icra eder.

5.8 Bilgisayarın Tamamlanmış Tanımı

Temel Bilgisayar İçin Akış Şeması

Buyruk sürecinin son durumu, kesme sürecini de içeren bir biçimde şekil 5.15 teki akış şemasında verilmiştir. Kesme yaz-bozu R, dolaylı ve icra evrelerinin herhangi bir anında 1 yapılabilir. SC sıfırlanınca denetim T_0 zaman sinyaline döner. Eğer $R = 1$ ise bilgisayar kesme sürecine gider. Eğer $R = 0$ ise bilgisayar buyruk sürecine gider. Eğer buyruk bellek adreslemeli buyruklardan biri ise bilgisayar önce dolaylı adresleme olup olmadığına bakar. Sonra kodu çözülen buyruğu şekil 5.11 deki akış şemasına uygun olarak icra etmeye devam eder. Eğer buyruk yazaç adreslemeli buyruklardan biri ise çizelge 5.3 te verilen mikro işlemlerden birisi ile icra edilir. Eğer buyruk bir G/Ç buyruğu ise çizelge 5.5 te verilen mikro işlemlerden biri ile icra edilir.

Akış şeması kullanmak yerine bilgisayarın işlemesini yazaç aktarım deyimleri listesi ile tanımlayabiliriz.. Bunun için bütün denetim fonksi-

yonlarını ve mikro işlemleri tek bir çizelgede toplayabiliriz. Bu çizelgenin girişleri şekil 5.11 ve 5.15 ile çizelge 5.3 ve 5.5 ten alınmıştır.

Tüm bilgisayar için denetim fonksiyonları ve mikro işlemler çizelge 5.6 da özetlenmiştir. Bu çizelgedeki yazaç aktarım deyimleri bilgisayarın iç yapısını özetlemektedir. Bunlar aynı zamanda mantık devrelerinin tasarımı için gerekli bütün bilgileri de vermektedir. Denetim fonksiyonları ve şartlı denetim deyimleri de, kapılar için gerekli Boole fonksiyonlarını bulmak için yararlıdır. Bu kapılar denetim birimini oluşturur. Mikro işlemlerin listesi, bellek ve yazaçlar için gerekli denetim girişlerinin tipini belirler. Yazaç aktarım dili sadece sayısal bir sistemin iç tasarımının tanımlanmasında değil, bunun tasarımı için gerekli devrelerin belirlenmesinde de yararlıdır.

5.9 Temel Bilgisayarın Tanımı

Temel bilgisayar aşağıdaki donanım bileşenlerinden oluşur.

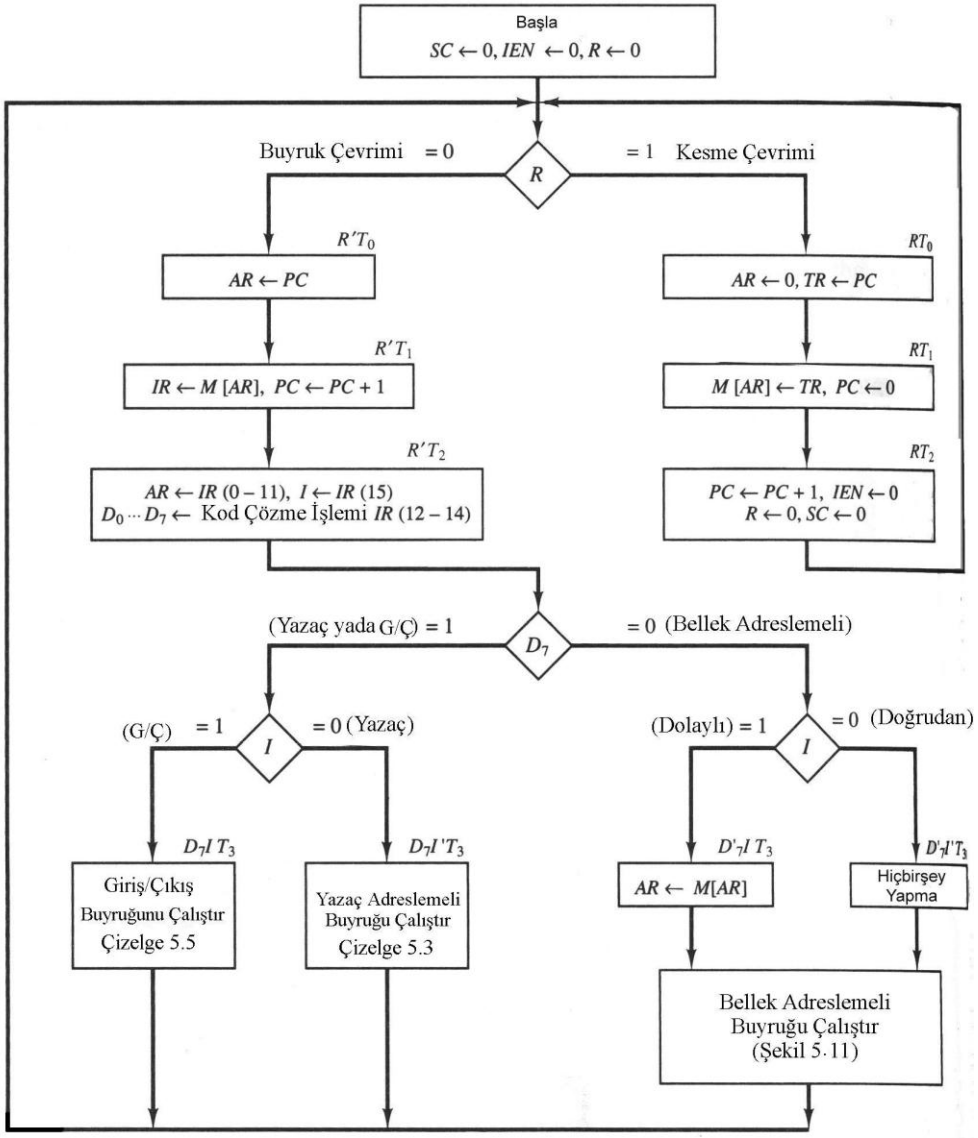
1. 16 bit 4096 kelimelik bir bellek
2. 9 yazaç: *AR*, *PC*, *DR*, *AC*, *IR*, *TR*, *OUTR*, *INPR* ve *SC*
3. 7 yaz-boz: *I*, *S*, *E*, *R*, *IEN*, *FGI* ve *FGO*
4. 2 kod çözücü: bir 3 × 8 kod çözücü bir 4 × 16 kod çözücü (zamanlama için)
5. 16 bit ortak veri yolu
6. Denetim mantık kapıları
7. *AC* nin girişine konan toplayıcı ve mantık devreler

Bellek birimi çarşıdan alınabilen standart bir birimdir. Yazaçlar şekil 2.11 de gösterildiği gibi IC 74163 benzeridir. Yaz-bozlar *D* veya *SR* tipidir. (bkz. böl 1.6). 2 kod çözücü bölüm 2.2 de anlatılan biçimdedir. Ortak veri yolu sistemi 16 tane 8 × 1 MUX ile ve şekil 4.3 teki gibi kurulabilir. Şimdi denetim mantık kapılarının nasıl tasarlanacağını gösterelim. Bundan sonraki kısım, *AC* ye bağlı toplayıcı ve mantık devreleri anlatacaktır.

Denetim Mantık Kapıları

Denetim Birimi

Denetim mantık kapılarının blok şeması şekil 5.6 da gösterilmiştir. Bu devrenin girişleri 2 kod çözücüden gelir. Ayrıca *I* yaz-bozu ve *IR* inin 0-11 bitleri de girdidir. Denetim mantık devresinin diğer girişleri; 0-15'e kadar *AC* bitleri, *AC* nin 0 olup olmadığını veya *AC* nin işaret bitini *AC*(15) denetim içindir. 0-15 arası *DR* bitleri *DR* nin 0 olup olmadığına ve 7 yaz-bozun değerlerine bakmak için kullanılır.



Şekil 5.15 Bilgisayar işlemleri için akış şeması

Çizelge 5.6 Temel bilgisayar için denetim fonksiyonu ve Mikro işlemler.

Al-getir	$R'T_0$:	$AR \leftarrow PC$
	$R'T_1$:	$IR \leftarrow M[AR]$, $PC \leftarrow PC + 1$
Kodunu çöz	$R'T_2$:	$D_0, \ldots, D_7 \leftarrow$ kodunu çöz IR (12–14)
		$AR \leftarrow IR$ 0–11), $I \leftarrow IR(15)$
Dolaylı	$D'7IT_3$:	$AR \leftarrow M[AR]$
Kesme:		
$T_0'T_1'T_2(IEN)\ (FGI + FGO)$:		$R \leftarrow 1$
	RT_0:	$AR \leftarrow 0$, $TR \leftarrow$ PC
	RT_1:	$M[AR] \leftarrow TR$, $PC \leftarrow 0$
	RT_2:	$PC \leftarrow PC + 1$, $IEN \leftarrow 0$, $R \leftarrow 0$, $SC \leftarrow 0$
Bellek adreslemeli buyruklar		
ADD	D_0T_4:	$DR \leftarrow M[AR]$
	D_0T_5:	$AC \leftarrow AC \wedge DR$, $SC \leftarrow 0$
	D_1T_4:	$DR \leftarrow M[AR]$
	D_1T_5:	$AC + DR$, $E \leftarrow C_{\text{çıkış}}$, $SC \leftarrow 0$
LDA	D_2T_4:	$DR \leftarrow M[AR]$
	D_2T_5:	$AC \leftarrow DR$, $SC \leftarrow 0$
STA	D_3T_4:	$M[AR] \leftarrow AC$, $SC \leftarrow 0$
BUN	D_4T_4:	$PC \leftarrow AR$, $SC \leftarrow 0$
BSA	D_5T_4:	$M[AR] \leftarrow PC$, $AR \leftarrow AR + 1$
	D_5T_5:	$PC \leftarrow AR$, $SC \leftarrow 0$
ISZ	D_6T_4:	$DR \leftarrow M[AR]$
	D_6T_5:	$DR \leftarrow DR + 1$
	D_6T_6:	$M[AR] \leftarrow DR$, eğer ($DR = 0$) ise ($PC \leftarrow PC + 1$), $SC \leftarrow 0$
Yazaç adreslemeli buyruklar		
	$D_7I'T_3 = r$ (tüm yazaç adreslemeli buyruklarda ortak)	
	$IR(i) = B_i$ ($i = 0, 1, 2, \ldots, 11$)	
	r:	$SC \leftarrow 0$
CLA	rB_{11}:	$AC \leftarrow 0$
CLE	rB_{10}:	$E \leftarrow 0$
CMA	rB_9:	$AC \leftarrow \overline{AC}$
CME	rB_8:	$E \leftarrow \overline{E}$
CIR	rB_7:	AC Shr AC, $AC(15) \leftarrow E$, $E \leftarrow AC(0)$
CIL	rB_6:	AC Shl AC, $AC(0) \leftarrow E$, $E \leftarrow AC(15)$
INC	rB_5:	$AC \leftarrow AC + 1$
SPA	rB_4:	Eğer ($AC(15) = 0$) ise ($PC \leftarrow PC + 1$)
SNA	rB_3:	Eğer ($AC(15) = 1$) ise ($PC \leftarrow PC + 1$)
SZA	rB_2:	Eğer ($AC = 0$) ise ($PC \leftarrow PC + 1$)
SZE	rB_1:	Eğer ($E = 0$) ise ($PC \leftarrow PC + 1$)
HLT	rB_0:	$S \leftarrow 0$
Giriş-çıkış buyrukları		
	$D_7IT_3 = p$ (tüm giriş-çıkış buyruklarında ortak)	
	$IR(i) = B_i$ ($i = 6, 7, 8, 9, 10, 11$)	
	p:	$SC \leftarrow 0$
INP	pB_{11}:	$AC(0–7) \leftarrow INPR$, $FGI \leftarrow 0$
OUT	pB_{10}:	$OUTR \leftarrow AC(0–7)$, $FGO \leftarrow 0$
SKI	pB_9:	Eğer ($FGI = 1$) ise ($PC \leftarrow PC + 1$)
SKO	pB_8:	Eğer ($FGO = 1$) ise ($PC \leftarrow PC + 1$)
ION	pB_7:	$IEN \leftarrow 1$
IOF	pB_6:	$IEN \leftarrow 0$

Denetim biriminin çıkışları şunlardır.

1. 9 tane yazacın girişlerini kontrol eden sinyaller.
2. Belleğin oku ve yaz girişlerini kontrol eden sinyaller.
3. Yaz-bozları 1, 0 yapan ve tümleyen sinyaller.
4. S_2, S_1 ve S_0 ı kontrol ederek veri yolu için bir yazaç seçen sinyaller.
5. *AC* nin toplayıcı ve mantık devrelerini kontrol eden sinyaller.

Muhtelif denetim sinyallerinin özellikleri çizelge 5.6 daki yazaç aktarım deyimlerinden doğrudan elde edilebilir.

Yazacın ve Belleğin Denetimi

Bilgisayarın ortak veri yolu sistemine bağlı yazaçları şekil 5.4 te gösterilmişti. Yazaçların denetim girişler LD(Yükle), INR(Arttır) ve CLR (sil) idi. Şimdi *AR* nin denetim girişleri için bir kapı devresi tasarlamak isteyelim. Çizelge 5.6 yı tarayarak *AR* nin içeriğini değiştiren bütün deyimleri bulabiliriz. Bunlar.

$$\begin{aligned} R'T_0&: \quad AR \leftarrow PC \\ R'T_2&: \quad AR \leftarrow IR(0\text{–}11) \\ D'_7IT_3&: \quad AR \leftarrow M[AR] \\ RT_0&: \quad AR \leftarrow 0 \\ D_5T_4&: \quad AR \leftarrow AR + 1 \end{aligned}$$

İlk üç deyim bir yazaçtan veya bellekten, *AR* ye bilgi aktarımını belirtir. Kaynak yazacın veya belleğin içeriği, veri yolu üzerine aktarılır sonra veri yolunun içeriği, LD girişi izinlendirilerek *AR* ye aktarılır. 4. Deyim *AR* yi temizler Son deyim *AR* yi 1 arttırır. Denetim fonksiyonları 3 Boole bağıntısında toplanabilir.

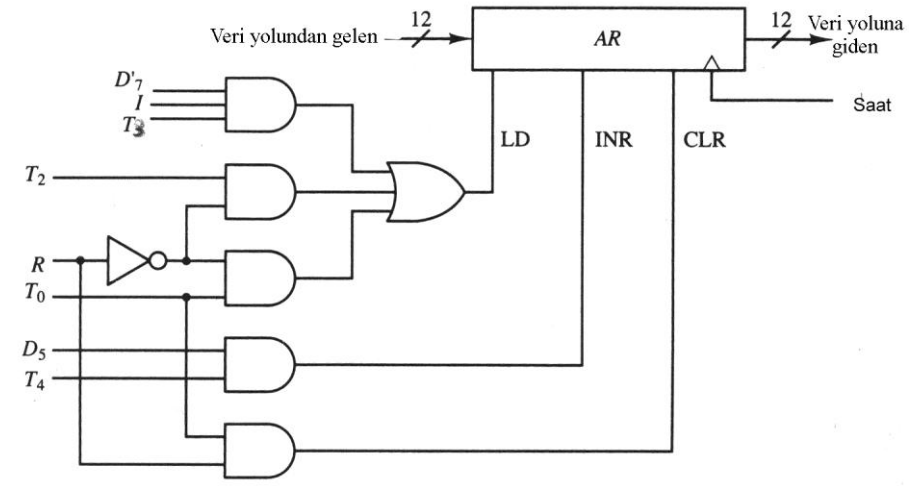
$$\begin{aligned} \text{LD}(AR) &= R'T_0 + R'T_2 + D'_7IT_3 \\ \text{CLR}(AR) &= RT_0 \\ \text{INR}(AR) &= D_5T_4 \end{aligned}$$

LD(*AR*), *AR* nın yükle girişi, CLR(*AR*) *AR* nin sil girişi, INR(*AR*) ise *AR* nin arttır girişini izinlendirir. Şekil 5.16

Benzer şekilde diğer yazaçlar için denetim kapıları oluşturulabilir. Belleğin oku ve yaz girişleri de oluşturulabilir. Belleğin oku girişi ile ilgili kapılar çizelge 5.6 nın oku işlemini gösteren deyimlerinin aranıp bulunmasıyla yapılır. Oku işlemi $\leftarrow M[AR]$ sembolüyle gösterildiğinden

$$\text{Oku} = R'T_1 + D'_7IT_3 + (D_0 + D_1 + D_2 + D_6)T_4$$

biçimindedir. Yukarıdaki fonksiyonu sağlayan çıkışlar belleğin oku girişine bağlanmalıdır.



Şekil 5.16 *AR* ile ilişkili denetim kapıları

Tek Yaz-bozların Denetimi

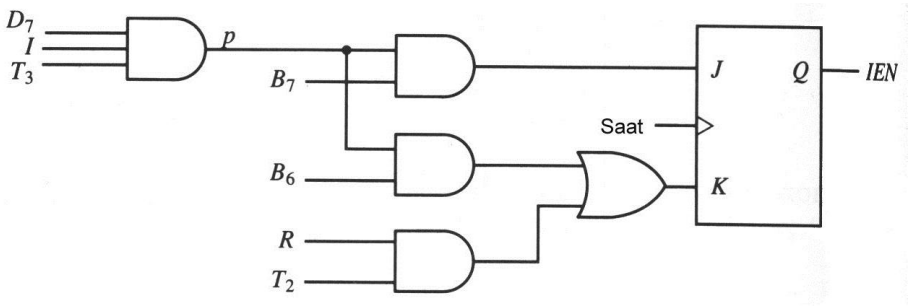
7 yaz-boz için denetim kapıları benzer biçimde bulunur. Örneğin çizelge 5.6 *IEN* in ION ve IOF buyrukları ile değiştiğini gösterir.

$$pB_7: \quad IEN \leftarrow 1$$
$$pB_6: \quad IEN \leftarrow 0$$

Burada $p = D_7IT_3$ tür. B_7 ve B_6, *IR* in 6 ve 7 ci bitleridir. Bundan başka kesme sürecinin sonunda *IEN* sıfırlanır.

$$RT_2: \quad IEN \leftarrow 0$$

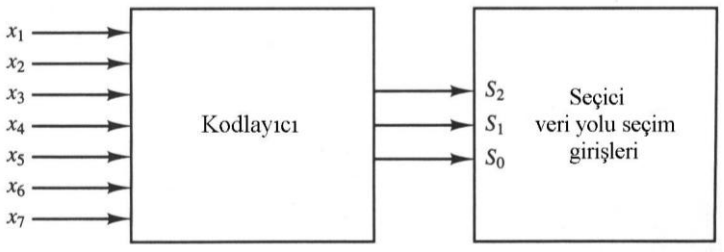
Eğer *IEN* için *JK* yaz-bozu kullanılırsa denetim kapısı şekil 5.17 deki gibidir.



Şekil 5.17 IEN için denetim girişleri

Çizelge 5.7 Veri yolu seçim devresi için kodlayıcı

Girişler							Çıkışlar			Veri yolu için yazaç seçimi
x_1	x_2	x_3	x_4	x_5	x_6	x_7	S_2	S_1	S_0	
0	0	0	0	0	0	0	0	0	0	Hiç
1	0	0	0	0	0	0	0	0	1	*AR*
0	1	0	0	0	0	0	0	1	0	*PC*
0	0	1	0	0	0	0	0	1	1	*DR*
0	0	0	1	0	0	0	1	0	0	*AC*
0	0	0	0	1	0	0	1	0	1	*IR*
0	0	0	0	0	1	0	1	1	0	*TR*
0	0	0	0	0	0	1	1	1	1	Bellek



Şekil 5.18 Veri yolu seçim girişleri için kodlayıcı

Ortak Veri Yolunun Denetimi

Şekil 5.4 te gösterilen 16 bit ortak veri yolu, S_2, S_1 ve S_0 seçim girişleri ile kontrol edilir. Girişlerde gösterilen sayılar her bir veri yolu girişi için ikili sayının onluk karşılığını gösteriyor. Her bir ikili sayı $x_1 \ldots x_7$ Boole değişkeni ile tanıtılır. Bu değişken veri yolu için yapılacak seçimde, yazaç veya belleğin numarası olup, bu mantık kapısı aktif durumda olmalıdır. Örneğin $x_1 = 1$ ise $S_2S_1S_0 = 001$ olmalıdır ve *AR* çıkışı veri yolu için seçilmiş olur. Çizelge 5.7 ikili kodlayıcı için doğruluk çizelgesidir. Kodlayıcı için girişlerin yerleştirilmesi şekil 5.18 de gösterilmektedir.

Kodlayıcı Boole fonksiyonları

$$S_0 = x_1 + x_3 + x_5 + x_7$$
$$S_1 = x_2 + x_3 + x_6 + x_7$$
$$S_2 = x_4 + x_5 + x_6 + x_7$$

Her bir kodlayıcı girişine mantık devresi çizmek için, istenen yazacı veri yoluna bağlayan denetim fonksiyonlarını bulmak gerekir. Örneğin $x_1 = 1$ olması için bütün yazaç aktarım deyimleri taranır. (çizelge 5.6) ve *AR* yi kaynak alan şu deyimler bulunur.

$$D_4T_4: \quad PC \leftarrow AR$$
$$D_5T_5: \quad PC \leftarrow AR$$

Dolayısıyla Boole fonksiyonu

$$x_1 = D_4T_4 + D_5T_5$$

olur.

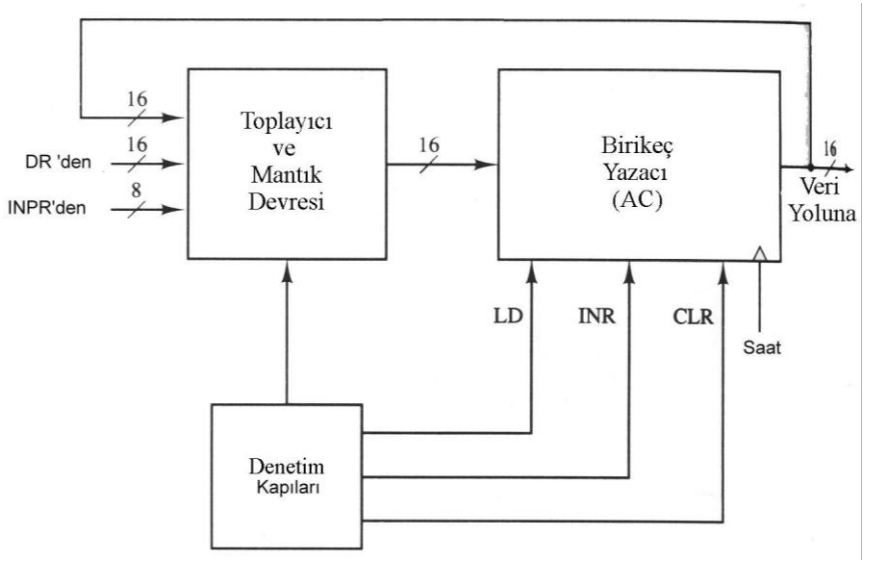
Bellekten veri çıkışı için $x_7 = 1$ ve $S_2S_1S_0 = 111$ olmalıdır. x_7 yi çalıştıran mantık aynı zamanda belleğin oku girişi içinde geçerlidir. Şu halde daha önce belleğin oku girişi için bulunanın aynısıdır.

$$x_7 = R'T_1 + D_7IT_3 + (D_0 + D_1 + D_2 + D_6)\,T_4$$

Benzer düşünce ile diğer yazaçlar içinde kapılar bulunur.

5.10 İşlemci Yazacı Mantık Tasarımı

AC yazacı ile bağlantılı devreler şekil 5.19 da gösterilmiştir. Toplayıcı ve mantık devrenin 3 giriş kümesi vardır. 16 girişli bir küme *AC* nin çıkışından gelir. Diğer bir 16 giriş veri yazacından (*DR*) gelir. 8 girişli 3 lü kü-



Şekil 5.19 *AC* ile ilişkili devreler

me giriş yazacından (*INPR*) gelir. Toplayıcı ve mantık devrenin yazaç (*AC*) için veri girişlerini oluşturur. Buna ilave olarak LD, INR ve CLR girişlerinin denetimi ve toplayıcı ve mantık devrenin çalışması için mantık devrelerine gerek vardır.

AC ile ilgili mantığı kurmak için çizelge 5.6 daki yazaç aktarım deyimlerine bakılmalıdır ve *AC* nin içeriğini değiştiren deyimler ayrılmalıdır.

D_0T_5:	$AC \leftarrow AC \wedge DR$	*DR* ile VE le
D_1T_4:	$AC \leftarrow AC + DR$	*DR* ile topla
D_2T_5:	$AC \leftarrow DR$	*DR* den *AC* ye aktar
pB_{11}:	$AC(0-7) \leftarrow INPR$	*INPR* den *AC* ye aktar
$r\mathrm{B}_9$:	$AC \leftarrow \overline{AC}$	*AC* yi tümle
$r\mathrm{B}_7$:	$AC \leftarrow shr\ AC,\quad AC(15) \leftarrow E$	Sağa kaydır
$r\mathrm{B}_6$:	$AC \leftarrow shl\ AC,\quad AC(0) \leftarrow E$	Sola kaydır
$r\mathrm{B}_{11}$:	$AC \leftarrow 0$	Sil
$r\mathrm{B}_5$:	$AC \leftarrow AC + 1$	Arttır

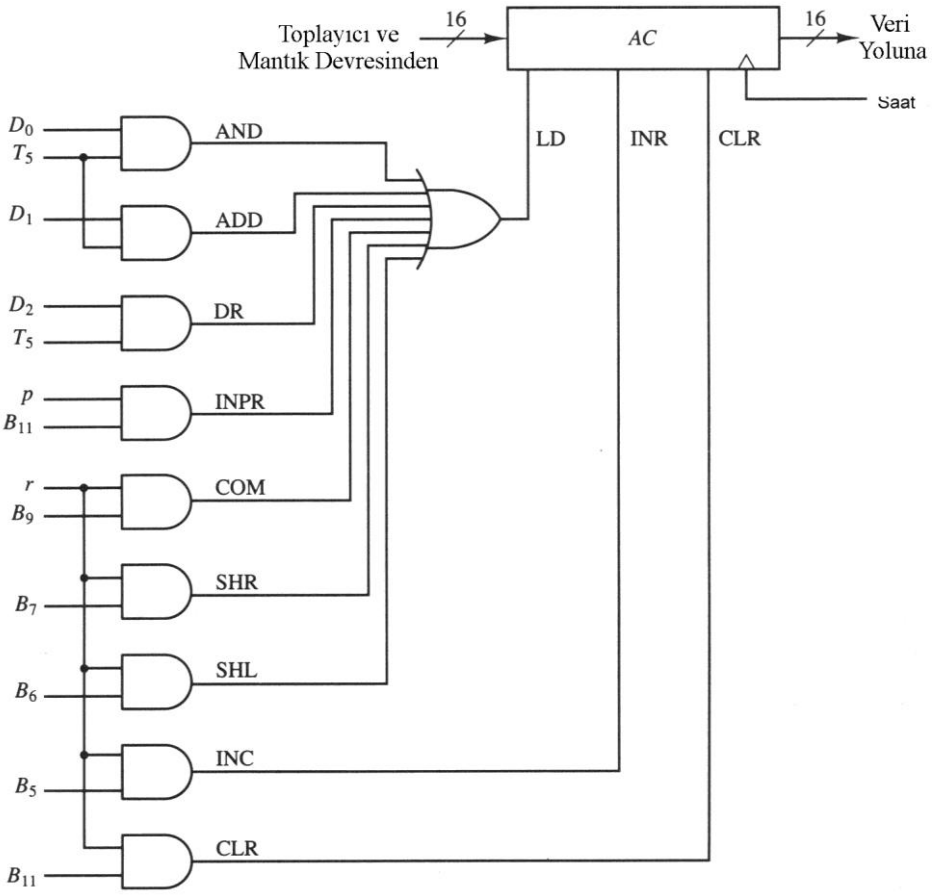
Bu listeden denetim mantık kapılarını ve toplayıcı ve mantık devreleri bulabiliriz.

AC Yazacının Denetimi

AC yazacının LD, INR ve CLR girişleriyle kontrol eden kapı tasarımı şekil 5.20 de gösterilmektedir. Kapı düzeni yukarıdaki denetim fonksiyonlarından türetilmiştir. Sil işlemi için mikro işlem rB_{11} dir. Burada $r = D_7I'T_3$, $B_{11} = IR(11)$ dir. Bu denetim fonksiyonunu üreten VE kapısı yazacın CLR girişine bağlanmıştır. Benzer şekilde arttır mikro işlemini sağlayan kapının çıkışı, yazacın INR girişine bağlanmıştır. Diğer 7 mikro işlem toplayıcı ve mantık devre içinde üretilir ve *AC* ye uygun zamanda yüklenir. Kapıların her birinin denetim fonksiyonu için çıkışları bir sembolik isimle gösterilir. Bu çıkışlar toplayıcı ve mantık devresi tasarımında kullanılır.

Toplayıcı ve Mantık Devre

Toplayıcı ve mantık devre 16 eşit alt parçaya ayrılabilir. Her bir parça *AC* nin bir bitine karşılık gelir. Yazacın iç tasarımı şekil 2.11 de görüldüğü gibidir. O şekle bakılırsa, her parçada bir *JK* yaz-bozu vardır. 2 VEYA kapısı, iki VE kapısı bulunur. LD(yükle) girişi VE kapılarının girişlerine bağlanmıştır. Şekil 5.21 de böyle bir *AC* yazacının bir parçası gösterilmiştir (VEYA kapıları gösterilmiyor). Girişler I_i ile ve çıkışlar $AC(i)$ ile

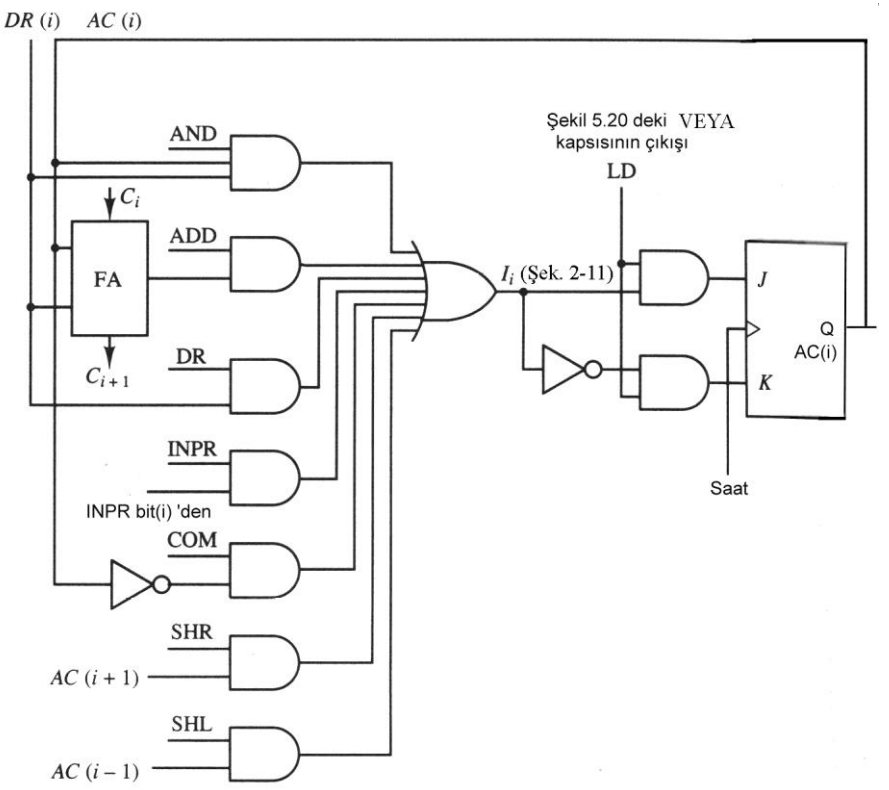


Şekil 5.20 *AC* nin LD, INR ve CLR denetimi için kapı tasarımı

gösterilmiştir. LD girişi izinlendirildiğinde 16 I_i girişi (i = 0, 1, 2, …, 15) *AC* nin 0-15 bitine aktarılır.

Toplayıcı ve mantık devrenin bir parçasında 7 VE kapısı, 1 VEYA kapısı ve 1 FA (tam toplayıcı) vardır. (Şekil 5.21). Kapıların sembolik adla verilmiş girişleri şekil 5.20 deki adlarla gösterilmiştir. Şekil 5.21 de ADD ile gösterilen giriş, şekil 5.20 deki ADD ile gösterilen çıktıdır.

AND işlemi *AC*(i) bitinin veri yazacının *DR*(i) inci bitiyle VE lenmesi ile yapılır. ADD işlemi şekil 4.6 da gösterilen FA in benzeridir.



Şekil 5.21 Toplayıcı ve mantık devresinin bir aşaması

SORULAR

5-1. Bir bilgisayar her biri 32 bitten oluşan 256K kelimelik bir bellek birimi kullanmaktadır. Belleğin bir kelimesinde bir ikili buyruk saklanmaktadır. Buyruk 4 bölümden oluşmaktadır: bir dolaylı adres biti, bir işlem kodu, 64 yazaçtan birini seçen bir yazaç kod bölümü ve adres bölümü.

a. İşlem kodu, yazaç kod bölümü ve adres bölümünde kaç bit vardır.

b. Buyruk kelime biçimini çiziniz ve her bir bölümün bit sayılarını belirtin.

c. Belleğin veri ve adres girişlerinde kaç bit vardır.

5-2. Doğrudan ve dolaylı adreslemeli buyrukların arasındaki farkı belirtiniz. Her bir buyruk çeşidi için bir işlemci yazacının içine bir veriyi aktarırken kaç tane bellek erişimi gerekmektedir.

5-3. Aşağıda belirtilen kontrol girişleri şekil 5.4 te gösterilen veri yolu sistemini kontrol etmektedir. Tüm durumlar için bir sonraki saat vuruşunda yazaca bilgi aktarımını yazın.

	S_2	S_1	S_0	Yazaca yükleme	Bellek	Toplayıcı
a.	1	1	1	*IR*	oku	—
b.	1	1	0	*PC*	—	—
c.	1	0	0	*DR*	yaz	—
d.	0	0	0	*AC*	—	topla

5-4. Aşağıdaki yazaç aktarımları şekil 5.2 deki gibi bir sistemde çalıştırılsın. Her bir aktarım için (1) veri yolu seçim girişlerini sağlayan S_2, S_1, S_0 ikili değerlerini (2) eğer varsa LD denetim girişleri aktif olan yazaçları (3) Eğer gerekli ise bellek okuma yada yazma işlemi ve eğer varsa toplayıcı ve mantık devresindeki işlemleri belirleyiniz.

a. $AR \leftarrow PC$

b. $IR \leftarrow M[AR]$

c. $M[AR] \leftarrow TR$

d. $AC \leftarrow DR$, $DR \leftarrow AC$ (aynı zamanda yapılması)

5-5. Şekil 5.4 gösterilen sistemde aşağıdaki mikro işlemlerin bir saat vuruşunda neden gerçekleşemediğini açıklayınız. İşlemi gerçekleştirecek mikro işlem sırasını belirtiniz.

a. $IR \leftarrow M[PC]$

b. $AC \leftarrow AC + TR$

c. $DR \leftarrow DR + AC$ (*AC* değer değiştirmeyecek)

5-6. Şekil 5.5 te gösterilen buyruk biçimi ve çizelge 5.2 de verilen buyruk listesine göre aşağıdaki 16 bitlik buyrukların her biri için onaltılık kod karşılıklarını ve bulduğunuz kelimelerin yerine getireceği buyruğun ne olduğunu açıklayınız.

a. 0001 0000 0010 0100

b. 1011 0001 0010 0100

c. 0111 0000 0010 0000

5-7. Temel bilgisayarda *E* yaz-bozunu 1 yapmak için gerekli olan iki buyruk nedir?

5-8. T_3 zamanında C_7 aktif ise *SC* nin 0 olduğunu kabul ederek şekil 5.7 ye benzer bir zamanlama şeması çiziniz.

$$C_7T_3: \; SC \leftarrow 0$$

5-9. Temel bilgisayardaki *AC* yazacının başlangıç değeri A937 ve *E* yaz-bozunun değeri 1 dir. CLA buyruğunun çalışmasından sonra *AC*, *E*, *PC*, *AR* ve *IR* yazaçlarının içeriklerini onaltılık tabanda gösteriniz. *PC*

nin başlangıç değerini onaltılık tabanda 021 alarak yazaç adreslemeli buyrukların her biri için bunu 11 kez tekrarlayın.

5-10. 021 adresindeki buyruk için $I = 0$ buyruk AND ve adres kısmı 083 olarak verilmektedir. 083 adresindeki veri B8F2 ve *AC* nin değeri A937 dir. Buyruğu çalıştırıp çalışma sonucunda *PC*, *AR*, *DR*, *AC* ve *IR* nin içeriklerini belirtin. Problemi diğer bellek adreslemeli buyruklar ile 6 kez daha tekrarlayın.

5-11. Bir dolaylı adreslemeli buyruk ISZ bellekten getirilip icra edilmesi sırasında *PC*, *AR*, *DR*, *IR* ve *SC* yazaçlarının içeriklerini gösteriniz. *PC* nin başlangıç değeri 7FF dir. 7FF adresindeki veri EA9F dir. A9F adresindeki veri 0C35 dir. C35 adresindeki değer ise FFFF dir. Yapılan işlemleri bir çizelge ile gösteriniz.

5-12. Temel bilgisayarın *PC* yazacının değeri 3AF dir. *AC* nin değeri 7EC3 tür. 3AF adresindeki değer 932E dir. 32E adresindeki veri ise 09AC dir. 9AC bellek adresinin içeriği ise 8B9F tir.

a. Bellekten alıp getirilecek ve çalıştırılacak buyruk hangisidir.

b. Buyruk çalıştığında *AC* içinde gerçekleşecek ikili işlemi gösteriniz.

c. Son komut icra edildiğinde *PC*, *AR*, *DR*, *AC* ve *IR* yazaçlarının içeriklerini ve *E*, *I*, *SC* nin değerlerini belirtiniz.

5-13. Çizelge 5.4 te listelenen temel bilgisayardaki 6 bellek adreslemeli buyruğun aşağıdaki çizelgede belirtilmiş buyruklara çevrildiğini kabul edelim. *EA*, T_4 zamanında *AR* de bulunan etkin adrestir. Varsayalım ki şekil 5.4 teki toplayıcı ve mantık devre ÖZEL-VEYA işlemini gerçekleştirebilmektedir ($AC \leftarrow AC \oplus DR$). Ayrıca toplayıcı ve mantık devresi çıkarma işlemini doğrudan yapamamaktadır. Çıkarma *AC* nin tümleyenini alıp 1 arttırarak bulunmuş olan 2 ye göre tümleyen kullanılarak yapılabilir. T_4 zamanından başlayarak listelenmiş her buyruğun yürütülmesi için ihtiyaç duyulan yazaç aktarımlarının sırasını belirtin. *TR* yi *AC* nin içeriğini geçici olarak saklamak için kullanabilirsiniz yada *DR* ve *AC* nin yerini değiştirebilirsiniz.

Sembol	İşlem kodu	Sembolik gösterim	Açıklama
XOR	000	$AC \leftarrow AC \oplus M[EA]$	*AC* ile ÖZEL-VEYA işlemi
ADM	001	$M[EA] \leftarrow M[EA] + AC$	*AC* yi bellekteki ile topla
SUB	010	$AC \leftarrow AC - M[EA]$	*AC* den bellektekini çıkar
XCH	011	$AC \leftarrow M[EA]$, $M[EA] \leftarrow AC$	*AC* ve belleğin yer değiştirmesi
SEQ	100	Eğer ($M[EA] = AC$) ise $PC \leftarrow PC + 1$	Bellek ve *AC* değeri eşit ise bir sonraki buyruğu atla
BPA	101	Eğer ($AC > 0$) ise ($PC \leftarrow EA$)	*AC* pozitif ise dallan

5-14. Temel bilgisayara aşağıdaki değişiklikleri yaptırınız.

1. $S_2S_1S_0 = 000$ durumunda seçilmek üzere veri yolu sistemine *CTR*(sayma yazacı) ekleyin.

2. ISZ buyruğunu *CTR* yazacına bir sayı yükleyen bir buyruk ile değiştirin.

LDC adres $CTR \leftarrow M[adres]$

3. Bir ICSZ yazaç adreslemeli buyruk ekleyiniz: *CTR* yi arttırsın eğer 0 ise bir sonraki buyruğa geçsin. Bu değişikliğin avantajını tartışınız.

5-15. Şekil 5.3 te gösterilen temel bilgisayar bellek birimi 16 bit bellek adres uzunluğu olan 65536 x 16 lık bir bellekle değiştirilmiştir. Şekil 5.5(a) da gösterilen bellek adreslemeli buyruklar $I = 1$ için aynı kalmakta fakat $I = 0$ olduğu zaman çalışacak buyruğun adresi, buyruğu takip eden kelimede 16 bit olarak verilmektedir. Bu değişikliğe göre T_2, T_3 gerektiğinde T_4 ü değiştiriniz.

5-16. Bir bilgisayar her biri 8 bitten oluşan 65536 kelime bulunduruyor. Bilgisayarın yazaçları *PC*, *AR*, *TR* 16 bit, *AC*, *DR*, *IR* 8 bit. Bellek adreslemeli buyruk 3 kelimeden oluşuyor. İlk kelime işlem kısmını diğer iki kelime adresi gösteriyor.

a. Bellek ve yazaçları şekil 5.3 teki gibi blok şemasını çizerek gösteriniz.

b. 3 kelimelik buyruk ve buna karşılık gelen 8 bitlik işlenenin bellekteki yerleşimini gösteren şemayı çiziniz.

c. T_0 zaman sinyalinde başlayan, bir bellek adreslemeli buyruğun bellekten alınması ile ilgili mikro işlemlerin sıralı listesini yazınız.

5-17. Sayısal bir bilgisayar her biri 40 bit olan 16384 kelimeden oluşun bir beleğe sahip. Bir buyruk kodu 6 bit işlem kodu ve dolaylı kip biti olmayan 14 bit adres alanından oluşmaktadır. İki buyruk bir bellek kelimesine paketlenmektedir ve kontrol biriminde 40 bitlik *IR* yazacı vardır. Bu bilgisayar için al-getir ve çalışma evreleri ile ilgili buyrukları yazınız.

5-18. Bir çıktı programı belleğin 2300 adresinden itibaren bulunmaktadır. Bu program $FGO = 1$ olduğunda MİB nin kesme isteğini almasından sonra çalışmaktadır.

a. 1 adresine hangi buyruk yerleştirilmelidir?

b. Bu çıktı programının son iki buyruğu ne olmalıdır?

5-19. *R* yazacı için yazaç aktarımları ve hafızanın durumu aşağıdaki gibidir. (X ler rasgele seçilen denetim fonksiyonlarıdır.) Belleğin veri giriş, veri çıkış, adres giriş ve okuma, yazma için denetim girişleri vardır (şekil 2.12 deki gibi). Belleğin ve *R* yazacının işleyişini blok şema şeklinde gösteriniz. *R* nin yükleme kontrol girişini seçen X_1 den X_3 e kadar olan denetim fonksiyonlarını, şemaya eklediğiniz MUX un seçim girişlerini ve belleğin okuma yazma girişlerini gösteriniz.

X'_3X_1: $R \leftarrow M[AR]$ Okunan bellek kelimesinin *R* ye aktarılması
X'_1X_2: $R \leftarrow AC$ *R* nin *AC* ye aktarımı
X'_1X_3: $R \leftarrow M[AR]$ *R* nin belleğe yazılması

5-20. Temel bilgisayarda kullanılmayan F yaz-bozu ile gerçekleştirilen işlemler aşağıda belirtilen yazaç aktarım durumlarına özeldir. Aynı durumda F in değeri değişmemelidir. Denetim fonksiyonlarını düzenleyen kapıların bağlantılarını ve F yaz-bozunun girişlerini gösteren blok şemayı çiziniz.

xT_3: $F \leftarrow 1$ F e 1 aktar
xT_1: $F \leftarrow 0$ F e 0 aktar
zT_2: $F \leftarrow \mathrm{F}$ F in tümleyenini al
wT_5: $F \leftarrow \mathrm{G}$ G yi F e aktar

5-21. Temel bilgisayarda PC için denetim kapılarını elde ediniz.

5-22. Temel bilgisayarda belleğin yazma girişi için kontrol kapılarını elde ediniz.

5-23. Temel bilgisayarda bulunan kesme yaz-bozu R nin mantık devresini çiziniz. Bir JK tipi yaz-boz ve minimum sayıda kapı kullanınız.

5-24. Çizelge 5.7 ye bakarak x_2 nin Boole mantık ifadesini elde ediniz. Bir VEYA ve birde VE kapısı ile oluşabileceğini gösteriniz.

5-25. Sıra sayıcı SC yi 0 yapan kapı yapısının Boole fonksiyonunu yazınız. Kapıların blok şemasını çiziniz ve SC nin INR ve CLR girişlerinin çıkışa nasıl bağlandığını gösteriniz (şekil 5.6 ya bakarak). Kapı sayısını minimum tutunuz.

KAYNAKLAR

1. Bell, C. G. J. C. Mudge, and J. E. McNamara, *Computer Engineering*. Bedford, MA: Digital Press, 1980.
2. Booth, T. L., *Introduction to Computer Engineering*, 3rd ed. New York: John Wiley , 1984.
3. Gibson,G. A., *Computer Systems Concepts and Desing*. Englewood Cliffs, NJ: Prentice Hall, 1991.
4. Gray, N. A. B., *Introduction to Computer Systems*. Englewood Cliffs, NJ: Prentice Hall, 1987.
5. Hill, F. J., and G. R. Peterson, *Digital Systems: Hardwire Organization and Desing*, 3rd ed. New York: John Wiley , 1987.
6. Lewin, M. H., *Logic Desing and Computer Organization*. Reading, MA: Addison-Wesley, 1983.
7. Mano, M. M., *Computer Engineering: Hardwire Desing*, Englewood Cliffs, NJ: Prentice Hall, 1988.
8. Patterson, D. A., and J. L. Hennessy, *Computer Architecture: A Quantitative Approach*. San Mateo, CA: Morgan Kaufmann Publishers, 1990.

9. Prosser, F. P., and D. E. Winkel, *The Art of Digital Desing*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1987.
10. Shiva, S. G., *Computer Desing and Architecture*. 2nd ed. New York: HapperCollins Publishers, 1991.

BÖLÜM 6

Temel Bilgisayarın Programlanması

BU BÖLÜMDE

6-1 Giriş
6-2 Makine Dili
6-3 Birleştirici Dil
6-4 Derleyici
6-5 Program Döngüleri
6-6 Aritmetik ve Mantıksal İşlemlerin Programlanması
6-7 Alt Programlar
6-8 Giriş/Çıkışın Programlanması

6.1 Giriş

Bir bilgisayar sistemi donanım ve yazılımdan oluşur. Donanım fiziksel parçalar ve ilgili donatımdan oluşur. Yazılım bilgisayar için yazılan programlara verilen addır. Bilgisayar donanımlarının işleyiş ayrıntılarıyla ilgilenmeksizin bilgisayar yazılımının birçok yönleri kavranabilir. Aynı zamanda yazılım kapasitesini bilmeksizin donanımın parçalarını tasarlamak mümkündür. Fakat bilgisayar mimarisi ile ilgilenenler hem donanım, hem de yazılım bilgisine sahip olmalıdırlar. Çünkü her iki dalda birbirini etkilemektedir.

Bir program yazmak, doğrudan veya dolaylı olarak, bir dizi makine buyruğunu belirlemek demektir. Bilgisayarın içindeki makine buyrukları ikili bir yapı oluştururlar. Bunu anlamak çok zordur. Alfabenin bilinen karakterleri ile program yazmak tercih edilir. Bunun sonucu olarak kullanıcının anlayabildiği sembolik programları makinenin anlayabileceği ikili programlara dönüştürmek gerekir.

Bir kullanıcının yazdığı program, bu programı çalıştıran bilgisayara bağlı veya bilgisayardan bağımsız olabilir. Örneğin standart fortran ile yazılan bir program bilgisayardan bağımsızdır. Çünkü her makine için bu program, programın çalıştırılacağı bilgisayarın buyruklarına çeviren programlar mevcuttur. Fakat bu çevirici programlar makineye bağımlıdır.

Bu bölüm bazı ilkel programlama kavramlarını ve bunların donanımın oluşturduğu buyruklarla bağlantılarını ele alacaktır. Birinci kısımda temel işlem ve kullanıcının sembolik programını eşdeğer ikili programa çeviren bir programın yapısını gösterecektir. Buradaki anlatım çeviricinin temel kavramlarını içermektedir. Çeviricinin ayrıntılarına girilmeyecektir. Bundan sonra muhtelif makine buyruklarının faydası, temel programlama örnekleri ile verilecektir. Son kısımda bilgisayar sistemlerinde bulunan yazılım bileşenleri kısaca ele alınacaktır.

Buyruk Kümesi

Bölüm 5 te anlatılan temel bilgisayarın buyruk kümesi, program yazımında kullanılan teknikleri anlatmakta kullanılacaktır. Bu sayede programla, programı çalıştıran donanım işlemleri arasındaki bağıntı kurulabilecektir.

Çizelge 6.1 Bilgisayar buyrukları

Sembol	Onaltılık kod	Tanımlama
AND	0 veya 8	*M* yi *AC* ile VE le
ADD	1 veya 9	*M* yi *AC* ye ekle, elde yi *E* ye aktar
LDA	2 veya A	Bellekten *AC* ye yükle
STA	3 veya B	*AC* yi belleğe sakla
BUN	4 veya C	*m* e şartsız dallan
BSA	5 veya D	Geri dönüş adresini bellekte *m* e sakla ve *m* + 1 e dallan
ISZ	6 veya E	*M* yi 1 arttır eğer 0 ise atla
CLA	7800	*AC* yi sil
CLE	7400	*E* yi sil
CMA	7200	*AC* yi tümle
CME	7100	*E* yi tümle
CIR	7080	*AC* ve *E* yi dairesel sağa kaydır
CIL	7040	*AC* ve *E* yi dairesel sola kaydır
INC	7020	*AC* yi 1 arttır
SPA	7010	*AC* pozitif ise bir sonraki buyruğu atla
SNA	7008	*AC* negatif ise bir sonraki buyruğu atla
SZA	7004	*AC* sıfır ise bir sonraki buyruğu atla
SZE	7002	*E* sıfır ise bir sonraki buyruğu atla
HLT	7001	Bilgisayarı durdur
INP	F800	Bilgiyi al ve bayrağı temizle
OUT	F400	Bilgiyi yolla ve bayrağı temizle
SKI	F200	Eğer giriş bayrağı varsa sonraki buyruğu atla
SKO	F100	Eğer çıkış bayrağı varsa sonraki buyruğu atla
ION	F080	Kesme yi çalıştır
IOF	F040	Kesme yi kapat

Temel bilgisayarı oluşturan 25 buyruk çizelge 6.1 de verilmiştir. Her bir buyruk 3 harfli bir sembolle gösterilmektedir. İlk 7 buyruk bellek adreslemeli buyruklardır. Sonraki 18 i yazaç adreslemeli veya G/Ç buyruklarıdır. Bir bellek adreslemeli buyruk 3 bölümden oluşmakta: bir kip biti, 3 bit işlem kodu ve 12 bit adres. Bellek adreslemeli buyrukların birinci rakamı kip biti ve işlem kısmını oluşturmaktadır. Diğer 3 rakam adresi belirlemektedir. Bir dolaylı adres buyruğunda kip biti 1 dir ve birinci onaltılık aralığı 8-E değerleri arasındadır. Doğrudan kip aralığı 0-6 arasındadır. Diğer 18 buyruk bir 16 bit işlem koda sahiptir. Her bir buyruk için kod listesi 4 rakamlı onaltılık sayıdır. Yazaç adreslemeli buyrukların birinci rakamı 7 dir. Bir G/Ç buyruğunun birinci rakamı her zaman F tir. *m* sembolü efektif adreste belirtilen sütunu tanımlar. *M* harfi efektif adreste bulunan bellek kelimesini gösterir.

6.2 Makine Dili

Bir program, istenen bir veri işleme işlemlerini icra etmesi için bilgisayarı komuta eden buyruklar listesidir. Bir kullanıcının program yazabileceği pek çok dil vardır. Ancak bir bilgisayar, içindeki program ikili kodda ise bunu icra edebilir. Diğer dillerde yazılan programlar icradan önce ikili şekle çevrilmelidir. Bilgisayar için yazılan programlar aşağıdaki sınıflardan birinde olabilirler.

İkili kod
Sekizlik veya Onaltılık kod
Sembolik kod

1. *İkili kod*: Buyrukların ve verilerin ikili listesidir.
2. *Sekizlik veya Onaltılık kod*: İkili kodların sekizlik veya Onaltılık biçiminde verilmişidir.
3. *Sembolik kod*: Kullanıcı semboller (harfler, rakamlar veya özel karakterler) kullanarak işlem kısmını yazar. Diğer kısımlarda bunlarla yazılabilir. Her bir sembolik buyruk bir ikili buyruğa dönüştürülür. Böyle bir programa birleştirici dil adı verilir. Yazılan programa da birleştirici dilinde program denir.
4. Yüksek seviyeli programlama dilleri: Bu diller daha çok programın çözümüne yönelik diller içerirler. Bilgisayar donanımından etkilenmezler. Buna bir örnek fortran'dır. Probleme yönelik sembol ve biçimleri içerir. Program insanların problemi çözmek için düşünecekleri sırada yazılacak deyimlerden oluşur. Yazılan bu program ikili koda dönüştürülür. Bu dönüştürmeyi yapan programa derleyici denir.

Makine Dili

Kısaca birleştirici dil bir program 1 inci sınıftır. İkili ve sekizlik veya onaltılık programlar arasındaki bire bir ilişkiden dolayı ikinci sınıftakilerde birleştirici dil gibi sayılır. Benzer biçimde birebir aktarma yapılabil-

mesinden dolayı birleştirici dile makine seviyesinde dil denir. Bilgisayar biliminde programlama dilleri, yüksek seviyeli programlama dillerinin incelenmesi anlamına gelir. Bu çalışmalar belirgin bir bilgisayara bağlı değildir, bu yüzden donanımdan bağımsızdır. Buradan yapılan çalışmaların donanım ile yazılım arasındaki bağıntı olduğundan ve makine dilinde programlarla ilgilenildiğinden anlamı makine seviyesi dil olacaktır.

Bu kitapta tanımlanan temel bilgisayar çeşitli bilgisayar dilleri arasındaki bağıntıyı anlatmak için kullanılacaktır. Çizelge 6.2 deki ikili program ele alındığında, birinci sütun buyruk veya verinin bellek adresini göstermektedir. Adres ikili olarak verilmiştir. İkinci sütun bellekteki adresin ikili içeriğini verir. (buyruk içindeki adresle buyruğun adresini karıştırmamak gerekir). Program belleğin gösterilen adresine yerleştirilir ve bilgisayar tarafından 0 adresinden başlayarak icra edilir ve istenen sonuç bulunur. Bir kişi bu programa baktığında, program icra edilince ne elde edileceğini bulmakta zorlanır. Buna karşılık donanım yalnız bu kodlardan anlar.

Onaltılık Kod

Her bir buyruk için 16 bit yazmak oldukça uzundur. Eğer sekizlik olarak yazılırsa bu azalabilir. Bunun için buyruk başına 8 basamak gerekir. Diğer yandan Onaltılık kodlarla bu iş 4 basamağa indirilebilir. (Çizelge 6.3) Onaltılık gösterim kullanıma çok uygundur. Bilgisayarın icra etmesi için bu kodlar ikili koda çevrilmelidir.

Çizelge 6.2 İki sayının toplamı ile ilgili ikili program

Bellekte yerleşimi	Buyruk Kodu			
0	0010	0000	0000	0100
1	0001	0000	0000	0101
10	0011	0000	0000	0110
11	0111	0000	0000	0001
100	0000	0000	0101	0011
101	1111	1111	1110	1001
110	0000	0000	0000	0000

Çizelge 6.3 2 sayının toplamı Onaltılık program

Bellekte yerleşimi	Buyruk Kodu
000	2004
001	1005
002	3006
003	7001
004	0053
005	FFE9
006	0000

Çizelge 6.4 Sembolik işlem kodlarıyla program

Adres	Buyruk		Açıklama
000	LDA	004	Birinci veriyi *AC* ye al
001	ADD	005	AC deki ile ikinci veriyi topla
002	STA	006	006 daki bellek adresine toplamı depola
003	HLT		İşlemi durdur
004	0053		Birinci veri
005	FFE9		İkinci veri
006	0000		Toplamı buraya depola

Çizelge 6.4 te, çizelge 6.1 deki programın ikili veya onaltılık kodları yerine buyruklar, sembolik kodlarıyla verilmiştir. Buyrukların adres kısımları ve veriler onaltılık olarak bırakılmıştır. 005 adresindeki veri negatiftir. Çünkü en sol bit 1 dir. Açıklama sütununda her buyruk için bir açıklama verilmiştir. Sembolik programların kullanımları kolay olduğundan tercih edilirler. Bunlardaki semboller ikili kodlara dönüştürülürler.

Bir adım daha ileri giderek onaltılık adresleri sembolik adreslerle ve onaltılık verileri, ondalık verilerle değiştirebiliriz. Bu daha uygundur. Çünkü programı yazan kişi verilerin bellek adreslerini bilmeyebilir. Eğer veriler belleğe buyruklardan hemen sonra yerleştirilir ve programın uzunluğu önceden bilinmezse, verilerin sayısal adresleri programın sonuna gelinene kadar bilinmez. Ayrıca ondalık sayılar onaltılık sayılara göre daha alışkın olduğumuz sayılardır.

6.5 teki program iki sayının toplamı ile ilgili bir birleştirici dil programıdır. Sonunda bir sayı bulunan ORG, bir makine buyruğu değildir. Amacı, altındaki buyruğun bellek adresini gösteren başlangıç işaretidir. Bundan sonraki 3 satır sembolik adreslerdir. Değerleri birinci sütunda yer alan başlıklarla verilmiştir. Verinin ondalık olduğu DEC sembolüyle belirtilmiştir. Sayılar pozitif veya negatif olabilirler. Ancak negatif sayılar 2 nin tümleyenine çevrilmelidir. Son satır END programın sonudur. ORG, DEC ve END sembollerine sözde buyruk denir. Bütün açıklamalar / ile başlar.

Çizelge 6.5 İki sayı toplamının birleştirici dilde programı

```
        ORG 0       / program 0 adresinden başlıyor
        LDA A       / A dan veriyi yükle
        ADD B       / B deki veri ile topla
        STA C       / C ye toplamı sakla
        HLT         / programı durdur.
A,      DEC 83      / ondalık değer
B,      DEC –23     / ondalık değer
C,      DEC 0       / toplamın yazılacağı ondalık değer
        END         / sembolik programın sonu
```

Çizelge 6.6 Fortran programı iki sayının toplamı

```
INTEGER A, B, C
DATA A,83  B,-23
C = A + B
END
```

Buna eşdeğer fortran programı çizelge 6.6 da verilmiştir.

A ve *B* değerleri INPUT veya DATA buyruğu ile verilir. Değişkenler arasındaki toplama işlemi tek bir basit işlemle verilmiştir. Bu fortran programının ikili programa dönüşmesi toplanan sayılır için 3 ayrı bellek adresi bulunması ve ayrılması ile olur. Üçüncü sayı toplam yani sonuçtur. Bundan sonra derleyici vasıtası ile işin yapılması için gerekli ikili buyruklar dizisi oluşturulur. Böylece derleyici programı, fortran programının sembollerini çizelge 6.2 deki ikili değerlere çevirir.

6.3 Birleştirici Dil

Bir programlama dili bir dizi kuralla tanımlanır. Kullanıcı dilin bütün biçim kurallarına uyar ise programı doğru olarak ikili koda dönüştürülür. Hemen hemen tüm ticari bilgisayarların kendi özel birleştirici dilleri vardır.

Birleştirici dilinde birim komut çizgisidir. Bu özel dil bir dizi kurallarla belirlenmiştir. Bu kurallar kullanılacak sembolleri ve bu sembollerin komut çizgisinde nasıl bir araya getirileceğini belirler. Şimdi temel bilgisayarımız için sembolik programlar yazmak üzere birleştirici dil kurallarını belirleyeceğiz.

Dilin Kuralları

Birleştirici dil programların her satırı *alanlar* adı verilen 3 sütundan oluşur. Sütunlar aşağıdakilere uymalıdır.

1. Başlık alanı boş olabilir veya bir sembolik adres belirler
2. Buyruk alanı bir makine buyruğu veya açıklama vardır
3. Açıklama alanı boştur veya açıklama vardır

Sembolik adres

Sembolik bir adres bir, iki veya üç alfasayısal karakterden oluşabilir. Asla üç karakterden fazla olamaz. İlk karakter harf olmalıdır, sonraki iki tane harf veya rakam olabilir. Sembol programcı tarafından keyfi seçilebilir. Başlık içindeki sembolik adres bir virgülle sona ermelidir. Böylece derleyici bunun başlık olduğunu anlar.

Buyruk alanı birleştirici dil programdır. Aşağıdaki koşullara uyar.

1. Bellek adreslemeli buyruk (MRI)
2. Yazaç adreslemeli buyruk (MRI olmayan)
3. Sözde buyruk. verili veya verisiz

Bellek adreslemeli buyruk iki veya üç sembolden oluşur. Boşluklarla birbirinden ayrılmıştır. İlk sembol üç harfli olup 6.1 çizelgesinden bir sembol ile MRI işlem kodunu tanımlar. İkincisi sembolik adrestir. Üçüncü sembol *I* dır. (kullanılması zorunlu değildir.) Eğer *I* varsa dolaylı yoksa doğrudan adreslemedir.

MRI olmayan bir deyim adres kısmı olmayan bir buyruk olarak tanımlanır. MRI olmayan bir programın buyruk alanında çizelge 6.1 deki sembollerden biri bulunur. Bunlar yazaç adreslemeli ve G/Ç buyruklarıdır.

Aşağıdakiler, programın buyruk alanında bulunan sembolleri anlatan örneklerdir.

```
CLA          MRI-değil
ADD OPR      MRI doğrudan adresli
ADD PTR I    MRI dolaylı adresli
```

İlk üç harfli semboller çizelge 6.1 deki buyruklardan biri olmalıdır. ADD gibi MRI buyrukları bir adres sembolü tarafından takip edilmelidir. *I* harfi var olabilir veya olmayabilir.

Buyruk alanındaki adres sembolü bir verinin bellek adresini belirtir. Bu adres programın herhangi bir yerinde ilk sütundaki başlık ile verilmelidir. Sembolik bir birleştirici programının ikili hale dönüştürülebilmesi için, buyruk alanındaki her bir sembolik adresin başlık alanında mutlak surette bir daha görünmesi gereklidir.

Sözde Buyruk

Bir sözde buyruk bir makine buyruğu değildir. Daha çok bir derleyiciye bilgi veren bir buyruktur ve dönüşümün bir aşaması için bilgi verir. Birleştirici tarafından tanınan 4 sözde buyruk çizelge 6.7 de verilmiştir.

Diğer birleştirici dilleri bunların dışında çok sayıda sözde buyruğu tanırlar. ORG sembolü onu takip eden satırdaki buyruk veya verinin ORG un yanındaki adreste bulunduğunu gösterir. Bir program içinde ORG u bir defadan fazla kullanmak ve bir çok bellek kesimini göstermek mümkündür. END sembolü programın sonuna yazılarak derleyiciye programın

Çizelge 6.7 Sözde buyrukların tanımı

ORG N	Onaltılık N sayısı buyruk veya veri listelerinin bellekteki başlangıç yeri
END	Sembolik programın bittiğini belirtir
DEC N	İkiliye çevrilecek işaretli ondalık N sayısı
HEX N	İkiliye çevrilecek onaltılık N sayısı

Çizelge 6.8 2 sayının çıkarılması için birleştirici dili program

	ORG 100	/ Program 100 adresinden başlamakta
	LDA SUB	/ Çıkarılanı *AC* ye yükle
	CMA	/ *AC* nin tümleyenini al
	INC	/ *AC* yi 1 arttırır
	ADD MIN	/ Çıkanı *AC* ye topla
	STA DIF	/ Farkı depola
	HLT	/ Programı durdur
MIN,	DEL 83	/ Çıkan
SUB,	DEC –23	/ Çıkarılan
DIF,	HEX 0	/ Fark buraya saklanacak
	END	/ Sembolik programın bitişi

bittiği haber verilir. Diğer iki sözde buyruk derleyiciye sayıların ikiliye çevrilebileceğini bildirir.

Programın üçüncü alanı açıklamalar için ayrılmıştır. Bir satırda açıklama olabilir veya olmayabilir. Fakat eğer varsa derleyicinin anlaması için mutlaka / ile başlamalıdır. Açıklamalar dönüşümde dikkate alınmaz.

Bir Örnek

Çizelge 6.8 deki birleştirici programı bir örnektir. Birinci satırda sözde buyruk ORG programın başlangıç noktasını $(100)_{16}$ bellek adresi olarak göstermekte. Bundan sonraki 6 satır makine buyruklarını, son 4 satır ise 4 sözde buyruğu gösteriyor. 3 sembolik adres kullanılmıştır. Her biri 1. sütunda başlık olarak vardır. 2. Sütunda ise mevcut buyrukların üç tanesi MRI olarak verileri ile bulunmaktadır. Son buyruk ise programın sonunu belirtmektedir.

Program ikili koda dönüştürüldüğünde ve bilgisayar tarafından icra edildiğinde iki sayı arasındaki farkı hesaplar. Çıkarma birinci sayı ile ikinci sayının 2 ye göre tümleyeninin toplamıdır. Çıkarılan sayı (ikinci sayı) negatiftir. Bu sayı 2 ye göre tümleyene çevrilir. Çünkü bütün negatif sayıların 2 ye göre tümleyen biçiminde ele alınmasına karar verilmiştir. -23, +23 olur ve 83 + (–23) = 106 olur.

İkili'ye Çevrilme

Derleyici

Sembolik programın ikili koda dönüştürülmesi derleyici adı verilen bir programla yapılır. Eğer dönüşüm işi önce kağıt üzerinde yapılırsa daha kolay anlaşılır. Çizelge 6.8 deki sembolik programın eşdeğer koda dönüştürülmesi, programın taranması ve sembollerin eşdeğeri olan ikili makine kodlarıyla değiştirilmesi yoluyla yapılır. İlk satırdan başlayarak önce ORG sözde buyruğuna gelinir. Bu buyruk bellekte onaltılık 100 adresinden başlanacağını söyler. İkinci satırda iki sembol var. Bu satır 100 adre-

Çizelge 6.9 Çizelge 6.8 deki programın dönüştürülmüş listesi

Onaltılık kod			
Adres	İçerik		Sembolik Program
			ORG 100
100	2107		LDA SUB
101	7200		CMA AC
102	7020		INC AC
103	1106		ADD MIN
104	3108		STA DIF
105	7001		HLT
106	0053	MIN,	DEC 83
107	FFE9	SUB,	DEC –23
108	0000	DIF,	HEX 0
			END

sine yerleştirilecek MRI buyruğunundur. *I* harfi olmadığından buyruk kodunun ilk biti 0 olmalıdır. İşlemin sembolik adı LDA dır. Çizelge 6.1 e bakıldığında buyruğun ilk onaltılık basamağının 2 olması gerekir. Adresin ikili değeri adres sembolü SUB dan elde edilmelidir. Başlık sütununa bakıldığında bu sembol dokuzuncu satırda görülür. Bu satırda belleğin 107 adresi vardır. Bu buyruğun iki parçası birleştirilince 2107 onaltılık kodu elde edilir. Makine buyruklarını gösteren diğer buyruklar benzer şekilde onaltılık kodlara dönüştürülür. (Çizelge 6.9)

Sembolik programdaki iki satır ondalık verileri DEC sözde buyruğu ile vermektedir. Üçüncü bir satırda HEX sözde buyruğu ile 0 değeri verilmiştir. Bu iş için DEC te kullanılabilirdi. 83 (ondalık) sayısı ikiliye çevrilmiş ve 106 adresine ondalık olarak yazılmıştır. Ondalık -23 negatif bir sayıdır ve ikiliye çevrilirken 2 ye göre tümleyen haline getirilmelidir. İkili sayının onaltılık eşdeğeri 107 adresine yazılmıştır. END sembolü programın bittiğini gösterir.

Adres Sembol Tablosu

Eğer sembolik programın tamamı iki defa taranırsa dönüşüm işlemi basitleştirilir. Birinci taramada hiçbir dönüşüm yapılmaz. Sadece her buyruk ve veri için bir bellek adresi ve yeri ayrılır. Bu atama işlemi başlıklara adres değeri ve ikinci taramada dönüşümü sağlar. Çizelge 6.9 da 100 adresi ORG dan sonra ilk buyruk için ayrıldı. Bundan sonra birbirini takip eden adresler programdaki satırlara verilir. Bu iş programın sonuna kadar devam eder. ORG ve END birer buyruk veya adres göstermedikleri için bunlara bir adres verilmez. Birinci tarama sona erince her bir başlık ve buna karşılık gelen adres çizelgeye yazılır. Burada adresler onaltılık olarak yazılmıştır. Bu program için adres-sembol çizelgesi aşağıdadır.

Adres sembolü	Onaltılık adres
MIN	106
SUB	107
DIF	108

İkinci tarama sırasında MRI buyrukları için adres bulmakta adres sembol çizelgesine başvurulur. Örneğin LDA, SUB kod satırı, ikinci tarama sırasında çizelge 6.1 den LDA nın onaltılık değeri alınarak, SUB ın onaltılık değeri ise yukarıdaki adres-sembol çizelgesinden alınarak oluşturulur. Bundan sonra iki parçayı birleştirerek 4 basamaklı onaltılık buyruk elde edilir. Onaltılık kodlar, eğer ikili değerlerin bellekte nasıl olacakları görmek istenirse kolayca ikiliye çevrilir.

Sembollerden ikiliye geçişte derleyici programı kullanılıyorsa birinci taramaya *ilk geçiş*, ikinciye *ikinci geçiş* denir.

6.4 Derleyici

Derleyici sembolik programı alır ve bunu birleştirici dil ikili eşdeğerine çevirir. Girilen sembolik programa *kaynak program*, sonuçta elde edilen ikili programa *amaç program* denir. Derleyici karakter dizileri üzerinde işlem yapar ve ikili karşılıklarını bulur.

Sembolik Programın Bellekteki Görünüşü

Derleme işlemine başlamadan önce sembolik program bellekte yerleştirilmiş olmalıdır. Kullanıcı sembolik programı klavyeden girer. Karakterlerin girişi için bir yükleyici program kullanılabilir. Program sembolik karakterler içerdiğinden, bellekteki görünümü alfasayısal karakter kodlarıyla oluşmalıdır. Temel bilgisayarımızda her bir karakter 8 bit kodla gösterilir. Önemli bit daima 0 dır. Kalan 7 bit ASCII olarak adlandırılır. ASCII kümesi onaltılık olarak çizelge 6.10 da verilmiştir. Her bir karakter iki onaltılık basamakla verilir. Bunlar kolayca 8 bit koda çevrilebilir. Çizelgedeki son karakter yazmak için değildir. Ekrandaki (yazıcıdaki) imlecin hareket ettirilmesi içindir. CR kodu ↵ tuşuna basılınca elde edilir. Bu kod icra edilince yeni satırın başına geçilir. Birleştirici CR kodunu bir kod satırının sonu olarak anlar.

Bir kod satırı ardışık adreslere yerleştirilir. Bir adres 16 bit (1 kelime) uzunluğunda olduğundan her adres sembolleri bir boşluk ile sona erer. Satır sonu CR ile bellidir. Örneğin bir satır kodu

```
PL3, LDA SUB I
```

Çizelge 6.10 Onaltılık karakter kodu

Karakter	Kod	Karakter	Kod	Karakter	Kod	
A	41	Q	51	6	36	
B	42	R	52	7	37	
C	43	S	53	8	38	
D	44	T	54	9	39	
E	45	U	55	boşluk	20	
F	46	V	56	(	28	
G	47	W	57	)	29	
H	48	X	58	*	2A	
I	49	Y	59	. +	2B	
J	4A	Z	5A	,	2C	
K	4B	O	30	-	2D	
L	4C	1	31	.	2E	
M	4D	2	32	,	2F	
N	4E	3	33	=	3 D	
O	4F	4	34	CR	0D	(yeni satır başı)

ardışık 7 bellek adresine yerleştirilir (çizelge 6.11). PL3 etiketi iki kelime içerir ve virgülle sona erer. Kod satırındaki buyruk alanı bir veya daha fazla sembol içerebilir. Her sembol boşluk (20) ile sona erer. Sadece son sembol CR (0D) ile sona erer. Eğer komut çizgisinde açıklama varsa birleştirici bunu /(2F) kodundan tanır. Derleyici açıklama alanındaki bütün karakterleri atlar ve CR kodunu arar. Bu kod bulununca son sembolden sonra yazılan boşluk CR kodu ile değiştirilir ve CR kodu satırın son karakteri olarak yazılır.

Derleyici programı için giriş, kullanıcının sembolik programıdır. Bu program ASCII koduyla yazılmıştır. Bu kod derleyici tarafından iki kez taranarak ikili program elde edilir. Dönüştürme işlemi sırasında derleyici tarafından yapılan görevleri açıklayalım.

Çizelge 6.11 PL3, LDA SUB I nın bilgisayarda gösterimi

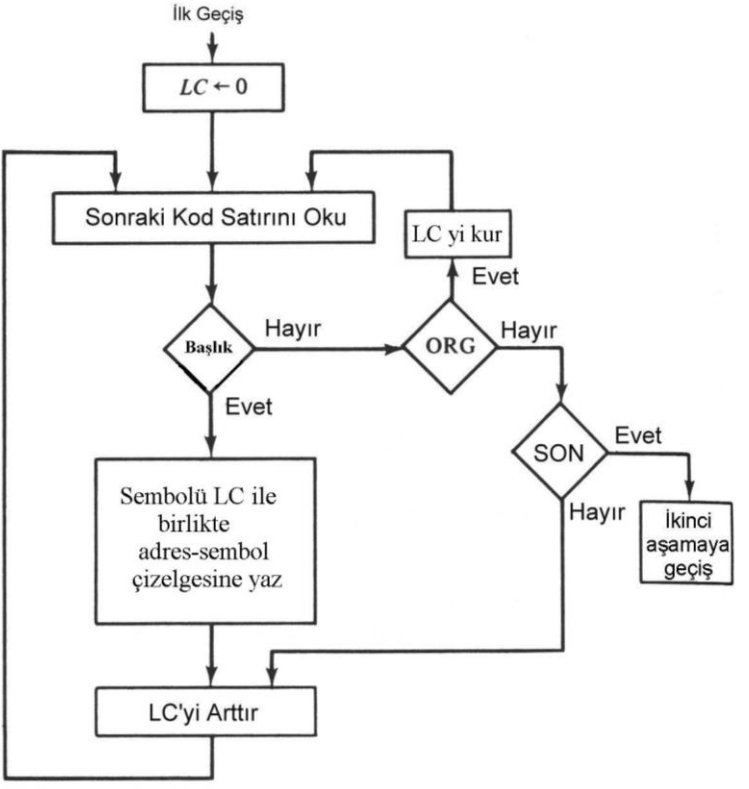
Bellek kelimesi	Sembol	Onaltılık kod		İkili gösterimi			
1	PL	50	4C	0101	0000	0100	1100
2	3 ,	33	2C	0011	0011	0010	1100
3	L D	4C	44	0100	1100	0100	0100
4	A	41	20	0100	0001	0010	0000
5	S U	53	55	0100	0011	0101	0101
6	B	42	20	0100	0010	0010	0000
7	I CR	49	0D	0100	1001	0000	1101

İlk Geçiş

Adres Sayıcı (LC)

İki geçişli bir derleyici sembolik programı iki defa tarar. İlk geçiş sırasında derleyici bir çizelge oluşturur ve bu çizelgeye kullanıcı tarafından tanımlanmış tüm adres sembollerini bunların ikili eşdeğerlerini yazar. İkili dönüşüm ikinci geçiş sırasında yapılır. Buyrukların adreslerinin sırasını takip edebilmek için derleyici bir bellek kelimesini kullanır. Bu adrese Adres sayıcı (LC) adı verilir. LC nin içeriği her bir yerleştirme işleminden sonra 1 arttırılır. ORG un olmaması halinde karışıklık olmaması için birleştirici LC yi önceden 0 yapar.

Derleyici tarafından ilk geçiş sırasında yapılan görevler 6.1 akış diyagramında gösterilmektedir. LC başlangıçta 0 yapılmaktadır. Daha sonra bir satır sembolik kod analiz edilir. Kodda bir başlık bulunup bulunmadığı, virgülün var olup olmaması yoluyla tespit edilir. Eğer satırda başlık yoksa sözde buyruğu varsa derleyici LC ye ORG dan sonra gelen rakamı yazar, sonra bir sonraki satıra gider. Eğer satırda END sözde buyruğu



Şekil 6.1 Derleyicinin birinci aşaması için akış şeması

Çizelge 6.12 Çizelge 6.8 deki program için adres sembol çizelgesi

Bellek kelimesi	sembol veya LC	Onaltılık kod	İkili gösterimi
1	M I	4D 49	0100 1101 0100 1001
2	N ,	4E 2C	0100 1110 0010 1100
3	(LC)	01 06	0000 0001 0000 0110
4	S U	53 55	0101 0011 0101 0101
5	B ,	42 2C	0100 0010 0010 1100
6	(LC)	01 07	0000 0001 0000 0111
7	D 1	44 49	0100 0100 0100 1001
8	F ,	46 2C	0100 0110 0010 1100
9	(LC)	01 08	0000 0001 0000 1000

varsa derleyici birinci geçişi bitirir ve ikinci geçişe geçer. ORG veya END içeren satırda Başlık bulunmaz. Eğer satırda başlık varsa adres-sembol çizelgesine başlık adresle beraber yazılır. Eğer bir başlık yoksa çizelgeye bir şey yazılmaz. Bundan sonra LC 1 arttırılır. Ve yeni bir satır işleme alınır.

Çizelge 6.8 deki program için derleyici, çizelge 6.12 deki adres-sembol çizelgesini yaratır. Her bir başlık sembolü iki bellek adresine yerleştirilir ve bir virgül ile sona erer. Eğer başlıkta 3 karakterden az karakter varsa bellek kelimeleri boşluk karakterleriyle doldurulur. Satır işleme girdiğinde LC içinde bulunan değer bir sonraki bellek adresine yerleştirilir. Programda 3 tane sembolik adres vardır, MIN, SUB ve DIF. Bu semboller onaltılık 106,107 ve 108'e karşılık gelen 12 bit adres içerirler. Adres-sembol çizelgesi her bir başlık için 3 kelime içerir.

İkinci Geçiş

Çizelgeye Bakma

Makine buyrukları ikinci geçiş sırasında dönüştürülürler. Dönüşümde çizelgeler kullanılır. Çizelgeye bakma işlemi, çizelgede verilen buyruğun aynısı olup olmadığına bakılır. Derleyici 4 çizelge kullanır. Programdaki her şey bu çizelgelerden birinde bulunmalıdır. Aksi takdirde dönüşüm işlemi yapılamaz. Dört çizelge şunlardır:

1. Sözde buyruk çizelgesi
2. MRI buyrukların çizelgesi
3. MRI olmayan buyrukların çizelgesi
4. Adres-sembol çizelgesi

Sözde buyruk çizelgesinin girişleri 4 tanedir, ORG, END, DEC ve HEX. Her bir giriş programda yer aldığında, bu buyruğun icrası için der-

leyici bir alt program gönderir. MRI çizelgesi 7 adet MRI işlem kodunu içerir. MRI olmayan çizelgesi 18 adet yazaç adreslemeli buyruğu ve G/Ç buyruklarını içerir. MRI olmayan buyruk girişlerinin karşılıkları olan 16 bitlik ikili değerler ikinci geçiş sırasında oluşturulur. Derleyici bu çizelgeyi arayarak dönüştürülecek sembolü bulur ve bunun ikili eşdeğerine erişir.

İkinci geçiş sırasında yapılan işler 6.2 deki akış diyagramında verilmektedir. LC önce 0 yapılır. Her bir satırın analizi bir defada yapılır. İkinci geçiş doğrudan buyruk alanına gittiğinden başlıklar ihmal edilir. Buyruk alanında sembollerden ilki araştırılır. Önce sözde-buyruk çizelgesine bakılır. Eğer ORG ile karşılaşırsa bu durumda derleyici bir alt programa gider ve LC ye bir ilk değer verilir. END ile karşılaşılırsa dönüşüm biter. Veri ile ilgili bir işlemci buyruğu verinin ikiliye çevrilmesine neden olur. Bu veri LC nin içindeki adrese yerleştirilir. Bundan sonra LC 1 arttırılır ve birleştirici bir sonraki satırdaki kodu analiz eder.

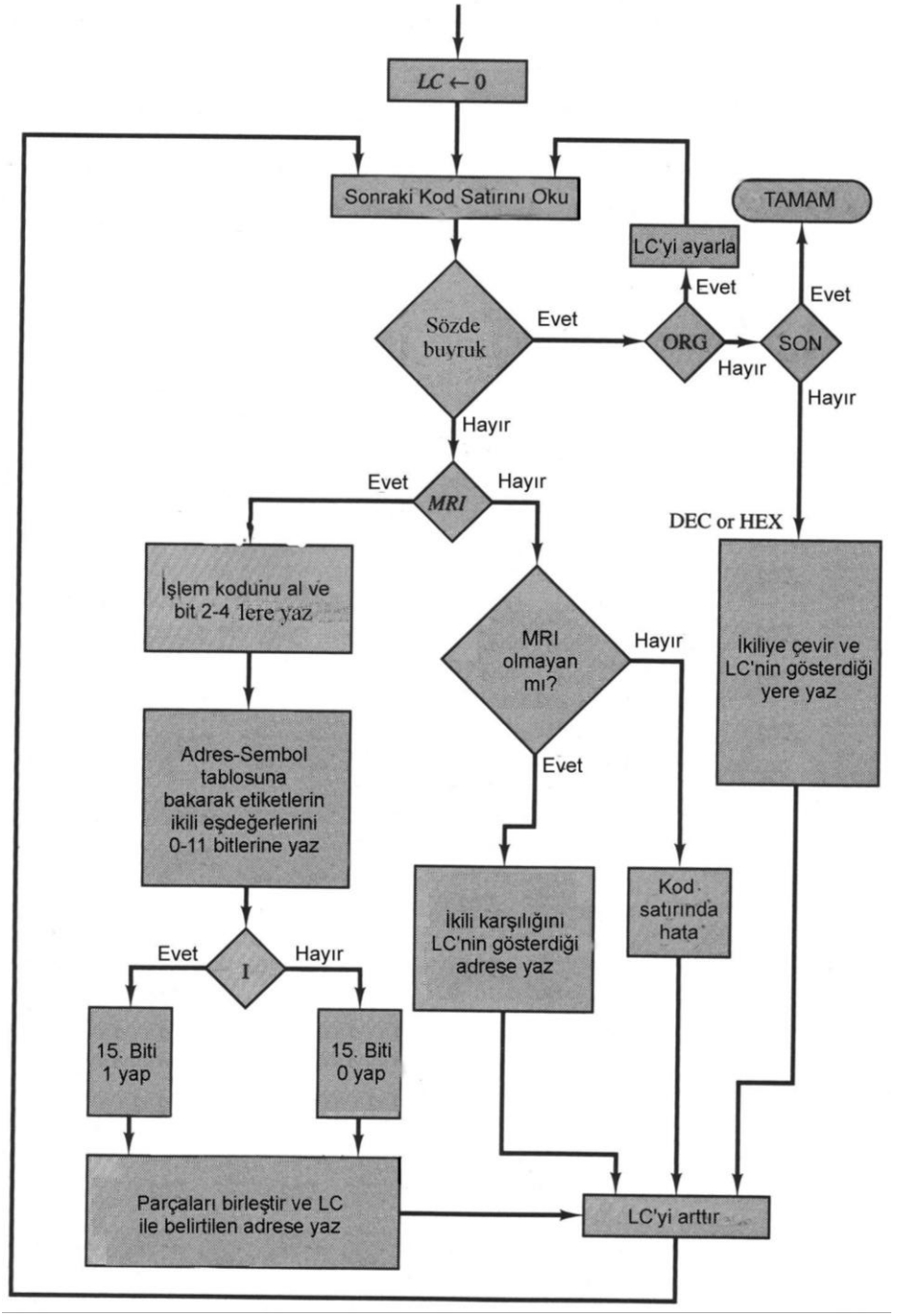
Eğer girilen sembol bir sözde buyruk değilse birleştirici MRI çizelgesine gider. Eğer sembolü burada bulamazsa MRI olmayan çizelgesine bakar. Bu çizelgede bulunacak bir sembol bir yazaç adreslemeli buyruk veya bir G/Ç buyruğudur. Derleyici bulduğu sembolün ikili karşılığını LC nin gösterdiği yere yazar. LC bir arttırılır ve yeni satıra geçilir.

Eğer sembol MRI çizelgesinde bulunursa derleyici 3 bit kodu alır ve kelimenin 2-4 bitlerine yerleştirir. Bir MRI buyruğu 2 veya 3 sembollü verilmiştir. İkinci sembol sembolik adrestir. Üçüncüsü var olabilir veya olmayabilir, varsa I dır. Sembolik adres, adres-sembol çizelgesi taranarak ikili koda çevrilir. I nın var olup olmamasına göre buyruğunu birinci biti 1 veya 0 yapılır. Buyruk kodunun 3 parçası birleştirilir ve LC tarafından verilen adrese yerleştirilir. LC bir arttırılır ve derleyici işleme devam eder.

Hata Bulma

Derleyicinin önemli görevlerinden biri sembolik programdaki hataları bulmaktır. Buna hata bulma denir. Bu tür hata, yanlış yazılmış bir sembol olabilir. Sembol MRI veya MRI olmayan çizelgesinde bulunmaz ise hata ortaya çıkar. Bir hata adresin sembolünün verilmeyişinden doğar. Her başlık programda iki yerde yazılmalıdır. Adres sembolüne karşılık gelen adresin başlığını adres-sembol çizelgesinde bulamayan derleyici adresi ikiliye çeviremez. Başka hatalarda olabilir ve pratik bir derleyici bu hataları yazmalıdır.

Pratik bir derleyici burada anlatılandan çok daha karmaşıktır. Birçok bilgisayar, kullanıcılara, derleyici programı yazmakta olanaklar verir. Örneğin adres, sembol veya sayı kullanılabilir. Bir çok derleyici adreslerin aritmetik bir bağıntı olarak verilmesine imkan verir. Derleyicinin daha kullanışlı hale gelmesiyle derleyici programı daha karışık olur.



Şekil 6.2 Derleyicinin ikinci aşaması için akış şeması

6.5 Program Döngüleri

Bir program döngüsü bir çok kez icra edilen buyruklar sırasıdır. Her bir çalışmada veriler farklı olabilir. Döngü fortran da DO deyimi ile tanımlanır. Fortran da 100 tamsayının toplamı ile ilgili örnek

```
DIMENSION A(100)
INTEGER TOPLAM, A
TOPLAM = 0
DO 3 J = 1, 100
3 TOPLAM = TOPLAM + A(J)
```

şeklindedir. 3 satırı 100 kez A(J) farklı değerleri için tekrarlanmaktadır.

Derleyici

Yukarıdaki gibi bir yüksek seviyeli programlama dilini bir makine diline çeviren sistem programına derleyici denir. Derleyici, bir çeviriciden daha karmaşıktır ve işleyişini tam olarak anlamak sistem programlama bilgisi gerektirir. Fakat bir derleyicinin temel fonksiyonlarını yukarıdaki programı bir çevirici dil programına çevirmek sureti ile gösterebiliriz. Derleyici, bir çevirici dili çeviride bir ara aşama olarak kullanılabilir yada programı doğrudan ikili koda çevirebilir.

Derleyici önce A(100) için 100 kelime yer açar. Değişkenlere değerler bir giriş durumuyla verilir. (Bu satır programda yoktur). İkinci durum derleyiciye sayıların tamsayı olduğunu bildirir. Eğer sayılar gerçel tipte olsaydı, derleyici hem dizi hem diğerleri için yer açacaktı, ayrıca aritmetiğin kayan noktalı aritmetik olması için tedbir olacaktı. İlk iki deyim icra edilemez deyimlerdir. Bunlar derleyici de sözde buyruk gibi yer alırlar. Varsayalım ki derleyici dizinin 100 elemanı için $(150)_{16}$ dan $(1B3)_{16}$ ya kadar yer ayırsın. Bu ayrılan bellek kelimeleri dönüşmüş programda (Çizelge 6.13) 19 dan 118 e kadar yer alırlar. Bu iş 18 inci satırdaki ORG sözde buyruğu ile yapılır. İlk ve son veriler programda ondalık olarak gösterilmiştir. Bunlar aslında program çalışmadan belli değildir ve bu verilerin programa bir giriş deyimi ile girileceği varsayılmıştır. Ancak basitlik için bu kısım programa konmamıştır. Derleyici önce bu alanı ayırır sonra değerleri alarak buraya yerleştirilir. Sembolik programdaki satır numaraları dönüştürülmüş programa ait değildir. Sadece kolaylık için yazılmıştır.

DO deyimi 2-5 satırlarında, sabitler ise 13-16 satırlarında bulunmaktadır. Fortran programında 3 nolu satır 100 kez çalışarak toplam hesaplanır. Birinci verinin adresi (150), 13 cü satırda ADS sembolik adresi ile verilmiştir. ADS değeri *AC* ye yüklenir ve *AC* den PTR ye aktarılır. Bundan sonra –100 CTR ye aktarılır. Bu alan bir sayaç olarak çalışır ve her döngü sonunda 1 azaltılır. CTR 0 olduğunda döngü tamamlanmış olur. Program döngüden çıkar.

Çizelge 6.13 100 sayının toplamını yapan sembolik program

Sıra			
1		ORG 100	/Program bellekte 100 üncü adreste bulunmakta
2		LDA ADS	/İşlenenin ilk adresini yükle
3		STA PTR	/Göstergeye aktar
4		LDA NBR	/ –100 ü yükle
5		STA CTR	/Sayıcıya aktar
6		CLA	/*AC* yi sil
7	LOP,	ADD PTR I	/Bir ileneni *AC* ile topla
8		ISZ PTR	/Göstergeyi 1 arttır
9		ISZ CTR	/Sayıcıyı 1 arttır
10		BUN LOP	/LOP a şartsız dallan
11		STA TPM	/TOPLAM a yükle
12		HLT	/Programı durdur
13	ADS,	HEX 150	/Verilerin ilk adresi
14	PTR,	HEX 0	/Bu adres gösterge için ayrılmıştır
15	NBR,	DEC -100	/İlkleştirilmiş sayaç için sabit
16	CTR,	HEX 0	/Bu adres sayaç için ayrılmıştır
17	SUM,	HEX 0	/Toplam buraya yüklenecektir
18		ORG 150	/Verilerin bellekteki başlangıç adresi
19		DEC 75	/İlk veri
•			
•			
•			
118		DEC 23	/Son veri
119		END	/Sembolik programın sonu

Bazı derleyiciler TOPLAM = 0 icra için TOPLAM ın adresine 0 yazar. Döngüdeki her icrada bu adrese yazma yapılır. Daha iyi bir derleyici bu toplamı işlemci yazacında toplayabileceğini anlar. Ancak son toplam TPM adresine yazılır. Bu derleyici 6 cı satırda CLA ile *AC* yi temizler. Sonra döngünün bitmesiyle *AC* deki toplamı TPM adresine yazar.

Programdaki döngü 7-10 satırlardadır. 7 ci satır dolaylı bir ADD buyruğudur. (*I* var). Geçerli verinin adresi PTR adresinde bulunmaktadır. Bu adres dolaylı olarak işaret edildiğinde bilgisayar, PTR nin değerini verinin adresi olarak alır. Bunun sonucu olarak 150 adresindeki veri *AC* ye eklenir. PTR deki değer bundan sonra arttırılır. Arttırmayı ISZ yapar. Böylece adresin değeri bir sonraki veriyi gösterir. CTR adresi eğer CTR sıfır değilse arttırılır. Bilgisayar bir sonraki buyruğu atlamaz. Atlanmayan buyruk BUN buyruğudur. Program çalışmasını döngü başına götürür. Böylece döngü bir defa daha icra edilir. Döngü 100 defa icra edildikten sonra CTR adresinin değeri sıfıra erişir. Bir sonraki buyruk (yani BUN) atlanır. Bilgisayar 11 ve 12 inci satırları icra eder. *AC* içindeki toplanmış değer TPM adresine aktarılır. Sonra bilgisayarın çalışması durur. HLT buyruğu buraya açıklık getirmesi için konmuştur. 8 inci satırdaki ISZ

buyruğu sadece arttırma için konmuştur. PTR bir adrestir ve adres 0 olmayacağından asla atlama işlemi olmaz.

Gösterge Sayacı

Çizelge 6.13 teki programda gösterge ve sayaç kavramları verilmiştir. Bir döngü oluşturulurken geçerli verinin adresi gösterge tarafından belirtilir. Sayaç ise döngünün kaç defa döndüğünü sayar. Bu fonksiyonlar iki adreste verilmiştir. Bir taneden fazla işlemci yazacı bulunan bilgisayarlarda bu yazaçlardan bir tanesi gösterge olarak, diğeri sayaç olarak, üçüncüsü de *AC* olarak kullanılabilir. Bunlar gösterge ve sayaç olarak kullanıldıkları zaman *indis yazaçları* adı verilir. Bu tip yazaçlar bölüm 8.5 te incelenecektir.

6.6 Aritmetik ve Mantıksal İşlemlerin Programlanması

Büyük sistemlerde buyrukların sayısı birkaç yüz olabilir. Küçük sistemlerde ise birkaç düzine civarındadır. Bazı bilgisayarlar bir işlemi bir makine buyruğu ile yaparlar, başkaları ise çok sayıda buyruk kullanır. Örnek olarak bazı bilgisayarların 4 işlem için buyrukları vardır. Diğerlerinin, örneğin temel bilgisayarın tek bir ADD buyruğu vardır. Diğer matematiksel işlemler program yoluyla yerine getirilmelidir. Çizelge 6.8 de iki sayıyı çıkarmak için benzer biçimde program yapılabilir.

Bilgisayar içinde tek bir makine buyruğu ile yerine getirilebilen işlemlere donanım tarafından yapılıyor denir. Buna karşılık bir küme buyruk yardımı ile yapılan işlemlere yazılımca yapılıyor denir. Bazı bilgisayarlarda çok sayıda ortak işlemin hızla yapılabilmesi için çok kalabalık buyruk kümesi vardır. Diğerlerinin ise daha küçük buyruk kümesi vardır ve bir çok işlemin yapılabilmesi için yazılıma ihtiyaç vardır. Donanımla işlemlerin yapılması pahalıdır. Çünkü çok sayıda ek devreye gerek vardır. Yazılım yöntemi ise uzun programlara neden olur. Bunlarda hem buyruklar kalabalık hem de icra süresi uzundur.

Bu bölümde birkaç aritmetik ve mantıksal işlem yazılımla yapılacaktır. Herhangi bir aritmetik işlem için program yapılabilir. Bu programlarda sabit noktalı aritmetik veya kayan noktalı aritmetik kullanılabilir. Aritmetik işlemlerin donanımla yapılması 9 ve 10 cu bölümlerde incelenecektir.

Çarpma Programı

8 bitlik iki sayıyı çarpmak için bir program düşünülsün. Basitleştirmek için işaret bitlerine dikkat etmeyip sayılar pozitif kabul edilmektedir. Bu yapılacak programı 8 bitten daha büyük (16 bit) veya işaretli sayılara

genişletmek mümkündür. Çarpım 31 bit olur ve bellekte 2 kelime yer kaplar.

Şekil 6.3 teki sayısal örnekte gösterildiği gibi işlem çarpanın *Y* nin bitlerini kontrol ve çarpılan (*X*) i *Y* içindeki 1 ler kadar toplamaktır. *X* in değeri her *Y* biti için bir sola kaydırılır. 1 olmayan yerler için 0 konur ve her işlemde iki sayı toplanır. Bellekteki *P* adresi bu ara toplamlar için ayrılmıştır. Bu ara toplamlara kısmi sonuçlar denir. Çünkü çarpım tamamlanana kadar (yani bütün toplamlar tamamlanana kadar) bunlar ara sonuçtur. Örnekte görüldüğü gibi kısmi çarpım 0 ile başlar. Çarpılan *X P* nin içeriğine *Y* nin 1 olan her biti için bir defa eklenir (sola kaydırılarak). *P* nin son durumu çarpımı verir. Sayısal örnekte 4 önemli bit var. Çarpım sonucu, çarpım 8 önemli bit içerir. Bilgisayar 8 önemli biti olan sayıları çarpmak için 16 bite kadar çarpım alanı kullanır.

Şekil 6.3 teki akış şemasında programda 8 defa geçilen bir döngü vardır. Döngü çarpanın her önemli biti için bir defa icra edilir. İlk olarak *X* çarpılanı, *Y* ise çarpanı tutar. Sayaç CTR -8 yapılır ve *P* adresi sıfırlanır.

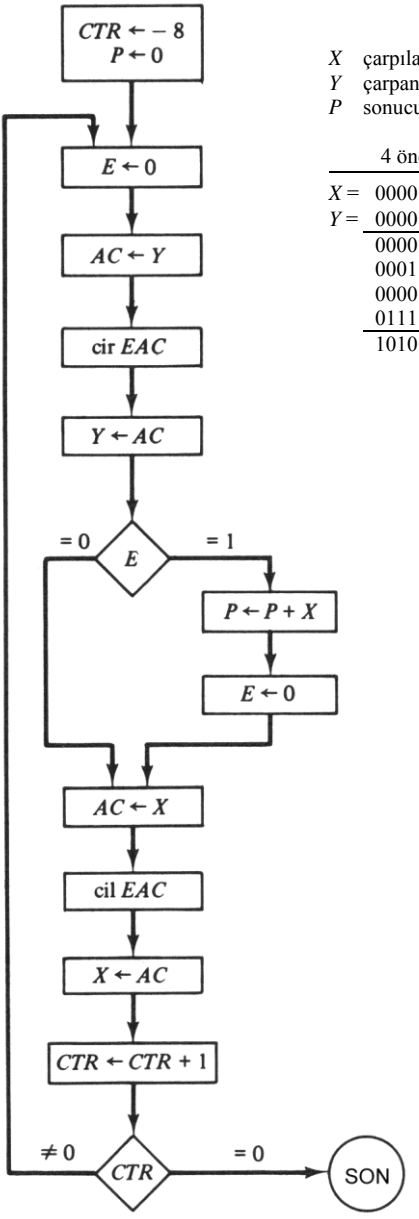
Çarpan biti eğer *E* ye aktarılırsa kontrol edilebilir. Bunu sağlamak için *E* temizlenir, *Y* değeri *AC* ye yüklenir *E* ve *AC* sağa döndürülür kaydırılan sayı tekrar *Y* adresine yazılır. *E* ye konan bit çarpanın en düşük önemsiz bitidir. Eğer 1 ise çarpılan kısmı çarpıma eklenir. Eğer 0 ise kısmı çarpım değişmez. Bundan sonra *X* değeri *AC* ye yüklenerek bir sola kaydırılır. Kaydırma sola kaydırma olarak *E* ve *AC* için yapılır. Döngü CTR adresi arttırılarak 8 defa tekrarlanır. CTR nin 0 olup olmadığı kontrol edilir. CTR 0 olunca program döngüden çıkar. Çarpım *P* dedir.

Çizelge 6.14 teki program işaretsiz iki sayının çarpımını sağlayan buyrukları içerir. Programda değerler verilmemiştir. Bu işlemle çarpan ve çarpılan sayılar *X* ve *Y* adreslerine yerleştirilir. Sayaca –8, *P* adresine 0 yazılır. Eğer bu değerler programa verilmez ise program yanlış veri ile çalışır.

Bu örnek eğer makine de istenen buyruklar yoksa bunun programlanabileceğini gösterir. Donanım 10.3 te verilecektir.

Çift Duyarlıklı Toplama

İki adet 16 bit sayı çarpılırsa sonuç 32 bit olur. Bu sonuç 2 kelimeye yerleştirilebilir. İki kelimeye yerleştirilen sayıya çift duyarlıklı denir. Kısmi çarpım hesaplanırken çift duyarlıklı bir sayının kaydırılmış çarpılana eklenmesi gerekir. Tabi oda çift duyarlıklıdır. Daha iyi bir duyarlık için programcı çift duyarlıklı sayıları kullanmak ve aritmetik veriler için bellek kelimesi işgal etmek ister. Şimdi çift duyarlıklı sayılar için toplama programı yapalım.



X çarpılanı tutar
Y çarpanı tutar
P sonucu tutar

4 önemli bit için örnek

			P	
X =	0000	1111		
Y =	0000	1011	0000	0000
	0000	1111	0000	1111
	0001	1110	0010	1101
	0000	0000	0010	1101
	0111	1000	1010	0101
	1010	0101		

Şekil 6.3 çarpma programı akış şeması

Çizelge 6.14 İki pozitif sayının çarpımı ile ilgili program

```
        ORG  100
LOP,    CLE          /E yi temizle
        LDA  Y       /Çarpanı yükle
        CIR          /Çarpan biti E ye aktar
        STA  Y       /Kaydırılmış çarpanı sakla
        SZE          /E = 0 mı diye bak
        BUN  ONE     /bit = 1 ise ONE a git
        BUN  ZRO     /bit = 0 ise ZRO ya git
ONE,    LDA  X       /çarpılanı yükle
        ADD  P       /Kısmi çarpıma ekle
        STA  P       /Kısmi çarpımı sakla
        CLE          /E yi temizle
ZRO,    LDA  X       /Çarpılanı yükle
        CIL          /Sola kaydır
        STA  X       /Kaymış çarpılanı sakla
        ISZ  CTR     /Sayacı arttır
        BUN  LOP     /Sayaç sıfır değil; döngüyü tekrarla
        HLT          /Sayaç sıfır; durdur
CTR,    DEC  –8      /Bu adres sayaç olarak çalışır
X,      HEX  000F    /Çarpılan buraya saklanır
Y,      HEX  000B    /Çarpan buraya saklanır
P,      HEX  0       /Sonuç buraya saklanır
        END
```

Çift duyarlıklı sayılardan birisi ardışık iki bellek adresine yerleştirilsin Bunlar AL ve AH olup AL düşük seviyeli 16 biti içerir. Diğer sayı BL ve BH tadır. Program 6.15 te verilmektedir. İki düşük seviyeli kısım toplanır ve elde *E* ye konur. Sonra *AC* temizlenir ve *E* nin değeri *AC* nin önemsiz bitine aktarılır. Sonra sayıların önemli kısımları *AC* ye eklenir. Çift duyarlıklı toplam CL ve CH a konur.

Çizelge 6.15. İki çift duyarlıklı sayının toplanması

```
        LDA  AL      /A nın düşük seviyeli bitlerini AC ye yükle
        ADD  BL      /B nin düşük seviyeli bitlerini AC de toplayıp elde yi E ye aktar
        STA  CL      /AC deki toplamı CL ye aktar
        CLA          /AC yi sil
        CIL          /E ve AC sola kaydırılarak E yi AC nin en önemsiz bitine aktar
        ADD  AH      /AC ile A nın yüksek seviyeli bitlerini topla
        ADD  BH      /AC ile B nin yüksek seviyeli bitlerini topla
        STA  CH      /AC dekini CH a aktar
        HLT          /Dur
AL,     —            /Verilerin yerleri
AH,     —
BL,     —
BH,     —
CL,     —
CH,     —
```

Mantıksal İşlemler

Temel bilgisayarımızda 3 tane mantıksal işlem yapan makine buyruğu vardır. AND, CMA ve CLA. LDA buyruğu *AC* ye taşıyan mantıksal buyruk olarak düşünülebilir. Bölüm 4.5 te 16 farklı mantık işlemi vardı. Bu işlemlerin hepsi yazılım olarak, AND ve CMA işlemleriyle elde edilebilir. Örneğin VEYA işlemi temel bilgisayarda buyruk olarak yoktur. DeMorgan teoreminden $x + y = (x'y')'$ olduğunu biliyoruz. Birde VE ve tümle işlemlerinin bilgisayar buyruklarında olduğunu. Böylece VEYA işlemini yapan bir program

```
LDA A    /A nın değerini AC ye yükle
CMA      /Ā nın tümleyenini al
STA TMP  /AC yi TMP ye aktar
LDA B    /B nin değerini AC ye yükle
CMA      /B̄ nin tümleyenini al
AND TMP  /A ile VE le (Ā ∧ B̄)
CMA      /Tekrar tümleyen alınarak A ∨ B elde edilir
```

şeklindedir. Diğer mantıksal işlemler aynı biçimde yazılımla elde edilir.

Kaydırma İşlemleri

Temel bilgisayarda dairesel kaydırma işlemleri makine buyrukları olarak vardır. Diğer aritmetik ve mantıksal kaydırma işlemleri az sayıda buyruklar ile programlanabilir.

Mantıksal kaydırma işlemi en uç bitlere 0 girilmesini gerektirir. Bu durum *E* 0 yapılarak *AC* ve *E* nin dairesel döndürülmesi ile elde edilir. Dolayısıyla mantıksal bir sağa kaydırma işlemi için iki buyruğa gerek vardır.

```
CLE
CIR
```

Aynı biçimde mantıksal bir sola kaydırma için iki buyruk

```
CLE
CIL
```

Aritmetik kaydırma, negatif sayıların nasıl gösterildiğine bağlıdır. Temel bilgisayarda negatif sayılar 2 ye göre tümleyen şeklinde kullanılmaktadır. Sağa bir aritmetik kaydırma için en soldaki işaret biti değişmemelidir. Sağa aritmetik kaydırma programı *E* yi işaret biti ile aynı değere getirir. Sonra sağa kaydırır.

```
CLE   /E yi siler
SPA   /AC > 0 ise sonraki satırı atla
CME   /E nin tümleyenini al
CIR   /E AC yi dairesel sağa kaydır.
```

Aritmetik sola kaydırmada en önemsiz bite eklenecek değer 0 yapılarak kolay elde edilir. Kaydırma sırasında işaret biti değişmemelidir. CIL buyruğu ile işaret biti *E* ye kayar. Bundan dolayı işlemden sonra işaret biti ile *E* karşılaştırılmalıdır. Eğer ikisi eşit ise aritmetik kaydırma doğru olarak yapılmış demektir. Eğer eşit değil ise taşma durumu oluşur. Taşma oluşunca kaydırılmamış sayının çok büyük olduğu anlamına gelir. Kaydırma yoluyla 2 ile çarpılınca elde edilen sayı *AC* nin kapasitesini aşmaktadır.

6.7 Alt Programlar

Bazen bir program parçasının programın bir çok yerlerine tekrar tekrar yazılması gerekir. Bunun yerine bu parça bir defa yazılabilmeli ve birçok defa çağrılabilmelidir. Buna alt program denir. Bir alt program icrası bitince çağıran programa geri dönülür.

Bir alt program genelde tam bir programdır ve verilen bir işi yapar. Dallanma ana programın herhangi bir yerinden alabilir. Bir programdan bir alt program birçok defa çağrıldığından, dönüşte alt programın nereye döneceği önemli bir konudur, bu yüzden dönüş adresinin bir yere yazılması gerekir. Bütün bilgisayarlarda çağrılma ve dönüşü sağlayacak buyruklar vardır.

Temel bilgisayar da ana program ile alt program arasındaki bağlantı BSA ile sağlanır. Bu buyruğun nasıl çalıştığını görmek için *AC* nin içeriğini 4 defa sola kaydıran bir alt program yazılsın. Bir kelimenin 4 defa kaydırılması ikili kodlanmış ondalık sayılar (BCD) ve alfanümerik karakterler için faydalıdır. Böyle bir işlem için bazı bilgisayarlarda buyruk yoktur. Çizelge 6.16 daki program *X* değerini *AC* ye yükleyerek başlıyor. Sonraki buyruk BSA SH4 tür. BSA buyruğu 101 adresindedir. SH4 alt programı işini bitirince 102 adresine dönmelidir. BSA icra edilince kontrol birimi 102 dönüş adresini SH4 sembolik adresi ile (109) verilen adrese yazar. Daha sonra SH4 + 1 değerini *PC* a yazar. Bu buyruk icra edildikten sonra 109 bellek adresi 102 nin onaltılık karşılığını içerir. *PC* ise onaltılık 10A nın ikili karşılığını tutar. Bu hareketle alt programın dönüş adresi saklanmış ve alt programın icrası 10A adresinden itibaren başlamış olur.

Çizelge 6.16 Alt program kullanan bir program gösterimi

Adres		Program		Açıklama
		ORG	100	/Ana program
100		LDA	X	/X i *AC* ye yükle
101		BSA	SH4	/Alt programa git
102		STA	X	/*AC* yi X e aktar
103		LDA	Y	/*AC* ye Y yi aktar
104		BSA	SH4	/Alt programa git
105		STA	Y	/*AC* yi Y ye aktar
106		HLT		/Dur.
107	X,	HEX	1234	
108	Y,	HEX	4321	
				/4 kez sola kaydıran alt program
109	SH4,	HEX	0	/Buraya geri dönüş adresini yaz
10A		CIL		/Bir kez sola kaydır
10B		CIL		
10C		CIL		
10D		CIL		
10E		AND	MSK	/*AC* nin 13–16 bitlerini sıfırla
10F		BUN	SH4	/Ana programa dön
110	MSK,	HEX	FFF0	/Maskeleme işleneni
		END		

Alt program içindeki işlem *AC* nin içeriğini 4 defa sola kaydırır. Mantıksal kaydırma işlemi yapılması için düşük mertebeli bitler 0 yapılmıştır. Bunun için FFF0 ile *AC* VE lenmiştir. Bir maskeleme işlemi mantıksal VE işlemidir. Mask yazacının bitlerine karşılık gelen *AC* deki bitler 0 lanır. Eğer Mask yazacının bitleri 1 ise karşılık gelen *AC* deki bitler değer değiştirmez.

Alt programdaki son buyruk alt programı ana programa döndürür. Bu dolaylı adreslemeli bir buyruktur. Adres alt programın adıyla verilmiştir. SH4 adresinde dönüş adresi bulunur (102). Bilgisayar 102 adresine döner kaydırılmış sayıyı X adresine yerleştirilir. Yeni bir sayı Y adresinden *AC* ye yüklenir. Sonra alt programa bir daha dallanır. Bu defa SH4 adresinde 105 vardır. Çünkü BSA dan sonraki buyruğun adresi 105 tir.

Buradan anlaşılan her alt programın ilk bellek adresi alt programdan ana programa dönüş adresinin bulunduğu yerdir. Alt programa gidiş ve oradan ana programa dönüşe bağlantı denir. BSA nın yaptığı işleme alt program çağırma denir.

Alt programın son buyruğuna alt programdan geri dönüş denir. Burada tanımlanan dallanma işlemi bir tek işlemci yazacı bulunan bilgisayarlarda böyledir. Çok sayıda işlemci yazacı bulunan bilgisayarlarda bu yazaçların bazılarına indis yazaçları denir. Yeni dönüş adresi indis yazacında tutulur.

Alt Program Parametreleri ve Veri Bağlantıları

Alt program çalıştığında ana program, alt programın çalışması için gerekli ve ana programda bulunan verileri alt programa aktarması gerekir. Önceki örnekte veri *AC* yoluyla aktarıldı. Dallanmadan önce veri *AC* ye aktarıldı. *AC* kaydırıldı ve ana program bunu dönüşte istediği bir yere (burada X) yazdı. *AC* alt programa tek bir giriş ve çıkış parametresi olarak kullanılabilir. Çok sayıda işlemci yazacı olan bilgisayarlarda çok sayıda parametre bu yolla aktarılabilir. Alt programa veri aktarmanın bir diğer yolu belleği kullanmaktır. Veriler blok şeklinde de bulunabilirler. Bu takdirde veri bloğunun başlangıç adresi çağırma işleminden sonra belleğe yazılır. Hangi halde olursa olsun dönüş adresi ana program ile alt program arasındaki bağlantıyı kurar.

Örnek olarak mantıksal VEYA işlemi yapan bir alt program düşü-/nülsün. Alt programa iki veri gönderilmekte ve alt program işlemin sonucunu geri döndürmektedir. *AC* verinin birini aktarabilir ve sonucu geri alabilir. Diğer veri BSA buyruğunu takip eden adrese konmuştur. Durum çizelge 6.17 deki programda görülüyor. X adresindeki ilk veri *AC* ye yüklenir. İkinci veri BSA yı takip eden 202 adresine yerleştirilmiştir. Dallanmadan sonra alt programdaki ilk adreste 202 var. Bu 202 dönüş adresi değildir, ikinci verinin adresidir. Alt program VEYA işlemine başlar ve birinci verinin tümleyenini alır, sonra geçici bir TMP adresine saklar. Sonra ikinci veriyi *AC* ye, OR adresi yardımıyla dolaylı olarak akta-

Çizelge 6.17 Parametre aktarımı ile ilgili program

Location			
		ORG 200	
200		LDA X	/İlk veriyi *AC* ye yükle
201		BSA OR	/OR alt programa git
202		EX 3AF6	/İkinci veri burada saklanmakta
203		STA Y	/*AC* dekini Y ye aktar
204		HLT	/Programı durdur
205	X,	HEX 7B95	/İlk veri burada saklanmakta
206	Y,	HEX 0	/Sonuç buraya saklanacak
207	OR,	HEX 0	/OR alt programı
CMA			/İlk verinin tümleyenini al
STA	TMP		/*AC* yi geçici yazaca yaz
20A		LDA OR I	/*AC* ye ikinci veriyi yükle
20B		CMA	/İkinci verinin tümleyenini al
20C		AND TMP	/İlk veri ile VE le
20D		CMA	/VEYA işlemini elde etmek için tümleyenini al
20E		ISZ OR	/Geri dönüş adresini 1 arttır
20F		BUN OR I	/Ana programa dön
TMP,	HEX	0	/geçici sonuç değişkeni
END			

rır. OR adresinde 202 sayısı var. Buyruk bu adrese dolaylı olarak ulaştığında 202 adresindeki veri *AC* ye yüklenir. Alt program bu veriyi tümler ve TMP deki veri ile VE ler. Bunu da tümleyince sonuç elde edilir.

Alt programdan dönüşte ana program 203 adresinden devam etmelidir. Bunun için OR adresi ISZ ile arttırılır. Şimdi OR adresi 203 sayısını içerir ve dolaylı BUN buyruğu uygun yere dönüşü sağlar.

BSA buyruğundan sonra bir taneden fazla veri bulunabilir. Çağıran programdan alt programa bir veri kullanınca dönüş adresi 1 arttırılmalıdır ve her veriden sonra bu devam etmelidir. Böylece ana program içinde istenen yere dönülebilir. Bundan başka ona program alt program tarafından bulunan sonuçları saklamak için yer ayırmış olmalıdır. Alt program içindeki ilk adresteki değer bu nedenle arttırılmalıdır. Eğer çok miktarda veri varsa bu veriler blok halinde saklanır. Bloktaki ilk verinin adresi bağlama parametresi olarak kullanılabilir.

100 adresinden başlayan bir blok veriyi 200 adresinden başlayan bir bloğa gönderen bir alt program çizelge 6.18 de verilmiştir. Blok uzunluğu 16 kelimedir. İlk buyruk alt program MVE ye dallanmadır. Alt programın ilk parçası 3 parametreyi 100, 200 ve -16 yı ana programdan kendi yerleştirme bölgesine taşır. İki gösterge yardımıyla veriler aktarılır. Sayaç döngünün 16 defa dönmesini sağlar. Sonuçta HLT a dönülür.

Çizelge 6.18 Veri bloğunun taşınması alt programı

```
                        /Ana program
BSA     MVE             /Alt programa dallanma
        HEX   100       /Kaynak verinin başlangıç adresi
        HEX   200       /Hedef verinin başlangıç adresi
        DEC   –16       /Aktarılacakların sayısı
        HLT
MVE,    HEX   0         /MVE alt program
        LDA   MVE I     /kaynak verinin başlangıç adresini AC ye aktar
        STA   PT1       /AC yi PT1 göstergesine yükle
        ISZ   MVE       /Dönüş adresini arttırır.
        LDA   MVE I     /Hedefin başlangıç adresini AC ye getir
        STA   PT2       /AC yi PT2 ye yükle
        ISZ   MVE       /(MVE) Dönüş adresini 1 arttır
LOP,    LDA   PT1   I   /PT1 in gösterdiği adresin içeriğini AC ye aktar
        STA   PT2   I   /AC nin içeriğini PT2 nin gösterdiği adrese aktar
        ISZ   PT1       /PT1 in değerini 1 arttır
        ISZ   PT2       /PT2 nin değerini 1 arttır
        ISZ   CTR       /CTR nin değerini 1 arttır
        BUN   LOP       /Döngüyü 16 Kez tekrarla
        BUN   MVE I     /Ana programa dön
PT1,    —
PT2,    —
CTR,    —
```

6.8 Giriş/Çıkışın Programlanması

Kullanıcılar bilgisayarlarını bir programlama diliyle yazarlar Bu dille yazılan programlar karakter dizilerinden oluşur. Her karakter ikili koda çevrilir ve makineye girilir. Giriş için INP ve çıkış için OUT komutları kullanılır. OUT gelen ikili kodu karaktere çevirir.

Çizelge 6.19(a) da bir karakter girişini ve çıkışını veren iki program vardır. Karakterler belleğe yerleştirilir. SKI buyruğu aktarılacak karakter olup olmadığını anlamak için giriş bayrağını kontrol eder. Bayrak biti 1 ise BUN atlanır. INP buyruğu BCD karakterini AC(0–7) ye aktarır. Bu karakter OUT buyruğu ile yazdırılır. SKI ile giriş bayrağı kontrol edilir. Eğer 1 ise sonraki buyruk atlanır. Eğer 0 ise giriş bayrağını kontrol etmek için başa döner. Bayrak 1 olunca giriş karakterini alıp bayrağı temizler. Karakteri yazıcıya aktarır. *AC* deki karakteri belleğe aktarır.

Çizelge 6.19(b) deki program bellekte daha önce yerleştirilmiş bir karakteri basar. İlk olarak karakter *AC* ye yüklenir. Çıkış bayrağı kontrol edilir. Eğer bayrak 0 ise bilgisayar ikinci buyruk ile döngüye girer. Bayrak 1 e değiştiğinde karakter *AC* den yazıcıya aktarılır.

Karakter İşleme

Bilgisayar sadece bir hesap makinesi değil aynı zamanda bir karakter işleyici (kelime işlemci) dir. Böyle bir işlem iki karakterin bir kelimeye yerleştirilmesi olabilir. Böylece bellekte bir kelime doldurulabilir. Çizelge 6.20 deki IN2 programı iki karakteri 16 bit kelimeye yerleştirir. Paketlenen kelime (2 karakter toplam 16 bit) *AC* de kalır. Çizelge 6.16 daki SH4 alt programı iki defa çağrılarak *AC* deki bilgi 8 defa sola kaydırılır.

Çizelge 6.19 Bir karakter giriş, çıkış programları

```
(a) Bir karakter alma
    CIF,    SKI          /giriş bayrağını kontrol et 1 ise bir sonraki komutu atla
            BUN   CIF    /Bayrak = 0 ise CIF a git
            INP          /bayrak = 1 ise giriş karakterini al
            OUT          /karakteri bas
            STA   CHR    /AC dekini belleğe aktar
            HLT
    CHR,    —
(b) Bir karakterin basılması
            LDA   CHR    /karakteri AC ye yükle
    COF,    SKO          /bayrağı kontrol et 1 ise bir sonraki komutu atla
            BUN   COF    /bayrak = 0 ise COF a git
            OUT          /karakter çıkışa gönder
            HLT
    CHR,    HEX   0057   /karakter "W" dir
```

Sembolik program belleğin buffer denen kısmında tutulur. Sembolik program bir giriş biriminden buffer alanına girilir. Çizelge 6.21 deki program klavyeden bir sembolik program girişi için kullanılabilir. Girişte iki karakter 1 kelimeye yerleştirilir. Buffer in ilk adresi 500 dür. İlk iki karakter 500 adresine yerleştirilir. Sonrakiler takip eden adreslere yazılırlar. Program buffer daki ilk başlangıç adresi bulmak için bir gösterge kullanır. Sayaç kullanılmayarak programın tamamının girilmesi sağlanır. Eğer FFFF adresini taşarsa burada durur. Çünkü daima buffer a bir boyut vermek gerekir. (Burada FFFF = 2^{16} = 65535 = 64K)

Çizelge 6.20 İki karakterin paketlenmesi ve aktarılması alt programı

```
IN2,    —
FST,    SKI
        BUN  FST
        INP              /Birinci karakterin alınması
        OUT              /Karakterin yazılması
        BSA  SH4         /4 basamak sola kaydırma
        BSA  SH4         /4 basamak daha sola kaydırma
SCD,    SKI
        BUN  SCD
        INP              /İkinci karakterin alınması
        OUT
        BUN  IN2 I       /Ana programa dönüş
```

Çizelge 6.21 Giriş karakterinin buffer a depolanması programı

```
        LDA ADS     /Buffer in ilk adresini AC ye yükle
        STA PTR     /AC yi PTR ye aktar
LOP,    BSA IN2     /IN2 alt programına git
        STA PTR     /PTR nin belirttiği yere iki karakteri yaz
        ISZ PTR     /Göstergeyi bir arttır
        BUN LOP     /Bir sonraki karakterleri almak için LOP
                     a dallan
        HLT
ADS,    HEX 500
PTR,    HEX 0
```

Çizelge 6.22 İki kelimenin karşılaştırılması programı

```
        LDA  WD1    /Birinci veriyi AC ye yükle
        CMA         /AC nin tümleyenini al
        INC         /AC yi 1 arttırır
        ADD  WD2    /İkinci veri ile topla
        SZA         /AC sıfır ise bir sonraki komuta atla (iki buyruk eşit ise)
        BUN  UEQ    /UEQ yordamına şartsız dallan
        BUN  EQL    /EQL yordamına şartsız dallan
WD1,    —
WD2,    —
```

Derleyici ikinci geçişte programdaki sembolün çizelgede olup olmadığına bakar. Bunu anlamak için iki sembol karşılaştırılır, eğer eşit çıkarsa derleyici sembolün karşılığı ikili değeri kullanır. Böyle bir program çizelge 6.22 dedir. Karşılaştırma sembolün ASCII kodunun 2 ye göre tümleyeni haline getirilmesi ve sembolle toplanması yoluyla yapılır. Bu program iki kelimeyi karşılaştırmakta ve esas karşılaştırma, programının alt programı olarak kullanılmaktadır.

Program Kesmeleri

Bilgisayarın çalışma zamanı, dış aletlerin bayrak kaldırmalarını beklemekle geçer. G/Ç programı durmadan dönerek bayrakları denetler. Çevre birim bayrak kaldırdığında bilgisayara haber verebilirse bilgisayar bu denetim zamanını başka işler yapmak için kullanabilir. Bu haber verme olayına kesme denir. Bilgisayarda tek bir program varsa yapacağı başka bir iş yoktur. Eğer birden fazla program varsa kesmeleri kullanır.
Eğer bilgisayarın belleğinde birden fazla program varsa, bilgisayar bir anda sadece bir programı çalıştırabilir. Bu programa çalışan program denir. Diğer programlar G/Ç için beklerler veya uyurlar. Kesme işlemi bir veya daha fazla programın G/Ç işlemlerini yapmasını ve orada işlemcinin başka işle uğraşabilmesini sağlar. Çalışan program kesme geldiğinde ION buyruğu çalıştırabilmelidir. Eğer kesme kullanılmayacaksa IOF yapılabilmelidir.

Kesme olayı programa G veya Ç aletlerinden biri kendi bayrağını 1 yapana kadar çalışmasını sağlar. Bilgisayar bir kesme gelince çalıştırmakta olduğu buyruğun çalışmasını tamamlar sonra kesme ile uğraşır. Dönüş adresini 0 adresine saklar, sonra 1 adresindeki buyruk çalıştırılır. 1 adresinden gerekli kesme yordamına dallanılabilir. Kesme yordamı belleğin herhangi bir yerinde olabilir. Kesme servis yordamı içinde şunlar bulunmalıdır.

1. İşlemci yazaçlarının içeriklerinin saklanması.
2. Hangi bayrağın kaldırıldığının bulunması.
3. Bayrağı kaldırılan aletin hizmet görmesi.
4. İşlemci yazaç içeriklerinin tekrar yerlerine yazılması.
5. Kesme olayının tekrar açılması.
6. Çalışan programa dönüş.

İşlemci yazacının içerikleri kesmeden önce ve sonra aynı olmalıdır. Aksi halde program hatalı çalışır. Servis yordamı bu yazaçları kullanacağından bu yazacın içeriklerinin bir yerde saklanması zorunludur. Bu iş yordamın sonuna kadar devam etmelidir. Yordamın sonunda yazaç içerikleri tekrar yerine yazılmalıdır. Her bir aletin bayrağı aletin önem sırasına göre kontrol edilir. Eğer iki veya daha fazla alet aynı anda bayrak

kaldırırsa yalnızca bir çevre birime hizmet edilebilir. Bu yüzden aletler öncelik sıralarına göre hizmet alırlar.

Bir kesmenin olması diğer kesmeleri pasif yapar (daha düşük önemli kesmeleri). Diğer kesmelerin çalışabilmesi için önce servis edilen kesmenin servisinin bitip kesmenin açılması gerekir. Kesmenin açılması için dönüş adresinin *PC* a yazılması zorunludur.

Bir kesme servis yordamı 6.23 te verilmiştir. 0 adresinde bulunmaktadır. Programda ION buyruğu vardır. Varsayılsın ki bilgisayar 103 adresindeki buyruğu icra ederken bir kesme oluşsun. Kesme 104 dönüş adre-

Çizelge 6.23 Bir kesme servis programı

Adres				
0	ZRO,	—		/Dönüş adresi buraya saklanıyor
1		BUN	SRV	/Servis yordamına şartsız dallanıyor
100		CLA		
101		ION		
102		LDA	X	
103		ADD	Y	/Kesme burada oluştu
104		STA	Z	/Kesmeden sonra program buraya döner
•		•		
•		•		
•				/Kesme servis yordamı
200	SRV,	STA	SAC	/*AC* nin içeriğini SAC ye sakla
		CIR		/*E* yi *AC*(1) e aktar
		STA	SE	/*E* nin içeriğini SE ye sakla
		SKI		/Giriş bayrağı 1 ise *PC* yi 1 arttırır
		BUN	NXT	/Şartsız olarak dallan
		INP		/Bayrak kalktığında giriş karakterini al
		OUT		/Karakteri yaz
		STA	PT1 I	/Girilen karakteri giriş buffer ına depola
		ISZ	PT1	/Giriş göstergesini 1 arttır
	NXT,	SKO		/Çıkış bayrağını kontrol et eğer on ise atla
		BUN	EXT	/Şartsız EXT e başlığına dallan
		LDA	PT2 I	/Karakteri çıkış buffer ına yükle
		OUT		/karakteri çıkış birimine gönder
		ISZ	PT2	/çıkış göstergesini arttır
	EXT,	LDA	SE	/SE yi yani *AC*(1) i *AC* ye yükle
		CIL		/sola kaydırma ile E ye eski değerini ver
		LDA	SAC	/SAC yi *AC* ye yükle
		ION		/Kesmeyi aç
		BUN	ZRO I	/Daha önce çalıştırılan programa geri dön
	SAC,	—		/*AC* burada saklandı
	SE,	—		/*E* burada saklandı
	PT1,	—		/Giriş buffer göstergesi
	PT2,	—		/çıkış buffer göstergesi

sini 0 adresine yazar ve 1 adresine gider. 1 adresindeki buyruk Bilgisayarı SRV yordamına gönderir. Servis yordamı yukarıda belirtilen 6 görevi yerine getirir. *AC* ve *E* nin içerikleri özel adreslere yazılır. (Bunlar temel bilgisayardaki işlemci yazaçlarıdır.) Bayraklara sırayla bakılır (giriş önce çıkış sonra). Bunlardan biri aktif yapılmışsa veri gerekli buffer a yazılır ve kesme açılır. Son buyruk 0 adresindeki adrese dallanarak geri dönüş adresini *PC* ye aktarır. Program kaldığı yerden çalışmaya devam eder.

Tipik bir bilgisayarda kesmeler ile bağlanmış çok sayıda giriş/çıkış aleti bulunabilir. Ayrıca kesmeler sadece G/Ç aktarımlarıyla sınırlı değildir. Bilgisayar içindeki hata durumlarında veya özel alarm koşullarında da kullanılabilir, programcılar da kullanır.

SORULAR

6-1. Aşağıdaki program temel bilgisayarın bellek biriminde saklıdır. Her buyruk çalıştıktan sonra onaltılık olarak *AC*, *PC* ve *IR* nin içeriğini gösteriniz. Aşağıda listelenen numaraların hepsi onaltılıktır.

Adres	Buyruk
010	CLA
011	ADD 016
012	BUN 014
013	HLT
014	AND 017
015	BUN 013
016	C1A5
017	93C6

6-2. Aşağıdaki program onaltılık kodlardan oluşan buyruk listesidir. Bilgisayar 100 adresinden itibaren buyrukları çalıştırmaya başlıyor. Programın çalışması bittikten sonra *AC* ye ve 103 adresindeki bellek kelimesinin içeriği ne olur.

Adres	Buyruk
100	5103
101	7200
102	7001
103	0000
104	7800
105	7020
106	C103

6-3. Değişkenleri tamsayı kabul edip aşağıdaki Fortran programının birleştirici dil karşılığını ve ikili değerlerini yazınız.

```
TOPLAM = 0
TOPLAM = TOPLAM + A + B
FARK = FARK - C
TOPLAM = TOPLAM + FARK
```

6-4. Birleştirici dil programında I harfi sembolik adresleme için kullanılabilir mi?

6-5. Bir ORG veya END sözde buyruğuna sahip olan kod satırının bir başlık alanı varsa birleştirici dilin ilk geçişi sırasında ne olur? Eğer bu gerçekleşirse hata mesajını belirtmek için akış şemasını çiziniz.

6-6. Birleştirici dildeki bir kod satırı

```
DEC -35
```

gibidir.

a. Kod satırını saklamak için 4 bellek kelimesinin gerekli olduğunu gösteriniz ve bunların ikili içeriğini yazınız.

b. Bir bellek kelimesinin ikiliye dönüştürülmüş kodu sakladığını gösteriniz ve ikili içeriğini yazınız.

6-7. **a.** Çizelge 6.13 teki programın ilk geçiş sırasında oluşan adres-sembol çizelgesini elde ediniz.

b. Programı onaltılık koda çeviriniz.

6-8. Sözde buyruk olan BSS N bazen bir gurup veriyi saklamak için N bellek kelimesi kullanır. Örneğin komut satırı

```
A,  BSS 10
```

olduğunda birleştirici dile A dan başlayarak 10 alandan oluşan bir blok bilgisi verir. Bu Fortrandaki DIMENSION A(10) ile aynıdır. Şekil 6.1 deki akış şemasını bu sözde buyruk için biraz değiştiriniz.

6-9. Şekil 6.2 deki akış şemasını bir sembolik adres bir etiketle tanımlanmadığı zaman bir hata mesajı verecek şekilde değiştiriniz.

6-10. MRI ve MRI olmayan buyrukların bellekte nasıl saklandığını gösteriniz.

6-11. Aşağıdaki koşul ifadesi için derleyici tarafından oluşturulan birleştirilmiş dil programını ikili karşılığı ile yazınız.

```
IF (A - B) 10, 20, 30
```

Eğer A – B sıfırdan küçük ise program 10 ifadesine, A – B sıfıra eşit ise 20 ifadesine A – B sıfırdan büyük ise 30 ifadesine dallanacak.

6-12. **a.** Aşağıdaki program çalışınca ne olacağını kelimelerle anlatınız. Programın çalışması bittiğinde CTR nin değeri nedir?

b. Birleştirici dilin ilk aşaması sırasında oluşturulan adres sembol tablosunu yazınız.

c. Programı onaltılık koda çeviriniz.

```
        ORG 100
        CLE
        CLA
        STA CTR
        LDA WRD
        SZA
        BUN ROT
        BUN STP
ROT,    CIL
        SZE
        BUN
        AGN
        BUN
        ROT
AGN,    CLE
        ISZ
        CTR
        SZA
        BUN
        ROT
STP,    HLT
CTR,    HEX 0
WRD,    HEX 62C1
        END
```

6-13. Bir gösterge ve sayaç kullanarak 500 ile 5FF arasındaki onaltılık adreslerin içeriklerini sıfırlayan bir program döngüsünü yazınız.

6-14. Tekrarlı toplama ile iki pozitif sayının çarpımını hesaplayan bir program yazınız. Örneğin 5 × 4 çarpımı 4 kez 5 in toplamına denktir veya 5 + 5 + 5 + 5 dir.

6-15. Çizelge 6.14 teki programa başlangıç değerleri atanmamıştır. Program bir kez çalıştıktan sonra CTR ye 0 yerleştirilir. Program tekrar çalıştırıldığında programın 100 den başlayacağını ve döngünün 65536 kez tekrarlayacağını gösteriniz. Başlangıç değer ataması için gerekli buyrukları programa ilave ediniz.

6-16. Her biri 16 anlamlı bit içeren bir işaretsiz çarpım verecek iki işaretsiz pozitif sayının çarpımını yapan programı yazınız.

6-17. İki işaretli sayının çarpımını yapan programı yazınız. Negatif sayılar 2 ye göre tümleyen şeklinde olsun. Sonuç tek duyarlıklı olacak ve eğer negatif çıkarsa sonucun 2 ye göre tümleyeni alınacak.

6-18. İki çift duyarlıklı sayının çıkarılması işlemini yapan programı yazınız.

6-19. İki mantıksal veriye ÖZEL-VEYA işlemini uygulayan bir program yazınız.

6-20. Aritmetik sola kaydırma için bir program yazınız. Bir taşma durumu oluştuğunda OVF ye dallansın.

6-21. İki sayının çıkarma işlemini yapan bir alt program yazınız. BSA buyruğunu çıkan ve çıkarılan sayılar takip etsin. BSA yı takip eden üçüncü satıra alt programdan gelen fark değeri yazılsın.

6-22. Bir data bloğunun her kelimesinin tümleyenini alan bir alt program yazın. Bu alt programı çağıran programda BSA komutunu takiben iki veri bulunmalıdır. Bunlar veri bloğunun başlangıç adresi ve bloktaki veri sayısıdır.

6-23. *AC* ve *E* yi 4 kez dairesel sağa kaydıran bir alt program yazınız. Eğer *AC* nin içeriği onaltılık 079C ve $E = 1$ ise bu alt program çalıştıktan sonra *AC* ve *E* nin içerikleri ne olur.

6-24. Giriş karakterini alıp 2 karakteri 1 kelimeye paketleyen ve bu kelimeleri ardışık bellek adreslerinde saklayan programı yazınız. Verilerin aktarılacağı bellek adresi $(400)_{16}$ olsun. Bellek uzunluğu $(512)_{10}$ olsun. Eğer bellek dolarsa program dursun.

6-25. WRD adresindeki iki karakter içeren bir kelimeyi açarak CH1 ve CH2 nin 0-7 karakterlerine aktarınız. 8-15 karakterleri 0 olsun.

6-26. Bellek bufferinde bir CR(onaltılık 0D) kodunu kontrol eden bir programın akış şemasını çiziniz. Bufferin her kelimesinde 2 karakter bulunsun. Bir CR kod ile karşılaşıldığında LNE adresindeki 8-15 bitlerini dağıtmaksızın 0-7 bitlerine yazsın.

6-27. Çizelge 6.23 teki SRV servis alt programını onaltılık koda dönüştürünüz. Alt programın 200 adresinden başlayarak bellekte bulunduğunu varsayınız.

6-28. Bütün fonksiyonları yerine getirecek fakat giriş cihazı yalnız özel bir adres, MOD, tüm 1 leri içerir ise çalışacak bir kesme alt programı yazınız. Çıkış cihazı yalnız adres, MOD, tüm 0 ları içerir ise çalışsın.

KAYNAKLAR

1. Booth, T. L., *Introduction to Computer Engineering*. 3rd ed. New York: John Wiley , 1984.
2. Gear, C. W., *Computer Organization and Programming*, 3rd ed. New York: McGraw-Hill, 1980.
3. Gibson,G. A., *Computer Systems Consepts and Desing*. Englewood Cliffs, NJ: Prentice Hall, 1991.
4. Gray, N. A. B., *Introduction to Computer Systems*. Englewood Cliffs, NJ: Prentice Hall, 1987.
5. Levy, H. M., and R. H. Eckhouse , Jr., *Computer Programming and Architecture*: The VAX-11. Bedford, MA: Digital Press, 1980.

6. Lewin, M. H., *Logic Desing and Computer Organization*. Reading, MA: Addison-Wesley, 1983.
7. Prosser, F. P., and D. E. Winkel, *The Art of Digital Desing*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1987.
8. Shiva, S. G., *Computer Desing and Architecture*. 2nd ed. New York: HapperCollins Publishers, 1991.
9. Tanenbaum, A. S., *Structured Computer Organization*, 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1990.
10. Wakerly, J. F., *Microcomputer Architecture and Programming*. New York: John Wiley, 1981.

BÖLÜM 7

Mikro Programlanmış Denetim

BU BÖLÜMDE

7-1 Denetim Belleği
7-2 Adres Sıralama
7-3 Mikro Program Örneği
7-4 Denetim Birimi Tasarımı

7.1 Denetim Belleği

Bir sayısal bilgisayardaki denetim biriminin görevi mikro işlemlerin sırasını başlatmaktır. Verilen bir sistem içinde bulunan farklı tipteki mikro işlemlerin sayısı sınırlıdır. Sayısal sistemin karmaşıklığı, çalıştırılan mikro işlem sıralarının sayısından kaynaklanır. Denetim sinyalleri donanım tarafından kararlaştırılmış mantık tasarım teknikleri kullanılarak oluşturulduğunda, denetim birimi donatımsal (tellerle bağlanmış donanım) olarak söylenir. Mikro programlama sayısal bilgisayarların tasarımı için ikinci bir yoldur. Mikro programlamanın kuralı sayısal bilgisayarda mikro işlem sıralarının denetimi için üstün ve sistematik bir yöntem bulmaktır.

Denetim fonksiyonu ikili bir değişkendir ve mikro işlemleri belirler. Denetim fonksiyonu ikili bir ifade belirttiğinde uygun gelen mikro işlem işletilir. Denetim fonksiyonunda bulunmayan bir denetim değişkeni olursa sistemdeki yazaçlarda bir değişiklik olmaz. Denetim değişkeninin geçerli durumu uygulamaya bağlı olarak 1 ve 0 durumunun her ikisi olabilir. Organize edilmiş veri yolu sisteminde mikro işlemleri belirleyen denetim sinyalleri seçiciler, kod çözücüler ve aritmetik mantık birimdeki yolları seçen bitlerin bir araya gelmesiyle oluşur.

Denetim birim mikro işlemlerin ardışık basamaklarının bir dizisini başlatır. Herhangi bir zamanda verilen denetim değişkeni 1 ve 0 lardan

Denetim Kelimesi, Mikro Buyruk, Mikro Program

oluşan bir dizidir ve denetim kelimesi olarak adlandırılır. Denetim kelimesi sistemin parçaları üzerinde çeşitli işlemleri yerine getirebilecek şekilde programlanabilir. Bir denetim biriminin ikili denetim değişkenleri bellekte saklanır, mikro programlanmış denetim birimi olarak adlandırılır. Denetim belleğindeki her kelime bir mikro buyruk içerir. Mikro buyruk bir veya daha fazla mikro işlemi sistem için belirler. Mikro buyrukların sırası bir mikro programı belirler. Denetim birim işlemdeyken mikro programda değişiklikler yapılamaz. Denetim belleği ROM olabilir. ROM içindeki kelimenin içeriği sabittir ve basit programlama ile değiştirilemez. ROM kelimeleri donanım ile birlikte yüklenir.

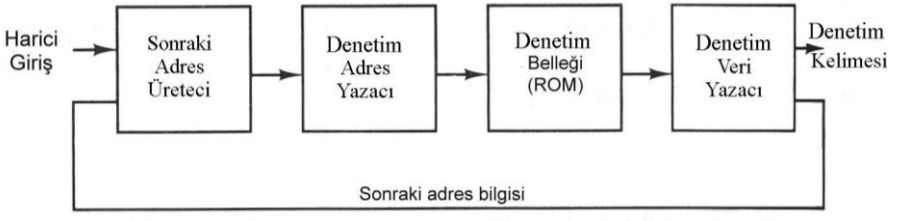
Bir mikro programın kullanımında, ardışık okuma işlemleri boyunca, denetim birim tarafından kullanılması için tüm denetim değişkenleri ROM kelimelerine yerleştirilir. Belirtilen bir adresteki ROM kelimesinin içeriği bir mikro buyruğu belirler. Daha ileri bir gelişme de dinamik mikro programlama olarak bilinir. Manyetik disk gibi bir yardımcı bellekten ilk olarak yüklenen bir mikro programa izin verir. Denetim biriminin bir parçası olan bellek, denetim belleği olarak anlatılır.

Denetim belleği

Mikro programlanmış denetim birimi içeren bilgisayarın iki ayrı belleği vardır, ana bellek ve denetim belleği. Ana bellek kullanıcılarının programlarını depolaması için kullanılır. Ana belleğin içeriği veri işlendiğinde ve program değiştirildiğinde değişebilir. Ana bellekteki kullanıcıların programları makine buyrukları ve verileri içerir. Ters olarak denetim bellek kullanıcının istediği zaman değiştirmediği sabit bir mikro programı elinde tutar. Mikro buyrukları içeren mikro program, yazaç mikro işlemlerinin icrası için çeşitli bellek ana denetim sinyallerini belirler. Her bir makine buyruğu denetim belleğinde bir dizi mikro buyrukları başlatır. Bu mikro buyruklar buyruğun bellekten alınıp getirilmesi mikro işlemlerini oluşturur; etkin adresi hesaplamak, buyruk tarafından belirlenen işlemi yürütmek ve bir sonraki buyruğun al-getir evresini yapmak için denetime geri dönmek gibi.

Denetim Adres Yazacı

Mikro programlanmış denetim biriminin genel görünümü şekil 7.1 deki blok şemada gösterilmiştir. Denetim belleği ROM olarak düşünülür.



Şekil 7.1 Mikro programlanmış denetim organizasyonu

Denetim bilgileri denetim belleğinin içinde sürekli saklanır. Denetim adres yazacı, mikro buyrukların adresini belirler ve denetim veri yazacı bellekten okunan mikro buyrukları tutar. Denetim kelimesi veri işlemcisi için bir veya daha fazla mikro işlem belirleyen mikro buyruğu içerir. Mevcut işlemler icra edilir, denetim sonraki adresi belirlemelidir. Bir sonraki mikro buyruğun yeri sıra içinde bir sonraki olabilir, veya denetim bellekte herhangi bir yere yerleşebilir. Bu durumda o anki mikro buyruğun bazı bitleri ile bir sonraki mikro buyruğun adresi, sonraki adresi hesaplayan devrede hesaplanır ve sonraki adresi okumak için denetim bellek adres yazacına gönderilir. Böyle bir mikro buyruk denetim belleği için adres sırasını belirler ve veri işlemcisi içinde mikro işlemlerin başlatılması için bitleri içerir.

Sıralayıcı

Sonraki adres üreteci çoğunlukla sıralayıcı mikro programı olarak adlandırılır, denetim belleğinden okunacak adres sırasını belirler. Sonraki mikro buyruğun adresi sıralayıcının girdilerine bağlı olarak birkaç yolla belirlenebilir. Sıralayıcı bir mikro programın tipik işlevi denetim adres yazacını 1 ile arttırma, denetim adres yazacına denetim belleğinden bir adres yükleme işlemini yapar.

Boru Hattı Yazacı

Sonraki adres hesaplanıp bellekten okunur iken denetim veri yazacı mevcut mikro buyruğu tutar. Bu veri yazacı bazen *boru hattı yazacı* olarak ta isimlendirilir. Veri yazacı denetim kelimesi tarafından belirlenen mikro işlemlerin icrası ile birlikte sonraki mikro buyruğun oluşturulmasını da sağlar.

Denetim kelimeleri ve sonraki adres bilgileri doğrudan denetim belleğinden alınır. Bilinmesi gereken bir gerçekte, bir ROM çıkış olarak ve giriş olarak adres değeri ile bir birleşik devre olarak işleme girer. RAM daki gibi bir okuma sinyaline ihtiyaç duyulmaz. Her bir saat vuruşunda denetim kelimesi tarafından yürütülür ve yeni adres denetim adres yazacına yerleştirilir.

Donanımsal Denetim

Mikro programlanmış denetimin temel kazancı, donatımsal düzenin kurulmasıdır. Daha ileri bir donanım ve veri yolu değişikliğine ihtiyacı yoktur. Eğer sistem için farklı bir denetim sırası yapmak istenirse, yapılması gereken denetim belleği için farklı bir mikro buyruk kümesi belirlenmelidir, farklı işlemler için donanımda bir değişiklik yapılmamalıdır, sadece denetim belleğinde mikro programın yerleşimi değiştirilmelidir.

7-2 Adres Sıralama

Yordam

Mikro buyruklar, her gurubun bir yordam belirttiği guruplardaki denetim belleklerinde saklanır. Her bilgisayar buyruğu kendi denetim belleğinde kendi mikro programına sahiptir. Bu mikro program buyruğu çözen mik-

ro işlemler üretir. Denetim hafızasının adres sıralamasını kontrol eden donanım bir yordam ile mikro buyrukların sıralamasını ve bir yordamdan diğer yordama dallanabilmesini yapmak zorundadır. Bir mikro program denetim biriminde sıralama adresini değerlendirmek için adımların sayılması gerekir ki denetim bilgisayar buyruğunun işlenmesi sırasında gerçeklenebilsin.

Bilgisayar açıldığında ilk adres denetim adres yazacına yüklenir. Bu adres genelde ilk mikro buyruğun adresidir ve buyruk al-getir yordamını içerir. Al-getir yordamı denetim adres yazacının tüm mikro buyruklarını arttırma ile sıralanabilir. Al-getir yordamının sonunda buyruk bilgisayarın buyruk yazacının içindedir.

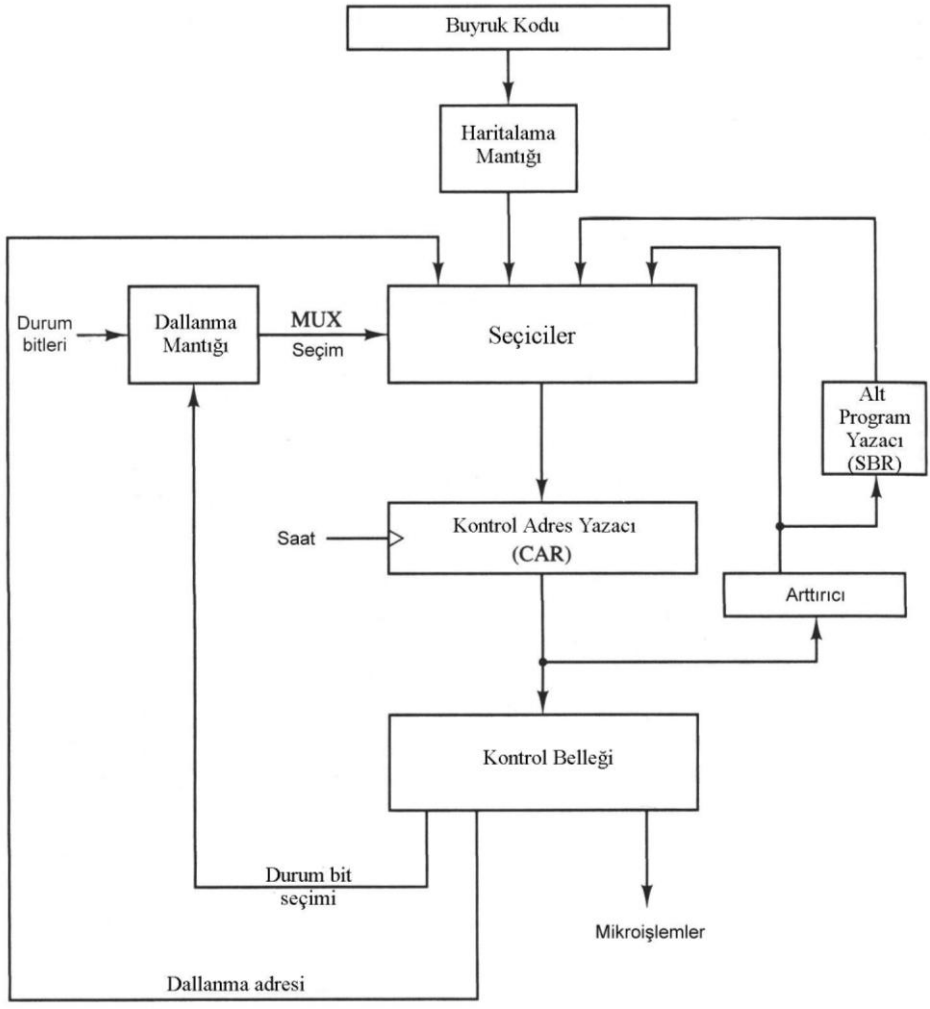
Daha sonra denetim belleği yordamı inceler ve böylelikle işlenenin etkin adresi elde edilir. Bir makine buyruğu çeşitli adres kiplerini belirleyen bitlere sahiptir. Örnek olarak dolaylı adres ve indis saklayıcılar gibi. Denetim belleğindeki etkin adres hesaplama yordamına, bir dallanma mikro buyruğu ile ulaşılır. Efektif adres hesaplama yordamı bittiğinde işlenenin adresi bellek adres yazacında bulunmaktadır.

Haritalama

Denetim belleğinde etkin adres hesaplama programı buyruğun kip bitlerinin durumu üzerinde koşullandırılan dallanma programı tamamlandığından, işlenenin adresi bellek adres yazacında bulunur. Bir sonraki adım bellekten getirilen buyruğu, işlem kodu kısmına bağlı olarak işlemci yazacında oluşturmalıdır. Her bir buyruğun kendi mikro program yordamı vardır ve bu program denetim belleğinde verilen bir adrese yerleştirilir. Denetim belleğine yerleştirilmiş olan rutin buyruk kod bitlerinin bir adres dönüşümü ile bulunması *haritalama* olarak adlandırılır. Haritalama işlemi buyruk kodunu, denetim bellek adresine dönüştüren bir kuraldır. İhtiyaç duyulan programa erişilmelidir. Buyrukları icra eden mikro buyruklar denetim adres yazacının arttırılması ile sıralanır. Fakat bazen mikro işlemlerin sırası işlemci yazacındaki belirlenmiş durum bitlerinin değerlerine bağlıdır. Alt program içeren mikro programlar dönüş adresini saklamak için harici bir yazaca ihtiyaç duyarlar. Dönüş adresi ROM a yazılamaz.

Buyruğun icrası tamamlandığında denetim al-getir yordamına dönmelidir.Bu al-getir yordamının ilk adresine şartsız olarak dallanma ile sağlanır. Özetle denetim belleğinde adres sıralama işlemleri

1. Denetim adres yazacında artma.
2. Durum bitlerinin kuralına bağlı olarak şartlı veya şartsız dallanma.
3. Buyruk bitlerinden bir adrese denetim belleği için haritalama işlemi.
4. Alt programa giriş ve çıkış için kolaylık.



Şekil 7.2 Denetim belleği için adresin seçimi

Şekil.7.2 sonraki mikro buyruğun adresinin seçimi için ihtiyaç duyulan denetim belleği ve ilişikli donanımın blok şemasını gösteriyor. Denetim belleğindeki mikro buyruk bilgisayar yazaçlarında mikro işlemleri başlatan bit grubunu ve sonraki adresi belirleyen bitleri içerir. Şema, adresleri alan denetim adres yazacında bulunan 4 farklı yolu gösterir. Arttırıcı CAR ın içeriğini bir arttırır. Bir sonraki mikro buyruk sırasını seçer. Dallanma mikro buyruk alanının birinden belirlenen dallanma adresi sağlanır.

Şartlı dallanma ise mikro buyruk içindeki özel durum bitlerinden şartın elde edilmesi ile mümkün olur. Dış adres, denetim belleğine haritala-

ma mantık devresi yardımıyla aktarılır. Alt programın dönüş adresi özel bir yazaca yazılır. Mikro program alt programdan dönmek istediğinde bu yazacı kullanır.

Şartlı Dallanma

Özel Bitler

Şekil 7.2 deki dallanma mantığı denetim birimindeki karar verme yeteneğini gösterir. Sistemdeki durum şartları özel bitlerle belli olur. Bu bitler parametre gibi görev yaparlar. Örneğin toplayıcının çıkış elde si, sayının işaret biti, buyruğun kip biti ve G/Ç durum şartları gibi. Bu bitler içindeki bilgiler test edilebilir ve ilgili işlem başlatılabilir. Bunun için bitlerin 1 veya 0 olduklarına bakılır. Bu bitlerle mikro buyruk içinde olan dallanma adresini oluştururlar. Şartlı dallanma denetimi dallanma mantığı içinde verilen kararla yapılır.

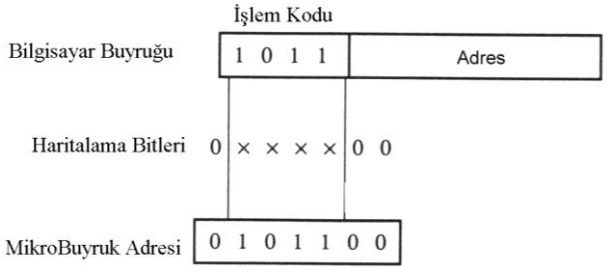
Dallanma Mantığı

Dallanma mantığı donanımı muhtelif yollarla elde edilebilir. En basit yol, verilen koşulun testi ve eğer koşul sağlanıyorsa gösterilen adrese gitme, aksi takdirde adres yazacının arttırılmasıdır. Bu bir MUX ile sağlanabilir. Sistemde 8 tane durum biti bulunduğunu varsayalım. Mikro buyruk içindeki 3 bit, 8 tane durum bit şartından herhangi birini belirlemek için kullanılır. Bu 3 bit MUX için seçim değişkenlerini içerir. Eğer seçilen durum bitleri 1 ise MUX un çıkışı 1, aksi halde 0 çıkışı adres yazacının arttırılmasına neden olur. Bu biçimde mikro program iki mümkün yoldan birini, seçilen durum bitine bağlı olarak takip eder.

Şartsız dallanma mikro buyruğu, dallanma adresinin denetim belleğinden denetim adres yazacına aktarılmasıyla sağlanır. Bunun için, MUX girişindeki durum bitlerinden biri sabit hale getirilir, bu bit daima 1 yapılır. Bit seçim telinin bu durum bitini bulmasıyla dallanma adresi şartsız olarak denetim adresi yazacına yüklenir.

Buyruğun Haritalanması

Bir mikro buyruk denetim belleğinin ilk kelimesine dallanmak istediğinde özel bir dallanma vardır. Denetim belleğinin ilk kelimesinde bir buyruk için bir mikro program yordamı mevcuttur. Bu tip dallanma buyruğun durum bitleri, buyruğun işlem kod bitleridir. Örneğin şekil 7.3 teki basit buyruk biçimi olan bir bilgisayarda, bir buyruğun 4 bit işlem kodu vardır. 4 bit 16 farklı buyruk belirler. Bundan başka denetim belleğinin 128 kelime olduğu düşünülsün. Adresleme için 7 bite gerek vardır. Denetim belleğinde mikro program yordamı olan her işlem, işlem kodu buyruğu icra edebilir. 4 bit işlem kodunu, denetim belleği için 7 bit adrese dönüştüren basit bir haritalama işlemi şekil 7.3 te gösterilmiştir. Bu haritalamada adresin en önemli biti 0 dır. 4 işlem kod biti olduğu gibi



Şekil 7.3 Buyruğun haritalanması

aktarılır. Denetim adres yazacındaki en önemsiz 2 bit sıfırlanır. Bu her bilgisayar buyruğu için bir mikro program yordamı sağlar. Her bir rutinde 4 mikro buyruk vardır. Eğer rutin 4 mikro buyruktan fazlasına ihtiyaç duyarsa 1000000 dan 1111111'e kadar olan adresleri kullanabilir. Eğer 4 mikro buyruktan daha azını kullanırsa kullanılmayan bellek adresleri diğer rutinlere açıktır.

Bu kavram çok daha genel bir haritalama kuralına uzatılabilir ve ROM haritalama fonksiyonu için kullanılabilir. Böyle bir haritalamada buyruğun bitleri bu iş için kullanılabilir. ROM un adresini belirtir. Haritalama yapan ROM un içeriği, denetim adres yazacı için bitler verir. Bu yolla buyruğu icra eden mikro program yordamı, denetim belleğinin herhangi bir yerine yerleştirilebilir. Haritalama kavramı gerektiğinde denetim belleğine buyrukların eklenebilmesi için esneklik sağlar.

Haritalama fonksiyonu bazen PLD (programlanabilir mantık aleti) adı verilen bir IC ile elde edilir. PLD, kavram olarak ROM a benzer iç elektronik sigortalarla birlikte VE, ve VEYA kapıları kullanılır. Girişler Arasındaki bağlantı VE ve VEYA kapılarıyla çıkışlar ise ROM ile programlanır. Boole bağıntılarıyla belirlenen bir haritalama fonksiyonu, PLD ile uygun biçimde kurulabilir.

Alt Programlar

Alt programlar diğer rutinler tarafından özel bir işlemi yapmak için kullanılan programlardır. Bir alt program, mikro programın ana kısmının herhangi bir yerinden çağrılabilir. Bir çok mikro program birbirinin aynı kod parçaları içerirler. Alt programlar mikro buyrukların hepsinde ortak olarak kullanılır. Tüm bellek adreslemeli buyruklarda yaygın alarak kullanılan, buyrukların verilerinin etkin adreslerinin üretimi için bir dizi mikro işleme ihtiyaç vardır. Bu dizi bir alt program olur ve birçok rutin tarafından etkin adres için çağrılabilir.

Alt Program Yazacı

Alt program kullanan mikro programlar, alt program çağrılması sırasında dönüş adresini saklayacak ve alt programdan dönüş sırasında bu

adresi tekrar alabilecek yetenekleri olmalıdır. Bunun elde edilmesi için, denetim adres yazacından çıkan arttırılmış çıktının, alt program yazacı, ana programa dönüş için kaynak olur ve buradaki adres kullanılır. Bir yazaç dosyası oluşturmak için en iyi yol yazaçlar, LIFO yığıt şeklinde düzenlenmektedir. Bu dosyada alt programlar için adresleri tutar. Yığıt bölüm 8.7 de anlatılacak.

7.3 Mikro Program Örneği

Bir bilgisayarın konfigürasyonu ve mikro programlanmış denetim birimi kurulduktan sonra, tasarımcının görevi denetim belleği için mikro kodları yaratmaktadır. Bu kod yaratmaya mikro programlama denir. Bu işlem bilinen birleştirici dil programlamanın benzerdir. Bunu anlamak için burada basit bir bilgisayar alacak ve bunun nasıl mikro programlanacağını göreceğiz. Buradaki bilgisayar bölüm 5 te tanımlanan bilgisayara benzer fakat özdeş değildir.

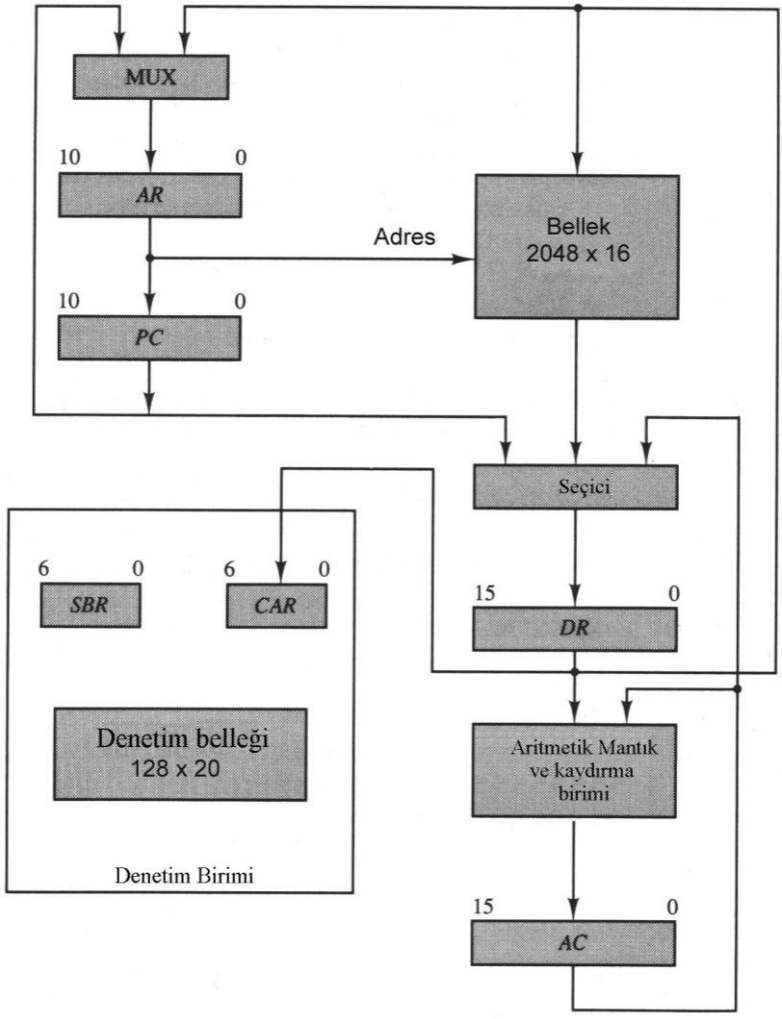
Bilgisayar Konfigürasyonu

Bilgisayarın blok şeması şekil 7.4 te gösterilmiştir. Bunun iki bellek birimi vardır. Ana bellek veri ve buyrukları saklamak içindir. Denetim belleği ise mikro programları saklamaya yarar. İşlemci birimine bağlı 4 yazaç vardır. Denetim birimine bağlı 2 yazaç bulunur. İşlemci yazaçları, *PC*, *AR*, *DR* ve *AC* yazaçlarıdır. Bu yazaçların fonksiyonları, bölüm 5 teki temel bilgisayardakinin aynıdır. Denetim biriminin bir *CAR* (denetim adres yazacı) ve bir (alt program yazacı) *SBR* vardır. Denetim belleği ve bunun yazaçları, mikro programlanmış denetim birimi olarak organize edilmiştir. (şekil 7.2).

Yazaçlar arasında bilgi aktarımı ortak veri yolundan çok MUX larla yapılır. *DR*, *AC* den, *PC* den veya bellekten bilgi alabilir. *PC* sadece *AR* den bilgi alabilir. ALU birimi veya *DR* den aldığı bilgilerle mikro işlemleri yapar ve sonucu *AC* ye yazar. Bellek adresleri *AR* den alır. Giriş dataları *DR* ye gelir ve oradan belleğe yazılır. Bellekten okunan datalar sadece *DR* ye gidebilir.

Buyruk Biçimi

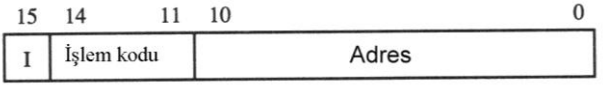
Bu bilgisayarın buyruk biçimi şekil 7.5(a) da verilmiştir. 3 alanı vardır. 1 bitlik dolaylı adres alanı I ile gösterilir. 4 bit işlem kısmıdır. 11 bit adres alanıdır. Şekil 7.5(b) de 16 mümkün bellek adreslemeli buyruğun 4 tanesi vardır. ADD buyruğu etkin adresteki işleneni *AC* nin içeriğine toplar. BRANCH buyruğu eğer *AC* deki veri negatif ise etkin adrese dal-



Şekil 7.4 Bilgisayar donanım düzeni

lanma sağlar. Eğer *AC* negatif değilse program bir sonra gelen buyruk ile devam eder. *AC* nin işaret biti *AC* negatifse 1 olur. STORE buyruğu *AC* nin içeriğini verilen etkin adrese yazar. EXCHANGE buyruğu etkin adresteki bilgi ile *AC* nin içeriğini değiştirir.

Daha sonra gösterileceği gibi buyrukların hepsi mikro programlanmalıdır. Mikro programlanma örneğinin karışık olmaması için sadece 4 buyruk alındı. 12 buyruk daha eklenebilir. Bunlar da aşağıdaki yöntemle programlanabilmelidir.



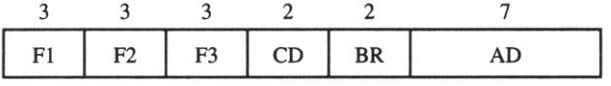
(a) Buyruk biçimi

Sembol	İşlem kodu	Tanımlama
ADD	0000	$AC \leftarrow AC + M\,[EA]$
BRANCH	0001	$(AC < 0)$ $(PC \leftarrow EA)$
STORE	0010	$M\,[EA] \leftarrow AC$
EXCHANGE	0011	$AC \leftarrow M[EA], M[EA] \leftarrow AC$

EA Efektif Adrestir

(b) Dört bilgisayar buyruğu

Şekil 7.5 Bilgisayar buyruk biçimi



F1,F2,F3 : Mikroişlem alanları
CD : Dallanma için şart
BR : Dallanma alanı
AD : Adres Alanı

Şekil 7.6 Mikro buyruk kod biçimi (20 bit).

Mikro Buyruk Biçimi

Mikro Buyruk Biçimi

Denetim belleği için mikro buyruk biçimi şekil 7.6 da gösterilmiştir. 20 bitlik biçim 4 fonksiyonel parçaya ayrılır. F1, F2 ve F3 alanları bilgisayar için mikro işlemleri belirler. CD alanı durum şartlarını gösteren bitlerdir. BR alanı kullanılacak dallanmanın tipini belirtir. Adres alanı 7 bit uzunluktadır. Çünkü denetim belleği $128 = 2^7$ ($n = 7$) kelimedir.

Mikro İşlemler

Mikro işlemler her biri 3 bitlik 3 alana bölünür. Her bir alandaki 3 bit 7 ayrı mikro işlem tanımlar. Çizelge 7.1 toplam 21 mikro işlem eder. Bir mikro buyruk için 3 ten fazla mikro işlem seçilemez. Yani her alandan 1

adet eğer alanlardan biri kullanılamıyorsa, yani mikro işlem sayısı 3 ten az ise, işlemsiz alanlara 000 yazılır. Örnek olarak iki eş zamanlı mikro işlem F2 ve F3 ve boş olan F1 olsun.

$$DR \leftarrow M[AR] \qquad F2 = 100$$

$$PC \leftarrow PC + 1 \qquad F3 = 101$$

Dolayısıyla mikro işlem alanları 000 100 101 olur. Aynı anda birbirleri ile çelişkili olan mikro işlemler yazılamaz. Örneğin 010 001 000 buyruğunun anlamı yoktur. Çünkü bunlar *AC* yi temizle ve *DR* yi aynı zamanda *AC* ile topla anlamındadır.

Çizelge 7.1 deki mikro işlemlerin her biri yazaç aktarım diliyle yazılmış ve sembolik mikro programlamada kullanılması için bir sembol verilmiştir. Aktarma tipi mikro işlemler 5 harfle verilir. İlk iki harf kaynak yazacını belirtir. Üçüncü harf daima *T* dir. Son iki harf ise hedeflenen yazacı gösterir. $AC \leftarrow DR$ (F1 = 100) mikro işlemi DRTAC sembolüyle verilir.

Şart Alanı

CD şart alanı iki bitten oluşur. Çizelge 7.1 deki 4 durum şartı belirtir. İlk şart daima 1 olduğundan (U sembolü için) CD = 00 şartı daima doğru olarak bulunur. Şart BR ile beraber kullanıldığında (dallanma alanı ile) şartsız dallanma işlemi oluşur. Dolaylı biti buyruk bellekten okunmasından sonra *DR* nin 15 inci biti olarak belli olur. *AC* nin işaret biti bir sonraki durum bitini belirler. Z ile gösterilen sıfır değeri, bir ikili değişkendir. *AC* nin bitleri 0 ise değeri 1 dir. 4 durum bitini U, I, S ve Z ile gösterelim. Mikro programlar sembolik gösterimle yazılır.

Dallanma Alanı

Dallanma (BR) alanı iki bitten oluşur. Adres alanı ile beraber kullanılır ve bir sonraki mikro buyruğun adresini seçmekte yararlanılır. Çizelge 7.1 de görüldüğü gibi BR = 00 iken denetim bir atlama (JMP) işlemi yapar. Bu dallanmaya benzerdir. BR = 01 iken bir alt program çağrılır. Bu işlem normal alt program çağırmanın aynıdır, sadece dönüş adresi *SBR* (alt program yazacı) ye konur. JMP ve CALL (alt program çağırma) işlemleri CD alanının değerine bağlıdır. Eğer durum bit şartı (CD alanı) 1 ise, AD alanındaki bir sonraki adres alanı denetim adres yazacı (*CAR*) a aktarılır. Aksi halde *CAR* 1 arttırılır.

Alt programdan dönüş BR = 10 olmasıyla sağlanır. Bununla *SBR* deki dönüş adresi *CAR* a aktarılır. Buyruğun işlem kod bitlerinden, *CAR* da bir adrese dönüşüm, BR alanı 11 olunca gerçekleşir. Bu haritalama şekil 7.3 de gösterilmiştir. İşlem kodunun bitleri bir buyruğun bellekten okunmasından sonra *DR*(11-14) ile bulunurlar. BR alanının son iki şartı AD ve CD alanlarının değerlerinden bağımsızdır.

Çizelge 7.1 Mikro buyruk olanları için sembol ve ikili kodları

	F1	Mikro işlem	Sembol
	000	None	NOP
	001	$AC \leftarrow AC + DR$	ADD
	010	$AC \leftarrow 0$	CLRAC
	011	$AC \leftarrow AC + 1$	INCAC
	100	$AC \leftarrow DR$	DRTAC
	101	$AR \leftarrow DR(0\text{-}10)$	DRTAR
	110	$AR \leftarrow PC$	PCTAR
	111	$M[AR] \leftarrow DR$	WRITE
	F2	Mikro işlem	Sembol
	000	None	NOP
	001	$AC \leftarrow AC - DR$	SUB
	010	$AC \leftarrow AC \vee DR$	OR
	011	$AC \leftarrow AC \wedge DR$	AND
	100	$DR \leftarrow M[AR]$	READ
	101	$DR \leftarrow AC$	ACTDR
	110	$DR \leftarrow DR + 1$	INCDR
	111	$DR(0\text{-}10) \leftarrow PC$	PCTDR
	F3	Mikro işlem	Sembol
	000	None	NOP
	001	$AC \leftarrow AC \oplus DR$	XOR
	010	$AC \leftarrow \overline{AC}$	COM
	011	$AC \leftarrow$ Shl AC	SHL
	100	$AC \leftarrow$ Shr AC	SHR
	101	$PC \leftarrow PC + 1$	INCPC
	110	$PC \leftarrow AR$	ARTPC
	111	Ayrılmış	

CD	Şart	Sembol	Yorumu
00	her zaman = 1	U	şartsız dallanma
01	$DR(15)$	I	dolaylı adres bit
10	$AC(15)$	S	AC nin işaret biti
11	$AC = 0$	Z	AC nin değeri sıfır

BR	Sembol	Fonksiyon
00	JMP	$CAR \leftarrow AD$ eğer şart = 1 ise $CAR \leftarrow CAR + 1$ eğer şart = 0 ise
01	CALL	$CAR \leftarrow AD$, $SBR \leftarrow CAR + 1$ eğer şart = 1 ise $CAR \leftarrow CAR + 1$ eğer şart = 0 ise
10	RET	$CAR \leftarrow SBR$ (Alt programdan geri dönüş)
11	MAP	$CAR(2\text{-}5) \leftarrow DR(11\text{-}14)$, $CAR(0,1,6) \leftarrow 0$

Sembolik Mikro Buyruk

Bir sembolik mikro program derleyici vasıtası ile ikili karşılığına dönüştürülür. Bir mikro program için birleştirici dil tanımlamanın en kestirme yolu, mikro buyruğun her alanı için semboller tanımlamak ve kullanıcılara kendi adreslerini sembollerle belirtme yeteneğini vermektir.

Birleştirici dilindeki bir mikro programın her satırı sembolik bir mikro buyruktur. Bu mikro buyrukların her biri beş alana ayrılır. Başlık, mikro işlemler, CD, BR ve AD. Bu alanlar aşağıda belirtilmiştir.

1. Başlık alanı boş olabilir veya bir sembolik adres verebilir. Başlık iki nokta (:) karakteri ile sona erer.
2. Mikro işlem alanında virgül (,) ile ayrılmış bir, iki veya 3 sembol bulunabilir. Bu semboller çizelge 7.1 de tanımlanmıştır. NOP sembolü mikro işlem olmadığında kullanılır. NOP, derleyici tarafından 9 tane 0 a dönüştürülür.
3. CD alanında U, I, S veya Z harflerinden biri bulunur.
4. BR alanında çizelge 7.1 de gösterilen 4 sembolden biri bulunur.
5. AD alanı adres için ayrılmış olup 3 farklı biçim olabilir.
   a. Sembolik adres aynı zamanda bir başlık olarak ta yer almalıdır.
   b. NEXT sembolü sıradaki bir sonraki adresi gösterir.
   c. BR alanında RET veya MAP sembolü varsa AD alanı boştur ve derleyici tarafından 7 tane 0 olarak aktarılır.

ORG ORG sözde buyruğu başlangıç veya ilk adresi belirtmek için kullanılmakta. Örneğin ORG 64, ikili 1000000'a denktir.

Al-getir Yordamı

Denetim belleği 128 kelime ve her kelime 20 bittir. Denetim belleğini mikro programlamak için 128 kelimenin bitlerinin tek tek belirlenmesi gerekir. 0-63 adresleri arasındaki 64 kelime 16 buyruğa ayrılmıştır. Son 64 kelime başka amaçlarla kullanılabilir. Al-getir alt yordamı için başlangıç adresi 64 olabilir. Al-getir yordamı için gerekli mikro buyruklar

$$AR \leftarrow PC$$
$$DR \leftarrow M[AR], \quad PC \leftarrow PC + 1$$
$$AR \leftarrow DR(0\text{-}10), \quad CAR\ (2\text{-}5) \leftarrow DR(11\text{-}14), \quad CAR\ (0,1,6) \leftarrow 0$$

Buyruğun adresi *PC* dan *AR* ye aktarılır ve buyruk bellekten *DR* ye okunur. Buyruk yazacı olmadığından, buyruk kodu *DR* de kalır. Adres parçası *AR* ye aktarılır ve denetim 16 yordamdan birine geçer. Bunun için işlem kodu *DR* dan alınır ve haritalanarak *CAR* a gönderilir.

Al-getir ve Kodu-çöz

Al-getir yordamında 3 mikro buyruk olacaktır. Bunlar 64, 65 ve 66 adreslerinde bulunacaklar. Daha önce tanımlanan birleştirici dil esasları ve tanımları ile al-getir evresi için mikro program

```
        ORG 64
FETCH:  PCTAR           U    JMP    NEXT
        READ, INCPC     U    JMP    NEXT
        DRTAR           U    MAP
```

Sembolik programın ikili ye aktarılması aşağıdaki ikili programı verir. Bit değerleri çizelge 7.1 den alınmıştır.

İkili adres	F1	F2	F3	CD	BR	AD
1000000	110	000	000	00	00	1000001
1000001	000	100	101	00	00	1000010
1000010	101	000	000	00	11	0000000

gösterir. Yazaç aktarımı, her bir mikro buyruğun icra edeceği aktarımları Sembolik yazılım birleştirici dil program yazmakta kolaylık sağlar. İkili şekil ise denetim belleğine konan gerçek biçimdir. Biz programlarımız sembolik biçimde yazıp bir derleyici vasıtası ile ikiliye dönüştürürüz.

Sembolik Mikro Program

Üçüncü al-getir mikro buyruğu (MAP) bir dallanma adresi 0xxxx00 verir. xxxx işlem kısmının 4 bitidir. Örneğin eğer buyruk bir topla buyruğu ise bunun işlem kodu 0000 dır. Haritalama mikro buyruğu CAR a 0000000 adresini aktarır. Buda denetim belleğindeki ADD yordamının başlangıç adresidir. BRANCH ve STORE rutinlerinin başlangıç adresler 12, 16, 20, ..., 60 tır. Bu şekilde denetim belleğinde her rutine 4 kelime düşer.

Her rutinde, etkin adresi hesaplayacak mikro buyruk bulunmalıdır. Dolaylı adres kipi bütün bellek adreslemeli buyruklarda vardır. Eğer dolaylı adresle ilgili mikro buyruklar bir alt programda saklanırsa, kısalık sağlanır. Bu alt program INDRCT dir. Hemen al-getir yordamının arkasından gelir. Çizelge 7.2 de al-getir ve dolaylı rutinlerinin sembolik yazılımları görülmektedir.

Dolaylı alt programına geçiş ve oradan dönüş için; al-getir yordamının sonunda ADD yordamına yani 0 adresine dallanıldığı düşünülsün ADD yordamındaki ilk mikro buyruk INDRCT alt programını çağırır. Çünkü durum biti I = 1 dir. Bu rutine dallanır ve 1 adresi SBR yazacına yazılır. INDRCT alt programında 2 mikro buyruk vardır.

Çizelge 7.2 Sembolik mikro program

ETİKET	Mikro işlemler	CD	BR	AD
	ORG 0			
ADD:	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ADD	U	JMP	FETCH
	ORG 4			
BRANCH:	NOP	S	JMP	OVER
	NOP	U	JMP	FETCH
	NOP	I	CALL	INDRCT
	ARTPC	U	JMP	FETCH
	ORG 8			
STORE:	NOP	I	CALL	INDIRECT
	ACTOR	U	JMP	NEXT
	WRITE	U	JMP	FETCH
	ORG 12			
EXCHANGE	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ALTDR, DRTAC	U	JMP	NEXT
	WRITE	U	JMP	FETCH
	ORG 64			
FETCH:	PCTAR	U	JMP	NEXT
	READ, INCPC	U	MAP	
	DRTAR	U	JMP	
INDRCT:	READ	U	JMP	NEXT
	DRTAR	U	RET	

Dolaylı adres durumunda belleğe veriyi değil de, etkin adresi almak için erişilmelidir. Bu adres daha sonra *AR* ye aktarılmalıdır. Alt programdan dönüş (RET) adresi *SBR* den *CAR* a aktarır. Dolayısıyla ADD alt yordamının ikinci mikro buyruğuna dönülür.

Buyrukların İcrası

ADD buyruğunu icra için 1 ve 2 adreslerindeki mikro buyruklar icra edilmelidir. İlk mikro buyruk veriyi bellekten *DR* ye aktarır. İkinci mikro buyruk ADD işlemini *DR* ve *AC* nin içerikleriyle yerine getirir ve sonra al-getir yordamının baş tarafına dallanır.

BRANCH buyruğu etkin adreste bir dallanma meydana getirir. Bunun için $AC < 0$ olmalıdır. *AC* nin < 0 olması için işaret bitinin negatif (1) olması gerekir. Bu ise durum biti *S* nin 1 olmasıyla anlaşılır. Çizelge 7.2 deki BRANCH alt yordamı önce *S* nin değerini kontrol eder. Eğer S = 0 ise dallanma olmaz ve sonraki mikro buyruk al-getir yordamının başına dallanma sağlar. Bu durumda *PC* nin içeriği değişmez. Eğer S = 1 ise ilk JMP, denetimi OVER adresine gönderir. Bu adresteki mikro buyruk, I = 1 ise INDRCT alt programını çağırır. Bundan sonra etkin adres *AC* den *PC* a aktarılır ve mikro program tekrar al-getir alt yordamına gider.

STORE alt yordamı eğer I = 1 ise INDRCT alt programını çağırır. *AC* nin içeriği *DR* ye aktarılır. Bundan sonra *AR* içindeki bellek adresine *DR* nin içeriğini yazma işlemi başlatılır.

EXCHANGE alt yordamı işleneni etkin adresten okur ve onu *DR* ye yazar. *DR* ve *AC* nin içerikleri üçüncü mikro buyruk ile yer değiştirir. *AC* nin içeriği şimdi *DR* dedir ve tekrar belleğe yazılır.

Çizelge 7.2 de 16 mümkün mikro buyruktan 4 tanesi programlanarak, mikro programın bir kısmı verilmiştir. Ayrıca denetim belleğinin 69 dan 127 ye kadar adresleri kullanılmamıştır. Çarpma, bölme gibi diğer işlemler daha uzun mikro işlemlere gerek duyarlar. Bu yüzden 4 ten daha fazla mikro işleme gerek vardır. Denetim belleğinin 69 dan 127 ye kadar adresleri bu iş için kullanılabilir.

İkili Mikro Program

Sembolik mikro program insanların okuma ve anlamaları için mikro program yazımı olarak daha kolay bir yoldur. Sembolik mikro program, mikro programların yazımı için yararlıdır. Bu program derleyici yardımı ile ikiliye çevrilmelidir. Eğer buradaki gibi kısa ise elle de çevrilebilir.

Bu programın ikili eşdeğeri çizelge 7.3 te verilmiştir. Buradaki semboller ve ikili eşdeğeri çizelge 7.1 den çevrilmiştir.

Üçüncü adresin sembolik programda karşılığı yoktur. Çünkü ADD alt yordamında sadece 3 mikro buyruk vardır. Bunlar 0, 1, 2, adreslerindedir. Bundan sonraki rutin üçüncü adresi atlar ve dördüncü adresten başlar. İkili programda ise 3 adresinde 0 bulunur. Herhangi bir durumda 3 adresine hata veya yanlışlıkla dallanması hali için buyruğun sonuna al-getir alt yordamının ilk adresi yazılmıştır.

Denetim Belleği

Çizelge 7.3 te verilen ikili mikro program denetim belleğinin kelime içeriğini gösterir. Denetim belleği olarak ROM kullanıldığında, ikili mikro program, birimin yapılması için gerçeklik çizelgesi görevi yapar. Birimin yapılması bir donanım olayıdır. Her bir kelimedeki 1 ve 0 ları oluşturarak maskelerin yapılması gerekir. ROM'un bitleri bir kez bağlanır. ROM, IC paketlerinden oluşmuştur. Gerekirse çıkarılıp başka paketlerle değiştirilebilir. Bilgisayarın buyruk kümesini değiştirmek için yeni bir mikro program ve yeni bir ROM (ve yeni maske) gerekir.

Eğer yazılabilir denetim belleği kullanılıyorsa ROM yerine RAM kullanılır. RAM kullanmanın kazancı, mikro programın kolayca değiştirilebilmesi 1 ve 0 ların kolayca yazılabilmesidir. Donanım işlemi gerekmez. Böyle bir denetim belleği çok esnek olur. Bilgisayara her zaman yeri buyruklar eklenebilir veya değiştirilebilir. Fakat çoğu mikro programlanmış sistemler, denetim belleği için ROM kullanılır. Daha ucuz ve kolaydır. Ayrıca sıradan kullanıcının bozması olasılığı yoktur.

Çizelge 7.3 Denetim belleği için ikili mikro program parçası

Mikro Yordam	Adres		İkili mikro buyruk					
	Ondalık	İkilik	F1	F2	F3	CD	BR	AD
ADD	0	0000000	000	000	000	01	01	1000011
	1	0000001	000	100	000	00	00	0000010
	2	0000010	001	000	000	00	00	1000000
	3	0000011	000	000	000	00	00	1000000
BRANCH	4	0000100	000	000	000	10	00	0000110
	5	0000101	000	000	000	00	00	1000000
	6	0000110	000	000	000	01	01	1000011
	7	0000111	000	000	110	00	00	1000000
STORE	8	0001000	000	000	000	01	01	1000011
	9	0001001	000	101	000	00	00	0001010
	10	0001010	111	000	000	00	00	1000000
	11	0001011	000	000	000	00	00	1000000
EXCHANGE	12	0001100	000	000	000	01	01	1000011
	13	0001101	001	000	000	00	00	0001110
	14	0001110	100	101	000	00	00	0001111
	15	0001111	111	000	000	00	00	1000000
FETCH	64	1000000	110	000	000	00	00	1000001
	65	1000001	000	100	101	00	00	1000010
	66	1000010	101	000	000	00	11	0000000
INDIRECT	67	1000011	000	100	000	00	00	1000100
	68	1000100	101	000	000	00	10	0000000

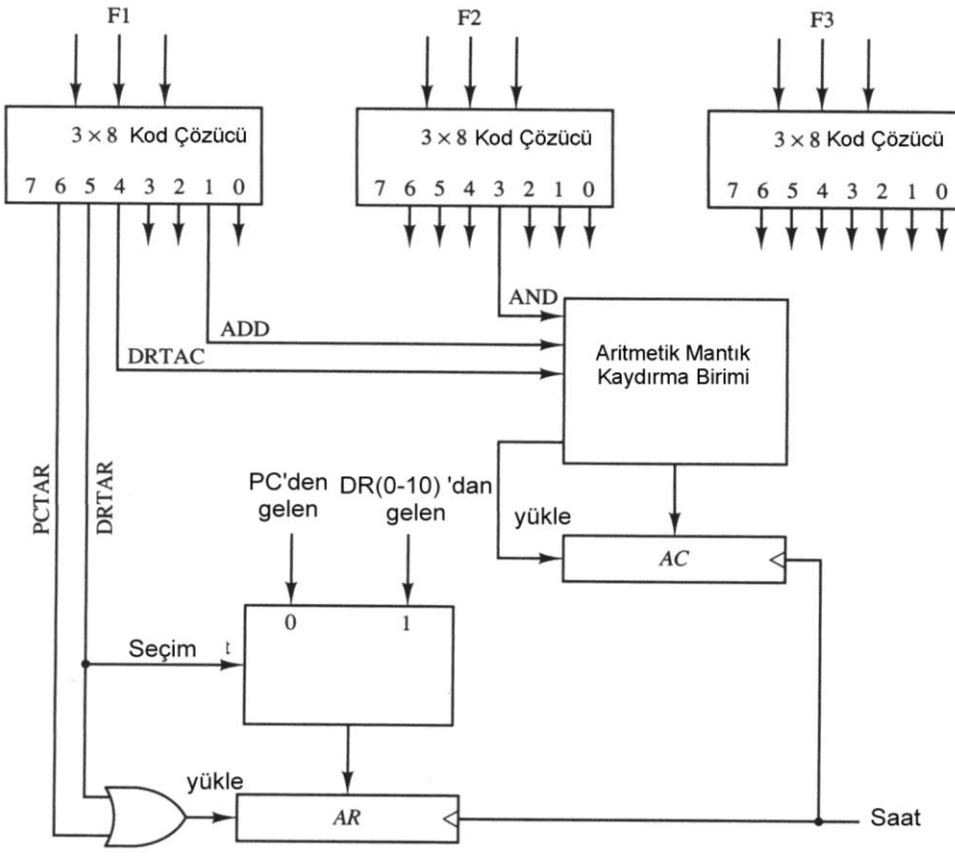
7.4 Denetim Birimi Tasarımı

Çoğunlukla mikro buyruğun bitleri bir takım alanlar şeklinde gruplanmıştır. Her bir alan farklı ve belirgin bir fonksiyon tanımlar. Bu fonksiyonlar veya alanlar sistemdeki mikro işlemleri başlatır. Özel bitlerle adresin nasıl elde edileceği belli olur. Dallanma için adres verilir. Mikro işlemlerin sayısı guruplama ile azaltılabilir. Bir alan k bit olduğunda 2^k tane mikro işlemli gruplar oluşur. Her alan, gereken denetim sinyallerini oluşturmak için bir kod çözücüye gerek duyar. Bu düşünce tarzı mikro buyrukların büyüklüğünü azaltır. Fakat donanımı çoğaltır. Bu durum denetim sinyallerinin süresinin uzamasına neden olur. Çünkü kod çözücü devreleri geçmeleri gerekir.

Denetim bitlerinin kodunun çözülmesi bir önceki kısımda anlatılmıştı. Mikro işlem alanının 9 biti 3 alana ayrılmış olup her biri 3 bittir. Denetim belleği çıkışı ayrık mikro işlemler tanımlamalıdır. Kod çözücülerin çıkışları işlemcinin uygun girişlerine bağlanmalıdır.

F Alanlarının Kodunun Çözülmesi

Şekil 7.7 de 3 kod çözücü ve bunların çıkışlarının bazı bağlantıları görülüyor. Mikro buyruğun 3 alanının her bir alanı denetim belleğinin çıkışında halen mevcut olup bir kod çözücü ile 8 çıkış sağlar. Bu çıkışların her biri, bu işlemi yapacak özel devrenin çalışmasını başlatacak yere bağlanmıştır (çizelge 7.1). Örneğin F1 = 101 olduğunda (ikili5) bir sonraki saat vuruşu *DR*(0-10) içeriğini *AR* ye aktarır (bu şekil DRTAR ile gösterilir). Benzer şekilde F1 = 110 (ikili 6) ise *PC* dan *AR* ye aktarma vardır. Şekil 7.7 de gösterildiği gibi F1 kod çözücünün 5 ve 6 çıkışları *AR* nin yükle girişine bağlıdır. Dolayısıyla bu iki çıkıştan biri aktif olunca MUX lardaki bilgi *AR* ye aktarılır. 5 çıkışı aktif olduğunda MUX bilgiyi *DR* den alır. 5 çıkışı aktif olmayınca bilgi *PC* den alınır. Aktarma 5 veya 6 çıkışları aktif olduğu sürede ve saat vuruşu sürecinde meydana gelir. Kod çözücülerin diğer çıkışları yazaçlar arası aktarmaları benzer biçimde yapar.



Şekil 7.7 Mikro işlem alanlarının kodunun çözülmesi

Aritmetik, Mantık ve Kaydırma Birimi

Aritmetik mantık ve kaydırma birimi şekil 5.19 ve 5.20 deki gibi tasarlanabilir. Bu şekillerde oluşturulan AND, ADD ve *DR* denetimleri yerine, bu denetimler kod çözücü çıkışları olarak gelebilir. Bunlar AND, ADD ve DRTAC buyrukları ile ifade edilebilir. (şekil 7.7) Kod çözücülerin diğer çıkışları benzer şekilde bağlanabilir. *AC* işlemi ile ilgili kod çözücülerin diğer çıkışları, benzer şekilde aritmetik mantık ve kaydırma birime bağlanmalıdır.

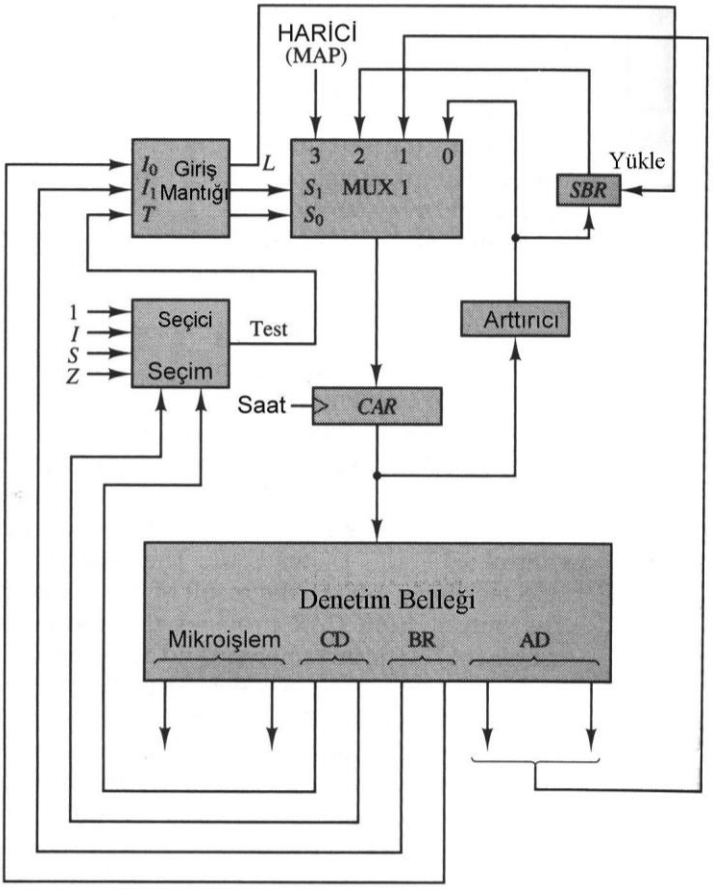
Mikro Program Sıralayıcı

Mikro programlanmış denetimin temel bileşenleri denetim belleği ve bir sonraki adresi seçen devreleridir. Adres seçimi yapan kısma mikro program sıralayıcı adı verilir. Böyle bir sıralayıcı, özel bir işlevi yerine getirmek üzere tasarlanabilir. Bunlar daha genel mikro program sıralayıcı IC devreler şeklinde de olabilir. Bunların iç tasarımlarının çok geniş çapta kullanıcılar tarafından kullanılmasını sağlayacak biçimde olması gerekir.

Bir mikro program sıralayıcısının amacı bir mikro buyruğunun okunup icra edilmesini sağlayacak şekilde denetim belleğine bir adres sunmaktır. Sıralayıcının sonraki adres mantığı denetim adres yazacına yüklenecek özel adres kaynağını belirlemektir. Adres kaynağının seçimi sıralayıcının hali hazırdaki mikro buyruktan aldığı sonraki adres bilgi bitleri tarafından yönlendirilir. Ticari sıralayıcılar içlerinde mikro program döngüsü ve yordam çağrılması esnasında geçici adres depolaması için kullanılan bir dahili yazaç yığıtı içerir. Bazı sıralayıcılarda denetim belleği için adres yazacı işlevi gören bir çıkış yazacı vardır.

Burada önceki bölümde tanımlanan bilgisayar örneği için bir mikro program sıralayıcı veriliyor. Blok şeması şekil 7.8 dir. Denetim belleği de şemada verilerek, sıralayıcı ile bağlı belleğin etkileşimi gösterilmiştir. Devrede iki MUX vardır. Birinci MUX kaynaktan birinden adresi seçer ve onu *CAR* a yollar. İkinci MUX seçilen durum bitini test eder ve test sonucu giriş mantık devresine uygulanır. *CAR* ın çıkışı denetim belleği için adresi verir. *CAR* ın içeriği bir arttırılırsa MUX girişlerinden birine verilir ve aynı zamanda *SBR* yazacına gönderilir. 1 numaralı MUX un diğer 3 girişi halen icra edilecek mikro buyruğun adres alanından, *SBR* nin çıkışından ve buyruğu haritalayan dış kaynaktan gelir. Her ne kadar blok şemada tek bir *SBR* yazacı görülüyorsa da tipik bir sıralayıcıda, 4 ile 7 uzunluğunda bir yığıt bulunur. Böylece aynı anda çalışan alt programların sayısı artar. Bunlar için uygun POP ve PUSH işlemleri CALL ve RET işlemlerine cevap verir.

İkinci MUX, mikro buyruğun CD alanındaki durum bitlerinden birini seçer. Eğer bit 1 ise, Test değişkeni 1 dir ve 0 ise 0 dır. *T* (test) değişkeni BR alanının iki biti ile birlikte giriş mantık kapılarına gelir. Bu birim



Şekil 7.8 Bir denetim belleği için mikro program sıralayıcı

mevcut işlemin tipini belirler. Bunlar arttırma, dallanma veya atlama, alt program çağırma veya alt programdan çıkma, dış adresi yükleme ve yığıttan PUSH ve POP tur. Başka adres sıralama işlemleri de olabilir. 3 girişle sıralayıcı 8 adres sıralama işlemi belirleyebilir. Ticari sıralayıcılar, T girişine ek olarak 3 veya 4 girişe sahip olup, daha fazla sayıda işlem belirleyebilirler.

Giriş Mantık Devresi Tasarımı

Şekil 7.8 deki giriş mantık devresinin 3 girişi vardır. I_0, I_1 ve T. 3 de çıkışı vardır. S_0 S_1 ve L. S_0S_1 değişkenleri kaynak adreslerden birini seçer (Bu adres *CAR* a gidecek). L değişkeni ise *SBR* nin yükle girişidir. İki değişkenin ikili değerleri MUX içinde yol seçer. Örneğin $S_1S_0 = 10$ ise MUX2 seçilir ve *SBR* den *CAR* a bir geçiş sağlanır. 4 giriş ve MUX1 in çıkışları 7 bit adres içerir. Giriş mantık devresinin doğruluk çizelgesi çizelge 7.4 te gösterilmiştir.

Çizelge 7.4 Mikro program sıralayıcı için giriş mantık devresi doğruluk çizelgesi

BR Alanı		Giriş			MUX 1		SBR yükle
		I_1	I_0	T	S_1	S_0	L
0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0
0	1	0	1	0	0	0	0
0	1	0	1	1	0	1	1
1	0	1	0	×	1	0	0
1	1	1	1	×	1	1	0

I_0 ve I_1 girişleri BR alanındaki bitlerdir. Her bir girişi bir fonksiyon belirler. Bu fonksiyon çizelge 7.1 de verilmiştir. S_0 ve S_1 bitleri verilen fonksiyondan ve MUX taki yoldan belirlenir. BR = 01 olunca *CAR* daki adres bir arttırılır ve CALL mikro buyruğu ile SBR yazacına yüklenir. Bu durum $T = 1$ ise gerçekleşir. Doğruluk çizelgesi yardımıyla basitleştirilmiş Boole fonksiyonları giriş mantık devresi için elde edilir.

$$S_1 = I_1$$

$$S_0 = I_1 I_0 + I_1' T_1$$

$$L = I_1' I_0 T$$

Bu devre üç VE, bir VEYA ve bir TERSLEYEN ile yapılabilir. Şekil 7.8 deki sıralayıcı bir sayaç olmayıp, kapılarla yapılan bir birleşik devredir. Bu devre yarım toplayıcılarla yapılabilir (şekil 4.8) .

SORULAR

7-1. Bir mikroişlemci ile bir mikro program arasındaki fark nedir? Mikro programsız bir mikro işlemci tasarlamak mümkün mü? Tüm mikro programlanmış bilgisayarlar aynı zamanda mikroişlemciler midir?

7-2. Donanımsal denetim ile mikro programlanmış denetim arasındaki farkı açıklayın. Donanımsal denetimle denetim belleğini beraberce kullanmak mümkün müdür?

7-3. Aşağıdakileri tanımlayın. (a) mikro işlem; (b) mikro buyruk; (c) mikro program; (d) mikro kod.

7-4. Şekil 7.1'de gösterilen mikro programlanmış denetim kurumunda şu yayılma gecikmeleri vardır. 40 ns sonraki adresi oluşturmak için, 10 ns adresi denetim adres yazacına aktarmak için, 40 ns denetim belleği ROM'a erişmek için, 10 ns mikro buyruğu denetim veri yazacına aktarmak için ve 40 ns denetim kelimesinin belirlediği mikro işlemcileri

icra etmek için. Denetimin kullanacağı saat frekansını bulun. Eğer denetim veri yazacı kullanılmazsa saat frekansı ne olmalıdır?

7-5. Şekil 7.2'de gösterilen denetim belleği her biri 3 bitlik 1024 kelimeden oluşuyor. Mikro buyruklar, şekilde görüldüğü gibi 3 alandan oluşuyor. Mikroişlemciler alarak 16 bittir.

a. Dallanma adres alanı ve seçim alanı kaçar bittir?

b. Eğer sistemde 16 durum biti varsa, bir durum bitini seçmek için dallanma mantığı kaç bit kullanır 4096 kelimeden oluşuyor?

c. Seçicide bir girişi seçmek için kaç bit kaldı?

7-6. Şekil 7.2'deki denetim belleği, her biri 24 bitlik.

a. Denetim adres yazacı kaç bittir?

b. Seçicilere giden 4 girişin her biri kaçar bittir?

c. Her bir seçicinin kaçar girişi vardır ve kaç tane seçici gerekir?

7-7. Şekil 7.3'te tanımlanan haritalama işlemlerini kullanarak aşağıdaki işlem kodları için ilk mikro buyruk adresini söyleyin. (a) 0010; (b) 1011; (c) 1111.

7-8. 8 ardışık mikro buyruğu bulacak yordam için bir haritalama yöntemi bulun. İşlem kodu 6 bit ve denetim belleği 2048 kelimedir.

7-9. Yalnız okunabilir bir bellek yardımıyla, bir buyruk kodundan bir mikro buyruk adresine haritalama yapılabilir? Bunun, 7.3'te açıklanana göre üstünlüğü nedir?

7-10. Şekil 7.4'te görülen bilgisayar donanım kurulumunda neden iki tane seçiciye gereksinimimiz var? Farklı kaynaklardan tek bir ortak hedefe bilgi aktarımı yapmanın başka yolu var mı**?**

7-11. Çizelge 7.1'i kullanarak, aşağıdaki mikro işlemciler için 9 bitlik mikro işlem alanını bulun.

a. $AC \leftarrow AC + 1, \quad DR \leftarrow DR + 1$

b. $PC \leftarrow PC + 1, \quad DR \leftarrow M[AR]$

c. $DR \leftarrow AC, \quad AC \leftarrow DR$

7-12. Çizelge 7.1'i kullanarak aşağıdaki sembolik mikroişlemcileri yazaç aktarım deyimlerine ve 2 li sistem çevirir.

a. READ, INCPC

b. ACTDR, DRTAC

c. ARTPC, DRTAC, WRITE

7-13. Varsayalım ki Çizelge 7.2'deki ADD yardımını aşağıdaki iki mikro buyrukla değiştirdik.

```
ADD :  READ   I      CALL   INDR 2
       ADD    U      JMP    FETCH
```

Bu INDR2 yordamı ne olmalıdır?

7-14. Aşağıdaki sembolik mikro program bölüm 7.3'te tanımlanan bilgisayarın bir buyruğu içindir?

```
ORG 40
NOP          S    JMP     FETCH
NOP          Z    JMP     FETCH
NOP          I    CALL    INDRCT
ARTPC        U    JMP     FETCH
```

a. Bu buyruk icra edildiğinde yapılanı işi söyleyin.
b. 4 mikro buyruğu eşdeğer ikili biçime dönüştürün.

7-15. Bölüm 7.3'teki bilgisayarın aşağıdaki ikili mikro programı var

Adres	İkili mikro program																			
60	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	1
61	1	1	1	1	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0
62	0	0	1	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0
63	1	0	1	1	1	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0

a. Bunu çizelge 7.2'deki gibi sembolik mikro programa çevirin. (FETCH; 64 adresinde, INDRCT; 67 adresinde)
b. Bu program bilgisayarda icra edildiğinde yanlış olacak satırları yazın.

7-16. Bölüm 7.3'teki bilgisayarlara aşağıdaki buyrukları ekleyin (*EA* etkin adrestir). Çizelge 7.2'deki her bir yordam için sembolik mikro program yazın. (Dikkat: Bir buyruk değerini değiştirmedikçe *AC* değişmez)

Sembol	İşlem kodu	Sembolik fonksiyon	Tanım
AND	0100	$AC \leftarrow AC \wedge M[EA]$	VE
SUB	0101	$AC \leftarrow AC - M[EA]$	Çıkarma
ADM	0110	$M[EA] \leftarrow M[EA] + AC$	Belleğe toplama
BTCL	0111	$AC \leftarrow AC \wedge \overline{M[EA]}$	Bit silme
BZ	1000	Eğer ($AC = 0$) ise ($PC \leftarrow EA$)	*AC* sıfır ise dallan
SEQ	1001	Eğer ($AC = M[EA]$) ise ($PC \leftarrow PC + 1$)	Eşit ise atla
BPNZ	1010	Eğer ($AC > 0$) ise ($PC \leftarrow EA$)	Pozitif ve sıfır değilse dallan

7-17. ISZ (Arttır ve sıfırsa atla) buyruğu için Bölüm 5 (Çizelge 5.4) sembolik mikro program yazın. Bölüm 7.3'teki mikro buyruk biçimini kullanın. Bölüm 7-3'te tanımlanan bilgisayarda, CD alanında $DR = 0$ durum şartı yoktur. Buna karşılık *AC* ile *DR* yi yer değiştirebilir ve 7 bitiyle beraber $AC = 0$ mı diye bakabilirsiniz.

7-18. Bilgisayara (dallan ve dönüş adresini sakla) buyruğu için (Çizelge 5-4) mikro program yordamları yazın. Bölüm 7.3'te mikro buyruk biçimini kullanın. Mikro buyrukların sayısını en aza indirin.

7-19. Şekil 7.7 deki F3 kod çözücünün 5 ve 6 çıkışlarının *PC* ye nasıl bağlanacağını gösterin.

7-20. Bir mikro buyruk içindeki 9 bit mikro işlem alanının 46 mikro işlemi belirtecek biçimde nasıl alt alanlara bölüneceğini bulun.

7-21. Bir bilgisayarda 16 yazaç, bir ALU, 32 işlem ve 8 işlemli bir kaydırıcı bulunuyor. Hepsi ortak veri yoluna bağlıdır.

a. Bir mikroişlem için bir denetim kelimesi yaratın

b. Denetim kelimesi içindeki her bir alanı ve alandaki bilgisayarlarını yazın. Genel bir kod dönüştürme yöntemi belirleyin.

c. $R4 \leftarrow R5 + R6$ mikro işlemini belirleyecek bir denetim kelimesinde bunu belirten bitleri gösterin

7-22. Varsayalım ki Bölüm 7.8'deki mikro program sıralayıcının 4 girişli, I_2, I_1, I_0 ve T (test) ve 3 çıkışı, S_1 S_0 ve L den oluşur. Bu birimde yapılabilen işlemler aşağıdaki çizelgede verilmiştir. en az sayıda kapı kullanarak giriş mantık devresini çizin.

I_2	I_1	I_0	İşlem
0	0	0	$T = 1$ ise *CAR*'ı arttır. $T = 0$ ise *AD* ye atla
×	0	1	Şartsız olarak *AD*'ye atla
1	0	0	Şartsız olarak *CAR*'ı arttır
0	1	0	$T = 1$ ise *AD*'ye atla, $T = 0$ ise *CAR*'ı arttır.
1	1	0	$T = 1$ ise yordamı çağır, $T = 0$ ise *CAR*'ı arttır
0	1	1	Şartsız olarak yordamdan dön
1	1	1	Şartsız olarak dış adresi haritala

7-23. Şekil 7.8'deki (Şekil 4.8'e de bak) mikro program sıralayıcı için 1 arttırıcı birleşik devre (7-bit) çizin. Derleyici, D denetim girişi koyarak değiştirin. Öyle ki $D = 0$ iken arttırıcı 1 arttırsın fakat $D = 1$ iken 2 arttırsın

7-24. Şekil 7.8'deki MUX 2 ile giriş mantık devresi arasında bir YADA kapısı yerleştirin. Kapının bir girişi MUX un ters çıkışından diğeri ise denetim belleğindeki mikro buyruk P ile gösterilen bit ten gelsin. Kapının çıkışı giriş mantık devresinin T girişine gelsin P biti ne işe yarar?

KAYNAKLAR

1. Dasgupta, S., *Computer Architecture : A Modern Synthesis*. Vol. 1. New York : John Wiley, 1989.
2. Gorsline, G. W., Computer *Organization: Hardware/Software*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1986.

3. Hamacher, V. C., Z. G. Vranesic, and S. G. Zaky, *Computer Organization*, 3rd ed. New York: McGraw-Hill, 1990.
4. Hays, J. F., Computer Architecture and Organization. 2nd ed. New York: McGraw-Hill, 1988.
5. Langholz, G., J. Francioni, and A. Kandel, Elements of Computer Organization. Englewood Cliffs, NJ: Prentice Hall, 1989.
6. Lewin, M. H., Logic Desing and Computer Organization. Reading, MA: Addison-Wesley, 1983.
7. Mano, M. M., Computer Engineering :Hardwire Desing, Englewood Cliffs, NJ: Prentice Hall, 1988.
8. Rafiquzzaman, M., and R. Chandra, Modern Computer Architecture. St Paul, MN: West Publishing, 1988.
9. Stallings, W., Computer Organization and Architecture, 2nd ed. New York:Macmillan, 1989.
10. Tanenbaum, A. S., Structured Computer Organization, 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1990.
11. Ward. S. A., and R. H. Halstead, Jr., Computer Structures. Cambridge, MA: MIT Press, 1990.

BÖLÜM 8

Merkezi İşlemci Birimi

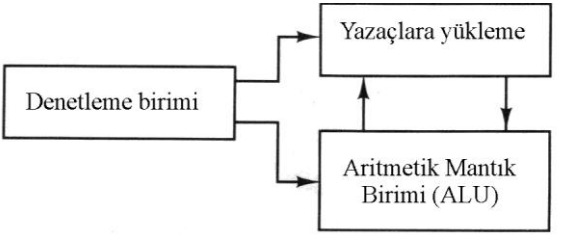
BU BÖLÜMDE

8-1 Giriş
8-2 Genel Yazaç Kurulumu
8-3 Yığıt Kurulumu
8-4 Buyruk Biçimleri
8-5 Adresleme Kipleri
8-6 Veri Aktarımı ve İşleme
8-7 Program Denetimi
8-8 Azaltılmış Buyruk Kümeli Bilgisayar (RISC)

8.1 Giriş

MİB

Veri işleme işlemlerini yerine getiren bilgisayarın önemli bir kısmı merkezi işlemci birimidir ve MİB ile gösterilir. Şekil 8.1 de görüldüğü gibi MİB üç ana bölümden oluşur. Yazaç kümesi, buyrukların çalışması boyunca kullanılan ara verileri depolar. ALU buyrukların çalışması için gerekli mikro işlemleri yerine getirir. Denetim birimi, yazaçlar içindeki bilginin transferinde ve ALU buyruklarında yönetici durumundadır.



Şekil 8.1 MİB nin başlıca elemanları

Bilgisayarda bulunan buyruk tipleri tarafından yazdırılan fonksiyonları MİB yerine getirir. Bilgisayar mimarisi, bilgisayar donanımı ve makine dili buyrukların kullanıldığı programlardan oluşur. Bu da buyruk biçimleri, adresleme kipleri, buyruk kümeleri ve MİB yazaçlarının genel kurulumlarından oluşur.

Bilgisayarı tasarlayan ve programlayanlar için bir sınırlama MİB ile buyruk kümeleri arasında ilişki kurmakta ortaya çıkmaktadır. Tasarımcıya göre, bilgisayar buyruk kümeleri MİB tasarımı için şartlar içerirler. Bir MİB in tasarımı başlı başına bir iştir. Çünkü seçilen donanımda makine buyruklarını çalıştırmak için geniş bir bölüm içerir.

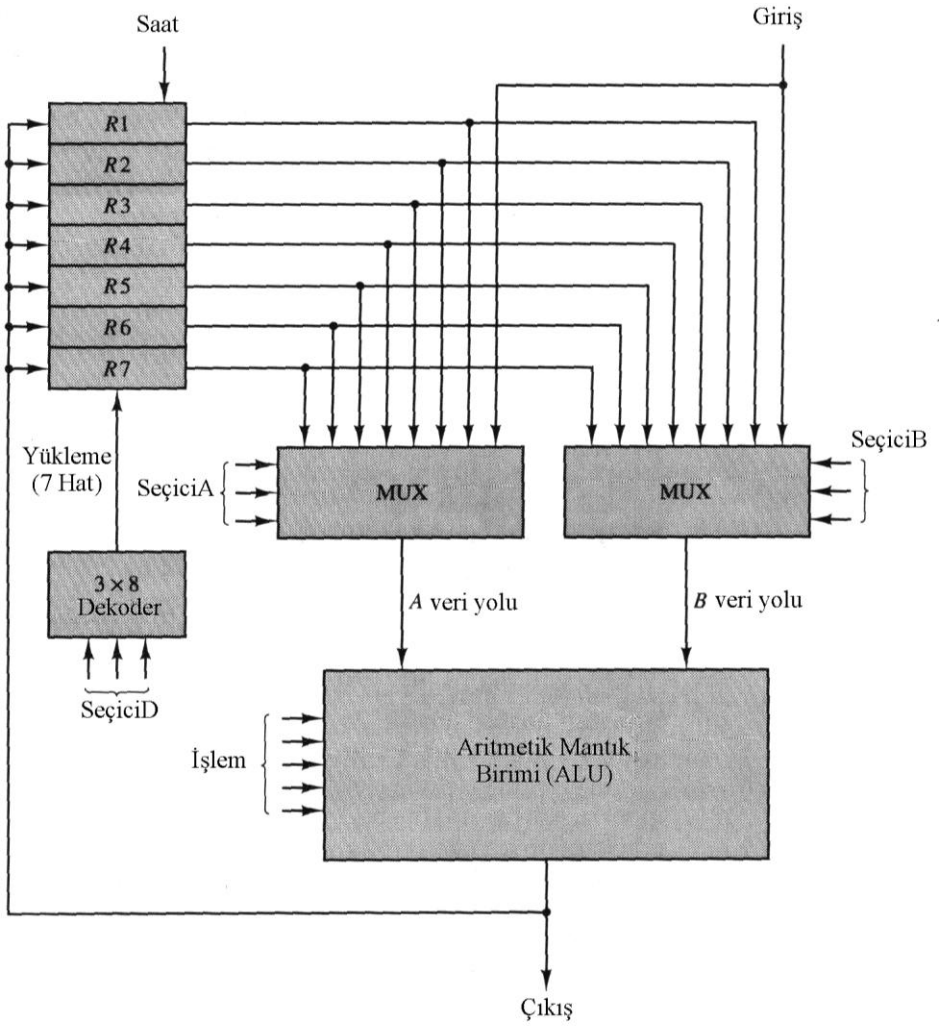
Basit bir MİB tasarım örneği bölüm 5 ve 7'de vardı. Bu bölümde MİB in mimari ve tasarımı kullanıcı gözü ile tanımlanmaktadır. Yazaçların ALU ile veri yolları aracılığı ile nasıl haberleştiği anlatılmakta ve bellek yığıtının işleyişi açıklanmaktadır. Daha sonra mevcut buyruk biçimleri, bellekten veri alma için kullanılan adresleme kipleri ve bilgisayarlarda yaygın olarak kullanılan belli başlı buyruklar sunulmaktadır. Son bölüm RISC kavramını anlatmaktadır.

8.2 Genel Yazaç Kurulumu

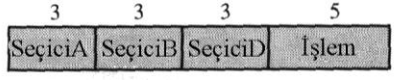
Bölüm 6'da verilen program örneğinde yerleştirme göstergeleri, sayaçlar, dönüş adresleri, geçici sonuçlar ve kısmi çarpımları yerleştirmek için bellek kısımlarına gerek vardır. Bu işler için belleği kullanmak çok zaman alır. Çünkü MİB içindeki en yavaş iş bellekle haberleşmedir. Daha uygun bir yol bu değerlerin işlemci yazaçlarına konmasıdır. MİB içinde çok sayıda yazaçlar kullanıldığında bunları bir veri yolu sistemiyle bağlamak veya çok küçük bir bellek parçasına hepsini toplayıp hızlı erişim sağlamak işi hızlandırır (ön bellek). Yazaçlar sadece veri aktarımları için değil mikro işlemlerin icrasında da kullanılır.

Veri Yolu Sistemi

Şekil 8.2'de 7 tane MİB yazacı için bir veri yolu kurulumu gösterilmiştir. Her bir yazacın çıkışları iki MUX a bağlanarak A ve B giriş veri yolları oluşturulmuştur. Her bir MUX içindeki seçim telleri veri yoluna bilgi girişi için bir yazacı seçer. *A* ve *B* veri yolları ortak ALU için girişleri oluştururlar. Burada ALU için seçilen fonksiyon icra edilmesi gereken aritmetik veya mantıksal mikro işlemleri yapar. Mikro işlemlerin sonuçları çıkış verisi olarak vardır. Bunlar hedef veri yoluna gider ve bütün yazaçların girişlerine ulaşabilirler. Hedef yazaç bilgiyi hedef veri yolundan alır. Bu seçim bir kod çözücü tarafından yapılır. Kod çözücü yazaç yükle girişlerinden birini aktif hale getirir. Seçilen hedef yazaç ile hedef veri yolu içindeki veri arasında bir aktarma köprüsü kurulmuş olur.



(a) Blok Şema



(b) Denetim Kelimesi

Şekil 8.2 ALU ile yazaç kümesi.

MİB i çalıştıran denetim birimi bilgiyi, sistem içindeki çeşitli bileşenler yardımıyla ALU üzerine gönderir. Örneğin aşağıdaki mikro işlemin uygulamasına bakılırsa

$$R1 \leftarrow R2 + R3$$

için denetim birim, ikili seçim değişkenleri bulmalı ve aşağıdaki seçim girişlerine uygulamalıdır.

1. MUX *A* seçici (SELA): *R*2 nin içeriğini *A* veri yoluna
2. MUX *B* seçici (SELB): *R*3 en içeriğini *B* veri yoluna
4. Kod çözücü hedef seçici (SELD): Çıkış veri yolunun içeriğini *R*1'e aktarmak için.

Bu dört denetim fonksiyonu aynı anda üretilmeli, bir saat vuruşunda hazır olmalıdır. İki kaynak yazaçtan ikili bilgiler MUX içindeki birleşik kapılardan geçerek çıkış veri yoluna ve hedef yazacın girişlerine aynı saat vurusunda varmalıdır. Bir sonraki *CP* geldiğinde çıkış veri yolundaki bilgiler *R*1'e aktarılmış olmalıdır. Hızlı sonuç alınması için ALU, hızlı devre elemanlarından yapılmaktadır.

Denetim Kelimesi

Denetim Kelimesi

Bir denetim kelimesi 14 ikili seçici girişin değişik birleşimleri ile tanımlıdır. 14-bitlik denetim kelimesi şekil 8.2(b) de gösterilmiştir. 4 alandan oluşur. 3 alanın her biri 3 bitten 1 alan 5 bitten oluşur. 3 bitlik SELA ALU nun *A* girişi için kaynak yazacı seçer. 3 bitlik SELB ALU nun *B* girişi için kaynak yazacı seçer. 3 bitlik SELD kod çözücü (ve onun 7 yükle girişini) kullanarak hedef yazacı seçer. 5 bitlik OPR ALU nun işlemlerinden birini seçer.

Yazaç seçimlerinin kodlanması çizelge 8.1 de verilmektedir. 3 bitlik ikili kod birinci kolonda listelenmektedir. SELA ve SELB 000 olduğunda MUX lar dış girişi alır. SELD 000 olduğunda hedef yazaç seçilmez, fakat çıkış veri yolunun içeriği çıkış için geçerlidir.

Çizelge 8.1 Yazaç seçim alanlarının kodlanması

İkili kod	SELA	SELB	SELD
000	Giriş	Giriş	Bir şey yok
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

Çizelge 8.2 ALU işlemlerinin açıklanması

OPR seçimi	İşlem	Sembol
00000	*A* nın aktarımı	TSFA
00001	*A* yı 1 arttırma	INCA
00010	*A* + *B* işlemi	ADD
00101	*A* – *B* işlemi	SUB
00110	*A* yı 1 azaltma	DECA
01000	*A* ve *B* yi VE le	AND
01010	yi VEYA la	OR
01100	yi ÖZEL-VEYA la	XOR
01110	*A* nın tümleyenini al	COMA
10000	*A* yı sağa kaydır	SHRA
11000	*A* yı sola kaydır	SHLA

ALU

ALU aritmetik ve mantıksal işlemleri sağlar. Toplama işlemi MİB nin kaydırma işlemleri ile sağlanır. Kaydırıcı ALU nun girişinde yer alır. Bazı durumlarda kaydırma işlemleri ALU ile birlikte bulunur. Bölüm 4.7 de bir aritmetik, mantıksal ve kaydırma birimi düzenlenmişti. ALU için fonksiyon çizelgesi, çizelge 4.8 de listelenmiştir. ALU işlemlerinin kod çözümü MİB için çizelge 8.2 de tanımlanmıştır. OPR alanı 5 bitten oluşmakta ve her bir işlemin sembolik adı bulunmakta.

Mikro İşlem Örnekleri

14 bitlik bir denetim kelimesinde MİB nin bir mikro işlemi vardır. Mikro işlem için bir denetim kelimesi seçim girişlerinden elde edilir. Örneğin

$$R1 \leftarrow R2 - R3$$

çıkarma işleminde ALU nun *A* girişi için *R*2, *B* girişi için *R*3, hedef yazacı için *R*1 belirlenmektedir. ALU nun işlemi *A* – *B* dir. Bu yüzden denetim kelimesi 4 alanla tanımlanmakta ve her bir alan için uygun ikili değerler çizelge 8.1 ve 8.2 den elde edilmektedir. Çıkarma işlemi için denetim kelimesi 010 011 001 00101 dir ve aşağıdaki gibi elde edilir.

Alan:	SELA	SELB	SELD	OPR
Sembol:	*R*2	*R*3	*R*1	SUB
Denetim kelimesi:	010	011	001	00101

Bu mikro işlem ve birkaç tanesi için denetim kelimesi çizelge 8.3 te gösterilmektedir.

Çizelge 8.3 MİB mikro işlem örnekleri

	Sembolik Gösterim				
Mikro işlem	SELA	SELB	SELD	OPR	Denetim Kelimesi
$R1 \leftarrow R2 - R3$	*R*2	*R*3	*R*1	SUB	010 011 001 00101
$R4 \leftarrow R4$ V R5	*R*4	*R*5	*R*4	OR	100 101 100 01010
$R6 \leftarrow R6 + 1$	*R*6	—	*R*6	INCA	110 000 110 00001
R7 ← R1	*R*1	—	*R*7	TSFA	001 000 111 00000
Çıkış ← R2	R2	—	None	TSFA	010 000 000 00000
Çıkış ← Giriş	giriş	—	None	TSFA	000 000 000 00000
R4 ← ShlR4	R4	—	R4	SHLA	100 000 100 11000
R5 ← 0	R5	R5	R5	XOR	101 101 101 01100

Arttırma ve transfer mikro işlemleri ALU nun *B* girişini kullanmaz. Bu durumda *B* alanı – ile gösterilmiştir. TSFA işlemi belirtilen yazaç veya girişi çıkışa yönlendirir. $x \oplus x = 0$ olduğundan ÖZEL-VEYA işlemi ile *A* sıfırlanabilir.

Bu örneklerden görüldüğü gibi diğer bir çok mikro işlem MİB içinde de üretilebilir. Denetim kelimelerini depolayan bir denetim birimi denetim belleği olarak isimlendirilir. Bellekten denetim kelimelerin ardışık olarak okunması MİB içindeki mikro işlemlerin istenilen sırada başlatılması mümkündür.

8.3 Yığıt Kurulumu

LIFO

MİB içine dahil edilmiş çok faydalı bir şeyde yığıt veya lifo dur. Yığıt bir bellek parçasıdır ve son depolanan bilgi ilk geri dönen bilgi olur. Yığıta aktarılan son bilgi yığıtın en tepesinde bulunmaktadır.

Yığıt Göstergesi

Sayısal bilgisayarlarda yığıt temel olarak bir bellek birimidir. Bir adres yazacı vardır. Fakat bu sadece sayabilir, başka hiçbir değer bu adres yazacına yazılamaz. Bu tip adres yazaçlarına yığıt göstergesi adı verilir. Çünkü bunun değeri daima yığıttaki son kelimenin adresini gösterir. Yığıtın fiziksel yazaçları yazma ve okuma için hazırdır. Yani bir kelime araya dahil edilir veya silinir. Yığıt göstergesi daima son olarak dahil edilen kelimenin adresini tutar.

Yığıt ile ilgili iki işlem ekleme ve silmedir. Ekleme işlemi *itme* olarak adlandırılır. Çünkü üste bir şey konur ve alttakiler aşağıya gider. Bunun tersi işlem silme işlemidir ve *çekme* olarak adlandırılır. Çünkü bu işlem sonucu alttakilerin bir yukarı çıktıkları varsayılır. Gerçekte tabi ki hiçbir şey itilmez veya çekilmez. Sadece yığıt göstergesinin içeriği azaltılıp çoğaltılarak olay benzetilir.

Yazaç Yığıtı

Bir yığıt, belleğin bir bölümünde yer alır veya yığıt sonlu sayıda bellek kelimeleri ve yazaçların toplamı ile düzenlenebilir. Şekil 8.3 64 kelimelik yazaç yığıt kurulumunu gösteriyor. Yığıt göstergesi *SP* yığıtın halen en tepesinde bulunan kelimenin adresinin ikili değerini içerir. Yığıtta 3 bilgi var. *A*, *B* ve *C*. *C* en tepede. Dolayısıyla *SP* deki adres 3 yani *C* nin adresi. Tepedeki bilgiyi silmek için 3 adresindeki bilgi okunur sonra *SP* nin içeriği azaltılır. *B* şimdi yığıtın tepesindedir. Çünkü *SP* de 2 adresi vardır. Yeni bir bilgiyi yığıta yazmak için *SP* bir arttırılır ve yeni adrese yeni bilgi yazılır. *C* bilgisi evvelce okunmuş, fakat fiziksel olarak silinmemişti. Fakat bu durum bir sorun oluşturmaz. Çünkü oraya yeni bir kelime yazılır.

64 kelimelik bir yığıtta *SP* 6 bit yer içerir. Çünkü $2^6 = 64$. Bu yüzden *SP* 6 bittir ve değeri 63'ten (ikili 111111) büyük olamaz. 63 1 arttırıldığında 0 sonucuna ulaşılır. 111111 + 1 = 1000000 ikili olarak. Fakat *SP* en küçük 6 önemli biti tuttuğundan 0 elde edilir. Benzer olarak 000000 dan 1 çıkarıldığında sonuç 111111 olur. Bir bitlik *DOLU* yaz-bozu, yığıt dolu olduğunda 1 yapılır ve bir bitlik *BOŞ* yaz-bozu, yığıt boş ise 1 yapılır. Veri yazacı *DR* yığıta yazılan veya yığıttan okunan ikili veriyi tutar.

İtme

Başlangıçta yığıt göstergesi 0 ile temizlenir, *BOŞ* 1 yapılır ve *DOLU* 0 ile silinir, dolayısıyla *SP* 0 adresini gösterir ve yığıtın boş olduğunu işaret eder. Eğer yığıt dolu değil ise (yani *DOLU* = 0 ise) itme işlemi ile yeni bilgi girilebilir. İtme işleminin mikro işlemleri

$SP \leftarrow SP + 1$	*SP* yi 1 arttırır.
$M[SP] \leftarrow DR$	*DR* deki bilgiyi yığıtın en üstüne yaz.
Eğer ($SP = 0$) ise ($DOLU \leftarrow 1$)	Yığıtın doluluğunu kontrol et
$BOŞ \leftarrow 0$	Yığıtın boş olmadığını işaret et

SP bir arttırıldığından bir yukarıdaki kelimenin adresini gösterir durumuna geldi. Belleğe yazma işlemi ile *DR* deki bilgi yığıtın en üst kısmına yazılır. Çünkü *SP* de daima en üst kısmın adresi vardır. Eğer *SP* sıfır olursa yığıt doludur ve *DOLU* = 1 yapılır. *SP* nin 0 olması, son adres 63 iken buna bir eklenince *SP* taşar ve kalan kısım 0 olur. Dolayısıyla yeni bilgi 0 adresine yazılmağa çalışılır. 0 adresine bilgi yazılınca yığıtta yer kalmamış demektir. Dolayısıyla *BOŞ* sıfır yapılır.

Çekme

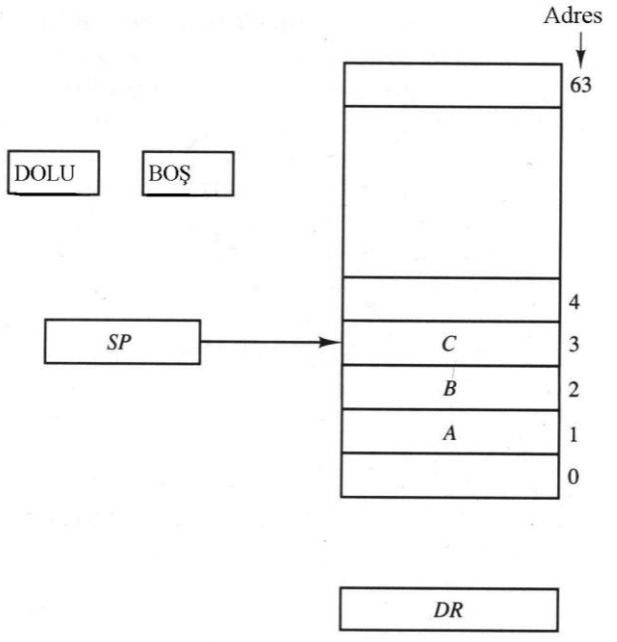
Eğer yığıt boş değilse ($BOŞ = 0$) bu taktirde son girilmiş bilgi yığıttan silinebilir. Çekme işlemi aşağıdaki mikro işlemleri içerir.

$DR \leftarrow M[SP]$	Yığıttan bilgiyi oku
$SP \leftarrow SP$-1	*SP* yi 1 azalt
Eğer ($SP = 0$) ise ($BOŞ \leftarrow 1$)	Yığıtın boşluğunu kontrol et
$DOLU \leftarrow 0$	Yığıtın dolu olmadığını işaretle

İlk mikro işlem yığıttan bilgiyi *DR* ye okur, sonra *SP* bir azaltılır. Eğer $SP = 0$ olursa yığıt boştur ve $BOŞ \leftarrow 1$ yapılır. Bu koşul eğer okunan bilgi 1 nolu adreste ise yapılır. Bu adres okunduğu zaman yığıttaki bütün yazaçlar boştur. Eğer okuma işlemi 1 adresinden değil de 0 adresinden yapılırsa, *SP* azaltılınca *SP* nin değeri 63 yani ikili 111111 olur, 0 adresi yığıttaki son değeri alır. $DOLU = 1$ iken itme veya $BOŞ = 1$ iken çekme işlemi yapılırsa hatalı işlem yapılmış olur.

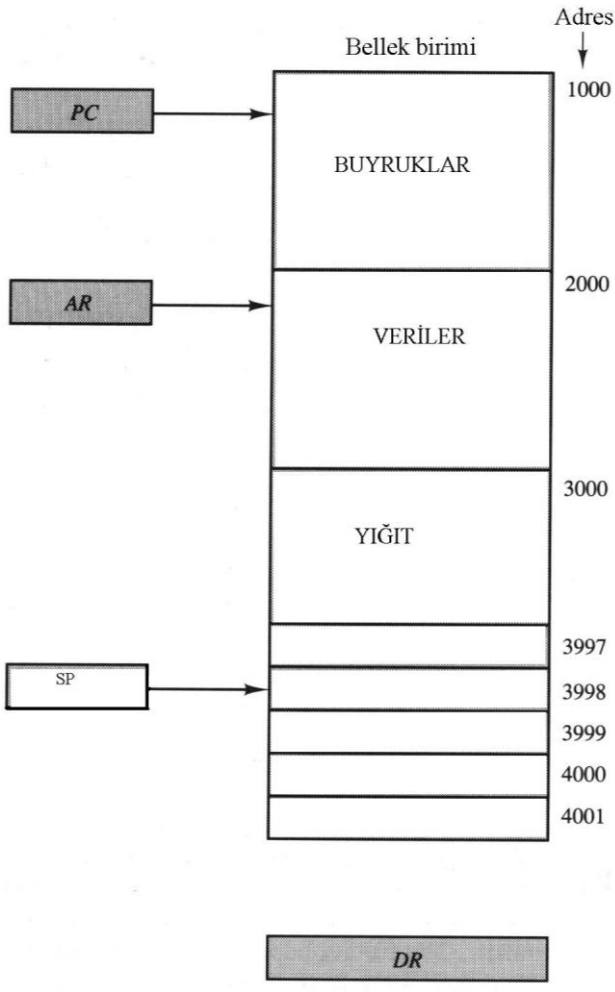
Bellek Yığıtı

Bir yığıt şekil 8.3'te olduğu gibi tek başına, bir birim olarak bulunabilir veya MİB ile ilişkili olarak RAM da tutulabilir. Bunun için belli bir bellek bölgesi yığıt olarak ayrılır ve *SP* olarak bir işlemci yazacı tahsis edilir.



Şekil 8.3 Bir 64 kelimelik yığıtın blok şeması

Bu işlemci yazacı bu işe özeldir, başka amaçla kullanılmaz. Şekil 8.4 üç bölüme ayrılmış bilgisayar hafızasının bir bölümünü göstermektedir. (Bu bölümler program, veri ve yığıt). *PC* programda bir sonraki buyruğun adresini gösterir. *AR* adres yazacı bir sonraki veriyi gösterir. *SP* yığıtın en üstünü gösterir. Bu üç yazaç ortak adres veri yolu ile ilişkilidir ve herhangi biri, bellek için bir adres sağlar. *PC* al-getir evresi boyunca bir buyruk okumak için kullanılır. *AR* çalışma evresi boyunca bir veri okumak için kullanılır. *SP* yığıta atılacak veya yığıttan alınacak verilerin itme yada içme işleminde kullanılır.



Şekil 8.4 Program, veri ve yığıt kesimleri ile bilgisayar hafızası

Şekil 8.4 te görüldüğü gibi *SP* nin başlangıç değeri 4001 ve yığıt adresleri azaltarak ilerlemekte. Buna göre birinci veri girildiğinde değeri 4000, ikinci veri girildiğinde adres değeri 3999 ve en son veride adres 3000 değerindedir. Yığıt limit denetimi için uygun koşul yoktur.

Yığıttaki verilerin *DR* ile ilişkili olduğu varsayılsın. O zaman yeni bir verinin itme işlemi

$$SP \leftarrow SP\text{-}1$$

$$M[SP] \leftarrow DR$$

şeklinde olur. *SP* azaltılarak sonraki kelimenin adresi gösterilir. Belleğe yazma işlemi *DR* den yığıtın en üstüne yazma işlemidir. Silme işlemi de

$$DR \leftarrow M[SP]$$

$$SP \leftarrow SP + 1$$

en üstteki veri yığıttan *DR* nin içini aktarılır. *SP* arttırılarak yığıttaki bir sonraki veriyi gösterir.

Yığıt Limitleri

Çoğu bilgisayarlar yığıt taşması (dolu yığıt) veya yığıt bitmesi (boş yığıt) denetimini donanım ile sağlamazlar. Yığıt limitleri için iki işlemci yazacı kullanarak denetim yapılır. Yazacın birisi üst limiti (burada 3000) ve diğer yığıtta alt limiti (burada 4000) tutar. Bir sonraki itme işleminde *SP* üst-limit yazacı ile ve çekme işleminde de *SP* alt-limit yazacı ile karşılaştırılır.

İtme yada çekmeden herhangi biri için ihtiyaç duyulan iki mikro işlem (1) *SP* vasıtasıyla belleğe giriş, (2) *SP* yi güncelleştirme. Şekil 8.4 te bellek adresi küçülerek yığıt büyümekte, şekil 8.3 te bellek adresi artarak yığıt büyümektedir. Bu durumda itme işleminde *SP* artmakta, çekme işleminde *SP* azalmaktadır. *SP* bir sonraki boş yeri (en üsttekini) göstermektedir. Bu durumda mikro işlemlerin sırası değişebilir.

Bir yığıt göstergesine bir başlangıç değeri yüklenir. Buraya yığıt başlangıç değeri olarak en alttaki adres atanır. Bundan sonra *SP* değeri otomatik olarak her itme veya çekme işlemi ile azaltılır veya arttırılır.

Zıt Polonyalı Gösterimi

Bir yığıt kurulumu aritmetik ifadelerin değerlendirilmesinde oldukça etkilidir. Matematiksel ifadeler genel yazıldığında formüllerin bilgisayar ile değerlendirilmesi zordur. Genel aritmetik ifadeler, veriler arasına birer operatör yazarak elde edilir.

$$A * B + C * D$$

basit aritmetik ifade ele alınsın. Çarpım işlemini gösteren yıldız *A* ile *B* ve *C* ile *D* arasında yer alır. + ise bu iki çarpımın arasındadır. Bu aritmetik ifadenin değerlendirilmesi için $A * B$ çarpımını hesaplamak ve $C * D$ çarpımını hesaplarken onu saklamak ve sonrada bu iki çarpımı toplamak gerekir. Bu örnekten, aritmetik ifadeleri değerlendirmek için, çalıştırılarak bir sonraki işleme karar verirken ifade üzerinde ileri ve geriye göz gezdirmenin gerekli olduğu görülür.

RPN

Polonyalı matematikçi Lukasiewicz'e göre aritmetik ifadeler ön ek gösteriminde belirtilebilir. Bu gösterim polish gösterimi olarak ta bilinir ki burada verilerden önce operatörler yer alır. Zıt Polonyalı Gösterimine karşılık gelen art ek gösteriminde verilerden sonra operatörler yer alır. Üç gösterim için örnek verilirse

$A + B$	Matematikte
$+ AB$	Ön ek veya polish gösterimi
$AB +$	Art ek veya zıt polonyalı gösterimi

Zıt polonyalı yöntemi, yığıt işletimi için uygun bir yöntemdir.

$$A * B + C * D$$

ifadesi zıt polonyalı gösteriminde

$$AB * CD * +$$

biçiminde yazılır ve şu şekilde değerlendirilir. İfade soldan sağa doğru aratılır, bir operatöre ulaşıldığında, operatörün solunda yer alan iki veri çalıştırılır. Bu iki veri ve operatör kaldırılarak aynı işlem tekrarlanır. Elde edilen sonuçlar birer veridir. Bu tekrarlama işlemleri operatör kalmayıncaya kadar yapılır.

Yukarıdaki ifadede *A* ve *B* den sonra * operatörü bulununca $A * B$ işlemi yerine getirilir. elde edilen büyüklük ($A * B$) değeridir. ve sonra diğer veriler operatör bulununcaya kadar sağa doğru izlenir. *C D* den sonra * ya rastlandığında ($C * D$) elde edilir. Buda

$$(A * B)\ (C * D) +$$

ifadesidir. Bir sonraki operatör + ve eklenen iki veride iki çarpımdır. Sonucu elde etmek için iki veri toplanır.

RPN ye Dönüştürme

Zıt polonyalı gösterimine dönüştürmeden önce matematikteki işlemsel hiyerarşiye dikkat edilmelidir. Matematikte ilk parantez içleri sonra parantez dışındakiler işleme girmekte, toplama ve çıkarmadan önce, çarpma ve bölme işlemleri yapılmaktadır.

$$(A + B) * [C * (D + E) + F]$$

ifadesi ele alınsın. İfade değerlendirilirken ilk önce $(A + B)$ ve $(D + E)$ parantez içleri hesaplanır. Daha sonra köşeli parantez içi hesaplanır. $C * (D + E)$ çarpımı F in toplanmasından öncelikli olarak yapılmalıdır, çünkü çarpmanın toplamaya göre önceliği vardır. Son işlem parantez ile köşeli parantez arasındaki çarpma işlemidir.

Hiyerarşi operasyonunu ele alarak parantezleri kullanmaksızın bu ifade zıt polonyalı gösterimine dönüştürüldüğünde

$$AB + DE + C * F + *$$

dır.

Soldan sağa işlem yaparak önce A, B sonra D, E toplanır. Toplamdan elde edilen sonuçlar olduğu yerde kalır.

$$(A + B)\ (D + E)\ C * F + *$$

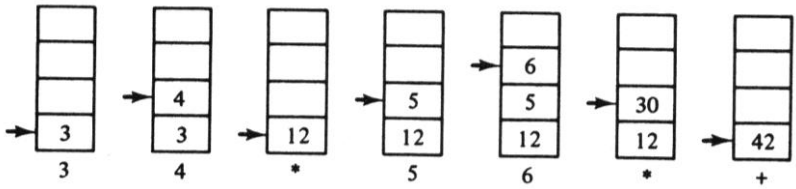
Bir sonraki çarpanın verisi C ve $(D + E)$ dir. Bu iki sayı çarpılır ve F e eklenir. Son çarpma $(A + B)$ ile $C * (D + E) + F$ verilerinin çarpımını sağlar.

Aritmetik İfadelerin Değerlendirilmesi

Zıt polonyalı gösterimindeki matematiksel ifadelerin değerlendirilmesi için en etkili yol yazaçların bir yığıt düzenlemesi ile bulunmasıdır. Bu yöntem bazı elektronik hesap makinelerinde ve bazı bilgisayarlarda kullanılır. Bu yığıt özellikle zincir hesaplamaları (yani birbirlerine bağlı hesaplamalar) içeren uzun ve karmaşık problemler ele alındığında yararlıdır. Buda, herhangi aritmetik ifadenin polish gösteriminde parantezden bağımsız ifade edilebileceği gerçeğine dayandırılır.

Önce matematiksel işlem zıt polonyalı gösterimine dönüştürülür. Veriler göründükleri sırada yığıta yerleştirilir. İşlemin başlangıcı hesap makinesi yada bilgisayara göre değişir. Bir hesap makinesinde, operatörler klavyeden girilir. Bir bilgisayarda ise işlem sahası içeren buyruklar tarafından başlatılır (adres alanı istenmez).

Bir işlem bir hesap makinesine girildiği zaman, yada bir bilgisayarda denetim vasıtasıyla çalıştığı zaman aşağıdaki mikro işlemler yığıtta çalıştırılır.(1)Yığıtın en üstteki iki veri işlem için kullanılır, (2) Yığıt çekilir. İşlem sonucu üstteki veri ile yer değiştirir. Verileri sürekli olarak yığıt içine iterek ve işlemleri yukarıdaki gibi çalıştırarak, ifadeler uygun sırada değerlendirilir ve sonuç yığıtın en üstünde kalır.



Şekil 8.5 3 ∗ 4 + 5 ∗ 6 için yığıt işlemleri

Yığıt İşlemleri

Aşağıdaki sayısal örnek bu işlemleri açıklamaktadır.

$$(3 * 4) + (5 * 6)$$

aritmetik ifadesi ele alındığında, zıt polonyalı yönteminde bu ifade

$$34 * 56 * +$$

şeklinde gösterilir. Şekil 8.5 te gösterilen yığıt işlemleri ele alındığında, her kutu bir yığıt işlemini göstermekte ve ok her zaman yığıtın en üstünü işaret etmektedir. İfade soldan sağa gözden geçirildiğinde veriler iki, iki bulunmaktadır. İlk olarak 3 sayısı ve sonrada 4 sayısı yığıta itilir. Bir sonraki sembol ∗ dır. Bu yığıttaki en üst iki parçanın çarpımına sebep olur. Sonra yığıt çekilir ve çarpım yığıtın en üstünde yer alır. Sonra 5 ve 6 verilerine rastlanmakta, bunun için bu veriler yığıta itilir. Bir sonraki ∗ dan sonuçlanan yığıt işlemi bu iki sayının çarpımıyla, bu iki sayının yerini değiştirir (yani bu iki sayı yerine çarpımlarını getirir). Son işlem, 42 sonucunu üreten yığıtın en üstündeki iki sayının aritmetik toplamına sebep olur.

Bir iç yığıt kullanan bilimsel hesap makineleri kullanıcıdan, aritmetik ifadeleri zıt polonyalı gösterimlere çevirmesini bekler. Bir yığıt MİB kurulumu kullanan bilgisayarlarda kullanıcı için bu dönüşümü yapacak bir sistem programı vardır. Bir çok derleyici, MİB kurulumlarına bakmaksızın, aritmetik ifadeleri polish gösterimlerine çevirirler. Çünkü bu yöntem aritmetik ifadeleri birleştirilmiş dil komutlarına çevirmek için en etkili yoldur. Bunun için bir yığıt MİB kurulumu (yığıt kurulumu yapılmış MİB) yığıtsız bir MİB den başka uygulamalarda daha etkili olabilir.

8.4 Buyruk Biçimleri

Bilgisayarların fiziksel ve mantıksal yapıları bilgisayarla beraber verilen referans el kitaplarında bulunmaktadır. Bu elkitapları MİB in iç şartlarını, mevcut yazaçları ve bunların mantıksal yeteneklerini belirtir. Bütün donanımla gerçekleştirilebilecek buyrukları, bunların ikili kod biçimlerini

ve her buyruğun kesin tanımını burada bulmak mümkündür. Bir bilgisayarda çeşitli buyruk biçimleri mevcuttur. Bu buyrukların açıklanması ve buyruk icra edilmesi için gerekli denetim fonksiyonlarının üretilmesi MİB içindeki denetim biriminin görevidir.

Bir buyruğun biçimi bir dikdörtgen içinde verilir. Dikdörtgen buyruğun bitlerini gösterir. Bellek kelimesi ve denetim yazacı böyle gösterilir. Bir buyruğun bitleri alanlar adı verilen gruplara ayrılır. Bir buyruk biçiminde bulunan alanlar;

1. İcra edilecek işlemi gösteren OPR kod alanı
2. Bir adres alanı: Bellek alanını veya bir işlemci yazacını gösterir
3. Bir kip alanı: Verinin yolunu veya etkin adresin bulunduğunu gösterir

Bazı diğer özel alanlar belli durumlarda kullanılır. Örneğin kaydırma komutu içinde kaymaların miktarını veren bir alan.

Bir buyruk içinde OPR kod alanı bir grup bitten oluşan bir alandır. Bunlarla toplama, çıkarma, tümleme, kaydırma gibi çeşitli işlemci işlemleri tanımlanır. Bilgisayar buyrukları içinde ortak olan bölüm 8.6 da açıklanmıştır. Bir buyruğun kip alanındaki bitler, bir adres alanından verinin seçimi için birçok alternatif belirtir. Sayısal bilgisayarlar için çeşitli adresleme yöntemleri bulunmuş olup bölüm 8.5 te açıklanmıştır. Bu kısımda bir buyruğun adres alanı ile ilgilenilmektedir. Ayrıca bir buyruk içine birçok adres alanı yazmanın etkisi incelenecektir.

Yazaç Adresi

Bir bilgisayarın buyrukların icrası bazı veriler üzerinde olur. Bunlar bellek veya işlemci yazaçlarında bulunurlar. Bellek içinde bulunan veriler yazaç adresiyle bellidir. Bir yazaç adresi k bitlik bir ikili sayıdır. MİB içindeki 2^k tane yazaçtan bir tanesini belirtir. Eğer MİB da $R0$ dan başlayıp $R15$ e kadar giden 16 tane yazaç varsa yazaç adresleri için 4 bit yeter. Örneğin 0110 adresi $R5$'i gösterir

Bilgisayarların buyrukları muhtelif uzunluklarda olabilir. Veri adresleri de muhtelif uzunluklarda olabilir. Adres alanının uzunluğu, yazaçlarının iç kurulumuna bağlıdır. Bilgisayarların çoğu 3 farklı MİB kurulumuna bağlıdır.

1. Tek *AC* kurulumu
2. Genel yazaç kurulumu
3. Yığıt kurulumu

Bölüm 5'te *AC* tipi kurulum anlatıldı. Bütün işlemler bir AC yazacı yardımıyla yapıldı. Bu tip kurulumda buyruk biçimi içinde bir tane adres alanı bulunur. Örneğin birleştirici dil buyruğu

```
ADD     X
```

içindeki X verinin adresidir. Bu işlem $AC \leftarrow AC + M[X]$ ile verilir. Genel yazaç kurulumunda bir örnek şekil 7.1 de gösterilmiştir. Bu buyruk biçiminde 3 yazaç adres alanı gerekir. Böyle bir kurulumda aritmetik toplama işlemi birleştirici dilinde

```
ADD     R1, R2, R3
```

şeklinde verilir ve $R1 \leftarrow R2 + R3$ ile açıklanır. Eğer hedef yazaç, iki kaynak yazaçtan biri ise yazaç sayısı 2 ye indirilebilir.

```
ADD     R1, R2
```

ve yazaç aktarım diliyle $R1 \leftarrow R1 + R2$.

Çok sayıda yazacı bulunan bilgisayarlarda aktarma buyruğu MOV sembolik koduyla verilir.

```
MOV     R1, R2
```

yani ya $R1 \leftarrow R2$ veya $R2 \leftarrow R1$ olur. Ayrım bilgisayara bağlıdır.

Genel yazaç tipi bilgisayarlarda 2 veya 3 adres alanı bir işlemci yazacı veya bellek kelimesini gösterir. Aşağıdaki buyruk

```
ADD     R1, X
```

$R1 \leftarrow R1 + M[X]$ işlemini gösterir. İki adres alanı var. Biri işlemci yazacı $R1$'i, diğeri bellek adresi X i gösterir.

Yığıt kurulumlu MİB şekil 8.4 te gösterilmişti. Bu tip bilgisayarlar PUSH ve POP buyruklarına sahiptir ve bir tek adres alanı gerektirirler. Örneğin

```
PUSH    X
```

buyruğu X adresindeki kelimeyi yığıtın tepesine itmektedir. Yığıt göstergesi otomatik olarak değişir. İşlem tipi buyrukların adrese gereksinimleri yoktur. Çünkü işlemler daima yığıtın tepesindeki iki veri arasında yapılır. Buyruk

```
ADD
```

içinde sadece işlem kodu vardır ve adres alanı yoktur. Toplama işlemi iki tepe verisi arasında yapılır ve sonuç yığıta itilir.

Bilgisayarlar bu gruplardan birine girdiği gibi bazıları (şimdi hepsi) bu özellikleri birleştirerek kullanırlar. Örneğin intel 8080 işlemcisinin 7 MİB yazacı mevcut olup bir tanesi AC yazacı olarak kullanılır. Dolayısıyla mikro işlemci hem genel yazaç tipi hem de AC tipinin özelliklerini taşır. Bütün aritmetik, mantık, yükleme ve saklama gibi buyruklar AC yi kullanır. Dolayısıyla bu buyrukların tek bir adres alanı vardır. Diğer taraftan işlemci yazaçları arasında veri aktarımı yapan buyruklar iki adres

alanı biçimini taşırlar. Bundan başka 8080 işlemcisinin bellekte bir yığıtı vardır ve buna itme ve çekme yapmak için yığıt göstergesi de vardır. Bunlara karşılık yığıt tipi işlemcilerin karakteristiği olan sıfır adres tipi komutları yoktur.

Buyruklar içindeki adres sayısının etkisini incelemek için aşağıdaki aritmetik komuta bakılırsa,

$$X = (A + B) * (C + D)$$

bunu çalıştırmak için sıfır, bir, iki ve üç adres ve ADD, SUB, MUL, DIV işlemlerini, MOV aktarımını, LOAD ve STORE komutlarında ise bellekle *AC* yazacı arasındaki aktarımları kullanalım. Veriler bellekte *A*, *B*, *C*, *D* adreslerindedir. Sonuç *X* adresine yazılacaktır.

Üç Adresli Buyruklar

Üç adres buyruklu bilgisayarlarda bu adresler bellek adresleri veya işlemci yazaç adresleri olabilir. $X = (A + B) * (C + D)$ yi yapan birleştirici dil programı.

```
ADD     R1, A, B      R1 ← M[A] + M[B]
ADD     R2, C, D      R2 ← M[C] + M[D]
MUL     X, R1, R2     M[X] ← R1 * R2
```

Bilgisayarın iki işlemci yazacının mevcut olduğu varsayıldı. Bu 3 adresli sistemin kazancı aritmetik işlemler için kısa programlar yazılabilmesidir. Zararı ise 3 adresin yazılabilmesi için çok bit gerekmesidir. İki adresli buyruk kullanan ticari bilgisayarlara bir örnek cyber 170 verilebilir. Cyber bilgisayarların buyruk biçimleri 3 yazaç adres alanından veya iki yazaç adres alanı ile bir bellek adres alanından oluşmaktadır.

İki Adresli Buyruklar

Ticari bilgisayarlarda en çok bunlar vardır. Adresler bellek veya işlemci yazacı adresi olabilir. $X = (A + B) * (C + D)$ işlemini yapan birleştirici dil programı.

```
MOV     R1 A        R1 ← M[A]
ADD     R1,B        R1 ← R1 + M[B]
MOV     R2,C        R2 ← M[C]
ADD     R2,D        R2 ← R2 + M[D]
MUL     R1,R2       R1 ← R1 * R2
MOV     X, R1       M[X] ← R1
```

Birinci adresin hem kaynak, hem de hedef yazacı olduğu varsayılıyor.

MOV buyruğu veriyi bellekten işlemci yazacına aktarır.

Bir Adresli Buyruklar

Bir adresli buyruklar tüm veri işlemleri için *AC* yi kullanırlar.Çarpma ve bölme işlemleri için ikinci bir yazaca ihtiyaç vardır. Buna rağmen ikinci bir yazaç ihmal edilip bütün işlemler *AC* de yapılır ve sonuçlar *AC* de saklanır.

```
LOAD    A     AC ← M[A]
ADD     B     AC ← A + M[B]
STORE   T     M[T] ← AC
LOAD    C     AC ← M[C]
ADD     D     AC ← AC + M[D]
MUL     T     AC ← AC * M[T]
STORE   X     M[X] ← AC
```

Bütün işlemler *AC* yazacı ile bellekteki veriler arasındadır. *T* bellek adresi ara değerin saklanması içindir.

Sıfır Adresli Buyruklar

Yığıt kurulumlu bir bilgisayarda buyrukların adrese gereksinimi yoktur. ADD, MUL, PUSH ve POP adressizdir, fakat yığıtla bellek haberleşmesi için bellek adreslerine gerek vardır. (TOS = yığıt göstergesi). $X = (A + B) * (C + D)$ işleminin yığıt kurulumlu bilgisayarda birleştirici dil programı.

```
PUSH    A       TOS ← A
PUSH    B       TOS ← B
ADD             TOS ← (A + B)
PUSH    C       TOS ← C
PUSH    D       TOS ← D
ADD             TOS ← (C + D)
MUL             TOS ← (C + D) * (A + B)
POP     X       M[X] ← TOS
```

Aritmetik işlemlerin yığıt kurulumlu bir bilgisayarda yapılabilmesi için Polish gösterimine çevrilmesi gerekir.

RISC Buyrukları

RISC makinesinin yararları 8.8 de anlatılacaktır. Tipik bir RISC buyruk kümesi, bellekle MİB arasındaki aktarımlar için LOAD ve STORE buyrukları ile kısıtlıdır. Diğer bütün buyruklar MİB içinde icra edilir. Yani buyruklar içinde bellek adreslemeli buyruk yoktur. RISC için yazılmış bir programda LOAD ve STORE buyrukları bir bellek ve bir yazaç adresi içerir. Hesaplama tipi buyruklar ise 3 adresli buyruk olup, adreslerin hepsi yazaç adresleridir. Aşağıdaki program $X = (A + B) * (C + D)$ yi bulur.

```
LOAD     R1, A          R1 ← M[A]
LOAD     R2, B          R2 ← M[B]
LOAD     R3, C          R3 ← M[C]
LOAD     R4, D          R4 ← M[C]
ADD      R1, R1, R2     R1 ← R1 + R2
ADD      R3, R3, R4     R3 ← R3 + R4
MUL      R1, R1, R3     R1 ← R1 * R3
STORE    X, R1          M[X] ← R1
```

Yükleme buyruğu verileri bellekten MİB yazaçlarına aktarır. Toplama ve çarpma buyrukları veriler üzerinde belirtilen matematiksel işlemleri yapar ve depolama buyruğu hesaplanan sonucu hafızada saklama yazacına aktarır.

8.5 Adresleme Kipleri

Bir buyruğun OPR alanı icra edilecek işlemi belirler. Bu işlem bazı veriler üzerinde yapılır. Bu veriler yazaçlarda veya bellekte bulunur. Programın icrası sırasında verinin seçilme şekli, buyruğun adresleme kipine bağlıdır. Adresleme kipi verinin bulunması ve getirilmesinden önce buyruğun içindeki adres alanının (veya adresin) nasıl değiştirileceği hakkında bir kural verir. Bu adresleme kipinin örneği temel bilgisayarda I bitidir. Bu bit sayesinde adres ya doğrudan kullanılıyor veya adresle asıl adres bulunuyor (dolaylı). Bilgisayarlarda aşağıdaki özellikleri sağlamak için adresleme kipi teknikleri kullanılır.

1. Kullanıcıya göstergeler, sayıcılar, indisleme ve yeniden yükleme (bilgisayar belleğinin herhangi bir yerinde çalıştırılabilecek şekilde kodlanmış program) gibi faydalı yöntemler kazandırarak programların kolay olmasını sağlamak.
2. Buyruğun adres alanındaki bit sayısını azaltmak.

Deneyimsiz bir kullanıcı için bazı bilgisayarlardaki adresleme kipleri çok karışık gözükür. Deneyimli bir kullanıcı için ise program yazımında etkinlik, kısalık, buyrukları etkin kullanma ve zamanda kısalık sağlar.

Bu bölümde anlatılan değişik adresleme kiplerini anlamak için bilgisayarın temel işleyiş döngüsünü anlamak zorunludur. Bilgisayarın denetim birimi üç ana aşamaya bölünmüş bir buyruk döngüsünden geçecek şekilde tasarlanmıştır:

1. Bellekten buyruğun getirilmesi
2. Buyruğun kodunun çözülmesi
3. Buyruğun icrası

Program Sayıcı (PC)

Bilgisayarda program sayıcı yada bellekte depolanmış program buyruklarının izini süren PC diye adlandırılan bir yazaç vardır. PC bir sonra icra edilecek buyruğun adresini tutar ve bellekten her buyruk getirilişinde değeri 1 arttırılır. İkinci adımdaki kod çözme işlemi, yapılacak işlemi, buyruğun adresleme kipini ve işlenenlerin yerini belirler. Ardından bilgisayar buyruğu icra eder ve sıradaki buyruğu getirmek için birinci adıma geri döner.

Bazı bilgisayarlarda tıpkı işlem kodunun belirtildiği gibi buyruğun adresleme kipi farklı bir ikili kod ile belirtilir. Diğer bilgisayarlar hem işlemi hem de buyruğun kipini belirten tek bir ikili kod kullanılırlar. Buyruklar değişik adresleme kipleri ile tanımlanabilirler ve bazen iki yada daha fazla adresleme kipi bir tek buyrukta birleştirilir.

Kip Alanı

Farklı adresleme kip alanına sahip bir buyruk biçimi örneği şekil 8.6 da gösterilmektedir. İşlem kodu yapılacak işlemi belirtmektedir. Kip alanı ,işlem için gerekli işlenenlerin yerini saptamak için kullanılır. Buyrukta bir adres alanı olabilir yada olmayabilir. Eğer bir adres alanı varsa bu bir bellek adresi yada bir işlemci yazacı olabilir. Bundan başka bir önceki bölümde anlatıldığı gibi buyruk birden fazla adres alanına sahip olabilir ve her adres alanı kendi özel adresleme kipi ile ilişkilendirilmiş olabilir.

Her ne kadar adresleme kipleri buyruğun adres alanını değiştiriyor olsa da, 2 adresleme kipinde hiç adrese gerek yoktur. Bunlar ima edilmiş ve doğrudan kiplerdir.

***İma Edilmiş Kipi*:** Bu kipte veriler buyruğun tanımında kapalı olarak verilmektedir. Örneğin *AC* yi tümle bu türden bir buyruktur. *AC* kullanan bilgisayarlarda bütün yazaç adreslemeli buyruklar ima edilmiş kiptedirler. Yığıt tipi bilgisayarlarda ise sıfır adresli buyruklar ima edilmiş kiptedir. Çünkü veriler yığıtın en üst kısmında bellidir.

Derhal Kipi: Bu kipte veri buyruğun içinde verilmiştir. Yani adres alan, yerine buyruğun içinde veri alanı vardır. Bu buyrukları yazaçlara başlangıç değerlerinin verilmesi ve sabit değer yazılması için uygundur.

Bir buyruğun adres alanı ya bellek adresini, yada bir yazaç adresini göstermektedir. Eğer adres alanı bir işlemci yazacını işaret ediyorsa bu buyruk yazaç kipindedir denir.

***Yazaç Kipi*:** Bu kipte veriler MİB içindeki yazaçlar içindedirler. Hangi yazacın adreslendiği k bit ile bellidir (2^k yazaçtan).

***Yazaç Dolaylı Kip*:** Bu kipte adres alanında bir yazacın adresi vardır. yazacın içinde ise verinin bellekte bulunduğu adres vardır. Bu buyruğu kullanmadan önce yazaç içinde istenen adresin bulunduğuna emin olunmalıdır. Bu iş ancak daha önceki bir buyruk ile gerçekleştirilebilir. Bu kipin kazancı yazaç adresinin çok kısa olabilmesidir.

***Otomatik Artma veya Otomatik Azalma Kipi*:** Bu kip yazaç dolaylı kipe çok benzer. Ancak yazaç içindeki bellek adresi kullanılmadan önce veya kullanıldıktan sonra arttırılır veya azaltılır. Bu yöntem tablolar için kullanılır. Yazaca yerleştirilen adres bir tabloyu göstermektedir. Böylece tablonun her elemanına sırayla erişilir. Bazı bilgisayarlarda, içindeki veriye erişilince otomatik artan veya azalan yazaçlar vardır.

Bir buyruğun içindeki adres alanı, MİB içindeki denetim birimi tarafından kullanılarak bellekten veri elde edilir. Bazen buyruğun adres alanı içinde verinin adresi bulunur. Bazen buyruk içinde verilen adres yardımıyla verinin adresi hesaplanır. Muhtelif adresleme kiplerini birbirinden ayırmak için buyruğun içindeki adres kısmı ile etkin adresin birbirinden ayrılması gerekir.

Etkin Adres

Buyruk içinde verilen adresten ve adresleme kipinden elde edilen bellek adresidir, hesaplama tipi buyruklarda verinin adresidir, kontrol-dallanma buyruklarında dallanılan adrestir.

Daha önce belirtilen iki tip adresleme kipine tekrar bakılırsa;

***Doğrudan Adres Kipi*:** Bu kipte etkin adres, buyruğun adres alanındaki adrestir. Veri bellekte, adresi buyruğun içindedir. Dallanma tipi bir buyrukta gerçek dallanma adresi verilir.

***Dolaylı Adres Kipi*:**Bu kipte buyruğun adres alanı içindeki adres, verinin bellekteki gerçek adresinin bulunduğu yerin adresidir. Denetim, buyruğu bellekten alır getirir. Bunun adresini alır ve tekrar belleğe aldığı gider ve etkin adresi okur. Dolaylı adres kipi bölüm 5.1 de açıklandı.

Birkaç tane adresleme kipi buyruğun adresinin, MİB içindeki belirli bir yazacın içeriğine eklenmesini gerektirir. Bu kiplerde etkin adres

$$\text{etkin adres} = \text{buyruğun adres kısmı} + \text{MİB yazacının içeriği}$$

ile elde edilir. Bu MİB yazacı, bir sayıcı veya indis yazacı veya bir temel yazaç olabilir. Her bir durum farklı adresleme kipi olup farklı uygulamalar için kullanılır.

***Göreceli Adres Kipi*:** Bu kipte program sayıcının içeriği buyruğun adres kısmına eklenir ve etkin adres elde edilir. Buyruğun adres kısmı genellikle işaretli bir sayıdır (2 nin tümleyeni biçiminde verilmiştir). Pozitif veya negatif olabilir. Bu sayı PC ın değerine eklendiğinde bir sonraki buyruğun adresine göre göreceli bir bellek adresi bulunmuş olur. Bir örnek ile açıklanırsa; PC ın içeriği 825 olsun. Buyruğun adres kısmında 24 bulunsun. 825 adresindeki buyruk okunur ve PC 826 ya arttırılır. Göreceli adres kipinde bulunulduğundan 826 + 24 = 850 olup, bir sonraki buyruktan 24 bellek birimi uzaklıktadır. Bu kip, eğer dallanılacak yer, buyruk civarında ise dallanma için kullanılır. Faydası uzun adres yerine buyruğun içine kısa adres yazılmasıdır.

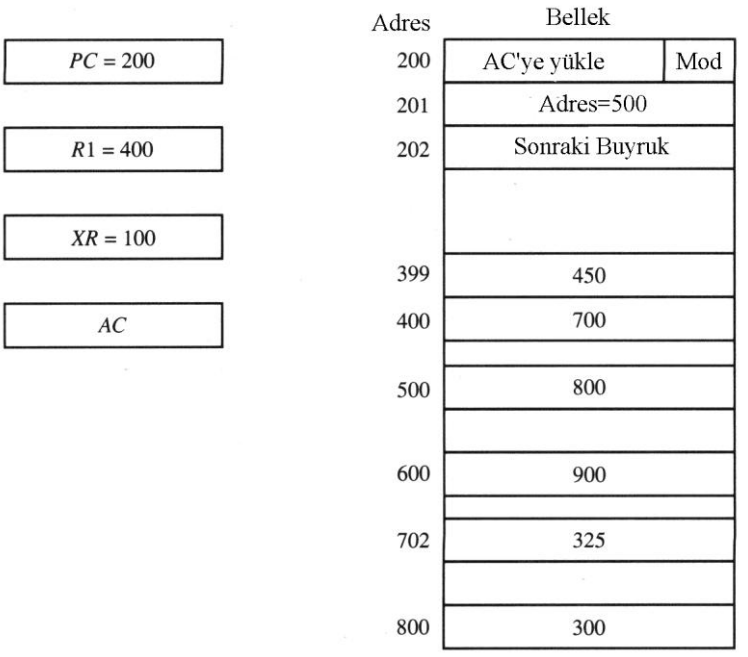
***İndislenmiş Adres Kipi*:** Bu kipte indis yazacının içeriği buyruğun adres değerine eklenir. İndis yazacı özel bir MİB yazacı olup bir indis değeri içerir. Buyruk içindeki adres değeri çizelgenin başlangıç adresidir. Dizideki her veri bellekle başlangıç adresine göreceli bir konumda bulunur. Başlangıç adresi ile verinin adresi arasındaki fark indis yazacında bulunan değerdir. Dizideki her veri aynı buyruk ile erişilir. Yeter ki indis yazacında doğru indis değeri bulunsun. İndis yazacının değeri ardışık verilere erişmek için arttırılabilir. Eğer indis tipi bir buyruğun içinde adres alanı bulunmuyorsa buyruk bu komutu yazaç-dolaylı kip gibi kullanır.

Bazı bilgisayarlarda bir MİB yazacı sadece ve yalnız indis yazacı olarak ayrılmıştır. İndis kipi kullanıldığında bu yazaç devreye girer. Çok sayıda yazacı bulunan bilgisayarlarda MİB yazaçlarından herhangi biri bir indis içerebilir. Bu durumda buyruk içindeki yazaç alanında hangi yazacın bu işi yapacağı açık biçimde belirtilmelidir.

***Taban Yazaç Adres Kipi*:** Bu kipte taban yazacının içeriği buyruğun adres değerine eklenir ve etkin adres elde edilir. Bu indis yazaç kipinin aynıdır. Tek farkı taban yazaç değerinin eklenmesidir. İkisinin farkı ise kullanımdadır. Taban adresi, buyruğun verisinin adresini taban adresine göre göreceli verir. Taban adres kipi programların yeniden konumlandırılmasında, kesim değişimlerinde, çok kullanıcılı sistemlerde mantıksal adres, gerçek adres dönüşümlerinde kullanılır.Aynı ofset adresi farklı kesimlerde kullanılır.

Sayısal Örnek

Muhtelif adresleme kipleri arasındaki farkı göstermek için, şekil 8.7 deki buyruklar üzerindeki etkisi incelenirse, iki kelimelik buyruk bellekte 200 ve 201 adresinde bulunuyor. “LOAD TO AC” buyruğu 500 adresini içeriyor (yani adres alanında 500 adresi var). Buyruğun ilk kelimesi OPR kodunu ve kipi veriyor. İkinci kelime ise adresi içeriyor. *PC* de buyruğu



Şekil 8.7 Adresleme kipleri için sayısal örnek

al-getir evresi için 200 adresi var. İşlemci yazacı *R*1 de 400 var, ve indis yazacı *XR* de 100 mevcut. Buyruğun icrasından sonra *AC* veriyi alır. Şekilde adreslerin içeriği ve bazı adresler var.

Buyruk kip alanında kiplerden herhangi biri olabilir. Verilen kipe göre etkin adresi hesaplanacak ve veri *AC* ye yüklenecek. Doğrudan adres kipinde etkin adres, buyruğun adres kısmıdır. Bu durumda *AC* ye 800 değeri yüklenir. Derhal kipte buyruğun ikinci kelimesi adres değil verinin kendisidir. *AC* ye 500 yüklenir. Bu kipte etkin adres 201 dir. Dolaylı kipte etkin adres bellek adresi 500 de bulunmaktadır. Burada 800 vardır ve .veri 800 de olup 300'dür. Göreceli kipte etkin adres 500 + 202 = 702 olup veri 325'tir. Al-getir evresi ve icra evresi sırasında *PC* nin içeriği 202'dir. İndis kipinde etkin adres *XR* + 500 = 100 + 500 = 600 ve dolayısıyla veri 900'dür. Yazaç kipinde veri *R*1 dedir ve 400 *AC* ye yüklenir (Bu kipte etkin adres yoktur). Yazaç dolaylı kipinde etkin adres 400 dür, yani *R*1 in içeriğidir. *AC* ye yüklenen değer 700 dür. Otomatik artma kipi aslında yazaç dolaylı kipin aynıdır. Yalnızca *R*1 yazacı buyruk icrasından sonra 401 olur. Otomatik azalma kipinde R1 içindeki değer icradan önce 1 azaltılır yani 399 olur. Bu durumda *AC* ye yüklenecek veri 450 olur. Çizelge 8.4 dokuz adresleme kipi ve etkin adreslerle verileri göstermektedir.

Çizelge 8.4 Nümerik örnek listesi

Adres kipi	Etkin adres	*AC* nin içeriği
Doğrudan adres	500	800
Derhal veri	201	500
Dolaylı adres	800	300
Göreceli adres	702	325
indislenmiş adres	600	900
Yazaç	—	400
Yazaç dolaylı	400	700
Otomatik artma	400	700
Otomatik azalma	399	450

8.6 Veri Aktarımı ve İşleme

Temel İşlemler Kümesi

Bilgisayarlar kullanıcılara, muhtelif programlama ve hesaplama işlerini kolay yapabilmeleri için çok geniş buyruk kümesi verirler. Farklı bilgisayarların birbirlerinden farkları verilerinin adreslerini hesaplamaları ve kipleri dır. Buyruk kümesi içinde mevcut gerçek işlemler, bilgisayarlar arasında çok farklı değildir. Buradan anlaşılacağı gibi aynı işlem için bile bilgisayarların OPR kodları farklı farklıdır. Aynı nedenden birleştirici dil kodları da farklı farklı olacaktır. Her şeye rağmen çoğu bilgisayarlarda aynı olan bir Temel işlemler kümesi mevcuttur. Bu kümenin dışında farklı buyruklarda vardır. bu ve sonraki kısımlarda bilgisayarlarda mevcut temel işlemler kümesi anlatılacak.

Bilgisayar buyrukları 3 kategoride sınıflandırılabilir.

1. Veri aktarım buyrukları
2. Veri işleme buyrukları
3. Program denetim buyrukları

Veri aktarım buyrukları verinin bir yerden başka bir yere içeriği değiştirilmeden aktarılmasını sağlar. İşlem buyrukları aritmetik, mantıksal ve kaydırma işlemlerini yerine getirirler. Program denetim buyrukları karar verme imkanı sağlar ve programın icrası sırasında programın yolunu değiştirebilir. Belli bir bilgisayarın buyruk kümesi, yazaç aktarım işlemlerini ve denetim kararlarını kullanıcıya sağlar.

Veri Aktarım Buyrukları

Veri aktarım buyrukları verileri bir yerden başka bir yere içeriğini değiştirmeden gönderirler. En çok kullanılanlar; bellek ile işlemci yazaçları arasında, işlemci yazaçları ile G/Ç arasında ve işlemci yazaçlarının kendi aralarındaki aktarımlardır. Çizelge 8.5 birçok bilgisayarda kullanılan 8

Çizelge 8.5 Veri aktarım buyrukları

İsim	Birleştirici dil sembolleri
Yükle	LD
Aktar	ST
Taşı	MOV
Yer değiştir	XCH
Giriş	IN
Çıkış	OUT
İtme	PUSH
Çekme	POP

buyruğu ve bunların birleştirici dil sembollerini göstermektedir. Farklı bilgisayarlar aynı buyruklar için farklı birleştirici dil sembolleri kullanıyor olabilirler.

Yükle buyruğu genellikle bellekten işlemci yazacına (genellikle *AC* ye) aktarım için kullanılır. Aktar buyruğu işlemci yazacından belleğe aktarım için kullanılır. Taşı buyruğu bir yazaçtan başka yazaca aktarımı sağlar. Bu buyruk aynı zamanda bellekle *MİB* yazacı arasındaki veya iki bellek kelimesi arasındaki aktarım içinde kullanılabilir.

Yer değiştir buyruğu iki yazaç veya bir yazaç ile bir bellek kelimesinin içeriklerini yer değiştirir. Giriş ve çıkış buyrukları işlemci yazacı ile giriş veya çıkıç uçları arasında veri aktarımı yapar. İtme ve çekme buyrukları işlemci yazacı ile bellek yığıtı arasında veri aktarımı yapar.

Çizelge 8.5 teki buyruklar veya bu bölümde diğer çizelgelerdeki buyruklar genellikle bir takım adresleme kipleriyle birlikte düşünülmelidir. Bazı birleştirici dilleri farklı adres kipleri için farklı birleştirici dil sembolleri kullanırlar. Örneğin Yükle buyruğunun doğrudan kipi LDI ile gösterilebilir. Başka birleştirici dil ise adresleme kipini belirtmek için özel karakterler kullanır. Örneğin doğrudan kipi veriden önce yazılan # işareti ile bellidir. Mutlaka ki her buyruğun kodu bir çok adresleme kipi ile beraber yazılabilir. Örnek olarak LDA (*AC* ye yükle) buyruğunun 8 farklı adresleme kipiyle kullanılışını düşünelim. Çizelge 8.6 tavsiye edilen 8 birleştirici dili göstermektedir.

Çizelgede gerçekten meydana gelen aktarımlar da yazılmıştır. *ADR* ile adres gösterilmiş, NBR verinin numarası, *X* indis yazacı, *R*1 işlemci yazacı, *AC* işlemci yazacıdır. @ karakteri dolaylı adrestir. $ işareti adresi *PC* ye göre göreceli yapar. Veriden önce gelen # işareti doğrudan kipi gösterir. İndisli adresleme kipi, sembolik adresten sonra parantez içine yazılan yazaç ile bellidir. Yazaç kipi, yazacın adının yazılmasıyla anlaşılır. Dolaylı yazaç kipinde yazaç adı parantez içine yazılır. Otomatik artma kipi de parantez içinde yazaç adının yanına yazılan + işaretinden anlaşılır.

Çizelge 8.6 Yükleme buyruk için sekiz adresleme kipi

Kip	Kabul edilen birleştirici dil	Yazaç aktarımı
Doğrudan adres	LD ADR	$AC \leftarrow M[ADR]$
Dolaylı adres	LD @ADR	$AC \leftarrow M[M[ADR]]$
Göreceli adres	LD $ADR	$AC \leftarrow M[PC + ADR]$
Derhal veri	LD #NBR	$AC \leftarrow NBR$
İndislenmiş adres	LD ADR(X)	$AC \leftarrow M[ADR + XR]$
Yazaç	LD R1	$AC \leftarrow R1$
Yazaç dolaylı	LD (R1)	$AC \leftarrow M[R1]$
Otomatik artma	LD (R1)+	$AC \leftarrow M[R1], R1 \leftarrow R1 + 1$

Otomatik azalmada eksi işareti konur. Bir bilgisayarda birleştirici dilde program yazabilmek için buyrukların tipini ve adresleme kiplerini bilmek gerekir.

Veri İşleme Buyrukları

Bunlar veriler üzerine işlem yaparak bilgisayarın yetenek kısmını oluştururlar. Tipik bir bilgisayarda 3 tipe ayrılırlar.

1. Aritmetik buyruklar
2. Mantıksal ve bit işleme buyrukları
3. Kaydırma buyrukları

Veri işleme buyruklarının listesi 4. bölümde verilen mikro işlemler listesine benzer. Her bir buyruk icra edildiğinde önce al-getir evresine gitmelidir. Buradan kendi ikili kodunu okur. Veriler ise işlemci yazacına adresleme kipi yardımıyla getirilmelidir. Son adım buyruğun işlemci içinde icrasıdır. Bu son adım bölüm 4 te anlatılan mikro işlemlerle yerine getirilir. İcra işlemi ALU ve kaydırıcı ile de yerine getirilebilir. Bazı aritmetik işlemler özel devreler gerektirir.

Aritmetik Buyruklar

Dört temel işlem için bilgisayarlarda buyruklar vardır. Bazı küçük bilgisayarlarda buyruklar vardır. Bazı küçük bilgisayarlarda sadece toplama ve belki çıkarma buyrukları vardır. Bu durumda çarpma ve bölme yazılımla ve alt programlarla halledilir. 4 temel aritmetik işlem nümerik analiz yöntemleriyle bağlandıklarında bilimsel problemlerin çözümlerinin elde edilmesine yeterli olurlar.

Tipik aritmetik işlemler çizelge 8.7 de verilmiştir. Arttırma buyruğu yazaçtaki veya bellekteki değere 1 ekler. Eğer bu buyruk işlemci yaza-

Çizelge 8.7 Tipik aritmetik buyruklar

İsim	Birleştirici dil sembolleri
Arttır	INC
Azalt	REC
Topla	ADD
Çıkar	SUB
Çarp	MUL
Böl	DIV
Elde ile topla	ADD C
Borç ile çıkar	SUB B
2 ye göre tümleyen al	NEG

cında tümüyle 1 olan bir değere eklenirse, değerin bütün bitleri 0 olur. Azalma buyruğu benzer şekilde 1 çıkartır. Bütün bitleri 0 olan bir sayı 1 azaltılınca bütün bitleri 1 olan bir sayı elde edilir.

Veri Tipi

Topla, çıkar, çarp ve böl buyrukları değişik veri tipleri için mevcuttur. Veri tipinin buyruğun icrası sırasında işlemci yazacında mevcut olduğu ve OPR kodunun tanımında belirtildiği varsayılır. Aritmetik ve buyruğun gösterdiği işlem sabit noktalı veya kayan noktalı, ikili veya onlu, tek duyarlıklı veya çift duyarlıklı olabilir. Bunlar 3. bölümde verilmiştir.

Bilgisayarlarda 3 veya daha fazla toplama işlemi bulmak olasıdır. Birisi ikili tam sayılar için, bir diğeri kayan noktalı veriler için ve bir başkası da ondalık veriler için olabilir. Bu üç buyruk için birleştirici dil sembolleri aşağıdadır.

```
ADDI    2 ikili tamsayıyı toplar
ADDF    2 kayan noktalı sayıyı toplar
ADDD    2 ondalık sayıyı toplar
```

Bu işlemler için algoritmalar bölüm 10 da açıklanacaktır.

Bir yazaç içindeki bitlerin sayısı sınırlıdır. Bu yüzden aritmetik işlemlerin sonuçları sınırlı duyarlıktadır. Bazı bilgisayarlar çift duyarlıklı sayılar için donanıma sahiptirler. Bunlarda her bir verinin uzunluğu 2 bellek kelimesine eşittir. Bir çok küçük bilgisayar çift duyarlıklı aritmetik için özel buyruklar içerir. Özel bir elde yaz-bozu son elde yi tutar. *Elde ile topla* buyruğu iki veriyi topladığı gibi bir önceki buyruktan gelen elde değerini de bunlara toplar. Benzer şekilde *borç ile çıkar* buyruğu iki kelimeden birini diğerinden çıkarır. Aynı zamanda bir önceki buyruktaki ödünç almayı da hesaba katar. NEG buyruğu bir sayının 2'nin tümleyenini alır.

Çizelge 8.8 Mantıksal ve bit işleme buyrukları

İsim	Birleştirici dil sembolleri
Silme	CLR
Tümleme	COM
VE	AND
VEYA	OR
ÖZEL-VEYA	XOR
Elde yi silme	CLRC
Elde yi 1 yapma	SETC
Elde yi tümleme	COMC
Kesmeyi açma	EI
Kesmeyi kapatma	DI

Mantıksal ve Bit İşleme Buyrukları

Mantıksal buyruklar yazaçlara yüklenmiş bitler üzerine işlem yaparlar. Tek tek bitler üzerine işlem yapmakta veya ikili olarak kodlanmış bilgilerin grup halindeki bitlerini işlemekte yararlıdırlar. Mantıksal buyruklar verinin her bitini ayrı ayrı ele alır. Bunu bir Boole değişkeni olarak işleme sokar. Mantıksal buyrukların uygun biçimde uygulanmasıyla bit değerini değiştirmek, bir grup biti değiştirmek veya veriler içine yeni bitler yerleştirmek mümkün olur.

Bazı tipik mantıksal ve bit işleme buyrukları çizelge 8.8 de verilmiştir. Silme buyruğu verinin bitlerini 0 yapar.

Tümleme buyruğu verinin 1 e göre tümleyenini alır. Yani bitleri ters çevirir. Ve, veya, özel-veya buyrukları gösterdikleri işlemi verinin bitlerine uygularlar. Her ne kadar Boole işlemleri yapılıyor olsa da mantıksal buyruklar bit işleme buyrukları olarak düşünülür. 3 mantıksal buyruk ile seçilen bir, bit 0 yapılır (silme), 1 yapılır veya tümleyeni alınır.

Seçilen Bitlerin Silinmesi

Ve buyruğu bir bit veya bir grup bitin temizlenmesi (sıfırlanması) için kullanılır. x.0 = 0 ve x.1 = x tir. Bir ikili değişken 0 ile VE lendiğinde sonuç 0, 1 ile VE lendiğinde sonuç 1 dir. Böylece ve buyruğu seçilen bitlerin silinmesi için kullanılır. AND buyruğuna aynı zamanda *maske* de denir.

Seçilen Bitlerin 1 Yapılması

OR buyruğu bir veya bir grup bitin 1 yapılması için kullanılır. $x + 1 = 1$ ve $x + 0 = x$ tir. Seçilen bir ikili değişken 0 ile VEYA landığında değer değiştirmez, 1 ile VEYA landığında değeri 1 olur. Bu yüzden OR buyruğu bir gurup bitin değerini 1 yapmak için kullanılır.

Seçilen Bitlerin Tümlenmesi

XOR buyruğu bir veya bir grup bitin tümlerinin alınması için kullanılır. $x \oplus 1 = x'$ $x \oplus 0 = x$ tir. İkili değişken 0 ile ÖZEL-VEYA landığında değer değiştirmez, 1 ile ÖZEL-VEYA landığında değeri 1 olur. Üç mantıksal işlem için örnek bölüm 4.5 te verilmiştir.

Diğer bazı bit işleme buyrukları çizelge 8.8 de verilmiştir. Örneğin elde yi uygun buyruk ile silme, 1 yapma veya tümleme buyrukları ile tek bitler işlenebilir. Bir diğer örnek kesme olayını kontrol eden yaz-bozun aktif veya pasif yapılmasıdır. Bu iş bit işleme buyrukları ile yapılır.

Kaydırma Buyrukları

Bir verinin kaydırılması buyrukları çok yararlı olup çeşitli biçimlerde buyruklar mevcuttur. Sağa veya sola kaydırma mümkündür. Bir kelimenin sonuna gelen bit kaydırmanın tipini belirler. Kaydırma buyrukları mantıksal, aritmetik veya dairesel olabilir.

Çizelge 8.9 da 4 tip kaydırma buyruğu veriliyor. Mantıksal kaydırmada sondaki bitlere 0 konur. Mantıksal sağa kaydırma için en sol bit son pozisyondur ve mantıksal sola kaydırma için en sağ bit son pozisyondur. Aritmetik kaydırmalar işaretli sayıların 2 ye göre tümleyeni için kullanılan kurallar ile uyum halindedir. Bu kurallar bölüm 4.6 da verilmiştir. Aritmetik sağa kaydırma buyruğu en soldaki işaret bitini muhafaza etmelidir. İşaret biti sayının diğer bitleri ile birlikte sağa kaydırılır fakat işaret bitinin kendisi değişmeksizin kalır. Bu son bitin aynı kaldığı sağa kaydırma işlemidir. Aritmetik sola kaydırma işlemi son pozisyona 0 koyar ve mantıksal sola kaydırma buyruğu ile aynıdır. Bu nedenle bir çok bilgisayar eğer mantıksal sola kaydırma buyruğu varsa ayrıca farklı bir aritmetik sola kaydırma buyruğuna sahip değildir.

Döndürme buyrukları dairesel kaydırmalar üretirler. Sözcüğün sonundan kaydırılan bitler mantıksal kaydırmadaki gibi kaybolmaz, bunun yerine tekrardan diğer uca girerler. Elde buyruğu aracılığıyla döndürme elde bitini sözcüğü döndürülen yazacın bir uzantısı olarak görür. Dolayısıyla elde buyruğu aracılığıyla bir sola döndürme elde bitini yazacın en sağ bit pozisyonuna, en sol bit pozisyonunu elde ye aktarır ve aynı zamanda bütün yazacı sola kaydırır.

Çizelge 8.9 Kaydırma buyrukları

İsim	Birleştirici dil sembolleri
Mantıksal sağa kaydırma	SHR
Mantıksal sola kaydırma	SHL
Aritmetik sağa kaydırma	SHRA
Aritmetik sola kaydırma	SHLA
Dairesel sağa kaydırma	ROR
Dairesel sola ka ydırma	ROL
Elde ile dairesel sağa kaydırma	RORC
Elde ile dairesel sola kaydırma	ROLC

Bilgisayarlar kaydırma buyrukları için çoklu biçim gösterir. Bu biçimde birçok alan vardır. Bu alanlardan biri OPR kodudur. Diğer alanlar kaydırmanın tipini ve kaydırma sayısını (kaç defa tekrarlanacağını) verir. Mümkün bir biçim

```
OP    REG   TYPE   RL    COUNT
```

şeklindedir. OP = işlem kodu, REG = verinin bulunduğu yazacın adresi, TYPE = 2 bit bir alan olup 4 tipten birini gösterir. RL = sağa-sola, COUNT ise kaydırmanın kaç defa olacağıdır. RL 1 bittir. Sayı k bittir. 2^{k-1} sayıya imkan verir. Bu şekilde her şey bir buyruk içinde verilebilir.

8.7 Program Denetimi

Buyruklar daima ardışık bellek adreslerinde bulunurlar. MİB içinde işlenmek için bu buyruklar ardışık adreslerden alınıp getirilirler ve sonra icra edilirler. Bir buyruk bellekten alınıp getirilince PC arttırılır. Böylece bir sonraki buyruğun adresi *PC* da bulunur. Bir buyruğun icrasından sonra denetim al-getir evresine döner ve bir sonraki buyruğun *PC* da buyruk adresi mevcuttur. Bu olay veri aktarım ve veri işleme buyrukları için geçerlidir. Diğer taraftan program denetim buyruklarının icrasıyla programın akış sırası değişebilir. Ayrıca veri aktarım ve işleme işlemleri için bazı şartlarımız mevcut olabilir. Program sayıcıda bir değişme olunca programın akışında bir kesiklik olabilir. Bu durum sayısal bilgisayarlarda önemli bir özelliktir, farklı program parçalarına geçişi mümkün kılar.

Bazı tipik program denetim buyrukları çizelge 8.10 da verilmiştir. Dallan ve Atla buyrukları bir biri yerine kullanılabilir. Fakat bazen farklı adresleme kipleri için kullanılırlar. Dallan buyruğu tek adresli bir buyruktur. Birleştirici dil olarak BR ADR şeklinde yazılabilir. İcra edildiğinde ADR *PC* içine yerleştirilir.

Çizelge 8.10 Program denetim buyrukları

İsim	Birleştirici dil sembolleri
Dallan	BR
Atla	JMP
Atla	SKP
Çağır	CALL
Geri dön	RET
Çıkarma ile kıyaslama	CMP
VE işlemi ile kontrol	TST

Dallan ve Atla buyrukları şartlı veya şartsız olabilir. Şartlı bir dallanma buyruğu bir şart belirtir. Şart pozitifse dallan veya şart 0 ise dallan. şeklindedir. Şart sağlanırsa sağlanan şartın adresi *PC* ye yerleştirilir ve bir sonraki buyruk bu adresten başlar. Şart sağlanmazsa *PC* değiştirilmez. Bir sonraki buyruk aynı sırada devam eder.

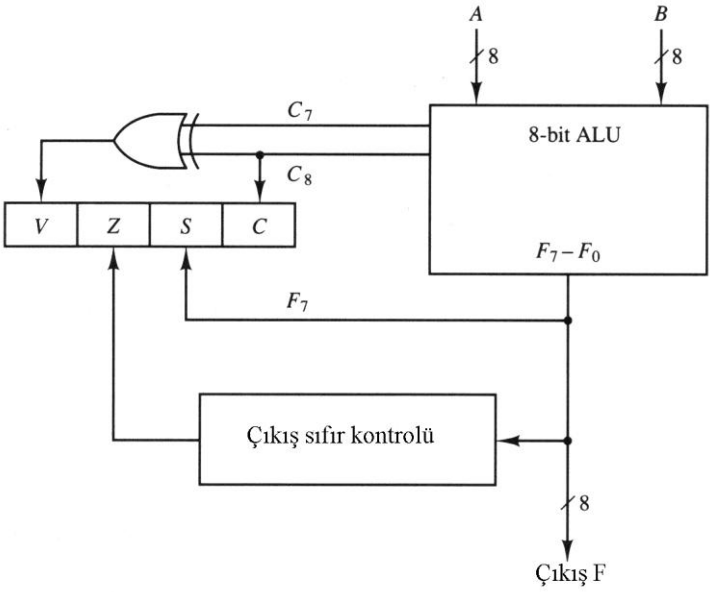
SKP buyruğunun adres alanına ihtiyacı yoktur. Dolayısıyla sıfır adresli buyruktur. Şartlı bir SKP buyruğu, şart sağlanıyorsa bir sonraki buyruğa atlar. Bunu sağlamak için al-getir evresi dışında icra evresi sırasında da *PC* arttırılır. Sart sağlanmazsa *PC* arttırılmaz ve bir sonraki buyrukla program sürer. Bir sonraki buyruk şartsız dallan buyruğudur ve dallanma olur. Bir tek şartlı dallanma buyruğu şart sağlanırsa dallanma yapar.

Çağırma ve Geri dön buyrukları alt programlarla beraber kullanılır. Bunların kullanışı bu kısım somununda anlatılacak. Karşılaştırma ve test buyrukları program akışını doğrudan doğruya değiştirmezler. Bunlar çizelge 8.10 da verilmiştir. Çünkü bunların uygulanışı ardından gelen şartlı dallanma buyrukları için şartı oluştururlar. CMP buyruğu bir çıkarma işlemi gerçekleştirir. Fakat çıkarma işleminin sonucu geri dönmez. Nitekim bazı durum bit şartları işlemin sonucu olarak 1 yapılır. Benzer şekilde TST buyruğu mantıksal VE işlemini iki veri arasında gerçekleştirir. Fakat sonucu saklamadan veya verileri değiştirmeden bazı durum bitlerini 1 yapar. İşe yarayan durum bitleri elde biti, işaret biti, sıfır belirticisi ve taşma şartıdır. Önce bu bitlerin oluşturulması anlatılacak sonra şartlı dallanma için nasıl kullanılacağı anlatılacaktır.

Durum Bit Şartları

Çoğu zaman ALU in yanına bir durum yazacı konur. Durum bit şarları bu yazacın bitlerine yazılır ve sonra yararlanılır. Durum bitlerine şart kodu ve bayrak bitleri de denir. Şekil 8.8 8 bit ALU ile 4 bit durum yazacını gösteriyor. 4 bit durum yazaçları *C*, *S*, *Z* ve *V* ile gösteriliyor. Durum bitleri ALU içindeki bir işlemin sonucuna göre 1 veya 0 yapılır.

1. *C*(elde) = 1 yapılır. Eğer C_8 yani son elde 1 ise. Eğer $C_8 = 0$ ise $C = 0$ yapılır.
2. Eğer en yüksek mertebeli bit $F_7 = 1$ ise *S* (işaret) biti 1 yapılır. $F_7 = 0$ ise $S = 0$ dır.
3. ALU nun bütün bitleri 0 ise (ALU nun bütün çıkışları = 0) $Z = 1$ yapılır. Eğer çıkış sıfır değil ise ve $Z = 0$ dır.
4. Son iki elde nin ÖZEL-VEYA sı 1 ise *V*(taşma) = 1 yapılır. Yani taşma vardır. Aksi halde 0 dır. Eğer negatif sayılar 2'nin tümleyeni biçiminde verilmişse bu taşma şartıdır (Bölüm 3.3). 8 bit ALU için çıkış +127 den büyük veya –128 den küçük olunca $V = 1$ olur.



Şekil 8.8 Durum yazacı bitleri

Durum bitleri ALU içindeki bir işlemden sonra kontrol edilerek *A* ve *B* arasındaki bazı bağlantılar anlaşılır. Eğer iki işaretli sayının toplama işleminden sonra $V = 1$ ise bir taşma vardır. Bir ÖZEL-VEYA işleminden sonra $Z = 1$ olmuşsa buradan $A = B$ çıkar. Çünkü eşit iki sayının ÖZEL-VEYA lanması sonucu çıkışlar 0 olur. $Z = 1$ yapılır. Tek bir bitin 0 mı yoksa 1 mi olduğu maskeleme işlemi sonucu, çıkışların hepsinin 0 olması yani $Z = 1$ ile anlaşılır. Örneğin $A = 101x1100$ olsun x aranılan bittir. *A* nın $B = 00010000$ ile maskelenmesi sonucu x = 0 ise bütün çıkışlar 0 olur ve $Z = 1$ olur. Eğer x = 1 ise çıkışlar 0 ve $Z = 0$ olur. VE işlemi çizelge 8.10 daki TST buyruğu yardımıyla elde edilebilir. Böylece *A* nın ilk şekli korunmuş olur.

Şartlı Dallanma Buyrukları

Çizelge 8.11 çoğunlukla kullanılan dallanma buyruklarını gösteriyor. Her bir birleştirici dil sembolü buyruk B ile (Dallan) başlar. Sonra şartın kısaltılmışı gelir. Eğer şart ters olarak kullanılacak ise N (No) harfi konmuştur. Dolayısıyla BC elde ile dallan ve BNC eldesiz dallan anlamına gelir. Eğer şart doğru ise program buyruk tarafından verilen adrese gider. Eğer şart yanlış ise program bir sonraki buyruk ile devam eder. Şartlı buyruklar JMP, SKP, CALL ve RET tipi program denetim buyrukları ile birleştirilebilir.

Çizelge 8.11 Şartlı dallanma buyrukları

Birleştirici dil sembolleri	Dallanma Şartları	Test Şartları
BZ	sıfır ise dallan	$Z = 1$
BNC	sıfır değilse dallan	$Z = 0$
BC	elde varsa dallan	$C = 1$
BNC	elde yoksa dallan	$C = 0$
BP	sayı pozitifse dallan	$S = 0$
BM	sayı negatifse dallan	$S = 1$
BV	taşma varsa dallan	$V = 1$
BNV	taşma yoksa dallan	$V = 0$
	İşaretsiz karşılaştırma şartları ($A - B$)	
BHT	yüksek ise dallan	$A > B$
BHE	yüksek veya eşit ise dallan	$A \geq B$
BLO	düşük ise dallan	$A < B$
BLOE	düşük veya eşit ise dallan	$A \leq B$
BE	eşit ise dallan	$A = B$
BNE	eşit değil ise dallan	$A \neq B$
	İşaretli karşılaştırma şartları ($A - B$)	
BGT	büyük ise dallan	$A > B$
BGE	büyük veya eşitse dallan	$A \geq B$
BLT	küçük ise dallan	$A < B$
BLE	küçük veya eşitse dallan	$A \leq B$
BE	eşit ise dallan	$A = B$
BNE	eşit değil ise dallan	$A \neq B$

Z (sıfır) durum biti, ALU içindeki bir işlemin sonucu test için kullanılır ve sonucun 0 olup olmadığını belirtir. Taşma biti en önemli bitten elde ye bir aktarma (taşma) olup olmadığını denetim için kullanılır. Taşma biti dönme işleminde sondaki bitin elde ye gelip gelmediğini anlamak için kullanılabilir. İşaret biti en önemli bitin durumunu gösterir. Eğer $S = 0$ ise çıktı pozitif, 1 ise negatiftir. O halde pozitif üzerine bir dallanma $S = 0$'a, negatif üzerine bir dallanma $S = 1$'e bakar. Doğal olarak bu iki şartlı dallan buyruğu sadece işareti değil, ALU çıktısının en önemli bitini denetim için kullanabilir Taşma biti 2'nin tümleyeni şeklindeki işaretli sayılarda aritmetik işlemler için kullanılır.

Daha önce söylendiği gibi CMP buyruğu iki verinin örneğin A, B farkını alır. Sonuç, sonuç yazacına aktarılmaz. Fakat durum bitleri etkilenir. Durum yazacı A ve B nin göreceli büyüklükleri için bilgi verir. Bazı bilgisayarlar bazı şartlı dallanma buyruklarının kıyaslama buyruğunun hemen arkasından uygulanmasına imkan verirler. Özel şartlar, karşılaştırılan iki sayının, yani A ve B nin işaretli veya işaretsiz olmasına bağlı olarak

test edilmelidir. Yüksek ve düşük kelimeleri işaretsiz sayılar için, büyüktür ve küçüktür kelimeleri işaretli sayılar için kullanılmıştır. Göreceli büyüklük test şartının yanına yazılmıştır. İşaretli ve işaretsiz sayılar için aynı gibi görünmektedir. Fakat durum böyle değildir ve her biri ayrı ayrı düşünülmelidir. Aşağıda buna bir örnek verilmektedir.

Sayısal Örnek

Şekil 8.8 deki 8 bit ALU ya bakılırsa 8 bite sığabilecek en büyük işaretsiz sayı 255 tir. İşaretli sayıların tanım aralığı +127 ile –128 arasındadır. İki sayının farkı eğer negatif sayı 2 nin tümleyeni şeklinde verilmişse değişmez.Yani işarete bağlı olmadan aynı çıkar. A = 11110000 ve B = 00010100 olsun. A - B işlemini yapmak için ALU B nin 2 nin tümlerini alır ve onu A ile toplar.

$$\begin{array}{rll} A: & 11110000 & \\ \overline{B}+1: & +\,11001100 & \\ A-B: & 11011100 & C=1 \quad S=1 \quad V=0 \quad Z=0 \end{array}$$

Kıyaslama buyruğu durum bitlerini yukarıdaki gibi düzenler. C = 1 dir. Çünkü son elde de taşma mevcuttur. S = 1 dir çünkü en soldaki bit 1 dir. V = 0 dır. Çünkü son iki elde eşittir ve 1 dir. Z = 0 dır. Çünkü sonuç 0 değildir.

Eğer işaretsiz sayılar düşünülürse A nın değeri 240 ve B nin değeri 20 dir. Çıkarma işleminin sonucu ondalık olarak 240 – 20 = 220 dir. İkili sonuç 11011100 gerçekten 220 ye eşittir ve biz $A > B$ ve $A \neq B$ sonuçlarını alırız. Bu sonuç C = 1 ve Z = 0 dan da çıkarılabilirdi. Bu karşılaştırmadan sonraki dallanmalar BHI, BHE ve BNE dir.

Eğer işaretli sayıları düşünürsek A = 16'dır çünkü 11110000 ikili sayısı 00010000 ikili sayısını 2 nin tümleridir. 00010000 ise +16 dır. B nin ondalık karşılığı +20 dir. Çıkarma işlemi (–16) – (+20) = 36 ve ikili olarak 11011100 (yani 00100100 ın 2 nin tümleridir) yani –36 dır. (–16) < (+20) olduğundan $A < B$ ve $A \neq B$ elde edilir. Bu sonuçlar durum bitinde S = 1, V = 0 ve Z = 0 sonuçlarından da çıkarılabilir. Bu mukayese işlemi sonucu dallanmalar BLT, BLE ve BNE dir. BNE ve BNZ buyrukları eşdeğerdir. Benzer şekilde BE ve BZ eşdeğerdirler. Bunlar çizelgenin tamam olması için çizelge 8.11 de 3 defa tekrarlanmıştır.

Örnekten de açık olarak belli olduğu gibi iki işaretsiz sayının mukayesesi durum bitleri olan S, V ve Z den belli olur (Problem 8.7 ye bakınız).

Bazı bilgisayarlar C bitini çıkarma işlemi sırasında ödünç alma biti olarak kullanırlar. Eğer $A \geq B$ ise ödünç alma olmaz. Eğer $A < B$ ise bir bit daha önemli bitten ödünç alınmalıdır. Ödünç alma için şart çıkarma işleminin yapılmasından sonra elde nin tümleyeninin alınmasıdır. Tabi önce B nin 2 nin tümleri alınır. Bu sebepten C bitini ödünç alma biti ola-

rak kabul eden bir işlemci çıkan sayının 2 nin tümlerini alıp diğer sayıyla toplayıp çıkarma işlemini tamamladıktan sonra C bitinin tümlerini alır ve bunu ödünç biti olarak kabul eder.

Alt Program Çağırma ve Geri Dönüş

Alt program belli bir işi yapan ayrı bir programdır. Bir icra sırasında alt program kendi fonksiyonunu icra için birçok farklı kısımdan birçok defa çağrılabilir. Bir alt program çağrılınca bir dallanma buyruğu çalışır. Alt programın icrasından sonra esas programa geri dönüş için tekrar dallanma yapılır.

Program denetimini alt programa aktaran buyruğun birçok farklı adı vardır. En çok kullanılanlar alt program çağırma, alt programa atlama, alt programa dallanma, veya dallan ve geri dönüş adresini sakla. Bir alt program çağıran buyrukta bir OPR kısmı ve alt programın başlangıcını gösteren bir adres vardır. Buyruk iki operasyonun icrasıyla gerçekleşir: (1) *PC* içindeki adres (geri dönüş adresi) geçici olarak saklanır. Böylece alt programdan sonra nereye döneceği bilinir. (2) Denetim alt programın başlangıç adresine aktarılır. Her alt programın son buyruğu alt programdan geri dönüş olup geçici yerdeki adresi *PC*'a yerleştirir. Bunun sonucu olarak program denetimi geçici olarak saklanan adrese geçmiş olur.

Farklı bilgisayarlar farklı geçici adresler kullanırlar. Bazıları dönüş adresini alt programın ilk bellek adresinde tutar. Bazıları ise bir işlemci yazacında saklar. Bazıları da yığıta yazar. En etkin yol dönüş adresini yığıta yazmaktır. Çünkü ardışık rutinler çağrıldığında hepsinin dönüş adresleri ardışık olarak yığıta yazılabilir. Alt programdan dönüş buyruğu yığıtın tepesindeki adresi alır ve *PC* ye yerleştirir. Böylece daima son alt programı çağıran programa dönülür. Bir alt program çağrısı aşağıdaki mikro işlemler ile yapılır.

$SP \leftarrow SP + 1$	Yığıt göstergesini 1 arttır
$M[SP] \leftarrow PC$	*PC* ın içeriğini yığıta yaz
$PC \leftarrow$ etkin adres	Denetimi alt program aktar

Eğer içinde bulunulan alt programdan başka bir alt program çağrılırsa bu sefer yeni dönüş adresi yığıta yazılır. Bulunulan alt programdan dönüş için kullanılan mikro işlemler

$PC \leftarrow M[SP]$	Yığıttan al ve PC a yaz
$SP \leftarrow SP + 1$	Yığıt göstergesini 1 arttır

Eğer bir alt program yığıtı kullanılırsa bütün adresler donanım tara-

fından otomatik olarak tek bir birim içine yazılır. Böylece programcı dönüş adreslerini nereye yazdığını düşünmek zorunda değildir.

Özyineli Alt Programlar

Bu alt program kendini çağıran bir alt programdır. Eğer dönüş adresi için tek bir yazaç veya bellek adresi kullanılırsa ve özyineli alt program kendini çağırırsa bir önceki adres silinir. Bu istenmeyen bir durumdur. Çünkü en önemli bilgi silinmiş olur. Bu problem farklı saklama adresleri kullanıldığında çözülebilir. Eğer yığıt kullanılırsa her dönüş adresi önceki adresleri bozmadan yığıta yerleşebilir. Bu, özyineli alt program problemini çözer. Çünkü daima bir sonraki alt program bir öncekine döner.

Program Kesme

Program kesme kavramı programın normal akışı sırasında ortaya çıkan birtakım problemleri düzeltmek için kullanılır. Program kesmesi olunca halen çalışmakta olan program kesilir. Denetim bu programdan bir servis programına geçer. Kesme iç veya dış istek üretilmesiyle ortaya çıkar. Servis programının işi bitince denetim tekrar normal programda kaldığı yere döner.

Kesme programı prensip olarak bir alt program çağrısına benzerdir. Sadece 3 noktada farklılık vardır: (1) Kesme genellikle iç veya dış bir sinyal tarafından başlatılır. Alt program ise bir buyruğun icrasıyla ortaya çıkar (sadece yazılım kesmesi böyle ortaya çıkar daha sonra açıklanacak). (2) Kesmede servis rutininin adresi buyruğun içinde bir adres alanından gelmez donanım tarafından belirlenir. (3) Kesme işlemi MİB nin durumunu belirleyecek bütün gerekli bilgileri depolar. Halbuki alt program sadece *PC* yi saklar. Bu üç kavram aşağıda açıklanacaktır.

Bir program bir kesme ile kesildikten sonra servis rutini devreye girer. Servis rutini işini bitirdikten sonra program, kesme geldiği andaki buyruğa olarak geri döner. MİB da kesmeden önce bulunduğu aynı durumda olmalıdır. Ancak bu olduğu taktirde program sanki hiçbir şey olmamış gibi icraya devam eder. MİB in durumu icra evresinin sonunda kesmenin haber alındığı durumdur. MİB in durumu;

1. *PC* nin içeriği
2. Bütün işlemci yazaçlarının içerikleri
3. Bazı durum şartlarının içerikleri

Program Durum Kelimesi Yönetici Kipi

MİB içindeki bütün durum bit şartlarına program durum kelimesi denir. PSW ayrı bir donanım yazacında saklanır. Bu yazacın içeriği MİB in durumunu karakterize eder. Tipik olarak Son ALU işleminden sonraki durum bitlerini içerir. Bu aynı zamanda müsaade edilen kesmeleri ve MİB in yönetici mi yoksa kullanıcı kipinde mi çalışmakta olduğunu belir-

tir. Birçok bilgisayarlar yerleşik bir işletim sistemine sahiptirler. bu işletim sistemi bilgisayardaki diğer bütün programları kontrol eder ve onlara yol gösterir. Eğer MİB işletim sisteminin parçası olan bir programı icra ediyorsa, yönetici veya sistem kipin dedir denir. Bazı buyruklar sadece bu kipte çalışabilirler ve icra edilebilirler. MİB normal olarak kullanıcı kipindedir. Kullanıcı programını bu kipte icra eder. MİB in içinde bulunduğu durum PSW içindeki özel durum bitlerinden anlaşılır.

Bazı bilgisayarlar bir kesmeye cevap verirken sadece *PC* ı saklarlar. dolayısıyla kesme servis programı durumu ve yazaç içeriklerini saklamak için gerekli buyrukları içermelidir. Ancak bundan sonra öz kaynakları kullanabilir. Çok az sayıda bilgisayar *PC* yi ve durum bitlerini ve yazaç içeriklerini ve kesme gelince saklar. Bilgisayarların çoğu *PC* yi ve PSW yi saklar. Bazı durumlarda bilgisayar içinde iki farklı işlemci yazaç kümesi mevcuttur. Her biri bir MİB kipi için kullanılır. Böyle bir makinede kesme servisi için yönetici kipine geçtiğinde yazaçların içeriklerini saklamak gerekmez.,

Bir kesmeyi servis etmek için gerekli donanım alt program çağırma buyruğunun icrasına çok benzer. MİB in durumu yığıta yazılır. Servis rutininin başlangıç adresi *PC* a aktarılır. Bu adres donanım tarafından belirlenir. Bazı bilgisayarlar kesmelerin yazıldığı bellek bölgeleri ayırırlar. servis rutini kesmeye neyin sebep olduğunu anlar ve sonra onu servis eder. Bazı bilgisayarlar her muhtemel kesme için bellekte yer ayırırlar. Bazen donanım kesmesi kendi adresini içerir ve MİB ini servis rutinine gönderir. Her ne olursa olsun MİB bir dallanma adresi seçmek ve kesmeyi servis etmek için bir donanıma sahip olmalıdır.

Bir kesme geldiğinde, bir buyruk icra ediliyorsa buyruğun icrası bitene kadar MİB kesmeyi dikkate almaz. Bir sonraki al-getir evresine geçmeden önce denetim herhangi bir kesme sinyali olup almadığına bakar. Eğer bir kesme gelmiş ve beklemede ise denetim donanım kesme evresine gider. Bu evre sırasında *PC* ve PSW nin içirikleri yığıta yazılır. Gelen kesmeye ait servis rutininin adresi *PC* ye yazılır ve PSW yeni bir duruma getirilir. Şimdi servis rutini icraya hazırdır. Servis rutininin icrası sırasında bilgisayar MİB kipindedir. Bu durum PSW ile belirlenmiştir. MİB kipi sistem kipi, yönetici kipi olarak ta isimlendirilir.

Servis rutininin son buyruğu kesme alt programından geri dönüştür. Bu buyruk icra edilince yığıttaki PSW ve dönüş adresi yerlerine tekrar yazılır. MİB kipi biter program kullanıcı kipine döner. Program kaldığı yerden icraya devam eder.

Kesme Çeşitleri

Programın normal çalışmasını engelleyen 3 büyük tipte kesme vardır.

1. Dış kesmeler
2. İç kesmeler
3. Yazılım kesmeleri

Dış kesmeler G/Ç aletlerinden, zaman aletinden, güç kaynağından veya herhangi bir dış kaynaktan olabilir. Bazı dış kesme örnekleri: G/Ç aleti veri aktarımı yapmak istemektedir, G/Ç aleti veri aktarımını bitirmiştir, bir olayın süresi uzamıştır veya güç kesintisi vardır. Zaman geçti kesmesi, sonsuz döngü ile olur ve ayrılan zamanı taşar. Güç kaybında bir servis rutini MİB nin bütün değerlerini, kesilmeden birkaç milisaniye önce, silinmeyecek bir bellek bölgesine aktarır.

İç kesmeler bir verinin veya buyruğun yanlış veya illegal kullanımından doğar. İç hata şartlarından dolayı meydana gelen kesme örnekleri; sıfırla bölmeye kalkma, geçersiz OPR kodu, yığıt taşması ve korumalı bölge sınırına tecavüz. Bu şartlarda bir buyruğun icrası tamamlanmadan icra durdurulur. Servis programı düzeltici önlemleri alabilir.

Dış ve iç kesmeler arasındaki fark şudur. İç kesme, programın neden olduğu olağandışı şartlardan dolayı meydana gelir. Dış bir olay buna neden olmaz. İç kesmeler programla aynı zamanlıdırlar. Dış kesmeler ise programla aynı zamanlı değildirler. Eğer program tekrar tekrar çalıştırılırsa aynı iç kesme aynı yer ve zamanda tekrar meydana gelir. Dış kesmeler dış olaylara bağlıdır, icra edilmekte olan programdan bağımsızdırlar.

Yazılım Kesmeleri

İç ve dış kesmeler MİB nin donanımında meydana gelen sinyaller ile başlatılır. Yazılım kesmesi bir buyruğun icrasıyla başlatılır. Yazılım kesmesi özel bir CALL buyruğudur. Fakat çağrılan bir programdan çok bir kesme gibi davranır. Programın herhangi bir yerinde programcı tarafından kullanılabilir. Yazılım kesmesinin en çok kullanımı yönetici çağırma buyruğu iledir. Bu buyruk MİB in kullanıcı kipinden yönetici kipine geçiş yapmasını sağlar. Bilgisayar içinde bazı işlemler ancak yönetici kipinde yapılabilir. Örneğin karışık bir G/Ç aktarım işlemi. Programcı tarafından yazılan program kullanıcı kipinde çalışmak zorundadır. Bir G veya Ç aktarma işlemi gerektiğinde yönetici çağırma buyruğu ile yönetici kipi çağrılır. Bu buyruk bir yazılım kesmesi oluşturur ve MİB nin durumu saklanarak yönetici kipine ait PSW getirilir. Çağıran program bu işin gereken bilgileri işletim sistemine vermelidir.

8.8 Azaltılmış Buyruk Kümeli Bilgisayar (RISC)

CISC

Bilgisayar tasarımında önemli noktalardan biride işlemcinin buyruk kümesinin belirlenmesidir. Belirli bir bilgisayar için seçilen buyruk kümesi bu bilgisayarın makine diliyle programlanmasını belirler. Eski bilgisayarda küçük ve basit buyruk kümeleri mevcuttur. Bunun nedeni buyrukların konacağı donanımı küçük tutmaktı. Sayısal donanım ucuzlamaya başlayınca ve tabi ki entegre devreler ileri bir düzeye varınca bilgisayar buyrukları da hem sayı hem de karışıklık bakımından arttı. Bazı bilgisayarlar 100 hatta 200'ün üstünde buyruk kümesine sahip oldular. Bu bilgisayarlar çok farklı veri tiplerini kullanabiliyorlardı, çok sayıda adresleme kipi bulunuyordu. Bilgisayar donanımlarının karışık olma eğilimi bir çok faktörün sebep olduğu bir olaydır. Örneğin mevcut kiplerin güncellenmesi, yüksek seviyeli dilden makine diline geçişin sağlanması ve yazılım temelli fonksiyonların donanım temelli olmasının sağlanması bu nedenlerden bazılarıdır. Çok sayıda buyrukları bulunan bir bilgisayar CISC olarak adlandırılır.

RISC

1980'lerin başında bilgisayar tasarımcıları az sayıda buyruğu bulunan ve sık sık belleğe erişmeyen makinelerin MİB içinde daha hızlı olduklarını tavsiye ettiler. Bu tip bir bilgisayar RISC olarak adlandırılır. Bu kısımda RISC ve CISC mimarilerinin temel özellikleri verecek ve RISC işlemcinin buyruk kümesi ile buyruk biçimlerini anlatacağız.

CISC Özellikleri

Bir bilgisayarın tasarımında buyruk kümesinin tespiti sadece makine diliyle programlama ihtiyaçlarını değil, yüksek seviyeli dille programlama ihtiyaçlarını da dikkate almalıdır. Yüksek seviyeli bir dilden makine diline çevirme derleyici programı ile yapılır. Karışık buyruk kümesinde bir diğer önemli amaç derlemenin kolay yapılmasını sağlamak ve bilgisayar verimini arttırmaktır. Derleyicinin görevi yüksek seviyeli dilin her bir deyimi için bir dizi makine buyruğu üretmektir. Eğer komutun tam karşılığı olan buyruklar mevcutsa bu iş çok kolaylaşır. CISC mimarisinin temel hedefi, yüksek seviyeli bir dille yazılmış bir programın her bir komutu için tek bir makine buyruğu bulabilmesini sağlamaktır. CISC mimarisinin örnekleri IBM 370 ve DEC WAX tır.

CISC mimarisinin önemli bir özelliği de değişik uzunlukta buyruk biçimlerinin kullanımına imkan vermesidir. Yazaç verilerini kullanan buyruklar 2 byte uzunlukta olabilirler. Fakat 2 bellek adresi kullanan buyruklar en az 5 byte uzunlukta olmalıdır. Eğer bilgisayar 32 bit kelime kullanıyorsa (4 byte) yazaç adreslemeli buyruk bunun yarısını kaplar. Bellek adreslemeli buyruk ise 1 kelime dışında 1 byte ister. Farklı uzunlukta

buyruk biçimleri kullanan makinelerde eğer bellek kelime uzunluğu sabit ise kod çözücü devrelere gerek vardır. Bunlar kelime içinde byteları sayabilmeli ve buyrukları uzunluklarına göre sergileyebilmelidirler.

Tipik bir CISC işlemcisi bellekteki verileri doğrudan işleyebilir. (Bellek içinde değil). Örneğin ADD buyruğu bellekte bir veriyi indis adresleme yoluyla ve ikinci veriyi doğrudan adresleme ile belirtebilir. Buyruk içinde yer alan bir diğer bellek adresi ise toplamın sonucunu yazmak için belirtilmiş olabilir. Bu durumda buyruğun icrası 3 adresli buyruk olarak yapılacaktır. Her ne kadar CISC işlemcileri sadece işlemci yazaçlarını kullanan buyruklara da sahip iseler de başka kiplerde işlemlerin varlığı yüksek seviyeli dillerin derlenmesini basitleştirmek için kullanılabilir. Ancak çok sayıda kipin kullanılması daha fazla donanım mantığına gereksinim gösterir. Bu nedenle hesaplamalar yavaşlar. Özetle CISC in özellikleri;

1. Çok sayıda buyruk. Tipik olarak 100 ile 250 arası
2. Bazı buyruklar çok özel işler yaparlar ve çok nadir kullanılırlar
3. Çok sayıda adresleme kipi tipik olara 5-20 arası
4. Değişen uzunlukta buyruk biçimleri
5. Buyrukların verileri bellekten kullanabilmesi (Bellek içinde değil fakat doğrudan işlemcide işleme girebilir).

RISC Özellikleri

RISC mimarisinin temel kavramı buyruk kümesinin basitleştirilmesi ile icra zamanının azaltılmasıdır. RISC işlemcisinin temel özellikleri

1. Göreceli olarak az sayıda buyruk
2. Az sayıda adresleme kipi
3. Bellek erişimi sadece LOAD ve STORE buyrukları ile yapılır
4. Bütün işlemler MİB yazaçlarında yapılır
5. Sabit uzunlukta kolay çözülebilir kod
6. Tek evreli buyruk icrası
7. Mikro programlı denetim yerine donanımsal denetim

RISC işlemcinin küçük buyruk kümesi içinde çoğunlukla yazaçtan yazaca işlemler ve basit LOAD ve STORE buyrukları bulunur. Dolayısıyla her veri LOAD buyruğuyla bir işlemci yazacına getirilir. Bütün hesaplamalar yazaçlarda bulunan veriler arasında yapılır. Sonuçlar belleğe STORE buyruğu ile gönderilir. Bu yapısal olay buyruk kümesini basitleştirir. Çok az adresleme kipi vardır. Çünkü bütün buyrukların adreslemeleri basit yazaç adreslemeleridir. Diğer adresleme kiplerinden derhal veri ve göreceli adresleme ilave edilebilir.

Oldukça basit buyruk biçimi kullanılmasından dolayı buyruk uzunluğu sabitleştirilebilir ve bir kelime içine sığabilir. Bunun sonucu olarak RISC buyruk biçiminin kodu kolayca çözülebilir. Böylece OPR kodu ve yazaç alanlarına denetim tarafından aynı anda erişilebilir. Buyrukların ve biçimlerinin basitleştirilmesi denetim mantığını basitleştirir. Daha hızlı işlemler için M programlanmış denetim yerine donanımsal denetim daha iyidir. Donanımsal denetim örneği bölüm 5 te temel bilgisayarın denetim birimiyle beraber verilmişti. Mikro programlanmış denetim bölüm 7 de anlatılmıştır.

RISC işlemcisinin en önemli bir özelliği bir buyruğu bir saat evresinde icra edebilmesidir. Bu; al-getir, kod çöz ve icra evrelerinin üst üste getirilmesiyle sağlanır. Üst üste gelen aynı buyruğun değil farklı buyrukların evreleridir. bunun için pipelining adı verilen bir usul kullanılır. Böylece 3 buyruk 3 evrede icra edilince bir buyruk 1 evre sürmüş olur. Yükle ve STORE buyruğu iki zaman evresi gerektirir. Çünkü belleğe erişim yazaç işlemlerinden daha fazla zaman alır. Etkili boru hattı işlemi ve bazı başka özellikler RISC işlemcilere mal edilir. Fakat bu özellikler RISC dışı bilgisayarlarda da vardır. RISC makinelere ait özellikler şunlardır;

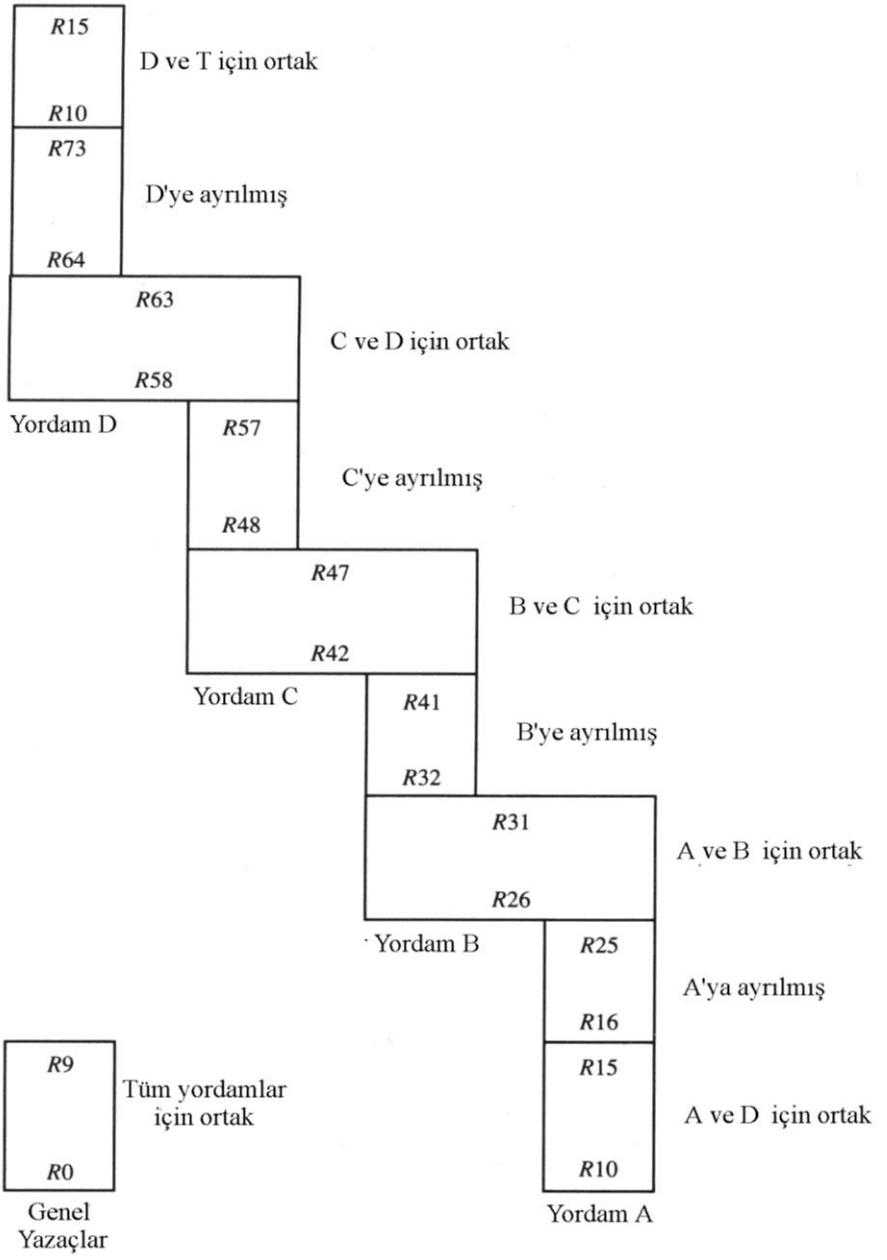
1. MİB içinde oldukça çok sayıda yazaç mevcut
2. Alt program çağırma ve geri dönüş işlemlerini hızlandırmak için üst üste binen yazaç pencereleri
3. Etkili buyruk boru hattı
4. Yüksek seviyeli dillerdeki programların makine diline etkin çevrilmesi

Boru Hattı

Çok sayıda yazaç bulunması ara sonuçların saklanması ve verilerin adreslenmesinde kolaylık sağlar. Bellek yerine yazaç kullanılması çok yararlıdır. Çünkü yazaç arası aktarımlar, bellekten yapılan aktarımlara göre çok hızlıdır. Bu durumda sık sık kullanılan verilerin yazaçlarda sürekli tutulması bellekle yazaç arası veri aktarımlarını minimize eder (ön bellek prensibi). RISC mimarisinin gelişmiş performansını test için yapılan çalışmalar büyük yazaç kümeleri ile azaltılmış buyruk kümesi etkisi arasında bir fark olmadığını göstermiştir. Bu sebeple MİB içindeki çok sayıda yazaçtan bazıları RISC işlemcisi ile birleştirilmiştir. Üst üste binen yazaç pencereleri aşağıda anlatılacaktır. Bu pencereler bir alt program çağrısından sonra program denetiminin aktarılmasında kullanılır. RISC içindeki buyruk boru hattı bu kavramın tanıtılmasından sonra 9.5 te anlatılacaktır.

Üst Üste Binen Yazaç Pencereleri

Alt program çağrısı ve geri dönüş yüksek seviyeli dillerde sık sık olur. Birleştirici diline çevrilince bir alt program çağrısı, bu alt programın kul-



Şekil 8.9 Üst üste binen yazaç pencereleri

lanacağı yazaç değerlerini saklamak için bir dizi buyrukları çalıştırır ve bu değerleri saklar. Alt program çağrısında bu olay olur. Alt program geri dönünce eski yazaç değerlerini yerlerine yazar. Yazaç değerlerinin saklanması ve tekrar yazılması, parametre ve sonuçların aktarılması çok zaman alıcı işlemlerdir. Bazı bilgisayarlar ayrı yazaç bankaları ayırırlar. Her bir alt program kendi bankasını kullanır. Bu şekilde saklama ve tekrar yerine yazma olaylarına gerek kalmaz. Bazı bilgisayarlar ise bu işi yığıt ile yapar. Ancak yığıt bir bellek erişimi gerektirir.

RISC işlemcilerin bazılarının bir özelliği üst üste binen yazaç pencereleri kullanmalarıdır. Böylece saklama ve geri yazma işleminden kurtulmuş olunur. Her alt program çağrısı yeni bir dizi yazaç içeren bir pencere açılması ile sonuçlanır. Bu pencere bu yeni alt program aittir. Her bir alt program çağrısı bir göstergeyi arttırarak yeni bir pencere açılmasını sağlar. Her bir geri dönüş komutu ise göstergeyi bir azaltır ve bir önceki pencerenin aktif hale gelmesini sağlar. Komşu veya takip eden alt programların pencerelerinin üst üste gelen yazaçları vardır. Bunlarla parametre ve sonuçlar birbirlerine aktarılır.

Bu kavram şekil 8.9 da gösterilmiştir. Sistemin toplam 74 yazacı vardır. R0 ile R9 arasındaki yazaçlar ortak yazaçlar olup bütün alt programlarca paylaşılır. Diğer 64 yazaç 4 pencereye bölünmüş olup *A*, *B*, *C* ve *D* alt programlarını tutar. Her bir pencerede 10 yerel yazaç ve birbirinin takip eden alt programlarda ortak kullanım için altışarlık 2 küme vardır. Yerel yazaçlar yerel değişkenler için kullanılır. Ortak yazaçlar takip eden alt programlar arasında parametrelerin ve sonuçların değişimi için kullanılır. Ortak yazaçların varlığı verinin gerçekten hareket etmeden kullanılmasını sağlar. Bir zamanda yalnız bir pencere açıktır ve gösterge onu gösterir. Her bir alt program çağrısı göstergeyi arttırır ve yeni bir pencere açılır. Çağıran alt programın yüksek yazaçları çağrılan alt programın düşük yazaçları ile çakışır. Dolayısıyla parametreler otomatik olarak çağırandan çağrılana aktarılmış olur.

Örnek olarak *A* nın *B* yi çağırdığı varsayılsın. 26 dan 31 e kadar olan yazaçlar iki alt program ortaktır. Alt program *A*, *B* ye gerekli parametreleri buraya saklar. *B* yordamı 32'den 41 e kadar yazaçları yerel olarak kullanır. Eğer *B C* yi çağırırsa parametreleri 42 den 47 ye kadar olan yazaçlara aktarır. *B* yordamı hesaplamanın sonunda geri dönmeye hazırsa sonuçları 26 - 31 yazaçlarına yazar ve *A* penceresine geri döner. *R*10 - *R*15 yazaçları *A* ve *D* arasında ortaktır. Çünkü ardışıklık dairesel kabul edilmektedir.

Daha önce söylendiği gibi *R*0 - *R*10 yazaçları bütün alt programlara açıktır. Şekil 8.9 daki her bir alt program aktif olduğu anda 32 yazaca sahiptir. Bu sayıya 10 tane ortak yazaç dahildir. 10 yerel yazaç, 6 yüksek

ve 6 düşük üst üste binen yazaçlar toplam 32 eder. Başka biçimde de sabit büyüklükte pencere sistemi yaratılabilir. Bunlar yazaçların sayısı ve pencere büyüklüğü bakımından fark eder. Genel bağıntı;

Ortak yazaçların sayısı = G

Her bir penceredeki yerel yazaç sayası = L

İki pencere arasındaki ortak yazaç sayısı = C

Pencere Sayısı = W olduğunda

$$\text{bir penceredeki yazaç sayısı} = L + 2C + G$$

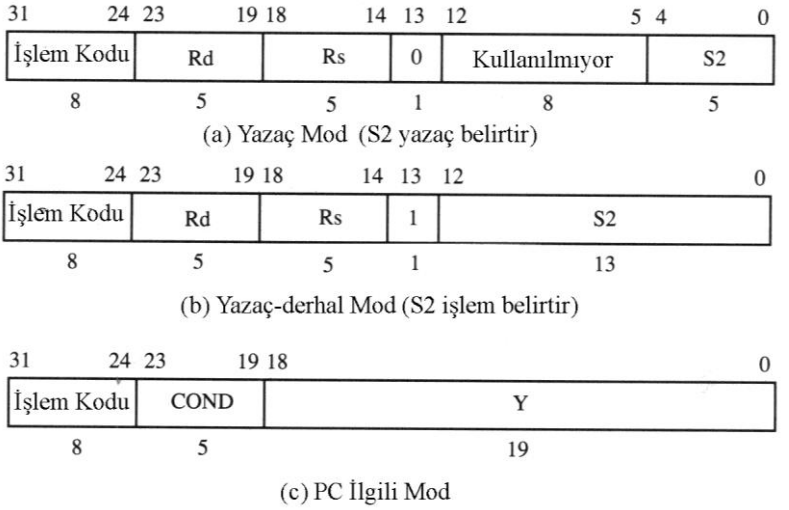
$$\text{İşlemcinin ihtiyacı olan yazaç sayısı} = (L + C)\ W + G$$

Şekil 8.9 da $G = 10$ $L = 10$ $C = 6$ $W = 4$ tür.

Berkeley RISC I

RISC mimarisinin avantajlarını göstermek için yapılan örneklerden biride University of California, Berkeley de oluşturulmuştur. Berkeley RISC I 32 bit IC MİB ne sahiptir. 32 bit adreslemeyi destekler. 8-16-32 bit veri ile kullanılabilir. 32 bit buyruk biçimi vardır. Toplam 31 buyruk vardır. 3 temel adresleme kipi mevcuttur. Yazaç adresleme, derhal adresleme ve *PC* ye göreceli adresleme (dallanmalar için). 138 yazacı vardır. 10 ortak yazaç, 8 pencere ve her bir pencerede 32 yazaç. Bunların kurulumu şekil 8.9 dakine benzer. Sadece 32 lik bir yazaç kümesi bir pencerede olduğundan, bir zamanda 32 yazaca erişilebilir. Buyruk biçimi 5 bit yazaç alanı ile bir işlemci yazacını belirler.

Şekil 8.10 da yazaçtan yazaca aktarımlar ve bellek aktarımları için 32 bit buyruk biçimini göstermektedir. OPR alanındaki 7 bit operasyonu belirler. 8. bit bir ALU işleminden sonra durum bitlerinin yenilenip yenilenmeyeceğini gösterir. Yazaçlar arası işlem buyruklarında 5 bit Rd alanı işlemin sonucunun yazılacağı 32 yazaçtan birini belirler. İşlem Rs ve S2 ile gösterilen alanlardaki belirtilen veriler arasında yapılır. RS kaynak yazaçlarından biridir. Buyruğun 13 üncü biti 0 ise S2'nin düşük seviyeli 5 biti başka bir kaynak yazacını gösterir. Eğer 13 üncü bit 1 ise S2 işareti ile beraber 13 bit bir sabit gösterir. böylece buyruk 3 adresli hale gelmiş olur. Fakat ikinci kaynak bir yazaç olabilir veya derhal veri olabilir. Bellek adreslemeli buyruklar bir yazaç içindeki 32 bit adresi gösterir. S2 ise ofset belirler. R0 yazacı 0 ile doludur. Dolayısıyla 0 olması gerekli bir alan yerine kullanılabilir. 3 üncü buyruk biçiminde son 3 alan birleştirilerek 19 bitlik göreceli adres için kullanılır. Genelde JUMP ve CALL buyrukları ile kullanılır. COND alanı JUMP buyrukları için Rd alanı yerine geçer ve 16 tane dallan buyruğundan biri için kullanılır.



Şekil 8.10 Berkeley RISC I Buyruk biçimleri

RISC I in 31 buyruğu çizelge 8.12 de gösterilmiştir. 3 grupta toplanmışlardır. Veri işleme buyrukları, aritmetik, mantıksal ve kaydırma işlemlerini yapar. İşlem kodu ve veri sütunlarındaki kodlar birleştirici dil programı yazarken kullanılırlar. Yazaç transfer ve tanımlama sütunları buyruğu yazaç aktarım diliyle açıklıyor. Bütün buyruklar 3 adreslidir. S2 bir yazaç veya derhal veri olarak kullanılabilir. Bu olay # işareti ile gösterilmiştir. ADD buyruğu ile bu olayın nasıl olduğuna bakılırsa

```
ADD R22, R21, R23    R23 ←R22 + R21
ADD R22, #150, R23   R23 ←R22 + 150
ADD R0, R21, R22     R22 ←R21 (taşıma)
ADD R0, #150, R22    R22 ←150 (veri aktarımı)
ADD R2, #1, R22      R22 ←R22+1(arttırma)
```

Daima 0 olan *R*0 yazacının kullanılmasıyla bir yazaç içeriğinin diğerine aktarımı mümkündür. Arttırma işlemi 1 eklenerek yapılır.

Veri aktarım buyruklarında 6 yükle buyruğu vardır. 3 tane sakal buyruğu var. 2 buyruk ise PSW nin aktarımı için mevcut PSW yi tutan yazaç MİB nin durumunu gösterir, PC yi içerir, ALU daki durum bitlerini içerir, yazaç pencereleriyle ilgili göstergeyi içerir ve MİB nin durumunu gösteren diğer bilgileri de içerir.

Çizelge 8.12 Berkeley RISC I buyruk kümesi

İşlem kodu	Veriler	Yazaç aktarımları	tanımlama
Veri İşleme Buyrukları			
ADD	Rs, S2, Rd	$Rd \leftarrow Rs + S2$	tamsayı toplama
ADDC	Rs, S2, Rd	$Rd \leftarrow Rs + S2$ + elde	elde ile toplama
SUB	Rs, S2, Rd	$Rd \leftarrow Rs - S2$	tamsayı çıkarma
SUBC	Rs, S2, Rd	$Rd \leftarrow Rs - S2$ – elde	elde ile çıkarma
SUBR	Rs, S2, Rd	$Rd \leftarrow S2 - Rs$	yer değiştirilmiş çıkarma
SUBCR	Rs, S2, Rd	$Rd \leftarrow S2 - Rs$ – elde	elde ile çıkarma
AND	Rs, S2, Rd	$Rd \leftarrow Rs \wedge S2$	VE
OR	Rs, S2, Rd	$Rd \leftarrow Rs \vee S2$	VEYA
XOR	Rs, S2, Rd	$Rd \leftarrow Rs \oplus S2$	ÖZEL-VEYA
SLL	Rs, S2, Rd	$Rd \leftarrow Rs\ S2$ ile kaydırma	sola kaydırma
SRL	Rs, S2, Rd	$Rd \leftarrow Rs\ S2$ ile kaydırma	mantıksal sağa kaydırma
SRA	Rs, S2, Rd	$Rd \leftarrow Rs\ S2$ ile kaydırma	aritmetik sağa kaydırma
Veri Aktarım Buyrukları			
LDL	(Rs)S2, Rd	$Rd \leftarrow M[Rs + S2]$	uzun kelime yükle
LDSU	(Rs)S2, Rd	$Rd \leftarrow M[Rs + S2]$	kısa işaretsiz kelime yükle
LDBU	(Rs)S2, Rd	$Rd \leftarrow$M$[Rs + S2]$	kısa işaretli kelime yükle
LDBS	(Rs)S2, Rd	$Rd \leftarrow M[Rs + S2]$	işaretsiz byte yükle
LDHI	Rd, Y	$Rd \leftarrow Y$	derhal yüksek veri yükle
STL	Rd,(Rs)S2	$M[Rs + S2] \leftarrow Rd$	uzun kelime sakla
STS	Rd,(Rs)S2	$M[Rs + S2] \leftarrow Rd$	kısa kelime sakla
STB	Rd,(Rs)S2	$M[Rs + S2] \leftarrow Rd$	byte sakla
GETPSW	Rd	$Rd \leftarrow PSW$	durum kelimesini yükle
PUTPSW	Rd	$PSW \leftarrow Rd$	durum kelimesini sakla
Program Denetim Buyrukları			
JMP	COND,S2(Rs)	$PC \leftarrow Rs + S2$	şartlı atlama
JMPR	COND, Y	$PC \leftarrow PC + Y$	göreceli atlama
CALL	Rd, S2(Rs)	$Rd \leftarrow PC$	alt program
		$PC \leftarrow Rs + S2$	çağır ve
		$CWP \leftarrow CWP - 1$	pencereyi değiştir
CALLR	Rd, Y	$Rd \leftarrow PC$	göreceli
		$PC \leftarrow PC + Y$	çağır ve
		$CWP \leftarrow CWP - 1$	pencereyi değiştir
RET	Rd, S2	$PC \leftarrow Rs + S2$	geri dön ve
		$CWP \leftarrow CWP - 1$	pencereyi değiştir
CALLINT	Rd	$Rd \leftarrow PC$	kesmeleri kapat
		$CWP \leftarrow CWP - 1$	
RETINT	Rd,S2	$PC \leftarrow Rd + S2$	kesmeleri aç
		$CWP \leftarrow CWP - 1$	
GTLPC		$Rd \leftarrow PC$	

Yükle ve sakla buyrukları veriyi yazaç ile bellek arasında hareket ettirilirler. Yükle buyrukları işaretli ve işaretsiz 8 bit (byte) veri veya 16 bit (kısa kelime) veriyi içerebilirler. Uzun-kelime buyruklar 32 bit veri üzerine işlem yapar. Her ne kadar yazaç değerine ekleme ile yapılan göreceli adresleme kipi mevcut ise de yazaç dolaylı ve yazaç doğrudan adresleme mümkündür. Aşağıda uzun-kelime yükle buyruklarının farklı adresleme kipiyle yazılışı görülüyor.

```
LDL (R22)#150, R5   R5 ←M[R22]+ 150
LDL (R22)# 0, R5    R5 ←M[R22}
LDL (R 0)#500, R5   R5 ←M[500]
```

İlk buyrukta etkin adres R22 yazacının içeriğinden (+150) yer değiştirme ile elde edilmiştir. İkinci buyruk 0 yer değiştirme kullanır ve yazaç dolaylı kip ile elde edilir. 3 üncü buyruk R0 daki 0ları kullanır, doğrudan adres kipi buyruk elde edilir.

Program denetim buyrukları PC ile beraber program akış sırasını kontrol eder. İki atlama ve iki alt program çağırma buyruğu vardır. Birisi indis+yer değiştirme adresini kullanır. İkincisi göreceli adres kullanır. 19 bit Y değeri göreceli adresi oluşturur, alt program çağırma ve geri dönüş buyrukları, 3 bit *CWP* (geçerli pencere göstergesi) kullanırlar. Bu halen aktif olan yazaç penceresini gösterir. Programın her yeni alt program çağırışında *CWP* 1 azaltılarak komşu düşük seviyeli pencereyi gösterir.

SORULAR

8-1. Veri yolu kurulumlu bir MİB, şekil 8.2 ye benzemekte ve 32 bitlik 16 yazaç içermektedir. Ayrıca bir ALU ve bir hedef kod çözücüsü vardır.

- **a.** Veri yolu içinde kaç tane MUX vardır? Her birinin büyüklüğü ne kadardır?
- **b.** MUX A ve MUX B için kaçar tane seçim girişi gerekir.
- **c.** Kod çözücünün kaç girişi kaç çıkışı vardır?
- **d.** ALU da veri girişi ve çıkışı için (giriş çıkış taşma biti dahil) kaç tane giriş ve çıkış vardır?
- **e.** ALU da 35 işlem olduğunu varsayarak sistemin denetim kelimesini bulun.

8-2. Şekil 8.2 deki veri yolu sisteminde şu yayılma gecikmeleri olmaktadır. 30 ns; sinyallerin MUX lardan geçişi, 80 ns; ALU içinde ADD işlemi, 20 ns; hedef kod çözücü gecikmesi, 10 ns; verinin hedef yazaca alınması süresi. Saat için kullanılacak en küçük çevrim zamanı ne kadardır?

8-3. Aşağıdaki mikro işlemleri yerleştirmek için şekil 8.2 deki işlemciye uygulanacak denetim kelimesini bulun.

a. $R1 \leftarrow R2 + R3$
b. $R4 \leftarrow R4$
c. $R5 \leftarrow R5 - 1$
d. $R6 \leftarrow$ shl $R1$
e. $R7 \leftarrow$ giriş

8-4. Aşağıdaki 14 er bitlik denetim kelimeleri şekil 8.2 deki işlemcide uygulandığında icra edilecek mikro işlemleri bulun.
a. 00101001100101
b. 00000000000000
c. 01001001001100
d. 00000100000010
e. 11110001110000

8-5. Şekil 8.3 teki yığıtta SP = 000000 dır. Eğer
a. DOLU = 1 ve BOŞ = 0 ise
b. DOLU = 0 ve BOŞ = 1 ise
Yığıtta kaç bilgi vardır?

8-6. Bir yığıt öyle kurulmuştur ki SP daima bitişik boş adresi göstersin. Bunun anlamı, şekil 8.4 teki yığıt 4000'e ilkleştirilebilir ve ilk bilgi 4000 adresine yazılabilir. İtme ve çekme işlemleri için mikro işlemleri yazın.

8-7. Aşağıdaki aritmetik işlemleri zıt polonyalı gösterimine çevirin.
a. $A * B + C * D + E * F$
b. $A * B + A * (B * D + C * E)$
c. $A + B * [C * D + E * (F + G)]$
d. $\dfrac{A*[B+C*(D+E)]}{F*(G+H)}$

8-8. Aşağıdaki aritmetik işlemleri zıt polonyalı gösteriminden sabit gösterime çevirin.
a. $A\ B\ C\ D\ E + * * - /$
b. $A\ B\ C\ D\ E * / - +$
c. $A\ B\ C * / D - E\ F / +$
d. $A\ B\ C\ D\ E\ F\ G + * + * + *$

8-9. Aşağıdaki sayısal aritmetik bağıntıyı zıt polonyalı gösterimine çevirin ve aşağıdaki sayısal sonucu ortaya koyacak yığıt işlemlerini bulun.

$$(3 + 4) [10 (2 + 6) + 8]$$

8-10. İlk giren ilk çıkar (LIFO) tipi bir bellek kurulumunda belleğe ilk yazılan bilgi, ilk geri alınan olur. Böyle bir bellek 3 sayaçla nasıl çalışır? Bir yazma sayacı (*WC*) bellekte yazılacak adresi tutar. Okuma sayacı (*RC*) okunacak bilginin adresini tutar. Halen kaç bilginin yazılı olduğunu gösteren mevcut durum sayacı (*ASC*) dır. Her bilgi yazılışında ASC arttırılır.ve her okunuşta yani geri alışta azaltılır. *ASC* nin birimi kelimedir.

8-11. Bir bilgisayarda 12 bit adres ve 32 bit buyruk bulunuyor. Eğer 250 tane 2 adresli buyruk varsa, kaç tane tek adresli buyruk bulunabilir.

8-12. Aşağıdaki aritmetik işlemi yapan bir programı

$$X = \frac{A - B + C * (D * E - F)}{G + H * K}$$

a. 3 adresli genel yazaçlı bilgisayar için
b. 2 adresli genel yazaçlı bilgisayar için
c. birikeç tipi tek adres buyruklu bilgisayar için
d. Yığıt kurulumlu bilgisayar için. Sıfır adres işlemli buyrukları olacak.

8-13. Bir bilgisayarın belleği 32 bit kelime uzunluğu olan 256K kelimeliktir. Bilgisayarın buyruk biçimi 4 alandan oluşmaktadır. İşlem kodu alanı, 7 adresleme kipinden birini seçen kip alanı, 60 yazaçtan birini belirten yazaç adresi alanı ve bir bellek adresi. Bir kelimelik bir bellek alanında bulunan bir buyruk için buyruk biçimini ve alanların bit uzunluklarını verin.

8-14. İki kelimelik bir buyruk, bellekte *W* sembolü ile gösterilen bir adreste bulunmaktadır. Bu buyruğun adres alanı (*W* + 1) adresinde olup *Y* sembolü ile gösterilmektedir. Buyruğun icrası sırasında kullanılacak veri *Z* ile gösterilen adreste bulunsun. Bir indis yazacı *X* değerini içeriyor. Buyruğun adresleme kipi aşağıdakilerden birisi iken *Z* diğer adreslerden nasıl hesaplanır.
a. Doğrudan
b. Dolaylı
c. Göreceli
d. İndisli

8-15. Göreceli kipte dallanma tipi bir buyruk bellekte ondalık 750 adresinde bulunuyor. Dallanma ondalık 500 adresine yapılıyor.
a. buyruğun ondalık olarak göreceli adres değeri ne olmalıdır.
b. Göreceli adres değerini 12 bit kullanarak ikiliye çeviriniz (Neden değer 2 nin tümleyeni olmalıdır?)
c. Al-getir evresinin tamamlanmasından, 500 adres değerinin ikiliye çevrilmesinden sonra *PC* içindeki değeri ikili düzende ifade edin.

8-16. Dolaylı adreslemeli bir buyruk alınıp getirildiğinde ve sonra icra edildiğinde
a. Buyruk eğer hesap tipi bir buyruksa
b. Buyruk eğer dallanma tipi bir buyruksa
denetim birimi belleğe kaç defa gider?

8-17. Eğer bir buyruk indisli adresleme kipinde ise ve buyruk dolaylı yazaç adreslemeli buyruğun aynısı ise adres alanı ne olmalıdır?

8-18. Bir buyruk 300 adresinde ve buyruğun adres alanı 301 adresinde bulunsun. Adres alanında 400 adresi var. Bir işlemci yazacı R1, 200

değerini içeriyor. Eğer adresleme buyruğu kipi (a) Doğrudan; (b) Derhal; (c) Göreceli; (d) Yazaç dolaylı; (e) Eğer *R*1 indis yazacı ise etkin adresi hesaplayın.

8-19. 8 bitlik bir bilgisayar düşününüz. İki tane 32 bitlik işaretsiz sayı onaltılık düzende

$$(6E\ \ C3\ \ 56\ \ 7A) + (13\ \ 55\ \ 6B\ \ 8F)$$

dir. Bu iki sayıyı "taşma biti ile topla" buyruğu ile nasıl toplarsınız?

8-20. İkili düzende verilmiş olan 10011100 ve 10101010 stringlere mantıksal VE, VEYA, ÖZEL-VEYA işlemlerini uygulayın.

8-21. 16 bitlik 1001101011001101 değeri veriliyor.
- **a.** İlk sekiz biti temizlemek için
- **b.** Son sekiz biti 1 yapmak için
- **c.** Ortadaki sekiz bitin tümleyenini almak için hangi işlem uygulanmalıdır?

8-22. 8 bit bir yazaç 01111011 değerini içeriyor. Taşma bitinde 1 var. Çizelge 8.9 da verilen buyrukları uygulayarak 8 kaydırma işlemi yapınız. Her defasında bu değerden başlayın.

8-23. Aşağıdaki işaretli sayıları 8 bit kullanarak ikili düzene çeviriniz. +83; –83; +68; -68
- **a.** (–83) + (+68) işlemini ikili düzende yapın. Elde edilen sonucu açıklayın
- **b.** (–68) – (+83) işlemini ikili düzende yapın ve taşma olup olmadığını açıklayın
- **c.** İkili düzende (–68) i bir defa sağa kaydırın. Kaymış sayıyı ondalık düzende söyleyin
- **d.** –83 ü bir defa sola kaydırın ve taşma olup olmadığını bulun

8-24. Şekil 8.8 de "çıkı sıfır kontrolü" başlığını taşıyan kısmın 8 bitlik VEYA-DEĞİL kapısı olduğunu gösterin.

8-25. 8 bitlik bir bilgisayarda bir yazaç *R* olsun. Şekil 8.8 de gösterildiği gibi durum bitleri olan *C*, *S*, *Z* ve *V* yi aşağıda gösterilen buyruklarla bulun. *R* yazacının ilk değeri her durum başında onaltılık düzende 72 dir.
- **a.** Derhal veri C6 yı *R* ye ekleyin
- **b.** Derhal veri IE yi *R* ye ekleyin
- **c.** Derhal veri 9A yı *R* den çıkarın
- **d.** Derhal veri 8D yi *R* ye VE leyin
- **e**. *R* yi *R* ile ÖZEL-VEYA layın

8-25. İki işaretsiz sayı çıkartma işlemi yoluyla karşılaştırılıyor. ($A - B$) taşma durum biti karşılaştırma buyruğundan sonra ticari bilgisayarlarda ödünç alma biti olarak kabul edilir. Yani eğer $A < B$ ise $C = 1$ dir. Soru 8.26 için çizelge 8.11 de belirtildiği gibi *C* ve *Z* gibi durum bitlerinin incelenmesinde *A* ve *B* nin birbirlerine göreceli durumlarının bulunabileceğini gösterin.

Problem 8.26 için çizelge

Bağıntı	Durum bitleri
$A > B$	$C = 0$ ve $Z = 0$
$A \geq B$	$C = 0$
$A < B$	$C = 1$
$A \leq B$	$C = 1$ ve $Z = 1$
$A = B$	$Z = 1$
$A \neq B$	$Z = 0$

8-27. İki işaretli sayı A ve B, 2 nin tümleyeni biçiminde veriliyor. Bunlar çıkartma yoluyla karşılaştırılıyor. Durum bitleri S, Z ve V işlemin sonucuna göre dolu (1) veya temiz (0) dır. (Eğer taşma olursa işaret değişir). A ve B nin göreceli durumunun, durum bitlerinin incelenmesinde aşağıda gösterilen çizelgenin yardımıyla anlaşılacağını gösterin.

Problem 8.27 için çizelge

Bağıntı	Durum bitleri
$A > B$	$(S \oplus V) = 0$ ve $Z = 0$
$A \geq B$	$(S \oplus V) = 0$
$A < B$	$(S \oplus V) = 1$
$A \leq B$	$(S \oplus V) = 1$ ve $Z = 1$
$A = B$	$Z = 1$
$A \neq B$	$Z = 0$

8-28. 4 girişi (C, S, Z ve V) ve 10 çıkışı olan bir devre çizilmek isteniyor. Bu çıkışlar soru 8.26 ve 8.8.27 nin dallanma durumlarını gösteriyor (Eşit ve eşit değil durumu her iki çizelgede de ortak). Gerekli devreyi 2 VEYA, 1 ÖZEL-VEYA kapısı ve 5 tane TERSLEYEN kullanarak çiziniz.

8-29. İki tane 8 bitlik $A = 01000001$ ve $B = 10000100$sayı veriliyor.

a. Her bir sayının ondalık karşılığını (1) bunlar işaretsiz sayılarsa (2) işaretli sayılarsa gösterin

b. Aynı şekilde sayıların (1) işaretsiz (2) işaretli olmaları durumunda toplamı söyleyin

c. Toplama işleminden sonra C, S, Z ve V durum bitlerinin değerlerini verin.

8-30. Bir bilgisayardaki bir program iki işaretsiz sayıyı çıkarma yöntemiyle karşılaştırmakta ve bir durum biti güncellemektedir. $A = 01000001$ ve $B = 10000100$ olsun.

a. Farkı bulun ve ikili düzendeki sonucu açıklayın

b. Durum bitlerinden C (ödünç) ve Z nin değerlerini bulun

c. Çizelge 8.11 deki şartlı dallanma buyruklarından doğru koşulu içerenleri listeleyin.

8-31. Bir bilgisayardaki bir program iki işaretli sayıyı çıkarma yöntemiyle karşılaştırmakta ve bir durum biti güncellemektedir. A = 01000001 ve B = 10000100 olsun.
a. Farkı bulun ve ikili düzendeki sonucu açıklayın
b. Durum bitlerinden C (ödünç) ve Z nin değerlerini bulun
c. Çizelge 8.11 deki şartlı dallanma buyruklarından doğru koşulu içerenleri listeleyin.

8-32. Bellek yığıtının tepesinde 5320 var. Yığıt göstergesi SP = 3560. İki kelimelik bir yordam çağırma buyruğu 1120 bellek adresindedir. Adres alanında 6720 var ve 1121 adresinde bulunuyor. *PC*, *SP* ve yığıtın tepesinin içerikleri nelerdir.
a. Çağırma yordamı bellekten alınıp getirilmeden önce
b. Çağırma buyruğu icra edildikten sonra
c. Yordamdan geri dönüldükten sonra

8-33. Bir dallanma buyruğu, bir alt yordam çağırma buyruğu ve bir kesme işlemi arasındaki temel farklar nelerdir?

8-34. Dış kesme istemleri için ve iç kesme istemleri için beşer örnek verin. Yazılım kesme istemi ile alt yordam çağırma arasında ne fark vardır?

8-35. Bir bilgisayar kesme istemine yanıt olarak yığıta *PC* nin içeriği ile *PSW* yi itiyor. Bundan sonra yeni *PSW* yi *IAD* ile gösterilen bir kesme adresiyle verilen bir bellek adresinden okuyor. Hizmet programının ilk adresi *IAD* + 1 adresinden alınıyor.
a. Kesme çevrimi için mikro işlemler dizisini sıralayın
b. Kesme buyruğundan dönüş için gerekli mikro işlemler dizisini sıralayın

8-36. Değişen buyruk biçimine sahip bilgisayarlar IBM 370, VAX II ve Intel 386 dır. RISC I sabit buyruk biçimini kullanır. Bu iki biçimi karşılaştırın.

8-37. Üç bilgisayar yazaç penceresi kullanıyor. Pencerelerin özellikleri aşağıdadır.Her bir bilgisayardaki pencere boyutunu ve yazaçların sayısını söyleyin.

	Bilgisayar 1	Bilgisayar 2	Bilgisayar 3
Yazaçlar	10	8	16
Yerel yazaçlar	10	8	16
Ortak yazaçlar	6	8	16
Pencere sayısı	8	4	16

8-38. RISC I buyruklarından aşağıdaki işlemlere birer örnek verin.
a. Yazacı azaltsın
b. Yazacı tümlesin

c. Yazacı negatif yapsın
d. Yazacı sıfırlasın
e. İşaretli bir sayıyı 4 e bölsün
f. şlem yok

8-39. RISC I buyruklarını birleştirici dilde kullanarak, eğer 7 (durum) biti 1 ise 3200 adresine atlasın.
a. Derhal kip kullanın
b. Göreceli adres kipini kullanın (*PC* = 3400 varsayın)

KAYNAKLAR

1. Gear, C. W., *Computer Organization and Programming*,. 3rd ed. New York: McGraw-Hill, 1980.
2. Gorsline, G.W., *Computer Organization: Hardware/Software*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1986.
3. Gray, N. A. B., *Introduction to Computer Systems*. Englewood Cliffs, NJ: Prentice Hall, 1987.
4. Hamacher, V. C., Z. G. Vranesic, and S. G. Zaky, *Computer Organization*, 3rd ed. New York: McGraw-Hill, 1990.
5. Hays, J. F., *Computer Architecture and Organization*. 2nd ed. New York: McGraw-Hill, 1988.
6. Langholz, G., J. Francioni, and A. Kandel, *Elements of Computer Organization*. Englewood Cliffs, NJ: Prentice Hall, 1989.
7. Levy, H. M., and R. H. Eckhouse, Jr., *Computer Programming and Architecture:* The VAX-11. Bedford, MA: Digital Press, 1980.
8. Lippiatt, A. G., and G. L. Wright, *The Architecture of Small Computer Systems,* 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1985.
9. Mano, M. M., *Computer Engineering: Hardwire Desing*, Englewood Cliffs, NJ: Prentice Hall, 1988.
10. Murray, W. D., *Computer and Digital System Architecture,* Englewood Cliffs, NJ: Prentice Hall, 1990.
11. Patterson, D. A., and J. L. Hennessy, *Computer Architecture: A Quantitative Approach*. San Mateo, CA: Morgan Kaufmann Publishers, 1990.
12. Patterson, D. A., and C. H. Sequin, "A VLSI RISC ." *IEEE Computer*, September 1982, pp. 8-22.
13. Pollard, L. H., *Computer Desing and Architecture,* Englewood Cliffs, NJ: Prentice Hall, 1990.
14. Rafiquzzaman, M., and R. Chandra, *Modern Computer Architecture* . St.Paul, MN: West Publishers, 1988.

15. Siewiorek, D., C. G. Bell, and A. Newell, *Computer Structures: Principles and Examples*. New York: McGraw-Hill, 1982.
16. Stallings, W., *Computer Organization and Architecture*, 2nd ed. New York:Macmillan, 1989.
17. Tanenbaum, A. S., *Structured Computer Organization*, 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1990.
18. Tomek, I., *Introduction to Computer Organization*. Rockville, MD: Computer Science Press, 1981.
19. Toy, W., and B. Zee, *Computer Hardware/Software Architecture*. Englewood Cliffs, NJ: Prentice Hall, 1986.
20. Ward. S. A., and R. H. Halstead, Jr., *Computer Structures*. Cambridge, MA: MIT Press, 1990.

BÖLÜM 9

Boru Hattı ve Vektör İşlemleri

BU BÖLÜMDE

9.1 Paralel İşleme
9.2 Boru Hattı İşlemleri
9.3 Aritmetik İşlem Boru Hattı
9.4 Buyruk Boru Hattı
9.5 RISC Boru Hattı
9.6 Vektör İşlemleri
9.7 Dizi İşlemcileri

9.1 Paralel İşleme

İşlem Kapasitesi

Paralel işleme geniş bir sınıfı gösteren bir terimdir. Bu terimde kastedilen olaylar aynı zamanda yapılan veri işleme olaylarıdır. Aynı zamanda birden fazla veya çok sayıda işlemin yapılabilmesi bilgisayar isteminin hızını arttırır. Bilgisayarlarda buyruklar sıra ile icra edilir. Paralel işlemede ise aynı anda birçok buyruk icra edilir. Böylece icra hızı artar. Örneğin bir buyruk ALU içinde icra edilince bir sonraki buyruk bellekten okunur. Sistemde 2 veya daha fazla ALU bulunabilir ve aynı anda 2 veya daha fazla buyruk icra edilir. Bundan başka sistemin aynı anda çalışan iki veya daha fazla işlemcisi de bulunabilir. Paralel işlemenin amacı bilgisayarın işlem kapasitesini hızlandırmaktır, böylece giriş, çıkışta hızlanır. Dolayısıyla birim zamandaki işlem miktarı büyür. Paralel işleme ile donanım miktarı ve fiyatta artar. Fakat teknolojik gelişmeler donanım fiyatlarını düşürdüğünden paralel işleme teknikleri uygulanabilir hale gelebilmiştir.

Paralel işleme çeşitli karışıklık açılarından ele alınabilir. En düşük seviyede karışıklık olarak paralel ve seri işlemleri görebiliriz. Kaydırma yazaçları seri olarak çalışırlar ve bir zaman biriminde bir bit işlerler. Pa-

ralel yüklemeli yazaçlar bütün bitleri 1 zamanda işlerler. Daha karışık bir düzeyde paralellik, iş yapan birimlerin çoğaltılmasıyla elde edilebilir. Bu birimler aynı veya farklı işleri yapıyor olabilirler. Fakat aynı anda yaparlar. Paralel işleme verinin iş yapan birimlere dağıtılmasıyla kurulabilir. Örneğin aritmetik, mantıksal ve kaydırma işlemleri 3 ayrı birime ayrılabilir ve veriler bir denetim biriminin yönetiminde her bir birime dağıtılabilir.

Çok Fonksiyonlu Birimler

Şekil 9.1 de paralel çalışan 8 işleme birimine ayrılmanın mümkün bir yolu görülüyor. Yazaçlardaki veriler buyruğun belirlediği birimlerden birine gönderilir. İşlemci birimin yaptığı işlem şekilde her bir birimin üstünde yazılıdır. Toplayıcı ve tamsayı çarpıcı tamsayıların üzerine aritmetik işlemler yaparlar. Kayan noktalı işlemler paralel olarak çalışan 3 devreye parçalanmıştır. Mantıksal kaydırma ve arttırma işlemleri değişik veriler üzerine aynı anda uygulanabilir. Bütün birimler birbirinden bağımsızdır. Dolayısıyla bir sayı kaydırılırken diğer bir sayı arttırılabilir. Bu şekilde çok yönlü bir organizasyon çok karışık bir denetim birimine ihtiyaç duyar. Ancak böylece birçok bileşen arasındaki işlemler koordine edilebilir.

Paralel işlemenin sınıflandırılması için birçok yöntem vardır. İşlemcilerin organizasyonu, işlemciler arası bağlantıların yapısı veya bilginin sistem içinde akışı bir sınıflama yaratabilir. M.J. Flynn tarafından bulunan bir yöntem aynı anda işleme giren buyrukların ve verilerin sayısını göz önüne almaktadır. Bir bilgisayarın normal işlevi buyruğu bellekten alıp getirmek ve buna işlemci içinde icra etmektir. Buyrukların bellekten okunmasıyla bir buyruk akımı oluşur. İşlemci içinde verilerin işlenmesi ise bir veri akımı oluşturur. Paralel işleme buyruk akımında olabilir, veri akımında olabilir veya her ikisinde olabilir.
Bu düşünce ile 4 büyük grup ortaya çıkar.

Tek buyruk akımı, tek veri akımı SISD
Tek buyruk akımı, çok veri akımı SIMD
Çok buyruk akımı, tek veri akımı MISD
Çok buyruk akımı, çok veri akımı MIMD

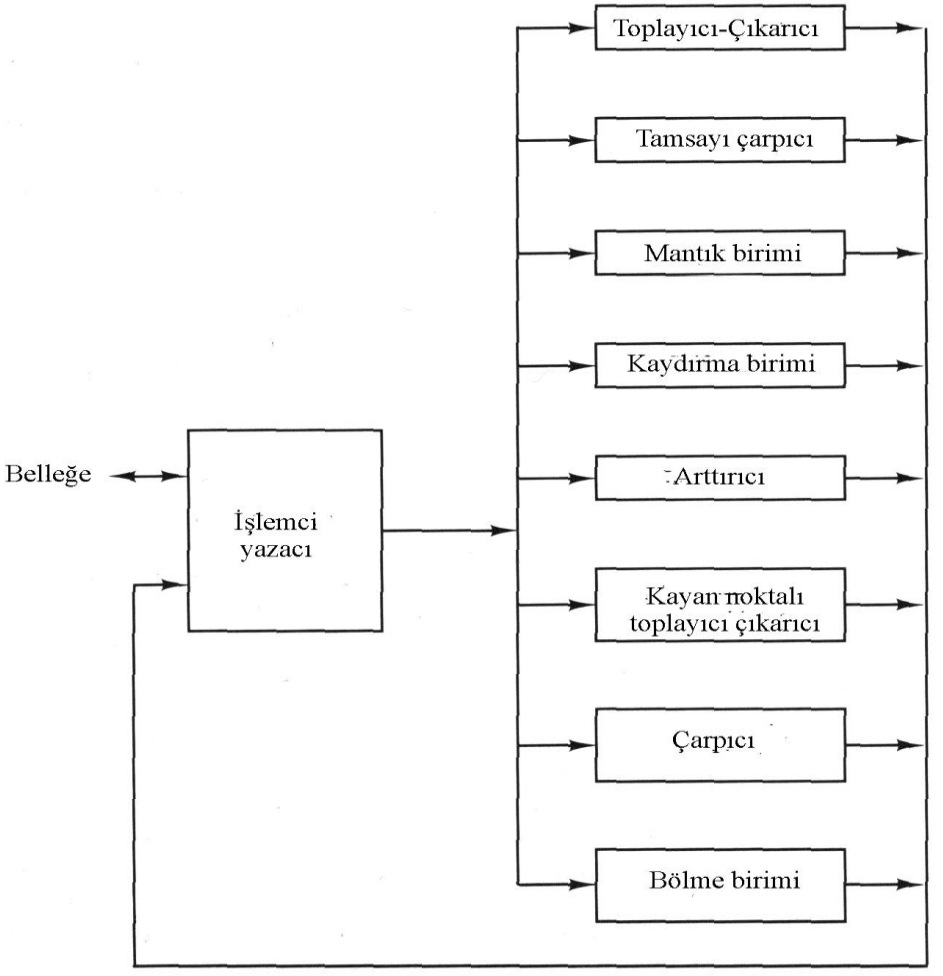
SISD tek bir denetim birimi, bir tane işlemcisi ve bir belleği olan bir bilgisayarı göstermektedir. Buyruklar sırayla icra edilmekte ve sistemde paralel özellikler bulunmamaktadır. Bu durumda paralel işleme, çok sayıda işlem birimi veya boru hattı işlemiyle erişilebilir.

SIMD, MIMD

SIMD durumunda tek bir denetim biriminin yönetimi altında birçok işlem birimi vardır. Bütün işlemciler aynı buyruğu alır fakat farklı verilere uygularlar. Paylaşılan bellek farklı birimler içermelidir. Bu sayede her biri aynı anda teorik olarak önemlidir, çünkü bu yapıda bir bilgisayar

yoktu. MIMD organizasyonu aynı anda farklı programları icra edebilen bir organizasyonu göstermektedir. Birçok çok işlemcili makineler ve çoklu bilgisayar sistemleri bu sınıfa dahildir.

Flynn'ın sınıflandırması denetim birimi ile veri işleme biriminin verimleri arasındaki ayırıma bağlıdır. Böylece bilgisayar sisteminin işlemsel ve yapısal karakteristiklerinden çok davranışsal karakteristiklerini öne çıkarır. Flynn'ın sınıflandırmasına uymayan bir paralel işleme tipi boru hattı dır. Bu sınıflamadan sadece iki sınıf, "SIMD dizi işlemci" bölüm 9.7 de ve "MIMD çoklu işlemciler" bölüm 13 te anlatılacaktır.



Şekil 9.1 8 Çok işlevsel birimli işlemci

Bu bölümde paralel işleme aşağıdaki başlıklar altında incelenecek;

1. Boru hattı işlemleri
2. Vektör işlemleri
3. Dizi işlemcileri

Boru hattı bir yürütme tekniğidir. Aritmetik alt işlemler veya bir buyruğun evreleri icra sırasında üst üste gelirler. Vektör işlemleri büyük vektör veya matris hesaplamalarını çözmeye yöneliktir. Dizi işlemcileri büyük veri dizileri üzerine hesap yaparlar.

9.2 Boru Hattı İşlemleri

Boru hattı işlemi bir yöntem olup sıralı bir işlemi alt işlemlere böler. Her bir alt işlem kendine ayrılmış bir kesimde icra edilir. Diğer kesimlerde de aynı anda başka alt işlemler icra edilir. Boru hattı olayına bunun bir takım işlemlerin bir topluluğu olduğu, her bir alt işlemin bir kesimde bulunduğu şeklinde bakılabilir. İkili bilgiler bu kesimlerden geçtikçe, o kesimde öngörülen alt işlem icra edilir. Her bir kesimde elde edilen sonuç bir sonraki kesime aktarılır. Veri bütün kesimlerden geçince sonuç elde edilir. Boru hattı ismi bilginin akışını gösterir ve endüstrideki montaj bandını andırır. Boru hattının en önemli özelliği farklı hesaplamaların farklı kesimlerde aynı anda yapılıyor olmasıdır. Hesaplamaların bu biçimde üst üste gelebilmesi boru hattı içindeki her kesime bir yazaç bağlamakla mümkün olmaktadır. Yazaçlar kesimler arasında izolasyonu sağlarlar. Böylece her kesim başka veri üzerinde aynı anda işlem yapabilir.

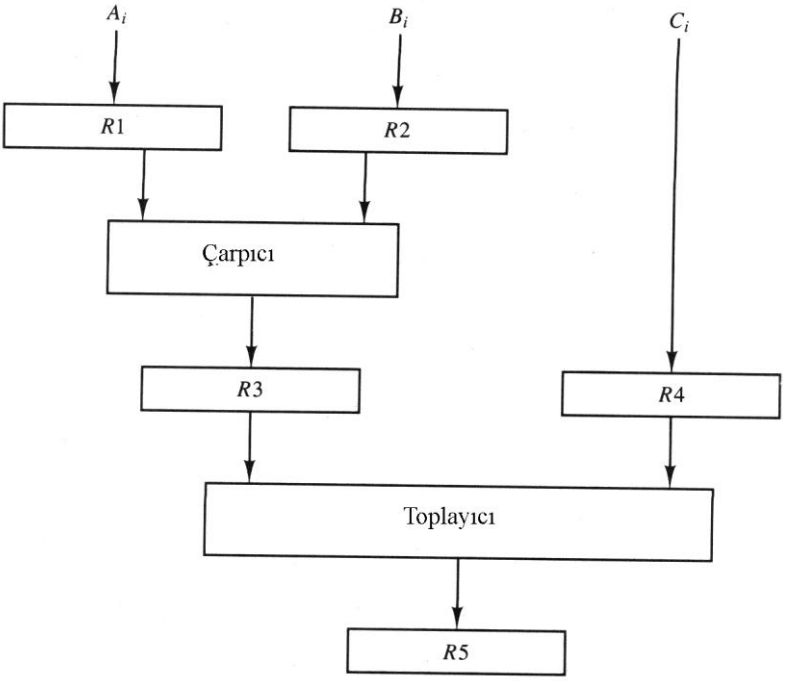
Boru hattı yapısını anlamak için en basit yol, her bir kesimde bir giriş yazacı bulunması ve yazaca bağlı bir birleşik devre bulunduğunu düşünmektir. Yazaç veriyi tutar. Birleşik devre ise alt işlemi yerine getirir. Devrenin çıktısı bir sonraki kesimin giriş yazacına gönderilir. Bütün yazaçlar bir saate bağlı olup, bu kesimde yapılacak iş için yeterli zaman tanıdıktan sonra saat sinyali çalıştırılır. Böylece boru hattı boyunca bilgi her zaman biriminde bir adım olarak akar.

Bir Örnek

Boru hattı organizasyonunu basit bir örnekle gösterelim;

$$A_i * B_i + C_i \quad i = 1, 2, ..., 7$$

olacak şekilde hesaplamak isteyelim. Her alt işlem boru hattı üzerindeki bir kesime yerleştirilmelidir. Her bir kesim bir veya iki yazaca sahiptir ve bir birleşik devresi vardır. Devre şekil 9.2 de gösterilmiştir. $R1$ den $R5$ e kadar olan yazaçlar her saat vuruşunda yeni veri alan yazaçlardır. Çarpıcı ve toplayıcı birleşik devrelerdir. Her bir kesimde icra edilecek alt işlemler aşağıda gösterilmiştir;



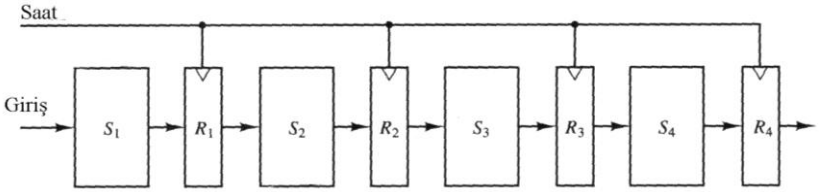
Şekil 9.2 Boru hattı işlemi örneği

$R1 \leftarrow A_i, R2 \leftarrow B_i$	A_i ve B_i yi gir
$R3 \leftarrow R1 * R2, \quad R4 \leftarrow C_i$	çarp ve C_i yi gir
$R5 \leftarrow R3 + R4$	sonuçla C_i yi topla

Beş yazaç her saat vuruşunda yeni veri ile yüklenir. Saat vuruşları ile olan değişim çizelge 9.1 de gösterilmiştir.

Çizelge 9.1 Boru hattı örneğindeki yazaçların içerikleri

Saat vuruş sayısı	kesim1		kesim2		kesim3
	$R1$	$R2$	$R3$	$R4$	$R5$
1	A_1	B_1	—	—	—
2	A_2	B_2	$A_1 * B_1$	C_1	
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6	A_6	B_6	$A_5 * B_5$	C_5	$A_4 * B_4 + C_4$
7	A_7	B_8	$A_6 * B_6$	C_6	$A_5 * B_5 + C_5$
8	—	—	$A_7 * B_7$	C_7	$A_6 * B_6 + C_6$
9	—	—	—	—	$A_7 * B_7 + C_7$



Şekil 9.3 Dört kesimli boru hattı

İlk saat vuruşu A_1 ve B_1 i $R1$ ve $R2$ ye aktarır. İkinci saat vuruşu $R1$, $R2$ çarpımını $R3$ e aktarır ve C_1 i $R4$ e yükler. Aynı saat vuruşu A_2 ve B_2 yi $R1$ ve $R2$ ye aktarır. Üçüncü saat vuruşu 3 kesim üzerinde aynı anda çalışır A_3 ve B_3 e $R1$ ve $R2$ yi yükler, $R1$, $R2$ çarpımını $R3$ e aktarır. C_2 yi $R4$ e yükler ve $R3$, $R4$ toplamını $R5$ e yerleştirir. Boru hattının doldurulması 3 saat vuruşu sürer. Aynı zamanda ilk sonuç $R5$ e gelmiştir. Böylece her saat vuruşu bir çıktı verir ve aynı zamanda boru hattına bir veri yollar. Bu olay yeni veri akışı olduğu sürece böyle sürer. Son veriden sonra saat son çıktı alınana kadar aynı biçimde vurur (olay son verinin verilip son çıktı alınmasına kadar sürer).

Genel Düşünceler

Aynı biçimde alt işlemlere ayrılabilen ve aynı karışıklığa sahip olan her işlem bir boru hattı işlemcisi tarafından işlenebilir. Bu yöntem aynı görevi farklı veri kümeleri için tekrar tekrar yapan uygulamalar için çok uygundur. 4 kesimli bir boru hattı yapısı şekil 9.3 te verilmiştir.

Görev

Veriler bütün kesimlerden sıra ile geçerler. Her bir kesimde bir birleşik devre vardır. Bu devre boru hattından gelen her veriye gerekli alt işlemi uygular. Kesimler R yazaçları ile ayrılmıştır. Bunlar ara sonuçları tutarlar. Bilginin akışı bütün yazaçlara aynı anda uygulanan saat vuruşu ile sağlanır. Boru hattı içindeki bütün kesimlerde sıra ile uygulanan işleme *görev* adı verilmektedir.

Uzay-zaman Diyagramı

Bir boru hattının davranışı en iyi uzay-zaman diyagramı ile gösterilir. Bu diyagram kesimlerin kullanılışını zamanın fonksiyonu olarak gösterir. 4 kesimli boru hattı için uzay-zaman diyagramı şekil 9.4 te gösterilmiştir. Yatay eksen saat vuruşlarında zaman eksenidir. Dikey eksen kesim numarasıdır. Diyagram T_1 den T_6 ya kadar 4 kesimde icra edilen 6 görevi göstermekte. Başlangıçta T_1 görevini kesim 1 yerine getirir. İlk saat vuruşundan sonra kesim 2 T_1 ile meşgulken kesim 1 T_2 göreviyle meşguldür. Böyle devam ederek T_1 görevi 4 saat vuruşu sonra tamamlanır. Bu andan itibaren her saat vuruşu sonrasında bir görev tamamlanır. Sistemde kaç kesim olduğu, boru hattı dolu olduğu sürece hiç önemli değildir. Çünkü her saat vuruşu sonunda bir görev biter.

kesim	1	2	3	4	5	6	7	8	9	saat
1	T_1	T_2	T_3	T_4	T_5	T_6				döngüleri
2		T_1	T_2	T_3	T_4	T_5	T_6			
3			T_1	T_2	T_3	T_4	T_5	T_6		
4				T_1	T_2	T_3	T_4	T_5	T_6	

Şekil 9.4 Boru hattı için uzay-zaman diyagramı

Daha genel olarak n tane görevin k kesimli boru hattı içinde t_p saat vuruşu ile işlenmesine bakılırsa; Birinci görev $k.t_p$ kadar zaman ister. Çünkü k tane kesimi her biri t_p zamanında geçecektir. Toplam süre $k.t_p$ olur. Kalan n-1 tane görev her bir saat vuruşu içinde bir tane olarak yapılacağından gerekli zaman $(n-1).t_p$ olacaktır. O halde n görevin k kesimde tamamlanması için $k + (n-1)$ zaman gerekir. Örneğin şekil 9.4 te 4 kesim 6 görev gösteriliyor. Bütün işlemlerin bitmesi için gerekli zaman $4 + (6-1) = 9$ saat çevrimi olur.

Şimdi boru hattı olmayan bir bilgisayar düşünelim. Her bir görevi bitirmesi için gerekli zaman t_n olsun. n görev için gerekli zaman $n.t_n$ olacaktır. Bu durumda boru hattının hızlandırması S

$$S = \frac{nt_n}{(k+n-1)t_p}$$

ile verilir. Görev sayısı artınca n, $k-1$ den çok büyük olur. $k+n-1$ de n e yaklaşır. Bu durumda oran

$$S = \frac{t_n}{t_p}$$

olur. Eğer boru hattı ve boru hattı olmayan bilgisayarlarda bir görevin tamamlanması için geçen sürelerin aynı olduğunu varsayarsak $t_n = k.t_p$ olacağından

$$S = \frac{kt_p}{t_p} = k$$

bulunur. O halde bir boru hattının toplam hızlandırması teorik olarak k maksimum değerini almaktadır. Buda boru hattı içindeki kesim sayısıdır.

Hızlanma oranı için bir örnek verilirse; Bir alt işlemin her kesimde işlenmesi için gerekli zaman $t_p = 20$ ns olsun. Eğer boru hattı 4 kesimli ise ve sırayla 100 görev icra edecekse boru hattı sistemi $(k+n-1)t_p = (4 +$

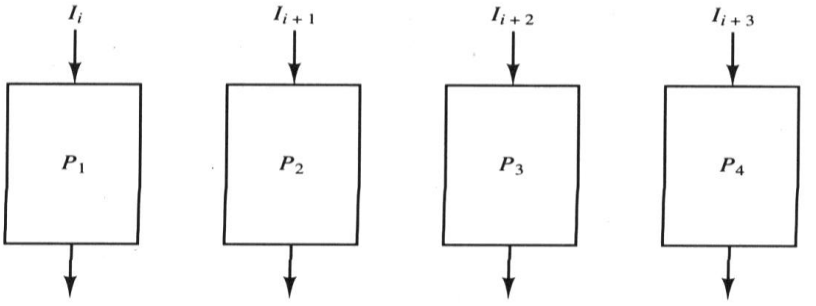
99) × 20 = 2060 ns. $t_n = kt_p = 4 \times 20 = 80$ ns boru hattı olmayan bilgisayar için bir görev zamanı ve tüm görev için $nkt_n = 100 \times 80 = 8000$ ns. Buradan hızlanma oranı $S = 8000 / 2060 = 3.88$. Görev sayısı arttıkça oran 4 e yaklaşır. Yani kesim sayısı eğer t_n =60 ns sayarsak hızlanma oranı 60 / 20 = 3 olur.

Bu hız avantajından yararlanmak için k tane eşdeğer birim yapılarak paralel çalıştırılabilir. Bir tane k kesimli boru hattı, boru hattı olmayan birimden k tanesinin verimine eşit verimde çalışır. Bu durum şekil 9.5 te gösterilmiştir. Burada 4 tane eşdeğer devre paralel bağlanmıştır.

Her bir P devresi eşdeğer boru hattı devresinin işini görür. Bunlar 4 veriyi aynı anda alır ve aynı anda işlemi bitirirler. Sadece hız düşünüldüğünde bu biçim 4 kesimli boru hattına eşittir. Şekil 9.5 teki 4 birimli devre tek buyruk çok veri (SIMD) organizasyonudur. Çünkü aynı buyruk çok veri üzerine paralel olarak uygulanmaktadır.

Boru hattının teorik maksimum hızda çalışmaması için birçok neden vardır. Farklı kesimler kendi alt işlemlerini tamamlamak için farklı zamanlar harcarlar. Saat zamanı bu gecikmeleri dikkate alarak seçilmelidir. Bu durumda diğer kesimler geciken kesimi beklerken boşuna zaman kaybederler. Ayrıca boru hattı olmayan bir devrenin, boru hattındaki devreyle eş zamanlı çalışacağı ve aynı miktarda gecikeceği her zaman doğru değildir. Tek bir devrede, bizim kesimler arasında kullandığımız yazaçlara gerek olmayabilir. Bütün bunlara rağmen boru hattı tekniği, teorik maksimum hıza erişmese bile hızlı işlem yapmaktadır.

Boru hattı tekniğinin uygulanabileceği iki alan vardır. Aritmetik boru hattı aritmetik işlemleri alt işlemlere böler. Buyruk boru hattı ise al-getir, kod çöz ve icra evresiyle üst üste gelen bir buyruk akımı içinde çalışabilir. Bu iki tip boru hattı aşağıda açıklanmıştır.



Şekil 9.5 Paralel bağlanmış çok fonksiyonlu birim.

9.3 Aritmetik İşlem Boru Hattı

Boru hattı aritmetik birimlere yüksek hızlı bilgisayarlarda rastlanmaktadır. Çoğunlukla kayan noktalı işlemler ve sabit sayı çarpımları ve bilimsel çalışmalardaki benzer hesaplamalarda kullanılır. Bir boru hattı çarpıcı bir "dizi" çarpıcı olup şekil 10.10 da çizilmiştir. Bunun özel toplayıcıları vardır. Kısmi çarpımların elde si sırasında elde yayılma zamanını minimize etmek üzere tasarlanmıştır. Bölüm 10.5 te gösterildiği üzere kayan noktalı işlemler kolayca alt işlemlere ayrılabilir. Şimdi kayan noktalı toplama ve çıkarma işlemi için bir boru hattı birimi vereceğiz.

Kayan noktalı toplayıcının girişleri iki tane normalize kayan noktalı ikili sayı olsun.

$$X = A \times 2^{a}$$

$$Y = B \times 2^{b}$$

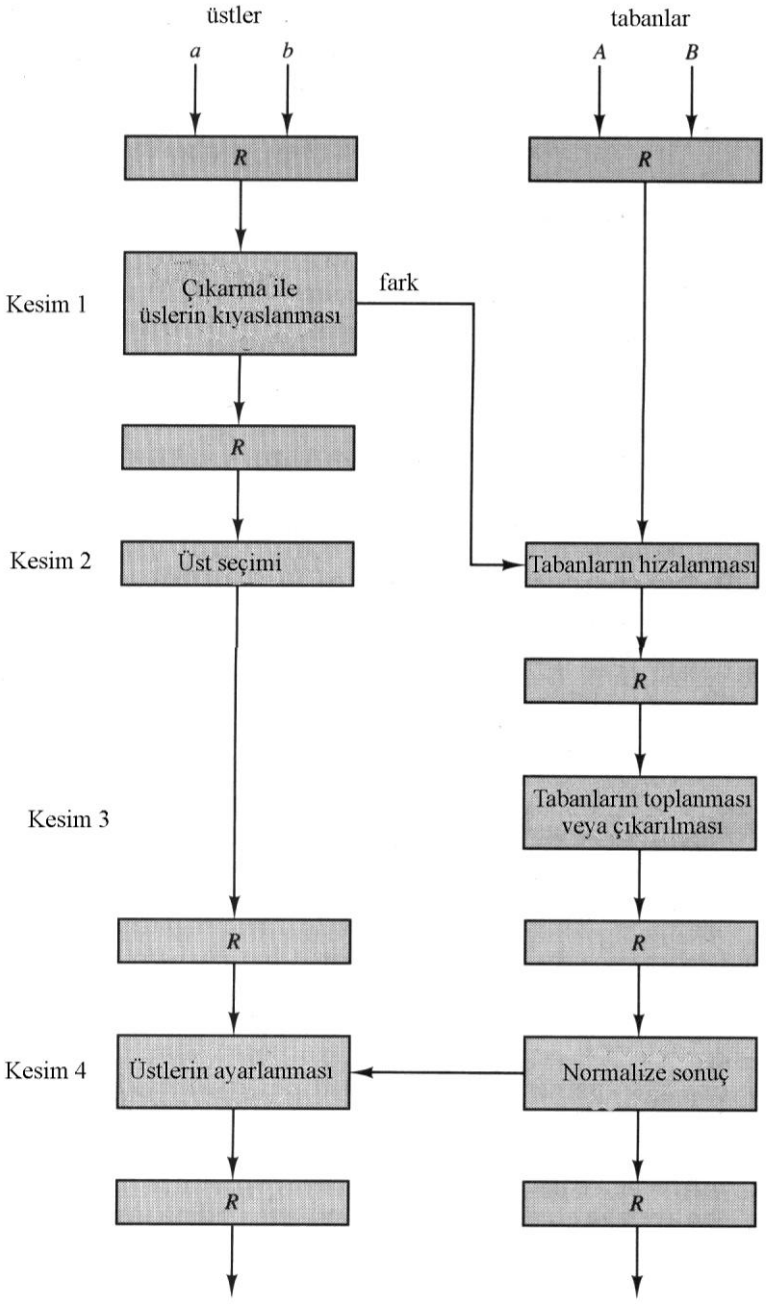
A ve *B* iki kesirleri gösteriyor, *a* ve *b* ise üstürler. Kayan noktalı toplama ve çıkarma 4 kesimde icra edilebilir (şekil 9.6). *R* adı verilen yazaçlar kesimler arasına yerleştirilmiş olup ara sonuçları tutarlar. 4 kesimle yapılan alt işlemler.

1. Üslerin karşılaştırılması
2. Kesirlerin hizalanması
3. Kesirlerin toplanması veya çıkarılması
4. Sonucun normalize edilmesi

Bu durum şekil 10.15 teki akış şemasına uygundur. Fakat bazı alt işlemlerde icra zamanını azaltmak için değişiklik yapılmıştır. Üsler birbirinden çıkarılarak mukayese edilmiş ve farkları bulunmuştur. Daha büyük üs sonucun üssü olarak kabul edilmiştir. Üslerin farkı, küçük üssü olan sayının kesrinin kaç defa sağa kaydırılacağını belirler. Bu şekilde iki kesir hizaya sokulur. Kayma işlemini kısaltmak için kaydırma birleşik devre olarak yapılmalıdır.

İki kesir kesim 3 te toplanır veya çıkarılır. Sonuç kesim 4 te normalize edilir. Eğer bir taşma oluşursa toplamın veya farkın kesri sağa kaydırılır ve üs bir arttırılır. Eğer sayı sıfır olmadığı halde sıfır görünürse kesir içinde soldaki sıfırların sayısı, sola kaydırma sayısını verir. Üsten kaydırma sayısı çıkarılmalıdır.

Aşağıdaki sayısal örnek her bir kesimdeki alt işlemleri belirgin hale getirecektir. Basit olması için ondalık sayılar kullanılmakta. Şekil 9.6 ikili düzendeki sayılar içindir. Normalize edilmiş iki tane kayan noktalı sayı düşünelim.



Şekil 9.6 Kayan noktalı sayılarda toplama ve çıkarma için boru hattı

$$X = 0.9504 \times 10^3$$

$$Y = 0.8200 \times 10^2$$

İlk kesimde üsler çıkarılarak 3 – 2 = 1 elde edilir. Büyük üs 3 sonucun üssü olarak alınır. Bir sonraki kesim *Y* nin kesrini sağa kaydırır ve

$$X = 0.9504 \times 10^3$$

$$Y = 0.0820 \times 10^3$$

elde edilir. Bu şekilde iki kesir aynı üsse sahip hale gelirler. Kesim 3 te iki kesir toplanır ve

$$Z = 1.0324 \times 10^3$$

Bulunur. Bu sonuç noktadan sonra ilk basamak sıfır olacak şekilde sağa bir kaydırılarak ve üs bir arttırılarak normalize edilmiş toplam

$Z = 0{,}10324 \times 10^4$ elde edilir.

Karşılaştırma devresi, kaydırıcı, toplayıcı-çıkarıcı, arttırıcı ve azaltıcı devreler birleşik devreler şeklinde kayan noktalı sayılar için boru hattı içinde bulunur. Şimdi 4 kesimde harcanan zaman $t_1 = 60$ns $t_2 = 70$ns $t_3 = 100$ns ve $t_4 = 80$ns ve ara yüzey yazaçları $t_r = 10$ns harcasınlar. $t_p = t_3 + t_r = 110$ns olur. Boru hattı olmayan bir makinede $t_n = t_1 + t_2 + t_3 + t_4 + t_r = 320$ns olur. Bu taktirde hızlanma oranı 320 / 110 = 2.9 olur.

9.4 Buyruk Boru Hattı

Boru hattı işlemi sadece veri akımı üzerinde olmaz. Buyruk akımı üzerinde de uygulanabilir. Bir buyruk boru hattı ardışık buyrukları bellekten okur ve aynı zamanda daha önceki buyrukları kesimler içinde işler. Bu şekilde al-getir ve icra evreleri üst üste gelebilir. Aynı anda birden fazla buyruk icra edilir. Burada bir sorun buyruğun bir dallanmaya neden olmasıdır. Bu durumda dallanma buyruğundan sonraki buyruklar iptal edilir ve boru hattı boşaltılır.

Bir bilgisayar düşünülsün; iki kesimli olsun ve kesimleri *al-getir* ve *icra* olsun. Buyruk al-getir kesimi FIFO şeklinde oluşturulabilir. Bu bir kuyruk olup yığıt biçimi değildir. İcra biriminin belleği kullanmadığı anlarda denetim birim program sayacını arttırır ve ardışık buyrukları bellekten okur. Buyruklar FIFO buffer ına yerleştirilir. Böylece ilk giren ilk çıkar şeklinde işleme girerler. Yani buyruklar bir kuyruğa sokulmuş olur. Bu kuyrukta kodlarının çözülmesi ve icra kesimi tarafından işlem için beklerler. Buyruk akıntısı kuyruğu mekanizması, buyrukların bellekten

okunması için ortalama erişim süresini kısaltan etkin bir yol oluşturur. FIFO buffer ında yer olunca, denetim birimi yeni bir buyruğu al-getir evresiyle getirir. Buffer, buyrukların icra birimine gitmeleri için kuyruk gibi çalışır.

Buyruk Çevrimi

CISC tipi bilgisayarlarda buyruklar al-getir ve icra evreleri dışında da evrelere gerek duyarlar. En genel durumda bilgisayar her bir buyruğu işlemek için aşağıdaki adımlara gerek duyar.

1. Buyruğu bellekten al-getir
2. Buyruğun kodunu çöz
3. Etkin adresi hesapla
4. Verileri bellekten al-getir
5. Buyruğu icra et
6. Sonucu uygun bir yere yaz

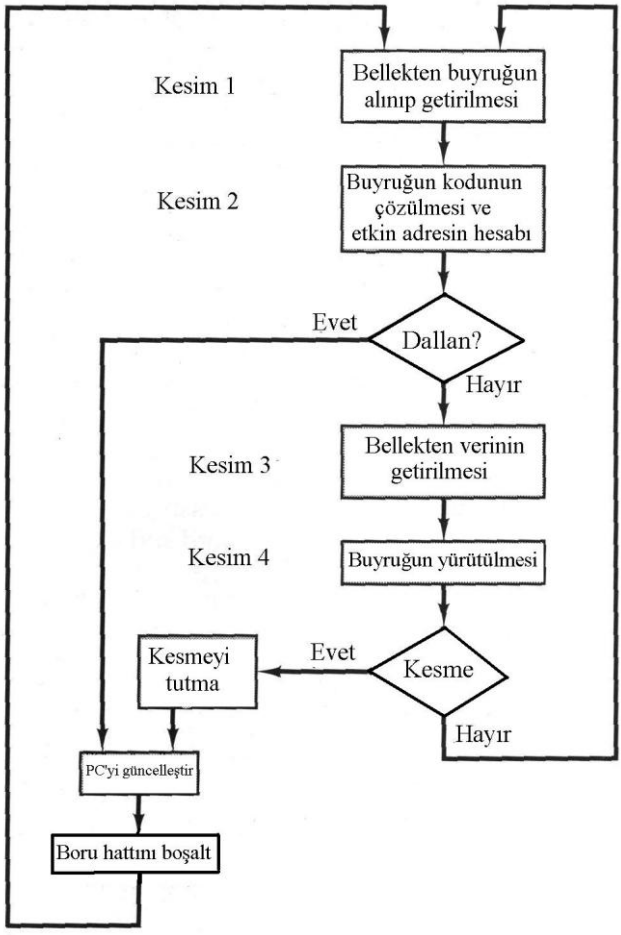
Buyruk boru hattının maksimum hızda çalışmasını önleyen bazı zorluklar vardır. Farklı kesimler gelen bilgiler üzerine farklı zaman sürelerinde işlem yaparlar. Bazı kesimler bazı işlemlerde atlanır. Örneğin yazaç adreslemeli bir buyruk etkin adres hesabı gerektirmez. İki veya daha fazla kesim aynı anda belleğe erişmek isterler. Bu durumda bir kesim öbürünü bekler. Belleğe erişim sorunları bazen, farklı modüllerdeki veri ve buyruklara erişmekte iki bellek veri yolu kullanılarak çözülebilir. Bu yolla bir buyruk kelimesi ve bir veri kelimesi aynı anda farklı modüllerden okunabilir.

Buyruk boru hattının tasarımı eğer buyruk evreleri eşit süreli yapılabilirse, yani eşit sürede işlem yapabilecek kesimlere ayrılabilirse en etkili tasarım elde edilmiş olur. Bu süreler buyruğa ve bunun işlenmesinde kullanılan yola bağlıdır.

Örnek: Dört Kesimli Buyruk Boru Hattı

Bu örnekte buyruk çözümleme işi ile, etkin adresin hesabının aynı kesimde yapıldığını varsayalım. Ayrıca buyrukların icra sonuçlarının bir yazaçta tutulduğunu varsayalım. Böylece buyruğun icrası ile sonucun bir yere yerleştirilmesi tek bir kesimde gerçekleşebilir. Bu varsayımlarla buyruk boru hattını 4 kesime indirebiliriz.

Şekil 9.7 dört kesimli bir boru hattında işlenebilecek buyruk evresini gösteriyor. Bir buyruk 4 üncü kesimde icra edilirken bir sonraki buyruk 3 üncü kesimde bir veri alıp getirmektedir. 3 üncü buyruk için etkin adres ayrı bir aritmetik devrede hesaplanabilir. Bellek müsait olduğunda 4 üncü buyruk ve bunu takip eden bütün buyruklar al-getir evresiyle getirilirler. Böylece 4 alt işlem üst üste gelir ve 4 buyruk aynı anda icra ediliyor olur.



Şekil 9.7 Dört kesimli MİB boru hattı

Zaman içinde sıradaki buyruk bir program denetim buyruğu olsun ve bir dallanma göstersin. Bu durumda son iki kesimdeki işlemler tamamlanmış ve buyruk buffer ındaki bütün bilgiler silinmiş olur. Boru hattı *PC* a yazılan adresten itibaren yeniden buyruk almaya başlar. Benzer şekilde bir kesme isteği alındığında boru hattının boşaltılması ve yeniden yeni adres değerinden başlanması gerekir.

Şekil 9.8 buyruk boru hattının çalışmasını gösteriyor. Yatay eksende zaman eşit süreli aralıklara adımlara bölünmüştür. 4 kesim uygun kısaltmalarla gösterilmiştir.

1. FI buyruğu al-getir
2. DA buyruk kodunu çöz ve etkin adresi hesapla

Adım		1	2	3	4	5	6	7	8	9	10	11	12	13
Buyruk	1	FI	DA	FO	EX									
	2		FI	DA	FO	EX								
(Dallanma)	3			FI	DA	FO	EX							
	4				FI	–	–	FI	DA	FO	EX			
	5					–	–	–	FI	DA	FO	EX		
	6									FI	DA	FO	EX	
	7										FI	DA	FO	EX

Şekil 9.8 Buyruk boru hattı zamanı.

3. FO veriyi al-getir
4. EX buyruğu icra et

İşlemcinin ayrı buyruk ve veri bellekleri bulunduğunu ve ayrı veri yolları olduğunu varsayacağız. Dolayısıyla FI ve FO aynı anda çalışabilirler. Dallanma olmadığı sürece her bir kesim farklı işlemler üzerinde çalışır. Dolayısıyla 4 üncü adımda 1 inci buyruk kesimde icra edilmektedir. Aynı adımda 2ci buyruğun verisi getirilmekte (FO), buyruk 3 ün kodu çözülmekte (DA) ve 4 üncü buyruk ise FI kesimine bellekten getirilmektedir.

Şimdi 3 üncü buyruğun dallanma olduğun varsayalım. Bu buyruğun DA kesiminde kodu çözülür (4 üncü adım). Bu durumda FI dan DA ya buyruk geçişi durur. Dallanma buyruğu altıncı adımda icra edilene dek FI dan DA ya geçiş olmaz. Yeni buyruk adım 7 de getirilir. Eğer dallanma olmaz ise 4 üncü adımda getirilmiş bulunan buyruk kullanılabilir. Eğer dallanma olur ise yani program normal akış sırasının dışına çıkarsa yeni bir buyruk getirilir. (PC den yeni adresi okunarak). Bundan sonra boru hattı yeni bir dallanmaya kadar aynen devam eder.

Başka bir gecikme EX kesimi sonucu belleğe yazmak ister ve aynı zamanda FO bir bellek verisini getirmek isterse meydana gelir. Bu durumda EX kesimi önce işini bitirir FO bekler.

Boru Hattı Çatışması

Buyruk boru hattında normal işlemlerde genel olarak 3 büyük zorluk vardır.

1. *Kaynak çatışması;* İki kesim aynı anda belleğe erişmek ister. Bu iki ayrı veri yolu ve iki bellek kullanarak çözülebilir.
2. Veri beklentisi; Bir buyruk bir önceki buyruğun sonucunu kullanacaktır veya sonuca bağlıdır.

3. *Dallanma zorluğu;* Dallanma veya diğer buyrukların PC ı değiştirmesinden kaynaklanır.

Veri Beklentisi

Bir veri çarpışması bir buyruğun bir önceki buyruğun bazı işlemleri bitirmemiş olmasından kaynaklanan gecikmedir. Çünkü bu buyruğun verisi henüz hazır değildir. Örneğin FO kesiminde bir buyruk aynı anda EX kesiminde oluşarak veriyi beklemektedir. Dolayısıyla ilk buyruğun veriyi bir yere yazmasından sonra onu kullanabilir. Aynı biçimde etkin adres hesaplanamamaktır. Çünkü gerekli bilgi henüz hazır değildir. Örneğin yazaç dolaylı kipinde bir buyruk veriyi alıp getirmek için önceki buyruğun adresi yazaca yüklemesini beklemek zorundadır. Böylece verinin belleğe gitmesi adres gelene kadar bekleyecektir. Boru hattı ile çalışan bilgisayarlar bu zorluklara karşı bir takım yollarla çözümler getirmişlerdir.

Donanım Kilitleri

En kestirme yol donanım kilitleri kullanmaktır. Kilit bir devredir. Bu devre verilerin kaynakları boru hattı içinde ilerde olan buyrukları bulur. Sorun çözülene kadar buyruk geciktirilir. Bu yaklaşım istenen gecikmeleri sağlar ve bu iş için donanımı kullanır.

Veri İlerlemesi

Bir diğer yöntem veri ilerlemesi adını taşır. Bu iş için özel donanım kullanılır ve bir sıkışma durumu bulunursa boru hattı kesimleri arasında veri özel yollarla kaydırılır veya yer değiştirilir. Örneğin bir ALU sonucunun hedef yazacına yazılmasından önce bu donanım sonucun hedefini belirler ve eğer bu sonuç bir sonraki buyruğun kullanacağı bir veri ise yazaç kümesi atlanır. Sonuç ALU nun girişine yollanır. Bu yöntem sıkışmayı bulacak bir donanımı ve seçiciler arasında donanımı gerektirir.

Gecikmiş Yükleme

Bazı bilgisayarlarda kullanılan yöntem bu tür sıkışmalarda sorumluluğun derleyiciye verilmesidir. Derleyici bu sıkışmaları bulur ve buyrukların sırasını öyle bir şekilde değiştirir ki sıkışmaya neden olan verinin yüklenmesini araya işlem yapmayan buyruklar koyarak geciktirir. Bu olaya gecikmiş yükleme denir. Bunun örneği gelecek kısımda verilmiştir.

Dallanma İşlemlerinin İdaresi

Bir buyruk boru hattı işletmekteki en büyük zorluklardan biriside dallanma buyruklarının gelmesidir. Bir dallanma buyruğu şartlı veya şartsız olabilir. Şartsız dallanma programın akışını daima değiştirir. Bunun içinde PC a yeni bir hedef adresi yükler. Şartlı dallanmada eğer şart sağlanırsa denetim birim hedef adresi seçer. Sağlanmazsa sıradaki bir sonraki buyruğa gider. Daha önce söylendiği gibi bir dallanma programın normal akış sırasını bozar. Bu nedenle buyruk boru hattı içinde işlemleri zorlaştı-

rır. Boru hattı olan bilgisayarlar dallanmadan dolayı meydana gelen gecikmeleri önlemek için muhtelif donanım yöntem ve teknikleri kullanırlar.

Hedef Adresli Buyruğun Önceden Seçilmesi

Şartlı dallanmayı idare etmek için bir yol dallanma buyruğundan sonraki buyruk ile birlikte hedef buyruğunda alınıp getirilmesidir. Her iki buyrukta dallanma buyruğu icra edilene kadar korunurlar ve sonra hangisine dallanma olduysa yani eğer şart sağlandıysa boru hattı hedef buyruk ile devam eder. Bu yöntemin bir ilerisi dallanma kararı verilene kadar her iki taraftan buyruk alıp getirmektir. Zaman, gelince denetim birimi doğru buyruk akımını seçer.

Dallanma Hedef Bufferı

Bir diğer olasılık dallanma hedef bufferı (BTB) kullanmaktır. BTB bir yardımcı bellek alanıdır. (Bölüm 12.4). Boru hattının al-getir kesiminde bulunur. BTB içindeki her giriş daha önce icra edilmiş olan dallanma buyruğu ile bunun hedef buyruğunu içerir. Bu ayrıca hedef buyruğundan sonraki birkaç buyruğu da içerir. Boru hattı bir dallanma buyruğun kodunu çözdüğünde BTB yardımcı belleği tarar ve buyruğun adresini arar. Eğer buyruk BTB içinde ise önceden al-getir olayı yeni yönde devam eder. Eğer buyruk BTB içinde değilse boru hattı yeni bir buyruk akımına geçer ve hedef buyruğu BTB ye yerleştirir. Bu yöntemin avantajı daha önceden icra edilmiş olan buyruğun hazır olması ve hiçbir gecikme olmadan oraya geçilebilmesidir.

Döngü Bufferı

BTB nin bir değişik şekli döngü bufferidir. Bu çok küçük fakat hızlı bir yazaç kümesidir. Boru hattının buyruk al-getir kesimi tarafından çalıştırılır. Program içinde bir döngü tespit edildiğinde bütünüyle döngü buffera yerleştirilir. Bütün dallanmaları da buraya yerleştirilir. Program döngüsü bitene kadar doğrudan doğruya bu bufferdan icra edilir. Belleğe gitmeye gerek kalmaz. Döngü bitince ve son dallanma yapılınca bufferın işi bitmiş olur.

Dallanma Tahmini

Bazı bilgisayarların kullandığı bir yöntem dallanmayı önceden tahmin etmektir. Dallanma tahmini yapan boru hatlarında ilave mantık devrelerine gerek vardır. Bir dallanma buyruğu geldiğinde bu buyruk daha icra edilmeden nereye dallanacağı tahmin edilir. Boru hattı bu durumda tahmin edilen buyrukları önceden alıp getirir. Doğru tahmin zaman kaybını önler.

Gecikmeli Dallanma

RISC işlemcilerin çoğunda kullanılan yöntem gecikmeli dallanmadır. Bu yöntemde derleyici dallanma buyruklarını bulur ve makine dili programı yeniden düzenler. Düzenlemede dallanma buyruğunun sonucu belli olup dallanma adresi anlaşılana kadar araya faydalı buyruklar koyar ve kesilme olmadan boru hattının çalışmasını temin eder. Bunun örneği bir sonraki kısımda verilecektir.

9.5 RISC Boru Hattı

RISC e ait özelliklerden bir tanesi buyruk boru hattının etkin kullanımıdır. Buyruk kümesinin basit ve kısa oluşu boru hattının içindeki alt işlemlerin az sayıda olmasını ve bir zaman biriminde icra edilmesini sağlar. Sabit uzunluklu buyruk biçiminden dolayı işlemin kodunun çözülmesi ile yazaç seçimi aynı anda yapılabilir. Bütün veri işlemleri yazaçtan yazaca işlemlerdir. Bütün veriler yazaçlarda olduğundan dolayı etkin adresin hesaplanması ve verinin bellekten getirilmesi işlemlerine gerek yoktur. Bundan dolayı buyruk boru hattı 2 veya 3 kesime sığar. Bir kesim buyruğu bellekten alır. Diğeri bunu ALU da icra eder. Üçüncü kesim sonuçlarının hedef yazaca yazılmasını sağlayabilir.

RISC içinde veri aktarım buyrukları yükleme ve depolama buyruklarıdır. Bu buyruklar yazaç dolaylı adresleme kullanırlar. Boru hattı içinde 3 veya 4 adım kullanırlar. Bellekten bir buyruğun getirilmesiyle bir verinin getirilmesi veya sonucun belleğe yerleştirilmesi sırasındaki çarpışmalardan kaçınmak için RISC makineler çoğunlukla iki veri yolu ve iki bellek kullanırlar. Birisi buyrukların tutulması diğeri veriler içindir. Bu iki bellek CPU saatiyle aynı hızda birbiriyle eşzamanlı olarak çalışabilirler. Bunlara ön bellekler denir.

Tek Çevrimli Buyruk İcrası

Bölüm 8.8 de söylendiği gibi RISC in en önemli avantajı bir buyruğu bir zaman çevrimi içinde icra edebilmesidir. Bir buyruğun bir zaman çevrimi içinde bellekten alınıp icrası düşünülmez. Bunun yerine her bir zamanda yeni bir buyruğun icrasını başlatmak aynı sonucu verir. RISC in CISC e göre avantajı RISC in boru hattı kesimlerine bir zaman biriminde erişebilmesidir. CISC kendi boru hattı içinde çok sayıda kesim kullanır. En uzun kesim iki veya daha fazla zaman birimine gerek duyar.

Derleyici Desteği

RISC in bir diğer karakteristiği derleyicinin desteğidir. Derleyici yüksek seviyeli programı makine dili programa çevirir. RISC işlemcilerinde veri çarpışmaları ve dallanma zorlukları bulup gecikmeler en az olacak şekilde programların düzenlemesi istenir. Aşağıda bir derleyicinin bu zorluklardan kurtulmak için makine dili programı nasıl en iyi biçime çevirdiği örneklerle verilmiştir.

Örnek: Üç Kesimli Boru Hattı

Bir RISC işlemcisi için tipik buyruklar çizelge 8.12 de verilmişti. Bu çizelgeden görüldüğü gibi 3 tip buyruk vardır. Veri işleme buyrukları veriler üzerine işlemci yazaçları içinde işlem yaparlar. Veri aktarım buyrukları yükleme ve depolama olup etkin adresi hesaplayarak veriyi getirir

veya götürürler. Etkin adres buyruğu içindeki bir yer değiştirme miktarı bir yazaç değerinin toplanması veya iki yazaç değerlerinin toplanması ile elde edilir. Program denetim buyrukları yazaç değerlerini ve bir sabit yardımıyla dallanma adresini bulur ve PC a yazarlar.

Böyle bir bilgisayar için dananım işlemleri düşünülürse, denetim kısmı buyruğu program belleğinden bir buyruk yazacına alıp getirir. Bu buyruğun kodu çözülürken aynı zamanda buyruğun icrası için gerekli yazaçların seçimi yapılır. İşlemci birimi içinde yazaçlar ve ALU vardır. Bu birim gerekli aritmetik, mantıksal ve kaydırma işlemlerini yapar. Bir veri belleği, yazaç kümesi içindeki bir yazaçtan belleğe veya tersi yükleme ve depolama işlemlerini yapar. Buyruk evresi 3 alt işleme bölünebilir ve 3 kesime yerleştirilebilir.

I: Buyruğu al-getir

A: ALU işlemi

E: Buyruğun icrası

I kesimi buyruğu program belleğinden alır getirir. A kesiminde buyruğun kodu çözülür ve ALU işlemi yapılır. ALU 3 farklı fonksiyon için kullanılır. ALU da kodu çözülen buyruk, ya bir veri işleme buyruğundaki işlemi yapar, ya yükleme ve depolama buyruğu için etkin adresi hesaplar yada program denetim buyruğu için dallanma adresini hesaplar. E kesimi ALU çıkışını buyrukça belirlenen üç yönden birine doğru yönlendirir. Bu üç yön yazaç kümesi içindeki hedef yazacına ALU işleminin sonucunun aktarılması, yükleme ve depolama için etkin adresin veri belleğine aktarılması, dallanma adresinin *PC* a yazılması.

Gecikmeli Yükleme

Aşağıdaki 4 işleme bakıldığında

1. LOAD: $R1 \leftarrow M[\text{adres 1}]$

2. LOAD: $R2 \leftarrow M[\text{adres 2}]$

3. ADD: $R3 \leftarrow R1 + R2$

4. STORE: $M[\text{Adres 3}] \leftarrow R3$

Eğer 3 kesimli boru hattı kesiksiz çalışırsa 3 üncü buyrukta bir veri çatışması olacaktır. Çünkü *R*2 deki veri henüz A kesimi içinde değildir. Bu olay şekil 9.9(a) daki boru hattı zamanlamasından görülebilir. 4 üncü zaman evresinde E kesimi bellek verisini *R*2 ye yerleştirmektedir. A ke-

simi 4 üncü evresinde *R*2 deki veriyi kullanmaktadır. Fakat bu veri yanlıştır, çünkü henüz bellekten aktarılmamıştır. Bu durum derleyiciye bağlıdır ve derleyici buyruğun kullanacağı verilerin bellekten alındığından emin olmalıdır. Eğer derleyici yükleme işleminden sonra kullanacağı faydalı bir buyruk bulamazsa buraya işlem yok buyruğu yerleştirir. Bu tip bir buyruk bellekten alınır fakat bir işlem yapmaz. Dolayısıyla bir zaman evresini ziyan eder. Bu şekilde verinin bellekten yüklenmesine gecikmiş yükleme adı verilir.

Şekil 9.9(b) de aynı programın işlem-yok buyruğunun *R*2 nin yüklenmesinden sonra yerleştirilmiş şekli görülüyor. Topla buyruğu *R*2 değerini 5 ci adımda kullanıyor. Burada arada işlem-yok buyruğunun kullanılması ile toplama işlemi bir zaman birimi geciktirilerek veri çatışması yok edilmiştir. İşlem-yok buyruğu 4 cü adımda A veya 5 ci adımda E kesimi tarafından icra edilir. Olay veri çatışmasının derleyici tarafından çözülmesidir. Olay donanım tarafından çözülmediği için daha basit bir kesim yapısı gerektirir.

Saat vuruşu	1	2	3	4	5	6
1- R1'e Yükle	I	A	E			
2- R2'ye yükle		I	A	E		
3- R1+R2'yi topla			I	A	E	
4- R3'e depola				I	A	E

(a) Veri çatışması ile boru hattı zamanı

Saat vuruşu	1	2	3	4	5	6	7
1- R1'e Yükle	I	A	E				
2- R2'ye Yükle		I	A	E			
3- İşlem Yok			I	A	E		
4- R1+R2'yi topla				I	A	E	
5- R3'e depola					I	A	E

(b) Gecikmiş yükleme ile boru hattı zamanı

Şekil 9.9 Üç kesimli boru hattı zamanı

Gecikmiş Dallanma

Şekil 9.8 de bir dallanma buyruğunun boru hattı işlemini, dallanma adresi belli olana kadar geciktirmesi gösterilmişti. Daha önceki kısımda da dallanma hatalarını azaltacak bazı yöntemler tartışılmıştı. RISC işlemcilerin çoğunda kullanılan yöntem derleyiciye dayanır. Derleyici dallanmaları yeniden düzenleyerek zamanında boru hattı içinde olmalarını sağlar. Buna *gecikmiş dallanma* denir.

Gecikmiş dallanmayı kullanan bir derleyici, dallanmadan önceki ve sonraki buyrukları analiz eder. Geciktireceği buyrukların önüne faydalı buyruklar yazarak program akışını yeniden düzenler. Örneğin derleyici, dallanma buyruğundan önceki bazı buyrukları programın akışını bozmamak ve dallanmaya etki etmemek koşuluyla gecikme adımlarına kaydırır. Bu buyruklar bellekten alınır ve boru hattı içinde dallanma buyruğu icra edilirken diğer kesimlerde icra edilirler. Bu durum buyrukların orijinal sıralarında icra edilmelerinin aynıdır fakat dallanma gecikmesi yok edilmiştir. Faydalı ve program akışını bozmayacak buyrukların bulunması derleyiciye bağladır. Derleyici faydalı buyruk bulamaz ise işlem-yok buyruğu yazar.

Gecikmiş dallanma şekil 9.10 da gösterilmiştir. Bu örnek için programda 5 buyruk vardır.

1. Bellekten *R*1 e yükle
2. *R*2 yi arttır
3. *R*3 e *R*4 e topla
4. *R*5 i *R*6 dan çıkar
5. *X* adresine dallan

Şekil 9.10(a) da derleyici dallanmadan sonra iki tane işlem-yok buyruğu yerleştirmiştir. Dallanma adresi zaman evresinde 7 inci adımda *PC* a aktarılmıştır. *X* teki buyruğun alınıp getirilmesi 2 zaman birimi işlem-yok buyrukları ile geciktirilmiştir. *X* buyruğu 8 ci zaman adımında *PC* yenilendikten sonra al-getir evresine başlar.

Şekil 9.10(b) deki program dallanma buyruğundan önce yer alan topla ve çıkar buyruklarının dallanmadan sonra yazılması ile yeniden düzenlenmiştir. Boru hattı zamanlamasının incelenmesi *PC* a *X* değerinin 5 inci zamanda yazıldığını gösteriyor. Fakat topla ve çıkar buyrukları bellekten alınarak normal sırada işleme girmişlerdir. Diğer bir deyişle yükle buyruğu 101 adresindedir ve *X*, 350 ye eşittir. Dallanma buyruğu 103 adresinden alınmıştır. Topla buyruğu 104 adresinden alınmış ve 6 ncı evrede icra edilmiştir. Çıkar buyruğu 105 adresinden alınmış ve 7 inci evrede icra edilmiştir. *X* in değeri *PC* a 5 inci evrede *E* tarafından

Saat	1	2	3	4	5	6	7	8	9	10
1- Yükle	I	A	E							
2- Bir arttır		I	A	E						
3- Topla			I	A	E					
4- Çıkar				I	A	E				
5- X'e dallan					I	A	E			
6- İşlem yok						I	A	E		
7- İşlem yok							I	A	E	
8- X'teki buyruk								I	A	E

(a) İşlem yok buyruğu kullanarak

Saat	1	2	3	4	5	6	7	8
1- Yükle	I	A	E					
2- Bir arttır		I	A	E				
3- X'e dallan			I	A	E			
4- Topla				I	A	E		
5- Çıkar					I	A	E	
6- X'teki buyruk						I	A	E

(b) Buyrukları tekrar sıralayarak

Şekil 9.10 Gecikmiş dallanma örneği

aktarıldığından bu buyruk 6 ncı evrede 350 adresinden alınmıştır. Bu da dallanma adresidir.

9.6 Vektör İşlemleri

Bazı problemler sıradan bilgisayarların kapasitesinin dışındadır. Bunlar muazzam sayıda işleme gerek duyarlar ve sıradan bilgisayarların bu işlemleri yapması günler hatta haftalar sürebilir. Birçok bilimsel ve mühendislik uygulamalarında problemler vektör ve matris işlemleri halinde hesaplanır. Bunlara vektör işlemleriyle çözüm getirmek gerekir.

Uygulamalar

Vektör işlemleri yapan bilgisayarlar özel uygulamalar için istenirler. Aşağıdaki örneklerde vektör işlemleri olması en önemli zorunluluktur.

Geniş zamanlı hava tahminleri
Petrol arama
Sismik veri analizi
Tıbbi teşhis
Aerodinamik ve uzay uçuş benzetimleri
Yapay zeka ve uzman sistemler
İnsan genetiği
Görüntü işleme

Çok ileri bilgisayarlar olmadan yukarıdaki hesaplamalar mantıklı bir zaman aralığında yapılamazlar. İstenen seviyede verime ulaşmak için iyi donanımlı makinelerin vektör ve paralel işlem özellikleri olması gerekir.

Vektör İşlemleri

Birçok bilimsel problemler büyük sayı dizileri üzerine aritmetik işlemler gerektirir. Bu sayılar çoğunlukla kayan noktalı sayıların vektörleri ve matrisleri şeklinde toplanmışlardır. Bir vektör bir boyutlu bir veri dizisidir. *n* boyutlu bir *V* satır vektörü $V = [V_1\ V_2\ V_3\ ...\ V_n]$ şeklinde gösterilir. Sütun vektörünün verileri de tek bir sütunda bulunurlar. Satır vektörü, sütun vektörün her bir bileşenini bir defada işleyebilir. Bundan dolayı vektör elemanları indisli elemanlar olarak ele alınır. Elemanları V_i olan *V* vektörü *V*(*I*) şeklinde yazılır ve *I* indisi verilerin saklandığı bir bellek yada yazaç adresini gösterir. Bir vektör olmayan işlemci ile vektör işlemcinin farkını anlamak için şu döngüye bakalım.

```
      DO 20 I = 1, 100
20    C(I) = B(I) + A(I)
```

Bu 100 elemanlı iki A ve B vektörünü toplayan bir programdır. Bu işlem makine dili ile şu işlem sırası ile yapılır.

```
      Initialize I = 0
20    Read A(I)
      Read B(I)
      Store C(I) = A(I) + B(I)
      Increment I = I + 1
      If I < 100 goto 20
      Continue
```

Bu program A ve B dizilerinden bir çift sayıyı okuyarak kayan noktalı toplayıcıda icra eder. Döngü denetim değişkeni 100 kez tekrarlanan adımlarda güncellenmektedir.

İşlem kodu	kaynak 1 taban adresi	kaynak 2 taban adresi	varış birimi taban adresi	Vektör uzunluğu

Şekil 9.11 Vektör işlemcisi için buyruk biçimi

Vektör işlemi yapabilen bir bilgisayarda bu program tek bir vektör buyruğu ile

$$C(1:100) = A(1:100) + B(1:100)$$

şeklinde yazılır. Buyruk verilerin başlangıç adreslerini içerir. Vektör büyüklüğü de buyruk içinde bellidir. Bütün işlem birleşik bir buyruk ile yapılır. Toplama işlemi boru hattı içinde yer alan ve şekil 9.6 daki kayan noktalı toplayıcıya benzer bir toplayıcıyla gerçekleştirilir.

Muhtemel bir buyruk biçimi şekil 9.11 de gösterilmiştir. Bu aslen 3 adresli bir buyruktur. 3 adres taban adreslerini göstermektedir. Ek bir alan vektörün uzunluğunu verir. Burada vektör verilerinin bellekte olduğu varsayılmıştır. Fakat bunları yazaçlar içindede düşünebiliriz. Bu taktirde adresler ve vektör uzunluğu yerine yazaç adresleri bulunur.

Matris Çarpımı

Matris çarpımı vektör işlemcili bilgisayarların yoğun yaptıkları işlemlerden birisidir. İki $n \times n$ matris çarpımının sonucu n^2 veya toplama-çarpma işlemlerinin sonucu n^3 tür. Bir $n \times m$ matris n satır ve m sütundan oluşmaktadır. Örnek olarak 3×3 lük A ve B matrislerinin çarpımı;

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

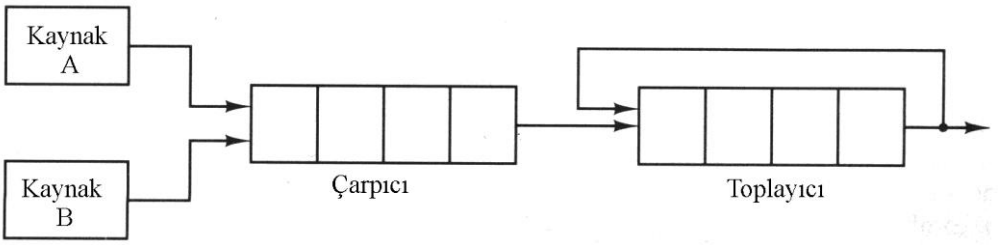
C sonuç matrisi 3×3 tür ve A ve B matrislerinin elemanlarından elde edilmiştir. Yani

$$c_{ij} = \sum_{k=1}^{3} a_{ik} \times b_{kj}$$

dir. Örnek olarak C nin birinci satır ve birinci sütundaki değeri elde edilirse

$$c_{11} = a_{11}\, b_{11} + a_{12}\, b_{21} + a_{13}\, b_{31}$$

Bu sonuç 3 çarpım ve 3 toplamdan elde edilir. 3x3 matrislerin çarpımı için gerekli toplam ve çarpım sayısı $3 \times 9 = 27$ tanedir. $n \times n$ iki matris



Şekil 9.12 İç çarpım hesabı için boru hattı

için gerekli çarpım ve toplam sayısı n^3 tür. Her seferinde işlem yapılmadan önce sonuç sıfırlanır. İç çarpım

$$C = A_1B_1 + A_2B_2 + A_3B_3 + A_4B_4 + \ldots + A_kB_k$$

dır. Bir boru hattı vektör işlemcisinde iç çarpım şekil 9.12 de gösterilmiştir. A ve B değerleri bellekte veya yazaçlardadır. Kayan noktalı sayıların çarpımı boru hattı ve kayan noktalı sayıların toplamı boru hattı dörder kesim içerirler. Çarpan ve toplayan kesimlerin bütün yazaçları sıfırlanır. Dolayısıyla ilk 8 işlemde yani boru hattı dolana kadar çıkışlar sıfırdır. A ve B çiftleri çarpım boru hattına alınır ve her saat vuruşunda bir çift çarpılır. İlk dört zamandan sonra çarpımlar toplayıcının çıktısına eklenirler. Sonraki 4 zamanda toplama boru hattına giren çarpımlara sıfır eklenir. 8 zaman birimi sonra 4 tane $A_1\ B_1$ den $A_4\ B_4$ e kadar çarpımlar toplam kesimlerindedirler. $A_5\ B_5$ ten $A_8\ B_8$ e kadar çarpımlar ise çarpım kesimlerindedirler. 9 uncu evrenin başında toplayıcının çıktısı $A_1\ B_1$ ve çarpıcının çıktısı $A_5\ B_5$ tir. Böylece 9 uncu evre toplama işlemini başlatır ve $A_1\ B_1 + A_5\ B_5$ toplayıcıda toplanır. 10 uncu evre $A_2\ B_2 + A_6\ B_6$ yı bulur v.b. Bu olay 4 kısımda şöyle son bulur:

$$\begin{aligned} C = {} & A_1B_1 + A_5B_5 + A_9B_9 + A_{13}B_{13} + \ldots \\ & + A_2B_2 + A_6B_6 + A_{10}B_{10} + A_{14}B_{14} + \ldots \\ & + A_3B_3 + A_7B_7 + A_{11}B_{11} + A_{15}B_{15} + \ldots \\ & + A_4B_4 + A_8B_8 + A_{12}B_{12} + A_{16}B_{16} + \ldots \end{aligned}$$

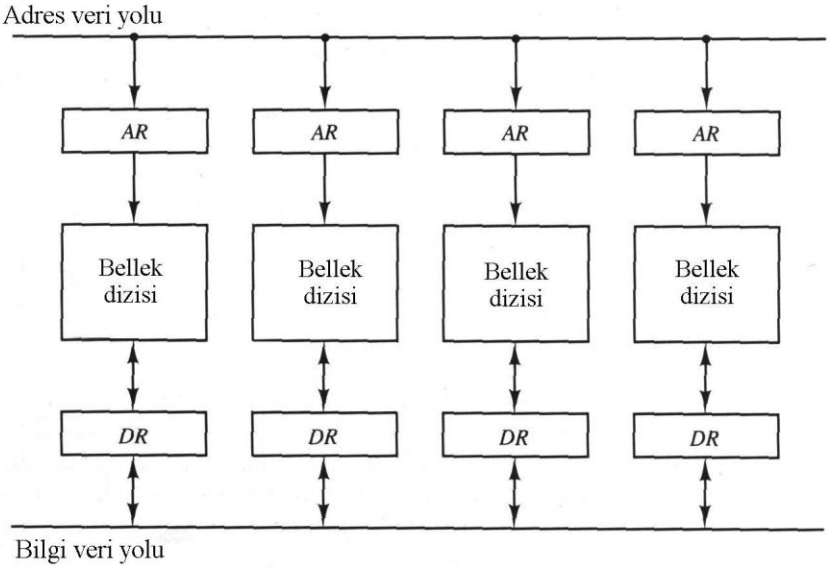
Çarpılacak eleman kalmayınca çarpıcıya 4 tane sıfır girilir. Toplayıcının her bir kesiminde kısmi toplamlar vardır. Bunlar elde edildikten sonra dördü toplanarak sonuç bulunur.

Bellek Ayrıştırma

Boru hattı ve vektör işlemcileri iki veya daha fazla kaynaktan belleğe aynı anda giriş yapmak isterler. Bir buyruk boru hattı bir buyruğu ve bir veriyi aynı anda iki kesimden almak ister. Benzer biçimde bir aritmetik boru hattı iki veya daha fazla verinin aynı anda boru hattına girmesini ister. Aynı anda geçiş için iki bellek veri yolu ı kullanmak yerine bellek belli bir sayıda modüllere ayrılabilir. Bu modüller ortak bellek adresi ve veri yoluna bağlıdırlar. Bir bellek modülü kendi adres ve veri yazaçlarına sahiptir. Şekil 9.123 te 4 modüllü bir bellek dizisi görülüyor. Her modül kendi adres yazacı AR ve veri yazacı DR ye sahiptir. Adres yazaçları ortak adres veri yolundan ve veri yazaçları iki yönlü veri yoluyla haberleşirler. Adresin önemsiz iki biti 4 modülü ayırır.

Modüler sistem diğer modüller bir kelimeyi yazarken veya okurken veya işlemde iken bir modülün bir belleğe erişim başlatmasını ve yapmasını sağlar. Her bir modül diğer modüllerin durumdan bağımsız olarak bellek erişimi sağlayabilir.

Modüler belleğin avantajı ayrıştırma adı verilen bir tekniğin kullanılmasına izin vermesidir. Bu şekilde bölünen bir bellekte farklı adres kümeleri farklı bellek modüllerine atanırlar. Örneğin iki modüllü bellekte çift ve tek adresler ayrı modüllere ait olur. Modüllerin sayısı 2 nin kuvveti olursa, adresin önemsiz bitleri modül seçer. Kalanlar ise modül içindeki adrestir.



Şekil 9.13 Çoklu modül bellek organizasyonu

Modüler bellekler boru hattı ve vektör işleyen sistemlerde faydalıdır. n tane modüllü bir belleği olan bir vektör işlemcisi n farklı modülden n tane veri alabilir. Etkin bellek evresi modül sayısı oranında azaltılabilir. Buyruk boru hattı olan bir MİB çok sayıda bellek modülü avantajını kullanabilir. Çünkü her bir kesim diğer kesimlerden bağımsız olarak belleğe erişilebilir.

Süper Bilgisayarlar

Vektör buyrukları olan ve kayan noktalı aritmetik işlemler yapabilen boru hattı bulunan ticari bilgisayarlara süper bilgisayar denir. Süper bilgisayarlar çok güçlüdürler, yüksek verimleri vardır ve bilimsel hesaplamalarda daha çok kullanılırlar. İşlemlerin hızlandırılması için bileşenler birbirine çok sıkı şekilde bağlanmış ve elektronik sinyallerin kat ettikleri yollar minimuma indirilmeye çalışılmıştır. Bu süper bilgisayarlarda parçaların yakın oluşundan doğan ısınma problemi özel yöntemlerle giderilmiştir.

Süper bilgisayarların buyruk kümesi standart veri aktarımı veri işleme ve program denetim buyruklarından oluşur. Bunlar sıradan bilgisayarlarda da böyledir. Bunlara ilave olarak vektör işlemleri ve vektör-sabit karmaşık işlemleri yapan buyruklarla takviye edilmişlerdir. Bir süper bilgisayar çok büyük hızı ile, hızlı ve büyük belleği ile ve paralel işlemeyi çok yoğun kullanması ile belli olur. Her biri kendi boru hattına sahip olan çeşitli birimleri vardır. Her ne kadar süper bilgisayarlar genel amaçlı işleri yapabilirlerse de özellikle vektör ve matris işlemlerini kayan noktalı olarak yapacak sayısal hesaplamalar için en iyi şekilde dizayn edilmişlerdir.

Süper bilgisayarlar normal bir bilgi işlem merkezinde günlük kullanıma elverişsizdirler. Sadece bir takım bilimsel hesaplamalar için kullanılırlar. Bu hesaplamalar sayısal hava tahmini, sismik dalga analizi, uzay çalışmaları v.s dir. Fiyatlarının çok yüksek olmasından dolayı kullanım ve satışları kısıtlıdır.

Bilgisayarlar için bir ölçü birimi, birim zamanda yapabildiği kayan noktalı işlem sayısı flops tur. Mega flops, milyon flops, giga flops, milyar flops anlamına gelir. Tipik bir süper bilgisayarın bir evresi 4 ile 20 ns civarındadır. Bunun hızı 50 ile 250 milyon flops civarında olur. Bu zaman boru hattının dolma süresini içermez. Cevapların boru hattından çıkma anından başlayan süredir.

İlk süper bilgisayar da 1976 da yapılan Cray-I dir. Vektör işlemleri ile beraber paralel çalışan 12 işlem birimi mevcuttur. Her işlem biriminin kendi boru hattı mevcut olup hepsi kesimlere ayrılmıştır. Bütün birimler aynı anda çalışabilirler. CPU içinde 150 den fazla yazaç vardır. Buradaki

veriler bütün birimlerin aynı anda çalışmalarına imkan verir. Bir kayan noktalı işlemi iki tane 64 bit veri üzerinde bir saat vuruşu içinde 12,5 ns de tamamlanır. Bu süper bilgisayara 80 mega flops boru hattı hızı kazandırır. Belleği 4 milyon 64 bit kelime den oluşmuştur. Bellek 16 bankaya bölünmüş olup erişim süresi 50ns dir. Bunun anlamı 16 bankaya aynı zamanda erişildiğinde bellek aktarım hızı saniyede 320 milyon kelime olur. CRAY şirketi çok işlemcili Cray X-MP ve Cray Y-MP modellerini geliştirmiştir. Yeni Cray-2 vektör işlemle Cray-1 den 12 kat hızlıdır.

Bir diğer model Fujitsu VP-200 idi. Bir sabit ve bir vektör işlemcisi olup aynı anda çalışabiliyorlardı. Aynı Cray ler gibi çok sayıda yazaçları, işlem birimleri mevcuttu ve yazaçtan yazaca vektör işlemi yapabiliyordu. Vektör işlemcisi içinde 4 boru hattı mevcut olup hepsi çalıştığında 500 milyon flops a erişebiliyordu. Bellek 32 milyon kelime olup, vektör yazaçlarına yükleme ve depolama boru hatları ile bağlıydı. VP-200 ün 83 vektör buyruğu ve 195 sabit buyruğu mevcuttu. Yeni VP 2600 ün evresi 3.2ns ve doruk hızı 5 milyar flops tur.

9.7 Dizi İşlemcileri

Çok büyük veri dizileri üzerine işlem yapan işlemcilerdir. İki farklı tip için aynı ad kullanılır. Eklenmiş dizi işlemcisi; genel amaçlı bir işlemciye eklenmiş bir işlemcidir. Özel nümerik hesaplamalar için esas makinenin verimini arttırmaya yarar. SIMD dizi işlemcisi ise tek buyruk çok veri tipi bir işlemcidir. Vektör işlemlerini tek buyruk ile birçok birimde farklı veriler üzerine aynı işlemi yaparak yerine getirir. Her ne kadar iki dizi işlemcide vektör işlemleri yapıyorlarsa da iç yapılar farklıdır.

Eklenmiş Dizi İşlemci

Bir bilgisayar için çevre birimi gibi yapılmıştır, bilgisayarın verimini arttırır. Bunun için karışık bilimsel problemlerde vektör işlemleri yapar. Yüksek verimin nedeni çok sayıda birimle paralel işlem yapmasıdır. İçinde bir veya daha fazla kayan noktalı toplayıcı ve çarpıcı bulunan aritmetik birimi vardır. Bu işlemci kullanıcılar tarafından karışık aritmetik problemler için programlanabilir.

Şekil 9.14 te eklenmiş dizi işlemcinin bir bilgisayar ile birleşimi görülüyor. Bilgisayar genel amaçlı ticari bir bilgisayardır. Ekli işlemci ise ana makine tarafından kullanılan bir ilave aygıttır. Dizi işlemci bir G/Ç denetim birimi üzerinden ana bilgisayara bağlanmıştır. Ana bilgisayar bu işlemciyi dış arabirim olarak görür. Ekli dizi işlemciye veriler ana bellekten yerel belleğe gönderilerek aktarılır. Aktarma hızlı bir veri yolu ile yapılır.



Şekil 9.14 Ana bilgisayar ile eklenmiş dizi işlemci

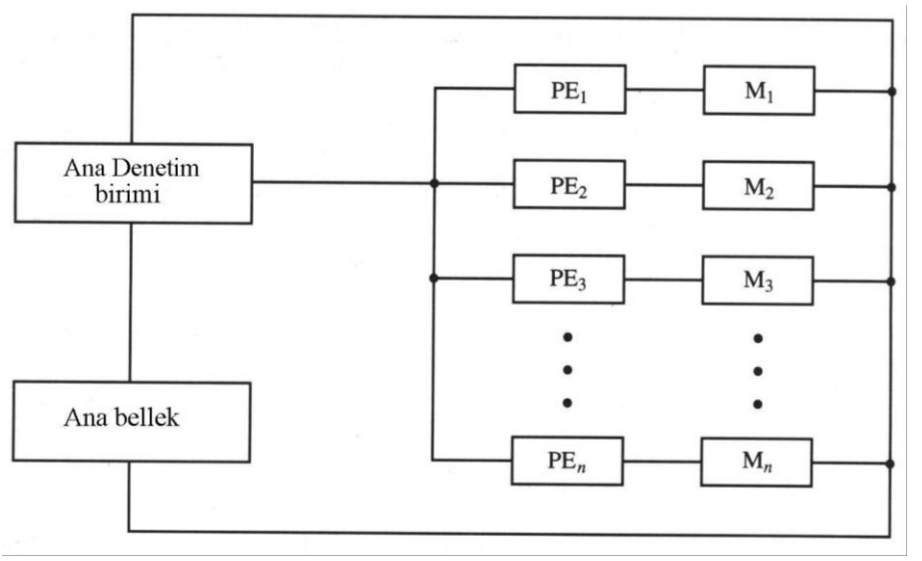
Bazı üreticiler ekli dizi işlemcileri birçok farklı modele bağlanacak şekilde yapmışlardır. Örneğin VAX 11 e bağlanan FSP-164/MAX bilgisayarı, 100 milyon flops a götürür. Amaç genel amaçlı bilgisayara, süper bilgisayarların çok daha azı bir harcama ile vektör işlem yeteneği kazandırmaktır.

SIMD Dizi İşlemcisi

SIMD dizi işlemcisi bir çok sayıda paralel çalışan işlemcisi bulunan bir bilgisayardır. İşlemciler ortak bir denetim biriminin emri altında aynı işlemi yerine getirirler. Dolayısıyla tek bir buyruk akımı vardır. Fakat çok sayıda veri akımı mevcuttur. Bilgisayarın genel blok şeması şekil 9.15 te verilmektedir. İçinde birbirinin aynı olan bir işlemciler kümesi(PE) vardır. Her bir işlemcinin ayrı bir yerel belleği mevcuttur. Her işlemci içinde ALU, kayan noktalı aritmetik birimi ve yazaçlar vardır. Denetim birimi işlemciler içindeki işlemleri kontrol eder. Esas bellek programların yüklenmesini sağlar. Denetim birimi buyrukların kodunu çözer ve buyrukların nasıl icra edileceğine karar verir. Sabit ve program denetim buyrukları denetim birimi içinde doğrudan icra edilir. Vektör işlemleri işlem birimlerine dağıtılır. Her işlemcinin verileri kendi belleklerinde saklıdır. Bu veriler belleklere paralel işlem yapılması için uygun biçimde dağıtılır.

$C = A + B$ vektör toplama işlemine bakılırsa, denetim birimi önce a_i ve b_i bileşenlerini M_i lere $i = 1, 2, 3, \ldots n$ olacak şekilde dağıtır. Sonra kayan noktalı sayılar için topla buyruğunu bütün PE lere yollar ve bütün birimlerde aynı anda $c_i = a_i + b_i$ işlemi yapılır. c_i bileşenleri her bir M_i içinde belli yerlere yazılır. Dolayısıyla bir topla evresi içinde vektör işlemi tamamlanır.

Her bir işlemcinin PE durumunu denetim için maskeleme yolları kullanılabilir. Her işlemcinin bir bayrağı vardır ve işlemci çalışıyorsa 1 ve çalışmıyorsa 0 durumundadır. Örneğin bir SIMD dizi işlemcisinin 64



Şekil 9.15 SIMD dizi işlemci tasarımı

işlemcisi (PE) olsun. Eğer 64 bileşenli bir vektör varsa sorun yok. Bileşen sayısı 64 ten az ise denetim birimi gerekli sayıda işlemciyi çalıştırmaz. 64 ten fazla bileşeni olan vektörler için altmışdörtlük parçalar halinde işlem yapılır.

En iyi bilinen SIMD dizi işlemci ILLIAC IV olup, Illinois Üniversitesinde geliştirilmiştir. Burroughs tarafından imal edilmiştir. Ancak şimdi çalışmıyor. SIMD bilgisayarlar hayli özel bilgisayarlardır. Onlar asıl matris yada vektör formunda ifade edilen problemler için uygundurlar. Buna rağmen geleneksel veri işleme programlarındaki diğer hesaplamalar için uygun değildirler.

SORULAR

9-1. Bazı aritmetik işlemlerde $(A_i + B_i)$ $(C_i + D_i)$ işleminin büyük dizilerle yapılması gerekir. Bu görevi yapacak bir boru hattı tasarlayın. Boru hattındaki tüm yazaçların $i = 1$ den 6 ya kadar alacakları değerleri sıralayın.

9-2. Altı parçalı bir boru hattı için bir uzay-zaman diyagramı çizerek 8 görevi yapması için gerçek zamanı gösterin.

9-3. 6 parçalı bir boru hattında 200 görevi yapması için geçecek zaman çevrimlerinin sayısını bulun.

9-4. Bir boru hattı sisteminde bir görevin icrası 50 ns zaman alır. Aynı görev 6 parçalı bir boru hattında 10 ns de yapılabiliyor. 100 görev için boru hattının hızlanma oranını bulun. Elde edilebilecek en büyük oran nedir?

9-5. Şekil 9.2 deki boru hattında şu yayılma zamanları geçerlidir. 40 ns; verilerin bellekten alınıp *R*1 ve *R*2 yazaçlarına aktarılması, 45 ns; sinyalin çarpma işlemcisinde ilerlemesi, 5 ns *R*3 aktarımı ve 15 ns iki sayıyı R5 içinde toplama.

a. Kullanılan zaman ne kadardır.

b. R3 ve R4 ü kaldırarak aynı işlem boru hattı dışında bir bilgisayar tarafından da yerine getirilebilir. Boru hattı olmadan verileri çarpmak ve toplamak ne kadar sürer?

c. Hızlanma 10 görev için ne kadardır ve 100 görev için ne kadardır?

d. Erişilebilecek en büyük hızlanma ne kadar olur?

9-6. 8 bitlik ikili sistemde iki tamsayıyı çarpacak sabit noktalı çarpma yapan bir boru hattı tasarımı gerekiyor. Her parçada birkaç tane VE kapısı ve bir tane ikili toplayıcı olmalıdır. Toplayıcı şekil 10.10 daki dizi çarpıcıya benzemekte.

a. Her bir kesimde kaç tane VE kapısı vardır? Ve ne büyüklükte bir toplayıcı gerekir?

b. Boru hattında kaç kesim var?

c. Her kesimde yayılma gecikmesi 30 ns ise, boru hattı içinde iki sabit noktalı sayıyı çarpmak ortalama ne kadar sürer?

9-7. Şekil 9.6 daki boru hattında 4 kesimdeki gecikmeler $t_1 = 50$ ns, $t_2 = 30$ ns, $t_3 = 95$ ns ve $t_4 = 45$ ns. Ara yüz yazaçlarındaki gecikme $t_r = 5$ ns.

a. Boru hattında 100 çift sayıyı toplamak ne kadar zaman alır.

b. A şıkkında bulunan zamanı nasıl yarıya indirebiliriz?

9-8. Şekil 9.6 daki kayan noktalı boru hattı toplayıcıyı, 100 tane kayan noktalı sayıyı toplamak ($X_1 + X_2 + X_3 + ... + X_{100}$) için nasıl kullanırız?

9-9. Bir bilgisayar için 6 kısımlı buyruk boru hattı kullanılıyor. Her kısımda icra edilecek işlemleri belirleyin?

9-10. Buyruklardaki dallanmalardan dolayı buyruk boru hattında verimdeki azalmayı en aza indirecek 4 olası donanım çizin.

9-11. Aşağıdaki programda dört buyruk verilmiştir. Varsayalım ki ilk buyruk şekil 9.8 de kullanılan boru hattının ilk adımından başlasın. Dördüncü adımda hangi işlemlerin yapılacağını söyleyiniz.

```
LOAD     R1 ← M[312]
ADD      R2 ← R2 + M[313]
INC       R3 ←R3 + 1
STORE    M[314] ← R3
```

9-12. Bölüm 9.5 teki üç kısımlı boru hattında veri karmaşasına yol açacak bir program örneği verin.

9-13. Bölüm 9.5 teki 3 kısımlı boru hattında gecikmiş yüklemenin kullanımına bir örnek verin.

9-14. Bölüm 9.5 teki 3 kısımlı boru hattında dallanma cezasına neden olacak bir program örneği verin.

9-15. Bölüm 9.5 teki 3 kısımlı boru hattında gecikmiş dallanmayı kullanan bir program örneği verin.

9-16. İki 40 × 40 matrisin çarpımını düşünün. İşlem vektör işlemcisiyle yapılacaktır.

a. Her bir iç çarpımda kaç tane çarpım terimi vardır? Kaç tane iç çarpım elde edilmelidir?

b. Çarpım matrisini hesaplayabilmek için kaç tane çarpma-toplama işlemi gerekir?

9-17. Şekil 9.12 deki boru hattına bakalım. Bu boru hattı ile iki tane 60 × 60 matrisi çarpacağız. Bu işi yaparken iç çarpım işlemi kaç saat çevrimi alır? Kaç tane iç çarpım var? Çarpım matrisinin bulunması kaç saat çevrimi alır?

9-18. Şekil 9.13 te tanımlanan belleğe 1024 kelimelik bir veri dizisi yerleştirilecektir. Adresleri atayın.

9-19. Hava tahmini, yaklaşık 250 milyar kayan noktalı sayılar üzerine işlem gerektirir. Bu iş bir süper bilgisayarda yapılmaktadır. Bu süper bilgisayar 100 milyon flop işlem yapabilmektedir. Bu işlemleri yapması ne kadar sürer?

9-20. İçinde 4 tane kayan noktalı boru hattı işlemcisi bulunan bir bilgisayar düşünün. Her bir işlemcinin 40 çevrim süreci var. 400 kayan noktalı işlem yapması ne kadar zaman alır? Eğer aynı sayıda işlem tek bir boru hattı olan ve çevrim süreci 10 ns bir bilgisayarda yapılsaydı fark olur mu idi? Ne kadar olurdu?

KAYNAKLAR

1. Dasgupta, S., *Computer Architecture: A Modern Synthesis*, Vol. 2. New York: John Wiley, 1989.
2. DeCegama, A. L., *Parallel Processing Architecture and VLSI Hardware*. Englewood Cliffs, NJ: Prentice Hall, 1989.
3. Cliffs, NJ: Prentice Hall, 1991.Gibson, G. A., *Computer Systems Concepts and Desing*. Englewood
4. Hays, J. F., *Computer Architecture and Organization,* 2nd ed. New York: McGraw-Hill, 1988.
5. Hwang, K., and F. A. Briggs, *Computer Architecture and Parallel Processing*. New York: McGraw-Hill, 1984.

6. Kain, R., *Computer Architecture: Software and Hardware*. Vol. 2. Englewood Cliffs, NJ: Prentice Hall, 1989.
7. Lee, J. K. F., and A. J. Smith, "Branch Prediction Strategies and Branch Target Buffer Desing." *Computer,* Vol.17, No. 1 (January 1984), pp. 6-22.
8. Lilja, D. J., "Reducing The Branch Penalties in Pipeline Processors." *Computer*, Vol.21, No.7 (July 1988), pp. 47-55.
9. Patterson, D. A., and J. L. Hennessy, *Computer Architecture: A Quantitative Approach*. San Mateo, CA: Morgan Kaufmann Publishers, 1990.
10. Pollard, L. H., *Computer Desing and Architecture,* Englewood Cliffs, NJ: Prentice Hall, 1990.
11. Stone, H. S., *High-Performance Computer Architecture,* 2nd ed. Reading, MA: Addison-Wesley, 1990.
12. Tabak, D., *Multiprocessors*. Englewood Cliffs, NJ: Prentice Hall, 1990.

BÖLÜM 10

Bilgisayar Aritmetiği

BU BÖLÜMDE

10-1 Giriş
10-2 Toplama ve Çıkarma
10-3 Çarpma Algoritmaları
10-4 Bölme Algoritmaları
10-5 Kayan Noktalı Aritmetik İşlemler
10-6 Ondalık Aritmetik Birim
10-7 Ondalık Aritmetik İşlemler

10.1 Giriş

Sayısal bilgisayarlarda hesaplama problemlerinin sonuçlandırılması için verileri işleyen aritmetik buyruklar vardır. Bu buyruklar aritmetik hesaplamaları yaparlar ve bilgisayar içindeki tüm veri işlemelerinden sorumludurlar. 4 temel işlem toplama, çıkarma, çarpma ve bölmedir. Bu dört temel işlemden aritmetik fonksiyonları elde etmek ve sayısal hesaplamalarla bilimsel problemleri çözmek mümkündür.

Aritmetik işlemci, işlem biriminin bir parçası olup aritmetik işlemleri icra eder. Aritmetik işlemlerin icrası sırasında işlemci yazaçlarında bulunan veri tipleri buyruğun içinde belirtilmiştir. Bir aritmetik buyruk ikili veya ondalık veriyi belirler. Her iki durumda da veri sabit noktalı veya kayan noktalı olabilir. Sabit noktalı sayılar tamsayıları veya kesirli sayıları gösterebilirler. Negatif sayılar işaretli veya tümleyen gösterimiyle verilebilir. Eğer ikili sabit noktalı sayılar için toplama buyruğu icra edilecekse aritmetik işlemci çok basittir. Eğer 4 aritmetik işlemin hepsi hem ikili hem ondalık veri için sabit noktalı ve kayan noktalı olarak yapılacaksa aritmetik işlemci daha karışık olacaktır.

Algoritma

Önceki bölümlerde temel aritmetik işlemlerin işaretli sayılarla nasıl yapılacağı verilmişti. Eğer işlemler donanım içinde mevcut ise, yani donanım tarafından yapılıyorsa önceden anlatılanlar doğrudur. Ancak tasarımcı devreyi çizerken bütün adımları çok iyi bilmelidir. Herhangi bir problemin çözümü eğer bir dizi sonlu sayıda ve iyi tanımlanmış işlemlerle gerçekleştirilebiliyorsa bu adımların hepsine algoritma denir. İki sabit noktalı sayı için toplama algoritması bölüm 3.3 te anlatılmıştı. O algoritmada negatif sayıların 2 ye göre tümleyen olarak verildiği kabul edilmişti. Çalışması için, ikili paralel toplayıcı devreye ihtiyacı olan, basit bir algoritma idi. Eğer sayılar 2 ye göre tümleyen olarak değil de işaretli olarak verilirse algoritma biraz daha karışık hale gelir. Devrenin yapımı ise toplama ve çıkarma, işaretleri karşılaştırma ve sayıları karşılaştırma gibi devreleri gerektirir. Genellikle bir algoritma bir dizi işlem adımları gerektirir. Her adım bir önceki adımın sonucuna bağlıdır. Algoritmalar akış şemalarıyla ifade edilir. Hesaplamalar dikdörtgenlerle, kararlar baklava ile gösterilir, bunlardan iki veya daha fazla yol çıkar.

Bu bölümde birçok aritmetik algoritma geliştirecek ve bunların sayısal donanımda elde edilişi gösterilecektir. 4 işlem aşağıdaki veriler için düşünülecek.

1. Sabit noktalı ikili veri işaretli gösteriminde
2. Sabit noktalı ikili veri 2 ye göre tümleyen gösteriminde
3. Kayan noktalı ikili veri
4. İkili kodlanmış ondalık veri

10.2 Toplama ve Çıkarma

Bölüm 3.3 te gösterildiği gibi negatif sabit nokta sayıları göstermek için üç yol vardır. İşaretli, 1 in tümleyeni, 2 ye göre tümleyen. Çoğu bilgisayarlar 2 ye göre tümleyen gösterimini kullanırlar. Ancak kayan noktalı işlemler için çoğu bilgisayarlar işaretli gösterimini kesir kısmı için kullanırlar. Burada işaretli sayılar ve 2 ye göre tümleyene göre sayılarda toplama ve çıkarma algoritmaları elde edilecektir. Önemli bir nokta şudur ki kabul edilen gösterim aritmetik işlemden önce ve sonra yazaçta bulunacak gösterimdir. Dolayısıyla ancak ara hesaplarda, örneğin çıkarma işlemi yapılırken sayıların 2 ye göre tümleyeni şeklinde kullanılıp sonucun aynı biçimde alınması uygundur. Sonuç işaretli sayıya dönüştürülür.

İşaretli Sayılarda Toplama ve Çıkarma

İşaretli sayılar gösterimi günlük hesaplarda kullanılan sayılardır. Bunlarla kağıt üzerinde toplama ve çıkarma çok basittir. Bu işlemlerin tekrar incelenmesi donanım algoritmalarını çıkarmaya yardımcı olacaktır.

Çizelge 10.1 İşaretli sayıların toplanması ve çıkarılması

İşlem	İşaretli toplama	İşaretli çıkarma		
		A > B ise	A < B	A = B
(+A) + (+B)	+(A + B)			
(+A) + (–B)		+(A – B)	–(B – A)	+(A – B)
(–A) + (+B)		–(A – B)	+(B – A)	+(A – B)
(–A) + (–B)	–(A + B)			
(+A) – (+B)		+(A – B)	–(B – A)	+(A – B)
(+A) – (–B)	+(A + B)			
(–A) – (+B)	–(A + B)			
(–A) – (–B)		–(A – B)	+(B – A)	+(A – B)

Büyüklük

İki sayının büyüklüğü *A* ve *B* ile gösterilsin. İşaretli sayılar toplandığında veya çıkarıldığında 8 farklı durum ortaya çıkar. Bunlar sayıların işaretlerine ve yapılan işleme bağlıdır. Bu durumlar çizelge 10.1 in ilk sütununda verilmiştir. Diğer sütunlarda sayıların gerçek değerleri ve işaretleri üzerine yapılan gerçek işlemleri gösterir. Son sütün negatif sıfırı engellemek içindir. Yani iki eşit sayı çıkarıldığında kalan +0 dır –0 değildir. Toplama ve çıkarma algoritması bu çizelgeden türetilebilir. Parantezler çıkarma algoritmasında kullanılacak.

Toplama ve Çıkarma Algoritması; A ve B nin işaretleri aynı olduğunda farklı iki sayı işaretsiz olarak toplanır ve A nın işareti sonuca işaret olarak verilir. Eğer A ve B nin işaretleri farklı ise işaretsiz değerlerini veya mutlak değerlerini karşılaştır ve küçük olanı büyük olandan çıkar. Eğer A büyükse A nın işaretini, tersi ise tümleyenini işaret olarak ver. Eğer değerler eşit ise B yi A dan çıkar ve sonucu pozitif yap.

İki algoritma işaret karşılaştırması hariç aynıdır. Aynı işaretli iki sayının toplamı için uygulanan algoritma, farklı işaretli sayıların çıkarması için olanın aynıdır ve tersi de doğrudur.

Donanımın Kurulması

İki aritmetik işlemin donanım yoluyla kurulması için iki sayının önce yazaçlarda saklanması gerekir. *A* ve *B* sayıların işaretsiz (mutlak) değerlerini tutan iki yazaç ve A_S ve B_S de bunların işaretlerini tutan yaz-bozlar olsun. İşlemin sonucu üçüncü bir yazaca gönderilir. Eğer sonuç *A* ve A_s de tutulursa donanımdan kar edilir. Dolayısıyla *A* ve A_s beraber işlemci yazacını oluşturmuş olur.

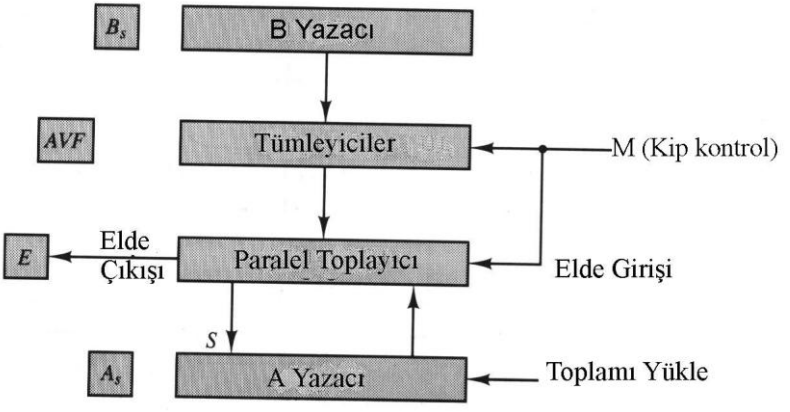
Yukarıdaki algoritmanın donanımının kurulması için, önce $A + B$ mikro işlemini yapacak bir paralel toplayıcıya gerek var. Sonrada karşılaştırma işlemi yani $A > B$, $A = B$, veya $A < B$ olduğunu bulmak için karşılaştırıcıya gerek var. Sonra $A - B$ ve $B - A$ mikro işlemlerini yapmak

için iki paralel çıkarma devresine, ihtiyaç vardır. İşaretin durumu A_S ve B_S girişleri olan bir ÖZEL-VEYA kapısı ile bulunabilir.

Yukarıdaki işlemler için bir karşılaştırıcı, bir toplayıcı ve iki çıkarıcıya gerek var. Fakat işlemin değişik biçimde yapılmasıyla daha az parça gerekebilir. Çıkarma işlemi, tümleme ve toplama işlemleriyle gerçekleşebiliyordu. Karşılaştırma işlemi ise çıkarma işleminin sonunda son elde den elde edilebilir. Bu olasılıkların karşılaştırılması tümleme yoluyla çıkarma işleminin ve çıkarma yoluyla karşılaştırmanın daha etkili olduğunu göstermektedir. Bu yöntemde bir toplayıcı ve bir tümleyiciye gerek vardır.

Şekil 10.1 toplama ve çıkarma işlemleri için kurulacak donanımın blok şemasını göstermektedir.

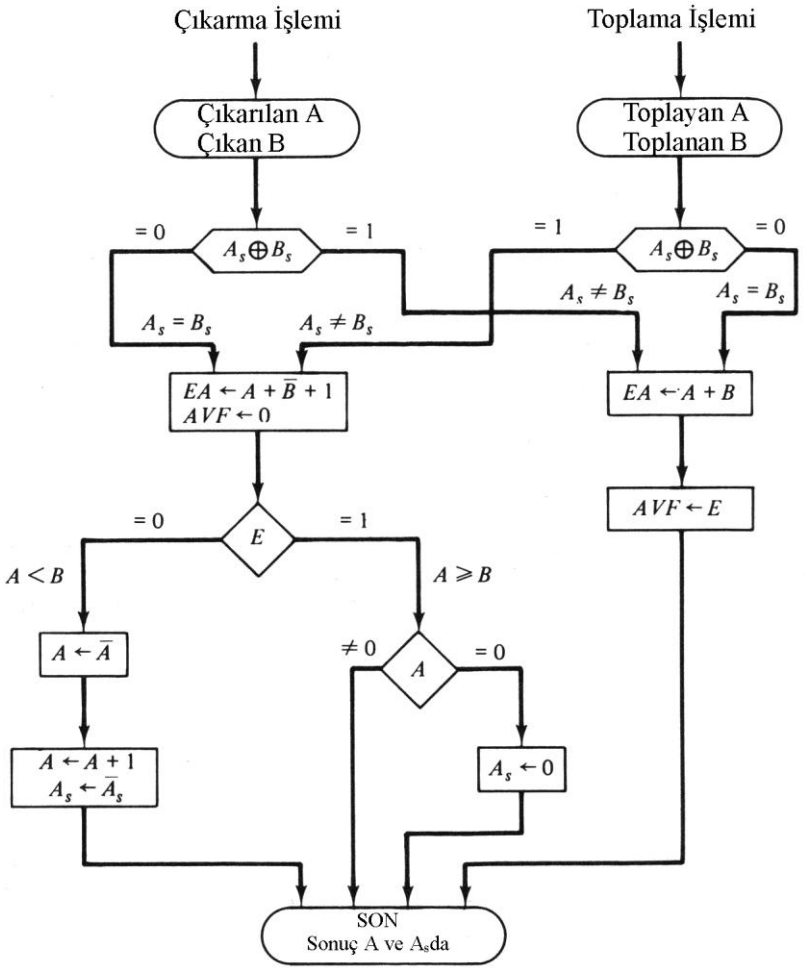
Donanımda (şekilde görüldüğü gibi) *A* ve *B* yazaçları ile A_S ve B_S yazbozları bulunuyor. Çıkarma işlemi, *A* ile *B* nin 2 ye göre tümleyeninin toplanmasıyla gerçekleşmektedir. Çıkış eldesi *E* yaz-bozuna aktarılmaktadır. *E* yaz-bozundan da iki sayının göreceli durumu anlaşılır. Toplamada taşma yaz-bozu *AVF*, *A* ve *B* toplandığında taşma bitini tutar. *A* yazacı algoritmada yer alan diğer adımlardaki işlemleri yapar. $A + B$ işlemi paralel toplayıcıda yapılır. Toplayıcının *S* (toplam) çıkışı *A* yazacına yollanır. Tümleyici, kip denetimi *M* in durumuna göre *B* nin çıkışının tümlerini alır ve *B* yi doğrudan çıkışa yollar. Tümleyici içinde ÖZEL-VEYA kapıları ve paralel toplayıcı içinde tam toplayıcılar vardır. *M* sinyali aynı zamanda toplayıcının giriş elde sine de yollanır. $M = 0$ iken *B* nin çıkışı toplayıcıya yollanır. Giriş elde si sıfırdır, toplayıcının çıktısı $A + B$ olur. $M = 1$ iken, *B* nin 2 ye göre tümleyen toplayıcıya gönderilir. Giriş elde si 1 dir. Çıktısı $A + B + 1$ dir. Buda $A - B$ dir.



Şekil 10.1 İşaretli sayıların toplam ve çıkarılma donanımı

Donanım Algoritması

Dananım algoritmasının akış şeması şekil 10.2 de verilmiştir. İki işaret A_S ve B_S bir ÖZEL-VEYA kapısı ile karşılaştırılmıştır. Eğer kapının çıktısı 0 ise işaretler aynıdır, eğer çıktı 1 ise işaretler farklıdır. Bir toplama işlemi için aynı işaret mutlak değerlerin toplanacağını gösterir. Mutlak değerler $EA \leftarrow A + B$ mikro işlemiyle toplanır. ($EA = ACC$ + Elde). Eğer toplama işleminden sonra elde de 1 varsa toplamada taşma olmuştur. E nin değeri AVF yaz-boz una aktarılır.



Şekil 10.2 Toplama ve çıkarma işlemleri için akış şeması.

Tümleme ve Arttırma

Eğer toplama işlemi için işaretler farklı ise ve eğer çıkarma işlemi için işaretler aynı ise iki sayının mutlak değeri çıkarılır. Bunun için *B* nin 2 ye göre tümleyeni *A* ya toplanır. Sayılar çıkarıldığından taşma olmaz, *AVF* sıfır yapılır. *E* içinde 1 varsa $A > B$ dir ve *A* sayısı doğru sonuçtur. Eğer bu sayı 0 ise negatif sıfırdan kaçınmak için A_S pozitif yapılmalıdır. *E* içinde 0, $A < B$ yi gösterir. Bu taktirde sonuç değeri *A* nın değerinin 2 ye göre tümleyen alınır. Bu işlem tek bir $A \leftarrow A + 1$ mikro işlemi ile yapılabilirdi. Fakat *A* yazacının sadece tümleme ve arttırma işlemleri yapabildiği kabul edildiğinden bu işlem 2 mikro işlemle yapılabilir. Akış şemasının sağ kısmında sonucun işareti A nın işaretinin aynı olduğundan A_S nin değeri değiştirilmez. Nitekim $A < B$ olduğunda sonucun işareti *A* nın işaretinin tümleridir. Dolayısıyla A_S yi tümlemek gerekir. Sonuç *A* yazacında işareti de A_S de bulunur. *AVF* nin değeri taşmayı gösterir. *E* nin son değeri önemsizdir.

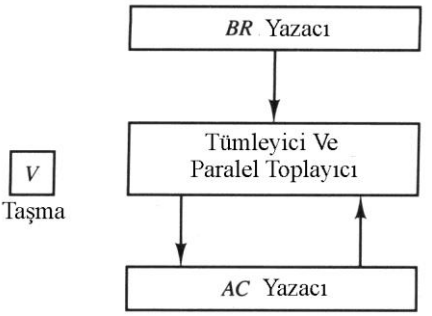
İşaretli 2 ye Göre Tümleyen ile Toplama Çıkarma

İşaretli sayılarda 2 ye göre tümleyen gösterimi ile toplama ve çıkarma için aritmetik algoritmalar bölüm 3.3 te verildi. Burada hatırlanması için kısaca anlatılacak. İkili sayının en sol biti işaret bitidir. 0 pozitif 1 negatif sayıları gösterir. Eğer işaret biti 1 ise bütün sayı 2 ye göre tümleyen biçimindedir. Dolayısı ile +33 sayısı 00100001 ve –33 11011111 ile gösterilir. 11011111 sayısı 00100001 sayısının 2 ye göre tümleyendir.

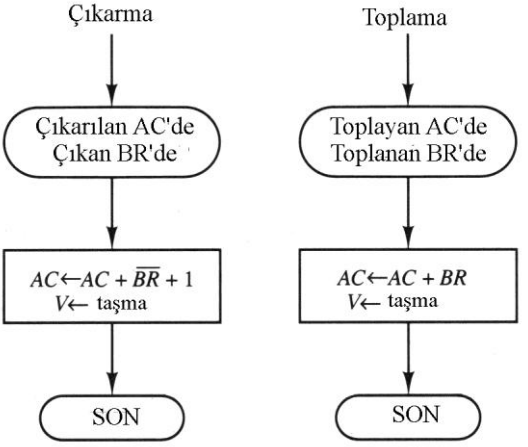
Toplamada 2 ye göre tümleyen biçiminde verilmiş iki sayıda işaret bitleri tıpkı diğer bitler gibi muamele görür. İşaret biti dışındaki elde ihmal edilir. Çıkarma işleminde önce çıkacak sayının 2 ye göre tümleyen alınır sonra iki sayı toplanır.

n basamaklı iki sayı toplandığında toplam $n + 1$ basamak yer kaplar. Bu durumda taşma oldu denir. Taşmanın 2 ye göre tümleyen şeklindeki sayıların toplamı üzerine etkisi 3.3 te incelenmişti. Taşma durumu toplamdaki son iki elde nin ÖZEL-VEYA kapısına uygulanması ile bulunur. Kapının çıkışı 1 ise taşma vardır. Donanımın yazaç yapısı şekil 10.3 te verilmektedir. Bu şekil 10.1 deki konfigürasyonun aynısıdır. Sadece işaret bitleri diğer yazaçlardan ayrılmamıştır. *A* yazacını *AC* olarak ve *B* yazacını *BR* olarak isimlendiriyoruz. *AC* ve *BR* den en sol bitler sayıların işaret bitlerini gösterir. Tümleyici ve paralel toplayıcıda işaret bitleri diğer bitlerle beraber toplanır veya çıkarılır. Taşma yaz-bozu *V* eğer taşma varsa 1 olur. Bu durumda çıkış elde si dikkate alınmaz.

2 ye göre tümleyen biçimindeki 2 ikili sayının toplama ve çıkarmasının algoritması 10.4 teki akış şemasında gösterilmektedir. Toplam *AC* ve *BR* nin içeriklerinin işaret bitleri dahil toplanmasıyla elde edilir. Son iki elde nin ÖZEL-VEYA sı 1 ise *V* 1 yapılır. Aksi halde 0 yazılır. Çıkarma



Şekil 10.3 2 ye göre tümleyenli toplama ve çıkarma için donanım.



Şekil 10.4 Toplama ve çıkarma algoritmaları çıkan sayı 2 ye göre tümleyen şeklinde

işlemi için *AC* ye *BR* nin 2 ye göre tümleyeni eklenir. Yani pozitif sayı negatif hale getirilir veya tersi olur. Negatif sayı pozitif hale getirilir. Taşma durumu işlem sırasında denetlenmelidir. İki sayı aynı işaretli olabilir (Programcı buna dikkat etmelidir). Çünkü eğer bir taşma olursa *AC* yazacındaki sonuç yanlıştır.

10.3 Çarpma Algoritmaları

İki sabit noktalı ikili sayının işaretli gösterimde çarpımı, kalem kağıtla yapılırsa bir dizi ardışık kaydırma ve toplama işlemleriyle gerçekleşir. Bunu bir sayısal örnekle gösterelim.

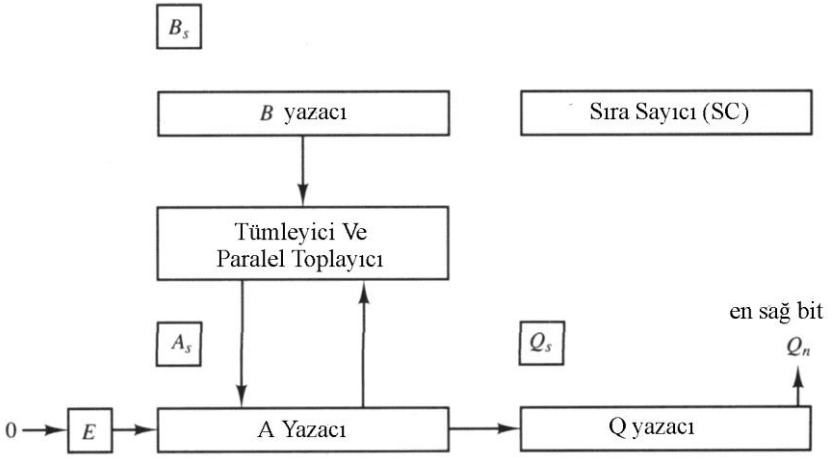
```
 23        10111    Çarpılan
 19    ×   10011    Çarpan
           10111
          10111
         00000      +
        00000
       10111
437  110110101      Sonuç
```

İşlem çarpmanın en önemsiz bitinden başlayarak sırayla bütün bitlerine bakılarak yapılır. Eğer çarpanın biti 1 ise çarpılan aşağıya kopya edilir. Aksi halde aşağıya sıfır kopya edilir. Her kopya işleminde sola bir kaydırma yapılır. Son bit içinde kaydırma ve yazma yapıldıktan sonra bunlar toplanır. Toplam çarpımı verir.

Çarpımın işareti çarpanla çarpılanın işaretlerinden bulunur. İşaretler aynı ise çarpım pozitif, farklı ise negatiftir.

İşaretli Sayılar İçin Donanımın Kurulması

Sayısal bir bilgisayarda çarpma işlemi yapılacağı zaman, işlemi biraz değiştirmek uygun olur. Önce yukarıdaki örnekteki gibi bütün ikili sayıları yazaçlara yazmak yerine, sadece iki sayının toplamını almak daha uygun olur. Daha sonra her sayı bu toplama eklenerek sonuca gidilir. İkincisi çarpımı sola kaydırmak yerine, kısmi toplam sağa kaydırılır. Böylece çarpılan daima yerinde kalmış olur. Üçüncüsü doğal olarak çarpan bit 0 ise, sıfırları toplamaya gerek yoktur.



Şekil 10.5 Çarpma işlemi için donanım

Çarpım için donanım şekil 10.1 de gösterilene 2 yazaç eklemekle elde edilir (Şekil 10.5). Çarpan Q yazacında, işareti Q_S dedir. Sıra sayıcıya çarpandaki bit sayısı daha önceden yazılır. Her kısmi toplam elde sinde bu sayı 1 azaltılır. Sayaç sıfır olunca çarpma işlemi bitmiştir, işlem durur.

İşlemin başında çarpılan B yazacında, çarpan Q dadır. A ve B nin toplamı kısmi çarpımı oluşturur. Kısmi çarpım EA yazacına aktarılır. Kısmi çarpım ve çarpan sağa kaydırılır. Bu işlem shr EAQ ile gösterilmektedir. A nın en önemsiz biti Q nun en önemli bitine kaydırılır. E biti A nın en önemli bitine ve E ye de 0 kaydırılır. Kaydırma işleminden sonra kısmi çarpımın 1 biti Q ya kaymış ve çarpanın bitlerini 1 bit sağa doğru itmiştir. Bu durumda çarpanın en sağ biti Q_n e kayar. Kayma işleminden sonra bu bitin 0 mı 1 mi olduğuna bakılır.

Donanım Algoritması

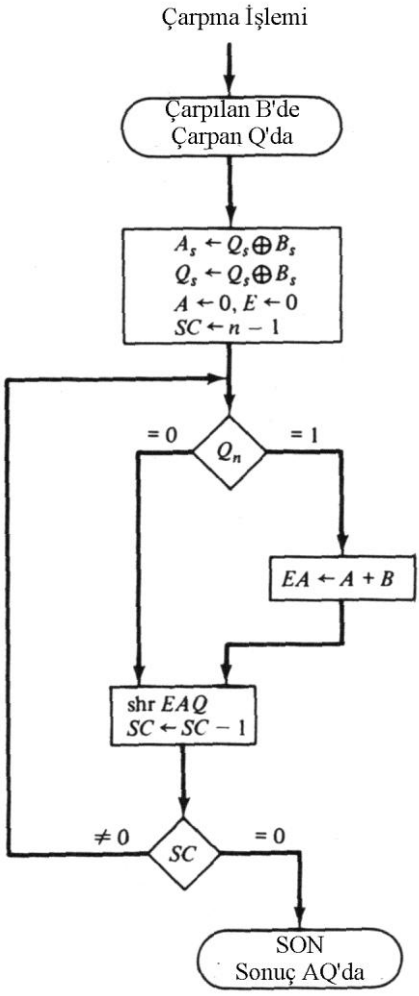
Şekil 10.6 da donanım algoritması verilmektedir. Baştan çarpılan B de ve çarpan Q dadır. Bunların işaretleri B_S ve Q_S dedir. Bunlar karşılaştırılır ve sonuç A_S ve Q_S ye yazılır. İşlemin sonucu tek yazaca sığmaz. Sonuç A ve Q yazaçlarında olacaktır. A ve E temizlenir. (SC) Sıra sayıcıya çarpanın bit sayısı yazılır. Burada verilerin bellekten yazaçlara aktarıldığı ve n bit uzunlukta olduğu varsayılıyor. Bitin biri işaret olduğundan, verilerin sayı kısmı $n - 1$ bit olur.

Yerleşmeler tamamlandıktan sonra çarpanın en önemsiz biti Q_n dedir. Bu bit test edilir. Eğer 1 ise çarpılan sayı A daki kısmi çarpıma eklenir. Eğer 0 ise hiçbir şey yapılmaz. Her iki durumdan sonra EAQ yazacı 1 bit sağa kaydırılır. SC 1 azaltılır, sonra sıfır olup olmadığına bakılır. Eğer 0 değil ise işlem tekrarlanır. Eğer $SC = 0$ ise işlem durur. Kısmi çarpan her defasında Q nun içine 1 bit 1 bit girer ve çarpanın en önemsiz biti kaybolur. En sonunda çarpan kaybolur. Sonuç A ve Q dadır.

Bir önceki sayısal çarpma örneği şekil 10.6 daki akış şemasına uygun olarak çizelge 10.2 de adım adım verilmektedir.

Booth Çarpım Algoritması

Booth algoritması 2 ye göre tümleyen biçiminde verilmiş olan ikili tam sayıların çarpımını belirler. Çarpandaki 0 lar üzerinde çalışılırsa toplama gerektirmez, sadece kaydırma gerekir. Eğer 1'ler üzerine çalışılırsa çıkarma ve kaydırma gerekir. Bu kısaca 1 lerin dizisinin en yüksek mertebesinden 0 ların dizisi atılır şeklinde ifade edilebilir. Örneğin 14 sayısı



Şekil 10.6 Çarpma işlemi için akış şeması.

ikili olarak 001110 şeklinde gösterilir. Burada 1 lerin dizisi 2^3 ten 2^1 e kadar devam etmektedir. Yani $2^3 + 2^2 + 2^1$ dir. Bunun yerine $2^4 - 2^1$ yazılabilir. Genel olarak 1 lerin dizisi k dan m e kadar gidiyorsa sayı = $2^{k+1} - 2^m$ şeklinde yazılabilir. ($k = 3$, $m = 1$) Eğer 1 tek ise yani iki yanında sıfır varsa $2^{k+1} - 2^k = 2$ yazılabilir. Bu durumda $M \times$ Sayı = $M2^{k+1} - M \times 2^m$ olur. Yani çarpılan sola kaydırılacak, m defa kaydırılınca ise çarpılan kısmi çarpımdan çıkarılacaktır.

Çizelge 10.2 İkili çarpma için sayısal örnek

Çarpılan $B = 10111$	E	A	Q	SC
Çarpan Q da	0	00000	10011	101
$Q_n = 1$; B yi topla			10111	
İlk kısmi çarpım		0	10111	
Mantıksal sağa kaydır EAQ yu	0	01011	11001	100
$Q_n = 1$; B yi topla			10111	
İkinci kısmi çarpım		1	00010	
Mantıksal sağa kaydır EAQ yu	0	10001	01100	011
$Q_n = 0$; Mantıksal sağa kaydır EAQ yu	0	01000	10110	010
$Q_n = 0$; Mantıksal sağa kaydır EAQ yu	0	00100	01011	001
$Q_n = 1$; B yi topla			10111	
İlk kısmi çarpım		0	11011	
Mantıksal sağa kaydır EAQ yu	0	01101	10101	000
Sonuç $AQ = 0110110101$				

Bütün çarpım algoritmalarında olduğu gibi Booth algoritması da, çarpan bitin incelenmesini ve kısmi çarpımın kaydırılmasını gerektirir. Çarpılan sola kaydırılacağına kısmi çarpım sağa kaydırılır. Kaydırmadan önce çarpılan kısmi çarpıma eklenebilir, çıkarılabilir veya hiç bir işlem yapılmadan kaydırma yapılabilir. Bunun kuralları şunlardır.

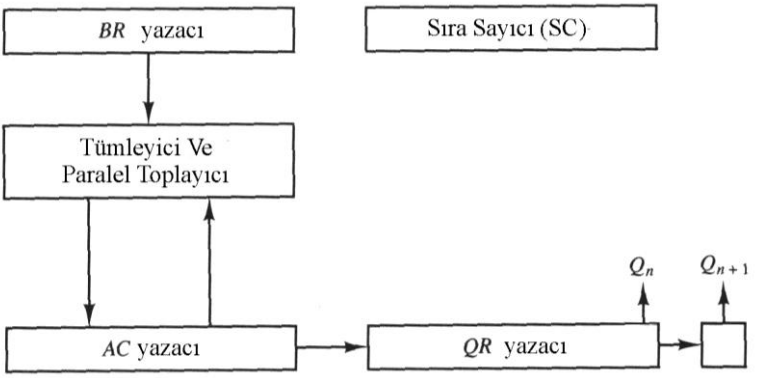
1. Çarpılan ilk önemsiz 1 biti ile çarpılacağı zaman kısmi çarpımdan çıkarılır.
2. İlk 0 ın gelmesiyle, yani 1 çarpanından sonra 0 çarpanının gelmesiyle çarpılan kısmi çarpıma eklenir.
3. Eğer çarpan bit değişmez ise kısmi çarpım değiştirilmez sadece kaydırılır.

Bu algoritma 2 ye göre tümleyenli pozitif veya negatif çarpanlar için çalışır. Örneğin bir çarpan –14 ikili olarak 110010 şeklinde 2 ye göre tümleyen olarak yazılır ve $-2^4 + 2^2 - 2^1 = -14$ olur.

Booth algoritması şekil 10.7 de gösterilen donanıma gerek duyar. Bu donanım şekil 10.5 tekinin aynıdır. Sadece işaret bitleri yazaçlardan ayrılmamıştır. Ayrıca karşılaştırma için Q_{n+1} biti olarak bir yaz-boz ilave edilmiştir. A, B ve Q yazaç isimleri AC, BR ve QR ile yer değiştirmiştir. Booth algoritmasının akış şeması şekil 10.8 de gösterilmektedir. Taşma olmaz. Çünkü çarpılanın toplanması veya çıkarılması birbirini takip eder.

Dolayısıyla toplanan sayılar daima ters işaretlidir. Bundan sonraki adım kısmi çarpımın sağ kaydırılmasıdır. Kaydırmaya Q_{n+1} dahildir. Kaydırma aritmetik kaydırmadır. *AC* ve *QR* yi beraberce kaydırır. Ancak işaret bitini olduğu gibi bırakır. *SC* yi bir azaltır. Bu döngü *n* defa devam eder.

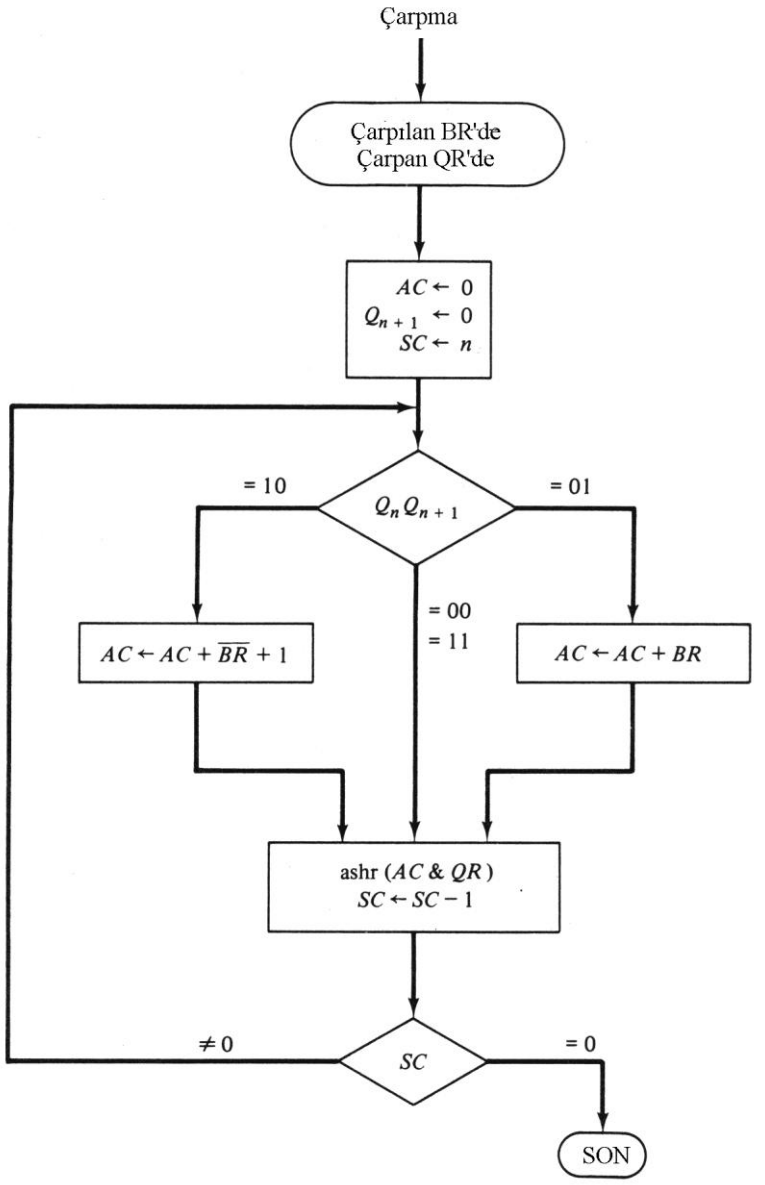
Booth algoritmasının sayısal örneği çizelge 10.3 te $n = 5$ için verilmiştir. -9 ile -13 ün çarpımı adım adım gösterilmektedir. Hem çarpan *QR* hem de çarpılan *BR* negatiftir. *AC* ve *QR* de elde edilen sonuç 10 bittir ve pozitiftir. *Q* da görünen bit çarpanın orijinal işaretidir ve sonuca dahil değildir.



Şekil 10.7 Booth algoritması için donanım

Çizelge 10.3 Booth algoritması için çarpma örneği

$Q_n\ Q_{n+1}$	$BR = 10111$ $BR + 1 = 01001$	*AC*	*QR*	Q_{n+1}	*SC*
	Başlangıçta	00000	10011	0	101
1 0	*BR* yi çıkar	01001			
		01001			
	ashr	00100	11001	1	100
1 1	ashr	00010	01100	1	011
0 1	*BR* yi topla	10111			
		11001			
	ashr	11100	10110	0	010
0 0	ashr	11110	01011	0	001
1 0	*BR* yi çıkar	01001			
		00111			
	ashr	00011	10101	1	000



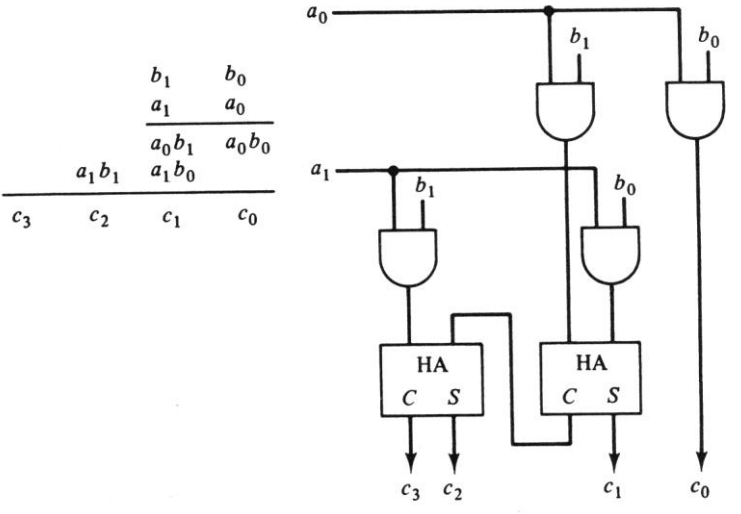
Şekil 10.8 2 ye göre tümleyenli sayıların çarpımı için Booth algoritması

Dizi Çarpım

Çarpmada çarpanın bitlerinin her seferinde denetlenmesi, kısmi çarpanların oluşturulması sıralı bir işlem olup toplama ve kaydırma mikro işlemlerini de gerektirir. İki ikili sayının çarpımının bir birleşik devre yardımı ile bir tek mikro işlemle yapılması ve sonucun bir defada elde si mümkündür. Bu çarpma işlemini yapmanın kısa yoludur. Sinyaller kapılardan geçer ve çarpma dizisini oluşturur. Bütün zaman kapılarda harcanan zamandır. Dizi çarpımda çok sayıda kapıya ihtiyaç vardır. Bu yüzden IC lerin yapılması kadar ekonomik değildir.

Dizi çarpıcının nasıl yapıldığını anlamak için şekil 10.9 da 2 bitlik iki sayının çarpımının algoritması ve devresi verilmektedir. Çarpan a_1 ve a_0, çarpılan bitler b_1, b_0 ve çarpım ise c_3, c_2, c_1, c_0 olsun. İlk kısmi çarpım b_1 b_0 ile a_0 ın çarpımından oluşur. İki bit yani a_0 ile b_0 ın çarpımı eğer iki bitte 1 ise 1 dir, aksi halde 0 dır. Bu ise AND işlemidir, bir tane VE kapısıyla sağlanır. İkinci kısmi çarpım a_1 in b_1 ve b_0 ile çarpılmasıyla elde edilir, bir basamak solda olmalıdır. İki kısmi çarpım iki kısmi toplayıcı ile toplanır. Kısmi toplamlarda çok sayıda bit bulunuyorsa tam toplayıcılar kullanmak gerekir. Çarpımın ilk bitinin tam toplayıcıya gitmesine gerek yoktur. Bu bir VE kapısı ile bulunur.

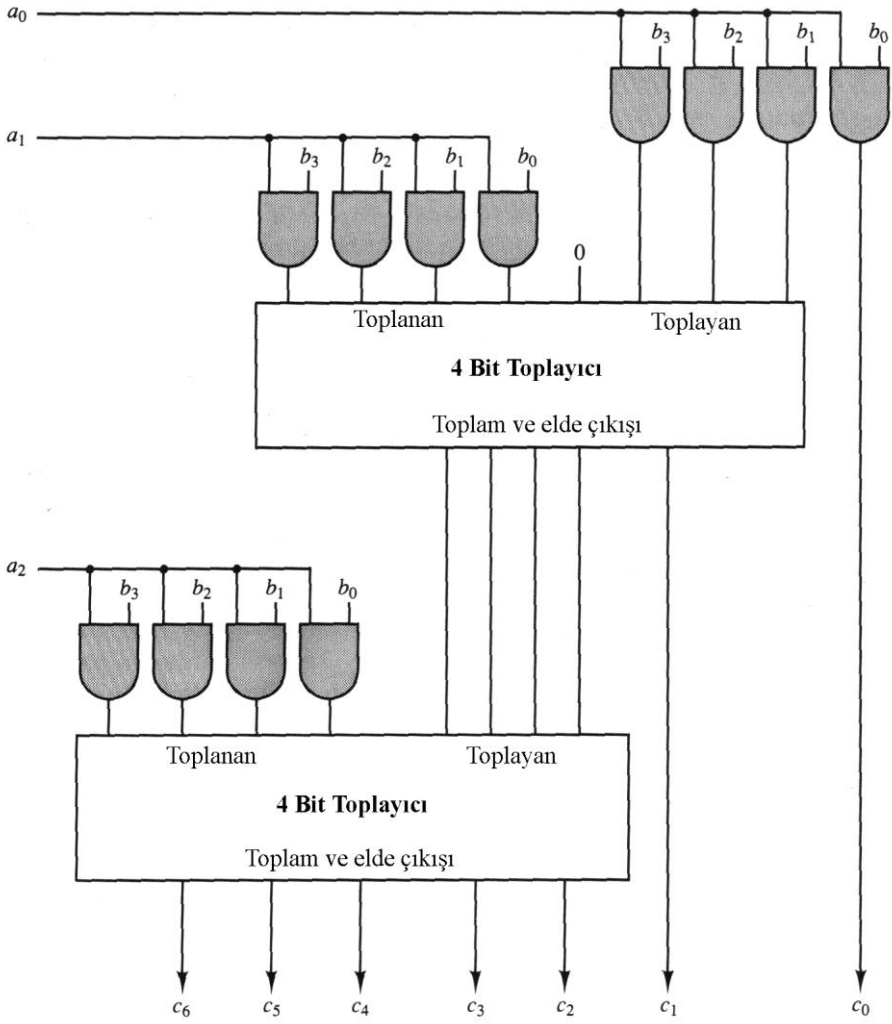
Daha fazla bit için ikili çarpımı oluşturacak bir birleşik devre benzer şekilde çizilebilir. Çarpanın bir biti çarpılanın her bitiyle VE lenir. Bütün çarpan bitleri için aynı devreler yapılır. Her aşamanın kısmi çarpımları



Şekil 10.9 2 bitlik iki sayının dizi çarpımı

tam toplayıcılarla diğerlerine eklenir. Son aşama çarpımı verir. Çarpan j bit ve çarpılan k bit ise $j \times k$ tane VE kapısı ve $(j - 1)$ tane k bit toplayıcıya gerek vardır. Sonuç $j + k$ bit olur.

İkinci bir örnek olarak 4 bit ikili sayıyı 3 bit ikili sayı ile çarpan bir çarpıcı düşünelim. Çarpılan b_3, b_2, b_1, b_0 ve çarpan a_2, a_1, a_0 olsun. $k = 4$ ve $j = 3$ olduğundan 12 VE kapısı ve 2 tane 4 bit toplayıcı gerekir. Sonuç 7 bit olacaktır. Şekil 10.10 mantıksal şemasını vermektedir.



Şekil 10.10 Dört bitlik ile üç bitliğin dizi çarpımı

10.4 Bölme Algoritmaları

Kısmi Kalan

; İki sabit noktalı ikili sayının işaretli sayı biçiminde gösteriminin bölümü kağıt ve kalemle bir dizi ardışık karşılaştırma, kaydırma ve çıkarma işlemlerini gerektirir. İkili bölme, ondalık bölmeden daha basittir, işleme giren sayılanı basamaklarında sadece 0 ve 1 vardır. Ayrıca bölenin kalan içinde kaç defa olduğunu bulmak kolaydır. Bölme işlemi şekil 10.11 de sayısal olarak gösterilmektedir. Bölüm *B* 5 bit, bölünen *A* 10 bittir. Bölünenin önemli 5 biti bölenle karşılaştırılır. Bu 5 bit bölenden küçük olduğundan bölünenin önemli bitlerinden 6 tanesi alınır. 6 bit sayı bölenden büyük olduğundan bölüm hanesine 1 yazılır. Bölümün yazılmasında 1 bitini bölünenin altıncı biti hizasına yazarak hangi bitin kalan olarak gönderileceği daha kolay bulunur. Bölün 1 bit sağa kaydırılarak yazılır ve bölünen sayıdan çıkarılır. Eğer kısmi kalan bölenden küçük ise bölüme 0 yazılır ve bölen tekrar sağa kaydırılır. Her durumda bölen sağa kaydırılır. Sonuçta hem bölüm, hem de kalan elde edilir.

İşaretli Sayılar İçin Donanım Kurulması

Sayısal bir bilgisayara bölme işlemi için donanım konacaksa, işlemleri biraz değiştirmek gerekir. Böleni sağa kaydırmak yerine bölünen veya kısmi kalan sola kaydırılır. Böylece sayılar istenen göreceli konumlara gelmiş olurlar. Çıkarma işlemi *A* ya *B* nin 2 ye göre tümleyenn eklenmesiyle yapılır. Sayıların göreceli durumu çıkış elde sinden elde edilir.

Bölen:	11010	Bölüm = *Q*
B = 10001	)0111000000	Bölünen = *A*
	01110	*A* < *B* nin 5 biti, bölünenin 5 biti
	011100	*A* ≥ *B* nin 6 biti
	-10001	*B* yi mantıksal sağa kaydır ve çıkart; *Q* ya 1 gir
	-010110	Kalan ≥ *B* nin 7 biti
	- -10001	*B* yi mantıksal sağa kaydır ve çıkart; *Q* ya 1 gir
	- -001010	Kalan < *B*; *Q* ya sıfır gir; *B* yi mantıksal sağa kaydır
	- - -010100	Kalan ≥ *B*
	- - - -10001	*B* yi mantıksal sağa kaydır ve çıkart; *Q* ya 1 gir
	- - - -000110	Kalan < *B*; *Q* ya 0 gir
	- - - - -00110	Son kalan

Şekil 10.11 İkili bölme işlemi için örnek

Bölen $B = 10001$, $\overline{B} + 1 = 01111$

	E	A	Q	SC
Bölünen:		01110	00000	5
shl EAQ	0	11100	00000	
$\overline{B}$ + 1 i topla		01111		
$E = 1$	1	01011		
$Q_n = 1$ yaz	1	01011	00001	4
shl EAQ	0	10110	00010	
$\overline{B}$ + 1 i topla		01111		
$E = 1$	1	00101		
$Q_n = 1$ yaz	1	00101	00011	3
shl EAQ	0	01010	00110	
$\overline{B}$ + 1 i topla		01111		
$E = 0$; $Q_n = 0$ kalır	0	11001	00110	
B yi topla		10001		
Kalan tekrar bulunur	1	01010	2	
shl EAQ	0	10100	01100	
$\overline{B}$ + 1 i topla		01111		
$E = 1$	1	00011		
$Q_n = 1$ yaz	1	00011	01101	1
shl EAQ	0	00110	11010	
$\overline{B}$ + 1 i topla		01111		
$E = 0$; $Q_n = 0$ kalır	0	10101	11010	
B yi topla		10001		
Kalan tekrar bulunur	1	00110		
E ihmal edilir				
Kalan A da		00110		
Bölüm Q dadır			11010	

Şekil 10.12 Sayısal donanım için ikili bölme örneği

Bölme işlemi için gerekli donanım, çarpma işlemi için gerekenin aynısıdır. (şekil 10.5). EAQ yazacı Q_n e 0 yazılarak bu defa sola kaydırılır. E nin eski değeri kaybolur. Olayı açıklayan sayısal örnek şekil 10.12 de verilmiştir.

Bölen B ve bölünen A ve Q dadır. Bölünen sola 1 kaydırılır ve bölenin 2 ye göre tümleyen kaydırmaya eklenir. Sayıların birbirine göre durumu ($A \geq B$ veya $A < B$) E den bellidir. Eğer $E = 1$ ise $A \geq B$ dir ve bölüm olarak 1 sayısı Q nun sağındaki en önemsiz bite yazılır. Kısmi kalan sola 1 kaydırılır. Eğer $E = 0$ ise $A < B$ demektir ve Q_n den 0 girilir. B nin değeri kısmi kalana eklenir. Böylece A daki kısmi kalanın eski değeri

toplanana kadar devam eder. Kısmi kalan sola kaydırıldığında bölüm bitleri yani *Q* larda sola kaydırılır. 5 olunca bölüm *Q* da son kalan ise *A* dadır.

Algoritmanın akış şemasından önce sonucun işareti ve taşma durumu düşünülmelidir. Bölümün işareti bölenle bölünenin işaretinden bulunur. Eğer iki işaret aynı ise bölümün işareti pozitif, farklı ise negatiftir. Kalanın işareti bölünenin işareti ile aynıdır

Bölüm Taşması

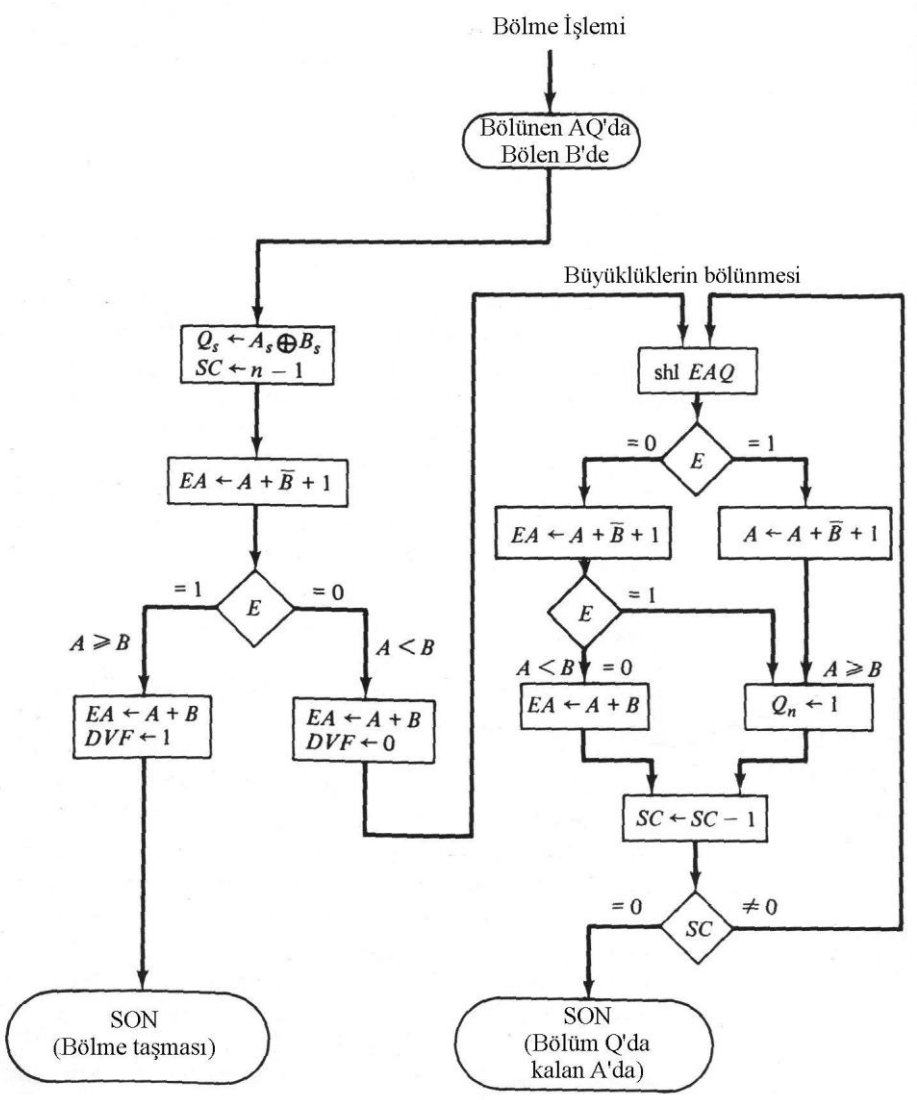
Bölme işlemi bölümde bir taşma ile sonuçlanabilir. Bu halen kağıtta önemli değil ama donanımda önemli bir problemdir. Çünkü yazaçların uzunlukları sınırlı olduğundan, standart uzunluktan farklı bir sayı tutamaz. Bunu anlamak için 5 bit yazaçları olan bir sistem düşünelim. Birinci yazaçta bölen, 2. yazaçta bölünen olsun. Eğer bölünenin ilk beş biti bölenden büyük ise bölüm 6 bit olur. Bölüm standart 5 bite konduğunda taşma biti için 1 bit yer gerekir. Yani bir fazla yaz-boz gereklidir. Normal bilgisayar işlemlerinde bölüm taşma şartlarından kaçınılmalıdır. Çünkü bölüm çok uzun olabilir. Bellek kelimeleri standart uzunlukta olduğundan bölüm belleğe yazılamayabilir. Bu durum önceden anlaşılmalı veya donanım yada yazılım veya ikisiyle birlikte bir çözüm getirilmelidir.

Eğer bölünen, bölenin 2 katı uzunlukta ise taşma şöyle tanımlanabilir. Eğer bölünenin bitlerinin önemli yarısının oluşturduğu sayı bölen sayıdan büyük veya eşitse bölüm taşma şartı oluşur. Sıfır ile bölme işleminden de kaçınmak gerekir. Taşma şartı buna da dikkat etmelidir. Taşma şartının denetimi özel bir yaz-boz kümesi ile yapılır.
Bu yaz-boz kümesine bölme taşma yaz-bozu veya *DVF* denir.

Aslında bölme taşması bir çok biçimde denetlenebilir. Bazı bilgisayarlarda her bölme işleminden sonra *DVF* nin 1 yapıldığına dikkat etmek ve önlem almak programcıların görevidir. Önlem olarak bir alt programa dallanırlar. Bu alt programda taşmadan kaçınmak için sayılarda basamak kaydırması işlemi yapılabilir. Eski bilgisayarlarda bölme işlemi durdurulur ve bölme hatası olarak kullanıcıya iletilirdi. Ancak bilgisayarın durdurulması zaman kaybı yarattığından iyi değildir. Birçok bilgisayarda *DVF* 1 yapılmışsa bir kesme isteği olur. Kesme bilgisayarın programın icrasını durdurmasını ve taşmaya karşı bir düzeltme yapılacak ise servis alt programına dallanılmasını sağlar. En çok kullanılan alt program, bir hata mesajı ile programın neden tamamlanamadığını bildirmektir. Bu durumda programcı bir basamak kaydırma programı yazabilir veya başka bir önlem alabilir. Bölme taşmasından kurtulmanın en iyi yolu kayan noktalı sayı kullanmaktır. Bu gösterimde taşma çok basit bir şekilde önlenebilir.

Donanım Algoritması

Bölüm algoritmasının donanımı şekil 10.13 te verilmiştir. Bölünen *A* ve *Q* da bölen *B* dedir. Sonucun işareti Q_s ye yazılır. *SC* ye bölümdeki bit sayısı yazılır. Çarpmada olduğu gibi verilerin yazaçlara bellekten aktarıldığı ve *n* bitlik kelimeler olduğu varsayılmaktadır. Veriler işaretleri ile yerleştirildiğinden kelimenin bir biti işaret bitidir ve sayı *n*-1 bit olur.



Şekil 10.13 Bölme işlemi için akış şeması

Bölüm taşma durumu B nin, bölünenin önemli bitlerinin yarısından çıkarılmasıyla anlaşılır. Bölünenin bu kısmı başlangıçta A dadır. Eğer $A \geq B$ ise taşma vardır. *DVF* 1 yapılır ve işlem durdurulur. Eğer $A < B$ ise bölüm taşması yoktur. B tekrar A ya eklenir.

AQ sola kaydırılarak bölme işlemi başlatılır. En yüksek önemli bit E ye geçer. $E = 1$ ise $EA > B$ dir. B, EA dan çıkarılır ve Q_n e 1 girilir. Bu bölümün bir biti olur. A yazacı bölünenin en yüksek önemde bitini E ye geçirmiştir. Dolayısıyla A nın değeri $EA - 2^{n-1}$ dir. Bu değere B nin 2 ye göre tümleyen eklenirse

$$(EA - 2^{n-1}) + (2^{n-1} - B) = EA - B$$

Bu toplama işlemi sonucunda eğer E nin 1 kalması isteniyorsa toplama işlemi sonucu elde E ye aktarılmaz.

Eğer sola kaydırma işlemi E ye 0 yükledi ise B nin 2 ye göre tümleyen A ya eklenir ve elde E ye taşınır. Eğer $E = 1$ ise $A \geq B$ dir $Q_n = 1$ yapılır. Eğer $E = 0$ ise $A < B$ dir, A ya B eklenir ve Q_n e 0 yazılır (kaydırma sırasında).

$SC = 0$ olana kadar işlemler tekrarlanır. A da kısmi kalan bulunur. n-1 defa tekrardan sonra bölüm Q da işareti Q_S de olur. Kalan A da ve işareti A_S de bölünenle aynı işaretlidir.

Diğer Algoritmalar

İadeli Yöntemi

Yukarıda anlatılan yönteme iadeli yöntemi denir. Bunun nedeni kısmi kalanın negatif çıkması halinde bölenin tekrar toplanmasıdır. Bu yöntemin dışında daha iki yöntem vardır. Bunlar karşılaştırma yöntemi ile iadesiz yöntemleridir. Karşılaştırma yönteminde, çıkarma işleminden önce iki sayı karşılaştırılır $A \geq B$ ise çıkarma yapılır. $A < B$ ise bir şey yapılmaz ve kısmi kalan sola kaydırılır, sonra sayılar tekrar karşılaştırılır. Karşılaştırma işlemi E ye aktarılmadan son elde nin (paralel toplayıcıdaki son elde) incelenmesiyle anlaşılır.

Kıyaslama ve İadesiz Yöntem

İadesiz yönteminde eğer fark negatifse B bu farka toplanmaz. Bunun yerine fark sola kaydırılır, B toplanır. Bunun neden yapıldığını anlamak için $A < B$ durumunu düşünelim. Şekil 10.13 deki akış şemasından yapılan işlemin $A - B + B$ olduğu anlaşılır. Çünkü kalan sola kaydırılır yani 2 ile çarpılır sonra B çıkarılır. Böylece $2(A - B + B) - B = 2A - B$ olur. Bu yöntemde $A - B$ olduğu gibi bırakılır ve döngü tekrar başladığında kalan sola kaydırılır. Yeni $2(A - B)$ elde edilir. Buna B eklenince $2(A - B) + B = 2A - B$ elde edilir ki aynı sonuç elde edilmiş olur. Böylece bu yöntemde Q_n in değeri 1 ise B çıkarılır. Eğer Q_n in değeri 0 olursa B eklenir. Dolayısıyla kısmi kalanın eski değerine dönmesi gerekmez. Bu işlem $A < B$ halinde B nin eklenmesi işlemini ortadan kaldırır. Fakat bir önceki sonucu

hatırlaması için özel denetim mantığı gerektirir. Bölünen ilk defa kaydırıldığında *B* çıkarılmalıdır. Eğer bölünün son biti 0 ise doğal olarak son kalana *B* eklenmelidir.

10.5 Kayan Noktalı Aritmetik İşlemler

Tamsayı Bildirim Deyimi

Yüksek seviyeli programlama dillerinin çoğunda kayan noktalı sayıların belirtilmesi ve kullanılması için buyrukları vardır. En çok kullanılan ortak yöntem bu sayıların sabit noktalı sayıların aksine olarak gerçel tanımıyla tanımlanmasıdır. Sabit noktalı sayılar tamsayı tanımıyla tanımlanırlar. Bu tür derleyicileri çalıştıran bilgisayarların kayan noktalı sayılarla işlem yapacak aletleri bulunmalıdır. İşlemler genellikle donanım içinde yapılır. Eğer işlemi yapacak donanım yoksa, derleyicinin içinde kayan noktalı işlemleri yapacak yordamlar bulunmalıdır. Her ne kadar donanım yöntemi pahalı ise de çok etkin çalışır. Çok küçükleri hariç bütün bilgisayarlarda donanım mevcuttur.

Temel Düşünceler

Bilgisayarın yazacında mevcut bir kayan noktalı sayı iki kısımdan oluşur. Kesir *m* ve üs *e*. Bu iki parçayla sayı *r* taban olmak üzere

$$m \times r^e$$

şeklinde verilir. Kesir bir tamsayı kesridir. Kesir noktası ve taban yazaçlarda yer olmaz, sadece varsayılır. Şimdi kesir ve taban olarak 10 sayısını düşünelim. Örnek 537.25 ondalık kesri yazaçta $m = 53725$ ve $e = 3$ ile gösterilir ve

$$.53725 \times 10^3$$

sayısını temsil eder.

Kayan noktalı bir sayı kesrin en önemli basamağı 0 olmayacak şekilde normalize edilir. Bu şekilde maksimum sayıda önemli basamak olur. Sıfır sayısı normalize edilemez. Çünkü sıfır olmayan basamağı yoktur, kesri ve üssü sıfır olarak temsil edilir.

Kayan noktalı gösterimi sayıların temsil edebileceği büyüklüğü arttırır. 48 bit kelime olan bir bilgisayar düşünelim. Bir bit işaret için ayrılmalıdır. Bu durumda sabit noktalı tamsayıların gösterebileceği en büyük sayılar $(2^{47} - 1)$ yani yaklaşık 10^{14} kadar olur. 48 bit kayan noktalı gösterim için kullanılırsa 36 bit kesir 12 bit üs için alınır. İki bit işaret için (1 bit kesir, 1 bit üs için) kullanılırsa en büyük sayı bulunur. Bu sayılar 35

tane 1 içeren kesirden ve işaret biti dahil 12 bit üsten bulunmuştur. $2^{11} - 1 = 2047$ dir. Ondalık olarak yaklaşık 10^{615} sayısı ifade edilebilir. Kesir 35 bitten (işaret biti hariç) oluşur. Bunun tamsayı olduğu düşünülürse $2^{35} - 1$ yani yaklaşık 10^{10} sayısı elde edilir.

Mikro kelimeleri daha kısa olan bilgisayarlar kayan noktalı sayıları göstermek için iki veya daha fazla kelime kullanırlar. 8 bitlik bir mikro bilgisayarın 4 kelime kullanması gerekir. 8 bit üs için 24 bit kesir için kullanılır.

Kayan noktalı sayılarla aritmetik işlemler çok karışıktır. İcraları daha uzun zaman alır ve donanımları sabit noktalı sayılara göre çok daha karışık olur. İki sayının toplama ve çıkarması üs noktasının hizalanmasını gerektirir. Kesirlerin toplanması veya çıkarılmasından önce üsler eşit yapılmalıdır. Aşağıdaki iki kayan noktalı sayının toplamına bakıldığında

$$\begin{aligned} &.5372400 \times 10^{2} \\ &+ .1580000 \times 10^{-1} \end{aligned}$$

önce iki üssün eşit yapılması gerekir, sonra kesirler toplanabilir. Ya birinci sayı 3 basamak sola yada ikinci sayı 3 basamak sağa kaydırılarak üsler eşitlenebilir. Kesirler yazaçlarda bulunduğundan sola kaydırma en önemli 3 basamağın kaybolmasına neden olur. Sağa kaydırma ise önemsiz basamakların kaybolmasına neden olacaktır. Kaydırma üslerin farkı kadar yapılır. Bu yapıldıktan sonra kesirler toplanabilir.

$$\begin{aligned} &.5372400 \times 10^{2} \\ &+ .0001580 \times 10^{2} \\ \hline &.5373980 \times 10^{2} \end{aligned}$$

İki normalize edilmiş kesir toplandığında toplamda bir taşma olabilir. Bu durumda toplam bir defa sağa kaydırılarak ve üsse 1 eklenerek kolayca taşmadan kurtulunur. İki sayı birbirinden çıkarıldığında sonuçta en önemli basamaklarda sıfır bulunabilir. Örneğin

$$\begin{aligned} &.56780 \times 10^{5} \\ &- .56430 \times 10^{5} \\ \hline &.00350 \times 10^{5} \end{aligned}$$

en önemli basamakta sıfır bulunan bir kesirde çökme vardır denir. Çökme içeren bir sayıyı normalize etmek için kesir sola kaydırılır ve üs azaltılır. En önemli basamakta sıfır kalmayıncaya kadar bu işleme devam edilir. Yukarıdaki örnekte iki defa sola kaydırma sonucunda 0.35000×10^{3} elde

edilir. Bir çok bilgisayarda her işlemin sonunda sonuçlar normalize edilir. Böylece bilgisayarda bütün değerler normalize durumdadır.

Kayan noktalı çarpma ve bölme hizalama işlemi gerektirmez. Çarpım iki kesrin çarpılması ve üslerin toplanmasıyla elde edilir. Bölme işleminde ise kesirler bölünür ve üsler çıkarılır. Kesirlerle yapılan işlemler sabit noktalı sayılarla yapılanların aynıdır. Dolayısıyla iki sayı sabit noktalı sayılarda kullanılan yazaçları ve donanımı kullanabilir. Üslerle yapılan işlemler ise; bunların karşılaştırılması ve arttırma (kesirlerin hizalanması için); toplama ve çıkarma (çarpma ve bölme için) ve azaltma (normalize için) dir. Üs, 3 gösterimden herhangi biriyle yani işaretli sayı, 2 ye göre tümleyen veya 1 e göre tümleyen şeklinde olabilir.

Birçok bilgisayarda kullanılan dördüncü bir gösterim şekli kısıtlanmış üstür. Bu türde işaretler kaldırılır ve bütün üsler pozitif yapılır. Bu bir örnekle açıklandığında, bir üssün kısıtları –50 ile + 49 olsun. Üs işaret hariç iki basamakta ifade edilmektedir. Buna göre gerçek üs e ise kısıtlı üs e + 50 olursa negatif üs kalmaz. Yani bilgisayarda üsler 0 ile 99 arasında olur, hepsi pozitiftir. 0 - 49 arası üsler negatif, 50 – 99 arası üsler pozitiftir. Üsten 50 çıkarılarak gerçek üs bulunur.

Bunların avantajı sadece pozitif sayılar bulunmasıdır. Ayrıca bunları karşılaştırmakta çok kolaydır. Sonuçta hizalama sırasında bu gerçek üsler karşılaştırılabilir. Bir diğer avantajı en küçük kısıtlı üs sıfırlardan oluşur. Dolayısıyla sıfırların kayan noktalı gösterimi sıfır kesir ve mümkün en küçük üs olur.

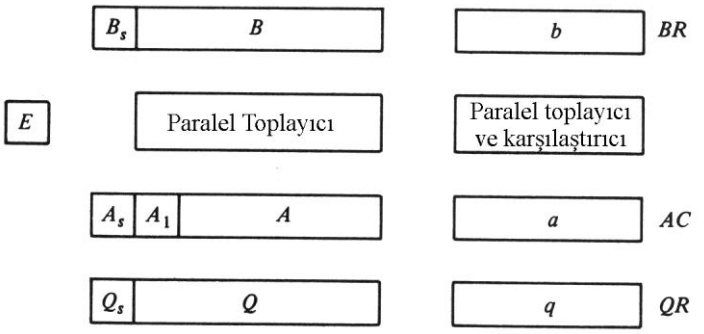
Yukarıdaki örneklerde kavramların anlaşılması için ondalık sayılar kullanıldı, aynı kavramlar ikili sayılar içinde geçerlidir. Burada geliştirilecek algoritmalar ikili sayılar içindir.

Yazaç Yapılanması

Kayan noktalı sayılar için yazaç yapılanması sabit noktalı sayılarınkine büyük benzerlik gösterir. Aynı yazaçlar ve toplayıcı, kesir üzerindeki işlemler için kullanılır. Üslerin işleminde ise farklılık vardır.

Kayan noktalı sayılar için yazaç organizasyonu şekil 10.14 te verilmektedir. 3 adet yazaç vardır. Bunlar *BR*, *AC* ve *QR* dir. Her bir yazaç iki parçaya bölünmüştür. Kesir kısmı, sabit noktalı sayılardaki ile aynı harflerle gösterilmiştir. Üs kısmı ise küçük harflerle gösterilmiştir.

Her kayan noktalı sayının kesrinin işaretli sayı gösteriminde verildiği, üssün ise kısıtlanmış üs olduğu varsayılıyor. Böylece *AC*, işareti A_S de ve sayı kısmı *A* da olan kesri tutar. Üs, *a* ile gösterilen kısımdadır. *A* nın en önemli bit A_1 ile gösterilmektedir. Normalize edilmiş bir sayı için bu bit 1 olmalıdır. *AC* sembolü bütün yazacı gösterir.



Şekil 10.14 Kayan noktalı aritmetik işlemler için yazaçlar

Benzer şekilde *BR* yazacı B_S, *B* ve *b*, *QR* yazacı Q_s, *Q* ve *q* alanlarına ayrılmıştır. Paralel toplayıcı iki kesri toplar ve toplamı *A* ya, taşmayı *E* ye aktarır. Üsler için ayrı bir toplayıcı kullanılır. Üsler kısıtlanmış olduğundan bunların belli bir üs işareti yoktur. Bunun yerine kısıtlı pozitif sayı ile gösterilirler. Kayan noktalı sayıların çok büyük oldukları ve bir taşma olmasının uzak ihtimal olduğu varsayılır. Dolayısıyla üslerde bir taşma ihmal edilir. Üsler bir karşılaştırıcıya bağlanmışlardır. Bunların göreceli karşılaştırılması 3 farklı ikili çıkış ile belirtilir.

Kesirdeki sayı daima bir kesir olarak kabul edilir ve ikili kesir noktasının bunun solunda olduğu varsayılır. Kayan noktalı için tamsayı kullanılması çarpma ve bölme işlemleri sırasında boyut problemlerine neden olur. Bundan dolayı burada kesir gösterimi kullanılacaktır.

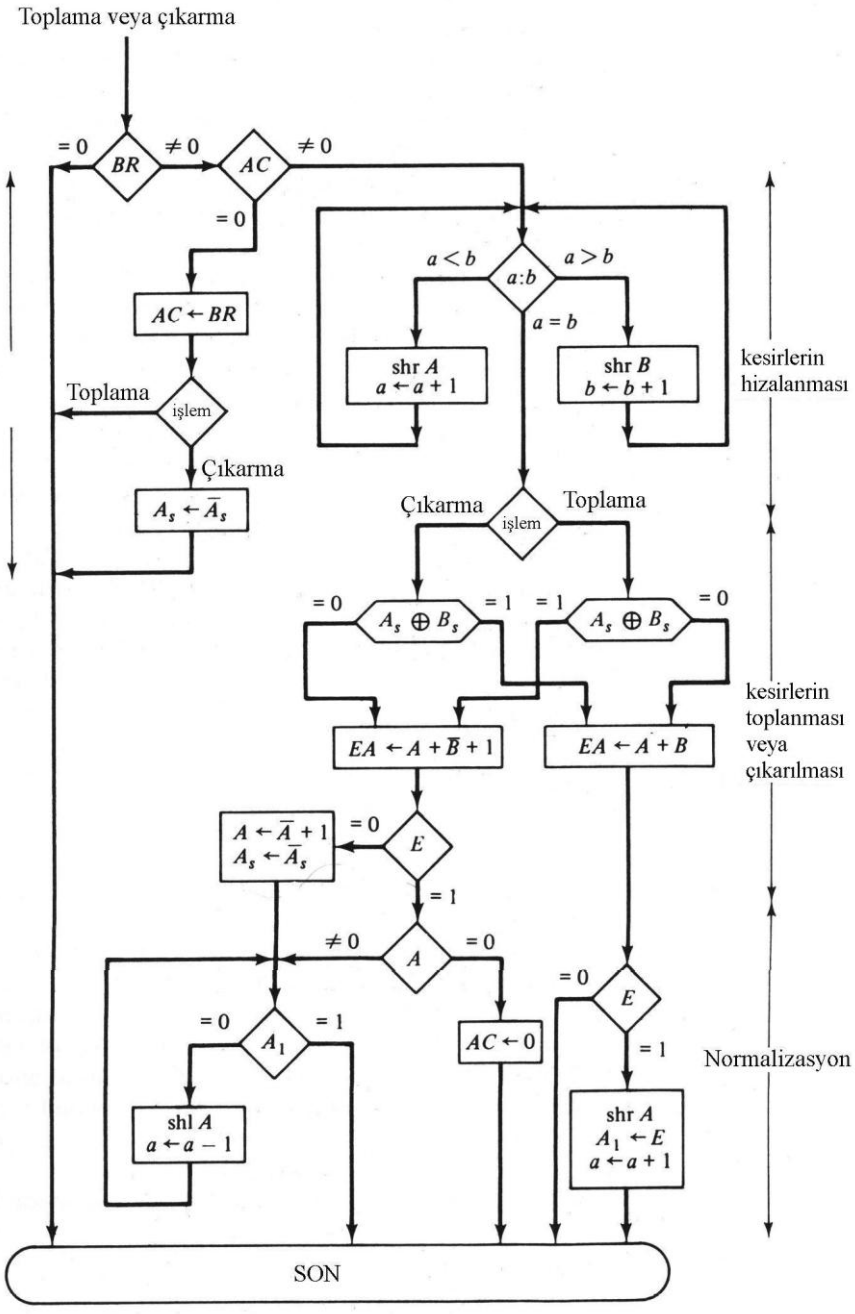
Yazaçlardaki sayıların daha önceden normalize edildikleri varsayılıyor. Her aritmetik işlemden sonra sonuç normalize edilir. Dolayısıyla bellekten gelen ve belleğe giden tüm veriler normalizedirler.

Toplama ve Çıkarma

Toplama ve çıkarma sırasında kayan noktalı veriler *AC* ve *BR* dedir. Fark veya toplam *AC* de oluşur. Algoritma 4 ardışık parçaya ayrılır.

1. Sıfırların denetimi
2. Kesirlerin hizalanması
3. Kesirlerin toplanması veya çıkarılması
4. Sonucun normalizasyonu

Kayan noktalı sıfır normalize edilemez. Normalize sırasında sıfır denetimi yapmak yerine gerekirse işlemin başlangıcında ve sonunda denetim yapılabilir. Kesirlerin hizalanması, onların işleme girmesinden önce yapılmalıdır. Kesirlerin toplanması veya çıkarılmasından sonra sonuç



Şekil 10.15 Kayan noktalı sayıların toplanması ve çıkarılması

normalize almayabilir. Normalize işlemiyle, belleğe gidecek verinin belleğe gitmeden önce normalize edilmiş olması sağlanır.

İki kayan noktalı sayının toplama ve çıkarmasına ait akış şeması şekil 10.15 te verilmiştir. Eğer *BR* sıfıra eşitse işlem sona erer. *AC* deki sayı sonuçtur. Eğer *AC* sıfır ise, *BR*, *AC* ye aktarılır ve eğer sayılar çıkarılıyorsa işareti tümlenir. Eğer sayıların hiçbiri sıfır değil ise sayıların hizalanmasına geçilir.

a ve *b* üslerine bağlı karşılaştırıcı bu iki sayının karşılaştırılması ile ilgili 3 sonuç verir. Eğer üsler eşit ise işlem derhal yapılır. Eşit değilse, üssü küçük olan kesir sağa kaydırılarak ve üssüde arttırılarak iki üs eşit hale getirilir. İşlem üsler eşit olana dek sürer.

İki kesrin toplanması veya çıkarılması sabit noktalı toplama ve çıkarma algoritmasının aynıdır (şekil 10.2). Sayı kısmı işleme bağlı olarak toplanır veya çıkarılır. İşareti verilir. İki sayı toplandığında taşma olursa, taşma *E* yaz-bozuna aktarılır. Eğer $E = 1$ ise bit A_1 e aktarılır (*E* biti) ve *A* nın diğer bitleri sağa kaydırılır. Üs arttırılarak doğru üs elde edilir. Çökme olayı olmaz. Çünkü kesrin biri normalize durumdadır ve kaydırılmamıştır.

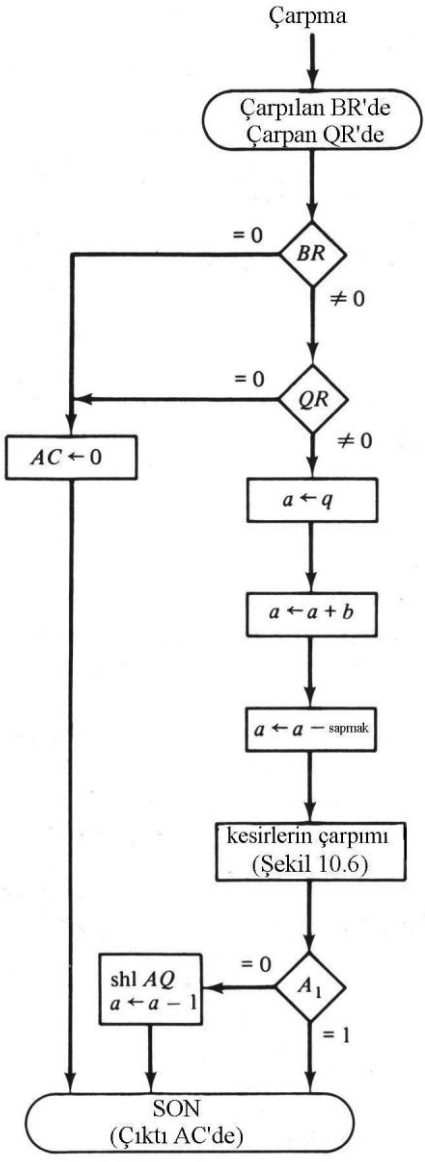
Eğer sayılar çıkarılırsa, sonuç sıfır olabilir veya çökme meydana gelebilir. Eğer kesir 0 ise *AC* içindeki kayan noktalı sayı sıfır yapılır. Aksi halde kesir içinde 1 olan en az bir bit vardır. Eğer kesrin en önemli bitinde sıfır varsa (yani $A_1 = 0$ ise) çökme vardır. Kesir sola kaydırılır ve üs azaltılır. A_1 tekrar denetlenerek gerekiyorsa işlem tekrarlanır ve $A_1 = 1$ olması sağlanır. Kesir normalize edildiğinde işlem biter.

Çarpma

İki kayan noktalı sayıların çarpımında kesirlerin çarpılması ve üslerin eklenmesi gerekir. Kesirlerin çarpımı aynen sabit noktalı sayılarda olduğu gibidir. Farklı olarak sabit noktalı sayılarda çift duyarlıklı olarak sonuç elde edilir. Sabit noktalı sayılarda doğru sonuçları arttırmak için çift duyarlıklı kullanılır. Kayan noktalı sayılarda tek duyarlıklı kesir ve üs genellikle yeterli olmaktadır. Dolayısıyla çarpımın önemli bitlerden oluşan yarım kesri çarpımın kesri olarak alınır, bu yüzden tek duyarlıklıdır.

Çarpma algoritması 4 parçaya ayrılabilir

1. Sıfırların denetimi
2. Üslerin toplanması
3. Kesirlerin çarpımı
4. Sonucun normalize edilmesi



Şekil 10.16 Kayan noktalı sayılarda çarpma

Eğer ayrı toplayıcılar varsa 2 ve 3 adımları aynı anda icra edilebilir. Kayan noktalı sayıların çarpımının akış şeması şekil 10.16 da verilmektedir. İki verinin sıfır olup olmadıkları denetlenir. Eğer verilerden biri sıfırsa çarpım (*AC*) sıfır yapılır. İşlem sona erer. Eğer sayılardan hiçbiri sıfır değilse işlem üslerin toplanmasıyla devam eder.

Çarpanın üssü q da ve toplayıcı a ve b arasındadır. q daki üssün a ya aktarılması, iki üssün toplanması ve toplamın a ya aktarılması gerekir. Her iki üste kısıtlı olduğundan üslerin toplanması kısıtı taşacaktır. Gerçek üs, bu üsten kısıt çıkarılarak bulunur.

Kesirlerin çarpımı sabit noktalı sayılardaki gibi yapılır. Kesirlerin çarpım A ve Q dadır. Çarpım sırasında taşma olmayacağından denetim gerekmez.

Çarpımda çökme oluşabilir. Dolayısıyla A daki en önemli bit denetlenir. Eğer 1 ise çarpım zaten normalizedir. Eğer 0 ise AQ daki kesir sola kaydırılır, üs azaltılır. Sadece 1 kaydırma gerekir. Çünkü çarpan ve çarpılan normalize edilmiş sayılardır. En küçük normalize veri 0.1 dir dolayısıyla çarpım en küçük 0.01 dir. Her ne kadar çarpımın düşük önemli yarısı Q da ise bu kayan noktalı çarpımda kullanılmaz. Sadece AC alınır.

Bölme

Kayan noktalı bölme işlemi kesirlerin bölünmesi ve üslerin çıkarılmasını gerektirir. Kesir bölünmesi, sabit noktalı sayılardaki gibidir.

Bölüm taşması denetimi aynı sabit noktalı sayılar için olduğu gibidir. Fakat kayan noktalı sayılarda bu bir sorun getirmez. Eğer bölünen bölenden büyük veya ona eşitse, bölünenin kesri sağa kaydırılır ve üssü 1 arttırılır. Normalize edilmiş veriler için hiçbir taşma olmaması böylece garantilenir. Yukarıdaki işleme bölünenin hizalanması denir.

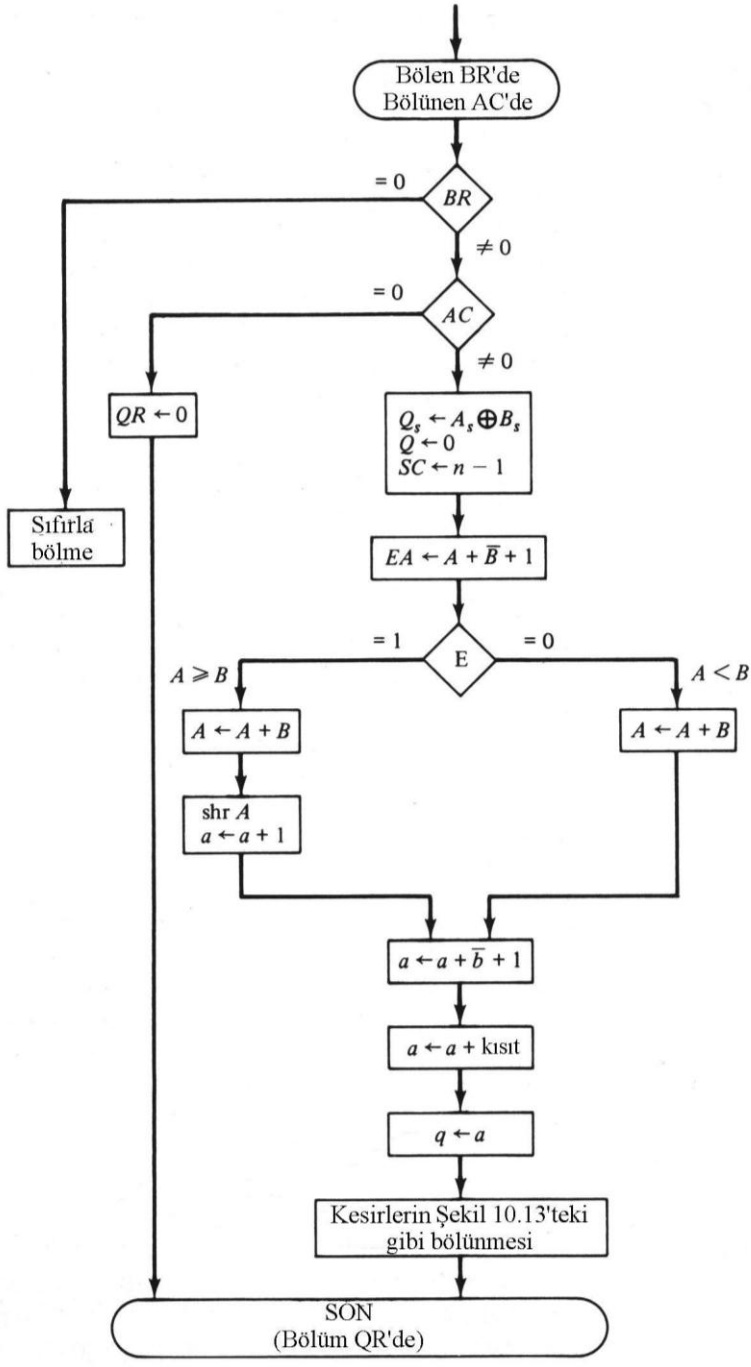
Bölenin Hizalanması

Eğer bölenin hizalanması işi bölme işleminden önce yapılırsa iki kayan noktalı normalize sayının bölümü daima normalize bölüm elde si ile sonuçlanır. Dolayısıyla diğer işlemlerin aksine bölme işleminden sonra elde edilen bölüm normalize ihtiyacı göstermez.

Bölme algoritması 5 parçaya ayrılır.

1. Sıfırların denetimi
2. Yazaçlara ilk değerlerin verilmesi ve işaretlerin bulunması
3. Bölünenin hizalanması
4. Üslerin çıkarılması
5. Kesirlerin bölümü

Kayan noktalı bölme akış şeması şekil 10.17 de verilmiştir. İki verinin 0 denetimi yapılır. Eğer bölen 0 ise sıfırla bölme olduğundan geçersiz bir işlemdir. İşlem hata mesajı ile son bulur. Eğer bölünen 0 ise bölüm QR sıfır yapılır ve işlem sona erer. Buna alternatif bir işlem olarak eğer sayı pozitifse bölümü (yani QR yi) en büyük sayı olarak kabul etmektir.



Şekil 10.17 Kayan noktalı sayıların bölünmesi

Eğer veriler sıfır değilse, bölümün işareti bulunarak Q_S ye yerleştirilir. A_S nin işareti kalanın işaretini belirlemek için olduğu gibi kalır. Q yazacı sıfırlanır ve sıra sayısı (SC) bölüm içindeki bitlerin sayısına eşit yapılır.

Bölümün hizalanması, sabit noktalı sayılardaki bölüm taşması işleminin aynısıdır. Uygun hizalamada bölünen kesrin bölenden küçük olması gerekir. İki kesir çıkarma testi ile karşılaştırılır. E elde si iki sayının göreceli büyüklüklerini verir. Bölünen kesir, bölenin eklenmesiyle eski değerine dönüşür. Eğer $A \geq B$ ise A yı B defa sağa kaydırma ve üssünü arttırmak gerekir. Her iki veri de normalize olduğundan hizalama $A < B$ olmasını sağlar.

Bundan sonra bölenin üssü, bölünenin üssünden çıkarılır. Her iki üs daha önceden kısıtlandırıldığı için, çıkarma işlemi kısıtlı olmayan sonuç verir. Kısıt toplanır ve sonuç q ya aktarılır. Çünkü bölüm QR de oluşacaktır.

Kesirler, sabit noktalı sayılar gibi bölünürler. İşlemden sonra bölümün kesri Q da kalan A dadır. Kayan noktalı bölüm normalizedir ve QR de bulunmaktadır. Kalanın üssü, bölünenin üssü ile aynı olmalıdır. Kalan kesrin ikili noktası A_1 in (n-1) basamak solundadır. Kalan kesri, normalize kesir haline getirilir. Bunun için bölünenin üssünden (n-1) çıkarılır ve A_1 deki bit 1 oluncaya kadar kaydırma ve azaltma işlemine devam edilir. Bu durum akış şemasında gösterilmemiş olup alıştırma için bırakılmıştır.

10.6 Ondalık Aritmetik Birim

Bir bilgisayar kullanıcısı verilerini ondalık sayılarla hazırlar. Sonuçları da ondalık olarak alır. ALU bulunan bir MİB aritmetik mikro işlemleri ikili verilerle yapar. Aritmetik işlemlerin ondalık sayılarla yapılması için girilen ondalık sayıların ikili hale çevrilmesi, işlemden sonra sonuçların tekrar ondalık sisteme çevrilmesi gerekir. Bu yöntem aritmetik işlemlerin çok yoğun fakat G/Ç işlemlerinin az olduğu durumlarda iyi bir yöntemdir. Fakat G/Ç işlemleri çok çok fazla ve aritmetik işlemleri oldukça az ise, işlemlerin ondalık sayılarla yapılması uygun olur. Ondalık sistemle bilgisayarların hesap yapması mümkündür. Ancak bunlar ondalık olarak (BCD biçiminde) saklanmalıdır.

Hesap makineleri daima bir ondalık aritmetik birimle hesap yapar. Çünkü G/Ç işlemleri çok sıktır. Tuşlarla girilen sayıların ikiliye çevrilmesi hem zaman alır, hem de donanımı pahalıdır. Birçok bilgisayar hem ondalık hem ikili bilgiler için aritmetik işlemler yapacak donanma sahiptirler. Kullanıcılar bu durumu programda belirtebilirler.

Ondalık aritmetik birimi sayısal bir birimdir. Ondalık mikro işlemleri yapar. Ondalık sayıları toplar ve çıkarır. Çıkan sayıyı genelde 9 veya 10 a göre tümleyen şeklinde kullanır. Birim gelen sayıları BCD olarak alır ve sonuçları BCD olarak verir. Bir aritmetik birimin bir katı (bir parçası) 9 tane ikili giriş değişkenine ve 5 ikili çıkış değişkenine sahiptir. Çünkü BCD bir sayıyı göstermek için en az 4 bit gerekir. Her bir kat, toplanan sayılar için 4 er tane girişe ve bir giriş elde sine sahip olmalıdır. Çıkış uçları ise toplam için 4 tane, ve çıkış elde si için 1 tane olmalıdır. Kullanılan BCD kodlarına bağlı olarak çok sayıda muhtemel devre mevcut olacaktır·

BCD Toplayıcı

İki ondalık rakamın, daha önceki toplamdan gelen elde yi de düşünerek toplamı, her bir rakam 9 dan büyük olamayacağından, toplam sonucu 9 + 9 + 1 = 19 dan büyük olamaz. (1 giriş elde sinden gelmekte). Şimdi iki BCD kodlanmış rakamı 4 bit toplayıcıda toplayalım. Toplayıcı toplamı ikili olarak yapacak ve sonuç 0 ile 19 arasında olacaktır. Bu ikili sayılar çizelge 10.4 te verilmektedir ve K, Z_8, Z_4, Z_2 ve Z_1 ile gösterilmektedir. K elde değeri, indisler ağırlıkları göstermektedir. Bunlar BCD kodundaki 4

Çizelge 10.4 BCD toplayıcının elde edilmesi

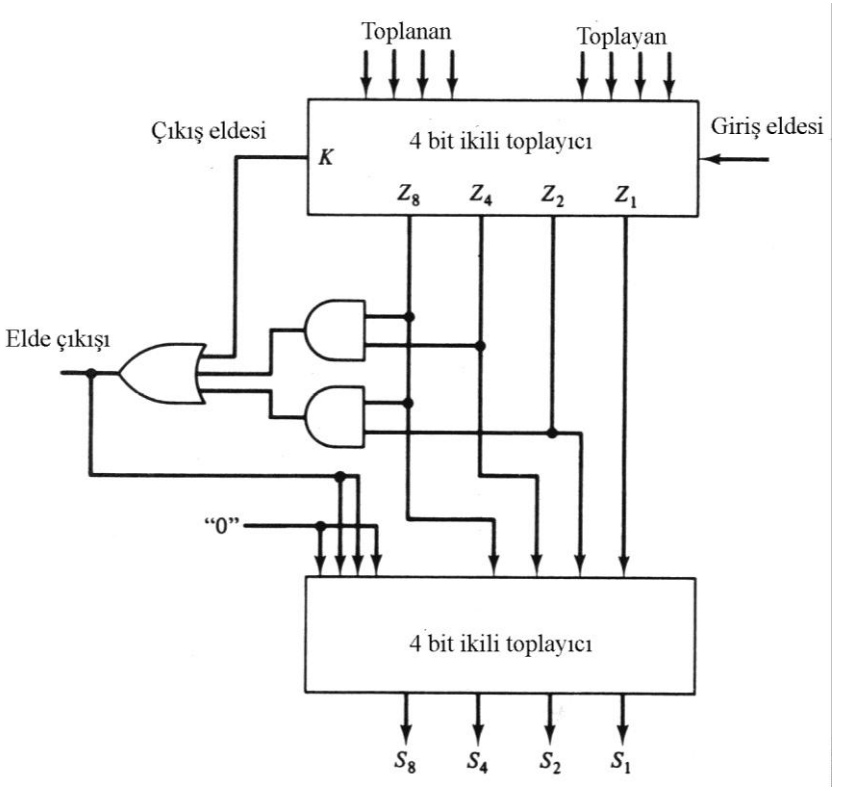
İkili Toplam						BCD toplam				
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	Ondalık
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

bite verilebilir. (Her bir bitin ağırlığı yerine göre değişir). Çizelgedeki birinci sütun toplamın ikili değerini, ikinci sütun bunun BCD karşılığını göstermektedir. Problem birinci sütundaki ikili sayıları ikinci sütundaki BCD lere dönüştürecek kuralı bulmaktır.

Çizelgenin incelenmesiyle ikili toplam 1001 den küçük veya eşitse buna karşılık gelen BCD sayısı ikili sayıya eşittir, bu yüzden dönüşüme gerek yoktur. İkili toplam 1001 den büyük olunca, BCD gösterimi için değişiklik gerekir. İkili 6 nın (0110 ın) ikili sisteme eklenmesiyle doğru bir BCD gösterimi ve çıkış elde si elde edilir.

Ondalık sayıların toplanmasında bir yöntem bir tane 4 bit ikili toplayıcı kullanmak ve her zaman biriminde bir basamak toplama yapmaktır. En düşük basamaktaki rakamların ikili karşılıkları toplanır. Eğer sonuç 1001 den büyük veya eşitse 0110 eklenerek düzeltilir. Sonra diğer basamaklar sırayla toplanır.

Gerekli düzeltmeyi yapan devre, çizelge 10.4 ten elde edilebilir. Buradan görüldüğü gibi BCD toplamının elde si 1 olduğu zaman düzeltme



Şekil 10.18 BCD toplayıcının blok şeması

gerekir. Düzeltme gereken 1010 dan 1111 e kadar olan kısma bakılırsa bunlardan Z_8 in ve bunun yanında Z_4 veya Z_2 ninde 1 olması gerekir. Ayrıca 19 a kadar K nında 1 olduğu 4 sayı vardır. O halde bunun Boole fonksiyonu

$$C = K + Z_8 Z_4 + Z_8 Z_2$$

olmalıdır. $C = 1$ olunca 0110 ikili toplama eklenmelidir.

Bir BCD toplayıcı devresi iki BCD rakamı paralel olarak toplar toplam tek BCD rakamıdır. BCD toplayıcı iç yapısında bir düzeltme mantığına sahip olmalıdır. İkili toplama 0110 ın toplanması için şekil 10.18 deki gibi ikili toplayıcıya ihtiyaç vardır. Giriş elde si ile beraber iki ondalık rakam 4 bitlik ikili toplayıcıda toplanarak ikili toplam elde edilir. Çıkış elde si 0 ise toplama bir şey eklenmez. Eğer çıkış elde si 1 ise ikili toplayıcının çıkışına 0110 ikili değeri eklenir. İkili toplayıcının çıkış elde si ihmal edilebilir. Çünkü çıkış elde si ayrıca belirlenmiştir.

n basamaklı ondalık sayıları toplamak için şekildeki toplayıcılardan n tane gerekir. Bir tanenin çıkış elde si, sonrakinin giriş elde sine bağlanır. Yayılma gecikmelerinin önlenmesi için BCD toplayıcılar, diğerinin elde sini bekleyecek devrelerle donatılmıştır. Ayrıca düzeltme için kullanılan toplayıcının içindeki 4 tam toplayıcının hepsi birden kullanılmaz. Bundan dolayı devre en iyi duruma getirilebilir.

BCD Çıkarma

İki ondalık sayının doğrudan doğruya çıkarılması işlemi için BCD toplayıcıdan farklı olarak bir çıkarıcıya gerek vardır. Bunun yerine 9 veya 10 a göre tümleyen alarak çıkarma yapmak daha kolaydır. Çıkan sayının 9 veya 10 a göre tümleyeni alınıp diğerine toplanır. Fakat BCD sayılar hemen tümleyeni alınan sayılar değildir. Bunun için her bir basamağın 9 dan çıkarılarak tümleyeninin alınması gerekir.

BCD gösteriminde bulunan ondalık basamağın tümleyeni, bitlerin tümleyeninin alınması ve bir düzeltme yapılarak elde edilebilir. İki düzeltme yöntemi vardır. Birinci yöntemde ikili (ondalık on) 1010 her bir tümlenmiş basamağa toplanır ve toplamdan sonraki elde atılır. İkinci yöntemde ikili 0110 (ondalık 6) basamağın tümlenmesinden önce toplanır. Sayısal bir örnek olarak BCD 0111 (ondalık 7) nin 9 a göre tümleyeni, önce her bir bitin tümlenmesiyle 1000 olarak elde edilir. İkili 1010 ın eklenmesiyle ve elde atılarak 0010 (ondalık 2) bulunur. İkinci yöntemde 0110 sayısı 0111 e eklenir. 1101 elde edilir. Her bir bitin tümlenmesiyle sonuç 0010 bulunur. 4 bit ikili sayının tümlenmesi 1111 (ondalık 15 ten) çıkarılmasına eşittir. Ondalık 10 sayısının eklenmesi ile $15 - N + 10 = 9 - N + 16$ olur. 16 sayısı elde de 1 i gösterir ve atılır dolayısı ile $9 - N$ elde

edilmiş olur. Tümleme işleminden önce 6 nın eklenmesi 15 – (*N* + 6) = 9 – *N* sonucunu verir.

BCD basamağın 9 a göre tümleyeni bir devre yardımıyla da elde edilebilir. Bu devre BCD toplayıcıya eklenince sonuçta bir BCD toplayıcı-çıkarıcı olmuş olur. Toplanan veya çıkan sayının bitlerini B_8, B_4, B_2 ve B_1 ile gösterilsin. *M* toplama ve çıkarmayı seçen kip biti olmakta. $M = 0$ ise toplama, $M = 1$ ise çıkarma yapılacak. x_8, x_4, x_2 ve x_1 ikili değişkenleri, 9 a göre tümleyen devresinin çıktıları olsunlar. Devrenin doğruluk çizelgesinden (problem 10.30) B_1 in daima tümlenmesi gerektiği, B_2 nin 9 a göre tümleyende daima aynı kaldığı, B_2 ÖZEL-VEYA $B_4 = 1$ ise x_4 ün 1 olduğu ve $B_8B_4B_2 = 000$ ise $x_8 = 1$ olduğu görülür. 9 a göre tümleyen devresinin Boole fonksiyonları

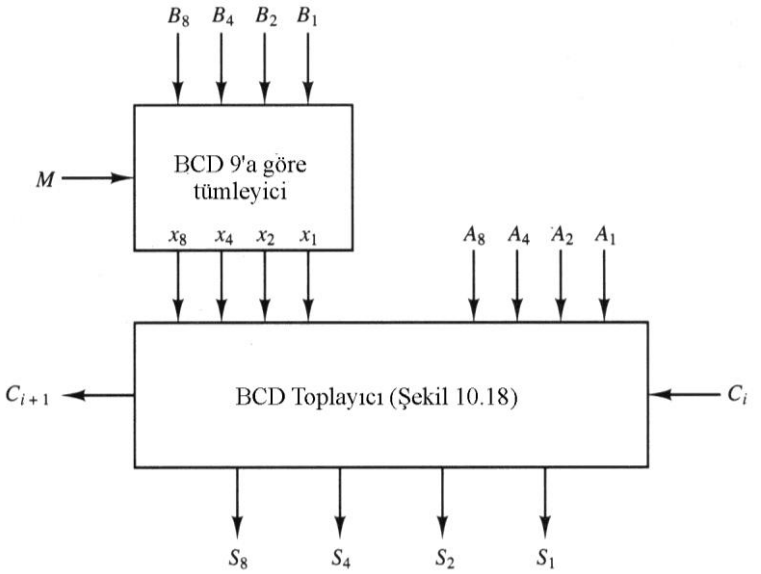
$$x_1 = B_1M' + B'_1 M$$

$$x_2 = B_2$$

$$x_4 = B_4M' + (B'_4 B_2 + B_4B'_2) M$$

$$x_8 = B_8M' + B'_8 B'_4 B'_2 M$$

Burada $M = 0$ iken $x = B$, $M = 1$ olunca x *B* nin 9 a göre tümleyenini verir.



Şekil 10.19 Ondalık aritmetik işlemcinin bir aşaması

İki BCD sayının birer basamaklarını toplayan ve çıkaran ondalık aritmetik birimi şekil 10.19 da gösterilmektedir. Devre BCD toplayıcı ile 9 a göre tümleyen devrelerini içerir. Kip denetimi M, birimin işlemini denetler. $M = 0$ iken S, A ve B nin toplamı, $M = 1$ iken S, A ile B nin 9 a göre tümleyeninin toplamıdır. n ondalık basamaklı sayılar için bu devrelerden n tane gerekir. Çıkış elde si C_{i+1} bir sonraki yani bir yukarı mertebeden birimin giriş elde sine bağlanır. İki ondalık sayıyı çıkarmak için en iyi yol $M = 1$ yapmak ve birinci basamağın giriş elde sine 1 yollamaktır. Çıkışlar A ile B nin 10 a göre tümleyeninin toplamını verir. Böylece son elde çıkışını da dikkate almaya gerek olmadan çıkarma işlemi tamamlanmış olur.

10.7 Ondalık Aritmetik İşlemler

Ondalık sayılar için aritmetik işlemlerin algoritmaları, çok küçük bazı değişiklikler haricinde ikili sayılardakinin aynıdır. Gerçekten çarpma ve bölme algoritmalarında ufak değişikliklerle akış şemaları her iki tip veri için kullanılabilir. Ancak mikro işlem sembollerini uygun şekilde değerlendirmek şarttır. Ondalık sayılar bilgisayar yazaçlarında 4 bitlik gruplar halinde bulunurlar. Her 4 bitlik grup bir ondalık basamağı gösterir. Ondalık mikro işlemler uygulanırken bu 4 bitlik grup bir birim olarak işleme girer.

Ondalık ve ikili aritmetik mikro işlemler için aynı semboller kullanılacak. Fakat farklı anlamlar taşıyacaklar. Çizelge 10.5 te görüldüğü gibi yazaç harf sembolü üzerindeki çizgi 9 a göre tümleyeni gösteriyor. 9 a göre tümleyene 1 eklenince 10 a göre tümleyen bulunur. Dolayısıyla ondalık sayılarda $A \leftarrow A + \overline{B} + 1$ sembolik yazılımı A ya B nin 10 a göre tümleyeninin eklendiğini gösterir. Eğer sistemde hem 9 hem de 1 e göre tümleyen kullanılıyorsa bu durum kafa karıştırır. Böyle durumlar için 9 a göre tümleyende farklı sembol kullanmak yerinde olur. Eğer tek tip veri varsa bu sembolik gösterim kullanılabilir.

Çizelge 10.5 Ondalık aritmetik mikro işlem sembolleri

Sembolik gösterim	Tanımlama
$A \leftarrow A + B$	Ondalık sayıları topla ve toplamı A ya aktar
$\overline{B}$	B nin 9 a göre tümleyeni
$A \leftarrow A + \overline{B} + 1$	B nin 10 a göre tümleyenini alıp A ile topla sonucu A ya aktar
$Q_L \leftarrow Q_L + 1$	Q_L sayısının değerini 1 arttır
dshr A	A yazacını ondalık sağa kaydır
dshl A	A yazacını ondalık sola kaydır

Yazaçların arttırılması ve azaltılması ikili ve ondalık sayılar için aynıdır. Sadece yazaçların alabileceği durumlar farklıdır. İkili bir sayıcı 16 durum gösterir. 0000 dan 1111 e kadar. Ondalık sayı son rakam 9 olacağından 10 durum gösterir ve 0000 dan 1001 e kadar. Artmada 0000 dan 1001 e gider ve azalmada 0000 a geri döner. Azalmada ikili sayıcı 1111 den 0000 a gider.

Ondalık bir sağa kaydırma 4 bit üzerine olup en başa (en sola) konan bir *d* harfiyle belirlenir. Sayısal bir örnek olarak bir *A* yazacı ondalık 7860 sayısını BCD gösteriminde 16 yaz-bozda

0111 1000 0110 0000

şeklinde tutar. *dshr A* mikro işlemi ondalık sayılı 1 basamak sağa kaydırır ve yazacın içeriği

0000 0111 1000 0110

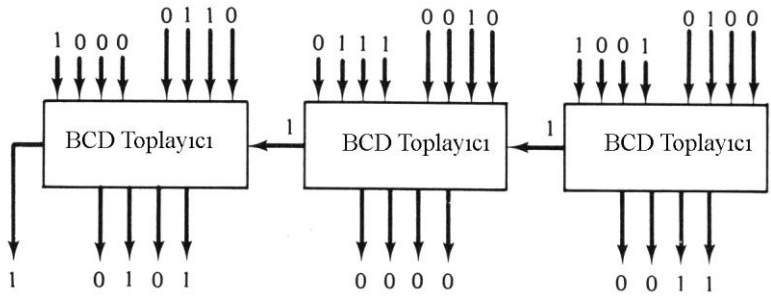
olur.

Toplama ve Çıkarma

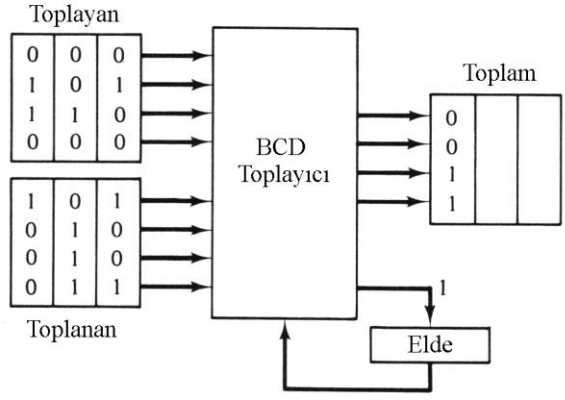
İşaretli sayıların toplama ve çıkarma algoritmaları ondalık işaretli sayılara da uygulanır. Mikro işlem sembolleri ondalık sayılara uygun biçimde değerlendirilir. Aynı biçimde 2 ye göre tümleyen algoritmaları 10 a göre tümleyene uygulanabilir. İkili veri bir ikili toplayıcı ve tümleyici kullanır. Ondalık veri ondalık bir toplayıcı ve 9 a göre tümleyeni hesaplayan bir tümleyici kullanmalıdır (Şekil 10.19).

Ondalık veri 3 farklı şekilde toplanabilir. (Şekil 10.20). Paralel yöntem sayıdaki basamak sayısı kadar BCD toplayıcı kullanır. Toplam paralel olarak oluşur ve tek bir mikro işlem yeterlidir. Basamak seri bit paralel kipinde basamaklar tek bir BCD toplayıcıya gönderilir. Fakat bu rakamın bitleri paralel olarak toplayıcıya gönderilir. Toplam, ondalık sayıların her bir zaman biriminde bir basamak olarak toplayıcıdan alınmasıyla elde edilir. *k* ondalık basamak için bu yöntem *k* mikro işlem gerektirir. Tamamen seri toplayıcıda her saat vuruşunda bir bit tam toplayıcıya gönderilir. 4 bit sonunda oluşan toplam düzeltilerek doğru bir BCD basamağı elde edilir. Bu düzeltme 10.6 da verilmiştir. Eğer toplam 1010 dan büyükse buna 0110 eklenir ve taşma sonraki basamağa aktarılır.

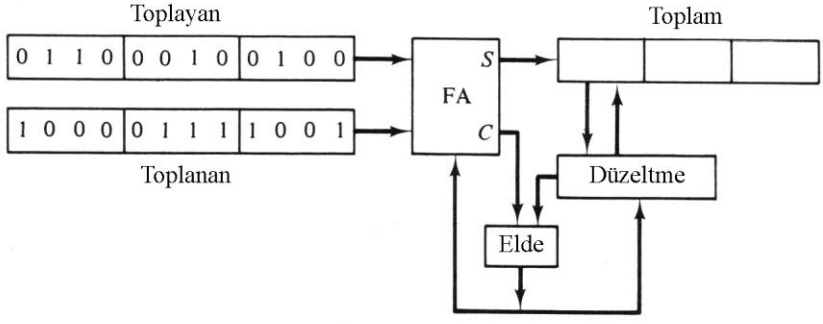
Paralel yöntem hızlı fakat çok sayıda toplayıcı ister. Basamak seri, bit paralel yöntemde tek bir BCD toplayıcı gerektirir. Bütün basamaklar bunu kullanır. Paralel yöntemden yavaştır. Tümüyle seri yöntem en az donanım gerektirir, çok yavaştır.



(a) Paralel ondalık toplayıcı

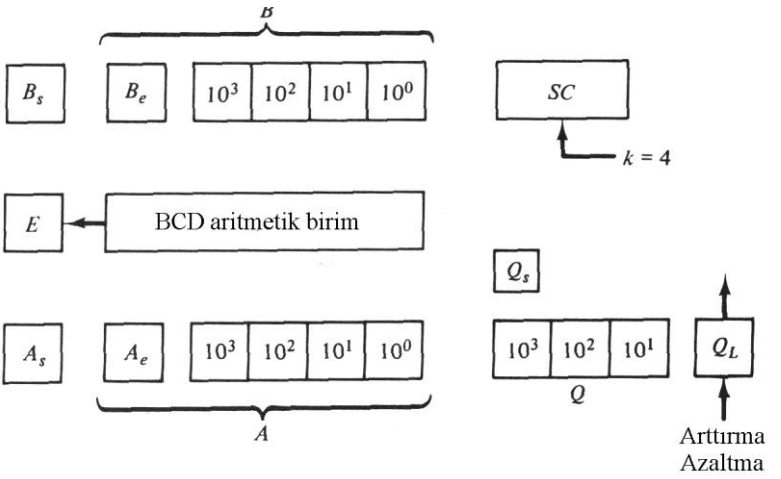


(b) Basamak seri, bit paralel ondalık toplayıcı



(c) Tümü seri ondalık toplayıcı

Şekil 10.20 Ondalık sayıların toplanması için üç yol



Şekil 10.21 Ondalık aritmetik çarpma ve bölme için yazaçlar

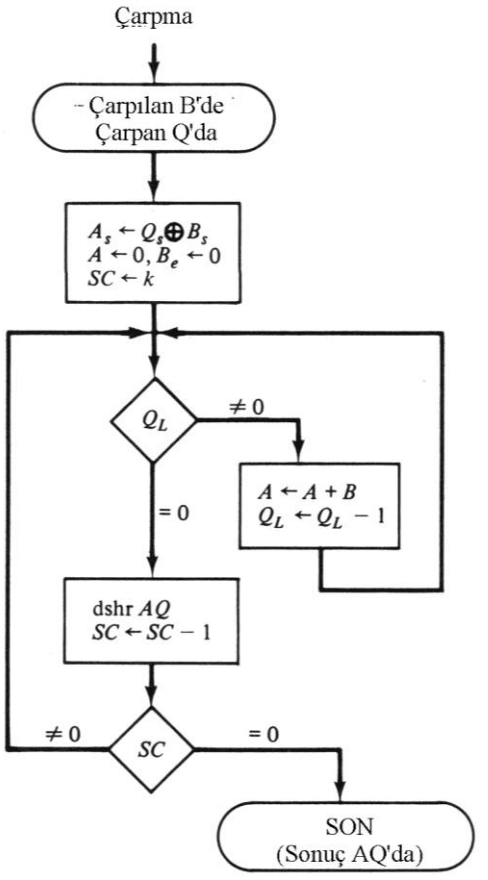
Çarpma

Sabit noktalı ondalık sayıların çarpımı, ikili sayılarınkine benzer. Fakat kısmi çarpımların oluşumu farklıdır. Ondalık bir çarpanın 0 ve 1 olmak üzere 2 basamağı vardır. İkili durumda çarpılan kısmi çarpıma eklenir. Ondalık durumda çarpılan, çarpanın ilgili basamağı ile çarpılmalı ve sonuç toplanmalıdır. Bu durum çarpılanın, çarpan basamak defa toplanmasıyla elde edilebilir.

Ondalık çarpım için yazaç durumu şekil 10.21 de gösterilmektedir. Burada 4 basamaklı sayılar göze alınmıştır. Her basamak 4 bit içerdiğinden sayı 16 bittir. *A*, *B* ve *Q* olmak üzere 3 yazaç var. Her birinin bir işaret yaz-bozu bulunmakta. *A* ve *B* yazaçlarının 1 er fazla biti vardır. A_e ve B_e. Bunlar sayesinde bu iki yazaç birer basamak fazladırlar. BCD aritmetik birimi 5 basamağı paralel olarak toplar ve sonucu 5 basamaklı *A* yazacına son elde yi de *E* yaz-bozuna yerleştirir. A_e basamağının amacı bir taşma olduğunda bunu tutmak içindir. Taşma çarpılanı kısmi toplama eklerken çıkabilir. B_e nin görevi ise bölenin kısmi kalandan çıkarılması sırasında 9 un tümleyenini oluşturmaktır. *Q* yazacındaki en önemsiz basamak Q_L olsun. Bu basamak arttırılabilir veya azaltılabilir.

Bellekten gelen bir veri 17 bit içermektedir. 1 bit (işaret) B_S ye aktarılır. Verinin sayı kısmı ise *B* nin 16 bitine yerleşir. $B_e$ve A_e önceden temizlenir. İşlemin sonucu da 17 bit uzunluğundadır ve *A* yazacının A_e parçasını kullanmaz.

Ondalık çarpma algoritması şekil 10.22 de gösterilmiştir. İlk olarak *A* yazacı ve B_e sıfırlanır. Sıra sayıcı ise çarpıcı içindeki basamakların sayısı olan *k* ya eşitlenir. Çarpanın düşük mertebeli basamağı Q_L denetlenir.



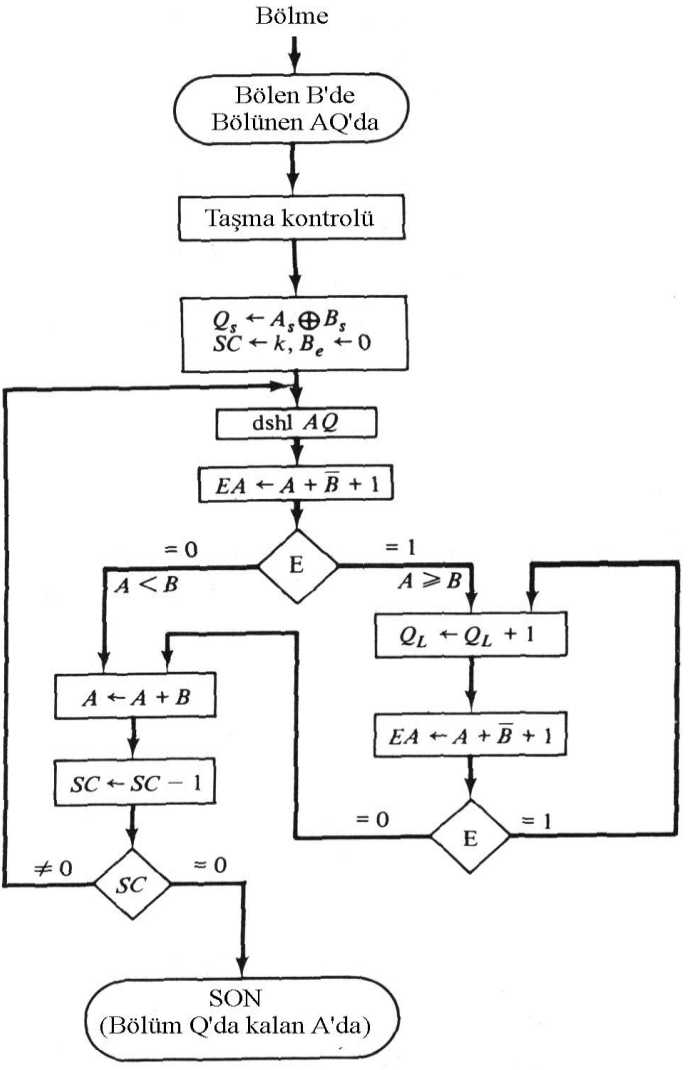
Şekil 10.22 Ondalık çarpma için akış şeması.

Eğer 0 değilse B deki çarpan kısmi çarpıma bir defa eklenir. Q_L azaltılır, tekrar denetlenir ve sıfır değilse işlem tekrarlanır. Böylece B deki çarpılan kısmi toplama çarpan basamağı kadar defa eklenmiş olur. Herhangi bir taşma A_e içinde kalır ve değeri 0 ile 9 arasında olabilir.

Sonra kısmi çarpım ve çarpan bir defa sağa kaydırılır. A_e sıfır yapılır. Çarpanın önemsiz basamağı Q_L ye yazılır. İşlem tekrarlanır ve AQ içinde 2 katı uzunlukta çarpım elde edilir.

Bölme

Ondalık bölme ikili bölmeye benzer. Sadece bölüm basamakları 0 dan 9 a kadar 10 değerden birini alır. Bölümün yeniden oluşturulması ile bölme yönteminde bölen, bölünenden veya kısmi kalandan, negatif bir kalan



Şekil 10.23 Ondalık bölme için akış şeması

oluşana dek çıkarılır. Sonra bu kalan yeniden düzenlenir. Bölümdeki sayı son çıkarma hariç bölenin kaç defa çıkarıldığını gösterir.

Ondalık bölme algoritması şekil 10.23 te gösterilmiştir. İkili sayılarla bölme algoritmasının benzeridir. Bölen bitlerinin oluşumu farklıdır. Bölünen veya kısmi kalan sola kaydırılır. En önemli basamak A_e dedir. Bölenin daha sonra 10 a göre tümleyeni alınıp bölünene eklenmesiyle, bölünenden bölen çıkarılır. B_e baştan temizlendiğinden bunun tümleyeni 9 dur. E deki elde A ve B nin durumlarını bildirir. Eğer $E = 0$ ise $A < B$ dir

ve bölen kısmi kalana eklenir ve Q_L 0 olarak kalır. Eğer $E = 1$ ise $A \geq B$ dir Q_L deki basamak 1 arttırılır ve bölen tekrar çıkarılır. Bu işlem, çıkarma işlemi negatif sonuç verene kadar sürdürülür (yani $E = 0$ oluncaya kadar). $E = 0$ olunca bölüm basamağı arttırılmaz. Fakat bölen toplanarak kalanın pozitif olması sağlanır. Bu yolla bölüm basamağı bölenin, kısmi kalanda kaç defa bulunduğunu gösterir.

Kısmi kalan ve bölüm bitleri bir defa sola kaydırılır ve işlem k defa tekrarlanır. Böylece k bölüm basamağı elde edilir. Kalan A yazacında, bölüm Q dadır. E nin değeri ihmal edilir.

Kayan Noktalı İşlemler

Ondalık kayan noktalı sayılarla işlemler ikili işlemlerdeki yolları aynen izler. 10.5 teki algoritmalar ondalık sayılar için uygulanabilir. Dikkat edilmesi gereken mikro işlem sembollerinin doğru olarak alınmasıdır. Kesirlerin çarpımları ve bölümleri sabit noktalı ondalık sayılardaki yöntemlerle yapılır.

SORULAR

10-1. Şekil 10.1 deki tümleyiciye, eğer A + $\overline{B}$ +1 yerine B + $\overline{A}$ (B artı A nın 1e göre tümleyeni) icra edilirse gerek duyulmaz. İşaretli büyüklük gösterimindeki iki sabit noktalı sayının toplanması ve çıkarılması için akış şeması gösterimi kullanarak bir algoritma üretin. Çıkarma işlemini iki mikro işlemle, $A \leftarrow \overline{A}$ ve $EA \leftarrow A + B$ ile gerçekleştirin.

10-2. Şekil 10.2 deki akış şemasının parçalarına birer numara verin. Sonra aşağıdaki işaretli büyüklük olarak yazılmış sayılar arasında gerçekleşen işlemlerin her biri için yolu belirtin. Her bir durumda *AVF* nin değerini söyleyin. Aşağıdaki sayılarda en soldaki bitler işaret bitidir.

a. 0 101101 + 0 011111
b. 1 011111 + 1 101101
c. 0 101101 – 0 011111
d. 0 101101 – 0 101101
e. 1 011111 – 0 101101

10-3. Aşağıdaki işlemleri ikili düzende, negatif sayılar 2 ye göre tümleyen biçiminde olmak üzere yapın. Her sayıyı işaretiyle beraber 7 bit içine yerleştirin. Her durumda işaret bitinin giriş ve çıkış elde lerini izleyerek taşma olup olmadığını söyleyin.

a. (+35) + (+40)
b. (–35) + (-40)
c. (–35) – (+40)

10-4. İkili düzendeki sayıları 2 ye göre tümleyen biçiminde düşünelim. Her bir sayıda n bit var. Bir tanesi işaret biti $k = n - 1$ tanesi büyüklük. Negatif bir sayı $-X$, $2^k + (2^k - X)$ ile gösterilir. İlk 2^k işaret bitini gösteriyor. $(2^k - X)$ ise X in tümleyenidir. Pozitif bir sayı $0 + X$ ile gösterilir. 0 işaret bitini gösterir. X ise k bit büyüklüğünü gösterir. Bu genelleştirilmiş sembolleri kullanarak, $(\pm X) + (\pm Y)$ toplamının, işaret bitleri dahil sayıların toplanması ve işaret bitlerinin çıkış elde sinin atılması ile elde edilebileceğini ispatlayın. Diğer bir deyişle 2 ye göre tümleyen biçiminde verilmiş iki sayının toplamını gösteren algoritmayı ispatlayın.

10-5. Toplanan sayıların işaretleri ile toplamın işareti karşılaştırılarak, varsa taşmayı bulan bir donanım işlemi tasarlayın. Sayılar 2 ye göre tümleyen biçimindeler.

10-6. **a.** $(-9) + (-6) = -15$ işlemi ikili düzende ve 1 e göre tümleyen biçiminde yazılmış olarak gerçekleştirilebilir. Her bir sayı, işaret biti dahil 5 bit yer kaplasın. Taşmayı anlamak için son iki elde yi karşılaştırmanın işe yaramayacağını gösterin.

b. İşaretli 1 e göre tümleyen biçimindeki sayılar kullanıldığında taşmayı bulmak için değiştirilmiş bir yöntem önerin.

10-7. İkili gösterimde iki sayının toplama ve çıkarılması için akış şeması düzeninde bir algoritma türetin. Negatif sayılar 1 e göre tümleyen biçiminde olsun.

10-8. r tabanlı, n rakamlı iki sayının çarpımının $2n$ uzunluğundan fazla olamayacağını ispatlayın. Ayrıca bu durumda çarpma işlemi sonunda taşma olmayacağını gösterin.

10-9. İkili düzende verilmiş iki sayı çarpılan = 11111 ve çarpan = 10101 olsun. Bunların çarpma işlemi sırasında E, A, Q ve SC yazaçlarının (çizelge 10.2 de olduğu gibi) içeriklerini gösterin.

10-10. Çizelge 10.2 deki E, A, Q ve SC yazaçlarının, aşağıdaki bölme işlemleri sırasındaki içeriklerini söyleyin. (a) 10100011 sayısı 1011 ile (b) 00001111 sayısı 0011 ile bölünüyor.(Bölünen 8 bittir.)

10-11. $A + \overline{B} + 1$ işlemi sonunda elde edilen değere B eklersek A yi elde ederiz. Çıkış elde si ne olacaktır?

10-12. Bir bölme işleminden sonra neden kalanla bölünenin işareti aynı olur?

10-13. İki tane 4 bit sayıyı çarpan bir dizi çarpıcı tasarlayın. Tasarımda VE kapıları ve ikili toplayıcılar kullanın.

10-14. Booth algoritması kullanarak (çizelge 10.3 teki gibi) aşağıdaki çarpımları ikili düzende sayılarla adım adım yapın. İşaretli sayılar 5 bitlik yazaçlarda bulunuyor. Çarpılan her iki durumda da +15 tir.

a. $(+15) \times (+13)$

b. $(+15) \times (-13)$

10-15. Sabit noktalı ikili gösterimde bölme işlemi için iadesiz yöntemle akış şeması biçiminde bir algoritma türetin.

10-16. Sabit noktalı ikili gösterimde bir sayının karekökünü bulan bir algoritma türetin.

10-17. İkili düzende kayan noktalı bir sayının üssü kısıtlı olup 7 bittir. Kısıt için kullanılan 64 tür.

- **a.** –64 ten +63 e kadar tüm kısıtlı üsleri yazın.
- **b.** İki üssün göreceli büyüklüklerinin karşılaştırılması için, 7 bit büyüklük karşılaştırıcının kullanılabileceğini gösterin.
- **c.** İki kısıtlı üssün toplanmasından sonra toplamında kısıtlı olması için 64 çıkarmak gerektiğini gösterin. 64 ün 2 ye göre tümleyenini ekleyerek nasıl çıkartırsınız?
- **d.** İki kısıtlı üssün çıkarılmasından sonra kısıtlı üs farkını elde etmek için 64 eklemek gerektiğini gösterin.

10-18. İki tane ikili gösterimdeki işaretli sayıyı birbiriyle karşılaştırmak için akış şeması biçiminde algoritma türetin. Negatif sayılar 2 ye göre tümleyen biçimindedir.

- **a.** Karşılaştırma, çıkarma yöntemiyle ve çıkarılan sayının 2 ye göre tümleyen alınarak.
- **b.** Soldan sağa doğru bitleri çift çift tarayarak ve karşılaştırarak gerçekleşsin.

10-19. Soru 10.18 i, ikili düzende işaretli büyüklük olarak verilmiş sayılar için tekrarlayın.

10-20. İkili düzende yazılmış kayan noktalı bir sayının kesir parçasındaki bitlerin sayısı n olsun. Toplama veya çıkarma için hizalandığında üs farkı $n - 1$ den büyük olabilir. Böyle bir durumda üssü küçük olan kesir bütünüyle yazaç dışı kalabilir. Şekil 10.15 teki hizalamayı, kaydırmaların sayısını tutan bir *SC* sayacı ekleyerek değiştirin. Eğer kaydırma sayısı $n - 1$ den büyükse, büyük sayı sonucu göstersin.

10-21. Toplama veya çıkarma sırasında kayan noktalı sayıların kesirlerinin hizalanması işlemi aşağıdaki gibi yapılır. Küçük üs büyükten çıkarılır. Küçük üssün kesri üslerin farkı kadar sağa kaydırılır. Toplamın (veya farkın) üssü büyük olan üsse eşittir. Büyüklük karşılaştırıcı kullanmadan, üslerin kısıtlı olduğu ve sadece *AC* nin kaydırılabildiği varsayılarak, kesirleri hizalamak ve *AC* deki büyük üssü yerleştirmek için akış şeması şeklinde bir algoritma türetin.

10-22. Çarpma işleminden sonra taşma olmayacağını gösterin.

10-23. İki normalize edilmiş kayan noktalı sayının bölünmesi öncesinde bölünen hizalaması yapılmazsa sonucun daima normalize edilmiş biçimde olacağını gösterin.

10-24. Şekil 10.17 deki akış şemasını, normalize edilmiş kayan noktalı sayının kalanı *AC* de olacak şekilde genişletin.

10-25. Bölüm 10.5 te görülen kayan noktalı sayılara ait aritmetik işlem algoritmaları, üs taşması veya çökme ihtimallerini ihmal eder.
- **a.** Üç akış şemasını da inceleyerek nerede üs taşması olabileceğini bulun.
- **b.** a şıkkındakini çökme için uygulayın. Üs çökmesi, yazaca yazılabilecek en küçük sayıdan daha küçük bir üs olursa meydana gelir.
- **c.** Taşma ve çökmenin donanım tarafından nasıl fark edilebileceğini gösterin.

10-26. Kayan noktalı sayıların kesir kısımlarının tamsayı gösterimini düşünelim. Bu sayıların çarpma ve bölmelerinde bazı ölçekleme sorunlarıyla karşılaşırız. Kesrin büyüklük kısmındaki bit sayısı $(n - 1)$ olsun. Tamsayı gösterimi için:
- **a.** Eğer tek duyarlıklı bir çarpım kullanılırsa, *AC* deki çarpım üssüne $(n - 1)$ eklenmelidir.
- **b.** Tek duyarlıklı kesir bölünen olarak kullanılırsa , 0 silindiğinde bölünenin üssünden $(n - 1)$ çıkarılmalıdır.

10-27. İki işaretli büyüklük gösterimindeki ondalık sayının toplanma ve çıkarılmasında donanım kullanılmalıdır. Taşma nasıl fark edilir?

10-28. 673 – 356 işlemi 673 sayısına 356 nın 10 a göre tümleyeni eklenerek ve son elde atılarak yapılabilir. 3 kademeli bir aritmetik birim çizerek bunun nasıl yapılabileceğini gösterin. Birimin giriş ve çıkış bitlerini yazın.

10-29. Şekil 10.1 deki ikili toplayıcının düşük 4 bitinin, bir tane tam, iki tane yarım toplayıcı ile yer değiştirebileceğini gösterin.

10-30. Birleşik devre tasarımı kullanarak, şekil 10.19 daki BCD 9 a göre tümleyenli Boole fonksiyonlarını türetin. Mantık şemasını çizin.

10-31. Çizelge 3.6 daki excess–3 kodunun iki ondalık rakamı için bir toplayıcı tasarlamak gerekiyor. İki rakamın toplanmasından sonra 4 bit ikili toplayıcıda yapılacak düzenlemeler aşağıda verilmiştir.
- **a.** Çıkış elde si düzeltilmemiş elde ye eşittir.
- **b.** Eğer çıkış elde si 1 ise 0011 ekleyin
- **c.** Çıkış elde si 0 ise 1101 ekleyin ve toplamın elde sini ihmal edin. Excess–3 toplayıcısının 7 tane tam toplayıcı ve 2 tersleyiciden oluşacağını gösterin.

10-32. Ondalık rakamlar excess–3 kodu ile (çizelge 3.6) gösterildiğinde, 9 a göre tümleyenini bulan bir devre çizin. Bir kip denetim girişi rakamın tümlenmiş olup olmadığını belirtir. Bu kodu BCD yerine kullanmanın faydası nedir?

10-33. Negatif sayılar 10 a göre tümleyeni gösteriminde olmak üzere iki ondalık sayının toplama ve çıkarılmasını yapan bir donanım bulun. Taşma nasıl fark edilir? Bir akış şeması gösterimi ile bir algoritma üretin ve deneyerek algoritmanın doğru olduğunu bulun.

10-34. Aşağıdaki ondalık çarpımlar sırasında (şekil 10.22) *A*, *B*, *Q* ve *SC* yazaçlarının içeriğini söyleyin. Yazaçları 3 rakamlı olarak düşünün. İkinci sayıyı çarpan olarak alın. (a) 470×152 ve (b) 999×199

10-35. Ondalık bölme işlem sırasında *A*, *E*, *Q* ve *SC* yazaçlarının (şekil 10.22) içeriklerini söyleyin. Yazaçları 2 rakamlı varsayın.

10-36. Şekil 10.21 deki alt yazaç A_e nin (a) Şekil 10.22 deki ondalık çarpmanın sonunda, (b) Şekil 10.23 deki ondalık bölmenin sonunda, sıfır olacağını gösterin.

10-37. Bölüm 10.5 teki kayan noktalı aritmetik algoritmaları ikiliden ondalığa çevirin. Bir çizelge yaparak her bir mikro işlem sembolünün nasıl açıklanacağını yazın.

KAYNAKLAR

1. Blaauw, G., *Digital System Implementation*. Englewood Cliffs, NJ: Prentice Hall, 1976.

2. Cavanagh, J. J. F., *Digital Computer Arithmetic*. New York: McGraw-Hill, 1984.

3. Hamacher, V. C., Z. G. Vranesic, and S. G. Zaky, *Computer Organization*, 3rd ed. New York: McGraw-Hill, 1990.

4. Hays, J. F., *Computer Architecture and Organization*. 2nd ed. New York: McGraw-Hill, 1988.

5. Hill, F. J., and G. R. Peterson, *Digital Systems: Hardware Organization and Desing*, 3rd ed. New York: John Wiley, 1987.

6. Hwang, K., *Computer Arithmetic*. New York: John Wiley, 1979.

7. Kulisch, V. W., and W. L. Miranker, *Computer Arithmetic in Theory and Practice*. New York: Academic Press, 1980.

8. Schmid, H., *Decimal Arithmetic*. New York: John Wiley , 1979.

BÖLÜM 11

Giriş-Çıkış Organizasyonu

BU BÖLÜMDE

11.1 Çevre Aletleri
11.2 Giriş / Çıkış Ara Yüzeyi
11.3 Eş zamanlı olmayan Veri Aktarımı
11.4 Aktarım Kipleri
11.5 Kesme İstemi Önceliği
11.6 Doğrudan Bellek Erişimi
11.7 Giriş / Çıkış İşlemcisi (GÇİ)
11.8 Seri Haberleşme

11.1 Çevre Aletleri

G/Ç; Bir bilgisayarın G/Ç alt sistemi, merkezi sistemle dış çevre arasında etkin bir haberleşme sağlar. Programlar ve veriler işlenmek için bilgisayar belleğine girilmeli ve sonuçlar kaydedilmeli veya kullanıcıya görüntülenmelidir. Bir bilgisayar dış bir kaynaktan bilgi almaz ve sonuçları anlamlı bir biçimde aktarmazsa işe yaramaz.

Bir bilgisayara bilgi girmenin en bilinen yolu daktilo benzeri bir klavye kullanmaktır. Böylece alfa sayısal bilgiler doğrudan doğruya girilebilir. Bir tuşa basıldığında klavye bilgisayara ikilik kodlanmış bir karakter gönderir. Bu yolla bilgi girmede hız, klavyeyi kullananın hızıdır. Oysa MİB çok hızlı çalışan bir alettir. Eğer bilgisayara yavaş bir klavye yardımıyla girilirse, MİB klavyeyi bekleyerek büyük zaman harcar. Bilgisayarın etkin bir biçimde kullanılabilmesi için program ve verilerin önceden hazırlanmış ve manyetik teyp ya da sabit disk ortamına aktarılmış olması gerekir. Bundan sonra veriler bilgisayar belleğine hızlı bir biçimde aktarılır. Programların sonuçları da aynı şekilde sabit diske aktarılabilir. Sonrada bir yazıcıdan dökülebilir.

Çevre Birimler

Bilgisayarın doğrudan denetimi altındaki aletlere çevrim içi bağlı denir. Bu aletler belleğe bilgi aktarabilir veya bellekten bilgi okuyabilirler. Bunlara komutlar MİB tarafından verilir ve bilgisayarın parçası sayılırlar. Bilgisayara bağlı G/Ç aletlerine ayrıca çevre aletleri de denir. Çevre aletleri elektro manyetik veya elektro mekanik olabilir. Çevre aletleri arasında klavye, ekran ve yazıcı sayılabilir. Sabit disk ve teyplerde yardımcı depolama birimi olarak çevre aleti sayılırlar.

Ekran ve Klavye

Video ekranları en çok kullanılan çevre aletlerindendir. Klavyeler bilgi girişi, ekranlar çıkış birimi olarak kullanılır. Birçok çeşit ekran vardır. En çok kullanılanları CRT katot ışınları tüpüdür. CRT içinde elektronik bir tabanca bulunur. Bu tabanca tüpün ön kısmındaki floresan ekrana katot ışınları gönderir. Işınlar yatay veya dikey olarak denetlenebilir. Ekran üzerinde bir şekil oluşturmak için tüp içindeki bir ızgara değişken bir gerilim alır ve bununla ekranın istenen noktalarında parlamasını sağlar. Yatay ve düşey sinyaller ışınları saptırır, tüpü süpürtür ve ekran üzerinde şekiller oluşmasını sağlar. Görüntü ekranın karakteristik bir olayı imleçtir. Bir sonra yazılacak karakterin yerini belirler. İmleç, ekran üzerinde herhangi bir yere gönderilebilir. Bu tek bir karakter, bir kelimenin başlangıcı veya herhangi bir satır olabilir. Edit tuşları imlecin bulunduğu yere göre yazar veya silerler. Görüntü ekranı, tek karakter kipinde çalışır yani ekrana karakter karakter yazar. Klavyede bir tuşa basılınca, bu tuşa ait karakter bilgisayara bildirildiği gibi ekrana da yazılır. Blok kipinde yazılan metin, uç birim içindeki bellekte tutulur, sonra bilgisayara veri bloğu olarak aktarım yapılır.

Yazıcı

Yazıcılar kağıt üzerinde kalıcı kayıt oluştururlar. 3 temel tip yazıcı vardır. Papatya çarklı, iğneli ve lazer. Papatya çarklı yazıcılarda bir tekerlek üzerinde karakterler vardır. İstenen karakteri basmak için tekerlek döner ve elektro mıknatısla şeride ve kağıda vurur. İğneli yazıcıda noktaları vuracak iğneler vardır. Bunlar 24 tane veya az olabilir. Genelde her karakter için 7×5 bir matris olacak şekilde basarlar. 80 veya 132 karakter basabilirler. 80 lik bir satır için $5 \times 80 = 400$ nokta basarlar. Lazer yazıcı fotokopi makinesinin yarısı ile drama yazan bir lazer yazıcıdan oluşur. Drama bir kere yazıldıktan sonra kalanı sadece fotokopi makinesi gibidir.

Manyetik teyp, Sabit disk

Teyp genelde arşiv amacıyla kullanılır. Erişim sıralıdır. Çok ucuz ve yavaş bir yöntemdir. Fakat ucuz ve kapasitesi çok büyüktür. Manyetik bir ortam kaplı 1 veya birkaç diskten oluşur. Bir veya fazla okuyucu kafası vardır. Bu iki konu 12.1 de anlatılacak

Bunun yanı sıra bilgisayar sistemine bağlanabilen bazı aletler, çiziciler, manyetik veya optik karakter okuyucular, analog-sayısal çeviriciler vs. dir. Bütün veriler insanlardan gelir. Her sonuç insanlara verilmeyebi-

lir. Bilgisayarlar ayrıca muhtelif işlemlerin gerçek zamanlı denetiminde kullanılabilirler. Örneğin imalat sanayii, kimyasal ve endüstriyel sanayiler, birleştirme işlemleri vs. Bu uygulamalarda temel yöntem durumu ve şartları algılamak ve buna göre denetim sinyalleri göndermektir. Böylece işlem denetim altında tutulur.

G/Ç organizasyonu, bilgisayarın ve ona bağlı aletlerin büyüklüğüne bağlıdır. Küçük bir sistem ile büyük bir sistem arasındaki fark, sisteme bağlı çevre aletlerin sayısı ile bunlarla haberleşmek için sistemde bulunan donanımın miktarına bağlıdır. Her bir çevre aleti diğerinden farklı davrandığından bunlarla haberleşmek için çok ayrıntılı bağlantılar gerekir. Çoğunlukla kullanılan çevre aletleriyle haberleşme için bazı teknikler bu bölümde anlatılacaktır.

ASCII Alfasayısal Karakterler

ASCII

Bilgisayar ve giriş birimlerinin kullanıcılarla haberleşmesi genellikle bilgisayar ve çevre aletleri arasında alfasayısal kodların karşılıklı aktarımına bağlıdır. Alfasayısal kodlar için standart ikilik kodlar ASCII kodlarıdır. 7 bit kullanılır, 128 karakter mevcuttur (Çizelge 11.1) 7 bit b_1 den b_7 ye kadar bitlerden oluşur, b_7 en önemli bittir. A harfi, ASCII olarak 100 0001 olarak kabul edilir. 100 sütun ve 0001 satırı. ASCII karakterlerden 94 tanesi karakterler için 34 tanesi ise denetim karakteri olarak kullanılır. Baskı karakteri olarak 26 büyük, 26 küçük ingiliz alfabesi, 10 rakam, 32 özel karakter (%,*,$ gibi) mevcuttur.

34 tane denetim karakteri ASCII çizelgesinde kısaltılmış olarak yer alır. Bunlar verinin yerleştirilmesi ve baskı metninin istenen bir biçimde hazırlanmasını sağlarlar. 3 tip denetim karakteri vardır. Biçim belirleyiciler, bilgi ayırıcıları ve haberleşme denetim karakterleri. Biçim belirleyiciler baskının dökümünü denetlerler. Bunların içinde daktilodan bildiğimiz denetimler vardır. Örneğin bir karakter geri al (BS), yatay sekme (HT) ve satır başı (CR) bunlardandır. Bilgi ayırıcıları paragraf ve sayfa gibi verileri bölümlere ayırmakta kullanılır. Bunların içinde kayıt ayırıcı (RS), dosya ayırıcı (FS) gibi. Haberleşme denetim karakterleri metnin uzak uç birimler arasında aktarımı için kullanılırlar. Metin başı (STX), metin sonu (ETX) metin mesajlarını çerçevelemekte kullanılırlar.

Byte

ASCII bir 7-bit koddur. Halbuki çoğu bilgisayarlar 8 bitlik byte adı verilen birimle işlem görürler. Dolayısıyla ASCII kodları byte başına 1 tane saklanırlar. Fazla bit bazen uygulamaya göre başka amaçlarla kullanılır. Örneğin bazı yazıcılar 8 bit ASCII karakterlerini önemli biti 0 olmak şartıyla tanırlar. Önemli fazla biti 1 olan diğer 128 karakter yunan alfabesi, italik harfler ve diğer grafik karakterler için kullanılır. Haberleşme olduğunda fazla bit eşlik biti olarak kullanılır.

Çizelge 11.1 ASCII karakterleri

	$b_7b_6b_5$							
$b_4b_3b_2b_1$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	‘	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	″	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	’	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	–	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	—	o	DEL

Kontrol karakterleri

NUL	Boş	DLE	Veri bağlantı kaçışı
SOH	Başlık başlangıcı	DC1	Alet 1 denetimi
STX	Metin başlangıcı	DC2	Alet 2 denetimi
ETX	Metin sonu	DC3	Alet 3 denetimi
EOT	İletilenin sonu	DC4	Alet 4 denetimi
ENQ	Sorgu	NAK	Negatif algılandı
ACK	Algılama	SYN	Eş zamanlamayı sağlama
BEL	Zil	ETB	İletilen bloğun sonu
BS	Geri tuşu	CAN	İptal
HT	Yatay sekme	EM	Ortanın sonu
LF	Satır besleme	SUB	Yerine koyma
VT	Düşey sekme	ESC	Kaçış
FF	Sayfa besleme	FS	Dosya ayırıcı
CR	Yeni satır başı	GS	Grup ayırıcı
SO	Aralık kapama	RS	Kayıt ayırıcı
SI	Aralık açma	US	Birim ayırıcı
SP	Boşluk	DEL	Silme

11.2 Giriş / Çıkış Ara Yüzeyi

G/Ç ara yüzeyi bilgisayarda bulunan bilgilerin dış G/Ç aletleri arasında aktarmanın yapılması için bir yöntem sunar. Bilgisayara bağlanan çevre aletleri MİB ne bağlanmaları ve onunla haberleşmeleri için özel haberleşme bağlantısına gerek duyarlar. Bu bağlantının amacı merkezi bilgisa-

yar ile her bir çevre aleti arasındaki farkların ayrı ayrı çözülmesi ve giderilmesidir. En büyük farklar şunlardır.

1. Çevre aletleri elektro mekanik ve elektro manyetik aletlerdir. MİB ve bellek ise elektronik aletlerdir. Bu yüzden çalışma şekilleri çok farklıdır. Sinyal değerlerinin dönüştürülmesi gerekebilir.
2. Çevre aletinin aktarım oranı MİB nin aktarım oranından çok yavaştır. Sonuçta eş zamanlama mekanizması gerekir.
3. Veri kodları ve biçimleri çevre aletten farklı, MİB ve bellekte farklıdır.
4. Çevre aletlerin her birinin çalışma kipleri birbirinden farklıdır. Bu yüzden her biri öyle denetlenmelidir ki birbirlerinin çalışmalarını bozmasınlar.

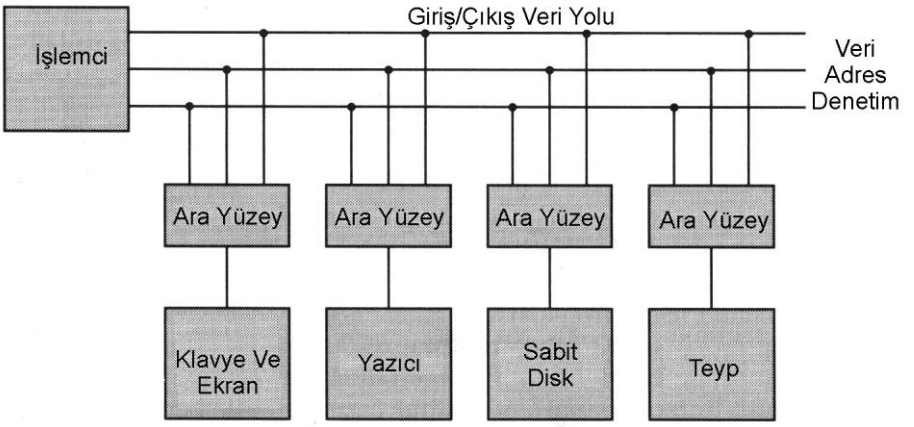
Ara Yüzey

Bu farkları ortadan kaldırmak için MİB ile çevre aletleri arasında özel donanımlar bulunur. Bu donanım G/Ç aktarımlarını yönetir ve eş zamanlı yapar. Bu bileşenlere ara yüzey birimleri adı verilir. Çünkü bunlar işlemci veri yolu ile çevre aleti arasında aracılık yaparlar. Buna ek olarak her bir çevre aletinin kendi denetimleri vardır ve çevre aleti içindeki işlemleri denetler.

G/Ç Veri Yolu ve Ara yüzey Modülleri

İşlemci ve muhtelif çevre aletleri arasında tipik bir haberleşme bağlantısı şekil 11.1 de görülmektedir. G/Ç veri yolu veri hatları, adres hatları ve denetim hatlarından oluşur. Sabit disk, yazıcı ve uç birim genel amaçlı bir bilgisayarın parçalarıdır. Teyp bazı bilgisayarlarda yedekleme amacıyla kullanılır. Her çevre aletinin kendi ara yüzeyi vardır. Her ara yüzey adresi çözer ve G/Ç veri yolundan gelen denetim sinyali yardımıyla bunu icra eder (veya icra edilir hale getirir). Ayrıca çevre aletleri denetim birimlerine sinyaller gönderir. Aynı zamanda veri akışını eş zamanlı yapar ve işlemci ile çevre aleti arasındaki aktarımı yönetir. Her çevre aletinin kendi özel denetim birimi vardır ve bu birim bu elektro mekanik birimin çalışmasını denetler. Örneğin yazıcı denetimleri, yazıcıda kağıt akışını, baskı zamanını ve baskı karakterlerinin seçimini denetler. Denetim birimler çevre aletinden ayrı veya onunla bütünleşik olabilir.

İşlemciden gelen G/Ç veri yolu, çevre aleti ara yüzlerine bağlanmıştır. Bu birimle haberleşmek için, işlemci aletin adresini adres veri yoluna yazar. Her ara yüzeyde bir adres çözücü vardır. Ara yüzey adres veri yolunda kendi adresini görünce, kendi denetiminde olan aletle veri yolu hatları arasını aktif hale getirir. Diğer aletler adresleri olmayınca ara yüzey tarafından etkisiz halde tutulurlar.



Şekil 11.1 G/Ç aletleri ile G/Ç veri yolunun bağlantısı

G/Ç Komutu

Adresin, adres veri yoluna yazılmasıyla birlikte işlemci denetim hatlarına bir fonksiyon yazar. Seçilen ara yüzey bu fonksiyonu alır ve bunu icra eder. Bu fonksiyon koduna G/Ç komutu denir. Bir buyruğun icra edilmesi için bunun gönderilmesi şarttır. Komutun anlaşılması, gönderildiği aletin (adreslenen aletin) bunu anlamasına veya nasıl anladığına bağlıdır. Ara yüze gönderilen 4 tip komut vardır. Bunlar denetim, durum, veri çıkış ve veri giriştir.

Denetim Komutu

Çevre aletinin aktif hale getirilmesi ve ona ne iş yapacağının bildirilmesi işidir. Örneğin manyetik teyp birimine, bir kayıt geriye gitmesi için emir verilebilir, yada geriye sarma veya ileri gitme komutları verilebilir. Verilecek özel komut çevre aletinin cinsine bağlıdır. Her çevre aleti, kendine özel belirgin komut dizilerini alır.

Durum

Bir durum komutu çevre aleti ve ara yüzeydeki muhtelif durumları test etmek için kullanılır. Örneğin bilgisayar aktarım yapmadan önce çevre aletinin durumunu bilmek isteyebilir. Aktarım sırasında bir veya fazla hata olabilir. Hatalar ara yüzey tarafından bulunmaktadır. Bu hatalar durum bitlerinin set edilmesiyle belli olur. İşlemci bu bitleri belli aralıklarla okur.

Veri Çıkış Komutu

Veri yolu içindeki verinin ara yüzey yazaçlarından birine aktarılmasını sağlar. Bir teyp düşünelim. Bilgisayar bir denetim komutu göndererek teybi çalıştırır. Sonra teybin durumunu bir durum komutuyla anlar. Eğer teyp doğru durumda ise, işlemci bir veri çıkış komutu gönderir. Ara yüzey bu komutlara cevap olarak veri yolu içindeki veriyi kendi buffer yazaçlarından birine aktarır. Bundan sonra ara yüzey teyp denetim birimi ile haberleşir ve verinin teybe kaydedilmesini sağlar.

Veri Giriş Komutu

Veri çıkış komutunun tersidir. Bu komutta ara yüzey çevre aletinden veri alır ve onu kendi buffer yazacına yazar. İşlemci, durum komutuyla

verinin hazır olup olmadığına bakar, sonra veri giriş komutu gönderir. Ara yüzey veriyi veri hatlarına aktarır.

G/Ç a Karşı Bellek Veri Yolu

İşlemci G/Ç ile haberleşmenin yanı sıra bellek ile de haberleşmek durumundadır. Tıpkı G/Ç veri yolu gibi, bellek veri yolu da veri, adres ve oku/yaz denetim hatlarından oluşur. Bellek ve G/Ç ile haberleşmede kullanılan veri yolları için 3 yol vardır.

1. Bellek ve G/Ç için ayrı 2 veri yolu kullanmak.
2. Tek bir veri yolu kullanmak fakat denetim hatlarının farklı olması.
3. Hepsinin ortak olması.

GÇİ

Birinci yöntemde veriler, adresler ve denetimler ayrı ayrıdır. Birisi bellek diğeri G/Ç için. Bu olay, G/Ç işlemi için ayrı bir işlemcisi bulunan bilgisayarlarda vardır. Bellek, MİB ve GÇİ (G/Ç işlemcisi) ile bellek veri yolu aracılığıyla haberleşir. GÇİ aynı zamanda bütün G/Ç birimleriyle ayrı ve bağımsız bir veri yolu ile haberleşir. GÇİ yönteminin amacı çevre aletleriyle belleğin bağımsız bir biçimde haberleşebilmesi ve verilerin aktarılabilmesidir. GÇİ bazen veri kanalı olarak adlandırılır. Gçi nin fonksiyonları bölüm 11.7 de ayrıntılı anlatılacaktır.

Yalıtılmışa Karşı Bellek-Haritalanmış G/Ç

Birçok bilgisayar, bellek veya G/Ç ile işlemci arasında ortak veri yolu kullanılır. Bellek ve G/Ç aktarımları arasındaki ayırım ayrı oku/yaz hatları yardımıyla olur. MİB, verdiği adresin bir bellek kelimesini mi yoksa bir G/Ç ara yüzeyi mi gösterdiğini mümkün oku/yaz hatlarını seçerek belirtir. G/Ç oku ve yaz hatları sadece G/Ç aktarımı sırasında izinlendirilir. Bellek O/Y hatları ise bellek aktarımı sırasında izinlendirilmiş durumundadır. Bu yöntem G/Ç adreslerini bellek adreslerinden yalıtır. Bu yönteme ortak veri yolunda yalıtım yöntemi denir.

Yalıtılmış G/Ç

Yalıtılmış G/Ç durumunda, MİB belirgin G ve Ç buyruklarına sahiptir. Bu buyrukların her birinde ara yüzey yazacının adresi bulunmaktadır. MİB buyruk işlem kodunu aldığı ve kodunu çözdüğü zaman bunun bir G/Ç buyruğu olduğunu görünce, buyruktaki adresi ortak adres veri yoluna yerleştirir. Bu sırada MİB G/Ç yaz ve G/Ç oku denetim hatlarından gerekeni izinlendirir. Bu işlem bilgisayara bağlı çevre aletlerine, adres veri yolunda yazılı adresin çevre aletlerinden birine ait olduğunu ve bir bellek kelimesinin adresi olmadığını gösterir. Diğer taraftan MİB bir buyruğu veya bir operandı bellekten alıp getirirken, bellek adresini adres

veri yoluna yazar ve bellek oku veya bellek yaz hattını izinlendirir. Bu dış birimlere adresin bir bellek kelimesine ait olduğunu ve bir G/Ç ara yüzey adresi olmadığını belirtir.

Bellek-haritalanmış

Yalıtım yöntemi, bellek ve G/Ç adreslerini birbirinden yalıtır, bu yüzden bunlar birbirinden etkilenmez. Bir diğer yöntem bellek ve G/Ç için aynı adres uzayının kullanılmasıdır. Yani bir tek oku ve bir tek yaz hattı bulunmaktadır. Dolayısıyla bellekle, G/Ç adresleri arasında bir ayırım yapılmaz. Buna bellek-haritalanmış G/Ç adı verilir. Bilgisayar, ara yüzey yazacını, belleğin bir parçası sayar, bu yüzden ara yüzey adresleri bellek için kullanılmaz. Ve bellek adres kapasitesi oldukça azalır.

Bu yöntemde özel G/Ç buyrukları yoktur. MİB ara yüzey yazaçlarında bulunan veriyi, bellek kelimelerini işlemek için kullandığı buyruklar ile okur veya yazar. Ancak bu durumda ara yüzey yazaçlarının, normal adres alanındaki oku/yaz komutlarına uyması gerekir. Tipik olarak bir kesimin adres uzayının tamamı ara yüzey yazaçları için ayrılmıştır. Fakat genel olarak yazaçlar herhangi bir adreste olabilirler, yeter ki aynı adreste bellek kelimesi olmasın.

Bu şekilde yapılmış bilgisayarlar G/Ç verisi için bellek adreslemeli buyrukları kullanırlar. Böylece bilgisayarın G/Ç aktarımı ve bellek aktarımı için aynı buyrukları kullandığı görülür. Tipik bir bilgisayarda bellek adreslemeli buyrukların sayısı G/Ç buyrukların sayısından fazladır. Halbuki bu yöntemde bütün bellek buyrukları G/Ç içinde geçerlidir.

G/Ç Ara Yüzey Örneği

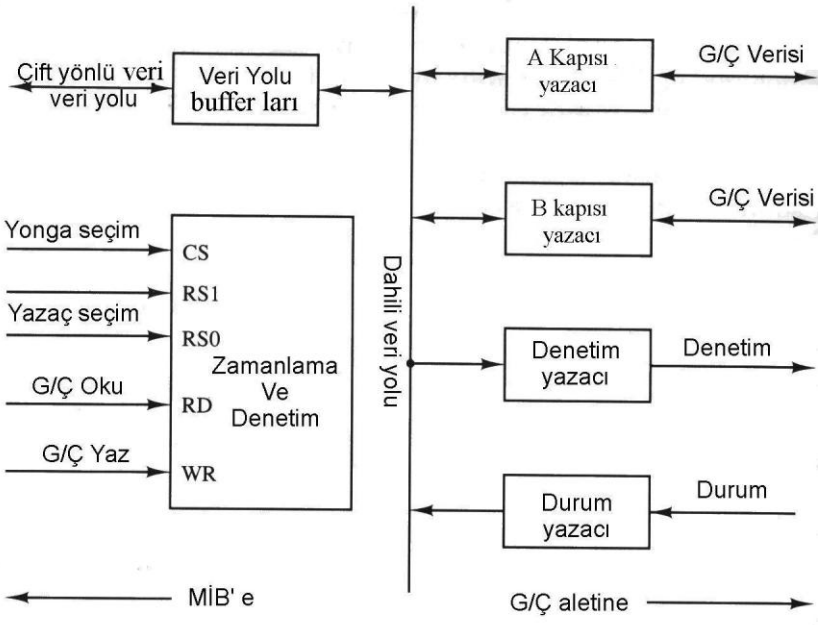
G/Ç kapısı

G/Ç ara yüzeyine bir örnek şekil 11.2 de görülmektedir. Burada kapı adı verilen 2 veri yazacı vardır. Bunlardan başka bir denetim yazacı, bir durum yazacı, veri yolu buffer ları ve zamanlama ve denetim devreleri vardır. Ara yüz, MİB ile bir veri yolu üzerinden haberleşir. Yonga seçim ve yazaç seçim girişleri ara yüzeye tahsis edilen adresi belirler. G/Ç oku ve yaz iki denetim hattı olup giriş ve çıkışı belirlerler. 4 yazaç ara yüze eklenmiş olan G/Ç aletiyle doğrudan haberleşebilir.

G/Ç verisi alete veya aletten dışarı *A* veya *B* kapısıyla aktarılabilir. Ara yüzey bir giriş, bir çıkış veya hem giriş hem de çıkış isteyen bir aletle çalışabilir. Eğer ara yüzey bir yazıcıya bağlanmışsa yalnızca veriyi çıkışa gönderir. Eğer bir karakter okuyucuya hizmet veriyorsa sadece veriyi girişten alır. Sabit disk her iki yöne farklı zamanlarda veri aktarır. Bu yüzden ara yüzey iki yönlü hatları kullanır. Ara yüze gelen uygun bir kelime ile geçiş izni alınan komutu alete aktarılır. Böyle bir sistemde G/Ç veri yolu içindeki fonksiyona ihtiyaç yoktur. Çünkü denetim, denetim yazacına gönderilmiştir, durum bilgisi ise durum yazacına gitmiştir. Veriler ise *A* ve *B* yazaçlarından gidip gelirler. Dolayısıyla denetim, veri ve

durum bilgileri aynı veri yolu üzerinden yollanır. Veri, denetim veya durum bilgilerinin ayırımı MİB ile haberleşmekte olan ara yüzey yazaçları tarafından yapılır.

Denetim yazacı, denetim bilgilerini MİB den alır. Uygun bitlerin denetim yazacına gelmesiyle ara yüzey ve ona bağlı G/Ç aleti bir çok farklı kipe geçirilebilir. Örneğin *A* kapısı giriş, *B* ise çıkış kapısı olarak tanımlanabilir. Manyetik teyp geriye sarma ve başa gelme yapması veya ileri dönmesi için emir alabilir. Durum yazacındaki bitler durum şartlarını belirtmekte ve veri aktarımı sırasında kayıt hatalarını tutmakta kullanılır. Örneğin bir durum biti *A* kapısının G/Ç aletinden bir veri aldığını gösterir. Başka bir bit aktarma sırasında eşlik hatası olduğunu gösterebilir.



CS	RS1	RS0	Seçili Yazaç
0	×	×	Hiçbiri : Veri yolu yüksek dirençli
1	0	0	A Kapısı yazacı
1	0	1	B Kapısı yazacı
1	1	0	Kontrol Yazacı
1	1	1	Durum Yazacı

Şekil 11.2 G/Ç ara yüzey birim örneği

Ara yüzey yazaçları, MİB ile, iki yönlü veri yolu kullanarak haberleşirler. Adres veri yolu ara yüzey birimini, yonga seçimi yardımıyla ve iki yazaç seçim girişleriyle seçer. Ara yüzey yazaçlarına tahsis edilen adresin algılanabilmesi için dış bir devreye (genellikle kod çözücü) gerek vardır. Ara yüzey adres veri yolu tarafından seçilince bu devre *CS* yi izinlendirir. İki yazaç seçim girişleri *RS*1 ve *RS*0 genellikle adres veri yolunun en önemsiz iki hattına bağlıdır. Bu iki giriş ara yüzdeki 4 yazaçtan birini, şekildeki çizelgede belirtildiği gibi seçer. Seçilen yazacın içeriği, *RD* sinyali izinlendirilince bilgi veri yolu yardımıyla MİB ne aktarılır. MİB G/Ç yazma girişi izinlendirilince ikilik bilgiyi veri yolu üzerinden seçilen yazaca aktarır.

11.3 Eş Zamanlı Olmayan Veri Aktarımı

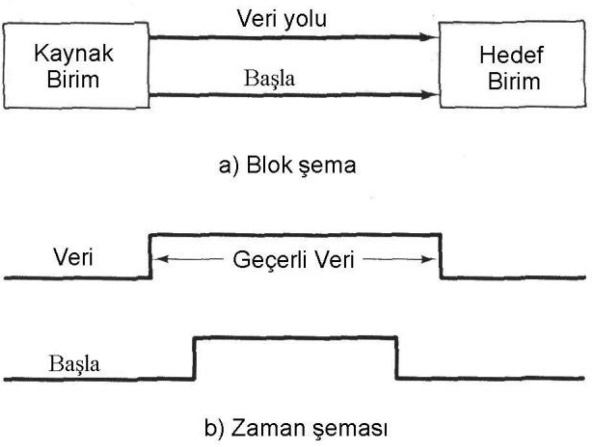
Sayısal bir sistemde iç işlemler saat vuruşu yardımıyla eş zamanlı olarak yapılır. Saat vuruşları ortak bir darbe üzereci tarafından üretilir. Saat vuruşları bir birim içindeki bütün yazaçlara gönderilir. İç yazaçlar arasındaki bütün veri aktarımları bir saat vuruşu anında hep birden oluşur. MİB ve G/Ç ara yüzeyi gibi iki birim birbirlerinden ayrı olarak tasarlanmış ve imal edilmişlerdir. Eğer ara yüzdeki yazaçlar ile MİB deki yazaçlar ortak bir saati paylaşırlarsa bu iki birim arasındaki aktarım eş zamanlıdır denir. Birçok durumlarda her bir birimin iç zamanlaması diğerininkinden farklıdır. Ayrıca her bir birim kendi iç yazaçları için kendi saatini kullanır. Böyle bir durumda iki birim birbirine eş zamanlı olmayandır denir. Birçok bilgisayar sisteminde bu yaklaşım kullanılır.

Başla vuruşu, Onaylama

İki bağımsız birim arasında eş zamanlı olmayan veri aktarımı, verinin aktarıldığını ve aktarma zamanını bildiren sinyallerin iki birim arasında gönderilmesini gerektirir. Buna erişmenin bir yolu birimlerden birinin başla vuruşudur. Bu vuruşla diğer birime aktarım olduğunu haber verir. Çok kullanılan diğer bir yöntem verinin denetim sinyaliyle beraber gönderilmesidir. Bu sinyal yardımıyla veri yolu içinde verinin varlığı anlaşılır. Veriyi alan birim, veriyi aldığına dair bir sinyal gönderir. İki bağımsız birim arasında bu anlaşmaya onaylama denir.

Zaman Şeması

Başla vuruşu ile onaylama yöntemi sadece G/Ç aktarımlarında kullanılmaz, herhangi iki bağımsız birimin haberleşmesinde de yoğun biçimde kullanılır. Genel olarak aktarımı yapan birim kaynak ve alıcı birimde hedef olarak adlandırılır. Örneğin MİB yazma ve çıkış işlemleri için kaynak, giriş ve okuma aktarımlarında hedeftir. İki bağımsız birim arasındaki eş zamanlı olmayan aktarımı zaman şemasıyla genelde gösterilir. Zaman şemasıyla denetim sinyalleri ve veriler arasındaki ve veri yolu içindeki gerekli zaman bağlantıları gösterilmektedir. Eş zamanlı olmayan veri



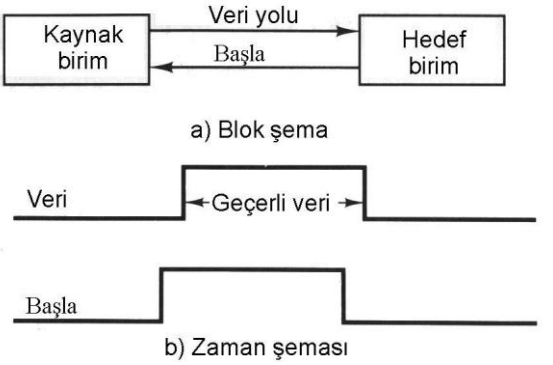
Şekil 11.3 Veri aktarımı için kaynaktan çalıştırılan başla

aktarımı sonrasında denetim sırası, aktarımın kaynak tarafından mı, yoksa hedef tarafından mı başlatıldığına bağlıdır.

Başla Denetimi

Başla denetim yöntemi, eş zamanlı olmayan aktarımında her bir aktarım için tek bir denetim hattı kullanır. Başla kaynak veya hedef tarafından çalıştırılabilir. Şekil 11.3(a) da kaynak tarafından başlatılan aktarım gösterilmiştir. Veri yolu ikilik bilgileri kaynaktan hedefe taşır. Tipik olarak veri yolu, tüm byte veya kelimeyi taşımak için yeterli hatta sahiptir.

Başla tek bir hattır. Hedef birimini uyarır, bu uyarma veri yolunda geçerli veriler olduğu anlamına gelmektedir. Şekil 11.3(b) de görüldüğü gibi kaynak önce veriyi veri yoluna yerleştirir. Kısa bir gecikmeden sonra kaynak başla vuruşunu aktifler. Kısa gecikmenin sebebi veri yoluna yerleştirilen veriler ile veri yolunun kararlı bir değer durumuna gelmesidir. Veri yolundaki veri ve başla vuruşu, hedef birimin veriyi alması için yeterli bir zaman kadar aktif kalırlar. Genellikle hedef birim, veriyi kendi iç yazaçlarından birine almak için başla vuruşunun iniş kolunu bekler. Başla vuruşunun bitmesinden kısa bir süre sonra kaynak, veri yoluna veri verme sürecini durdurur bir başka deyişle veri çıkış uçlarını kapatır. Gerçekte kaynak veri yolu içindeki bilgilerini değiştirmez. Başla sinyalinin kesilmesi sadece veri yolundaki verilerin artık geçersiz olduğunu bildirir. Yeni veriler ancak başla vuruşunun gönderilmesinden sonra hedef birim için hazır demektir.



Şekil 11.4 Veri aktarımı için varıştan çalıştırılan başla

Şekil 11.4 hedef tarafından başlatılan veri aktarımını gösteriyor. Bu durumda hedef başla vuruşunu çalıştırır ve kaynağa veri vermesini söyler. Kaynak birimi cevap olarak istenen ikilik veriyi veri yoluna yerleştirir. Veriler, alıcının bu verileri almasına yetecek bir zaman kadar veri yolunda geçerli olmalıdır. Başla vuruşunun inen kolu bu işi başlatır. Hedef birimi bundan sonra başlayı seçilmez kılar. Kaynak, önceden tayin edilmiş bir süre kadar sonra veri yolundaki veriyi kaldırır. (Beslemeyi keser).

Birçok bilgisayarda başla vuruşları MİB içindeki saat tarafından denetlenir. MİB daima bunları denetlemekte ve dış birimlere veriyi nasıl aktaracaklarını söylemektedir. Örneğin şekil 11.3 teki başla, MİB tarafından, bellek denetim birimine gönderilen belleğe-yaz denetim sinyali olabilir. Kaynak MİB dir. Veri yoluna bir kelime yazar ve bellek birimini haberdar eder, bellek hedef birimdir. İşlem ise belleğe yaz işlemidir. Benzer şekilde, şekil 11.4 teki başla, aslında bir bellek okuma sinyali olabilir. MİB okuma işlemini başlatmak için kaynak olan belleği uyarır. Bellek seçilen kelimeyi veri yoluna yükler.

MİB ile ara yüzey arasındaki aktarımlar burada tanımlandığı gibidir. Bu ara yüzey ile bir G/Ç aleti arasındaki veri aktarımı ise genellikle, onaylama hatlarıyla gerçekleştirilir.

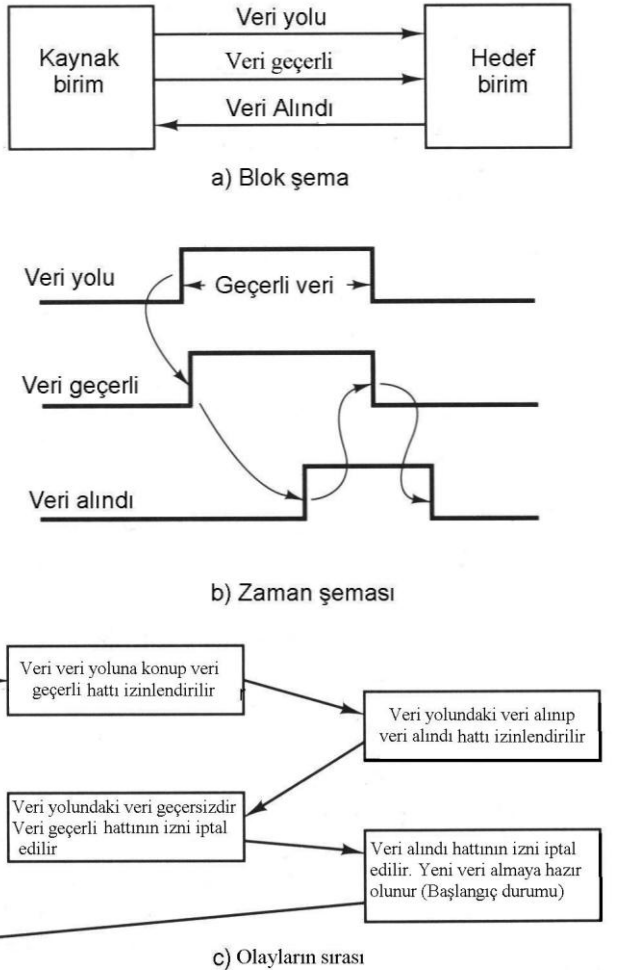
Onaylama

İki-hatlı Denetim

Başla yönteminin dezavantajı aktarımı gerçekleştirecek olan kaynak biriminin gönderdiği, yani veri yoluna yerleştirdiği veriyi hedef birimin gerçekten alıp almadığını bilmemesidir. Benzer şekilde eğer hedef aktarımı başlatıyorsa kaynağın veriyi, veri yoluna yerleştirip, yerleştirmediğini bilmez. Onaylama yöntemi ikinci bir denetim sinyali yollayarak bu sorunu çözer. Bu sinyal aktarımı yapan birime yanıttır. İki hatlı denetim onaylama yönteminin temel prensibi şöyledir. Denetim hatlarının birisi

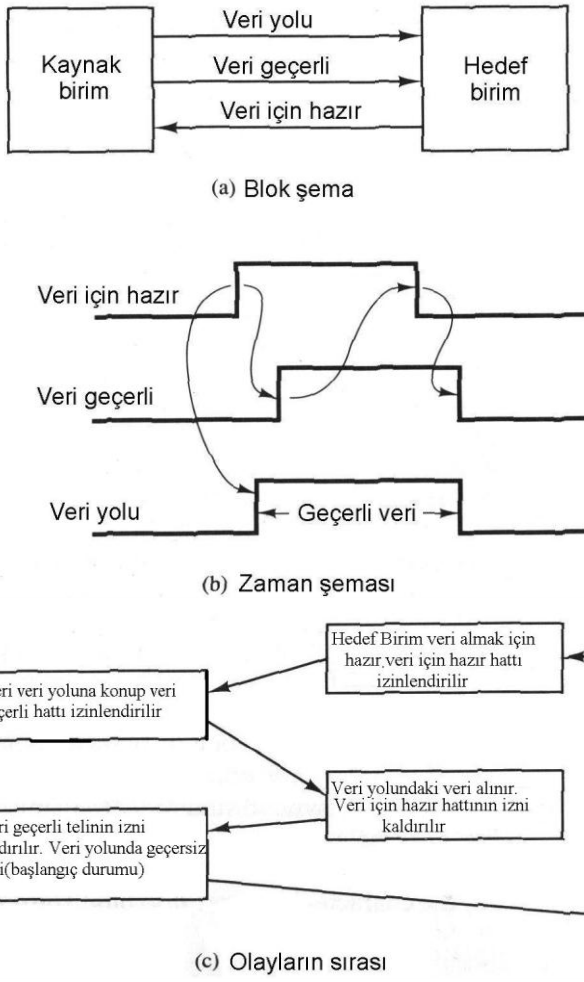
veri akışı ile aynı yöndedir ve kaynaktan hedefe gider. Kaynak birimi tarafından kullanılır ve hedefin veri yolu üzerinde geçerli veri bulunup bulunmadığını anlamasını sağlar. Diğer denetim hattı aksi yöndedir ve kaynağa veriyi alıp alamadığını bildirir. Denetim sırası aktarımı başlatan birimce yapılır.

Şekil 11.5 kaynak tarafından başlatılan aktarmayı gösteriyor. İki onaylama hatlarından birisi veri geçerli hattı olup kaynak tarafından, diğeri veri algılandı olup hedef tarafından üretilen sinyalleri taşırlar. Zaman şeması iki birimce alışverişi yapılan sinyalleri gösteriyor. (c) kıs-



Şekil 11.5 Onaylama yöntemi kullanılarak kaynak başlangıçlı aktarım

mında listelenen, olaylar sırası muhtemel 4 durumu gösteriyor. Sistem herhangi bir anda bu 4 temel durumdan birinde bulunur. Kaynak birimi, veriyi veri yoluna yazarak ve veri geçirli sinyalini göndererek aktarmayı başlatır. Hedef, veriyi alınca, veri algılandı sinyalini yollar. Bundan sonra kaynak veri göndermeyi keser ve veri geçerli sinyalini durdurur. Bu sinyal kesilince hedef veri algılandı sinyalini durdurur. Sistem ilk durumuna döner. Veri algılandı sinyalinin kesilmesi hedefin yeni veri almaya hazır olduğunu belirtir. Bu sistemde rast gele gecikmeler olabilir veya böylece her birimin kendi hızına göre iş görmesi sağlanmış olur. Aktarım, hızı veya oranı en yavaş olan birime uygun olarak (onun hızıyla) gerçekleşir.



Şekil 11.6 Onaylama yöntemi kullanılarak hedef başlangıçlı aktarım.

Hedef tarafından başlatılan ve onaylama ile denetlenen veri aktarımı şekil 11.6 da gösterilmektedir. Hedef tarafından gönderilen sinyalin adı veri için hazır olarak değişir. Bu durumda kaynak birimi veri için hazır sinyalini almadan veriyi veri yoluna yazmaz. Onaylama yöntemi kaynak başlatmalıda olduğu gibi devam eder. Her iki yöntemde birbirinin aynıdır. Sadece veri için hazır sinyali yerine veri algılandı sinyali isim olarak değişir. Birde ilk başlatan değişir.

Zaman Bitti

Zaman bitti onaylama yöntemi yüksek derecede esneklik ve güvenilirlik sağlar. Çünkü her iki birimde aktif olarak katılmaktadır.

Eğer birimlerden birisi bozuk ise aktarım gerçekleşmez. Böyle bir hata zaman bitti mekanizmasınca fark edilir. Belli bir zaman içinde aktarım tamamlanmazsa bir alarm gönderir. Zaman bitti mekanizması iç saat tarafından denetlenir. Onaylama hatlarından biri izinlendirilince saat saymaya başlar. Eğer belli bir zaman içinde karşı sinyal gelmezse, bir hata olduğuna karar verilir. Zaman bitti sinyali işlemciyi durdurmak ve hatayı düzeltmek için bir servis yordamı çalıştırmakta kullanılır.

Eş Zamanlı Olmayan Seri Aktarım

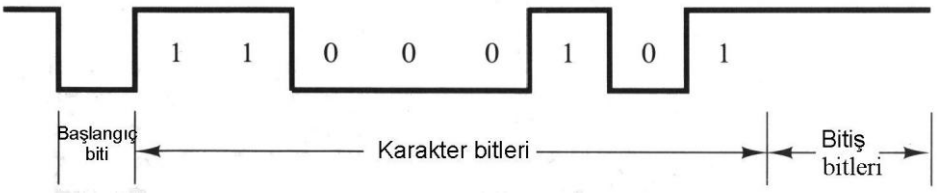
İki birim arasında aktarım paralel veya seri olarak yapılabilir. Paralel veri aktarımında mesajın her biti kendi özel yolundan gider. Tüm mesaj aynı anda hedefine ulaşır. Bunun anlamı n bit mesajın n farklı yolla yollanmasıdır. Seri veri aktarımında her bir bit bir zaman sırasıyla yollanmalıdır. Bunun için bir tek hat kullanılır. Paralel aktarım hızlıdır, fakat çok hat ister. Kısa mesafeler için kullanılır. Seri aktarım yavaştır, fakat ucuzdur. Bir çift iletkene gereksinimi vardır.

Eş Zamanlı

Seri aktarım eş zamanlı veya eş zamanlı olmayan olabilir. Eş zamanlı aktarımda iki birim ortak saat frekansına sahiptir. Bitler sürekli bir biçimde aktarılır. Aktarım saat hızıyla olur. Uzun mesafeli seri aktarımda her bir birim ayrı saatler tarafından denetlenmektedir. Fakat saatler aynı frekansta çalışabilir. Eş zamanlama sinyalleri iki birim arasında belli sürelerle gönderilir ve her iki birimin saati eş zamanlı olarak çalıştırılır. Eş zamanlı olmayan aktarımda ikilik bilgiler hazır olunca gönderilir. Gönderilecek bilgi olmayınca hat boş kalır. Bu eş zamanlı aktarıma ters bir yaklaşımdır. Çünkü eş zamanlı aktarımda her iki birimin eş zamanlı çalışması için bitler sürekli gönderilmelidir. Eş zamanlı seri haberleşme 11.8 de anlatılacaktır.

Eş Zamanlı Olmayan, Başlama Biti

Etkileşimli uç birimlerde kullanılan eş zamanlı olmayan seri veri aktarımı özel bitler kullanır. Bunlar karakter kodunun başına ve sonuna eklenmiştir. Bu yöntem ile her bir karakter 3 parçadan oluşur. Başlama biti, karakter bitleri, bitiş biti. Hiçbir karakter aktarılmadığında taşıyıcı sürekli 1 durumunda kalır. Başlangıç biti daima 0 dır ve karakter başlan-



Şekil 11.7 Eş zamanlı olmayan Seri Aktarım

gıç biti olarak iş görür. Son bit daima 1 dir. Bu biçim şekil 11.7 de gösterilmiştir.

Aktarılan karakter, alıcı tarafından aktarma kuralları ile bulunur.

1. Bir karakter yollanmadığında hat 1 durumunda tutulur.
2. Karakter aktarımının başlangıcı daima 0 olana başlangıç bitinden anlaşılır.
3. Karakter başlangıç bitinin hemen arkasından gelir.
4. Karakterin son bitinin aktarılmasından sonra bitiş biti gelir, hat 1 değerine döner.

Bitiş Biti

Bu kurallar yardımıyla alıcı başlangıç bitini hattın 1 den 0 a gidişinden anlar. Alıcının saati hattı uygun bit zamanlarında inceler. Çünkü alıcı bitlerin aktarılma hızını bilir. Alınacak karakter bit sayısını bilir. Karakterin gönderilmesinin bitişinden sonra 1 veya 2 bitiş biti gönderilir. Bitiş bitleri daima 1 dir. Bitiş biti karakterin sonunu ve bekleme veya aylaklığın başlangıcını belirler.

Karakterden sonra hat 1 veya 2 zaman kadar 1 durumunda tutulur. Böylece iki birim gönderici ve alıcı eş zamanlı yapılır. Bu bir veya iki bitlik süre aletlerin eş zamanlanması için gerekli olan süre olarak seçilir. Eski elektro mekanik uç birimler 2 bit kullanıyordu. Yeni uç birimler 1 bit kullanır, hat başka karakter aktarılana kadar 1 durumunda kalır. Bitiş zamanı yeni karakterin bir veya 2 zaman takip edilmemesini sağlar.

Aktarım Hızı

Örnek olarak saniyede 10 karakter hızı olan bir aktarım düşünelim. Her bir karakter bir başlangıç bitine sahiptir. Bunun yanında 8 bilgi biti ve bir veya 2 bitiş biti vardır. 10 cps in anlamı her bir karakter aktarım için 0.1 saniye zaman harcanır. Aktarılacak 11 bit olduğundan dolayı bit başına aktarım süresi 9.09 ms olur. Baud oranı: saniye başına aktarılan bit sayısı olarak tanımlanır. 11 bit biçimli 10 karakterin saniyede aktarım sayısı 110 baud tur.

Uç birimin bir klavyesi ve bir yazıcısı olduğunu varsayalım. Bir tuşa basıldığında uç birim seri olarak 11 bit yollar. Bir başka hatla yazıcının 1 karakter basması için 11 bit bir mesaj alması gerekir. Uç birim ara yüzeyinde bir gönderici ve bir alıcı vardır. Gönderici bilgisayardan 8 bit bir

karakteri alır ve 11 bit mesajı yazıcı hattından yollar. Alıcı 11 bit mesajı klavye hattından alır ve 8 bit karakter kodunu MİB ne yollar. Bilgisayar ile benzer etkileşimli uç birimler arasında ara yüzey olacak IC ler mevcuttur. Böyle bir devreye eş zamanlı olmayan veya evrensel eş zamanlı olmayan alıcı-gönderici adı verilir (UART).

Eş Zamanlı Olmayan Haberleşme Ara yüzeyi

Eş zamanlı olmayan haberleşme ara yüzeyinin blok şeması şekil 11.8 de gösterilmiştir, hem alıcı hem gönderici görevi yapar. Ara yüzey denetim girişlerine yüklenen bir denetim byte ı yardımıyla başlangıç durumuna getirilir. Gönderici MİB den veri yolu üzerinden bir veri byte ı alır. Bu byte bir kaydırma yazacına aktarılır. Böylece seri aktarma sağlanır. Alıcı kısmı ise bilgiyi yine bir kaydırma yazacına alır.

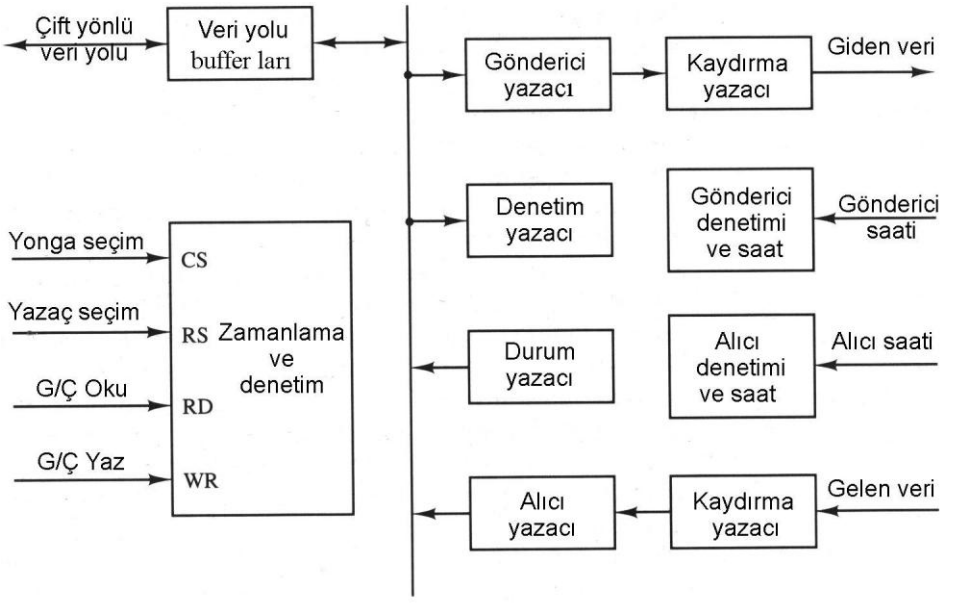
Bir byteın tamamı kaydırma yazacında birikince alıcı yazacına aktarılır. MİB alıcı yazacını veri yolundan bir byte okuması için kullanabilir. Durum yazacındaki bitler giriş ve çıkış bayrakları olarak kullanılır. Aktarma sırasındaki hatalar da bu bit yardımıyla tutulabilir. MİB durum yazacını bayrak bitlerinin durumunu denetim için veya herhangi bir hata olup olmadığını anlamak için kullanır.

Yonga seçim ve oku yaz denetim hatları MİB ile haberleşirler. Yonga seçim girişi, ara yüzeyi adres veri yolundan seçmek için kullanılır. Yazaç seçim, oku (RD) ve yaz (RW) denetimleriyle beraber kullanılır. İki yazaç sadece yaz) ve 2 yazaç sadece oku dur. Seçilen yazaç RS fonksiyonunun değerine ve RD ve WR durumuna bağlıdır. Durum şeması ekli çizelgede verilmiştir.

Eş zamanlı olmayan haberleşme ara yüzeyinin başlangıç durumuna getirilmesi MİB tarafından denetim yazacına bir byte gönderilerek yapılır. Başlangıç durumuna getirilen ara yüzde bazı parametrelerin tanımlandığı belli bir işlem durumu ortaya çıkar. Örneğin bu parametreler Baud oranı, bir karakterdeki bit sayısı, eşlik denetimi yapılıp yapılmayacağı, kaç tane bitiş biti kullanılacağı gibi. Durum yazacında iki bit bayrak olarak kullanılır. Bir bit gönderici yazacının boş olup olmadığını, diğer bit ise alıcı yazacının boş olup olmadığını gösterir.

Gönderici

Gönderici kısmının çalışması şöyledir. MİB durum yazacını okur ve buradan gönderici yazacının boş olup olmadığına ve bayrağa bakar. Eğer boşsa MİB bir karakteri gönderici yazacına aktarır. Ara yüzey bayrağı sıfırlar ve yazacın dolu olduğunu belirtir. Gönderici içindeki kaydırma yazacının birinci biti 0 yapılarak başlangıç biti hazırlanmış olur. Bundan sonra karakter gönderici yazacından, kaydırma yazacına paralel olarak gönderilir. Sonra kaydırma yazacına uygun sayıda bitiş biti eklenir. Bundan sonra gönderici yazacı boş olarak işaretlenir. Karakter ve ekleri her



CS	RS	işlem	Seçili yazaç
0	×	×	Hiçbiri : Veri yolu yüksek dirençli
1	0	WR	Gönderici yazacı
1	1	WR	Denetim yazacı
1	0	RD	Alıcı yazacı
1	1	RD	Durum yazacı

Şekil 11.8 Bir tipik eş zamanlı olmayan haberleşme ara yüzey blok şeması

saat vuruşunda bir karakter olarak ve istenen baud oranıyla gideceği yöne aktarılır. MİB durum yazacı içindeki bayrağa bakarak, boşsa buna yeni bir karakter gönderir. Ara yüze çift buffer denir. Çünkü eski karakter aktarılmaya başlar başlamaz gönderici yazacına yeni bir karakter yüklenir.

Alıcı

Ara yüzeyin alıcı kısmı da göndericiye benzer. veri giriş hattı boş olduğu zaman 1 durumundadır. Alıcı denetimi alıcı hattındaki sinyalini bekler. Eğer bir 0 sinyali gelirse gelen karakter bitleri kaydırma yazacına belirlenen baud oranıyla aktarılır. Veri bitlerinin alınmasından sonra ara yüzey eşlik ve bitiş bitlerini denetler. Bundan sonra karakter bitleri, ekli bitler gönderilmeyerek, paralel biçimde alıcı yazacına gönderilir. Durum

yazacı içindeki bayrak kaldırılarak yazacın dolu olduğu belirtilir. MİB durum yazacına ve bayrağa bakar ve dolu ise veriyi yazaçtan okur.

Ara yüz, aktarma sırasında oluşabilecek hataları, durum yazacında uygun bitleri bir yaparak belirler. MİB, durum yazacına herhangi bir anda bakarak bir hata olup olmadığını görebilir. Ara yüzey aktarma sırasında 3 tür hataya bakar. Bunlar eşlik hatası, çerçeve hatası, tez gelen hatası. Eşlik hatası, eğer gelen 1 lerin sayısı doğru eşlikte değilse olur. Çerçeve hatası, eğer alınan karakterin sonunda gerekli sayıda bitiş biti bulunmaz ise olur. Tez gelen hatası, eğer MİB kaydırma yazacında bir sonraki karakter oluşmadan alıcı yazacını okumaz ise oluşur. Bu hata karakterlerin kaybolmasına neden olur.

FİFO İlk Giren İlk Çıkar Buffer

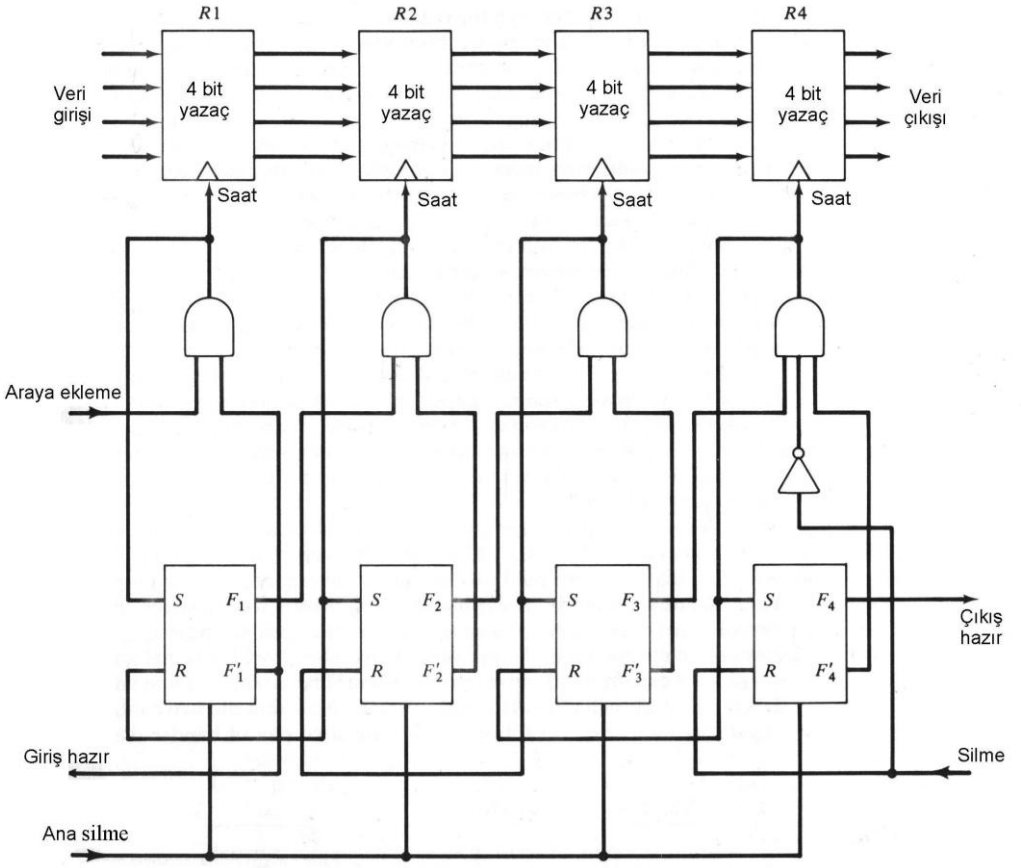
FIFO

FIFO buffer bir bellek birimidir. Bilgiyi öyle tutar ki ilk giren ilk çıkar. FIFO buffer ayrı giriş ve çıkış uçlarıyla verilir. Bu buffer in önemli yanı giren veriyi ve çıkan veriyi farklı oranlarda alabilmesi ve çıkarabilmesidir. Tabi ki çıkan veri girdiği sırada çıkar. Buffer iki birim arasına konduğunda, veriyi bir birimden istenen hızda alır ve diğer birime istenen hızda aktarır. Eğer kaynak birim hedef birimden yavaşsa buffer çok yavaş dolar ve sonunda boş kalır. Eğer kaynak yazacı hedeften hızlı ise FIFO, kaynak verisinin patlamalar şeklinde zaman zaman geldiği durumlarda işe yarar. Bu durumda FIFO buffer ı dolar fakat boşalması için verinin bir kısmını veya tamamını aktaracak zaman bulur. Veri eş zamanlı olmayan olarak aktarıldığından FIFO bazı uygulamalarda faydalıdır.

4 × 4 FIFO buffer ın mantıksal şeması şekil 11.9 da verilmiştir. 4 tane 4 bit yazaç vardır. *RI*, $I = 1, 2, 3, 4$. Bir denetim yazacı vardır. Burada her yazaç için bir yaz-boz F_i $i = 1, 2, 3, 4$ mevcuttur. FIFO 4 bitlik 4 kelimeyi tutabilir. Her bir yazaç içindeki bitlerin sayısı arttırılarak kelime sayısı arttırılabilir.

Denetim yazacında bir yaz-bozun örneğin $F_1 = 1$ olması, bu yaz-bozun yazacında verinin yüklendiğini gösterir. F_1 in sıfır olması, ilgili yazaçtaki verinin geçerli olmadığını gösterir. Denetim yazacı verinin yazaçlardaki hareketini denetler. Denetim yazacının F_i biti 1 yapıldığında ($F_i = 1$) ve F_{i+1} biti 0 durumundadır ($F'_{i+1} = 1$). Bir saat vuruşu üretilerek $R(I + 1)$ yazacının veriyi *RI* yazacından alması sağlanır. Aynı *CP* F_{i+1} yaz-bozunu 1 yapar ve F_i yi de 0 yapar. Bu denetim bayrağının veri ile bir yer sağa kaymasına neden olur.

Önünde boş yazaçlar bulunduğu sürece veri FIFO nun çıkışına doğru hareket eder. İşlem bitene kadar bu kayma durumu devam eder. Son veride *R*4 ten çıkınca bir ana silme ile bütün yaz-bozlar sıfırlanır.



Şekil 11.9 4 × 4 FIFO buffer devre şeması

Buffer a veri, giriş hazır sinyali izinlendirildikten sonra gönderilir. Giriş hazır hattı F_1 denetim yaz-bozu 0 durumunda olunca izinlendirilir. F_1 yaz-bozu 0 olunca $R1$ yazacının boş olduğu (geçersiz veri bulunduğu) belli olur. Yeni veri giriş hatlarından gelir. Bunun için araya ekleme hattı yardımıyla R1 in saati izinlendirilir. Aynı saat F_1 i 1 yapar. Böylece hazır denetimi izni iptal olur. Buda FIFO nun o anda meşgul olduğunu gösterir, yeni veri olamaz. Sağa doğru kayan verilerden dolayı $R2$ boşalınca, $R1$ deki veri $R2$ ye aktarılır, F_1 silinir. Bununla giriş hazır durumu tekrar oluşur. Böylece yeni bir kelime alınabilecek hale gelir. FİFO dolu ise F_1 1 durumunda kalır, hazır 0 dadır. Giriş hazır ve araya ekleme hatları onay çifti oluştururlar. Başlatma hedef tarafındandır.

Yazaçlardan geçen veri, FIFO nun sonunda birikim yapar. Bu durumda çıkış hazır denetim hattı izinlendirilir. Son denetim yaz-bozu F_4 doldurulur . Geçerli verinin $R4$ yazacında bulunduğu bildirilir. Buradaki veri

bir hedef birimi tarafından kabul edilecektir. Kabul sonunda bu birim silme denetimini gönderir. Bununla F_4 0 yapılır ve çıkış hazır izni iptal olur. Bu çıkış verisinin (yeniden 1 yapılıncaya kadar) artık geçersiz olduğunu gösterir. Silme sinyalinin 0 olmasından sonra $R3$ teki veri $R4$ e hareket eder. Eğer FIFO boş ise $R3$ te veri bulunmaz ve F_4 0 durumunda kalır. Çıkış hazır ve silme hatları kaynak başlatmalı bir onaylama çifti oluşturur.

11.4 Aktarma Kipleri

Dış bir birimden gelen ikilik bilgiler daha sonra kullanılmak üzere çoğunlukla bellekte tutulur. Merkezi bilgisayardan, dış bir birime aktarılan bilgilerde genellikle bellek kökenlidir. MİB sadece G/Ç buyruklarını icra eder ve bazı verileri geçici olarak tutabilir. Fakat verilerin en son gideceği veya ilk çıkacağı yer daima bellektir. Merkezi bilgisayar ile G/Ç aletleri arasındaki veri aktarımı bir çok yoldan ele alınabilir ve gerçekleştirilebilir. Bazı kipler MİB yu ara yol olarak alırlar. Diğerleri ise bellekten ve belleğe doğrudan aktarım yaparlar. Çevre aletlerinden ve çevre aletlerine veri aktarımı 3 yolla olabilir.

1. Programlanmış G/Ç
2. Kesme yoluyla başlatılan G/Ç
3. Doğrudan bellek aktarımı (DMA)

Programlanmış G/Ç; Yazılmış olan bilgisayar programındaki G/Ç buyruklarının sonucu oluşur. Her bir veri parçasının aktarımı, program içindeki buyruk tarafından başlatılır. Çoğunlukla aktarım çevre aleti ile MİB yazacı arasında gidiş ve geliş şeklindedir. Diğer buyruklar verinin MİB ile bellek arasında aktarımı için gereklidir. Program denetimi yoluyla veri aktarımı, çevre aletinin daima MİB tarafından yönetilmesini gerektirir.

Kesme

Programlanmış G/Ç yönteminde MİB bir döngüye girerek, G/Ç birimi veri aktarımına hazır olduğunu bildirine dek bu döngüde kalır. Bu işlemciyi gereksiz olarak meşgul ettiğinden zamanı boşa harcatan bir yöntemdir. Bundan kaçınmak için kesme yolu kullanılır. Bu yöntemde MİB bir programı icra ederken ara yüzey çevre aletini denetler. Ara yüzey aletin veri göndermeye hazır olduğunu anlayınca bilgisayara bir kesme isteği gönderir. MİB dışardan gelen bir kesme isteği alınca icra etmekte olduğu programı keser, bir servis programına dallanır. Bununla G/Ç işlemi yerine getirilir ve tekrar eski programa geri dönülür.

Doğrudan Bellek Aktarımı

Programlanmış G/Ç işleminde veri aktarımı çevre aleti ile MİB arasında olur. Doğrudan belleğe erişiminde, veriyi belleğe bellek veri yolu ile alır ve bellekten de bellek veri yolu ile gönderir. MİB veri aktarımını, ara yüze verinin başlangıç adresini ve aktarılarak kelimelerin sayısını vererek başlatır. Sonra tekrar eski işine devam eder. Aktarım başlayınca DMA bellek veri yolu üzerinden istekte bulunur. Eğer bellek denetimleri bu isteği olumlu bulursa DMA veriyi doğrudan belleğe aktarır. MİB bellek erişimlerini geciktirir. Böylece belleğe G/Ç işleminin tamamlanmasına imkan tanır. Çevre aletinin hızı işlemci hızından yavaş olduğundan belleğe yapılan G/Ç aktarımları belleğe MİB nin erişmesine göre daha seyrektir. DMA aktarımı bölüm 11.6 da anlatılacaktır.

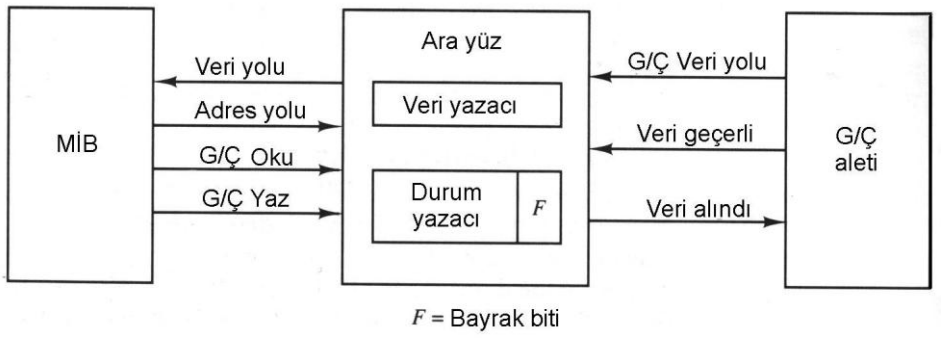
G/Ç İşlemcisi

Birçok bilgisayar ara yüzey mantığı ile DMA olanaklarını birleştirir. Böyle bir birime G/Ç işlemcisi denir (GÇİ). GÇİ birçok çevre aletini DMA ve kesme olanakları ile idare eder. Böyle bir sistemde bilgisayar 3 ayrık modüle ayrılır. Bellek birimi, MİB ve GÇİ. GÇİ 11.7 de anlatılacaktır.

Programlanmış G/Ç Örneği

Programlanmış G/Ç yönteminde G/Ç yapan alet belleğe doğrudan erişemez. G/Ç aletinden belleğe yapılacak aktarımlar birçok buyruğun MİB tarafından icrasını gerektirir. Bunların içinde verinin çevre aletinden MİB ne aktarımını sağlayacak buyruk ile verinin MİB dan belleğe aktarımını sağlayarak bir yerleştirme buyruğu da bulunur. Diğer buyruklar verinin çevre aletinde hazır olduğunu anlamak için kullanılır veya aktarılacak kelimelerin sayısını saymak için kullanılırlar.

Bir G/Ç aletinden MİB ne ara yüzey üzerinden bir aktarma örneği şekil 11.10 da görülmektedir. Alet veriyi veri hazır oldukça byte byte aktarır.



Şekil 11.10 G/Ç aletinden MİB ne veri aktarımı

Bir byte veri hazır olunca alet bunu G/Ç veri yoluna aktarır. Sonra veri geçerli hattını izinlendirir. Ara yüzey bu byte ı kendi veri yazacına alır ve veri alındı hattını izinlendirir. Ara yüzey durum yazacında bir biti 1 yapar. Bu bit bayrak veya F bitidir. Alet veri geçerli hattının iznini iptal eder fakat veri kabul edildi hattının izni iptal edilene kadar yeni veri göndermez. Bu durum şekil 11.5 teki onay yöntemine uygundur.

Bilgisayardaki program durum yazacındaki bayrağı denetler. Eğer G/Ç aleti tarafından veri yazacına veri yerleştirilmişse bunu anlar. Eğer bayrak biti 1 ise MİB veri yazacından veriyi okur, sonra bayrak bitini temizler. Temizlemeyi MİB veya ara yüzey yapabilir. Hangisinin temizleyeceği devrelerin tasarımına bağlıdır. Bayrak temizlenince ara yüzey veri alındı hattının iznini iptal eder, sonra çevre alet bir sonraki veri byte ını aktarır.

MİB için yazılmış bir programın akış şeması şekil 11.11 de gösterilmektedir.

Aletin gönderdiği byte ların bellekte yerleştirileceği kabul edilmektedir. Bir byte ın aktarımı 3 buyruğu gerektirir.

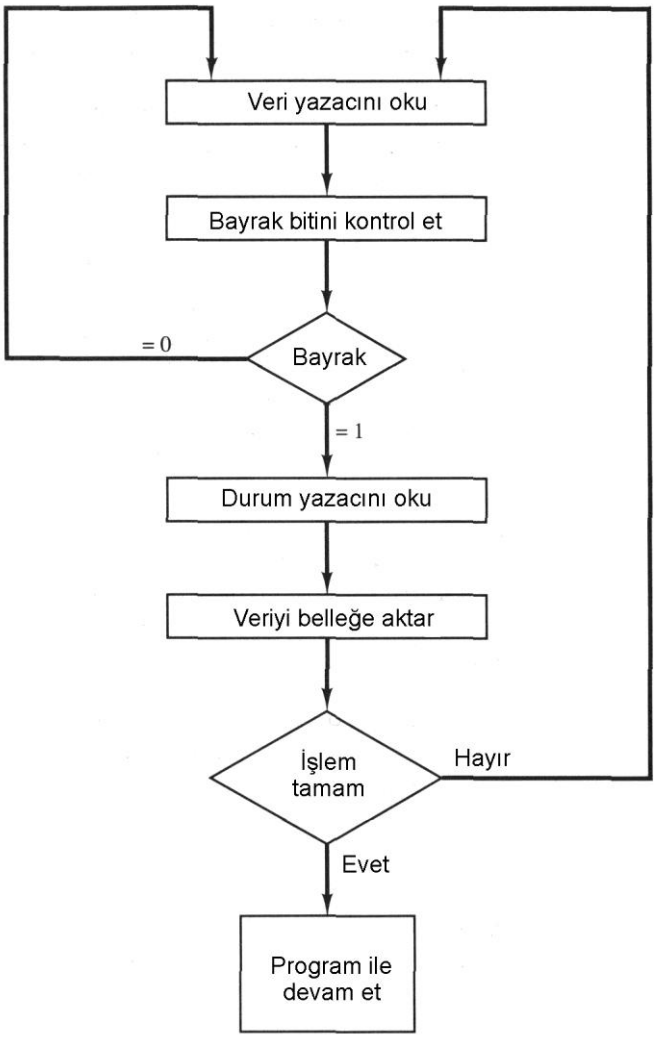
1. Durum yazacının okunması.
2. Durum yazacındaki bayrak bitinin denetimi ve eğer 0 yapılmışsa birinci adıma, 1 edilmişse adıma dallanması.
3. Veri yazacının okunması.

Her byte MİB ne okunur ve sonra bir depola buyruğu ile belleğe aktarılır. G/Ç programlamada ortak yol bir blok kelimenin G/Ç aletinden alınıp bellek buffer ına yazılmasıdır. Bölüm 6 da verilen buyrukları kullanarak girilen karakterleri bellek buffer ına yazan bir program çizelge 6.21 de verilmiştir.

Programlanmış G/Ç yöntemi düşük hızlı küçük bilgisayarlarda veya bir aleti sürekli olarak yönetecek sistemlerde faydalıdır. MİB ile G/Ç aleti arasındaki aktarma oranı bu aktarımın etkin olmasını engeller. Sebebini anlamak için; tipik bir bilgisayarın 2 buyruk icra ettiği var sayılsın. Bu iki buyruk durum yazacını okuma ve bayrağı denetleme 1 μs de. Giriş birimi verileri ortalama 100 byte/sn hızla aktarırsa, bu değer 10000 μs de1 byte eder. Bunun anlamı MİB her bir aktarım arasında bayrağı 10000 kez kontrol eder ve bu yüzden boşa zaman harcayarak faydalı bir iş yapmaz.

Kesme Başlatmalı G/Ç

MİB nin sürekli olarak bayrağı denetlemesine alternatif bir yöntemi ara yüzeyin veriyi aktarmaya hazır olduğu anda bilgisayarı bundan haberdar etmesidir. Bu aktarma kipi kesmeyi kullanır. MİB bir programı çalıştırırken bayrağı denetlemez. Bayrak 1 yapılınca bilgisayara yaptığı iş



Şekil 11.11 Giriş verisi için MİB programının akış şeması

bıraktırılarak bayrağın 1 olduğu haber verilir. MİB, yaptığı işi bırakarak, G/Ç aktarımına bakar. Aktarımın sona ermesinden sonra bilgisayar eskiden çalıştırmakta olduğu programa döner ve kesmeden önce yaptığı işe kaldığı yerden devam eder.

Vektörlü Kesme

MİB, kesme sinyaline bakmak için *PC* daki dönüş adresini bellek yığıtına yazar ve sonra denetim bir servis yordamına dallanır. İşlemcinin dallanacağı adres birimden birime değişir. Genelde 2 yöntem kullanılmaktadır. Biri vektörlü kesme diğeri vektörsüz kesmedir. Vektörsüz kes-

mede dallanma adresi bellekte sabit bir yeri gösterir. Vektörlü kesmede ise, kesmeyi oluşturan çevre alet dallanma bilgisini bilgisayara verir. Bu bilgiye kesme vektörü denir. Bazı bilgisayarlarda kesme vektörü G/Ç servis yordamının ilk adresidir. Diğerlerinde ise kesme vektörü dolaylı adres olup servis yordamının başlama adresinin adresidir. Vektörlü kesme 11.5 te anlatılacaktır.

Yazılım İncelemesi

G/Ç Yordamları

Bir bilgisayar çevre aletlerini denetim ve işlemci ile çevre aletler arasında veri aktarımları için yazılım yordamlarına gerek duyar. G/Ç yordamlarının çevre aletini çalıştırmak için denetim komutları olmalıdır. Aynı zamanda durum yazacını denetleyerek veri aktarımı için ne zaman hazır olacağını bilmelidir. Hazır olunca, bütün veriler aktarılana dek veriler bir bir aktarılmalıdır. Bazı durumlarda bir alet fonksiyonunun icrası gerekebilir. Bu fonksiyon örneğin teybin durdurulması veya yazıcının karakter basmaması olabilir. Hata gözlem ve denetimleri ile ilgili adımlar, aktarımla beraber yürütülür. Kesme denetimli aktarımlarda G/Ç yazılımı çevre aletine hazır olduğu zaman kesme komutu gönderebilmeli ve kesme oluştuğunda bunu servis edebilmelidir. DMA aktarımında, G/Ç yazılımı, DMA yolunu çalıştırmaya başlamalıdır. G/Ç aletlerinin yazılımla denetimi çok karışık bir iştir. Bu nedenle standart çevre aletleri için G/Ç yordamları, imalatçı tarafından bilgisayar sisteminin bir parçası olarak verilir. Bunlar çoğunlukla işletim sistemi içinde yer alırlar. Bir çok işletim sisteminde, yüklü bulunduğu bilgisayar için kullanılabilecek çevre aletlerine uyacak değişik sayıda G/Ç programları bulunur. G/Ç yordamları genellikle işletim sistemi yordamları içinde mevcuttur ve kullanıcı isteği aktarımın yapılması için ayrıntılı makine dili programları yazmak yerine bunları çağırır.

11.5 Kesme İstemi Önceliği

MİB ile G/Ç aleti arasındaki veri aktarımı MİB tarafından başlatılır. Ancak alet hazır olana dek MİB aktarımı yapamaz. Aletin hazır olduğu bir kesme ile anlaşılır. MİB kesme isteğine dönüş adresini PC dan bellek yığıtına yerleştirerek cevap verir. Bundan sonra program bir servis yordamına dallanır ve bu yordam istenen aktarımı yapar. Bazı işlemciler geçerli PSW yi yığıta yerleştirir ve servis yordamı için yeni PSW oluştururlar. Karışıklığı engellemek için PSW ye dikkat edilmeyecektir.

Genelde uygulamalarda birçok G/Ç aleti bilgisayara bağlanmaktadır. Her bir alet kesme isteğinde bulunabilir. Kesme sisteminin ilk görevi

kesmenin geldiği çevre aletini bulmaktır. Bir çok alet aynı anda kesme isteğinde bulunabilir. Bu durumda sistem önce kime hizmet vereceğine karar vermelidir.

Öncelik Kesmesi

Öncelik kesmesi birçok aletin aynı anda istekte bulunmaları halinde hangisinin önce hizmet göreceğini belirten bir sistemdir. Bu sistem ayrıca kesmenin birine hizmet verilirken, hangi koşullarda bilgisayarın kesileceğini de belirlemelidir. İsteklere karşılık devreye giren yüksek seviyeli kesme hizmetlerinin kesilmesi çok ciddi sonuçlar doğurabilir. Yüksek hızlı aktarımlar (örneğin sabit disk gibi) yüksek önceliğe sahiptir. Klavye ise düşük öncelik alır. Aynı anda iki alet istekte bulunursa bilgisayar seviyesi yüksek olana öncelik verir.

Yoklama

Aynı anda gelen kesme isteklerinin önceliği yazılım veya donanım tarafından belirlenebilir. Yoklama işlemi en yüksek öncelikli kaynağı yazılım yoluyla bulur. Bu yöntemde bütün kesmeler için tek bir ortak dallanma adresi vardır. Kesmelerle uğraşan program ise dallanma adresinden başlar ve sırayla kesme kaynaklarına bakar. Kaynakların test sırası her bir kesmenin öncelik sırasıdır. En yüksek öncelikli kaynak en önce test edilir. Eğer kesme sinyali çalışıyorsa denetim bu kaynağa hizmet etmek üzere dallanır. Yüksek öncelikli kaynaktan başlayarak kesme sinyali çalışmıyorsa bir düşük öncelikli kaynağa bakar ve bu işlem sırayla devam eder. Dolayısıyla başlangıç servis yordamı bütün kesmeleri önem sırasıyla test eden bir programa sahiptir. Test sonucuna göre bir çok mümkün servis yordamından birine dallanır. Erişilen servis yordamı bilgisayara kesme gönderen aletler içinde önceliği en yüksek olandır. Yazılım yönteminin kötü tarafı çok sayıda kesme olması halinde test süresinin, alete ayrılan G/Ç süresini taşabilmesi olasılığının olmasıdır. Böyle bir durumda donanım öncelikli bir kesme ile işlemi hızlandırılabilir.

Donanım öncelikli kesme birimi, kesme sistemi içinde her şeyin müdürü gibi davranır. Kesme istemlerini kabul eder, gelen isteklerin hangisinin en yüksek öncelikli olduğunu bulur ve buna bağlı olarak bilgisayara kesme isteği gönderir. İşlemin hızlı olması için her aletin kendi kesme vektörü vardır. Böylece kendi özel servis yordamına doğrudan dallanır. Böylece yoklamaya gerek kalmaz. Çünkü bütün kararlar donanım öncelik kesme birimince alınmıştır. Donanım öncelik fonksiyonu kesme hatlarının paralel veya seri bağlantısıyla sağlanır. Seri bağlama papatya zinciri yöntemi olarak bilinir.

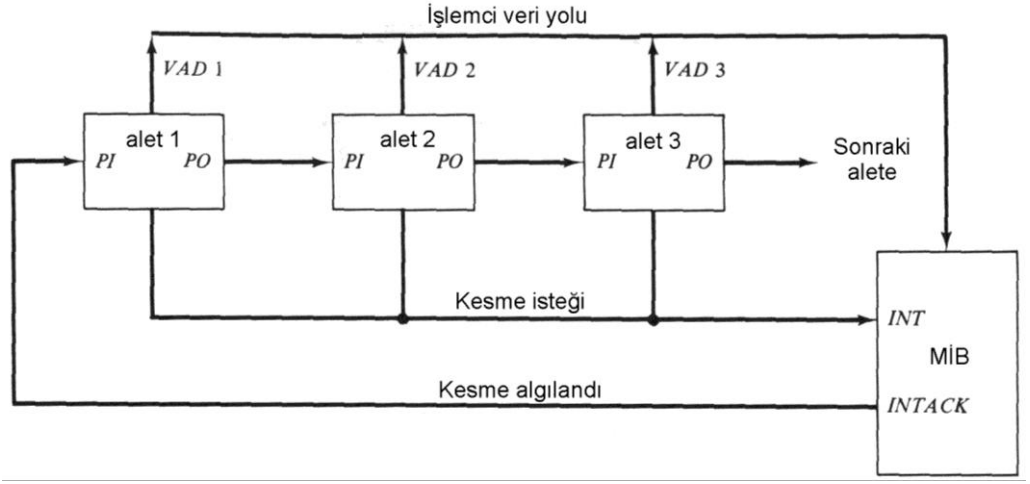
Papatya Zinciri Önceliği

Vektör Adresi (VAD)

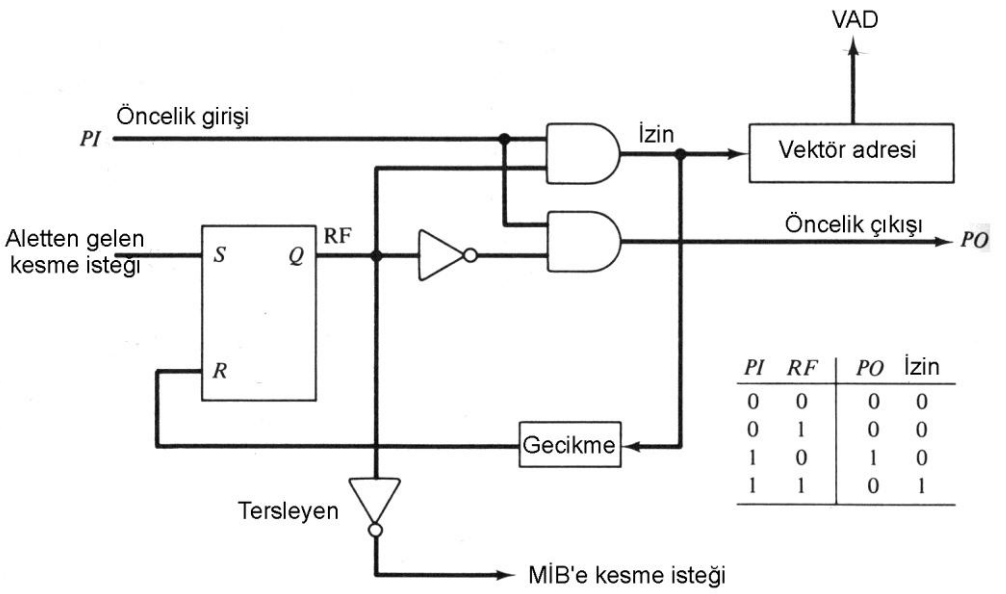
Önceliğin kurulmasında papatya zinciri yöntemi bütün aletlerin seri bağlanmasıyla oluşur. En yüksek öncelikli alet 1 ci sıraya yerleştirilir. Bunu önceliği daha az aletler takip eder. Zincirdeki en son alet önceliği en az

olandır. MİB ile 3 aletin bu şekilde bağlanışı şekil 11.12 de gösterilmektedir. Kesme isteme hattı bütün aletlerde ortaktır ve hatla yapılmış bir mantık devresi eğer, herhangi bir aletin kesme sinyali düşük seviyede ise kesme hattı düşük seviyeye gider ve MİB içindeki kesme girişini izinlendirir. Eğer hiçbir kesme gelmiyorsa kesme hattı yüksek seviyeli durumunda kalır ve MİB hiçbir kesmeyi teşhis etmez. Bu negatif mantıklı VEYA işlemine denktir. MİB, bir kesme işlemine, kesme algılandı hattını izinlendirerek cevap verir. Bu sinyal alet 1 in *PI* girişi tarafından alınır. Algılandı sinyali diğer alete eğer alet 1 kesme istemiyorsa *PO* yoluyla geçer. Eğer alet 1 kesme bekliyorsa algılandı sinyalini tutar. PO çıkışına 0 yollar. Bundan sonra kendi kesme vektörünün adresini (*VAD*) veri yoluna yükleyerek MİB nin kesme sürecinde kullanmasını sağlar.

PI girişinde 0 olan bir alet *PO* çıkışına bir 0 gönderir. Böylece bir sonraki düşük öncelikli alete sinyalin tutulduğunu haber verir. Bir kesme isteyen ve PI girişinde 1 bulunan bir alet algılandı sinyalinin yolunu keser *PO* çıkışına 0 yazar. Eğer alet kesme beklemiyorsa algılandı sinyalini yandaki alete aktarır, yani *PO* çıkışına 1 yazar. Dolayısıyla *PI* = 1 ve *PO* = 0 olan alet en yüksek öncelikli alettir ve kesme istemektedir. Veri yoluna *VAD* sini yazar. Papatya zinciri düzeni en yüksek önceliği MİB den kesme algılandı sinyalini alan alete verir. İlk durumdan en uzaktaki alet en düşük önceliğe sahiptir.



Şekil 11.12 Papatya zinciri öncelikli kesme

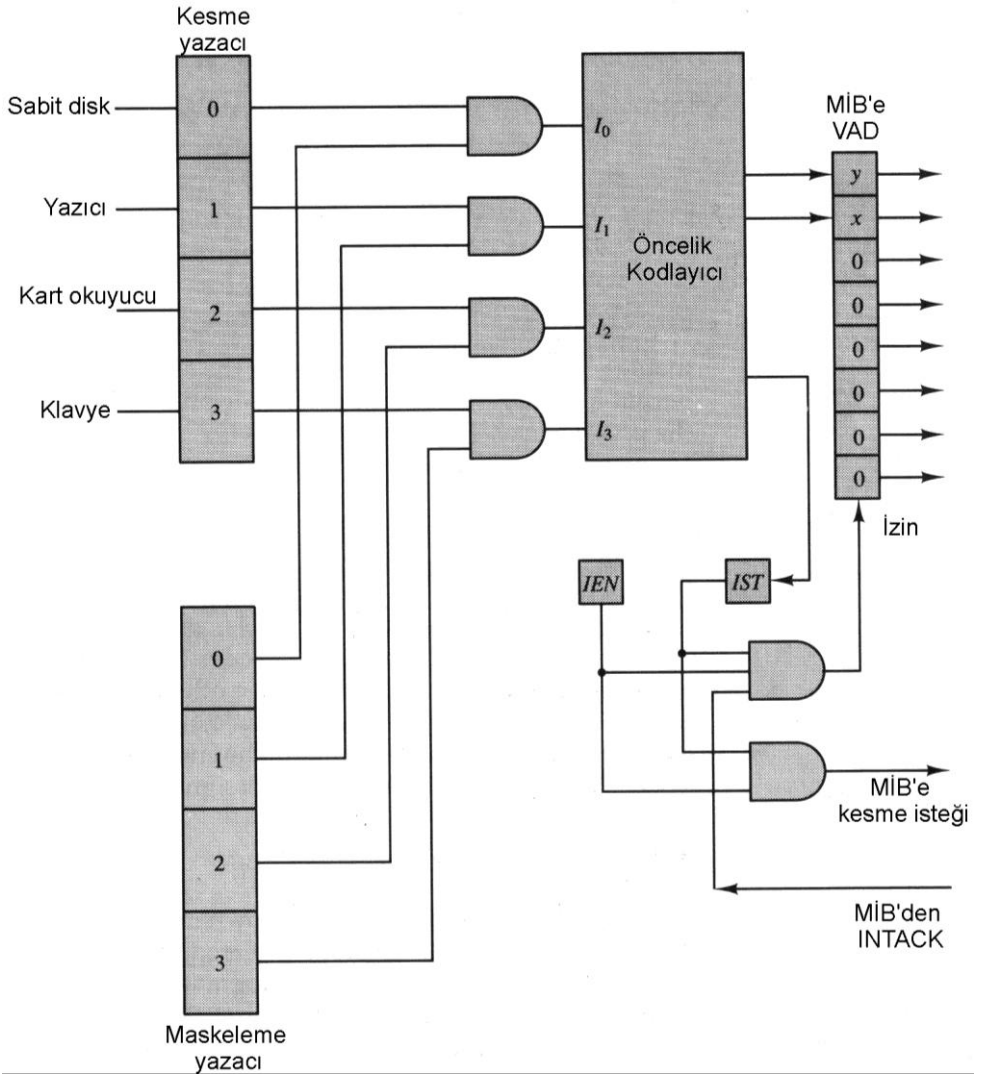


Şekil 11.13 Papatya zinciri öncelik düzenlemesinin bir aşaması

Şekil 11.13 te, papatya zinciri yönteminde bağlanacak her aletin içinde olması gereken mantık devresi görülmektedir. Bu alet *RF* yaz-bozunu MİB ne kesme göndereceği zaman 1 yapar. *RF* yaz-bozunun çıkışı açık toplayıcı-tersleyen üzerinden *PO* ya gider. Bu da ortak kesme hattıdır. Eğer *PI* = 0 ise *PO* ve *VAD* ye giden izin hattı 0 dır. *RF* nin değerine bağlı değildir. Eğer *PI* = 1 ise ve *RF* = 0 ise *PO* = 1 olur vektör adresin izni iptal edilir. Bu şart algılandı sinyalini *PO* üzerinden yandaki alete aktarır. *PI* = 1 iken alet aktiftir ve *RF* = 1 dir. Bu koşul *PO* ya 0 yazar ve vektör adresini izinlendirir. Adres veri yoluna geçer. Her aletin kendi özel vektör adresi olduğu varsayılıyor. MİB nin vektör adresini almasına yetecek bir zaman sonra *RF* 0 yapılır.

Paralel Öncelikli Kesme

Bu yöntemde bir yazaç vardır. Yazacın bitlerinin her biri bir aletten kesme alır. Öncelik bitlerin yazaç içindeki yerine bağlıdır. Kesme yazacına ilave olarak devrede bir maskeleme yazacı vardır. Bunun amacı her bir kesme isteminin durumunu denetim etmektir. Maskeleme yazacı öyle programlanabilir ki yüksek seviyeli kesme servis edilirken, düşük seviyeli kesmeler izni iptal edilebilir. Ayrıca öyle bir devre eklenebilir ki MİB düşük seviyeli bir kesmeyi servis ederken, yüksek seviyeli kesme MİB ni kesebilir.



Şekil 11.14 Öncelik kesme donanımı

Öncelik Mantığı

4 kesme kaynağı olan bir sistem için öncelik mantığı şekil 11.14 te gösterilmiştir. Burada bir kesme yazacı vardır. Bunun bitleri program buyrukları ile temizlenir ve dış şartlarla doldurulurlar. Sabit disk yüksek hızda bir alet olduğundan en yüksek önceliğe sahiptir. Yazıcı ikinci önceliktedir. Bundan sonra karakter okuyucu ve klavye gelir. Maskeleme yazacı kesme yazacı ile aynı sayıda bite sahiptir. Program buyrukları ile maskeleme yazacındaki herhangi bir bitin set veya 0 durumunda edilmesi mümkündür. Her bir kesme biti ve buna karşılık gelen maskeleme biti bir

VE kapısına bağlanarak kod çözücüye 4 giriş sağlanmış olur. Bu yolla bir kesme ancak karşılık gelen maskeleme biti program yardımıyla 1 yapılmışsa tanınır. Öncelik) kod çözücüsü 2 bitlik vektör adresi üretir. Bunu MİB ya aktarır.

Kod çözücüden bir başka çıkış kesme durum (*IST*) yaz-bozudur. Bu yaz-boz maskelenmeyen bir kesme olduğunda doldurulur . Kesme izin yaz-bozu IEN, kesme sistemi üzerinde tam hakimiyet kurulabilmesi için bir program tarafından 1 veya 0 yapılır. *IST* çıkışı, *IEN* ile VE lenerek ortak bir kesme sinyali üretir ve MİB ne gönderir. Kesme *INTACK* sinyalini MİB dan alır ve veri yolu buffer larını izinlendirir ve vektör adresi veri yoluna yerleştirilir.

Öncelik Kodlayıcı

Öncelik kodlayıcısı öncelik fonksiyonunu oluşturan bir devredir. Mantığı aynı anda iki veya daha fazla girdinin gelmesi halinde, girdilerden en yüksek önceliğe sahip olanın öne çıkmasını sağlamaktır. 4 girdili öncelik kodlayıcının doğruluk çizelgesi çizelge 11.2 de verilmektedir. Çizelgedeki × ler dikkate almama şartlarıdır. I_0 girişi en yüksek önceliklidir. Diğer girişlerin değerinden bağımsızdır. Bu giriş 1 olunca çıkışta xy = 00 olur. I_1 bir sonraki öncelik seviyesidir. I_1 = 1 ise çıkış 01 dir. I_0 = 0 dır. Bu değerler diğer iki düşük seviyeli girdiden bağımsızdır. I_2 için diğer iki yüksek seviyeli girdiler 0 ise çıktı oluşturulur. Kesme durum *IST*, bir taneden fazla girdi 1 ise 1 yapılır. Bütün girdiler 0 ise *IST* 0 yapılır, diğer çıkışlar kullanılmaz. Bunlar dikkate almama şartları ile gösterilmiştir. Çünkü *IST* = 0 olunca vektör adresi MİB ye aktarılmaz. Genelde bir bilgisayarın 4 ten fazla kesme kaynağı vardır. 8 bitlik bir kodlayıcının 3 çıkışı vardır.

Öncelik kodlayıcının çıktıları her bir kesme kaynağı için vektör adresinin bir kısmı olarak kullanılır. Vektör adresinin diğer bitleri herhangi bir değer alabilirler. Örneğin vektör adresi kodlayıcının *x*, *y* çıkışına 6 tane 0 eklenerek bulunabilir. Böyle seçilen kesme vektörleri G/Ç aletlerini, 0, 1, 2, 3 ikilik sayıları ile gösterir.

Çizelge 11.2 Öncelik kodlayıcı doğruluk çizelgesi

Girişler				Çıkışlar			
I_0	I_1	I_2	I_3	x	y	*IST*	Boole fonksiyonları
1	×	×	×	0	0	1	
0	1	×	×	0	1	1	$x = I'_0 I'_1$
0	0	1	×	1	0	1	$y = I'_0 I_1 + I'_0 I'_2$
0	0	0	1	1	1	1	$(IST) = I_0 + I_1 + I_2 + I_3$
0	0	0	0	×	×	0	

Kesme Çevrimi

Şekil 11.14 te gösterilen kesme izin yaz-bozu *IEN* program buyrukları ile 1 ve 0 yapılabilir. *IEN* 0 yapıldığında *IST* ten gelen istek MİB tarafından dikkate alınmaz. Program tarafından denetlenen *IEN* biti kullanıcının kesme yeteneğini kullanıp kullanmayacağı konusunda ona yetki sağlar. Eğer *IEN* bitini sıfırlayan bir buyruk programa konursa, kullanıcının bu programın kesilmesini istemediği anlamını taşır. *IEN* in 1 yapılmasını sağlayan bir buyruk kesme yeteneğinin program çalışırken kullanılabileceğini gösterir. Birçok bilgisayar *IEN* i sıfır yapan donanıma sahiptir. Bunun için işlemcinin bunu kabul etmesi yeterlidir.

Buyruk çevriminin sonunda MİB, *IEN* i denetler. *IST* ten gelen sinyale de bakar. Eğer bunlardan biri 0 ise sonraki buyruk ile devam eder. Eğer her ikisi de 1 ise MİB kesme sürecine girer. Kesme süreci boyunca MİB aşağıdaki mikro işlemleri icra eder.

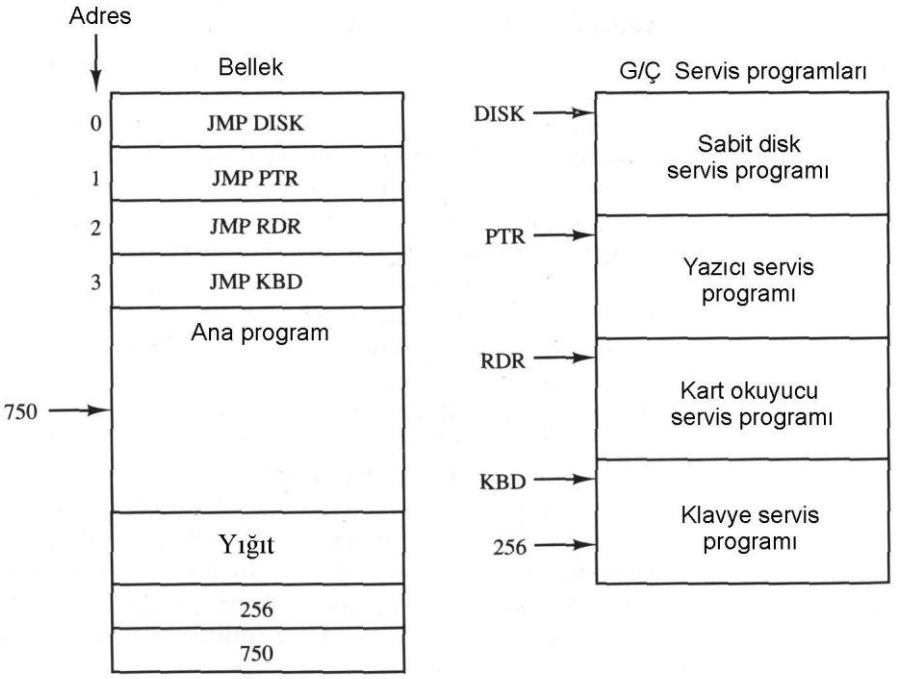
$SP \leftarrow SP\text{-}1$	Yığıt göstergesini azalt
$M[SP] \leftarrow PC$	*PC* yi yığıta it
$INTACK \leftarrow 1$	Kesme alındıyı izinlendir
$PC \leftarrow VAD$	Vektör adresini *PC* ye yaz
$IEN \leftarrow 0$	Gelecek kesmelerin iznini iptal et
Bir sonraki buyruğu al-getir	

MİB dönüş adresini PC dan alıp yığıta yazar. Bundan sonra *INTACK* hattını izinlenlendirerek kesmeyi kabul eder. Kesme önceliği birimi tek kesme vektörünü, MİB veri yoluna yazar. MİB vektör adresini *PC* a yazar ve *IEN* i silerek al-getir evresine girer. Al-getir evresinde buyruk bellekten okunur. Bu buyruk vektör adresinin gösterdiği buyruktur.

Yazılım Yordamları

Servis Programı

Öncelik kesme sistemi yazılım ve donanım tekniklerinin bileşkesidir. Şimdiye kadar donanım kısmı anlatıldı. Bilgisayarın kesme istemlerini servis etmek ve donanımdaki kesme yazaçlarını denetlemek için yazılım yordamlarına gereksinmesi vardır. Şekil 11.15 te bellekte duran ve kesme sistemi kullanan programlar görülmektedir. Her aletin kendi özel servis programı vardır. Bu programa JMP buyruğu ile erişilebilir. JMP buyruğunun vektör adresinde bulunmaktadır. Her yordamın sembolik ismi, servisin başlayacağı adresi gösterir. Şekilde görülen yığıt her bir kesme için dönüş adresini saklamak üzere kullanılır.



Şekil 11.15 Kesmeler için servis programların hafızada depolanması

Özel bir örnek verelim. Klavye, kesme bitini, MİB, 749 numaralı adreste bulunan esas programa ait buyruğu icra ederken 1 yapsın. Buyruk Çevriminin sonunda bilgisayar kesme çevrimine girer. Dönüş adresi 750 yi yığıta yazar ve vektör adresi 00000011 ı veri yolundan alır ve PC a yazar. Bundan sonra 3 numarada bulunan buyruğu icra eder. Denetim KBD yordamına aktarılır. Şimdi sabit diskin kesme bitini, MİB; KBD programının 255 inci buyruğunu icra ederken 1 yaptığını varsayalım. 256 adresi yığıta itilir ve denetim DISK servis programına geçer. Her yordamın son buyruğu kesme buyruğundan dönüştür. Disk servisi tamamlanınca geri dönüş buyruğu yığıttan çeker ve PC a 256 adresini yazar. Bu denetimi KBD yordamına gönderir ve bilgisayar klavye kesmesini servis etmeye devam eder. KBD nin son buyruğu denetimi esas programda 750 adresine gönderir. Böylece daha yüksek öncelikli alet daha düşük öncelikli aleti kesme edebilir. Yüksek seviyeli aletin servis zamanının çok kısa olduğu ve bu zaman içinde düşük seviyeli aletten bilgi kaybı olmadığı varsayıldı.

Başlangıç ve Sonuç İşlemleri

Her bir kesme servis yordamında başlangıç ve sonuç işlemleri grubu bulunmalıdır. Bunlar donanım kesme sistemindeki yazaçları denetlerler. Kesme çevriminin sonunda IEN 0 yapılmaktaydı. Bu yaz-boz daha yüksek öncelikli kesme istemleri için 1 yapılmalıdır. Fakat bundan önce düşük seviyeli kesmelerin izni iptal edilmelidir. Her kesmenin servis yordamının başlangıç dizisinde kesme donanımı, aşağıdaki sırada kullanılmalıdır.

1. Daha düşük seviyedeki maskeleme yazaç bitlerini temizle
2. *IST* Kesme durum bitini temizle
3. İşlemci yazacının içeriğini sakla
4. *IEN* Kesme izin bitini 1 yap
5. Servis yordamını çalıştır.

Kesme edilen kaynağın biti dahil düşük seviyeli maskeleme yazacının bitleri temizlenir. Böylece bu koşulların kesmeyi izinlendirmesi önlenir. Her ne kadar düşük seviyeli kesme kaynakları maskeleme yazacında yüksek numaralı bitler ile gösteriliyorsa da, programcı maskeleme yazacındaki her hangi bir bit konfigürasyonunu kullanarak önceliği değiştirebilir. Kesme durum biti sıfırlanmalıdır. Böylece yüksek seviyeli bir kesme meydana gelirse bu bit 1 yapılabilir. Kesilen program servisten sonra çalıştığında işlemci yazaçlarının içeriklerine gerek duyabilir diye bu yazaçların içerikleri saklanır. Bundan sonra IEN yaz-bozu set edilerek diğer kesmelerin, bilgisayar halen bu kesmeye servis verirken kesme yapmalarına imkan sağlanır.

Her kesmenin servis yordamının son dizisinde, kesme donanımı yöneten buyruklar bulunmalıdır. Bu şu şekilde olur.

1. *IEN* bitini temizle
2. İşlemci yazacına eski içeriğini tekrar yaz
3. Halen servis görmekte olan kaynağa ait kesme yazacındaki biti temizle
4. Maskeleme yazacında düşük seviyeli öncelik bitlerini 1 yap
5. Dönüş adresini *PC* a yaz ve *IEN* i 1 yap

Kesme kaynağına ait kesme yazacındaki bit temizlenmelidir. Böylece aynı kaynağın tekrar kesme isteme durumu için hazır hale getirilmiş olur. Maskeleme yazacı içindeki düşük öncelikli bitler (kesme kaynağının biti dahil) 1 yapılır. Böylece kesmeler izinlendirilmiş olur. Kesilen programa dönüş, dönüş adresinin *PC* a tekrar yazılmasıyla sağlanır. Donanım öyle düzenlenmelidir ki 2 den 5 e kadar olan adımlar sırasında kesme olmamalıdır. Bu adımlar sırasında eğer kesme olursa dönüş adresi kaybolabilir,

maskeleme yada işlemci yazaçlarındaki bilgiler belirsizleşebilir. Bu sebeple *IEN* ilk baştan temizlenir ve dönüş adresi *PC* a yüklendikten sonra aktarılır.

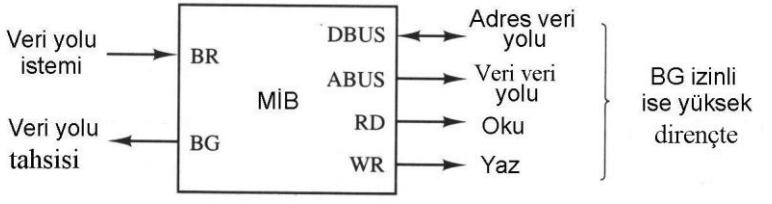
Yukarıda anlatılan başlangıç ve sonuç işlemlerine ek yük veya toparlama adı verilir. Bunlar servis programının parçası değildir. Özeldirler fakat kesmelerin işlemesi için gereklidir. Bütün ek yük işlemleri yazılım tarafından yerleştirilebilir yada donanım tarafından yapılabilir. Bu iş uygun buyrukların servis yordamının başına ve sonuna yerleştirilmesi ile yapılır. Ek yük işlemleri donanım tarafından otomatik olarak yapılabilir. İşlemci yazacının içeriği yığıta donanım tarafından servis rutinine dallanmadan önce aktarılabilir. Diğer başlangıç ve sonuç işlemleri donanıma aktarılabilir. Bu yolla bir kesmenin alınışı ile kaynağa servis yapan buyrukların icrası arasındaki zamanı azaltmak mümkün olabilir.

11.6 Doğrudan Bellek Erişimi (DMA)

Verilerin sabit disk ile bellek arasındaki aktarımı çoğu zaman MİB nin hızıyla sınırlıdır. MİB ni devreden çıkarır ve bir çevre aletinin bellek veri yolunu yönetmesi sağlanırsa aktarma hızı arttırılabilir. Buna DMA denir. DMA sırasında MİB aylaktır ve bellek veri yolu üzerinde hükmü yoktur. DMA denetimleri veri yolların hakimiyetini devralarak, aktarmayı G/Ç aletiyle bellek arasında doğrudan doğruya yapar.

Veri Yolu İstemi

MİB birçok şekilde aylak duruma getirilebilir. Mikro işlemcilerde çok kullanılan bir yöntem, özel denetim sinyalleri ile veri yollarının iznini iptal etmektir. Şekil 11.16 da DMA aktarımını sağlayacak MİB içindeki denetim sinyalleri görülüyor. Veri yolu sistemi (*BR* = Veri yolu isteği) girişi, DMA tarafından yapılır ve MİB nin veri yollarının denetimini bırakmasını ister. Bu giriş aktif olunca MİB icra ettiği buyruğun icrasını keser ve adres veri yolunu, veri veri yolunu ve oku yaz hatlarını yüksek direnç durumuna geçirir. Yüksek empedans durumu açık devre durumu oluşturur ve çıkışların bağlantıları kesilir. Mantıksal etkisi kalmaz.



Şekil 11.16 DMA aktarımı için MİB veri yolu sinyalleri

Veri Yolu Tahsisi

MİB bunun üzerine veri yolu tahsisi çıkışına 1 verir. Böylece DMA veri yollarının yüksek empedans durumunda olduğunu anlar. Veri yolu isteminde bulunan DMA şimdi veri yollarının denetimlerini alabilir ve MİB nin etkisi olmadan bellek aktarımını yapabilir. DMA sona erince veri yolu istemi hattı sıfırlanır. MİB da veri yolu tahsisini sıfırlar, veri yollarının denetimini tekrar alır ve normal görevine döner.

Patlama Aktarımı

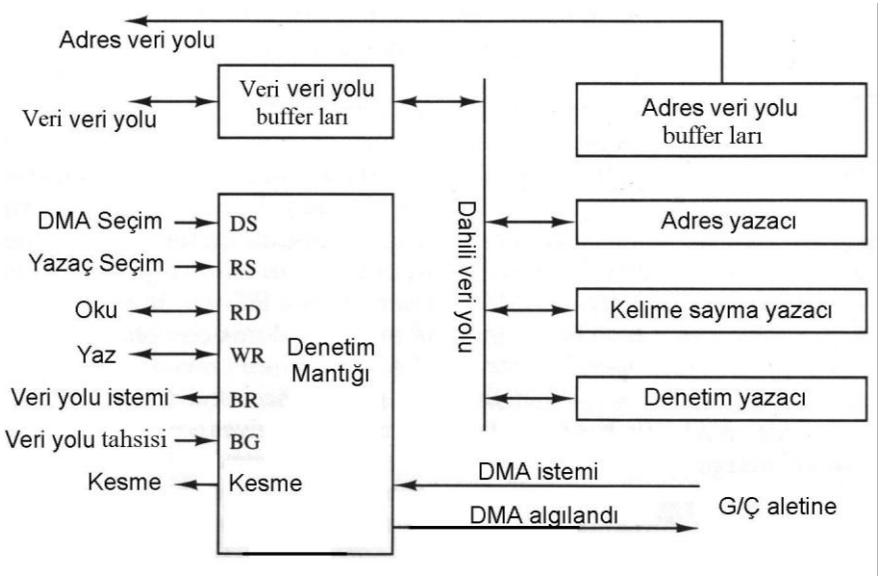
DMA veri yolu sisteminin denetimini aldığında bellekle doğrudan haberleşebilir. Aktarım çeşitli yollarla yapılır. DMA patlama aktarımında, bir miktar kelime dan oluşan blok, sürekli bir patlama şeklinde aktarılır. Bu durumdayken DMA bellek veri yollarına hükmetmektedir. Bu tür aktarımlar sabit disk gibi hızlı aletler için gereklidir. Bunlarda bloğun tamamı aktarılana kadar aktarım durdurulamaz ve yavaşlatılamaz. Alternatif bir yöntem evre çalması. DMA nın bir veri kelimesini veri yollarının denetimini MİB ne geri verdiği sırada aktarmasıdır. MİB bir bellek evresi kadar gecikir ve G/Ç aktarımının yapılmasına izin çalınmasına göz yumar.

DMA Denetim Birimi

DMA denetim birimi, ara yüzeyin normal devrelerinden fazlasını aramaz. Ancak bir adres yazacına, bir kelime soyma sayacına, bir adres hatları grubuna gerek duyar. Adres yazacı ve adres hatları bellekle doğrudan haberleşme için kullanılır. Kelime sayacı aktarılacak kelimelerin sayısını tutar. Veri aktarımı alet ile bellek arasında DMA nın denetiminde doğrudan yapılır.

Şekil 11.17 DMA denetim birimin tipik blok şemasını gösteriyor. Birim MİB ile veri yolu ve denetim hatları yardımıyla haberleşir. DMA içindeki yazaçlar MİB tarafından adres veri yolu yardımıyla seçilir. Bunun için *DS* ve *RS* girişleri izinlendirilir. *RD* ve *WR* girişleri iki yönlüdür. *BG* = 0 iken MİB, DMA yazaçlarıyla veri yolu üzerinden haberleşebilir ve *DMA* yazaçlarını okur veya bunlara yazar. *BG* = 1 ise MİB veri yollarını serbest bıraktığından DMA bellekle doğrudan haberleşebilir. Bunun için adres veri yoluna bir adres yazar ve *RD* veya *WR* denetimlerini çalıştırır. DMA dış çevre aletleriyle onun isteği ve DMA algılandı hatlarıyla yani onaylama yöntemiyle haberleşir.

DMA denetimlerinde 3 yazaç, bir adres yazacı, bir kelime sayacı ve bir denetim yazacı vardır. Adres yazacı bellekte erişilecek yerin adresini tutar. Adres bitleri, veri yolu buffer larından adres veri yoluna geçerler. Adres yazacı belleğe her kelime aktarılışından sonra arttırılır. Kelime sayacı aktarılacak kelimelerin sayısını tutar. Her kelime aktarımından sonra azaltılır ve 0 testi yapılır. Denetim yazacı aktarımın kipini belirler.



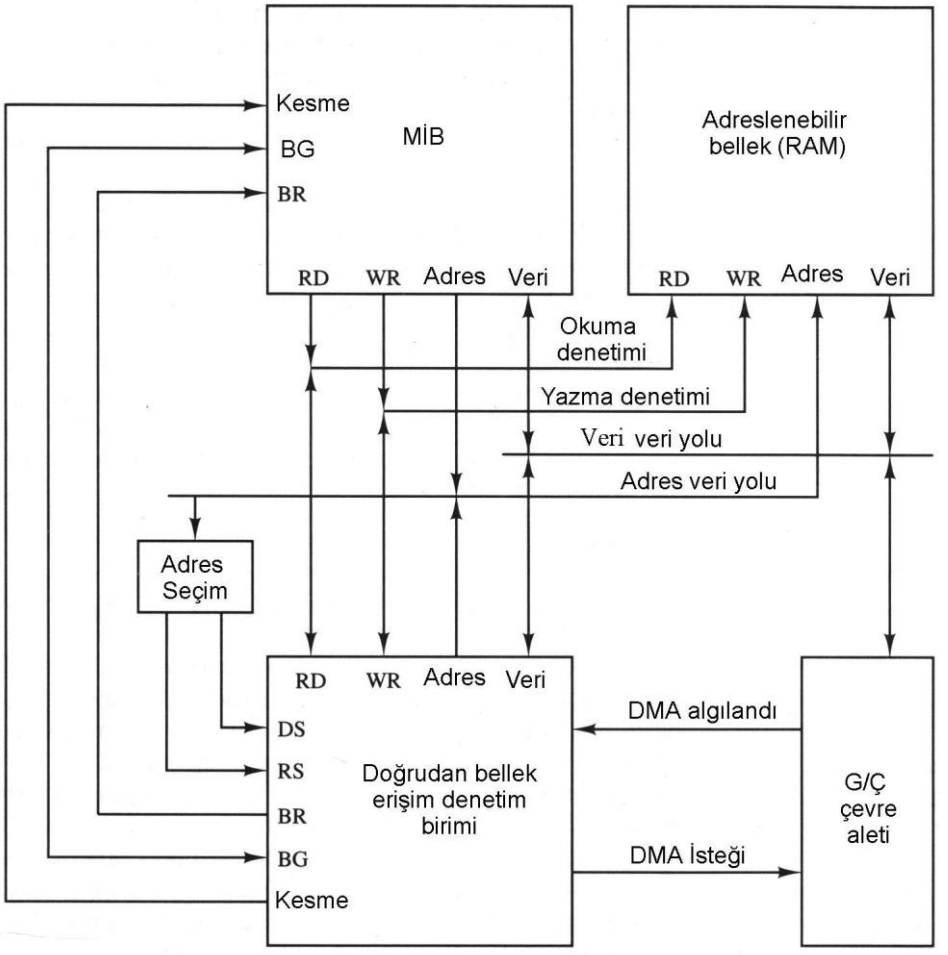
Şekil 11.17 DMA denetim biriminin blok şeması.

DMA içindeki bütün yazaçlar MİB ne G/Ç yazaçları olarak görünür. Dolayısıyla MİB, DMA yazaçlarına veri yolu yardımıyla yazabilir ve aynı yolla bunları okuyabilir.

DMA, MİB tarafından ilklendirilir. DMA çalışıp çevre aleti ile bellek arasında veri aktarımı başlayınca, bloğun tamamı aktarılmadıkça aktarmaya devam eder. İlklendirme işlemi temel olarak bazı DMA yazaçlarını seçen veya seçilecek yazaçların adreslerini içeren G/Ç buyruklarından oluşan bir programdır. MİB, DMA yı aşağıdaki bilgileri veri yolu üzerinden yollayarak ilklendirir.

1. Bellek bloğunun adresi. Burada okunacak veriler vardır veya veriler buradan başlayarak yazılacaktır.
2. Kelime sayısı. Bellek bloğundaki kelime sayısı.
3. Okuma veya yazma yapılacağını belirten denetim.
4. DMA aktarımını başlatan denetim sinyali.

Başlangıç adresi, adres yazacına yazılır. Kelime sayısı, kelime sayacına yazılır. Denetim bilgisi denetim yazacındadır. DMA ilklendirme yapıldıktan sonra, MİB DMA ile haberleşmeyi durdurur. Ancak bir kesme ile MİB, DMA ya erişip kaç kelime aktarıldığına bakabilir.



Şekil 11.18 Bir bilgisayar sisteminde DMA aktarımı

DMA Aktarımı

DMA denetim biriminin bilgisayar sistemindeki diğer bileşenler arasındaki yeri 11.18 de gösterilmektedir. MİB, DMA ile adres ve veri yolları üzerinden haberleşebilir. DMA nın kendi adresi vardır. *DS* ve *RS* hatlarını aktifler. MİB DMA yı veri yolu yoluyla ilklendirir.

DMA bir denetim komutun alınca bellekle çevre aleti arasında aktarım başlar. Çevre aleti DMA istemi gönderdiğinde DMA denetimleri *BR* hattını aktifler. Böylece MİB ne veri yollarını bırakması söylenmiş olur. MİB *BG* hattı ile cevap verir. Buradan DMA, veri yollarının serbest bırakıldığını anlar. Bundan sonra DMA adres yazacındaki değeri adres veri

yoluna yazar. *RD* veya *WR* sinyallerini çalıştırır. Sonra çevre aletine algılandı gönderir. *RD* ve *WR* sinyalleri iki yönlüdür. Aktarımın yönü *BG* hattının durumuna bağlıdır. *BG* = 0 ise *RD* ve *WR* hatları giriş hatları olurlar ve MİB nin DMA nın iç yazaçlarıyla haberleşmesini sağlarlar. *BG* = 1 ise *RD* ve *WR* çıkış hatları olurlar ve DMA dan belleğe giderek okuma veya yazma yapılacağını bildirirler.

Çevre aleti DMA algılandı yı alınca veri yoluna bir kelime yazar veya veri yolundan bir kelime alır. Dolayısıyla DMA okuma ve yazma işlemlerini denetler. Ayrıca bellek adresini de DMA hazırlar. Çevre aleti böylece bellekle veri yolu üzerinden haberleşebilir.

Aktarılan her kelimeden sonra DMA adres yazacını arttırır ve kelime sayacını azaltır. Eğer kelime sayısı 0 olmazsa, DMA çevre aletinden gelen istek hattına bakar. Hızlı bir alet için önceki aktarım tamamlanır tamamlanmaz, hat aktif duruma geçer. İkinci aktarım başlatılır. Sonra işlem tüm blok aktarılana dek sürer. Eğer çevre aletinin hızı az ise DMA istek hattı bir zaman sonra gelir. Bu durumda DMA veri yolu istek hattının izni iptal eder ve MİB kendi programını icraya devam eder. Çevre aleti aktarma isteğinde bulununca DMA tekrar veri yollarını ister.

Eğer kelime sayacı sıfırlanırsa DMA aktarımı durdurur ve veri yolu istemini geri çeker. MİB ne bir kesme ile işlemin bittiğini haber verir. MİB kesme gelince kelime sayacını okur. Bu yazacın sıfır olması bütün kelimelerin başarıyla aktarıldığını gösterir. MİB bu yazacı her zaman okuyarak ne kadar aktarılacak kelime olduğunu anlar.

DMA denetimlerinin birden fazla kanalı olabilir. Bu durumda her kanalın istek ve algılandı çifti mevcut olup, farklı çevre aletlerine bağlanmıştır. Her kanalın kendi özel adres yazacı ve kelime sayacı DMA denetim birimi içinde mevcuttur. Kanallar arasında öncelik kurulabilir ve öncelikli olana önce hizmet verilebilir.

DMA aktarımı birçok uygulamalarda çok faydalıdır. Bellekle sabit disk arasında hızlı bilgi aktarımında kullanılır. Etkileşimli uç birim ekranlarının güncellenmesinde de kullanılır. Tipik olarak ekrandaki görüntü bellekte tutulur ve program denetiminde değiştirilir. Belleğin içeriği periyodik olarak DMA üzerinden uç birime aktarılır.

11.7 Giriş / Çıkış İşlemcisi (GÇİ)

Her bir ara yüzeyin ayrı ayrı MİB ile haberleşmesi yerine, bilgisayar bir veya daha fazla işlemci kullanabilir. Bunlara G/Ç aletleriyle doğrudan doğruya haberleşme görevi verilebilir. Bir G/Ç işlemcisi, DMA yeteneği olan ve G/Ç aletleriyle doğrudan haberleşebilen bir işlemci olarak tanımlanabilir. Bu konfigürasyonda bilgisayar sistemi; bellek birimi, MİB ni

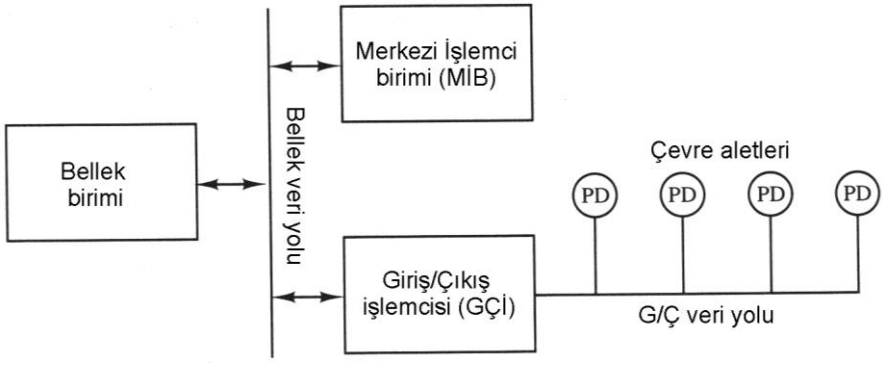
oluşturan birkaç tane işlemci ve bir veya daha fazla GÇİ. Her bir GÇİ, G ve Ç görevlerini yapar, MİB ni G/Ç işlemlerinden ayırır. Telefon veya başka yöntemlerle uzak uç birimlere haberleşebilen bir işlemci, bu haberleşmeyi seri biçimde yapar. Buna veri haberleşme işlemcisi (VHİ) adı verilir.

G/Ç İşlemleri

GÇİ MİB ne benzer. Ancak G/Ç işlemlerini yapabilecek şekilde tasarlanmıştır. Tamamen MİB tarafından yönetilen DMA dan farklı olarak, GÇİ kendi buyruklarını alıp getirebilir ve icra edebilir. GÇİ buyrukları özel olarak G/Ç aktarımlarını yapabilecek şekilde tasarlanmışlardır. Buna ek olarak GÇİ aritmetik, mantıksal, dallanma ve kod aktarımı gibi diğer bazı işlemleri de yapabilir.

Şekil 11.19 da iki işlemcili bir bilgisayarın blok şeması vardır. Bellek birimi merkezi bir durumda bulunur ve her iki işlemci ile haberleşebilir. Bunun için DMA kullanır. MİB bilgisayarın yapması gereken işlemlerden sorumludur. GÇİ işlemcisi ise bellekle muhtelif çevre aletleri arasında veri aktarımı için bir yol sağlar. MİB genellikle G/Ç programını başlatır. Fakat o andan sonra GÇİ, MİB dan bağımsız olarak çalışır ve dış birimden belleğe veri aktırır.

Çevre aletlerinin veri biçimleri, bellek ve MİB nin veri biçiminden farklıdır. GÇİ birçok farklı kaynağın veri kelimelerini bir yapıya sokmalıdır. Örneğin giriş biriminden 4 byte olarak bunları 32 bit kelime haline getirdikten sonra belleğe aktarmak gerekebilir. Veriler GÇİ ye aletin hızıyla gelir, MİB kendi programını icra etmektedir. Verinin bir bellek kelimesinde birleştirilmesinden sonra bunlar GÇİ den belleğe doğrudan doğruya bir MİB zaman çevrimi çalınarak gönderilir. Benzer şekilde bir çıkış kelimesi bellekten GÇİ ye aktarıldıktan sonra GÇİ den çıkış aletine yönlendirilir ve bu alete, alet hız oranı ve bit kapasitesiyle verilir.



Şekil 11.19 G/Ç işlemcisi ile bir bilgisayarın blok şeması

GÇİ ve ona bağlı aletler arasındaki haberleşme, aktarımın programla denetimine benzer. Bellek ile haberleşme ise DMA yöntemine benzer. MİB ile GÇİ un haberleşmesi aletin gelişmişlik seviyesine bağlıdır. Çok büyük çaplı bilgisayarlarda her bir işlemci diğerlerinden bağımsızdır ve işlemcilerden herhangi biri bir işlem başlatabilir. Bir çok bilgisayar sistemlerinde ise MİB hakimdir. GÇİ ler ise köledir. Bütün işlemleri başlatma görevi MİB ne verilmiştir. Fakat G/Ç işlemleri GÇİ ler içinde icra edilir. MİB buyrukları G/Ç aktarımını başlatmak G/Ç durum şartlarını test etmekte işe yarar. Durum şartları muhtelif G/Ç işlemlerinde karar verebilmek için gereklidir. GÇİ ise bir kesme göndererek MİB nin dikkatini çeker. Aynı zamanda bir durum kelimesinin belli bir adrese yollanmasını isteyen MİB nin isteğini yerine getirir. MİB bu durum kelimesini daha sonra inceler. Bir G/Ç işlemi istendiğinde MİB, GÇİ yi bilgilendirir. GÇİ bununla programını bulur. MİB bundan sonra aktarım ayrıntılarını GÇİ ye bırakır.

Komutlar

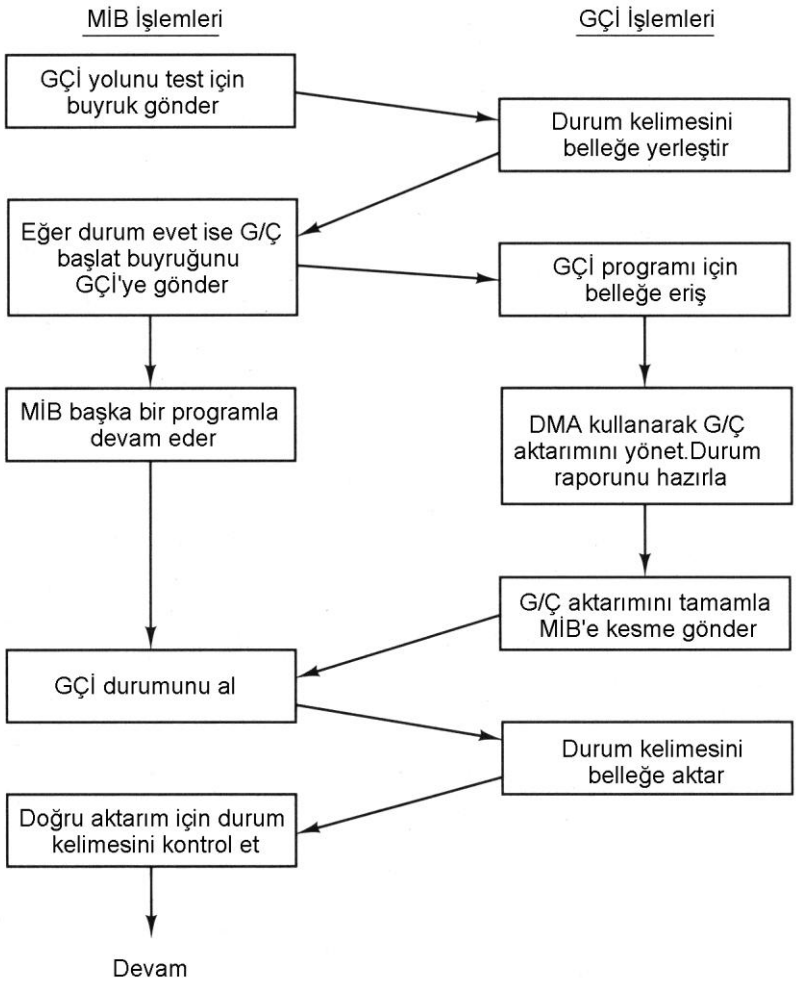
Bellekten bir GÇİ tarafından okunan buyruklara bazen komut adı verilir. Bunun nedeni MİB tarafından okunanları GÇİ tarafından okunanlardan ayırmaktır. Komutlar deneyimli programcılar tarafından hazırlanır ve belleğe yerleştirilir. Komut kelimesi GÇİ için bir program oluşturur. MİB, GÇİ ye bellekte komutları nerede bulacağını bildirir. Böylece G/Ç programı icra edilmiş olur.

MİB-GÇİ Haberleşmesi

MİB ile GÇİ arasındaki haberleşme ele alınan bilgisayara göre çok değişik biçimler alabilir. Çoğu durumda bellek bir haberleşme merkezi durumundadır ve her bir işlemci diğerine ulaşmasını istediği bilgiyi buraya yazar. Tipik bir GÇİ haberleşmesi bir örnekle gösterilirse işlemlerin sırası şekil 11.20 deki akış şemasında olduğu gibi yürür.

MİB, GÇİ yolunu test etmek için bir buyruk yollar. GÇİ buna cevap olarak bir durum kelimesi gönderir. Bu kelime MİB tarafından incelenecektir. Kelimenin bitleri GÇİ ve G/Ç aletlerinin durumlarını göstermektedir. Örneğin GÇİ çok yüklü, alet başka bir aktarımla uğraşıyor veya alet G/Ç aktarımı için hazır gibi. MİB bellekteki durum kelimesine bakarak bundan sonra ne yapacağına karar verir. Her şey yolunda ise MİB bir buyruk göndererek G/Ç aktarımını başlatır. Bu buyruk ile alınan bellek adresi GÇİ ye programı nerede bulacağını gösterir.

Şimdi MİB başka bir programı çalıştırır ve GÇİ meşgul olduğu sürece bu devam eder. Her iki programda DMA aktarımı ile bellekle irtibat halindedir. GÇİ işini bitirince MİB ne bir kesme istemi yollar. MİB bu kesmeye karşılık olarak GÇİ deki durum kelimesini belli bir bellek



Şekil 11.20 MİB-GÇİ Haberleşmesi

adresine yazar. Durum kelimesi aktarımın tamamlandığını veya aktarım sırasında hata olduğunu gösterir. Durum kelimesinin bitlerinin denetiminden sonra MİB, G/Ç işleminin hatasız olarak tamamlandığına karar verir.

GÇİ, birçok birimle bellek arasındaki veri aktarımlarına göz-kulak olur. Bu arada MİB başka bir programı icra etmektedir. GÇİ ile MİB belleği kullanmak için yarışırlar.Bu yüzden işlemde olan aletlerin belleği kullanma süreleri sınırlıdır. Belleği tamamen çevre birimlere vermek mümkün değildir. Çünkü bu birimler MİB den çok daha yavaştırlar. Çok hızlı birimler örneğin sabit disk daha hızlı bellek erişimi yapabilir. Fakat

bu durumda MİB nin çalışması yavaşlar. Çünkü belleğe erişmek için GÇİ yi beklemek zorunda kalır.

IBM 370 G/Ç Kanalı

IBM 370 teki G/Ç işlemcisine bir kanal adı verilir. Tipik bir bilgisayar sistemi birçok kanala sahiptir. Bu kanalların her biri bir veya daha fazla G/Ç aletine bağlıdır. 3 tip kanal vardır. *MUX*, *seçici* ve *blok-MUX*. *MUX* kanalı birkaç tane yavaş veya orta hızda alete bağlanmıştır ve bu aletleri eş zamanlı çalıştırabilir. *SEÇİCİ* kanal bir anda sadece bir tane G/Ç işlemi yapar ve genellikle tek bir yüksek hızda aleti çalıştırmakta kullanılır. *BLOK-MUX* kanalı ise bu iki kanalın özelliklerini bir araya getirir. Birkaç tane yüksek hızda aletin bağlanmasına imkan verir. Bütün G/Ç aktarımları MUX kanalıyla mukayese edildiğinde tek bir veri bloğu ile bağlantılıdır ve bir zamanda bir byte aktarım yapar.

MİB kanallarla, bu iş için ayrılmış denetim hatları ile doğrudan haberleşir. Bazen de bellekte ayrılmış alanlar yardımıyla haberleşir. Şekil 11.21 kanal operasyonu ile ilgili kelime biçimlerini gösteriyor.

G/Ç buyruk biçiminde 3 alan var. İşlem kodu kanal adresi ve alet adresi. Bilgisayar sisteminin birçok kanalı bulunabilir. Her birinin bir adresi vardır. Benzer biçimde her kanal birkaç tane alete bağlı olabilir ve her aletin bir adresi vardır. İşlem kodu, 8 tane G/Ç buyruklarından bir tanesini gösterir. G/Ç ı başlat, G/Ç hızlı serbest bırakmayla başlat, G/Ç ı test et,



Şekil 11.21 IBI 370 G/Ç ile ilişkili kelime biçimi

G/Ç ı sil, G/Ç ı durdur, aleti durdur, kanalı test et ve kanalı belirten tanımları depola. Adresi verilen kanal, G/Ç buyruklarına cevap verir ve buyruğu icra eder. Ayrıca PSW yazacına (MİB içindeki) 4 koşul kodundan birini 1 yazar. Buradan MİB işlemin sonucunu anlar. 4 koşulun anlamı her G/Ç buyruğu için farklıdır. Fakat genel olarak, kanal veya aletin meşgul olup olmadığını, çalışıp çalışmadığını, kesmelerin bekleyip beklemediğini, G/Ç işleminin başarıyla başlayıp başlamadığını ve kanal tarafından belleğe bir durum kelimesi yerleştirilip yerleştirilmediğini belirtir.

Kanal durum kelimesinin biçimi şekil 11.21 (b) de gösterilmiştir. Bu daima belleğin 64 adresinde yer alır. Anahtar alanı bir koruma yöntemidir ve yetkisiz kullanımı önler. Böylece bir kullanıcı başka kullanıcıya ait veya işletim sistemine ait bilgileri kullanamaz. Durum kelimesi içinde adres kısmında son komutun adresi bulunmaktadır. Bu komut kanal tarafından icra edilen son komuttur. Sayaç kısmı aktarım durduğunda kalanı verir. Eğer aktarım başarıyla bitmişse sayaç alanında sıfır bulunur. Durum alanı kanal ve aletteki durumları ve aktarım sırasındaki hataları gösterir.

İki tane başlama komutu var. *G/Ç ı başlat* ve *G/Ç hızlı serbest bırakmayla başlat*. Bunların farkı ikinci komutun daha az MİB zamanı istemesidir. Kanal bu iki buyruktan birini alınca, bellek adresi 72 ye giderek kanal komut kelimesinin (CCW) adresini almak ister. CCW nin biçimi şekil 11.21 de verilmiştir. Veri adres alanı, bellek buffer ının ilk adresidir ve sayaç alanı, aktarıma girecek byte sayısıdır. Komut alanı G/Ç işlemini belirler ve bayrak bitleri kanal için ilave bilgiler içerirler. Komut alanındaki işlemler 6 temel tip G/Ç işleminden biridir. Bu işlemler

1. Yazma*;* Verinin bellekten G/Ç aletine aktarılması
2. *Okuma;* Verinin G/Ç aletinden belleğe aktarılması
3. *Geriye okuma;* Teybin geriye doğru sarılması
4. *Denetim;* Veri aktarımı ile ilgili olmayan bir işlemin başlatılması. Bunlar teybin geriye sarımı veya diske erişim mekanizmasının yerleştirilmesi gibi işlemlerdir.
5. *Algılama;* Kanalın, kanal durum kelimesini 64 adresine göndermesini söyler
6. *Kanalın aktarımı;* JMP buyruğu yerine kullanır. Burada veri adres alanındaki adres, bir sonraki komut kelimesinin adresidir. Bu komut kanal tarafından icra edilecektir.

Kanal programının bir örneği çizelge 11.3 te gösterilmiştir. Bunda 3 komut kelimesi vardır. Birincisi belleğin 4000 adresinden 60 byte ın manyetik teybe aktarılmasına neden olur. Diğer iki komut kelimesi belleğin farklı bölgeleri ve farklı sayıda byte ları içindir. Her bir denetim ke-

Çizelge 11.3 IBM-370 Kanal Program Örneği

Komut	Adres	Bayraklar	Sayma
Teybe yazma	4000	100000	60
Teybe yazma	6000	010000	20
Teybe yazma	3000	000000	40

limesindeki 6 bayrak, komut kelimeleri arasındaki iç bağlantıları belirler. Birinci komut kelimesindeki 1 yapılmış olan birinci bayrak "veri zinciri"ni belirtir. Bunun sonucu olarak ilk komut kelimesinden gelen 60 byte, takip edenin 20 byte ı ile 80 byte lık bir kayıt oluşturur.

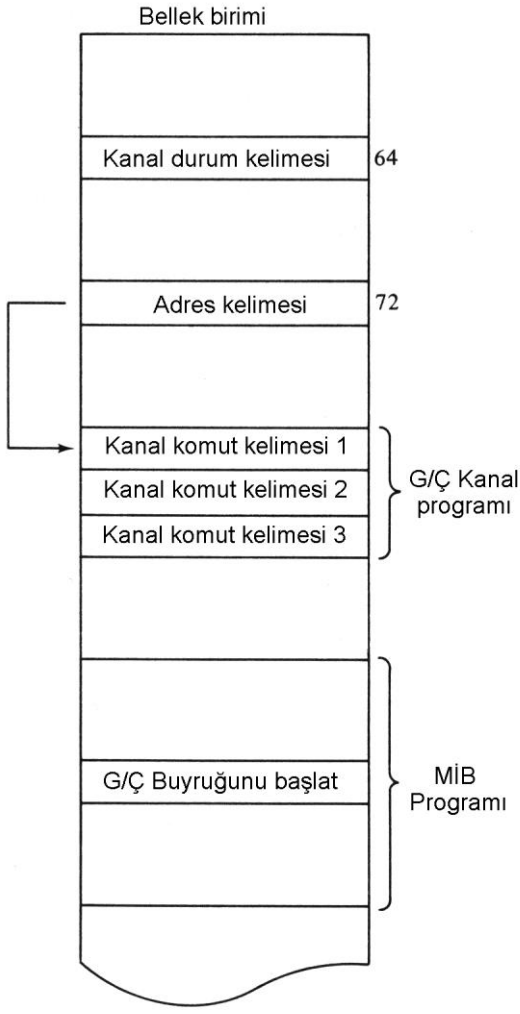
İki farklı bellek adresinden gelen bu veriler teyp üzerinde ayırma işareti veya boşluk olmadan birleşirler. İkinci komutta 1 yapılmış olan ikinci bayrak, *komut zinciri* anlamını taşır. Bundan sonraki komutun aynı G/Ç aleti tarafından kullanılacağını kanala haber verir. Burada alet teyptir. Kanal teybe, bir ara boşluğu yazmasını söyler ve sonra bir sonraki komut kelimesini bellek okumak için harekete geçirir. Üçüncü komutun 40 byte ı teybe ayrı bir kayıt olarak yazılır. Bayraklar 0 olduğunda belli bir alet için G/Ç işleminin sona erdiğini belirtir.

Burada geçen bilgilerin bulunduğu bellek haritası şekil 11.22 de verilmiştir. İşlem MİB programı başla G/Ç buyruğuna gelince başlar. GÇİ bundan sonra bellek adresi 72 ye giderek kanal adres kelimesini alır. Bu kelime G/Ç kanal programının başlangıç adresini gösterir. Bundan sonra kanal adresi verilen programı icraya başlar. Kanal, aktarım sırasında bir durum kelimesi oluşturur ve 64 konumuna bunu aktarır. Kesmeye bağlı olarak MİB 64 adresine giderek durum kelimesine bakar.

Intel 8089 GÇİ

Intel 8089 GÇİ 40 ayaklı IC paketi içinde bulunur. 8089 içinde kanal adı verilen iki bağımsız birim bulunur. Her bir kanal DMA denetim özelliklerini taşıyan bir kısımla işlemci özelliklerini taşıyan bir kısmın birleştirilmesinden oluşur. 8089 birimi, MİB 8086 olan mikro bilgisayar sisteminde GÇİ görevi yapmak üzere tasarlanmıştır. 8086 işlemcili MİB bellekte ve icra edilecek fonksiyonu belirten bir mesajla G/Ç işlemini başlatır. 8089 GÇİ mesajı bellekten okur, işlemi yapar, bitirdiğinde MİB ne haber verir.

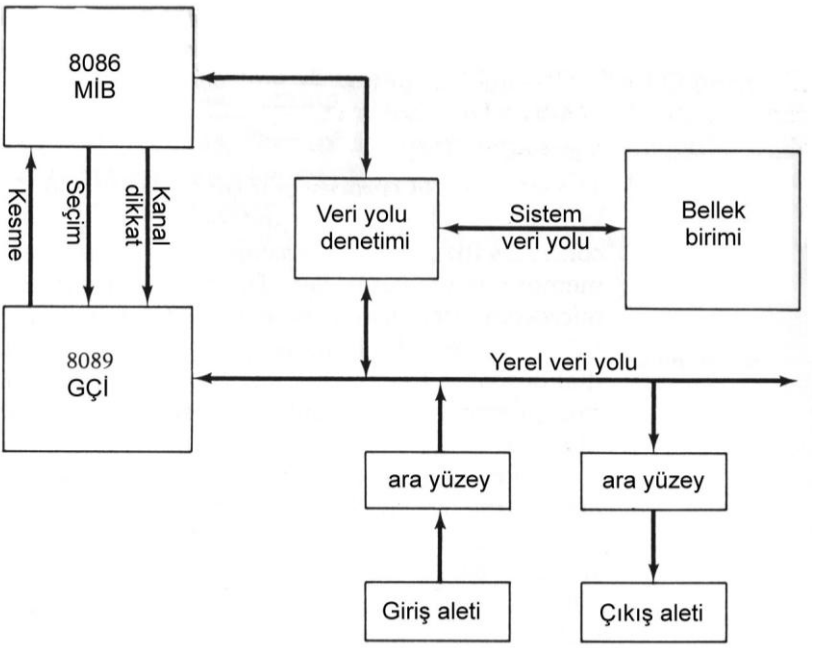
8089 GÇİ nin 50 temel buyruğu vardır. 370 in ise 6 temel G/Ç komutu vardır. 8089 GÇİ nin buyrukları tek tek bitler, byte lar veya 16 bit kelime üzerinde çalışabilir. GÇİ tıpkı MİB gibi programlar icra edebilir. Fakat GÇİ nin buyruk kümesi özel olarak etkin biçimde G/Ç işlemleri



Şekil 11.22 IBM 370 için G/Ç işlemlerinin bellekte yerleşimi

yapabileceği şekilde seçilmiştir. Buyruk kümesi genel veri aktarım buyruklar; temel aritmetik ve mantıksal buyruklar, şartlı veya şartsız dallanma işlemleri, alt yordam çağırma ve dönüş yeteneği veren buyruklar içerir. Bu küme içinde ayrıca DMA yı başlatacak ve MİB ne kesme istemi yollayacak özel buyruklarda vardır. Bu küme sayesinde sistem veri yoluna bağlı iki bileşen etkin veri aktarımı yapabilir. Örneğin G/Ç ile bellek, bellek ile bellek, G/Ç ile G/Ç gibi aktarımlar yapılabilir.

Intel 8086-8089 entegre devre çiftini kullanan bir mikro bilgisayar şekil 11.23 te gösterilmiştir. 8086 nın MİB olarak ve 8089 un GÇİ olarak

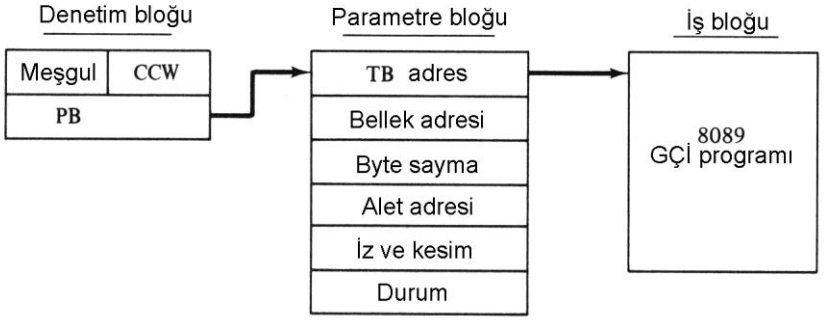


Şekil 11.23 Intel 8086/8089 mikro bilgisayar sistem blok şeması

görev yaptığı bilinmektedir. Bu iki birim, intel tarafından çoklu veri yolu adı verilen bir sistem veri yoluna bağlanmıştır. Bellek ise bir veri yolu denetim birimi yardımıyla veri yoluna bağlıdır ve ortak bellek olarak iki işlemci tarafından paylaşılır. GÇİ bir yerel veri yolu yarımıyla, G/Ç aletlerine bağlı ara yüzlere bağlıdır. MİB, GÇİ ile haberleşmek için kanal dikkat hattını izinlendirir. 8089 içindeki iki kanaldan birini seçmek için seçim hattını kullanır. GÇİ, MİB nin dikkatini çekmek için kesme istemi gönderir.

MİB ile GÇİ sistem belleğine mesajlar yazarak birbirleriyle haberleşebilirler. MİB mesaj alanını hazırlar ve GÇİ ye kanal dikkat hattını izinlendirerek sinyal gönderir. GÇİ mesajı okur, istenen G/Ç işlemini yapar ve uygun kanal programını çalıştırır. Kanal programını bitirince, MİB ya bir kesme istemi gönderir. Haberleşme için blok adı verilen program kısımları kullanılır. Şekil 11.24.

Her bir blok denetim ve parametre bilgileri, takip eden bloğun adres göstergesini de beraberinde taşır. Denetim bloğunun adresi her bir denetim kanalına ilklendirme sırasında verilir. Meşgul bayrağı GÇİ nin meşgul mü olduğu yoksa yeni bir G/Ç işlemi yapmaya hazır mı olduğunu



Şekil 11.24 Intel 8086/8089 bilgisayar sisteminde G/Ç işlemleri için bilginin hafızadaki yerleşimi.

gösterir. GÇİ ye gereken işlemin tipini belirleyen MİB bunu CCW ye yazar. 8089 içindeki CCW, IBM içindeki CCW ile aynı anlamı taşımaz. Burada CCW G/Ç buyruğuna benzer. GÇİ için bir işlem gösterir. Örneğin başlama, bekleme, geri dönüş işlemi, G/Ç programını durdurma gibi.

Parametre bloğu, GÇİ programı kendi görevini yaparken kullanmak zorunda olduğu değişken verileri içerir. Görev bloğu ise GÇİ içinde icra edilecek gerçek programı taşır. MİB ve GÇİ, denetim ve parametre blokları üzerinden beraberce çalışırlar. MİB, meşgul bayrağını denetleyerek GÇİ nin serbest olup olmadığını anlar. Sonra ortak belleği kullanır ve parametre bloğuna bilgi doldurur. CCW ye ise bir başlama komutu yazar. Haberleşme blokları dolunca, MİB kanal dikkat hattını izinlendirerek, GÇİ ye G/Ç işlemini başlatmasını söyler. Sonra MİB başka bir programla çalışır. GÇİ kanal dikkat sinyaline, denetim bloğunun adresini kendi program sayacına yazarak cevap verir. Bundan sonra meşgul bayrağını 1 yapar. CCW içindeki işlemi denetler. Bundan sonra PB ve TB adresleri GÇİ nin iç yazaçlarına aktarılır. GÇİ programı icraya başlar. Program TB içindedir ve bilgi PB içindedir. PB içindeki girişler G/Ç aletine bağlıdır. Parametreler şekil 11.24 te verilmiştir. Bunlar sabit diskten veya sabit diske veri aktarımı için gerekli olanlardır. Bellek adresleri, bellek buffer ının başlangıç adresidir. Byte sayısı aktarılarak byte sayısıdır. Alet adresi, kullanılacak G/Ç aletinin adresidir. İz ve sektör numaraları verinin sabit disk üzerinde yerini saptamaya yarar. G/Ç işlemi tamamlanınca GÇİ, durum bitlerini parametre bloğundaki durum kelime için ayrılan yere yerleştirir ve MİB ne kesme gönderir. MİB, aktarımın başarıyla tamamlanıp tamamlanmadığını anlamak için durum kelimesine bakar.

11.8 Seri Haberleşme

Veri haberleşme işlemcisi (VHİ) uzak uçlardan uç birimlerden telefon veya başka araçlarla yapılan haberleşme ağından bilgi toplayan veya yollayan bir G/Ç işlemcidir. Özelleştirilmiş bir G/Ç işlemcisi olup, haberleşme ağlarıyla doğrudan haberleşebilir. Bir haberleşme ağında çok değişik aletler bulunabilir. Örneğin yazıcılar, etkileşimli görüntü birimleri, sayısal algılayıcılar ve uzaktan bilgi işlem aletleri. VHİ nün kullanılmasıyla, bilgisayar ağın parçalarına hizmet verebilir. Bunu öyle gerçekleştirebilir ki ağın her parçası aynı anda hizmet veriyormuş gibi olur. Bu biçimde bilgisayar zaman paylaşımlı ortamda çalışma imkanı bulur.

Veri Haberleşme İşlemcisi

GÇİ ve VHİ arasındaki en büyük fark, bunların G/Ç aletleriyle haberleşme yöntemidir. Bir G/Ç işlemci çevre aletleriyle, veri ve denetim hatlarından oluşan ortak G/Ç veri yolu üzerinden haberleşir. Bütün çevre aletleri ortak veri yolu kullanırlar ve G/Ç işlemcisine veya G/Ç işlemcisinden, ortak veri yolu ile bilgi aktarımı yaparlar. VHİ her bir uç birim ile tek bir çift hat üzerinden haberleşir. Bütün veri ve denetim bilgileri seri olarak aktarılır. Bunun sonucu olarak aktarım oranı çok yavaştır. VHİ nin görevi sayısal bilgileri her bir uca göndermek veya toplamaktır. Bu işlem bilgilerin veri mi yoksa denetim mi olduğunu belirler. Bütün isteklere daha önceden belirlenmiş ve hazırlanmış işlemlerle cevap verir. VHİ doğal olarak MİB ve bellekle de G/Ç işlemcisi şeklinde haberleşmek zorundadır.

Modem

Uzak uçların VHİ ne bağlanması telefon telleri ve diğer özel haberleşme yollarıyla olur. Telefon telleri ses haberleşmesi için yapılmıştır. Bilgisayar ise sayısal sinyallerle haberleşmekte bu yüzden dönüştürücüler *modem* kullanılmaktadır. Bir modem sayısal sinyalleri ses tonlarına dönüştürerek telefon hatlarından yollar veya ses tonlarını sayısal sinyallere dönüştürür. Bir çok modülasyon çeşitleri ve farklı düzeylerde haberleşme ortamları mevcuttur. Bir haberleşme teli eş zamanlı veya eş zamanlı olmayan ara yüze bağlanabilir. Bu uzak ucun aktarma yöntemine bağlıdır. Eş zamanlı olmayan çalışan bir ara yüzey seri veriyi her bir karakter için başlangıç-bitiş biti ile birlikte alır. Şekil 11.7. Bu tip bir ara yüzey eş zamanlı olmayan çalışan ara yüze benzer. (Şekil 11.8)

Eş zamanlı haberleşme karakterleri çerçevelemek için başla-bitiş bitleri kullanmaz. Dolayısıyla haberleşme bağlantısı daha etkin kullanılır. Yüksek hıza sahip aletler eş zamanlı haberleşmeyi kullanırlar. Eş zamanlı aktarımda kullanılan modemlerin iç saatleri vardır ve bu saat haberleşme hattında akan bitlerin frekansına eşit frekansa getirilir. Uygun bir aktarım için aktarıcı ve alıcı modemlerin saatlerinin daima eş zamanlı durumda

kalması gerekir. Haberleşme hattında sadece veri bitleri vardır ve saat bilgisi bu bitlerden elde edilmelidir. Alıcı modemde eş zamanlılık, gelen verideki geçişler yardımıyla elde edilir. Gönderici ve alıcı saatleri arasında meydana gelebilecek frekans kayması, alıcı saatinin sürekli olarak gelen bitlerin frekansına ayarlanmasıyla giderilir. Modem alınan veriyi saatle birlikte ara yüze gönderir. Gönderici tarafındaki uç veya ara yüzey de kendi modeminden gelen saat bilgisini kullanmaktadır. Bu yolla gönderici ve alıcı aynı oranda tutulur.

Eş zamanlı olmayan aktarıma ters olarak, eş zamanlı aktarım, eş zamanlılığı bozmamak için sürekli olarak mesaj göndermek zorundadır. Mesajda veri bloğu gibi aktarılan bir grup bit vardır. Bütün blok, bloğun başında ve sonunda özel denetim karakterleri ile aktarılır. Bloğun başındaki denetim karakteri, gelen bit katarının karakterlere ayrılmasını sağlayarak bilgiyi verir.

VHİ nın önemli görevlerinden biride, aktarma hatalarını denetlemektir. Hata, eşlik bitinin denetimi ile bulunabilir. Eşlik alınan her karakter için denetlenir.

Blok Aktarımı, CRC

Eş zamanlı aktarımda bütün bir blok aktarılır. Her karakterde bir eşlik biti mevcuttur. Tüm blok gönderildikten sonra gönderici bir fazla karakter gönderir. Bu bütün mesaja ait eşlik tir. Bu karaktere boylamasına artıklık denetim (LRC) adı verilir ve aktarılan tüm karakterlerin ÖZEL-VEYA larının birikimidir. Alan istasyon karakterleri aldıkça LRC yi hesaplar ve gönderilen LRC ile karşılaştırır. Hatasız mesajlarda her iki LRC eşit olmalıdır. Eğer alıcı gönderilen mesajda bir hata bulursa göndericiye aynı mesajı tekrar göndermesini söyler. Aktarmada hataların denetlenmesi için bir başka yöntem dönüşsel artıklık denetimi (CRC) dir.

Veriler iki nokta arasında 3 farklı kipte gönderilebilir; tek yönlü, yarı çift yönlü, tam çift yönlü, tek yönlü hattı bilgiyi tek yönde taşır. Bu kip veri aktarımında nadiren kullanılır. Çünkü alıcı meydana gelen hataları göndericiye haber veremez. Bunun örnekleri radyo-tv yayınlarıdır.

Yarı çift yönlü aktarımda her iki yöne aktarma yapılabilir. Ancak bir zaman biriminde veri sadece bir yöne yollanır. Bu kip için bir çift hat gerekir. Çok rastlanan örnek bir modemin gönderici diğer modemin alıcı olmasıdır. Bu yöntemde bir yöne aktarım tamamlanınca modemler yön değiştirerek tersine aktarım sağlanır. Yarı çift yönlü hattı, bir yönden diğerine çevirmek için gerekli zamana dönüş süresi denir.

Tam Çift Yönlü

Tam çift yönlü aktarımda veriler aynı anda her iki yönde gönderilebilir. Bu durum 4 hatlı bağlantı ile sağlanabilir Bir çift hat bir yöne ayrılabilir (diğer çift diğer yöne ait). İki hatlı bir devre, birbirine etki etmeyecek iki frekans bandı bulunabilirse (üst üste gelmeyecek) tam çift yönlü olarak kullanılabilir. Aynı fiziksel hat çifti alıcı ve gönderici görevi yaparlar.

Tutanak

Haberleşme hatları, modemler ve diğer aletler, iki veya daha fazla sayıda istasyon arasında bilginin gönderilmesi için kullanılır. Buna veri bağlantı adı verilir. Bilginin bu istasyonlar arasında düzenli ve kurallı bir biçimde gönderilmesi tutanak sayesinde mümkün olur. Veri bağlantı denetim tutanağı bir takım kurallardan oluşur. Bilginin düzgün ve kurallı biçimde aktarılabilmesi için bağlanan bilgisayar ve uçların bu kurallara uyması gerekir. Veri bağlantı tutanağının amacı, iki istasyon arasında bağlantı kurmak ve sona erdirmek, gönderici ve alıcıyı tanımak, bütün mesajların hatasız ve doğru biçimde geçmesini sağlamak ve veri aktarımı sırasında kullanılan bütün denetim fonksiyonlarını kullanmaktır. Tutanaklar mesaj çerçeveleme tekniklerine göre iki büyük kategoriye ayrılırlar. Karakter esaslı tutanak ve bit esaslı tutanak.

Karakter Esaslı Tutanak

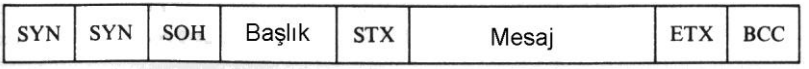
Karakter esaslı tutanak, karakter kümesinin ikilik kodlarına dayanır. En çok kullanılan kod ASCII dir. 7 bit kod olup 8 ci bit eşlik dir. Kodda 128 karakter mevcuttur. 95 tanesi grafik karakterleri, 32 tanesi denetim karakteridir. Denetim karakterleri veriyi yönlendirmek, istenen biçimde test yapmak ve yazıcıda sayfa düzenlemek için kullanılır. Aktarımı denetim için kullanılan karakterlere haberleşme denetim karakterleri denir. Bunlar çizelge 11.4 te verilmiştir.

SYN Karakteri

SYN karakteri alıcı ve gönderici arasında eş zamanlama görevi yapar. 7 bit ASCII kodu ve tek eşlik bit kullanıldığında 8 bit olarak SYN karakteri 00010110 koduna sahiptir. Gönderici 8 bit karakterler yollamaya başladığında önce birkaç karakter yollar, sonra gerçek mesajı yollar. Bunun için başlangıçtaki sürekli bit katarı SYN karakteri için denetlenir.

Çizelge 11.4 ASCII aktarım denetim karakterleri

Kodu	Sembol	Anlamı	Fonksiyon
0010110	SYN	hat boşken eş zamanlılığı sağlamak	Eş zamanlılığı sağlamak
0000001	SOH	Başlık başlangıcı	Mesaj bloğunun başlangıcı
0000010	STX	Metin in başlangıcı	Metin bloğundan önce bulunur
0000011	ETX	Metin in bitişi	Metnin sonunda bulunur
0000100	EOT	Aktarımın bitişi	Aktarımın bittiğini belirtir
0000110	ACK	Algılandı	Olumlu algılama
0010101	NAK	Negatif algılandı	Olumsuz algılama
0000101	ENQ	sorma	Uç birimin açık olup olmadığının sorulması
0010111	ETB	Aktarılan veri bloğunun bitişi	Veri bloğunun sonu
0010000	DLE	Veri bağlantı kaçışı	Özel denetim karakteri



Şekil 11.25 Karakter esaslı tutanak için tipik mesaj biçimi

Yani alıcı her saat vuruşunda son 8 bitin SYN karakteri olup olmadığına bakar. Eğer bu bitler SYN karakteri değilse en önemli biti bırakır ve yeni bir bit alır, yine SYN karakteri olup olmadığına bakar. Ne zaman SYN karakterini yakalarsa bundan sonraki her 8 bit 1 karakterdir. Genelde tek bir SYN karakteri yerine ardışık iki SYN karakteri istenir ve yollanır. Böylece her hangi bir gürültü sinyalinin tesadüfen karakter oluşturması önlenmiş olur. Ayrıca gönderici boş olduğu zaman sürekli olarak ardışık SYN karakterleri yollar. Alıcı bunları sürekli alır ve hat sürekli eş zamanlı durumda ama herhangi bir mesaj yollanmadan durur. Doğal olarak alıcı eş zamanlı durumunu korur. İki birim bit ve karakter bazında eş zamanlı durumda kalır.

Yollanan mesajların bir de mesaj biçimi vardır. Bu biçimde bir başlık alanı bir metin alanı ve bir hata denetim alanı bulunur. Karakter esaslı bir tutanağın tipik bir mesaj örneği şekil 11.25 te gösterilmektedir. Mesajın başındaki iki SYN karakteri uygun eş zamanlılığı sağlar. SYN karakterlerini başlık olarak yollanan SOH karakteri takip eder. Başlık adres ve denetim bilgileri taşır. STX karakteri başlığın sonu ve metnin başlangıcını belirler. Metin kısmı değişen uzunlukta olabilir ve denetim karakterleri haricindeki ASCII karakterlerinden oluşur. Metin kısmı ETX karakteriyle sona erer. Son alan blok denetim karakteri (BCC) dir. Hata denetiminde kullanılır. Bu LRC veya CRC olabilir.

Alıcı mesajı kabul eder ve kendi BCC sini hesaplar. Hesaplanan BCC alınan BCC ile uyuşmazsa alıcı bir NAK karakteri ile cevap verir. Mesaj tekrar gönderilir ve denetlenir. Mesajın hatalı olduğuna karar vermek için tekrarlamanın birkaç defa yapılması gerekir. Eğer alınan BCC hesaplanan ile çakışırsa göndericiye ACK karakteri yollanır.

Aktarma Örneği

VHİ nın çalışmasını ve işlevini anlamak için özel bir örnekle uç birim ve işlemcinin haberleşmesine bakılırsa, Bellek ile MİB nin haberleşmesi herhangi bir G/Ç işlemcisininkine benzer.

Uç birimden işlemciye gönderilen tipik bir mesaj çizelge 11.5 teki gibi olabilir Mesajın oluşması için birçok denetim karakteri kullanılmaktadır. Her karakter ve denetim karakterleri seri olarak 8 bit ikilik kodla aktarılır. 8 bit kodun 7 biti ASCII ve 1 biti (en önemli 8 ci bit) tek eşlik bitidir. İki SYN karakteri alıcı ve göndericiyi eş zamanlı yapar. Başlık

Çizelge 11.5 Bir uç birimden işlemciye tipik bir aktarım

Kod	Sembol	Yorumu
0001 0110	SYN	Birinci eş zamanlama karakteri
0001 0110	SYN	İkinci eş zamanlama karakteri
0000 0001	SOH	Başlığın başlangıcı
0101 0100	T	Uç birimin adresi T4 tür.
0011 0100	4	
0000 0010	STX	Metin aktarımına başlangıç
0101 0010		
0100 0101	1234 Nolu	Gönderilen metin 1234 nolu hesabın
.	hesabın	bakiyesini yolla şeklindedir.
.	bakiyesini	
.	yolla	
1011 0011		
0011 0100		
1000 0011	ETX	Metin aktarımının sonu
1101 0101	LRC	Boylamasına artıklık denetim karakteri

SOH karakteri ile başlar ve uç birimin adresini belirleyen iki karakterle devam eder. Bu özel örnekte adres T4 tür. Fakat genelde iki veya daha fazla grafik karakterden oluşan bir küme olabilir. STX karakteri başlığı sona erdirir ve metin aktarımının başladığını belirtir. Mesajı oluşturan karakterler çok yer kaplayacağından burada gösterilmemiştir. Her bir karakter biti 8 bitlik ve her bit seri olarak aktarılmaktadır. Metnin kendisi "1234 numaralı hesabın bakiyesini yolla" şeklindedir. ETX denetim karakteri metinin bitişini belirtir. ETX i takip eden karakter LRC karakteridir. Bu karakterin içindeki her bit bir eşlik biti olup, çizelgenin kodlama kısmında bütün bitlerinden hesaplanmıştır.

VHİ bu mesajı alır ve onu analiz eder. Mesajdan T4 uç birimini görür ve mesajı yükler. Karakterleri alırken işlemci her bir karakterdeki eşliğe bakar ve LRC yi hesaplar. Hesaplanan LRC, alınan LRC ile karşılaştırılır. İkisi çıkışırsa pozitif algılandı (ACK) yollanır. Eğer çakışmazlık varsa negatif algılandı (NAK) gönderilir. Eğer NAK gelirse uç birim aynı mesajı tekrar işlemciye gönderir. Eğer işlemci mesajın hatasız olduğunu görürse mesajı belleğe yazar ve MİB ne bir kesme gönderir. MİB kesmeyi alınca onu analiz eder ve bir metin mesajı hazırlar. Bu mesaj yapılan istemin cevabıdır. MİB, VHİ ne bir buyruk yollar. Bu buyruk ile VHİ cevabi mesajı uç birime gönderir.

Uç birime işlemciden tipik bir cevap çizelge 11.6 da verilmiştir. İki SYN karakterinden sonra işlemci önceki mesajı aldığını ACK karakteri ile belirtir. Hat SYN karakteri yollayarak boş durmaya devam eder. MİB den alınan mesaj işlemci tarafından uygun biçime getirilir. Yani metnin

Çizelge 11.6 İşlemciden uç birime tipik aktarım

Kod	Sembol	Yorumu
0001 0110	SYN	Birinci eş zamanlama karakteri
0001 0110	SYN	İkinci eş zamanlama karakteri
1000 0110	ACK	Önceki mesajın pozitif kontrol karakteri
0001 0110	SYN	Hat boş
.	.	
.	.	
.	.	
0001 0110	SYN	Hat boş
0000 0001	SOH	Başlığın başlangıcı
0101 0100	T	Uç birimin adresi T4 tür
0011 0100	4	
0000 0010	STX	Metin aktarımını başlangıcı
0100 0010		
1100 0001	bakiye	Gönderilen metin istenen hesabın bakiyesi
.		
.	100.00 $	
.		
1011 0000		
1000 0011	ETX	Metin aktarımının bitişi
1101 0101	LRC	Boylamasına artıklık denetim karakteri

başına ve sonuna uygun denetim karakterleri yazılır. Mesajda SOH başlığı ve uç birimin T4 adresi vardır. Mesaj uç birime bakiyenin 100.00 $ olduğunu bildirir. Bir LRC karakteri hesaplanıp uç birime yollanır. Eğer uç birim NAK karakteri yollarsa işlemci mesajı tekrarlar.

İşlemci bu uç birim ile uğraşırken, diğer uç birimleri de çalıştırmak zorundadır. Karakterler seri olarak geldiklerinden 8 bitin alınabilmesi için belli bir sürenin ayrılması gerekir. Bu süre içinde işlemci diğer haberleşme hatlarına bakar ve her birine sırayla hizmet verir. En uzak uç birimlerin hızı işlemciye göre en yavaştır. Bu özellik bir çok kullanıcıya zaman paylaşımlı sistemde daha büyük etkinlik sağlama imkanı verir. Bu sayede birçok kullanıcı insanların (klavyede yazma) hızıyla aynı anda çalışıyor olurlar.

Veri Saydamlığı

Karakter esaslı sistem klavye, yazıcı ve ekranlarla haberleşmek için kurulmuştur. Çünkü bu aletler alfasayısal karakterleri yoğun biçimde kullanırlar. Fakat bilgi haberleşme alanı genişleyince, ASCI metin biçiminde olmayan ikilik bilgilerin aktarılması gerekli olur. Örneğin iki uzak bilgisayarın birbirlerine program veya veri göndermeleri gibi. Bir metin mesa-

jı içinde rasgele bir bit kombinezonu, karakter esaslı tutanak içinde sorun olur. Çünkü bir denetim karakteri yanlış biçimde yorumlanabilir. Örneğin mesajın içindeki yani metin kısmındaki 1000 0011 bit kombinezonu, alıcı tarafından ETX karakteri olarak algılanır. Yani metnin sonu sanılır. Eğer mesajın metin kısmı herhangi bir koda ait değilse buna saydam veri adı verilir. Bu durumda alıcının tanıma mantığı devresi kapatılır ve bit katarının denetim bilgisi olması olasılığı ortadan kaldırılır.

DLE Karakteri

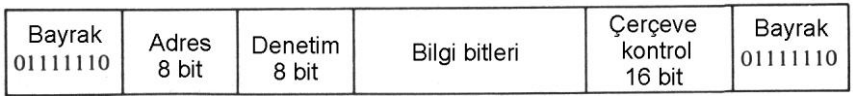
Veri saydamlığına karakter esaslı tutanakta erişmek istersek DLE karakterine (veri bağlantı kaçışı) gerek vardır. Her bir haberleşme denetim karakterinden önce DLE karakteri yollanır. Böylece başlık iki karakterle DLE SOH, metin alanı ise DLE ETX karakterleriyle belli olur. Eğer DLE karakteri olan 0001 0000 bit katarı mesajın metin kısmı içinde geçerse gönderici ardından başka bir DLE bit katarı yollar. Alıcı DLE karakterlerinin hepsini ortadan kaldırır ve sonraki 8 biti denetler. Eğer bu 8 bit DLE karakteri ise alıcı bunun metnin bir parçası olduğunu düşünür ve metni almaya devam eder. Aksi taktirde alıcı DLE yi takip neden 8 biti denetim karakteri olarak kabul eder.

Veri saydamlığında DLE karakterinin kullanılması etkin olmayıp bir şekilde kurulması da karışıktır. Bu yüzden başka tutanaklar geliştirilmiş ve saydam verinin aktarımı etkin hale getirilmiştir. DEC (digital Equipment Corporation) tarafından kullanılan bir tutanak, byte sayım alanı kullanır ve ardından gelen mesajdaki byte sayısını verir. Alıcı her metin alanındaki byte ları sayar ve gelenle karşılaştırılır. Bu tutanak, saydamlık problemini çözmekte kullanılmaktadır. Karakter esaslı tutanaktaki başka problemleri ve veri saydamlığını çözmekte çok kullanılan tutanak bit esaslı tutanaktır.

Bit Esaslı Tutanak

Bit esaslı tutanak denetim alanlarında karakter kullanmaz. Her türlü tutanaktan bağımsızdır. Seri halindeki bitlerin aktarımını sağlar. Karakter sınırlarını belirlemez. Özel bir biçimde oluşturulan çerçeve içinde mesajlar yollanır. Bilgi alanı haricinde çerçevede adres denetim ve hata denetim olanları vardır. Çerçeve sınırları bayrak adı verilen özel 8 bitlik bir sayıdan elde edilir.

Herhangi bir veri haberleşme bağlantısında en az iki bağlantı birimi vardır. Veri bağlantısındaki işlerden, bu birimlerden bir tanesi sorumludur. Komutları o verir. Buna baş istasyon adı verilir. Diğer birim ikincil istasyondur. Bit esaslı tutanak bir tane baş istasyonun ve bir veya daha fazla ikincil istasyonun varolduğunu kabul eder. Veri aktarım bağlantısında bütün haberleşme boş istasyondan ikincil istasyonlara veya ikincil istasyonlardan boş istasyona doğrudur.



Şekil 11.26 bit esaslı tutanak için çerçeve biçimi

8 bit bayrak

Bit esaslı tutanağın çerçeve biçimi şekil 11.26 da gösterilmiştir. Bir çerçeve 8 bit 0111 1110 bayrağı ile başlar adres ve denetim kısmıyla devam eder. Bilgi alanı biçim ve içerik yönünden kısıtlı olmayıp herhangi bir uzunlukta olabilir. Çerçeve denetim alanı CRC tipindedir. Mesajın başındaki iki SYN karakteri uygun eş zamanlılığı sağlar. SYN karakterlerini başlık olarak yollanan SOH karakteri takip eder. Başlık adres ve denetim bilgileri taşır. STX karakteri başlığın sonu ve metnin başlangıcını belirler. Metin kısmı değişen uzunlukta olabilir ve denetim karakterleri haricindeki ASCII karakterlerinden oluşur. Metin kısmı ETX karakteriyle sona erer. Son alan blok denetim karakteri (BCC) dir. Hata denetiminde kullanılır. Bu LRC veya CRC olabilir.

Sıfır Yollama

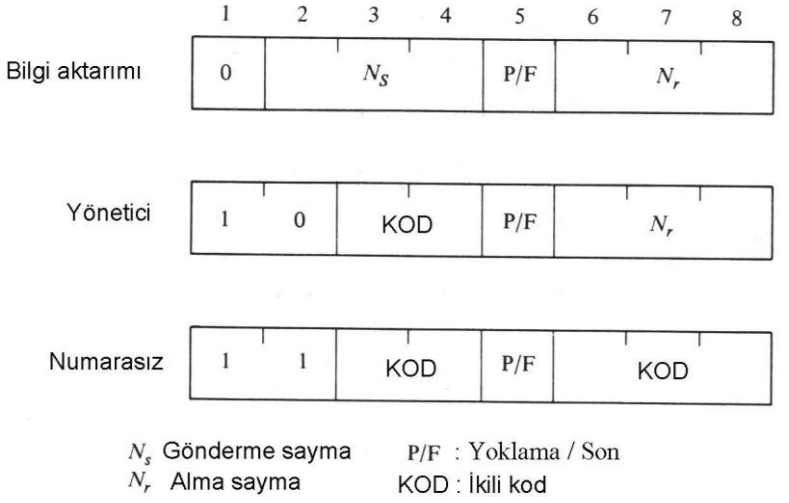
Bir çerçevenin orta yerinde bayrak yollamaktan kaçınmak için bit esaslı tutanak sıfır yollama yöntemini kullanır. Bu yöntemde 5 tane sürekli 1 in adından gönderici istasyon tarafından bir 0 ı dikkate almaz. Dolayısıyla 0111111 şekli 01111101 şeklinde yollanır ve alıcı tarafından esas şeklinde algılanır. Dolayısıyla asla iki bayrak arasında 0111 1110 biçiminde bir şekli yollamak mümkün değildir.

Bayrağı takip eden alan adres alanıdır. Baş istasyon tarafından, ikincil istasyonun adresini belirlemek için kullanılır. İkincil istasyon tarafından yollanan mesajda baş istasyonun adresi vardır. Adres alanı 8 bittir. En fazla 256 adres belirtebilir. Bazı bit esaslı tutanaklar uzatılmış adres alanı kullanılmasına izin verirler. Bunu yapmak için adres byte ının en önemsiz biti 0 yapılır ve bunu bir adres byte ının takip ettiği haber verilir. En önemsiz bitte 1 bulunursa bunun son adres byte ı olduğu anlaşılır.

Denetim Alanı

Adres alanından sonra denetim alanı gelir. Denetim alanı 3 farklı biçimde gelebilir. (Şekil 11.27) Bilgi aktarması veri aktarması gibidir. Bu kiple her bir çerçevede yollanan ve alınan sayılar vardır. Bir istasyon arka arkaya çerçeveler yolluyorsa çerçevelerin sayısını sayar ve onlara numara verir. Bu sayı gönderme sayısı N_S dir. Alıcı istasyon sırayla çerçeveler alıyorsa hatasız çerçeveleri sayar. Bu sayı N_r dir. N_r sayısı çerçeve alınıp hatasız olduğu anlaşılınca 1 arttırılır. Alıcı, alıcı sayısı N_r yi göndericiye göndererek olayın doğruluğunu tahkik eder.

P/F biti baş istasyon tarafından kullanılırsa, ikincil istasyona aktarmayı başlatması için bir sinyaldir. Eğer ikincil istasyon tarafından kullanılıyorsa aktarılan son çerçeve olduğunu belirtir. Dolayısıyla baş istasyon için yoklama (P) ve ikincil istasyon için son (F) bitidir. Her çerçeve ikin-



Şekil 11.27 bit esaslı tutanağın denetim alan biçimi

cil istasyona, baş istasyondan P biti 0 olarak gönderilir. Baş istasyon bitirip ikincil istasyonun cevabı için hazır olduğunda P biti 1 yapılır. İkincil istasyon bir miktar çerçeve yollayarak cevap verir. Bunlarda F biti 0 dır. İkincil istasyon son çerçeveyi gönderdiğinde F yi 1 yapar. Böylece P/F biti bir istasyondan yapılan aktarmanın bittiğini anlamak için kullanılır.

Denetim alanının yönetici biçimi ilk iki bitin 1 ve 0 olmasından tanınır. Bundan sonraki iki bit komutun tipini belirtir. Bu bir P/F biti ile takip edilir. Bunun ardından çerçeve sayısı gelir. Yönetici biçimde bilgi alanı bulunmaz. Bunlar bilginin aktarımına yardımcı olmak, gelen çerçevelerin kabul edildiklerini bildirmek, hazır veya meşgul durumlarını bildirmek ve çerçeve sayma hataları için kullanılır.

Numarasız biçim ilk iki bitin 11 olmasından tanınır. Bu biçimdeki 5 kod biti 32 komut veya cevabı taşıyabilir. Baş istasyon denetim alanını ikincil istasyona bir komut belirtmekte kullanır. İkincil istasyon denetim alanını boş istasyona cevap göndermek için kullanılır. Numarasız biçim çerçeveleri bağlantı fonksiyonlarının ilkleştirmesinde işlem hatalarının raporlanmasında, istasyonların bağlantısız kipe geçirilmesinde ve diğer veri bağlantı denetim işlemlerinde kullanılır.

SORULAR

11-1. Şekil 11.2 deki G/Ç ara yüzeyinin dört yazacına verilen adresler ondalık 12, 13, 14 ve 15 in ikili karşılıklarına eşittir. İşlemcinin 8 bit G/Ç adresi ile ara yüzeyin *CS*, *RS*1, *RS*0 girişleri arasındaki devreyi çizin .

11-2. Şekil 11.2 de gösterilen tipte, altı ara yüzey, 8 bit G/Ç adresi kullanılan MİB ne bağlanmıştır. 6 tane yonga seçim girişinin her biri farklı bir adres hattına bağlıdır. Böylece yüksek seviyeli adres hattı ara yüzeyin *CS* girişine bağlanmış olur ve altıncı adres hattı altıncı ara yüzeyin *CS* girişine bağlanmış olur. İki tane düşük seviyeli adres hattı altı tane ara yüzün tüm *RS*1 ve *RS*0 uçlarına bağlanmıştır. Her bir ara yüzeydeki her bir yazacın 8 bit adresini bulun .

11-3. Bir kişinin anlayabileceği biçimde çıktı üreten dört çevre birimi sayınız.

11-4. ASCII kodu kullanarak tam adınızı yazın. En soldaki bit sıfır olmak üzere her karakter için 8 bit kullanınız. Adlar arasına boşluk ve ilk veya ortadaki tek harften sonra nokta konmalıdır.

11-5. Yalıtılmış G/Ç ve bellek haritalamalı G/Ç arasındaki fark nedir? Her birinin fayda ve zorlukları nelerdir.

11-6. Aşağıdakilerin her birinin denetim mi, durum mu yoksa veri aktarım emri mi olduğunu söyleyin.

a. Eğer bayrak kalkmışsa sonraki buyruğa atla
b. Verilen bir kaydı diskte ara
c. G/Ç aletinin Hazır olup olmadığına bak
d. Yazıcı kağıdını sonraki sayfanın başına yolla
e. Ara yüzey durum yazacını oku

11-7. Ticari bir ara yüzey, G/Ç aletinden ara yüzey birimine veri aktaran hatlarla birlikte farklı adlar kullanır. Ara yüzey onay giriş hattı *STR* başla, onay çıkış hattı *IBF* (giriş bufferı dolu) olarak adlandırılır. *STB* üzerinde düşük bir sinyal G/Ç veri yolundan ara yüzey veri yazacına veri yükler. *IBF* üzerinde yüksek bir sinyal, verinin ara yüzey tarafından kabul edildiğini belirtir. *IBF*, MİB den gelen oku sinyalinin ardından veri yazacının içeriğini okuduktan sonra düşük seviyeye geçer.

a. Bir blok şema çizerek MİB, ara yüzey ve G/Ç aletini, bu üçü arasında geçerli bağlantılarla beraber gösterin.
b. Onay aktarımını zaman şemasında gösterin.
c. Bir akış şeması çizerek aletten ara yüzeye ve ara yüzeyden MİB ne aktarımlar sırasında olan olayları üzerinde gösterin.

11-8. Erişim zamanı 40 ns olan bellek 200 MHz saati olan bir MİB ne bağlıdır. Okuma başlaması ve yazma başlaması için ayrı zaman şemaları çiziniz. Adresi zaman şeması içinde gösterin.

11-9. Şekil 11.8 de gösterilen eş zamanlı olmayan iletişim ara yüzeyi MİB ile yazıcı arasına bağlıdır. MİB yazıcıya basılmak üzere karakterler yolladığında ara yüzeyin gönderici parçasında meydana gelen işlemleri sırayla bir akış şeması çizerek gösteriniz.

11-10. Eş zamanlı olmayan iletişim ara yüzeyinin durum yazacındaki bitlerin en az altı durum şartında alacakları değerleri tek tek belirtiniz.

11-11. Şekil 11.8 deki aktarıcı kaydırma yazacı, ara yüzey tek bir dur bitine gerek duyan bir uca bağlandığında, içinde kaç tane bit vardır? *W* harfi

ASCII ve tek eşlik kullanarak gönderildiğinde kaydırma yazacındaki bitleri ve durumları sıralayın.

11-12. Aşağıdaki kiplerin her birinde 1200 baud bir hat üzerinden saniyede kaç karakter gönderilir. (Karakterin 8 bit olduğu varsayılacak)
- **a.** Eş zamanlı seri aktarım
- **b.** Eş zamanlı olmayan seri aktarı, iki dur biti
- **c.** Eş zamanlı seri aktarım, bir dur biti

11-13. Bir FIFO (İGİÇ) buffere saniyede m byte hızıyla bilgi yükleniyor.Bilgi saniyede n byte oranında gecikiyor. Bufferin en toplam hacmi K byte tır.
- **a**. $m > n$ olduğunda boş bir bufferın dolması ne kadar sürer?
- **b**. $m < n$ olduğunda dolu bir bufferın boşalması ne kadar sürer?
- **c.** $m = n$ olduğunda FIFO (İGİÇ) buffer gerekir mi?

11-14. FIFO nun denetim yazacındaki (şekil 11.19) bitler, $F_1F_2F_3F_4 = 0011$ dır. FIFO daki veri silindiğinde ve yeni bir bilgi yazıldığında iç işlemlerin sırasını söyleyin.

11-15. Şekil 11.19 daki giriş hazır, çıkış hazır ve F_1 den F_4 e kadar denetim bitlerinin
- **a.** Buffer boş iken
- **b.** Buffer dolu iken
- **c.** Buffer iki bilgi içerirken değerleri nedir?

11-16. Şekil 11.10 daki blok şemaya benzer bir şekil çizerek, MİB den bir ara yüzeye oradan da G/Ç aletine veri aktarılışını gösteriniz. Bayrak bitinin doldurulması ve silinmesi işlemlerini belirleyin.

11-17. Soru 11.16 da kurulan durumdan yararlanarak (şekil 11.11 e benzerdir) MİB den veri çıkışı yapan programın akış şemasını elde edin.

11-18. Kesme işleminin başlattığı aktarımın, kesme istemsiz, program denetiminde yapılan aktarıma göre temel üstünlüğü nedir?

11-19. Bir çok bilgisayarda kesme istemi ancak buyruğun icrasının bitmesinden sonra fark edilir. Kesme işleminin, buyruğun icrası sırasında herhangi bir anda onaylandığını varsayıp oluşacak zorlukları tartışın.

11-20. Şekil 11.12 de gösterilen papatya zinciri kesme önceliğinde, eğer alet1, alet2 nin MİB ne kesme istemi yollamasından sonra fakat MİB nın onayından önce kesme isteminde bulunursa ne olur?

11-21. Kesme istemi önceliği olmayan bir bilgisayar düşünelim. Herhangi bir kaynak bilgisayarın çalışmasını kesebilir. Bu durumda kesme istemleri dönüş adreslerini ve kesme yordamına dallanma istemlerini bir yere yazarak saklarlar. Kesme istemi hizmet yordamında nasıl bir öncelik düzeni kurulabilir.

11-22. Birleşik devre tasarım yöntemlerini kullanarak çizelge 11.2 de gösterildiği biçimde öncelik kodlayıcı için Boole bağıntılarını bulun. Devreyi çizin.

11-23. Sekiz kesme kaynağı olan bir sistem için paralel öncelikli kesme donanımı tasarlayın.

11-24. 8 × 3 öncelik kodlayıcı için doğruluk çizelgesi yapın. Varsayın ki öncelik kodlayıcısının 3 çıktısı *xyz*, 101*xyz*00 biçiminde bir vektör oluştursun. En yüksek öncelikli olandan başlayarak sekiz vektör adresini yazın.

11-25. Şekil 11.14 teki dört *VAD* değerinin 76, 77, 78 ve 79 un ikili değerlerine ulaşması için ne yapılabilir.

11-26. Bir kesme işlemi sonucunda, kesme hizmet yordamı çalışırken aynı kaynaktan başka bir kesme istemi gelirse bunu denetlemek için hangi programlama adımları gerekir.

11-27. Bir DMA denetleyicisindeki oku ve yaz denetim hatları neden iki yönlüdür? Hangi koşul altında ve hangi amaçla bunlar giriş olarak kullanılır? Hangi koşulda ve hangi amaçla bunlar çıktı olur?

11-28. Diskten 256 kelimenin, bellekte 1230 dan başlayan adrese aktarılması gerekiyor. Aktarım şekil 11.18 de görüldüğü gibi DMA yoluyla yapılacaktır.

a. MİB ın DMA denetimcisine aktarmak zorunda olduğu ilk değerleri verin.

b. İlk iki kelimenin girişi sırasında yapılan hareketlerini adım adım söyleyin.

11-29. Bir DMA denetimcisi 16 bit kelimeyi çevrim çalma yöntemiyle belleğe aktarıyor. Kelimeleri saniyede 2400 karakter hızıyla aktaran bir aletten alınarak birleştirilmiştir. MİB buyrukları alıp getirmekte ve icra etmektedir. Al-getir ve icra hızı ortalama saniyede 1 milyon buyruktur. DMA aktarımından dolayı MİB ne kadar yavaşlar?

11-30. Her ikisi de bellek aktarımı isteğinde bulunduklarında neden DMA nın, MİB ne göre önceliği vardır?

11-31. Şekil 11.20 dekine benzer bir akış şeması çizerek IBM370 te MİB–G/Ç haberleşmesini anlatın.

11-32. Veri iletişim işlemcesine bağlı olan bir ucun adresi alfabenin iki harfinden veya bir harfle bunu takiben 10 rakamdan oluşuyor. Kaç tane farklı adres üretilebilir.

11-33. Bir veri işletim işlemcisi ile uzak bir uç arasında iletişim için gerekli karakter sırası ile hat işlemlerini sıralayın. İşlemci, uç çalışıyorsa istem yapar. Uç, evet veya hayır yanıtını verir. Eğer evet derse işlemci veri bloğunu yollar.

11-34. Bir veri işletim bağlantısı, DLF karakteri kullanarak veri saydamlaştırma işlemini, karakter denetimli tutanak ile beraber kullanmaktadır. STX ile ETX arasında göndericinin yolladığı haber şöyledir.

```
DLE  STX  DLE  DLE  ETX  DLE  ETX  DLE  ETX
```

Saydam verinin ikili değeri nedir?

11-35. Bit yerleşimli bir tutanakta bir çerçevenin sahip olması gereken en az bit sayısı kaçtır?

11-36. Bit esaslı tutanakta sıfır doldurma işlemi, eğer sıfırdan sonra 1023 sayısının ikili eşdeğeri olan 10 bit gerekiyorsa nasıl çalışır?

KAYNAKLAR

1. Gorsline, G. W., *Computer Organization: Hardware/Software,* 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1986.
2. Hays, J. F., *Computer Architecture and Organization*. 2nd ed. New York: McGraw-Hill, 1988.
3. Hill, F. J., and G. R. Peterson, *Digital Systems:Hardwire Organization and Desing*, 3rd ed. New York: John Wiley , 1987.
4. Hwang, K., and F. A. Briggs, *Computer Architecture and Parallel Processing*. New York: McGraw-Hill, 1984.
5. Lippiatt, A. G., and G. L. Wright, *The architecture of Small Computer Systems,* 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1985.
6. Patterson, D. A., and J. L. Hennessy, *Computer Architecture :A Quantitative Approach*. San Mateo, CA: Morgan Kaufmann Publishers, 1990.
7. Pollard, L. H., *Computer Desing and Architecture,* Englewood Cliffs, NJ: Prentice Hall, 1990.
8. Rafiquzzaman, M., and R. Chandra, *Modern Computer Architecture*. St. Paul, MN: West Publishing, 1988.
9. Toy, W., and B. Zee, *Computer Hardware/Software Architecture*. Englewood Cliffs, NJ: Prentice Hall, 1986.
10. Wakerly, J. F., *Microcomputer Architecture and Programming*. New York: John Wiley , 1981.
11. Ward. S. A., and R. H. Halstead, Jr., *Computer Structures*. Cambridge, MA: MIT Press, 1990.

BÖLÜM 12

Bellek Tasarımı

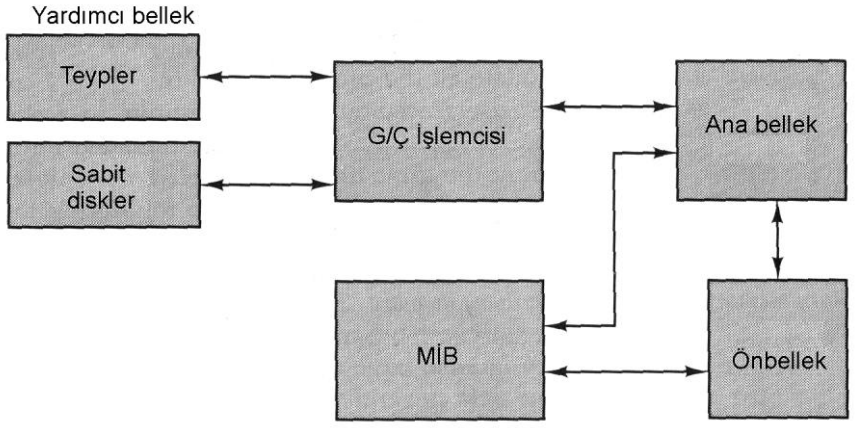
BU BÖLÜMDE

12.1 Bellek Hiyerarşisi
12.2 Ana Bellek
12.3 Yardımcı Bellek
12.4 İçerik Adreslemeli Bellek
12.5 Ön Bellek
12.6 Sanal Bellek
12.7 Bellek Yönetim Donanımı

12.1 Bellek Hiyerarşisi

Yardımcı Bellek

Bellek herhangi bir sayısal bilgisayarın temel bileşenlerinden biridir. Veri ve programlar burada tutulur. Birçok genel amaçlı bilgisayarda ana belleğin kapasitenin üstünde ek saklama kapasitesi bulunmasının, bilgisayarın etkin çalışmasına çok yararlı olduğu bilinmektedir. Çünkü belleğin kapasitesinin tüm programları tutması mümkün değildir. Bunun dışında pek çok kullanıcı verileri işlemek için çok sayıda program ve veri bulundurmaktadır. Bu verilerin hepsi aynı anda işlemciye gerekli değildir. Bu yüzden o anda MİB tarafından kullanılmayan verilerin daha ucuz bir yardımcı bellekte tutulması uygun olur. Bu yardımcı bellek yedekleme aleti olarak ta kullanılır. MİB ile doğrudan haberleşebilen belleğe ana bellek denir.Yedekleme görevi yapan aletlere ise yardımcı bellek denir. En çok kullanılan yardımcı bellekler sabit disk ve teyplerdir. Bunlar sistem programlarının büyük veri dosyalarının ve diğer yedekleme bilgilerinin saklanmasında kullanılırlar. Sadece kullanılan programlar ve o anda işlemciye gerekli veriler bellekte bulunur ve gerektikçe ana belleğe getirilir.



Şekil 12.1 Bir bilgisayar sisteminde bellek hiyerarşisi

Bir bilgisayarın bellek kapasitesi bileşenlerinin sıralanması ile elde edilir. Yani yavaş fakat geniş yardımcı belleklerden daha hızlı ana bellek ve küçük fakat hızlı ön bellek bu hiyerarşiyi oluşturur. (Şekil 12.1). Hiyerarşinin en altında yavaş manyetik teypler vardır. Bunlara değiştirilebilir dosyalar konur. Bundan sonra yedekleme yapmak için sabit diskler gelir. MİB ile haberleşmesinden dolayı ana bellek merkezi yerde bulunur. Bellek ayrıca G/Ç işlemcisi yardımıyla yardımcı belleklerle de haberleşir. Ana bellekte bulunmayan programlar, MİB tarafından istenince yardımcı bellekten ana belleğe getirilir. MİB nin ihtiyaç duymadığı programlar yardımcı belleğe aktarılarak diğer programlar ve veriler için yer açılır.

Ön Bellek

Özel ve çok hızlı bir bellek parçasıdır. Bazen işlemin hızını arttırmak için bazen de MİB ne daha hızlı program ve veri yetiştirmek için kullanılır. Bilgisayar sistemlerinde ön bellek, ana belleğe erişim zamanı ile MİB arasındaki hız farkını yok etmek için kullanılır. MİB genellikle bellek erişim zamanından daha hızlı çalışır. Bu durumda bilgisayarın hızı bellek erişim hızıyla sınırlanır. Bunu ortadan kaldırmak için çok çok hızlı fakat küçük bir bellek parçası bellekle MİB arasına konur. Ön belleğin hızı MİB nin hızına yakındır. Ön bellek içinde halen o anda MİB içinde icra edilmekte olan programın parçaları ile bu programın sık sık ihtiyaç duyduğu geçici veriler tutulur. Programların ve verilerin hızlı bir şekilde MİB ne iletilebilmesi, bilgisayarın verimini arttırır.

G/Ç işlemcisi yardımcı bellek ile ana bellek arasında veri aktarımı yaparken, ön bellek tasarımı da bellekle MİB arasındaki bilgi aktarımıyla ilgilenir. Böylece her biri bellek hiyerarşisi içinde farklı seviyelerle uğraşmış olurlar. 2 veya 3 seviyeli bellek hiyerarşisine sahip olmamızın nedeni ekonomiktir. Belleğin saklama kapasitesi arttıkça, bit başına düşen

bilgi saklama fiyatı azalır ve belleğe erişim süresi artar. Yardımcı belleğin kapasitesi büyüktür. Fiyatı oldukça ucuzdur. Fakat ana belleğe kıyasla erişim süresi uzundur. Ön bellek çok küçük ve çok pahalıdır, çok çok hızlıdır, hız arttıkça fiyat artar. Bellek hiyerarşisi kullanmanın genel amacı, en yüksek ortalama hıza ve en düşük fiyata erişmektir.

Yardımcı bellekler ve ön bellekler farklı amaçlar için kullanılır. Ön bellek program ve verilerin en yoğun kullanılan kısımlarını tutar. Yardımcı bellek, o anda MİB tarafından kullanılmayan kısımları tutar. Bundan başka MİB nin ön belleğe ve ana belleğe direk erişimi vardır. Fakat MİB nin yardımcı belleğe direk erişimi yoktur. Yardımcı bellekten ana belleğe aktarım DMA yoluyla ve büyük veri bloklarının aktarımı biçiminde olur. Ön bellek ile ana bellek arasındaki erişim oranı 1/7 dir. Örneğin tipik bir ön belleğin 100 ns erişim süresi vardır. Ana belleğin erişim süresi 700 ns civarındadır. Yardımcı belleğin erişim süresi ana belleğin 1000 katıdır. Yardımcı belleğin genel blok uzunluğu 256 ile 2048 kelime arasındadır.

Çoklu Programlama

Birçok işletim sistemi MİB nin bağımsız programları bitişik zamanlarda icra etmesini sağlar. Bu biçimde programlar aynı anda icra ediliyormuş görüntüsü verir. Bu kavrama çoklu programlama adı verilir. Burada 2 veya daha fazla program, belleğin farklı sıra düzensel bölgelerinde bulunurlar. Bu yolla bilgisayarın bütün parçalarının çalışması sağlanır. Programlar kısa aralıklarla arka arkaya icra edilirler. Örneğin bir program MİB de icra ediliyor olsun ve bir G/Ç aktarımına gerek duysun. MİB, G/Ç işlemcisini çalıştırır ve aktarımı başlatır. Bu durumda MİB boş kalır, başka bir programı icraya başlar. Çoklu programlamada bir program G/Ç aktarımı için beklerken, başka bir program MİB de icra edilmeye hazırdır ve icra edilir.

Çoklu programlama ile programları parçalar halinde icra etmek, belleğin değişken bir parçasının bu program tarafından kullanılması ve programların bellek hiyerarşisi içinde hareketini gerekir. Bilgisayar programları bazen tümüyle ana belleğe sığmaz. Ayrıca bilgisayar aynı anda birçok programı çalıştırır. Bütün bu programların ana bellekte devamlı durmaları imkansızdır, program ve veriler normalde yardımcı bellekte durur. Bir program veya programın bir kesimi icra edileceği zaman ana belleğe aktarılır ve MİB de icra edilir. Dolayısıyla yardımcı bellek, bilgisayar sisteminin tüm bilgilerini tutar. İşletim sistemi bu programlardan çalışan programın gerekli kısımlarının ana bellekte olmasını sağlar. Bilgilerin ana bellekle yardımcı bellek arasında gidip gelmesini sağlayan parçaya bellek yönetim sistemi denir. Bu sistemin 12.7 de verilecektir.

12.2 Ana Bellek

RAM

Ana bellek bilgisayar sisteminin merkezi depolama birimidir. Oldukça büyük ve hızlıdır. Bilgisayarın çalışması sırasında programlar ve veriler burada bulunur. Ana bellek için kullanılan teknoloji yarı iletken IC lerdir. IC ram yongaları iki olası kipte bulunur. Bu işlem kipleri durağan ve dinamiktir. Durağan RAM, ikili bilgileri depolayan yaz-bozlardan oluşur. Depolanan bilgi, birime akım verildiği sürece geçerli kalır. Dinamik RAM, bilgileri elektrik akımı şeklinde tutar. Tutan devre kapasitör dür. Kapasitörler MOS transistörleriyle yapılır. Üzerlerindeki yük zaman içinde azalır. Dolayısıyla belli sürelerle yeniden yüklenmelidir. Buna yenileme diyelim. Yenileme birkaç milisaniyede bir tüm kelimelere uygulanır. Dinamik RAM enerji tüketimini azaltır. Yüksek depolama kapasitesi sağlar. Kullanımı kolay okuma ve yazma süreleri kısadır.

ROM

Ana belleğin büyük kısmı RAM IC leriyle yapılmıştır. Belleğin bir kısmı ROM yongalarıyla yapılabilir. Orijinal olarak RAM rasgele erişilebilir bellek olarak kullanılır. ROM aynı zamanda rasgele adreslenebilir bellektir. RAM değiştirilebilir programlar ve verilerin tamamını depolar. ROM ise bilgisayarda değişmeden duracak programlarla, değişmeyecek ve bilgisayarın kuruluşu sırasında oluşturulan çizelge ları depolar.

Ön Yükleme Yükleyicisi

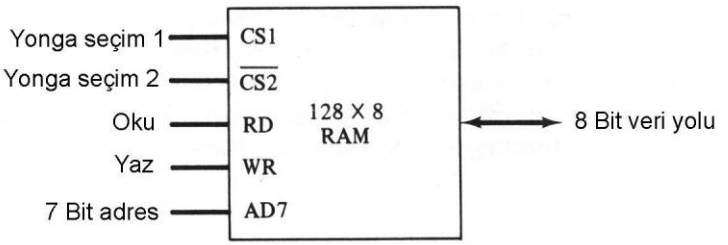
Belleğin ROM kısmında başlatıcı bir program bulunur. Buna ön yükleme yükleyicisi denir. Görevi bilgisayarın düğmesi açıldığında bilgisayar yazılımlarını çalıştırmaktır. Düğme kapatıldığı sırada RAM ın içeriği bozulduğu için buna gerek yoktur. ROM un içeriği ise açılıp kapamayla bozulmaz. Bilgisayarın çalışması için düğmeyi açmak ve ön yükleme programını çalıştırmak gerekir. Düğme açılınca donanım PC yi ön yükleme yükleyicisinin ilk adresine 1 yapar. Ön yükleme yükleyicisi işletim sisteminin bir kısmını sabit diskten belleğe yükler ve denetim işletim sistemine aktarılır. İşletim sistemi bilgisayarı genel kullanıma hazırlar.

RAM ve ROM yongaları çeşitli boylarda bulunur. Bilgisayara gerekli bellek bir yonganın kapasitesinden büyükse birkaç tane yonganın bir araya getirilmesi gerekir. Şimdiki durum budur ve istenen kapasite bu yolla elde edilir. Yonga bağlantılarını göstermek üzere 128 × 8 RAM ve 512 × 8 ROM yongaları 1024 × 8 bellek yapılışını örnekleyeceğiz.

RAM ve ROM Yongaları

Çift Yönlü Veri Yolu

RAM yongaları MİB ile haberleşmek için tasarlanmıştır. Çünkü bir veya daha fazla denetim girişi vardır ve yongayı gerektiğinde seçecek yonga seçim girişleri vardır. Diğer ortak bir özellik iki yönlü veri yoludur. Bu şekilde iki yönlü veri yolu, veriyi oku işlemiyle bellekten MİB ne veya yaz işlemiyle MİB den belleğe aktarır. İki yönlü veri yolu, 3 zamanlı



(a) Blok şema

CS1	$\overline{CS2}$	RD	WR	Bellek fonksiyonu	Veri yolu durumu
0	0	×	×	tutma	yüksek direnç
0	1	×	×	tutma	yüksek direnç
1	0	0	0	tutma	yüksek direnç
1	0	0	1	yaz	RAM'a giriş bilgisi
1	0	1	×	oku	RAM'den çıkış bilgisi
1	1	×	×	tutma	yüksek direnç

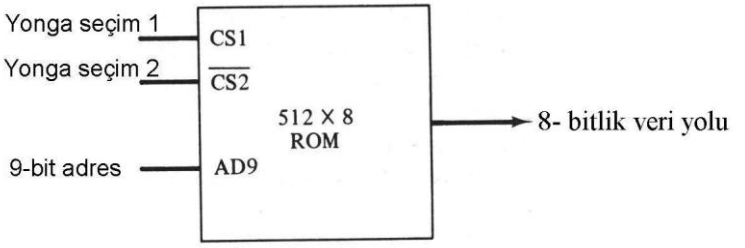
(b) Fonksiyon çizelgesi

Şekil 12.2 Tipik RAM yongası

buffer larla yapılabilir. 3 zamanlı buffer 3 farklı durumdan birine getirilebilir. Mantıksal 1 e eşit bir sinyal, mantıksal 0 a eşit bir sinyal ve yüksek direnç durumu. Mantıksal 1 ve 0 normal sayısal sinyallerdir. Yüksek direnç durumu kapalı bir devreyi belirtir. Bunun anlamı çıkış herhangi bir sinyal taşımaz ve mantıksal önemi yoktur (0 veya 1 değildir).

Şekil 12.2 de RAM yongasının blok şeması görülüyor. Kapasitesi 128 kelimedir. Burada 1 kelime 8 bittir. Bu 7 adres ve 8 bit iki yönlü veri yolunu gerektirir. Oku ve yaz girişleri ve iki tane yonga seçim denetim girişi vardır. Bunlarla yonga ancak ve sadece mikroişlemci tarafından seçildiğinde izinlendirilir. Bir taneden fazla denetim girişinin bulunması, birden fazla yonga kullanıldığında adresin kodunun çözülmesi içindir. Oku ve yaz girişleri bazen R/W şeklinde 1 tane olabilir. Yonga seçildiğinde, hattaki ikili değer oku ve yazı belirler.

12.2(b) deki çizelge RAM yongasının işlemlerini göstermektedir. Birim ancak CS1 = 1 ve $\overline{CS2}$ = 0 iken çalışır. Dolayısıyla $\overline{CS2}$ = 0 ise CS2 = 1 olur, yani $\overline{CS2}$ = 0 ise çalışır. Eğer yonga seçim girişleri izinlendirilmemişse veya bunlar izinlendirilmiş fakat R veya W girişleri izinlendirilmemişse bellek tutulmuştur ve veri yolu yüksek direnç durumundadır. CS1 = 1 ve $\overline{CS2}$ = 0 ise bellek O veya Y kipine getirilebilir. WR girişi izinlendirilirse, bellek veri yolundan bir byte ı adres hatlarıyla verilen



Şekil 12.3 Tipik ROM yongası

adrese yerleştirir. RD girişi izinlendirilirse, seçilen byte ın içeriği veri yoluna yollanır. RD ve WR sinyalleri ile veri yolu buffer ları iki yönlü veri yoluna bağlanmışlardır.

Bir ROM yongası dışardan RAM yongası gibidir. Fakat ROM sadece okunur. Veri yolu çıkış kipindedir. ROM yongasının blok şeması şekil 12.3 te gösterilmiştir. Aynı büyüklükteki ROM yongası RAM yongasından daha fazla bit içerir. Çünkü ROM un içindeki ikili hücreler daha az yer kaplar. Bu nedenle şekildeki yonga 512 byte ROM dur. (RAM 128 idi).

ROM yongasındaki 9 adres hattı herhangi bir byte ı adresleyebilir. İki seçim girişi CS1 = 1 ve $\overline{CS2} = 0$ olmalıdır. Aksi halde veri yolu yüksek direnç durumundadır. O ve Y denetimlerine gerek yoktur. Çünkü yalnızca okunabilir bellektir. İki yonga seçim girişleri izinlenince adresin içeriği veri yoluna dolar.

Bellek Adres Haritası

Bilgisayarın tasarlayıcısı sisteme gerekli bellek miktarını hesaplamalı ve RAM ve ROM olarak tahsis etmelidir.Bundan sonra işlemci ile RAM ve ROM yongalarının tipine göre ve bellek büyüklüğüne göre bağlantı yapılabilir. Belleğin adreslemesi bir çizelge yardımıyla yapılır. Bu çizelge de her bir yonga ile tahsis edilen adresler bulunur. Çizelgeye bellek adres haritası denir. Her bir yonganın adres uzayının şekilsel gösterimidir.

Özel bir örnek ile gösterirsek, bilgisayarımızın 512 byte RAM ve 512 byte ROM gerektirdiğini varsayalım. RAM ve ROM yongaları şekil 12.2 ve 12.3 te verilenler olsun. Bellek adres haritası (bu yongalar için) çizelge 12.1 de verilmiştir. Bileşen sütunu kullanılan RAM ve ROM yongasını belirtir. Onaltılık adres sütunu her bir yonganın adres uzaylarını onaltılık olarak verir. Üçüncü sütunda adres yolu hatları verilmiştir. Her ne kadar adres yolunda 16 hat varsa da burada sadece 10 tanesi gösterilmiştir. 6

Çizelge 12.1 Mikro bilgisayar için bellek adres haritası

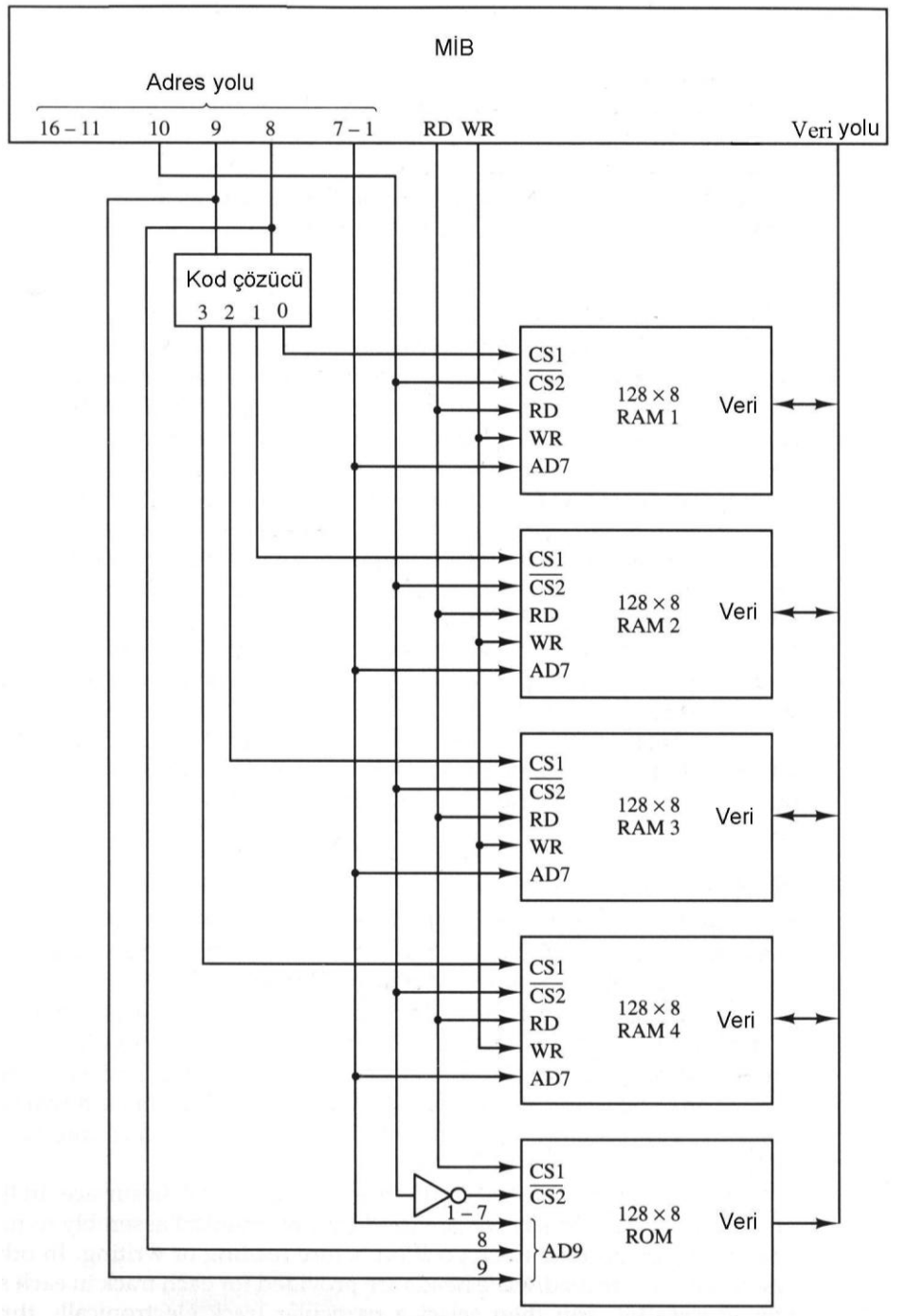
Bileşen	Onaltılık adres	Adres yolu									
		10	9	8	7	6	5	4	3	2	1
RAM 1	0000–07F	0	0	0	x	x	x	x	x	x	x
RAM 2	0080–0FF	0	0	1	x	x	x	x	x	x	x
RAM 3	0100–17F	0	1	0	x	x	x	x	x	x	x
RAM 4	0180–1FF	0	1	1	x	x	x	x	x	x	x
ROM	0200–3FF	1	x	x	x	x	x	x	x	x	x

tanesi bu örnekte kullanılmamış olup 0 varsayılmıştır. Adres hatları altındaki küçük x ler, her bir yongadaki adres girişlerine bağlanacaktır. RAM yongaları 128 byte olup 7 adres hattına ihtiyaç vardır. ROM yongası 512 byte olup 9 hatta ihtiyaç vardır. x ler daima düşük önemde adres hatlarına verilmiştir ve RAM için 1-7, ROM için 1-9 arasıdır. Şimdi RAM yongalarını her birine farklı adresler vererek ayırt etmeliyiz. Bu örnekte 8 ve 9 numaralı hatları 4 yonga için 4 farklı ikili sayıyı vermekte kullanacağız. Herhangi iki ve kullanılmayan hat da bu iş için kullanılabilirdi. Çizelgeden 9 düşük seviyeli hattın $2^9 = 512$ byte adresleyebildiği ve bunun RAM bellek uzayına eşit olduğu görülüyor. RAM ve ROM yongaları arasındaki ayırım başka bir veri yolu hattıyla yapılır. Burada bu iş için onuncu hat kullanılmakta bu hat 0 olunca MİB RAM ı, 1 olunca ROM u seçmektedir.

Eşdeğer onaltılık adresler adres yolu atamasıyla elde edilir. Sekizinci ve dokuzuncu hatlar 4 gurup oluşturur. Bu yüzden bir tek onaltılık basamakla ifade edilebilir. İlk onaltılık basamak 13 ten 16 ya kadar olan 4 tane hattı ifade eder ve daima 0 dır. Sonraki onaltılık basamak 9-12 hatları ifade eder. 11 ve 12 ci hatlar daima 0 dır. Onaltılık adresler bağlı x lerden hesaplanır. x ler 0000 dan 1111 e değer alırlar.

Bellekle MİB Bağlantısı

RAM ve ROM yongaları MİB ne veri ve adres yollarıyla bağlanmıştır. Adres yolundaki düşük önemdeki bitler yonga içindeki byte seçimi için, diğer bitler yonga seçimi için kullanılır. Bellek yongalarının MİB bağlantısı şekil 12.4 te verilmiştir. Bu şekil 512 byte RAM ve 512 byte ROM bağlantısı sağlar. Her bir RAM yongası düşük seviyeli 7 biti 128 byte tan birini seçmek için kullanılır. İstenen yonga sekizinci ve dokuzuncu hatlarla seçilir. Bunun için 2×4 kod çözücü kullanılır ve CS1 girişlerine gider. 8 ve 9 hatları 0 0 olunca ilk RAM yongası seçilmiş olur. RD ve WR hatları yongaların girişlerini denetler.



Şekil 12.4 Belleğin MİB ile bağlantısı

RAM ile ROM arasındaki seçim onuncu hatla sağlanır. RAM lar onuncu hattın 0, ROM ise 1 değeri ile seçilir. ROM yongasının CS girişine RD denetimi bağlanmıştır. 1 ile 9 arasındaki adres hatları kod çözücüye gitmeden ROM yongasına gelir. Bu şekliyle 0-511 adresleri RAM ve 512 ve 1023 adresleri ROM a aittir. ROM un veri yolu sadece çıkış içindir.

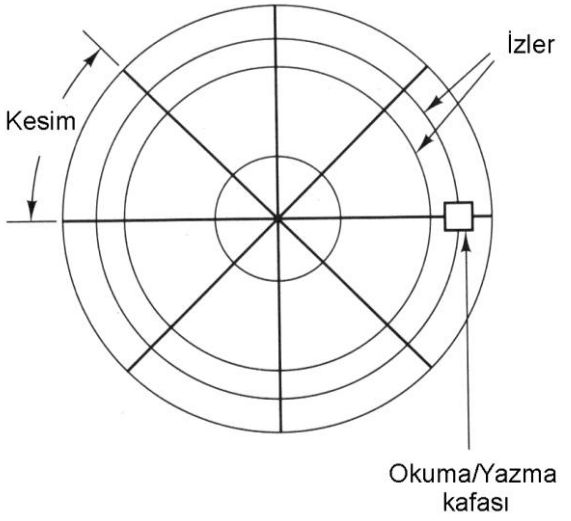
Örnek, bellek yongaları ile MİB arasındaki karmaşık bağlantıyı gösterir. Daha fazla yonga bağlanması için daha fazla kod çözücüye gerek vardır. Tasarımcı mutlaka bir bellek haritası çizmeli ve muhtelif yongalara tahsis edilen adres uzaylarını ve gerekli bağlantıları belirlemelidir.

12.3 Yardımcı Bellek

Bilgisayar sistemlerinde en çok kullanılan yardımcı bellek aletleri sabit disk ve teyplerdir. Başka aletlerden az olarak kullanılır. Sabit disk ve teyplerin çalışma mekanizmalarının anlaşılması için manyetik, elektronik ve elektro mekanik bilmek gerekir. Bu aletlerin çalışmaları çok karışık olmakla birlikte bunların mantıksal özellikleri birkaç parametre ile anlatılabilir ve karşılaştırılabilir. Herhangi bir aletin önemli karakteristikleri, erişim kipi, erişim süresi, aktarma oranı, kapasitesi ve fiyatıdır.

Bellekte bir adrese erişmek ve onun içeriğini almak için geçen ortalama süreye erişim süresi denir. Elektro mekanik aletlerde, örneğin sabit disk ve teyplerde erişim süresi, okuma yazma kafasının, okuma yazma yapacağı yere gitmesi için gerekli arama süresi ve aletten verinin aktarımı için gerekli aktarma süresinden oluşur. Arama süresi, aktarım süresinden çok uzundur. Bundan dolayı yardımcı bellekler, kayıtlar veya bloklar biçiminde tasarlanırlar. Aktarma oranı saniyede karakter veya kelime olarak tanımlanır. Ancak bu süre kafanın kaydın başına gelmesinden sonra hesaplanır.

Manyetik dram ve diskler işleyiş açısından oldukça benzerdirler. Her ikisi de manyetik kayıt ortamı ile kaplı yüksek hızda dönen yüzeylerden oluşur. Dramın dönen yüzeyi silindir diskin ki ise yuvarlak yassı bir yüzeydir. Kayıt yüzeyi sabit hızla döner ve giriş işlemleri esnasında başlatılmaz ve durdurulmaz. Bitler yazma kafası diye adlandırılan duran mekanizmadan geçtikçe yüzeye manyetik noktalar olarak kaydedilir. Depolanan bitler bir okuma kafasından geçtikçe yüzeye kaydedilen bir noktanın ürettiği manyetik alan değişimi ile tespit edilir. Bir diskteki kayıt için kullanılabilir alan miktarı aynı fiziksel büyüklükteki bir dramdakinden daha fazladır. Dolayısı ile bir disk üzerine, benzer ölçülerdeki bir dramdan daha fazla bilgi kaydedilebilir. Bundan ötürü daha yakın dönemdeki bilgisayarlarda diskler dramların yerini almıştır.



Şekil 12.5 Manyetik Disk

Manyetik Disk

Manyetik disk dairesel bir tabakadır. Manyetik alanlı plastik bir madde ile kaplanmıştır. Genellikle iki yüzü birden kullanılır ve ayrıca birkaç tabaka tek bir eksene bağlanmıştır. Her bir diskin alt ve üst kafaları vardır. Bütün diskler hızla döner ve durmazlar. Bitler manyetik yüzey üzerine depolanır. Bu noktalar disk üzerinde es merkezli değişik yarı çaplı daireler şeklindedir. Bunlara iz denir. İzler sektör adı verilen parçalara ayrılmışlardır. Çoğu sistemlerde aktarılan bilgiler bir sektör kadar olur (Şekil 12.5).

Bazı birimler her yüzey için tek kafa kullanır. Bu tip bir birimde iz adres bitleri kullanılarak kafa istenen iz üstüne getirilir, sonra okuma veya yazma yapılır. Diğer disk sistemlerinde her bir iz için ayrı bir kafa bulunur. Bu durumda hangi kafanın iş göreceği adres bitleri yardımı ile bir kod çözücü ile belli olur. Bu çok pahalı bir sistem olup ancak çok büyük makinelerde bulunur.

Sabit zaman izleri ile bitler ve sektörler bulunur. Disk sistemi adres bitleri ile belli olur. Adres disk numarasını gösterir. Disk yüzeyi, sektör ve iz numaraları da adres içinde yer alır. İstenen iz üzerine kafa yerleştikten sonra, dönen diskin dönerek istenen sektörün kafayla aynı hizaya gelmesi beklenir. Sektör gelince çok hızlı olarak aktarma yapılır. Çok kafa ve çok yüzey varsa aynı anda birçok iz ve yüzeyden okuma / yazma yapılabilir.

Çevreye yakın bir iz, merkeze yakın bir izden daha uzundur. Eğer bitler eş yoğunluklu olarak kaydedilirse bazı izler diğerlerinden daha fazla kayıt tutarlar. Bütün kayıtların eşit olması için bazı diskler değişen kayıt yoğunluğu kullanır. Dolayısıyla merkeze yakın izler daha yüksek yoğunlukta olur. Bu şekilde verilen bir sektör için bütün izlerdeki bit sayısı eşit olur.

Birime sabit olarak bağlı ve çıkarılamayan disklere sabit disk denir. Seyyar disklerin kullanıldığı alete disket sürücü denir

Manyetik Teyp

Bir manyetik teyp elektriksel, mekanik ve elektronik elemanları üzerinde taşır ve denetim mekanizması bulunmaktadır. Bitler bir çok iz üzerine manyetik spotlar olarak kaydedilir. Aynı anda 7 veya 9 bit kaydedilir. Genelde eşlik biti ile beraber 1 karakter. Oku veya yaz kafaları birer iz üzerinde durur.

Manyetik teypler durdurulabilir ileri veya geri sarılabilir. Ancak bu iş karakterler arasında veya karakter karakter yapılamaz. Kayıt adı verilen bloklar şeklinde başlatma ve durdurma olabilir. Teyp buralarda durdurulabilir. Boşluktayken teyp harekete başlar ve sabit hızla bir sonraki kayda kadar gider. Her kaydın bir tanıtma bit grubu vardır. Kaydın başında ve sonunda bulunurlar. Boştaki tanıtma bitlerini okuyarak teyp kayıt numarasını bulur. Sondaki tanıtma bitlerini okuyarak kaydın bittiğini ve boşluk geldiğini anlar. Kaydın tanıtımı kayıt numarası ve kayıttaki karakter sayısıdır. Kayıt uzunlukları sabit veya değişken olabilir.

12.4 İçerik Adreslemeli Bellek

Bir çok veri işleme uygulamalarında bellekte bulunan bir çizelgeden veri aranır. Bir birleştirici dil programı sembol-adres çizelgesini araştırır, sembolü bulur ve adresini okur. Bir hesap numarası bir dosyadan aranır ve buradan müşterinin adı ve bakiyesi okunur. Arama için arananların arka arkaya adreslerde bulunduğu bir çizelgeyi aramak gerekir. Arananı bulmak için çizelge elemanı aranan elemanla karşılaştırılır. Eşit olup olmadığına bakılır. Bir çok arama algoritması vardır.

İçerik Adreslemeli Bellek

Bellekte bulunan bir değirin aranıp bulunması, adres yerine verinin içeriği ile olursa arama süresi çok kısaltılabilir. Buna içerik adreslemeli bellek CAM denir. Bu belleğe aynı anda ve paralel olarak erişilebilir. Adrese veya yer belirtmeye gerek yoktur. Bellek kullanılmayan boş bir alanı bularak kelimeyi yerleştirir. Eğer bu bellekten bir kelime okunacak ise, kelimenin içeriği veya onun bir kısmı belirtilir. Bellek bu verilene

uygun ve onunla çakışan bütün kelimeleri belirler ve bunları okumak için işaret koyar.

Tasarımından dolayı bu bellek paralel aramalara uygundur. Ayrıca aramalar bütün bir kelime veya kelime içinde belli bir alan üzerinde yapılabilir. Bu bellek RAM bellekten çok daha pahalıdır. Çünkü her hücre saklama yeteneği dışında içeriğin karşılaştıracak devreler içerir. Dolayısıyla ancak arama süresinin çok kritik olduğu ve çok kısa olması gerektiğinde kullanılır.

Donanım Tasarımı

İçerik adreslemeli belleğin donanım tasarımı şekil 12.6 da (bakınız sayfa 458) verilmiştir. İçinde bir bellek dizisi ve mantık devreleri vardır. Bellek kelime başına *n* bit olarak *m* kelimeden oluşur. Değişken yazacı *A* ve anahtar yazacı *K*, *n* bit içerir. Çakıştırma yazacında *m* bit vardır. Her bir bellek kelimesi değişken yazacının içeriği ile paralel olarak karşılaştırılabilir. Değişken yazacının bitleriyle çakışan kelimeler, çakıştırma yazacında karşılık gelen biti 1 yapar. Çakışma işleminden sonra, çakıştırma yazacı içinde 1 yapılan bitler karşılık gelen kelimelerin çakıştığını belirtir. Okuma işlemi, çakıştırma yazacında karşılık gelen bitleri 1 olan kelimelerde sıralı olarak yapılır.

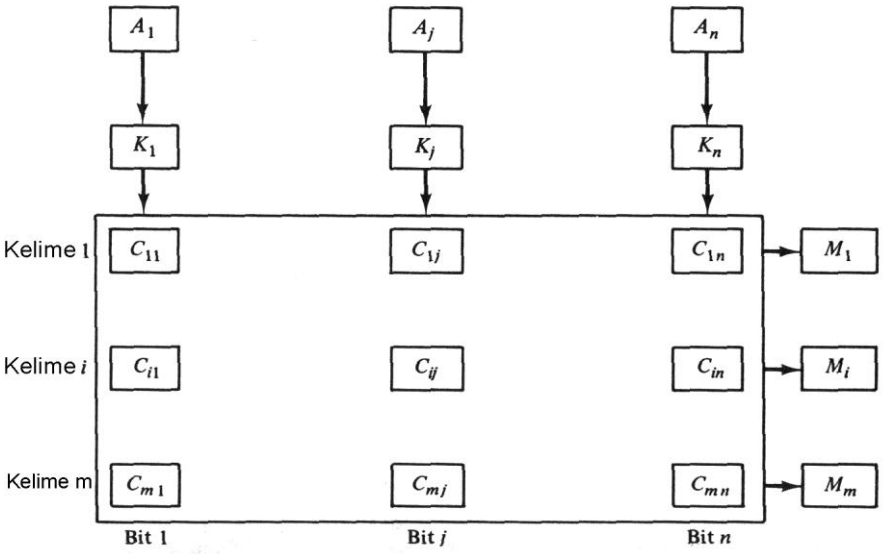
Anahtar yazacı Değişken kelimesindeki belli bir alanı seçmek için bir maskeleme oluşturur. Eğer anahtar yazacı bütün 1 lerden oluşuyorsa değişken yazacının tamamı her bir bellek kelimesiyle karşılaştırılır. Aksi durumda anahtar yazacında 1 olan bitlerin değişken yazacındaki karşılığı, bellek kelimelerinin karşılık gelen bitleriyle karşılaştırılır. Dolayısıyla anahtar yazacı bir maskeleme oluşturur. Bu maskeleme bellekte aranacak bilgi parçasını gösterir. Bu sayısal bir örnekle gösterilirse,

A ve *K* yazaçlarında aşağıda gösterilen değerler bulunsun. *K* nın durumundan dolayı *A* nın sadece 3 sol biti bellek kelimeleri ile karşılaştırılacaktır.

A	01 111100	
K	111 000000	
Kelime 1	100 111100	çakışma yok
Kelime 2	101 000001	çakışma var

Kelime 2 maskelenmeyen değişken bitleriyle çakışır.

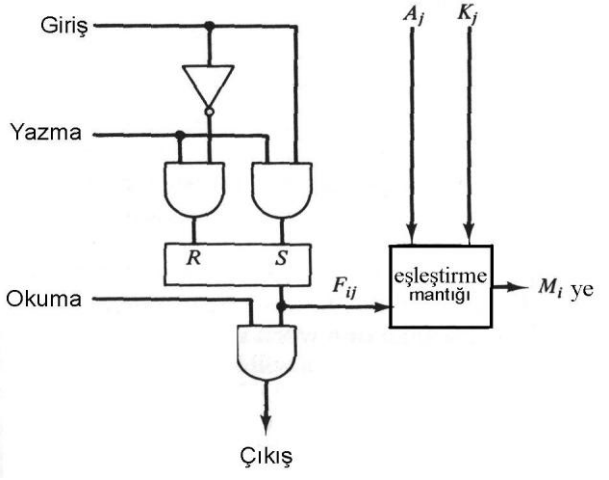
Bellek dizisi ile dış yazaçlar arasındaki bağlantı şekil 12.7 de gösterilmiştir. Dizi içindeki hücreler *C* ve iki indis ile gösterilmiştir. Birinci indis kelime numarasını, ikinci indis bit numarasını gösterir. C_{ij} *i* nci kelimedeki *j* inci bittir. Değişken yazacındaki *j* inci *A* biti, *j* inci sütundaki bütün bitlerle karşılaştırılır. (Tabi $K_j = 1$ ise). Bu iş $j = 1, 2, ..., m$ için



Şekil 12.7 Her kelime *n* bitlik içerik adreslemeli bellek

bütün sütunlara uygulanır. Eğer çakışma olursa, çakıştırma yazacındaki M_i biti 1 yapılır. Değişkenin bir veya daha fazla maskelenmemiş bitleri çakışmazsa M_i 0 yapılır.

C_{ij} hücresinin tipik iç tasarımı Şekil 12.8 de gösterilmiştir. Bir tane F_{ij} yaz-bozu, okuma, yazma ve çakıştırma devreleri vardır. Yaz işlemi sırasında giriş biti depolama hücresine aktarılır. Depolanan bit okuma işlemi



Şekil 12.8 İçerik adreslemeli belleğin tek bir hücresi

ile okunur. Çakıştırma mantığı depolama hücresi ile değişken yazacının maskelenmemiş bitini karşılaştırır ve bir çıkış yollar. Çıkışta bir karar vardır ve uygunsa M_i biti 1 yapılır.

Çakıştırma Mantığı

Çakıştırma mantığı her bir kelime için karşılaştırma algoritmasından türetilir. Önce anahtar bitleri ihmal edilerek, *A* içindeki değişken hücrelerdeki kelimelerle karşılaştırma yapılsın. *i* inci kelime *A* değişkenine eğer $A_j = F_{ij}$ (j = 1, 2 ... *n*) ise eşittir. İki bit ya ikisi de 1 veya ikisi de 0 ise eşittir. Boole fonksiyonu

$$x_j = A_j F_{ij} + A'_j F'_{ij}$$

dir. İstenen sağlanırsa $x_j = 1$ olur

A ya eşit olan *i* inci kelime için bütün x_j değişkenleri 1 olur. Bu da M_i bitini 1 yapar. Bunun Boole fonksiyonu şöyledir:

$$M_i = x_1 x_2 x_3 \ldots x_n$$

Şimdi karşılaştırma mantığına K_j anahtar biti de katılsın. Eğer $K_j = 0$ ise A_j ve F_{ij} bitlerinin, karşılaştırılmalarına gerek yoktur. Sadece $K_j = 1$ ise karşılaştırılırlar. Bu durum her bir terim VEYA lanarak elde edilir.

$$x_j + K'_j = \begin{cases} x_j & \text{eğer } K_j = 1 \text{ ise} \\ 1 & \text{eğer } K_j = 0 \text{ ise} \end{cases}$$

$K_j = 1$ olunca $K'_j = 0$ ve $x_j + 0 = x_j$ elde edilir. $K_j = 0$ olunca $K'_j = 1$ ve $x_j + 1 = 1$ olur. $x_j + K'_j$ terimi, ilgili bit çifti karşılaştırılmıyorsa 1 durumundadır. Çünkü her bir terim diğer terimlerle VE leneceğinden 1 in etkisi olmaz. Bitlerin karşılaştırılması sadece $K_j = 1$ ise etkilidir.

İçerik adreslemeli bellekte çakıştırma mantığı *i* inci kelime için aşağıdaki Boole fonksiyonu ile ifade edilebilir.

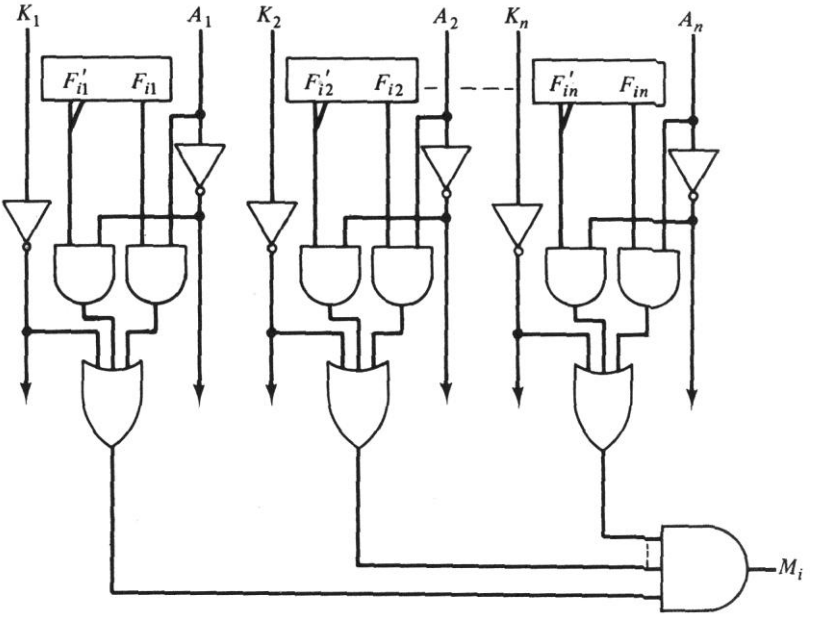
$$M_i = (x_1 + K'_1)\,(x_2 + K'_2)\,(x_3 + K'_3) \ldots (x_n + K'_n)$$

Eğer $K_j = 0$ ise her terim 1 olur. $K_j = 1$ ise terim 0 veya 1 olabilir. Bu da x_j ye bağlıdır. Eğer bütün terimler 1 ve $M_i = 1$ ise çakışma olur.

İlk tanıma uygun olarak içerik adreslemeli belleğin *i* inci açılımı

$$M_i = \prod_{j=1}^{n} \left(A_j F_{ij} + A'_j F'_{ij} + K'_j \right)$$

şeklinde yazılır. Her *i* = 1, 2, 3... *m* için *m* tane böyle fonksiyona gerek vardır.



Şekil 12.9 İçerik adreslemeli bellekte bir kelime için çakıştırma mantığı

Bir kelimeyi çakıştırma işini yapacak devre şekil 12.9 da gösterilmiştir. Her bir hücre 2 VE kapısı ve 1 VEYA kapısı gerektirir. *A* ve *K* daki tersleyen sadece birer defa konur. Sütundaki bütün bitler için kullanılır. Aynı kelimenin hücreleri içindeki VEYA kapılarının çıkışları, ortak VE kapısının girişlerine gider. Bununla M_i için çakışma sinyali üretilir. Eğer çakışma varsa $M_i = 1$ ve yoksa $M_i = 0$ dır. Eğer anahtar yazacı hep 0 ise M_i çıkışı A ya bağlı olmaksızın 1 olacaktır. Normal işlemler sırasında bundan kaçınılır.

Okuma İşlemi

Eğer bellekte birden fazla kelime maskelenmemiş değişken alanı ile çakışırsa bu kelimelerin çakıştırma yazacındaki karşılık gelen bitleri 1 olur. Dolayısıyla çakıştırma yazacının bitleri her zamanda 1 tane olarak taranır. Çakışan kelimeler sırayla okunur.

Uygulamaların çoğunda verilen bir anahtar alanında özdeş iki değer bulunmaz. Bu durumda tek bir kelime maskelenmemiş değişken alanıyla çakışır. M_i çıktısı okuma hattına doğrudan bağlanarak çakışan kelime otomatik olarak bildirilmiş olur. İçeriği sıfır olan kelimeleri dikkate almazsak, sıfır çıktısı çakışma olmadığı ve arananın bellekte olmadığını gösterir.

Yazma İşlemi

CAM belleğin bilgiyi aramak için önce o bilgiyi yazabilmesi gerekir. Bu belleğe yazma işlemi uygulamaya göre değişir Eğer belleğin tümüne yeni bilgi girişi toplu halde yapılacaksa, her kelime sırayla adreslenerek doldurulur. Bu durumda belleğimiz yazarken adreslenebilir bellek, okurken CAM olacaktır. Buradaki avantaj tıpkı RAM daki adresleme yeteneği olmasıdır. Çünkü m tane kelimesi olan bu bellek d tane ($2^d = m$) bitle adreslenebilir.

Eğer istenmeyen kelimeler silinecek ise bir zamanda bir tane olarak yeni kelimeler girilir. Yalnız bir yazaç ile işe yarayan ve yaramayan kelimelerin listesi tutulmalıdır. Bu yazaca bazen başlık yazacı denir. Bellekteki kelime sayısı kadar biti vardır. İşe yarayan kelimelerin bitleri 1, yaramayanlar 0 yapılır. Bir kelimenin silinmesi için başlık biti 0 yapılır. Yeni kelime yazılacağı zaman başlık yazacına bakılır ve ilk 0 yerine yeni kelime yazılır. Kelime yazıldıktan sonra başlık biti 1 yapılır. Bir kelimeyi tamamen silmek yani orayı boş yapmak için 0 lar girilir. Başlık biti 0 olan kelimeler, çakışmaya alınmamalıdır. Bunun için K yazacı ile maskelenir.

12.5 Ön Bellek

Kaynakların Yöreselliği

Tipik programların analizi, belleğe erişimin belli bir zaman aralığında bellekte bazı bölgelere yapıldığını göstermektedir. Buna kaynakların yöreselliği denir. Bu özelliğin nedenine bakılırsa; tipik bir program doğrusal bir çizgide hareket eder. Fakat sık sık döngüler ve alt program çağrıları olur. Bir döngüye girildiğinde MİB tekrar tekrar aynı buyrukları çağırır. Yani tekrar tekrar belleğin aynı bölgelerine hatta aynı adreslere gider. Bir alt programın her çağrılışında aynı buyruklar bellekten alınıp getirilir, dolayısıyla buyrukların al-getir durumunda, alt programlar ve döngüler belleğin daima bazı bölgeleriyle ilişkilidir. Daha az olmakla beraber verilerde de aynı durum bulunmaktadır. Dizilerin bakımını yapan işlemler daima dizinin bulunduğu bölgeye giderler yinelenen işlemler ve dizi işlemleri daima belleğin aynı bölgelerinde bulunurlar. Bunlar hep kaynakların yöreselliği gözlemleridir. Kısa zaman aralıklarında belleğin çoğunlukla hep aynı bölgeleri kullanılır. Diğer kısımlar ise nadiren kullanılır.

Programın çalışan kısımları ve verilerin o anda kullanılan kısımları küçük ve hızlı bir belleğe konursa ortalama bellek erişim süresi kısaltılabilir. Böylece programın icra süresi de kısalabilir. Böyle bir belleğe ön bellek denir. Bu bellek küçük ve hızlıdır. Erişim süresi ara belleğin 5 te biri ile 10 da biri arasındadır. MİB ile ana bellek arasında bulunur. (Şekil 12.1) Bellek hiyerarşisinde en hızlı bellektir. Hızı MİB bileşenlerine yakındır.

Ön bellek tasarımının temel fikri en sık kullanılan buyruk ve verileri bu bellekte tutmaktır. Böylece ortalama bellek erişim süresi ön bellek erişim süresine yaklaşır. Ön bellek küçük bir bellektir. Ancak kaynakların yöreselliğine uygun çalıştırıldığından bellek erişimlerinin çoğu buradan karşılanır.

Ön belleğin temel işlemi şöyledir. MİB belleğe erişmek istediğinde ön belleği inceler. İstenen ön bellekte ise buradan okunur. Eğer bu adres ön bellekte bulunmazsa o zaman ana belleğe gidilerek okunur. Sonra bu okunan bilginin bulunduğu blok ön belleğe aktarılır. Blok büyüklüğü erişilen kelimeye komşu yaklaşık 16 kelimeden oluşur. Böylece erişilene yakın verilere (veya buyruklara) yeniden erişmek kolay hale gelir.

Bulma Oranı

Ön belleğin verimi bulma oranı denilen bir sayı ile ölçülür. MİB belleğe erişmek ister ve aradığını ön bellekte bulursa bu bir bulmadır. Eğer aranan kelime ön bellekte değilse ve bellekte bulunursa, buna bir bulamama denir.

$$\text{Oran} = \frac{\text{Bulma sayısı}}{\text{Toplam erişim sayısı}}$$

veya

$$\text{Oran} = \frac{\text{Ön bellek erişim sayısı}}{\text{Ön bellek erişim sayısı + ana bellek erişim sayısı}}$$

olarak tanımlanır. Bunlar bazı programlarla deneysel olarak ölçülerek 0.9 veya daha yüksek oranlar rapor edilmiştir. Bu oranlar kaynakların yöreselliği ile ön bellek olayının doğruluğunu kanıtlar.

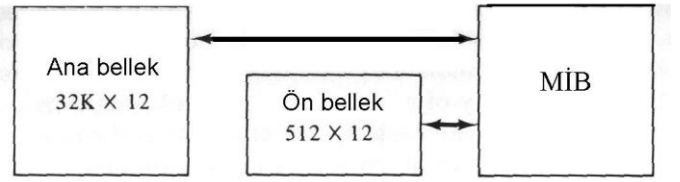
Bir bilgisayarda ortalama bellek erişim süresi ön bellek kullanılmasıyla azaltılabilir. Eğer bulma oranı yüksek ise MİB erişimlerinin çoğu ön belleğ olur ve ortalama süre ön belleğin erişim süresine yaklaşır. Örneğin ön bellek erişim süresi 100 ns olan bir bilgisayarın bellek erişim süresi 1000 ns olsun. Bulma oranı 0.9 ise ortalama bellek erişim süresi 200 ns olur. Bu benzer ve ön belleksiz bir bilgisayara göre çok iyi bir süredir.

Haritalama

Ön belleğin temel karakteristiği hızlı erişim süresi, yani kısa erişim süresidir. Dolayısıyla bir kelimenin ön bellekte aranma süresinin çok kısa olması gerekir. Ana bellekten ön belleğe verilerin aktarılmasına haritalama işlemi denir. 3 tip haritalama işlemi kullanılmaktadır.

1. İçerik adreslemeli haritalama
2. Doğrudan haritalama
3. Kümelenmiş içerik adreslemeli haritalama

Bunların anlaşılmasını kolaylaştırmak için şekil 12.10 da gösterilen bellek tasarımı örneği kullanılacaktır. Ana bellek 12 bitlik 32 K kelime-

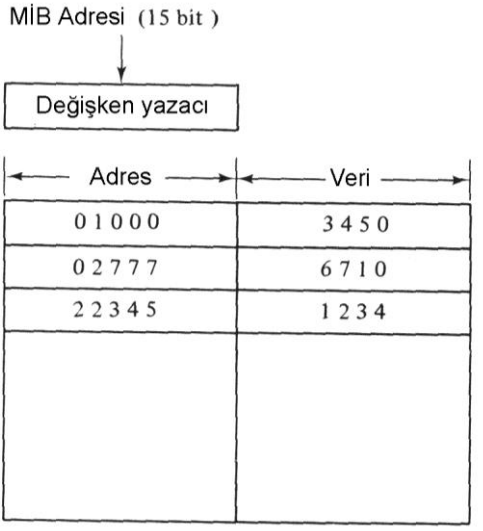


Şekil 12.10 Ön bellek örneği

den oluşsun Ön bellek ise bunların 512 tanesini taşıyor olsun. Ön bellekte bulunan her kelimenin ana bellekte bir kopyası bulunur. MİB her iki bellekle haberleşir. Önce ön belleğe 15 bit adres yollar. Eğer aranan kelime ön bellekte varsa buradan 12 bit veriyi okur. Eğer aranan kelimeyi ön bellekte bulamaz ise MİB kelimeyi ana bellekten okur ve kelimeyi ön belleğe aktarır.

İçerik Adreslemeli Haritalama

En hızlı ve çok yönlü kullanılabilen ön bellek tasarımı içerik adreslemeli (CAM) bellek alanıdır. (Şekil 12.11). Buradaki cam bellek hem adresi hem veriyi tutar. Böylece belleğin herhangi bir alanındaki veriler veya buyruklar ön belleğin herhangi bir yerine yazılabilir. Şekilde bellekte duran 3 adres beş basamak sekizlik düzende (15 bit) sayılarla ve bunlara



Şekil 12.11 Ön belleğin içerik adreslemeli haritalaması (tüm sayılar sekizlik düzende)

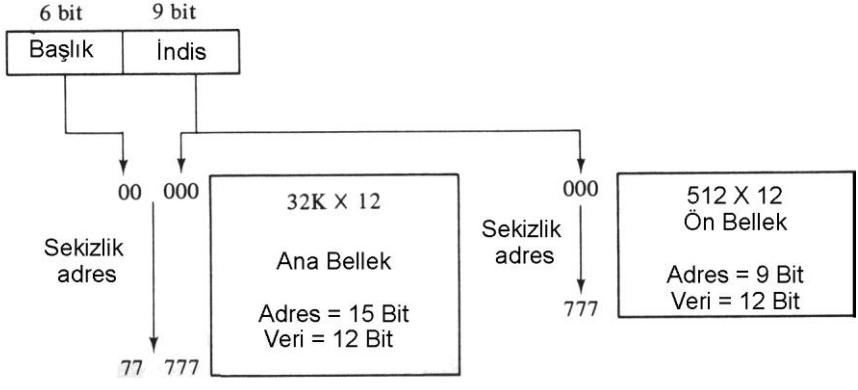
karşılık gelen veriler 4 basamak (12 bit) sekizlik düzende sayılarla verilmiştir. 15 bit MİB adresi değişken yazacına yazılır ve çakıştırma yapılır. Adres varsa veri okunur, sonra bulunduğu adres ve veri ön belleğe aktarılır. Eğer ön bellek doluysa burada ihtiyaç olmayan bir adres-veri çifti silinir. Hangi çiftin silineceği yer değiştirme algoritmasıyla bulunur. Algoritmayı ön belleği tasarlayan tasarımcı seçer. En basit yöntem dolap beygiri yöntemidir. Bu FIFO tipine denktir.

Doğrudan Haritalama

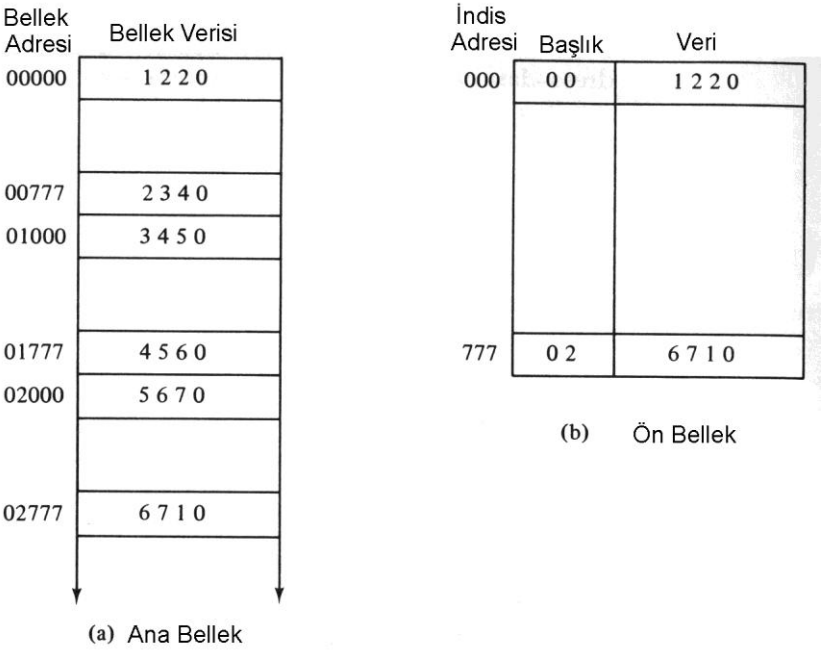
Başlık Alanı

CAM bellekler RAM a göre pahalıdırlar. Çünkü hücrelere bir çok devre eklenmiştir. RAM belleği ön bellek olarak kullanma durumu şekil 12.12 de gösterilmiştir. 15 bit MİB adresi 2 alana bölünür. 9 önemsiz bit indis alanını oluşturur. Kalan 6 bit ise başlık alanı olur. Şekilde görüldüğü gibi ana bellek adresleme için her iki alana gerek duyar. Halbuki ön bellekte adresleme için sadece indis alanı yani 9 bit yeterlidir. Dolayısıyla indis alanı, ön belleği adreslemek için gerekli bit sayısı kadar seçilir.

Genel olarak 2^k kelime ön bellek ta ve 2^n kelime bellekte bulunuyorsa n bit bellek adres alanı k ve $n - k$ olarak 2 alana ayrılır. k bit indis olup ön belleği adresler. Kelimelerin ön bellek içindeki durumu 12.13 (b) de gösterilmiştir. Ön bellek içindeki her bir kelime veriyi ve bunun ekli başlığını bulundurur. Yeni bir kelime ön belleğe getirildiğinde başlık bitleri veri bitlerinin yanına yazılır. MİB bir bellek isteminde bulunduğunda ön belleğe erişmek için indis alanı kullanılır. MİB adresinin başlık alanı, ön bellekten okunan kelimenin başlık alanı ile karşılaştırılır. İki başlık çakışırsa istenen kelime ön bellek içindedir. Çakışma yoksa istenen kelime bellekten okunur. Bu kelime daha sonra başlığıyla beraber ön belleğe



Şekil 12.12 Ana ve ön bellek adresleri arasındaki ilişki

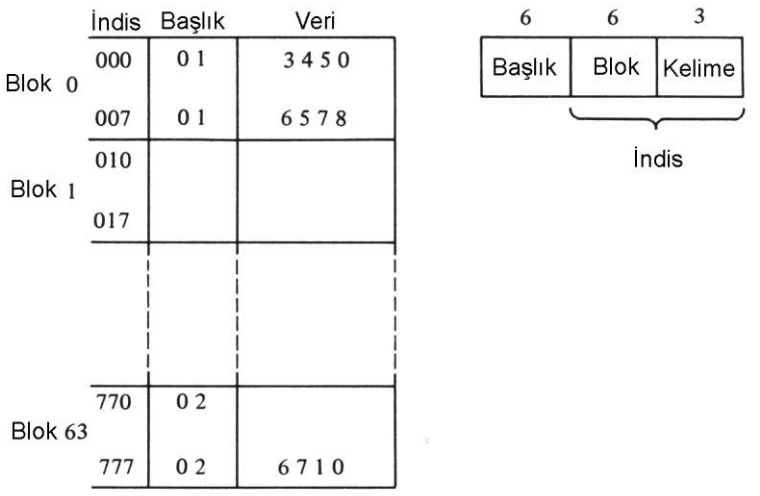


Şekil 12.13 Ön bellek tasarımında doğrudan haritalama

yazılır. Eski kelime atılır. Doğrudan haritalamada bit sayısı değişebilir. Çünkü iki veya daha fazla kelimenin indis alanları aynı ise ve başlık alanları farklı ise ve bunlara tekrar tekrar erişiliyorsa bu durum ortaya çıkar. Fakat bu tip kelimeler belleğin oldukça farklı bölgelerindedir.

Doğrudan haritalamanın nasıl çalıştığına bakılırsa, Şekil 12.13 te görüldüğü gibi sıfır adresinde bulunan kelime ön bellek içindedir. İndisi 000 ve başlığı 00 dır. Veri 1220 dir. MİB nin 02000 adresindeki kelimeyi istediğini varsayalım. İndis 000 dır. Ön belleğe gidilir ve başlıklar karşılaştırılır. Ön bellek başlığı 00 ve adres başlığı 02 dir. Çakışma olmaz. Ana belleğe gidilir ve 5670 verisi ön belleğe aktarılır. Bunun indisi 000 ve başlığı 02 dir.

Bu örnek sadece 1 kelimelik blok büyüklüğü için verilmekte aynı organizasyon 8 kelimelik blok için şekil 12.14 te verilmiştir. İndis alanı 2 parçaya bölünmüştür. Blok alanı ve kelime alanı 512 kelimelik bir ön bellekte her biri 8 kelimelik 64 blok vardır ($64 \times 8 = 512$). Blok numarası 6 bitlik bir alanla verilir. Kelime alanı ise blok içinde 3 bit yer kaplar. Ön bellek içindeki başlık alanı blok içindeki 8 kelime için aynıdır. Bir bulamama olduğunda 8 kelimenin tamamı bellekten ön belleğe aktarılır, bu da zaman alır. Fakat daha büyük bir blok büyüklüğü ile bulma oranının artacağı tahmin edilir.

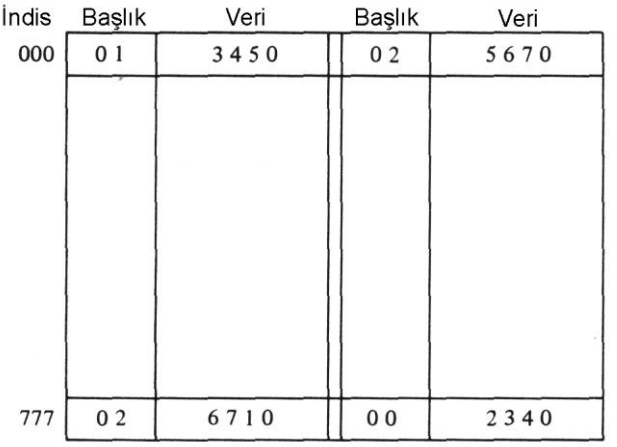


Şekil 12.14 8 kelimelik bloklar ile doğrudan haritalanmış ön bellek

Kümelenmiş İçerik Adreslemeli Haritalama

Daha önce belirtildiği gibi, doğrudan erişimli ön bellek tasarımında, adreslerinde aynı indis fakat farklı başlık değeri bulunan iki kelime, aynı anda ön bellekte bulunamazlar. Buda direk adreslemenin dezavantajıdır. Üçüncü bir ön bellek tasarımı kümelenmiş içerik adreslemeli haritalama adını taşır. Direk adreslemenin geliştirilmiş şeklidir. Aynı ön bellek kelimesi içinde birden fazla aynı indis adresli bellek kelimesi bulunabilir. Her bir veri kelimesi başlığı ile beraber depolanır ve bunlar bir küme oluştururur. Küme sayısı birden fazladır. Şekil 12.15 te iki yollu kümelenmiş içerik adreslemeli bir ön bellek tasarımı görülüyor. Her bir indis adresi iki veri kelimesini gösterir. Her bir başlık 6 bit ve her bir veri kelimesi 12 bittir. Kelime uzunluğu 2 (6 + 12) = 36 bit olur. 9 bit indis adresi ile 512 kelime taşıyabilir. Ön belleğin büyüklüğü 512 × 36 olur. Ana belleğin 1024 kelimesini olarak *k* büyüklüğünde kümelenmiş içerik adreslemeli ön bellekte bir kelimede ana belleğin *k* kelimesi bulunur.

Şekil 12.15 te gösterilen sekizlik sayılar şekil 12.13 (a) daki ana bellekte bulunan sayılardır. Ana belleğin 01000 ve 02000 adresindeki sayılar, ön bellekte 000 indis adresinde yer alırlar. Benzer şekilde 02777 ve 00777 adresleri ön belleğin 777 indis adresinde bulunur. MİB bir bellek isteminde bulunduğunda adresin indis kısmı ile ön belleğe erişilir. MİB adresinin başlık alanı ön bellek içindeki başlıklarla karşılaştırılır ve çakışmaya bakılır. Karşılaştırma ve arama mantığı CAM belleğin aynıdır. Küme sayısı arttıkça bulma sayısı artar. Çünkü aynı indis farklı başlık



Şekil 12.15 İki yollu kümelenmiş içerik adreslemeli haritalamalı ön bellek

numaraları ön bellekte bulunur. Ancak küme sayısının artması ön bellekteki bit sayısını arttırır ve karşılaştırma mantığını karışık hale sokar.

Yer Değiştirme Algoritmaları

Bir bulamama olduğunda veya ön bellek dolu ise bir tane başlık-veri elemanını değiştirmek gerekir. En çok kullanılan algoritmalar; rasgele değiştirme, FIFO, LRU dur. Rasgelede herhangi biri değiştirilir. FIFO, ön bellekte en çok kalanı değiştirir. LRU son kullanımı en eski olanı değiştirir. FIFO ve LRU nun kullanımı için ön belleğin her bir kelimesine bir kaç bit eklenmesi gerekir.

Ön Belleğe Yazma

Ön bellek tasarımının önemli bir kısmı belleğe yazma istemiyle ilgilidir. Okuma işlemi sırasında kelimeyi ön bellekte bulur ve okur. Ana bellek bu olayın dışındadır. Fakat işlem yazma işlemi ise sistemin bu işi yapmak için 2 seçeneği vardır.

Doğrudan Doğruya Yazma

Basit ve çok kullanılan yol, her yazma isteminde ana belleğe yazmaktır. Eğer yazılan adres ön bellek içinde ise ön bellekte beraberce güncelleştirilir. Buna doğrudan doğruya yazma adı verilir. Avantajı bellekle ön belleğin daima aynı bilgileri taşımalarıdır. Özellikle DMA işlemlerinde bu önemlidir. Bellek daima en yeni bilgileri taşır.

Geri Yazma

İkinci yöntem geri yazma dır. Bu yöntemde yazma isteminde sadece ön belleğe yazılır. Bu alan bir bayrakla belirlenir. Bu kelime ön bellekten silineceği zaman belleğe yazılır. Bunun nedeni ön bellekte bulunan bir değerin ön bellekten çıkmadan çok fazla değişmesidir. Ön bellekten çıkmadığı için bellekteki değerin güncellenmesi gerekmez. Ön bellekten

çıkmadığı sürece problem olmaz. Çıkarsa zaten belleğe yazılır. Araştırmalar tipik bir programın bellek erişimlerinin %10 ile 30 unun belleğe yazma olduğunu göstermektedir.

Ön Bellek İlklendirmesi

Geçerli Bit

Ön bellek tasarımının bir başka sorunu ilklendirme sorunudur. Ön bellek bilgisayara akım verildiği ve ana bellek, yardımcı bellekten programlar yüklendiği zaman ilklendirilir. İlklendirme işleminden sonra ön belleğin boş olduğu kabul edilir. Fakat gerçekte bazı geçersiz eski veriler bulunabilir. O zaman ön bellekteki her kelimeye geçerli veri biti koymak gerekir.

Dolayısıyla ilklendirme işlemi bütün geçerli veri bitlerini sıfır yapmakla sağlanır. Ana bellekten bir kelime yüklenince bu kelimenin geçerli veri biti 1 yapılır. Bu bit ön bellek tekrar ilklendirilinceye kadar öyle kalır. Eğer geçerli veri biti 0 olursa yani bir kelime buraya yazılır.

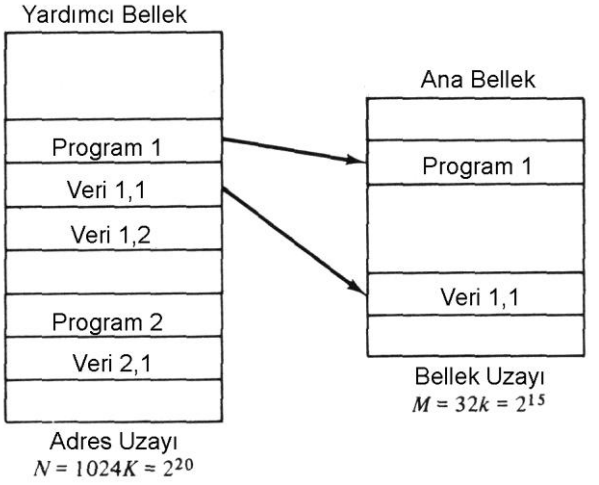
12.6 Sanal Bellek

Bellek hiyerarşisi içinde programlar ve veriler önce yardımcı belleğe (sabit disk) yüklenir. MİB tarafından ihtiyaç hissedilen program ve veri kısımları ana belleğe getirilir. Sanal bellek büyük bilgisayar sistemlerinde kullanılan bir kavramdır. Bununla kullanıcılar yardımcı belleğin kapasitesine eşit belleği olan SANAL ve bireysel makinede çalışıyormuş gibi olurlar. MİB tarafından verilen her adres, bir haritaya gider ve sanal adres, fiziksel adrese dönüştürülür. Sanal bellek olayı programcılara, oldukça küçük ana bellekle çalışsalar bile, emirlerinde çok büyük bellek varmış görüntüsü verir. Sanal bellek adres sistemi ile programdaki adresler, gerçek ana bellek adreslerine dönüştürülür. Bu iş, programlar MİB de icra edilirken dinamik olarak yapılır. Dönüşüm veya haritalama donanım tarafından bir haritalama çizelgesi yardımıyla otomatik olarak yapılır.

Adres Uzayı ve Bellek Uzayı

Adres Uzayı, bellek Uzayı

Kullanıcı tarafından kullanılan adrese sanal adres denir. Bu adreslerin kümesine adres uzayı adı verilir. Ana bellekteki bir adrese fiziksel adres veya yer adı verilir. Bunların kümesine bellek uzayı denir. Dolayısıyla adres uzayı programlar tarafından oluşturulan buyrukların ve verilerin adresleridir. Bellek uzayı ise gerçek bellek adresleri olup işlemler için doğrudan verilmelidir. Bir çok bilgisayarda adres ve bellek uzayları özdeştir. Sanal belleği olan bilgisayarlarda adres uzayı, bellek uzayından büyüktür.



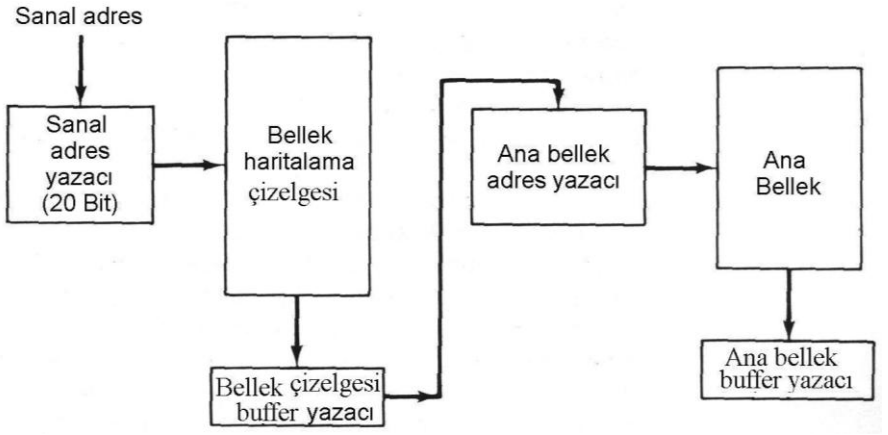
Şekil 12.16 Sanal bellek sistemdeki adres ve bellek uzayı arasındaki ilişki

Örnek olarak bir bilgisayar düşünülsün, belleği 32 K kelime olsun. Bu bilgisayarın 1024 K kelimelik bir yardımcı belleği olduğunu varsayalım. 2^{20} = 1024 K olduğundan, yardımcı belleğin kapasitesi 32 ana bellek eder. Adres uzayı N ve bellek uzayı M ile gösterilirse N = 1024 K ve M = 32 K dır.

Çok programlı bir bilgisayar sisteminde programlar ve veri, yardımcı bellek ile ana bellek arasında aktarılır. Ana bellek ise MİB nin isteklerine uyar. Varsayalım ki program 1, MİB tarafından halen icra edilmekte olsun. Program ve onun datasının bir kısmı yardımcı bellekten ana belleğe aktarılır. (Şekil 12.16). Program ve veri bölümleri, bellekte birbirine komşu ve birbirini takip eden adreste bulunmazlar. Çünkü bilgiler gidip gelmektedir. Boş yerler ise bellekte dağınık yerlerde bulunur.

Sanal bellek sisteminde, programcılara tüm adres uzayının emirlerinde olduğu söylenir. Ayrıca buyruğun adres alanı bütün adres uzayını adresleyebilecek sayıda bite sahip olmalıdır. Örnekte buyruğun adres alanı 20 bit olacaktır. Fakat fiziksel adresler en fazla 15 bitle adreslenebilir. Dolayısıyla MİB buyruklara ve verilere 20 bit adresle ulaşmaya çalışmalıdır. Halbuki yardımcı bellekten tek tek veri veya buyruk okumak çok uzun zaman alır. Dolayısıyla bunları ana bellekten okumalıdır. O halde şekil 12.17 gibi bir çizelge gerekir.

20 bit sanal adresi 15 bit fiziksel adrese dönüştürür. Dönüşüm dinamik bir olay olup, MİB tarafından verilen her adres anında fiziksel adrese dönüştürülür.



Şekil 12.17 Bir sanal adreste haritalama için bellek çizelgesi

Haritalama çizelgesi ayrı bir bellekte tutulabilir. Şekil 12.17 veya ana bellekte de tutulabilir. İlk halde ek bir bellek birimi ve zaman gerekir. İkinci durumda çizelge bellekte yer alır ve belleğe iki erişim gerektirir. Program yarı hızda çalışır. Üçüncü bir olasılık, CAM bellek kullanmaktır.

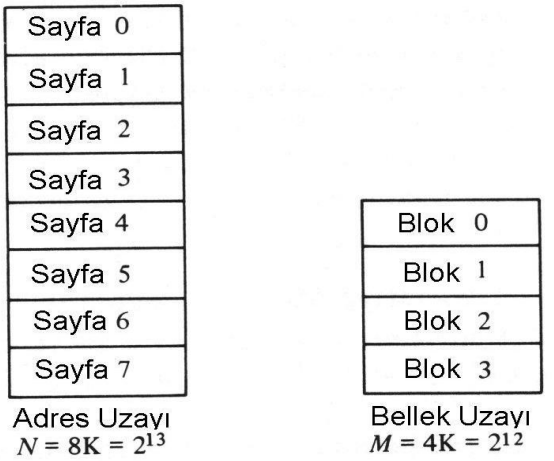
Sayfa Kullanarak Adres Haritalama

Sayfalar ve Bloklar, Sayfa Çerçevesi

Adres haritalama işlemi, adres ve bellek uzaylarının her biri sabit uzunlukta guruplara bölünürse çok basit hale getirilebilir. Fiziksel bellek blok adı verilen eşit büyüklükte guruplara bölünmüştür. Bu büyüklük 64 ile 4096 kelime arasında olabilir. Sayfa deyimi, adres uzayında eşit büyüklükteki grupların adıdır. Örneğin eğer bir sayfa veya blok 1 K kelime ise bir önceki örneğe göre adres uzayı 1024 sayfa ve ana bellek 32 bloktur. Her ne kadar büyüklükleri eşit ise de sayfa adres uzayı tasarımı blok bellek uzayı tasarımını gösterir. Programlarında sayfalara bölündüğü düşünülür. Program parçaları yardımcı bellekten ana belleğe bir sayfa büyüklüğüne eşit kayıtlar halinde aktarılır. Sayfa çerçevesi deyimi bazen bir bloğu göstermek için kullanılır.

Adres uzayı 8 K ve bellek uzayı 4 K olan bir bilgisayar düşünelim. Eğer her birini 1 K kelimelik gruplara bölersek 8 sayfa ve 4 blok elde ederiz. Şekil 12.18 Verilen herhangi bir zamanda adres uzayının 4 sayfaya kadar kısmı bellekte 4 bloktan birinde bulunur.

Adres uzayından bellek uzayına haritalama sanal adresin iki farklı sayının birleşimi alınmasıyla sağlanır. Bunlar sayfa sayısı ve sayfa içindeki satır sayısıdır.

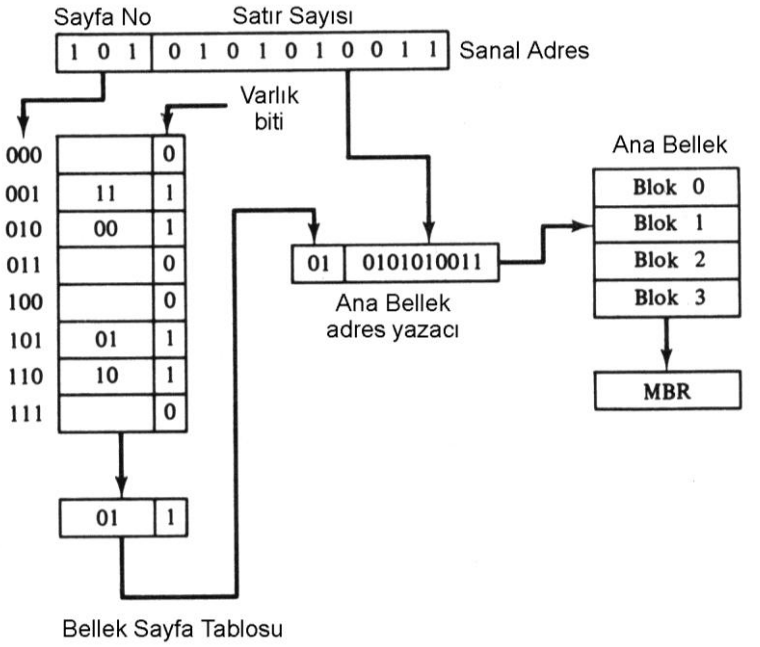


Şekil 12.18 1 K kelimelik gruplara bölünmüş adres ve bellek uzayı

Sayfa başına 2^P kelime düşen bir bilgisayarda p bit ile satır adresi belirlenir. Kalan yüksek seviyeli bitler ise sayfa numarasını belirler örneğin şekil 12.18 de bir sanal adres 13 bittir. Her sayfa $2^{10} = 1024$ kelime olduğundan kalan 3 bit 8 sayfa gösterir. 10 bit satır adresidir. Sayfa büyüklüğü blok büyüklüğüne eşitse satır adresi değişmez. Haritalama sayfa numarasından blok numarasına yapılır.

Sayfalı bir sistemde bellek haritalama çizelgesi şekil 12.19 da gösterilmiştir.

Bellek sayfa çizelgesinde 8 kelime vardır. Her sayfa için bir kelime sayfa çizelgesindeki adres sayfa numarasını gösterir. Kelimenin içeriği ise ana bellekte sayfanın bulunduğu blok numarasını verir. Çizelge, 1, 2, 5 ve 6 numaralı sayfaların ana bellekte 3, 0, 1 ve 2 numaralı bloklarda olduğunu gösterir. Çizelgedeki 1 bit, varlık biti olarak tanımlanır ve sayfanın yardımcı bellekten ana belleğe aktarılmış olup olmadığını gösterir. 0 aktarılmadığını yani sayfanın halen yardımcı bellekte olduğunu gösterir. MİB bellekten bir kelimeyi 13 bitlik sanal adres ile arar. Sanal adresin önemli 3 biti sayfa numarasını ve doğal olarak bellek sayfa çizelgesinde bir adres gösterir. Bellek sayfa çizelgesindeki bir kelimenin içeriği, sayfanın ana bellekteki blok numarasını verir (Tabii sayfa ana bellekte ise yani varlık biti = 1 ise). Bu içerik bellek çizelgesi buffer yazacına aktarılır. Bu 2 bit ana bellek adres yazacındaki önemli bitleri oluşturur. Satır sayısı ise sanal adresten doğrudan doğruya ana bellek adres yazacına aktarılarak ana bellek adresi hazırlanır. Ana belleğe gönderilen oku sinyali ile ana bellekteki kelimenin içeriği MBR de *CP* nin kullanımına hazır olur. Eğer bellek sayfa çizelgesindeki varlık biti 0 ise, sanal adresin gös-



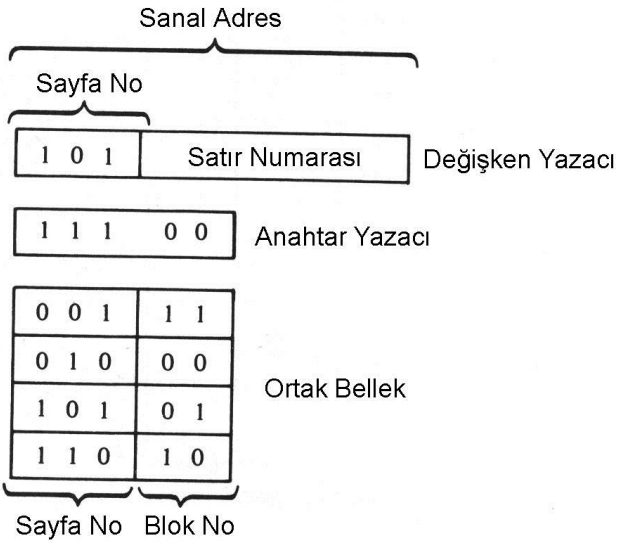
Şekil 12.19 Bir Sayfalık Sistemin bellek çizelgesi

terdiği kelime bellekte değildir. İşletim sistemi çağırılarak istenen sayfanın ana belleğe aktarılması ve bu işin hesaplamadan önce yapılması sağlanır.

İçerik Adreslemeli Bellek Sayfa Çizelgesi

RAM sayfa çizelgesi kullanışsızdır. Şekil 12.19 daki örnekle belleğin 8 kelimesine gerek vardı. Her sayfa bir tane. Bunlardan 4 tanesi daima boş kalır. Çünkü bellek ancak 4 blok olmakta. Genel olarak *n* sayfa ve *m* bloktan oluşan bir sistemde *n* kelime gerekir. Bunların sadece *m* tanesi doludur, diğerleri boştur. İkinci bir örnek 1024 K kelimelik adres uzayı ve 32 K kelimelik bellek düşünelim. Her sayfa ve blok 1 K kelime ise sayfa sayısı 1024 ve blok sayısı 32 dir. O halde bellek sayfa çizelgesi 1024 kelime olacak ve sadece 32 tanesinde varlık biti 1 olabilecektir. Herhangi bir anda en az 992 yer boş olacak ve kullanılmayacaktır.

Çizelgeyi oluşturmak için daha etkin bir yol kelime sayısının bellekteki blok sayısına eşit alınmasıdır. Bu halde çizelge büyüklüğü küçülür ve tamamen kullanılır. Bu çizelge bir CAM bellekte tutulabilir. Bu çizelgedeki her kelimede sayfa numarası ve buna karşılık gelen blok numarası bulunur. Kelimedeki sayfa alanı ile sanal adresteki sayfa alanı karşılaştırı-



Şekil 12.20 Bir içerik adreslemeli bellek sayfa çizelgesi

lır. Eğer çakışma olursa kelime bellekten okunur ve karşılık gelen blok numarası elde edilir.

8 sayfa ve 4 blok örneği düşünüldüğünde çizelge, 4 kelimelik CAM bellekte tutulsun (Şekil 12.20). CAM bellekteki dizide iki alan vardır. İlk 3 bit sayfa numarasını tutan alandır. Son iki bit ise blok numarasını tutar. Sanal adres değişken yazacındadır. Bu yazaçtaki sayfa numarası bütün sayfa numaraları ile karşılaştırılır. Eğer sayfa numarası bulunursa 5 bit kelime bellek MAR yazacına aktarılır. Çakışma olmazsa işletim sistemine çağrı yapılarak istenen sayfanın yardımcı bellekten getirilmesi sağlanır.

Sayfa Değiştirme

Sanal bellek sistemi donanım ve yazılımın bir birleşimidir. Bellek yönetim yazılım sistemi bütün yazılım işlemleri ile ilgili CAM bellek uzayının etkin kullanımını sağlar. Yönetim şunlara karar vermelidir. (1) Ana bellekten hangi sayfa yenisinin yazılması için çıkarılmalıdır, (2) Yeni bir sayfa ne zaman yardımcı bellekten ama belleğe aktarılmalıdır, (3) Sayfa ana bellekte nereye yazılmalıdır.Donanımsal haritalama mekanizması ile bellek yönetim yazılımı beraberce sanal bellek mimarisini oluştururlar.

Sayfa Hatası

Bir programın icrasına başlandığında bir veya daha fazla sayfa çizelgesi bunların yerini göstermek üzere düzenlenir. Program ana bellekte icra edilir. Fakat eğer program halen yardımcı bellekte bulunan bir kısma gerek duyarsa, bu sayfa bellekte olmadığından buna bir sayfa hatası adı verilir. Sayfa hatası olunca bu programın icrası, aranan sayfa belleğe

gelinceye kadar ertelenir. Yardımcı bellekten ona belleğe bir sayfanın aktarılması bir G/Ç işlemidir. İşletim sistemi bu görevi G/Ç işlemcisine verir. Denetim ise bu programı bekleyen başka bir programa geçer. Aranan sayfa için bellek bloğu tahsis işlemi bitip sayfanın aktarımı tamamlanınca eski program kaldığı yerden devam eder.

Sanal bellek sisteminde bir sayfa hatası olursa, MİB nin istediği sayfanın ana bellekte olmadığı anlaşılır. Yardımcı bellekten istenen sayfa ana belleğe aktarılır. Eğer ana bellek dolu ise bu yeni sayfaya yer açmak için bir bellek bloğundaki bir sayfanın kaldırılması gereklidir. Kaldırılacak sayfayı kullanılan sayfa değiştirme algoritması belirler. Yer değiştirmedeki temel amaç sayfayı kaldırmaktır.

FIFO

En çok kullanılan algoritmalar, FIFO ve LRU dur. FIFO algoritması en uzun süre bellekte kalan sayfayı kaldırır. Her defasında belleğe yeni bir sayfa gelir. Sayfayı tanıtan numara FIFO yığıtına yazılır. Bellekte boş blok kalmazsa FIFO yığıtı dolar. Yeni bir sayfa yüklenmesi gerektiğinde en eski girmiş olan sayfa kaldırılır. Bu sayfayı bulmak kolaydır, çünkü numarası yığıtın tepesindedir. FIFO yer değiştirme yöntemi çok kolay kurulabilir. Dezavantajı bazen kaldırılan sayfalar çabucak tekrar yüklenir.

LRU

LRU yöntemi kurulması çok zor bir yöntemdir. Fakat son kullanma zamanı en eski olan sayfa kaldırılmaya en iyi adaydır. FIFO dan iyidir. Her sayfaya bir sayaç konur. Sayfa arandığında sayaç sıfırlanır. Eşit aralıklı zamanlarda sayfaların sayaçları 1 arttırılır. LRU sayfası, sayacı en yüksek olan sayfadır. Bu sayaçlara yaşlanma yazaçları denir. Çünkü sayıları yaşlarını gösterir. Yani en son çağrıldıkları zamandan beri geçen zamanı gösterir.

12.7 Bellek Yönetim Donanımı

Çoklu bir işletim sisteminde, bir çok program bellekte bulunur. Ayrıca programların ve verilerin sürekli gidip gelmesi gerekir. Bir programın kaplayacağı alan değişir. Bunların sağlanması ve bellekte olup çalışmayan programların korunması gerekir. Bilgisayar belleği istekleri, çoklu programlama yapan bir sistemde bir bellek yönetim sistemi ihtiyacını doğurur. Bellek yönetim sistemi yazılım ve donanım işlemlerinin bir bileşkesidir. Bununla bellekte bulunan programlar idare edilir. Bellek yönetim yazılımı bir çok bilgisayarda bütün işletim sisteminin bir parçasıdır. Burada sadece donanımdan bahsedilecek.

Bellek yönetim biriminin parçaları şunlardır:

1. Mantıksal bellek adreslerini fiziksel adreslere dönüştüren dinamik bir dönüşüm parçası

2. Bellekte bulunan programların farklı kullanıcılarca kullanılmasını sağlayan bir parça.
3. Bilginin yetkisiz kullanıcılardan korunması ve kullanıcıların işletim sisteminin değişen fonksiyonlarından korunması

Adres haritalama işlemi 12.6 da anlatılan haritalama işlemine benzer.

Kesim

Sanal sistemde kullanılan sabit sayfa büyüklüğü, program büyüklüğü ve programların mantıksal yapısı nedeniyle bazı zorluklar oluşturur. Programları, verileri kesim denilen mantıksal parçalara bölmek uygun olur. Bir kesim mantıksal olarak bağlı buyrukların oluşturduğu bir gruptur ve bir adla anılır. Kesimler programcı veya işletim sistemi tarafından oluşturulur. Kesim örnekleri bir alt program, bir veri dizisi, bir sembol çizelge su veya bir kullanıcı programı olabilir.

Bazı ortak programların kullanımı çoklu programlamanın bir parçasıdır. Örneğin herkes aynı derleyiciyi kullanır. Sistem programları da bellekte durur ve herkesçe kullanılır. Herkes kendine ayrı bir kopya yapmaz.

Çoklu programlamanın üçüncü parçası bir programı diğerlerinin istenmeyen etkisinden korumaktır. Örneğin bir programcının yetkisiz olarak başkasının programını kopyalamasını önlemek gibi. Bir diğer koruma sıradan bir kullanıcının sistemi değiştirmesini önlemek. Bunun için bazı programlar gizlidir.

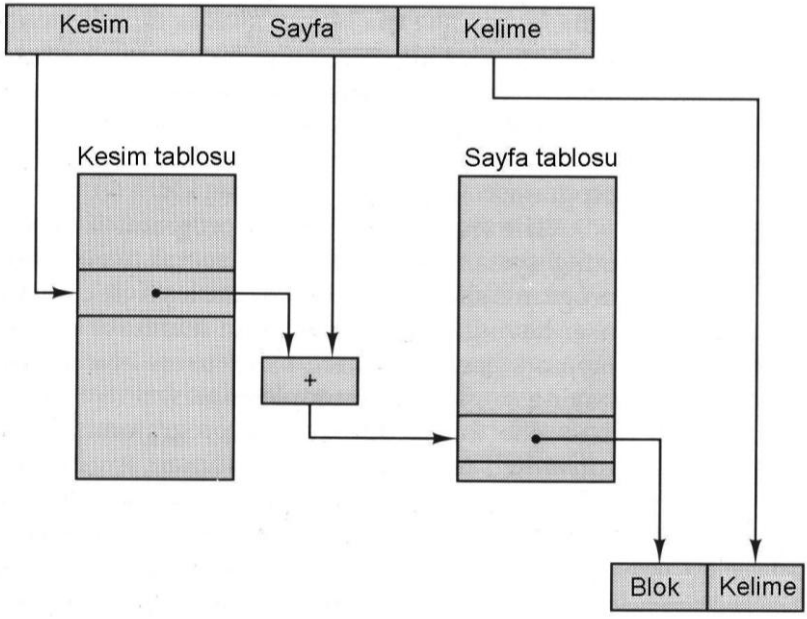
Mantıksal Adres

Kesimlenmiş bir programın adresine mantıksal adres denir. Bir sanal adrese benzerdir. Ancak mantıksal adres değişken uzunluklu kesimlere göre adreslenmiştir. Mantıksal adres fiziksel adresten büyük olabilir. Fakat bazen eşit veya küçükte olabilir. Paylaşılan programlar her bir kullanıcının mantıksal adres uzayında yer alırlar. Böylece tek bir fiziksel kopya paylaşılabilir. Burada yine mantıksal-fiziksel adres dönüşünü gerekir.

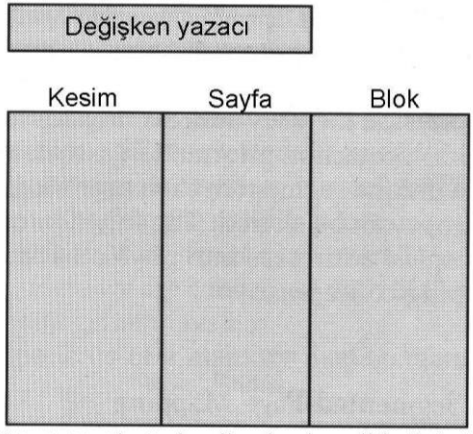
Kesimlenmiş-Sayfa Haritalaması

Mantıksal adres uzayının özelliği değişen uzunlukta kesimler olmasıdır. İcra edilen programın ihtiyacına göre bu kesim büyüyebilir veya küçülebilir. Kesim uzunluğunu belirlemekte kolay bir yol kesimi sayfa olarak ifade etmektir. Şekil 12.21 deki mantıksal adrese bakılırsa 3 kısma ayrılmıştır. Kesim alanı kesim numarasını belirtir. Sayfa alanı kesim içindeki sayfayı ve kelime alanı sayfa içindeki kelimeyi belirtir. k bitlik sayfa alanı ile 2^k sayfaya kadar adreslenebilir. Bir kesim numarası ile bir sayfa veya en fazla 2^k sayfa olabilir. Dolayısıyla kesimin uzunluğu, ona bağlı sayfa sayısı kadardır.

Mantıksal adresin, fiziksel adrese dönüşümü iki çizelge yardımıyla yapılır. Şekil 12.21 (a). Mantıksal adresin kesim numarası kesim çizelgesi



(a) Mantıksaldan fiziksel adrese haritalama



(b) Dönüştürülmüş içerik adreslemeli bellek (TLB)

Şekil 12.21 Kesimlenmiş sayfa bellek yönetim biriminin haritalaması

için olan adresi belirler. Kesim çizelgesindeki giriş, sayfa çizelgesi için göstergedir. Sayfa çizelgesi tabanı mantıksal adresle verilen sayfa numarasına eklenir. Toplam sayfa çizelgesi içinde bir giriş bir gösterge adresi-

dir. Sayfa çizelgesinde bulunan değer fiziksel bellekte bir blok numarasıdır. Blok alanı ile kelime alanının arka arkaya yazılarak birleştirilmesi son fiziksel adresi verir.

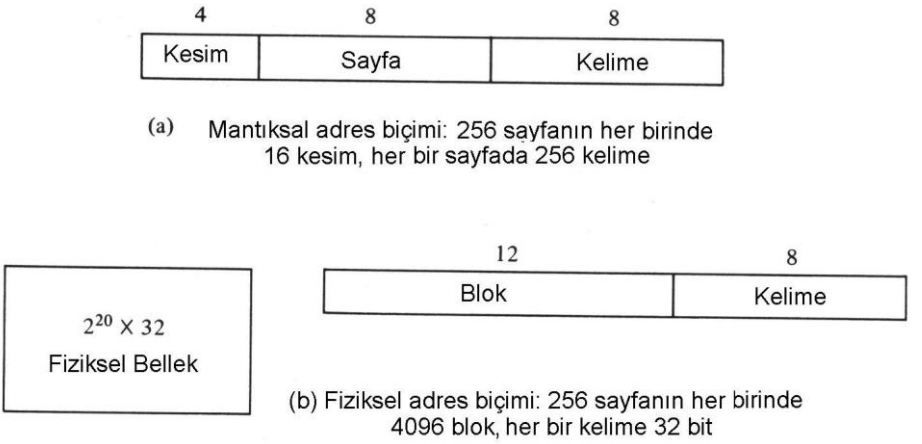
İki haritalama çizelgesi iki ayrı küçük bellekte tutulabilir veya ana bellekte tutulabilir. Her iki durumda MİB den belleğe bir geçiş isteği, belleğe 3 erişim gerektirir. Biri kesim çizelge sinden, biri sayfa çizelgesinden, üçüncüsü ana bellekten. Bu durum sistemi önemli biçimde yavaşlatır. Halbuki normal bir sistemde belleğe bir istek bir tek geçiş gerektirir. Bu durumdan kaçınmak için CAM bellek kullanılır ve en son bellek erişimlerinin adresleri burada tutulur. (Bu tip belleğe dönüştürülmüş içerik adresli bellek (TLB)) denir. İlk olarak bir blok işaret edildiğinde, karşılık gelen kesim ve sayfa numarası işlemi verilen kesim ve sayfa numarası ile yapılır. Eğer başarılı olunursa dönüşüm sadece CAM belleğin hızı kadar gecikir. Eğer çakışma olmazsa şekil 12.21(a) daki yavaş çizelgeli dönüşüm kullanılır. Sonuç CAM belleğe, sonra kullanılmak üzere aktarılır.

Sayısal Örnek

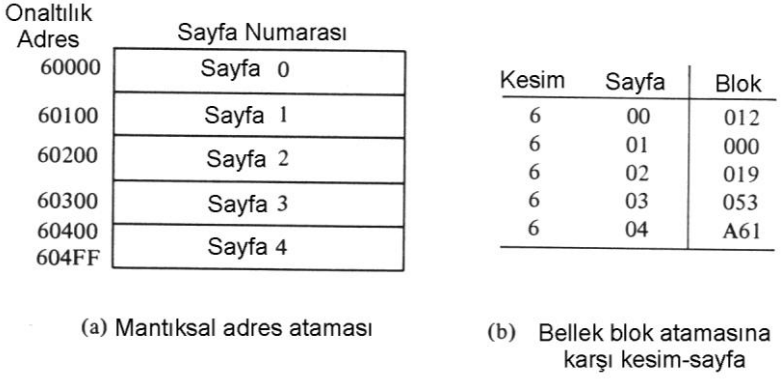
Sayısal bir örnek bellek yönetim biriminin işlevini daha iyi anlatır. Şekil 12.22 (a) da gösterildiği gibi 20 bit bir mantıksal adres düşünülsün. (12.22.a) Her bir sayfa 256 kelime, her biri 256 sayfalık 16 kesim 4 bit kesim numarası 16 mümkün kesimden birini belirler. 8 bit sayfa numarası 256 sayfadan birini belirler. 8 bit kelime alanı 256 ya kadar sayfa büyüklüğünü belirler. Bu konfigürasyon her kesimi n istenen sayıda sayfası olmasını sağlar (en fazla 256 ya kadar). En büyük kesimde 256 sayfa vardır. Toplam 64 K kelime yapar ($256 \times 256 = 64$ K)

Fiziksel bellek şekil 12.22 (b) ile gösterilmiştir. Her biri 32 bitlik 220 kelime içerir. 20 bit adres iki alana ayrılmıştır. 12 bit blok numarası ile 8 bit kelime numarası. Dolayısıyla fiziksel bellek 4096 bloğa bölünmüştür. Her bir blokta 256 kelime vardır. Mantıksal adresteki bir sayfanın fiziksel bellekte karşılık gelen bir bloğu vardır. Hem mantıksal hem fiziksel adres 20 bittir. Bellek yönetim biriminin olmaması halinde, MİB den gelen 20 bit adres, doğrudan doğruya fiziksel bellekte kullanılabilir.

Belleğe yüklenen bir programın 5 sayfaya sığacağı varsayılsın. İşletim sistemi bu programa 6 numaralı kesimden 0 dan 4 e kadar sayfaları tahsis etsin. Şekil 12.23 (a). Programın adres büyüklüğü 60000 ile 604FF arasındadır. Program fiziksel belleğe yüklenince, işletim sisteminin boş bulacağı 5 bloğa yüklenir. Sayfa ve buna karşılık gelen blok numarası bir çizelgeye yazılır. Şekil 12.23 (b). Bu çizelgeden alınan bilgiler şekil 12.24(a) da gösterilmiştir.

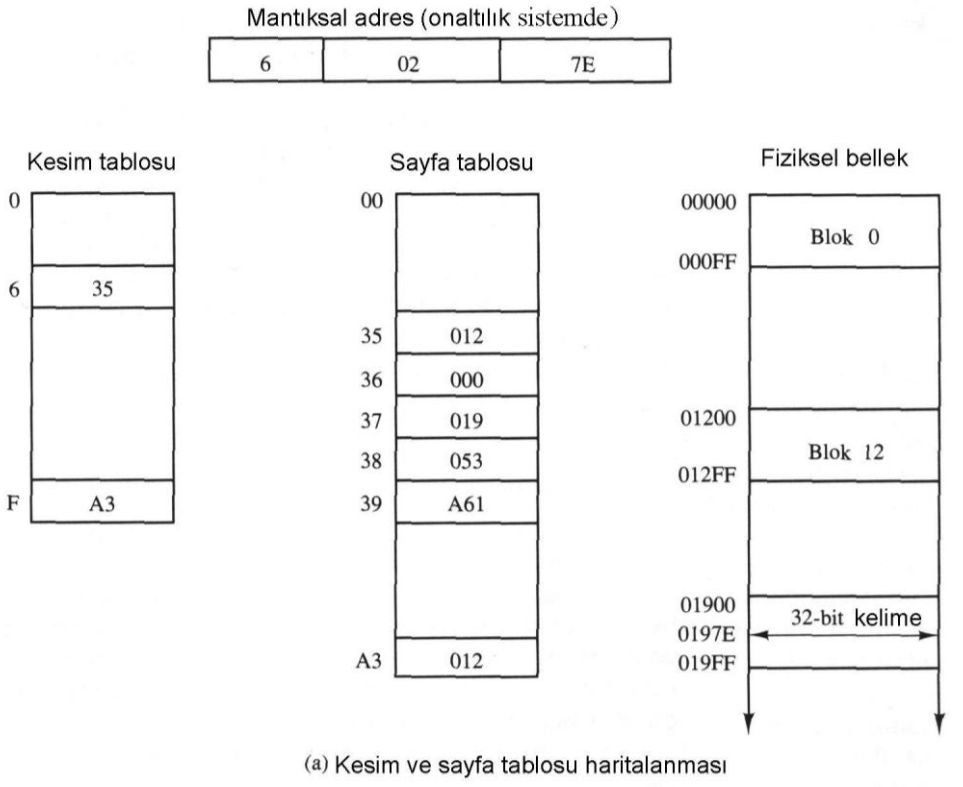


Şekil 12.22 Mantıksal ve fiziksel adrese bir örnek.



Şekil 12.23 Mantıksal ve fiziksel adres atama örneği

Şekil 12.24 te verilen özel mantıksal adrese bakılırsa, 20 bit adres ve 5 basamak onaltılık sayı olarak verilmiştir. Bu adres altıncı kesimin *n* ikinci sayfasında 7E numaralı kelimeyi gösteriyor. Kesim çizelgesinin altıncı kesimi sayfa çizelgesinin 35 adresinden 39 uncu adrese kadar yer alır. Altıncı kesimin ikinci sayfası 35 + 2 = 37 inci sayfadadır. Sayfa çizelgesinden bulunan fiziksel blok adresi 019 dur. 7E kelime adresi 19 uncu blokta 20 bit adresleme ile 0197E olur. Kesim 6 nın 0 ıncı sayfası 12 inci bloğa, sayfa 1 ise 0 ıncı bloğa karşılık gelir. Şekil 12.24(b) deki CAM bellek kesim 6 nın 2 ve 4 üncü sayfalarını gösteriyor. Bunların tekabül eden blok numaraları da bellekte görülüyor.



(a) Kesim ve sayfa tablosu haritalanması

Kesim	Sayfa	Blok
6	02	019
6	04	A61

(b) İçerik adreslemeli bellek (TLB)

Şekil 12.24 Mantıksaldan fiziksele bellek dönüşüm örneği (tüm sayılar onaltılık sistemde)

Bu örnekten görüleceği gibi bellek yönetim sistemi her bir kesime istenen sayıda sayfa atayabilir. Her bir mantıksal sayfa, fiziksel bellekteki herhangi bir bloğa tekabül edebilir. Sayfaların, fiziksel bellekteki bloklara atanabilmesi bellek ihtiyaçlarına bağlıdır. Tek değişecek şey sayfa çizelgesindeki blok numarasıdır. Kesimler büyüyebilir veya küçülebilir. Değişmeleri birbirlerinden bağımsızdır. Farklı kesimler, eğer birçok kullanıcı aynı programı paylaşmak isterse, aynı bloğu kullanabilirler. Örneğin fiziksel bellekteki 12 numaralı blok, 50000 den F00FF ye kadar ikinci bir mantıksal adrese atanabilir. Bu 15 numaralı kesimde 0 numaralı sayfadır. Şekil 12.24(a) da görüldüğü gibi 12 numaralı bloğa karşılık gelir.

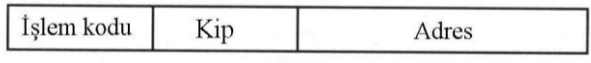
Bellek Koruması

Bellek koruması fiziksel veya mantıksal adreslere uygulanabilir. Fiziksel adreslerden bellek koruması için, bellekteki her bloğa koruma bitleri tahsis edilir. Bu bitler, bu bloğa erişmek için geçerli geçiş şifresini taşır. Bir bloktan başka bir bloğa bir sayfanın aktarılmasında blok koruma bitlerinin yenilenmesi gerekir. Daha iyi koruma fiziksel adresten çok mantıksal adres uzayı üzerinde olur. Koruma bilgisi kesim çizelgesi içinde veya bellek yönetim donanımının kesim yazacında tutulabilir.

Kesim çizelgesindeki veya kesim yazacındaki her girişe tanımlayıcı adı verilir. Bir tanımlayıcıda taban adres alanı dışında koruma amaçlı bir veya iki ek alan bulunur. Bir tanımlayıcının tipik biçimi şekil 12.25 te gösterilmiştir.

Taban adres alanı kesimlenmiş sayfa düzeninde sayfa çizelgesi adresinin tabanını verir. Veya kesim yazaç düzeninde blok taban adresini verir. Bu adres mantıksaldan fiziksel adrese haritalamada kullanılan adrestir. Uzunluk kesime atanmış en büyük sayıda sayfa sayısı ile kesim büyüklüğünü verir. Uzunluk alanı mantıksal adresteki sayfa numarası ile karşılaştırılır. Eğer sayfa numarası kesim uzunluğu sınırının dışına düşerse büyüklük hatası meydana gelir. Dolayısıyla verilen bir program ve verisi ona işletim sistemi tarafından verilen bellek miktarını taşamaz.

Kesim tanımlayıcı alanındaki koruma alanı erişim haklarını belli bir kesim için belirler. Kesimlenmiş sayfa tasarımında, sayfa çizelgesinde her giriş kendi özel koruma alanına sahiptir. Bununla her sayfanın erişim hakları belirlenir. Koruma bilgileri tanımlayıcı içine, işletim sisteminin



Şekil 12.25 Tipik kesim tanımlayıcı biçimi

ana denetim programı tarafından yazılır. Bellekte duran programların korunması için bazı haklar;

1. Okuma ve yazma hakları
2. Sadece okuma hakkı (yazma koruması)
3. Sadece kullanma hakkı (program koruması)
4. Sistem koruma (işletim sisteminin koruması)

Okuma ve yazma koruması, programın buyrukları içindir. Yazma koruması, ortak kullanılan yardımcı program ve kütüphane yordamları içindir. Bunlar belleğin öyle alanlarında tutulur ki herkes tarafından ortaklaşa kullanılabilir. Her program tarafından okunabilir fakat yazılamaz. Böylece başka programlar tarafından değiştirilmeleri önlenir.

Yalnızca icra yetkisi programların kopyalanmalarını önler. Bu koruma kesimi buyrukları al-getir evresinde korur. Fakat icra evresinde korumaz. Dolayısıyla bu programlar kullanıcılar tarafından kullanılabilir, fakat kopya edilemez.

İşletim sisteminin bazı parçaları herhangi bir anda bellekte bulunur. Bunların yetkisiz kişilerin erişiminden korunması gerekir. Bütün işletim sistemi programları tanımlayıcıları içinde bu koruma vardır. İşletim sistemi kesimlerine erişilmesi önlenir.

SORULAR

12-1. **a.** 2048 byte lık bir bellek elde etmek için 128 × 8 RAM yongalarından kaç tane gereklidir?
b. 2048 byte belleği adreslemek için kaç hat gereklidir? Bunlardan kaç tanesi tüm yongalarda ortaktır?
c. Yonga seçimi için kaç hatlık kod çözülmeli? Kod çözücülerin büyüklüğünü belirtin.

12-2. Bir bilgisayar 1024 × 1 lik yonga kullanıyor.
a. 1024 byte lik bir bellek elde si için kaç tane yonga gereklidir ve adres hatları nasıl bağlanmalıdır?
b. 16 K byte lık bir bellek elde si için kaç tane yonga gereklidir? Yongaların adres yoluna nasıl bağlanacaklarını açıklayınız.

12-3. 1024 × 8 bitlik bir ROM yongasının 4 seçim girişi var ve 5 Volt bir güç kaynağı ile çalışıyor. IC paketinin kaç ucu olmalıdır? ROM un blok şemasını çizerek ve adlarını yazarak tüm giriş ve çıkış uçlarını gösterin.

12-4. Şekil 12.4 teki bellek sistemini 4096 byte ROM a genişletin. Bellek adres haritasını yazın ve kod çözücünün büyüklüğünü gösterin.

12-5. Bir bilgisayar 256 × 8 lik RAM ve 1024 × 8 lik ROM yongaları kullanıyor. Bilgisayar 2 K byte RAM ve 4 K byte ROM gerektiriyor. Ayrıca dört yazaç için dört tane ara yüz gerekir. Bellek adreslemeli G/Ç

düzeni kullanılıyor. Adres yolunun en önemli 2 biti RAM için 00, ROM için 01 ve ara yüz yazaçları için 10 olarak seçiliyor

a. Kaç tane RAM ve ROM yongası gerekir?
b. Sistem için bellek adres haritası çizin.
c. RAM, ROM ve ara yüzler için adres büyüklüklerini onaltılık düzende verin.

12-6. 8 bit bir bilgisayarın 16 bit adres yolu vardır. Adres yolunun ilk 15 hattı belleğin 32 K byte lık parçalarını seçmek için kullanılır. Yüksek seviyeli bit, veri yolunun içeriğini alacak yazacı seçmek için ayrılmıştır. Böyle bir sistem, toplam 256 K byte lık bellek ve her bir parça 32 K byte olan bir sisteme genişletmek için nasıl kullanırız?

12-7. Bir sistem diskinin parametreleri:

T_0: bir kafayı iz üzerinde yerine yollamak için ortalama zaman
R: diskin dönme hızı, saniyede dönüş sayısı olarak
N_t: iz başına bit sayısı
N_s: kesim başına bit sayısı

Bir izi okumak için gerekli süreyi (T_0) hesaplayın.

12-8. Manyetik bir teyp sekiz izlidir ve hızı 120 inç/saniyedir. Yoğunluğu inç başına 1600 bittir. Bunun aktarma hızı nedir?

12-9. Bir kelimenin içerik adreslemeli bellekte çakışma fonksiyonunun tümleyenini bulun. Diğer bir deyişle M_i'nin ÖZEL-VEYA fonksiyonlarının toplamı olduğunu gösterin. M_i'lerin mantık şemasını çizin ve bunu M_i yi elde etmek için tersleyici ile sonlandırın.

12-10. İçerik adreslemeli bellekte bir kelimenin çakışma mantığı için Boole fonksiyonunu yazın. Bunu yaparken kelimenin işlevli mi yoksa duran mı olduğunu belirten başlık bitini göz önünde bulundurun.

12-11. İçerik adreslemeli bellekteki bir kelimeden tüm anahtar bitleri sıfır olduğunda "çakışma yok" sonucu gelmesi için gerekli ilave mantığı söyleyin.

12-12. **a.** İçerik adreslemeli bellekteki bir kelimenin tüm hücreleri için mantık şeması çizin. Şekil 12.8 deki oku ve yaz mantığı ile şekil 12.9 daki çalışma mantığını buna ekleyin.
b. İçerik adreslemeli bellekteki bir diğer sütunda bulunan (sütun j) tüm hücrelerin mantık şemasını çizin. Aynı sütundaki tüm bitler için ortak çıkış hattı oluşturun.
c. (a) ve (b) deki şemalardan, M_i çıkışının aynı kelimenin oku hattıyla bağlantılı olduğunu gösterin. Böylece çakışan kelime maskelenmiş veri ile çalışan tek kelime olarak okunmuş olur.

12-13. Blok şemayla ve sözlerle, içerik adreslemeli bellekte çakışan birden fazla kelimenin içerik adreslemeli bellekten nasıl okunabileceğini belirtin.

12-14. İçerik adreslemeli belleğin bir kelimesinin tümü ve bir hücresi için bir mantık üreterek, maskelenmemiş veri içerik adreslemeli bellekteki kelimeden büyük (fakat eşit değil) olduğu zaman bunu göstermesini sağlayın.

12-15. İki yönlü kümelenmiş içerik adreslemeli ön bellek dört kelimelik bloklar kullanıyor. Ön, bellekten 2048 kelime alabilir. Ana bellek büyüklüğü 128 K × 32 dir.

a. Bir ön bellek kurmak için gerekli tüm bilgileri söyleyin.

b. Ön belleğin büyüklüğü ne olur?

12-16. Bir ön belleğin erişim süresi 100 ns, ana belleğinki 1000 ns dir. Bellekten isteklerin yüzde 80 i okuma, yüzde 20 si yazmadır. Okuma erişiminde bulma oranı 0.9 dur. Doğrudan yazma yöntemi kullanılıyor.

a. Sadece bellek okuma çevrimleri varsa ortalama erişim süresi nedir?

b. Eğer sistemde hem okuma hem yazma varsa ortalama erişim süresi nedir?

c. Yazma çevrimi göz önüne alınırsa bulma oranı nedir?

12-17. Dört yollu kümelenmiş içerik adreslemeli ön bellek her kümede dört kelime içerir. LRU (son kullanım sırası içinde en eski olanı bulan) algoritmasına dayanan bir yerleştirme işlemi kullanılır. LRU algoritması her bir kelimeyle bağlantılı 2 bit sayaçlar yardımıyla kurulmuştur. Her kelime için 0 dan 3 e kadar kaydedilebilir. Bir bulma gerçekleştiğinde bulunan kelimeye bağlı sayaç sıfırlanır. Bu sayaçların, söylenenden küçük olanların değerleri 1 arttırılır. Diğerleri değişmeden kalır. Eğer çakışmama olursa sayacının değeri 3 olan kelime kaldırılır. Yerine yeni kelime yazılır. Sayacı sıfırlanır. Diğer 3 sayaç bir arttırılır. Bu yöntemin çalışmasını aşağıdaki kelime sırası için gösterin. A, B, C, D, B, E, D, A, C, E, C, E. (A, B, C, D ile başlayın. A kullanım sırası içinde en eski olsun).

12-18. Sayısal bir bilgisayarın 64 K × 16 lık belleği 1 K kelimelik ön belleği vardır. Ön bellek dört kelimelik bloklardan oluşur ve doğrudan haritalama kullanır.

a. başlık, indis, blok ve kelime kaç bitten oluşur.

b. Ön belleğin her bir kelime içinde kaç bit vardır? Bunlar görevlerine göre kaçar bit olarak parçalara ayrılır? Bir geçerli biti ekleyin.

c. Ön bellek içinde kaç blok bulunur?

12-19. Bir adres uzayı 24 bit ve adreslenen bellek uzayı 16 bitliktir.

a. Adres uzayında kaç kelime var?

b. Bellek uzayında kaç kelime var?

c. Eğer bir sayfa 2 K kelime ise sistemde kaç sayfa ve kaç blok var?

12-20. Sanal bir belleğin bir sayfası 1 K kelimedir. Sekiz sayfa ve dört blok vardır. İçerik adreslemeli bellek sayfa çizelgesi aşağıdaki değerleri gösteriyor.

Sayfa	Blok
0	3
1	1
4	2
6	0

Eğer MİB tarafından kullanılırsa sayfa hatası verecek.Tüm sanal adreslerin dökümünü yapın (ondalık olarak).

12-21. Sanal bir bellek sisteminde 8 K kelimelik adres uzayı, 4 K kelimelik bellek uzayı, 1 K kelime sayfa ve blok büyüklüğü (şekil 12.18) var. Verilen bir süreçte aşağıdaki sayfa kaynağı değişiklikleri oluyor. (sadece sayfa değişiklikleri yazıldı. Aynı sayfa tekrar gösterilirse tekrar yazılmadı.)

4 2 0 1 2 6 1 4 0 1 0 2 3 5 7

(a) FIFO, (b) LRU yer değiştirme algoritmaları kullanılarak her sayfa kaynağı değişikliğinden sonra ana bellekte duran dört sayfayı bulun.

12-22. Şekil 12.24 (a) , fiziksel bellekteki on altılık 012AF adresine geçmek için iki tane mantıksal adres bulun.

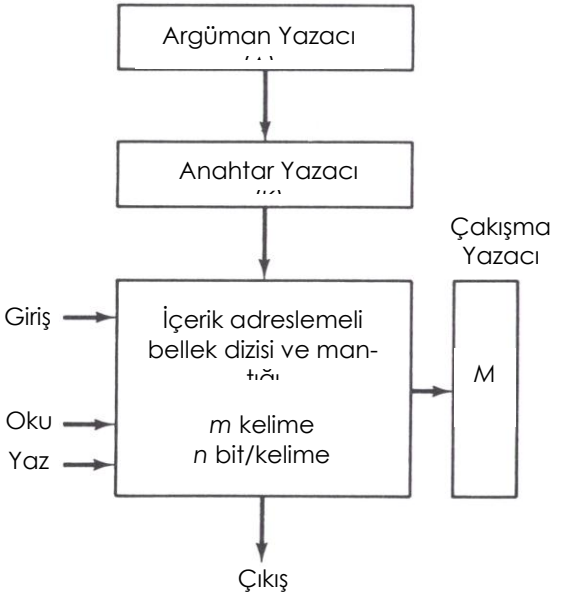
12-23. Bir BS sistemindeki adres uzayı 128 kesim içeriyor. Her kesimde 4 K kelimelik 32 sayfaya kadar olabilir. Fiziksel bellek her biri 4 K kelimelik 4 K bloktan oluşur. Mantıksal ve fiziksel adres biçimlerini oluşturun.

12-24. Soru 12.13 te oluşturulan mantıksal adresin ikili karşılığını 15 inci sayfada 36 ıncı kesim ve 2000 numaralı kademe için verin.

KAYNAKLAR

1. Baer, J. L., *Computer Systems Architecture*. Potomac, MD: Computer Science Press, 1980.
2. Dasgupta, S., *Computer Architecture: A Kipern Synthesis*. Vol. 1. New York: John Wiley , 1989.
3. Gibson,G. A., *Computer Systems Consepts and Desing*. Englewood Cliffs, NJ: Prentice Hall, 1991.
4. Hamacher, V. C., Z. G. Vranesic, and S. G. Zaky, *Computer Organization*, 3rd ed. New York: McGraw-Hill, 1990.
5. Hwang, K., and F. A. Briggs, *Computer Architecture and Parallel Processing*. New York: McGraw-Hill, 1984.
6. Kain, R., *Computer Architecture: Software and Hardware*.Vol. 2. Englewood Cliffs, NJ: Prentice Hall, 1989.
7. Langholz, G., J. Francioni, and A. Kandel, *Elements of Computer Organization*. Englewood Cliffs, NJ: Prentice Hall, 1989.

8. Murray, W. D., *Computer and Digital System Architecture*, Englewood Cliffs, NJ: Prentice Hall, 1990.
9. Patterson, D.A., and J.L. Hennessy, *Computer Arcbulmaecture :A Quantitative Approach*. San Mateo, CA: Morgan Kaufmann Publishers, 1990.
10. Pollard, L.H., *Computer Desing and Architecture*, Englewood Cliffs, NJ: Prentice Hall, 1990.
11. Stone, H. S., *Introduction to Computer Architaecture*, 2nd ed. Chicago: Science Research Associates, 1980.



Şekil 12.6 İçerik adreslemeli bellek şeması.

BÖLÜM 13

Çoklu İşlemciler

BU BÖLÜMDE

13-1 Çoklu İşlemcilerin Karakteristikleri
13-2 İç Bağlantı Yapıları
13-3 İşlemciler Arası Seçim
13-4 İşlemciler Arası İletişim ve Eş zamanlılık
13-5 Ön Bellek Ardışıklığı

13.1 Çoklu İşlemcilerin Karakteristikleri

Çoklu işlemcili bir sistem, iki veya daha fazla merkezi işlem biriminin (MİB) bellek ve giriş-çıkış aletleriyle bağlanmasıyla oluşur. Çoklu işlemci içinde bir "işlemci" terimi ya MİB veya giriş/çıkış işlemcisi (GÇİ) anlamına gelir. Tek bir MİB li bir sistemle bir veya daha fazla GÇİ den oluşan sistem çoklu işlemci tanımına girmez. Eğer GÇİ de MİB le karşılaştırılabilecek seviyede bilgi işlem özellikleri varsa o zaman olabilir. Her zaman tanımlandığı gibi, bir çoklu işlemcili sistem birden fazla sayıda MİB ile, adet olduğu üzere bir veya da fazla GÇİ den oluşur. Bölüm 9.1 de çoklu işlemciler, çok buyruk akımı, çok veri akımı (MIMD) sistemleri olarak sınıflandırılırlar.

MIMD

Çoklu işlemcili sistemler ile çoklu bilgisayar sistemleri arasında bazı benzerlikler vardır. Çünkü her ikisi de eş zamanlı işlemleri destekler. Halbuki, çok bilgisayarlı bir sistemle, çoklu işlemcili bir sistem arasında önemli bir ayrıcalık vardır. Bilgisayarlar birbirlerine, ağ oluşturacak haberleşme hatları ile bağlanabilirler. Böyle bir ağ, belirli sayıda, birbirleriyle haberleşen veya haberleşmeyen bağımsız bilgisayarlardan oluşur. Bir çoklu işlemcili sistem tek bir işletim sistemi tarafından denetlenir. Bu işletim sistemi, işlemciler ve sistemin diğer bileşenleri arasında etkileşimi sağlar. Böylece bir problemin çözümünde hepsi yardım 13.2 etmiş olur.

Mikro İşlemci, VLSI

Her ne kadar, büyük çaplı bilgisayarlar, tüm sistem içinde iki veya daha fazla MİB içerirlerse de, çok işlemcili sistemlerde temel çıkış noktası mikro işlemcidir. Mikro işlemciler gerçekten çok az yer kaplarlar ve çok ucuzdurlar. Böylece bunların çok sayıda bir sistem içinde bir araya gelmeleri her bakımdan uygun olur. Çok büyük eşel de birleşik devre teknolojisi, bilgisayar parçalarını, öyle düşük seviyelere çekti ki, sistemin istenen verim ve özelliklerde olmasını sağlamak çekici bir hal aldı.

Çoklu işlemciler, sistemin güvenilirliğini arttırır. Böylece sistemin bir parçasındaki bir hata veya bozukluk sistemin tamamı üzerinde çok sınırlı bir etki yapar. Eğer bir hata, işlemcinin birini durdurursa, ikinci bir işlemci bu duran işlemcinin işlerini yapmak üzere atanabilir. sistem bütününde doğru çalışmaya devam eder. Belki küçük bir etkinlik kaybı olur.

Çoklu işlemcili kurulumun sağladığı kazanç sistem verimindeki gelişmedir. Sistem, yüksek verimini, işlemcilerin iki yoldan birinde paralel olarak ilerlemesinden elde eder. Bu yollar.

1. Çoklu bağımsız işler paralel olarak çalışabilir.
2. Tek bir iş, çoklu paralel görevlere bölünebilir.

Bütün bir iş, her bir işlemcinin bireysel olarak yerine getirebileceği görevlere bölünebilir. Sistem görevleri, belli bazı işlemleri çok etkin biçimde yerine getirmek üzere özel olarak tasarlanmış işlemcilere dağıtabilir. Örneğin bir bilgisayar sisteminde bir işlemci, endüstriyel işlem denetim hesaplarını yaparken, diğerleri sıcaklık ve akış oranı gibi muhtelif parametreleri gözler ve denetler. Diğer bir örnek, bir bilgisayarda bir işlemci, yüksek hızda kayan noktalı matematik işlemleri yaparken, diğeri her zamanki bilgi işleme görevleri yapar.

Çoklu işlem, bir programı paralel icra edilebilir görevlere ayırarak verimi arttırabilir. Bu iki yolla mümkün olur. Ya kullanıcı açık olarak bazı görevlerin paralel olarak yapılmasını ister. Bu, program yüklenmeden önce paralel icra edilebilir parçalara bölerek yapılabilir. Birçok çoklu işlemcili bilgisayar üreticileri, makine ile beraber bir işletim sistemi ve bir dil verirler. Bunlar paralel işleme uygun işlemler yapar. Diğer daha etkin yol ise, programcının programı içinde, derleyicinin otomatik olarak paralellikleri bulması ve ayırt etmesidir. Derleyici, program içinde verilere bağımlılığı araştırır. Eğer program, başka bir bölümde elde edilen verilere bağımlı ise, bu verinin üretildiği parça daha önce icra edilmelidir. Nitekim, birbirlerinin ürettiği verilere gereği olmayan parçalar aynı süreçte icra edilebilirler. Paralelleştiren derleyici, tüm programı gözden geçirir ve herhangi bir veri bağımlılığı olup olmadığına bakar. Birbirlerine veri bağımlılığı olmayan parçalar farklı işlemcilerde eş süreçte icra edilirler.

Sıkı Paket

Çoklu işlemciler, belleklerinin kurulumuna göre sınıflandırılır. Çoklu işlemcili bir sistemde bellek ortak ise, bu *paylaşımlı bellek* veya *sıkı paket çoklu işlemci* olarak adlandırılır. Bu durum her bir işlemcinin kendi özel yerel belleği olmamasını gerektirmez. Gerçekte, bir çok ticari sıkı paket çoklu işlemciler her bir MİB le birlikte kendi ön belleğine sahiptir. Buna ek olarak genel bir ortak bellek de vardır. Ve bütün MİB ler buna erişebilir. Bilgi, MİB ler arasında ortak genel belleğe konarak paylaşılabilir.

Gevşek Paket

Buna karşıt bir mikro işlemci kurulumu *dağıtılmış bellek* veya *gevşek paket* sistemidir. Böyle bir sistemde her bir işlemcinin kendi özel belleği vardır. İşlemciler bir anahtar devresiyle bağlıdır. Bilgi bir işlemciden diğerine bir haber yollama devresiyle yollanır. İşlemciler yedek programları ve verileri diğer işlemcilere paketler halinde yollar. Bir paket, bir adres, veri içeriği ve bazı hata bulma kodlarından oluşur. Paketler belli bir işlemcinin adresine yollanır veya ilk boş işlemci tarafından alınır. Bu kullanılan iletişim sistemine bağlıdır. Gevşek paket sistemlerinde, görevler arasında etkileşim en az olduğu zaman çok etkilidirler. Mamafih bu tip sistemler oldukça yüksek derecede etkileşimi kaldırabilirler.

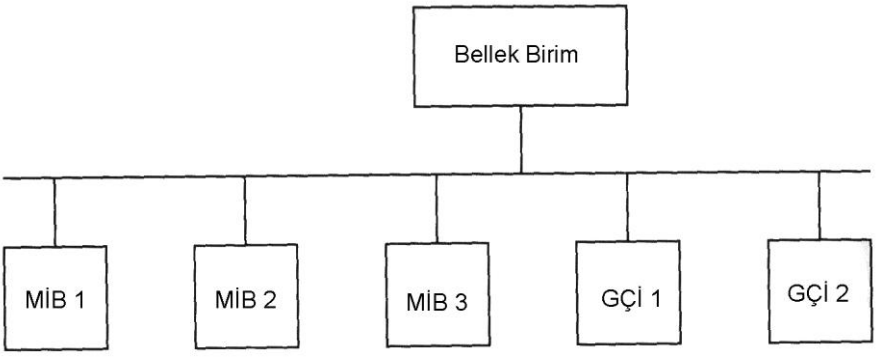
13.2 İç Bağlantı Yapıları

Çoklu işlemcili bir sistemin bileşenleri MİB ler ve GÇİ ler, giriş-çıkış aletleriyle veya bir bellek birimi ile bağlıdır. Bellek ayrı parçalara bölünmüş olabilir. Bu bileşenler arasında iç bağlantılar farklı fiziksel biçimlerde olabilir. Bu farklı biçimler işlemciler ve bellek arasındaki aktarım yollarının sayısına veya işlenecek elemanların sayısına bağlıdır. İç bağlantı ağını kurmak için birçok farklı fiziksel biçim mevcuttur. Bunlardan bazıları bu kısımda anlatılacaktır.

1. Zaman paylaşımlı ortak veri yolu
2. Çok girişli bellek
3. Çapraz bağlantı anahtarı
4. Çok adımlı anahtarlı ağ
5. Hiper küp sistemi

Zaman Paylaşımlı Ortak Veri Yolu

Ortak veri yolu çoklu işlemcili sistemde işlemciler ortak bir yolla belleğe bağlanırlar. Zaman paylaşımlı ortak yola sahip 5 işlemcili bir durum Şekil 13.1 de gösterilmiştir. Bir anda yalnız bir işlemci bellek veya diğer bir işlemci ile haberleşebilir. Aktarma işlemleri, o anda veri yolunu denetle-



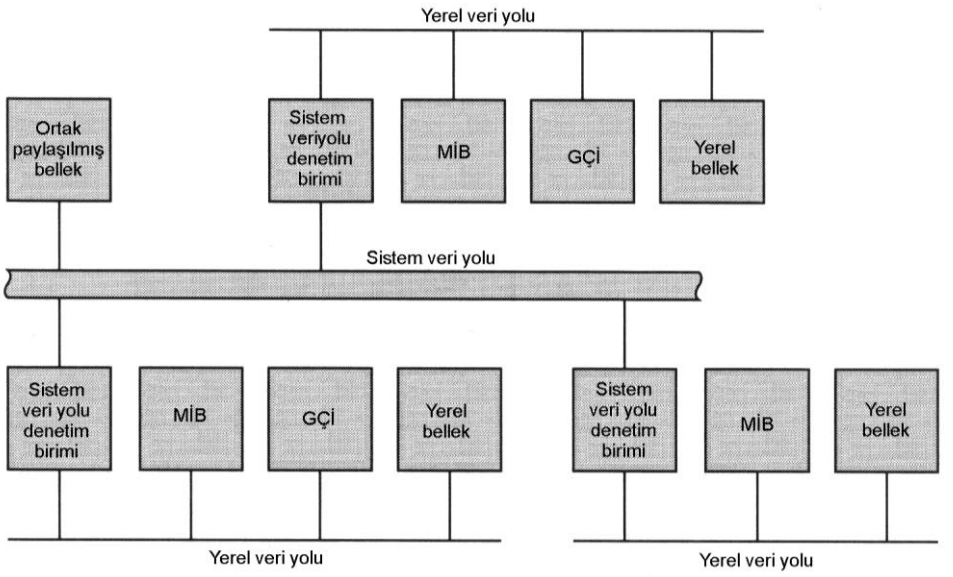
Şekil 13.1 Zaman paylaşımlı ortak veri yolu tasarımı

yen işlemci tarafından gerçekleştirilir. Aktarma yapmak isteyen bir başka işlemci önce veri yolunun müsait olup olmadığına bakmalıdır. Ve ancak yol müsait olduktan sonra işlemci hedefin adresini yazar ve aktarımı başlatır. Hedefteki birime bir buyruk yollanarak, işlemin yapılacağı haber verilir. Hedefteki birim, göndericinin adresini gelen buyruktan anlar ve aktarım başladıktan sonra gelen sinyalleri yanıtlar. Bu düzende, tüm işlemciler aynı veri yolunu kullandıkları için bazı aktarım zorlukları ortaya çıkabilir. Bu zorluklar, veri yolu denetleyicinin istekte bulunan birimler arasında öncelikler kurmasıyla çözülmelidir.

Ortak veri yolu olan bir sistem bir anda sadece bir tek aktarım yapabilir. Bunun anlamı şudur; diğer bir işlemci veya bellekle iletişim kurmak isteyen işlemciler ya iç işlemlerle meşguldürler ya da veri yolu için aylak beklemektedirler. Bunun sonucu olarak, sistemde veri aktarım oranı tek bir veri yolunun hızıyla sınırlıdır. Bunu önlemek için iki veya daha fazla bağımsız veri yolu kullanılabilir. Böylece aynı süreçte daha fazla sayıda aktarım yapılabilir. Fakat böyle bir şey maliyeti çok yükseltir ve karmaşıklığı çok arttırır.

Paylaşılmış Bellek

Daha ekonomik bir *çift veri yolu* yapısı Şekil 13.2. de gösterilmiştir. Burada belli sayıda yerel veri yolu var. Bunların her biri kendi yerel belleğine bir veya daha fazla sayıda işlemci ile bağlıdır. Her yerel veri yolu bir MİB, bir GÇİ veya işlemcilerin herhangi bir kurulumuna bağlıdır. Sistem veri yolu denetleyicisi her bir yerel veri yolunu sistemin ortak veri yoluna bağlar. Yerel GÇİ ye bağlı G/Ç aletleri veya yerel bellek, yerel işlemciye bağlıdır. Ortak veri yoluna bağlı olan bellek, tüm işlemciler tarafından paylaşılır. Eğer bir GÇİ, sistem veri yoluna doğrudan bağlı ise, ona bağlı G/Ç aletleri tüm işlemcilere açık olabilir. Sadece tek bir işlemci paylaşılan bellekle veya diğer ortak kaynaklarla sistem veri yolundan

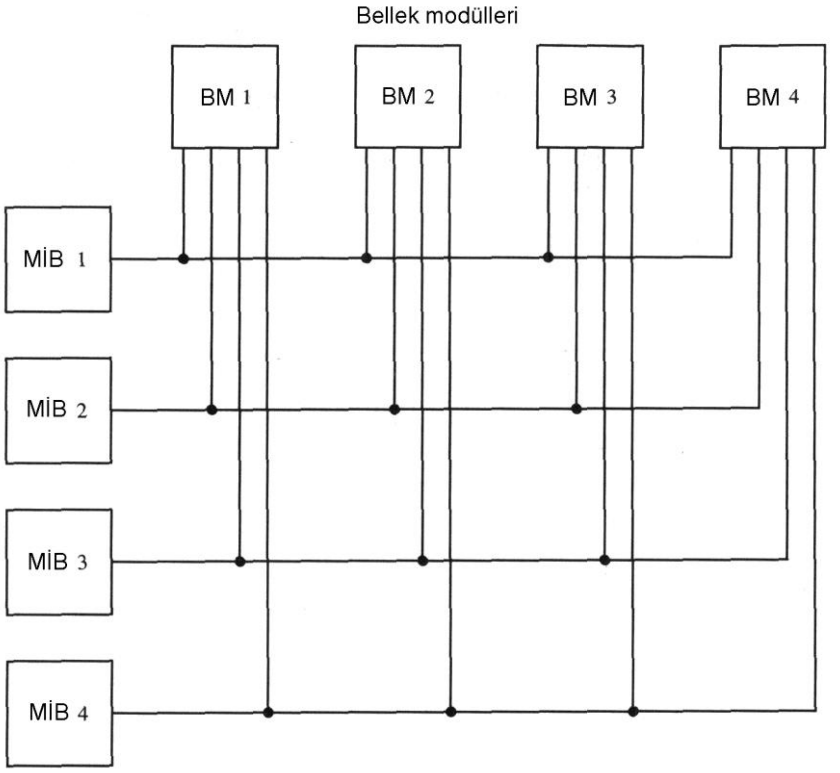


Şekil 13.2 Çoklu işlemciler için sistem veri yolu yapısı

iletişim kurabilir. Diğer işlemciler yerel bellek veya G/Ç aletleriyle haberleşmekle meşguldürler. Yerel belleğin bir kısmı öyle tasarlanabilir ki MİB e bir kaş ön bellek bağlanabilir. Böylece yerel belleğe ortalama 1 erişim süreci, MİB nin çevrim sürecine yakın değerde olabilir.

Çok Kapılı Bellek

Çok kapılı bellek sisteminde, her bir MİB ile her bir bellek birimi arasında ayrık veri yolları bulunur. Bu şekil 13.3 te gösterilmiştir. Şekilde 4 tane MİB ile 4 bellek birimi var. Her bir işlemci veri yolu bir bellek birimine bağlanmıştır. Bir işlemci veri yolu, bellekle iletişimde olmak için gerekli adres, veri ve denetim elemanlarını bulundurur. Bellek biriminde 4 kapı ve her kapıda bir veri yolu mevcuttur. Bu birimin mutlaka bir denetim mantık yapısı olmalıdır. Bu mantık hangi kapının hangi zamanda belleğe erişeceğini hesaplamalıdır. Belleğe erişim kargaşaları her bir kapıya değişmeyen öncelikler verilerek çözülür. Her bir işlemci ile belleğe erişim önceliği, her bir veri yolunun bağlı olduğu kapının fiziksel yerleşim durumuna bağlıdır. Yeni MİB 1, MİB 2 den daha öncelikli, MİB 2, MİB 3 ten daha önceliklidir. MİB 4 en düşük önceliğe sahiptir.

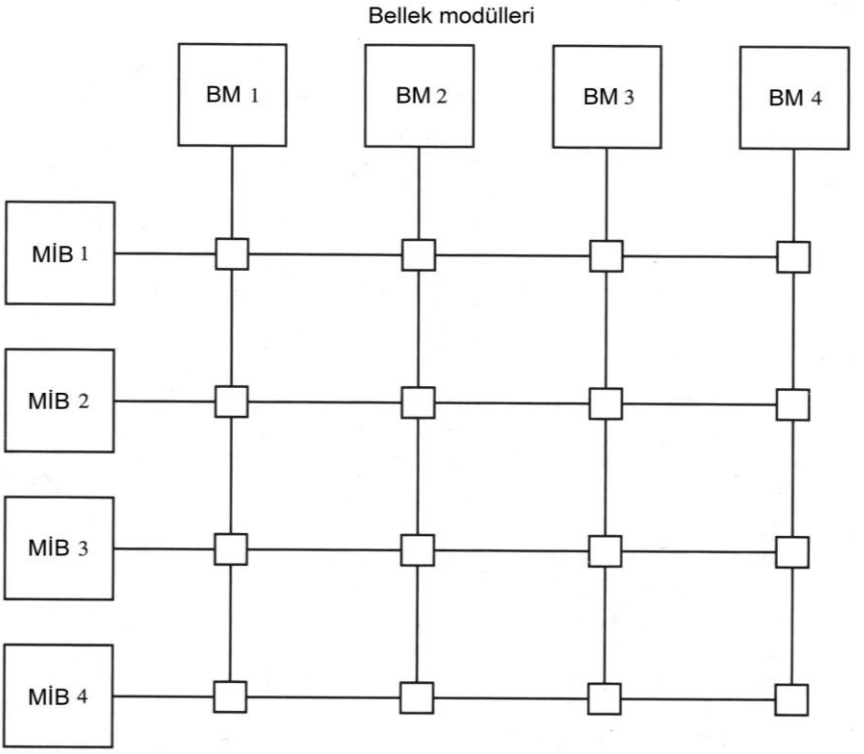


Şekil 13.3 Çok kapılı bellek tasarımı

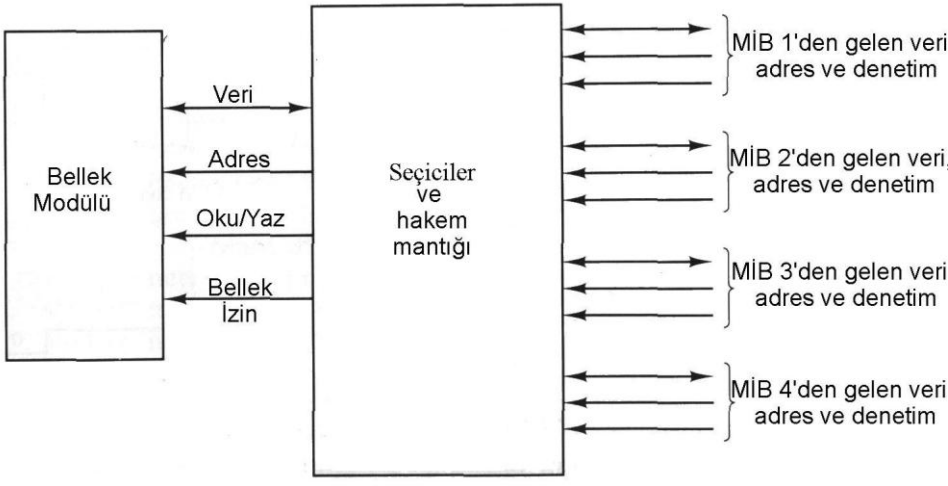
Çok kapılı bellek kurulumunun faydası, çok hızlı aktarım sağlamasıdır. Zorluğu ise pahalı bellek ve denetim mantığıdır. Çok sayıda kablo ve bağlantı parçası gereklidir. Sonuç olarak bu iç bağlantı yapısı, az sayıda işlemci bulunduran sistemlere uygundur.

Çapraz Bağlantı Anahtarı

Çapraz bağlantı anahtarı kurulumunda, belleklerle, işlemciler arasındaki bağlantıların kesim noktaları mevcuttur. Şekil 13.4 te bir çapraz bağlantı, 4 MİB ve 4 bellek birimi için çizilmiştir. Kesim noktalarındaki küçük kareler birer anahtardır. İşlemciden bellek birimine giden bir yol belirler. Her anahtar noktası bir denetim mantığına sahiptir. Bu mantık yardımıyla işlemci ile bellek arasında bir yol kurulabilir. Mantık, önce veri yolundaki adresi inceleyerek, kendi bellek biriminin adreslenip adreslenmediğine bakar. Benzer şekilde aynı bellek adresine erişim için yapılan adresleri de, daha önceden belirlenmiş bir öncelik sırasına göre inceler.



Şekil 13.4 Çapraz bağlantı anahtarı



Şekil 13.5 Çapraz bağlantı anahtarı mantık şeması

Şekil 13.5 te çapraz bağlantının bir bellek birimine bağlanışı görülüyor. Devre, veriyi seçmek için MUX lar, adres ve MİB den bellekle iletişim için denetim buyruğu içerir. Öncelik seviyeleri, iki veya daha fazla MİB, bellek için baş vurduğunda, hakem mantığı ile belirlenir. MUX lar hakem mantığı ile seçim sırasında üretilen ikili kod ile denetlenir.

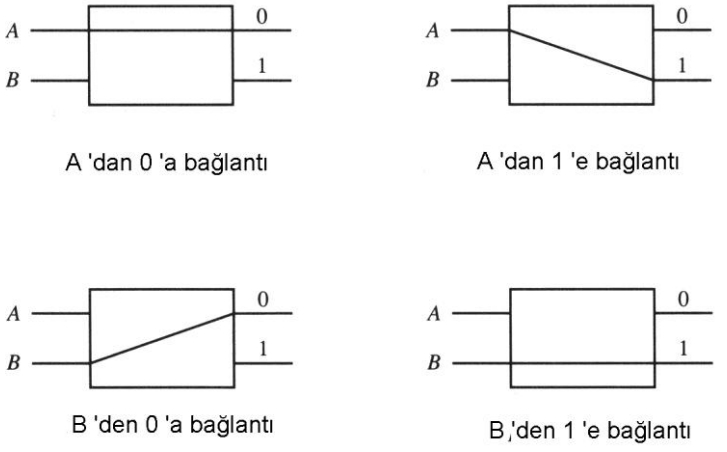
Çapraz bağlantı, tüm bellek birimlerinden es süreli aktarımlar için uygundur. Çünkü her birim için ayrık bir yol mevcuttur. Fakat anahtar için gerekli donanım çok büyük ve karmaşık olabilir.

Çok Seviyeli Anahtarlamalı Ağ

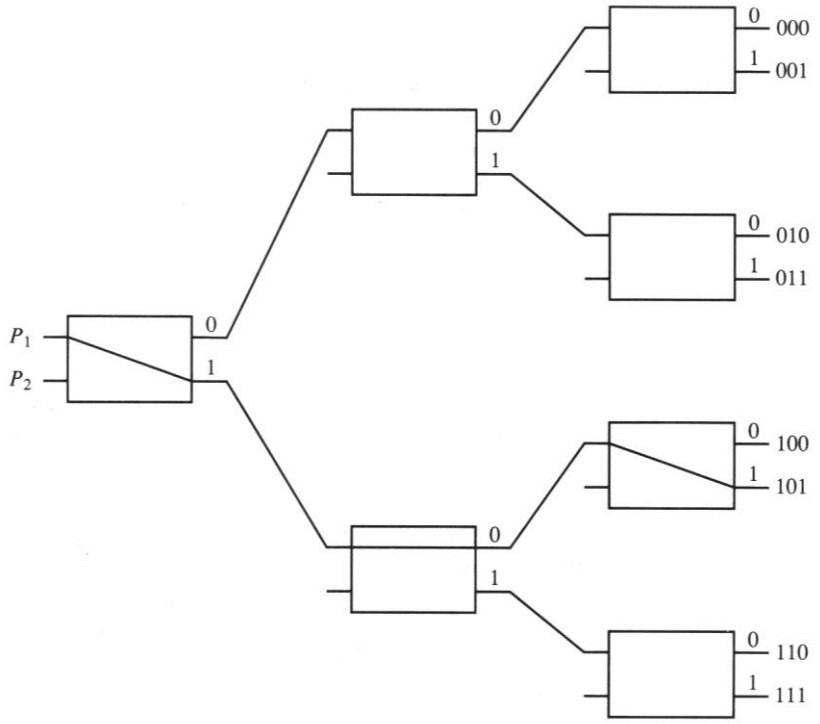
İç Değişim Anahtarı

Çok seviyeli anahtarlamalı ağın temel bileşeni, iki girişli iki çıkışlı iç çevirmeli anahtardır. Şekil 13.6 da gösterildiği gibi, 2 × 2 anahtarın 2 girişi olup *A* ve *B* ile gösterilmiştir. 2 de çıkışı vardır bunlar 0 ve 1 olarak belirtilmiştir. Denetim sinyalleri vardır. Bunlar burada gösterilmemiştir, fakat giriş ve çıkış uçları arasında bağlantı kurarlar. Anahtar *A* girişini çıkışlardan birine bağlar. *B* ucu da aynı biçimde çalışır. Anahtarın, çelişkili istekler durumunda hakemlik yaparak karar verme yeteneği vardır. Eğer *A* ve *B* girişlerinin aynı çıkış ucunu isterse, sadece biri bağlanacak, diğeri ise durdurulacaktır.

2 × 2 anahtarını yapı elemanı olarak kullanarak, çok seviyeli bir ağ oluşturmak mümkündür. Bu ağ, bir dizi kaynakla, hedefleri arasındaki bağlantıyı denetleyebilir. Bunun nasıl yapıldığını görmek için şekil 13.7 deki ikili ağaca bakalım. P_1 ve P_2 işlemcileri, bu ağ üzerinden, 000 dan, 111 e karar numaralandırılmış 8 tane bellek birimi ile bağlantılı olsun. Bir



Şekil 13.6 2 × 2 değişim anahtarının işlemleri

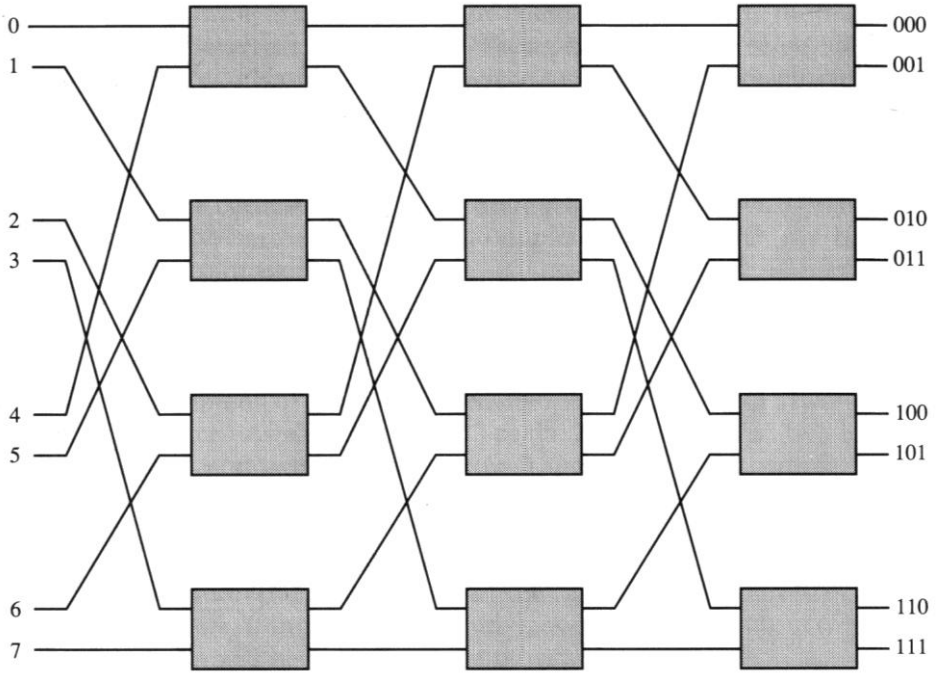


Şekil 13.7 2 × 2 anahtarlar ile ikili ağaç

kaynaktan bir hedefe bir yol, hedefin ikili bitleri ile kurulur. Hedef sayısının ilk biti ilk seviyede, anahtar çıkışını belirtir. İkinci bit, ikinci seviyedeki anahtar çıkışını belirtir, üçüncü bit ise üçüncü seviyedeki anahtarın çıkışıdır. Örneğin, P_1 i 101 belleğine bağlamak için, P_1 den bir yol yapmak gerekir. Yolda ilk seviye anahtarının çıkışı 1 olmalıdır. İkinci seviyedeki anahtar çıkışı 0, üçüncü seviye anahtar çıkışı 1 olmalıdır. Besbelli ki P_1 veya P_2, belleklerden herhangi birine bağlanabilir. Bazı istekler aynı anda karşılanamaz. Örneğin eğer P_1 000 dan 011 e kadar hedeflerden birine bağlanmışsa, P_2 100 dan 111 e kadar hedeflere bağlanabilir.

Omega Anahtarlamalı Ağ

Sıkı paket işlemci-bellek bağlantılarında veya gevşek paket işlemci öğeler arasında iletişimi denetlemek için çok seviyeli anahtarlamalı ağlarda kullanılmak üzere çok farklı topolojiler önerilmiştir. Böyle bir topoloji omega anahtarlamalı ağ olup, şekil 13.8 de gösterilmiştir. Bu kurulumda her bir kaynaktan, seçilen hedefe tek bir yol vardır. Bazı isteklerdeki yollar aynı anda yerine getirilemez. Örneğin, herhangi iki kaynak, aynı anda 000 ve 001 hedeflerine bağlanamaz.



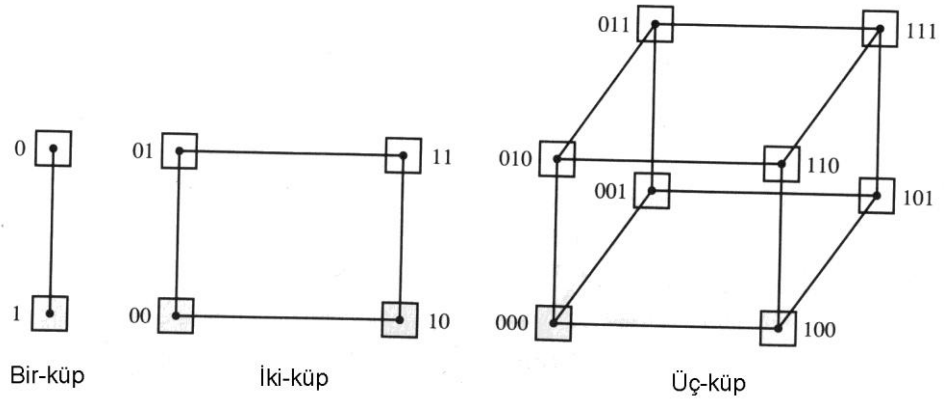
Şekil 13.8 8 × 8 omega anahtarlamalı ağ

Anahtarlamalı ağ üzerinde yapılan bir istek, 3 bitlik bir hedef sayısı oluşturur. Bu hedef sayısı ağ üzerinde ilerledikçe her seviye farklı bir bite bakarak 2 × 2 anahtarındaki uç kararını verir.Birinci seviye en önemli bite bakar İkinci seviye ortadaki bite bakar. Üçüncü seviye ise en düşük seviyeli bite bakar. İstek 2 × 2 anahtarının girişine erişince eğer bit 0 ise yukarı çıkışa, 1 ise aşağı çıkışa yönlendirilir.

Sıkı paket bağlı çok işlemcili bir sistemde kaynak bir işlemci, hedef bellek birimidir. Ağ üzerindeki ilk geçiş, yolu belirler. Takip eden geçişler adresi belleğe aktarmak ve isteğe bağlı olarak verileri yazmak yada verileri okumak için kullanılır. Gevşek bağlı bir çok işlemcili sistemde hem kaynak hem hedef işlemcilerdir. Yol kurulduktan sonra kaynak işlemci, verileri hedef işlemciye aktarır.

Hiper Küp İç Bağlantısı

Hiper küp veya ikili *n*-küp çoklu işlemci yapısı, gevşek bağlı bir sistem olup, $N = 2^n$ işlemciden oluşur. Bu işlemciler *n*-boyutlu bir ikili küpte birbirlerine bağlıdırlar. Her bir işlemci küpte bir köşe oluşturur. Her ne kadar her bir köşede bir işlemci olacağı varsayılırsa da sadece bir MİB değil, bir bellek ve G/Ç ara yüzeyi de bulunur. Her bir işlemci diğer *n*



Şekil 13.9 n = 1, 2, 3 için hiper küp yapısı

tane komşusuna doğrudan iletişim bağlantısına sahiptir. Bu bağlantılara küpün kenarları denir. Burada 2^n tane ayrık n-bit ikili adres vardır ve her bir adres bir işlemciye tahsis edilmiştir. Her bir işlemci adresi, n tane komşusundan sadece 1 bit farklıdır.

Şekil 13.9 da bir hiper küp yapısı n = 1, 2 ve 3 de gösterilmiştir. Tek boyutlu bir küpte n = 1 ve 2^n = 2 dir. Yani iki işlemci vardır. Bunlar tek bir yolla birbirine bağlanmıştır. İki boyutlu bir küpte n = 2 ve 2^n = 4 olduğundan 4 köşesi vardır. Köşeler kare şeklinde bağlıdır. 3 boyutlu bir küpte 8 köşe vardır. Küp şeklinde bağlanmışlardır. n boyutlu bir küpte 2^n tane köşe vardır ve her bir köşede bir işlemci bulunur. Her köşeye bir ikili adres verilmiştir. Öyle ki her adres diğer iki komşusundan sadece 1 bit farklıdır. Örneğin adresi 100 olan köşenin üç komşusu 000, 110, 101 dir. Bu sayılar 100 adresinden sadece 1 bit farklıdır.

n-küp yapıda, bir kaynaktan yollanan bir haber hedefe ulaşıncaya kadar bir ile n tane arasında kenardan geçer. Örneğin 3 boyutlu yapıda, 000 köşesi 001 köşesiyle doğrudan haberleşebilir. 011 ile haberleşebilmek için en az iki tane (000 dan 001 e ve sonra 001 den 011 e) kenardan geçmek zorundadır. 000 köşesinden 111 köşesinde gitmek için en az 3 kenardan gitmelidir. Kaynak ile hedef arasında bir bağlantı, bunların adreslerinin ÖZEL-VEYA lanmasıyla bulunabilir. Sonuçta bulunan ikili değer içinde 1 bitleri bulunur. Bu 1 ler iki köşeyi ayıran kenarları belirtir. Haber, iki köşeyi birleştiren herhangi kenarlar üzerinden yollanır. Örneğin 3 küp yapısında 010 dan 001 e giden bir haber, iki adresin ÖZEL-VEYA lanması sonucu bulunan 011 den dolayı ikinci ve üçüncü kenarlardan 001 e ulaşır.

Hiper küp örneği INTEL in iPSC bilgisayarıdır. Bu bilgisayarda 128 (n = 7) mikro bilgisayar, birbirine bağlanmıştır. Her köşede bir MİB, bir

kayan noktalı işlemci, yerel bellek bir tane seri haberleşme ara yüzeyi vardır. Köşelerdeki bilgisayarlar, bağımsız olarak yerel belleklere konmuş verilerle ve içlerindeki programlarla bağımsız olarak çalışırlar. Veriler ve programlar diğer köşelerden ve küp-yöneticisi tarafından yollanır. Uygulama programları küp yöneticisi üzerinde geliştirilir ve derlenir. Sonra köşelere yollanır. İşler, sistem üzerinde dağıtılmış ve eş zamanlı olarak icra edilmiştir.

13.3 İşlemciler Arası Yargı

Sistem Veri Yolu

Bilgisayar sistemlerinin bileşenler arasında iletişimi sağlamak için muhtelif seviyelerde veri yolları vardır. MİB içinde iç veri yolları vardır. Bu iç veri yolları işlem yazaçlarıyla ALU arasında veri aktarımı yapar. Bellek veri yolu, verilerin, adreslerin ve oku/yaz bilgilerinin aktarımı için gereklidir. Bir G/Ç veri yolu, giriş-çıkış aletlerinden veya aletlerine aktarma yapar. Çok işlemcili bir sistemde, temel bileşenleri olan, MİB ler, GÇİ ler ve belleği bağlayan veri yoluna sistem veri yolu denir. Bir sistem veri yolunun devreleri, bir birinin aynı baskı devreler şeklinde yapılmıştır. Her bir baskı devre tek bir birime aittir. Bu baskı devre uçları ana kart üzerindeki yuvalara paralel biçimde bağlanırlar. Böylece sistem veri yolu, bir miktar bir birine tamamen eşit baskı devreler içerir. Her birimde, bu baskı devreyi içeren bir kart vardır. Ana kart bağlantı yuvalarına paralel olarak bağlanmış devreler içerir. Bağlantı yuvasının bir ucu bir hatta bağlıdır. Diğer bütün bağlantı yuvalarındaki aynı uca karşılık gelen uçlar aynı hatta bağlıdır. Dolayısıyla yuvaya uyan kart, sistem veri yolunu oluşturur.

Paylaşılan belleği olan çok işlemcili sistemde işlemcilerden birinin ortak belleğe veya diğer ortak kaynaklara geçiş isteği, sistem veri yolu üzerinden karşılanır. Eğer başka bir işlemci veri yolunu kullanmıyorsa, istekte bulunan işlemcinin arzusu derhal karşılanır. Eğer başka bir işlemci halen veri yolunu kullanıyorsa, istekte bulunan işlemci beklemek zorundadır. Daha ötesi, diğer işlemcilerde aynı zamanda veri yolu isteminde bulunabilirler. O zaman mutlaka bir seçim yargısı gerekir. Tüm ortak kaynaklar için bir seçim yargısı gereklidir. Seçim yargısı, sistem veri yolu denetleyicisinin parçasıdır ve yerel veri yolu ile sistem veri yolu arasında yer almalıdır.

Sistem Veri Yolu

Tipik bir sistem veri yolunda, yaklaşık 100 tane kadar sinyal hattı vardır. Bu hatlar 3 eylem grubuna ayrılır. Veri adres ve denetim. Buna ek olarak, bileşenlere elektrik akımı dağıtan güç dağıtım hatları vardır. Örneğin

IEEE 796 standardı çoklu veri yolu sisteminde 16 veri 24 adres hattı, 26 denetim hattı ve 20 tane güç hattı, toplam 86 hat vardır.

Veri hatları ortak bellekle işlemciler arasında veri aktarır. Veri hatlarının sayısı çoğunlukla 8 in katıdır. 16 ve 32 çok rastlananlardır. Adres hatları bir bellek adresi, başka bir kaynak veya hedefi belirlemelidir. Örneğin giriş çıkış kapıları gibi. Adres hatları, sistem içinde olabilecek en büyük kapasiteye uygun olanıdır. Mesela 24 hat 2^{24} (16 Mega) kelime belleği adresleyebilir. Veri ve adres hatları üç durumlu buffer lar ile sona ererler (Şekil 4.5). Adres buffer ları tek yönlüdür. İşlemciden belleğe doğru gidiş vardır. Veri hatları iki yönlüdür. Böylece her iki yöne doğru aktarım yapılabilir.

Eş Zamanlı Veri Yolu

Sistem veri yolunda veri aktarımları eş zamanlı veya eş zamanlı olmayan olabilir. Eş zamanlı veri yolunda her bir veri, kaynak ve hedef tarafından daha önceden tanınan süreç içinde aktarılır. Eş zamanlı aktarımda eş zamanlılık, kaynak ve hedefin aynı saat tarafından zamanlandırılması ile olur. Karşıt bir yöntem farklı saatler kullanmaktadır. Fakat bunlar yaklaşık aynı frekansta olmalıdır. Saatlerin eş zamanlı olması için periyodik olarak ayar sinyali yollanır.

Eş Zamanlı Olmayan Veri Yolu

Eş zamanlı olmayan veri yolunda her bir veri, beraberinde onay denetim sinyalleriyle beraber aktarılır (Şekil 11.a). Bunun amacı, verinin ne zaman kaynaktan çıktığının ve hedefe vardığının anlaşılmasıdır.

Denetim hatları, birimler arasındaki aktarımı denetleyen sinyaller üretir. Zaman sinyalleri verilerin ve adres bilgisinin geçerliliğini denetler. Buyruk sinyalleri, icra edilecek işlemi belirler. Tipik denetim hatları, bellek okuma veya yazma, aktarımı, onaylama, kesme istemleri, veri yolu denetim sinyalleri ve veri yolu tahsisi ve seçim yargısı gibi bilgiler taşır.

Çizelge 13.1 de IEEE, 76 standardı 796 çoklu veri yolunda mevcut 86 hattı gösteriyor. Çoklu veri yolunda 16 veri hattı, 24 tane adres hattı var. Çoklu veri yolunda bütün sinyaller aktif durumdadır veya düşük seviyede izinlendirilmişlerdir. Veri aktarımı denetim sinyalleri ve bellek okuma yazma ve G/Ç oku, yaz denetimlerini içerir. Bunun sonucu olarak adres hatları, bellek ve G/Ç uzaylarını ayırmakta kullanılabilir. Bellek veya G/Ç, aktarım sona erince bir onay ile yanıt verirler. Her bir işlemci bir tane kesme onay girişi ile 8 e kadar sayıda kesme istemi çıkışları vardır. bunlar çoğunlukla bir kesme istemi öncelik denetleyicisine bağlıdır (Şekil 11.21). Farklı denetim sinyalleri zamanlama ve ilkleştirme olanakları sunarlar. Özellikle veri yolu kilit sinyalleri, çok işlemcili sistemlerde temel olaydır. İşlemci tarafından harekete geçirilen sinyaller, diğer işlemcilerin veri yolunu kullanmalarını engeller. Kilit sinyalleri, işlemci test et ve 1 yapma buyruğunu icra ederken çalışır. bu buyruk işlemcilerin zaman uyumlu olmaları için gereklidir (Bak. bölüm 13.4)

Çizelge 13.1 IEEE standart 796 çoklu veri yolu sinyalleri

	Sinyal ismi
Veri ve adres	
Veri hatları (16 hat)	DATA0-DATA15
Adres hatları (24 hat)	ADRS0-ADRS23
Veri Aktarımı	
Bellek okuma	MRDC
Bellek yazma	MWTC
G/Ç okuma	IORC
G/Ç yazma	IOWC
Aktarım Onayı	TACK
Kesme denetimi	
Kesme istemi (8 hat)	INT0-INT7
Kesme onayı	INTA
Çeşitli denetimler	
Ana saat	CCLK
Sistem ilkleştirme	INIT
Byte yüksek izinli	BHEN
Bellek tutma (2 hat)	INH1-INH2
veri yolu kilidi	LOCK
Veri yolu seçim kararı	
Veri yolu istemi	BREQ
Ortak veri yolu istemi	CBRQ
Veri yolu meşgul	BUSY
Veri yolu saati	BCLK
Veri yolu önceliği çalışıyor	BPRN
Veri yolu önceliği kapalı	BPRO
Güç ve Toprak (20 hat)	

IEEE nin izniyle basılmıştır.

Altı veri yolu seçim yargısı için 6 tane seçim veri yolu yargı sinyali kullanılır. Bu sinyaller, seri ve paralel yargı işlemleri tartışmasından sonra açıklanacaktır.

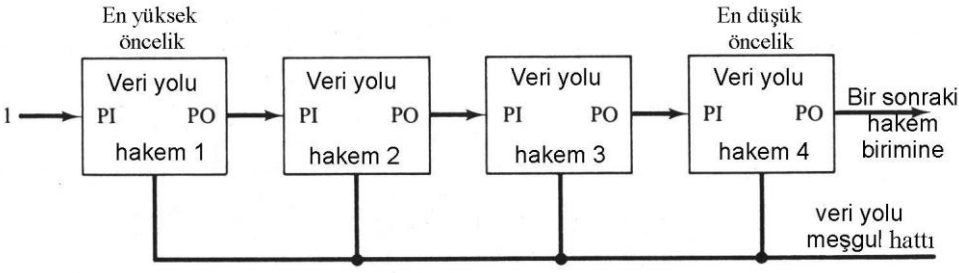
Seri Veri Yolu Seçim Kararı

Seçim kararı, tüm işlemci isteklerinde, daha önce kararlaştırılmış önceliklere göre yapılır. Birimler seri veya paralel bağlanmış olabilir. Bunlardan birinin sistem veri yolu isteği ve ilgili öncelikler donanımsal olarak kurulabilir. Seri bağlantıda öncelik çözme yöntemi papatya zinciri bağlantısından elde edilir. Veri yolu seçim kararı veren devreler papatya zinciri biçiminde bağlanır. Tıpkı 11.5 te anlatıldığı gibi kesme istemi önceliği mantığına uygun yapılır. Sistem veri yoluna bağlanan işlemcilere bağlanış sırasına göre öncelik verilir. Öncelik hattına en yakın alet, en yüksek

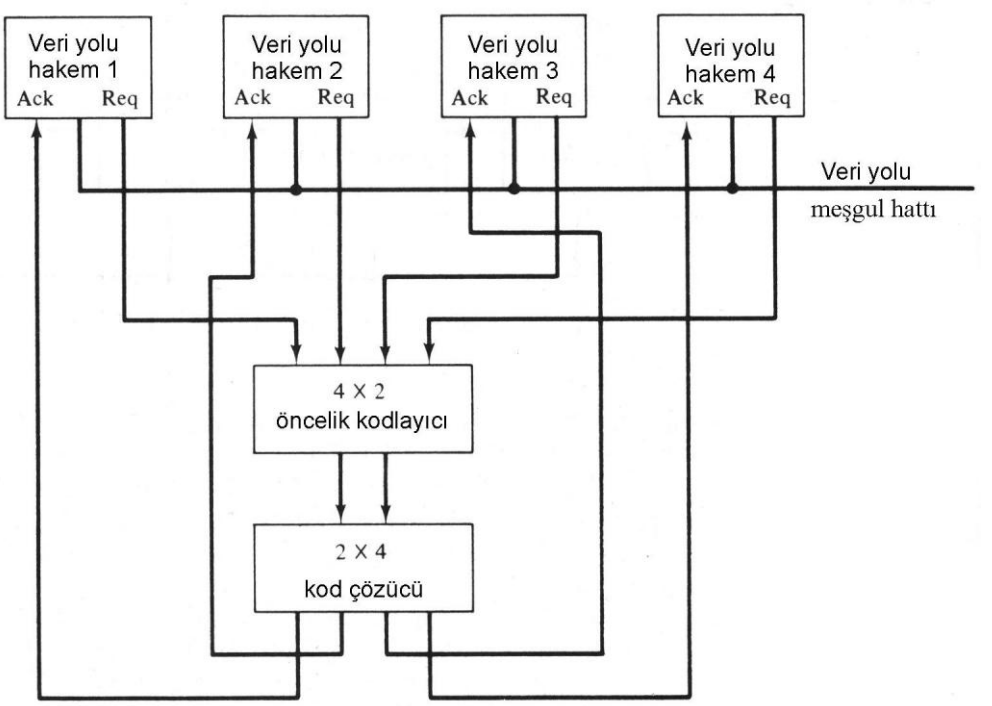
önceliğe sahiptir. Bir çok alet aynı anda istemde bulunursa, önceliği fazla olan alır.

Şekil 13.10 papatya zinciri bağlantısını ve 4 tane karar vericiyi gösteriyor. Her bir işlemcinin kendi özel seçim kararı verme mantığı olduğu ve öncelik giriş çıkış hatları bulunduğunu varsayıyoruz. Her bir karar vericinin öncelik çıkışı (*PO*), sonraki, daha az öncelikli karar vericinin öncelik girişine (*PI*) bağlanmıştır. Önceliği en yüksek kara vericinin, *PI* girişine mantıksal 1 verilir. Bu en yüksek önceliği olan sistem, sistem veri yoluna her istediğinde erişiri. Herhangi bir karar vericinin, eğer *PI* girişi 1 ise, *PO* çıkışı da 1 dir. Fakat eğer bu karar vericinin işlemcisi veri yolu denetimini istemiyorsa, denetim bir sonraki birime geçer. Bu önceliğin aktarılmasıdır. Eğer işlemci, veri yolunun denetimini istiyorsa ve *PI* girişi 1 ise, bu durumda işlemci *PO* = 0 yapar. Böylece önceliği daha az olanların *PI* girişleri sıfır, doğal olarak *PO* çıkışları da sıfır olur. Sonuçta *PI* = 1 ve *PO* = 0 olan işlemci veri yolunu kullanır.

Daha yüksek önceliği olan bir işlemci veri yolu denetimini istediği anda, veri yolunu kullanan daha az öncelikli işlemci, yaptığı işin ortasında olabilir. Veri yolunu serbest bırakması için işini tamamlaması gerekir. Veri yolu meşgul hattı şekil 13.10 da gösterilmiştir. Bu hat sayesinde denetimin sırayla aktarılması sağlanır. Meşgul hattı, her bir birimdeki toplayıcı devrelerinden gelir ve donanımsal VEYA mantığına bağlıdır. Bir karar verici *PI* = 1 ve *PO* = 0 dan veri yolu denetimini alınca, meşgul hattına bakar. Eğer hat boşsa, bunun anlamı başka bir işlemci veri yolunu kullanmıyor demektir. Karar verici, meşgul hattını doldurur ve onun işlemcisi denetimi alır. Eğer karar verici, meşgul hattını dolu bulursa, bunun anlamı başka bir işlemci halen veri yolunu kullanıyordur. Karar verici meşgul hattına bakmaya devam eder ve düşük öncelikli işlemci bitirip, denetimi bırakırsa, meşgul hattı boşalır. Yüksek öncelikli karar verici meşgul hattını doldurur ve işlemcisi veri yoluna bağlanır.



Şekil 13.10 Seri papatya zinciri hakem birimi



Şekil 13.11 Paralel hakem birimi

Paralel Veri Yolu Seçim Kararı

Paralel veri yolu, seçim kararı için bir tane öncelik kodlayıcı ve bir tane kod çözücü kullanır (şekil 13.11). Her bir veri yolu karar vericisinin, paralel bağlantıda bir tane veri yolu istem çıkış hattı ve bir tane veri yolu onay giriş hattı vardır. Her bir karar verici, eğer kendi işlemcisi sistem veri yoluna gerek duyuyorsa, istek hattını doldurur. İşlemci veri yolu denetimini de alır, sonra onay hattını doldurur. Meşgul hattı, denetimin düzenli biçimde aktarımını sağlar. Şekil 13.11 de 4 tane karar vericiden istek hatları görülüyor. Bu hatlar 4 × 2 öncelik kodlayıcısına bağlıdır. Kodlayıcının çıkışı 2 bit kod üretir. Bu kod veri yoluna istemde bulunanlar arasında, en yüksek öncelikli olanı belirtir. Kod değiştiricinin doğruluk çizelgesi çizelge 11.2 de (bölüm 11.5) ten görülebilir. Kodlayıcıdan çıkan 2 bit, bir 2 × 4 kod çözücüye yollanır. Bu da uygun onay hattını doldurur ve bu isteyenler içinde en yüksek öncelikli işlemci veri yoluna ulaşır.

Şimdi veri yolu seçimi karar sinyallerini (çizelge 13.1) açıklayalım. Veri yolu önceliği giriş (BPRN) ve veri yolu önceliği çıkış (BPRO) hatları veri yolu öncelik devreleri için papatya zinciri bağlantısı olarak kullanı-

lır. Meşgul hattı sinyali BUSY bir açık toplayıcı devre çıkışıdır ve tüm karar vericilere veri yolu meşgulken, meşgul olduğunu bildirir. Ortak veri yolu istemi CBRQ da açık toplayıcı devre çıkışıdır ve karar vericiye daha düşük seviyede sistem veri yolu istemi olup olmadığını bildirir. Paralel karar verme işlemleri oluşturmakta kullanılan sinyaller, veri yolu istemi BREQ ve öncelik istem girişi BPRN sinyalleridir. Bunlar istem ve onay sinyallerine karşılık gelir. Veri yolu saati, tüm veri yolu aktarımların zaman uyumlu yapmak için kullanılır.

Dinamik Seçim Kararı Algoritması

Biraz önce tanımladığımız veri yolu seçim kararı işlemleri durağan seçim öncelik algoritmasına dayanır. Çünkü her aletin önceliği önceden belirlenmiştir ve değişmez. Buna karşılık dinamik bir öncelik algoritması, sistem çalışırken bile öncelikleri değiştirmek olanağı sağlar. Şimdi dinamik öncelik algoritmaları kullanan bazı seçim kararı işlemlerini tartışalım.

Zaman Dilimi

Zaman dilimi algoritması, sabit uzunlukta zaman dilimleri ayırır, sonra sırayla işlemcilere tahsis eder. Bu durumda her bir işlemciye verilen hizmet işlemcinin veri yolundaki yerinden bağımsızdır. Hiçbir alete özel bir öncelik sağlanmaz. Çünkü hepsi eşit miktarda veri yolundan yararlanır.

Yoklama

Yoklama kullanan bir veri yolu sisteminde veri yolu tahsis sinyali yerine yoklama hatları adı verilen bir dizi hat konmuştur. Bu hatlar bütün birimlere bağlanmıştır. Hatlar, veri yolu denetimi tarafından veri yoluna bağlı her bir alet için adres tanımlamakta kullanılır. Denetim adresler içinde daha önce belirlenen biçimde dolaşır. Eğer bir işlemci veri yoluna erişmek isterse önce adresini bulur. Sonra meşgul hattını doldurur ve veri yoluna erişir. Bir dizi veri yolu çevriminden sonra yoklama işlemi, değişik bir işlemci seçerek devam eder. Yoklama programlanabilir bir işlemdir. Bunun sonucu olarak seçim önceliği program denetiminde değiştirilebilir.

LRU

LRU algoritması, en uzun zamandır veri yoluna erişim yapmamış alete en yüksek önceliği verir. Öncelikler bir zaman kullanımından sonra ortaya çıkar. En başta hiç bir işlemcinin diğerine üstünlüğü yoktur. Zaman içinde veri yoluna erişmeyenlerin puanı artarken, veri yoluna erişenin puanı sıfırlanır. Her saniye başında tüm işlemcilere puan verilir. Puanı yüksek olan veri yoluna ulaşır.

FIFO

İlk gelen, ilk erişimi yapar mantığı, alınan isteklerin sırasına göre erişimin sağlanmasıdır. Bunun için denetim bir kuyruk oluşturur. İstekte bulunan kuyruğa girer. Sıra gelince veri yoluna ulaşır. Her işlemci sırasını bekler.

Dönen Papatya Zinciri

Bu algoritma, papatya zinciri algoritmasının dinamik bir halidir. Bu mantıkta merkezi veri yolu denetimi yoktur. En düşük seviyeli işlemcinin *PO* ucu, en yüksek seviyeli işlemcinin *PI* ucuna bağlanmıştır. Dolayısıyla veri yolunu kullanan işlemciden sonraki işlemci en yüksek öncelikli işlemci olur. İlk başlangıç haricinde öncelik böylece sırayla döner. Doğal olarak veri yolundan çıkan işlemci en düşük öncelikli işlemcidir.

13. 4 İşlemciler Arası İletişim ve Eş Zamanlılık

Çok işlemcili bir sistemde, her işlemci diğerleriyle iletişim kurabilmelidir. İletişim yolu ortak giriş çıkış kanlarıyla kurulabilir.Paylaşılan bellekli, çok işlemcili bir sistemde en çok kullanılan yöntem, belleğin bir kısmını tüm işlemcilere açmaktır. Bu kısmın ilk kullanılışında, burası posta kutusu gibi görev yapar. Her bir işlemci, diğer işlemciler için not bırakır veya kendisine yollananları alır.

Haberi yollayan işlemci, ya bir istekte bulunmuş, ya bir not yazmış veya bir işlem yapmıştır. Ve bunu posta kutusuna bırakır. Ortak bellekte bulunan durum bitleri, çoğunlukla posta kutusundaki durumları gösterirler. Buna göre posta kutusundaki bilgiler anlamlımıdır veya hangi işlemciye yollanmıştır gibi. Alıcı durumunda olan işlemci, sürekli olarak posta kutusunu tarar ve kendisi için geçerli haber var mı diye araştırır. Cevap zamanı oldukça uzundur. Çünkü bir işlemci istekleri yalnızca haberleri yoklama yaparken alabilir. Daha etkili bir yollama yöntemi, alıcı işlemciye bir kesme sinyali yollamaktır. Bunu sağlamak için, işlemciler arası kesme isteminin yazılım tarafından yollanmasıdır. Kesme istemi başlatmak için uygun bir buyruk bir işlemci tarafından icra edilince, ikinci işlemcide bir dış kesme istemi oluşmalıdır. Bu kesme istemini alan işlemci, bir haber yollandığını anlar.

Paylaşılan belleğe ek olarak, çok işlemcili bir sistemin başka paylaşılan kaynaklara gereksinimi vardır. Örneğin, bir sabit disk GÇİ ye bağlanmış olarak tüm MİB lere açık olabilir. Bu sayede sistem programlarının paylaşılması sağlanır. İki işlem birimi arasındaki iletişim yolu, bu iki MİB ile birleşik olan iki GÇİ bağlantısı ile sağlanabilir. Bu tip bir bağlantı, her MİB için diğer birimlerin birer G/Ç aleti olarak algılanmasını sağlar. Böylece her haber G/Ç yoluyla aktarılabilir.

Birçok işlemci tarafından, kaynakların kullanımında kargaşa olmaması için, kaynakların işlemciler tarafından kullanımı düzenleyecek kurallar olmalıdır. Bu görev, işletim sistemine verilmiştir. Çok işlemcili sistemlerde kullanılabilecek, üç ayrı kurulum vardır. Efendi-köle kurumu, ayrı

işletim sistemleri kurumu ve dağıtılmış işletim sistemi kurulumudur. Efendi-köle kurulumunda, bir işlemci efendi olarak tanımlanır. İşletim

sistemini o kullanır. Kalan işlemciler yani köleler, işletim sistemini kullanmaz. Eğer bir işlemci, işletim sisteminin yapacağı bir işe gerek uyarsa, efendiye bir kesme istemi yollar ve çalışan program kesilene kadar bekler.

Ayrı işletim sistemlerinde, her işlemci istediği işletim sistemi yordamını çalıştırır. Bu kurulum gevşek bağlı sistemler için daha uygundur. Çünkü her işlemci tüm işletim sisteminin bir kopyasına sahiptir.

Dağıtılmış işletim sistemi kurulumunda, işletim sisteminin yordamları, mevcut işlemciler arasında dağıtılmıştır. Belli bir anda işletim sisteminin bir fonksiyonu, sadece bir işlemcide çalışır. Bu tip işletim sistemine yüzen işletim sistemi adı verilir. Çünkü yordamlar bir işlemciden diğerine gidebilir ve icraları farklı zamanlarda farklı işlemciler tarafından yapılabilir.

Gevşek bağlı çok işlemcili bir sistemde, sistem belleği işlemciler arasında dağıtılmıştır ve paylaşılan bellek yoktur. Bu yüzden bu yolla bilgi aktarımı olmaz. İşlemciler arasında iletişim, G/Ç kanallarından haber yollanması yoluyla gerçekleştirilir. İletişim, işlemcinin birinin, haberleşmek istediği işlemcinin belleğinde bulunan bir işleme istekte bulunmasıyla başlar. Haberi yollayan ve alan işlemci birbirlerini kaynak ve hedef olarak tanıdıklarında bir haberleşme kanalı açılmış olur. Sonra bir başlıkla bir haber ve muhtelif veriler kullanılarak iki düğüm arasında haberleşilir. İki düğüm arasında haberleşebilmek için birçok mümkün yol olabilir. Her bir düğümdeki işletim sistemi, farklı yollar için kılavuz bilgisine sahiptir. Bunun yardımıyla, diğer düğümlere de haber yollanabilir. İşlemciler arası ağda iletişim etkinliği, haberleşme kılavuzuna, işlemci hızına, veri aktarma hızına ve ağın topolojisine bağlıdır.

İşlemciler Arası Zaman Uyumluluğu

Çoklu işlemcili bir sistemin buyruk kümesi, haberleşme ve zaman uyumluluk için temel buyrukları içerir. Haberleşme farklı işlemler arasında veri alışverişidir. Örneğin, farklı bir işlemcideki bir işleme yollanan veriler, işlemciler arası iletişimdir. Zaman uyumluluğu, özel bir durumdur. İşlemciler arasında haberleşmek için kullanılan veriler, denetim bilgileridir. Zaman uyumluluğu işlemlerin doğru sırada yapılmasını sağlamak ve yorulabilen verilere karşılıklı olarak etkin olacak biçimde erişebilmek için gereklidir.

Çok işlemcili sistemlerde kaynakların zaman uyumluluğunu sağlayacak muhtelif yöntemler vardır. Düşük seviyedeki işlemler, donanım tarafından sağlanır. Bu işlemler, yazılım tarafından karşılıklı olarak yerine getirilecek çok daha karışık işlemler için temel teşkil eder. Karşılıklı ola-

rak icra için geliştirilmiş bazı donanımlar mevcuttur. En iyi bilinen yöntemlerden birisi ikili semafor yöntemidir.

Semafor ile Karşılıklı Dışlamalar

Kritik Süreç

Düzgün çalışan çok işlemcili bir sistem, ortak belleğe ve diğer kaynaklara erişimde eşitliği sayacak düzenlemeleri içermelidir. Bu sayede verilerin iki veya daha fazla işlemci tarafından aynı anda değiştirilmesi önlenmiş olur. Bu olaya karşılıklı dışlama denir. Çok işlemcili bir sistemde, bir işlemci, diğer işlemcilerin kritik süreçte, paylaşılan kaynaklara erişimini önleyebilmeli veya o kaynakları diğer işlemcilere kilitleyebilmelidir. Kritik süreç, icrası başladıktan sonra mutlaka tamamlanması gereken bir programın icra sürecidir. Bu sürede hiçbir işlemci paylaşılan kaynaklara erişmemelidir.

Semafor, ikili bir değişkendir. Bir işlemcinin kritik süreçte olup olmadığını gösterir. Yazılım denetiminde olan bir bayraktır. Tüm işlemcilerin erişebileceği bir bellek adresinde bulunur. 1 olduğu zaman bir işlemci kritik süreçtedir. Dolayısıyla paylaşılan bellek, diğer işlemcilere kapalıdır. Bayrak 0 olunca paylaşılan kaynaklar açıktır. Aynı bellek bölgesini kullanan işlemciler, semafor bayrağı inene kadar yani 0 olana kadar, bu bellek bölgesini kullanmamak için anlaşmalıdır. Aynı zamanda, kritik sürece girdiklerinde bayrağı 1 yapmalı, işi bitince bayrak 0 yapılmalıdır.

Bir semaforun test ve doldur işlemi de kritik bir süreçtir. Dolayısıyla bölünmeden tek bir işlem olarak yerine getirilmelidir. Böyle yapılmazsa, iki veya daha fazla sayıda işlemci, semaforu test eder ve doldurur. Ve aynı anda kritik sürece girerler. Sonuçta denetim parametreleri yanlış değerler alır ve bilgiler kaybolur.

Donanım Kilidi

Bir semafor, test ve doldur buyruğu ve varolan donanım kilidi yardımıyla kurulabilir. Kilit, işlemci tarafından üretilen bir sinyaldir. Diğer işlemcilerin veri yolunu kullanmalarını engeller. Test ve doldur buyruğu, semaforu test eder, doldurur ve buyruk icra edildiği süre içinde kilidi çalıştırır. Böylece işlemci, semaforu test eder ve doldururken, diğer işlemcilerin semaforu değiştirmesini engeller. Varsayalım ki semafor bir bellek kelimesinin en önemsiz bitidir. Adresi SEM olsun sembolik buyruk TSL, test et ve doldur buyruğunu göstersin. Aşağıdaki buyruk

```
TSL    SEM
```

iki zaman sürecinde icra edilir. Bunlar

$R \leftarrow M[SEM]$ semaforu test et
$M[SEM] \leftarrow 1$ semaforu doldur

Semafor, değeri *R* işlemci yazacına aktarılarak test edilir. Sonra 1 ile doldurulur. *R* nin değeri sonra ne yapılması gerektiğini belirler. Eğer işlemci *R* = 1 olarak bulursa, onun daha önce doldurulduğunu anlar. Tekrar doldurmak bir şey ifade etmez. Dolu olmasının anlamı, başka bir işlemcinin kritik süreçte olduğu anlaşılır. Dolayısıyla paylaşılan belleğe erişmek mümkün değildir. Eğer *R* = 0 ise paylaşılan bellek serbesttir. Semafor 1 yapılır. Şimdi kritik sürece geçilebilir. Son buyruk SEM adresini sıfırlar. Paylaşılan bellek diğer işlemcilere açılır.

Test ve doldur buyruğunun icrası süresince kilit sinyali devam etmelidir. Semafor doldurulduktan sonra kilit sinyali susabilir. Çünkü semafor bu işi yapmaktadır.

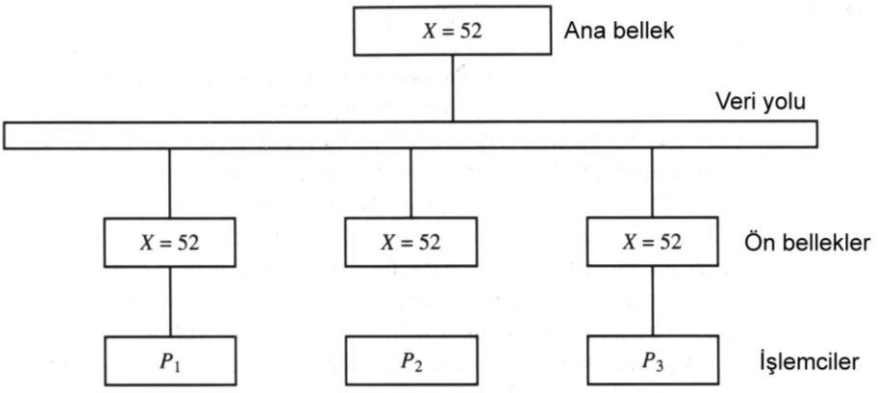
13-5 Ön Bellek Ardışıklığı

Ön bellek işlemleri bölüm 12.6 da anlatılmıştı. Ön belleğin en önemli faydası, tek işlemcinin belleğe erişim süresini çok azaltmasıdır. Eğer işlemci okuma işlemi sırasında bir kelimeyi ön bellekte bulursa, aktarımda ana bellek devreye girmez. Eğer işlem yazma işlemi ise, belleği güncelleştirmek için iki önemli yöntem vardır. *Beraber-yaz* yönteminde ön bellek ve ana bellek her yazma işleminde güncellenir. *Sonra-yaz* yönteminde sadece ön bellek güncellenir. Yer işaretlenir. Sonra ana belleğe aktarılır.

Paylaşılan belleği olan çok işlemcili sistemde tüm işlemciler ortak bir belleği paylaşırlar. Buna ek olarak her işlemcinin kendi özel belleği olabilir. Bunun bir kısmı da ön bellek olabilir. Her bir işlemci için ayrı bir ön bellek olmasının nedeni, her bir işlemcinin ortalama erişim süresinin kısaltılmasıdır. Aynı bilgiler ana bellekte ve ön belleğin bir kısmında olmak üzere birçok yerde birden yazılır. Sistemin bellek işlemlerini doğru olarak yapabilme yeteneğini sağlamak için tüm kopyalar birbirinin aynısı olmalıdır. Bu istek bir ön bellek ardışıklığı sorunu oluşturur. Şimdi ardışıklığı tanımlayalım. Bir bellek eğer okuma işlemiyle okunan bilgi daima en son saklama (yazma) işlemiyle o adrese yazılan bilgi ise bu sisteme ardışıktır denir. İki veya daha fazla işlemcisi bulunan veri yolu temelli çok işlemcili sistemlerde ön bellek ardışıklığı sorununa uygun bir çözüm bulunmazsa ön bellekler kullanılmamalıdır.

Ardışıksızlık Koşulları

Özel ön bellekleri bulunan çok işlemcili sistemlerde, yazılabilen verilerin paylaşılmasından dolayı ön bellek ardışıklığı sorunu vardır. Yalnızca okunabilen veriler emniyetli bir biçimde tekrarlanabilir. Bunun için ön



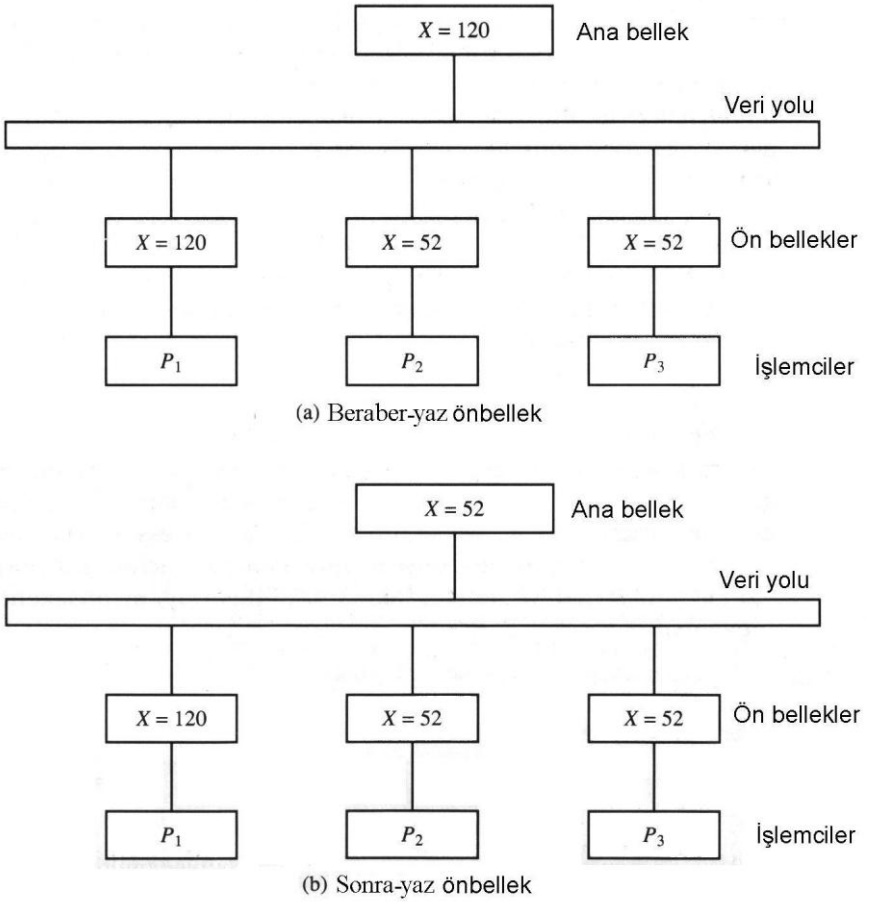
Şekil 13.12 *X* e bir yükleme yapıldıktan sonra ön bellek yapılanışı

bellek ardışıklığı kuvvetlendirme yöntemlerine gerek yoktur. Olayı daha iyi somutlaştırmak için üç işlemcili, özel ön bellekleri bulunan kurulumu düşünelim (Şekil 13.12). Bazen işlem sırasında ana bellekten *X* verisi, üç işlemci, P_1, P_2 ve P_3 e yüklensin. Sonuçta, veri üç işlemcinin de özel ön belleklerine yüklenmiş olur. Basitlik için *X* verisinin 52 olduğunu varsayalım. *X* in üç işlemciyle yüklenmesi, tüm ön belleklere ve ana belleğe yüklenir.

Eğer işlemcilerden birisi, *X* e başka bir değer yüklerse, ön bellekler içindeki *X* değerleri farklı olur. Diğer işlemciler tarafından yapılan işlemlerde dönen değer son değer olmaz. Ön bellek içinde kullanılan güncelleştirme yöntemine bağlı olarak, ana bellekte ön bellek ile uyumsuz olabilir. Bu durum şekil 13.13 te görülüyor.

Değeri 120 olan *X*, P_1 işlemcisinin ön belleğine yüklenince, beraber-yaz yöntemiyle, yeni değeri belleğe de yazar. Beraber-yaz işlemi ön bellek ile ana bellek arasındaki uyumu sağlar. Fakat diğer iki ön bellek uyumsuzdur. Çünkü hala içlerinde eski değer vardır. Sonra-yaz yönteminde, ana bellek yükleme anında güncellenmez. Diğer iki ön bellekteki ve ana bellekteki değerler uyumsuzdur. Ana bellek daha sonra ön bellek değeri yazılınca uyumlu hale gelir.

Uyumluluk sorunu gösteren bir diğer kurulum DMA (doğrudan belleğe aktarım) dır. Eğer DMA, sistem veri yoluna bağlı GÇİ yardımıyla çalışıyorsa bu durum ortaya çıkar. Giriş yapılırken, DMA, ana bellekteki adresleri değiştirir. Fakat ön belleklere dokunmaz. DMA çıkışı sırasında, sonra-yaz yöntemi kullanılıyorsa güncellenmeden önce bellek adresleri okunur. G/Ç temelli bellek uyumsuzluğu, GÇİ nin ön bellek uyumlu sistemin içinde olmasıyla önlenebilir.



Şekil 13.13 P_1 işlemcisi ile X e bir depolama yapıldıktan sonra ön bellek yapılanışı

Ön Bellek Uyumluluğu Sorununa Çözümler

Paylaşılmış belleği olan çok işlemcili sistemlerde ön bellek ardışıklığı sorununu çözmek için birçok çözüm yolu öne sürüldü. Bunlardan bazılarını kısaca görelim. Daha ayrıntılı tartışmalar için 3 ve 10 numaraları kaynaklara bakın.

Basit bir yöntem, tüm işlemcilerin ön belleklerini kapatmak ve paylaşılan bir ön bellek kurmaktır. Doğal olarak bu ön bellek, ana bellekle ortak olmalıdır. Tüm veri erişimleri, bu paylaşın ön belleğe yapılır. Bu yöntem MİB nin ön belleğe yakın olması, prensibini bozar ve ortalama bellek erişim zamanını uzatır. Gerçekte bu yöntem, sorundan kaçarak onu çözer.

Verimlilik düşüncesiyle her bir işlemciye bir tane özel ön bellek bağlamak arzu edilir. Bir yöntem sadece paylaşılmayan ve yalnız okunabilir verilerin ön belleğe yazılmasıdır. Böyle sistemlere ön belleklenebilir adı verilir. Paylaşılan yazılabilen veriler ön belleklenebilir değildir. Düzenleyici, verilere ön belleklenebilir ve ön belleklenebilir olmayan şeklinde başlıklar vermelidir. sistem donanımı ön belleklenebilir verilerin, ön belleklere yazılmasını sağlamalıdır. Ön belleklenebilir olmayan veriler ana bellekte kalır. Bu yöntem ön belleklere yazılan veri tiplerini kısıtlar. Ayrıca fazladan yazılım gerektirir. Buda verimi düşürür.

Bir diğer yöntemde, yazılabilir verilerin, en azından bir tane ön bellekte bulunmasıdır. Bu yöntemde derleyici içinde merkezselleştirilmiş küresel çizelge kullanılır. Bellek bloklarının durumu bu çizelgeye yazılır. Her bir blok, yalnız okunabilir (RO) veya okunur ve yazılabilir olarak belirlenebilir. Tüm ön belleklerde RO olarak adlandırılan blokların kopyaları vardır. Sadece bir tek blok RW bloğunun kopyasını bulundurur. böylece veriler ön bellek içinde RW bloğu ile güncellenirse diğer ön bellekler etkilenmez. Çünkü bu bloğun kopyası onlarda yoktur.

Ön bellek ardışıklığı sorunu yazılım ve donanımın beraberce kullanılmasıyla çözülebilir. Daha önce anlatılan iki yöntem, yazılım temelli idi. Bunların işlemesi için başlık bilgisine gerek vardır. Bu sayede yanılabilir bilginin ön belleğe gitmesi önleniyordu. Yalnız donanımla yapılan çözümler çok hızlıdır ve program saydamlığı vardır. Donanım çözümünde, ön bellek denetleyicisi özel olarak tasarlanmış ve MİB den veya GÇİ lerden gelen tüm istekleri yönetecek yetenekte yapılmışlardır. Veri yoluna bağlanmış olan tüm ön bellekler sürekli olarak ağı gözler ve muhtemel yaz işlemlerini bulurlar. Kullanılan yönteme bağlı olarak ya kendi ön bellek kopyalarını güncellerler veya geçersiz kılarlar. Bu hareketi yöneten veri yolu denetçisine Araştırıcı ön bellek denetçisi denir. Bu temelde bir donanımdır. Veri yolu gözlemi yapar ve veri yoluna bağlı ön bellekleri gözler.

Araştırıcı ön bellek tutanağı ile ön bellek ardışıklığı sorununa muhtelif çözümler önerilmiştir. En basit yöntem, bir beraber yazma yöntemi belirlemek ve sonra aşağıdaki işlemleri yapmaktır. Tüm araştırıcı ön bellek denetçileri, belleğe yazmak için veri yolunu gözlemektedir. Bir kelime ön belleğe yazılarak güncellendiğinde bellekte buna karşılık gelen adreste aynı anda güncellenir. Tüm araştırıcı denetçileri, kendi bellekleri ne bakarak yeniden yazılan kelimenin kopyasının bulunup bulunmadığını tespit ederler. Eğer bir kopya uzak bir ön bellekte mevcutsa bu adres geçersiz olarak işaretlenir. Çünkü tüm ön bellekler tüm veri yolu yazımında araştırıcıdırlar. Bir kelime yazıldığında bunun etkisi, ilk yazıldığı ön bellekte ve ana bellekte güncellemektir ve diğer tüm ön belleklerden kaldır-

maktır. Eğer bir zaman sonra bir işlemci geçersiz kelimeye ön bellekten erişirse, sonuç ön bellekte bulamamaktır ve kelime ana bellekten alınır. Böylece verinin güncellenmemiş biçimleri atlanmış olur.

SORULAR

13-1. Sıkı paket ve gevşek paket çoklu işlemcili sistemleri, donanım kurulumu ve program yapma açısından tartışınız.

13-2. Şekil 13.2 de gösterilen sistem veri yolu denetçisinin görevi nedir? Sistemin yerel belleklerle, paylaşılan ortak belleği ayırt etmesi nasıl ta sarlanabilir?

13-3. Çapraz anahtarlamalı ağda kaç tane anahtar ile m tane bellek modülü, p tane işlemciye bağlanabilir?

13-4. Şekil 13.8 deki 8×8 omega anahtarlamalı ağ, her kademede 4 anahtarla 3 kademede 12 anahtardan oluşur. Bir $n \times n$ omega anahtarlamalı ağı için kaç kademe ve her kademede kaç anahtar gerekir.

13-5. Varsayalım ki şekil 13.8 deki omega anahtarlamalı ağında, ilk satırda ikinci sütundaki anahtarla, ikinci satırda üçüncü sütundaki anahtar arasındaki tel kırılmıştır. Hangi yolların bağlantısı kesilir?

13-6. Bir 4×4 omega anahtarlamalı ağ şeması çizin. 3 girişini, 1 çıkışına bağlamak için gerekli bağlantıyı gösterin.

13-7. Çok kademeli iç bağlantılı ağda 3 tip anahtar kullanılır. Şekil 13.6 daki gibi 2 giriş, 2 çıkış iç çevrim anahtarı, iki girişli bir çıkışlı seçim kararı veren anahtar, ve iki çıkışı bir girişi olan dağıtım anahtarı

- **a.** Seçim kararı ve dağıtım anahtarları nasıl çalışırlar.
- **b.** Seçim kararı ve iç çevrim anahtarları kullanarak 8 x 4 bir ağ çizin. Bu ağda herhangi bir kaynakla herhangi bir hedef arasında tek bir yol olsun.
- **c.** Dağıtım ve iç çevrim anahtarları kullanarak 4×8 ağ kurun. Bu ağda her hangi bir kaynakla her hangi bir hedef arasında tek bir yol olsun.

13-8. 4 boyutlu hiper küpün yapısını gösteren bir şekil çiziniz. 7 düğümü 9 arasında en az sayıda düğümden geçen yolların listesini yapın.

13-9. Şekil 13.10 daki papatya zinciri karar aletinde, veri yolu karar düzenini, kapılar ve yaz-bozlar kullanarak çizin.

13-10. Şekil 13.11 de gösterilen paralel seçim kararı devresi başlangıçta boştur 2 ve 3 numaralı aletler aynı anda veri yolunu isterler. Kod değiştiricinin ve kod çözücünün giriş ve çıkış değerlerini, ikili olarak belirtin. Hangi alete veri yolu tahsisi yapıldığını söyleyin.

13-11. Şekil 13.10 da gösterilen seçim kararı devresi, nasıl döner papatya zinciri devresine dönüştürülebilir. Veri yolu hattının izinlendirilmemiş olduğunu belirleyen öncelik nasıldır? Açıklayın.

13-12. İki işlemcinin bellekteki bir buffer yardımıyla haberleştiği veri yolu topolojisini düşünün. İşlemcinin biri diğer işlemci ile haberleşmek isterse, bilgiyi bellek bufferına yazar ve bayrak çeker. Diğer işlemci bayrakları sırasıyla gözler ve alınacak bilgi var mı diye bakar. Bilgi yollamak ve almakta zamanı en aza indirmek ve zaman uyumluluğu sağlamak için ne yapılabilir?

13-13. Aşağıdaki deyimleri açıklayın.

(a) Karşılıklı dışlama; (b) Kritik süreç; (c) Donanım kilidi; (d) Semafor; (e) Test ve doldur buyruğu.

13-14. Ön bellek ardışıklığı nedir ve paylaşılan bellekli çok işlemcili sistemlerde neden önemlidir? Bu sorun araştırıcı ön bellek denetçisiyle nasıl çözülür?

KAYNAKLAR

1. Dasgupta, S., *Computer Architecture: A Modern Synthesis*. Vol. 1. New York: John Wiley, 1989.
2. DeCegama, A. L., *Parallel Processing Architecture and VLSI Hardware*. Englewood Cliffs, NJ: Prentice Hall, 1989.
3. Dubois, M. C., Scheurich, and F. A. Briggs, "Synchronization, Coherence, and Event Ordering in Multiprocessors" *IEEE Computer*, Vol. 21, No.2 (February 1988), pp. 9-21.
4. Gibson,G. A., *Computer Systems Concepts and Desing*. Englewood Cliffs, NJ: Prentice Hall, 1991.
5. Gorsline, G.W., *Computer Organization: Hardware/Software*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1986.
6. Hays, J. F., *Computer Architecture and Organization*. 2nd ed. New York: McGraw-Hill, 1988.
7. Hwang, K., and F. A. Briggs, *Computer Architecture and Parallel Processing*. New York: McGraw-Hill, 1984.
8. Kain, R., *Computer Architecture: Software and Hardware*. Vol. 2. Englewood Cliffs, NJ: Prentice Hall, 1989.
9. Langholz, G., J. Francioni, and A. Kandel, *Elements of Computer Organization*. Englewood Cliffs, NJ: Prentice Hall, 1989.
10. Stenstrom, P., "A Survey of Cache Coherence Schemes for Multiprocessors" *IEEE Computer*, Vol. 23, No.6 (June 1990), pp. 12-24.
11. Stone, H. S., *High-Performance Computer Architecture*, 2nd ed. Reading, MA: Addison-Wesley, 1990.
12. Tabak, D., *Multiprocessor*. Englewood Cliffs, NJ: Prentice Hall, 1990.

Dizin

A

AC, 122
Açıklama alanı, 170
Adres, 57, 125
Adres alanı, 213
Adres haritalaması, 443, 447
Adresleme kipleri, 244
  ani, 245
  doğrudan, 246
  dolaylı, 246
  göreceli, 247
  indislenmiş, 247
  otomatik artma, 246
  taban yazacı, 247
  yazaç, 246
  yazaç dolaylı, 246
Adreslenebilir bellek, 57, 422
Adres sembol çizelgesi, 173
Adres sayıcı, 176
Adres uzayı, 441
Adres veri yolu, 392–394
Adres yazacı, 122
Ağırlıklı kod, 81
  aktarma buyruğu, 241
Aktarım hızı, 374
Aktarım kipleri, 379
Alet, 359
Alfasayısal kod, 71, 82, 361
Al-getir evresi, 133–135, 149, 213
algoritma, 314
Alt yordam, 187, 260–61
Alt yordam çağırma, 142, 188–9, 260
Alt yordam parametreleri, 188–89
  veri bağlantısı, 189
Alt yordam programı, 187
Alt yordam yazacı, 188–9, 260
Alt yordamdan geri dönüş, 260
ALU, 111–112, 227–231
Amaç programı, 174
Ana bellek, 419–22
Ana-uydu yaz-bozu, 26
AND mikro işlemi, 104–106
  özel sembol, 105
Ani (Derhal) kip, 120, 245
AR (bak. Adres yazacı)
Ara yüzey, 362
Arama-çizelgesi, 177
Araştırıcı ön bellek denetim birimi, 482
Aritmetik algoritma, 314
Aritmetik boru hattı, 289
Aritmetik buyruklar, 251
Aritmetik devre, 95–103
Aritmetik işlemci, 313
Aritmetik kaydırma, 108–111, 254, 324
  taşma, 109
Aritmetik mantık birim, 111, 229
Aritmetik mikro işlemler, 97
Arttırıcı, 100–1
Arttırma, 55, 98, 103
ASCII, 72–73, 174, 361–362
Atlama, 211, 256
Ayrıştırılmış bellek, 305
Azaltma, 103

B

Başla Denetimi, 369
Başlangıç biti, 373
Başlık alanı, 170, 213, 437
Bayrak, 145, 413
Bayrak bitleri, 256
BCD, 70–71, 78–79
BCD çıkarıcı, 345–47
BCD toplayıcı, 343–45
Buffer kapısı, 5, 95
Belleğe yazma, 58, 97
Belleği okuma, 58, 96
Bellek:
  erişim zamanı, 435
  FIFO, 377
  IC, 39, 422–23
  içeriği adreslenebilir, 429
  içerik adreslemeli, 429
  ön bellek, 434–441
  rasgele erişilebilir, 57, 422
  sanal, 441
  yalnız okunabilir, 59, 422
  yardımcı, 427–29
  yonga, 422–23
Bellek adresi, 58, 125
Bellek adres haritası, 424–26
Bellek adreslemeli buyruklar, 138–43
Bellek aktarımı, 96–97
Bellek ayrıştırma, 305
Bellek çevrimi, 132

Bellek dizisi, 305
Bellek haritalanmış G/Ç, 365–66
Bellek hiyerarşisi, 419
Bellek kelimesi, 57, 97
Bellek koruma, 453
Bellek modülü, 464
Bellek organizasyonu, 419
Bellek sayfa çizelgesi, 444, 449
Bellek uzayı, 441
Bellek veri yolu, 364, 426
Bellek yığıtı, 234–36
Bellek yönetimi, 447–50
Berkeley RISC I, 269–272
Bilgisayar ağı, 459
Bilgisayar aritmetiği, 313
Bilgisayar buyruk çevrimi, 133
Bilgisayar buyrukları, 126–128, 166
Bilgisayar donanımı, 1
Bilgisayar mimarisi, 3
Bilgisayar organizasyonu, 3, 150–59
Bilgisayar tasarımı, 3, 150–59
Bilgisayar yazaçları, 121–123
Bilgisayar yazılımı, 2, 165
Bir adresli buyruklar, 243
Bir sayının tabanı, 122
Birikeç, 122
Birikeç yazıcı, 122–124
Birikeç yazıcı tasarımı, 156–158
Birleşik devre kaydırıcı, 110
Birleşik devreler, 18
Birleme, 22
  analizi, 19
  tasarımı
Birleştirici dil, 170–174
Birleştirici dil derleyicisi, 172, 176–179
Bit, 1
Bit esaslı tutanak, 412–3
Bit işleme buyrukları, 253–4
Bit tümleme, 253
Bitiş biti, 377
Bitişik kareler, 12
Bit-silme, 253
Blok, 443
Blok aktarım, 407
Blok şema, 18
Boole cebri, 7–11
Boole fonksiyonu, 7
Boole ifade, 8
Booth algoritması, 321–25
Boru hattı işlemi, 281
Boru hattı, 284–8
  aritmetik, 289–91
  buyruk, 291–6
  RISC te, 297–300
Boru hattı çelişmesi, 294
Bölen, 328
Bölenin hizalanması, 340
Bölme algoritmaları:
  iadeli yöntemi, 331
  işaretli-büyüklük, 328
  karşılaştırma yöntemi, 331
  kayan-noktalı, 340
  ondalık sayılar, 351–53
Bölme taşması, 330
Bölünen, 328
Bulma oranı, 435
Buyruk akımı, 282
Buyruk boru hattı, 291–96
Buyruk çevrimi, 133, 291–93
Buyruk biçimi, 239, 303
Buyruk kodu, 117, 126
Buyruk kümesi tamlığı, 128–29
Buyruk yazacı, 122
Buyruklar, 126, 166
Büyük ölçekli birleştirilmiş aletler, 40
Büyüklük, 315
Büyüklük karşılaştırıcı, 315–16
Byte, 57

C

CAM, 429
CAR(bak. Denetim adres yazacı)
CISC, 264
CMOS, 41
CRC, 407

Ç

Çakıştırma mantığı, 432
Çapraz bağlantı anahtarı, 464
Çarpan, 320
Çarpılan, 320
Çarpım, 319
Çarpımların toplamı, 14

Çarpma algoritmaları:
Booth algoritması, 321–25
işaretli 2 ye göre tümleyen, 321–5
işaretli gösterim, 320–1
kayan-noktalı, 338–40
onluk sayılar, 350–1
yazılım kurulumu, 182–4
Çarpma programı, 185
Çevre birimi, 360
Çevrim içi, 360
Çıkan, 99–100, 318
Çıkarılan, 317
Çıkarma algoritmaları:
işaretli 2 ye göre tümleyen, 84, 318
işaretli gösterim, 314–18
işaretsiz sayılar, 73
kayan noktalı, 336–8
ondalık sayılar, 345–7, 348
Çıkarma mikro işlemi, 98
Çıkış bayrağı, 145
Çıkış eldesi, 74
Çift duyarlı, 183
Çift yönlü kaydırma yazacı, 51–53
Çift yönlü veri yolu
Çok işlemcili, 282, 287
Çok kapılı bellek, 463
Çok seviyeli anahtarlamalı ağ, 466–7
Çoklu işlemci, 459
birbirine bağlantılı, 461–70
eş zamanlı, 477
iletişim, 476
seçim kararı, 470–75
Çoklu fonksiyonel birimler
Çoklu programlama, 420
Çökme, 334

D

D tipi yaz-boz, 23
Dağıtılmış anahtar, 483
Dağıtılmış bellek, 461
Dahili kesme, 263
Dahili yazaç aktarımı, 95
Dairesel kaydırma, 108
Dallanma, 141, 257
Dallanma buyrukları, 255–258, 295–96
Dallanma hedef bufferı, 296
Dallanma koşulları, 258
Dallanma mantığı, 206
Değişim anahtarı, 466
DeMorgan teoremi, 8–10
Denetim adres yazacı, 202, 204
Denetim belleği, 201–202, 216
Denetim birimi, 130–133
Denetim emri, 364
Denetim fonksiyonu, 91
Denetim kelimesi, 201, 229–231
Denetim mantık kapıları, 150
Denetim sinyalleri, 130–132
Denetim tasarımı, 150–56
Derleyici, 180
Derleyici desteği, 297
Devre sadeleştirme, 11–18
Dikkate almama, 17
Dinamik giriş, 22–23
Dinamik mikro programlama, 202
Dinamik seçim kararı birimi, 475
Disk, 360, 428
Disket, 428
Dizi çarpıcı, 326–7
Dizi işlemci, 307
Eklenmiş, 307–8
SIMD, 308–9
DLE karakteri, 412
DMA (Bak. Doğrudan bellek erişimi)
Doğrudan adres, 120, 246
Doğrudan bellek erişimi, 392
aktarım, 395
denetim birimi, 393
Doğrudan haritalama, 437
Doğrudan yaz, 440
Doğruluk çizelgesi, 4
Dolaylı adres, 120–1
Donanım, 2, 165
Donanım algoritmaları, 314
Donanım iç kilitleri, 295
Donanım kilidi, 478
Donanımsal denetim, 130
Döndürmeli papatya zinciri, 476
Dönüşsel artıklık denetimi, 407
Dönüştürülmüş içerik adreslemeli bellek, 450
DR (bak. Veri yazacı)
Dur biti, 374
Durum bitleri, 206, 256
Durum çizelgesi, 23, 29–30
Durum emri, 364
Durum şeması, 31–32

Durum yazacı, 381–82

E

EBCDIC, 82
ECL, 40
EEPROM, 60
Eklenmiş dizi işlemci, 307–8
Ekran, 360
Elde, 19, 73–4
Elde durum biti, 256
Entegre devre, 39–41
EPROM, 60–61
Erişim zamanı, 435
Eş değerlik, 5
Eş zamanlı aktarım, 406
Eş zamanlı olmayan aktarım, 368–373
Eş zamanlı olmayan haberleşme, 375–77
Eş zamanlı olmayan veri yolu, 471
Eş zamanlı silme, 55
Eş zamanlı veri yolu, 461, 471
Eş zamanlılık, 476–7
Eşlik biti, 83
Eşlik denetimi, 83–4
Eşlik üreteci, 83–4
Etiket, 170, 213
Etkin adres, 120
Excess-3 kod, 81

F

FA (bak. Tam toplayıcı)
FIFO, 377, 447
FIFO buffer, 377–9, 475
Flop, 306
Fortran, 170

G

G/Ç, 359
G/Ç aktarımı, 368, 375
G/Ç ara yüzeyi, 362–64, 366–68
G/Ç buyrukları, 145
G/Ç kapısı, 366
G/Ç veri yolu, 363–65
G/Ç yazılımı, 191
Gecikmeli dallanma, 296, 300
Gecikmeli yükleme, 295, 298
Geçerli biti, 441
Genel yazaç tasarımı, 228–232
Geri dönüş adresi, 141
Geri yaz, 440
Gevşek paket çoklu işlemciler, 461
Giriş bayrağı, 146
Giriş denkliği, 28
Giriş eldesi, 101
Giriş mantığı, 220
Giriş-Çıkış, 144–49, 359
  Aletleri, 359–61
  Aktarım, 368–77
  ara yüzey, 362–64, 366–68
  buyruklar, 145
  programlama, 191
  veri yolu, 363, 397
Giriş-çıkış işlemcisi, 396–99
Göreceli adresleme, 247
Görev, 286
Gösterge, 182
Grafik semboller, 22
Gray kod, 80

H

Haberleşme denetim karakterleri, 408
Harici kesme, 263
Haritalama, 204, 206–7, 435
Hata bulma, 178
Hata bulma kodu, 83
HDLC, 413
Hızlandırma, 287
Hiper küp, 468–70

I

IBM kanal, 400–402
IC (Bak. entegre devre)
Intel-8089 GÇİ, 402
IR (Bak. Buyruk yazacı)

İ

İcra evresi, 133, 215
İç üretim, 304
İçerik adreslemeli bellek, 429
  beraber yaz, 440
  donanım organizasyonu, 430–34
  sonra yaz, 440

İçerik adreslemeli bellek:
haritalaması, 436
İki adresli buyruklar, 242
İkili, 1
İkili ağaç, 467
İkili arttırıcı, 100–1
İkili değişken, 7, 201
İkili işaret, 4
İkili kodlanmış onluk, 70, 82
İkili kodlar, 70, 81
Alfanümerik, 71
ASCII, 71
BCD, 70, 82
Excess-3, 81
Gray, 80
hata bulma, 83
ikili n-küp, 468–70
İkili nokta, 75
İkili paralel toplayıcı, 98–100
İkili sayıcı, 53
paralel yüklemeli, 55
tasarımı, 32, 34
İkili sayılar, 2,66
İkili toplayıcı, 98
İlk giren ilk çıkar, 377–79
İma edilmiş kipi, 245
İndirgenmiş buyruk kümeli bilgisayar, 264–69
İndis adresleme, 247
İşaret, 4
İşaret durum biti, 256
İşaretli 2 ye göre tümleyen, 75
taşma, 75
İşaretli gösterim, 75
İşaretli sayılar, 75
İşaretsiz sayılar, 73
çıkarılması, 73
İşlem kapasitesi, 281
İşlem kodu, 118
İşlemci yazacı, 156–159
tasarımı, 156–159
İşlemciler arası bağlantı, 461–70
İşlemciler arası eş zamanlama, 476–7
İşlemciler arası seçim, 470
İşlemcilerarası zaman uyumu, 477
İşletim sistemi, 421, 454, 477
İz, 428
İzin girişi, 41

J

JK tipi yaz-boz, 24

K

Kalan, 328–29
Kanal, 400
Kapılar, 4–6
Karakter esaslı tutanak, 408
Karakter işleme, 191
Karakter kod, 71
Karakteristik çizelge, 22
Karnaugh haritası, 11
Kayan noktalı, 79, 333–36
aritmetik işlemler, 336–42
normalizasyon, 79, 333
Kayan noktalı boru hattı, 289–291
Kaydırıcı, 109–10
Kaydırma buyrukları, 348–50
Kaydırma işlemleri, 186
Kaydırma mikro işlemleri, 108
donanım kurulumu, 109–10
Kaydırma yazacı, 49
Kaynak çelişmesi, 294
Kaynak program, 174
Kelime, 57
Kelime sayma yazacı, 394
Kenar tetiklemeli yaz-boz, 25
Kendini tümleme, 81
Kesim, 284, 448
Kesim betimleyici, 453
Kesir, 79, 333
Kesme, 146–48, 261, 383–88
çeşitleri, 263–64
öncelik, 407
vektörlenmiş, 382
Kesme algılama, 388
Kesme başlatmalı G/Ç, 381
Kesme çevrimi, 389
Kesme istemi, 394
Kesme programı, 193
Kesme yazacı, 387–88
Kısıtlı üs, 337
Kısmi çarpım, 14, 521, 524
Kısmi kalan, 328
Kip alanı, 245–47
Klavye, 360
Kod çözme aşaması, 133

Kod çözücü, 41
Kodlayıcı, 44
Komutlar, 364, 398
Koruma, 453
Koşul kodu, 256
Koşullu dallanma, 257, 59
Kritik kesim, 478
Kurma zamanı, 25
Küçük ölçekli birleşik aletler, 39
Küme tipi içerik adreslemeli, 439

L

LD (Bak. Yükleme denetimi)
LIFO, 232
LRC, 407
LRU, 447
LSI, 40

M

M (bak. Bellek kelimesi)
Makine dili, 167–70
Mantık devre ailesi, 40
Mantık devresi, 105
Mantık kapıları, 4
Mantık şeması, 7
Mantıksal adres, 448
Mantıksal buyruklar, 253–4
Mantıksal işlemler, 186
Mantıksal kaydırma, 108, 254
Mantıksal mikro işlemler, 104
  donanım kurulumu, 105
  listesi, 104–5
Manyetik silindir, 427
Manyetik teyp, 360, 429
Maskeleme işlemi, 108
Maskeleme yazacı, 387
Matematiksel gösterim, 237
Matris çarpımı, 303
Merkezi işlemci birimi, 227
Mesaj biçimi, 409
Mesaj-geçme, 461
Metal oksit yarı iletken, 41
MIMD, 282, 459
MİB, 3, 227
Mikro buyruk, 202
  biçimler, 210–12
  ikili şekil, 216–17
  sembolik şekil, 214–17
Mikro işlem, 89, 97, 210, 231
  aritmetik, 97–8
  kaydırma, 108–9
  mantık, 103–5
  okuma, 96–7
  yazaç aktarımı, 97
  yazma, 97
Mikro işlemci, 460
Mikro program, 202
  denetim, 217–19
  ikili, 216
  örnek, 208–17
  sembolik, 214–16
  sıralayıcı, 219–21
Mikro programlanmış denetim birimi, 202
Mimari, 3
  interm, 11
  intermler toplamı, 11
Modem, 406
MOS, 41
MSI, 40
MUX (bak. Seçici)

N

NOR kapısı, 5, 9, 16
Normalizasyon, 80, 334–35

O

Okuma, 58, 97
Okuma mikro işlemi, 97
Omega ağ, 467–8
Onaltılık kod, 127, 168
Onaltılık sayılar, 66
  ikili kodlanmış, 69
  ikiliye dönüştürme, 68
Onaylama, 368, 370–73
Ondalık aritmetik, 347
  çarpma, 350
  bölme, 351
  toplama ve çıkarma, 348
Ondalık aritmetik birim, 342
Ondalık çıkarma, 345
Ondalık gösterim, 78
Ondalık kodlar, 81
Ondalık sayılar, 66, 78
Ondalık toplayıcı, 343–45

OR Kapısı, 5–6
OR mikro işlemi, 105
özel sembol, 104
ORG, 171
Orta ölçekli birleşik aletler, 40
Ortak veri yolu, 93, 123, 228–30
Otomatik artma, 246
Otomatik azalma, 246

Ö

Ön bellek, 420, 434–41
Ön bellek ardışıklığı, 482
Ön yükleme yükleyicisi, 422
Öncelik kesmesi, 383–88
papatya-zinciri, 384
paralel, 386
yazılım yordamları, 389
Öncelik kodlayıcı, 388
Öncelik mantığı, 386

P

Paket, 461
Papatya zinciri, 385, 473
Paralel aktarım, 49
Paralel işlem, 281
Paralel seçim kararı birimi, 474–6
Paralel toplayıcı, 315–16
Paralel yükleme, 51
Paralel yüklemeli yazaç, 49
Patlama aktarımı, 393
Paylaşılmış bellek sistemi, 461–2
PC (bak. Program sayıcı)
Program, 2, 165
Program denetim buyrukları, 255
Program döngü, 180
Program durum kelimesi, 180
Program kesme, 146–49, 193–95, 261
Program sayıcı, 123
Programlama dilleri, 168
Programlanmış G/Ç, 379–83
PROM, 60
PSW, 261

R

RAM, 57, 422
RISC, 265–66
buyrukları, 244, 269
boru hattı, 297–301
ROM, 59, 202, 216, 422
RPN, 237

S

Saat, 22, 25, 92
Saat çevrimi, 131, 285
Saatli sıralı devre, 22
Sabit noktalı, 75
ikili gösterim, 76–77
onluk gösterim, 78–79
Sakla, 140
Saklanmış program organizasyonu, 119
Sanal adres, 441–2
Sanal bellek, 441
Sayaç, 53–56
tasarımı, 32–35
Sayfa, 443
Sayfa çerçevesi, 446
Sayfa çizelgesi, 445
Sayfa hatası, 446
Sayfa yer değiştirme, 446–47
Sayı sistemi, 66
Sayı taban dönüşümü, 66
Sayısal, 1
Sayısal bilgisayar, 1
Sayısal entegre devreler, 39
Sayısal mantık ailesi, 40
Sayısal sistem, 1
Schottky TTL, 40
SDLC413
Seçici, 45–7, 93–4
Seçim girişleri, 46
Seçimlilik-doldurma, 106
Seçimlilik-silme, 107
Seçimlilik-tümleme, 107
Sekiz sistemindeki sayılar, 67–71
ikili kod, 68
ikiliye dönüştürme, 67
Sektör, 428
Semafor, 478
Sembol çizelgesi, 173
Sembolik adres, 170
Sembolik mikro program, 214
Sembolik program, 167, 174
Seri bağlantı ara yüzeyi, 144
Seri giriş, 50

Servis programı, 389
Servis yordamı, 194
Sıfır adresli buyruklar, 243
Sıfır algılama, 257
Sıfır durum biti, 256
Sıfır yollama, 413
SIMD, 282, 308
Sıkı paket çoklu işlemci, 461
Sıralayıcı, 203, 219–21
Sıralı devre, 27
  Tasarımı, 32–35
  Sıra yasıcı, 320
SISD, 282
Silme, 22–3, 107
Sistem veri yolu, 463
Sonraki aşama, 23,30
Sözde-buyruk, 171
SP (bak. Yığıt göstergesi)
SR tipi yaz-boz, 22
SSI, 39
Süper bilgisayar, 306
SYN karakteri, 408

T

T tipi yaz-boz, 24
Taban, 66
Taban yazacı, 247
Tam çift yönlü, 407
Tam-toplayıcı, 20–22, 99
Tamamlayıcı, 453
Tersleyen kapısı, 5
Taşma, 77
Taşma durum biti, 256
Tek fonksiyon, 6, 83
Tek yönlü, 407
Tek-çevrimli buyruk icrası, 297
Temel bilgisayar, 117
  akış şeması, 151
  seri haberleşme, 406
  buyruk çevrimi, 133, 136, 151
  buyruklar, 126–8, 166–7
  genel özet, 152
  denetim, 129–33
  giriş-çıkış, 144–6
  kesme, 146–8
  ortak veri yolu, 123–5
Temel bilgisayar için akış şeması, 151
Temel bilgisayar tasarımı, 150–56
Temel bilgisayar yazaçları, 122, 150
Test ve doldur, 478
Teyp, 360, 429
TLB, 450
Toplama algoritmaları:
  işaretli gösterim, 314–317
  işretli 2 ye göre tümleyen, 76, 317–319
  kayan-noktalı, 336–338
  ondalık sayılar, 343–345, 348–350
Toplama mikro işlemi, 97–9
Toplamada-taşma, 317
Toplamların sonucu:
  toplanan, 98–99
  toplayan, 317
Toplayıcı ve mantık devresi, 124–159
Toplayıcı-Çıkarıcı, 99–100
TTL, 40
Tutanak, 408
Tutma zamanı, 26
Tümleme, 10, 72–74
  çıkarma ile, 73
Tümleyici, 316

U

UART, 375
Uzay-zaman şeması, 286–87

Ü

Üç adresli buyruklar, 242
Üç durumlu kapı, 95
Üs, 79, 333
Üst üste binen pencereler, 266–69

V

VE-DEĞİL kapısı, 5, 9, 16
Vektör adresi, 384
Vektör işlemleri, 301
Vektör işleme, 302
Vektörlü kesme, 382
Veri akımı, 282
Veri aktarım buyrukları, 249
Veri bağlantısı, 407
Veri beklentisi, 295
Veri çeşitleri, 65, 251
Veri haberleşme, 405

Veri haberleşme tutanağı, 408
bit esaslı, 412–3
karakter esaslı, 408–13
mesaj biçimi, 409–11
örnek, 409–11
Veri işleme buyrukları, 251
Veri kümesi, 406
Veri saydamlığı, 411
Veri seçici, 46
Veri sunma, 65
Veri yazacı, 122
Veri yolu, 93, 94, 418–19, 470
Bağlantısı, 95–69
Bellek, 425–426
giriş-çıkış, 363
temel bilgisayarın, 123–25
Veri yolu aktarımı, 93–5
Veri yolu istemi, 392
Veri yolu organizasyonu, 228–230
Veri yolu seçim, 94–95
Veri yolu seçim kararı, 472–76
Veri yolu tahsisi, 392
VEYA kapısı, 5
VEYA-DEĞİL kapısı, 5
VLSI, 40, 460

Y

Yada, 6, 105
Yada değil, 5
Yakalama
Yalıtılmış G/Ç, 365
Yalnız okunabilir bellek, 59–60, 422
çeşitleri, 60
doğruluk çizelgesi, 59
programlanması, 60–61
Yardımcı bellek, 419, 427–429
Yarı çift yönlü, 407
Yarı-toplayıcı, 19
Yazaç, 48, 90
Yazaç adresi, 240
Yazaç adreslemeli buyruklar, 137–8
Yazaç aktarım dili, 92
temel semboller, 92
Yazaç aktarım durumları:
aritmetik, 97
kaydırma, 108
mantık, 103–5
şartlı kontrol, 91
Yazaç aktarımı, 90–91
Yazaç dolaylı kipi, 246
Yazaç kipi, 246
Yazaç yığıtı, 232–36
Yazaç yükleme, 48
Yaz-boz, 22, 28
ana-uydu, 26
çeşitleri, 22–25
eş zamanlı olmayan silme, 26
IC, 39–41
giriş fonksiyonu, 28
kenar tetiklemeli, 25

yükseltgeme çizelgesi, 26
Yazıcı, 360
Yazılım, 2, 165, 383, 389
Yazılım kesmesi, 263
Yazma, 58, 97
Yazma mikro işlemi, 97
Yer değiştirme algoritmaları, 446–47
Yerel veri yolu, 462–63
Yığıt, 232
Yığıt buyrukları, 243
Yığıt göstergesi, 247
Yığıt işlemleri, 239
Yığıt itme, 233
Yığıt limiti, 236
Yığıt organizasyonu, 232–34
Yığıtı çekme, 233
Yoklama, 384, 475
Yonga, 39, 422–3
Yonga seçim, 423
Yordam, 203
Yönetici kipi, 261–62
Yükle girişi, 123–25
Yükleme denetimi, 49, 123
Yüksek empedans, 95
Yükseltgeme çizelgesi, 26, 33

Z

Zaman bitti, 373
Zaman dilimi, 475
Zaman şeması, 26, 368
Zamanlama işareti, 130–32
Zamanlama ve denetim, 129
Zıt polonyalı gösterimi, 236–38
Aritmetik ifade, 238