

ENDÜSTRİYEL OKULLAR İÇİN

# Bilgisayar Donanımının Temelleri

Alan Clements



Endüstriyel Okullar İçin Bilgisayar Donanımının Temelleri

Satış fiyatı .....: 109.000 Lira  
KDV .....  
KDV'li SATIŞ FİYATI ..... 155000

#### TOPTAN SATIŞ

İstanbul Devlet Kitapları Müdürlüğü, Adana, Ankara, Burdur, Elazığ,  
Erzurum, İzmir, Samsun, Sivas, Trabzon, Van ve Zonguldak  
Bölge Şeflikleri.

#### PERAKENDE SATIŞ

Millî Eğitim Yayınevleri ve Bakanlık yayınları satıcısı kitapçılar.

2715

76

30 01 97  
F.B

ENDÜSTRİYEL OKULLAR İÇİN

# Bilgisayar Donanımının Temelleri

Alan Clements



Evren Ofset A.Ş. Web Ofset Tesisleri ANKARA - 1994

## SUNUŞ

Varlıklarını sürdürmek isteyen toplumlar, kalkınmanın gerektirdiği sayıda nitelikli insangücünü yetiştirmek için eğitime değer vermek ve ona bilimsel ve teknolojik bir nitelik kazandırmak mecburiyetindedirler.

Eğitim, Cumhuriyetin kuruluşundan beri ülkemizde yenileşme aracı olarak görülmüştür. Bugün Eğitim sistemimiz, bilim çağına girilen dünyamızda, toplumumuzun büyüyen ve çeşitlenen ihtiyaçlarına cevap vermede bir takım problemlerle karşı karşıyadır.

Eğitimle ilgili problemlerin çözümünde, yeni yöntemler, teknikler ve araçlar geliştirmek için araştırmalar yapmak, ayrıca daha önce yapılmış araştırmalar sonucu geliştirilen bilgi ve teknolojiyi ülkemize getirmek zorundayız.

Eğitime ayrılacak finansman kaynaklarının sınırlı olması, ülkemizi, genel bütçe dışındaki imkanlardan faydalanmaya zorlamaktadır. Devletimiz bu imkanları araştırmış, mesleki ve teknik öğretim kurumlarımızın bilim ve teknolojide meydana gelen gelişmelere paralel olarak modernleştirilmesi için Uluslararası İmar ve Kalkınma Bankası (Dünya Bankası - IBRD) ile yapılan ikraz Anlaşmasıyla Endüstriyel Okullar Projesi uygulamaya konulmuştur.

Bu projenin amaçları; Endüstriyel Okulların yeni teknoloji ürünü makina ve teçhizatla donatılarak yenilenmesi, çeşitli meslek alanlarında müfredat programlarının geliştirilmesi, burslar ve yurt dışından danışman temin edilmesi yoluyla öğretmenlerimizin eğitilmesi ve çeşitli meslek alanlarında ders kitaplarının tercüme ve yayın haklarının satın alınarak Eğitim Sistemimize kazandırılmasıdır.

Proje ile belirlenen hedeflere büyük ölçüde ulaşılmıştır. Projenin amaçlarından biri olan çeşitli meslek alanlarında (Hidrolik - Pnömatik, Soğutma ve İklimlendirme, CNC, Döküm, Elektronik, Bilgisayar, PLC ve Metal İşleri) teknik ders kitapları, uzmanlardan kurulu komisyonlarca seçilmiş ve tercüme edilerek yayımlanmıştır.

Büyük kaynak ve emek harcayarak Eğitim Sistemimize kazandırdığımız kitapların öğretmen ve öğrencilerimize faydalı olmasını dilerim.

Salih ÇELİK  
Projeler Koordinasyon  
Kurulu Başkanı

## YAZARIN ÖNSÖZÜ

Bu kitap, bilgisayar donanımı ilkeleri üzerine ilk düzey kursları alan üniversite ve politeknik öğrencileri için hazırlanmıştır.

Bilgisayar donanım ilkeleri kursu, iki amaçla düzenlenmiştir. Bunlardan ilki; öğrencilere, saklı-programlı sayısal bilgisayarların temel kavramlarını öğretmektir. Ele alınacak konular, ikili düzendeki bilgilerin gösterimi ve kullanımı, bir bilgisayarın yapısı ile içindeki bilgi akışını ve çeşitli çevre birimleriyle olan bilgi alışverişini kapsamaktadır. Daha açıkçası bu kitap, "Bilgisayarlar nasıl çalışır?" sorusuna yanıt vermektedir. İkincisi; kurslarının ikinci yılında görecekları daha ileri düzeydeki kurs için sağlam bir temel oluşturma amacına yöneliktir. Özellikle, kapılar ve Boole cebrinin giriş düzeyinde ele alınması, Bilgisayar Tasarımı'nın ikinci-düzen kursu için bir temel oluşturmaktadır.

Şu anda, bilgisayar bilimleri, okulların tümünde okutulmamakta olup bilgisayar ön bilgisi üniversite ve politeknik kurslarına devam için bir ön koşul değildir. Sonuç olarak, bu kitabın düzeyi, bilgisayar konusunda ön bilgisi bulunmayan kişilere yönelik olarak hazırlanmıştır. Bölüm 2'de Boole cebri ile çalışılacağından, okuyucu için yalnızca temel cebir kurallarının bilinmesi yeterli olacaktır. Bilgisayar bilimleri veya bilgisayar teknolojisi kursunu izleyen tüm öğrencilerin aynı zamanda yüksek düzeyli programlama (örneğin PASCAL) dillerini de öğrenmekte oldukları varsayımıyla, burada bu konuya girilmeyecek, okuyucunun yüksek düzeyli bir dile ilişkin temel kavramlara tanıdık olduğu varsayılacaktır.

Boole cebri, mantık devreleri, ikili aritmetik, bilgisayar mimarisi, bellek teknolojisi, assembly-dili ile programlama gibi çeşitli konuları tek tek ve kapsamlı bir biçimde işleyen bazı kitaplar bulunmasına karşın, bu konuların tümünü yeterli bir düzeyde inceleyen az sayıda eser bulunmaktadır. Buradaki amacım, okuyucunun yukarıda adı geçen konuların tümüne bir iç bakış sağlamasıdır.

Bu kitabı yazarken üç şeyi yapmaya çalıştım. Geleneksel olmayan yolu seçerek, yüreksiz bir programcının geleneksel yoldan ayrılma cesaretini gösterememesi sonucu geride kalması gibi bir durum yaratmaktan kaçınmak suretiyle hevesi kamçulamak ümidindeyim. Aynı zamanda öğrenciye, bilgisayar donanımına ilişkin derin bir bakış açısı kazandırmaya gayret ettim. Bunu da, tartışmasız kabul edilen ve bir doğma olarak öğrenilecek bir bilgiyi ona verme yerine, nesnelerin durumlarının niçinini açıklayarak yaptım. Bunun da ötesinde giriş düzeyinin üzerinde sayılabilecek birkaç konuyu da kitaba ekledim. Bu konular, örneğin ileri

bilgisayar aritmetiği, zamanlama diyagramları ve güvenilirlik, gerçek dünyadaki bilgisayarların temel kurslarda anlatılanlardan farklı olabileceğini göstermek üzere buraya alınmışlardır.

Sonuç olarak, bir teoriyi veya formülü olduğu gibi vermenin yeterli olmadığını, birçok öğrencinin formülün gerçek uygulama alanını görmek istediğini keşfettim. Sonuç olarak mümkün olan yerlerde çözülmüş örnekler verdim ve buna ek olarak, normalde ilk düzey kurslarında yer alabilen konuların sınırlarını genişlettim. Bilgisayar donanımı ile dolaylı ilişkisi nedeniyle işletim sistemleri ve yerel iletişim ağlarına ilişkin olarak tanıtım bilgileri sundum.

Bilgisayar mimarisini konu alan bir çok giriş kitabında olduğu gibi ben de model olarak belli bir mikroişlemciyi seçtim. Bilgisayar yapısının temel ilkelerinin öğretiminde mikroişlemciler kullanmanın avantajı, nisbeten basit yapıda olmasıdır. Birçok ana ve mini bilgisayarlar; daha kursun başlangıcından öğrenciyi çok fazla ayrıntı ile karşı karşıya burakan karmaşık bir mimariye sahiptirler. Mikroişlemciler, anabilgisayarların hemen tüm önemli özelliklerine sahip olup daha az karmaşıktırlar. Bu kursta göstermek üzere, 8-bitlik 6502 mikroişlemcisini seçtim. Bunun nedeni düşük-maliyetli öğretim programları ve kişisel bilgisayarlarda bu mikroişlemci kullanılmaktadır. Belirtmeliyim ki bu kitap, 6502 üzerine uygulamalı assembly programlama kitabı olarak tasarlanmamış olup assembly programı aracılığı ile merkezi işlem biriminin çalışmasını incelemektedir. Günümüzün en gelişkin 8-bit mikrobilgisayarı olarak kabul edilen 6809 MİB'e ilişkin ayrıntılar, mikroişlemci tasarımında modern yapıtaşları olarak benimsenmiştir.

Bu kitap, bilgisayar bilimleri kursu veya elektrik mühendisliği bilgisayar teknolojisi öğrencilerine ilişkin ilk yıl kursu için hazırlanmış olmakla birlikte, Yüksek Ulusal BEC/TEC bilgisayar kursları için, BCS sınavı 1. aşamaya katılacaklara ve bilgisayarların nasıl çalıştığını merak eden kişilere de hitab etmektedir.

### Okuma Kılavuzu

Daha önce belirtildiği üzere bu kitap, bilgisayar donanım ve mimarisini konu alan geleneksel kurslara elverişli bir temel sağlamakla kalmayıp; konu içeriklerini genişletmekte ve bu kurslardaki bazı eksikleri tamamlamaktadır.

Giriş kursunu izleyen öğrencilerin ön ve arka plan malzemelerini ayırtetmekte güçlük çekebileceklerini düşünerek, okumaları için bir kılavuzu ekledim.

Bölüm 2: Kapılar, Boole cebri, Karnaugh diyagramlarının ele alındığı Bölüm:2'nin

başlangıç kısmı temel okuma konularıdır. Bununla birlikte iki kararlılar ve flip-flopları anlatan kısım 2.5, kapılar ve benzer mantık elemanlarına göre anlayabilme açısından bir kuantum sıçramasını gerektirmektedir.

Bu kitabın amacı içinde, flip-flop temel fikrinin ve bir bellek elemanı olarak kaydedicilerdeki uygulamasının anlaşılması için gereken herşey vardır.

Yani, bu kursun sonunda öğrencilerden, kapılardan oluşan orta derecede karmaşık devreleri tasarımıyabilmeleri devrelerin oynadığı rolü değerlendirebilmeleri beklenmektedir.

Ardışık devreler ise, sayısal tasarımın ikinci düzey kurslarında ele alınmaktadır. Kısım 2.7 ve 2.8, bilgisayar tasarımında yaşanan problemlerden söz etmekte olup, kitap ilk okunurken atlanabilir.

Bölüm 3: Bu bölümde sayıların bilgisayar içindeki gösterimleme biçimleriyle basit aritmetik işlemlerde nasıl kullanıldıklarına ilişkin yöntemler işlenmektedir. Kısım 3.3.9'daki Huffman kodları dışında bu konunun hemen hepsi, temelde okuma parçalarından oluşmuştur. Kayan noktalı sayıların gösterimi ve kullanılmasını birkaç öğrencinin kavramış olabilmesi nedeniyle kısım 3.7'de, kayan noktalı aritmetik üzerine diğer konulardan daha fazla ayrıntıya inilmiştir.

Bölüm 4: Bu bölüm kitabın kalbi olup bilgisayarın yapısı ve çalışmasını işlemektedir. Kısım 4.2 bilgisayar denetim biriminin çalışmasıyla ilgili olup, ilk okumada atlanabilir. Denetim birimine normal olarak ikinci ve üçüncü kurslarında rastlanmakla birlikte; burada bütünlüğü sağlamak ve bilgisayarın ikili kodlanmış komutları, komutu icra eden olaylar dizisine nasıl çevirdiğine ilişkin bir bakış açısını öğrenciye vermek amacıyla alınmışlardır. 6809 üzerine yazılı kısım 4.5 arka plan malzemesi sağlamakta olup, ilk okumada atlanabilir.

Bölüm 5: Bu bölüm Giriş/Çıkış teknikleri ve ilgili bilgisayar çevre birimlerini işlemektedir. Temel okuma bölümü olup atlanmamalıdır.

Bölüm 6: Bu bölümde bilginin bilgisayar içinde gerçekte nasıl saklandığı açıklanmaktadır. Öğrenci, doğrudan ulaşılan bellek ve saklama birimlerinin çalışmasına ilişkin ilkelere dikkat etmelidir. Bununla birlikte bazı ayrıntılı tanımlar (ince film belleği, kayıt kodları) verilmiş olup istenirse ilk okumada atlanabilir.

Bölüm 7: Öğrencinin özel ilgi alanına girmediği bu konuya derinliğine girmesi sürpriz olan bilgisayarların bölme ve çarpmayı yapma yolu tamamen atlanabilir.

# İÇİNDEKİLER

|                                                   |            |
|---------------------------------------------------|------------|
| <b>1 Giriş</b>                                    | <b>1</b>   |
| 1.1 Sayısal bilgisayar                            | 5          |
| 1.1.1 Veri işlemcisi olarak bilgisayar            | 5          |
| 1.1.2 Bir Hesap Makinesi olarak bilgisayar        | 7          |
| 1.1.3 Otomatik denetimde bilgisayar               | 8          |
| 1.1.4 Bir eleman olarak bilgisayar                | 9          |
| 1.2 Anabilgisayar, mini ve mikrobilgisayarlar     | 11         |
| 1.3 Saklı programlı bilgisayarlar-genel bir bakış | 12         |
| <b>2 Mantık elemanları ve Boole cebri</b>         | <b>16</b>  |
| 2.1 Temel kapılar                                 | 18         |
| 2.2 Bazı kapı uygulamaları                        | 25         |
| 2.3 Boole cebrine giriş                           | 33         |
| 2.3.1 Karnaugh diyagramları                       | 44         |
| 2.3.2 Özel amaçlı birleşik mantık elemanları      | 58         |
| 2.4 Üç durumlu mantık                             | 64         |
| 2.5 İki kararlılar ve kaydediciler                | 69         |
| 2.5.1 RS tipi flip-floplar                        | 70         |
| 2.5.2 D tipi flip-floplar                         | 76         |
| 2.5.3 Saatlenmiş flip-floplar                     | 78         |
| 2.5.4 JK tipi flip-floplar                        | 82         |
| 2.6 Ardışık devreler                              | 83         |
| 2.6.1 Özel amaçlı ardışık mantık elemanları       | 90         |
| 2.7 Mantıksal tasarımda bazı pratik düşünceler    | 96         |
| 2.8 Bilgisayarlar ve Güvenilirlik                 | 104        |
| Problemler                                        | 111        |
| <b>3 Bilgisayar aritmetiği</b>                    | <b>117</b> |
| 3.1 Karakterler, sözcükler ve baytlar             | 117        |
| 3.2 Sayı tabanları                                | 118        |
| 3.3 Sayı-tabanlarının birbirine dönüştürülmesi    | 121        |
| 3.3.1 Ondalıktan ikiliye                          | 121        |
| 3.3.2 Ondalıktan sekizliye                        | 122        |
| 3.3.3 Ondalıktan onaltılıya                       | 122        |
| 3.3.4 İkiliden ondalığa                           | 123        |

## İçindekiler

|                                                                     |            |
|---------------------------------------------------------------------|------------|
| 3.3.5 Sekizlikten ondalığa                                          | 123        |
| 3.3.6 Onaltılıktan ondalığa                                         | 124        |
| 3.3.7 İkili, sekizli ve onaltılı sistemler arasında dönüşümler      | 124        |
| 3.3.8 Kesirlerin dönüştürülmesi                                     | 125        |
| 3.3.9 BCD ve diğer kodlar                                           | 127        |
| 3.4 İkili aritmetik                                                 | 136        |
| 3.4.1 Yarım toplayıcı                                               | 137        |
| 3.4.2 Tam toplayıcı                                                 | 138        |
| 3.4.3 Sözcüklerin toplanması                                        | 140        |
| 3.5 Negatif sayılar                                                 | 143        |
| 3.5.1 İşaret ve değer gösterimi                                     | 144        |
| 3.5.2 Tümler aritmetik                                              | 145        |
| 3.5.3 İkiye Tümleyen Biçimde Gösterme                               | 146        |
| 3.5.4 Bire Tümleyen Biçimde Gösterme                                | 151        |
| 3.6 Bilgisayar aritmetiğine ve Assembly-dilinde programlamaya giriş | 152        |
| 3.7 Kayan noktalı sayılar                                           | 158        |
| 3.7.1 Kayan-noktalı sayıların gösterilmesi                          | 160        |
| 3.7.2 Kayan-noktalı sayıların normalize edilmesi                    | 162        |
| 3.7.3 Tipik Bazı Kayan-Noktalı Sistemler                            | 164        |
| 3.7.4 Kayan Noktalı Aritmetik                                       | 167        |
| 3.7.5 Bazı çözülmüş örnekler                                        | 171        |
| Problemler                                                          | 178        |
| <b>4 Merkezi İşlem Birimi</b>                                       | <b>185</b> |
| 4.1 MİB'nin yapısı                                                  | 186        |
| 4.1.1 Bir Programlama Örneği                                        | 193        |
| 4.2 Denetim birimi                                                  | 199        |
| 4.2.1 Mikroprogramlı denetim birimi                                 | 199        |
| 4.2.2 Rastgele mantık denetim birimleri                             | 205        |
| 4.3 6502 mikroişlemcisi                                             | 214        |
| 4.3.1 6502 komut takımına giriş                                     | 218        |
| 4.4 Adresleme modları                                               | 224        |
| 4.4.1 Dolaysız adresleme                                            | 225        |
| 4.4.2 İndekslenmiş adresleme                                        | 229        |
| 4.4.3 Bağlı adresleme                                               | 234        |
| 4.4.4 Dolaylı adresleme                                             | 236        |
| 4.4.5 Yığın                                                         | 239        |
| 4.5 6809 mikroişlemcisi                                             | 250        |
| 4.5.1 Adresleme modları ve 6809                                     | 252        |
| Problemler                                                          | 262        |

## İçindekiler

|                                       |     |
|---------------------------------------|-----|
| <b>5 Giriş/Çıkış</b>                  | 269 |
| 5.1 Programlı G/Ç                     | 273 |
| 5.2 Kesme ile sürülen G/Ç             | 276 |
| 5.3 Doğrudan bellek erişimi           | 280 |
| 5.4 Kanal G/Ç                         | 282 |
| 5.5 G/Ç birimleri                     | 284 |
| 5.5.1 VDU (Görsel Görüntüleme Birimi) | 284 |
| 5.5.2 Yazıcı                          | 295 |
| 5.5.3 Kullanıcı-yönelimli arabirimler | 302 |
| <b>6 Bilgisayar belleği</b>           | 309 |
| 6.1 Yarıiletken bellekler             | 313 |
| 6.2 Ferrit çekirdekli bellekler       | 326 |
| 6.3 İnce-film türü bellek             | 333 |
| 6.4 Bellek hiyerarşisi                | 338 |
| 6.4.1 Manyetik kayıt                  | 340 |
| 6.4.2 Disk sürücüler                  | 349 |
| 6.6.3 Teyp birimleri                  | 360 |
| 6.5 Manyetik kabarcıklı bellekler     | 369 |
| Problemler                            | 373 |
| <b>7 İleri bilgisayar aritmetiği</b>  | 377 |
| 7.1 Çarpma                            | 377 |
| 7.1.1 İşaretsiz ikili çarpma          | 378 |
| 7.1.2 İşaretlı çarpma                 | 381 |
| 7.1.3 Yüksek-hızda çarpma             | 384 |
| 7.2 Bölme                             | 388 |
| 7.2.1 Geri-getirmeli bölme            | 391 |
| 7.2.2 Geri getirmesiz bölme           | 391 |
| 7.2.3 Çarpma yaparak bölme            | 392 |
| <b>8 Bilgisayar iletişimi</b>         | 397 |
| 8.1 Seri veri iletimi                 | 399 |
| 8.1.1 Asenkron seri iletim            | 400 |
| 8.1.2 Senkron seri iletim             | 401 |
| 8.2 Protokoller                       | 411 |
| 8.3 Yerel iletişim ağları             | 419 |

## İçindekiler

|                             |     |
|-----------------------------|-----|
| <b>9 İşletim sistemleri</b> | 431 |
| 9.1 Sistem yazılımı         | 435 |
| 9.2 Çoklu programlama       | 440 |
| 9.3 Bellek yönetimi         | 444 |
| <b>Ekler</b>                | 461 |
| A1 Bazı çözülmüş örnekler   | 461 |
| A2 6502 komut takımı        | 479 |
| <b>Kaynakça</b>             | 494 |
| <b>İndeks</b>               | 497 |

Öncelikle donanım ve yazılım terimlerini tanımlayarak başlanmayı bir zorunluluk olarak hissettim. Tamamen soyut bir tanım yerine deneysel bir tanım vermek, konunun anlaşılması açısından çok daha yararlı olacaktır. Eğer bilgisayar sisteminin herhangi parçası yere düştüğünde ses çıkarıyorsa donanım, değilse yazılımdır.

Donanım, bilgisayarı oluşturan tüm fiziksel bileşenleri içerir. Bu, merkezi işlem biriminden belleğe ve giriş/çıkış aygıtlarına kadar uzanır (çeşitli bileşenleri içermektedir). Bilgisayarın çalışmasını denetleyen programlar ise yazılım adını alır. Bir bilgisayarda bulunan programların fiziksel varlığı, bilgisayarın içindeki elektronik anahtarların durumu ya da disk veya manyetik teypteki küçük parçacıkların manyetizasyonu ile anlaşılır.

Yazılımla donanım arasında sık sık bir alışveriş durumu söz konusudur. İşlemler, özel amaçlı donanım sistemleriyle ya da bellekte saklı bulunan yazılımlarla gerçekleştirilir. Genelde bir işi gerçekleştirmenin en kolay yolu, o işe özel devre yapmaktır. Fakat bu pahalı bir çözümdür. Mevcut bilgisayarı kullanarak sözkonusu işi gerçekleştirmek için program yazılması, daha ucuz olacaktır, ancak uygulama uzunca bir süre alabilir. Burada alışveriş terimi hızı parayla değiştirmek anlamında kullanılmaktadır. Benzer bir alışveriş insan ilişkilerinde de vardır. Örneğin, bir fabrika, hırsıza karşı korunmak istensin. Olası bir çözüm elektronik algılayıcılar yerleştirmek ve her hareketi rapor etmek suretiyle donanımsal bir çözüm üretmektir. Diğer bir çözüm ise bir insanı gece bekçisi olarak programlayıp yazılıma dayalı bir çözüm getirmektir. Burada yazılım, genel-amaçlı makinanın (burada insan) beynindeki programdır.

Bilgisayar bilimcileri tarafından insan beyni sık sık "etten bilgisayar" olarak adlandırılır.

Neden bilgisayar donanım ilkelerini öğretiyoruz? UHF alıcısının nasıl çalıştığını anlamadan televizyon izleyebilir, termodinamik sözcüğünün ne anlama geldiğini dahi bilmeden bir jumbo jette uçabiliriz. Ve neden bir bilgisayarın içinde ne olup bittiğini öğrenmeye zorlayarak, bilgisayar bilimcilerinin ve yazılımcılarının hayatlarını çekilmez hale getirelim.

Bilgisayar donanımının temellerini öğretmek için çeşitli nedenler vardır. En önemlileri şunlardır;

1. Programlama, dolaylı bir biçimde de olsa donanım ile sıkı sıkıya bağlıdır. Programcı veri girişi yapar veya çıkış alırken, önceden bilgi sahibi olması gereken bir donanım aygıtını kullanıyor olacaktır. Genellikle bilinmesi gerekenler, (satır başına sütun ve sayfa başına satır sayısı gibi) verilerin formatına ilişkin bilgilerdir. Örneğin verilerin, televizyon tipi bir görüntüleme biriminde (VDU) görüntülenmesi durumunda, bir ekran görüntüsünden, herhangi bir cümleyi orta yerinden bölerek diğerine geçmek, dikkatsiz bir okuyucunun cümlelerin ilk yarısını hatırlamak zorunda kalmasına neden olacağından, ekran görüntüsünün belirli mantıksal noktalarda bölümlere veya sayfalara ayrılması çok daha akılcı bir çözüm olacaktır.

2. Donanım, bilgisayarın sınırlarını belirler. Herhangi bir bilgisayar kullanıcısı, kullandığı bilgisayarın sınırlarının farkında olması gerekir. Örneğin, 80 öğrencinin 80 terminalden eşzamanlı olarak erişim yapması istenen bir projede yalnızca 40 terminale hizmet verebilen bir bilgisayar sistemi satın almanın hiç bir anlamı olmayacaktır. Benzer biçimde, bir uçak üzerindeki seyrüsefer bilgisayarının, rota düzeltmelerini, uçak rotadan çıkmadan yapabilecek hızda olması gerekir.

3. Programlama, ancak belli bir yere kadar donanımdan bağımsız düşünülebilir. Yerel bir yönetim ofisinde çalışanların maaşlarını hesaplayan COBOL dilinde bir program yazan bir programcı, donanımın ayrıntılarından oldukça uzak olmasına karşın, diğer programcılar sık sık sistemle bilgisayar arasında bir arabirimi kullanmak durumundadır. Örneğin bir mikroişlemci, bir kimyasal reaksiyonun sıcaklığını denetlemek üzere kullanılabilir. Böyle bir sistemde sıcaklık önce gerilime dönüştürülür, sayısal veriler durumuna getirilip bilgisayara okunur, bu değeri işleyen bilgisayar başka bir sayısal değer üretir, bu değer de gerilime dönüştürülerek ısıtıcıyı denetlemede kullanılır. Olayların bu sırada gerçekleşmesi, elektrik mühendisleri ve yazılım uzmanlarına gerek duyar. İşin başarılabilmesi, iki tarafın birbirleriyle iletişim kurabilmesi gerektirmektedir. Hatta bu iki taraf aynı kişi, yani yazılımcı bir elektrik mühendisi veya tersi de olabilir.

4. Donanımı anlamak, estetik açıdan zevklidir. Bu kısmın başında verilen örnekler tamamen yerinde örnekler değildir. Yolcu, jet motorunun nasıl çalıştığından haberdar değildir ancak pilotun, bu konuyla ilgilenmesi normaldir. Motorun ayrıntılarını bilmek, işini yapması için zorunlu olmamakla birlikte anlamak, pilota bir ölçüde tatmin duygusu verir. Benzer biçimde, programın bilgisayara yüklenmesinin ardından neler olup bittiğini ayrıntılarıyla bilen bir programcı, bilgisayarın iç çalışmasını bir sihirli-kutu gibi algılayan birine göre çok daha kişisel tatmin sahibidir.

Bir bilgisayar donanım kursunun neleri içereceği ya da hangi konuları dışarıda bırakacağını tam olarak bilinmesi oldukça zordur. Herhangi bir konu istenildiği

ölçüde genişletilebilir. Eğer mantık kapıları ve Boole cebri ile başlarsak, yarı iletken aygıtlarla ve ardından da yarı iletken fiziği ile mi devam edeceğiz? Bu kurs içinde, yukarıda (1)'den (4)'e kadar konulara, giriş düzeyinde yer verdim. Bununla birlikte, kullandığım bazı öğretim araçları geleneksel giriş metinlerine aşina olanlar için şaşırtıcı olabilir. Birçok değişik alandan eğitim malzemesi kullanmanın nedeni bilgisayarların etki alanının son yıllarda geniş ölçüde artmış olmasıdır.

Teori ve uygulama arasındaki farkı vurgulayabilmek için normal yöntemden bir parça ayrıldım. Örneğin kapılar ve Boole cebrine 'genel' bir giriş yaparken, tasarımcının, mantıksal tasarımın yanında diğer (ekonomik) faktörlerle de ilgilenme durumunda olduğuna açıklık getirdim. Bu kitap kapsamında ele alınacak ana konular aşağıda verilmiştir:

*Bilgisayar aritmetiği* On tabanına dayalı aritmetik sistemimiz binlerce yıldır gelişerek günümüze gelmiştir. Bilgisayar ise kendi iç işlemlerini ikili tabanda oluşturduğu sayılarla yapmaktadır. Bu farklılık, ikili sistemin büyüleyici gücünden çok onluk tabana dayalı bilgisayar tasarımının ekonomik olmayışından kaynaklanmaktadır.

*Temel mantık elemanları ve Boole cebri* Bugün sahip olduğumuz teknoloji, bir bilgisayarın ne yapabileceğini belirlemektedir. Bilgisayarın yapıldığı basit mantıksal elemanlardan yani kapılardan başlayarak, bu kapıların çalışmasıyla aritmetik işlemlerin nasıl gerçekleştirilebildiği, bir programın çalışması için bilgisayarın işlevsel parçalarının birbirleriyle nasıl karşılıklı etkileşimde bulunduğunu anlayabiliriz. Ayrıca, flip-flop'lar ve bunların ardışık devrelerde uygulanmalarına bir giriş de yapılmıştır. Flip-flop'lar, tek bir ikili basamağı saklayabilen (hatırlayabilen) mantıksal eleman olup, birçok bellek biriminin temelini oluşturmaktadır.

*Bilgisayar mimarisi* Bu konu, bilgisayarın kavramsal (blok diyagramları) düzeyde nasıl çalıştığına ilişkindir. Yani burada bilgisayarın bir komutu bellekten nasıl aldığı, nasıl çözdüğü ve nasıl işleme soktuğu gösterilmektedir. Bir bilgisayarın, işlevsel birimleri açısından yapısı onun mimarisi olarak adlandırılır.

*Assembly dili* Bilgisayarın çalışmasını doğrudan denetleyen ilkel komutlar makine-kodu komutları olup, bellekte saklı bulunan ardışık ikili sayılardan oluşmuştur. Makine dilinde programlama oldukça sıkıcı olduğundan makine-kodunda programlamaya yardımcı olmak üzere assembly dili geliştirilmiştir. Assembly dili, programcının makine-kodlu komutları, basit komut kısaltmaları ve kısaltılmış sade bir dil formunda yazabilmesine olanak sağlayan bir kısa yazım biçimidir. Cobol, Pascal, Fortran gibi yüksek düzeyli dillerde, bir derleyici aracılığıyla bir dizi

assembly dili komutuna çevrilmiş deyimler bulunmaktadır. Yüksek düzeyli dil programcılarının tersine, assembly dilinde program yazan bir programcı, kullandığı bilgisayarın iç mimarisini ayrıntılı olarak bilmek zorunludur.

**Giriş/Çıkış** Bir bilgisayarın yeni bilgileri (program ve verileri) alabilme ve hesaplanan sonuçları dış dünyaya verebilme olanağı yoksa, bu işe yarar bir durum değildir. Bu bölümde bilgilerin bilgisayara nasıl girilebileceğini ve çıktılarının nasıl alınacağını göstereceğiz. Üç temel Giriş/Çıkış birimi tanımlanmış olup bunlar klavye, VDU (ekran) ve yazıcıdır.

**Bellek birimleri** Bir bilgisayar, ileride gerek duyacağı programları, verileri ve diğer bilgileri tutabilmek için bir belleğe ihtiyaç duyar. Bu bölümde, "ani erişimli bellekler" ile "saklama birimlerini" inceleyeceğiz. Temelde, saklama birimlerinin çok daha yavaş olmalarına karşın, ani erişimli bellekler, bilgisayarın assembly dili düzeyi işlemlerini yürütebilmesi için gerekli olan verileri çok kısa sürelerde sağlayabilirler. Saklama birimleri, ani erişimli belleklerden çok daha düşük fiyatlıdır. Saklama birimlerinin en bilinen şekilleri disk ve teyp birimleridir. Bunlar, bilgileri kaydetmek için hareketli manyetik malzemenin sıkıştırılması ilkesine göre çalışırlar.

**İleri bilgisayar aritmetiği** Bu bölümde, sayısal bir bilgisayarda çarpma ve bölmenin hangi yöntemlerle yapıldığı anlatılmaktadır. En ileri düzeydeki bölümlerden biri olup, istenirse ilk okumada atlanabilir.

**Bilgisayar iletişimi** Bazı insanlar, mikroişlemcilerin ortaya çıkışıyla büyük bilgisayar sistemlerinin sonunun geldiğini düşünmüşlerdi. Bunun tersine gerçekleştiği birçok biçimde kanıtlanmış ve günümüzde bilgisayarlar, birbirlerine bir iletişim ağı biçiminde bağlanmasıyla daha da büyüklerine doğru büyümüşlerdir. Bu türde yapılanma, ağ içinde yer alan çok sayıda bilgisayarın, satır-yazıcı gibi pahalı çevre birimlerini ortaklaşa kullanabilmeleri gibi hiç de azımsanmayacak birçok üstünlükler sağlamaktadır. Buna ilişkin olarak, bilgisayarların birbirleriyle nasıl iletişim kurduklarının açıklandığı bir altbölüm eklenmiştir. Bu bölümde, bilgisayar iletişiminin üç değişik yönü ele alınmaktadır. Birinci kısımda, bilgisayar içindeki sayısal bilginin, seri port üzerinden iletişimine olanak sağlayacak biçimde kodlanması ve bunun alıcıda tekrar oluşturulması incelenmektedir. İkinci kısımda, bilgisayarların birbirleriyle bir düzen içinde konuşmalarını sağlayan protokol veya kurallardan söz edilmektedir. Üçüncü kısımda ise yerel iletişim ağları kısaca gözden geçirilmiştir. Bunlar, bir ofis veya bina kompleksleri içinde dağılmış bilgisayarların oluşturduğu iletişim ağlarıdır.

**İşletim Sistemleri** İşletim sistemi, bilgisayarın tüm işlevsel birimlerini koordine eden çok sayıda küçük yazılımlardan oluşmuştur. Donanım üzerine yazılmış bu kitapta işletim sistemlerinden söz etmemizin başlıca nedeni, birçok bilgisayarda bulunan iki donanım (Kesme-yönetim mekanizması ve yığın) özelliğinin, birçok işletim sisteminin önemli bir karakteristiği olan çoklu programlamayı kolaylaştırıcı nitelikte olmasıdır. Çoklu programlama, bilgisayarın iki veya daha fazla programı görünüşte aynı anda çalıştırabilmesidir. Bu kısmın sonunda, bellek yönetiminin kısa bir tanıtımı yapılmıştır. Bellek yönetimi; küçük, yüksek-hızlı rastgele erişimli bir bellekle birlikte, büyük, düşük-hızlı, seri erişimli belleğe sahip bir bilgisayarı, tek parça, büyük, yüksek hızlı rastgele erişimli bir belleğe sahipmiş gibi gösterebilmeyi sağlayan bir tekniktir.

## 1.1 Sayısal Bilgisayar

Bilgisayar donanım kavramını tartışmaya başlamadan önce, bir bilgisayarın ne olduğunu (veya ne olmadığını) açıklamak ve bazı tanımlar yapmak gerekir.

Eğer, bilgisayar bilimleri alanında, yanlış bilgilendirme suçları için bir ödül verilsediydi, bu ödül, "2001'de HAL" ve K9'un yaratıcılarına verilir. Bu ve benzer makinalar öyle bir efsane yarattılar ki, bilgisayarlar, belleklerinde sonsuz sayıda verilerin bulunduğu ve tüm bilgileri içine alabilecek bir araç olarak algılanmaya başlandı. Gerçekte durum bir parça sıradandır: bilgisayarlar, bilgileri dış dünyadan alıp, önceden belirlenmiş bir dizi işlemlerden geçirdikten sonra tekrar dış dünyaya işlenmiş olarak çıkaran makinalardır. Bu tanım, "bilgisayar" sözcüğünü, onun kadar karmaşık olan bilgi, işlem, süreç gibi terimlerle tanımlamaya çalıştığından oldukça yetersiz bir tanımlamadır. Bilgisayarın ne yapabileceği konusunda aşağıdaki örnekler, belki daha iyi bir yaklaşım olacaktır.

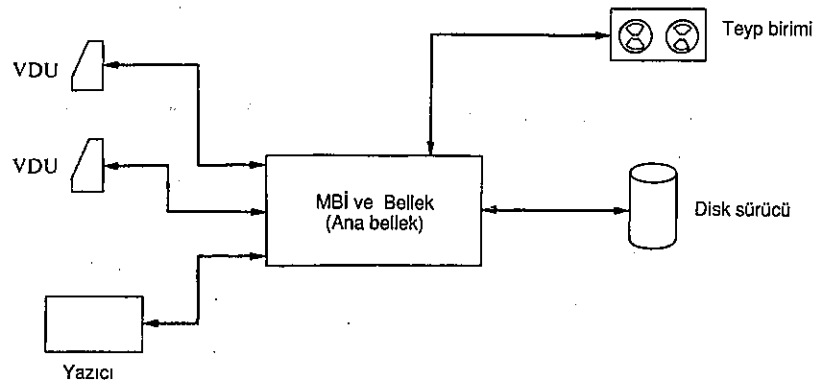
### 1.1.1 Veri işlemcisi olarak bilgisayar

Şekil 1.1'de, geleneksel bir bilgisayar tanımı gösterilmektedir. Buradaki, büyük bir fabrikanın, bordro işlemlerini yürütmek üzere tasarlanmış bir sistemdir. Yalnızca MİB (Merkezi İşlem Birimi)'ni bilgisayar, diğer tüm aygıtları da çevre birimi olarak tanımlayanların tersine ben, tüm sistemi bilgisayar olarak adlandıracam. MİB'nin içindeki hemen erişilebilir bellekte yüklü durumda bir program yer almaktadır. Bu program, bir grup ilkel makine-kodlu işlemlerden oluşmuş olup görevi; çalışanların ücretlerini, çalışılan saat, birim ücret ve fazla çalışma ücreti bazında hesaplanmaktadır. Kuşkusuz bu program, vergi, ulusal sigorta ve diğer kesintiler

gibi diğer ayrıntıları da içermektedir. Yazıcının görevi ise doğrudan adı ile açıklanabilmektedir. Yazıcı bir elektromekanik aygıt olup, mektupları, çok sayıda kağıdı yazar, bordroları doğrudan basıp çıkarabilir. Bilgisayarlardaki hemen erişilebilir bellekler diğerlerine göre daha pahalı olduklarından, sadece o anda işlenmekte olan veri ve programları tutabilecek büyüklükte olması tercih edilir. Çalışanlara ilişkin bilgiler, normal olarak saklama birimlerinde birer "disk-dosyası" olarak tutulurlar. Bilgisayar, belirli bir çalışana ilişkin bir bilgiye gerek duyduğunda, bu bilgi diskten alınarak (daha doğru bir tanımla kopyalanarak) hemen erişilebilir belleğe yerleştirilir. Bu işlemi yürütmek için gereken süre, saniyenin küçük bir kesridir ancak bu, okuma işleminden defalarca yavaştır. Ne var ki, bilgileri diskin üzerinde saklamanın maliyeti çok düşüktür ve bu da bağlı yavaşlığı dengeler.

Teyp birimi, diskten çok daha ucuz bir saklama birimidir. Birçok kuruluştaki disk üzerindeki veri periyodik olarak (örneğin her dört saatte bir) teypler üzerine kopyalanır ve bunlar, güvenlik gerekçesiyle giriş katında saklanırlar. Herhangi bir nedenle disk bozulduğu (disk crash) veya diskteki bilgiler kaybolduğu zaman, arızadan önce en son alınmış 'teyp kopyası' sisteme tekrar yüklenerek güncel bilgilere yeniden erişilmiş olur. Burada "crash" (= çarpma) sözcüğü, disk kafasının kaza sonucu, dönen disk yüzeylerine çarpması ve bu çarpmadan dolayı, yüzeyin manyetik kaplamasının bozulması anlamında kullanılmıştır. Pratikte "crash" herhangi bir sistem arızası anlamına gelmektedir.

Ekran birimi (VDU), operatörlerin veriyi doğrudan sisteme girebilmelerini sağlar. Eğer bu işlem için insanlar görevlendirilseydi, bir çalışanın o hafta içinde çok uzun saatler boyunca çalışması gerekebilecekti. Bunun tersine, bir VDU, "Bay XYZ



Şekil 1.1 Veri işlemcisi olarak bilgisayar

Kasım ayında ne kadar vergi ödedi?" gibi belirli soruların sordurulmasında kullanılabilir. Daha doğru ifade edilirse, VDU'nun gerçekte soru sormadığı fakat yazılımcıya, ilgili soruyu içeren yazılımı çalıştırma olanağı sağladığını söyleyebiliriz. VDU, programın kendisi üzerinde de değişiklik yapılabilmesini sağlayarak, sistem büyüdükçe yeni olanakların eklenebilmesine izin verir.

### 1.1.2 Bir Hesap Makinesi olarak bilgisayar

Yukarıdaki örnekte verilen bilgisayar, zamanının çoğunu, firmada çalışanlara ilişkin bilgileri temsil eden semboller biçimindeki verileri işlemeye ayırmaktaydı. Böyle bir bilgisayarlı uygulamada, bilgisayarın sayılarla, aritmetik (bir matematikçinin anladığı anlamdaki aritmetik) işlemleri yapması için geçen süre, verileri işlemekten çok daha kısadır.

Bilgisayar uygulamalarına sayı kırma (number crunching) denmesinin nedeni, bilgisayarın, bir iş için çok fazla sayıda -genellikle milyarlarca- matematik işlemi yapıyor olmasıdır.

Sayı kırmanın birçok uygulaması olup, birçoğu "bilimsel" olarak tanımlanır. Örneğin, atmosferik olaylara ilişkin işlem sürecini modelleyen bir bilgisayar uygulamasını dikkate alalım. Atmosfer, süreklilik gösteren, üç boyutlu ve çok değişik sayıda gaz moleküllerinden oluşmuş bir ortamdır. Bilim adamları, süreklilik gösteren ortamları basit ve kolay biçimde ele alamaz, fakat atmosferin çok sayıda küçük küpten oluştuğu varsayımıyla problem, üzerinde işlem yapılabilir bir forma sokulabilir. Bu küplerden herbirinin, her noktada eşit sıcaklığa, yoğunluğa ve basınca sahip olduğu varsayılır. Yani küpü oluşturan gaz kütlesi, fiziksel özellikleri bakımından herhangi bir değişim göstermez. Küplerin altı yüzü vardır ve bilim adamları bu yüzlerin, komşu diğer yüzeylerle olan etkileşimini inceleyen modeller yaratabilirler.

Bilim adamları, tüm küplerin özdeş olduklarını (başlangıçta kendi aralarında herhangi bir etkileşimin olmadığını) varsayarak başlayabilirler ve sonra bir enerji kaynağının, güneşin, bu modele uygulanmasıyla neler olacağını ele alırlar. Herbir küpün, komşu küpe olan etkisi hesaplanır ve tüm işlem yinelenerek tamamlanır. Kesin sonuçları elde etmek için, küplerin boyutları küçük olmalıdır, aksi halde küplerin tümünün aynı şekilde olduğu varsayımı geçerli olmayacaktır. Bundan başka, kararlı durumdaki değeri elde edebilmek için gerekli olan yinelemelerin sayısı oldukça büyüktür. Sonuç olarak bu tip bir problem olağanüstü güçlü bir bilgisayarla ya da kimilerinin adlandırdığı gibi bir süper bilgisayarla çok uzun süre çalışmayı gerektirir.

Sayı kırmanın yapıldığı bir diğer alan da bilgisayarların gerçek-zamanlı uygulamalarıdır. Buradaki gerçek-zaman kavramı, hesap sonuçlarının bilgisayarın çalışmaya başlamasından itibaren verilen bir süre içinde elde edilebilmesi gerekliliğidir. Örneğin uçuş trafik denetimindeki bir bilgisayarlı uygulamayı dikkate alalım. Döner radar anteni her uçağın (hedefin) yükseklik ve uzaklığını (menzili) ölçer.  $t$  zamanında,  $p(i,t)$  konumundaki  $i$  hedefi; geriye,  $r(i,t)$  olarak menzili ve  $b(i,t)$  olarak yönü belirten bir yansıma sinyali gönderir. Maalesef radar sinyalinin doğası gereği, hedeften yansıyan yansıma sinyallerine rastgele bir hata eklendiğinden ve bu etki de hesaba katılmak zorundadır.

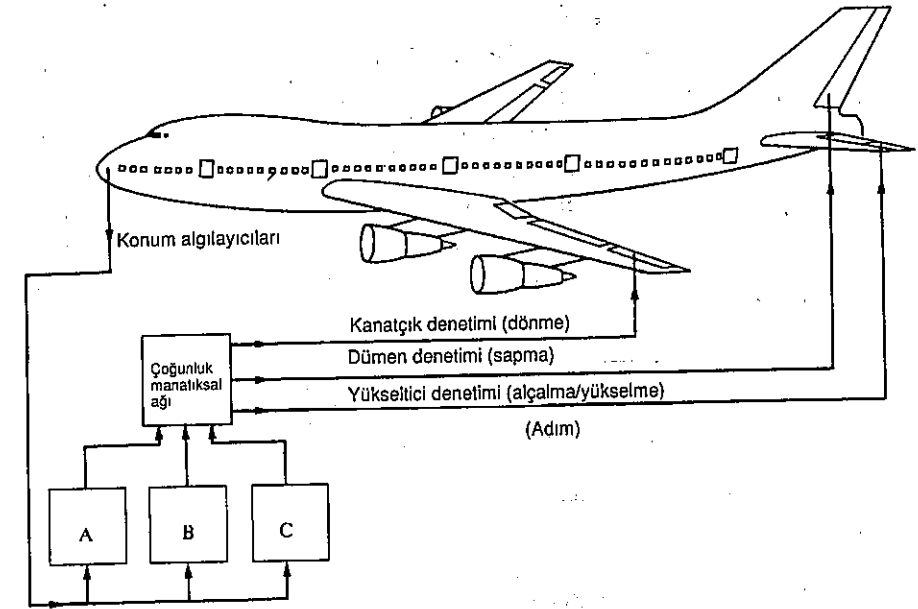
$N$  sayıda hedef için radar alıcısından gelen veriler bilgisayara girilir ve her dakikada  $p$  kez yenilenir. Bu ham bilgiden bilgisayar, her uçağın konumunu, rotasını hesaplayarak uçuş trafik denetleyicisini olası çarpışmalara karşı uyarır. Bütün bunlar önemli ölçüde yüksek-hızlı sayısal hesaplamayı gerektirmektedir.

### 1.1.3 Otomatik denetimde bilgisayar

Bilgisayarların büyük çoğunluğu, yukarıda sözü edilen iki tipten daha çok, otomatik denetim alanında kullanılmaktadırlar. Denetim elemanı olarak bilgisayar, daha büyük sistemler içine yerleştirilmiş olup dışarıdan bakan için 'görünmez' durumdadır. Görünmez derken söylemek istediğim, operatörün (veya dışarıdan izleyenlerin) bilgisayarın varlığının farkında olamayabileceğidir. Örneğin bir benzin pompasına monte edilmiş bir bilgisayar, aldığı parayı sayar, denetler ve uygun miktarda benzini verir. Kullanıcı, pompa doğru çalıştığı sürece, onun en son mikroişlemci ya da hassas bir mekanizma tarafından mı denetlendiği ile ilgilenmez.

Bilgisayarların otomatik kontrol alanında kullanımına güzel bir örnek, Şekil 1.2'de gösterildiği gibi uçakların otomatik iniş sistemleridir. Bu örnekte uçağın konumu (yükseklik, konma noktasına uzaklık ve pist orta çizgisinden uzaklık) ve hızı, yer indirme sistemiyle bağlantılı olarak çeşitli radyo teknikleriyle belirlenir. Bu bilgiler, herbiri uçak rotasındaki hataları belirleyen üç ayrı bilgisayara iletilir. Burada hata, uçağın ölçüm sonucu bulunan konumuyla, bulunması gereken konum arasındaki farktır. Bilgisayardan alınan sonuçlar, uçak denetim yüzeylerini (kanatçık, yükseltici, dümen) harekete geçirmek için gerekli sinyallerdir ve eğer gerekirse motor gücünü de ayarlar.

Bu durumda bilgisayar programı, bir çeşit ani erişimli bellek olan okunabilir ancak üzerine yazılamaz özellikteki 'sadece-okunur bellekte' (ROM) tutulur. Program, uçağı indirmeyi bir kez başardığında artık üzerinde değişiklik yapılması



Şekil 1.2 Denetim elemanı olarak bilgisayar

gerekmeyecektir. Burada aynı problem üzerine aynı giriş bilgileriyle çalışmakta olan üç ayrı bilgisayar olduğuna dikkat ediniz. Bilgisayarların çıkışları çoğunluk-mantık devresine iletilir. Bu devrenin üç girişi de aynı ise çıkış girişlere özdeşdir. Eğer bir bilgisayarlardan biri arızalanırsa, devre, bu bilgisayarları ilişkin girişi, çalışan diğer iki bilgisayara ilişkin girişe eşitleyerek yüksek sistem güvenilirliği sağlar.

### 1.1.4 Bir eleman olarak bilgisayar

Bu örnek, gerçekte bilgisayarın otomatik indirme sistemlerindeki kullanımının diğer bir uygulamasıdır. Mikroişlemciler ilk ortaya çıktıklarında, sosyal etkilerinden biri de, üniversitelerin ve politeknik okullarının bilgisayar bilimleri bölümleri ile, elektrik mühendisliği bölümleri arasında sürtüşme yaratması oldu. Her iki bölüm de mikroişlemcinin kendilerine ait olduğunu ileri sürdüler. Bir bilgisayar bilimcisi mikroişlemciyi, düşük maliyetli bir minibilgisayardan biraz daha fazla, fakat temelde aynı özelliklere sahip bir yapı olarak görüyor; buna karşın elektronik mühendisleri ise mikroişlemciyi, daha karmaşık ve gelişmiş ancak yine de yalnızca bir eleman olarak görmekteydi. Ne varki, mikroişlemci uygulamaların çoğunluğu gerçekten de onu bir eleman sınıfına sokmaktadır.

Örneğin, bir hi-fi kaset-çalarla ilgili mikroişlemci uygulamasını ele alalım. İlk bakışta analog sinyalleri işleyen bir sistemde, mikro işlemcinin yeri yokmuş gibi görünebilir. Mikroişlemciler şu anda, konuşma ve müzik sinyallerinin işlenmesinde rol almamaktadırlar fakat kaset-teyplerin denetimi üç yoldan kolaylaştırabilirler.

1983'den bu yana piyasada, sayısal saklama ve işleme sistemleri kullanan ses sistemleri görülmektedir, fakat bunlar sinyal işlemek için genel amaçlı mikroişlemciler kullanmamaktadır.

1. *Sistemin mekanik parçalarının denetimi.* Eski ve daha ucuz kaset-çalarlarda denetimler (kayıt, çalma, ara verme ve geriye sarma), kaset-çalar içindeki mekanik bağlantıları ve dişlileri doğrudan harekete geçiren büyük tuşlarla sağlanıyordu. Bugün kaliteli kaset-çalarlarda bu fonksiyonlar, elektronik olarak çalışan ve solenoid adı verilen anahtarlarla gerçekleştirilmektedir. Ön paneldeki anahtarların konumlarını bir mikroişlemci aracılığıyla okumak için ileri düzeyli bir denetim gerekir. Örneğin kaset-çalar hızlı ileri modundayken, kullanıcı hızlı geriye sarma tuşuna basarsa, mikroişlemci bandı önce yavaşlatarak durduracak ve sonra geriye saracaktır. Eğer bu işlem elle yapılıyorsa, en iyi olasılıkla teyp gerilecek ya da kopabilecekti.

2. *Bant sayacının denetimi.* Daha eski kaset çalarlarda bant konum göstergesi, basit olarak mekanik devirli bir sayaçtı ve tipik olarak 000 ile 999 aralığında olup, teyp hareket ettikçe ilerlerdi. Bu işlem mikroişlemci aracılığıyla gerçekleştirilirse, çok sayıda yeni olanaktan yararlanılabilir. Örneğin kullanıcı, kasedin bulunduğu konumdan, belirlenen diğer bir konuma kadar geriye sararak önceki bölümlerdeki parçaları çalmak isteyebilir. Bant sayıcısıyla, bant uzunluğu arasında bir ilişki bulunduğundan, bandın başlangıcından itibaren geçen veya bandın sonuna olan süreyi göstermek mümkün olacaktır.

3. *Dengeleme sistemlerinin ve ön gerilimlerin ayarlanması.* Tüm ses kayıt sistemleri, bu sistemlerle ilişkili olan iki parametreye, öngerilimleme ve dengeleme parametrelerine sahiptir. Hemen hemen tüm kaset-deklerde bulunan bir anahtar, üç temel tip banda (ferro, krom ve metal) karşılık gelen üç çift ön gerilim ve dengeleme değerlerinden birini seçer. Daha gelişmiş bazı deklerde bir mikroişlemci, bandın baş kısmına bir ses dizisi kaydederek öngerilimi ve dengelemeyi bu banda özgü biçimde ayarlar.

## 1.2. Ana bilgisayar, mini ve mikrobilgisayarlar

Yukarıdaki dört örnek, 'anabilgisayar', 'minibilgisayar' ve 'mikroişlemci' arasında ayrım yapabilmek için yeterlidir. Başlangıçta yalnızca anabilgisayar vardı. Bu, fiziksel olarak büyük, inanılmaz ölçüde pahalı bir canavardı ve çevresinde, yazılımcılar, operatörler ve bakım mühendislerden oluşan bir hizmet ordusu bulunuyordu. Böyle bir dev ancak büyük kuruluşlarda bulunabiliyor ve genel amaçlı bir makina olarak kullanılıyordu. Endüstride anabilgisayarlar, bordro hesabı gibi sıradan işler yanında ürün tasarımında ve davranışlarının simülasyonunda da kullanılmaktaydı. Belki bir anabilgisayarın en dikkate değer özelliklerinden biri de, geniş bir çeşitliliğe sahip olan çevre birimleridir - VDU'lar, yazıcılar, delikli kart okuyucular, delgi makinaları, disk ve teyp sürücüler.

Bilgisayarın gücüne, yalnızca zenginler ve güçlüler ihtiyaç duymazlar. Küçük laboratuvarlardaki bilim adamları, bilimsel veri analizlerini hızlandırmak isterler ve örneğin mühendisler, suyun akış hızını ölçmek ve böylece su düzeyinin gelecekteki durumunu tahmin etmek suretiyle barajları denetim altında tutmak isterler. İki durumda da, anabilgisayar kullanmak hiç de ekonomik olmayacağından, bilgisayar üreticileri bu tür problemlere minibilgisayarlar adı altında kullanışlı bir çözüm getirdiler. Minibilgisayarın kesin bir tanımı olmamakla birlikte aşağıdaki özellikler onu tanımlamakta kullanılabilir.

*Fiyat* Minibilgisayar bir anabilgisayardan çok daha ucuzdur. Tipik olarak 5.000 dolar ile 50.000 dolar arasındadırlar.

*Fiziksel boyut* Anabilgisayar genişçe sayılabilecek bir arsa büyüklüğünde yer kaplar. Genellikle bir binanın tüm bir katını kaplarlar. Diğer taraftan bir mini ise, gayet küçük olup, genellikle ortalama bir küçük dolaptan daha büyük değildir.

*Çevre birimleri* Tek bir anabilgisayarın hemen hemen kesinlikle özellikle üniversitelerde, politeknik okullarında veya araştırma kuruluşlarında olmak üzere geniş bir kullanıcı kitlesi olup, herbirinin kendine özgü istekleri vardır. Bu durum, çok sayıda çevrebiriminin kullanılmasını zorunlu kılar. Kullanıcıların program ve verilerini saklamak için disk sürücüler, VDU ve yazıcılar ve günümüzde de karmaşık diyagramların çizimi için grafik çizici aygıtlar gerekir.

Miniler ise çoğunlukla, (örneğin bir deneyin denetimi ve değerlendirilmesi gibi) tek bir işe adanmıştır ve genellikle az sayıda, belki tek bir disk sürücü, bir çift VDU ve yazıcı gibi çevre elemanı vardır. Şurası belirtilmelidir ki, bazı anabilgisayarların yerini aşamalı olarak minibilgisayarlar ya da bir iletişim ağı oluşturacak biçimde

minibilgisayar grupları almaktadır.

**Performans** En son minilerin hesaplama gücü, en son arabilgisayarın hesaplama gücünün küçük bir oranı kadardır. En son dememizin nedeni, bilgisayar teknolojisindeki gelişme öyle hızlı biçimde gerçekleşmektedir ki, bugünün düşük fiyatlı minileri, dünün anabilgisayarlarından çok daha güçlü duruma gelmektedir. Minibilgisayarların daha düşük hesaplama gücüne sahip olmasının nedenlerinden biri, kullandıkları sözcük-uzunluğunun anabilgisayarlara göre daha kısa olmasıdır. Bir bilgisayarın sözcük-uzunluğu, bilgi saklama ve işleme birimlerindeki bitlerin (birlerin ve sıfırların) sayısıdır. Bunun etkilerinden ileride söz edilecektir, ancak burada, kısa sözcük-uzunluğunun, belirli bir hesaplama işleminin verilen bir duyarlık düzeyinde gerçekleştirilebilmesi için çok daha fazla aritmetik işlem gerektirdiğini belirtmek yeterli olacaktır.

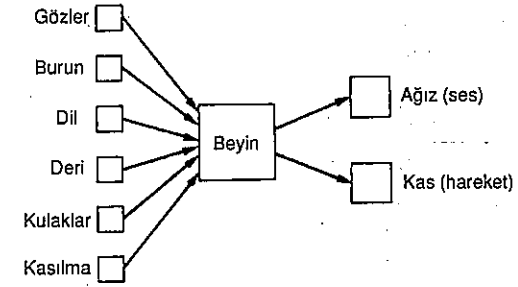
Aslında mikroişlemciler, tek bir silikon yongası üzerinde bulunan bir MİB'den başka birşey değildir. Mikroişlemcinin önemi onun şaşılacak ölçüde ucuz ve küçük oluşudur. Bir mikro işlemci tekbaşına hiçbir şey yapamaz. Bellek, güç kaynağı ve gerekli çevre birimleriyle birlikte bir mikrobilgisayar biçimine geleceği söylenebilir. Bugünkü teknolojik düzeyde mikrobilgisayarlar, mevcut minibilgisayarların altında bir performansa sahiptirler. Bir mikrobilgisayarın fiyatı 200 ile 5000 sterlin arasında değişmektedir (bazı eğitici oyuncaklar 100 sterlinin altında olup, basit mikroişlemci-tabanlı, denetleyicilerde bu fiyat sınırları içinde yer alabilir).

Kayda değer bir nokta olarak belirtmek gerekir ki, birkaç minibilgisayar üreticisi, tek başlarına veya diğer üretici firmalarla birlikte, minibilgisayarlarında kullandıkları MİB'ne işlevsel eşdeğerde mikroşlemciler ürettiriler. Bu yolla, mevcut minibilgisayar programları, (minilere oranla daha yavaş bir çalışan) bu mikrobilgisayarlar da çalıştırılabilir.

### 1.3 Saklı-programlı bilgisayar - genel bir bakış

Saklı programlı bilgisayarları tartışmaya başlamadan önce, insanı ele alalım. Bugünün harikası olan bilgisayarın insanla karşılaştırılması, tutucuların kendi teknolojileriyle karşılaştırmaları gibi doğaldır. Onlar, kendi teknolojilerini insana özgü terimlerle ifade etmek için 'bir vidası gevşek' veya 'buharı bitti' gibi deyimler icad etmişlerdir. Bilgisayarların insan beyniyle karşılaştırıldığı zamanlar olmuş, insan beyni karşısında yetersiz bulunduğu zamanlarda da zavallı bilgisayar, 'yüksek hızlı aptal' şeklinde adlandırılmıştır.

Şekil 1.3'de, bir insanın, girişleri, işlem birimi ve çıkışlarıyla birlikte nasıl bir sistem olarak ele alındığı gösterilmektedir. Girişler; görme (göz), koklama (burun) tatma (dil), dokunma (deri), ses (kulak) ve hareket ettirme (kasılma)'dır. Bu algılayıcılardan gelen bilgiler beyin tarafından işlenir.



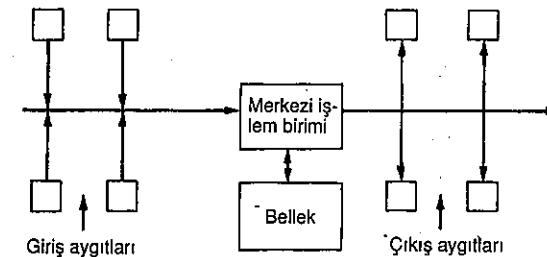
Şekil 1.3 Bir insanın organizasyonu

Beyin iki işlevi yerine getirir. Yeni bilgileri saklar ve duyu organlarından gelen giriş bilgilerini işler. Beynin saklama özelliği, öğrenme adını verdiğimiz süreç içinde beynin çalışmasını değiştirdiğinden dolayı önemlidir. Çünkü beyin bütün yeni uyarılardan yeni şeyler öğrenir, verilen bir uyarıya her zaman aynı tepkide bulunmaz. Ateşten eli yanan bir çocuk ateşle tekrar karşılaştığında daha farklı davranacaktır.

Beynin çıkışları, konuşmak veya vücudu hareket ettirmek için gereken adale kasılmalarının denetiminde de kullanılır.

Şekil 1.4'de Bilgisayarın insanla nasıl karşılaştırılabileceği gösterilmektedir.

Bilgisayarlar insanların sahip olduğu tüm girişlere duyarlı olabileceği gibi bizim algılayamayacağımız bazı şeyleri de giriş olarak kabul edebilir. Bilgisayarlar,



Şekil 1.4 Bir bilgisayarın organizasyonu

morötesi, kızılaltı, X ışını ve radyo dalgalarını doğrudan algılayabilirler. Bilgisayar çıkışları, insana oranla daha çeşitlilik gösterir. Motorlar aracılığıyla mekanik hareket sağlayabilir, doğrudan (TV ekranlarında) ışık, (hoparlörlerden) ses ve (ısıtıcılardan) ısı üretebilir.

Beynin, bilgisayardaki karşılığı, merkezi işlem birimi artı saklama birimi (bellek)'dir. Beyin gibi bilgisayarlar da, çeşitli girişleri işler ve bunlara karşılık çıkışlar üretirler.

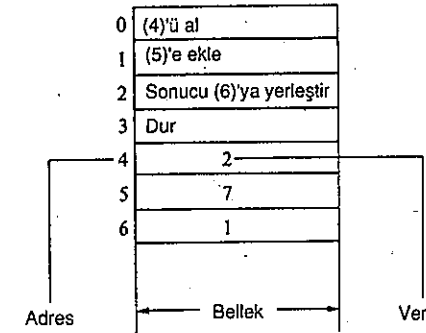
Bu kitap, insan beyni ile bilgisayar arasındaki farkları inceleyen bir kitap değildir. Ancak, beynin bilgi işleme ve bellek işlevlerinin kendi aralarında sıkı bir ilişki içinde olmasına karşın bir bilgisayarda bu birimler birbirinden fiziksel olarak ayrıdır. Bilgi işlem alanında esaslı çıkışın, bilgisayar mimarisinin, insan beyninin özelliklerinin yerini daha da almaya başlamasıyla ortaya çıkacağını düşünenler bulunmaktadır.

Bilgisayarın belleğinde saklananlar programdır. "Program" sözcüğü, 'çalışma programı' veya 'müzik programı' ifadelerinde geçen programla aynı anlamdadır. Bir bilgisayar programı, bilgisayar tarafından sırayla uygulanacak işlemlerin bir toplamıdır. Bilgisayar programına klasik bir benzetim, bir yemek kitabındaki yemek tarifleridir. Yemek tarifleri de tek tek ve sırayla uyulması gereken bir komutlar dizisidir. Bu benzetme, yemek pişirmede uyulacak komutlar, bilgisayarın bellekte saklı bulunan veriler üzerinde işlem yapmasında olduğu gibi, malzemelere ilişkin işlemleri gerektirdiğinden bu benzetme konuya açıklık getirmesi açısından özellikle uygun olmaktadır.

Merkezi işlem birimi, programı oluşturan komutları birer birer okur ve herbir komutu bellekten getirdikçe yürütür. Burada "yürütmek" sözcüğü yapmak, işleme sokmak anlamındadır. Örneğin 'A'yı B'ye ekle' komutu, A olarak adlandırılan değerin B olarak adlandırılan değerle toplanmasına neden olur. Burada bu komutların gerçek yapılarından söz edilmemiştir. Burada önemli olan, bilgisayar tarafından yürütülen en karmaşık işlemlerin, belirli sayıda daha basit işlemlere bölünebileceği gerçeğidir. Tekrar edersek, Einstein'ın en yüksek düzeydeki düşünceleri bile, beyninin hücrelerinden, sinirler boyunca iletilen kısa süreli elektrik darbelerine indirgenebilir.

Bilgisayarlar işlemlerini, saklama biriminde (bellek) tutulan ve 'veri' olarak adlandırılan bilgiler üzerine gerçekleştirmektedir. Bu birim aynı zamanda içinde programında da yer aldığı bellektir. Yani bilgisayar tarafından kullanılan veri ve programlar aynı bellekte fakat farklı bölgelerde yerleşmiştir.

**Not:** Bu kitap boyunca köşeli parantezler içinde verilen rakamlar, 'bellek içeriğini' göstermektedir, aşağıdaki örnekte verilen 4, bellek yeri 4'ün içeriği 2 diye okunur.



Şekil 1.5 Program ve verilerin bellekteki yerleşimi

Şekil 1.5'de program (komutlar) ve verilerin aynı bellekte nasıl bir arada bulunduklarını göstermektedir. Bu belleğin, 0'dan 6'ya kadar sıralanan 7 konumu bulunmaktadır. Bu konumlara, kutu veya güvercin deliği (yuva) da denmektedir. İçinde verilerin tutulduğu bu 'kutulardan' herbiri tek bir yere veya adrese sahiptir. Bellekler normalde, bir dizi saklama konumu olarak kabul edilir. Şekil 1.5'de 5 numaralı adres 7 değerini içermektedir. Bilgisayarlarla insanlar arasındaki en önemli farklardan biri de insanların sıralamayı 1...m şeklinde, bilgisayarların ise 0...m-1 şeklinde yapmalarıdır. Bunun nedeni 0'ın geçerli bir tanıttıcı olmasıdır. Maalesef insanlar sık sık 0 (tanıttıcı) ile 0 (hiçbirşey)'i karıştırmaktadırlar.

Program, 0'dan 3'e kadar ardışıl bellek konumlarını kaplamaktadır. Veriler ise 4 ile 6. konumlar arasında yerleşmiştir. İlk komut "[4]'ü al", bellekte 4 numaralı bellek konumundaki içeriği al getir anlamındadır. Ondan sonra gelen komut olan "onu [5]'e ekle" ise, bu değerin 5 nolu bellek konumu içeriğine ekleneceğini göstermektedir. Böylece bilgisayar 2 ile 7'yi toplayarak 9 elde eder. Üçüncü komut ise sonucu [6]'ya yerleştir, bilgisayara, sonucu (yani 9'u) 6 nolu konuma yerleştirmesini söyler. [6] numaralı konumda daha önce bulunan 1 yerine 9 yazılır. 3 numaralı konumdaki son komut ise bilgisayara durmasını söyler.

## 2

## MANTIK ELEMANLARI VE BOOLE CEBRİ

Bu bölümde, bir bilgisayarı oluşturan temel bileşenlerin davranışlarını inceleyeceğiz. Bilgisayarda iki belirgin bileşen sınıfı bulunmaktadır. Kapılar ve flip-floplar. Bunlar "birleşik" ve "ardışıl" mantık elemanları olarak bilinir.

Bu bölüm, bazı temel kapıların özelliklerini tanımlamakla başlar. Ardından bir ev veya okul inşa etmek için tuğlaların bir araya getirilmesi örneğinde olduğu gibi, yararlı fonksiyonları başarmak için kapıların nasıl bir araya getirilebileceğini de gösterir. Kapıların tanımlarını verdikten sonra, kapı içeren devrelerin çözümlemesi için formal gereç olan Boole cebri bölümü gelmektedir. Bu, Boole denklemlerini basitleştirmek ve kullanmak için grafik bir teknikten başka bir şey olmayan Karnaugh diyagramlarını gündeme getirir.

Konunun ikinci bölümü, flip-floplardan oluşmuş olan ardışıl devrelerle ilgilidir. Adından da anlaşılacağı üzere bu devreler zaman faktörünü içermekte olup davranışları, zamanın bir fonksiyonu olarak ele alınacaktır.

Bölümün son kısmında tasarımcıya ilginç gelebilecek, bazı bilgisayar devre yapılarına kısaca bir göz atacağız. Bu kısım ayrıntılı bir inceleme olmayıp bir gözden geçirme niteliğindedir. Verilen konular, kapıların elektriksel karakteristikleri, ekonomik yönleri ve genelde sistemin güvenilirliğidir.

Sayısal bilgisayarın en kayda değer gerçeklerinden biri de, tüm devrelerinin kapı denilen birkaç basit elemanın bağlantılarının toplamından başka birşey olmamasıdır. Üç temel kapı devresi vardır. VE, VEYA ve TERSLEYİCİ (INVERTOR), ayrıca diğer üçünden türetme iki kapı (yani VEDEĞİL ve VEYADEĞİL) devreleri de bulunmaktadır. Daha sonra göreceğimiz gibi tüm diğer kapılar, sonuç olarak sayısal devreler, sadece VEDEĞİL ve VEYADEĞİL kapılarının uygun bağlantılarıyla oluşturulabilir. Böylece en karmaşık sayısal devreler de, bir VEDEĞİL kapılar topluluğuna indirgenebilir. Bu gerçek bilgisayarın değerini düşürmez, insan beyni de özel olarak karmaşık biçimde bağlanmış nöronlardan oluşmuştur.

Boole Cebri (yani kapıların özelliklerini tanımlamak için kullanılan aritmetik) çok zevkli olduğundan ya da sadece kapıları sevdiğimiz için onları bilgisayar oluşturmada kullanmayız. Bilgisayarın içindeki bilgiler, sayısal biçimde gösterilirler. Bir sayısal sistemde tüm değişkenler ve sabitler, değerlerini, seçilen 'alfabe' adlı bir değerler grubundan almalıdırlar. Onluk aritmetikte 0,1,2,...,9 gibi sembollerden oluşmuş bir alfabemiz vardır. Mors kodunda ise nokta, çizgi, uzun aralık ve kısa

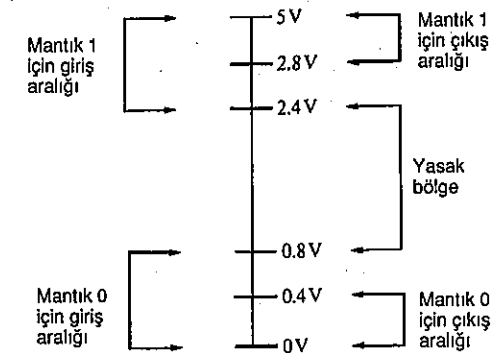
aralıktan meydana gelen alfabemiz vardır. Diğer sayısal sistemler Braille ve haftanın günleridir. Sayısal göstermenin bir avantajı da, bir sembolün bozulması halinde, eğer bozulma düzeyi, bir sembolün diğer bir sembol gibi algılanmayacağı düzeydeyse, asıl sembolün daima anlaşılıp tekrar oluşturulabilmesidir.

Sayısal bilgisayarlar için seçilen alfabede iki sembol vardır. 0 ve 1 (Bazen, doğru ve yanlış, alçak ve yüksek veya açık-kapalı olarak adlandırılır).

Böyle bir alfabenin avantajı, sembollerin birbirine benzemeyecek ve ayırtedilmelerine olanak tanıyacak biçimde oluşturulabilmeleridir. Bazı bilgisayarlar bilgileri kağıt bant üzerinde saklarlar. Delik, ikili biçimde 1 değerini, delik olmayan kısımlar ise 0'ları gösterir. Bandı okurken bilgisayar sadece delikli ve deliksiz kısımları ayırteder. Şimdi aynı şeyi, ikili bilgisayar yerine onluk bilgisayarla yapmaya kalktığımızı varsayalım. Aynı kağıt-bant kullanılmakta olsun. Kağıtta onluk bir sayı, bir deliğin veya, deliksiz kısmın uzunluğunun 9 katı uzunlukta gösterebilecektir (Hepsi on sembol). Farklı büyüklükteki delik veya deliksiz kısımları, yani örneğin 5 ve 7'i birbirinden nasıl ayıracacağız. Bu işlem son derecede hassas bir elektroniği gerektirmektedir.

Tek bir ikili basamak, "bit" (Binary digiT) olarak adlandırılır ve, olabilecek en küçük bilgi birimidir. Yani bir bit daha küçük birimlerine bölünemez. Günümüz teknolojisinde, bilgisayar içindeki ikili bilgi değerleri, iki farklı gerilim düzeyi ile gösterilmektedir. Şekil 2.1'de, sayısal bilgisayar içinde bulunan mantık elemanlarının gerilim düzeyleri gösterilmiştir.

Sayısal eleman üreticileri, kullanıcıya belli sözler veya garantiler verir. Başta, bir kapının mantıksal 0 çıkışının 0.4 V'la 0 bölgesinde, mantıksal 1 çıkışının ise 2.8



Şekil 2.1 Tipik mantık elemanının iki durumu

V ila 5 V bölgesinde olacağı garanti edilir. Giriş bölgesini (verilen mantıksal değeri için) çıkış bölgesinden daha geniş tutarak, tasarımcı, gürültü veya istenmeyen sinyallerinin karışmasına izin vermiş olur. Örneğin 0.4 Voltluk mantıksal 0 çıkışı 0.2 Voltluk geçiş (spike) darbesi eklenmesiyle 0.6 Volta çıkar ve bu değere kadar olan gerilim düzeylerinin mantıksal 0 olarak algılanacağı garanti edilir. Bir mantıksal değer için giriş ve çıkış bölgeleri arasındaki fark, bu mantıksal değer için garanti edilen gürültü bağırsıklığı olarak bilinir. Mantık elemanlarının elektriksel karakteristiklerine ait ayrıntılar Kısım 2.7'de verilmiştir.

### Mantıksal değerler üzerine notlar

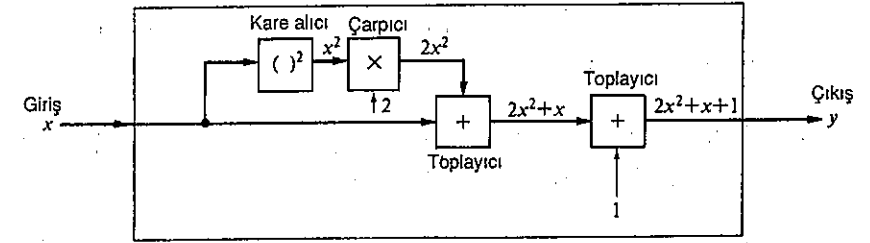
1. Daima iki ayrı durum vardır ve her mantıksal değer bu durumlardan birine sahip olmalıdır. Geçerli bir ara değer yoktur (Durum ya 0, ya da 1'dir).
2. Her mantıksal giriş veya çıkış verilen bir zaman aralığında yalnızca tek bir durumda olabilir.
3. Her mantıksal durumun bir tümleyeni, tersi veya zıddı vardır. Doğru (bir) durumunun tümleyeni yanlış (sıfır) durumudur.
4. Mantıksal değer bir sabit veya değişken olabilir. Eğer bir sabitse daima aynı durumda kalır, eğer değişkense 0 ya da 1 değerini alabilir.
5. Değişken sık sık neden olduğu olayla isimlendirilir. Takibeden mantıksal değişkenler kendilerini tanımlamaktadır. BAŞLA, DUR, SIFIRLA, SAY, EKLE.
6. Fonksiyonu, kendi adıyla ifade edilen bir değişkenin mantıksal değeri, seçime bağlıdır. Eğer işlem, mantıksal bir 'le yerine getiriliyorsa, değişken yüksekte-etkindir, işlemi mantıksal 0 gerçekleştiriyorsa alçakta-etkin adını alır. Böylece, eğer bir yüksekte-etkin bir değişken BAŞLA olarak etiketlenmişse işlemi, bir mantıksal 1 (yani BAŞLA=1) değeri başlatacaktır. Eğer alçakta-etkin bir değişken BAŞLA olarak etiketlenmişse, işlemi mantıksal sıfır başlatacaktır.

### 2.1 Temel kapılar

Bu kısımda tüm sayısal sistemlerin oluşturulduğu beş temel kapının özelliklerini tanımlayacağız. Kapı terimi, iki konumlu -açık veya kapalı olarak çalışan- birim temeline dayalıdır.

Kapılar, girişleri ve bir çıkışı olan bir kara-kutu olarak düşünülebilir. Giriş uçlarına gelen sayısal sinyaller kapı tarafından işlenerek çıkış terminalinde bir sayısal sinyal üretirler. Kapının işlevi, yapılacak işlemin ne olacağını belirleyecektir. Eğer bir kapı, A ve B gibi iki giriş ucuna ve C gibi bir çıkış ucuna sahipse çıkış  $C = F(A,B)$  yazılabilir. Bu denklem klasik bir cebir ifadesi şeklinde gösterilmiştir, buradaki A, B, C; iki değerlileri, F ise bir mantıksal fonksiyonu gösterir.

Kapı düşüncesine bu tür bir yaklaşım getirmek sadece basit değil, aynı zamanda da sıradandır. Analog dünyasından bir örnek alalım.  $y=F(x)$  denklemi,  $y=2x^2+x+1$  olarak verilmektedir. x, karakutunun girişi ve y ise çıkışıdır. Şekil 2.2'de, x üzerine olan bir dizi işlemden sonra y'nin nasıl üretildiğini göstermektedir. Buradaki işlemler, toplama, çarpma ve kare alma işlemleridir.

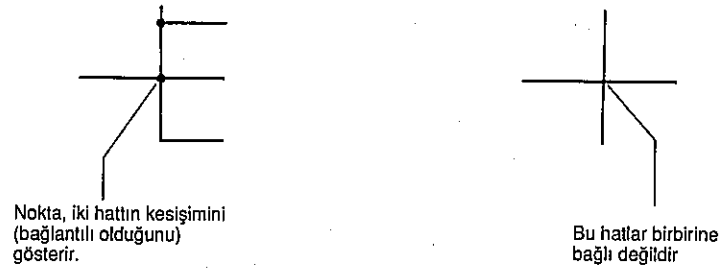


Şekil 2.2 Analog işlemci

Bu işlemlerin yapıldığı kutucuklar, tamamiyle sayısal dünyadaki kapılara benzerler.

Kapıların kendilerinden ve birbirine bağlanarak oluşturdukları sayısal devrelerden söz etmeden önce, birkaç temel kuralı açıklamak gerekir. Genelde soldan sağa yazdığımızdan, mantık devreleri de soldan sağa okunacaktır. Kapı girişleri sol tarafta olacak şekilde, bilgi akışı soldan sağadır. Bu, genelde birçok bilgi yolu bulunduğu ve diyagram da iki-boyutlu kağıt üzerine çizileceğinden, bilgi yollarının birbirini kesmesi gibi bir zorunluğu doğurmaktadır. Şekil 2.3'de, bunun nasıl yapıldığı gösterilmektedir. Genelde kesişen iki çizginin birbiriyle bağlantılı olması, üzerlerine konan bir noktayla gösterilir.

Genelde, çapraz biçimde çizilen iki çizginin birbirine bağlı olmadığı kabul edilir. İki çizginin birbiriyle bağlantılı oldukları, kesişme noktalarına konulan bir nokta ile gösterilir. Hat (yani iletken) boyunca gerilim sabittir, böylece mantık düzeyi

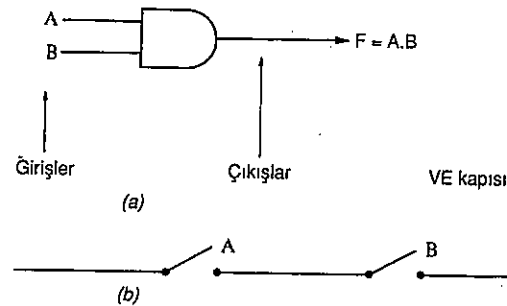


Şekil 2.3 Devre çizim kuralları

sabit kalır, yani iletkenin bir ucu P mantık düzeyinde ise, hat boyunca her noktada aynı kalır.

### VE (AND) kapısı

VE kapısı için mantıksal sembol Şekil 2.4 (a)'da verilmiştir. VE kapısı, ancak tüm girişler doğru olduğunda doğru çıkış verir. Bir VE kapısını incelemenin (anlamanın) klasik bir yolu; Şekil 2.4 (b)'de gösterildiği gibi, onu elektrik devresi veya yol terimleri ile göstermektir.



Şekil 2.4 (a) VE kapısı (b) - VE kapısının anahtarlar devresi ile gösterilmesi

Eğer A ve B anahtarları (köprü) kapalıysa, elektrik akımı (taşıt trafiği), devre (yol) boyunca akar. VE için mantıksal sembol bir nokta olup, A VE B ifadesi A.B olarak yazılır. Normal cebirde, nokta ihmal edilir ve A.B, AB olarak yazılır.

Kapıların (ve diğer devrelerin) giriş ve çıkışları arasındaki bağıntıları tanımlamanın elverişli bir yolu, doğruluk tablolarıdır. Bkz: Tablo 2.1. Doğruluk tablolarında, tüm olası giriş kombinasyonlarına karşılık gelen çıkışlar yazılır. Girişler çift değeri

olduklarından, n girişli bir devrenin doğruluk tablosunda  $2^n$  satır olacaktır.  $2^n$  olası girişin hangi sırada verildikleri önemli değildir, fakat genel bir kural olarak sıra, 0.....00, 0.....01, ....., 1.....10, 1.....11 biçiminde düzenlenir.

Tablo 2.1 VE kapısı için doğruluk tablosu

| Girişler |   | Çıkış     |                                                    |
|----------|---|-----------|----------------------------------------------------|
| A        | B | $F = A.B$ |                                                    |
| 0        | 0 | 0         | } Bir veya daha fazla giriş yanlışsa, çıkış YANLIŞ |
| 0        | 1 | 0         |                                                    |
| 1        | 0 | 0         |                                                    |
| 1        | 1 | 1         | Tüm girişler doğru ise, çıkış DOĞRU                |

Bu örnekte, iki girişli VE kapısını ele aldık. Bir kapıya birden fazla giriş bağlanmaması için hiçbir neden yoktur. Uygulamada bazı kapıların üç veya dört girişi olabilir, 74133 kapısı, 13 girişli bir VEDEĞİL kapısıdır. Kısım 2.3.2'de, yaygın biçimde kullanılan bazı kapıların ayrıntıları ele alınmıştır. Daha sonraki kısımlarda bilgisayar mimarisi ve Assembly diline geldiğimizde, VE işleminin sözcüklere de uygulanabileceğini göreceğiz. Sözcükler bitlerden oluşan gruplar olup, normalde tek bir birim olarak değerlendirilir. Örneğin, aşağıdaki iki sözcüğe VE'lenmesine bakalım;

```

1 1 0 1 1 1 0 0  A
0 1 1 0 0 1 0 1  B
-----
0 1 0 0 0 1 0 0  A.B

```

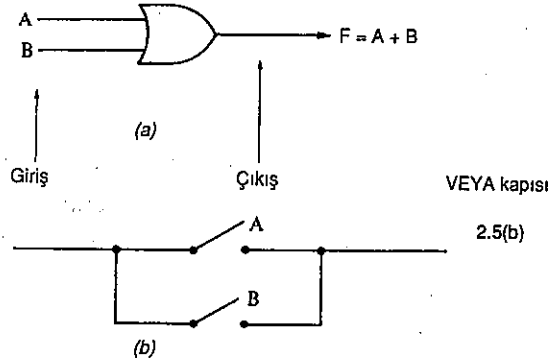
Burada VE işlemi her bir bit çifti için yapılmıştır. Öyle ki A sözcüğünün i'inci biti ile B sözcüğünün i'inci biti VE'lenmiştir. VE işlemi sözcüklere uygulandığı zaman, toplama veya çıkarma gibi aritmetik işlemlerden onu ayırdetmek için 'mantıksal işlem' olarak adlandırılır.

Mantıksal VE işlemi, sözcüklerdeki bazı bitlerin maskelenmesinde de kullanılır. Örneğin 8-bitlik bir sözcüğün son dört bitini sıfırlamak isteyelim, sözcüğü 00001111 ile VE'lemek, istenen sonucu verecektir.

### VEYA (OR) Kapısı

VEYA kapısının mantıksal sembolü, Şekil 2.5 (a)'da verilmiştir. Veya kapısının çıkışı, her iki giriş veya girişlerden biri doğru olması halinde doğrudur. VEYA

kapısı, Şekil 2.5 (b)'de olduğu gibi, anahtar devresiyle de gösterilebilir. Bu işlemin mantıksal sembolü artı işaretidir, böylece A VEYA B ifadesi  $A+B$  olarak yazılır.



Şekil 2.5 (a) VEYA kapısı (b) Veya kapısının anahtar devreleriyle gösterilmesi

Şekil 2.5 (b)'deki devre eğer A ve B kapalı ise tamamlanmıştır. VEYA sözcüğünün buradaki kullanılışı, dilimizdeki kullanımından farklıdır. Boole cebrinin VEYA'sı (A veya B) veya (A ve B) anlamına gelmektedir. Konuşma dilindeki kullanımda ise A veya B anlamına gelip A ve B'yi içermez. VEYA'nın dilimizdeki biçimiyle kullanılışı, Boole cebrinde ÖZEL VEYA 'EXOR' olarak bilinen fonksiyona karşı gelmektedir. Özel-VEYA önemli bir fonksiyon olup, İngilizce'de EOR veya XOR olarak da gösterilebilmektedir.

Tablo 2.2 VEYA kapısı için doğruluk tablosu

| Girişler |   | Çıkış   |                                                                                        |
|----------|---|---------|----------------------------------------------------------------------------------------|
| A        | B | $F=A+B$ |                                                                                        |
| 0        | 0 | 0       | Tüm girişler yanlışsa, çıkış YANLIŞ<br>Bir veya daha fazla giriş doğruysa, çıkış DOĞRU |
| 0        | 1 | 1       |                                                                                        |
| 1        | 0 | 1       |                                                                                        |
| 1        | 1 | 1       |                                                                                        |

Aşağıdaki örnekte olduğu gibi bir bilgisayar, sözcükler üzerinde de mantıksal VEYA işlemi yapılabilir.

```

1 0 0 1 1 1 0 0  A
0 0 1 0 0 1 0 1  B
-----
1 0 1 1 1 1 0 1  A+B

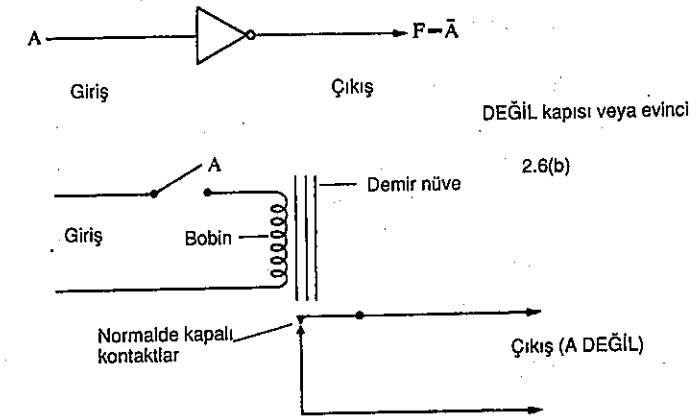
```

Mantıksal VEYA işlemleri bir sözcükteki bir veya birkaç biti kurmada da kullanılabilir.

Kurmak, mantıksal 1 yapmaktır. Çözmek ise mantıksal olarak 0'a getirmektir. Örneğin bir sözcüğün en küçük değerlikli biti, sözcüğü 00....01 ile VEYA'layarak elde edilebilir.

### DEĞİL (NOT) kapısı

Bir DEĞİL kapısının mantıksal sembolü Şekil 2.6 (a)'da verilmiştir. DEĞİL kapısı (invertör veya komplementör), girişindeki sinyalin durumunu tersine çevirir. DEĞİL kapısının çalışması Şekil 2.6 (b) olduğu gibi, röleler aracılığı ile gerçekleştirilebilir.



Şekil 2.6 (a) DEĞİL kapısı veya tersleyici (b) DEĞİL kapısının elektronik röle ile gösterimi

Eğer A anahtarı açıksa bobinden akım geçmez ve demir nüve mıknatıslanmamıştır. Normalde-kapalı kontaklar, kapalı olup; A açık olduğu zaman, kapalı olan bir anahtar oluştururlar. A kapatıldığında, bobinden akım geçer ve demir mıknatıslanır. Normalde-kapalı anahtarın bir kısmı demire doğru çekilerek devreyi açar ve A anahtarına göre ters konuma geçer. Röle, bilgisayar tarafından, dış aygıtların denetiminde kullanılabilir ve Kısım 5.5.3'de daha ayrıntılı olarak ele alınacaklardır.

Tablo 2.3 Tersleyici için doğruluk tablosu

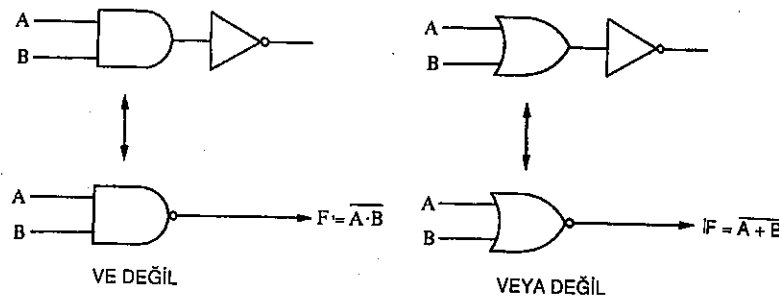
| Giriş<br>A | Çıkış<br>$F=\bar{A}$ |
|------------|----------------------|
| 0          | 1                    |
| 1          | 0                    |

DEĞİL'leme işlevi sözcüklere de uygulanabilir.

1 1 0 1 1 1 0 0 A  
0 0 1 0 0 0 1 1  $\bar{A}$  veya A DEĞİL

VEDEĞİL (NAND) ve VEYADEĞİL (NOR) kapıları

VEDEĞİL ve VEYADEĞİL (NAND ve NOR) kapılarına ilişkin mantıksal semboller Şekil 2.7'de verilmiştir. Bu iki kapı, bir VE kapısı ve ardından bir tersleyici (VEDEĞİL=NAND) ve bir VEYA (VEYA=OR) kapısı ve ardından bir tersleyiciye (VEYADEĞİL=NOR) eşdeğerdir.



Şekil 2.7 VEDEĞİL kapısı ve VEYADEĞİL kapısı

Şu noktaya dikkat edilmelidir ki burada  $\overline{A \cdot B}$ ,  $\bar{A} \cdot \bar{B}$  ile aynı şey olmayıp  $\overline{A+B}$  de  $\bar{A} + \bar{B}$  den farklıdır.

Tablo 2.4 VEDEĞİL kapısı için doğruluk tablosu

| Girişler |   | Çıkış                      |
|----------|---|----------------------------|
| A        | B | $F = \overline{A \cdot B}$ |
| 0        | 0 | 1                          |
| 0        | 1 | 1                          |
| 1        | 0 | 1                          |
| 1        | 1 | 0                          |

Tablo 2.5 VEYADEĞİL kapısı için doğruluk tablosu

| Girişler |   | Çıkış                  |
|----------|---|------------------------|
| A        | B | $F = \overline{A + B}$ |
| 0        | 0 | 1                      |
| 0        | 1 | 0                      |
| 1        | 0 | 0                      |
| 1        | 1 | 0                      |

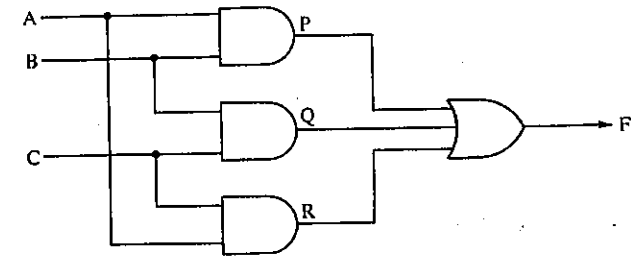
Öğrencinin, A ve B gibi iki girişi olan kapılara ilişkin farklı sonuçları kavrayabilmesi için yukarıda adı geçen dört kapının çıkışlarının bir karşılaştırması Tablo 2.6'da, verilmiştir. Bu tabloya, referans olarak ÖZEL VEYA (XOR) ve onun tümleneni olan ÖZEL VEYADEĞİL (XNOR) eklenmiştir. Özel-VEYA (EXCLUSIVE OR); VE, VEYA ve DEĞİL kapılarından türetilmiş olup, bu bölümün sonunda ele alınacaktır.

Tablo 2.6 Altı temel kapı için doğruluk tablosu

| Girişler |   | Çıkışlar        |                   |                             |                                         |                               |                                               |
|----------|---|-----------------|-------------------|-----------------------------|-----------------------------------------|-------------------------------|-----------------------------------------------|
| A        | B | VEYA<br>$A + B$ | VE<br>$A \cdot B$ | DEĞİL<br>$\overline{A + B}$ | VEYA<br>DEĞİL<br>$\overline{A \cdot B}$ | ÖZEL-<br>VEYA<br>$A \oplus B$ | ÖZEL-<br>VEYADEĞİL<br>$\overline{A \oplus B}$ |
| 0        | 0 | 0               | 0                 | 1                           | 1                                       | 0                             | 1                                             |
| 0        | 1 | 1               | 0                 | 0                           | 1                                       | 1                             | 0                                             |
| 1        | 0 | 1               | 0                 | 0                           | 1                                       | 1                             | 0                                             |
| 1        | 1 | 1               | 1                 | 0                           | 0                                       | 0                             | 1                                             |

## 2.2 Bazı kapı uygulamaları

Bu bölümde; kapı uygulamalarına ait çok sayıda uygulama, örnekler şeklinde sunulmuştur. Amacımız, işlevi ve önemi, okuyucu tarafından değerlendirilmek üzere, birkaç kapıyı birbirine bağlayarak bir devre kurmaktır.



Şekil 2.8 Kapıların kullanılışı - Örnek 1

Örnek 1 Şekil 2.8'den, A, B ve C gibi üç girişin fonksiyonu olarak F çıkışı üretecek biçimde bir doğruluk tablosu (Tablo 2.7) oluşturabiliriz. Üç giriş olduğundan, doğruluk tablosunda sekiz satır olmalıdır. İşlemleri kolaylaştırmak için araya P, Q ve R gibi ara sonuçlar eklenmiştir.

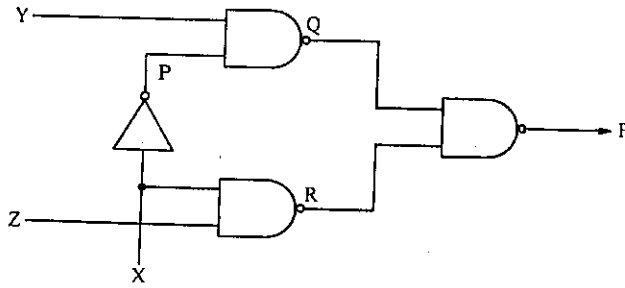
Çıkış fonksiyonu F,  $F = P + Q + R$  olarak verilir. Burada  $P = AB$ ,  $Q = BC$ ,  $R = AC$  olup,  $F = AB + BC + AC$  olarak yazılabilir. Eğer doğruluk tablosu incelenirse iki veya daha fazla giriş A, B ve C doğru ise, çıkış da doğru olacaktır. Yani, bu devre yapısı,

çoğunluk mantık devresini göstermekte olup çıkışı, girişlerin çoğunluğunun durumu belirlemektedir. Yani çıkış, çoğunluk girişlerle aynı değere sahiptir.

Tablo 2.7 Şekil 2.8 için doğruluk tablosu

| Girişler |   |   | Ara değerler |        |        | Çıkış         |
|----------|---|---|--------------|--------|--------|---------------|
| A        | B | C | P = AB       | Q = BC | R = AC | F = P + Q + R |
| 0        | 0 | 0 | 0            | 0      | 0      | 0             |
| 0        | 0 | 1 | 0            | 0      | 0      | 0             |
| 0        | 1 | 0 | 0            | 0      | 0      | 0             |
| 0        | 1 | 1 | 0            | 1      | 0      | 1             |
| 1        | 0 | 0 | 0            | 0      | 0      | 0             |
| 1        | 0 | 1 | 0            | 0      | 1      | 1             |
| 1        | 1 | 0 | 1            | 0      | 0      | 1             |
| 1        | 1 | 1 | 1            | 1      | 1      | 1             |

Böyle bir devrenin, bir uçağı otomatik olarak yere indiren, otomatik indirme sistemlerinde nasıl kullanılabildiğini daha önce görmüştük.



2.9 Kapıların kullanılması-Örnek 2

Örnek 2. Daha önceki örnekte olduğu gibi Şekil 2.9 devresi, üç giriş bir çıkış ve üç ara değere sahiptir.

Tablo 2.8 Şekil 2.9 için doğruluk tablosu

| Girişler |   |   | Ara değerler  |                |                | Çıkış          |
|----------|---|---|---------------|----------------|----------------|----------------|
| X        | Y | Z | P = $\bar{X}$ | Q = $\bar{Y}P$ | R = $\bar{X}Z$ | F = $\bar{Q}R$ |
| 0        | 0 | 0 | 1             | 1              | 1              | 0              |
| 0        | 0 | 1 | 1             | 1              | 1              | 0              |
| 0        | 1 | 0 | 1             | 0              | 1              | 1              |
| 0        | 1 | 1 | 1             | 0              | 1              | 1              |
| 1        | 0 | 0 | 0             | 1              | 1              | 0              |
| 1        | 0 | 1 | 0             | 1              | 0              | 1              |
| 1        | 1 | 0 | 0             | 1              | 1              | 0              |
| 1        | 1 | 1 | 0             | 1              | 0              | 1              |

Doğruluk Tablosunun (Tablo 2.8) incelenmesiyle görülebilir ki giriş x, 0 olduğu zaman, F, Y'ye eşittir. X, 1 olunca, çıkış da Z'ye eşit olur. Yani devre bir elektronik anahtar olup, denetim sinyali X'in durumuna göre çıkışı, iki girişten, yani Y ve Z'den birine bağlamaktadır.

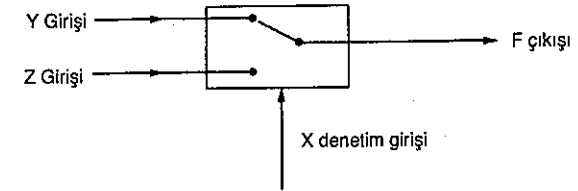
Bu devre iki girişli çoğullayıcı (multiplexer) adını alır ve Şekil 2.10'da verilen bir düzenlemeyle gösterilebilir. Çoğullayıcı sözcüğü sıkça kullanıldığından, genelde MUX veya MPLX olarak kısaltılır.

Şekil 2.9'daki devre diyagramından, kapıların girişler üzerindeki etkisini dikkate alarak F çıkışı için bir eşitlik türetilir.

$$F = \bar{Q}R, Q = \bar{Y}P, P = \bar{X}$$

$$\text{böylece } Q = \bar{X}\bar{Y}, R = \bar{X}Z$$

$$\text{Sonuç olarak } F = \bar{X}\bar{Y}Z + \bar{X}YZ \text{ elde edilir.}$$



Şekil 2.10 Şekil 2.9'un mantıksal gösterilmesi

Boole cebri bölümüne geldiğimizde, bu tip bir ifadenin nasıl basitleştirilebileceğini göreceğiz. Boole ifadesini elde etmenin diğer bir yolu ise doğruluk tabloları kullanmaktır. Çıkış sütununda her ne zaman 1 görülürse, çıkışı 1 yapacak şekilde bir giriş kümesi yazılabilir. Bu yüzden aşağıda verilen durumlar için çıkış doğru (1) olur:

- (i)  $X = 0, Y = 1, Z = 0$  ( $\bar{X}YZ$ )
- (ii)  $X = 0, Y = 1, Z = 1$  ( $\bar{X}YZ$ )
- (iii)  $X = 1, Y = 0, Z = 1$  ( $X\bar{Y}Z$ )
- veya (iv)  $X = 1, Y = 1, Z = 1$  ( $XYZ$ )

Çıkışı doğru duruma getiren dört olası kombinasyon vardır. Böylece çıkış

$$F = \bar{X}YZ + \bar{X}YZ + X\bar{Y}Z + XYZ$$

olarak ifade edilebilir.

Yani, yukarıdaki dört grup doğru ise fonksiyon da doğrudur. Belirli sayıda değişkene VE işlemi uygulanması ile elde edilen terimlere, VEYA işlemi uygulandığından dolayı; elde edilen fonksiyonu "çarpımların toplamı" adı verilir.

(i)'den (iv) kadar yukarıdaki her bir terim minterm adını alır. Minterm bir VE (çarpım) terimi olup, herbir değişkeni, kendisi veya tümleyeni biçiminde içermektedir. Örneğin yukarıdaki durumda  $\overline{X}YZ$ , bir minterm'dir; fakat,  $\overline{X}Y$ , iki değişken içerdiğinden değildir. İster doğruluk tablosundan, isterse de mantık diyagramının türetilsin, devrenin çıkışı aynı olmak zorunda olduğundan, yukarıdaki iki denklem eşit olmalıdır. Sonuç olarak,

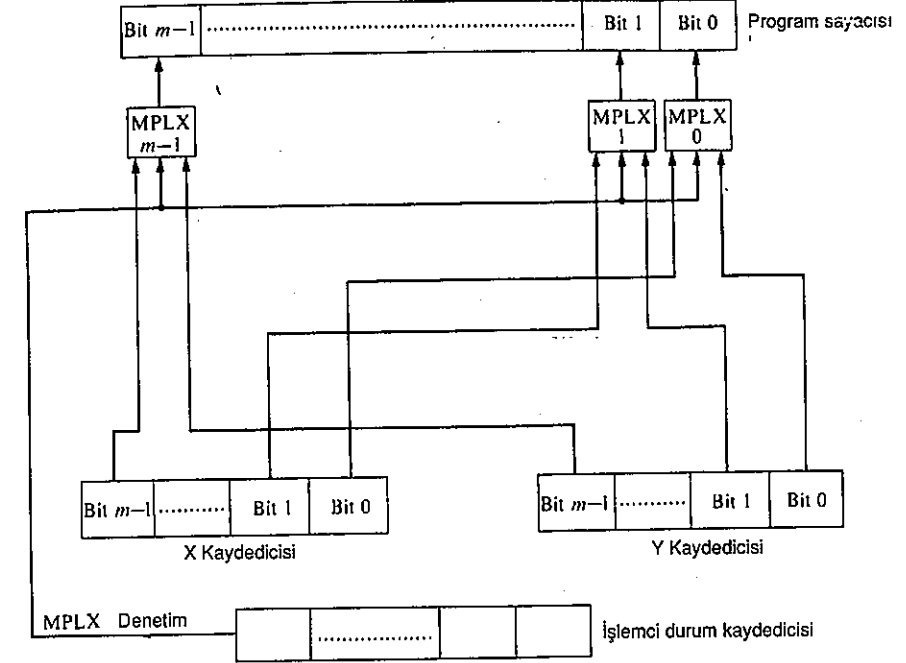
$$\overline{X}Y \cdot \overline{X}Z = \overline{X}YZ + \overline{X}YZ + X\overline{Y}Z + XYZ$$

Çoğullayıcı ile programlamanın ne ilgisi var diye düşünülebilir. Fakat onu buraya, yazılımla donanım arasındaki ilişkiyi göstermek üzere ekledik. Sayısal bilgisayarın (veya insan beyninin) gücü, onun karar verebilme yeteneğindedir. Bilgisayar dünyasında karar verme, koşullu dallanma veya atlamalarla sağlanır. Örneğin, yüksek düzeyli dillerde yazılı aşağıdaki ifadeleri dikkate alalım.

|                         |         |
|-------------------------|---------|
| IF (X-2.0*Z) 10, 20, 30 | FORTRAN |
| IF X=5 THEN Z=27        | BASIC   |
| IF t>=0 THEN I:=I+2;    | Pascal  |

BASIC ve PASCAL komutları, iki yollu dallanma oluşturdıkları halde, FORTRAN üç yollu dallanmaya sahiptir. Uygulamada, üç yollu dallanma, iki ardışıl iki yollu dallanmayla sağlanır. Koşullu dallanma, verilen bir koşulun test edilmesi (örneğin X=0), ve eğer koşul sağlanıyorsa (then) kısmının işleme koyulması, eğer sağlanmıyorsa bir sonraki komuta geçilmesi ile gerçekleştirilir. Uygulamada dallanma, program sayıcısına, iki olası sayıdan birini yüklemekle yapılır. Program sayacı MİB'nin bir bölümü olup, yürütülecek bir sonraki komutun adresini (yani konumunu) tutar. Bu adresi değiştirerek, iki farklı işlem arasında seçim yapılabilir. Şimdi Şekil 2.11'deki düzenlemeyi dikkate alalım. Okuyucu eğer MİB'nin temel yapısı hakkında bilgi sahibi değilse, 4. Bölümü okuyuncaya kadar, bu örnek atlanabilir.

İki X ve Y kaydedici, m-bitlik adresler içerirler. Bu kaydedicilerdeki m-bit, çıkışları program sayıcısının m-bitini besleyen, m çoğullayıcıyı besler. Çoğullayıcıların denetim girişleri birbirine bağlı olup, işlemci durum kaydedicisinin bir bitine de bağlantılıdır.



Şekil 2.11 Bir çoğullayıcı uygulaması

İşlemci durum kaydedicisinin (PSR) içeriği, belirli hesaplamalar sonucunda belirlenmekte olup, çeşitli bitler bize son işlem sonucunun sıfır, pozitif, negatif vb. olduğunu gösterir. Şekil 2.11'de PSR'nin özel bir bitinin, X veya Y kaydedicilerinin içeriklerini program sayıcısı için beslediğini, böylece, bir sonraki etapta iki olası işlem sırasından hangisinin gerçekleştirileceğini dikte ettiğini görebiliriz. Bu örnek bir yazılım işleminin nasıl yapılabileceğini göstermek üzere seçilmiştir. Uygulamada, koşullu atlamaların, gerçek bilgisayarlarda gerçekleştirilme yolu, yukarıda anlatılan işlem dizisinden biraz farklı olmaktadır.

Örnek 3 Şekil 2.12'de verilen devre için verilen doğruluk tablosunda (Tablo 2.9) iki giriş, iki ara değer ve bir çıkış vardır.

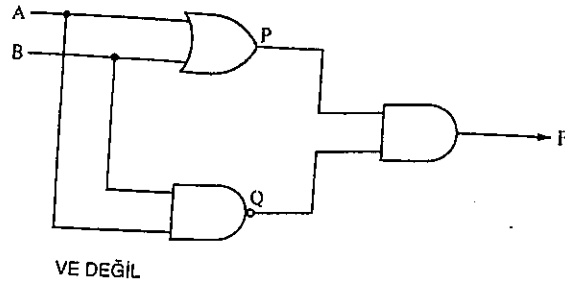
Bu devre sayısal elektroninin en önemli devrelerinden biridir. Çıkış, eğer girişlerden biri doğru ise doğrudur, fakat her iki giriş de doğru olursa çıkış yanlışır. Buna "EXCLUSIVE OR" (XOR) özel VEYA işlevi adını veriyoruz. Buradaki VEYA, dilimizdeki veya sözcüğü ile eşanlamlıdır. Bu işlevin çok popüler olması nedeniyle, ayrı bir sembolü vardır.

$$F = A \oplus B$$

$$F = A \text{ EOR } B$$

$$F = A \oplus B$$

$$F = A \text{ EOR } B$$



Şekil 2.12 Kapıların kullanılışı-Örnek 3

Tablo 2.9 Şekil 2.12 için doğruluk tablosu

| Girişler |   | Ara değerler |                                 | Çıkış    |
|----------|---|--------------|---------------------------------|----------|
| A        | B | $P = A + B$  | $Q = \overline{A} \overline{B}$ | $F = PQ$ |
| 0        | 0 | 0            | 1                               | 0        |
| 0        | 1 | 1            | 1                               | 1        |
| 1        | 0 | 1            | 1                               | 1        |
| 1        | 1 | 1            | 0                               | 0        |



Şekil 2.13 Özel-VEYA (XOR)

Şekil 2.13 Mutlak veya kapısının sembolünü vermektedir. Doğruluk tablosundan  $A=0$   $B=1$  veya  $A=1$   $B=0$  olduğu zaman  $F=1$  olmaktadır,  $F = \overline{A}B + A\overline{B}$  dir. Şekil 2.12'den;

$$F = PQ$$

$$P = A + B$$

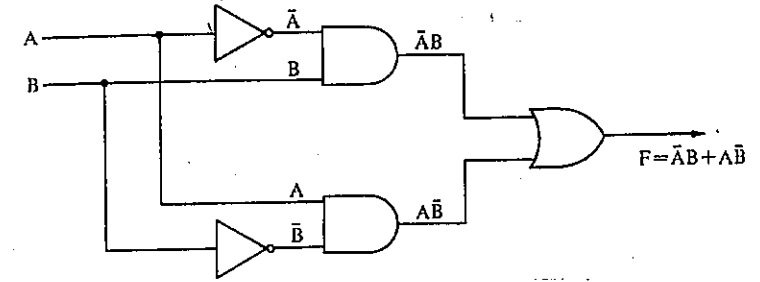
ve  $Q = \overline{A} \overline{B}$  olduğundan,

Böylece  $F = (A+B) \cdot \overline{A} \overline{B}$  elde edilir.

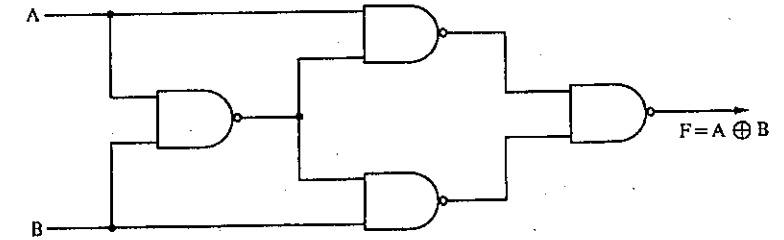
Bu iki denklemin (yani  $F = \overline{A}B + A\overline{B}$  ve  $F = (A+B) \cdot \overline{A} \overline{B}$ ) eşdeğer olması gerektiğinden bir Özel VEYA fonksiyonunu Şekil 2.14'de çizildiği gibi oluşturabiliriz.

Gerçekte bir Özel VEYA kapısı, Şekil 2.15'de gösterildiği gibi 4 adet VEDEĞİL kapısı içeren bir entegre devre ile gerçekleştirilebilir. Buradaki düzenin gerçekte bir Özel-VEYA fonksiyonu olup olmadığının incelenmesini, okuyucuya bir alış-

tırma olarak bırakıyorum.



Şekil 2.14 Özel VEYA'nın bir diğer şekilde gösterilmesi



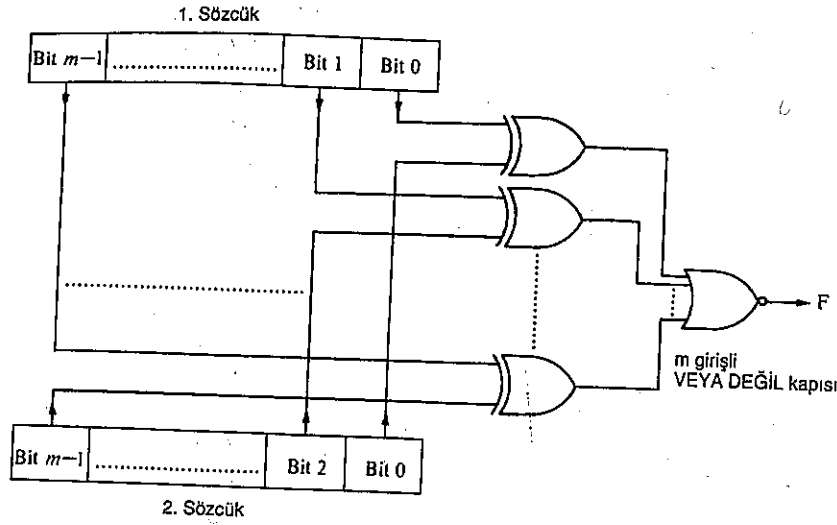
Şekil 2.15 Yalnızca VEDEĞİL kapıları kullanılarak yapılmış bir XOR Özel-VEYA

Özel VEYA XOR, kullanım alanı geniş bir mantık elemanı olup, sayısal elektrotikte birçok alanda kullanılmaktadır.

Eğer girişler farklı ise Özel VEYA'nın çıkışı doğru, girişler aynı ise yanlıştır. Bu özellik, XOR'un Özel VEYA'nın, iki sözcüğün özdeş olup olmadığını gösteren bir eşitlik test aygıtı yapımında kullanılabilmesini sağlar (Şekil 2.16). Eğer iki sözcük aynı ise tüm XOR'un Özel-VEYA kapılarının çıkışı 0 olup, m girişli XOR'un Özel-VEYA kapısının çıkışı mantıksal 1'dir.

Bu çıkış, işlemci durum kaydedicisinin bir bitine bağlanabilir. Böylece koşullu dallanma ile karşılaşıldığında, iki işlemten hangisinin seçileceğini belirleyebilir. İşlemci durum kaydedicisi ve merkezi işlem birimindeki rolü, Bölüm 4'de ele alınmıştır.

Bu aşamada açıkça anlaşıldığı üzere, verilen bir fonksiyon ancak tek bir Boole ifadesiyle gösterilebildiği halde, aynı fonksiyonu gerçekleştirmek üzere farklı kapı kombinasyonları kullanılabilir. Burada ayrıntılı mantık devrelerinin derini-



Şekil 2.16 Özel VEYA kapılarının bir eşitlik test edicisinde kullanımı.

ne inilmeyecek, ancak tasarımcının belirli bir uygulamayı diğerine nasıl tercih ettiğini görmesi ilginç olacaktır. Devrelerin seçiminde kullanılan temel kriterlerden bazıları aşağıda verilmiştir. Mantık devrelerinin tasarımı genelde, burada tanımlananlar dışında, başka faktörlerin de etkisi altındadır. Kısım 2.7'de, bu faktörlerin bazıları ele alınmıştır.

**Hız** Bir devrenin hızı (yani çıkışın, girişteki bir değişime ne kadar sürede cevap verdiği) yaklaşık olarak, değişikliğin, geçmek durumunda olduğu kapıların maksimum sayısıdır. Şekil 2.12'de, seri olarak bağlanmış 2 kapı, Şekil 2.14'de de 3 kapı vardır. Burada verilen ilk devre, diğerine oranla %50 daha hızlı çalışır. Dikkat edilirse, uygulamadaki tüm kapıların yayılma gecikmesinin aynı olmadığı görülür, Bazı kapılar değişikliğe, diğerlerine göre daha hızlı cevap verirler.

**Arabağlantı sayısı** Kapıların birbirine bağlanmalarının belli bir maliyeti vardır. Tek bir baskılı devre kullanılsa bile, bu devrenin biri tarafından tasarlanması gerektirir ve ne kadar fazla bağlantı kullanılırsa o kadar pahalı olacaktır. Devre tasarımında bir parametre de kapılara gelen girişlerin toplam sayısıdır. Şekil 2.14'de 8 giriş, Şekil 2.12'de ise 6 giriş vardır.

**Yonga sayısı** Burada sözkonusu basit kapılar, piyasadan 14-bacaklı yongalar halinde sağlanabilmektedir. (2 bacak, besleme için kullanılmaktadır). Gerçekte silikon yongasına yeni kapılar eklemek hiçbir ek maliyet getirmemekle birlikte, toplam kapı sayısı yalnızca verilen bacak sayısı ile belirlenmektedir.

Bir tersleyici için 2 bacak gerektiğinden, bu yonga üzerinde 6 tersleyici olabilir. Benzer biçimde iki girişli VE, VEDEĞİL, VEYA, VEYADEĞİL kapılarının üçer bacağa ihtiyacı olduğundan, bu kapılardan dördü, yonga içine konabilir. Bu devrelerden herbiri, üç farklı tip kapı kullandığından, her iki devre içinde 14-bacaklı bir entegre devre gerekir. Böyle dahi olsa, Şekil 2.12'deki devre, Şekil 2.14'deki devreden daha iyidir. Çünkü, entegre devrede kullanılmamış daha fazla kapısı bulunmakta olup geri kalan kapılar, bilgisayar sisteminin diğer birimlerinin kullanımı için serbest durumdadır.

## 2.3 Boole Cebrine giriş

George Boole bir İngiliz matematikçisidir ve 19. yüzyılda, mantığın matematiksel analizini geliştirmiştir. Boole cebri; 20. yüzyılda elektronğin gelişmesine katkıda bulunmadan önce, muhtemelen felsefeciler için bir malzeme olarak kullanılmıştı.

1938'de Claude Shannon, röleler kullanarak, Boole Cebri'ni, anahtarlama devrelerine uyguladı. Böyle devreler önce, telefon santrallerinde ve daha sonra da sayısal bilgisayarlarda kullanıldı.

Bugün Boole Cebri, Sayısal devreleri tasarlamak, davranışlarını analiz etmek ve onları basitleştirmek için sık sık kullanılmaktadır.

Cebirle ilk kez, 12 yaşında bir öğrenciyken tanıştım ve denklem çözmede bir anlam göremediğim için öğretmenime niçin cebir okuduğumuzu sordum. Öğretmen, "Clemence" dedi, "cebrî, öğrencilerin hayatını çekilmez hale getirmek için öğretiyoruz". Boole cebri için benzer sorular, öğrencilerimin kafalarından da sık sık geçmektedir. Sayısal tasarım, tasarıların, gerçek donanımına dönüştürülmesi olarak düşünülür. Boole cebri ise bu işlemi kolaylaştırmaktadır. Boole cebri, özellikle bir tasarımın matematiksel biçimde ifade edilmesine ve elde edilen sonuç ifadenin basitleştirilip, gerçek kapı donanımlarına ve diğer mantık elemanlarına dönüştürülmesine olanak tanımaktadır.

Boole cebri, 0 veya 1 değerini alabilen değişkenlerle ilgilenir. Harfler veya sabitler de kullanılır ve onlar da 0 veya 1 değerini almaktadırlar. Boole cebrinde, sadece üç tip işleme izin verilmektedir. İlk ikisi, + ile gösterilen mantıksal VEYA ve bir nokta ile gösterilen mantıksal VE'dir. Günlük hayatta kullandığımız toplama ve çarpma işlemleri de aynı sembollerle ifade edildiğinden bu semboller kafamızı karıştırabilir. Bu özel sembollerin seçilmesinin nedeni, (Boole cebri önerme (postulate)'lerine göre) klasik toplama ve çarpmaya oldukça benzemeleridir. Bazı metinler +'yı

göstermek üzere  $\cup$  (birleşim) veya  $\vee$ ; ve noktayı göstermek üzere de  $\cap$  (kesişim) veya  $\wedge$  kullanmaktadır.

İzin verilen üçüncü işlem, deęilleme veya tümleme'dir. Deęişkenin veya terimin üzerine konan bir çizgi ile temsil edilir. Sıfırın tümleyeni (yani  $\bar{0}$ ), 1 ve birinin tümleyeni (yani  $\bar{1}$ ) ise 0'dır.  $X+Y\bar{Z}=A$  ifadesi, 'X veya Y ve Z deęil eşittir A' olarak okunur. Bir VE operatörünün öncelięi, VEYA operatöründen yüksek olup, bu ifade  $A=X+(Y\bar{Z})$  olup,  $A=(X+Y)\bar{Z}$  deęildir. Dikkat ediniz, bu eşitlik okunduęunda, ne anlama geldięi pek anlaşılamamaktadır. Çıkarma ve bölme işlemlerinin Boole cebrinde bulunmadıęına dikkat edilmelidir.

+, . ve DEĞİL işlemlerinin sonuçları doğruluk tablolarıyla (Tablo 2.10) gösterilebilir. Bu kurallar herhangi bir sayıdaki deęişkene uygulanabilir.

Tablo 2.10 Sabitler üzerine Boole işlemleri

| DEĞİL         | VE      | VEYA    |
|---------------|---------|---------|
| $\bar{0} = 1$ | $0.0=0$ | $0+0=0$ |
| $\bar{1} = 0$ | $0.1=0$ | $0+1=1$ |
|               | $1.0=0$ | $1+0=1$ |
|               | $1.1=1$ | $1+1=1$ |

Boole deęişkenleri, aşıęıda listelendięi üzere klasik cebirin, deęişme, daęılma ve birleşme kurallarına uyarlar.

$$A + B = B + A$$

$$A \cdot B = B \cdot A$$

VE ve VEYA operatörleri deęişme özellięine sahiptir. Yani işlemcileri sıra deęişmez olup, deęişkenlerin sırası, toplam veya çarpım grubunda önem taşımamaktadır.

$$A.(B+C)=AB+AC$$

$$A+BC=(A+B)(A+C)$$

VE operatörü çarpma gibi, VEYA operatörü ise toplama gibi davranır. Daęılma özellięi, VE ve VEYA operatörlerinin her ikisini de içeren bir eşitlikte, VE operatörün VEYA operatörüne göre öncelięi olduęunu belirtmektedir.

$$A.(B.C)=(A.B).C$$

$$A+(B.C)=(A+B).C$$

VE ve VEYA operatörleri birleşik olduęundan ara toplamaların hesaplanma sırası önemli deęildir.

Boole cebrinin temel kuralları aşıęıda verilmiştir.

$$0.X = 0 \quad 1.X = X \quad X.X = X \quad X.\bar{X} = 0$$

$$0+X = X \quad 1+X = 1 \quad X+X = X \quad X+\bar{X} = 1$$

$$\bar{\bar{X}} = X$$

Bu eşitlikler, X yerine olası deęerler (yani 0 veya 1) koyarak ispatlanabilirler. Örneęin  $0.X=0$ 'ı ele alalım. Eęer  $X=1$  ise  $0.1=0$  elde edilir ki bu doğrudur.  $X=0$  alınırsa,  $0.0=0$  olup bu da doğrudur. Böylece  $0.X=0$  tüm X deęerleri için sağlamış olur.

İfadelerin basitleştirilmesine yardımcı olmak üzere, yukarıdaki kuralları kullanarak bazı teoremler türetebiliriz:

$$\text{Teorem 1} \quad X+XY=X$$

$$\text{İspat: } X+XY = X(1+Y) = X(1) = X$$

$$\text{Teorem 2} \quad X+\bar{X}Y = X+Y$$

$$\begin{aligned} \text{İspat: } X+\bar{X}Y &= (X+XY)+\bar{X}Y \quad (\text{Teorem 1'den}) \\ &= X+XY+\bar{X}Y \\ &= X+Y(X+\bar{X}) \\ &= X+Y(1) = X+Y \end{aligned}$$

$$\text{Teorem 3} \quad XY+\bar{X}Z+YZ = XY+\bar{X}Z$$

$$\begin{aligned} \text{İspat: } XY+\bar{X}Z+YZ &= XY+\bar{X}Z+YZ(X+\bar{X}) \quad (\text{hatırlatma } (X+\bar{X})=1) \\ &= XY+\bar{X}Z+XYZ+\bar{X}YZ \\ &= XY(1+Z)+\bar{X}Z(1+Y) \\ &= XY(1)+\bar{X}Z(1) \\ &= XY+\bar{X}Z. \end{aligned}$$

$$\text{Teorem 4} \quad X(X+Y) = X.$$

$$\begin{aligned} \text{İspat: } X(X+Y) &= XX+XY \\ &= X+XY \\ &= X(1+Y) \\ &= X(1) \\ &= X. \end{aligned}$$

$$\text{Teorem 5} \quad X(\bar{X}+Y) = XY.$$

$$\text{İspat: } X(\bar{X}+Y) = X\bar{X}+XY = XY.$$

$$\text{Teorem 6} \quad (X+Y)(X+\bar{Y}) = X.$$

$$\begin{aligned} (X+Y)(X+\bar{Y}) &= XX+X\bar{Y}+XY+Y\bar{Y} \\ &= X+X\bar{Y}+XY \end{aligned}$$

$$= X(1 + \bar{Y} + Y)$$

$$= X.$$

Teorem 7  $(X+Y)(\bar{X}+Z) = XZ + \bar{X}Y$ .

$$\text{İspat: } (X+Y)(\bar{X}+Z) = X\bar{X} + XZ + \bar{X}Y + YZ$$

$$= XZ + \bar{X}Y + YZ$$

$$= XZ + \bar{X}Y \quad (\text{Teorem 3'den})$$

Teorem 8  $(X+Y)(\bar{X}+Z)(Y+Z) = (X+Y)(\bar{X}+Z)$ .

$$\text{İspat: } (X+Y)(\bar{X}+Z)(Y+Z) = (XZ + \bar{X}Y)(Y+Z) \quad (\text{Teorem 7'den})$$

$$= XYZ + XZZ + \bar{X}YY + \bar{X}YZ$$

$$= XZ(Y+1) + \bar{X}Y(1+Z)$$

$$= XZ + \bar{X}Y$$

$$= (X+Y)(\bar{X}+Z). \quad (\text{Teorem 7'den})$$

Teorem 8 için alternatif bir ispat daha sonra verilecektir

Teorem 9  $\bar{X} \cdot Y \cdot Z = \bar{X} + \bar{Y} + \bar{Z}$ .

Teorem 10  $\overline{X+Y+Z} = \bar{X} \cdot \bar{Y} \cdot \bar{Z}$ .

Teorem 9 ve Teorem 10, De Morgan teoremleri adını alırlar.

#### Gözlemler

Aşağıdaki gözlemler, en çok rastlanan hatalar olmaları nedeniyle burada belirtilmiştir.

Gözlem 1:  $XY + \bar{X}\bar{Y}$  1'e eşit değildir.  
 $XY + \bar{X}\bar{Y}$  basitleştirilemez

Gözlem 2:  $\bar{X}Y + X\bar{Y}$  1'e eşit değildir.  
 $\bar{X}Y + X\bar{Y}$  basitleştirilemez

Gözlem 3:  $\bar{X}\bar{Y} \cdot \bar{X}\bar{Y}$ 'ye eşit değildir.

Gözlem 4:  $\overline{X+Y} \cdot \overline{X+Y}$ 'ye eşit değildir.

#### Boole Cebri'nin kullanımına ilişkin bazı örnekler

Boole Cebri'nin temel kuralları görüldükten sonra, şimdi bu kuralların Boole ifadelerinin basitleştirilmesinde nasıl kullanılacağını göstereceğiz. Bu eşitlikler

'Şapka'dan çıkarılmamış' olup, sıklıkla, bir özel mantık devresinin tanımlarından türetilmiştir. Bu eşitliklerin basitleştirilmeleriyle, mantık devresinin daha ucuz bir tipini üretmek mümkün olabilir.

(d) Aşağıdaki ifadeleri basitleştirin.

- $X + \bar{Y} + \bar{X}Y + (X + \bar{Y})\bar{X}Y$
- $\bar{X}YZ + \bar{X}YZ + X\bar{Y}Z + XYZ$
- $\bar{X}Y \cdot XZ$
- $(X + \bar{Y})(\bar{X} + Z)(Y + \bar{Z})$
- $(W + X + YZ)(\bar{W} + X)(\bar{X} + Y)$
- $WX\bar{Z} + \bar{X}YZ + WX\bar{Y} + XYZ + \bar{W}YZ$
- $\bar{W}XZ + WZ + XY\bar{Z} + \bar{W}XY$
- $(X + Y + Z)(\bar{X} + Y + Z)(\bar{X} + Y + \bar{Z})$

#### Çözümler

Not: Boole ifadelerini basitleştirirken, değişkenleri alfabetik sırada tutmaya özen gösterdim. Bu, mantıksal grupların kolayca ayırt edilebilmelerini sağlamaktadır.

$$(a) X + \bar{Y} + \bar{X}Y + (X + \bar{Y})\bar{X}Y = X + \bar{Y} + \bar{X}Y + X\bar{X}Y + \bar{X}\bar{Y}Y$$

$$= X + \bar{Y} + \bar{X}Y \quad (A\bar{A} = 0 \text{ olduğundan})$$

$$= X + \bar{Y} + \bar{X} \quad (A + \bar{A}B = A + B \text{ olduğundan})$$

$$= 1 \quad (A + \bar{A} = 1 \text{ olduğundan})$$

Not: Bir Boole ifadesi 1'e indirgendiğinde, ifade daima doğru olup değişkenlerden yani X, Y'den bağımsızdır.

$$(b) \bar{X}YZ + \bar{X}YZ + X\bar{Y}Z + XYZ = \bar{X}Y(\bar{Z} + Z) + XZ(\bar{Y} + Y)$$

$$= \bar{X}Y(1) + XZ(1)$$

$$= \bar{X}Y + XZ.$$

$$(c) \bar{X}Y \cdot XZ = \bar{X}Y + \bar{X}Z \quad (\text{Teorem 9'dan})$$

$$= \bar{X}Y + XZ \quad \bar{F} = F \text{ olduğundan}$$

Not: (b) ve (c) örneklerindeki her iki ifade  $\bar{X}Y + XZ$  ifadesine basitleştirildiğinden birbirine eşdeğerdir. Bu eşitlikler, çoğullayıcıya ait olup, (b) doğruluk tablosundan, (c) ise Şekil 2.9'daki devre diyagramından türetilmiştir.

$$\begin{aligned}
 (d) (X+Y)(\bar{X}+Z)(Y+\bar{Z}) &= (X\bar{X}+XZ+\bar{X}\bar{Y}+\bar{Y}Z)(Y+\bar{Z}) \\
 &= (XZ+\bar{X}\bar{Y}+\bar{Y}Z)(Y+\bar{Z}) = (XZ+\bar{X}\bar{Y})(Y+\bar{Z}) \\
 &= XYZ+XZ\bar{Z}+\bar{X}\bar{Y}Y+\bar{X}\bar{Y}\bar{Z} \\
 &= XYZ+\bar{X}\bar{Y}\bar{Z}.
 \end{aligned}$$

$$\begin{aligned}
 (e) (W+X+YZ)(\bar{W}+X)(\bar{X}+Y) &= (W\bar{W}+\bar{W}X+\bar{W}YZ+WX+XX \\
 &\quad +XYZ)(\bar{X}+Y) \\
 &= (\bar{W}X+\bar{W}YZ+WX+X \\
 &\quad +XYZ)(\bar{X}+Y) \\
 &= (X+\bar{W}YZ)(\bar{X}+Y) \\
 &= X\bar{X}+XY+\bar{W}XYZ+\bar{W}YYZ \\
 &= XY+\bar{W}XYZ+\bar{W}YZ \\
 &= XY+\bar{W}YZ(X+1) \\
 &= XY+\bar{W}YZ.
 \end{aligned}$$

Nor:  $(\bar{W}+X)(\bar{X}+Y)$  nın  $(A+B)(\bar{A}+C)=AC+\bar{A}B$  şeklinde bir ifade olduğunu düşünseydik, yukarıdaki işlemler dizisi kısaltılabilirdi. Bu satırlara devam ederek.

$$\begin{aligned}
 (W+X+YZ)(\bar{W}+X)(\bar{X}+Y) &= (W+X+YZ)(XY+\bar{W}\bar{X}) \\
 &= WXY+W\bar{W}\bar{X}+XXY+\bar{W}\bar{X}X \\
 &\quad +XYYZ+\bar{W}\bar{X}YZ \\
 &= WXY+XY+XYZ+\bar{W}\bar{X}YZ \\
 &= XY(W+1+Z)+\bar{W}\bar{X}YZ \\
 &= XY+\bar{W}\bar{X}YZ \\
 &= XY+\bar{W}YZ. \quad \text{elde edilir.}
 \end{aligned}$$

$$\begin{aligned}
 (f) WX\bar{Z}+\bar{X}YZ+WX\bar{Y}+XYZ+\bar{W}YZ \\
 &= WX\bar{Z}+YZ(\bar{X}+X+\bar{W})+WX\bar{Y} \\
 &= WX\bar{Z}+YZ+WX\bar{Y} \\
 &= WX(\bar{Y}+\bar{Z})+YZ \\
 &(A+\bar{A}B=A+B \text{ olduğundan } YZ=\bar{\bar{Y}}+\bar{\bar{Z}} \text{ yazabiliriz}) \\
 &= WX+YZ.
 \end{aligned}$$

$$\begin{aligned}
 (g) \bar{W}XZ+WZ+XYZ+\bar{W}XY &= Z(\bar{W}X+W)+XYZ+\bar{W}XY \\
 &= Z(X+W)+XYZ+\bar{W}XY \\
 &= XZ+WZ+XYZ+\bar{W}XY \\
 &= X(Z+Y\bar{Z})+WZ+\bar{W}XY \\
 &= X(Z+Y)+WZ+\bar{W}XY \\
 &= XZ+XY+WZ+\bar{W}XY \\
 &= XZ+XY(1+\bar{W})+WZ \\
 &= XZ+XY+WZ.
 \end{aligned}$$

$$\begin{aligned}
 (h) (X+Y+Z)(\bar{X}+Y+Z)(\bar{X}+Y+\bar{Z}) \\
 &= (Y+Z)(\bar{X}+Y+\bar{Z}) \\
 &= ((A+B)(A+\bar{B})=A \text{ olduğundan}) \\
 &= Z(\bar{X}+Y)+\bar{Z}Y \\
 &= ((A+B)(\bar{A}+C)=AC+\bar{A}B \text{ olduğundan}) \\
 &= \bar{X}Z+YZ+\bar{Z}Y \\
 &= \bar{X}Z+Y(Z+\bar{Z}) \\
 &= \bar{X}Z+Y.
 \end{aligned}$$

### De Morgan Teoreminin Uygulanması

Bir VE işlevinin, bir VEYA ile birlikte bir tersleyici tarafından gerçekleştirilmesine olanak tanıdığı için, 9 ve 10. teoremler tasarımcıya son derece güçlü bir gereç vermektedir. Mantık işlevleri, yalnızca VEDEĞİL (NAND) ve VEYADEĞİL (NOR) ile gerçekleştirilmeleri durumunda, bunun çok büyük bir önemi vardır. Tüm fonksiyonun tümleyeni alındığında, DE MORGAN teoremi VE'lerin VEYA'lara VEYA'ların VE'lere dönüşeceğini öngörmektedir. Değişkenlerin (ve harfle gösterilen herhangi diğer terimlerin) de tümleyenleri alınabilir. Burada dikkat edilecek nokta, değişkenlerin aynı zamanda, değişken grupları anlamına da geldiğidir.

Okuyucu; devreleri, niçin yalnızca VEDEĞİL (veya VEYADEĞİL) kullanarak, gerçekleştirmek istediğimizi merak edebilir. Bunun için birkaç neden bulunmaktadır, fakat genelde VEDEĞİL devreleri VE kapılarından daha hızlı çalışmakta olup, daha az sayıda bileşenle oluşturulabilirler. İleride, bir devrenin, nasıl tümüyle VEDEĞİL kapılarından tasarımlanabileceğini ayrıntılı bir şekilde inceleyeceğiz. Aşağıdaki örnekler DE MORGAN teoreminin uygulanmasını göstermektedir.

$$\begin{aligned}
 (1) F &= \overline{XY+XZ} & \text{'+' , '.' biçimine dönüşür ve, iki değişken grubunun} \\
 &= \bar{X}\bar{Y} \cdot \bar{X}\bar{Z} & \text{tümleyeni alınır.} \\
 &= (\bar{X}+\bar{Y})(\bar{X}+\bar{Z}). & \text{İşlemler, iki tümlenmiş gruba De Morgan kuralının} \\
 & & \text{uygulanmasıyla devam eder.}
 \end{aligned}$$

$$\begin{aligned}
 (2) F &= \overline{AB+CD+AD} \\
 &= \bar{A}\bar{B} \cdot \bar{C}\bar{D} \cdot \bar{A}\bar{D} \\
 &= (\bar{A}+\bar{B})(\bar{C}+\bar{D})(\bar{A}+\bar{D})
 \end{aligned}$$

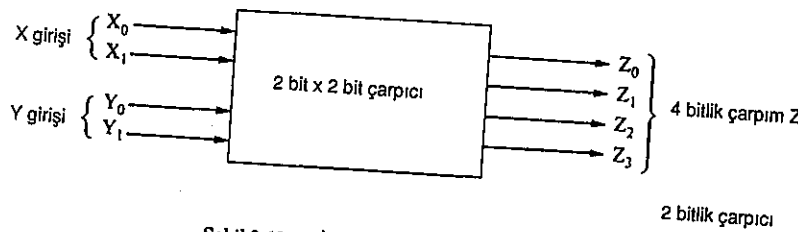
$$\begin{aligned}
 (3) F &= \overline{AB(C+BD)} \\
 &= \bar{A}+\bar{B}+\bar{C}+\bar{B}\bar{D} \\
 &= \bar{A}+\bar{B}+\bar{C}.\bar{B}\bar{D} \\
 &= \bar{A}+\bar{B}+\bar{C}(\bar{B}+\bar{D}).
 \end{aligned}$$

(4) Teorem 8'in, De Morgan teoremiyle ispatı

$$\begin{aligned}
 (X+Y)(\bar{X}+Z)(Y+Z) &= (X+Y)(\bar{X}+Z)(Y+Z) \\
 &= \overline{\overline{X+Y} \cdot \overline{\bar{X}+Z} \cdot \overline{Y+Z}} \\
 &= \overline{\bar{X} \cdot \bar{Y} \cdot \bar{X} \cdot \bar{Z} \cdot \bar{Y} \cdot \bar{Z}} \\
 &= \overline{\bar{X} \bar{Y} \bar{X} \bar{Z}} \quad (\text{Teorem 3'den}) \\
 &= \overline{\bar{X} \bar{Y} \bar{X} \bar{Z}} = (X+Y)(\bar{X}+Z)
 \end{aligned}$$

İki-bitlik bir çarpma devresinin tasarımı

Şimdiye kadar incelenen örnekler, gerçek uygulamadan çok test kitabı soruları olup, Boole cebri için bir soru-bankası oluşturmaktaydı. Aşağıdaki örnek, Boole Cebirinin nasıl bir uygulamalı probleme uygulanabileceğini göstermektedir. Var sayalım ki tasarımcı, bir 2 bit x 2 bit çarpan bir devre üretmek istesin. henüz ikili aritmetiğe gelmedik (Bkz. konu 3), fakat burada bir zorluk bulunmamaktadır. Sistemin blok diyagramını (Şekil 2.17 (a)) ele alarak ve doğruluk tablosunu oluşturarak (Tablo 2.10) işe başlayalım.



Şekil 2.17 (a) İki-bitlik çarpıcı

Tablo 2.10 2 bit x 2 bit çarpıcı için doğruluk tablosu

|       | Girişler       |                |                |                | Çıkışlar       |                |                |                |
|-------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
|       | X              |                | Y              |                | Z              |                |                |                |
| XxY-Z | X <sub>1</sub> | X <sub>0</sub> | Y <sub>1</sub> | Y <sub>0</sub> | Z <sub>3</sub> | Z <sub>2</sub> | Z <sub>1</sub> | Z <sub>0</sub> |
| 0x0-0 | 0              | 0              | 0              | 0              | 0              | 0              | 0              | 0              |
| 0x1-0 | 0              | 0              | 0              | 1              | 0              | 0              | 0              | 0              |
| 0x2-0 | 0              | 0              | 1              | 0              | 0              | 0              | 0              | 0              |
| 0x3-0 | 0              | 0              | 1              | 1              | 0              | 0              | 0              | 0              |
| 1x0-0 | 0              | 1              | 0              | 0              | 0              | 0              | 0              | 0              |
| 1x1-1 | 0              | 1              | 0              | 1              | 0              | 0              | 0              | 0              |
| 1x2-2 | 0              | 1              | 1              | 0              | 0              | 0              | 0              | 1              |
| 1x3-3 | 0              | 1              | 1              | 1              | 0              | 0              | 1              | 0              |
| 2x0-0 | 1              | 0              | 0              | 0              | 0              | 0              | 0              | 1              |
| 2x1-2 | 1              | 0              | 0              | 1              | 0              | 0              | 0              | 0              |
| 2x2-4 | 1              | 0              | 1              | 0              | 0              | 1              | 0              | 0              |
| 2x3-6 | 1              | 0              | 1              | 1              | 0              | 1              | 0              | 0              |
| 3x0-0 | 1              | 1              | 0              | 0              | 0              | 0              | 1              | 0              |
| 3x1-3 | 1              | 1              | 0              | 1              | 0              | 0              | 1              | 0              |
| 3x2-6 | 1              | 1              | 1              | 0              | 0              | 1              | 1              | 0              |
| 3x3-9 | 1              | 1              | 1              | 1              | 1              | 0              | 0              | 1              |

Dört giriş (doğruluk tablosundaki 16 satırı gösterir) ve dört çıkış bulunmaktadır. Her 4-bit, iki bitlik iki sayının çarpımını gösterdiğinden, örneğin 1100, 11x10 veya 3x2 yi göstermektedir. Buna karşılık gelen çıkış, 4-bitlik çarpım olup, buradaki örnekte 6 veya ikili biçimde 0110 dir.

Tablo 2.10'dan, Z<sub>0</sub> dan Z<sub>3</sub>'e ifadeler türetebiliriz. Dikkat ediniz ki, doğruluk tablosunda m tane çıkış sütunu bulunduğundan, m Boole eşitlik grubu türetilmelidir. Bir eşitlik m sütunun herbiriyle bağlantılıdır. Z<sub>0</sub> için bir eşitlik türetmek için, Z<sub>0</sub> sütununda dört minterm mantıksal olarak VEYA'lanır.

$$\begin{aligned}
 Z_0 &= \bar{X}_1 \bar{X}_0 \bar{Y}_1 \bar{Y}_0 + \bar{X}_1 \bar{X}_0 Y_1 \bar{Y}_0 + X_1 \bar{X}_0 \bar{Y}_1 \bar{Y}_0 + X_1 \bar{X}_0 Y_1 \bar{Y}_0 \\
 &= \bar{X}_1 \bar{X}_0 Y_0 (\bar{Y}_1 + Y_1) + X_1 \bar{X}_0 Y_0 (\bar{Y}_1 + Y_1) \\
 &= \bar{X}_1 \bar{X}_0 Y_0 + X_1 \bar{X}_0 Y_0 \\
 &= X_0 Y_0 (\bar{X}_1 + X_1) \\
 &= X_0 Y_0
 \end{aligned}$$

$$\begin{aligned}
 Z_1 &= \bar{X}_1 \bar{X}_0 Y_1 \bar{Y}_0 + \bar{X}_1 \bar{X}_0 Y_1 Y_0 + X_1 \bar{X}_0 \bar{Y}_1 \bar{Y}_0 + X_1 \bar{X}_0 Y_1 \bar{Y}_0 \\
 &\quad + X_1 \bar{X}_0 \bar{Y}_1 Y_0 + X_1 \bar{X}_0 Y_1 Y_0 \\
 &= \bar{X}_1 \bar{X}_0 Y_1 (\bar{Y}_0 + Y_0) + X_1 \bar{X}_0 Y_0 (\bar{Y}_1 + Y_1) + X_1 \bar{X}_0 \bar{Y}_1 Y_0 + X_1 \bar{X}_0 Y_1 \bar{Y}_0 \\
 &= \bar{X}_1 \bar{X}_0 Y_1 + X_1 \bar{X}_0 Y_0 + X_1 \bar{X}_0 \bar{Y}_1 Y_0 + X_1 \bar{X}_0 Y_1 \bar{Y}_0 \\
 &= X_0 Y_1 (\bar{X}_1 + X_1 \bar{Y}_0) + X_1 Y_0 (\bar{X}_0 + X_0 \bar{Y}_1) \\
 &= X_0 Y_1 (\bar{X}_1 + \bar{Y}_0) + X_1 Y_0 (\bar{X}_0 + \bar{Y}_1) \\
 &= \bar{X}_1 X_0 Y_1 + X_0 Y_1 \bar{Y}_0 + X_1 \bar{X}_0 Y_0 + X_1 \bar{Y}_1 Y_0
 \end{aligned}$$

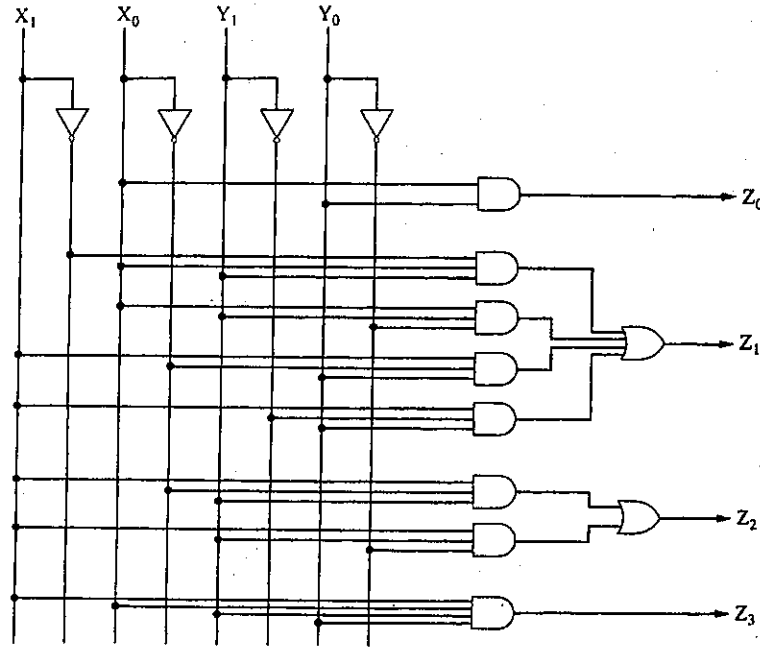
$$\begin{aligned}
 Z_2 &= X_1 \bar{X}_0 Y_1 \bar{Y}_0 + X_1 \bar{X}_0 Y_1 Y_0 + X_1 X_0 Y_1 \bar{Y}_0 \\
 &= X_1 \bar{X}_0 Y_1 (\bar{Y}_0 + Y_0) + X_1 X_0 Y_1 \bar{Y}_0 \\
 &= X_1 \bar{X}_0 Y_1 + X_1 X_0 Y_1 \bar{Y}_0 \\
 &= X_1 Y_1 (\bar{X}_0 + X_0 \bar{Y}_0) \\
 &= X_1 Y_1 (\bar{X}_0 + \bar{Y}_0) \\
 &= X_1 \bar{X}_0 Y_1 + X_1 Y_1 \bar{Y}_0
 \end{aligned}$$

$$Z_3 = X_1 X_0 Y_1 Y_0$$

Şimdi elimizde, Z<sub>0</sub>, Z<sub>1</sub>, Z<sub>2</sub> ve Z<sub>3</sub> için basitleştirilmiş ifadeler bulunmaktadır.

$$\begin{aligned}
 \text{Yani } Z_0 &= X_0 Y_0, \\
 Z_1 &= \bar{X}_1 \bar{X}_0 Y_1 + X_0 Y_1 \bar{Y}_0 + X_1 \bar{X}_0 Y_0 + X_1 \bar{Y}_1 Y_0, \\
 Z_2 &= X_1 \bar{X}_0 Y_1 + X_1 Y_1 \bar{Y}_0, \\
 \text{ve } Z_3 &= X_1 X_0 Y_1 Y_0.
 \end{aligned}$$

Buradaki ifadeler X ve Y'e göre simetriktir. Bunun böyle olması beklenmelidir. Eğer problemin kendisi de X ve Y'e göre simetrikse (yani  $3 \times 1 = 1 \times 3$  ise) sonucun da bu simetriyi göstermesi gerekir.  $Z_0-Z_3$ ' için, ifadeleri gerçekleştirmenin birçok yolu vardır. Şekil 2.17 (b), olası bir yolu göstermektedir.



Şekil 2.17 (b) İki bit çarpıcı için olası bir devre

### Yalnız VEDEĞİL kapıları kullanarak mantık işlevlerinin gerçekleştirilmesi

Belirtildiği üzere, bazı kapılar, özde diğerlerinden daha iyidir. VEDEĞİL kapısı, buna karşılık gelen VE kapısına göre hem daha hızlı, hem daha ucuzdur. Aynı konu NOR VEYADEĞİL ve VEYA kapıları için de doğrudur. Sonuç olarak, bazen devrenin tek tip kapı kullanılarak gerçekleştirilmesi gerekebilir. Bu, kapıların ucuz olup, piyasada bolca bulunması nedeniyle sık sık karşılaşılan bir durumdur. Dikkat edilmelidir ki, tasarımcının kullanımına açık olan kapı çeşitlerinin yapısı aynı değildir. Örneğin, dört adet iki girişi olandan, onüç girişi olana kadar birçok tip VEDEĞİL kapısı bulunmakla birlikte sadece birkaç tip VE kapısı bulunmaktadır. Sadece VEDEĞİL kapılarından oluşmuş bir devre gerçekleştirmek için, De Morgan Teoremi bir '+'ı, '-'ya çevirecek şekilde kullanılmalıdır. Örneğin eğer  $F = A + B + C$  ifadesini, sadece VEDEĞİL kapıları kullanarak üretmek istesek, aşağıdaki işlemleri yaparız:

$$F = A + B + C$$

$$F = \overline{\overline{F}} = \overline{\overline{A + B + C}}$$

$$F = \overline{A \cdot B \cdot C}$$

Çift eğilleme işlevin değeri üzerinde bir etkisi yoktur.  
De Morgan Teoremini uygulayın

Şimdi, VEYA işlevini VEDEĞİL işlevine çevirmiş bulunuyoruz. Üç DEĞİL işlevi, DEĞİL kapıları veya girişleri birbirine bağlı iki girişli VEDEĞİL kapılarıyla gerçekleştirilebilir. Eğer VEDEĞİL kapısının girişleri A ve B ve çıkışı C ise,  $C = \overline{A \cdot B}$  olur. Fakat  $A = B$  ise  $C = \overline{A \cdot A}$  veya  $C = \overline{A}$  dır. Okuyucu, eğer VEDEĞİL kapısı doğruluk tablosuna başvurup,  $A, B = 0, 1$  ve  $A, B = 1, 0$  satırlarının kaldırılmasının etkilerini değerlendirebilirse, bunu dahi iyi anlayabilecektir. Burada, DE MORGAN teoremini Boole Cebri ifadelerini sadeleştirmek için kullanmadığımızı önemle vurgulamak isterim. Biz burada yalnızca, ifadeleri, VEDEĞİL, (VEYADEĞİL) kapılarına ilişkin terimlerle gerçekleştirilebilecek uygun biçimlere dönüştürmeye, çevirmeye, çalışmaktayız. Gerçekten, son ifade, asıl şekilden çok daha karmaşık olabilir.

Yukarıdaki teknikleri, çarpıcımıza uygulayarak

$$Z_0 = X_0 Y_0 = \overline{\overline{X_0 Y_0}} \text{ (yani VEDEĞİL kapısı ve DEĞİL kapısı = VE kapısı)}$$

$$\begin{aligned} \text{ve } Z_1 &= \overline{\overline{X_1 X_0 Y_1} + \overline{X_0 Y_1 Y_0} + \overline{X_1 X_0 Y_0} + \overline{X_1 Y_1 Y_0}} \\ &= \overline{\overline{X_1 X_0 Y_1} + \overline{X_0 Y_1 Y_0} + \overline{X_1 X_0 Y_0} + \overline{X_1 Y_1 Y_0}} \\ &= \overline{\overline{X_1 X_0 Y_1} \cdot \overline{X_0 Y_1 Y_0} \cdot \overline{X_1 X_0 Y_0} \cdot \overline{X_1 Y_1 Y_0}}, \end{aligned}$$

$$\begin{aligned} Z_2 &= \overline{\overline{X_1 X_0 Y_1} + \overline{X_1 Y_1 Y_0}} \\ &= \overline{\overline{X_1 X_0 Y_1} + \overline{X_1 Y_1 Y_0}} \\ &= \overline{\overline{X_1 X_0 Y_1} \cdot \overline{X_1 Y_1 Y_0}}, \end{aligned}$$

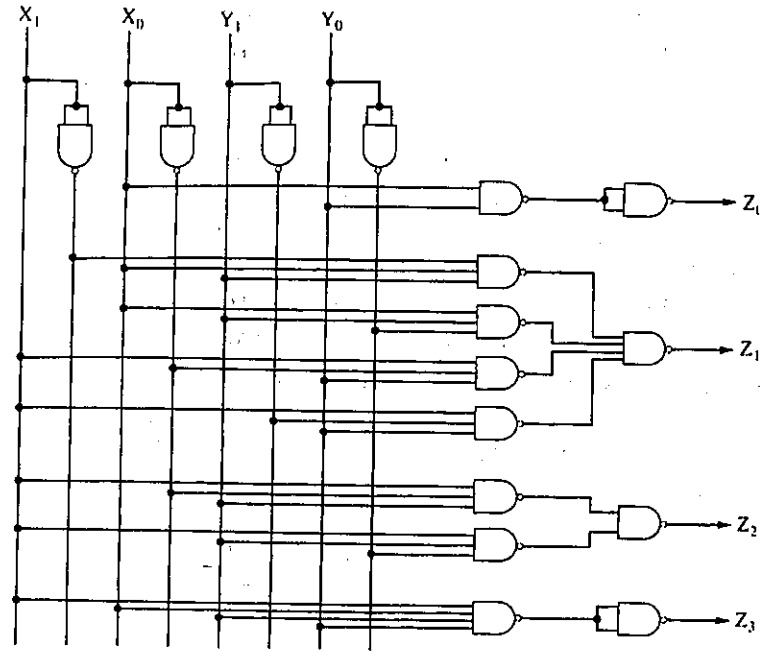
$$\text{ve } Z_3 = \overline{\overline{X_1 X_0 Y_1 Y_0}} = \overline{\overline{X_1 X_0 Y_1 Y_0}}.$$

elde edilir.

Şekil 2.18'de, VEDEĞİL kapıları kullanarak bir çarpıcının nasıl gerçekleştirilebileceği gösterilmiştir. Bu devrenin Şekil 2.17 b'deki devreyle aynı işlevi gerçekleştirdiğine dikkat ediniz.

### Yalnızca VEYADEĞİL kapıları kullanarak mantık işlevlerinin gerçekleştirilmesi

Yukarıda tanımlanan işlemler dizisi, eşit olarak yalnızca Özel-VEYA kapıları



Şekil 2.18 Yalnızca VEDEĞİL (NAND) mantığı kullanarak bir çarpıcunun gerçekleştirilmesi

kullanan devrelerin gerçekleştirilmesine uygulanabilir. Bunu göstermek amacıyla yukarıdaki  $Z_3$  değeri VEYADEĞİL mantık düzenine dönüştürülecektir.

$$\begin{aligned} Z_3 &= X_1 X_0 Y_1 Y_0 \\ &= X_1 X_0 Y_1 Y_0 \\ &= \overline{X_1} + \overline{X_0} + \overline{Y_1} + \overline{Y_0}. \end{aligned}$$

$\overline{X}$ 'nin bir tersleyici veya girişleri birbirine bağlanmış VEYADEĞİL (NOR) kapısıyla gerçekleştirilebileceğine dikkat ediniz.

### 2.3.1 Karnaugh diyagramları

Karnaugh diyagramları veya sadece K diyagramı, Boole ifadelerinin, basitleştirilmesi ve sunulması için kullanılan bir grafik tekniktir.

Karnaugh diyagramları beş değişkenli denklemleri basitleştirebilmesine rağmen üç veya dört değişkenli denklemler için daha uygundur. Bu kitabın içeriğinin dışında kalan bazı teknikler, dört değişkenden fazla olan Boole ifadelerinin basitleştirilmesinde kullanılabilir. Öğrencilerin, Karnaugh diyagramlarını sevdiklerini gördüm,

çünkü Boole ifadeleri bu diyagramlarla kışkırtıya yer vermeyecek biçimde en basit biçimlerine indirgenebilmektedir. Öğrenciler, Boole ifadesini basitleştirmek için cebirsel teknikler kullandıklarında sık sık, basitleştirmenin ve dolayısıyla daha fazla ilerlemenin artık mümkün olmadığı bir noktaya gelmekteydiler. Bununla birlikte onlar, denklemin gerçekte en basit şekilde olup olmadığından emin değiller ve bir sonraki adımı tam olarak görememekteydiler.

Karnaugh diyagramı, Boole ifadelerinin basitleştirilmesinin, diyagramdaki bir'lerin yerlerinden hemen görülebileceği bir şekilde çizilmiş olup tek boyutlu doğruluk tablosunun, iki boyutlu biçimidir. Karnaugh diyagramının anahtarı, düşey ve yatay olarak (çapraz değil) komşu karelere geçişte yalnızca tek bir değişkenin değişmiş olması temeline dayanır.

|   |                |   |
|---|----------------|---|
| A | 0              | 1 |
|   | $\overline{A}$ | A |

(a) Bir değişkenli diyagram

|    |                            |                 |      |                 |
|----|----------------------------|-----------------|------|-----------------|
| AB | 00                         | 01              | 11   | 10              |
|    | $\overline{A}\overline{B}$ | $\overline{A}B$ | $AB$ | $A\overline{B}$ |

(b) İki değişkenli diyagram

|    |                                        |                             |                             |                  |
|----|----------------------------------------|-----------------------------|-----------------------------|------------------|
| AB | 00                                     | 01                          | 11                          | 10               |
| C  |                                        |                             |                             |                  |
| 0  | $\overline{A}\overline{B}\overline{C}$ | $\overline{A}\overline{B}C$ | $\overline{A}B\overline{C}$ | $\overline{A}BC$ |
| 1  | $A\overline{B}\overline{C}$            | $A\overline{B}C$            | $AB\overline{C}$            | $ABC$            |

(c) Üç değişkenli diyagram

|    |                                                    |                                         |                                         |                              |
|----|----------------------------------------------------|-----------------------------------------|-----------------------------------------|------------------------------|
| AB | 00                                                 | 01                                      | 11                                      | 10                           |
| CD |                                                    |                                         |                                         |                              |
| 00 | $\overline{A}\overline{B}\overline{C}\overline{D}$ | $\overline{A}\overline{B}C\overline{D}$ | $\overline{A}B\overline{C}\overline{D}$ | $\overline{A}BC\overline{D}$ |
| 01 | $\overline{A}\overline{B}C\overline{D}$            | $\overline{A}\overline{B}CD$            | $\overline{A}BC\overline{D}$            | $\overline{A}BCD$            |
| 11 | $\overline{A}B\overline{C}\overline{D}$            | $\overline{A}B\overline{C}D$            | $\overline{A}BC\overline{D}$            | $\overline{A}BCD$            |
| 10 | $\overline{A}B\overline{C}\overline{D}$            | $\overline{A}B\overline{C}D$            | $\overline{A}BC\overline{D}$            | $\overline{A}BCD$            |

(d) Dört değişkenli diyagram

Şekil 2.19 Karnaugh diyagramları

Şekil 2.19'da birden-dörde kadar değişkenler için Karnaugh diyagramları gösterilmiştir. Bir ve iki değişkenli diyagramlar, fazla önemi olmayan durumları ifade ettiğinden bundan böyle konu edilmeyeceklerdir. Şimdi üç değişkenli diyagramı ele alalım. Şekil 2.20'de, üç değişkenli fonksiyonun doğruluk tablosu ve buna ilişkin Karnaugh diyagramı gösterilmiştir.

| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

| AB | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| C  | 0  | 1  |    |    |
| 1  |    | 1  |    | 1  |

Şekil 2.20  $F = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}C$  için doğruluk tablosuyla, Karnaugh diyagramı arasındaki ilişki

Doğruluk tablosunda bulunan üç adet bir'den herbiri, Karnaugh diyagramında uygun olan karesine yerleştirilir. Üç değişkenli Karnaugh diyagramında, üç değişkenden ikisinin olası dört değerinden herbiri için bir tane olmak üzere dört sütun vardır. Örneğin, üç değişken A, B ve C ise, dört sütunda A, B'nin alabileceği tüm olası değerler gösterilir. Böylece, ilk (en soldaki) sütun 00 ile etiketli olup, A=0; B=0 durumlarını göstermektedir. Diğer sütun ise 01 ile etiketli olup, A=0 B=1 durumunu göstermektedir. Diğerleri ise 11 olup (10 değil) A=1 B=1 dir. Komşu sütunlar arasındaki geçişlerin yalnızca tek bir değişim gösterdiğini hatırlayın. Dördüncü sütunda ise, 10, A=1 ve B=0 durumu gösterilmiştir.

Tüm üç değişkenli Karnaugh diyagramları iki yatay sıraya sahip olup, üst satır C=0'a, alt satır ise C=1'e ilişkindir. Karnaugh diyagramı üzerindeki herhangi bir kare, üç değişkenin,  $\overline{A}\overline{B}\overline{C}$ 'den ABC'ye kadar olan kombinasyonlarından birini göstermektedir. Tablodaki giriş değerlerinin Karnaugh diyagramına nasıl taşınaca-

ğı Şekil 2.20'de açık olarak gösterilmiştir. Bu noktada belirtmek yerinde olacaktır ki Şekil 2.20'de çizilmiş olan Karnaugh diyagramında, hiçbir bir, birbiriyle komşu olmadığından  $F = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}C$  ifadesi basitleştirilemez. Karnaugh diyagramını olabildiğince temiz ve karışıklıktan uzak tutmak için, bir içermeyen kareler sıfır içerdikleri ve içermeleri gerektiği halde, işaretli olarak (boş) bırakılırlar, bir içeren bir karenin, bir ile örtülü olduğu söylenir. Şimdi  $F_1 = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C}$  ve  $F_2 = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C}$  ifadelerini inceleyelim. (Şekil 2.21)

| AB | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| C  | 0  |    | 1  |    |
| 1  |    |    | 1  |    |

$$F_1 = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C}$$

| AB | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| C  | 0  |    | 1  |    |
| 1  |    |    | 1  |    |

$$F_2 = AB$$

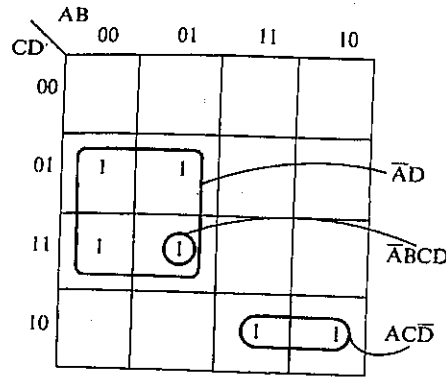
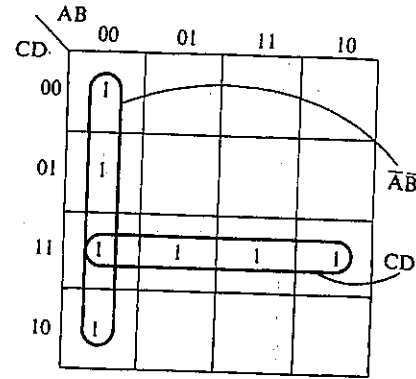
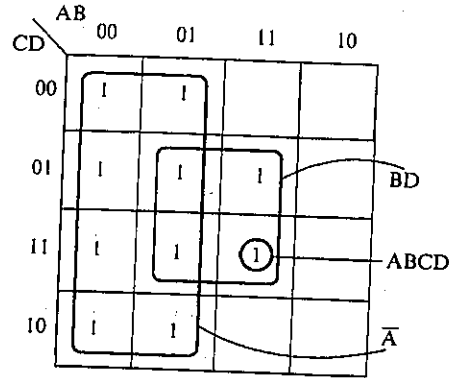
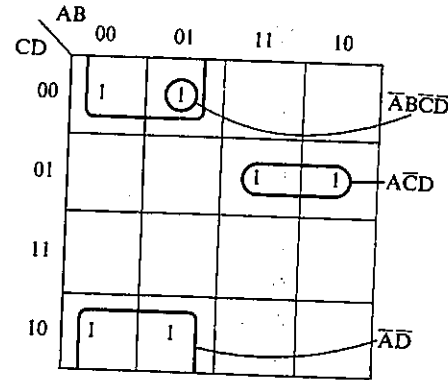
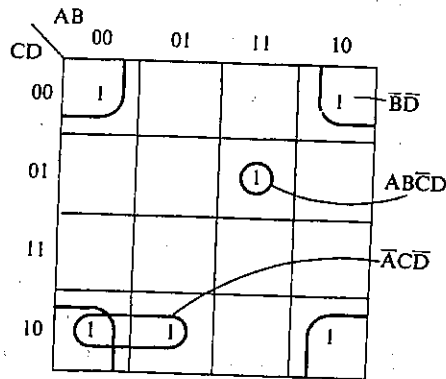
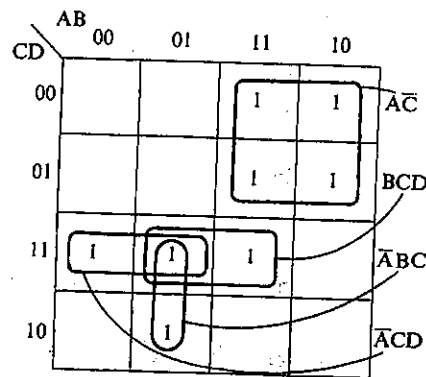
Şekil 2.21 fonksiyonların, Karnaugh diyagramında çizilmesi

$F_1$  için çizilen Karnaugh diyagramında  $\overline{A}\overline{B}\overline{C}$  ve  $\overline{A}B\overline{C}$ 'ye ilişkin olarak iki örtülü kare bulunmaktadır.  $F_2$  için oluşturulan Karnaugh diyagramında, A=1, B=1 sütununun örttüğü iki kare bulunmaktadır.  $F_2$  işlevi içinde C değişkeni bulunmadığından, A=B=1, C=0, A=B=1 ve C=1'a ilişkin karelere bir girilir.

Bu iki Karnaugh diyagramının birbirine özdeş olduğu hemen görülmektedir, bu yüzden  $F_1 = F_2$  ve  $\overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} = \overline{A}\overline{B}\overline{C}$ 'dir. Boole Cebri kurallarından  $\overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} = (\overline{A}\overline{B})(\overline{C} + C) = \overline{A}\overline{B}(1) = \overline{A}\overline{B}$  elde edilir.

Bir Boole cebri ifadesini Karnaugh diyagramlarıyla basitleştirmenin ilk adımı, işlev doğruluk tablosunda tüm 1'leri Karnaugh diyagramına aktarmaktır. İkinci adım, komşu birlerin birerli, ikiyeşli, dörderli, sekizerli veya onaltışarlı gruplar halinde toplanmasıdır. Minterm topluluğu olabildiğince geniş tutulmalıdır. Dört en küçük terimden oluşan basit bir grup, iki minterm oluşan iki gruba göre daha basit bir ifadeye yol açar. Sonuçta gruplar, çarpımlar toplamı ifadesi oluşturmak üzere kullanılırlar. Her grup, çarpım terimlerinden birine ilişkindir. Bu işlemler, en iyi biçimde bir örnek üzerinde açıklanabilir. Aşağıda, bu örneği göstermek üzere dört değişkenli diyagram seçilmiştir.

Şekil 2.22, Karnaugh diyagramına işlenmiş altı işlevi göstermektedir.

(a)  $F = \overline{A}D + ACD + \overline{A}BCD$ (b)  $F = \overline{A}B + CD$ (c)  $F = \overline{A} + BD + ABCD$ (d)  $F = \overline{A}D + ACD + \overline{A}BCD$ (e)  $F = \overline{B}D + \overline{A}BCD + \overline{A}CD$ (f)  $F = \overline{A}CD + \overline{A}BC + BCD + \overline{A}C$ 

Şekil 2.22 İşlevlerin Karnaugh diyagramı üzerinde işlenmesi

Bu diyagramlardaki değişik çarpımların toplamı ifadeleri; doğruluk tablosunun mintermden ziyade, doğrudan eşitliklerin kendisinden türetilerek diyagrama işlenmiştir. Aşağıdaki notlar, bu diyagramların anlaşılmasına yardımcı olacaktır.

### 1. Dört değişkenli diyagram için

- 1 değişkenli çarpım terimleri 8 kare örtmektedir.
- 2 değişkenli çarpım terimleri 4 kare örtmektedir.
- 3 değişkenli çarpım terimleri 2 kare örtmektedir.
- 4 değişkenli çarpım terimleri 1 kare örtmektedir.

2. Bir '1'le örtülen bir kare, çarpımların toplamı ifadesinde birden fazla terime ait olabilir. Örneğin Şekil 2.22 (b)'de minterm  $\overline{A}BCD$ ,  $\overline{A}B$  ve  $CD$  gibi iki gruba aittir. Eğer Karnaugh diyagramında, bir '1' iki grupta birden görülüyorsa bu, diyagramda iki kez işlenen fonksiyon için, ilgili en küçük terimin tüm ifadeye eklenmesine eşdeğerdir. Bu, onun değerini değiştirmez, zira Boole cebirinde  $X+X=X$ 'dir.

3. Karnaugh diyagramı, bu diyagramlarda görüldüğü üzere, kare veya dikdörtgen şeklinde değildir. Karnaugh diyagramları, ya da tor çörek biçimindedir. Yani yukarı kenar aşağı kenara, sol kenar da sağ kenara komşudur. Örneğin Şekil 2.22 d'de  $\overline{A}D$  terimi, diyagramın üst tarafında  $\overline{A}B\overline{C}\overline{D}$  ve  $\overline{A}D\overline{C}\overline{D}$  mintermlerini; diyagramların alt tarafında  $\overline{A}B\overline{C}D$  ve  $\overline{A}D\overline{C}D$  mintermlerini örter. Benzer biçimde Şekil 2.22 (e)'deki,  $CD$  terimi, diyagramın dört köşesini de örtmektedir.

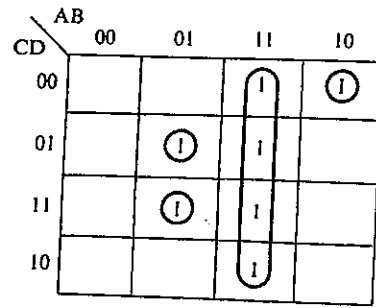
4. Çarpım terimlerini, diyagramdan okumak ya da diyagrama işlemek için bir soruyu sormak gerekir. Bu terim tarafından hangi minterm (kareler) örtülecektir?

Terimlerin Karnaugh diyagramına nasıl işlendiklerini görmüş olduğumuzdan, bir sonraki adımda, bu diyagramları, ifadelerin basitleştirmesinde kullanacağız. Bu işlemler dizisi örnekler halinde gösterilmiştir. Her bir durumda orijinal işlev, sayfanın solunda ve tekrar gruplananlar ise sağ tarafında işlenmiştir.

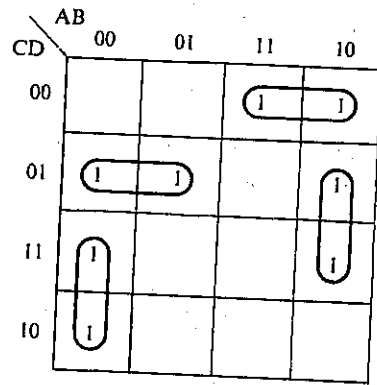
**Örnek 1**  $F = AB + \overline{A}BCD + \overline{A}BCD + \overline{A}B\overline{C}\overline{D}$  (Şekil 2.23). Basitleştirilmiş fonksiyon  $F = AB + BD + \overline{A}C\overline{D}$ 'dir

**Örnek 2**  $F = \overline{A}C\overline{D} + \overline{A}BC + \overline{A}CD + \overline{A}BD$  Şekil (2.24). Bu durumda yalnızca bir tekrar gruplama mümkündür. Basitleştirilmiş işlev  $F = \overline{B}D + \overline{A}C\overline{D} + \overline{A}CD + \overline{A}BC$ 'dir.

**Örnek 3**  $F = \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}B\overline{C}D + \overline{A}B\overline{C}D + \overline{A}B\overline{C}D + \overline{A}B\overline{C}D + \overline{A}B\overline{C}D$



Şekil 2.23. Örnek 1'e ilişkin Karnaugh diyagramı



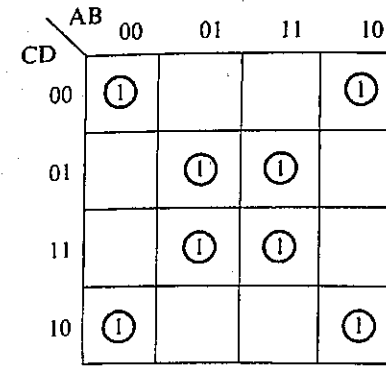
Şekil 2.24. Örnek 2'ye ilişkin Karnaugh diyagramı

+  $\overline{A}\overline{B}\overline{C}\overline{D}$  (Şekil 2.25). İşlev, iki çarpım terimine indirgenebilir. Sonuç  $F = \overline{B}\overline{D} + BD$ 'dir.

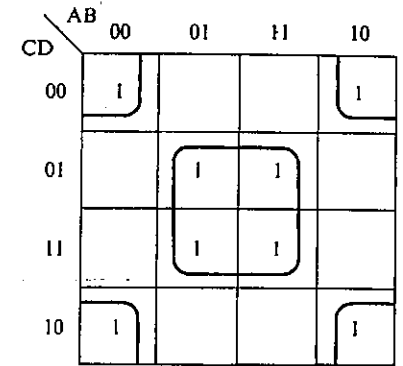
#### Karnaugh diyagramlarının uygulanması

Karnaugh diyagramı, Boole ifadelerinin basitleştirilmesinden başka, çarpımlar toplamının, bir toplamlar çarpımı biçimine dönüştürülmesinde kullanılabilir. Bu işlemler dizisinde ilk adım, ifadenin tümleyeninin üretilmesidir.

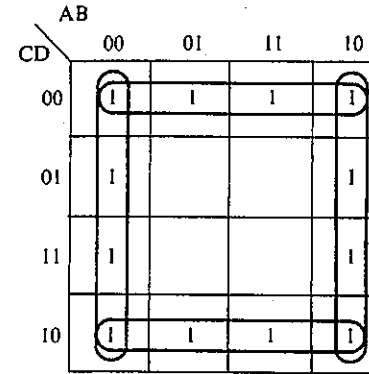
Şekil 2.26'da verilen örnekte,  $F = \overline{C}\overline{D} + \overline{A}\overline{B} + A\overline{B} + C\overline{D}$ 'nin tümleyeninin nasıl bulunduğu gösterilmiştir. Eğer 1'lerle örtülü kareler F'yi gösteriyorsa, 0'la örtülü diğer kareler de  $\overline{F}$ 'ü gösterecektir. Sağ taraftaki diyagramdan  $\overline{F} = BD$  elde edilir.



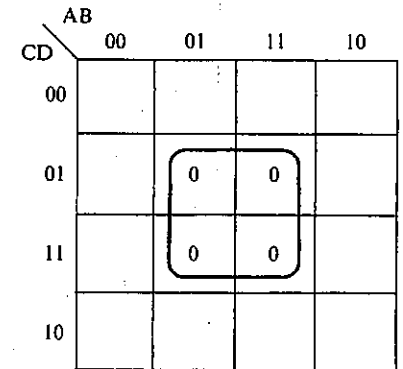
Şekil 2.25. Örnek 3'e ilişkin Karnaugh diyagramı



Çarpımların toplamını, toplamların çarpımına dönüştürmek için; önce çarpımların toplamında F'yi hesaplamak, sonra da bu ifadenin tümleyenini almak gerekir.



$$F = \overline{C}\overline{D} + \overline{A}\overline{B} + A\overline{B} + C\overline{D}$$



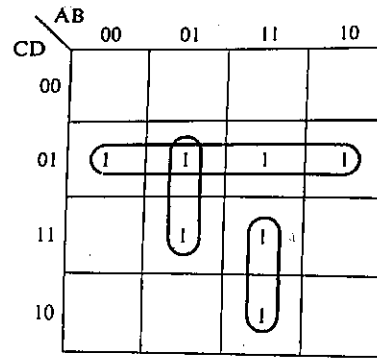
$$\overline{F} = BD$$

Şekil 2.26. Karnaugh diyagramından bir tümleyeninin bulunması

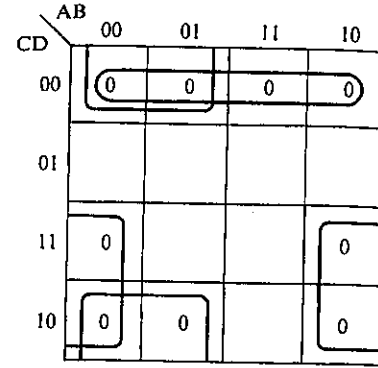
Örnek  $F = ABC + \overline{C}\overline{D} + \overline{A}\overline{B}D$  ifadesini, toplamlar çarpımı biçimine dönüştürünüz (Şekil 2.27). F'nin tümleyeni, sağ taraftaki diyagramdan, aşağıdaki gibi okunabilir.

$$\begin{aligned} \overline{F} &= \overline{C}\overline{D} + \overline{B}\overline{C} + \overline{A}\overline{D} \\ \overline{F} &= \overline{C}\overline{D} + \overline{B}\overline{C} + \overline{A}\overline{D} \\ &= (C+D)(B+\overline{C})(A+D) \text{ elde edilir.} \end{aligned}$$

Dolayısıyla



F'ye ilişkin Karnaugh diyagramı



F-bar'a ilişkin Karnaugh diyagramı

Şekil 2.27

### Karnaugh diyagramı kullanılarak yapılan çözümlü bir örnek

Karnaugh diyagramlarının, Boole cebri ifadelerinin nasıl basitleştirebildiğini gösterdikten sonra, nasıl kullanıldıklarına ilişkin bir örneği inceleyeceğiz. Bir oda yangına karşı, dört adet sensörle korunmaktadır. Bunlar alev dedektörü, duman dedektörü, ve odanın karşılıklı köşelerine yerleştirilmiş iki aşırı sıcaklık dedektörüdür. Bu türde dedektörler, hataya veya sahte alarmlara eğilimli olduklarından, öyle bir düzenleme yapılmıştır ki, yangın alarmı, ancak iki veya daha fazla sensörün aynı anda alarm vermesi durumunda çalışmaya başlar.

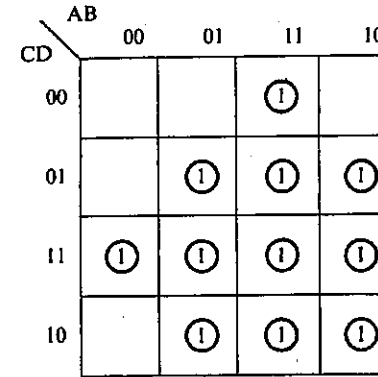
Herhangi bir yangın durumunda sensörün çıkışı mantıksal bir olup, diğer durumlarda mantıksal sıfırdır. Devrenin çıkışı, iki veya daha fazla çıkış bir olduğunda bir verecek biçimde olmalıdır. Bu devre, yalnızca, iki ve üç girişli VEYADEĞİL (NOR) devrelerinden oluşturulacaktır.

Tablo 2.11, bu devrenin doğruluk tablosudur. Sensörlerden gelen dört giriş A,B,C,D olarak etiketlenmiştir. Herhangi bir satırda, iki veya daha fazla mantıksal bir saptanması yeterli olacağından, A,B, C ve D'nin nasıl sıralandığının fazla bir önemi yoktur. Devrenin çıkışı F, doğruluk tablosundan yazılabilir.

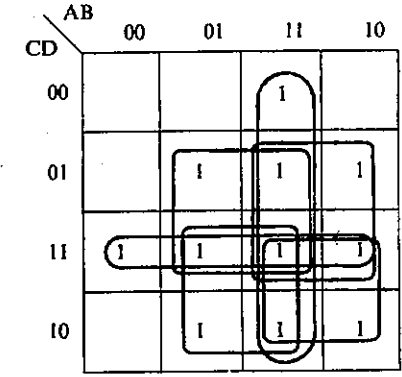
$$F = \bar{A} \bar{B} \bar{C} D + \bar{A} \bar{B} C \bar{D} + \bar{A} B \bar{C} \bar{D} + \bar{A} B C D + A \bar{B} \bar{C} \bar{D} + A \bar{B} C \bar{D} + A \bar{B} C D + A B \bar{C} \bar{D} + A B C \bar{D} + A B C D$$

Tablo 2.11, Yangın dedektörü için doğruluk tablosu

| Girişler |   |   |   | Çıkış |
|----------|---|---|---|-------|
| A        | B | C | D | F     |
| 0        | 0 | 0 | 0 | 0     |
| 0        | 0 | 0 | 1 | 0     |
| 0        | 0 | 1 | 0 | 0     |
| 0        | 0 | 1 | 1 | 1     |
| 0        | 1 | 0 | 0 | 0     |
| 0        | 1 | 0 | 1 | 1     |
| 0        | 1 | 1 | 0 | 1     |
| 0        | 1 | 1 | 1 | 1     |
| 1        | 0 | 0 | 0 | 0     |
| 1        | 0 | 0 | 1 | 1     |
| 1        | 0 | 1 | 0 | 1     |
| 1        | 0 | 1 | 1 | 1     |
| 1        | 1 | 0 | 0 | 1     |
| 1        | 1 | 0 | 1 | 1     |
| 1        | 1 | 1 | 0 | 1     |
| 1        | 1 | 1 | 1 | 1     |



(a)



(b)

Şekil 2.28 Tablo 2.11'e ilişkin Karnaugh diyagramı

Bu terimleri Karnaugh diyagramına işleyerek Şekil 2.28 (a)'yı elde ederiz. Bu terimler 6 adet 4'lü grup halinde toplanabilir, Şekil 2.28 (b). Böylece basitleştirilmiş çarpımlar toplamı,

$$F = AB + AC + AD + BC + BD + CD \quad \text{olarak elde edilir.}$$

İfadeyi, yalnızca VEDEĞİL (NAND) mantığına dönüştürmek için beş adet '+'nın yok edilmesi gerekmektedir.

$$F = \overline{\overline{AB + AC + AD + BC + BD + CD}} \\ = \overline{AB \cdot AC \cdot AD \cdot BC \cdot BD \cdot CD}$$

İfade, istenen biçimde VEDEĞİL mantığı ifadesi haline dönüştürüldüğünde altı-girişli VEDEĞİL kapısını içermektedir. Eğer F ifadesi incelenirse altı terimi VEDEĞİL'lendiği görülecektir, bu altı terimi VE'lemek, ardından, sonucu tersin çevirmekle aynı anlamı taşımaktadır. Boole değişkenlerinin birleşim özelliği nedeniyle,  $X(YZ) = (XY)Z$  yazılabilir. Bunu denklemimize uygulayarak,

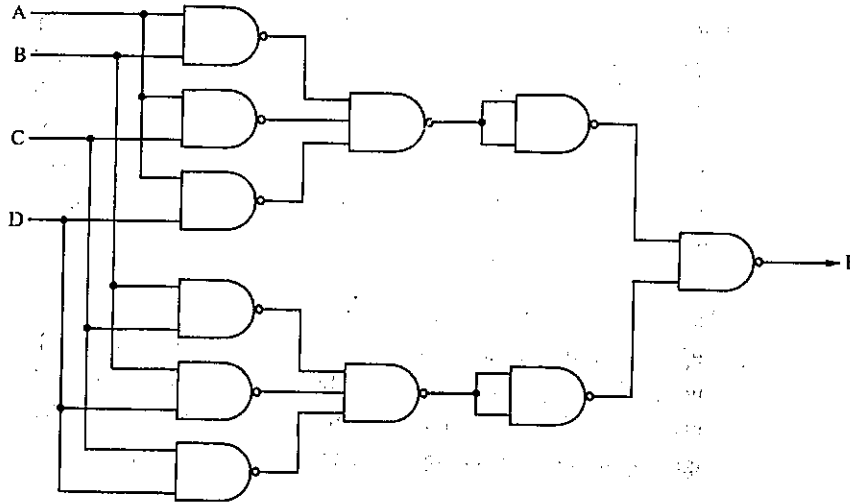
$$F = \overline{\overline{AB \cdot AC \cdot AD \cdot BC \cdot BD \cdot CD}}$$

elde ederiz.

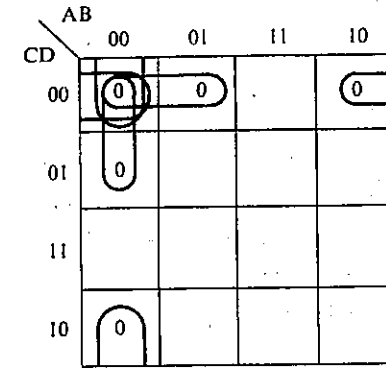
Şekil 2.29'da, bu ifadenin nasıl iki ve üç-girişli VEDEĞİL kapılarıyla gerçekleştirilebileceğini gösterilmiştir.

Bu probleme biraz farklı bir şekilde yaklaşılabılır. Eğer Karnaugh diyagramı incelenirse, birçok karenin 1 ile örtülü olduğu açıkça görülecektir, kalan kareler sıfırla örtülü olup, F ifadesini nispeten basit bir şekilde elde etmek olanaklı olur.

$$\begin{aligned} \overline{F} &= \overline{ABD + ABC + ACD + BCD} \\ \text{ve } F &= \overline{ABD + ABC + ACD + BCD} \\ &= \overline{ABD} \cdot \overline{ABC} \cdot \overline{ACD} \cdot \overline{BCD} \end{aligned}$$



Şekil 2.29 Yangın dedektörü devresinin gerçekleştirilmesi



Şekil 2.30  $\overline{F}$ 'ye ilişkin Karnaugh diyagramı

Bu ifadenin yukarıda olduğu gibi birkaç kapıyla gerçekleştirilip, gerçekleştirilemeyeceğini incelenmesini bir alıştırmaya bırakıyorum.

### Karnaugh diyagramları ve dikkate alınmaz durumlar

Burada, verilen sisteme ait doğruluk tablosunun tamamen belirlenmediği bir grup probleme, Karnaugh diyagramlarının nasıl uygulanacağını göstereceğiz. Yani, belirli giriş koşullarında, çıkış belirsizdir.

Şimdiye kadar karşılaştığımız mantık tasarım problemlerinin hepsinde, belirli bir çıkış değeri, buna ilişkin olası giriş değerleriyle bağlantılıydı. Ender olarak, belirli giriş kombinasyonları olmadan da mevcut olabilen sistemler de olabilir. Böyle durumlarda çıkış, doğru veya yanlış olarak tanımlanabilir. Bütün bunlardan sonra, eğer belirli bir giriş olanaklı değilse buna ilişkin çıkış da anlamlı olmayacaktır. Veya öyle midir? Daha sonra bu 'anlamsız' çıkışları nasıl kullanılabilir şekilde getirebileceğimizi göreceğiz.

'Olanaksız giriş durumları', kavramını biraz daha açıklayabilmek için, aşağıdaki örneği ele alalım. Bir klima sisteminin iki denetim girişi vardır. Biri, C, 10 °C'nin altında doğru, diğer durumlarda yanlış mantığı üreten soğuga-duyarlı termostat; diğeri, H, 25 °C'nin üstünde doğru, diğer durumlarda yanlış mantığı üreten, sığağa-duyarlı termostat aracılığıyla üretilmektedir.

Dört adet giriş olduğundan, Tablo 2.12'de gösterildiği üzere, dört olası mantıksal durum söz konusudur.

Tablo 2.12 termostat doğruluk tablosu

| Girişler |   | Anlamı          |
|----------|---|-----------------|
| C        | H |                 |
| 0        | 0 | Sıcaklık Normal |
| 0        | 1 | Çok sıcak       |
| 1        | 0 | Çok soğuk       |
| 1        | 1 | ?               |

C=1 ve H=1 girişleri, ortam aynı anda hem çok sıcak, hem çok soğuk olamayacağından olanaksızdır. Bu giriş koşulu, ancak termostatın biri arızalanması halinde ortaya çıkabilir. Şimdi, klima için verilen dört girişli, dört çıkışlı örneği inceleyelim.

Tablo 2.13 Denetleyicinin çıkış tablosu

| Giriş | Adı   | Giriş = 0          | Giriş = 1          |
|-------|-------|--------------------|--------------------|
| H     | Sıcak | sıcaklık üst limit | sıcaklık üst limit |
| C     | Soğuk | sıcaklık alt limit | sıcaklık alt limit |
| W     | Nemli | nem üst limit      | nem üst limit      |
| D     | Kuru  | nem alt limit      | nem alt limit      |

Denetleyicinin dört çıkışı vardır; P,Q,R ve S. P=1 iken, ısıtıcı, Q=1 iken de soğutucu devreye girer. Benzer şekilde P=1 ile nemlendirici S=1 ile de nemalıcı çalışır. Her durumda mantık sıfır, uygun cihazı kapatmaktadır. Giriş ve çıkış arasındaki bağıntı aşağıdaki gibidir.

Cihaz; eğer, sıcaklık ve nem belirlenen limitler içindeyse ısıtıcı ve soğutucuyu kapat, aksi belirtilmedikçe nemlendirici ve kurutucunun her ikisi de kapalıdır. Eğer nemlilik, verilen limitler içindeyse ve sıcaklık çok düşükse, ısıtıcıyı; eğer sıcaklık çok yüksekse, soğutucuyu açar.

Eğer sıcaklık, verilen limitler içindeyse ve nem çok düşükse ısıtıcıyı, nem çok yüksekse soğutucuyu açar. Eğer nem yüksek, sıcaklık düşükse ısıtıcıyı, eğer nem düşük ve sıcaklık yüksekse, soğutucuyu açar.

Eğer, sıcaklık ve nemin her ikisi birden yüksekse, soğutucuyu ve kurutucuyu, sıcaklık ve nem çok düşükse, ısıtıcıyı ve nemlendiriciyi açar.

Giriş ve çıkışlar arasındaki bağıntı doğruluk tablosu ile açıklanabilir (Tablo 2.14). P,Q,R,S için 0,1 veya gerektiğinde X yazarak bir Karnaugh diyagram oluşturabiliriz. X'in bir 'dikkate alınmaz' durumu gösterdiğini hatırlayınız. Önce P'yi dikkate alınız.

Tablo 2.14 Denetleyicinin giriş ve çıkışları arasındaki ilişki

| Girişler |   |   |   | Çıkışlar       |   |   |   |   |
|----------|---|---|---|----------------|---|---|---|---|
| H        | C | W | D | Durum          | P | Q | R | S |
| 0        | 0 | 0 | 0 | Normal         | 0 | 0 | 0 | 0 |
| 0        | 0 | 0 | 1 | Kuru           | 1 | 0 | 0 | 0 |
| 0        | 0 | 1 | 0 | Nemli          | 0 | 1 | 0 | 0 |
| 0        | 0 | 1 | 1 | Olanaksız      | X | X | X | X |
| 0        | 1 | 0 | 0 | Soğuk          | 1 | 0 | 0 | 0 |
| 0        | 1 | 0 | 1 | Soğuk ve kuru  | 1 | 0 | 1 | 0 |
| 0        | 1 | 1 | 0 | Soğuk ve nemli | 1 | 0 | 0 | 0 |
| 0        | 1 | 1 | 1 | Olanaksız      | X | X | X | X |
| 1        | 0 | 0 | 0 | Sıcak          | 0 | 1 | 0 | 0 |
| 1        | 0 | 0 | 1 | Sıcak ve kuru  | 0 | 1 | 0 | 0 |
| 1        | 0 | 1 | 0 | Sıcak ve nemli | 0 | 1 | 0 | 1 |
| 1        | 0 | 1 | 1 | Olanaksız      | X | X | X | X |
| 1        | 1 | 0 | 0 | Olanaksız      | X | X | X | X |
| 1        | 1 | 0 | 1 | Olanaksız      | X | X | X | X |
| 1        | 1 | 1 | 0 | Olanaksız      | X | X | X | X |
| 1        | 1 | 1 | 1 | Olanaksız      | X | X | X | X |

Tablo 2.15 P için Karnaugh diyagramı

| WD | HC |    |    |    |
|----|----|----|----|----|
|    | 00 | 01 | 11 | 10 |
| 00 |    | 1  | X  |    |
| 01 | 1  | 1  | X  |    |
| 11 | X  | X  | X  | X  |
| 10 |    | 1  | X  |    |

'Dikkate alınmaz' durumu, bunlara ilişkin çıkış olmadığından bu şekilde adlandırılır. Pek doğaldır ki herhangi bir gerçek birleşik mantık devresinin, 2 girişin herbiri için belirli bir çıkışa sahip olması gerekir. Eğer belli bir giriş 'dikkate alınmaz' olarak nitelendirilmişse, çıkışta ya mantıksal 0 ya da mantıksal 1 üretmelidir. Devre tasarımını basitleştirmek için çıkış 1 veya 0 olarak seçeceğiz. Yani Karnaugh diyagramında, eğer X'in bir olarak alınması, birler grubunu, daha da genişletiyorsa X'i 1; aksi durumlarda ise 0, olarak alacağız.

Aşağıda verilen Karnaugh diyagramı P çıkışına ilişkindir. Gruplamalar için altı, 'dikkate alınmaz' durumu kullanılarak  $P = C + \overline{H}D$  elde edilir. Şekil 2.31 (a)

Q'ya ilişkin basitleştirilmiş ifade elde etmek için, bir Karnaugh diyagramı kullanılabilir. Bkz: Şekil 2.3 (b). R ve S için Karnaugh diyagramları Şekil 2.31 (c), (d)'de verilmiştir.

| HC | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| WD |    |    |    |    |
| 00 |    | 1  | X  |    |
| 01 | 1  | 1  | X  |    |
| 11 | X  | X  | X  | X  |
| 10 |    | 1  | X  |    |

$P = C + HD$

Tablo 2.31 (a) P için karnaugh diyagramı

| HC | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| WD |    |    |    |    |
| 00 |    |    | X  |    |
| 01 |    | 1  | X  |    |
| 11 | X  | X  | X  | X  |
| 10 |    |    | X  |    |

$R = CD$

Tablo 2.31 (c) R için karnaugh diyagramı

| HC | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| WD |    |    |    |    |
| 00 |    |    | X  | 1  |
| 01 |    |    | X  | 1  |
| 11 | X  | X  | X  | X  |
| 10 | 1  |    | X  | 1  |

$Q = H + CW$

Tablo 2.31 (b) Q için karnaugh diyagramı

| HC | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| WD |    |    |    |    |
| 00 |    |    | X  |    |
| 01 |    |    | X  |    |
| 11 | X  | X  | X  | X  |
| 10 |    |    | X  | 1  |

$S = HW$

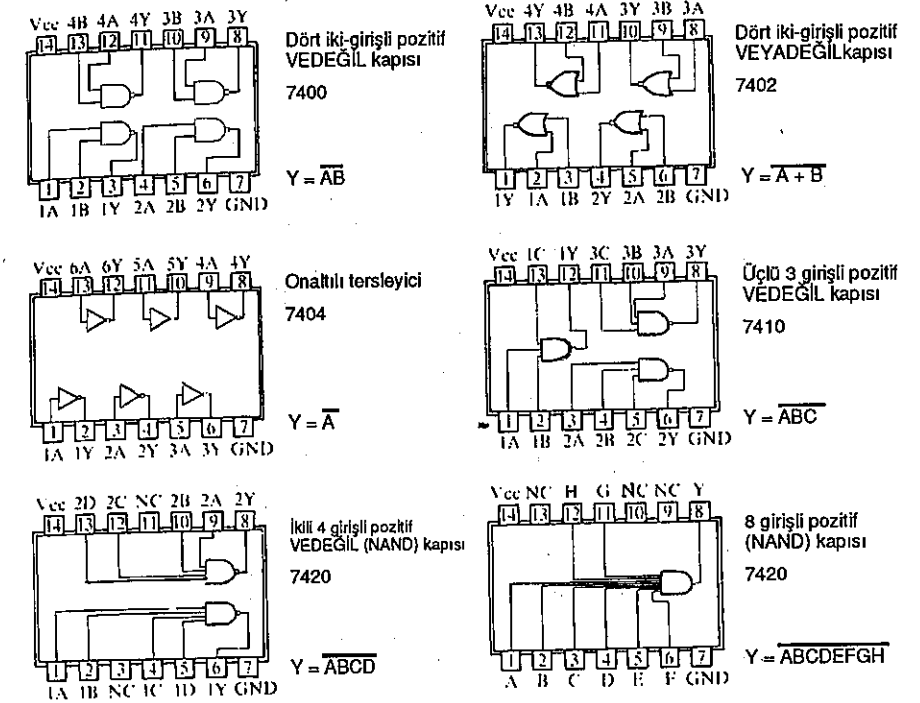
Tablo 2.31 (d) S için Karnaugh diyagramı

### 2.3.2 Özel-amaçlı birleşik mantık elemanları

Şimdiye kadar, tüm sayısal sistemlerin oluşturulabileceği temel mantık elemanlarından söz ettik. Teknoloji ilerledikçe, daha çok sayıda bileşenin tek bir yonga içine sığdırılabilmesi ile daha karmaşık devreler üretilmiştir. Günümüzde, belirli bir mantık işlevin gerçekleştirilebilecek, birkaç kapı devresi içeren yongalar piyasadan sağlanabilmektedir. Böyle devreler küçük ölçekli entegrasyon (SSI) sınıfına giren kapı devrelerinden farklı olarak, orta-ölçekli entegrasyon (MSI) devreler adını alırlar. 1970'lerde tüm sistemin tek bir yonga üzerinde olabildiği devreler piyasaya çıkmaya başladı. Mikroişlemci bunun en göze alıcı örneği olup, işlemciyi üretmede kullanılan teknoloji büyük ölçekli entegrasyon (LSI) olarak adlandırılmaktadır.

1980'ler ise, çok geniş ölçekli entegrasyon (VLSI) çağı olarak adlandırılmaktadır. Oldukça ilginçtir ki, çok geniş ölçekli tümleştirme başta, tek bir yonga üzerinde çok güçlü bilgisayarlar oluşturabilmek için değil de; 256 Kbit (yani  $256 \times 210 = 218 = 262144$  bit) veriyi tutabilen bellek dizilerinin üretimine uygulanmıştır.

Şekil 2.32'de, 14-bacaklı DIL paketleri halinde sağlanabilen bazı popüler SSI (küçük ölçekli entegrasyon) kapıları gösterilmektedir. DIL (Dual in line = Çift sıralı), iki paralel bacak sırasının varlığını belirtir. Sıralar arasında 0,3 inçlik mesafe

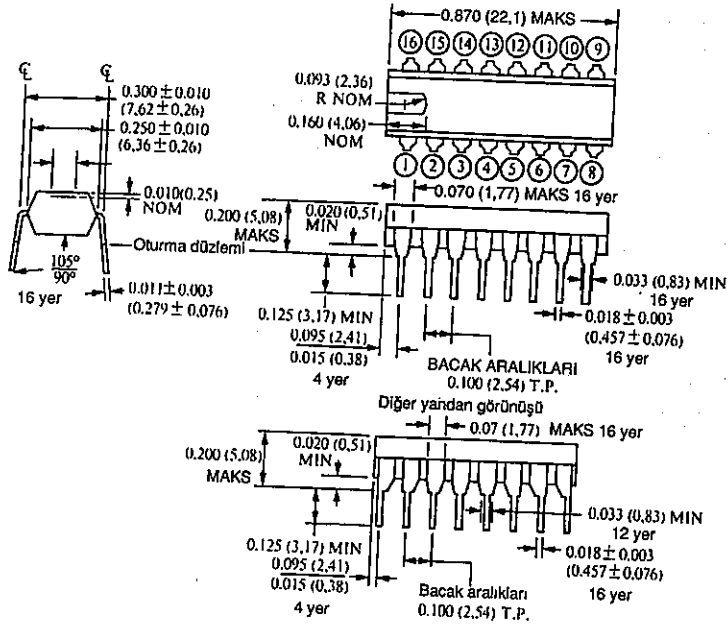


Şekil 2.32 14-bacaklı DIL paketleri halinde bazı SSI (küçük ölçekli entegrasyon) devreleri

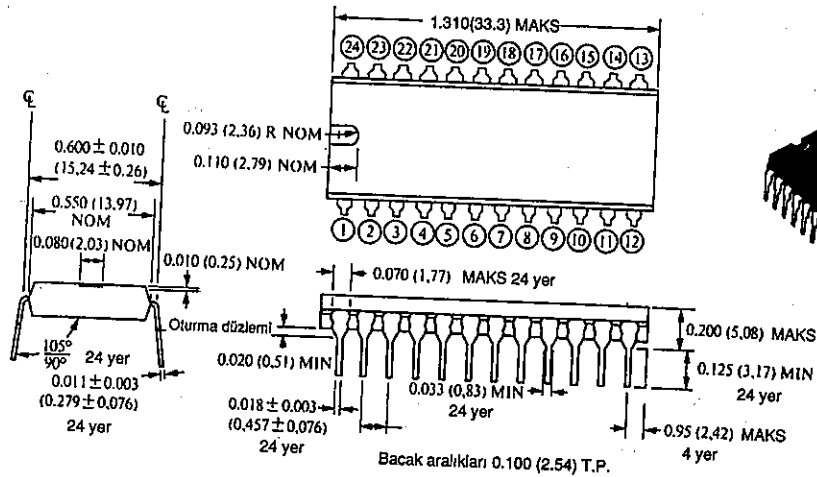
olup, bacaklar birbirinden 0,1 inç uzaklıktadır. Güç kaynağı için iki bacağa ( $V_{cc} = +5.0$  Volt ve toprak=0 Volt) gereksinim vardır. Paket içine, verilen bir tipte kapılardan mümkün olduğu sayıda konulur. Kapıların sayısı yalnızca, bacak sayısı tarafından sınırlanır. Şekil 2.33'de, bu tipte entegre devrelerle birlikte daha geniş entegre devre paketleri gösterilmiştir. Paketin fiziksel boyu ile içerdiği devrelerin karmaşıklığı arasında bir ilişki olmayıp, boyuttan çok, dış bağlantı elemanlarının sayısı (bacaklar) ile ilişkilidir.

Şekil 2.34'de, bazı temel, özel-amaçlı VE-VEYA-DEĞİL adı verilen mantık devrelerini göstermektedir. Bunlar SSI devreleri olup, sıkça karşımıza çıkan mantık işlevlerini gerçekleştirirler. Örneğin 7454, çarpımların toplamının tersini  $F = \overline{AB+CD+EF+GH}$  üretir. Eğer bir eşitlik yukarıdaki şekilde ifade edilebilirse, temel kapıların oluşturduğu iki veya üç yonga yerine, tek bir yonga ile gerçekleştirilebilir.

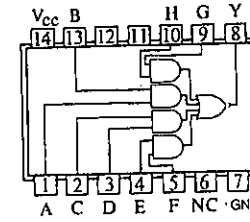
16 - bacaklı N paketi



24 - bacaklı N paketi

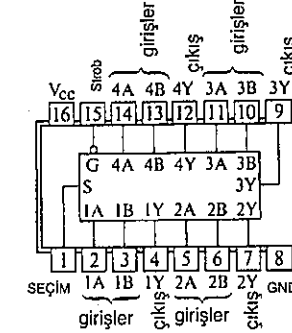


Şekil 2.33 DIL paketi



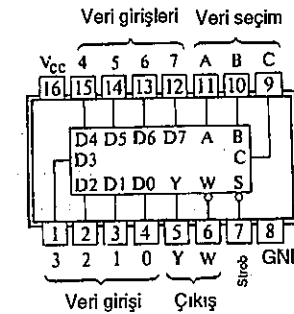
$$Y = \overline{AB + CD + EF + GH}$$

Şekil 2.34 7454 VE-VEYA-DEĞİL kapısı



Şekil 2.35 74157 Dörtlü iki-girişli çoğullayıcı

Sayısal tasarımda, düzenli olarak karşımıza çıkan, özellikle yaygın bir işlevsel birim de çoğullayıcılardır. Sayfa 26'da örnek 2'de basit iki-girişli bir çoğullayıcı verilmişti. Şekil 2.35'de, dört adet iki girişli çoğullayıcı 74157 gösterilmiştir. Bu bir 16-bacaklı MSI (Orta ölçekli entegrasyon) devresidir. Dört Y çıkışından herbiri, SEÇİM(SELECT) = 0 olduğu zaman, ilişkin A girişine ve SEÇİM=1 olduğu zaman da B girişine bağlanır. STROB girişi, STROB=1 olduğu zaman tüm Y çıkışlarını mantık 0 durumuna getirmeye zorlar. Çoğullayıcının iki olası veri kaynağı arasında geçiş yapması ile ilgili kullanım, daha önce koşullu dallanmanın gerçekleştirilmesi örneğinde görülmüştü.



Şekil 2.36 Sekizden-biri seçen çoğullayıcı 74151

Şekil 2.36'da, 74151, sekizden-biri seçen veri çoğullayıcısı gösterilmektedir. Burada 8 adet giriş;  $D_0, D_1, \dots, D_7$  ve bir çıkış  $Y$  (artı tümleyeni  $W = \bar{Y}$ ), üç veri-seçim girişi;  $A, B, C$  bulunmaktadır.  $A, B, C = 0, 0, 0$  olduğunda çıkış,  $Y = D_0$  ve  $A, B, C = 1, 1, 1$  olduğunda da  $Y = D_7$  vb. gerçekleşir. Yani, veri-seçim girişinin ikili değeri  $i$  ise çıkış  $Y = D_i$  ile verilir. Bu devrenin tipik bir uygulaması, bir sayısal sistemde bulunan mantıksal koşullardan birinin seçilmesidir. Sekiz giriş, sistemdeki sekiz noktaya bağlanır.  $A, B, C$ 'ye uygun bir kod uygulamak ve  $Y$ 'deki çıkışı gözlemleme suretiyle 8 noktadan biri test edilebilir.

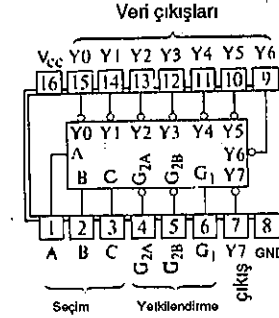
Çoğullayıcının tersi yönünde işlev gören devreler tekilleyici olarak adlandırılırlar. Şekil 2.37(a)'da, 74138 devresi yani, 3hat giriş/8-hat çıkış tekilleyici gösterilmektedir. Bu elemanın doğruluk tablosu, Tablo 2.16'da verilmiştir.

Tablo 2.16 74138 tekilleyici için doğruluk tablosu

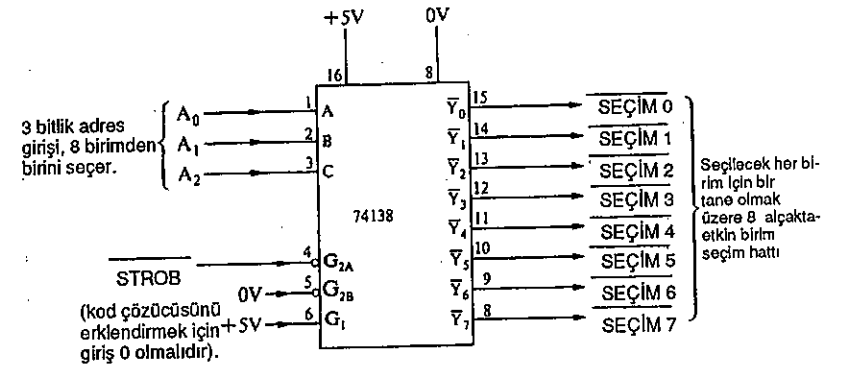
| Girişler      |          |          |     |     |     |             |             |             |             |             |             |             |             |
|---------------|----------|----------|-----|-----|-----|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Yetkilendirme |          | Seçim    |     |     |     |             |             |             |             |             |             |             |             |
| $G_1$         | $G_{2A}$ | $G_{2B}$ | $C$ | $B$ | $A$ | $\bar{Y}_0$ | $\bar{Y}_1$ | $\bar{Y}_2$ | $\bar{Y}_3$ | $\bar{Y}_4$ | $\bar{Y}_5$ | $\bar{Y}_6$ | $\bar{Y}_7$ |
| X             | 1        | 1        | X   | X   | X   | 1           | 1           | 1           | 1           | 1           | 1           | 1           | 1           |
| 0             | X        | X        | X   | X   | X   | 1           | 1           | 1           | 1           | 1           | 1           | 1           | 1           |
| 1             | 0        | 0        | 0   | 0   | 0   | 0           | 1           | 1           | 1           | 1           | 1           | 1           | 1           |
| 1             | 0        | 0        | 0   | 0   | 1   | 1           | 0           | 1           | 1           | 1           | 1           | 1           | 1           |
| 1             | 0        | 0        | 0   | 1   | 0   | 1           | 1           | 0           | 1           | 1           | 1           | 1           | 1           |
| 1             | 0        | 0        | 0   | 1   | 1   | 1           | 1           | 1           | 0           | 1           | 1           | 1           | 1           |
| 1             | 0        | 0        | 1   | 0   | 0   | 1           | 1           | 1           | 1           | 0           | 1           | 1           | 1           |
| 1             | 0        | 0        | 1   | 0   | 1   | 1           | 1           | 1           | 1           | 1           | 0           | 1           | 1           |
| 1             | 0        | 0        | 1   | 1   | 0   | 1           | 1           | 1           | 1           | 1           | 1           | 0           | 1           |
| 1             | 0        | 0        | 1   | 1   | 1   | 1           | 1           | 1           | 1           | 1           | 1           | 1           | 0           |

$Y_0$ 'dan  $Y_7$ 'ye kadar olan çıkışlar alçakta-etkin olup, uygun giriş seçilmediği sürece mantıksal bir durumda kalırlar. Elemanın üç yetkilendirme girişi bulunmaktadır,  $G_1, G_{1A}, G_{2B}$ , yonganın seçilebilmesi için bu girişler sırayla 1,0,0 değerini almalıdırlar.

Yonga seçildiğinde, sekiz kapıdan biri (ve yalnızca biri) mantıksal sıfır durumundadır. Seçilen çıkış, seçme girişleri  $A, B, C$ 'deki üç-bitlik koda bağlıdır. Bu devrenin bir uygulaması da 'birim seçici'dir. Varsayalım ki elimizde sekiz birim bulunsun, ancak verilen zamanda bunlardan yalnızca bir tanesi etkin (kullanımda) olabilsin. Eğer her bir birim, yetkilendirme girişlerine mantıksal sıfır uygulanarak yetkilendirilirse  $C, B, A$  'ya uygulanan ikili kod, hangi birimin seçileceğini belirleyecektir. Şekil 2.37 (b)'de, 74138'in, bu türde bir uygulaması gösterilmiştir.



Şekil 2.37 (a) 74138 3-hat giriş/8-hat çıkış tekilleyici

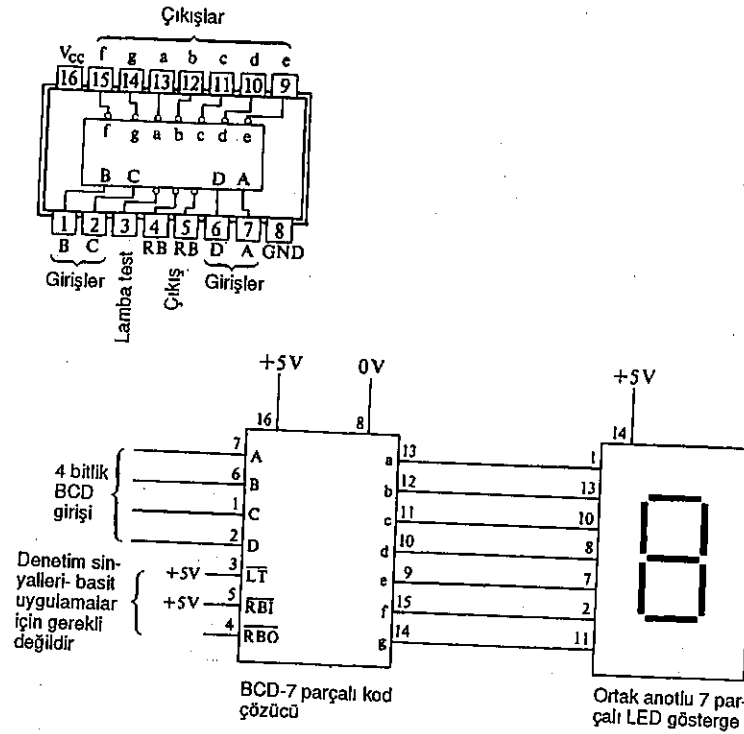


Not: Belli bir anda, SEÇİM 0 - SEÇİM 7 çıkışlarıyla yetkilendirilen 8 birimden yalnızca biri aktif yapılabilir.

Şekil 2.37 (b) 74138'in birim seçici olarak kullanılması

Birçok diğer özel-amaçlı mantık elemanları arasında, kod-dönüştürücüler ve aritmetik elemanlar bulunmaktadır. Kod-dönüştürücüler bir ikili kodu diğerine dönüştürürler. Örneğin Şekil 2.38'de, 7447'nin, BCD yedi-parçalı kod çözücü olarak nasıl kullanıldığı gösterilmektedir.

0'dan 9 kadar olan on sayı, yedi adet çizgi şeklindeki parçalarla oluşturulabilir. Bunun örneğini herhangi bir sayısal saatte görebilirsiniz. 7447 entegre devresi, 4-bitlik BCD kodunu (Bkz:Kısım 3.3.9), ilgili ondalık sayıyı oluşturabilmek için gereken parça birleşimine dönüştürür.



Şekil 2.38 7447 BCD-yedi-parçalı kod çözümü

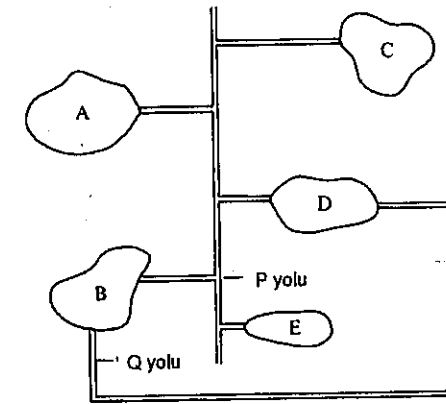
Ayrıca üç bitin toplamını sağlayan tam toplayıcıdan, komple 8-bitlik aritmetik birimine kadar bir dizi aritmetik elemanlar da vardır. Tam toplayıcı, Kısım 3.4.2'de incelenmiştir.

Yukarıda sözü edilen birleşik mantık devrelerine ek olarak, tek bir yonga üzerinde gerçekleştirilmiş birçok ardışıl devre mevcuttur. Kısım 2.6'da, ardışıl devreler ele alınmıştır.

## 2.4 Üç-durumlu mantık

Bu bölümün başında verilen mantıksal-eleman türleri, bir veya daha fazla mantıksal çıkışın belli sayıda girişlerden üretildiği, işlevsel birimleri oluşturmakta kullanılmaktadır. Bilgisayarlar, Kısım 2.5'de tanımlanacak olan saklama elemanlarıyla bu türde işlevsel birimlerin birbirine bağlanmasıyla oluşturulmuşlardır. Biz burada,

bilgisayarların çeşitli işlevsel birimlerinin birbiriyle bağlantısına olanak tanıyan özel bir tip kapıyı (üç-durumlu çıkışı olan kapıyı) inceleyeceğiz.

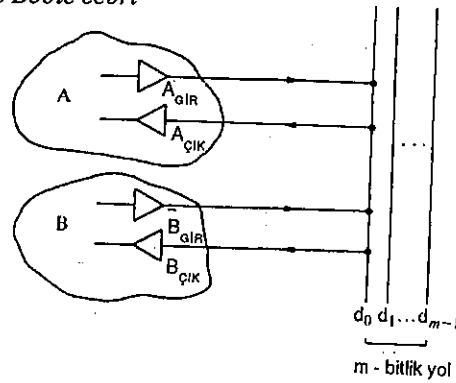


Şekil 2.39 İşlevsel birimler ve yollar

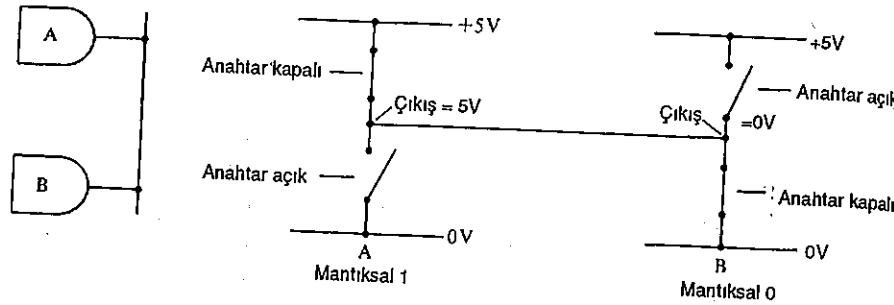
Şekil 2.39'da, beş işlevsel birimden, A,B,C,D ve E, oluşmuş bir sistem gösterilmektedir. Bu birimler birbirlerine P ve Q gibi iki veri anayoluyla (yoluyla) bağlanmış olup, verinin bir birimden diğerine taşınabilmesine izin verirler. Arada yolların bulunması tümüyle gerekli değildir. Bilgi değiş-tokuşunda bulunan sayı sistem birimleri arasında doğrudan bağlantılar sağlamak da mümkündür. Yine, günlük yaşamımızda da anayolların bulunması da zorunlu değildir. Her ev, gerek duyduğu yerlere yani diğer evlere fabrikalara, dükkanlara ve servislere arayollarla bağlı olabilir. Her iki durumda da, birkaç işlevsel birim (ev) dışında bağlantıların tümüne yakın kısmı ekonomik olmayacaktır.

Yollar, diyagramlarda, Şekil 2.39'da olduğu gibi tek (veya çift) çizgi ile gösterilirler. Pratikte yollar, birçok hatlardan yapılmış olup, Şekil 2.40'da nasıl düzenlendikleri gösterilmiştir. Buradaki A ve B devreleri, verileri yol aracılığıyla gönderip alabilirler. Biz burada A ve B devrelerinin nasıl çalıştıklarıyla ilgilenmeyip, sadece basit olarak birbirleriyle nasıl iletişim kurduklarını göstermeyi amaçlamaktayız. Karmaşıklığa yol açmamak için, yalnızca yolun bir hattına olan bağlantılar gösterilmiştir. Benzer düzenlemeler  $d_1 - d_{m-1}$  için de mevcuttur.

A'nın B'ye veri göndermek istediğini varsayalım. A devreleri veriyi yola,  $A_{çık}$  kapısıyla çıkarır, B ise veriyi  $B_{gır}$  kapısıyla alır. Bu düzenleme, gerçekte, Şekil 2.41'e bakıldığında görüldüğü gibi uygulanabilir değildir. Yolun her bir hattı, kendisine bağlı birtakım çıkışlara sahiptir (A,B...den). Şekil 2.41'de, farklı mantıksal durumlardaki iki çıkışın birbirine bağlanmasının sonucunu göstermektedir.



Şekil 2.40 Devrelerin yollara bağlanmaları

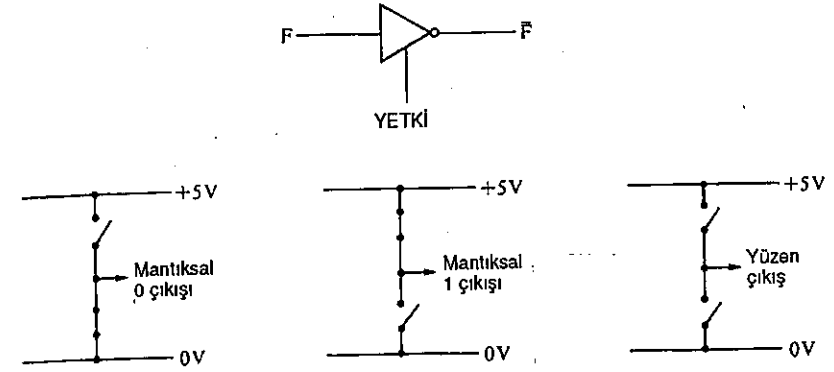


Şekil 2.41 Çıkışları birbirine bağlı iki kapının çıkışları

A çıkışı mantıksal bir durumundadır ve kapının içindeki bir anahtar aracılığıyla +5 Volta çekilmiştir. Benzer biçimde B çıkışı da mantıksal sıfır durumundadır ve diğer bir anahtar tarafından 0 Volta çekilmiştir. Bu anahtarlar, ya iletim ya da kesim durumunda çalışan transistörlerdir. İki çıkış farklı mantıksal durumda olup, çıkışlarının birbirine bağlı olması nedeniyle iki sorun bulunmaktadır: Birincisi işin felsefesiyle ilgilidir. Gerilim, iletken boyunca her noktada sabit olduğundan, iletken boyunca mantıksal düzey de sabit değerdedir. Bununla birlikte uçları farklı gerilimlere bağlı olduğundan, iletken üzerindeki gerilim düzeyi belirsizdir ve Boole cebri kurallarından birinin dışına çıkmaktadır. Yani mantıksal sıfırla mantıksal bir arasında yer alan geçerli bir aradeğer bulunmamaktadır. İkinci olarak ve daha da pratik olanı, +5V güç kaynağı ile toprak (0V) arasında doğrudan fiziksel bir yolun bulunmasıdır. Bu bir kısa devredir ve iki çıkış devresinden akan akım bu devreleri tahrip edecektir.

Çıkışların birbirine bağlanmasındaki güçlük üç durumlu kapı yardımıyla giderilir. Üç durumlu mantık, isminin çağrıştırdığı gibi Boole cebrinin üçlü ya da üç değerli mantıkla ilgili bölümü olmayıp; veri iletişimine etkin olarak katılmayan kapıların

yoldan izole edilmek suretiyle, Şekil 2.41'deki çelişkinin giderilmesini sağlayan bir yöntemdir.



Şekil 2.42 üç durumlu çıkış

Şekil 2.42'de, üç durumlu çıkışı olan bir kapı gösterilmektedir. İş basitleştirmek için burada gösterilen kapı, bir tersleyicidir (tümleyici). Gerçekte, herhangi bir tip kapı üç durumlu çıkışa sahip olabilir. Bu kapının, YETKİ (ENABLE) adlı özel bir girişi bulunmaktadır. YETKİ=1 olduğunda, kapı normal biçimde davranır ve çıkışı, girişine bağlı olarak mantıksal 0 veya mantıksal 1'dir. YETKİ=0 olduğunda da, kapının çıkış devresindeki her iki anahtar da açık olup çıkış, fiziksel olarak kapının iç devrelerinden izole olmuş durumdadır. YETKİ=0 sıfır olduğunda çıkışın hangi durumda olduğu sorulursa buna, bu türde bir sorunun anlamsız olduğu cevabı verilmelidir. Gerçekte, yetkilendirilmemiş üç durumlu kapının çıkışı normalde yola bağlı olduğundan, çıkış ucundaki mantıksal düzey, bağlı bulunduğu yolun mantıksal düzeyiyle aynı olacaktır. Bu nedenle üç durumlu kapının, üçüncü durumu "yüzer" çıkış olarak adlandırılır. Yolun trafiğine göre yukarı veya aşağı doğru yüzmektedir. Tablo 2.17, üç durum çıkışlı bir tersleyicinin doğruluk tablosudur.

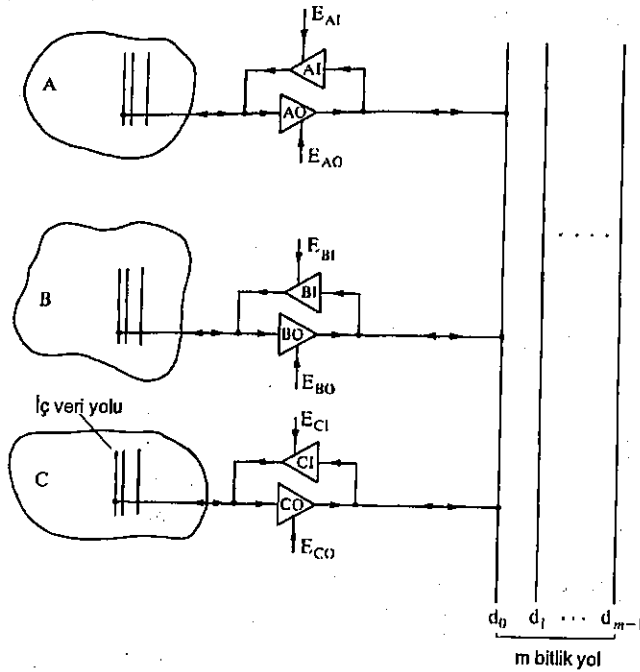
Tablo 2.17 Üç durumlu tersleyici için doğruluk tablosu

| YETKİ | A | $\bar{A}$ çıkışı |
|-------|---|------------------|
| 0     | 0 | X                |
| 0     | 1 | X                |
| 1     | 0 | 1                |
| 1     | 1 | 0                |

Burada X'ler, 'yüzer' değerleri gösterir.

Şekil 2.43'de, üç durumlu kapıların, yolla bağlantılı bir kapıyı gerçekleştirmek üzere uygulamada nasıl kullanıldıkları gösterilmektedir. Yetkilendirildiklerinde çıkışlarının girişlerine eşit olması nedeniyle bu kapılar tampon olarak adlandırılır. Basit olarak, devreleri yollara bağlamak veya ayırmak için kullanılırlar.

Biyelge



Şekil 2.43 Üç-durumlu ve yol

A,B,C devrelerinin çıkışları yola, sırayla  $E_{AO}$ ,  $E_{BO}$  ve  $E_{CO}$  sinyalleri tarafından yetkilendirilen AO, BO, ve CO üç durumlu tamponları tarafından yerleştirilir. Eğer bir devre, yola veri çıkarmak isterse, yetki sinyalini (örneğin  $E_{BO}$ 'yi) mantıksal 1'e çekmelidir. Burada,  $E_{AO}$ ,  $E_{BO}$  ve  $E_{CO}$  sinyallerinden birden fazlasının aynı anda 1 olmaması hayati önem taşımaktadır.

Devrelerin yoldan veri alması, kendi giriş tamponları (AI, BI ve CI) aracılığıyla gerçekleşir. Eğer devrelerden biri veri almak isterse, giriş tamponunu  $E_{AI}$ ,  $E_{BI}$  veya  $E_{CI}$  aracılığıyla yetkilendirir. Örneğin C devresi, A devresine veri iletmek istesin, gerekli olan tek şey  $E_{CO}$  ve  $E_{AI}$ 'nin aynı anda mantıksal 1'e çekilmeleridir. Diğer tüm yetki sinyalleri, bilgi aktarımı süresince mantıksal sıfır düzeyinde kalırlar. Dikkat ediniz ki giriş tamponları (AI, BI, CI) her zaman gerekli değildir. Eğer, yoldan devre içine alınan veri, bir veya daha fazla kapının girişine gidiyorsa, bir tampona gerek olmayacaktır. Bununla birlikte, eğer giriş verisi, diğer kapıların çıkışlarını yerleştirebilecekleri bir iç yola bağlı bulunuyorsa tampon gerekli olacaktır.

Yukarıdaki tanımda, kapıların adları ve ilişkin denetim sinyalleri dikkatli bir şekilde seçilmiştir. AO,  $A_{\text{çık}}$  yerini ve AI da  $A_{\text{gir}}$  yerini tutmaktadır. Bu, kullanımda

olan devreye aktarılabilecek veri ve ilişkin olarak kapıyı ve yönü belirleyecektir. Benzer biçimde  $E_{AO}$ , ' $A_{\text{çık}}$  kapısının yetkilendirilmesi'  $E_{AI}$  ise Ağır kapısının yetkilendirilmesi' yerine geçmektedir. Tutarlı ve anlamlı kelimeler seçilirse devre diyagramlarının ve ilişkin metinlerin okunması kolaylaşmış olacaktır.

Yolla bağlantılı bir sisteme ilişkin ayrıntılar Kısım 2.6.1'de incelenecektir. MİB yapısına ilişkin olan Bölüm 4'de, MİB'nin gerçekte temel bilgisayar işlemlerini nasıl başardığını anlatırken; yolların, MİB tarafından yoğun bir şekilde kullanımı ele alınacaktır.

## 2.5 İki-kararlılar ve kaydediciler

Bu noktaya kadar karşılaştığımız mantık devreleri, çıkışın yalnızca girişin işlevi olduğu birleşik devrelerdir. Çıkışları, sadece o andaki girişlerine değil de, geçmişte ki girişlerine de bağlı olan devreler ardışıl devreler adını alır. Ardışıl devrelerin temel yapı taşları flip-flop'lar ve ikili kararlılardır.

Burada amacım; ardışıl devreleri, tümüyle incelenmeleri sayısal mühendisliğin başlı başına bir dalını oluşturacak hacimde olmaları nedeniyle, yalnızca giriş düzeyinde incelemektir. Ardışıl devreler, tümü merkezi işlem biriminin çalışmasında temel birimler olan kaydedicilerin, sayıcıların ve kaydırıcıların gerçekleştirilmelerinde gerekli olduklarından, bilgisayar donanımı üzerine yazılı metinlerin kapsamı dışında tutulamazlar.

İki-kararlılar denmesinin nedeni, verilen bir giriş için çıkışın, iki kararlı durumdan birinde sonsuz olarak kalabilmesidir. Yani, belirli bir giriş grubu için çıkış değeri, önceki girişlere bağlı olmak üzere, mantıksal sıfır veya mantıksal bir değerini alabilir. Böyle bir devre geçmişini hatırlama yeteneğindedir. Bellek elemanları ile ilgili daha ayrıntılı inceleme Bölüm 6'da verilmiştir. Bir iki kararlı devre, mevcut en küçük bellek hücresi olup, yalnızca 1-bitlik bilgi depolayabilir. İki kararlı ile eşanlamlı olan flip-flop bir ses türetmeli sözcük olup, devrenin bir duruma geçtiği, daha sonra, tümleyenine döndüğü esasına göre çalışmaktadır.

Flip-flopların tanımına iki farklı yaklaşım vardır: Bunlardan biri tanesi, birimi bir soyut model olarak ele alıp karakteristiklerini tanımlamak, ne yaptığını göstermek ve ardından da nasıl tasarlanabileceklerini belirtmek şeklindedir. Diğer ise, bir flip-flop'un yalnızca iki kapı kullanılarak nasıl gerçekleştirildiğini göstermek ve ardından özelliklerinin nasıl kullanılabileceğini açıklamaktır. Ben burada ikinci yolu izleyeceğim.

## 2.5.1 RS tipi flip-flop'lar

Şekil 2.44'de, bu kitaptaki en karmaşık devrelerden biri gösterilmektedir. Basit girişli, iki VEYADEĞİL (NOR) kapısında fazla karmaşık olmadığı halde, çalışması ilk bakışta hemen anlaşılamamaktadır. Bu devrenin A ve B olmak üzere iki girişi ve X, ve Y olmak üzere iki çıkışı vardır. VEYADEĞİL (NOR) kapısı için doğruluk tablosu Şekil 2.44 içinde referans amacıyla verilmiştir.

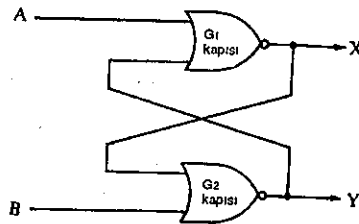
Kapılara ilişkin eşitliklerden, X ve Y için aşağıdaki ifadeleri doğrudan yazabiliriz:

$$(i) \quad X = \overline{A + Y}$$

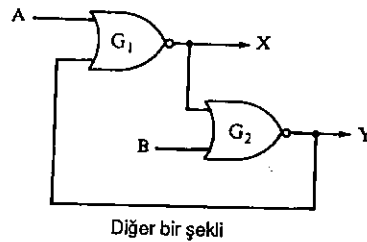
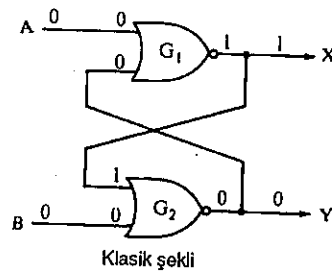
$$(ii) \quad Y = \overline{B + X}$$

OYA kapısı için doğruluk tablosu

| A | B | $\overline{A+B}$ |
|---|---|------------------|
| 0 | 0 | 1                |
| 0 | 1 | 0                |
| 1 | 0 | 0                |
| 1 | 1 | 0                |



Şekil 2.44 İki çarpaz bağlı VEYADEĞİL (NOR) kapısı



Şekil 2.45 Şekil 2.44'ün çalışması

Y'ye ilişkin değeri, (i) eşitliğinde yerine koyarsak

$$X = \overline{A + B + X} = \overline{A} \cdot \overline{B + X}$$

$$= \overline{A} \cdot (\overline{B} + \overline{X})$$

$$= \overline{A} \cdot \overline{B} + \overline{A} \cdot \overline{X}$$

elde edilir.

Boole Cebirinde, bölme ve çarpma işlemleri tanımlanmadığından, çıkışın, yine çıkışın işlevi olduğu, oldukça kötü-görünümlü denklemler elde edilmektedir. Bu devreyi anlamanın daha iyi bir yolu belki, X için bir değer belirlemek sonra, bunun bizi nereye götüreceğini izlemektir.

Şekil 2.45 devre X=1 ve A=B=0 değerleri için gösterilmiştir. G<sub>2</sub> kapısına girişler X=1, B=0 olduğundan, çıkışı yani  $\overline{X+B}$ , 0 olmalıdır. G<sub>1</sub> kapısına girişler Y=0 ve A=0 olup, X çıkışı,  $\overline{Y+A}$ , 1'dir. Bu durumun kendiliğinden oluştuğuna dikkat ediniz. G<sub>1</sub> kapısının çıkışı 1'dir ve X'i mantıksal 1 düzeyinde tutmak üzere G<sub>1</sub> kapısına geri beslenmektedir. Yani çıkış, gerçekte kendisini sürdürmektedir. Böyle bir devreye, çıkış, girişe geri beslendiğinden çarpaz-bağlı adı verilir.

Eğer G<sub>2</sub> kapısının B girişi, A girişi alçak düzeyde kalırken yüksek ise, G<sub>2</sub> kapısının çıkışı (yani Y) etkilenmez, çünkü VEYADEĞİL kapısının çıkışı, girişlerden biri alçak ise, alçak düzeydedir. X zaten yüksek olduğundan, B'nin durumunun, Y'nin durumu üzerinde etkisi yoktur.

Eğer, A girişi, B alçak düzeyde kalırken yüksek olursa, G<sub>1</sub>'in X çıkışı mantıksal 0 durumuna dönmelidir. G<sub>2</sub>'nin girişlerinin her ikisi de şimdi mantıksal 0 durumunda olup, Y çıkışı mantıksal 1 düzeyine yükselir. Bununla birlikte Y, G<sub>1</sub> kapısının girişine geri beslendiğinden, A sıfıra gitse bile X çıkışı, mantıksal 0'da kalır. Böylece A'nın bire kurulması, X çıkışının 1'den 0'a dönmesine neden olur ve A sıfıra dönünceye kadar aynı durumda kalır. Tablo 2.18, Şekil 2.44'e ilişkin doğruluk tablosudur.

Tablo 2.18 (a),(b) Şekil 2.44'deki devreye ilişkin doğruluk tabloları

| Girişler      |   |   |                | Çıkış         |   |   |                |
|---------------|---|---|----------------|---------------|---|---|----------------|
| A             | B | X | X <sup>+</sup> | R             | S | Q | Q <sup>+</sup> |
| 0             | 0 | 0 | 0              | 0             | 0 | 0 | 0              |
| 0             | 0 | 1 | 1              | 0             | 0 | 1 | 1              |
| 0             | 1 | 0 | 1              | 0             | 1 | 0 | 1              |
| 0             | 1 | 1 | 1              | 0             | 1 | 1 | 1              |
| 1             | 0 | 0 | 0              | 1             | 0 | 0 | 0              |
| 1             | 0 | 1 | 0              | 1             | 0 | 1 | 0              |
| 1             | 1 | 0 | 0              | 1             | 1 | 0 | ?              |
| 1             | 1 | 1 | 0              | 1             | 1 | 1 | ?              |
| Eski X Yeni X |   |   |                | Eski Q Yeni Q |   |   |                |

Doğruluk tablosu şöyle yorumlanabilir. Devrenin şu andaki çıkışı, X (veya Q) ve, giriş uçlarına uygulanacak yeni girişler de A, B (veya R,S) olsun. Devreye bu yeni girişler uygulandığında X çıkışı, X<sup>+</sup> (veya Q<sup>+</sup>) olarak verilir. Örneğin, şu andaki

çıkış  $X$ , 1 ve yeni  $A$  ve  $B$  değerleri  $A=1$   $B=0$  ise, yeni çıkış  $X^+$ , 0 olacaktır.  $X^+$  bu değeri, bir sonraki yeni girişler  $A$  ve  $B$  devreye uygulandığında,  $X^+$ 'in bir sonraki değerini oluşturacaktır.

Tablo 2.18 (a), Şekil 2.44'deki VEYADEĞİL kapılarına ve Tablo 2.18 (b) de, R flip-flop olarak adlandırılan bu devrenin 'idealleştirilmiş' biçimine karşılık gelmektedir.



Şekil 2.46'da, RS flip-flop'unun devre sembolü verilmektedir. Bu doğruluk tablolarında iki farklılık vardır. Sağ tarafta verilen biçimde; R ve S girişli çıkışlı bir RS flip-flopun klasik etiketlenmesi kullanılmıştır. Diğer farklılık, doğruluk tablolarının  $A=B=1$  ve  $R=S=1$  durumlarına ilişkin en son giriş değerindedir. Bu farklılıklar daha sonra ele alınacaktır.

Açıkladığımız üzere, Şekil 2.44’de ilişkin devre, kendi (yani  $X = \overline{A}B + \overline{A}X$ ) terimle-  
riyle belirlenen, kendi çıkışına sahiptir. Doğruluk tablosu yaklaşımı, bu problemin  
üstesinden  $X^+$  (veya  $Q^+$ ) değişkenini oluşturmakla gelir. Burada  $X^+$ , eski çıkış  $X$ ’e  
ve şu andaki girişler de, A ve B’ye karşılık gelen yeni çıkıştır. Benzer biçimde,  
eşitlik  $X^+ = \overline{A}B + \overline{A}X$  olarak yeniden yazılabilir. Artık, doğruluk tablosunun giriş ve  
çıkış sütunları, salt konum (yani giriş hatları solda, çıkış hatları da sağda) olarak  
değil zaman olarak da ayrılmıştır. Şu anki  $X$  çıkışı, A ve B girişleriyle birleştirilerek  
yeni çıkış  $X^{++}$ ’ı verir,  $X^{++}$ ’ı üreten  $X$ , artık mevcut olmayıp, geçmişte kalmıştır.

Sağ taraftaki doğruluk tablosundaki R ve S etiketleri, sırasıyla Reset ve Set sözcüklerinin baş harflerini göstermektedir. Reset, 0'a kurmak demektir. (Silme de aynı anlama gelmektedir). 'Set' ise, 1'e kurmak demektir. Tüm flip-flopların çıkış tarihi bir gelenek olarak Q ile gösterilir. Doğruluk tablosunun incelenmesiyle görülecektir ki, her  $R=1$  olduğunda, Q çıkışı da 0'a çekilmektedir. Benzer biçimde,  $S=1$  iken, çıkış 1'e kurulur. R ve S girişlerinin her ikisi birden 0 iken, çıkış değişmez, yani  $Q^+=Q$  olarak kalır.

Eğer R ve S ikisi birden aynı anda yüksekse, çıkış, aynı zamanda hem 0 ve hem de 1 olamayacağından tanımlanmamıştır. (Doğruluk tablosundaki X işaretlerinin nedeni de budur). RS flip-flop'unun iki VEYADEĞİL kapısıyla gerçekleştirilmesi

durumunda,  $A=B=1$  olduğu zaman,  $X$  çıkışı alçak düzeye gider. Uygulamada, RS flip-flop'unun kullanıcısı  $R=S=1$  durumunun ortaya çıkmamasını sağlamalıdır.

İki VEYADEĞİL-kapı devresinin X ve Y olmak üzere iki çıkışı vardır. Devrenin  $A=B=1$  dışındaki tüm girişleri incelenirse, X ve Y'nin, birbirinin tümleyeni olduğu görülecektir. Benzer biçimde, tüm flip-flopların Q ve tümleyeni  $\bar{Q}$  olmak üzere iki çıkışı vardır.

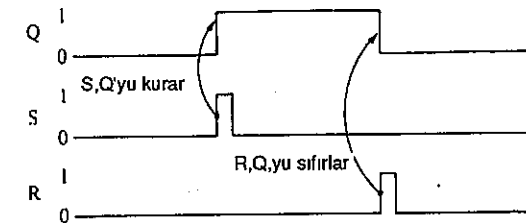
RS tipi veya diğer flip-flopların doğruluk tablolarının gösterimi için iki genel yol vardır. Buraya kadar verilen doğruluk tabloları, her olası giriş grubu için (biri  $Q=0$  için, diğeri  $Q=1$  için) iki hat gerektiğinden kapsamlıdır. Alternatif bir yaklaşım,  $Q$  değerini tablo 2.19 da olduğu gibi kullanmaktadır.

**Tablo 2.19 RS flip-flop için doğruluk tablosu (tablo 2.18 (b) nin alternatif biçimi)**

| Girişler |   | Çıkış          | Tanımı          |
|----------|---|----------------|-----------------|
| R        | S | Q <sup>+</sup> |                 |
| 0        | 0 | Q              | Değişmez        |
| 0        | 1 | 1              | Çıkış 1 (Set)   |
| 1        | 0 | 0              | Çıkış 0 (Reset) |
| 1        | 1 | X              | Yasaklı         |

$R=S=0$  olduğu zaman yeni çıkış  $Q^+$ , basitçe eski çıkış  $Q$  olacaktır. Diğer bir deyişle, çıkışın durumu değişmez.  $R=S=1$  girişleri,  $Q^+=X$  sonucunu verir. 'X' özel sembolü, doğruluk tablolarında, belirsizlik durumunu veya daha önce gördüğümüz gibi 'dikkate alınmaz' durumunu gösterir. Bir 'dikkate alınmaz' durumu, tasarımcıya, çıkışın bir anlam ifade etmediğini söylemekte olup, 'belirlenemez' durumu ise, çıkışın hesaplanmadığı durumdur.

RS flip-flopun önemli bir uygulaması kısa-ömürlü olayların kaydedilmesinde ortaya çıkar. Eğer flip-flopun Q çıkışı sıfır durumunda ise, S girişindeki (R girişi sıfır varsayılmaktadır) mantıksal bir darbe Q'yu bire çeker ve R girişi Q'yu sıfırlayana kadar birde kalır.

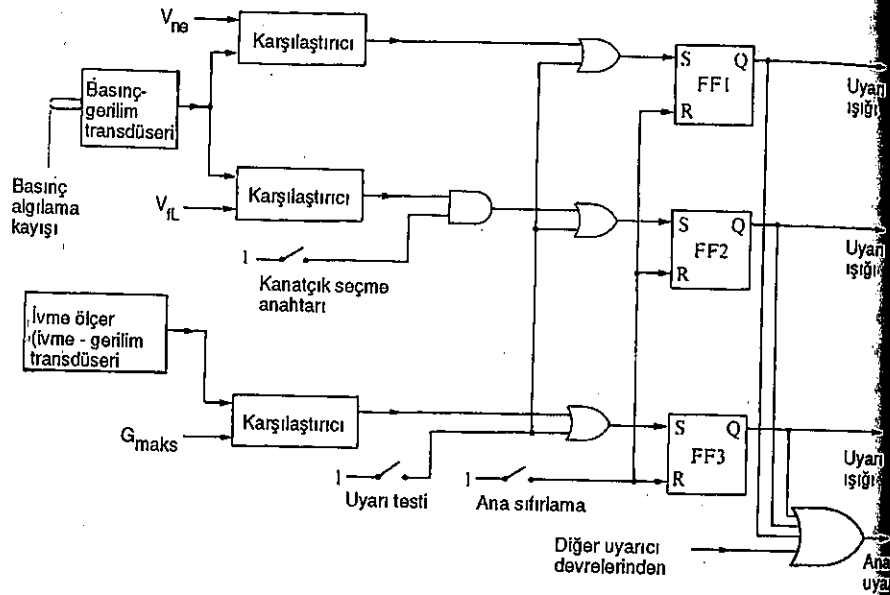


Şekil 2.47 RS flip flopunda, bir darbenin etkisi

Bir darbe ve RS flip flop'u üzerindeki etkisi Şekil 2.47'de gösterilmiştir.

Aşağıda, RS flip-floplarının bir gösterge devresindeki uygulaması ele alınmıştır. Eğer bir uçak, normal 'performans alanı' dışında uçmaya çalışırsa, ani bir hasar söz konusu olmamakla birlikte yapısı sürekli bir şekilde zayıflayabilir. İşin basitliğini sağlamak için, uçağı tehlikeye düşürebileceği ve hasara neden olabileceği düşünülebilen üç olayı ele alalım.

1. İzin verilen maksimum hız  $V_{ne}$ 'nin aşılması.
2. Kanatçıkların (flapların),  $V_{fl}$  flap-sınırlayıcı hızın  $V_{fl}$ 'nin üzerinde artırılması, yani eğer uçak  $V_{fl}$ 'den hızlı gidiyorsa, kanatçıklar indirilmelidir.
3. Maksimum ivme (g-kuvveti)  $G_{maks}$ 'in aşılması.



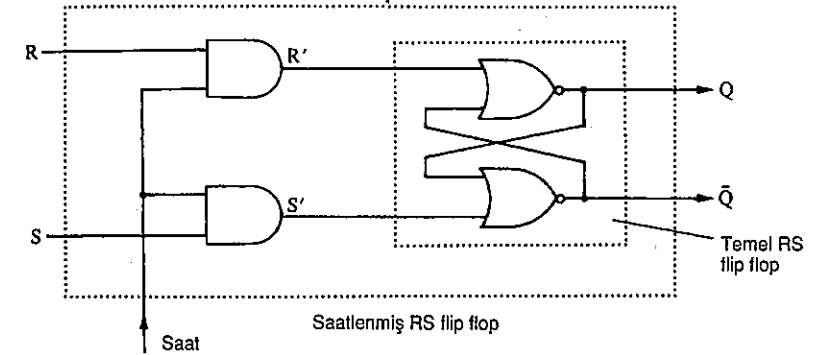
Şekil 2.48 RS flip-flopların bir uyarı sistemine uygulanması

Transdüserler, fiziksel büyüklükleri gerilime dönüştüren elemanlardır. Transdüserden gelen gerilimler, karşılaştırıcılar içindeki üç eşik düzeyi ( $V_{ne}$ ,  $V_{fl}$ ,  $G_{maks}$ ) ile karşılaştırılır. Eğer eşik düzey aşılmışsa karşılaştırıcıların çıkışı bir, aşılmamışsa sıfırdır. Kanatçıkların, kanatçık-sınırlama hızının üzerine olan açılmalarını algılayabilmek için, karşılaştırıcının çıkışı, kanatçıklar aşağıda olduğu zaman doğru (1) çıkışı veren, kanatçık sürme devresinden gelen sinyalle VE'lenir.

Karşılaştırıcılardan gelen sinyaller, VEYA kapıları aracılığı ile, üç RS flip-flopunun S girişine uygulanır. Sistemin açılmasıyla birlikte, tüm R girişlerine aynı anda mantıksal bir darbesi gönderilerek, flip-floplar otomatik olarak sıfırlanırlar. Eğer herhangi bir anda, S girişlerinden biri 1 olursa, bu flip-flopun çıkışı mantıksal bir kurularak alarm devreye sokulur. Tüm çıkışlar beraberce VEYA'lanarak, ana uyarı lambasını yakarlar. Bu, pilotun, periyodik olarak tüm uyarı lambalarını taraması zorunluluğunu ortadan kaldırır. Devrenin ek bir özelliği de test olanağıdır. Uyarı test düğmesine basıldığı zaman, tüm uyarı lambaları yanmalı ve sıfırlama düğmesine basılıncaya kadar öyle kalmalıdır. Bu olanak sistemin düzgün çalışıp çalışmadığını test eder.

#### Saatlenmiş denetimli RS flip-floplar

Şekil 2.44'deki RS tipi bir flip-flop, doğruluk tablosuna uygun olarak girişlerine derhal cevap verir. Bununla birlikte, RS flip-flopunun girişlerinin, uygun bir süre için dikkate alınmaması istenen durumlar vardır. Şekil 2.49'daki devre bunun nasıl başarıldığını göstermektedir.



Şekil 2.49 Saatlenmiş RS flip-flop

Buradaki iç kutuda, normal, değiştirilmemiş RS flip-flop yer almaktadır. R' ve S' girişleri, R ve S girişlerinin C saat girişiyle VE'lenmesinden elde edilir. C=0 olduğu sürece RS flip-flopunun, R', S' girişleri, R ve S'nin ne değerinde olduğuna bakılmaksızın sıfırda tutulmaya zorlanır. Bu girişler sıfır olduğu sürece, RS flip-flopunun çıkışı sabit kalır. C=1, R'=R ve S'=S durumu ortaya çıktığında, flip-flop uygun biçimde yanıt verir. Saat girişi, flip-flopun olması gereken zamana kadar çalışmaktan alıkoyan bir yasaklayıcı olarak düşünülebilir. Saat denetimli flip-floplar, daha ayrıntılı olarak bu bölümde daha sonra ele alınacaktır.

## 2.5.2 D tipi flip-floplar

RS flip-flopu gibi, D flip-flopu da, biri D diğeri C olarak adlandırılan iki girişi sahiptir. D girişi genelde veri ve C girişi de saat palsleri olarak belirtilir. D flip-flop, doğası gereği, saatlenmiş bir flip-flop olup, C girişinin yüksek düz kurulumunu, D flip-flopunun saatlenmesi olarak adlandıracağız. D flip-flopun her saatlenmesi durumunda, D girişindeki değer Q çıkışına aktarılır. Tekrar saatlenene kadar çıkış sabit kalır. Bazıları, D flip-flopunu, D girişini kaydedip, sakladığından, sabitleyici olarak adlandırırlar. Bazıları da, eğer D girişi T zamanında durum değiştiriyor ve  $t$  saniye sonra saatleniyorsa, Q, girişten  $t$  saniye sonrasına kadar durum değiştirmedikinden gecikme elemanı derler. Ben, onu sayı memurlarının sayım yapmasına benzetirim. Saatlendiği zaman girişin sayımını yapar ve diğer sayıma kadar bu bilgileri hatırlar. Tablo 2.20, D flip-flopu için doğruluk tablosunu göstermektedir.

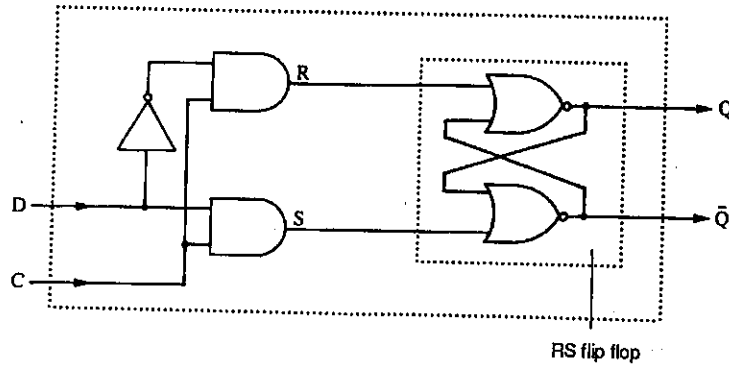
Tablo 2.20 D flip-flopu için doğruluk tablosu

| Girişler |   |   | Çıkış |
|----------|---|---|-------|
| C        | D | Q | Q*    |
| 0        | 0 | 0 | 0     |
| 0        | 0 | 1 | 1     |
| 0        | 1 | 0 | 0     |
| 0        | 1 | 1 | 1     |
| 1        | 0 | 0 | 0     |
| 1        | 0 | 1 | 0     |
| 1        | 1 | 0 | 1     |
| 1        | 1 | 1 | 1     |

$Q^* \leftarrow Q$   
 $Q^* \leftarrow D$

| Girişler |   |    | Çıkış |
|----------|---|----|-------|
| C        | D | Q' |       |
| 0        | 0 | Q  |       |
| 0        | 1 | Q  |       |
| 1        | 0 | 0  |       |
| 1        | 1 | 1  |       |

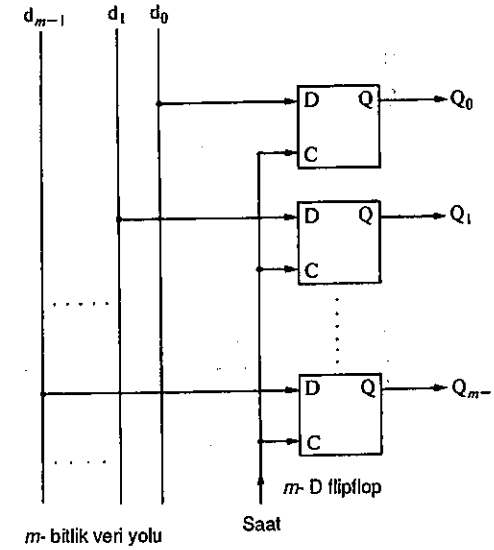
Alternatif biçim



Şekil 2.50 D tipi flip-flop devresi

Şekil 2.50 den görülebildiği üzere, D flip-flopunu, birkaç kapı ekleyerek bir RS flip-flopundan türetmek (elde etmek) mümkündür. İki VE kapısıyla, RS flip-flopu, saatlenmiş türüne çevrilmektedir. VE kapısına gelen C girişi alçak olduğu sürece R ve S girişleri sıfıra kilitlenir ve Q değişmez.

C, yükseğe gittiği zaman, S girişi D'ye, R girişi ise  $\bar{D}$ 'ye bağlanır. Böylece (R, S), eğer D=1 ise (0,1) veya D=0 ise (1,0) olmalıdır. Sonuç olarak D=1 değeri, RS flip-flopunu 1'e kurar, D=0 ise sıfırlar. D flip-flopunun tipik bir uygulaması Şekil 2.51'de verilmiştir.



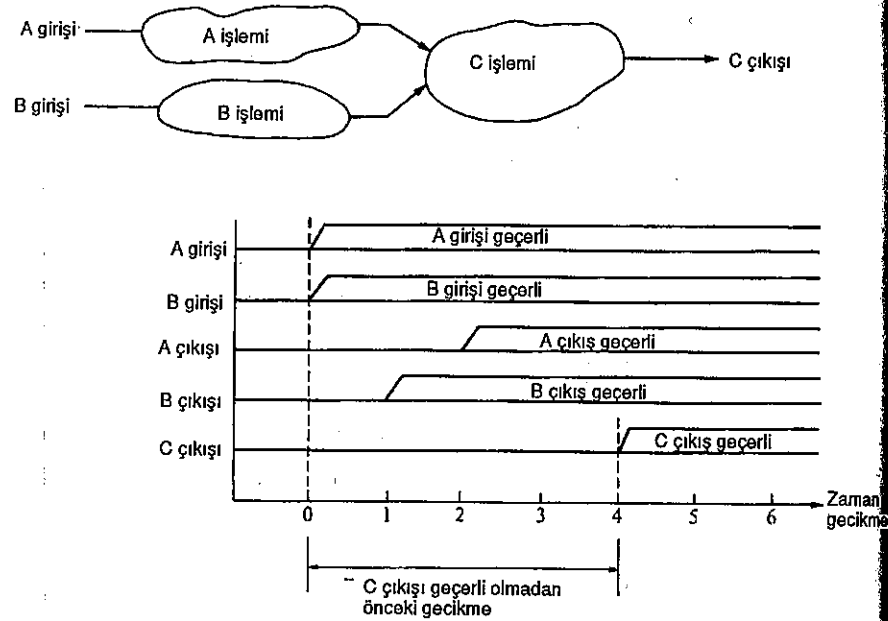
Şekil 2.51 D tipi flip-flopun bir uygulaması

Burada  $m$  bit genişliğindeki veri yolu, veriyi sistemin bir yerinden diğerine taşımaktadır. Yol üzerindeki veri, çeşitli devrelerin kendi verilerini iletmeleri nedeniyle sürekli değişmektedir.  $m$  sayıda D flip-flop grubu, D girişleriyle yolun  $m$  hattına bağlıdır. Eşzamanlı olarak saatlenebilmeleri için tüm flip-flopın saat girişleri birbirine bağlanmıştır. C=0 olduğu sürece, flip-flop, yol üzerindeki veriyi dikkate almazlar ve Q çıkışları değişmeden kalır. Bir cihazın, içerdiği veriyi flipflop'a aktarmak istediğini varsayalım. Devre önce, veriyi yol üzerine yerleştirir ve ardından flip-flop saatlenerek veriyi kendi üzerlerine mandallarlar. Saat sıfıra düştüğünde, veri flip-floplarda dondurulmuş olarak kalır.

## 2.5.3 Saatlenmiş flip-floplar

Flip-floplar, önce tipiyle (örn. RS, D, JK) ve sonra, saatleniş biçimiyle olmak üzere iki şekilde tanımlanırlar. Bir sayısal cihaz, saatlendiğinde, girişlerine cevap verir. Saatlenmiş devrelerin arkasındaki ana fikir; içerdiği mantık elemanlarının, ancak girişlerin geçerli olması halinde cevap vermeleri. Bu nokta Şekil 2.52'de gösterilmiştir.

A ve B işlemlerine ilişkin girişlerin,  $t=0$  zamanında, geçerli (yani işlemler tarafından işlenecek girişlerin doğru) olduğunu düşünelim. Burada işlemin ne olduğu fazla önemli olmayıp, bizi ilgilendiren, geçerli olan girişler arasındaki gecikmeye, geçerli çıkıştır. Şekil 2.52'de, A işlemi, iki birim gecikmeye, B işlemi de bir birim gecikmeye neden olmaktadır.



Şekil 2.52 Saat denetimli sistemler

A ve B işlemlerinin çıkışları, iki birimlik bir gecikmeye sahip olan C işlemine uygulanır. Açıkçası,  $t=0$ 'dan sonra en az 4 zaman birimi geçinceye kadar, A ve B girişlerinden dolayı, C'den istenen çıkış elde edilmez. C çıkışı, istenilen değere gelmeden önce birkaç kez geçersiz olmaktadır. Bu ise bir problem yaratır. C işleminin çıkışını gözleyen bir gözlemci, C den gelen veriye göre ne zaman hareket geçeceğini nasıl bilecektir?..

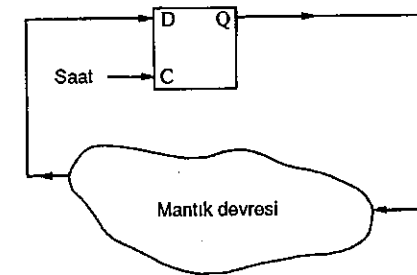
Eğer, C işleminin çıkışına bir D tipi flip-flop yerleştirilir ve sözcüğü,  $t=0$ 'dan beş birim süre sonra saatlenirse, istenen veri flip-flop üzerinde mandallanır ve bir sonraki darbe gelinceye kadar sabit tutulabilir. Saatlenmiş bir sistemde sayısal bilgiler, flip-floplarda tutulur ve bir mantık elemanları grubu üzerinde Şekil 2.43'dekine benzer biçimde işlenir. Saat darbeleri arasında, flip-flopların çıkışları mantık elemanları tarafından işlenir ve yeni veri değerleri flip-flopların girişine verilir.

Uygun bir zaman gecikmesinden (en yavaş işlemin tamamlanması için geçen süreden daha uzun bir süre) sonra flip-floplar saatlenir. Bu aşamada işlemlerin çıkışları, flip-floplara saat darbesi olarak uygulanır ve flip-floplar, bir sonraki saat darbesi gelene kadar sabit tutulur. Saatlenmiş bir sistemde tüm işlemler, genelde, her yeni saat darbesiyle birlikte, çalışmaya aynı anda başladığından senkron olarak adlandırılır. Asenkron sistemler ise, bir işlemin sona ermesinin diğer işlemi başlattığı sistemlerdir. Bir asenkron sistemin, buna karşılık gelen senkron bir sistemden daha hızlı olmak zorunda olduğu anlaşılmaktadır. Bununla birlikte, asenkron sistemler hem daha karmaşık hem de tasarımları senkron sistemlere göre daha zordur.

Üç tipte saatlenmiş flip-flop vardır.

1. Düzeye-duyarlı
2. Kenara-duyarlı
3. Ana-uydu

Bir düzeye-duyarlı saat, saat darbesi mantıksal 1 durumunda iken flip-flopu tetikler (bazı flip-floplar sıfırla tetiklenirler). Şekil 2.49'da verilen tetiklenmiş flip-flop, düzeye-duyarlı bir flip-floptur. Ancak, düzeye-duyarlı bir saat belirli uygulamalarda sorun çıkarabilir. Şekil 2.53'ü inceleyiniz.



Şekil 2.53 Geri besleme ve düzeye-duyarlı saat

D flip-flopunun çıkışı, mantık devresi aracılığıyla geriye D girişine dönecek biçimde beslenir. Eğer flip-flopun çıkışını şu anki Q olarak adlandıırırsak, bu Q değeri, D 'yi üretmek üzere mantık devresi aracılığıyla beslenir. Flip-flop saatlendiğinde de, D değeri çıkışa aktarılarak yeni Q değerini üretir.

Eğer saat, düzeye-duyarlı tipte ise, yeni Q değeri, mantık devresi aracılığıyla D'yi ve böylece çıkışı değiştirmiş olur. Bu olaylar zinciri bir osilasyon yaratarak, kendi kuyruğunu yakalamaya çalışan köpek örneğinde olduğu gibi sürer gider. Bundan sakınabilmek için, son derece dar bir saat darbesine gerek vardır. Böyle bir şey olamayacağından, geri-besleme sorununun çözümü için, kenara-duyarlı saat kullanılması önerilmiştir.

Kenar-tetiklemeli saate sahip bir flip-flop, saat darbesinin düzeyiyle (yani yüksek veya alçak) değil, saat sinyalinin yüksekte alçığa veya alçaktan-yükseğe geçişi ile tetiklenir. Bunlardan birincisine pozitif veya yükselen kenara-duyarlı, ikincisine de negatif veya alçalan kenara-duyarlı saat adı verilir. Birçok darbenin yükselen (veya alçalan) kenarı 10 nsn den az bir süreye sahip olduğundan kenar tetiklemeli saatler, son derece kısa süreli darbeye tetiklenen düzeye-duyarlı bir saat olarak düşünülebilir. Bu nedenle, Şekil 2.53'de sözü edilen problem ortaya çıkmaz, çünkü yeni çıkışın, geri dönüp girişi besleyebilmesi için gereken süre, bir kenarın yükselme süresinden çok daha uzundur. Yani bir kenarın yükselmesi için gereken süre içinde sinyal geriye beslenemez. Nanosaniye (nsn), saniyenin bin milyonda birine ( $10^{-9}$ ) eşittir.

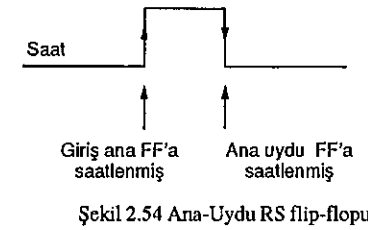
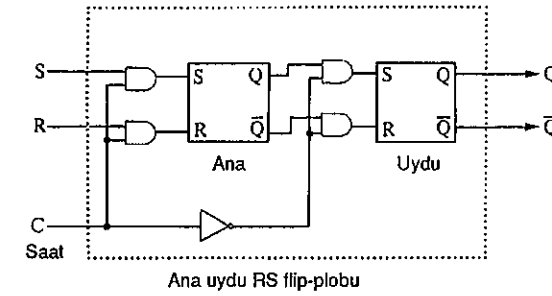
Kenar tetiklemeli flip-flopın yetersiz kaldığı bazı durumlar vardır. Bu, büyük ölçüde saat farklılığından kaynaklanmaktadır. Eğer bir sayısal sistemde birden fazla sayıda kenar tetiklemeli flip-flop, darbenin aynı kenarıyla tetikleniyorsa, flip-flopın tetiklendiği gerçek zamanlar birbirinden farklı olabilir.

Bu farklılık (saatteki sapma) çok küçük (birkaç nsn) olup, saat darbesinin her bir flip-flopa ulaşmak için izlediği yolun farklı olmasından kaynaklanmaktadır. Elektrik palsler, devrelerde ışık hızının biraz altında nanosaniyede 30 cm gibi bir hızla hareket ederler. Herbir flip-flop, saat darbesi kaynağından eşit uzaklığa yerleştirilmedikçe ve herbir yol üzerindeki mantık elemanların oluşturduğu gecikmeler özde olmadıkça, saat darbelerinin flip-floplara varış anları farklı olacaktır. A flip-flopunun çıkışının B flip-flopuna, girişine bağlı olduğunu ve saat darbesinin iki devreye de birlikte uygulandığını varsayalım. İdealde, saatleme anında A'nın eski çıkışı B'ye saat darbesi olarak uygulanır. Eğer, tasarımın kötülüğünden veya şansızlık gibi nedenlerle A flip-flopunun, B flip-flopundan birkaç nanosaniye önce tetiklenirse, B devresi bunu, A'nın yeni çıkışı olarak algılayacak ve eski A çıkışı olarak

görmeyecektir. Böylece A, sanki ayrı bir saat tarafından daha önce tetikleniyormuş gibi görünecektir. Bu sorunun bir çözümü olarak, saatlenmiş ana-uydu flip-flopları, geliştirilmiştir.

### Ana-Uydu flip-flopu

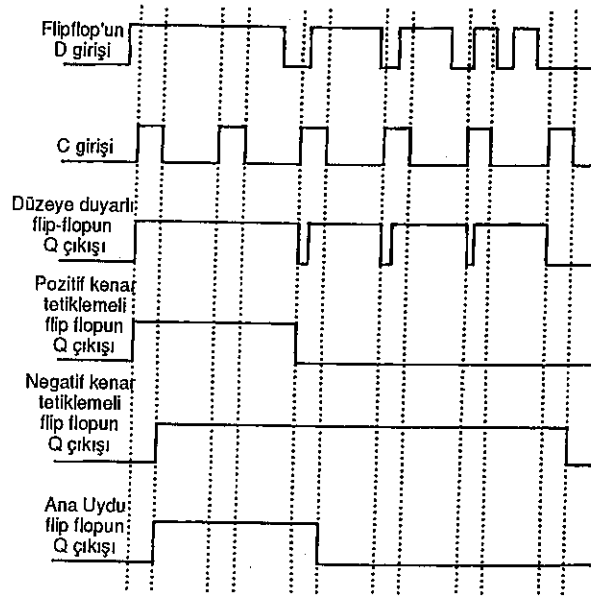
Ana-uydu (MS) flip-flopu, dış görünüşüyle tek bir flip-flop olup, iç yapısında ise seri çalışan iki flip-floptan oluşmuştur. Bunlardan birine Ana (Master), diğerine Uydu (Slave) adı verilir. Saat darbesinin yükselen kenarında, giriş verisi, Ana flip-flop üzerine mandallanır (bkz: Şekil 2.54). Bu noktada MS flip-flopunun çıkış uçları etkilenmez ve durum değiştirmezler.



Saatın düşen kenarında, Ana flip-flop verisi, uydu üzerine mandallanır ve çıkış uçları ancak bundan sonra durum değiştirebilir. Bu tip flip-flop'da giriş uçları ile çıkış uçları birbirlerinden tamamen yalıtılmıştır. Çünkü, flipflopun çıkışı, giriş durumları örneklenene ve dahili olarak mandallanana kadar değişmez. Kavram olarak ana-uydu flip-flopları, bir denizaltındaki veya uzay gemisindeki hava kilidi gibi davranır. Hava kilidi insanları, iki farklı basınç bölgesi arasında doğrudan geçişi engelleyecek biçimde, farklı basınç bölgelerine (havadan-vakuma veya havadan-suya) taşıyabilmek içindir. Bu, çıkışı, girişine doğrudan beslenmeyen flip-flopun bir benzeridir. Denizaltındaki bir hava kilidini çalıştırmak için dalgıç, hava kilidini açar, içeri girer ve kapağı arkasından kapatır. Dalgıç artık, dışarıdaki sudan ve denizaltındaki havadan yalıtılmış durumdadır. Denizaltına açılan kapağı

açtığı zaman, içeri adımı atar ve arkasındaki hava kilidini kapatır.

Üç farklı tipte, saatlenmiş flip-flopların nasıl çalıştığını anlamak için, Şekil 2.55'i inceleyiniz, burada, verilen bir giriş dalgabıçımı için, saatlenmiş D tipi flip-flop ilişkin üç farklı tip çıkış dalgabıçımı verilmiştir.



Şekil 2.55 Flip-flopların saatlenme modlarının karşılaştırılması

### 2.5.4 JK tipi flip-floplar

JK tipi flip-floplar, belki burada tanımlanmış olan en kullanışlı flip-floptur. Çünkü iki farklı moddan birinde çalışmak üzere düzenlenebilir veya programlanabilir. Tüm JK flip-flopları saatlenmiş olup, birçoğu ana-uydu ilkesine göre çalışırlar. JK flip-floplarına ilişkin doğruluk tabloları, Tablo 2.21 (a), (b)'de verilmiştir.

$J=K=1$  durumu dışında, J ve K'nın tüm değerleri için, JK flipflop, tam anlamıyla bir RS flip-flopudur. Burada J, kurma girişi, K ise sıfırlama girişi gibi davranır. Flip-flop, J ve K'nın her ikisi de birden doğru yani bir olduğu zaman, her saat darbesinde durum değiştirir. JK flip-flopunun bu özelliği onu, hemen hemen tüm sayıcıların kalbi konumuna getirir. Sayıcının çalışması, bir sonraki bölümde incelenmiştir. JK tipi flip-flopların mantık sembolü, Şekil 2.56'da verilmiştir.

Tablo 2.21 (a) JK flip-flop için doğruluk tablosu

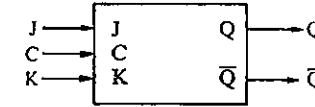
| Girişler |   |   | Çıkış |
|----------|---|---|-------|
| J        | K | Q | Q'    |
| 0        | 0 | 0 | 0     |
| 0        | 0 | 1 | 1     |
| 0        | 1 | 0 | 0     |
| 0        | 1 | 1 | 0     |
| 1        | 0 | 0 | 1     |
| 1        | 0 | 1 | 1     |
| 1        | 1 | 0 | 1     |
| 1        | 1 | 1 | 0     |

Değişmez  
Q'yu sıfır yapar  
Q'yu bir yapar  
Q dönüşümlüdür.

Tablo 2.21 (b) Doğruluk tablosunun alternatif şekli

| Girişler |   |    | Çıkış |
|----------|---|----|-------|
| J        | K | Q' | Q     |
| 0        | 0 | Q  | Q     |
| 0        | 1 | 0  | 0     |
| 1        | 0 | 1  | 1     |
| 1        | 1 | Q  | Q     |

Cebirsel form



Şekil 2.56 JK flip-flopunun gösterimi

### 2.6 Ardışık devreler

Mantık kapılarının birbiriyle bağlanarak, birleşik devreleri oluşturmasında olduğu gibi, toplayıcılar, çoğullayıcılar ve flip-floplar da birbirine bağlanarak, ardışık devreler adı verilen devreler oluştururlar. Biz burada iki özel tipte ardışık devre ile ilgileneceğiz: kaydırmalı kaydediciler ve sayıcılar.

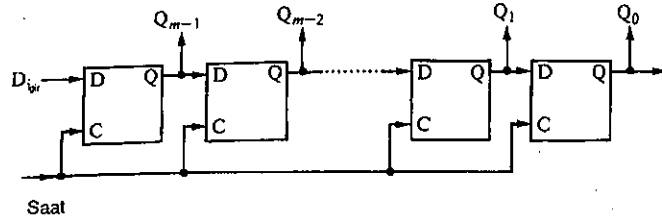
#### Kaydırmalı kaydedici

Daha önce gördüğümüz gibi,  $m$  tane D tipi flip-flop topluluğu  $m$ -bitlik sözcüğü saklayabilir. Kaydedici devresini biraz değiştirerek, bütün bitlerin bir basamak sağa kaydırıldığı, bir sağa kaydırmalı kaydedici yapabiliriz. Örneğin,

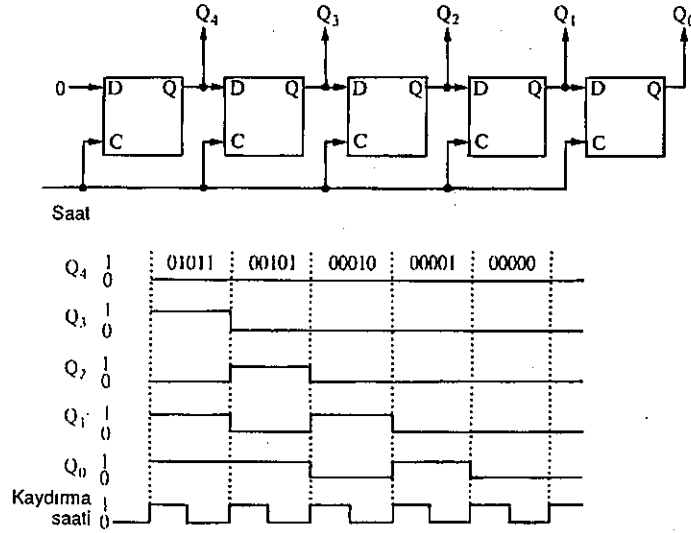
01110101

bit deseni sağa kaydırılınca, 00111010 olur.

Dikkat ediniz ki, sol tarafa bir 'sıfır' kaydırılmış olup, sağ taraftaki 'bir' kaybolmuştur. Yukarıda 'ikili desen' ifadesini kullanmamızın nedeni, daha sonra göreceğimiz gibi 01110101 baytı çok şeyi gösterebilmesidir. Bununla birlikte, bir desen, sabit-noktalı ikili bir sayıyı gösterdiğinde, bu sayıyı bir basamak sağa kaydırmak, sayıyı ikiye bölmek gibi bir sonuç doğurur.

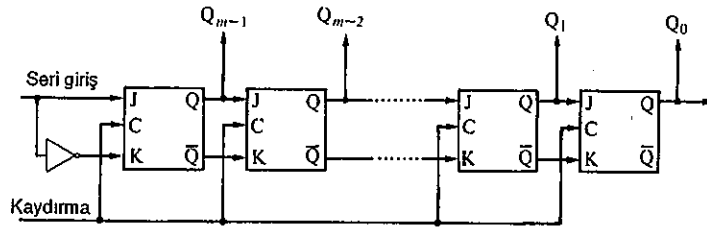


Şekil 2.57 Bir kaydırmalı kaydedici



modu) arasında anahtarlar (değiştirir). Tüm çoğullayıcıların denetim girişleri ya le/kaydır şeklinde etiketli olup, mod denetimini oluşturmak üzere birbirine bağlanmıştır. Bir değişkenin bu metodla etiketlenmesi, hat yüksekte-etkin olduğu zaman yüksele komutunun; hat alçakta-etkin olduğu zaman da kaydırma komutunun çalıştırılacağı anlamındadır.

JK flip-floplarından oluşmuş bir kaydırmalı kaydedici, Şekil 2.61'de gösterilmiştir. Bu düzenin nasıl çalıştığının açıklanmasını, bir alıştırmaya okuyucuya bırakıyorum. Bilgisayar komutlarının ele alındığı ilerideki bir kısımda görüleceği üzere birkaç farklı tipte (dairese, aritmetik, mantıksal) kaydırmalı kaydedici vardır.



Şekil 2.61 JK flip-floplarından oluşan bir kaydırmalı kaydedici

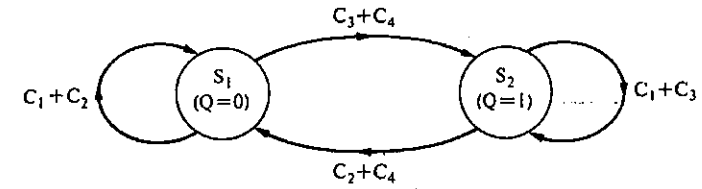
Uygulamada rastlanan diğer tip kaydırmalı kaydediciler, yukarıda tanımlanan temel tip kaydedicilerden türetilmişlerdir. Burada sadece, sağa kaydırmalı kaydedicilerden söz etmiş olmamıza rağmen, sola kaydırmalı kaydedicinin de tasarımı aynı derecede kolaydır.  $i$ 'inci katın girişi  $D_i$ ,  $i - 1$ 'inci katın çıkışına bağlanır ve her saat darbesinde  $Q_i \leftarrow Q_{i-1}$  olur. Bir önceki örnekteki terimlerle gösterirsek;

01110101,  
11101010 durumuna gelir.

#### Sayılar (Counters)

Sayılar, tek bir saat girişi ve birçok çıkışı olan özel bir tipte ardışıl devredir. Sayıcı, her saatlemede, bir veya daha fazla çıkış durum değiştirir. Bu çıkışlar  $N$  sayıda özgün değerli bir dizi oluşturur. Sayıcının çıkış uçlarında  $N$ 'inci değer gözlemlendikten sonra, bir sonraki saat darbesi ile dizi başlatılarak aynı çıkış tekrar elde edilir. Yani dizi döngüselidir. Örneğin bir sayıcıda, 01234501234501....veya 973109731097310 ... dizisi görüntülenebilir.

Sıralı devrelerin çalışmasını göstermek üzere sıklık kullanılan araçlardan biri de durum diyagramlarıdır. Kendi iç belleği (yani flip-flop) ve dış girişleriyle bir sistemin, kendi iç ve dış girişlerinin bir işlevi olan bir durumda bulunduğu söylenebilir. Durum diyagramları, verilen bir sistemin birkaç (veya tüm) olası değerlerini gösterir. Durumlardan herbiri, normalde bir daire ile gösterilir. Durumlar birbirine, bir durumdan diğerine dönüştüğünü gösteren- tek yönlü hatlarla bağlanırlar.



Şekil 2.62 JK flip-flopunun durum diyagramı

Şekil 2.62 JK flip-flopun durum diyagramını göstermektedir. Burada, sadece  $S_1$  ve  $S_2$  olmak üzere iki durum bulunmaktadır.  $S_1$ ,  $Q=0$  durumunu,  $S_2$  ise  $Q=1$  durumunu göstermektedir. Durumlar arası geçişler JK giriş değerleriyle belirlenir. Tablo 2.22'de dört durum  $C_1$ ,  $C_2$ ,  $C_3$  ve  $C_4$ , J ve K terimleriyle tanımlanmıştır.

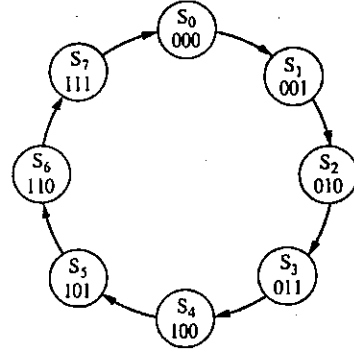
Tablo 2.22

| J | K | Durum |
|---|---|-------|
| 0 | 0 | $C_1$ |
| 0 | 1 | $C_2$ |
| 1 | 0 | $C_3$ |
| 1 | 1 | $C_4$ |

Flip-flop her saat darbesinde durum değiştirir. Şekil 2.62'den de görülebileceği üzere,  $C_3$  veya  $C_4$  durumu,  $S_1$ 'den  $S_2$ 'ye geçişe neden olur. Benzer biçimde,  $C_2$  veya  $C_4$  durumu da,  $S_2$ 'den  $S_1$ 'e geçiş sağlanmaktadır. Burada,  $C_4$  durumunun, hem  $S_1$ 'den  $S_2$ 'ye ve hem de  $S_2$ 'den  $S_1$ 'e geçişe neden olduğuna dikkat ediniz. Doğal olarak bu,  $J=K=1$  durumudur. Bazı koşullar durumların kendine dönmelerine neden olur, yani aslında herhangi bir değişim olmaz. Böylece  $C_1$  ve  $C_2$  durumları,  $S_1$  durumundaki sisteme uygulandığında, sistemin yine  $S_1$  durumunda kalmasına neden olmaktadır.

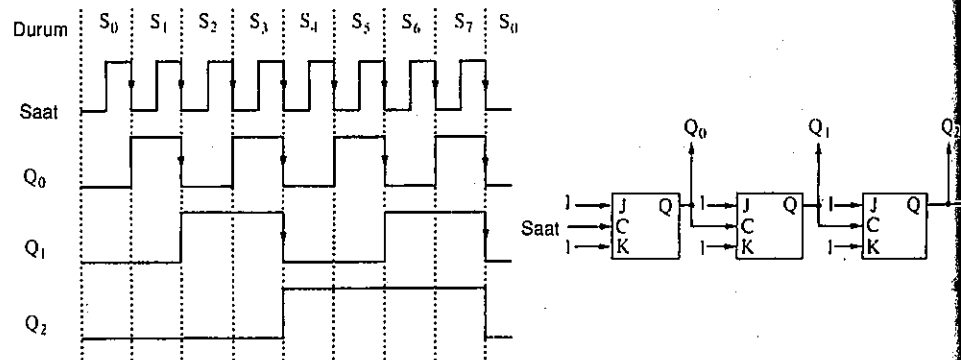
Üç-bitlik ikili sayıcının durum diyagramı Şekil 2.63'de verilmiştir. Bu diyagramda bir durumdan komşu duruma, tek bir yol bulunmaktadır. Yani sistem saatlendikçe durum -0 dan 7'ye doğal ikili sayıları göstererek-  $S_0$ 'dan  $S_7$ 'ye döner. Temel

ilkeleri; "flip-flopları,  $S_i$  durumundan  $S_{i+1}$  durumuna değiştirmek için hangi giriş koşulları gereklidir" sorusunu sormak olmasına karşın, sayıcıların pratikteki tasarımları gayet kapsamlı olabilir.



Şekil 2.63 Üç-bitlik bir sayıcının durum diyagramı

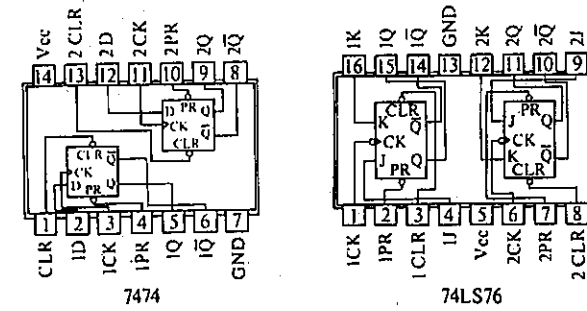
Bir asenkron ikili sayıcının tasarımı nisbeten oldukça basittir. JK flip-floplarından oluşmuş üç-bitlik ikili bir sayıcı, zamanlama diyagramıyla birlikte Şekil 2.64'de verilmiştir. Herbir flip-flopa ilişkin JK girişleri sabit mantıksal 1 düzeyine bağlanmıştır. Sonuç olarak, flip-flopa her saat darbesi uygulandığında, çıkışı durumu değiştirir. Flip-floplar, bir elemanın Q çıkışının, diğer elemanın saat girişini tetik-



Şekil 2.64 İkili sayıcı

leyeceği biçimde düzenlenir. Şekil 2.64'deki flip-floplar, saat darbesinin negatif kenarında çıkış değiştiren MS (Ana-uydu) flip-floplardır.

Sayısal bilgisayarlarda, sayıcılar birçok alanda uygulanabilirler. Bölüm 4'e bir gözetarsak, bilgisayarın kalbinde daima, yürütülecek bir sonraki komutun adresini tutan bir "program veya" "komut sayıcısı" bulunduğunu göreceğiz. Herbir komut



74LS74 İşlev Tablosu

| Ön ayar |   | Girişler |   | Çıkışlar       |                |
|---------|---|----------|---|----------------|----------------|
|         |   | Saat     | D | Q              | Q              |
| 0       | 1 | X        | X | 1              | 0              |
| 1       | 0 | X        | X | 0              | 1              |
| 0       | 0 | X        | X | 1              | 1              |
| 1       | 1 | ↑        | 1 | 1              | 0              |
| 1       | 1 | ↑        | 0 | 0              | 1              |
| 1       | 1 | 0        | X | Q <sub>0</sub> | Q <sub>0</sub> |

74LS76 İşlev Tablosu

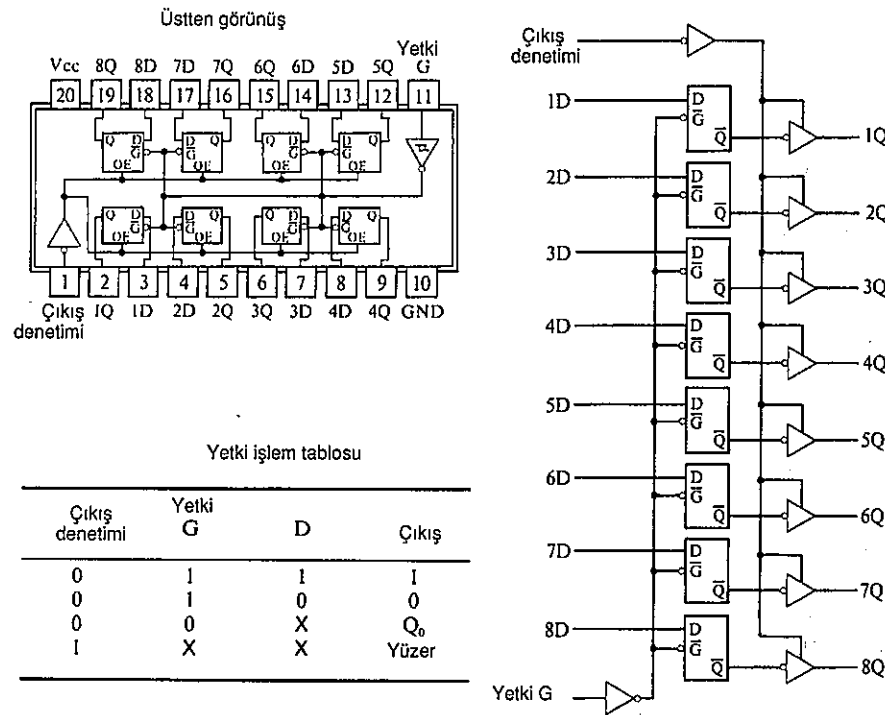
| Ön ayar |   | Girişler |      | Çıkışlar |   |                |
|---------|---|----------|------|----------|---|----------------|
|         |   | Sil      | Saat | J        | K | Q              |
| 0       | 1 | X        | X    | X        | X | 1              |
| 1       | 0 | X        | X    | X        | X | 0              |
| 0       | 0 | X        | X    | X        | X | 1              |
| 1       | 1 | ↓        | 0    | 0        | 0 | Q <sub>0</sub> |
| 1       | 1 | ↓        | 1    | 0        | 0 | 0              |
| 1       | 1 | ↓        | 0    | 1        | 0 | 0              |
| 1       | 1 | ↓        | 1    | 1        | 1 | Dönüşümlü      |
| 1       | 1 | 1        | X    | X        | X | Q <sub>0</sub> |

Şekil 2.65 DTL paketleri halinde sağlanabilen iki flip-flop

yürütüldükten sonra, program sayacı bir arttırılır. Buna program sayacı denmesinin nedeni, programı saydığından değil, yapısında bir sayıcı devresi ya da işlevini bulundurmasındandır.

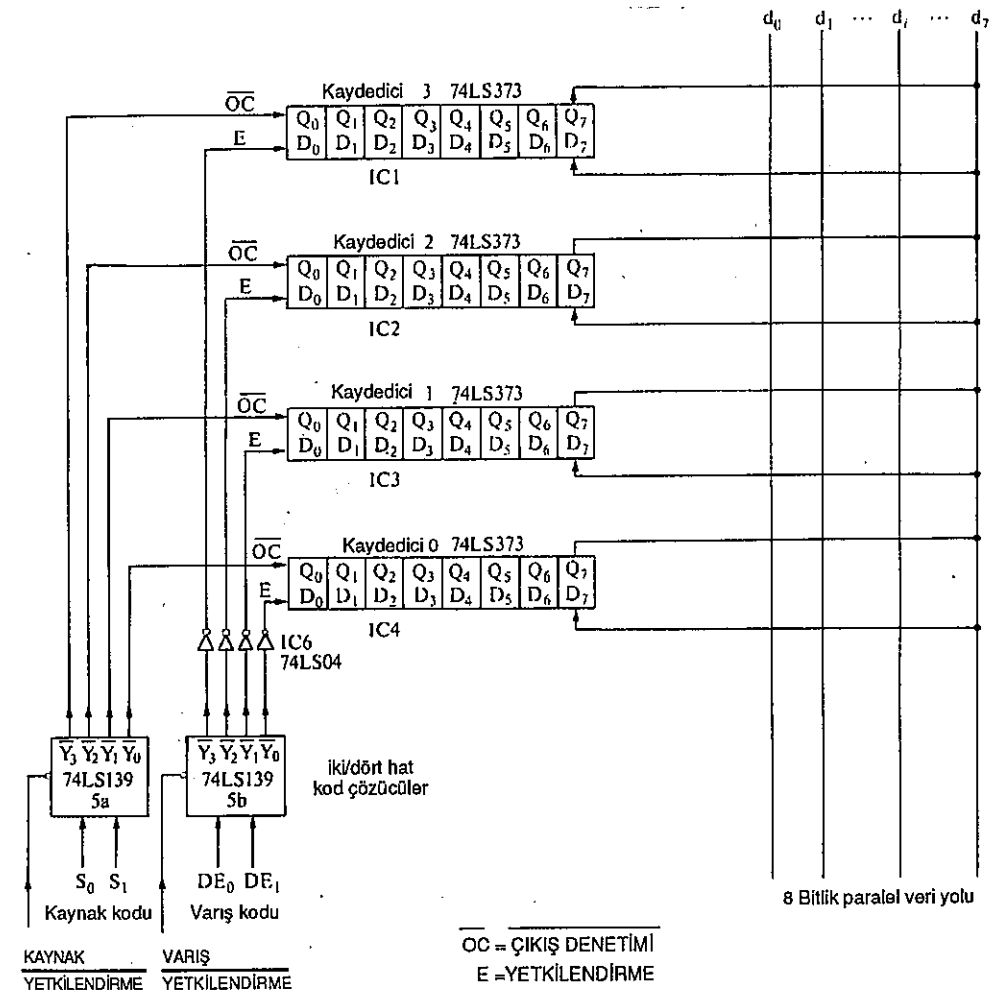
### 2.6.1 Özel Amaçlı ardışık mantık elemanları

Yariletken üreticileri, nasıl bir dizi özel-amaçlı birleşik devreleri tek bir paket içinde üretiyorlarsa aynı şey ardışıl mantık elemanları için de geçerlidir. Eğer bir fark varsa, o da, özel-amaçlı ardışıl devrelerin, sayıca özel-amaçlı birleşik devrelerden daha fazla olduğudur. Şekil 2.65'da, piyasadan sağlanabilen bazı flip-floplar verilmiştir. Flip-floplar, tüm sıralı devrelerin temel yapı taşlarıdır. Bunlar genellikle, bu bölümde şimdiye kadar sözü edilenlerden daha karmaşıktırlar. Gerçek devreler, günlük yaşantımızdaki gerçek problemlerin çözümünü sağlarlar. Belirtmiş olduğumuz üzere bir flip-flopun çıkışı, o andaki girişlerinin ve bir önceki çıkışının bir işlevidir. Bir flip-flopa ilk enerji uygulandığı anda ne olur? Yanıt gayet basittir. Q çıkışı, Q'yu 1 ve 0 değerini almaya zorlayacak herhangi bir giriş olmadığını varsayarak rastgele bir durum alır.



Şekil 2.66 74LS373 sekizli kaydedici

Rastgele durumlar, Monte Carlo'daki kumar masalarında işe yarayabilir. Fakat bir nükleer reaktör ilk defa enerjileneceği zaman hiçbir yardımları olmaz. Bu nedenle, onları belirli duruma getiren özel denetim girişli flip-flop'lar üretilmiştir. 7474 devresi, ikili pozitif-kenar tetiklemeli D tipi flip-flop olup Şekil 2.65'de gösterildiği üzere -preset (önayar) ve clear (silme) ( $\overline{PR}$  ve  $\overline{CLR}$  olarak kısaltılır) olmak üzere iki alçakta etkin denetim girişine sahiptir. Normal çalışmada,  $\overline{PR}$  ve  $\overline{CLR}$  mantıksal 1 durumunda kalır. Eğer  $\overline{PR}=0$  olursa Q çıkışı mantıksal 1'e kurur. Eğer  $\overline{CLR}=0$  ise Q çıkışı mantıksal 0'a sıfırlanır. RS flip-flopunda olduğu gibi  $\overline{PR}=\overline{CLR}=0$



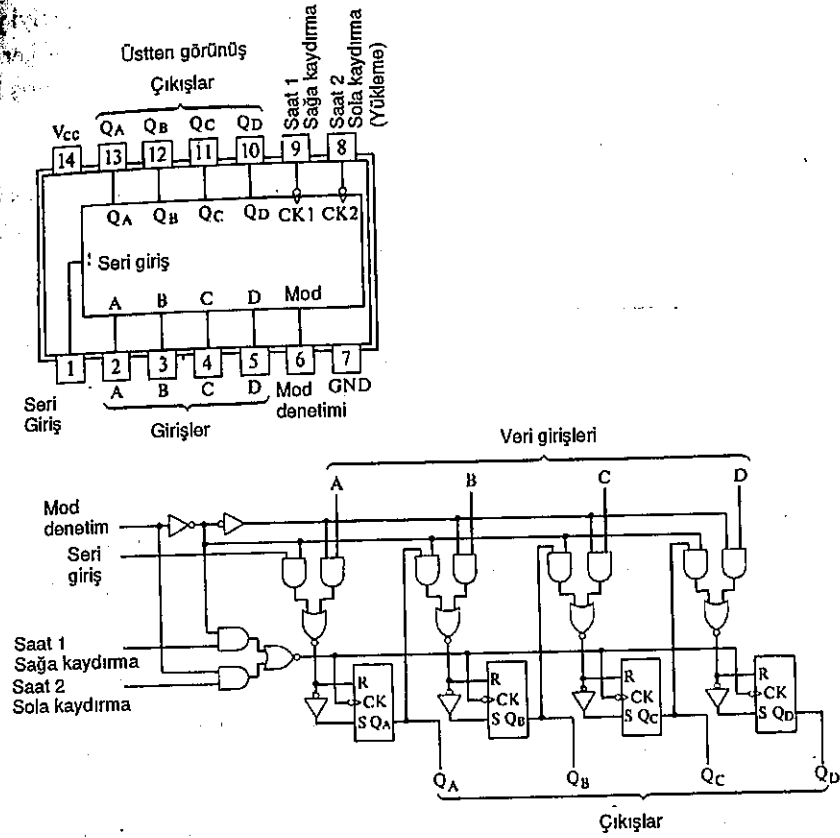
Şekil 2.67 74LS373 Sekizli D flip-flopu kullanan bir kaydedici dizisi

durumuna izin verilmez. Bu girişler, flip-flopun tüm diğer giriş eylemlerini barındırdığından bir anlamda koşulsuz girişlerdir. Birçok flip-floptan oluşmuş bir sayı sistemde, flip-flopların, enerjilendiklerinde kurulması veya sıfırlanması zorunludur. Flip-flopların  $\overline{PR}$  ve  $\overline{CLR}$  hatları ortak bir  $\overline{SIFIRLA}$  hattına bağlı olup, bu enerji verildikten hemen sonra tek bir darbeye, çok kısa bir süre içinde mantık seviyesine getirilir.

Flip-flopların dört genel uygulama alanı vardır: tek bir bitin (örneğin bir toplama işlemindeki elde bitinin) saklandığı birimler, tüm bir sözcüğünün saklandığı (akümülatör veya yazmaç) birimler, kaydırmalı kaydediciler veya sayıcılar. Şekil 2.66'da 8 adet D flip-floptan oluşan 74LS373 kaydedicisi görülmektedir. Sekiz giriş/sekiz çıkışlı, iki güç kaynağı bacağı olan ve iki denetim girişi devre 20-bacaklı DIL halinde piyasadan sağlanabilir. YETKİ girişi kenara duyarlı olup, yüksek olduğu zaman  $D_i$ 'nin değerinin  $Q_i$ 'ye aktarımını gerçekleştirir. Sekiz saatın tümü, içerde birbirine bağlı olduğundan, YETKİ girişi her bir flip-flop eşzamanlı olarak saatler. Bu tip devre, bir kaydedici dizisi oluşturmak için kullanılabilir, (Şekil 2.51). Diğer denetim girişi ÇIKIŞ DENETİMİ adını alır. ÇIKIŞ DENETİMİ = 0 olduğunda, flip-flop tam beklenen biçimde davranır. ÇIKIŞ DENETİMİ = 1 olduğunda da, sekiz Q çıkışının, elemanın içindeki çıkış bacaklarıyla olan bağlantısı kesilir. Yani 74LS373'nin üç durumlu çıkışı vardır ve ÇIKIŞ DENETİMİ, yonganın çıkış devrelerini, yolu sürmediği sürelerde kapatmaktadır.

Şekil 2.67'da, bu yonganın, bir bilgisayarda veya benzer bir sayısal sistemde nasıl kullanılabileceği gösterilmiştir. Dört 74LS373 devresi, sekiz-bitlik kaydedicinin girişleri ve 8-bitlik yola çıkışları bulunmaktadır. Dikkat etmelisiniz ki her bir flip-flopa ilişkin  $D_i$  ve  $Q_i$  hatları, yolun i'nci hattına bağlıdır. 74LS139'un bir yarısı, 2-hat/4-hat kod çözücü olup iki-bitlik ikili değerin kodunu çözerek, kaydedicilerin ÇIKIŞ DENETİMİ girişine bağlı dört hattan birini seçer. 74LS139 entegre devresi, iki tane 2-hat/4-hat kod çözücüsünü tek bir 16-bacaklı DIL içinde vermekte olup, Kısım 2.32'de verilen 74LS138'e benzemektedir. IC5a'daki iki-bitlik ikili kod, çıkış hatlarından birinin düzeyini düşürerek ilişkin kaydedicinin verisini, veri yolu üzerine yerleştirebilmesini sağlar. Örneğin, IC5a'nın girişindeki kod 01 ise, kaydedici 1'in çıkışı ertelenir ve kaydedici 1'in içeriği yola çıkarılır. Diğer tüm kaydedicilerin çıkışları, dahili olarak bağlantıları kesilmiş durumda bulunurlar.

İkinci 2-hat/4-hat kod çözücüsü IC5b, dört tersleyici ile birlikte, iki-bitlik varış-kodunu, dört yüksekte-etkin birine dönüştürür. IC5b girişindeki dört olası koddan her biri için kaydedicilerin YETKİ girişine bağlı bulunan dört hattan biri yükseğe giderek bu kaydediciyi saatler.



| İşlev Tablosu |                |       |      |                |                |                |   |                 |                 |                 |                 |
|---------------|----------------|-------|------|----------------|----------------|----------------|---|-----------------|-----------------|-----------------|-----------------|
| Girişler      |                |       |      |                |                |                |   | Çıkışlar        |                 |                 |                 |
| Mod denetimi  | Saat darbeleri |       |      | Paralel        |                |                |   | Q <sub>A</sub>  | Q <sub>B</sub>  | Q <sub>C</sub>  | Q <sub>D</sub>  |
|               | 2 (L)          | 1 (R) | Seri | A              | B              | C              | D |                 |                 |                 |                 |
| 1             | 1              | X     | X    | X              | X              | X              | X | Q <sub>A0</sub> | Q <sub>B0</sub> | Q <sub>C0</sub> | Q <sub>D0</sub> |
| 1             | ↓              | X     | X    | a              | b              | c              | d | a               | b               | c               | d               |
| 1             | ↓              | X     | X    | Q <sub>B</sub> | Q <sub>C</sub> | Q <sub>D</sub> | d | Q <sub>Bn</sub> | Q <sub>Cn</sub> | Q <sub>Dn</sub> | d               |
| 0             | 0              | 1     | X    | X              | X              | X              | X | Q <sub>A0</sub> | Q <sub>A0</sub> | Q <sub>A0</sub> | Q <sub>A0</sub> |
| 0             | X              | ↓     | 1    | X              | X              | X              | X | 1               | Q <sub>An</sub> | Q <sub>Bn</sub> | Q <sub>Cn</sub> |
| 0             | X              | ↓     | 0    | X              | X              | X              | X | 0               | Q <sub>An</sub> | Q <sub>Bn</sub> | Q <sub>Cn</sub> |
| ↑             | 0              | 0     | X    | X              | X              | X              | X | Q <sub>A0</sub> | Q <sub>B0</sub> | Q <sub>C0</sub> | Q <sub>D0</sub> |
| ↓             | 0              | 0     | X    | X              | X              | X              | X | Q <sub>A0</sub> | Q <sub>B0</sub> | Q <sub>C0</sub> | Q <sub>D0</sub> |
| ↓             | 0              | 1     | X    | X              | X              | X              | X | Q <sub>A0</sub> | Q <sub>B0</sub> | Q <sub>C0</sub> | Q <sub>D0</sub> |
| ↑             | 1              | 0     | X    | X              | X              | X              | X | Q <sub>A0</sub> | Q <sub>B0</sub> | Q <sub>C0</sub> | Q <sub>D0</sub> |
| ↑             | 1              | 1     | X    | X              | X              | X              | X | Q <sub>A0</sub> | Q <sub>B0</sub> | Q <sub>C0</sub> | Q <sub>D0</sub> |

Şekil 2.68 74LS373 kaydırmalı-kaydedici

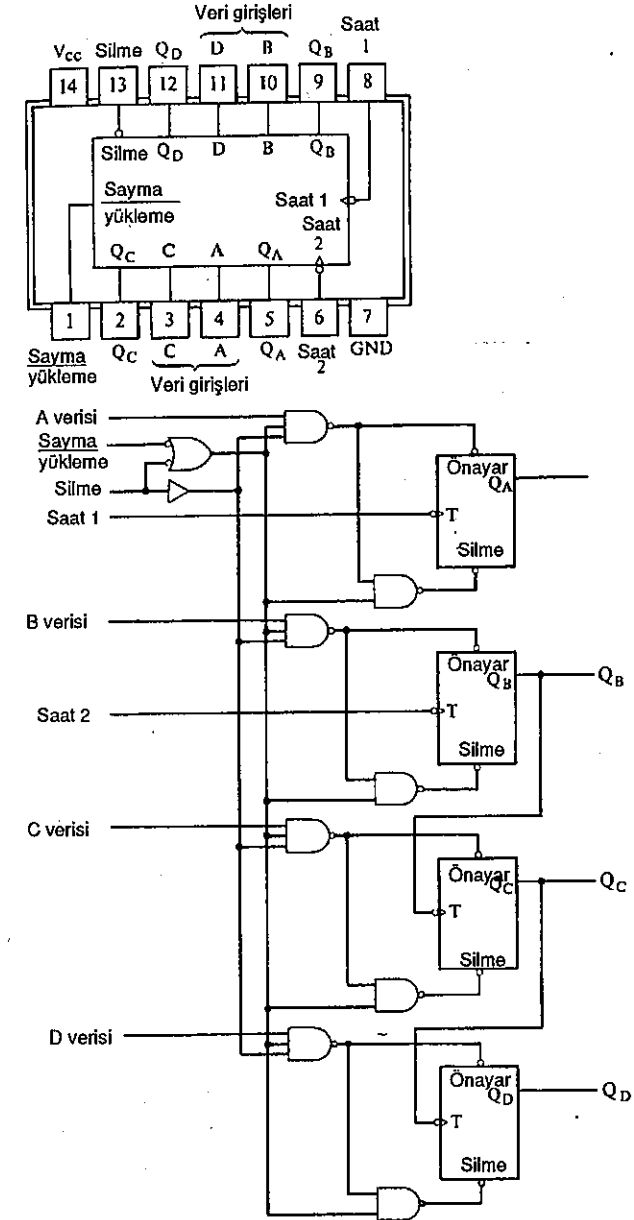
IC5a'nın girişi, kaynak kodu olarak, IC5b'nin girişi ise varış kodu olarak adlandırılır. Her iki kod çözücü de yetkilendirildiklerinde, kaynak kodu, veriyi yol çıkaracak kaydediciyi ve varış kodu da, veriyi alacak kaydediciyi seçer. Eğer kaydedici 1'in içeriği, kaydedici 3'e kopyalanacaksa, kaynak kodu 01 ve varış kodu 11 kurulur ve iki kod çözücü yetkilendirilerek veri aktarımı yapılır.

Özel amaçlı ardışık devrenin başka bir tipi de kaydırmalı kaydedecedir. Şekil 2.68'de, 4 bitlik, paralel erişimli, iki yönlü kaydırmalı kaydedici 74LS95 gösterilmektedir. Aygıtın hem bacak konfigürasyonunu, hem de iç düzenini vermektedir. Kullanıcı, kaydırmalı kaydediciye bacakları aracılığıyla erişir ve devrenin iç parçaları arasında bağlantı yapamaz. Gerçekte iç yapı, kitapta verilen devreden farklı olabilir. Bu devre gibi davrandıkça, mantıksal işlevin duyarlı bir şekilde nasıl gerçekleştirildiğinin bir önemi yoktur. 74LS95 özellikle, geniş kapsamlı bir kaydırmalı kaydedici olup, aşağıdaki işlevlere sahiptir.

1. **Paralel yükleme** Kaydırmalı kaydediciye yüklenecek dört bitlik veri; paralel girişlere uygulanmış, mod denetim girişi mantıksal 1'e kurulmuş ve saat darbesi ise, 2 numaralı saat girişine uygulanmış durumdadır. Verilerin sayıcı yüklenmesi, 2 numaralı saat darbesinin düşen kenarında gerçekleşir.
2. **Sağa-kaydırma** Sağa kaydırma, mod denetim girişinin mantıksal sıfıra kurulması ve saat darbesinin, 1 numaralı saat girişine uygulanmasıyla gerçekleştirilir. Sağa kaydırma, saat darbesinin düşen kenarında gerçekleşir.
3. **Sola-kaydırma** Sola kaydırma, mod denetim girişinin mantıksal sıfıra kurulması ve 2 numaralı saat girişine darbe uygulanmasıyla gerçekleştirilir. Sola kaydırma, saat darbesinin düşen kenarında gerçekleşir. Sola-kaydırma işlemi, her bir flip-flop çıkışlarının kendisinden bir önceki flip-flopun paralel girişlerine bağlantılı olmasını ve seri düzendeki verilerin D girişinden giriş yapmalarını gerektirir.

Burada tanımlanan özel-amaçlı ardışık devrelerin sonuncu türü sayıcılardır. Çok çeşitli sayıda sayıcı mevcuttur, Şekil 2.69'da dört-bitlik önayarlanabilir bir sayıcı olan 74177, tipik bir sayıcı örneği olarak verilmektedir. Şekil 2.69'da, iç düzenlemesiyle birlikte sayıcının bacak çıkışları da verilmiştir. Şekil 2.69'da verilen sayıcının T tipi bir flip-flop olduğuna dikkat ediniz. T simgesi, "toggle(=dönüşümlü)" çalışma biçimini ifade eder, çünkü çıkışlar, flip-flopun her saat darbesiyle tetiklenmesi sonucu durum değiştirmektedir. T tipi flip-floplar;  $J=K=1$  olması durumunda JK tipi flip-floplarla işlevsel eşdeğerde olmaktadır.

Sayıcılar, önce bir ikiye-bölme katı ve ardından da sekize-bölme katı olacak



Şekil 2.69 74177 4-bitlik ikili sayıcı

biçimde düzenlenmişlerdir. 16'ya bölmek (yani 0000'dan 1111'e kadar saymak) için, ilk katın ( $Q_A$ ) çıkışlarının, bir sonraki katın saatiyle (2 numaralı saat) bağlantılı olması gerekmektedir. Sayma girişi, 1 numaralı saate uygulanır ve saat darbesinin her düşen kenarında sayıcının içeriği bir artar.  $\overline{SILME}$  girişi,  $\overline{SILME}=0$  olduğunda

Q çıkışlarının tümünü sıfıra getirir. SAYMA/YÜKLEME girişi yüksek olduğunda normal sayma işlemine izin verir, alçak olduğunda da dört-bitin sayıcıya yüklemesini sağlar. Bu olanak, (sıfırdan farklı) bir I değerine kadar sayabilecek biçimde sayacı önayarlayabilmek için kullanılır, buradaki I, sayıcıya yüklenen değeri ifade eder.

74177'nin işlev tablosu

| Sıra | Çıkış          |                |                |                |
|------|----------------|----------------|----------------|----------------|
|      | Q <sub>D</sub> | Q <sub>C</sub> | Q <sub>B</sub> | Q <sub>A</sub> |
| 0    | 0              | 0              | 0              | 0              |
| 1    | 0              | 0              | 0              | 1              |
| 2    | 0              | 0              | 1              | 0              |
| 3    | 0              | 0              | 1              | 1              |
| 4    | 0              | 1              | 0              | 0              |
| 5    | 0              | 1              | 0              | 1              |
| 6    | 0              | 1              | 1              | 0              |
| 7    | 0              | 1              | 1              | 1              |
| 8    | 1              | 0              | 0              | 0              |
| 9    | 1              | 0              | 0              | 1              |
| 10   | 1              | 0              | 1              | 0              |
| 11   | 1              | 0              | 1              | 1              |
| 12   | 1              | 1              | 0              | 0              |
| 13   | 1              | 1              | 0              | 1              |
| 14   | 1              | 1              | 1              | 0              |
| 15   | 1              | 1              | 1              | 1              |

Not A: Q<sub>A</sub> çıkışı, saat girişi-2'ye bağlıdır.

Gelinen bu noktada genel-amaçlı bir sayısal bilgisayar tasarımında bize gereken tüm devreleri incelemiş bulunuyoruz.

## 2.7 Mantıksal tasarımda bazı pratik düşünceler

Mantıksal tasarım giriş dersini alan ya da Boole cebri konusunda temel bir kaynak okuyan herhangi bir öğrenci, sayısal bir sistem tasarımının birkaç Boole eşitliğinin basitleştirilmesinden, sonuçların devre diyagramına aktarılmasından, ve bunları biraya getirmekten fazla birşey olmadığını biçiminde bir izlenime kapılabilir. Ancak uygulamada bu böyle olmamaktadır.

Gerçek dünyada, tasarımcı bir dizi kısıtlama ile karşılaşmaktadır. Kullanabilecek mantık elemanları, yalnızca Boole cebri ile belirlenemez, bunun yanında gözardı edilemeyecek önemli başka faktörler de söz konusudur. Ayrıca tasarımcı, için her ne olursa olsun bir çözümle değil; fakat ucuza çıkacak bir çözümle ilgilenmek durumundadır. Klasik Boole cebrinin gözü kapalı uygulanması bizi maliyet-etkin

bir çözüme ulaştırmayacaktır. Bir Amerikalı astronota bir uzay kapsülü içinde olmanın nasıl bir duygu olduğu sorulmuş, onun da bu soruyu, "Herbiri, en düşük fiyatı veren firmalardan alınıp monte edilmiş, bir milyon kritik parçanın üzerine oturduğunuzda ya siz neler hissedersiniz?" şeklinde, yine bir soruyla yanıtladığı belirtilmişti. Bu bölümde, tasarım mühendisi için en çok yarar sağlayacak mantık elemanlarının bazı özelliklerine göz atacağız. Ancak bu incelemenin amacı; sayısal tasarımın tüm pratik yönlerini içine alacak ölçüde kapsamlı bir yaklaşım değildir, yalnızca sayısal tasarımın, mühendisi ilgilendiren bazı alanlarında yol gösterici bir nitelik taşımaktadır.

### Kapıların elektriksel karakteristikleri

En basit mantık elemanlarının bile olsa üretici veri tablolarına göz atıldığında, bir ya da daha fazla sayfa dolusu bilgi içerdiği görülecektir. Gerçekte bu veri tabloları, üreticinin kullanıcıya, yerine getireceğine ilişkin verdiği 'sözleri' ifade ederler. Tipik mantık elemanların temel parametreleri, Tablo 2.23'de verilmektedir.

Mantık elemanları, çok sayıda üretim aşamalarının bir sonucu olarak ortaya çıkmaktadırlar. Temel mantık elemanlarından biri de, TTL (Transistör-Transistör Mantık)'in özel bir türü de düşük güçlü Schottky TTL olarak adlandırılır. Gerçek üretim ayrıntıları ve çeşitli mantık ailelerinin özellikleri bu kitabın kapsamını aşmaktadır.

Çalışma sıcaklığının değişim aralığı tasarımcıya, bu donanımı, içerdiği mantık elemanlarına ilişkin karakteristik değerleri, belirtilen değişim aralığı dışına çıkarmasızın ne kadar soğuk ya da ne kadar sıcak ortamlarda bırakabileceğini bildirilmektedir. Sıcaklık faktörü, ortam sıcaklıkları yaklaşık 10 °C ile 25 °C arasında olan kapalı mekanlarda çalışacak donanımların tasarımcısı için çok fazla önem taşımaz. Bununla birlikte otomobil üreticileri için mantık elemanlarının ısı davranışları en önemli parametrelerden biri olabilir. Çünkü arabalarını Alaskadan Sahra çöllerine kadar çok farklı iklim koşullarının geçerli olduğu ortamlarda satmak durumunda dırlar.

Mantık elemanlarının gerilim karakteristikleri, en kötü-durumda giriş ve çıkış değerlerini belirtirler. Örneğin  $V_{OL}$  çıkışın alçak düzeyde olması durumunda maksimum çıkış gerilim değerini belirtir. Eğer  $V_{OL}$  0.4 V olarak verilmişse bu, mantıksal sıfır durumu için çıkışın 0 ile 0.4 V arasında olabileceğini gösterir. Alçak düzeyli çıkış gerilimi, elbette aygıtın arızalı olmaması koşuluyla, kesinlikle 0.4 V'un üzerinde bir değer alamayacaktır.

Tablo 2.23 Bir mantık elemanının (düşük güçlü Schottky TTL) örnek verileri

|                                      |                         |
|--------------------------------------|-------------------------|
| Sıcaklık değişim aralığı (ticari)    | = 0 °C - +70 °C         |
| Sıcaklık değişim aralığı (askeri)    | = -55 °C - +125 °C      |
| Çalışma gerilimi                     | = 5 V ± %5              |
| Maksimum değerler                    |                         |
| Saklama Sıcaklık aralığı             | -65 °C - +150 °C        |
| Kaynak gerilimi                      | -0.5 V - +7 V           |
| DC giriş akımı                       | -30 mA - +5.0 mA        |
| DC karakteristikleri                 |                         |
|                                      | Mtn. Tipik Maks. Birimi |
| $V_{OH}$ çıkış yüksek gerilim düzeyi | 2.7 3.4 Volt            |
| $V_{OL}$ çıkış alçak gerilim düzeyi  | 0.4 Volt                |
| $V_{IH}$ giriş yüksek gerilim düzeyi | 2.0 Volt                |
| $V_{IL}$ giriş alçak gerilim düzeyi  | 0.8 Volt                |
| $I_{IL}$ giriş alçak akım düzeyi     | -0.36 mA                |
| $I_{IH}$ giriş yüksek akım düzeyi    | 40 µA                   |
| $I_{OH}$ çıkış yüksek akım düzeyi    | -400 µA                 |
| $I_{OL}$ çıkış alçak akım düzeyi     | +8 mA                   |
| Anahtarlama karakteristikleri        |                         |
| $t_{LH}$ alçaktan-yükseğe gecikme    | 10 ns (maks.)           |
| $t_{HL}$ yüksektan-alçığa gecikme    | 12 ns (maks.)           |

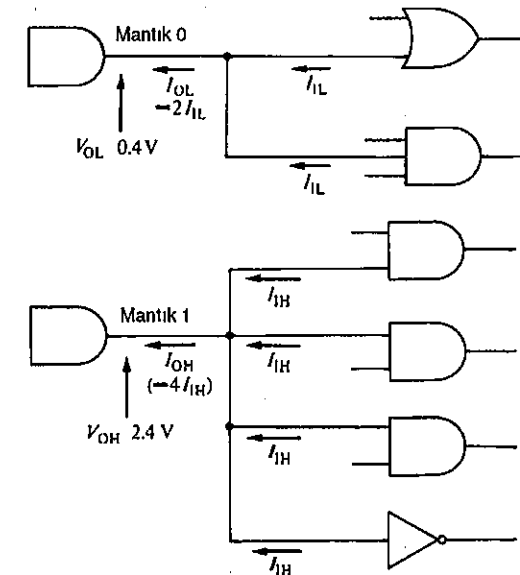
Benzer biçimde,  $V_{IL}$  değeri de, mantık elemanının mantıksal sıfır düzeyini güvenilir bir biçimde ayırt edebileceği maksimum giriş gerilimini gösterir.  $V_{IL}$  gerilim değerinin 0.8 V olması, 0 V ile 0.8 V arasındaki giriş geriliminin mantıksal sıfır düzeyi olarak algılanabileceğini garanti etmektedir.  $V_{IL}$  değerinin 0.8 V ve  $V_{OL}$  değerinin de 0.4 V olarak belirtildiğine dikkat ediniz. Bu iki değer arasındaki fark aygıtın "gürültü aralığı" olarak adlandırılır. Eğer alçak durumda bir çıkış değerinin 0.4 V ya da daha düşük olacağını biliyorsak ve giriş de alçak durumu olarak en fazla 0.8 V düzeyini alacaksa, herhangi bir hata ortaya çıkmaksızın çıkış değerinin üzerine 0.4 V'a kadar bir ekleme yapılabilir. Bu "ek 0.4 V" sistem içinde gürültüye neden olur. Burada gürültü, istenmeyen tüm sinyaller için genel adlandırmadır.

Elektriksel parametrelerin diğer önemli grubu da, kapılar arasında akan akımdır. İdealde kapılar arasında herhangi bir akım akmadığı düşünülür. Ancak, bir kapının girişine uygulanan bir mantık düzeyi algılayabilmesi için, bu kapıyı süren, yani bağlı olduğu çıkıştan, bir miktar akım çekmesi gerekmektedir. Örneğin, Tablo 2.23'den yüksek durumda kapıya uygulanan maksimum giriş akımı,  $I_{IH}$ 'ın, 40 µA ( $40 \times 10^{-6}$  A) olduğu görülebilmektedir.

Mantıksal bir durumunda bir çıkıştan akan maksimum akım,  $I_{OH}$ , - 400 µA olarak verilmektedir. Buradaki eksi işareti akımın kapıdan dışarı doğru aktığını gösterir.

Akım, mantıksal sıfır durumunda çıkışa doğru, mantıksal bir durumunda da çıkıştan dışarı doğru akar.  $I_{IH}$  ve  $I_{OH}$  arasında temel bir fark vardır.  $I_{IH}$ , bir kapının girişinden alınan akımı gösterirken,  $I_{OH}$  ise bir kapı devresinin çıkışından alınabilecek maksimum akımı göstermektedir. Bir kapının gerçekte çıkış akımı, daima 0 ile  $I_{OH}$  veya  $I_{OL}$  değerleri arasında olacaktır. Bir kapının çıkışının bir diğer kapının girişine bağlı olması durumunda, birinci kapının çıkış akımı, bağlı olduğu kapının giriş akımına tümüyle eşit olmalıdır. Elektroniğin temel yasası, bir yoldan akan bir akımın, yol boyunca her noktada aynı değeri taşıması gerektiğini öngörür. Şekil 2.70'de, iki girişi süren mantıksal sıfır çıkışlı bir kapı ile dört girişi süren mantıksal bir durumlu bir kapının elektriksel koşulları gösterilmiştir.

Bir tek çıkışa birkaç girişin bağlı olması durumunda, girişlerin çektiği toplam akım, çıkışın verebileceği maksimum akım değerini aşmamalıdır. Eğer  $n$  sayıda giriş varsa, güvenli çalışma için  $I_{OH}$  değeri,  $nI_{OH}$  değerinden daha büyük ya da bu değere eşit olmalıdır. Yukarıda verilen örnekte, yüksek düzeyli durumda,  $400/40=10$  girişi sürmeye yetecek akımı tek bir çıkış sağlayabilecektir. Kapıların elektriksel çalışma koşullarını belirleyen dört temel eşitlik aşağıda verilmektedir. Bunlar;

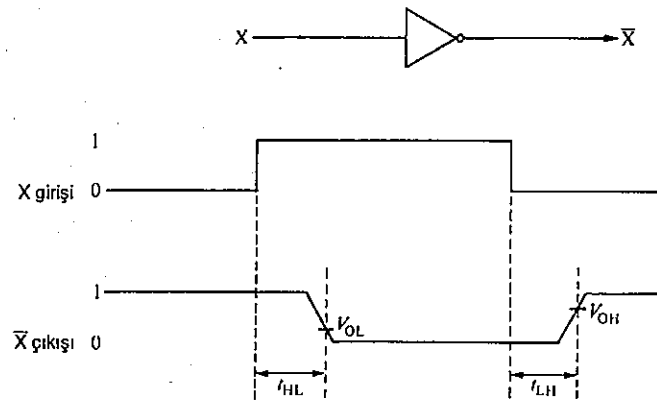


Şekil 2.70 Mantıksal sıfır durumunda (Sayfa 89'un alt kısmında verilen) ve (yukarıda verilen şekildeki) Mantıksal 1 durumunda kapıların elektriksel karakteristikleri

$$\begin{aligned}
 V_{OL} &< V_{IL}, \\
 V_{OH} &> V_{IH}, \\
 I_{OL} &> nI_{IL} \quad \text{tek bir çıkışa bağlı n sayıda giriş durumu için,} \\
 I_{OH} &> nI_{IH} \quad \text{tek bir çıkışa bağlı n sayıda giriş durumu için,}
 \end{aligned}$$

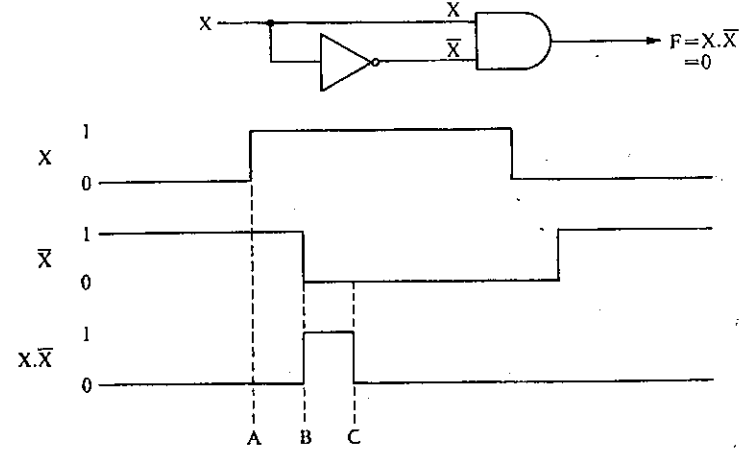
Mantık elemanlarının tümüne ilişkin önemli diğer bir özellik grubu da bu elemanların zamanlama karakteristikleridir. Şimdiye kadar genelde, eğer belli sayıda Boole koşulu, kapılardan oluşan bir düzenlemenin giriş uçlarına uygulanırsa (kapıların Boole eşitlikleriyle tanımlanmış bulunan) doğru çıkışının, hemen devrenin çıkışında gözükceği varsayılmıştır. Ancak bu böyle olmamaktadır. Genelde temel kapı devreleri, yaklaşık 10 nsn.'lik bir yayılma gecikmesine sahiptirler. On nanosaniyelik bir zaman parçası bizler için kavranamayacak kadar kısa olmasına karşın elektronik anlamda hiç de kısa sayılmaz. Bu gecikme, tasarımcıların ilgilenmek zorunda kalacakları en büyük sorunlardan biri durumundadır. Gecikmenin bazı etkilerini Kısım 2.5'de görmüştük.

Yayılma gecikmesinin tek bir tersleyici üzerindeki etkileri Şekil 2.71'de gösterilmektedir. Yükselen (dikey) ve düşen kenarları, keskin bir darbe, bir tersleyicinin girişine uygulanmıştır. Tersine çevrilen darbe çıkıştan, girişe oranla belli bir gecikme ile elde edilmiş olup, artık kenarları dikey değildir. Buradaki  $t_{HL}$  süresi, giriş darbesinin yükselen kenarıyla kapı çıkışının  $V_{OL}$  değerine ulaştığı nokta arasındaki gecikme süresidir. Benzer biçimde,  $t_{LH}$  de, girişin düşen kenarıyla çıkışın  $V_{OH}$  değerine ulaştığı süre arasındaki zamanı ifade eder.



Şekil 2.71 Bir tersleyicinin yayılma gecikmesi

Sinyallerin kapılar içinden geçişteki gecikme etkisinin sayısal sistemin çalışma hızını düşürdüğü söylenebilir. Maalesef yayılma gecikmesi, Şekil 2.72'de görüldüğü gibi, çok daha kötü etkilerde bulunur. Boole cebri kurallarına göre, VE kapısının çıkışı sürekli olarak sıfır olmak durumundadır.



Şekil 2.72 Bir kapıda oluşan yayılma gecikmesinin kenar etkisi

Şimdi de, kapının zamanlama diyagramlarını inceleyelim. A noktasındaki X girişi, sıfır düzeyinden bir düzeyine yükselir. Ancak, VE kapısına uygulanan  $\bar{X}$  girişi, eviricinin yayılma gecikmesine eşit bir sürede sıfıra düşmez. Sonuç olarak, VE kapısı girişlerinin her ikisi de kısa bir süre için doğru'dur ve çıkışları, (kendi iç gecikme süresinin ardından), B ile C noktaları aralığında mantıksal bir'e yükselir. VE kapısının çıkışında ortaya çıkan kısa süreli darbe çapak olarak adlandırılır ve sayısal sistemler için oldukça sorun yaratabilir. Bu sorunun iki çözümü vardır. Bunlardan biri, çapak'ortadan kaldırmak için Boole mantığına özel tasarım teknikleri uygulamaktır. Diğer çözüm ise, çıkışı bir flip-flop'a bağlamak ve flip-flop'u 'çapak'ın yok olmasından sonra saat darbesiyle tetiklemektir.

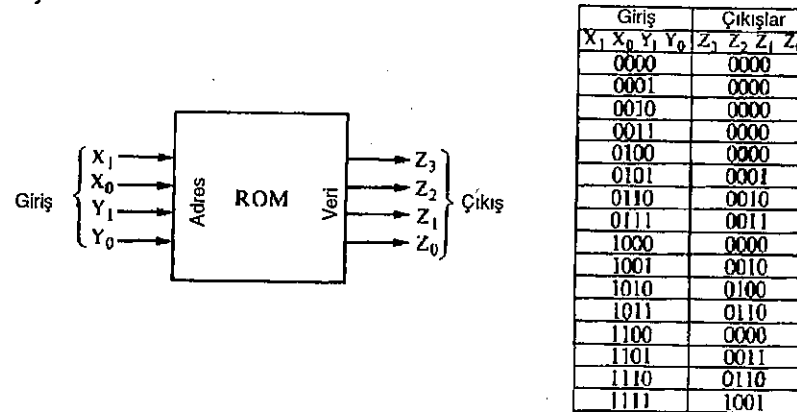
#### Sayısal tasarıma ilişkin ekonomik hususlar

Sayısal sistemlerin tasarımındaki ekonomik hususlar uygulamaya bağlı olarak çok değişik biçimlerde farklılıklar gösterir. Örneğin, bir uyduda karta-yerleşik bir bilgisayar tasarlayan bir mühendis, büyük ölçüde şu üç parametrenin en uygun biçime getirilebilmesiyle ilgilenir: ağırlık, güvenilirlik ve güç tüketimi. Bir bulaşık makinesi denetim birim tasarlayan bir tasarımcı ise, hemen hemen tüm dikkatini devrenin maliyeti en aza indirmek üzerine yoğunlaştırır.

Önemsiz olanların dışında tüm sayısal devreler, çok değişik tipte ve çok sayıda ka (iki-girişli VEYA, beş-girişli VE vb.) kullanmayı gerektirir, buna karşın tasarımı da, gereken entegre devre sayısını en aza indirebilmek için kullanacağı ka tiplerinin sayısını azaltmanın yolunu arayacaktır. Kimi zaman da sorunu çözme için, "Acaba, benim gerek duyduklarımın en yakın işlevi görecektir bir tümleşik dev üreticisi var mıdır?" sorusuna yanıt bulmaya çalışacaktır. Eğer yanıt evet ise, doğru dan raftan satın-alınabilen bir devre ya da ek bir mantık devresi kullanılabilir. Büyük yarı-iletken malzeme üreticileri günümüzde, çoğullayıcılardan sayı çarpım devrelerine kadar değişen çok geniş bir yelpazede temel geliştirme kitleri üretmi lerdir.

Devreyi gerçekleştirmek için bir diğer yaklaşım da, arama tablosu olarak bir sadece-okunur bellek (ROM) kullanmaktır. ROM'lar, içerdiği birbirinden farklı sayıda konum tanımlayan  $n$  sayıda adres giriş hattı bulunan bir birimlerdir. Her bir konum, erişim yapıldığında,  $m$  sayıdaki çıkış hattında  $m$ -bitlik bir değer üretir. Bu elemanın "sadece-okunur" biçiminde adlandırılması, verilen bir girişe karşılık alınan çıkış üzerinde kullanıcının herhangi bir değişiklik (örneğin yazma işlemi) yapamaması nedeniyledir.

Şekil 2.73'de, bir 16 x 4 ROM'un, Şekil 2.17(b)'de verilen çarpma devresini nasıl gerçeklediği gösterilmektedir. Dört adres girişleri olan  $X_1$ ,  $X_0$ ,  $Y_1$  ve  $Y_0$ , herbiri istenen sonuca karşılık gelen dört-bitlik sözcüğü içeren, olası 16 konumdan birini seçer.



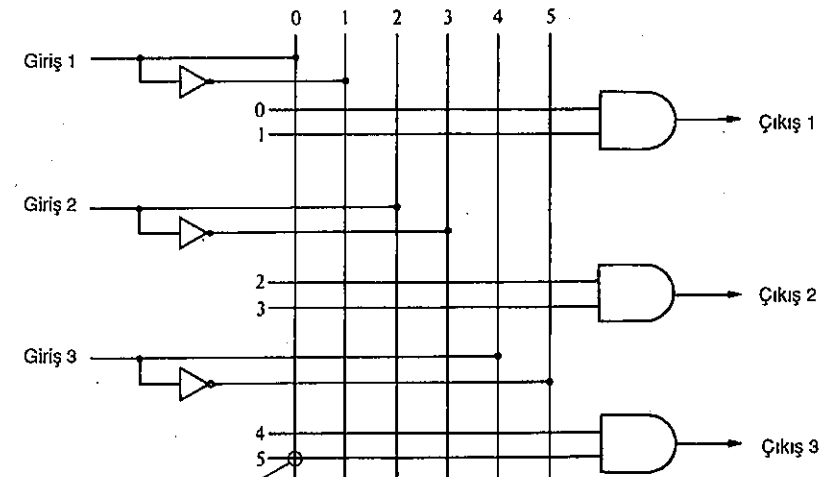
Şekil 2.73 Mantuk elemanlarının değiştirilmesi için bir ROM'un kullanılması

ROM üreticileri, bu onaltı konumdan herbirinin içine uygun çıkışları kaydederler. Örneğin, 1011 konumu,  $10 \times 11 (= 2 \times 3)$ 'e karşılık gelmekte olup bu konuma 0110 (=6) değeri kaydedilmiştir. ROM, devreyi değil doğrudan doğruluk tablosunu gerçeklemektedir. Çıkışa karşılık gelen girişlerin herbiri saklanmış durumdadır.

Ayrıca bu teknik, doğruluk tablosundan türetilmiş çarpımlar-toplamı ifadesini basitleştirmek için Boole cebrine de gerek duymaz.

Tüm bunlara karşın, tasarımcının kullanabileceği teknik, programlanabilir mantık devreleriyle ilgili olacaktır. Genel-amaçlı mantık dizileri (tek bir yonga üzerinde bulunan farklı tipteki kapı grupları) günümüzde üretilmiş bulunmaktadır. Kapılar, yonganın içinde bulunan "eriyen" tel bağlantılar yoluyla, çok çeşitli biçimde düzenlenebilirler. Sayısal tasarımcı, uygun aygıtı, üretici kataloglarından seçer ve Boole eşitliklerini, yonganın içerdği kapıların tiplerine uyarlayabilecek biçimde değiştirir. Ve ardından bu yongayı, kapılar arasında isteğine uygun biçimde bağlantı kuran özel programlama makinesindeki soketine takar.

Gerçekte, bu arabağlantılar mevcut bulunduğundan, programcı yeni bağlantılar yapmaz, ancak istenmeyenleri bozarak kaldırır. Bu, Şekil 2.74’de gösterilmiştir. Düşey ve yatay hatların birbirleriyle kesişme noktaları, eriyebilir incecik tel bağlantılardır. Eğer bu eriyebilir tel bağlantılar bozulmadan bırakılırsa, elektrik bağlantısı, düşey ve yatay hatları birleştirecek biçimde kalır. Eğer, yonganın programlanması sırasında yüksek bir akım darbesi bu tel bağlantılardan geçecek olursa, tel eriyeceğinden bağlantı da kesilmiş olacaktır. Programlanabilir mantık dizilerinin kullanılması bizi, çok az sayıda entegre devre paketiyle ve üretim



Numaralandırılmış yatay ve düşey hatların kesiştiği noktalarda istenen çıkışların alınabilmesi için bağlantılar yapılabilir.

Şekil 2.74 Programlanabilir dizi mantık devresinin basitleştirilmiş bir örneği

maliyetlerini büyük ölçüde azaltacak biçimde mantıksal fonksiyonları gerçekleştirilmeye götürür.

Bu işlem, bir adım daha ileri götürülebilir. Yonga üzerinde çok fazla sayıda kapı üretilmiş ancak, kapılar arasındaki arabağlantılar, üretim aşamasında son adıma gelinceye kadar yapılmamıştır. Yani, üretici, tek bir yonga üzerine belli sayıda kapı yerleştirir ve sonunda, kapılar arasında kurulacak bağlantıları müşterinin belirleyebilmesine olanak tanır. Şurası unutulmamalıdır ki müşteri, mevcut kapıların tümünü kullanmayabilir. Bu türde yongalar, bağımsız mantık dizileri (ULA) olarak adlandırılırlar ve yalnızca, büyük-ölçekli üretim söz konusu olduğunda maliyet-etkinlik kazanırlar. Çünkü yonga üretiminde kullanılacak makine ve ekipmanın ayarlanma maliyeti, hiç de azımsanacak bir büyüklükte değildir. Sinclair'in birkaç parçayla ZX81 ve Spectrum bilgisayarlarını üretebilmesini sağlayan işte bu bağımsız mantık dizileridir.

## 2.8 Bilgisayarlar ve güvenilirlik

Güvenilirlik, hem bir bilgisayarda ve hem de onun gerçek tasarımında kullanılacak bileşenlerin seçiminde temel etmenlerden biridir. Yerli tüketiciler, genelde güvenilirliğin öneminin doğrudan farkında değildirler. Bir fotoğraf makinesi almaya çıktığımda, satıcı "İyi bir parçadır bu, f1.4, kendinden-odaklamalı, otomatik ışık ayarına sahip, motorlu vb. der" ve ben bunu alırım. Fakat bir hafta sonra kırılır. Eğer patronum için bir bilgisayar alırsam, satıcıya ilgisiz veya anlamsız sorular sormak yerine, ürünün 'arızalar arası ortalama süre' (MTBF) ve 'onarımına kadar ortalama süre' (MTTR) gibi özelliklerine ilişkin sorular sorarım. Profesyonel donanımlar, genelde uzunca bir süre kesintisiz biçimde çalışabilmelidirler. Bazen, bu türde makinelerin arızaları hem pahalı ve hem de ödenmesi güç bedellere mal olabilmektedir. Bazen de arıza yapmaları, giderilmesi güç sonuçlara neden olmaktadır. Örneğin, çok kullanıcı ortamında kullanılan bir bilgisayarın arıza yapması, doğrudan gelirlerin düşmesine ve bunun yanında da itibar kaybına neden olur. Bilgisayarla çalışan bir nükleer reaktörün denetim kollarındaki toptan bir arızayı düşünmek bile ürktüçüdür. Güvenilirlik kavramına geçmeden önce, son cümlede verilen 'toptan' sözcüğüne dikkatinizi çekmek isterim- ileride, kısmi sistem arızalarını tolere edebilecek tiplerin seçimiyle, sistemlerin nasıl daha güvenli olabileceklerini göreceğiz.

Güvenilirlik konusunda yaygın biçimde inanılan iki teori vardır: Bunların ilki, tüm endüstriyel aygıtların içlerinde kötü ruhların yaşadığı ve amaçlarının, en olmadık zamanlarda bir arızaya neden olduğu şeklindedir. İkincisi, bu türde aygıtların, tam

garanti süresinin sona erdiği anda arızalandıklarıdır. Ne bu teoriler tümüyle doğrudur -çünkü aygıtlar, çok çeşitli nedenlerle arızalanabilirler, ve ne de cihazların ne zaman arızalanacağını söyleyebilmek de hemen hemen imkansızdır. Ne var ki, çok sayıda (özdeş) cihaz söz konusu olduğunda, bir ortalama aygıttan söz edebilmek mümkün olacaktır. Bir cihazın güvenilirliği,  $1 - p$  olarak tanımlanır, burada  $p$ ; verilen bir süre içinde cihazın arızalanabilme olasılığıdır. Örneğin, belirli bir parçasının bir yıl içinde arıza yapma şansı on'da bir ise, cihazın, yüzde doksan oranında güvenilir olduğu söylenebilir.

Elimizde, çok fazla sayıda parçanın olduğu, bunların  $N$  kadarının, belli bir  $t$  süresi sonra, halen çalışmakta, geriye kalanların ise arızalandıklarını düşünelim.  $N$ 'nin değişimi, zamana bağlı olarak  $dN/dt$  ile ifade edilebilir ve  $N$  azalmakta olduğundan değeri eksidir.  $N$ 'nin azalma hızının çalışan cihaz sayısına oranı, arıza oranı olarak adlandırılır ve aşağıda verilen biçimde ifade edilir:

$$L = - \frac{dN}{dt} / N.$$

Arıza oranı,  $L$ , burada verilen 1000 saatlik periyot, uygulamadan uygulamaya değişse de, genelde her 1000 saate karşılık arıza yüzdesi birimiyle ifade edilir. Örneğin, eğer bir parçanın yüzde 0.003 oranında bir arıza oranı göstereceği söylendiğinde, 1.000.000 parça içinden, 1000 saatlik kullanım sonrasında, 30'unun arıza yapacağı tahmin edilebilir.

Yukarıda verilen ifadede  $dN/N = -Ldt$  bulunur. Bir an için arıza oranının sabit olduğunu varsayarsak, verilen ifadenin integrali ile;

$$\ln N = -Lt + C \quad (\text{buradaki } C, \text{ integral sabitidir})$$

veya  $N = Ke^{-Lt}$  sonucunu elde ederiz.

$K$ , parçaların ilk sayısını gösterir ve normalde  $N_0$  olarak yazılır. Buradan,

$$N = N_0 e^{-Lt}$$

Bu arada, verilen bu eşitlik aynı zamanda radyoaktif maddelerin azalma ifadesine ilişkin eşitlikle aynıdır. Eğer,  $N/N_0$ , yani halen çalışmakta olan parça sayısının başlangıçtaki sayıya oranı yerine,  $r(t)$  terimi koyulursa, ifadeyi

$$r(t) = e^{-Lt} \quad \text{olarak yazabiliriz.}$$

$r(t)$  fonksiyonu, güvenilirlik fonksiyonudur ve bir  $t$  süresi sonunda halen çalışan parçalara ilişkin miktarı verir. Buradan, arıza oranı, bir eksi güvenilirlik arıza oranı  $= 1 - r(t)$  olarak bulunur.

Arıza oranından daha yararlı olacak bir kavram ise MTBF, yani arıza ortalama süre kavramıdır. Çünkü, bir parçanın tahmini kullanım ömrüne ilişkin fikir vermektedir. MTBF,

$$\int_0^{\infty} r(t) dt.$$

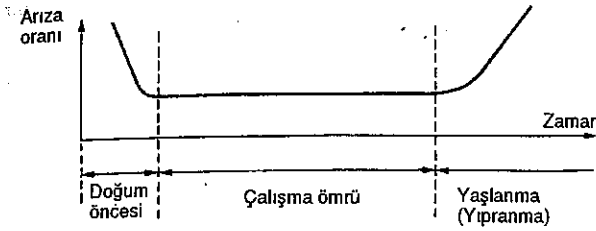
ifadesiyle tanımlanır. Bu ifadenin integrali alınırsa,  $MTBF = m = 1/L$  ve buradan  $r(t) = e^{-t/m}$  bulunur. Şurası belirtilmelidir ki, burada verilen  $m$ , ölçüm sonucunda elde edilmiştir. Yani, bir parça grubunun arıza oranı, bu parçalarının davranış gözlenmek suretiyle ölçülebilir. Elbette, satın almış olduğunuz parçalar, ürünün istatistiklerinde verilen ölçülerde olmayabilir ve bu nedenle de belirlenmeden arızalanabilecektir.

Parçaların yüksek MTBF değerine sahip olmaları istenirse de, tam olarak parçanın ya da sistemin arızalandıklarında hızlı bir biçimde onarılabilmesi için diğer yararlı bir parametre de, MTTR (Onarımına kadar ortalama süre)'dir. Bu parametre, sistemin kullanılabilirliği veya çalışırılık oranını verecek biçimde belirlenebilir. Yani,

$$\text{kullanılabilirlik} = \frac{MTBF}{MTBF + MTTR}.$$

Burada arıza oranı,  $L$ , ve dolayısıyla  $m$ 'nin sabit olduğu varsayılmıştır. Uygun da, bu tümüyle doğru değildir. Şekil 2.75'de, bir parçanın arıza oranının, "kuvvet" biçimli eğrisi zamanın fonksiyonu olarak verilmiştir. Eğrinin sol kısmı parçanın "doğum-öncesi" dönemidir ve parça topluluğunun, üretim hatalarıyla başlangıçta yüksek bir arıza oranına sahip olduğunu ifade eder. Eğrinin ortası, sabit arıza oranına sahip olduğu döneme karşılık gelir ve parça topluluğu faydalı kullanım ömrünü temsil eder. Yığının ömür-süresinin sonuna doğru ise yukarıya doğru çıkması, parçanın yaşlanması ve dolayısıyla yıpranması anlamına gelmektedir.

Günümüzde bazı üreticiler, parçaları piyasaya çıkarmadan önce "pişirdi" görülmektedir. Yani parçalar, yaşlandırmayı hızlandırmak ve doğum öncesi nemlerini atlatmak için genelde yüksek sıcaklık altında çalıştırılırlar. Bu süreçte süresinde arızalanan parçalar üretimin dışına alınırlar ve satıcı, bu yolla



Şekil 2.75 Güvenilirlik eğrisi

yüksek bir parça güvenilirliği elde etmiş olur. Elbette, pişirilen bu parçalar, üretim hatlarında ayıklanan parçalardan daha pahalıya gelmektedir.

Eğer yukarıda tanımlanan kavramlar, tasarımcının, arıza oranı bilinen birçok bileşenden oluşan bir sistemin genelde arıza oranını belirleyebilmesini sağlamak üzere topluca bir bileşen grubuna uygulanamazsa, bilgisayar mühendisleri açısından kullanışlı olmazlar.

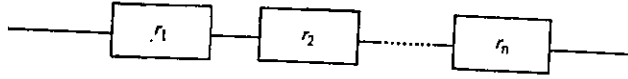
Bir sistemin,  $R_1$  ve  $R_2$  olmak üzere iki bileşenden oluştuğunu varsayalım. Eğer bu iki bileşene ilişkin güvenilirlik değerlerinin sırasıyla  $r_1$  ve  $r_2$  olduğunu bilirsek, buradan sistemin güvenilirliğini hesaplayabiliriz. Bu hesaplamada,  $R_1$  veya  $R_2$  'den herhangi biri arıza yaparsa, sistemin tümüyle de çalışamaz hale geleceğini varsayacağız.

Seri biçimde (yani, biri çalışmazsa tümü çalışmaz) biçiminde çalışan bileşenlerin oluşturduğu bir sistemin güvenilirliği, bileşenlerin teker teker güvenilirlikleri çarpımıdır. Dolayısıyla, iki bileşenli sistemimizin güvenilirliği,  $r = r_1 \times r_2$  'dir. Örneğin,  $R_1$  bileşeni, yüzde 99 oranında güvenilir ve  $R_2$  bileşeni de yüzde 98 oranında güvenilir ise, genel güvenilirlik;  $0.99 \times 0.98 = 0.97$  değerinde olacaktır. Şu noktaya dikkat edilmelidir ki bu formül, genel kanıyı doğrular niteliktedir. Genel güvenilirlik, ifadede yer alan en düşük bileşenin güvenilirliğine bağlıdır, yani sistem, bir zincirin halkalarında olduğu gibi, ancak en zayıf birimi oranında bir iyilik düzeyine sahiptir.

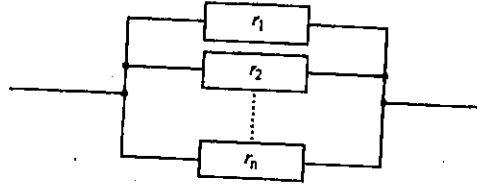
Seri birimlerden ya da bileşenlerden meydana gelen bir sistemin güvenilirliği, Şekil 2.76'da verilmiştir.

Bir sistemi, ancak birden fazla bileşenin arızalanması durumunda arıza yapabilecek biçimde tasarlamak mümkündür. Bu durum, Şekil 2.77'de gösterilmektedir. Bu türde bir sistem, "ekleme" ya da "yedekleme" yapılmasını gerektirir. Örneğin, Şekil

2.77'deki, paralel modüller, bir bilgisayar sistemindeki disk sürücülerdir. Burada tüm sistem, ancak tüm disk sürücülerin arızalanması halinde çalışmaya duruma gelecektir.



Şekil 2.76 Seri düzende bileşenlerin güvenilirlikleri, tek tek güvenilirlilik değerlerinin çarpımıdır. Yani,  $r = r_1 \cdot r_2 \cdot \dots \cdot r_n$ .



Şekil 2.77 Paralel düzende bileşenlerin güvenilirlikleri; 1- güvenilmezlikleridir ve  $1 - (1 - r_1)(1 - r_2) \dots (1 - r_n)$  ifadesiyle verilir.

Bileşenlerin paralel düzende çalışmaları halinde, sistemin arızalanması ancak tüm bileşenlerin arızalanması halinde gerçekleşebilecektir. Eğer,  $i$ . bileşenin güvenilirliliği  $r_i$  ise, sözkonusu bileşenin güvenilmezliği (güvenilir olmama değeri) ya da arızalanma olasılığı,  $(1 - r_i)$  ile ifade edilir. Eğer,  $n$  sayıda bileşen paralelse, bunların tümünün arızalanma olasılığı,  $(1 - r_1)(1 - r_2) \dots (1 - r_n)$  olacaktır. Dolayısıyla sistemin güvenilirliliği;

$$1 - \text{sistem güvenilmezliği} = 1 - (1 - r_1)(1 - r_2) \dots (1 - r_n)$$

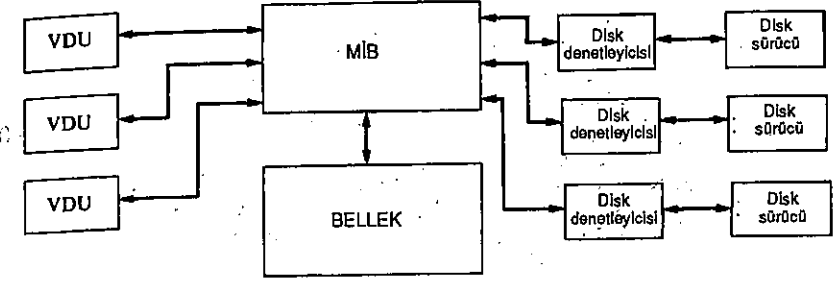
olacaktır.

Şimdi de, paralel çalışan ve güvenilirlikleri sırasıyla yüzde 99 ve yüzde 98 olan  $R_1$  ve  $R_2$  bileşenlerini ele alalım. Sistemin genel güvenilirliliği;

$$\begin{aligned} &= 1 - (1 - 0.99)(1 - 0.98) \\ &= 1 - (0.01)(0.02) = 1 - 0.0002 \\ &= 0.9998, \text{ yani yüzde } 99.98 \text{ olacaktır.} \end{aligned}$$

Paralel bileşenlerin sistemin genel güvenilirliliğini nasıl etkileyebildiğine dikkat edin. Uygulamada, paralel sistem kullanımında, bileşenler veya kollar açısından ödenmesi gereken çok küçük bir bedel vardır, o da bağlantının paralel birimlere vereceği zarardır.

Gerçek sistemlerde, güvenilirlik hesaplanabilmekte ve zincirdeki zayıf halkaların güçlendirmek için gereken yerlerde fazladan eklemeler yapılabilmektedir. Örneğin, Şekil 2.78'de verilen sistemi göz önüne alalım. Bu sistem, bir ALU (Aritmetik Mantık Birimi), bellek modülleri, üç adet VDU ekran, üç disk denetleyici ve üç disk sürücüsü içermektedir. Tablo 2.24'de, bileşenlerin tek tek güvenilirlilikleri verilmiştir.

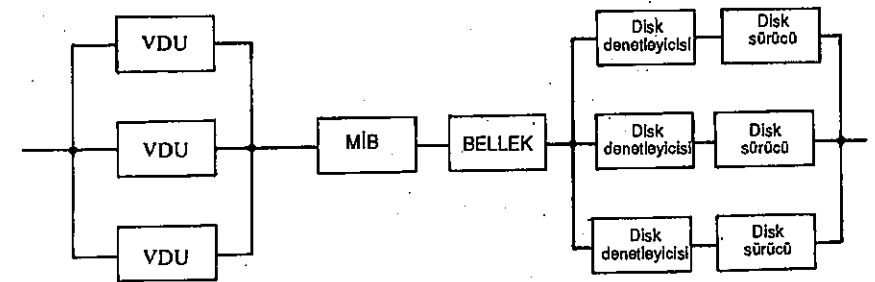


Şekil 2.78 Küçük bir bilgisayar sistemi

Tablo 2.24 Şekil 2.78'de verilen bileşenlerin güvenilirlilikleri

| Bileşen            | Güvenilirlilik |
|--------------------|----------------|
| VDU                | 0.98           |
| MİB                | 0.999          |
| Bellek             | 0.8            |
| Disk denetleyicisi | 0.995          |
| Disk sürücüsü      | 0.8            |

Eğer, sistemin; yalnızca bir MİB, bir bellek modülü ve tek bir disk sürücüsü içerecek biçimde küçültüleceği, ancak hala kabul edilebilir hizmet düzeyine çekileceği



Şekil 2.79'de verilen bilgisayarın güvenilirliliği

Şekil 2.79'da gösterilmiştir. Genel güvenilirlik seri olarak bağlı şu dört grupta, yani VDU ekranlar, MİB, bellek, disk denetleyicisi tarafından belirlenecektir. VDU ekranların, paralel olduklarını ve ancak her üç ekranın da arızalanması halinde çalışmayacağına dikkat edin. Benzer biçimde, her bir disk denetleyici birimin ve birliktede bulunduğu disk sürücüler de, sistem arızaya geçmeden önce arızalanmaları gerekir.

Bu dört birimden her birinin güvenilirlikleri;

VDU ekranları için  $1 - (1-0.98)(1-0.98)(1-0.98) = 1 - 0.000008 = 0.999992$

MİB için 0.999

### Bellek 0.8

$$\begin{aligned} \text{Disk sürücü ve denetleyicileri} &= 1 - (1 - 0.995 \times 0.8)(1 - 0.995 \times 0.8) \\ &= 1 - (1 - 0.796)(1 - 0.796)(1 - 0.796) \\ &= 1 - 0.008495 = 0.9915 \end{aligned}$$

Sistemin genel güvenilirliği, seri olarak çalışan bu dört kolun tek tek güvenilirliklerinin çarpımı ile verilir;

$$0.999992 \times 0.999 \times 0.8 \times .9915 = 0.7924 \approx \% 79$$

Yukarıdaki sonuçlar üzerinde, hemen hemen tümüyle bellek modülünün düşük güvenilirliğinin baskın olduğu sonucu görülmelidir. Üreticinin böyle iki modülü paralel olarak yerleştirdiğini varsayalım. Belleğin güvenilirliği bu durumda,  $1 - (1 - 0.8)(1 - 0.8)$  ya da 0.96 değerine, ve sistemin güvenilirliği de  $0.9509 \approx \%95$  'e yükselecektir.

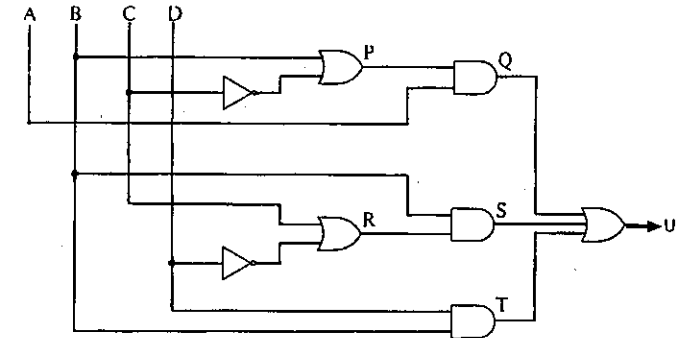
Yukarıda sözünü ettiklerimizden, güvenilirliğin bir bilim dalı olduğu sanılabilir. Aslında bir bilim değil, göz boyamadır. Bilgisayar tasarımcısı ya da herhangi diğer bir mühendis, bu sonuçları düşünerek güvencede olduğunu çok fazla düşünmemelidir. Güvenilirlik hesaplamaları iki varsayıma dayanır. İlki, bir bileşenin güvenilirliği, kullanım ömrünün büyük çoğunluğunda sabit olduğudur (Şekil 2.75). Bu varsayım, çalışmada yeni arıza olasılıklarını ihmal eder. Örneğin, dünyanın ilk jet yolcu uçağı, Comet, zamanında bilinen güvenilirlik teorileri dikkate alınarak tasarlanmıştır. Ancak ne yazık ki, metal yorgunluğu teorisi iyi anlaşılamadığından, çatlaklar büyümüş ve uçağın kare biçimli pencerelerinden yayılmaya başlamıştı. Bu da, uçağın gövdesinin ayrılmasına ve sonuçta da uçağın kaybedilmesine yol açmıştı.

İkinci varsayım, Şekil 2.75’de verilen sistem (paralel ve seri devre) için teknikleri kullanarak güvenilirliğin hesaplanabileceği şeklindeydi. Bu, bir uçak veya bir nükleer reaktör gibi büyük sistemler için değil de ancak basit sistemler için nisbeten doğrudur. Bu türde büyük sistemler incelendiğinde, inceleme, bir mühendis tarafından, belirli varsayımlar altında çalışarak yapılır. Bu varsayımların bir bölümü de insan davranışlarına ilişkindir. Örneğin, bir sistemin güvenilirliği önemli ölçüde bakımına bağlıdır. Eğer, bakımlar ekonomik nedenlerle üreticinin belirlediği standartlara göre uygulanmıyorsa, sistemin güvenilirliği, kağıt üzerindeki güvenilirlikle aynı olmayacaktır.

## Problem 1

## Mantık elemanları

1. Şekil 2.80'deki A, B, C ve D değişkenlerinin mümkün olan tüm değerlerine karşılık P, Q, R, S, T ve U değişkenlerinin alacağı değerleri gösteren bir tablo hazırlayın.



Şekil 2.80

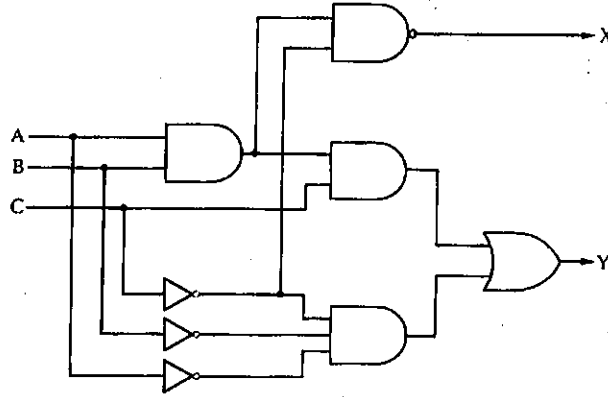
*Not:* Bu soruya ilişkin doğruluk tablosu, Tablo 2.25’de verilen biçimde olmalıdır.

Tablo 2.25 Şekil 2.80 için doğruluk tablosu

[illegible]

2. 1. sorudaki diyagram için, U çıkışına ilişkin Boole ifadesini A, B, C girişlerinin fonksiyonu olarak bulun. Bulduğunuz bu ifadeyi sadeleştirmeyin.

3. Doğruluk tablosu kullanarak, Şekil 2.81'deki devre için, X ve Y arasındaki ilişkiyi ve A, B, C giriş değişkenlerini bulun. Doğruluk tablosundan, X ve Y için Boole ifadelerini yazın. Şekil 2.81'deki devre için; Kapılara ilişkin Boole eşitliklerini göz önünde tutarak X ve Y için eşitlikler türetin.



Şekil 2.81

Boole eşitliklerindeki A, B, C yerine (000, 001 vb. gibi) karşılıklarını koyarak iki sonucun (yani, doğruluk tablosundan türetilen sonuçlarla Boole eşitliklerinden türetilen sonuçların) birbirine eşit olduğunu gösterin.

4. Aşağıda verilen Boole eşitliklerine ilişkin mantık diagramlarını, yalnızca VE, VEYA ve DEĞİL kapılarını kullanarak çizin:

- (a)  $F = A\bar{B} + \bar{A}B$
- (b)  $F = (A + B + C)(AB + AC)$
- (c)  $F = (\bar{A} + \bar{C})(\bar{A} + B\bar{D})$
- (d)  $F = A + \bar{C}.A + BD$
- (e)  $F = (A\bar{B} + \bar{A}B + A\bar{C})(\bar{A}\bar{B} + \bar{A}B + A\bar{C})$

Eşitlikleri sadeleştirmeyin.

### Boole cebri

1. Aşağıda verilen eşitlikleri Boole cebri aracılığıyla sadeleştirin. Ancak, Karnaugh diyagramını kullanmayın.

- (a)  $A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C$
- (b)  $ABC + \bar{A}BC + A\bar{B}C + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}\bar{B}C$
- (c)  $ABC + \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}B\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C}$
- (d)  $(A + B + C)(A + \bar{B} + \bar{C})(A + B + \bar{C})(A + \bar{B} + C)$
- (e)  $(\bar{A}\bar{B} + \bar{A}B + \bar{A}C)(\bar{A}\bar{B} + \bar{A}B + \bar{A}C)$
- (f)  $A + \bar{B} + (A + CD)(A + \bar{B}\bar{C})$
- (g)  $\bar{A}\bar{B} + \bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C}D$
- (h)  $(\bar{A} + B)(A + \bar{B} + \bar{C})(A + B + \bar{C})(\bar{A} + B + \bar{C})(\bar{A} + \bar{B} + C)$

2. Aşağıdaki eşitliklerin tümleyenlerini almak için de Morgan teoremini kullanın. Tümleme işleminden önce veya sonra eşitlikleri sadeleştirmeyin.

- (a)  $\bar{X}\bar{Y} + XY$
- (b)  $XYZ + \bar{X}\bar{Y}$
- (c)  $XYZ + \bar{X}\bar{Y}$
- (d)  $WX(\bar{Y}Z + YZ)$
- (e)  $WX(W\bar{Z} + \bar{Y}Z)$
- (f)  $XY + \bar{X}\bar{Y}(WZ + \bar{W}Z)$

3. Aşağıdaki eşitlikleri, çarpımların toplamı biçimine dönüştürün.

- (a)  $(A + B)(\bar{B} + AC)(\bar{A} + C)$
- (b)  $(C + D)(\bar{A}\bar{B} + AC)(\bar{A}\bar{C} + B)$
- (c)  $(A + B + C)(A + CD)(D + F)$

4. Aşağıdaki eşitlikleri, toplamaların çarpımı biçimine dönüştürün.

- (a)  $AB + \bar{A}\bar{B} + BC$
- (b)  $AB + \bar{A}C + \bar{B}C$
- (c)  $\bar{A}\bar{B}(\bar{C}\bar{D} + E(B + \bar{C} + D))$
- (d)  $A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C$

### Boole cebrinin uygulaması

1. Bir devre, 0000=0'dan 1111=15'e kadar ikili doğal sayıları gösteren P, Q, R ve S olmak üzere dört girişe sahiptir. En büyük değerlikli bit P'dir. Devrenin, tek bir çıkışı, X vardır. Çıkış, ancak girişlerin üçle bölünebilir sayılar olduğunda doğru (1) olmaktadır. (Sıfır değeri, üçe bölünebilir bir giriş olarak kabul edilecektir.)

Bu devrenin doğruluk tablosunu düzenleyin ve buradan X çıkışı P, Q, R ve S ifadeleri olarak bulun. Bu devreyi gerçekleştirebilmek için VE, VEYA ve DEĞİŞTİRİCİ kapılarından oluşan düzenlemenin devre şemasını çıkarın.

Bu fonksiyonu yalnızca VEDEĞİL kapılarıyla gerçekleştirebilmek için ikinci devre tasarlayın.

2. Bir mantık devresinin, A ve B olmak üzere 2-bitlik ikili doğal girişi bulunmaktadır. A, A<sub>1</sub> ve A<sub>0</sub> olarak verilmiştir, burada A<sub>1</sub>, en büyük değerlikli biti göstermektedir. B girişi de benzer biçimdedir, yani B<sub>1</sub>, en büyük değerlikli biti göstermektedir. B<sub>1</sub> ve B<sub>0</sub>'dan oluşmuştur. Devrenin X, Y ve Z olmak üzere üç çıkışı bulunmaktadır. A ve B ile X, Y, Z arasındaki ilişki aşağıda verildiği gibidir.

|       | X | Y | Z |
|-------|---|---|---|
| A > B | 1 | 0 | 0 |
| A < B | 0 | 1 | 0 |
| A = B | 0 | 0 | 1 |

Bu fonksiyonu gerçekleyecek bir devre tasarlayın.

3. Bir mantık devresi, A, B ve C olmak üzere üç girişe sahiptir. Burada A, en düşük değerlikli biti göstermektedir. Devrenin Y<sub>0</sub>'dan Y<sub>7</sub>'ye kadar olmak üzere sekiz adet çıkışı bulunmaktadır. Giriş uçlarına (A, B ve C) herhangi bir ikili giriş uygulandığında, çıkışlardan yalnız biri doğru (1) çıkış vermekte, diğerleri yanlış (0) olarak çıkmaktadır. Buradan, eğer C=1, B=0 ve A=0 ise, Y<sub>4</sub>, 1 olmaktadır. Bu devreyi gerçekleyecek bir mantık devresi tasarlayın. Böyle bir devre tekilleyici olarak adlandırılır ve Bölüm 2'de tekilleyici devresini görmüştünüz.

4. A, B, C ve D olmak üzere dört girişi bulunan devrenin girişlerine 4-bitlik ikili bir sayı uygulanmıştır. Devrenin F olmak üzere tek bir çıkışı vardır. Giriş değerleri (verilen bu iki değer de dahil olmak üzere) üç-oniki sayı aralığında olması halinde çıkış da doğru (1) olmaktadır.

Bu probleme ilişkin bir doğruluk tablosu çizin ve F çıkışı için, girişler cinsinden sadeleştirilmiş eşitliği bulun.

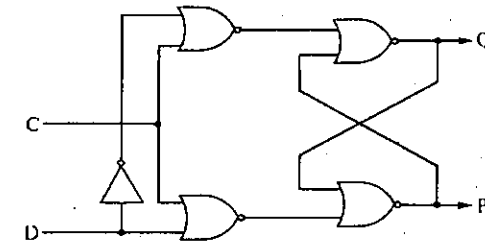
Devreyi, (a) yalnızca VEDEĞİL kapıları kullanarak ve (b) yalnızca VEYADEĞİL kapıları kullanarak gerçekleştirin.

### Ardışık devreler

1. Şekil 2.82'de verilen devrenin başlangıç durumunun C=1, D=1, P=1 ve Q=0 olduğunu varsayarak Tablo 2.26'yı tamamlayın. Bu problem, C ve D'nin yeni değerlerinin çapraz-bağlı VEDEĞİL kapı çiftinin girişleri ve buradan da P ve Q çıkışları üzerindeki etkileri hesaplanarak çözülmeye çalışılmalıdır. P ve Q çıkışları, aynı zamanda VEDEĞİL kapılarının girişleri olduklarından, C ve D'nin bir sonraki girişlerinin etkileri hesaplanırken, P ve Q çıkışlarının değişmesi de dikkate alınmalıdır. Bir VEDEĞİL kapısının çıkışlarının, eğer girişlerin her ikisi de 0 olduğunda 1 ve aksi halde de 0 olduğunu hatırlayalım.

Tablo 2.26

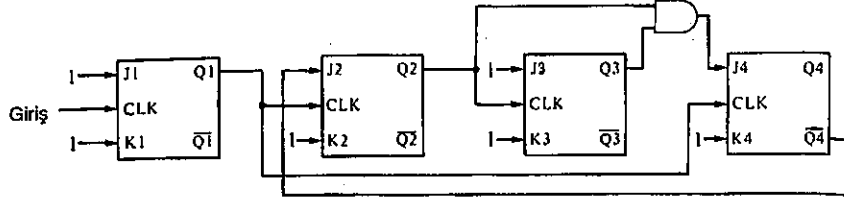
| C | D | P | Q |
|---|---|---|---|
| 1 | 1 | 1 | 0 |
| 1 | 0 |   |   |
| 0 | 0 |   |   |
| 1 | 1 |   |   |
| 0 | 1 |   |   |
| 1 | 1 |   |   |
| 0 | 1 |   |   |
| 0 | 0 |   |   |
| 1 | 0 |   |   |



Şekil 2.82

Devreyi, eklenecek yeni bir S girişi ile, S girişi 1 olduğunda, P'yi 1 ve Q'yu da 0 yapacak biçimde değiştirin. Devreye ikinci bir R girişi ekleyin. Bu giriş de, benzer biçimde P'yi 0 ve Q'yu da 1 yapacak biçimde olsun. Burada, R ve S girişlerinin her ikisinin de aynı anda 1 olamayacağını hatırlınızdan çıkarmayın. Bu nedenle, R=S=1 koşulunun dikkate alınması gerekmeyecektir.

2. Şekil 2.83'de verilen devrenin çalışmasını inceleyin. Her bir JK flip-flop'u, kendi saat girişlerinin düşen kenarında tetiklenmektedir. Q1 ve Q4 çıkışlarının sırasını zamanın fonksiyonu olarak ifade eden bir diyagram çizin.



Şekil 2.83

3. Eşzamanlı (senkron) sayıcı ele alınmamış olmalarına rağmen, yapıları fazla karmaşık değildir. Saat darbeleri, sayıcıdaki her bir flip-flop'a aynı anda uygulanır. Her bir saat darbesinden sonra, yeni çıkışlar, flip-floplara saat darbesi uygulandığı andaki, (JK flip-floplarının kullanıldığını varsayarsak) JK girişlerinin durumları tarafından belirlenirler.  $n$ -bitlik bir sayıcı için,  $i$ . çıkış,  $Q_i \leftarrow J_i, K_i$  tarafından verilir, burada  $J_i \leftarrow f_a(Q_0, Q_1 \dots Q_{n-1})$  ve  $K_i \leftarrow f_b(Q_0, Q_1 \dots Q_{n-1})$ 'dir. Yani, bir sonraki çıkışları üretmek için gereken  $J$  ve  $K$  değerlerini oluşturmak üzere bir önceki çıkışları,  $Q_0$ 'dan  $Q_{n-1}$  kadar olan çıkışları, kullanmaktayız. Bu alıştırma, sayıcının her bir katı için uygulanmaktadır.

Aşağıda verilen sıralamayı üretecek biçimde negatif-kenar tetiklemeli JK tipi flip-flop kullanan 3-bitlik bir sayıcı tasarlayın.

| $Q_2$ | $Q_1$ | $Q_0$ |              |
|-------|-------|-------|--------------|
| 0     | 0     | 0     | Temel sıra   |
| 0     | 1     | 0     |              |
| 0     | 1     | 1     |              |
| 0     | 0     | 1     |              |
| 1     | 0     | 0     |              |
| 1     | 1     | 0     | Sıra tekrarı |
| 0     | 0     | 0     |              |
| 0     | 1     | 0     |              |

## BİLGİSAYAR ARİTMETİĞİ

Endüstriyel yoldan üretilen ikili mantık elemanlarıyla kolayca gerçekleştirilebilirlikleri ve fiyatlarının çok düşük düzeyde olmaları nedeniyle, bilgisayar içindeki sayısal verileri göstermek için kaçınılmaz bir şekilde ikili sayı sistemleri seçilmiştir. Daha ileriye geçmeden önce şunu vurgulamak gerekir ki, ikili aritmetikteki kurallar, ondalık aritmetik kurallarıyla aynıdır, değişen yalnızca sayı tabanıdır.

### 3.1 Karakterler, sözcükler ve baytlar

Bilgisayar içinde saklanabilecek ve üzerinde işlem yapılabilecek en küçük bilgi birimi bittir. Sayısal bilgisayarlar bilgileri, belleklerinde, sözcük olarak adlandırılan bit grupları biçiminde saklarlar. Bir sözcükte yer alan bit sayısı, bilgisayardan bilgisayara değişiklik gösterir. ICL 1900 serisi, 24-bit, UNIVAC 1100 sistemi 36-bit, PDP-11 sistemi ise 16 bit uzunlukta sözcükler kullanır, birçok mikroişlemci ise 8-bitlik sözcüklerle çalışır. Sekiz bitten oluşan grup ise, bayt olarak bilinelemiştir. Genelde, sözcükler, 2 veya 4 bayt uzunluk olarak ifade edilirler, çünkü içerdiği bitler, 2 veya 4 tane sekiz bitlik gruplar halinde düzenlenebilirler. Mikroişlemciler ve mikrobilgisayarlar normalde, sözcük-uzunlukları ve adresleri 8, 16 veya 32 bit uzunlukta olduğundan bayt-yönelimlidir. Fakat bazı bilgisayarların da altı bitten oluşan gruplar kullandıkları ve bunları bayt olarak adlandırdıklarını da unutmamalıyız.

$n$ -bit uzunlukta bir sözcük,  $2^n$  kadar değişik bit deseni içerebilir. Sözcükler birçok şeyi gösterebilirler, 1'ler ve 0'lardan oluşan desenin özel bir anlamı yoktur. Belirli bir bit deseninin anlamı, programcı tarafından ona verilen anlamdan ibarettir. Humpty Dumpty'nin Alice'e söylediği gibi; "Sözcükler, tam olarak neyi ifade etmesini istersem o anlama gelirler, ne fazlasına ne de eksikliğine..." . Bilgisayarın kendisi, sözcüğün ne anlama geldiğini söyleyemez, fakat basitçe onun üzerinde, programcı belirlediği yoldan işlem yapar. Örneğin, bir programcı bir kişinin adını bilgisayara okutabilir ve daha sonra da kişinin adını gösteren bit deseni üzerinde, sözcüğü iki ile çarpma gibi, bir işlem yapar. Çıkan sonuç anlamsız olsa da bilgisayar, verilen işlemi mutlu bir biçimde uygular. Bir Çin lokantasında, 46, "bambu çubukları"nı ve 27 de "kızartılmış yumurtalı pilav"ı gösterebilir. Eğer, bir porsiyon 73 istersem, büyük bir olasılıkla karşıma yanında bambu çubuklarıyla birlikte kızartılmış yumurtalı pilav gelecektir.

Aşağıda, bir sözcükle gösterilebilen bazı nesneler verilmiştir:

1. **Komut** MİB tarafından işleme koyulacak bir komut, normalde tek bir sözcük gösterilir. Komutun bit-deseni ile ne yapacağı arasındaki ilişki seçime bağlıdır. Bilgisayarı tasarlayan tarafından belirlenir.

2. **Sayısal nicelikler** Sözcükler, tek başlarına ya da bir sözcük dizisinin bir parçası olarak, sayısal nicelikleri gösterebilirler. Sayılar; BCD tamsayı, işaretsiz ikili tamsayı, işaretli ikili tamsayı, BCD kayan noktalı, ikili kayan noktalı, karmaşık kayan noktalı, çift duyarlılık tamsayı vb. gibi çok çeşitli formlardan biriyle gösterimlenebilir. Burada verilen terimlerin bazılarının anlamları bilgisayarın, bu terimlerle gösterilen sayı sistemlerindeki işlemleri uygulama yetemi, ileride ele alınacaktır.

3. **Karakterler** Metinlerin girilip işlendiği ve sonuçların basıldığı çok çeşitli bilgisayar uygulamaları bulunmaktadır. Buna en açık ve en bilinen örnek kelime işlemcilerdir. Programların kendileri de bilgisayara ilk girildiklerinde metin biçimindedirler. Alfabetik karakterler (A'dan Z'ye, a'dan z'ye, 0'dan 9'a) ve semboller \*, -, +, !, ? vb. ; bilgisayarda saklanabilecek ve işlenebilecek biçimde bit desenlerine atanırlar. Günümüzde, bilgisayar endüstrisinde yaygın kullanım kazanan özel bir kodlama sistemi vardır. Bu kod, ASCII (Bilgi Değişimi için Amerikan Standart Kodu) olarak bilinir ve bu kodlamayla karakterler, 7 bitle, maksimum  $2^7=128$  farklı karaktere olanak tanıyacak biçimde, gösterilir. Bu 128 karakterin 96 tanesi, (büyük ve küçük harfler de dahil olmak üzere) normal basım karakterleridir. Kalan 32 karakter ise, "basılmayan" karakterlerdir. Bunlar, satır başı, girilme, satır atlama vb. karakterleridir. Sözcükleri oluşturan bitlerle, bunların ASCII karşılıkları Şekil 3.1'de tanımlanmıştır.

Bir karaktere ilişkin 7-bitlik ikili kodun ne olduğunu bulabilmek için, bu karakterin içinde bulunduğu sütundaki en büyük değerlikli üç bite ve yine bu karakterin içinde bulunduğu satırdaki en küçük değerlikli dört bite bakın. Dolayısıyla, "W" karakteri, 101 sütununda ve 0111 satırında bulunduğundan, bu karaktere ilişkin ASCII kodu 1010111 (veya onaltılı biçimde 57) olduğu görülür. Benzer biçimde, CR (satır başı) karakterinin ASCII kodunun da, 0001101 olduğu görülebilir.

### 3.2 Sayı tabanları

Ondalık sistemde sayıları, bulundukları basamak yoluyla gösteririz. Bir basamak-taki değer bir konum sağa taşınmışsa, onla (yani sayı tabanıyla) çarpılmış, bir

Şekil 3.1 ASCII kodu

|             | $b_6b_5b_4$ | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   |
|-------------|-------------|-----|-----|-----|-----|-----|-----|-----|-----|
| $b_7b_6b_5$ |             | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| 0           | 0000        | NUL | DLC | SP  | 0   | @   | P   | .   | p   |
| 1           | 0001        | SOH | DC1 | !   | 1   | A   | Q   | a   | q   |
| 2           | 0010        | STX | DC2 | "   | 2   | B   | R   | b   | r   |
| 3           | 0011        | ETX | DC3 | #   | 3   | C   | S   | c   | s   |
| 4           | 0100        | EOT | DC4 | \$  | 4   | D   | T   | d   | t   |
| 5           | 0101        | ENQ | NAK | %   | 5   | E   | U   | e   | u   |
| 6           | 0110        | ACK | SYN | &   | 6   | F   | V   | f   | v   |
| 7           | 0111        | BEL | ETB | '   | 7   | G   | W   | g   | w   |
| 8           | 1000        | BS  | CAN | (   | 8   | H   | X   | h   | x   |
| 9           | 1001        | HT  | EM  | )   | 9   | I   | Y   | i   | y   |
| A           | 1010        | LT  | SUB | *   | :   | J   | Z   | j   | z   |
| B           | 1011        | VT  | ESC | +   | ;   | K   | [   | k   | {   |
| C           | 1100        | FF  | FS  | ,   | <   | L   | /   | l   |     |
| D           | 1101        | CR  | GS  | -   | =   | M   | ]   | m   | }   |
| E           | 1110        | SO  | RS  | .   | >   | N   | ^   | n   | ~   |
| F           | 1111        | SI  | VS  | /   | ?   | O   | _   | o   | DEL |

Denetim karakterleri

Basılan karakterler

konum sola taşınmışsa da, onla bölünmüş olmaktadır. Dolayısıyla, 95 sayısındaki 9 basamağı, 59 sayısındaki 9'un on katı değerindedir. Eğer bu ifade yeterince açıksa Romen rakamlarını ele alalım. Dünyanın büyük bölümünü fethetmiş, Latin gramerini bulmuş, aslanlar beslemiş, şenlikler düzenlemiş olsalar da, matematik sistemleri çok kullanışsızdı. Basamak sistemi kullanmamış olduklarından her yeni büyük sayının kendine özel bir sembolü vardı. Sayı sistemleri bir 'takas' ilkesine dayalıydı ve buradan, eğer  $X=10$  ve  $I=1$  ise  $XI=11$  ( $10+1$ ) ve  $IX=9$  ( $10-1$ ) değerini göstermekteydi.

$b$  tabanında ifade edilen  $N$  sayısı;

$$a_n a_{n-1} a_{n-2} \dots a_1 a_0 . a_{-1} a_{-2} \dots a_{-m}$$

şeklinde yazılır ve

$$N = a_n \times b^n + a_{n-1} \times b^{n-1} + \dots + a_1 \times b^1 + a_0 \times b^0 \\ + a_{-1} \times b^{-1} + a_{-2} \times b^{-2} + \dots + a_{-m} \times b^{-m}$$

$$= \sum_{i=-m}^n a_i b^i$$

şeklinde tanımlanır.

Örneğin 1984 ondalık sayısı,  $1 \times 10^3 + 9 \times 10^2 + 8 \times 10 + 4 \times 1$ 'e eşittir. Benzer şekilde 10110 ikili sayısı,  $1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$  ile veya  $16 + 2 = 22$  ile verilir. Basamak notasyonunda, sayının tam ve kesir kısmını ayırmak için taban noktası kullanılır. Yukarıdaki denklemlerde yer alan  $a$ 'lara basamak (diğer) denir ve  $b$  olası değerlerinden birine sahip olabilir. Günümüzde bilgisayarlar ondalık, ikili, sekizli ve onaltılı sayı tabanları olmak üzere dört taban ile ilgilenir.

|          |        |                                         |
|----------|--------|-----------------------------------------|
| Ondalık  | $b=10$ | $a=\{0,1,2,3,4,5,6,7,8,9\}$             |
| İkili    | $b=2$  | $a=\{0,1\}$                             |
| Sekizli  | $b=8$  | $a=\{0,1,2,3,4,5,6,7\}$                 |
| Onaltılı | $b=16$ | $a=\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$ |

İnsanlar ondalık, bilgisayarlar ise ikili sayılarla işlem yaparlar. İleride, sekizli ve onaltılı sistemler kullanmanın, insan belleğine yardımcı olma amacını taşıdığını göreceğiz. Uzun ikili basamak dizilerini anımsamak hemen hemen imkansızdır. Bunları (çok kolayca) sekizli veya onaltılı tabanlara dönüştürülürse, daha kısa sekizli veya onaltılı sayılar bellek tarafından kolayca kabul edilecektir. Bundan başka, sekizli ve onaltılı sayılar ikili sayılardan daha toplu olduklarından (1 sekizli basamak= 3 tane ikili basamak ve 1 onaltılı basamak= 4 tane ikili basamak uzunluğundadır) bilgisayar metinlerinde ve 'bellek-indirme' (core-dump) işlemlerinde kullanılır. Bu ikinci terim bilgisayar belleğinin bir kısmının basılı-çıkışını alma anlamına gelir ve normalde öteki tüm alternatifler başarısız olduğunda teşhise yönelik bir yardımcı işleme yardımcı olmak üzere yapılır. Örneğin 10001001 ikili sayısı onaltılı 89 sayısının eşdeğeridir. Açıkça görüldüğü gibi 89'u anımsamak 10001001'i anımsamaktan çok daha kolaydır.

İkili sayıların, diğer gösterilme biçimlerine göre insanlara avantaj sağladığı durumlar vardır. Sözelimi bilgisayar-denetimli bir kimya fabrikasında üç ısıtıcı, üç valf ve iki pompa vardır. Bunlar sırasıyla H1, H2, H3, V1, V2, V3 ve P1, P2 terimleriyle gösterilsin. bilgisayardan çıkan sekiz bit bir sözcük bir arabirimine uygulansın. Bu birim, ikili birleri ve sıfırları elektrik sinyaline dönüştürür ve bu sinyaller de ilgili devreyi açar (mantıksal bir) veya kapatır (mantıksal sıfır). Örneğin denetim birimine, 01010011 ikili sözcüğü uygulandığında Tablo 3.1'de görülen sonuçlar elde edilir.

Denetim sözcüğünün ikili değerinin incelenmesiyle tüm cihazların durumu kolayca görülebilir. Eğer çıktı ondalık (83), onaltılı (53) veya sekizli (123) biçimde gösterilseydi sayı ile amaçlanan işlevi arasındaki ilişki bu kadar açık olmayacaktı.

Tablo 3.1 01010011'in kodunun çözülmesi

| Bit | Cihaz | Durum  |
|-----|-------|--------|
| 0   | H1    | kapalı |
| 1   | N2    | açık   |
| 0   | H3    | kapalı |
| 1   | V1    | açık   |
| 0   | V2    | kapalı |
| 0   | V3    | kapalı |
| 1   | P1    | açık   |
| 1   | P2    | açık   |

### 3.3 Sayı tabanlarının birbirine dönüştürülmesi

Kimi zaman, sayıların bir tabandan diğerine dönüştürülmesinde "kalem-kağıt" tekniğinin kullanılması gerekir. Bu, özellikle mikroişlemcilerle assembly-dili veya makine-kodu düzeyinde çalışırken doğrudur. Genelde bilgisayar kullanıcısının, tabanlararası dönüşümle ilgilenmesi zorunlu değildir. Çünkü, bilgisayarın, ondalık girişleri kendi iç ikili gösterilmesine dönüştürecek bir yazılımı mutlaka vardır. Bilgisayar işini yaptıktan sonra, basımını yapmadan önce ikili sonuçları ondalık biçime dönüştürür. Sayı tabanlarının etkisine ilişkin bilgi kimi zaman oldukça HAYATİDİR, çünkü en basit ondalık kesirlerin bile (sözelimi  $1/10 = 0.1$ ) tam ikili eşdeğerleri yoktur. Bilgisayardan; 0.1'lik artırımlarla toplama yapması ve sonuç bir olduğu zaman durması istensin. Bilgisayar hiçbir zaman durmayabilir. Çünkü 0.1 tam olarak bir ikili sayı ile gösterilemez ve bu da 0.1'in ikili gösterilmesi toplamının hiçbir zaman tam 1 olmayacağı sonucunu verir. Sonuç 1.0000000000001 veya 0.999999999999 olabilir, ki bu hemen hemen birdir; fakat 1'e eşit değildir ve 1 ile eşitlik için yapılacak bir test daima başarısızlıkla sonuçlanacaktır.

#### 3.3.1 Ondalıktan ikiliye

Ondalık bir sayıyı ikiliye çevirmek için sayılar ardarda 2'ye bölünür ve her bölümden kalan yazılır. İşlem, ANCAK bölmenin sonucu 0 ve kalan 1 olduğunda sonlandırılır. Aşağıda verilen tüm dönüştürmelerde R'nin bölmeden sonraki kalan olduğuna dikkat ediniz.

Örneğin,  $123_{10}$ 'u ikili biçime dönüştürelim.

$$\begin{array}{rcl}
123 \div 2 & \rightarrow & 61, \quad R=1 \\
61 \div 2 & \rightarrow & 30, \quad R=1 \\
30 \div 2 & \rightarrow & 15, \quad R=0 \\
15 \div 2 & \rightarrow & 7, \quad R=1 \\
7 \div 2 & \rightarrow & 3, \quad R=1 \\
3 \div 2 & \rightarrow & 1, \quad R=1 \\
1 \div 2 & \rightarrow & 0, \quad R=1.
\end{array}$$

Sonuç en büyük değerlikli bittten (sonuncu kalandan)) yukarı doğru okunarak  $123_{10} = 1111011_2$  sonucu bulunur.

### 3.3.2 Ondalıktan sekizliye

İşlem yukarıdaki gibidir, tek fark bölmenin sekizle yapılmasıdır.

Örneğin  $4629_{10}$

$$\begin{array}{rcl}
4629 \div 8 & \rightarrow & 578, \quad R=5 \\
578 \div 8 & \rightarrow & 72, \quad R=2 \\
72 \div 8 & \rightarrow & 9, \quad R=0 \\
9 \div 8 & \rightarrow & 1, \quad R=1 \\
1 \div 8 & \rightarrow & 0, \quad R=1.
\end{array}$$

Böylece,  $4629_{10} = 11025_8$  sonucu elde edilir.

### 3.3.3 Ondalıktan Onaltılıya

Dönüştürme ikili ve sekizli'de olduğu gibidir, fakat kalanın 0-15 ondalık aralığında, yani 0-F onaltılı aralığında olduğuna dikkat edin.

Örneğin  $53241_{10}$

$$\begin{array}{rcl}
53241 \div 16 & \rightarrow & 3327, \quad R=9 \\
3327 \div 16 & \rightarrow & 207, \quad R=15_{10} = F \\
207 \div 16 & \rightarrow & 12, \quad R=15_{10} = F \\
12 \div 16 & \rightarrow & 0, \quad R=12_{10} = C.
\end{array}$$

Böylece  $53241_{10} = CFF9_{16}$  sonucu elde edilir.

### 3.3.4 İkiliden Ondalığa

İkinin gereken kuvvetlerini toplayarak ikili bir sayıyı ondalığa çevirmek mümkündür. Bu teknik yedi, sekiz bite kadar olan nispeten küçük ikili sayılar için uygundur.

Örneğin  $1010111_2$

$$\begin{array}{r}
64 \ 32 \ 16 \ 8 \ 4 \ 2 \ 1 \\
1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 1 = 64 + 16 + 4 + 2 + 1 = 87
\end{array}$$

ile gösterilir.

Daha metodik bir teknik aşağıdaki gibi bir kendi kendini tekrarlayan algoritmaya dayanır. En soldaki sıfır olmayan bit alınır, ikiyle çarpılır, sağdaki bite eklenir. En düşük değerlikli bit ekleninceye kadar bu şekilde devam edilir. Bu yöntem matematik olarak şöyle ifade edilebilir:

$$(a_0 + 2(a_1 + 2(a_2 + \dots)))$$

Burada  $a_0$  en küçük değerlikli bittir.

Örneğin  $1010111_2$ 'i ele alalım

$$\begin{array}{ccccccc}
1 & 0 & 1 & 0 & 1 & 1 & 1 \\
\downarrow & & \downarrow & & \downarrow & & \downarrow \\
2 & \rightarrow & 4 & \rightarrow & 10 & \rightarrow & 20 \\
& & \downarrow & & \downarrow & & \downarrow \\
& & 5 & \rightarrow & 10 & \rightarrow & 21 \\
& & & & \downarrow & & \downarrow \\
& & & & 10 & \rightarrow & 42 \\
& & & & & & \downarrow \\
& & & & & & 43 \rightarrow 86 \\
& & & & & & & \downarrow \\
& & & & & & & 87
\end{array}$$

Böylece  $1010111_2 = 87_{10}$  sonucu bulunur.

### 3.3.5 Sekizliden Ondalığa

Yukarıda yöntem iki yerine sekiz koyarak kullanılabilir ve şu formül ile verilir:  $(a_0 + 8(a_1 + 8(a_2 + \dots)))$

Örneğin  $6437_8$ 'i ele alalım

$$\begin{array}{cccc}
6 & 4 & 3 & 7 \\
\downarrow & & \downarrow & & \downarrow \\
48 & \rightarrow & 416 & \rightarrow & 3352 \\
& & \downarrow & & \downarrow \\
& & 52 & \rightarrow & 419 \\
& & & & \downarrow \\
& & & & 3359
\end{array}$$

Böylece,  $6437_8 = 3359_{10}$  sonucu elde edilir.

### 3.3.6 Onaltılıdan Ondalığa

Buradaki yöntem, ikili ve sekizlideki yöntemle aynıdır, sadece çarpan olarak 16 kullanılır.

Örneğin  $1AC_{16}$ 'yı ele alalım

$$\begin{array}{r} 1 \quad A \quad C \\ \downarrow \quad \downarrow \quad \downarrow \\ 16 \quad 16 \quad 16 \\ \hline 256 \quad 416 \quad 428 \end{array}$$

Böylece,  $1AC_{16} = 428_{10}$  sonucu bulunur.

### 3.3.7 İkili, sekizli ve onaltılı sayılar arasında dönüşümler

Bu kitabın birçok yerinde ikili sayılar, onaltılı biçiminde gösterilecektir. Bazı metinler sekizli formata yardımcı olsalar da, sekizli sayıların 8 ve 16-bitlik ikilideğerleri ile gösterilmesini pek uygun bulmuyorum. İkili ile onaltılı sayılar arasında yapılabilecek dönüştürmelerin kolaylığından dolayı ikili sayıların onaltılı gösterimini kullanacağız.

#### İkiliden Sekizliğe

Bitler, ikili noktadan başlayarak ve sola doğru ilerleyerek üçerli gruplara ayrılır. Üç-bitlik her bir grubun yerine sekizli (0-7) karşılığı konur.

Örneğin  $11001011101_2$ 'yi ele alalım.

$$\begin{array}{cccc} 11 & 001 & 011 & 101 \\ 3 & 1 & 3 & 5 \end{array}$$

Böylece,  $11001011101_2 = 3135_8$  değeri bulunur.

İkili sayıların daha kullanılabilir bir büyüklüğe nasıl sıkıştırıldığına dikkat ediniz.

#### İkiliden Onaltılıya

Bu yöntemde bitler, ikili noktadan başlayarak 4 bitlik gruplara ayrılır. Her grup yerine, eşdeğeri olan onaltılı bir basamak ile, 0-9, A,B,C,D,E,F konulur.

Örneğin  $11001011101_2$ 'yi ele alalım.

$$\begin{array}{ccc} 110 & 0101 & 1101 \\ 6 & 5 & D \end{array}$$

Böylece  $11001011101_2 = 65D_{16}$  sonucu elde edilir.

#### Sekizliden İkiliye

Burada ikiliden sekizliye dönüştürme yönteminin tersi kullanılır. Her sekizli basamak üç-bit ikili eşdeğeri ile değiştirilir. Önemli olan sözelimi, 3'ün 11 ile değil de 011 ile değiştirilmesi gerektiğidir.

Örneğin  $41357_8$ 'i ele alalım.

$$\begin{array}{ccccc} 4 & 1 & 3 & 5 & 7 \\ 100 & 001 & 011 & 101 & 111 \end{array}$$

Böylece  $41357_8 = 100001011101111_2$  sonucu bulunur.

#### Onaltılıdan İkiliye

Her bir onaltılı digit 4-bitlik ikili eşdeğeri ile değiştirilir.

Örneğin  $AB4C_{16}$ 'yi ele alalım.

$$\begin{array}{cccc} A & B & 4 & C \\ 1010 & 1011 & 0100 & 1100 \end{array}$$

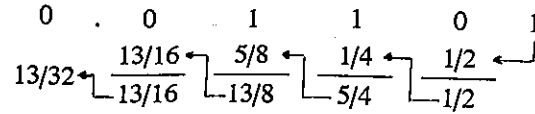
Böylece  $AB4C_{16} = 1010101101001100_2$  bulunur.

### 3.3.8 Kesirlerin Dönüştürülmesi

#### İkiliden Ondalığa

En sağdaki sıfır-olmayan bitten başlayarak, bu bit alınır ve ikiye bölünür. Ardından sonucu solundaki bite eklenir. Sonuç ikiye bölünür ve solundaki bite eklenir. İşleme, ikili noktaya ulaşana kadar devam edilir.

Örneğin  $0.01101_2$ 'yi ele alalım.



Böylece  $0.01101_2 = 13/32$  sonucu elde edilir.

#### Ondalıktan İkiliye

Ondalık kesir iki ile çarpılır ve (sonucun) tam kısmı yazılır. Bu tam sayı, 1 ya da 0 olacaktır. Daha sonra kesirli kısmı bırakmak üzere bu sayı ilk çarpımdan çıkan kesirli kısmı ile çarpılır ve tam kısım not edilir. Bu işleme gerektiği kadar devam edilir. İkili kesir, tam kısımları aşağıda gösterildiği gibi yukarıdan aşağıya doğru okuyarak oluşturulur.

Örneğin  $0.6875_{10}$ 'i ele alalım.

$$0.6875 \times 2 = [1]. 3750$$

$$0.3750 \times 2 = [0]. 7500$$

$$0.7500 \times 2 = [1]. 5000$$

$$0.5000 \times 2 = [1]. 0000$$

Böylece  $0.6875_{10} = 0.1011_2$  sonucu elde edilir.

Şimdi de  $0.1_{10}$ 'i ele alalım.

$$0.1000 \times 2 = [0]. 2000$$

$$0.2000 \times 2 = [0]. 4000$$

$$0.4000 \times 2 = [0]. 8000$$

$$0.8000 \times 2 = [1]. 6000$$

$$0.6000 \times 2 = [1]. 2000$$

$$0.2000 \times 2 = [0]. 4000$$

$$0.4000 \times 2 = [0]. 8000$$

$$0.8000 \times 2 = [1]. 6000$$

$$0.6000 \times 2 = [1]. 2000$$

$$0.2000 \times 2 = [0]. 4000$$

$$0.4000 \times 2 = [0]. 8000$$

vb.

Böylece  $0.1_{10} = 0.00011001100_2$ 'dir.

Daha önce de belirtildiği gibi  $0.1$  değeri, ikili kesir cinsinden tam olarak ifade edilemez.

### 3.3.9 BCD ve Diğer Kodlar

Bu kitap boyunca bir grup ikili basamak genellikle üç şeyden birini gösterecektir: sayısal bir değeri, bir komutu veya bir ASCII karakteri. Öte yandan bilgi sayma ve sayısal sistemler dünyasında birçok farklı kod vardır ve her biri tasarlanmış oldukları belli bir işe en iyi şekilde uyarlar. Özellikle yaygın bir kod BCD veya İkili Kodlanmış Ondalık'tır.

Teorik olarak BCD, pastanızı alıp yenmesi durumudur. Daha önce de belirtildiği gibi bilgisayar tasarımcıları tamamıyla ekonomik olmalarından dolayı iki-durumlu mantık elemanlarına güvenmeye zorlanmıştır. Dolayısıyla bu da, ikili aritmetik dünyasına ve sayısal değerlerin ikili ve ondalık gösterimleri arasındaki dönüştürmelerden doğan problemlere yol açmıştır. İkili kodlanmış ondalık sayılar her bir ondalık basamak 4-bitlik gruplara kodlayarak iki-durumlu mantığın kullanılabilirliğini kabul eder. Tablo 3.2'de, 0'dan 9'a kadar on basamak BCD'de nasıl gösterildiğini ve ondalık bir sayının BCD biçimine nasıl dönüştürüldüğünü göstermektedir.

Tablo 3.2 BCD kodları

| ondalık | BCD  |
|---------|------|
| 0       | 0000 |
| 1       | 0001 |
| 2       | 0010 |
| 3       | 0011 |
| 4       | 0100 |
| 5       | 0101 |
| 6       | 0110 |
| 7       | 0111 |
| 8       | 1000 |
| 9       | 1001 |

Ondalık bir sayıyı BCD formuna dönüştürmek için, her bir ondalık basamak 4-bitlik BCD grubu olarak kodlanır. Örneğin,

$$1942 \rightarrow 0001\ 1001\ 0100\ 0010$$

1010'dan 1111'e kadar ikili kodlar fazlalıktır ve BCD kodu olarak bir anlamı yoktur.

BCD aritmetiği ondalık aritmetikten farklıdır ve sadece on basamak gösterimlenmesinde farklılık gösterir. Aşağıdaki örnek bir BCD toplamasının nasıl yapıldığını göstermektedir.

$$\begin{array}{rcl}
 1942 & \rightarrow & 0001\ 1001\ 0100\ 0010 \\
 + 2379 & \rightarrow & +\ 0010\ 0011\ 0111\ 1001 \\
 \hline
 4321 & \rightarrow & 0100\ 0011\ 0010\ 0001
 \end{array}$$

BCD, ondalıktan ikiliğe dönüştürmeyi kolaylaştırdığından iyi bir çözüm gibi görünse de, iki dezavantaja sahiptir. İlk BCD aritmetiğinin ikili aritmetikten çok daha karmaşık olmasıdır. Bunun nedeni ikili tabloların aşırı derecede küçük oluşu ve donanımının birkaç kapı ile gerçekleştirilebilmesidir. Ne var ki, ondalık tablolar

0-9 arası basamaktan tüm kombinasyonlarını içerir. Bugünün sayısal teknolojisi bu dezavantajlar, her bir kapının büyük ve pahalı birer parça olduğu bilgisayar teknolojisinin ilk günlerinden daha az değerlidir. Ne var ki, eğilimler bir başladıktan sonra, hareket kazanma ve ilk başlatan kuvvet ortadan kalktıktan sonra bile çok daha ileri gitme yolundadırlar. Bu bilgi sayma dünyası için özelli doğrudur. Burada, ilk zamanlarda, devre tasarımının en ana kriterlerinden devredeki toplam lamba veya transistör sayısını en aza indirmektir. Bugün için devredeki transistör sayısı tasarımcı için yalnızca küçük bir problemdir.

BCD'nun en büyük dezavantajı verimsiz saklama alanına ilişkindir. Bilgisayarlar, bellek modülleri için para ödemek zorunda olduklarından, mantıksal olarak verilen bir iş için minimum bellek istenmesidir. Bir BCD basamak 4-bitlik saklama alanı gerektirir fakat sadece on sembol onaltı olası ikili kodun on eşlenirler. Bunun sonucu olarak, 1010-1111 (10-15) arası ikili kodlar fazlalıktır. Fazladan bir saklama ortamını ifade ederler. Saf ikili sayılar ise, ondalık basama başına ortalama yaklaşık 3.3 bite ihtiyaç duyar.

BCD'nun bu dezavantajlarına rağmen, cep tipi hesap makineleri veya sayı saatler gibi çok az saklama gerektiren uygulamalarda sık sık kullanılırlar. Bazı mikroişlemcilerde BCD işlemlerine yardımcı olması için özel komutlar vardır. Birçok BASIC dili yorumlayıcısı tüm sayısal işlemleri BCD sayılarıyla yapar. Aslında yukarıda verilen temel BCD koduna ek olarak, BCD sayılarını göstermenin bir dizi farklı biçimi vardır. Bu kodların her birinin, kendisini belli bir uygulama için uygun hale getiren has özellikleri vardır. Diğer BCD kodları bu konuyu kapsamı dışındadır.

#### Ağırlıksız Kodlar

Kısım 3.2'de açıklanan ikili koda çoğu zaman saf ikili, doğal ikili veya 8421 ağırlıklı ikili denir. 8, 4, 2 ve 1 basamaklı koddaki her bir basamağın ağırlığı gösterimler. Her birinin kendine has özellikleri olan ve bu tipte bir ağırlıklı uygulanmadığı birçok kod vardır. Ağırlıksız bir kod biçimi Birim Uzaklık Kod'tur. Herhangi iki ikili değer aralarındaki Hamming uzaklığına göre karşılaştırılır. İki sözcük arasındaki Hamming uzaklığı aralarındaki basamak sayısı farkıdır. Aşağıdaki örneklerle bu konu daha da açıklığa kavuşacaktır:

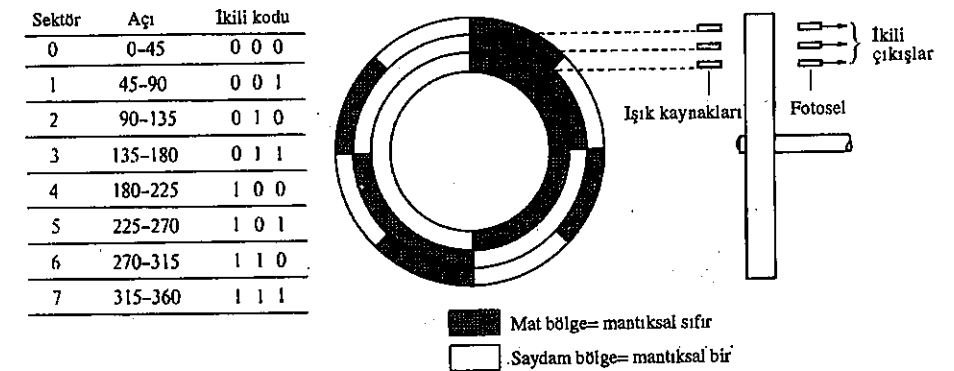
|                   |          |          |          |          |
|-------------------|----------|----------|----------|----------|
| 1. Sözcük         | 00101101 | 00101101 | 00101101 | 00101101 |
| 2. Sözcük         | 00101100 | 11101100 | 11101101 | 00100101 |
| Farklı basamaklar | ✓        | ✓✓       | ✓        | ✓✓       |
| Hamming uzaklığı  | 1        | 3        | 2        | 1        |

Birim uzaklık kodunda, ardışık kod sözcükleri arasındaki uzaklık sabittir. Doğal ikili sayılar için bu doğru değildir, çünkü burada, örneğin, 0111 ve 1000 ardışık değerlerdir fakat aralarında 4 birimlik bir uzaklık vardır. En yaygın olarak karşılaşılan birim uzaklık kodu Grey Kod'dur ve ilk 16 değeri Tablo 3.3'te verilmiştir.

Grey Kod çoğu zaman optik kodlayıcılarda, bir milin açısını ikili bir değere dönüştüren teknikte birlikte kullanılır. Şekil 3.2'de doğal bir ikili kod kullanan optik bir kodlayıcı, Şekil 3.3'te de aynı düzenleme Grey kodlanmış bir disk olarak gösterilmiştir.

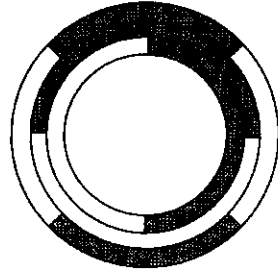
Tablo 3.3 Gray kodu

| Ondalık değer | Doğal ikili değer | Gray kodu |
|---------------|-------------------|-----------|
| 0             | 0000              | 0000      |
| 1             | 0001              | 0001      |
| 2             | 0010              | 0011      |
| 3             | 0011              | 0010      |
| 4             | 0100              | 0110      |
| 5             | 0101              | 0111      |
| 6             | 0110              | 0101      |
| 7             | 0111              | 0100      |
| 8             | 1000              | 1100      |
| 9             | 1001              | 1101      |
| 10            | 1010              | 1111      |
| 11            | 1011              | 1110      |
| 12            | 1100              | 1010      |
| 13            | 1101              | 1011      |
| 14            | 1110              | 1001      |
| 15            | 1111              | 1000      |



Şekil 3.2 Doğal ikili kodlanmış bir optik kodlayıcı

| Sektör | Açı     | Grey kodu |
|--------|---------|-----------|
| 0      | 0-45    | 0 0 0     |
| 1      | 45-90   | 0 0 1     |
| 2      | 90-135  | 0 1 1     |
| 3      | 135-180 | 0 1 0     |
| 4      | 180-225 | 1 1 0     |
| 5      | 225-270 | 1 1 1     |
| 6      | 270-315 | 1 0 1     |
| 7      | 315-360 | 1 0 0     |



Şekil 3.3 Gray kodlanmış bir optik kodlayıcı

Optik kodlayıcı; milin açısal konumunun, mil ile ölçü aleti arasında herhangi fiziksel bağlantı olmadan elektronik olarak belirlenebilmesine olanak sağlar. Mil ucunda, milin konumunu gösteren koddaki her bir bit için eş merkezli bir grup olan cam veya plastik bir disk vardır. Dört-bitlik bir kod, bir rüzgar fırıldığına benzer rüzgar yönü göstergesi için uygundur. Bir milin makine içindeki konumunu belirtmek için 10-bitlik bir kod gerekebilir. İzlerin her biri mat veya saydam olan bir grup sektöre bölünmüştür. Diskin bir tarafında, her izde bir tane olmak üzere bir grup ışık kaynağı (LED veya akkor-telli lambalar) yer alır. Diskin öteki tarafında da aynı sayıda fotoelektrik algılayıcılar vardır. Her algılayıcı ışık kaynağına karşısına yerleştirilmiştir. Böylece, diskin herhangi bir konumu için, belli bir fotoelektrik-hücre kombinasyonu, ışık hüzmelerini, ışık kaynağı ile algılayıcı arasında saydam bir sektör olup olmamasına bağlı olarak algılayacaktır.

Doğal ikili kodun yarattığı problem belli sektörler arasında iki veya daha çok bit durum değiştirmesidir. Ne yazık ki, fotoelektrik hücreler, kusursuz biçimde hıza ulaşamadıklarından, iki bitin durum değiştirmesinde, bir bit ötekenden önce değişmektedir. Dolayısıyla, sözgelimi, 001'den 010'a kadar değişme, 001, 000, 010 sırası ile görülebilir. En küçük değerlikli bit (EKDB) ortadaki bittten önce değiştiği için gerçekte olmayan bir '000' kodu anlık olarak üretilir. Bu, bazı uygulamalar için oldukça problem yaratabilir. Şekil 3.3'te de görülebileceği gibi Grey kodlanmış bir diskin herhangi bir anda sadece bir bitin değişmesi gibi bir özelliği vardır. Böylece, doğal ikili sistemde ortaya çıkan problemleri çözer. Grey kod sayısal bir sistemi bir kez okunduktan sonra normal şekilde işlenmek üzere doğal ikili koda dönüştürülebilir.

### Hata Algılama Kodları

Hata algılama kodları konusu bir grup ders kitabını dolduracak hacimdedir. Burada sadece hata algılama kodlarının en basitlerine, eşlik biti kullananlara bakacağız. Elektronik bir sistemde, sistemin doğru çalışmasını engelleyen ve genelde gürültü adı verilen, istenmeyen rastgele sinyaller bulunur. Bu rastgele sinyaller, sayısal sistemdeki elektronların termik hareketinden, civardaki yıldırımlarda kaynaklanacak elektromanyetik radyasyona kadar çeşitli nedenlerden doğabilir. Bu istenmeyen sinyallerin genlikleri, bilgisayarın içindeki sayısal sinyallerle karşılaştırıldığında genelde çok düşük düzeydedir. Sıfır ve bir ikili durumlarını gösterimleyen elektriksel iki düzey birbirinden o kadar iyi ayrılmıştır ki, iyi-yapılmış bir bilgisayarın içinde bir düzey, diğerine, hemen hemen hiç bir zaman 'kendiliğinden' dönüşmez.

Sayısal sinyallerin kablolarla uzun mesafelere iletildiği durumlarda, sayısal sinyallerin genlikleri küçülür ve harici gürültü sinyallerinin sayısal sinyallerin düzeyini geçerek onlar üzerinde bozucu etkiye bulunabilir. Bu pek çok kişi için bilinen bir durumdur. Çünkü radyo veya televizyonlarını uzak istasyonlara ayarlamaya çalışanlar ses veya görüntü kalitesinin bölgesel bir istasyonunkinden daha düşük olacağını bilirler. Sayısal sinyallerin alınmasında, hata ortaya çıkan her durumda, bu hatanın algılanabilmesi çok önemlidir; böylece bozulmuş verinin tekrar gönderilmesi için yeni bir istek yapılabilir. Algılama, istenen sayısal bilgiler, artı genliği, bilgi bitlerinin bir fonksiyonu olan bir veya daha çok kontrol biti göndererek yapılır. Bu kontrol bitleri yeni bir bilgi taşımadığından, fazladan bit adını alırlar. Bilgi bitleri, veri hattının alıcı ucunda kontrol bitlerini (yerel olarak) hesaplamak için kullanılır. Eğer alınan kontrol bitleri ile yerel olarak üretilen kontrol bitleri aynıysa, iletimin hatasız olduğu kabul edilir; aksi takdirde alıcı vericiye kayıp veriyi tekrar iletmesini isteyen bir mesaj gönderir. Bu konu 8. Bölümde tekrar ele alınacaktır.

Hata algılama kodlarının temelinde yatan düşünce  $m$ -bitlik bir kod-sözcüğünün  $2^m$  farklı mesaj gönderebilmesidir. Eğer  $n$ -bitlik bir kod sözcüğü yaratmak için  $m$  tane mesaj bitine  $r$  tane kontrol biti eklenirse  $2^n = 2^{m+r}$  tane olası kod sözcüğü ortaya çıkacaktır. Bu kod sözcüklerinden sadece  $2^m$  tanesi geçerlidir. Eğer bu  $2^m$  taneden birine eşit olmayan bir kod sözcüğü alınır, bir hata durumu söz konusudur. En kolay hata algılama kodu tek bir eşlik bitini içerir.

İki tür eşlik vardır; çift eşlik ve tek eşlik. Önce çift eşlik kodlarını ele alacağız. Tek bir çift eşlik kontrol biti kullanan bir hata algılama kodu, eşlik bitini sözcükteki toplam bir sayısını çift yapmak üzere seçer. Burada toplam bitlerinin sayısı eşlik bitinin kendisini de kapsar. Böylece, eğer mesaj 0101101 ise, sözcükte dört tane

bir bulunduğundan eşlik biti sıfır olarak seçilir. Böylece kod sözcüğü 001011 olur ve eşlik biti en büyük değerlikli bit konumuna eklenir, fakat ilke olarak eşlik bitinin belirli bir noktaya yerleştirilmesi gibi bir zorunluluk yoktur.

Yukarıda verilen sözcüğün gönderildiğini ve karşıdan 00101100 olarak alındığını varsayalım. Sözcüğün alıcıda hesaplanan eşliki tektir (3 tane 1 var) ve bir hata söz konusu olduğu varsayılır. Ne var ki, alınan sözcükteki hangi bitin hatalı olduğu söyleyemeyiz böylece hatayı düzeltemeyiz. İki hata olsaydı eşlik ihlalinin algılanmayacağına dikkat ediniz. Tek eşlik kontrol bitleri, hataların nispeten sık olmaması ve tek tek oluşma eğiliminde olduğu yerlerde yararlıdır. Tek eşlikli bir sistemde, eşlik biti toplam birlerin sayısını tek yapmak üzere seçilir. Tablo 3.4, hem çift hem de tek eşlik için 3-bitlik bir mesajda geçerli sekiz kodu verir. Her iki durumda da eşlik biti en ağırlıklı bittir.

Tablo 3.4 Tek ve çift eşlik kodları

| Mesaj | Kod sözcüğü<br>(çift eşlik) | Kod sözcüğü<br>(tek eşlik) |
|-------|-----------------------------|----------------------------|
| 000   | 0000                        | 1000                       |
| 001   | 1001                        | 0001                       |
| 010   | 1010                        | 0010                       |
| 011   | 0011                        | 1011                       |
| 100   | 1100                        | 0100                       |
| 101   | 0101                        | 1101                       |
| 110   | 0110                        | 1110                       |
| 111   | 1111                        | 0111                       |

Yukarıdaki düşünce, blok (matris)-eşlik hata algılama kodlarına genişletilebilir. Bu durumda, iki tür eşlik kullanılır. Her bir ikili sözcük bitlerin bir sütunu olarak yazılırsa, bir sütuna eklenen eşlik biti "boyuna (düşey) eşlik biti" olarak adlandırılır. Eğer bir sözcük bloku bir grup sütundan meydana geliyorsa bitlerin karşısında eşlik hesaplanarak bir eşlik sözcüğü oluşturulabilir. Her sözcük, sözcüğü, dörtlük bittense oluşsun: D<sub>0</sub>, D<sub>1</sub>, D<sub>2</sub> ve D<sub>3</sub>. Böylece D<sub>0</sub>'ın eşlik biti, bloktaki tüm D<sub>0</sub>'ların eşliklerini alarak hesaplanabilir. D<sub>1</sub>, D<sub>2</sub> ve D<sub>3</sub>'ün eşlik bitleri de benzer şekilde üretilir. Bu eşlik biçimine yatay veya enine eşlik denir. Bu tür bir sözcük blokunda tek bir hata oluşursa düşey eşlik biti, hatalı sözcüğü, yatay eşlik biti de hatalı biti gösterecektir. Hatalı sözcüğün ve bitin bilinmesiyle hatayı düzeltmek mümkün hale gelmektedir.

Blok hata algılama koduna basit bir örnek Tablo 3.5'te verilmiştir. Burada çift eşlik kullanılmıştır ve her sütun dört-bitlik bir sözcüğü gösterimler - üç veri biti artı bir eşlik biti. 7. Sözcük, önceki sözcüklere karşı oluşturulan yatay eşlik sözcüğüdür.

Tablo 3.5 Düşey ve yatay eşlik

| Bit            | 1.sözcük | 2.sözcük | 3.sözcük | 4.sözcük | 5.sözcük | 6.sözcük | 7.sözcük |                          |
|----------------|----------|----------|----------|----------|----------|----------|----------|--------------------------|
| D <sub>0</sub> | 0        | 1        | 1        | 0        | 1        | 0        | 1        |                          |
| D <sub>1</sub> | 1        | 0        | 0        | 1        | 0        | 1        | 1        |                          |
| D <sub>2</sub> | 1        | 1        | 0        | 1        | 1        | 0        | 0        |                          |
| D <sub>3</sub> | 0        | 0        | 1        | 0        | 0        | 1        | 0        | ←, düşey eşlik bitleri   |
|                |          |          |          |          |          |          |          | ↑<br>yatay eşlik bitleri |

Eğer blokta tek bir hata oluşursa sütununda ve satırındaki eşlik hatası belirlenir. Aynı veri blokundaki tek bir hatayı ele alalım. Blok seri olarak, her seferinde bir bit biçiminde gönderilir ve yukarıdaki blok ikili dizi olarak,

0110101010010110101001011100  
şeklinde görünür. Alınan dizinin

0110101011010110101001011100  
olduğunu varsayalım.

Alınan dizi, alıcıda, Tablo 3.6'daki gibi sözcüklere dönüştürülür.

Tablo 3.6 Tek bir hatanın algılanması

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | ✓ |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | X |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | ✓ |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | ✓ |
| ✓ | ✓ | X | A | ✓ | ✓ | ✓ |   |

Tik işareti eşliğin doğru olduğu satır ve sütuna, çarpı işareti de doğru olmadığı yerlere konur. Yukarıdaki örnekte hatalı bit, eşlik ihlalinin olduğu satır ve sütunun kesiştiği yerde algılanır. Böylece 1001 sözcüğü 1101 şeklinde hatalı olarak alınsa bile düzeltilebilir. Blok eşlik kodu tek hataları algılayabilip düzeltebilse de, çoklu hata kombinasyonları algılayabilir fakat düzeltemez.

### Huffman Kodları

Huffman kodlarına yapılan bu kısa giriş, konunun birden çok biçimi olduğunu göstermek için eklenmiştir. Huffman kodları bu kitapta bulunan diğer tüm kodlama tekniklerinden farklıdır. Çünkü değişken-uzunlukta bir kod sözcüğü kullanır. Huffman kodu kavramı yeni değildir. Samuel Morse ünlü kodunu tasarladığında asis-

tanını saklı oldukları her bir gözdeki harflerin sayısını (yani A'dan Z'ye) sayın için yazıcının başına gönderdi. Örneğin İngiliz dili metinlerinde E harfi oldukça sık bulunduğundan, birçok E vardı. Aynı şekilde, Q da o oranda az bulunuyor. Samuel Morse sıkça kullanılan harflerin kısa, seyrek kullanılan harflerin uzun kodları olan bir kod yarattı. Örneğin, E harfinin Mors kodu '.' ve Q harfinin '---.'dir.

Benzer bir düzenleme ikili kodlara genişletilebilir. Bu tür kodların bazı harf ve harf gruplarının diğerlerinden daha sık yer aldığı bilgilere uygulandığını belirtmek gerekir. Normal bir metin (örneğin İngilizce yazılmış) bu türden duruma örneklerdir. İş kolaylaştırmak için aşağıdaki örneği ele alalım. Bir manavın sadece dört cins sebze; patates, soğan, fasulye ve avokado sattığını varsayalım (işleri kolay tutacağımızı söylemişim).

Manavımız, modern bir manav ve bir bilgisayar bilimcisi olarak, işini bilgisayarla yürütmektedir. Bir çeşitten bir malın her satın alınmasında, mallar ikili biçimde kodlanmakta ve diskte saklanmaktadır. İşin çok hareketli ve disk alanının da sınırlı olması nedeniyle, işlemleri mümkün olan en az saklama ortamını kullanacak şekilde kodlamak istemektedir. Başlangıçta iki-bitlik ikili kodu denemiştir (Tablo 3.7).

Tablo 3.7 İki-bitlik ikili kodlar

| Cinsi   | Kodu |
|---------|------|
| Patates | 00   |
| Soğan   | 01   |
| Fasulye | 10   |
| Avokado | 11   |

Eğer  $n$  tane işlem varsa, bunları kayıt etmek için toplam  $2n$  saklama alanı gerekir. İlk bakışta her işlemi kodlamak için ikiden daha az bit sayısından kaçabilmek mümkün değilmiş gibi görünür. Fakat, birazcık düşündükten sonra, müşterilerinin çoğunluğunun patates aldığını fark eder. Böylece düzenlemeyi Tablo 3.8'de görüldüğü gibi değiştirir.

Tablo 3.8 Huffman kodu

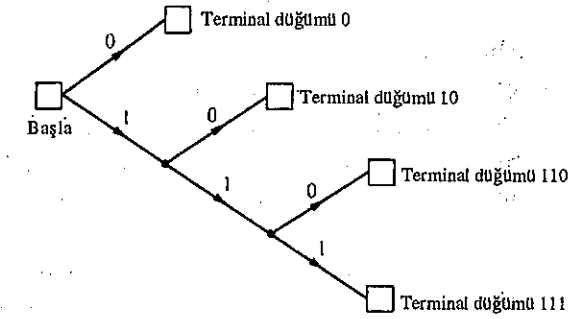
| Cinsi   | İşlem %'si | Kodu |
|---------|------------|------|
| Patates | 75         | 0    |
| Soğan   | 12.5       | 10   |
| Fasulye | 6.25       | 110  |
| Avokado | 6.25       | 111  |

Şimdi elinde farklı uzunlukta dört kod bulunmaktadır. Bir kodun uzunluğu 1-bit, diğerinin iki-bit, ikisinin de üç-bittir. Bir haftalık bir satıştan sonra, kaplanan toplam saklama alanı, her cins için işlem sayısı çarpı kod uzunluğu olacaktır. Ortalama kod uzunluğu yani,

$$1 \times \frac{3}{4} + 2 \times \frac{1}{8} + 3 \times \frac{1}{16} + 3 \times \frac{1}{16} = 1 \frac{3}{8} = 1.375$$

olacaktır.

Bu kodu, bir Huffman kodu kabul edip ortalama saklama alanı, işlem başına iki bitten 1.375 bite düşürülmüş ve % 31.25'lik bir alan tasarruf edilmiştir. Huffman kodları sık sık ikili bir ağaç biçiminde gösterimlenir. Şekil 3.4'teki ağaç manav örneğine karşılık gelmektedir.



Şekil 3.4 Huffman kodu

Bu şekle kimi zaman ağaç denir ve soldan sağa doğru okunur. Soldan itibaren, (düğüm 0, düğüm 10 vs. şeklinde etiketli) dört terminal nodunun her birine işaretli yolları izleyerek ulaşılabilir. Bu yollar kodu çözülecek bite bağlı olarak 1 veya 0 ise işaretlenmiştir. Aşağıdaki örnekte bu açıklanmıştır.

Manavın diskinde şu bit dizisi yer almaktadır; 001100101110.

Bu dizi hangi koda karşılık gelmektedir?

Dizinin ilk (ensoldaki) biti 0'dır. Ağaçtan görülebileceği gibi 0'lık bir ilk bit doğrudan 1. terminal düğümüne götürür. Dolayısıyla ilk kodu 0'dır. Benzer şekilde ikinci kod'da sıfırdır. Üçüncü kod bir 1 ile başlar ve devam etmek için bir başka biti incelememiz gerekir. Bu da birdir ve bir bitin daha okunması gerekir. Üçüncü bit 0'dır ve 110 terminal düğümüne götürür. Bu iş dizi aşağıdaki sıraya ayrılana kadar devam eder:

0 0 110 0 10 111 0.

## 3.4 İkili Aritmetik

İkili aritmetik ondalık aritmetikle tamamen aynı kuralları izler. İkili sayılar çalışmak için yapılması gereken tek şey ikili tabloları öğrenmektir; bunlar ondalık eşdeğerlerinden daha kolaydır. Tablo 3.9'da, toplama, çıkarma ve çarpma için ikili tabloları verilmiştir.

Tablo 3.9 İkili işlemler tablosu

| Toplama   | Çıkarma   | Çarpma    |
|-----------|-----------|-----------|
| 0 + 0 = 0 | 0 - 0 = 0 | 0 x 0 = 0 |
| 0 + 1 = 1 | 0 - 1 = 1 | 0 x 1 = 0 |
| 1 + 0 = 1 | 1 - 0 = 1 | 1 x 0 = 0 |
| 1 + 1 = 1 | 1 - 1 = 0 | 1 x 1 = 1 |

İkili aritmetik ile ilgili dikkate değer bir gerçek şudur ki; eğer toplamadaki 'elde' çıkarmadaki 'ödünç' kavramını hesaba katmazsak, toplama ve çıkarma işlemleri birbirinin aynısı olacaktır. Bu türde bir aritmetik vardır ve uygulandığı bazı önemli alanlar vardır; buna modül-2 aritmetiği denir.

$n$ -bitlik sayıların toplanması bir istisna dışında tümüyle kolaydır. O da her basamağındaki iki biti toplarken, bir önceki kattan gelen elde biti de toplama eklenmelidir.

$$\begin{array}{r}
 00110111 \\
 + 01010110 \\
 \hline
 11111 \quad \leftarrow \text{eldeler için boşluk} \\
 10001101
 \end{array}$$

Çıkarmada bilinen şekilde yapılabilir, fakat daha sonra göreceğiz ki, bilgisayarlar çıkarmayı bizim yaptığımız biçimde yapamazlar. Çünkü negatif sayılar çoğunlukla işaret artı değer biçiminde değil de tümleyen biçiminde gösterilir.

$$\begin{array}{r}
 01010110 \quad 86 \\
 - 00101010 \quad - 42 \\
 \hline
 11 \quad \leftarrow \text{ödünçler için boşluk} \quad 44 \\
 10001101
 \end{array}$$

İkili sayıların çarpımı, kaydırma ve toplama yoluyla "kalem-kağıt metodu" ile yapılabilir fakat pratikte bilgisayarlar, bir şekilde değiştirilmiş bir teknik kullanır. Şekil 3.5'e bakınız.

$$\begin{array}{r}
 01101 \quad 13 \\
 \times 01010 \quad \times 10 \\
 \hline
 00000 \\
 01101 \\
 \hline
 00000 \\
 01101 \\
 \hline
 001000010
 \end{array}$$

Şekil 3.5 İkili bir çarpma örneği

## 3.4.1 Yarım Toplayıcı

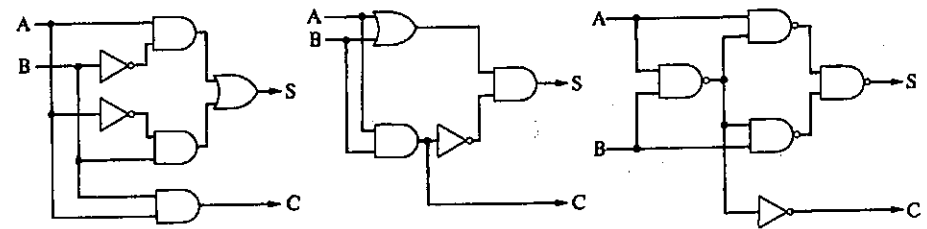
Kapıları, Boole cebirini, ikili aritmetiği görmüş olduğumuzdan, artık ikili sayıları toplamak için bir devre tasarımı üzerinde düşünebiliriz. En ilkel devreye yarım toplayıcı veya YT denir ve bu devre, S toplamı ve C eldesini vermek üzere iki biti toplar; Tablo 3.10'a bakınız.

Tablo 3.10 Yarım toplayıcının doğruluk tablosu

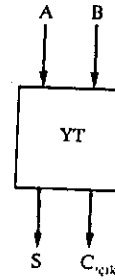
| A | B | S | C |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Doğruluk tablosundan da görülebileceği gibi bu iki bitin toplamı, bunların ÖZEL VEYA'larının alınmasıdır. Yani  $S = \bar{A}B + A\bar{B} = A \oplus B$ . Elde biti  $C = AB$  ile verilir.

Kapılarla ilgili daha önceki kısımlardan, bu devrenin en az üç farklı biçimde gerçekleştirilebileceğini biliyoruz; Şekil 3.6'ya bakınız. Bu devre, hangi biçimde gerçekleştirilmiş olursa olsun, birçok metinde çoğu kez Şekil 3.7'deki sembol ile gösterilir.



Şekil 3.6 Bir yarım toplayıcının üç değişik devrelendirilmesi



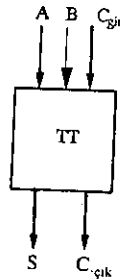
Şekil 3.7 Bir yarım toplayıcının devre sembolü

### 3.4.2 Tam Toplayıcı

Ne yazık ki, yarım toplayıcının kullanıldığı yerler çok kısıtlıdır. İki tane  $n$ -bit sayı toplanırken elde bitlerini de dikkate almalıyız. A'nın (sözelimi)  $a_i$  ve B'nin  $b_i$ 'nin toplanması,  $a_i$  ve  $b_i$ 'nin sağındaki basamağın toplanması sonucundan gelen  $c_{i-1}$  elde bitini ekleme şartını içermelidir. Bu diyagramsal olarak şu şekilde gösterimlenir:

$$\begin{array}{r} a_{n-1} \dots a_2 a_1 a_0 \\ + b_{n-1} \dots b_2 b_1 b_0 \\ \hline \end{array} \rightarrow \begin{array}{r} a_{n-1} \dots a_2 a_1 a_0 \\ b_{n-1} \dots b_2 b_1 b_0 \\ + c_{n-2} \dots c_1 c_0 \end{array}$$

İnsanlar toplama yaparken, eldeyi düşünmeden, otomatik olarak hesaba katarlar. Daha açık olarak 'Eğer bir elde ortaya çıkarsa, onu bir sonraki basamağa ekleyiz, eğer üretilmezse hiçbir şey yapmayız' derler. İnsan açısından 'hiçbir şey yapmamak' ve 'sıfır eklemek' eşdeğerdedir. Toplamayı yapabilmek için gereken mantık dikkate alındığında, bir önceki kattan gelen elde daima eklenmektedir. Burada eldenin değeri 0 veya 1 olur.



Şekil 3.8 Bir tam toplayıcının devre gösterimi

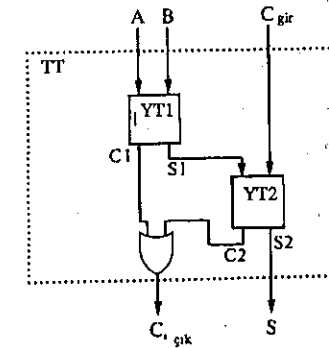
Tam toplayıcı, Şekil 3.8'deki sembollerle gösterilir, bir toplam ve elde-çıkışı üretmek için iki biti ve bir önceki kattan gelen elde-girişini toplar. Tablo 3.11'de, bir tam toplayıcının doğruluk tablosu verilmiştir.

Tablo 3.11 Bir tam toplayıcının doğruluk tablosu

| $C_{gir}$ | A | B | S | $C_{çık}$ |
|-----------|---|---|---|-----------|
| 0         | 0 | 0 | 0 | 0         |
| 0         | 0 | 1 | 1 | 0         |
| 0         | 1 | 0 | 1 | 0         |
| 0         | 1 | 1 | 0 | 1         |
| 1         | 0 | 0 | 1 | 0         |
| 1         | 0 | 1 | 0 | 1         |
| 1         | 1 | 0 | 0 | 1         |
| 1         | 1 | 1 | 1 | 1         |

Yarım toplayıcıyla aynı

Tam toplayıcı, gerçekleştirmenin klasik biçimi iki yarım toplayıcıyı birbirine bağlamaktır. Kavramsal olarak bir tam toplayıcı, A ve B'nin iki bitinin toplanmasını ve sonuca elde-girişinin eklenmesini gerektirir. Şekil 3.9'da, bir tam toplayıcının, iki yarım toplayıcı cinsinden nasıl gösterildiği verilmiştir.



Şekil 3.9 Bir tam toplayıcının gösterimi

Tam toplayıcının toplama çıkışı, ikinci yarım toplayıcının, YT2, toplama çıkışı ile ve elde-çıkışı de,  $C_{çık}$ , iki yarım toplayıcının eldelerinin VEYA'lanmasıyla verilir. Yukarıdaki devrenin tam toplama işlemini nasıl yaptığını göstermek için bir doğruluk tablosu kullanılabilir: Tablo 3.12'ye bakınız.

Tablo 3.12 İki yarım toplayıcı kullanılarak elde edilen bir tam toplayıcının doğruluk tablosu

| $C_{gir}$ | A | B | $S_1$ | $C_1$ | $S_2$ | $C_2$ | $C_{çık}$ |
|-----------|---|---|-------|-------|-------|-------|-----------|
| 0         | 0 | 0 | 0     | 0     | 0     | 0     | 0         |
| 0         | 0 | 1 | 1     | 0     | 1     | 0     | 0         |
| 0         | 1 | 0 | 1     | 0     | 1     | 0     | 0         |
| 0         | 1 | 1 | 0     | 1     | 0     | 0     | 1         |
| 1         | 0 | 0 | 0     | 0     | 1     | 0     | 0         |
| 1         | 0 | 1 | 1     | 0     | 0     | 1     | 1         |
| 1         | 1 | 0 | 1     | 0     | 0     | 1     | 1         |
| 1         | 1 | 1 | 0     | 1     | 1     | 0     | 1         |

↑ Toplam      ↑ Elde çıkış

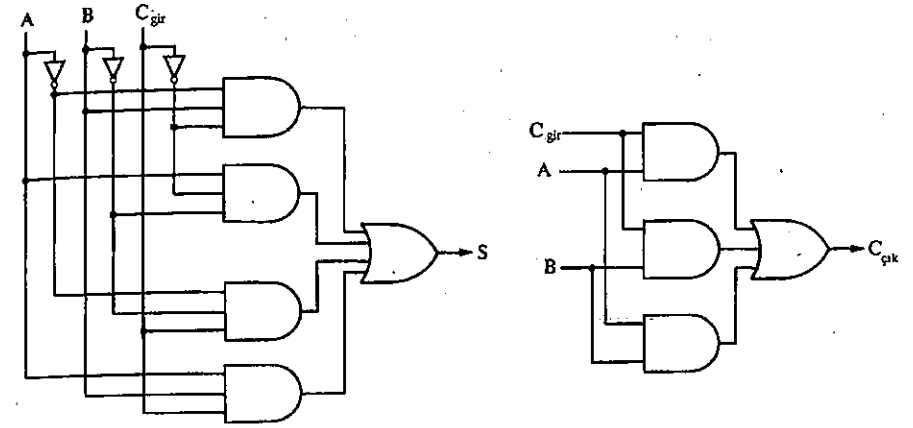
Okla gösterilen iki sütunun değeri, tam toplayıcının doğruluk tablosunda bunlar karşılık gelen sütunlarla aynı olduğundan, devremizin bir tam toplayıcı devresi olduğu sonucuna varırız. Pratikte çoğu zaman tam toplayıcı bu şekilde gerçekleştirilmez çünkü iki yarım toplayıcıdan geçen yayılma yolu altı birimlik gecikmeyi kapsar. Alternatif bir tam toplayıcı devresi doğruluk tablosundaki toplam ve elde denklemlerinden doğrudan türetilir. Toplam S, çıkış-eldesi  $C_0$  ve giriş-eldesi  $C_{gir}$  olsun.

$$\begin{aligned}
 S &= \bar{C} \bar{A} B + \bar{C} A \bar{B} + C \bar{A} \bar{B} + C A B \\
 \text{ve } C_0 &= \bar{C} A B + C \bar{A} B + C A \bar{B} + C A B \\
 &= \bar{C} A B + C \bar{A} B + C A (\bar{B} + B) \\
 &= \bar{C} A B + C \bar{A} B + C A \\
 &= \bar{C} A B + C (\bar{A} B + A) = \bar{C} A B + C B + C A \\
 &= A (\bar{C} B + C) + C B = A (B + C) + C B \\
 &= C A + C B + A B
 \end{aligned}$$

Çıkış-eldesinin değerinin mantık fonksiyonlarının çoğunluğunu oluşturduğuna ve iki veya daha çok giriş doğruyken doğru olduğuna dikkat edin. Yukarıdaki denklemlere karşılık gelen tam toplayıcının devre diyagramı Şekil 3.10'da verilmiştir. Bu devrede yarım toplayıcılar cinsinden gerçekleştirildiği eşdeğerinden daha çok kapı vardır (9'a karşı 12) fakat daha hızlıdır. Maksimum yayılma gecikmesi seri olarak 3 kapıdır.

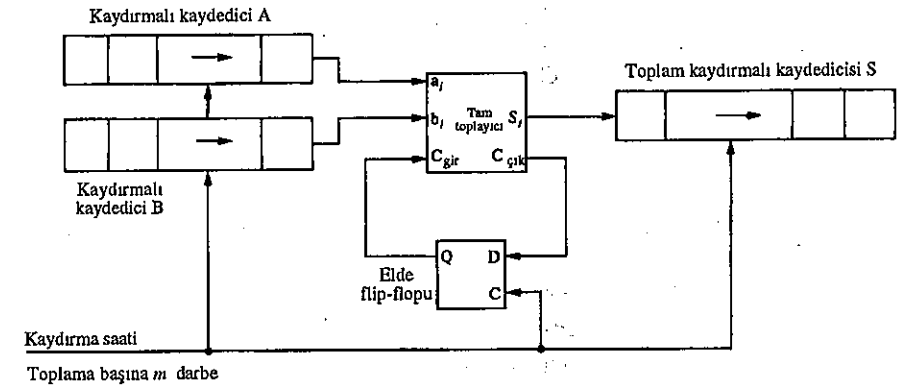
### 3.4.3 Sözcüklerin Toplanması

Genelde, iki tane  $n$ -bitlik sayıyı toplamak istediğimizden tam toplayıcının kendisi bile yeterince yardımcı değildir. Şekil 3.11'de verilen şema aracılığıyla sayıları seri



Şekil 3.10 Bir tam toplayıcı devresi

şekilde her seferinde bir bit olmak üzere eklemek mümkündür. A ve B'yi içeren kaydırmalı kaydediciler, her seferinde bir bit olmak üzere içerdikleri değeri tam toplayıcıya kaydırırlar. Her toplamın sonucu, sonuç veya toplam kaydedicisine kaydırılır. Tek bir flip-flop elde bitini tutar; böylece eski çıkış-eldesi bir sonraki giriş-eldesi olmuş olur.

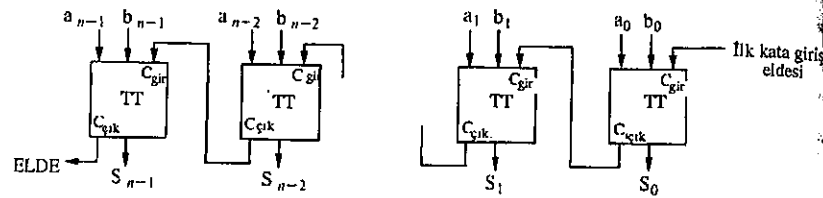


Şekil 3.11 Bir seri toplayıcı

$n$ . saat darbesinden sonra, S toplam kaydedicisi A ile B'nin toplamını içerir. Bu düzenleme bazı (genelde eski) ders kitaplarında şu yorum ile yer alır 'her seferinde sadece bir bit toplamak daha yavaş olsa da sadece bir tam toplayıcı kullandığından  $n$  tane tam toplayıcı kullanan bir paralel toplayıcıdan daha ucuzdur.' Bu eski çağlarda doğru olabilirdi ama artık değil. Günümüzde 100.000 veya daha çok

transistörlü tek bir yonga kullanabilme olanağı ile, devreler, kapı sayısını azaltan çok performansını optimize etmek üzere tasarlanabilir. Buraya, seri işlemler (bit bit) bir kerede sözcüğün tüm bitlerini kapsayan paralel işlemler arasında karşılığı göstermek için seri toplayıcıyı ekledim. 8. bölümde, bilgisayarların (mesafelerde) birbirleriyle, her seferinde bir sözcük gönderilen paralel iletişim çok, her seferinde bir bit olacak biçimde seri olarak iletişim kurduklarını göreceğiz.

Toplayıcıların daha pratik bir biçimi iki tane  $n$ -bitlik sayıyı toplamak için tam toplayıcıyı kullanan Şekil 3.12'deki paralel toplayıcıdır. Her tam toplayıcı çıkış-eldesi solundaki katın giriş-eldesi olur. Paralel bir toplayıcıda  $A$  sözcüğü  $n$  tane biti eşzamanlı tek bir işlemle  $B$  sözcüğünün  $n$  tane bitine eklenir. Paralel toplayıcı, teorik olarak buna karşılık gelen seri toplayıcıdan  $n$  kere daha hızlıdır. Pratikte, gerçek bir paralel toplayıcı elde-bitinin neden olduğu yayılma etkisiyle yavaşlar.



Şekil 3.12 Paralel toplayıcı

Şekil 3.12 ile ilgili dikkate değer birkaç nokta vardır. İlk olarak, en düşük değeri ilk katın bir yarım toplayıcı ile değiştirilebileceği çünkü giriş-eldesinin bulunmadığı (sağda kat yoktur) düşünülebilir. Öte yandan bu aşamada tam bir toplayıcı kullanarak, giriş-eldesi normal toplama için sıfıra veya  $A+B+1$  üretmek için bir kurulum yapılabilir (burada '+' mantıksal VEYA'yı değil, toplama işlemini ifade etmektedir). Eğer  $B$  sıfıra kurulusa,  $A+1$  üretilecek ve devre, arttırıcı işlevi görecektir. Bu olanağın çok yararlı olduğunu tüm aritmetiğe geldiğimizde gösterilecektir.

Bu devrenin bir başka özelliği son (en değerlikli bit) katından gelen çıkış-eldesi ile ilgilidir. Eğer iki tane  $n$ -bitlik sözcük toplanırsa ve sonuç  $111...1$ 'den büyükse bir çıkış-eldesi üretilir. Bilgisayar  $n$ -bitten uzun sözcükleri saklayamadığından toplam değeri, bellekte tek bir değer olarak saklanamaz. Son çıkış-eldesi (normalde bilgisayarın işlemci durum kaydedicisinin bir bölümünü oluşturan) bir flip-flop'a mandallanabilir. Toplama, programın bir parçası olarak yazılım ile yapılırken, programcının, sonucun aralık dışına taşıp taşmadığını kontrol etmek için elde bitini test etmesi normaldir.

Paralel toplayıcıya ilişkin son nokta paralel sözcüğünün anlamıyla ilgilidir. Açıkça görülmelidir ki ilk kat  $S_0$ 'ı elde etmek için  $a_0$  ile  $b_0$ 'ı toplarken ikinci kat sonucunun geçerli olduğundan emin olmadan önce ilk katın çıkış-eldesini beklemelidir.  $111...1+1$  şeklindeki en-kötü durumda elde tüm katlar boyunca dalgalanır. Bu devre bir grup ayrı saat çevrimlerine gereksinim duyulmadan  $A$ 'nın tüm bitlerinin  $B$ 'nin tüm bitlerine, tek bir işlemde eklenmesi açısından paraleldir.  $A$  ve  $B$ 'nin değerleri tam toplayıcıların girişlerine uygulandıktan sonra sistem bir sonraki işlem başlatılmadan önce devre oturana ve tüm eldeler yayılana kadar beklemelidir. Birçok sayısal sistemde saat periyodu, sistemdeki en yavaş devrenin en kötudurumdaki oturma süresidir. Daha gelişmiş sayısal sistemlerde, saat darbeleri, o an yerine yapılmakta olan işlem için olası en kötü durumdaki gecikmeye göre ayarlanmıştır ve her darbenin süresi ile değişen bir saat darbesi bulunur. Elde nedeniyle dalgalanma etkisini kısmen azaltmak mümkündür. 'Eldeden ötesini gören devreler' denen düzenlemeler bir grup sözcüğü, dört tam toplayıcı üzerinden bir eldeyi önceden tahmin etmek için kullanılabilir. Yani,  $i+5$  katına giren çıkış-eldesi,  $i+4$ ,  $i+3$ ,  $i+2$  ve  $i+1$  katlarının girişleri ile  $i+1$  katının giriş-eldesini özel bir yüksek-hızlı devre aracılığıyla incelenerek hesaplanır. Bu önceden tahmin edilen elde, eğer elde nedeniyle dalgalanma kullanılsaydı maruz kalınacak olan gecikmeden kaçınmak için beşinci katı besler. Bu devrelerin ayrıntılı yapısı kitabın kapsamının dışındadır.

### Tam Çıkarıcı

Tam toplayıcıya uyguladığımız teknikleri kullanarak, benzer şekilde bir tam çıkarıcı tasarlamak da mümkündür. Tam çıkarıcı yaygın olarak kullanılmadığından tasarımını, okura bir alıştırmaya bırakıyorum.

### 3.5 Negatif Sayılar

Her gerçek bilgisayar, negatif sayılar üzerinden de işlem yapabilmelidir. İnsanlar gerçekte negatif sayı kullanmazlar. Pozitif sayıları kullanır ( $-5$  olarak verilen  $5$ ,  $+5$  ile aynı değerdedir) ve aritmetik işlemlerde geçtiğinde, özel olarak ele alınmalarını anımsayabilmek için sayının önüne bir  $(-)$  işareti koyarlar. Böylece şu sonucu elde ederiz:

$$\begin{array}{r} 8 \\ + 5 \\ \hline 13 \end{array} \quad \text{ve} \quad \begin{array}{r} 8 \\ - 5 \\ \hline 3 \end{array}$$

Her iki durumda da sayılar aynı fakat onlarla yapılan işlemler farklıdır: ilk durumda

topladık, ikinci durumda çıkarttık. Bu teknik negatif bir sayının işaret ve büyüklük gösterilmesini vermek için bilgisayar aritmetiğine genişletilebilir.

### 3.5.1 İşaret ve Değer Gösterimi

$n$ -bitlik sözcüğün 0'dan  $2^{n-1}$ 'e dek değişen  $2^n$  olası değeri vardır. Örneğin, sekiz bitlik bir sözcük 0,1,...,254,255 sayılarını gösterilebilir. Negatif sayıları göstermenin bir yolu da en değerlikli biti alıp, sayının işaretini belirtmek üzere onun tersini almaktır; pozitifse 0, negatifse 1.

Örneğin, 8 bitte şunlar vardır:

|        |               |    |        |               |
|--------|---------------|----|--------|---------------|
| 0      | 0001101 = +13 | ve | 1      | 0001101 = -13 |
| işaret |               |    | işaret |               |
| biti   | sayının       |    | biti   | sayının       |
|        | değeri        |    |        | değeri        |

Bu, yaygın şekilde kullanılsa da negatif sayıların kusursuz şekilde geçerli bir gösterme şeklidir. İşaret + değer biçiminde  $n$ -bitlik sayılar şu ifade ile verilir:

$$-(2^{n-1} - 1) \text{ 'den } + (2^{n-1} - 1) \text{ 'e yani } -2^{n-1} + 1 \text{ 'den } + 2^{n-1} - 1 \text{ 'e.}$$

Yaptığımız tek şey,  $n$ -bitlik bir sayı alıp, bir bitini işaretini göstermek için kalan  $n-1$  bitini de sayıyı göstermek için kullanmaktır. Böylece 8-bit bir sayı -127'den (1111111) + 127'ye dek (0111111) sayıları gösterebilir. Bu sisteme itirazlardan biri sıfır için iki değer olmasıdır:

$$00000000 = +0 \text{ ve } 10000000 = -0$$

Kişisel görüşüm, karmaşık sayısal elemanları ekonomik olarak üretebildikten sonra bunun özel bir problem olmayacağıdır. Olasılıkla işaret ve değer notasyonu her bir kapının çok pahalı parçalar olduğu bilgisayarların ilk dönemlerinde terkedilmiştir. Bu sistemin reddedilmesindeki en büyük neden, ayrı ayrı çıkarıcılar gerektirmesidir. Aşağıda negatif sayıları diğer gösterme biçimlerinin ayrı toplayıcı ve çıkartıcı gereksinimlerini ortadan kaldırdığını göreceğiz. Öte yandan bugün MİB'deki ek bir çıkartıcının maliyetinin dikkate alınmayacak kadar düşük olduğunu vurgulamadan edemeyeceğim.

İşaret ve değer aritmetiğinde toplama ve çıkartma örneklere aşağıda verilmiştir. En değerlikli bitin işaret biti olduğu ve hesaplama da yer almadığı tekrar

anımsanmalıdır. Bu durum, işaret biti hesaplamalarda kullanılırken sayının bütünlüleyici bir kısmını oluşturduğu, ikiye tümler aritmetik ile (daha sonra görülecek) bir tezat oluşturur.

- $$\begin{array}{rcl} 001011 & + & 01011 \\ + 001110 & \rightarrow + & 01110 \\ & + & 11001 \end{array} \rightarrow 011001 \quad (6\text{-bit işaret ve değer sayısı olarak})$$
- $$\begin{array}{rcl} 001011 & + & 01011 \\ + 100110 & \rightarrow - & 00110 \\ & + & 00101 \end{array} \rightarrow 000101$$
- $$\begin{array}{rcl} 001011 & + & 01011 \\ + 110110 & \rightarrow - & 10110 \\ & - & 01011 \end{array} \rightarrow 101011$$
- $$\begin{array}{rcl} 001011 & + & 01011 \\ - 001001 & \rightarrow - & 01001 \\ & + & 00010 \end{array} \rightarrow 000010$$

### 3.5.2 Tümler Aritmetik

Tümler aritmetikte sayının negatifliği sayının kendisinin içinde tutulur. Bundan dolayı, sonuçta işaret kavramından (+ ve -) vazgeçilir. Eğer  $X$ 'i  $Y$ 'ye eklersek,  $X$  ve  $Y$  pozitifse işlem toplama, fakat  $Y$  negatifse ( $Y$ 'nin negatif biçimde gösterimlendiğini varsayarak) sonuç çıkartmadır. Tümler aritmetiğin sihirbazlık olmadığını göstermek için, ondalık tümlemeyi incelemekte yarar vardır.

$n$ -basamaklı bir ondalık sayının  $10$ 'a göre tümleyeni,  $N$ ,  $10^n - N$  olarak tanımlanır. Tümleyen aynı zamanda  $N$ 'nin her basamağını  $9$ 'dan çıkarıp sonuca  $1$  ekleyerek de hesaplanabilir. 4-hanelik ondalık sayı olan 1234'ü ele alalım. Bu sayının  $10$ 'a göre tümleyeni şudur:

$$\begin{array}{rcl} (a) 10^4 - 1234 & = & 8766 \text{ veya } (b) \quad 9999 \\ & & - 1234 \\ & & \hline & & 8765 + 1 = 8766 \end{array}$$

Bu sayıyı, bir başka sayıya, sözgelimi, 8576'ya ekleyelim. Şunu elde ederiz.

$$\begin{array}{r} 8576 \\ + 8766 \\ \hline 17342 \end{array}$$

Şimdi de 8576'dan 1234'ü çıkarmanın sonucunu klasik yoldan inceleyelim:

$$\begin{array}{r} 8676 \\ - 1234 \\ \hline 7342 \end{array}$$

İki işlemin de sonucunun en küçük değerlikli dört bitte aynı olduğuna, fakat beşinci basamakta  $10^4$ 'lük bir fark bulunduğu dikkat edin. Bunun nedenini bulmak zor değildir.  $Y$ 'nin  $X$ 'den çıkarılmasını ele alalım.  $Z = X - Y$  dir.  $Y$ 'nin çıkarılmasını  $10^4$  tümleyenini ekleyerek yaparsak, ki bu  $10^4 - Y$ 'dir, şunu elde ederiz:

$$Z = X + (10^4 - Y) = 10^4 + (X - Y)$$

Bir başka deyişle, istenen sonucu,  $X - Y$ , en solda istenmeyen bir basamak ile elde ettik. Bu basamak dikkate alınmayabilir.

### 3.5.3 İkiye Tümleyen Biçiminde Gösterme

$10^4$ 'a tümleyeninin ikili aritmetikteki eşdeğeri ikiye tümleyenidir.  $n$ -bitlik ikili bir sayının,  $N$ , ikiye tümleyenini bulabilmek için  $2^n - N$  değeri bulunur. Örneğin 5-bit'te,  $N = 00101$  ise,  $N$ 'nin ikiye göre tümleyeni  $100000 - 00101$  ile verilir.

$$\begin{array}{r} 100000 \\ - 00101 \\ \hline 11011 \end{array}$$

Böylece,  $-N$  11011 ile gösterilir.

Eğer bunu bir başka ikili değere eklersek, çıkarma işlemi yapmış oluruz.

$$\begin{array}{r} 01100 \quad 12 \\ + 11011 \quad + (-5) \\ \hline 100111 \quad 7 \end{array}$$

Ondalık örnekte olduğu gibi, doğru cevabı, dikkate alınmayacak  $2^n = 2^5$  terimi ile birlikte elde ettik. Devam etmeden önce, bir çift sayının pozitif ve negatif birleşimlerinin tümünün toplanmasını incelemekte yarar vardır.

Sözgelimi,  $X = 01001$  (9) ve  $Y = 00110$  (6) olsun.

$$\begin{aligned} -X &= 100000 - 01001 = 10111 \\ -Y &= 100000 - 00110 = 11010 \end{aligned}$$

$$\begin{array}{r} + X \quad 01001 \\ + Y \quad + 00110 \\ \hline 01111 = +15 \end{array} \quad \begin{array}{r} + X \quad 01001 \\ - Y \quad + 11010 \\ \hline 100011 = +3 \end{array}$$

$$\begin{array}{r} - X \quad 10111 \\ + Y \quad + 00110 \\ \hline 11101 = -3 \end{array} \quad \begin{array}{r} - X \quad 10111 \\ - Y \quad + 11010 \\ \hline 110001 = -15 \end{array}$$

Son iki durumda  $-3$  ve  $-15$  sonuçlarını beklemeliyiz.  $-3$ 'ün ikiye tümleyeni  $100000 - 00011 = 11101$  ve  $-15$  değerinin ikiye tümleyeni de  $100000 - 01111 = 10001$ 'dir. Sonuç,  $-X + Y$  doğrudur ve  $-3$ 'ü gösteren ikiye tümleyen biçimindedir. Son sonuç  $-X + -Y$ ,  $-15$ 'i gösterimler fakat  $2^n$ 'in eklenmesi ile. İki sayının da negatif olduğu bu durumda  $(2^n - X) + (2^n - Y) = 2^n + (2^n - X - Y)$  elde ederiz. İfadenin ilk kısmı fazlalık olan  $2^n$ 'dir ve ikinci kısım da  $-X - Y$ 'nin ikiye tümleyeni biçimidir. Şimdi artık ikiye tümleyen sistemi, pozitif ve negatif sayıların olası tüm birleşimleri için nasıl işlem yapıldığını görebiliriz.

İkiye tümleyenlerin oluşturulma kolaylığı olmasaydı ikiye tümleme sistemi bu kadar çekici olmazdı.  $N$ 'nin tümleyenini ele alalım:

$$-N \rightarrow 2^n - N$$

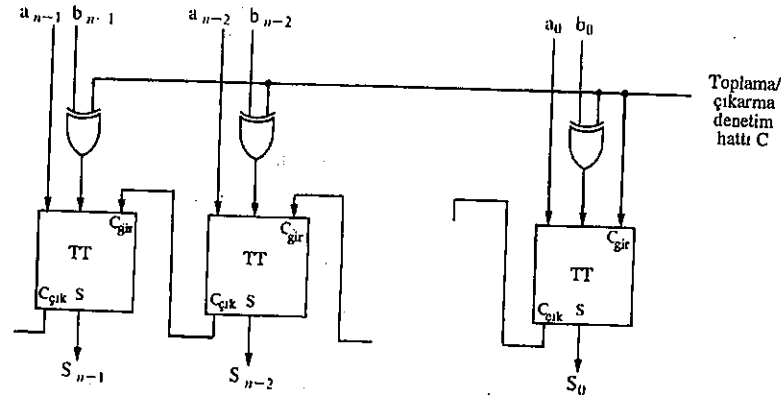
Diyelim ki,  $2^n$ 'den 1 çıkarılıp bunu sonuca ekleyerek denklemi yeniden düzenledik:

$$\begin{aligned} -N &\rightarrow (2^n - 1) - N + 1 \\ &\rightarrow \underbrace{111 \dots 1}_{n \text{ tane}} - N + 1 \end{aligned}$$

Bu işlem özellikle çok kolaydır, çünkü  $N$ 'nin bir biti 0 ise, bunu 1'den çıkarmak 1 sonucunu, bit 1 ise onu 1'den çıkarmak 0 sonucunu verecektir. Bir başka deyişle,  $1 - N_i = \bar{N}_i$ . Yani bir sayının ikiye tümleyenini oluşturmak için basitçe bitlerin tersini alıp 1 ekleriz.

$$\begin{aligned} \text{Örneğin 5 bitte, } 7 &= 00111 \\ -7 &= \overline{00111} + 1 = 11000 + 1 = 11001 \end{aligned}$$

Bu işlemin çekici olmasının nedeni donanım ile yapılabilmesinin kolay oluşudur. Bir toplayıcı/çıkarıcıyı gerçekleştirmek için Şekil 3.13'teki devre kullanılır. Şekil, birkaç ekstra devrenin eklenmesiyle, paralel bir ikili toplayıcının kolayca ikiye tümleyen sayıları için bir toplayıcı/çıkarıcıya nasıl dönüştürülebileceğini göstermektedir.



Şekil 3.13 Bir toplama/çıkarma devresi

Özel-VEYA kapılarının her birinin  $b_i$  ( $i=0$ 'dan  $n-1$ 'e kadar) ve  $C$ , bir denetim sinyali olmak üzere iki girişi vardır. Özel-VEYA'nın çıkışı  $b_i \overline{C} + \overline{b_i} C$ 'dir. Eğer  $C=0$  ise,  $\overline{C}=1$ 'dir, eğer  $\overline{C}=1$  ise,  $\overline{C}=0$ 'dır. ÖZEL VEYA kapıları, eğer  $C=1$  ise çıkışı giriş tersine çeviren,  $C=0$  ise girişi değişmeden geçiren programlanabilir bir tersleyici zinciri oluşturur. Aynı zamanda ilk tam toplayıcının giriş-eldesi girişini  $C$  olduğuna dikkat edin. Toplama yapılırken  $C=0$ 'dır ve giriş-eldesi sıfırdır. Çıkartma yapılırken  $C=1$ 'dir ve böylece toplamanın sonucuna bir eklenir. Bu bir,  $B$ 'nin ikiye tümleyenini oluşturmak için gereklidir.  $B$ 'yi daha önce tersine çevirmiştik; böylece bir eklemek  $B$ 'nin ikiye tümleyenini oluşturur ve  $B$ 'nin  $A$ 'ya çıkartılmasını sağlar.

#### İkiye Tümleyen Biçimindeki Sayıların Özellikleri

1. İkiye tümleyen sistemi gerçek bir tümleyici sistemdir; şöyle ki  $-X + (-X) = 0$ 'dır. Örneğin 5 bitte,  $+13 = 01101$  ve  $-13 = 10011$ 'dir.  $+13$  ile  $-13$ 'ün toplamı şudur:

$$\begin{array}{r} 01101 \\ + 10011 \\ \hline 100000 = 0 \end{array}$$

2. Tek bir tane sıfır vardır 00...0.

3. Eğer sayı pozitifse en büyük değerlikli bit 0, eğer negatifse EBDB 1'dir. Böylece EBDB işaret bitidir.

4.  $n$ -bitte ikiye tümleyen biçimindeki sayıların aralığı  $-2^{n-1}$ 'den  $+2^{n-1}-1$ 'e kadardır.  $n=5$  iken bu aralık -16'dan +15'e kadardır. Farklı sayıların toplam sayısının 32 (16 negatif, sıfır ve 15 pozitif) olduğuna dikkat edin. Bunun gösterdiği şey beş-bitlik bir sayının 32 farklı birimi açıklayabildiğidir ve bunlara, 0'dan 31'e doğal ikili sayılar veya -16'dan +15'e işaretli sayılar demek bize bağlıdır.

Şimdi de eğer ikiye tümleyen biçimindeki sayı aralığının dışına çıkarsak neler olacağına bakalım. Beş-bitlik göstermeyi seçersek, geçerli işaretli sayı aralığının -16'dan +15'e kadar olduğunu biliyoruz. Önce 5 ile 6'yı toplamayı, sonra da 12 ile 13 toplamayı deneyeceğimizi varsayalım.

$$\begin{array}{r} 5 = 00101 \\ + 6 = 00110 \\ \hline 01011 = 11_{10} \end{array} \quad \begin{array}{r} 12 = 01100 \\ + 13 = 01101 \\ \hline 11001 = 7_{10} \end{array} \quad (\text{ikiye tümleyen sayı olarak})$$

İlk durumda beklenen +11 sonucunu elde ettik, fakat ikinci durumda negatif bir sayı bulduk, çünkü işaret biti 1 idi. Eğer sonuç işaretsiz ikili bir sayı olarak ele alınsaydı, +25 olacaktı ki doğru cevap da budur. İkiye tümleyen sistemi işaretli sayıları göstermek için seçildiğinde, tüm cevaplar bu açıdan yorumlanmalıdır.

Benzer şekilde, eğer toplamı -16'dan küçük olan iki negatif sayıyı toplarsak, yine aralığın dışına taşarız. Örneğin,  $-9 = 10111$  ve  $-12 = 10100$  böylece:

$$\begin{array}{r} -9 \quad 10111 \\ -12 \quad 10100 \\ \hline 01011 \end{array} \quad \text{pozitif bir sonuç verir; } 01011 = +11_{10}$$

Bu durumların her ikisi de aritmetiksel bir taşmayı gösterir. İkiye tümleyen aritmetiğinde aritmetiksel taşma, iki pozitif sayı toplandığında negatif bir sonuç verirse veya iki negatif sayı toplandığında da pozitif bir sonuç verirse ortaya çıkar. Taşma cebirsel olarak ifade edilebilir. Eğer  $a_{n-1}$   $A$ 'nın,  $b_{n-1}$   $B$ 'nin,  $S_{n-1}$  de  $A$  ve  $B$  toplamının

işaret biti ise

$$V = a_{n-1} b_{n-1} s_{n-1} + a_{n-1} b_{n-1} s_{n-1} \text{ 'dir.}$$

Yani, A ve B'nin işaret bitleri aynı fakat sonucun işaret biti farklıysa aritmetik taşma oluşur. Aritmetik taşmanın ikiye tümleyeni aritmetiğinin sonucu olduğuna ve sayıların en değerlikli iki bitinin toplanmasıyla üretilen elde biti olan elde-çıkışı ile karıştırılmaması gerektiğine dikkat etmek gerekir.

#### İki Tümleyen Biçimindeki Sayılara Alternatif bir Bakış

$0 \leq N < 2^{n-1} - 1$ , aralığında bulunan ikili bir  $N$  tam sayısının negatif biçimde  $2^n$  ifadesi ile gösterildiğini, ve ayrıca bu ifadenin  $N$ 'nin bitlerini tersini alıp, sonuca bir ekleyerek kolayca değerlendirilebileceğini görmüştük.

İkiye tümleyen biçimindeki bir sayıya bir başka bakış yöntemi, onu en değerlikli bitinin negatif bir değeriğe sahip olduğu konumsal notasyondaki  $n$ -bit olarak gösterildiği şekliyle ele almaktır.

$$-N = -d_{n-1}2^{n-1} + d_{n-2}2^{n-2} + \dots + d_02^0$$

burada  $d_{n-1}, d_{n-2}, \dots, d_0$  ikiye tümleyeni biçimindeki  $D$  sayısının bitleridir. 14'ün ve ikiye tümleyeni biçimindeki -14'ün beş bit olarak gösterilmesini ele alalım.

$$\begin{aligned} +14 &= 01110 \\ -14 &= 2^n - N = 2^5 - 14 = 32 - 14 = 18 = 10010 \\ \text{veya } -14 &= \overline{01110} + 1 = 10001 + 1 = 10010 \end{aligned}$$

-14'ün bu gösterilmesi şu şekilde dikkate alınabilir:

$$\begin{aligned} &-1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ &= -16 + (0 + 0 + 2 + 0) \\ &= -16 + 2 = -14 \end{aligned}$$

İkiye tümleyen biçimindeki bir sayının aslında bu şekilde gösterildiğini ispatlamak mümkündür. Aşağıdaki ifadede  $N$  pozitif bir tamsayıyı,  $D$ 'de  $-N$ 'nin ikiye tümleyeni biçimini gösterir.  $-N = D$  olduğunu ispatlamak istiyoruz. Yani

$$-N = -2^{n-1} + \sum_{i=0}^{n-2} d_i 2^i \quad (1)$$

$N$  ve  $D$ 'nin bitleri cinsinden elimizde

$$-(N_{n-1}N_{n-2} \dots N_1N_0) = d_{n-1}d_{n-2} \dots d_1d_0 = D. \quad (2)$$

vardır.  $D$ 'nin bitleri, tersine çevirip bir ekleyerek  $N$ 'nin bitlerinden oluşturulur.

$$N_{n-1}N_{n-2} \dots N_0 + 1 = d_{n-1}d_{n-2} \dots d_0 \quad (3)$$

$D$ 'yi yok etmek için (3)'üncü denklemi (1). denkleminde yerine koyarak

$$-N = -2^{n-1} + \sum_{i=0}^{n-2} N_i 2^i + 1 \quad (4)$$

elde ederiz.

Fakat  $\bar{N}_i = 1 - N_i$ . Böylece;

$$\begin{aligned} -N &= -2^{n-1} + \sum_{i=0}^{n-2} (1 - N_i) 2^i + 1 \\ &= -2^{n-1} + \sum_{i=0}^{n-2} 2^i - \sum_{i=0}^{n-2} N_i 2^i + 1 \\ &= -2^{n-1} + (2^{n-1} - 1) - \sum_{i=0}^{n-2} N_i 2^i + 1 \\ &= -2^{n-1} + (2^{n-1} - 1) + 1 - N \\ (N &= \sum_{i=0}^{n-1} N_i 2^i = \sum_{i=0}^{n-2} N_i 2^i \quad N \text{ 'nin belirtilen aralığın içinde olması için} \\ &\quad \text{en ağırlıklı biti } N_{n-1}, \text{ sıfır olduğundan}) \\ &= -2^{n-1} + 2^{n-1} - N = -N \end{aligned}$$

#### 3.5.4 Bire Tümleyen Biçiminde Gösterme

İkiye tümleyeni aritmetiğine alternatif,  $n$ -bitlik bir  $N$  negatif sayısının gösterilmesinin  $2^n - N - 1$  ile verildiği birin tümleyeni aritmetiğidir. Bu, kendisine karşılık gelen ikiye tümleyeni biçiminde gösterilmeden bir azdır ve daha kolay şekilde  $N$ 'nin bitlerini tersini alarak oluşturulur. Örneğin,  $n = 5$  iken 4'ün 9'dan çıkarılmasını ele alalım (Şekil 3.14).

$$\begin{array}{r}
 + 9 = 01001 \rightarrow 01001 \\
 - 4 = \overline{00100} \rightarrow + 11011 \\
 \hline
 100100 \\
 + \quad \quad \quad 1 \\
 \hline
 00101
 \end{array}$$

Şekil 3.14 Bire tümleyen biçiminde çıkarma işlemi

Cevap bulunduktan sonra, en soldaki bit elde-etrafında-bitiş denen bir düzenlemede sonucun en küçük değerlikli bitine eklenir. Bire tümleyen sistemi gerçek bir tümleyen değildir. Çünkü  $X+(-X)$  sıfır değildir. Bundan başka, sıfır için iki gösterilme vardır: 00...0 ve 11...1. İşaretli sayıları göstermede bire tümleyen sistemi ikiye tümleyen sistemi kadar yaygın değildir fakat bazı bilgisayarlar tarafından kullanılır.

Şimdiye dek gördüğümüz sayı gösterimlerini çeşitli biçimlerde karşılaştırmak öğretici olacaktır. Tablo 3.13'de,  $n=5$  iken saf ikili sayıların, işaret ve değer, bire tümleyen ve ikiye tümleyen biçimindeki gösterimleri verilmektedir.

### 3.6 Bilgisayar Aritmetiğine ve Assembly Dilinde Programlamaya Giriş

Toplama ve çıkarmayı tam toplayıcılara göre donanım düzeyinde ele aldıktan sonra sıra, bu işlemlerin yazılım düzeyinde assembly diline göre nasıl yürütüldüğünü görmeye gelmiştir. İleriki kısımların assembly dili tarafından daha başından işgal edilmesini istemiyorum, fakat aşağıdaki notlar bize yardımcı olacaktır.

Assembly dili programların yazıldığı en ilkel dildir (bir an için, çoklu programlamayı-laf arasında bunun mikroişlemcilerle bir ilgisi yoktur-unutursak). Assembly dili, bilgisayar tarafından yapılabilecek çeşitli işlemleri göstermek için komut kısaltmaları kullanır. Örneğin ADD= topla, SUB= çıkar, BRA= dallan vs. Komut kısaltmaları, birçoğu apaçık olduğu halde, bilgisayardan bilgisayara değişir. Komut kısaltmalarını, çoğu zaman bir değişkene karşılık gelen bir AD izler. Eğer ADD FRED yazarsak, FRED bir sayıya değil fakat bellekteki konumuna karşılık gelir. Birçok bilgisayar komutta bir isim kullanmamızı olanaklı kıldığından, tüm ara sonuçlar için geçici bir tutma kaydedicisine sahip olmalıdırlar. Bu kaydedici akümülatör olarak bilinir. Çünkü tarihsel olarak sonuçları topladığı söylenebilir. Eğer komutları (örneğin ADD NUM1, NUM2, NUM3, NUM1= NUM2+NUM3 anlamına gelecek şekilde) iç adresi belirtebilen bir bilgisayarımız olsaydı akümülatöre gerek kalmayacaktı. Bu kitap boyunca, pek çok örnek 6502 mikroişlemcisine gönderme yapacaktır. Bunun nedenleri şunlardır:

Tablo 3.13 Negatif sayıların gösterilmesi

| İkili kodu | Saf ikili | İşareti ve büyüklüğü | Bire tümleyeni | İkiye tümleyeni |
|------------|-----------|----------------------|----------------|-----------------|
| 00000      | 0         | 0                    | 0              | 0               |
| 00001      | 1         | 1                    | 1              | 1               |
| 00010      | 2         | 2                    | 2              | 2               |
| 00011      | 3         | 3                    | 3              | 3               |
| 00100      | 4         | 4                    | 4              | 4               |
| 00101      | 5         | 5                    | 5              | 5               |
| 00110      | 6         | 6                    | 6              | 6               |
| 00111      | 7         | 7                    | 7              | 7               |
| 01000      | 8         | 8                    | 8              | 8               |
| 01001      | 9         | 9                    | 9              | 9               |
| 01010      | 10        | 10                   | 10             | 10              |
| 01011      | 11        | 11                   | 11             | 11              |
| 01100      | 12        | 12                   | 12             | 12              |
| 01101      | 13        | 13                   | 13             | 13              |
| 01110      | 14        | 14                   | 14             | 14              |
| 01111      | 15        | 15                   | 15             | 15              |
| 10000      | 16        | -0                   | -15            | -16             |
| 10001      | 17        | -1                   | -14            | -15             |
| 10010      | 18        | -2                   | -13            | -14             |
| 10011      | 19        | -3                   | -12            | -13             |
| 10100      | 20        | -4                   | -11            | -12             |
| 10101      | 21        | -5                   | -10            | -11             |
| 10110      | 22        | -6                   | -9             | -10             |
| 10111      | 23        | -7                   | -8             | -9              |
| 11000      | 24        | -8                   | -7             | -8              |
| 11001      | 25        | -9                   | -6             | -7              |
| 11010      | 26        | -10                  | -5             | -6              |
| 11011      | 27        | -11                  | -4             | -5              |
| 11100      | 28        | -12                  | -3             | -4              |
| 11101      | 29        | -13                  | -2             | -3              |
| 11110      | 30        | -14                  | -1             | -2              |
| 11111      | 31        | -15                  | -0             | -1              |

1. Öğrenci örnekleri için kullanılabilir bir büyüklük olan 8-bitlik veri sözcüğü kullanır.

2. Fiziksel tasarımını (mimarisini) ve assembly dilini hem kullanmak hem de anlamak kolaydır. Bundan başka, mimarisi tipik olmayan bir ana bilgisayar mimarisi değildir. Bazı bilgisayarlar, genel olarak daha az tipik bilgisayar özelliği gösterirler ve bu nedenle de öğretim amaçlarına daha az uygundur.

3. 6502 ucuz ve yaygın olarak kullanımdadır. Özellikle Apple ve BBC bilgisayarlarını temeli buna dayanır ve AIM 65 mikrobilgisayar sistemi, öğrencilere assembly dilinde programlar girme, bunları düzenleme, birleştirme, çalıştırma, ayıklama imkanları sağlamak için özel olarak tasarlanmıştır. Ayıklama, hatalardan arındırma anlamına gelir.

6502'nin 8-bitlik bir veri sözcüğü vardır ve bu yüzden de işaretli sayıları 0-255 (yani 0-2<sup>8</sup>-1) aralığında ve ikiye tümleyen biçimindeki işaretli sayıları da, -128+127 aralığında gösterir.

Sözgelimi, *P* ve *Q* sayılarını toplamak ve sonucu da *R* olarak adlandırmak istiyoruz. Artık *P*, *Q* ve *R*, 6502'nin belleğindeki fiziksel konumlara karşılık gelen adlardır. Ad kullanıyoruz çünkü, adresleri hatırlamaya çalışmakla uğraşmak istemeyiz. Bunu gerçekleştirecek program şöyledir:

```
LDA P
CLC
ADC Q
STA R
```

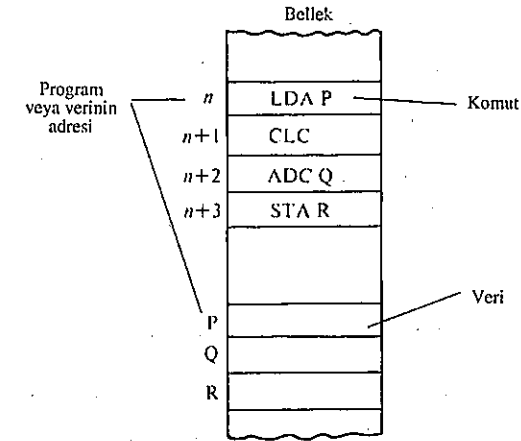
İlk komut *P* diye adlandırdığımız bellek konumunun içerdiği değerin akümülatöre taşınmasına neden olur. *LDA* komut kısaltması akümülatöre yükle anlamına gelir. Gerçek bir programda normalde programcının çeviriciye *P*, *Q* ve *R*'nin bellekte hangi konumlarda olduğunu 'söylemesi' gerekir. *LDA* işlemi *P*'deki sayıyı etkilemez. En önemli kurallardan biri de; içine yeni bir sayı yazarak bir bellek konumu veya kaydedicinin içerdiği değer değiştirilmediği sürece, bunların değişmeyeceğidir.

İkinci olarak, *CLC*, Eldeyi Sıfırla anlamına gelir. Elde biti işlemci durum kaydedicisinin bir bölümüdür (6800 teknolojisinde bu koşul kodu kaydedicisi olarak bilinir) ve bu komut onu temizler (yani sıfıra kurar). Eğer elde bitinin halihazırda sıfır olduğunu bilseydik bu komuttan vazgeçebilirdik.

Üçüncü komut, *ADC*, 'akümülatöre eldeli ekle' anlamındadır. Bu komutun etkisi *Q* bellek konumunun değerini *PSW*'deki kaydediciyle birlikte akümülatörde halihazırda bulunan veriye eklemektir. Birçok bilgisayarda bir sayıyı akümülatöre ekleyen basit bir *ADD* komutu vardır. 6502'de ise yoktur. Elde bitini az önce sıfırladığımızdan, *ADD*'in yaratacağı ile aynı etkiyi sağlayacağız. Şekil 3.15, programın ve verisinin bellekteki konumunu göstermektedir.

Son komut, *STA R*, akümülatörün değerini bir bellek konumunda saklar. Bizim örneğimizde bu konuma *R* denir.

Bu örnekle ilgili bir sorunuz yoktur. Okur 'işaretsiz ikili sayılarla mı yoksa, ikiye tümleyen biçimindeki sayılarla mı işlem yapıyoruz?' diye sorabilir. Cevap şudur: Bunun önemi yoktur. Aynı toplama işlemi her iki durumda da hizmet verecektir.



Şekil 3.15 Program ve verinin bellekteki konumu

Önemli olan programcının sonucu nasıl ele alacağıdır.

Bir aritmetik (ve başka bazı türden) işlemin bitiminde *MIB*, işlemci durum kaydedicisini günceller. 6502'de bu kaydedici, aşağıdaki biçimdedir.

|     |   |   |   |   |   |   |   |   |
|-----|---|---|---|---|---|---|---|---|
| bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|     | N | V |   | B | D | I | Z | C |

Burada sadece bit 0 ve 6 ile ilgileneceğiz. Bunlar sırasıyla elde ve taşmayı gösterir. Bilgisayar, programcının ne yapacağını bilemez. Dolayısıyla bir aritmetik işlemin sonunda sorar.

Elde üretildi mi (bit 7'den dışarı bir 1 çıktı mı)?

Eğer sayının ikiye tümleyeni biçiminde olduğu varsayılırsa, bir taşma var mı?

İşlemci durum kaydedicisindeki bilgiyi kullanmak, uygulamasına bağlı olarak, kullanıcıya kalmıştır.

Şimdi de Şekil 3.16'da görülen iki programı ele alalım.

| 1. durum | 2. durum |
|----------|----------|
| LDA P    | LDA P    |
| CLC      | CLC      |
| ADC Q    | ADC Q    |
| BCS HATA | BVS HATA |
| STA R    | STA R    |
| HATA     | HATA     |

Şekil 3.16 İşaretili ve işaretli toplama işlemine ilişkin programlar

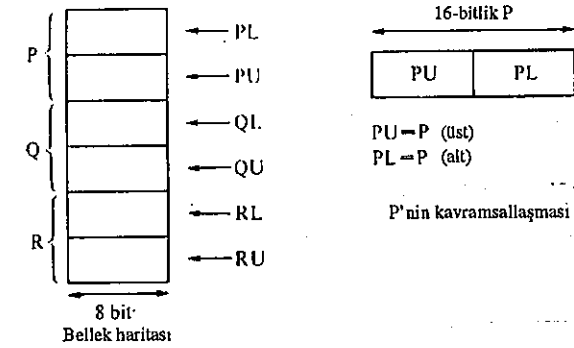
Durum 1'de aritmetiğin işaretli ikili olduğunu varsaydık. Eğer, ADC Q işleminden sonra, iki sayının toplamı 255'ten büyük çıkarsa, en değerlikli bit konumundan eldeye 1 gönderilir. Bunu izleyen komut, BCS, Elde 1'e kurulmuşsa Dallar anlamına gelir. Böylece eğer bir elde üretilmişse bir sonraki komut (STA R) yürütülmez. Bunun yerine programın HATA denilen kısmına bir dallanma veya atlama yapılır. Programın bu bölümü (burada gösterilmemiştir) problemi ele almak için tasarlanmış olmalıdır. Eğer bir dallanma yapılmazsa, koşullu dallanmanın hemen ardından gelen bir sonraki komut yürütülür.

Durum 2'de aritmetiğin ikiye tümleyen biçiminde olduğu varsayılmıştır. Toplamadan sonra taşıma bitinin kurulup kurulmadığını görmek için test etmenin dışında program Durum 1 ile aynıdır.

Bu örneklerin temel sınırlaması, sadece sekiz-bitlik sayıları ele almalarıdır. Küçük sayılara karşı kişisel bir şeyin olmadığı halde 255'ten daha büyük sayıların alanına girdiği zamanlar olmuştur. Bu durumda ya (ucuz) 6502'yi atıp (daha pahalı) bir 16-bitlik veya 32-bitlik makine satın alırım ya da büyük sayıları sekiz bitlik parçalar halinde ele alırım. Hızın çok önemli olmadığı durumlarda, ikinci çözüm daha uygun veya meslek argoyla söylenirse maliyet-etkindir.

Sözelimi 50.000'e kadar olan işaretli sayıları ele almam gerekti. 50.000'den ilk büyük  $2^{16} = 65536$ 'dır. 16-bitlik bir sözcük yaratmak için yapmak gereken tek şey, bellekten iki tane 8-bitlik sözcük alıp bunları 16-bitlik bir birim olarak dikkate almak olmalıdır. P, Q ve R'li bir önceki örneği kullanarak, bir bellek haritası çizebiliriz; Şekil 3.17'ye bakınız. Bellek haritası bellekteki verinin ve komutların yerleşiminin diyagramsal gösterilmesidir.

16-bitlik sözcüğümüz artık iki yarım parçadan oluşmaktadır. P; PU (en değerlikli 8 bit) ve PL ve (en küçük değerlikli 8 bit) ile gösterilir. Bellek haritasını PL ve PU



Şekil 3.17 8-bitlik bir bilgisayarda 16-bitlik sayıların kullanılması

birbirine bitişik olacak şekilde çizdim, fakat pratikte böyle yapılması için bir neden yoktur. R'yi elde etmek için P ve Q'yu toplamak üzere aşağıdaki işlemleri yerine getirmeliyiz:

$$\begin{array}{r} \text{PU} \quad \text{PL} \\ + \quad \text{QU} \quad \text{QL} \\ \hline \text{RU} \quad \text{RL} \end{array}$$

Yani, RL'yi elde etmek için PL ve QL'yi; ardından da RU'yu elde etmek için PU ve QU'yu toplarız. Dikkat edilecek tek nokta PL, QL'e eklenirken üretilen elde'nin (PU+QU) sütununa eklenmesi gerektiğidir. Assembly dili anlamında P ve Q'yu toplayacak program şöyledir:

```
LDA    PL
CLC
ADC    QL
STA    RL
LDA    PU
ADC    QU
STA    RU
```

İlk dört satır daha önceki gibidir. Şimdi, sıra PU ve QU'yu toplamaya geldiğinde elde bitini sıfırlamayız; böylece PL ve QL'yi toplarken üretilmiş olan bir elde otomatik olarak PU ve QU'ya eklenir. Bu düzenleme istenilen sayıda bir grup bayttan oluşan sayıları ele almak üzere genişletilebilir. Verilen bir sayıda bayt oluşan işaretli tam sayıların değişim aralığı şöyledir:

| Bayt | Aralık          |
|------|-----------------|
| 1    | 0-255           |
| 2    | 0-65.535        |
| 3    | 0-16.772.215    |
| 4    | 0-4.294.967.295 |

Çıkarma da toplamaya (yani, programcı açısından) çok benzer bir şekilde yapılır. R'yi elde etmek üzere P'den Q'yu çıkarmak için aşağıdaki program kullanılır:

```
LDA P   P'yi akümülatöre yükle
SEC      elde bitini kur
SBC Q    Q'yu elde bitiyle birlikte çıkar
STA R    sonucu R'ye sakla
```

Dikkat edilmesi gereken ilk şey assembly dili komutlarına ilişkin açıklamaların yanında yazılmış olmasıdır. PASCAL gibi üst-düzeyli bir dilde yazılmış bir program kendinden-açıklamalıdır. Çünkü program deyimleri okunursa, komutların ne olduğunu açık ve karışıklığa yol açmayacak şekilde anlaşılır. Ne yazık ki, bir assembly dili programında ayrıntılar, programın belkemiğini bütünüyle gölgede bırakır. Bunun sonucu olarak, uzun açıklamalı bir yorum olmadan program kısa sürede -yazarı için bile anlaşılabilir hale gelir.

Ele alınacak iki nokta ise şudur: Toplamayı yapmadan önce elde biti sıfırlanmıştır. Çıkarma için bir SEC (eldeyi kur) komutu ile 1'e kurulmalıdır. Bu işlem, makineye bağlı bir işlemdir. Örneğin çıkarmayı 6800 ile yaparken (6502'nin bir kardeşi işlemcisi) elde biti kurulmamalı, silinmelidir. Bu da göstermektedir ki assembly dili seviyesinde çalışma, kullanılmakta olan belli bir bilgisayarın işlemesiyle ilgili en eksiksiz bilgiye gerek duymaktadır.

SBC kısaltması 'eldeli çıkar' anlamındadır. Elde bit çıkarma sırasında ödünç bit olur. Ödünç ile elde arasındaki fark terminoloji ile ilgilidir. Bilgisayarda tek-bir bit yazmacı vardır ve insanların kendisine elde-biti veya ödünç-biti demeyi seçmesi onun için fazlaca önem taşımaz.

### 3.7 Kayan-Noktalı Sayılar

Daha ileri gitmeden önce, itiraf etmeyelim ki kayan-noktalı aritmetik bilgisayar biliminin çok-eğlenceli konularından biri değildir. Kayan noktalı aritmetiğin temel fikirleri oldukça basit olsa da, daha çok gerçekleştirilme ayrıntıları ile ilgilenilecektir.

Daha önce, sözcük uzunluğu küçük olan bir makinede büyük sayıları kullanabilmek için sözcükleri birbirine zincirlediğimizi göstermiştik. Fakat ikili kesirlere henüz bakmadık. İkili (veya ondalık) kesirlerde herhangi bir sorun yoktur. Aşağıdaki iki hesaplamayı ele alalım.

$$\begin{array}{r} 7632135 \text{ ve } 763.2135 \\ + 1794821 \quad + 179.4821 \\ \hline 9426956 \quad 942.6956 \end{array}$$

İlk durumda (soldaki örnek) tamsayılarla, ikincisi ise kesirli sayılarla ilgili olsa da, yapılan hesaplama tamamen aynıdır. Bu ilke bilgisayar aritmetiğine genişletilebilir. Bilgisayar programcısının yapması gereken tek şey, ikili noktanın bulunduğunu varsaydığı yeri hatırlamasıdır. Bilgisayara girilen tüm girişler bu kurala uymak üzere ve tüm çıkışlar da, benzer şekilde sıraya konurlar. İç işlemler, sayılar sanki tamsayı biçimindeymiş gibi yapılırlar. Bu düzenlemeye sabit-noktalı aritmetik denir. Sayıların sabit-noktalı biçimde gösterilmesinin avantajı bunu gerçekleştirmek için özel karmaşık bir donanım veya yazılıma ihtiyaç duyulmamasıdır.

Basit bir örnek sabit-noktalı aritmetik düşüncesini daha da açıklığa kavuşturacaktır. En büyük değerlikli dört biti tamsayı kısmını, en küçük değerlikli dört biti de ondalıklı kısmı gösteren 8-bitlik bir sabit-noktalı sayıyı ele alalım.

3.625 ve 6.5 gibi iki sayıyı toplayıp, sonucunu da bastırmak istediğimizde ne olacağına bakalım. Bir giriş programı, önce bu sayıları ikili biçime dönüştürür.

$$\begin{array}{l} 3.625 \rightarrow 11.101 \rightarrow 0011.1010 \text{ (8 bit)} \\ 6.5 \rightarrow 110.1 \rightarrow 0110.1000 \text{ (8 bit)} \end{array}$$

Bu aşamada bilgisayar sayıları, sırasıyla 00111010 ve 01101000 olarak dikkate alacaktır. İkili noktaların yalnızca bir sembol olarak bulunduğunu hatırlayın. Bu sayılar,

$$\begin{array}{r} 00111010 \\ + 01101000 \\ \hline 10100010 = 162 \text{ (eğer işaretli ikili değer olarak düşünülürse)} \end{array}$$

değerini vermek üzere normal yoldan toplanabilirler.

Çıkış programı sonucu alıp bunu, tamsayı kısmı (1010) ve ondalık sayı kısmı (.0010) şeklinde ikiye böler ve doğru cevap olan 10.125'i basar. Sabit noktalı bir

sayının tek bir sözcük tarafından mümkün olandan daha büyük değer aralıklarına ulaşmak üzere birkaç sözcük genişliğine yayılabileceğine dikkat edin.

Kesirli sayıların sabit-noktalı gösterilmesi bazı şartlarda, özellikle muhasebe hesaplamalarında, çok yararlıdır. Buralarda en küçük kesirli kısım (sözgelimi)  $\frac{1}{10}$  şun 0.1'i veya liranın 0.001'i olabilir. En büyük tamsayı kısmı (sözgelimi) 1.000.000 lira olabilir. Böyle bir miktarı (sözgelimi) BCD şeklinde göstermek için toplam  $6 \times 4 + 3 \times 4 = 36$  bit gerekir. Bayt-tabanlı bir bilgisayarda her bir sayı için 4 bayt gerekecektir.

Sabit-noktalı sayıların da sınırları vardır. Ya güneşin davranışını inceleyen ast fizikçi? O da,

[1990000000000000000000000000000000 gram]

gibi güneşin ağırlığı veya

[0.00000000000000000000000000000000910956 gram]

gibi bir elektronun ağırlığı gibi büyüklüklerle uğraşmaktadır.

Eğer astrofizikçimiz sabit-noktalı aritmetiğe başvurmak zorunda kalsaydı, sayıları göstermek için aşırı derece büyük miktarda bir grup bayta ihtiyaç duyacaktı. Tek bir bayt 0-255 aralığındaki, ya da yaklaşık olarak 0-1/4 bin aralığındaki sayıları gösterir. Böylece fizikçimiz kabaca tamsayı kısmı için 14 bayta, kesirli kısım için de 12 bayta gereksinim duyacaktı, yani -26 baytlık (208 bit) bir sayı! Bu ikilemden bizi kurtaracak yöntem için ipucu, her iki sayının da büyük miktarda bir grup sıfır ve çok az değerlikli basamaklar içeriyor olduğuna dikkat etmektir.

### 3.7.1 Kayan-Noktalı Sayıların Gösterilmesi

Sayısal bilgisayarlar, sayıları genelde kayan-noktalı bir formatta gösterip saklarlar. 1234.56 sayısını  $0.123456 \times 10^4$  ile gösterdiğimiz gibi bilgisayar da ikili sayıları benzer bir biçimde ele alır. Örneğin 1101101.1101101 sayısı sistem içinde  $0.11011011101101 \times 2^7$  şeklinde gösterilebilir (elbette 7'de ikili formatta saklanır). Kayan noktalı notasyona kimi zaman bilimsel notasyon denir. Kayan noktalı sayılara daha ayrıntılı olarak bakmadan önce sayıların kayan-noktalı formatta gösterilmesi biçimiyle yakından ilişkili olan aralık, duyarlık ve kesinlik kavramlarını ele almakta yarar vardır.

*Aralık*

Bir sayının aralığı bize onun ne kadar büyük veya ne kadar küçük olabileceğini söyler. Astrofizikçi örneğinde  $2 \times 10^{33}$  kadar büyük, sayılardan  $9 \times 10^{-23}$  kadar küçük sayıları ele alıyorduk. Bu da yaklaşık olarak  $10^{61}$  veya 61 basamaklık bir aralığı gösterir. Sayısal bir bilgisayarda gösterilen sayıların aralığı, yapılabilecek tüm olası hesaplamaların büyük çoğunluğu için yeterli olmalıdır. Eğer bilgisayar ele alınacak veri aralığının oldukça az olduğunun bilindiği özel bir uygulamada kullanılacaksa, geçerli sayıların aralığı sınırlandırılarak donanım/yazılım gerekleri basitleştirilebilir.

## Duyarlık

Bir sayının duyarlılığı, tamlığının bir ölçüsüdür ve onu göstermek için kullanılan sembol sayısına karşılık gelir. Örneğin sabiti 3.142 olarak veya 3.141592 şeklinde yazılabilir. İkinci durum ilkinden daha duyarlılıktır, çünkü birincisi  $\pi$ 'yi  $10^4$ 'te bir olarak gösterirken bu,  $10^7$ 'de bir olarak gösterir.

### *Kesinlik*

Kesinlik buraya, büyük ölçüde, çoğu kez duyarlılıkla aynı anlama geldiği şeklindeki yanlış düşünüşe bir tezat oluşturmak üzere eklenmiştir. Kesinlik, bir değerin kesin olma ölçüsüdür. Örneğin  $\pi = 3.141$  veya  $\pi = 3.141592$  diyebiliriz. Şimdi görebiliriz ki birinci durumda daha az duyarlıklı, ancak yanındaki daha büyük duyarlıktaki sayıya göre daha kesin bir sayı yer almaktadır. İdeal bir dünyada duyarlık ve kesinlik el ele gidecektir. Mevcut duyarlığın izin verdiği ölçüde kesinliği koruyan sayısal algoritmalar tasarlamak bilgisayar programcısına bağlıdır. Bilgisaymanın potansiyel kara-noktalarından biri şu şekildeki hesaplamalardır:

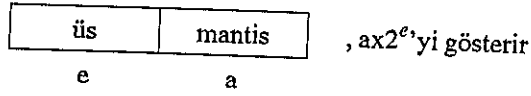
$$\frac{A+B}{A-B} \text{ örneğinin } \frac{1234.5687 + 1234.5678}{1234.5687 - 1234.5678}$$

ifadenin paydası hesaplandığında, 0.0009 kalır, ki bu sadece bir ondalık basamak duyarlığında olan bir sayıdır. Hesaplama yapıldığında sonuç sekiz basamaklı bir duyarlık göstermekle birlikte aslında kesinliği çok düşük olabilir.

Kayan noktalı bir sayı şu biçimde gösterilir:

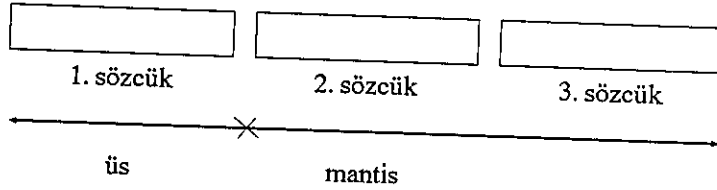
$a \times r^e$

Burada  $a$  mantis (aynı zamanda argüman da denir),  $r$  kök ve  $e$  de üs karakteristiktir. Bilgisayarlar, kayan noktalı sayıları, onların ikili gösterilmesi alana bölerek saklarlar:



$r$  kökü bilinmektedir ve bilgisayar tarafından ayrıca saklanmasına gerek yoktur. Kısımın kalan tarafındaki tüm kayan-noktalı sayılarda kökün değerinin iki oldu varsayılmıştır. Bazı bilgisayarlarda üssün kökü sekizli veya onaltılıdır ve böyle mantis sırasıyla  $8^e$  veya  $16^e$  ile çarpılır.

Bir kayan-noktalı sayının tek bir saklama konumu işgal etmesi gerekli değildir. Aslında sekiz-bitlik bir sözcükte bu tür bir gösterilme yetersiz olacaktır. Çoğu zaman kayan-noktalı sayıyı oluşturmak için bir grup sözcük biraraya getirilir. Üs ile mantis arasındaki bölmenin bir sözcük sınırına düşmesinin gerekmediği belirtilmekte yarar vardır.



### 3.7.2 Kayan-Noktalı Sayıların Normalize edilmesi

Kural gereği kayan-noktalı mantis kısmı, (eğer sıfıra eşit değilse) daima normalize edilir. Böylece  $0.1...x2^e$  biçiminde ifade edilir. Şu an için sadece pozitif mantisleri inceleyeceğiz. Eğer bir hesaplamamız sonucu  $0.01...x2^e$ 'yi verecekse sonuç  $0.1...x2^{e-1}$  olarak normalize edilir. Mantisini normalize ederek mevcut duyarlıktan maksimum şekilde yararlanılır. Burada mühendisler ve bilimadamları tarafından kullanıldığı şekliyle normalize edilmiş ondalık sayılarla normalize edilmiş ikili sayılar arasında ufak bir fark olduğunu belirtmekte fayda vardır. Kayan-noktalı ondalık bir sayı normalize edilerek mantisi  $1.00...0$  ile  $9.99...9$  arasında yer alır. Pozitif kayan noktalı normalize edilmiş bir ikili mantis şu biçimdedir:

$$0.100...0 \text{ ile } 0.11...11$$

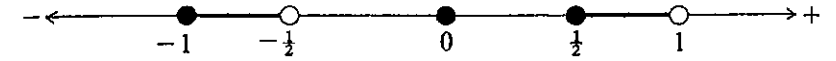
Eğer kayan-noktalı mantis  $x$  ise,  $\frac{1}{2} \leq x < 1$ 'dir. Özel bir istisna, sıfır durumunda

yapılmalıdır. Çünkü doğal olarak bu sayı normalize edilemez. Negatif, ikiye tümleyen, kayan-noktalı bir mantis ise şu biçimde saklanır:

$$1.01...1 \text{ ile } 1.00...0 \text{ arasında}$$

Bu durumda, negatif mantis  $x$ , sınırlandırılmıştır; böylece  $-\frac{1}{2} > x \geq -1$ .

$$-1 \leq x < \frac{1}{2} \quad \text{veya} \quad x = 0 \quad \text{veya} \quad \frac{1}{2} \leq x < 1.$$



Sayıların kayan noktalı gösterilmesinde pozitif ve negatif sayılar ile pozitif negatif üsler için önlem alınmalıdır. Örneğin ondalık notasyonda bu, şuna karşılık gelir:

$$\begin{array}{ll} + 0.123 \times 10^{12}, & - 0.756 \times 10^9 \\ + 0.176 \times 10^{-3}, & - 0.459 \times 10^{-7} \end{array}$$

Kayan-noktalı bir sayının mantisi çoğu zaman ikiye tümleyen biçiminde bir sayı olarak gösterilir. Öte yandan üs, kimi zaman SAPMALI bir biçimde gösterimlenir. Eğer  $m$ -bitlik bir üssü örnek alırsak,  $00...0$ 'dan  $11...1$ 'e kadar  $2^m$  olası işaretli tam-sayı değeri vardır. Diyelim ki bu sayıları  $0$ 'dan  $2^{m-1}$ 'e kadar değil de,  $-2^{m-1}$ 'den  $+2^{m-1}-1$ 'e kadar etiketledik. Bu iş, sayıların her birinden  $2^{m-1}$  gibi sabit bir değerin çıkarılmasıyla yapılır. Bir başka deyişle, SAPMALI üs,  $b'$ ,  $b' = b + 2m - 1$  şeklinde tanımlanır. Burada  $b$  gerçek üstür. SAPMALI  $b'$  üssü, üssün dahili gösterilmesidir.  $m = 4$  olduğunda ne olacağını ele alalım (Tablo 3.14). Gerçek üs  $-8$  ile  $+7$  aralığında bulunarak bize ikinin katlarını  $2^{-8}$ 'den  $2^{+7}$ 'ye dek gösterilme imkanı sağlar. Bu notasyonun avantajı en negatif üssün sıfır ile gösterilmesidir. Kural olarak, sıfırın kayan-noktalı değeri  $0.0...0x2^1$  ile gösterimlenir. Burada  $1$  en negatif üstür. SAPMALI üs sistemini seçerek sıfırın bir sıfır mantis ve sıfır üs ile gösterilmesini düzenleriz:

$$0.0 = \begin{array}{|c|c|} \hline 0...0 & 0...0 \\ \hline e & m \\ \hline \end{array}$$

Bu gösterilmenin ikinci avantajı üslerin, ikili biçimlerinde monoton olmasıdır. Yani; üssü bir arttırmak, ikili üsse bir eklemek ve üssü bir azaltmak da, ikili üsten bir çıkarmak demektir. Her iki durumda da ikili üs işaretli bir ikili sayı olarak davranıyormuş gibi ele alınır.

Tablo 3.14 Sapmalı üs

| Üssün ikili gösterimi | Doğru üs | Sapmalı biçim |
|-----------------------|----------|---------------|
| 0000                  | -8       | 0             |
| 0001                  | -7       | 1             |
| 0010                  | -6       | 2             |
| 0011                  | -5       | 3             |
| 0100                  | -4       | 4             |
| 0101                  | -3       | 5             |
| 0110                  | -2       | 6             |
| 0111                  | -1       | 7             |
| 1000                  | 0        | 8             |
| 1001                  | 1        | 9             |
| 1010                  | 2        | 10            |
| 1011                  | 3        | 11            |
| 1100                  | 4        | 12            |
| 1101                  | 5        | 13            |
| 1110                  | 6        | 14            |
| 1111                  | 7        | 15            |

### 3.7.3 Tipik Bazı Kayan-Noktalı Sistemler

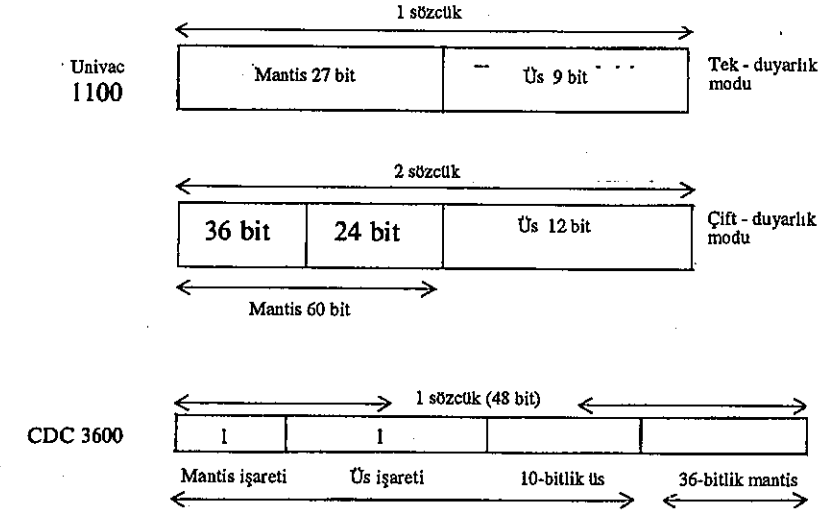
Verilen bir bilgisayar için kayan-noktalı bir gösterimi kullanmak üzere program şunları seçmelidir:

1. Kullanılan sözcük sayısı (yani toplam bit sayısı)
2. Mantisin gösterilmesi (2'ye tümleyen vs.)
3. Üssün gösterilmesi (sapmalı vs.)
4. Mantis ve üsse ayrılan bit sayısı
5. Mantisin konumu (önce üs veya önce mantis)

4. nokta özenle incelenmelidir. Programcı bir kez kayan-noktalı gösterim toplam bit sayısı (bütünleşik bir grup sözcük uzunlukları) hakkında karar verdiğinde, sonra bu gösterimdeki mantis ve üsse bölüştürmelidir. Eğer üs için büyük sayı bit ayırırsa sonuçta ortaya, çok büyük bir aralığı olan kayan-noktalı sayı çıkar. (üs) bitler mantisin zararına elde edilir ki bu da kayan-noktalı sayının duyarlılığı azaltır. Bunun tersi olarak, mantis için mevcut bitleri arttırmak, aralığın zarar duyarlılığı düzeltecektir.

Yukarıda verilen beş noktadan dolayı, kayan-noktalı bir sayının gösterilebileceği yol sayısı o kadar çoktur ki, (neredeyse) aynı formatı kullanan makina sayısı bile değildir. İşler mikroişlemcilerin kullanımı ile daha iyiye gidiyor olabilir.

daha sonra göreceğimiz gibi IEEE kayan-noktalı sayılar için önerilen bir formatın teknik özelliklerini taslak halinde üretmiştir. Aşağıdaki örnekler bazı bilgisayarlar da bulunan gösterimi belirtmektedir.



Univac'taki tek duyarlık modunda aralık, yaklaşık olarak  $10^{-76}$  dan  $10^{76}$  ya, duyarlık ise noktadan sonra 8 basamağa kadardır. Çift duyarlık modunda aralık  $10^{-64}$  den  $10^{64}$  e kadar ve duyarlık da noktadan sonra 18 basamağa kadar çıkar. Çift duyarlık; kimi zaman çok büyük duyarlığın gerekli olduğu sayısal aritmetikte kullanılır. Genelde; çift duyarlık kullanımı, bilgisayarın özel yüksek-hızlı bir kayan-noktalı birimi yoksa, programların çalışmasını önemli ölçüde yavaşlatır.

### IEEE Kayan-Noktalı Format

Elektronik ve elektrik mühendisleri enstitüsü (IEEE) mini ve mikrobilgisayarlardaki aritmetik işlemler için standart bir kayan-noktalı format üretmiştir. Bir grup farklı uygulamanın ihtiyacını karşılamak için IEEE tek, çift ve dörtlü denen üç temel format belirlemiştir. Tablo 3.15 bu üç kayan-noktalı formatın temel özelliklerini tanımlamaktadır.

X gibi bir kayan-noktalı sayı, tek formatta, mantisi  $f$   $1 \leq f < 2$  aralığında olmak üzere normalize edilir. Bu aralık tamsayı kısmı bire eşit olan mantise karşılık gelir. X şöyle bir formülle tanımlanır:

$$X = (-1)^S \times 2^{E-127} \times (1 + F).$$

Burada;

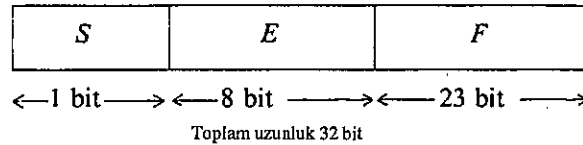
$S$ = işaret biti, 0= pozitif mantis 1= negatif mantis

$E$ = 127 ile sapmalı 8-bit üs

$F$ = 23-bit kesirsel mantis (1 başta ve örtük biçimde)

İlginç iki özel nokta vardır. İlki mantis için işaret ve değer gösterilmenin bir senmiş olmasıdır. Bu cebirsel olarak  $(1)^S$  ile gösterilir; burada  $S$  işaret bitidir.  $S=0$  ise  $(-1)^0$  olur ki bu +1'dir.  $S=1$  ise  $(-1)^1$  olur ve bu da -1'dir.

İkinci nokta ise mantisin daima normalize edilmesi ve 1.00...0 ile 1.11...1 aralığı bulunmasıdır. IEEE kayan-noktalı sayılarının daha önce karşılaştığımız kayan-noktalı sayılardan daha farklı şekilde normalize edildiğine dikkat ediniz.  $E$  mantis daima normalize edilirse baştaki 1, tamsayı kısmı, sayı bellekte saklandığında fazlalık olmuş olur. Eğer kesirli mantisin soluna bir 1 yerleştirilmesi gerektiğini biliyorsak, bunu saklamaya gerek yoktur. Bu şekilde bir bitlik saklama alanından tasarruf edilerek mantisin duyarlılığının bir bit genişletilmesi sağlanır. Bellekte saklanan sayının formatı aşağıda verilmiştir.



Bu formatın kullanımına bir örnek olarak sözcük uzunluğu 16-bit olan bir makine -2345.125 ondalık sayısının gösterilmesini ele alalım.

$$\begin{aligned} -2345.125_{10} &= -100100101001.001_2 \\ &= -1.00100101001001_2 \times 2^{11} \end{aligned}$$

Mantis negatiftir; böylece  $S$  işaret biti 1'dir.

Üs,  $+11+127=138=10001010_2$  ile verilir.

Kesirli kısım ise şöyledir: 001001010010010000000000 (23 bitlik)

Böylece, -2345.125'in IEEE tek duyarlık gösterilmesi şöyledir:

11000101000100101001001000000000.

Bu sayı ardışık iki 16-bit sözcük olarak saklanır.

1100010100010010 1001001000000000  
↑ E F  
S

Saklama alanını en aza indirmek için kayan-noktalı sayılar paketlenir ve böylece bilgisayar 32-bit veya daha uzun bir sözcük uzunluğu olmadığı sürece işaret biti, üs ve mantis iki veya daha çok makine sözcüğünün kısımlarını paylaşır. Kayan-noktalı işlemler yapılırken önce sayılar açılır ve böylece mantis üsten ayrılmış olur. Örneğin temel tek duyarlık formatı, 23-bitlik bir kesirli mantis belirleyerek 24-bitlik bir mantis verir. Eğer kayan-noktalı sayıların işlendiği işlemcinin sözcük-uzunluğu 16-bit ise açılmış mantis iki sözcükten alınma 32 bitten 24 bitini işgal edecektir.

Eğer, bir sayı açıldığında, üs ve mantisindeki bitlerin mevcut alanı doldurmasına izin verilirse bu formatın genişletilmiş olduğu söylenir. Formatı bu şekilde genişleterek kayan-noktalı sayının aralığı ve duyarlılığı önemli ölçüde artar. Yukarıdaki örneğe göre, 24-bitlik mantis iki makine sözcüğü uzunluğundadır (yeni 32 bit) ve tüm kayan-noktalı hesaplamalar 32-bitlik bu mantis üzerinde yapılır. Bu özellik özel fonksiyonlar (yani  $\sin x$ ,  $\cos x$ ) hesaplarken oldukça yardımcı olur. Genişletilmiş formatta bir dizi kayan işlemler yapıldıktan sonra, kayan noktalı sayı tekrar paketlenir ve bellekte temel biçiminde saklanır.

### 3.7.4 Kayan-Noktalı Aritmetik

Tamsayı ve sabit-noktalı sayı gösterilmesinden, farklı olarak, kayan noktalı sayılar tek bir basit işlemde toplanamazlar. Bir an düşününce, bunun neden böyle olduğu anlaşılacaktır. Ondalık aritmetikteki bir örneği ele alalım. Aşağıdaki örneklerde kullanılan kayan-noktalı sistem bu kısımda daha önce anlatılan sistem olacaktır--saptırılmış üs ile ikinin tümleyeni mantis.  $A=12345$  ve  $B=567.89$  olsun. Kayan-noktalı biçimde bu sayılar şu şekilde gösterilir.

$$A=0.12345 \times 10^5 \text{ ve } B=0.56789 \times 10^3$$

Bu sayılar elle toplansaydı sorun doğmayacaktı.

Tablo 3.15 Kayan noktalı sayılara ilişkin temel IEEE formatı

| Tip                      | Tek      | Çift     | Dörtlü   |
|--------------------------|----------|----------|----------|
| Bit olarak alan uzunluğu |          |          |          |
| S- işaret                | 1        | 1        | 1        |
| E- üs                    | 8        | 11       | 15       |
| L- baştaki bit           | 1        | 1        | 1        |
| F- kesir                 | 23       | 52       | 111      |
| Toplam uzunluk           | 32       | 64       | 128      |
| İşaret biti              | 0=+, 1=- | 0=+, 1=- | 0=+, 1=- |
| Üs                       |          |          |          |
| Maksimum E               | 255      | 2047     | 32767    |
| Minimum E                | 0        | 0        | 0        |
| Sapma                    | 127      | 1023     | 16383    |

Normalize sayılar: (tüm formatlar için)  
 Üs aralığı (Min E+1) (Maks E-1)  
 Gösterilen sayı  $(-1)^S \times 2^{E-1} \times (L+F)$

İşaretili bir sıfır, minimum üs, L=0 ve F=0 ile gösterilir (tüm formatlar için geçerli)

Maksimum üssün özel bir fonksiyonu vardır ve üç formatın hepsi için de işaretili sonsuzu göstermek için kullanılır.

$$\begin{array}{r} 12345 \quad A \\ + \quad 567.89 \quad B \\ \hline 12912.89 \end{array}$$

Öte yandan bu sayılar normalize edilmiş bir kayan-noktalı formatta tutuldukları için aşağıdaki gibi bir problem vardır:

$$\begin{array}{r} 0.12345 \times 10^5 \\ + \quad 0.56789 \times 10^3 \end{array}$$

Bu toplama üsler farklı olduğu sürece yerine getirilemez. Kayan-noktalı bir toplama (veya çıkarma) yapmak için aşağıdaki adımlar uygulanmalıdır:

1. Daha küçük üslü sayıyı belirle.
2. Küçük sayının mantisini üssünün arttırıldığı orana bölerek küçük üssü büyük üsse eşitle.
3. Mantisleri topla (veya çıkar).
4. Gerekliyse, sonucu normalize et (sondan-normalizasyon).

Yukarıdaki örnekte elimizde  $A = 0.12345 \times 10^5$  ve  $B = 0.56789 \times 10^3$  vardır. B'nin üssü A'ninkinden küçüktür ve bu da B'nin  $0.0056789 \times 10^5$  olması ile sonuçlanır. Şimdi A ile B'yi toplayabiliriz.

$$\begin{array}{r} A = 0.1234500 \times 10^5 \\ + B = 0.0056789 \times 10^5 \\ \hline 0.1291289 \times 10^5 \end{array}$$

Buradaki durumda sonucun, sondan normalizasyona ihtiyacı yoktur. A ve B'nin duyarlığı beş basamaklı olduğu halde sonucun duyarlığının yedi basamaklı olduğuna dikkat edin.

İnsanlar aritmetik yaparken sık sık 'kayan duyarlık' diye adlandırılacak şeye başvururlar; daha büyük duyarlık isterlerse, basitçe daha çok basamak kullanırlar. Bilgisayarlar sabit bir gösterilme kullanırlar ve böylece duyarlık hesaplamının sonucu olarak artmaz. Aşağıdaki ikili örneği ele alalım.

$$\begin{array}{r} A = 0.11001 \times 2^4 \\ B = 0.10001 \times 2^3 \end{array}$$

B'nin üssü 1 arttırılmalı ve mantisi 2'ye bölünmelidir (yani bir basamak sağa kaydırılmalıdır).

$$\begin{array}{r} A = 0.11001 \times 2^4 \\ B = \frac{0.010001 \times 2^4}{1.000011 \times 2^4} \end{array}$$

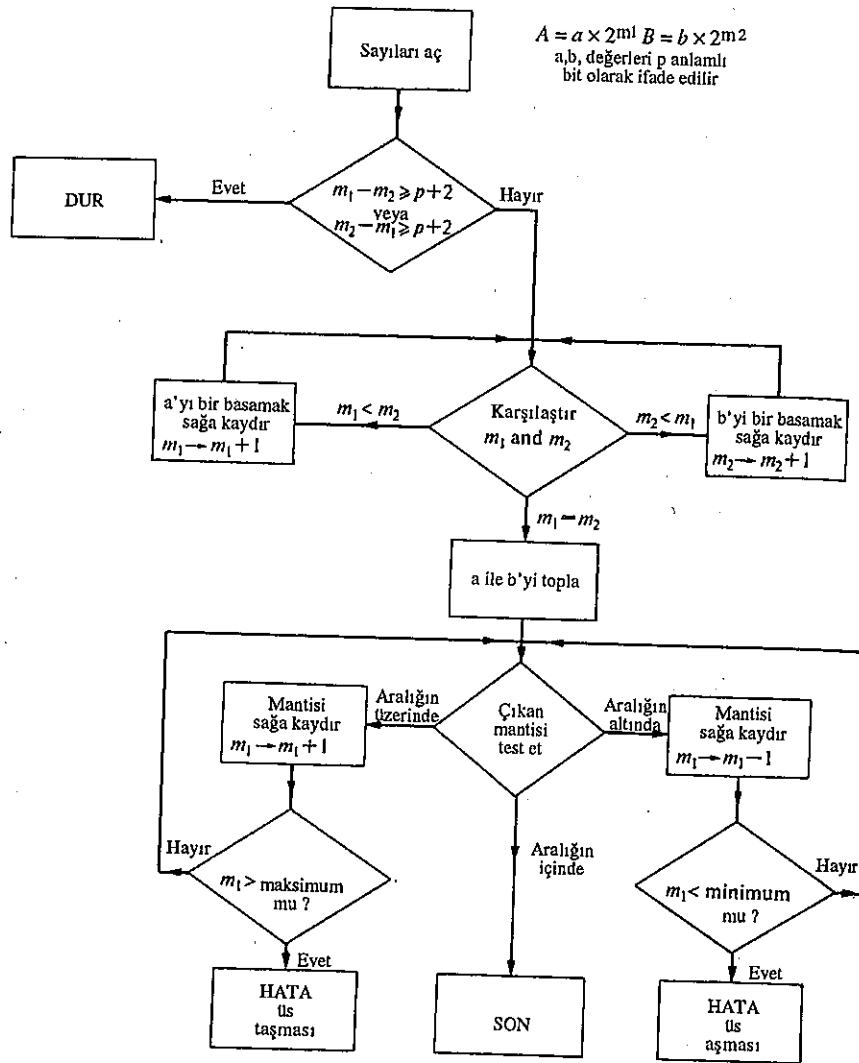
Sonuç taşma göstermiştir ve mantisi ikiye bölüp üssü arttırarak sondan-normalize edilmelidir.

$$A + B = 1.000011 \times 2^4 = 0.1000011 \times 2^5$$

Aynı zamanda duyarlıklı fazladan iki basamağımız oldu. Bu da bizi, birşeyler yapmaya zorlar. Örneğin basitçe sayıyı yuvarlayarak,

$$A + B = 0.10000 \times 2^5$$

elde edebiliriz. Kayan noktalı sayıların toplanmasıyla ilgili daha formül niteliğinde bir yöntemin akış diyagramı Şekil 3.18'de verilmiştir.



Şekil 3.18 Kayan-noktalı toplama işlemine ilişkin akış diyagramı

Bu akış diyagramı ile ilgili dikkat edilecek birkaç nokta şöyledir:

1. (Bir çok uygulamada) üs, mantis ile bir sözcüğü paylaştığından toplama işlemi başlamadan önce, bunları ayırmak gerekir. Buna 'açma' denir.
2. Eğer iki üs arasındaki fark  $p+1$ 'den büyükse (burada p mantisteki değerlikli bitlerin sayısıdır), küçük sayı büyüğü etkileyemeyecek kadar küçüktür ve dolayısıyla da sonuç etkin olarak büyük sayıya eşittir ve başka işlem yapılmaz.

sayıyla da sonuç etkin olarak büyük sayıya eşittir ve başka işlem yapılmaz.

3. Sondan normalizasyon sırasında, üsün olası en küçük değerinden küçük ve olası en büyük değerinden büyük olup olmadığını görmek için kontrol edilir. Bu sırasıyla üs alt-taşmasını ve taşmasını test etmeye karşılık gelir. Bu durumların her birinde sayı bilgisayarın işlem yapabileceği aralığın dışındadır. Üs alt-taşması genellikle sayının sıfıra eşitlenmesine yol açarken; üs taşması, bir hata koşulu sonucunu verir ve işletim sisteminin araya girmesini gerektirir.

### Yuvarlama ve Kesme

Kayan-noktalı aritmetiğin kapsadığı bazı işlemlerin mantisteki bit sayısının artmasına yol açtığını ve mantisteki bit sayısını sabit tutmak için bazı tekniklerin yardımı çağırıldığını görmüştük. En basit teknik kesmedir ve istenmeyen bitlerin atılmasından başka birşeyi içermez. Örneğin, 0.1101101 4 ağırlıklı bite kesilir ve 0.1101 olur. Daha iyi bir teknik, yuvarlamadır. Kaybolan basamakların değeri kalan basamakların en küçük değerlikli bitlerinin yarısından büyükse kalan basamakların EKDB'ine 1 eklenir. Örneğin aşağıdaki sayıları 4 değerlikli bite yuvarlamayı ele alalım.

$$0.1101101 \rightarrow 0.1101 + 0.0001 = 0.1110 \text{ (yukarı yuvarlama)}$$

$$0.1101011 \rightarrow 0.1101 \text{ (aşağı yuvarlama)}$$

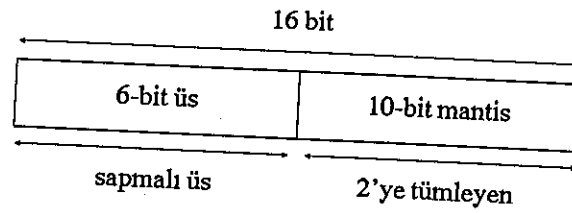
Yuvarlama, daima kesmeye tercih edilir. Çünkü kısmen daha doğrudur, kısmen de sapmasız bir hataya neden olur. Kesme daima sonucun değerini azaltarak sistematik bir hataya yol açarken kaydırma kimi zaman sonucu azaltır kimi zaman da artırır. Yuvarlamanın en büyük dezavantajı sonuç üzerinde yerine getirilecek fazladan bir aritmetik işlemine gereksinim duymasıdır.

### 3.7.5 Bazı Çözülmüş Örnekler

Kayan-noktalı aritmetiğin karmaşıklığından ve iki sayının toplanmasında veya çıkarılmasında dikkate değer ölçüde çok miktarda adımın işe katılmasından dolayı, aşağıdaki tümüyle çözülmüş örnekler buraya eklenmiştir.

#### Soru 1

Aşağıdaki kayan-noktalı aritmetik formatını kullanarak 10.125 ve 32.1'i toplayınız. Her durumda sayıların bilgisayarda nasıl saklandığını gösteriniz.

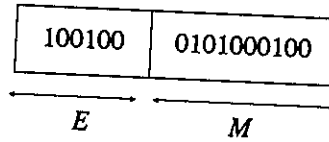


Çözüm 1

$$10.125 \rightarrow 1010.001_2$$

$$\rightarrow 0.101000100_2 \times 2^4$$

Üs 4'tür veya sapmalı biçimiyle  $4 + 32 = 36 = 100100$ 'dür. 10.125'in kayan-nokta biçimde gösterilmesi şöyledir:

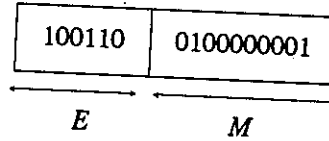


$$32.1 \rightarrow 100000.00011001100 \dots_2$$

$$\rightarrow 0.100000000110011_2 \times 2^6$$

$$\rightarrow 0.100000001_2 \times 2^6 \text{ yuvarlamadan sonra}$$

Üs 6'dır veya sapmalı biçimde,  $6 + 32 = 38$ 'dir. 32.1'nin kayan-nokta biçimde gösterilmesi şöyledir:

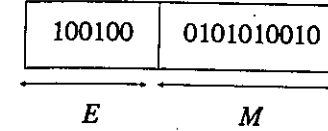


Bu sayıların üsleri olduğundan küçük mantis büyüklük sırasına konulmalıdır.

$$\begin{array}{ccccc} 100100 & 0.101000100 & \rightarrow & 100110 & 0.00101000100 \\ E & M & & E & M \end{array}$$

$$\begin{array}{r} \text{Şimdi mantisleri toplayabiliriz:} \\ 0.00101000100 \\ + 0.100000001 \\ \hline 0.101010010000 \end{array}$$

Bu sonuç şimdi 10 bite (0.101010010) yuvarlanabilir ve sonuçta elde edilen değer şu olur:



Bu ikili sayının ondalık değeri  $0.101010010 \times 2^6 = 101010.010 = 42.25$  ile gösterimlenir. Doğru cevabın 42.225 olduğuna dikkat edin. Hata ortaya çıkmıştır çünkü mantis 10 bit ile sınırlandırılmıştır.

Soru 2

6-bit sapmalı üs ve 10-bit 2'ye tümleyen mantis olduğunu varsayarak, aşağıdaki işlemleri yapınız. Her durumda da hesaplanan sonucu, doğru sonuç ile karşılaştırınız. Hesaplama sırasında mantisin 10-bitlik yuvarlaması kullanılabilir. Ardışık kayan nokta işlemleri boyunca (yani mantisi büyüklük sırasına koyarken) kesme kullanılmalıdır.

$$\begin{array}{llll} \text{(a)} & 157.3 & \text{(b)} & 157.3 \\ & + 257.1 & & - 12.6 \\ & \hline & & & \hline \end{array}$$

$$\begin{array}{llll} \text{(c)} & 157.3 & \text{(d)} & 157.3 \\ & - 142.7 & & + 158.3 \\ & \hline & & & \hline \end{array}$$

Çözüm 2a

$$\begin{array}{r} 157.3 \\ + 257.1 \\ \hline 414.4 \end{array}$$

Adım 1: 157'yi ikiliye dönüştür

$$\begin{array}{lll} 157 \div 2 = 78 & R=1 \\ 78 \div 2 = 39 & R=0 \\ 39 \div 2 = 19 & R=1 \\ 19 \div 2 = 9 & R=1 \\ 9 \div 2 = 4 & R=1 \\ 4 \div 2 = 2 & R=0 \\ 2 \div 2 = 1 & R=0 \\ 1 \div 2 = 0 & R=1 \end{array}$$

Böylece  $157_{10} = 10011101_2$ 'dir.

Adım 2: 0.3'ü ikiliye dönüştürün.

$$\begin{aligned} 0.3 \times 2 &= 0.6 \\ 0.6 \times 2 &= 1.2 \\ 0.2 \times 2 &= 0.4 \\ 0.4 \times 2 &= 0.8 \\ 0.8 \times 2 &= 1.6 \end{aligned}$$

Adım 3: Sabit-noktalı ikili sayıyı normalize edin.

$$\begin{aligned} 157.3 &= 10011101.01001..._2 \\ &= 0.1001110101001_2 \times 2^8 \end{aligned}$$

Adım 4: Mantisi 10 bit'e yuvarla

$$\begin{aligned} &0.100111010 \underline{1001} \\ &\quad \text{kalan 10 bitin EKDB'sinin } \frac{1}{2} \text{'sinden büyük} \\ &0.100111011 \text{ yuvarlanmış mantistir.} \end{aligned}$$

Böylece, 157.3'ün kayan-noktalı biçimde gösterilmesi  $0.100111011 \times 2^8$  dir. 8 sekizdir (gerçek üs). Sapmalı biçimde  $2^5 (=32)$ 'yi buna ekleyerek 40'ı elde ederiz. Bu sayı gerçekte 1010000100111011 şeklinde saklanacaktır.

Adım 5: 257.1, 10 bite yuvarlanmış normalize edilmiş kayan-noktalı sayıya dönüştür.

$$257.1_{10} = 0.100000001_2 \times 2^9$$

Adım 6: Üsleri karşılaştır

$$\begin{aligned} &0.100111011 \times 2^8 \\ &0.100000001 \times 2^9 \end{aligned}$$

Adım 7: Küçük üssü büyük üsse eşitle ve mantisini, üsteki her bir arttırma için 2 kere sağa kaydır.

$$\begin{aligned} &0.0100111011 \times 2^9 \\ &0.100000001 \times 2^9 \end{aligned}$$

Adım 8: Mantisleri topla

$$\begin{array}{r} 0.010011101 \times 2^9 \\ 0.100000001 \times 2^9 \\ \hline 0.110011110 \times 2^9 \end{array} \quad \begin{array}{l} \text{küçük mantisin 10 bitte kesildiğine} \\ \text{dikkat edin.} \end{array}$$

Sonuç zaten normalize edilmiş haldedir ve daha başka işleme gerek yoktur. Bu  $0.110011110_2 \times 2^9 = 11001110_2 = 414_{10}$ 'a karşılık gelir ve tam sonuçla (414.4) arasında 0.4'lük bir fark vardır.

Çözüm 2b:

$$\begin{array}{r} 157.3 \\ - 12.6 \\ \hline 144.7 \end{array}$$

Adım 1: 157.3'ü ikiliye dönüştürün.

$$157.3_{10} = 0.100111011_2 \times 2^8 \quad (\text{normalize edilmiş ve yuvarlatılmış})$$

Adım 2: 12.6'yı ikiliye dönüştürün.

$$\begin{aligned} 12.6_{10} &= 1100.10011001100_2 \\ &= 0.110010011011 \times 2^4 \\ &= 0.110010011 \times 2^4 \quad (\text{kayan noktalı ve yuvarlatılmış ikili}) \end{aligned}$$

Adım 3: Mantisin 2'nin tümleyeni biçimini oluşturun.

$$\begin{aligned} 0.110010011 &\rightarrow 1.011100010 \quad (\text{bitleri ters çevir}) \\ &\rightarrow 1.011100011 \quad (1 \text{ ekle}) \end{aligned}$$

Adım 4: Üsleri eşitle

$$\begin{aligned} 1.001101101 \times 2^4 &\rightarrow 1.111100110(1101) \times 2^8 \\ &\rightarrow 1.111100110 \times 2^8 \end{aligned}$$

Not 1: Negatif sayıları sağa kaydırırken işaret biti yayılır.

Not 2: Dışarı kaydırılan bitler bırakılır.

Adım 5: Toplamayı yerine getir

$$\begin{array}{r} 0.100111011 \times 2^8 \\ 0.011100011 \times 2^8 \\ \hline 10.000011110 \times 2^8 \end{array}$$

Not: Elde-bit konumundaki en solda yer alan 1, ikinin tümleyeni biçiminde aritmetiğin sonucudur ve ihmal edilir.

$$\text{Sonuç} = 0.100100001 \times 2^8 = 10010000.1_2 = 144.5_{10}$$

Tam cevap 144.7'dir.

Çözüm 2c.

$$\begin{array}{r} 157.3 \\ - 142.7 \\ \hline 14.6 \end{array}$$

Adım 1: 157.3'ü ikiliye dönüştür

$$157.3_{10} = 0.100111011_2 \times 2^8 \text{ (normalize edilmiş ve yuvarlanmış)}$$

Adım 2: 142.7'yi ikiliye dönüştür

$$\begin{aligned} 142.7_{10} &= 10001110.1011001_2 \\ &= 0.100011101 \times 2^8 \text{ (normalize edilmiş ve yuvarlanmış)} \end{aligned}$$

Adım 3: Mantisin, ikiye tümleyeni al.

$$\begin{aligned} 0.100011101 &\rightarrow 1.011100010 \text{ (bitleri ters çevir)} \\ &\rightarrow 1.011100011 \text{ (1 ekle)} \end{aligned}$$

Adım 4: Toplamayı yerine getir

$$\begin{array}{r} 0.100111011 \times 2^8 \\ 1.011100011 \times 2^8 \\ \hline 10.000011110 \times 2^8 \end{array}$$

Cevap  $0.000011110 \times 2^8$ 'dir.

Adım 5: Sonucu normalize et.

$$\begin{aligned} 0.0000011110 \times 2^8 &\rightarrow 0.1111000000 \times 2^4 \\ &\rightarrow 15_{10}. \end{aligned}$$

Tüm cevap 14.6'dır.

Çözüm 2d

$$\begin{array}{r} 157.3 \\ - 158.3 \\ \hline -001.0 \end{array}$$

$$\begin{aligned} \text{Adım 1: } 157.3 &= 0.100111011 \times 2^8 \text{ (normalize edilmiş ve yuvarlanmış)} \\ \text{Adım 2: } 158.3 &= 10011110.01001 \text{ (sabit-noktalı ikili)} \\ &= 0.1001111001001 \times 2^8 \text{ (kayan-noktalı ikili)} \\ &= 0.100111101 \times 2^8 \text{ (kayan-noktalı, yuvarlatılmış ikili)} \end{aligned}$$

Adım 3: Mantisin, ikiye tümleyeni al.

$$\begin{aligned} 0.100111101 &\rightarrow 1.011000010 \text{ (bitleri ters çevir)} \\ &\rightarrow 1.011000011 \text{ (1 ekle)} \end{aligned}$$

Adım 4: Toplamayı yerine getir (üsler burada aynıdır)

$$\begin{array}{r} 0.100111011 \times 2^8 \\ 1.011000011 \times 2^8 \\ \hline 1.111111110 \times 2^8 \end{array}$$

Sonuç negatiftir (işaret biti= 1) fakat normalize edilmemiştir.

Adım 5: Sonucu normalize et.

$$\begin{aligned} 1.111111110 \times 2^8 &\rightarrow 1.000000000 \times 2^0 \\ &\rightarrow -1_{10}. \end{aligned}$$

Aritmetik sola kaydırma işleminin şöyle olduğuna dikkat ediniz:

$$\begin{aligned} 1.111111110 \times 2^8 &\rightarrow 1.111111100 \times 2^7 \\ &\rightarrow 1.111111000 \times 2^6 \\ \text{vb.} &\rightarrow 1.111110000 \times 2^5 \end{aligned}$$

Tam cevap - 1'dir.

## Problemler

## Sayı-tabanı dönüştürme

1. Aşağıdaki ondalık sayıları ikili eşdeğerlerine dönüştürünüz.

(a) 15 (b) 42 (c) 235 (d) 4090 (e) 40900

2. Aşağıdaki doğal ikili sayıları ondalık eşdeğerlerine dönüştürünüz.

(a) 110 (b) 1110110 (c) 110111 (d) 1111110111

3. Tablo 3.16'yı tamamlayınız.

Tablo 3.16

| Ondalık | İkili       | Sekizli | Onaltılı |
|---------|-------------|---------|----------|
| 37      |             |         |          |
| 73      |             |         |          |
|         | 10101010    |         |          |
|         | 11011011101 |         |          |
|         |             | 42      |          |
|         |             | 772     |          |
|         |             |         | 256      |
|         |             |         | ABC      |

4. Aşağıdaki beş-tabanlı sayıları 9-tabanlı eşdeğerlerine dönüştürünüz. (örneğin  $23_5 = 14_9$ )

(a) 24 (b) 144 (c) 1234 (d) 444

## Sayı-tabanı dönüştürme ve kesirli sayılar

1. Aşağıdaki ondalık sayıları ikili eşdeğerlerine dönüştürünüz. Cevabı ikili beş basamağa kadar hesaplayınız ve gereklikçe yukarıya veya aşağıya yuvarlayınız.

(a) 1.5 (b) 1.1 (c) 1/3 (d) 1024.0625 (e) 3.141592

2. Aşağıdaki ikili sayıları ondalık eşdeğerlerine dönüştürünüz.

(a) 1.1 (b) 0.0001 (c) 101.101  
(d) 11011.101010 (e) 111.111111 (f) 10.1111101

3. Tablo 3.17'yi tamamlayınız. Tüm değerleri kök noktasından sonra dört basamağa kadar hesaplayınız.

Tablo 3.17

| Ondalık | İkili        | Sekizli | Onaltılı |
|---------|--------------|---------|----------|
| 0.37    |              |         |          |
| 0.73    |              |         |          |
|         | 11011.011101 |         |          |
|         | 111.1011     |         |          |
|         |              | 0.70    |          |
|         |              | 1.101   |          |
|         |              |         | 2.56     |
|         |              |         | AB.C     |

4. Aşağıdaki ondalık kesirleri, ikili kesirlere dönüştüğünde doğacak hataları (hem mutlak hem de yüzde olarak) hesaplayınız ve noktadan sonra 5 ikili basamağa kadar düzeltiniz.

NOT: Ondalık sayıyı altı ikili digitli olarak dönüştürün ve eğer 6. bit 1 ise 5. biti yukarıya yuvarlayınız.

(a) 0.675 (b) 0.42 (c) 0.1975 (d) 0.1 (e) 0.01 (f) 0.001

5. Bir elektronik mühendisi üç durumu olan bir mantık elemanı icat etmiştir: -1,0,+1. Bu durumlar sırasıyla,  $\bar{1}$ , 0, 1 ile gösterilir. Bu düzenleme kökü 3 olan 'dengeli üç parçalı' bir sistemi oluşturmak için kullanılabilir; fakat burada basamaklar 0,1,2 yerine -1,0,+1'i gösterir. Tablo 3.18'deki örnekler, sistemin nasıl çalıştığını göstermektedir.

Tablo 3.18

| Üçlü | Dengeli üçlü | Ondalık |
|------|--------------|---------|
| 11   | 11           | 4       |
| 12   | 111          | 5       |
| 22   | 101          | 8       |
| 1012 | 1111         | 32      |

İlk 15 ondalıklı sayıyı dengeli-üç parçalı tabanında yazınız.

6. Bir deneyin sonuçları -4 ile +9 sayı aralığındadır. Bir bilimadamı sonuç bilgisiyara okuyup sonra da onları işlemek istemektedir. Olası girişlerin her birini göstermek için dört-bitlik ikili-kod kullanmaya karar verir. Sayıları -4 ile aralığında göstermeye yeterli 4-bitlik bir kod tasarlıyoruz.

7. Doğal-ikiliden Gray-koda bir dönüştürücü tasarlayınız. Devrenin 4 girişi, 4 çıkışı vardır. Örneğin 1000 girişi 1100 çıkışı verir. (Kısım 3.3.9'daki Tablo 3.3.9'a bakınız).

8. Geçerli kodların P= 0, Q= 10, R= 110 ve S= 111 olduğunu varsayarak aşağıdaki Huffman Kodunu çözünüz. P, Q, R ve S sırasıyla 00, 01, 10 ve 11 olan kodlansaydı kaç bite ihtiyaç duyulacaktı?

00000111011100000010111111101010001111100010

9. Neredeyse tüm bilgisayar donanım derslerinde, sayı tabanları ve tabanlar arasında da sayıların dönüştürülmesiyle ilgili bir bölüm vardır. Bilgisayarın sayıları gösterdiği tabanın bilgisayar kullanıcısı ve hatta bir bilgisayar bilimlari öğrenen için bir anlamı var mıdır?

### İkili Aritmetik

1. Aşağıdaki ikili aritmetik işlemini yapınız.

$$\begin{array}{rcl}
 \text{(a)} & 10110 & \text{(b)} \quad 100111 \\
 & + 101 & \quad 111001 \\
 & \hline & & \text{(c)} \quad 11011011 \\
 & & \quad 10111011 \\
 & & \quad 00101011 \\
 & & + 01111111 \\
 & & \hline
 \end{array}$$

2. Aşağıdaki sekizli toplamaları yapın.

$$\begin{array}{rclcl}
 \text{(a)} & 42 & \text{(b)} & 3357 & \text{(c)} & 777 & \text{(d)} & 437 \\
 & + 53 & & + 2741 & & 543 & & 426 \\
 & \hline & & & & & + 420 & & 772 \\
 & & & & & & & + 747 \\
 & & & & & & & \hline
 \end{array}$$

3. Aşağıdaki onaltılı toplamaları yapın.

$$\begin{array}{rclcl}
 \text{(a)} & 42 & \text{(b)} & 3357 & \text{(c)} & 777 & \text{(d)} & ABCD \\
 & + 53 & & + 2741 & & 543 & & FE10 \\
 & \hline & & & & & + 420 & & + 123A \\
 & & & & & & & \hline
 \end{array}$$

4. Sadece 8-bitlik aritmetiği kullanarak, aşağıdaki ondalık sayıları 2'nin tümleyeni ikili biçiminde ifade ediniz.

$$\begin{array}{l}
 \text{(a)} - 4 \quad \text{(b)} - 5 \quad \text{(c)} 0 \quad \text{(d)} - 25 \quad \text{(e)} - 42 \quad \text{(f)} - 128 \quad \text{(g)} - 127 \\
 \text{(h)} - 111
 \end{array}$$

5. Aşağıdaki ondalık çıkarmaları 8-bitlik 2'ye tümleyen aritmetiği ile yapınız. Bazı sonuçların aritmetik taşma ile sonuçlanacağına dikkat edin. Taşmanın nerelerde oluştuğunu belirtin.

$$\begin{array}{rclcl}
 \text{(a)} & 20 & \text{(b)} & 127 & \text{(c)} & 127 & \text{(d)} & 5 \\
 & - 5 & & + 126 & & - 128 & & - 20 \\
 & \hline & & & & & & & \\
 \text{(e)} & 69 & \text{(f)} & - 20 & \text{(g)} & - 127 & \text{(h)} & - 42 \\
 & - 42 & & - 111 & & - 2 & & + 69 \\
 & \hline & & & & & & + 120 \\
 & & & & & & & \hline
 \end{array}$$

6. 12-bitlik bir sözcükte 2'ye tümleyen biçiminde ikili aritmetiği kullanarak ve en küçük ve en büyük sayıları vererek gösterilebilen sayı aralığını yazınız (hem ondalık hem de ikili formatta). En küçük ve en büyük sayılar

(a) arttırıldığında (b) azaltıldığında  
ne olur?

7. n-bitlik sözcüklerle 2'ye tümleyen aritmetiğine taşma ve elde terimleri uygulandığında bu terimler arasındaki farkı belirtiniz.

8. N, ikinin tümleyeni biçiminde bir sayıyı gösterimlerken;  $N = a_{n-1}a_{n-2} \dots a_1a_0$  şeklinde n-bitlik tamsayının değerini veren cebirsel bir ifade yazınız.

Dolayısıyla  $n+1$  bitlik işaretli ikili bir sayının gösterimlenmesinin en solda tekrarlayarak  $n$ -bitlik gösterilmesinden türetililebileceğini ispat edin.

9. Aşağıdaki dört-bitlik ikili sayılar arasındaki toplamaları yapınız.

$$\begin{array}{rclcl} \text{(a)} & 0011 & \text{(b)} & 1111 & \text{(c)} & 0110 & \text{(d)} & 1100 \\ & + 1100 & & + 0001 & & + 0111 & & + 1010 \\ \hline & & & & & & & \end{array}$$

Her durumda da sayıları (i) işaretsiz tam sayı (ii) ikiye tümleyen biçiminde tam ve (iii) işaret-ve-değer şeklinde tam sayı olarak ele alın. Cevabı hesaplayın gereken yerlerde yorumlayın.

10. (a)  $n$ -basamaklı en büyük ondalıklı pozitif sayıyı yazın.

(b)  $m$ -bitlik en büyük ikili sayıyı yazın.

(c)  $n$ -basamaklı ondalıklı sayıları 2 tabanında göstermek gerekmektedir.

Olası tüm  $n$ -basamaklı ondalık sayıları göstermek için gereken en küçük bit sayısı  $m$  kaçtır? İpucu: en büyük  $m$ -bit ikili sayı en büyük  $n$ -bitlik ondalık sayıdan büyük veya ona eşit olmalıdır.

11. 4-bitlik ikili bir toplayıcı, iki tane işaretsiz 4-bitlik sayıyı,  $A$  ve  $B$ , toplayarak 4-bitlik bir  $S$  toplamı ile tek-bitlik bir  $C$  çıkış-eldesi üretmektedir. Toplayıcının üretebildiği çıkışların aralığı (yani en büyük ve en küçük değerleri) nedir? Cevabınızı hem ikili hem de ondalık biçimde veriniz.

Bir toplayıcı iki tane ikili kodlanmış ondalıklı (BCD) sayıyı toplayarak 4-basamaklı bir toplam ile bir-bitlik bir çıkış eldesi üretmek üzere tasarlanmıştır. Bu devrenin üretebileceği geçerli çıkışların aralığı nedir?

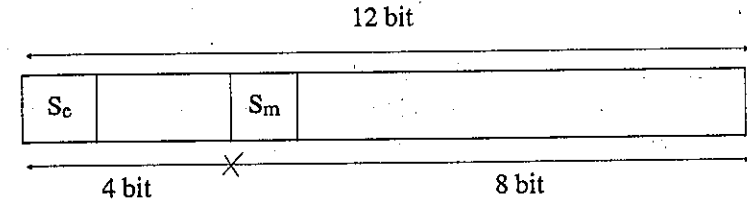
Bir BCD toplayıcısının tasarımcısı, iki BCD basamağını toplamak için sanki 4-bit saf ikili sayıların gibi saf bir ikili toplayıcı kullanmaya karar verir. İki kodlayıcı, hangi koşullar altında doğru BCD sonucunu verecektir? Hangi koşullar altında sonuç yanlıştır (yani 4-bit ikili sonuç istenen BCD sonucundan farklılık gösterir)?

Tasarımcı ikili toplayıcının 4-bitlik çıkışını bir BCD toplayıcısına dönüştürmek için hangi algoritmayı uygular?

12.  $X$  bitini  $B_1$  ödünç-girişi ile birlikte  $Y$  bitinden çıkararak  $D = Y - X - B_1$  fark bitini ve  $B_0$  ödünç-çıkışını üretecek bir tam çıkarıcı devresi tasarlayınız.

### Kayan-noktalı Aritmetik

1. Aşağıdaki kayan-nokta formatı size kayan-noktalı sayılara kolay bir giriş sunmak için tasarlanmıştır 2. ve 3. sorularda daha gerçekçi (ve daha sıkıcı) örnekler yer almaktadır.



Üs 3-bitlik mutlak değer artı  $S_c$  işaret-bitinden meydana gelir. Mantis ise 7-bitlik mutlak değer artı  $S_m$  işaret-bitinden oluşur. Örneğin 1101011010000 ikili deseni  $+0.11010000 \times 2^{-5}$ 'i gösterir.

Mantisin gösterilmesinin, ikiye tümleyen biçiminde değil işaret-ve-değer biçiminde olduğuna dikkat edin. Benzer şekilde, üs de daha sık kullanılan sapmalı biçimden çok işaret-ve-değer biçiminde gösterilir.

Yukarıdaki format için, gösterilebilen sayıların aralığını yazın. Bu formatı kullanarak aşağıdaki işlemleri yerine getirin.

$$\begin{array}{rclcl} \text{(a)} & 25 & \text{(b)} & 25 & \text{(c)} & 12.25 & \text{(d)} & 1.125 \\ & + 16 & & - 16 & & + 1.125 & & - 0.625 \\ \hline & & & & & & & \end{array}$$

2. Bir bilgisayarın, kayan-nokta işlemleri için 8-bitlik sapmalı üs ve 16-bitlik ikiye tümleyen biçiminde mantise bölünmüş 24-bitlik sözcük uzunluğu vardır. Bu formatta gösterilebilecek sayıların aralığını ve duyarlıklarını yazın.

3. 2. sorudaki formata göre, aşağıdaki işlemleri yapın.

$$\begin{array}{rclcl} \text{(a)} & 276.123 & \text{(b)} & 276.123 & \text{(c)} & -276.123 \\ & - 159.014 & & + 276.123 & & + 275.123 \\ \hline & & & & & \\ \text{(d)} & 1563.123 & \text{(e)} & 276.123 & \text{(f)} & 276.123 \\ & - 0.042 & & + 76.123 & & - 76.123 \\ \hline & & & & & \end{array}$$

Aşağıdaki varsayımları kullanın:

- (i) Sayılar 24-bitlik sözcükler olarak saklanır
  - (ii) Kayan-noktalı işlemler 18-bitlik mantislerle gerçekleştirilir (2 fazladan ALU'dan gelir). Yani bellekteki 16-bitlik mantis kayan-noktalı hesaplamalar ya ne getirilirken 18bitlik mantis olur.
  - (iii) Kayan-noktalı işlemler boyunca mantisin 19. basamağında üretilecek tüm bitler ortadan kaldırılır (yani kesme veya yuvarlama).
  - (iv) Sonuç değer, 18-bitlik mantis; 16-bite yuvarlanır, üs ile birlikte 24-bitlik sözcüğe paketlenir ve bellekte saklanır.
- Her bir durumda da sonuçtaki 24-bitlik kayan-noktalı sayıyı ondalıklı bir değere dönüştürün ve beklenen sonuç ile bunu karşılaştırın.

4. Normalize edilmiş kayan-noktalı  $X$  gibi bir sayının ikiye tümleyen biçiminde kesirli bölümü şu üç aralıktan birinde bulunacak şekilde sınırlanmıştır:

$$-1 \leq X < -\frac{1}{2}, \quad X = 0 \text{ veya} \quad \frac{1}{2} \leq X < 1.$$

Bunun neden böyle olduğunu açıklayın ve cevabınızı beş-bitlik mantis olarak belirtin.

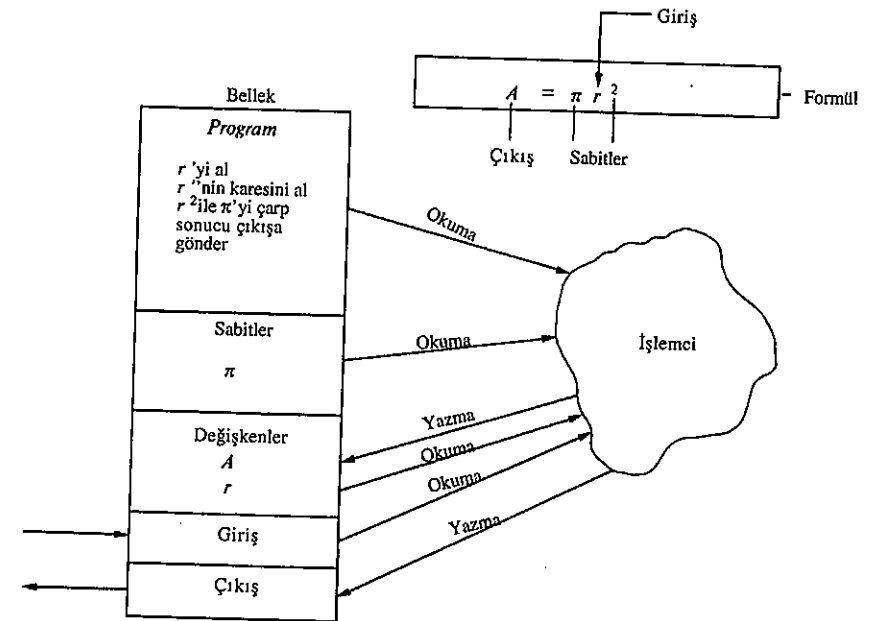
4

## MERKEZİ İŞLEM BİRİMİ

Merkezi işlem birimi (MİB)(Central Processing Unit-CPU), bir bilgisayar sisteminin kalbidir ve bir programın komutları boyunca sırasıyla ilerleyerek bu komutları yürütmekten ve bilgisayar belleği ve giriş/çıkış birimlerinin çalışmasını denetlemekten sorumludur.

MİB'nin nasıl çalıştığına göz atmadan önce, işlemci, bellek ve program arasındaki ilişkiyi anlamak önemlidir. Gelin, bir dairenin alanını hesaplayan basit bir programı ele alalım ve bilgisayarın buna ilişkin ne gibi işlemler yapacağını görelim. Aşağıdaki bilgisayar, gerçek dünyada karşılaşılan sorunların tümünden bağımsız hayali bir makinedir.

Bir dairenin alanının,  $\pi r^2$  ifadesine eşit olduğunu biliriz. İnsanlar bir dairenin alanını bulurlarken, gereken tüm adımları bilinçaltından otomatik olarak yaparlar. Ne var ki, sıra bir program yazmaya geldiğinde, bilgisayara ne yapması gerektiğini tam olarak adım adım anlatmak gerekir. Bunu göstermek üzere, ' $\pi \times r \times r$ ' ifadesine bir göz atalım. Burada  $r \times r$  yazmakla, aslında, ' $r$ ' sembolü ile verilen bir değer kendisiyle çarpılacağını anlatmak isteriz. Şek. 4.1'de, program, bellek ve işlemci arasındaki ilişki gösterilmiştir.



Şekil 4.1 Program, bellek ve işlemci arasındaki ilişkiyi gösteren diyagram.

Bellek; program, sabitler, değişkenler, giriş ve çıkış olmak üzere dört bölüme ayrılmıştır. Programlar, işleme koyulacak ya da diğer deyimle yürütülecek işin sırasını ifade eder. Sabitler (burada verilen örnekte tek bir sabit bulunmaktadır) program tarafından kullanılan ancak, yürütme süresince değişmeyen sayılardır. Örnekte  $\pi$  sayısı bir sabittir. Değişkenler, program tarafından yaratılan ya da değiştirilen sayıları gösterir. Dolayısıyla, program,  $r$ 'nin karesini alırken,  $r$  olarak adlandırılan değeri içeren bellek konumundaki sayının değerini okur ve bulduğu sonucu da geriye, aynı bellek konumuna yerleştirir. Değişkenler genelde sayı büyüklükler olsalar da, daima sayı olmaları zorunlu değildir. Örneğin, bir kelime işlemcinin kullandığı değişkenler, işlenen metnin içindeki harfler (veya diğer semboller) olmaktadır. Bir değişkenin bir başka program olması pekala mümkündür. Yani, bir program bir başka program üzerinde işlem, ya da değişiklik, yapabilir. Bu kavram bizi, kendi kendine programlama yapan (akıllı ?) bilgisayar düşüncesine götürecektir.

Herhangi bir program, mutlak surette dış dünya ile iletişim kurabilecek biçimde olmalıdır, aksi halde tüm çalışmalarının herhangi bir etkisi olmayacaktır. Giriş'ten okuma yapılması, bir giriş biriminden (sözelimi bir klavyeden) bilgi alınmasını ve çıkış'a yazılması da bilgilerin bilgisayardan dışarıya (sözelimi bir ekrana) aktarılmasını sağlayacak biçimde bellek konumlarından ikisini "giriş" ve "çıkış" olarak etiketledim. Giriş ve çıkışların birer bellek konumu olduğunu düşünürsek bunun hayali olmadığı görülür, kaldı ki, bazı bilgisayarlarda tüm giriş/çıkış işlemleri bellek üzerinden yapılmaktadır.

İşlemci, ya bir bellek konumundan veri okur ya da bir bellek konumuna veri yazar. Yukarıda tanımlanan beş bölgeden üçü, sadece-okunur, biri sadece-yazılır ve bir tanesi de hem okunur, hem de yazılabilir niteliktedir. İşlemciye geldiğimizde de, bellekteki bilgileri izlemek, MİB'nin en önemli işlevlerinden biri olduğunu göreceğiz.

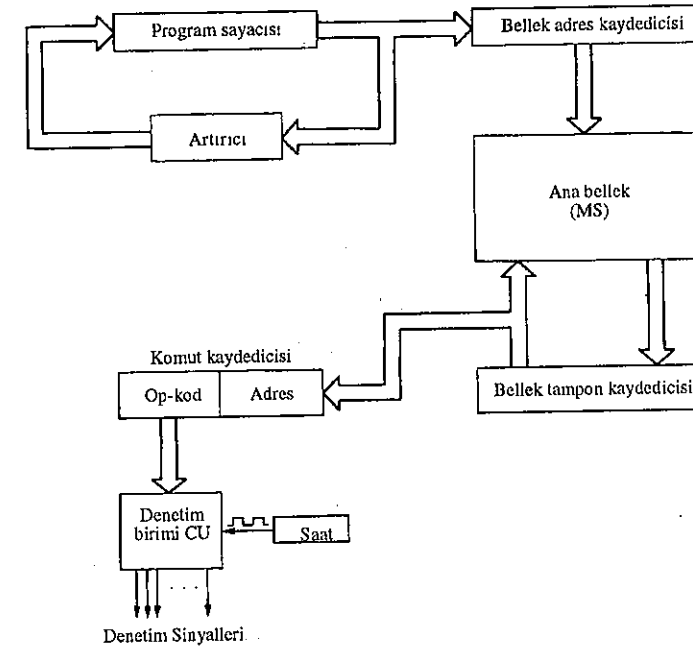
#### 4.1 MİB'nin yapısı

Şekil 4.2'de, MİB'nin bir bölümünün blok diyagramı verilmiştir. Bu diyagramda anlaşılır olması için, yalnızca adres yolları gösterilmiştir. Adres yolları, adreslerin, MİB'nin bir biriminden bir başka birimine taşındıkları anayollarıdır. Adresler, bellekteki bir veri parçasının konumunun gösterilmesidir. Bir bilgisayarın içinde, bir yerden bir yere taşınan iki tür bilgi vardır: adresler ve veriler (aslında adresler, aynı zamanda veridirler ancak burada veri denince komutlar, sabitler ve değişkenler anlatılmaktadır).

Şekil 4.2'yi açıklamak üzere, kaydedici-aktarım dili denilen bir kısaltma kullanılmıştır. Bir ya da daha fazla sayıda harf, veya harfler ve ardında sayılar şeklindeki ifadeler, kaydedici veya saklama konumlarını gösterirler. Köşeli parantezler, kaydedicilerin içermiş olduğu değerleri ve sola ok ( $\leftarrow$ ) ise, kaydedicinin içeriğinin aktarımını gösterir. Dolayısıyla, aşağıda verilen ifade,

$$[MAR] \leftarrow [PC]$$

program sayıcısının içeriğinin bellek adres kaydedicisine aktarıldığı (yani kopyalandığı) anlamına gelmektedir.



Şekil 4.2 MİB'nin adres yolları

Program sayıcısı (ya da komut sayıcısı) adı, bir bakıma yanlış verilmiş bir ad olarak gözükmemektedir. Bu sayıcı, ne programları ve ne de herhangi bir şeyi sayar, aslında yaptığı iş, bellekteki, yürütülecek bir sonraki komutun adresini içermektir. İşlevinin, yürütülecek bir sonraki komutu göstermek olduğu da söylenebilir. Bir komutun yürütülmesi, program sayıcısının içeriğinin bellek adres kaydedicisine taşınmasıyla başlar. Bunun ardından, ya da büyük olasılıkla aynı anda, program sayıcısının içeriği artırılır (1 artar) ve tekrar geriye program sayıcısına döndürülür. Bu yolla, yürütülükteki komut yürütülürken, program sayıcısı bir sonraki komutu göstermektedir.

Bellek adres kaydedicisi (MAR), verilerin bir yazma saykılında içine yazıldığı da okuma saykılında içinden okunduğu, ana bellekteki (MS) konumun adresi tutar. Bellek adres kaydedicisi, artık, PC'nin içeriğinin bir kopyasını içermektedir. Böylelikle bir okuma çevrimi gerçekleştirilirken, yürütülecek komut bellekte okunacak ve bellek tampon kaydedicisine (MBR) aktarılacaktır. MBR, bir okuma çevriminde bellekten alınan veriler veya bir yazma çevriminde belleğe aktarılacak olan veriler için geçici bir saklama yeridir. Bu aşamada MBR, yürütülecek komutun ikili değerini içerir.

Bu adımdan sonra komut, komut kaydedicisine (IR) taşınır ve burada iki alana ayrılırlar. Alan, içinde bitlerin mantıksal büyüklükler olarak gruplandıkları bir sözcük parçasıdır. Örneğin, kişi adları; Adı ve Soyadı olmak üzere iki alana ayrılmıştır. Alanların birinde işlem kodu (op-kod) ya da diğer ifadeyle, bilgisayarın ne yapacağını bildiren komut bulunur.

Diğer alanda ise, komut tarafından kullanılacak verilerin bulunduğu konumu gösteren adresi yer alır. İşlem kodlarının tümü bir bellek konumu göstermediğinden bazen bu ikinci alan fazlalık olmaktadır.

Denetim Birimi (CU), MİB'nin en karmaşık bölümüdür. Bu birim, saat darbesi dizisiyle birlikte Komut Kaydedicisinden işlem kodunu alır ve MİB'nin diğer tüm birimlerini denetleyen bir dizi sinyal üretir. Bir çok bilgisayarda, her bir saat darbesi arasındaki süre 0.01 'den 1 mikrosaniyeye kadar değişen değerdedir. Program sayıcısı içeriğinin Bellek Adres Kaydedicisine taşınması, okuma çevriminin yürütülmesi ve Bellek Adres Kaydedicisi içeriğini Komut Kaydedicisine taşınmasından bu denetim birimi sorumludur.

Yukarıda verilen işlem sırası, "getirme saykılı" olarak bilinir. Tüm komutlar, getir-yürüt adı verilen iki aşamalı bir işlemde yürütülürler. Getirme saykılı süresinde, komut bellekten okunur ve denetim birimi tarafından kodu çözülür. Getirme saykılı, yürütme çevrimi izler. Yürütme saykılı süresinde, komutu yürütmek için gerekli tüm sinyaller denetim birimi tarafından üretilir. Getirme saykılı, kaydedici-aktarım diline göre Tablo 4.1'de tanımlanmıştır.

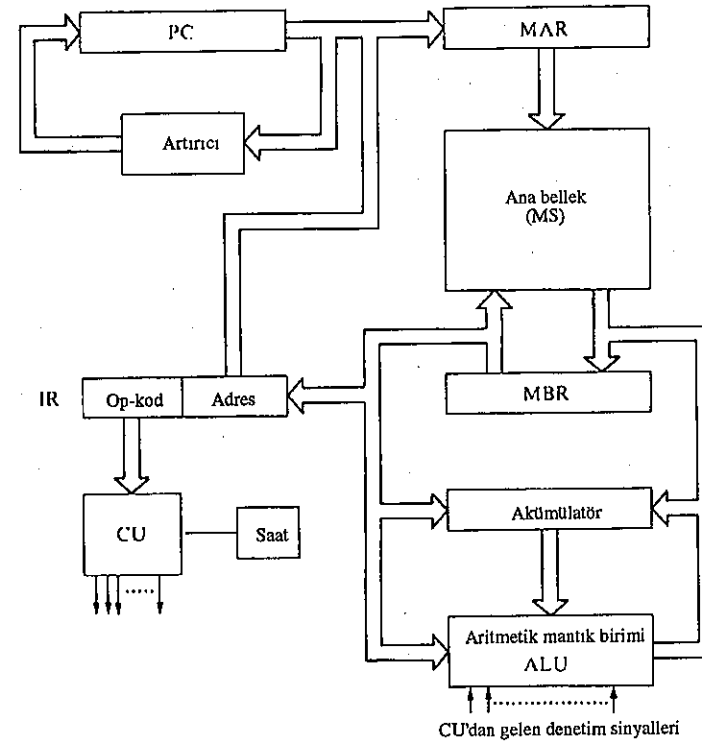
Tablo 4.1 Getirme saykılı

|         |          |                |
|---------|----------|----------------|
| GETİRME | [MAR]    | ← [PC]         |
|         | ARTIRICI | ← [PC]         |
|         | [PC]     | ← ARTIRICI     |
|         | [MBR]    | ← [MS[MAR]]    |
|         | [IR]     | ← [MBR]        |
|         | CU       | ← [IR(OP-KOD)] |

Bu tablodaki GETİRME deyiminin bir etiket olduğuna ve IR(OP-COD) ifadesinin de komut kaydedicisinin op-kod alanı anlamına geldiğine dikkat edin. [MS[MAR]] terimi, 'adresini MAR'ın içeriği ile verilen Ana Belleğin içerdiği değer' olarak okunur. Bazen, işlem kolaylığı için, 'MS' kısmının atılması yararlı olmaktadır, çünkü ifadeden anlaşılmaktadır. Bu nedenle, [MS[MAR]] terimi, [[MAR]] olarak yazılır.

Getirme saykılına gördük, şimdi de diğer saykılı, komut yürütmede nelere gerek olduğunu görelim. Şekil 4.3'de, MİB'mize veri yolları eklenmiştir.

MİB'ine eklenen yeni birimler, akümülatör, A, ve aritmetik ve mantık birimi, ALU'dur. Akümülatör, hesaplama sırasında geçici ve ara sonuçları tutar. ALU ise, (eğer gerekiyorsa) bellek tampon kaydedicisi ile birlikte akümülatörün içeriği üzerinde gerekli işlemleri gerçekleştirir. ALU'nun çıkışı, akümülatöre geri beslenir. ALU tarafından uygulamaya konulan iki türde işlem vardır: aritmetik ve mantıksal işlemler. Aritmetik ve mantıksal işlemler arasındaki temel fark, mantık işlemlerde, A sözcüğünün Ai biti ile B sözcüğünün Bi biti işleme sokulduğunda bir



Şekil 4.3 MİB'nin adres ve veri yolları

elde üretilmez. Bazı tipik aritmetik ve mantıksal işlem örnekleri Tablo 4.2'de verilmiştir.

Bilgisayarımızda bir adım daha atmış olarak, artık basit bir programı çalıştırabiliriz.

Tablo 4.2 Aritmetik ve mantıksal işlemler

| ARİTMETİK                     | MANTIKSAL |
|-------------------------------|-----------|
| Toplama                       | VEYA      |
| Çıkarma                       | VEDEĞİL   |
| Eksilme                       | VE        |
| Çarpma                        | VEDEĞİL   |
| Bölme                         | DEĞİL     |
| İkiyle bölme (sağa kaydırma)  | Özel-VEYA |
| İkiyle çarpma (sola kaydırma) | KAYDIRMA  |

$P = Q + R$  işlemini ele alalım. Burada, '+' işareti toplama işlemini göstermektedir. Buna ilişkin program, assembly dilinde aşağıdaki gibi yazılabilir:

LDA Q Akümülatöre Q'nun içeriğini yükle  
 CLC Elde bitini sil  
 ADC R Akümülatöre R'nin içeriği ve elde bitini ekle  
 STA P Akümülatörün içeriğini P'de sakla

MİB'nin çalışma yöntemi en iyi biçimde; kaydedici-aktarım dilini (bkz: Tablo 4.3) kullanarak ADC R komutunun yürütülmesi incelendiğinde görülebilir. Bu bize bilgisayarın içinde gerçekte neler olup bittiğini açıklar.

Tablo 4.3 Kaydedici-Aktarım Dili (RTL) 'nde ifade edilmiş ADR C

|         |                        |                                    |
|---------|------------------------|------------------------------------|
| GETİRME | [MAR] ← [PC]           | PC'nin içeriğini MAR'a aktar       |
|         | ARTIRICI ← [PC]        | PC'nin içeriğini artırıcıya aktar  |
|         | [PC] ← ARTIRICI        | artırıcının içeriğini PC'ye aktar  |
|         | [MBR] ← [MS[MAR]]      | ana bellekten oku                  |
|         | [IR] ← [MBR]           | MBR'nin içeriğini IR'ye aktar      |
|         | CU ← [IR (OP-KOD)]     | opkodu IR'den CU'ya aktar          |
| ADC     | [MAR] ← [IR (ADRES)]   | işlenenin adresini MAR'a aktar     |
|         | [MBR] ← [MS[MAR]]      | veri oku                           |
|         | ALU ← [MBR], ALU ← [A] | toplama işlemini yap               |
|         | [A] ← ALU              | ALU'nun çıkışını akümülatöre aktar |

İki işlemin aynı hattı paylaştığı yerde, işlemler eşzamanlı olarak yürütülürler. Burada, yukarıdaki işlemler genelde mikrokod olarak anılırlar. Assembly düzeyindeki her bir komut (örneğin LDA, ADC, STA) bir dizi mikrokod olarak

yürütülür. Genelde bazı makineler kullanıcı tarafından mikroprogramlanabilir olsa da mikrokodlar ve mikroprogramlama bilgisayar tasarımcısının uzmanlık alanlarıdır. Denetim biriminin çalışması ile ilgili mikrokodlar, Kısım 4.2.1'de biraz daha ayrıntılı olarak ele alınmıştır.

Şimdiye dek, basit programları tümüyle sıralı modda yürütebilen bir bilgisayarın mimarisini ele aldık - henüz burada işlem sırasını değiştirecek herhangi bir şey bulunmamaktadır. Yani, bilgisayar ne mutlak atlama (GOTO) ne de koşullu atlamalar yapmaz. Koşullu atlama IF THEN ve ELSE gibi üst-düzye yapıların gerçekleştirilmelerine olanak tanır. Bu açık, Şekil 4.4'de verilen düzenleme ile kapatılır.

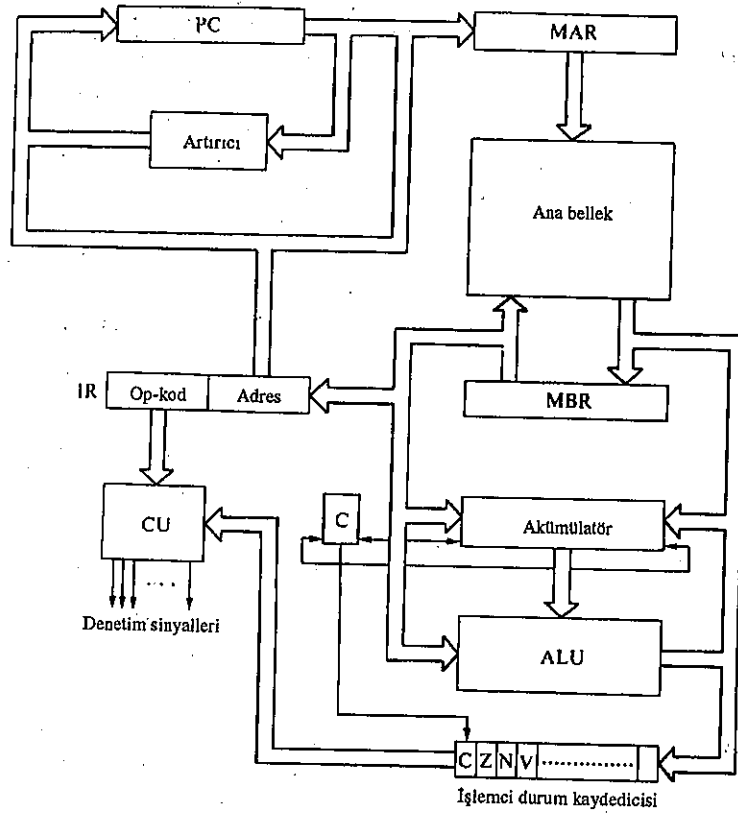
Bilgisayarımıza üç şey eklenmelidir: bir elde flip-flop'u, bir işlemci durum kaydedicisi (PSR) ve bir de komut kaydedicisinin adres alanıyla program sayıcısı arasında bir yol. Elde flip-flop'unun yaptığı iş oldukça açıktır. Her toplama veya çıkarma işleminde sonuç, akümülatöre aktarılır ve elde, veya ödünç, biti elde flipflop'unda saklanır. Benzer şekilde, akümülatörün değeri sola ve sağa bir bit taşınırsa (kaydırılırsa) "kenardan düşen bit", elde flip-flop'una aktarılır. Bunun sonucu olarak, elde biti akümülatörün bir-bitlik uzantısı olarak düşünülebilir.

İşlemci durum kaydedicisinin bir aritmetik/mantıksal işlemin sonucu ile ilişkili bazı bitleri vardır. Buradaki ilgili PSR bitleri şunlardır:

C = Elde son işlemlerde eğer elde üretilmişse kurulur  
 Z = Sıfır son işlem sıfır sonucunu üretmişse kurulur  
 N = Negatif son işlem negatif (2'nin tümleyenine göre) bir sonuç üretmişse (EDB= 1) kurulur  
 V = Taşma son işlem bir aritmetik taşma ile sonuçlanmışsa kurulur (sonucun işaret biti, her iki işlenenin işaret bitinden de farklıdır).

Bazı bilgisayarlar (6800, 6809) işlemci durum kaydedicilerine koşul kod kaydedicisi (CCR) der. PSR denetim birimine bağlanır ve bu da belli türde komutların PSR'yi sorgulamasını sağlar. Doğal olarak sonuçları etkilemediği sürece sorgulama yapmanın bir anlamı yoktur.

Bilgisayarımızda yapacağımız son değişiklik, komut kaydedicisinin adres alanı ile program sayıcısı arasına bir yolun eklenmesidir. Bilgisayarın, PSR'nin sorgulanmasına karşılık vermesini sağlayan işte bu özelliktir. Bir komut PSR'nin bir veya daha çok bitini test edebilir ve eğer bit sıfır ise, bir sonraki komut normal şekilde alınır; 1 ise, komut, adresi komut kaydedicisinde olan konumdan alınır. Bu tip



Şekil 4.4 Merkezi işlem birimi

komutlar koşullu dallanma veya atlama olarak bilinir. 'Dallanma' ile bir veya daha çok alternatif arasında geçiş yapmak kastedilirken; 'atlama', sıralı-olmayan bir denetimakışını ifade eder. Her iki terim de, bilgisayar biliminde birbirinin yerine kullanılır. Yukarıdaki örnekte eğer bit, 1'e kurulmuşsa bir dallanma yapılır ve benzer biçimde eğer bit sıfırlanmışsa bir dallanma yapılabilir. Koşullu dallanmaların bilgisayarda pratikte gerçekleştirilme biçimi Denetim Birimi hakkındaki Kısım 4.2'de tartışılmıştır.

Tablo 4.4 6502 mikroişlemcisi dallanma komutları

|     |                                    |               |
|-----|------------------------------------|---------------|
| BCS | elde sıfırda dallan                | C=0'sa dallan |
| BCS | elde birse dallan                  | C=1'se dallan |
| BEQ | sonuç sıfırda dallan               | Z=1'se dallan |
| BNE | sonuç sıfırdan-farklıysa dallan    | Z=0'sa dallan |
| BMI | sonuç eksiye dallan (2'ye tümly.)  | N=1'se dallan |
| BPL | sonuç artıysa dallan (2'ye tümly.) | N=0'sa dallan |
| BVC | taşma sıfırda dallan (2'ye tümly.) | V=0'sa dallan |
| BVS | taşma birse dallan (2'ye tümly.)   | V=1'se dallan |

Tablo 4.4'te 6502 mikroişlemcisinde gerçekleştirilen dallanma komutları listelenmiştir.

Motorola 6809 mikroişlemcisinin (Kısım 4.5'e bakınız) daha da büyük sayıda dallanma komutları vardır. Bunlar  $Z + (N \oplus V) = 0$  ise dallanmaya neden olan BGT (sıfırdan büyükse dallan) gibi eklemeleri içerir. BPL ile bunun arasındaki fark; BPL, sayı sıfırda veya görünüşte pozitifse (yani  $N = 0$ ) basit olarak bir dallanmaya neden olur. Oysa, BGT sayının sıfırdan büyük olmasını ister ve aynı zamanda taşma meydana geldiğinden ve hem  $N = 1$  hem de  $V = 1$  olduğundan, sayı negatif görünürse de dallanmaya neden olur.

Dallanma işlemleri kaydedici-aktarım dilinde 'eğer koşul sağlanırsa işi yap' formatında yazılır. Aşağıda kaydedici-aktarım dili terimleriyle ifade edilen iki koşullu dallanma örneği görülmektedir.

BCC eğer  $[C] = 0$  ise  $[PS] \leftarrow [IR(ADRES)]$   
 BEQ eğer  $[Z] = 1$  ise  $[PS] \leftarrow [IR(ADRES)]$

#### 4.1.1 Bir Programlama Örneği

Bu bölümde öğrendiğimiz şeyleri birbirine uydurmak için hayali makinemiz üzerindeki bir programlama örneği bize yardımcı olacaktır. İlk olarak, bilgisayar için bir komut takımı seçmemiz gerekmektedir. Birkaç istisnai durum dışında Tablo 4.5'teki komutlar, 6502 mikroişlemcisinininkilerdir. Aşağıda M ve N, sembolik değerleri gösterir.

Komut 09 (INC M) M bellek konumunun içerdiği değere bir ekler. Bilgisayar bilimlerine ilişkin kitaplarda 'arttırma' bir arttır anlamına gelir. Benzer şekilde, azaltma da bir azalt anlamına gelir.

'Sıfırda dallan' ve 'sıfır değilse dallan' komutları (16 ve 17) ALU'deki Z bayrağına bağımlıdır. ALU tarafından yerine getirilen son işlem, sıfır sonucunu vermişse, Z bayrağı bire kurulur.

Bu komut takımında, mevcut bilgisayarda yapılmayacak üç işlem vardır. Bunlar IN M, OUT M ve SET M'dir. Bu komutları gerçekleştirmek için sırasıyla akümülatörden çevrebirimimize, çevrebirimimizden akümülatöre ve komut kaydedicisinin adres alanından akümülatöre uzanan, veri yollarına ihtiyacımız vardır.

Örneğimiz için, giriş/çıkış cihazımızın bir VDU ve cihaz numarasının da 0 oldu-

ğunu varsayacağız. Veriyi VDU'nun klavyesinden okumak için IN 0 komutunu, ekranında bir sayı görüntülemek için de OUT 0 komutunu yürüteceğiz.

Tablo 4.5 Bilgisayar komut takımı

| Komut | Kom. Kıslt. | İşlem                                                                           |
|-------|-------------|---------------------------------------------------------------------------------|
| 00 M  | LDA M       | Akümülatöre M bellek konumunun içeriğini yükle                                  |
| 01 M  | STA M       | Akümülatörün içeriğini M bellek konumuna sakla                                  |
| 02 M  | IN M        | Cihaz numarası M'den akümülatöre bir sözcük gir                                 |
| 03 M  | OUT M       | Akümülatörün içeriğini birim numarası M'ye çık.                                 |
| 04 M  | ADC M       | M'nin içeriğini elde bitiyle beraber akümülatöre ekle                           |
| 05 M  | SUB M       | M'nin içeriğini akümülatörden çıkar                                             |
| 06 M  | AND M       | M'nin içeriğini akümülatörle mantıksal olarak VE'le Sonuç akümülatörde saklanır |
| 07 M  | ORA M       | M'nin içeriğini akümülatörle mantıksal olarak VEYA'la Sonuç akümülatörde kalır  |
| 08    | COM         | Akümülatörün değerini değille                                                   |
| 09    | INC M       | M bellek konumunun değerini artır                                               |
| 10    | DEC M       | M bellek konumunun değerini azalt                                               |
| 11    | CLC         | Eldeyi sil                                                                      |
| 12    | SEC         | Eldeyi kur                                                                      |
| 13    | ROL A       | Akümülatörü elde üzerinde sola döndür                                           |
| 14    | ROR A       | Akümülatörü elde üzerinde sağa döndür                                           |
| 15 M  | SET M       | M sayısını akümülatöre koy                                                      |
| 16 N  | BEQ N       | [Z] - 1 ise N konumuna dallan                                                   |
| 17 N  | BNE N       | [Z] - 0 ise N konumuna dallan                                                   |
| 18 N  | BCC N       | [C] - 0 ise N konumuna dallan                                                   |
| 19 N  | BCS N       | [C] - 1 ise N konumuna dallan                                                   |
| 20 N  | JMP N       | N konumuna koşulsuz dallan                                                      |
| 21    | STOP        | Dur                                                                             |

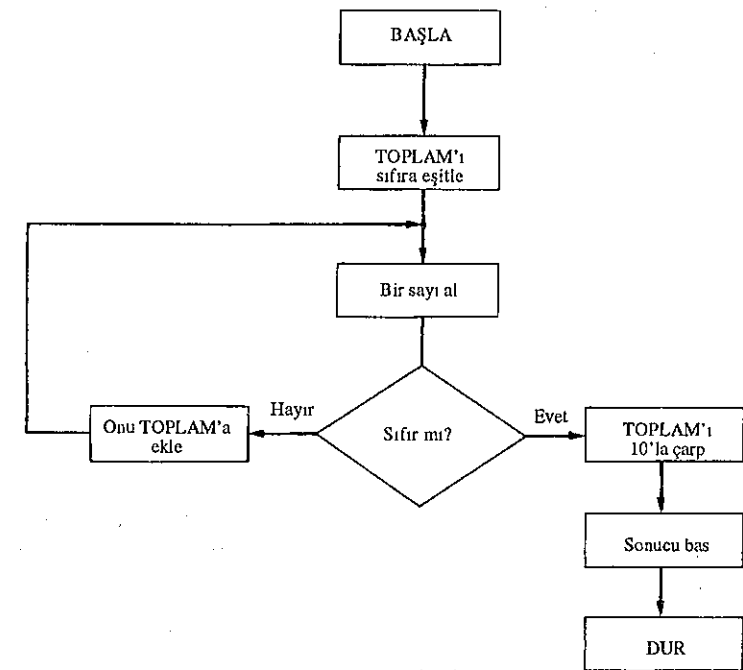
Çözmek istediğimiz problem şudur: sıfır ile sonlandırılan bir dizi sayıyı oku, bunları topla, sonucu 10 ile çarp ve cevabı bas. Üst-düzeyli bir dilde bu program Tablo 4.6'daki gibi yazılabilir.

Programları assembly dili düzeyinde yazan bazı kişiler, önce algoritmalarının bir akış diyagramını çizer sonra da bunu assembly diline dönüştürürler; bazıları da, programı üst-düzeyli bir dilde (çoğu kez Pascal ile) yazar ve deyimlerin her birini bir grup assembly dili işlemlerine kodlarlar. Fakat ilk grup assembly dilini doğru dan algoritmadan yazarlar. İkinci grup insanların programları hiçbir zaman doğru olmaz ve sınavlarda başarısız olurlar. Bu programın akış diyagramı Şekil 4.5'te verilmiştir.

Sayıyı 10'la çarpmamız gerektiğine dikkat ediniz. Komut takımı doğrudan çarpma olanaklarından yoksun olduğu için ya tekrarlamalı toplamaya başvurmalı ya da

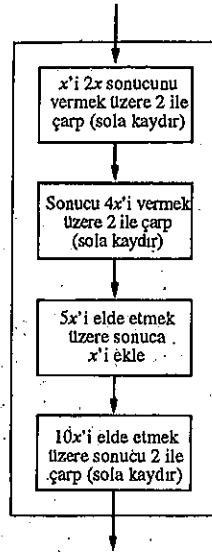
Tablo 4.6 Toplayıp onla çarpan program

| BASIC                  | Pascal                       |
|------------------------|------------------------------|
| 10 T=0                 | Program Sum (Input, Output); |
| 20 INPUT N             | Var n, t: Integer;           |
| 30 IF N=0 THEN GOTO 60 | BEGIN                        |
| 40 T=T+N               | t:=0;                        |
| 50 GOTO 20             | read(n)                      |
| 60 T=T*10              | WHILE n < > 0 DO             |
| 70 PRINT T             | BEGIN                        |
| 80 STOP                | t:=t+n;                      |
|                        | read(n)                      |
|                        | END;                         |
|                        | t:=t*10;                     |
|                        | write(t)                     |
|                        | END                          |



Şekil 4.5 Programın akış diyagramı

çarpma yapan bir program yazmalıyız. Alternatif bir çözüm de,  $10x = (2 \times 2x + x) \times 2$  olduğuna dikkat etmektir. 10 ile çarpmak için akış diyagramı Şekil 4.6'da gösterilen adımları içerir. Programı yazmaya başlamadan önce, bilgisayar belleğinde nereye koyacağımıza karar vermeliyiz. Kolaylık açısından programın, konum 0'dan başlayacağını varsayabiliriz. Henüz programın ne kadar uzun olacağını bilmiyoruz, ama akış diyagramından (sözelimi) 20 komutu geçmemesi (kutu sayısı çarpı 2) gerektiğini çıkarabiliriz. Bu nedenle de değişkenleri, konum 30'dan daha öteye yerleştirmek güvenli olacaktır.



Şekil 4.6 10'la çarpmada izlenecek yöntem

Tablo 4.7 Assembly Dili Program

| No. | Açıklama |        |
|-----|----------|--------|
| 1.  | NAM      | ORNEK  |
| 2.  | ORG      | 30     |
| 3.  | TOPLAM   | RMB 1  |
| 4.  | ORG      | 0      |
| 5.  | SET      | 0      |
| 6.  | STA      | TOPLAM |
| 7.  | TEKRAR   | IN 0   |
| 8.  | BEQ      | CARP   |
| 9.  | CLC      |        |
| 10. | ADC      | TOPLAM |
| 11. | STA      | TOPLAM |
| 12. | JMP      | TEKRAR |
| 13. | CARP     | CLC    |
| 14. | LDA      | TOPLAM |
| 15. | ROL      | A      |
| 16. | CLC      |        |
| 17. | ROL      | A      |
| 18. | CLC      |        |
| 19. | ADC      | TOPLAM |
| 20. | ROL      | A      |
| 21. | OUT      | 0      |
| 22. | STOP     |        |
| 23. | END      |        |

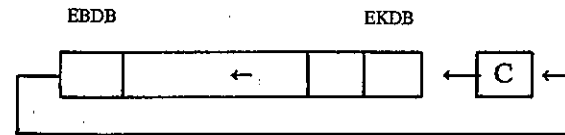
1'den 23'e kadar sayıları içeren ilk sütun, programın bir parçası değildir ve sadece sonraki göndermeler için satır numaralarıdır. TOPLAM, TEKRAR ve CARP ifadelerini içeren ikinci sütun, programın etiket alanıdır. Bu üç sözcük assembly dili komutları tarafından gönderme yapılabilecek etiketlerdir. Örneğin BEQ CARP sıfır sonucu durumunda CARP ile etiketlenmiş komutun adresine dallan anlamına gelir. Daha sonra, program bir derleyici ile makine koduna çevrildiğinde bu etiketlere yapılan tüm göndermeler, etiketledikleri satırın adresine çevrilecektir.

İlk bakışta bu programda, bazı komut kısaltmaları bulunduğu, fakat bunların komut takımında görünmediği anlaşılmaktadır (1, 2, 3, 4 ve 23. satırlar). Bunlar assembly dili işlemleri değil, çevirici yönergeleridir. Sonuçta program, bellekte saklı durumda, sayısal veya makine-kodu biçiminde olacaktır. Tablo 4.7'deki versiyon okunup yazılabilmeye uygun şekilde yazılmıştır ve sonuçta makine koduna çevrilecektir. Normalde bu, çevirici denen bir programla yapılır. Bu iş elle de yapılabilir ama bu önerilen bir işlem değildir.

Çevirici yönergeleri, çeviriciye programla ilgili bilmek istediği şeyleri söyler. İlk assembler yönergesi, NAM programa bir ad verir (bu örnekte ORNEK). İkinci assembler yönergesi, ORG, başlangıç konumunu 30'a kurar. Yani, bu deyimi işleyen program veya veriler, adres 30 ve ardından gelen olan konumlara yerleştirilecektir. RMB, adlandırılmış değer için bir veya daha çok bellek konumu ayırır. Böylece TOPLAM RMB 1, TOPLAM adı verilen parça için tek bir konum ayırır. İkinci ORG başlangıç noktasını 0'a kurmak suretiyle programın adres 0'dan başlayarak belleğe yüklenmesini sağlar. Son çevirici yönergesi, END, çeviriciye programın sonuna ulaşıldığını söyler.

Kullanılan dallanma komutuna; bir sayıdan çok, bir etiket eşlik eder. Bu da programcının adresleri takip edebilme işinde kolaylık sağlar. Programda dikkate değer diğer nokta ise, kaydırma işlemleri başlamadan önce CLC komutunun kullanılmasıdır. Bunun nedeni, ROL'un akümülatörü elde üzerinden sola döndürmesidir.

Yani;



Açıkça görüldüğü gibi, başlangıçta elde biti sıfırlanmalıdır; aksi takdirde akümülatörün EDBD'ne hatalı bir bit kaydırıyor olabiliriz. Kolaylık açısından sayıların

toplamı 10 ile çarpıldığında aralık-dışında bir sonuç üretmeyeceği varsayılır. Yani, ROL yürütüldüğünde en büyük değerlikli bit (EBDB) kullanımından çıkış-eldesinin üretileceği konusunda kaygı duymamamız gerekir.

Program bir kez birleştirilip bilgisayarın ana saklama ortamına yerleştirildikten sonra yürütülebilir. Bilgisayarın bellek haritasının planı Tablo 4.8'de verilmiştir.

Tablo 4.8 Programın bellek haritası

| Adres | İçerik |
|-------|--------|
| 0     | 1500   |
| 1     | 0103   |
| 2     | 0200   |
| 3     | 1608   |
| 4     | 11XX   |
| 5     | 0430   |
| 6     | 0130   |
| 7     | 2002   |
| 8     | 11XX   |
| 9     | 0030   |
| 10    | 13XX   |
| 11    | 11XX   |
| 12    | 13XX   |
| 13    | 0430   |
| 14    | 13XX   |
| 15    | 0300   |
| 16    | 21XX   |
| .     | .      |
| .     | .      |
| .     | .      |
| 30    |        |
| .     | .      |
| .     | .      |
| .     | .      |
| .     | .      |

← TOPLAM için ayrılmış

Her bellek konumunda 4-basamak tutulur. Bir komutun ilk iki basamağı işlem kodunu tanımlar. İkinci ikili ise opkod tarafından erişilen bellek konumunu göstermektedir. Bunun istisnaları IN ve OUT komutlarıdır ve burada bir çevrebirimi tanımlanır, bunu izleyen iki basamağın akümülatöre yüklendiği SET işlemi ve bellek konumlarına gönderme yapmayan CLC ve ROL gibi komutlardır. Bu son durumda XX, 'dikkate alınmaz' durumunu gösterir - bu basamakların herhangi bir değeri olabilir. Pratikte çoğu zaman sifıra kurulurlar.

## 4.2 Denetim Birimi

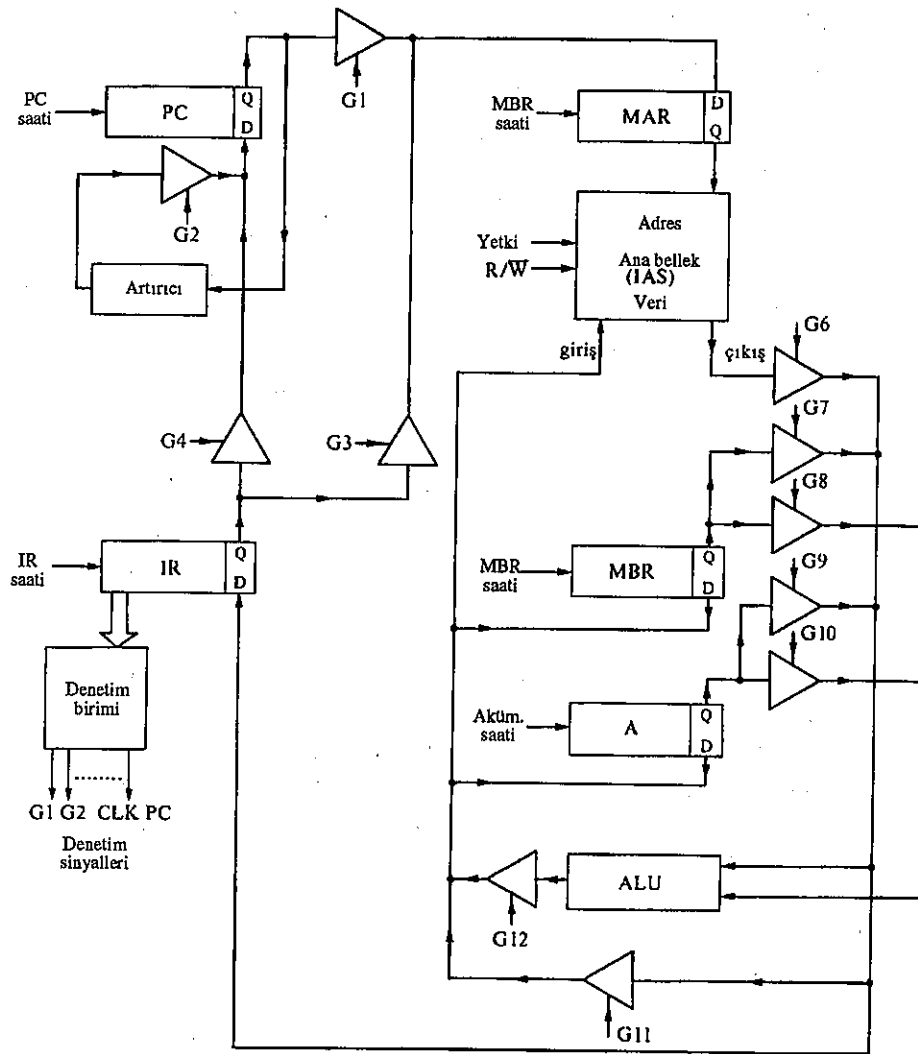
Sayısal bir bilgisayarın makine kodu komutlarını yorumlamasının tam şekli, oldukça karmaşıktır ve bu nedenle de bilgisayarın denetim birimi ile ilgili tartışma çoğu kez bilgisayar mimarisi hakkındaki daha ileri derslere bırakılır. Biz burada tek bir makine kodu komutunun yürütülmesini ele alacağız. Kullandığımız model, temel işlemin anlaşılmasını daha kolay kılmak için gereksiz tüm ayrıntılardan ayıklanmıştır. Denetim biriminin tasarımı ile ilgili temelde farklı iki yaklaşım vardır. İlki (Kısım 4.2.1) denetim biriminin kendisini, bilgisayarın içinde bulundurmak ve makine-düzeyindeki komutları mikrokompüt denilen çok daha ilkel komutlar dizisine çevirmektir. Alternatif yaklaşım ise (Kısım 4.2.2) bir komutu işleme koymak için ne tür bir mantık ve aritmetik işlem dizisi gerektiğini sormak ve sonra da bunun gerçekleştirilmesini sağlayacak uygun mantık devresini tasarlamaktır.

### 4.2.1 Mikroprogramlı Denetim Birimi

Denetim biriminin tanımlanmasından önce, şu üç terimin tanımlaması gerekir. Bunlar makrokompütler, mikrokompütler ve yorumlamadır. Bilgisayarların en ilkel diline makine kodu ve komut kısaltması biçimindeki gösterimine de assembly dili denir. Bu komutlara aynı zamanda makrokompütler adı da verilir. Her makrokompüt, mikrokompüt denen bir grup işlem aracılığıyla sonuçlandırılır. Mikrokompüt veya mikro-işlem bir bilgisayarda ortaya çıkabilecek en küçük olaydır. Bir mikroişlem bir flip-flopu saatlemekten veya veriyi bir kaydediciden diğerine taşımaktan oluşabilir. Bir makrokompütün bir dizi mikro-işlem ile yürütülmesine yorumlama denir. Burada kullanılan makro örneğinin programlamadaki kullanımından farklı olduğuna dikkat edin.

Bir MİB'in iç yapısının bir bölümü Şekil 4.7'de gösterilmiştir. Bu şekil, bilgilerin MİB içinde nasıl taşındığını göstermesi açısından Şekil 4.2-4'den farklıdır. Şekil 4.7'deki kaydedicilerin (program sayıcısı, akümülatör vs.) her biri, D flip-floplarından yapılmıştır. Bir kaydedicinin saat girişine darbe uygulandığında giriş uçlarındaki veri çıkış uçlarına aktarılır ve kaydedici yeniden saatlenene kadar sabit tutulur. Kaydediciler arasındaki bağlantı m-bitlik genişlikte bir veri anayolu aracılığıyla yapılır. Şekil 4.7'de, karmaşıklığa yol açmamak için, veri yollarının sadece bir biti gösterilmiştir.

Sözgelimi, bilgisayarımızın, opkodu toplama işlemi olduğu bir getir-yürüt saykılını gerçekleştirmesini istiyoruz. İşlem dizisini, yalnızca kaydedici dili ile değil, aynı zamanda kapıların yetkilendirilmesi ve flip-flop'ların saatlenme biçimine göre de yazabiliriz. Tablo 4.9'a bakınız.



Şekil 4.7 MİB içindeki veri akışının denetimi

Tablo 4.9 Bir op-kodun yorumlanması için gereken işlemler

| Adım | Kaydedici Aktarım Dili | Gereken işlem                                                                                                |
|------|------------------------|--------------------------------------------------------------------------------------------------------------|
| 1    | [MAR] ← [PC]           | G1'in yetkilendirilmesi, MAR'ın saatlenmesi                                                                  |
| 1a   | INC ← [PC]             |                                                                                                              |
| 2    | [PC] ← INC             | G2'nin yetkilendirilmesi, PC'nin saatlenmesi                                                                 |
| 3    | [MBR] ← [MS[MAR]]      | MS'nin yetkilendirilmesi, R/W = 1<br>G6'nın yetkilendirilmesi, G11'in yetkilendirilmesi, MBR'nin saatlenmesi |
| 4    | [IR] ← [MBR]           | G7'nin yetkilendirilmesi, IR'nin saatlenmesi                                                                 |
| 4a   | CU ← [IR(OP-CODE)]     |                                                                                                              |
| 5    | [MAR] ← [IR(ADDRESS)]  | G3'ün yetkilendirilmesi, MAR'ın saatlenmesi                                                                  |
| 6    | [MBR] ← [MS[MAR]]      | MS'nin yetkilendirilmesi, R/W-1<br>G6'nın yetkilendirilmesi, G11'in yetkilendirilmesi, MBR'nin saatlenmesi   |
| 7    | ALU ← [MBR]            | G7'nin yetkilendirilmesi                                                                                     |
| 7a   | ALU ← [A]              | G10'un yetkilendirilmesi                                                                                     |
| 8    | [A] ← ALU              | G12'nin yetkilendirilmesi, A'nın saatlenmesi                                                                 |

Not 1. Üçüncü sütunda girişin olmadığı yerlerde işlem otomatik olarak gerçekleşir. Örneğin program sayıcısının çıkışı, daima artırıcının girişine bağlanır ve böylece PC'nin değerini artırıcıya taşımak için dışardan bir işleme gerek duyulmaz.

Not 2. Üç-durumlu kapılar aksi belirtilmedikçe yetkilendirilmezler.

Not 3. Bazı işlemler (örneğin 7 ve 7a adımları) eşzamanlı olarak gerçekleştirilir.

Şimdi denetim birimi çıkışının; 1'den 12'ye kadar olan kapıları yetkilendirmek üzere oniki sinyal, MS'yi denetlemek için iki sinyal, ve PC, MAR, MBR, IR ve A kaydedicilerinin saat girişlerine darbe uygulayacak beş sinyalden oluştuğunu düşünelim. Tablo 4.10, denetim biriminin çıkışlarını bir dizi ikili değer olarak vermektedir.

Eğer, Tablo 4.10'daki sekiz adımdan her biri için, Şekil 4.7'de verilen MİB'in çeşitli birimleri ondokuz sinyalle beslenirse, getir/yürüt saykılı yerine getirilecektir. Gerçek bir bilgisayarda, CU tarafından üretilen 19 donanım sinyalinden çok daha fazla sinyal vardır. Tipik sayı 64 ile 200 arasındadır.

Tablo 4.10'daki sekiz adım, ardından bir ADD komutunun geldiği bir getirme fazı yorumlayan mikroprogramı göstermektedir.

Denetim biriminin çalışması açısından bir sonraki adım mikroprogramın nasıl üretildiğini anlamaktır. Mikroprogram; bilgisayarın kendisine benzer bir mekanizmayla üretilir (tekerlekler içinde tekerlekler?). Örneğin, program sayıcısı yerine burada, mikroprogram sayıcısı vardır.



'Yükleme denetim' alanı mikroprogram sayıcısına bir sonraki adresini ya arttırıcıdan ya adres eşleyiciden (aşağıya bakınız) alacağını ya da mikrokomut kaydedicisindeki adresi kullanacağını söyler.

'Koşul seçim' alanı koşullu atlamaları gerçekleştirir. Eğer çoğullayıcının çıkışı doğruysa, bir sonraki mikroprogram adres konumuna bir atlama yapılır; aksi takdirde mikroprogram normal sırasını izleyerek devam eder. İlk iki koşul MİB'deki PSW'den alınır. İkinci iki koşul 'hiç bir zaman dallanma' (devam et) ve 'sürekli dallan' (koşulsuz atlama)'dır.

Makrokomut düzeyindeki koşullu bir atlama, (örneğin BEQ) mikrokomutlar tarafından aşağıdaki şekilde yorumlanır. Mikrokomutun koşul seçme alanı, test edilmesi için, PSR'nin uygun durum bitini seçer. Örneğin, eğer makrokomut BEQ ise Z biti seçilir. Yüklem denetim alanı, 'seçilen koşul doğru olduğunda mikroprogram kaydedicisindeki adrese dallan' işlemini içermektedir. Böylece, eğer koşul doğruysa (yani Z= 1) makroprogramda karşılığı olan atlamayı gerçekleştiren mikroprogramda belli bir noktaya bir atlama yapılır. Eğer seçilen koşul yanlışsa (yani Z= 0), o anki mikroprogram dizisi yeni bir getir-yürüt saykılının başlamasıyla sonlandırılır.

Denetim birimi tarafından yürütülen her bir mikroprogramın ilk fazı, bir getirme fazına karşılık gelir. Bu faz opkodun komut kaydedicisine saklanmasıyla biter. Bu opkod, MİB tarafından yürütülen olası tüm opkodlar için her bir mikroprogramın başlama adresini içeren adres eşleyiciye beslenir.

Getirme fazının son mikrokomutu, mikroprogram sayıcısının, komut kaydedicisinde bulunan opkoda karşılık gelen mikroprogramın başlangıç adresiyle yüklenmesine neden olur. Bu mikroprogram yürütüldükten sonra getirme fazına karşılık gelen mikroprogramın başlangıcına koşulsuz bir atlama yapılır, ve işlem devam eder.

Mikroprogramlanmış denetim biriminin yukarıdaki tanımı elbette ki oldukça basitleştirilmiştir. Pratikte mikroprogramda kesmeleri, ana belleği (IAS) ve giriş/çıkışı yöneten bir çok olanak vardır. Mikroprogramlanmış denetim biriminin en büyük avantajlarından biri de, mikroprogram belleğinin (kimisi zaman denetim saklama ortamı denir) içeriğini değiştirmenin ve böylece kendi-makine-düzeyi-komutlarını-yuvarlamanın mümkün olmasıdır.

Aslında tamamıyla farklı bir bilgisayarın makine kodunu yürütecek bir mikroprogram takımı seçmek pekala mümkündür. Bu durumda bilgisayarın, diğer bilgisayarın

emüle ettiği söylenir. Bu tip bir olanak, eğer eski makinenizi yeni bir makineyle değiştiriyorsanız ve yeni makinenin makine-kodu eski programlarınızla uyumlu değilse oldukça yararlı olacaktır. Bu durum teyp veya disketteki ikili (amaç kodu) biçimde bulunan programlar için geçerlidir. Eski makinenin makine-kodunu yorumlayabilmek üzere mikroprogramlar yazarak (yeni makinede) eski yazılımı kullanıp yine de yeni makinenin avantajlarından yararlanmak mümkündür.

#### 4.2.2 Rastgele Mantık Denetim Birimleri

Kısım 4.2.1'de verilen denetim birimi türü, makine-kodu komutlarının yorumunu sadece-okunur bellekte saklı bir mikroprogram aracılığıyla yapar. Tıpkı makine-kodu düzeyindeki bir bilgisayarın karmaşıklığının, bu makinede çalıştırılan üst-düzeyli programların karmaşıklığıyla ilişkili olmamasında olduğu gibi, mikroprogramlanmış denetim biriminin karmaşıklığı da doğrudan yorumladığı makine-kodu komutlarının karmaşıklığıyla ilgili değildir.

Bir mühendis rastgele mantık denetim birimi tasarlarırken sorar: 'Her bir makine-kodu komutunu yürütmek için hangi mikro-işlemler dizisine gerek duyulur ve bunları gerçekleştirmek için hangi mantık elemanlarına ihtiyacım vardır? Bir başka deyişle, tasarımcı daha önce 2. Bölümde karşılaştığımız Boole tekniklerine başvurur. 'Rastgele' sözcüğü ile denetim birimini oluşturan kapı düzenlemesinin geniş ölçüde bilgisayardan bilgisayara değiştiği anlatılmak istenmektedir. Aynı mikroprogramlanmış denetim birimi çok az değişikliklerle farklı bir bilgisayara uyarlanabilir fakat rastgele mantık denetim birimi için bu oldukça zor bir ilişkidir.

Rastgele bir mantık denetim birimi tasarlamadan önce, son derece basit bir bilgisayar ele alalım. Bunun için, bu bölümde daha önce anlatılan MİB yapılarını kullanmak mümkündür. Öte yandan, bir başka tasarımı benimseyerek okura MİB'in mimarisinin mühendis tarafından seçildiğini ve kesin olarak sabit bir yapı olmadığını göstermeyi amaçlıyorum. Bir bilgisayar tasarlanırken, mühendis hesaplama gücü, hızı ile maliyeti arasındaki dengeyi iyi hesaplamalıdır.

Şekil 4.9'da, ilkel bir MİB'in yapısı gösterilmiştir. İlkel olmasının nedeni yol ve işlevsel birim sayısının en alt düzeyde tutulmuş olmasıdır. Bu, tasarımın kolayca üretilebilmesine neden olmakla birlikte sistemin çalışma hızını azaltacaktır. Sadece bir yol (sistem yolu) olduğundan birden daha fazla mikro-işlem eşzamanlı olarak yapılamayacaktır. Örneğin program sayıcısı için ayrı bir arttırıcı yoktur. Bu da ALU'yu PC'nin değerini arttırmak için kullanılmaya zorlar. Bunun sonucu olarak program sayıcısının içeriği arttırılırken ALU ve ilgili veri yolları diğer işlemler için kullanılabilir durumda değildir.









tasarımının ayrıntılarına inemeyiz ve bu yüzden de yalnızca mikroprogramlanmış ve rastgele mantık denetim birimlerinin en önemli özelliklerini belirtmekle yetineceğiz:

1. Rastgele mantık denetim birimleri, mikroprogramlanmış eşdeğerlerinden daha hızlıdır. Rastgele mantık denetim birimleri özel uygulamalar için optimize edilmişlerindendir, bu daima böyledir. Ayrıca, mikroprogramlanmış denetim biriminin mikroprogram belleğinden mikrokod okuması gerektiğinden dolayı yavaşlar. Bellek erişimleri genellikle temel Boole işlemlerinden daha yavaştır.

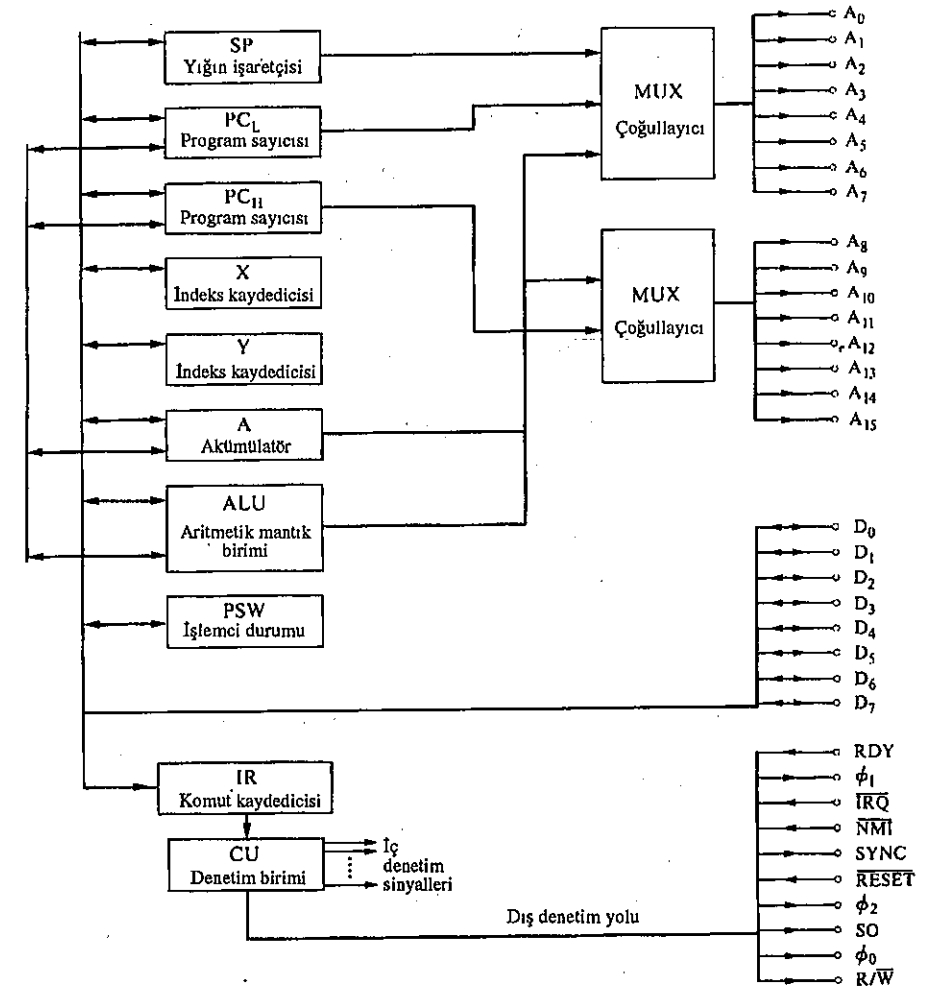
2. Mikroprogramlama tasarımcıya esnek bir tasarım sunar. Mikroprogram (sadece okunur) bellekte bulunduğundan, tasarım veya üretim aşamalarında kolayca değiştirilebilir. Rastgele mantık denetim birimi ise kesinlikle böyle değildir ve işlemdeki yeni özelliklere (örneğin ek makine-düzeyi komutlarına) uyarlanmak üzere kolayca değiştirilemez. Ayrıca kimi zaman donanımda önemli ölçüde değişiklik yapmadan tasarım hatalarını ortadan kaldırmak zordur.

#### 4.3 6502 Mikroişlemcisi

8-bitlik mikroişlemcilerinin ilk zamanlarında çok popüler olan iki aygıt vardı; Intel 8080 ve Motorola 6800. Bazı mühendisler, Intel'i, Z80 denen 8080'in daha gelişmiş bir versiyonunu üretmek için, bazıları da Motorola'yı, 6500 serisi mikroişlemcileri üretmek için terk etmiştir. Yukarıdaki mikroişlemcilerden herhangi biri bilgisayar mimarisini öğretmek için araç olarak kullanılabilir. Burada 6502'nin seçilmesinin nedeni, anlaşılmasının nispeten kolay olması ve düşük-maliyetli kişisel bilgisayarlarda yaygın olarak kullanılmasıdır.

Mikroişlemcilerle (yonga üzerindeki MİB olarak mikroişlemci) diğer bilgisayarlar arasındaki en önemli farklılıklardan biri, mikroişlemci tasarımcılarının iki sınırlamayla karşı karşıya olmalarıdır. İlk olarak, tek bir yongaya konabilecek kapı sayısı mevcut teknoloji ile sınırlıdır. İkinci olarak; yongalar artık 48 ve 64 pinli olarak üretilebilse de yongaya dışardan bağlanabilen bağlantı sayısı yaklaşık 40 ile sınırlıdır. Çok karmaşık yapıya bazı 16-bitlik mikroişlemciler olsa da, bilgisayar mimarisine girişte sekiz-bitlik yonga kullanılması, karmaşıklık ve sadelik açısından idealdir.

6502'nin basitleştirilmiş iç yapısı Şekil 4.15'te verilmiştir. Bir mikroişlemci, veya herhangi bir bilgisayar, üç farklı şekilde incelenebilir. Tasarımcı iç mimariye bakarsa, cihazı o yapmıştır ve tüm konularda, özellikle de denetim birimiyle ilgili genel bir



Şekil 4.15 6502 mikroişlemcisinin iç yapısı

bilgiye sahip bulunmalıdır. MİB ile sistem arasındaki arabirimini oluşturan mühendis onun elektriksel özellikleriyle ilgilidir. Her bir pindeki gerilim ve akımlar, yongaya giren veya çıkan çeşitli sinyallerin gerektirdiği zamanlama gerekleri ve protokoller hakkında bilgi sahibi olmak ister. Programcı bu ince ayrıntılarla ilgilenmez. Onun gerek duyduğu, makinenin programlama modelidir. Yani; komut takımını ve iç kaydedicilerin düzenlenmesini bilmek zorundadır. Burada verilen 6502 diyagramı programcının modelidir.

MİB ile ilgili daha önceki tartışmamızda veri-sözcüklerinin uzunluklarına bakılmamıştır. Gerçek bir bilgisayar, belli bir veri sözcük-uzunluğuna sahiptir, bu

uzunluk, normalde ana saklama ortamındaki konumlarda saklanan sözcüklerle büyüklüktedir. Bit, bir bilgisayarda saklanan en küçük veri birimi ise de, aritmetik ve mantıksal işlemlerde kullanılan temel bilgi birimi sözcüklerdir.

Doğal olarak programcı kendisini aritmetik işlemlerinin duyarlılığı veya tek sözcük ile adresleyebileceği aralığa ilişkin kaygılardan kurtaracak, pratikte sonuç sözcük-uzunluğu isteyecektir. Ne yazık ki ana belleğin maliyeti çoğu zaman bilgisayar sisteminin en büyük dilimini oluşturur ve dolayısıyla da programcının istekleri yerine getirilmez.

Bir bilgisayarla ilişkili gerçek sözcük-uzunluğu birçok faktörün uzlaşmasıyla belirlenir ve tasarımcının, bilgisayar pazarının belli bir sektörün cevabına yönelik çabasına karşılık gelir. İlk bakışta bir bilgisayarın minimum sözcük büyüklüğünün komut kaydedicisinin opkod alanındaki bit sayısı artı adres alanındaki bit sayısına eşit olması gerektiği düşünülebilir. Yani;

$$\begin{aligned} \text{minimum sözcük-uzunluğu} &= \log_2 (\text{komut sayısı}) \\ &+ \log_2 (\text{bellek konum sayısı}) \end{aligned}$$

Örneğin, 256 komut varsa ve çalıştırılacak en büyük program 65.536 konum işgal ediyorsa, minimum sözcük-uzunluğu şu olacaktır:

$$\log_2 (256) + \log_2 (65\,536) = 8 + 16 = 24.$$

Bereket versin ki, en azından mikroişlemci tasarımcıları için, yukarıdaki denklemle önerilen minimum sözcük-uzunluğu zorunlu değildir. Belli bir bilgisayarın istediği minimum sözcük-uzunluğunu azaltmanın iki yolu vardır. Adres bitlerinin sayısı, sayfalandırma ile azaltılabilir. Mevcut adres alanı sayfa olarak adlandırılan gruplara bölünür; böylece 65.536 ( $2^{16}$ ) bellek alanı, sözgelimi, 256 konumluk 256 sayfaya ayrılır. Bilgisayarın adres belirttiği her seferde, belli bir sayfadaki konumu seçmek için 8-bite, ve belli bir sayfayı seçmek için de 8-bite ihtiyacı olacaktır. Yani sonuçta sekiz-bitlik adresleri işleyecektir. Açıkça görüldüğü gibi bu düzenleme, programın yürütülmesi sırasında birçok kereler bir sayfadan diğerine bir geçiş gerektiğinde de, karmaşık ve zaman-kaybettirici olur.

Minimum sözcük-uzunluğunu azaltmanın ikinci yolu, bir komutun opkod ile adres alanlarını ayrı ayrı saklamaktır. Bu yaklaşım gerçekte tüm sekiz-bitlik mikroişlemciler tarafından benimsenmiştir. 8-bitlik bir sözcük-uzunluğu, 256 farklı komuta kadar izin verir ki bu da en çok talepte bulunanlar dışındaki tüm uygulamalar için yeterlidir. Getirme saykılı boyunca işlem kodu bellekten okunur ve gözden geçirilir.

İr. Eğer komut, bir adrese ihtiyaç duyarsa, iki okuma daha yapılır ve iki tane sekiz-bitlik sözcükten 16-bitlik bir adres oluşturulur. Bunun sonucu olarak ardışık üç tane 8-bitlik sözcük olarak elde edilen 24-bitlik bir sözcük-uzunluğuna sahip oluruz.

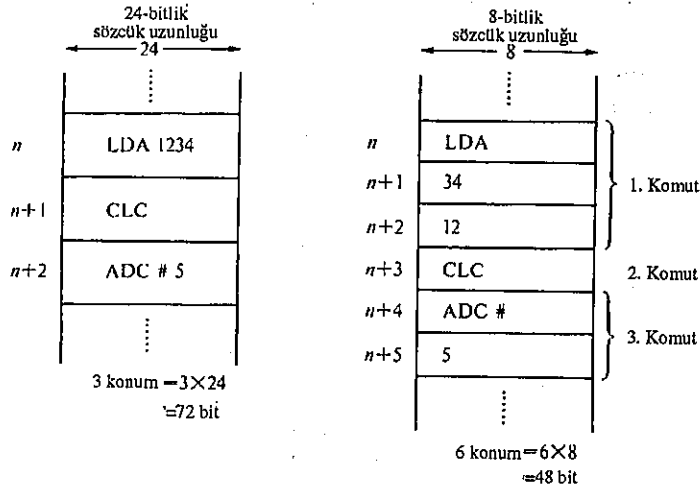
Bu düzenlemenin iki avantajı vardır. İlk olarak, sekiz-bitlik bir sözcük uzunluğu kullanarak sistemin maliyeti büyük ölçüde azaltılabilir. Çünkü sekiz-bit uzunlukta bir veri anayolu, 24-bitlik bir veri anayolundan çok daha ucuza mal olur. Bundan başka, bir mikroişlemcinin bacak sayısı (dış bağlantılar) halen 64 ile sınırlıdır ve bu da mikroişlemcilerin çok büyük sözcük-uzunluğuna sahip olamayacakları anlamına gelmektedir. İkinci olarak, tüm komutların 24-bitlik uzunlukta olması gerekmez; böylece sekiz veya onaltı-bitlik bir komut, değerli bellek alanını harcamamış olacaktır. Bazı komutlar, örneğin LDA, işlenen adresine gereksinim duyarken diğer, CLC (eldeyi sıfırla) veya INX (X kaydedici içeriğini arttır) gibi komutlar, bu adres alanına ihtiyaç duymazlar. Bu komutların doğal adresleme moduna sahip olduğu söylenir, çünkü işlenenin adresi, komutun içinde doğal olarak belirtilmektedir.

Yukarıdaki noktalar Şekil 4.16'da verilen örnekte daha belirgin olarak açıklanmaktadır. 24-bitlik bilgisayar, bu kadar büyük bir uzunluk gereksiz olduğunda bile, 24-bitlik bir sözcük kullanmaya zorlanırken 8-bitlik bir makine tek bir komutu tutmak için 1, 2 veya 3 ardışık konumu kullanmaktadır. Bu yapının, daha baştan bir dezavantaja sahip olduğuna dikkat ediniz - bellekten üç tane 8-bitlik sözcük okumak 24-bitlik tek bir sözcük okumaktan daha uzun zaman alır. Daha önce, MIB'in yapısı açıklanırken program sayıcısının içeriğinin bir opkod getirme sırasında arttırıldığı belirtilmişti. Eğer değişken-uzunlukta bir komut kullanılırsa, komutun ilk baytı komut getirme saykılı sırasında incelenir ve program sayıcısı sırasıyla 1, 2 veya 3 olarak arttırılır.

#### 6502'nin Kaydedicileri

6502'nin programcı tarafından erişilebilecek (yonga-içi) yedi tane kaydedicisi vardır: akümülatör, işlemci durum kaydedicisi (PSR veya CCR), program sayıcısı (üst), program sayıcısı (alt), X indeks kaydedicisi, Y indeks kaydedicisi, yığın işaretçisi (SP). 16-bitlik program sayıcısının, 6502'nin iç devresi tarafından iki tane 8-bitlik kaydedici olarak görüldüğüne dikkat edin. Programcı açısından sadece bir tane 16-bitlik program sayıcısı vardır.

6502'nin 'yeni' kaydedicileri, X ve Y kaydedicileri ile yığın işaretçisidir. Bu



Şekil 4.16 24-bitlik sözcük uzunluğuna karşılık 8-bitlik sözcük uzunluğu

kaydediciler çok önemlidir ve özel adresleme modları ile kullanılmaları gerekir. Şimdiye kadar bilgisayarların ne yapabildiğine veri üzerindeki işlemler açısından baktık. Bundan sonraki kısımda, özel adresleme modlarının veri tablolarına hızla erişilebilmesini, işlem süreçlerinin (altyordamların) gerçekleştirilmesini, programların fiziksel olarak yerlerini değiştirmelerini, bunlar üzerinde değişiklik yapmadan nasıl oluşturulabildiklerine bakacağız.

#### 4.3.1 6502'nin Komut takımına Giriş

Bu kısımda, 6502'nin komut takımında bulunan bazı assembly dili komutları açıklanacaktır. Amacım, bir mikroişlemcinin yapabildiği bir kaç şeyi göstermektir. 6502'nin komut takımının ayrıntılı tanımı Ek A2'de verilmiştir. İşlemcinin gerçekte veriye erişme şekli, (onunla ne yapacağına bağlı olarak) adresleme modlarının konusudur ve bu bölümde daha sonra ele alınacaktır.

6502'nin komut takımı, nispeten ilkindir ve karmaşık işlemleri içermez. Karmaşık bir makine-düzeyi işlemi çarpma, bölme, kayan noktalı aritmetik veya verilen bir veri değeri için belleğin bir bölgesini araştırmak olabilir. Nispeten karmaşık makine-düzeyi işlemleri içeren birkaç 8-bitlik mikro işlemciden biri de Z80'dir. Tıpkı karmaşık bir bilgisayarın büyük bir grubu, ilkel kapılardan oluşturulabileceği gibi, elbette ki karmaşık bir işlem de bir dizi ilkel makine kodu işlemleriyle gerçekleştirilebilir.

Makine kodu komutlarının beş grubu vardır: veri taşıma, aritmetik, mantıksal, dallanma ve denetim. Veri taşıma, bir veri parçasını bir yerden başka bir yere taşımadan daha karmaşık değildir. Tipik bir programdaki assembly dili komutlarının yüzde 70'inin veri taşıma olduğu belirtilmiştir. Bazı tipik 6502 veri taşıma işlemleri, Tablo 4.15'te verilmiştir.

Tablo 4.15 Bazı tipik 6502 veri taşıma işlemleri

| Assembly biçimi | RTL       | Açıklama                                                      |
|-----------------|-----------|---------------------------------------------------------------|
| LDA M           | [A] ← [M] | M bellek konumunun içeriğini akümülatöre kopyala              |
| STA M           | [M] ← [A] | Akümlatörün içeriğini M bellek konumuna kopyala               |
| LDX M           | [X] ← [M] | M bellek konumunun içeriğini X kaydedicisine kopyala          |
| STX M           | [M] ← [X] | X kaydedicinin içeriğini M bellek alanına kopyala             |
| TAX             | [X] ← [A] | Akümlatörün içeriğini X kaydedicisine kopyala                 |
| TXA             | [A] ← [X] | X kaydedicisinin içeriğini akümülatöre kopyala                |
| PHA             |           | A'yı yığma it -onu, son-giren-ilk çıkar kuyruğuna sok         |
| PLA             |           | A'yı yığımdan çıkar-son-giren-ilk çıkar kuyruğundan geriye al |

X kaydedicisi özel-amaçlı bir kaydedici olup işlevi, daha sonra açıklanacaktır. Burada TAX (Akümülatörü X kaydedicisine aktar) gibi işlemler, kaydediciden-belleğe veri taşımaya (veya tersine) karşılık, kaydediciden-kaydediciye veri taşımayı belirtir. Bazı bilgisayarlar, bellekten-belleğe veri taşımasına izin verirler. PHA (akümülatörü it) işlemi, akümülatörün değerini bir kuyruğa sokar. Bunun özelliği parçaların, kuyruktan kuyruğa sokulduğuyla ters sırada çıkarılmasıdır. Bu türde bir kuyruğa, yığın denir ve tam anlamıyla gelenler-kutusundaki mektup destesi gibi işlem görür. Yani, mektuplar geldiklerinde destenin üstüne eklenir ve alınırken de destenin üstünden alınırlar. PLA, veri parçasını yığımdan çıkarır ve akümülatöre yerleştirir. Yığın uygulamaları bu bölümde ileride, ele alınacaktır. Bir başka veri taşıma komut türü karşılıklı değildir ve 6502'de yer almaz. Karşılıklı değiştirme komutu iki konumun değerini karşılıklı olarak yer değiştirir. Eğer 6502'de EXG X, A olsaydı, etkisi [X] ← [A] [A] ← [X]'i eşzamanlı olarak yerine getirmek olacaktı.

Veri üzerinde etki eden aritmetik işlemler, bu veriler, sanki sayısal değerlermiş gibi davranan işlemlerdir. Daha önce ADC M'nin M bellek konumunun içeriğini akümülatörün içeriğine eklediğini ve sonucu akümülatöre sakladığını görmüştük. Mantık işlemleri ise, bu sözcüğün bitleri üzerinde Boole işlemlerini gerçekleştirir. Tablo 4.16'da belirtilen komutlar 6502'de bulunan bazı aritmetik ve mantıksal komutlardır.

Tablo 4.16'daki bazı komutların, bir bellek konumunun ya da akümülatörün içeriği

Tablo 4.16 Bazı tipik 6502 aritmetik ve mantıksal işlemler

| Assembly biçimi | RTL                                    | Açıklama                                                                                         |
|-----------------|----------------------------------------|--------------------------------------------------------------------------------------------------|
| ADC M           | $[A] \leftarrow [A] + [M] + [C]$       | M bellek konumunun içeriğine elde ile birlikte akümülatöre ekle                                  |
| SBC M           | $[A] \leftarrow [A] + [M] - [\bar{C}]$ | M bellek konumunun içeriğine elde bitinin tümleyeni ile akümülatörden çıkar                      |
| CLC             | $[C] \leftarrow 0$                     | Elde bitini sıfırla                                                                              |
| SEC             | $[C] \leftarrow 1$                     | Elde bitini 1'e kur                                                                              |
| INC M           | $[M] \leftarrow [M] + 1$               | M bellek konumunun içeriğine bir ekle                                                            |
| DEC M           | $[M] \leftarrow [M] - 1$               | M bellek konumunun değerinden bir çıkar                                                          |
| CMP M           | $[A] - [M]$                            | Akümülatörle M bellek konumunun içeriklerini karşılaştır                                         |
| ASL A           |                                        | Akümülatörün (veya M'nin) içeriğini bir basamak sola kaydır (Bu iki ile çarpmaya karşılık gelir) |
| AND M           | $[A] \leftarrow [A] \cdot [M]$         | Akümülatörün bitleri ile M'nin içeriği arasında mantıksal VE işlemi yap                          |
| ORA M           | $[A] \leftarrow [A] + [M]$             | Akümülatörün bitleri ile M'nin değeri arasında mantıksal VEYA işlemi yap                         |
| EOR M           | $[A] \leftarrow [A] \oplus [M]$        | Akümülatörün bitleri ile M'nin değeri arasında mantıksal ÖZEL VEYA işlemi yap                    |
| LSR A           |                                        | A'nın veya M'nin değerini bir basamak sağa kaydır                                                |
| LSR M           |                                        |                                                                                                  |

üzerinde işlem yaptıklarına dikkat ediniz. Örneğin, LSR M, M bellek konumunun içeriği üzerinde mantıksal bir sağa kaydırma yaparken; LSR A komutu da, akümülatörün içeriği üzerinde mantıksal bir sağa kaydırmayı gerçekleştirir. 6502 komutlarının bazıları, sadece bellek konumu içerikleri üzerinde işlem yapmak için kullanılabilir. Örneğin INC (arttır) ve DEC (azalt) komutları akümülatöre uygulanamaz. Bu, 6502'nin küçük eksikliklerinden biridir. Aşağıdaki komutlarla sözde bir (INC A) yaratmak mümkündür:

SEC elde bitini 1'e kur ( $[C] \leftarrow 1$ )  
 ADC #0 akümülatöre sıfır artı eldeyi ekle ( $[A] \leftarrow [A] + 0 + [C]$ )

Dallanma veya atlama komutları komutların yürütülme sırasını etkilerler. 6502'de bir atlama komutu, JMP M, vardır; bu komut işlemcinin bir sonraki işlem olarak M bellek konumundaki komutu yürütmesine neden olur. Daha önce de görmüştük ki 6502, koşul doğruysa işlemcinin dallanmasına neden olan ve koşul yanlışsa sıradaki bir sonraki komutu yürütmesini sağlayan bir grup koşullu dallanma komutlarına sahiptir.

Denetim grubu komutları, (özellikle kesme ve altyordamlarla ilişkili olarak) özel işlevler görürler ve burada ele alınmayacaktır.

## Bazı 6502 Komut Uygulamaları

Bazı 6502 komutlarının gerçek veri üzerindeki etkilerini göstermek üzere, komut dizileriyle birlikte örnek veriler, Tablo 4.17a ve b'de verilmiştir. Sembolik adları NUM1, NUM2 ve NUM3 olan üç veri konumu, işlemlerde yer alan veri için kaynak ve varışyerini oluşturur. Bazı işlemler çiftli olup, NUM3 üretmek için NUM1 ve NUM2 üzerinde çalışırken; diğerleri teklidir ve NUM2 üretmek için NUM1 üzerinde çalışır. Aşağıdaki tüm komutların yaptığı işler kendi kendilerini tanımlasalar da, tüm 6502 komutlarının tam tanımları Ek A2'de verilmiştir.

Sekiz örneğin her birinde; komut dizileri üzerinde çalıştıkları verinin bellek haritasının üzerinde verilmiştir. Buradaki bellek haritalarının, komutların yürütümünden sonraki haritalar olduklarına dikkat ediniz.

Tablo 4.17 (a) Komutların etkileri üzerine örnekler

| 1 Toplama                               | 2 Çıkarma                               | 3 Mantıklı VE                    | 4 Mantık VEYA                    |
|-----------------------------------------|-----------------------------------------|----------------------------------|----------------------------------|
| LDA NUM1<br>CLC<br>ADC NUM2<br>STA NUM3 | LDA NUM1<br>SEC<br>SBC NUM2<br>STA NUM3 | LDA NUM1<br>AND NUM2<br>STA NUM3 | LDA NUM1<br>ORA NUM2<br>STA NUM3 |
| NUM1 01010101                           | NUM1 01010101                           | NUM1 01010101                    | NUM1 01010101                    |
| NUM2 00010111                           | NUM2 00010111                           | NUM2 00010111                    | NUM2 00010111                    |
| NUM3 01101100                           | NUM3 00111110                           | NUM3 00010101                    | NUM3 01010111                    |

Tablo 4.17 (b) Komutların etkileri üzerine örnekler

| 5 Mantık ÖZEL VEYA               | 6 Sola kaydırma               | 7 Sağa kaydırma               | 8 Sola döndürme               |
|----------------------------------|-------------------------------|-------------------------------|-------------------------------|
| LDA NUM1<br>EOR NUM2<br>STA NUM3 | LDA NUM1<br>ASL A<br>STA NUM2 | LDA NUM1<br>LSR A<br>STA NUM2 | LDA NUM1<br>ROL A<br>STA NUM2 |
| NUM1 01010101                    | NUM1 01010101                 | NUM1 01010101                 | NUM1 01010101                 |
| NUM2 00010111                    | NUM2 10101010                 | NUM2 00101010                 | NUM2 10101010                 |
| NUM3 01000010                    |                               |                               |                               |

Örnek 8'de NUM2'deki C, NUM2'nin EDDB'ne kaydırılan psw'nin elde bitini gösterir.

Örnek 8'de; NUM2'deki C, NUM2'nin EDDB'ne kaydırılan PSW'nin elde bitini gösterir.

Diğer birçok işlemci gibi 6502 işlemcisi de, belli işlemlerin ya akümülatörün ya da bir bellek konumunun içeriği üzerinde çalışmasına izin verir. Tablo 4.17'deki 6. ve 8. örneklerde; kaydırma işlemleri akümülatör üzerinde yapılmıştır; böylece okur,

NUM1 işlemlerden önceki, NUM2'de sonraki veri olacak şekilde etkinin, 'önce' ve 'sonra'sını görebilir. Pratikte birçok programda kaydırma işlemleri, örneğin ASL NUM1 gibi, doğrudan bellek konumlarına uygulanır; .

Herkes, günlük yaşantıda sayısal hesaplamalara aşina olduğundan; aritmetik işlemlerin, gerçek uygulamalara nasıl uygulanacağı doğrudan açık bir şekilde anlaşılabilir. Mantıksal işlemlerin uygulaması o denli açık değildir. 3. Bölümde bilgisayara bağlanan aygıtların durumunun, bir grup bit yardımıyla nasıl gösterimlediğini gördük. Verilen örnekler denetim sözcüğündeki belli bir bitin değeri ile açık kapattığımız pompalar, ısıtıcılar ve valflerdi. Mantıksal işlemlerin gerekliliği, bir sözcükteki bitlerin tek tek işlenmesi durumunda ortaya çıkar.

Sekiz tane tek-bitlik girişi (P,Q,R,S,T,U,V,W) ve sekiz de tek-bitlik çıkışı (A,B,C,D,E,F,G,H) olan bir sistemin denetimini ele alalım. Burada giriş/çıkış tekniklerinin ayrıntılarıyla ilgilenmeyeceğiz ve adresi 'GİRİŞ' olan bir bellek konumunu okumakla, P'den W'ya kadar değerlerin akümülatöre yükleneceğini varsayacağız. Benzer şekilde, akümülatörün değerini 'ÇIKIŞ' bellek konumuna yazmanın A'dan H'ye sekiz çıkışı 1'e kurmak etkisine sahip olacağını varsayacağız. Giriş ve çıkış denetim sözcüklerinin formatı aşağıda verilmiştir:

Giriş     PQRSTUW  
Çıkış     ABCDEFGH

Örneğin, aşağıdaki denetleme işlemlerinin yapılacağını varsayalım:

```
IF ((P=1)AND (Q=0)) OR ((P=0)AND(S=1))
THEN
  BEGIN
    C:=1; E:=0
  END
ELSE
  BEGIN
    C:=0; E:=1
  END;
```

Yukarıdaki işlem, GİRİŞ'in üç bitini (P, Q ve S) test etmeye ve sonra da ÇIKIŞ'ın iki bitini (C ve E) 1'e ve 0'a kurmaya ilişkindir. Dikkat edilmesi gereken önemli bir nokta; algorithmada bulunmayan ÇIKIŞ bitlerinin, C ve E bitleriyle yapılan işlemlerden etkilenmemeleridir. Tablo 4.18'de listelenen işlem sırası istenen işi yapacaktır.

Tablo 4.18 Mantıksal işlemlerin kullanılması

|        |     |            |                                        |
|--------|-----|------------|----------------------------------------|
|        | LDA | GIRIS      | Giriş durumuna al                      |
|        | AND | ##11000000 | P ve Q dışındaki tüm bitleri maskeleye |
|        | CMP | ##10000000 | P=1, Q=0 mı test et                    |
|        | BEQ | DOGRU      | Test doğruysa işleme git               |
|        | LDA | GIRIS      | Giriş durumuna tekrar al               |
|        | AND | ##10010000 | P ve S dışındaki tüm bitleri maskeleye |
|        | CMP | ##00010000 | P=0, S=1 mi test et                    |
|        | BEQ | DOGRU      | Test doğruysa işleme git               |
| YANLIS | LDA | CIKIS      | Çıkış denetim sözcüğünü al             |
|        | AND | ##11011111 | C bitini sıfırla                       |
|        | ORA | ##00001000 | E bitini kur                           |
|        | STA | CIKIS      | Yeni çıkış denetim sözcüğünü oluştur   |
|        | JMP | CIK        | Test doğruysa önceki işlemlere dallan  |
| DOGRU  | LDA | CIKIS      | Çıkış denetim sözcüğünü al             |
|        | AND | ##11110111 | E bitini sıfırla                       |
|        | ORA | ##00100000 | C bitini kur                           |
|        | STA | CIKIS      | Yeni çıkış denetim sözcüğünü oluştur   |
|        |     | CIKIS      |                                        |

% assembly dili sembolü, ardından gelen sayının çevirici tarafından ikili bir değer olarak yorumlanacağı anlamına gelir. Bir kez daha belirtmek gerekirse, ikili olarak programlamanın amacı (belli şartlarda) kendinden-açıktır; çünkü (örneğin) AND # % 11000000, okura AND # \$C0 biçimindeki onaltılı işlenenden çok daha anlamlı gelmektedir.

# assembly dili sembolü, (diyez diye okunur) çeviriciye, takip eden değer işleneni içeren bellek konumunun adresi değil, işlenenin kendisi olduğunu söyler. Böylece AND # % 11000000; akümülatörün içeriği ile 11000000 ikili değeri arasında mantıksal VE'leme işlemi yapılması anlamına gelir. Sonuç akümülatöre yerleştirilir. #'nin kullanımına dolaysız adresleme denir ve bu bölümde daha sonra ayrıntılı olarak ele alınacaktır.

CMP işlemi, karşılaştırma yapılmasına sağlar ve PSR'nin bitlerini ona göre kurar. CMP # % 00010000'ün etkisi, akümülatörün içeriği ile 00010000 ikili değerini karşılaştırmaktır. Bu çıkarma ile yapılır (yani [A] - 00010000). Çıkarmanın sonucu dikkate alınmayarak akümülatörün içeriği etkilenmemiş olarak bırakılır. Sadece PSR'nin bitleri değiştirilir. Eğer akümülatörde, 00010000 değeri varsa çıkarma sıfır sonucunu vererek PSR'nin Z (sıfır) bayrağını 1'e kurar ve 'BEQ DOGRU' işlemi, adresin, DOGRU etiketli komuta dallanmasına neden olur.

YANLIS etiketi, kukla bir etikettir ve assembly programınca, herhangi bir şekilde kullanılmaz. Sadece testin sonucu yanlış çıktığında yapılacak işi programcıya

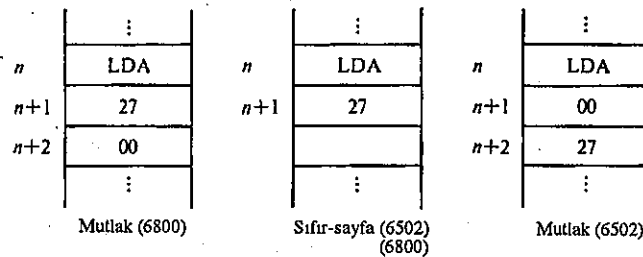
anımsatma görevi vardır. Bu dizinin sonundaki komut 'JMP CİK'tır. Bu bir 'GOTO'ya eşdeğerdedir ve testin sonucu doğru olduğunda, yapılan işlem den bir dallanmaya neden olur.

#### 4.4 Adresleme Modları

Adresleme modu konusu, bir işlenenin adresinin MİB tarafından nasıl hesaplandığıyla ilgilidir. Bu noktaya kadar yalnızca mutlak adresleme yöntemini ele aldık. Mutlak adres, işlenenin gerçek adresidir. Dolayısıyla LDA 1234 'akümülatöre 1234 bellek konumunun içeriğini yükle' anlamına gelir. Bazı bilgisayar üreticileri, bunu doğrudan adresleme olarak adlandırırlar.

16-bitlik adres yolu olan 8-bitlik bir mikroişlemci, bir adresi saklayabilmek için ardışık iki bellek konumuna gerek duyar. Programların büyüklüğünü azaltmak için bazı üreticiler sıfır-sayfa adresleme denen bir adresleme biçimi ortaya attılar. 16-bitlik mutlak bir adres XXYY olarak yazılır. Burada X ve Y'ler 4-bitlik onaltılı karakterleri gösterir. XX'in değeri 00, 01, 02 ... FE, FF aralığında bulunur. İlk değere, yani 00'a, sıfır-sayfa denir. İşlenenin, sıfır-sayfada bulunduğu özel bir adresleme modunun, tanımlandığını düşünelim. Bu durumda bir adresi sadece bu sayfadaki konumu ile belirlemek mümkün olacaktır.

Örneğin, LDA 0027 (mutlak adresleme) LDA 27 (sıfır-sayfa adresleme) şeklinde yazılabilir. Bu komutun büyüklüğünü üç bayttan iki bayta düşürerek bir baytı korumuş olur.



Şekil 4.17 Bir sıfır-sayfa adresleme örneği

Şekil 4.17'de 6502 ve 6800 mikroişlemcilerinin bellek alanından tasarruf yapmak için sıfır-sayfa adreslemeyi nasıl kullandıkları gösterilmiştir (3 baytlık bir komut 2 bayt olur). Bunun sonucu olarak, programcılar 6502 ve 6800'de sıfırıncı sayfayı,

veri saklama için ayırmak eğilimindedir. 6502 ve 6800'ün iki-baytlık mutlak adresleri farklı sırada sakladıklarına dikkat ediniz. 6800, önce adresin üst sıralı 8-bitini (insanların sayı yazmaları biçiminde) saklarken, 6502 önce düşük 8-biti saklar. 6502, 6800'den daha verimlidir. Bunun nedeni, böyle olması gerektiği üzerinde çalışmayı okura bırakıyorum.

Bu aşamada okurun kafasında bir soru belirmiş olmalıdır: MİB, sıfır-sayfa adreslemenin kullanıldığını nereden bilecektir? Cevap şudur: LDA komutu için, kodların biri mutlak adreslemeyi; diğeri de, sıfır-sayfa adreslemeyi göstermek üzere iki farklı kod olmalıdır.

##### Örnekler

|                     |                     |
|---------------------|---------------------|
| 6502 MİB            | 6800 MİB            |
| LDA mutlak= AD      | LDA mutlak= B6      |
| LDA sıfır-sayfa= A5 | LDA sıfır-sayfa= 96 |

Programcı, mutlak/sıfır-sayfa adresleme modunun seçimi konusunda endişe etmemelidir. Çoğu zaman çevirici programı, adresleme modunu otomatik olarak seçer. Böylece LDA 13AE mutlak adresleme moduna zorlarken, LDA 67 sıfır-sayfa adresleme modu ile sonuçlanır. Doğal olarak, bu adresle birlikte mutlak adresleme modunu kullanmak mümkündür. Bunlar da 00 ve 67 olarak saklanacaktır.

Çok yeni 8-bitlik mikroişlemcilerden biri, Motorola 6809, sıfır-sayfa adresleme ile ilgili ilginç bir yaklaşıma sahiptir. O anki sayfa numarasını tutan özel bir sayfa-kaydedicisi vardır. Sıfır-sayfa adreslemesi kullanılırken, sayfa-adresi komutunun işlenene eklenir. Bu şekilde sıfır-sayfanın olası 256 sayfadan biri olması olanaklı kılınırken sıfır-sayfa adreslemenin bellekten-tasarruf avantajı da elde tutulur. Kısım 4.5.1'de 6809'un adresleme modlarıyla ilgili daha fazla ayrıntı verilmiştir.

#### 4.4.1 Dolaysız Adresleme

Bu adresleme modu (kimi zaman gerçek adresleme de denir) bilgisayar ve mikro-ışlemcilerin büyük çoğunluğunda bulunur ve programcıya bir sabiti belirtebilme olanağı sağlar. Yani, komutu izleyen değer, işlenenin adresine bir gönderme değil gerçek işlenenin kendisidir. Bazı assembly dillerinde # sembolü işlenenin başına gelerek dolaysız adreslemeyi belirtir. Bazı çevirici programları, bunun için @ işaretini kullanır. Çeviriciler, çoğu kez bağlı bulundukları kurallarda farklılık gösterirler. Aşağıdaki iki komut, mutlak ve dolaysız adresleme modlarının sırasıyla nasıl gösterildiğini vermektedir:

LDA 1234 [A] ← [1234] demektir, mutlak adresleme  
LDA # 24 [A] ← 24 demektir, dolaysız adresleme

Not 1: 8-bitlik veri sözcüğü olan bir mikroişlemcide 8-bitlik akümülatör sadece 8-bitlik sabit ile yüklenebildiğinden dolaysız adresleme modu 2-bayt komut formatı kullanır.

Not 2: # sembolünün komutun bir parçası olmadığını anımsayınız. Bu, çeviriciye 'akümülatöre yükle' için dolaysız adresleme modunu kullanan bir kod seçmesini söyleyen bir mesajdır. 6502'de bu durumda kod A9'dur. # sembolünü % veya % sembolleri ile karıştırmayınız. \$ sadece takip eden sayının onaltılı olduğunu gösterir. % ise izleyen sayının ikili olduğunu belirtir. Bu iki sembolün kullanılması nedeni pekçok bilgisayarın indisleri ele alamaması ve tabanı indis ile etiketleyen klasik biçimin pratik olmamasıdır. Örneğin LDA # 25, LDA # \$ 19 ve LDA # % 00011001 aynı etkiye sahiptir.

#### Dolaysız Adreslemenin Kullanımına Örnekler

Dolaysız adresleme modu bir komutta yer alan işlenenin değerinin programın yazıldığı sırada bilindiği durumda kullanılır. Yani, değişkenlere karşılık sabitler üzerinde de işlem yapabilmek üzere kullanılır. Bu adresleme modu, komut sadece bir bellek referansına gereksinim duyduğundan, mutlak adresleme modundan daha hızlı olma avantajına sahiptir. LDA # 5 işlemi, bellekten, bir getirme fazında okunduğunda, işlenen 5, gerçek işleneni okumak üzere 5 konumuna bir başka erişime gerek duymadan hemen kullanılabilir durumdadır. Dolaysız adreslemenin bazı uygulamaları aşağıda verilmiştir:

##### 1. Aritmetik sabite olarak

LDA NUM  
CLC  
ADC #22  
STA NUM

Bu işlem dizisi, NUM konumundaki veriye 22 eklenmesine neden olacaktır. Yani  $[NUM] \leftarrow [NUM] + 22$ .

2. Bir sabit ile karşılaştırmalarda. Bir NUM değişkeninin  $7 < NUM < 25$  (Tablo 4.19) aralığında bulunup bulunmadığını belirlemek için bir testi gözönüne alalım.

3. Döngü yapılarını sonlandırma yöntemi olarak. Tipik bir döngü yapısı, hem

BASIC hem de Pascal için Tablo 4.20'de verilmiştir.

Tablo 4.19 Bir dolaysız adresleme örneği

|        |        |                        |
|--------|--------|------------------------|
| LDA    | NUM    |                        |
| CMP    | # 8    | 8'le karşılaştır       |
| BMI    | YANLIS | negatifse $NUM \leq F$ |
| CMP    | # 25   | 25'le karşılaştır      |
| BPL    | YANLIS | pozitifse $NUM > 24$   |
| DOGRU  | ...    |                        |
| YANLIS | ...    |                        |

Tablo 4.20'deki yapı, 6502 assembly diline kolayca çevrilebilir. Döngü sayıcısını, I bellek konumunda saklanmaktadır (Tablo 4.21).

Tablo 4.20 Bir döngü yapısı

| BASIC             | Pascal             |
|-------------------|--------------------|
| 10 FOR I = 1 TO N | FOR I := 1 TO N DO |
| .                 | BEGIN              |
| .                 | .                  |
| 50 NEXT I         | END;               |

Tablo 4.21 Döngünün assembly dilinde gerçekleştirilmesi

|      |         |                                             |
|------|---------|---------------------------------------------|
| LDA  | # 1     | akümülatörü I'nın başlangıç değeriyle yükle |
| STA  | I       | döngü sayıcısını ayarla                     |
| SONR | ...     | döngünün başlangıcı                         |
| .    | .       | döngü gövdesi                               |
| .    | .       | .                                           |
| INC  | I       | bellekteki sayıcıyı artır                   |
| LDA  | I       | sayacı akümülatöre al                       |
| CMP  | # N + 1 | döngü testi sonu                            |
| BNE  | SONRA   | bitmemişse döngüyü tekrarla                 |

Döngünün sonunda sayıcı; I konumunun içeriğinin okunmasına, artırılmasına ve tekrar belleğe konulmasına sebep olan INC I komutu ile artırılır. I'nın N+1 ile karşılaştırılabilmesi için, bu değer, bellekten LDA I komutu ile getirilmesi gerekir. Karşılaştırmanın N+1 ile yapıldığına dikkat ediniz. Bunun nedeni sayıcının test edilmeden önce artırılmasıdır. Döngüdeki son turda  $I=N$ , artırmadan sonra ise  $N+1$  olur ve SONRA altıyordamına dallanma yapılmayarak döngüden çıkılması sağlanır.

Aynı döngü yapısı, -test, I değerinin artırılmasından önce yapılacak biçimde- Tablo 4.22'de gösterildiği gibi de yazılabilir. Bu tabloda verilen teknik, bir öncekine göre fazladan bir komut gerektirmektedir.

Tablo 4.22 Alternatif bir döngü yapısı

|      |     |      |                           |
|------|-----|------|---------------------------|
|      | LDA | # 1  |                           |
|      | STA | I    |                           |
| SONR | ... |      |                           |
|      |     |      |                           |
|      |     |      |                           |
|      | LDA | I    | sayıcıyı al               |
|      | CMP | # N  | bu son tur mu             |
|      | BEQ | CIK  | öyleyse döngüden çık      |
|      | INC | I    | bellekteki sayıcıyı artır |
|      | JMP | SONR | döngünün başına geri dön  |
| CIK  | ... |      |                           |

Pratikte, döngülerin gerçekleştirilmesine ilişkin yöntemler normalde kullanılmaz, çünkü 6502 ve aslında diğer tüm işlemciler, döngü sayıcısı olarak kullanılabilen bir veya daha çok özel-amaçlı indeks kaydedicisi içerirler. Daha önce veri taşıma komutları konusunda X kaydedicisi ile kısaca tanışmıştık. 6502'nin X kaydedicisi 8-bitlik bir kaydedici olup INX ile arttırılabilir, DEX ile de azaltılabilir. Artık döngüyü Tablo 4.23'te verilen biçimde yeniden yazabiliriz.

Tablo 4.23 Bir döngünün gerçekleştirilmesinde X kaydedicisinin kullanımı

|      |     |         |                      |
|------|-----|---------|----------------------|
|      | LDX | # 1     | sayıcıyı 1'e kur     |
| SONR | ... |         |                      |
|      |     |         |                      |
|      |     |         |                      |
|      | INX |         | sayıcıyı artır       |
|      | CPX | # N + 1 | döngünün N. turu mu? |
|      | BNE | SONR    | değilse tekrarla     |

Bu yöntem, sayıcı olarak bir bellek konumunu kullanmaktan daha şık bir çözümdür. Bu işlem geriye sıfıra kadar sayarak ve döngüyü sonlandırmak için bir BNE (sıfır değilse dallan) kullanarak bir adım ileri götürülebilir. Bu ise, dışardan bir karşılaştırma komutunun gerekliliğini ortadan kaldırır. Tablo 4.24'teki iki örneği ele alalım.

Tablo 4.24 Bir döngünün aşağıya doğru sayıyla gerçekleştirilmesi

| Pascal                  | 6502 assembly dili |                      |
|-------------------------|--------------------|----------------------|
| FOR I := 7 DOWN TO 1 DO | LDX # 7            | sayıcıyı kur         |
| BEGIN                   | SONR               | ...                  |
| .                       | .                  |                      |
| .                       | .                  |                      |
| .                       | .                  |                      |
| END;                    | DEX                | sayıcıyı kur         |
|                         | BNE SONR           | sıfıra kadar döngüle |

#### 4.4.2 İndekslenmiş Adresleme

İndekslenmiş adreslemenin nasıl çalıştığını tanımlamadan önce, onsuz hayatın programcı için çok zor olacağını göstermekte yarar vardır. Üç-beş sayıyı nasıl toplayacağımızı hepimiz biliriz. Şimdi ardışık konumlarda saklı yüz tane sayıyı toplamayı hayal edin.

```
CLC
LDA NUM1
ADC NUM2
ADC NUM3
.
.
.
ADC NUM99
ADC NUM100
```

Açıktır görüldüğü gibi bu problemin, daha kolay bir çözümü olmalıdır. Bu güçlükten kurtulmanın bir yolu kendini-değiştiren programa başvurmaktır. Kendini-değiştiren program kullanımının son derece kötü bir tecrübe olduğunu ve hiç bir programcının hiç bir zaman başvurmaması gereken bir şey olduğunu belirtmeliyim. Hatta bu, ziyafette yanlış bıçak kullanmaktan bile daha kötü bir şeydir.

Tablo 4.25'de kendini-değiştiren bir kod örneği verilmiştir.

Bu programın anahtarı iki komuttur: 'DONGU ADC NUM2' ve 'INC DONGU+1'. İlk komut NUM2 bellek konumunun içeriğini akümülatörün içeriğine ekler. İzleyen komut DONGU + 1, bellek konumunun içeriğini; yani NUM2 işlenenin adresini arttırır. Bu şekilde NUM2, 99 kez NUM1'e eklenir fakat her seferinde elimizde, INC DONGU + 1 işlemi ile değiştirilmiş bulunan farklı bir NUM2

Tablo 4.25 Kendini-değiştiren bir program parçasının kullanımı

|       | NAM | SELFMOD                                                                        |
|-------|-----|--------------------------------------------------------------------------------|
| SAYI  | EQU | \$00                                                                           |
| NUM1  | EQU | \$10                                                                           |
| NUM2  | EQU | \$11                                                                           |
|       | ORG | \$0080                                                                         |
|       | LDA | #99 [A] ← 99                                                                   |
|       | STA | SAYI [SAYI] ← 99                                                               |
|       | CLC | [C] ← 0                                                                        |
|       | LDA | NUM1 [A] ← [NUM1]                                                              |
| DONGU | ADC | NUM2 [A] ← [A] + [NUM2] + [C]                                                  |
|       | INC | DONGU + 1 [DONGU + 1] ← [DONGU + 1] + 1                                        |
|       | DEC | SAYI [SAYI] ← [SAYI] - 1                                                       |
|       | BNE | DONGU EĞER Z = 0 İSE [PC] ← DONGU<br>(yani sıfırdan-farklı ise DONGU'ye dallen |

100 sayıyı toplayan programın sonu)

bulunmaktadır. Şekil 4.18'de 6502 makine kodunda yazılmış ve yukarıdaki programı gerçekleştirecek sistemin bellek haritası görülmektedir.

Program, \$0080 - \$008E arasındaki bellek konumlarına yerleşmiştir. Döngü sayıcısı \$0000'e yerleştirilir ve 100 tane sayı ise \$0010 - \$0073 aralığında saklanır. Artık INC DONGU + 1 komutu (E6 88), işaretçiyi NUM2'yi artırmaya zorlar (başlangıçta 11). Böylece eklenecek bir sonraki sayının \$ 0012 konumunda olacağı daha anlaşılır hale gelmiş olmalıdır. Ayrıca BNE DONGU komutunu dallanma adresinin F8 olduğuna dikkat ediniz. Bu nokta, bağıl adreslemeye geldiğimizde açıklanacaktır.

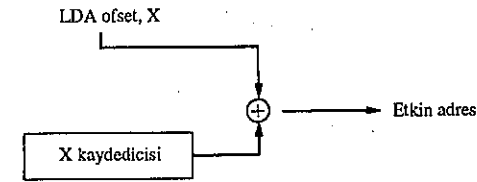
Kendini-değiştiren koda itiraz edilmesinin, birkaç nedeni vardır. Kendini-değiştiren kod (belli sebeplerden dolayı) kalıcı sadece-okunur belleklere (ROM) yerleştirilemez. Daha da önemlisi, kendini-değiştiren programdaki bir hata, büyük bir olasılıkla bir felakete neden olacaktır. Eğer kazara (bir işlenenin yerine) işlem-kodu değiştirilmişse, program anlamsız bir şekilde (yani çılgınca) çalışacaktır. Öte yandan kendini-değiştiren kod kullanımına ilişkin diğer bir itiraz da, kullanıcı açısından hemen hemen tümüyle anlaşılabilir görünmesidir. Kendini-değiştiren koda dayalı olarak yazılmış bir programı, yazarı dışındaki kişilerin anlaması aşırı derecede zor olacaktır, hatta belki yazarı bile anlamakta zorluk çekebilecektir.

İndekslenmiş adresleme yukarıdaki soruna zekice bir çözüm getirir. İndekslenmiş adreslemede bir işlenenin etkin adresi (EA), indeks kaydedici içeriği ile komuttaki

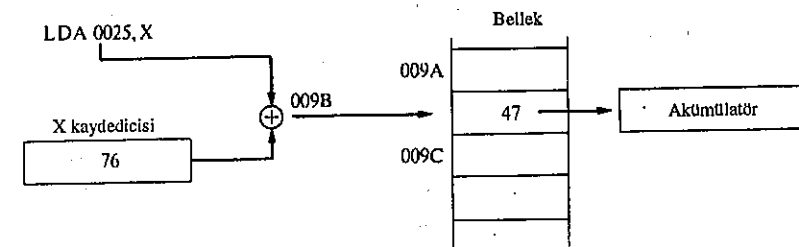
|                  |    |                                                |
|------------------|----|------------------------------------------------|
| 0080             | A9 | LDA #99 (63 <sub>16</sub> → 99 <sub>10</sub> ) |
| 0081             | 63 |                                                |
| 0082             | 85 | STA SAYI (SAYI=00)                             |
| 0083             | 00 |                                                |
| 0084             | 18 | CLC                                            |
| 0085             | A5 | LDA 10 (sayıların başlangıcı 1'dur)            |
| 0086             | 10 |                                                |
| DÖNGÜ → 0087     | 65 | ADC 11 (add NUM 2)                             |
| DÖNGÜ + 1 → 0088 | 11 |                                                |
| 0089             | E6 | INC 88                                         |
| 008A             | 88 |                                                |
| 008B             | C6 | DEC SAYIM                                      |
| 008C             | 00 |                                                |
| 008D             | D0 | BNE DONGU yani BNE 0087                        |
| 008E             | F8 |                                                |

Şekil 4.18 Kendini-değiştiren bir programın bellek haritası

offsetin toplamı olarak verilir. Offset terimi, opkodu izleyen sayıya karşılık gelir. Şekil 4.19(a)'da etkin adresin nasıl hesaplandığı gösterilmiştir.



Şekil 4.19 (a) İndekslenmiş adresin gösterimi



Şekil 4.19 (b) LDA 0025, X komutunun yürütülmesi

LDA 0025, X komutunu ele alalım. Burada X kaydedicisinin içerdiği değer 76 olsun (Şekil 4.19(b)). İşlenenin etkin adresi \$009B'dir. RTL anlamında LDA OFFSET, X komutu [OFFSET + [X]] şeklinde yazılır.

6800 MIB'inde, 8-bitlik offset ve 16-bitlik indeks-kaydedicisi varken; 6502, 16-bitlik offset ve sekiz-bitlik indeks kaydedicisine sahiptir. 6502 aynı zamanda, 8-bitlik

ofsetli bir sıfır-sayfa indeks moduna da sahiptir. 6502'de biri X, diğeri de Y olarak adlandırılan iki indeks kaydedicisi vardır. Bu olanak çok kullanışlıdır ve 6800'un bir indeks kaydedicisine karşılık bir gelişmişliği gösterimler. Yeni 6809 bir adım daha ileridedir ve iki tane 16-bitlik indeks kaydedicisine sahiptir.

6502, doğrudan indeks kaydedicisi üzerinde işlem yapabilecek bazı özel komutlara sahiptir. Bunlar, Tablo 4.26'da verilmiştir.

Tablo 4.26 6502'nin indeks kaydedicilerini etkileyen işlemler

|     |                                                   |
|-----|---------------------------------------------------|
| CPX | X kaydedicilerinin içeriğini bellekle karşılaştır |
| CPY | Y kaydedicilerinin içeriğini bellekle karşılaştır |
| DEX | X kaydedicilerinin içeriğini 1 azalt              |
| DEY | Y kaydedicilerinin içeriğini 1 azalt              |
| INX | X kaydedicilerinin içeriğini 1 artır              |
| INY | Y kaydedicilerinin içeriğini 1 artır              |
| LDX | X indeks kaydedicisini bellekten yükle            |
| LDY | Y indeks kaydedicisini bellekten yükle            |
| STX | X kaydedicisinin içeriğini belleğe sakla          |
| STY | Y kaydedicisinin içeriğini belleğe sakla          |
| TAX | Akümülatörü X kaydedicisine aktar                 |
| TAY | Akümülatörü Y kaydedicisine aktar                 |
| TXA | X kaydedicisini akümülatöre aktar                 |
| TYA | Y kaydedicisini akümülatöre aktar                 |
| TSX | Yığın işaretçisini X indeks kaydedicisine aktar   |
| TXS | X indeks kaydedicisini yığın işaretçisine aktar   |

Bu tablodaki tüm işlemler, yalnızca X kaydedicisi için geçerli olan son iki işlem dışında, X ve Y için simetrik. Asimetrik komut takımlarına pek iyi gözle bakmam. Çünkü programcı, bir kaydedici için bir kural grubunu, bir diğeri için de farklı bir kural grubunu anımsamak zorunda kalmaktadır.

İndekslenmiş adreslemeyi tanımladıktan sonra bunu, yüz sayının toplanmasına uygulayalım (Tablo 4.27).

Tablo 4.27 İndekslenmiş adreslemenin kullanılması

|       |     |         |                                |
|-------|-----|---------|--------------------------------|
|       | LDX | #0      | [X] ← 0                        |
|       | CLC |         | [C] ← 0                        |
|       | LDA | NUM1    | [A] ← [NUM1]                   |
| DONGU | ADC | NUM2, X | [A] ← [A] + [NUM2 + [X]] + [C] |
|       | INX |         | [X] ← [X] + 1                  |
|       | CPX | #99     | [X] - 99                       |
|       | BNE | DONGU   | EĞER Z=0 İSE [PC] ← DONGU      |

Bu örnekte; akümülatörün içeriğine, NUM2 ile verilen bellek konumunun içeriğini artı X yazmacındaki sayıyı ekleriz. Başlangıçta [X]=0'dır. Böylece NUM2 ilk işlenen olur. Döngü ikinci kez tekrarlandığında X yazmacındaki değer 1 olur ve NUM2+1 konumundaki (yani NUM3) değer alınır.

### Bir İndekslenmiş Adresleme Örneği

Tüm hesaplamalar içinde en yaygın olanlardan biri, (çünkü birçok farklı alanlarda beklenmedik şekilde ortaya çıkar) iki vektörün birbiriyle iç (skalar) çarpımına ilişkin hesaplamadır. Sözelimi A ve B iki tane n-bileşenli vektördür. A ve B'nin birbiriyle çarpımı S, şu şekilde ifade edilir.

$$S = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \dots + a_n b_n.$$

A ve B'nin 8-bitlik tamsayı olduğu durumu ele alalım. Aşağıda bir 'MUL M' işleminin olduğunu varsayacağız. Bu işlem akümülatörün içeriği ile M bellek alanının içeriğini çarpıp sonucu akümülatöre yerleştirir. Yani [A] ← [A] x [M]; Tablo 4.27'ye bakınız. 6502, 8080 ve Z80 mikroişlemcilerinde çarpma komutu bulunmaz.

Tablo 4.27 İç çarpımı bulmak için X kaydedicisinin kullanılması

|       | NAM | INNER   | Programın adı               |
|-------|-----|---------|-----------------------------|
|       | ORG | \$0000  | Verinin başlangıcı          |
| S     | RMB | 2       | Çarpım için iki bayt ayır   |
| VEK1  | EQU | \$0050  | Vektör 1'de ilk konum       |
| VEK2  | EQU | \$0100  | Vektör 2'de ilk konum       |
| N     | EQU | \$20    | 32 Bileşen (n=32)           |
|       | ORG | #0200   | Programın başlangıcı        |
|       | LDA | #\$00   | [A] ← 0                     |
|       | STA | S       | [S] ← 0                     |
|       | STA | S + 1   | [S + 1] ← 0                 |
|       | LDX | #\$00   | [X] ← 0                     |
| DONGU | LDA | VEK1, X | [A] ← [[X] + VEK1]          |
|       | MUL | VEK2, X | [A] ← [A] * [[X] + VEK2]    |
|       | CLC |         | [C] ← 0                     |
|       | ADC | S + 1   | [A] ← [A] + [S + 1] + [C]   |
|       | STA | S + 1   | [S + 1] ← [A]               |
|       | LDA | S       | [A] ← [S]                   |
|       | ADC | #\$00   | [A] ← [A] + 0 + [C]         |
|       | STA | S       | [S] ← [A]                   |
|       | INX |         | [X] ← [X] + 1               |
|       | CPX | #N      | [X] - N                     |
|       | BNE | DONGU   | EĞER Z = 0 İSE [PC] ← DONGU |



|         |        |           |            |         |
|---------|--------|-----------|------------|---------|
| NAM     | MOVE   | 0200      | A2         | LDX #00 |
| ORG     | \$0200 |           |            |         |
| BURADAN | EQU    | \$1800    | 0201       | 00      |
| BURAYA  | EQU    | \$8300    | DÖNGÜ 0202 | BD      |
| NUM     | EQU    | \$10      |            |         |
|         | LDX    | #S00      | 0203       | 00      |
| DONGU   | LDA    | BURADAN,X | 0204       | 18      |
|         | STA    | BURAYA, X | 0205       | 90      |
|         | INX    |           |            |         |
|         | CPX    | #NUM      | 0206       | 00      |
|         | BNE    | DONGU     | 0207       | 83      |
|         |        |           | 0208       | E8      |
|         |        |           | 0209       | E0      |
|         |        |           | 020A       | 10      |
|         |        |           | 020B       | D0      |
|         |        |           | 020C       | F5      |
|         |        |           | 020D       |         |

Şekil 4.21 Bağlı adreslemeye bir örnek

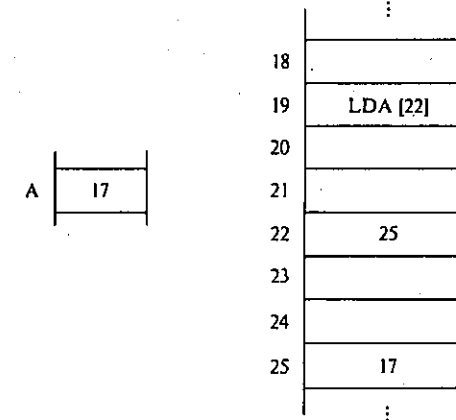
Bir veri bloğu, belleğin bir bölgesinden diğerine taşınacaktır. Taşınacak bloğun ilk konumu 'BURADAN'dır ve hedefin ilk konumu da 'BURAYA'dır. Taşınacak sözcük sayısı NUM ile verilmiştir.

Programcı, normalde bağlı dallanma ofsetlerinin hesaplanması ile ilgili endişelenmemelidir - assembler bu işleri otomatik olarak yapacaktır. BNE DONGU komutu, kurulan PSR'deki sıfır bit durumunda geriye, LDA BURADAN,X komutuna bir dallanmaya neden olur. Bellek haritasından gördüğümüz gibi dallanma işleminin adresi \$020B ve LDA BURADAN,X işleminin adresi de \$0202'dir. Dolayısıyla 'BNE'nin başlangıcından itibaren 9 konum ya da bitiminden itibaren 11 konum atlamalıyız. MİB, bir dallanmanın başında PC'yi daima 2 arttırdığından, saklanan ofset -11 veya -\$B'dir - ikiye tümleyen biçiminde ise \$F5'tir.

#### 4.4.4 Dolaylı Adresleme

Mutlak veya doğrudan adreslemede, komutun istediği işlenen adresi, opkodu izleyen adres tarafından verilir. Bu adres, işlenenin gerçek adresidir. Dolaylı adreslemede işlenenin gerçek adresi, opkodu izleyen adresin gösterdiği bellek

konumunun içeriği ile verilir. Bir başka deyişle komut, veri adresinin adresini vermektedir. Bu nokta, Şekil 4.22'de LDA [22] komutu ile belirtmiştir. Köşeli parantez genelde dolaylı adres modunu göstermek için kullanılır.



Şekil 4.22 Dolaylı adresleme

LDA [22] komutu, işlemciyi, işlenen adresini (25) almak için 22 konumunu okumaya zorlar. 25. konumunda 17 sayısı vardır ki bu akümülatöre yüklenen gerçek değerdir.

Dolaylı adresleme, dolaysız adresleme ile başlayan ilerlemenin bir aşaması olarak düşünülebilir. Tablo 4.28'de dolaysız adresleme, mutlak adresleme ve dolaylı adreslemenin RTL tanımları verilmiştir. Ayrıca işlemin sonsuza kadar götürülebileceğini göstermek için 'dolaylı dolaylı adreslemeyi' de ekledim.

Tablo 4.28 Adresleme modları

| Adresleme modu  | Assembly dili biçimi | RTL biçimi        |
|-----------------|----------------------|-------------------|
| Dolaysız        | LDA # değer          | [A] ← DEĞER       |
| Mutlak          | LDA adres            | [A] ← [ADRES]     |
| Dolaylı         | LDA [adres]          | [A] ← [[ADRES]]   |
| Dolaylı dolaylı | LDA [[adres]]        | [A] ← [[[ADRES]]] |

Dolaylı adreslemenin bu tanımından; indekslenmiş adreslemenin, aslında dolaylı adreslemenin bir başka biçimi olduğu görülmelidir. Çünkü indeks kaydedicisi, işlenenin adresini içerir. Gerçekte, dolaylı adresleme tüm mikroişlemcilerde bulunmaz çünkü eğer indekslenmiş adresleme mevcutsa onun da bulunması mutlaka gerekmemektedir. Elbette ki dolaylı adresleme programcıya neredeyse sınırsız sayıda indeks kaydedicisi olanağı sağlamaktadır. 6502'de, Ek A2'de açıklandığı üzere, dolaylı adreslemenin sınırlı bir biçimi vardır.

Dolaylı adresleme, adreslerin hesaplanmasının program yürütülürken çalışma-zamanı içinde yapılmasını sağlar. Tablo 4.29'daki program, 100 tane sayının toplanması için dolaylı adreslemenin nasıl kullanıldığını göstermektedir. Bu programın 6502'de çalıştırılmayacağını anımsayın. Çünkü dolaylı adreslemenin genel biçimi burada mevcut değildir.

Tablo 4.29 Bir Dolaylı adresleme uygulaması

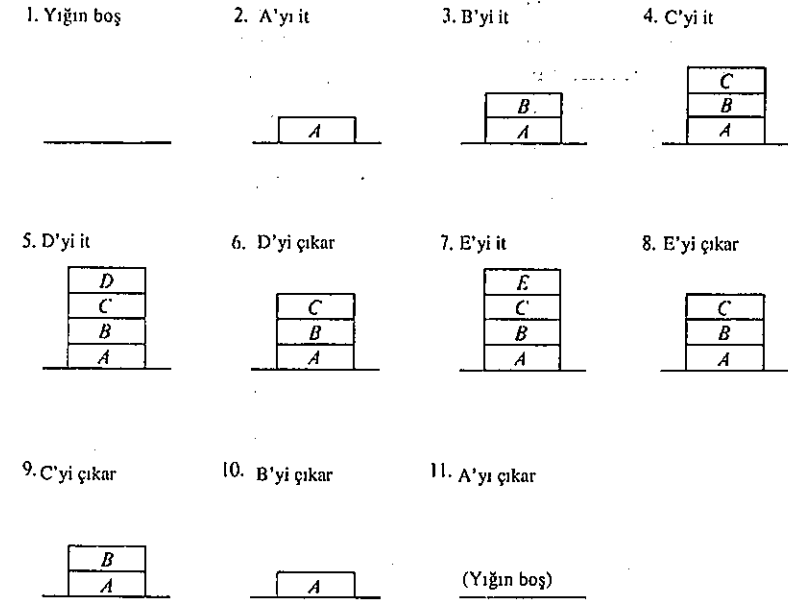
|          | NAM | ORNEK      |                                         |
|----------|-----|------------|-----------------------------------------|
|          | ORG | \$0000     | verinin başlangıç noktası               |
| NUMARA   | RMB | 100        | 100 konum ayır                          |
| ISARETCI | RMB | 1          | sayı konumları işaretçisi               |
|          | ORG | \$0200     | programın başlangıç noktası             |
|          | LDA | \$99       | sayıcıyı kur                            |
|          | STA | ISARETCI   | işaretçi 100. üçüncü sayıyı gösteriyor. |
|          | CLC |            | elde-bitini sıfırla                     |
|          | LDA | #00        | toplamı sıfırla                         |
| SONRAKI  | ADC | [ISARETCI] | işaretlenen sayıyı ekle                 |
|          | DEC | ISARETCI   | işaretçiyi azalt                        |
|          | BPL | SONRAKI    | tümü eklenene dek devam et              |

Bu program sadece dolaylı adresleme uygulamasını belirtme amacıyla konulmuştur. Sözcük-uzunluğunun, sayıların toplanmasında taşma olmadan işleme sokulabilecek yeterlilikte olduğunu varsayalım. Bu programda olup da 6502'de mevcut olmayan tek komut ADC [ISARETCI]'dir.

Bu programda ISARETCI olarak adlandırılan değişken, akümülatördeki toplama eklenecek sayının adresini içerir. Başlangıçta, ISARETCI 99'a yani son sayının adresine kurulur. Her bir sayı ADC [ISARETCI] ile eklendikçe ISARETCI'nin değeri bir azaltılır. ISARETCI sıfıra ulaştığında toplama tamamlanmış olur. Assembly dili programlamada geriye doğru, N'den 0'a sayarak döngüyü sonlandırmak için sıfır testi kullanmanın yaygın olarak uygulandığını anımsayın.

#### 4.4.5 Yığın

Yığın; nesnelerin girilme sırasının tersi doğrultuda çıkarıldığı özel türde bir veri yapısıdır. Bu nedenle yığına sık sık LIFO (son-giren-ilk-çıkart) kuyruğu denir. Şekil 4.23'te nesnelerin eklenip çıkarılması şeklindeki yığın işlemlerini gösteren bir dizi diyagram verilmiştir.



Şekil 4.23 Yığının işleyişi

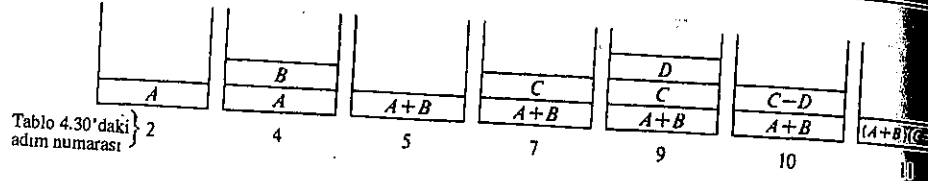
Görüldüğü gibi yığın, nesne eklendikçe büyür, çıkarıldıkça küçülür. Yığının sadece bir ucu olduğuna ve nesnelerin daima bu noktadan eklenip, çıkarıldığına dikkat ediniz. Bu özel konuma yığının tepesi denir (TOS). Bir alttaki konum ise yığının bir sonraki (NOS) olarak anılır. Yığına bir nesne eklendiğinde, buna yığının itilmesi ve yığından çıkarıldığında da, yığından çekilmesi şeklinde ifade edilir.

Yığınlar, bilgisayar biliminde, içinden değer çıkarmak için kullanılan en yararlı aygıtlardan biridir ve birçok farklı alanlarda uygulamaları vardır. Veriyi bellekten yığına taşıyabilen (veya tersini yapabilen), yığının tepesindeki nesne ile işlem yapabilen (tekli çalışan) veya yığının tepesindeki iki nesne ile işlem yapabilen (çiftli çalışan) bir bilgisayarı düşünelim. Çiftli bir işlem meydana geldiğinde (artı, çarpı, VE, VEYA vs.) yığının tepesinden iki nesne çıkartılır ve işlemin sonucu,

çıkan bu değerlerin yerini alır. Tablo 4.30'da bu nokta, hayali yığın-tabanlı sayarda,  $(A+B) (C-D)$  ifadesinin hesaplanması ile gösterilmiştir. Şekil 4.24 işlem süreci sırasında yığının çeşitli aşamalarındaki durumu gösterilmiştir. 4.24'teki her bir diyagramın altındaki sayı Tablo 4.30'daki programın satırına karşılık gelir.

Tablo 4.30 Hayali yığın-tabanlı bilgisayarda işlemler

|     |       |                                                                            |
|-----|-------|----------------------------------------------------------------------------|
| 1.  | LDA A | A'yı al                                                                    |
| 2.  | PUSH  | yığına it                                                                  |
| 3.  | LDA B | B'yı al                                                                    |
| 4.  | PUSH  | yığına it                                                                  |
| 5.  | ADD   | tepedeki iki değeri yığından çıkar, topla ve sonucu yığına it              |
| 6.  | LDA C | C'yı al                                                                    |
| 7.  | PUSH  | yığına it                                                                  |
| 8.  | LDA D | D'yı al                                                                    |
| 9.  | PUSH  | yığına it                                                                  |
| 10. | SUB   | tepedeki iki değeri yığından çıkar, çıkarma işlemi yap ve sonucu yığına it |
| 11. | MULT  | tepedeki iki değeri yığından çıkar, birbiriyle çarp ve sonucu yığına it    |



Şekil 4.24 Aritmetik ifadelerin hesaplanmasında yığının kullanımı

Bir yığını gerçekleştirmenin birçok yolu vardır - bunun için bazı bilgisayarlarda özel-amaçlı donanım kullanılırken, diğerlerinde de bellekte bir yerde bir yığın modeli oluşturmak için klasik programlama teknikleri kullanılır. Popüler mikroislemcilerin birçoğu iki tekniğin karışımını kullanır. Yığını sistem belleğinde yerleştirir, fakat yığına eklenen veya yığından çıkarılan değerlerin kaydını tutmak için özel bir kaydedici kullanırlar. Bu tür bir kaydediciye yığın işaretçisi (SP) denir.

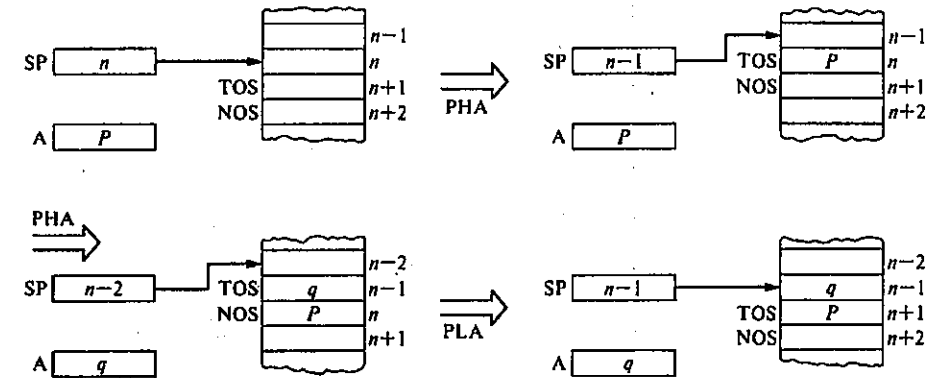
### 6502 Yığını

Pekçok 8-bitlik mikroislemci yığınlarını 65 K'lık normal bellek alanının herhangi bir yerine yerleştirilmesine olanak tanıyan 16-bitlik bir yığın işaretçisine sahiptir. 6502'de 8-bitlik bir yığın işaretçisi vardır ve bu da yığının büyüklüğünü 256 baytla sınırlar - \$0100 - \$01FF (yani 1. sayfa) bölgesi 6502 MİB'i tarafından yığına ayrılmıştır. Bunun sonucu olarak, programcı belleğin bu bölgesinden kaçınmalıdır. Doğal olarak, eğer yığın mekanizması kullanılmıyorsa (ki bu az rastlanır bir

durumdur), 1. sayfayı belleğin herhangi bir bölgesi gibi ele almak mümkündür.

Bazı (donanım) yığınlarında, yığına yeni verinin her eklenişinde halihazırda yığında olan tüm değerler aşağı doğru itilir (itme terimi bu nedenle kullanılır). Benzer şekilde, yığından bir değer çıkarıldığında NOS, TOS olur ve tüm değerler yukarı çekilir. Yığının bir mikroislemci tarafından gerçekleştirilmesi halinde, yığındaki değerler kendi kendilerine hareket etmezler, ancak yığının tepesindeki işaretçi yığın giderek büyüyüp küçüldükçe değiştirilir. Bazı uygulamalarda yığın işaretçisi, yığındaki ilk boş konumu gösterir. Bazılarında ise işaretçi, yığının o anki tepesini belirtir. Aşağıda 6502 bir örnek olarak kullanılmıştır:

Akümülatörü yığına itmek için assembly dilindeki işlem PHA'dır. Bir değeri yığından çıkarıp akümülatöre yerleştirmeye karşılık gelen komut kısaltması, PLA'dır. Şekil 4.25'te PHA ve PLA işlemlerinin 6502 yığınınındaki etkileri gösterilmiştir. 6502'de yığın, parçalar eklendikçe bellekte geriye doğru büyür. Yani, eğer yığın işaretçisi, \$014C değerini içeriyorsa ve yığına bir değer itilmişse, yığın işaretçisinin yeni değeri \$014B olacaktır. Yığının - belleğin alt veya üst sırasına doğru - büyüme biçimi, tümüyle konumuzun dışındadır.



Şekil 4.25 6502'nin yığın yapısı.

6502 için, PHA (itme) şu şekilde tanımlanır:

$$[MS[SP]] \leftarrow [A], [SP] \leftarrow [SP] - 1$$

PLA (çekme) ise şöyle tanımlanır:

$$[SP] \leftarrow [SP] + 1, [A] \leftarrow [MS[SP]]$$

Yığın işaretçisinin bir itmeden sonra azaltıldığına, çıkarmadan önce de artırıldığına dikkat ediniz. Bunun nedeni TOS'un daima, üstündeki ilk boş konumu göstermesidir. [MS[SP]] notasyonu, adresi yığın işaretçisinin içeriği ile verilen belleğin içeriği anlamına gelir. MS bilindiğinden bu ifade [[SP]]'ye kısaltılabilir.

PLA işlemlerinden sonra yığın aşağı doğru büyüdüğünde, yığındaki değerler fiziksel olarak silinmez, bir PHA işlemi ile üzerine yazılana kadar bellekte kalırlar.

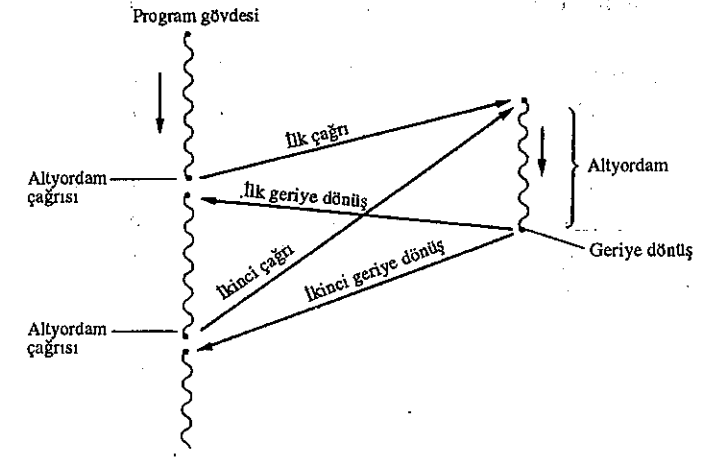
Yığının bilgisayar biliminde birçok uygulaması vardır. Bazıları biraz özel uygulamalardır ve cebirsel ifadelerin işlenip hesaplanmalarını kapsar. Bu kullanımlar bu metnin konusu dışındadır. Yığının, geçici veri saklama ortamı olarak daha sıkı bir kullanımı da vardır. Bir PHA komutunun yürütülmesi, akümülatörün içeriğini yığına saklanmasına, bir PLA komutunun yürütülmesi de, akümülatörün içeriğini geriye döndürür. Bu, verinin dış bellek konumlarında saklanmasını önler. Bilgisayar mimarisi açısından yığının en önemli uygulama alanları, altıyordamlar (bir sonraki kısımda tartışılacaktır) ve kesmelerdir (Kısım 5.2). Kısım 9.2'de, yığının işletim sistemi tarafından çoklu programlamayı, yani iki veya daha çok programı aynı anda yürütmeyi, gerçekleştirmek üzere nasıl kullanıldığı gösterilmiştir.

### Yığın ve Altyordamlar

Sözelimi bir programın yürütülmesi sırasında belli bir işlem dizisi iki veya daha çok kez yapılacak olsun. Aynı assembly-dili komut dizisini yazmak; hem programcı için sıkıcıdır, hem de bellek alanının boş yere kullanılmasına neden olur. Altyordamlar, bu probleme bir çözüm getirir. Altyordamlar, bir program içinde herhangi bir noktadan çağrılabilen program parçalarıdır. Burada 'altyordam çağırma' ifadesi altyordamın giriş noktasına (yani yürütülebilecek ilk komutuna) bir atlama yapıldığı anlamına gelmektedir. Altyordam yürütüldükten sonra çağırının yapıldığı noktayı izleyen komuta dönüş yapılmalıdır. Şekil 4:26'da bu kavram gösterilmiştir. Üst-düzey dillerde altyordam çoğu zaman yöntem olarak bilinir. Müteahhitler için taşeron ne ise, bilgisayar uygulamaları için de altyordam odur.

Altyordamlar için anahtar konu, dönüş mekanizmasıdır. Eğer birkaç noktadan aynı kod parçasına girilebiliyorsa, doğru yere dönmek için bir mekanizma bulunması gerekir. Data General Nova (4-akümülatörlü bir makinedir), dönüş adresini A3 akümülatörüne koyar. Altyordamın bitiminde programcı basitçe 'A3'ün değerini program sayıcısına yükle' komutunu yürütür ve geriye dönüş bu yolla sağlanmış olur. Ne yazık ki, bu düzenlemede beklenmedik bir pürüz vardır, - A3, bir altyordam çağırma komutu yürütüyorken, programcı tarafından kolayca kullanılamaz. A3'ün

içeriği bellekte saklanabilir ve A3 programcı için kullanılabilir duruma gelir, fakat bunun, altyordamın dönüş adresinin izini sürmek gibi bir bedeli vardır.



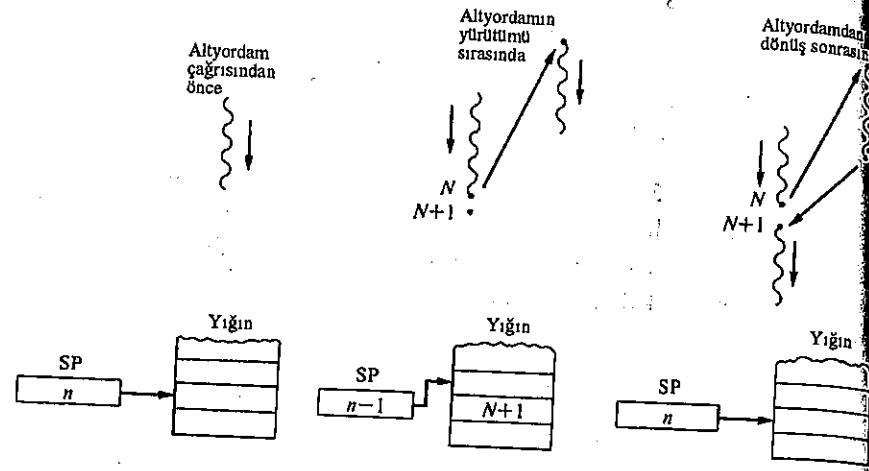
Şekil 4.26 Altyordam çağırma

Altyordamların dönüş adreslerine ilişkin işlemleri yönetebilmenin çok daha iyi bir yolu, bunları yığında saklanıktır. Bu, pekçok mikroişlemci ve mikrobilgisayar tarafından şu şekilde yapılır:

1. Altyordam çağırma: PC'nin içeriği yığına itilir, altyordama atlanır.
2. Altyordamdan dönüş: dönüş adresi yığından çıkarılır ve program sayıcısına yerleştirilir.

Bu işlemlerin yığın üzerindeki etkisi Şekil 4.27'de belirtilmiştir. Kolaylık açısından dönüş adresleri, yığında bir yer işgal ediyor olarak gösterilmiştir. Kısaca açıklanır- sa, 6502 dönüş adresini saklamak için iki konum kullanır. Altyordam çağırıldığında yığının büyüdüğüne ve dönüşten sonra küçüldüğüne dikkat ediniz. Yığına en son saklanan değer ilk çıkarılacak değer olduğundan, yığın içiçe altyordamlar açısından oldukça uygundur. Yani, bir altyordam bir başka altyordamı çağırabilir ve bu işlem sınırsız sayıda tekrarlanabilir. Gerçekte, yığına ayrılan tüm bellek dolana kadar devam edebilir ve bu aşıldığından da yığın taşması oldu denir. İçiçe altyordam kullanılması durumunda yığının davranışına bir örnek Şekil 4.28'de verilmiştir.

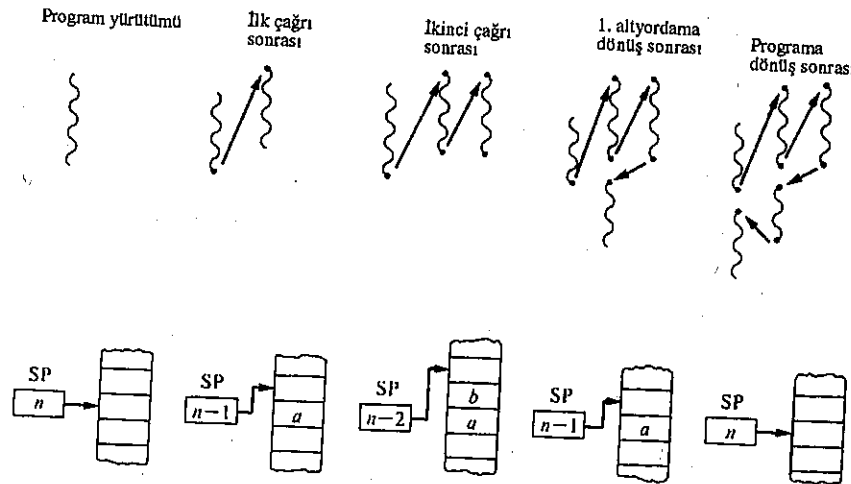
Assembly dilinde bir altyordam yazmak oldukça basittir. Bir kod blokunu bir altyordama çevirmek için yapılması gereken tek şey, blokun sonuna bir RTS komutu eklemektir. Akümülatördeki 8-bitlik bir sayının, programın çalışması



Şekil 4.27 Yığının, altyordam dönüş adresini tutmak üzere kullanılması

boyunca birkaç kez dört'e bölünüp, üç eklenmesi gerektiğini varsayalım. Tablo 4.31'de verilen altyordamla bu yapılabilir.

6502'de bir altyordam çağırmak için sadece JSR ADRES (JSR= altyordama atla, ADRES= altyordamın uygulamadaki başlangıç noktası) komutunu yürütmek gerekir. Yukarıdaki altyordamı çağırmak için 'JSR DIV4PLUS3' yazmak gerekir.



Şekil 4.28 İç içe altyordam örnekleri

Tablo 4.31 Bir altyordam örneği

|           |        |   |                        |
|-----------|--------|---|------------------------|
| DIV4PLUS3 | ASR    | A | sağa kaydır (2'ye böl) |
|           | ASR    | A | sağa kaydır (2'ye böl) |
|           | CLC    |   | eldeyi sıfırla         |
|           | ADC #3 |   | 3 ekle                 |
|           | RTS    |   | geriye dön             |

Tablo 4.32'deki programa @ yazılana kadar mikrobilgisayarın klavyesinden bir metin girilir. Sonra metin basılır. Klavyeden bir karakter girmek için bir altyordam yazmak sıkıcı olacağından, işletim sisteminde \$E993'de yer alan altyordam veri girişi için kullanılır. Benzer şekilde \$E9BC'deki bir altyordam da tek bir karakter çıkışı yapar. \$E993'teki altyordam çağrıldığında, dönüş, basılmış olan ve akümülatörde saklanan tuşun ASCII kodu ile yapılır. Benzer biçimde, \$E9BC'deki altyordamı çağırmak, akümülatördeki verinin ASCII koduna karşılık gelen karakterin basılmasına neden olur. Tablo 4.32'de yeralan programda TAMPON saklanacak veri için ayrılmış bir bellek bölgesidir.

Bu örnekte, kitapta diğer konularda geçenlerden çok belli bir 6502 -tabanlı sisteme ilişkin çevirici yönergeleri benimsenmiştir. Bu yönergelerin anlamları, okur tarafından anlaşılmış olmalıdır. Bu örnekte (NAM, END) çevirici yönergelerinde

Tablo 4.32 Bir altyordam kullanım örneği

|         | NAM BASCIKTI  | programın adı                        |
|---------|---------------|--------------------------------------|
|         | *=\$0000      | başlangıç nok. belirle               |
| TAMPON  | *=\$+40       | 40 bayt ayır                         |
| GIRIS   | =\$E993       | giriş yordam adresini tanımlar       |
| CIKIS   | =\$E9BC       | çıkış yordam adresini tanımlar       |
|         | *=\$0200      | programın başlangıç nok.             |
| SONRGIR | LDX #00       | indeks kaydedicisini sıfırla         |
|         | JSR GIRIS     | bir karakter al                      |
|         | STA TAMPON, X | aldığın karakteri sakla              |
|         | CMP # '@      | karakter @ ' mı?                     |
|         | BEQ BAS       | eğer oysa karakteri bas              |
|         | INX           | tampondaki bir sonraki konumu göster |
|         | JMP SONRGIR   | tekrarla                             |
| BAS     | LDX #00       | indeks kaydedicisini sıfırla         |
| SONRCIK | LDA TAMPON, X | bir karakter al                      |
|         | CMP # '@      | karakter @ ' mı?                     |
|         | BEQ BITTI     | oysa dur                             |
|         | JSR CIKIS     | karakter bas                         |
|         | INX           | tampondaki bir sonraki konumu göster |
|         | JMP SONRCIK   | tekrarla                             |
| BITTI   | BRK           | durdur                               |
|         | .END          |                                      |

örnek olarak bir nokta vardır. '\*' karakteri program sayıcısına karşılık gelir, böylece \*= 0000, ORG 0000'a eşdeğerdedir. TAMPON\*= \*+40, o anki 0000 konumunun çağır, ona TAMPON sembolik adını ver ve program sayıcısını ondan 40 konum ileriye taşı anlamına gelir. CMP #' @ komutu akümülatörün içeriği ile ASCII kodu @ sembolüne karşılık gelen baytı karşılaştır anlamına gelir. Üstten ayırma işareti, izleyen karakterin ASCII kodu ile değiştirilmesini göstermek için çeviriciler tarafından sık sık kullanılır. Örneğin LDA #'A ve LDA #41 eşdeğerdedir.

6502'nin ilginç bir noktası, JSR ve RTS'lerin pratik gerçekleştirilme biçimidir. JSR ADRES komutu yürütüldüğünde, üç-baytlık JSR komutunun son baytının iki-baytlık adresi, yığına itilir. JSR'nin iki-baytlık işleneni program sayıcısına yüklenir ve bir atlama işlemi yapılır. Yığındaki adresin, dönüş adresi olmayıp hemen bir önceki adres olduğuna dikkat edin.

Altyordamın sonunda bir RTS ile karşılaşıldığında, yığının tepesindeki iki-baytlık adres çıkarılır ve program sayıcısına yerleştirilir. Program sayıcısının içeriği, sonra bir artırılır böylece PC artık JSR'yi izleyen bir sonraki komutun başlangıcını göstermektedir.

#### Altyordamlar, Yığın ve Parametre Aktarma

Altyordamların işlevi, belli bir görevi yerine getirmektir. Bunu yapmak üzere hemen her zaman altyordam ile onu çağırın program arasında veri aktarmak gerekir. Tek istisnai durum, altyordamın, bir olayın başlatılmasında kullanılmasıdır. Örneğin bir altyordam bir zili çalması veya bir uyarı alarmı yayınlaması için tasarlanabilir. Basitçe onu çağırma, önceden tanımlanmış işin yerine getirilmesine neden olur ve altyordamla onu çağırın program arasında herhangi bir iletişim bulunmaz.

Şimdi de altyordamları veri giriş ve çıkışında ele alalım. Klavyeden veri almak (veya onu bir görüntüleme cihazına ya da yazıcıya aktarmak) karmaşık bir işlemdir. Bunun sonucu olarak giriş (ve çıkış) işlemleri sık sık altyordamlarla birlikte ele alınır. Popüler bir 6502-tabanlı mikrobilgisayarda, işletim sisteminin içinde \$E993'te yerleştirilmiş INALL adlı bir giriş altyordamı vardır. INALL sözcüğü altyordamın simgesel adıdır ve bilgisayarın tasarımcısı tarafından seçilmiştir. Başka bir önemi yoktur.

INALL altyordamı, JSR INALL (veya JSR \$E993) işlemi ile çağırıldığında \$E993'e bir atlama yapılır. Bu adres, altyordama giriş noktasıdır ve \$E993' yapılan bir atlamanın bir tuşa basılana kadar klavyeyi okuma etkisi vardır. Bu olduğunda, ilgili

karakter LED ekranda gösterilir ve karakterin ASCII kodu akümülatörde olacak şekilde çağırın noktaya dönüş yapılır.

Bu durumda, altyordam ile çağırın program arasında bir iletişim sözkonusu olmaktadır. Altyordamdan çağırın programa sadece tek bir bayt aktarıldığından, bunun için akümülatör elverişli bir araçtır. Altyordam çağırılmadan önce akümülatörde bulunan bilgilerin silineceğine dikkat ediniz.

Altyordam ile çağırın program arasındaki yukarıda yeralan veri aktarma metodu veri aktarımında kaydedicileri kullanır. Örneğimizde, veri akümülatöre aktarılmıştır. İndeks kaydedicilerinin kullanılmaması için hiç bir neden yoktur. PSR'nin elde bitinin sık sık altyordamdan onu çağırın programa bilgi aktarmak için seçildiğinde dikkat ediniz. Tek bir bit önemli miktarda bilgi gibi görünmeyebilir. Öte yandan, kimi zaman altyordam içinde bazı problemler ortaya çıkabilir ve bu problemin, çağırın programa bildirilmesi gerekir. Bir terminalden veri girişi yapmak için altyordamın çağırıldığını ve terminalin arızalı veya kapalı olduğu durumu ele alalım. Altyordamdan bir dönüş yapılmadan önce elde bitini kırarak, çağırın programa bir hatanın olduğu söylenebilir. Tablo 4.33'te listelenen bir program parçası bu noktayı belirtmektedir.

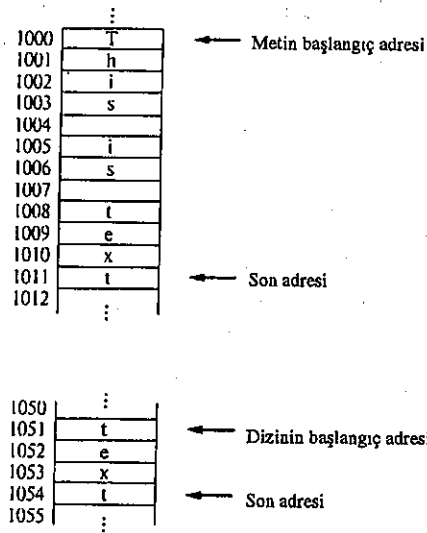
Tablo 4.33 Elde biti aracılığıyla bilgi aktarılması

|                        |                                                                                             |
|------------------------|---------------------------------------------------------------------------------------------|
| JSR VERIAL<br>BCS HATA | altyordamı çağır ve veri ile birlikte akümülatöre dön<br>eğer elde 1 ise hatalı bir şey var |
| .                      | veriyi gözden geçir                                                                         |
| HATA                   | hatalı durumdan kurtul                                                                      |

Eski veriyi silmiş olmak dışında, kaydedicilere bilgi aktarılmasının önemli bir dezavantajı yoktur. Ne yazık ki, büyük miktarlarda veri aktarımı gereken durumlarda uygulanabilir bir yöntem değildir. Bir altyordamın, bir bellek bölgesindeki metin üzerinde, verilen bir dizinin ilk tekrarını bulmak için arama yapmak amacıyla yazıldığını düşünelim. Bu durumda altyordama arama yapılacak bellek bölgesinin başlangıç ve bitiş adresleriyle bulunacak ilgili karakter dizisinin başlangıç ve bitiş adresleri hakkında bilgi verilmelidir. Şekil 4.29'daki adres haritası bu problemi göstermektedir.

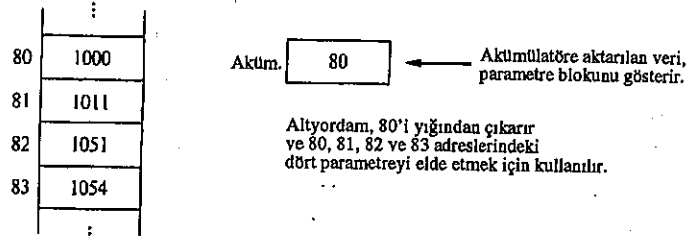
Altyordamın istediği bilgi 1000, 1011, 1051 ve 1054 sayılarıdır. Yani dört tane 16-bitlik adrestir. Altyordam, çağırın programa 1008 değerini ('metin' dizisinin

başlangıç noktasını) döndürmelidir. Açıkça görüldüğü gibi bu bilgi, 6502'nin kaydedicileri aracılığıyla aktarılamaz.



Şekil 4.29 Dizi-eşleme problemine ilişkin bellek haritası

Bilgilerin, altıyordam ile çağırılan program arasında aktarılması için yine kaydedicilere dayanan olası bir çözümde; bilgilerin ne olduklarından çok konumunun aktarılması söz konusu olmaktadır. Yani, altıyordama bilmek istediği bilgiyi söylemek yerine ona bu bilgiyi nerede bulacağını söyleyebiliriz. Şekil 4.30'da parametrelerin bir blokta nasıl gruplandırılıp, altıyordama, bloğun başlangıcının aktarıldığı görülmektedir.



Şekil 4.30 Parametrelerin adresleriyle birlikte aktarılması

Hem altıyordama, hem de çağırılan programa ait ortak bir veri alanı ayırmak mümkündür. Bu, parametre aktarımını ortadan kaldırır çünkü altıyordam tarafından istenen değişkenler çağırılan programdakilerle aynıdır. Bu türde bir çözüm yaygın

değildir çünkü böyle bir uygulamada altıyordam, bir başka program tarafından kesilip, çağırılamaz. Dışardan adlandırılmış konumlarda saklanan veriler, kesen program tarafından bozulacaktır. Kısım 5.2'de kesme mekanizması ve gerçekleştirilmesi konusu ayrıntılı olarak ele alınacaktır. Bu kısım için kesmenin kısa bir açıklaması yeterli olacaktır. Kesmeler, işlemciyi beklenen iş sırasından bir saptırma yöntemi olup ortaya çıkar çıkmaz, istenen gereken giriş çıkış işlemleri (ve diğer işlemlerle) ilgilenilmesini gerektirir. İşlemci, herhangi bir birimden gelen kesme isteğini her aldığı anda, o anki yürütmekte olduğu komutunu durdurur ve kesmenin gerektirdiği işlemi içeren programa atlar. Kesmenin gerektirdiği işlemler yapıldıktan sonra, dönüş, kesme gelmeden önce yürütülen son komutu izleyen bir sonraki noktaya yapılır. Kesmenin dönüş mekanizması altıyordamınkiyle hemen hemen aynıdır - yani burada da dönüş adresi yığında saklanmaktadır.

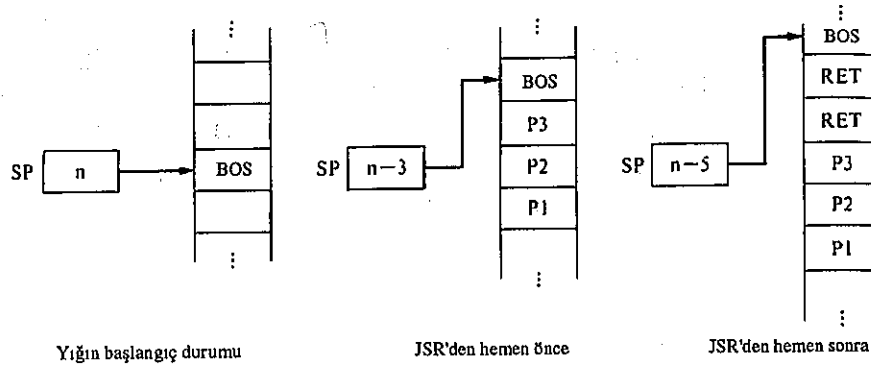
Bir altıyordamın, yürütümü sırasında kesildiği bir durumu göz önüne alalım. Eğer kesme yönetim yordamı da altıyordamı kullanmak isterse, altıyordamın yeniden kullanılması halinde, dışardan adlandırılmış bulunan bellek konumlarındaki verilerin üzerine yazım yapılacaktır ve bu veriler bozulmuş olacaktır. Eğer veriler yazmaçlarda saklanmış ve kaydedici içerikleri de kesme yönetim yordamı tarafından yığına itilmiş olsaydı, altıyordamdaki hiç bir veri, yeniden kullanım nedeniyle kaybolmayacaktı. Altıyordamın kesme-yönetim yordamı tarafından yeniden kullanılmasının ardından, yığındaki akümülatörün içeriği geriye getirilir ve kesmeden dönüş, kesmeye hizmet verildiği andaki kaydedici içerikleriyle aynı yapılırdı. Altıyordam ile çağırılan program arasında bilgi aktarmada ideal yol, yığın kullanmaktır. Sözgelimi altıyordam tarafından P<sub>1</sub>, P<sub>2</sub> ve P<sub>3</sub> gibi üç parametreye ihtiyaç duyulmaktadır. Parametreler, altıyordam çağırılmadan hemen önce yığına itilmiştir (Tablo 4.34).

Tablo 4.34 Parametrelerin yığına aktarılması

|     |     |
|-----|-----|
| LDA | P1  |
| PHA |     |
| LDA | P2  |
| PHA |     |
| LDA | P3  |
| PHA |     |
| JSR | SUB |

Şekil 4.31'de yığının başlangıçtaki, altıyordam çağırılmadan önceki ve hemen sonraki durumları verilmiştir. Dönüş adresinin yığında iki konumluk yer kapladığına dikkat ediniz. 6502'de yığın işaretçisi, TOS'u değil, TOS'un üstündeki ilk boş konumu gösterir. Şekil 4.31'de bu BOS sözcüğü ile etiketlenmiştir.

Altyordama giriş yapılırken, veriler birkaç şekilde geri getirilir. Bir seçenek, yığın işaretçisinin içeriğini TSX komutu ile indeks kaydedicisine aktarmaktır. Artık P1 parametresinin akümülatöre yüklenmesi, LDA 5,X komutu ile gerçekleştirilebilir. Benzer biçimde, P2, LDA 4,X ile ve P3 de LDA 3,X ile yüklenebilir.



Şekil 4.31 Parametrelerin yığına aktarılmaları

Bu teknikte ortaya çıkan tek güçlük, RTS komutu ile altyordamdan dönüş yapıldıktan sonra, yığın işaretçisinin içeriğinin SP-3 olmasıdır. Buradaki SP, P1, P2 ve P3 yığına itilmeden önceki değerdir. Yığın, başlangıç değerine üç tane ardışık (kukla) PLA işlemi yürüterek getirilir. Bu işleme "yığını temizleme" denir.

Altyordama parametre aktarmak için yığın kullanmanın avantajı şudur: parametreleri bozulmaksızın, altyordam kesilip, bu değerler kesen program tarafından kullanılabilir. Veri yığında saklandıkça altyordam kesildiğinde değerinin üstüne yazılmaz çünkü veri yığının tepesine eklenir ve kesmeye hizmet verildikten sonra da dışarı alınırlar.

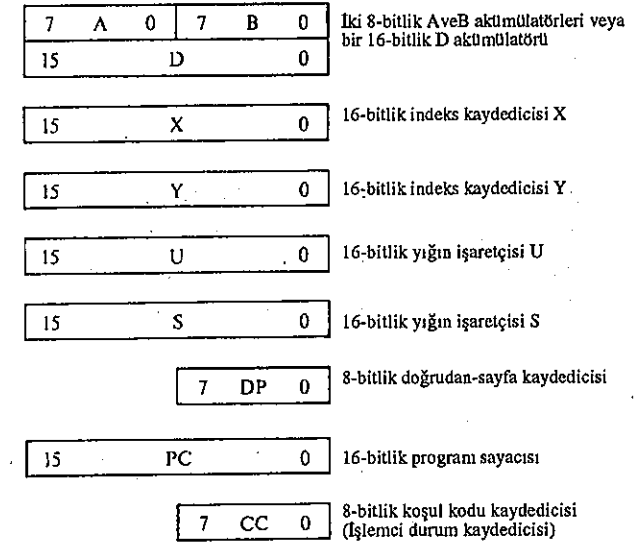
#### 4.5 6809 Mikroişlemcisi

Sekiz-bitlik Motorola 6809 mikroişlemcisi bugün mevcut olanların en gelişmiş türüdür. 6809'un en önemli özelliği veri işlemeyi nispeten kolaylaştıran adresleme modu zenginliğidir. 6809'un iç kaydedici yapısının blok diyagramı Şekil 4.32'de verilmiştir.

6809, 8-bitlik veri sözcükleriyle çalışır ve 64 Kbayta kadar bellek adresleyen 16-bitlik bir adres yoluna sahiptir. Bu açıklamadan 6809'un 6502 veya 6800 ile

aynı olduğu düşünülebilir 6502'ye benzemez. Ne var ki 6809'un 16-bitlik adres veya işaretçi kaydedicisi vardır. Bundan başka iki indeks kaydedicisi (X, Y) iki de yığın işaretçisi (U, S) bulunur. S ve U yığın işaretçilerinin ikisi de programcının kullanımına açıktır. MIB Kesme yönetimi veya altyordam dönüş adresleri için yığını kullandığı zaman, S yığın işaretçisini de kullanır.

İki tane 8-bitlik akümülatör vardır: A ve B. 6809'un özellikle hoş bir özelliği; programcıya iki akümülatörü, 16-bitlik tek bir akümülatör, D kaydedicisi, olarak kullanabilme olanağı sağlamasıdır. Bu yolla 6809, gerçek 16-bitlik bir bilgisayar gibi davranır, itiraf edilmelidir ki 6809'daki verilere ilişkin işlemlerin yalnızca bir grubu D kaydedicisine uygulanabilir.



Şekil 4.32 6809'un iç kaydedicileri

6809'a özgü bir kaydedici, sekiz bit DP doğrudan sayfa kaydedicisidir. 6502'de 'sıfır sayfa adresleme' 6800'de de 'doğrudan adresleme' vardır. Bunların ikisi de belleğin ilk 250 baytının (sıfırıncı sayfanın) tek bir 8-bitlik işlenene gereksinim duyan bir adresleme moduyla kullanılmasını sağlar. Açıkçası bu adresleme modu, programcıyı sık kullanılan veriyi \$0000-\$00FF aralığına yerleştirmeye zorlar. Bellek alanının bu biçimde kullanımının uygun olmadığı birçok durum vardır. Doğrudan sayfa kaydedicisi bu adresleme moduyla birlikte kullanılacak (256 taneden bir) sayfayı işgal eder. Bu sonuçta, 6809 sıfır-sayfa adreslemeyi elverişli

ve ekonomik yaparken programcının herhangi bir sayfayı, sıfır-sayfa olarak seçmesini sağlayarak daha büyük bir esneklik verir.

#### 4.5.1 Adresleme Modları ve 6809

6809'da on adresleme modu vardır: doğal, dolaysız, doğrudan, genişletilmiş, bağlı dallanma, indeksli, genişletilmiş dolaylı, program-sayıcısına bağlı, indekslenmiş dolaylı ve kaydedici. Bu adlar, Motorola'nın kendi dilinde kullandığı adlardır. Bu adresleme modlarının çalışması aşağıdaki gibidir:

**Doğal adresleme:** Bu mod, sırf reklam olsun diye, adresleme mod sayısını artırmak için ortaya atılmış değildir. Doğal (veya örtük) adreslemede adres, komutun içinde örtülü biçimde belirtilir. Örneğin 'A akümülatörünü sıfırla' komutu, hem ne yapılacağını, hem de işlemde yer alan verinin kaynağını ve varış yerini belirtir. Doğal adresleme tek baytlık opkod kullanır.

**Dolaysız adresleme:** Bu modda, bir işlemde yeralan veri doğrudan opkodun ardından gelmektedir. 6809, 8 ve 16-bitlik veri değerleriyle çalıştığından; dolaysız adresleme, opkodun yanında bir veya iki baytlık bir işlenene gereksinim duyar. Aşağıdaki örneklerde bu, daha açık biçimde gösterilmiştir - 6809 iki akümülatöre sahip olduğundan, hangisinin kullanılacağını belirtmesi gerektiğine dikkat edin. Dolayısıyla programcı ya LDA 1234 ya da LDB 1234 komutunu yazmalıdır.

LDA #37      [A] ← 37  
LDX #1234    [X] ← 1234  
LDD #1234    [D] ← 1234, veya [A] ← 12, [B] ← 34

**Genişletilmiş adresleme:** Diğer terminolojilerde bu modun adı, mutlak adreslemedir. Burada opkodu izleyen işlenen, aşağıdaki örneklerde de olduğu gibi, işlemde kullanılacak verinin mutlak adresini belirtir.

LDA #37      [A] ← [0037]  
LDX #1234    [X] ← [1234], [1235]

16-bitlik bir değeri X kaydedicisine yükleyebilmek için; MİB bellekten, -bir 1234 adresinden bir de 1235 adresinden olmak üzere- iki bayt getirmelidir. 6809'un (6800 gibi ve 6502'den farklı olarak) adresleri, bellekte en değerlikli bayt önce yazılacak şekilde (yani daha alt sıralı adreste) sakladığına dikkat edin.

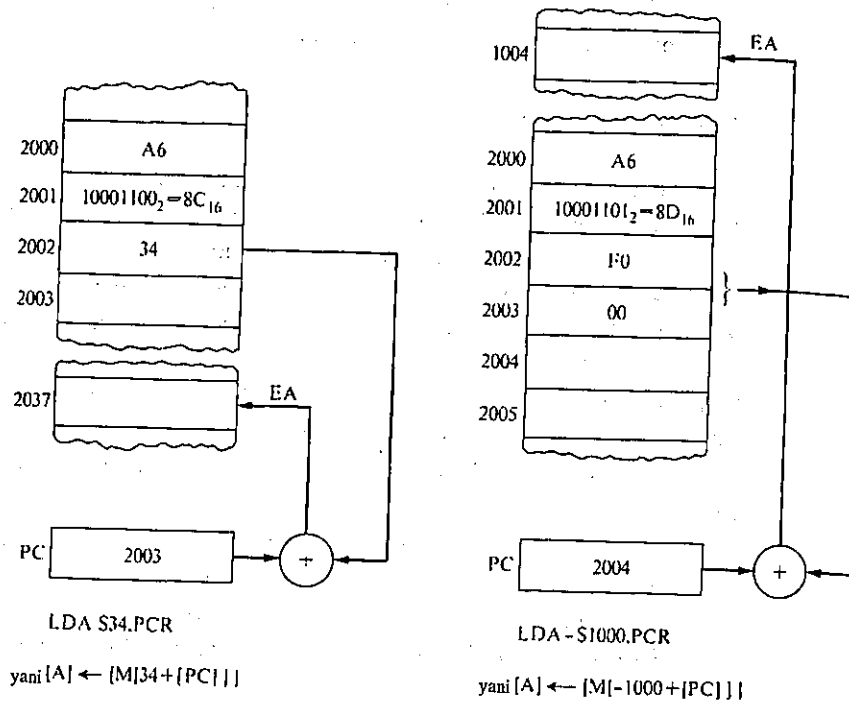
**Doğrudan adresleme:** Bu, 6502'deki sıfır-sayfa adresleme ile tamamıyla bir analogi oluşturur. Tek istisnai durum, bu adresleme yöntemini kullanmadan önce sayfa sayısını 6809'un doğrudan sayfa kaydedicisine yükleyerek herhangi bir sayfanın sıfırıncı sayfa yapılabilmesidir. Aşağıdaki doğrudan adresleme örneklerini inceleyin.

LDA #12      [A] ← 12  
EXG A, DP    [DP] ← [A], [A] ← [DP]  
LDA 34      [A] ← [1234]

Bu örnekte A akümülatörü dolaysız bir değer olan 12 ile yüklenmiştir. 'EXG A,DP' komutu A akümülatörünün içeriği ile doğrudan sayfa kaydedicisinin içeriğini kendi aralarında yer değiştirir - 'DP'ye dolaysız bir değer yükle' komutu yoktur. Son komut olan LDA 34, tek baytlık bir işlenene sahiptir, böylece çevirici, doğrudan sayfa adresleme modunu seçer ve sayfa sayısı (12), istenen onaltı bit adres olan 1234'ü vermek üzere 34 işlenenin önüne eklenir. Eğer A'ya 1234 üzerine 34 yüklemek isteseydik, doğrudan adresleme modu için bastırmak üzere LDA 0034 yazardık.

**Dallanmaya bağlı adresleme:** 6502 gibi 6809'da dallanma komutlarının ardından bağlı adreslemeyi kullanır. Bir dallanmayı takip eden tek-baytlık ikiye tümleyen biçimindeki işlenen, dalın gerçek adresini vermek üzere program sayıcısının içeriğine eklenir. 6502 veya 6800'deki bağlı adreslemenin dezavantajı, dallanmanın -126, +129 aralığında sınırlandırılmış olmasıdır. 6809'da bir grup uzun-dallanma komutu yer alır. Burada opkodu, program sayıcısının o anki içeriğinden itibaren -32764 - +32771 bayt aralığında bir dallanmaya izin veren ikiye tümleyen biçiminde iki-baytlık bir ofset takip eder.

**Program sayıcısına bağlı adresleme:** Bu adresleme modunda (ki 6502, 6800, 8080, Z80'de vb. tümüyle mevcut değildir) bir işlenenin gerçek adresi, program sayıcısının içeriği artı komutu izleyen 8-bitlik veya 16-bitlik bir ofset ile verilir. Bu adresleme modunun gücü (bağlı adresleme ile birleştirildiğinde) bizi, tümüyle konumdan-bağımsız kod sonucuna götürmektedir. Tüm veri konumları program sayıcısına göre belirtildiğinde, program ve onun veri alanı bellek içinde herhangi bir adresin değiştirilmesine gerek duyulmadan taşınabilir. Bu olanak özellikle çoklu programlama için faydalıdır (Kısım 9.2). Eğer belleğe yeni bir program saklanırsa, diğer programlarda değişiklik yapılmaksızın aşağı veya yukarı taşınabilir. Program sayıcısına bağlı adreslemenin formatı ve çalışmasına iki örnek Şekil 4.33'te verilmiştir:



Şekil 4.33 Program-sayıcısına bağlı adresleme

Soldaki örnekte (sekiz-bitlik ofset durumu), opkodu izleyen iki tane 8-bitlik sayı vardır. İlkine, 10001100, (Motorola tarafından) sonbayt denir ve adresleme modunu belirtir. 6809'da o kadar geniş bir adresleme modu çeşitlenmesi vardır ki, olası tüm opkodların tümü sekiz bayta yerleştirilemez. Sonuç olarak, 8bitlik komutu 16-bitlik bir komuta çevirmek için bu son-bayt gerekir. Son-baytla ilgili diğer ayrıntılar Tablo 4.35'te verilmiştir.

Son-baytı izleyen sayı (ofset), \$34, program sayıcısının içeriğine (yani \$2003'e) eklenerek gerçek adres olan \$2037'yi verir. Program sayıcısı adreslemede program sayıcısındaki ofsete eklenecek sayının, yürütülecek bir sonraki komutun adresi olduğuna dikkat ediniz. Sağdaki örnekte sonbayt 10001101'dir. En küçük değerlikli bit, bire kurulmuştur ve böylece MİB'e 16-bitlik ofsetin kullanıldığını belirtir. Bu ofset - \$1000 veya ikiye tümleyen biçiminde \$F000'dır. Bu değer, program sayıcısının içeriğine eklendiğinde (\$2004), \$1004, etkin adresi üretilir.

Son-baytlarla ofsetler arasındaki hesaplamaların, hemen hiç bir zaman programcı

tarafından yapılmayıp otomatik olarak çevirici tarafından yapıldığına dikkat ediniz. Programcı komutu sembolik biçimde yazar (örneğin LDA DEGER1, PCR) ve çevirici opkodu, son-baytı ve ofseti hesaplar.

#### İndekslenmiş adresleme:

İndekslenmiş adreslemede işlenenin gerçek adresi, opkodu izleyen ofsetin indeks kaydedicisinin içeriğine eklenmesiyle elde edilir. 6800 bir tane 16-bitlik indeks kaydedicisi ile sınırlıyken, 6502 iki tane 8-bitlik indeks kaydedicisiyle sınırlıdır. 6809'un tasarımcıları indeksleme konusunda daha da ileriye gitmiş ve 6800 ve 6502'nin kısıtlamalarını intikam ateşiyle ortadan kaldırmışlardır.

6809'da iki tane 16-bitlik indeks kaydedicisi, X ve Y, vardır ve indekslenmiş adresleme üzerine dört değişik biçim içerir. Bunlar: sıfır-sayfa indekslenmiş, sabit-ofset indekslenmiş, akümülatör-ofset indekslenmiş ve otomatik-artımlı (otomatik-azalımlı) indekslenmiş adreslemedir. İlk iki mod tamamıyla açıktır. Sıfır sayfa indekslenmiş adreslemede, indeks kaydedicisi işlenenin gerçek adresini içermektedir. Bu adresleme modu, 6800 programcısı tarafından kullanılan en yaygın indekslenmiş adresleme modunun LDA 00,X olduğunun gözlenmesiyle yer almıştır. Sıfır ofsetini düşürerek bir baytlık yer tasarruf edilir (sıfır sayfa adresleme).

Sabit ofset indekslenmiş adreslemede gerçek bir adresi üretmek için 5, 8 veya 16-bitlik işaretli bir ofset dört işaretçi kaydedicisinden birine eklenir. İndeksleme türü, daha önce tartışıldığı gibi, opkodu izleyen sonbayt ile belirlenir. Aslında, 5-bitlik işaretli bir ofset kullanılırken ofsetin kendisi sonbaytın bir kısmını oluşturmaktadır. 5-bitlik işaretli bir ofset kulağa tuhaf gelebilir fakat Motorola, indeks kaydedicisinin birçok uygulamasında -16, +15 aralığındaki bir ofsetin yeterli olduğunu belirlemiştir.

8-bitlik işaretli bir ofset gerektiğinde, sonbaytı izleyen tek-baytlık bir ofset, -128, +127 aralığında bir indeksleme sağlar. 16-bitlik işaretli bir ofset, -32.768 - +32.767 aralığında bir indeksleme olanağı vermek üzere sonbaytı izleyen iki-bayt olarak sunulur.

6809'un önemli bir özelliği akümülatör-ofset indekslenmiş adreslemedir. Burada akümülatörün içeriği bir işaret kaydedicisine eklenerek işlenenin gerçek adresini verir. 8-bitlik akümülatörler A ve B, kullanılabileceği gibi, D kaydedicisini kullanarak 16-bitlik bir ofset de yaratılabilir. Yukarıdaki indekslenmiş adresleme modlarında olduğu gibi, 6809, akümülatördeki ofsetin işaretli bir değer olduğunu kabul

eder. Bu adresleme modunun yararlı olmasının nedeni, çalışma-zamanında hesaplanan değişken ofset üretimini olanaklı kılmasıdır. Öteki indekslenmiş adresleme modlarında ofset sabit olup program yazılırken programcı tarafından tanımlanır.

Bir tablodaki veriler üzerinden indekslenmiş adresleme ile devam ederken indeks kaydedicisi de, indekslenmiş her işlemten sonra programcı tarafından artırılır. Otomatik-artımlı indekslenmiş adresleme modunda indeks kaydedicisinin içeriği, bu mod her kullanıldığında otomatik olarak artırılır. Otomatik-artımı çiftli mod ile kullanmak da mümkündür; böylece 16-bitlik sözcüklerden oluşan tablolarda da işlem yapabiliriz. Otomatik-artımlı modun yanısıra, 6809'da bir veya iki ile otomatik-azalım da vardır. Otomatik-artımın indekslenmiş bir işlemten sonra oluştuğuna dikkat edilmelidir. 6809 çeviricisi, otomatik-artım/otomatik azalımı +/- ile; iki ile otomatik-artım/otomatik-azalımı ise ++ / -- ile gösterir.

Aşağıda, bir ile otomatik-artım ve iki ile de, otomatik-azalım işlemleri belirtilmiştir. 6809'un çeşitliliğini göstermek için A ve B akümülatörleri ile X ve Y kaydedicilerini kapsayan işlemleri seçtim.

LDA, X+      [A] ← [[X]]  
                  [X] ← [X] + 1  
 LDA, -- Y    [Y] ← [Y] - 2  
                  [D] ← [[Y]]

Daha önce de belirtildiği gibi 6809 tarafından sağlanan indekslenmiş adresleme modlarının zenginliği, opkodu nitelikle için ek bir bayta (son-bayt-postbyte) gereksinim duyar. Tablo 4.35'te son-baytın sekiz bitinin indekslenmiş adresleme işleminin modunu nasıl belirlediği gösterilmiştir.

**Dolaylı adresleme:** Dolaylı adreslemede işlenenin gerçek adresi komut tarafından belirtilen konumda bulunur. Elbette ki dolaylı adreslemenin indeksleme gibi diğer adresleme modlarıyla birlikte kullanılamaması için bir neden yoktur. Aslında, dolaylı adresleme 6809'un bir ile otomatik-artım/otomatik-azalım modu hariç tüm indekslenmiş modlara uygulanabilir. Son-baytın 4-biti dolaylı adresleme modunu seçer. Sıfırken, işlenenin adresi yukarıdaki tabloda olduğu gibi EA ile verilir. Birken dolaylı adresleme seçilir ve işlenenin adresi [EA] ile verilir. Yani işlenen; gerçek adresi seçilen belli indekslenmiş moda göre hesaplanan konumun içerdiği değerdir. Örneğin; eğer indeks kaydedicisinde 5 varsa; 'LDA 7,X' işlemi, 12 bellek konumunun içerdiği değerin akümülatöre getirilmesine neden olur. Eğer işlem LDA [7,X] olsaydı, 12-bellek konumunun içerdiği değer akümülatöre yüklenecek verinin adresini sağlamak üzere kullanılacaktı.

Tablo 4.35 6809 indekslenmiş işlem izleyen son-bayt

| Son-bayt biti |   |   |   |   |   |   |   | Etkin adres                     |
|---------------|---|---|---|---|---|---|---|---------------------------------|
| 7             | 6 | 5 | 4 | 3 | 2 | 1 | 0 |                                 |
| 0             | R | R | X | X | X | X | X | EA = [R] + 5-bitlik ofset       |
| 1             | R | R | 0 | 0 | 0 | 0 | 0 | EA = [R], [R] ← [R] + 1         |
| 1             | R | R | 1 | 0 | 0 | 0 | 1 | EA = [R], [R] ← [R] + 2         |
| 1             | R | R | 0 | 0 | 0 | 1 | 0 | [R] = [R] - 1, EA = [R]         |
| 1             | R | R | 1 | 0 | 0 | 1 | 1 | [R] = [R] - 2, EA = [R]         |
| 1             | R | R | 1 | 0 | 1 | 0 | 1 | EA = [R]                        |
| 1             | R | R | 1 | 0 | 1 | 1 | 0 | EA = [R] + [A]                  |
| 1             | R | R | 1 | 1 | 0 | 0 | 0 | EA = [R] + [B]                  |
| 1             | R | R | 1 | 1 | 0 | 0 | 1 | EA = [R] + 8-bitlik ofset       |
| 1             | R | R | 1 | 1 | 0 | 1 | 1 | EA = [R] + 16-bitlik ofset      |
| 1             | 1 | X | 1 | 1 | 1 | 0 | 0 | EA = [R] + [D]                  |
| 1             | 1 | X | 1 | 1 | 1 | 0 | 1 | [PC] + 8-bitlik ofset           |
| 1             | X | X | 1 | 1 | 1 | 1 | 1 | EA = [PC] + 16-bitlik ofset     |
|               |   |   |   |   |   |   |   | EA = [16-bitlik işaretli ofset] |

1 - dolaylı bit (kurulduğunda dolaylı adreslemeyi gösterir). X - dikkate alınmaz (0 veya 1 olabilir). 5-bit ofset durumunda ise X'ler ofsetin değerini tanımlar. Son-baytın 5 ve 6 bitleri iki tane kaydedici seçme biti olup gerçek bir adres üretiminde kaydediciyi indeks kaydedicisi gibi davranıyor olarak tanımlar. 6-bit, 5-bit - 0,0 ise R - X kaydedicisi, 6-bit, 5-bit - 0,1 ise R - Y kaydedicisi. 6-bit, 5-bit - 1,0 ise R - U kaydedicisi, 6-bit, 5-bit - 1,1 ise R - S kaydedicisi.

Dolaylı adresleme bir komutun işleneni, köşeli parantez içinde belirtilir. Aşağıda verilen üç örnekte bu nokta gösterilmiştir:

Assembly dili biçimi

RTL biçimi

LDA [25]

[A] ← [[25]]

LDA [10, X]

[A] ← [[X] + 10]]

LDA [25, PCR]

[A] ← [[PC] + 25]]

**Kaydedici adresleme:** Kaydedici adresleme, verinin taşınması (değiştirme) için iki kaydedici kullanılması dışında aslında bir doğal adresleme biçimidir. Kaydedici aktarımı veya karşılıklı değiştirmeyi bir sonbayt izler. Son-baytın en büyük değerlikli dört biti; kaynak kaydedicisi, en küçük değerlikli dört biti; de varış kaydedicisini belirtir.

6809'un komut takımıyla ilgili önemli noktalar

Bu kitap, karşılaştırmalı mikroişlemci mimarisine ilişkin bir ders kitabı olarak hazırlanmamıştır. İşlemci yapısını göstermek için, 6502 işlemcisi seçtim. Gün-

müzde artık bu işlemci biraz demode olduğundan, bu kısımda, benzer mimar komut takımına ve daha fazla sayıda iç kaydedici ve adresleme moduna sahip, yeni bir işlemci olan 6809 ile ilgili ayrıntılara yer verilmiştir. 6809'un komut takımının tümünü sunmanın pek bir anlamı yoktur. Öte yandan 6502'den farklı olan bazı 6809 komutlarını açıklamak yararlı olacaktır. 6809, 6502'den daha çok kaydediciye sahip olduğundan, bir çok komutta kaydedicileri birbirinden ayırmak gerekmektedir. İki akümülatör (A ve B), iki indeks kaydedicisi (X ve Y) ve iki yığın işaretçisi (U ve S) olduğundan bu, özellikle doğrudur. İki akümülatör 16-bitlik bir kaydedici olarak her kullanılışında D kaydedicisi olarak anılır.

#### Veri taşıma

LEA (Etkin adresi yükle) çok güçlü bir komuttur ve 16-bitlik işaretçi kaydedicilerle biriyle yapılan bir işleme ilişkindir. Genel biçimi şöyledir:

LEAP OFFSET, Q

Burada P; X, Y, S ve U'dan biri, Q; X, Y, S, U ve PC'den biri olup; OFFSET de işaretli bir ofsettir.

LEAP OFFSET Q'nun sonucu şudur:  $[P] \leftarrow \text{OFFSET} + [Q]$ .

LEA komutu 6502 veya 6800'de bulunan birkaç komutun yerini alır. Örneğin 'LEAX - 1, X', DEX'in; 'LEAX 1, X', INX'in eşdeğeridir. 'LEAY 50, X' işleminin etkisi; X kaydedicisinin içeriği, artı 50'nin Y kaydedicisine aktarımıdır.

PSH (kaydedici içeriklerini yığma it) iki istisna dışında 6502'nin PHA komutu gibi davranır. İki yığın olduğundan, komut ya PSHS ya da PSHU şeklinde yazılmalıdır. Bu komut bir seferde çeşitli kaydedici grubunun içeriğini yığma itebilir (kullanılmakta olan yığın işaretçisi hariç). Örneğin 'PSHS, X, Y ve A' komutu, X, Y ve A'yı yığma itmektedir. Bu özellikle, altyordam yürütülmeden önce kaydedici içeriklerinin saklanması kural olduğu, altyordam girişlerinde ve altyordamdan dönmeye önce bunları geri almada, çok kullanışlıdır. PSH işleminin tersi PUL'dur. PULS PC (program sayıcısını sistem yığınının çıkar) işleminin, RTS ile aynı etkiye sahip olduğuna dikkat ediniz.

TFR (kaydediciden kaydediciye aktar) bir kaydedicinin içeriğini diğerine kopyalar. Akümülatörlerden biri diğerine 'TFR A, B' (veya TFR B, A) veya 16-bitlik bir kaydedicinin içeriği 16-bitlik bir başka kaydediciye kopyalanabilir. TFR X, Y işlemi X'in içeriğini Y'ye kopyalar ve 'LEAY 0, X' ile aynı etkiye sahiptir. 'TFR

X, PC' işleminin etkisi X kaydedicinin içeriğini program sayıcısına aktarmaktır ve dolayısıyla da X kaydedicisindeki adrese atlama yapılmasını zorlar.

EXG (kaydedici içeriklerini karşılıklı değiştir) iki kaydedicinin içeriklerinin yer değiştirmesine neden olur. Karşılıklı değiştirme işlemi simetrik olduğundan assembly biçimindeki 'EXG, A, B' ve 'EXG B, A' eşdeğerde komutlardır.

#### Aritmetik ve mantıksal komutlar

6809'un aritmetik ve mantıksal komutları birkaç istisna dışında 6502'ninkilerle benzerlik gösterir. 6502'nin komut takımına yapılan bazı faydalı eklemeler şunlardır:

ABX (B akümülatörünün içeriğini X indeks kaydedicisine ekle) çok kullanışlıdır çünkü aritmetik işlemlerin indeks kaydedicisi üzerinde yapılmasını sağlar.

ADD (A, B veya D) akümülatörlerinin içeriğini bellek konumuna ekle) 6502'deki CLC, ADC dizisine eşdeğerdir. A ve B akümülatörleri 16-bitlik D akümülatörünü oluşturmak için birleştirilebildiğinden, ADD M'nin etkisi, M ve M+1 bellek konumlarının içeriğini D'ye eklemektir. Aşağıdaki işlem dizisini ele alalım:

LDD P

ADD Q

STD R

Bunların sonucunda, 16-bitlik P ve Q değerleri toplanıp sonuç R'ye konmaktadır. CLR (Akümülatörü veya belleği sıfırla) akümülatörün (CLRA veya CLRB) veya bir bellek konumunun içeriğini sıfıra kurmak etkisine sahiptir.

COM (bir akümülatörün veya bellek konumunun içeriğini tümle) bire tümleme etkisine sahiptir. Örneğin COMA, sonuçta A'nın tüm bitlerinin tümlenmesine veya terslenmesine neden olur.

DEC'in (bir akümülatörün veya bellek konumunun içeriğini azalt) etkisi ilgili sözcüğün İngilizce'sinden (DECrement=azalt) yapılan bir kısaltmadır. 6502'de bu komut, bellek referans biçiminde olup akümülatör eşdeğerinde (yani DECA) değildir. 6809 ayrıca INC (artır) komutuna sahiptir.

NEG (bir akümülatör veya bellek içeriğini negatif yap) seçilen sözcüğün ikiye tümleme etkisine sahiptir.

SEX (B akümülatöründen A akümülatöre işaret uzat) B akümülatöründe iki tümleyen biçimindeki 8-bitlik sayıyı D akümülatöründeki 16-bitlik ikinin tümü yeni bir sayıya dönüştürür. SEX'in etkisi A akümülatörünü B akümülatörünün bitinin (EBDB) sekiz kopyası ile doldurmaktır.

#### 6809 programına bir örnek

Sadece kısa bir program yazarak 6809'un 6502'ye göre üstünlüklerini göstermek neredeyse imkansızdır. Bu arada, 'güç'ün ne anlama geldiğini belirtmedik. Anlamak istediğimiz şey hız mı, amaç programın fiziksel büyüklüğü mü (makine kodundaki bayt sayısı), veya şıklık (bir assembly dili programını yazılma ve anlaşılma kolaylığı) mı? Gerçek şu ki, 6809, yukarıdaki üç noktada da 6502'ye göre daha ileri bir düzeyi gösterir.

Aşağıdaki program, parametre aktarma kısmında açıklanan metin-eşleme algoritmasını gerçekleştirir. Belleğin bir bölgesi, başlangıç adresi \$2000 olan bir metin satırı içermektedir. Satır, ASCII kodu \$0D olan alt-satır ile sonlandırılmıştır. Bir karakter dizisi de \$0080'den başlayarak devam eden bir tamponda saklıdır ve o da bir alt-satır ile sonlandırılmıştır. Altyordam biçiminde yazılacak programda, satırın, dizinin ilk karakteriyle eşleşen ilk karakterinin adresi, dönüşten önce kullanıcı yığına yerleştirilir. Eşleşme başarısızsa yığma, boş adres, 0000 itilir.

Aşağıda metin satırı, 'satır' olarak; karakter dizisi de, 'dizi' olarak anılacaktır. Problem; dizideki her bir karakter satırda karşılık gelen karakterle eşleşene kadar, diziyi satır boyunca kaydırarak çözülebilir. Tablo 4.36(a), (b)'ye bakınız.

Tablo 4.36 (a) İki dizinin eşleştirilmesi

| THIS | THAT | THEN | THE | OTHER | satır |        |
|------|------|------|-----|-------|-------|--------|
| THEN | THE  |      |     |       | dizi  |        |
| THIS | THAT | THEN | THE | OTHER | Adım  | Eşleri |
| THEN | THE  |      |     |       | 1     | 2      |
| THEN | THE  |      |     |       | 2     | 0      |
| THEN | THE  |      |     |       | 3     | 0      |
| THEN | THE  |      |     |       | 4     | 0      |
| THEN | THE  |      |     |       | 5     | 0      |
| THEN | THE  |      |     |       | 6     | 2      |
| THEN | THE  |      |     |       | 7     | 0      |
| THEN | THE  |      |     |       | 8     | 0      |
| THEN | THE  |      |     |       | 9     | 1      |
| THEN | THE  |      |     |       | 10    | 0      |
| THEN | THE  |      |     |       | 11    | 8      |

(tümü eş)

Tablo 4.36 (b) 6809 programı

|     |        |      |        |                                                        |
|-----|--------|------|--------|--------------------------------------------------------|
| 1.  | SATIR  | EQU  | \$2000 | metin satırının başlangıç noktası                      |
| 2.  | DIZI   | EQU  | \$0080 | dizinin başlangıç noktası                              |
| 3.  | CRET   | EQU  | \$0D   | satır başı = CRET = satır-sonu                         |
| 4.  |        | LDX  | #SATIR | X, metin satırını gösterir                             |
| 5.  |        | LDY  | #DIZI  | Y, diziyi gösterir                                     |
| 6.  | SONR   | LDA  | , X +  | satırdan bir karakter al                               |
| 7.  |        | CMPA | #CRET  | satır-sonu (col) karakterini test et                   |
| 8.  |        | BEQ  | OLUMS1 | eğer karakter col ise, sonuç olumsuz                   |
| 9.  |        | CMPA | , Y    | dizinin ilk karakteriyle karşılaştır.                  |
| 10. |        | BNE  | SONR   | eğer eşleşmiyorsa satırın bir sonraki karakterini al   |
| 11. |        | PSHS | X, Y   | her iki işaretçiyi yığma it                            |
| 12. |        | LEAY | 1, Y   | dizideki 2.nci karakteri göster                        |
| 13. | TEKRAR | LDB  | , Y +  | diziden bir karakter al                                |
| 14. |        | CMPB | #CRET  | dizinin-sonu karakterini test et                       |
| 15. |        | BEQ  | OLUMLU | eğer karakter dizinin-sonu ise, eşleştirme olumlu      |
| 16. |        | LDA  | , X +  | satırdan bir karakter al                               |
| 17. |        | CMPA | #CRET  | col karakterini test et                                |
| 18. |        | BEQ  | OLUMS  | eğer bir col karakteri bulursak bulma ümidinden vazgeç |
| 19. |        | CMPA | - 1, Y | karakterleri karşılaştır                               |
| 20. |        | BEQ  | TEKRAR | aynıysa alt-eşlemeye devam et                          |
| 21. |        | PULS | X, Y   | farklıysa X ve Y'yi geriye al                          |
| 22. |        | BRA  | SONR   | satırdaki bir sonraki karakteri dene                   |
| 23. | OLUMS  | PULS | X, Y   | yığından değerleri geriye al                           |
| 24. | OLUMS1 | LDX  | #0000  | X kaydedicisini sıfırla                                |
| 25. |        | PSHU | X      | olumsuzluk durumu için kullanıcı yığına it             |
| 26. |        | RTS  |        | geriye dön                                             |
| 27. | OLUMLU | PULS | X, Y   | yığındaki değerleri geriye al                          |
| 28. |        | LEAX | - 1, X | satır işaretçisinin otomatik-artımını geri al          |
| 29. |        | PSHU | X      | işaretçiyi, yığındaki eşlemenin başlangıcına it        |
| 30. |        | RTS  |        | geriye dön                                             |

Aşağıdaki notlar 6809'un bazı komutlarıyla ilgili daha fazla bilgi vermektedir.

1. Akümülatöre ilişkin tüm göndermeler, bunları ayırt etmek için, A ya da B ile sonlanır.
2. İtme ve çıkarma komutları, kullanıyor oldukları yığına belirtmelidir.
3. Yukarıdaki programın 12. satırında X kaydedicisi 'LEAY 1,Y' ile artırılmıştır. Benzer şekilde 28. satırda 'LEAX - 1,X' işlemi X kaydedicisini azaltır. Ofsetin işaretli bir değer olduğuna dikkat ediniz.

## Problemler

Aşağıdaki üç soru Aim 65 mikrobilgisayarı olanlar için düşünülmüştür. Bunlar 6502-tabanlı mikrobilgisayarlara da uyarlanabilirler.

## Aim 65'e giriş

1. Aim 65'i sayıları bellekte saklayıp, yeniden okumak için kullanın. Not: Sayılar bayt başına iki onaltılı karakter olacak şekilde ifade edilmiş onaltılı formatta olmalıdır. Her bir bayt \$00 - \$FF arasında değişebilir. O harfi ile 0 (sıfır) sayısını birbirine karıştırmayın. Sayılarınızı \$0000 konumundan başlayarak saklayın. Örneğin iki onaltılı karakterin, A5, girilmesi 10100101'in bellekte saklanması neden olacaktır. Bir adresin, 6502'nin 16-bitlik adres yolu olması nedeniyle, 4 tane onaltılı basamağa gereksinim duyacağını anımsayın.
2. Tablo 4.37'deki program, 0000 ve 0001 bellek konumlarındaki sayıları birbirine ekleyip, sonucu 0002 konumuna yerleştirecektir.

Tablo 4.37

| Adres | Veri |                                            |
|-------|------|--------------------------------------------|
| 0000  | XX   |                                            |
| 0001  | XX   |                                            |
| 0002  | ZZ   |                                            |
|       |      |                                            |
| 0080  | A5   | } A'ya 0000'ın içeriğini yükle             |
| 0081  | 00   |                                            |
| 0082  | 18   |                                            |
| 0083  | 65   | } Elde bitini sıfırla                      |
| 0084  | 01   |                                            |
| 0085  | 85   | } Sonucu 0002'ye koy                       |
| 0086  | 02   |                                            |
| 0087  | 00   |                                            |
|       |      | BRK ('durma' veya işletim sistemine dönüş) |

Bu programı, belleğe girmek için 'M' fonksiyonunu kullanın. XX ve YY için kendi verilerinizi kullanın. Program sayıcınızı '\*' fonksiyonu ile \$0080'e kurduktan sonra 6 fonksiyonu ile programı çalıştırın. 0002 konumundaki sonucu incelemek üzere M fonksiyonunu kullanın. Programı önce basit sayılarla (sözelimi 04 ve 07) çalıştırın sonra (sözelimi AB, CD) gibi daha zor sayılar deneyin

3. 2. soruyu model olarak kullanıp Tablo 4.38'de listelenen komutların çalışmasını gözden geçirin.

Tablo 4.38

| Kom. Kısalt. | İşlem                   | Op-kod | İşlem (RTL)                             |
|--------------|-------------------------|--------|-----------------------------------------|
| ADC MM       | eldeli topla            | 65     | $[A] \leftarrow [A] + [MM] + [C]$       |
| AND MM       | mantıksal VE            | 25     | $[A] \leftarrow [A] \cdot [MM]$         |
| ASL MM       | aritmetik sola kaydırma | 06     |                                         |
| LSR MM       | mantıksal sağa kaydırma | 46     |                                         |
| SBC MM       | eldeli çıkar            | E5     | $[A] \leftarrow [A] - [MM] - [\bar{C}]$ |
| CLC          | eldeyi sıfırla          | 18     | $[C] \leftarrow 0$                      |
| SEC          | eldeyi kur              | 38     | $[C] \leftarrow 1$                      |
| LDA MM       | akümülatörü yükle       | A5     | $[A] \leftarrow [MM]$                   |
| STA MM       | akümülatörü sakla       | 85     | $[MM] \leftarrow [A]$                   |

## Adresleme modları

1. Aşağıdaki bellek haritası için şu ifadeleri hesaplayın. Burada [N], N bellek konumunun içeriği anlamına gelmektedir. Örneğin  $[3] = 4$ .

|                               |    |    |
|-------------------------------|----|----|
| (a) [7]                       | 00 | 12 |
| (b) [[4]]                     | 01 | 17 |
| (c) [[[0]]]                   | 02 | 7  |
| (d) $[2 + 10]$                | 03 | 4  |
| (e) $[[9] + 2]$               | 04 | 8  |
| (f) $[[9] + [2]]$             | 05 | 4  |
| (g) $[[5] + [13] + 2 * [14]]$ | 06 | 4  |
| (h) $[0] * 3 + [1] * 4$       | 07 | 7  |
| (i) $[9] * [10]$              | 08 | 0  |
|                               | 09 | 5  |
|                               | 10 | 12 |
|                               | 11 | 7  |
|                               | 12 | 6  |
|                               | 13 | 3  |
|                               | 14 | 2  |

2. Bir yığın işaretçisi ve iki indeks kaydedicili MİB'in blok diyagramını çizin. MİB'in çeşitli bölümleri arasında bilginin taşındığı yolları gösterin.

3. 2. sorudaki blok diyagram için, kaydedici transfer dilini kullanarak aşağıdaki işlemlerin yürütülmesi sırasında yapılacak işlemleri belirtin.

- |                     |                                                                                    |
|---------------------|------------------------------------------------------------------------------------|
| (i) PSH             | akümülatörü yığına it                                                              |
| (ii) PLA            | akümülatörü yığından çıkar                                                         |
| (iii) LDA OFFSET, X | akümülatöre indekslenmiş yükle                                                     |
| (iv) STA, Y --      | akümülatöre indekslenmiş (ikiye otomatik-azalım modunda) sakla                     |
| (v) ADD [OFFSET, X] | akümülatörün içeriğini dolaylı adresi indekslenmiş bellek konumunun içeriğine ekle |

4. Eğer bir bilgisayar tasarlıyor ve koşullu dallanmalar listesi çıkarıyor olsaydı (örneğin BEQ, BCC, BOV vs.) ne tür yeni koşullu atlamalar eklemek isterdiniz?

5. Bazı bilgisayarlarda aşağıdaki gibi bir indekslenmiş adresleme biçimi vardır:

LDA OFFSET, R1, R2

(R1 ve R2 iç kaydedicilerdir).

Bu komutun etkisi  $[A] \leftarrow [OFFSET + [R1] + [R2]]$  olup, R1 ve R2'nin değerlerinin toplamı gerçek adresi üretmek için kullanılır. Bu komut sizce nerelerde kullanılabilir?

6.

|   |    |    |    |
|---|----|----|----|
| X | 12 | 30 | 19 |
|   |    | 31 | 46 |
|   |    | 32 | 31 |
|   |    | .  | .  |
|   |    | 46 | 32 |

Yukarıdaki bellek haritası için, aşağıdaki komutların her biri yürütüldükten sonra akümülatörün içerdiği değeri söyleyin. Komutlar herhangi bir gerçek-makinedeki komutları göstermez; fakat metinde kullanılanlar benzerdir. Tüm değerler ondalıktır.

- (a) LDA 31    (b) LDA 32    (c) LDA 19, X  
(d) LDA [46]    (e) LDA 34, X    (f) LDA [34, X]

7. Birçok bilgisayarda JMP OFFSET, X gibi indekslenmiş bir atlama komutu vardır. Bu komutun nasıl çalışabileceğini belirtiniz.

8. Şekil 4.7'deki MİB yapısını; bir iç yığın işaretçi, bir de dış yığın içeren bir yığın mekanizmasını (yani 6502'de Hemen Erişilebilir Bellek; (IAS)'deki yığın örneğinde olduğu gibi) ekleyecek şekilde düzeltin. Veri ve adres yollarını gösterin ve gereken ek üç-durumlu kapılar varsa belirtin.

9. 7. soruya verdiğiniz cevap için PSH (akümülatöre it) işlemini belirtmek üzere RTL'de bir program yazın. Sonra da bu işlemin etkisini göstermek için, Tablo 4.10 türünde bir denetim sinyalleri tablosu oluşturun.

#### Assembly dili örnekleri

1. İki tane 16-bitlik işaretli ikili tamsayıyı toplamak için hem komut kısaltması, hem de onaltılı biçimde bir program yazınız. Toplanacak P ve Q sayıları aşağıdaki gibi saklıdır.

Adres  
0080  
0081  
0082  
0083  
0084  
0085

|    |            |
|----|------------|
| 12 | P = \$1234 |
| 34 |            |
| AB | Q = \$ABCD |
| CD |            |
|    | R          |
|    |            |

## Notlar:

(a) Programınızı \$0000 bellek konumundan başlatınız.

(b) ADC, CLC, LDA ve STA'nın işlem kodları sırasıyla 6D, 18, AD, ve 8D'dir (uygun yerlerde) 16-bitlik bir mutlak adrese gereksinim duyarlar. 6502 mikrolemcisi için adres, düşük değerlikli bayt ilk önce gelecek şekilde (yani alt-sıra adreste) saklanır.

Programı yazdıktan sonra, program sayıcısının, akümülatörün, işlemci durum sözcüğünün ve 0080 - 0085 arasındaki bellek konumlarının her komut yürütülmeden sonraki durumlarını gösteriniz.

2. İki tane 16-bitlik işaretli ikili tamsayıyı çıkarmak için bir program yazınız. Sorudaki formatı ve değerleri (yani R= Q-P) kullanın.

3. İki tane 8-bitlik işaretli ikili tamsayıyı çarpmak için bir program yazın. Çarpma \$0040 konumunda, çarpılan \$0041'de ve 16bitlik sonuç da \$0042 ve \$0043'de saklanacaktır. Çarpma, tekrarlamalı toplama ile yapılmalıdır.

Çarpılacak sayılar, klavyeden \$E3FD'deki RBAYT altyordamı ile giriş yapılabilir. Bu altyordam iki tane onaltılı karakteri girer ve bu sayıları akümülatörde yükler.

Akümlatördeki sayı \$EA46'daki altyordamı çağırarak iki onaltılı karakter olarak basılabilir. Satır atla ve yeni satır karakterini basmak için \$EA13'teki CRLOW altyordamını çağırın.

Bu altyordamı çağırmanın A, X ve Y kaydedici içeriklerinin değiştirilmesiyle sonuçlanabileceğine dikkat edin. Örneğin CRLOW'u çağırmak akümülatörün içeriğini, çağrıdan önce yok etmek etkisine sahiptir. Bundan kaçınmak isteniyorsa, akümülatörün içeriği çağrıdan önce saklanıp, sonra da geriye alınmalıdır. Aşağıda akümülatörün içeriğinin altyordam içinde kullanılmasıyla bozulmasını önlemenin iki yöntemi gösterilmiştir:

| 1. Yöntem  | 2. Yöntem  |
|------------|------------|
| STA GECICI | PHA        |
| JSR \$EA13 | JSR \$EA13 |
| LDA GECICI | PLA        |

## İndeks kaydedicisinin kullanımı

1. Klavyeden bir metin girip (\$E993'teki INALL altyordamını kullanın) karakterleri \$0200'den başlayarak ardışık konumlarda saklayacak bir assembly dili programı yazın. Yükleme işlemi, nokta karakteri girildiğinde sona ermelidir. Noktanın ASCII kodu \$2E'dir. Nokta girildiğinde yeni bir satır çıkışı yapılacaktır (\$EA13'deki CRLOW'u kullanın) ve saklı metin basılacaktır. (Bir karakter basmak için \$E9BC'deki OUTALL'u kullanın). Programı \$0040 konumuna yerleştirin. Programı çalıştırdıktan sonra \$0200'den itibaren bellek konumlarını inceleyerek girdiğiniz karakterleri içerdiklerini gözden geçirin. Programınızı; saklı veriyi, girildiği sıranın tersinde basmak üzere değiştirin.

2. Bir dizi baytı girip (\$E3FD'deki RBYTE'ı kullanın) bunları \$0200'den başlayarak sıralı biçimde saklayın. Dizi 00 ile sonlandırılacaktır. Çift sayıları, \$EA46'daki NUMA'yı kullanarak sırayla basınız. Bunu yapmak için her bir sayıyı test etmek gerekir - çift ise bas, değilse diğer sayıya geç. Tek/çift testi için, en düşük ağırlıklı biti, elde flip-flopuna kaydırın.

3. Bir grup 8-bitlik işaretli tamsayı, artan sırada \$0000 - \$000F bellek konumları arasında saklıdır. Bir başka sayı \$0020'de saklı olup, dizi içinde doğru yerine yerleştirilecektir. Böylece dizi \$0000 - \$0010 konumları arasını kaplamış olacaktır. Bunu yapmak için bir program yazın ve kendi verinizi kullanarak test edin. Programınızı \$0080'den başlayarak yerleştirin.

4. Bir nokta ile sonlandırılmış bir metin satırı okumak ve sesli harfleri saymak üzere bir program yazın. Örneğin 'YAŞAM EVREN VE HER ŞEY' satırının girilmesi şu çıkışla sonuçlanmalıdır:

A = 1  
E = 6  
I = 3  
O = 0  
U = 1

Bunun için, aşağıda verilen altyordamlar yararlı olacaktır:

|        |        |                                               |
|--------|--------|-----------------------------------------------|
| CRLOW  | \$EA13 | satır atla ve yeni satır basma                |
| INALL  | \$E993 | akümülatöre bir ASCII karakteri gir           |
| OUTALL | \$E9BC | akümülatördeki ASCII karakteri yazıcıya çıkar |

Boşluk karakterinin ASCII kodu \$20, '-' inki ise \$3D'dir. OUTALL altı akümülatörün içeriğini ilgili ASCII karakter biçiminde çıkış yaptığında, değerlerin, basılmadan önce ASCII eşdeğerlerine çevrilmesi gerekir. Çıktı herhangi bir seslinin 9'dan daha fazla olmayacağını varsayın. Dokuzdan tamsayılarla işlem yapabilmek için sayısaldan ASCII'ye dönüştürme işleminin hangi değişikliğe gereksinim duyulur?

## GİRİŞ / ÇIKIŞ (INPUT/OUTPUT)

Şimdiye kadar sayısal bilgisayarın iç yapısını ve çalışmasını inceledik. Eğer bilgisayarlar, insanlar için bir değer ifade edecekse, onlarla iletişim kurabilecek bazı olanaklara sahip olması gerekir. Herşeyin ötesinde yaşamın, evrenin ve herşeyin en zor sorusunu çözebilen ayakkabı kutusu büyüklüğünde süper bir bilgisayar yaratmanın, eğer cevabı bize söyleyemeyecekse, hiçbir değeri yoktur.

Bilgisayar giriş/çıkışı (G/Ç) (input/output (I/O)) biraz karmaşık bir konudur ve en iyisi aşağıdaki gibi üç kısma ayırmaktır :

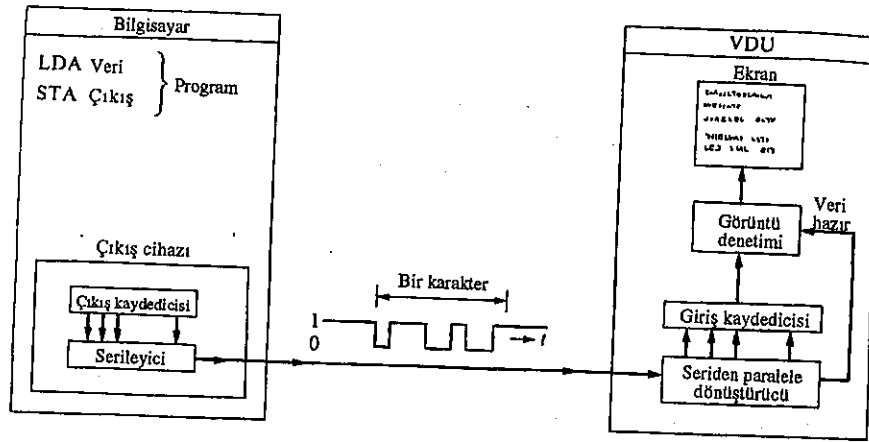
1. Bilginin (verinin) bilgisayara ve ondan dışarıya taşınma stratejisi
2. Uygulamada veriyi taşıyan donanım
3. Veriyi kullanılabilir biçimlere (veya tersi) dönüştüren G/Ç birimlerinin kendileri. Kullanılabilir biçimler terimini kullanıyorum çünkü veriler, insan sesine oldukça yakın bir benzerlikten bir kimya fabrikasındaki valfi açan veya kapatan bir sinyale kadar neredeyse sonsuz sayıdaki biçime dönüştürülebilirler. G/Ç birimleri çoğu kez çevrebirimleri olarak adlandırılırlar.

Bunlar arasındaki fark örnekler aracılığıyla kolayca gösterilebilir : Önce bir ekran birimine (VDU veya ABD terminolojisindeki biçimiyle CRT terminali) bağlı küçük bir bilgisayarı ele alalım. Veriler bilgisayara veya bilgisayardan dışarıya, programlı veri aktarımlarıyla taşınırlar. Bilgisayarın VDU'ya ilişkin herhangi bir veri içerdiği zamanlarda, programda yer alan bir komut, verinin, VDU ile iletişimde olan çıkış portuna yazılmasını sağlar. Benzer biçimde bilgisayar veriye gereksinim duyduğunda da bir başka komut giriş portundan veriyi okur. Port terimi bilgisayar ile dış bir G/Ç birimi arasındaki arabirimi belirtir. Programlı veri aktarımı veya programlı G/Ç, stratejiyi temsil eder.

Kolay olması amacıyla sadece, bilgisayardan çıkışı ele alalım. Bilgisayar çıkış portuna bir sözcük gönderdiğinde, çıkış portu bu veriyi VDU'ya gönderir. Çıkış portu normalde karmaşık yapıda bir entegre devredir ve yapısındaki bu karmaşıklık neredeyse MİB'in karmaşıklığına yakındır. Bu türde bir 'yarı-akıllı' cihaz bilgisayarın, VDU ile doğrudan iletişimde bulunma eylemindeki yükünü hafifletir ve ona daha yararlı hesaplamalar yapma olanağı sağlar. Pratikte bilgisayar ile VDU arasındaki bağlantı, genelde bükülü bir çiftten oluşur (bükülmüş iki paralel kablo). MİB tarafından çıkış portuna yazılan veri paralel biçimde olduğu sürece (sözgelimi

8-bit) çıkış portu veriyi seri biçime dönüştürüp bükülü kablo çifti üzerinden -birer birer olmak üzere VDU'ya göndermelidir. Bundan başka çıkış portu, VDU'nun kendisinin bilgisayardan gelen bit akışı ile senkronize edebilmesi için başlatma ve durdurma bitleri sağlamalıdır. Kısım 8.1'de, seri veri iletimi daha ayrıntılı olarak ele alınacaktır. Böylece çıkış portu, gerçekte, veriyi işlemci ile çevrebirimi arasında taşımaktan sorumlu olan aygıttır.

VDU, çıkış birimi özelliğindedir. Bilgisayardan seri veri kabul eder, onu paralel biçimde yeniden oluşturur ve bu bilgiyi semboller tablosundan bir karakteri seçmek için kullanır. Daha sonra semboller televizyon tipi taramalı bir ekranda görüntülenir. Gönderilen karakter kimi zaman ekrana yeni bir karakter eklemekten çok onun formatını etkileyen bir denetim sembolü de (örneğin satır-başı, satır-atlatma, gerial) olabilir. Şekil 5.1'de, yukarıdaki örnekte belirtilen kavramlar arasındaki ilişki gösterilmiştir.



Şekil 5.1 MİB ve VDU arasındaki ilişki

Bir başka örnek olarak, bir veri bloğunun diske yazılmasını ele alalım. Bu durumda çoğu kez, ya çok yavaş ya da önemli bir hesaplama zamanını alan programlı veri aktarımı kullanmak pratik olmayacaktır. Bu konuda yardımına en çok başvurulacak çıkış stratejisi doğrudan bellek erişimidir (DMA). Burada veri bilgisayarın ana belleğinden bir çevrebirimine (ya da çevrebiriminden ana belleğe) MİB'in yazmaçlarından geçmeden doğrudan aktarılır. MİB basitçe DMA donanımına bir blok veri taşımasını söyler ve DMA donanımı da MİB'in bilgi işleme temel işlevine devam etmesini sağlamak üzere kendi işini sürdürür.

DMA donanımına DMA denetleyicisi (DMAC) denir ve pratikte veriyi bellek ile çevrebirimi arasında taşımaktan sorumludur. Verinin RAM'daki kaynak veya

hedef adresleri ile çevrebirimi için verinin istendiğini veya hazır olduğunu belirtmek üzere bir sinyal sağlamalıdır. Bundan başka DMAC, bilgisayarın iç veri ve adres yollarını veri aktarımı boyunca denetim altında tutmalıdır. Bu, MİB ile yolların kullanımında bir çakışmaya yol açmayacak şekilde yapılmalıdır. Bu örnekte çevrebirimi, bir disk sürücüdür. Disk sürücüler, yüksek bir hızda dönen bir diskin yüzeyinin manyetik özelliklerini kısmi olarak etkileyerek veriyi saklamak için tasarlanmış elektronik ve yüksek duyarlılıklı makine mühendisliğinin kompleks bir karışımıdır.

### Giriş/çıkış stratejileri

G/Ç işlemlerini gerçekleştirmenin dört temel yolu vardır :

- 1- Programlı G/Ç
- 2- Kesme ile sürülen G/Ç
- 3- Doğrudan bellek erişimi
- 4- Kanal G/Ç

Bu kısımda bu G/Ç modlarının her birinin çalışmasını tek tek inceleyeceğiz. Burada ne bu stratejileri gerçekleştirmek için gereken donanımın ayrıntılarıyla ilgileneceğiz ve ne de çevrebirimlerinin kendileriyle. Bu stratejileri ele almadan önce, her türlü bilgi değiş-tokuşunun temeli olduğu için onay sinyalleri konusu tanıtılacaktır.

### Onay

Verinin işlemci ile çevrebirimi arasındaki taşıma stratejisinden bağımsız olarak tüm veri aktarımları iki sınıftan birine girer: açık uçlu veya kapalı döngü. Açık uçlu bir G/Ç işleminde çıkarılacak veri tek başına gönderilir ve güvenli biçimde kabul edileceği varsayılır. Bu postahanedeki sunulan temel hizmet düzeyine karşılık gelir. Bir mektup yazılır ve posta kutusuna atılır. Gönderici, makul bir süre geçtikten sonra mektubun alınacağına inanır.

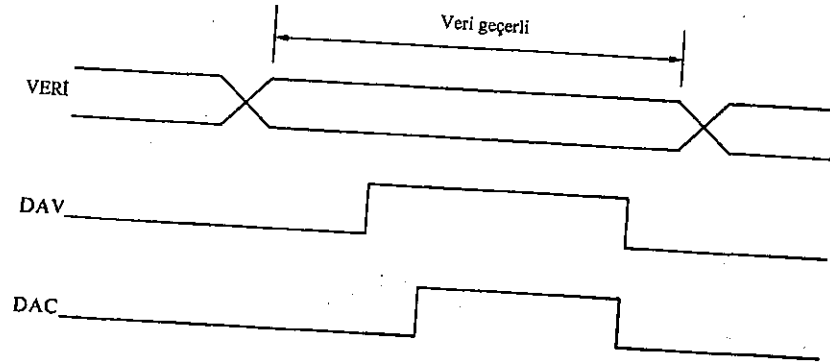
Pek çok sistemde, hem insan hem de bilgisayar, açık uçlu veri aktarımı yeterince tatminkardır. Verinin kaybolması veya bozulması olasılığı çok düşüktür ve kaybı o kadar da önemli olmayabilir. Örneğin, Ahmet Bey, kendisine gönderilen tebrik kartını doğum gününde hala almamışsa elbette dünyanın sonu gelmeyecektir. Şimdi de aşağıdaki bilgi değiş tokuşunu ele alalım :

Kontrol Kulesi: 'Golf Zulu Victor 23 numaralı piste iniş için Teessi yaklaşmanıza izin verildi. Rüzgar 270 derece 10 knot @ 1019. Görüş alanını bildir.

Uçak: 'Pist 23 QFE 1019 Golf Zulu Victor'

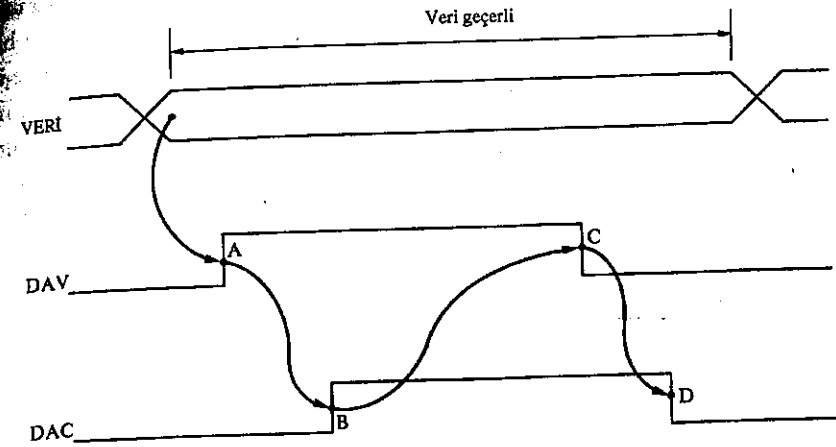
Bu durumda bilginin kaybı veya bozulması muhtemelen felaket getirir ve verinin açık uçlu aktarımı yerine bir kapalı döngü sistemi tercih edilir. Uçak, mesaj alındığını bildirir ve çok önemli herhangi bir veriyi tekrar okur. Veri iletilen sistemlerinde (8. Bölüme bakınız) veri bütünlüğü mesajın tekrarlanmasıyla değil de bazı hata algılama kod biçimleriyle sağlanır.

Bilgisayar ile çevrebirimi arasındaki bir kapalı-döngü veri aktarımında verici (yani verinin kaynağı) önce veriyi hazır hale getirir, sonra verinin geçerli olduğunu belirtmek üzere bir sinyal çıkarır, DAV (Veri kullanılabilir). Veriyi alan birim, verinin hazır olduğunu belirten DAV'ın çıkarılmış olduğunu görür ve veri yolundan veriyi okur. Daha sonra verinin tatmin edici olarak alındığını belirtmek üzere vericiye cevap niteliğinde DAC (veri alındı) adlı bir sinyal çıkarır. Verici, daha sonra DAV'ı geri alır. Tüm bu işlem onaylama olarak bilinir. Verinin alındığını belirtmenin yanı sıra bu, aynı zamanda yavaş çevrebirimlerin gereklerini de karşılar çünkü aktarım, birim DAC sinyali üreterek hazır olduğunu belirtene dek bekletilir. Şekil 5.2'de onaylamanın nasıl yapıldığı görülmektedir.



Şekil 5.2 Bir Onaylama örneği

Şekil 5.3'te onay işleminin; alındı bildiriminin kendisinin de alındığını bildirerek nasıl bir adım daha ileri götürülebileceği görülmektedir. Buna tam-kilitleşimli veri aktarımı denir. A noktasında verici, verinin hazır olduğunu belirtmek üzere bir DAV sinyali çıkarır. B'de alıcı, DAV'ın gözlendiğini ve verinin alındığını belirt-



Şekil 5.3 Tam kilitleşimli onaylama

mek üzere bir DAC sinyali çıkarır. Buraya kadar yapılanlar, Şekil 5.2'de belirtilenlerle aynıdır. Verici DAC'ın doğru olduğunu görür ve verinin artık geçerli olmadığını ve alıcının veriyi aldığını bildirdiğini belirtmek üzere DAV'ı geriye alır. Son olarak da D'de alıcı, çevrimi tamamlamak ve vericinin seri alındı bildirimini gördüğünü belirtmek için DAC'ı geri alır.

Kapalı-döngü veri aktarımı kullanan birçok gerçek sistemde, onaylama işlemi tümüyle özel amaçlı donanımlar aracılığıyla otomatik olarak gerçekleştirilir. Bilgisayarın kendisi işe karışmaz. Ancak eğer birşeyler hatalı giderse işlemci devreye girer.

Onaylamayı kullanan herhangi bir veri aktarımında, eğer vericinin çıkardığı DAV sinyaline karşılık bir DAC çıkarılmazsa, bir problem doğar. Makul bir sistemde DAV çıkarıldığında bir zamanlayıcı çalıştırılır ve eğer verilen süre geçtiği halde DAC çıkarılmamışsa işlem durdurulur. Bu durumda, bilgisayarı bir bir önlem almaya zorlamak üzere bir kesme (bir sonraki kısma bakınız) üretilir. Makul olmayan bir sistemde ise, onaylamanın tamamlanmaması için beklenmesi, verinin sonsuza dek bir DAC beklemesine neden olur. Bu nedenle bu duruma sistem kilitlendi denir.

## 5.1 Programlı G/Ç

Programlı G/Ç'de (Input-output I/O) veri aktarımı, programdaki bir komut ile başlatılır. Bazı mikroişlemcilerde (örneğin Intel 8080) ayrı G/Ç komutları vardır.

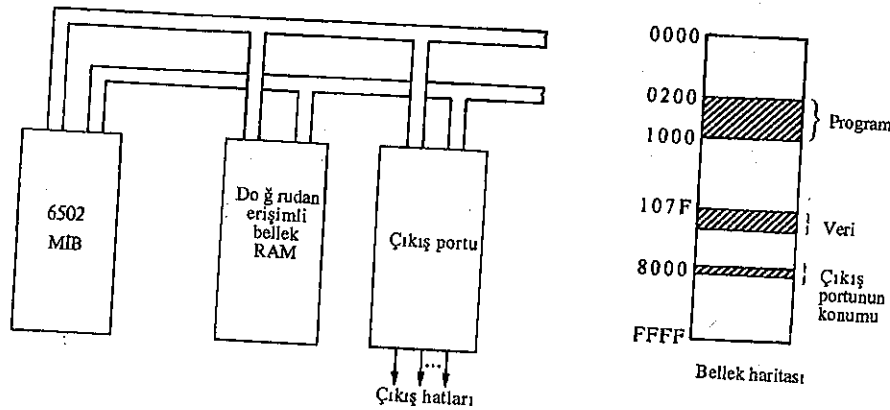
Örneğin 8080'de bir 'OUT 123' komutunu yürütülmesi; akümülatörün içerdiği değerin veri yoluna, 123 sayısının adres yolunun en küçük değerlikli 8 biti ile yerleştirilmesine ve sistemin  $\overline{IOW}$  hattında (giriş/çıkış yazma) bir darbe üretilmesine neden olur. Verilen bir sistemde çevrebirimlerin her biri adres hatlarını gözetir. Daha doğru olması açısından pratikte bu işlevi yerine getiren, çevrebirimlerle bağlı olan G/Ç portlarıdır. Bir G/Ç portu, kendi adresiyle birlikte bir portu okuyucu portta yaz sinyalini gördüğünde sinyalin gösterdiği doğrultuda davranır.

Bazı mikroişlemciler (özellikle 6800, 6809, 6502) herhangi bir G/Ç komutu içermez. Eğer bu birimler programlı G/Ç kullanacaksa, bellek haritalı G/Ç'nin yardımına başvurmalarıdır. Yani, MIB'in normal bellek alanının bir kısmı G/Ç işlemlerine ayrılır. Bu, G/Ç portunun normal bir bellek konumu gibi görülmesi anlamına gelir. Bu sistemin bir dezavantajı, programlar ve veri için mevcut olan bellek alanının bir kısmının G/Ç sistemi için kaybedilmesi.

Bir VDU'ya bağlı ve bellek konumu \$8000 olan çıkış portuna sahip bir 6502 mikroişlemcisi ele alalım. Bir başka deyişle, \$8000 bellek konumuna bir sözcük saklamak demek onu, VDU'ya göndermek demektir. İşlemciye gelince, o yalnızca belleğe bir sözcük saklar. \$0200'deki bir program (\$1000'den başlayarak), VDU'ya 128 karakter gönderir. Sistem ile bellek haritası Şekil 5.4'te verilmiştir.

Tablo 5.1'deki program verinin çıkışını yapacaktır.

Sağdaki sütunda yeralan sayılar, 1 Mhz.'lik bir saat hızı olduğu varsayılarak, her bir komutu yürütmek için gereken zamanı mikrosaniye cinsinden vermektedir. 128 sözcüğün çıkışını yapmak yaklaşık olarak  $128 \times (4+4+2+2) = 1792$  mikrosaniye olacaktır. Bu da, saniyenin ikibinde birinin biraz altındadır.



Şekil 5.4 Bellek-haritalı G/Ç

Tablo 5.1 Programlı çıkış

| Komutlar            | Çevrimler                      |
|---------------------|--------------------------------|
| LDX #00             | [X] ← 0                        |
| DONGU LDA \$1000, X | [A] ← [[X]+1000] veriyi al (4) |
| STA \$8000          | [8000] ← [A] veriyi çık (4)    |
| INX                 | [X] ← [X]+1 (2)                |
| CPX #\$80           | [X] ← 80 bitiş testi (2)       |
| BNE DONGU           | EĞER Z=0 İSE [PC] ← DONGU (2)  |

Bu program, teorik olarak çalışacaksa da, programlı çıkışla ilgili tüm pratik durumlar için hemen hemen uygun değildir. Bir çıkış portuna bağlı çevrebirimlerin çoğu nispeten yavaştır ve bu hızdaki veri çıkışı en basitinden, kaybolacaktır. Öte yandan bazı çevrebirimleri yüksek hızda verinin kısa çıkışları ile işlem yapabilir çünkü veriyi yavaş yavaş işlerken onu bir tamponda (ilk-giren ilk-çıkart kuyruğu olarak düzenlenen geçici bir bellek) tutar.

Bilgisayar ile çevrebirimi arasındaki hız uyumsuzluğu probleminde bir çözüm, çevrebirimine veri için hazır olup olmadığını sorularak ve almaya hazır olana dek veriyi bekleterek bulunur. Neredeyse tüm bellek haritalı G/Ç portları iki veya daha çok bellek konumu kaplar. Konumlardan biri, girişi veya çıkışı yapılacak veri için ayrılır; diğeri de portla ilgili durum sözcüğünü tutar. Örneğin, \$8000 verinin gönderdiği, \$8001 de durum sözcüğünün tutulduğu konum olsun. Durum sözcüğünün sıfırıncı bitinin, port veri için hazırsa bir, hazır değilse sıfır olduğunu varsayalım. Tablo 5.2'de verilen programda bu durum dikkate alınmıştır.

Bu program üçüncü satırdan yedinci satıra kadar olan kısım hariç, bir önceki program ile aynıdır. Üçüncü satırda çıkışı yapılacak veri, geçici olarak yığına itilir. Bunun nedeni akümülatörü kullanmak istiyor olmamızdır ve bu yüzden de içerdiği değeri saklamamız gerekmektedir. Dördüncü satırda çevrebiriminin durum sözcüğü okunur ve altıncı satırda, eğer çıkış portu meşgulse dördüncü satıra bir dallanma

Tablo 5.2 Programlı çıkış ve tarama-döngüsü

|     |                     |                                                    |
|-----|---------------------|----------------------------------------------------|
| 1.  | LDX #00             | [X] ← 0                                            |
| 2.  | DONGU LDA \$1000, X | [A] ← [[X]+1000]                                   |
| 3.  | PHA                 | [[SP]] ← [A], [SP] ← [SP]-1 (A'yı yığına sakla)    |
| 4.  | BEKLE LDA \$8001    | [A] ← [8001]                                       |
| 5.  | AND #1              | [A] ← [A].00000001 (EKDB hariç tümünü maskle)      |
| 6.  | BEQ BEKLE           | IF Z=0 THEN [PC] ← BEKLE                           |
| 7.  | PLA                 | [SP] ← [SP]+1, [A] ← [[SP]] (A'yı yığından çıkart) |
| 8.  | STA \$8000          | [8000] ← [A]                                       |
| 9.  | INX                 | [X] ← [X]+1                                        |
| 10. | CPX #\$80           | [X] ← 80                                           |
| 11. | BNE DONGU           | IF Z=0 THEN [PC] ← DONGU                           |

yapılır. Port serbest olduğunda dallanmaya gidilmez ve veri, yedinci satırda yığından geriye alınır. Ve program daha önce olduğu gibi devam eder.

Dört, beş ve altıncı satırdaki işlemlere sırayla tarama döngüsü denir. Çıkış birimi, programın devam etmesine imkan tanımak için serbest olduğunu belirtene kadar sürekli olarak taranır (sorgulanır). Tipik bir ucuz yazıcı saniyede 30 karakter veya yaklaşık 33.000 mikrosaniyede bir karakter hızında çalışır. Eğer tarama döngüsü 4 mikrosaniye alıyorsa döngü her bir karakter için 4700 kere yürütülür. Kabaca söylenirse bilgisayar, yetersiz bir modda çalışmaktadır. Bu her zaman kötü bir şey değildir. Küçük bir mikroişlemci-tabanlı bilgisayarın (örneğin Pet, Apple, Aim 65), veri girişi veya çıkışı sırasında, çoğu kez 'yapacak daha iyi bir şeyi' yoktur. Öte yandan çoklu programlama ortamında çalışan bir anabilgisayar G/Ç portu meşgulken başka birinin programına devam edebilir. Bu durumda daha iyi bir G/Ç stratejisi, veri aktarımına hazır olana kadar çevrebirimini yok saymak ve sonra çevrebirimine MİB'in dikkatini çekmek için sorgulatmaktır. Bu tip bir stratejiye kesme ile sürülen G/Ç denir.

## 5.2 Kesme ile sürülen G/Ç

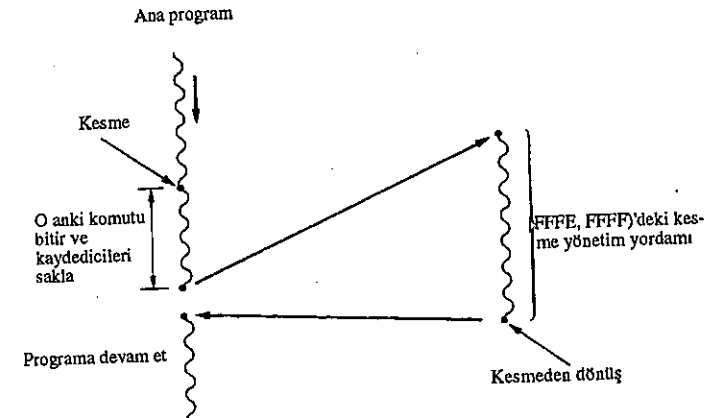
Bilgisayar, bir atlama veya dallanma yapılmadığı sürece komutları sıra ile yürütür. Bu kuralın kesme adı verilen önemli bir istisnası vardır. Kesmeler, MİB'ni, eğer kesme ortaya çıkmasaydı normal sırasında yürütülecek olan, işlem dizisini değiştirmeye zorlayan bir olaydır. Biz burada tümüyle donanım kesmelerini ele alacağız ve esasen işletim sistemine programlı bir çağırma (veya atlama) olan yazılım kesmelerini dahil etmeyeceğiz.

Pek çok mikroişlemcinin kesme istek girişi denen bir veya daha çok giriş hattı vardır. 6502'de bu hat  $\overline{IRQ}$  (kesme isteği değil diye okunur) olarak gösterilir.  $\overline{IRQ}$ 'nun üzerindeki çizgi hattaki alçak (mantıksal 0) düzeyin kesme isteğini ifade eder. İstek sözcüğünün, bünyesinde kesme isteğinin kabul edilip edilmeyeceğini taşıdığına dikkat edin. 6502'nin işlemci durum sözcüğünde 2 bit, kesme bayrağıdır ve kurulduğu zaman kesme isteklerinin dikkate alınmamasına neden olur.

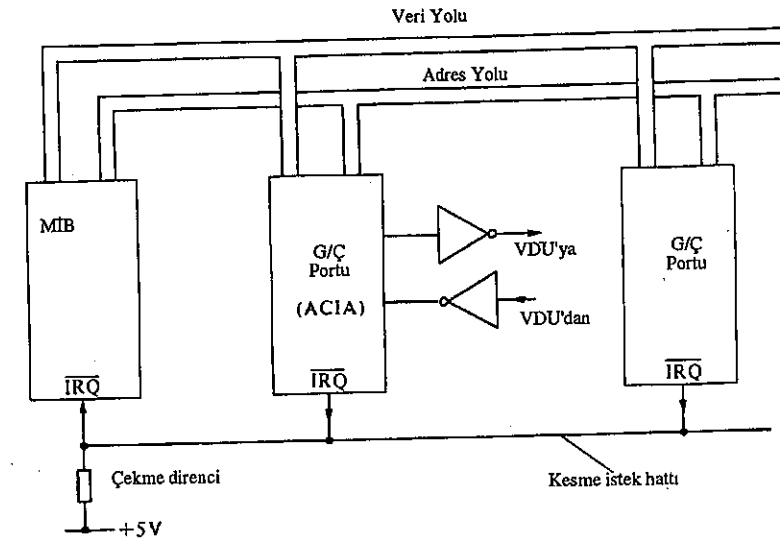
6502'nin  $\overline{IRQ}$  hattı alçak düzeye geçtiğinde ve PSW'nun kesme maskeleyme bitinin sıfırlanmış olduğunu varsayarsak, aşağıdaki olaylar meydana gelir. Önce MİB'in yürütmekte olduğu komutu tamamlamasına izin verilir. Tüm 8-bitlik mikroişlemciler ve daha büyük birçok bilgisayarlar yürütmenin ortasında durdurulamazlar. Makine komut kodu bölünemez ve sonuna kadar yürütülmelidir. İkinci olarak, program sayacının ve program durum sözcüğünün içerdiği değerler yığına itilir.

Üçüncü olarak PSW'nin kesme maskeleyme biti daha başka kesmeleri durdurmak (yetkisini kaldırmak) için kurulur. Son olarak, program sayıcısına \$FFFE ve \$FFFF bellek konumlarının içerdiği değerler yüklenir (\$FFFF adresin en büyük değerlikli baytını, \$FFFE de EKDB'yi tutar).

Şimdi MİB, \$FFFE ve \$FFFF'nin içerdiği değerlerin gösterdiği kesme yönetim yordamını yürütür. Bu yordamın sonunda bir RTI (kesmeden dönüş) komut yürütülür. Bu komut; MİB'in sanki kesme hiç olmamış gibi işine devam edebilmesi için, yığına saklanan kaydedici içerikleri geriye alınır ve bu değerleri MİB kaydedicilerine aktarır. Kesmeleri, dıştan gelen bir olayla ( $\overline{IRQ}$ 'nun alçak'a düş-



Şekil 5.5 Bir kesme sırasında meydana gelen olaylar dizisi



Şekil 5.6 6502 sisteminde kesme yönetimi mantığı

mesi) bir programın içine sıkıştırılan veya zorlanan bir altyordam gibi düşünme hoşuma gider. Şekil 5.5'te 6502 sistemindeki bir kesme eylemi gösterilmektedir. Bir 6502 sisteminde kesme ile sürülen bir G/Ç 1'i gerçekleştirmek için gerekli donanım bir örnek Şekil 5.6'da verilmektedir. Burada MİB'in  $\overline{IRQ}$  hattı bir direnç tarafından mantıksal 1'e çekilir. Herhangi bir G/Ç aygıtı bir kesme ürettiğinde,  $\overline{IRQ}$  hattını mantıksal 0'a çeker. Şekil 5.6'nın basitliğine dikkat edin - küçük bir mikroişlemci sisteminde kesme ile sürülen G/Ç'yi gerçekleştirmek için çok az bileşene ihtiyaç vardır. Bu düzenleme ile bir trendeki alarm kolu arasında bir benzerlik kurulabilir. Çekildiğinde, vatman sadece dikkatinin çekildiğinin farkına varır ancak, kolu kimin çektiğini bilmez.

Kesme ile sürülen G/Ç, programlı G/Ç'dan daha karmaşık bir programa gereksinim duyar. Bunun nedeni, bilgi aktarımının programcının istediği veya beklediği zaman değil de, verinin hazır olduğu zaman gerçekleşmesidir. Tipik bir kesme yönetim çıkış yordamının bir parçası Tablo 5.3'te listelenmiştir.

Tablo 5.3 Kesme sürücülü çıkışın işlenmesi

|               |                      |
|---------------|----------------------|
| LDX ISARETCI  | [X] ← [ISARETCI]     |
| LDA TAMPON, X | [A] ← [TAMPON + [X]] |
| STA \$8000    | [8000] ← [A]         |
| INX           | [X] ← [X] + 1        |
| STX ISARETCI  | [ISARETCI] ← [X]     |
| RTI           | kesmeden dönüş       |

Kesme yönetim yordamı her çağrıldığında, veri bir tampondan alınır ve \$8000'deki bellek haritalı çıkış portuna gönderilir. Gerçek bir sistemde, tamponun bittiğini test etmek için bazı denetimlere gerek vardır.

Kesmelerin temel düşüncesi pek çok bilgisayar için ortak olsa da kesme yönetim yapısının asıl doğasında bilgisayardan bilgisayara önemli ölçüde farklılıklar bulunur. Bu farklılıklardan bazıları aşağıda listelenmiştir.

**Kesme alındısı** Çevrebirim bir kesme ürettiğinde bilgisayarın, kesmenin alındığını ve ona göre davranılacağına ilişkin olarak çevrebirimine bir sinyal göndermesi kim zaman işe yarar. Bu olanak, alındı bildiriminin, MİB'e hangi belli kesme yönetim yordamının kullanılacağı ile ilgili ayrıntıları sağlamak için düzenlenebildiği ve kısıtlandırılmış kesmelerde özellikle yararlı olmaktadır. Bazı mikroişlemcilerin (örneğin 8080) bir kesme alındı mekanizması varken diğerlerinin (örneğin 6800, 6502) yoktur. Elbette ki 6502 için sözde bir kesme alındısı türetmek mümkündür. 6502'nin bir kesmeye program sayıcısına \$FFFE ve \$FFFF konumlarının içerdiği

değerleri yüklemekle karşılık verdiğini söylemiştik. Bu konumlardan bir okumayı tuzağa düşürerek (yani algılayarak) bir kesme alındısı sağlayabiliriz.

**Maskelenemez kesme** Bazı mikroişlemcilerde (örneğin 6502, 6800, Z80) bir maskelenemez kesme ( $\overline{NMI}$ ) girişi vardır. İşlemci durum sözcüğündeki bir bitin kurulmasıyla maskelenememesi (kapatılması veya yetkisinin kaldırılması) dışında  $\overline{IRQ}$  gibi davranır. Bu tür kesme çoğu zaman iki uygulamada yer alır. İlk olarak, kesmenin kaçırılmaması gereken bir olay tarafından oluşturulduğundadır. Tipik olarak bu, güç kaynağından gelen bir kesmedir. Böyle olduğunda sistem, kondansatörlerindeki (tüm güç kaynaklarında bulunan enerji depolama aygıtları) yük ile birkaç mikrosaniye daha çalışır. Güç kaynağının ilk işaretinde üretilen bir maskelenemez kesme, bilgisayarın düzenli bir şekilde kapatılmasını sağlamak için düzenlenir. Böylece bilgisayar daha sonra çok az bir veri kaybı ve disk dosyalarında bir bozulma (kazara üzerine yazma) olmadan yeniden açılabilir.

Maskelenemez kesmelerin ikinci bir uygulaması, gerçek-zamanlı sistemlerdedir. Karmaşık bir kimyasal işlemde sıcaklık ile basıncın değişik noktalarda ve düzenli aralıklarla ölçülmesi gerektiğini varsayalım. Eğer bu iş, noktaların her birinin sırayla taranması programlı bir temel ile yapılmazsa düzenli aralıklarla bir NMI akışı istenen sonucu verecektir. Her bir NMI'de bir kaydedicinin içerdiği değer güncelleştirilir ve uygun bir zaman aralığı geçmişse okuma yapılır. Kaydedicinin içerdiği değer ile gösterilen günün saatinin bir kopyasını korur.

**Öncelikli kesmeler** Bazı bilgisayarlarda birden fazla kesme istek hattı vardır. Bunlardan çok azı anında ilgi isteyen çevrebirimlerine bağlanırken (örneğin bir disk sürücüsü) diğerleri daha az ivedilikte ilgi isteyen çevrebirimlerine (örneğin bir VDU'ya) bağlanır. Eğer disk sürücüsü veri kullanılır durumdayken ilgilenilmezse (hizmet verilmezse) veri hemen kaybolacaktır. Veri kaybolur çünkü yerini yeni bir veri alacaktır. Bu gibi durumlarda her bir kesme istek hattına bir öncelik atamak oldukça mantıklıdır. Örneğin bir işlemcide ilki  $\overline{IRQ0}$ 'dan (en önemli) sonuncusu  $\overline{IRQ7}$ 'ye kadar sekiz tane kesme istek hattı bulunabilir. Eğer  $\overline{IRQ3}$ 'ün neden olduğu bir kesme varsa ve askıda başka kesme yoksa buna hizmet verilir. Eğer  $\overline{IRQ0}$ 'dan  $\overline{IRQ2}$ 'ye kadar kesmeler etkinleştirilirse bunlar kesmeyi (yani  $\overline{IRQ3}$ 'ü) kesecek ve hizmet alacaklardır. Öte yandan  $\overline{IRQ4}$ 'ten  $\overline{IRQ7}$ 'ye kadar hatlardan biriyle üretilen kesmeler,  $\overline{IRQ3}$ 'ün bitimine dek askıda bekletileceklerdir. NMI'nin aslında yüksek öncelikli özel tür bir kesme olduğuna ve daima hemen hizmet verilmesi gerektiğine dikkat edin.

**Taranmış kesmeler** 6502'nin tek bir  $\overline{IRQ}$  hattı vardır ve kesme üretebilen çevrebirimlerinin  $\overline{IRQ}$  çıkış bacakları MİB'in  $\overline{IRQ}$  hattına bağlıdır. Bunun sonucu olarak,

bir çevreirim  $\overline{IRQ}$ 'yu alçak düzeye çekerek bir kesme ürettiğinde, MİB kimin sorumlu olduğunu anında belirleyemez. Bu zorluğu aşacak bir yol rimlerin durum kaydedicilerini taramaktır. Bu elbette kesme yönetim yordamında yapılır. Pek çok çevreiriminde bir veya daha çok kesme istek bayrağı içeren bir durum kaydedicisi vardır. Eğer bir çevreirimi kesmeye neden olan kesme bayrak biti kurulacaktır. Bir kesmenin ardından yapılan çevreirim taramasının sırasını programcı seçse de tarama işlemine sınırlı ölçüde bir öncelik olabilir. Eğer bir kesmeye neden olabilecek birçok birim varsa tarama çok yavaş olabilir. Genelde herkesçe bilinen bir kural vardır ki, o da, bir dergi yığınının bir sayıyı bulmak için tarama yaparken istenen sayı aramaya başlanıncaya kadar tam aksi ucundadır. Benzer şekilde, kesmeyi üreten birim taramanın sonuncusu olacaktır.

**Vektörlü kesmeler** Vektörlü kesmelerde, kesme yapan aygıtın kendisi, ilgili kesme yönetim yordamını belirleyecek bazı olanaklar sağlar. Bu da tarama yapılıp yapılmadığını ortadan kaldırır. Tipik bir düzenlemede, çevreirim  $\overline{IRQ}$ 'yu alçak düzeye çekerek bir kesme üretir ve MİB de kesme alındı hattına bir sinyal çıkarır (mantıksal 1 veya mantıksal 0 darbesi) bunu bildirir. Çevreirim, alındı bildirimi gördüğünde, adres yolunun alt basamaklarındaki bitlerine bir adres yerleştirir. Daha sonra işlemci, kesme yapan aygıtın verdiği karşılığı okuyarak kesme kaynağını belirler. Bu yolla MİB, program sayıcısına her bir çevreirimine bir adres olmak üzere, sekiz ayrı kesme üretim yordamına olanak tanıyan sekiz farklı konumdan birinin içerdiği değeri yükleyebilir.

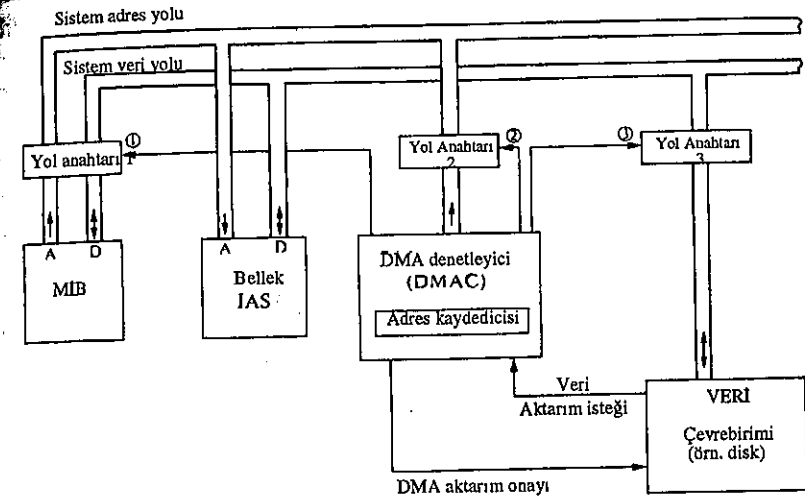
**Kesmelerin zarar verici olarak düşünülmesi** Yaşamdaki her şey, şu iki sınıfa birine girer: İyi ya da kötü. Kesmelerin kötü bir şey olduğuna kesin olarak inanmayan kişiler vardır. Arada bir kesme üreten tek bir çevreirim çok fazla baş ağrısı neden olmaz. Fakat çok sayıda çevreirim olan ve hepsi de asenkron olarak (yani rastgele) kesme üreten bir sistemi düşünelim. Sistemin bütünü, bu durumda rastgele kararlı bir şekilde davranamaz ancak stokastik (deterministik olmayan) hale gelir ve bu durum, en iyi rastgele işlemler matematiği ile açıklanabilir. Bu sistem ile ilgili bardaki büyük bir insan grubu arasında analogi yapılabilir - hiç servis yapılmaması gibi görünen biri daima vardır.

### 5.3 Doğrudan bellek erişimi

Doğrudan bellek erişimi (DMA) MİB'in doğrudan erişimli belleği ile bir çevreirim arasında MİB'in doğrudan müdahalesi olmadan verinin taşındığı bir tekniktir. Veri aktarımının bu biçimi, olası en hızlı biçimdir ve MİB'e fazladan iş çıkarmadığı sürelerde yararlı işler yapmak üzere MİB'i serbest bırakır. Yaşamla ilgili pek çok

şeyde olduğu gibi, eğer bir şey alınmaya değerse pahalıdır. DMA da bu kurala uyulması gerekir, çünkü gerçekleştirilmesi biraz karmaşıktır ve nispeten daha büyük miktarda bir donanıma gereksinim duyar. Şekil 5.7'de DMA'nın çalışması gösterilmiştir.

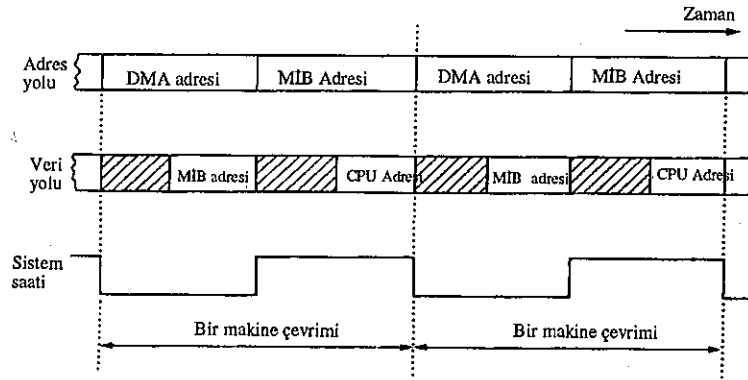
Bilgisayarın normal çalışması sırasında, yolun birinci anahtarı kapalı iken ikinci ve üçüncü anahtarları açıktır. MİB, adres yoluna bir adres çıkarır ve veri yolu aracılığıyla bellekten veri okurken veya belleğe veri yazarken yolları denetler. Bir DMA aktarımı yapıldığında yolun birinci anahtarı açık, ikinci ve üçüncü anahtarları kapalıdır. DMA denetleyicisi (DMAC) adres yoluna ve dolayısıyla da IAS'ye bir adres sağlar. DMAC, aynı zamanda o sırada doğrudan doğruya IAS'ye yazabilen veya ondan okuyabilen çevreirimine de DMA-onay (DMAG) sinyali gönderir. DMA işlemi tamamlandığında DMAC yol denetimini MİB'e iade eder.



Şekil 5.7 DMA ile giriş/çıkış

Gerçek bir DMA denetleyicisi oldukça karmaşık bir aygıttır. Birkaç iç kaydedicisi vardır - en azından yürürlükteki adresi tutmak için bir ve aktarılacak sözcükleri tutmak için bir kaydedici gereklidir. Birçok DMAC birkaç çevreirimine ilişkin işlemlerin üstesinden gelebilecek yapıdadır. Bu, kaydedicilerin kopya edilmesi anlamına gelir. Normal olarak DMA iki moddan birinde çalışır: Patlama modu ve çevrim çalma modu. Patlama modunda DMA denetleyicisi veri aktarımı (veya en azından çok sayıda bir sözcük aktarımı boyunca) boyunca sistem yoluna el koyar. Bu, verinin belleğe zincirdeki en zayıf halka örneğindeki gibi bellek-yol-çevreirim üçlüsünün izin verebileceği - en düşük hızda taşınmasını sağlar. Ne var ki, patlama modunda MİB fiilen durdurulur çünkü yolları kullanamaz.

Çevrim çalma modunda, DMA işlemleri bilgisayarın normal bellek erişimleriyle içiçe geçmiş haldedir. Bilgisayar, zamanının %100'ünde sistem yollarına erişme ihtiyacı duymadığı sürece DMA yolların serbest kaldığı sürelerde gerçekleşir. Birçok mikroişlemci sisteminde, bu 'serbest zaman'; MİB bir bellekten okuma veya belleğe yazma saykılı için adres üretmekle meşgulken oluşur. Şekil 5.8'da, çevrim-çalma gösterilmiştir.



Not: Tarama alanlar geçersiz veriyi gösterir.

Şekil 5.8 Çevrim-çalma ile DMA

Sistem saati alçak düzeydeyken MİB'in yol kullanmaya ihtiyacı yoktur. Böylece DMAC bunları kapar ve veri aktarımını gerçekleştirir. Saat yüksek düzeye çıktığında MİB, normal bellek erişim çevrimini gerçekleştirir. DMA'nın bu biçimine görünmezdir denir çünkü bilgisayar bunu görmez. Yani aktarım, bilgisayara görünmezdir ve kaybedilen herhangi bir işlem zamanı söz konusu değildir. Bir DMA işlemi, MİB'in başlangıç adresi ile aktarılabilecek sözcük sayısını DMAC'ın kaydedicilerine yazmasıyla başlatılır. DMA işlemi bittiğinde DMAC, MİB'e veri aktarımının bittiğini ve yeni bir aktarımın başlatılabileceğini ya da gerçekleşen bu aktarımın sonuçlarının kullanılabileceğini belirtmek üzere bir kesme gönderir. Küçük mikroişlemci sistemlerinin çoğunluğu DMA olanaklarını kullanmazlar.

#### 5.4 Kanal G/Ç

Bilgisayarlar gittikçe güçlendikçe ve bunlara artan sayıda G/Ç birimleri bağlandıkça DMA'nın bile G/Ç trafiğiyle başatmakta yetersiz kaldığı görülmüştür. İnsanlar kendilerine aşırı yüklenildikçe belli işlerin sorumluluklarını başka insanlara havale ederler. Bilgisayarlar da aynı şeyi yapmaya zorlanmıştır ve büyük miktarda G/Ç yönetimini başka bilgisayarlara vermiştir.

Kanal G/Ç anabilgisayar ile çevrebirimleri arasında küçük bir bilgisayar konmasının sonucu olarak karşımıza çıkmıştır. Bu küçük bilgisayara çoğu zaman ön-işlemci (FEP front-end processor) denir çünkü ana işlemciye girişi veya çıkışı yapılan veriyi işler. Anabilgisayarlar daha da güçlendikçe bunlara ilişkin FEP'lerin de güçlendiğine dikkat edilmesi gerekir. Hatta yalnızca FEP olarak kullanılan bilgisayardan söz ederek anabilgisayarının gücü hakkında böbürlenilen insanların varolduğunu duydum. Burada kastedilen şey şudur : FEP'in gücü buyusa anabilgisayarın neler yapabileceğini artık siz düşünün !

FEP başka adlarla da bilinir. IBM, FEP tarafından yönetilen veri akışını tanımlamak için kanallar terimini kullanır. CDC, FEP'in adını çevresel işlem birimi (PPU) olarak değiştirmiştir. IBM'in yaygın etkisinden dolayı, G/Ç kanalları kavramı artık bilgisayar endüstrisinin bir standardı olmuştur. FEP veya PPU, G/Ç sistemiyle aynı anlama gelirken, kanal sözcüğünün, sistemin sunduğu olanakları tanımladığına dikkat edin.

FEP, anabilgisayarın G/Ç işlemlerini yönetirken iki tür G/Ç'nin göz önünde bulundurulması gerekir. Anabilgisayar tarafından bakıldığında görülen G/Ç ile, FEP tarafından görülen G/Ç. FEP çevrebirimlerinin bizzat kendileriyle ilgilenir ve veriyi kendisi ile çevrebirimi arasında taşımak için daha önce açıklanan tekniklerden birini kullanır. Açıkçası FEP, hizmet verdiği her çevrebiriminin özelliklerini bilmelidir.

Anabilgisayar tarafından bakıldığında görülen G/Ç işlemleri oldukça basittir. Anabilgisayarın artık her bir çevrebiriminin ne olduklarını tam olarak bilmesine gerek yoktur. İşlemcinin yapması gereken tek şey, FEP'e tek bir buyruk, kanal denetim sözcüğü (CCW) göndermektir. Bu tip bir buyruk aşağıdaki bilgiyi içerebilir : \$002F00 konumundan başlayarak belleğime 256 baytlık veriyi oku. Veriyi B kanalı, 4 numaralı birimden al. Bu buyruk verildikten sonra FEP görevini sürdürürken anabilgisayar başka bir programla ilgilenir. FEP işini bitirdiğinde anabilgisayara bir kesme gönderir ve veri aktarımını başardığını (veya başaramadığını) belirten durum sözcüğü iletir.

Son birkaç yılda G/Ç'yi yönetim biçiminde yeni bir etkenin ortaya çıktığı görüldü. Buna sözde-akıllı çevrebirimi denir. Bir zamanlar çevrebirimleri, bilgisayar ile arabirimleri açısından farklılıklar gösterir ve her çevrebirimi, düzenli bir veri akışından emin olmak için belli bir sinyal dizisinin kendisi ile işlemci arasında değiş-tokuş edilmesine gereksinim duymuş. Bu işlem sırası, bir çevrebiriminden diğerine büyük farklılıklar göstermiştir. Bugün en ilkel çevrebirimleri dışındaki tüm çevrebirimleri mikroişlemcinin kendi içine yerleştirilmektedir. Bu, çevrebirimlerin

artık standart arabirimlerle satın alınabileceği sonucuyla mikroişlemcinin bazı arabirim karmaşıklıklarıyla ilgilenebileceği anlamına gelmektedir. Bu tip bir örnek IEEE-488 yoludur. IEEE-488 yolu ile uyumlu olarak satılan herhangi bir aygıt aynı uyumluluğu gösteren öteki tüm aygıtlara bağlanabilir. Bu, yeni aygıtların donanımında herhangi bir değişiklik yapmadan yazılımda da (eğer gerekiyorsa) çok az değişiklikle sistemdeki yerine oturtulabilmesine olanak tanır.

### 5.5 Giriş/Çıkış birimleri

Şimdiye kadar bilginin sayısal biçimde bilgisayar tarafından nasıl okunduğunu, program aracılığıyla yönlendirilip nasıl işlendiğini ve sonra da sayısal biçimde çıkışının nasıl yapıldığını inceledik. Bilginin gerçek dünyadaki biçimi ile sayısal biçimi arasında nasıl dönüştürüldüğünü henüz ele almadık.

Bu kısımda ilk olarak en sık karşılaşılan iki bilgisayar arabirimi açıklanmıştır. Bunlar VDU ile yazıcıdır. Mikroişlemci sistemleri artık çok çeşitli arabirimlerle ilişki içinde oldukça konuşma sentezleyici (synthesizer)'lerinden robotlara dek kısa bir bölüm de elektronik cihazların doğrudan denetim ve bilginin analog ve sayısal biçimleri arasındaki dönüştürülmesine ayrılmıştır.

#### 5.5.1 VDU

VDU artık mikrobilgisayar sistemlerinin büyük çoğunluğu için temel giriş/çıkış birimi olmuştur (mini ve anabilgisayarlar hariç). Bunun nedeni neredeyse tamamıyla elektronik bir cihaz oluşu ve ucuza üretilebilmesidir. Bunun yanı sıra bütün teletayplar ve yazıcılar elektromekanik cihazlardır ve yapısı gereği daha pahalıdır. VDU mantıksal olarak iki bölüme ayrılır : Bir giriş birimi (klavye) ve bir çıkış birimi (ekran). Buradaki giriş ve çıkış terimlerinin MİB'den görünen aygıtlara karşılık geldiğine dikkat edin. Yani klavye bir çıkış sağlar; bu da daha sonra MİB'in girişi olur.

#### Klavye

Klavye iki bölümden oluşur : Parmağın basıncını algılayan bir tuş takımı ve bir tuşun çıkışını o tuş gösteren tek bir ikili koda dönüştüren kodlayıcı.

Parmağın basıncını (tuşvuruşunu) algılayan tuşlar, normal bir mekanik aygıttır. Tipik bir tuşta, bir yayın kuvvetine karşı parmak aracılığıyla hareket ettirilen bir

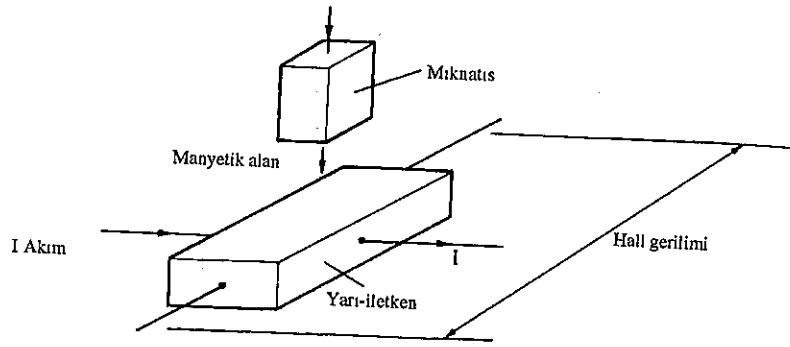
piston yer alır. Hareketinin sonunda piston, iki teli birleşmeye böylece devreyi tamamlamaya zorlar. Bu cihazın çıkışının, doğası gereği ikili (iki durumlu) olduğuna dikkat edin. Piston ile teller arasında, piston tarafından aşağıya doğru eğildiğinde duyulabilen bir klik sesi çıkaran küçük paslanmaz çelikten bir duyarlı-plaka bulunur. Benzer bir ses, piston bırakıldığında da çıkar. Bu şekilde, bir tuşa basma işinin, dokunmadan dolayı uyandırdığı olumlu bir duygu vardır. Aslında ucuz ve profesyonel bir klavye arasındaki farklardan biri de bu geribeslemenin varlığı veya yokluğudur.

Bir başka mekanik anahtar biçimi de bir ucuna ufak bir mıknatıs gömülü olan pistondur. Bu mıknatıs aşağı doğru itildikçe cam bir tüp (ince kamış röle) içindeki iki altın kaplamalı metal kontağa ulaşır. Bu kontaklar mıknatıs tarafından mıknatıslanır, birbirini çeker ve devreyi kapatırlar. Mekanik anahtarın üstün ergonomik özellikleri olduğu halde elektriksel özellikleri nispeten daha azdır. Özellikle kontaklar kirlenebilir ve arada temasa neden olurlar veya bir araya geldiklerinde geriye doğru titreşirler. Böylece tek ve temiz bir darbe yerine bir dizi darbe üretilmesine neden olurlar.

Anahtarın üç alternatifi vardır : Hall-etkili anahtar, elastometrik anahtar ve kapasitif anahtar. Hall etkili anahtar, bir Hall hücreğine doğrultulmuş durumdaki yayın kuvvetine karşı itilen bir mıknatıstan meydana gelir. Hall hücresi; üzerinden düzenli bir akımın geçtiği yarı iletken bir birimdir. Manyetik alan akıma dik gelecek biçimde uygulandığında hücrenin uçları arasında hem manyetik alana hem de akıma dik doğrultuda bir gerilim üretilir. Şekil 5.9'da bu tip bir anahtarın çalışması gösterilmiştir. Bu tip bir anahtar kontak titreşimine neden olmaz ama nispeten pahalıdır.

Kapasitif anahtarlar; bir parmakla iki metalik kontağa basıldığında bunlar arasındaki kapasitif kuplajdaki değişme esasına göre çalışırlar. Kapasitif anahtarlı bir klavyenin en büyük avantajı son derece düşük maliyetli ve küçük oluşudur, çoğu zaman baskılı bir devre kartından büyük değildir. Kontaklar basitçe yüzey üzerine asitle oyulmuştur. Ne yazık ki bu tür klavyede dokunmadan dolayı bir geribesleme yoktur ve kullanımı hoş değildir. Bazı tasarımcılar dokunma geribeslemesinin azlığının üstesinden işitilebilir bir geribesleme sağlayarak gelirler. Her seferinde bir tuş vuruşu yapıldığında hoparlörden kısa bir duyulabilir biip sesi üretilir. Kapasitif anahtar bazı çok ucuz kişisel bilgisayarlarda, televizyonlardaki dokunmatik tunerlerde, tehlikeli ortamlardaki çalışmalar için lehimle sıkıca kapatılmış olması gereken profesyonel donanımda bulunur.

Elastometrik tekniklere dayanan klavyeler, basınç söz konusu olduğunda elektrik-



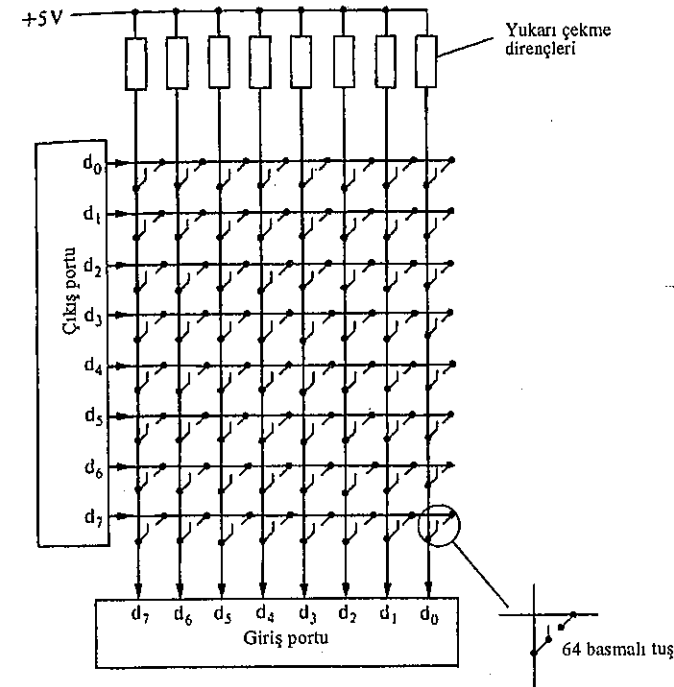
Şekil 5.9 : Hall-etkili anahtar

sel dirençlerini değiştiren belli türde malzemeleri kullanır. Malzeme üzerine parmakla basıldığında elektrik direncindeki düşme uygun bir arabirim tarafından algılanır. Yukarıda olduğu gibi, bu tür anahtar da dokunsal geribeslemeye eksiklik gösterir ve yumuşak ve kötü-tanımlanmış olarak adlandırılan bir duygu yarattığı söylenir.

Daktilodan dolayı pek çok elektronik klavyenin yerleşim düzeni QWERTY klavyesine yakındır - QWERTY bir kısaltma değil bir daktilonun en üst sırasındaki (soldan sağa doğru) ilk altı karakterlerin sıralanışıdır. Bazı klavyelerde, 0-9 arası rakamlar, ondalık nokta ve (gerial, satır-başı, satır atlatma gibi) bazı imleç karakterlerini içeren ayrı bir sayısal tuştakımı vardır.

Toplam tuş sayısını dolayısıyla da klavyenin boyut ve maliyetini azaltmak için birçok tuşun iki hatta üç işlevi vardır. Bu, iki özel tuş yardımıyla yapılır : Shift ve Control (Üstkarakter ve Kontrol). Bu tuşların amacı öteki tuşların anlamını değiştirmektir. Shift tuşu daktiloda buna karşılık gelen tuşla tamamiyle benzer işlevi görür : Küçük harfi, karşılık gelen büyük harfe dönüştürür veya iki alternatif sembolden birini seçer (örneğin : veya \*). Kontrol tuşunun yaptığı iş normal alfasayısal karakterlerin basılmayan ASCII denetim karakterleri üretmek üzere kullanılmasına olanak sağlamaktır. Şekil 3.1'deki ilk iki sütun, denetim karakterlerini göstermektedir. Denetim karakterleri, normalde, bir bilgisayar ile uzak bir cihaz arasındaki veri bağlantısının denetimi içindir.

Bir tuş takımının ASCII kodlanmış eşdeğerine dönüştürülmesi çoğu zaman klavye kodlayıcısı denilen özel amaçlı bir LSI yongası tarafından yerine getirilir. Bu tip bir yonganın temelinde yatan düşünce, Şekil 5.10'da gösterilmiştir. 8-bitlik bir çıkış portunun her bir çıkış hattına bağlı bir teli vardır. Bu teller Şekil 5.10'daki gibi

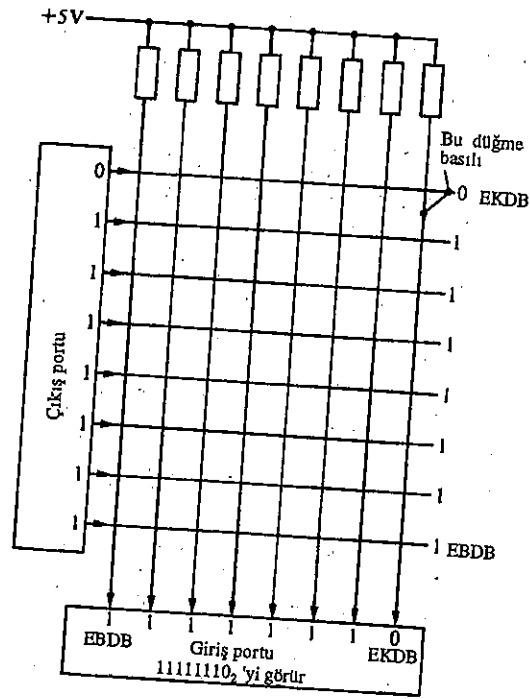


Şekil 5.10 Klavye kodlayıcısı

yatay olarak uzanmaktadır. Benzer biçimde 8-bitlik bir giriş portunun sekiz teli vardır ve bunlar Şekil 5.10'da dikey olarak çıkarlar. 8x8=64 kesişme noktasının herbiri, basıldığı zaman dikey çizgi ile ilgili yatay çizgi arasında bir bağlantı sağlayan tuş anahtarıdır. Hiçbir tuşa basılmadığı sürece herhangi bir dikey ya da yatay çizgi arasında bir bağlantı oluşmaz.

Giriş hatlarının her biri +5 Volta bağlı bir dirençle sonlandırılır. Böylece bu hatlar mantıksal 1'e çekilebilir. Yani, giriş portundan bir bayt okunacaksa bu 11111111 olacaktır. Şimdi de çıkış portunun sekiz çıkış hattına ikili sözcük olan 11111101'i yerleştirdiğini varsayalım. Bu durum Şekil 5.11'de gösterilmiştir. Eğer MİB giriş portundan okumayı, sözcüğü, en sağdaki tuşun basılması halinde yapmışsa 11111101'i görecektir. Eğer soldaki bir sonraki tuşa basılırsa 11111101'i görecektir. Açıkçası en üst satırdaki bir tuşa basmak bu tuşa karşılık gelen dikey konuma bir 0'ın okunmasına neden olacaktır. Öteki herhangi bir satırdaki tuşa basmanın okunan veri üzerinde etkisi yoktur.

Bundan sonra MİB, 11111101 baytının çıkışını yapar ve ikinci tuş satırını sorgu-



Şekil 5.11 Bir tuşa basılı iken klavyenin durumu

lamak üzere giriş hatlarını okur. Bu işlem, MIB'in 11111011'den 01111111'nin çıkışını yapana kadar, 0'ın her seferinde bir basamak sola kaydırılmasıyla periyodik olarak devam eder. Bu yolla sekiz satırın hepsi de tek tek sorgulanır. Bu iş daha çok Kısım 6.3'te açıklanan, ferrit göbekli belleklerin çalışmasıyla benzerlik gösterir. Tablo 5.4'te listelenen çevirici dilindeki program, klavyeyi çalıştırmak için gerekli olan yazılım hakkında bir fikir vermektedir.

### Ekran

Bir görüntüleme sisteminin tüm ayrıntıları bu kitabın kapsamı dışındadır. Öte yandan taramalı bir ekran çalışma şekliyle ilgili bir özet sunmak mümkündür.

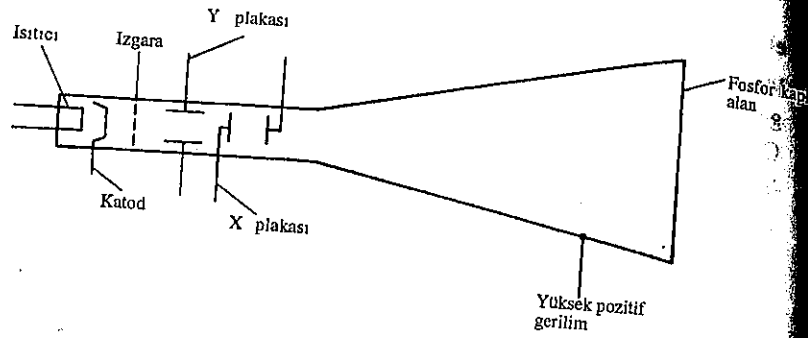
Pek çok görüntüleme sisteminin kalbinde iki moddan birinde çalışan katot-ışınlı tüp (CRT) yatar. Bu modlar : nokta-çizme ve raster-taramalıdır. Şekil 5.12'de nokta çizme ekranının görüntüleme kısmının blok diyagramı verilmiştir. CRT'nin katodu, sıcaklığı bir ısıtıcı eleman aracılığıyla 600 °C'ye yükseltilirse negatif yüklü

elektron yaymaya başlar. CRT'nin arka yüzü, katoda göre oldukça yüksek bir pozitif gerilime bağlanmış iletken bir maddeden yapılmış tabaka ile kaplıdır. Bu gerilim normalde 3000 ile 20000 V arasındadır. Bu potansiyel altında elektronlar CRT'nin ön yüzüne (yüzüne) bir kinetik enerji ile çarpmak üzere hızlandırılırlar. ön-yüzey yüksek enerjili elektronların bombardımanına uğradığında parlayacak biçimde fosforla kaplanmıştır. Elektronlar dar bir demet halinde odaklandıkça, ekranın önünde küçük parlak bir nokta yaratılır.

Tablo 5.4 Klavye kodlayıcının okunması

|                                                    |     |             |                                                  |
|----------------------------------------------------|-----|-------------|--------------------------------------------------|
|                                                    | ORG | \$0200      | bellekte programın başlangıcı                    |
| XHATLARI                                           | EQU | \$8000      | yatay hatlar için çıkış portu                    |
| YHATLARI                                           | EQU | \$8001      | dikey hatlar için giriş portu                    |
| XDEGERI                                            | RMB | 1           | geçici saklama konumu                            |
| XSAYISI                                            | RMB | 1           | o anki X değeri (çıkış hatları)                  |
| YSAYISI                                            | RMB | 1           | o anki Y değeri (giriş hatları)                  |
| BASLA                                              | CLR | XSAYISI     | yatay sayıcıyı sıfırla                           |
|                                                    | LDA | ## 11111110 | X hatlarını stroblamak için başlangıç değeri ver |
|                                                    | STA | XDEGERI     | başlangıç değerini sakla                         |
| DONGU                                              | LDA | XDEGERI     | X çıkışının o anki değerini al                   |
|                                                    | STA | XHATLAR     | bunu yatay hatlara çık                           |
|                                                    | LDA | YHATLAR     | Y girişini al                                    |
|                                                    | CMP | ## 11111111 | bu satırda basılı tuş var mı bak                 |
|                                                    | BNE | DONGU1      | eğer varsa kodunu çöz                            |
|                                                    | INC | XSAYISI     | X döngü sayıcısını artır                         |
|                                                    | SEC |             | elde bitini kur (birlerde kaydırmamız gerek)     |
|                                                    | ROL | XDEGERI     | bellekte kalan X değerini döndür                 |
|                                                    | BCS | DONGU       | elde 1 varsa son çevrim değildi                  |
|                                                    | JMP | BASLA       | son döngüyse baştan başla                        |
| * şimdi karakteri 6-bitlik bir değere kodlamalıyız |     |             |                                                  |
| DONGU1                                             | CLR | YSAYISI     | YHATLARI 3-bitlik bir değere sıkıştırılmalı      |
| DONGU2                                             | CMP | ## 11111110 | değeri sıfır mı ?                                |
|                                                    | BEQ | DONGU3      | öyleyse başka işlem yok                          |
|                                                    | SEC |             | eldeyi kur                                       |
|                                                    | ROR | A           | Y değerini sağa kaydır                           |
|                                                    | INC | YSAYISI     | Y döngü sayıcısını artır                         |
|                                                    | JMP | DONGU2      | tekrarla                                         |
| DONGU3                                             | CLC |             | eldeyi temizle                                   |
|                                                    | LDA | XSAYISI     | X çevrim sayacını al                             |
|                                                    | ASL | A           | üç basamak sola kaydır                           |
|                                                    | ASL | A           | ikinci kaydırma                                  |
|                                                    | ASL | A           | üçüncü kaydırma                                  |
|                                                    | ORA | YSAYISI     | bunu Y döngü sayıcısı ile birleştir              |
|                                                    | RTS |             | akümülatördeki tuş değeriyle dön                 |

Katodun önünde ızgara denen ve negatif olarak yüklendiğinde katoddan gelen elektronları geri çeviren ve bu yolla da noktanın yoğunluğunu azaltan ince telden yapılmış bir ağı vardır. Işımlar, biri X düzleminde biri de Y düzleminde olmak üzere



Şekil 5.12 CRT (Katot Işınlı Tüp)

iki çift levhanın arasından geçer. Bu levhalara elektrostatik yüklemeye yapmak suretiyle ışınların tübün yüzü üzerinde herhangi bir noktaya saptırmak mümkündür. Modern VDU'lar ve televizyonlar biraz farklı bir ilke ile çalışır. Bunlar ışığı CRT'nin dışında birbirleriyle dik açı ile yönlendirilmiş bir sargı çiftinden akam geçirerek manyetik olarak saptırır. Bir elektron demetini geniş bir açıyla manyetik olarak saptırmak elektrostatik olarak saptırmaktan daha kolaydır.

İki tane ikili giriş, ikili sayıyı, sayısal değeriyle orantılı bir gerilime dönüştüren iki n-bitlik sayısal-analog dönüştürücüye uygulanır. Analog gerilimler yükselteçlerle X ve Y saptırma levhalarına (veya saptırma sargılarına) uygulanır. Böylece ikili girişler noktanın X ve Y koordinatlarını ekran üzerinde doğrudan belirler. Ek bir giriş ışını açıp kapatmak (boşluk) için ızgaraya uygulanır. Bu tür bir ekranı çalıştırmak için bir X-Y değerler tablosu bilgisayarın IAS'sinde korunur ve değer çiftleri sırasıyla iki çıkış portunu besler. Portlar, ışını saptırmak için gereken gerilimi sağlayan iki sayısal-analog dönüştürücüye bağlanır. Noktalar, görüntü kısa bir süre sonra yazılırlarken, kırışmayı önlemek üzere hızla çizilir. Tüm noktaları saniyede yaklaşık 50 kere görüntülemek gerekir.

Alternatif bir görüntüleme raster-taramalı modda çalışır (raster; ışının ekranın yüzeyi üzerindeki yoldur). Burada elektron ışını ekranın yüzeyinin tamamını kapsamak üzere periyodik olarak ekranın bir yanından diğerine, satır satır taranır. Bu modda CRT denetleyicisinin, çizmek istediği noktanın X ve Y koordinatlarını açık açık belirtmesi gerekmez. Bunun yerine, CRT denetleyicisi, ışın uygun X ve Y koordinatlarındayken, bir nokta üretmek için ekranı aydınlatır. Raster taramalı ekranın televizyon görüntüsünün temelini oluşturan çalışması ayrıntılı olarak aşağıdaki gibidir.

Doğrusal biçimde artan testere dişi gerilim CRT'nin X saptırma katına uygulanır. (Şekil 5.12'ye bakınız). Bu, elektron ışının ekran boyunca bir yatay hattı taramasına

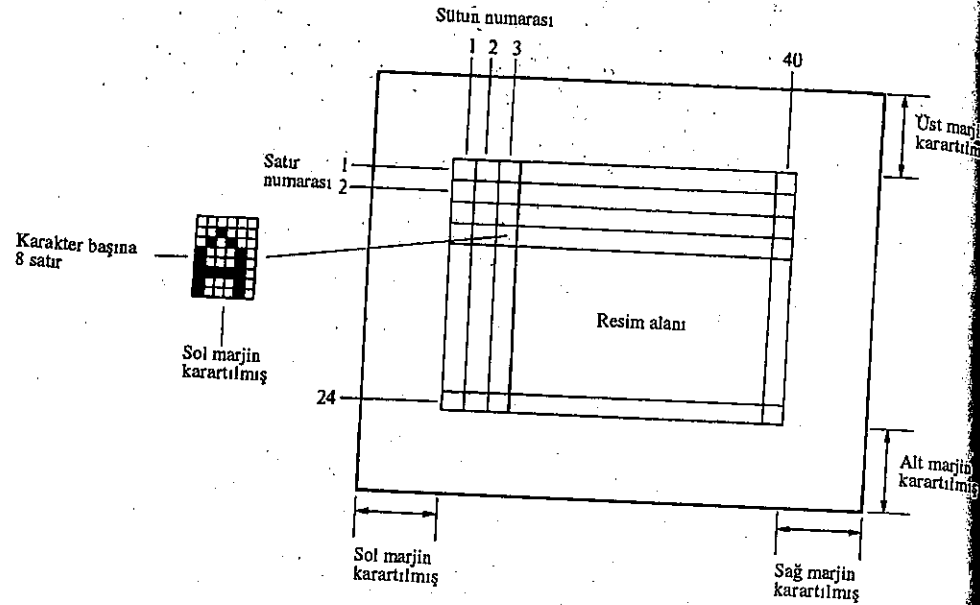
neden olur. Televizyonda ışının yoğunluğu, resmi oluşturmak için alınan sinyal tarafından modüle edilir. VDU'da ışın, bir noktalar dizisi üretmek üzere tamamiyle açılır veya tamamiyle kapatılır. Işın ekranın sağ kenarına eriştiğinde kapatılır ve bir sonraki taramaya hazır şekilde yatay başlama konumuna döndürülür (geri uçuş). Işın yatay düzlemde tarama yaparken, çok daha düşük bir hızla düşey düzlemde de taranır. Bu şekilde her bir yatay tarama (bir satır) bir önceki taramanın altına düşer. İngiltere'de saniyede 50 düşey tarama vardır ve her bir düşey tarama (buna bir alan denir)  $312\frac{1}{2}$  satırdan oluşur. Bir çerçeve biri  $312\frac{1}{2}$  tek sayılı biri de  $312\frac{1}{2}$  çift sayılı satır içeren iki ardışık alandan meydana gelir. Çerçevedeki toplam satır sayısı  $2 \times 312\frac{1}{2} = 625$ 'tir.

Tek sayılı ve çift sayılı satırları ardışık alanda görüntülemenin nedeni 615 satırın tümünün, saniyenin beşte birinde taranmasının güçlüğüdür. Bu tip bir düzenlemeye geçişli yapma denir. Geçişli yapma bazı VDU'larda 625 satırlık düşey çözünürlük vermek için kullanılır ama birçok VDU, geçişli yapma tekniğini kullanmaz ve  $312\frac{1}{2}$ 'lik satır çiftleriyle bırakılır. Hemen aşağıda, görüntülenen satır sayısının bir ekrana uyacak karakterin satır sayısını etkilediğini göreceğiz.

VDU ekranı, görüntüleme biçimini tanımlamak üzere satır ve sütun matrisine bölünmüştür. Tipik VDU formatları 80x24, 40x24, 64x16 ve 32x16'dır. Her bir sayı çiftinin ilk değeri satırdaki sütun sayısını ikinci değeri de çerçevedeki satır sayısını verir. Şekil 5.13'de, tipik bir ekranın formatı verilmiştir. Her karakter 6 noktaya 8 satırlık bir blok içinde font denilen 5x7'lik dot (nokta) matris olarak görüntülenir.

Şekil 5.14'te basit bir 40 sütuna 24 satırlık görüntü üreticinin blok diyagramı verilmiştir. Bu devrenin çıkışı "karma görüntü" (kompozit video) sinyalidir ve normal bir televizyonun video (görüntü) katını besler. Bir başka deyişle, bu blok diyagram, bir televizyonun sayısal bir veriyi görüntüleyebilmesi için gereken ek devreyi temsil eder. Devrenin çalışmasının temeli 5.76 MHz hızında darbeler sağlayan saat üretici. Bu frekansa nokta-saati denir ve ardışık noktaların görüntülenme hızıdır. Nokta-saati altıya bölünerek 0.960 Mhz veya 960 kHz.'de karakter saati üretilir. Bölen altıdır çünkü, her karakterde beş nokta artı bitişik karakterler arasında boş bir nokta boşluğu vardır.

Karakter saati (yani karakter başına bir darbe) her satır için 60 karakter konumu, veya zaman aralığı elde etmek üzere 60'a bölünür. Sadece 40 karakter görüntülenecektir. 20 zaman yarığı ekranın sol ve sağ marjlarını oluştururlar ve geri-uçuş için zaman sağlarlar. Karakter saatini 60'a böldükten sonra, 960 kHz/60'lık (yani

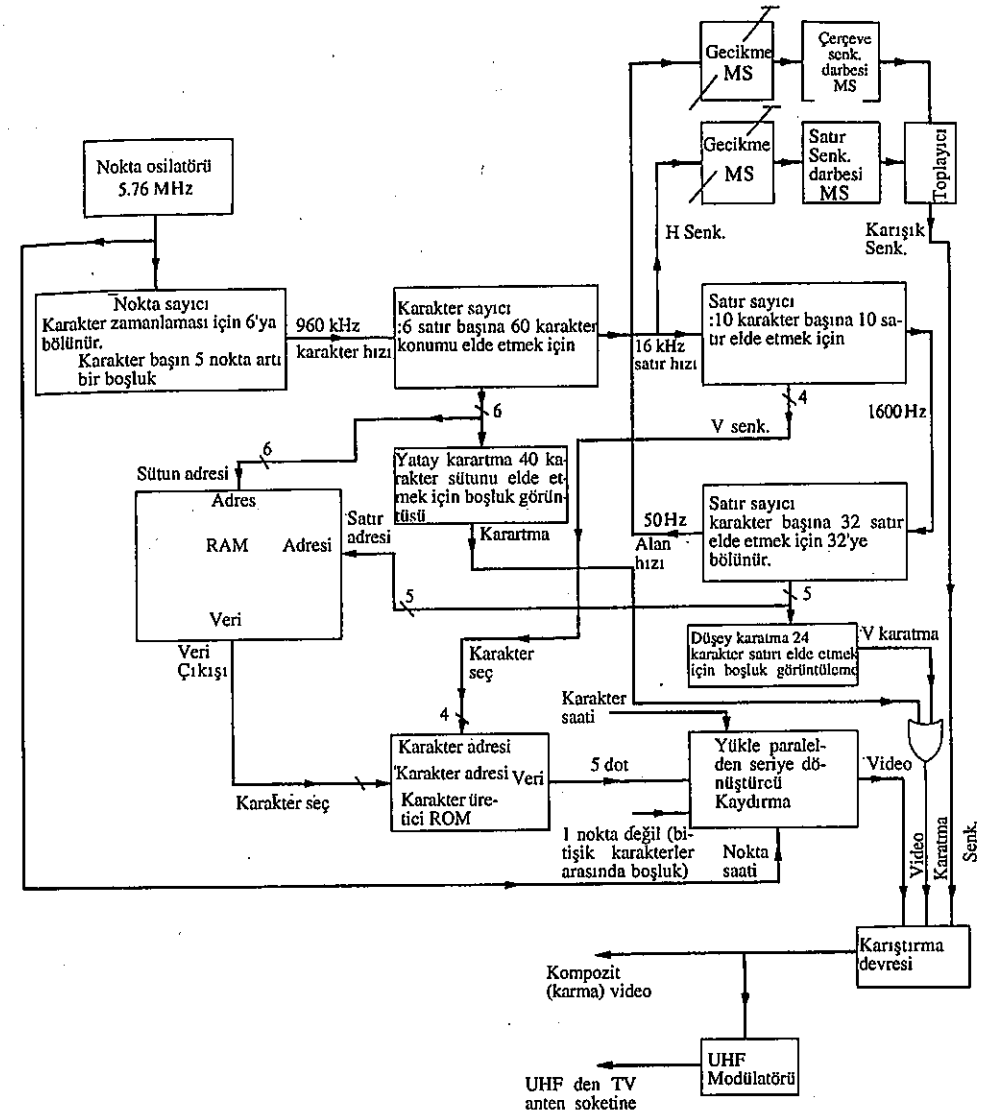


**Şekil 5.13 Raster taramalı bir ekranın formatı**

16 kHz) bir frekans elde edilir. Bu satır hızıdır ve satır senk. denilen bir satır senkronlama sinyali üretmek için de kullanılır. İşte bu senkronlama sinyali televizyona yeni bir satıra ne zaman başlanacağını söyler. Satır saati her bir karakter satırı için 10 dikey satır vermek üzere 10'a bölünür. Karakteri oluşturmak için yedi satır gerekir ve üç satır da komşu karakter satırları arasındaki boşluğu oluşturur.

Satır sayıcısının çıkışı  $16 \text{ kHz}/10 = 1.6 \text{ kHz}$ , alan başına 32 satır aralığı vermek üzere 32'ye bölünür. Bu 32 aralık içinden sadece 24 tanesi karakter görüntüler, kalan sekiz tanesi, üst ve alt marjları meydana getirir. Satır sayıcısının çıkışı  $1.6 \text{ kHz}/32 = 50 \text{ Hz}$ 'dir ki bu İngiltere'deki alan hızıdır. Alan hızı, ayrıca televizyonu görüntü üreticisiyle aynı derecede görüntüleme yapmak için alan senk. darbeleri üretmek üzere kullanılır.

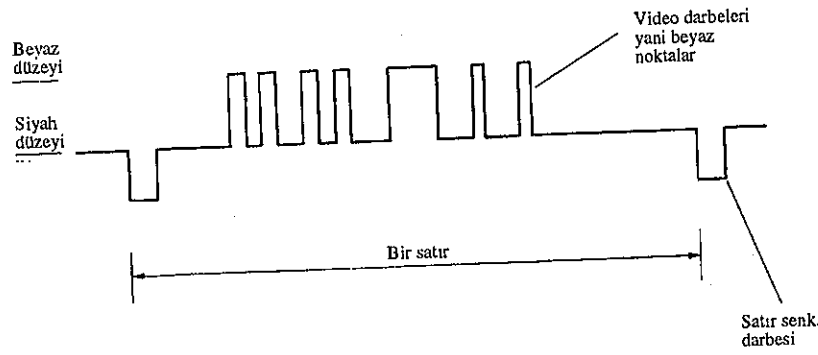
Bölenleri ya da zamanlama zincirini kurduktan sonra, görüntünün kendisini üretmek oldukça kolay bir iştir. Karakter ve satır sayıcıları sıralı olarak  $40 \times 24 = 960$  karakter konumunun tümünü derecelendirdiğinde bu sayıcıların ikili çıkışları, her bir karakter konumu için ASCII kodlanmış bir sözcük üretmek üzere rastgele-erişimli bir bellek blokunu sorgular. Yani satır ve sütun sayımları belleğe adres girişleri olarak hareket eder ve bellekten gelen veri görüntülenecek karakterin kodunu gösterimler. Bu bellek bloku VDU devresinin veya MİB'in kendi IAS'sinin bir



**Şekil 5.14 Ekran üretici**

bölümü olabilir. İkinci durum birçok küçük kişisel bilgisayar için geçerlidir (Apple, Pet, ZX81). Rastgele-erişimli belleğin çıkışı, karakter kodunu, bu karakteri oluşturan nokta desenlerine dönüştüren ve karakter üreticisi denilen özel-amaçlı sadece okunur belleğin (ROM) adres girişlerini besler. ROM'un o sırada görüntülenmekte olan karakterin yedi satırından birini seçen ek bir adres girişi vardır. Bu adres 10 ile bölünen satır sayıcısından alınır.

ROM'un çıkışı karakterin yedi satırından birini oluşturan 5-bit bir sözcükten oluşur. Bitler her karanlık bir nokta için 0, görüntülenen bir nokta için 1'dir. Bu bitler altı bitlik kaydırmalı kaydediciyi besler (bir bit karakterler arasında boşluk vermek için daimi olarak 0'a kurulu). Kaydırmalı kaydedici, karakter saati tarafından her bir yeni karakterin başlangıcında yüklenir ve noktalar nokta saati tarafından kaydırılır. Bu noktalar daha sonra televizyon alıcısını besleyen sinyalleri üreten devreleri besler. Devrenin öteki bölümleri; video çıkışını karakterler görüntülenmediği sırada sıfırlayan ekran karartma ile video sinyalini senkronizasyon (sync.) darbeleriyle birleştiren karıştırma devreleridir. Çıkan dalga desenine karma video sinyali denir ve Şekil 5.15'te gösterilmiştir.



Şekil 5.15 Karma (kompozit) video sinyali

Yukarıdaki görüntü üreticisi bir VDU'ya dönüştürülmek istenirse, karakterleri RAM'a yazmak için, bazen video RAM veya görüntü belleği denen, bazı ek devrelere gereksinim duyulur. Sayıcılara donanım olsun yazılım olsun bir sonraki kaydın nereye gideceğinin hesabını tutmak için gerek vardır. Denetim karakterleri ihtiyacını sağlamak için özel düzenlemeler gerekir. Örneğin, bir Satır-başı karakteri, VDU tarafından alındığında VDU bunu görüntülemeyebilir ama sütun konum sayacını sıfırlamak için kullanır. Benzer şekilde yeni bir satır karakteri basitçe satır konum sayıcısını bir artırır.

Hemen hemen tüm VDU'larda bir imleç bulunur. İmleç, bir sonraki karakterin gideceği konumu belirtmek için ekranda görüntülenen özel bir semboldür (bazen yanıp söner haldedir). Bir imleç üretmek için görüntü üreticisinin satır ve sütun sayıcılarının içerdiği değerler, ekranda yer alan bir sonraki serbest konumun işaretçileri ile karşılaştırılır. Bu iki değer birbirine eşit ise CRT'nin ışını bir sonraki serbest karakterin konumunda olmalıdır ve bu yüzden bir imleç üretmek üzere açılmalıdır.

Yukarıdaki VDU açıklamaları, zorunlu olarak, birçok ince ayrıntı eklenecektir.

olduğu sürece basittir. Öte yandan pek çok VDU bu ilkeler üzerinde çalışır. Bugün bir görüntü üretici devrenin çoğu, birkaç sterlinlik tek bir entegre devrede mevcuttur. Genel olarak, VDU'nun maliyeti, içinde kullanılan bileşenler azaldıkça düşmektedir.

### 5.5.2 Yazıcı

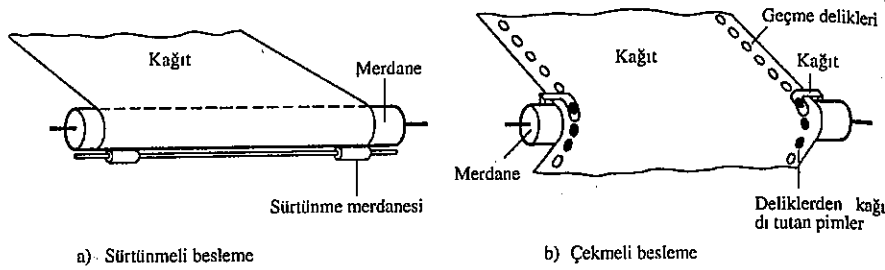
Yazıcı; sayısal bilgiyi kağıt üzerindeki işaretlere dönüştürerek bilgisayardan kalıcı bir 'basılı kopya' çıkışı üretir. Yazıcılar hassas biçimde üretilmiş hareketli mekanik parçalar temeline dayandığından, tümüyle elektronik olarak üretilmiş VDU'lardan daha pahalıdır ve çoğu kez daha az güvenilirlerdir. Bundan başka yazıcı fiyatlarının değişim aralığı, VDU'nunkinden çok daha geniştir. ZX81 ile kullanılan ucuz maliyetli cinsten bir yazıcının fiyatı 50 sterlinken, kapsamlı bir satır yazıcı 10 000 sterlinden daha da pahalı olabilir.

Yazıcı türleri ve mekanizmaları, her biri maliyet, hız, güvenilirlik ve basma kalitesi açısından avantaj ve dezavantajlara sahip çok geniş bir çeşitlilik gösterir. Öte yandan tüm yazıcılar aynı temel işlevleri yerine getirmelidir. Bunlar :

1. Kağıdı, verilen satıra taşıma
2. Baskı kafasını, satır üzerinde istenen bir noktaya taşıma
3. Basılacak karakteri seçme
4. Kağıt üzerinde bu karaktere karşılık gelen işareti basma

İlk ve son noktaların açıklanması nispeten daha kolaydır ve bu nedenle de başta bunlar ele alınacaktır. Uygulamaya bağlı olarak kağıt, tek bir yaprak, sürekli form veya üstüste katlanmış şekilde olabilir. Yazıcıların besleme mekanizması daktilolardan fazla bir farklılık göstermez. Kağıt; bir motor tarafından sürülen bir merdane ile kağıda basınç uygulayan baskı merdaneleri arasında tutulur ve sürtünmeli besleme ile hareket ettirilir. Merdane (veya silindir) hareket ettikçe kağıt onunla beraber sürüklenir. Şekil 5.16(a)'da sürtünmeli besleme sürücünün çalışması gösterilmiştir. Bu tip bir sürücü kusursuz değildir ve kağıdın kaymasına neden olabilir veya en azından kağıt tam olarak hizalanmayabilir. Alternatif bir kağıt besleme mekanizması çekmeli (veya dişli) beslemedir (Şekil 5.16(b)). Burada, merdanelerin uçlarında, üzerinde konik çıkıntılar bulunan halka bulunur. Kağıdın (zig-zag katlanmış kağıt) kenarlarında delikler bulunur, silindir döndükçe kağıt çekilerek yazıcıya doğru alınır.

Kağıt üzerinde bir işaret yapmak dolaylı olarak karakterin oluşma biçimine bağlıdır, ancak bir grup temel ilke belirlenebilir. Kağıdı işaretlemenin en yaygın yöntem-



Şekil 5.16 Yazıcıya kağıt takma mekanizmaları

mi, mürekkep emdirilmiş sert bir cismin vurması temeline dayanır ki bu da kağıdın üzerinde cismin şeklinde bir işaretin basılmasına neden olur. Pekçok büro tipi daktilonun çalışması da bu şekildedir. Yazıcılar, çoğu zaman işaretin kağıt üzerinde basılma yöntemiyle anılırlar. Vurma yoluyla basmayı kullanan yaygın düzenlemeler şunlardır: Golf topu, silindir, papatya çarkı, satır yazıcı ve tambur yazıcı.

Bir başka teknik, yaklaşık 110 C'ye kadar ısıtıldığında rengi siyaha (veya maviye) dönen bir madde emdirilmiş özel bir kağıda baskı yapar. Bu tür yazıcılara termik yazıcı denir. Tipik olarak (VDU'da olduğu gibi) 7'ye 5'lik bir matristeki noktaların belli bir kombinasyonu, ısıtılmak suretiyle kağıda bir karakter olarak basılır. Termal yazıcılar çoğu zaman (özel kağıdın fiyatı işin içine girene kadar) oldukça ucuzdur ve çalışması nispeten sessizdir. Bir başka mekanizma siyah renkli, yüzeyi ince parlak alüminyumdan bir filmle kaplı bir kağıt kullanır. İğne biçimindeki elektrot bir yüzeye uygulandığında, yüzeyden büyük bir akım geçer ve alüminyum film, altındaki koyu tabakayı ortaya çıkaracak biçimde kısmen buharlaşır.

Daha yeni basma yöntemlerinden biri de kağıda ince bir mürekkep püskürtme tekniğini kullanır. Bu teknik aynı zamanda karakterin seçilme ve oluşturulma biçimini de içerdiğinden daha sonra ayrıntılı olarak ele alınacaktır.

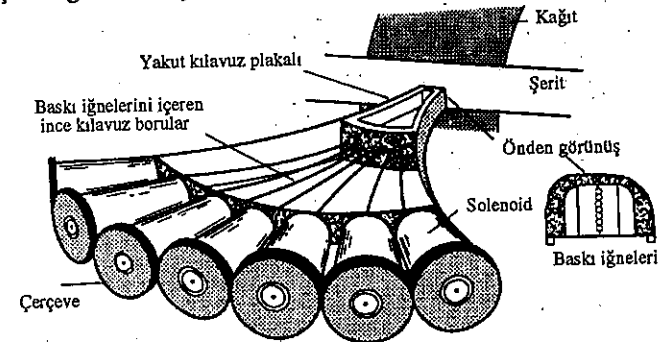
Pratikte karakterleri basan mekanizmaya baskı-kafası denir. Satır yazıcılarda iki grup baskı kafası vardır : tekli baskı kafası, çoklu baskı kafası. Daktilolarda baskı kafası sabittir ve herbir karakter basılırken kağıt ve merdane hareket eder. Merdane'nin ve kağıdın kütlesi ve dolayısıyla da ataleti büyük olduğundan bu tip yazıcılar yüksek hızda basıma uygun değildir. Baskı kafasının kütlesi silindirinkinden daha küçük olduğundan, pek çok yazıcı, kağıt bulunduğu yerde kalacak ve baskı kafası satır boyunca hareket edecek biçimde düzenlenmiştir. Baskı kafasını hareket

ettirmenin bir yolu da kafayı vida dişli bir çubuk (ana mil) üzerindeki bir halkaya oturtmaktır. Milin sonunda, mili belli bir zamanda sabit bir açı ile döndüren adım (stepping) motoru vardır. Kolun her dönüşünde baskı kafası (dönmenin yönüne bağlı olarak) sola veya sağa doğru hareket eder. Bir başka düzenlemede baskı kafası kağıdı hareket ettiren mekanizmaya benzer bir mekanizma ile bir kayışa bağlanır. Kayış; biri serbest olarak hareket eden diğeri ise adım motoru tarafından denetlenen iki teker arasında giderebilir.

Karakterin oluşma biçimi yazıcının türünü belirlediğinden bir grup yazıcı türü aşağıda açıklanmıştır.

### Dot-Matris (Nokta-Basırlı) Yazıcı

Dot-matris yazıcılar, karakterleri VDU'nun kendi karakterlerini oluşturma biçimine çok benzeyen bir biçimde noktalardan oluşan matrisler aracılığıyla oluştururlar. Noktalar, (iğne denen) bir grup tel aracılığıyla mürekkepli şeridi kağıt yüzeyine iterek oluşturulur veya iğneler kıvılcım erozyonu tekniğiyle kullanılabilir ya da termik bir yazıcıda ısıtıcı elemanlarıyla değiştirilebilir. Dot-matris yazıcılarda nispeten birkaç hareketli parça vardır ve fiyatları ortalama düzeydedir. Kişisel bilgisayar sistemlerinde bulunan pek çok yazıcı dot matris tekniğini kullanır ve 100-600 sterlin arasında bir fiyattadır. Şekil 5.17'de, dot-matris baskı kafasının çalışması gösterilmiştir.



Şekil 5.17 Dot-matris baskı kafası

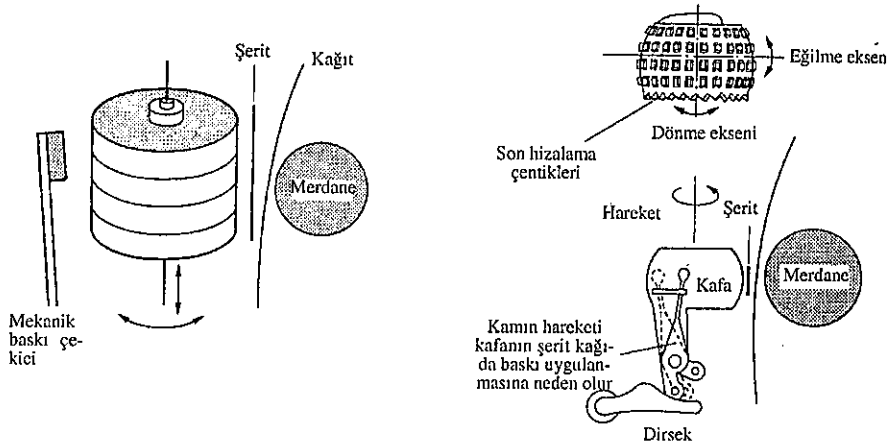
Yedi ayrı solenoid, yedi telin her birini denetler. Solenoidler, içinden bir geçtiğinde güçlü bir manyetik alan yaratan sargılardır. Teller demirden yapıldığından, içlerinden bir akım geçirilerek solenoidin içine veya dışına doğru hareket ettirilebilirler.

Buradaki yedi tel, yedi noktalık bir sütun oluşturmak üzere yakuttan bir kılavuz ile biraraya getirilir. Solenoidlerin belli bir kombinasyonu enerjilendirildiğinde, iğneleri, bu grubu kağıt üzerinde noktalar deseni halinde basmak üzere şeride doğru iterler. Noktalardan oluşan beş sütun basıldığında, bir karakter tamamlanmış olur.

Karakter küçük noktalardan oluştuğundan, dot-matris yazıcıları, herhangi bir karakter takımı için programlamak mümkündür. Bazı yazıcılar da noktalardan oluşan grafiklerinin basılabilesine olanak tanır. Ne yazık ki, bir dot-matris yazıcının basım kalitesi, görüntünün herşey olduğu iş uygulamalarında kabul edilemeyecek kadar düşüktür. Bazı dot-matris yazıcılar, kesiksizlik izlenimi uyandırabilmek için noktaları biraz üst üste basarak ortalama kalitede çıktı üretirler.

#### Silindir ve golf topu yazıcılar

Şekil 5.18'de silindir baskı kafasının ve golf topunun çalışması görülmektedir. Bu iki baskı kafasının çalışması birbirine çok benzer. Silindir basma kafası teletayplarda bulunur ve üzerinde herbirinde onaltı kabartma karakter bulunan dört satırlık metal bir silindirler. Şerit ve kağıt, silindirin hemen önünde ve çekiç de arkasına yerleştirilmiştir. Silindire dik bir eksen etrafında döner ve istenen sembol şeridin karşısına gelinceye kadar aşağı yukarı hareket ettirilir. Bir solenoid tarafından sürülen çekiç, merdanenin arkasından vurarak sembolü şeritten kağıdın üstüne doğru iter.



Şekil 5.18 Silindir ve golf-topu baskı kafaları

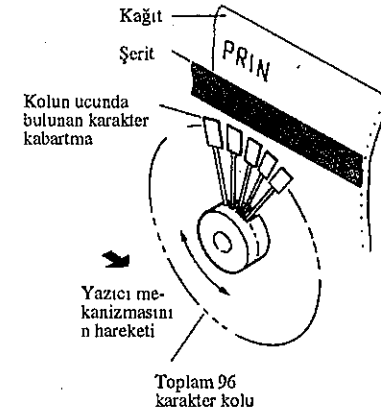
Golf topu kafası ilk olarak IBM'in elektrikli daktilolarında kullanılmıştır, fakat artık daha geniş yazıcı grubunda görülmektedir. Karakterler metalik bir kürenin

yüzeyine kabartılmıştır. Golf topu, silindirle benzer biçimde döner fakat farklı karakter satırlarına erişmek için aşağı yukarı hareket etmek yerine eğilir. Silindirden farklı olarak, golf topu arkasından vuran bir çekiçle değil de bir kam mekanizmasıyla şeride ve kağıda doğru sürülür.

Silindirle golf topu arasında kavramsal bir fark bulunmamasına rağmen halen kullanımda olan silindir mekanizması yalnızca ASCII kodun küçük harf 64 karakterlik bir altkümelerini sağlar. Golf topu üzerinde hem büyük hem de küçük harfleri bulunur ve bu tip baskı kafası değiştirilebilir özelliktedir. Bu, golf toplarını değiştirerek farklı yazı tiplerinin elde edilebileceği anlamına gelir. Golf topu yazıcının istenen kalitede basım yapabildiği söylenir.

#### Papatya çarklı yazıcı

Adını aldığı papatyada olduğu gibi, papatya çarklı yazıcının, çevresi etrafında düzenlenmiş çok sayıda incecik taç yaprakları olan bir diski vardır. Bu taç yapraklarının veya kolların her birinin ucunda kabartma bir karakter bulunur. Çark plastik veya metalden yapılır ve ataleti çok düşük değerde olacak şekilde hafif malzemeye yapılmıştır. Tipik bir papatya çarkında ASCII kodun küçük ve büyük harf altkümelerine karşılık gelen 96 kol vardır. Şekil 5.19'da papatya çarkı görülmektedir.



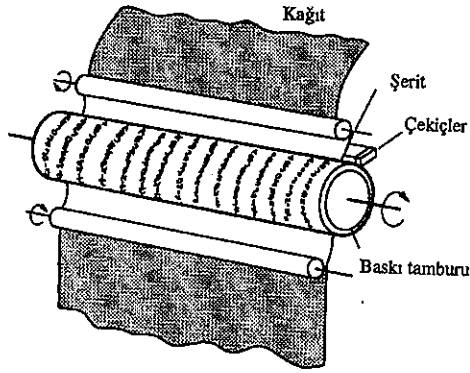
Şekil 5.19 Papatya-çarklı yazıcı

Papatya çarkı, şeridin önünde, düşey düzlemde ve hızla döner. Çark döndükçe karakterlerin her biri solenoid sürümlü bir çekiç ve şeridin arasından geçer. İstenen karakter basma konumundayken solenoide akım uygulanır ve çekiç kağıdı basmak üzere kolu şeride doğru iter. Burada dikkat edilmesi gereken şey; bir papatya çarkı yazıcısı tarafından ihtiyaç duyulan karmaşık zamanlamayı gerçekleştirebilmek için önemli oranda (mikroişlemci tarafından denetlenen) bir elektroniğe gerek olması-

dır. Hatta bazı yazıcılar solenoid'deki akım miktarını (dolayısıyla da çekim gücünü) karakterin büyüklüğüne göre denetleyebilir. Bu her karaktere eşit güçlük verir. Bazı vurmalı yazıcılar nokta karakterini öyle güçlü basarlar ki kâğıdın üzerinde küçük bir delik açılır! Papatya çarklı yazıcılar yüksek kalitede basımlardan dolayı çoğu zaman kelime işlemcilerle birlikte kullanılır.

#### Satır yazıcı

Satır yazıcısına bu adın verilmesinin nedeni karakterleri sırayla basmaktır. Yazının tüm bir satırını bir kerede basmasıdır. Satır yazıcılar pahalıdır, çoğu zaman düşük kalitede çıktı üretir ve büyük hacimli, hızlı basım almak için üretilmişlerdir. Satır yazıcıların yaygın biçimlerinden biri, tambur yazıcı, Şekil 5.20'de gösterilmiştir.



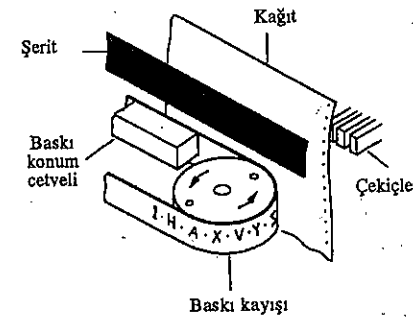
Şekil 5.20 Tambur yazıcı

Şeridin önünde kâğıdın eni boyunca uzanan bir tambur bulunur. Basılacak karakter takımının kabartma desenleri, tamburun çevresi boyunca sıralanmıştır. Bu karakter takımı tambur boyunca her bir karakter konumu için tekrarlanmıştır. Tipik bir satır yazıcıda 132 karakter konumu ve 64 karakterlik bir karakter takımı vardır. Bunun sonucu olarak tamburda,  $132 \times 64 = 9448$  adet kabartma karakter bulunur. Tambur döndükçe karakterlerin oluşturduğu halkalar 132 basma konumunun her birinden geçer ve her dönüşte karakter takımının tümü her basma noktasının önünden bir kez geçmiş olur. Öteki diğer yazıcılardan farklı olarak, çekicinin kâğıda vurması ve şerit boyunca kafaya doğru itmesiyle karakterler kâğıda basılırlar. Çekicinin enerjileneceği nokta zamanlanarak istenen karakterler basılabilir. Her karakter konumu için bir çekiç bulunduğundan bir satırın tamamı tamburun tek bir turu boyunca basılabilir.

Sözcüğü basılacak satırda tek bir sözcüğün, örneğin ALAN sözcüğünün olduğunu varsayalım. Her karakter halkasındaki ilk karakter A, ardından B ve diğerleridir. Kâğıdın bulunduğu satıra çekici mekanizmayla getirildiğini düşünelim. Tambur döndükçe bir zamanlama sinyali üretilir ve çekicileri denetleyen elektronik devreye gönderilir. Birinci karakter (yani A) tarafından bir senkron darbesi üretildiğinde bir ve üçüncü çekiciler enerjilendirilir ve A A deseni basılır. Tambur yirminci konumunu da geçtikten sonra, ikinci çekiç enerjilendirilir ve satırda artık ALA kelimesi vardır. Tamburun bir sonraki iki adımında, satırı bitirmek ve ALAN sözcüğünü basmak üzere dördüncü çekiç enerjilendirilir. Tambur tam bir turunu bitirdikten sonra kâğıt bir satır kaydırılır ve dizi tekrarlanır.

Tambur yazıcılar, dakikada 100-400 satır hızında çalışırlar. Her bir kısa süre içinde kâğıda vuran 132'ye kadar çekiç olduğundan, tambur yazıcı biraz gürültülü bir makinedir. Aslında, satır yazıcıda bir melodi çalacak karakter dizisi olasıdır. Bu aynı zamanda bilgisayar odasındaki işletmenlerin canını sıkmanın çok iyi bir yoludur.

Zincir veya kayış yazıcı, satır yazıcının bir başka modelidir ve Şekil 5.21'de gösterilmiştir. Sonsuz şerit, her biri kabartmalı bir karakter içeren halkalardan meydana gelmiştir. Kayış yatay düzlemde döner ve şerit, kayış ile kâğıt arasında yerleştirilir. Uygun karakter çekicinin önüne geldikçe çekiç enerjilendirilir ve karakter şeritten kâğıda doğru itilir.

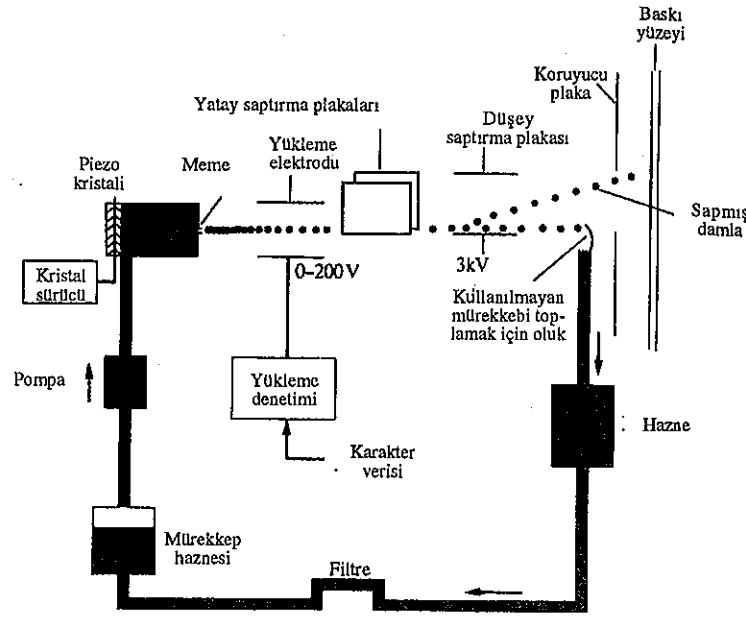


Şekil 5.21 Kayışlı yazıcı

#### Ink-jet (mürekkep-püskürtmeli) yazıcı

Ink-jet yazıcı; çalışabilmesi için CRT'yi vurmalı yazıcıya göre daha çok kullanan ve henüz yaygınlaşmamış bir yazıcı türüdür. Bir ink-jet yazıcının temel özellikleri Şekil 5.22'de gösterilmiştir. İnce bir mürekkep hüzmesi, hızlı bir mürekkep akışı

yaratmak için ince bir memeden püskürtülür. Mürekkebin tek tek damlaları minde ayrılabilmesi için memeler ultrasonik olarak titreştirilir. Her bir damla memeyi terk ederken elektrikle yüklenir. Böylece akış, CRT'deki elektron ışını olduğu gibi, elektrostatik olarak saptırılır. Işının hareket ettirilmesi yolu ile karakterler kağıdın yüzeyine istenen biçimde yazılabilirler. Kağıdın, saptırma yapıldığında mürekkeple kirlenmemesi için kağıt, ışının merkez ekseninin dışına olacak şekilde hizalanır. Kullanılmayan mürekkep tekrar kullanılabilme üzere haznede toplanır.



Şekil 5.22 İnk jet yazıcı

İnk-jet yazıcılar çok hızlı aygıtlardır ve sessiz çalışırlar. Henüz çok pahalıdır ve geliştirilmeleri sırasında çok sayıda 'baş ağrısı' çekmişlerdir. Özellikle memenin tıkanması gibi bir sorunları bulunur.

### 5.5.3 Kullanıcı yönelimli arabirimler

Şimdiye kadar geleneksel bilgisayar giriş/çıkış birimlerini tanımladık: Klavye, VDU, yazıcı. Öte yandan her biri belli bir uygulama için tasarlanmış bulunan giriş/çıkış birimleri çok geniş bir çeşitlilik gösterirler. Örneğin bir haritacı bir haritayı bilgisayara girmek için bir sayısallaştırıcı, işlendikten sonra haritayı çiz-

mek için bir grafik-çizici kullanır. Öte yandan hava alanlarındaki otomatik bagaj sistemi, girişleri insan konuşması (değişen bir hava basıncı) biçiminde alabilir ve çıkışları da bagajları bir taşıyıcı banttandır diğerine aktarmak için gereken hidrolik birimleri denetlemede kullanır.

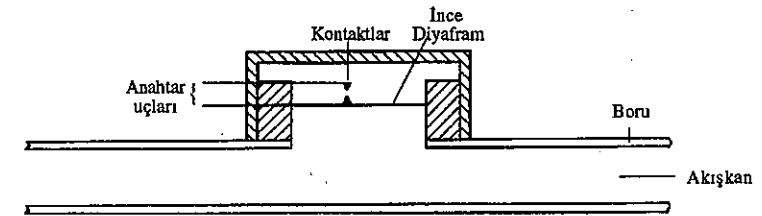
Özel amaçlı giriş/çıkış birimleri konusu sınır tanımaz bir şekilde genişlese de biz burada yalnızca geniş bir özet vereceğiz. G/Ç birimlerinin iki temel sınıfı vardır: sayısal ve analog. Giriş ve çıkışları doğrudan ikili bir biçimde gösterilebilen birimler sayısal birimler olarak sınıflandırılır (şimdiye dek tartışılanlar dahil). Giriş ve çıkışları sürekli olarak değişen ve sonlu sayıda bir değeri olan sinyal biçimindeki birimler de analog birimler olarak sınıflandırılır. Örneğin mikrofon, hoparlör ve termometrenin tümü gerçek anlamda birer analog birimlerdir.

### Sayısal giriş

Sayısal bir giriş bilgisini girebilmek için gereken tek şey, bir mantık 1 ile bir mantıksal 0 arasında seçim yapabilecek bir anahtar kullanmaktır. Pek çok mekanik anahtar elektronik açıdan tümüyle anlamsızdır: İki iletken ya bir devre oluşturmak üzere birbirine itilir ya da devreyi açmak için birbirinden ayrılırlar. Bir durumdan diğerine geçerek çalışan klasik anahtarlar, basma düğmeleri, (bazı radyo ve televizyonların ses ayarlamalarında bulunan) döner düğmeler günlük yaşamımızda sürekli olarak karşımıza çıkar. Kısım 5.5.1'de çok çeşitli anahtar biçimlerinden bazılarını gördük. Birçok anahtar tümüyle gözden geçirilirken diğerlerine hiç değinilmedi.

Hareketli bir birimin daima en uç hareketini algılamak için tasarlanan basit bir sınır anahtarını ele alalım. Eğer asansörün denetleme elektroniki çalışmazsa halat sarma çarkları kabini en üst kattan daha da ileriye çekmeye çalışır ve bu da bir kazaya neden olabilir. Eğer kaldıraca bağlı bir anahtar normal taşıma üst sınırını aşarsa, bu istenmeyen yukarı doğru hareket kaldırıcı iterek bir devreyi kapatacak ve güvenlik birimlerinin devreye girmesine neden olacaktır.

Bir başka mekanik anahtar biçimi Şekil 5.23'te gösterilmiştir. Çoğu zaman bir boru

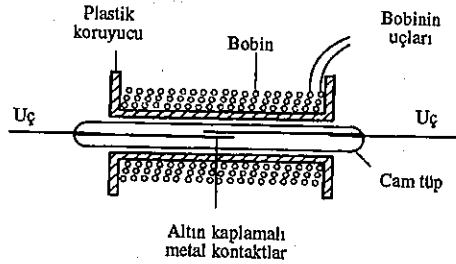


Şekil 5.23 Basınca-duyarlı anahtar

içindeki sıvı ya da gazın basıncının verilen bir değerin altında veya üstünde olduğunu bilmek gerekir. Sıvının, basınçla dışa doğru esneyen ince ve elastik bir diyaframla temas halinde olması sağlanır. Açıkça görüldüğü gibi, diyaframın hareket miktarı sıvının basıncının bir fonksiyonu olarak ortaya çıkacaktır. Diyaframa bir kontak bağlayarak, basınç yeterince arttığında sabit bir kontakın devreyi kapatması sağlanır. Tümüyle mekanik anahtarlara ek olarak, ışık gibi ısı gibi diğer uyaranlara karşılık veren iki-durumlu anahtarlar da vardır. Yukarıdaki tüm sayısal girişler, belli bir olay meydana geldiğinde - anahtarın açılması veya kapanması gibi - bilgisayarı bazı işleri yapmaya zorlarlar.

### Sayısal çıkış

Tüm sayısal bilgisayarların çıktısı, her biri mantıksal 0 için 0.4 volt, mantıksal 1 için de 2.8-5 volt arasında sinyaller biçiminde gösterilen bir veya da ha çok ikili değerlerdir. Bu sinyalleri bazı birimleri açmak (veya kapatmak) için en kolay yol bir röle kullanmaktır. Şekil 5.24'te tipik bir dilli rölenin yapısı görülmektedir.



Şekil 5.24 Röle

Sargıdan akım geçtiğinde demir çekirdekte manyetik bir alan indüklenir. Yeterince yüksek bir değerde, manyetik alan bir parça demiri çekecek güce erişir. Bu parçanın hareketi bir kontakı açar (veya kurar). Sargı, doğrudan bilgisayarın çıkışı ile toprak arasına bağlanabilir. 0.4 volt, sargıyı enerjilendirmek için çok düşüktür, 2.8 volt ise bir çıkış anahtarı çalıştırmak için yeterli olacaktır. Pratik bir sistemde, en küçüğü dışında tüm röleler sargıyı enerjilendirmek için ortalama bir büyüklükte akıma ihtiyaç duydukları sürece bilgisayarın çıkışını yeterli bir düzeye yükseltmek için bazı ek elektronik devrelere muhtemelen gerek olacaktır. Röle mekanik bir anahtarla tamamiyle aynı işlevi yerine getirir ve bu yüzden de bir insan tarafından bir düğmeye basarak veya bir anahtarı çevirerek işletilen her şey bir bilgisayar tarafından da yapılabilir.

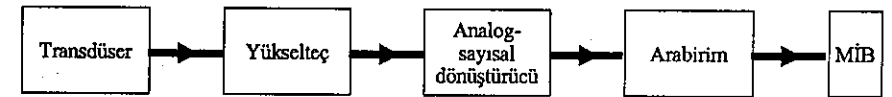
### Analog giriş

Çoğu zaman ölçülebilen niceliklerin büyük çoğunluğunun verilen bir aralık içindeki sonsuz sayıda değerleri alabilen analog bir dünyada yaşadığımız söylenebilir. Bir odanın sıcaklığı analog bir niceliktir çünkü bir değerden diğerine sonsuz küçüklükteki artışlardan geçerek değişir. Benzer şekilde, hava basıncı, hız, ses yoğunluğu, ağırlık ve zaman hep analog niceliklerdir. Şekil 5.25'te değişen bir analog değer zamanın fonksiyonu olarak gösterilmiştir.



Şekil 5.25 Zamanın bir fonksiyonu olarak analog bir nicelik

İlk bakışta analog ve sayısal dünyaların karşılıklı olarak birbirleriyle uyumsuz olduğu görülebilir. Bereket versin ki analog ve sayısal dünyalar arasında nicemleme (quantization) denilen bir geçiş vardır. Analog niceliğin sonsuz aralıkta bir değere sahip olması konu dışıdır. Eğer biri sabah 09.00'da geleceğini söylerse, doğru söylemiyordur. 09.00 sonsuz küçüklükte küçücük bir periyod olarak var olur, dolayısıyla da 09.00'da yer alabilecek hiç bir olaydan bahsedilemez. Elbette ki adam aslında saat 09.00 artı/eksi belirtilmemiş bir saat değerini kastetmektedir. Bir başka deyişle, analog bir değeri ölçüp amacımıza uygun yeterlilikte bir doğrulukta belirtebilirsek gerçek analog değerle, ölçülen değeri arasındaki hata önemsizdir. Analog değer bir kere ölçüldükten sonra ikili bir değere dönüştürülebilen ve bir bilgisayarı besleyebilen sayısal bir biçimde var olur.



Şekil 5.26 Analog niceliklerin okunması

Şekil 5.26'da analog bir niceliği sayısal bir niceliğe dönüştürmek için gereken donanım gösterilmiştir. Transdüserler, analog bir niceliği (sıcaklık, basınç vs.), elektrik sinyaline dönüştüren aygıttır. Burada elektrik sinyalinin düzeyi, ölçülmekte olan niceliğin tanımlanmış bir fonksiyonudur. Pek çok transdüserin çıkışı çok düşük düzeydedir ve bir yükselteç ile uygun düzeye yükseltilmelidir. Ardından

yükseltecin çıkışı, bir analog-sayısal dönüştürücü (ADC) ile sayısal biçime dönüştürülür. ADC'nin çalışması fazla karmaşık değildir ama elektronik bilgisi olmadan açıklanamayacağı için bu kitabın kapsamının dışındadır. ADC'nin çıkışı çoğu zaman ikili biçimdedir ve normal bir sayısal arabirim aracılığıyla bilgisayara uygulanır.

Hava sıcaklığını ölçen bir sistemi ele alalım. Transdüserler birkaç yoldan biriyle çalışır. Bunların hepsi de sıcaklığın değişmesiyle elektriksel özelliklerindeki değişimlere ilişkindir. Transdüserin, sıcaklık  $-10^{\circ}\text{C}$ 'den  $+40^{\circ}\text{C}$ 'ye değiştiğinde 0 volt-tan 1 volta bir çıkış ürettiğini varsayalım. Transdüserin çıkışı, 1 volt bir giriş için \$FF çıkış üreten 8-bitlik bir analog-sayısal dönüştürücüye bağlanır. Böylece  $-10^{\circ}\text{C}$ 'den  $+40^{\circ}\text{C}$ 'ye olan  $50^{\circ}\text{C}$ 'lik aralık, 00000000 ile 11111111 aralığında ikili bir çıkış üretir. Çıkışın her bir biti  $50/256=0.2^{\circ}\text{C}$ 'ye karşılık gelir. Bu desene dönüştürücünün çözünürlüğü denir ve güvenle algılanabilen en düşük sıcaklık değişikliğidir.

Analog nicelikler sürekli olduğundan daima ölçülen gerçek sinyal ile nicemlenen değeri arasındaki fark kadar bir hata bulunur. Bu hata dönüştürücünün çözünürlüğünden küçük veya ona eşittir. En azından teorik olarak tüm ADC'lerde eğrisellik denen küçük bir eksiklik vardır ve bu nedenle kusursuz biçimde davranamazlar. ADC'deki hata genellikle en küçük değerlikli bitinin yarısından daha azdır. Bir bilgisayar için analog bir giriş tasarlarken istenen çözünürlükteki ADC'yi seçmek gerekir. Bunu söylemek yapmaktan daha kolaydır. Sekiz-bitlik tek yongalık ADC'lerin fiyatı çok düşüktür. Çözünürlük arttıkça fiyat da üstel olarak artar.

Bir sinyali bir değerde nicemledikten sonra, zaman olarak da nicemlemelidir. Değişen bir analog sinyali sürekli olarak örneklemek imkansızdır. Her durumda, tipik bir mikroişlemci için bir girişi okumak birkaç mikrosaniye alır. Dolayısıyla bir sinyali ne kadar sıklıkta örneklemeliyiz? Cevap günlük tecrübelerimizden çıkarılabilir. Televizyondaki görüntü hareket ediyor gibi görünür, aslında (bir sinema filmi gibi o da) hareketsiz görüntüler dizisinden meydana gelir. Açıkça görüldüğü gibi hızlı bir hareketsizler (yani örnekler) dizisini izlemek sürekli hareket etkisi uyandırır. Şimdi bir yüzme havuzunun sıcaklığının örneklenmesini ele alalım. Bu gibi büyük bir su kitlesinin yüksek bir termik ataleti olduğundan, sıcaklığı çok yavaş değişecektir. Böylece birkaç dakikada bir alınan örneklemeye yeterli olacaktır. Bu iki örneğin de gösterdiği gibi analog bir niceliğin örneklemeye hızı niceliğin değişme hızıyla orantılıdır.

Buna ilişkin bir teorem vardır ve bir sinyalin içerdiği en büyük frekansa sahip bileşenin frekansının iki katından büyük frekansta örneklenirse tümüyle yeniden oluşturulabileceğini belirtir. Eğer bir ses bilgisayar tarafından işlenmek üzere

örnekleniyorsa ve konuşma sinyalindeki en yüksek frekanslı bileşen 3000 Hz ise saniyede en az 6000 kere örneklemeye yapılmalıdır. Bu teorik açıdan en küçük değerdir ve gerçek bir sistemde daha büyük bir örneklemeye hızına (sözgelimi 10 000 Hz) gereksinim duyulur. Bu, her 0.1 milisaniyede (yani 100 mikrosaniye) bir örneklemeye karşılık gelir. Bir sinyal bir kere örneklenip bilgisayara girildikten sonra öteki herhangi bir sayısal değerin işlenmesiyle benzer biçimde sayısal olarak işlenir.

### Analog çıkış

Bir dizi sayısal değerden analog bir sinyal üretmek bunun tersini yapmaktan çok daha kolay ve ucuzdur. Sayısal-analog dönüştürücüler, çıkışlarında, girilen sayısal girişin ikili değeriyle orantılı bir gerilim üreten bir yonga biçimindedirler.

Sayısal-analog dönüştürücüler (DAC) konuşma veya müzik gibi zamanla değişen sinyal üretmek için de kullanılabilir.

Bellek; bazı olayların maddenin bir veya daha çok özelliğinde neden olduğu uzun veya kısa vadeli değişiklik olarak tanımlanabilir. Yani olay meydana geldikten sonra, neden olduğu değişiklik kalır. Örneğin; havanın soğuk olduğu bir dönemde göl üzerinde oluşan buz, hava ısındığında kolay kolay çözülmez. Su soğutulmuş olduğunu hatırlar. Belleğimiz olmasaydı film izleyemezdik, çünkü, zaman içinde o anki noktadan önce görüntülenenler artık yok olmuştur. Film izlendikçe, gözün arka tabakasında oluşan optik sinyal bellekte birtakım değişikliklere neden olur. Olay geçmiş ancak etkisi belleğimizde kalmıştır. Filmin kendisi bir bellektir. Bir sahnede üretilen ışık fotonları plastik bir levha üzerindeki ince gümüş tabakasının kimyasal yapısını değiştirir. İnsan belleği ve fotoğraf filminin her ikisinin de unutkanlık denilen ortak bir özelliği vardır. İnsan belleği (tekrarlamayla veya bir anımsatıcıyla) yeniden canlandırılmazsa gittikçe zayıflar. En azından belli bir tipteki bilgisayar belleği periyodik olarak anımsatılmazsa sakladığı bilgileri unutacaktır.

Bilgisayarlar, iki nedenden ötürü belleğe ihtiyaç duyar. Birinci olarak, saklı-programlı (von Neumann) bilgisayar kavramı, tümüyle bir dizi işlemin ardışık yürütülmesine dayanır. Açıkça görüldüğü gibi, program komutlar dizisi halinde gerçekleştirilecekse saklanması (anımsatılması) gerekir. Eğer tüm komutlar aynı anda yürütülseydi belleğe ihtiyaç kalmazdı. Bundan başka bu komutlar, geçici veya doğrudan sonuçlar ürettiğinden bunları tutmak için de belli bir miktarda belleğe gereksinim duyulur. İkinci olarak, bilgisayarlar normalde birçok farklı programı yürütürler ve o sırada kullanılmamakta olanlar bir yerde saklanmalıdır. Programlardan başka, çoğu kez elde tutulması gereken çok miktarda veri de vardır.

Bilgisayar teknolojisinin (yani MİB'in) çoğu ikili aritmetik ve Boole cebri ile ilgili olduğundan, makul olan bilgisayar belleğinin de bu yönü izlemesidir. Pek çok bellek sistemi bilgileri, saklama ortamının iki-değerli özelliğini kullanarak ikili biçimde saklar. İnsanlar bazen bunu bilgisayar-dışı uygulamalarda da yaparlar. Örneğin, bir şerit parçasıyla yapılan bildiğimiz fiyongu düşünelim. Bilgiler, tek bir bit olarak saklanır: fiyong bağlı veya bağlı-değil biçiminde. İnsanların neden farklı türde fiyonglar atıp bir parça şeridin bilgi-taşıma kapasitesini arttırmadıklarını hep merak etmişimdir.

İkili bir bellek elemanı için gerekli olan en önemli şey; kararlı iki durumun (en azından kısa bir süre için) varolması ve bu durumların bir enerji engeli ile birbirinden ayrılmış olmasıdır. Durumları ayıran bir enerji engeli olmasaydı ikili bir

değerin durumunu, zorlamayla değiştirmek mümkün olurdu. Şerit örneğimizde, bir fiyong atmak veya onu çözmek için belli bir enerjiye ihtiyaç vardır.

İkili bilgiyi saklamanın birçok yolu vardır ve her bir tekniğin de (maliyet,hız, güç tüketimi, büyüklük vs. olarak) avantajları ve dezavantajları bulunmaktadır. Aşağıda verilenler en popüler saklama yöntemlerindendir:

**Geribeslemeli Elektriksel Saklama:** Elektronik bir anahtar verilen bir mantıksal durumda tutulur çünkü çıkışlar, girişleri, onu sabit bir durumda tutmak üzere geriye besler. Flip-flop veya iki-kararlı aygıt bu tip bir bellek elemanıdır.

**Depolanmış Yükle Elektriksel Saklama:** Elektrik yükü (elektron fazlalığı veya azlığı) yükün boşalmasını engellemek için çevresi elektriksel olarak yalıtılmış bir iletkenle saklanır.

**Manyetik Saklama:** Atomlarda, elektronların kendi etrafında dönmesinin yolaştığı manyetik bir alan vardır. Bu dönüş iki yönde olabilir (soldan sağa veya sağdan sola) ve bu da, iki olası manyetik durum yaratır. Pek çok maddede elektronların dönmesinin yönü rastgele olduğundan (atomların çok güçlü termik titreşimlerinden dolayı) sonuçta bir manyetik etki üretilmez. Öte yandan bir grup madde, ferromanyetik özellik gösterir. Burada birbirine komşu elektronların dönmeleri nedeniyle oluşan etkileşim, bunların paralel bir biçimde hizalanmalarına neden olur. Bu koşullar altında büyük hacimli maddeler mıknatıslandırılabilir. Maddeleri, elektronlarının dönmeleriyle iki durumdan birinde mıknatıslandırıp, ve bu durumları algılayabildiğimizden dolayıdır ki, bilgisayar belleklerinde manyetik maddeler kullanılmaktadır.

**Yapısal Saklama:** Bazı nesnelerin yapısını değiştirerek çok fazla sayıda olası durumlar elde edebiliriz. Pratikte gerçek sistemler, kağıt üzerindeki delikler örneğindeki var-yok ilkesini kullanır. Bu (hızla yokolan) bellek biçimi delikli kartlarda ve kağıt bantlarda bulunur. Bir gramafon plağı, plastik bir diskin içine kesilmiş spiral bir yivin biçimini deforme ederek analog bilgiyi saklayan yapısal bir bellektir. Herhangi bir andaki analog bilgi yivin kenarındaki deformasyonunu derinliğinin fonksiyonudur.

**Uzaysal Saklama:** Süpermenin geçmişi görmesini sağlayan iyi bir numarası vardır. Işık hızından daha hızlı olarak dünyadan dışarı doğru yükselir ve sabit hızla, 300.000 km/san., dünyadan uzaklaşmakta olan eski olayları ışığından izler. Olayların geçmişi, uzayda hareket halinde olan foton akışıyla gösterilir. İlk bilgisayarlar bilgiyi, ses darbelerine, cıva ile doldurulmuş geciktirme hatlarını katederek dönüştürüyorlardı. Darbe dizisi bir uca ulaştığında, tekrar geriye doğru döndürülür. Bu

tür belleğe geciktirme hatlı bellek denir ve artık sayısal bilgisayarlarda yer almamaktadır. Kısım 6.5'te verilen manyetik kabarcıklı bellek uzaysal belleğin modern bir biçimidir.

### *Bellek teknolojisi*

Her birinin kendine has özellikleri olan birçok farklı bellek türü olsa da, bunları tanımlamak için ortak terimler kullanılır. Bellek teknolojisine ilişkin kaynaklarda en çok karşılaşılan bir kaç terim, aşağıda sunulmuştur.

**Bellek Hücresi :** Bellek biriminin en küçük bilgi saklama birimidir ve tek bir mantıksal sıfır veya mantıksal biri tutabilir. Bellek hücreleri çoğu zaman sözcükleri oluşturmak üzere bir araya getirilebilir. Bellekteki her bir hücrenin konumu, adresi ile belirlenir.

**Erişim Süresi :** Erişim süresi, herhangi bir bellek elemanının en önemli parametrelerinden biridir ve bir veriyi verilen bir bellek konumundan okumak için gereken süreyi ifade eder. Okuma çevriminin başlamasıyla ölçülür. Erişim süresi; istenen bellek hücresinin yerini saptamak için geçen süre ve verinin bellek hücresinden kullanılabilir duruma gelene kadar geçen süre olmak üzere iki bölümden oluşur. Bir çok yarıiletken belleğin okuma ve yazma erişim süreleri aynı olduğundan, erişim süresi normalde okuma veya yazma erişim süresini belirtmek için geçen süre olmaktadır. Bu, bütün bellek türleri için doğru değildir çünkü, bazı aygıtların biraz farklı okuma ve yazma erişim süreleri vardır. Bazı bellekler ise çevrim süresine göre tanımlanır. Bu süre, ardışık iki okuma ve yazma erişimi arasında geçmesi gereken süredir. Erişim süresi ile saykıl süresi çoğu zaman birbirinin aynısıdır. Öte yandan bu, yarıiletken dinamik bellekler ve ferrit-çekirdekli bellekler için doğru değildir.

**Rastgele-erişim :** Bu terimle ilk karşılaştığımda safça bunun, rastgele bir bellek hücresi seçilir ve eğer bu istenen hücre değilse bir başka rastgele erişim yapılır anlamına geldiğini düşünmüştüm. Fakat bu o anlama gelmez. Bir bellek içindeki her bir hücrenin erişim süresi sabit (ya da fiili olarak sabit) ve hücrenin gerçek konumundan bağımsız olarak hazırlanırsa bu belleğe rastgele-erişimli bellek (RAM) denir. Yani, rastgele-erişimli belleğin erişim süresi sabittir ve erişilmekte olan verinin hangi konumda bulunduğuyla ilgili değildir. Pratikte bu, MİB'in, bellekten bir sözcük okuması için gereken süreyle ilgili olarak endişelenmemesi, çünkü, tüm okuma çevrimlerinin aynı süreyi aldığı anlamına gelmektedir. Eğer bir bellek, okuma çevrimi açısından rastgele-erişimliyse, değişmez olarak yazma saykalları açısından da rastgele-erişimlidir. Ne yazık ki RAM terimi çoğu zaman

(sadece-okunur belleğe karşı olarak) verinin bellekten okunabildiği veya ona yazılabildiği okuma-yazma belleğini tanımlamak için de kullanılmaktadır. Bu kullanım hatalıdır çünkü rastgele erişim sadece sabit erişim süresi özelliğini belirtir ve belleğin içerdiği veriyi değiştirme (yani yazma) yeteneğiyle bir ilişkisi yoktur. Rastgele erişim için bir başka terim de ani erişimdir. Günlük yaşamımızda rastgele erişimli bellekle yapılabilecek en yakın benzeştirme aramalı telefon sistemidir. Bir aboneyle bağlantı kurma (erişim) süresi sabit ve fiziksel olarak bulunduğu yerden bağımsızdır.

**Seri Erişim :** Seri erişimli bir bellekte, veriye erişmek için geçen süre, verinin bellekteki fiziksel konumuna bağlıdır ve verilen bir sistem için geniş bir aralıkta değişir. Çoğu kez veri okuma/yazma aygıtının bulunduğu yeri geçer ve verilen bir bellek hücresine erişimde bekleme süresi bellek hücrelerinin okuma/yazma birimine gitmesi için gereken zamana bağlıdır. Seri erişimli bellek örnekleri, manyetik teyp birimleri, disk sürücüler, kaydırmalı kaydediciler, manyetik kabarcıklı belleklerdir. Seri erişim aynı zamanda sıralı erişim olarak da anılır.

**Geçici Bellek :** Geçici bellekler, güç kaynağı kapatıldığında içerdiği değerleri kaybeder. Bu olay; verinin bir kondansatörde yük olarak veya iki-kararlı bir devrede transistörün durumu (kapalı veya açık) olarak saklandığı yarı iletken bellek türlerinin pek çoğuna uyar.

**Sadece Okunur Bellek (ROM) :** Sadece okunur belleğin içerdiği değerler okunabilir ancak (normal çalışma koşullarında) değiştirilemez. Gerçek sadece okunur bellekler, tanım gereği geçici değildir. Sadece okunur bellekler, çoğu zaman işletim sistemlerini, yorumlayıcıları, çevirici programlarını ve küçük mikroişlemci sistemlerindeki diğer sistem yazılımlarını tutmak için kullanılır.

**Statik Bellek :** Veriler, statik bir bellek hücresine bir kez yazıldığında, ya üzerine yeni bir veri yazılarak değiştirinceye ya da -eğer bellek kalıcı-olmayan türde bir bellekse- güç kaynağı kapatılıncaya kadar silinmeden kalır. Statik yarı iletken bellek hücreleri, veriyi tutmak için genellikle çapraz-bağlı transistörleri (yani flip-flop) kullanırlar.

**Dinamik Bellek :** Dinamik bir bellekte veriler, alan-etkili transistörün elektrotları arasındaki kapasitansında depolanmış yük biçiminde saklanır. Kondansatör kusursuz olmadığı için yük azar azar sızarak boşalmaya ve verinin kaybolmasına neden olur. Dinamik belleklerin kullanıldığı her zaman (bunlar statik belleklerden çok ucuzdur) kondansatörlerdeki yükü, bellek tazeleme olarak bilinen bir işlemle periyodik olarak (2 milisaniyede bir) yenilemek için ek bazı devrelere gereksinim vardır.

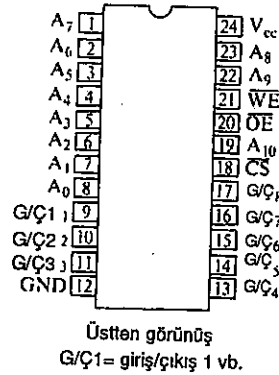
## 6.1 Yarıiletken bellekler

Rastgele erişimli yarıiletken bellekler, mikroişlemcilerin ve öteki benzer sayısal cihazların üretilmesindekine benzer bir işlemle silikon yongalar üzerine imal edilir. Ucuz maliyetli yarı iletken bellekler olmasaydı mikroişlemciler anabilgisayarların ferit-çekirdekli belleklerini kullanmakla sınırlı kalacak ve mikroişlemciler alanındaki devrim ciddi bir gecikmeye uğrayacaktı.

Yarıiletken belleğin temel özellikleri düşük maliyeti, yüksek yoğunluğu (yonga başına düşen bit) ve kullanım kolaylığıdır. Yarıiletken belleğin iki büyük grubu vardır : statik ve dinamik.

## Statik Yarıiletken Bellek

Statik yarıiletken bellekler, tasarımcı veya mühendis açısından kullanımı en kolay olan ve genellikle küçük ve orta boy belleklerde bulunan türdür. Büyük boyutlu belleklerde, düşük maliyetlerinden dolayı dinamik bellekler kullanılır. Tipik bir yarıiletken bellek yongası, 6116 16K CMOS RAM, Şekil 6.1'de gösterilmiştir. CMOS kısaltması, üretiminde kullanılan teknolojisini göstermektedir (TTL ile karşılaştırmız). 16K da belleğin bit cinsinden kapasitesini gösterir. Tüm yarıiletken bellekler kapasitelerine (1K, 4K, 16K, 64K veya 256K) ve organizasyonuna göre belirtilir. Bazı bellekler bit- yapılıdır. Böylece 16K'lık bir yonga, bir bitlik 16K kapasitede konum olarak düzenlenir.



Şekil 6.1 6116 2K x 8 RAM

Ötekiler yarım-bayt (4 bitten oluşan sözcük = nibble) yapılıdır ve örneğin 4K'lık bir yonga her biri 4 bit içeren 1024 konumlu olarak düzenlenmiştir. 6116, 8-bitlik 2K sözcüklü olarak bayt-yapılıdır. Bu türde bayt-genişliğinde yongalar, bir veya

iki yonganın işlemcinin okuma-yazma belleği için yeterli olabildiği küçük bellekli mikroişlemci sistemleri için uygundur. 256K kapasiteli yeni kuşak yongalar yeni yeni piyasaya çıkmaktadırlar.

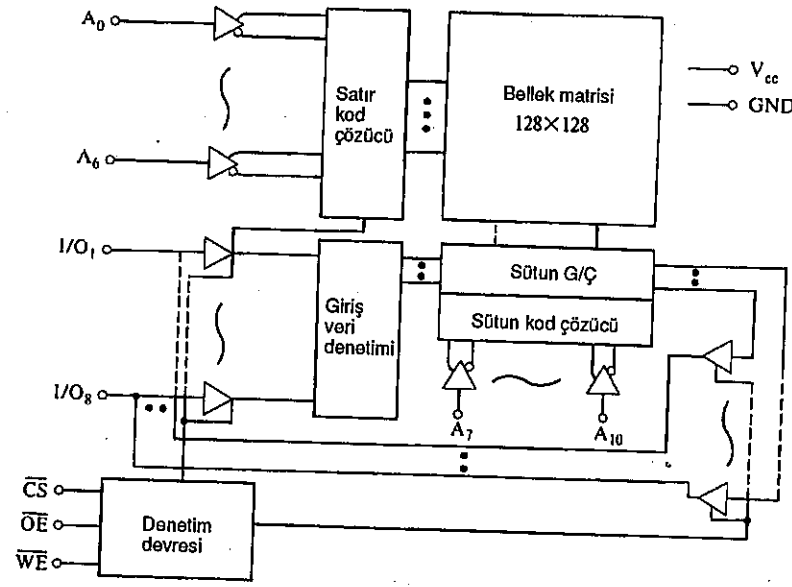
Şekil 6.1'de 6116'nın bacak-çıkışları görülmektedir. 24 bacak vardır. Bunlar içinden 11 tanesi  $2^{11}=2.048$  farklı konumu seçmek için gereken adres girişleridir. Bir MİB okuma çevrimi boyunca 6116'dan veri ileten, MİB yazma çevrimi sırasında işlemciden veri alan, sekiz veri hattı vardır. Yonganın iki d.c. besleme bacağı vardır. Üç tane de denetim bacağı vardır :

|                  |                     |
|------------------|---------------------|
| $\overline{CS}$  | yonga seçim         |
| $R/\overline{W}$ | okuma/yazma         |
| $\overline{OE}$  | çıkış yetkilendirme |

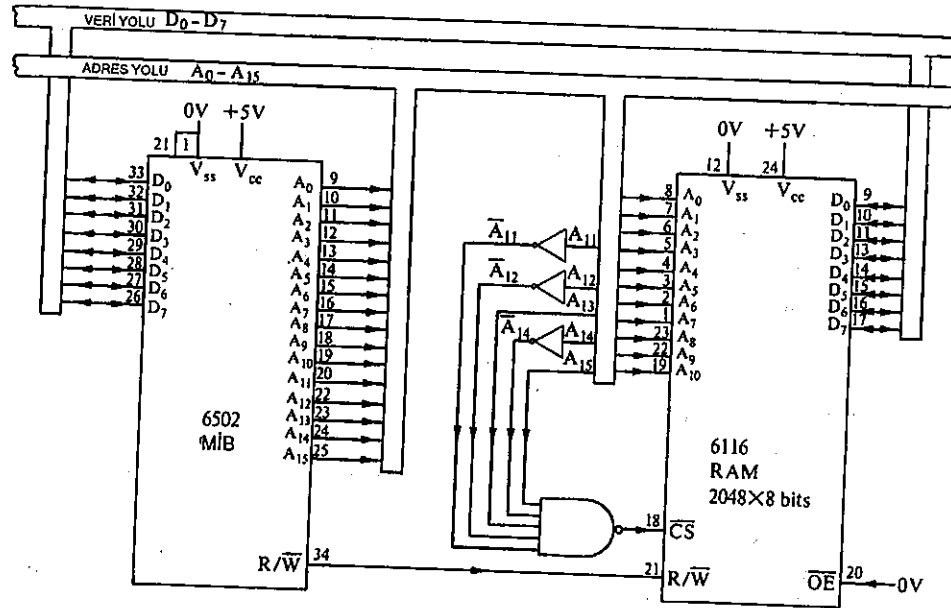
Bir okuma veya yazma işleminde yonganın yeralabilmesi için  $\overline{CS}$  bacağının mantık sıfır durumunda olması gerekir.  $\overline{CS}$ 'nin yüksek olduğu zamanlarda bellek, diğer bacaklardaki işlemleri dikkate almaz. Bu, herhangi bir anda sadece bir 6116 erklendiği sürece birkaç 6116'nın veri yolunu paylaşarak kullanmasını mümkün kılar.  $R/\overline{W}$  girişi yonganın, sekiz veri bacağında veri mi saklıyor ( $R/\overline{W}=0$ ) yoksa bu bacaklara veri mi aktarıyor ( $R/\overline{W}=1$ ) olduğunu belirler.  $\overline{OE}$  bacağı yazma saykılı sırasında yonganın üç durumlu yol sürücülerini kullanma açmak için kullanılır.

Veriler, bu ve öteki statik RAM yongalarında, her biri dört transistörden oluşan flip-floplarda saklanırlar. Şekil 6.2'de bu tip bir yonganın iç düzenlemesi görülmektedir. Bereket versin ki tüm adres kodlayıcı ve okuma/yazma devresi, bellek sisteminin tasarımını büyük ölçüde basitleştirmek üzere yonganın üzerinde konumlandırılmıştır. Şekil 6.3'de, 6116'nın bir 6502 MİB'ine nasıl bağlandığı gösterilmiştir. Bu bellek birimi MİB'in 64K'lık bellek alanından sadece 2K'lık konumuna karşılık verebildiğinden 6116'nın bellek alanının haritasını MİB'inin üstüne çıkarmak gerekir. Yani, yonga sadece MİB'in 2K'lık 32 bloktan birine düşen adreslere karşılık vermek üzere yapılmalıdır. Şekil 6.3'te 6116 için bir  $\overline{CS}$  girişi sağlamak üzere birkaç kapı MİB'in  $A_{11}$ 'den  $A_{15}$ 'e kadar büyük basamaklı beş adres bitinin kodunu çözer.

$A_{15}=1$ ,  $A_{14}=0$ ,  $A_{13}=1$ ,  $A_{12}=0$ ,  $A_{11}=0$  olduğu her seferde yonga, A000 ile A7FF aralığındaki bir adres tarafından seçilir. Yani adres yolu, X'lerin  $A_0$ 'dan  $A_{10}$ 'a kadar olan bacakların durumunu gösterdiği 10100XXXXXXXXXXXX bit deseni içermektedir. Bu, bellek tasarımında bir düzensizliği göstermekten çok bellek yongalarının kullanım kolaylığını belirtmektedir. Birkaç basit kapının dışında sadece 6502 ile 6116 arasında uygun bağlantılar gerekir.

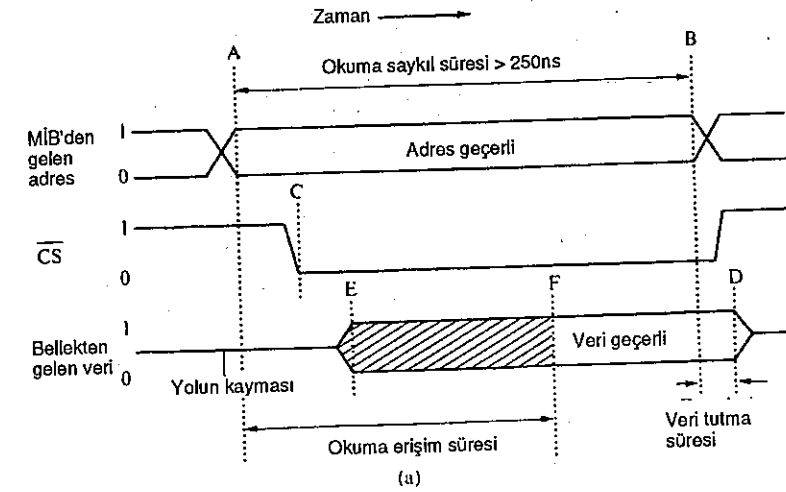


Şekil 6.2 Statik bir RAM'in iç düzenlemesi



Şekil 6.3 6116 statik RAM'ı 6502 MİB'ine bağlanması

Tasarımcının en çok ilgisini çeken, bellek yongasının en önemli parametrelerinden biri olan zamanlama diyagramlarıdır. Bu bir neden-sonuç diyagramıdır ve bir okuma veya yazma saykılı boyunca meydana gelen olaylar dizisini gösterir. Tasarımcı, adres ve veri yolları ile denetim girişleri arasındaki ilişkiyle ilgilenir. Şekil 6.4(a)'da bir okuma çevrimi boyunca bellek yongasının basitleştirilmiş zamanlama diyagramı görülmektedir.



Şekil 6.4(a) Statik bir RAM'in okuma çevrimi zamanlama diyagramı

Şekil 6.4(a)'da adres yolunun zamanlama diyagramı, A ve B noktalarında kesişen iki paralel çizgi olarak görülmektedir. İki paralel çizgi kullanmak genel kuraldır ve onbir adres hattının bazılarının mantıksal sıfır ve bazılarının da mantıksal bir durumda olduğu anlamına gelir. Bu, ilgilenilen adres hatlarının gerçek mantıksal durumları değildir fakat o anki bellek erişim saykılı boyunca adres yolunun içerdiği değer kararlı hale geldiği zamandır.

Şekil 6.4(a)'daki A noktasında adres yolunun içerdiği değer, bir önceki değerinden tamamiyle değişmiştir ve artık kararlıdır. Bu nokta belleğin zamanlama parametrelerinden bazıları için referans olarak alınır. Mantık geçişleri hiçbir zaman anlık olmadığı için durum değişikliğinin tümünü eğimli bir çizgi ile göstermek normaldir.

Adres yolu, A ve B noktaları arasında o esnada okunmakta olan bellek konumunun adresini içerir. Bu süre boyunca MİB'den gelen adres değişmemelidir. A ile B arasındaki süre belleğin, minimum çevrim süresidir. Anılan değer, 250 nanosaniye,

o anki çevrimden sonra en az 250 nanosaniye geçmeden bir başka bellek erişimini başlatabileceği anlamına gelir.  $R/\bar{W}$  hattı, okuma çevriminin tümü boyunca mantıksal bir durumda olmalıdır.

Şimdi de okuma saykılındaki bir yonganın çalışmasını ele alalım. Adres hatlarının büyük basamaklarının kodu çözülür ve Şekil 6.3'teki gibi bunlar, yongayı seçmek için kullanılır. C noktasında yonga seçim girişi,  $\bar{CS}$ , alçak düzeye iner. Bu, veri bacaklarına bağlı üç durumlu yol sürücü çıkışlarının açılmasına neden olur. E noktasına kadar veri yolunun içerdiği değer, iki mantık düzeyi arasında ortayol olarak tek bir çizgi ile gösterilir. Bu kural veri yolunun yüzer durumda olduğunu gösterir ve belleğin veri çıkış devresiyle bağlantısı kesilir.

E noktasında  $\bar{CS}$ 'in alçak düzeye inmesiyle üç-durumlu çıkış devreleri açıldığında, veri yolunun yüzer olma durumu sona erer ve veriler, çıkış uçlarında görünür. Ancak adreslenmiş bellek sözcüğünü yerleştirmek için gereken süre henüz tamamlanmamıştır ve içerdiği değer geri getirilir. Bunun sonucu olarak E ve F noktaları arasında veri yolunun içerdiği değer geçerli değildir ve kullanılmaz. F noktasında veri geçerlidir ve A ile F noktaları arasında geçen süreye yonganın okuma erişim süresi denir.

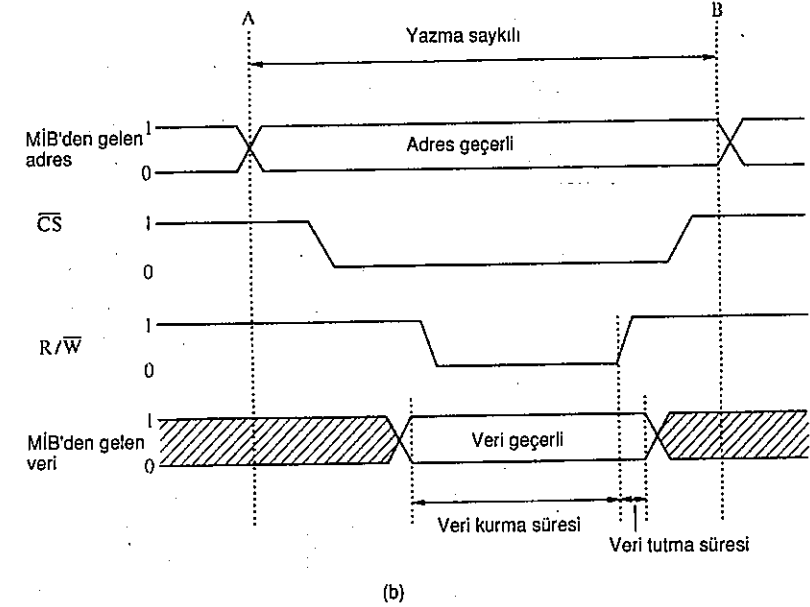
B noktasının gösterdiği okuma saykılıının sonunda, adres yolunun içerdiği değer yüzer duruma geçer. Yongadaki yayının gecikmelerinden dolayı, garanti edilmiş minimum bir süre dolmadan çıkış bacaklarındaki veri değişmez. Buna veri tutma süresi denir ve B ile D noktaları arasındaki süredir. Yukarıdaki açıklama bir okuma çevriminin nispeten basitleştirilmiş bir biçimidir.

Yazma çevrimi,  $R/\bar{W}$ 'in mantıksal sıfır durumunda olması ve verinin, MİB tarafından yonganın veri girişlerine yerleştirilmesi dışında okuma çevrimine benzer. Şekil 6.4(b)'de tipik bir statik yarıiletken RAM'in basitleştirilmiş yazma çevrimi zamanlama diyagramı görülmektedir.

### Dinamik Bellek

Dinamik rastgele-erişimli okuma yazma belleği veya basitçe dinamik bellekler, bugün mevcut olan en elverişli ve düşük fiyatlı yarıiletken bellek biçimidir. Halen dinamik belleğin endüstri standardı olarak kapasitesi 64Kx1 bittir ve 256Kx1 bitlik yongalar üretim aşamasındadır. Dinamik RAM'lar, statik türlerine oranla sadece daha yoğun ve ucuz değil, aynı zamanda daha az elektrik tüketirler. O halde statik bir RAM ile dinamik bir RAM arasındaki fark nedir? Bu sorunun klasik cevabı şudur: 'Statik RAM çalışır ve dinamik RAM çalışmaz'. Bu ifadenin ardında şu

gerçek yatar: statik belleklerden farklı olarak, dinamik bellek sistemleri bunları denetlemek için dikkate değer miktarda bir devreye gereksinim duyar. Ayrıca, bu denetleme devresinin çalışması da biraz kritiktir.

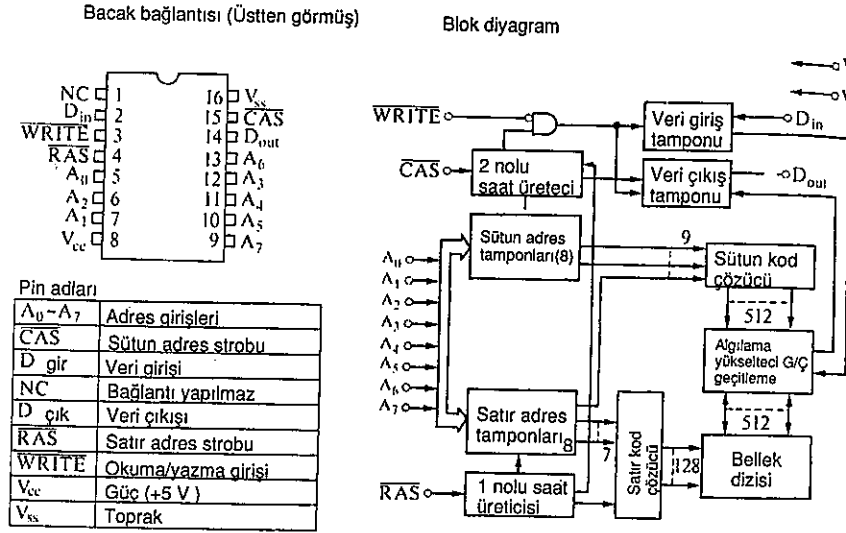


Şekil 6.4 (b) Statik bir RAM'in yazma-saykılı zamanlama diyagramı

Statik RAM verileri, RS flip-flopu gibi işlev gören çarpaz-bağlı iki transistörün durumu biçiminde saklar. Bu ikilideki transistörün biri kesimde, diğeri ise iletimdedir. Tipik bir statik bellek yongası her flip-flop için iki tane daha ek transistöre ihtiyaç duyar. Böylece bu, saklı bilginin her bir biti için toplam dört transistör eder.

Dinamik bellekler, bir bitlik bilgiyi tek transistörlü bir bellek hücresinde saklamak üzere çalışır. Bunun sonucu olarak, yonga başına verilen bir grup transistör için dinamik bir bellek, statik karşılığında dört kat daha fazla miktarda veri saklayabilir. Veriler, dinamik bir bellek hücresinde bir alan- etkili transistörün uçlarında elektrik yükü olarak saklanır. Yükün etkisi, transistörün diğer iki ucu arasındaki akımın yönünü değiştirmektir. Dinamik bir bellek yongasında, bir yazma saykılında mantıksal 1 veya mantıksal 0 yazmak ve bir okuma saykılında içerdiği değeri okumak üzere verilen bir hücreye ulaşmak için gereken tüm elektronik devreler mevcuttur.

Ne var ki, bir bellek hücresinde bulunan transistördeki yük, azar azar kaybolur. transistör içerdiği bilgiyi unuttur. Pek çok dinamik bellek yazıldıktan sonra 2 msn'lik (yani 2000 mikrosaniye) süre için korumayı garanti eder. Dinamik bellekler, verileri 2 msn'den daha uzun süreyle koruyabilmek için verinin hücreye periyodik olarak tekrar yazılmasını ister. Bu işleme tazeleme denir. Dinamik bellek kullanımındaki zorluk, büyük ölçüde bu işleminden dolayıdır. Pratik bir bellek hücresine erişmek onu tazeler ve az ileride göreceğimiz gibi herhangi bir anda 128 veya 256 hücrelik bir grubu tazelemek mümkündür.



Şekil 6.5 64K'lık dinamik bir RAM'ın iç düzenlemesi

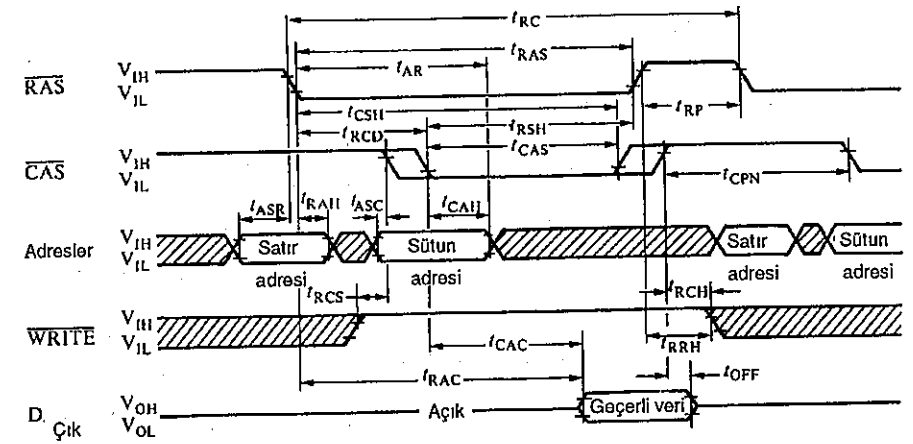
Şekil 6.5'de tipik bir 64Kx1'lik dinamik bellek yongasının iç düzenlemesi görülmektedir. Bu birim 16-bacaklı DIL paketleri halinde bulunmaktadır. Yarıiletken üreticileri, 16-bacaklı bir pakete 64K RAM yerleştirmekle, tasarımcılar için bir başka problem yaratmışlardır. 64K'lık bir bellek alanı için 16 adres hattı ( $2^{16}=64K$ ) gereklidir. Böylece 16-bacaklı bir paket; açıkça adres girişleri, veri hatları, güç kaynağı bacakları ve denetim hatlarının gerektirdiği koşulları yerine getiremez. Bu bacak yetersizliği sorunu, adres yolunun çoğullanmasıyla çözülür - adresler, yongaya 8-bitlik iki bayt olarak girerler. Bu dinamik RAM kullanımındaki güçlüğü bir kat daha artırır. Bu, bellek bileşeni içindeki her bir 64K, adreslenebilir konum için tek bir bit içerdiğinden, sekiz bit genişliğinde bir bellek modülü yapmak için bu yongalardan sekiz tane kullanılması gerekir.

Dinamik bir RAM okuma saykılının basitleştirilmiş zamanlama diyagramı Şekil 6.6'da verilmiştir. Şekil 6.5'te de görüldüğü gibi RAM, yonga içinde dört dizi

128x128'lik bellek hücresi içerecek biçimde düzenlenmiştir. Bu hücelere, istenen hücrenin satır ve sütunu belirtilerek erişim yapılır.

Okuma çevrimi, RAM'ın adres girişlerine uygulanan o anki bellek konumunun 8-bitlik satır adresi ile başlar. Sonra yonganın RAS'ı (satır adres strobu) mantık 0 durumuna getirilir ve satır adresi yonganın içindeki flip-floplara mandallanır. Bundan sonraki adım, 2-bitlik sütun adresini yongaya uygulamak ve sütun adres strobunu (CAS) alçak düzeye getirmektir. 16-bitlik adres ile erişilen hücredeki veri, daha sonra RAS alçak düzeye geçtikten, tipik olarak 150-300 nsn'lik bir gecikmeden, sonra veri çıkış hattında gözükür.

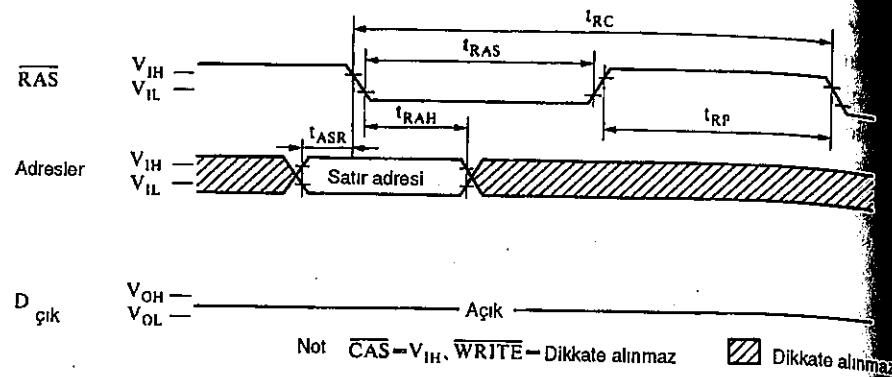
Aslında yazma çevrimi de CAS alçak düzeye geçmeden önce yazma girişinin alçak düzeye geçmesi ve adreslenmiş hücrede saklanacak verinin veri giriş hattında uygulanması dışında okuma saykılıyla benzerlik gösterir.



Şekil 6.6 64K'lık dinamik bir RAM'ın okuma-çevrimi zamanlama diyagramı

Tazeleme işlemini gerçekleştirebilmenin birkaç yolu vardır, ama en kolaylarından biri yalnızca RAS tazeleme saykılı olarak bilinir ve Şekil 6.7'de gösterilmiştir. Satır tazeleme adresi, yonganın adres girişine uygulanır ve RAS girişi, belirlenmiş bir süre için alçak düzeye çekilir. Her tazeleme işleminden sonra, satır tazeleme adresi bir artırılır ve böylece sonuçta, 128 satırın tümü tazelanmış olur. Tazeleme adresi yonganın dışındaki bir devreden sağlanmalıdır. En son bazı 64K dinamik yongaları, kendi içlerinde tazeleme saykıl sayıcıları içermektedir.

Çevrim süresine eşit bir erişim süresi bulunan statik RAM'dan farklı olarak, dinamik RAM'da saykıl süresi erişim süresinden daha uzundur. Bunun nedeni bir

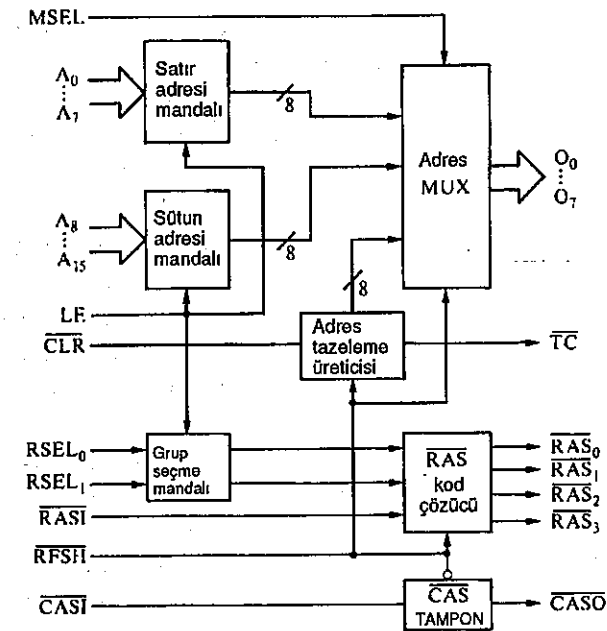


Şekil 6.7 Yalnızca-RAS tazeleme saykılı

başka erişim başlamadan önce belli iç işlemlerin yapılmasının zorunlu oluşudur. Tipik bir yongada erişim süresi 150 nsn., saykıl süresi ise 300 nsn'dir.

Dinamik bir belleği denetlemek için gereken devrenin nasıl belirlendiği, Şekil 6.8'de verilmiştir. Dinamik bellek denetimi, adres çoğullamayı gerçekleştirmeli ve gereken  $\overline{RAS}$  ve  $\overline{CAS}$  sinyallerini üretmelidir. Aynı zamanda her 2 msn'de 128 satır tazelemeyi de yapmalıdır. Bunu yapabilmek için MİB'e bir tazeleme isteği göndermeli ve MİB'den bir tazeleme onayı beklemelidir. Tazeleme isteği onaylandığında denetleyici, belleğe bir satır tazeleme adresi gönderir ve  $\overline{RAS}$ 'ı alçak düzeye geçmeye zorlar. Tazeleme adres sayıcısı otomatik olarak artırılır. Bereket versin ki dinamik bir bellek denetleyicisini gerçekleştirmek için gereken devrelerin büyük çoğunluğu, piyasada tek bir yonga biçiminde bulunmaktadır.

Yariletken dinamik bellekler, bu tip belleklere özgü iki zayıflığın etkisi altındadır. Bir bellek hücresine erişildiğinde ve elektrotlararası kapasitör yüklendiğinde, dinamik bellek güç kaynağından o kadar çok yük çeker ki bu, güç kaynağı hatlarında gerilim düşmesine neden olur. Bu sorun, bellek sistemi devresinin dikkatlice planlanmasıyla aşılabılır. Dinamik belleğin bir başka zayıflığı alfa parçacıklarına karşı duyarlılığıdır. Yonga, az da olsa radyoaktif madde içeren plastik veya seramik bir maddenin içine konur. Radyoaktif bozunmanın ürünlerinden biri, yüksek şekilde iyonize olan ve geçtiği hücrelerdeki verinin bozulmasına neden olan alfa parçacığdır (helyum çekirdeği). Alfa parçacıklarının sayısı kapatma maddesinin seçiminde, dikkatli kalite kontrolü yapılmasıyla azaltılabilir ama hiçbir zaman sıfıra indirilemez.



**Şekil 6.8 Dinamik belleklerin denetlenmesi**

Kişisel bilgisayarların pek çoğunda araya ortaya çıkan bit bozulması yalnızca sinir bozucu olmakla kalırken, profesyonel sistemlerde çok daha ciddi sonuçlara neden olabilir. Bu problemin pratikte çözümü hata düzeltme kodlarıyla yapılır. 16-bitlik bir veri sözcüğünde, 21-bitlik bir kod sözcüğü üretmek üzere ona eklenen beş bit vardır. Eğer kod sözcüğü dinamik RAM'dan geri okunduğunda hatalı bir bit varsa bunun hangi bit olduğunu saptamak ve böylece hatayı düzeltmek mümkündür.

Dinamik RAM kullanmak, statik RAM'dan daha zor olduğundan; daha çok mini-bilgisayar ve anabilgisayarlarda bulunur. Öte yandan son birkaç yıldır dinamik RAM'lar, birçok kişisel bilgisayar üreticileri tarafından benimsenmektedir. Sadece (tipik olarak 16K'dan az) çok küçük bellekli mikroişlemci sistemlerinin statik RAM kullandığı görülmektedir. Daha büyük birkaç mikroişlemci sistemleri dinamik RAM'dan daha güvenilir olduğu için, daha pahalı da olsa statik RAM kullanmaktadır. Popüler 8-bitlik mikroişlemcilerden biri olan Z80, özellikle dinamik bellek ile arabirim kurabilecek biçimde tasarlanmıştır. Z80'de, periyodik olarak 8-bitlik bir tazeleme adresi üretilir.

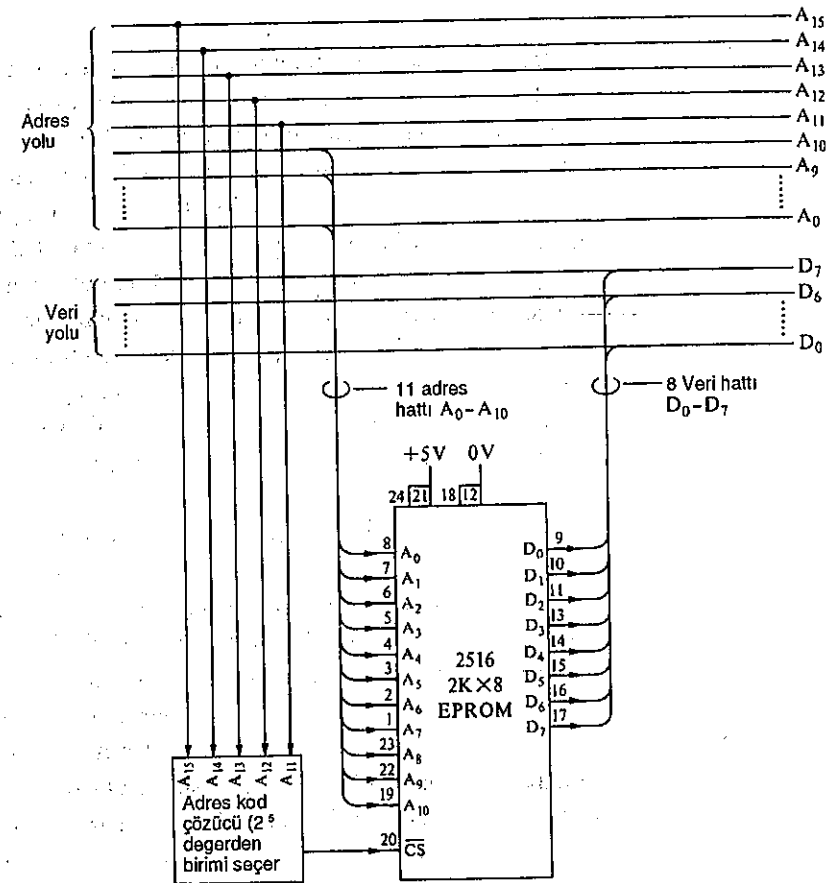
## Öteki Yarıiletken Bellek Türleri

**ROM:** Yarıiletken teknolojisi yüksek-yoğunlukta, düşük-maliyetli, sadece-okunur bellek üretimine son derece uygundur. Birkaç ROM türü vardır: maske-programlı PROM, EPROM ve EAROM en sık karşılaşılan türlerdir. Genelde, yarıiletken ROM'lar, aksi belirtilmediği sürece maske-programlı türdedir ve yonga üretilirken programlanır. Maske-programlı ROM, 8K'dan 256K bite kadar sırasıyla 1Kx8 ve 32Kx8 şeklinde organize edilmiş, 24 veya 28-bacaklı DIL paketleri halinde bulunmaktadır. ROM'un uygulanması yarıiletken statik RAM'dan bile daha kolaydır. ROM yongasında, üzerlerine hiçbir zaman yazma yapılamadığından, erişilecek konumun adresi ve yonganın veri yolunun çıkış devrelerini etkinleştirecek bir yonga seçim (veya yonga yetkilendirme) sinyalinden başka bir şeye ihtiyaç duyulmaz. Şekil 6.9'da, tipik bir ROM'un bacak çıkışları verilmiş ve ROM'un bir işlemcinin adres ve veri yollarına nasıl bağlandığı gösterilmiştir.

Kişisel bilgisayarlar ve diğer düşük-fiyatlı bilgisayarların geliştirilmesinde, öteki bileşenler kadar ROM'un da payı vardır. Tipik bir işletim sistemi ve BASIC yorumlayıcısının yaklaşık 8K ile 16K bayt arası belleğe gereksinimi vardır ve bu yüzden de (tipik olarak) iki veya dört tane 32 Kbitlik ROM yeterli olmaktadır. Eğer bu tipte büyük ROM'lar olmasaydı sistem yazılımının disketlerde (genellikle disket sürücü ve ilgili güç kaynağı ile denetleyicileri 300 sterlin veya daha yüksek fiyatlardadır) ya da ucuz, yavaş ve güvenilir olmayan dahili kaset teyplerinde saklanması gerekecekti. ROM'lar yeniden programlanamadığı veya değiştirilemediği halde bilgisayarın sistem yazılımını içeren ROM'ları çıkarıp bunları güncelleştirilmiş bir versiyon ile değiştirmek hiç de pahalı değildir.

ROM'ların bir başka uygulaması da özel amaçlı mikroişlemci-tabanlı denetleyiciler alanındadır. Bir mikrobilgisayar belli bir görev, sözelimi bir otomobilin kontakt denetleme sistemi için ayrıldığında, yazılım, aygıtın kullanım ömrü boyunca sabit kalmaktadır. Bu türde yazılımı saklamanın en maliyet-etkin yöntemini ROM'lar sağlarlar.

**EPROM.** EPROM'lar, silinebilir programlanabilir sadece-okunur belleklerdir. Maske-programlı ROM ile temel bazı özellikleri ortaktır ve hatta bir çok EPROM, maske-programlı ROM için hazırlanmış soketlere takılabilir. EPROM'lar, fiyatı yüz dolar veya daha yüksek özel bir programlama cihazı ile kullanıcı tarafından programlanabilirler. EPROM'daki bilgiler, oldukça iyi yalıtılmış iletkenler üzerinde elektrostatik yükler biçiminde saklanır. Yük boşalması olmadan on yıldan fazla bir süreyle saklanabilir.



Şekil 6.9 Bir yarıiletken ROM'un bacak çıkışları ve uygulaması

EPROM'un maliyeti aynı yoğunluktaki bir maske-programlı ROM'un birkaç katıdır. Fakat işletim sistemlerinin EPROM'da yüklü bulunduğu küçük ölçekli mikroişlemci sistem üretimleri için oldukça uygun olmak gibi bir avantajı vardır. ROM'lar çok ucuz olmalarına karşın üretici normalde fiyatın üzerine, birkaç bin sterlin tutarında maske maliyeti ekler. EPROM'lar, içerdikleri verilerin, silikon yonganın kızılötesi ışık kaynağı altında silinebilmesinden dolayı, bir çok kez kullanılabilirler. Silikon yongalar, kızılötesi ışığı geçiren quartz bir pencerenin altında DIL paketine yerleştirilirler. Kızılötesi (Ultraviyole) ışık, saklı yükün yalıtıktan sızıp boşalmasına neden olur.

EPROM'lar programlanıp, silinip, yeniden programlanabildiğinden hem küçük ölçekli projeler için hem de laboratuvarındaki geliştirme çalışmaları için son derece uygun birer gereç olmaktadır. Bir program bir kez EPROM'a yazılıp hatalı ayıklandıktan sonra gerekiyorsa bir maske-programlı ROM'a da aktarılabilir. EPROM üzerinde yapılacak bir hata, maske-programlı ROM'da yapılacak hatalarla oranla daha düşük bir maliyetle çözülebilecektir.

**EAROM** : EPROM'un en son gelişmiş biçimi elektriksel olarak değiştirilebilir ROM'lardır. Bu EPROM türü, elektriksel olarak silinip programlanabilir. Bunun sonucu olarak bu haliyle programlanabilir ve ayrıca bir kızılötesi (UV) ışık kaynağı altına konması da gerekmez. Bu aşamada okur, bir okunur-yazılır RAM ve EAPROM arasındaki farkın ne olduğunu sorabilir. EAROM tipik bir yarı iletken RAM'dan farklı olarak, geçici değildir. Ne yazık ki yarı iletken RAM'dan çok daha pahalıdır, çoğu zaman standart-dışı güç kaynaklarına ihtiyaç duyar ve nispeten uzun bir erişim süresine sahiptir. EAROM enerji kesilse de verinin korunmasının gerektiği özel uygulamalarda kullanılır. Tipik bir uygulama radyo alıcısıdır. Alıcı bir grup farklı frekansları saklayabilir ve enerji tekrar uygulandığında bunları anımsar.

**PROM** : PROM'lar (programlanabilir sadece okunur bellekler), EPROM'lar gibi kullanıcı tarafından programlanabilirler. Öte yandan PROM yongasındaki ince metalik bağlantılar o denli büyük bir akımla eriterek programlar ki sonunda bağlantılar tamamen erir. Sürekli bir bağlantı, mantıksal 1'i ve ayrılmış (erimis) bir bağlantı da mantıksal 0'ı gösterir. Açıkça görüldüğü gibi, PROM'lar yalnız ve yalnız bir kez programlanabilirler. Yapılış biçiminden dolayı PROM'un çok düşük bir erişim süresi (50 nsn) vardır ve çoğunlukla program saklamaktan çok bir devre elemanı olarak kullanılırlar.

**Sözde ROM** : Yakın zamana kadar 6502 de dahil olmak üzere LSI yongalarının çoğu NMOS işlemi ile üretiliyordu. Bu yongalar normal çalışmada birkaç on ile birkaç yüz miliamper aralığında bir akıma gereksinim duyarlar. LSI yongaları üretmek için bir başka işlem de CMOS'tur ve bu üretim biçiminde çok daha az güç harcanarak yonga üretilmektedir. CMOS teknolojisi bir süredir kullanılmakta olduğu halde 1980'lere kadar CMOS'un maliyeti özel-amaçlı cihazlar hariç diğer tüm aygıtlar için yanına yaklaşılmaz oranda yüksekti. CMOS bellekleri, erişim yapılmadığı sürelerde o kadar az bir akım çekerler ki, yaklaşık 2.8 V'luk küçük bir pil ana güç kaynağı kapatıldığında saklı veriyi korumak için yeterli gelebilmektedir. Bu tür bellek kesinlikle okunur-yazılır tür bellektir, fakat pratikte geçici olmadıklarından ROM'un bazı özelliklerine sahiptir.

## 6.2 Ferrit-çekirdekli bellekler

Ferrit-çekirdekli bellekler, bilgisayar endüstrisinin ilk gençlik çağına aittir. Bilgisayarlı çalışmaların ilk günlerinde, İkinci Dünya Savaşından hemen sonra, sayısal verileri saklamak için her türde değişik ve garip cihazlar kullanıldı. Örneğin 1949'da Williams, verileri, bir santimetre kareye 10 bite kadar yoğunlukla bir tübün yüzünde elektrostatik yükler biçiminde saklamak üzere bir katot ışıma tüpü kullandı. Ferrit-çekirdek ilk olarak 1953'te MIT Whirlwind bilgisayarında tanıtıldı. Ferrit-çekirdekler verileri, içindeki manyetik alanda saatın dönme yönünde veya tersi yönde saklayan toroid biçimli küçük bir manyetik malzemedir.

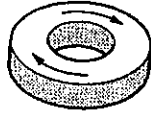
Birkaç yıl öncesine kadar, boyutu, maliyeti ve hızı ile ferrit-çekirdek, bilgisayarların ana doğrudan erişimli belleğinde en mantıklı seçim haline gelmişti. Yarıiletken bellekler, daha yüksek yoğunluklar (yonga başına bit sayısı) daha yüksek hızlar (daha az erişim süreleri) ve çok daha azaltılmış güç tüketimi sunarken maliyetinde de diklemesine bir inişi sürdürdükçe ferrit-çekirdekli bellekler hızla demode olmaktadır. Ferrit-çekirdekli bellekler, kalıcı oldukları ve erişim yapılmadığı sürelerde (yarı iletkenlerden farklı olarak) enerji tüketmediklerinden birkaç alanda hala kullanılmaktadır. Düşük akım çekme özellikleriyle CMOS yarıiletken belleğinin gelişi, yakında ferrit-çekirdekli belleklerin bu alanlardan da çıkmasına neden olacaktır. Ancak şunu da eklemeliyim ki ferrit-çekirdekli bellekler, öteki tür belleklerden daha dayanıklı ve dolayısıyla da koşulları ağır ortamlara daha iyi uyum sağladıklarından, belli bazı endüstriyel uygulamalarda hala kullanılmaktadır. Çekirdek belleğin daha modern bir türü olan manyetik-kaplamalı tel bellekler, artık geniş ölçüde ferrit-çekirdekli belleğin yerini almıştır.

Ferrit-çekirdekli belleğin ilkelerinin bugün hala öğretiliyor olmasının üç nedeni vardır: Birincisi, sayısal aygıtların iki-durumlu yapısını çok açık bir biçimde göstermektedir. İkincisi, ve daha garip olanı, çalışmasının birkaç sayfada kolayca açıklanabilir ve 'Ferrit-çekirdekli bir belleğin ilkelerini açıklayın' sorusu, temel bilgisayar bilimleri sınavlarında boşluk doldurmada kullanılır. Ne yazık ki yarı iletken belleğin temellerini 'birkaç sayfada' açıklamak neredeyse imkansızdır çünkü yarı iletken belleğin çalışması son derecede karmaşıktır ve maddenin elektronik özelliklerine ilişkin dikkate değer bir bilgiye sahip olmayı gerektirir. Üçüncü olarak, ferrit-çekirdekli belleklere ilişkin temel ilkeleri, saklama ortamları olarak hala saltanatını sürdüren disk ve teyp birimlerine de genişletilebilir.

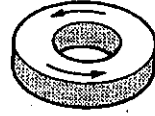
Ferrit-çekirdeği veya toroid, çapı normalde 0.01 ile 0.07 mm arasında olan halka biçimli bir manyetik malzeme parçasıdır. Mıknatıslandırıldığında çekirdeğin içindeki manyetik alan, çekirdek düzlemine doğru eğilir ve manyetik alanın yönü, Şekil

6.10'da da gösterildiği gibi saat yönünde veya tersi yönde olur.

Pratik bir bellekte ferrit-çekirdek kullanmak için iki şey, yani; verileri saklamak için çekirdeği istenen durumda mıknatıslandırabilmesi ve verileri okumak (geri almak) üzere çekirdeğin mıknatıslanma yönünün belirlenebilmesi gerekir.



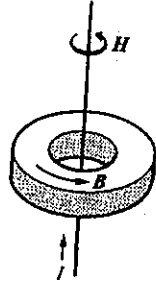
Manyetik alan, saatin dönme yönünde  
(mantıksal 0 durumu)



Manyetik alan, saatin tersi yönünde  
(mantıksal 1 durumu)

Şekil 6.10 Ferrit çekirdek

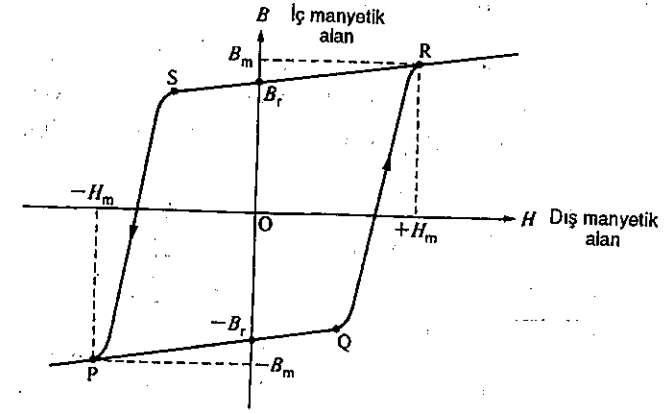
Çekirdekte veri saklamak zor değildir ve "İçinden  $I$  akımı akan bir iletken çevresinde bir  $H$  vektörel manyetik alanı üretilir" ilkesine dayanmaktadır. Şekil 6.11'de  $I$  akımı taşıyan ve bir ferrit-çekirdek içinden geçen bir iletken görülmektedir.



Şekil 6.11  $I$ ,  $H$  ve  $B$  arasındaki ilişki

$I$  akımı iletkenin etrafında  $H$  yoğunluğunda bir manyetik alan yaratır.  $H$  manyetik alanı, akan  $I$  akımı ile orantılıdır. Çekirdek içinde, dıştaki  $H$  alanı ile çekirdeği oluşturan malzemenin iç mıknatıslanmasının bileşimi olarak bir  $B$  manyetik alanı üretilir. Şekil 6.12'de, dik histerezisli bir ferrit için  $B$  yoğunluğundaki iç manyetik alan ile  $H$  yoğunluğundaki dış manyetik alan arasındaki ilişkinin grafiği verilmiştir. Bu eğriye histerezis eğrisi denir.

Bir histerezis eğrisinden çıkarılacak sonuçları ele almadan önce ferromanyetik malzemenin yapısını incelemek yararlı olacaktır. Daha önce de belirtildiği gibi mıknatıslanmanın temelinde elektronların kendi yörüngelerindeki hareketi yatmaktadır. Pek çok maddede elektron dönüşünün manyetik etkisi, atomların ısı titreşimleriyle üretilen -ve komşu atomlar arasındaki manyetik etkileşimi durdurma



Şekil 6.12 Histerezis eğrisi

etkisine sahip bulunan- daha büyük bir güç ile tümüyle giderilebilir. Demir ve onun bazı bileşimleri gibi ferromanyetik malzemelerde, elektronların dönüşünün, malzemenin domen adı verilen bir bölgesi üzerinde hizalanması ile sonuçlanan güçlü bir etkileşimi vardır. Bu domenlerin büyüklüğü 1 mikrometre ile birkaç santimetre aralığındadır. Elektron dönüşleri bir domen içinde aynı hızda olduklarından bu bölge, çok güçlü ve kendiliğinden bir mıknatıslama sergiler ve küçük bir mıknatıs gibi davranır. Bununla birlikte, büyük bir parça ferromanyetik malzemede, domenlerin herbiri, rastgele düzenlenmiş manyetik eksenlere sahiptir. Bundan dolayı büyükçe bir malzemede genel bir manyetik alan yoktur. Bu aşamada, histerezis eğrisi tanımlayacak ve yukarıda verilen ferromanyetizma modeli ile aralarındaki ilişkiyi göstereceğiz.

Dış manyetik alanın sıfır olduğunu varsayalım. Yani  $I=0$  olduğundan  $H, 0$ 'a eşittir.  $B$ 'nin olası iki değeri vardır;  $+B_r$  mantıksal 1 durumunu,  $-B_r$  ise mantıksal 0 durumunu gösterir.  $B_r$  sembolündeki r soneki ortak alanı gösterir ve dış manyetik alan sıfıra eşitken çekirdekte kalan mıknatıslanmaya karşılık gelir. Flip-floplarda olduğu gibi, ferrit-çekirdeğinin de iki kararlı durumu vardır ve bu durumların her birinde sınırsız süreyle kalabilir. Flip-flop'dan farklı olarak, ferrit-çekirdekler kalıcı birer saklama ortamıdır ve veriyi korumak için ek bir güç kaynağı gerektirmezler.

Başlangıçta, çekirdeğin mantıksal 0 durumuna mıknatıslandırıldığını varsayalım. Eğer dışardan negatif bir manyetik alan (yani negatif  $I$  ve dolayısıyla negatif  $H$ ) uygulanırsa,  $B$ 'nin değeri,  $B_r$ 'den biraz daha fazla negatife gider ve Şekil 6.12'deki P noktasına doğru gelmiş oluruz. Eğer  $H$ , bu aşamada sıfıra indirilirse kalan

mıknatıslanma  $-B_r$ 'ye geri döner. Bir başka deyişle çekirdeğin durumunda net değişme yoktur.

Şimdi de  $H$  manyetik alan değerinin küçük bir pozitif değerde olduğunu varsayalım. Eğri boyunca  $Q$ 'ya doğru hareket ederiz. Eğer dış manyetik alan azaltılırsa  $-B_r$ 'ye geri döneriz. Ne var ki, eğer  $H$ ,  $+H_m$ 'nin daha da ötesine arttırılırsa çekirdeğin mıknatıslanması  $Q$ 'da sıçrama yapar ve  $R$  noktasında son noktaya ulaşırız. Eğer bu aşamada  $H$ 'yi sıfıra indirirsek,  $-B_r$ 'ye değil  $+B_r$ 'ye döneriz. Yani çekirdek başlangıçta negatif durumdaysa dış manyetik alan şiddetini  $+H_m$ 'nin ötesine kadar arttırmak, çekirdeğin pozitif bir durumu almasına neden olur.  $H_m$ 'den daha az bir manyetik alan çekirdeğin durumunu değiştirmek için yetersiz kalır.

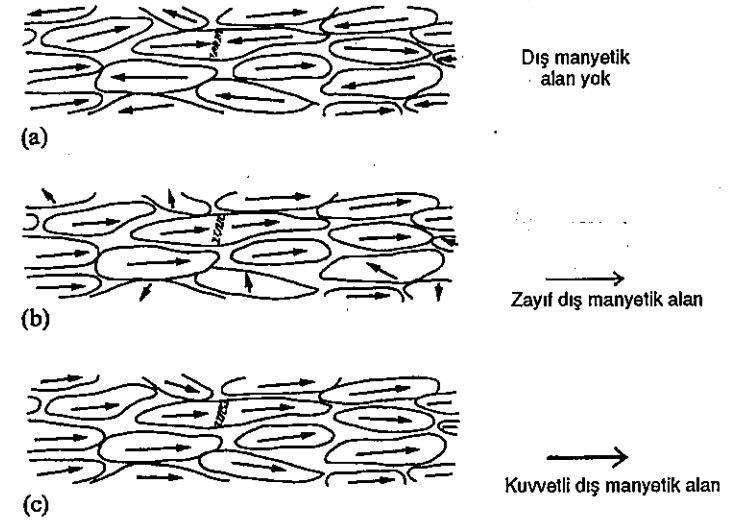
Benzer biçimde, eğer çekirdek belli bir durumdaysa ( $+B_r$ )  $H$ 'nin aldığı pozitif bir değerinin çok az bir etkisi vardır fakat  $-H_m$ 'den daha az bir negatif  $H$  değeri çekirdeği sıfır durumuna ( $-B_r$ ) değiştirecektir.

Çekirdeğin bir durumdan diğerine değiştirilmesi, çekirdekten geçen bir iletken  $I_m$ 'den büyük bir akım uygulayarak yapılır.  $+I_m$  değerindeki bir akım darbesi çekirdeği daima mantıksal bir durumuna,  $-I_m$ 'lik bir darbe de mantıksal sıfır durumuna geçmeye zorlayacaktır.

Histerezis eğrisi, domenlerin davranışları açısından kolayca açıklanabilir. Şekil 6.13'te ferromanyetik malzemenin belli bir bölgesi üç aşamada görülmektedir. (a) aşamasında, manyetik malzemenin, domenlerin rastgele düzenlenmesiyle ilk durumunda olduğu ve genel bir mıknatıslanmanın bulunmadığı söylenir. Bu, histerezis eğrisinin merkezine karşılık gelir; burada  $H=0$  ve  $B=0$ 'dır.

(b) aşamasında dışardan bir manyetik alan uygulanır ve manyetik alanla aynı hizaya gelebilmek için domenlerin bazıları kendi manyetik eksenlerini döndürürler. Dışardaki alan arttıkça daha çok domen döner ve öyle bir nokta gelir ki dışardaki alanla aynı hizaya gelmiş olan domenler manyetik alanın daha da güçlenmesine böylece daha çok domenin dönmeye neden olur. Kısa bir süre sonra bu işlem iç manyetik alan hızla oluştuğunda, bir çığ biçiminde gelişir ve tüm domenler dış manyetik alanla aynı hizaya gelirler. Bu noktada ((c) aşaması) malzemenin tümü tamamiyle mıknatıslanmıştır ve böylece doymuş olduğu söylenir.

Şekil 6.12'deki histerezis eğrisine  $B-H$  eğrisi denir ve bu eğri, bir manyetik malzemenin diğerine farklılık gösterir. Genelde, veri saklamada en elverişli  $B-H$  eğrisi, kare olanıdır ve bir durumdan diğerine (yani  $-B_r$ 'den  $+B_r$ 'ye) geçiş,  $H$ 'de görülmeyecek kadar küçük değişimle meydana gelir. Bu tip bir manyetik malze-



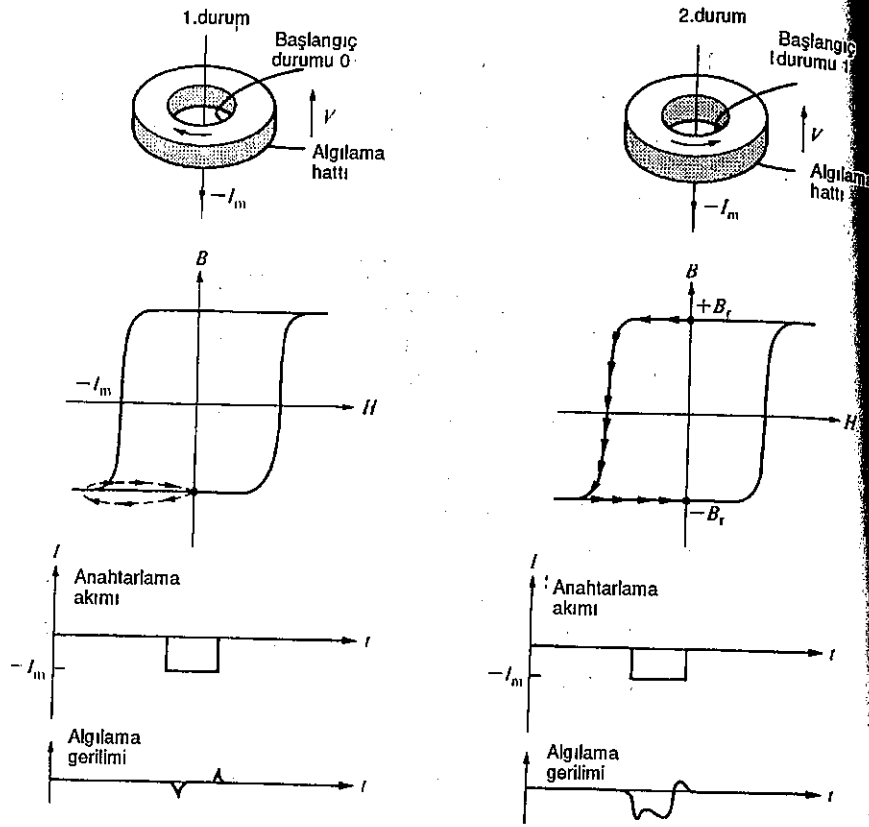
Şekil 6.13 Ferromanyetik bir malzemenin mıknatıslanılması. (a) Mıknatıslanmanın olmadığı durum - net toplam mıknatıslanma sıfır. (b) Bir manyetik alanın olması halinde domenlerin dönmeleri - dış manyetik alanın yönünde %50 oranında bir mıknatıslanma (→) (c) Dış manyetik alan yönünde %100 oranında bir mıknatıslanma (doyma).

menin dik histerezisli  $B-H$  özelliğine sahip olduğu söylenir. Güçlü histerezis etkisi gösteren manyetik malzemelere 'sert', az veya hiç histerezis göstermeyenlere de 'yumuşak' malzeme denir.

Bir çekirdeği mıknatıslandırdıktan sonra mantıksal 1 mi yoksa 0 mı saklıyor olduğunu öğrenmek üzere durumunun belirlenmesi gerekir. Ne yazık ki çekirdeğin hangi durumda olduğunu doğrudan belirlemek fiilen imkansızdır, ancak dolaylı yoldan belirlenebilir.

Çekirdekten geçen ve algılama iletkeni denilen ikinci bir iletkeni ele alalım. Böyle adlandırmanın nedeni çekirdeğin durumunu algılamasındandır. Çekirdeğe, bir  $-I_m$  akım darbesi uygulandığında iki şeyden biri olabilir. Eğer çekirdek başlangıçta sıfır ( $-B_r$ ) durumundaysa, bu durumda kalır, eğer bir  $(+B_r)$  durumundaysa, darbe ile sıfır durumuna değiştirilecektir. Şekil 6.14'te bu iki işlemin, algılama iletkenine verilen gerilimle nasıl değiştiği gösterilmiştir.

Birinci durumda çekirdek durum değiştirmez ve algılama hattındaki gerilim, iki



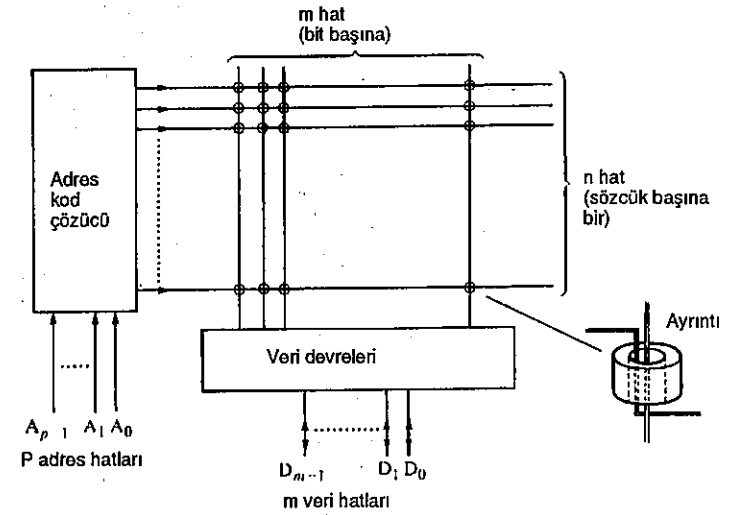
Şekil 6.14 Ferrit çekirdeğin okunması

küçük darbeden (değiştirme darbesinin integralinden) oluşur. 2. durumda algılama hattında, çekirdeğin durum değiştirmesiyle oluşan manyetik akıdaki değişikliğin neden olduğu çok daha büyük bir darbe vardır. Şekil 6.14'te gösterilen iki histerezis diyagramında, bu iki durumda çekirdeğin çizdiği yollar gösterilmiştir.

Burada anlaşılmış olması gereken şey bir çekirdeğin okunmasıyla, içerdiği verinin bozulacağıdır. Bu nedenle ferrit-çekirdekler, bozucu-okuma (DRO) aygıtları olarak bilinirler. Bir okuma saykılı meydana geldikten sonra sorgulanan tüm çekirdekler sıfır durumunda bırakılır. Bu nedenle, veriyi geriye almak için bir okuma saykılına bir yazma çevrimi izlemelidir. Bu nedenle bir ferrit-çekirdeğinin okuma erişim süresi saykıl süresinden daha kısadır, çünkü veriler, bir okuma işleminin ardından tekrar yazılmadan önce yeni bir erişim başlayamaz.

### Çekirdek bellek düzlemi

Şimdiye dek bir bellek hücresini gösteren bir çekirdeğin özelliklerini inceledik. Pratikte bir ferrit-çekirdekli bellek düzenlemesinde,  $n$  sözcüğe karşılık  $m$  bit gerekmektedir. Olası bir sistem, Şekil 6.15'te gösterilmiştir.



Şekil 6.15 İki-boyutlu (2-D) çekirdek bellek

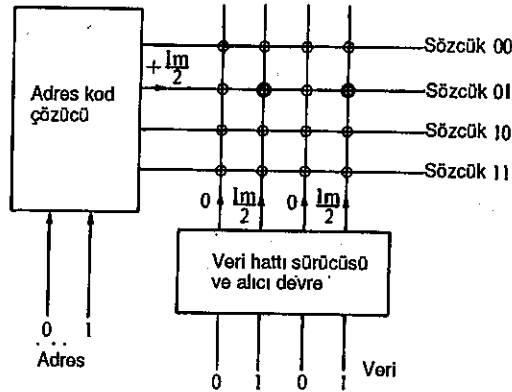
Bellek düzenlemelerinin çalışmasındaki anahtar, çakışan-akımla seçme tekniğidir. Bir çekirdeği değiştirmek için gereken akım en az  $I_m$  kadardır ve bir iletkendeki tek bir darbe ile veya ayrı iletkenlerdeki bir grup darbe ile sağlanır. Yani her biri  $I_m/2$  taşıyan iki iletken, içinden  $I_m$  akımı geçen tek bir iletkenin işlevini görebilir. Şimdi dizinin  $i$ . sözcüğüne veri yazma durumunu ele alalım. Adres kod çözücü  $n$  tane sözcük hattından birini seçmek için  $p$  bitlik sözcük adresi kullanır ( $n=2^p$ ). Eğer  $-I_m$ 'lik bir darbe  $i$ . sözcük hattına (yatay olarak) uygulanırsa, o hatta sıralanmış  $m$  tane çekirdeğin tümü de sıfır durumuna geçmeye zorlanacaktır.

$i$ . sözcüğe mantıksal 1 yazmak için  $+I_m/2$  bir akım darbesi uygun sözcük hattına uygulanır. Akım, elbette, tek başına çekirdeklerin mıknatıslanmasını değiştirmeye yetmez. Eğer  $+I_m/2$ 'lik bir darbe,  $m$  tane veri hattından birine uygulanırsa bu sözcük tarafından dizilen çekirdek ve veri hattından akmakta olan toplam akım  $+I_m$  olur ve bu da çekirdeği, mantıksal bir durumuna değiştirmek için yeterlidir. Şimdi artık buna neden, çakışan-akımla seçme tekniği dendiği anlaşılmış olmalıdır.

Şekil 6.16'da 4-bitlik belleğin 4 sözcüğüne, 0101 veri değerini, 01 sözcüğüne yazdığı gösterilmiştir. Eğer 01 sözcüğünde tüm çekirdekler bir  $-I_m$  darbesiyle durumuna kurularsa, yalnızca bu iki çekirdek kendilerine doğru akan toplam  $+I_m$ 'lik bir akıma sahip olduklarından, koyu gölgeli iki çekirdek mantıksal 1 durumuna değiştirilecektir. Düzlemdeki diğer tüm çekirdeklerden ya hiç akım akmaz, ya da bunlara doğru akan ve durumlarını değiştirmek için yeterli olmayan  $+I_m/2$ 'lik bir akım söz konusudur.

Bir okuma saykılı, uygun bir sözcük hattına  $-I_m$ 'lik bir darbe uygulayarak gerçekleştirilir. Burada veri hatları algılama hatları olarak işlev görür ve çekirdeğin sıfır mı yoksa birde mi olduğunu algılar. Şekil 6.17'de yukarıdaki örnekteki verinin nasıl okunduğu gösterilmiştir.

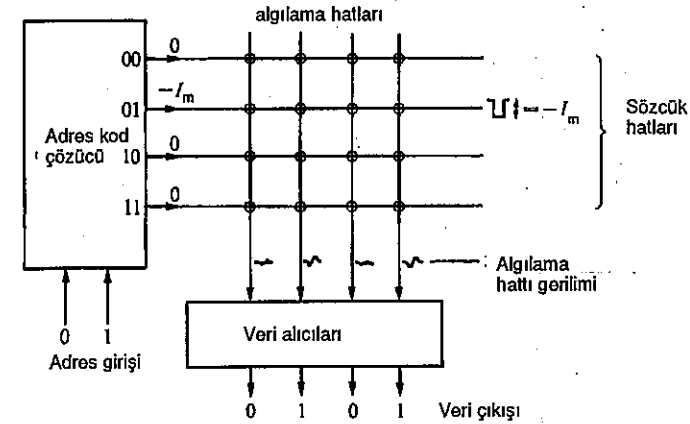
Bir kez daha belirtmek gerekirse, okuma saykılı veriyi bozmuştur ve geri almak için bir yazma saykılına gerek olacaktır. Bu nedenle ferrit-çekirdeğin erişim süresi saykıl süresinden daha kısadır. Saykıl süresi, çekirdek saklama alanının ardışık okuma erişimi arasında geçmesi gereken minimum süredir.



Şekil 6.16 Bir yazma-çevrimindeki 2-D çekirdek bellek örneği. 0101 verisinin 01 sözcüğüne yazılması, 01 adresine karşılık gelen sözcük satırına  $+I_m/2$  akımı ve dört veri hattına da sırasıyla, 0,  $I_m/2$ , 0,  $I_m/2$  akım darbelerinin uygulanmasıyla gerçekleştirilir.

### 6.3 İnce film türü bellek

İnce film türü bellekler, ferrit-çekirdekli belleğin mantıksal bir devamıdır. Burada ferrit-çekirdek yerine, ince bir film biçiminde manyetik bir malzeme, ya da bir iletken üzerine kaplanmış manyetik malzeme tabakası kullanılır. İkinci tip düzenlemeye kaplanmış-tel bellek denir.



Şekil 6.17 2'D çekirdek belleğin okunması.  $-I_m$ 'lik bir akım darbesinin 01 sözcük hattına uygulanmasıyla, bu hatla bağlantılı dört çekirdeğin tümü mantık sıfır durumuna geçmeye zorlanır. Çekirdeklerin okunması, dört veri hattındaki gerilimin algılanmasıyla gerçekleştirilir.

Ferrit-çekirdekli belleğin sahip olduğu iki dezavantaj, daha iyi bir çözüm arayışına yol açmıştır. Bunlar çekirdeğin nispeten yavaş (tipik olarak 1 mikrosaniye ile 50 ns.'den az bir aralıkta) değiştirme hızı ve bunlar arasındaki bağlantıların üretilmesindeki zorluklardır.

Çok ince nikel-demir filmler, soy bir altkatmanda kalıntı bırakır. Yaklaşık 1 mm. kalınlığı olan bu filmler iki önemli manyetik özellik gösterirler. Ferrit malzemeyle aynı domen yapısına sahip değildir ve mıknatıslanmanın yönünü değiştirmek, domenleri döndürmekten çok moleküler düzeyde meydana gelir. Tüm pratik uygulamalar için ince filmler, tek bir domen gibi davranırlar.

İlgi çekici diğer bir özelliği de, manyetik anizotropisidir (yönsüzlüğüdür). Manyetik özellikleri, büyük ölçüde uygulanan dış manyetik alanın yönünden bağımsız olan birçok ferrit malzemeden farklı olarak ince filmler, manyetik alanın düzenlenmesine göre farklı manyetik özellikler sergilerler. Anizotropi birçok malzemenin ortak bir özelliğidir - ağaçları damarları boyunca yarmaya çalışmak tersini yapmaktan çok daha zahmetsizdir.

Film, ilk olarak alt katman üzerine yatırıldığında sürekli bir dış manyetik alanın varlığından dolayı tortu bırakır. Bunun sonucu olarak filmde, iki tane manyetik eksen ortaya çıkar. Eksenlerin birine kolay-yön denir ve filmin üretimi sırasında mevcut bulunan dış manyetik alanın yönündedir. İnce film kolay-eksende ferrit-çekirdeğe çok benzer bir dik-histerezisli  $B-H$  özelliği gösterir.

Öteki eksen, mıknatıslanmanın zor-ekseni, kolay-eksene diktir ve neredeyse dorusal  $B-H$  ilişkisine sahiptir. Yani, iç manyetik alan yoğunluğu,  $B$ , dış alan yoğunluğu  $H$  ile, doğru orantılıdır ve bir miktar histerezis gösterir.

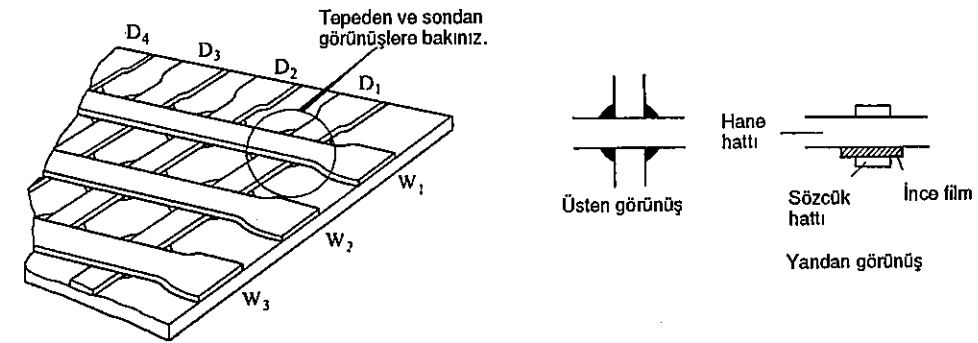
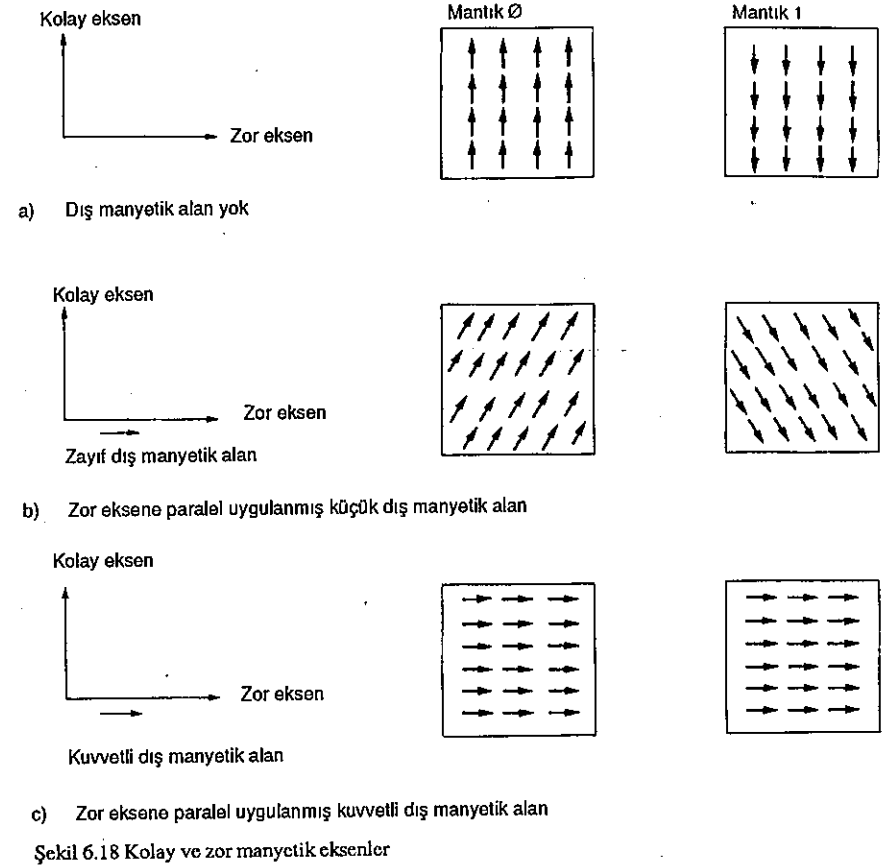
Okur aşağıdakileri en iyisi ince filmi neredeyse bir ferrit-çekirdekli saklama ortamı gibi davranıyormuş biçiminde dikkate alırsa daha kolay kavrayabilecektir. İnce film ile ferrit-çekirdeği arasındaki en büyük fark, çekirdeğin iki-durumlu, ince filmin ise dört-durumlu bir eleman olmasıdır. İnce filmin durumlarından iki tanesi kararlıdır ve mantıksal 1 veya 0'a karşılık gelir. Diğer ikisi ise kararsızdır ve dış bir alanın mevcut bulunmadığı zaman korunamazlar. İnce filmi, madeni bir parça olarak düşünün. Yere düştüğünde üç şekilden birinde olur: yazı veya tura (kararlı durumlar) ve diklemesine (kararsız durum).

Şekil 6.18 (a)'da dış bir manyetik alanın olmadığı bir durumdaki iki ince film görülmektedir. Biri mantıksal 0 diğeri de mantıksal 1 durumundadır. Şekil 6.18(b)'de zor-eksene paralel dış bir alan uygulanmıştır ve bu da, ince filmdeki alanın görüldüğü gibi dönmesine neden olur. İç alan zor ve kolay-eksenlerdeki alanların vektörel toplamına eşittir. Eğer dış alan sıfıra indirilirse filmin mıknatıslanması Şekil 6.18(a)'daki gibi ilk durumuna geri döner (çözülür).

Eğer dış alan zor-yönde Şekil 6.18(c)'deki düzeye çıkarılırsa filmin mıknatıslanması kendisini zor-yönle paralel hizaya getirir. Bu esnada iki örneğin de, ne mantıksal 1 ve ne de mantıksal 0 durumunda bulunmadığına dikkat ediniz. Film bir kez bu duruma geldikten sonra zor-yöndeki alan ortadan kaldırıldığında, kolay-eksene paralel küçük bir manyetik alan uygulanması, filmin durumunu belirleyecektir. Bu, madeni paranın diklemesine durmasıyla olan benzerliğe karşılıktır. Yani madeni parayı herhangi bir tarafa ya da diğerine doğru itip düşürmek için çok az bir güç gerekir.

Şekil 6.19'da iki boyutlu sözcük-yapılı ince film türü saklama ortamının nasıl gerçekleştirildiği görülmektedir. Sayı hatları ince filmlerin mıknatıslanmasının zor-eksenine paralel gider. Sayı hatlarına dik olarak sözcük hatları bulunur. Bunlar saç tokası döngüleri şeklinde düzenlenmiş olup sayı hatlarının üstünden karşıya geçip altlarından geriye dönerler.

Filmden veri okumak için, sözcük hattı altından, filmlerin zor-ekseni boyunca bir manyetik alan yaratmak üzere sözcük akımı geçirilir. Bu, seçilen sözcük hattı tarafından içinden geçilen tüm ince filmlerin iç alanlarını kararlı eksenlerinden 90 derece döndürmelerine neden olur. Dönen manyetik alan, filmin üstünden geçen her bir sayı hattı üzerinde gerilim indüklenmesine yol açar. Dönme yönü, indüklen-



Şekil 6.19 2-D ince-film bellek

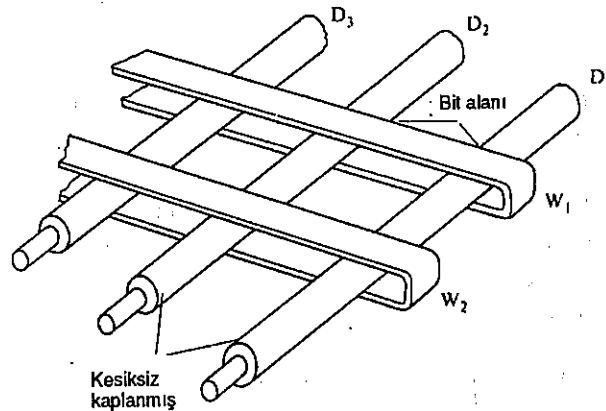
nen gerilimin polarizasyonunu (kutuplamasını) belirler ve dolayısıyla da filmin sıfır durumunda mı yoksa bir durumda mı olduğu bu yolla anlaşılır.

Veri; bir sözcüğün bitlerine, bunları gösteren filmleri zor-eksen boyunca manyetize olmak için zorlayacak bir sözcük darbesinin uygulanmasıyla yazılır. Ardından sayı hatlarına, kolay-eksenin altında veya üstünde manyetik alanlar üretmek üzere, bir akım uygulanır. Sözcük akımı kesildiğinde filmlerin mıknatıslanması kolay-eksenle saat yönünde veya tersi yönünde döner. Dönmenin yönü sayı hatlarındaki akımın polarizasyonu ile belirlenir.

İnce film türü bellekler, ferrit-çekirdekler gibi bozucu okuma modunda ya da bozucu-olmayan okuma modunda (NDRO) kullanılabilirler. Eğer sözcük akımı mıknatıslanmayı tümüyle 90 derece döndürmek için yeterliyse, okuma bozucudur. Eğer sözcük akımı, dönme 90 dereceden az olacak şekilde dikkatle kontrol altında tutulursa, çekirdeğin yönü yine de sayı hatlarında indüklenen gerilim aracılığıyla belirlenebilir. Sözcük akımı kesildiğinde iç mıknatıslanma kendi halinde kalır.

Ancak, NDRO modunda çalışırken sözcük akımının dikkatli biçimde kontrol altında tutulması gerekir. Akım şiddeti çok düşük düzeydeyse manyetik alan, sayı hattında güvenilir bir algılama yaratacak oranda yüksek bir gerilim indüklemeye yetecek miktarda dönmez. Çok yüksekse, mıknatıslanma 90 derece kadar döndürülebilir ve saklı bulunan veri kaybedilecektir.

Yukarıda açıklanan düzlemsel ince-film türü saklama ortamı, algılama hattıyla manyetik filmler arasındaki zayıf kuplajdan oldukça fazla etkilenir. Şekil 6.20'de hatlarının ince filmle kaplanmış olduğu alternatif bir düzenleme görülmektedir. Sözde kaplanmış-tel saklama ortamı düzlemsel bir ince-film türü saklama ortamıyla tamamiyle benzer biçimde çalışır fakat üretimi daha kolaydır. Sayı hattıyla ince film arasındaki kuvvetli kuplajdan dolayı aynı sözcük-sürme akımı için veri hattına indüklenen sinyal, düzlemsel bir saklama ortamında buna karşılık gelen sinyalden



Şekil 6.20 Kaplanmış-tel bellek

yüz kat daha fazladır. Gerçek bir ince film türü saklama ortamının hızı ince filmin kendisiyle değil, ilgili elektronik devre tarafından belirlenir. Kaplanmış-tel saklama ortamlarının sayıl süreleri 250 nsn. civarındadır ve birçok anabilgisayarda ferrit-çekirdeğinin yerini almıştır. Kaplanmış-tel saklama ortamı, çoğu kez uzay gemileri ve uydulardaki sistem kartında-yerleşik bilgisayarlarda bulunur. Kalıcı olmaları, kullanılmadığı sürelerde sıfır güç tüketimleri ve radyasyondan etkilenmemeleri en önemli özellikleridir.

#### 6.4 Bellek hiyerarşisi

Herhangi bir gerçek bilgisayar sisteminde bilgi saklamanın birçok yolu vardır. Örneğin, MİB'in kendi içinde akümülatör, yığın işaretçileri, indeks kaydedicileri ve program sayıcısı gibi iç kaydediciler vardır. Ve yine doğrudan erişimli saklama ortamları (ana bellek, çekirdek bellek veya RAM) bulunur. Hatta bu bile, kimi zaman yüksek-hızlı önbellek ve daha yavaş çekirdek bellek olarak iki gruba ayrılır. Önbellek Kısım 9.3'te tartışılacaktır. Ayrıca disk bellekler ve teyp birimleri mevcuttur. Eğer bunların hepsi veri saklıyorsa neden bu kadar çok çeşitli saklama ortamlarına gereksinim duyarız?

Bunun ve aşağıdaki sorunun cevabı aynıdır: Neden uçak yerine trenle seyahat ederim? Yaşamın her alanında olduğu gibi burada da ekonomiklik belirleyici olmaktadır. Bilgisayar tasarımcısının bir bellek biriminde görmek istediği özellikler genelde tam bir uyum içinde değildir. İdeal bellek şu özelliklere sahiptir:

1. Yüksek hız. Erişim süresi çok düşük olmalıdır, tercihan 2 nsn veya daha az.
2. Küçük boyut. Fiziksel olarak küçük olmalıdır. Santimetrekübe bin megabayt idealdir.
3. Düşük enerji tüketimi. Bellek sisteminin tamamı bir kol saati piliyle on yıl çalışmalıdır.
4. Oldukça sağlam. Bellek, sözelimi mantıksal 1'in bir anda mantıksal 0'a dönüşmesi veya tersi gibi hatalara açık olmamalıdır. Aynı zamanda -60 °C ile 200 °C arasındaki sıcaklıklarda çalışabilmelidir (Askeri açıdan bu çok önem taşımaktadır).
5. Düşük maliyet. Bellek bedelsiz biçimde alınabilmeli ve yazılımla birlikte parasız dağıtılmalıdır.

Şimdi de gerçek bellek sistemlerinin özelliklerine bir bakalım.

**MİB içi bellek.** MİB'lerdeki kaydediciler, MİB'in kendisiyle aynı teknolojiyle yapıldıklarından çok düşük bir erişim süresine sahiptirler. Çok pahalıdır ve enerji tüketirler. Çoğu kez bu, iç kaydedicilerin ve MİB'in kendi içindeki 'yaz-belleğin sayısını sınırlar. Bir yongaya eklenebilecek kaydedici sayısı son yıllarda müthiş bir şekilde artsa da silikon yonga üzerinde yapılan MİB'ler için bu özellik hala geçerlidir.

**Ani Erişimli Saklama Ortamları.** Bu bellek, yürütülmeleri süresince program ve veriyi tutar ve nispeten daha hızlıdır (50 nsn. - 1 mikrosaniye arası). Normal yarıiletken veya ferrit-çekirdekli bellekler (kaplanmış-tel bellekler, ferrit-çekirdekli belleklerin bir türüdür) olarak gerçekleştirilir. Belleğin maliyeti yüksektir, nispeten çok enerji tüketir ve fiziksel olarak çok büyüktür.

**Manyetik disk.** Manyetik diskler, küçük bir alanda çok büyük miktarda veri saklayabilir ve bit başına maliyeti çok düşüktür. Ne var ki seri erişimli bir aygıtın erişim süresi (insan hızına oranla hızlı olsa da) doğrudan erişimli saklama ortamlarından daha düşük bir değerdedir. Tipik bir disk sürücü, 300 megabayt saklayabilir ve 25 msn'lik bir erişim süresine sahiptir.

**Manyetik teyp.** Manyetik teyp aşırı derecede ucuz bir seri erişim cihazıdır ve maliyeti 6 sterlin olan bir teypte 46 megabayt kadar veri saklanabilir. Ne yazık ki ortalama erişim süresi aşırı uzundur.

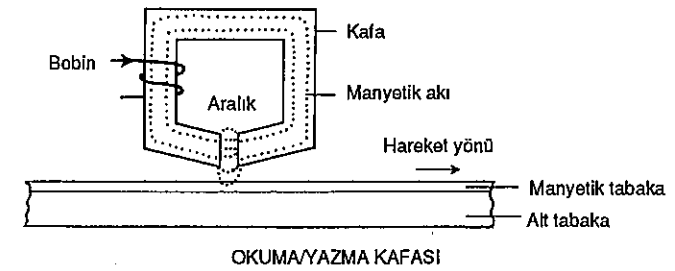
Bilgisayar tasarımcısı, tüm bu bellek türlerini tek bir bilgisayar sisteminde birleştirerek büyük bir olanak elde edebilir ve hız performansı doğrudan erişimli belleğinkinden yüzde birkaç kez daha düşük olan nispeten düşük maliyetli bir bellek sistemi oluşturabilir. Bilgisayar bellek tasarımının anahtarı, doğru veriye doğru yerde ve doğru zamanda sahip olmaktır. Çok büyük bir bilgisayar sisteminin binlerce programı, milyonlarca veri dosyası olabilir ancak herhangi bir anda MİB yalnızca bunlardan birkaçına ihtiyaç duyar. Eğer biz, MİB'in istediği veriyi daimi (veya buna yakın) doğrudan erişimli bellekte bulabilecek biçimde verileri, disklerden ve teyplerden doğrudan erişimli saklama ortamına taşıyabilen bir işletim sistemi tasarlırsak, çok düşük bir maliyetle dev hızda bir saklama ortamına sahip bir sistem elde etmiş oluruz. Bu tip bir düzenlemeye sanal (virtüel) bellek denir. Çünkü, sözelimi, aslında bir megabaytlık gerçek bir doğrudan erişimli saklama ortamı ve bin megabaytlık disk saklama ortamı varken, toplam bellek büyüklüğü MİB'e 1000 megabayt olarak görünecektir. Kısım 9.3'te sanal bellek sistemleriyle ilgili daha fazla ayrıntı sunulmuştur.

### Saklama birimleri

Bir bilgisayar belleğinin; yüksek-hızlı, yüksek-maliyetli, düşük kapasiteli doğrudan erişimli bellek ve düşük hızlı, düşük maliyetli, yüksek kapasiteli saklama ortamı şeklinde iki gruba ayrıldığını görmüştük. Yakın gelecekte kabarcık bellekler kullanıma girecek ve bazı optik bellek biçimleri (örneğin Video disk) kendini köşeden gösteriyor olsa da bugün yedekleme ile saklama ortamı, teyp birimleri ve disk sürücülerle eşanlamlı olarak kullanılmaktadır.

#### 6.4.1 Manyetik kayıt

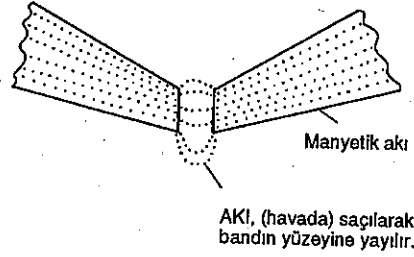
Disk sürücülerle teyp birimlerinin çalışması sanal olarak aynıdır; biri veriyi esnek ince bir banda kayıt ederken diğeri manyetik bir malzemeyle kaplı düz bir plakaya kaydeder. Bu, Şekil 6.21'de gösterilmiştir.



Şekil 6.21 Okuma/yazma kafası

Veriyi yazmak için kullanılan yazma kafası (veya kayıt kafası) etrafı bir bobinle sarılı yüksek geçirimsizlikte yumuşak manyetik malzemeden yapılmış bir halkadan meydana gelir. Yüksek geçirimsizlik, malzemenin manyetik bir alana karşı düşük bir direnç göstermesi anlamına gelir. Yazma kafası ile içinden bir yazma hattı geçen ferrit-çekirdek arasındaki benzerliğe dikkat ediniz. Yazma kafasının yapılmış olduğu malzemenin, ferrit-çekirdek ile aynı dik histerezis özelliği yoktur ve artıklık ya da kalıntı mıknatıslanma sergilemez. Yazma kafasının çok önemli bir özelliği, halkada bulunan ince bir hava aralığıdır. Bobinden akım akarken çekirdek içinde manyetik bir akı yaratılır. Bu akı çekirdeğin etrafında akar fakat hava aralığıyla karşılaştığında Şekil 6.22'de görüldüğü gibi etrafa saçılır.

Aralık etrafında yaratılan manyetik alan saklama ortamını kaplayan manyetik malzemeden geçer. Bu alan yeterince güçlüyse, kaplamanın içindeki manyetik taneciklerin (domenlerin) alanla aynı hizaya gelmesine neden olur. Manyetik yüzey hareket ettikçe yüzey kesiksiz bir bant halinde manyetize edilir. Eğer akımın yönü



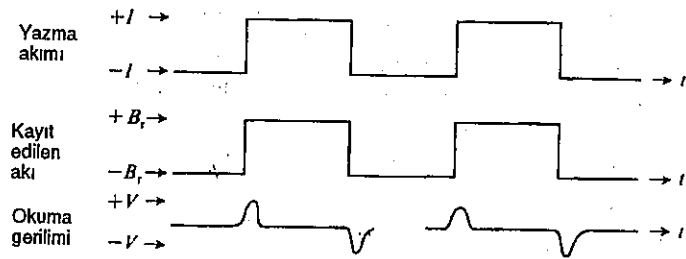
Şekil 6.22 Kafadaki hava aralığı

değiştirilirse alan tersine döner ve kaplama tabakasındaki manyetik tanecikler ters yönde manyetize olur.

Okuma kafası esasen yazma kafasıyla aynıdır (kimi zaman aynı kafa hem okuma hem de yazma kafası olarak da işlev görür). Manyetize olmuş malzeme okuma kafasındaki aralığının önünden geçtiğinde kafada manyetik bir akı indüklenir. Gerilim, manyetik akının mutlak değeriyle değil, akının değişim hızıyla orantılıdır. Şekil 6.23'te yazma ve okumayla ilgili dalga biçimleri görülmektedir. Okuma kafasından gelen gerilim ;

$$V = K \frac{d\Phi}{dt}$$

ile verilir.  $K$ , sistemin fiziksel parametrelerine bağlı olan bir sabit,  $\Phi$  de bant tarafından üretilen alandır. Sabitin türevinin sıfır olduğuna dikkat edin. Böylece sadece manyetik akı geçişleri algılanabilmektedir. Bantın sabit mıknatıslanmış bir bölgesinden alınan çıkış sıfırdır. Birazdan görüleceği gibi bu, sayısal veriyi teyp veya diskin üstüne doğrudan kaydetmeyi zorlaştırmaktadır.



Şekil 6.23 Okuma/yazma dalga biçimleri

### Veri kodlama teknikleri

Manyetik bir ortama bilgi kaydetmeye ilişkin temel işlemi açıkladıktan sonra, sayısal verinin kayıt edilmeden önce kodlama biçimlerinden birkaçına göz atacağız. Tüm manyetik saklama birimleri veriyi, her seferinde bir bit şeklinde, yazma kafasının altındaki manyetik ortamın hareketiyle tanımlanan yol boyunca seri olarak kaydeder. Teyp birimleri, bant üzerinde enine bir grup paralel izi aynı anda kayıt eder. Öte yandan her bir iz boyunca kayıt edilen veriler, seri olarak kayıt edilmiştir.

Yazılacak bir ve sıfırları yazma kafasına doğrudan göndermek mümkün değildir. Önce veri desenlerini uygun bir biçime dönüştürmek için kodlama veya modülasyon denen bir işlemin kullanılması gerekir. Benzer biçimde bilgiler banttı geriye okunduğunda da, orijinal sayısal veriyi çıkarmak için kodu çözülmesi veya demodüle edilmelidir. Uygulamada seçilen kod çözme işlemi; sistemin güvenilirliğini korurken, verilen bir yüzey alanına mümkün olduğu kadar çok biti yüksek yoğunlukta sığdırabilme isteği ile sistemin karmaşıklığını makul sınırlar içinde tutma arasındaki uzlaşma ile belirlenir.

Çeşitli kodlama teknikleri karşılaştırılırken, tüm kayıt teknikleriyle ilgili bir grup özelliğe dayanarak yapılır. İlginçtir ki bunlar sayısal veriyi uzun mesafelere göndermek için sistem seçimindeki kriterlerle kabaca aynıdır. Bu anlamda 'Akı terslenmesi' terimi çok sık kullanılır. Terim, teyp veya disk tabakasında kayıt edilmiş manyetik alandaki değişmeyi belirtir. Yazma kafasındaki akımın yönünü ters çevirmek basitçe bir akı terslemesine neden olur.

1. **Verimlilik** : Herhangi bir kodu saklama verimliliği, akı terslemesiyle saklanan bit sayısı olarak tanımlanır ve yüzde oranıyla ifade edilir. En büyük değer %100'dür ve her akı terslemesinde bir bite karşılık gelir.

2. **Sembollerarası korelasyon** : Saklanacak veriyi gösteren semboller, birbirinden olabildiğince farklı olmalıdır. Bu yolla, kayıt etme/okuma işlemindeki hatalardan dolayı semboller anlaşılmasız biçimde değişime uğramış olsalar bile, bunların ayırt edilmesi kolaylaşmış olacaktır. İki değerli sayısal bir sistemde semboller aynı fakat ters işaretli (yani terslenmiş) olmalıdır. Bu %100 korelasyon olarak tanımlanır.

3. **Bant Genişliği** : Bir sinyalin kaplamış olduğu bantgenişliği, sinyalin değişim hızının bir ölçüsüdür. Matematik teorilerine göre, herhangi bir dalgabıçımı,  $f$ ,  $2f$ ,  $3f$ , ... frekanslarındaki sonsuz sinüs dalga dizileri şeklinde ifade edilebilir. Bir sinyalin kapladığı bantgenişliği, sinyalin yayıldığı frekansların aralığıdır. Bant

genişliği, eski saniyedeki sayıl sayısına karşılık gelen ve Hertz (Hz) birimiyle ölçülür. Örneğin telefonun bant genişliği 300 Hz ile 3000 Hz arasındadır. Sınırların altında veya üstünde yer alan frekanslar iletilmezler. İnsan sesi, büyük bir bant genişliğine sahip olsa da telefon hatlarında, düşük ve yüksek frekanslar atıldığı için bozulur. Bunun sonucu olarak telefon konuşmasının kalitesi daha düşüktür. Mühendislik açısından çok düşük ve çok yüksek frekansların kullanılması oldukça zordur. Bant genişliği düşük olan kodlar, bant genişliği büyük olan kodlar tercih edilirler. Özellikle DC'ye (doğru akım veya sıfır frekans) yaklaşan düşük frekanslardan kaçınılmalıdır.

4. **Kendinden Saatleme** : Son aşamada, kodlanmış verinin kodu çözümlenmeli ve tek bitlerine ayrılmalıdır. Bitleri birbirinden ayırma metodu sağlayan bir kod kendinden saatleme denir ve oldukça gereklidir.

5. **Karmaşıklık** : Kodlama ve kod çözme işlemleri ne kadar basit olursa o kadar ucuza mal olur. Kayıt etme ve okuma işlemleri zamanla değişen analog sinyallerle ilgili olduğundan kesinliği ve toleransı, maliyeti aşırı yükseltecek oranda büyük olmamalıdır. Sayısal kayıt işlemindeki sinyaller ismen sayısal (yani iki durumlu) olsalar da pratikte bir teyp veya diskten okunan sinyal analog bir sinyalin tüm özelliklerine sahiptir.

6. **Gürültü Bağışıklığı** : İdeal bir kod, dışarıdan gelen yabancı sinyallere karşı geniş bağışıklığa sahip olmalıdır. Manyetik kayıt sistemlerinde gürültüye, içeri veya dışarı kaçmanın yol açtığı manyetik tabakadaki kusurlar neden olur. Dışarı kaçmak eksik manyetik malzemenin neden olduğu sinyal kaybıdır. İçeri kaçmak ise gürültü-darbesidir. Bir başka gürültü kaynağı ise bitişik izlerden toplanan bir sinyal olan diyafoni (cross-talk) 'dir.

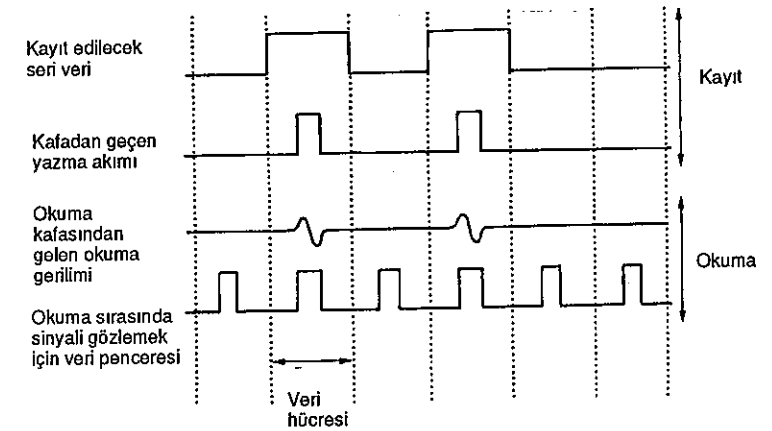
Aşağıda bir grup olası kodlama/kod çözme tekniğinin özellikleri sunulmuştur. Liste ayrıntılı değildir fakat teyp birimleri ve disk sürücülerinde kullanılan teknikleri içerir.

### Sıfıra Dönüştürme

En sade biçimiyle sıfıra dönüştürme (RZ) kayıt, yüzeyin mantıksal sıfır için manyetize edilmemesi ve mantıksal bir için de küçük bir darbe ile manyetize edilmesidir. Sıfır süresince yazma kafasına hiçbir sinyal uygulanmadığından, halihazırda bant üzerinde kayıtlı bir mantıksal 1 silinemez veya üzerine yazılamaz. RZ kayıt tekniğinin değiştirilmiş bir biçimi, öndeğere-dönüştürme (Return-to-bias:BS) kayıt tekniğidir.

Burada kayıt, ters kutuplamanın sonucu kısa bir darbeyle oluşan mantıksal 0, manyetik tabakayı belli bir yönde, mantıksal 1 de bunu tersi yönünde doyuma götürerek gerçekleştir.

Şekil 6.24'te, RB okuması görülmektedir. Uygulamada kullanılan darbe genişliği kafanın ve bandın özelliklerine bağlıdır. Geniş bir darbe, kayıt edilmiş verinin maksimum sıkıştırma yoğunluğunu azaltırken, çok dar bir darbenin algılanması da çok güç olabilir.



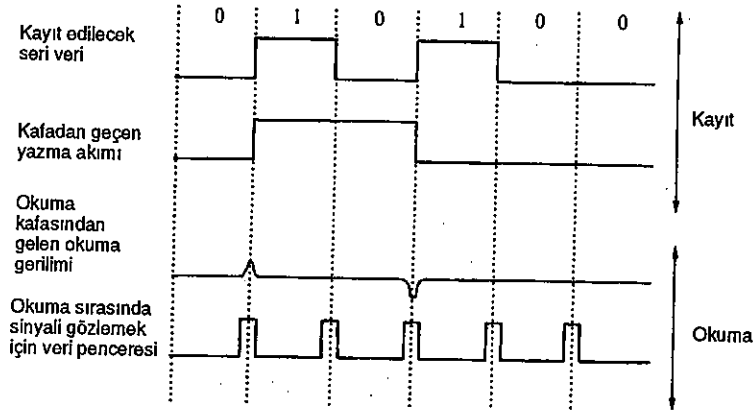
Şekil 6.24 Öndeğere-dönüştürme kayıt tekniği

Teypten bilgiyi okumak için bir veri penceresi (yani teypten gelen çıktının örneklemesi zamanı) vardır ve okuma kafasının çıkışı, bu pencereden incelenir. Ne var ki, bir 0 dizisi banttan hiçbir çıktı üretmez ve veri penceresinin, veri hücresini tam olarak ortaladığını tümüyle sağlamanın basit bir yolu yoktur. Bu nedenle öndeğere-dönüşün kendinden saatlemeli olmadığı söylenir. RB kayıt türünün verimliliği %50'dir, korelasyonu iyi, fakat gürültü duyarlılığı zayıftır. Banttan gelen sinyali kullanmak için düşük frekanslara karşılık verilebilmesi gerekir. Tüm bu nedenlerden dolayı RB kayıt etme türü yaygın değildir ve nadiren kullanılır.

### Sıfıra-dönüştürme

Çok yaygın şekilde kullanılan kodlama tekniklerinden biri, değiştirilmiş sıfıra-dönüştürme veya NRZ1 diye adlandırılır. Bir mantıksal 1'in kayıt edileceği her seferde kafanın içinde akan akım tersine çevrilir. Veriyi okurken manyetik akıdaki her değişim, bir mantıksal 1 olarak yorumlanır. Şekil 6.25'te NRZ1 kayıt tekniği

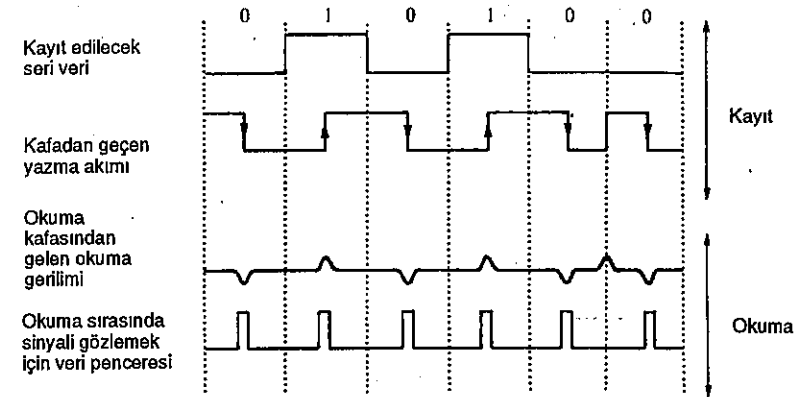
gösterilmiştir. Burada, saklı verinin bir bitine karşılık maksimum bir akı geçişi gerekir ve bu da %100'lük optimum sıkıştırma yoğunluğunu gösterir. NRZ1'in korelasyonu zayıftır, düşük bant genişliği gerektirir ve gürültüye karşı iyi bir duyarlılığı vardır. NRZ1'in en büyük dezavantajı kendinden saatlemeli olmamasıdır.



Şekil 6.25 Sıfıra-dönüşsüz 1 (NRZ1) kayıt tekniği

### Faz kodlama

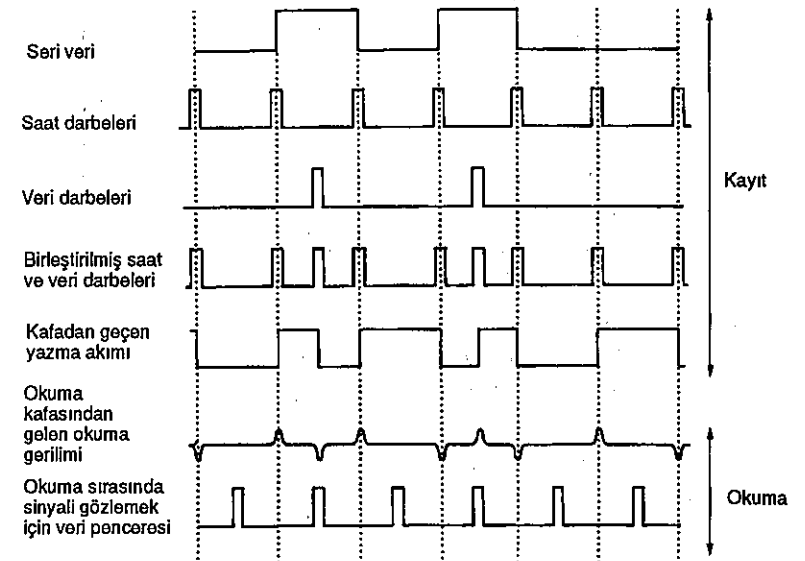
Yüksek yoğunlukta manyetik teyp mekanizmalarının büyük çoğunluğu, verileri kaydetmek için faz kodlama veya Manchester kodlaması kullanır. Her bir hücredeki bitin merkezinde bir akı geçişi vardır: alçaktan yükseğe geçiş, bir mantıksal 1'i, yüksekten alçağa geçiş ise bir mantıksal 0'ı gösterir. Her bir veri hücrelerinin merkezinde daima bir geçiş olduğundan, kayıt edilmiş veriden bir saat sinyali üretilebilir ve bu nedenle de bu kodlama tekniği kendinden saatlemelidir. Sıra ile birbirini izleyen sıfır ve birlerin akışı bit başına bir akı geçişi gerektirirken, sıfır ve birler, bit başına iki akı değişikliğine ihtiyaç duyar. Şekil 6.26'da faz kodlama tekniği gösterilmiştir. Faz kodlamanın verimliliği düşüktür (%50) çünkü bit başına maksimum iki geçiş gerekir. İlişki %100'dür. Çünkü birlerle sıfırlar arasında maksimum bir fark vardır. Bant genişliği gerekleri iyidir. Çünkü kaydedilmiş sinyalde düşük frekanslı bileşenler yoktur. Öte yandan bit başına iki akı geçişi olduğundan; eşdeğer bir bit yoğunluğunda, kayıt edilmiş maksimum frekans NRZ1'inin iki katıdır. Uygun kodlayıcı/kod çözücüler tek yongalarda mevcut olsa da devre karmaşıklığı NRZ1'inkinden daha büyüktür. Bu niteliklerden dolayı faz kodlaması, sayısal veri iletim sistemlerinde yaygın olarak kullanılır.



Şekil 6.26 Faz-kodlamalı kayıt tekniği

### Frekans modülasyonu

Frekans modülasyonu (FM) disket sistemlerinde veri kodlamak için kullanılır. Faz kodlaması gibi bu kodlama tekniği de kendinden saatlemelidir. Kodlanmış dalga-biçimleri, bir saat darbesiyle her bir veri hücrelerinin kenarlarını işaretleyerek yaratılır. Ardından mantıksal 1'i göstermek üzere hücrenin ortasına bir darbe yerleştirilir, mantıksal 0 durumu için hücre boş bırakılır. Şekil 6.27'de saat darbe-



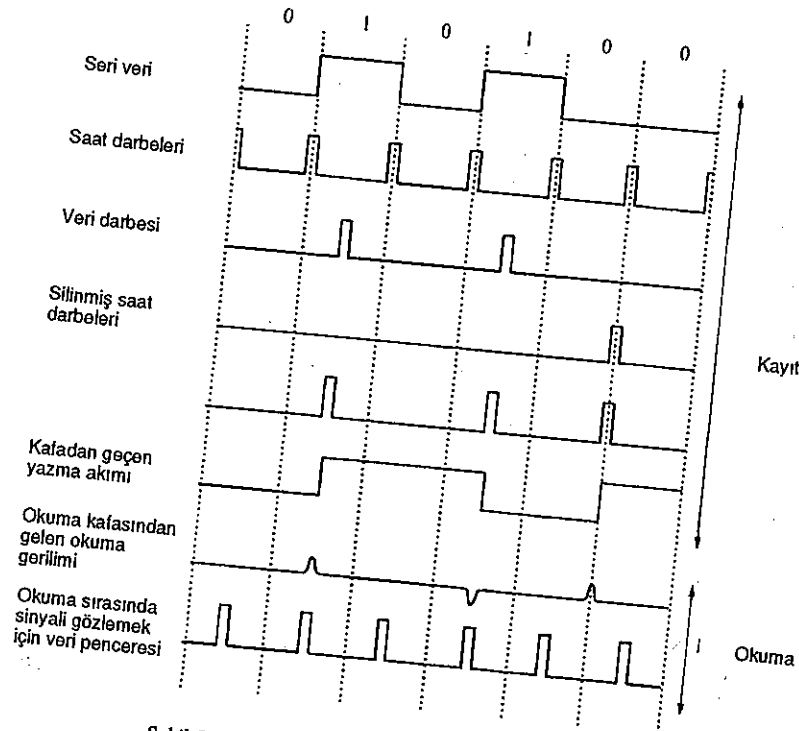
Şekil 6.27 Frekans modülasyonu

leri, veri darbeleri ve saat ile veri darbelerinin birleştirilmiş dalga biçimleri görülmektedir. Sonra bu dalgabıçımı, bilgileri, her bir darbede, yazma kafasındaki akımın (ve dolayısıyla da akımın) yönünü ters çevirerek kayıt etmek için kullanılır. FM kayıt etme türü, Faz kodlama tekniğinin sahip olduğu tüm özelliklere sahiptir. Aradaki fark, sıfır veya hücrenin ortasındaki geçişin yönünden çok geçişin varlığı veya yokluğuyla kayıt yapılmasıdır. Frekans modülasyonu adı altında hatalı veriler ya da yanlış frekans modülasyonu, veri iletimini ele alırken Kısım 8.1'de açıklanacaktır.

### Değiştirilmiş frekans modülasyonu

Değiştirilmiş frekans modülasyonu (MFM) disketlere veri kayıt etmede, hızla bilinen standart olarak FM'in yerini almaktadır. Aslında FM ve MFM terimleri teknik literatür dışında çok az kullanılır. Genelde FM, tek-yoğunluklu ve MFM de çift-yoğunluklu kayıt tekniği olarak anılır, çünkü MFM, verilen bir yüzey alanına iki misli veri saklayabilmektedir. Bunun nedeni MFM'nin %100 verimli olması ve bir başına sadece tek bir geçiş gerektirmesidir.

Şekil 6.28'de MFM sinyalinin nasıl kodlandığı görülmektedir. FM'de olduğu gibi



Şekil 6.28 Değiştirilmiş frekans modülasyonu (MFM)

bir veri darbesi mantıksal 1 içeren her hücrenin merkezine yerleştirilir. FM'den farklı olarak, hücrelerin sınırlarındaki darbeler silinir fakat bir istisna vardır. İki sıfırın ardarda kayıt edildiği her seferde, bunların arasına bir saat darbesi yerleştirilir. MFM'nin ilişkisi düşük olsa da FM ve Faz kodlamaya benzer özelliklere sahiptir. Akı geçişleri arasındaki maksimum aralık;  $T$  bir veri hücresinin genişliği olmak üzere,  $2T$ 'den daha geniş olmadığı için, MFM kendinden saatlemelidir. Şimdiye kadar açıklanan kodlar içinde en karmaşık olanı bu kod olsa da; bu teknikte kodlama yapıp, kod çözmek üzere tasarlanmış bir grup yonga vardır. Bazı yongalar, FM ve MFM'yi her iki sistemin de donanımında değişiklik yapmadan verileri kaydedip okumak için kullanabilir. Bu da, disklerin bir sistemden diğerine aktarılabilmesini olanaklı kılar.

### Grup kodları

Son zamanlarda ortaya çıkan ve manyetik disk ve teyp birimlerinde kullanılan kodlama tekniklerinden biri de grup kodudur. Öteki tüm düzenlemeler her bir bite belli bir dalgabıçımı tayin etmektedir. Bu, bant genişliği, sembollerarası korelasyon ve akı yoğunluğu açısından optimum bir kodun bazı gerekleriyle uyumsuzluk göstermektedir. Grup kodları,  $m$ -bitlik bir kodu göstermek için  $n$ -bite gerek duyar. Böylece  $2^m$  sayıda olası dalgabıçımı olsa da sadece  $2^m$  tanesi geçerli veri değerlerini gösterimlemek için kullanılır. Bu, zayıf nitelikli dalgabıçımlarının koddan yararlanarak çıkarılabileceği anlamına gelir. Örneğin, aşağıdaki 4/5 grup kodu, 4-bitlik bir kodun 16 olası değerini gösterimlemek için beş bit kullanmaktadır.

#### Kodlanacak veriler

#### Kaydedilecek GCR 4/5 kodu

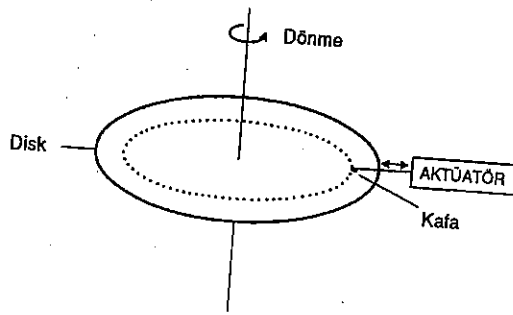
|      |       |
|------|-------|
| 0000 | 11001 |
| 0001 | 11011 |
| 0010 | 10010 |
| 0011 | 10011 |
| 0100 | 11101 |
| 0101 | 10101 |
| 0110 | 10110 |
| 0111 | 10111 |
| 1000 | 11010 |
| 1001 | 01001 |
| 1010 | 01010 |
| 1011 | 01011 |
| 1100 | 11110 |
| 1101 | 01101 |
| 1110 | 01110 |
| 1111 | 01111 |

Veri değerlerini, kayıt edilecek 5-bitlik koda eşleyen algoritma ikiden fazla sardarda gelmesine engel olur. En azından bu özellik, kodu kendinden saatler yapmak üzere, kayıtlı bulunan üç bit başına bir akı değişikliğini garanti eder.

#### 6.4.2 Disk sürücüler

Diskler; ince manyetik malzemeden bir tabakayla kaplı düz, yuvarlak ve sert alüminyum plakalardır. Disk, kendi eksenini etrafında plağın pikapta dönüşü benzer bir şekilde döner. Bilgiler, disk üzerindeki eş merkezli izler üzerine saklanır ve okuma/yazma kafası, o esnada kendisine yazılan veya kendisinde okunan izin üstüne konumlandırılır. Burada yalnızca hareketli-kafalı diskleri ele alacağız. Disk kafası, pikabın kolu gibi, bir izden diğerine hareket eder. Şekil 6.29'da bu tip bir disk sürücünün iç yapısı görülmektedir. Müzik plakalarıyla manyetik diskler arasındaki en büyük fark şudur: Müzik plağında yivler, fiziksel olarak yüzeyinin içine oyulur. Manyetik bir diskteki izler basitçe diskin okuma/yazma kafası altındaki hareketinin çizdiği dairesel yollardır. Akım geçtikçe, kafa iz boyunca verileri yazar. Benzer biçimde, veriyi okurken de istenen ize gider ve mıknatıslanmış yüzeyin dönüşüyle, okuma kafasının sargılarında küçük bir gerilim indüklenir.

Hareketlendirici (aktüatör) denen hassas bir servo-mekanizma kafayı yatay olarak izden ize hareket ettirir. Manyetik disk ile plak arasındaki farkı anımsayın: ilkinde izler eş merkezlidir ve kafanın izden ize adımlaması gerekir, ikincisinde ise spiral bir yiv, diskin iç yüzeyinde oyulur ve iğne, disk döndükçe aşama aşama merkeze doğru hareket eder.

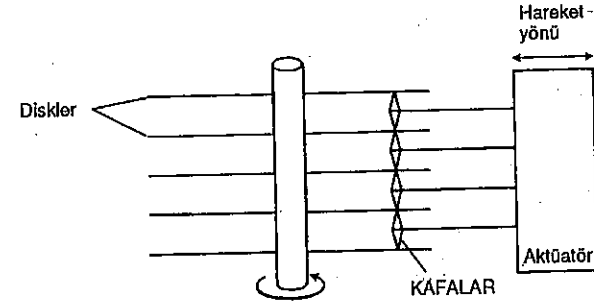


Şekil 6.29 Manyetik disk

Disk sürücülerin özellikleri üreticiden üreticiye değişir ve sürekli olarak geliştirilmektedir. Tipik bir disk sürücüsü inç başına 200 ize sahip olup, veriyi inç başına 2200

bit yoğunluğunda saklar ve 2400 rpm'de döner. Disk sürücülerin saklama kapasiteleri megabayt olarak ifade edilir ve 2 ile 1000 megabayt arasında değişir.

Disk sürücüler çok pahalı parçalardır ve sistem kuruluşlarında maliyetleri MİB ve Doğrudan erişimli belleklerini aşar. Bir disk sürücünün maliyeti, karmaşıklık düzeyi ve mekanik yapısının hassaslığı ile belirlenir. Bu nedenle üreticiler disk sürücülerin etkin maliyetini, diskleri üst üste sıralayarak ve birden çok kafa kullanarak düşürürler. Bu türde bir düzenleme, Şekil 6.30'da gösterilmiştir.



Şekil 6.30 Çokdiskli bir disk sürücü

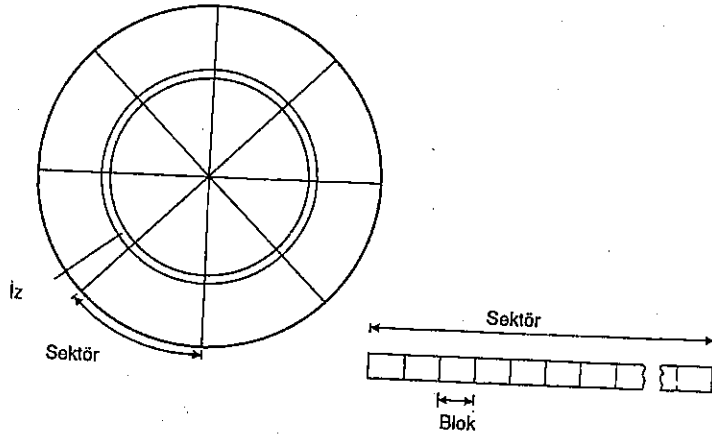
Burada, tipik olarak 22 yüzeyi olan 11 disk vardır. Bunlardan 20 tanesi, veri saklamak için kullanılır (en alt ve en üst yüzeyler kullanılmaz). 20 kafa birlikte hareket eder ve ortak bir hareketlendirici ile sürülür. Kafaların hareketi bir silindiri tanımlar. Yani her bir yüzeydeki ilgili izler topluca bir silindir olarak bilinir.

Kafaların kendileri oldukça karmaşıktır. Sadece doğru elektrik ve manyetik özellikleri değil (okuma/yazma kafasındaki hava aralığı sadece 50 mikrometre genişliğinde olabilir) fakat aynı zamanda doğru mekanik özelliklere de sahip olmalıdır. Eğer kafa gerçekte disk yüzeyiyle fiziksel olarak temas halinde olsaydı, diskin yüzeyi üzerindeki geçiş hızı 10 km/saat olduğundan aşındırıcı manyetik tabaka kısa sürede yıpranmasına neden olurdu. Kafalar, 'kızak' denilen ve diskin üstünde, yüzeyden 0.5-2.5 mikron kadar yukarıda yerleştirilen bir tutucuya takılır. Gereken hassaslık düzeyi, bu türde mevcut teknikler kullanılarak yapılan bir düzenlemeyle mekanik olarak elde edilemez. Ne var ki, kızaklara uçak kanadının (aerofoil) sahip olduğu bazı özellikler eklenirse kızaklar, disk yüzeyinin hemen üzerinde, hareketli havanın oluşturduğu aralıkta kayacaktır. Nesneler hareket ettiklerinde yüzeylerindeki hava tabakası da (sınır katmanı) onunla birlikte hareket eder. Hava yüzeyinden belli bir mesafedeki hava kütlesi ise hareketsizdir. Sonuç olarak, durgun hava

kütlesi ile diskin yüzeyi arasında bir hız farklılığı mevcuttur. Disk yüzeyini üzerindeki belli bir noktada, kafanın üzerinden akan havanın hızı, kafayı disk doğru iten yayın karşı-kuvvetini karşılamaya yetecek oranda bir kaldırma kuvveti üretir. Bu noktada kafa dengededir ve disk yüzeyinin hemen üzerinde kayar. Modern kızaklar, 0.5 mikronun altında bir mesafede kayarlar ve içinde, kaldırma kuvvetinin bir bölümünü azaltmak için uzunlamasına yivler bulunur. Modern kızaklarda hassaslık, o denli fazladır ki, dikkatsiz kullanma sonucu parmak ucundaki asit bile dış aerodinamik hatlarını bozabilir.

Kafa bazen disk yüzeyine temas edebilir ve buna çarpma denir. Bu çoğu zaman izlerin bir kısmına zarar verir. Zarar görmüş bu iz 'bozuk' olarak etiketlenmelidir ve eğer verilerin bir yedeği varsa, bu kopyadan geriye tekrar yazılabilir.

İzler sektör denilen birimlere, sektörler de veri bloklarına bölünürler (Şekil 6.31).



Şekil 6.31 Disk üzerindeki verilerin formatı

Normalde disk denetleyicisi (yani bir disk sürücüsünün çalışmasını denetleyen elektronik sistem), iz ve sektörleri belirterek, ya içerdği değeri tampona (yani geçici bir bellek alanına) ya da tamponun içerdği değeri de diske yazar. Bazıları disk sürücüler, -okuma yapmadan önce, istenen sektör kafanın altına gelinceye kadar beklenmesi gereğinden dolayı- sıralı erişimli birimler olarak adlandırılırsa da, bu özellikleri nedeniyle yani istenen bir ize gidebilme yeteneği açısından rastgele erişimli aygıtlar olarak ele alınırlar. Teyp birimleri ise, yavaşlığından dolayı hiçbir zaman rastgele erişimli bir birim sınıfına sokulmazlar.

Bir disk sürücü kullanıcısının en çok ilgilendiği üç parametre genelde şunlardır: Sistemin toplam kapasitesi, verinin diske yazılma veya diskten okunma hızı ve ortalama erişim süresi. Tipik saklama kapasiteleri 5 ile 200 megabayt arasındadır. Veri hızları genellikle saniyede birkaç megabayt ve ortalama erişim süreleri de 20 msn ile 60 msn arasında değişir. Ortalama erişim süresi, istenen ize gitmek için gereken süre (arama süresi), okunacak sektörün kafanın altına gelmesini sağlamak üzere diskin dönüşü için gereken süre (gecikme) ve verinin okunma süresi olmak üzere üç bölümden meydana gelir. Pratikte okuma süresi, çoğu kez belirtilen erişim sürelerinin dışında tutulur.

İzden ize gitmek için geçen ortalama süreyi belirleme biraz zordur çünkü kafa sürekli olarak sabit hızda hareket etmez ve kafanın oturma süresi gibi faktörlerin de hesaba katılması gerekir. Kafanın oturma süresi, yüklendikten (diskin yüzeyine itildikten) sonra kafanın yüzeye çarpmaması için durmasına kadar geçen süredir ve (daha sonra görüleceği gibi) disket sürücülere ilişkindir. Bundan başka erişim başına düşen ortalama adım sayısı verinin düzenlenme biçimine ve ardışık erişimler arasında ne olup bittiğine çok bağlıdır. Eğer kafa, diskin dış çevresinde park edilmişse, izlerin merkezinde park edilmiş olma durumuna göre daha fazla hareket etmelidir. Eğer herhangi başka bir bilgi yoksa, ortalama adımlama süresi kabaca, iz sayısının yarısıyla bir izden komşu diğer bir ize gitme süresinin çarpımına eşittir. Verilen bir sektörü belli bir kafanın altına götürmek için geçen ortalama süreyi (dönme gecikmesini) hesap etmek daha kolaydır. Kabaca, bir dönüş periyodunun yarısıdır. 3600 rpm hızında dönen bir disk için bu değer:

$$\frac{1}{2} \times \frac{60}{3600} = \frac{1}{120} \text{ sn} \\ = 8.33 \text{ msn. 'dir.}$$

Disk çalışmasına ilişkin çözümlü bir örnek, bu bölümde ileride verilecektir.

#### Disk Sürücü Terminolojisinin Açıklamalı Kısa Bir Sözlüğü

Disk sürücülerine ilişkin değişik konuları ayrıntılı olarak incelemek, giriş niteliğindeki bu yazının kapsamı dışında kalmaktadır. Bu nedenle de aşağıda verilen açıklamalı sözlük okura yardımcı olmak amacıyla sunulmuştur.

**Hareketli-Kafalı Diskler :** Tek bir okuma/yazma kafası diskin yüzeyini bir izden diğerine adımlayarak tarar. Bu, halihazırda incelenmiş olan disk sürücü tipidir ve aşağıda açıklanan sabit-kafalı disk sürücü ile bir zıtlık oluşturur.

**Sabit Kafalı Diskler :** Her bir iz için ayrı bir okuma/yazma kafası bulunur. Kafalar izler üzerinde sabit bir şekilde yerleştirilir ve birim, normalde tozdan uzak tutulmak için bir asal gazın basıncı altında tutulur. Bu tür bir düzenleme çok düşük erişim süresinin gerekli olduğu uygulamalarda kullanılır, çünkü erişim süresi doğal olarak, dönme gecikmesinin periyoduna eşittir.

**Değiştirilebilir Disk Paketi :** Bazı hareketli-kafalı disk sürücülerinde kapalıdır ve disklerin kendileri dışarı alınamaz. Bazılarında değiştirilebilir disk paketi vardır ve böylece diskler çıkarılıp diğerleriyle değiştirilebilir.

**Fiziksel-bölümlenmiş Diskler :** Fiziksel-bölümlenmiş disklerde bir izi oluşturan sektörlerin başlangıcı, fiziksel yöntemlerle belirlenir. Tipik olarak bunlar diskler optik olarak algılanabilen delik veya yarık biçimindedirler.

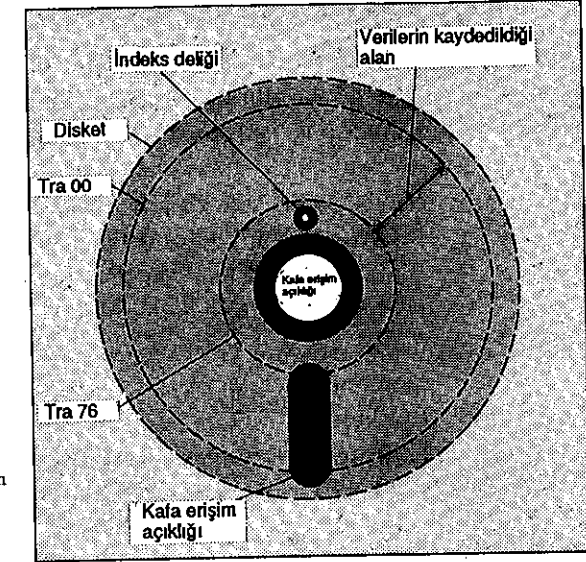
**Mantıksal-bölümlenmiş Diskler :** Mantıksal-bölümlenmiş bir diskin üzerine, herhangi bir bilgi yazılmadan önce formatlanır. Bu, her bir sektörün iz ve sektör numarasının diskin üzerine yazılması demektir. Mantıksal-bölümlenmiş diskler daha büyük esneklik sağlarlar ve yazılımın denetimi altında değişik sektör büyüklükleri yaratılabilir. Mantıksal-bölümlenmiş diskler daha verimlidir çünkü bir sektörün başlangıcı, etiketlemeyle belirtildiğinden yer kaybı olmaz. Aşağıdaki kısımda yer alan Şekil 6.34'de, mantıksal-bölümlenmiş bir disket örneği verilmiştir.

#### Disket (floppy-disk) Sürücüler

Disket sürücülerle daha önce açıklanan sabit-disk sürücüler arasındaki ilişki, bir mikrodalga'nın bir jumbo jetle olan ilişkisine benzer. Burada anlatılmak istenen şey, küçümseme dolu bir karşılaştırma değildir. Çünkü disketler, mikrobilgisayarlar için makul bir fiyatla dosya yaratma, üzerlerinde işlem yapma, saklama ve geri alma olanağı sağlarlar.

Disket sürücüler, IBM tarafından 1960'ların sonunda klasik disk sürücüyeye düşük-maliyetli bir alternatif olarak geliştirilmiştir. Standart disketler, manyetik bir malzeme ile kaplı plastikten üretilir ve 8 inç'lik koruyucu bir zarfın içine konurlar. Disketler, taşınabilir veri saklama ortamlarıdır ve bunlarla, bir sistemden diğerine veriler taşınabilir. Şekil 6.32'de, merkezi göbek aralığı, kafanın diske erişimini sağlayan bir açıklık ve bir indeks deliği olan disket görülmektedir. Disketlere, floppy disk adının verilme nedeni, üretilmiş oldukları plastik malzemenin çok ince, esnek ve eğilebilir olmasındandır. Disketler kolayca bükülebilir ve (bu nedenle de) zarar görebilirler. Bu yüzden çok dikkatli kullanılmalıdır.

■ Koruyucu zarfı kaplanmış manyetik ortam  
■ Koruyucu zarf üzerindeki açıklıklardan görülen manyetik ortam



Şekil 6.32 Disket

Disket sürücüler, hareket etmeyecek biçimde takıldıklarından ağızları, monte edildikleri cihazın ön tarafında bulunur. Bu, büyük boyutlu ve bilgisayardan ayrı olarak bulunan klasik sabit diskle bir çelişki teşkil eder. Disket sürücüler normalde, bilgisayar üzerinde yer alırlar. Disketin sokulduğu ağız, disketi içeri sokmak için açılan ve kilitlenmek üzere kapanan bir kapak tarafından kapatılır. Disketler, yüzeyine karşı basınç uygulayan ikinci bir yay-yüklemeli göbek aracılığıyla yerinde tutulurlar.

Disketler, 360 rpm'de veya bir sabit disk sürücüsünün onda biri hızında dönerler. Bu düşük hız, sürme mekanizmasını basitleştirir ve koruyucu zarfındaki disketin sürtünme hızını azaltır fakat bunun bedeli olarak dönme gecikmesi (gecikme periyodu) 160 ms'n'e çıkar.

Bir izin başlangıcını algılayabilmek için disketin yüzeyi üzerinde küçük bir delik açılır. Deliğin, disketin bir tarafındaki ışık kaynağı, diğer tarafındaki fotoelektrik bir algılayıcı arasından her geçişte bir elektrik darbesi üretilir. Bu da disket sürücüsüne, bir iz başlangıcını bildirir. Bu deliğe, indeks deliği denir. Fiziksel-bölümlenmiş bir diskette, her bir sektörün başlangıcının bir darbe ile belirtilmesi gerektiğinden, disketin çevresi etrafında 26 delik bulunur.

Disketlerde, okuma/yazma kafasının istenen bir ize götürülmesi, kısaca açıklanacağı gibi bir adımlama mekanizmasıyla gerçekleştirilir. Kafa disketin üstüne yerleştirilir ve disket yüzeyine erişim, koruyucu zarfın üzerindeki bir açıklıktan yapılır. Bir okuma veya yazma erişimi sırasında, kafaların yüklenmesi, bir baskı yastığının

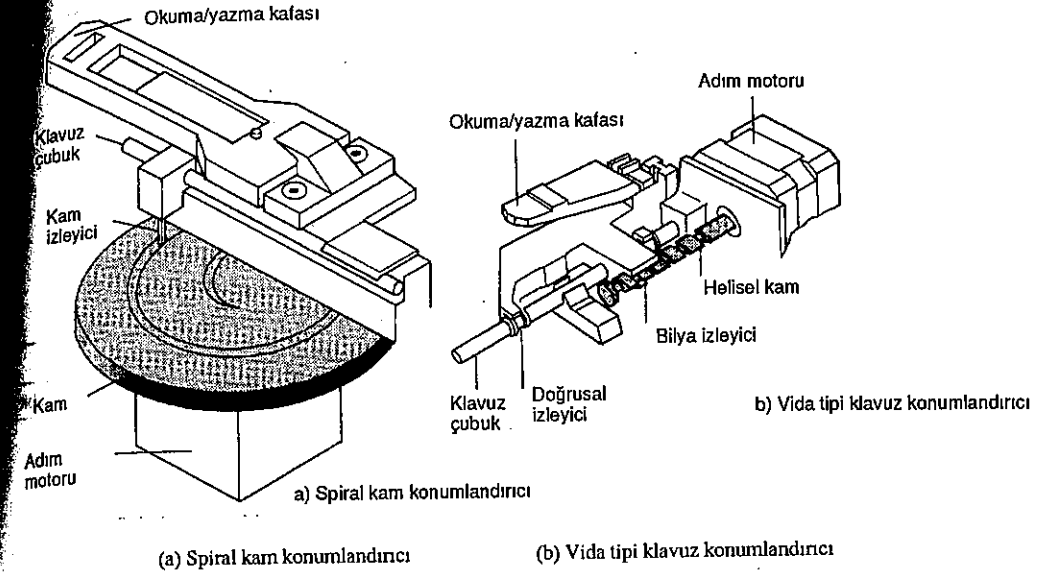
disketin arkasına teması ve disketi kafaya doğru itmesiyle gerçekleştirilir. 6.33'de, bu düzenleme gösterilmektedir. Burada kafa, diskette tam bir temas halinde olup herhangi bir kayma söz konusu değildir.

Şekil 6.33'de, kafa konumlandırma mekanizmasının nasıl çalıştığı da gösterilmektedir. Burada adım motoru ana mile bağlanmıştır. Motorun dönmesiyle bir mil üzerinde helisel biçimli yivler bulunan mil de döner. Milin üzerinde bir halka takılıdır ve kafa taşıyıcısına tutturulmuştur. Mil döndükçe halka, ona takılı bir kafa ya ileri veya geriye doğru hareket ettirir. Motor, bir adım motorudur ve diğer motorlardan farklı olarak, kesik kesik her seferinde birkaç derece olmak üzere hareket eder. Motora kısa elektrik darbeleri uygulanır ve bu da hassas aralıklarla belirlenen açı oranında adımlamasına ve bu yolla, takılı kafanın istenen ize doğru hareket etmesine neden olur.

Disket sürücüler, nispeten düşük hassaslıkta bir aygıt olduğundan ve çoğunlukla düşük-maliyetli plastik aksam kullandığından, bir izden diğerine olan mesafe, sabit disklerde bulunandan çok daha büyüktür. 8 inçlik bir diskette her inçte 48 iz olacak şekilde 77 iz bulunur. İzler, 26 sektöre bölünmüştür ve standart bir 8 inçlik diskette toplam kapasitesi 300 Kbayttır. Bu kapasite, disket formatlandıktan sonra elde edilen kapasitedir ve kullanıcılar için kullanılabilir alanı belirtir. Burada, diskette bulunan her bir sektörün iz ve sektör numaralarını etiketlemek gibi hazırlık işlemlerini yapmak için gereken bilgileri içermez. Bu kısmın sonunda verilen çözümlü örnekte, fiziksel kapasitenin 500 Kbayta yakın olduğu görülecektir. Sabit diske oranla bu kapasite çok düşüktür. Fakat tipik bir mikrobilgisayarın kapasitesi açısından bu, özellikle disketlerin birkaç sterlin ettiği anımsanırsa, pek çok uygulamaya için yeterli olmaktadır.

Disketlerin yeni bir çeşidi de mini disketlerdir. Bu hemen hemen bilinen disketlerin küçük bir modelidir. Ancak boyutu, 5 1/4 inçkaredir, 80 - 640 Kbayt kapasitesine sahiptir ve bunları süren disket sürücüler, standart bir disket sürücü fiyatının yarısını kadardır. Standart 8-inç disketler, profesyonel donanımlarda, bu mini disketler de, özellikle kelime işlemcilerle çalışılırken geniş ölçüde kişisel mikrobilgisayarlarda bulunur.

Son yıllarda hem standard hem de mini disketlerde saklanan veri miktarı büyük ölçüde artmıştır. Disketlere uygulanan standart veri kayıt tekniğine, tek-yoğunluklu kayıt denir ve bu teknikte frekans modülasyonu (FM) kullanılır. Birçok sistem artık kapasiteyi arttırmak için MFM veya çift-yoğunluklu tekniği kullanmaktadır. Kapasiteyi artırmanın bir başka yolu da, diskin her bir yüzü için bir tane olmak üzere iki kafa kullanmaktır. Üretimdeki titizlik arttıkça bazı disk sürücüler iz aralığını



Şekil 6.33 Kafa konumlandırma mekanizmaları

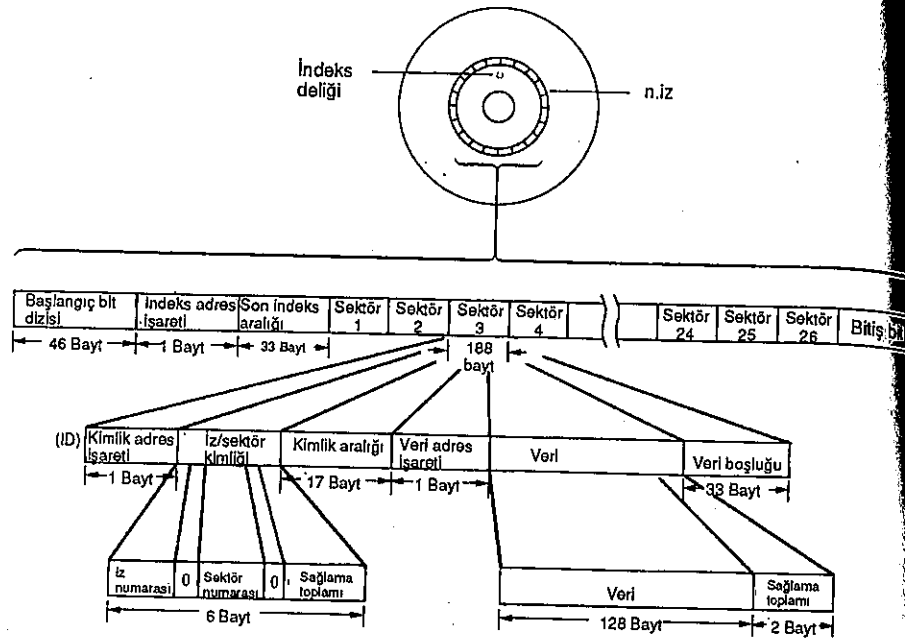
yarıya indirmiş böylece toplam kapasiteyi iki misline çıkarmışlardır. Tüm bu verilenler birleştirildiğinde kapasitede sekiz katlık bir artış elde edilmiş olur. Ne yazık ki bu, programların bilgisayarlar arasındaki aktarımını zorlaştırmıştır çünkü artık iki kayıt tekniği (tek ve çift yoğunluklu) tek taraflı veya çift taraflı disketler ve iki farklı iz yoğunluğu vardır.

#### Kayıt formatı

IBM, sadece disket sürücülerini değil aynı zamanda fiili bir kayıt standardı, IBM 3740 formatı, da sağlamıştır. Büyük yarıiletken üreticileri, bu formatı gerçekleştiren bir grup tek-yongalı disket denetleyicileri yapmıştır - Şekil 6.34'te, IBM 3740 standardına göre formatlanan bir diske verinin nasıl yazıldığı görülmektedir.

#### Winchester disk sürücüler

1980'lerin başlarında bilgisayar teknolojisinde gelişme gösteren alanlardan biri de minibilgisayar ve yüksek performanslı mikroişlemci sistemlerinde kullanılmak üzere sağlam, düşük-maliyetli sabit disk sürücüleriydi. Bu tip disk sürücülere genelde Winchester disk denir - Winchester terimi geneldir ve geniş bir küçük boyutlu disk sürücü ailesini belirtir - bir sistemi Winchester yapan özgün bir nitelik yok gibi görünmektedir. Bu terim, aslında IBM ile ilgilidir ve bazılarınca Winc-



Şekil 6.34 IBM 3740 disket formatı. Format, 'mantıksal-bölümlenmiş', yani izlere yazılan bilgilerin, her bir izdeki bilgilerin nasıl düzenleneceğini denetler. Disket üzerindeki 77 veri izinin herbiri, sektörleri oluşturmak üzere gruplanmış bulunan veri, adres ve denetim alanlarını içerir. Her bir sektörde, diğerlerinde de bulunan bir dizi alan yer alır. Bu alanlar da alt gruplara bölünerek tek tek baytları oluştururlar. İndeks deliği, bu formatta yalnızca donanımsal senkronizasyonu sağlamak için bulunur.

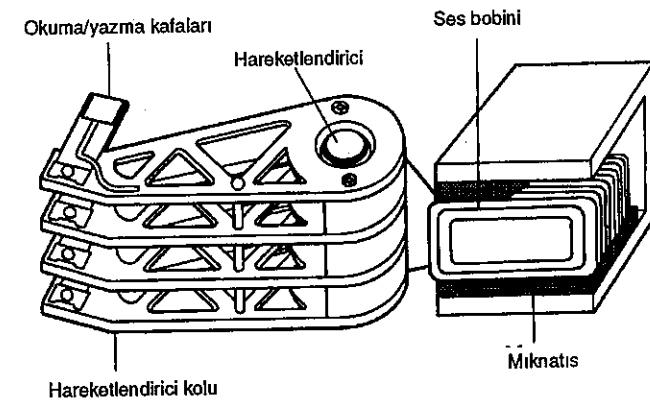
hester tüfeği ile bazılarınca da Winchester kasabası ile ilintili olduğu söylenmektedir.

Kayıt yoğunluğunun artması ve izler arası mesafenin azalmasıyla, kafanın, erişmekte olduğu izin tam üstünde kayıyor olmasını sağlayabilmek üzere çeşitli yeni yöntemler bulmak gittikçe daha da kaçınılmaz olmuştur. Bu da sürekli artan karmaşık kafa konumlandırma mekanizmalarıyla bunların ilgili elektronik devrelerinin ortaya çıkmasına yol açmıştır. Winchester teknolojisi, kafa ayarlama problemini disklerin okuma/yazma kafalarını, konumlandırıcıyı tek bir entegre birim yaparak çözmüştür. Diskler değiştirilmediklerinden, disk üzerinde bir başka birim tarafından yazılmış bir izi arayıp bulmaya ve izlemeye çalışma problemi artık söz konusu olmaz. 'Disk kafası elektromanyetik grubu', kafa ayarlaması gerektirmedikten izler arası mesafe azaltılabilir ve bu yolla saklama yoğunluğu artırılabilir.

Winchester disk sürücü, pislik ve toz girmesini engelleyen kapalı bir metalik çerçeve içindedir. Pek çok sürücü birimde, içinde iç ve dış hava basıncını eşitlemek için, altında bir hava filtresi bulunan bir delik bulunur. Diskin tozdan arındırılmış temiz bir ortamda dönmesi halinde, kafanın kayma yüksekliği azaltılır ve bu yolla kayıt yoğunluğu artar.

Klasik sabit disk sürücülerden farklı olarak birim, kullanımda değilken kafaların, disklerin en dışına çekilmesi (park) zorunlu değildir. Kafalar, yalnızca diskler dönerken kaydığında ve disk sabitken geri çekilmediğinde, disk yüzeyinin bir bölümünü iniş alanı yapmak gerekir. Bu düzenlemenin sonucu olarak şu ortaya çıkar ki; disklerin kafalarla temasta olduğu süreyi azaltabilmek için diskler, mümkün olduğunca çabuk hızlandırılabilir (ve durdurulabilir).

Tümü olmasa da bazı Winchester disk sürücüleri, klasik sabit sürücülerde bulunan doğrusal (içeri ve dışarı) konumlandırıcılardan çok, kendi eksenini etrafında dönen bir kafa konumlandırıcı kullanırlar. Şekil 6.35'te ses-bobinli aktüatörün, bir kolu, bir eksen etrafında döndürmesinin, kafa grubunun disklerin yüzeyi üzerinde nasıl bir gidip gelme hareketine neden olduğu gösterilmiştir. Ses-bobinine böyle bir ad verilmesinin nedeni bir hoparlör gibi çalıştığı içindir. Sabit bir mıknatıs tarafından üretilen güçlü bir manyetik alanın içine yerleştirilmiş bobinden akım geçirilirse, bobinden geçen akım bir manyetik alan üretecek ve bobinin sabit mıknatısa çekilmesine veya itilmesine neden olacaktır. Bu da, bir eksene bağlı bulunan kolu hareket ettirecektir. Bu, halen kullanımda olan birçok kafa grubundaki hareketlendiricilerin çalışma ilkesini göstermektedir.



Şekil 6.35 Winchester kafa-grubu konumlandırma mekanizması

Winchester teknolojisi ilk olarak 14-inç disketlerde uygulanmıştır. Sonra 8-inç disketlere genişletilmiştir ve bugün 5  $\frac{1}{4}$  inçlik Winchester'lar mevcuttur. Bir Winchester'ın fiyatı, 500 sterlinden daha azdır ve 10 megabayt kapasitesindedir. Piyasada herbiri kendine özel parametreleriyle çok geniş sayıda Winchester'da sürücüler vardır. Öte yandan tipik dönme hızı 3600 rpm., veri aktarım hızı Mbit/san., kapasitesi 10 Mbayt ve erişim süresi 30 msn'dir. Disket sürücüler sürekli artan Winchester baskısının altında geleceğine inanmak mantıkdışı değildir.

#### Çözümlü örnek

8-inçlik disket sürücüler, iki taraflı disketler kullanır ve verileri, her bir yüzünde bulunan 77 iz üzerine kaydederler. Her iz, 26 sektör içerir ve 128 veri baytı saklayabilir. Disket ise, 360 rpm.'de döner ve izden ize arama süresi 10 msn'dir. Kafa, oturma süresi 10 msn. ve kafa yükleme süresi de 200 msn'dir. Yukarıdaki bilgilerden şunları hesaplayın :

1. Disketin bayt cinsinden toplam kapasitesi
2. Ortalama dönme gecikmesi
3. Verilen bir sektörün yerinin bulunmasında ortalama süre. Burada, kafanın başlangıçta 0 izinde park edilmiş ve yüklenmemiş durumda olduğunu varsayın. Kafa istenen izin yeri bulunduktan sonra yüklenecektir.
4. Yeri bulunmuş bir sektörün okuma süresi
5. Bir sektörün okunuşu sırasında verinin diskten işlemciye taşınmasındaki ortalama hız.
6. Merkezden 3 inç uzakta bulunan bir iz etrafında diskin paketleme yoğunluğunun bit/saniye cinsinden bulunması.

#### Çözüm

1. Toplam kapasite = taraflar x izler x sektörler x sektör başına bayt sayısı  
 $= 2 \times 77 \times 26 \times 128$   
 $= 512\,512$  bayt (yaklaşık  $\frac{1}{2}$  megabayt)
2. Ortalama dönme gecikmesi = dönüş periyodunun  $\frac{1}{2}$ 'si.  
 $360 \text{ rpm saniyede } 360/60 = 6 \text{ dönüşe karşılık gelir}$

1 dönüş  $\frac{1}{6}$  saniyedir.

Böylece ortalama gecikme  $\frac{1}{12}$  saniye = 83.3 msn'dir.

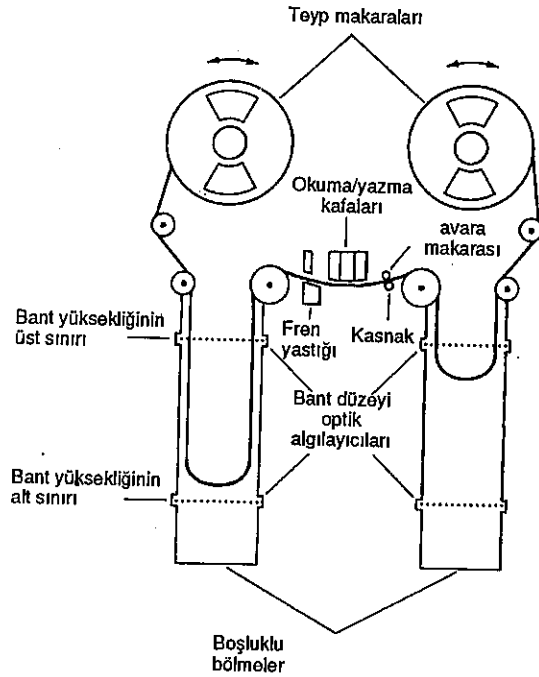
3. Bir sektörü ortalama olarak bulma süresi  
 $= \text{gecikme} + \text{kafa yükleme süresi} + \text{kafa oturma süresi} + \text{arama süresi}$   
 $= 83.3 \text{ msn} + 200 \text{ msn} + 10 \text{ msn} + 77/2 \times 10 \text{ msn}$   
 $= 678.3 \text{ msn}$
4. Bir dönüşte (1/6 saniye) kafanın altından 26 sektör geçer. Böylece bir sektörü okumak için geçen süre  $1/6 \times 1/26 = 6.41 \text{ msn.}$ 'dir.
5. Bir sektörün okunuş süresince 6.41 msn.'de 128 bayt okunur. Ortalama veri hızı okunan bit sayısı bölü geçen süredir.  
 $(128 \times 8) / 0.00641 = 159\,750 \text{ bit/saniye}$
6. Paketleme yoğunluğu = toplam bit sayısı / iz uzunluğu  
 $= 26 \times 128 \times 8 / (2 \times 3.142 \times 3)$   
 $= 1412.3 \text{ bit/inç}$

#### 6.4.3 Teyp birimleri

Bir teyp sisteminde bilgiler, manyetik bir malzemeyle kaplı ince polyester şerit bant üzerine kaydedilirler. Bant  $\frac{1}{2}$  inç genişliğindedir ve (tipik olarak) 2400 feet'lik 10.5 inç makaralarda saklanır. Sayısal veri için bir teyp sürücü evlerimizde kullandığımız yüksek kaliteli bir makaradan-makaraya teyp cihazından çok fazla farklı değildir. Sayısal bir teyp cihazının olası bir düzenlemesi Şekil 6.36'da verilmiştir.

Teyp biriminin temel işlevi bandı, sabit hızla okuma/yazma kafasının önünden geçirmektir. Belli bir veri blokunu ararken teyp nispeten daha hızlı hareket ettirilir ve blok başlangıcı bulunduğunda durdurulur. Ataletlerinden dolayı teyp makaraları bir anda durdurulamaz. Sonuç olarak, eğer basit bir teyp mekanizması kullanılsaydı bant, durdurulduğu her seferinde kopabilecek veya en azından gerilecekti. Yüksek kaliteli teyp mekanizmalarında kafaların önünden geçen bandın hareketi, durdurma ve başlatma makaralarının hareketinden ayrılırlar. Bunun nasıl gerçekleştirildiği kısaca aşağıda açıklanmıştır :

Bant, bir kasnak veya avara makarası düzeneğiyle okuma/yazma kafa(lar)nın önüne çekilir. Kasnak; hassas bir biçimde yapılmış ve parlatılmış, sabit bir hızla dönen sert bir metal silindirdir. Bant kasnağın arasından geçer ve bir avara makarası (veya kısma merdanesi) ile çekilir. Kasnak, bant ve avara makarası arasındaki sürtünme, bandın çekilmesine neden olur. Avara makarası, bandın karşı yöndeki hareketinden dolayı döner, herhangi bir mekanizma tarafından sürülmez. Bant



Şekil 6.36 Bant sürme mekanizması

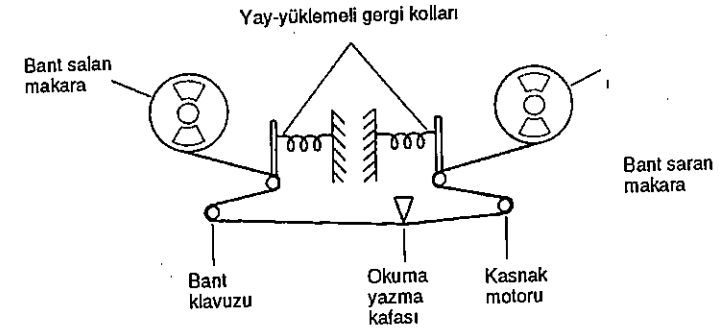
durdurulduğunda bir solenoid, avara makarasını banttan ileriye iter. Durdurmak için aynı zamanda frenleme yastıkları devreye girer.

Kasnaklarla makaralar arasındaki bant, teyp makarasının altında bulunan ve içi boşaltılmış bir bölmede asılıdır. Bölmenin altından, bölme içindeki hava emilerek bantın içeride kalan kısmı asılı halde tutulur. Boşluklu bölmelerin altına ve üstüne bantın varlığını veya yokluğunu algılamak için foto-elektrik algılayıcılar yerleştirilir. Bu algılayıcıları kullanarak, makaraları döndüren motorların hızlarını denetlemek için, boşluklu bölmedeki bantın uzunluğu aşağı yukarı sabit tutulur. Örneğin, soldaki makara bantı çok hızlı salıyorsa, bantın bölme içindeki kısmı uzar. Bantın bölme içindeki kısmı alttaki algılayıcılar hızasına ulaştığında soldaki makaranın dönüşü yavaşlatılır ve bant parçasının boyu kısaltmaya başlar.

İçi boşaltılmış bir bölme kullanmanın amacı, bantın kafaların önünden geçmesiyle ani olarak durdurulabilmesine imkan tanımadır. Avara makarası kapstandan kurtulduğunda bant frenleri devreye girer ve makaraları süren motorlar durdurulur. Kafaların altındaki bant durur fakat makaralar bantı salmak veya toplamak için kısa bir süre daha dönmeye devam eder. Bu, bölmelerdeki iki bant uzantısından birinin

uzamasına diğerinin de kısaltmasına yol açar. Boşluk tamponlarının dezavantajları vardır. Çekiş gücünde ortaya çıkan bir sorun, tekrar güç uygulandığında bantın sarkmasına neden olabilir. Bu tamponlar aynı zamanda önemli miktarlarda enerji tüketirler ve hataya-açık birimler olarak kabul edilirler. Ortamdaki alçak hava basıncından dolayı uçaklarda kullanılamazlar. Bölmedeki havanın emilmesinde kullanılan sistem, havadaki tozları da emer ve maalesef, bu tozları en az nerede isteniyorsa orada, yani bantın üzerinde biriktirir. Son olarak, boşluklu bölmedeki havanın sabit biçimde ses çıkarması operatörleri rahatsız eder.

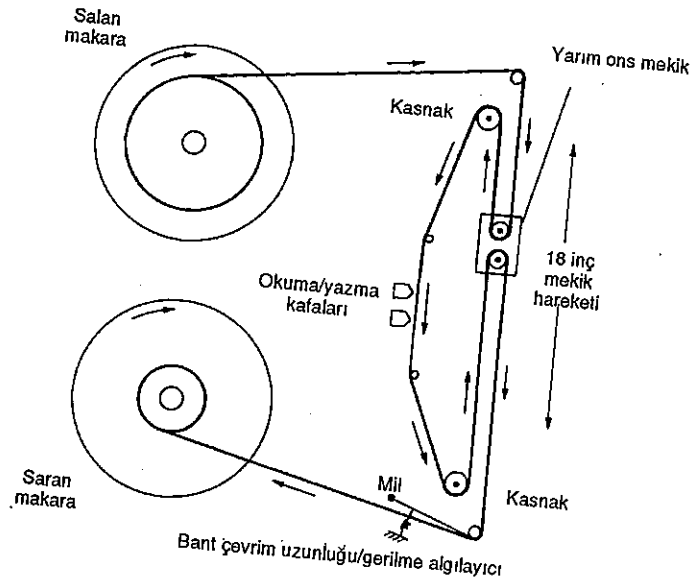
İlk teyp mekanizmaları bantı tamponlamak için - evlerimizde kullandığımız teyp deklerinde hala yaygın biçimde kullanılan bir teknik olan- gergi kolları kullanmışlardır. Şekil 6.37'de bantın fazlalık kısmını saran ve germe kuvveti arttıkça bantı salan gergi-kolu görülmektedir. Bu düzenleme, düşükten orta düzeye (saniyede 45'e) kadar doyurucu bit hız değeri ortaya koyar.



Şekil 6.37 Gergi-kolu kullanan bir bant tamponu

Gergi-kolu 'kayan-mekik' denilen bir düzenele saniyede 75'lik bir bant hızı elde edebilmek için geliştirilmiştir ve bu düzenele, Şekil 6.38'de gösterilmiştir. Gerilim kollarının birinin, bantın fazlalık kısmını alarak, diğerinin de bantı salarak uyum içinde çalıştıkları gözlenmiştir. İki bölmedeki fazlalık kısımları, serbestçe hareket eden bir mekikte birleştirilerek performansı artırmak mümkündür çünkü mekik, çok hafif (15 gr) maddelerden yapılabilir. Mekik, etrafında makaralardan bant sarıp onlara bant salmak için yönlendirilmiş iki kasnaklı küçük bir bloktan oluşmuştur. Mekik, merkezinden iki veya üç inç mesafede durabilmesi için karmaşık bir motor denetleme sistemine ihtiyaç vardır.

Teyp birimine ilişkin parametreler, teyp hızı ve kayıt formatı arasında geniş bir farklılık gösterirler. Tipik olarak hız, saniyede 0.3 ile 6.0 arasında değişir. Popüler bir kayıt formatının (dokuz-izli faz kodlanmış veri) detayları, Şekil 6.39'da verilmiştir.

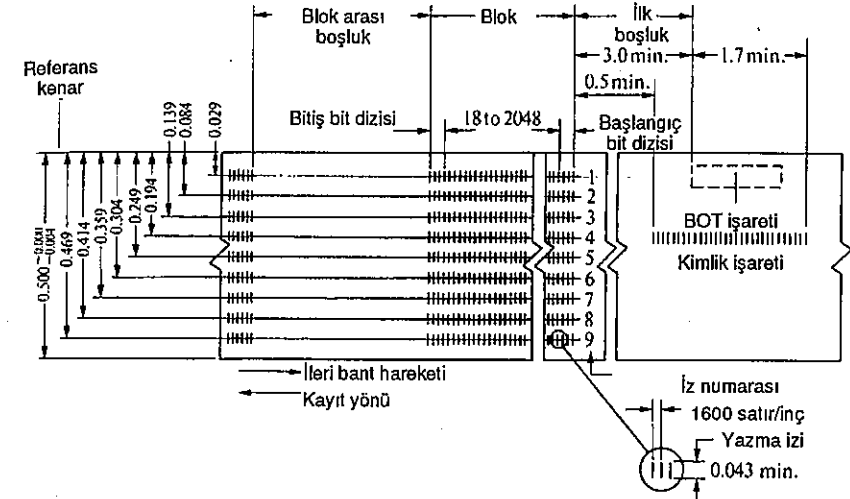


Şekil 6.38 Kayan mekik

Okuma/yazma kafalarının dokuz izi vardır. Bunlardan sekizi, her seferinde bir bayt olmak üzere enine kayıt yapar. Dokuzuncu izde, bir tek-eşlik biti bulunur. Yani eğer dokuz iz bir veri kolonunu oluşturmak üzere ele alınırsa eşlik bitinin tutulduğu izdeki bit, kolondaki birlerin toplam sayısını tek yapmak üzere seçilir. Bu, hataların algılanmasına yardımcı olur çünkü çift sayıda bit içeren bir kolon bulunursa bu, en azından bir hatayı gösterecektir.

Veriler, bloklar halinde kayıt edilir çünkü her seferinde teypleri başlatıp durdurarak bir bayt okumak imkansızdır. Blokların arasında bir okuma/yazma işleminden sonra teybe durmak için ve bir sonraki işlemden önce normal hızına erişmek için zaman tanımak üzere bloklar arasında (tipik olarak 20 mm kadar) bir boşluk bulunur. Her bloktan önce bir ön ve ardından gelen bir son bit dizisi bulunur. Bunlar bir okuma işlemi süresince elektronik devreleri senkronize etmek için kullanılırlar. Ön ve son bit dizilerini simetrik yaparak bantı her iki yönde de okumak mümkündür.

Bandın fiziksel olarak başlangıcını ve bitişini göstermek üzere uçlarında işaretler vardır. İşaretler, banda eklenmiş metalik, şeritler veya küçük delikler biçimindedir. Bant üzerinde aynı zamanda, BOT (bandın başlangıcı) ve EOT (bandın sonu) gibi yazılım işaretleri de vardır.



Şekil 6.39 Bant üzerinde kayıtlı verilerin formatı.

Dokuz-izli faz-kodlanmış format, CRC ve LRC hata denetimlerini kullanmaz. Her bir veri blokunu, 40 tane hepsi-sıfır olan bayt izler, ardından da, hepsi-bir olan baytlar (ön bayt dizisi) gelir. Ön bayt dizisi, birbirini izleyen sıfırlar ve birlerin doğru biçimde algılanabilmesi için, okuma algılama devrelerini senkronize eder. Ön ve son bayt dizilerinin simetrisi, okumanın her iki doğrultuda da okunabilmesine olanak tanır. Yükleme noktasında, (yani bandın başlangıcında), eşlik izine, inç başına 1600 değerindeki bir akı terslemesinden ibaret bir 'kimlik işareti' yazılır. Kabul edilen eşlik denetimi, tek eşlik denetimidir. Bant İşareti, 1, 2, P, 5, 7 ve 8. izlerde, inç başına 3200'lük bir akı değişiminde, 64 ila 250 değerinde bir akı terslemesinden ibarettir.

Not 1: Bandın manyetik yüzeyi, okura doğrudur, okuma/yazma kafası ise okurla aynı düzlemindedir.

Not 2: Tüm boyutlar inç olarak verilmiştir.

Not 3: İz konumlandırma toleransı,  $\pm 0.003$  inçtir.

UNIVAC 16 teyp mekanizmasının parametreleri aşağıda verilmiştir :

|                                  |        |             |
|----------------------------------|--------|-------------|
| Veri hızı (bayt/sn)              | 96.000 |             |
| Yoğunluk (bit/inç)               | 1.600  | (63 bit/mm) |
| Bant hızı (inç/sn)               | 125    | (3.2 m/sn)  |
| Bloklararası mesafe (inç)        | 0.75   | (19 mm)     |
| Bloklararası boşluk süresi (msn) | 12     |             |
| Başa sarma süresi (dak)          | 2      |             |

## Akan (streaming) teyp sürücüler

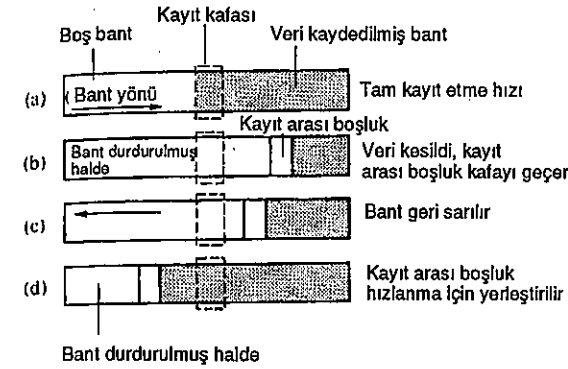
Halihazırda tartıştığımız teyp mekanizması türleri mini ve anabilgisayarlar için saklama birimleridir. Parametreleri, performans ve maliyeti anabilgisayarlar için aittir. 1970'lerin başında teyp birimlerinin, 250 bayt/mm'lik kayıt yoğunluğu, 5 m/sn hızları bulunmaktaydı ve 8 mm'lik bir kayıtlararası boşluk kullanılarak 1 ms'n içinde başlayıp durabiliyordu.

Bugün düşük-maliyetli, yüksek performanslı mikrobilgisayarda, yüksek saklama ortamları sağlayan Winchesterler kullanılmaktadır. Winchesterler, saklama ortamlarına sahip olduklarından, bilgisayarlararası program transferi için yeni yöntemler bulmak ve disk yüzeyinin bir kısmının bozulmasına yol açan sorunları durumunda Winchester'in içerdigi bilgileri koruyabilmek gerekliliği ortaya çıkmıştır. Disketler, program aktarmada uygun bir yol sunarlardı fakat yedekleme için yeterince elverişli değillerdir.

Bu problemin çözümü olarak, Winchester'ler için saklama olanakları sağlanıyorsa üzerine tasarlanmış, düşük-maliyetli bir teyp birimi olan streaming teyp sürücüsü ortaya çıkmıştır. Yüksek performanslı teyp birimlerinin maliyetini, bant hızını durdurup başlatma mekanizması belirlemektedir. Streaming teypte bundan yararlanılmıştır. Sonuçta ortaya, 8-inçlik bir disket sürücünün sığabileceği büyüklükte bir mekanizma çıkmıştır.

Streamer teyp, bantı bloklararası boşlukta durdurup başlatamaz. Şekil 6.40'da bantın, bir blokun sonunda nasıl durdurulduğu ve gelişmiş bir bant tamponlama mekanizmasının yokluğundan dolayı onu hedeften nasıl kaçırdığı gösterilmiştir. Daha sonra bant, başa sarılıp bloklararası boşluğun ötesine alınmalıdır. Böylece bant, yeniden başlatıldığında bir sonraki blok okuma/yazma kafasını geçerek hızlanacaktır. Bu işlem yaklaşık 1 saniye alır ve boşluklu-bölmeli teyp mekanizmasından bin kez daha yavaştır.

Bloklararası boşlukta durup başlayamamak, görüldüğü kadar kötü değildir. Streamer teyp, çoğu kez diskteki büyük veri bloklarını saklamak için kullanılır ve bu nedenle de bir veri blokundan diğerine atlamak yerine sürekli biçimde çalışır. Üretilmiş son streaming teyplerden bazıları durdurma-başlatma problemini 'kütsüz tampon' sağlayarak çözmüşlerdir. Artık yarıiletken bellek maliyeti çok düşük olduğundan büyük miktardaki verilerin, streaming teyp içindeki belleğe okunması mümkün olmuştur. Daha sonra da anabilgisayar, veriyi doğrudan teypten değil, yarıiletken bellekten okur. Bu şekilde, bloktan bloka neredeyse ani olarak atlamak mümkün olmaktadır çünkü veriler artık, rastgele-erişimli bellekte tutulmaktadır.



Şekil 6.40 Teyp içindeki bantın yeniden konumlandırılması.

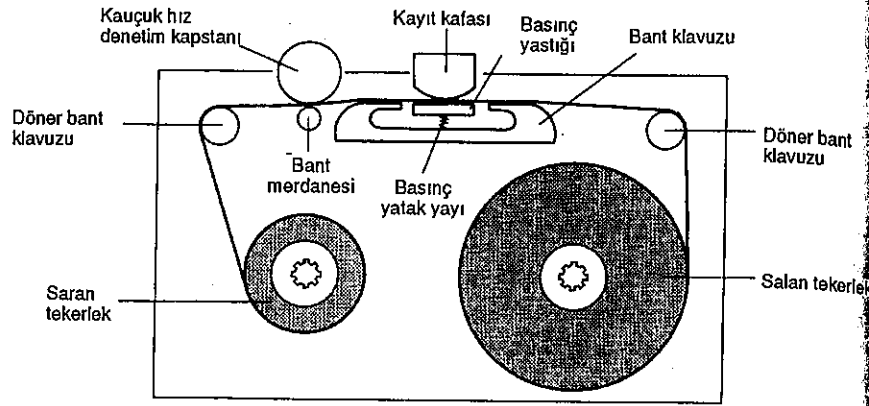
Bantın yeniden konumlandırılması, veri akışı kesilmesi halinde, akış mekanizmasının, dar bir kayıtlararası boşlukta durdurulmaması nedeniyle gerekli olmaktadır. Bant, tam hızla dönerken (a), veri akışının kesilmesi, bantın yavaşlamasına neden olur, ancak bu yavaşlama, bantın hemen kafanın altında durmasına yetecek oranda kesin değildir (b). Ardından bant, veri akışı tekrar başladığında tam hızla hızlanabilmesi için, kayıtlararası boşluk kafanın yeterince arkasına gelinceye kadar (d) geriye sarılır (c).

Bazı üreticiler evlerimizde kullanılan sıradan video cihazlarından (VCR) bir streaming teyp olarak yararlanmışlardır. VCR'ler, yalnız önemli oranda ucuz olmakla kalmayıp, aynı zamanda tek bir video kasetine 100 megabaytlık kayıt yapabilme olanağı da sağlamaktadır. VCR'ler, sayısal verileri güvenli şekilde saklamak için tasarlanmamış olduğundan tek bir veri blokunu bile güvenlik açısından altı kere veya daha fazla kayıt etmek gerekir.

## Teyp kasetleri ve kartuşlar

Sayısal kaset kayıt cihazı, manyetik teyp mekanizmasının en ucuz biçimidir ve büyüklük açısından evlerde kullanılan kasetçalarla karşılaştırılabilir. Kasetlerin kendileri çok daha yüksek standartta yapılmış ve sinyal kaybı gibi kusurlardan arındırılmış olma özellikleri dışında müzik kaydetmek için kullanılan kasetlerle aynıdır. Şekil 6.41'de evlerde kullanılan biçiminden güçlükle ayırt edilebilen bir kaset sürücünün yapısı görülmektedir. Aslında, ana fark tüm denetimlerin bazı evlerde kullanılan birimlerde bulunan tuşlar yerine solenoidler tarafından elektronik olarak yapılmasıdır.

Tipik bir kaset kayıt edici, TEAC MT-2, sayısal verileri 1600 bpi (inç başına bit) yoğunluğunda kaydederler. Kayıt sürücünün dokuz-izli makaradan-makaraya teyp



Şekil 6.41 Kaset sürücüsünde, tekerlekleri döndürmek ve bantın gerginliği korumak için iki motor kullanılır. Hız, bantın bant yuvarlayıcı ile kauçuk hız denetim kasmağı arasında çekilmesiyle denetlenir. Kasetlerin aşınma süresi sabittir, ancak kaset sürücüler en ucuz teyp sürücüsüdür. Kasetler ucuzdur ve piyasadan kolayca bulunabilir.

mekanizmasından farklı olarak verileri, bir iz boyunca seri biçimde kaydettiğine dikkat edin. Veriler faz kodlanmıştır. Aslında MT-2'de iki izli bir kafa vardır fakat izler birlikte kullanılmaz. Bunlar, sadece bir kafanın üst yüzeyine bir kafanın da alt yüzeyine erişmesini sağlayarak teybin etkin uzunluğunu ikiye çıkarırlar. Bant, okuma veya yazma modlarında saniyede 15 inç hızında hareket eder ve veriler, 24 Kbit/saniye hızla aktarılırlar. Kasetin toplam kapasitesi 760 Kbayttır. MT-2'nin önemli parametreleri Tablo 6.1'de verilmiştir.

Okuma/yazma kafasında, aynı anda okuma ve yazmayı sağlayan bir çift aralık vardır. Bu, kayıt edilmiş verinin kayıt edildikten hemen sonra tekrardan okunabilmesini sağlar ve böylece herhangi bir kayıt hatası algılanabilir. Bir hata bulunursa o anki bloğun yazımı iptal edilir, teyp geriye sarılır ve blok yeniden yazılır.

Kaset sürücünün fiyatı, 5 1/4 inç ile 8 inç disket sürücülerin fiyatları arasında bir değerdir. Kasetin ana uygulamaları, verinin bilgisayarlar arasında değiş-tokuşu, sayısal denetim ve veri kaydetmektir. Veri toplamak; bilgi toplamaya özgü bir sistemi tanımlamak için kullanılan bir terimdir. Örneğin, bir kaset mekanizmasıyla bir mikroişlemci sistemi küçük bir kapalı birime yerleştirilebilir ve trafiğin zamanlamasıyla yoğunluğunu ölçmek için kalabalık bir yolun kenarına bırakılabilir. Birkaç gün sonra bir mühendis gelir ve kutuyu açar, kasedi değiştirir ve eski kaseti genellikle bir anabilgisayarda, analiz etmek için alıp götürür.

Tablo 6.1: TEAC MT-2 Kaset biriminin Ayrıntıları

| Parametre                     | Değeri                                                                                                                                                                                             |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Veri formatı                  | 1600 b.p.i, faz kodlanmış                                                                                                                                                                          |
| Bant hızı (Okuma/yazma)       | 15 i.p.s.                                                                                                                                                                                          |
| (ileri sarma)                 | 45 i.p.s.                                                                                                                                                                                          |
| (geri sarma)                  | 74 i.p.s.                                                                                                                                                                                          |
| Başlatma mesafesi (15 i.p.s.) | 0,4 inç                                                                                                                                                                                            |
| (45 i.p.s.)                   | 2,4 inç                                                                                                                                                                                            |
| Başlatma süresi (15 i.p.s.)   | 40 msn                                                                                                                                                                                             |
| (45 i.p.s.)                   | 80 msn                                                                                                                                                                                             |
| Durdurma mesafesi (15 i.p.s.) | 0,11 inç                                                                                                                                                                                           |
| (45 i.p.s.)                   | 1,8 inç                                                                                                                                                                                            |
| Durdurma süresi (15 i.p.s.)   | 40 msn                                                                                                                                                                                             |
| (45 i.p.s.)                   | 80 msn                                                                                                                                                                                             |
| Veri aktarım hızı             | 24 Kbit/sn                                                                                                                                                                                         |
| Kayıt yoğunluğu               | 1600 b.p.i.                                                                                                                                                                                        |
| MTBF                          | 1000 saat                                                                                                                                                                                          |
| Komutlar                      | Blok yazma<br>Bant işareti yazma<br>Silme<br>Bir blok okuma<br>Bir blok atlama<br>Bir blok geriye<br>Yükleme noktası ayarlama<br>Geri sarmayı başlatma<br>Bant işareti arama<br>Yüksek hızda arama |

Kartuşlar, kasetin sadece bir üst versiyonudur. Tipik bir kartuş içinde, 1/4 inç genişliğinde manyetik bant bulunur, bir kasetten on kat daha fazla veri saklayabilir ve genelde kaset performansını her bakımdan geçer.

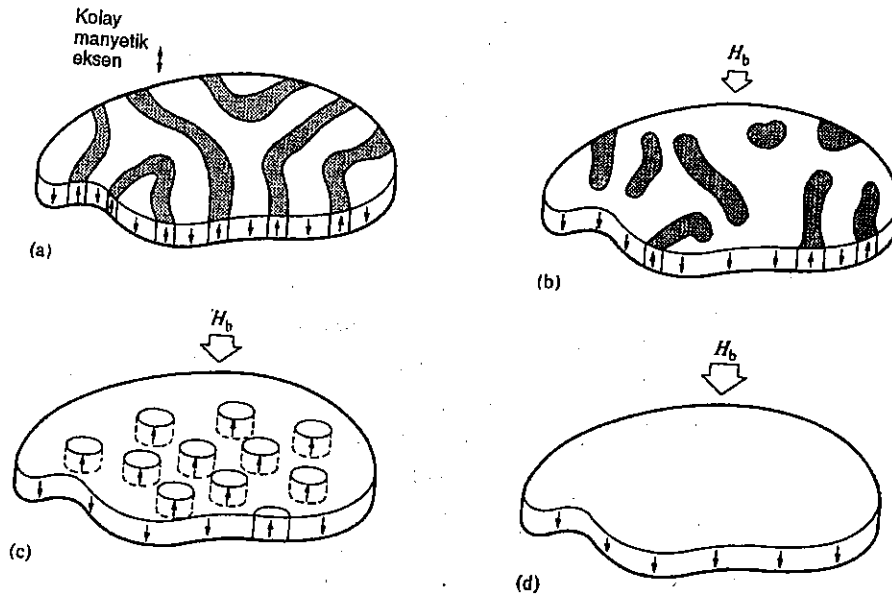
#### Ses kasetleri kayıt cihazları ve veri saklama

Neredeyse tüm ucuz maliyetli kişisel bilgisayarlar, üzerinde hiç bir değişiklik yapılmamış ev tipi bir kaset kayıt edicisinden veri okuyabilir ve veri yazabilirler. Bu kaset kayıt ediciler, sayısal dalga biçimlerinden çok konuşma ve müzik sinyallerini işlemek için tasarlanmış olduklarından, sayısal verileri bu biçimleriyle kullanamazlar. Bu zorluğun üstesinden sayısal sinyalleri iki tona (sabit frekansa) dönüştürerek gelinir. Örneğin, bir mantıksal 0, 1200 Hz'de bir ton olarak ve mantıksal 1 de, 2400 Hz'de bir ton olarak kayıt edilebilir. Ne yazık ki bu işlemin yetersizliği birçok ucuz kaset kayıt edicisinin düşük performansı ile birleşince ancak, saniyede 300 ile 2400 bit gibi bir veri hızına ulaşılabilir. Bu da kabaca pratikteki veri hızlarından saniyede 30 ile 240 baytlık olan aralığa karşılık gelir. Yukarıdaki işleme, frekans modülasyonu denir (sayısal FM ile karıştırılmamalıdır) ve Kısım 8.1'de açıklanmıştır.

## 6.5 Manyetik-kabarcıklı bellekler

Manyetik-kabarcıklı bellekler, küçük, yüksek yoğunluklu ve düşük gecikmeli seri erişimli saklama ortamıdır. Manyetik-kabarcıklı belleklerle 1960'ların sonlarında ilk karşılaştığımızda bana, geleceğin yedekleme ortamlarının bunlar olacağını söylemişti. 1980'lerin başında kabarcıklı bellekler, hala şöhretin, kapılarını çalmasını bekliyorlar. Şimdi artık ticari ürünlerde yer aldıkları ve birkaç uygulamada kullanılıyor oldukları halde bunlara hiçbir şekilde rağbet edilmemiştir. Aslında bazı yarıiletken firmaları bunların üretimini durdurmuştur. Manyetik-kabarcıklı bellekler, iki yönden baskı altındadır - disk sürücülerin maliyet ve boyutlarındaki sabit bir azalma ile birlikte, yonga başına bit sayısında da muazzam bir artış. Yarıiletken rastgele-erişimli bellek fiyatında da büyük düşüşlere neden olmaktadır.

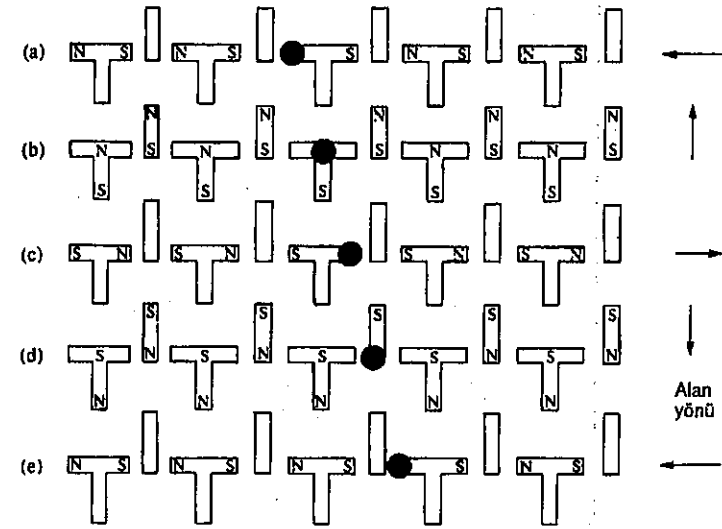
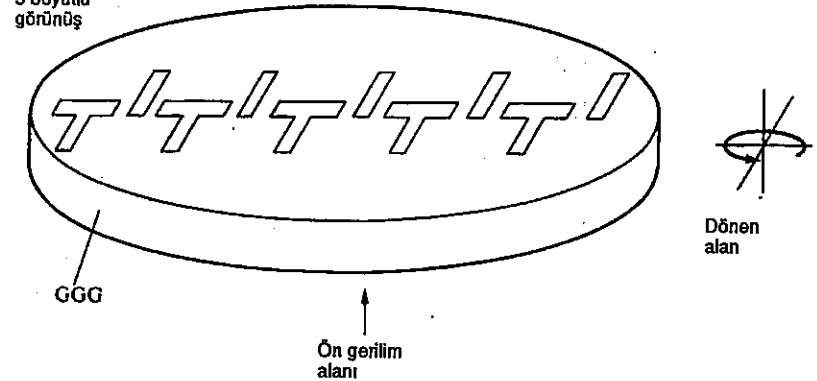
Manyetik-kabarcık teknolojisinin keşfinin, 1967'de Andrew Boback tarafından Bell Laboratuvarlarında yapıldığı bilinir. Manyetik-kabarcıklar özel manyetik malzemeden ince bir katmanın içindeki silindirik bölgeciklerdir. Manyetik-kabarcıklı cihazları üretmek için kullanılan en yaygın malzeme, gadolinyum galyum garnettir (GGG). Daha önce tartıştığımız ince filmler gibi, GGG de, tek eksenli manyetik anizotropi sergiler ve onu bir düzlemde manyetize etmek diğerinden daha kolaydır.



Şekil 6.42 Manyetik kabarcıklar

İnce bir GGG filminin kendi düzlemine dik olarak yönlendirilmiş bir kolay-ekseni vardır. Şekil 6.42'de, derece derece değişen dış bir mıknatıslanma etkisi altındaki bir GGG levhasının yapısı görülmektedir.

Şekil 6.42(a)'da dış alan yoktur ve bölgecikler çizgili veya yılanakavi biçimde rastgele olarak düzenlenmiştir. Eğer filme dik açıyla bir  $H_b$  alanı uygulanırsa, kolay mıknatıslanma düzleminde, dış alanın daralmaya başlamasıyla bölgecikler de onunla aynı yönde mıknatıslanmazlar. Büyük boyutlu bir malzemenin mıknatıslanması dış alana paralel ve aşağı yöndedir. Birbirine paralel olmayan bölgeciklerin

3 boyutlu  
görünüş

Şekil 6.43 Kabarcıkların hareketi

mıknatıslanmasının yukarı yönde olduğu söylenir. Şekil 6.42(b)'de,  $H_p$ 'nin çeşitli değerlerin etkisi görülmektedir.

Dış alan arttıkça, (Şekil 6.42(c)), yukarı mıknatıslanmanın düzensiz şekilli daha da daralır ve yukarıdan bakıldığında da kabarcık şeklini alır. Kabarcık mikrometre ile 0.01 mikrometre arasında olmak üzere çok küçüktür. Tipik kat çapı, yaklaşık 10 mikrometredir. Eğer dış alan Şekil 6.42(d)'de gösterilenne kadar artırılırsa kabarcıklar kararsız hale gelir ve kaybolurlar.

Kabarcık doğal haliyle iki durumlu (var veya yok) olduğundan, ikili veriyi kodlamak için kullanılabilir. Tipik aygıtlar 1 Mbit saklar. Gereken tek şey, istenildiğinde kabarcık üreten bir yöntem ve bunların algılanması, adreslenmesi ve silinmesidir.

Kabarcıklar malzeme düzlemindeki manyetik alanın etkisindeyken çok hareketli değildirler. Kabarcıklar film boyunca enine hareket etseler de gerçekte fiziksel hareket yoktur. Bu, sadece değişen kristaldeki atomların yön değiştirmeleridir.

GGG filmi, hareketsiz altkatmanlara yatırıldığında, bir sonraki katman için taşıyıcı sağlamak ve katmanlar arasında bir engel rolü oynamak üzere üstte bir silisyum dioksit katmanı yerleştirilir. Silisyumdioksitin üstüne bir bakır ve alüminyum alaşımı katman konur. Bu katman fotolitografik tekniklerle ve kabarcık üreten transfer eden, çoğaltan, algılayan ve yok eden yapılar üretmek üzere yerleştirilir. Bunun üstünde ikinci bir silisyumdioksit katmanı bulunur ve bunu da yine fotolitografik tekniklerle ile yerleştirilmiş bir permalloy katmanı izler. Permalloy, manyetik kabarcıklarla reaksiyona giren yüksek geçirimsizlikte manyetik bir malzemedir. Manyetik bir kabarcık belleği kabarcıkları korumak için gereken sabit (statik) alan,  $H_b$ , üretmek üzere iki kalıcı mıknatıs arasına yerleştirilir.

Şekil 6.43'te filmin yüzeyine konulmuş permalloy desenlerinin kabarcıkların hareketine film boyunca kılavuzluk etmek için nasıl kullanıldığı gösterilmiştir. Bu diyagramda film üzerinden aşağı doğru bakıyoruz. Permalloy desenleri T ve I çizgi biçimini alır. Eğer film düzleminde bir  $H_p$  manyetik alanı uygulanırsa permalloy'un T ve I eksenleri manyetize olur. Kutuplama işlevi gören  $H_p$  alanının, sadece kabarcıkları korumak üzere hizmet etmek için  $H_b$  alanıyla dik açıda olduğuna dikkat edin.  $H_p$  değişken (dönen) bir alandır ve film boyunca kabarcıkların hareket etmesi için işlev görür.

Şekil 6.43(a)'da mıknatıslanmanın yönü sağdan sola T'lerin üstü boyunca ve T'lerin üstlerinin mıknatıslanmasına ve ince çubuk bir mıknatıs gibi davranmalara

na neden olur. N kutbu T'nin solundadır. Kabarcıkların, kutupları üst ve altlarında olmak üzere birer fiili mıknatıslanma sütunları olduğunu anımsayın. Kabarcıkların S kutuplarının üstte olduğunu varsayalım. Şekil 6.43(a)'da başlangıçta manyetik kabarcıklar T'lerin N kutbuna doğru çevrilir. Herkesin bildiği gibi eş kutuplar iter, zıt kutuplar çeker.

Düzlem-içi manyetik alan,  $H_p$ , Şekil 6.43(b)'deki durumu oluşturmak üzere saat yönünde 90 derece döndürülür. Burada T'lerin ve I'lerin dikey kısımları manyetize olur. Kabarcığın ilk durumunun solundaki I'nin merkezi S kutbu olur ve kabarcığı iter. T'nin merkezi N kutbu olur ve onu çeker. Bunun sonucu olarak kabarcık T'nin merkezine doğru hareket eder.

Şekil 6.43(c)'de, manyetik alan, saat yönünde bir 90 derece daha döndürülmüştür ve T'lerle I'lerin mıknatıslanması N ve S kutupları birbirine ters olacak şekilde, Şekil 6.43(a) ile aynı biçime gelir.

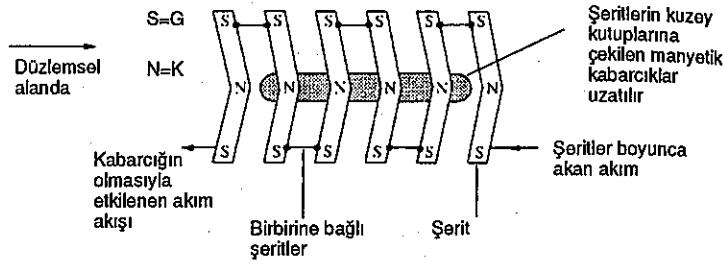
$H_p$ 'nin bir kez daha 90 derece döndürülmesi Şekil 6.43(d)'deki duruma neden olur. Kabarcık, T'nin S kutbu tarafından itilir ve sağındaki I'nin N kutbuna doğru çekilir. Kabarcık T'yi bırakır ve I'ya doğru hareket eder. Kabarcık şimdi Bir T'den bir I'ya hareket ederek kabarcıklık açısından küçük bir adımı fakat bellek teknolojisi açısından dev bir sıçramayı temsil etmektedir.

Şekil 6.43(e)'de alan, bir kez daha 90 derece döndürülmüştür (toplam 360 derece). T'lerle I'ların mıknatıslanması şimdi Şekil 6.43(a)'dakiyle tamamen aynıdır. Kabarcık I'yı terk eder ve sağındaki T'nin N kutbu tarafından çekilir. Kabarcık artık bir T'den diğer bir T'ye hareket etmiştir.

Şekil 6.43'de verilen yapı, bir kaydırmalı kaydedici gibi davranır. Düzlem-içi manyetik alanın bir tam dönüşü flip-floplardan oluşan klasik bir kaydırmalı kaydedicideki saat darbesine karşılık gelir.

Kaydırmalı kaydediciye yapılan benzetme devam ettirilirse veri her saat darbesinde kaydırmalı kaydedicinin en küçük değerlikli konumuna kaydırılır. Manyetik kabarcıklı bellek terimlerinde, kabarcık, mantıksal 1 için T ve I hattına girilerek yaratılır; mantıksal 0 için yaratılmaz. Kabarcık, bakır-alüminyum katmanındaki ince bir iletken döngü boyunca bir akım darbesi geçirilerek yaratılır. Bu, mıknatıslanmanın kolay yönünde bir manyetik alan üretir ve filmin mıknatıslanmasını bir kabarcık yaratmak üzere ters çevirir.  $H_b$  alanını, güçlendirmek için aynı yöndeki döngü boyunca yeterince büyük bir akım geçirilirse kabarcık yok edilir.

Manyetik kabarcıkları algılayan mekanizma, Şekil 6.44'te verilmiştir. E metalik şerit, bir kabarcığı çubuk şekline getirmek suretiyle kabarcığın dış magnetik alanının yoğunluğunu artırma etkisine sahiptir. Kabarcık alanının üzerindeki etkisiyle, onlardan akan akıma karşı elektrik dirençlerini değiştirir dolayısıyla da şeritlerin dirençlerini ölçerek bir kabarcığı algılamak mümkündür.



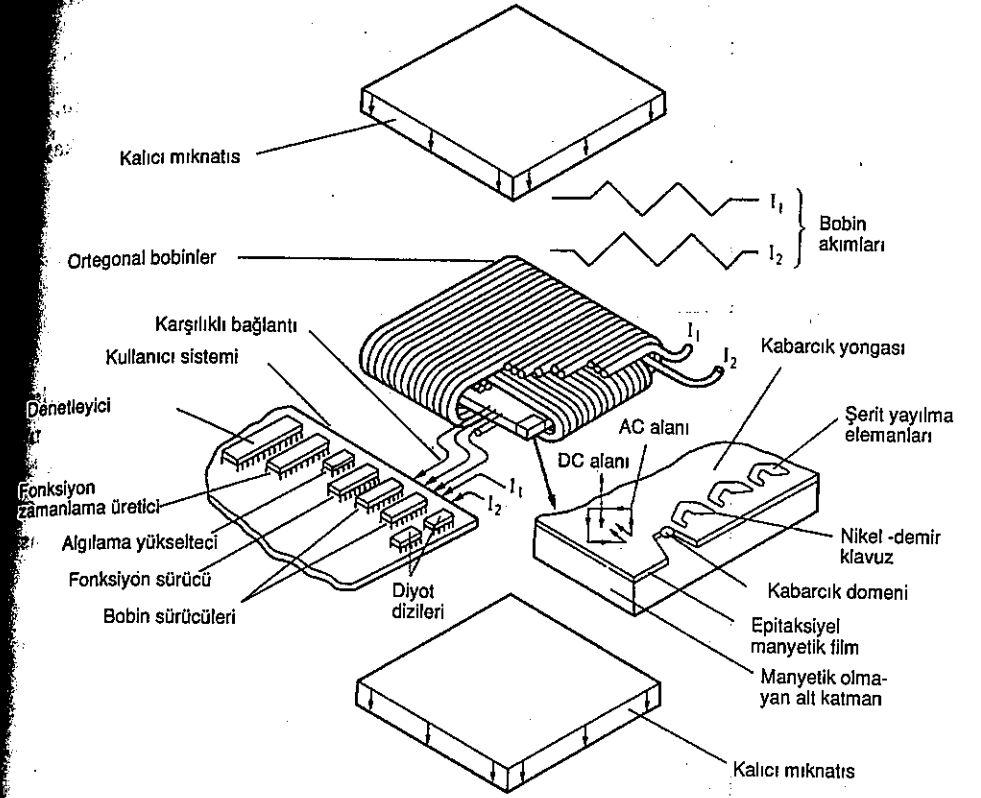
Şekil 6.44 Kabarcık algılayıcı

Artık kabarcıkların nasıl yaratıldığını, kaydırıldığını, algılandığını ve yok edildiğini gördük. Gerçek bir manyetik-kabarcıklı bellek modülünün ayrıntısı, özellikle fiziksel yapısı, düzlem-içi dönen manyetik alan üretmek için gereken elektronik devresi ile üretim/algılama devresi biraz karmaşıktır. Şekil 6.45'te tipik bir manyetik-kabarcıklı bellek modülünün yapısı gösterilmektedir.

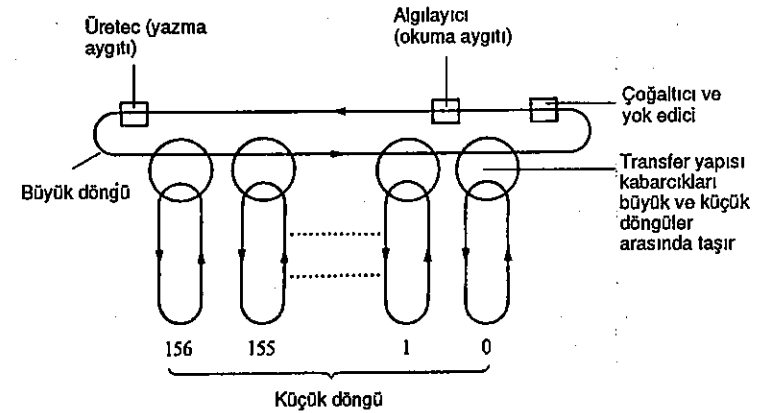
Gerçek bir manyetik-kabarcıklı bellek ile yukarıda verilen açıklama arasındaki büyük fark kaydırmalı kaydedici düzenlenmesindedir. Tek bir büyük döngü uzun bir erişim süresi sonucunu verir ve döngüdeki herhangi bir yanlışlık tüm yongayı kullanımsız kılar. Şekil 6.46'da tipik bir aygıtın büyük-küçük döngü organizasyonu görülmektedir. Bir grup küçük döngü (sözgelimi her biri 513 bit içeren 128 tane) veri saklar. Veri istenen küçük döngüye, kabarcık üretme/algılama devresiyle tüm küçük döngülere bağlı büyük döngü aracılığıyla transfer edilir. Modern cihazların yaklaşık 4 ms'n'lik erişim süreleri, saniyede 0.1 Mbitlik transfer hızları ve 1 Mbitlik kapasiteleri vardır.

### Problemler

1. Dinamik bir RAM'ın fiyatı 3 dolardır ve 65536 x 1 bit olarak organize edilmiştir. Bu yongalarla 0.5 megasözcüklük bir bellek oluşturulacaktır. Her bellek sözcüğü 48 bit genişliğindeyse kaç tane yongaya ihtiyaç vardır? Öteki bileşenlerin maliyeti



Şekil 6.45 Manyetik-kabarcıklı bellek modülünün yapısı



Şekil 6.46 Büyük-küçük döngü düzenlemesi

bellek yongalarının fiyatının %20'si olduğu varsayılırsa belleğin maliyeti ne olacaktır? 1 Megasözcüğün  $2^{20}$  sözcük olduğuna dikkat edin.

2. Manyetik bir bandın inç başına 800 karakterlik bir paketleme yoğunluğu,  $\frac{1}{2}$  inç bloklararası boşluğu vardır ve üzerinde kayıt bulunmaktadır. Her bir kayıt 4 karakter içermektedir. Kayıtlar eğer

- a) tek kayıtlık bloklar      b) 4 kayıt içeren bloklar

şeklinde yazılırsa bandın, kullanışlı veri içeren miktarını bulunuz.

3. Hareketli bir disk sürücünün kayıt için mevcut 10 diski ve 18 yüzeyi vardır. Her yüzey eş-merkezli izden meydana gelmiştir ve diskler 2400 rpm hızında dönmektedir. Her iz, 256 tane 32 bitlik sözcükten oluşan 8 bloğa bölünmüştür. Yüzey başına bir okuma/yazma kafası vardır ve aynı anda, bir silindirin 18 izini okumak mümkündür. İzden ize adımlama süresi 1 msn'dir ( $10^{-3}$  saniye). Aktarımlar arasında kafa, diskin en dıştaki izine park edilir. Aşağıdakileri hesaplayınız:

- Disk sürücünün bit cinsinden toplam kapasitesi
- Saniye başına bit cinsinden maksimum veri aktarım hızı
- Milisaniye cinsinden ortalama erişim zamanı
- Diskte rastgele yerleştirilmiş 256 sözcüklük blokların okunması halinde, ortalama aktarım hızı
- Eğer diskin çapı 12 inçse ve en dıştaki iz, diskin kenarından 1 inç içerideyse en içteki ve en dıştaki izlerin kayıt yoğunluğunu (inç başına bit cinsinden) hesaplayınız. İz yoğunluğu inç başına 200 izdir.

4. Bir mühendis, video kaset kayıt edicilerin fiyatları gün geçtikçe düştüğünden, sayısal verileri saklamak için bunları kullanmayı amaçlamaktadır. Bir saatlik bir kasetin (bit cinsinden) saklama kapasitesi nedir ve veriler, hangi hızda aktarılabilir?

TV'ler, görüntüleri, 625 satır olarak ve saniyede 25 kere tekrar ederek gönderirler. Her bir satırın kullanılan bölümü 256 biti saklamak için kullanılabilir.

5. 3. sorudaki kafa-hareketiyle ilgili varsayım genelde doğru değildir. Bir disk sürücünün 0'dan  $N-1$ 'e kadar numaralandırılmış  $N$  tane eş merkezli silindiri oldu-

ğunu varsayalım. En içteki silindirin numarası 0'dır.

Bir silindirden diğerine ortalama rastgele harekete ilişkin (kafa hareketi sayısı cinsinden) bir ifade türetiniz. Bir sonraki silindir aranırken tüm silindirlerin eşit seçilme olasılıkları olduğunu varsayın. Böylece, veya diğer durumda, ortalama hareketin  $N$ 'nin büyük değerleri için  $N/3$ 'e yaklaştığını gösteriniz.

İpucu : K. silindiri ele alın ve J. silindire gitmek için gereken adım sayısını hesaplayın. Burada J, 0 ile  $N-1$  arasında değişir.

İleri bilgisayar aritmetiği, çarpma ve bölmeyi içeren sayısal işlemlere ilişkin tür işlemler ve türevleri (terslenmiş değerler, karekök alma, üstel ve trigonometrik fonksiyonlar) birçok hesaplama alanlarında özellikle de sayısal hesaplamalarda önemlidir. Çarpma ve bölme üzerine inceleme yapmanın en azından iyi bir bu işlemleri yapmanın sınırsız yolu olması ve dolayısıyla da çarpanın yeni bu işlemleri keşfetmekten kazanılacak sınırsız sayıda doktora ünvanının (veya A.B.D. diplomasının) yapılacağı ve giderleri ödenmiş konferans ziyaretlerinin) olmasıdır.

Bilgisayar dünyasına yeni girenler için, 8-bitlik popüler mikro işlemcilerin çoğunun çarpmayı doğrudan yapamadığını görmek oldukça şaşırtıcı olmaktadır. İki sayıyı, sözgelimi 6502'de çarpmak isteyen herhangi bir programcı, 6502 (genelde kaydırma ve toplama) komutlarını içeren bir algoritmaya başvurmalıdır. Benzer biçimde, yeni başlayan bir kişi, çarpmanın MİB'in komutlarını da ihmal edilmiş olmasını, mikro işlemcinin değerini tümüyle düşüren korkunç dikkatsizlik olarak düşünebilir. Oysa durum böyle değildir. Çarpma veya bölme daha yüksek dereceli aritmetik işlemler birçok programda oldukça düşük oranda yer almaktadır. Örneğin, düzenleyiciler, çeviriciler, derleyiciler ve metin işlemleri tarafından yapılan temel işlemler, verilen bir diziyle eş olanı bulmak üzere alanlarını araştırmaya yöneliktir. Aslında çarpmanın en sık uygulaması aritmetik değil, dizi elemanlarının adreslerinin hesaplanmasıdır. Örneğin  $m$ -satırlı  $n$ -tunluk bir  $X$  matrisindeki  $X_{ij}$  elemanının konumu  $(A + ni + j)$  ile verilir. Burada  $A$  elemanın adresidir.

Bu bölümde sadece çarpma ve bölmeyi ele alacağız. Öteki matematiksel fonksiyonlar ise, normalde çarpmadan türetilirler. Aslında bölmenin kendisi de ileri çarpmayı kapsayan tekrarlamalı bir işlem olarak tanımlanacaktır.

## 7.1 Çarpma

İkili çarpma, temelde, ondalık çarpmadan daha karmaşık değildir. Tüm ikili çarpma tablosunun küçültülmesi, aşağıda verilen biçimde olduğu gibi bir çok yöne yapılabilir:

$$\begin{aligned} 0 \times 0 &= 0 \\ 0 \times 1 &= 0 \\ 1 \times 0 &= 0 \\ 1 \times 1 &= 1 \end{aligned}$$

İki bitin çarpımının mantıksal VE işlemiyle aynı olduğuna dikkat edin. Bit dizisi çarpımını ele aldığımızda işler biraz daha karmaşıklaşır ve çarpımın yapılma veya mekanikleştirilme biçimi geniş ölçüde makineden makineye değişir.

Basitçe, bilgisayar hızlı (ve daha pahalı) oldukça, çarpmayı yapmak için kullanılan donanım da karmaşıklaşacaktır. Basit makineler iki sayının çarpımını insanların yaptığına çok yakın bir şekilde, kaydırma ve toplama yaparak gerçekleştirirler. Yüksek hızlı bilgisayarlar çarpmayı, yüzlerce kapının devreye girdiği tek bir işlemde yerine getirir.

### 7.1.1 İşaretsiz ikili çarpma

Kalem-kağıt algoritması denen ve insanlar tarafından iki çok basamaklı sayının çarpımını hesaplamakta kullanılan algoritma,  $n$  basamaklı bir sayının tek bir basamakla çarpılmasını ve bunların toplanmasını içerir. Bu yaklaşım işaretsiz ikili sayılara, aşağıda verilen biçimde uygulanabilir. En düşük ağırlıklı bitinden başlayarak her seferinde bir tane olmak üzere, çarpan bitleri incelenir. İncelenen çarpan biti bir ise çarpılan, eğer sıfırsa bunun yerine  $n$  tane sıfır yazılır. Ardından, çarpanın diğer bitleri incelenir fakat bu kez çarpılan (veya sıfırı) yazmış olduğumuz basamakların bir soluna kaydırarak yazarız. Bu  $n$  basamakların her birine kısmi çarpım denir. Kısmi çarpımların hepsi yapıldığında toplanarak çarpımın sonucu bulunur. Bu işlem, Örnek 7.1'de açık bir biçimde gösterilmiştir.

Örnek 7.1  $10 \times 13$

10 (çarpan) =  $1010_2$   
13 (çarpan) =  $1101_2$

|         |         |
|---------|---------|
| 1010    |         |
| 1101    |         |
| 1010    | 1. adım |
| 0000    | 2. adım |
| 1010    | 3. adım |
| 1010    | 4. adım |
| 1000010 | 5. adım |

1. adım İlk çarpan biti -1, çarpılanı yaz
2. adım İkinci çarpan biti -0, sola kaydırılmış sıfırları yaz
3. adım Üçüncü çarpan biti -1, sola kaydırılmış çarpılanı yaz
4. adım Dördüncü çarpan biti -1, sola kaydırılmış çarpılanı yaz
5. adım Dört kısmi çarpım sonucunu topla

Sonuç,  $1000010_2 = 130$ , sekiz bit uzunluğundadır. Yukarıdaki algorithmadan,  $n$ -bitlik iki sayının çarpımının  $2n$  bitlik bir çarpım sonucu verdiği şeklinde genel bir sonuç çıkarılmalıdır.

Sayısal bilgisayarlar kalem-kağıt algoritmasını yukarıda verilen biçimde yaparlar, çünkü burada  $n$  tane kısmi toplamın saklanması ve bunun ardından da sözcüğün aynı anda toplanması gerekir. Daha iyi bir teknik kısmi toplam yapıldıkça üstüste toplanmasıdır. 2 tane  $n$ -bitlik işaretli sayının çarpımı için bir algoritma Tablo 7.1(a)'da verilmiştir. Tablo 7.1(b)'de, bu algoritmanın  $1101_2$  ve çarpılan  $1010_2$  olduğunda nasıl çalıştığı gösterilmektedir.

Tablo 7.1 (a) İşaretsiz çarpma algoritması

- Sayıcuyu  $n$ 'ye eşitle
- $2n$ -bitlik kısmi çarpım kaydedicisini sil
- Çarpanın en sağdaki (başlangıçta en küçük değerlikli olan) bitini incele. Eğer  $b$  bit 1'se, çarpılan kısmi çarpımın en değerlikli  $n$  bitine ekle
- Kısmi çarpımı bir basamak sağa kaydır
- Çarpılanı bir basamak sağa kaydır (en sağdaki bit doğal olarak kaybolacaktır)
- Sayıcı bir azalt. Sonuç sıfır değilse, yukarıdaki işlemleri (c) adımıdan itibaren tekrarla. Sonuç sıfırsa, çarpımın sonucunu çarpım kaydedicisinden oku.

Tablo 7.1 (b)  $1010_2 \times 1101_2$ 

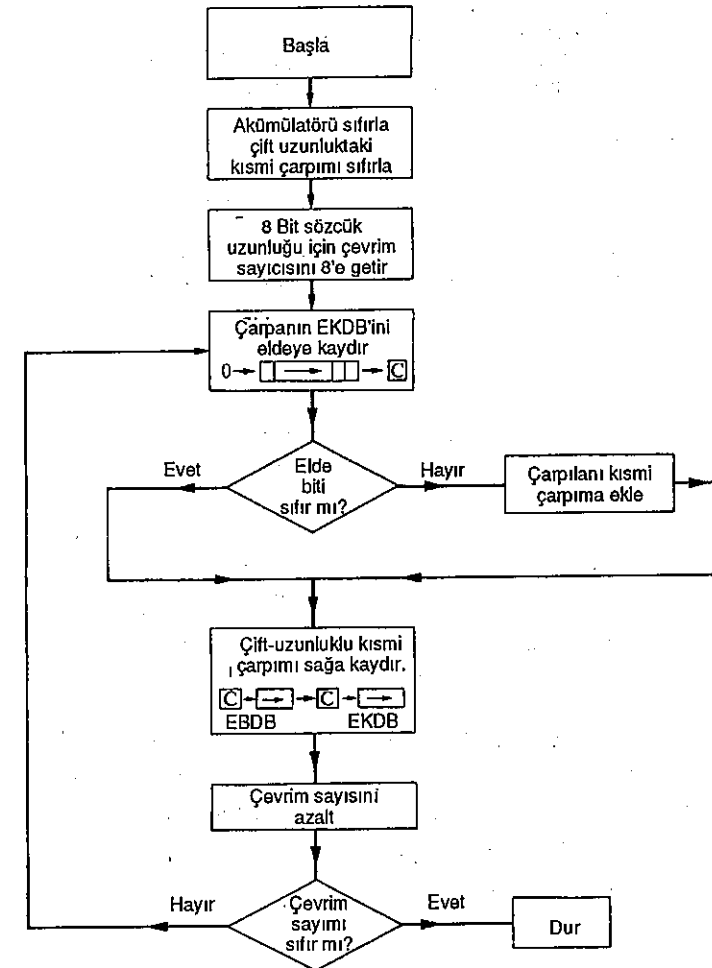
| Adım   | Sayıcı | Çarpan | Kısmi çarpım | Çevrim |
|--------|--------|--------|--------------|--------|
| a ve b | 4      | 1101   | 00000000     |        |
| c      | 4      | 1101   | 10100000     | 1      |
| d ve e | 4      | 0110   | 01010000     | 1      |
| f      | 3      | 0110   | 01010000     | 1      |
| c      | 3      | 0110   | 01010000     | 2      |
| d ve e | 3      | 0011   | 00101000     | 2      |
| f      | 2      | 0011   | 00101000     | 2      |
| c      | 2      | 0011   | 11001000     | 3      |
| d ve e | 2      | 0001   | 01100100     | 3      |
| f      | 1      | 0001   | 01100100     | 3      |
| c      | 1      | 0001   | 100000100    | 4      |
| d ve e | 1      | 0000   | 10000010     | 4      |
| f      | 0      | 0000   | 10000010     | 4      |

### 8-bitlik bir mikroişlemcide çarpma işlemi

Yukarıdaki algoritma tipik bir 8-bitlik mikroişlemciye kolayca uygulanabilir. 6502'de 16-bitlik bir veri kaydedicisi olmadığından, kısmi çarpımının bellekteki ardışık iki konumda saklanması gerekir. Aşağıdaki bellek haritası, çarpan (P), çarpılan (Q) ve kısmi çarpımların (RU) ve (RL) düzenlenmesini gösterir.

|      |    |   |                     |
|------|----|---|---------------------|
| 0000 | P  | ← | çarpan              |
| 0001 | Q  | ← | çarpılan            |
| 0002 | RU | ← | H.Y. baytın çarpımı |
| 0003 | RL | ← | EKD baytın çarpımı  |

Şekil 7.1'deki akış diyagramı, 6502 için çarpma algoritmasını göstermektedir. Şekil 7.1'e karşılık gelen çevirici dilindeki program Tablo 7.1(c)'de verilmiştir.



Şekil 7.1 8-bitlik çarpma algoritması

Bu programda, akümülatörün kendisi kısmi çarpımın en büyük değerlikli tutmaktadır. Düzenleme sırasında kısmi çarpımları sıfırlayan komutlar, fazlalıktır, çünkü zaten sonuç, RU ve RL'de başlangıçta bulunan verilerin yazılmaktadır.

### 7.1.2 İşaretli çarpma

Az önce tartıştığımız çarpma algoritması yalnızca işaretsiz sayılar veya noktalı sayılar için geçerlidir. Birçok bilgisayarlar negatif sayıları, bunların tümleyeni şeklinde gösterildiklerinden; ikiye tümleyen sayılarının çarpımını tutacak bazı yöntemler bulmak gerekmektedir. Elbette ki negatif sayıları ikiye tümleyene dönüştürüp, çarpımı hesapladıktan sonra, eğer negatifse ikiye tümleyene dönüştürmek mümkündür. Ancak bu yaklaşım zaman kaybına yol açar.

Tablo 7.1 (c) Şekil 7.1'e karşılık gelen assembly dili program

|        |     |        |                                      |
|--------|-----|--------|--------------------------------------|
| P      | EQU | 0      | ; çarpanın konumu                    |
| Q      | EQU | 1      | ; çarpılanın konumu                  |
| RU     | EQU | 2      | ; sonucun EBD baytının varışyeri     |
| RL     | EQU | 3      | ; sonucun EKD baytının varışyeri     |
|        | ORG | \$0200 | ; programın başlangıç konumu         |
|        | LDA | # \$00 | ; akümülatörü sıfırla                |
|        | STA | RU     | ; kısmi çarpımın EBD'ını sıfırla     |
|        | STA | RL     | ; kısmi çarpımın EDDB'ını sıfırla    |
|        | LDX | # \$08 | ; sayıcıyı 8'e ayarla                |
| DONGU  | LSR | P      | ; çarpan bitini eldeye kaydır        |
|        | BCC | KAYDIR | ; sıfırsa, kısmi çarpımı sağa kaydır |
|        | CLC |        | ; birse, çarpılana ekle              |
|        | ADC | Q      | ; toplamayı yap                      |
| KAYDIR | ROR | A      | ; sonucun EBD baytını döndür         |
|        | ROR | RL     | ; sonucun EKD baytını döndür         |
|        | DEX |        | ; sayıcıyı azalt                     |
|        | BNE | DONGU  | ; tüm bitler yapıldı kadar döngüle   |
|        | STA | RU     | ; çarpımın EBD baytını sakla         |

Uygun bir algoritma tanıtmadan önce negatif sayıların ikiye tümleyen biçim gösteriminin doğrudan doğruya çarpmada kullanılamayacağını belirtmenin vardır. Örneğin  $X$  ve  $-Y$ 'nin çarpımını ele alalım.

$$-Y \text{ nin ikiye tümleyen biçimi} = 2^n - Y$$

$$X(-Y) \text{ çarpımı} = X(2^n - Y) = 2^n X - XY$$

Beklenen sonuç,  $-XY$ , ikiye tümleyen biçiminde  $2^{2n} - XY$  ile gösterimlenir. Dikkat edilirse en ağırlıklı bit  $2^{2n}$ 'dir ( $2^n$  değil). Çünkü çarpma otomatik olarak

uzunlukta bir sonuç verir. Doğru ikinin tümleyeni sonucunu elde etmek için

$$2^{2n} - 2^n X = 2^n(2^n - X)$$

'lik bir düzeltme payı eklemek gerekir.

Bu değer,  $X$ 'in  $2^n$  ile ölçülen ikiye tümleyenidir. Örnekte;  $X=15$  ve  $Y=-13$ 'ün çarpımı beş bit olarak ele alındığından (Örnek 7.2)

$$X=15 \rightarrow 01111_2 \text{ ve } Y=-13 \rightarrow 10011_2$$

On bitlik sonuç,  $1100111101_2 = -195_{10}$  doğrudur. Benzer biçimde;  $X$  negatif,  $Y$  pozitif olduğunda sonuca,  $2^n(2^n - Y)$ 'lik bir düzeltme payı eklenmelidir.

Çarpan ve çarpılanın ikisi de negatif olduğunda, aşağıdaki durum ortaya çıkar :

$$(2^n - X)(2^n - Y) = 2^{2n} - 2^n X - 2^n Y + XY$$

Bu durumda sonuca  $2^{2n}$  ve  $2^{2n}$ 'lik düzeltme payları eklenmelidir.  $2^{2n}$  en ağırlıklı bir konumundan çıkış eldeleridir ve ihmal edilebilir.

Tablo 7.2 Düzeltme payı ekleyerek işaretli çarpım işlemi

| $2^9$ | $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |                     |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|---------------------|
|       |       |       |       |       |       | 0     | 1     | 1     | 1     | (X)                 |
|       |       |       |       |       |       | 1     | 0     | 0     | 1     | (Y)                 |
|       |       |       |       |       | 0     | 1     | 1     | 1     | 1     |                     |
|       |       |       |       | 0     | 1     | 1     | 1     | 1     |       |                     |
|       |       |       | 0     | 0     | 0     | 0     | 0     |       |       |                     |
|       |       | 0     | 0     | 0     | 0     | 0     |       |       |       |                     |
|       | 0     | 1     | 1     | 1     | 1     |       |       |       |       |                     |
| 0     | 1     | 0     | 0     | 0     | 1     | 1     | 1     | 0     | 1     | Düzeltilmemiş sonuç |
| 1     | 0     | 0     | 0     | 1     |       |       |       |       |       | Düzeltilme payı     |
| 1     | 1     | 0     | 0     | 1     | 1     | 1     | 1     | 0     | 1     | Düzeltilmiş sonuç   |

### Booth algoritması

İşaretli sayıların ikiye tümleyen biçimindeki çarpımına klasik yaklaşım, Booth algoritması ile sağlanır. Bu algoritma iki pozitif, bir negatif veya iki negatif sayı için çalışır. Algoritma geniş bir ölçüde klasik işaretsiz çarpımla benzerlik gösterse de aşağıdaki farklılıkları da içerir : Booth algoritmasında, bir sonra yer alacak üç

biçimden hangisinin ortaya çıkacağını belirlemek için çarpanın iki biti incelenir. Algoritma aşağıda tanımlanmıştır :

1. Eğer o anki çarpan biti 1 ve bir sonraki küçük değerlikli bit 0 ise; çarpılanı çarpımdan çıkar
  2. Eğer o anki çarpan biti 0 ve bir sonraki küçük değerlikli bit 1 ise; çarpılanı çarpıma ekle
  3. Eğer o anki çarpan biti ile bir sonraki küçük değerlikli bit aynı ise bir şey yap
1. Not : Çarpılanı, kısmi çarpıma eklerken toplama sırasında üretilecek hiçbir bitini dikkate almayın.

2. Not : Kısmi çarpım kaydırılırken, aritmetik kaydırma kullanılır ve işaret bit nakledilir.

3. Not : Başlangıçta; çarpanın o anki biti en küçük değerlikli bit ise, bir sonraki küçük değerlikli bitin 0 olduğu varsayılır.

Şekil 7.2'de Booth algoritmasının akış diyagramı verilmektedir. Booth algoritmasının çalışmasını göstermek için aşağıdaki 3 çarpım göz önüne alınmaktadır:  $13 \times 15$ ,  $-13 \times 15$ ,  $-13 \times (-15)$  (Örnekler: 7.3a, 7.3b ve 7.3c şeklinde tablolarda verilmektedir.)

Örnek 7.3 (a) Çarpılan = 01111 (yani +15), çarpan = 01101 (yani +13)

| Adım                      | Çarpan biti | Kısmi çarpım |
|---------------------------|-------------|--------------|
| Çarpılanı çıkar           | 011010      | 000000000    |
| Kısmi çarpımı sağa kaydır |             | 1000010000   |
| Çarpılanla topla          | 01101       | 1100010000   |
| Kısmi çarpımı sağa kaydır |             | 10011110000  |
| Çarpılanı çıkar           | 01101       | 0001111000   |
| Kısmi çarpımı sağa kaydır |             | 1010011000   |
| Hiçbir şey yapma          | 01101       | 1101001100   |
| Kısmi çarpımı sağa kaydır |             | 1101001100   |
| Çarpılanla topla          | 01101       | 1110100110   |
| Kısmi çarpımı sağa kaydır |             | 10110000110  |
|                           |             | 0011000011   |

Sonuç, +195'e eşit olan 0011000011<sub>2</sub>'dir. Burada, altı çizili sayıların her bir adımda incelenecek bitleri gösterdiğine dikkat edin.

Örnek 7.3 (b) Çarpılan = 01111 (yani +15), çarpan = 10011 (yani 13)

| Adım                                                       | Çarpan biti | Kısmi çarpım |
|------------------------------------------------------------|-------------|--------------|
| Çarpılanı çıkar                                            | 100110      | 000000000    |
| Kısmi çarpımı sağa kaydır                                  |             | 1000010000   |
| Hiçbir şey yapma                                           | 10011       | 1100010000   |
| Kısmi çarpımı sağa kaydır                                  |             | 1110001000   |
| Çarpılanla topla                                           | 10011       | 10101101000  |
| Kısmi çarpımı sağa kaydır                                  |             | 0010110100   |
| Hiçbir şey yapma                                           | 10011       | 0010110100   |
| Kısmi çarpımı sağa kaydır                                  |             | 0001011010   |
| Çarpılanı çıkar                                            | 10011       | 1001111010   |
| Kısmi çarpımı sağa kaydır                                  |             | 1100111101   |
| Sonuç, -195'e karşılık gelen 1100111101 <sub>2</sub> 'dir. |             |              |

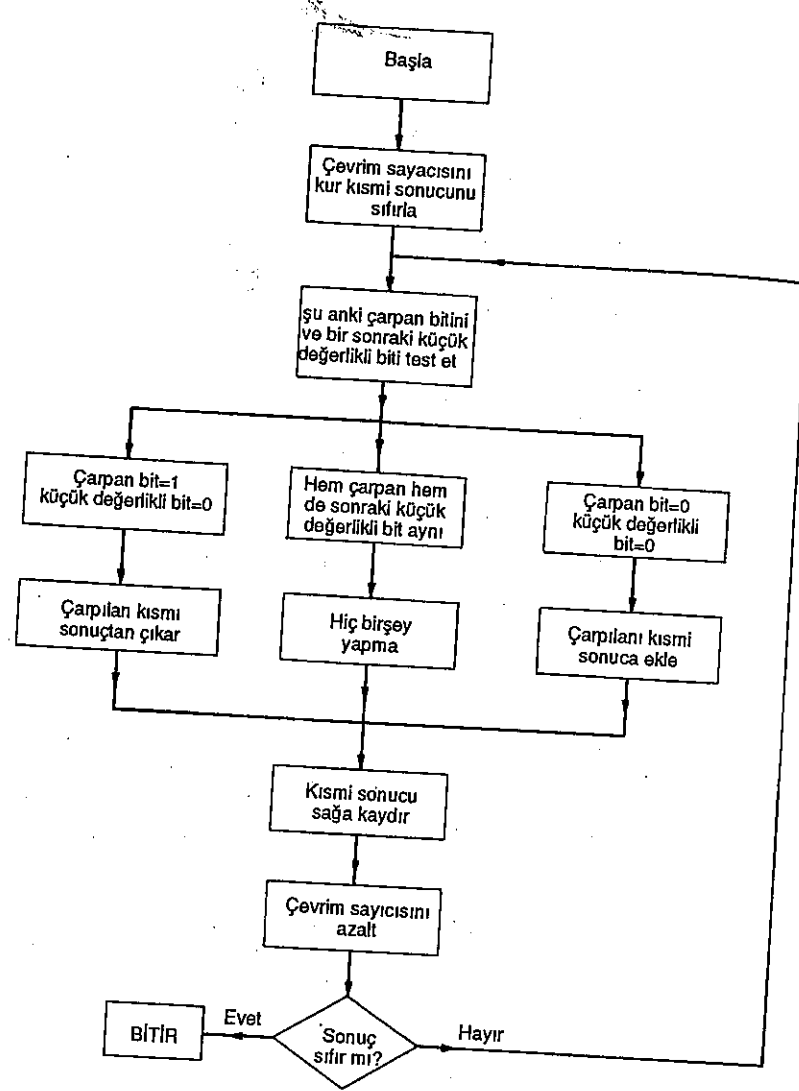
Örnek 7.3 (c) Çarpılan = 10001 (yani -15), çarpan = 10011 (yani -13)

| Adım                                                       | Çarpan biti | Kısmi çarpım |
|------------------------------------------------------------|-------------|--------------|
| Çarpılanı çıkar                                            | 100110      | 0111100000   |
| Kısmi çarpımı sağa kaydır                                  |             | 0011110000   |
| Hiçbir şey yapma                                           | 10011       | 0011110000   |
| Kısmi çarpımı sağa kaydır                                  |             | 0001111000   |
| Çarpılanla topla                                           | 10011       | 1010011000   |
| Kısmi çarpımı sağa kaydır                                  |             | 1101001100   |
| Hiçbir şey yapma                                           | 10011       | 1101001100   |
| Kısmi çarpımı sağa kaydır                                  |             | 1110100110   |
| Çarpılanı çıkar                                            | 10011       | 10110000110  |
| Kısmi çarpımı sağa kaydır                                  |             | 0011000011   |
| Sonuç, +195'e karşılık gelen 0011000011 <sub>2</sub> 'dir. |             |              |

### 7.1.3 Yüksek hızda çarpma

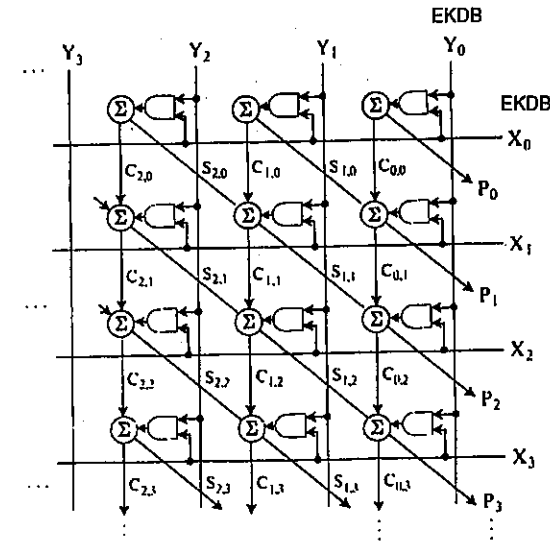
İleri ders kitaplarında genişçe bir bölüm yüksek-hızlı çarpma konusuna ayrılmış olduğundan bu konuyu derinlemesine incelemeyi pek düşünmüyorum. Burada kaydırma ve toplama yöntemlerine alternatif bazı çarpım yöntemleri açıklanmıştır.

Daha önceki örnekte, mantık kapıları ile iki bite iki bit çarpım yapabilmenin mümkün olduğunu gördük. Bu işlem daha çok sayıdaki bitler için genişletilebilir. Halen piyasadan, yaklaşık 50 nsn'de 32 bitlik çarpımlar yapabilen 16-bite 16-bit tek yonga çarpanlar bulunabilmektedir. Şekil 7.3'de, doğrudan iki sayıyı çarpmak için kullanılan mantık dizisi verilmiştir.



Şekil 7.2 Booth algoritmasına ilişkin akış diyagramı

Alternatif bir yaklaşım, iki sayının çarpımının olası tüm sonuçlarının salt okunur bellekte saklandığı bir arama tablosu kullanmaktır. Aşağıdaki tablo,  $2^8=256$  olası sonucu bir ROM'da saklayarak, iki 4-bitlik sayının nasıl çarpılabileceğini göstermektedir. Tablo 7.2, 2. Bölümdeki Şekil 2.73'de bulunan tablo türünün daha geniş bir versiyonudur.



Şekil 7.3 Çarpan dizisi

4-bit çarpan ile 4-bit çarpılan, ROM'daki 256 konumdan birini seçen 8-bitlik bir adres oluşturur. Bu konumların her birinde çarpan (en küçük değerlikli 4 adres biti) ile çarpılanın (en küçük değerlikli 4 adres biti) çarpımı saklanır. Örneğin 2 ile 3'ün çarpımı 00100011 konumunun içerdiği değerle verilir ki bu da 00000110'dir.

Bu tekniğin dezavantajı çarpan ve çarpılanın bit sayısı arttıkça ROM'un kapasitesinin de artması gerektiğidir. Bu nokta, Tablo 7.3'de gösterilmiştir.

8-bitlik iki sayının çarpımı 1.048.576 bitlik bir bellek kapasitesine gereksinim duyar. Daha da büyük sayıların çarpımını arama tablosu ile yapabilmek, artık pratik olmaktan çıkacaktır. Bereket versin ki n-bitlik bir çarpan kullanarak iki tane 2n-bitlik sayının çarpımını bulabilmek mümkündür.

İkili sayılarla nasıl devam edebileceğimize geçmeden önce iki tane 2 basamaklı ondalık sayının çarpımına bir göz atalım ve sonra da bu tekniği ikili aritmetiğe genişletelim.

$$\begin{aligned}
 34 \times 27 &= (3 \times 10 + 4) (2 \times 10 + 7) \\
 &= 3 \times 2 \times 10^2 + 3 \times 7 \times 10 + 4 \times 2 \times 10 + 4 \times 7
 \end{aligned}$$

Tablo 7.2 Arama tablosu ile çarpma

| Adres  |          | Veri     |
|--------|----------|----------|
| ÇARPAN | ÇARPILAN | SONUÇ    |
| 0000   | 0000     | 00000000 |
| 0000   | 0001     | 00000000 |
| .      | .        | .        |
| 0000   | 1111     | 00000000 |
| 0001   | 0000     | 00000000 |
| 0001   | 0001     | 00000001 |
| .      | .        | .        |
| 0001   | 1111     | 00001111 |
| 0010   | 0000     | 00000000 |
| 0010   | 0001     | 00000010 |
| 0010   | 0010     | 00000100 |
| 0010   | 0011     | 00000110 |
| .      | .        | .        |
| 0010   | 1111     | 00011110 |
| .      | .        | .        |
| 1111   | 1101     | 11000011 |
| 1111   | 1110     | 11010010 |
| 1111   | 1111     | 11100001 |

Tablo 7.3 Sözcük uzunluğu ile tablo boyutu ilişkisi

| Çarpan bitleri<br>$n$ | Adres bitleri<br>$2n$ | Tablodaki satırlar<br>$2^{2n}$ | ROM'daki toplam bit<br>$2n \times 2^{2n}$ |
|-----------------------|-----------------------|--------------------------------|-------------------------------------------|
| 2                     | 4                     | 16                             | 64                                        |
| 3                     | 6                     | 64                             | 384                                       |
| 4                     | 8                     | 256                            | 1024                                      |
| 5                     | 10                    | 1024                           | 10240                                     |
| 6                     | 12                    | 4096                           | 49152                                     |
| 7                     | 14                    | 16384                          | 229376                                    |
| 8                     | 16                    | 65536                          | 1048576                                   |

$$\begin{aligned}
 &= 6 \times 10^2 + 21 \times 10 + 8 \times 10 + 28 \\
 &= 6 \times 10^2 + 29 \times 10 + 28 \\
 &= 600 + 290 + 28 \\
 &= 918
 \end{aligned}$$

Şimdi de 4-bit çarpanlarla 8-bitlik iki sayının çarpımını ele alalım. İki 8-bitlik sayı A ve B olsun ve şu şekilde gösterilsin:

$$A = \begin{array}{|c|c|} \hline A_u & A_l \\ \hline \end{array} \quad B = \begin{array}{|c|c|} \hline B_u & B_l \\ \hline \end{array}$$

$\leftarrow 4 \rightarrow \quad \leftarrow 4 \rightarrow \quad \quad \quad \leftarrow 4 \rightarrow \quad \leftarrow 4 \rightarrow$

$A_u$ , A'nın en büyük değerlikli 4 bitini,  $A_l$  de en küçük değerlikli dört bitini gösterir. Daha önce Kısım 3.6'da 8-bitlik bir mikroişlemcide 16-bitlik toplama işleminde sayıların ayrılması konusuyla karşılaşmıştık.

A ve B cebirsel olarak şu şekilde gösterilir :

$$A = A_u \times 16 + A_l \text{ ve } B = B_u \times 16 + B_l$$

$$\text{Bunun sonucu olarak } AB = (A_u \times 16 + A_l) (B_u \times 16 + B_l)$$

$$= 256A_uB_u + 16A_uB_l + 16A_lB_u + A_lB_l$$

Bu ifade dört 4-bitlik çarpımın ( $A_uB_u, A_uB_l, A_lB_u, A_lB_l$ ), bunların 8 ve 4 basamak kaydırılması (yani 256 ve 16 ile çarpılması) ve dört kısmi çarpımın toplanmasıyla ele alınmasını gerektirir. Şekil 7.4'de, bunun nasıl gerçekleştirildiği gösterilmektedir.

## 7.2 Bölme

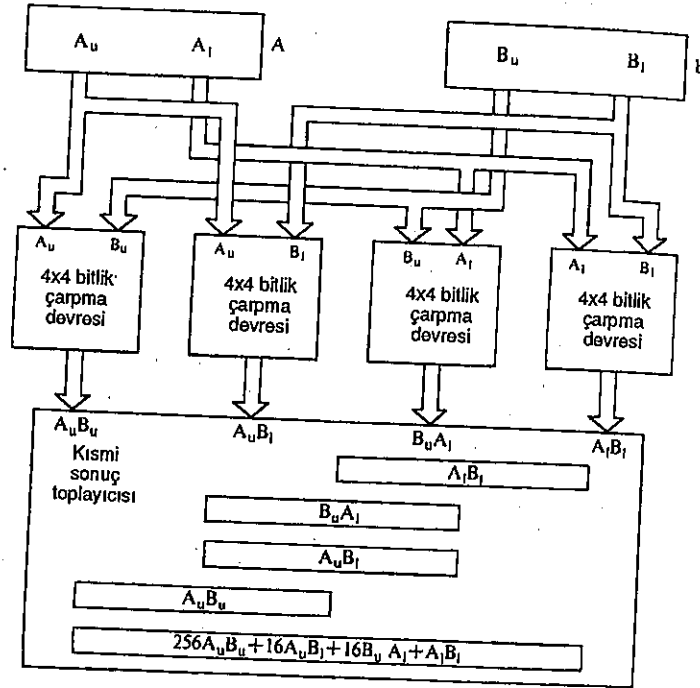
Bölme çarpmanın tersidir ve tekrarlı olarak sonuç, sıfır veya bölenden küçük olana kadar bölünen bölünenden çıkarmakla yerine getirilir. Bölünen çıkarılma sayısına bölüm ve son çıkarmadan sonra geriye kalan sayıya da kalan denir.

$$\frac{\text{bölünen}}{\text{bölen}} = \text{bölüm} + \frac{\text{kalan}}{\text{bölen}}$$

Buna alternatif olarak,

$$\text{bölünen} = \text{bölüm} \times \text{bölen} + \text{kalan}$$

şeklinde de yazabiliriz.



Şekil 7.4 Yüksek-hızda çarpma

İkili bölmeyle ele almadan önce klasik kalem-kağıt tekniğini kullanarak ondalık bölmeyle inceleyelim. Aşağıdaki örnek, 575'in 25'e bölümünü gösterir.

$$\begin{array}{r} \text{bölüm} \\ \text{bölün} \quad \text{bölünen} \quad 25 \overline{) 575} \end{array}$$

İlk adım bölünenin iki basamağı ile bölünenin en büyük değerlikli iki basamağını karşılaştırmak ve bu iki basamakta kaç kere var olduğunu sormaktır. Sonuç ikidir (yani  $2 \times 25 = 50$ ) ve  $2 \times 25$ , 57'den çıkarılır. 2 sayısı aşağıdaki durumu vermek üzere bölümün en büyük değerlikli basamağı olarak girilir.

$$\begin{array}{r} 2 \\ 25 \overline{) 575} \\ \underline{50} \\ 7 \end{array}$$

Bölünenin diğer basamağı aşağı indirilir ve bölün 75 ile karşılaştırılır. 75, 25'in tam katı olduğundan bir tane 3, bir sonraki basamağı olarak bölüne aşağıdaki sonucu vermek üzere girilir.

$$\begin{array}{r} 23 \\ 25 \overline{) 575} \\ \underline{50} \\ 75 \\ \underline{75} \\ 00 \end{array}$$

Bölünenin en küçük değerlikli bitini incelediğimizden ve bölün 75'in tam bir katı olduğundan bölme tamamlanmıştır. Bölüm 23, kalan sıfırdır.

Bölmeyle ilgili bir zorluk da, bölünenin bölünen içinde kaç kere bulunduğu tahmin edilmeye çalışılmasıdır (yani 57, 2 bölümünü ve 7 kalanını vermek üzere 25'e bölündü). İnsanlar bunu zihinlerinden yapsalar da bilgisayara uygulamak üzere bu işlemi mekanikleştirebilmek için bazı yöntemler olmalıdır. Bereket versin ki bu işlem, ikili aritmetikte daha kolaydır. Yukarıdaki örneği, işaretli ikili aritmetiği kullanarak ele alalım.

$$\begin{array}{r} 25 = 11001_2 \quad 575 = 1000111111_2 \\ 11001 \overline{) 1000111111} \\ \underline{11001} \end{array}$$

Bölünenin beş biti bölünenin ilk beş bitinden büyüktür. Böylece bölüne bir sıfır eklenir ve bölün, bölünenin ilk altı biti ile karşılaştırılır.

$$\begin{array}{r} 01 \\ 11001 \overline{) 1000111111} \\ \underline{11001} \\ 001010 \end{array}$$

Bölün bölünenin ilk altı bitinde, 001010 (1111) kısmi kalanını vermek üzere bir kere vardır. Bölünenin bir sonraki biti aşağı indirilir ve

$$\begin{array}{r} 010 \\ 11001 \overline{) 1000111111} \\ \underline{11001} \\ 010101 \\ \underline{11001} \end{array}$$

Kısmi bölünen, bölünenin küçüktür ve bölümün bir sonraki biti olarak bir 0 eklenir. İşlem aşağıdaki gibi devam eder :

|       |            |
|-------|------------|
|       | 010111     |
| 11001 | 1000111111 |
|       | 11001      |
|       | 00101011   |
|       | 11001      |
|       | 000100101  |
|       | 11001      |
|       | 000011001  |
|       | 11001      |
|       | 000000000  |

Bu durumda kısmi bölüm, sıfırdır ve böylece sonuç, 10111 ve kalan 0'dır.

### 7.2.1 Gerigetirmeli bölme

Az önce tartıştığımız klasik kalem-kağıt algoritması çok az değişikliklerle sayısal biçime uyarlanabilir. Yapılan tek değişikliklik bölünen kısmi bölünenle karşılaştırılma biçimidir. İnsanlar bu karşılaştırmayı zihinlerinden yaparlar; bilgisayarlar ise yaptıkları çıkarma işlemi sonucunun işaretini test etmelidir. Eğer çıkarmanın sonucu pozitifse bölüme bir 1 eklenir, sonuç negatifse bölüme bir 0 eklenir ve bölün kısmi bölüme eklenerek eski haline geri getirir. Gerigetirmeli bölme için uygun bir algoritma Tablo 7.4'te verilmiştir. Bu algoritmaya ilişkin akış diyagramı Şekil 7.5'te verilmiştir. Bu algoritmaya bir örnek olarak,  $01100111_2$ 'in  $1001_2$ 'ye bölünmesini ele alalım. Bu,  $103$ 'ün  $9$ 'a bölünmesine karşılık gelir ve 11 bölümünü ve 4 kalanını vermelidir. Bölme işlemi Tablo 7.5'te adım adım gösterilmektedir.

Tablo 7.4 Gerigetirmeli bölme algoritması

1. Bölün, bölünenin en değerlikli biti ile hizala.
2. Bölün, kısmi bölünenden çıkar.
3. Kısmi bölünen sonuçta negatifse, bölüme bir sıfır ekle, kısmi bölüneni geriye getirebilmek için bölüneni tekrar ekle.
4. Kısmi bölünen sonuçta pozitifse, bölüme bir ekle.
5. Bölünen sonucunu bulabilmek için test et. Bölünen en düşük ağırlıklı biti, kısmi bölünenin en düşük değerlikli bitine karşılık geliyorsa, işlemi durdur. Sonuçta elde edilen kısmi bölünen kalan olacaktır. Aksi halde 6. adıma git.
6. Bölün bir basamak sağa kaydır. 2. adımdan tekrar et.

### 7.2.2 Gerigetirmesiz bölme

Şekli 7.5'teki gerigetirmeli bölmeyi, bölme işleminin süresini azaltmak üzere değiştirmek mümkündür. Basitçe, gerigetirmesiz bölme, gerigetirmeli bölmenin

hemen hemen aynıdır. Tek fark, gerigetirme işleminin çıkarılmış olmasıdır. Gerigetirmeli bölmenin akış diyagramından, Şekil 7.5, görülebilir ki bölün ekleyerek kısmi bölünen gerigetirildikten bir sonraki aşamada bölünen yarısı, kısmi bölümden çıkarılmaktadır. Bunun nedeni her aşamada, bölünen, ikiye bölmeye eşdeğerde bir bölün-sağa-kaydır işlemini içermesidir. Böylece o anki aşamadaki bölün-gerigetirme ve bunu izleyen bir sonraki aşamada bölünen yarısını çıkartma işlemi, kısmi bölünene bölünen yarısını tek bir işlemde eklemeye eşdeğerdir. Yani  $D-D/2 = +D/2$  ( $D$  bölündür). Gerigetirmesiz bölme için akış diyagramı Şekil 7.6'da verilmiştir. Bölün kısmi bölünenden çıkarıldıktan sonra, yeni kısmi bölünen test edilir. Eğer negatifse bölümün düşük değerlikli konumuna bir 0 kaydırılır ve bölünen yarısı, kısmi bölünene geri getirilir. Eğer pozitifse bölümün düşük ağırlıklı konumuna bir 1 kaydırılır ve kısmi bölünenden bölünen yarısı çıkarılır. Tablo 7.6'da, gerigetirmesiz bölme kullanılarak Tablo 7.5'te verilen örnek tekrarlanmaktadır.

Tablo 7.5 Geri getirmeli Bölme: 1001 01100111

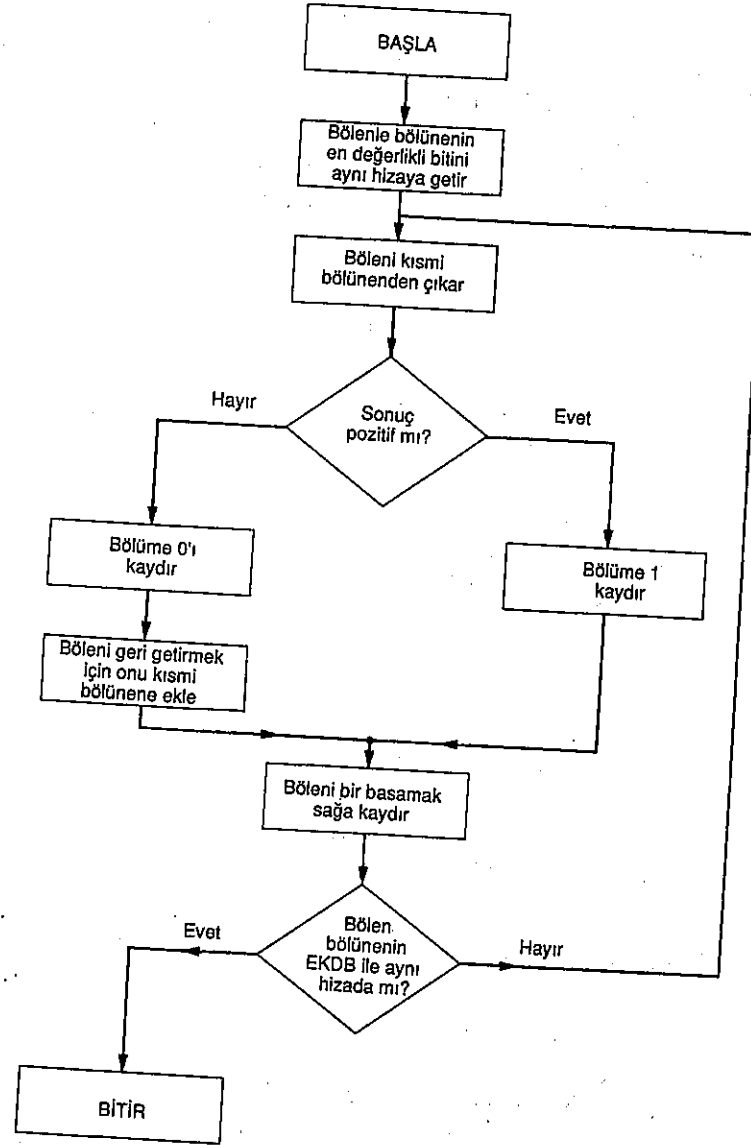
| Adım | Açıklama                       | Kısmi bölünen | Bölün    | Bölüm    |
|------|--------------------------------|---------------|----------|----------|
|      |                                | 01100111      | 00001001 | 00000000 |
| 1    | Hizala                         | 01100111      | 01001000 | 00000000 |
| 2    | Bölün kısmi bölünenden çıkar   | 00011111      | 01001000 | 00000000 |
| 4    | Sonuç pozitifse, bölüme 1 ekle | 00011111      | 01001000 | 00000001 |
| 5    | Test et                        |               |          |          |
| 6    | Bölün bir basamak sağa kaydır  | 00011111      | 00100100 | 00000001 |
| 2    | Bölün kısmi bölünenden çıkar   | 00000101      | 00100100 | 00000001 |
| 3    | Bölün geri getir bölüme 0 ekle | 00011111      | 00100100 | 00000010 |
| 5    | Test et                        |               |          |          |
| 6    | Bölün bir basamak sağa kaydır  | 00011111      | 00010010 | 00000010 |
| 2    | Bölün kısmi bölünenden çıkar   | 00001101      | 00010010 | 00000010 |
| 4    | Sonuç pozitifse, bölüme 1 ekle | 00001101      | 00010010 | 00000101 |
| 5    | Test et                        |               |          |          |
| 6    | Bölün bir basamak sağa kaydır  | 00001101      | 00001001 | 00000101 |
| 2    | Bölün kısmi bölünenden çıkar   | 00000100      | 00001001 | 00000101 |
| 4    | Sonuç pozitifse, bölüme 1 ekle | 00000100      | 00001001 | 00001011 |
| 5    | Test et                        |               |          |          |

Bölüm =  $1011_2$  kalan =  $100_2$

### 7.2.3 Çarpma yaparak bölme

Hem bilgisayarlar hem de mikroişlemciler, bölmeyi çarpmadan çok daha az sıklıkta yaptıklarından bölme için daha az özel-amaçlı donanım vardır. Öte yandan bölmeyi çarpma, toplama ve kaydırma aracılığıyla yapmak mümkündür. Yüksek-hızlı çarpanlar piyasada kolayca mevcut olduğundan bunlar bölmeye uygulanabilir.

$Q$  bölümünü elde etmek için  $N$  bölüneni  $D$  bölüne bölmek istediğimizi varsayalım. Yani,  $Q=N/D$ . İlk adım  $D$ 'yi büyüklük sırasındaki yerine koymaktır; böylece  $D$ ,  $\frac{1}{2} \leq D < 1$  aralığına konulur.



Şekil 7.5 Geri getirmeli bölmeyle ilişkin akış diyagramı

Yukarıdaki işlem  $D$ 'yi sola veya sağa kaydırarak ve kaydırma sayısını kaydederek yapılır. Bu kayan noktalı aritmetiğin normalize edilmesiyle tamamen bir analogi oluşturur. Yeni bir sayı,  $Z$ ,  $D$  biçiminde  $Z=1-D$  olarak tanımlanır.  $D$  değeri,  $\frac{1}{2}$  ile 1 arasında bulunduğundan  $Z$  de, 0 ile  $\frac{1}{2}$  arasında bulunur.

Aritmetiğin basit bir kuralı vardır : Bir kesrin payı ve paydası aynı sayı ile çarpılırsa kesrin değeri değişmez.

Böylece,  $Q = N/D = KN/KD$ .

$K=1+Z$  olduğunu varsayarak

$$Q = \frac{N}{D} = \frac{N(1+Z)}{D(1+Z)} = \frac{N(1+Z)}{(1-Z)(1+Z)} = \frac{N(1+Z)}{1-Z^2}$$

Şimdi eğer işlemi  $K=1+Z^2$  ile tekrarlırsak  $Q$ ,

$$\frac{N(1+Z)}{1-Z^2} \cdot \frac{1+Z^2}{1+Z^2} = \frac{N(1+Z)(1+Z^2)}{1-Z^4}$$

olur. Bu işlem, aşağıdaki sonucu vermek üzere  $n$  kere tekrar edilebilir :

$$Q = \frac{N}{D} = \frac{N(1+Z)(1+Z^2)(1+Z^4) \dots (1+Z^{2^{n-1}})}{1-Z^{2^n}}$$

$Z$  birden küçük olduğu için  $n$  arttıkça  $Z$ 'nin değeri hızla sıfıra yaklaşır. Bunun sonucu olarak  $Q$ 'nun yaklaşık değeri;

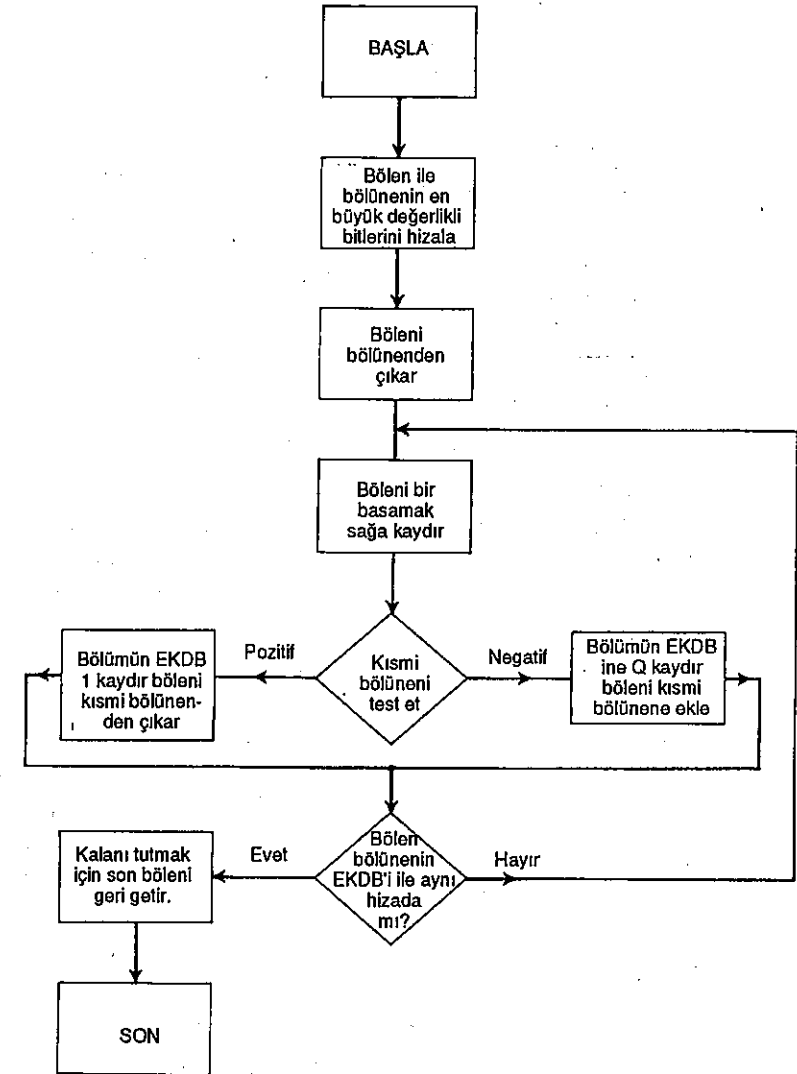
$$Q = N(1+Z)(1+Z^2)(1+Z^4) \dots (1+Z^{2^{n-1}}),$$

ile verilir.

8-bitlik duyarlılık için  $n$ 'nin 3 olması gerekir; eğer  $n=5$  ise bölüm 32-bitlik bir hassasiyet verir. Büyüklük sırasına konulursa yukarıdaki formülden hesaplanacak ilgili bölüm,  $Q$ , istenen sonucu üretmek üzere aynı faktör ile sıraya konulmalıdır.

Tablo 7.6 Bir gerigetirmesiz bölme örneği

| Adım | Açıklama                                                             | Kısmi bölünen | Bölen     | Bölüm    |
|------|----------------------------------------------------------------------|---------------|-----------|----------|
| 1    | Bölüni hizala                                                        | 01100111      | 00001001  | 00000000 |
| 2    | Bölüni kısmi bölünenden çıkar                                        | 01100111      | 01001000  | 00000000 |
| 3    | Bölüni sağa kaydır                                                   | 00011111      | 01001000  | 00000000 |
| 4    | Kısmi bölüneni test et, bölüme 1 ekle, bölüni kısmi bölünenden çıkar | 00011111      | 00100100  | 00000000 |
| 6    | İşlemin sona erip ermediğini test et                                 | - 00000101    | 00100100  | 00000001 |
| 3    | Bölüni sağa kaydır                                                   | - 00000101    | 00100100  | 00000001 |
| 5    | Kısmi bölüneni test et, bölüme 0 ekle, bölüni kısmi bölünene ekle    | - 00000101    | 00010010  | 00000001 |
| 6    | İşlemin sona erip ermediğini test et                                 | 00001101      | 00010010  | 00000010 |
| 3    | Bölüni sağa kaydır                                                   | 00001101      | 00010010  | 00000010 |
| 4    | Kısmi bölüneni test et, bölüme 1 ekle, bölüni kısmi bölünenden çıkar | 00001101      | 00001001  | 00000010 |
| 6    | İşlemin sona erip ermediğini test et                                 | 00000100      | 00001001  | 00000101 |
| 3    | Bölüni sağa kaydır                                                   | 00000100      | 00001001  | 00000101 |
| 4    | Kısmi bölüneni test et, bölüme 1 ekle, bölüni kısmi bölünenden çıkar | 00000100      | 0000100.1 | 00000101 |
| 6    | İşlemin sona erip ermediğini test et                                 | - 00000000.1  | 0000100.1 | 00001011 |
| 7    | En son kısmi bölüneni geri getir                                     | - 00000000.1  | 0000100.1 | 00001011 |
|      |                                                                      | 00000100      | 0000100.1 | 00001011 |



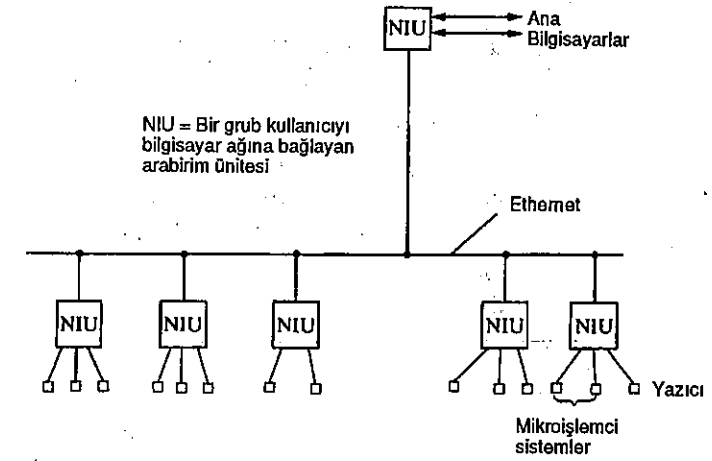
Şekil 7.6 Gerigetirmesiz bölmeyle ilişkin akış diyagramı

Bu bölümde bilgisayarların birbirleriyle iletişim kurma biçimi incelenecektir. Bir bilgisayar neden bir diğeri ile iletişim kurmak ister? Bunu sorduğunuzu duyar gibiyim. Nasıl insanlar kendi aralarında işbirliğine giderlerse, pratikte bilgisayarlar da birbirleriyle iletişim kurarlar. Olanakları ve kaynakları paylaşarak  $n$  sayıda bir grup insan bir kişinin tek başına başarabileceğinin  $n$  katından daha fazlasını yapabilir. Bu durum şu deyimle özetlenmiştir: Bütün, parçalarının toplamından daha fazladır.

Bir ağ oluşturmak için bilgisayarları birbirine bağlamanın getireceği avantajları basit biçimde göstermek üzere, aşağıdaki örneği ele alalım. Bir bilimadamı, bir deneyi denetlemek için bilgisayar kullanmaktadır. Bir gün, sayısal bir hesaplama yapmak zorunda kalır. Bu hesaplama öyle karmaşıktır ki birkaç gün için minibilgisayarın tüm zamanını buna ayırması gerekecektir. Bu süre boyunca deney durdurulabilir. Eğer deneyin sürekli çalıştırılması gerekiyorsa bilim adamı hem deneyin hem de hesaplamanın üstesinden gelebilecek yetkinlikte yeni ve daha güçlü bir bilgisayar satın alabilir. Ne yazık ki bu yaklaşım, yeni bilgisayarın hesaplama tamamlandıktan sonra çok az kullanılacak olduğundan çok savurganca bir çözüm olacaktır. Çok daha iyi bir çözüm büyük bir anabilgisayardan süre satın almak ve sadece yapılan iş için ücret ödemektir. Bu, iki yoldan biriyle sonuçlanabilir: Bunlardan biri; problemi, manyetik bir bant veya disket aracılığıyla fiziksel olarak anabilgisayara aktarmaktır. Bir diğeri de; problemi, veri taşıyan bir ağ üzerinden minibilgisayardan anabilgisayara iletmek ve sonra da sonuçları anabilgisayardan almaktır.

Bilgisayarlar arasında fiziksel bağlantılar kurmak çok pahalıya geleceğinden bağlantı, kamuya açık telefon şebekesi üzerinden yapılabilir. Telefon şebekesi verileri, yüksek hızda işleyebilmek için tasarlanmamış olduğundan bu ideal bir çözüm değildir. Oldukça (sözgelimi 1 km'den fazla) uzak mesafede bulunan bilgisayarları birbirine bağlayan bu tip ağlara, genel iletişim ağları (WAN) denir.

Son birkaç yılda bilgisayar kullanım biçimlerinde önemli değişiklikler meydana gelmiştir. Düşük maliyetli mikrobilgisayar ve minibilgisayar seli, bilgisayar tarafından denetlenebilen çevreirim sayısında büyük bir artışa yol açmıştır. Artık birçok farklı bilgisayar ve çevreiriminin, verilen bir yerde, örneğin bir fabrikada, birbirine bağlamak yaygınlaşmıştır. Bu, verinin paylaşılmasını, merkezi olarak denetlenmesini ve verimliliğinin artırılmasını sağlar. Bu tür ağlara da yerel iletişim ağı (LAN) denir.



Şekil 8.1 Teesside Politeknik yerel iletişim ağı

Buna güzel bir örnek Teesside Politeknik'in Bilgisayar Bilimleri Bölümü'nde kurulu bulunan LAN'dır. Şekil 8.1'de, bilgisayar ve çevreirim gruplarının NIU'lara (ağ arabirimlerine) nasıl bağlandığını ve birbirleriyle bir Ethernet, birçok üreticide yok-satan LAN türlerinden biridir, üzerinden bağlanan NIU'lar gösterilmiştir. NIU'ların her birinin kendi mikroişlemcileri vardır ve bir grup aygıtı birbirleriyle ağ üzerinden bağlamak üzere hizmet verir. Kavram olarak, bunlar bir telefon santraliyle aynı işlevi görürler. Ders veren her görevlinin odasına bir VDU yerleştirilir. Bu, yakındaki UNIVAC 1100 veya PRIME 750 anabilgisayarı ile iletişim kurulmasını sağlar. Bu kitap, kendi odamdan PRIME bilgisayarına girilerek yayına hazırlanmıştır. Anabilgisayarların yanısıra yazıcılar ve diğer çevreirimleriyle birlikte bir grup mini ve mikrobilgisayar da ağa bağlanmıştır.

Yakın gelecekte, otomatik ofis olanakları sağlamak üzere bir yazılım geliştirileceği düşünülmektedir. Örneğin, ders veren bir görevli, saklama ortamına, ihtiyacı olanların yararlanmasına sunulmak üzere mesajlar yazabilecektir. Kullanıcılar, belleğe eriştiğinde kendisiyle ilgili olan mesajları sorgulayabilecektir. Mesaj VDU'da görüntülenebilecek ve gerekli olduğunda da, mesajın basılı bir kopyasını alabileceklerdir.

Bu bölümün ilk kısmında, sayısal verilerin her seferinde bir bit olmak üzere bir noktadan diğerine nasıl taşındığı konusu işlenmiştir. Yine burada, verilerin paralel biçimden seri biçime dönüştürülme teknikleri ve telefon kanalları üzerinden iletilme tekniklerine de yer verilmiştir.

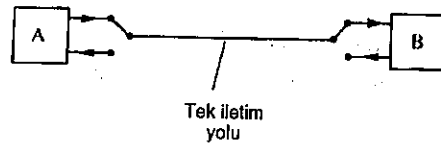
Öykünün tümü, basitçe verilerin bir noktadan diğerine taşınması değildir. Kısım 8.2'de, bilgisayarlarda düzenli şekilde veri alış-verişini sağlayan protokollere (kurallara) bir giriş yapılmıştır. Protokoller verinin doğru hedefe gitmesini sağlarlar ve kaybolan ya da bozulan verilere ilişkin problemleri ele alırlar.

Kısım 8.3, yerel iletişim ağlarına ayrılmıştır ve bu kısımda, halen kullanımda olan bazı LAN'ların özellikleri açıklanacaktır. LAN'ların önemli bir özelliği, bilgisayarlarla çevrebirimlerinin görünürde bir çelişki yaratmadan aynı ağı paylaşabilmeleridir.

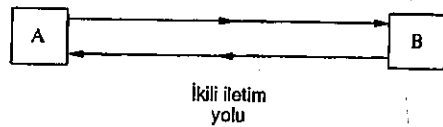
### 8.1 Seri veri iletimi

İdeal olarak, bilgi bir bilgisayardan diğerine her seferinde bir sözcük olarak ve sözcüğün  $m$  bitinin tümü aynı anda olacak şekilde taşınır. Kavramsal olarak, bu bir bilgisayarın veri yolunu diğerlerinininkine bağlamayı gerektirir. Bu birbirleriyle birkaç metre uzaklıktaki bilgisayarlar için makul düzeydeyken daha uzun mesafelerde için pratik değildir.  $m$ -bitlik paralel veri anayolu, veriyi taşımak için en az  $m$  tane bağlantı teline ve bilginin akışını denetlemek için iki ile dört arasında ek tele gereksinim duyar.

Hemen hemen tüm iletişim ağları, verileri noktadan noktaya her seferinde bir bit olacak şekilde seri olarak iletirler. Bu düzenleme yalnızca iki hatta gereksinim duyar; biri veriyi taşımak için diğeri de toprak dönüşü içindir. Gerilimin ancak, toprak veya yer gibi bir referans noktasına göre belirtilmesi halinde bir anlamı olacağına dikkat edin. Eğer iki nokta tek bir yol üzerinden bağlanırsa herhangi bir anda veri sadece tek bir yönde taşınabilecektir. Bu mod Şekil 8.2'de gösterilmiştir ve buna yarı-dupleks iletim denir. Eğer dönüş hattı için Şekil 8.3'teki gibi fazladan bir yol sağlanırsa, eşzamanlı iki yönlü iletim mümkün olur. Buna tam-dupleks denir ve iki tarafın da aynı anda konuşup dinleyebildiği telefon konuşmasına karşılık gelir.



Şekil 8.2 Yarı-dupleks iletim

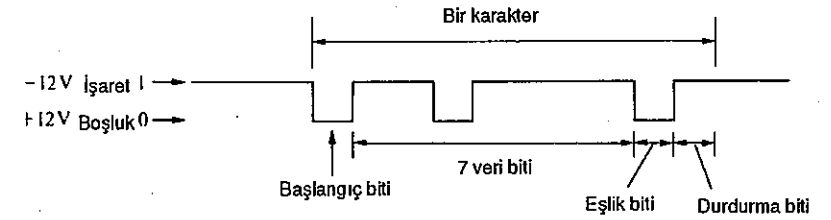


Şekil 8.3 Tam-dupleks iletim

Seri veri iletiminde iki problem söz konusudur. Veri akışı tek tek bitlere ve bitler de ayrı sözcüklere nasıl bölünecektir? Veriler, uzun mesafelerde fiziksel olarak nasıl iletilir? Veri akışının bitlere ve sözcüklere bölünüşü iki yoldan yapılabilir: asenkron ve senkron. Bu iletim biçimleri ayrı ayrı ele alınacaktır. Verinin uzun mesafelerde iletimi, modülasyon ve demodülasyon kavramlarını içermektedir ve bu bölümde, daha sonra ele alınacaktır.

#### 8.1.1 Asenkron seri iletim

Asenkron seri iletim sisteminde verici ve alıcılardaki veri akışını bitlere bölmekten sorumlu saatler, birbiriyle eşzamanlı olarak çalışmazlar. Şekil 8.4'de tek bir 7-bitlik karaktere karşılık gelen dalga biçimi gösterilmiştir. Vericinin çıkışı verinin gönderilmediği sürelerde mantıksal 1'de durur ve hat boştur. Bu 'işaret' durumuna karşılık gelir ve kısa mesafelerde çalışan birçok sistemde -12 V ile gösterilir.



Şekil 8.4 Asenkron seri iletim

Bunu takiben, bit periyodu hattın mantıksal 1 veya 0 durumunda olabileceği en kısa süredir. Verici bir sözcük göndermek istediğinde hattı, bir bit periyodu kadar mantıksal 0 (boşluk) durumuna getirir. Mantıksal 0, +12 V ile gösterilir. Alıcı bu mantıksal 0'ı gördüğünde, ki buna başlatma biti denir, bir karakterin gelmek üzere olduğunu anlar. Gelen veri akışı daha sonra yedi bit periyoduna bölünür ve veri her bir bitin merkezinde örneklenir. Alıcının saati vericinininkiyle eşzamanlı değildir ve bitler tam olarak merkezde örneklenemez. Öte yandan eğer alıcı saati vericinin saatinin yaklaşık %4'ü değerindeyse, sistem iyi biçimde çalışır.

Yedi veri biti gönderildikten sonra bir hatadan korunma ölçüsü vermek üzere bir eşlik biti iletilir. Eğer alıcı, alınan eşlik değerinin hesaplanan eşlik değerine eşit olmadığını görürse hata bayrağını kaldırır ve o anki karakter kabul edilmez.

Eşlik bitinin ardından mantıksal bir düzeyinde bir (veya seçmeli olarak iki) durdurma biti bulunur. Durdurma biti herhangi bir bilgi taşımaz ve sadece ardışık karakterler arasında bir ayırıcı olarak işlev görür. Durdurma biti gönderildikten

sonra istendiği zaman yeni bir karakter gönderilebilir. Asenkron seri veri iletişimi genelde veriyi (ASCII kodlanmış) karakter biçiminde iletmek için kullanılır.

Tek bir bitin aldığı süre  $T$  saniye ise karakterin uzunluğu

$$\text{başlatma biti} + 7 \text{ veri biti} + \text{eşlik biti} + \text{durdurma biti} = 10T$$

ile verilir. Asenkron iletimin verimsiz bir iletim biçimi olduğuna dikkat. Çünkü faydalı yedi bilgi bitini göndermek için on veri bitine ihtiyaç duyulmaktadır. Yaygın kullanımda, asenkron veri iletişimi için birkaç format vardır. Bazılarında her ter başına yedi veri biti, bazıları ise sekiz veri biti gönderir. Bazılarında tek bit, bazılarında çift eşlik biti varken bazılarında da eşlik biti bulunmaz.

Seri veri bağlantısının çalışma hızı saniyede bit cinsinden ifade edilir ve tipik olarak 110 ile 9600 bps arasındadır. İkili (yani iki düzeyli) bir sinyal için saniyede bir baud denir. Böylece 2400 bps hızında veri ileten bir VDU'nun 2400 baud çalıştığı söylenir. Bir iletim sisteminin baud hızı saniyedeki sinyal durum değişikliklerinin sayısı olarak tanımlanır ve aynı zamanda sinyalleme hızı olarak da adlandırılır. Eğer veri 2400 baudda sekiz düzeyli sinyal biçiminde iletiliyor olursa her sinyal bileşeni 3 bit ( $8=2^3$ ) bilgiye eşit olduğundan bit hızı 7200 bps olacaktır.

Alıcı, karakterin tüm bitlerini elde ettikten sonra karakter 5. Bölümde tartışılan tekniklerin biri kullanılarak bilgisayar tarafından okunur. Aslında seri veri iletişiminin gönderilmesi ve alınması, tamamiyle özel-amaçlı entegre devreler tarafından gerçekleştirilir. ne getirildiği için bilgisayarın kendisi seri iletiminin ayrıntıları konusunda endişelenmek zorunda değildir.

### 8.1.2 Senkron seri iletim

Asenkron veri hatları geniş ölçüde çevre birimleri ile (VDU'lar, yazıcılar, modemler) bilgisayarları bağlamak için kullanılır. Bilgilerin, bir ağdaki bilgisayarlar arasında gönderilmesi gerektiği yerlerde genel olarak senkron seri iletişimi kullanılır. Senkron bir seri veri iletim sisteminde bilgi, komşu bit grupları arasında boşluk olmadan kesiksiz biçimde gönderilir. Bit grubu ifadesini kullanıyorum çünkü senkron sistemler, çoğu zaman ASCII kodlanmış bir karakter dizisinden çok daha seferinde sadece ikili bilgiden oluşan blokların bütünü gönderirler.

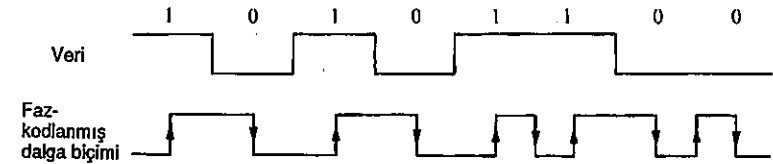
Senkron bir veri sistem tasarımcısının karşılaştığı iki problem vardır: Biri, veri akışının bitlere; ve diğeri ise, veri bitlerinin de anlamlı gruplara nasıl bölüneceğidir?

### Bit senkronizasyonu

Eğer gönderici saatinin bir eşi de alıcıda mevcut olsaydı veri akışını bitlere ayırmak zor olmayacaktı. Ne yazık ki bu yöntem, saat için ek bir yola gereksinim duyacak ve dolayısıyla da veri hattının maliyetini arttıracaktır. Daha iyi bir çözüm, senkronizasyon sinyalinin, veri sinyali içinde olacak şekilde kodlanmasıdır. 6. Bölümde sayısal verinin manyetik ortamda kayıt edilmesine ilişkin kodlama teknikleri tanıtılmıştır. Bu tekniklerden bazıları kolaylıkla veri iletimine uygulanabilir. Eğer veri akışı faz kodlamalı ise, alınan sinyalden bir başka sinyal türetilir ve veri de bu yolla elde edilir. Şekil 8.5'te faz-kodlanmış bir sinyal görülmektedir.

### Sözcük senkronizasyonu

İlk bakışta, kesintisiz bit akışını anlamlı sözcüklere bölmek, güç bir iş gibi görünebilir.



Şekil 8.5 Faz-kodlamalı senkron seri iletim

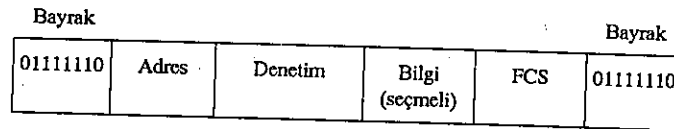
Aslında bitlerin sözcüklere dönüştürmek oldukça kolaydır. Burada, metindeki sözcükler arası boşlukları kaldırarak cümlelerin okunmasını imkansız hale getirmedim, ancak zorlaştırdım. Zorluk nedeni, okurun harf dizileri araştırması ve sözcüklere karşılık gelen tanımlanabilir grupları aramaya çalışmasından dolayıdır. Bu teknik, ikili veri akışına da uygulanabilir. Senkron iletim sistemleri iki moddan birinde çalışır: karakter yönelimli ve blok-iletimli. İlki veri akışını karakterlere ayırır, ikincisi ise akışı çok daha uzun saf ikili veriden oluşan bloklar halinde gruplar.

### Karakter-yönelimli iletim

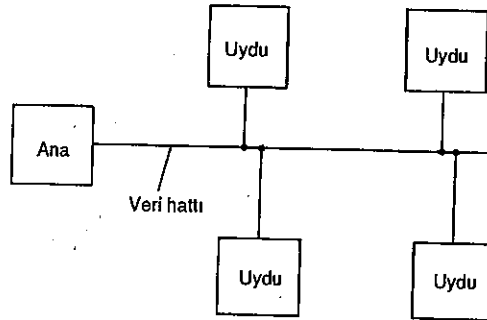
Karakter-yönelimli iletimde gönderilecek veriler (çoğu zaman) ASCII karakter biçiminde kodlanırlar. Örneğin; Alan [harf] dizisi, aşağıdaki dört tane 7-bit karakter sırası şeklinde gönderilecek. Her bir harf şu şekilde kodlanır: 'A'=\$41, 'I'=\$6C, 'a'=\$61, 'n'=\$6E.

1000001001101110000110111011





Şekil 8.7 HDLC çerçeve formatı



Not: Tüm iletişim bir uydu ile ana istasyon arasındadır. Doğrudan uydudan uyduya iletişime izin verilmez.

Şekil 8.8 HDLC ile Ana-uydu iletişimi

Adres alanını takiben 8-bitlik bir denetim alanı vardır. Bu, düzenli bir mesaj akışından emin olmak üzere veri hattını denetler ve kayıp mesajların sonuçlarıyla ilgilenir. Bu konu protokoller kısmında daha ayrıntılı olarak ele alınacaktır.

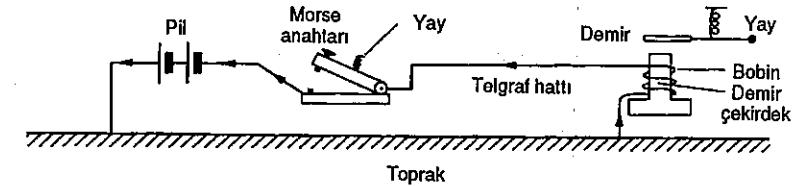
Adres alanını, iletilecek gerçek veriyi içeren seçmeli bir veri alanı (veya bilgi veya I-alanı) takip eder. Bu alan seçmelidir. Çünkü kimi zaman çerçeveler sadece kontrol amaçlı olarak gönderilir. Bu alanın uzunluğu 5,6,7 veya sekiz bit sözcük olabilir. Veri alanının hemen ardından 16-bitlik çerçeve kontrol dizisi (FCS) gelir. Bu, gönderme sırasındaki hataları algılayan güçlü bir yöntemdir. Burada FCS'nin ayrıntılarına inmek mümkün değildir. Fakat aşağıdaki bilgi, üretilmesi ve uygulanmasıyla ilgili bir fikir verebilecektir.

Açılış bayrağı ile FCS'nin kendisi arasında yeralan paketteki  $p$  tane bit,  $p$  dereceden bir polinomun katsayıları olarak düşünülür. Bu polinom modül 2 aritmetiği kullanılarak ve bir bölüm ile bir kalan üretmek üzere standart bir polinoma bölünür. Bölüm dikkate alınmaz. 16-bitlik kalan FCS'dir ve veri alanından sonra gönderilir. Alıcıda adres alanının başlangıcından FCS'ye kadar olan bitler aynı polinoma bölünür ve yeni elde edilen kalan değeri alınan FCS ile karşılaştırılır. Eğer elde edilen yeni kalan değeri, alınan FCS ile aynıysa her şey yolundadır. Değilse çerçevenin tümü iptal edilir. Üreticinin polinomu dikkatle seçilerek iletişim hatalarının büyük çoğunluğunu algılamak mümkündür.

FCS'yi çerçevenin bitimini belirten bit kapatma bayrağı izler. Aslında, alıcı veri alanının bitişini yalnızca kapatma bayrağından 16 bit geriye sayarak belirler.

### Modülasyon ve veri iletişimi

Sayısal veri iletiminin geçmişi, sayısal bilgisayarların bir yüzyıl öncesine kadar gider. 1809 kadar eski bir tarihte Kral Maximillian, Bavarian Bilimler Akademisi'nden, uzun mesafelerde yüksek hızda iletişim için bir şema önermelerini istemişti. Çünkü Fransız görsel semafor sisteminin Napolyon'un askeri etkinliklerine nasıl yardımcı olduğunu görmüştü. Sonuç olarak, Sommering (her biri bir karakter için) 35 iletker kullanan kaba bir telgraf tasarlamıştır. Bir iletkenden elektrik geçirildiğinde alıcı tarafta iletkenden akan bu akım, suyun içindeki hidrojen ve oksijeni ayrıştırarak ve oluşan kabarcıklardan akımın varlığı anlaşılmış olacaktı.

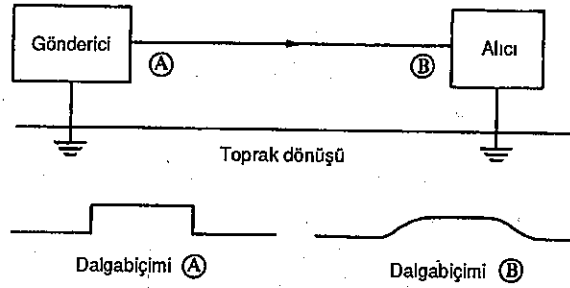


Şekil 8.9 Telgraf

Elektrikle manyetizma arasındaki ilişki keşfedildikten kısa bir süre sonra telgraf icat edildi ve 1830'larda iyice yaygınlaştı. Şekil 8.9'da, telgrafın çalışması gösterilmiştir. Anahtara basıldığında devreden bir akım geçerek solenoidi (bobin içindeki mıknatıslayıcı demir çekirdeği) enerjilendirir. Küçük bir demir levha demir çekirdek tarafından çekildikçe duyulabilir bir klik sesi çıkar. 1843'te Morse, asistanı A. Vail'i baskılarını ayarlamak için kullandıkları harflerin oransal hızlarını saymak için yazıcının başına göndermişti. Sonuçta elde edilen kod dört sembolden oluşuyordu: nokta, tire (=3 nokta), noktalar ve tireler arasında boşluk ve sözcükler arasında boşluklar. Sıkça tekrarlanan 'E' gibi harflere kısa kodlar ('E'='.') seyrek tekrarlanan harflere uzun semboller ('Q'='---') verilmişti. Morse kodunun Huffman'ın İngiliz dili için optimum koduna çok yakın olduğunu belirtmek gerekir.

19. yüzyılda teknolojiye ilerlemelerle birlikte iletişim hatlarının uzunluğu fiziksel olarak arttıkça iletişim sırasında sinyallerin distorsiyona (bozulma) uğradıkları görülmüştür. Göndericiden gelen keskince yükselen bir darbe, uzun yükselme ve düşme süresiyle birlikte oldukça bozulmuş darbe olarak alınır. Transatlantik kablo projesinin destekleyicileri bu bozulmanın etkisinden kaygıya düşmüşler ve sonun-

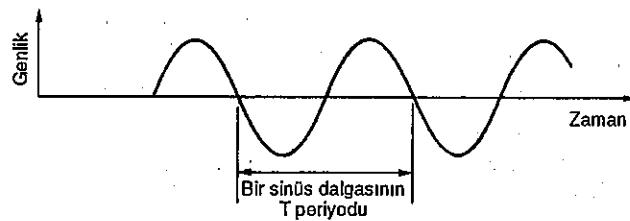
da durumu Lord Kelvin'e aktarmışlardı. 1855'te Kelvin, Royal Society'ye darbenin bozulma etkisini çözümleyen bir rapor sunmuştur. Bu rapor şimdi iletim hatları teorisi denen şeyin temel taşıdır.



Şekil 8.10 Telgraf bozulması

Kelvin'in araştırdığı problemlerin nedeni elektriksel iletkenlerin fiziksel özelliklerine ilişkindi. En basitiyle, iletim yolunun etkisi, sinyallerin durum değiştirme hızını azaltmasıdır.

1880'den sonra telefon bilgi iletim sistemlerine egemen olmuştur. Ne yazık ki telefon analog sinyal iletir ve telefon ağları sayısal veri iletimi için yeterince uygun değildir. Bunun nedeni telefon ağının hem yavaş ve hem de hızlı değişen sinyalleri iletememesidir. İlginçtir ki bu, manyetik kayıt etme aygıtlarının probleminin aynısıdır.

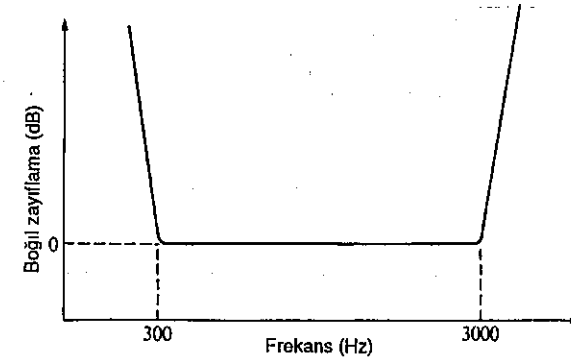


Şekil 8.11 Sinüs dalgabıçımı

Elektronğin temel dalgabıçımı Şekil 8.11'deki gibi sinüs dalgasıdır. Bir iletim sistemi, iletim ağı üzerinden iletilen sinüs dalgasının karşılaştığı zayıflama (yani gücündeki azalma) ile karakterize edilebilir. Şekil 8.12'de bir telefon kanalının frekans karşılığı görülmektedir. Burada 300 Hz ile 3000 Hz aralığındaki frekanslar çok düşük düzeyde bir zayıflama ile iletilir. Bu sınırların altında veya üstündeki

frekanslar ise zayıflarlar. Sayısal bir darbenin,  $f$  darbenin genişliğini göstermek üzere,  $f$ ,  $2f$ ,  $3f$ ,... frekanslı bir dizi sinüs dalgasından oluştuğu gösterilebilir. Teorik olarak, sinüs dalgaları sonsuz frekanslara kadar genişletilebilir.

Eğer bir dizi ikili sinyal telefon şebekesine verilirse o kadar ciddi şekilde bozulurlar ki sonunda devrenin alıcı ucunda kullanılmaz hale gelmiş olur. Bunun nedeni, darbeyi oluşturan yüksek ve alçak frekanslı sinüs dalgalarının ağ tarafından iletilmemesidir.

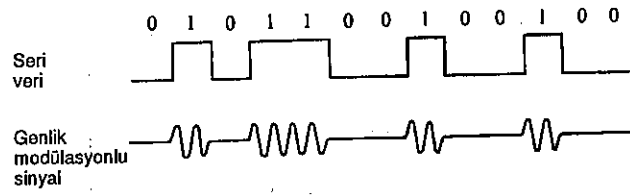


Şekil 8.12 Telefon şebekesinin karakteristik eğrisi

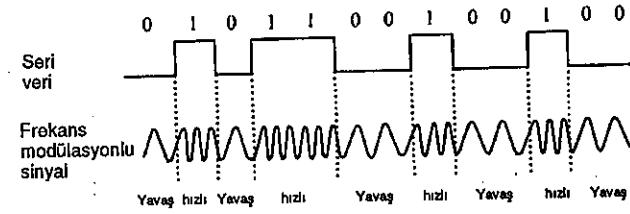
Telefon ağları, konuşma sinyallerini iletebildiklerinden sayısal bilgiyi konuşma--türü sinyallere dönüştürmenin çeşitli yolları araştırılmıştır. Şekil 8.13'de sayısal verinin, bir sinüs dalgasının genliğini değiştirmek veya modüle etmek için nasıl kullanılabileceği görülmektedir. Buna genlik modülasyonu (GM) veya AM (Amplitude Modulation) denir. Bu tür bir sinyal üretmek için gereken donanıma modülatör, sonuçtaki sinyalden veriyi yeniden elde etmek için gereken donanıma da demodülatör denir. Bilgisayar ile telefon sistemi arasındaki arabirime modem (MODülatör, DEModülatör) denir. AM'nin gürültü duyarlılığı diğer modülasyon tekniklerinden daha yüksek olduğundan veri iletiminde yaygın olarak kullanılmaz.

Sinüs dalgasının genliğini değiştirmek yerine, sayısal veriye bağlı olarak frekansını değiştirmek mümkündür. İkili bir sistemde belli bir frekans bir ikili değeri farklı bir frekans da diğerini gösterir. Şekil 8.14'te frekans modülasyonu (FM) görülmektedir. FM yaygın olarak kullanılmaktadır. Çünkü gürültüye karşı AM'den daha bağıstıktır.

Bir başka modülasyon biçimi Şekil 8.15'te görülmekte olan faz modülasyonudur (PM). Bu durumda sinüs dalgasının fazı, sayısal sinyale bağlı olarak değiştirilir.

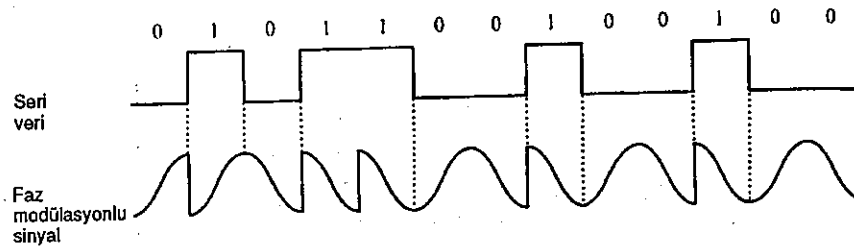


Şekil 8.13 Genlik modülasyonu



Şekil 8.14 Frekans modülasyonu

PM yaygın olarak kullanılır ve FM ile benzer özellikleri vardır.

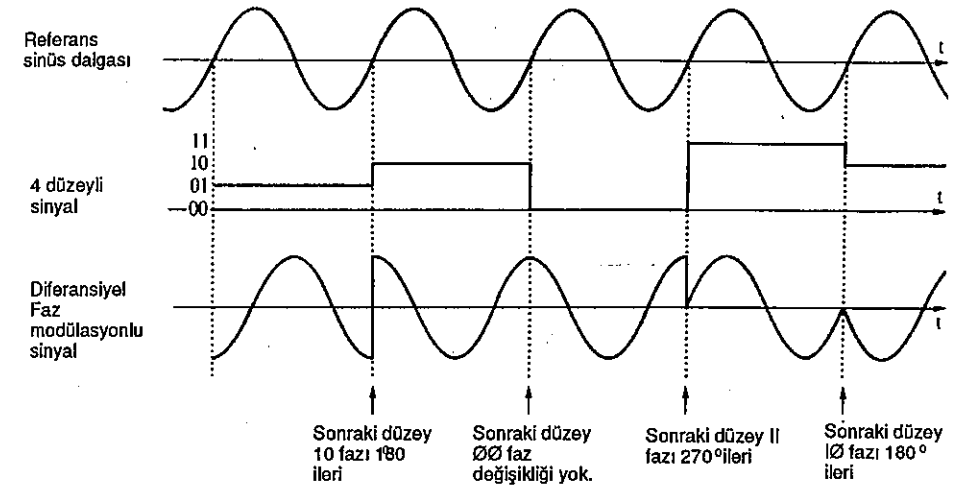


Şekil 8.15 Faz modülasyonu

Eğer, mantıksal 1'e karşılık gelen faz değişimi 180 derece ve mantık bir 0'a karşılık gelen de 0 derece (değişme yok) ise, her bir zaman aralığında bir bitlik bilgi iletilir (Şekil 8.15). Öte yandan faz değişimi 90 derecenin katlarıyla kaydırılırsa her seferinde iki bit gönderilebilir (Şekil 8.16, Tablo 8.1).

Tablo 8.1 Faz kodlamalı çok-düzeyle sinyal

| Gönderilen bitler | Faz değişimi |
|-------------------|--------------|
| 00                | 0            |
| 01                | 90           |
| 10                | 180          |
| 11                | 270          |



Şekil 8.16 Diferansiyel faz modülasyonu

Modemler 300 bps.'ten 9600 bps'e kadar geniş bir bit hızı aralığında çalışırlar. Genelde, düşük bit hızları bazı hatların çok kötü olduğu ve sinyal zayıflamalarının veri hızını 2400 bps'e veya daha altına düşürdüğü anahtarlamalı telefon ağlarına karşılık gelir. 4800 bps ve 9600 bps'lik daha yüksek hızlar telefon şirketinin daha yüksek düzeyde hizmet verdiği özel kiralık hatlarda bulunur.

Bugün verileri sıradan telefon hatları üzerinden ve 19200 bps'te iletmek çok karmaşık gönderme ve alma teknikleri kullanarak mümkün olmaktadır. Bir sinyalin genliğini ve fazını eşzamanlı olarak değiştirerek bir kerede birkaç bit bilgi gönderilebilir. Buna kuantize genlik modülasyonu (QAM) denir. Tipik olarak, 16-nokta QAM bir gönderide dört bit göndermek için 16 farklı dalgabıçımını kullanır. Bu, veri hızının, sinyalleme hızının dört katı olduğu anlamına gelir.

Telefon kanalı, yüksek sinyalleme hızında gönderilen dalga biçimlerini ciddi şekilde bozduğundan orijinal sinyali yeniden oluşturmak için bazı yöntemler gerekmektedir. Buna eşitleme denir ve bazı yüksek hızlı modemlerde bulunur. Telefon kanalının özellikleri, yapılan çağrının türüne bağlı olarak değiştiğinden; eşitleyicinin o an kullanımında olan kanalın özelliklerine uyması gerekir.

## 8.2 Protokoller

Kısım 8.1'de veri iletiminin fiziksel özelliklerini ele aldık. Burada iki taraf arasındaki konuşmaların yönetilme biçimine bakacağız. Diyelim ki bankadaki hesabımda bulunan paradan daha fazla kredi kullandım ve hesabımı kapatmak için bir çek gönderdim. Eğer birkaç gün sonra müdürden tehditkar bir mektup alırsam bundan hangi sonucu çıkarırım? Acaba çekim, müdürün mektubu gönderildikten sonra mı alındı? Diğer borçlarımdan biri gelip borç miktarını mı arttırdı? Çekim postada mı kayboldu?

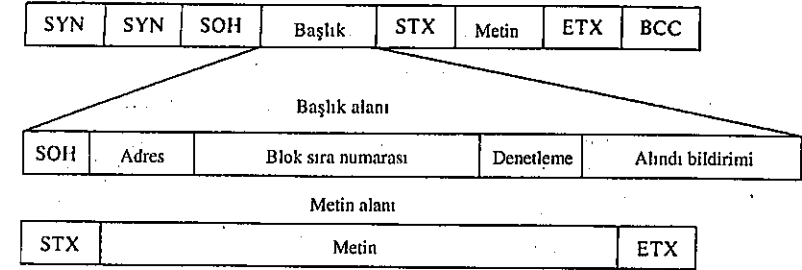
Yukarıdaki durum bilginin gözü kapalı iletiminin kafa karıştırıcı durumlara yol açabileceğini göstermektedir. Her iki tarafın da hangi mesajların alınıp hangilerinin alınmadığını tam olarak bilmeleri gerekir. Bu tür iki veya daha çok taraf arasındaki karşılıklı bilgi alışverişini yöneten kural takımına protokol denir. 5. Bölümde giriş/çıkış ve onaylamayı ele alırken karşılıklı veri değiş-tokuşuna ilişkin temel bir protokol kavramıyla tanışmıştık.

İki nokta arasında mesaj iletimi için uluslararası kabul görmüş bir grup protokol vardır. Bunlar iki gruba ayrılır: bayt denetim-yönelimli protokoller (BCP) karakter biçiminde iletilen veri için tasarlanmıştır; bit-yönelimli protokoller (BOP) saf ikili veri akışıyla birlikte kullanım için düzenlenmiştir. Burada bu protokol türlerinden herbirine ayrı ayrı bakacağız. BISYNC, IBM tarafından tasarlanmış BCP protokolüdür ve HDLC de (yüksek düzeyli veri bağlantısı denetimi) yaygın şekilde kullanılan bir BOP'tur. Genelde, yönelimli protokoller hızla dünya standardı haline gelmektedir ve BISYNC gibi karakter-yönelimli protokollerin kullanımının azalacağı beklenmektedir.

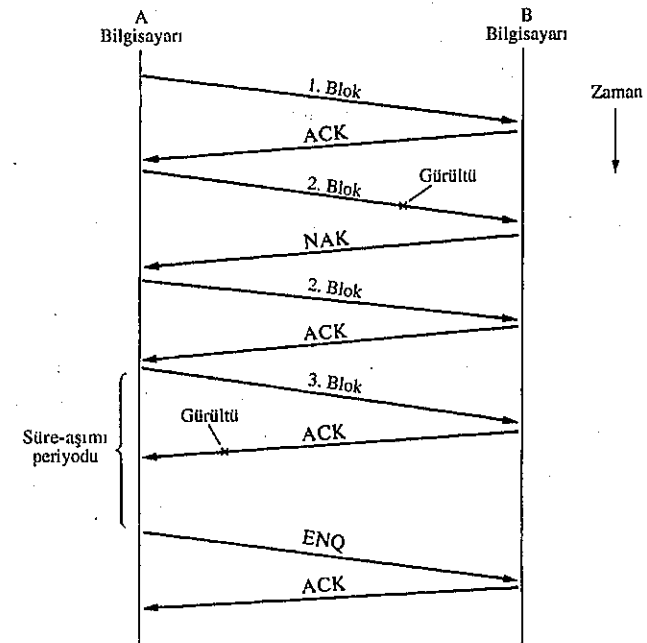
## BISYNC

BISYNC mesajı Şekil 8.17'de verilmiştir. BISYNC belli denetim işlemlerini yerine getirmek için ASCII koddaki özel karakterleri kullanır. İki senkronlama karakteri, SYN, mesajın başlangıcını belirtir. Bir sonraki alan SOH (başlık başlangıcı) karakteriyle başlayan başlık alanıdır. Başlık alanı aslında bir denetim alanıdır ve özel ASCII karakterler aracılığıyla verinin akışını denetlemek için kullanılır. Başlık alanından sonra STX (metin başlangıcı) karakteri ile başlayan metin alanı gelir. Metin alanını ETX (metin sonu) karakteri ve bir hata algılama kodu (BCC) izler.

Şekil 8.18'de iki sistem, A ve B, arasındaki karşılıklı mesaj değiş-tokuşunun grafik olarak nasıl gösterileceği yer almaktadır. Düşey eksen geçen zamanı gösterir ve böylece diyagram yukarıdan aşağıya doğru okunur. Başlangıçta A bilgisayarı B



Şekil 8.17 BISYNC çerçevesi formatı



A'dan 2. Blok gürültü ile bozulur. B bunu algılar ve A'ya negatif alındı bildirimi (NAK) göndererek A'yı mesajı yeniden göndermeye davet eder. 3. Blok B tarafından yeterli derecede alınır fakat alındı bildirimi A'ya giden yolda kaybolur. Uygun bir gecikmeden sonra A bir soruşturma isteği gönderir ve B alındı bildirimini tekrarlar. Bu düzenleme güvenilir değildir ve karışıklık içerir (metne bakınız).

Şekil 8.18 BISYNC ile iki-yollu iletim

bilgisayarına Blok 1 adlı bir mesaj gönderir. Bu B tarafından alınır ve ACK adlı bir alındı mesajı gönderilir.

İkinci blok gürültü tarafından bozulmuştur ve B olumsuz alındı, NAK, mesajı göndererek A'yı iletimini tekrarlamaya davet eder. Blok 3, B tarafından yeterli biçimde alınır fakat B'nin alındı bildirimi kaybolur. Eğer bir şey yapılmazsa sistem, A'nın sonsuza kadar bir karşılık bekler şekilde durmasına neden olacaktır. Bu açmazdan kurtulmak için A, mesajı gönderdiğinde bir zamanlayıcıyı başlatır. Eğer makul bir süre zarfında karşılık alınmazsa bir süre-aşımı üretilerek A, B'ye bir karşılık isteği (ENQ) mesajı göndermeye zorlanır. Daha sonra B son mesajı alıp almadığına bağlı olarak bir ACK veya NAK ile cevap verir.

Ne yazık ki yukarıdaki şema potansiyel bir karışıklık içermektedir. Eğer A tamamıyla kaybolan bir mesaj gönderirse ne ACK ne de NAK alacak ve bir süre-aşımından sonra bir karşılık-isteği gönderecektir. B karşılık-isteğini aldığı anda aldığı son mesaja karşılık bir alındı bildirimi gönderir. Bu, A'nın en son gönderdiği mesaj olmayıp, ondan bir öncekidir. A'nın gönderdiği son mesaj kaybolmuştur ve ne A'nın ne de B'nin bundan haberi yoktur. A, B'den ACK'yi aldığı anda son mesajı tekrarlamak yerine bir sonraki mesajı gönderir.

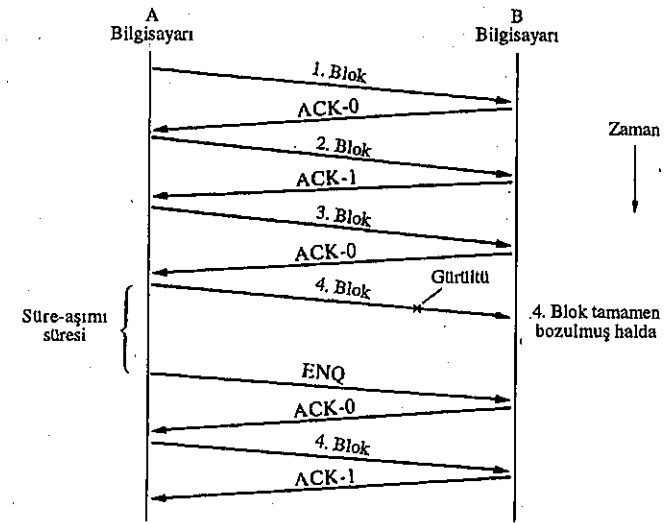
Bu karışıklıktan çıkış yolu, numaralı alındı bildirimi kullanmaktır. En basit düzenlemede ACK-0 ve ACK-1 olarak adlandırılan iki alındı bildirimi kullanılır. Bu kodlar sıra ile kullanılır. Şekil 8.19'da iki ACK'nin bu karışıklığı nasıl çözdüğü görülmektedir.

Karakter-yönelimli portokolün yukarıdaki ele alınış biçimi, sadece mesajların düzgün şekilde karşılıklı olarak nasıl değiştirildiği hakkında bir fikir veremeyi amaçlamıştır. Aramaların hazırlanması ve bir iletimden sonra temizlenmesi buraya alınmamıştır. Bu konuda daha fazla ayrıntı isteyen okurlar, kaynakçaya başvurmalıdır. Özellikle Housley, protokollere çok açık bir giriş yapmıştır.

### HDLC

Saf ikili veri iletimi için yüksek düzeyde veri bağlantı denetim şeması artık uluslararası bir standart haline gelmiştir ve birçok farklı veri bağlantı türlerinde bulunur. Daha önce HDLC çerçeve formatının bir grup alana bölündüğünü görmüştük. 8-bitlik denetim alanı mesaj sıralamaya ayrılmıştır. Denetim alanının formatı Şekil 8.20'de görüldüğü gibi 0 ve 1 bitlerle belirlenir.

Denetim alanının 0. biti 0 olduğunda, çerçevenin bir bilgi çerçevesi olduğu söylenir, çünkü çerçevede bir bilgi alanı bulunmaktadır. Bilgi çerçevesinin denetim alanı iki tane 3-bitlik sayı içerir, N(S) ve N(R). Bunların işlevi birbirleriyle iletişimdeki iki bilgisayar arasındaki mesaj akışının kaydını tutmaktır.



Bu örnekte alındı bildirimi mesajı için iki alternatif değer kullanılır. Bunlar ACK-0 ve ACK-1 olup daima dönüşümlü biçimde gönderilir: ACK-0, ACK-1, ACK-0, ACK-1, ACK-0 ..... Bir ACK-0 aldıktan sonra A 4-blok gönderir. Bu blok ciddi biçimde bozulmuştur ve B tarafından alınmaz. Süre aşımından sonra A bir sorgulama isteği gönderir ve cevap olarak ACK-0 alır. A, ACK-0'ın 3. Bloka cevap olduğunu bilir. Eğer 4-blok alınmış olsaydı A, ACK-1'i görmüş olacaktı.

Şekil 8.19 ACK-0 ve ACK-1'in kullanılması

| 0               | 1     | 2   | 3     | 4 | 5 | 6 | 7 | 0                   | 1 | 2 | 3   | 4     | 5 | 6 | 7 | 0                 | 1 | 2 | 3   | 4 | 5 | 6 | 7 |
|-----------------|-------|-----|-------|---|---|---|---|---------------------|---|---|-----|-------|---|---|---|-------------------|---|---|-----|---|---|---|---|
| 0               | N (S) | P/F | N (R) |   |   |   |   | 1                   | 0 | S | P/F | N (R) |   |   |   | 1                 | 1 | M | P/F | M |   |   |   |
| Bilgi çerçevesi |       |     |       |   |   |   |   | Denetleyici çerçeve |   |   |     |       |   |   |   | Numarasız çerçeve |   |   |     |   |   |   |   |

N (S) - Gönderme sırası sayısı, N (R) - Alma sırası sayısı, S - Denetleyici işlev bitleri, M - Değiştirici işlev bitleri, P/F - tarama sonuç biti

Şekil 8.20 Mesaj tipleri

Gönderme sıra sayımı, N(S), tek başına o anki bilgi çerçevesini belirler. Örneğin, N(S)=101 ise çerçeve 5 ile numaralandırılır. Bu çerçeve alındığında, N(S)'nin değeri incelenir ve bir önceki değerle karşılaştırılır. Eğer bir önceki değer 4 ise, mesaj sıradan alınmıştır. Eğer değer 4 değilse, bir dış gıcırdatması ve mesajın kaybının üzüntüsü vardır. Sıra sayımının modül 8 olduğuna dikkat ediniz. Böylece sıra 67012345670 ... şeklinde devam eder. Bunun sonucu olarak, eğer sekiz mesaj kaybedilirse N(S)'nin bir sonraki değeri görünürde doğru olacaktır.

Alma sıra sayımı, N(R), alıcı tarafından beklenen bir sonraki bilgi çerçevesinin numarasına karşılık gelir. Yani, vericiye N(R)-1 tane çerçevenin doğru şekilde

alındığı bilgisini verir. A'nın B ile iletişim kurduğunu ve B'ye  $N(S)=3$  ve  $N(R)=6$  ile bir bilgi çerçevesi gönderdiğini varsayalım. Bu A'nın 3 numaralı çerçeveyi gönderdiği ve B'den güven içinde 5'e kadar çerçeveleri aldığı anlamına gelir. B'den  $N(S)$  değeri 6 olan bir bilgi çerçevesini görmeyi beklediğini söylemektedir.

P/F bitinin ikili bir anlamı vardır. Ana istasyon bunu iletişim içinde olduğu uydu (slave) istasyondan bir cevap alabilmek için tarama biti olarak kullanır. Ana istasyon, her biti sıfıra eşit P bitlik (sözelimi) beş bilgi çerçevesi gönderebilir. Bir sonraki I-çerçevesi  $P=1$  ile gönderilir ki bu uydu istasyonu cevap vermeye zorlar. Ana istasyon daha sonra mesajlarının alınıp alınmadığını uydu istasyondan alınan  $N(R)$  değerini inceleyerek belirler. Uydu istasyon P/F bitini kullandığında, buna sonuç biti denir ve basitçe ana istasyona o anki yığına gönderilecek başka I-çerçevesi kalmadığını söyler.

Denetim alanının 0. ve 1. bitleri sırayla 1 ve 0 olduğunda çerçevenin denetleyici çerçeve olduğu söylenir ve bilgi alanı içermez. Denetleyici çerçevenin (veya 3-çerçevesi) bir tane 3-bit  $N(R)$  sayımı vardır ve bundan dolayı I-çerçevelerinin alındısını bildirmek için kullanılabilir. Denetim alanının iki ve üçüncü bitleri (S ile etiketli) Tablo 8.2'de görülen dört denetleyici fonksiyon için ayrılmıştır.

Tablo 8.2 S-çerçevesinin dört tipi

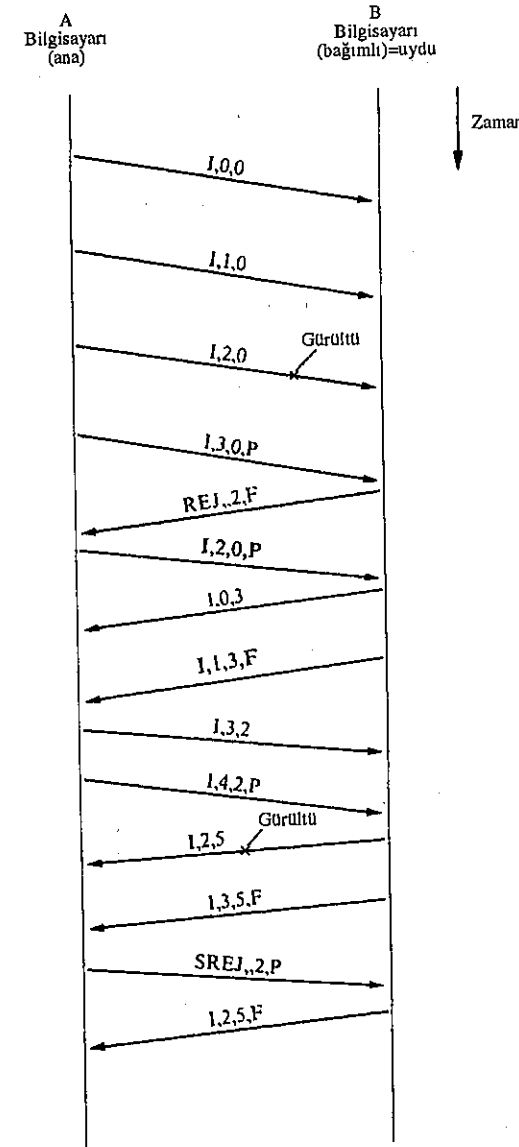
| S  | Kısaltması | Anlamı                                                                                                                                                                   |
|----|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 00 | RR         | Alıcı hazır. Bu mesajı gönderen istasyon bilgi çerçevelerini alacak durumdadır.                                                                                          |
| 01 | REJ        | Kabul etme. Bu $N(R)$ çerçeve numarası ile başlayan bütün mesajların gönderimi için istektir.                                                                            |
| 10 | RNR        | Alıcı hazır değil. Bu, meşgul durumunu gösterir ve vericiye I-çerçevesi göndermemesini söyler.                                                                           |
| 11 | SREJ       | Seçmeli kabul etme. Yukarıdaki REJ mesajı vericiye $N(R)$ ile başlayan tüm mesajları göndermesini söyler. Seçmeli kabul etme yeniden gönderilecek $N(R)$ mesajını sorar. |

Denetim alanının üçüncü türüne, numaralanmamış denir. Çünkü  $N(R)$  alanı yoktur. Çalışma modunu tanımlayarak, çağrılarını oluşturmak ve bunları bitirmek veri hattını denetlemek için kullanılır.

Şekil 8.21, yarı-dupleks modunda çalışan A (ana istasyon) ile B (uydu istasyon) arasındaki karşılıklı mesaj değişimini göstermektedir. Her çerçeve tür,  $N(S)$ ,  $N(R)$ , P/F ile tanımlanır. Burada tür I (bilgi alanı için), RR, REJ, RNR veya SREJ'dir.

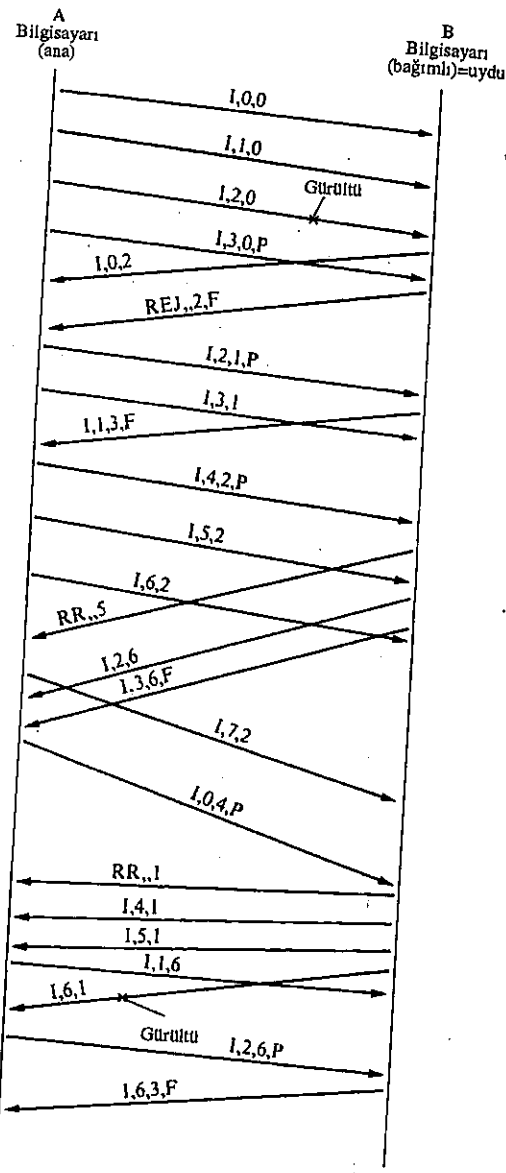
Şekil 8.22 tam-dupleks modunda çalışan ve her iki yönde de eşzamanlı olarak karşılıklı mesaj değişimine izin veren bir HDLC sisteminin çalışması gösterilmektedir.

Şimdiye dek iki tür protokolü ele aldık. İki nokta arasında bit iletimini ele alan protokole, fiziksel katman protokolü denir ve iletim yolunun elektriksel özellikleri, bitlerin zamanlamasıyla ve iletim hattının mekanik yönleriyle ilgilidir.



Şekil 8.21 HDLC yarı-dupleks iletim

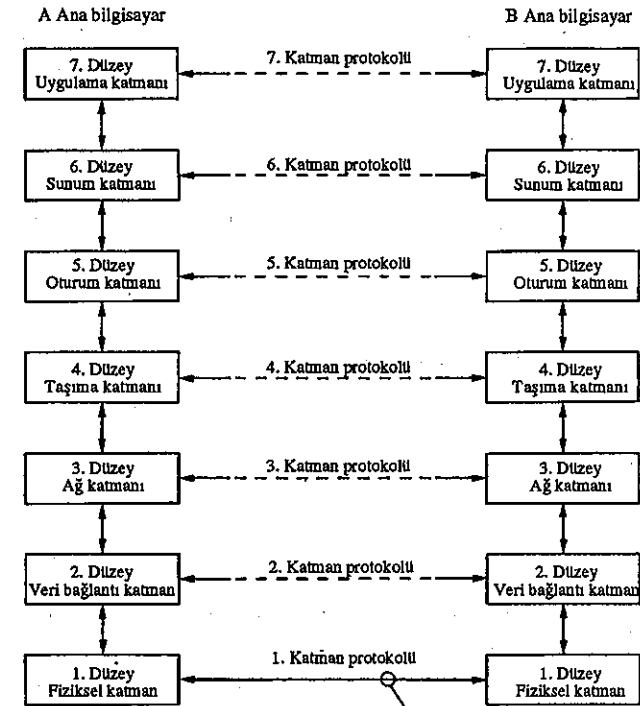
Mesajlar; tip,  $N(S)$ ,  $N(R)$ , P/F ile gösterilir. Örneğin I, 3, 0, P; 3 numaralı, 0  $N(R)$  sayımlı, cevabın gerektiğini gösteren tarama bitinin kurulmasıyla bilgi çerçevesini belirtir. A'nın 2. mesajının (yani I, 2, 0) kaybolduğuna dikkat eden. Bu nedenle A I, 3, 0, P mesajını tarama biti kurulu olarak gönderdiğinde B REJ, 2, F ile cevap verir. Bu B'nin A'dan gelen ve 2 veya daha büyük tüm mesajları reddettiğini belirtir ve F bitinin kurulmuş olması da B'nin A'ya gönderecek başka mesajı olmadığını gösterir.



Şekil 8.22 HDLC tam-dupleks iletim

Bu örnekte mesajlar A ile B arasında eşzamanlı olarak gönderilebilir. İlk yedi mesajı ele alalım. (1) A bir çerçeve gönderir; 1, 0, 0 (0 numaralı bilgi çerçevesi; A, B'den 0 numaralı bir çerçeve bekliyor) (2) A 1, 1, 0 çerçevesi gönderir (1 numaralı bilgi çerçevesi, A hala B'den 0 numaralı bir çerçeve bekliyor). (3) A 1, 2, 0 çerçevesini gönderir. Bu çerçeve gürültü yüzünden bozulur ve B tarafından doğru olarak alınmaz. (4) B 1, 0, 2 çerçevesini gönderir. (0 numaralı bilgi çerçevesi; B, A'dan 2 numaralı bir çerçeve bekliyor) A'nın 1, 4, 0 çerçevesi kaybolduğunda B hala A'dan N(S) - 2 etiketli bir çerçeve görmeyi bekliyor olduğuna dikkat edin. (5) A 1, 3, 0, P gönderir (3 numaralı bilgi çerçevesi; A, B'den 0 numaralı çerçeve bekliyor). A aynı zamanda B'yi cevap için taniyor. Bu noktada A bir önceki mesajının kaybolduğunu bilmiyordu ve A henüz B'nin son mesajını almamıştır. (6) B, A'nın taramasına bir cevap gönderir. Bu REJ, 2, F olup A'nın 2 ve üzerindeki tüm mesajlarını reddettiğini belirtir-Son bit, F, B'nin şu an artık gönderecek bir şeyi kalmadığını göstermek üzere kurulmuştur-. (7) A şimdi 1, 2, 1 HDLC tam-dupleks iletim gönderir- (2 numaralı bilgi çerçevesi; A, B'den 1 numaralı bir çerçeve görmeyi bekliyor)- Bu a'nın daha önce kaybolan 2. numaralı bilgi çerçevesinin tekrarıdır.

Bu katmanın üstünde, bir ağdaki iki bitişik nokta arasında mesaj iletimini ele alan veri-hattı katmanı protokolü yer alır. HDLC protokolü veri-hattı katmanına karşılık gelir ve kayıp çerçevelerin tekrar edilmesi amacıyla sorgulayarak hatasız iletim sağlamak üzere gereken olanakları içerir. Uluslararası Standartlar Organizasyonu (ISO) açık sistemlerin karşılıklı bağlanması (OSI) için yedi katlı bir referans modeli sunmuştur. Açık sistemler, öteki sistemlere bağlanabilen sistemlerdir. Bu modelin amacı tüm iletişim ağlarının tasarımı ve tanımı için bir çatı sağlamaktır. Şekil 8.23'de, ISO OSI referans modelini belirtilmiştir. Bu model, ne kadar karmaşık olursa olsun, tüm iletişim ağlarıyla ilgili işlemlerin üstesinden gelebilmektedir.



Şekil 8.23 ISO referans modeli

Ağ katmanı, veri-bağlantı hattının üstünde kurulur ve ağ etrafında bilgi paketleri taşır. Bir uzun-mesafe iletişim ağında arka arkaya birkaç veri-hattı kullanılabilir ve ağ bağlantısı bilgi paketlerini ağ boyunca noktadan noktaya yönlendirmekten sorumludur.

Taşıma katmanı, altındaki bağlantı hatlarını kullanarak uçtan-uca iletişim sağlar. Ağdaki bir noktadan ağ hattına verilen bir mesaj hatasız-arınmış olarak ve gönderildiği sırada ağ üzerindeki bir başka noktada görünecektir. Sistem, ağ katmanını ağ boyunca mesaj taşımak için kullanır.

Oturum katmanı, karşılıklı bağlantıları yönetmekten sorumludur. Aramaları düzenler, siler ve oturum adresini ağ adresine dönüştürebilir. Oturum adresi, kullanıcı açısından bakıldığında uzakta yeralan düğümün adresi olup, taşıma adresi ise, karşılıklı bilgi değişiminde yer alan modun gerçek adresidir.

Sunum katmanı, kullanıcıya-özü belli işlevleri yerine getirir. Örneğin bu işlevler, uygulama katında kullanılan ve sunum katmanının altındaki katmanlarda taşınan arasında veri formatının çevrilmesini içerebilir.

Uygulama katmanı, OSI modelinin en üst katmanıdır. Bu katmanın protokolu uygulama için elverişli hizmetler sunarak doğrudan son kullanıcıya hizmet eder.

### 8.3 Yerel iletişim ağları

Yerel iletişim ağlarının, günümüzdeki bilgisayarlı uygulamaların görünüşünü değiştiriyor olduğunu söylemekle konuyu abartmış olmayız. LAN'ların avantajı yüksek performansı ve düşük maliyetleridir. Böylece en küçük organizasyonlar için bile tüm bilgisayarları ile bunlara ait öteki sayısal donanımı birbirine bağlamayı mümkün kılar.

Son yirmi yıl boyunca noktadan noktaya sayısal bilginin iletimini olanaklı kılmak için yepyeni bir mühendislik dalı geliştirilmiştir. Bu yeni mühendislik bilgisayar iletişimleri, veri iletimi veya tele-işlem gibi farklı isimler almaktadır ve klasik elektronik, bilgi teorisi ve bilgisayar bilimleri arasındaki bir birlikteliği temsil eder. Çok az miktarda veri, uzun, genelde kıtalararası mesafelerde iletildiği sürece, veri iletiminin bilgisayarlı uygulama üzerindeki toplam etkisi, temelde en düşük düzeyde idi. Düşük maliyetli mikroişlemci ve mikrobilgisayarlı sistemler ile bunların bir grubun nispeten küçük (ofis, fabrika, politeknik, üniversite veya laboratuvar) alanlarda kullanımı geçmesiyle, pahalı olmayan iletişim ağlarının özelleştirilmiş bir biçimine ihtiyaç duyuldu. Bu baskıdan dolayı, coğrafi olarak yayılmış bilgis-

yarlar arasında yüksek hızda, büyük miktarda bilgi aktarımını ele alan bir veri iletim dalı ortaya çıktı. Bu dal, yerel iletişim ağları dalı olarak bilinmektedir.

Yerel bir iletişim ağının tam bir tanımı yoktur ve pek çok yazar bunları temel özelliklerine göre tanımlama eğilimindedir. LAN, verilen bir alanda bilgisayar olanaklarını birarada bağlamak için hizmet verir. Bu alan tek bir oda, bir bina veya bir bina grubu olabilir. Teesside Politeknik'teki LAN, önemli tüm bilgisayar olanaklarını ve terminalleri birbirine bağlar. Bir grup farklı LAN oluşturup bunları geçit olarak bilinen noktalarda birbirine bağlamak için kullanılan bağlantı noktaları vardır. Geçit, iki veya daha çok ayrı ağın birbirine bağlanmasıdır. Günlük terimlerle; Leicester Square metro istasyonu bir geçittir. Çünkü Northern Hattındaki metro yolcularının Piccadilly Hattına geçmesini sağlar.

LAN tek bir alana ilişkin olduğundan özel bir sistemdir. Yani, kamuya açık telefon şebekesinin bir bölümü değildir, bu yüzden de normalde kamuya açık bir ağ ile ilgili sorunlar burada sözkonusu değildir.

LAN pahalı değildir. Düşük-maliyetli sistemleri bağlamak için tasarlanmıştır ve bu yüzden de burada, pahalı teknoloji veya iletim aygıtlarının kullanımı hoş görülemez. Tipik LAN'larda, düğümler birbirine bükülü-çift tellerle veya koaksiyal kablo ile bağlanırlar. Sadece ucuz olmakla kalmayıp kurulumlarında çok az işçilik gerektirir. Bu bağlamda düğümler, ağa bağlı diğer düğümlere mesaj göndermek veya onlardan mesaj almak için ağı kullanan istasyonlardır.

LAN oldukça güvenilir bir iletişim yöntemidir. Bunun nedeni bir sistemi nispeten kısa mesafeler üzerinden bağlar ve telefon şebekesinin uzun mesafeli iletim hatlarında ortaya çıkan parazitler burada sözkonusu değildir. Bundan başka, LAN, telefon sistemi ile ilgili hata ve gürültüye-açık mekanik veya elektronik mesaj anahtarlama tekniklerini kullanmaz.

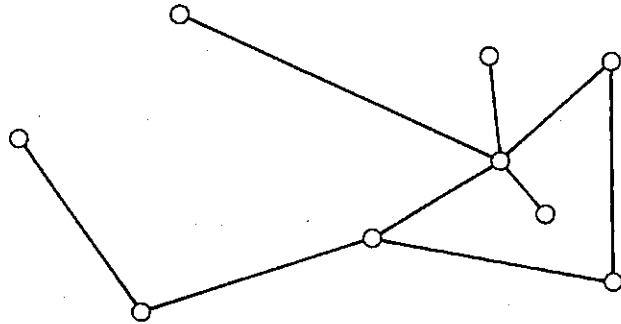
Yerel iletişim ağları çok yüksek iletim hızları sunabilirler. İletim yolu nispeten kısa olduğundan ve taşıdığı sinyallerin bant genişliğini (telefon şebekeleri gibi) sınırlayan donanımlar içermediğinden onlarca kilobit/saniye hızında veriyi kolayca iletmek mümkündür. Bazı ağlar 10 Mbit/saniye'lik veri hızında çalışır ki bu değer, neredeyse tipik bir mikrobilgisayarda noktadan noktaya verinin taşınma hızına eşittir.

#### Ağ topolojisi

Ağ topolojisi; ağın kullanıcılarının birbirine bağlanma biçimini tanımlar. Bir

LAN'de kullanmaya elverişli dört basit topoloji vardır. Bunlar: Kısıtlamasız topoloji, yıldız ağı, yol ve halka.

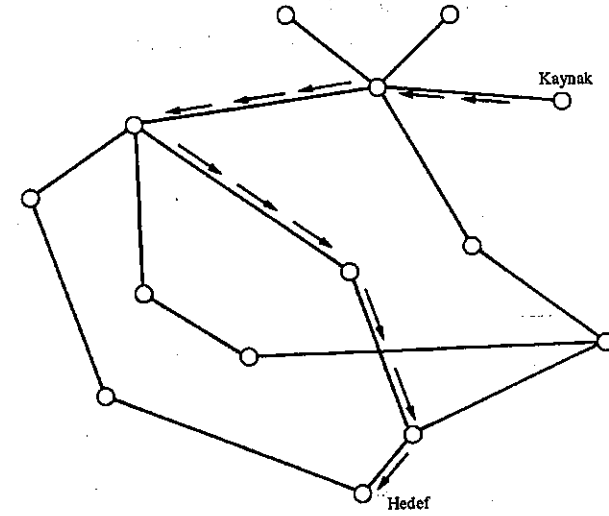
**Kısıtlamasız topoloji:** En genel topoloji, Şekil 8.24(a)'deki kısıtlamasız ağıdır. Düğümler birbirine istenilen şekilde bağlanır. Avantajı bir grup düğüm arasındaki yoğun trafiğin yol açtığı tıkanıklıkları azaltmak için ek hatların bağlanabilmesidir. Daha fazla düğümler ve hatlar, varolan sistemin donanımını olumsuz yönde etkilemeden kolayca eklenebilir. Pek çok ülkede karayolu yol ağları, yeni yolların istenen yer ve zamanda eklenmesiyle, kısıtlamasız topolojiye bir örnektir.



Şekil 8.24 (a) Kısıtlamasız topoloji

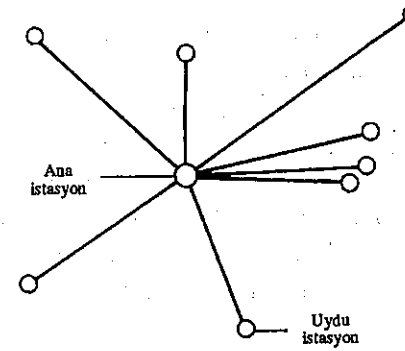
Kısıtlamasız topolojinin dezavantajı, her düğümde hedefine doğru giden mesajı en iyi şekilde yönlendirmek için bir karar verilmesinin zorunlu oluşudur. Yol sistemiyle benzeştirme yapılırsa, sürücünün bir yerden başka bir yere gidebilmesi için elinde bir yol haritası bulunması zorunludur. Mesajlar bir düğümden, o düğüme bağlı tüm düğümlere aynı anda gönderilemez çünkü bu mesajın her düğümde çoğaltılarak ağ boyunca sonsuz sayıya çıkmasına yol açar. Bunun yerine her düğüm kendisine ait bir yol-haritasına sahip olmalı ve mesajın hedefine hangi yoldan iletileceğiyle ilgili karar vermelidir. Bu işlem, yönlendirme algoritmalarıyla ilgili çalışmalar yapılması gibi ek bir yük getirir. Bundan başka, ağa yeni bir hat veya düğüm eklendiğinde, yönlendirme bilgisinin her düğüm üzerinde değiştirilmesi gerekir. Şekil 8.24 (b)'de bir mesajın kısıtlamasız bir topoloji boyunca nasıl yönlendirilebileceği görülmektedir.

**Yıldız ağı:** Şekil 8.25'de, tüm mesajların, kaynaktan hedefe doğru, merkezi bir düğümden yönlendirilmesiyle, yıldız ağının, düğümlerin yönlendirme kararı alma gereksinimini nasıl ortadan kaldırdığı gösterilmiştir. Yıldızın basit bir topolojisi vardır ve ağın fiziksel topolojisi mantıksal topolojiyle uyması halinde avantajlı



Şekil 8.24 (b) Kısıtlamasız bir topoloji içinde bir mesajın yönlendirilmesi

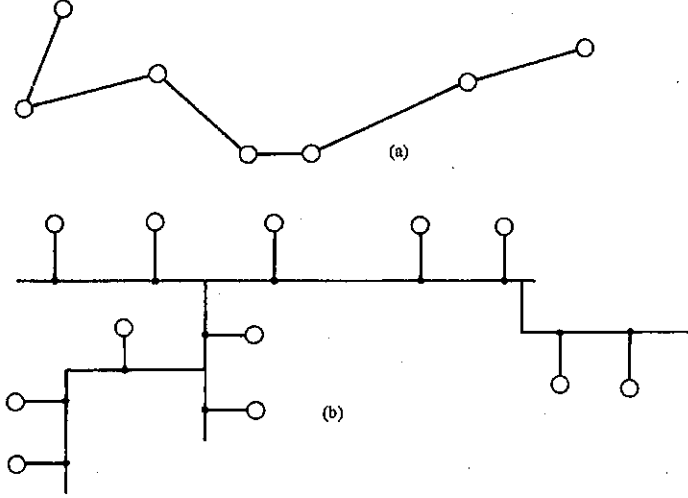
olmaktadır. Açıkçası, düğümlerin dağıtıldığı öyle durumlar vardır ki, bazı düğümlerle merkezi düğüm arasındaki hatlar ekonomik olarak uygulanamazlar.



Şekil 8.25 Yıldız biçimli ağ

Yıldız ağının iki belirgin dezavantajı vardır. Tüm mesajlar merkezi düğümden geçtiğinden, merkezi düğümde ortaya çıkacak bir sorun ağın çalışmasını engelleyecektir. Öteki ağlar; eğer ağın bir kısmı arızalanırsa, dar fakat kullanılabilir bir hizmet sunabilirler. Bundan başka, tüm trafik merkezi düğümden geçtiğinden, bu düğümün bağlı olduğu tüm düğümleri idare edebilmek için yeterince yüksek hızda çalışabiliyor olması gerekir.

**Yol:** Yol topolojisi Şekil 8.26(a) ve (b)'de belirtilmiştir. Yol (ve aynı zamanda halka) hem özel amaçlı merkezi düğümü hem de her bir düğümün yönlendirme kararları alma gerekliliğini ortadan kaldırarak ağı karmaşıklığını en aza indiren çabasıdır.

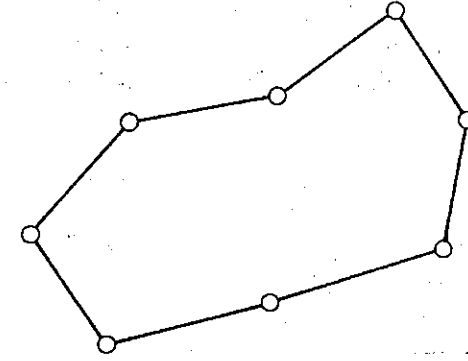


Şekil 8.26 (a) Yol Basit yol (b) Daha genel bir yol biçimi

Yol topolojisinde tüm düğümler ortak bir veri anayoluna bağlanır. Bu anayol, düğümleri bağlayan tek bir yol olabilir ya da her bir yolu bağlayan kollarıyla birlikte bir yol grubu olabilir. Bu tür bir topolojiye köksüz ağaç denir. Mesaj bir düğüm tarafından yola çıkarıldığında, ileri doğru tüm yönlerde akar ve sonunda ağdaki tüm noktalara ulaşmış olur. Yolun, bir topolojik bir de pratik kısıtlaması vardır. İki nokta arasında sadece bir yol bulunur, böyle olmasaydı, mesajlar bir döngü etrafında sonsuza kadar dönebilecekti. Pratik sınırlama ise yolun bir uçtan öteki uca normalde maksimum bir uzaklığın üzerine çıkılamamasıdır.

Tasarımcılar tarafından karşılaşılan temel yol topolojisi problemi; yolu aynı anda kullanmak isteyen bir grup düğümün nasıl ele alınacağı problemidir. Buna yol çakışması denir ve daha sonra incelenecektir.

**Halka:** Şekil 8.27 halka topolojisini göstermektedir. Burada düğümler bir halka biçiminde birbirine bağlanmışlardır. Yol topolojisi gibi bu da, merkezi olmayan bir yapıdadır, çünkü halkayı denetlemek için merkezi bir düğüme gerek yoktur. Düğümler mesajları, hemen yanındaki komşusundan alır ve diğer komşusuna gönderir. Mesajlar, halka etrafında tek bir yönde akarlar.



Şekil 8.27 Halka

Her bir düğüme, yalnızca yönlendirme olanakları yerleştirilmiş olduğundan kendisine yönlendirilmiş mesajları tanıyabilmektedir. Halkada, yol topolojisindeki gibi yol çakışmasına ilişkin bir problem yoktur. Fakat burada da, yeni bir mesajın, halkadaki bir düğümün mevcut trafik akışına nasıl ekleneceği sorunu söz konusudur.

LAN'ların ilk dönemlerinde, halkalar, arızaya açık bir topoloji olarak görülürdü. Kopmuş bir hat mesajların halkayı tamamlamasını imkansız hale getirir. Bu durumda artık, her bir düğüm arasında iki hattın bulunduğu bir grup çift halka yapıları vardır. Eğer bir hat koparsa halkanın kendini yeniden düzenleyerek bu kopuk bağlantıyı atlaması mümkündür.

#### Yollardaki çakışma denetimi

Çakışmalı bir ağda, mesaj iletmek isteyen düğüm sadece ileri çıkar ve mesajını yola koyar. Bir düğümün ne zaman iletebileceği ile ilgili bir denetim olmadığından iki veya daha fazla düğümün aynı anda mesaj göndermeye kalkışmasını engelleyebilecek herhangi bir şey de yoktur. Eğer böyle bir durum ortaya çıkarsa, iletilmekte olan tüm mesajların toparlanamaz şekilde sırası karışır ve mesajlar kaybedilir. Çakışma denetiminin en basit biçimi göndericilere mesajlarını yeniden gönderme olanağı sağlamaktır. Ne yazık ki çakışan düğümler, mesajları tekrar gönderdikçe ve mesaj sırasında karışma söz konusu olduğu sürece, bu tür bir düzenleme çalışmayacaktır. Burada dikkat edilecek iki şey vardır. İki veya daha fazla mesajın birbiri üzerine binmesinden dolayı kaybolmasının teknik adı çarpışmadır. Bu problem, iki kişinin aynı döner kapıya birlikte girmeye çalışmasıyla aynıdır: İçeriye aynı anda giremezler, bir adım geriye gelir ve tekrar denemeye çalışırlar; bu da çarpışmaya neden olur. Böylece tekrar geri adım atarlar, tekrar girmeye çalışırlar ve sonuçta yine çarpışırlar...

Çarpışmayı algılayacak daha iyi bir strateji, çerçeveyi yeniden göndermeyi denmeden önce gerilemek veya rastgele bir süre beklemektir. Bu şartlar altında çarpışma düğümlerin, tam olarak aynı anda gönderimde bulunmaları düşük bir olasılıktır. Dönen kapı örneğinde, iki kişi çarpıştığında, derhal zar atarlar ve tekrar denemeden önce zarlar gelene kadar beklerler. Aynı sayıyı atma olasılıkları çok azdır, böylece kapıdan teker teker geçebilirler. Bu çarpışma denetimi biçiminde çalışan ağlar yoğun trafik için çok uygundur. Yani, ortalama trafik (yolun maksimum kapasitesinden) çok düşük olduğu sürece düzen çalışır, trafik yoğunluğu artarsa, öyle bir nokta gelir ki bu çarpışmalar mesaj tekrarlamalarını, bu da daha fazla çarpışmayı ve daha fazla tekrarlamayı üretir ve sonunda sistem çöker.

Çarpışma denetiminin daha iyi bir biçimi, düğüme, mesajını göndermeyi denmeden önce yolu dinleme imkanı sağlamaktır. Açıkça görüldüğü gibi, eğer bir düğüm halihazırda bir mesaj gönderiyorsa, öteki düğümler 'burunlarını' sokmayacaklardır. Artık bir çarpışma, ancak ve ancak iki düğümün, aynı anda göndermeyi denemesi halinde ortaya çıkacaktır. Düğüm bir kez göndermeye başladığında ve sinyali ağ boyunca çoğalarak ilerlediğine, diğer hiçbir düğüm araya giremez. Neredeyse tüm sistemler için bu tehlikeli aralık, yani bir mesajın ağın bir ucundan ötekine yayılma süresi, çok küçüktür ve mesaj iletim süresinin çok az bir oranıdır.

Bu düzenlemedeki bir başka değişiklik de vericilere gönderme boyunca yolu dinleme imkanı sağlamaktır. Herhangi bir vericinin, yolun boş olduğunu sanarak mesaj göndermeye başladığı ve aynı anda bir başka vericinin de benzer işlemi yaptığı durumu göz önüne alalım. Kısa bir süre sonra her iki verici de yolun kullanımda olduğunu fark edecek ve mesajlarını iptal edeceklerdir. Bu şekilde çarpışma etkisi azaltılır çünkü vericiler çarpışmayı algıladıkları anda iletim durdururlar. Gönderirken-dinle-mekanizmasının olmaması durumunda, çarpışma dolaylı olarak mesaja karşılık bir alındı bildiriminin gelmemesiyle algılanmayacaktır.

En eski LAN'lardan biri Hawaii Üniversitesi'ndeki ALOHA ağı idi ve Üniversitenin odalarına dağılmış bir grup istasyon arasında iletişim sağlamak üzere tasarlanmıştı. ALOHA bir yol ağı olmayıp yıldız ağı idi ve merkezinde, düğümler arasındaki tüm mesajların üzerinden geçmesi gereken bir ana istasyon vardı. O zamandan, ALOHA'da yapılan ilk çalışmalar ve çarpışma denetimini için kullanılan teknikler bugünkü yol-merkezli LAN'ların ortaya çıkmasına yol açtı. Adaların dağınık olmasından dolayı, iletişim hattı olarak VHF bandı seçildi. Ana-uydu iletişimi için bir frekans ayrıldı (burada sadece bir verici ve birçok alıcı olduğundan çarpışma problemi yoktur). Bir diğer frekans tüm uydu istasyonlar tarafından paylaşmakta ve ana istasyona mesajlarını iletmek için kullanılır. Çarpışma prob-

leminin doğduğu yer işte burasıdır çünkü iki veya daha fazla uydu istasyon aynı anda göndermeye başlayabilirler. Gönderme cihazı radyo yayını olduğu için çarpışma oluşacaktır.

Saf ALOHA, 32-bitlik başlık, 16-bitlik başlık içeren eşlik denetimi ve bir veri eşlik denetiminin izlediği 80 bayta kadar veriden oluşan paketler kullanır. Her bir paketi göndermek yaklaşık 73 milisaniye alır ve çoğalma süresi, (73 msn'lik paket ile karşılaştırıldığında) ihmal edilebilir. Eğer bir istasyon sabit bir süre içinde merkezi düğümden alındı bildirimi almazsa, paketinin kaybolduğunu varsayar ve rastgele bir süre bekledikten sonra yeniden bir paket gönderir. ALOHA göndermeden önce veya gönderme boyunca kanalı dinlemez ve bu yüzden de olası en kaba çarpışma denetim biçimini gösterimler.

Bir paket göndermek isteyen istasyonun olasılığının bir Poisson dağılım eğrisine sahip olduğunu varsayarak, bu sistemin maksimum veriminin, maksimum kanal kapasitesinin % 18'ine ulaştığı gösterilebilir. Bunu daha açık hale getirmek için yine döner kapımızı ele alalım. Eğer insanlar kapıya rastgele girmeye çalışırlarsa ve kapıların maksimum kapasitesi, tüm insanların (çarpışmadan daha nazik bir davranışla) birbirini izlemesi halinde geçiş kapasitesi saatte 1000 kişi ise, insanlar rastgele bir süreyle bekleyeceklerinden, maksimum verim de saatte 180 kişiyi aşmayacaktır.

Kanal süresini, maksimum paket uzunluğuna eşit bir şekilde sabit aralıklara bölerek (bu nedenle aralıklı ALOHA denir) ve bir paketin iletimine, sadece zaman aralığının başlangıcında izin vererek, çarpışma şansı büyük ölçüde azaltılır ve maksimum verimlilik % 36'ya çıkarılır. Eğer paketler rastgele gönderilirse, bir paketin başlangıcı bir diğerinin bitimi ile çakışabilir ve iki paket uzunluğuna eşit bir zaman periyodu kaybedilir. Tüm paketler sabit zaman aralıklarında gönderilirse, çarpışma komşu iki aralığı etkileyemez ve sadece bir paket uzunluğuna eşit bir süre kaybedilir. Elbette ki bu düzenleme tüm vericilerin zaman aralıklarına uyabilmek için mesajlarının saatini saptamak üzere hassas bir saate erişebilmelerini gerektirmektedir.

ALOHA'nın en popüler türevlerinden biri Ethernet'tir ve DEC, Xerox, Intel tarafından desteklenmektedir. Bu tür bir LAN, fazkodlanmış verinin 10 Mbit/saniyede iletildiği bir taban-bantlı koaksiyel kablo kullanır. Taban-bandı terimi sayısal verinin modeme gerek duymadan doğrudan iletilmesi anlamına gelir. Ethernet tarafından benimsenen çarpışma mekanizmasına 'taşıyıcıya duyarlı çoklu erişim (CSMA)' denir. Ethernet istasyonu bir paket göndermek istediğinde, önce yolun durumunu dinler, yol kullanımdaysa boşalana kadar bekler. Ethernet terminolojisinde buna erteleme denir. Başarılı bir gönderme, istasyon bir kez göndermeye

başladıktan sonra kanalı ele geçirir ve ancak, ağın uçtan uca dolaşım süresine bir gecikmeden sonra çarpışmasız olarak garanti edilmiş olur.

Paketin ağ boyunca yayınımindan önce bir 'çarpışma penceresi' oluşur. Burada istasyon birbirinden habersiz olarak göndermeye başlayabilirler. Bir istasyon kendi paketinin bir başka paket tarafından bozulduğunu anlaması halinde, parazit paketi göndererek 'tıkanıklığı' zorlamaya çalışır. Eğer gönderiyi anında durdursaydı, öteki verici çarpışmayı fark etmeyebilirdi. Çarpışma çok daha sonra dolaylı olarak gönderilen çerçevenin bir kısmını oluşturan hata-algılama kodu ile algılanır. Bu işlem verimsizdir ve zaman kaybına yol açar. Küçük bir parazit paketi göndermek tüm dinleyicilere çarpışmayı duyurur. Parazit gönderildikten sonra rastgele bir gecikmeden sonra bir başka deneme yapılır. Eğer tekrarlı denemeler başarılı olamazsa, gönderici meşgul kanala uymaya çalışarak gecikme süresini artırır.

### Halka

Halka ağı tüm istasyonları birbirine, sürekli bir döngü biçiminde bağlar. Yol ağındaki istasyonlarından farklı olarak, (bunlar kendileri için olmadığı sürece, yol üzerindeki veriyi pasif olarak dinler) halkanın istasyonları tüm veri aktarımlarında aktif bir rol alırlar. Basitçe, gelen veriyi alırken, bunu test etmeli ve kendileri için saklayacaklarına veya bir sonraki komşularına göndereceklerine karar vermelidir. Genelde, halkaya hangi istasyonun erişebileceğini belirlemek için kullanılan mekanizmalar yol için olanlardan çok farklılık gösterir. Üç popüler denetleme tekniği şunlardır: token geçişi, kaydedici ekleme ve aralıklı halkalar. Yol ağındaki şekliyle çarpışma denetimi halkalarda bulunmaz. Birinin buna en az bir istisnai durum olduğunu belirteceğine bahse girerim!

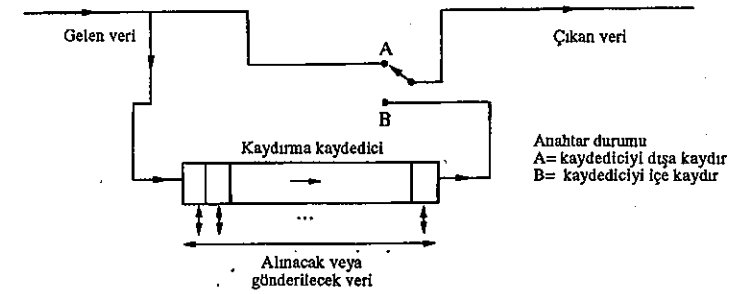
### Token halkaları

LAN'larla ilgili büyüleyici şeylerden biri de soyut gibi görünen teknolojik yönlerinden birçoğunun günlük yaşantı ile ilişkilendirilebilmesidir. Klasik demiryolu problemi tek bir hat üzerinde trenlerin denetimidir. Eğer iki tren tek bir hattın uçlarından zıt yönde hareket ederse çarpışma meydana gelir. Bu çarpışmalar elbette veri ağlarındakilerden daha zararlı olurlar. Bir çözüm hat boyunca metal bir halka veya token bulundurmaktır. Sadece token'a sahip olma durumundaki sürücü hattı kullanma hakkına sahiptir. Eğer sürücü hattın bir ucuna gelirse ve token arada değilse beklemelidir; eğer aradaysa onu alır, hatta girer ve hattın diğer ucunda token'ı bırakır. Token ondayken hatta, önünden veya arkasından kimse giremez. Token'ı bıraktıktan sonra, bir başka tren onu alabilir ve hatta geri döner.

Bir token halkada özel bir bit deseni (token) halka boyunca istasyondan istasyona geçirilir. Token'ı o an elinde bulunduran istasyon, isterse veriyi iletebilecek olan istasyondur. Eğer veri gönderme hakkını kullanmak istemiyorsa, token halka boyunca geçirir. Örneğin, token'ın özel desen olarak 11111111 olduğunu ve bu deseni eşsiz tutmak için HDLC'de olduğu gibi sıfır-doldurmalı olduğunu varsayalım. Göndermek isteyen bir halka istasyonu gelen trafiği gözlemler. Yedi tane biti algıladığında TOKEN'in sonuncu bitini ters çevirir ve bu haliyle gönderir. Böylece, 'bağlayıcı' (11111110) bir desen halkadan geçer. Bağlayıcı, sekizinci 1'in gönderilmesinden böylece token'ın geçmesini önleyebilmek için yaratılır. Artık istasyon verisini gönderebilir. Verisini gönderdikten sonra, halkaya yeni bir token gönderir. Sadece bir token olduğu sürece halkada çarpışma oluşmaz, elbette ki bir istasyon dışarıya kapalı olup ikinci bir token göndermezse... Pratikte, gerçek bir sistem biraz daha karmaşıktır, çünkü düzenlemeler kayıp token'leri de ele alacak şekilde olmalıdır.

### Kaydedici ekleme halkaları

Kaydedici ekleme halkası kavramı Şekil 8.28'de gösterilmiştir. Gönderilecek



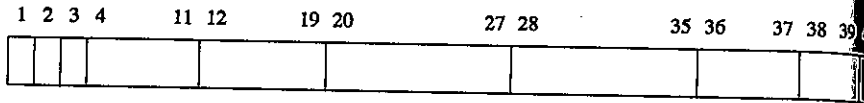
Şekil 8.28 Kaydedici ekleme halkaları

mesaj önce bir kaydırmalı kaydedici yüklenir. Bu kaydırmalı kaydedici, başlangıçta halkanın bir parçası değildir. İstasyon halkanın boş veya iki ayrı mesaj arasında olduğuna dikkat ederse çevrimi kırar ve kaydırmalı kaydedicisini araya ekler. Gönderilecek mesaj; kaydırmalı kaydediciden, verinin halka etrafındaki taşınma hızına eşit bir hızla saatlenir. Mesaj, kaydırmalı kaydediciyi bir uçtan terk ederken, öteki uçtan halkadaki bir veri içeri kaydırılır. Aslında kaydedici, halkanın boyunu uzatmıştır. Kaydırmalı kaydedici artık halkanın bir parçası olduğundan, ne ortadan kaldırılabilir ne de öteki mesajları göndermek için kullanılabilir.

Mesaj, halka etrafında hareket ettiğinde, sonuçta, başlangıçtaki kaydediciye ulaşır. Bu anda kaydedici devreden dışarı çıkarılabilir ve mesajılır. Eğer mesaj halka etrafındaki dolaşımı sırasında zarara uğrarsa, gönderilmeli kaydırmalı kaydedici dışarı alınmadan önce halka boş olana kadar beklemelidir.

#### Aralıklı halkalar

Aralıklı bir halkada bir veya daha çok paket taşıyıcısı (aralık) halka etrafında geçerilir. Taşıyıcılar, (demiryolu taşıyıcısı gibi) fiziksel bir nesne değildir. Sadece bitlerin üzerinden geçtiği düğümler tarafından değiştirilebilen özel bir paketlerdir. Şimdiye dek tüm paketlerin (yani çerçevelerin) üretildiğini ya da alındığını fakat değiştirilmediğini gördük. Her taşıyıcının bir başlığı, bir kuyruğu ve bir boş/dolu biti vardır. Boş bir taşıyıcı veri göndermek isteyen bir istasyondan geçişinde, istasyon taşıyıcıyı veri ile doldurur ve boş/dolu bitini kurarak taşıyıcı dolu olduğunu işaretler. Hedefinde veri taşıyıcıdan çıkarılır ve dolu bayrağı silinir. Cambridge halkası bu temel üzerine çalışır.



- 1: Ön veya ara çerçeveleme biti (daima 1)
- 2: Dolu/boş biti (1=dolu)
- 3: Ekran denetim biti
- 4-11: Varış adresi
- 12-19: Kaynak adresi
- 20-27: 1. Veri baytı
- 28-35: 2. Veri baytı
- 36: A tip biti
- 37: B tip biti
- 38-39

#### Cevap bitleri

| 38 | 39 | Cevap                                   |
|----|----|-----------------------------------------|
| 0  | 0  | MEŞGUL - hedef modu paketle ilgilenemez |
| 0  | 1  | KABUL - hedef modu paketi kabul eder    |
| 1  | 0  | SEÇİLMEMİŞ                              |
| 1  | 1  | DİKKATE ALMA - paketi okuyan düğüm yok  |

40: Çift eşlik biti

Şekil 8. Cambridge halkası üzerindeki verinin formatı

Şekil 8.29, Cambridge halkası tarafından kullanılan paketin formatını göstermektedir. İlk bit senkronizasyon bitidir ve paketin başlangıcını belirtir. Asenkron sistemlerdeki başlangıç biti gibi. İkinci bit boş/dolu bitidir. Üçüncü bit bağımsız istasyon tarafından kullanılır ve gözlemci adını alır. Halka yapısı herkese eşit şans tanır fakat açılışta boş paketler üretmek ve bozulmuş paketlere pusu kurmak için özel bir istasyona, gözlemciye gereksinim duyar.

Pakette; 8-bit bir varış adresi ve bunu takiben 16-bit bir veri alanı bulunur. Veri alanını takiben iki-bitlik bir denetim alanı ve tek bir eşlik biti yer alır. HDLC formatı veya Ethernet Formatı ile karşılaştırıldığında Cambridge Halkası etrafında akan paketler nisbeten kabardır.

Çeşitli yerel iletişim ağlarının karşılaştırmalı avantaj ve dezavantajlarıyla ilgili birçok yayın vardır. Bazıları Ethernet'i bazıları da token halkasını tercih eder. Bazıları yolu tercih ederler, çünkü genişletilmesi kolaydır. Ötekileri bunu kullanmayacaklarını çünkü teorik olarak mesaj, sonsuz sayıda çarpışmadan etkilendiğinden mesaj alındısının sanki bir süre sonunda garanti edilemeyeceğini söyler. Diğerleri (daha akıllı) mühendisler LAN'ın pratikteki biçiminin, kullanıcı programı ile LAN (yani işletim sistemi) arasındaki arabirimin normalde pek çok ilkel LAN biçimlerinden daha küçük olduğu sürece çoğu zaman önemli olmadığını söyler.

İşletim sistemiyle (çoğu zaman OS olarak kısaltılır) bilgisayar arasındaki ilişki, orkestra ile orkestrası arasındaki ilişkiye benzer. Orkestranın şefi uluslararası bir üne sahiptir; televizyondaki tüm sohbet programlarına davet edilir ve en yüksek toplu ödülleri ona verilir. Fakat yine de konsere tek bir nota eklemeyebilir. Şefin, orkestra için ne denli önemi olduğu çok iyi bilinir. Diğer müzisyenleri o koordine eder. Ayrıca elemanlarının güçlü yönlerini ve zayıflıklarını bilir ve böylece toplu performanslarını optimize edecek şekilde çalıştırır.

İşletim sistemi, çoğu zaman bir bilgisayar sistemindeki yazılımın en önemli parçasıdır; fakat hiçbir problemi çözmez ve faydalı hesaplamalar yapmaz. Rolü bilgisayarın (yazılımı da içine alacak şekilde) tüm işlevsel bölümlerinin çalışmasını koordine etmek ve böylece sistemin verimliliğini en üst düzeye çıkarmaktır. Burada verimlilik, MİB'in pratikte kullanıcı programlarını yürütme süresi olarak alınabilir. Eğer işletim sisteminin, sistemdeki verimsizliği ortadan kaldırmak üzere tasarlandığını söylersek daha doğru ifade etmiş oluruz. Örneğin eğer bir program bir listeleme için satır yazıcıyı çağırırsa, yazıcı çalışırken MİB durdurulur. İşletim sistemi, normalde araya girerek yazıcının işini bitirmesini beklerken bir MİB'e bir başka şey verir. İşletim sisteminin ikinci rolü kullanıcı ile bilgisayar arasında bir arabirim gibi işlev görmektir. Kullanıcı, İş Denetim Dili (JCL) sayesinde, sistemin çalışma ayrıntılarını bilmeye gerek duymaksızın, bilgisayardan (örneğin bir programı düzenlemek, yükleyip yürütmek gibi) bir grup görevi yerine getirmesini isteyebilir.

Aslında kullanıcının bakış açısından, işletim sistemi kusursuz bürokrat gibi davranmalıdır. Etkin ve yardımcı olmalı, ve (tüm iyi bürokratlar gibi) arka planda kalmalıdır. Örneğin, zayıf bir işletim sistemi, bir dosya biçimlendirilmek istendiğinde, ERROR 53 (HATA 53) karşılığını verebilir. Programcı bu durumda HATA 53'e bakmak için bir işletim sistemi el kitabı bulmalıdır. Bunların tümü de zaman kaybıdır. Bunun nedeni, son kullanıcı hata mesajlarının açıklandığı sayfayı koparmıştır çünkü sürekli oraya başvurmaktan artık gına gelmiştir.

Gerçekten iyi bir işletim sistemi ise şöyle cevap verebilirdi: 'Merhaba günışığı! Üzgünüm ama disklerim tamamen dolu. Bak ne diyeceğim, programının birçok yedek kopyası olduğuna dikkat ettim ve eğer birkaç tanesini silersen sanırım dosyana bir yer bulabiliriz. İyi günler.'

Donanıma giriş dersinde, işletim sistemleriyle ilgili bir bölüm bulunmasının biraz

ilgisiz olduğu düşünülebilir. Bu konuyu buraya almamın iki sebebi vardır. İlki, işletim sisteminin donanım ile sıkı sıkıya ilişkili olmasıdır. Donanımın çalışmasını denetler ve aynı zamanda donanım olanaklarını kullanıcı programlarının ortak kullanımına sunar. İkinci olarak, bilgisayar donanımı dersi alan pek çok öğrenci, bununla paralel üst-düzey dilleri, veri işleme ve veri yapılarıyla ilgili dersleri de alırsa, bu konuların donanıma ilişkin derslerde ele alınmasına gerek kalmayacaktır. Öte yandan işletim sistemlerinin formal biçimde ele alınışı, kurs içinde çok ileri bir aşamada öğrencilerin karşısına çıkacaktır. Bu yüzden burada, işletim sistemleriyle ilgili kısa bir giriş ve özet sunuyorum.

Tüm bilgisayarların bir işletim sistemi yoktur. Bilgisayarın bir denetleyici olduğu veya tek bir göreve ayrılmış olduğu yerlerde işletim sistemine ihtiyaç yoktur. Görev basitçe bellekte (çoğu zaman ROM'da) makine kodu biçiminde tutulur. Normalde bir işletim sistemi tarafından yerine getirilen işlevlere gerek duyulan her seferde bunlar görevin içine alınırlar. Sıralamada, özel denetleyici ile genel amaçlı bilgisayarların arasında bir yere konabilen kişisel bilgisayarlarda bir grup işletim sistemi olanağı üst-düzey dilleriyle birlikte gelir.

Genel-amaçlı işletim sistemleri bir zamanlar sadece büyük bilgisayarlarda bulunuyordu fakat günümüzde bazı kişisel bilgisayarlarda da bulunmaktadır. Bazı bilgisayar üreticileri işletim sistemlerine 'yönetici' derken, ötekiler 'monitör' veya 'gözetmen' olarak adlandırırlar. İşletim sisteminin ayrıntılarına inmeden önce bir işletim sisteminin gerçekte ne yaptığı ile ilgili bir örnek sunmak gerekmektedir.

Bir öğrenci bir programı düzenleyip sonra da çalıştırmak istiyor. Birçok 16-bitlik mikroişlemci ile birkaç 8-bitlik makinede gelişmiş işletim sistemleri olsa da bilgisayarın bir anabilgisayar veya mini olduğunu varsayacağız. Bir terminale gider ve sisteme girer. Kullanıcı, sisteme girişiyle varlığını bilgisayara bildirmiş (hizmet için bir istekte bulunmuş) olur ve aynı zamanda işletim sistemi onun kullanıcı kimliğini ve (gerekliyorsa) şifresini girmesini isteyebilir. Bu, yetkisiz kişilerin sisteme erişmesini engeller ve tüketilen sistem kaynaklarının kaydını tutarak bir muhasebeci gibi işlev görür.

Bilgisayarın birkaç terminali olduğunu varsayarsak, işletim sisteminin bir başka işlevi de, kullanıcıların her birine çalışma zamanı ve bellek alanı ayırmaktır. İşletim sistemi aynı zamanda bir kullanıcının, bir başka kullanıcının programlarına kazara (veya kötü niyetle) müdahale etmesine engel olmalıdır.

Öğrenci programını düzenlemek istemektedir. Böylece düzenleyiciyi ve düzenlenecek programını yükler ve bilgisayara yeni versiyonun adının ne olacağını söyler.

Örneğin; eğer programının adı DENKLEMLER.3 ise şu komutu girebilir: EDIT, DENKLEMLER.3, DENKLEMLER.4. Bu bir iş-denetim dili örneğidir ve öğrencinin sistemin iç çalışmasıyla ilgili ne kadar az şey bilmesi gerektiğini gösterir. İşletim sistemi daha sonra düzenleyicinin yerini saptar, onu diskten IAS'ye (doğrudan erişimli belleğine) aktarır, DENKLEMLER.3'ü yükler, diski DENKLEMLER.4 denen düzenlenmiş versiyonu almak için hazırlar. Eğer herhangi bir anda bu işlerden biri tamamlanamazsa, işletim sistemi bunu kullanıcıya bildirir. Düzenleme bittiğinde, öğrenci işini çalıştırmak üzere bilgisayara verir.

Program kaynak-kodu biçiminde olduğundan, yürütülmeden önce amaç koduna çevrilmelidir. Kaynak kodu; bir üst-düzey veya assembly-düzeyi dillerde programcı tarafından yazıldığı biçimiyle bir programa karşılık gelir. Amaç kodu, bilgisayar tarafından yürütülecek gerçek ikili bilgidir. İşletim sistemi tüm bu görevleri şu komut aracılığıyla yerine getirir:

PASCAL RUN, DENKLEMLER.4

Artık derlemeyi yapabilmek için PASCAL derleyicisinin ve DENKLEMLER.4 kaynak programının yerini bulmak, DENKLEMLER.4'ün derlenmiş versiyonunu yükleyip yürütmek işletim sisteminin görevidir. Bundan başka, işletim sistemi programının bellekte nereye yerleşmesi gerektiğine karar vermeli ve buna bağlı olarak tüm komutların ve verilerin adreslerini belirlemelidir.

Oturumun sonunda öğrenci, sistemden çıkarak terminali ve kullanmakta olduğu kaynakları serbest bırakır. İşletim sistemi tüm bu kaynakları havuzuna geri gönderir ve öğrencinin işiyle ilgili gerekli hesaplama bilgilerini ve istatistikleri hazırlar. Bu istatistikler, öğrencinin terminale bağlı olduğu süreyi, gerçekte de kullanılan MIB çalışma zamanını, ayrılan disk alanı büyüklüğünü ve üretilen çıkış satır sayısını içerir.

#### *İşletim sistemi tipleri*

İşletim sistemleri dört gruba ayrılabilir: tek-kullanıcı, toplu-modu, istek-modu ve gerçek-zamanlı. Gerçek bir işletim sistemi, iki veya daha çok gruba ait özelliklere sahip olabilir. Tek-kullanıcı işletim sistemi en ilkel türdür ve birçok mikroişlemci sistemlerinde bulunur (örneğin CP/M, Flex). Her seferinde yalnızca bir kullanıcının sisteme erişimine izin verir. Genelde; ya çok az kaynak kullanımı sağlar, ya da hiç sağlamaz. İşletim sistemlerinin en eski biçimleri toplu-modda çalışır. Yürütülecek işlemler bilgisayarlara, genelde delikli kartlar biçiminde girilirler. Her bir kullanıcı işi, işletim sisteminin hangi olanaklara gerek duyacağını söyleyen belli sayıda

iş-denetim kartlarıyla başlar. İşletim sistemi daha sonra işleri istedikleri kaynağa ve öncelik sırasına göre zamanlar ve sonuçta çıkışları genellikle bir satır yazıcıdan üretir. Toplu-mod ile kuru temizleme sistemi arasında bir benzetme kurulabilir. Giysiler temizlendikten sonraki bir tarihte teslim edilir ve işlemlere yeniden başlanır. Toplu-mod sistemlerinin dezavantajı, iş verip/alma süresinin uzunluğudur. Bir çıkış için beş saat bekleyip sonunda kartların birindeki basit bir hatadan dolayı işin çalışmamış olduğunu görmek oldukça sinir bozucudur.

İstek-modunda her kullanıcı, kendi işlerini çalıştırabileceği terminale erişir. Bu, toplu-moda göre büyük bir ilerlemedir çünkü kullanıcı her bir adımı, bir sonrakine geçmeden önce tamamlayabilir. Bu tip bir düzenlemeye etkileşimli denir çünkü işletim sistemi ile kullanıcı arasında bir diyalog söz konusudur. Kullanıcı her seferinde bir işlemi (sözelimi dosya düzenlemeyi) başarı ile tamamladığında, başarısıyla ilgili olarak bilgilendirilir ve mesajla, işine devam etmeye davet edilir. Eğer belli bir komut bir hata ile sonuçlanırsa kullanıcı, işletim sistemi tarafından uyarılır ve böylece gerekli düzeltme işlemlerini yapması sağlanmış olur.

Gerçek-zamanlı işletim sistemi, geniş ölçüde endüstrideki işlem denetleme dünyasına aittir. Temel özelliği bir olaya belirlenmiş bir süre içinde cevap vermesidir. Bilgisayar-denetimli bir petrokimya fabrikasını ele alalım. Bu fabrikayı, değişmez biçimde çalıştıran bilgisayar gerçek-zamanlı bir işletim sistemine sahiptir. Fabrikanın birçok yerindeki koşullar düzenli olarak ölçülüp bilgisayara bildirilir. Açıkta ki denetim işleri, fabrikadaki koşullar değiştiğinde yerine getirilmelidir. Bir reaksiyon teknesindeki ani basınç artışı görmezlikten gelinemez. Bilgisayar ya bir iş tarafından üretilen kesmeye cevap vermelidir ya da ölçme aletlerini yeterli sıklıkta taramalıdır. Gerçek-zamanlı işletim sistemleri; sistemlerin bilgisayar-denetimli olduğu ve bilgisayarın cevap süresinin, denetlemekte olduğu sistemi çok yakından izlemesi gerektiği her yerde bulunur.

Gerçek-zamanlı işletim sistemlerine böyle denmesinin nedeni; bilgisayarın, insanların 'saat zamanı' dediği şeyle programların harici olaylara makul bir süre zarfında cevap verebilecek biçimde senkronize olmalarıdır. Makul sözcüğü muğlak bir terimdir. Çünkü bilgisayarın nükleer bir reaktördeki değişikliklere vereceği cevap süresi bilgisayar denetimli bir seradaki değişikliklere verilecek olandan daha hızlı olmalıdır. Gerçek-zamanlı-olmayan sistemler, bilgisayarın kendi zamanında çalışırlar. Bir iş verildiğinde sonuçlar, belli bir süre geçtikten sonra alınır. Geçen zamanla günün zamanı arasında belli bir bağlantı yoktur. Gerçekte geçen süre bilgisayar ile çalıştığı belli işler karışımını yüklemenin bir fonksiyonudur. Bilgisayarın tahrik edilen olaylara cevap verme süresi, gerçek-zamanlı bir sistemde garanti edilmiştir.

### 9.1 Sistem yazılımı

Sistem yazılımı; bir kullanıcının yaşantısını, gerçekte mutlu olmasa da, daha kolaylaştırmak için tasarlanmış programların toplamıdır. Kısaca açıklanacak bazı sistem programları şunlardır: İşletim sistemi yardımcı programları, düzenleyici, metin işlemci, çevirici, hata ayıklayan programlar ve derleyicilerdir.

#### İşletim sistemi yardımcı programları

İşletim sistemi yardımcı programları: işletim sistemi tarafından sağlanan ve kullanıcının dışardan erişebildiği programlardır. Bu programlar, daha sonra düzenleyici, assembler, derleyici gibi diğer sistem yazılımı tarafından işlenebilecek dosyaların yaratılmasını sağlar.

Her işletim sisteminin kendisine özgü belli bir yardımcı program grubu vardır. Tipik bir sisteme örnek olarak, birçok 6809-tabanlı mikrobilgisayarlarda çalışabilen Flex 09 yardımcı programlarını seçtim. Dosyaadı, dosyanın adıdır ve en çok sekiz karakterden oluşur. Dosyaadına eklenmiş ve dosyanın tipini tanımlayan üç karakterlik bir uzantı vardır. Örneğin, TXT uzantısı dosyanın bir metin dosyası olduğunu gösterir. BIN bir ikili dosyasını ve CMD de bir komut dosyasını belirtir. Dosyaadı şu biçimde tanımlanır:

[<sürücü>.] <ad>[. <uzantı>]

'<>' karakterleri kullanıcı tarafından girilen alanları içine alır. '['ve']' sınırlayıcıları ise seçmeli bir parametreyi içine alır. Eğer disk sürücü belirtilmezse, işletim sistemi disk olarak sürekli kullanılmakta olan diski varsayar (başlangıçta 0. sürücü). Tipik olarak geçerli bazı dosyaadları şunlardır:

1. Pascal.TXT  
RunJob  
TEST.CMD

Aşağıda, Flex 09 tarafından sunulan en kullanışlı yardımcı programların bazıları yer almaktadır:

**CAT:** Katalog fonksiyonu kullanıcıya belli bir diskin kataloğu ve rehberinde o anda bulunan dosyaları terminalden listeleme olanağı sağlar. Katalog gerçekte diskin bir parçasıdır ve ondaki dosyaların adlarını; büyüklükleri ve her dosyanın ilk sektörünün konumunun bir listesini tutmak için saklanmıştır. Kullanıcıya diskte hangi

dosyalar olduğunu söyler.

**COPY:** Kopya komutu bir dosyanın bir diskten diğerine kopyalanmasını sağlar. Sözdizimi şöyledir: COPY, <dosyaadı 1>, <dosyaadı 2>. Bu komutun sonucu <dosyaadı 1>'i <dosyaadı2> adında yeni bir dosyaya kopyalamaktır. Örneğin COPY, 0. PROGRAM7.TXT, 1.PROG7 komutunun sonucu disk 0' daki PROGRAM7 adlı dosyanın disk 1'de PROG7 adlı bir dosyaya kopyalanmasıdır. Yeni dosyanın uzantısı da TXT olacaktır.

**DELETE:** Bu komut istenmeyen dosyaların silinebilmesine izin verir. DELETE, 1.PROG7.TXT komutunu girmenin sonucu disk 1'deki katalogdan PROG7.TXT'nin silinmesidir. Dosyayı fiziksel olarak ortadan kaldırmaya gerek olmadığına dikkat edin. Basitçe adının katalogdan çıkarılması artık onun bir dosya olmamasına yeterli olmaktadır.

**LIST:** Bu komutun sonucu, adı verilmiş dosyanın içeriğinin terminalde listelenmesidir.

**PROT:** Koruma fonksiyonunun sözdizimi

[PROT, <dosyaadı>, seçenek listesi]

olup adı verilmiş dosyanın kazara zarar görmesine karşı bir ölçüde koruma sağlar. Seçenek listesi aşağıdaki parametrelerden bir ve birkaçını içerir; D, W, C, X. 'D' dosyayı silme işleminden korur ve herhangi bir doğrudan veya dolaylı silmeyi içeren bir komutla silinmesini engeller. 'X', dosyaları yazma işlemine karşı korur ve dosyaya başka bilgilerin yazılmasına engel olma etkisi vardır. Aynı zamanda dosyanın silinmesine de engel olur. 'C', dosyayı katalogdan korur ve her katalog fonksiyonunun kullanılışında kataloğun içerdiği listede görünmesine engel olur. Eğer önceden orada olduğunu bilmiyorsa, katalogdan-korunmuş bir dosya kullanıcıya görünmez (yani gizli) olur. 'X', öteki seçenekler tarafından dosyaya yüklenmiş bulunan korumaları ortadan kaldırma etkisine sahiptir.

**SAVE:** Sakla komutunun genel sözdizimi şöyledir:

SAVE, <dosyaadı>, <başlangıç adresi>, <bitiş adresi> [ , <transfer adresi> ].

Bu komut; işlemcinin IAS'inde 'başlangıç adresi'nden 'bitiş adresi'ne kadar olan verinin ikili bir görüntüsü (kopyası) olan bir dosya yaratır. Dosyanın uzantısı, tersi belirtilmediği sürece BIN'dir. Seçmeli 'transfer adresi' parametresi, saklanan veri

daha sonra belleğe yüklendiğinde, yürütmenin başlayacağı giriş noktasını gösterir. SAVE, Prog6, 0100, IF0C, 0200 komutu, \$0100 ile \$IF0C arasındaki bellek konumlarında yer alan verinin disk 0'da Prog6 adlı bir dosyada saklanması sonucunun verir. Eğer bu dosya daha sonra belleğe yüklenirse, program sayıcısı \$0200 değerine kurulacaktır ki bu programın yürütülmesinin başlayacağı noktadır.

### Düzenleyici (Editör)

Düzenleyici, metin dosyalarını işlemek ve metni işletmen tarafından istenen şekilde formatlamak için tasarlanmış bir programdır. Pekçok düzenleyici, toplu-modda çalışmaktan çok çevrim-içi işlem için düşünülmüştür. VDU'lar ve çevrim-içi işlem bu denli popüler olmadan önce bilgi çoğu zaman bilgisayara delikli kart biçiminde girilirdi. Düzenleme geniş şekilde istenmeyen kartların atılıp yeni kartlarla yer değiştirilmesinden oluşurdu. Delikli kağıt bandı düzenleme, delikli bantları kesip ucuca eklemeyi içeriyordu ve iş tatmini açısından istenen hedefe erişilemiyordu.

Bugün geniş bir düzenleyici grubu vardır. Bilgisayardan bilgisayara geçirirler ve verilen bir bilgisayarda bile çoğu zaman farklı karmaşıklık düzeyinde bulunan bir grup değişik düzenleyici vardır. Bu, karşılaştığı her yeni düzenleyicinin sözdizimini öğrenmek zorunda olan bilgisayar kullanıcıları için kötü bir haberdır. Pekçok düzenleyicinin temel işlev birbirine benzer fakat komutları, düzenleyiciden düzenleyiciye bit açısından gerçek bir zahmet olacak kadar değişiklik gösterir. Bir keresinde, bir düzenleyiciden diğerine geçerken, 50 numaralı satırı silmek üzere D50 yazmıştım. Düzenleyici, 50. satırı silmek yerine ardışık 50 satırı silmişti. Meğerse, (# işareti satır numarası anlamına gelecek şekilde) D#50 girmem gerekiyordu.

Geleneksel olarak pekçok düzenleyici düzenlemekte oldukları metni, sanki kartların üzerine basılmış gibi, bir dizi satır olarak dikkate alır. Tüm işlemler, bir grup satır için tekrarlanıp uygulansa da, aslında metnin bir satırına uygulanır. Bu tür bir düzenleyiciye satır düzenleyicisi denir. Öteki düzenleyiciler metnin o sırada VDU'da görüntülenen sayfası üzerinde işlem yaparlar. Bu düzenleyicilere ekran düzenleyicisi denir ve bir işlemde tüm paragrafları veya metnin bir sütununu işleyebildiklerinden hızla satır düzenleyicilerin yerini almaktadır.

Pekçok satır düzenleyicisi, karmaşık olanaklar içerse de, burada sadece temel işlevleri açıklanacaktır. Bir satır düzenleyicisinin sunduğu en önemli beş işlev şunlardır:

I araya ekle, D sil, R yer değiştir, F bul, C değiştir.

Bir satır düzenleyicisi metin dosyasını, 1'den N'ye kadar numaralandırılmış satır toplamı olarak dikkate alır. Herhangi bir anda satır işaretçisinin konumlandırıldığı satıra yürürlükteki satır denir. Parametresiz bir komut, yürürlükteki satırda işlem yapar. Araya ekle komutu, ya yeni bir dosyanın girilmesini; ya da mevcut dosyaya bir ekleme (yani bir ya da daha çok yeni satır) yapılmasını sağlar. Silme komutu bir veya daha çok istenmeyen satırı siler. D, tipik olarak yürürlükteki satırı siler. D3, (yürürlüktekinden başlayarak) üç satır silerken; D#3 ise, 3 numaralı satırı siler.

Yer değiştirme komutu bir satırın bir başkasıyla yer değiştirmesine neden olur. Kullanımı, bir silmeyi izleyen araya eklemeye aynıdır. Ara komutu operatörün, metin içinde verilen bir karakter dizisini aramasına olanak tanır. Dizi terimi, herhangi bir ardışık karakterler sırasını adlandırmak için kullanılır. Bu komutun yokluğunda, programcı ya tüm satır numaralarını bilmelidir ya da önünde (satır numaraları ile) programın bir listesi bulunmalıdır. Örneğin operatör 'LDA DATA7'yi içeren satırın yerini bulmak istiyorsa, aşağıdaki komutu girer:

F/LDA DATA7/

Bölü işareti, /, dizi için bir sınırlayıcı oluşturur. Düzenleyici aşağıdaki satırı görüntüleyerek cevap verebilir.

0023 START LDA DATA get 'DATA'

Artık 23. satır, yürürlükteki satır olmuştur ve istenirse düzenlenebilir. "Değiştir" işlevi bir dizinin bir başkasıyla değiştirilmesini sağlar ve satırın tamamının yer değiştirilmesi zorunluluğunu ortadan kaldırır. Yukarıdaki örnekte, DATA7'yi VAL9 ile değiştirmek istenirse, C/DATA7/VAL9/ girilir.

Düzenleyiciler çoğu zaman bir grup satır üzerinde çalışan genel komutlar sunarlar. Örneğin, değiştir komutu, C/DATA7/VAL9/12 ile, tek bir işlemde, metinde gelen sonraki 12 satırdaki tüm DATA7'leri değiştirmek için kullanılır.

### Metin işlemci (Text processor)

Metin işlemci, veya kelime işlemci, işi, düzenleyicinin bıraktığı yerden devralır. Düzenleyici ya program ve öteki yapısal veri biçimleri ya da saf metin yaratmak için kullanılır. Örneğin, bu kitabın metni bir düzenleyici tarafından üretilip işlenmiştir. Metin işlemcinin işlevi, düzenleyici tarafından yaratılan metin dosyası üzerinde genel olarak işlem yapmak ve basıma uygun bir form üretmektir. Metin işlemciler sayfanın soluna ve sağına bloklama yapabilirler. Bu; metnin hem sol hem

de sağ marjlarının aynı hizada olması anlamına gelir. Bu da kitaplardaki veya gazetelerdeki metne benzer mektup veya diğer metinler üretilebilmesini sağlar.

Metin işlemciler aynı zamanda, sayfalara başlık koyma ve bunları numaralamayı dipnotları gibi başka işlemler de yaparlar.

### Çevirici (Assembler)

Çevirici; çevirici yönergeleri ve yürütülebilir komutlardan meydana gelen bir assembly-dili programını alır ve bunu makine-kodlu biçime dönüştürür. Çevirici tarafından yaratılan makine kodu, diskte bir dosya olarak saklanabilir veya doğrudan yürütülmek üzere bilgisayarın belleğine yüklenir.

Çeviricileri daha önce ele aldığımızdan, burada sadece birkaç ayrıntı daha eklenecektir. Çevirici, çoğu zaman programın üzerinde çalıştırılacağı makine için makine kodu üretir. Öte yandan kimi zaman bir programı bir makinede derleyip bir başkası için kod üretmek daha uygundur. Örneğin, bir UNIVAC 1100, daha sonra bir 6502-tabanlı bilgisayarda çalıştırmak üzere bir programı derlemek için kullanılabilir. Bu tür bir çeviriciye çapraz-çevirici denir.

Assembler ile yakından ilişkili program, emülatörlerdir. Emülatör, makine kodunu yazılım olarak yürüten (yani yorumlayan) programdır. Örneğin, bir 6502 programı bir UNIVAC 1100'de derlenebilir ve sonuçtaki 6502 makine kodu satır satır yorumlanır. Yani UNIVAC makine kodundaki bir program 6502 opkodlarını okur ve ardından yürütür. Bu işlemler, elbette UNIVAC'ın belleğinde bulunan 'sentetik' 6502 kaydedicilerinde gerçekleştirilir. Makine kodunun bu şekilde yorumlanması, nispeten yavaş ve verimsizdir. Fakat kodun, tasarlanan makinede çalıştırılmadan önce hatalar açısından kontrol edilebilmesine olanak tanır.

### Hata ayıklayıcı (Debugger)

Hata ayıklayıcı programlar, bir başka programın çalıştırılmasını görüntülemek için tasarlanmış programlardır. Burda makine düzeyli programlara ilişkin hata ayıklayıcı programlarıyla ilgileneceğiz. Bir program derlendikten sonra yürütülebilir. Kimi zaman program çalışmaz, veya beklenen sonuçları vermez. Böyle olduğunda, hata ayıklayıcı, problemi bulmaya yardımcı olmak üzere gerekmektedir. Hata ayıklayıcı ile test edilmekte olan programın aynı anda bellekte olması gerektiğine dikkat edin. Hata ayıklayıcıya böyle denmesinin nedeni hataları ayıklayıp temizle-

diği anlamına gelmesi nedeniyledir. Bir hata ayıklayıcının temel işlevleri, bellek konumlarını incelemek, durdurma noktaları kurmak ve programın yürütülmesini izlemektir.

'Belleği görüntüle' komutu, belirtilen bellek konumlarının içerdiği değerleri incelemek ve verinin beklenen şekilde değişip değişmediğini kontrol etmek için kullanılır. Aynı zamanda, bir programı çalıştırmadan önce bellek konumlarının içeriklerini sıfırlamak için de kullanılır. Bu, programın bilinen veriler ile test edilebilmesini sağlar.

Durdurma noktası, programa eklenen bir işaretir ve program çalışırken karşılaşıldığında önceden tanımlanmış bir işin yerine getirilmesine neden olur. Durdurma noktası makine kodu komut biçimini alır. 6502 buna, BRK (durdurma); 6800 ise SWI (yazılımsal kesme) der. Bir programın beklendiği biçimde çalışmadığı varsayalım. Problemin bulunduğu yer sanılan bir bölgeye bir durdurma noktası ekleyerek programın işleyişiyle ilgili bir kontrol yapılabilir. Program çalıştırıldığında, durdurma noktası ile karşılaşılan kadar normal işlem devam eder. Sonra programın yürütülmesi durdurulur ve MIB'in iç kaydedicilerinin içerdiği değerlerin dökümü alınır. Bu programcıya, durdurma noktasının hemen ardından bilgisayarın bulunduğu durumun bir fotoğrafını sunar. Durdurma noktasıyla karşılaşıldıktan sonra (kimi zaman) yürütmeyi devam ettirmek de mümkündür. Hata ayıklayıcının izleme fonksiyonu daha çok, program boyunca yavaş yavaş hareket ettirilen otomatik bir durdurma noktası gibidir. Her bir komut yürütüldükten sonra, tüm kaydedicilerin durumu görüntülenir. Bu şekilde, program çalıştırılırken tam olarak ne olduğunu görüntülemek mümkündür.

### Derleyiciler (Compilers)

Derleyiciler, üst-düzeyle bir dildeki kaynak programı alan ve yürütme için uygun makine koduna çeviren (çoğu zaman işletim sistemi kadar hatta daha da karmaşık) bir programdır. Derleyicilerin çalışması bu kitabın kapsamı dışındadır.

## 9.2 Çoklu programlama

Çoklu programlama, bir bilgisayarın bir seferde birden çok işi ele alabilme yeteneğidir. Öğrenciler çoğu zaman bana, biraz şaşırılmış bir ses tonuyla şunu sorarlar: Fakat bir bilgisayar herhangi bir anda birden çok programı nasıl çalıştırabilir? Herşeyin ötesinde, sadece bir program sayıcısı ve bir ALU'su vardır. Verdiğim cevap, kesin konuşmak gerekirse, haklı oldukları yolundadır. Bilgisayarlar, iki veya

daha çok programı eşzamanlı olarak yürütemez, fakat aynı anda birkaç program çalıştırıyor izlenimi uyandırabilir. Aşağıdaki örnek bu yanılsamanın nasıl mümkün olduğunu göstermektedir.

Eşzamanlı bir satranç oyununu ele alalım. Birinci sınıf bir satranç oyuncusu bir grup zayıf rakibiyle karşılaşmaktadır. O tahtadan tahtaya dolaşarak her seferinde bir hamle yapar. Rakiplerinden çok daha iyi olduğu için bir hamlesi, rakiplerinin kinden çok daha kısa sürer. Bunun sonucu olarak her oyuncu, karşılarında sanki sadece kendilerine ait tek bir rakip varmış sanırlar.

Eşzamanlı satranç oyunu organizasyonu kolayca sayısal bilgisayara da uygulanabilir. Gerekli olan tek şey, MİB'i bir işten diğerine geçirecek periyodik bir sinyal ve bilgisayara belli bir işi en son yürütürken nerede kalmış olduğunu söyleyecek bir mekanizmadır. Bu işler normalde görevler veya işlemler olarak anılır ve birkaç görevi birlikte yürütme kavramına, birlikte programlama veya çoklu programlama denir.

Çoklu programlamanın nasıl gerçekleştirilebildiğine bakmadan önce, ne gibi avantajları olduğunu belirtmek gerekir. Her bir görevin istediği MİB zamanından başka bir şey değilse çoklu programlamanın görevleri ardışık olarak çalıştırabilmenin yanında çok az bir avantajı vardır. Eşzamanlı satranç örneğini tekrar incelersek burada başarının, rakiplerinininkiyle karşılaştırıldığında usta oyuncunun olağanüstü hızına dayandığını görürüz. Her bir oyuncu büyük bir gayretle bir sonraki hamlesini düşünürken, usta oyuncu o anda birden çok hamle yapmakla meşguldür.

Benzer bir durum bilgisayarlar örneğinde de vardır. Bir kullanıcı bilgi okumakla veya basmakla veya bir disk sürücüsüne erişmekle meşgulken, bir başka kullanıcı MİB'i kendi kontrolüne alabilir. Bugün 40 veya daha fazla öğrencinin verilen bir anda anabilgisayara erişmek istemesi durumunda bu çok önem kazanmaktadır.

Şekil 9.1, A ve B adlı iki görev ile çoklu programlamanın avantajlarını göstermektedir. Bu görevlerin her biri yürütülmeleri esnasında bir grup farklı işin (örneğin VDU giriş/çıkış, disk erişimi vs) yerine getirilmesine ihtiyaç duyar. Bu iki görevin her biri tarafından yapılan işlerin sırası, yürütüldükçe, Şekil 9.1(a)'da verilmiştir.

Eğer B başlatılmadan önce, A'nın, tamamlanuncaya kadar çalışmasına izin verilsediydi, MİB ile ilgisi olmayan işler yapılırken değerli bir işlem-zamanı boşa gidecekti. Şekil 9.1(b)'de kaynakların daha verimli bir kullanımı sağlamak için görevlerin nasıl zamanlandırılacağı gösterilmiştir. Kutucuklar, verilen bir kaynağın belli bir görev için ayrıldığı zaman periyodunu gösterir. Örneğin A görevi MİB'i

|          |       |     |      |     |       |         |
|----------|-------|-----|------|-----|-------|---------|
| A görevi | VDU 1 | MİB | DİSK | MİB | VDU 1 | zaman → |
|----------|-------|-----|------|-----|-------|---------|

|          |       |     |      |       |     |
|----------|-------|-----|------|-------|-----|
| B görevi | VDU 2 | MİB | DİSK | VDU 2 | MİB |
|----------|-------|-----|------|-------|-----|

| Kaynak | Etkinlik |          |          |          |          |          |
|--------|----------|----------|----------|----------|----------|----------|
| VDU 1  | A görevi |          |          |          | A görevi |          |
| VDU 2  | B görevi |          |          |          | B görevi |          |
| DİSK   |          |          | A görevi | B görevi |          |          |
| MİB    |          | A görevi | B görevi | A görevi |          | B görevi |

Şekil 9.1 Bir çoklu programlama örneği

(a) Gerek duyulan sıralı işlemler olarak iki görev (b) İki görevin zamanlanması

ilk defa kullandıktan sonra, diske erişir. Diske, A görevi tarafından erişilirken, B görevi işlemcinin kontrolünü alabilir.

Çoklu programlama işletim sisteminin ince ayrıntıları, giriş niteliğindeki bu kitabın kapsamının çok ötesindedir. Bununla beraber burada, aşağıdaki ilkeler verilmiştir:

1. İşletim sisteminin bünyesinde bir işi en verimli şekilde zamanlamayı ve mevcut olanakları en iyi şekilde kullanmayı sağlamak üzere karmaşık algoritmalar vardır. Algoritmalar, çalıştırdığı işlerin türüne uyarlanabilir veya operatörler verimliliği maksimuma çıkarmak için bilgisayarı sistem parametreleri ile besleyebilirler. Eğer parametreler, operatörler tarafından değiştiriliyorsa buna işletim sistemini ayarlamak denir.

2. Olasılıkla işletim sistemi tarafından karşılaşılan en büyük problem, bellek yönetimidir. Eğer birden çok görev çalıştırılacaksa, işletim sistemi bunların herbirine bellek alanı ayırmalıdır. Bundan başka işletim sistemi bu görevleri, olası en iyi kullanımı sağlayacak şekilde belleğe yerleştirmelidir. Açıkçası bellekte görevler arası geçişler için, geniş boşluklar bırakılmamalıdır.

3. Eğer MİB, bir görev diske veya yazıcıya erişiyorken, bir başka görev için kullanılıyor olacaksa, bu aygıtlar (yani disk, yazıcı vb) kendi kendilerine çalışabiliyor olmalıdırlar. Yani, ya MİB etkin biçimde işe karışmadan DMA işlemlerinde yer alabilmeli ya da MİB'den yüksek-hızda bir yığın veri alıp bunları kendi isteğine göre işleyebilmelidir.

## Görevler arası geçişler

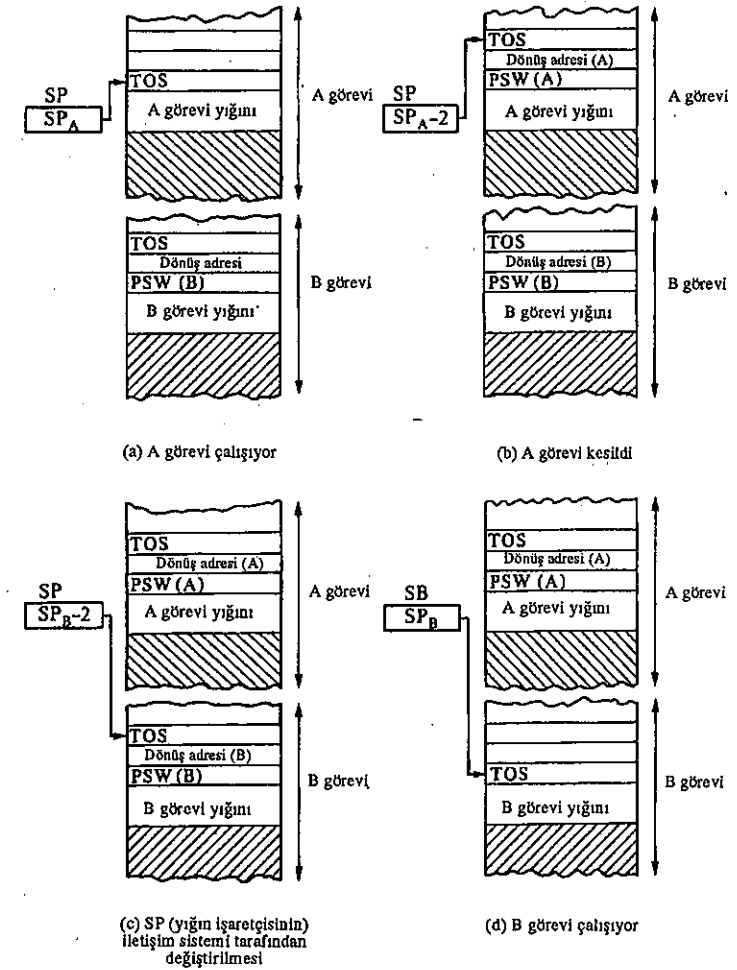
Daha önce bu kitabın, işletim sisteminin iç işlemlerine giremeyeceği belirtilmişti. Bununla birlikte görevler arası geçişin, meydana geliş biçimini özetlemek de mümkündür. Görevler arası geçiş konusunu buraya almamın önemli bir nedeni, daha önce açıklanan iki donanım mekanizmasını, kesme ve yığın'ı, kapsıyor olmasıdır. Bir elektronik darbe üreticinin (yani bir saatin) bir MİB'in  $\overline{\text{NMI}}$  hattına bağlanacağını varsayalım. Her darbeye (sözelimi her 0.01 saniyede) maskelenemez bir kesme üretilir. Bu, program sayıcısı ile PSW'nın yığında saklanması ve kesme-yönetim yordamına bir atlama yapılması ile sonuçlanır. Eğer kesme-yönetim yordamının sonunda bir RTI (kesmeden dönüş) yürütülürse, program sayıcısı ile PSW geriye alınır ve programın yürütülmesi kesildiği noktadan devam eder.

Şimdi de kesme-yönetim yordamının şeytani kurnazlıkta bir şey yaptığını, RTI yürütülmeden önce yığın işaretçisini değiştirmiş olduğunu varsayalım. Artık, RTI yürütüldüğünde yığından getirilen program sayıcısının içeriği, kesmeden önce yürütülmekte olan programa ait değil, başka bir programa ait bir değerdir. Burada görevler arasında geçişi etkileyen şey, yığın işaretçisinin değiştirilmesidir.

Şekil 9.2'de görevler arası geçişin nasıl çalıştığı görülmektedir. A ve B adlı iki görev belleğe yerleştirilmiştir. İşleri kolaylaştırmak için, görevler için ayrılan bellek alanları, görevlerin yürütülmesi boyunca değiştirilmemiştir. Her görevin kendi yığını vardır ve herhangi bir anda sistem yığın işaretçisi, A ya da B'nin yığınına işaret edebilmektedir. Şekil 9.2(a)'da A görevi çalışmaktadır. Şekil 9.2(b)'de bir NMI ortaya çıkar ve program sayıcısı ile PSW'nin içerikleri yığına itilir (yani A'nın yığına).

Şekil 9.2(c)'de işletim sistemi sistem yığın işaretçisinin değerini değiştirmiştir ve böylece işaretçi artık B görevinin yığınına göstermektedir. Son olarak, Şekil 9.2(d)'de işletim sistemi bir RTI yürütür ve B görevinin program sayıcısı, yığından geriye yüklenir, böylece B'nin yürütülmesine neden olur.

Dolayısıyla, işletim sistemi her kesmede, RTI'den önce yığın işaretçisinin yerini değiştirerek görevler arası geçişe neden olur. Pratik bir sistemde işletim sistemi, yürütülecek bir görevler listesi tutar. Yığın işaretçisine ek olarak, tablodaki her kayıt görevle ilgili daha fazla bilgi içerir. Tipik olarak burada görevin öncelik sırası, maksimum çalışma süresi ve o sırada çalıştırılabilir olup olmadığı ile ilgili ayrıntılar yer alır. Bir görev aşağıdaki üç durumdan birinde olabilir. Görev çalıştırılmakta (o sırada yürütülmekte) olabilir, çalışmak için sırasının gelmesini bekliyor olabilir veya bloklanmış olabilir (çalıştırılmadan önce bazı kaynakların kullanılabilir hale



Şekil 9.2. Görevlerarası geçiş

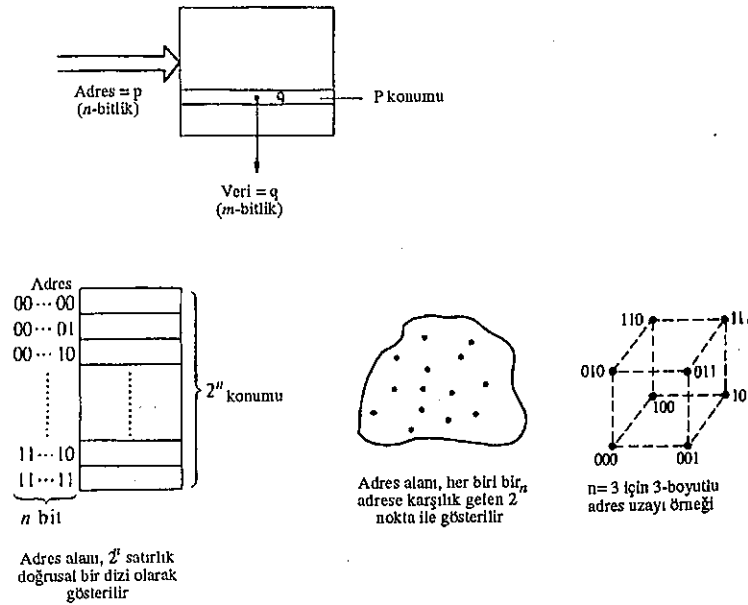
gelmesini beklediği sürece çalıştırılmaz. Yüksek öncelikteki bir görev daima düşük öncelikli bir göreve tercih edilecek şekilde, görevlere öncelik sırası verilebilir. Bir görev, sırası geldiğinde yürütülürse, çalıştırılabilir denir (öncelik sırası sınırlamalarına bağlıdır). Eğer görev çalıştırılabilir değilse (yani bloke edilmişse), bilgisayarda kalır sırası geldiği her seferde üzerinde işlem yapılmadan geçilir. Görev çalıştırılacaksa, çalışma bayrağı kurulur ve bir sonraki süre boyunca yürütülür.

## 9.3 Bellek yönetimi

Şimdiye kadar, bilgisayarın merkezi işlem biriminin, bir komut veya verinin adresini ürettiğini ve bu adresin bellekteki verinin gerçek adresine karşılık geldiğini

varsaydık. Böylece, eğer bilgisayar LDA \$1234'ü yürütürse, işlenen, IAS \$1234 konum adresinde bulunur. Bu pek çok mikroişlemci sistemleri için doğru, birçok minibilgisayar ve anabilgisayar için doğru değildir. MİB tarafından üretilen adres her zaman verinin gerçek konumuna karşılık gelmez. Bunun böyle olduğu, bu kısımda ele alınacaktır.

Bellek yönetimi genel bir terimdir ve MİB tarafından üretilen bir adresin belleğin verinin gerçek adresine dönüştürülmesinde kullanılan çeşitli teknikleri kapsar. Bellek yönetiminin gerekliliğine bakmadan önce, bellekteki verinin konumunu MİB tarafından üretilen verinin adresi arasındaki ilişkiyi vurgulayan bazı kavramları incelemekte yarar vardır. Klasik bellek kavramı Şekil 9.3'te gösterilmiştir. Burada bir okuma çevriminde p adlı bir adres 9 adlı bir veri sözcüğünü verir. Üzere, belleğin adres girişlerine uygulanır.  $n$ -bitlik bir adres, en çok  $2^n$  kadar farklı konum sağlar ve 0'dan  $2^n - 1$ 'e kadar doğrusal bir liste olarak gösterilebilir. Tüm konumlar, bellek alanı denen  $n$ -boyutlu bir alana eşlenebilirler. Bu bellek alanı fiziksel bellek alanı denir çünkü bellek alanındaki her bir nokta gerçek bellekten bir konuma karşılık gelir. Şekil 9.3'de fiziksel adres uzayı kavramı gösterilmiştir.  $n=3$  durumu için örnek verilmiştir.

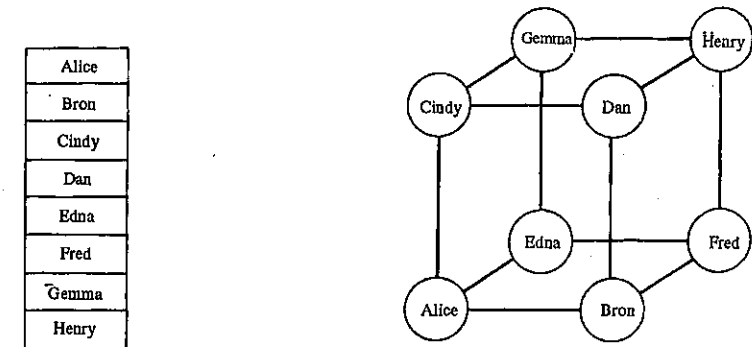


Şekil 9.3 Fiziksel adres uzayı

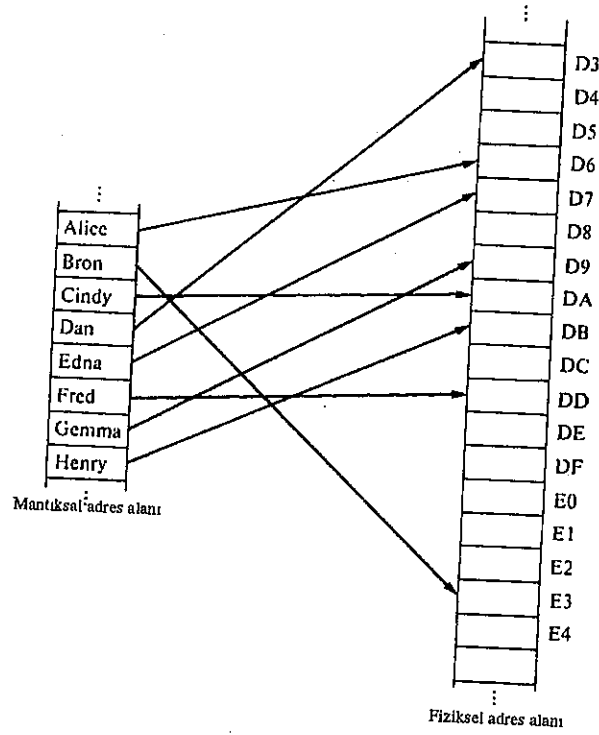
Fiziksel adres uzayı genel bir kavramdır ve belli bir bellek türü ile sınırlandırılmaz. Hatta yüksek-hızlı RAM'dan düşük hızlı manyetik teybe kadar bir sürü cihaza kadar geniş bir aralıkta kullanılabilir. Amacımıza uygun olarak, fiziksel belleğin yalnızca IAS'yi içerdiğini ve özelliklerindeki büyük farklılıklarından dolayı saklama ortamının hariç tutulduğunu varsayacağız. Daha sonra sanal bellek sistemlerinin saklama ortamını rastgele-erişimli fiziksel bellek uzayının bir bölümü olarak kullanabilmesini göreceğiz.

Programcı fiziksel adres uzayı ile ilgilenmez. Daha çok adres konumlarının adlarından meydana gelen adres alanı olan mantıksal adres uzayı ile ilgilenir. Aşağıdaki benzeştirme mantıksal ve fiziksel adres arasındaki farkı göstermektedir. Bir doktorun hasta kayıtlarını sakladığı bir dosya dolabı vardır. Bu kayıtlar alfabetik sırayla dosyalararak mantıksal olarak saklanmıştır. Hiçbir zaman hastaların fiziksel adreslerinin alfabetik sırasına göre saklanmazlar.

Mantıksal adres uzayına örnek olarak, sekiz addan meydana gelen bir adres uzayını ele alalım: ALICE, BRON, CINDY, DAN, EDNA, FRED, GEMMA, HENRY. Bu mantıksal adres uzayı hem doğrusal bir liste olarak, hem de 3-boyutlu bir uzay olarak Şekil 9.4'de gösterilmektedir. Bu isimlerin bir fiziksel adres uzayına haritalanması gerektiğini varsayalım. Bunu yapmanın bir yolu her ismin ilk üç harfinin ASCII kodlarını toplayarak her bir isim için fiziksel bir adres yaratmaktır. Böylece ALICE,  $41+4C+49=D6$  fiziksel onaltılı adresi olarak verilir. Şekil 9.5'ten de görebileceğimiz gibi (bu adres üretme algoritmasıyla garanti edilmese de) sekiz addan herbirinin birbirinden ayrı bir konumu vardır, ve buradaki sıralanış sıra mantıksal isim listesindeki sırayla aynı değildir. Ayrıca adların fiziksel adresleri bitişik (ardarda) değildir.



Şekil 9.4 Mantıksal adres uzayı



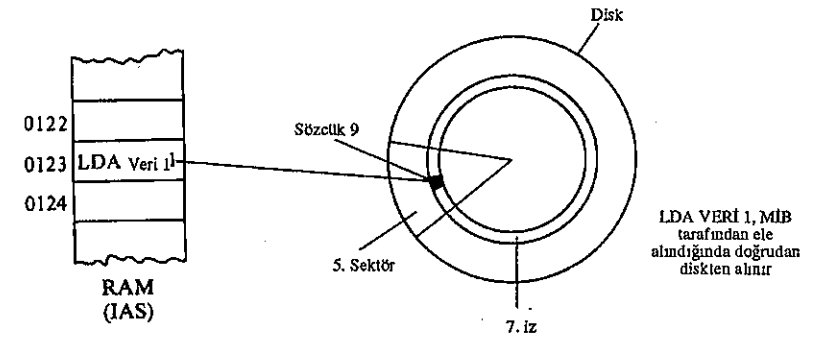
Şekil 9.5 Fiziksel ve mantıksal adres uzayı arasındaki ilişki

Mantıksal ve fiziksel adres uzayını birbirinden ayırmak güzel bir akademik çalışmadır fakat bu bilgisayarları tasarlayan programcı ve mühendisleri nasıl etkileyecektir? Varsayalım ki, 16K baytlık RAM'ı olan ucuz bir mikrobilgisayar satın aldım. 16K bayta kadar olan programları çalıştırabilirim fakat daha büyük programları çalıştıramam. Bir başka deyişle, (adres yolunu 16-bit varsayarak) bilgisayar, mantıksal adres alanı birbirinden farklı olarak adreslenebilen 64K tane konumu olanaklı kılarsa da, fiziksel adres alanı sadece 16K konum ile sınırlıdır. Benzer şekilde, 6502-tabanlı bir bilgisayar için 256K baytlık dinamik RAM blokunu almaya param yetebilir ama mantıksal adres alanı 64K ile sınırlı olduğundan tüm fiziksel bellek konumunu adresleyemem. MİB tarafından üretilen adres mantıksal bir adrestir çünkü fiziksel bellekte belli bir yerde bulunan bir konumu gösterimler.

20 bit veya daha fazla adreslerin yaygın olduğu anabilgisayar dünyasında problem, IAS'deki yüksek-hızlı RAM tarafından sağlanan fiziksel adres uzayının MİB'in mantıksal adres uzayından çok daha küçük olmasıdır ve bu da yürütülebilecek

programın büyüklüğünü sınırlar. Mantıksal adres alanının fiziksel adres alanından çok büyük olması durumunda ortaya çıkan bu problem, 'sanal bellek' tekniklerine başvurarak çözülür. Burada düşük hızlı, yüksek kapasiteli disk ve teyp birimlerinin, yüksek-hızlı IAS'ler gibi kullanılabilmesi sağlanır.

68000-tabanlı bir mikroişlemci sistemi, 16 Mbaytlık bir mantıksal adres alanına karşılık gelen 24-bitlik adres yoluna sahiptir. Pratikte bir sistem 256 Kbaytlık IAS ile 20 Mbaytlık bir sabit-disk belleği ile birlikte verilir. Problem, 256 Kbayttan büyük programları nasıl çalıştırabileceğimizdir. Olası bir çözüm doğrudan-erişimli belleği, disk alanını da içine alacak biçimde genişletmektir. Yani, MİB'den gelen bir adres, IAS'ye ya da diskin içerdiği değerlere doğrudan erişmek üzere kullanılabilir. Örneğin, LDA VERI1 komutunun, IAS'de \$1234 adresinde konumlandırılan LDA'ya karşılık geldiğini ve VERI1 işlenenin diskte 7. iz, 5. sektör, 9. sözcük fiziksel adresinde bulunduğunu varsayalım. Şekil 9.6'da bu duruma karşılık gelen bellek haritası görülmektedir. MİB, IAS'den LDA VERI1 komutunu okur, 7. iz, 5. sektör, 9. sözcük şeklinde kodu çözülün VERI1 mantıksal adresini üretir. Bu aşamada donanım, bu konumu diskten okumalı ve içerdiği değeri akümülatöre aktarmalıdır. Bu okuma işlemi gerçekleşirken MİB, bekleme-durumunda dondurulmalıdır (veya durdurulmalıdır). Artık fiziksel adres alanı programın mantıksal adres alanını ele alabilecek kadar geniş olsa da bu düzenleme çalışabilir halde değildir. Ortalama disk erişim süresi 100 ms civarındadır ki bu IAS'den yüzbin kez daha yavaşlığı gösterir. Daha iyi bir yaklaşım, yerpaylaşımlar kullanarak sağlanır.

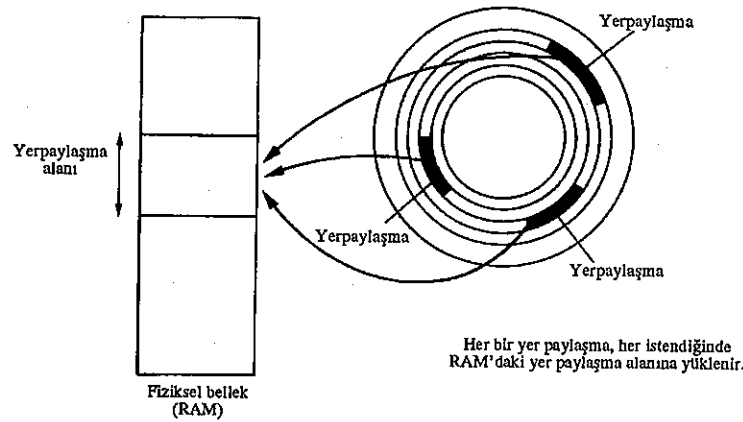


Şekil 9.6. Adres alanının disk alanını kapsamak üzere genişletilmesi

### Yerpaylaşma

Programcı; programını, bir ana bölüm ve yer paylaşma denen bir grup parçalar

biçiminde yazar. Bu yerpaylaşımlar karşılıklı olarak birbirleriyle bağdaşmaz biçimdedir, öyle ki herhangi bir anda programın ana bölümü tarafından sadece bir yerpaylaşmaya gereksinim duyulur. Yer paylaşmalar kullanan bir programın çalıştırılması gerektiğinde, işletim sistemi IAS'ye sadece ana bölümü yükleyerek yerpaylaşımları diskte bırakır. Çeşitli yer paylaşmalar, ana program tarafından istendikçe yer paylaşmaların getirilmesi için programcı tarafından saklı tutulan IAS'nin bir bölgesine yazılır. Aynı anda iki yer paylaşmaya ihtiyaç olmadığından, her bir yeni yerpaylaşma eskisinin üstüne yazılır. Şekil 9.7'de yerpaylaşımlar kullanılırken IAS ile disk saklama ortamı arasındaki ilişki görülmektedir.



Şekil 9.7. Disk yerpaylaşımları

Yerpaylaşımların ardındaki düşünceye kimi zaman, -işlemin %80'i, genellikle, programın %20 tarafından yapılır şeklindeki ampirik gözlemden dolayı- 80/20 kuralı denir. Bunun sonucu olarak, programın çalıştırıldığı tüm süre boyunca programın aktif %20'lik kısmı IAS'ye yüklenebilir ve kalan %80 yerpaylaşımlara bölünerek gerektiğinde yüklenir.

Yerpaylaşma, mantıksal adres alanının mevcut fiziksel adres alanından büyük olması problemini, yerpaylaşma alanının fiziksel adres alanını birkaç mantıksal adres alanını paylaşmasına izin vererek çözer. Yerpaylaşma kullanımı, basitlik avantajına sahiptir; yerpaylaşmalı belleği gerçekleştirmek için ek donanım ihtiyacı yoktur. Ne yazık ki, programcı üzerinde programları belli bir şekilde yazması için büyük baskı uygular ve her durumda sadece belli türde programlar için kabul

edilebilir. Buna güzel bir örnek bazı işlemlerin (örneğin DELETE, LIST, RENAME) yerpaylaşma olarak ele alındığı ve istendiği zaman (belleğe) yüklendiği işletim sistemleridir.

İşletim sisteminin diğer kısımları (örneğin disk dosya yönetmeni) bunlar sürekli olarak kullanımda olduklarından yerpaylaşmalı yapılamazlar. Sınırlı fiziksel adres alanı problemine daha iyi ve daha genel bir çözüm sayfalama ile sağlanır.

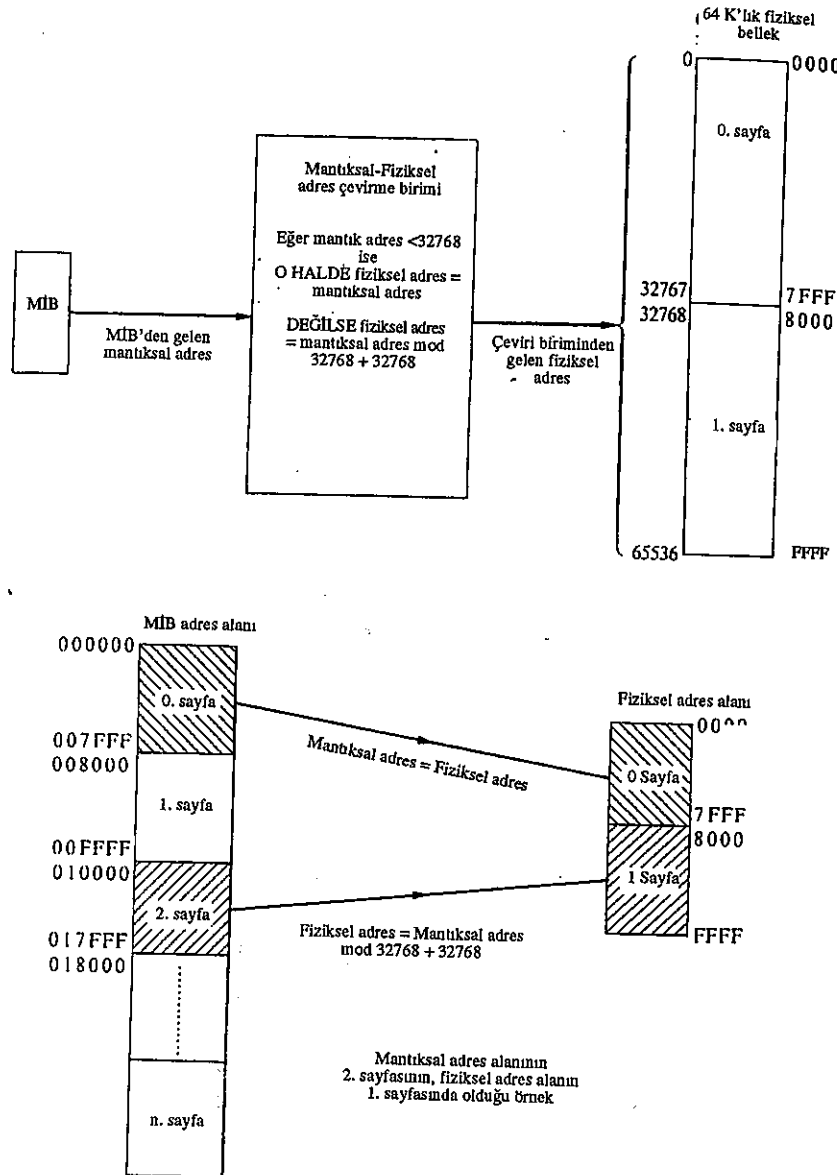
### Sayfalama

Adresleme tekniklerini ele alırken tıpkı bir kitabın sayfaları bölündüğü gibi belleğin de sayfa denen bir grup birime nasıl bölünebileceğini görmüştük. Örneğin, 16-bitlik bir adres 8-bitlik bir sayfa numarası ve 8-bitlik sayfa içindeki konum adresine bölünebilir. Bu kavram hem IAS'nin hem de saklama ortamının fiziksel adres alanını sayfaları bölerek sınırlı fiziksel adres alanı problemine uygulanabilir.

Sadece o sırada yürütülmekte olan programlar IAS'deki sayfaları doldurur, öteki tüm programlar ve veriler diskte kalır. MİB'in o anda IAS'deki bir sayfada bulunmayan bir veriye her ihtiyaç duyulduğunda, bu veriyi içeren sayfa diskten IAS'ye taşınır. İlk bakışta bu yapılması mantıksızmış gibi görünür. Diskten tek bir sözcük getirmek yerine, artık tüm bir sayfa dolusu sözcük getirmekteyiz. Bereket versin ki, bir bilgisayarda saklanan bilginin çoğu; q adresinde saklanan veriler, q'ya yakın adreslerde bulunan verilerle yakından bir ilişki içinde olacak şekilde mantıksal olarak düzenlenmiştir. Bir sayfa diskten getirildikten ve IAS'ye yerleştirildikten sonra, bunu takip eden birkaç erişimin bu sayfaya rastlama olasılığı oldukça yüksektir.

Sayfalamaya basit bir örnek Şekil 9.8'de sunulmuştur. Burada 64K'lık fiziksel bir bellek alanı iki sayfaya bölünmüştür. Sayfa 0'da 0 ile 32767 konumları arasında bulunan veri sabittir ve programın çalıştırılması sırasında değişmez. 32768'den 65535'e dek uzanan 1. sayfa, diskten gelen ve o sırada kullanımda olan sayfayı tutar. Adres haritalama veya adres çevirisi denilen bir işlem MİB'den gelen mantıksal adresi IAS'deki verinin fiziksel adresine dönüştürür. Mantıksal ve fiziksel adres arasındaki ilişki, aşağıdaki gibi ifade edilebilir:

```
IF    Mantıksal adres < 32768
THEN
      Fiziksel adres:=Mantıksal adres
ELSE
      Fiziksel adres:=Mantıksal Adres mod 32768+32768
```



Şekil 9.8 Basit bir sayfalama örneği

Örneğin MIB, \$001234 mantıksal adresine gönderme yaparsa bu, \$1234 fiziksel adresine çevrilir. Fakat MIB \$011234 adresine gönderme yaparsa \$0011234 32768'e (\$8000) bölünerek sayfa numarası (yani 2) elde edilir ve kalana (yani \$1234) \$8000 eklenerek verinin IAS'deki konumu (yani \$9234) bulunur. 2. Sayfanın bu esnada IAS'de olduğunu varsaydığımıza dikkat edin.

Program yürütülürken bazı noktalarda işlemci o esnada IAS'de olan sayfanın dışına düşen bir mantıksal adres üretebilir. Bu durumda diskten yeni bir sayfa getirilip, fiziksel belleğe 32768'den 65536'ya kadar yüklenmelidir. Bu işlem, işletim sistemi tarafından yerine getirilir. Adres haritalama algoritması böylece o esnada IAS'de olmayan adresleri algılayabilecek, içerecek şekilde genişletilebilir.

```

IF   Mantıksal Adres 32768
THEN
    Fiziksel/Adres:=Mantıksal Adres
ELSE
    IF   Yürürlükteki Sayfa = Fiziksel Adres 32768
    THEN
        Fiziksel/Adres:=Mantıksal Adres mod 32768+32768
    ELSE
        BEGIN
            Yenisayfa Al;
            Fiziksel Adres:=Mantıksal Adres mod 32768+32768
        END
    
```

Artık o esnada erişilmekte olan bellek konumunu içeren sayfanın IAS'de saklandığı ve işlemci tarafından üretilen mantıksal adresin otomatik olarak IAS'deki istenen verinin adresine çevrildiği anlaşılmış olmalıdır. Bu çevirme yüksek-hızlı donanım-la yapılır. Verinin diskten IAS'ye taşınma işlemine içeriye-değiş-tokuş denir ve işletim sistemi tarafından yerine getirilir. Mantıksal Adresin o esnada IAS'de bulunmayan sayfadaki bir konuma karşılık geldiği her seferinde adres çevirici bir kesme üreterek işletim sisteminin araya girmesini sağlar. Elbetteki bu düzenleme bir sayfanın IAS ile değiş-tokuş edilmesine neden olan bellek göndermelerinin sayısının, bir sayfadaki erişim sayısı ile karşılaştırıldığında düşük olması halinde verimlidir. Açıkçası, sayfalar mümkün olduğunca geniş olmalı ve veri iyi sıralanmalıdır. Ne yazık ki birçok pratik sistemde veri iyi-sıralanmış değildir ve sayfalar nispeten küçüktür - çoğu kez 2K veya 4K sözcük. Okur, sayfalama ile yerpaylaşmanın neredeyse aynı olduğunu düşünerek çelişkiye düşebilir. Bu, iki sebepten

dolayı doğru değildir. Çoğu zaman sadece bir yerpaylaşma alanı varken, sanal belleğin normal olarak büyük sayıda bir grup sayfaya genişletilebilir. Bunda programcı bir yerpaylaşma yazarken, yerpaylaşma alanı içine dizmek üzere ram ve verinin adreslerini kendisi seçer. Fakat sayfalama kullanıldığında, verisi tarafından üretilen mantıksal adresler, IAS'deki bilginin fiziksel adreslere karşılık gelecek şekilde değiştirilir. Yukarıdaki sayfalama düzenlemesi sanal belleğin daha genel bir biçime genişletilebilir.

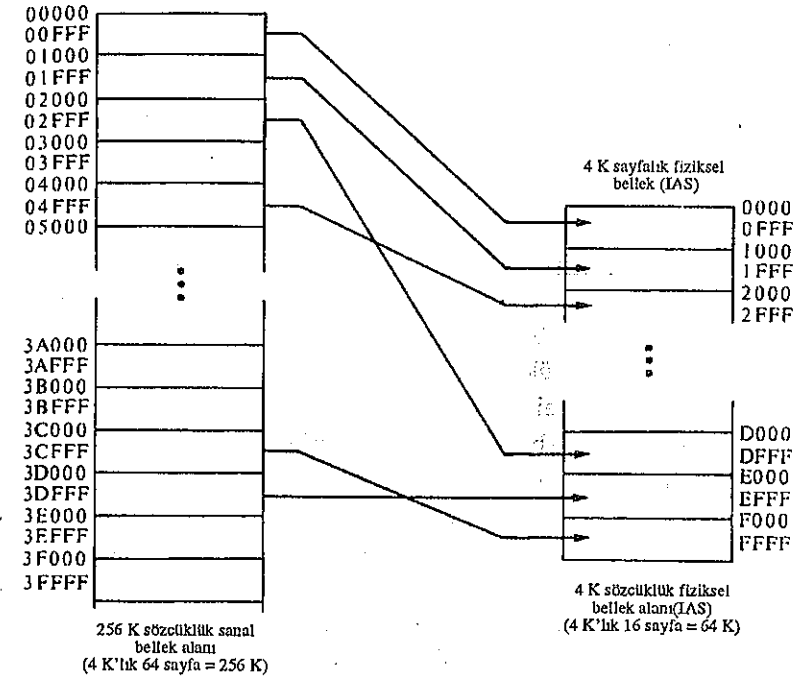
### Sanal bellek

Sanal bir bellek sisteminde programcı bütünüyle yüksek-hızlı IAS'den oluşan gibi görünen geniş bir fiziksel bellek dizisi (sanal bellek) görür. Gelecekte, fiziksel bellek küçük bir yüksek-hızlı RAM ile çok yavaş bir sıralı erişimli disk ortamıdır. Sanal belleğin iki avantajı vardır. Fiziksel belleğin normalde izin vermediğinden çok daha büyük programların yürütülmesini sağlar ve programcıya mevcut fiziksel adres aralığı içine düşen mantıksal adresleri seçme zorunluluğunu kurtarır. Programcı, programı ve değişkenleri için istediği mantıksal adreslere özgürlüğüne sahiptir. Seçtiği adreslerin hiç önemi yoktur çünkü mantıksal adresler işletim sistemi, uyduğunu gördüğü sürece otomatik olarak mevcut fiziksel bellek alanına eşlenirler.

Pratik sanal bellek sistemleri fiziksel rastgele-erişimli bellek alanını, tipik olarak 512'den 4096'ya kadar değişen sözcükten oluşan sayfalara bölerler. Bu, her bir anda IAS'de sanal bir programın birkaç parçasının bulunmasına izin verir. Açıkça görüldüğü gibi sürekli kullanılan veriyi IAS'de tutmak, gerektiğinde diskten getirmeden kaçınmak açısından daha mantıklıdır.

Şekil 9.9'da, sanal bellek alanı ile fiziksel bellek alanı arasındaki ilişki görülmektedir. Sanal bellek alanı, işlemcinin mantıksal bellek alanına karşılık gelir. Şekilde, 18-bitlik adres yolu olan bir MİB için toplam 256K sözcüğe kadar. Bu sanal adres alanı, 4K sözcüklük 64 sayfaya bölünür. İşlemcinin IAS'deki 4K sözcüklük fiziksel bellek, 4K'lık 16 sayfaya bölünür. Bu 4K bloklarının her biri bir sayfa-çerçevesi denir. Çünkü bunlar sanal bellekten bir sayfa tutarlar. Şimdi bir durumdayız: İşlemci 64 sayfasında herhangi bir yerde bulunan veriyi doğrudan adresleyebilir fakat bu sayfalardan sadece 16'sı aynı anda yüksek-hızlı IAS'de bulunabilir. Kalamı diskte saklanır.

Herhangi bir anda 16 sayfaya kadar sanal adres alanı IAS'de bulunabilir. Bir örnek Şekil 9.9'u daha da açıklığa kavuşturacaktır. MİB'in LDA DATA1'i yürütürken, verinin olduğunu ve DATA1'in mantıksal adresinin \$02000'den \$02FFF'e kadar olduğunu



Şekil 9.9. Sanal ve fiziksel adres alanı arasındaki ilişki

adreslere karşılık gelen 4K sözcüğün, fiziksel bellekte \$D000 - \$DFFF arasında saklandığını varsayalım. Bunun sonucu olarak \$02AED mantıksal işlenen adresi \$DAED fiziksel adresine çevrilmelidir.

Şekil 9.9'daki düzenleme iki soru doğurur. İlk olarak, MİB, bir işlenenin mantıksal adresini ne zaman üretir ve bu adres IAS'de işlenenin fiziksel adresine nasıl çevrilir? İkincisi, uygun sayfa IAS'de olmadığı için mantıksal adres hiçbir fiziksel adrese karşılık gelmezse ne olur?

İşlemci IAS'deki sayfaları işlemcinin kendi mantıksal adres alanına eşleyen bir sayfa-tablosu tutar (Şekil 9.10). İşlemci tarafından üretilen mantıksal adres, bir sayfadan bir konum seçen 12-bitlik bir adres ve belli bir sayfa seçen bir 6-bitlik sayfa adresi olmak üzere iki alandan oluşur. 12-bitlik ve 6-bitlik adresin birlikte 18-bitlik sanal (mantıksal) bellek alanına uzandığına dikkat edin. Şekil 9.10'dan görülebilir ki 000010 101011101101 mantıksal adresi sayfa-tablosunun ikinci kaydının (yani 000010) sorgulanmasına neden olur. Bu sayfa-çerçeve adresi 13'ü verir (yani \$D) ve veriye, IAS'de 1101 101011101101 konumundan erişilir.

Herhangi bir esnada IAS'de büyük bir sanal programın sadece bir bölümü bulunabileceğinden, IAS'de yeralmayacaktır. Yani, sayfa adresi sayfa tablosuna uygulandığında bu sayfaya karşılık gelen varlık biti sıfır olacaktır. Bu duruma sayfa-hatası denir ve burada işletim sistemi devreye girmelidir. Bir sayfa-hatası üretilir üretilmez, işletim sistemi eksik sayfayı saklama ortamından getirir, IAS'ye yükler ve buna bağlı olarak sayfa-tablosunu günceller. Sayfa-hatası, daha çok bir kesmeye benzer.

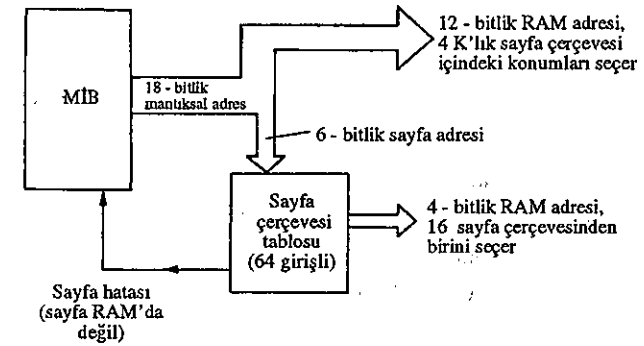
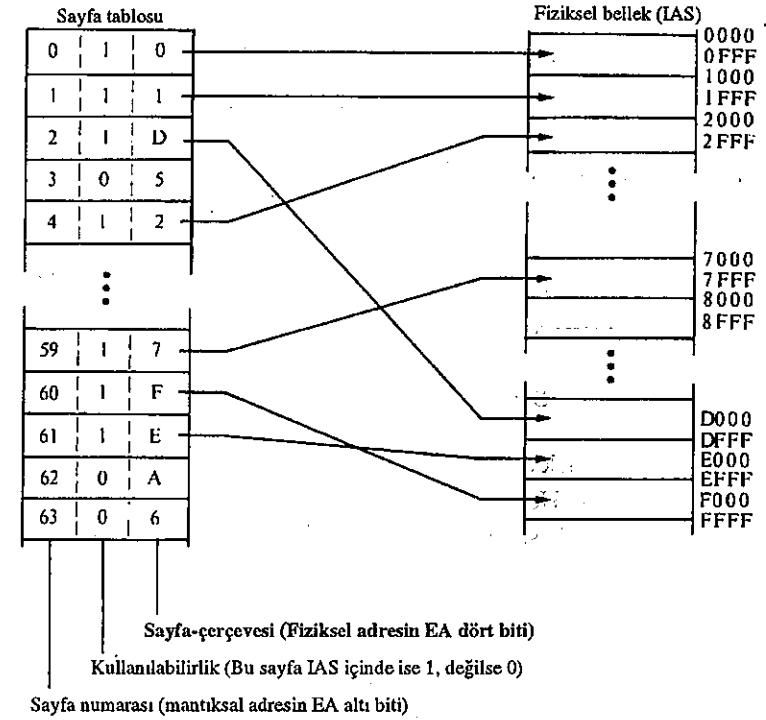
Bu işleme 'istek sayfalama' denir. Çünkü yeni bir sayfa istenene kadar getirilmez. IAS sayfalarla dolduğunda her bir yeni sayfa eski birinin üstüne yazılmalıdır. Bunu sürdürmenin olası bir yolu, yeni bir sayfanın diskten her getirilişinde en eski kullanılmış sayfayı gözden çıkarmaktır.

Gerçek sanal bellek sistemleri çok karmaşıktır ve gelişmiş donanım ve yazılıma gereksinim duyar. Henüz ele almadığımız bir problem de, bellek alanının iki kopya olmasıdır. Bir sayfa saklama ortamından doğrudan erişimli ortama her kopyalanışında, iki yerde var olur. Eğer MİB, IAS'indeki sayfayı hiç değiştirmezse sorun yoktur ve sayfanın herhangi bir sırada yeni bir sayfa tarafından üstüne yazılabilir. Öte yandan MİB bu sayfaya yazarsa, IAS'deki bu sayfa ile diskte buna karşılık gelen sayfa arasında bir farklılık ortaya çıkar. Bu şartlar altında işletim sistemi, güncelleştirilmiş versiyonu diske yazmadan; sayfayı, IAS'den dışarı-değiş-tokuş yapamaz. Birçok sistemde, her sayfanın kendisiyle ilgili bir 'kirli-bit' vardır. Sayfa-daki bir sözcük işlemci tarafından değiştirildiğinde, kirli-bit 1'e kurulur ve işletim sistemini, sayfa dışarıya-değiş-tokuş yapılırken diske tekrar-yazılması için uyarır.

Şimdiye dek sanal bellek tekniklerini sınırlı IAS probleminin bir çözümü olarak inceledik. Mantıksal adresi fiziksel adrese çevirme yeteneklerinin iki tane daha önemli çıkarımı vardır. Programcı, programının ve verisinin fiziksel bellekte nereye gideceğini düşünmekten tamamiyle kurtarılır. Programcı tarafından kullanılan tüm mantıksal adresler sayfa-tablosu tarafından IAS'deki adreslerine otomatik olarak çevrilecektir. Benzer şekilde bir çoklu programlama ortamında yeni programlar bitişik IAS bloklarına yerleştirilmek zorunda değildir; sayfa-tablosu adres çevirisi, bir programın IAS'de rastgele dağıtılmış sayfalara yerleştirilmesine izin verir.

### Sanal bellekler ve mikroişlemciler

İlk bakışta, mikroişlemcinin 64 K baytla sınırlı mantıksal adres alanından dolayı, sanal bellek tekniklerinin 16-bitlik adres yolu olan klasik 8-bitlik mikroişlemcilere



Şekil 9.10 Sayfa tablosu ve Şekil 9.9'a karşılık gelen adres eşleme

uygulanamayacağı görülebilir. Öte yandan, mikroişlemcinin mantıksal adres alanı bellek eşleme denen bir teknik ile genişletilebileceğinden ve sanal bellek tekniklerinin bir çoklu programlama ortamındaki çeşitli görevler için saklama ortamı ayırmada çok faydalı olduğundan, sanal belleklerin mikroişlemcilere uygulanması gereksinimi vardır.

Ne yazık ki sanal adres tekniklerini, klasik 8-bitlik mikroişlemcilere uygulamak son derecede zordur. Eğer o sırada IAS'de bulunan bir konuma erişim yapılırsa, otomatik donanımsal adres çeviricileri işlenenin mantıksal adresini fiziksel adresine birkaç nanosaniyede dönüştürür. Bu işlem MİB'e tamamiyle görünmezdir.

Şimdi, o esnada IAS'de olmayan bir mantıksal işlenen adresinin üretildiğini varsayalım. Adres çevirici donanım bu adresi gördüğünde, bir 'IAS'de değil durumu' (yani sayfa hatası) sinyaller ve uygun sayfanın diskten getirilmesi gerekir. Bu görünüşte basit sözün gerisinde iki problem gizlidir. O sıradaki komuta ne olur ve yeni sayfa diskten IAS'ye nasıl taşınır? RAM'da olmayan bir işlenene her erişildiğinde, o sıradaki komut ya iptal edilmeli ya yeniden çalıştırılmalı ya da işlenen IAS'ye gelene dek askıya alınmalıdır.

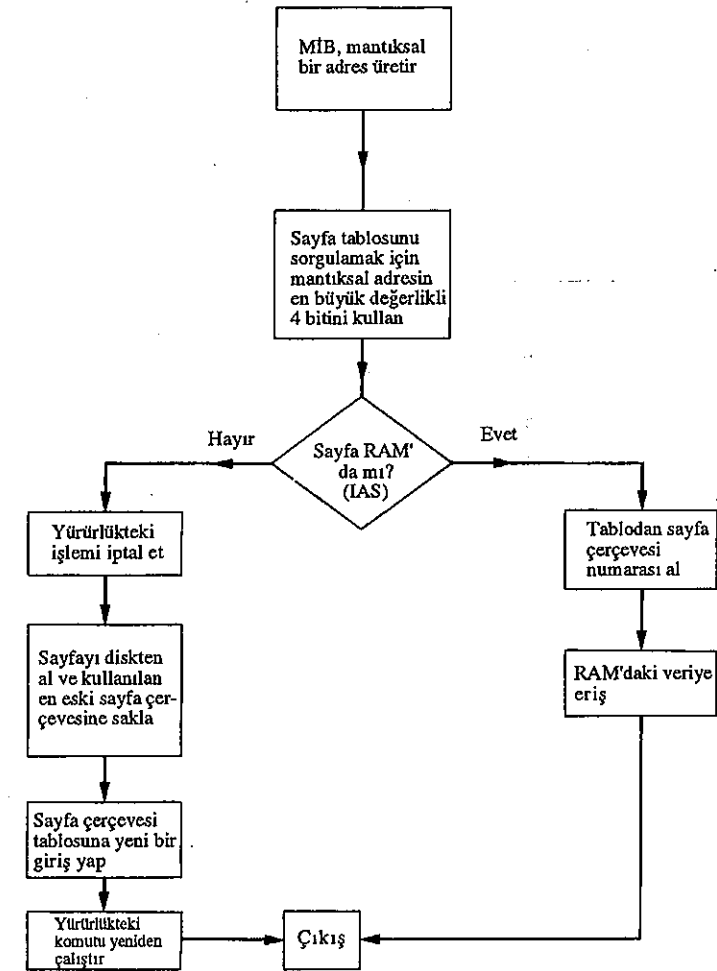
Eğer o sıradaki komut askıya alınmışsa, MİB iptal edilen komutu yeniden çalıştırmadan önce sayfayı kendisi getirebilir. Fakat komut, bekleme-durumuna giren bir MİB tarafından bekletilirse, sayfayı bir başka MİB getirmelidir. Şekil 9.11'de bu olaylar dizisi görülmektedir.

8-bitlik mikroişlemciler komutları iptal edip sonra yeniden çalıştırmak için gereken olanaklara sahip değildir. Eğer sanal bellekler 8-bitlik mikroişlemcilere uygulanacak olsaydı, işlenenin IAS'de bulunmadığı durumda MİB'in durumunu dondurmak ve diskten yeni sayfayı getirmek için ikinci bir mikroişlemci kullanmak gerekecekti. Sayfa IAS'ye yüklendikten sonra, ikinci işlemci uyumaya dönebilir ve dondurulmuş komutunu tamamlamasına izin verilen ana işlemci işlenenin IAS'de olduğunu bilir.

Bazı 16-bitlik mikroişlemciler sanal bellek tekniklerine karşı 8-bitlik rakiplerine göre daha uysaldır. Zilog ve Motorola'nın ikisi de artık komut-iptal bacakları olan özel 16-bitlik makine tipleri üretmektedir. Bunlar (bir sayfa-hatası üretildiğinde) anki komutu durdurur, makinenin durumunu saklar, o anki komut tarafından değiştirilen kaydedicileri komuttan önce bulundukları duruma geri alırlar. Bu işlem, kesmeye uğrayan komutun ardına yerleştirilme yerine, öncesine konulmak dışında klasik kesmeye benzerdir.

### Ön bellek

Sanal bellekte ele alınan temel kavramlar, herhangi bir hiyerarşik bellek yapısına da genişletilebilir. Özellikle, sanal bellek teknikleri bir bilgisayarın çalışmasını hızlandırmak için kullanılır. Sözgelimi, işlemcinin IAS'sinin erişim süresi, bilgi-



Şekil 9.11 Sayfa-hatası

sayarın çalışabildiği hızı sınırlıyor. Yani işlemcinin çevrim süresi IAS'nin çevrim süresinden daha küçüktür. Nispeten az miktarda çok yüksek hızlı IAS kullanarak, ana IAS'nin içerdigi değerlerin bir kısmını, daha hızlı IAS'ye yerleştirerek önemli bir verim elde edilir. Dolayısıyla, küçük yüksek hızlı belleğin daha yavaş IAS ile arasındaki ilişki RAM ile saklama ortamı arasındakiinkine aynıdır. Tipik olarak, yüksek hızlı IAS, 50 nsn'lik bir erişim süresine sahipken daha düşük hızlı IAS'lerin 400 nsn'lik bir erişim hızları vardır.

Önbellekler çoğu kez, MİB çevrim süresinin klasik mikroişlemcilerden daha kısa olduğu minibilgisayarlarda bulunur. Önbellek, o esnada içinde bulunan veri blokları tablosunun bir haritasını tutar. İşlemci tarafından mantıksal bir adres üretilirken en ağırlıklı bitler tabloyu sorgular ve önbellekteki verinin konumunu vermek üzere yeni bir adres üretir. Elbette ki, sanal bellek durumunda olduğu gibi, eğer veri önbellekte değilse, ön-belleğe, IAS'den yeni bir sayfa aktarılmalıdır.

Ön ve sanal belleğin ardındaki kavramlar soyut gibi görünse ve günlük yaşamın çok dışında gibi gelse de, aslında böyle değildir. Telefon rehberini ele alalım. Verilen bir bölgedeki tüm telefon numaralarını kapsar ve rehberin büyüklüğünden dolayı belli bir numarayı bulmak zaman alıcıdır. Pekçok insanın yaptığı şey, tüm önemli ve sıkça kullanılan numaraları rehberin başındaki boş bir sayfaya kayıtlı etmektir. Bu sayfa rehberdeki yüzbinlerce kayıttan 50'sini geçmez fakat tüm erişimlerin % 90'ı bu sayfada bulunabilir. Doğru tahmin ettiniz- bu sayfa bir önbellektir.

## EK A1: ÇÖZÜMLÜ ÖRNEKLER

Burada bir grup problem ile bunların çözümleri yer almaktadır. Sorular genellikle sayısal, cebir ve hesaplama esasına dayalı konular üzerine kuruludur. 'Ferrit-çekirdek belleğin çalışmasını açıklayın' gibi daha çok açıklamaya yönelik sorulardan, bu kitaptaki malzemenin bir kısmının doğrudan tekrarını gerektireceğinden kaçınılmıştır. Problemler, daha önceki çalışmanın tekrarından çok önceki bölümlerde işlenen malzemeye ek anlayış kazandırması için düşünülmüştür.

Bu problemleri ve çözümlerini eklemenin ikinci nedeni şudur: Yıllar geçtikçe öğrencilerin, derslerdeki konularla sınav sorularının amacını bütünüyle değerlendirmediklerini görüyorum. Öğretmenin yalnızca problemin çözülmüş olmasıyla ilgilendiklerini düşünürler. Aslında cevap önemli değildir. Değerlendirmedeki mantık, öğrencinin probleme nasıl yaklaştığıdır. Problemin son satırı çok fazla bir değere sahip değildir.

### Soru 1

Bir devrede 0000 (0)'dan 1111(15)'e dek onaltı doğal ikili sayıyı gösterimleyen dört giriş vardır; A, B, C, D. A en büyük değerlikli, D de en küçük değerlikli bittir. Devrenin çıkışı, F, eğer giriş 4, 5, 6 veya 7'nin katlarına tam bölünebiliyorsa doğrudur. Özel bir durum 15'tir. Bu durumda çıkış yanlışır. Sıfır, 4,5,6 veya 7 tam olarak bölünemez.

- Devreyi gösteren doğruluk tablosunu çizin.
- Doğruluk tablosundan Boole cebri teknikleriyle F için basitleştirilmiş bir çarpımlar-toplamı ifadesi çıkarın.
- Bir Karnaugh diyagramı çizin ve böylece F için basitleştirilmiş bir çarpımlar-toplamı ifadesi çıkarın.
- F'yi toplamların-çarpımı biçiminde ifade edin
- Sadece VEDEĞİL kapıları kullanarak F'yi gerçekleştirecek bir mantık devresini tasarlayın.

## Çözüm 1

(a) Q1'in doğruluk tablosu Tablo A1.1'de verilmiştir.

(b) Tablo A1.1'den aşağıdaki terimlerin toplamını yazarak F için bir çarpımların toplamı elde edilebilir.

$$F = \overline{A}\overline{B}\overline{C}D + \overline{A}B\overline{C}D + \overline{A}BC\overline{D} + \overline{A}BCD + A\overline{B}\overline{C}D + A\overline{B}C\overline{D} + ABC\overline{D} + ABCD$$

Boole cebiri ile ifade şu şekilde basitleştirilebilir:

Tablo A1.1 Çözüm 1(a) : Doğruluk Tablosu

| Girişler |   |   |   | Sayı değeri | F | Açıklama                                  |
|----------|---|---|---|-------------|---|-------------------------------------------|
| A        | B | C | D |             |   |                                           |
| 0        | 0 | 0 | 0 | 0           | 0 |                                           |
| 0        | 0 | 0 | 1 | 1           | 0 |                                           |
| 0        | 0 | 1 | 0 | 2           | 0 |                                           |
| 0        | 0 | 1 | 1 | 3           | 0 |                                           |
| 0        | 1 | 0 | 0 | 4           | 1 | 4'le bölünebilir                          |
| 0        | 1 | 0 | 1 | 5           | 1 | 5'le bölünebilir                          |
| 0        | 1 | 1 | 0 | 6           | 1 | 6'yla bölünebilir                         |
| 0        | 1 | 1 | 1 | 7           | 1 | 7'yle bölünebilir                         |
| 1        | 0 | 0 | 0 | 8           | 1 | 4'le bölünebilir                          |
| 1        | 0 | 0 | 1 | 9           | 0 |                                           |
| 1        | 0 | 1 | 0 | 10          | 1 |                                           |
| 1        | 0 | 1 | 1 | 11          | 0 | 5'le bölünebilir                          |
| 1        | 1 | 0 | 0 | 12          | 1 |                                           |
| 1        | 1 | 0 | 1 | 13          | 0 | 6'yla bölünebilir                         |
| 1        | 1 | 1 | 0 | 14          | 1 |                                           |
| 1        | 1 | 1 | 1 | 15          | 0 | 7'yle bölünebilir<br>tanımı gereği yanlış |

$$\begin{aligned} F &= \overline{A}\overline{B}(\overline{C}+D) + \overline{A}B(\overline{C}+D) + A\overline{B}(\overline{C}+C) + AB(\overline{C}+C) \\ &= \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}D + AB\overline{D} \\ &= \overline{A}B(\overline{C}+C) + A\overline{B}(\overline{C}+C) \\ &= \overline{A}B + A\overline{B} \end{aligned}$$

(c) F için Karnaugh diyagramı Tablo A1.2(a)'da verilmiştir. 1'lerin kapladığı kareler dörderli iki grupta toplanabilir (Tablo A1.2(b)). Bu  $F = \overline{A}B + A\overline{B}$  ifadesini verir ki bu da yukarıda (b) bölümünde elde edilen cevap ile (sağlama yaparcasına) aynıdır.

(d) Toplamlar-çarpımı ifadesini elde etmek için F'nin çarpımların toplamı biçiminde tümleyenini alıp sonra da bunu tümlemek gerekir.

Tablo A1.2(a)

| AB | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| CD |    |    |    |    |
| 00 |    | 1  | 1  | 1  |
| 01 |    | 1  |    |    |
| 11 |    | 1  |    |    |
| 10 |    | 1  | 1  | 1  |

Tablo A1.2(b)

| AB | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| CD |    |    |    |    |
| 00 |    | 1  | 1  | 1  |
| 01 |    | 1  |    |    |
| 11 |    | 1  |    |    |
| 10 |    | 1  | 1  | 1  |

$$F = \overline{A}B + A\overline{B}$$

$$F = \overline{A}B + A\overline{B}$$

$$= (A+B)(\overline{A}+D) \quad (\text{toplamların-çarpımı biçimindeki tümleyen})$$

$$= A\overline{A} + A\overline{B} + \overline{A}B + \overline{B}D$$

$$= A\overline{B} + \overline{A}B + \overline{B}D$$

$$= A\overline{B} + \overline{A}B$$

$$F = A\overline{B} + \overline{A}B$$

$$= (\overline{A}+D)(A+B) \quad (\text{çarpımların toplamı biçimindeki tümleyen})$$

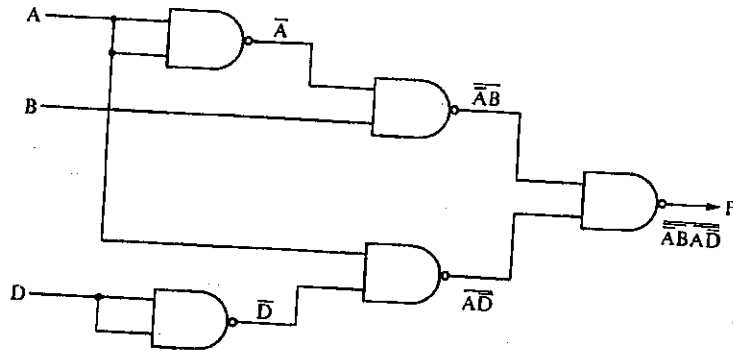
$$= (\overline{A}+D)(A+B) \quad (\text{istenen toplamların çarpımı biçimindeki fonksiyon})$$

Çarpımlar-toplamı biçimindeki F'nin tümleyeninin, 0'ların kapladığı alanları dikkate alarak F'nin Karnaugh diyagramından doğrudan doğruya elde edilebileceğine dikkat edin.

(e)  $F = \overline{A}B + A\overline{B}$  ifadesini VEDEĞİL mantık biçimine dönüştürmek için ' + ' ortadan kaldırılmalıdır.

$$F = \overline{\overline{F}} = \overline{\overline{A}B + A\overline{B}} = \overline{\overline{A}B} \cdot \overline{A\overline{B}}$$

$\overline{A}$  ve  $\overline{B}$  fonksiyonlarının tersi, girişleri birbirine bağlanan ikigirişli VEDEĞİL kapıları ile üretilebilir. Şekil A1.1'deki devre F'i sadece VEDEĞİL mantığı gerçeğe getirmektedir.

Şekil A1.1 F =  $\overline{A}B + A\overline{D}$ 'nin VEDEĞİL mantığıyla gerçekleştirilmesi

## Soru 2

8-bitlik bir kaydırmalı kaydedici, aşağıdaki işlemleri yerine getirebilir

- Her katı 8-bitlik veri yolundan yükle (paralel yükleme)
- Mantıksal sola kaydırma (0 girer, EBDB kaybolur)
- Mantıksal sağa kaydırma (0 girer, EKDB kaybolur)
- Aritmetik sola kaydırma (mantıksal sola kaydırma ile aynı)
- Aritmetik sağa kaydırma (EBDB kopyalanır, EKDB kaybolur)
- Dairesel sola kaydırma (EBDB, EKDB konumuna gider)
- Dairesel sağa kaydırma (EKDB, EBDB konumuna gider)

Devre, 8 tane Ana-Uydu JK Flip-Flop'undan meydana gelir ve yukarıdaki (i) ve (vii) işlemlerinin düşen kenarında yapılmasına neden olan bir saat girişi vardır. Aşağıdaki gibi, beş denetleme girişi vardır:

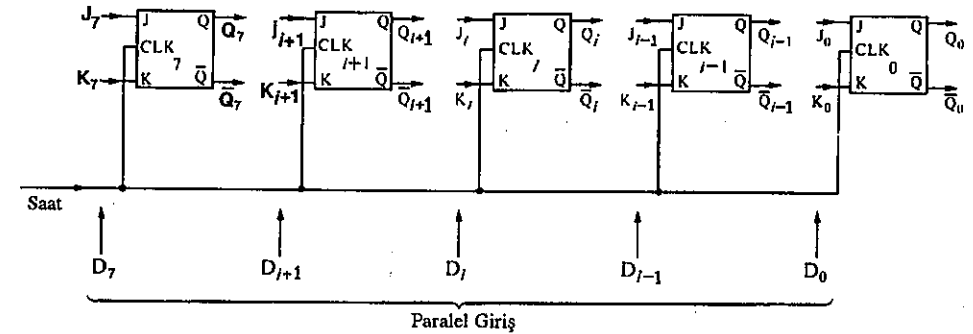
- S: S=1 ise bir kaydırma işlemi yap S=0 ise bir paralel yükleme  
 R: R=1 ise sağa kaydır R=0 sola kaydır (S=1 ise)  
 L: L=1 ise mantıksal bir kaydırma yap (S=1 ise)  
 A: A=1 ise aritmetik bir kaydırma yap (S=1 ise)  
 C: C=1 ise dairesel bir kaydırma yap (S=1 ise)

L, A ve C'nin kural dışı bir kombinasyonunun oluşamayacağına ve bu yüzden de L, A ve C'nin birden daha fazlasının hiçbir zaman aynı anda doğru olamayacağına dikkat edin.

Kaydırmalı kaydedicinin sekiz durumu için; de J ve K için R, S, L, A, C ve Flip-flop'ların çıkışları cinsinden birer cebirsel ifade elde ediniz.

## Çözüm 2

Şekil A1.2'de, beş katlı kaydırmalı kaydedici gösterilmiştir. Bunlar uc katlarıdır:  $Q_7$  ve  $Q_0$  ve sırasıyla en büyük değerlikli ve en küçük değerlikli bit katlarıdır. Uc olmayan kat, bir kat,  $Q_i$ , soldaki  $Q_{i-1}$  komşusu ve sağdaki  $Q_{i+1}$  komşusu ile birlikte ele alınmalıdır.



Şekil A1.2: Beş katlı kaydırmalı kaydedici

0 ve 7 dışındaki tüm katlar aynı işlevleri yerine getirirler: Paralel yükleme, sağa kaydırma ve sola kaydırma. JK flip-flop'ları, veriyi daima kaynakların birinden ya da diğerinden yüklediği için, sadece J=1, K=0 veya J=0, K=1 girişleri dikkate alınmalıdır. Bunun sonucu olarak,  $J=\overline{K}$  ve sadece J için bir ifade türetmemiz gerekir. Buna karşılık gelen K değerleri bir tersleyiciden elde edilebilir.

İ. kat:

|                 |                 |          |
|-----------------|-----------------|----------|
| Paralel yükleme | $J_i = D_i$     | S=0      |
| Sağa kaydırma   | $J_i = Q_{i+1}$ | S=1, R=1 |
| Sola kaydırma   | $J_i = Q_{i-1}$ | S=1, R=0 |

Böylece  $J_i = \overline{S}D_i + S(RQ_{i+1} + \overline{R}Q_{i-1})$

0. kat (EDDB)

|                 |                       |               |
|-----------------|-----------------------|---------------|
| Paralel yükleme | $J_0 = D_0$           | S=0           |
| Sağa kaydırma   | mantıksal $J_0 = Q_1$ | S=1, R=1, L=1 |
|                 | aritmetik $J_0 = Q_1$ | S=1, R=1, A=1 |
|                 | daireysel $J_0 = Q_1$ | S=1, R=1, C=1 |

|               |           |           |                 |
|---------------|-----------|-----------|-----------------|
| Sola kaydırma | mantıksal | $J_0=0$   | $S=1, R=0, L=1$ |
|               | aritmetik | $J_0=0$   | $S=1, R=1, A=1$ |
|               | dairesel  | $J_0=Q_7$ | $S=1, R=1, C=1$ |

$$\begin{aligned} \text{Böylece } J_0 &= \bar{S}D_0 + S(RLQ_1 + RAQ_1 + RCQ_1 + \bar{R}CQ_7) \\ &= \bar{S}D_0 + S(RQ_1 + \bar{R}CQ_7) \end{aligned}$$

Bu ifade basitleşir, çünkü  $S=1$  iken kaydırmanın bir ya da diğer türü yer alıyor olduğundan  $L + A + C$  doğru olmalıdır.

### 7. Kat (EBDB)

|                 |           |           |                 |
|-----------------|-----------|-----------|-----------------|
| Paralel yükleme |           | $J_7=D_7$ | $S=0$           |
| Sağa kaydırma   | mantıksal | $J_7=0$   | $S=1, R=1, L=1$ |
|                 | aritmetik | $J_7=Q_7$ | $S=1, R=1, A=1$ |
|                 | dairesel  | $J_7=Q_0$ | $S=1, R=1, C=1$ |
| sola kaydırma   | mantıksal | $J_7=Q_6$ | $S=1, R=0, L=1$ |
|                 | aritmetik | $J_7=Q_6$ | $S=1, R=0, A=1$ |
|                 | dairesel  | $J_7=Q_6$ | $S=1, R=0, C=1$ |

$$\begin{aligned} \text{Böylece } J_7 &= \bar{S}D_7 + S(RAQ_7 + RCQ_0 + \bar{R}LQ_6 + \bar{R}AQ_6 + \bar{R}CQ_6) \\ &= \bar{S}D_7 + S(R(AQ_7 + CQ_0) + \bar{R}Q_6). \end{aligned}$$

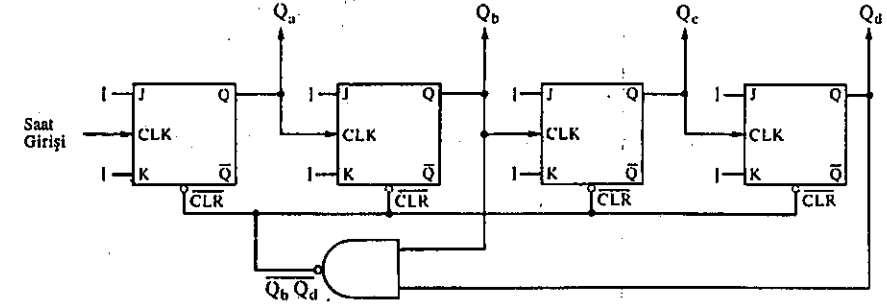
### Soru 3

Çevrimsel olarak 0'dan 9'a sayan 4-bit onluk bir sayıcı tasarlayın. Koşulsuz bir alçakta-etkin sıfırlama girişi ile JK Ana-uydu flip-flopları kullanın. Devrenin çalışmasını belirtmek için zamanlama diyagramlarını çizin.

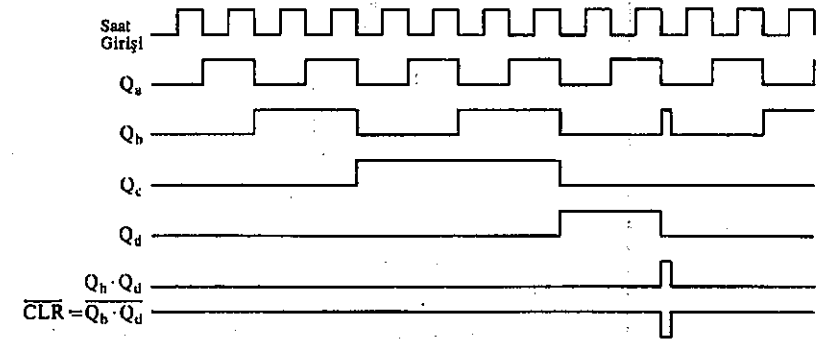
### Çözüm 3

Onluk bir sayıcı; uygun noktalarda sıfırlamak yoluyla ikili bir sayıcıdan türetilir. 4-katlı bir ikili sayıcı 0000'dan 1111'e (yani 0'dan 15'e) sayar. Onluk bir sayıcı yaratmak için 10 durumu (1010) algılanmalı ve flip-flopları sıfırlamak için kullanılmalıdır.

İkili sayıcı normalde 0'dan 9'a kadar sayar. Onuncu sayımda  $Q_a=1$  ve  $Q_b=1$ 'dir. Bu, çıkışı alçak düzeye giderek flip-flopları sıfırlayan VEDEĞİL kapısı ile algılanır. On sayımı bir an için varolur (Şekil A1.3(a), (b)'ye bakınız).



Şekil A1.3(a)



Şekil A1.3(b)

### Soru 4

A ve B gibi iki tane  $n$ -bit ikinin tümleyenini sayısı, S gibi bir toplamı vermek üzere toplandığında; aritmetik taşma olasılığı ortaya çıkar. Aritmetik taşmanın tanımı şudur:

$$TA = s_{n-1}a_{n-1}b_{n-1} + s_{n-1}a_{n-1}b_{n-1}$$

Burada  $s_{n-1}$ ,  $a_{n-1}$ , ve  $b_{n-1}$ , sırasıyla, toplamın, A'nın ve B'nin en değerlikli bitleridir.

Bu denklem genellikle gerçek sistemlerde taşmayı algılamak için kullanılmaz çünkü  $a_{n-1}$  ve  $b_{n-1}$ 'in saklanması gerektirir ki bunlardan biri normalde zaten toplama ile silinirler.

Gerçekte taşmayı algılama metodu; paralel toplayıcının en anlamlı katına giriş-el-

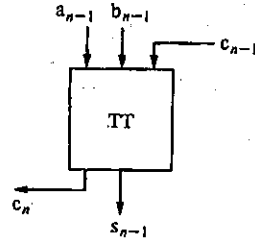
desini, yine aynı katdan çıkan çıkış-eldesi ile karşılaştırmaktır. Eğer bunlar farklıysa taşmanın olduğu söylenir. Yani;

$$TA = c_n \overline{c_{n-1}} + \overline{c_n} c_{n-1}$$

Paralel toplayıcının elde bitlerini inceleyerek taşmayı algılayan yukarıdaki yöntemin geçerli olduğunu ispat edin.

#### Çözüm 4

Paralel toplayıcının en önemli katı, Şekil A1.4'de verildiği gibi gösterilebilir.



Şekil A1.4

EBDB katı  $s_{n-1}$  toplam biti ile  $c_n$  çıkış-eldesini üretmek üzere  $a_{n-1}$ ,  $b_{n-1}$ , ve  $c_{n-1}$ 'i toplar. A ve B'nin olası dört toplanabilme kombinasyonu vardır:

- (+A) + (+B)
- (+A) + (-B)
- (-A) + (+B)
- (-A) + (-B)

Farklı işaretli iki sayıyı toplamak aritmetik taşma ile sonuçlanamayacağından, sadece A ve B'nin ikisinin de pozitif veya negatif durumları ele almamız gerekir.

1. Durum : A ve B pozitif  $a_{n-1} = 0$ ,  $b_{n-1} = 0$

Son kat  $a_{n-1}$  ve  $b_{n-1}$ 'in ikisi de 0 olduğundan  $c_{n-1}$  sonucunu vermek üzere  $a_{n-1}$ ,  $b_{n-1}$  ve  $c_{n-1}$ 'i toplar. Bu yüzden  $c_n$  elde-çıkışı, sıfırdır ve  $s_{n-1} = c_{n-1}$ . Taşmanın  $s_{n-1} = 1$  olduğunda meydana geleceğini biliyoruz; bu nedenle taşma; eğer toplam negatifse ve  $\overline{c_n} c_{n-1} = 1$  ise oluşur.

2. Durum: A ve B negatif.  $a_{n-1} = 1$ ,  $b_{n-1} = 1$

Son kat  $s_{n-1} = c_{n-1}$  toplamını ve çıkış-eldesi  $c_n = 1$  elde etmek üzere  $a_{n-1} + b_{n-1} + c_{n-1} = 1 + 1 + c_{n-1}$ 'i toplar. Eğer toplam pozitifse ve  $s_{n-1} = 0$  ise taşma oluşur. Yani  $c_n \overline{c_{n-1}} = 1$  ise.

İki durumunu dikkate alarak, eğer  $\overline{c_n} c_{n-1} + c_n \overline{c_{n-1}} = 1$  ise taşma meydana gelir.

#### Soru 5

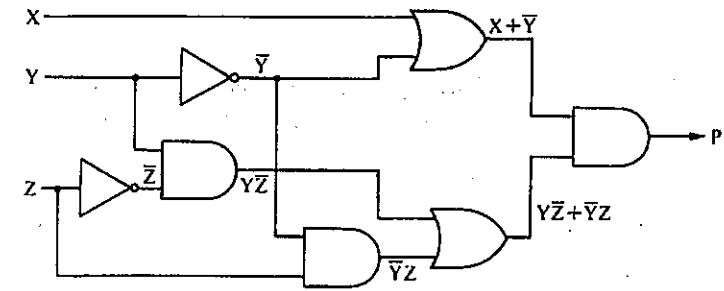
(a) Yalnızca VE, VEYA ve DEĞİL kapılarını kullanarak  $P = (X + \overline{Y})(Y \oplus Z)$  ve  $Q = \overline{Y}Z + XY\overline{Z}$  fonksiyonlarını, X, Y, Z girişlerine bağlı olarak üreten devre diyagramlarını tasarlayın. Bu ifadeleri basitleştirmeyin veya başka biçimlere çevirmeyin.

(b) Doğruluk tablosu yardımıyla P ve Q arasındaki ilişkiyi belirtin.

(c) Hız (Yayınım gecikmesi) ve gerçekleştirme maliyeti açısından P ve Q devre diyagramlarını karşılaştırın.

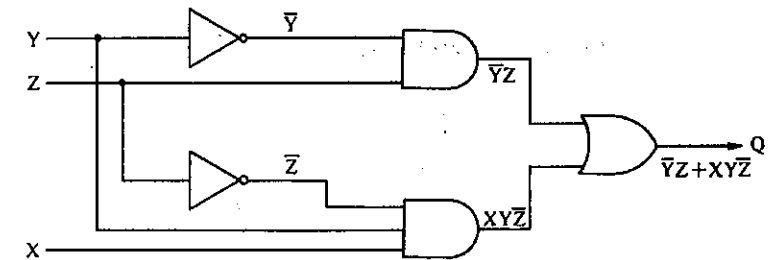
#### Çözüm 5 (a)

Şekil A1. 5(a)'da  $P = (X + \overline{Y})(Y \oplus Z) = (X + \overline{Y})(Y\overline{Z} + \overline{Y}Z)$ 'nin devre diyagramı verilmiştir.



Şekil A1.5 (a)

Şekil A1.5(b)'de ise  $Q = \overline{Y}Z + XY\overline{Z}$ 'nin devre diyagramı yer almaktadır.



Şekil A1.5 (b)

(b) Tablo A1.3 P ve Q'nun doğruluk tablosudur. Doğruluk tablosundan  $P=Q$  görülebilir.

Tablo A1.3 P ve Q için doğruluk tablosu

| X | Y | Z | $X+\bar{Y}$ | $Y\oplus Z$ | $P=(X+\bar{Y})(Y\oplus Z)$ | $\bar{Y}Z$ | $XYZ$ | $Q=\bar{Y}Z+XYZ$ |
|---|---|---|-------------|-------------|----------------------------|------------|-------|------------------|
| 0 | 0 | 0 | 1           | 0           | 0                          | 0          | 0     | 0                |
| 0 | 0 | 1 | 1           | 1           | 1                          | 1          | 0     | 1                |
| 0 | 1 | 0 | 0           | 1           | 0                          | 0          | 0     | 0                |
| 0 | 1 | 1 | 0           | 0           | 0                          | 0          | 0     | 0                |
| 1 | 0 | 0 | 1           | 0           | 0                          | 0          | 0     | 0                |
| 1 | 0 | 1 | 1           | 1           | 1                          | 1          | 0     | 1                |
| 1 | 1 | 0 | 1           | 1           | 1                          | 0          | 1     | 1                |
| 1 | 1 | 1 | 1           | 0           | 0                          | 0          | 1     | 1                |

(c) iki devrenin karşılaştırılması

**Yayılma Gecikmesi:** P devresindeki maksimum gecikme, Y yolundaki ardarda bağlı bulunan 4 kapıdır (yani DEĞİL kapısı, VE kapısı, VEYA kapısı, VE kapısı). Q devresindeki maksimum gecikme, Y ve Z yollarındaki 3 seri kapıdır (yani DEĞİL kapısı, VE kapısı, VEYA kapısı). Bu yüzden Q devresi P'den %33 daha hızlıdır.

**Maliyet:** P'yi gerçekleştirmek için gereken toplam kapı sayısı 7'dir. Q'yu gerçekleştirmek için gereken toplam kapı sayısı ise 5'tir. P devresindeki toplam giriş sayısı 12'dir. Q devresindeki toplam giriş sayısı ise 9'dur.

Açıkça görüldüğü gibi Q devresi P'den kapı sayısı ve kapılara giriş sayısı açısından daha iyidir.

#### Soru 6

(a) Özel VEYA (EOR) işleminin birleşme özelliği, yani  $A\oplus(B\oplus C)=(A\oplus B)\oplus C$  olduğunu gösterin,

(b) Herhangi bir mantık devresinin sadece Özel VEYA ve VE kapılarıyla gerçekleştirilebileceğini gösterin.

#### Çözüm 6

$$\begin{aligned}
 (a) \quad A\oplus(B\oplus C) &= A\oplus(\bar{B}\bar{C}+\bar{B}C) \\
 &= A(\bar{B}\bar{C}+\bar{B}C)+\bar{A}(\bar{B}\bar{C}+\bar{B}C) \\
 &= A(\bar{B}+C)(B+\bar{C})+\bar{A}\bar{B}\bar{C}+\bar{A}\bar{B}C
 \end{aligned}$$

$$\begin{aligned}
 &= A(\bar{B}\bar{C}+BC)+\bar{A}\bar{B}\bar{C}+\bar{A}\bar{B}C \\
 &= \bar{A}\bar{B}\bar{C}+ABC+\bar{A}\bar{B}\bar{C}+\bar{A}\bar{B}C \\
 (A\oplus B)\oplus C &= (\bar{A}\bar{B}+\bar{A}B)\oplus C \\
 &= (\bar{A}\bar{B}+\bar{A}B)\bar{C}+(\bar{A}\bar{B}+\bar{A}B)C \\
 &= \bar{A}\bar{B}\bar{C}+\bar{A}B\bar{C}+(\bar{A}\bar{B}+\bar{A}B)C \\
 &= \bar{A}\bar{B}\bar{C}+\bar{A}B\bar{C}+\bar{A}\bar{B}C+ABC.
 \end{aligned}$$

Bu ifadelerin ikisi de birbirine eşittir ve bundan dolayı  $\oplus$  işlemi birleşme özelliğine sahiptir.

(b)  $F=A\oplus B$  olarak ele alalım.

$$F=\bar{A}\bar{B}+\bar{A}B$$

$$A=0 \text{ ise } F=B$$

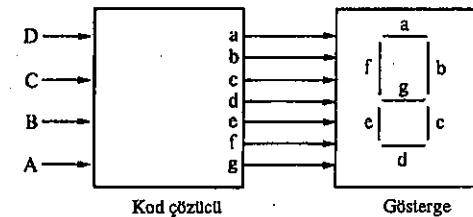
$$A=1 \text{ ise } F=\bar{B}$$

Bir başka deyişle Özel-VEYA kapısının bir girişine mantık 1 bağlanırsa öteki giriş, çıkışta tümlenmiş biçimde görülecektir. Bu nedenle EOR kapısı tersleyici olarak davranabilir.

Tersleyici olarak bağlanan bir Özel-VEYA kapısı, bir VE kapısının çıkışına uygulanırsa; VE kapısı VEDEĞİL kapısına dönüştürülmüş olur. Tüm mantık fonksiyonları sadece VEDEĞİL kapılarıyla üretilebileceğinden, tüm mantık fonksiyonları VE kapıları ile tersleyici olarak davranan Özel-VEYA kapıları aracılığıyla üretilebilir.

#### Soru 7

Bir BCD/7-parçalı kod çözücü tasarlayınız. Kod çözücü DCBA ile gösterilen 4-bitlik doğal ikili BCD girişine sahiptir. Burada A en küçük değerlikli bittir. BCD girişinin hiçbir zaman 9'dan büyük olamayacağını varsayın. Yedi-parçalı kod çözücü Şekil A1.6'da gösterilmiştir; ve yedi çıkışından biri (a'dan g'ye kadar) doğru olduğunda, görüntünün buna karşılık gelen parçası aydınlatılır.



Şekil A1.6

## Çözüm 7

Bu problemin doğruluk tablosu Tablo A1.4'te verilmiştir. Şimdi denklemi a'dan g'ye kadar parçalar için çözebiliriz. Karnaugh diyagramı kullanarak, dikkate alınmaz durumlar belirlenir. a parçası için Karnaugh diyagramı Tablo A1.5(a) ve (b)'de gösterilmiştir. Bu diyagramdan,  $a = CA + BA + D + \bar{C}B$  elde edilir.

Tablo A1.4

| Girişler |   |   |   | Karakter | Çıkışlar |   |   |   |   |   |   |         |
|----------|---|---|---|----------|----------|---|---|---|---|---|---|---------|
| D        | C | B | A |          | a        | b | c | d | e | f | g |         |
| 0        | 0 | 0 | 0 | 0        | 1        | 1 | 1 | 1 | 1 | 1 | 0 |         |
| 0        | 0 | 0 | 1 | 1        | 0        | 1 | 1 | 0 | 0 | 0 | 0 |         |
| 0        | 0 | 1 | 0 | 2        | 1        | 1 | 0 | 1 | 1 | 0 | 1 |         |
| 0        | 0 | 1 | 1 | 3        | 1        | 1 | 1 | 1 | 0 | 0 | 1 |         |
| 0        | 1 | 0 | 0 | 4        | 0        | 1 | 1 | 0 | 0 | 1 | 1 |         |
| 0        | 1 | 0 | 1 | 5        | 1        | 0 | 1 | 1 | 0 | 1 | 1 |         |
| 0        | 1 | 1 | 0 | 6        | 0        | 0 | 1 | 1 | 1 | 1 | 1 |         |
| 0        | 1 | 1 | 1 | 7        | 1        | 1 | 1 | 0 | 0 | 0 | 0 |         |
| 1        | 0 | 0 | 0 | 8        | 1        | 1 | 1 | 1 | 1 | 1 | 1 |         |
| 1        | 0 | 0 | 1 | 9        | 1        | 1 | 1 | 0 | 0 | 1 | 1 |         |
| 1        | 0 | 1 | 0 |          | X        | X | X | X | X | X | X | Yasaklı |
| 1        | 0 | 1 | 1 |          | X        | X | X | X | X | X | X |         |
| 1        | 1 | 0 | 0 |          | X        | X | X | X | X | X | X |         |
| 1        | 1 | 0 | 1 |          | X        | X | X | X | X | X | X |         |
| 1        | 1 | 1 | 0 |          | X        | X | X | X | X | X | X |         |
| 1        | 1 | 1 | 1 |          | X        | X | X | X | X | X | X |         |
| 1        | 1 | 1 | 1 |          | X        | X | X | X | X | X | X |         |

a'yı elde etmenin alternatif bir yolu da Tablo A1.5 (c), (d)'de gösterilen diyagramdaki sıfırları ele almaktır. Bu diyagramdan,  $\bar{a} = \bar{D} \bar{C} \bar{B} A + \bar{C} \bar{A}$ .

$$\begin{aligned}
 \text{Böylece } a &= \overline{\bar{D} \bar{C} \bar{B} A + \bar{C} \bar{A}} = (D + C + B + \bar{A})(\bar{C} + A) \text{ 'dır.} \\
 &= D\bar{C} + DA + C\bar{C} + CA + \bar{C}B + BA + \bar{C}\bar{A} + \bar{A}A \\
 &= D\bar{C} + DA + CA + \bar{C}B + BA + \bar{C}\bar{A} \\
 &= D\bar{C} + CA + BA + \bar{C}\bar{A}.
 \end{aligned}$$

Bu ifadenin ilk elde edilen ifadenin üstüne getirdiği bir gelişme yoktur.

b parçasının Karnaugh diyagramı Tablo A1.6(a), (b)'de gösterilmiştir ve

$$b = \bar{C} + \bar{B} \bar{A} + BA \text{ ifadesini verir.}$$

Tablo A1.5(a)

| DC | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| BA |    |    |    |    |
| 00 | 1  |    | X  | 1  |
| 01 |    | 1  | X  | 1  |
| 11 | 1  | 1  | X  | X  |
| 10 | 1  |    | X  | X  |

Tablo A1.5(b)

| DC | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| BA |    |    |    |    |
| 00 | 1  |    | X  | 1  |
| 01 |    | 1  | X  | 1  |
| 11 | 1  | 1  | X  | X  |
| 10 | 1  |    | X  | X  |

Tablo A1.5(c)

| DC | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| BA |    |    |    |    |
| 00 |    | 0  | X  |    |
| 01 | 0  |    | X  |    |
| 11 |    |    | X  | X  |
| 10 |    | 0  | X  | X  |

Tablo A1.5(d)

| DC | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| BA |    |    |    |    |
| 00 |    | 0  | X  |    |
| 01 | 0  |    | X  |    |
| 11 |    |    | X  | X  |
| 10 |    | 0  | X  | X  |

b'nin Karnaugh diyagramında sıfırları çizerek (Tablo A1.6(c) (d))

$$\bar{b} = \bar{C}\bar{B}A + C\bar{B}\bar{A} \text{ ifadesini elde ederiz.}$$

$$\begin{aligned}
 \text{Böylece } b &= \overline{\bar{C}\bar{B}A + C\bar{B}\bar{A}} = (\bar{C} + B + \bar{A})(\bar{C} + \bar{B} + A) \\
 &= \bar{C} + BA + \bar{B}\bar{A}
 \end{aligned}$$

Bu ifade, Karnaugh diyagramındaki birler dikkate alınırsa doğrudan elde edilen sonuç ile aynıdır.

Tablo A1.6(a)

| DC | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| BA |    |    |    |    |
| 00 | 1  | 1  | X  | 1  |
| 01 | 1  |    | X  | 1  |
| 11 | 1  | 1  | X  | X  |
| 10 | 1  |    | X  | X  |

Tablo A1.6(b)

| DC | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| BA |    |    |    |    |
| 00 | 1  | 1  | X  | 1  |
| 01 | 1  |    | X  | 1  |
| 11 | 1  | 1  | X  | X  |
| 10 | 1  |    | X  | X  |

Tablo A1.6(c)

| DC | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| BA |    |    |    |    |
| 00 |    |    | X  |    |
| 01 |    | 0  | X  |    |
| 11 |    |    | X  | X  |
| 10 |    | 0  | X  | X  |

Tablo A1.6(d)

| DC | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| BA |    |    |    |    |
| 00 |    |    | X  |    |
| 01 |    | 0  | X  |    |
| 11 |    |    | X  | X  |
| 10 |    | 0  | X  | X  |

Kalan beş parça için denklemler de benzer şekilde ele alınabilir.

### Soru 8

Klavyeden ASCII-kodlanmış iki onaltılı karakteri girmek ve bu iki onaltılık karakterin 8-bitlik ikili biçimini akümülatörde çıkarmak için bir altıyordam yazın. Örneğin, 'E' ve '4' tuşlarına basmak (sırasıyla \$45 ve \$34 ASCII kodlarda) akümülatörde \$E4 (yani 11100100)'ın dönmesine neden olur. Eğer ondalık-olma-

yan bir karakter (yani 0-9 ve A-F arasında olmayan bir tuş) girilirse; PSW'nun elde-biti, dönmeden önce kurulu. \$E993'teki INALL adlı altıyordamın klavyeden bir ASCII karakter okumak için kullanılabileceğini varsayın.

### Çözüm 8

Bu program, ASCII-kodlu iki karakterin girilmesini, bunların onaltılık biçime dönüştürülmesini ve bunların tek bir bayta paketlenmesini gerektirir. Şekil 3.1'de, ASCII karakterler ile onaltılı değerleri arasındaki ilişki gösterilmiştir. 0-9 ve A-F karakterlerinin sonuçları aşağıda tekrarlanmıştır.

ASCII karakter    0 1 ... 9 A ... F

Onaltılı değer    30 31 ... 39 41 ... 46

Bu tablodan görülebileceği gibi 0'dan 9'a kadar sayısal değerlerin, ASCII karakterlerinin kendileriyle onaltılı 30 değerinin toplanmasına eşittir. A'dan F'ye kadar harflerin ASCII karakterleri de, sayısal değerleri artı onaltılık 37'ye eşittir. Eğer klavyeden 'E4' girilirse, aşağıdaki işlemler yerine getirilmelidir.

E'nin 7-bit ASCII karşılığı (yani  $45_{10}$ ) 4-bit ikili biçime (yani  $14_{10}=1110_2$ ) dönüştür. 4'ün 7-bitlik karşılığı (yani  $34_{10}$ ) 4-bit ikili biçime (yani  $4_{10}=0100_2$ ) dönüştür.

Bu iki 4-bit değeri 8-bitlik tek bir sözcükte paketlemek için, en ağırlıklı karakter 11100000'i elde etmek üzere 4 basamak sola kaydırılır ve buna sonuç olarak 11100100'ı vermek üzere en küçük değerlikli karakter eklenir.

Bir ASCII karakterin onaltılık biçime dönüşmesi aşağıdaki gibi ifade edilebilir.

```

success:=true
char:=char-$30
IF char < 0 THEN success:=false
ELSE IF char < 10 THEN hexvalue:=char
ELSE IF (char < 17) OR (char > 22) THEN success:=false
ELSE hexvalue:=char-7

```

Program, 6502 assembly dilinde bir sonraki sayfada görüldüğü gibi kodlanabilir.

|       |     |       |                                             |
|-------|-----|-------|---------------------------------------------|
|       | JSR | INHEX | ; İlk onaltılı karakteri akümülatörün EDD ; |
|       |     |       | ; 4. bitine al                              |
|       | BCS | END   | ; Eğer elde kurulmuşsa onaltılı olmayan ;   |
|       |     |       | bir karakter ve hata                        |
|       | ASL | A     | ; EA 4 biti doldurmak için 4 basamak sola   |
|       |     |       | ; kaydır                                    |
|       | ASL | A     |                                             |
|       | ASL | A     |                                             |
|       | ASL | A     |                                             |
|       | STA | TEMP  | ; Akümülatörü TEMP'e sakla                  |
|       | JSR | INHEX | ; İkinci onaltılı karakteri akümülatörün    |
|       |     |       | ; EKD 4-bitine al                           |
|       | BCS | END   | ; Eğer elde kurulmuşsa onaltılı olmayan     |
|       |     |       | ; bir karakter ve hata                      |
|       | CLC |       | ; Toplamadan önce eldeyi sıfırla            |
|       | ADC | TEMP  | ; TEMP'te saklanan en ağırlıklı 4 biti topl |
| END   | RTS |       | ; Geriye dön                                |
| INHEX | JSR | INALL | ; Bir ASCII karakter al                     |
|       | SEC |       | ; Çıkarmadan önce eldeyi kur                |
|       | SBC | #\$30 | ; \$30 çıkar                                |
|       | BMI | FAIL  | ; Negatifse geçerli olmayan bir onaltılı    |
|       | CMP | #\$0A | ; 0-9 aralığında mı kontrol et              |
|       | BMI | OK    | ; 10'dan küçükse başarı ile dön             |
|       | CMP | #\$11 | ; Aralık 'A'dan küçük mü kontrol et.        |
|       | BMI | FAIL  | ; Negatifse geçerli olmayan karakter        |
|       | CMP | #\$17 | ; Aralık 'F'den büyük mü kontrol et         |
|       | BPL | FAIL  | ; Pozitifse geçerli olmayan karakter        |
|       | SEC |       | ; Çıkarmadan önce eldeyi kur                |
|       | SBC | #\$07 | ; Onaltılık biçime dönüştürmek için 7 çıkar |
| OK    | CLC |       | ; Eldeyi sıfırla (=başarı)                  |
|       | RTS |       | ; Geriye dön                                |
| FAIL  | SEC |       | ; Eldeyi kur (=başarısızlık)                |
|       | RTS |       | ; Geriye dön                                |

## Soru 9

Palindrom, normal ve tersten okunduğunda aynı olan kelimedir. Örneğin HAN-NAH bir palindromdur. Bir metin dizisi bellekte \$0300 konumundan başlayarak saklanmıştır ve ASCII kodu \$0D olan satırbaşı karakteri ile sonlandırılmıştır. Dizinin palindrom olup olmadığını belirleyecek bir assembly-dili programı yazın. Eğer dizi palindrom ise programın sonunda PSW'nin elde biti kurulmalıdır.

## Çözüm 9

Dikkat edilecek iki durum vardır; biri dizide çift sayıda karakter olma, diğeri de tek sayıda karakter olma durumudur.

Çift: ABCDDCBA      Tek: ABCDCBA  
 ↑                    ↑                    ↑                    ↑  
 I                    J                    I                    J

Eğer bir işaretçi dizinin her iki ucundan da hareket ettirilir ve işaret edilmekte olan soldaki ve sağdaki harfler karşılaştırılırsa, ortaya ulaşana kadar tüm çiftler birbirini tutuyorsa dizi polindromdur.

Sol işaretçi I ve sağ işaretçi J ise, orta noktaya; I=J iken (tek sayıda harf) veya I+1=J iken (çift sayıda harf) ulaşılır.

Program, 6502 assembly dilinde aşağıdaki gibi kodlanabilir.

|       |     |            |                                          |
|-------|-----|------------|------------------------------------------|
|       | NAM | PALINDROME |                                          |
|       | ORG | \$0000     |                                          |
| SAVEY | RMB | 1          | ; Y kaydedicisi için geçici              |
|       |     |            | ; saklama konumu                         |
|       | ORG | \$0080     | ; programın başı                         |
|       | LDX | #00        | ; x kaydedicisini sıfırla (sağ işaretçi) |
| NEXT  | LDA | \$0300,X   | ; bir karakter al                        |
|       | INX |            | ; işaretçiyi artır                       |
|       | CMP | #\$0D      | ; satır-başı karakteri mi?               |
|       | BNE | NEXT       | ; değilse devam et                       |
|       | DEX |            | ; X şimdi dizinin sağındaki              |
|       | DEX |            | ; karakteri işaret ediyor                |
|       | LDY | #00        | ; Y kaydedicisini sıfırla (sol işaretçi) |
| COMP  | LDA | \$0300,Y   | ; soldan bir karakter al                 |
|       | CMP | \$0300,X   | ; sağdaki karakter ile karşılaştır       |
|       | BNE | FAIL       | ; aynı değilse polindrom değil           |
|       | STY | SAVEY      | ; Y'yi bellekte sakla                    |
|       | CPX | SAVEY      | ; X ile Y'yi karşılaştır (tek uzunluk)   |
|       | BEQ | SUCCESS    | ; aynıysa ortaya ulaşıldı                |
|       | INC | SAVEY      | ; bellekteki Y'yi artır                  |
|       | CPX | SAVEY      | ; X ile Y'yi karşılaştır (çift uzunluk)  |
|       | BEQ | SUCCESS    | ; aynıysa ortaya ulaşıldı                |

|         |     |      |                       |
|---------|-----|------|-----------------------|
|         | INY |      | ; sol işaretçiyi taşı |
|         | DEX |      | ; sağ işaretçiyi taşı |
|         | JMP | COMP | ; tekrarla            |
| FAIL    | CLC |      | ; eldeyi sıfırla      |
|         | RTS |      | ; dön-palindrom değil |
| SUCCESS | SEC |      | ; eldeyi kur          |
|         | RTS |      | ; dön-palindromdur.   |

## EK A2: 6502 KOMUT TAKIMI

Bu ek bölümün, kitaba eklenmiş olmasının iki nedeni vardır. Okurun, başka kaynaklara ihtiyaç duymadan bazı assembly-dili problemleri üzerinde çalışabilmesini sağlamak ve tipik bir mikroişlemci komut takımını göstermek. Temelde herhangi bir problemi, herhangi bir mikroişlemci ile çözmek mümkün olsa da; 8-bitlik mikro işlemcilerin komut takımı, bir mikro işlemciden diğerine değişiklik gösterir. Pratikte bazı komut takımları nispeten kabadır ve güzel bir çözümü mümkün kılan zariflikten yoksundur. Komut takımlarının karşılaştırılması, çoğu zaman değerlendirme testi ile yapılır. Bir test programı, sözcüğü, üç farklı mikro işlemcinin assembly dilinde kodlanır ve sonra da programlar yürütülür. Her programı yürütmek için geçen süreyi ölçerek mikro işlemcinin hak ettiği yararlılık düzeyi belirlenir. Bir başka değer biçimi her programın büyüklüğünü (yani sözcük sayısını) inceleyerek elde edilir. İdeal olarak, en iyi mikro işlemci, programı en hızlı yürütmelidir ve en az sayıda baytlık yer kaplamalıdır.

Bununla birlikte, değerlendirme testlerine güvenilmez. Oldukça yoğun bir programı yazmak için programcının yeteneği, testin önemli bir parçası olduğu için; bir problemi üç farklı makina koduna eşit verimlilikte aktarmak zordur.

### 6502'nin komut takımı

Klasik olarak mikro işlemcilerin komut takımları, çoğu kez alfabetik sıradaki anımsatıcılar biçiminde verilir. Ben bunları yaptıkları işlem türüne göre mantıksal olarak gruplamaya karar verdim. Seçilen gruplar şunlardır: Aritmetik, mantıksal, veri taşıma, dallanma ve denetim. İlk dört grubun ne oldukları adlarından anlaşıl-maktadır, denetim grubu ise, mikro işlemcinin işlem sırasını etkileyen komutları içerir (dallanma komutları hariç). Ayrıca bu gruba çeşitli komutları da ekledim. Tablo A2.1'de gruplarına göre tüm komut kısaltmaları alfabetik sırada listelenmiştir.

Tablo A2.1 Alfabetik sırada düzenlenmiş 6502 komut kısaltmaları

| Kom. Kısalt. | Adı                                       | Grubu |
|--------------|-------------------------------------------|-------|
| ADC          | Elde ile toplama                          | Arit. |
| AND          | Mantıksal VE'leme                         | Mant. |
| ASL          | Aritmetik sola kaydırma                   | Arit. |
| BCC          | Elde sıfırsa dallan                       | Dall. |
| BCS          | Elde kuruluyrsa dallan                    | Dall. |
| BEQ          | Sıfıra eşitse dallan                      | Dall. |
| BIT          | Bellek bitlerini akümülatörle karşılaştır | Mant. |
| BMI          | Eksi ise dallan                           | Dall. |

|     |                                          |
|-----|------------------------------------------|
| BNE | Sıfıra eşit değilse dallan               |
| BPL | Artı ise dallan                          |
| BRK | Durdur                                   |
| BVC | Taşma sıfırda dallan                     |
| BVS | Taşma kuruluşu dallan                    |
| CLC | Elde bayrağı sıfırla                     |
| CLD | Ondalık bayrağı sıfırla                  |
| CLI | Kesme maskesini sıfırla                  |
| CLV | Taşma bayrağı sıfırla                    |
| CMP | Akümülatörle karşılaştır                 |
| CPX | X kaydedicisi ile karşılaştır            |
| CPY | Y kaydedicisi ile karşılaştır            |
| DEC | Belleği azalt                            |
| DEX | X kaydedicisini azalt                    |
| DEY | Y kaydedicisini azalt                    |
| EOR | Özel VEYA                                |
| INC | Belleği artır                            |
| INX | X kaydedicisini artır                    |
| INY | Y kaydedicisini artır                    |
| JMP | Adrese atla                              |
| JSR | Altyordama atla                          |
| LDA | Akümülatörü yükle                        |
| LDX | X kaydedicisini yükle                    |
| LDY | Y kaydedicisini yükle                    |
| LSR | Mantıksal sağa kaydırma                  |
| NOP | Hiç işlem yapma                          |
| ORA | VEYA fonksiyonu                          |
| PHA | Akümülatörü yığma it                     |
| PHP | İşlemci durumunu yığma it                |
| PLA | Akümülatörü yığından çıkar               |
| PLP | İşlemci durumunu yığından çıkar          |
| ROL | Elde üzerinden bir bit sola döndür       |
| ROR | Elde üzerinden bir bit sağa döndür       |
| RTI | Kesmeden dönüş                           |
| RTS | Altyordamdan dönüş                       |
| SBC | Elde ile çıkar                           |
| SEC | Eldeyi kur                               |
| SED | Ondalık moda geç                         |
| SEI | PSW'nin kesme maske bitini kur           |
| STA | Akümülatörü belleğe sakla                |
| STX | X kaydedicisini belleğe sakla            |
| STY | Y kaydedicisini belleğe sakla            |
| TAX | Akümülatörü X kaydedicisine aktar        |
| TAY | Akümülatörü Y kaydedicisine aktar        |
| TSX | Yığın işaretçisini X kaydedicisine aktar |
| TXA | X kaydedicisini akümülatöre aktar        |
| TXS | X kaydedicisini yığın işaretçiye aktar   |
| TYA | Y kaydedicisini akümülatöre aktar        |

Arit. - Aritmetik  
Mant. - Mantıksal  
Taşı. - Veri taşınma (aktarma)  
Dall. - Dallanma  
Dent. - Denetim

Dall.  
Dall.  
Dent.  
Dall.  
Dall.  
Arit.  
Arit.  
Dent.  
Arit.  
Arit.  
Arit.  
Arit.  
Arit.  
Arit.  
Mant.  
Arit.  
Arit.  
Arit.  
Dall.  
Dall.  
Taşı.  
Taşı.  
Taşı.  
Mant.  
Dent.  
Mant.  
Dent.  
Dent.  
Arit.  
Arit.  
Dent.  
Taşı.  
Taşı.  
Taşı.  
Taşı.  
Taşı.  
Taşı.  
Taşı.

Tablo A2.1'deki komutla yapılan işlemler, daha sonra mümkün olan her yerde kaydedici aktarım dili (RTL) şeklinde açıklanacaktır. Tablo A2.2'de, kullanılan kısaltmaları listelenmiştir.

Tablo A2.2 İşlem kodu açıklamalarında kullanılan kısaltmalar

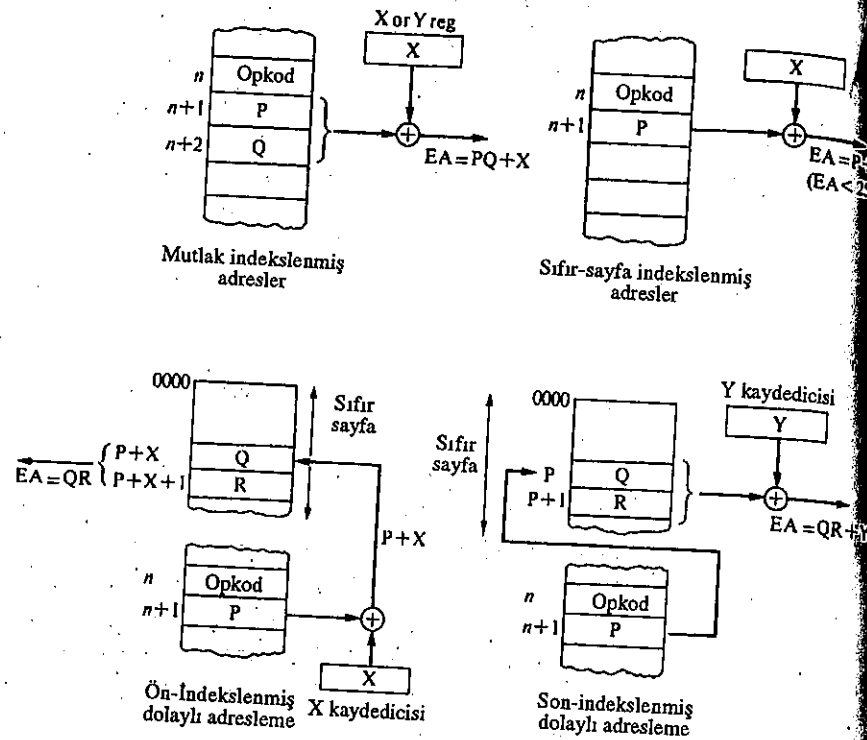
| Acc | Akümülatör              | PSW bitleri:            |
|-----|-------------------------|-------------------------|
| M   | Genel bir bellek konumu | N Negatif bit (EBDB)    |
| SP  | Yığın işaretçisi        | V Taşma biti            |
| X   | X indeks kaydedicisi    | B Durdurma noktası biti |
| Y   | Y indeks kaydedicisi    | D Ondalık mod biti      |
| PC  | Program sayıcısı        | I Kesme maske biti      |
| L   | Harf sel bir değer      | Z Sıfır biti            |
| PSW | İşlemci Durum Sözcüğü   | C Elde biti             |

### 6502'nin adresleme modları

6502'nin yedi temel adresleme modu vardır: örtük, akümülatör, dolaysız (birebir), sıfır-sayfa, mutlak, indekslenmiş ve bağıl adresleme. Bundan sonraki tablolarda dolaysız veya birebir adresleme L ile, sıfır-sayfa adresleme Z ile, mutlak adresleme Mutl. ile, indekslenmiş adresleme I ile gösterilmiştir. Sözdde örtük adresleme modu uygun sütunda 'örtük' sözcüğü ile gösterilmiştir ve doğrudan bir bellek referansı gerekmediğini ifade eder.

Akümülatör adresleme Acc ile gösterilmiştir ve örtük adreslemenin özel bir durumu mudur. Akümülatör adresleme, bir bellek konumundaki işlemle akümülatörün içerdiği işlemi ayırt etmek için kullanılır. İndekslenmiş adresleme daha alt-gruplara da bölünebilir. Bunlar X ve Y kaydedicilerini kullanan ve sırasıyla IX ve IY ile gösterilen mutlak indekslemeyi içerirler. Bu modlarda X veya Y kaydedicisinin içerdiği değer 16-bitlik etkin adresi üretmek üzere işlem kodunu takip eden 16-bit ofsete eklenir. Sıfır-sayfa indekslenmiş adresleme IXZ ile gösterilir ve X indeks kaydedicisinin içerdiği değer ile opkodu izleyen 8-bit ofsetin toplamından etkin bir adres üreterek sıfırıncı sayfadaki bir adrese erişmek için kullanılır.

İndekslenmiş dolaylı adresleme (IX) ile gösterilir, aynı zamanda ön-indekslenmiş dolaylı adresleme de denir. Bir işlenenin etkin adresi indeks kaydedicisinin içerdiği değer in opkodu izleyen 8-bitlik değere eklenmesiyle verilir. Daha sonra bu işlemin gerçek adresini içeren sıfırıncı sayfadaki bir bellek konumunu sorgulamak için kullanılır. Böylece etkin adres  $EA = [X] + \text{OFSET}$  olarak tanımlanır. Etkin adres tüm 64K bayt bellek alanına yayılan 16-bit bir değerdir ve iki ardışık baytta saklanır.  $[X] + \text{OFSET}$  değeri, yukarıda hesaplandığı gibi, işlenenin alt-sıralı baytının adresini ve  $[X] + \text{OFSET} + 1$  bellek konumu da, işlenenin üst-sıralı baytının



Şekil A2.1 İndekslenmiş adresleme ve 6502

adresini verir. Şekil A2.1'e bakınız.

Dolaylı indekslenmiş adresleme, (IY) ile gösterilir, aynı zamanda, buna son-İndekslenmiş dolaylı adresleme de denir. Bu durumda bir işlenenin etkin adresi sıfırıncı sayfada 8-bitlik bir değere bakmak için opkodu takip eden ofseti kullanarak ve bu değeri Y kaydedicisinin içerdiği değere ekleyerek verilir. Burada etkin adres  $EA = [OFSET + [Y]]$  olarak tanımlanır. İndekslenmiş dolaylı adreslemede olduğu gibi, etkin adres 16-bitlik bir değerdir. OFFSET sıfırıncı sayfadaki bir bellek konumunu gösteren 8-bitlik bir değerdir. OFFSET ve OFFSET+1 tarafından gösterilen iki ardışık konum, işlenenin 16-bitlik etkin adresini oluşturmak için kullanılır. Şekil A2.1'e bakınız. İndekslenmiş dolaylı adreslemede, sadece X kaydedicisinin kullanılabilirken; dolaylı indekslenmiş adreslemede, sadece Y kaydedicisinin kullanılabilirliğine dikkat ediniz.

Sunulan her bir komut gurubu için aşağıdaki format benimsenmiştir:

Komut kısaltması RTL olarak işlem

Sözcük olarak işlem

İşlemin etkilediği işlemci durum sözcüğü bayrak bitleri

Adresleme modu

Bir komut kısaltması için işlem kodlarının değerleri (herbir adresleme modu için) bu bölümün sonunda Tablo A2.3'te verilmiştir.

### Aritmetik Grup

ADC

$[A] \leftarrow [A] + [M] + [C]$

M bellek adresi, elde biti ile birlikte, akümülatöre ekle, sonucu akümülatörde sakla

Etkilenen bayraklar: N, V, Z, C

Adresleme modları: L, Z, Mutl., IX, IY, IXZ, (IX), (IY)

ASL

$C \leftarrow [7] \leftarrow [6] \leftarrow [5] \leftarrow [4] \leftarrow [3] \leftarrow [2] \leftarrow [1] \leftarrow [0] \leftarrow 0$

Akümülatörün (veya bellek konumunun) içeriğini bir basamak sola kaydır. EDB elde bitine kaydırılır ve EDDB'e sıfır girilir.

Etkilenen bayraklar: N, Z, C

Adresleme modları: Acc, Z, Mutl., IX, IXZ

CLC

$[C] \leftarrow 0.$

PSW'deki elde bitini sıfırla

Etkilenen bayraklar: C

Adresleme modları : örtük

CLD

$[D] \leftarrow 0.$

PSW'nin onlu mod bayrak bitini sıfırla

Etkilenen bayraklar: D

Adresleme modları : örtük

CLV

$[V] \leftarrow 0.$

PSW'nin taşıma bitini sıfırla

Etkilenen bayraklar: V

Adresleme modları: örtük

CMP

 $[A] \leftarrow [M]$ 

M bellek adresinin içeriğini akümülatörün içeriğinden çıkart. Sonuç atılır ve akümülatörün içeriği etkilenmez. Bu komutun amacı, ilgili PSW durum bitlerini kurmak/sıfırlamaktır.

Etkilenen bayraklar: N,Z,C

Adresleme modları: L,Z, Mutl., IX, IY, IXZ, (IX), (IY)

CPX

 $[X] \leftarrow [M]$ 

M bellek adresinin içeriğini X kaydedicisinin içeriğinden çıkart. Sonuç saklanmaz.

Etkilenen bayraklar: N,Z,C

Adresleme modları: L,Z,Mutl.

CPY

 $[Y] \leftarrow [M]$ 

M bellek adresinin içeriğini Y kaydedicisinin içeriğinden çıkart. Sonuç saklanmaz.

Etkilenen bayraklar: N,Z,C

Adresleme modları: L,Z,Mutl.

DEC

 $[M] \leftarrow [M] - 1.$ 

M bellek konumunun içeriğini azalt.

Etkilenen bayraklar: N, Z

Adresleme modları: Z, Mutl., IX, IXZ

DEX

 $[X] \leftarrow [X] - 1.$ 

X kaydedicisinin içeriğini azalt.

Etkilenen bayraklar: N,Z

Adresleme modları: örtük

DEY

 $[Y] \leftarrow [Y] - 1.$ 

Y kaydedicisinin içeriğini azalt

Etkilenen bayraklar: N,Z

Adresleme modları: örtük

INC

 $[M] \leftarrow [M] + 1.$ 

M bellek konumunun içeriğini artır

Etkilenen bayraklar: N,Z

Adresleme modları: Z, Mutl., IX, IXZ

INX

 $[X] \leftarrow [X] + 1.$ 

X kaydedicisinin içeriğini artır

Etkilenen bayraklar: N,Z

Adresleme modları: örtük

INY

 $[Y] \leftarrow [Y] + 1.$ 

Y kaydedicisinin içeriğini artır

Etkilenen bayraklar: N,Z

Adresleme modları: örtük

SBC

 $[A] \leftarrow [A] - [M] - [\bar{C}]$ 

M bellek konumunun içeriğini akümülatörün içeriğinden çıkart.

Elde bitinin tümleyeni de çıkartılır. Sonuç akümülatörde bırakılır.

Etkilenen bayraklar: N, V, Z, C

Adresleme modları: L, Z, Mutl., IX, IY, IXZ, (IX), (IY)

SEC

 $[C] \leftarrow 1.$ 

PSW'nin elde bayrak bitini kur.

Etkilenen bayraklar: C

Adresleme modları: örtük

SED

 $[D] \leftarrow 1.$ 

PSW'nin ondalık mod bayrak bitini kur.

Etkilenen bayraklar: D

Adresleme modları: örtük

## Mantıksal Grup

AND

 $[A] \leftarrow [A] \cdot [M].$ 

Akümlatörün içeriği ile M bellek konumunun içeriği arasında mantıksal VE işlemi yap. Sonucu akümülatörde sakla.

Etkilenen bayraklar: N, Z

Adresleme modları: L, Z, Mutl., IX, IY, IXZ, (IX), (IY)

BIT

 $[A] \cdot [M].$ 

Akümlatörün içeriği ile M bellek adresinin içeriği arasında mantıksal VE işlemi yap. Sonucu dikkate alma. Bu komut CMP'nin mantıksal eşitidir. [A] ve [M] aynıysa PSW'nin Z bitinin sıfırlanacağına dikkat edin.

[M]'nin bit 7'si PSW'nin N bayrak bitine, ve [M]'nin 6. biti PSW'nin V bayrak bitine transfer edilir.

Etkilenen bayraklar: N, V, Z

Adresleme modları: Z, Mutl.

EOR  $[A] \leftarrow [A] \oplus [M]$

Akümülatörün içeriği, M bellek adresinin içeriği ile özel VEYA'lanır.

Etkilenen bayraklar: N, Z

Adresleme modları: L, Z, Mutl., IX, IY, IXZ, (IX), (IY)

LSR  $0 \rightarrow \begin{array}{|c|c|c|c|c|c|c|c|} \hline 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \hline \end{array} \rightarrow C$

M bellek konumunun veya akümülatörün içeriği bir basamak sağa kaydırılır. EBDB'e bir 0, EKDB'de elde bitine girer.

Etkilenen bayraklar: N, Z, C

Adresleme modları: ACC, Z, Mutl., IX, IXZ

ORA  $[A] \leftarrow [A] + [M]$

Akümülatörün içeriği, M bellek adresinin içeriği ile VEYA'lanır.

Etkilenen bayraklar: N, Z

Adresleme modları: L, Z, Mutl., IX, IY, IXZ, (IX), (IY)

ROL  $\begin{array}{|c|c|c|c|c|c|c|c|} \hline 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \hline \end{array} \leftarrow C \leftarrow$

Akümülatörün veya M bellek adresinin içeriği bir basamak sola döndürülür (kaydırılır). Elde biti EKDB konumuna, EBDB'de elde biti konumuna kaydırılır. Bunun dokuz-bitlik bir işlem olduğuna ve kaybolan bitin bulunmadığına dikkat edin.

Etkilenen bayraklar: N, Z, C

Adresleme modları: ACC, Z, Mutl., IX, IXZ

ROR  $\begin{array}{|c|} \hline C \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|c|c|c|c|c|} \hline 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \hline \end{array}$

Akümülatörün veya M bellek adresinin içeriği bir basamak sağa döndürülür (kaydırılır). Elde biti EBDB konumuna, EKDB'de elde biti konumuna kaydırılır. Bunun dokuz-bitlik bir işlem olduğuna ve kaybolan bitin bulunmadığına dikkat edin.

Etkilenen bayraklar: N, Z, C

Adresleme modları: ACC, Z, Mutl., IX, IXZ

### Veri taşıma grubu (aktarma)

LDA  $[A] \leftarrow [M]$

Akümülatöre M bellek adresinin içeriğini yükle.

Etkilenen bayraklar: N, Z

Adresleme modları: L, Z, Mutl., IX, IY, IXZ, (IX), (IY)

LDX  $[X] \leftarrow [M]$

X kaydedicisine M bellek adresinin içeriğini yükle.

Etkilenen bayraklar: N, Z

Adresleme modları: L, Z, Mutl., IY, IYZ

LDY  $[Y] \leftarrow [M]$

Y kaydedicisine M bellek adresinin içeriğini yükle.

Etkilenen bayraklar: N, Z

Adresleme modları: L, Z, Mutl., IX, IXZ

PHA  $[[SP]] \leftarrow [A], [SP] \leftarrow [SP] - 1$

Akümülatörün içeriğini yığına it.

Etkilenen bayraklar: yok

Adresleme modları: örtük

PHP  $[[SP]] \leftarrow [PSW], [SP] \leftarrow [SP] - 1$

İşlemci durum sözcüğünü yığına it.

Etkilenen bayraklar: yok

Adresleme modları: örtük

PLA  $[SP] \leftarrow [SP] + 1, [A] \leftarrow [[SP]]$

Yığının en üstündeki sözcüğü çıkar ve akümülatöre yerleştir.

Etkilenen bayraklar: yok

Adresleme modları: örtük

PLP  $[SP] \leftarrow [SP] + 1, [PSW] \leftarrow [[SP]]$

Yığının en üstündeki sözcüğü çıkar ve işlemci durum sözcüğüne yerleştir.

Etkilenen bayraklar: yok

Adresleme modları: örtük

STA  $[M] \leftarrow [A]$

Akümülatörün içeriği M bellek adresine saklanır.

|     |                                                                                                                                             |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------|
|     | Etkilenen bayraklar:yok<br>Adresleme modları:Z, Mutl., IX, IY, IXZ, (IX), (IY)                                                              |
| STX | $[M] \leftarrow [X]$<br>X kaydedicisinin içeriği, M bellek adresine saklanır.<br>Etkilenen bayraklar:yok<br>Adresleme modları:Z, Mutl., IYZ |
| STY | $[M] \leftarrow [Y]$<br>Y kaydedicisinin içeriği, M bellek adresine saklanır.<br>Etkilenen bayraklar:yok<br>Adresleme modları:Z, Mutl., IXZ |
| TAX | $[X] \leftarrow [A]$<br>Akümülatörün içeriği, X kaydedicisine kopyalanır.<br>Etkilenen bayraklar:yok<br>Adresleme modları:örtük             |
| TAY | $[Y] \leftarrow [A]$<br>Akümülatörün içeriği, Y kaydedicisine kopyalanır.<br>Etkilenen bayraklar:yok<br>Adresleme modları:örtük             |
| TSX | $[X] \leftarrow [SP]$<br>Yığın işaretçinin içeriği, X kaydedicisine kopyalanır.<br>Etkilenen bayraklar:yok<br>Adresleme modları:örtük       |
| TXA | $[A] \leftarrow [X]$<br>X kaydedicisinin içeriği, akümülatöre kopyalanır.<br>Etkilenen bayraklar:yok<br>Adresleme modları:örtük             |
| TXS | $[SP] \leftarrow [X]$<br>X kaydedicisinin içeriği, yığın işaretçisine kopyalanır.<br>Etkilenen bayraklar:yok<br>Adresleme modları:örtük     |

|     |                                                                                                                                 |
|-----|---------------------------------------------------------------------------------------------------------------------------------|
| TYA | $[A] \leftarrow [Y]$<br>Y kaydedicisinin içeriği, akümülatöre kopyalanır.<br>Etkilenen bayraklar:yok<br>Adresleme modları:örtük |
|-----|---------------------------------------------------------------------------------------------------------------------------------|

### Dallanma Grubu

Koşullu dallanma grubundaki Program Sayıcısının değeri (PS), içeriğinin o andaki getirme çevriminden sonra artırılan PS olduğuna dikkat edin.

|     |                                                                                                                                                                                                                           |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BCC | EĞER $[C] = 0$ İSE $[PC] \leftarrow [PC] + \text{OFSET}$ .<br>Eğer PSW'nin elde bayrak biti sıfırsa, yürürlükteki adres artı opkodu izleyen işaretli ofsete dallan.<br>Etkilenen bayraklar:yok<br>Adresleme modları:bağıl |
| BCS | EĞER $[C] = 1$ İSE $[PC] \leftarrow [PC] + \text{OFSET}$ .<br>Eğer PSW'nin elde bayrak biti kuruluysa, yürürlükteki adres artı işaretli ofsete dallan.<br>Etkilenen bayraklar:yok<br>Adresleme modları:bağıl              |
| BEQ | EĞER $[Z] = 1$ İSE $[PC] \leftarrow [PC] + \text{OFSET}$ .<br>Eğer PSW'nin Z bayrak biti kuruluysa, yürürlükteki adres artı işaretli ofsete dallan.<br>Etkilenen bayraklar:yok<br>Adresleme modları:bağıl                 |
| BMI | EĞER $[N] = 1$ İSE $[PC] \leftarrow [PC] + \text{OFSET}$ .<br>Eğer PSW'nin N bayrak-biti kuruluysa, yürürlükteki adres artı işaretli ofsete dallan.<br>Etkilenen bayraklar:yok<br>Adresleme modları:bağıl                 |
| BNE | EĞER $[Z] = 0$ İSE $[PC] \leftarrow [PC] + \text{OFSET}$ .<br>Eğer PSW'nin Z bayrak biti sıfırsa, yürürlükteki adres artı ofsete dallan.<br>Etkilenen bayraklar:yok<br>Adresleme modları:bağıl                            |

- BPL** EĞER [N] = 0 İSE [PC] ← [PC] + OFFSET.  
Eğer PSW'nin N bayrak biti sıfırsa, yürürlükteki adres artı ofsete dallan.  
Etkilenen bayraklar:yok  
Adresleme modları:bağlı
- BVC** EĞER [V] = 0 İSE [PC] ← [PC] + OFFSET.  
Eğer PSW'nin V bayrak biti sıfırsa, yürürlükteki adres artı ofsete dallan.  
Etkilenen bayraklar:yok  
Adresleme modları:bağlı
- BVS** EĞER [V] = 0 İSE [PC] ← [PC] + OFFSET.  
Eğer PSW'nin V bayrak biti kuruluysa, yürürlükteki adres artı ofsete dallan  
Etkilenen bayraklar:yok  
Adresleme modları:bağlı
- JMP** [PC] ← OFFSET.  
Opkodu izleyen mutlak ofset ile verilen 16-bit adrese atla.  
Etkilenen bayraklar:yok  
Adresleme modları:Mutl.
- JSR** [[SP]] ← [PC] + 2, [SP] ← [SP] - 2, [PC] ← OFFSET.  
Adresi Opkodu izleyen 16-bit mutlak ofset ile verilen altyordama atla.  
Etkilenen bayraklar:yok  
Adresleme modları:Mutl.

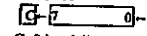
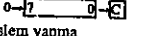
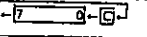
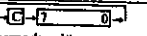
### Denetim Grubu ve Diğer Çeşitli gruplar

- BRK** [[SP]] ← [PC] + 2, [[SP] - 2] ← [PSW], [SP] ← [SP] - 3,  
[PC] ← [FFFE, FFFF].  
Program sayıcısının içeriği ile onu izleyen işlemci durum sözcüğünün içeriği yığına itilir. \$FFFE ve \$FFFF bellek konumlarının içeriği program sayacına yüklenir. Komut bir yazılım kesmesidir.  
Etkilenen bayraklar:B  
Adresleme modları:örtük
- CLI** [I] ← 0  
PSW'nin kesme bayrak biti sıfırlanır. Bu kesme isteklerini yetkilendirir.

Etkilenen bayraklar:I  
Adresleme modları:örtük

- NOP** Hiç işlem yapma  
Bu opkod iki saat sayıklık bir gecikme yaratmaktan başka programın yürütülmesinde bir etkisi yoktur.  
Etkilenen bayraklar:yok  
Adresleme modları:örtük
- RTI** [SP] ← [SP] + 1, [PSW] ← [[SP]], [SP] ← [SP] + 2, [PC] ← [[SP]].  
PSR ve program sayıcısı yığından çıkarılır.  
Etkilenen bayraklar:tümü  
Adresleme modları:örtük
- RTS** [SP] ← [SP] + 2, [PC] ← [[SP]], [PC] ← [PC] + 1.  
Program sayıcısı yığının en üstünden dışarı alınır.  
Etkilenen bayraklar:yok  
Adresleme modları:örtük
- SEI** [I] ← 1.  
PSW'nin kesme maske bitini kur. Bunun kesmeleri yetkisizleme etkisi vardır.  
Etkilenen bayraklar:I  
Adresleme modları:örtük

Tablo A2.3: 6502'nin işlem kodları.

| Komutlar  | Dolaysız                                                                               | Mutlak         | Sıfır sayfa | Aktüm. | (İad, X) |
|-----------|----------------------------------------------------------------------------------------|----------------|-------------|--------|----------|
| Kom. Kıs. | İşlem                                                                                  | OP n *         | OP n *      | OP n * | OP n *   |
| ADC       | A←M+C→A                                                                                | (4) (1) 69 2 2 | 6D 4 3      | 65 3 2 | 61 6 2   |
| AND       | A←M→A                                                                                  | (1) 29 2 2     | 2D 4 3      | 25 3 2 | 21 6 2   |
| ASL       |       |                | 0E 6 3      | 06 5 2 | 0A 2 1   |
| BCC       | C=0 ise dallan                                                                         | (2)            |             |        |          |
| BVS       | C=1 ise dallan                                                                         | (2)            |             |        |          |
| BEQ       | Z=1 ise dallan                                                                         | (2)            |             |        |          |
| BIT       | A←M                                                                                    |                | 2C 4 3      | 24 3 2 |          |
| BMI       | N=1 ise dallan                                                                         | (2)            |             |        |          |
| BNE       | Z=0 ise dallan                                                                         | (2)            |             |        |          |
| BPL       | N=0 ise dallan                                                                         | (2)            |             |        |          |
| BRK       | Durdur                                                                                 |                |             |        | 00 7 1   |
| BVC       | V=0 ise dallan                                                                         | (2)            |             |        |          |
| BVS       | V=1 ise dallan                                                                         | (2)            |             |        |          |
| CLC       | 0←C                                                                                    |                |             |        | 18 2 1   |
| CLD       | 0←D                                                                                    |                |             |        | D8 2 1   |
| CLI       | 0←I                                                                                    |                |             |        | 58 2 1   |
| CLV       | 0←V                                                                                    |                |             |        | B8 2 1   |
| CMP       | A←M                                                                                    | C9 2 2         | CD 4 3      | C5 3 2 |          |
| CPX       | X←M                                                                                    | E0 2 2         | EC 4 3      | E4 3 2 | CI 6 2   |
| CPY       | Y←M                                                                                    | C0 2 2         | CC 4 3      | C4 3 2 |          |
| DEC       | M←M-1                                                                                  |                | CE 6 3      | C6 5 2 |          |
| DEX       | X←X-1                                                                                  |                |             |        | CA 2 1   |
| DEY       | Y←Y-1                                                                                  |                |             |        | 88 2 1   |
| EOR       | A←M⊕A                                                                                  | (1) 49 2 2     | 4D 4 3      | 45 3 2 | 41 6 2   |
| INC       | M←M+1                                                                                  |                | EE 6 3      | E6 5 2 |          |
| INX       | X←X+1                                                                                  |                |             |        | EB 2 1   |
| INY       | Y←Y+1                                                                                  |                |             |        | CB 2 1   |
| JMP       | Yeni konuma atla                                                                       |                | 4C 3 3      |        |          |
| JSR       | Altıyordama atla                                                                       |                | 20 6 3      |        |          |
| LDA       | M←A                                                                                    | (1) A9 2 2     | AD 4 3      | A5 3 2 | AI 6 2   |
| LDX       | M←X                                                                                    | (1) A2 2 2     | AE 4 3      | A6 3 2 |          |
| LDY       | M←Y                                                                                    | (1) A0 2 2     | AC 4 3      | A4 3 2 |          |
| LSR       | 0←  |                | 4E 6 3      | 46 5 2 | 4A 2 1   |
| NOP       | İşlem yapma                                                                            |                |             |        | EA 2 1   |
| ORA       | A←M∨A                                                                                  | 09 2 2         | 0D 4 3      | 05 3 2 | 01 6 2   |
| PHA       | A←Ms                                                                                   |                |             |        | 48 3 1   |
| PHP       | P←Ms                                                                                   |                |             |        | 08 3 1   |
| PLA       | S←I→S                                                                                  |                |             |        | 68 4 1   |
| PLP       | S←I→S                                                                                  |                |             |        | 28 4 1   |
| ROL       |     |                | 2E 6 3      | 26 5 2 | 2A 2 1   |
| ROR       |     |                | 6E 6 3      | 66 5 2 | 6A 2 1   |
| RTI       | Kesmeden dön                                                                           |                |             |        | 40 6 1   |
| RTS       | Altıyordamdan dön                                                                      |                |             |        | 60 6 1   |
| SBC       | A←M-C→A                                                                                | (1) E9 2 2     | ED 4 3      | E5 3 2 | E1 6 2   |
| SEC       | 1←C                                                                                    |                |             |        | 38 2 1   |
| SED       | 1←D                                                                                    |                |             |        | F8 2 1   |
| SEI       | 1←I                                                                                    |                |             |        | 78 2 1   |
| STA       | A←M                                                                                    |                | 8D 4 3      | 85 3 2 | 81 6 2   |
| STX       | X←M                                                                                    |                | 8E 4 3      | 86 3 2 |          |
| STY       | Y←M                                                                                    |                | 8C 4 3      | 84 3 2 |          |
| TAX       | A←X                                                                                    |                |             |        | AA 2 1   |
| TAY       | A←Y                                                                                    |                |             |        | AB 2 1   |
| TSX       | S←X                                                                                    |                |             |        | BA 2 1   |
| TXA       | X←A                                                                                    |                |             |        | 8A 2 1   |
| TXS       | X←S                                                                                    |                |             |        | 9A 2 1   |
| TYA       | Y←A                                                                                    |                |             |        | 98 2 1   |

(1) Sayfa sınırı geçilmişse, N'ye 1 ekle  
 (2) Aynı sayfaya dallanma olursa N'ye 1 ekle  
 Farklı sayfaya dallanma olursa N'ye 2 ekle

(3) Elde yok = Ödünç  
 (4) Eğer ondalık modda ise, Z bayrağı geçersizdir.  
 Akümülatör, 0 sonucu için test edilmelidir

| (İad), Y | S. Sayfa, X | Mut, X | Mut, Y | Bağıl  | Dolaylı | S. Sayfa, Y | İşlemci durum kodları                       | Kom. Kıs. |
|----------|-------------|--------|--------|--------|---------|-------------|---------------------------------------------|-----------|
| OP n *   | OP n *      | OP n * | OP n * | OP n * | OP n *  | OP n *      | 7 6 5 4 3 2 1 0<br>N V ● B D I Z C          |           |
| 71 5 2   | 75 4 2      | 7D 4 3 | 79 4 3 |        |         |             | N V . . . . . Z C                           | ADC       |
| 31 5 2   | 35 4 2      | 3D 4 3 | 39 4 3 |        |         |             | N . . . . . Z .                             | AND       |
|          | 16 6 2      | 1E 7 3 |        |        |         |             | N . . . . . Z C                             | ASL       |
|          |             |        |        | 90 2 2 |         |             | . . . . . . .                               | BCC       |
|          |             |        |        | 80 2 2 |         |             | . . . . . . .                               | BVS       |
|          |             |        |        |        | F0 2 2  |             | . . . . . . .                               | BEQ       |
|          |             |        |        |        |         |             | M <sub>7</sub> M <sub>6</sub> . . . . . Z . | BIT       |
|          |             |        |        | 30 2 2 |         |             | . . . . . . .                               | BMI       |
|          |             |        |        | D0 2 2 |         |             | . . . . . . .                               | BNE       |
|          |             |        |        | 10 2 2 |         |             | . . . . . . .                               | BPL       |
|          |             |        |        |        |         |             | . . . . . 1 . . . . .                       | BRK       |
|          |             |        |        | 50 2 2 |         |             | . . . . . . .                               | BVK       |
|          |             |        |        | 70 2 2 |         |             | . . . . . . .                               | BVS       |
|          |             |        |        |        |         |             | . . . . . . 0 . . . . .                     | CLC       |
|          |             |        |        |        |         |             | . . . . . 0 . . . . .                       | CLD       |
|          |             |        |        |        |         |             | . . . . . 0 . . . . .                       | CLI       |
|          |             |        |        |        |         |             | . 0 . . . . . . .                           | CLV       |
| D1 5 2   | D5 4 2      | DD 4 3 | D9 4 3 |        |         |             | N . . . . . Z C                             | CMP       |
|          |             |        |        |        |         |             | N . . . . . Z C                             | CPX       |
|          |             |        |        |        |         |             | N . . . . . Z C                             | CPY       |
|          | D6 6 2      | DE 7 3 |        |        |         |             | N . . . . . Z .                             | DEC       |
|          |             |        |        |        |         |             | N . . . . . Z .                             | DEX       |
|          |             |        |        |        |         |             | N . . . . . Z .                             | DEY       |
| 51 5 2   | 55 4 2      | 5D 4 3 | 59 4 3 |        |         |             | N . . . . . Z .                             | EOR       |
|          | F6 6 2      | FE 7 3 |        |        |         |             | N . . . . . Z .                             | INC       |
|          |             |        |        |        |         |             | N . . . . . Z .                             | INX       |
|          |             |        |        |        |         |             | N . . . . . Z .                             | INY       |
|          |             |        |        |        | 6C 5 3  |             | . . . . . . .                               | JMP       |
|          |             |        |        |        |         |             | . . . . . . .                               | JSR       |
| B1 5 2   | B5 4 2      | BD 4 3 | B9 4 3 |        |         |             | N . . . . . Z .                             | LDA       |
|          |             |        |        |        |         |             | . . . . . . .                               | LDX       |
|          | B4 4 2      | BC 4 3 |        |        |         |             | N . . . . . Z .                             | LDY       |
|          | 56 6 2      | 5E 7 3 |        |        |         |             | 0 . . . . . Z C                             | LSR       |
|          |             |        |        |        |         |             | . . . . . . .                               | NOP       |
| 11 5 2   | 15 4 2      | 1D 4 3 | 19 4 3 |        |         |             | N . . . . . Z .                             | ORA       |
|          |             |        |        |        |         |             | . . . . . . .                               | PHA       |
|          |             |        |        |        |         |             | . . . . . . .                               | PHP       |
|          |             |        |        |        |         |             | N . . . . . Z .                             | PLA       |
|          |             |        |        |        |         |             | (Geri alınır)                               | PLP       |
|          | 36 6 2      | 3E 7 3 |        |        |         |             | N . . . . . Z C                             | ROL       |
|          |             |        |        |        |         |             | . . . . . . .                               | ROR       |
|          |             |        |        |        |         |             | (Geri alınır)                               | RTI       |
|          |             |        |        |        |         |             | . . . . . . .                               | RTS       |
| F1 5 2   | F5 4 2      | FD 4 3 | F9 4 3 |        |         |             | N V . . . . . Z (3)                         | SBC       |
|          |             |        |        |        |         |             | . . . . . . .                               | SEC       |
|          |             |        |        |        |         |             | . . . . . . .                               | SED       |
|          |             |        |        |        |         |             | . . . . . . .                               | SEI       |
| 91 6 2   | 95 4 2      | 9D 5 3 | 99 5 3 |        |         |             | . . . . . . .                               | STA       |
|          |             |        |        |        |         |             | . . . . . . .                               | STX       |
|          | 94 4 2      |        |        |        |         |             | . . . . . . .                               | STY       |
|          |             |        |        |        |         |             | N . . . . . Z .                             | TAX       |
|          |             |        |        |        |         |             | . . . . . . .                               | TAY       |
|          |             |        |        |        |         |             | N . . . . . Z .                             | TSX       |
|          |             |        |        |        |         |             | N . . . . . Z .                             | TXA       |
|          |             |        |        |        |         |             | . . . . . . .                               | TXS       |
|          |             |        |        |        |         |             | N . . . . . Z .                             | TYA       |

X İndeks X  
 Y İndeks Y  
 A Akümülatör

M Etkin adres başına bellek  
 Ms Yiğün işaretçisi başına bellek

+ Toplama  
 - Çıkarma  
 ^ VE

V VEYA  
 v ÖZELVEYA  
 M<sub>7</sub> 7. Bellek biti  
 M<sub>6</sub> 6. Bellek biti

n Çevrim sayısı  
 \* Bayt sayısı

## KAYNAKLAR

Aşağıda, öğrencilerin daha ileri çalışmalarında yararlanabilmeleri amacıyla bazı kaynak kitaplar verilmiştir.

Lippiatt, A. G. (1979). *The architecture of small computer systems*. Prentice-Hall. Bu, ince (173 sayfa) bir kitap olup, küçük bilgisayarların yapısına bir giriş niteliğindedir. Buradaki "küçük" sözcüğüyle mikroişlemciler ve mini bilgisayarlar anlatılmak istenmektedir. Boole cebri, assembly dilinde programlama ve çevre birimlerine kitap içinde yer verilmemiştir.

Lee, G. (1982). *From hardware to software: an introduction to computers*. Macmillan Press. Bu, kalın (454 sayfa) bir kitap olup, elinizdeki bu kitapta yer alan konuların çoğunu içermektedir. Ancak G. Lee'nin kitabı ile elinizdeki bu kitap arasında başlıca iki fark vardır: Lee kitabını "Basit Sayısal Bir Bilgisayar" çerçevesinde geliştirmiş olup, kitabında öğretim amacıyla tasarlanmış varsayımsal bir bilgisayara ve yazılım (assembly dili, yüksek düzeyli dil ve derleme) üzerine oldukça ayrıntılı bilgilere yer vermiştir.

Gibson, J. R. (1979). *Electronic and logic circuits*. Edward Arnold. Bu kitap, giriş düzeyinde yazılmış ve tümüyle Boole cebrine ayrılmıştır. Yalnızca 114 sayfa olmasına karşın, oldukça iyi yazılmış ve hazırlanmış bir kitaptır.

Cripps, M. (1977). *An introduction to computer hardware*. Edward Arnold. Küçük hacimli bu kitapta, işlemcilerin yapısından çevre birimlerinin tasarımına kadar çok sayıda donanım konusu yer almaktadır. Bu kitap nispeten küçük hacimli (130 sayfa) olduğundan, temel ders kitabı olarak değil, yardımcı bir kaynak olarak düşünülmelidir.

Scanlan, L.J. (1981). *AIM 65 laboratory manual and study guide*. John Wiley. Bu kitap, AIM 65 mikrobilgisayarıyla birlikte kullanılmak üzere hazırlanmıştır. Kitap, çok sayıda öğretici alıştırmadan ve bu alıştırmaların her biri için gerekli temel bilgilerden oluşmaktadır. Bu kitap, kendi kendine çalışma yapmak için ideal niteliktedir ve aradaki farklılıkları göz önünde bulundurmak kaydıyla üzerinde assembly dilinde programlar çalıştırılabilen ve erişilebilir G/Ç yordamları içeren 6502 tabanlı mikrobilgisayarlarıyla birlikte de kullanılabilir.

Ullmann, J.R. (1982). *Microcomputer technology: an introduction*. Pitman Books. "Mikrobilgisayar Teknolojisi" başlığı, kitabın, silisyum yongalarının üretilme-

siyle ilgili olduğu izlenimi verse de, bu kitap hemen hemen tümüyle Boole cebri, ardışıl devreler, mikrobilgisayarlar ve assembly dilinde programlama konularını içermektedir. Örnek mikroişlemci olarak Z80 mikroişlemcisi seçilmiştir. Verilen örneklerin çoğu, hatta assembly dili programları da, Pascal dili temel alınarak sunulmuştur. Çünkü Profesör Ullmann, assembly dilini, "yapısı değiştirilmiş Pascal" olarak tanımlamaktadır. Bu kitap, gerçekten iyi hazırlanmış bir kitap olup, burada konular, benim kitabımdan daha yüksek düzeyde işlenmiştir. Çok sayıda çözümlü örnek içermektedir. Bilgisayar mimarisinin ilkelerini ilkelerini giriş kitaplarında ele alındığından daha kapsamlı olarak öğrenmek isteyenlere, bu kitabı okumalarını öneririm.

Tanenbaum, A. S. (1981). *Computer networks*. Prentice-Hall. Bu 517 sayfalık kitapta Tanenbaum, hem geniş alanlı iletişim ağlarını, hem de yerel iletişim ağlarını derinliğine incelemiştir. Bu kitap, ileri düzeyli bir kitaptır ve üçüncü sınıf öğrencileri ile lisansüstü öğrencilerine hitap etmektedir. İyi hazırlanmış bir kitap olup, bilgisayar iletişimi konusunda temel kaynaklardan biridir. Veri iletişimin donanım ve elektroniğe ilişkin bölümlerine yer verilmemiştir.

Wilkinson, B., and Horrocks, D. (1982). *Computer peripherals*. Hodder and Stoughton. Bu kitap, tümüyle giriş ve çıkış birimleri ile saklama birimlerine ve daha az ayrıntılı olmak kaydıyla bilgisayar iletişimine ayrılmıştır. Tüm bu konular giriş düzeyinde ele alınmıştır. Bu kitap, çevre birimlerinin yapısı, çalışması ve özellikleri konusunda daha ayrıntılı bilgi edinmek isteyen okurlar için özellikle önerilebilir.

Housley, T. (1981). *Data communication and tele processing systems*. Prentice-Hall. Bu kitap, veri iletişimine giriş niteliğindeki bilgileri kolayca okunup anlaşılabilir bir biçimde sunmaktadır. Genelde lisansüstü kitaplarında geçen bu konuyu, çok açık bir biçimde işlemektedir. Bu kitapta Housley, veri iletişim sistemlerine ilişkin protokoller konusuna mükemmel bir giriş yapmaktadır.

## İNDEKS

- ABX komutu 259
- ACK (alındı-bildirimi) kodları 119,413,414
- açık sistem bağlantısı (OSI) referans modeli 417
- açıklama, assembly dili programlarda 158
- açık-uçlu veri iletimi 271, 411
- aç/kapa düğmeleri 303
- ADC komutu 154,190,194,220,477,481,490-1
- RTL tanımı 190,481
- ADC'ler (analog-sayısal dönüştürücüler) 305-6
- ADD komutu 154,259
- denetim sinyalleri 202
- işlem sırası 201
- adım motoru 354, 355
- adres alanı 216,217
- veri iletimi 405
- adres haritalama/çeviri işlemi 449
- adres yolları, MİB 186,187
- adresleme modları 224-250
- akümülatör adresleme 479
- bağlı 206,253
- bağlı dallanma 253-4
- doğrudan 224,253
- genişletilmiş 252
- dolaysız 225-229,252
- indekslenmiş 229-34, 255-6, 479-80
- dolaylı 156-7,257,479
- doğal (örtük) 217,252,257
- kaydedici 257
- mutlak 224,236,237,479,480
- problem soruları 262-3
- programsayıcısına bağlı 253-5
- RTL tanımları 238
- 6502-mikroişlemci 479-80,490-1
- sıfır-sayfa 224-5,251,479
- AIM-65 bilgisayarı
- avantajları 153
- ilgili kitaplar 494-5
- problem soruları 262-3
- akı terslenmesi 341
- akıllı çevre birimleri 283
- akış diyagramları
- Booth algoritması 384
- geri getirmeli bölme 392
- geri getirmesiz bölme 395
- kayan-noktalı sayıların toplanması 170
- program 195
- sekiz-bitlik çarpma 379
- aktarma komutları 219-20, 258-9, 485-7
- aktüatörler, manyetik disk 348,349
- akümülatör 152,189
- akümülatör adresleme 479
- alan-etkili transistörler 312, 318
- alfa parçacıkları, yarıiletken duyarlılığı 321
- alfabeler 16-17
- sayısal bilgisayar 16
- algılama iletkenleri, ferrit-çekirdekli bellekler 329, 330
- alışveriş, para-hız 1
- ALOHA iletişim ağı 424-5
- altyordamlar 242-6
- dönüş mekanizması 242-3
- veri giriş/çıkışı 246-50
- ALU (aritmetik ve mantıksal birim) 189
- AM (genlik modülasyonu) 407
- amaç kodu 432
- ana bilgisayarlar, açıklaması 11-2
- ana saklama ortamı (MS) 188
- anahtarlama, kapı uygulaması 27
- analog giriş/çıkış birimleri 304-7
- analog/sayısal dönüştürücüler (ADC'ler) 305-6
- çözünürlüğü 306
- için örnekleme frekansı 307
- ana/uydu flip-floplar 81-2
- ana/uydu veri iletim sistemi 403-4, 414
- AND komutu 193,194,220,477,483,490-1
- ani erişim, tanımı 311
- ani erişimli saklama ortamları (IAS) 4, 6
- erişim süreleri 338

sadece-okunur bellek 9  
 sanal bellek mekanizması 451-2  
 ağ arabirim üniteleri (NIU'lar) 397  
 ağ katmanı, veri iletimi 417-8  
 ağaç diyagramı 135  
 ağlar 5, 397-8, 418-29  
 ilgili kitaplar 493-4  
 arabirimler  
 kullanıcı-yönelimli 302-3  
 veri aktarımı 269  
 aralık, kayan-noktalı sayılar 161, 165  
 aralıklı ALOHA 425  
 aralıklı halkalar 428  
 arama-tabloları, çarpma 102-3, 385-386  
 ardışık devreler 83-90  
 özel amaçlı paketler 90-96  
 problem soruları 115-6  
 tanımı 69  
 ardışık mantık elemanları, DIL paketleri 90-96  
 aritmetik operatörler, mantıksal eşdeğerleri 190  
 aritmetik ve mantık birimi (ALU) 192  
 aritmetik ve mantıksal komutlar 220, 259-60, 481-4  
 arıza oranı, tanımı 106  
 arızalar arası ortalama süre (MTBF) 104  
 ASCII (Bilgi Değişimi İçin Amerikan Standart Kodu) 118-9, 286, 402, 410  
 asenkron seri veri iletimi 399-400  
 asenkron sistemler 79  
 ASL komutu 220, 477, 481, 490-1  
 assembly dili 3, 152, 199,  
 programlardaki açıklamalar 158  
 programlardaki etiket alanları 197  
 problem soruları 265-8  
 atlama, terimin anlamı 192  
 atlama komutları 194, 220, 224, 244-5, 246, 487-8  
 band genişliği 341-2  
 bant: ayrıca bkz. manyetik...; kağıt...  
 bağıl adresleme 234-6, 253  
 basamak (digit), tanımı 120  
 BASIC programları 128  
 dallanma örneği 28

döngü yapısı 226-9  
 toplama-ve-çarpma 195  
 BASIC yorumlayıcıları, BCD kod kullananı  
 basınca-duyarlı anahtar 303  
 baskı kafaları  
 golf-topu 298-9  
 hareket mekanizması 295-6  
 silindir 298-9  
 Baud, tanımı 400  
 Baud-hızları, örnekleri 400, 409  
 bayrak, veri iletimi 403, 428  
 bayt denetim-yönelimli protokoller (BCP'ler)  
 410, 412-3  
 baytlar (sekiz- veya altı-bitlik sözcükler) 117  
 BCC hata-algılama kodu 410  
 BCC komutu 193, 194, 477, 487, 490-1  
 BCD (ikili kodlanmış ondalık) kodu 127-3  
 BCP'ler (bayt denetim-yönelimli protokoller)  
 410, 412-3  
 BCS komutu 156, 193, 194, 432, 487, 490-1  
 bellek adres kaydedicisi (MAR) 188  
 bellek alanının iki kopya halinde çoğaltılması  
 454-455  
 bellek görüntüleme işlevi 439  
 bellek haritalama tekniği 454-5  
 bellek haritalı G/Ç 274-5  
 bellek haritası 197-8  
 bayt-yapılı 312  
 bit-yapılı 312  
 sözcük-yapılı 334  
 yarım bayt-yapılı 312  
 bellek hiyerarşisi 337-67  
 bellek hücresi, tanımı 310  
 bellek kapasitesi 312  
 bellek karakteristikleri, ideal bellek 337  
 bellek saklama birimleri 4, 309  
 geri beslemeli elektriksel 309  
 depolanmış yükü elektriksel 309  
 manyetik 309  
 uzaysal 309  
 yapısal 309  
 ayrıca bkz. ferrit-çekirdekli...; manyetik  
 kabarcıklı...; levhalı-tel...; yarı iletken

...,ince-film bellek  
 bellek tampon kaydedicisi (MBR) 188  
 bellek tazeleme 311, 318, 320  
 bellek teknolojisi 310-1, 337-72  
 problem soruları 372-75  
 bellek yönetimi 5, 441, 443-58  
 bellek  
 gerektiren nedenler 308  
 ikili saklama biçimi 308  
 konum adresleri 15, 154-5, 197-8  
 tanımı 308  
 veri ve programın birlikteliği 15, 154-5  
 BEQ komutu 193, 194, 223, 431, 487, 490-1  
 bileşenlerin "pişirilmesi" 106  
 Bilgi çerçeveleri, HDLC protokolü 412-4  
 Bilgi Değişimi için Amerikan Standart Kod (ASCII) 118-119  
 denetim karakterleri 118, 119, 286, 402, 410-11  
 klavye kodlayıcı 286-7  
 bilgisayar, tanımı 5  
 bilgisayar bilimcileri, mikroişlemci açısından 9  
 bilgisayarlarda kullanılan aritmetik sistemler 3  
 bilgisayarların sayısal yapısı 16-17  
 bilimsel notasyon, kayan-noktalı sayılar 160  
 bire tümler aritmetik 151-2  
 birim seçici devreleri 62-3  
 birim uzaklık kodu 128-30  
 birleşik (combinational) devreler, tanımı 69  
 birlikte (concurrent) programlama 440  
 BISYNC protokolü 410, 411-2  
 BIT komutu 477, 483, 490-1  
 bit (Binary digIT), tanımı 17, 117  
 bit-doldurma 403, 427  
 bit-hızları, veri iletimi 399-400, 410  
 bit-senkronizasyonlu iletim 401  
 bit-yapılı bellekler 312  
 bit-yapılı bellekler 312  
 bit-yönelimli protokoller (BOP'lar) 403-4, 410, 412-8  
 blok-yönelimli iletim 402-5  
 BMI komutu 193, 477, 488, 490-1  
 BNE komutu 193, 194, 235, 236, 478, 487, 490-1

Boole cebri 33-36  
 ilişkin kitaplar 492  
 kullanım örnekleri 368  
 grafik gösterimi 44-50  
 problem çözümleri 112-3  
 sık karşılaşılan hatalar 36  
 temel kuralları 34  
 teoremler 35  
 Boole ifadeleri  
 basitleştirilmesi 36-9  
 Karnaugh diyagramları kullanılarak  
 basitleştirilmesi 47, 49-50  
 simetrisi 42  
 Booth algoritması 381-3  
 BOP'lar (bit-yönelimli protokoller) 403-4, 410, 412-8  
 BOT (bandın başlangıcı) işareti 362  
 bozucu okuma (DRO) aygıtları 330-1  
 bozucu-olmayan okuma (NDRO) modu,  
 ince-film bellek 336  
 boşluklu-bölme teyp tamponları 360  
 bölme, tanımı 387  
 bölme işlemi 4, 387-95  
 Boole cebrinde tanımlı olmayışı 71  
 çarpım yöntemi 391-94  
 gerigetirmeli bölme işlemi 390, 392  
 gerigetirmesiz bölme işlemi 390-1, 394, 395  
 bölü işareti (/) 437  
 BPL komutu 193, 478, 488, 490-1  
 BRA komutları 154, 234  
 BRK komutu 439, 478, 489, 490-1  
 büyük ölçekli entegre (LSI) devreler 58  
 büyük sayılar, 8-bit sözcük gösterimi 158  
 BVC komutu 193, 478, 488, 490-1  
 BVS komutu 156, 193, 478, 488, 490-1  
 B-H eğrileri 327, 330  
 Cambridge halkası 428-9  
 veri formatı 428  
 CAS (sütun adres strobu) 319  
 CCR (koşul kodu kaydedicisi) 154, 191  
 CCW (kanal denetim sözcüğü) 283  
 CLC komutu 154, 190, 194, 197, 220, 478, 481, 490-1

- CLI komutu 478, 481, 490-1
- CLR komutu 228
- CLV komutu 478, 481, 490-1
- CMOS yarıiletken bellek yongası 312, 324
- CMP komutu 220, 223, 245, 478, 482, 490-1
- COM komutu 194, 259
- COPY komutu 434
- CPX komutu 232, 478, 482, 490-1
- CPY komutu 232, 478, 482, 490-1
- CRT terminalleri 269; bkz. görsel görüntüleme birimleri
- CRT (katot-ışınlı tüp); 288-91
- CSMA (taşıyıcıya duyarlı çoklu erişim) 424
- CU'lar (denetim birimleri) 188, 199-214
- çakışan-akımla seçme tekniği 331-2
- çakışma, yol ağları 423
- çakışma denetimi, yol ağları 422-5
- çapaklar 101
- çapraz-bağlı VEYADEĞİL kapıları 69-70, 73
- çapraz-çevirici program 437
- çarpımların toplamı biçimi, toplamların çarpımı biçimine dönüştürme 50-2
- çarpma 4, 376-87, 388
- 6502-mikroişlemcisi için algoritma 378-9, 380
- Boole anlamı 33
- çarpma yoluyla bölme 391-4
- ilişkin ikili tablolar 136, 376
- mantık operatörü 33, 190, 377
- mikroişlemcilerde olmayışı 232, 376
- ondalık 194-5
- yüksek hızda 383-7, 388
- çarpma devreleri
- mantık kapıları kullanılarak tasarımı 40-2, 42-3, 383-4
- ROM arama-tablosu 102-3, 385, 386
- çarpma (sistem arızası) 6; ayrıca bkz. kafa-çarpması
- çekici-beslemeli yazıcı mekanizması 296
- çerçeve denetim dizisi (FCS) 404
- çevirici programları 197, 438
- çevirici yönergeleri 197
- çevre birimleri 3, 269, 284-307
- ana bilgisayar/mini bilgisayar 11
- içindeki mikroişlemciler 284
- ilgili kitaplar 495
- ayrıca bkz. giriş/çıkış birimleri
- çevresel işleme birimi (PPU) 283
- çift-duyarlı mod 165
- çift-sıralı (DIL) paketler
- birleşik mantık elemanları 59, 60
- dinamik RAM 318
- ROM 322
- ardışıl mantık elemanları 89-96
- çizgi, Boole anlamı 34
- çıkarma devreleri 143
- çıkarma
- assembly dili komutu 158
- Boole cebirinde tanımlı olmayışı 71
- ikili tablolar 136
- işaret-ve-büyüklik aritmetiği 144-5
- mantık operatörü 190
- çok geniş ölçekli entegrasyon (VLSI) devreleri 58-9
- çoklu programlama 5, 439-43
- avantajları 440
- çok-diskli sürücüler 349
- çoğullayıcılar
- DIL paketleri 55-6
- iki-girişli 27-8, 85
- ilişkin Boole ifadeleri 36-9
- uygulaması 29
- çoğunluk-mantık devresi 9, 25-6
- çözümlü örnekler 171-77, 358-59, 459-76
- D kaydedicisi 258
- DAC (sayısal-analog dönüştürücü) 307
- daire, alan hesaplama programı 185
- dallanma, terim anlamı 192
- dallanma komutları, listesi 192, 487-88
- dallanmaya bağlı adresleme 253-5
- darbe aralıkları, saat 188
- darbe etkileri
- JK flip-flopu 86-7
- RS flip-flopu 75
- telgraf distorsiyonu 405

- darbe üreteçleri 293, 442
- de Morgan teoremi 35-6
- uygulamaları 39-43
- DEĞİL (NOT) kapıları 23-4, 25
- mantık sembolü 23, 34, 190
- yayılma gecikmesi etkileri 100-1
- DEĞİL (NOT) operatörü 24, 25, 190
- DEC, tarafından kullanılan ağlar 425
- DEC komutu 193, 194, 220, 221, 259, 478, 481, 490-1
- delikli kağıt bant
- üzerindeki veri 17
- düzenlenmeleri 436
- yapısal bellek olarak 310
- delikli kartlar
- düzenlenmeleri 436
- yapısal bellek olarak 309
- denetim alanları, veri iletim protokolü 404, 410, 411-2
- denetim birimleri (CU'lar) 188, 199-214
- karşılaştırılması 214
- mikroprogramlı 199-205, 212-4
- rastgele mantık 205-12
- denetim elemanı, olarak bilgisayar 9-10
- denetim karakterleri
- ASCII kodu 118-9, 286, 411-2
- klavyeden üretilen 294-5
- VDU'larda kullanılan 294-5
- veri iletim protokolü 410-2
- denetim komutları 440, 489-90
- denetim saklama ortamı 204
- denetim tuşları 286
- denetleyici çerçeveler, HDLC protokolü 413, 414
- değerlendirme testleri 477
- değilleme
- Boole anlamı 34
- mantık operatörü 34, 190
- değiştirilebilir disk paketleri 352
- derlenmiş programlar 432
- derleyici program 439
- devre çizim kuralları 20
- devre sembolleri, kapılar 20, 21, 22
- DEX komutu 228, 232, 478, 482, 490-1
- DEY komutu 232, 478, 482, 490-1
- diferansiyel faz modülasyonu 409
- dikkate alınmaz durumlar 55-8, 73, 198
- DIL (çift-sıralı paket) 59, 60, 89-96, 318, 322
- dinamik bellekler 311
- denetim devreleri 320-1
- dinamik yarıiletken bellekler 316-21
- dezavantajları 320
- disk denetleyici 350
- disk dosyaları 6
- disk formatları 350, 356
- disk sürücüler 271, 348-59
- dönme gecikmesi 351
- erişim süreleri 338, 351, 358
- fiyatları 349
- hareketli-kafalı 351-2
- kafanın oturma süresi 3521
- sabit-kafalı 352
- saklama kapasitesi 338, 349, 351, 354-55, 358
- terminoloji sözlüğü 351-2
- diskler
- fiziksel/mantıksal-bölümlemiş 352, 356
- tanımları 348
- ayrıca bkz. disket ...; Winchester...
- diyez işareti (#) 223, 225, 226, 436
- DMA (doğrudan bellek erişimi) 270, 280-2
- DMAC (doğrudan bellek erişim denetleyicisi) 270, 281-2
- dolaylı adresleme 236-8, 208, 256-7, 479, 480
- dolaysız adresleme 225-6, 237, 250-1
- örnekleri 226-9
- domenler, ferromanyetik malzemede 327, 328, 333
- donanım, tanımı 1
- dondurucu, olarak D flip-flopları 77
- doğal adresleme 217, 252, 257
- doğrudan adresleme 224, 252
- doğrudan bellek erişim denetleyicisi (DMAC) 270, 281-2
- doğrudan bellek erişimi (DMA) 270, 280-2
- çevrim-çalma modu 281-2
- doğruluk tabloları

BCD/7-parçalı gösterge 470  
 Boole operatörleri 34  
 çarpma devresi olarak ROM 102  
 D flip-flopları 76  
 flip-floplar 73, 76, 83  
 iki-bitlik çarpma devresi 40  
 kapılar 21, 22, 23, 24, 25, 26, 67  
 klima örneği 55  
 RS flipflop 72, 73  
 tam-toplayıcı 138  
 tekilleyiciler 62  
 üç-durumlu çıkışa sahip tersleyici 67  
 X sembolü, anlamı 57, 67, 73  
 VEYADEĞİL (NOR) kapısı 70  
 yarım-toplayıcı 137  
 doğum öncesi, bileşen güvenilirliği 107  
 dosyaadı, tanımı 434  
 dosyaadı uzantısı 434  
 dot-matris yazıcılar 297-8  
 baskı kalitesi 297-8  
 döngü yapıları 226-38  
 DRO (bozucu okuma) aygıtları 330-1  
 durdurma noktası işaretleri 439  
 durum diyagramları 87-8  
 duyarlık, kayan-noktalı sayılar 161, 164  
 duyarlı-plakalı tuşlar, klavyedeki 285  
 düğümler, yerel iletişim ağları 419  
 düzeltilmiş frekans modülasyonu (MFM)  
 kodlama teknikleri 346-7, 354  
 düzenleyici programları 436-7  
 D-tipi flip-floplar 76-7, 82  
 kaydırmalı kaydediciler olarak 82-4  
 MİB içindeki kaydediciler olarak 199  
 sekizli kaydediciler olarak 91-2  
 sembly dili kullanımı 225  
 EAROM (elektriksel olarak değiştirilebilir  
 sadece-okunur bellek) 325  
 ekonomiklik, sayısal tasarım 96-7, 101-2  
 ekran düzenleyicileri 436  
 elastometrik klavyeler 285  
 elde flip-flop 191  
 eldeden ötesini gören devreler 143  
 elde-bayrak bitleri 191

elde-etrafında-bitiş 152  
 elde-üzerinden dalgalanma 143  
 elektriksel olarak değiştirilebilir sadece-okunur  
 bellek) EAROM 324  
 elektron dönme etkileşim 309, 326  
 Elektronik ve Elektrik Müh. Enstitüsü (İTİB)  
 kayan nokta formatları 165-7  
 yolu 284  
 elektronik mühendisleri, mikroişlemciler  
 açısından 9  
 elektronun ağırlığı ifadesi 160  
 eleman, olarak bilgisayar 9-10  
 emülasyon mikroprogramları 204-5  
 emülatör programları 438  
 en büyük değerlikli bit (EBDB)  
 eşlik biti olarak 132  
 işaret biti olarak 144, 149  
 ENQ (karşılık-isteği) mesajı 119, 412  
 EOR komutu 220, 478, 484, 490-91  
 EOR (Özel-VEYA) operatörü 21, 190  
 çözümü örnek olarak 468  
 EOT (bandın sonu) işareti 361  
 EPROM (silinebilir programlanabilir  
 sadece-okunur bellek) 322-4  
 eriyen tel bağlantılar, genel-amaçlı mantık  
 dizilerinde 103  
 erişim süresi  
 tanımı 311, 317  
 disk sürücü 338, 351, 352, 358  
 ferritçekirdek 331-2  
 bellek 337, 338  
 erteleme, yollardaki çakışma denetimi 426  
 Ethernet 397, 425  
 kullandığı çarpışma denetimi 426  
 etiket alanları, assembly dili programı 197  
 etkileşimli işletim sistemleri 432-3  
 ETX denetim karakteri 119, 411  
 EXG komutu 219, 259  
 Eşitlik test edici devre 31, 32  
 eşlik bitleri/kodları 131-2, 400  
 çift/tek 131  
 enine/boyuna 132  
 eşzamanlı satranç, ile çoklu programlamanın

karşılaştırılması 440  
 kodlama (PE) tekniği 344-5, 363, 366, 401  
 modülasyonu (PM) 407  
 dan bitler 131  
 dan eklemeler, sistem güvenilirliği 109  
 (çerçeve denetim dizisi) 404  
 ler (ön-işlemciler) 283  
 çekirdekli bellekler 325-32  
 avantajları 325  
 çekirdek bellek düzlemi 331-32  
 histerezis eğrisi 327, 330  
 kullanımdan kalkmaları 325  
 okuma etkileri 330, 331, 332  
 teorik olarak öğretilme nedeni 325  
 yazma yapılması 331, 332  
 ferromanyetizma 309, 326  
 fayatlar  
 disk sürücüler 322, 349  
 kaset kayıt cihazları 367  
 mikrobilgisayarlar 12  
 mini bilgisayarlar 11  
 yazıcılar 295, 297  
 fiziksel adres alanı 444, 445  
 mantıksal adres alanıyla ilişkisi 446  
 sanal bellekle ilişkisi 447  
 fiziksel katman protokolü 415, 417  
 fiziksel-bölümlemiş diskler 352  
 Flex işletim sistemi 432  
 yardımcı programları 434-5  
 flip-floplar 3, 69-83, 309  
 ardışıl devrelerin temeli olarak 69, 83  
 bellek hücreleri olarak 309, 311, 313  
 uygulama alanları 93  
 ayrıca bkz. iki-kararlılar...  
 floppy-disk sürücüler 352-5  
 çözümü örnek 358  
 floppy-diskler, tanımı 352  
 FM (frekans modülasyonu) 345, 354, 407  
 font, VDU ekranda 291  
 formatlama, mantıksal-bölümlemiş diskler 352  
 formatlar, disklerde 350, 356  
 FORTRAN programları, dallanma örneği 28  
 FRED bellek konumu 152

frekans modülasyonu (FM)  
 kodlama teknikleri 345, 354  
 ses kasetleri 367  
 veri iletimi 407  
 gadolinyum galyum garnet (GGG) 368-69  
 GCR (grup kodu gösterimi) kodlama teknikleri  
 347-8  
 gecikme elemanları, D-tipi flip-flopun gecikme  
 elemanı olarak kullanılması 79  
 gecikme etkileri, mantık elemanları 78-9, 100-1  
 gecikme-hatlı bellekler 309  
 geçici bellek, tanımı 311  
 geçitler, yerel iletişim ağları 419  
 geçişli yapma, VDU'larda 291  
 Geniş alanlı iletişim ağları (WAN) 396  
 genişletilmiş adresleme 252  
 genlik modülasyonu (AM) 407  
 gerçek-zamanlı işletim sistemleri 433  
 gerçek-zamanlı sistemler, kesme-yönetim 279  
 gergi-kolu teyp tamponu 360-1  
 geri getirmeli bölme 390, 392  
 geri getirmesiz bölme 390-1, 394, 395  
 geri uçuş, CRT ekranda 291  
 getirme saykılı 188  
 getir/yürüt saykılı 188  
 işlem sırası 201  
 GETİR/YÜRÜT flip-flop 211  
 GGG (gadolinyum galyum garnet) 368-69  
 giriş/çıkış (G/Ç) birimleri 3, 284-307  
 ilgili kitaplar 492  
 klavye 284-8  
 VDU ekranları 288-95  
 yazıcılar 295-302  
 giriş/çıkış (G/Ç)  
 bellek-haritalı 273-4  
 kanal 282-4  
 kesme ile sürülen 276-80  
 kullanıldığı yerler 269  
 programlı 273-6  
 stratejileri 271  
 golf-topu yazıcılar 298  
 gönderirken-dinle-stratejisi, veri iletimi 424  
 göndermeden-önce-dinle stratejisi,

veri iletimi 424  
görevler arası geçişler 442-3  
görsel görüntüleme birimleri (VDU'lar) 284-95  
doğrudan veri girişi/sorgulama 7  
dan/ya veri aktarımı 269-70  
ekran formatı 291-92  
giriş olarak klavye 284-88  
çıkış olarak ekran 288-95  
gözetmen, sistem 431  
Gray kodu 128-30  
grup kodu gösterimi (GCR) kodlama teknikleri 347-8  
güç kaynağı kesilmesi, sistem kapanması 279  
güneş, ağırlık ifadesi 160  
gürültü bağışıklığı, veri kodları 342  
gürültü  
elektronik 131  
tanımı 98  
güvenilirlik, tanımı 104  
hakkında bilinen yanlışlar 104  
güvenilirlik faktörleri 104-11  
G/Ç Bkz: Giriş/Çıkış  
halka ağları 423, 426  
çift-halka yapısı 423  
denetim teknikleri 422-3  
Hall-etkili anahtar 285-286  
Hamming uzaklığı, birim uzaklık kodu 128  
hareketli-kafa disk sürücüler 351-2  
hata algılama kodları 131-3, 399-400  
hata ayıklama, tanımı 153  
hata ayıklama programı 438-39  
hata mesajları 431  
hava aralığı, manyetik kayıt 339-40, 349  
hava modeli örneği 7-8  
Hawaii Üniversitesi yerel iletişim ağı 424-5  
HDLÇ (yüksek düzeyli veri bağlantı denetimi) 403-5, 410, 412-18  
Hertz (Hz) 342  
hesap makinesi, olarak bilgisayarlar 7-8  
histerezis eğrisi, ferrit çekirdekli bellekler 326, 330  
Huffman kodları 133-5, 405  
Hz (Hertz) 342

IAS (ani erişimli saklama ortamları) 4, 6, 9, 338, 451-2  
IBM  
elektrikli daktilolar 298  
disket formatı 355-6  
disket sürücülerin gelişimi 352  
ön-işlemci 283  
veri iletim protokolü 410, 411-2  
ICL-1900 serisi bilgisayar, kullandığı sözcük-uzunluğu 117  
içiçe altyordamlar 244  
IEEE kayan-nokta formatları 165-7  
IEEE-488 standardı yol 284  
IF...THEN dallanmaları 28  
ikili alfabe, sayısal bilgisayar 17  
ikili ağaçlar, Huffman kodu 133  
ikili aritmetik 3, 136-43  
bölme 387-95  
çarpma 376-87  
problem soruları 178-82  
ikili kodlanmış ondalık (BCD) kodu 127-8  
kod çözücü devreleri 63-4, 469-70  
saklama sınırları 128  
ikili sayıcılar 87-8  
pratik paketler 94-6  
çözümlü örneklerde 464-5  
ikili sayılar  
avantajları 120  
basamak notasyonu 120  
ondalıktan/ğa dönüştürme 121-4  
Gray kodundan/a dönüştürme 129  
onaltılıdan-sekizliden/onaltılıya-sekizliye dönüştürme 124-5  
iki'ye tümler aritmetik 146-50  
çarpma 381-2  
iki'ye tümleyen sayılar 146-8, 154  
alternatif bakış 150-1  
özellikleri 148-50  
toplama/çıkarma devreleri 148  
iki-kararlılar 69-83, 309  
ana-uydu flip-floplar 81-2  
D-tipi flip-floplar 76-7, 83-4, 90-1, 199  
JK-tipi flip-floplar 82-3, 86-90, 209, 210,

463, 465  
RS-tipi flip-floplar 70-5, 211-2, 318,  
saatlenmiş flip-floplar 75, 78-82  
T-tipi flip-floplar 94, 95  
tanımı 69  
ileri bilgisayar aritmetiği 376-95  
iletişim sistemleri 4  
gerektiren nedenler 396  
ilişkin kitaplar 493  
ilk-giren-ilk çıkar kuyruğu 275  
imleç, VDU ekranları 294  
INALL altyordamı 246  
INC komutu 193, 194, 220, 221, 259, 482, 490-1  
ince-film bellek 332-7  
kolay/zor eksenler 333, 334  
okuma/yazma işlemleri 334-6  
indeks deliği, disket 353  
indeks kaydedicileri 185-6, 227-8  
çeşitli bilgisayarların karşılaştırılması 231, 255  
problem soruları 267-8  
ayrıca bkz. X..., Y kaydedicileri  
indekslenmiş adresleme 229-34, 255-6, 479-80  
örnekler 233-4  
iniş alanı, Winchester disk sürücü 357  
ink-jet yazıcılar 301-2  
INX komutu 228, 232, 478, 483, 490-1  
INY komutu 232, 478, 483, 490-1  
IR (komut kaydedicisi) 188  
istek sayfalaması 453  
istek-modlu işletim sistemleri 431-3  
istenen-kalitede basım yapan yazıcılar 299  
izleme işlevi, hata ayıklayıcı program 438  
izler, manyetik disk 348, 350  
iş denetim dili (JCL) 430, 432  
işaretleme durumu, asenkron seri iletim 399  
işaretili ikili çarpma 380-3  
işaretsiz ikili aritmetik  
bölme 377-8  
çarpma 377-80  
diyez (#) işareti  
assembly dili komutlarında 223, 225, 226  
program satır numarası olarak 436

işaret-ve-büyüklik gösterimi  
kayan noktalı sayılarda mantis 166  
negatif sayılar 144-5, 153  
işlemci durum kaydedicisi (PSR) 29, 154-5, 191  
işlemci durum sözcüğü (PSW)  
bitleri listesi 154-5, 191, 478  
kesme bayrak biti 276  
işlemciler; bkz. mikroişlemciler  
işletim sistemi yardımcı programları 434-6  
işletim sistemleri (OS'ler) 5, 430-43  
etkileşimli 432-3  
gerçek-zamanlı 433  
istek-modu 432-3  
işlevleri 430  
kullanım örneği 431-2  
orkestra şefi ile karşılaştırması 430  
tekkullanımlı 432  
toplu-mod 432  
türleri 432-3  
izgara, CRT 289-90  
ışık yayan diyot (LED), kod çözme devresi 63-4  
JCL (iş denetim dili) 430, 432  
JK tipi flip-floplar 82-3  
çözümlü örnekler 463-65  
darbe etkisi 87-8  
durum diyagramı 87  
ikili sayıcı 87-90  
kaydırmalı kaydedici 86  
üç-hattan-sekiz-hatta kod çözücü 209-10  
JMP komutu 194, 220, 224, 478, 488, 490-1  
JSR komutu 244, 246, 480, 489, 490-1  
K, bellek kapasitesini göstermek için kullanılan 312  
kafa çarpması 350  
kamış röle anahtarları 285, 304  
kanal denetim sözcüğü (CCW) 283  
kanal G/Ç 282-3  
kağıt bant 17, 310, 437  
kapalı-döngü veri aktarımları 271  
kapasitif anahtarlamalı klavye 285  
kapılar 3, 18-25  
analog örnek 18-19  
elektiriksel özellikleri 97-100

tanımı 18  
 uygulamaları 25-33  
 yayılma gecikmesi etkileri 100-1  
 tipleri 16  
 ayrıca bkz: VE...; ÖZEL VEYA...;  
 VEDEĞİL...; VEYADEĞİL...; DEĞİL...;  
 VEYA kapıları  
 kaplanmış-tel bellekler 332, 336-7  
 karakter üretici ROM 294  
 karakterler, sözcük olarak 118  
 karakter-yönelimli iletim 401, 410-2  
 Karnaugh diyagramları (K diyagramları) 16, 44-8  
 çizimi 45-50  
 çözümlü örnekler 460, 471, 472  
 dikkate alınmaz durumlar 55-8  
 bir değişkenli 45  
 dört değişkenli 45, 47-50  
 iki değişkenli 45  
 kullanarak basitleştirilmiş Boole ifadeleri 47, 49-50  
 örnekler 49-50  
 pratik örnek 52-5  
 biçimi 49  
 uygulamaları 502  
 üç değişkenli 45-7  
 kartuşlar, teyp ünitesi 367  
 karşılık isteği (ENQ) mesajı 119, 412  
 karşılıklı değiştirme komutları 219, 259  
 kaset kayıt cihazları 11, 364, 366  
 kasetler, teyp ünitesi 364-5  
 katalog fonksiyonu 434  
 katot-ışınli tüp (CRT) 288-90  
 nokta-çizme modu 288  
 raster-taramalı mod 290  
 kayan-mekikli teyp tamponları 361  
 kayan-noktalı sayı aritmetiği 167-71  
 çözümlü örnekler 171-77  
 gösterimi 161-2  
 normalize edilmeleri 162-4  
 problem soruları 183-4  
 saklama yöntemleri 162, 165-7, 171  
 tipik sistemler 164-7

kayan-noktalı sayılarda duyarlık 161  
 kaydedici adresleme, 6809-mikroişlemci 257  
 kaydedici-aktarım dili (RTL) 187, 188, 190, 201, 208, 238, 257, 481-89  
 kaydedici-ekleme-halkaları 427  
 kaydırma (shift) tuşu 286  
 kaydırmalar, tipleri 86  
 kaydırmalı kaydediciler 83-6  
 çözümlü örnekler 462-3  
 D-tipi flip-floplarla yapılan 83-4  
 JK-tipi flip-floplarla yapılan 85-6  
 manyetik kabarcıklı bellek olarak 371, 372  
 paralel-seri dönüştürücü olarak 84  
 paralel yükleme kapasitesiyle 85  
 pratik paketler 94, 95  
 seri-paralel dönüştürücü olarak 84-5  
 kaydırma-ve-toplama, ile çarpma yöntemi 377  
 kayışlı yazıcılar 301  
 kaynak kodu 432  
 kaynaklar 492-4  
 kendinden-saatlemeli kodlar 342, 344, 345, 346  
 kendini-değiştiren programlar 230  
 bellek haritası 232  
 karşı düşünceler 230-2  
 kesinlik, sayıların 161-2  
 kesirler  
 kayan-noktalı sayı aritmetiği 162  
 sayı-tabanı dönüştürme 125-6  
 problem soruları 178-9  
 kesme alındı mekanizması 278  
 kesme istek girişleri 276  
 kesme mekanizması 249  
 kesme sürücülü G/Ç 276-80  
 kesme teknikleri, kayan-noktalı aritmetik 169, 171  
 kesmeler, dezavantajları 280  
 kilitlemesi, sistemin 273  
 kirli-bit 455  
 kısıtlamasız-topoloji ağları 420  
 dezavantajları 420  
 kısmi bölümler 390  
 kısmi bölünenler 389  
 kısmi çarpımlar 377

kızak, disk sürücü 349  
 klavye kodlayıcısı 286-7  
 klavyeler 284-7  
 çalıştıracak yazılım 287-8  
 kod çözücüler  
 BCD/yedi-parçalı gösterge 63-4, 471-2  
 iki-hat giriş/dört-hat çıkışlı 63, 91-2  
 üç-hat giriş/sekiiz-hat çıkışlı 63, 91-2, 209, 210  
 kodlama teknikleri 341-48  
 kodlayıcılar  
 ikili/Gray kodu 129-130  
 optik 129-130  
 ayrıca bkz. kod çözücüler  
 kod-dönüştürücü devreler 63-4  
 kompozit (karma) video 291  
 kompozit (karma) video sinyali 294  
 komut kaydedicisi (IR) 187  
 komut kısaltmaları, assembly dili 154, 477-78  
 ayrıca bkz. komutlar  
 komut kod çözücü, rastgele-mantık denetim birimi 211, 212  
 komut sayıcıları 89, 187  
 komutlar  
 aktarım 219, 258-9, 485-6  
 aritmetik işlemler 482-3  
 dallanma 192-3, 487-89  
 denetim 488-9  
 indeks kaydedicisi 232  
 mantıksal işlemler 220, 259-60, 483-4  
 sözcük olarak 117-8  
 veri taşıma 219, 258, 485-7  
 veri üzerinde etkileri 221-2  
 uygulamaları 221-4  
 6502-mikroişlemcisi komutları listesi 193, 218-24, 477-90  
 6809-mikroişlemcisi komutları listesi 256-9  
 komutların yürütülmesi  
 mikroprogramlanabilir denetim-birimi 199-205  
 rastgele-mantık denetim-birimi 209-14  
 kontrol bitleri 131  
 konumdan-bağımsız kod 234  
 konumsal notasyon 118-9  
 koruma işlevi 435

koşul kodu kaydedicisi (CCR) 154, 191  
 koşullu dallanma 28, 192, 193, 204, 220  
 köşeli parantez [ ], anlamı 15, 187  
 kullanıcı-yönelimli arabirimler 302-7  
 kullanılabilirlik oranı 106  
 kurallar, mantık elemanları 19-20  
 kurma (set), anlamı 72  
 küçük ölçekli entegrasyon (SSI) devreleri 58, 59  
 LAN'lar (yerel iletişim ağları) 397-8, 418-29, 492-3  
 LDA komutu 154, 190, 194, 217, 480, 487, 490-1  
 LDX komutu 219, 232, 478, 485, 490-1  
 LDY komutu 232, 478, 485, 490-1  
 LEA komutu 258  
 LED'ler (ışık yayan diyet) 63-4  
 LIFO (son-giren-ilk-çıkart) yığın kuyruğu 239  
 LIST komutu 435  
 LSI (büyük ölçekli entegrasyon) devreleri 58  
 LSR komutu 220, 478, 484, 490-1  
 Makine-kodu komutları 3, 199  
 yorumlanması 213  
 makrokomutlar 199  
 Manchester kodlama tekniği 345  
 mantis, kayan-noktalı sayılar 162, 165  
 mantık elemanları 3  
 birleşik (kombinasyonel) 16, ayrıca bkz. kapılar  
 elektriksel karakteristikleri 17-8, 97-101  
 garanti edilmiş gürültü bağımsızlığı 18  
 gerilim karakteristikleri 17-8, 98  
 girişleri 21  
 giriş-çıkış aralıkları 17-8  
 gürültü aralığı 98  
 ikili durumları 17  
 maksimum akım 98-9  
 özel-amaçlı 58-64  
 performans bilgileri 98  
 problem soruları 111-6  
 sıcaklık etkileri 97-8  
 sıralı 14 ayrıca bkz. flip-floplar  
 zamanlama karakteristikleri 100-1  
 mantık değerleri, tanımı 18

- mantık değişkenleri  
adları 18  
yüksekte-etkin/alçakta-etkin 18  
mantık devresi tasarımı, etkileyen faktörler 33-4, 96-104  
mantıksal adres alanı 445  
fiziksel adres alanı ile ilişkisi 446  
mantıksal komutlar 220, 259-60, 495-6  
mantıksal operatörler 33-4  
aritmetik eşdeğerleri 190  
mantıksal-bölünlenmiş diskler 352, 356  
manyetik anizotropi 333, 368-69  
manyetik diskler  
bellek karakteristikleri 339  
iz/sektör 348, 350  
ses kayıt cihazları ile karşılaştırması 348-49  
ayrıca bkz. diskler...  
manyetik kabarcıklı bellekler 310, 368-72, 373  
algılama mekanizmaları 372  
büyük-küçük döngü organizasyonu 372, 373  
iki-durumlu mantık devresi 371  
ilişkin modüller 373  
kaydırmalı kaydedici olarak 371  
manyetik kayıt, teorisi 339-48  
manyetik teyp  
bellek karakteristikleri 339  
ayrıca bkz. teypeler ...  
MAR (bellek adres kaydedicisi) 188  
maskelenemez kesme (NMI) 279, 442  
MBR (bellek tampon kaydedicisi) 188  
merkezi işlem birimi (MİB)  
adres erişim yolları 186, 187  
erişim süresi 337  
iç kaydediciler 338  
programlama örneği 193-7  
programların yürütülmesi 14, 186  
RAM'a bağlantısı 315  
veri akış denetimi 199, 200, 280  
veri işlemede 6  
yapısı 186-93  
metin dosya adları 435  
metin işlemciler 437-8  
MFM (düzeltilmiş frekans modülasyonu)
- kodlama tekniği 346-7, 354  
mikrobilgisayarlar, ayrıca bkz. mikroişlemciler  
mikroişlemciler  
kullanılan işletim sistemleri 432-3  
kullanılan sözcük uzunluğu 117  
veri iletim gerekleri 418  
6502 214-24  
adresleme modları 224-50, 479-80  
bağıl adresleme modu 234  
çarpma algoritması 379-80, 381  
dallanma komutları 193, 221, 235, 487-89  
dolaylı adresleme 236, 479  
iç yapısı 215 indeks kaydedicisi 230  
kaydedicileri 217-8  
kesme yönetim mantığı 276-8  
komut takımı 194, 218-24, 477-91  
uygulamaları 221-4  
kullanılan veri sözcükleri 153  
opkod listesi 490-1  
sıfır-sayfa adresleme 224-5, 479  
veri taşıma komutları 219-20, 485-7  
yığın 240-2  
6800 253  
adresleme modları 224-5, 234, 251  
bağıl adresleme modu 234  
çıkarma komutları 146  
dallanma komutları 235  
indeks kaydedicisi 232  
sıfır-sayfa adresleme 224-5, 251  
6809 193, 250-61  
adresleme modları 225, 250-7  
iç kaydedicileri 251  
indeks kaydedicisi 232  
sıfır-sayfa adresleme 225, 249-50  
Z80 214, 219, 233, 253  
mikrokomutlar 190, 191, 199-200  
mikroprogramlama 191, 199-205  
mikroprogramlanabilir denetim birimleri ile karşılaştırması 214  
mikroprogramlanmış denetim birimleri 199-205, 224  
rastgele mantık denetim birimleri ile karşılaştırılması 224

- yapısı 200  
mimarisi 3  
ilişkin kitaplar 492  
6502 mikroişlemcisi 154  
mini disketler 354  
minibilgisayarlar açıklaması 11  
kullanılan önbellek 458  
veri iletim gerekleri 418  
mini-Winchester disk sürücüsü 358  
mintermler 28  
Karnaugh haritası çizimi 47, 49  
modelleme hesaplamaları 7-8  
modemler 407, 409  
modülasyon sistemleri 405-09  
mod-2 aritmetiği 136  
monitör sistemi 431  
Morse kodu 17, 134, 405  
Motorola mikroişlemcileri  
6800 253  
6809 193, 214, 250-61  
kullanılan sanal bellek teknikleri 454-6  
MPLX (çoğullayıcı) devreler 26  
MS (ana saklama ortamı) 188  
MS (ana/uydu) flip-floplar 81-2  
MSB (en büyük değerlikli bit) 132, 144, 149  
MSI (orta ölçekli entegrasyon) devreleri 58, 61  
MTBF (arızalar arası ortalama süre) 104  
MTTR (onarımına kadar ortalama süre) 104  
mutlak adresleme 224, 236, 237, 479, 480  
MİB; bkz. merkezi işlem birimleri  
NAK mesajı 119, 412  
NAM yönergesi 197  
NAND operatörü 190  
NDRO (bozucu-olmayan okuma) modu 337  
NEG komutu 259  
negatif sayılar 143-53  
çarpma 380-1  
işaret/büyükölçek gösterimi 144-5  
negatif-bayrak biti 191  
nicemleme 305  
NIU'lar (ağ arabirim üniteleri) 397  
NMI (maskelenemez kesme) 279, 443  
nokta saati, VDU ekranında 291
- noktalı grafikler, basımı 298  
nokta-çizimli CRT ekranı 288  
NOP komutu 478, 489, 490-1  
NOS (yığılma bir sonraki) konumu 239  
NRZ1 (değiştirilmiş sıfıra dönüşsüz) kodlama teknikleri 343, 344  
numarasız çerçeveler, HDLC protokolü 413, 414  
ofis-otomasyon olanakları 397  
ofsetler  
indekslenmiş adresleme 231, 255, 479  
bağıl adresleme 234  
okuma/yazma kafaları 339-4, 349, 353  
kaset kayıt cihazları 365  
teyp birimi 362  
onaltılı sayılar  
avantajları 118-20  
ikiliye/den dönüştürme 123  
ondalığa/tan dönüştürme 121-22  
çözümü örneklerde 473-4  
onaltı-bitlik mikroişlemciler, kullanılan sanal bellek teknikleri 454-6  
onaltı-bitlik sözcükler, 8-bit sözcüklerle gösterimi 156  
onarımına kadar ortalama süre (MTTR) 104  
onay 271-3  
ondalık sayılar  
BCD koduna dönüştürme 127  
ikili/sekizli/onaltılı ya/dan dönüştürme 121, 124  
Gray koda/koddan dönüştürme 129  
konumsal gösterim 118-20  
onluk sayıcı, çözümü örneklerde 466  
on'a tüm aritmetik 144-5  
opkodlar 187-8  
6502-mikroişlemcisi 492  
mikroprogramlanabilir denetim birimi 205  
optik kodlayıcılar 129-30  
ORA komutu 194, 220, 480, 485, 492-3  
ORG yönergesi 197  
orta ölçekli entegrasyon (MSI) devreleri 58, 61  
OSI (açık sistemler arabacılığı) referans modeli 418  
OS'ler (işletim sistemleri) 5, 431-44

- otomatik uçak iniş örneği 8-9  
 oturma katmanı, veri iletimi 417, 418  
 OUT komutu 193, 194  
 ömür-süresi eğrileri, güvenilirlik 106  
 öncelikli kesmeler 279, 442-3  
 önerilime-dönmeli (RB), veri kodlama tekniği 342-3  
 ön-ayarlı, flip-flop 91  
 ön-bellek (cache memory) 456-8  
 ön-işlemci (FEP) 283  
 ÖZEL VEYA kapıları 22, 25, 30-1  
 eşitlik test edici devre 31, 32  
 ikiye tümleyen toplama/çıkarma devresi 148  
 mantık sembolü 30  
 ÖZEL VEYADEĞİL (EXNOR) kapıları 25  
 özel-amaçlı mantık elemanları  
 ardışıl devreler 90-96  
 birleşik devreler 58-64  
 papatya-çarklı yazıcılar 299  
 paralel devreler, tanımı 143  
 paralel toplayıcılar 142-3  
 çözümlü örneklerde 466-7  
 paralel-seri dönüştürücüler, kaydırmalı  
 kaydedici olarak 85  
 paralel-yükleme yeteneği, kaydırmalı kaydedici 85, 94  
 parametre aktarma 246-50  
 parazit paketleri, yollardaki çakışma denetimi 427  
 PC (program sayıcısı) 29, 89-90, 187  
 PE (faz kodlama) teknikleri 344-5, 363, 366, 401  
 Permalloy, manyetik kabarcıklı bellekte 370  
 PHA komutu 219, 220, 241, 478, 485, 490-1  
 PHP komutu 478, 485, 490-1  
 PLA komutu 219, 220, 241, 478, 485, 490-1  
 PLP komutu 478, 485, 490-1  
 PM (faz modülasyonu) 407  
 portlar  
 veri aktarımı 269  
 giriş/çıkış (G/Ç) 269, 273-4  
 klavye 287  
 PPU (çevresel işleme birimi) 283  
 problem soruları  
 adresleme modları 263-4  
 AIM-65 mikroişlemcisi 262-3  
 ardışıl devreler 115-16  
 assembly dili 265-6  
 bellek teknolojisi 372-4  
 Boole cebri 112-14  
 ikili aritmetik 180-2  
 indeks kaydedicisi 267-8  
 kayan-noktalı sayılar aritmetiği 183-4  
 kesirler, sayı-tabanı dönüştürme 178-80  
 mantık elemanları 111-13  
 sayı-tabanı dönüştürme 178  
 program örnekleri  
 6809-mikroişlemcisi 260-1  
 dallanma 28  
 döngü yapısı 227-9  
 klavyenin çalışması 259  
 klavye/basım 245-6  
 metin/dizi eşleme 259-60  
 MİB tanıtımı 193-6  
 onaltılı karakterler 474  
 toplama-ve-çarpma 103  
 program sayıcısı (PC) 29, 89-90, 187  
 program sayıcısına bağlı adresleme 253-4  
 programcılar, tarafından bilinmesi gereken donanım bilgisi 1,2  
 programlama, sıralı yaklaşım 194  
 programlanabilir mantık dizileri 103  
 programlanabilir sadece-okunur bellek (PROM) 324  
 programlar, yemek tarifleriyle karşılaştırılması 14  
 programların yürütülmesi 13  
 programlı G/Ç 273-6  
 PROM (programlanabilir sadece okunur bellek) 324  
 protokol katmanları 414, 417-18  
 protokoller 271-3, 410-18  
 bayt denetim-yönelimli 410, 411  
 bit-yapılı 402-5, 410-1  
 ilgili kitaplar 493  
 PSH komutu 258  
 PSR (işlemci durum kaydedicisi) 29, 154-5, 191

- PSW (işlemci durum sözcüğü) 154-5, 191, 276, 480  
 PUL komutu 258  
 P/F bitleri, HDLC protokolü 414, 415, 416  
 QAM (Quadratur genlik modülasyonu) 409  
 QWERTY klavye 286  
 RAM (rastgele erişimli bellek) 310  
 RAS (satır adres strobu) 319  
 raster-taramalı 288  
 rastgele erişimli bellek (RAM) 310  
 büyüklükleri 312-3  
 görüntüleme ekranı 294  
 okuma/yazma çevrim zamanlama diyagramları 315, 317, 319  
 rastgele erişimli birimler, disk sürücüler olarak 350  
 rastgele mantık denetim birimleri 205-14  
 rastgele-erişim, tanımı 310  
 rastgele-yeniden gönderme stratejisi, veri iletimi 424  
 RB (öngerilime-dönüştürme) veri kodlama tekniği 342-3  
 RMB yönergesi 197  
 ROL komutu 194, 198, 478, 484, 490-1  
 ROM: bkz. sadece-okunur bellek  
 ROR komutu 194, 478, 484, 490-1  
 röleler, G/Ç birimleri olarak 285, 304  
 RS flip-flopları 70-5  
 darbe etkileri 73  
 GETİR/YÜRÜT flip-flopu olarak 212-3  
 uygulamaları 74-5  
 RTI komutu 227, 442, 478, 489, 490-1  
 RTL (kaydedici aktarım dili) 187, 188, 190, 201, 208, 238, 257, 481-89  
 RTS komutu 243, 246, 478, 489, 490-1  
 RZ (sıfıra-dönüştürme) veri kodlama tekniği 342  
 saat darbe aralıkları 188  
 saat üretici, VDU ekranda 291  
 saatlenmiş flip-floplar 78-83  
 ana-uydu 81-2, 84  
 düzeyeduyarlı 79, 82  
 kaydırmalı kaydedici olarak tipleri 84-5  
 kenara-duyarlı 79-80, 82, 84  
 modların karşılaştırılması 82  
 saatlenmiş RS flip-floplar 75  
 saatteki sapma 80  
 sabit-disk sürücüler: bkz. disk sürücüler; Winchester ...  
 sabit-kafalı disk sürücüler 352  
 sabit-noktalı sayı aritmetiği 159  
 sadece-okunur bellek (ROM) 9, 312  
 çarpma arama tablosu 385  
 ile değiştirilebilen mantık elemanları 96-7  
 karakter üretici 293  
 maske programlı 322  
 özel-amaçlı denetleyiciler 322  
 yarıiletken teknolojisi 322-5  
 saklama birimleri, türleri 309-10  
 saklama kapasiteleri, disk sürücü 338, 348-49, 351, 355-6, 358  
 saklı-programlı bilgisayarlar 309  
 insan beyni ile karşılaştırılması 13-4  
 sanal bellek alanı, fiziksel bellek alanıyla ilişkisi 452  
 sanal bellek sistemleri 338, 445, 447, 452-54  
 mikroişlemcilerde 454-6  
 örnek 452  
 sağa-kaydırmalı kaydediciler 83-6, 94  
 satır adres strobu (RAS) 319  
 satır düzenleyicileri 437-8  
 satır senkronizasyonu, VDU ekranında 292  
 satır yazıcıları 300-1  
 satranç oyunu, eşzamanlı oynama 410  
 SAVE komutu 435  
 sayfalama 216, 449-52  
 örnek 449-52  
 yer paylaşma ile tezatlığı 451-2  
 sayfa-hatası 454, 455, 457  
 sayfa-tabloları 453, 455  
 sayı hatları, ince-film bellekler 334  
 sayı sistemleri, bilgisayar belleği 15  
 sayı tabanları 118-20  
 dönüştürme 121-6  
 listesi 120  
 problem soruları 178-80  
 sayıcı devreleri 86-90

sayıcılar, pratik paketler 94, 95-6  
 sayılar  
 aralık/duyarlılık/kesinlik 161  
 formatı 118  
 sözcük olarak 118  
 sayıların "açılması", kayan-noktalı sayılar 167, 170  
 sayım-yapma, D tipi flip-floplar 76  
 sayısal giriş/çıkış birimleri 302-4  
 sayısal-analog dönüştürücüler (DAC'lar) 307  
 SBC komutu 159, 220, 478, 483, 490-1  
 SEC komutu 194, 220, 478, 483, 490-1  
 SED komutu 478, 483, 490-1  
 SEI komutu 478, 489, 490-1  
 sekize-yimmi (80/20) kuralı 448  
 sekizli kaydediciler 90-1  
 kaydedici dizilerindeki 91-2  
 sekizli sayılar  
 avantajları 118-9  
 ikiliden/ikiliye dönüştürme 124  
 ondalıktan/ondalığa dönüştürme 122, 123  
 sekiz-bitlik sayılar, 4-bitlik çarpma devreleri  
 kullanarak çarpım yapma 386-387  
 sektörler, manyetik disk 350  
 senkron seri veri iletimi 401-09  
 bit-senkronizasyonlu 401  
 karakter-yönelimli 401  
 sözcük-senkronizasyonlu 401-5  
 senkron (saatlenmiş) sistemler 78  
 seri toplayıcılar 141-2  
 seri veri iletimi 398-409  
 asenkron 399-400  
 senkron 400-09  
 tam-dubleks mod 398, 414, 416  
 yarı-dubleks mod 398, 414, 415  
 seri-erişim, tanımı 311  
 seri-erişimli birimler 338, 341  
 seri-paralel dönüştürücüler, kaydırmalı  
 kaydedici olarak 84-5  
 ses bobini 357  
 ses kaset kayıt cihazları, veri saklama ortamı  
 olarak kullanımı 367  
 SET komutu 193, 194

SEX komutu 259  
 SHIFT (KAYDIRMA) operatörü 190  
 silindir yazıcılar 298  
 silinebilir programlanabilir sadece-okunur bellek  
 (EPROM) 322-4  
 kızılötesi ışıkla silinmesi 324  
 silme komutları 435, 437  
 silme (clear), anlamı 72  
 sistem güvenilirliği  
 paralel devre 108  
 seri devre 107  
 sistem yazılımı 434-39  
 sistem yöneticisi 431  
 sıcaklık etkisi, devre elemanları 97-8  
 sıfır  
 işaret-ve-büyüklik aritmetiği 144  
 kayan-noktalı değeri 160-1  
 tümler aritmetik 150, 152  
 sıfıra-dönüştürme (RZ) veri kodlama tekniği 342  
 sıfıra-dönüştürme, değiştirilmiş (NRZ1) kodlama  
 tekniği 343, 344  
 sıfırlama, tanımı 72  
 sıfır-bayrak biti 191, 192, 208  
 sıfır-doldurma 427  
 sıfır-sayfa adresleme 224-5, 251  
 sınır anahtarı 303  
 sınırlamalar, donanım 2  
 SOH denetim karakteri 119, 412  
 sola-kaydırmalı kaydediciler 86, 94  
 solenoid anahtarlar  
 kaset kayıt cihazlarında kullanılan 10  
 DEĞİL (NOT) kapısı işlevi 23  
 Sommering telgrafı 405  
 son-bayt, 6809 adresleme 254-5  
 sözcük hatları, ince-film bellek 334  
 sözcük uzunlukları 11, 118  
 sözcükler  
 toplanması 140-3  
 gösterimi 118-21  
 sözcük-senkronizasyonlu iletim 400-5  
 sözcük-yapılı bellek 334  
 sözde-ROM 32  
 SP (yığın işaretçisi) 240, 241

SSI (küçük ölçekli entegrasyon) devreleri 58, 59  
 STA komutu 154, 190, 194, 219, 478, 486, 490-1  
 statik bellek, tanımı 311  
 statik RAM 317  
 statik yarıiletken bellek 312-6  
 STOP komutu 194  
 streamer (akan) teyp sürücüler 364-5  
 STX denetim karakteri 119, 411  
 STX komutu 219, 232, 478, 486, 490-1  
 STY komutu 232, 478, 486, 490-1  
 SUB komutu 152, 194  
 sunum katmanı, veri iletimi 417, 418  
 süper bilgisayarlar 8  
 sürtünmeli beslemeli yazıcı mekanizmaları 296  
 sürtünme-beslemeli yazıcı mekanizmaları 295  
 sütun adres strobu (CAS), dinamik bellek 319  
 SWI komutu 439  
 SYN denetim karakteri 119, 402, 410  
 şerit elemanı, manyetik-kabarcıklı bellek 372  
 T (dönüşümlü) flip-floplar 94  
 taban bandı, tanımı 425  
 tambur yazıcılar 300-1  
 tamponlar  
 G/Ç birimi 275  
 üç-durumlu kapı 67-8  
 tam-çıkarcı 143  
 tam-dubleks iletim 398, 414, 416  
 tam-kilitleşimli onaylama 273  
 tam-toplayıcılar 138-40  
 ardarda bağlı iki-yarım toplayıcı 139-40  
 tarama döngüsü 276-7  
 tarama mekanizması 279  
 taramalı kesmeler 279-80  
 tara/bitir (P/F) bitleri, HDLC protokolü 414, 415, 416  
 TAX komutu 219, 232, 478, 486, 490-1  
 TAY komutu 232, 478, 486, 490-1  
 tazeleme, dinamik bellek 311, 318, 320  
 taşıma katmanı, veri iletimi 417, 418  
 taşıyıcıya duyarlı çoklu erişim (CSMA) 425  
 taşıma  
 tanımı 465

ikiye tümleyen aritmetik 150  
 taşıma-bayrağı biti 191  
 TEAC MT-2 teyp kaseti 367  
 Teesside Polyteknik yerel iletişim ağı 396-7, 419  
 tekilleştiriciler 62  
 tek-kullanıcı işletim sistemleri 432  
 telefon rehberi, ön-belleğe örnek olarak 458  
 telefon şebekeleri  
 ile veri iletimi 406-7  
 karakteristikleri 406, 407  
 telgraf sistemi 405  
 telgrafın bozma (distorsiyon) etkisi 405-6  
 termal yazıcılar 296  
 tersleyiciler 16, 23-4  
 yayılma gecikme etkileri 100-1  
 teyp birimi 359-67  
 teyp kartuşları 366  
 teyp kasetleri 364  
 teyp tampon sistemleri  
 boşluklu bölme 360  
 gergi-kolu 360-1  
 kayan mekik 361  
 teypten yükleme 6  
 TFR komutu 258  
 token geçişi 426-7  
 token halkaları 426-7  
 toplama devreleri 137-43  
 toplama  
 Boole anlamı 33  
 için ikili tablolar 136  
 işaret-ve-değer aritmetiği 144-5  
 kayan-noktalı aritmetik 169-70  
 mantıksal operatör 190  
 toplama/çıkarma devresi, ikiye tümleyen sayılar  
 148-9  
 toplamaların çarpımı biçimi, çarpımların toplamı  
 biçiminden dönüştürme 49-52  
 toplu-mod işletim sistemleri 432  
 topoloji, yerel iletişim ağları 419  
 TOS (yığın tepesi) konumu 239  
 TSX komutu 232, 478, 486, 490-1  
 TTL devre elemanı, üretici verileri 98  
 tuş vurularının algılanması 287

- tuş vuruşlarının kodlaması 286  
 tümleme  
 Boole anlamı 34  
 Karnaugh diyagramı 51  
 tümler aritmetik 145-52  
 bir'e tümleyen 151-2  
 iki'ye tümleyen 146-51  
 on'a tümleyen 145-6  
 tümleyen 22-3  
 TXA komutu 219, 232, 478, 486, 490-1  
 TXS komutu 232, 478, 486, 490-1  
 TYA komutu 232, 478, 487, 490-1  
 ULA'lar (bağımsız mantık dizileri) 104  
 Uluslararası Standart Enstitüsü (ISO), protokol referans modeli 417  
 Uluslararası Standartlar Organizasyonu (ISO) protokol referansı modeli 417-8  
 Univac bilgisayarları, sayı aralığı/duyarlılığı 165  
 UNIVAC teyp birimi 363  
 UNIVAC-1100 bilgisayarları, kullanan ağlar 397  
 kullandığı sözcük-uzunluğu 117  
 üzerinde 6502 programı derlenmesi 438  
 uygulama katmanı, veri iletimi 417, 418  
 üç-durumlu kapılar 65  
 yol uygulaması 65-6  
 üç-durumlu mantık devreleri 64-9  
 üs, kayan-noktalı sayılar 162  
 üst-düzeyli diller, derleyici çevirisi 4  
 van Neumann bilgisayarı 308  
 VCR (video kaset kaydedici) streamer teyp olarak 365  
 VDU'lar, bkz. görsel görüntüleme birimleri  
 VE kapıları 20-21,25  
 3-hattan-8-hata kod çözümde 209-10  
 mantık sembolü 20  
 yayılma gecikmesi etkileri 100  
 VE mantıksal operatörü 20-21,33,34,190,377  
 VEDEĞİL (NAND) kapıları 24-5  
 hız/ucuzluk oranı 39, 42  
 ile tasarlanan mantık devreleri 52-3  
 kullanarak gerçekleştirilen mantık işlevleri 42-3  
 mantık sembolü 24  
 tipleri 42  
 ucuzluk/hız oranı 39, 42  
 vektörler, iç-çarpım hesaplaması 233  
 vektörlü kesmeler, 280  
 veri erişim yolları 200, 201  
 veri iletimi 398-409  
 ilişkin kitaplar 493  
 kapalı-döngüye karşılık açık-uçlu 271-2, 410  
 veri işlemcisi, olarak bilgisayarlar 6-7  
 veri kodlama teknikleri 341-48  
 veri taşıma komutları 219-20, 258, 485-7  
 veri-bağlantı katman protokolü 417  
 VEYA (OR) kapıları 21-3, 25  
 mantık sembolü 22  
 VEYA (OR) operatörü 21, 22, 33, 34, 190  
 VEYADEĞİL (NOR) kapıları 24-5  
 çapraz bağlı 69-70, 73  
 doğruluk tabloları 70, 71  
 kullanılarak gerçekleştirilen mantık işlevleri 43  
 mantık sembolü 24  
 VEYADEĞİL (NOR) operatörü 25, 190  
 VE-VEYA-TERSLEYİCİ kapılar 59,61  
 video kaset kaydedici (VCR) steramer teyp olarak 365  
 video RAM 294  
 VLSI (çok geniş ölçekli entegrasyon) devreleri 58-9  
 vurmali yazıcılar 295  
 WAN'lar (Geniş alanlı iletişim ağları) 396  
 Whirlwind bilgisayarı 325  
 Winchester disk sürücüler 355-8  
 ilişkin yedekleme ortamı 364  
 X kaydedicisi 218-9, 239, 254  
 Xerox, tarafından kullanılan ağlar 425  
 XOR işlevleri/kapıları 22, ayrıca bkz. ÖZEL  
 VEYA  
 Y kaydedicisi 217, 254  
 yangın dedektör devresi, Karnaugh diyagramı örneği 52-5  
 yararlılık düzeyi, mikroişlemcinin 477  
 yardımcı programlar  
 işletim sistemi 434-9

- yer paylaşma olarak 448  
 yarıiletken bellek 312-25  
 dinamik RAM 316-21  
 dezavantajları 321  
 ROM 321-5  
 statik 312-16  
 statik RAM 312-16  
 yarım-bayt (nibble) yapılı bellek 312  
 yarım-toplayıcılar 137  
 yarı-dupleks iletim 398, 414, 416  
 yayılma gecikmeleri  
 çözümlü örneklerde 468-69  
 mantık elemanları 78-9, 100-1  
 rastgele mantık denetim birimi 207  
 tam toplayıcı 138, 139  
 yazıcılar 295-302  
 dot matris 297  
 ink-jet 301-2  
 işlev listesi 295  
 kağıt-besleme mekanizması 295  
 kayışlı 300-301  
 papatya çarklı 299  
 satır 300  
 tambur 300-1  
 termal 296  
 ucuzluk/hız 276  
 vurmali 296  
 yazılım, tanımı 1  
 yazılımsal kesme komutu 439, 489  
 yedek saklama ortamları 5, 338  
 yedekleme, sistem güvenilirliği 107  
 yer paylaşmalar 447-8  
 sayfalama ile tezatlığı 451-2  
 yerel iletişim ağları (LAN'lar) 396-7, 418-29, 492-3  
 avantaj/dezavantajları 419, 428-29  
 ilgili kitaplar 492-3  
 topoloji 419-22  
 YETKİ (ENABLE) girişleri 67, 90  
 yinelenmeli hesaplamalar 7-8  
 yıldız ağları 420-1  
 ALOHA örneği 424  
 dezavantajları 421  
 yıldız (\*) işareti, program sayıcısının içeriği olarak 234-5, 246  
 yığın 219, 239-40  
 6502-mikroişlemcisi 240-2  
 altyordamların kullanımı 242-6  
 yığın işaretçisi (SP) 241, 242  
 yığının temizlenmesi işlemi 250  
 yol ağları 422  
 maksimum verimlilik 425  
 yollardaki çakışma denetimi 423-6  
 yol topolojisi 422  
 yollar 64-6  
 bağlantı problemleri 66-7  
 ve üç-durumlu devreler 67-8, 315  
 yollardaki çakışma 423  
 yorumlama, makine-kodu komutları 199, 213  
 yönergeler, assembly dilinde 197, 245  
 yöntemler, programlama 242  
 yuvarlama tekniği, kayan-noktalı aritmetik 171  
 yüksek düzeyli veri bağlantı denetimi (HDL) 403-5, 410, 412-18  
 mesaj türleri 413-4  
 yüksek hızda çarpma 383-87, 388  
 yüksek-geçirimlilikte malzeme 339-40  
 yüzde (%) sembolü 223, 226  
 yüzer/belirlenemez durumlar 67, 73  
 Z80 mikroişlemcisi 214, 253  
 kullanıldığı örnekli kitaplar 492  
 zamanlama darbe üreticileri  
 çıkışları 210  
 rastgele mantık denetim birimi 209  
 zamanlama diyagramları, bellek 317, 319, 320  
 zig-zag katlanmış kağıt 295  
 Zilog mikroişlemcisi  
 kullandığı sanal bellek teknikleri 456

## TERİMLER SÖZLÜĞÜ

**Adres yolu (Address Bus):** Üzerinden adres bilgilerinin taşındığı yol.

**Akümülatör (Accumulator):** Aritmetik ve mantık işlemleri sırasında, üzerinde işlem yapılacak verinin tutulduğu yer.

**Alçakta-etkin/Yüksekte-etkin (Low active/High active):** Bir elektronik birimin, mantıksal 0'da (denetim darbesinin mantıksal 0 düzeyinde) etkinleşmesi alçakta-etkin, mantıksal 1'de etkinleşmesi ise yüksekte-etkin olma özelliğidir.

**Altıyordam (Subroutine):** Bir programın içinde yer alan ve programda tekrarlanan belirli işlemleri yerine getirdikten sonra denetimi ana programa aktaran program parçalarıdır.

**ALU (Arithmetic Logic Unit: Aritmetik ve Mantık Birimi):** Bir merkezi işlem biriminde aritmetik ve mantık işlemlerinin yerine getirildiği birim.

**Amaç program (Object program):** Bilgisayar belleğine yüklenmeye hazır biçimde derlenmiş ya da çevrilmiş program.

**Analog (Analog):** Niceliklerin; gerilim, akım vb. gibi fiziksel değişkenlerle gösterimidir.

**Analog-sayısal dönüştürücü (Analog to Digital Converter):** Sürekli değişen analog sinyalleri, sayısal sinyallere dönüştüren elektronik devre.

**Arabirim (Interface):** Birimlerarası bağlantıyı ve bu bağlantıdaki uyumu sağlayan birim.

**ASCII (American Standard Code for Information Interchange):** Bilgi Değişimi için Amerikan Standard Kodu).

**Assembly dili (Assembly language):** Genellikle her bir komutu bir makine koduna karşılık gelen donanım-bağımlı sembolik dil.

**Baud:** Bit/sn olarak ifade edilen bit iletim hızı birimi.

**Bellek (Memory):** Sayısal bilgisayarlarda bilgileri geçici veya kalıcı olarak saklamak üzere kullanılan ve genelde yarıiletken birimlerden oluşan kesim.

**Bayrak (Flag):** Belli bir koşulu ya da durumu göstermek üzere 1 ya da 0 değeri alan bir bitlik gösterge.

Bayrak işaretleri

S - Sign (İşaret)

NZ - Not Zero (Sıfır değil)

Z - Zero (Sıfır)

C - Carry (Elde)

NC - Not Carry (Elde Yok)

AC - Auxiliary Carry (Yardımcı Elde)

PO - Parity Odd (Teş Eşlik)

PE - Parity Even (Çift Eşlik)

P - Plus (Pozitif)

M - Minus (Negatif)

**BCD (Binary Coded Decimal: İkili kodlanmış ondalık):** Ondalık tabandaki sayıların, bir grup ikili sayı ile temsil edildiği sayı sistemi.

**Bellek hücresi (Memory cell):** Bir belleğin, bir bitlik bilgi saklanabilen ve okunabilen elemanı.

**Bellek konumu (Memory location):** Bir bilgisayar belleğinde, genelde bir baytlık bilginin saklandığı ve belirli bir adrese sahip saklama yeri.

**Bit (Bit- BInary digiT):** İkili basamak ifadesinden türetilmiş en küçük bilgi birimini ifade eden birim.

**CP/M (Control Program for Microcomputers):** Digital Research firması tarafından geliştirilmiş olan ve birçok kişisel bilgisayarda kullanılan, ancak günümüzde kullanımdan kalkmış bulunan işletim sistemi.

**CRT (Cathode Ray Tube: Katot-ışınlu tüp):** Elektron tabancasından çıkan elektron hüzmelerinin, yatay ve düşey plakalarla saptırılarak ekrana yansıtılması ilkesine göre çalışan vakum tüpü.

**Çevirici (Assembler):** Sembolik komutları, bilgisayarda kullanılabilir komutlara çeviren program.

**Çapraz-çevirici (Cross-assembler):** Belli bir bilgisayar sistemi için yazılmış programları, farklı bir sistemde kullanılabilecek biçimde (örn. 6800'den

8080'e) çeviren çevirici program.

**Çevrebirimi (Peripheral):** Bilgisayarın kendi parçası olmayan, ancak bilgisayara arabirim üzerinden bağlanarak çalışabilen birimlerdir. Örn., yazıcılar, kart okuyucular, teyp birimleri, vb.

**Çoğullayıcı (Multiplexer):** Girişine uygulanan sinyallerden birini, denetim girişlerine göre çıkışa aktaran sayısal birim.

**Denetim yolu (Control Bus):** Bilgisayarı oluşturan birimlerin koordinasyon içinde çalışabilmesi için gereken denetim sinyallerinin iletildiği veya alındığı yol.

**Denetleyici (Controller):** Elektriksel cihazların çalışmasını denetleyen, denetim sinyallerinin alış-verişini yöneten donanım.

**DIL paketi (Dual-In Line package):** Çok kısa aralıklarla iki sıra bağlantı yapılabilmesini sağlayan entegre devre paketi.

**Dinamik RAM (Dynamic RAM):** İçerdiği bilgileri zaman içinde kaybedebilen ve bu nedenle belirli aralıklarla tazelenmesi gereken rastgele-erişimli bellek türü.

**Donanım (Hardware):** Bilgisayarı oluşturan fiziksel bileşenler ve devrelerin tümü.

**Döngü (Loop):** Bir bilgisayar programında, istenen sonuç elde edilinceye kadar tekrarlanan bir dizi komut.

**Durdurma biti (Stop bit):** Asenkron seri iletimde, gönderilen bir baytlık bir bit dizisinin sonunda yer alan ve 1 ya da 2 bit uzunlukta olabilen bit.

**Durum (State):** Bilgisayar içindeki herhangi bir birimin herhangi bir andaki durumu. Örn., wait state: bekleme durumu, vb.

**Düzenleyici (Editor):** Metinlerin yazılabilmesine ve çeşitli biçimlerde düzenlenebilmesine olanak tanıyan program.

**Emülasyon (Emulation):** Bir bilgisayar sisteminin, bir başka bilgisayar sistemi için hazırlanmış veri ve programları kullanabilmesine ve aynı sonuçları alabilmesine olanak tanıyan teknik.

**Erişim Süresi (Access time):** Verilerin bilgisayar belleğinden okunmak üzere çağrılmaları ile okunabilir ve kullanılabilir duruma geçmeleri arasındaki süre.

**Eşlik biti (Parity bit):** Verideki 1 veya 0'ların sayısının tek ya da çift olmasına göre 0 veya 1 değeri olarak veriye eklenen ve bu yolla, asenkron seri iletimde kontrol olanağı sağlayan bit.

**Etiket (Label):** 1. Bir adresin sembolik adı. 2. Bir bilgisayar programındaki bir deyimi veya bilgi elemanını tanımlayan bir ya da birkaç karakter.

**Ferrit-çekirdekli bellek (Ferrite-core memory):** İçinden okumayazma iletkenleri geçen ve ferrit malzemeden yapılmış çok küçük sargılardan oluşan bellek türü. Birkaç özel uygulama dışında artık kullanılmamaktadır.

**Firmware (Bellenim):** Genelde bir ROM'a kaydedilen ve temel makine komutlarından oluşan yazılım; bu yazılımın tutulduğu birim. (Yerleşik program)

**Flip-flop (Flip-flop):** Kararlı iki durumdan birinde bulunan ve bir darbe uygulanmadıkça durum değiştirmeyen elektronik devre.

**Format (Format: Biçim):** Bilgilerin bir tablo, sayfa, dosya ya da mesaj üzerinde önceden belirlenmiş bulunan biçimi, düzeni.

**Gerçek-zamanlı (Real-time):** İlgili fiziksel olayın ortaya çıktığı anda işlem yapabilen bilgisayar sistemi ya da programı.

**Getirme saykılı (Fetch cycle):** Bilgilerin bellekten alınıp mikroişlemciye getirilmesi.

**Hata ayıklama (Debug):** Bir programın içerdiği hataların ayıklanarak giderilmesi.

**Hata ayıklama programı (Debugger):** Programlardaki hataları bularak gidermek üzere hazırlanmış özel yardımcı program.

**İki-kararlılar (Bistables):** Kararlı iki durumu bulunan flip-flop.

**İki-yönlü (Duplex):** Her iki yönde de iletimin mümkün olduğu iletişim sistemi.

**İşkod (Opcode):** Bkz:İşlem kodu

**İşlem kodu (Operation code):** Makinenin belli bir işlemi yapmasını sağlayan ve bir veya birkaç bayt uzunlukta olabilen makine düzeyli komut.

**İşlenen (Operand):**Bilgisayarda bir işleme giren ya da bir işlemin sonucu olarak ortaya çıkan nicelik.

**Kalıcı (Non-volatile):** Uygulanan besleme gerilimi kesildiğinde bile içerdiği bilgileri tutabilen bellek türü.

**Kalıcı-olmayan (Volatile):** Uygulanan besleme gerilimi kesildiğinde içerdiği bilgilerin silineceği bellek türü.

**Karakter üretici (Character generator):** Bir bilgisayarda mevcut bulunan tüm kod ve sembollerin tutulduğu ve üretildiği genelde ROM veya EPROM şeklinde olan birim.

**Kaydedici (Register):** Genelde bilgisayarın sözcük boyuna eşit uzunlukta olan ve çeşitli amaçlarla kullanılan yerel bellek. Örn. A kaydedicisi.

**Kaynak program (Source program):** Bilgisayarda yürütülmeden önce derlenmesi, çevrilmesi ya da yorumlanması gereken program.

**Kesme (Interrupt):** Program akışının belli bir nedenle kesilmesi. Bu durumda, kesmeye neden olan işlem ya da olayla ilgilenir ve denetim, kesme hizmet yordamı adı verilen bir yordama aktarılır.

**Klavye (Keyboard):** Bilgisayara karakter girmek için kullanılan G/Ç birimidir.

**Kod çözücü (Decoder):** Bilgisayarlarda, girişine uygulanan sinyallere bağlı olarak bir ya da daha fazla çıkış veren devre ya da sistem.

**Kodlayıcı (Encoder):** Bilgisayarlarda, her seferinde sadece bir girişin uyarıldığı ve her girişin belli bir çıkış grubu çıkardığı devre ya da sistem.

**Komut (Instruction):** Bilgisayarda belli bir işlemi tanımlayan, bir veya birkaç adresle birlikte ya da adres olmaksızın verilen karakter takımı.

**Komut saykılı (Instruction cycle):** Bilgilerin bel-

lekten alınıp getirilmesi (getirme saykılı) ve yürütülmesinden (yürütme saykılı) oluşan çevrim.

**Komut takımı (Instruction set):** Bir merkezi işlem birimi tarafından yürütülen temel işlemlere karşılık gelen komutlar topluluğu.

**Koşullu atlama (Conditional jump):** Bilgisayar programının komut yürütme sırasının belli bir koşula (örneğin, bir değişkenin değerinin sıfırdan büyük ya da küçük olması koşuluna) bağlı olarak değişmesi.

**Koşulsuz atlama (Unconditional jump):** Bilgisayar programının komut yürütme sırasının, ilgili Atlama komutuna gelindiğinde herhangi bir koşula bağlı olmaksızın değişmesi.

**Kurma/Sıfırlama (Set/Reset):** Bir bitlik bellek alanının içeriğini 1 yapmak, Kurma (Set); sıfır yapmak ise Sıfırlama (Reset) olarak adlandırılır.

**Kütüphane (Library):** Standart bilgisayar programları, modülleri ve yordamlarından oluşan modüller dizisi.

**LED (Light-emitting diode):** Işık Yayan Diyot: İleri öngerilim uygulandığında ışık yayan bir diyot türü.

**Makro-çevirici (Macro-assembler):** Kaynak programın her bir deyimine karşılık makine kodunda birkaç komut içeren yöntem üretmek üzere komut kısaltmalarının kullanıldığı çevirici program.

**Mandal (Latch):** Saat darbeleriyle değil de, giriş işaretinin değişmesiyle konum değiştiren flip-floplardır.

**MİB-Merkezi İşlem Birimi (CPU-Central Processing Unit):** Bir bilgisayarın aritmetik, mantık ve denetim devrelerini içinde bulunduran bölümü.

**MODEM (Modulator+DEModulator):** Asenkron seri haberleşmede, bilgisayarın 0 ve +5 V düzeyleri şeklindeki mantıksal sinyallerini, iletim hatlarından iletebilecek biçimde değişen frekans-taki alternatif sinyallere; alışı da, bu alternatif sinyalleri tekrar mantıksal 0 ve 1 sinyallerine dönüştüren cihaz.

**MOS: Metal-oksit yarı iletken** anlamında kısaltma. MOS teknolojisinde, bir yarı iletken alt katman üzerinde oskit yalıtkan tabaka ve en üstte de bir metal elektrot bulunur. MOS tekniğiyle üretilmiş transistörler MOSFET (MOS-Alan Etkili Transistör) olarak adlandırılır. P-MOS teknolojisinde yalnızca P-tipi, N-MOS teknolojisinde yalnızca N-tipi kanal ve CMOS (Complementary MOS) teknolojisinde ise hem P-tipi ve N-tipi kanal kullanılır.

**NMI (Non-maskable interrupt: Maskelenemez kesme):** Bir programın yürütümünü kesebilen ve Kesmeleri Yetkisizle (DI) komutu ile devre dışı bırakılamayacak öncelikte bulunan kesme türü. Güç kaynağı arızası, maskelenemez kesmeye bir örnektir.

**Onay (Handshaking):** Asenkron seri veri iletiminde, karşılıklı iki birimin iletişim kurmak üzere yaptıkları sinyal alışverişidir.

**Osilatör (Oscillator):** Elemanların değerleri ile belirlenen frekansta salınım yapan devre.

**Paralel-seri dönüştürücüler (Parallel to Serial Converter):** Girişine uygulanan paralel biçimdeki veriyi seri biçime (ardışık bit dizilerine) dönüştüren elektronik devre.

**Polish form (Polish biçimi):** Polonyalı mantıkçı J. Lukasiewicz tarafından geliştirilen bir tekniktir. Bu gösterim biçiminin amacı, cebirsel ifadelerdeki operatörleri, işlenenler arasında değil, işlenenlerin ardından (örn. a+b yerine ab+ gibi) yazabilmektir.

**Port (Port):** Giriş ya da çıkış amacıyla kullanılan fiziksel bağlantı birimi.

**Saat (Clock):** Bilgisayardaki birimlerin eş zamanlı biçimde çalışabilmesini sağlamak üzere bir üreteçten üretilen periyodik sinyaller.

**Saat üretici (Clock generator):** Eşzamanlamayı sağlamak üzere periyodik sinyaller üreten ve genelde yüksek kararlılık sağlayabilmek için kristal içeren elektronik devre.

**Sayısal-analog dönüştürücü (Digital to Analog Converter):** Sürekli değişen analog sinyalleri, sayısal sinyallere dönüştüren elektronik devre.

**Sektör (Sector):** 1.Dairesel veri saklama ortamlarının en küçük adres bölümü. 2.Disk/disket üzerindeki izlerin komşu bölümlerinden her biri. Sektör büyüklükleri genelde 128, 256 veya 512 bayttır.

**Sembolik adres (Symbolic address):** Bir program içinde, bir sözcüğü, bir işlevi ya da bir bilgiyi, bilginin program içindeki yerinden bağımsız olarak tanıtmak üzere kullanılan etiket.

**Senkron veri iletimi (Synchronous data transmission):** Verici ve alıcı cihazların sürekli olarak aynı frekans ve faz ilişkisinde bulundukları iletim.

**Seri-paralel dönüştürücüler (Serial to Parallel Converter):** Girişine uygulanan seri (ardışık bit dizisi) biçimli veriyi paralel biçimdeki veriye dönüştüren elektronik devre.

**Sinyal (Signal):** 1. Bilgi iletebilecek görülür, işitilir ya da başka biçimlerde elektriksel işaret. 2. Bilgisayar içindeki çeşitli işlemleri denetlemek ve/veya zamanlamak için kullanılan elektriksel işaret.

**Simülasyon (Simulation: Benzetim):** Fiziksel sistemler veya olguların, işlev ve özelliklerinin modeller, bilgisayarlar veya diğer donanımlar aracılığıyla oluşturulması.

**Statik RAM (Static RAM):** Uygulanan enerjinin sürdürülmesi halinde, Dinamik RAM'in tersine tazeleme gerektirmeden içerdiği bilgileri tutan rastgele-erişimli bellek türü.

**Sürücü (Driver):** Birden fazla sayıda TTL sinyalinin bulunması halinde bir ara hattaki sinyalleri yeniden biçimlendirmede kullanılan yükselteç devresi.

**Tampon (Buffer):** Sürücü bir devre ile sürülen devreler arasındaki sinyal alışverişini düzenleyen ara devre.

**Tekilleyici (Demultiplexer):** Bir çoğullayıcı (multiplexer) tarafından birleştirilen iki veya daha çok sinyali birbirinden ayıran cihaz.

**Tek-yönlü (Simplex):** Her seferinde sadece tek

bir yönde bilgi göndermeye izin veren haberleşme türü.

**Teletayp (Teletype):** Bir klavye ve yazıcı üniteden oluşan ve konsol veya terminal olarak kullanılan giriş/çıkış birimi. (Bu terimin orijinali, Teletype Co. firmasının ürettiği teleks cihazlarına vermiş olduğu isimdir).

**Terminal (Terminal):** 1. Bağlantı ucu, uç. 2. Bilgilerin bir iletim ağına girdikleri veya ağdan çıktıkları nokta; bu amaçla kullanılan birim.

**Tersleyici (Inverter):** Girişin tersini çıkış veren mantık devresi çıkış girişin her zaman DEĞİL'i olur. Örnek. Giriş 1 ise çıkışı 0 olur. Giriş 0 ise çıkışı 1 olur.

**TTL-Transistör Transistör Mantık (Transistor-Transistor Logic):** Yüksek hızda entegre mantık devreleri ailesi. Genelde girişleri çok emittörlü, çıkışları ise push-pull'dur.

**UART:** Asenkron seri veri haberleşmesinde kullanılan genel-amaçlı alıcı-verici entegre devresi (Universal Asynchronous Receiver-Transmitter'in kısaltması)

**USART:** Seri veri haberleşmesinde kullanılan, hem senkron (eşzamanlı) ve hem de asenkron çalışabilen alıcı-verici entegre devresi. (Universal Synchronous Asynchronous Receiver-Transmitter).

**Üç-durumlu (Three-state):** Elektronikte, aynı hatta bağlı paralel birimlerin, giriş empedanslarının teorik olarak sonsuz olduğu, bu nedenle devreden akım çekmedikleri, devreye akım vermedikleri üçüncü durumu göstermek üzere kullanılan terim.

**VDU (Video Display Unit: Görsel Görüntüleme Birimi):** Bilgisayar ekranı anlamında İngilizlerin kullandığı bir kısaltma. Eş anl. CRT-Display.

**Veri yolu (Data Bus):** Verilerin, MİB ile bellek ya da çevre birimleri arasında alınıp verilmesinde kullanılan yol.

**Yardımcı program (Utility):** Bir bilgisayarın çalışmasına yardımcı olmak üzere, kullanıcıya bilgisayarın işletim sistemi veya uygulama programıyla birlikte sunulan ve kullanıcının sık sık ihtiyaç duyacağı hizmetlerin karşılanması için parametrelerce yönetilen yardımcı hizmet programı.

**Yarıiletken (Semiconductor):** Mutlak sıcaklıkta akım geçirmeyen, ancak enerji (ışık, gerilim) uygulandığında, içerdiği valans elektronların yörüngelerinden koparak serbest hale gelmesiyle iletken duruma geçen malzeme, bu malzemeden yapılmış elektronik eleman.

**Yarım-bayt (nibble):** Bit gruplarının dörtlü gruplar halinde gösterim biçimi.

**Yayınım (Propagation):** Bir sinyalin devre ya da devreler dizisi içinden geçerek ilerlemesi.

**Yazıcı (Printer):** Gönderilen sayıların, kodların veya sembollerin basılı kopyasını çıkaran çıkış birimi. Basım tekniğine bağlı olarak satır, nokta-basırlı, mürekkep-püskürtmeli, tambur, vb. adlar alırlar.

**Yazılım (Software):** Bilgisayarın kaynaklarını kullanmak için gereken programlar, yordamlar, yardımcı programlar ve kütüphaneler gibi donanımın dışında kalan bölümü.

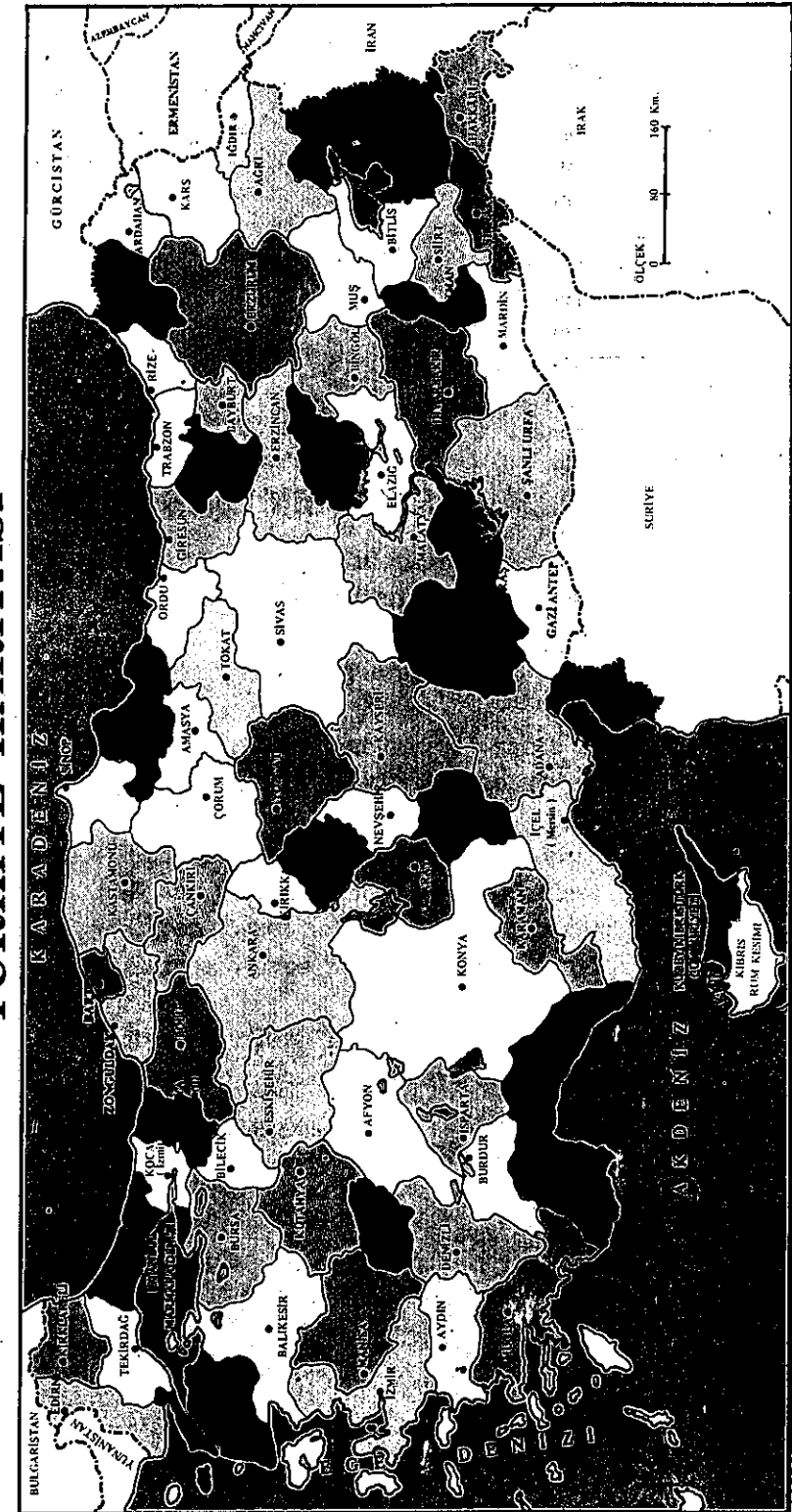
**Yığın (Stack):** Verileri geçici bir süreyle -örneğin, çıkışta geriye almak üzere bir altıyordama girerken- saklandığı ve bilgiler, belirli bir sırada giriş-çıkış yapacak şekilde düzenlenmiş bellek alanı.

**Yonga (Chip):** Milimetrik boyutlarda yüzbinlerce elektronik devre elemanından oluşmuş yarıiletken devre elemanı.

**Yordam (Routine):** Bir bilgisayar programı içinde yer alan ve işlevi, verilen değişkenlerle hep aynı işlemi gerçekleştirmek olan, ihtiyaç duyulduğunda ana program tarafından çağrılan program parçası.

**Yedi-parçalı gösterge (Seven-segmented display):** 0'dan 9'a kadar ondalık sayıları göstermek üzere çizgi biçimli yedi LED biriminden oluşan sayısal gösterge.

## TÜRKİYE HARİTASI



## ÖĞRETMEN MARŞI

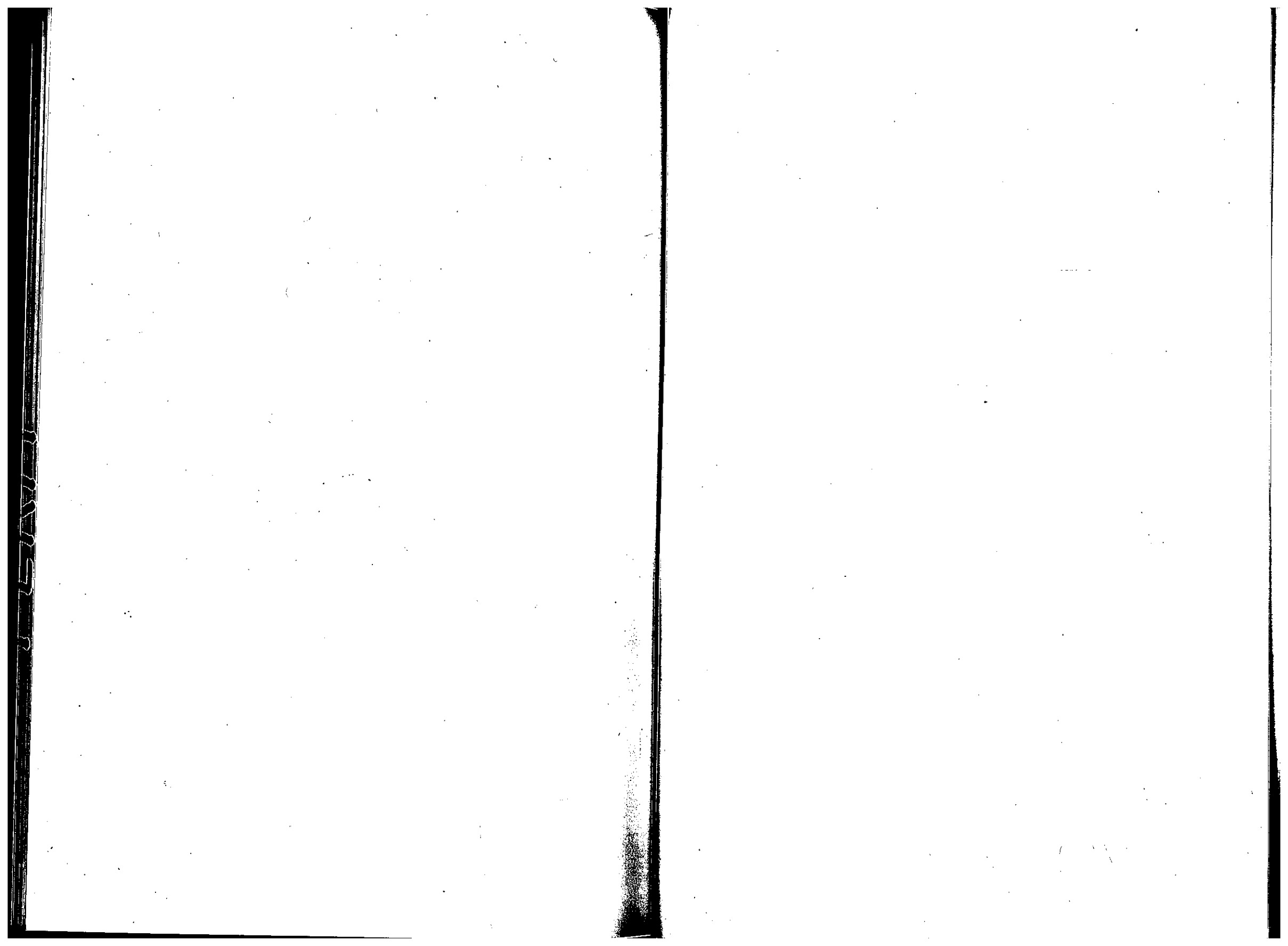
Alnımızda bilgilerden bir çelenk,  
Nura doğru can atan Türk genciyiz.  
Yer yüzünde yoktur, olmaz Türk'e denk;  
Korku bilmez soyumuz.

Şanlı yurdum, her bucağın şanla dolsun;  
Yurdum, seni yüceltmeye antlar olsun.

Candan açtık cehle karşı bir savaş,  
Ey bu yolda and içen genç arkadaşı!  
Öğren, öğret hakkı halka, gürle coş;  
Durma durma koş.

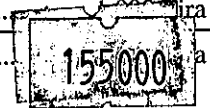
Şanlı yurdum, her bucağın şanla dolsun;  
Yurdum, seni yüceltmeye antlar olsun.

İsmail Hikmet ERTAYLAN



Satış fiyatı .....: 109.000 Lira  
KDV .....

KDV'li SATIŞ FİYATI .....



**TOPTAN SATIŞ**

İstanbul Devlet Kitapları Müdürlüğü, Adana, Ankara, Burdur, Elazığ,  
Erzurum, İzmir, Samsun, Sivas, Trabzon , Van ve Zonguldak  
Bölge Şeflikleri.

**PERAKENDE SATIŞ**

Millî Eğitim Yayınevleri ve Bakanlık yayınları satıcısı kitapçılar.