ENDÜSTRİYEL OKULLAR İÇİN

Mikroelektronik Sistemler

Kitap II

Fred Halsall

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

Millî Eğitim Bakanlığı Yayınları ..... : 2695
Yardımcı ve Kaynak Kitaplar Dizisi ..... : 56

ISBN 975 - 11 - 0869 - 1

Hükûmetimiz ile Dünya Bankası arasında imzalanan Endüstriyel Okullar Projesi çerçevesinde hazırlanan "Mikroelektronik Sistemler Kitap II" Millî Eğitim Bakanlığı, Talim ve Terbiye Kurulu Başkanlığının 05/05/1994 gün ve 3697 sayılı kararı ile kaynak kitap olarak uygun bulunmuş ve 10.000 adet bastırılmıştır.

Çeviri - Dizgi - Mizampaj : Üniversal Dil Hizmetleri ve Yayıncılık A.Ş.
Çevirmen : Bilg. Müh. Tanol TÜRKOĞLU
Editör : Yrd. Doç. Ahmet Coşkun SÖNMEZ
Baskı Hazırlık - Baskı - Cilt : Evren Ofset Basım Sanayii ve Ticaret A.Ş.

İSTİKLÂL MARŞI

Korkma, sönmez bu şafaklarda yüzen al sancak;
Sönmeden yurdumun üstünde tüten en son ocak.
O benim milletimin yıldızıdır, parlayacak,
O benimdir, o benim milletimindir ancak.

Çatma, kurban olayım, çehreni ey nazlı hilâl!
Kahraman ırkıma bir gül! Ne bu şiddet, bu celâl?
Sana olmaz dökülen kanlarımız sonra helâl...
Hakkıdır, Hakk'a tapan, milletimin istiklâl!

Ben ezelden beridir hür yaşadım, hür yaşarım.
Hangi çılgın bana zincir vuracakmış? Şaşarım!
Kükremiş sel gibiyim, bendimi çiğner, aşarım.
Yırtarım dağları, enginlere sığmam, taşarım.

Garbın âfâkını sarmışsa çelik zırhlı duvar,
Benim iman dolu göğsüm gibi serhaddim var.
Ulusun, korkma! Nasıl böyle bir imanı boğar,
"Medeniyet!" dediğin tek dişi kalmış canavar?

Arkadaş! Yurduma alçakları uğratma, sakın.
Siper et gövdeni, dursun bu hayâsızca akın.
Doğacaktır sana va'dettiği günler Hakk'ın...
Kim bilir, belki yarın, belki yarından da yakın.

Bastığın yerleri "toprak!"diyerek geçme, tanı :
Düşün altındaki binlerce kefensiz yatanı.
Sen şehit oğlusun, incitme, yazıktır, atanı :
Verme, dünyaları alsan da, bu cennet vatanı.

Kim bu cennet vatanın uğruna olmaz ki fedâ?
Şühedâ fışkıracak toprağı sıksan, şühedâ!
Cânı, cânânı, bütün varımı alsın da Huda,
Etmesin tek vatanımdan beni dünyada cüdâ.

Ruhumun senden, İlâhî, şudur ancak emeli :
Değmesin mabedimin göğsüne nâmahrem eli.
Bu ezanlar-ki şahâdetleri dinin temeli-
Ebedî yurdumun üstünde benim inlemeli.

O zaman vecd ile bin secde eder-varsa-taşım,
Her cerîhamdan, İlâhî; boşanıp kanlı yaşım,
Fışkırır ruh-ı mücerred gibi yerden na'şım;
O zaman yükselerek arşa değer belki başım.

Dalgalan sen de şafaklar gibi ey şanlı hilâl!
Olsun artık dökülen kanlarımın hepsi helâl.
Ebediyen sana yok, ırkıma yok izmihlâl:
Hakkıdır, hür yaşamış, bayrağının hürriyet;
Hakkıdır, Hakk'a tapan, milletimin istiklâl!

Mehmet Âkif ERSOY

ATATÜRK'ÜN
GENÇLİĞE HİTABESİ

Ey Türk gençliği! Birinci vazifen, Türk istiklâlini, Türk cumhuriyetini, ilelebet, muhafaza ve müdafaa etmektir.

Mevcudiyetinin ve istikbalinin yegâne temeli budur. Bu temel, senin, en kıymetli hazinendir. İstikbalde dahi, seni, bu hazineden, mahrum etmek isteyecek, dahilî ve haricî, bedhahların olacaktır. Bir gün, istiklâl ve cumhuriyeti müdafaa mecburiyetine düşersen, vazifeye atılmak için, içinde bulunacağın vaziyetin imkân ve şeraitini düşünmeyeceksin! Bu imkân ve şerait, çok nâmüsait bir mahiyette tezahür edebilir. İstiklâl ve cumhuriyetine kastedecek düşmanlar, bütün dünyada emsali görülmemiş bir galibiyetin mümessili olabilirler. Cebren ve hile ile aziz vatanın, bütün kaleleri zapt edilmiş, bütün tersanelerine girilmiş, bütün orduları dağıtılmış ve memleketin her köşesi bilfiil işgal edilmiş olabilir. Bütün bu şeraitten daha elîm ve daha vahim olmak üzere, memleketin dahilinde, iktidara sahip olanlar gaflet ve dalâlet ve hattâ hıyanet içinde bulunabilirler. Hattâ bu iktidar sahipleri şahsî menfaatlerini, müstevlilerin siyasî emelleriyle tevhid edebilirler. Millet, fakr u zaruret içinde harap ve bîtap düşmüş olabilir.

Ey Türk istikbalinin evlâdı! İşte, bu ahval ve şerait içinde dahi, vazifen; Türk istiklâl ve cumhuriyetini kurtarmaktır! Muhtaç olduğun kudret, damarlarındaki asîl kanda, mevcuttur!

Bilgi çağına girerken bütün ülkelerin üzerinde önemle durdukları ve giderek daha fazla kaynak ayırdıkları sektör eğitimdir. Bilim ve teknolojideki gelişmelere paralel olarak eğitimde kaliteyi yükseltmek, gençlerimize ileri sanayi toplumunun gerektirdiği bilgi, beceri ve davranışları kazandırmak Millî Eğitimimizin temel amaçlarından biridir.

Ülkemizde; ekonomik, sosyal ve kültürel alanlarda olduğu gibi, sanayi alanında da önemli gelişmeler olmaktadır. Nitelikli insangücü ihtiyacının giderek arttığı ülkemizde meslekî ve teknik eğitim büyük önem kazanmaktadır.

Bu alandaki ihtiyacı karşılayabilmek için; çağdaş bilim ve teknolojik metodları bilen, yorumlayan, kullanan, geliştiren ve alanındaki yeniliklere uyum sağlayan, üretken teknik insangücünün yetiştirilmesi gerekmektedir. Bu konuda, teknik öğretim kurumlarımıza büyük iş düşmektedir.

Bu kurumlarımızdaki öğrencilerin iyi yetişmeleri için devletimiz her türlü desteği sağlamakta ve Hükûmetimiz ile Dünya Bankası arasında imzalanan İkraz Anlaşmasıyla yürütülen Endüstriyel Okullar Projesiyle bu okullarımız, çağdaş eğitim imkanlarına kavuşturulmaktadır. Bu okullarımızda çeşitli meslek alanlarında ihtiyaç duyulan 42 adet yabancı teknik ders kitabının tercüme haklarının satın alınması, basım ve dağıtımlarının yapılarak öğrenci ve öğretmenlerimizin istifadesine sunulması, bu proje kapsamında yürütülen faaliyetlerden biridir.

Eğitim ve kültür düzeyleri yüksek, gelişen teknolojiye uyum sağlayabilen toplumlar, geleceğin dünyasının şekillenmesinde önemli rol oynayacaklardır.

Bu ve benzeri çalışmaların ülkemiz için yararlı olmasını diliyorum.

Nevzat AYAZ
Millî Eğitim Bakanı

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

İçindekiler

Yazarın Ön Sözü

Bu kitap, Hutchinson tarafından Teknik Eğitim Kurulu (TEK) adına yayımlanan mikroelektronik/mikroişlemci serisindeki kitaplardan biridir. Bu serideki kitaplar, Teknik Eğitim Kurulu programlarıyla bağlantılı kitaplarla kullanılmak üzere hazırlanmıştır.

1978 Haziranında, İngiltere Başbakanı, mikroişlemcilerin kullanıma girmesinin belirli sanayi dallarındaki istihdam modeli üzerinde yaratacağı etkiden duyduğu kaygıyı dile getirmiştir. Bundan, Sanayi Bakanlığı ve Ulusal Girişim Dairesi aracılığıyla, mikroişlemci teknolojisini kullanmayı ve geliştirmeyi özendirecek bir girişim doğmuştur.

Mikroelektronik araç ve gereçlerin, araştırılması, geliştirilmesi ve üretimi için personel eğitimi ve öğretimi ile bunların diğer sanayi dallarına uygulanması da, böyle bir geliştirme programının önemli bir yönü olarak görülmüştür. 1979'da, teknik eğitim ünitelerinin (ünite, bir öğrenci tarafından ulaşılacak hedeflerin tanımıdır) ve bunlara bağlı eğitim paketlerinin geliştirilmesi için Teknik Eğitim Kurulu tarafından bir proje oluşturulmuştur. Bu proje için gereken para Sanayi Bakanlığı tarafından sağlanmakta ve proje bakanlık namına National Computing Centre Ltd. tarafından yönetilmektedir.
TEK, mikroelektronik üreticileri ve kullanıcıları olarak sanayicileri ve eğitimcileri içeren bir komite kurmuş; buna ek olarak, çok sayıda kişi ve kuruluşa danışılmıştır. Mikroelektronik devrelerle ilgili tasarım, üretim ve servis alanlarında çalışan teknikerler ve teknik mühendisler için programa ilişkin kitaplar geliştirilmiştir. Yazılan beş kitap şunlardır:

Mikroelektronik Sistemler	Kitap 1
Mikroelektronik Sistemler	Kitap 2
Mikroelektronik Sistemler	Kitap 3
Mikroişlemci - Tabanlı Sistemler	Kitap 4
Mikroişlemci - Tabanlı Sistemler	Kitap 5

Aynı zamanda, mikroelektronik devrelerin uygulama alanları ve potansiyelleri ile ilgili genel bir bilgi sahibi olmak isteyen teknikerler için de kitaplar hazırlanmıştır:

Mikroişlemci Değeri	Kitap 3
Mikroişlemci İlkeleri	Kitap 4

Bu aşamayı, öğrenim paketlerinin üç yazım ekibi tarafından geliştirilmesi izlemiştir. Bu ekiplerde temel sorumluluğu üstlenen kişiler şunlardır:

Mikroelektronik Sistemler 1, 2, 3	--	P.Cooke
Mikroişlemci-Tabanlı Sistemler 4	--	A.Potton
Mikroişlemci-Tabanlı Sistemler 5	--	M.Morse
Mikroişlemcilerin Değerlendirilmesi 3	--	G.Martin
Mikroişlemci Temelleri 4	--	G.Martin

Kitapların temel özelliklerinin belirlenmesi aşamasında N.Bonnett proje sorumluluğunu, R.Bertie de onun yardımcılığını yürütmüştür. Bay Bonnett, kitabın yazımı aşamasında da danışman olarak görev yapmaya devam etmiştir. Projenin yönetici W.Bolton, yardımcısı da K.Snape'dir.

Eşim Jan'e,

Kitabın yazılmasındaki önemli çabaları ve bu kitabın hazırlanması esnasındaki teşvik ve daimi desteği nedeniyle teşekkür ederim.

DAVID WOOLLONS

Kendi Kendine Öğrenim

Kendi kendine öğrenime yardımcı olmak üzere, her bölüm içinde sorulara yer verilmiştir. Sorular, her bölümün sonunda bulunmakta ve bunlara ilişkin referanslar da, kendi kendine öğrenimde en uygun konumu gösterecek biçimde, ilgili metnin kenarında (örneğin S1.2) yer almaktadır. Her sorunun yanıtı kitabın sonunda verilmiştir.
Bu nedenle, bu dizideki kitaplar, hem kendi kendine öğrenimde hem de sınıf ortamında öğretim durumunda kullanılmak amacıyla geliştirilmiştir.

Giriş

Bu kitap, Mikroelektronik Sistemler U79/603 adlı TEK ünitesinde belirtilen hedeflere ulaşmak için yazılmıştır. Kitabın ana amaçları; ilk olarak, tipik bir mikroişlemci mimarisinin ve bu tür birimlerle birlikte kullanılan değişik komut tiplerinin tanımını yapmak, ikinci olarak da mikrobilgisayarın çeşitli sistem bileşenlerini tanımlayarak, bir mikrobilgisayarın ek arabirim devreleri aracılığıyla çeşitli çevrebirimleriyle birlikte nasıl kullanılabileceğini göstermektir.

Örnek olarak, halen piyasada mevcut bir mikroişlemci seçilmiştir. Bu yolla, kullanılan mikroişlemci komutlarının normal olarak piyasada mevcut diğer mikroişlemcilerin komutlarının tipik bir örneğini oluşturması sağlanmış ve bir mikroişlemciden diğerine değişebilen küçük farklılıkları tanımlama zorunluluğunun getireceği olası karışıklıklar önlenmiştir.

Bu kitap, burada verilen farklı komut türlerini ve programlama ilkelerini anlaşılır kılmak için çok çeşitli program örnekleri içermektedir. Bu programların, hiç değiştirilmeden ya da üzerinde çok az değişiklik yapılarak, uygun bir üreticinin tek sistem kartlı prototipleme sistemi üzerinde çalıştırılabilmesi ve böylece uygun bir destekleme laboratuar programlarına temel teşkil etmesi hedeflenmiştir.

Teşekkür

Sussex Üniversitesi'nden meslektaşım Phil Cooke'a, dizinin koordinatörü, Norman Bonnett'e müsveddeleri eleştirici gözle okudukları ve yararlı öneriler getirdikleri için teşekkür etmek isterim. Ayrıca, müsveddeleri daktilo eden Christine Thornton-Clough'a ve metne son halini verirken getirdiği önerilerden dolayı Peter Hill'e de teşekkür ederim.

FRED HALSALL

1.Bölüm : **Sayısal bilgisayarların temelleri**

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde:*

1 *Yarı-iletken teknolojisinde transistörün bulunmasından beri meydana gelen tarihsel gelişmeleri değerlendirebilmeli,*
2 *Saklı-program kavramının anlamını ve getirdiği avantajları anlayabilmeli,*
3 *Sayısal bir bilgisayarın temel çalışma modunu ve programlanabilir cihazların önemini anlayabilmeli,*
4 *Sayısal bir bilgisayarda bütün bilgilerin ikili-kodlanmış biçimde saklandığını anlayabilmeli,*
5 *Mikrobilgisayar sistemlerinde kullanılan değişik tipteki arabirim devrelerinden bazılarını açıklayabilmeli,*
6 *Ondalık bir sayıyı ikili biçimdeki eşdeğerine (veya tersine) çevirebilmeli,*
7 *Sekizli ve onaltılı sayıları ikili biçimdeki eşdeğerine çevirebilmeli,*
8 *Sekizli ve onaltılı bir sayıyı ikili biçimdeki eşdeğerine çevirebilmeli,*
9 *Tipik bir bilgisayarın blok diyagramını çizip her bir bileşenin işlevini açıklayabilmeli,*
10 *Bir mikrobilgisayarın saklı bir programı çalıştırmak için belleği ile nasıl iletişim kurduğunu açıklayabilmelisiniz.*

1.1 Tarihsel gelişim

Sayısal bilgisayarlar, Charles Babbage ve onun Analitik Motor'u zamanından başlayıp günümüze kadar gelen önemli teknolojik evrimin vardığı en son noktadır. Makinesi mekanik olduğu ve 1833'te ölene kadar tatmin edici olarak çalıştırılamadığı halde Babbage'in, genel olarak, sayısal bilgisayarın mucidi olduğu kabul edilir.

Bu tarihten sonraki gelişme yavaştır ve lambalar kullanılarak ilk elektronik sayısal bilgisayarın yapılması, II.Dünya Savaşı'nın bitişinden önce değildir. Bu makinelerin güvenilirlikleri fazla değildi. Ancak, 1947'de transistörün bulunuşu, bugün bilinen genel-amaçlı elektronik sayısal bilgisayarların doğuşunun müjdecisi oldu.

Transistör, sayısal bilgisayarların yapıldığı tüm elektronik devrelerin temelini oluşturur. 1947'de bulunuşundan beri transistörlerin yapımında kullanılan yarı-iletken teknolojisi çok hızlı bir evrim geçirmiştir. Örneğin, insanların yaklaşık bin transistörden oluşan elektronik bir devreyle yapılmış kol saati kullanıyor olması artık sıradan bir olaydır. Aynı şekilde, birçok kişi artık programlanabilir küçük hesap ma-

kineleri (sayısal bilgisayarların küçük bir modeli) kullanmaktadır. Bu makineler de birkaç bin transistörden oluşan tek bir entegre devreden yapılmışlardır.

Yarı-iletken teknolojisi, 10.000 transistörden oluşan tek bir entegre devre ile güçlü bir bilgisayar işlem biriminin yapılabilmesiyle 1970'lerin başında bir gelişme düzeyine ulaşmıştır. Ek bazı entegre devrelerle güçlü bir genel-amaçlı sayısal hesaplama sistemini oluşturan bu birim, *mikroişlemci* olarak bilinir.

Yarı-iletken teknolojisindeki gelişmeler günümüzde de sürmektedir. Bu nedenle, bugünkü teknoloji bir entegre devrede 100.000 transistör gibi pratik bir sınır koyduğu halde, bir milyon transistör içerecek olası birimler konusunda birçok spekülasyon vardır. Artık spekülasyon, böyle bir cihazın mümkün olup olmadığı üzerine değil, ne zaman mevcut hale geleceği üzerinedir. Tablo 1.1'de, yarı iletken teknolojisinin tarihsel gelişimi; Şekil 1.1'de de 20.000 transistörden oluşan tipik bir mikroişlemcinin mikroskoptan alınmış fotoğrafı görülmektedir.

1.2 Saklı-program kavramı

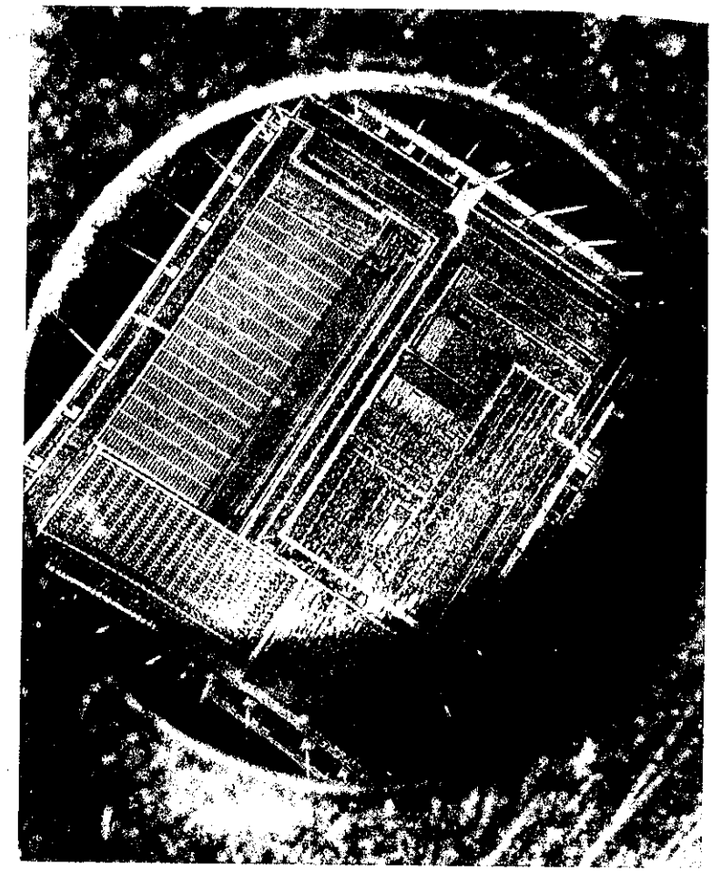
Basit bir elektronik hesap makinesi, sayısal bilgisayarların yaptığı aritmetik işlemlerin aynısını yapabiliyor olsa da hesap makineleri, kullanan kişinin her işlemi gerekli sırada girmesi esasına göre çalışmaktadır. Oysa sayısal bir bilgisayar, insan müdahalesine gerek duymaksızın milyonlarca işlemi gerçekleştirebilir. Bunu yapmak mümkündür, çünkü bilgisayardan istenen işlerin gerçekleştirilebilmesi için gereken işlem dizisi bilgisayarın içinde saklanabilmektedir.

Böylece bilgisayar bir işlemi yerine getirip sonrakine geçme işini, bir kişinin bu işlemleri tuşlama hızında değil, imal edildiği teknoloji düzeyinin belirlediği hızda yapar.

İstenilen bir işi yerine getiren işlemlerin listesine *program* denir. Programlar bilgisayarın içinde saklandıkları için de tüm sayısal bilgisayarlar, *saklı - program kavramını* kullanarak işlem yapar denir.

Tablo 1.1 *Bilgisayarların ve bilgisayar teknolojisinin tarihsel gelişimi*

Gelişme	*Tarih*
Babbage'in Analitik Motor'u	1833
Vakumlu lambanın bulunuşu	1910
İlk elektronik sayısal bilgisayar	1946
Transistörün bulunuşu	1947
İlk transistörlü sayısal bilgisayar	1960
Entegre devrenin bulunuşu (10 transistörlü)	1963
İlk mikroişlemci (10.000 transistörlü)	1970
100.000 transistörlü entegre devreler	1981



Şekil 1.1 Bir mikroişlemcinin mikroskoptan alınmış görüntüsü. (Intel Corporation'ın *izniyle*)

Saklı-program kavramının iki önemli avantajı vardır. Birincisi; daha önce de belirtildiği gibi, işlem sırası bir kez bilgisayara girilip saklandıktan sonra, hesaplamanın tümüyle yapılabilmesi için gereken zamanın kullanıcı kişiye bağlı olmamasıdır. Bu, bilgisayarın insan müdahalesine gerek duymadan bir işlemi bitirip, bir sonrakine geçmesi demektir. İkincisi de, işlem sırası bilgisayarın içinde saklı olduğundan, işlem sırası doğrultusunda yürütülen işlemler, yeni veriler için bilgisayara sadece bu veriler girilerek tekrar tekrar yapılabilmektedir.

1.3 Çalışma modu

Sayısal bilgisayarlar, geniş bir uygulama alanında kullanılabilecek çok esnek ve genel amaçlı makinelerdir. Bu, tasarımının doğasında bulunan genel amaçlılığın

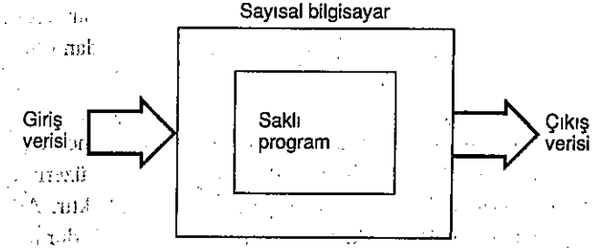
doğal bir sonuçudur. Bu, bir üretici tarafından yapılmış aynı bilgisayarların çok çeşitli uygulama alanlarına uygulanabileceği anlamına gelir. Çünkü sayısal bilgisayarlar, yapıldıktan *sonra*, özel uygulama türlerinin gereklerinin çözülmesine uygulanabilecek biçimde tasarlanmıştır.

Herhangi bir üretici tarafından yapılmış olan bir bilgisayar, çok değişik türden işlemleri yapabilen bir cihazdır. Bu işlemler; verileri (işlenecek verileri) bilgisayara girmek için gereken işlemleri, veri işlemek için gereken işlemleri (gerekli hesaplamaların yapılması), bilgisayardan veri çıkışı (sonuçlar) için gereken işlemleri içerir. İşte bilgisayar, kullanıcı tarafından, istenen görevi yürütmek üzere gerekli işlemleri (mevcut işlem listesinden) seçen, bu işlemleri uygun bir liste halinde sıraya koyan ve böylece programı oluşturan, ve bu programı (yani seçilen işlemler listesini) bilgisayarda saklayan belirli bir uygulama için kullanılır. Bu program daha sonra istenen görevi gerçekleştirmek için *çalıştırılır*. Aynı bilgisayar farklı işlem gruplarının seçilmesiyle (yani saklı programın değiştirilmesiyle) farklı uygulamaları yerine getirir. Bu, Şekil 1.2'de gösterilmiştir.

Sayısal bir bilgisayarın temel çalışma modunu daha iyi değerlendirebilmek için, modern bir elektronik hesap makinesinin çalışma modunu incelemekte yarar vardır.

Basit bir elektronik hesap makinesinde standart bir işlem grubu (toplama,çıkarma,çarpma,bölme gibi) sayılar üzerinde uygulanır. Böyle bir makinede belli bir işlem, istenen hesaplamanın yapılabilmesi için gereken işlem sırasının kullanan kişi tarafından tuşlara basılarak girilmesiyle gerçekleştirilir.

Bir hesap makinesi kullanarak belirli bir işlemin tamamlanması için gereken toplam süre, kullanan kişinin hızına bağlıdır. Bu yüzden bazı ileri elektronik hesap makinelerinde küçük bir *bellek* bulunur ve bu yüzden de *programlanabilir* adını alırlar. Yani kullanıcı istediği hesaplama işlemini gerçekleştirecek işlem sırasını (gerekli olan programı) hesap makinesinin belleğine önceden yükler.



Şekil 1.2 : Saklı program kavramı

Bu yüzden, programlanabilir bir hesap makinesi özel-amaçlı sayısal bir bilgisayar olarak kabul edilebilir. Klavyeden girilen veriler üzerinde belirli sayısal hesaplamaların yapacak şekilde düzenlendiği ve çıkışı sayısal bir ekranda veri olarak görüntülediğinden dolayı özel-amaçlıdır. Bununla birlikte, programlanabilir hesap makinelerinin çalışma modu, genel amaçlı sayısal bilgisayarlarla aynıdır. Çünkü her ikisi de saklı-program kavramını kullanarak çalışırlar.

Belirtildiği üzere, programın bilgisayar içinde saklanmış olması kavramı çok önemli bir kavramdır. Çünkü bu yolla bilgisayar, kendi hızında ve kullanıcıdan bağımsız olarak çalışabilmektedir. Programı oluşturan işlem listesi bilgisayarın içinde saklandıktan sonra bilgisayar, işlem adımlarını saklama biriminden sırasıyla teker teker almak (okumak), bunların kodunu çözmek (yorumlamak) ve ardından da belirli bir işlevi yerine getirmek (yürütmek) suretiyle çalışır.

Bilgisayar tarafından yürütülen temel komutlar *makine komutları* olarak bilinirler ve genelde makine komutları, hem işlemin tipini ve hem de üzerinde işlem yapılacak verinin kaynağını belirler. Bu nedenle, örneğin veri değerlerinin bilgisayara girişini sağlayacak komutlar, bu verileri işleyecek (aritmetik veya diğer hesaplamalar yapan) komutlar, sonuçların bilgisayardan çıkışını sağlayacak çeşitli komutlar vardır. Sayısal bir bilgisayarda bulunan değişik türdeki komutlar bundan sonraki bölümlerde incelenecektir.

rulara ilişkin
nsözdeki nota bakın

S 1.1

1.4 Sayısal devreler

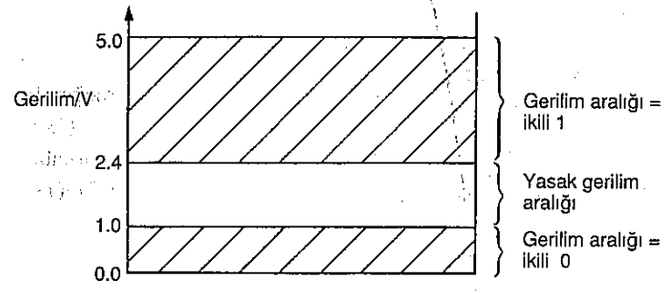
Bir bilgisayar, işlemleri (toplama ve çıkarma gibi) kesin verilerle yapar. Fakat bilgisayarlar da diğer elektronik devreler gibi değeri *tam olarak* bilinmeyen bileşenlerden yapılmıştır.

Her elektronik elemanın, -bu, entegre bir devre içindeki bir bileşen olsa bile- tasarlanmış olduğu bir nominal değeri vardır. Pratikteki değeri ise elemanla ilgili belirli üretim koşullarına bağlıdır ve %5'lik tipik bir tolerans düzeyi söz konusudur. Peki nasıl olur da bu tip belki de binlerce parçadan oluşan bir devre kesin işlemler yapabilmektedir ?

Bu sorunun cevabı modern elektronik için çok önemlidir. Çözüm, bir birim bilgiyi göstermek için, örneğin on farklı mutlak gerilim üzerinde değil de, yalnızca iki gerilim düzeyinde işlem yapan bir devre kullanmaktır. Açıkça ifade etmek gerekirse eğer devre yalnızca iki farklı düzeyi ayırt etmek durumundaysa her düzeye ilişkin gerilim; doğru düzeyi ve dolayısıyla kullanılacak olan sembolü göstermek üzere geniş bir aralığa sahip olur.

İki düzey veya durum kullanarak bilgiyi gösteren sisteme *ikili (iki durumlu) sistem* denir ve kullanılan bu iki sembol 1 ve 0 olarak yazılır. Bu temel bilgi birimine ikili basamak (digit) veya *bit* ve ikili veriler üzerinde işlem yapan bir devreye de *sayısal devre* denir.

İki gerilim düzeyi kullanarak bir ikili basamağı gösteren en yaygın biçim, Şekil 1.3'te gösterilmektedir. Şekilden, eğer devredeki gerilim 2.4 ile 5.0 V aralığındaysa ikili basamak 1'in ifade edildiği, eğer gerilim 0 V ile 1.0 V aralığındaysa ikili basamak 0'ın ifade edildiği anlaşılmaktadır. Bir devrenin bu düzeyleri kullanarak kesin işlem yapabilmesi için gereken tek koşul, gerilimin ya 2.4 V'tan büyük ya da 1.0 V'tan küçük olmasıdır. Yani 1.0-2.4 V aralığında gerilim üretmemelidir ki bu pratikte iki düzey arasında güvenilir bir *aralık* sağlamış olsun.



Şekil 1.3 Sayısal bir devrede ikili bir basamağın gösterimi

Eğer bir bilgisayar bu devrelerden yalnızca birini kullansaydı, tabii ki bir işe yaramayacaktı. Öte yandan, benzer devrelerden birkaçı bir grup ikili biti göstermek için kullanılırsa, neredeyse sınırsız sayıda bilginin uygun biçimde gösterilmesi mümkün olacaktır. Sayısal devrelerin bu şekilde gruplanması bir sayının ikili sistemde gösterilmesi olanağını verir, bu da verinin sayısal bir bilgisayarda işlenmesinde uygun ve önemli bir yaklaşım sağlar.

1.5 İkili sistem

Sayısal bir bilgisayarda tüm bilgiler ikili kodlanmış biçimde saklandığı ve işlendiği için, bir bilgisayarın çalışmasını ve yapısını incelemeden önce ikili sistemi anlamakta yarar vardır.

Önce ondalık sistemde tipik bir sayının yapısını dikkate alalım. Ondalık sistemde,

on tane sembol veya 0'dan 9'a kadar olan sayı basamakları kullanılır. Bu sistemde bir sayıyı yazarken belirli bir basamağın değeri, ondalık noktasının bulunduğu konuma göre belirlenir. Örneğin (186324) ondalık sayısı, $(1x10^5) + (8x10^4) + (6x10^3) + (3x10^2) + (2x10^1) + (4x10^0)$ şeklinde yorumlanır. Her bir basamak, 0-9 arasındaki on sembolden biridir ve bu basamak, ondalık noktasına göre bulunduğu konuma bağlı olarak 10'un üssü ile çarpılır.

Ondalık sayıların *tabanı* 10'dur. Çünkü her basamak bir öncekinden 10 çarpanıyla ayrılmaktadır. Başka birçok sayı sistemi kullanmak mümkündür; fakat taban olarak iki seçilirse sistem, sayıları göstermek için yalnızca iki sembole gerek duyacaktır. Basamaklar birbirinden ikinin katlarıyla ayrılırlar ve bu nedenle bu sistem ikili sayı sistemi olarak bilinir.

İkiliden Ondalığa Dönüştürme

İkili sistemde iki sembol, 0 ve 1 kullanılır ve örneğin 101101 ikili bir sayıya örnektir. Ondalık sistemle karşılaştırıldığında bunun, $(1x2^5)+(0x2^4)+(1x2^3)+(1x2^2)+(0x2^1)+(1x2^0)$ biçiminde olacağı sonucu kolayca çıkarılabilir. Bu sayının ondalık sistemdeki eşdeğeri, (1x32)+(0x16)+(1x8)+(1x4)+(0x2)+(1x1) = 45 sayısı olmaktadır. Bu durum çoğu kez şu şekilde gösterilir:

$$(101101)_2 = (45)_{10}$$

Burada indisler ilgili sayı sistemlerinin tabanını göstermek için kullanılmaktadır.

Örnek 1.1 : (10101100)'nin ondalık eşdeğerini türetin.

Cevap : $(10101100)_2 = (1x2^7)+(0x2^6)+(1x2^5)+(0x2^4)+(1x2^3)+(1x2^2)+(0x2^1)+(0x2^0)$
$= 128+32+8+4$
$= (172)_{10}$

Örnek 1.2 : $(1001001101100011)_2$ 'nin ondalık eşdeğerini türetin.

Cevap : $(1001001101100011)_2 = (1x2^{15})+(1x2^{12})+(1x2^9)+(1x2^8)+(1x2^6)+(1x2^5)+(1x2^1)+(1x2^0)$
$= 32768 + 4096 + 512 + 256 + 64 + 32 + 2 + 1$
$= (37731)_{10}$

İkinci örnek bize, elimizin altında, 1'den *n*'ye kadar sayıların 2 karşılıkları tablosunun bulunmasının, ikiliden ondalığa dönüştürme yaparken oldukça yararlı olacağını göstermektedir. Bu nedenle, Tablo 1.2'de böyle bir tablo referans olarak verilmiştir.

S 1.3

Ondalıktan İkiliye Dönüştürme

Ondalık nicelikleri bilgisayarda işlemek için önce, bunları ikili biçimlerine çevirmek gerekir. Bu işlem, ilgili değeri ardışık olarak 2 ile bölme ve kalanları kaydetmekle yapılır. Bu işlem aşağıdaki örnekte gösterilmiştir :

Örnek 1.3 : $(1833)_{10}$ 'un ikili eşdeğerini türetin.

Cevap 2 ile bölme:

916 kalan 1 — $2\overline{)1833}$

Tekrar bölme

458 kalan 0 — $2\overline{)916}$

$\vdots$

229 kalan 0 — $2\overline{)458}$

114 kalan 1 — $2\overline{)229}$

57 kalan 0 — $2\overline{)114}$

28 kalan 1 — $2\overline{)57}$

$\vdots$

14 kalan 0 — $2\overline{)28}$

7 kalan 0 — $2\overline{)14}$

3 kalan 1 — $2\overline{)7}$

1 kalan 1 — $2\overline{)3}$

$\vdots$

0 kalan 1 — $2\overline{)1}$

Sayının ikili eşdeğeri, kalanların aşağıdan yukarıya okunmasıyla elde edilen sayıdır. Yani (1833) = $(11100101001)_2$.

Bu sonucun sağlaması, tersine açma işlemiyle, yani $(11100\ 101001)_2$ 'nin ondalık eşdeğerini bularak yapılabilir:

$$(11100101001)_2 = 2^{10}+2^9+2^8+2^5+2^3+2^0$$

Tablo 1.2'yi kullanarak:

$$(11100101001)_2 = 1024+512+256+32+8+1=(1833)_{10}$$

sonucu bulunur.

Yukarıda anlatılanlar, ikili sistemin yalnızca iki sembol kullanarak bir sayıyı göstermeyi mümkün kıldığını göstermektedir. Bu nedenle, bir önceki kısımda da açıklandığı gibi, her birinde tek bir ikili basamağın gösterildiği bir grup sayısal devre kullanarak bir bilgisayarda, sayıları doğru olarak göstermek mümkündür. İkili sistem, bilgisayarda sayıları göstermek için kullanılmasının yanında, bilgisayar belleğinde saklı bulunan komutları göstermek için de kullanılır. Göstermenin nasıl yapıldığına ilişkin ayrıntılar ileride bir bölümde anlatılacaktır; ancak sonuç olarak, bir mikrobilgisayar sisteminin çalışması incelenirken her zaman ikili desenler kullanılır. Bu gösterim biçimi, çok sıkıcı gelebilir ve çoğunlukla ikili sayılar çok sayıda bitten oluştuğu için, bu bilgiyi başkalarına iletirken hata yapılma olasılığı yüksektir.

Tablo 1.2 *2'nin üsleri*

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768
16	65536
17	131072
18	262144
19	524288
20	1048576
21	2097152
22	4194304
23	8388608

Bu nedenle buna alternatif bir kısa gösterme biçimi kullanılır. Bu yöntem, belli sayıdaki bitleri bir grupta toplayıp, bu grupları eşdeğer bir sayı veya karakterle göstermektir. En çok kullanılan bit sayısı 3 ve 4'tür ve bunlar sırasıyla sekizli ve onaltılı sayı sistemine karşılık gelir.

S 1.2

Sekizli Sistem

Sekizli sayı sisteminin tabanı, adından da anlaşılacağı gibi sekizdir ve sistem, sekiz tane sembol (0-7) kullanır. Sekizli sayılar, yukarıda da belirtildiği gibi, ikili bir sayıyı göstermenin kısaltılmış biçimi olmanın dışında pek fazla kullanılmazlar. Bu gösterme biçimi, sekizli sistemin tabanı (sekiz sayısı), ikinin bir üssü olduğu için mümkün olmaktadır. Bu nedenle, ikiliden sekizliye ve sekizliden ikiliye sayı dönüştürmenin çok kolay bir yöntemi vardır.

İkili bir sayının basamaklarındaki 2'nin katlarını göz önüne alalım:

$$2^8\ 2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$$

Eğer bunlar üçer bitlik gruplara bölünürse, her grubun toplamı, sekizli bir basamağın maksimum değerini verecektir :

$2^8\ 2^7\ 2^6$ $=448$ $=7\text{x}8^2$

$2^5\ 2^4\ 2^3$ $=56$ $=7\text{x}8^1$

$2^2\ 2^1\ 2^0$ $=7$ $=7\text{x}8^0$

Bu nedenle, ikili bir sayının sekizli biçimde gösterimini elde etmek için ikili sayıyı üç bitlik gruplara bölüp, her üç-bitlik grubu da sekizli basamak eşdeğerine çevirmek yeterlidir. Örneğin:

$(101111001)_2$

sayısı; önce bu ikili sayı üç bitlik gruplara bölünerek

$\underbrace{101}\ \underbrace{111}\ \underbrace{001}$

daha sonra da her üç bitlik grup ikiliden sekizliye dönüştürülerek

$\underbrace{101}_{5}\ \underbrace{111}_{7}\ \underbrace{001}_{1}{}_2 = (571)_8$

biçiminde sekizli bir sayı olarak gösterilebilir.

Tablo 1.3 : *Bazı ikili ve bunların eşdeğeri sekizli sayılar.*

İkili	*Sekizli*
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	10
1001	11
1010	12
1011	13
1100	14
1101	15
1110	16
1111	17
10000	20

Böylece $(101111001)_7$ 'nin sekizli kısaltılmışı olan ve nispeten daha kolay anımsanacak bir sayı, $(571)_8$ sayısı elde edilir.

Sekizliden ikiliye dönüştürme ise yukarıdaki işlemin tersidir. Örneğin $(571)_8$; önce her sekizli basamağın 3-bitlik ikili eşdeğeri formunda gösterilmesi ve sonra da bu bitlerin birleştirilmesiyle dönüştürülür :

$$(5\quad 7\quad 1)_8 = (101\quad 111\quad 001)_2$$

$$101 \quad 111 \quad 001 = 5 \quad 7 \quad 1$$

Yani, $(571)_8 = (101111001)_2$. Referans olarak kullanılmak üzere, Tablo 1.3'de bazı ikili sayılarla bunların sekizli biçimdeki eşdeğerleri verilmiştir.

Sekizli sistem özellikle, 9 veya 12 gibi üç bitin katlarına sahip ikili sayıları göstermek için cazip bir kısaltma biçimidir. Öte yandan birçok bilgisayar 4, 8, 16, 32 gibi dört bitin katlarını kullanır. Dolayısıyla, bu makineler için sekizli sistem biraz yetersiz kalır ve bu durumda dört bitlik gruplama yapan alternatif bir sistemin bariz avantajları vardır. Bu, tabanı 16 olan ve onaltılı sistem denen bir sayı sistemine karşılık gelmektedir.

S 1.4

Onaltılı Sistem

Onaltılı sistemin tabanı 16'dır ve bu nedenle her bir basamağı göstermek için 16 sembol gereklidir. Ondalık sistem ancak on tane tek basamak sembolü (0-9) sağladığı için altı sembol daha gerekmektedir. Kullanılan 16 sembol; 10 ondalık basamak (0-9) ile 6 alfabetik karakterdir (A,B,C,D,E,F). Bu alfabetik karakterlerle ondalık sistemde iki basamak gerektiren altı sayıyı;

yani sırasıyla 10,11,12,13,14 ve 15'i göstermek için kullanılır. Örneğin, $(9A7F)_{16}$ onaltılı bir sayıya örnektir. Onaltılı sayıların tabanı 16 olduğu için, ikiliden onaltılıya dönüştürmek, ikili bitleri dörtlü olarak gruplayıp, bunları tek bir onaltılı sembol ile göstermekle yapılabilir. Tablo 1.4'te, on altı tane onaltılı sembol ve bunların eşdeğeri olan ikili sayılar gösterilmiştir.

Örneğin;

$\underbrace{1011}_{B}\ \underbrace{0111}_{7}{}_2 = B7_{16}$

Aynı şekilde:

$\underbrace{1100}_{C}\ \underbrace{0101}_{5}\ \underbrace{1001}_{9}\ \underbrace{1111}_{F}{}_2 = C59F_{16}$

Sayıların onaltılı gösterme şekillerinin ikiliye dönüştürülmesi de bu işlemin tersidir. Örneğin:

$$C\quad 5\quad 9\quad F = 1100010110011111_2$$

$$1100\quad 0101\quad 1001\quad 1111$$

Benzer biçimde:

$$2\quad A\quad F\quad 1 = (0010101011110001)_2$$

$$0010\quad 1010\quad 1111\quad 0001$$

Tablo 1.4 : *4 bitlik ikili sayılar ve bunların eşdeğeri onaltılı basamaklar*

İkili	*Onaltılı*
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Onaltılı sayılar, örneğin $(C59F)_{16}$, genelde C59FH (H, onaltılı anlamına gelen İngilizce "hexadecimal" sözcüğünü gösterir) veya C59F (onaltılı) şeklinde gösterilirler. Birçok mikrobilgisayar 8 veya 16-bit temel birimi kullandıkları için, onaltılı sistem bu sistemlerde bulunan ikili desenleri göstermede en yaygın kısa biçimdir ve bu kitapta da geniş bir şekilde kullanılacaktır.

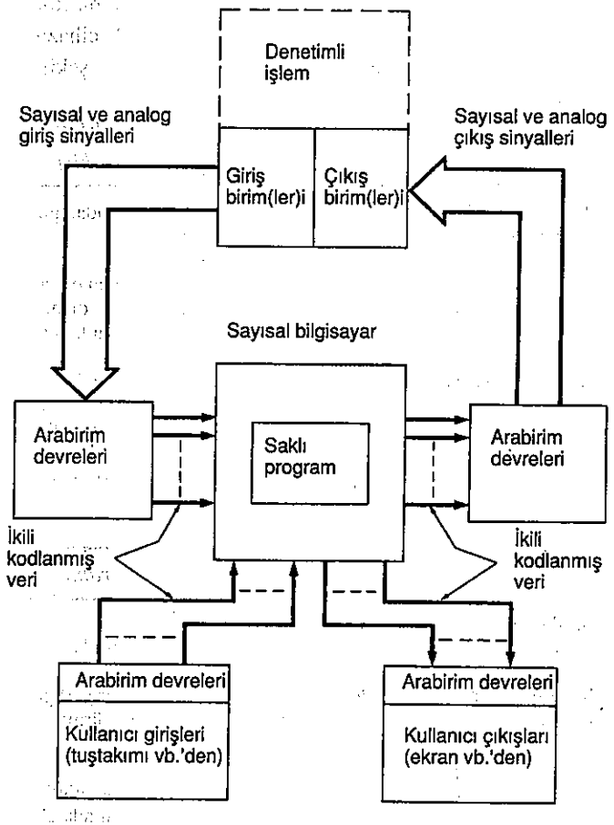
1.6 Bilgisayar Arabirimi

Elektronik bir hesap makinesine girilen veriler veya ondan üretilen sonuçlar daima sayısal (yani ondalık sayıları gösteren ve-

riler) olsa da, sayısal bilgisayarlar tarafından kullanılan giriş ve çıkış verileri, normal sayısal değerlerin yanında değişik türdeki parametreleri de gösterebilir.

Örneğin, bilgisayarın bir odadaki hava sıcaklığını denetlemek için kullanıldığı bir uygulamada, giriş verisi odanın o anki sıcaklığını, çıkış verisi de ısıtıcı tarafından üretilecek sıcaklık düzeyini gösterebilir. Benzer şekilde, bilgisayarın bir çamaşır makinesindeki işlemleri kontrol etmek için kullanıldığı bir uygulamada da, giriş verileri sıcaklık, çamaşır kazanındaki suyun düzeyi gibi parametreleri, çıkış verileri de motoru veya su pompasını çalıştırma veya durdurma sinyallerini gösterebilir.

Öte yandan, ne tür bir uygulama olursa olsun, giriş verileri ve bilgisayardan çıkan veriler daima ikili kodlanmış biçimdedirler. Bu nedenle, veriler ister sayısal, ister bir odanın sıcaklığı gibi analog bir değeri göstersin, aynı işlemler bu veriler üzerinde yapılabilir. Aslında bilgisayar verilerin neyi gösterdiğine ilişkin hiçbir bilgiye sahip



Şekil 1.4 Bilgisayar arabirimi

değildir. Uygulamaya bağlı olarak veriyi belirli bir değer olarak gösterecek biçimde işleyen ve yorumlayan sadece kullanıcıdır.

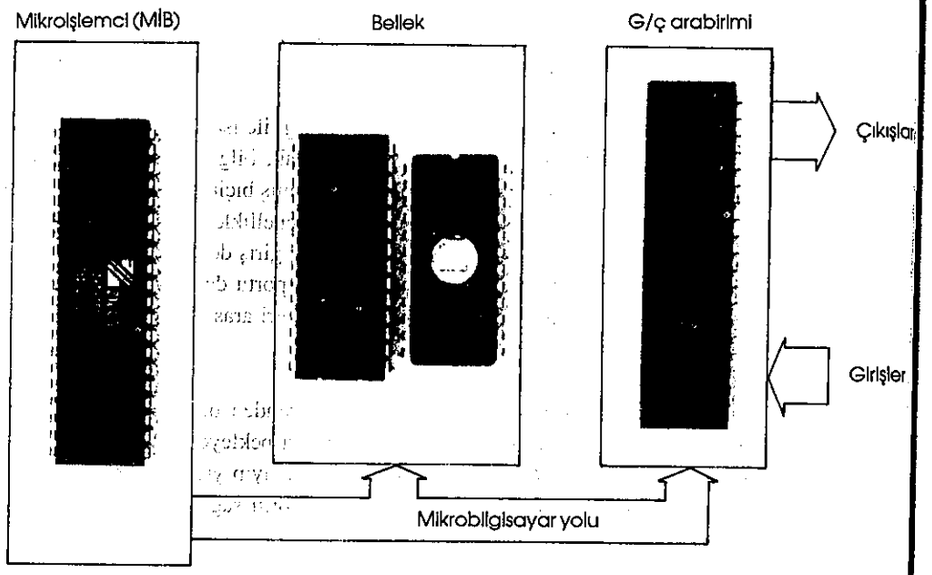
Bu, bilgisayar ile kullanılan giriş ve çıkış birimleri arasında uygun *arabirim devreleri* bağlanarak gerçekleştirilir. Bu devreler, cihaza bakmaksızın bilgisayara daima standart ikili kodlanmış arabirim sundukları için, bilgisayarı bu giriş çıkış birimlerinden fiilen ayırırlar.

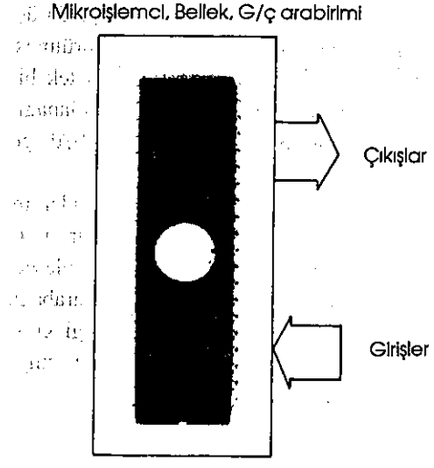
Eğer giriş verisi, sözgelimi, bir işlemin sıcaklığını gösteren ve sıcaklığa-duyarlı bir cihazdan gelen sürekli değişen (analog) sinyal ise, sıcaklığın sayısal (ikili kodlanmış) eşdeğerini elde etmek için bir *analog/sayısal dönüştürücü (ADC)* kullanılır. Benzer biçimde, eğer bir çıkış cihazı tarafından analog bir sinyale gerek duyulursa, bilgisayarca üretilen sayısal (ikili kodlanmış) çıkış bir *sayısal/analog dönüştürücü (DAC)* aracılığıyla analog eşdeğerine dönüştürülür.

Eğer giriş verisi zaten sayısal biçimdeyse, veri doğrudan kullanılır. Örneğin bir düzey anahtarının durumu (açık veya kapalı) veya bir kontrol valfının durumu (açık veya kapalı) tek bir ikili basamağın durumuyla (1, cihazın açık olduğunu; 0 ise kapalı olduğunu göstermek üzere) gösterilebilir . Aynı şekilde bir bilgisayarın tek bir

Tablo 1.5 *Sayısal bir bilgisayarla kullanılan bazı arabirim devre örnekleri*

Giriş Kaynağı	*Arabirim Devresi*
İki durumlu cihaz (düzey anahtarı, termostat, vs.)	Doğrudan giriş yapılabilir
Basit bir sayısal tuştakımından ondalık basamaklar (0-9 artı A-F)	Seçilen ondalık basamağın 4-bit İkili kodlanmış eşdeğerini üreten sayısal kodlama devresi
Alfasayısal karakterler (*alfa*betik A-Z, *sayısal* 0-9) tam klavyeden	Seçilen karakterin 8 bit ikili kodlanmış eşdeğerini üreten sayısal kodlama devresi
Analog sinyal (sürekli değişen), örneğin basınca veya sıcaklığa duyarlı bir cihaz	Analog-sayısal dönüştürücü (ADC)
İstenen Çıkış	*Arabirim Devresi*
İki-durumlu cihaz (röle, kontrol valfi v.s)	Doğrudan denetlenebilir
Sayısal ve onaltılı görüntüleme birimi	(7 parçalı) görüntüleme birimi için sürme sinyalleri üreten *kod çözme* devresi
Alfasayısal karakter ekranı	Görsel görüntüleme birimi (VDU) ekranında görüntülenmek üzere sürme sinyallerini üreten kod çözme devresi
Analog sinyal, yani herhangi bir gerilim düzeyi	Sayısal-analog dönüştürücü (DAC)





Şekil 1.5 Mikrobilgisayar entegre devreleri (a) bir mikroişlemci, bellek ve (G/Ç) arabirim devresi (gerçek ölçülerde) (b) tek-yongalı mikrobilgisayar

ikili basamak çıkışı, bir röleyi açıp kapamak için de kullanılabilir (1 açmak, 0 kapatmak için). Bu da büyük bir motorun açılıp kapanmasını kontrol edebilir. Bu

durum diyagramsal olarak Şekil 1.4'te gösterilmiş ve Tablo 1.5'te, sayısal bilgisayarlarla kullanılan bazı tipik arabirim devre örnekleri verilmiştir.

1.7 Temel yapı

Daha önce belirtildiği gibi, giriş verisinin kaynağı ile istenen çıkış verisinin tipi, bilgisayarın yer aldığı uygulamaya bağlıdır. Ancak, bilgisayardaki tüm veriler ve o anki giriş ve bilgisayar çıkış verileri ikili kodlanmış biçimdedir. Dolayısıyla, bir bilgisayardaki tüm giriş çıkış komutları, sabit (genellikle 8-bit) uzunlukta ikili kodlanmış olarak, ya bir *arabirim portuna* giren bir giriş değeri ya da bir arabirim portundan alınan çıkış değeridir. Bunlara arabirim portu denmesinin nedeni bilgisayar ile bir önceki kısımda ele alınan arabirim devreleri arasındaki arabirimi oluşturuyor olmalarıdır.

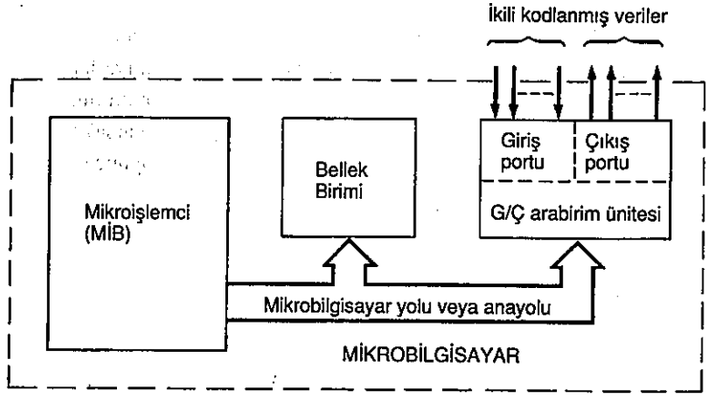
Dolayısıyla, sayısal bir bilgisayar şu üç bileşenden oluşmaktadır: hem programı oluşturan komutların listesini, hem de işlenmeyi bekleyen veri değerlerini tutan bir *bellek birimi*, her bir program komutunu yorumlayıp yürüten bir *merkezi işlem birimi (MİB)*, bir veya daha çok giriş ve çıkış portu sağlayan *giriş-çıkış (G/Ç) arabirim ünitesi*.

Eğer merkezi işlem birimi tek bir entegre devre halinde yapılmışsa, buna *mikroişlemci*, (mikroişlemci, bellek, G/Ç arabiriminden oluşan) sistemin tamamına da *mikrobilgisayar* denir. Örnek olarak, Şekil 1.5(a)'da üç entegre devreden oluşan ve halen kullanılmakta olan bir mikrobilgisayar görülmektedir. Aslında entegre devre teknolojisi şu an öyle bir aşamaya gelmiştir ki tek bir entegre devre üzerinde bir mikroişlemci, belli bir miktarda bellek ve G/Ç olanağı bulmak mümkündür. Bu tip bir tek-yongalı mikrobilgisayar örneği Şekil 1.5(b)'de görülmektedir.

Açıktır ki, örneğin her bir komutu bellek biriminden mikroişlemciye getirmek veya bir veri değerinin mikroişlemciye veya G/Ç arabirim ünitesine girişi ya da bunlardan çıkışı için, yukarıda adı geçen bu üç birim birbiriyle iletişim kurabilmelidir. Bu, *bilgisayar yolu* veya *anayolu* denilen elektriksel arabağlantı sistemi kullanılarak yapılır. Çünkü, bir sonraki bölümde de görüleceği gibi, bilgisayardaki tüm iletişim aynı (yol biçiminde düzenlenmiş) elektriksel sinyal hatları kullanılarak gerS1.6 çekleştirilir. Bu, Şekil 1.6'da gösterilmektedir.

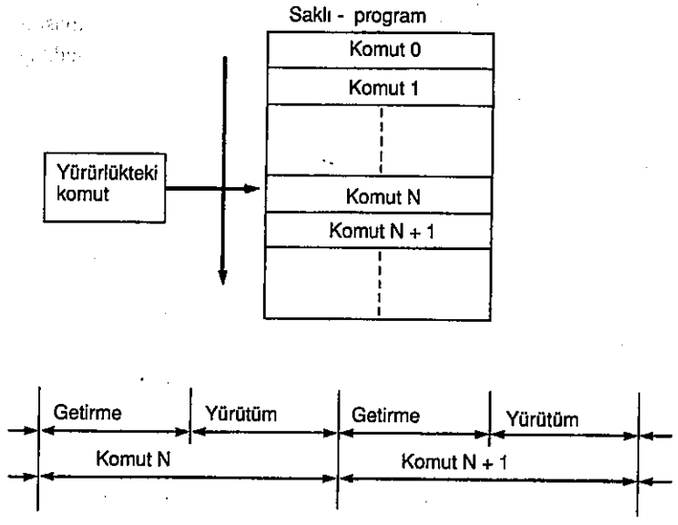
1.8 Getir-yürüt saykılı

Programı oluşturan komutlar listesi belli bir görevi başaracak biçimde geliştirildiğinde, tüm komut listesi mikrobilgisayarın belleğine yüklenir ve daha sonra program



Şekil 1.6 Bir mikrobilgisayarın temel yapısı

çalıştırılır (veya diğer deyimle yürütülür). Programın yürütümü sırasında mikroişlemci, her komutu, (bilgisayar yolu aracılığı ile) sırayla bellekten getirir, komutta belirtilen işi belirler ve bu işi yapar (yürütür). Bu yüzden "mikroişlemci iki-fazlı modda işlem yapar" denir. İlk faz sırasında bellekten komutu getirir - *getirme fazı*- ve ikinci faz sırasında da komutta beliritilen işi yürütür - *yürütme fazı*. Bu getir-yürüt saykılı, *komut saykılı* olarak bilinir ve Şekil 1.7'de gösterilmiştir.



Şekil 1.7 Getir-yürüt komut saykılı

3. Bölümde görüleceği gibi, bazı mikroişlemci komutları, bellekte saklanabilmek için tek bir *baytlık* (8 bit) bellek alanına ihtiyaç duyarlarken bazıları iki, bazıları da üç bayta ihtiyaç duyabilir. Öte yandan, komut saykılının getirme fazı boyunca mikroişlemci komutun ilk baytından kalan baytların (varsa) sayısını ve anlamını belirler. Böylece komutu yürütmeden önce getirme fazı boyunca gerekli sayıda baytı getirir.

Sorular

1.1 Aşağıdaki terimlerin anlamını açıklayın :
(a) saklı-program kavramı
(b) programlanabilir elemanlar.

1.2 Aşağıdaki ondalık sayıları ikili biçimdeki eşdeğerine dönüştürün:
27, 63, 226.

1.3 Aşağıdaki ikili sayıları ondalık biçimdeki eşdeğerine dönüştürün:
00101101, 10101010, 11011011.

1.4 Aşağıdaki ikili sayıları sekizli biçimde gösterin:
101011, 111001110, 100010110000.

1.5 Aşağıdaki onaltılı sayıların ikili biçimdeki eşdeğerlerini gösterin:
C2, 8E1, 3B2F.

1.6 Tipik bir mikrobilgisayarın blok diyagramını çizin ve her bir bileşenin işlevini açıklayın. Bu bileşenlerin iletişim mekanizmasını ve bir mikroişlemcinin, belleğinde saklı bir programı yürütme şeklini açıklayın.

2. Bölüm : **Mikrobilgisayar mimarisi**

Bu bölümün amaçları : *Bu bölümü bitirdiğinizde:*

1 *Bir bilgisayarı oluşturan üç bileşenin (bellek birimi, mikroişlemci, G/Ç ara - birim ünitesi) blok diyagramını şematik olarak çizip bunların nasıl çalıştıklarını açıklayabilmeli,*
2 *Bellek haritası teriminin anlamını ve RAM, ROM, UVPROM, EAROM te - rimleri arasındaki farkı anlayabilmeli,*
3 *Bilgisayar yolunun yapısını anlamış olmalı ve zamanlama diyagramları kullanarak bir bilgisayarı oluşturan birimler arasında bilginin nasıl aktarıldığını açıklayabilmelisiniz.*

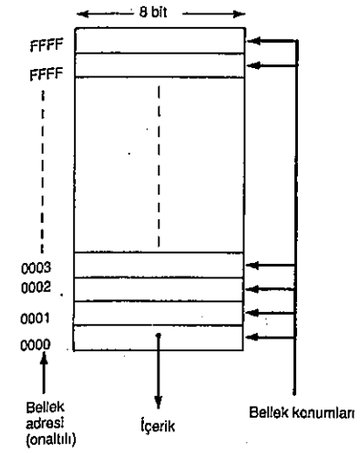
2.1 Giriş

Bir önceki bölümde bilgisayarın üç temel birimi tanıtılmıştı. Bunlar; programın ve onunla ilgili veri değerlerinin saklandığı bellek birimi, programı oluşturan komutlar listesini yorumlayıp yürüten mikroişlemci (MİB) ve uygulamada kullanılan belirli giriş ve çıkış birimleri ile bilgisayarı birbirine bağlamak için bir veya daha çok giriş çıkış portu sağlayan G/Ç arabirim ünitesidir. Bu bölümde, bu birimlerin yapısı ve çalışması ile bu birimleri birbirine bağlayan bilgisayar yolunun yapısı ayrıntılı olarak tanımlanacaktır.

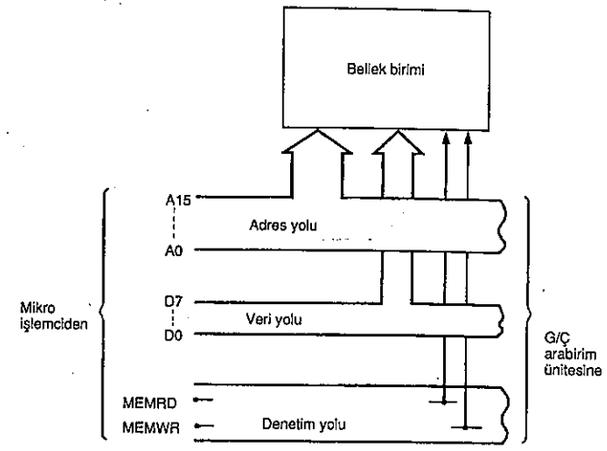
2.2 Bellek birimi

Bellek birimi, ayrı ayrı olarak tanımlanabilen ve *konum* (veya *göz*) adı verilen bir kutucuklar grubundan meydana gelen bir yapı olarak düşünülebilir. Her bir konum, tek bir sabit-uzunlukta ikili desen tutabilir (saklayabilir), mikroişlemci sistemlerinin çoğunda bu desen, 8-bit yani bir *bayt*tan oluşur. Bir bellek konumunda herhangi bir anda saklı olan 8-bitlik değer; o yerin *içeriği* olarak bilinir.

Her bir konum, o konumun *adresi* olarak adlandırılan farklı bir sayı ile tanımlanır. Birçok mikroişlemci uygulamaları için gerekli olan konum sayısı (ve dolayısıyla adres sayısı) az olabilse de -tipik olarak 2.000'den az-, daha karmaşık uygulamalar için çok fazla sayıda konuma ihtiyaç duyulur. Bu nedenle, belirli bir grup uygulamanın çalışmasını sağlayabilmek için, bir mikrobilgisayar belleğindeki bellek konumu sayısı normal olarak 65,536 (ondalık) değerine kadar genişletilebilir.



Şekil 2.1 Bellek birimi



Şekil 2.2 Bellek birimi ile iletişim

Bellekteki ilk konum, daima sıfırıncı adrese atanır ve bu nedenle de bir mikrobilgisayar belleğindeki her bir konumu belirtebilmek için gereken adres aralığı 0'dan 65.535'e kadardır. İkili sistemde 65.535 ondalık sayısı 1111111111111111 ikili sayısına karşılık gelir, yani 16 tane ikili birden oluşur. Bu değer onaltılı sistemde, FFFF olarak dört tane onaltılı karaktere karşılık gelir ve bu nedenle bir mikrobilgisayar sisteminde normalde yer alan bellek adres aralığı 0000 (onaltılı) → FFFF (onaltılı) olarak ifade edilir. Bu durum, diyagram olarak Şekil
S 2.1 2.1'de gösterilmektedir.

Bir bellek konumunun içeriğine erişebilmek için, bu yerin adresi onaltı ikili *adres hattı* (A_0'dan A_{15}'e) üzerinden bellek birimine gönderilir ve erişimi yani *okuma* işlemini başlatmak için bir sinyal verilir. Bu sinyale, *bellekten oku* sinyali (MEMRD) denir ve bellekten okuma *denetim hattına* çıkarılır. Bellek birimi bu konumun içeriğini (8-bitlik bir değeri) sekiz *veri hattına* (D_0'dan D_7'ye) çıkararak karşılık verir.

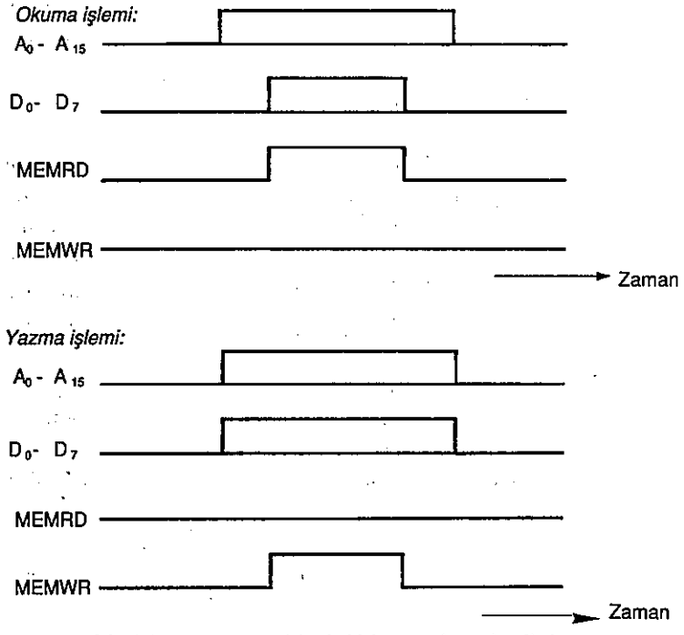
Benzer biçimde, herhangi bir değeri bir bellek konumuna yerleştirmek için de, bellek konumunun adresi onaltılı adres hattına, saklanacak 8-bitlik değer de veri hattına çıkarılır ve saklama, yani *yazma* işlemi için bir sinyal verilir. Bu sinyale,

bundan dolayı, *belleğe yazma* sinyali (MEMWR) denir ve belleğe yazma denetim hattına çıkarılır. Bu işlemler Şekil 2.2'de gösterilmiştir.

Şekilde, bellekteki bir konumdan bir değer okumak veya bu konuma değer yazmak amacıyla bellek birimi ile iletişim için kullanılan hat grupları gösterilmektedir. Görüleceği gibi, bellek birimi ile iletişimde kullanılan adres, veri ve denetim hatları, G/Ç arabirim ünitesi ile iletişim kurulması için de kullanılır. Bu hatlar sırasıyla *adres yolu*, *veri yolu* ve *denetim yolu* olarak adlandırılırlar ve bunların tümü, 1. Bölüm'de sözü edilen bilgisayar yolunu ya da anayolunu oluştururlar.

Burada dikkat edilmesi gereken nokta; veri yolunun, okuma işlemi sırasında mikroişlemciye *giriş*, yazma işlemi sırasında da mikroişlemciden *çıkış* olacak değeri tutmasıdır. Bu nedenle veri yolu *iki yönlü yol* olarak anılır ve elektriksel özellikleri daha sonraki bir bölümde izah edilecektir.

Şekil 2.3'te, bir bellek konumundan değer okumak veya ona bir değer yazmak için kullanılan tipik bir elektriksel veya *zamanlama sinyalleri* grubu görülmektedir. Bunlar *zamanlama diyagramları* olarak bilinirler, çünkü dalga biçimlerinin (ikili geçişler) her biri diğerine göre zaman sırasında gösterilmiştir. Her bir adres ve veri hattı, kullanılmakta olan belirli adres ve veri değerlerine bağlı olarak ikili 1 veya 0 olacağı için, bu hatlar şekilde yüksek (ikili 1) veya alçak (ikili 0) olarak görülmektedir.



Şekil 2.3 Okuma ve yazma işlemleri için zamanlama sinyalleri

Sadece-okunur ve rastgele erişimli bellekler

Özel birçok mikrobilgisayar uygulamaları (saklı programın sabit kaldığı uygulamalar) için, bellek birimi normalde iki parçadan oluşur :

1- *Sadece-okunur bellek birimi :* Bu birim; sabit, değişmeyen programı oluşturan makine komutları listesini tutmak (saklamak) için kullanılır.
2- *Rastgele erişimli bellek :* Bu birim ise, programla ilgili tüm veri değerlerini, yani işlenmeyi bekleyen veri değerleri, G/Ç arabirim ünitesinde bulunan ve bir porta çıkış için bekleyen veriler gibi değerleri tutmak için kullanılır.

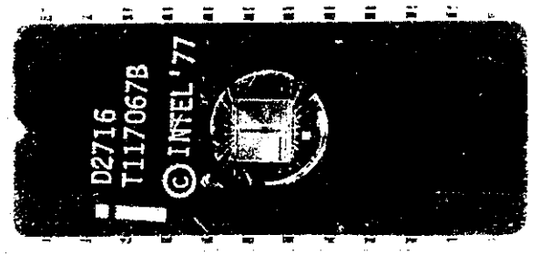
Sadece okunur belleğin (ROM) şöyle bir özelliği vardır: Üretimi sırasında içine yerleştirilen bilgiler değiştirilemez ve bunun sonucu olarak yalnızca sadece-okunur modda çalışabilir. Ayrıca bir de *kalıcı* bellek olma özelliği vardır, çünkü üretimi sırasında içeriği bir kez sabitlendiğinden aynı içerik, elektrik enerjisi kesildiğinde kaybolmayıp varlığını her zaman için korumaktadır. Bu nedenle, belli bir göreve özel bir uygulama için mikrobilgisayara her elektrik enerjisi uygulanması durumunda aynı program, bilgisayarın belleğine yüklenecektir.

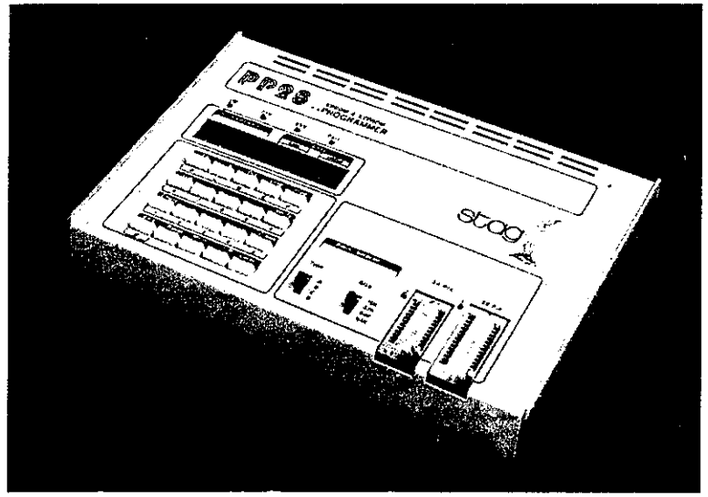
Rastgele erişimli bellek (RAM), belli bir bellek konumunun içeriğini okuyabilmesinin yanında, bellekteki bir konuma yeni bir bilgi yazabilme özelliğine de sahiptir. Bu nedenle, okunur/yazılır bellek olarak da adlandırılır. RAM normalde *kalıcı-olmayan* bir bellektir, çünkü kendisine elektrik uygulandığı sürece içeriği değişmese de, kaynak gerilimi kesildiğinde içeriği silinir ve güç yeniden uygulandığında rastgele değerler içerir.

Üretimi sırasında ROM'un içine sabitlenen ve program komutlarını gösteren ikili desenler değiştirilemez ve bu nedenle de programın, üretimden önce doğru ve hatasız olması şarttır. Ayrıca eğer bu tip (yani aynı programı içeren) ROM'lardan az sayıda üretilecekse, fabrikada ROM hazırlamanın bedeli yüksek olacaktır. Bu yüzden fabrikada hazırlanmış ROM'lar sadece büyük hacimli uygulamalarda, mesela çamaşır makinesi denetleyicileri ya da bilgisayar oyunlarında kullanılırlar.

Yalnızca az sayıda ROM elemanının kullanıldığı veya üretilmiş bulunan ROM'ların içerdiği program üzerinde değişiklik (örneğin programa ekleme yapmak veya bir hatayı düzeltmek gibi) olanağının istendiği uygulamalar için daha ekonomik çözüm şudur: *Silinebilir programlanabilir ROM* ya da diğer deyimle *EPROM* kullanmak.

Normal çalışma modu süresince EPROM, fabrikada yapılmış ROM ile tamamen aynı biçimde davranır. Ancak bir özelliği vardır; o da içinde kayıtlı olan bellek de-

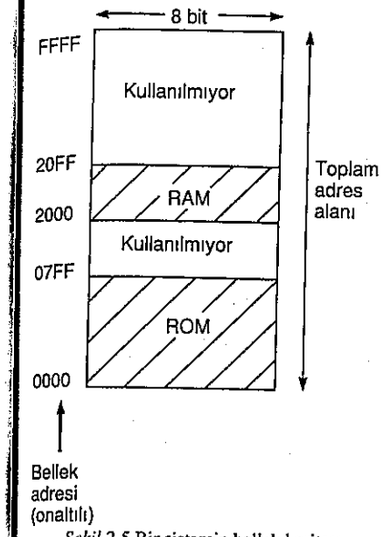




Şekil 2.4 : UVPROM (*üstte*) ve PROM programlayıcısı (*altta*) (Stag Electronic Designs Ltd.'in izniyle)

seninin bir kullanıcı tarafından denetlenen bir yöntemle değiştirilebilmesidir. Bunlar özellikle prototipleme sistemleri için çok yararlı olmaktadır, çünkü burada ürün geliştirilirken programda sürekli değişiklik yapmak gerekmektedir.

EPROM'ların içeriği, ya entegre devre üzerindeki bir pencereden onu yoğun ultraviyole ışınlara maruz bırakmak (*UV EPROM*) ya da üzerindeki belirli bağlantı uçlarına (pinlerine) gerilim uygulamak (elektriksel olarak değiştirilebilir ROM ya da *EAROM*) koşuluyla silinebilir. Yeni içerik daha sonra *PROM programlayıcı* denilen bir araç kullanılarak PROM'a yazılır (ya da kaydedilir). UV PROM ve ilgili PROM programlayıcısına bir örnek, Şekil 2.4'te gösterilmiştir. Bir kez daha vurgulamak gerekir ki, normal çalışma modları süresince EPROM'lar fabrikada yapılmış ROM'larla tümüyle aynı biçimde davranırlar.



Şekil 2.5 Bir sistemin bellek haritası

Bellek Haritası

Bir mikrobilgisayarın içerdiği toplam konum sayısı veya *adres alanı* geniş olabiliyorsa da, özel birçok küçük uygulama için gereken bellek (sadece-okunur ve rastgele erişimli) büyüklüğü oldukça düşük oranda olabilir. Belli bir uygulama için kullanılan bellek aralığı ve her aralıktaki bellek birim tipi, sistemin bir *bellek haritası* ile belirtilir.

Küçük fakat tipik bir mikrobilgisayar uygulamasının bellek haritası örneği Şekil 2.5'te gösterilmiştir. Bu uygulamada, sabit program belleği için 2048 bayta kadar [0000'dan 07FF'e (onaltılı) kadar] ROM ve veriler için de 256 bayta kadar [2000'den 20FF'e (onaltılı)] RAM'e ihtiyaç duyulduğu varsayılmıştır.

Bellek haritasında, birbiriyle bitişik olmayan bir adres alanı kullanmak (yani her bir bellek tipi arasında kullanılmayan konumlar bırakmak) normaldir. Bunun nedenleri; ileride ortaya çıkabilecek genişlemelere imkan sağlamak ve daha sonra görüleceği gibi, bir okuma veya yazma işlemi sırasında hangi bellek tipinin seçileceğini belirleyen elektronik devreyi basitleştirmektir.

Normalde bir mikrobilgisayar uygulamasındaki program için gerekli olan bellek büyüklüğü, 1024 baytın katlarıyla ifade edilir. Bu bir sadece-okunur bellek entegre devresinin temel birim ölçüsüdür ve 1 Kbayt olarak gösterilir. Böylece, örneğin yukarıda verilen uygulama, 2 Kbayt ROM kullanır; birçok mikroişlemcideki maksimum adres alanı da 64 Kbayttır, çünkü 64 Kbayt, 64x1024=65.536 ondalık konuma karşılık gelmektedir.

S 2.2, 2.3

2.3 Mikroişlemci

Bir mikroişlemci, belleğinde saklı bulunan programı her bir ko-

mutu sıra ile okuyarak yürütür. Her komut önce, onu yürütmek için gerekecek işlemleri belirlemek üzere yorumlanır (*kodu çözülür*) sonra da gereken işlemler uygulanır.

Programın yürütülmesi sırasında her bir program komutu ve programla ilgili ara (geçici) veri değerleri, mikroişlemcinin içinde, *kaydedici* adı verilen belli sayıdaki özel konumlarda saklanır. Bir mikroişlemci kaydedicisi, bellek birimindeki konumların bir benzeridir; çünkü kaydediciler, sekiz ve onaltı bitlik tek bir ikili sözcüğü tutar (saklar). Öte yandan, her biri belli bir işlevi yerine getirmek için kullanıldığından, mikrobilgisayar kaydedicileri bir adresle değil, sembolik adlarla anılırlar.

Daha önce belirtildiği gibi mikroişlemci, belleğinde saklı bulunan programı sırayla her program komutunu okuyarak yürütür. Bundan dolayı mikroişlemcinin, yürüteceği bir sonraki işlemin bellek birimindeki konumunun adresini anımsaması ya da kaydını tutması gerekir. Bu nedenle mikroişlemcide bu işlevi yerine getirmek için kullanılan ve *program sayıcısı* (PC) denilen bir kaydedici bulunur. Program sayıcısı, bellek adresini tuttuğu için 16-bitliktir.

Bundan başka mikroişlemcide *komut kaydedicisi* (IR) denilen bir kaydedici vardır. Bu kaydedici, komutu yürütmek için gereken işlemleri belirleyebilmek için kodu çözülmekte olan komutu tutmak amacıyla kullanılır. Kod çözme işlemi, *komut kod çözücü birimi* tarafından gerçekleştirilir. Temelde bu birim hangi komutun okunmuş olduğunu belirler ve sistem saatinden üretilen elektriksel zamanlama sinyalleriyle bağlantılı olarak yukarıda Şekil 2.3'te görülenlere benzer zamanlama ve denetim sinyalleri üretir. Komut kaydedicisi, zamanlama ve kod çözme birimi hep birlikte mikroişlemcinin *denetim birimi* denilen birimini oluştururlar, çünkü mikrobilgisayarda kullanılan tüm elektriksel sinyaller buradan üretilir.

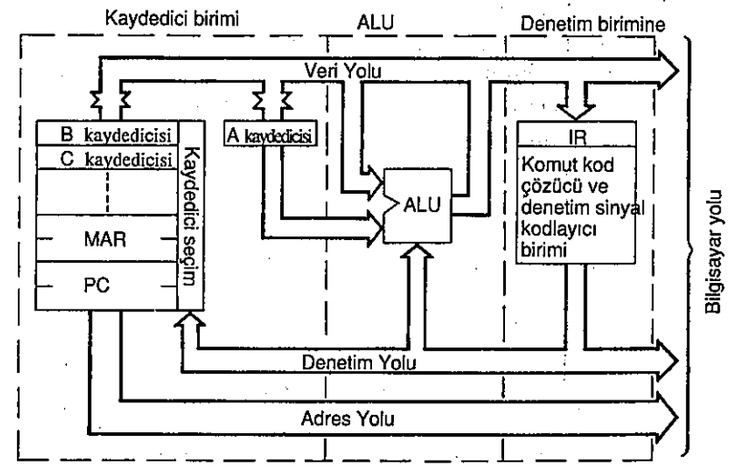
Toplama, çarpma gibi temel aritmetik işlemlerinin yanı sıra, bir mikroişlemci iki veri değeri arasında mantıksal işlemler de yapar. Bu komutların açık anlamı ve yararı daha sonraki bir bölümde açıklanacaktır, fakat burada bilinmesi gereken, mikroişlemci içinde bu veri işleme işlevini yerine getiren ve *aritmetik mantık birimi (ALU)* denilen bir birimin bulunduğudur.

ALU, iki değer üzerinde, (yürütülmekte olan komutta belirtilen) uygun aritmetik ve mantık işlemlerini yerine getirir. Bu iki değerden biri daima özel bir mikroişlemci kaydedicisi olan ve *A kaydedicisi* olarak adlandırılan kaydedicinin o an içerdiği değerdir. Öteki ise, ya belirli bir bellek konumunun, ya da mikroişlemcinin bir bölümünü oluşturan *kaydedici grubundaki* bir başka kaydedicinin içeriğidir. Bu kaydediciler işlenmek üzere bekleyen geçici değerleri tutmak için kullanılırlar ve normalde B kaydedicisi, C kaydedicisi şeklinde adlandırılırlar. Mevcut kaydedici sayısı bir mikroişlemciden diğerine farklılık gösterir.

Aritmetik veya mantıksal işlemlerin sonucu, normal olarak A kaydedicisine yerleştirilir ve dolayısıyla bu yeni değer orada var olan değerin yerini alır ya da diğer bir deyişle onun üzerine yazılır. Bu yüzden A kaydedicisi, içeriğinde belirli bir aritmetik işleminin sonucu biriktirildiğinden, *akümülatör* olarak adlandırılır.

Sonuç olarak, az önce de belirtildiği gibi, birçok mikroişlemci komutu, aritmetik veya mantıksal bir işlemde kullanılacak olan ikinci değerin tutulduğu bellek konumunun adresi örneğinde olduğu gibi, bellek konum adresini kendi içinde belirtir. Bu nedenle, ikinci bir 16-bitlik kaydedici (ya da bir çift 8-bitlik kaydedici) bellek adresini belirten komutların yürütülmesi sırasında bu adresi tutar. Bu kaydedici, *bellek adres kaydedicisi* (MAR) olarak bilinir.

Bu amaçla, tipik bir mikroişlemcinin şematik diyagramı Şekil 2.6'da gösterilmiştir. Bu mikroişlemci; yukarıda özetlenen ve çeşitli kaydedicileri içeren bir kaydedici birimi, çeşitli aritmetik ve mantıksal işlemleri yapan bir aritmetik-mantık birimi (ALU), mikrobilgisayarı oluşturan çeşitli birimler arasında ve mikroişlemcinin için-
S 2.4 deki kaydediciler ile ALU arasında bilgi (hem komut hem de veri) akışını denetlemek için zamanlama sinyalleri üreten bir denetim biriminden oluşmaktadır.



Şekil 2.6 Bir mikroişlemcinin blok şeması

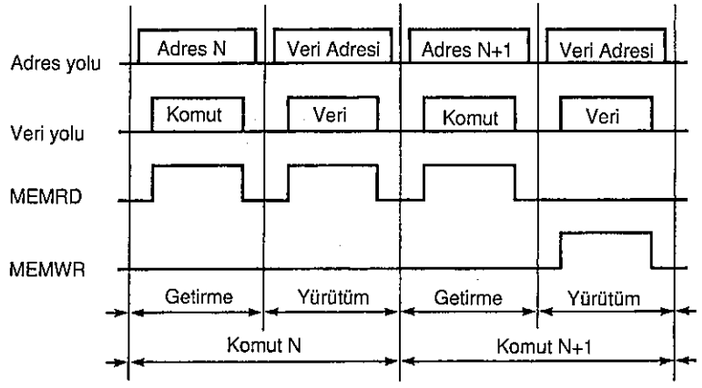
2.4 Bilgisayar yolu

Bu noktada belki de bazı tipik makine komutlarını yürütmek için mikroişlemci tarafından üretilen ve bilgisayar yolu üzerinde bulunan zamanlama sinyalleri üzerinde

durmakta yarar vardır. Bir programı yürütmek için mikroişlemci iki-fazlı ya da getir-yürüt modunda çalışır : Getirme fazı süresince sırada bulunan bir sonraki komut bellekten okunur; yürütme fazında da mikroişlemci, komutu yürütmek için gereken işlemleri uygulamaya koyar. Bu nedenle, her bir getir-yürüt saykılının tümüne komut saykılı denir.

Mikroişlemcinin içindeki program sayıcısı, daima yürütülecek bir sonraki komutun adresini tutar. Bundan dolayı, her bir komut saykılının getirme fazı boyunca mikroişlemci program sayıcısının içeriğini adres yoluna yerleştirir ve bir bellekten okuma sinyali üretir. Ardından, adreslenmiş konumun içeriği (yani komut) bellek birimi tarafından veri yoluna çıkarılır ve sonra da kod çözmeye ve yürütmeye hazır haldeki mikroişlemci tarafından komut kaydedicisine yüklenir.

Ardından, bellekten okunan bu komutun kodu çözülür ve yürütme fazı boyunca bu komutu yerine getirmek için gereken işlemler uygulamaya konulur. Örneğin komut, eğer bir değeri bellekteki bir konumdan, sözgelimi mikroişlemcideki bir kaydediciye okumaya ilişkinse; mikroişlemci, bellek konumunun adresini adres yoluna yerleştirir ve bir bellekten okuma sinyali üretir. Bellek birimi buna, adreslenmiş konumun içeriğini veri yoluna çıkararak karşılık verir ve sonra da mikroişlemci bunu uygun bir kaydediciye yükler.



Şekil 2.7 İki komut saykılı

Benzer biçimde, eğer komut herhangi bir değerin bir bellek konumuna yazılmasına ilişkinse, mikroişlemci konumun adresini adres yoluna, saklanacak değeri de veri yoluna yerleştirir ve bir belleğe yaz sinyali üretir. Şekil 2.7'de, iki ardışık komut saykılını gerçekleştiren örnek bir dalga biçimleri grubunu gösteren zamanlama diyagramı verilmektedir.

Şekilde ilk olarak, bir bellekten okuma işlemine, ikinci olarak da bir belleğe yazma

işlemine ilişkin komutu yürütme gösterilmiştir. Her bir getirme fazında, mikroişlemciden çıkan, adres program sayıcının o anki içeriğidir ve bu nedenle de sırayla N. adres ve N+1. adres olarak gösterilmiştir. Bundan dolayı program sayıcının içeriği, her getirme fazının ardından, bellekte yer alan bir sonraki işlemin başlangıcını göstermek üzere artırılır.

Daha sonraki bir bölümde görüleceği gibi, bazı mikroişlemci komutları komutu belirtmek için birden çok bayta ihtiyaç duyar. Örneğin, bazı komutlar bir bellek adresini ya da aslında onun içinde o an yer alan değeri içeriyor olabilirler. Bu nedenle, komut saykılının getirme fazı boyunca, komutu yürütmeye başlamadan önce mikroişlemcinin bellekten uygun sayıda baytı getirmesi gerekir.

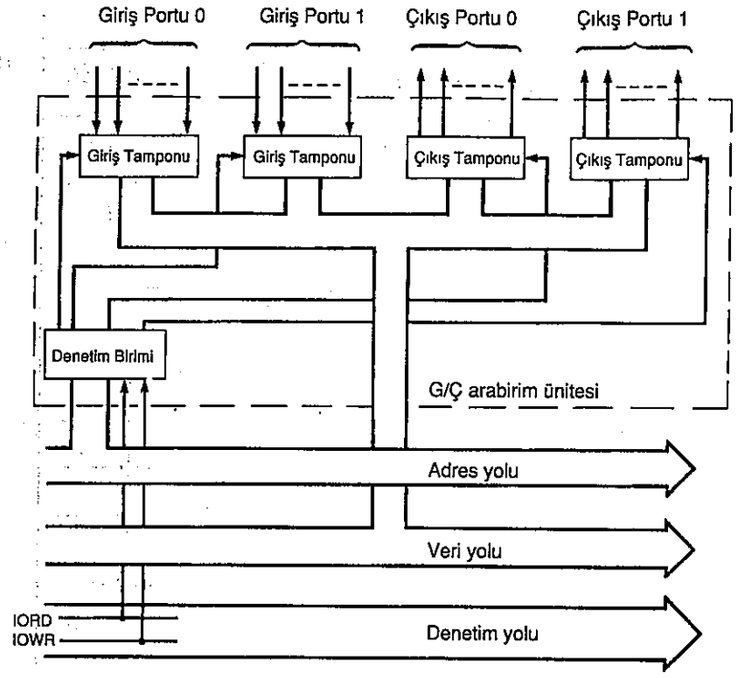
Bunu başarmak kolaydır, çünkü mikroişlemci her komutun ilk (yani komut kaydedicisine yüklenen) baytından, hem (varsa) bellekten getirilecek olan kalan bayt sayısını, hem de bunların anlamlarını çıkarabilir. Dolayısıyla, eğer (örneğin üç baytlık bir komut için) iki bayt daha gerekiyorsa, bunların kalan ikisi de getirme fazı sırasında iki ayrı okuma komutu ile iki ardışık bellek konumundan okunur. Ardından bu iki bayt, komuta bağlı olarak, uygun mikroişlemci kaydedicilerine, örneğin bu iki bayt bir bellek adresi oluşturuyorsa bellek adres kaydedicisine yüklenirler. Böylece her bir getirme fazı sırasında program sayıcısı, komutu oluşturan bayt sayısına bak-
S 2.5 maksızın (1,2,3 gibi) uygun bir sayıyla artırılır.

2.5 G/Ç Arabirim Devresi

G/Ç arabirim ünitesi, mikrobilgisayarı, uygulamada kullanılacak giriş çıkış birimlerine bağlamak için gerekli bir veya daha çok giriş çıkış portu sağlar. Bu birimler, mikrobilgisayarın dış dünyası (ya da çevresi) ile etkin biçimde bağlantı kurduğu için, çoğu zaman *çevresel cihazlar/birimler* olarak da ifade edilir.

1. Bölüm'de açıklandığı üzere, bir uygulama için kullanılan giriş ve çıkış birimlerinin türüne bakılmaksızın mikroişlemci, temelde ya ikili bir değerin (genelde 8-bit) bir giriş portundan girişini (okuma) ya da bir çıkış portuna çıkışını (yazma) yapar. Ardından bu değer, ek arabirim fazları aracılığıyla gerçek giriş çıkış birimlerine bağlanmak üzere uygun bir biçime dönüştürülür.

G/Ç arabiriminin mikroişlemciye bağlanması, bellek birimini mikroişlemciye bağlamada da kullanılan bilgisayar yolu aracılığıyla gerçekleşmektedir. Bu nedenle, mikroişlemcinin birbirinden ayrı her birim için veri okuma ve yazma yapabiliyor olması gerekmektedir. Bundan başka, eğer birden çok giriş ve/veya çıkış portu gerekiyorsa, mikroişlemci her bir giriş çıkış işlemi için gereken portu seçebilmelidir.

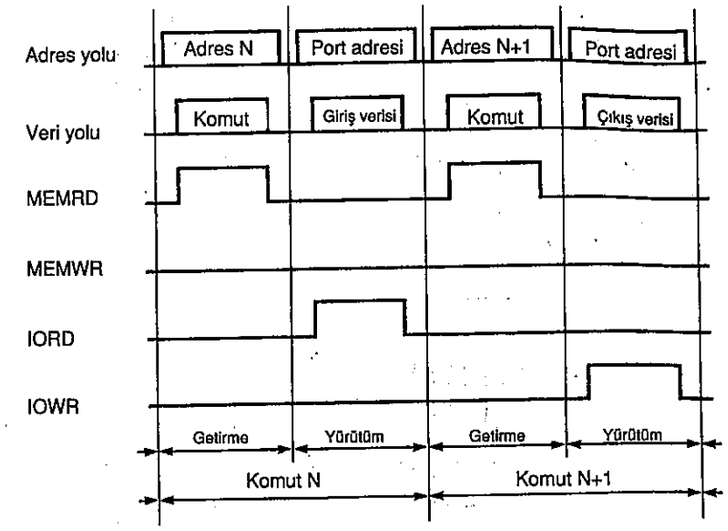


Şekil 2.8 : G/Ç arabirim şeması

Bu seçim, mikroişlemcinin, herhangi bir aktarımda gereken portu belirtmek üzere adres yolunun bir bölümünü kullanması ve herhangi bir giriş veya çıkış komutu yürütülürken, mikroişlemcinin denetim birimi tarafından iki ek denetim sinyali daha üretmesiyle gerçekleşmektedir. Bu sinyallere sırasıyla IORD ve IOWR denir ve komut saykılının yürütme fazı sırasında mikroişlemci tarafından kontrol yoluna çıkarılır. Bu sinyaller, bellek birimi ile G/Ç arabirim ünitesinin, yol üzerindeki bellek ile G/Ç okuma ve yazma işlemlerini ayırt edebilmesini sağlar.

G/Ç arabirimi içindeki her bir port, bir *port tamponu* içerir. Bu tampon, bilgisayar yolunu, denetimli çevresel cihazlar tarafından yapılmakta olan giriş çıkışlardan etkin biçimde ayırır. *Çıkış tamponu*, mikroişlemci tarafından çıkışı yapılan veri değerini bir çıkış birimi tarafından kabul edilmeden önce tutmak -veya diğer deyimle bir mandallamak- için kullanılan elektriksel bir devredir. Aynı şekilde *giriş tamponu* da, bir birim tarafından üretilen ve mikroişlemci tarafından girişininin yapılmasını bekleyen veriyi tutmak için kullanılır.

Dolayısıyla, giriş portundan bir değer okumak için mikroişlemci uygun *portun* adresini adres yoluna yerleştirir ve bir IORD sinyali üretir. G/Ç arabirim ünitesi de,



Şekil 2.9 G/Ç zamanlama diyagramı

buna adreslenmiş portun giriş tamponunun o an içerdiği değeri veri yoluna koyarak karşılık verir ve sonra da bu değer akümülatöre yüklenir.

Benzer biçimde, bir değeri bir çıkış portuna yazmak için mikroişlemci, port adresini adres yoluna, çıkışı yapılacak veriyi de (genellikle akümülatörün o anki içeriği) veri yoluna yerleştirir ve bir IOWR sinyali üretir. G/Ç çevrebirim ünitesi buna, veri yolunda bulunan ve bir çıkış cihazı tarafından alınmayı bekleyen değeri adreslenmiş çıkış portuna bağlı çıkış tamponuna mandallama yapmak suretiyle karşılık verir.

Şekil 2.8'de, iki giriş ve iki çıkış portu bulunan bir G/Ç arabirim ünitesinin şematik diyagramı verilmektedir. Ayrıca bir giriş (IN) ve çıkış (OUT) komut örneklerini gösteren zamanlama diyagramı da Şekil 2.9'dadır.

Şekil 2.8'deki denetim birimi, adres yolundan aktarıma *-adresin kodunu çözmeye-* ilişkin portu belirler ve ayrıca veri yolunu yetkileme ve yetkisizleme işlemlerini ger-
S 2.6 çekleştirilir

Sorular

2.1 12-bitlik adres sözcüğü kullanan 8-bitlik bir mikroişlemcinin maksimum bellek alanını belirleyin. Cevabınızı Kbayt cinsinden ifade edin.

2.2 Bir mikrobilgisayar sistemi, 8 Kbayt PROM ve 256 bayt RAM'e gerek duy-

maktadır. Mikroişlemcinin 16-bitlik adresler kullandığını varsayarak ve ayrık adresler kullanarak, her bir bellek bloğunun başlangıç ve bitiş adreslerini saptayın.

2.3 Bir mikrobilgisayar aşağıdaki adres haritasını kullanmaktadır :
0000 → 07FF ROM
2000 → 23FF RAM
Sistemdeki ROM ve RAM belleklerinin büyüklüklerini bulun.

2.4 Örnek bir mikroişlemcinin şematik blok diyagramını çizin ve her bir bileşenin işlevini açıklayın.

2.5 Ardışık iki komutun yürütülmesi sırasında bilgisayar yolundaki elektriksel sinyalleri gösteren zamanlama diyagramını çizin. İlk komutun belleğe ve ikincisinin de bir G/Ç işlemine ilişkin olduğunu varsayın.

2.6 G/Ç çevresel birimlerin şematik blok diyagramını çizin ve her bir bileşenin işlevini açıklayın.

3. Bölüm : **Mikroişlemci komutları**

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde:*

1. *Makine-düzeyli komutların, işlem kodu ve işlenen parçası olmak üzere iki parçadan oluştuğunu anlayabilmeli,*
2. *Bir komutun işlenen parçasının ya belli bir değeri ya da işlemin yapılacağı değerin konumunu belirttiğini kavrayabilmeli,*
3. *Örneklerin de yardımıyla, sembolik gösterim, işlem kodu kısaltması ve sembolik adres terimlerini açıklayabilmeli,*
4. *Tipik bir komut takımını oluşturan dört değişik grubun her birinden bir komut örneği verip işlevini açıklayabilmeli,*
5. *Mikrobilgisayar sistemlerinde kullanılan dört tür adresleme modunu anlamış olmalı,*
6. *Verinin çeşitli mikroişlemci kaydedicileri ve herhangi bir mikroişlemci kaydedicisi ile bellekteki bir konum arasında nasıl taşındığını, diğer bir deyimle nasıl aktarıldığını gösteren kısa programlar yazıp bunları açıklayabilmeli,*
7. *Sembolik gösterim biçiminde yazılmış bir programı onaltılı makine kodu biçimine çevirmek için bir mikroişlemci üreticisinin kodlama formunu kullana-bilmeli,*
8. *Bir üreticinin tek sistemkartlı mikrobilgisayar sistemiyle sunulan işlevleri değerlendirebilecek düzeye gelmiş olmalısınız.*

3.1 Giriş

Bir önceki bölümde bir mikroişlemcinin, sayısal bilgisayarlar gibi, saklı-program kavramıyla işlem yaptığı, yani mikroişlemcinin, belleğinde saklı bulunan programı yürüttüğü görülmüştü. Program, mikroişlemci tarafından sırayla yürütülen ikili kodlanmış temel makine komutlarından oluşmaktadır.

Bir programı oluşturan komutlar, kullanıcının (ya da *programcının*) istediği uygulama işlevini veya görevi yerine getirmek üzere, belirlenen mikroişlemcinin komut takımı içinden seçilen bir komutlar grubudur. Bu nedenle, makine düzeyli bir program yazmadan önce, normal olarak bir mikroişlemcide mevcut komutların tiplerinin ve bu komutlarla ne gibi işlemler yapılabileceğinin bilinmesi gerekir.

Tipik olarak mevcut komutlar; ikili bir değeri bir G/Ç arabirim portundan okuyan veya ona yazan, veriyi mikrobilgisayarın çeşitli bileşenleri arasında ve mikroişlemcide yer alan değişik kaydediciler arasında taşıyan (aktaran), veri üzerinde işlem, yani aritmetik ve mantıksal işlemler yapan ve son olarak da programın yürütme akışını denetleyen komutlardır.

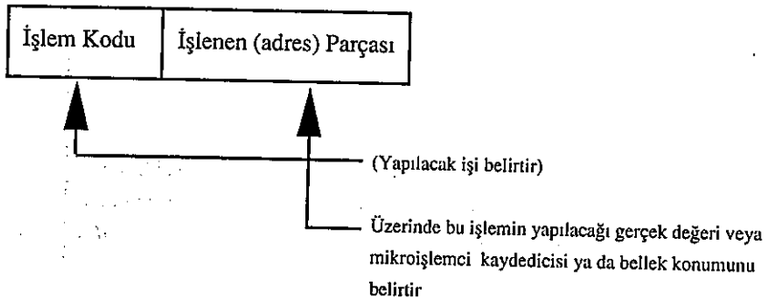
Bu bölümde, makine düzeyli bir komutun yapısı açıklanmakta, çeşitli komut tipleri

ve normalde mikroişlemci sistemlerinde kullanılan adresleme mod örnekleri verilmektedir. Son olarak da, makine düzeyli kısa programlar geliştirmek ve test etmek için üreticiler tarafından sunulan olanaklar açıklanmaktadır.

3.2 Makina düzeyli komutlar

Bir mikroişlemci tarafından yürütülen makine düzeyindeki tüm komutlar belirli bir işlevi veya işlemi yerine getirirler. Tipik olarak bu işlem; ya komutun içinde tanımlanan ya da mikroişlemcinin bir kaydedicisinde veya bellek birimindeki bir saklama gözünde saklanan bir veri parçası ile ilgilidir. Böylece komutların çoğu iki farklı bölümden (veya parçadan) oluşur: Yapılacak belirli bir işlemi belirten bir bölüm olan *işlemkodu* (veya *işkodu*) ile üzerinde bu işlemin yapılacağı veri değeri veya verinin konumunu belirten *işlenen parçası*.

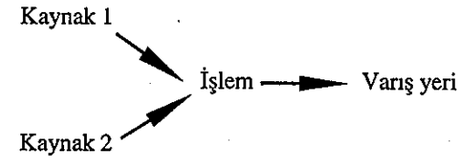
Bu ikincisi sıklıkla bir bellek konumuna ilişkin olduğundan bir komutun ikinci bölümü, *adres parçası* olarak da anılır. Bir makine komutunun yapısı şematik olarak şu şekilde gösterilebilir:



Makine düzeyli bir programı oluşturan komutlar, yürütülmeden önce belleğe saklanırlar. Her bir bellek konumunu oluşturan ikili basamakların (bitlerin) sayısı sabit olduğundan - tipik olarak mikroişlemci sistemleri için 8-bitlik (veya bir baytlık)- bir komutu saklamak için gereken bayt sayısı değişik komut türleri için farklılık gösterecektir. Örneğin; işlenen parçasında bir kaydediciyi belirten bir komut çoğu kez tek bir bayta gerek duyarken, veri değerinin kendi içinde belirtildiği bir komut için iki bayt veya işlenen parçası bir bellek konumunu belirtiyorsa da normal olarak üç bayt gerekir.

Öte yandan, komuttaki bayt sayısına bakmaksızın, komut saykılının getirme fazı sırasında mikroişlemci, komutun ilk baytını okur ve komutu oluşturan (varsa) diğer baytların sayısını ve anlamlarını çıkarır.

Bir komutun işlenen parçası bir adres olduğunda, bu bölümde üç ayrı adres gösterilmiş olacaktır: ilk ikisi işlenen değerlerin bulundukları yerleri belirtmek için - *kaynak adresi*- üçüncüsü de sonucun saklanacağı yeri belirtmek için - *varış yeri adresidir:*



Bu nedenle, gereken adres sayısını azaltmak için birçok sistemde, eğer bir komut iki kaynak adresine gereksinim duyuyorsa, varış yeri adresi kaynak adreslerden biri yapılır. Bundan başka, çoğu zaman işlem kodunda bu adres , örneğin akümülatör ya da A kaydedicisi gibi iç kaydedicilerden biri örtülü bir biçimde belirtilir.

Bu yüzden, bir komutun işlenen parçası adres belirttiğinde, normalde sadece tek bir adrese gerek duyulur ve ikinci adres (kaynak veya varışyeri) işlem kodunda belirtilir. Bu yaklaşım ile her bir komutu ve dolayısıyla tüm programı saklarken bellek alanından önemli miktarda tasarruf elde edilmiş olunur.

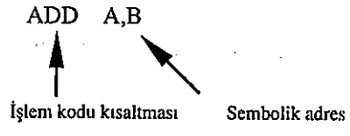
3.3 Sembolik gösterim

Bir makine komutu, mikrobilgisayarın belleğinde saklıyken ikili kodlanmış biçimde olsa da, kağıt üstünde bu biçimde gösterilmesi son derece sıkıcı ve zaman alıcıdır. Çok daha kullanışlı bir yöntem ise, her bir komutu *sembolik gösterim* biçiminde göstermektir.

Sembolik gösterim kullanılarak bir makine komutu şu şekilde gösterilir :

İŞLEM KODU KISALTMASI SEMBOLİK ADRES

İşlem kodu kısaltması, komutun işlem kodu parçasını (dolayısıyla da işlemin yerine getirilecek olduğu sırayı), *sembolik adres* de işlenen veya adres parçasını gösteren kısaltılmış isimlerdir. Örneğin:



ifadesi, ikili kodlanmış

10000000 = 80 (onaltılı)

komutunu göstermek için kullanılır.

Tipik olarak bu komut bir mikroişlemci kaydedicisinin (bu örnekte B kaydedicisi) o anki değerini bir başkasına (A kaydedicisi) eklemek için kullanılır. Dikkat edilirse bu örnekte komut, ilgili iki kaydediciyi de belirtmektedir, fakat birçok mikroişlemcide A kaydedicisi işlem kodu kısaltmasında örtülü biçimde belirtildiği için yalnızca B kaydedicisi komutun içinde belirtilir. Başka bir örnek:

LD A,B

İşlem kodu kısaltması Sembolik adres

ifadesi de, ikili kodlanmış

01111111 = 7F (onaltılı)

komutunu göstermek için kullanılabilir.

Tipik olarak bu komut da bir iç mikroişlemci kaydedicisinin (bu örnekte B kaydedicisinin) o an içerdiği değeri bir başkasına (A kaydedicisine) yüklemek (taşımak) için kullanılır. Benzer biçimde:

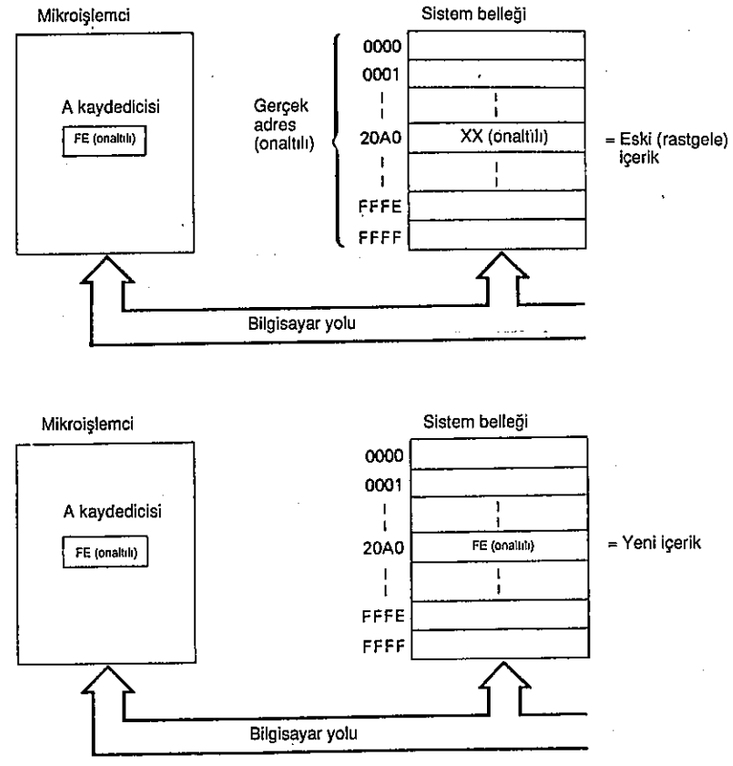
LD (20A0),A

İşlem kodu kısaltması Sembolik adres

ikili kodlanmış şu komutu göstermek için kullanılabilir :

00110010 = 32 (onaltılı) } LD
10100000 = A0 (onaltılı) } 20A0
00100000 = 20 (onaltılı) }

Tipik olarak bu komut, bir mikroişlemci kaydedicisinin (A kaydedicisi) o an içerdiği değeri bellek birimindeki bir konuma [20A0 (onaltılı)] yüklemek (saklamak) için kullanılır. Dikkat edilirse bu örnekte, sembolik adres, gerçek bellek adresini onaltılı



Şekil 3.1 LD (20A0), A komutunun yürütülmesi

biçimde (20A0) belirtmektedir. Bu nedenle, komutun etkisi Şekil 3.1'de gösterildiği gibidir. A kaydedicisinin başlangıç içeriğinin, (sözgelimi bir önceki komutun sonucu olarak üretilen) FE (onaltılı) olduğu kabul edilmiştir.

Yukarıdaki örneklerden de kolayca görülebileceği gibi, sembolik gösterim biçiminde yazılan bir komut çok daha okunaklıdır ve dolayısıyla anlaşılması da daha kolaydır. Öte yandan sembolik gösterme biçimi ile ikili kodlanmış biçim arasında bire-bir karşılık olduğu için dönüştürme çok kolay biçimde yapılabilir. Bu ve bundan sonraki bölümlerde verilen tüm komutlar sembolik gösterim biçiminde yazılacaktır, ancak iki biçim arasında nasıl dönüştürme yapılacağı da açıklanacaktır.

S 3.1

3.4 Komutların sınıflandırılması

Örnek bir mikroişlemci belki de yüz veya daha fazla farklı makine komutu yürütebilir ve bunların tümü, o cihazın *komut takımı* olarak anılır. Bu takımdaki her bir komutun anlamını anlamak yıldırıcı bir iş gibi görünmekle birlikte, neyse ki her bir

komutu şu dört farklı gruptan birinin üyesi olacak biçimde sınıflandırmak mümkündür:

1 Veri taşıma.
2 Veri işleme.
3 Denetim aktarma.
4 Giriş/çıkış.

Veri taşıma grubundaki komutlar ya mikroişlemcideki çeşitli kaydediciler arasında ya da bir mikroişlemci kaydedicisi ile bir bellek konumu arasında veri taşırlar (diğer bir deyimle aktarırlar). Örneğin:

```
LD  A,B
```

komutu, B kaydedicisinin o an içerdiği değeri A kaydedicisine *taşır* (*yükler*). Dikkat edilirse, varışyeri adresi (yani bu örnekte akümülatör veya A kaydedicisi) daima kaynak adresinden (B kaydedicisi) *önce* belirtilir. Veri taşıma komutları bu bölüm içinde daha sonra ele alınacaktır.

Veri işleme grubundaki komutlar ya belirtilen kaydediciyle ya da bellek gözlerindeki verilerle aritmetik ve mantıksal işlemler yaparlar. Örneğin:

```
ADD  A,B
```

A ve B kaydedicilerinin o an içerdiği değerlerin A kaydedicisinin de toplanmasıyla sonuçlanır. A kaydedicisinin sonucu nasıl biriktirdiğine dikkat edin. Bu grup 4. Bölüm'de ayrıntılı olarak işlenecektir.

Denetim aktarma grubundaki komutlar, normalde mikroişlemcinin, programı oluşturan komutların ardışık yürütme sırasını kesmesine ve yürütülecek bir sonraki komutu elde edebilmek için programın bir başka parçasına dallanmasına neden olur. Dolayısıyla,

```
JP  20FF
```

komutu, mikroişlemcinin yürütülecek bir sonraki komutu elde edebilmek için koşulsuz olarak yürütme sırası dışına 20FF (onaltılı) bellek konumuna dallanmasına (atlamasına) neden olur. Görüleceği üzere, denetim aktarma komutları, bir mikroişlemciye ilişkin bir program yazarken çok büyük esneklik sağlar. Bu komutlar 5. Bölümde açıklanmıştır.

Son olarak, *giriş/çıkış* grubundaki komutlar sistemin çeşitli G/Ç portları ile mikroişlemci kaydedicileri (çoğunlukla akümülatör ya da diğer bir deyimle A kaydedicisi) arasında veri taşırlar. Örneğin:


```
OUT  (20),A
```

komutu, A kaydedicisinin içerdiği değerin, onaltılı adresi 20 olan çıkış portuna aktarımına neden olur. G/Ç komutları 6. Bölüm'de ele alınmıştır.

Örnek olması amacıyla bu ve bundan sonraki bölümlerde seçilen komut kısaltmaları, Zilog Z80 mikroişlemcisine aittir. Bu cihaz çok yaygın olarak kullanılmakta ve yapısı ve komut tipleri ile 8-bitlik mikroişlemcilerinin birçoğu için örnek teşkil et-
S 3.2 mektedir.

3.5 Komut adresleme modları

Bir komutta kullanılan kaynak ve varış yeri adresleri belirgin farklılıklar gösterir. Örneğin, veri taşıma komutları kaynak ve varış yeri adresi olarak bir mikroişlemci kaydedicisi veya bir bellek konumu belirtebilirler. Buna alternatif olarak, daha sonra görüleceği gibi, veri taşıma komutları, adresin yerinde gerçek bir veri değeri de belirtebilirler.

Bir komut tarafından kullanılan kaynak ve varış yeri adreslerinin tipi *komut adresleme modu* tarafından belirlenir ve tüm mikroişlemcilerde çeşitli adresleme modları bulunmaktadır. Belli bir mikroişlemci tarafından sağlanması adresleme modları grubu, program yazarken dikkate değer esneklikler sağlanması ve verilen bir görevi yerine getirebilmek için daha az komut gerektirmesi açısından çok önemlidir.

Mikroişlemci sistemlerinde kullanılan dört temel adresleme modu şunlardır :

1 Kaydedici adresleme.
2 Dolaysız adresleme.
3 Doğrudan adresleme.
4 Kaydedici dolaylı adresleme.

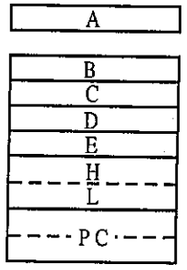
Bunlardan ilk ikisi temel olarak, mikroişlemcideki iç kaydedicilere ilişkin veri taşıma ve veri işleme komutları için kullanılır. Diğer ikisi ise temelde, bir mikroişlemci kaydedicisi ile bellekte bir konumuna ilişkin veri taşıma ve işleme komutları içindir.

Yukarıdaki adresleme modlarının kullanımını göstermek için, veri taşıma grubundan
S 3.3 örnek makine komutları ele alınacaktır.

3.6 Veri taşıma komutları

Veri taşıma grubundaki komutlar mikrobilgisayardaki bir kaydediciden veya bellek konumundan diğerine veri taşıma olanağı sağlarlar. Dolayısıyla, bu grupta verilen komutlar mikroişlemcideki çeşitli kaydediciler arasında ve bir mikroişlemci kaydedicisi ile bellekteki bir konum arasında veri taşımak içindir.

Veri taşıma grubundaki birçok komut mikroişlemcideki çeşitli kaydedicilerle işlem yaptığı veya onların durumlarını değiştirdiğinden, Z80'in temel kaydedicileri bir örnek olmak üzere Şekil 3.2'de gösterilmiştir. Her bir kaydedicinin kullanımı çeşitli makine komutları tanıtıldıkça açıklanacaktır.



Şekil 3.2 Zilog Z80 mikroişlemcisindeki temel kaydediciler :

A	8 bitlik aritmetik veya akümülatör kaydedicisi
B,C,D,E	4 tane 8 bitlik genel amaçlı kaydedici
H,L	normal olarak 16 bitlik bir bellek adres kaydedicisini oluşturmak için kulanılan iki adet 8 bitlik kaydedici
PC	yürütülecek bir sonraki komutu gösteren 16 bitlik bir bellek adresi içeren program sayıcı kaydedicisi

Kaydedici Adresleme

Kaydedici adresleme modunu kullanan komutlar, verileri iç mikroişlemci kaydedicileri arasında taşımak için kullanan komutlardır ve bu nedenle komutun kaynak ve varışyeri adresi, aktarım işlemi ile ilgili olan kaydedicileri belirtir.
Örneğin;

LD B,A

İşlem (yükle) İşlenen (B=varışyeri A=kaynak)

Bu komut, A kaydedicisinin içerdiği değerin B kaydedicisine taşınması (yüklenmesi) ile sonuçlanır. A kaydedicisinin içeriği değişmez. Bu çoğu kez şu şekilde yazılır:

[B]←[A] ; [A] değişmez

(köşeli parantezler "içerdiği değer" i göstermek için kullanılır).

Buna alternatif olarak, bir komutun yaptığı iş, *izleme tablosundaki* bir giriş aracılığıyla gösterilebilir. İzleme tablosundaki her bir giriş, komutun kendisinin yanı sıra, çeşitli kaydedicilerin komut yürütmeden önce ve sonra içerdiği değerleri de kapsar. Bu gösterme biçimi, özellikle iç kaydedicilerle ilgili komutlara ilişkin eylemlerin etkilerini gözlemeye yardımcı olur. Böylece yukarıdaki komut için bir izleme tablosu girişi Şekil 3.3(a)'da gösterilmiştir.

Sembolik komut	Mikroişlemci kaydedicisi içeriği															
	Yürütme öncesi								Yürütme sonrası							
	A	B	C	D	E	H	L	PC	A	B	C	D	E	H	L	PC
LD B,A	xx	E7	xx	xx	xx	xx	xx	xxxx	E7	E7	xx	xx	xx	xx	xx	xxxx
(a) LD B,A																
LD C,B	xx	E7	xx	xx	xx	xx	xx	xxxx	xx	E7	E7	xx	xx	xx	xx	xxxx
(b) LD C,B																
EX DE,HL	xx	xx	xx	20	80	28	00	xxxx	xx	xx	xx	28	00	20	80	xxxx
(c) EX DE,HL																
LD A,7F	xx	xx	xx	xx	xx	xx	xx	xxxx	7F	xx	xx	xx	xx	xx	xx	xxxx
(d) LD A,7F																
LD HL,2800	xx	xx	xx	xx	xx	xx	xx	xxxx	xx	xx	xx	xx	xx	28	00	xxxx
(e) LD HL,2800																
LD DE,2080	xx	xx	xx	xx	xx	xx	xx	xxxx	xx	xx	xx	20	80	xx	xx	xxxx
(f) LD DE,2080																

Şekil 3.3 Bazı değişik veri taşıma komutlarını gösteren bir izleme tablosu

Daha anlaşılır olması için, bu komutla etkilenmeyen kaydediciler, "dikkate alınmaz" durumunu belirtmek üzere xx şeklinde gösterilmiştir. Komutun yürütülmesinden önce B kaydedicisinin içerdiği değerin E7 (onaltılı) olduğu varsayılmış ve dolayısıyla yürütmeden sonra, A kaydedicisinin içeriği E7 (onaltılı) değeri olmuş ve B kaydedicisinin içeriği değişmemiştir.

Bir başka örnek:

LD C,B

B kaydedicisinin içeriğini C kaydedicisine yükle demektir :

[C]←[B]; [B] değişmez

ve böylece bu komut için örnek bir izleme tablosu girişi Şekil 3.3(b)'de gösterilmiştir. Burada da, B kaydedicisinin değerinin yürütmeden önce E7 (onaltılı) olduğu kabul edilmiştir.

Bundan başka, yapılan işlemlere 16-bitlik kaydedici çiftlerinin (örneğin Z80'deki D,E ve H,L kaydedicileri) katılmasına neden olan sınırlı sayıda veri taşıma komutları da vardır. Örnek bir komut;

EX DE,HL

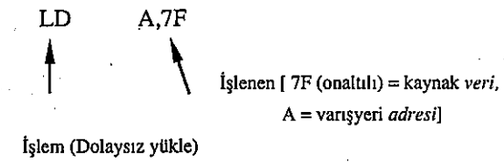
komutu, D,E kaydedici çiftinin içerdiği değerlerle H,L kaydedici çiftinin içerdiklerinin değiş tokuş edilmesine neden olur. Şu şekilde gösterilir:

[D][E] ⟷ [H][L]

dolayısıyla bu komut için bir izleme tablosu girişi Şekil 3.3.(c)'de gösterildiği gibi olabilir. Şekilde, D,E ve H,L kaydedici çiftlerinin başlangıç değerleri 2080 (hex) ve 2800 (onaltılı) olarak kabul edilmiştir. Bu nedenle, komutun yürütülmesinden sonra D,E kaydedici çifti 2800 (onaltılı) ve H,L de 2080 (onaltılı) değerlerini içerecektir.

Dolaysız Adresleme

Bu adresleme modunda, kaynak verinin yerleştirildiği komutun işlenen parçası, bir mikroişlemci kaydedicisini ya da bellek konumunu belirtmek için kullanılmaz. Bunun yerine, gerçek veri değerini belirtir. Böylece, komutun işlem parçasıyla ilgili kaynak veri *komutun kendisi tarafından* içerilir, bu nedenle de verinin *dolaysız* olarak varolduğu söylenir. Veri değeri normalde onaltılı biçimde belirtilir. Örneğin:



Bu örnekteki komut 7F (onaltılı) veri değerinin A kaydedicisine yüklenmesiyle sonuçlanır :

[A] ← 7F (onaltılı) veya [A] ← 01111111 (ikili)

Böylece bu komut için örnek bir izleme tablosu girişi Şekil 3.3(d)'de gösterilmiştir. Dikkat edilmelidir ki, komutu yürütme öncesinde A kaydedicisinin içeriği "dikkate alınmaz" olarak belirtilmiştir, çünkü başlangıçta içerdiği değer dikkate alınmaksızın, yürütme sonrasındaki içeriği daima 7F (onaltılı) değeri olacaktır.

Benzer şekilde, 16-bitlik kaydedici çiftleri de (Z80 için BC, DE veya HL) varışyeri adresi olarak belirtilebilir ve bundan dolayı bu komutlar iki bayt dolaysız veriye gereksinim duyarlar. Örneğin:

LD HL,2800

16-bitlik H,L kaydedici çiftinin 2800 (onaltılı) dolaysız verisi ile yüklenmesiyle sonuçlanır :

[H] ← 28 (onaltılı)
[L] ← 00 (onaltılı)
[H][L] ← 2800 (onaltılı)

Aynı şekilde;

LD DE,2080 ; [D][E] ← 2800 (onaltılı)

ve böylece izleme tablosu girişleri sırasıyla Şekil 3.3(e) ve Şekil 3.3(f)'dedir.

Program Örneği 3.1 : Kaydedici Veri Aktarımı

Şekil 3.4'te gösterilen program örneği, önce dolaysız adresleme ile A kaydedicisine bir değer yüklemek, ardından kaydedici adresleme ile bu değeri diğer kaydedicilere (B,C) yüklemek için yukarıdaki komutların oluşturduğu bir düzenleme kullanır. Daha sonra dolaysız adresleme ile H,L ve D,E kaydedici çiftlerine 16-bit veri değerleri yüklenir. Son olarak da kaydedici adresleme kullanılarak içerilen değerler değiş tokuş edilir.

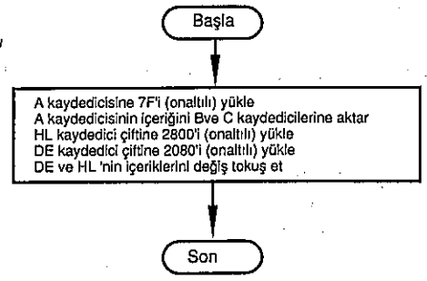
Şeklin *(a)* parçası programın *akış diyagramını* (*akış şemasını*) gösterir; *(b)* ve *(c)* bölümleri program komutlarının ve işlerinin listesinin gösterildiği iki alternatiftir. Bir programın yürütme sırasını veya akışını belirtmede akış diyagramı kullanılımının tam açıklaması 5. Bölüm'de denetim aktarımı komutları ele alınırken verilecektir. Yine de ilerideki bunu takip eden program örneklerinde bir akış diyagramı da eklenecektir; çünkü görüldüğü gibi, akış diyagramlarındaki açıklayıcı bilgi bir programın yaptığı işleri göstermenin uygun bir yoludur.

S 3.4, 3.5

Doğrudan (Genişletilmiş) Adresleme

Doğrudan adresleme, komutla birleşik haldeki veri değerinin sistem belleğinde bulunduğu durumlarda kullanılır. Bu yüzden de, komutun işlenen parçası bir bellek konumunun adresini belirtir. Öte yandan, tüm bellek adresleri 16-bit uzunluğunda olduğundan, adresin belirtilebilmesi için iki bayta gerek vardır ve bu nedenle bu mod çoğunlukla *genişletilmiş adresleme* olarak anılır. Örneğin aşağıdaki komut;

a) Akış diyagramı



b) Program kodu

Sembolik komutlar			İşlem
Komut kısaltması	OP1	OP2	
LD	A	7F	(A) ← 7F (onaltılı)
LD	B	A	(B) ← 7F (onaltılı)
LD	C	B	(C) ← 7F (onaltılı)
LD	HL	2800	(H)(L) ← 2800 (onaltılı)
LD	DE	2080	(D)(E) ← 2080 (onaltılı)
EX	DE	HL	(D)(E) ↔ (H)(L)

c) İzleme tablosu

Sembolik komutlar	Mikroişlemci kaydedici içeriği															
	Yürütme öncesi								Yürütme sonrası							
	A	B	C	D	E	H	L	PC	A	B	C	D	E	H	L	PC
LD A,7F	xx	xx	xx	xx	xx	xx	xx	xxxx	7F	xx	xx	xx	xx	xx	xx	xxxx
LD B,A	7F	xx	xx	xx	xx	xx	xx	xxxx	7F	7F	xx	xx	xx	xx	xx	xxxx
LD C,B	7F	7F	xx	xx	xx	xx	xx	xxxx	7F	7F	7F	xx	xx	xx	xx	xxxx
LD HL,2800	7F	7F	7F	xx	xx	xx	xx	xxxx	7F	7F	7F	xx	xx	28	00	xxxx
LD DE,2080	7F	7F	7F	xx	xx	28	00	xxxx	7F	7F	7F	20	80	28	00	xxxx
EX DE,HL	7F	7F	7F	20	80	28	00	xxxx	7F	7F	7F	28	00	20	80	xxxx

Şekil 3.4 Program Örneği 3.1

LD A,[2800]

2800 (onaltılı) bellek konumunun o anki içeriğinin A kaydedicisine taşınmasına (yüklenmesine) neden olur. Böylece şu şekilde ifade edilir:

[A] ← [2800] ; [2800] değişmez

Benzer şekilde, A kaydedicisinin içerdiği değer belirli bir bellek konumunda saklanabilir. Örneğin aşağıdaki komut:

LD [2800],A ; [28000] ← [A], [A] değişmez

A kaydedicisinin içerdiği değerin 2800 (onaltılı) bellek konumuna saklanmasına neden olur. A'nın içeriği değişmez.

Bir sonraki kısımda görüleceği gibi, H,L kaydedici çifti çoğunlukla birleşik halde 16-bitlik bellek adres kaydedicisi olarak kullanılır ve sonuçta, tek bir komut ve doğrudan adresleme ile, H,L kaydedici çiftinin yüklenebilmesi için iki komut sağlanmıştır. Örneğin:

LD HL,[2800]

komutu, L kaydedicisine 2800 bellek konumunun içeriğinin ve H kaydedicisine de, bellekte bu konuma komşu olan konumun (bu örnekte 2801) içerdiği değerin yüklenmesini sağlar. Dolayısıyla bu komut şu şekilde ifade edilir :

[L] ← [2800]
[H] ← [2801]

Benzer şekilde, H ve L'nin içerdiği değerler iki ardışık bellek konumuna saklanabilir:

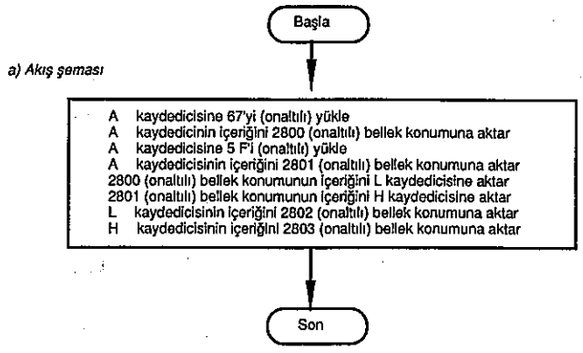
LD [2802],HL ; [2802] ← [L] ; [L] değişmez
[2803] ← [H] ; [H] değişmez

Program Örneği 3.2 : Doğrudan Adresleme

Şekil 3.5'teki program; önce bir veri değerini ardışık iki bellek konumunda saklar. Sonra da H,L kaydedici çiftini bu değer ile yüklemek için dolaysız ve doğrudan adresleme birleşimini kullanır. Son olarak da doğrudan adresleme kullanılarak aynı veri farklı ardışık iki bellek konumuna saklanır.

Kaydedici Dolaylı Adresleme

Kaydedici dolaylı adresleme, bir bellek konumuna yazma (saklama) veya ondan okuma (yükleme) yapmada doğrudan adreslemenin bir alternatifidir. İlerideki bölümlerde görüleceği gibi, bu adresleme, sözgelimi bellekteki bir tabloyu gösteren bir değerler listesine erişimde oldukça uygun ve etkin bir yöntemdir. Sonuçta, bu çok sayıda mikrobilgisayar uygulaması için ortak bir ihtiyaç olduğundan, dolaylı adreslemenin kullanılması, program için gereken bellek



b) Program kodu

Sembolik komutlar	İşlem
LD A,67	(A) ← 67 (onaltılı)
LD (2800),A	(2800) ← 67 (onaltılı)
LD A, 5F	(A) ← 5F (onaltılı)
LD (2801),A	(2801) ← 5F (onaltılı)
LD HL, (2800)	(L) ← (2800), yani 67 (onaltılı) (H) ← (2801), yani 5F (onaltılı)
LD (2802), HL	(2802) ← 67 (onaltılı) (2803) ← 5F (onaltılı)

Şekil 3.5 Program Örneği 3.2

miktarında önemli ölçüde tasarruf elde edilmesini sağlar.

Dolaylı kaydedici adresleme kullanarak veri değerleri, adresi H,L kaydedici çiftinin o anki 16-bit uzunluktaki içeriği olan bellek konumundan okunur ya da bu konuma yazılır. Bu nedenle de komut, bellek adresi içermez. Ancak bunun yerine, H,L kaydedici çiftinin o anki içeriği olan kullanılacak adres, işlem kodunda *örtülü biçimde belirtilir*. Bu nedenle, bellek adresi *dolaylı* olarak elde edilir. Örneğin:

```
LD A,[HL]
```

komutu, A kaydedicisinin, adresi; H,L kaydedici çiftinin o an içerdiği değer olan bellek konumunun içeriği ile yüklenmesine neden olur. Bu şu şekilde gösterilir :

[A] ← [[H][L]]

Benzer biçimde;

```
LD [HL],B ;  [[H][L]] ← [B] ; [B] değişmez
```

komutu, B'nin içeriğinin, H,L kaydedici çiftinde belirtildiği bellek konumuna taşınmasına neden olur.

Ayrıca, dolaysız bir veri değerinin doğrudan bir bellek konumuna saklanmasını sağlayan bir komut daha vardır. Burada da bellek adresi H,L kaydedici çiftinde saklı bulunmaktadır. Böylece:

```
LD [HL],AA   ; [[H][L]] ← AA (onaltılı)
```

komutu; AA (onaltılı) veri değerinin, adresi H,L kaydedici çiftinde belirtilen bellek konumuna saklanmasına neden olur.

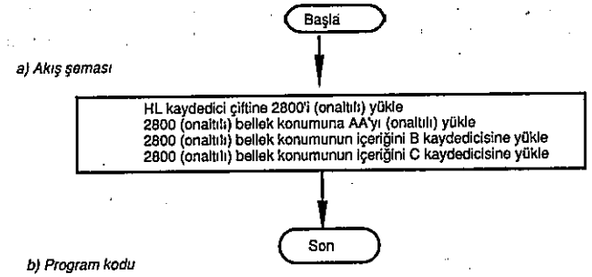
Program Örneği 3.3 : Kaydedici Dolaylı Adresleme

Şekil 3.6'daki program, dolaysız adresleme ve kaydedici dolaylı adresleme birleşimini kullanır. Önce dolaysız adresleme ile bellek adresi H,L kaydedici çiftine yüklenir. Sonra kaydedici dolaylı adresleme kullanılarak, bu bellek konumuna bir değer saklanır. Son olarak da, yine kaydedici dolaylı adresleme ile bu değer başka iki kaydediciye yüklenir.

S 3.6, 3.7

3.7 Zilog Z80 komut takımı

Şimdiye dek tanıtılan komutlar, kesinlikle veri taşıma grubundaki yegane komutlar değildi. Aslında, bunlar tüm komut listesinin bir örneği olacak biçimde verilmiştir. Örneğin veri taşıma grubunda, normal olarak kaydedici çiftleri arasında veri taşımayı sağlayan komutlar vardır (Z80 için A,B,C,D,E,H ve L). Ayrıca dolaysız veri



b) Program kodu

Sembolik komutlar	İşlem
LD HL,2800	(L) ← 00(onaltılı) (H) ← 28 (onaltılı)
LD (HL), AA	(2800) ← AA (onaltılı)
LD B, (HL)	(B) ← (2800), yani (B) ← AA (onaltılı)
LD C, (HL)	(C) ← (2800), yani (C) ← AA (onaltılı)

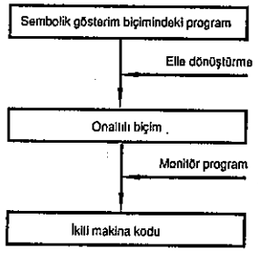
Şekil 3.6 Program Örneği 3.3

bu kaydedicilerden herhangi birine yüklenebilir.

Bu ve bundan sonraki bölümlerin temel amacı, her bir komut grubundan, anlamları ve uygulama alanları ile birlikte, komut örnekleri tanıtmaktır. Böylece okuyucu hem her bir komut grubunun işlevini anlayabilecek, hem de belli bir görevi yerine getirmede gereken komutları gruptan seçebilecektir. Z80'e ait daha kapsamlı bir komut listesi EK-1'de verilmiştir.

3.8 Çeviri işlemi

Bir mikroişlemci, program belleğinde ikili kodlanmış biçimde saklı bulunan komutları yürütür. Bu nedenle de bir program, yukarıdaki örnekler de dahil, yürütülmeden önce, sembolik gösterim biçimden ikili kodlanmış biçimdeki eşdeğerine dönüştürülmelidir. Bu yüzden büyük geliştirme sistemlerinden küçük tek sistem kartlı prototipleme sistemlerine kadar tüm mikrobilgisayar sistemleri, normalde sembolik gösterim biçiminde (kaynak program) yazılmış programları, çalıştırılabilir ikili koda (amaç programa) dönüştürmek için tasarlanmış bir takım programları (sistem programları) işin içine katarlar.



Şekil 3.7 Çeviri işlemi

Öte yandan bazı küçük mikrobilgisayar prototipleme sistemlerinde yerleşik işletim programı (normalde monitör olarak adlandırılır) kaynak programın dönüştürme işlemini tümüyle yerine getiremez; bunun yerine, her bir komut önce programcı tarafından ara bir onaltılı biçime dönüştürülmelidir. Monitör, sonra, her bir onaltılı karakter çiftini 8-bitlik ikili desen eşdeğerine çevirir. Bu işlem Şekil 3.7'de gösterilmiştir.

Görüldüğü gibi, sembolik gösterim biçiminden onaltılık biçime çeviri işlemi programcı tarafından sık sık yapılması gereken bir işlemdir. Bu nedenle, üreticinin sembolik makine komutları listesi normal olarak her komutun onaltılı kod karşılığını da içerir. Örneğin, EK-1'de gösterilen Z80 komutları listesi aynı zamanda her komutun onaltılı eşdeğerini de kapsamaktadır.

Elle Dönüştürme

EK-1'den kolayca anlaşılacağı gibi, her komutun onaltılık biçimde gösterilmesi için 1-3 bayta gerek vardır. Örneğin;

LD B,A ; [B]◄— [A]

komutu tek bir bayta gereksinim duyar:

4 7	İşlem

LD A,7F ; [A]◄— 7F (onaltılı)

komutunun iki bayta ihtiyacı vardır :

3 E	İşlem
7 F	dolaysız veri, yani 7F (onaltılı)

ve

LD HL,2800 ; [[H][L]]◄— 2800 (onaltılı)

için üç bayt gerekir :

2 1	İşlem
0 0	En küçük değerlikli bayt = [L]
2 8	En büyük değerlikli bayt = [H]

Bir komutun işlenen parçası 16-bitlik (2 bayt) bir değeri gösterdiğinde, daima ilk olarak en küçük değerlikli baytın (yukarıdaki örnekte 00) belirtildiğine dikkat edilmelidir.

Kodlama sayfaları

Her komut için onaltılı koda ek olarak, birçok üretici kullanıcıya program yazımında ve dönüştürme işleminde yardımcı olması amacıyla kodlama sayfaları da sağlarlar. Şekil 3.8'de, bir üretici program kodlama sayfası örneği gösterilmiştir.

Program örneği 3.1 'in programı gösterilmiştir. Görülebileceği gibi, programın sembolik gösterim biçimine ilişkin açıklamalar daha önce olduğu gibi sıralanmıştır. Her bir komutun yanında belirtilen ETİKET alanı dallanma komutları için kullanıldığından, bu örnekte boş bırakılmıştır. Bu formun kullanımı bu bölümde daha sonra izah edilecektir.

Programın bellekte 2000 (onaltılı) adresinden başlayarak saklandığı varsayılmıştır;

bu yüzden de komutlar, onaltılı biçimlerine dönüştürülürken, bu değerden başlayarak bir sonraki ardışık bellek adreslerine atanmıştır. Her komuta karşılık gelen onaltılı biçimleri EK-1'den elde edilebilir.

3.9 Bir programın yüklenmesi ve çalıştırılması

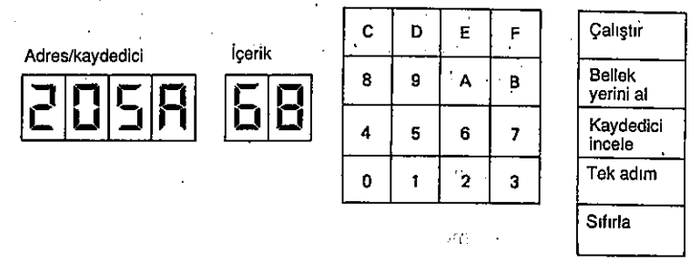
Birçok mikroişlemci uygulaması için sonuç sistemdeki program, sözgelimi bir sadece-okunur bellekte sürekli olarak saklanacak olsa da, programı, geliştirme ve test fazları sırasında kolayca yükleyebilmek, çalıştırabilmek ve gerekiyorsa düzeltebilmek oldukça yararlı olacaktır. Bu nedenle de üreticiler, normal olarak esas mikrobilgisayarın yanında, onaltılı biçime çevrilmiş küçük programları yüklemek, çalıştırmak ve test etmek için gerekli olanakları içeren tek sistem kartlı mikrobilgisayar sistemleri sunarlar.

S 3.8

Esasen tek sistem kartlı bir sistem içinde; bir mikroişlemci, sistem monitör programını tutmak için bir sadece-okunur bellek (ROM), test edilecek program ile bu programla ilgili veri değerlerini saklamak için bir rastgele-erişimli bellek (RAM) bulunur. Programı girebilmek için bir onaltılı tuştakımı ve program testi boyunca seçilmiş bellek konumlarının ve kaydedicilerin içeriklerini görüntülemek için de bir

Bellek		Sembolik Komut				Açıklamalar
Adres	İçerik	Etiket	Kom. Kısıt.	OP1	OP2	
2000	3E					
2001	7F		LD	A	7F	(A) ← 7F (onaltılı)
2002	47					
			LD	B	A	(B) ← (A)
2003	4F					
			LD	C	B	(C) ← (B)
2004	21					
2005	00		LD	HL	2800	(H)(L) ← 2800(onaltılı)
2006	28					
2007	11					
2008	80		LD	DE	2080	(D)(E) ← 2080(onaltılı)
2009	20					
200A	EB					
			EX	DE	HL	(D)(E) ← (H)(L)

Şekil3.8 Üreticinin program kodlama sayfası



Şekil 3.9 Tek bir kartlı sistemin tuştakımı ve ekranı

onaltılı ekran yer alır. Son olarak, monitör programına komutlar girmek için ek tuşlar vardır. Bu komutlarla ilgili açıklamalar ve kullanımları aşağıda verilmiştir.

Monitör Komutları

Tipik bir tuştakımının yerleşim düzeni ile tek sistem kartlı bir sistemle birlikte kullanılacak bir onaltılı ekran Şekil 3.9'da gösterilmiştir. Buradan da görülebileceği gibi tuştakımında, 16 adet onaltılı tuşa ek olarak beş tane monitör komut tuşu bulunmaktadır.

Komutların monitör programına girilmesi, seçilen komutun tuşuna basılarak yapılır, monitörün verdiği karşılıklar onaltılı ekranda görüntülenir. Görüntülenen tüm basamaklar, onaltılı biçimdedir ve iki parçaya ayrılmıştır : adres/kaydedici alanı ve veriler (veya içerik) alanı. Bu nedenle, örneğin şekilde örnek bir bellek adresi olarak 205A (onaltılı) ve bunun içeriği olarak da 68 (onaltılı) değeri görülmektedir.

Şekilde görülen beş monitör komutu tek sistem kartlı bir sistem için örnek teşkil eder. Bunlar; sıfırla, bellek yerini al, çalıştır, kaydediciyi incele, tek adım (veya izleme)'dir.

Sıfırla Bu komut kullanıcıya, monitör programını yeni baştan yürütme olanağı verir ve bu komuttan sonra monitör, başka komutları okumaya hazır durumdadır. Normal olarak sisteme ilk enerji verildiğinde veya kullanıcının sistemin denetimini yeniden almak istediğinde kullanılır.

Bellek Yerini Al Bu komut kullanıcıya ardışık bellek konumlarını inceleme ve gerekiyorsa içeriğini değiştirme olanağı verir. Dolayısıyla bu komut özellikle bir kullanıcının programını RAM'e girerken çok yardımcı olmaktadır. Kullanıcı önce programı sembolik gösterim biçiminde yazar, kodlama sayfalarının da yardımıyla bunu onaltılı biçime dönüştürür ve bu komutu kullanarak onaltılık formu RAM'e yükler.

Bu komut aynı zamanda kullanıcıya, eğer yürütme sırasında programda hatalar bulunursa, belirli RAM konumları -dolayısıyla da program komutları- üzerinde değişiklik yapabilme olanağı verir.

Çalıştır Bir program yazılıp, kodlandıktan ve belleğe yüklendikten sonra çalıştır komutu ile yürütülür (çalıştırılır). RAM'da saklı bulunan bir programı çalıştırmak için, ÇALIŞTIR tuşuna basılır ve ardından da programın ilk komutunun bellekteki 4-adresi girilir. Bu giriş onaltılı tuştakımı kullanılarak yapılır.

Program bir kez çalışmaya başladığında, monitörün sistem denetimini yeniden ele alabilmesi için ya SIFIRLA tuşuna basılmalıdır ya da program içinde, mikroişlemciyi monitör programının başına dallandıracak bir komut çalıştırılması gerekir.

Kaydediciyi İncele Bu komut kullanıcıya her bir kaydedicinin içeriğini görüntülemeyi ve gerekiyorsa değiştirme imkanı verir. Bu komut, 'Tek Adım' komutu ile birlikte kullanıldığında özellikle yararlı bir komuttur; çünkü kullanıcıya, kullanıcı programının yürütülmesi sırasında mikroişlemcinin durumunu görüntüleme olanağı tanımakta, dolayısıyla olası program hatalarını belirlemede kullanılabilmektedir.

Bir program RAM'e yüklendiğinde, yukarıda da belirtildiği gibi, çoğu kez çalıştır komutu ile birlikte yürütülür. Öte yandan, eğer program hatalar içeriyorsa ve istenen görevi yerine getiremiyorsa, hatalı komutların bulunması ve bu yolla düzeltilmesi gerekir. Bu işlem, çoğunlukla zor ve sistem yardımı olmazsa oldukça zaman alan bir iştir.

Tek Adım (izleme) Bu komut izlemeye ilişkin bir sistem yardımıdır; çünkü program komutları yürütülürken kullanıcıya tüm sistemin - kaydediciler, bellek konumları vs. - durumunu inceleme olanağı tanır. Böylece istenen sonucu üretmeyen program komutları kolayca bulunabilir.

Bellekte saklı program içinde adım adım gitmek için önce 'Tek Adım' tuşuna basılır ve ardından bellekteki ilk komutun adresi girilir. Bu, işlemcinin ilk komutu çalıştırmasını sağlar. Fakat yürütmeye devam etmek yerine, program beklemeye alınır ve monitör programı yeniden çalıştırılır.

Şimdi denetim monitörde olduğu için, devam etmeden önce kullanıcı, eğer isterse, bir önceki komutun doğru çalıştığını doğrulamak için istediği kaydedici ve / veya bellek konumlarının içeriğini inceleyebilir. Komutun doğru olarak çalıştığının görülmesinden sonra kullanıcı, tekrar Tek Adım tuşuna basarak bir sonraki komutu yürütebilir.

Sorular

3.1 Aşağıdaki terimlerin anlamını açıklayın:
a) Sembolik gösterim.
b) İşlem kodu kısaltması.
c) Sembolik adres.

3.2 Örnekler yardımıyla aşağıdaki her bir komut grubundan bir komutun yaptığı görevleri açıklayın:
a) Veri taşıma.
b) Veri işleme.
c) Denetim aktarma.
d) G/Ç.

3.3 Örnekler yardımıyla mikrobilgisayar sistemlerinde kullanılan aşağıdaki adresleme modlarının anlamlarını açıklayın:
a) Kaydedici adresleme.
b) Dolaysız adresleme.
c) Doğrudan adresleme.
d) Kaydedici dolaylı adresleme.

3.4 Bir izleme tablosu aracılığıyla aşağıdaki program komutlarının yaptığı işleri belirtin ve buradan da program yürütüldükten sonra A,B,C,D,E,H,L kaydedicilerinin içeriklerini bulun:

```
LD   DE,2800
LD   A,D6
LD   B,D
LD   C,E
LD   HL,28FF
EX   DE,HL
```

3.5 Aşağıdaki işleri yapacak bir program yazın:
a) FF (onaltılı) dolaysız veri değerini A kaydedicisine yükleyen,
b) 86 (onaltılı) dolaysız veri değerini B kaydedicisine yükleyen,
c) A kaydedicisinin içeriğini D kaydedicisine yükleyen,
d) B kaydedicisinin içeriğini E kaydedicisine yükleyen,
e) 28A6(onaltılı) dolaysız veri değerini H,L kaydedici çiftine yükleyen,
f) D,E kaydedici çiftinin içeriğini H,L kaydedici çiftininkiyle değiştokuş eden.

3.6 Aşağıdaki işleri yapacak bir program yazın:

a) AA (onaltılı) dolaysız veri değerini A kaydedicisine yükleyen,
b) Doğrudan adresleme ile A kaydedicisinin içeriğini 2800 (onaltılı) bellek konumuna yükleyen,
c) 00 (onaltılı) dolaysız veri değerini A kaydedicisine yükleyen,
d) 2800 (onaltılı) dolaysız veri değerini H,L kaydedici çiftine yükleyen,
e) Dolaysız adresleme ile 2800 (onaltılı) bellek konumun içeriğini A kaydedicisine yükleyen.

3.7 2800 (onaltılı) ve 2801 (onaltılı) bellek konumlarının içeriklerini aşağıdaki program çalıştıktan sonra bulun:

```
LD   B,EE
LD   C,FF
LD   HL,2801
LD   A,B
LD   [2800],A
LD   [HL],C
```

3.8 Yukarıdaki programları Şekil 3.7'dekine benzer bir kodlama sayfası üzerinde sıralayın ve EK-1'deki onaltılı kodları kullanarak her bir programa ilişkin onaltılı kodları türetin.

4. Bölüm : **Aritmetik ve mantıksal komutlar**

Bu bölümün amaçları : *Bu bölümü bitirdiğinizde:*

1 *Sayıların, ikiye tümleyen biçiminde gösterilmesini anlamış olmalı ve bu biçimde gösterilen sayılarla toplama ve çıkarma işlemi yapabilmeli,*
2 *Bir mikroişlemcide bulunan farklı türdeki toplama ve çıkarma komutlarını öğrenmeli ve bu komutları kullanarak programlar yazıp onları anlayabilmeli,*
3 *Bir bayrak kaydedicisinde kullanılan çeşitli bayrak bitlerini öğrenmeli ve elde ile yardımcı elde bayrağı uygulamasını tanımlayabilmeli,*
4 *Sayıların ikili kodlanmış ondalık biçiminde ifadesini anlamış olmalı ve bu biçimde gösterilen sayılarla toplama ve çıkarma işlemi yapabilmeli,*
5 *Ondalık ayarlama komutunun işlevini biliyor ve bu komutu kullanarak bir program yazabilmeli ve bir programı anlayabilmeli,*
6 *Bir mikroişlemcide bulunan AND(VE), OR (VEYA) veya XOR (ÖZEL VEYA) mantık komutlarını öğrenmiş olmalı ve her bir komutun uygulamasını açıklayabilmeli,*
7 *Kaydırma ve karşılaştırma komutlarıyla gerçekleştirilen işlemleri biliyor olmalısınız.*

4.1 Giriş

Mikroişlemciler birçok farklı uygulamada kullanılabilir. Bu nedenle, uygulamaya bağlı olarak, mikrobilgisayarda saklanan ikili veriler de birçok farklı parametreyi gösterebilir. Bir uygulamada, örneğin, bir kaydedicinin veya bellek konumunun 8-bit uzunluktaki içeriği, (+62, -27 gibi) işaretli bir değer veya sayıyı gösterirken; bir diğerinde de, sözgelimi, sekiz adet denetimli iki-durumlu birimin durumunu gösterebilir (mantıksal 1=açık, 0=kapalı).

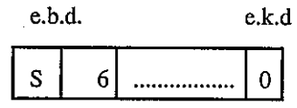
Bu nedenle, bu veriyi işlerken, yapılabilecek tipik bir işlem, işaretli bir değerle benzer bir başka değerin toplanması veya çıkarılması olabilir; öte yandan, eğer saklı değer sözgelimi sekiz adet denetimli birimin durumuna ilişkin ise; buradaki tipik bir işlem de, belli bir bitim (dolayısıyla da birimin) mantıksal 1 (açık) veya 0 (kapalı) durumunda olup olmadığının test edilmesi olabilir.

Bu yüzden veri işleme grubundaki komutlar sadece toplama çıkarma gibi normal aritmetik komutlarını değil, aynı zamanda VE, VEYA gibi mantık komutlarını da içerirler. Bunlar, daha sonra da görüleceği gibi, programcıya 8-bitlik bir dizinin içindeki ikili bitleri tek tek test edip işleme imkanı sağlar. Bu nedenle, mantık komutları özellikle ikili bitleri tek tek işleyen uygulamalarda çok kullanışlıdır.

4.2 İşaretli sayı gösterme

Bir mikroişlemcide bulunan değişik aritmetik komutlarını incelemeye geçmeden önce, bir mikrobilgisayarda işaretli sayıların nasıl saklandığını anlamak gerekir. Kullanılan standart yöntem, *ikinin tümleyeni biçiminde gösterimdir*; çünkü, daha sonra da görüleceği gibi, bu yöntemi kullanarak saklı veri ile aritmetik işlemler yapılırken, sonucun işareti otomatik olarak üretilir. Bu da, aritmetik-mantıksal birimi (ALU) oluşturan elektronik devrenin oldukça basitleştirilmesini sağlar.

İkiye tümleyen biçimindeki gösterimde her bir ikili sayının en büyük değerlikli biti, S, sayının işaretini göstermek için kullanılır. Böylece 8-bitlik bir sistemde sayılar şu şekilde gösterilir :

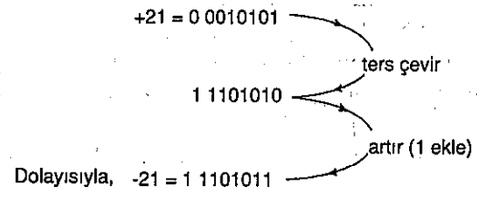


S= sıfır ve pozitif sayılar için 0
S= negatif sayılar için 1

Pozitif sayılar için geriye kalan yedi bit, sayının değerini gösterir. Örneğin:

S	S^6	S^5	S^4	S^3	S2	S^1	S^0	= ağırlıkları (2'nin üssü)
0	0	0	1	0	1	0	1	= +21 (ondalık)
0	1	0	0	1	0	1	0	= +74
0	1	1	0	0	0	0	1	= +97

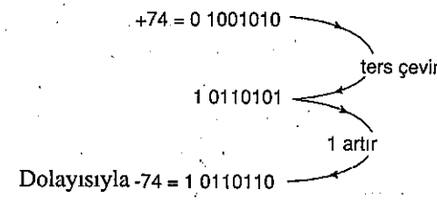
Negatif sayılar için geriye kalan yedi bit, sayının, ikiye tümleyeni biçimindeki gösteriminin değerini gösterir. Bir sayının ikiye tümleyenini bulmak için ikili sayı önce tamamen (işaret biti de dahil) çevrilir (tümlenir), sonra da bulunan ikili değer bir artırılır. Örneğin:



Tablo 4.1 : *8-bit ikiye tümleyen sayıların aralığı (2'nin üssü)*

Ondalık sayı	*İkilinin tümleyeni biçimi*
+127	0 1111111
+126	0 1111110
.	.
.	.
.	.
+3	0 0000011
+2	0 0000010
+1	0 0000001
0	0 0000000
–1	1 1111111
–2	1 1111110
–3	1 1111101
.	.
.	.
.	.
–127	1 0000001
–128	1 0000000

Benzer biçimde :



Dikkat edilirse tümleme işleminden sonra en büyük değerlikli bit (işaret biti) otomatik olarak 1 olmaktadır. Buradan, 8-bitlik bir sistem için mümkün olan sayı aralığı Tablo 4.1'de gösterilmiştir. En büyük pozitif 8-bitlik sayının +127, en küçük negatif sayının da -128 olduğuna dikkat edin.

S 4.1 , 4.2

4.3 Aritmetik komutlar

Elektronik bir hesap makinesi daima, işaretli sayılar üzerinde işlem yapar. Bu nedenle de, toplama ve çıkarma işlemlerine ek olarak, pek çok hesap makinesi, çarpma bölme ve hatta başka karmaşık aritmetik fonksiyonları da yapabilir. Öte yandan birçok mikroişlemci uygulamasında, işlenen ikili değerler, sayısal-olmayan bilgileri gösterebildiği halde, mikroişlemcilerin çoğunda yalnızca toplama ve çıkarma gibi temel aritmetik komutları yer alır. Bu, pek çok 8-bit mikroişlemci uygulamasında çarpma ve bölme işleminin tek bir komut kullanarak yapılamadığı anlamına gelir. Bunun yerine, programcı bu işlemleri yapabilmek için bir grup ya da bir dizi komut kullanmak zorundadır. Şimdi temel aritmetik komutlar olan toplama ve çıkarmayı inceleyelim.

Toplama işlemi

Tablo 4.2 *İki tane ikili bitin toplanması*

Bit 1	*Bit 2*	*Toplam*	*Elde*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

İki tane ikili sayının (bitin) toplanması Tablo 4.2'de özetlenmiştir. Tüm olası giriş birleşimlerini -bu örnekte Bit 1 ve Bit 2- buna bağlı (toplam ve elde) çıkışlarını listeleyen şekil, *doğruluk tablosu* olarak adlandırılır.

Bu tablodan kolayca çıkarılabileceği gibi, toplam biti yalnızca

(mantıksal) 1 veya 0 olduğundan, iki tane 1 toplandığında toplam biti 0 olur ve bir *elde biti* üretilir. Elde biti, bir sonraki üst sıralı bit çiftine eklenmelidir. Dolayısıyla toplama işlemi yaparken, bir önceki bit çiftinin toplamından (varsa) üretilmiş elde bitinin de (elde girişi) toplama eklenmesi gerekir. Böyle baştan sona bir toplama işlemi Tablo 4.3'te verilen doğruluk tablosunda özetlenmiştir.

Şimdi Tablo 4.3'teki bilgilere de başvurarak, iki tane tam ikili sayının toplamını almak mümkündür. Aşağıdaki örnekte iki pozitif sayının, A ve B'nin, pozitif bir sonuç vermek üzere toplanması gösterilmektedir:

```
A       = 0 0101011 = + 43 (Ondalık)
B       = 0 0110010 = + 50 (Ondalık)
Elde giriş  = 0 1000100
Elde çıkış (Eçık)  ──┐   ┌── Elde giriş (Egir)
Elde çıkış  = 0 0100010
A + B       = 0 1011101 = + 93(Ondalık)
```

Daha önce de belirtildiği gibi, negatif sayıları ikinin tümleyeni biçiminde göstererek, aynı toplama işlemi yapılabilir ve sonuç ikinin tümleyeni biçiminde elde edilebilir. Bu belki de örneklerle daha iyi bir şekilde gösterilebilir : İlk örnek, negatif bir sonuç vermek üzere bir negatif bir de pozitif sayının toplanmasını; ikincisi pozitif bir sonuç vermek üzere bir negatif ve bir pozitif sayının toplanmasını; üçüncüsü de negatif bir sonuç vermek üzere iki negatif sayının toplanmasını göstermektedir.

```
A = 0 0101011 =  + 43
B = 1 1001110 =  - 50
Elde giriş      = 0 0011100
Eçık (Elde çıkış)  ──┐   ┌── Egir (Elde giriş)
Eçık (Elde çıkış) = 0 0001110
A + B = 1 1111001 =  - 7

A = 1 1010101 =  - 43
B = 0 0110010 =  + 50
Elde giriş      = 1 1100000
Eçık (Elde çıkış)  ──┐   ┌── Egir (Elde giriş)
Elde çıkış      = 1 1110000
A + B = 0 0000111 =  + 7
```


Tablo 4.3 : *İki bitle elde girişi bitinin toplanması*

Bit 1	*Bit 2*	*Elde girişi*	*Toplam*	*Elde Çıkışı*
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

```
A = 1 1010101 =  - 43
B = 1 1001110 =  - 50
Elde giriş      = 1 0111000
Eçık (Elde çıkış)  ──┐   ┌── Egir (Elde giriş)
Elde çıkış      = 1 1011100
A + B = 1 0100011 =  - 93
```

Negatif sayıların ondalık eşdeğerlerinin sağlaması, ikili sonuçlar üzerinde tümleme işlemi ile yapılabilir.

Toplama komutları

Normalde, bir mikroişlemcide bulunan toplama komutları A kaydedicisinin içerdiği değer ile ya kaydedici adresleme, ya dolaysız adresleme, ya da kaydedici dolaylı adresleme kullanılarak belirlenen ikinci bir değeri dikkate alır. Şimdi bu adresleme modlarından her birini kullanan Z80 komut takımından bazı örnekler verilecektir :

Kaydedici Adresleme : Kaydedici adreslemeyi kullanan bir komuta örnek aşağıda verilmiştir :

```
ADD A,B
```

Bu komut B kaydedici içeriğinin A kaydedici içeriğine eklenmesini sağlar. A kaydedicisinin o an içerdiği değerin yerine çıkan sonuç konur, B kaydedicisinin içeriği

değişmez. Bu durum aşağıdaki gibi gösterilir :

[A] ← [A] + [B] ; [B] değişmez

Dolaysız Adresleme : Dolaysız adreslemeyi kullanan bir örnek aşağıda verilmiştir :

ADD A,8E

Bu komut, içerdiği kapsanan dolaysız veri değerinin (bu örnekte 8E (onaltılı)) A kaydedicisinin o anki içeriğine eklenmesine neden olur, yani:

[A] ← [A] + 8E (onaltılı)

Kaydedici Dolaylı Adresleme : Kaydedici dolaylı adreslemeyi kullanan bir örnek aşağıdaki verilmiştir :

S 4.3(a) ADD A,[HL]

Bu komut, adresi HL kaydedici çiftinde belirtilen bellek konumunun içeriğinin A kaydedicisinin o an içerdiği değere eklenmesiyle sonuçlanır. Bellek konumunun içeriği değişmez.

[A] ← [A]+[[H][L]] ; [[H][L]] değişmez

Çıkarma işlemi

İki tane ikili basamak arasındaki çıkarmayı en iyi şekilde açıklamak için de ondalık sistemde iki sayı arasındaki çıkarma işlemini incelemek daha yararlı olacaktır. Aşağıdaki örneği ele alalım:

```
A =  465
B = -173
      //
Borç =  100
A-B =  292
```

İkinci rakam ilkinden küçük veya ona eşitse, çıkarma işlemi doğrudan yapılır, borç alma gereği duyulmaz (örneğin 5 eksi 3). Eğer ikinci rakam ilkinden büyükse (bu örnekte 6 eksi 7) bir sonraki üst sıralı basamaktan bir borç almak gerekir. Dolayısıyla buna *borç-girişi* denir. Ondalık sistemde borç alınan değer o basamaktaki rakamın on fazlasına eşittir. Böylece bu basamaktaki çıkarma işlemi 9 fark sonucunu verir (10+6)-7 olur ve borç, bir sonraki basamağa taşınır. Bu nedenle buna da *borç çıkışı* denir ve ikinci rakam (bu örnekte 1), birinci rakamdan (bu örnekte 4'ten) çıkarılmadan önce eklenmelidir.

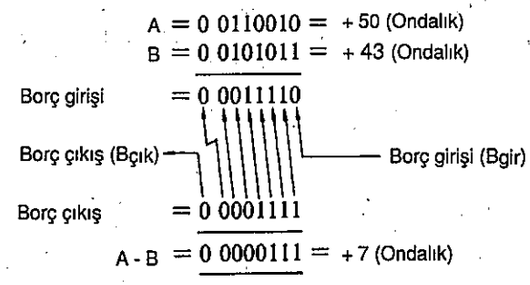
Tablo 4.4 : *İki bit ve borç-girişi biti ile çıkarma*

Bit 1	*Bit 2*	*Borç-girişi*	*Fark*	*Borç-çıkışı*
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

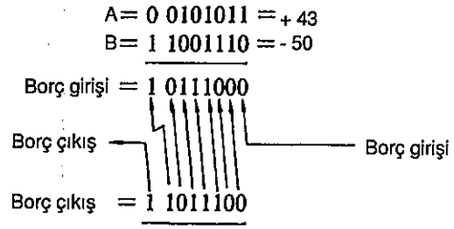
Özetlemek gerekirse, iki rakam arasında çıkarma yaparken, bir önceki basamaktan gelecek olası borç girişlerini dikkate almak gerekir. Ayrıca, eğer ikinci rakamla borç girişinin toplamı ilk rakamdan büyükse bir borç çıkışı üretilmelidir.

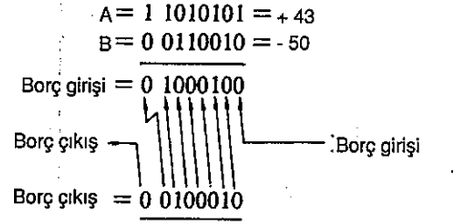
İki tane ikili sayı arasındaki çıkarma ve olası bir borç girişi Tablo 4.4'teki doğruluk tablosunda gösterilmiştir. Önce Bit 2 ve borç girişi toplanır ve daha sonra da bu toplama Bit 1'den çıkarılır. Eğer bu toplam Bit 1'den büyükse bir borç çıkışı gerekir. Bu, ikili sistemde iki bite eşittir ve böylece çıkarma işleminden önce Bit 1'e eklenir.

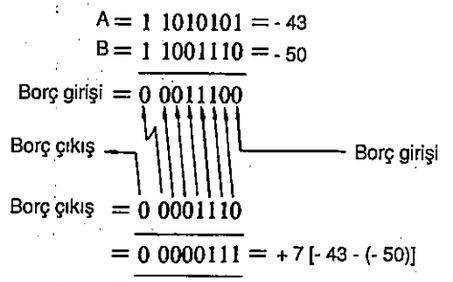
Böylece Tablo 4.4'teki bilgilere başvurarak, iki tane ikili sayı arasında çıkarma işlemi yapmak mümkündür. Bu durum aşağıda gösterilmiştir :



Bu örnekte sonucun pozitif çıkması için, ilk sayı olan A, ikinci sayı olan B'den özellikle büyük seçilmiştir. Öte yandan daha önce de belirtildiği gibi, ikinin tümleyeni biçiminde gösterilen iki sayı ile de benzer bit-bit çıkarma işlemi yapılabilir ve sonuç ikinin tümleyeni biçiminde ve işaretli olarak elde edilir. Bu durum aşağıdaki örneklerde gösterilmiştir :







Çıkarma komutları

Bir mikroişlemcide bulunan çıkarma komutları, daha önce açıklanan toplama komutlarıyla benzerlik taşır. Çıkarma komutlarında birinci değer A kaydedicisinin içeriğidir, ikinci değer ise kaydedici dolaysız ya da kaydedici dolaylı adresleme kullanılarak belirtilir. Mikroişlemci yukarıda tanımlanan bit-bit çıkarma işlemine eşit bir işlemi yerine getirir. Aşağıda Z80 komut takımından bazı örnekler yer almaktadır:

Kaydedici Adresleme:

SUB B

Yani:

[A]←[A]-[B] ; [B] değişmez

Dolaysız adresleme:

SUB 8E

Yani:

[A]← [A]-8E (onaltılı)

Kaydedici Dolaylı Adresleme:

SUB [HL]

Yani:

[A]← [A]-[[H][L]] ; [[H][L]] değişmez

Artırma ve azaltma

Mikroişlemcilere ilişkin birçok uygulama programında yer alan yaygın bir işlem de, bir mikroişlemci kaydedicisinin ve bellek konumunun yürürlükteki içeriğini bir artırmak veya azaltmaktır. Dolayısıyla, bu işlemlerin dolaysız adresleme kullanılarak yapılması mümkünse de pek çok mikroişlemcide bu işlevi yerine getiren özel komutlar bulunmaktadır. Örneğin:

INC A

komutu A kaydedicisinin yürürlükteki değerinin bir artırılmasını sağlar. Yani:

[A]← [A]+1

Benzer şekilde;

DEC B

komutu B kaydedicisinin içeriğinin bir azaltılmasına neden olur:

[B]← [B]-1

Özellikle bellekte saklı bir grup değere ulaşmada ve onları işlemede yararlı olan bir komut da şudur :

INC HL

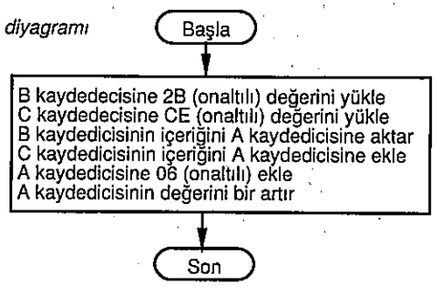
Bu komut H,L kaydedici çiftinin (bellek adres kaydedici) birleşik değerinin bir artırılması sonucunu verir :

[H][L] ← [H][L]+1

Program örneği 4.1 : Toplama Komutları

Şekil 4.1'deki program örneği, daha önce tanımlanan toplama komutlarının B ve C kaydedicilerinde saklı iki tane ikiye tümleyen biçiminde işaretli değer üzerindeki sonuçlarını göstermektedir.

a) Akış diyagramı



b) Program kodu

Sembolik komutlar		İşlem	
LD	B,2B	(B) ← 2B (onaltılı), yani	$+43_{10}$
LD	C,CE	(C) ← CE (onaltılı), yani	-50_{10}
LD	A,B	(A) ← (B), yani (A)	43_{10}
ADD	A,C	(A) ← F9 (onaltılı), yani	-7_{10}
ADD	A,06	(A) ← FF (onaltılı), yani	-1_{10}
INC	A	(A) ← 00 (onaltılı), yani	0_{10}

Şekil 4.1 Program örneği 4.1

Daha önce de belirtildiği gibi, mikroişlemci komutlarda belirtilen bit-bit toplama işlemini yapar. Çıkan sonuçtaki ikili desenleri anlamlı ikiye tümleyen biçiminde işaretli değerler olarak yorumlayan kullanıcıdır.

Program örneği 4.2 : Çıkarma Komutları

Şekil 4.2'deki program örneği, yukarıda açıkladığımız çıkarma komutlarının B ve C kaydedicilerine yüklenmiş olan ikiye tümleyen biçimindeki işaretli iki değer üzerinde ne gibi sonuçlar oluşturduğunu göstermektedir. Her çıkarma işleminden sonra sayıların işaretlerini doğru olarak yorumlamaya dikkat edin.

Program örneği 4.3 : Karışık Aritmetik

Şekil 4.3'teki program örneği; toplama ve çıkarma komutlarının iki tane ikinin tümleyeni biçiminde işaretli değer üzerindeki sonuçlarını göstermektedir. Burada gö-
S 4.3(b) rülen sonuçları kontrol etmek için bit-bit toplama ve çıkarma işlemlerini uygulayın.

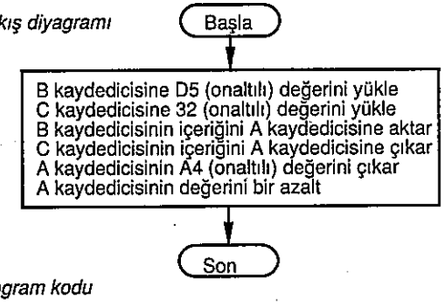
4.4 Bayrak kaydedicisi

Mikroişlemci çok yönlü bir birimdir, çünkü üretildikten sonra her türlü uygulama işlevini gerçekleştirmek üzere kullanılabilir. Kullanıcı, seçilen birimde mevcut komut takımından uygun bir komutlar grubunu (program) seçer ve bu program, istenen uygulama işlevini yerine getirmek için çalıştırılır.

Farklı kullanıcılar mikroişlemcide saklı ikili verileri farklı şekillerde yorumlayabileceklerinden, mikroişlemci üreticileri birimde geniş bir olanaklar dizisi sunarlar. Böylece, geniş bir kullanım esnekliği sağlanmış olur. Bu olanaklardan biri de aritmetik mantık birimindeki (ALU) *bayrak (veya durum) kaydedicisidir.*

Bayrak kaydedicisi durum veya koşul bitleri grubudur. Her bit, belirli kurallara göre, bir aritmetik komut yürütüldüğünde, otomatik olarak ya mantıksal 1'e ya da mantıksal 0'a kurulur. Programcı, daha sonra mikrobilgisayarda saklı olan veriyi istenen

a) Akış diyagramı

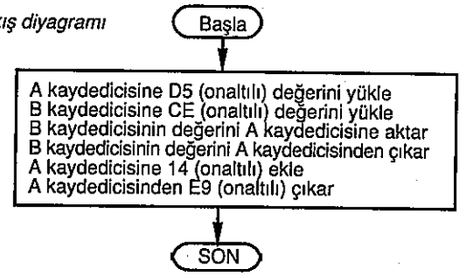


b) Program kodu

Sembolik komutlar		İşlem	
LD	B,D5	(B) ← D5 (onaltılı), yani	-43_{10}
LD	C,32	(C) ← 32 (onaltılı), yani	$+50_{10}$
LD	A,B	(A) ← (B), yani (A)	-43_{10}
SUB	C	(A) ← A3 (onaltılı), yani	-93_{10}
SUB	A4	(A) ← 01 (onaltılı), yani	1_{10}
DEC	A	(A) ← 00 (onaltılı), yani	0_{10}

Şekil 4.2 Program örneği 4.2

a) Akış diyagramı



b) Program kodu

Sembolik komutlar		İşlem	
LD	A,D5	(A) ← D5 (onaltılı), yani	-43_{10}
LD	B,CE	(B) ← CE (onaltılı), yani	-50_{10}
ADD	A,B	(A) ← A3 (onaltılı), yani	-93_{10}
SUB	B	(A) ← D5 (onaltılı), yani	-43_{10}
ADD	A,14	(A) ← E9 (onaltılı), yani	-23_{10}
SUB	E9	(A) ← 00 (onaltılı), yani	0_{10}

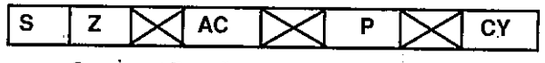
Şekil 4.3 Program örneği 4.3

biçimde işleyebilmek için seçilmiş bayrakların durumunu yorumlayan komutları gözden geçirir veya kullanır.

Örnek olarak Şekil 4.4'te, Z80'in bayrak kaydedicisini oluşturan bayrak bitlerinden bazıları gösterilmiştir. Bu bayrakların kullanımı bu ve bundan sonraki bölümde ele alınacaktır. Fakat daha önce burada, söz konusu aritmetik komutların bayraklar üzerindeki etkisi verilecektir.

İşaret bayrağı (S) : Bir aritmetik komutun sonucu negatif, yani ikiye tümleyen biçimindeki gösterimde A kaydedicisinin en büyük değerlikli biti 1 olduğunda, bu bayrak mantıksal 1'e kurulur. Değilse, komuttan sonra bayrak sıfırlanmış olarak bırakılır.

*Sıfır bayrağı (Z) :*Bu bayrak, bir aritmetik komutun sonucu A kaydedicisini sıfır yaptığı her zaman mantıksal 1'e kurulur; değilse bayrak sıfırlanır.



S= İşaret Bayrağı
Z= Sıfır bayrağı
AC= Yardımcı elde bayrağı
P= Eşlik bayrağı
CY= Elde bayrağı

Şekil 4.4 Bazı Z80 bayrakları

Yardımcı elde bayrağı (AC) : Bu bayrak BCD gösterim biçiminde kullanılır ve bu bölümde daha sonra ele alınacaktır. Bu bayrak bir aritmetik işlem; A kaydedicisinin en küçük değerlikli yarısından bir elde çıkışı ürettiği zaman kurulur.

Eşlik bayrağı (P) : Bu bayrak, bu bölümde daha sonra ele alınacak olan mantıksal işlemlerle birlikte kullanılır. Bayrak; mantıksal bir komut A kaydedicisininde çift sayıda 1 üretiyorsa kurulur; aksi halde, sıfırlanır.

Elde bayrağı (CY) : Bu aslında A kaydedicisinin en büyük değerlikli bitinin elde çıkışıdır. Bu nedenle de A kaydedicisinin en büyük değerlikli bitinden bir elde veya borç üreten toplama veya çıkarma komutlarından sonra kurulur. Temel kullanımlarından biri aşağıda izah edilecektir.

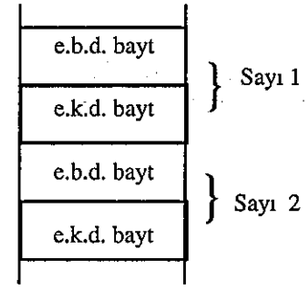
4.5 Elde bayrağı

Bir 8-bitlik mikroişlemcideki toplama ve çıkarma komutlarının büyük çoğunluğu aritmetik işlemleri 8-bitlik değerlerle yaparlar. Dolayısıyla bunlar, ikiye tümleyen biçimindeki ikili sayıları göstermede kullanıldıklarından, 8-bitlik bir sistemde yer alabilecek olan en büyük sayı aralığı (verilen alt ve üst sınırlar da dahil olmak üzere) +127'den -128'e kadardır.

Birçok mikroişlemci uygulaması için bu yeterlidir, çünkü yapılan gerçek işlem miktarı çoğu zaman oldukça azdır. Ne var ki, bazı uygulamalarda, bu yeterli değildir ve kullanıcı, birinci olarak her bir değeri tutmak (saklamak) için bellekte birden çok bayt imkanı tanıyan ve ikinci olarak da, bu değerleri işlerken, belli sayıdaki adımda yapabilen bir düzenlemeye gitmelidir.

Bir uygulamada işaretli sayıların +32,767 / -32,768 aralığında gösterilmek istendiğini (yani her sayı için 16-bit veya 2 bayt gerekmektedir) ve kullanılmakta olan mikroişlemcinin 16-bitlik aritmetik komutları içermediğini varsayalım.

İlk olarak, mikrobilgisayar belleğinde saklı her sayı için iki bayt ayrılmalıdır :



İkinci olarak; mikroişlemci tek bir komutla yalnızca, 8-bitlik değerler arasında toplama ve çıkarma yapabildiğinden, her bir toplama ve çıkarma işlemi iki adımda yerine getirilmelidir : Önce baytların en küçük değerlikli çiftleri, sonra da en büyük değerlikli çiftleri toplanmalı/çıkarılmalıdır. Öte yandan buna ek olarak, baytların ikinci (en büyük değerlikli) çiftleri arasında toplama/çıkarma yapılırken, baytların ilk (en küçük değerlikli) çifti işlendiğinde üretilebilecek elde/borçları da dikkate almanın gerekliliği kolayca görülebilir.

Bu nedenle de mikroişlemciler normalde iki toplama ve çıkarma komut alternatifi sunarlar : İlki daha önce açıklanan ve ikincisi de elde (borç) bitini dikkate alan komutlardır. Örneğin Z80'de:

```
ADC A,B
```

Bu komut B kaydedicisinin içeriğinin *ve* yürürlükteki elde bitinin (bayrağının) A kaydedicisinin yürürlükteki içeriğine eklenmesiyle sonuçlanır. Bu nedenle de yukarıdaki örnekte, baytların ilk çiftinin toplanması bir elde çıkışı üretirse, yukarıdaki elde ile topla (ADC) komutu kullanarak bu elde, baytların diğer çiftine eklenebilir. Yani;

$$[A] \leftarrow [A] + [B] + [CY]$$

Benzer biçimde;

```
SBC A,B
```

komutu da B kaydedicisinin içeriğinin ve elde (borç) bayrağının içeriğinin A kaydedicisinin içeriğinden çıkarılmasıyla sonuçlanır. Yani:

$$[A] \leftarrow [A] - [B] - [CY]$$

Program örneği 4.4 : Eldeli Toplama

Şekil 4.5'teki program örneği eldeli toplama komutunun kullanımını göstermektedir. Program, aşağıda alt alta verilen dört ardışık bellek konumunda saklı iki 16-bitlik (2 bayt) sayıyı birbirine ekler :

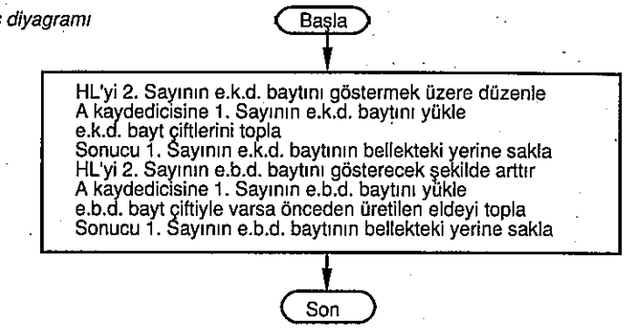
2800 = e.k.d.
2801 = e.b.d. } Sayı 1

2802 = e.k.d.
2803 = e.b.d. } Sayı 2

Baytların ilk çifti normal ADD komutu kullanılarak, ikinci çifti ise ADC komutu kullanılarak birbirine eklenir. Sonuç, bellekteki ilk sayının yerini alır.

Burada, Z80'in sınırlı sayıda da olsa 16-bitlik (2 bayt) aritmetik komutları içerdiği belirtilmelidir. Dolayısıyla, eğer bu komutlar kullanılsaydı, yukarıdaki örnek Program daha az sayıda komutla gerçekleştirilebilirdi. Öte yandan bu örneğin amacı, çok baytlı sayıların 8-bit mikroişlemci ile nasıl işlenebileceğini göstermektir. Benimsenen bu yaklaşım herhangi bir uzunluktaki sayılar için uygulanabilir.

a) Akış diyagramı



b) Program kodu

Sembolik komutlar		İşlem
LD	HL,2802	(H)(L) 2. Sayının e.k.d. baytını gösterir.
LD	A,(2800)	(A) ← 1. Sayının e.k.d. baytı
ADD	A,(HL)	e.k.d. baytlar toplanır
LD	(2800),A	e.k.d. bayt toplamı 1. Sayının e.k.d. baytının yerini alır
INC	HL	(H)(L) 2. Sayının e.b.d. baytını gösterir
LD	A, (2801)	(A) ← 1. Sayının e.a. baytı
ADC	A,(HL)	e.b.d. baytlarla elde toplanır
LD	(2801),A	e.b.d. bayt toplamı 1. Sayının e.b.d. baytının yerini alır

Şekil 4.5 Program örneği 4.4

4.6 İkili kodlanmış ondalık aritmetiği

Daha önce ele alınan aritmetik komutları; tamamen ikili biçimde saklanan veriler üzerinde yapılan aritmetik işlemlere yöneliktir. 1.Bölüm'de belirtildiği gibi, mikrobilgisayara girecek tüm bilginin işlenmeden önce bu biçime, çıkışı için de içeride saklı ikili sonuçların istenen çıkış biçimine dönüştürülmesi gerekmektedir.

Daha önce belirtildiği gibi, mikrobilgisayar tarafından işlenen veriler çok farklı kaynaklardan gelebilir. Örneğin, eğer giriş ve buna bağlı çıkış, sürekli değişen analog sinyallerse, analog giriş sinyallerini ikili biçime dönüştürmek için bir analog-sayısal dönüştürücü (ADC) kullanılır. Aynı şekilde, mikrobilgisayardan çıkan ikili değerleri analog biçimine dönüştürmek için de bir sayısal-analog dönüştürücü (DAC) kullanılır.

Benzer biçimde, eğer veriler klavyeden girilirse, mikroişlemci tarafından işlenmeden önce, mikrobilgisayarın giriş portunda (bulunan bir klavye kodlayıcı devreden) üretilen ikili-kodlanmış veri ikili biçime dönüştürülmelidir. Bu nedenle de, bu zor bir iş olmasa da ve küçük bir program parçasıyla (komut listesi) yapılabilse de, eğer onu izleyen veri üzerinde yapılacak toplam işlem miktarı az ise, mikroişlemcinin işleme gücünün dikkate değer bir oranı bu dönüştürme işlemini yapmak için harcanabilir.

Bu nedenle, pek çok mikroişlemci tamamen ikili kodlanmış olarak değil, ikili kodlanmış ondalık biçiminde saklanan verileri işlemek için sınırlı sayıda imkanlar sunar. Eğer bir uygulama giriş için basit bir sayısal tuştakımı ve çıkış için de ondalık basamaklardan oluşan bir ekran kullanıyorsa, bunlar özellikle yararlıdır. Böylece, bir ara dönüştürme gerektirmeden mikrobilgisayara girilen ve saklanan bilgi işlenebilir ve ardından çıkışı yapılabilir.

BCD Şeklinde Gösterme

Basit bir (ondalık) sayısal tuştakımından çıkışlar, normal olarak 4-bitlik ikili kodlanmış değerlerdir. Bu değerler, tek bir ondalık basamağı (0-9) gösterdiği için ikili kodlanmış ondalık değerler olarak anılırlar. BCD gösterilişi aslında 1.Bölüm'de tanıtılmış olan onaltılık sistemin bir altkümesidir. Tablo 4.5'te, on tane ondalık basamağın 4-bitlik BCD kodu eşdeğerlerinin listesi verilmiştir.

Tablo 4.5

Ondalık Basamak	*BCD kodu*
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

BCD gösterilişi ile her ondalık basamak için 4 bit gerektiğinden bir mikroişlemcide saklanan 8-bitlik bir ikili değer iki ondalık basamağı gösterebilir. Böylece, bazı saklı 8-bitlik desenler ile bunların eşdeğeri ondalık sayılar şöyledir :

0101 0111 = 57 (ondalık)
1000 1001 = 89 (ondalık)
0001 0000 = 10 (ondalık)
0010 0100 = 24 (ondalık)

BCD Toplama

Yukarıdaki açıklamadan da çıkarılabileceği gibi, mikrobilgisayar belleğinde saklı 8-bitlik bir ikili değer ya -128 ile +127 aralığında ikiye tümleyen biçimindeki işaretli bir değeri ya da 00-99 aralığında iki BCD basamağını gösterebilir. Öte yandan, daha önce ele alınan tüm aritmetik komutlar verilerin ikiye tümleyen biçiminde ikili olarak saklandığını kabul eder. Bu yüzden, BCD basamaklarını gösteren saklı değerleri işlerken, BCD biçiminde doğru sonuçları üretebilmek için normal ikili sonuçlarda bazı ayarlamalar yapmak gerekmektedir. Bu birkaç örnek ile daha iyi biçimde gösterilebilir.

Her biri iki BCD basamağını gösteren iki adet 8-bitlik ikili değeri, A ile B'yi ele alalım :

A = 0011 0100 = 34 BCD
B = 0010 0001 = 21 BCD

A+B = 0101 0101

Doğru BCD
toplamı ; 0101 0101 = 55 BCD

Bu örnekte görüldüğü gibi, normal bit-bit toplama işleminin ürettiği ikili toplam, BCD biçimdeki doğru sonucu verir. Öte yandan aşağıdaki toplamayı ele alalım :

A = 0111 0110 = 76 BCD
B = 0001 0101 = 15 BCD

A+B = 1000 1011

Doğru BCD
toplamı ; 1001 0001 = 91 BCD

Açıkça görüldüğü gibi, bit-bit toplama işlemi ile üretilen toplam, şimdi istenen doğru BCD toplamından farklıdır; çünkü normal ikili işlemin verdiği toplam, en küçük değerlikli dört bitte dokuzu aşan bir sonuç verir.

Benzer biçimde, şu toplamayı ele alalım :

A = 0100 1000 = 48 BCD
B = 0100 1001 = 49 BCD

A+B = 1001 0001

Doğru BCD
toplamı = 1001 0111 = 97 BCD

Burada da normal ikili toplamanın sonucu yanlıştır, çünkü bit-bit toplama işlemi doğal olarak en küçük değerlikli dört bitten bir elde çıkışı üretse de, sonuçtaki dört bit yine de doğru değildir.

Uygulamada, normal toplama işlemi tarafından üretilen sonuçtan elde edilen düzeltilmiş BCD toplamı üretebilen dikkate değer bir yöntem vardır. Yöntem, yukarıdaki örnekten kolayca çıkarılabilir ve aşağıda özetlenmiştir:

> Eğer her bir 4-bit grubunun ikili toplamı dokuzdan küçük bir sonuç üretirse *ve* en küçük değerlikli BCD basamakları birbirine eklenirken elde üretilmezse, sonuç doğrudur. Fakat eğer normal ikili toplamanın sonucu dokuzdan büyükse *veya* en küçük değerlikli iki BCD basamağının toplamı elde üretirse, normal ikili toplama sonucuna düzeltme payı olarak +6 eklenmelidir.

Bu yöntemin etkisini göstermek için az önce verilen örnekleri ele alalım. İlk örnekte her bir 4-bitlik grupta üretilen sonuç dokuzdan küçük olduğu için, ikili toplam doğru sonucu verir ve düzeltme yapmak gerekmez. Öte yandan ikinci örnekte, iki en küçük değerlikli BCD basamağının toplamı dokuzdan büyük bir sonuç ve üçüncü örnekte de iki en küçük değerlikli basamak toplanırken bir elde üretilmiştir. Bu yüzden son iki örneğin düzeltilmiş toplamları aşağıdaki gibi türetilir :

1 Düzeltilmemiş toplam (A+B) = 1000 1011
+6 düzeltme payı ekle + 0000 0110

Düzeltilmiş BCD toplamı = 1001 0001 = 91 BCD

2 Düzeltilmemiş toplam (A+B) = 1001 0001
+6 düzeltme payı ekle + 0000 0110

Düzeltilmiş BCD toplamı = 1001 0111 = 97 BCD

Kısım 4.4'te belirtildiği gibi, bayrak kaydedicisindeki bayraklardan biri de yardımcı elde bayrağı (AC) olarak bilinir. Bu bayrak aslında, özellikle BCD sayı gösterim biçimini kullanırken yarar sağlar, çünkü en küçük değerlikli 4-bit grubunun elde çıkışıdır. Bundan başka, bir mikroişlemci komut takımında normalde yukarıdaki düzeltmeyi otomatik olarak yerine getirecek bir komut vardır. Komut, yürütüldüğünde, yukarıdaki toplama işlemini, ya A.Y. bayrağının durumuna ya da normal toplama işleminin ürettiği sonuca göre yapar. Z80'de buna ondalık düzeltme akümülatörü (DAA) denir, çünkü normal ikili toplamanın sonucu otomatik olarak akümülatörde bırakılır.

Aslında, DAA komutu, normal 8-bit ikili bir toplamın her iki yarısını da, yani hem en küçük değerlikli hem de en büyük değerlikli 4-bitlik grupları düzeltmek üzere tasarlanmıştır. Daha önceki elde bayrağı ve çoklu-bayt sayılarının toplanması ile ilgili açıklamalarda olduğu gibi, bu komut da ikiden daha fazla sayıda basamaktan oluşan BCD sayılarını işleme olanağı sağlar. DAA komutunun etkisi aşağıdaki program örneğinde verilmiştir.

Program örneği 4.5 : BCD Toplama

Tablo 4.6'daki program örneği üç tane 8-bitlik sayıyı toplar. Her sayı önce bir mikroişlemci kaydedicisine yüklenir ve her biri, iki BCD basamağını gösterir. Her normal ikili toplamadan sonra 8-bitlik sonuç değerinin DAA komutları ile düzeltildiğine dikkat edin. Sonuçtaki toplam A kaydedicisindedir.

BCD Çıkarma

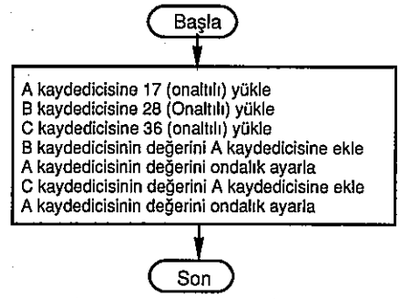
Normal ikili çıkarma işleminin ürettiği sonuçtan doğru BCD farkını üretmek için gereken yöntem az önce toplama işlemi için açıklanandan farklıdır. Tablo 4.6'daki düzeltme tablosunda da görüldüğü gibi, gerekli olan düzeltmeler biraz daha karmaşıktır. Tablo, her bir normal ikili çıkarma komutu yürütüldükten *sonra* A kaydedicisinin içerdiği değer ile iki elde bayrağının durumunu gösterir. Aşağıdaki örneği ele alalım :

A = 0111 0110 = 76 BCD
B = 0010 1000 = 28 BCD
A-B = 0100 1110 CY = 0, AC = 1

Düzeltmeler tablosuna bakarak CY bayrağı 0, en büyük değerlikli 4-bit grubu 4, AC bayrağı 1 ve en küçük değerlikli 4-bit grubu da E (onaltılı) değerindedir. Böylece *eklenecek* düzeltme payı FA (onaltılı) olmaktadır:

A-B = 0100 1110
Düzeltme payını ekle + 1111 1010 = FA (onaltılı)
Düzeltilmiş BCD farkı = 0100 1000 = 48 BCD

a) Akış diyagramı



b) Program kodu

Sembolik komutlar		İşlem
LD	A,17	(A) ←17 (onaltılı), yani 17 BCD
LD	B,28	(B) ←28 (onaltılı), yani 28 BCD
LD	C,36	(C) ←36 (onaltılı), yani 36 BCD
ADD	A,B	(A) ←3F (onaltılı)
DAA		(A) ← 45 (onaltılı), yani 45 BCD
ADD	A,C	(A) ←7B (onaltılı)
DAA		(A) ←81 (onaltılı), yani 81 BCD

Şekil 4.6 Program örneği 4.5

Tablo 4.6 *BCD farkını üretecek düzeltmeler*

Elde (CY)	*En büyük değerlikli 4-bit grubu*	*Yardımcı elde (AC)*	*En küçük değerlikli 4-bit grubu*	*Eklenecek düzeltme payı*
0	0-9	0	0-9	00(onaltılı)
0	0-8	1	6-F	FA
1	7-F	0	0-9	A0
1	6-F	1	6-F	9A

Mikroişlemcilerin sunduğu ondalık-ayar komutlarının büyük çoğunluğu doğru ayarlamayı yalnızca toplama işleminden sonra yaparlar ve bu nedenle de bu makinelerle çalışırken Tablo 4.6'da sıralanan kurallara göre gerekli düzeltme tipini çıkarıp, ona göre uygun düzeltmeleri yerine getirmek kullanıcının (programcının) sorumluluğundadır. Yalnızca çok az bir grup mikroişlemcide (örneğin, Z80) ondalık-ayarla komutu, hem toplama hem de çıkarma işleminden sonra uygun düzeltme yapabilir. Z80 bir önceki aritmetik komutunun toplama mı çıkarma mı olduğunu etkin biçimde anımsar ve bir DAA komutu yürütüldüğünde ona ilişkin uygun düzeltmeyi gerçekleştirir.

S 4.4 - 4.6

4.7 Mantık komutları

Elektronik hesap makinelerinden farklı olarak bazı mikroişlemci uygulamaları sayısal değerlerle ilgili değildir. Çoğu zaman bir G/Ç arabirim portundan okunan 8

bitlik değer bir sayıyı değil, sözgelimi, sekiz tane sayısal ünitenin durumunu gösterir.

Bu yüzden, bu veriyi işlerken, bazı yaygın işlemler şunlar olabilir: "açık (mantıksal 1) veya kapalı (mantıksal 0) olduğuna karar vermek için belli bir bitin (dolayısıyla da cihazın) test edilmesi", "değerin içinde yer alan herhangi bir bitin (dolayısıyla da cihaz durumunun) değişip değişmediğinin belirlenmesi", vs. Bu tip işlemleri yerine getirmek için kullanılan komutlar mantıksal VE (AND), VEYA (OR) ve Özel VEYA (XOR) komutlarıdır. Burada her biri ayrı ayrı ele alınacaktır.

Tablo 4.7 *Mantıksal VE (AND)işlemi*

Bit 1	*Bit 2*	*Bit1 VE Bit2*
0	0	0
0	1	0
1	0	0
1	1	1

Mantıksal VE (AND)

Mantıksal VE komutu, özellikle *maskeleme işlemi* olarak adlandırılan işlemi uygulamada kullanışlıdır. Bu, 8-bitlik bir gruptaki istenmeyen bitleri etkin biçimde olarak *maskeleyen* ve böylece sözgelimi, gruptaki belirli bir biti sınamak için bir mekanizma sağlayan bir işlemdir.

Tablo 4.7'de, iki tane ikili basamakla mantıksal VE işleminin doğruluk tablosu gösterilmiştir. Şekilden kolaylıkla çıkarılacağı gibi, Bit 1 ve Bit 2'nin her ikisi de 1 olduğunda, çıkış mantıksal 1'dir. Bunun dışındaki durumlarda çıkış mantıksal 0'dır.

Bir mikroişlemcideki mantıksal VE komutu, birbirine paralel sekiz tane VE işlemine eşdeğer bir işlem yapar; çünkü A kaydedicisinin içerdiği değer ile öteki herhangi bir mikroişlemci kaydedicisi veya bellek konumu veya doğrudan bir veri değeri arasında bit-bit mantıksal VE işlemini yerine getirmektedir.

Örneğin A ve B kaydedicilerinin değerlerinin sırasıyla B3 (onaltılı) ve D8 (onaltılı) olduğunu varsaydığımızda:

AND B

komutunun sonucu aşağıdaki gibi olacaktır :

(A) = 1011 0011 = B3 (onaltılı)
(B) = 1101 1000 = D8 (onaltılı)

(A) VE (B) = $\overline{1001\ 0000}$ = A'nın içerdiği yeni değer

Önce en küçük değerlikli bit çiftleri VE'lenir, sonra diğer çift, vb. Sadece aynı basamaktaki her iki değer de mantıksal 1 olduğu zaman çıkışın da 1 olduğuna dikkat edin.

Benzer biçimde A kaydedicisinin değerinin 6B (onaltılı) olduğunu varsaydığımızda:

AND 08

komutunun sonucunda

(A) = 0110 1011 = 6B (onaltılı)
Doğrudan değer = 0000 1000 = 08 (onaltılı)
(A) VE 08 = 0000 1000 = A'nın içerdiği yeni değer

ortaya çıkacaktır.

Bu örnekte, dolaysız veri değerinde 0 olan her basamağa karşılık sonuçtaki ilgili basamağın da 0 olduğuna dikkat edin. Öte yandan 1 olduğu yerde de çıkış, A'daki ilgili bitin durumuna göre belirlenir : Eğer bit 1 ise (örnekte olduğu gibi), çıkış 1 olacak; eğer bit 0 ise çıkış da 0 olacaktır.

Dolayısıyla buradan şu sonuç çıkarılabilir: 8-bitlik bir değerin içinde yer alan belirli bir bitin durumunu; ilkin bu değerle test edilecek bit konumundaki mantıksal 1 değerine sahip dolaysız veri değeri arasında mantıksal VE işlemi yapılmak suretiyle test etmek mümkündür. Eğer sonuçta elde edilen A kaydedicisinin değeri 0 ise bit de 0'dır; eğer değer 0 değilse (örnekte olduğu gibi) bit 1'dir.

Mantıksal komutlar, bayraklar kaydedicisindeki bayrak bitlerinin tümünü etkiler; dolayısıyla da, mantıksal VE işlemini yaptıktan sonra, bu işlemi, sözgelimi, sıfır bayrağının durumuna göre koşullu olacak bir komutun izlemesi olağandır. Koşullu dallanma veya atlama komutları olarak bilinen bu komutlar bir sonraki bölümde ele alınacaktır.

Mantıksal VEYA (OR)

Belirli işlemleri yaparken, kimi zaman, sözgelimi, bir değerin en büyük değerlikli dört biti ile ikinci bir değerin en küçük değerlikli dört bitinden oluşan yeni bir değer oluşturmak gerekebilir. Bu işlemi yerine getirmek için, (her ikisi için de mantıksal VE (AND) komutu kullanarak) önce birinci değerin en küçük değerlikli dört biti ile ikinci değerin en büyük değerlikli dört biti maskelenir , sonra da bu iki sonuç mantıksal VEYA (OR) komutu kullanarak *birleştirilir*. Bu birleştirme işlemi mantıksal VEYA komutu açıklandıktan sonra gösterilecektir.

Mantıksal VEYA komutunun doğruluk tablosu Tablo 4.8'de verilmiştir. Görüldüğü

Tablo 4.8 *Mantıksal VEYA (OR) işlemi*

Bit 1	*Bit 2*	*Bit 1 VEYA Bit 2*
0	0	0
0	1	1
1	0	1
1	1	1

gibi, çıkış; ya Bit 1 *veya* Bit 2 'den biri 1, *ya da* her ikisi de 1 iken mantıksal 1'dir; bunun dışındaki tüm durumlarda çıkış mantıksal 0'dır.

Mantıksal VE komutunda olduğu gibi burada da mantıksal VEYA komutu yürütülürken, mikroişlemci her bir ardışık bit çiftiyle sekiz kez VEYA işlemi yapar. Birinci değer A kaydedicisinde tutulur, ikinci değer ise ya bir mikroişlemci kaydedicisinin, ya bellek konumunun o anki içeriğidir, ya da komutta belirtilen dolaysız veri değeridir.

Örneğin, eğer A ve B kaydedicilerinin içerdiği değerler sırasıyla 63 (onaltılı) ve 0F (onaltılı) ise aşağıdaki

OR B

komutu, şu sonucu verir :

(A) = 0110 0011 = 63 (onaltılı)
(B) = 0000 1111 = 0F (onaltılı)

(A) VEYA (B) = 0110 1111 = A'nın içerdiği yeni değer

Bu örnekte şu noktaya dikkat edilmelidir: ikinci değerdeki (B) bitin değeri 1 ise, çıktı da daima 1'dir; fakat ikinci değerdeki bit 0 ise, sonuç ilk değerdeki (A) ilgili bitin durumuna göre belirlenir.

Benzer şekilde, A'nın içeriğinin aynı olduğunu varsayarak;

OR F0

komutu,

(A) = 0110 0011 = 63 (onaltılı)
Dolaysız veri = 1111 0000 = F0 (onaltılı)

(A) VEYA F0 = 1111 0011 = A'nın içerdiği yeni değer

sonucunu verir.

Daha önce de bahsedilen birleştirme işlemini göstermek için, mantıksal VE ve VEYA komutlarını kullanan aşağıdaki şu örneği inceleyelim.

Program örneği 4.6 : Mantıksal VE ve VEYA komutları

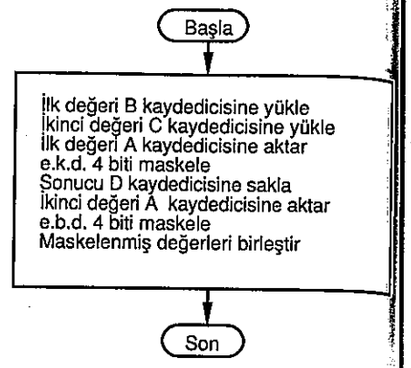
Şekil 4.7'deki program örneği, mantıksal VE (AND) ve VEYA (OR) komutlarının tipik kullanımını göstermektedir. Program,

Tablo 4.9 *Mantıksal ÖZEL VEYA (XOR) işlemi*

Bit1	*Bit2*	*Bit 1 ÖZ VEYA Bit2*
0	0	0
0	1	1
1	0	1
1	1	0

B kaydedicisinin içerdiği değerin en büyük değerlikli dört biti ile C kaydedicisinin içerdiği değerin en küçük değerlikli dört bitinden oluşan 8-bitlik bir değer üretir. Önce her bir değerdeki istenmeyen bitler,

a) Akış diyagramı



b) Program kodu

Sembolik komutlar		İşlem
LD	B,6A	(B) ← 6A (onaltılı)
LD	C,37	(C) ← 37 (onaltılı)
LD	A,B	İlk değeri A'ya yükle
AND	FO	E.k.d. 4 biti maskele
LD	D,A	Sonucu D'ye sakla
LD	A,C	İkinci değeri A'ya yükle
AND	OF	e.b.d. 4 biti maskele
OR	D	Maskelenmiş değerleri birleştir, yani [A] ← 67 (onaltılı)

Şekil 4.7 Program örneği 4.6

mantıksal VE komutu kullanılarak maskelenir ve sonra da istenen sonuç, mantıksal VEYA komutu kullanılarak maskelenen bu iki değerin birleştirilmesiyle üretilir. Sonuç A kaydedicisinde bırakılır.

Özel VEYA

Bir G/Ç arabirim portundan okunan 8-bitlik bir değerin sekiz sayısal birimin durumunu gösterdiğini varsayarsak, sık sık yapılan bir uygulama da, bu cihazların durumunu sürekli izlemek ve sadece bunlardan birinin durumu 1'den 0'a (açıktan kapalıya) veya 0'dan 1'e (kapalıdan açığa) değiştiğinde, bir başka işlemi başlatmaktır. Bu işi yerine getirmek için kullanılan

komut Özel VEYA ya da XOR komutudur.

XOR komutunun doğruluk tablosu Tablo 4.9'da gösterilmiştir. Görüleceği gibi, burada çıkış mantıksal VEYA işleminin benzeridir; ancak XOR'da çıkış, iki giriş biti birbirinden farklıysa (0,1 veya 1,0) 1; aynıysa (0,0 veya 1,1) 0'dır.

Burada da, XOR işlemi, A kaydedicisinin içerdiği değer ile ya başka bir mikroişlemci kaydedicisinin veya bir bellek konumunun içerdiği değeri ya da doğrudan bir veri değerini işin içine katar. Dolayısıyla, eğer A kaydedicisinin yürürlükteki değeri B3 (onaltılı) ise,

XOR AC

komutu:

(A) = 1011 0011 = B3 (onaltılı)
Dolaysız değer = 1010 1100 = AC (onaltılı)

(A) XOR B3 = 0001 1111 = A'nın içerdiği yeni değer

şeklinde olacaktır.

XOR komutunun bir uygulamasını göstermek için, sekiz tane denetimli cihazın yürürlükteki durumlarının (kapalı veya açık) B kaydedicisinde 8-bitlik bir değer olarak saklı tutulduğunu ve cihazların yeni durumunun A kaydedicisine okunduğunu varsayalım. Bu iki değer arasındaki tek bir XOR komutu, bu değerlerin farklı olup olmadığını ve dolayısıyla herhangi bir cihazın durumunun değişip değişmediğini gösterecektir. Bu, aşağıdaki komut kullanılarak gerçekleştirilir:

XOR B

ve bunun gördüğü işlev, aşağıdaki örnekte verildiği gibidir :

(A) = 1010 0110 = birimlerin yeni durumu
(B) = 1110 0110 = birimlerin eski durumu
(A) XOR (B) = 0100 0000 = hiçbir birimin durumu değişmediyse sıfır, bir veya daha çok birim değiştiyse sıfırdan-farklı

Öteki mantıksal komutlar gibi XOR komutu da çeşitli bayrak bitlerini etkiler, dolayısıyla normalde bu komutu, görevi sıfır bayrağının durumuna göre koşullu olan bir komut izler: Eğer bayrak (1'e) kurulmuşsa değişiklik yapılmaz ve işlem

devam eder; eğer kurulmamışsa bir değişiklik olmuş demektir ve dolayısıyla bu değişikliğe karşılık bazı işlemler yapılır. Bunlar bir sonraki kısımda açıklanacaktır.

S 4.7

4.8 Kaydırma komutları

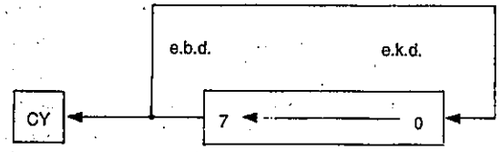
Az önce tanıtılan komutlara ek olarak bir mikroişlemcide ikili değerler üzerinde bit işleme veya kaydırma işlemleri yapan komutlar da vardır. Kaydırma komutlarının birçok kullanımı vardır. Örneğin; eğer iki değer arasında, bir önceki kısımda anlatıldığı şekilde, yapılan bir XOR komutundan sonra bir durum değişikliği algılanmışsa, belirli işlemleri yapabilmek için hangi bitin (dolayısıyla da birimin) değişmiş olduğunu belirlemek gerekir. Bunu yapmanın en uygun yolu, kaydırma komutları kullanmaktır.

Kaydırma işlemi A kaydedicisinin içeriğini sola veya sağa doğru bir basamak kaydırır. Ayrıca, bir mikroişlemcideki kaydırma komutları normalde, bu işlemin ardından onun durumuna ilişkin testler yapılabilmesi için elde bayrağını da (CY) işin içine katarlar.

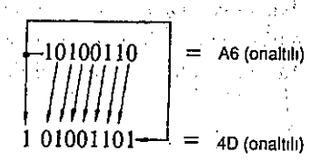
Örneğin, şu komutu ele alalım :

RLCA

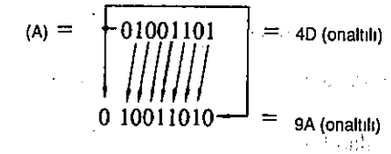
Bu komut, A kaydedicisinin içeriğinin sola doğru bir basamak kaydırılmasını sağlar. A kaydedicisinin en büyük değerlikli bit ise, hem kaydedicinin en küçük değerlikli bitinin değeri, hem de elde bayrağının içereceği yeni değer olur.



Örnek olarak, A kaydedicisinin yürürlükteki değerinin A6 (onaltılı) olduğunu varsayalım. RLCA komutunun sonucu şöyle olacaktır:



Yani A'nın yeni değeri 4D (onaltılı) olacak ve elde bayrağı da 1'e kurulacaktır. Bir başka RLCA komutu da şu etkiyi yaratacaktır:



Yani, A'nın yeni değeri 9A (onaltılı) olacak ve elde bayrağı da sıfırlanacaktır (0 olacaktır). Açıkça görüleceği gibi, eğer A'da tutulan o anki değer, daha önce sözü edilen bir XOR komutunun sonucuysa, elde bayrağı kurulana (yani 1 olana) kadar arka arkaya yürütülecek RLCA komutları, kurulmuş yani değişmiş olan bitin belirlenebileceği bir düzenek olacaktır.

Ayrıca kaydırma komutları, ikili çarpma ve bölme işlemlerini yaparken de yaygın biçimde kullanılır. Daha önce de sözü edildiği gibi, pek çok 8-bitlik mikroişlemcide çarpma ve bölme komutu bulunmaz ve bu nedenle de bu işlemler gerektiğinde programcı, bu işlevleri yerine getirmek için az önce açıklananlara benzer bir komutlar listesi kullanmalıdır. Örneğin ikili bir çarpma işlemi, arka arkaya yapılacak birkaç sola kaydırma komutuyla yerine getirilebilir, çünkü bir sola kaydırma işlemi iki ile çarpma işlemine eşdeğerdir. Benzer biçimde, ikili bölme işlemi de arka arkaya yapılacak sağa kaydırma işlemleriyle yerine getirilebilir, çünkü bu işlemde de her bir sağa kaydırma iki ile bölme işlemine eşittir.

S 4.8

4.9 Karşılaştırma komutu

Çok yaygın ve kullanışlı bir programlama işlemi de (bir sonraki bölümde görüleceği gibi) bir komutu veya komut listesini belli bir sayıda arka arkaya yürütmektir. Bu iş; listede baştan sona her geçişin ardından, istenen sayıda geçişin yapılıp yapılmadığını belirlemek için bir test yaparak gerçekleştirilir. Eğer istenen sayıya ulaşılmışsa, yürütüm normal sırasında devam eder; ulaşılmamışsa, tekrar listenin başına bir dallanılır.

Şimdiye kadar açıklanan komutların tümü de A kaydedicisinin içeriğini değiştirebilen komutlardı. Bu nedenle diyelim ki seçilen bir işlem 10 kez tekrarlanacaksa, bu, tipik olarak mikroişlemci kaydedicilerinden birini sayıcı olarak kullanmak suretiyle yapılabilir. Sayıcının değeri her işlem yapıldığında bir arttırılır ve değerin 10'a ulaşıp ulaşmadığını belirlemek için bir test yapılır. Ulaşmadıysa işlem tekrarlanır, ulaştıysa işlem sona erdirilir.

Dolayısıyla test yaparken kaydedicinin (sayıcının) değerinin bozulmaması gerekir. Açıkçası basit bir çıkarma işlemi yürürlükteki değeri (sayıyı) bozar. Bu nedenle daha uygun yöntem, karşılaştırma komutu kullanmaktır.

Karşılaştırma komutu iki değeri, her iki değer üzerinde de değişiklik olmayacak biçimde karşılaştırır. Bununla birlikte çeşitli bayrakları etkiler. Örneğin, sıfır bayrağı eğer iki değer birbirine eşitse 1'e kurulur. Karşılaştırılan iki değer, A kaydedicisinin değeri ile ya başka bir mikroişlemci kaydedicisinin ya bellek konumunun içeriği ya da dolaysız bir veri değeridir. Böylece aşağıdaki

```
CP  0A
```

komutu, A kaydedicisinin yürürlükteki değeri ile 0A (onaltılı), yani ondalık 10 dolaysız değerini karşılaştıracaktır. Eğer bunlar birbirine eşitse sıfır bayrağı 1'e kurulacak, değilse bayrak sıfırlanacaktır. Böylece A kaydedicisi az önce sözü edilen sayıyı tutsaydı, sıfır bayrağının durumuyla ilgili basit bir test, döngü işleminin sona erdiğini belirlemek için kullanılabilirdi. Bu teknik bir sonraki bölümde geniş olarak kullanılacaktır.

Sorular

4.1 Aşağıdaki ondalık sayıları, 8-bitlik ikiye tümleyen işaretli ikili sayı biçiminde ifade edin :

+63 +112 +37
-111 -27 -78

4.2 İkiye tümleyen işaretli sayı gösteriminin kullanıldığını varsayarak, aşağıdaki ikili sayıların ondalık eşdeğerlerini türetin:

01011011	00010111	01111111
11110111	10000101	11010101

4.3 Aşağıdaki ondalık sayıları 8-bitlik ikiye tümleyen işaretli ikili biçimlerine dönüştürün ve belirtilen toplama ve çıkarma işlemini yapın. Cevaplarınızı, doğru olup olmadıklarının sağlamasını yapmak için ondalık biçimlerine dönüştürün.

(a) 65+24	-28+81	-63+(-56)
(b) 94-36	-53-23	-68-(-84)

4.4 BCD sayı ifadesini göz önüne alarak, Kısım 4.6'da türetilen düzeltmeleri kullanıp aşağıdaki toplama ve çıkarma işlemlerinin BCD sonuçlarını bulun. Cevaplarınızı, doğru olup olmadıklarının sağlamasını yapmak için ondalık biçimlerine dönüştürün:

01100111 + 00100100
10001001 + 00001000
01110100 - 01001000
10011000 - 00111001

4.5 BCD sayı ifadesini göz önüne alarak, aşağıdaki komutlar yürütüldükten sonra A kaydedicisinde kalan ikili sayının ondalık eşdeğerini türetin:

```
LD    A,27
ADD   A,56
DAA
SUB   38
DAA
```

4.6 Önce aşağıdaki toplama işleminde görülen üç sayıyı sırasıyla B,C ve D kaydedicilerine yükleyen, ardından da BCD sayı gösterim biçimini esas alarak aşağıda gösterilen işlemleri yapan bir program yazın:

26+69-56

4.7 Aşağıdaki onaltılı sayı çiftleriyle yapılacak mantıksal VE, VEYA ve ÖZEL VEYA işlemlerinin sonuçlarını bulun:

A7, B8
E2, DC
3C, 73
5A, 6B

4.8 Aşağıdaki program çalıştırıldıktan sonra A kaydedicisinde kalan onaltılı sayı ile elde bayrağının durumunu bulun:

```
LD    A,6F
LD    B,A4
AND   B
OR    44
RLCA
RLCA
```

5. Bölüm : **Denetim aktarma komutları**

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde:*

1 *Tipik bir koşulsuz ve koşullu komutun yapısını anlayabilmeli,*
2 *Bu komutları uygularken bir mikroişlemcinin gerçekleştirdiği işlemleri açıklayabilmeli,*
3 *Atlama komutları kullanan programlar yazıp, bunları anlamalı ve sembolik adres etiketleri kullanmanın avantajlarını değerlendirebilmeli,*
4 *Program tasarımının akış diyagramı yönteminde yer alan sembolleri kullanabilmeli ve bunlar aracılığıyla basit programlar tasarlayabilmeli,*
5 Döngüleme *ve* ikili karar *terimlerini anlamış olmalı ve bunları kullanan basit programlar tasarlayıp gerçekleştirebilmeli,*
6 *Program tasarımında alt yordam kullanmanın önemini anlayabilmeli,*
7 *Bir altyordamı çağırma (CALL) veya altyordamdan dönüş (RET) komutlarını kullanırken mikroişlemci tarafından yapılacak işlemleri tanımlayabilmeli,*
8 *Bir yığının yapısını ve yığını kullanan çeşitli mikroişlemci komutlarının, yığının içeriğini nasıl etkilediğini tanımlayabilmeli,*
9 Parametre *terimini tanımlayabilmeli ve bir bellek bloku aracılığıyla bir altyordam parametre geçirmeyi içeren kısa programlar yazabilmeli,*
10 *İç içe altyordama terimini tanımlayabilmeli ve altyordam tasarımıyla olan ilişkisini anlayabilmelisiniz.*

5.1 Giriş

Şimdiye dek ele alınan tüm komutlar belli bir veri taşıma veya veri işleme işlemi yapmaktadırlar. Bu yüzden eğer elimizde kullanılabilir tek komut tipi bunlar olsaydı, mikroişlemci bir programı yürütürken, birinci komuttan başlayacak ve listenin sonuna ulaşana kadar tüm komutları sırayla yürütecekti; yani programın yürütümü sırasında, programdaki her bir komut mikroişlemci tarafından yalnızca bir kez işleme koyulacaktı.

Ancak, saklı-program kavramının temel avantajı şudur ki, program bir kez belleğe saklandıktan sonra, aynı komut listesi birçok defa yürütülebilmektedir. Doğaldır ki bu, işleticinin komut konsolundan programı yeniden yürütmesiyle yapılabilir. Daha da önemlisi, mikroişlemcinin, programın tek bir kez çalıştırılması süresinde aynı komut listesini veya grubunu birçok kez yeniden yürütebilmesidir.

Dolayısıyla böyle bir olanak sağlamak için bir mikroişlemcide, özel veri taşıma komutları veya aritmetik işlemler yapan komutların yanında, programın yü-

rütülmesinin mevcut sırasını denetleyen veya değiştiren komutlar da bulunmaktadır. Bunlar *denetim aktarma* komutları olarak bilinirler ve görüleceği gibi bir program yazarken programcıya oldukça büyük bir esneklik sağlarlar. Bunlar yalnızca programın belli bir bölümüne (komutuna) geriye dallanmayı sağlamakla kalmayıp aynı zamanda, programın yürütümü sırasında belli koşullara bağlı olarak dallanmayı gerçekleştiren komutlardır. Her iki tür de bu bölümde ele alınacaktır.

5.2 Koşulsuz atlama komutları

Tüm mikroişlemcilerde bir koşulsuz atlama komutu vardır. Bu komut programcıya, bir programın yürütülmesindeki normal ardışık sırayı yarıda kesip, ardışık işlemleri tekrar uygulamaya geçmeden önce, bir sonraki komutu almak için koşulsuz olarak programın farklı bir bölümüne dallanabilme olanağı sağlar.

Bu nedenle de koşulsuz bir atlama komutu; komut türünü belirleyen bir işlem ve bir adres parçasından oluşur. Burada adres, sözgelimi, işlenecek veriyi içeren bir bellek adresini belirtmeyip, yürütülecek bir sonraki komutun bellekteki adresini gösterir.

Tablo 5.1 *Koşulsuz atlama komutunun çalışması*

Bellek adresleri	*Sembolik komutlar*	*İşlem*
2000	-	
.	.	
.	.	
2010	- ←	Varış yeri komutu
.	.	
.	.	
2020	JP, 2010 ┘	2010 (onaltılı) adresine dallan
.	.	
.	.	
2030	-	

Örneğin, Z80'de koşulsuz atlama komutu şöyle yazılır:

```
JP  2010
```

Yürütülecek bir sonraki komutun bellek adresi

Koşulsuz atlama

Dolayısıyla, bir programı oluşturan komutlar listesinin bellekte 2000 (onaltılı) adresinden başlayarak saklandığı ve yukarıdaki atlama komutunun 2020'de (onaltılı) saklandığı varsayılarak, bu komutu yürütmenin sonucu Tablo 5.1'de gösterilmiştir.

Koşulsuz atlama komutu normalde üç baytlık bir komuttur ve Z80 için yukarıdaki komut şuna eşdeğerdir:

Bellek adresi	*İçerik (onaltılı)*	
2020	C3	= JP veya koşulsuz atlama
2021	10	= varışyeri adresinin en küçük değerlikli baytı
2022	20	= varışyeri adresinin en büyük değerlikli baytı

Böylece, bu komutu yürütürken, komut saykılının getirme fazı boyunca mikroişlemci, program sayıcısında o anda tutulmakta olan bellek adresinden [2020 (onaltılı)] komutun ilk baytını [C3 (onaltılı)] okur ve bunu komut kaydedicisine yükler. Sonra da bundan, komutun bir koşulsuz atlama olduğunu ve bu nedenle de bellekten iki baytın daha okunması gerektiğini belirler. Dolayısıyla, bellekten ardışık iki bayt daha getirilir ve her bir bayt okunduktan sonra program sayıcısının değeri bir artırılır.

Böylece komutun üç baytı da bellekten okunduktan sonra, program sayıcısı içinde, otomatik olarak sıradaki bir sonraki komutun ilk baytının bellek adresi [2023 (onaltılı)] bulunacaktır. Bununla birlikte, komut saykılının yürütme fazı sırasında mikroişlemci, getirme fazı boyunca bellekten okunan iki baytı [20 ve 10 (onaltılı) değerlerini] program sayıcısına yükler ve dolayısıyla o anki değerinin üzerine yazmış olur.

Mikroişlemci işleme, bir sonraki getirme fazıyla devam eder ve dolayısıyla program sayıcısında tutulan yürürlükteki adresten sonraki komutun ilk baytını okur. Öte yandan sayıcı, sıradaki bir sonraki komutun ilk baytını göstermek yerine artık 2010 (onaltılı) adresini, yani sıranın dışındaki bir komutun ilk baytını göstermektedir. Daha sonra programın yürütümü, başka bir denetim aktarma komutu işleme konulana kadar sıralı olarak devam eder.

Komut etiketleri

Koşulsuz atlama komutu; program içinde ya, yukarıda da belirtildiği gibi, daha önceki bir komuta ya da ilerideki, sonraki bir komuta dallanmak veya atlamak için kullanılır. Bu ikincisinin yararları bu bölümde ileride gösterilecektir. Öte yandan bundan şu sonuç çıkarılabilir: Programcı, atlama komutu içeren bir program yazarken, bellekte hedef komutunun saklandığı mutlak adresi her zaman bilemez.

Bu nedenle, atlama komutlarının yer aldığı bir program yazarken, programcının varışyeri komutlarının amaçlanan adreslerini belirtmek amacıyla etiketler (sembolik adres adları) kullanması olağandır. Bunlar yalnızca, çeviri işlemi sırasında mutlak adreslere dönüştürülürler. Bu durum, iki koşulsuz atlama komutu içeren Tablo 5.2'de gösterilmiştir: Biri sıra dışındaki LAB1 sembolik adresli (veya etiketli) komuta geriye doğru dallanır, öteki ise sıra dışındaki LAB2 simgesel adres etiketli komuta ileriye doğru dallanır.

Normalde etiketler için kullanılan sembolik adlar alfabetik ve sayısal karakterlerin

Tablo 5.2 : *Sembolik adres adları olarak etiketlerin kullanımı*

Bellek Adresleri	*Sembolik komutlar*	*İşlem*
Çevirme işlemi sırasında atanır	LAB1: Varışyeri komutu JP LAB1	Sembolik adres adı LAB1 olan komuta dallan
	JP LAB2	Sembolik adres adı LAB 2 olan komuta dallan
	LAB 2 : Varışyeri komutu	

bir karışımıdır ve genellikle programcı tarafından verilen anlamlı isimlerdir. Örneğin;


```
TOPLAM:
ALARM:
TEKRARLA:
SON:
```

Bunlar anlamları kolaylıkla anlaşılabilecek etiket örnekleridir. Bu, ileride görüleceği gibi, aslında denetim aktarma komutları içeren bir programın okunabilirliğini, do-
S 5.1 layısıyla da anlaşılabilirliğini artırır.

5.3 Koşullu atlama komutları

Koşulsuz atlama komutu oldukça yararlı bir işlev görse de, saklı program kavramının gerçek esnekliği, koşullu türdeki atlama komutlarının varlığından kaynaklanmaktadır. Böylece ardışık program yürütüm sırasının koşulsuz olarak kesilmesinin yerine, kesilme ancak belirtilmiş bir koşulla karşılaşıldığında gerçekleşir; aksi halde normal işlem sırasına devam edilir.

Koşullu atlama komutlarının kullanışlılığını göstermek için, sözgelimi, ardışık yüz bellek konumunda saklanan yüz adet değerin toplamını bulacak bir program parçası yazılacak olsun. Bunlar tipik olarak program içindeki daha önceki bir komut grubu tarafından bir G/Ç arabirim portundan okunmuş değerler olabilir.

Açıkçası bu durumda, bir atlama komutu kullanılmasaydı, her bir değeri sırayla okuyup, örneğin, akümülatörde tutulan yürürlükteki toplama ekleyecek, yüz kere tekrarlanan bir işlem listesi gerekecekti. Böylece her bir toplama işlemi için, -biri bellekte yer alan bir sonraki değerin adresini hazırlamak, diğeri de toplama işlemini yapmak için- iki komut kullanılacağı varsayılırsa, program parçası aşağıda görüldüğü gibi iki yüz komuta gerek duyacaktı:

```
.
.
.
LD    A,00        Toplamı sıfırla
LD    HL,2800     HL'yi ilk değeri gösterecek biçimde düzenle
ADD   A,(HL)      İlk değeri devreden toplama ekle
INC   HL          HL'yi ikinci değeri göstermesi için
                  artır
ADD   A,(HL)      İkinci değeri devreden toplama ekle
INC   HL          HL'yi üçüncü değeri göstermesi için
                  artır
ADD   A,(HL)      Üçüncü değeri devreden toplama ekle
```

(Devamı var)

```
INC   HL      HL'yi yüzüncü değeri göstermesi için
                artır
ADD   A,(HL)  Yüzüncü değeri devreden toplama ekle
```

Yukarıdan da kolayca anlaşılacağı gibi, akümülatör ve adres işaretçi (H,L) ilk kullanıma hazırlandıktan sonra, program tekrarlamalı bir işlemler listesinden oluşmaktadır. Böylece ikinci toplama komutu yerine mikroişlemciyi geriye birinci toplama komutuna dallandıracak bir atlama komutu kullanılırsa, her bir yükleme ve toplama işlemi tekrarlamalı olarak gerçekleştirilecek ve dolayısıyla tüm program parçasının oluşturulması için yalnızca beş komut yeterli olacaktır.

Bu çözümün yol açtığı şöyle bir sorun vardır: eğer koşulsuz bir atlama komutu kullanılırsa, mikroişlemci tüm değerlerin toplandığını bilmeyecek ve geriye dallanmaya (ya da *döngülemeye*) devam edecektir. Fakat koşullu bir atlama komutu kullanarak bu sorunun üstesinden kolayca gelinir. Daha sonra görüleceği gibi, dallanma işleminin tekrarlama sayısının kaydını tutarak (döngü sayıcı) bunun önceden saptanan bir sınır değere ne zaman ulaştığını belirlemek mümkündür. Böylece dallanma işlemine son verilir ve normal ardışık program yürütülmesine devam edilir.

Atlama koşulları

Koşullu atlama komutlarında belirtilen koşulların tümü mikroişlemcideki bayrak kaydedicisinde bulunan çeşitli bayrak bitlerinin durumlarıyla belirlenir. Örneğin, normalde sıfır veya Z bayrağına dayalı iki tane koşullu atlama komutu vardır. Biri ile atlama (veya dallanma), eğer sıfır bayrağı kurulmuşsa [1 ise] gerçekleşirken; diğeri ile, yalnızca sıfır bayrağı kurulmamışsa [1 değilse] gerçekleşir. Dolayısıyla; hangisinin kullanıldığına bağlı olarak, eğer uygun koşul yerine getirilmemişse atlama gerçekleşmez ve normal ardışık yürütüm devam eder. Örnek olarak, Tablo 5.3'te Z80 kullanılarak belirtilebilecek bazı atlama komutları görülmektedir. Listelenen bu komutlar sırasıyla

Tablo 5.3 *Koşullu atlama komut kısaltmaları*

Komut kısaltması	*Bayrak durumu*	*İşlem*
JP Z	Z=1	Sıfır bayrağı kurulmuşsa dallan
JP NZ	Z=0	Sıfır bayrağı kurulmamışsa dallan
JP C	C=1	Elde (bayrağı) kurulmuşsa dallan
JP NC	C=0	Elde (bayrağı) kurulmamışsa dallan
JP P	S=0	İşaret (bayrağı) pozitifse (0) dallan
JP N	S=1	İşaret (bayrağı) negatifse (1) dallan

sıfır, elde ve işaret bayraklarına bağlıdır. Böylece örnek bir komut :

JP NZ,TOPLAM

komutudur.

Bu komutun yaratacağı sonuç; yalnızca sıfır bayrağı kurulmuş değilse (Z=0 ise), sıra dışına çıkarak sembolik adres etiketi TOPLAM olan komuta atlamak veya dallanmaktır. Aksi halde (Z=1 ise), sıradan devam edilir.

Bu komutun kullanımını göstermek için, yukarıda verilen örneği, yani 2800 (onaltılı) başlangıç adresinden itibaren bellekte saklı olan yüz tane sayıyı toplayacak program parçasını ele alalım. Daha önce de belirtildiği gibi, dallanma veya döngüleme işlemini denetlemek ve böylece yüz tane değer toplandıktan sonra döngü işlemine son vermek için, koşullu bir atlama komutunu kullanmak gerekir.

Program Örneği 5.1 : Koşullu Atlama Komutları

```
        .
        .
        LD      B,00          Toplamı sıfırla
        LD      HL,2800       HL'yi ilk değeri gösterecek biçimde düzenle

DONGU:  LD      A,B           İlk (bir sonraki) değeri devreden
        ADD     A,(HL)          toplama ekle
        LD      B,A           Devreden toplamı (ara toplamı)
                                B'de sakla
        INC     HL            HL'yi sonraki değeri göstermek üzere
                                artır
        LD      A,L           100 değer de toplandı mı ?
        CP      64
        JP      NZ, DONGU     Hayır (Z=0): bir sonraki değeri eklemek
                                için döngüye devam et
                              Evet (Z=1) : sıradan devam et (toplam B'de)
```

Yukarıdaki program parçası, 2800 (onaltılı) adresinden başlayarak bellekteki 100 ardışık konumda saklı olan yüz tane sayıyı toplar. Şu kaydediciler kullanılmıştır:

B devreden toplamı ve son toplamı tutmak için (tek bir bayt genişliğinde olduğu varsayılmıştır) kullanılmıştır.

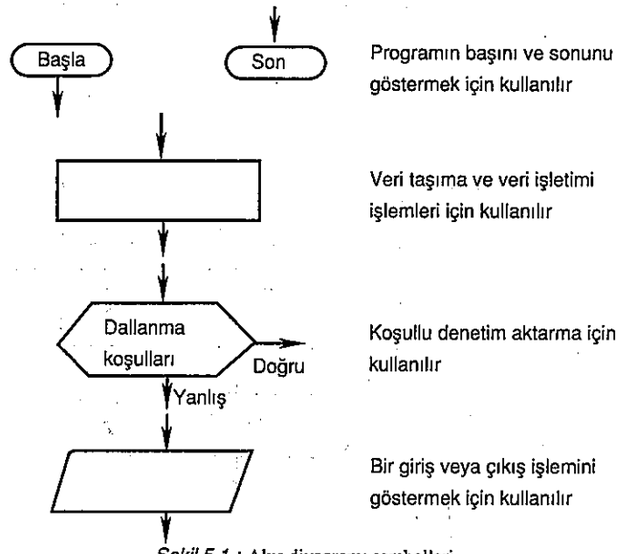
H,L listedeki her bir değer için bellek işaretçi olarak kullanılmıştır.

İlk değer bellekte 2800 (onaltılı) adresinde saklı olduğu için son değer de (yüzüncü değer) 2863 (onaltılı) bellek adresinde saklı olacaktır. Buradan, döngü sayıcı L kaydedicisinde tutulur ve bu nedenle de döngü sayıcının 64'e (onaltılı) ulaşıp ulaşmadığını belirlemek için dolaysız karşılaştır komutu kullanılır; yani L kaydedicisinin değeri 64'e (onaltılı) ulaştığında; program 99 kere döngülenmiş ve
S 5.2-5.4 dolayısıyla yüz tane değer toplanmış olacaktır.

5.4 Program tasarımı

Bir uygulama işi verildiğinde, herhangi bir program kodu yazılmadan önce bir programcı için gerekli olan ilk şey, program için bir tasarım modeli üretmektir. Bu; bir uygulama programının üretiminde önemli ve gerekli bir adımdır; çünkü uygun bir tasarım olmadan, istenen görevi yerine getirmek için gereken işlerin mantıksal sırasını planlamak çoğu zaman imkansızdır. Bu, eğer sonuçta ortaya çıkacak program birtakım koşullu ve koşulsuz atlama komutları içeriyorsa özellikle gerekli olacaktır.

Şu an kullanımda olan bir grup program tasarım yöntemi vardır ve benimsenecek yöntem genelde, en sonunda kullanılacak programlama diline bağlıdır. Öte yandan, bu kitapta sembolik (assembler) dilleri ele aldığımız için belki de en uygun yöntem



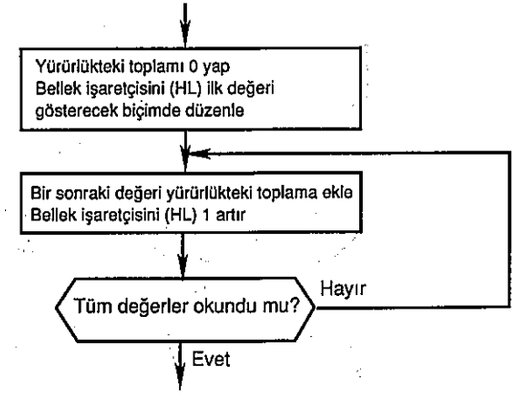
Şekil 5.1 : Akış diyagramı sembolleri

akış diyagramı yöntemidir ve bu nedenle kitabın kalan kısmında bu yöntem benimsenmiştir.

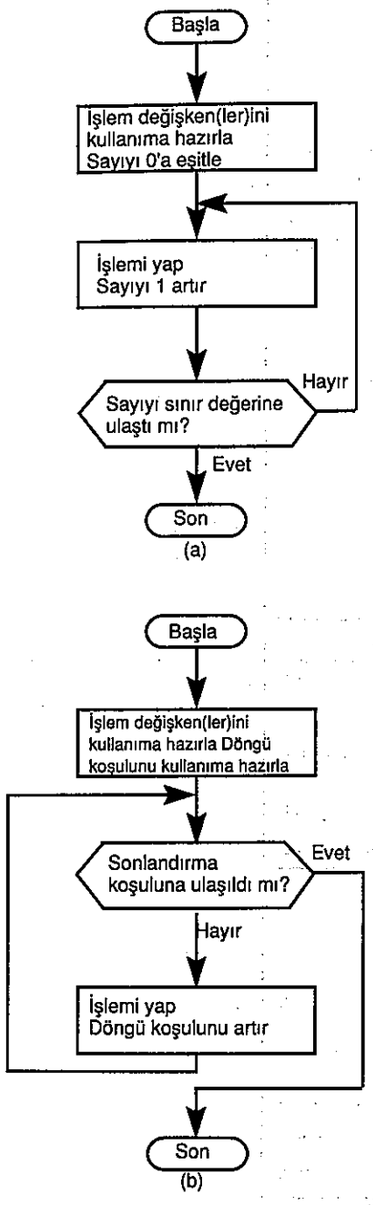
Programın yapısı, akış diyagramı yöntemi kullanılarak grafik veya şekil olarak ifade edilir. Akış diyagramları; her biri farklı bir işlem türünü gösteren ve uygulanacak belirli işlemleri tanımlayan, açıklayıcı bilgiyi içeren bir grup farklı şekildeki kutu ve sembollerden oluşur. Daha sonra bunlar; akış yönünü, dolayısıyla da yürütülecek işlemlerin mantıksal sırasını belirtmek amacıyla kullanılan tek yönlü çizgilerle birbirlerine bağlanırlar.

Aslında, mevcut olan semboller bir mikroişlemcinin komut setindeki değişik komut türleriyle yakından ilişkilidir. Böylece, örneğin, veri taşıma ve veri işleme işlemlerini gösteren bir sembol, koşullu denetim aktarma işlemlerini gösteren bir sembol ve G/Ç işlemlerini gösteren bir sembol vardır. Bunlar Şekil 5.1'de gösterilmiştir.

Veri taşıma/aritmetik ve koşullu denetim aktarmaya ilişkin iki sembolün kullanımını belirtmek için, daha önce Program Örneği 5.1'de sunulan program parçasına ait bir tasarım, Şekil 5.2'de gösterilmiştir. Bu şekilden çıkarılabileceği gibi, bir programın akış diyagramı çoğu zaman sonuçtaki programdan biraz daha yüksek düzeydedir. Böylece "Tüm değerler okundu mu ?" koşullu sembolünü yerine getirmek için aslında üç komuta ihtiyaç vardır. Gerçekte daha karmaşık problemlerde tasarım yönteminin kendisi de; sistem düzeyinde bir akış diyagramıyla başlayan ve asıl program kodunun gerçekleştirilmesine uygun sonuç akış diyagramına ancak birkaç geliştirme adımından sonra ulaşan, tekrarlamalı bir işlem sürecidir.



Şekil 5.2 : Program örneği 5.5 için akış diyagramı



Şekil 5.3 İki alternatif döngüleme akış diyagramı

Döngü İşlemi

Program örneği 5.1, "bilinen bir sayı ile döngüleme" olarak bilinen denetim aktarma yapısına bir örnektir. Bu çok yaygın bir yöntemdir ve kimi zaman tek bir programda bile birçok yerde kullanılır.

Öte yandan, Şekil 5.2'de görülen akış diyagramı, bu tür bir problemin çözümlerinden yalnızca birisidir ve bazen alternatif akış diyagramları da kullanılır. Şekil 5.3'te iki alternatif görülmektedir. İkinci çözümden çıkarılabileceği gibi, koşullu test işlemi, döngüde işlenen görev yerine getirilmeden önce yapılır. Örneğin, Bu bazen; döngüde işlenen görevin hiç yerine getirilmemesinin istendiği durumlarda, özellikle gerekli olabilir.

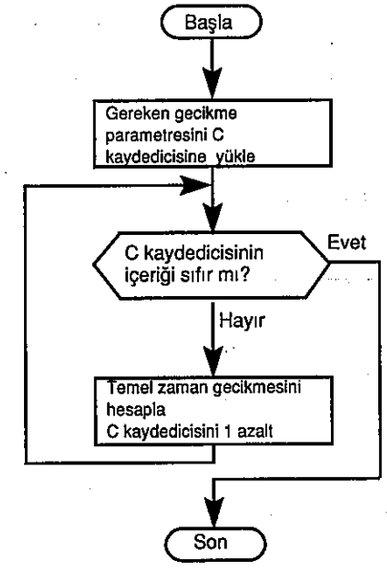
Çözüm *(b)*'yi kullanan bir uygulamayı göstermek için, aşağıdaki program örneğini ele alalım. Bu program, mikroişlemci kaydedicilerinden birine yüklenen bir değer ile orantılı olarak değişen bir zaman gecikmesini hesaplamak için kullanılabilir. Örneğin program, bir G/Ç arabirim portuna çıkışı yapılacak ardışık iki değer arasındaki belirli bir zamanın geçtiğinden emin olmak için zaman gecikmesini sık sık hesaplamak zorundadır.

Program Örneği 5.2 : Bilinen bir sayı ile döngü

Şekil 5.4'te gösterilen akış diyagramı ile ilgili program kodu bilinen bir sayı ile döngülemeye örnektir. Program; C kaydedicisine yüklenen değerle orantılı olan bir zaman gecikmesini hesaplar. Bu nedenle değer 00-FF (onaltılı) aralığında olabilir.

Mikroişlemcinin bir komutu yürütmesi az, ancak sonlu bir süre aldığından (tipik olarak birkaç mikrosaniye), hesaplanan gecikme döngüdeki komut sayısı ve yerine getirilen döngü sayısının her ikisiyle de orantılıdır. Dolayısıyla, C kaydedicisine yüklenen değer büyüdükçe yerine getirilen döngü de artacaktır ve bu nedenle hesaplanan zaman gecikmesi daha uzun olacaktır.

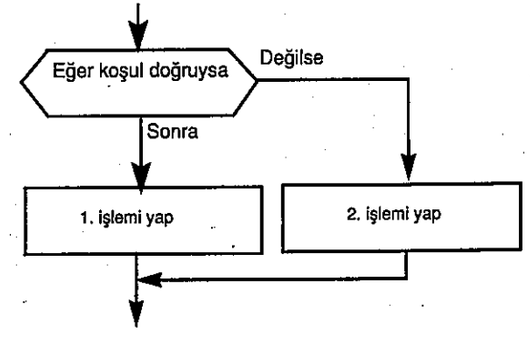
(a) Akış diyagramı



(b) Program kodu

Sembolik Komutlar			İşlem
	LD	C,XX	Gereken gecikme parametresini C kaydedicisine yükle
GECIK :	LD	A,C	C'nin değeri sıfır mı?
	CP	00	
	JP	Z,ZAMAN	
	NOP		Hayır. Temel zaman geciktirmesini gerçekleştir
	NOP		
	NOP		
	DEC	C	Gecikme parametresini 1 azalt ve döngüle
	JP	GECIK	
ZAMAN :			Evet. Zaman gecikmesi hesaplanmıştır

Şekil 5.4 Program örneği 5.2



Şekil 5.5 İkili karar veya IF THEN ELSE yapısı

Döngüde kullanılan basit gecikme komutları, üç adet 'işlem-yapma' (NOP) komutudur. Bunlar, isminden de anlaşılacağı üzere, mikroişlemci kaydedicilerinin içeriğini değiştirecek herhangi bir işlem yapmazlar. Fakat mikroişlemcinin bellekten komutu getirip kodunu çözmesi gerektiğinden, her bir NOP komutunu getirip yürütmek de birkaç mikrosaniye alır. Bu yüzden de temel zaman gecikmesi (3 NOP) 10 mikrosaniyenin katları şeklindedir. Bu çözümde, eğer C kaydedicisine yüklenen değer sıfır ise hiçbir gecikme hesaplanmayacağına; öte yandan eğer test, programın sonunda, gecikme komutlarından sonra yerine getirilirse, en az bir gecikmenin ya-
S 5.5 pılacağına dikkat edin.

İkili kararlar

Sık sık istenen bir başka programlama görevi de, belirtilen koşulun sonucuna bağlı olarak iki alternatif işlemden birini yerine getirmektir, yani eğer (IF) koşul sağlanıyorsa, o halde (THEN) işlemi yap, aksi takdirde (ELSE) başka bir işlem yap. Bu yapı, *ikili karar* ya da alternatif olarak IF-THEN-ELSE yapısı olarak bilinir.Şekil 5.5'te akış diyagramı biçiminde gösterilmiştir.

Bu yapının bir uygulamasını ve bir program kodu içinde nasıl gerçekleştirildiğini göstermek için, tipik olarak basit bir sıcaklık DENETİM programının tasarımında karşılaşılabilecek aşağıdaki program örneğini ele alalım.

Program Örneği 5.3 : İkili karar yapısı

Şekil 5.6'da gösterilen akış diyagramı ile buna bağlı program kodu, ikili karar veya IF-THEN-ELSE yapısını kullanan bir program parçası örneğidir. Program; basit bir sıcaklık kontrol sisteminde kullanılan ısıtma elemanının açılıp açılmayacağını belirlemek için kullanılabilir. Bu, istenen (ayarlama noktası) sıcaklıkla ilişkili olmak üzere, tipik olarak o anki mevcut sıcaklık durumu tarafından kararlaştırılır. Eğer (IF) o anki sıcaklık ($S_{iç}$) ayarlanan sıcaklık değerinden (S_{ref}) küçükse, (THEN) ısıtıcıyı aç veya açık bırak; aksi takdirde (ELSE) ısıtıcıyı kapat. Bu nedenle, ısıtıcı iki-durumlu cihaz olarak ele alınabilir ve bundan dolayı da tek bir bit ile kontrol edilebilir.

Şu mikroişlemci kaydedicileri kullanılır :

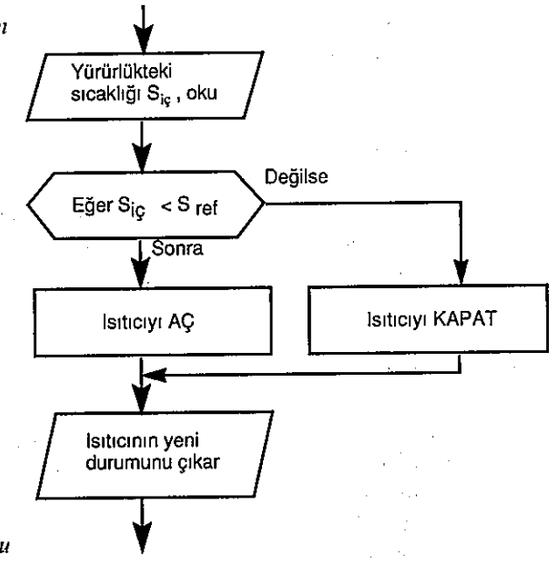
A içerideki o an mevcut sıcaklığı ($S_{iç}$) tutar (tipik olarak bir G/Ç giriş portundan okunur).

B ayarlanan (istenen) sıcaklık değerini (S_{ref}) tutar.

C C'nin en küçük değerlikli biti, ısıtıcının o anki durumunu tutar (0=açık 1= kapalı). (Tipik olarak öteki bitler öteki birimleri kontrol eder.)

Program; karşılaştırma komutu aracılığıyla o anki sıcaklığın ayarlanan sıcaklık değerinden küçük olup olmadığını belirler. Karşılaştırma komutundan sonra eğer elde bayrağı kurulmuşsa (yani 1 ise), karar doğrudur ($S_{iç} < S_{ref}$) ve dolayısıyla ısıtıcı ya açılır ya da açık bırakılır; aksi takdirde ısıtıcı kapatılır. Mantıksal VEYA ve VE komutları, ısıtıcı bitini sırasıyla kurmak ve sıfırlamak için kullanılır ve bu nedenle bu bit üzerinde işlem yapılması öteki herhangi bir bitin o anki durumunu etkilemez.

(a) Akış diyagramı



(b) Program Kodu

Sembolik komutlar	İşlem
.	S_{ref}'i B'de tut.
.	Isıtıcının yürürlükteki durumu C'de tut.
.	
Siç 'i A kaydedicisine gir	
CP B	
JP NC, HOFF	EĞER S iç < S ref
LD A,C	
OR 01	SONRA ısıtıcıyı AÇ
LD C,A	
JP HCON	
HOFF : LD A,C	
AND FE	DEĞİLSE ısıtıcıyı KAPAT
LD C,A	
HCON : Isıtıcının yeni durumunu (C) çıkar	
.	
.	
.	

Şekil 5.6 Program örneği 5.3

Altyordamların kullanımı

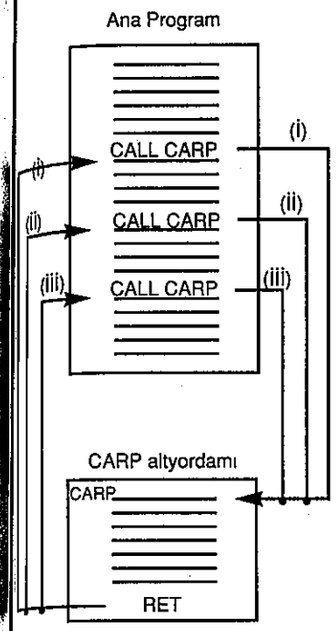
Bir program tasarlanırken, belli bir işleme (dolayısıyla bir program kod parçasına) aynı program içinde birkaç defa gereksinim duyulması, oldukça sık rastlanan bir durumdur. Örneğin, kullanılmakta olan mikroişlemcide bir çarpma işlemi olmadığını varsayarsak, aynı program içinde çarpma işlemini yerine getirecek bir kod parçasını (komutlar listesi) birkaç defa kullanmak gerekebilir. Açıkçası, kodun her yürütümünde tek farklılık, komutlardan çok, büyük olasılıkla kodun üzerinde işlem yapacağı değerlerde olacaktır.

Dolayısıyla, aynı işlem için her seferinde aynı kodu tekrar tekrar yazmak can sıkıcıdır ve büyük olasılıkla belleğin verimsiz şekilde kullanımına neden olur. Bu nedenle tüm mikroişlemcilerde, programcıya, sık sık kullanılan kod parçasını bir kez yazıp, program içinde gerek duyulan zamanlarda bu program parçasına dallanmayı sağlayan bir çift komut vardır.

Program parçası tarafından işlenecek değerler, işlemin uygulandığı her seferde, birbirinden farklı olabileceğinden, programcının, işlenecek değerleri düzenlemesi ve bu değerleri dallanma öncesinde belleğin bilinen bir alanına yerleştirmesi (saklaması) olağandır. Benzer biçimde, program parçası tarafından üretilen sonuçlar da bilinen bir alana yerleştirilirse, (ana) program yeniden çalıştırıldığında bu sonuçlara kolaylıkla erişebilir.

Sık kullanılan işlemleri yapmak için kullanılan program parçalarına *altyordam* denir. Bu program parçalarına dallanmaya neden olan komut da *CALL* komutudur. Öte yandan altyordam çalıştırıldıktan sonra, ana programdaki altyordamın çağrıldığı noktaya geriye dallanmak veya dönmek gerekir. Bu iş, *RET* (dön) komutu kullanılarak gerçekleştirilir. Şekil 5.7, aynı program içindeki farklı noktalardan tek bir altyordamın birkaç defa nasıl çağrıldığını göstermektedir.

Altyordamlar; programların yazılması sırasında, çok sayıda komutun birçok kez tekrarlanmasını önlemenin (dolayısıyla da bellekten tasarruf sağlamanın) yanı sıra, yapısal olarak iyi düzenlenmiş, bu nedenle de daha kolay okunup anlaşılır programların tasarlanabilmesinde önemli bir rol oynarlar. Belirli bir işlemi gerçekleştirmek için ayrı bir program parçası yazabilmek olanağı ile bununla iletişimde bulunmayı sağlayacak iyi tanımlanmış bir düzeneğin kullanımı; uzun ve kompleks bir programı daha küçük ve daha kolaylıkla anlaşılabilir altyordamlara (altprogramlara) bölmesi için programcıya ideal bir araç sağlar. Bundan başka, her biri daha küçük altyordamlar yazabileceğinden, büyük bir programın tasarlanması ve yazımı sırasında birden fazla programcı görevlendirmek mümkün olabilecektir.



Şekil 5.7 Altyordam çağırma (CALL) ve dönüş (RET) mekanizması

Birkaç altyordam içeren tipik bir uygulama programının yapısı Şekil 5.8'de gösterilmiştir. Bir sonraki kısımda görüleceği gibi, bir altyordamdan bir başkasını çağırmak oldukça normaldir ve bu nedenle de ana program parçası çoğu kez altyordamları çağıran çağrıları uygun bir sırada kullanıma hazır hale getiren küçük bir kod parçasından ibaret olmaktadır.

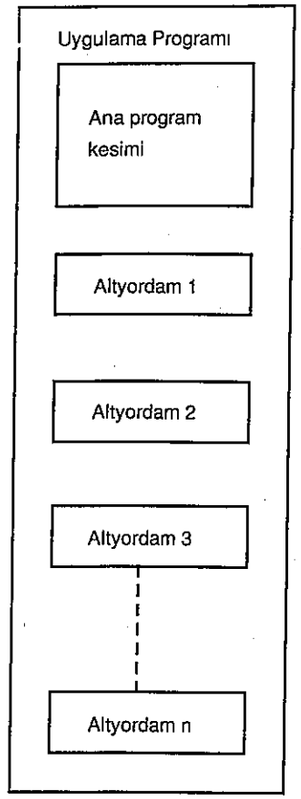
Genelde bu kitaptaki programlar oldukça küçük olsalar da, program tasarımında altyordamların kullanılması çok önemlidir ve bu nedenle ilgili tekniği göstermek için altyordamlar mümkün olan her yerde kullanılacaktır. Bir sonraki kısımda altyordamların uygulanması daha ayrıntılı olarak açıklanacaktır.

5.5 Altyordamların gerçekleştirilmesi

Şekil 5.7'den CALL komutunun koşulsuz bir atlama komutuna çok benzediği görülebilir : altyordamdaki ilk komuta sembolik bir adres adı veya etiketi verilir ve altyordamın her çağrılışında bu etiket kullanılır. Fark belki de en iyi şekilde, çağırma mekanizmasını değil, RET (dön) komutundan çıkarılacak anlamları ele alarak görülebilir.

CALL komutunun yürütüldüğü her seferde, dallanma hep aynı yere olsa da; RET komutu yürütüldüğünde gerçekleştirilecek olan dallanma, ana programda (ve dolayısıyla CALL'da) altyordam ilk olarak hangi noktadan çalıştırıldıysa, o nokta tarafından belirlenir. Şekilde geriye ana programa dallanmanın; altyordamın çalıştırılmasını başlatan CALL komutunun ardından gelen bir sonraki komut olduğuna dikkat edin.

Dolayısıyla, RET komutu yürütüldüğünde kullanılacak varışyeri adresi, ilgili CALL komutunun ardından gelen bir sonraki ardışık adres olacaktır. Bu da elbette, uygulamaya konulacak altyordama dallanmadan önce mikroişlemci program sayıcısında (PC) bulunan değer olacaktır. Bu yüzden CALL komutu yürütülürken, mikroişlemci program sayıcısının o anki değerini saklar ve bunun gösterdiği RET komutu yürütüldüğünde de, çağıran programın geriye doğru hangi noktasına dallanacağını bilir. Aslında, basitçe ifade edilirse, saklı



Şekil 5.8 Altyordam kullanan uygulama program yapısı

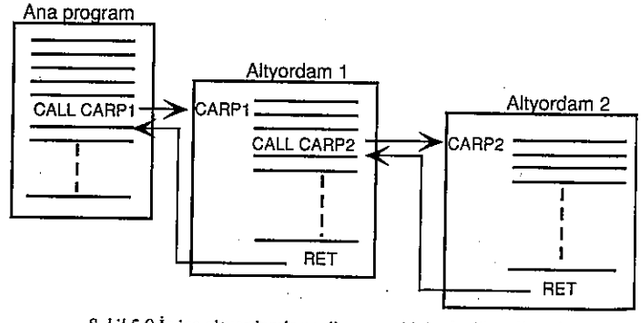
adresi tekrar program sayıcısınına alır ve böylece otomatik olarak çağıran programdaki noktaya dallanılır.

Yığın

Daha önce de sözü edildiği gibi, programcı için bir altyordam içinden bir başka altyordamı çağırmak (CALL) olağan bir işlemdir. Bu durum Şekil 5.9'da diyagram olarak gösterilmiştir ve *iç içe altyordamlar* olarak adlandırılır.

Bu düzenekten çıkarılacak anlam şudur: mikroişlemci RET (dön) komutunu yürütmeden önce bir dönüş adresleri grubu saklamak zorundadır. Örneğin, Şekil 5.9'da gösterilen programı yürütürken, mikroişlemci bir dönüş adresini, CALL ALT1 ve bir tane de CALL ALT2 yürütüldüğünde saklamalıdır. Bundan başka, altyordam 2'nin sonundaki RET (dön) komutu, altyordam 1'den önce yürütüleceğinden bu saklı adreslere ters sırada ihtiyaç duyulur.

Altyordam çağrılarına ilişkin dönüş adreslerini saklamak veya bu adresleri hatırlamak için mikroişlemci tarafından kullanılan düzenek *yığın* olarak adlandırılır. Yığın, bir son-giren-ilk-çıkar kuyruğudur. Çünkü tüm işlemler yığının başında yer almaktadır. Bu nedenle, bir değer kuyruğa girdiğinde daha önce kuyrukta bulunanların en üstüne yerleştirilir; bir değer kuyruktan çıkarılırken de, çıkarılan değer daima kuyruğun tepesinde bulunan değer olacaktır.



Şekil 5.9 İç içe altyordamların diyagram biçimde gösterilmesi

Yığın mekanizması, altyordam dönüş adresleriyle ilgili işlemleri uygun şekilde yürütebilmek için ideal bir mekanizmadır; çünkü, bir CALL komutu yürütüldüğünde, bir sonraki adres (dönüş adresi) yığının (kuyruğun) tepesine yerleştirilir, böylece RET komutu yürütüldüğünde, buna karşılık gelen dönüş adresi her zaman için yığının başında olacaktır.

Yığına bir değer girmek değeri 'itme' (PUSH) ve yığından bir değer almak da değeri 'çıkarmak' (POP) olarak bilinir. Dolayısıyla, mikroişlemci bir CALL komutu yürüttüğünde, program sayıcısının yürürlükteki değerini (dönüş adresi) yığına iterek (PUSH) saklar ve sonra da altyordamın başlangıcına dallanır. Benzer biçimde, mikroişlemci bir RET (dön) komutu yürütürken de, önce dönüş adresini yığının başından çıkarır (POP), sonra da bunu program sayıcısına yükler.

Normalde bir mikrobilgisayar sisteminde yığın, bellek alanının bir parçası olarak gerçekleştirilir. Bu nedenle de, mikroişlemcide daima yığının başlangıcının bellekteki adresini tutan 16-bitlik *yığın işaretçisi* denilen özel bir kaydedici vardır. Böylece mikroişlemci, yığın işaretçisine erişip, üzerinde işlem yaparak yığına kolayca değer (dönüş adresleri) girebilir veya yığından değer çıkarabilir.

Altyordam parametreleri

Altyordam kullanımının temel nedenlerinden biri de sık kullanılan işlemleri gerçekleştirmek için gerekli program parçasının (kodun) yalnızca bir kez yazılmasıdır. Böylece kodun tekrar tekrar yazılması gerekmez. Öte yandan, bir altyordam tarafından işlenecek değerler, altyordamın her çağrılmasında değişebilir. Örneğin, iki değeri çarpacak bir altyordam yazarsak, kod aynı kalacak ancak altyordamın yürütüldüğü her seferde çağrılacak değerler, muhtemelen farklı olacaktır.

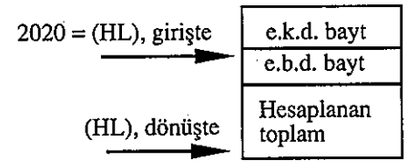
Bir altyordam tarafından işlenecek değerler *parametre* (ya da *argüman*) olarak adlandırılır ve bunlar, işlenmek üzere altyordama *geçirilir* biçiminde ifade edilir. Bu yüzden çağıran programla altyordam arasında parametre geçirmek ve altyordam tarafından üretilen sonuçları -yukarıdaki örnekte çarpma işleminin sonucu olan çarpımı- çağıran programa geçirmek için bir mekanizma kurmak gerekmektedir.

Değişik mekanizmalar mevcuttur, ama en bilinen yaklaşım bellekten bir alan kullanmaktır. Çağıran program işlenecek değerleri bilinen bir sırayla bu bellek blokuna (konumlar listesine) yerleştirir ve blokun başlangıç adresini altyordama geçirir. Ardından, altyordam bloktan (doğru sırada) işlenecek bu değerlere erişir, bunları işler ve sonuç(lar)ı (yine bilinen bir sırada) bloka yerleştirir. Daha sonra da çağıran program bloktaki sonuç(lar)a erişir ve böylece devam eder. Bu yaklaşım, yukarıda ele alınan bazı konuları göstermek için yazılan aşağıdaki program örneklerinde kullanılacaktır.

Program örneği 5.4 : Parametre geçirme

```
                .
                .
                .
        LD      HL,2050     HL'yi bellek blokunu göstermek üzere hazırla
        LD      (HL),00     Komutlar listesinin başlangıç adresini,
        INC     HL          2800 (onaltılı), bellek blokunda sakla
        LD      (HL),28
        DEC     HL          HL'yi bellek blokunun başlangıcını
                            gösterecek biçimde tekrar düzenle
        CALL    TOPLAM      TOPLAM altyordamını çağır
        LD      B,(HL)      Toplamı B'ye yükle ve devam et
                .
                .
                .
        DURDUR              Ana programın sonu
-------------------------------------------------------------------
TOPLAM: LD      E,(HL)      DE'deki komutlar listesinin başlangıç
                            adresini yükle
        INC     HL
        LD      D,(HL)
        (  )    (  )
        INC     HL          HL'deki başlangıç adresini al ve bellek
        EX      DE,HL        bloku işaretçisini DE'de sakla
        LD      B,00        Toplamı sıfırla
        (  )    (  )
DONGU:  LD      A,B         İlk (bir sonraki) değeri devreden toplama ekle
        ADD     A,(HL)
        LD      B,A         Devreden toplamı (ara toplamı) B'de sakla
        INC     HL          HL'i bir sonraki değeri gösterecek biçimde artır
        LD      A,L         100 değer de toplandı mı ?
        CP      64
        JP      NZ DONGU    Hayır: bir sonraki değeri toplamak için geri döngü

        EX      DE,HL       Evet : DE'den bellek blok işaretçiyi al
        LD      (HL),B      Toplamı bellek blokuna sakla
        RET                 Çağıran programa dön
```

Yukarıdaki program örneği, bellek bloku aracılığıyla bir altyordama parametre geçirme mekanizmasını göstermektedir. TOPLAM altyordamı, daha önce program örneği 5.1'de gösterilen program parçasında olduğu gibi, bellekte tutulan yüz tane değerin toplamını hesaplar. Değerler listesinin başlangıç adresi ile altyordamdan dönecek toplam (ki, 8-bit olduğu varsayılmıştır) 2050 (onaltılı) adresinden başlayan bir bellek blokunda saklanır. Bu aşağıda gösterilmiştir:



Mikroişlemci kaydedicileri altyordamda, daha önce program örneği 5.1'de kullanılan biçimde kullanılır. Dolayısıyla L kaydedicisinin, yüz değerin hepsinin de toplandığını belirten bir sayıcı olarak kullanıldığı varsayılmıştır. Farklı bir değerler listesi olduğunda da, bellekteki blokun başlangıç adresi altyordama parametre olarak geçirilerek kolayca hesaplanabilir. Öte yandan, L'yi sayıcı olarak kullanmaya devam edebilmek için, en küçük değerlikli adres baytı her zaman 00 (onaltılı) olmalıdır.

S,5.8-5.10

Program örneği 5.5 : İç içe altyordamlar

```
                .
                LD      HL,2800     Gecikme parametresini (XX),
                                    bellek konumuna 2800 (onaltılı) yükle
                LD      (HL),XX
                CALL    GECIK1      İlk altyordamı çağır
                .
-------------------------------------------------------------------
GECIK1:         LD      C,(HL)      Gecikme parametresini C'ye yükle
GECIK 2:        LD      A,C         C'nin değeri sıfır mı ?
                CP      00
                JP      Z,ZAMAN
                INC     HL          Hayır: ikinci gecikme parametresini
                LD      (HL),YY     (YY) 2801 (onaltılı) bellek konumuna
                                    yükle ve ikinci altyordamı çağır
                CALL    GECIK 2
                DEC     C           Gecikme parametresini azalt ve
                JP      GECIK       geriye döngüye devam et
ZAMAN:          RET                 Evet: çağıran programa dön
-------------------------------------------------------------------
GECIK 2:        LD      B,(HL)      Gecikme parametresini B'ye yükle
```

```
DONGU:   LD    A,B      B'nin değeri sıfır mı ?
         CP    00
         JP    Z,SON
         NOP            Hayır: Temel zaman gecikmesini işleme koy
         NOP
         NOP
         DEC   B        Gecikme parametresini azalt ve geriye döngüye
         JP    DONGU    devam et
SON:     RET            Evet : çağıran programa dön
```

Yukarıdaki program örneğinden, programcının bir altyordam içinden başka bir altyordamı çağırabileceği görülmektedir. Daha önce Şekil 5.4'te gösterilen programda hesaplanan zaman gecikmesini -bu üç NOP komutu yerine söz gelimi, bilinen bir sayıyla döngüleme işlemi konularak- uzatabilmek mümkün olsa da; burada gecikme, NOP komutları yerine bir gecikme altyordamını çağıran CALL komutu konularak sağlanmıştır. Bu nedenle, bilgisayar zaman gecikmesi, birinci altyordamda döngü gerçekleştiği her seferinde ikinci altyordam çağırılarak uzatılır.

GECIK1 birinci altyordam için gecikme parametresi 2800 (onaltılı) bellek adresinde ve GECIK 2 ikinci altyordam için gecikme parametresi 2801 (onaltılı) bellek adresinde geçirilir.

Yığın işlemleri

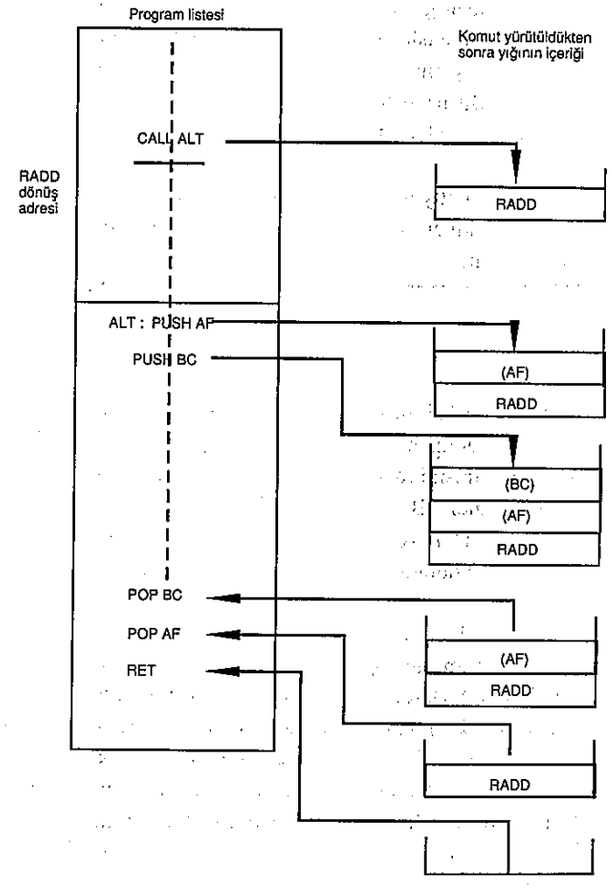
Program örneği 5.5 altyordamların kullanımıyla ilgili önemli bir noktaya ışık tutar. Örnekteki program listesinde de görüleceği gibi, ikinci altyordam (GECIK 2), gecikme parametresini tutmak için birinci altyordamın (GECIK 1) kullandığından farklı bir kaydedici (B kaydedicisi) kullanır. Aksi takdirde ikinci altyordam çalıştırıldığında C kaydedicisini sıfır yapacak, böylece dönüşte birinci altyordam yalnızca tek bir döngü yaptıktan sonra, vaktinden önce ana programa dönecektir.

Bu örnekte ikinci bir kaydedici kullanarak bunun oluşmasını önlemek mümkündü; ama hem ana program hem de altyordam(lar) mevcut olan kaydedicileri sürekli kullandıklarından, bu çoğu zaman mümkün olmayabilir. Bu nedenledir ki, bir altyordam yazılırken, altyordam içinde kullanılan kaydedicilerin yürürlükteki değerlerini altyordama girer girmez saklayıp, çağıran programa dönmeden önce de bu değerleri orijinal durumlarına getirmek yaygın bir alışkanlıktır. Böylece her program parçası eldeki tüm kaydedicileri, bir altyordam çağrıldığında bunların bozulmayacağını veya değiştirilmeyeceğini bilerek serbestçe kullanabilir.

Mikroişlemcinin bir bellek alanını, altyordam dönüş adreslerini tutmak için bir yığın olarak kullanmasının yanında, mikroişlemciler kullanıcıya, yığını çeşitli mikroişlemci kaydedicilerinin içeriklerini saklamak için geçici bir saklama alanı olarak kullanma imkanını sağlayan komutlar da sunar. Bu anlamda, Z80'de aşağıdaki kaydedici çiftlerinin yürürlükteki içeriklerini yığına saklamak için (PUSH) dört komut bulunmaktadır :

```
PUSH  AF
PUSH  BC
PUSH  DE
PUSH  HL
```

Benzer biçimde aşağıdaki komutlar da çeşitli kaydedici çiftlerini yığından geriye yüklemek içindir :



Şekil 5.10 Farklı komut türlerinin yığının içeriğine etkileri

```
POP AF
POP BC
POP DE
POP HL
```

Yığın, son-giren-ilk-çıkar düzeninde çalıştığından, altyordamı terketmeden önce doğru değerlerin doğru kaydedici çiftlerine geri alınmaları için programcının PUSH ve POP komutlarını kullanması esastır. Örneğin, bir altyordamın A kaydedicisi, (ve dolayısıyla F) ve C'yi kullandığı varsayılırsa, kullanıcı için normal olanı hem A,F kaydedici çiftinin hem de B,C kaydedici çiftinin içeriklerini altyordamın başında saklamak ve sonra da bitiminde bunları geri almaktır. Bu, Şekil 5.10'da görüldüğü gibi yapılır.

Bu şekilden de görüleceği gibi, POP komutları her zaman için bunlara karşılık gelen PUSH komutlarının tersi yönünde olmalıdır. Ayrıca, mikroişlemcinin kendisi, yığını, dönüş adresini saklamak için kullandığından, altyordamın sonunda kullanılan POP komutlarının, başlangıçta kullanılan PUSH komutlarıyla aynı sayıda olmasını sağlamak gerekmektedir. Dolayısıyla, mikroişlemci RET komutunu yürüttüğünde, yığından program sayıcısına olan doğru dönüş adresini geriye yükleyebilecektir.

S 5.11 Şekilde yığının içeriği, kolaylık açısından tek bir 16-bitlik sözcük olarak gösterilmiştir; fakat uygulamada, bunlar bellek içinde ardışık iki bayt olarak saklanırlar.

Sorular

5.1 EK-1'deki bilgileri kullanarak ve ilk komutun ilk baytının 2000 (onaltılı) bellek adresinde saklandığını varsayarak, aşağıdaki program parçasını onaltılık makine kodu biçimine çevirin:

```
DONGU: LD    A,E7
       INC   A
       JP    DONGU
       -
```

5.2 Aşağıdaki program parçası çalıştırıldıktan sonra B kaydedicisinin içereceği değeri bulun:

```
       LD    B,FF
SIFIRDGL: DEC B
       JP    NZ,SIFIRDGL
       -
```

5.3 İlk komutun ilk baytının 2020 (onaltılı) bellek adresinde saklandığını varsayarak, Soru 5.2'de verilen program parçasını onaltılık makine kodu biçimine çevirin.

5.4 Aşağıdaki program parçası çalıştırıldıktan sonra A kaydedicisinin içereceği son değeri ve yürütülmüş olacak komut sayısını bulun:

```
       LD    A,00
DONGU: INC   A
       JP    NC,DONGU
       -
```

5.5 1'den 10'a kadar olan sayıları toplamak için bir akış diyagramı tasarlayın ve sembolik (assembler) dilde bir program yazın. (*İpucu*: Devreden toplamı tutmak için mikroişlemci kaydedicilerinden birini ve bu toplama eklenecek bir sonraki sayıyı tutmak için de bir başkasını kullanın. Ardından program, bilinen bir sayıyla döngü kullanarak uygun biçimde tasarlanabilir).

5.6 A kaydedicisinin içeriği sıfırsa ona 15 (ondalık) ekleyen, sıfır değilse 20 (ondalık) ekleyen bir akış diyagramı tasarlayın ve sembolik (assembler) dilinde bir program yazın. Program parçası çalıştırıldığı sırada A kaydedicisinin değerinin bilinmiyor olduğunu varsayın.

5.7 Yerine getirilmesi gereken koşulu; "sıfırdan küçük veya eşitse A kaydedicisine 10 (ondalık) ekle, sıfırdan büyükse 20 (ondalık) ekle" şeklinde değiştirerek 5.6 numaralı soruyu bu değerlerle tekrar çözün.

5.8 Soru 5.5'te geliştirilen programı; sayı aralığı ve hesaplanan toplamı, başlangıç adresi H,L kaydedici çiftine aktarılan bir bellek blokuna parametre olarak geçirilecek bir altyordam şeklinde değiştirin.

5.9 Program örneği 5.2'yi üç temel gecikme komutunu (NOP'ları) basit bir döngüleme işlemiyle değiştirmek kaydıyla düzeltin. Döngü, bilinen 100 sayısıyla döngü şeklinde düzenlenmelidir.

5.10 Eğer her bir komutu getirip yürütmek 4 mikrosaniye alıyorsa, Soru 5.9'da geliştirilen programın hesaplayacağı maksimum ve minimum zaman gecikmesini belirleyin.

5.11 Soru 5.8'deki altyordamı değiştirerek parametrelerin yığın aracılığıyla geçirilmesini sağlayın, yani toplanacak sayıların aralığı, altyordamı çağırmadan önce ve sonuç, dönüş yapılmadan önce yığına itilmelidir (PUSH). (*İpucu* : Altyordam yürütülmeye başladığında dönüş adresinin yığının ba-şında olması ve dolayısıyla hesaplama yapılırken çıkarılıp (POP) saklan-ması gerektiğini anımsayın.

6. Bölüm : G/Ç Komutları ve ilişkili programlama teknikleri

Bu bölümün amaçları : *Bu bölümü bitirdiğinizde:*

1 *Bir mikroişlemcide bulunan giriş ve çıkış komutlarını anlayabilmeli ve kullanabilmeli,*
2 *Sayısal sinyaller üreten veya kullanan tipik cihazları tanımlayabilmeli ve bir mikrobilgisayar ile bunlar arasında arabirim sağlayan kısa programlar yazabilmeli,*
3 *Bir ondalık ve onaltılı tuştakımı arabirim devresinin işlevini açıklayabilmeli ve bu tip tuştakımlarından bir basamak dizisi girmek için bir program yazabilmeli,*
4 *Mikrobilgisayarlarda kullanılan alternatif görüntü birimlerini açıklayabilmeli ve bunların kullanımlarını göstermek için programlar yazabilmeli,*
5 *Verinin bir mikrobilgisayar tarafından bit dizisi biçiminde giriş/çıkışının nasıl gerçekleştirildiğini anlayabilmeli,*
6 *Asenkron iletim terimini ve bir UART tarafından yerine getirilen işlevleri açıklayabilmeli,*
7 *Bir DAC'nin çalışma ilkesini ve analog bir değerin bir mikrobilgisayar tarafından nasıl üretildiğini tanımlayabilmeli,*
8 *Bir ADC'nin çalışma ilkesini ve analog bir gerilim değerinin bir mikrobilgisayar tarafından nasıl izlenebildiğini tanımlayabilmelisiniz.*

6.1 Giriş

Bir mikrobilgisayar, sayısal veri girişlerini okuyan ve bu verileri, belleğinde saklı komutlar listesine (program) göre işleyen ve elde edilen sonuçların sayısal çıkış verisi biçiminde çıkışını sağlayan bir birim olarak düşünülebilir. Ayrıca, farklı uygulamalar için farklı giriş/çıkış birimleri söz konusu olsa da, bu durum, mikrobilgisayarın çalışmasını olumsuz yönde etkilemeyecektir; çünkü, mikrobilgisayarla G/Ç birimleri arasında, uygulamadan bağımsız standart bir sayısal G/Ç mekanizması oluşturacak şekilde uygun arabirim devreleri kullanılmaktadır. Böylece, uygun miroişlemci komutlarını kullanmak suretiyle, uygulamaya ve kullanılan arabirim devrelerine bağlı olarak farklılık gösterebilen çeşitli birimlerden sayısal (ikili) değer giriş ya da çıkışı yapılabilir veya bu değerler, bu birimler tarafından kullanılabilir.

Bu bölümde; normalde bir mikroişlemcide bulunan temel giriş ve çıkış komutları ele alınmakta; ayrıca, hem mikrobilgisayarda kullanılan tipik G/Ç birimlerini hem de bunları kullanmak için gereken ilişkisel programlama teknikleri tanımlanmaktadır.

6.2 G/Ç Portları

Bir mikroişlemcide bulunan G/Ç I/o komutları 8-bitlik ikili kodlanmış bir değerin, belirtilen bir porttan girişini ya da bundan çıkışını yapar. Bu yüzden, en basit biçimiyle G/Ç arabirim ünitesi, Şekil 6.1'de de görüldüğü gibi, mikrobilgisayar yoluna bağlı giriş ve çıkış portlarının grubu olarak düşünülebilir.

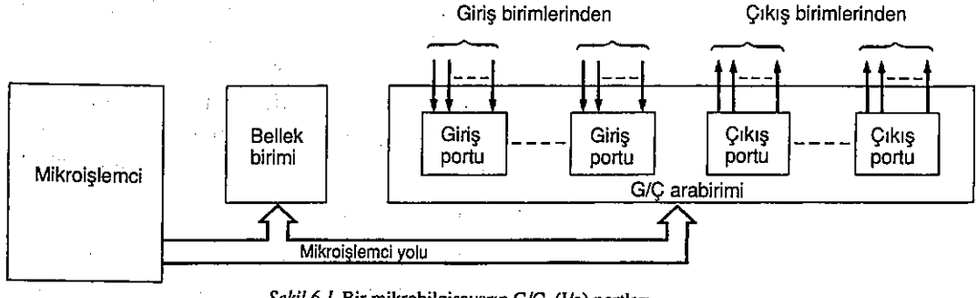
Mevcut port sayısı normalde çok fazla sayıda olabilir - tipik olarak 256 (ondalık). Dolayısıyla, her ne kadar birçok uygulama için gerçekte kullanılan port sayısı çoğu zaman oldukça az olacaksa da, bir mikroişlemcide bulunan giriş ve çıkış komutları normalde 00-FF (onaltılı) aralığında bir giriş veya çıkış portunu belirtebilir. Böylece, örneğin, Z80'de bulunan tipik bir giriş komutu :

IN A,(01)

şeklindedir. Bu komutun sonucu şudur : port adresi [bu örnekte 01 (onaltılı)] mikroişlemci tarafından mikrobilgisayarın adres hatlarına yerleştirilir, G/Ç arabirim ünitesi buna, adreslenmiş portun girişinde hazır bulunan o anki 8-bit değerini veri yoluna hatlarına koyarak karşılık verir. Ardından bu değer mikroişlemci tarafından A kaydedicisine yüklenir. Dolayısıyla eğer portun girişinde bulunan o anki değer, söz gelimi, E7 (onaltılı) ise, yukarıdaki komut yürütüldükten sonra A kaydedicisinin içereceği değer E7 (onaltılı) olacaktır. Benzer şekilde, Z80'de sunulan tipik bir çıkış komutu da:

OUT (02),A

biçimindedir. Bu komut, mikroişlemcinin A kaydedicisinin o anki değerinin mikrobilgisayar veri yoluna ve port adresinin de [02 (onaltılı)] adres yoluna çıkışına neden olur. Daha sonra G/Ç arabirim ünitesi bu değeri, seçilmiş port tampon kaydedicisine yükler -diğer bir deyimle kilitler. Dolayısıyla, eğer A kaydedicisinin içerdiği değer, yukarıdaki komut yürütülmeden önce, sözgelimi 7E (onaltılı) ise, komut yürütüldükten sonra bu değer 02 (onaltılı) port kaydedicisine yüklenecektir.



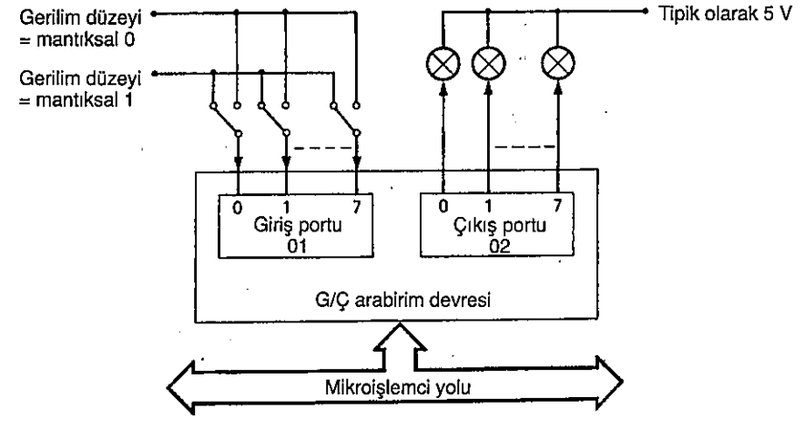
Şekil 6.1 Bir mikrobilgisayarın G/Ç (I/o) portları

Giriş ve çıkış portları iki önemli işlev görür: giriş yapılması durumunda, mikroişlemci bir porta ilişkin bir IN komutu yürütene kadar, bir giriş birimi veya arabirim devresi tarafından bu port için üretilen veri değerlerinin, mikrobilgisayar yolundan izole edilmesi için kullanılırlar. Çıkışta da, çıkış birimi veya arabirim devresi almaya hazır hale gelene dek mikroişlemci tarafından çıkışı yapılan bir veri değerinin tutulması (mandallanması) için kullanılırlar. Bu nedenle, mikrobilgisayar yolu ile uygulamada kullanılmakta olan G/Ç birimi ve arabirim devreleri arasında bir tampon görevi gördüklerinden, giriş ve çıkış portları G/Ç tamponları olarak anılırlar.

6.3 Sayısal giriş ve çıkış

G/Ç arabirim portlarına giriş ve çıkış yapılan veriler sayısal nitelikte (iki durumlu) olduğundan, bu sayısal cihazlarla bir mikrobilgisayar arasında kolayca bir arabirim sağlamak mümkündür. Dolayısıyla da sekize kadar olabilecek bir dizi anahtar (açık-kapalı) veya basmalı-düğme (açık-kapalı) durumu, bir giriş portundan doğrudan girilebilir. Benzer biçimde, sekize kadar bir ışık yayan diyot (LED) dizisi de, bir çıkış portundan çıkan ikili kodlanmış bir değeri görüntülemek üzere doğrudan sürülebilir.

Buna alternatif olarak, uygun arabirim devrelerinin portlara bağlanmasıyla, bir giriş portundan girilen ikili bir değer, sözgelimi denetimli işleme bağlı bir grup düzey anahtarlarının veya termostatların yürürlükteki durumlarını gösterebilir. Benzer şekilde, bir porta ilişkin ikili bir çıkış değeri de bir grup röle veya solenoidli valfin çalıştırılması için kullanılabilir. Bazı alternatif G/Ç birimleri ve bunları denetlemek için kullanılan ilişkisel programlama teknikleri aşağıda açıklanacaktır.



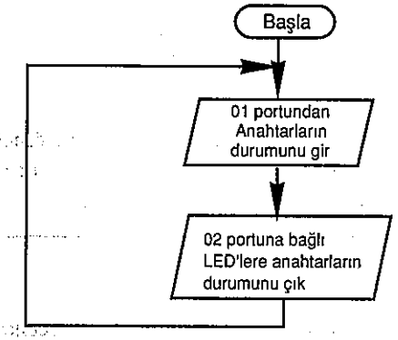
Şekil 6.2 Anahtarlar ve LED'ler kullanan sayısal G/Ç

Bir grup anahtar veya LED'den gelen sayısal giriş ve çıkış

Giriş ve çıkış birimlerinin en basit biçimleri, ikili bir değeri doğrudan girebilmek için bir grup anahtar yardımıyla mikrobilgisayardan çıkan ikili bir değeri görüntülemek için kullanılan LED grubudur. Normalde bu cihazlar ek arabirim devrelerine gerek duymaksızın sırasıyla bir giriş ve çıkış portuna doğrudan bağlanabilirler. Tipik bir düzenleme Şekil 6.2'de gösterilmiş ve bu devreyi kullanan bazı program örnekleri aşağıda verilmiştir.

Program Örneği 6.1

Şekil 6.3'te gösterilen akış diyagramı ve program kodu, sekiz anahtarın durumunu sürekli olarak okuyan ve okunan bu değerleri,



Sembolik komutlar		İşlem
DONGU:IN	A, (01)	01 Portundan A'ya anahtarların durumunu gir
OUT	(02),A	A'nın değerini 02 portuna çık
JP	DONGU	Döngüye devam et

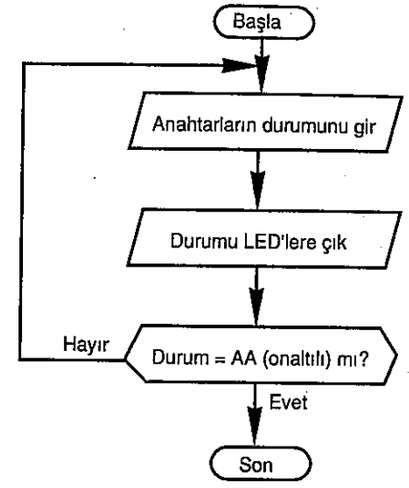
Şekil 6.3 Program örneği 6.1

Şekil 6.2'de görülen sekiz tane LED'e çıkan bir program içindir. Anahtarların durumu program döngüsünün içinde girildiğinden, durumlardaki herhangi bir değişiklik ilgili LED'in durumuna anında yansıtılacaktır.

Program Örneği 6.2

Program örneği 6.1'de tanımlanan program açıkça sonsuz bir döngüde anahtarların durumunu okuyup bu durumları LED'lere çıkmaktadır. Öte yandan, Şekil 6.4'de gös-

Simgesel komutlar			İşlem
DONGU :	IN	A,(01)	Anahtarın durumunu A'ya gir
	OUT	(02),A	A'nın değerini 02 portuna çık
	CP	AA	Durum = AA (onaltılı) mı?
	JP	NZ, DONGU	Hayır : döngüye devam et
	HALT		Evet : son



Şekil 6.4 Program örneği 6.2

terilen program, anahtarlardan belli bir desen [bu örnekte AA (onaltılı)] okunana kadar anahtarların durumunu okur ve bu durumları LED'lerde görüntüler. Program
S 6.1 daha sonra , yani AA okunduktan sonra son bulur.

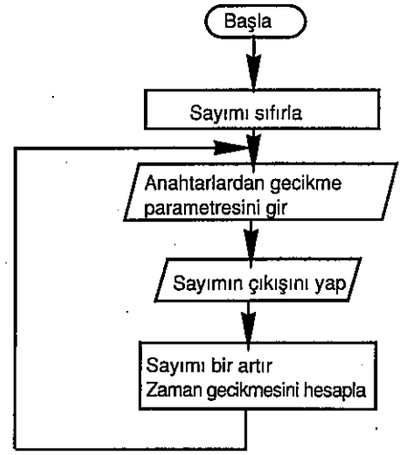
Program Örneği 6.3

Şekil 6.5'teki programda, hesaplanan bir zaman gecikmesinin bir program içinde nasıl kullanılabileceği gösterilmektedir. Program, 00'dan FF'e (onaltılı) kadar sayar ve yürürlükteki sayıyı, sayma döngüsü içindeyken LED'lerde görüntüler. Öte yandan, basit bir sayma döngüsü çok hızlı olduğundan, ışıklı göstergenin en küçük değerlikli bitleri, bunu takip edemeyecek kadar hızlı değişecektir. Bu yüzden bu örnekte anahtarlar, 8-bitlik değer girmek için kullanılır. Bu değer daha sonra sayımdaki ardışık değerlerin görüntülenmesi arasında oluşan gecikmeyi hesaplamak için parametre olarak kullanılır. Dolayısıyla, anahtarlardaki değer büyüdükçe sayma işlemi daha yavaş olacaktır.

Bilgisayar gecikme programı daha önce Şekil 5.4'de gösterilenin bir benzeridir. Program tarafından kullanılan kaydediciler; (genel çalışma kaydedicileri olarak) A , (yürürlükteki sayıyı tutmak için) B ve (gecikme parametresini tutmak için) C kay-
S 6.2 dedicileridir.

Denetimli işlemden sayısal giriş çıkış

Mikrobilgisayarın giriş ve çıkış portları arasında elektriksel bir arabirim sağlayan uygun devrelerle, daha önce ele alınan sayısal girişler bir grup anahtardan türetilmek



Sembolik komutlar			İşlem
	LD	B,00	Sayımı sıfırla
DONGU :	IN	A,(01)	Anahtarlardan gecikme parametresini gir
	LD	C,A	Gecikme parametresini C'ye sakla
	LD	A,B	
	OUT	(02),A	Yürürlükteki sayımın çıkışını yap
	INC	B	Sayımı bir artır
	LD	A,C	Zaman gecikmesini hesapla
GECIK :	CP	00	
	JP	Z, DONGU	
	NOP		
	NOP		
	NOP		
	DEC	A	
	JP	GECIK	

Şekil 6.5 Program örneği 6.3

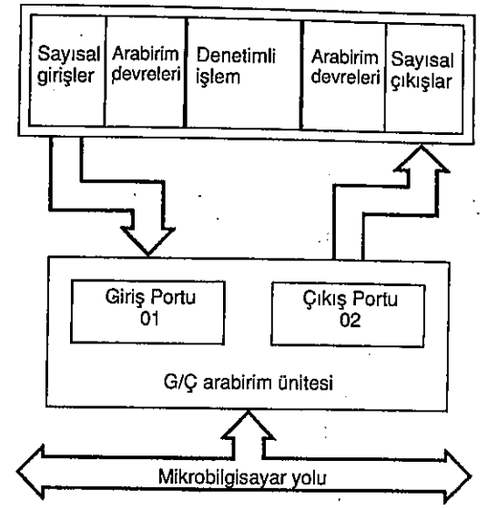
yerine kolayca, sözgelimi bir mikrobilgisayar tarafından denetlenen denetimli bir işlemle birlikte çeşitli sayısal cihazlardan türetilebilir.

Örneğin; bir grup termostat (açık-kapalı) veya sınır anahtarları (açık-kapalı). Benzer biçimde, sayısal çıkışlar işlemle birlikte kullanılan çeşitli solenoid valflerin (kapalı/ açık) ve rölenin (açık-kapalı) durumunu denetlemek için kullanılabilir. Şekil 6.6'da, bu durum diyagram olarak gösterilmiştir.

Bu veriler üzerinde işlem gören giriş ve çıkışın programlama teknikleri, açıkça görüldüğü gibi, az önce ele alınanlardan farklıdır. Şimdi denetimli bir işlemden giriş çıkışlarla ilişkili tipik işlem örnekleri ele alınacaktır.

Program Örneği 6.4 : Sıralama - I

Bir işlemi denetlerken yapılan en yaygın işlerden biri tipik olarak mikrobilgisayar belleğindeki bir tabloda saklı bir durum sırasının çıkışını yapmaktır. Örneğin; tab-

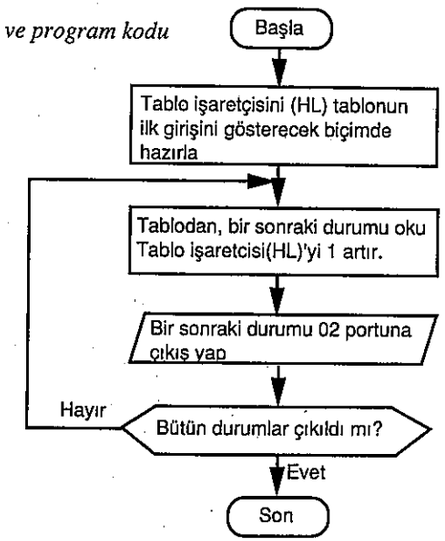


Şekil 6.6 Denetimli bir işlemden sayısal giriş ve çıkış

a) Durum tablosu

Durum No :	Denetimli ünite durumları							
	8	7	6	5	4	3	2	1
0	0	0	0	1	0	1	0	1
1	0	0	1	0	1	0	1	1
2	0	0	1	1	1	1	1	1
3	0	1	0	0	1	0	0	1
4	0	1	0	1	0	1	0	0
5	1	1	0	0	0	0	1	0
6	0	0	0	0	0	0	0	0

b) Akış diyagramı ve program kodu



Sembolik komutlar			İşlem
	LD	HL,2800	H,L'yi tablonun ilk değerine getir
TEKRAR	LD	A,(HL)	Tablodan bir sonraki durumu oku
	OUT	(02),A	Bir sonraki durumu 02 portuna çık
	INC	HL	Tablo işaretçisini 1 artır
	LD	A,L	Tüm durumların çıkışı yapıldı mı?
	CP	07	
	JP	NZ TEKRAR	Hayır : Tekrarla
	HALT		Evet : Son

Şekil 6.7 Program örneği 6.4

loda saklı bulunan her bir değer, mikrobilgisayar çıkış portuna bağlanmış bir grup valf solenoidlerin veya rölenin durumunu (açık-kapalı) gösterebilir. Bundan dolayı bu denetim işlevi *sıralama*, mikrobilgisayar da *sıra denetleyicisi* olarak bilinir.

Böylece sekiz denetimli cihazdan tipik işlem sırası örneği, Şekil 6.7(a)'daki gibi gösterilebilir. Normalde bu, sistemin sırası veya *durum tablosu* olarak anılır. Dolayısıyla, tablodaki ilk kayıt 1, 3 ve 5 numaralı cihazların açık; 2,4,6,7 ve 8 numaralı cihazların da kapalı olması gerektiğini gösterir. Benzer şekilde, tablodaki bir sonraki kayıt, çıkışı yapılacak bir sonraki durumu belirtir ve bu kez 1,2,4 ve 6 numaralı cihazların açılması; 3,5,7 ve 8 numaralı cihazların da kapatılması gerektiğini gösterir. Daha sonra bu, sıranın sonunda tüm cihazlar kapatılana kadar (7. durum) devam eder.

Böylece bu sıralamayı yerine getirecek bir akış diyagramı ve buna bağlı program kodu Şekil 6.7(b)'de gösterilmiştir. Program her yeni durumu bellek adresi 2800'den (onaltılı) başlayarak saklanacak olduğu varsayılan tablodan okur ve daha sonra bu durumu (değeri) çıkış portuna çıkarır. Bu işlem, yedi durumun hepsinin çıkışı yapılana kadar tekrarlanır.

S 6.3

Şekil 6.7'de görülen program, tablodaki her bir yeni ünite durumları grubuna erişmek için bir döngüleme işlemi gerçekleştirir ve ardından da bunu, uygun çıkış portuna çıkarır. Böylece çıkışı yapılan her bir yeni ünite durumları grubu arasındaki zaman gecikmesi çok kısa olacaktır ve birçok uygulama için bu açıkça yetersiz kalacaktır. Bu yüzden bu tip bir uygulamada, mikrobilgisayar için her bir yeni durumun çıkışı arasında bir zaman gecikmesi hesaplamak olağandır. Öte yandan, her bir durum arası için istenen zaman gecikmesi farklı olacağından, bu gecikmeleri belirtmek de gerekmektedir. Bu durum bir sonraki örnekte ele alınacaktır.

Program örneği 6.5 : Sıralama - II

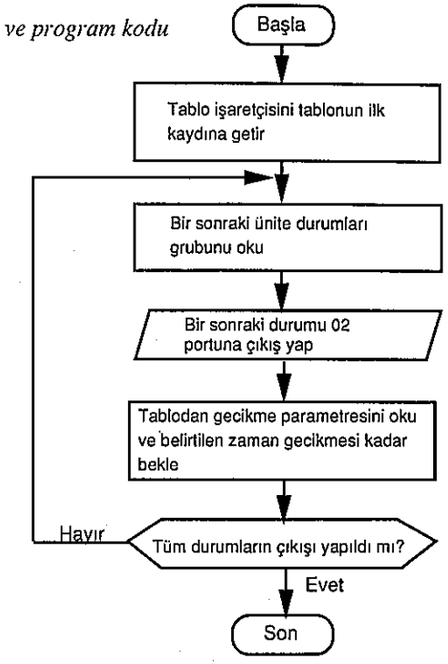
Şekil 6.8(a)'daki durum tablosunda, sekiz denetimli ünitenin istenen durum sırası ve her bir durum arası için ilgili zaman gecikmesi gösterilmiştir. Bu yüzden Şekil 6.8 (b)'de görülen akış diyagramı ve program kodu, her bir durumun çıkışını yapar ve tabloda belirtilen ilgili zaman gecikmesini daha önce Şekil 5.4'te verilene benzer bir zaman gecikme altyordamı için parametre olarak kullanır. Altyordamda kullanılan temel gecikme bir saniyeye eşit olmalıdır; dolayısıyla, gecikme parametresi altyordam içinde gerçekleştirilecek döngüleme sayısını gösterir. Belirtilen zaman değerleri ondalık olduğundan, bunlar tabloda saklanmadan önce onaltılı biçime dönüştürülmelidir.

Programda; durum tablosu verisinin bellekte 2800 (onaltılı) adresinden başlayan bir

a) Durum tablosu

Durum no :	Durumlar arası zaman gecikmesi	Denetimli Ünite durumları							
		8	7	6	5	4	3	2	1
0	1	0	0	0	1	0	1	0	1
1	20	0	0	1	0	1	0	1	1
2	120	0	0	1	1	1	1	1	1
3	120	0	1	0	0	1	0	0	1
4	1	0	1	0	1	0	1	0	0
5	250	1	1	0	0	0	0	1	0
6	0	0	0	0	0	0	0	0	0

b) Akış diyagramı ve program kodu



Sembolik komutlar			İşlem
	LD	HL,2800	Tablo işaretçisini hazırla
TEKRAR :	LD	A,(HL)	Tablodan bir sonraki ünite durumunu oku
	INC	HL	
	OUT	(02),A	Bir sonraki durumu 02 portuna çıkış yap
	LD	C, (HL)	Tablodan gecikme parametresini oku ve belirlenen zaman gecikmesi kadar bekle
	CALL	GECİKME	
	INC	HL	Tüm durumların çıkışı yapıldı mı?
	LD	A,L	
	CP	DE	
	JP	NZ TEKRAR	Hayır : Tekrarla
	HALT		Evet : son
GECİK:	-		Şekil 5.4' deki ne benzer bir zaman gecikme altyordamı
	-		
	.		
	.		
	.		
	RET		

Şekil 6.8 Program örneği 6.5

tabloda saklandığı varsayılmıştır. Kolaylık açısından her bir cihaz durumu grubunun ardından istenen zaman gecikmesi parametrelerinin verildiği tek bir tablo kullanılmıştır. Dolayısıyla (2800 adresindeki) tablodaki ilk değer, istenen ünite durumlarını göstermek üzere 15 (onaltılı) olacak; bir sonraki (yani 2801 adresindeki) 01 (onaltılı) ise, istenen zaman gecikmesini gösterecektir. Böylece durum grubunun sonuncusuyla buna bağlı zaman gecikmesi değerleri bellekte sırasıyla 280C (onaltılı) ve 280D (onaltılı) adreslerinde saklanacak ve bu nedenle de L kaydedicisi 0E
S 6.4 (onaltılı) değerine ulaştığında döngü işlemi sona erecektir.

Yukarıdaki iki örnekte de temel veya *koşulsuz* sıralamayı gerçekleştirmek için kullanılan tipik programlama işlemleri gösterilmiştir. Öte yandan bazı sıralama uygulamalarında, her bir yeni ünite grubunun çıkışı arasında, belirtilen bir zaman gecikmesi kadar beklemek yerine, belli giriş koşullarının (ünite) ortaya çıkmasını beklemek gerekir.

Örneğin çıkış durumları grubu, bir grup solenoid valfinin çalışmasını ve bunlar da bir sıvının çeşitli tanklara akışını denetliyorsa, her bir yeni çıkış durumu arasında, birleşik bir düzey anahtar grubuna ilişkin durumların belli bir koşula ulaşmasını beklemek gerekebilir. Dolayısıyla, bu işlem modu *koşullu sıralama* olarak bilinir ve aşağıda buna bir örnek verilmiştir.

Program Örneği 6.6 : Koşullu Sıralama

Şekil 6.9(a)'da görülen tabloda, sekiz denetimli birimin her bir yeni durumunun çıkışı yapıldıktan sonra, gereken sekiz giriş ünite grubunun koşullu durumları listelenmiştir. Bu nedenle de ünite ilk grubunun çıkışını yaptıktan sonra mikroişlemci, ikinci ve daha sonraki ünite gruplarının çıkışını yapmadan önce, 3,4, ve 6 numaralı giriş birimlerin açık; 1,2,5,7 ve 8 numaralı giriş birimleri kapalı olana kadar beklemelidir.

Bu işlemi gerçekleştirecek programın akış diyagramı ve program kodu, Şekil 6.9 (b)'de gösterilmiştir. Görüldüğü gibi, her bir yeni durumu çıkışa verdikten sonra, program belirli giriş koşulu sağlanana kadar sürekli bir döngüde bekler.

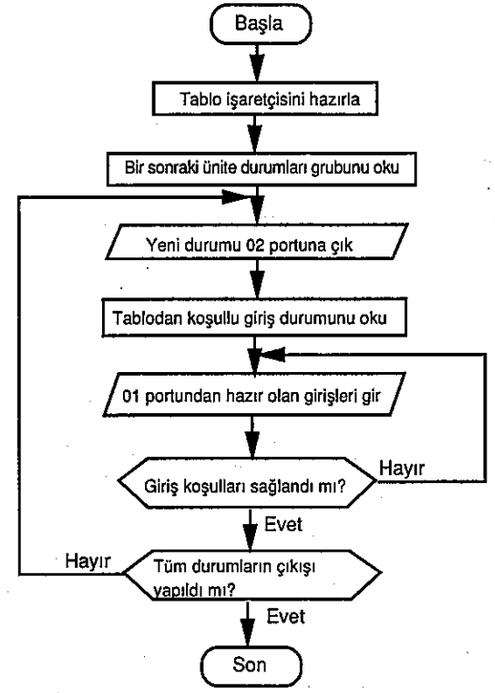
Programda, çıkış durumlarının bellekte, 2800 (onaltılı) adresinden başlayan bir tabloda saklandığı varsayılmıştır. Ve yine, program verimliliği açısından, sekiz giriş cihazının koşullu durumlarının her bir grubu, tabloda her bir ilgili çıkış ünite grubundan hemen sonra saklanmıştır. Böylece (2800 adresindeki) tablodaki ilk değer, ilk çıkış birimi durumları grubuna karşılık gelen 3C (onaltılı) ve (2801 adresindeki) bir sonraki değer de gerekli olan koşullu giriş durumlarına karşılık gelen 2C (onaltılı) değerleridir. Dolayısıyla, L kaydedicisi OE'ye (onaltılı) artırılmış olduğunda,
S 6.5 tüm durumların çıkışı yapılmış olacaktır.

---

a) Durum tablosu

Durum no :	Koşullu giriş durumları								Denetimli ünite durumları							
	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
0	0	0	1	0	1	1	0	0	0	0	1	1	1	1	0	0
1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
2	0	1	0	0	0	0	1	1	0	1	0	0	0	0	1	0
3	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
4	0	0	1	0	1	1	0	0	0	0	1	0	1	1	0	0
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b) Akış diyagram ve program kodu



Sembolik komutlar			İşlem
	LD	HL,2800	Tablo işaretçisini hazırla
TEKRAR :	LD	A,(HL)	Tablodan bir sonraki cihaz durumunu oku
	INC	HL	
	OUT	(02),A	Yeni sonraki durumu 02 portuna çıkış yap
	LD	B, (HL)	Tablodan koşullu giriş durumunu oku
	INC	HL	
BEKLE :	IN	A, (01)	Hazır olan girişleri 01 portundan gir
	CP	B	Giriş koşulları sağlandı mı?
	JP	NZ BEKLE	Hayır : Koşullar sağlanana kadar bekle
	LD	A,L	Evet : Tüm durumların çıkışı yapıldı mı?
	CP	0E	
	JP	NZ TEKRARLA	Hayır : Tekrarla
	HALT		Evet : son

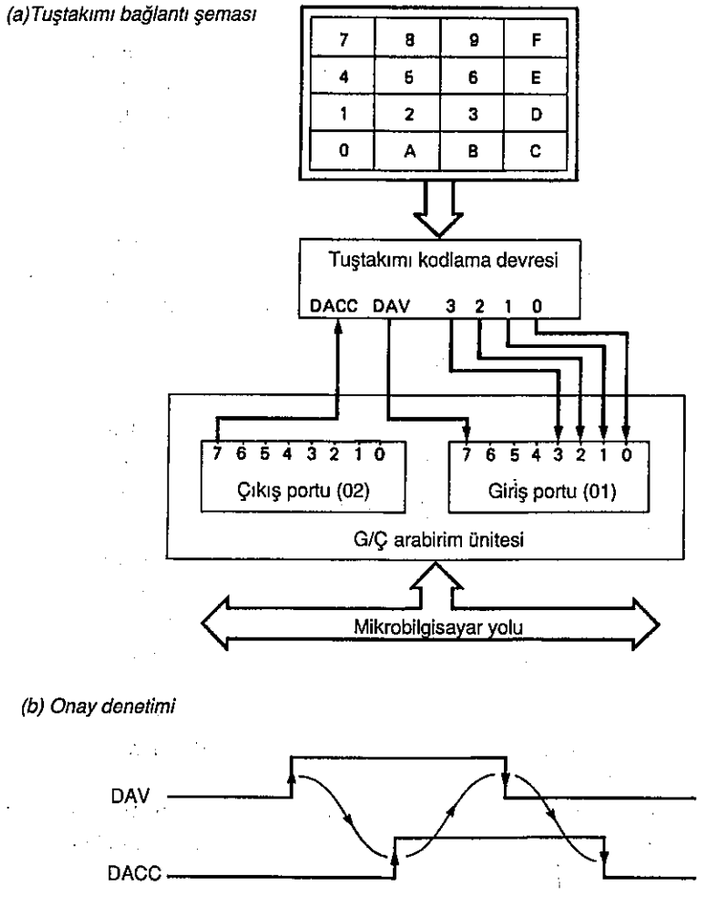
Şekil 6.9 Program örneği 6.6

6.4 Bazı alternatif giriş ve görüntüleme cihazları

Bir önceki kısımdan da görülebileceği gibi, birtakım mikrobilgisayar uygulamaları için, basit ikili desen veya değerlerin giriş çıkışı tümüyle yeterlidir. Öte yandan diğer bazıları, hem işlenecek verinin kaynağı için, hem de buna bağlı sonuçların görüntülenebilmeleri için daha karmaşık olanaklar gerektirirler. Burada bazı alternatif cihaz ve teknikler ele alınacaktır.

Bir tuştakımından sayısal giriş

Bir anahtar grubu, mikrobilgisayara ikili bir değer girmenin en basit yolu olsa da;



Şekil 6.10 : Bir tuş takımından sayısal giriş

sözgelimi, bir grup ondalık basamağın girişinin gerektiği uygulamalar için bu kullanışsız ve zaman alıcı olacaktır. Bu nedenle, bir grup ondalık basamak girişi birçok uygulamada ortak bir gereksinim olduğundan, bir tuştakımından belli bir tuşun seçimini, mikrobilgisayar tarafından okunabilecek ikili kodlanmış biçime dönüştürecek kodlama işlemini gerçekleştiren özel arabirim devreleri bulunmaktadır.

Örneğin; ondalık veya onaltılı bir tuştakımı kodlayıcı devresi, on veya on altı tuşlu bir tuştakımından seçilen ondalık veya onaltılı bir sayının 4-bitlik ikili kodlanmış eşdeğerini üretir. Bundan başka, tuştakımından bir dizi basamak girilebileceğinden yeni bir basamağın seçildiğinin belirlenmesi gerekir ki,bu yeni basamak girilmeden önce son girilen basamağın ikili kodlanmış eşdeğeri mikrobilgisayar tarafından okunabilsin.

Normalde bu iş, yeni bir basamak seçildiğinde bunu mikrobilgisayara belirtmek için ayrı bir denetim hattı üzerinde bir sinyal üreten devre tarafından gerçekleştirilir. Buna karşılık mikrobilgisayar da basamağı okuduğunda başka bir denetim hattı üzerinde sinyal üretir. Bu yöntem veya *protokol*, *onay* olarak bilinir ve iki cihaz arasındaki veri aktarımını senkronize etmek için sayısal sistemlerde yoğun biçimde kullanılır.

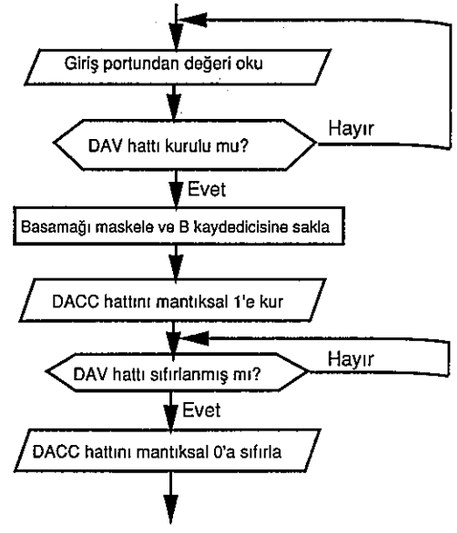
Bir giriş veya bir çıkış portu kullanarak onaltılı bir tuştakımı ile birlikte bulunan kodlama devresinin bir mikrobilgisayara nasıl bağlanacağı şematik olarak Şekil 6.10(a)'da gösterilmiştir. Görüldüğü gibi, onay denetim hatları giriş ve çıkış portlarının 7. bitlerine bağlanmıştır. Normalde, bunlar sırasıyla veri-kullanılabilir (DAV) ve veri-alındı (DACC) hatları olarak anılır.

Tuştakımı üzerindeki bir tuşa basıldığında, tuştakımı kodlama devresi dört veri hattı üzerinde 4-bitlik ikili kodlanmış eşdeğerde bir basamak üretir ve mikrobilgisayara DAV hattı üzerinden yeni bir basamağın girildiğini ve okunmaya hazır olduğunu sinyalle bildirir. Mikrobilgisayar, DAV hattının kurulduğunu algılar, basamağı okur ve kodlama devresine basamağın okunduğunu bildirmek için DACC hattını kurarak karşılık verir. Kodlayıcı devre, bunu basamağın okunmuş olduğuna ilişkin bir alındı bildirimi olarak yorumlar ve dolayısıyla DAV hattını sıfırlar. Son olarak, mikrobilgisayar DAV hattının sıfırlandığını algılar ve yeni bir aktarıma hazır hale getirmek için DACC hattını sıfırlar.

Program Örneği 6.7 : Tuştakımından bir basamak okuma

Şekil 6.11'deki program parçasında, daha önce Şekil 6.10(a)'da gösterildiği gibi, bir mikrobilgisayarın G/Ç portlarına bağlanmış onaltılı bir tuştakımından bir basamağın nasıl okunacağı gösterilmiştir. Program önce DAV hattı (7. bit) kurulana kadar giriş portundaki değeri okumak için döngü yapar. Daha sonra, basamağı alır ve DAV'ın

a) Akış diyagramı



b) Program kodu

Sembolik komutlar			İşlem
GIRIS :	IN	A,(01)	Giriş portundan değeri oku
	LD	B,A	B kaydedicisinde sakla
	RLA		DAV hattı 1mi?
	JP	NC, GIRIS	Hayır : DAV hattı kurulana kadar bekle
	LD	A,B	Evet : Basamağı maskele ve B kaydedicisinde sakla
	AND	OF	
	LD	B,A	DAC hattını 1'e kur
	LD	A,80	
	OUT	(02),A	
DAVBEKLE:	IN	A,(01)	DAV hattı sıfırlanmış mı?
	RLA		
	JP	C, DAVBEKLE	Hayır : DAV hattı sıfırlanana kadar bekle
	LD	A,00	Evet : DACC hattını sıfırla
	OUT	(02),A	

Şekil 6.11 Örneği 6.7

sıfırlanmasını beklemek üzere döngüye girmeden önce DACC hattını (çıkış portunun 7. biti) kurar. Son olarak, DAV sıfırlandığında program onay saykılını tamamlamak üzere DACC hattını sıfırlar.

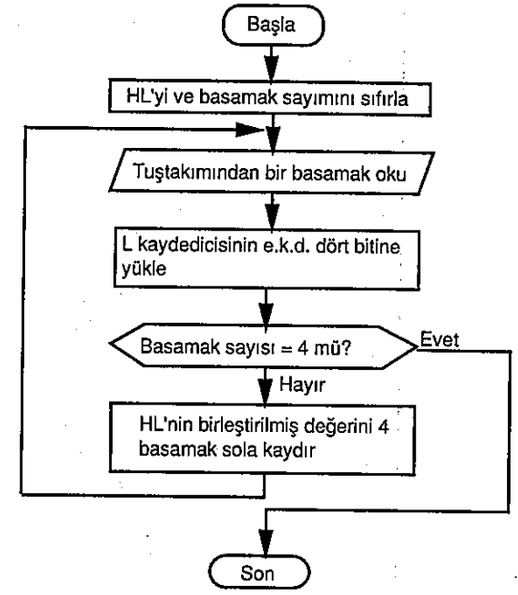
S 6.6

Yukarıdaki program örneği tek bir basamağın bir tuştakımından nasıl okunabileceğini göstermektedir. Öte yandan çoğu zaman, tuştakımından girilen bir basamak dizisinin işlenmeden önce okunması gerekmektedir. Bu işlem, tek bir basamağı okumak için geliştirilen kodu bir altprograma dahil ederek ve daha sonra dizideki her bir basamağı okumak için bu altyordamı çağırarak tamamlanır. Son basamak, ya dizideki basamak sayısını tanımlayıp basit bir sayım mekanizması kullanarak, ya da okunduğunda dizinin bittiğini ve başka işlemlere hazır olduğunu belirtecek özel bir sonlandırma basamağı tanımlayarak belirlenir. Bu durum aşağıdaki örnek aracılığıyla gösterilmiştir.

Program örneği 6.8: Bir tuştakımından bir basamak dizisinin okunması

Şekil 6.12'de görülen akış diyagramı ve program kodu, Şekil 6.11'den türetilen program parçasını, bir tuştakımından girilen dört tane onaltılı basamaktan oluşan bir dizinin okunmasını sağlayan altyordam olarak kullanır. Bu dört basamak, tipik olarak 16-bitlik bir bellek adresini gösterebilir; dolayısıyla, birleştirilmiş bu 16 bit, H,L kaydedici çiftinde saklanır. Girilen ilk basamağın, en büyük değerlikli basamağı gösterdiği ve bu nedenle de H kaydedicisinin en büyük değerlikli yarısında saklandığı varsayılmıştır.

a) Akış diyagramı



b) Program kodu

Sembolik komutlar			İşlem
	LD	HL,0000	H,L'yi ve basamak sayımını (c) sıfırla
	LD	C,00	
TEKRAR :	CALL	GIRIS	Tuştakımından bir basamak oku (B'de kalan)
	LD	A,B	L kaydedicisinin e.k.d. 4 bitine yükle
	ADD	L	
	LD	L,A	Basamak sayısını artır
	INC	C	Basamak sayısı = 4 mü?
	LD	A,C	
	CP	04	
	JP	Z,SON	
	SLA	L	Evet : son (işlem dizisi)
	RL	H	Hayır : HL'nin birleştirilmiş değerini 4 basamak sola kaydır
	SLA	L	
	RL	H	
	SLA	L	
	RL	H	
	SLA	L	
	RL	H	
	JP	TEKRAR	Bir sonraki basamağı okumak için döngüle
SON :	HALT		Son
GIRIS :	-		Tuştakımından bir basamak okuyan altyordam
	.		
	.		
	RET		

Şekil 6.12 Program örneği 6.8

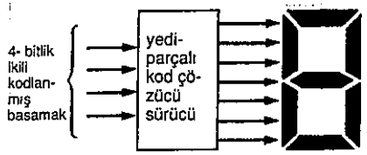
Program iki kaydırma komutu kullanır (SLA L ve RL H). Bunlar H,L kaydedici çiftinin içeriğini birlikte sola kaydırma işini yapan Z80'e özel komutlardır. İlk komut (SLA L); L kaydedicisinin içeriğini, en büyük değerlikli biti elde bitine (CY) kaydırıp, en küçük değerlikli bitini sıfırlayarak bir konum sola kaydırır. Ardından, ikinci komut (RL H) ise, H'nin içeriğini, en küçük değerlikli biti, elde bitinin yürürlükteki değerine eşit olacak biçimde bir basamak sola kaydırır. Böylece bu komut çiftini dört kere tekrarladıktan sonra birleştirilmiş H,L içeriği dört basamak sola kaydırılmış ve L'nin en küçük değerlikli dört bitini dört tane sıfır biti işgal etmiş olacaktır.

Ondalık basamakların görüntülenmesi

Daha önce de açıklandığı gibi, bir LED grubu kullanmak, çıkış portu kullanan mikrobilgisayar tarafından çıkışı yapılan ikili bir değeri görüntülemenin en basit yoludur. Ayrıca, eğer her bir LED, belli bir anlam taşırsa (örneğin belli bir denetim valfinin açık veya kapalı olduğu gibi), denetimli bir işlemde birleşik bir grup sayısal birimin durumunu görüntülemek için kullanışlı bir yöntem sağlar.

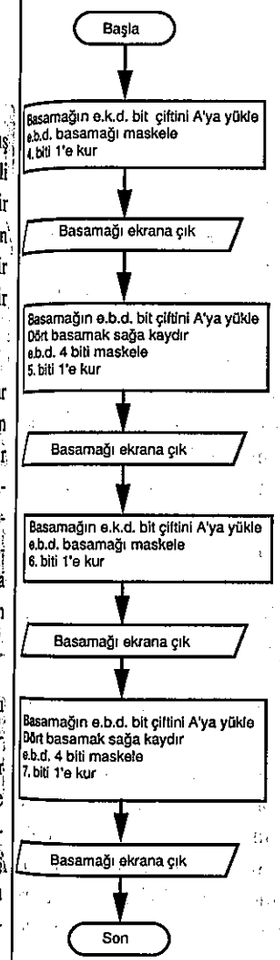
Öte yandan birçok uygulamada, örneğin, bir mikrobilgisayar denetimli tartı aletinde ölçülen ağırlığı veya denetimli işlemin sıcaklığını bir kontrol panelinde görüntülemek için ondalık bir ekran gerekir. Doğrusu her bir basamağı ikili kodlanmış biçimde görüntülemek için dört ayrı LED kullanmak uygun değildir; dolayısıyla da normalde, mikrobilgisayar tarafından üretilen ikili kodlanmış çıkışlarla ondalık ekran arasında dönüştürme işlemini yerine getirmek üzere özel bir arabirim devresi kullanılır.

Ondalık basamakları görüntülemek için kullanılan ekran, çoğu zaman LED veya sıvı kristalli (LCD) parçalar kullanılarak yapılmış yedi-parçalı göstergelerdir. On nümerik basamağın her birinin yedi parçalı bir göstergede görüntülenmesi Şekil 6.13'te verilmiştir. Bu birimlerde kullanılan arabirim devresi, mikrobilgisayar tarafından çıkışı yapılan 4-bitlik ikili-kodlanmış değeri giriş olarak alır ve görüntüleme biriminin yedi parçasını etkinleştirmek (açmak ve kapamak) için gereken sürme sinyallerini üretir.

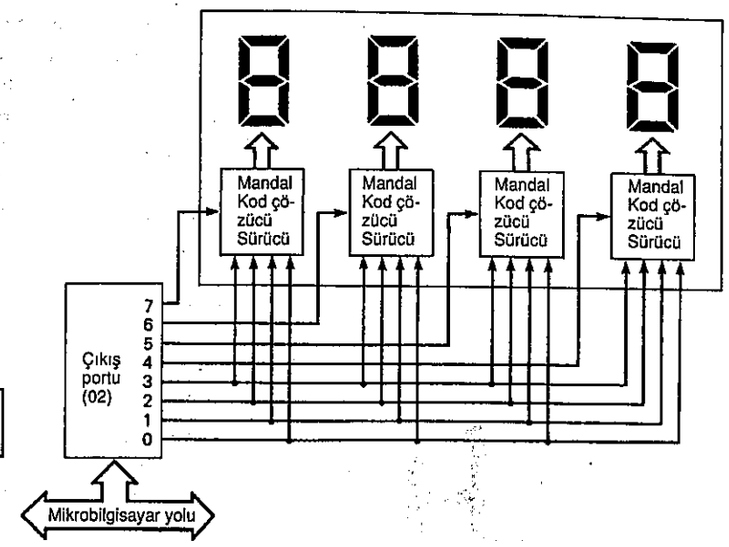


Ondalık basamak	Yedi-parçalı gösterme
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Şekil 6.13 Ondalık bir basamağın yedi-parçalı gösterimi



Şekil 6.15 Program örneği 6.9



Şekil 6.14 4 basamaklı kod çözücü ve görüntüleme birimi

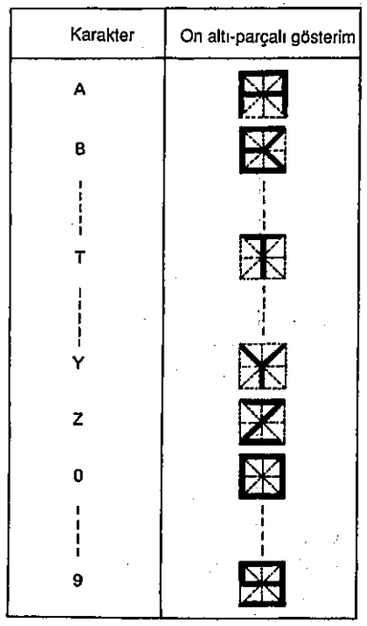
Çoğu zaman bir grup basamağı görüntülemek gerektiğinden, tek bir çıkış portundan dört basamağa kadar görüntüleme yapabilen arabirim devreleri mevcuttur. Şekil 6.14'de, böyle bir devre şematik olarak gösterilmiştir. Bu devreden de görülebileceği gibi, çıkış portunun en küçük değerlikli dört hattı bütün kilit/kod çözücü/sürücü devrelerine bağlanmıştır ve belli bir basamak konumu, en büyük değerlikli dört hat kullanılarak seçilir. Dolayısıyla, çıkış portundan gelecek 00010111 ikili çıkış değeri, en küçük değerlikli basamak sütununda ondalık yedi sayısının görüntülenmesiyle sonuçlanacaktır. Benzer şekilde, 10000110 ikili değeri en büyük değerlikli ondalık konumunda, ondalık 6'nın görüntülenmesi sonucunu verir.

Program Örneği 6.9 : Sayısal basamakların ondalık görüntülenmesi

Şekil 6.14'te verilen devreyi kullanarak H,L kaydedici çiftinde tutulan ikili kodlanmış basamakların nasıl görüntülendikleri, Şekil 6.15'teki akış diyagramı ve ilgili program kodunda gösterilmektedir. Önce istenmeyen basamaklar uygun kaydedicilerden (H veya L) maskelenirler ve ardından, belirli bir

Sembolik Komutlar	İşlem
LD A,L AND OF OR 10 OUT (02),A	L'den ilk değeri al Ekrana çık
LD A,L RRA RRA RRA RRA	L'den ikinci değeri al
AND OF OR 20 OUT (02),A	Ekrana çık
LD A,H AND OF OR 40 OUT (02),A	H'den ilk değeri al Ekrana çık
LD A,H RRA RRA RRA RRA	H'den ikinci değeri al
AND OF OR 80 OUT (02),A	Ekrana çık

porta çıkışı yapılmadan önce uygun bir seçim basamağı eklenir.



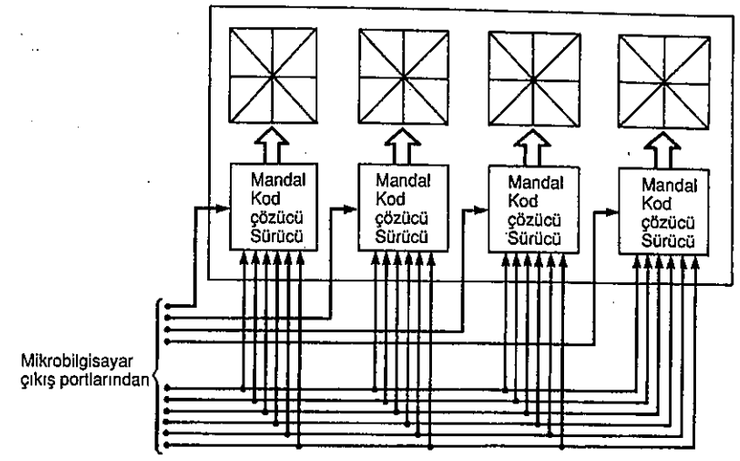
Şekil 6.16 On altı parçalı karakter ekranı

Alfasayısal basamakların görüntülenmesi

Açıkça görüldüğü gibi, yedi-parçalı bir göstergede (display), ek basamak sayıları görüntülenemez. Tabii ki, altı alfabetik karakteri (A, B, C, D, E ve F) bu gösterge ile görüntülemek mümkündür; dolayısıyla, yedi-parçalı bir gösterge, on altı adet onaltılı basamağı görüntülemede kullanılabilir. Bu, özellikle tek sistemkartlı bilgisayarlar gibi küçük bilgisayarların çeşitli bellek konumlarının veya mikroişlemci kaydedici içeriklerinin görüntülenmesinde kullanışlı olmaktadır.

Öte yandan alfabetik ve sayısal (alfasayısal) basamak veya karakterlerin tüm aralığının görüntülenmesini gerektiren uygulamalar için daha genişletilmiş bir ekran kullanmak gerekir. Bu amaçla kullanılan bir ekrana örnek, 16-parçalı görüntüleme birimidir. Bu görüntüleme biriminin çalışma ilkesi, daha önce ele alınan yedi-parçalı göstergeyle aynıdır, ancak bölme sayısının artmış olmasıyla daha geniş bir karakter aralığı görüntülenebilmektedir. Bu durum Şekil 6.16'da gösterilmiştir.

Bu birimlerle, daha geniş aralıkta karakter görüntülenebildiğinden, mevcut örnek birimler, görüntülenecek belli bir karakter ya da basamağı belirtmek için altı adet ikili giriş biti kullanır. Bu nedenle, giriş bitlerinin, tüm alfabetik (A-Z) ve sayısal (0-9) karakter aralığı ile bazı ek karakterleri (?,,, vb.) belirtebilecek yeterli sayıda (64) birleşimi vardır.



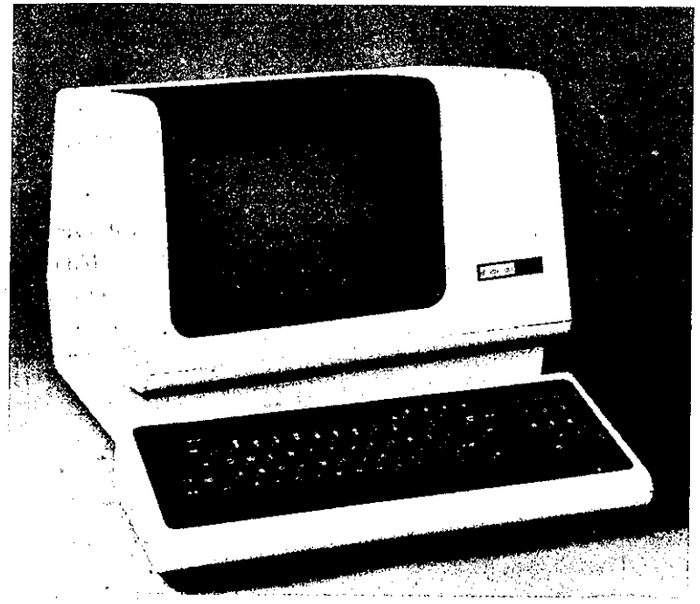
Şekil 6.17 4 karakterlik gösterge (display)

Tekrarlarsak, günümüzde mevcut arabirim cihazları normalde birden çok karakteri görüntüleyebilme yeteneğine sahiptir ve bir örnek olarak dört karakterlik görüntüleme birimi içeren bir gösterge, Şekil 6.17'de gösterilmiştir. Birim, birleşik mandallama, kod çözme ve parça sürme işlevleri sağlamaktadır; bu nedenle bir karakter, altı-bitlik ikili kodun çıkışının yapılması ve uygun mandal (basamak seçim) giriş hattının yetkilenmesiyle (mantıksal 1'e kurulması) suretiyle, belirli bir basamak konumunda görüntülenir.

6.5 Verilerin seri giriş ve çıkışı

Şimdiye dek tanımlanan giriş ve çıkış birimleri grubu, bir mikrobilgisayar uygulamasında kullanılan tipik örneklerdir. Öte yandan, mikroişlemcinin gelişkinlik ve işlem yeteneği arttıkça, mikrobilgisayarların uygulama alanları da genişler. Örneğin, günümüzde mikroişlemcinin, çok güçlü kişisel bilgisayar sistemlerinin yapımında kullanıldığını görmek artık oldukça sıradan bir durumdur. Mikroişlemcilerin işlem kapasiteleri arttıkça, bu sistemlerin gücü de artmaya devam edecektir.

Bu tip uygulamalarda en yaygın biçimde kullanılan giriş ve çıkış birimleri ya bir yazıcı ya da görsel görüntüleme birimidir (VDU). Bu birimlerin ikisi de, hem karakter ve hem de sayısal değer dizilerinin bir klavye aracılığıyla mikrobilgisayara girilmesi için elverişli ortamı ve ayrıca, bir yazıcıdan çıkan basılı bir form veya alfasayısal bir VDU ekranından mikrobilgisayar tarafından çıkışı yapılan sonuçları kaydedip inceleme olanağı sağladıklarından, bu tür uygulamalar için son derece uygundurlar.



Şekil 6.18 VT100 görsel görüntüleme birimi (VDU)

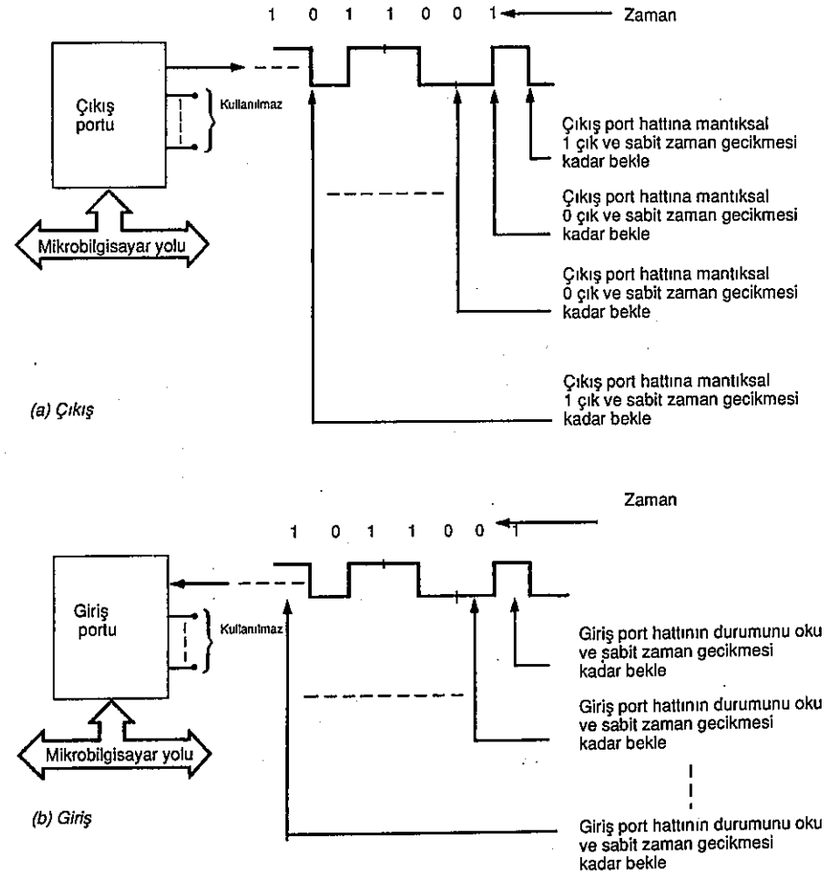
Bir örnek olarak, piyasada bulunan tipik bir VDU, Şekil 6.18'de gösterilmiştir. Görüleceği gibi, bu birim (hem alfabetik hem de sayısal karakter aralığı ile bir grup denetim karakterini girmek için kullanılan) bir klavye ve (hem klavyeden girilen karakterleri hem de mikrobilgisayar tarafından üretilen sonuçları görüntülemek için kullanılan) televizyon tipi bir görüntüleme ekranından oluşmuştur. Normalde klavyeden girilen denetim karakterleri, tüm alfabetik ve sayısal bilgi aralığının görüntü ekranına girilmesinde (yeni satır, boşluk vs.) ya da mikrobilgisayarda çalıştırılmakta olan yürürlükteki programa belli bir komutu belirtmek için (araya karakter sok, karakter dizisi sonu, vb. gibi) kullanılır.

Bu cihazların ikisinin de daha önce ele alınanlardan önemli farkları vardır. Çünkü, gerek teletayp veya VDU'dan gelen giriş verileri, gerek bu cihazlara çıkan veriler paralel olmayıp seri biçimdedir. Bu şu demektir: Klavyeden girilen bir basamağa karşılık gelen ikili kodlanmış değerin mikrobilgisayara bir *grup* hat üzerinde ikili bitler topluluğu olarak sunulmasının yerine bu değer, tek bir hat üzerinde *zaman serisi* biçiminde sunulur. Benzer şekilde, bu cihazlar tarafından görüntülenecek veri de mikrobilgisayar tarafından tek bir hat üzerinde bir zaman serisi biçiminde çıkarılmalıdır.

İkili kodlanmış bir değerin çıkışı, mikrobilgisayar tarafından çıkış portunun tek bir

hattını kullanarak ve bitler arasında sabit bir zaman gecikmesi ile her seferinde bir bit çıkışı yapan mikroişlemci programı kullanarak bir zaman serisi biçiminde yapılır. Eğer bu zaman aralığı (bit iletim hızı) alıcı cihaz tarafından biliniyorsa, veri eşzamanlı olarak okunur ve yorumlanır. Bu durum, diyagramsal olarak Şekil 6.19 (a)'da gösterilmiştir.

Benzer biçimde, mikroişlemciye tek bir giriş port hattı üzerinden bir zaman serisi biçiminde gönderilen ikili kodlanmış bir değer de, mikrobilgisayar tarafından giriş hattının durumunun, gönderilmekte olan veri ile zamansal uyumluluk içinde okunmasıyla yorumlanıp yeniden derlenebilir. Burada da mikroişlemci programı giriş hattında gösterilen her bir yeni veri biti arasındaki bilinen zaman gecikmesini he-
S 6.7 saplamalıdır. Bu durum Şekil 6.19(a) 'da diyagramsal olarak gösterilmiştir.



Şekil 6.19 Program denetiminde seri G/Ç

Karakter formatı

Bir teletayp veya VDU ile birleşik bir klavyeden girilen karakterleri belirlemede kullanılan bit sayısı normalde yedidir ve aslında, bu verinin formatını tanımlayan uluslararası onaylanmış standartlar bulunmaktadır. Bu standartlar referans olarak kullanılmak üzere EK-2'de verilmiştir ve ISO (Uluslararası Standartlar Organizasyonu) 7-bit veri değişim kodu veya alternatif olarak ASCII (Bilgi Değişimi için Amerikan Standart Kodu) adıyla bilinir.

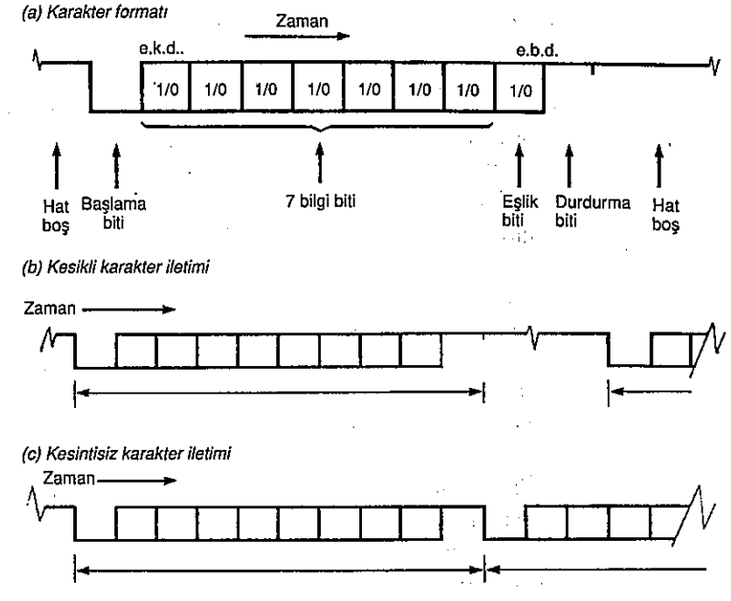
Mikrobilgisayarlarla, fiziksel olarak farklı mekanlarda bulunan teletayp veya VDU birimlerinin birbirine bağlanması oldukça yaygındır. Aslında az önce özetlenen *seri iletişim* yaklaşımının kullanılması, özellikle bu olasılık göz önünde tutularak benimsenmiştir. Böylece terminal laboratuvarın bir bölümüne, mikrobilgisayar da başka bir bölümüne veya ek bazı donanım ve sürme devrelerinin yardımıyla farklı bir binaya yerleştirilebilir.

İki cihazın fiziksel olarak birbirinden ayrı yerlerde bulunması, bu ikisi arasındaki veri aktarımının *iletim hataları* olarak adlandırılan etkilere (elektriksel girişim yoluyla) açık olacağı anlamına gelmektedir. Yani; bir karakterde yer alan herhangi bir bitin bozulması, böylece mantıksal 1'den mantıksal 0'a ya da tersine değişmesi olasılığı söz konusudur. Açıkçası bunun etkisi, alıcının aldığı veriyi hatalı bir şekilde yorumlaması olacaktır.

Bu yüzden, bu olasılığa karşılık, alıcı cihazın bir karakterin iletimi sırasında herhangi bir hata oluşup oluşmadığını algılayabilmesi için, iletilen her bir 7-bitlik karaktere, bir ek (sekizinci) bit eklenmektedir. Bu sekizinci bit, *eşlik biti* olarak adlandırılır ve aşağıda tanımlanan biçimde üretilip yorumlanır.

Terminal (veya mikrobilgisayar) tarafından üretilen her 7-bitlik karakter, iletilmeden önce ek bir sekizinci bit içerir. Sonuçta eşlik bitinin kendisinin de katılmasıyla elde edilen 8-bit uzunluktaki zarfta bulunan 1'lerin toplamı ya çifttir (*çift eşlik* denir) ya da tektir (*tek eşlik*). Böylece, örneğin, iletilecek karakter 1000001 (ASCII A karakteri) ise ve çift eşlik kullanılıyorsa, sekizinci bit 0 olacak ve böylece gönderilen 8-bitlik karakter 01000001 biçimini alacaktır. Benzer şekilde, eğer karakter 0110111 (ASCII 7 karakteri) ise, sekizinci bit 1 olacak ve gönderilen karakter 10110111 haline gelecektir.

Her bir karakterin alınmasında, mikrobilgisayar (veya terminal), bir iletim hatasının olup olmadığını belirler. Bunu, (eğer çift eşlik seçilmişse), alınan toplam bit sayısının çift olup olmadığının kontrolünü yaparak gerçekleştirir. Eğer herhangi bir hata ortaya çıkmışsa, mikrobilgisayar bu hatayı algılar ve yeniden gönderilmesi gereken hatalı karakterin düzeltilmesine ilişkin işlemleri yapar.



Şekil 6.20 Asenkron karakter formatı ve iletimi

Asenkron İletim

Terminal tarafından (ve tersi yönünde mikrobilgisayar tarafından) iletilen her karakter, sabit sayıda (8) bitten oluşur. Böylece karakterin başlangıcı ile bit hızı bilindikten sonra, gelen hattan alınan verileri senkronize etmek ve bütün karakteri tekrar birleştirmek oldukça kolay olacaktır. Ayrıca, eşlik biti aracılığıyla da herhangi bir iletim hatası belirlenebilecektir.

Öte yandan buradaki önemli varsayım da her karakterin başlangıcının bilinmesi konusudur. Oysa, bu durum her zaman geçerli değildir; çünkü bir karakter dizisinin mikrobilgisayardan çıkışı devamlı ve bilinen bir hızda olacak biçimde düzenlenebilse de, sözgelimi klavyeden bir karakter dizisi giren bir kullanıcı sürekli olarak giriş yapmayabilecektir. Bu nedenle, bu iletim modu, bir karakteri oluşturan her bir bit sabit bir hızda iletilse de, karakterler arası süre, dolayısıyla da karakterlerin iletim hızı değişken olacağından, *asenkron iletim* olarak adlandırılır.

İletilen karakterler, tümüyle birlerden veya tümüyle sıfırlardan meydana gelebildiği için; alıcının, iletilmekte olan her yeni karakterin başlangıcını algılayabileceği bir mekanizmaya sahip olması gerekir. Bu, 8-bitlik karakter zarfından önce, *başlatma biti* olarak bilinen bir bit eklenerek yapılır. Ayrıca, iletim hattının, ardışık karakterler arasında, sürekli olarak boştaki duruma (hat boş durumuna) dönmesini sağlamak için, her 8-bitlik karakter bir *durdurma biti* ile sonlandırılır. Böylece, tipik bir iletim hattının, bir veya daha çok karakter iletme durumu Şekil 6.20'de gösterilmiştir.

Yukarıdaki açıklamadan çıkarılabileceği gibi, bir hattaki gerçek veri iletim miktarı, gerçekleşen bit iletimi sayısından daha azdır. Örneğin, yukarıdaki karakter formatı ele alındığında, her bir 7-bitlik karakter için en az 10 bit iletilmesi gerekecektir. Başlatma ve durdurma bitleri dahil olmak üzere bir saniyedeki bit iletim sayısı, *baud hızı* olarak bilinir ve asenkron bir hattan elde edilen gerçek veri hızından daha büyük değere sahiptir. Kullanılmakta olan tipik standart baud hızları, görsel ekran birimleri için 1200, 2400, 4800 ve 9600'dür. Şu teletayp yazıcıda kullanılan standart ise 110 baud'tur. Şu noktaya dikkat edilmelidir: bu sonuncusunda bir değil, iki durdurma biti kullanılmaktadır; dolayısıyla 110 baud, saniyede 10 gerçek bilgi karakterine eşdeğerdedir.

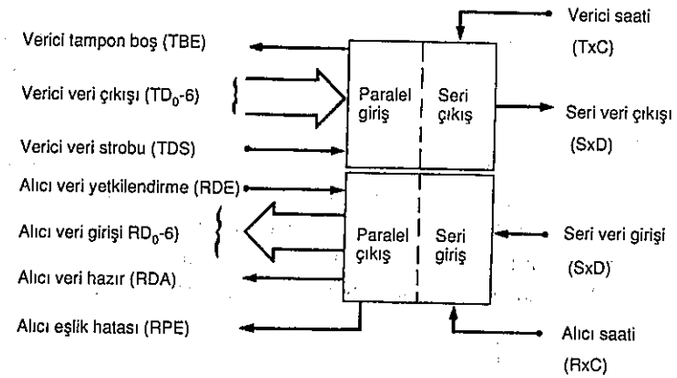
UART

Seri verinin giriş çıkışını bir programın denetimi altında yapmak mümkün olsa da, daha yaygın bir yaklaşım, mikrobilgisayar G/Ç portlarında kullanılan paralel ikili biçimi iletim hatlarında kullanılan bit dizisine dönüştürmek için *UART* (Genel Amaçlı Asenkron Alıcı Verici) olarak bilinen özel bir arabirim devresi kullanmaktır.

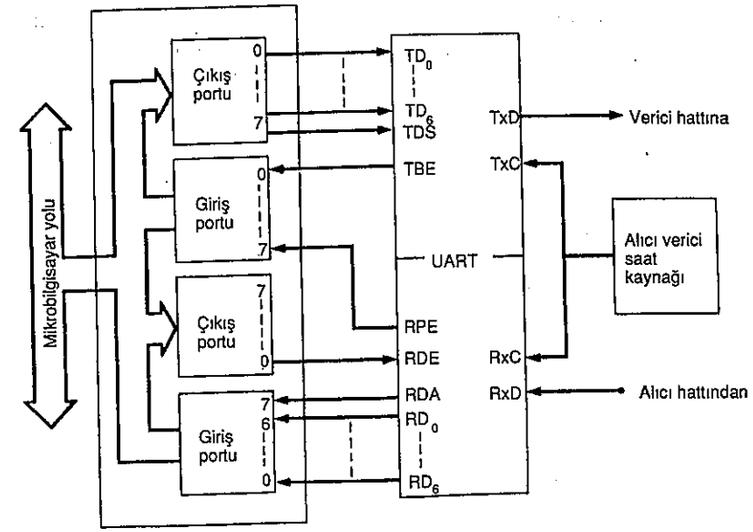
Bir UART'ın şematik diyagramı Şekil 6.21'de gösterilmiştir. UART, biri çıkan seri veri dizisinin iletimini denetlemek için (verici), diğeri de gelen seri veri dizisinin kabulünü denetlemek için (alıcı) olmak üzere iki bileşenden oluşur.

Verici bölümü aslında, mikrobilgisayardan çıkan paralel ikili biçim ile iletişim hattından iletilen bit dizisi biçimini paralelden-seriye dönüştürür. Dolayısıyla da, içerdiği değerin paralel olarak yüklendiği ve ardışık bit dizisi biçiminde çıkarıldığı bir kaydediciden oluşmuştur. Bu kaydedici, *verici tampon* kaydedicisi olarak bilindiği gibi, paralel-giriş, seri-çıkış kaydedicisi olarak da bilinir.

Bir karakter verici tampon kaydedicisine yüklendiğinde, otomatik olarak vericinin



Şekil 6.21 Bir UART'ın şematik diyagramı



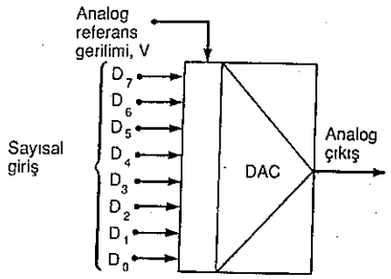
Şekil 6.22 G/Ç portları aracılığıyla bir UART'ın bağlanması

saati ile eşzamanlı olarak kaydırılır (saatle denetlenir) (TxC). Bu saat, dışardaki bir üreteçten üretilmeli ve istenen baud hızına eşitlenmelidir. Bundan başka, verici, devre içinde yüklenen karakter için istenen eşlik bitini üretir ve iletim sırasında başlatma ve durdurma bitlerini ekler. Son olarak, durdurma biti iletildikten sonra verici, mikrobilgisayara verici tamponun şimdi boş olduğunu (TBE) ve bir sonraki karakterin yüklenebileceğini sinyalle bildirir. Bu işlem, karakterin yedi bitini paralel girişlere vererek ve iletilen veri strob (TDS) girişini açıp kapayarak (0 ➔ 1 ➔ 0) yapılır.

Alıcı bölümü, az önce tanımlanan verici kesiminin tersi doğrultuda işlev görür ve iletim hattından gelen seri girişleri, mikrobilgisayarın gerektirdiği paralel çıkışlara dönüştüren bir seri-giriş paralel-çıkış kaydedicisinden oluşur. Ayrıca, her yeni karakterin başlangıç ve bitişini algılamak için gerekli bir denetim devresi de içerir. Seri veriler, alıcı saati (RxC) ile bit uyumluluğu içinde alıcı kaydedicisine saat denetimi altında girilir ve dolayısıyla bu teletayp veya VDU terminalinde kullanılan ilgili verici saatine eşitlenmelidir.

Her bir karakterin alınması süresinde alıcı, gelen bit dizisinin eşlik değerini yeniden hesaplar ve durdurma biti alındıktan sonra alınan karakterdekiyle aynı olup olmadığını belirler. Eğer bu iki eşlik değeri aynıysa, alıcı eşlik hatası (RPE) hattını sıfırlar (0); fakat farklıysa RPE hattı 1'e kurulur. Böylece, mikrobilgisayar alınan karakteri okuduğunda, bir iletim hatası algılanmışsa bunu kolayca belirleyecektir.

Bundan başka, durdurma biti alındığında alıcı, mikrobilgisayara bir karakterin alındığını ve şimdi okunabileceğini belirtmek için alma verisi hazır (RDA) hattını man-



Giriş bit numarası	Analog ağırlık
D_0	V/256
D_1	V/128
D_2	V/64
D_3	V/32
D_4	V/16
D_5	V/8
D_6	V/4
D_7	V/2

Sayısal giriş								Analog çıkış gerilimi
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	V/256
0	0	0	0	0	0	1	0	V/128
0	0	0	0	0	0	1	1	3V/256
0	0	0	0	0	1	0	0	V/64
0	0	0	0	0	1	0	1	5V/256
1	1	1	1	1	1	0	0	63V/64
1	1	1	1	1	1	0	1	253V/256
1	1	1	1	1	1	1	0	127V/128
1	1	1	1	1	1	1	1	255V/256

Şekil 6.23 Sayısal-analog dönüştürücü (DAC) ilkeleri

tıksal 1'e kurar. Böylece mikrobilgisayar alma verisi yetkilendirme (RDE) hattını mantıksal 1'e kurar ve o an giriş portunda hazır olan karakteri okur.

UART'lar, mikrobilgisayar sistemlerinde yaygın olarak kullanıldıklarından, bir mikrobilgisayar yoluna doğrudan bağlanabilecek biçimde tasarlanmışlardır. Dolayısıyla, mikroişlemci açısından bakıldığında bir grup giriş-çıkış portu olarak görülürler. Öte yandan bunları, bir grup standart giriş-çıkış portuna bir arabirim devresi olarak dışardan bağlamak da mümkündür ve örnek bir düzenleme Şekil 6.22'de görülmektedir. Yukarıda özetlenen çeşitli denetim sinyalleri, bir mikroişlemci programı ve daha önce ele alınan temel IN ve OUT (giriş-çıkış) komutlarının denetiminde izlenip etkinleştirilebilir.

6.6 Analog giriş ve çıkış

Kısım 6.3'te belirtildiği gibi, bir mikrobilgisayar endüstriyel bir işlemi denetlemek için kullanılırken, birçok giriş ve bunlara bağlı çıkış sinyalleri sayısal niteliktedir; dolayısıyla, uygun arabirim devrelerinin yardımıyla doğrudan G/Ç arabirim ünitesine girilebilir veya ondan çıkış yapılabilir.

Öte yandan, sayısal sinyallere ek olarak, birçok endüstriyel işlem, analog nitelikte sinyaller de üretir ve aynı zamanda bunlara bağlı bazı denetim çıkışlarının da analog biçimde olması istenir. Örneğin, birçok dönüştürücü (sıcaklık, basınç vb.) istenen parametreyle orantılı olarak değişen bir (analog) gerilim üretir. Benzer biçimde, bir grup denetim cihazı; büyüklüğü, gerçekleştirilen denetleme işleminin derecesini belirleyen bir giriş gerilim sinyaline gereksinim duyar.

Bu yüzden bu tip cihazlarda; mikrobilgisayar giriş ve çıkış portlarında kullanılan sayısal giriş ve çıkışla denetimli işlemde kullanılan analog sinyaller arasında bir dönüştürme işlemi yapmak gerekir. Bu amaçla yapılan iki tür arabirim devresi vardır: Biri analog sinyalden sayısal sinyale dönüştürmeyi, diğeri de sayısal sinyalden analog sinyale dönüştürmeyi gerçekleştirir. Her ikisi de aşağıda ele alınacaktır.

Sayısaldan-analoğa dönüştürme

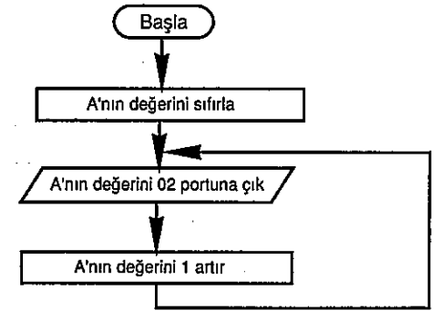
Bir mikrobilgisayar portundan çıkan sayısal bir değeri analog biçimdeki eşdeğerine dönüştürmek için kullanılan arabirim devresi, sayısal/analog dönüştürücü veya DAC olarak bilinir. Esasen, DAC'a giren sayısal girişler, ikili ağırlıklama biçiminde düzenlenen belli bir grup ayrık eleman gerilimlerini seçmek için kullanılır. Daha sonra, seçilen gerilim değerleri, buna eşdeğerde analog gerilim değerlerini oluşturmak üzere birleştirilir. Böylece 8-bitlik bir dönüştürücü ile her bir gerilimin ağırlıklaması, Şekil 6.23'te gösterilmiştir. Şekil aynı zamanda, farklı sayısal girişler için çıkan analog çıkış gerilimini de gösterir.

DAC'a giren sayısal girişler mikrobilgisayarın bir çıkış portundan kolayca alınabilirler. Böylece mikrobilgisayar, belirtilmiş büyüklükteki bir analog gerilim çıkışını,istenen gerilimin sayısal eşdeğerini bir çıkış portuna çıkararak başlatabilir.

Program örneği 6.10 : Sayısal/Analog Dönüştürücü

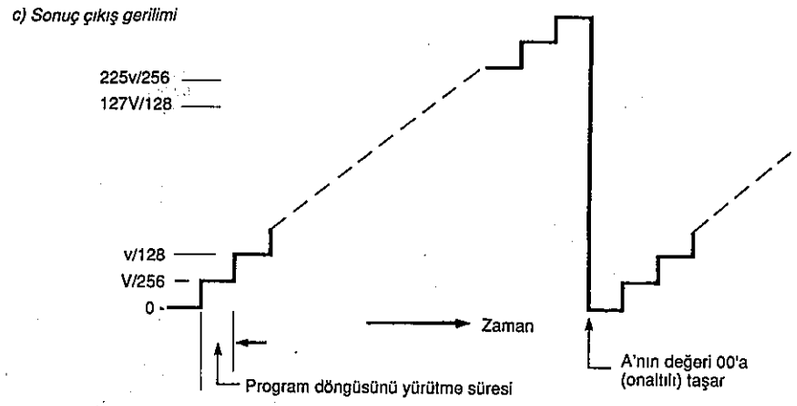
Şekil 6.24'teki program örneği; bir DAC'a, artan bir sayısal değer çıkılmasının etkisini göstermektedir. DAC'ın mikrobilgisayarın 02 çıkış portuna bağlandığı var-

a) Akış diyagramı



b) Program kodu

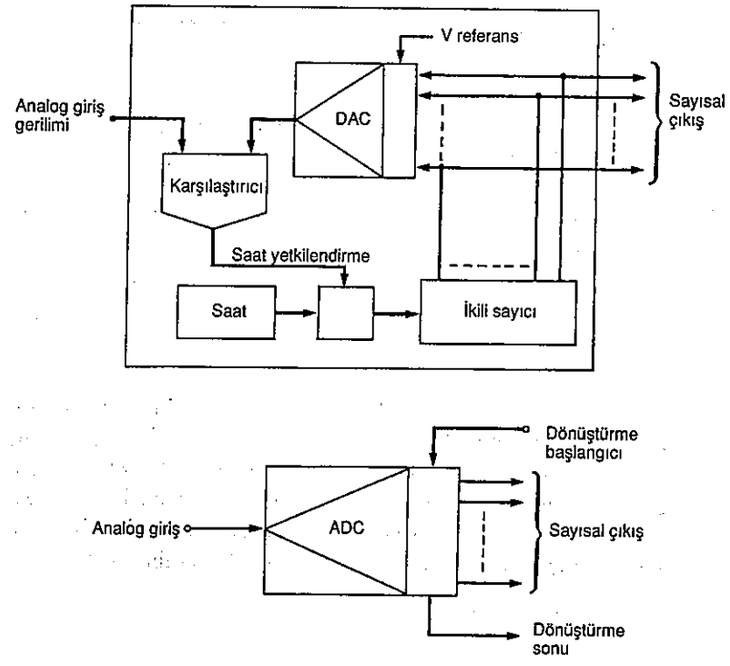
Sembolik komutlar		İşlem
	LD A,00	A'nın değerini sıfırla
TEKRAR:	OUT(02),A	A'nın değerini 02 portuna çık
	INC A	A'nın değerini 1 artır
	JP TEKRAR	Tekrarla

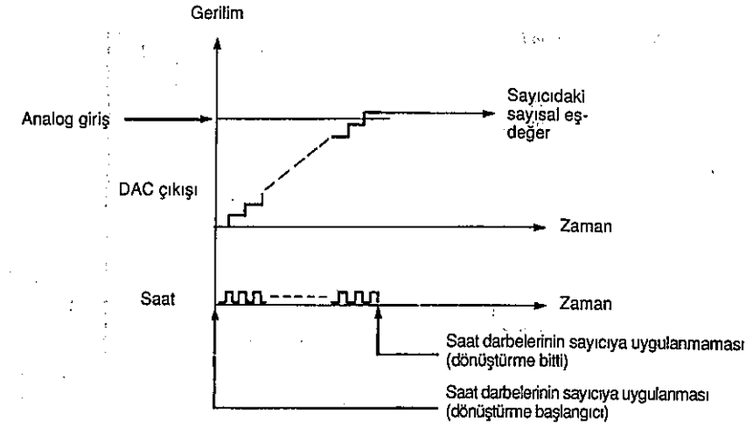


Şekil 6.24 Program Örneği 6.10

sayılmıştır. Çıkan sayısal değer döngü içinde bir artırılır ve sonuçta FF (onaltılı) değerine ulaştığında, kendisini 00'a (onaltılı) sıfırlayıp döngüyü tekrar eder. Böylece çıkan analog gerilim, daha önce Şekil 6.23'te gösterildiği gibidir ve şekilde, zamanla değişimi gösterilmiştir. DAC'a her bir yeni değer çıkıldığında, sonuçtaki analog gerilimde de bir artış adımı gözlenecektir.

S 6.8





Şekil 6.25 Analog-sayısal dönüştürücü ilkeleri

Analogdan-sayısala dönüştürme

Analog bir gerilimi, bir mikrobilgisayara giriş için uygun sayısal biçimdeki eşdeğerine dönüştürmede kullanılan arabirim devresi Analog/Sayısal Dönüştürücü veya ADC olarak bilinir.

Bir takım alternatif ADC devreleri mevcutsa da, bu devrelerin temel işlem modu aynıdır. Bir ADC, aslında, kendi içinde bir DAC içerir. Devre, otomatik olarak program örneği 6.10'dakine benzer bir artırma işlemini gerçekleştirir. Bundan başka devre içinde bir de *karşılaştırıcı* bulunur. Bu, adından da anlaşılacağı gibi, DAC tarafından üretilen analog gerilim ile gerçek analog gerilim girişini karşılaştırır.

Eğer analog gerilim girişi DAC'tan çıkan gerilimden büyükse, devre DAC'a giren sayısal değeri artırmaya devam eder. Öte yandan DAC'ın çıkış gerilimi analog girişi geçerse, karşılaştırıcı bunu algılar ve artırma işlemini durdurur. Böylece, DAC'a girilen o anki sayısal giriş, analog giriş geriliminin istenen sayısal eşdeğeridir. Bu durum, Şekil 6.25'te diyagram olarak gösterilmiştir.

Bu işlem modunun iki etkisi vardır : İlk olarak, her yeni dönüştürme için sayısal değeri siler; sıfırdan başlatması için ADC'ye özel bir "dönüşümü başlat" komutu vermek gerekir. Ve ikinci olarak da, her bir dönüştürme işlemini gerçekleştirmek için gereken zaman, analog giriş geriliminin büyüklüğüne bağlı olarak değişir - gerilim arttıkça, sayısal değeri, istenen düzeye kadar artırmak için gereken sayma saykıl sayısı da artar. Bu ikincisi, örneğin, analog bir gerilim sürekli olarak izlenirken, dolayısıyla da dönüştürülürken çok önemlidir.

Bu nedenle bu etkilerin her ikisinin de varolabilmesi için bir ADC'nin iki denetim hattı vardır : Yeni bir dönüştürme başlatan dönüştürme başlatma girişi ve dönüşümün tamamlandığını göstermek için ADC tarafından üretilen "dönüşüm ta-
S6.9 mamlandı" çıkış sinyali.

Sorular

6.1 Sekiz anahtar ve sekiz LED'lik bir grubun iki mikrobilgisayar portuna, Şekil 6.2'de görüldüğü gibi bağlandığını varsayarak:
a) sekiz giriş anahtarının da sürekli olarak durumunu okuyan ve sekiz LED üzerinde bu durumların tümleyenlerini (tersini) görüntüleyen;
b) anahtarların durumunu sürekli olarak okuyan ve seçilen son durumun tümleyeni, FF (onaltılı) değerine ulaştığında sona eren bir program yazın.

6.2 Program örneği 6.3'ü, sayma sayısı FF'e (onaltılı) kadar artırıldığında 00'a (onaltılı) sıfırlanacağına, geriye saymaya (azalmaya) başlayacak şekilde değiştirin. Ayrıca, 00'a (onaltılı) kadar azalttıktan sonra program, yeniden artırmaya başlamalıdır. Sonuçtaki dalga biçiminin taslağını çizin ve buna bir ad önerin.

6.3 Şekil 6.7'de görülen bir grup durumu bellekte bir tabloda saklayan ve her bir yeni durum arasında sabit bir zaman gecikmesiyle, bu sıranın ilgili çıkışını 8-LED'lik gruba yapan bir program yazın.

6.4 Şekil 6.8'de gösterilen değişken zaman gecikme parametrelerini ve buna bağlı denetimli cihaz durumlarını bellekte bir tabloda saklayan ve her bir yeni durum arası için ilgili zaman gecikmesiyle, belirtilen sıranın ilgili çıkışını bir grup LED'e yapan bir program yazın.

6.5 Şekil 6.9'da görülen denetimli giriş durumlarını ve bunlara karşılık gelen denetimli birim durumlarını bellekte bir tabloda saklayan ve buna bağlı olarak ilgili çıkış durumlarını 8 LED'lik bit gruba çıkmadan önce belirlenen giriş durumlarını sekiz tane anahtarlık bir gruptan okuyan bir program yazın.

6.6 Beş anahtar ve beş LED kullanarak, Şekil 6.10'da gösterilen bir tuştakımından ondalık veya onaltılı basamak girişini simüle eden bir program yazın. Beşinci anahtar ve LED, Şekil 6.10'da gösterilen onay işlemini belirtmek için kullanılmaktadır. Kalan dört anahtar, basamağın ikili kodlanmış biçimde girilmesi, dört LED de girilen basamağın görüntülenmesi için kullanılmalıdır.

6.7 Mikroişlemci kaydedicilerinden birinde seri biçimde saklı 8 bitlik bir değeri, bir

LED bağlantılı tek bir çıkış hattı kullanarak çıkaran bir program yazın. Bit çıkış hızı, saniyede bir bit olacak şekilde ayarlanmalıdır.

6.8 Sayısal/analog dönüştürücünün çalışma ilkesini, gereken yerlerde diyagramların da yardımıyla açıklayın. Mikrobilgisayarın, aşağıdaki sayısal değerleri sabit bir zaman sırası içinde çıkarması halinde, dönüştürücünün üreteceği çıkış gerilimini gösteren diyagramı çizin.

00000000
00000001
00000010
00000100
⋮
100000000

6.9 Diyagramların yardımıyla bir analog/sayısal dönüştürücünün çalışma ilkesini açıklayın. DAC'a sayısal girişi türetmek için dönüştürme işlemini hızlandıran basit ikili sayma işlemine bir alternatif önerin.

7. Bölüm : **Mikrobilgisayar Donanımı**

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde:*

1 *Tipik bir 8-bit mikroişlemcinin şematik diyagramını çizebilmeli ve diyagram üzerinde, hem adres ve veri yollarındaki bilgi akışını hem de bazı tipik komut saykıllarının getirme ve yürütüm fazları boyunca üretilen denetim sinyallerini gösterebilmeli,*
2 *Sayısal bir cihazın yükleme etkilerini ve bunun bir mikrobilgisayar yoluna bağlanabilecek cihaz sayısını nasıl belirlediğini değerlendirebilmeli,*
3 *Tekyönlü ve ikiyönlü üç-durumlu tamponların işlevini ve işlem modunu açıklayabilmeli,*
4 *Bir adres kod çözücünün işlevini ve nasıl gerçekleştirildiğini değerlendirebilmeli,*
5 *Tipik sadece-okunur bir bellek ve rastgele-erişimli bir belleğin şematik diyagramını çizebilmeli ve zamanlama diyagramları yardımıyla çalışmalarını açıklayabilmeli,*
6 *Tipik bir mikrobilgisayar sistemi için bir bellek haritası geliştirebilmeli ve farklı bellek cihazlarının mikrobilgisayar yoluna bağlantısını sağlamak için gereken ilgili adres kod çözücü devresini belirleyebilmeli,*
7 *Şematik diyagramların yardımıyla tamponlu bir giriş portu ve tamponlu bir çıkış portu içeren bir G/Ç arabirim ünitesinin çalışmasını ve bu üniteden mikrobilgisayar yoluna nasıl bağlantı yapıldığını açıklayabilmeli,*
8 *Programlanabilir bir giriş/çıkış (PIO) biriminin sağladığı olanakları ve bundan bir mikrobilgisayar yoluna nasıl arabirim sağlandığını anlamış olmalısınız.*

7.1 Giriş

2. Bölümde bir mikrobilgisayarı oluşturan üç ana birimin - mikroişlemci, bellek birimi ve G/Ç (I/o) arabirim ünitesi- temel işlevleri ve işlem modu tele alınmıştı. Bundan başka, mikrobilgisayar yolunun yapısı ve genel çalışması da açıklanmıştı.

Bu bölümde bu birimlerin her birinin ayrıntılı olarak çalışması ve gerçek donanım tasarımı ile bunları gerçekleştirmek için kullanılan ilgili denetim devreleri ele alınacaktır.

7.2 Mikroişlemci

Tipik bir 8-bit mikroişlemcinin şematik blok diyagramı Şekil 7.1'de gösterilmiştir. Bu şekil daha önce 2. Bölümde gösterilene benzemektedir; ama, burada daha sonraki bölümlerde tanıtılan bazı ek kaydediciler da dahil edilmiştir. Böylece aritmetik mantık biriminde (ALU) onunla ilgili bir bayrak kaydedicisi ve 16-bitlik bir yığın işaretçisi içeren bir kaydedici birimi yer almaktadır. Bundan başka, kaydedici birimine eklenmiş bir adres artırıcı/azaltıcı birim vardır. Bu birim, örneğin, her ardışık getirme fazından sonra program sayıcısını artırmak, ayrıca ardışık her yığın iş-

leminden sonra yığın işaretçisini artırmak veya azaltmak için kullanılır. Programın her bir komutunu yürütmek için mikroişlemci iki-fazlı veya getir-yürüt modunda çalışır. Getirme fazı boyunca komut o sırada program sayıcısında tutulan adres kullanılarak bellek biriminden okunur ve yürütme fazı sırasında da mikroişlemci komutun gerçekleştirilmesi için gerekli olan işleri yapar. Şimdi, mikroişlemci tarafından üretilen sinyalleri ile veri ve adres yolu üzerindeki bilgi akışını göstermek üzere, farklı komut türleri için tipik bazı getir-yürüt saykılları ele alınacaktır.

Getirme Fazı

Daha önceki bölümlerde de belirtildiği gibi, bir komutu tanımlamak için gereken bayt sayısı farklı komut türleri için değişiklik gösterir. Örneğin, iki tane dahili mikroişlemci kaydedicisini işin içine katan basit bir yükleme (taşıma) komutu tek bir bayta, dolaysız veri değeri içeren bir komut iki bayta ve bir bellek adresi içeren bir komut da üç bayta ihtiyaç duyar. Böylece, komutttaki bayt sayısına bağlı olarak mikroişlemcinin, komutu bellekten getirmesi için çeşitli sayıda temel *makine durumuna* veya *saat saykılına* gereksinimi vardır.

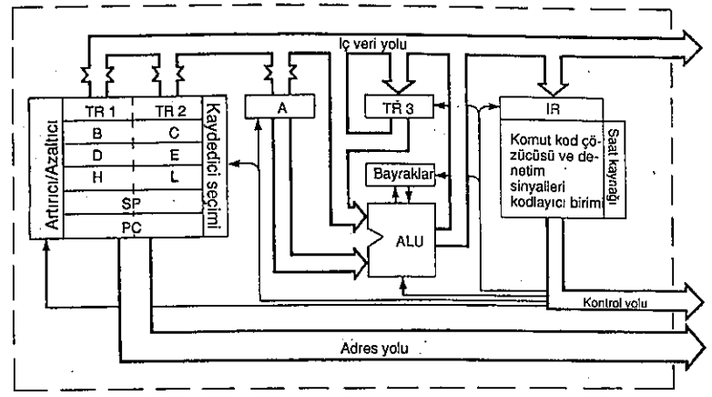
Bir komuttaki bayt sayısına bakılmaksızın, komutun ilk baytı daima komut kaydedicisine (IR) yüklenir. Sonra, eğer komut iki veya daha çok bayttan oluşuyorsa, ilgili baytlar yürütme fazı sırasında kullanılmadan önce bellekten getirilir ve bir veya daha çok geçici kaydediciye (Şekil 7.1'deki TR1, TR2 ve TR3) yüklenir.

Bir komutun ilk baytını getirmek için mikroişlemci tarafından üretilen sinyaller ile adres ve veri yollarındaki bilgi akışı sırasıyla Şekil 7.2'nin (a) ve (b) bölümlerinde gösterilmiştir. Önce Program sayıcısı (PC) seçilir ve içerdiği değer adres yoluna çıkarılmak üzere yetkilendirilir. Sonra bir bellekten okuma (MEMRD) sinyali üretilir ve adreslenmiş konumun içerdiği değer (komut baytı) veri yolundan komut kaydedicisine (IR) mandallanır. Son olarak, program sayıcısı, bellekteki bir sonraki ardışık baytı göstermesi için bir artırılır.

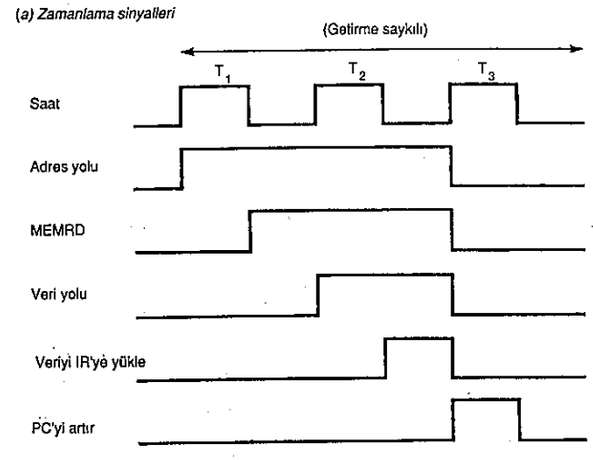
Daha sonra komut kaydedicisinin içeriğinin kodu çözülür ve okunan komut tek bayt bir komutsa yürütme fazına geçilir. Fakat ilk bayt komutun iki veya üç baytlık bir komut olduğunu belirtiyorsa, takip eden baytlar şimdi bellekten getirilmelidir. Örnek olarak, bir iki-bayt ve bir de üç-bayt komut için adres ve veri yollarındaki bilgi akışı, Şekil 7.3'ün sırasıyla (a) ve (b) bölümlerinde gösterilmiştir.

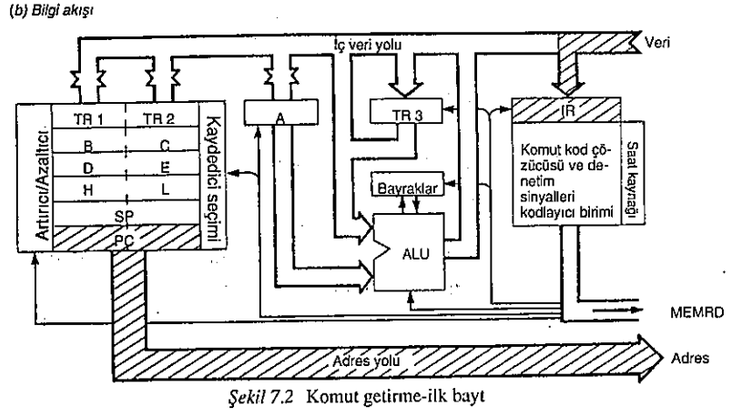
İlk örnekte, Şekil 7.3 (a), bellekten okunan ikinci baytın bir aritmetik komutuna bağlı dolaysız bir veri değeri olduğu (sözgelimi ADD A,7E) varsayılır. Böylece okunan bayt, örnekteki 7E (onaltılı), TR3 geçici kaydedicisine yüklenir.

İkinci örnekte, Şekil 7.3 (b), bellekten getirilen iki ek baytın, kaydedici birimindeki



Şekil 7.1 Tipik bir 8-bit mlikroişlemcinin şematik blok diyagramı





Şekil 7.2 Komut getirme-ilk bayt

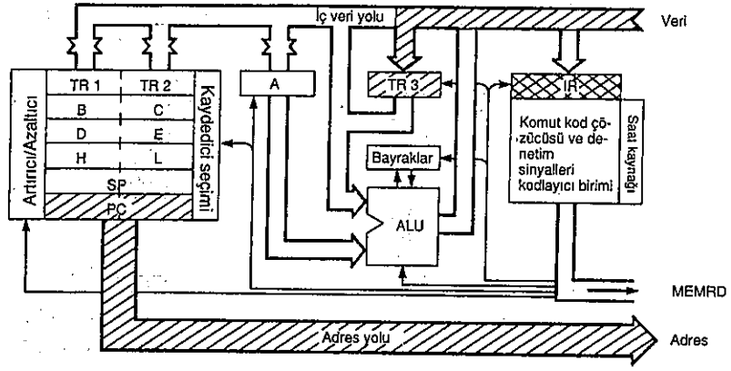
bir kaydedici çiftiyle ilgili iki direkt veri değeri olduğu varsayılır (sözgelimi LD BC,72A8). Dolayısıyla, ilk değer [örnekte 72 (onaltılı)] TR1 geçici kaydedicisine, ikinci değer de [örnekteki A8 (onaltılı)] TR2 geçici kaydedicisine yüklenir.

Bir komutun her bir baytı (her biri için farklı MEMRD sinyali kullanılarak), bellekten getirildikten sonra program sayıcısı daima bellekte yer alan bir sonraki ardışık baytı göstermek üzere bir artırılır. Son olarak, komutu oluşturmak için gereken bayt sayısı kadar bayt bellekten
S 7.1 getirilip uygun kaydedicilere yüklendikten sonra, denetim birimi yürütüm fazını başlatır.

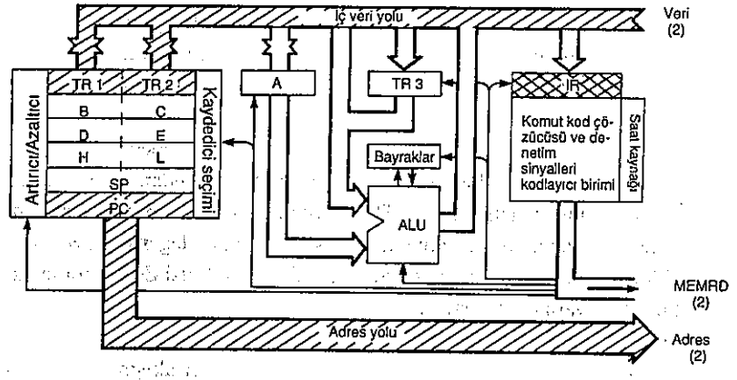
Yürütüm Fazı

Getirme fazında olduğu gibi burada da, farklı komut türlerini yürütmek için farklı sayıda saat saykılına gerek duyulur. Örneğin, bir dolaysız topla komutu için getirme

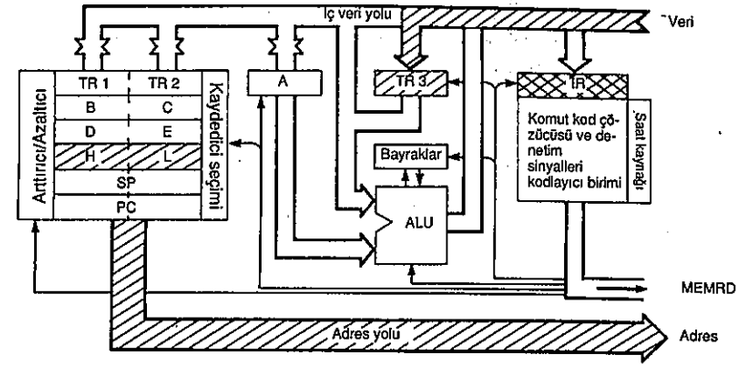
(a) İki-baytlık komut

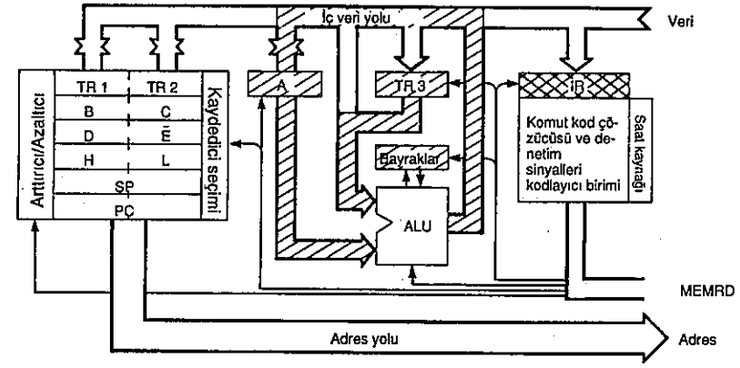


(b) Üç-baytlık komut



Şekil 7.3 Komut getirme- ek baytlar





Şekil 7.4 Bellekten topla komutunun yürütme fazı

fazı sırasında dolaysız veri değeri TR3 geçici kaydedicisine yüklendikten sonra, toplama işlemini gerçekleştirmek için tek bir ek saykıl gereklidir. Öte yandan bir bellekten topla komutu için, örneğin ADD A,(HL), iki saykıl gerekir : biri bellekten değeri getirmek için, diğeri de toplama işlemini yapmak için.

Bundan şu çıkarılabilir : Dolaysız topla komutu sadece dahili mikroişlemci kaydedicileriyle ilgili olduğu için mikroişlemci tarafından dış sinyallerin üretilmesine gerek yoktur. Akümülatör ve TR3 kaydedicisinin içerdiği değerler önce ALU'nun girişlerine uygulanır ve sonuçtaki mevcut toplam değerinin üzerine yazılmak üzere akümülatöre mandallanır. Ayrıca bayrak kaydedicisi de bu komut için tanımlanmış kurallar uyarınca değiştirilir.

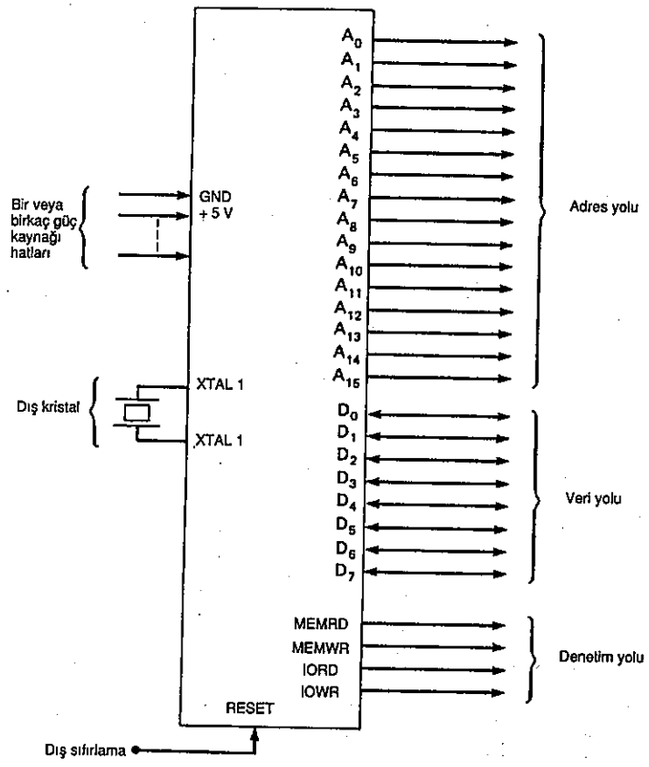
Bir yürütüm fazı boyunca, mikroişlemcideki bilgi akışını göstermek için bellekten

topla komutunun yürütme saykılının iki parçası, sırasıyla Şekil 7.4'ün (a) ve (b) bölümlerinde gösterilmiştir.

Şekil 7.4 (a)'da görüleceği gibi, saykılın ilk kısmı boyunca ikinci işlenen önce H,L kaydedici çiftinin o anki değeri tarafından belirtilen bellek adresinden getirilir. Sonra, saykılın ikinci kısmı sırasında A kaydedicisinin yürürlükteki değerine eklenir. Toplam, A kaydedicisinin eski değerinin üstüne yazılır ve bayrak kaydedicisi
S7.2 yapılan işlemlerden önceden tanımlı kurallar uyarınca etkilenir.

Başlatma İşlemi

Mikrobilgisayara enerji ilk verildiğinde, yani mikrobilgisayar açıldığında, mikroişlemci doğrudan çalışmaya başlar. Öte yandan, çeşitli kaydediciler -program sayıcısı da dahil- rastgele değerler içereceklerdir; bu nedenle böyle bırakılması durumunda, mikroişlemci ilk baytı bilinmeyen rastgele bir bellek konumundan getirecektir.



Şekil 7.5 8-bit bir mikroişlemcinin bazı giriş ve çıkış bacakları

Bu nedenle, mikrobilgisayara ilk enerji uygulandığında, mikroişlemcideki program sayıcısını bilinen bir duruma (çoğu kez sıfıra) getirmek veya sıfırlamak gerekir. Bu yüzden tüm mikroişlemciler, etkinleştirildiğinde bu işlevi gerçekleştiren bir dış RESET denetim sinyali kullanır. Dolayısıyla, eğer saklı programın ilk baytı bu konumda, yani program sayıcısında saklanacak biçimde düzenleme yapılırsa mikroişlemci, programı, bilinen ve denetimli bir yoldan derhal yürütmeye başlayacaktır. Bu yöntem çoğunlukla *açılışta-sıfırlama* olarak bilinir.

Şuna dikkat edilmelidir: RESET denetim sinyali normalde öteki mikroişlemci kaydedicilerinden herhangi birinin içerdiği değeri etkilemez ve bu nedenle programcı, saklı program ilk yürütülmeye başlandığında bunların içerdiği değerlerin rastgele veya bilinmeyen olduğunu varsaymalıdır. Bu yüzden, bir program yazarken normalde yapılan ilk iş ilgili mikroişlemci kaydedicilerini (örneğin yığın işaretçisini) veya bellek konumlarını, bilinen değerleri içerebilmeleri için temizlemek veya kullanıma hazır hale getirmek gerekir. Bu iş çoğunlukla *kullanıma hazırlama* olarak anılır.

Şekil 7.5'te tipik bir 8-bit mikroişlemcide kullanılan bacak tanımlarından bazıları görülmektedir. Şekilde, mikroişlemcinin fiili entegre devre yongasında bir saat kaynağı (devre) içerdiği; fakat bunun, dıştan bağlanmış bir kristale (XTAL) gereksinim duyduğu varsayılmıştır. Böylece bu devre kararlı bir referans saati sağlar ve her komut, getirme ve yürütme için belli sayıda saat saykılına ihtiyaç duyduğundan, her komut, bilinen bir zaman süresinde gerçekleşir. Programcı kimi zaman bir program kod parçasını yürütmek için gereken süreyi bilmek isteyeceğinden, bu oldukça yararlıdır. Dolayısıyla, kod parçasındaki her bir komutu yürütmek için gereken saykıl sayısını sayarak ve her bir saat saykılının tam süresini bilerek bu süre kolayca belirlenir.

Yüklemede Dikkate Alınacak Noktalar

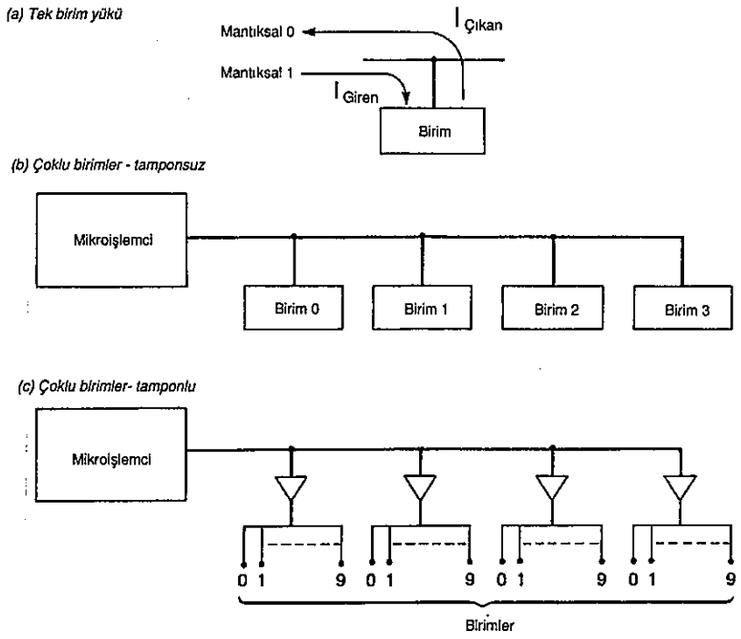
Bir mikroişlemci kullanırken, çıkış bacaklarında görülen çeşitli sinyalleri (adres, veri, denetim) üretmek için mikroişlemci içinde kullanılan sayısal birimlerin elektriksel özelliklerinin bilinmesi oldukça önemlidir. Bunlar çoğu zaman, bu birimlere doğrudan bağlanabilecek birimlerin sayı ve türüne belli sınırlamalar getirir.

Mikroişlemci sistemlerinde kullanılan pek çok devre - bellek entegreleri, G/Ç birimleri vb-, girişlerine mantıksal 0'a karşılık gelen bir gerilim düzeyi uygulandığında dışarıya sonlu değerde bir akım ($I_{çık}$) çıkaracak biçimde çalışırlar. Girişlerine mantıksal 1'e karşılık gelen bir gerilim düzeyi uygulandığında da sonlu değerde bir akım (I_{gir}) çekerler . Bu durum, Şekil 7.6(a)'da diyagram olarak gösterilmiştir.

Bu nedenle bu cihazlarla doğru biçimde çalışabilmek için mikroişlemci, çıkış sinyali

mantıksal 0 ise sonlu miktarda bir akımı alabilme (*çekme*) veya çıkış sinyali mantıksal 1 ise sonlu miktarda bir akımı üretme (*kaynaklama*) yeteneğine sahip olmalıdır. Ancak, bu akımlar normalde eşit değildirler ve çoğu zaman bir cihazın çekme gerekleri (mantıksal 0) kaynak gereklerinden (mantıksal 1) daha büyüktür - örneğin transistör transistör mantık (TTL).

Örnek olarak, bir mikroişlemcinin çıkış bacağının 1.8 mA'lik bir akımı kaynaklayıp çekebildiğini ve bu bacağa bağlanan cihazların mantıksal 0'a sürüldüğünde 0.4 mA *ürettiğini*, mantıksal 1'e sürüldüğünde de 0.2 mA *çektiğini* varsayalım. Böylece mikroişlemciye bağlanabilecek bu tip cihazların maksimum sayısı mikroişlemcinin mantıksal 0 çekme yeteneği ile belirlenir.



Şekil 7.6 Yüklemede dikkate alınacak noktalar

Bu örnekte, mikroişlemciye bu tür birimlerden en çok dört tane (4x0.4=1.6mA) bağlanabilir ve dolayısıyla mikroişlemcinin bu tür dört *birim yükünü sürebileceği* söylenir. Bu sayı aşıldığında mikroişlemci herhangi bir zarar görmeyecekse de, bu sınırın ötesinde, çıkış gerilim düzeyleri etkilenecek ve bu nedenle de bağlı olan cihazlar çıkmakta olan mantıksal sinyali yanlış yorumlayabileceklerdir - 1 yerine 0, 0 yerine 1 gibi. Bu durum Şekil 7.6 (b)'de gösterilmiştir.

Mikroişlemcinin sürme sınırına ulaşılırsa, mikroişlemcinin çıkış bacakları ile bağlı

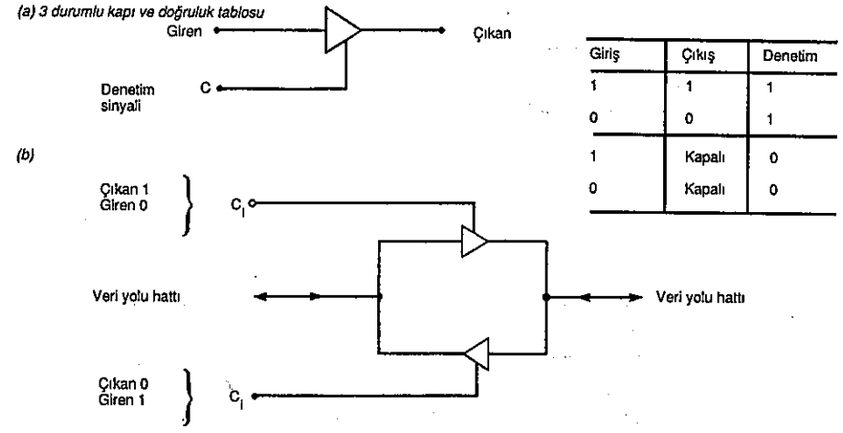
devreler arasına bir veya daha çok sayıda *tamponlama devresi* eklemek gerekir. Bu da, Şekil 7.6(c)'de gösterilmiştir. Şekildeki mikroişlemcinin dört cihaz sürebildiği varsayılmıştır; ancak,

S 7.3 kullanılan her bir tampon devresi aynı türden maksimum on cihazı sürebilir.

İki yönlü tamponlar

Tampon; bir mikroişlemci hattının akım kaynaklama/çekme yeteneğini artırmak için kullanılan bir cihazdır. Örneğin, Şekil 7.6'da görülen tamponlar *tek yönlü* tamponlar olarak bilinir ve bu nedenle de, sözgelimi, mikroişlemcinin adres ve denetim hatlarının sürme yeteneğini artırmak için elverişlidir.

Öte yandan veri yolu hatları, belli miktardaki bir akımı kaynaklayabilme/ çekebilmeye ek olarak, aynı zamanda iki yönlü modda da çalışabilmelidir ve dolayısıyla da iki yönlü tamponlar bir veri hattını tamponlarken kullanılmalıdırlar.



Giriş	Çıkış	Denetim
1	1	1
0	0	1
1	Kapalı	0
0	Kapalı	0

Şekil 7.7 İki yönlü tampon

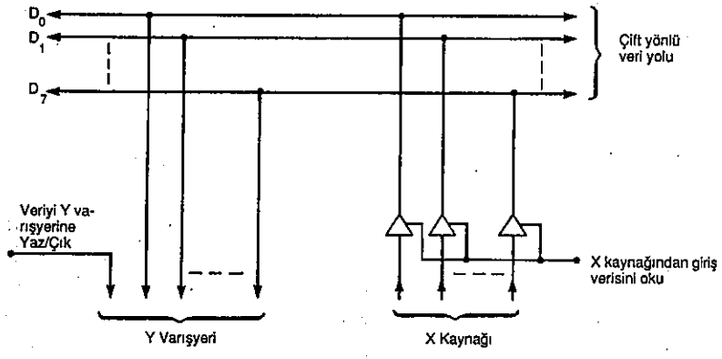
İki yönlü bir tampon, iki tane üç-durumlu kapıdan oluşur. Üç durumlu bir kapının doğruluk tablosu Şekil 7.7 (a)'da verilmiştir. Görülebileceği gibi, denetim girişi mantıksal 1 iken devre, normal bir tek yönlü tampon gibi çalışır ve mantıksal çıkış girişle aynıdır. Denetim girişi mantıksal 0 iken, devre ne akım kaynaklamakta ne de çekmekte olduğundan, çıkış *üçüncü kapalı duruma* kurulur.

Bu nedenle, iki yönlü bir tampon, Şekil 7.7 (b)'de gösterilen türde iki tane kapı kullanarak gerçekleştirilir. Şekilde, kapıların mikroişlemcinin veri hattını tamponlamak için kullanıldığı varsayılmıştır. Şekilden de çıkarılabileceği gibi, C_0 mantıksal 1 ve C_1 mantıksal 0 iken veri hattı, bir çıkış hattı gibi işlev görecektir. C_0 mantıksal 0 ve C_1 mantıksal 1 iken de veri hattı, bir giriş hattı işlevi görür. Dolayısıyla, C_0 tipik olarak

sırasıyla MEMWR veya IOWR'den ve C ,de MEMRD ve IORD'den sü-
S 7.4 rülebilecektir.

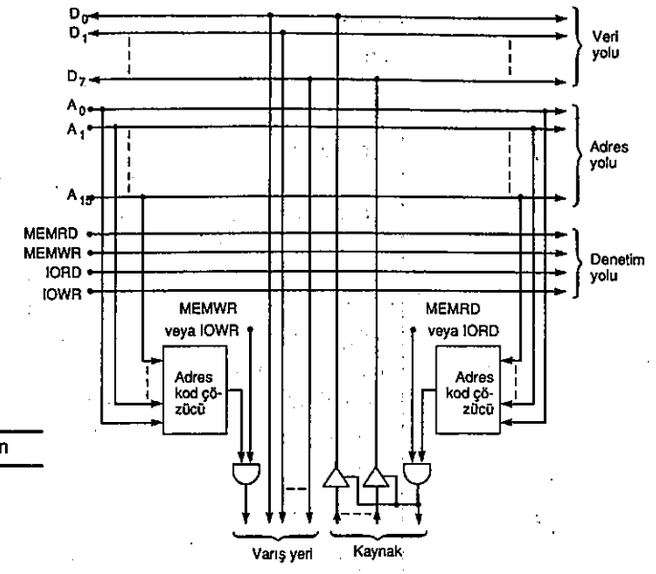
7.3 Mikrobilgisayar yolu

Mikrobilgisayar yolu, tüm bellek ve G/Ç birimlerini mikroişlemciye bağlamak için kullanılır. Dolayısıyla, tüm bu birimler aynı yol hatlarını ve sinyallerini kullandıklarından; birinci olarak, yola bağlı birimlerin mikroişlemciyle iletişim kurmak (veri aktarmak) istediğinde bu isteği belirleyebilmesi ve ikinci olarak da, veri yolu iki yönlü olduğundan, belli bir veri aktarımıyla ilgisi olmayan birimlerin yapılmakta olan aktarımı engellememesi sağlanmalıdır. Örneğin böyle bir durum, mikroişlemci bir cihaza mantıksal 0 gönderirken başka bir cihazın veri hattına bir mantıksal 1 çıkarmaya çalışması durumunda ortaya çıkabilir ve Şekil 7.8'de diyagramsal olarak gösterilmiştir.



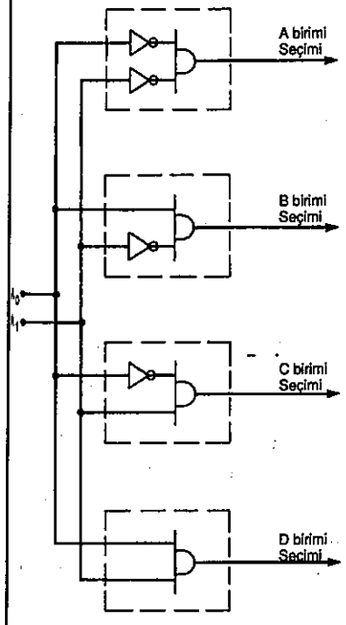
Şekil 7.8 Veri yolu aktarımları

Şekilden de çıkarılabileceği gibi, veri yoluna bir değer giriş veya çıkışı yapabilmek için veriye özel kaynak veya varışyerinin seçilmiş olması gerekir. Bu, adres yolunu ve kaynağın ve varışyerinin her biri için ek bir *adres kod çözücü* devreyi kullanmak suretiyle gerçekleştirilir. Mikroişlemci, adres yoluna aktarımla ilgili ya bellek konumunun veya G/Ç portunun kimliğine ilişkin bilgileri çıkarır ve adres kod çözücü devresi aracılığıyla, bu isteğe, yalnızca seçilmiş olan bellek veya G/Ç birimleri karşılık verir. Ardından, bir birim, adres yolunun durumundan belli bir yol aktarımıyla ilişkili olduğunu belirlerse, denetim hatlarında gösterilmiş bulunan seçilmiş işlemi gerçekleştirir: giriş işlemi için MEMRD veya IORD çıkış işlemi için MEMWR veya IOWR. Bu durum, Şekil 7.9'da diyagramsal olarak gösterilmiştir.



A_1	A_0	Seçilen birim
0	0	A
0	1	B
1	0	C
1	1	D

Şekil 7.9 Cihaz seçimi



Şekil 7.10 Adres kod çözme

Adres Kod Çözme

Adres kod çözücü, bir yol aktarım isteği işleminde kullanılıyorsa, mikrobilgisayar yoluna bağlı belirli bir bellek veya G/Ç portunu seçmek için kullanılır. Örnek olarak, sadece iki adres hattını, A_0 ve A_1, kullanan bir adres kod çözücüyü ele alalım. Böylece bu iki hattın (bitin) dört olası birleşimi Şekil 7.10'da gösterilmiştir.

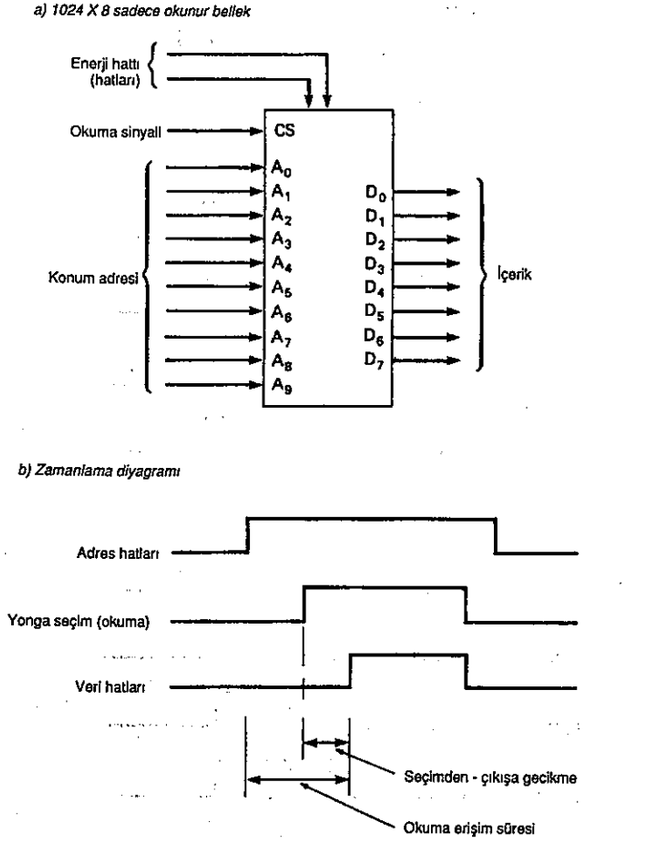
Şekilden de görüldüğü gibi, A_0 ve A_1'in her ikisi de mantıksal 0 olduğunda A cihazı seçilir (mantıksal 1); B,C,D cihazları için kullanılan kalan seçme hatlarının hepsi mantıksal 0 olacaktır. Benzer biçimde, A_0 = 1 ve A_1'i = 0 olduğunda da B cihazı seçilir ve diğer çıkışlar mantıksal 0 olur. Görüleceği gibi, bir adres kod çözücü, mantıksal VE ve tersleyici kapılar kullanarak kolaylıkla gerçekleştirilebilir. Burada gösterilen kod çözücüler sadece iki giriş kullanıyor olsa da, aynı ilkeler aracılığıyla daha büyük kod çözücüler de kolayca gerçekleştirilebilir. Bundan sonraki kısımlarda bu cihazların kullanımları ayrıntılı olarak izah edilecektir.

7.4 Bellek birimi

2. Bölümde belirtildiği gibi, bir mikrobilgisayar bellek birimi iki tip üniteden meydana gelir: komutlar listesini (program) tutmak (saklamak) için kullanılan sadece-okunur bellek ve bu programla ilgili veriyi tutmak için kullanılan rastgele-erişimli bellek. Burada her bir cihaz türünün iç organizasyonu ve mikrobilgisayar yolu ile
S 7.5 olan arabirim üzerinden bağlantısı ele alınacaktır.

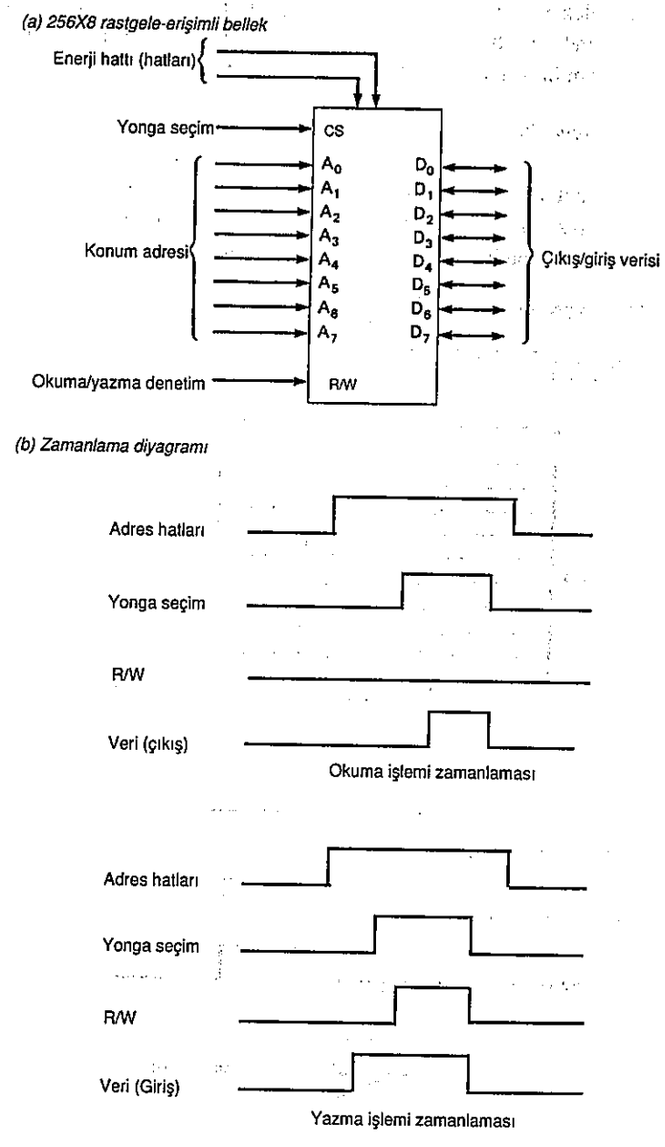
Sadece-Okunur Bellek

Halen mevcut ve kullanılmakta olan farklı türde -örneğin maske programlı (ROM) veya elektriksel olarak yeniden programlanabilir (EPROM)- bir grup sadece-okunur bellek varsa da, bu birimlerin temel işlem modu aynıdır. İstenen konumun adresi bi-



Şekil 7.11 Sadece-okunur belleğin çalışması

rimin giriş adres hattına çıkarılır ve bir bellekten okuma sinyali verilir. Ardından cihaz da buna, seçilen yerin içeriğini veri çıkış hatlarına çıkararak karşılık verir. Bu, Şekil 7.11'de gösterilmiştir. Şekilde her biri 8-bitlik 1024 konumdan oluşan tipik bir sadece-okunur bellek cihazı görülmektedir.



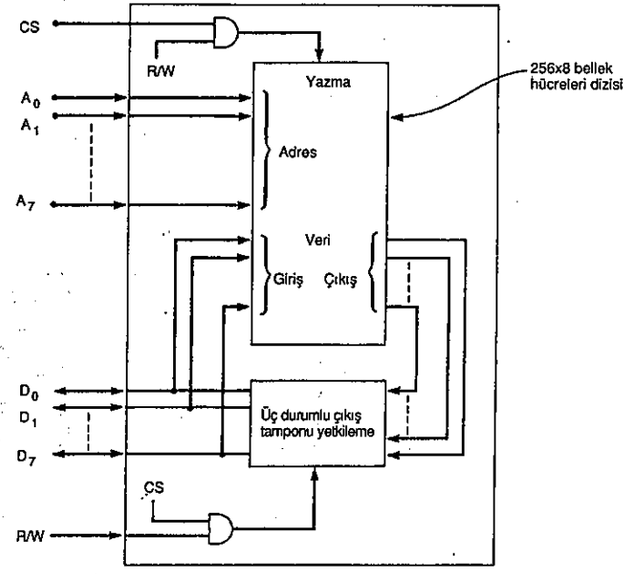
Şekil 7.12 Rastgele-erişimli belleğin çalışması

Şekilden de görülebileceği gibi, oku sinyali birim yonga seçim (CS) girişine bağlanır, çünkü adından da anlaşılacağı gibi bu, seçilmiş olan birimi (yongayı) belirtmek için kullanılır. Daha sonra görüleceği gibi, oku sinyali denetim yolundaki bir MEMRD olmayıp, daha çok bu sinyalin ve ek adres kod çözücü devreleri çıkışının bir birleşimidir.

Şekilde görülen birleşik zamanlama diyagramından, adres ve yonga seçim sinyallerinin çıkarılmasıyla verinin çıkış bacaklarından çıkarılması arasında kısa bir zaman gecikmesi ortaya çıktığını görülmektedir. Bu gecikme, yongada bulunan bellek dizisi bileşenlerinin elektriksel özelliklerinden kaynaklanmaktadır, bunlar sırasıyla cihazın okuma erişim süresi ve seçimden-çıkışa-gecikme olarak bilinirler. Anımsanacağı gibi, mikroişlemcinin getirme saykılının zamanlama sırası ele alınırken (Şekil 7.2), önce bellek adresini çıkarmak, sonra da saklı komut baytını komut kaydedicisine (IR) yüklemek için farklı zaman (saat) periyodları kullanılmıştı. Bu, özellikle, her ikisi de mikrosaniyenin tipik kesirleri oranındaki bü-
S 7.6 yüklüklerle gösterilen zaman gecikmelerine olanak sağlamak için yapılmıştır.

Rastgele-Erişimli Bellek

Tipik bir rastgele-erişimli bellek biriminin şematik diyagramı Şekil 7.12'de gösterilmiştir. Gösterilen bu cihaz her biri 8-bitlik olan 256 konumdan meydana gelmiştir. Şekilde de görüldüğü gibi, bir değer (adres giriş hatlarında belirtilen) bir ko-



Şekil 7.13 RAM'in iç organizasyonu

numdan okunsa ya da ona yazılsa da , birim yonga seçim girişine ek olarak yalnızca bir tane okuma/yazma (R/W) denetim bacağına sahiptir. Birim için gerekli olan bacak sayısını, dolayısıyla birimin maliyetini azaltmak için böyle yapılmıştır.

Bir bellek işlemini (okuma veya yazma) gerçekleştirmek için, önce uygun konumun adresi, adres giriş hatlarına çıkarılır ve yonga seçim girişi etkinleştirilir. Daha sonra R/W hattının durumuna bağlı olarak, ya adreslenmiş konumun içeriği okunur (R/W = 0) ya da veri hatlarının o anki değeri adreslenmiş ilgili konuma yazılır (R/W = 1). Tek bir R/W denetim hattının çalışmasını göstermek üzere, tipik bir RAM yongasının iç organizasyonunun şematik diyagramı Şekil 7.13'te gösterilmiştir. R/W girişinin durumu ile belirtilen biçimde, adreslenmiş konum ya okunur ya da ona yazılır. Eğer R/W girişi alçak düzeydeyse (yani, 0 ise), bir okuma işlemi belirtilmiş olur ve dolayısıyla bellek dizisinin veri çıkış hatlarında hazır olan veri, birimin veri hatlarına uygulanır. Eğer R/W girişi yüksek düzyerdeyse (yani 1 ise), bellek dizisinden gelen çıkış yetkisizlenir ve onun yerine veri hatlarında bulunan veri diziye
S 7.7 yazılır.

Sistem Bellek Haritası

Tipik olarak bir mikrobilgisayarın bellek birimi, bu türde birçok ROM ve RAM birimleri içerir. Örneğin bir uygulama; sözgelimi, saklı program için 3 Kbayta gereksinim duyuyorsa, 3 tane 1024x8-bitlik ROM'a ihtiyaç olacaktır. Benzer şekilde, eğer RAM bölümü veri için 1 Kbayta gereksinim duyuyorsa, dört tane de 256x8-bitlik RAM gerekecektir.

Bu nedenle, bir mikrobilgisayarın bellek birimini gerçekleştirirken her bir birim türü için hangi bellek adres aralığının kullanılacağına karar verilmesi gerekir. Bu, sistemin bellek haritası olarak bilinir ve örnek olarak yukarıdaki sisteme uygun bir bel-
S 7.8 lek haritası Şekil 7.14'te gösterilmiştir.

	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0000-03FF	0	0	0	0	0	0	Yonga adresi										ROM 1
0400-07FF	0	0	0	0	0	1	Yonga adresi										ROM 2
0800-0BFF	0	0	0	0	1	0	Yonga adresi										ROM 3
0C00-0CFF	0	0	0	0	1	1	0	0	Yonga adresi								RAM 1
0D00-0DFF	0	0	0	0	1	1	0	1	Yonga adresi								RAM 2
0E00-0EFF	0	0	0	0	1	1	1	0	Yonga adresi								RAM 3
0F00-0FFF	0	0	0	0	1	1	1	1	Yonga adresi								RAM 4

Şekil 7.14 Bir sistem belleği haritası

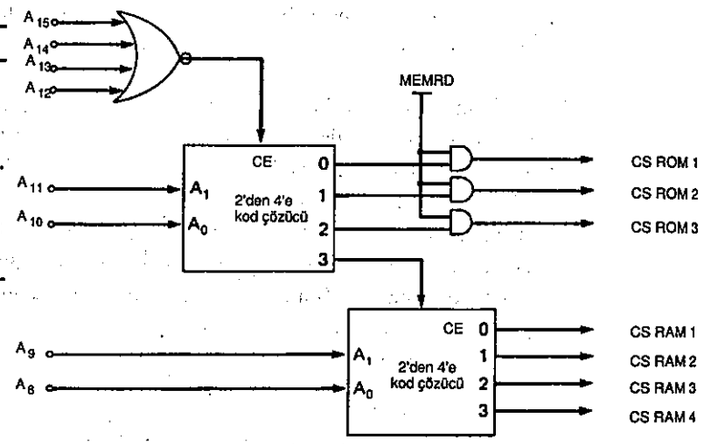
Haritada, üstte on altı adres hattı, altta da her bir birim tipi için gerekli olan adres aralığı sıralanmıştır. Haritadan şunlar çıkarılabilir :

1 Geçerli bir bellek adresi için A_{12}-A_{15} arası adres hatları mantıksal 0 olmalıdır.
2 A_{10} ve A_{11} adres hatlarının ikisi de mantıksal 1 iken adresin bir RAM birimi için, değilse adresin bir ROM için olduğu düşünülmüştür.
3 Seçilen belirli bir ROM birimi, A_{10} ve A_{11} adres hatlarıyla belirlenir.
4 Seçilen belirli bir RAM birimi, A_8 ve A_9 adres hatlarıyla belirlenir.
5 Seçilen ROM birimindeki konum A_0 - A_9 arası hatlarla belirlenir.
6 Seçilen RAM birimindeki konum A_0 - A_7 arası hatlarla belirlenir.

(b) Kod çözücü doğruluk tablosu

CE	A_1	A_0	0	1	2	3
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	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

(a) Kod çözme devresi



Şekil 7.15 Bellek adres kod çözme

Yukarıdaki şemayı gerçekleştirmek için gereken gerçek adres kod çözücü devre, daha önce izah edilene benzer iki tane 2-hattan - 4-hatta kod çözücü kullanılarak kolayca meydana getirilebilir. Uygun bir düzenleme Şekil 7.15'te gösterilmiştir.

Görüldüğü gibi, adres kod çözücülerde, bir bellek biriminde yonga seçim girişinin oynadığı role benzer bir rol oynayan ek bir yonga yetkileme (CE) giriş hattı bulunur. Böylece yonga yetkileme girişi mantıksal 1 olduğunda, çıkış iki girişin birleşimleri tarafından belirlenir; fakat erkleme mantıksal 0 iken, dört çıkışın tümü de mantıksal 0'dır.

Dolayısıyla birinci kod çözücüde bu, geçerli bir bellek adresinin hazır olduğunu (A12'den A15'e hepsi mantıksal 0) belirlemek için dört giriş, VEYADEĞİL kapısı (VEYA ve onu takip eden bir tersleyici kapısı) ile bağlantılı olarak kullanılır. İkinci kod çözücüde ise bir RAM birim adresininin hazır olduğunu belirlemek için kullanılır.

S 7.9

7.5 G/Ç Arabirim ünitesi

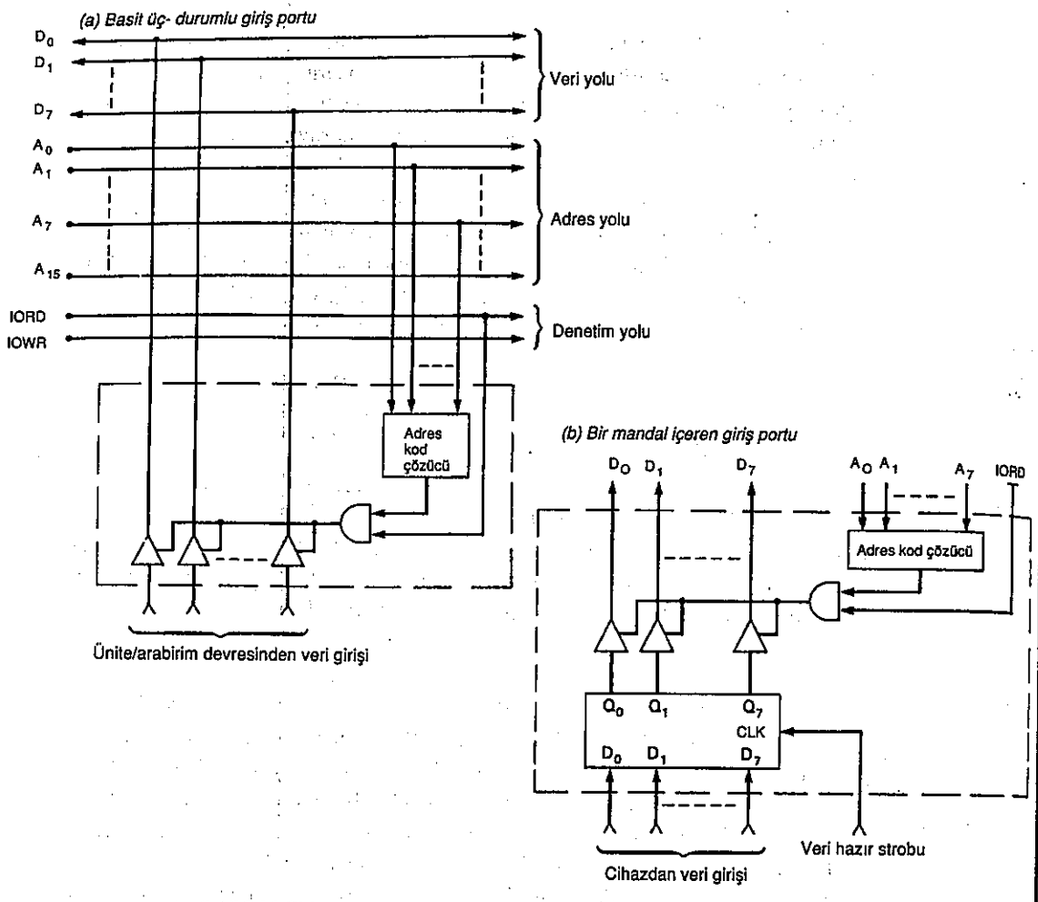
Bir önceki bölümde açıklandığı gibi, G/Ç (I/o) arabirim ünitesi temelde bir grup giriş ve çıkış portundan oluşur. Bunlar mikrobilgisayar ile mikrobilgisayara bağlı bulunan cihaz veya arabirim devreleri arasında arabirimi meydana getirirler ve etkin olarak mikrobilgisayar yolunu yapılmakta olan herhangi bir giriş veya çıkış işleminden ayırırlar.

G/Ç arabirim ünitesi, bellek birimi ile aynı mikrobilgisayar yoluna bağlanır; dolayısıyla adres yolu da, veri aktarımı ile ilgili portu belirlemede kullanılır. Öte yandan, ayrı IORD ve IOWR denetim hattı kullanımı da, birime bellek ile G/Ç aktarımını ayırt etme ve istenen aktarımın yönünü belirleme olanağı sağlar. Bundan başka, pek çok mikrobilgisayarda maksimum 256 giriş ve/veya çıkış portu bulunduğundan, belli bir port adresini belirtmek için normalde sadece sekiz tane adres hattı kullanılır. Bunlar çoğunlukla adres yolunun en küçük değerlikli sekiz (A_0 - A_7 arası) hattıdır.

Giriş Portu

Giriş portunun en basit biçimi, Şekil 7.16(a)'da da görüldüğü gibi, sekiz tane üçdurumlu tampondan oluşan bir dizidir. Dolayısıyla mikroişlemci, bir portun girişinde bulunan o anki değeri okumak istediğinde, adres yolunun en küçük değerlikli sekiz hattına portun adresini çıkarır ve bir IORD sinyali üretir.

Açıkçası, bu yaklaşımda giriş birimi veya arabirim devresi, veri değerini, mikroişlemci tarafından okunana kadar, tampon girişinde sabit olarak tutabilmelidir. Bu yüzden alternatif ve daha esnek bir düzenleme, Şekil 7.16(b)'de de görüldüğü gibi porta 8-bit bir mandalı eklemek olmaktadır.



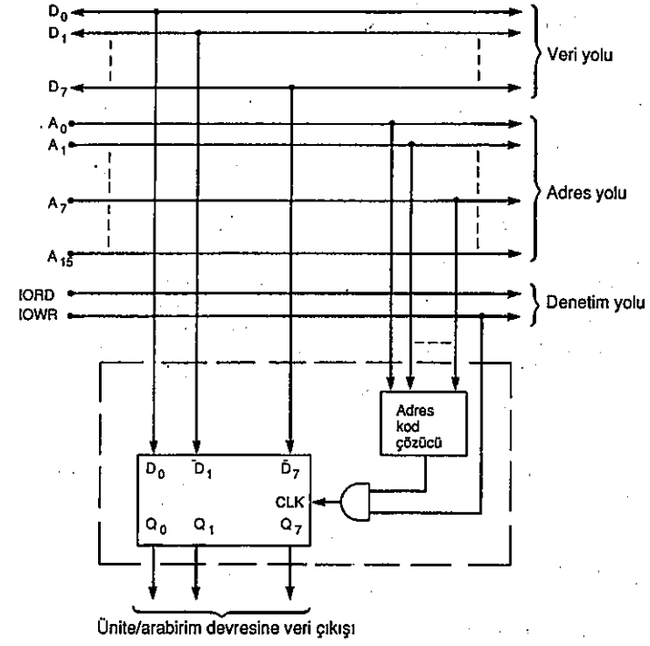
Şekil 7.16 Giriş portu şeması

Mandallar, sayısal bir değeri tutmak veya saklamak için kullanılan sayısal birimlerdir. Değer bir mandalda tutulmak ya da saklanmak istendiğinde, mandalın veri girişlerine (D) verilir ve ortak saat girişi (CLK) açılıp kapatılır ya da saat darbesi uygulanır. Giriş değeri daha sonra değişebilir; ancak çıkış (Q), cihazın saati tekrar kurulana kadar aynı kalır.

Bu yüzden, mandal içeren bir giriş portunu kullanırken harici birim ya da devre, tipik olarak mandalın saatini belli bir değere kurar ve mikroişlemci bunun ardından, hazır olduğunda bu değeri okur.

Çıkış Portu

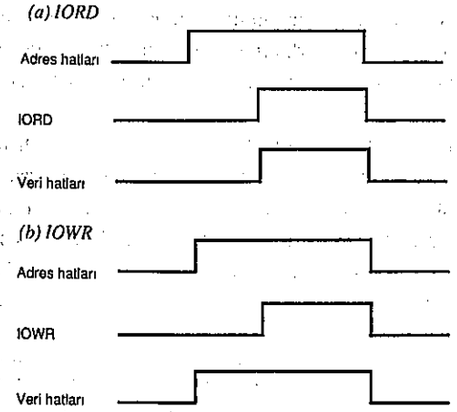
En basit biçimdeki bir çıkış portunun, yalnızca, (kendi adresinin adres yolunda bulunup bulunmadığını belirlemek için) adres kod çözücü devresi ve adres seçimi ile IOWR denetim sinyalini birleştirmek için bir VE kapısı içermesi yeterlidir. Öte yandan bu da, çıkış biriminin birinci olarak çıkışı yapılacak her bir veri değerini beklemesini, ikinci olarak da, veri değeri çıktığında onu koruyabilmesini ya da saklayabilmesini gerektirir. Bu nedenle bir çıkış portunun, bir adres kod çözücüye ek olarak, -mikroişlemci tarafından çıkarılan bir veri değerinin bağlı bir birim veya arabirim devresi tarafından kabul edilene kadar çıkış portunda kalmasını sağlamak üzere- 8-bit lik bir mandal içermesi de olağandır. Bu, Şekil 7.17'de gösterilmiştir.



Şekil 7-17 Çıkış portu şeması

Zamanlamada Dikkate Alınacak Noktalar

Mikroişlemci tarafından üretilen IN (giriş) ve OUT (çıkış) komutları için üretilen zamanlama dalga biçimleri, sırasıyla daha önce verilmiş olan bellekten okuma ve belleğe yazma komutlarına ilişkin dalga biçimleriyle benzerlik gösterir. Dolayısıyla, bir IN komutunu yürütürken mikroişlemci, önce istenen (komutta belirtilen) port adresini en küçük değerlikli sekiz adres hattına çıkarır ve bir IORD sinyali üretir. Ardından, seçilen port buna veri yolu üzerindeki üç-durumlu tamponlarda hazır bulunan değeri geçitleyerek karşılık verir ve sonra da mikroişlemci bu değeri A kaydedicisine yükler. Bir IN komutunun tipik zamanlama dalga biçimleri grubu Şekil 7.18(a)'da gösterilmiştir.



Şekil 7.18 G/Ç zamanlama dalga biçimleri

Bir OUT komutunun tipik bir zamanlama dalga biçimleri grubu Şekil 7.18(b)'de görülmektedir. Mikroişlemci önce istenen (komutta belirtilen) port adresini adres yoluna ve çıkarılacak değeri de (A kaydedicisinin, yürürlükteki değeri) veri yoluna çıkarır. Daha sonra da bir IOWR sinyali üretir. Seçilen port buna, veri yolundan değerin saatini port mandalına saat denetimi altında göndererek (mandallayarak) karşılık verir.

S 7.10

Programlamabilir G/Ç Birimleri

Az önce özetlenen temel mandallama ve tamponlama işlevlerini yapabilmek için mevcut tümleşik devreler bulunsa da, G/Ç (I/o) arabirim devrelerinin varlığı pek çok mikrobilgisayar uygulamalarının temelini oluşturduğundan, mikroişlemci üreticileri G/Ç arabirim devresinin tamamını tek bir entegre devrede sunarlar. Tipik olarak bunlar, iki veya daha çok giriş ve/veya çıkış portu ile her bir port ile ilgili onay denetim hatlarını içerir. Bunlar *programlanabilir giriş/çıkış birimleri veya PIO* olarak bilinir.

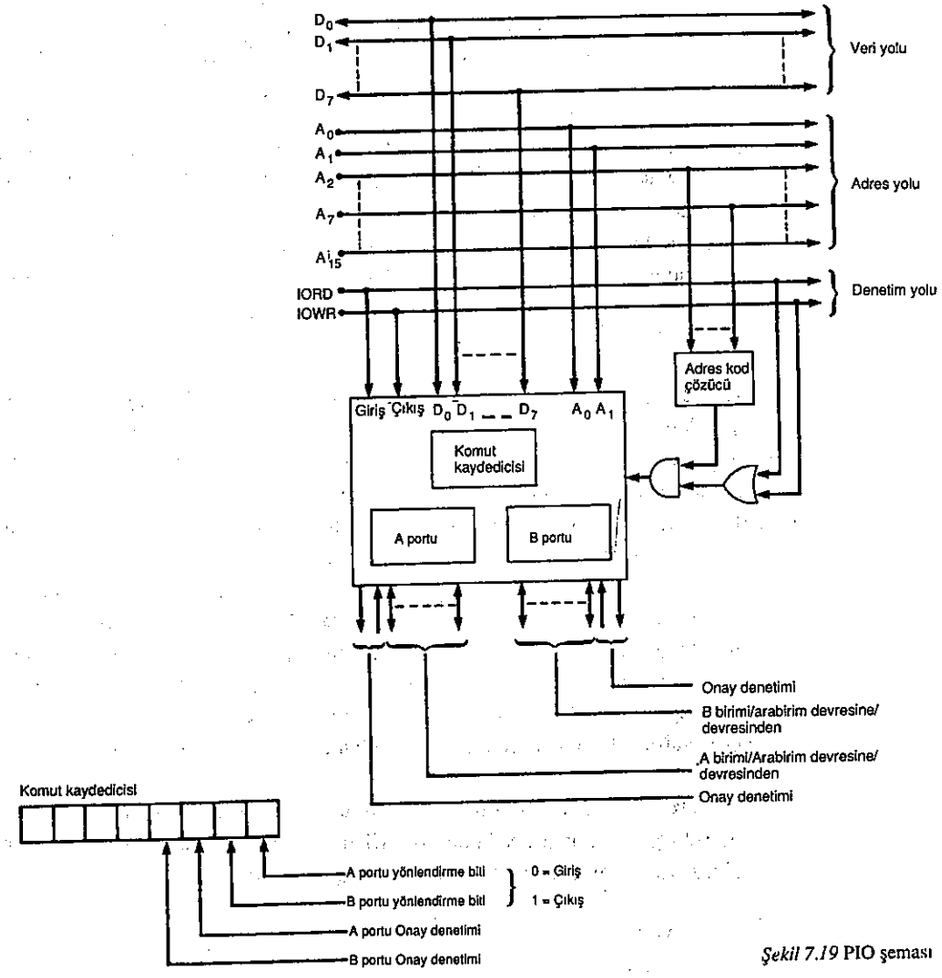
Bir G/Ç portu ile bir çevresel birim veya arabirim devresi arasındaki veri aktarımının düzenlenmesini sağlayan onay yönteminin çalışması bir önceki bölümde ele alınmıştı. Verinin kaynağı (çıkış için mikroişlemci, giriş için cihaz veya arabirim devresi) veriyi veri hatlarına yerleştirdiğinde, bunu DAV hattı üzerinde; verinin alıcısı veriyi kabul ettiğinde (okuduğunda) ise bunu, DACC hattı üzerinde belirtir.

Tipik bir PIO ve bunun mikrobilgisayar yolu ile arabiriminin şematik diyagramı Şekil 7.19'da gösterilmiştir. Şekilde gösterilen PIO'da onay denetim hatlarıyla birlikte iki tane iki yönlü portun, A ve B, bulunduğu varsayılmıştır.

Portların iki yönlü özelliği, her bir portun ya giriş portu ya da çıkış portu olarak

programlanabileceği anlamına gelmektedir. Kullanıcı, istenen fonksiyonu, programın kullanıma hazırlanma bölümünde, komut kaydedicisindeki belli bitlere (giriş için) mantıksal 0 ve (çıkış için de) mantıksal 1 yazarak seçer.

Komut kaydedicisindeki öteki bitler onay denetim hatlarıyla bağlantılı olarak kullanılır. Örneğin, A portunun bir giriş portu olarak programlanmış olduğunu varsayarsak, yeni bir veri değerinin, bağlı birim veya arabirim devresi aracılığıyla port mandalına girildiği her durumda, komut kaydedicisindeki belli bir bit 1'e kurulur. Dolayısıyla, mikroişlemci düzenli zaman aralıklarıyla komut kaydedicisinin içerdiği değeri girer ve bu biti gözden geçirir. Daha sonra bitin 1'e kurulmuş olduğunu be-



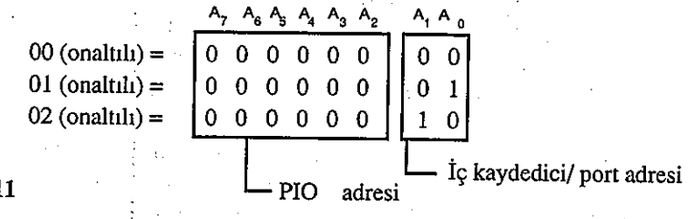
Şekil 7.19 PIO şeması

lirlediğinde (dolayısıyla port mandalına yeni bir değer girildiğinde) bu değeri porttan okur. Mikroişlemci tarafından porttan bir değer okuma (giriş) işi normalde komut kaydedicisindeki ilgili biti sıfırlar; ardından da PIO, bağlı birime yeni bir değerin girilebileceğini belirtmek için onay denetim hattı üzerinde bir DACC (veri alındı) strobu üretir.

Yukarıdaki açıklamadan çıkarılabileceği gibi, mikroişlemci her bir giriş ve çıkış portuna ve aynı zamanda komut kaydedicisine hem giriş hem de çıkış yapmak isteyebilir. Ayrıca, PIO üç kaydedici içerdiğinden, kullanılan adres kod çözücü bu adreslerden herhangi birinin veri yolu üzerinde hazır olduğunu belirlemelidir.

Şekilden de görüldüğü gibi, bu iş normalde adres kod çözücü girişleri için altı yüksek-sıralı adres biti (A_2-A_7) ve aktarımla ilgisi olan belirli kaydedicileri -Komut, A portu veya B portu- belirtmek için de iki en küçük değerlikli adres bitini (A_0 ve A_1) kullanarak gerçekleştirilir. Dolayısıyla, eğer üç adres, sırasıyla 00 (onaltılı) 01 (onaltılı) ve 02 (onaltılı) ise; adres kod çözücünün çıkışı, A_2'den A_7'ye kadar altı adres hattının tümü de mantıksal 0 olduğu zaman, yüksek düzeyde (mantıksal 1) olacaktır:

S 7.11



Sorular

7.1 Tipik bir 8-bit mikroişlemcinin şematik blok diyagramını çizin ve bellekten aşağıdaki komutları getirmek için gereken adres ve veri yollarındaki bilgi akışını ve mikroişlemci tarafından üretilen denetim sinyallerini diyagram üzerinde gösterin:
(a) LD A,B
(b) LD B,7E
(c) LD HL,2A7E

7.2 Soru 7.1'i aşağıdaki önceden getirilmiş komutların yürütülmesini göstermek için tekrarlayın:
(a) ADD A,B
(b) ADD A,48
(c) ADD A, (HL)

7.3 Bir mikroişlemciye neden sınırlı sayıda sayısal birim bağlanabileceğini ve bu sınırın bir tampon yardımıyla nasıl genişletilebileceğini açıklayın.

7.4 Üç-durumlu bir kapının doğruluk tablosunu çizin ve buradan da tek yönlü ve iki yönlü bir tamponun çalışmasını açıklayın.

7.5 Doğruluk tablosu yardımıyla, bir adres kod çözücünün çalışmasını açıklayın. İki-girişli bir kod çözücünün yalnızca iki girişli VE kapıları ve tersleyiciler kullanarak nasıl gerçekleştirilebileceğini gösterin.

7.6 Bir 1024x8-bit ROM'un şematik diyagramını çizin ve bir zamanlama diyagramı yardımıyla çalışmasını açıklayın. Okuma erişme süresini ve seçimden-çıkışa-gecikmeyi diyagram üzerinde gösterin.

7.7 Tipik bir 256x8-bit RAM'ın iç organizasyonunu göstermek için şematik bir diyagram çizin ve zamanlama diyagramlarının da yardımıyla bir okuma ve yazma saykılı sırasındaki çalışmasını açıklayın.

7.8 Bir mikrobilgisayar sistemi, 4 Kbayt ROM ve 256 bayt RAM'e ihtiyaç duymaktadır. Eğer bunlar, iki blok 0000 (onaltılı) adresinden başlayarak bitişik bellek adreslerini işgal ediyorlarsa, her bir bellek blokunun başlangıç ve bitiş adreslerini belirleyin.

7.9 Soru 7.8'de açıklanan sistemin bellek haritasını çizin ve 1 Kbayt ROM ve 256 bayt RAM yongalarının kullanıldığını varsayarak, bir bellek blokunun mikrobilgisayar yolu ile arabirim üzerinden bağlantısını sağlamak için gereken adres kod çözücü devreyi geliştirin.

7.10 Tamponlu bir giriş portu ile tamponlu bir çıkış portunu da işin içine katan bir G/Ç arabirim ünitesinin şematik diyagramını çizin. Her blokun çalışmasını ve mikrobilgisayar yolu ile arabirim üzerinden bağlantısının nasıl yapıldığını açıklayın.

7.11 Programlanabilir bir G/Ç biriminin (PIO) şematik blok diyagramını çizin. Bu birimin çalışmasını ve mikrobilgisayar yolu ile arabirim üzerinden bağlantısının nasıl yapıldığını açıklayın.

Ek-1 : Zilog Z80 komut takımının bir alt kümesi

Zilog Z80 mikroişlemcisinde yer alan komutlardan bazıları aşağıdaki tabloda verilmiştir. Komutlar sembolik gösterim biçiminde sunulmuş olup, her birinin onaltılı kodu da birlikte verilmiştir. Aşağıdaki kısaltmalar kullanılmıştır :

A,B,C,D,E,H,L : Mikroişlemcinin kaydedicilerini gösterir
(HL) : H,L kaydedici çiftinde o sırada tutulan bellek adresini gösterir
n : 8-bitlik (2 onaltılı karakter) bir değeri veya baytı gösterir
nn : 16-bitlik (2 bayt) bir değeri gösterir

Burada seçilen komutlar Intel 8080 ve Intel 8085 mikroişlemcilerinde de yer almaktadır; dolayısıyla, 8-bitlik bir mikroişlemcide yer alan tipik bir komut takımı listesidir.

I VERİ TAŞIMA KOMUTLARI

Bu komutlar, mikrobilgisayardaki bir kaydedici veya bellek konumundan bir diğerine veri taşıma olanağı sağlarlar. Koşul bayrakları bu komutlar tarafından etkilenmez.

Kaydedici adresleme - 8-bitlik veri

LD	A,A	7F	LD	B,A	47	LD	C,A	4F	LD	D,A	57
	A,B	78		B,B	40		C,B	48		D,B	50
	A,C	79		B,C	41		C,C	49		D,C	51
	A,D	7A		B,D	42		C,D	4A		D,D	52
	A,E	7B		B,E	43		C,E	4B		D,E	53
	A,H	7C		B,H	44		C,H	4C		D,H	54
	A,L	7D		B,L	45		C,L	4D		D,L	55

LD	E,A	5F	LD	H,A	67	LD	L,A	6F
	E,B	58		H,B	60		L,B	68
	E,C	59		H,C	61		L,C	69
	E,D	5A		H,D	62		L,D	6A
	E,E	5B		H,E	63		L,E	6B
	E,H	5C		H,H	64		L,H	6C
	E,L	5D		H,L	65		L,L	6D

Kaydedici adresleme - 16-bitlik (2 bayt) veri

EX DE,HL EB

Dolaysız adresleme - 8-bitlik veri

LD	A,n	3E
	B,n	06
	C,n	0E
	D,n	16
	E,n	1E
	H,n	26
	L,n	2E
	(HL),n	36

16-bitlik veri

LD	BC,nn	01
	DE,nn	11
	HL,nn	21
	SP,nn	31

Doğrudan (genişletilmiş) adresleme - 8-bitlik veri

LD	A,(nn)	3A
	(nn),A	32

16-bitlik veri

LD	HL,(nn)	2A
	(nn),HL	22

Kaydedici dolaylı adresleme - 8-bitlik veri

LD	A,(HL)	7E
	B,(HL)	46
	C,(HL)	4E
	D,(HL)	56
	E,(HL)	5E
	H,(HL)	66
	L,(HL)	6E

LD	(HL),A	77
	(HL),B	70
	(HL),C	71
	(HL),D	72
	(HL),E	73
	(HL),H	74
	(HL),L	75

2 ARİTMETİK VE MANTIK KOMUTLARI

Kaydedici adresleme - Toplama (*)

ADD	A,A	87
	A,B	80
	A,C	81
	A,D	82
	A,E	83
	A,H	84
	A,L	85

ADC	A,A	8F
	A,B	88
	A,C	89
	A,D	8A
	A,E	8B
	A,H	8C
	A,L	8D

Kaydedici adresleme - Çıkarma (*)

SUB	A	97
	B	90
	C	91
	D	92
	E	93
	H	94
	L	95

SBC	A,A	9F
	A,B	98
	A,C	99
	A,D	9A
	A,E	9B
	A,H	9C
	A,L	9D

VE (*)

AND	A	A7
	B	A0
	C	A1
	D	A2
	E	A3
	H	A4
	L	A5

VEYA (*)

OR	A	B7
	B	B0
	C	B1
	D	B2
	E	B3
	H	B4
	L	B5

Özel VEYA (*)

XOR	A	AF
	B	A8
	C	A9
	D	AA
	E	AB
	H	AC
	L	AD

Karşılaştırma (*)

CP	A	BF
	B	B8
	C	B9
	D	BA
	E	BB
	H	BC
	L	BD

Artırma (**)

INC	A	3C
	B	04
	C	0C
	D	14
	E	1C
	H	24
	L	2C

Azaltma (**)

DEC	A	3D
	B	05
	C	0D
	D	15
	E	1D
	H	25
	L	2D

Artırma/Azaltma kaydedici çifti (++)

INC	BC	03
	DE	13
	HL	23

DEC	BC	0B
	DE	1B
	HL	2B

Dolaysız adresleme - Toplama/Çıkarma (*)

ADD	A,n	C6	SUB	A,n	D6
ADC	A,n	CE	SBC	A,n	DE

VE/VEYA (*)

AND	n	E6	OR	n	F6

Özel VEYA/Karşılaştırma (*)

XOR	n	EE	CP	n	FE

Kaydedici dolaylı adresleme (*)

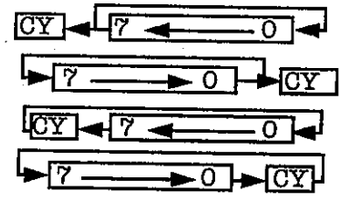
ADD	A,(HL)	86	SUB	(HL)	96
ADC	A,(HL)	8E	SBC	(HL)	9E
INC	(HL)	34	DEC	(HL)	35
AND	(HL)	A6	OR	(HL)	B6
XOR	(HL)	AE	CP	(HL)	BE

Ondalık ayarlama akümülatörü (*)

DAA	27

Kaydırma komutları (+)

RLCA	07
RRCA	0F
RLA	17
RRA	1F



Notlar : * Tüm bayraklar etkilenir
** Elde bayrağı dışındaki tüm bayraklar etkilenir
+ Yalnızca elde bayrağı etkilenir
++ Hiçbir bayrak etkilenmez

3 DENETİM AKTARMA KOMUTLARI

Bu komut grubu; program yürütmesinin normal sıralı akışını etkiler. Aşağıdaki koşul kodları kullanılır :

NZ	= sıfırdan farklı	(Z=0)	PO	= tek eşlik	(P=0)
Z	= sıfır	(Z=1)	PE	= çift eşlik	(P=1)
NC	= elde-yok	(CY=0)	P	= işareti pozitif	(S=0)
C	= elde	(CY=1)	M	= işareti negatif	(S=1)

Atlama

JP	nn	C3	JP	PO, nn	E2
JP	NZ,nn	C2	JP	PE,nn	EA
JP	Z,nn	CA	JP	P,nn	F2
JP	NC, nn	D2	JP	M, nn	FA
JP	C, nn	DA			

Altyordam çağırma/dönüş

CALL	nn	CD	RET		C9

4 GİRİŞ/ÇIKIŞ KOMUTLARI

Bu komutlar A kaydedicisi ile belirtilen bir port arasında bir G/Ç işlemi gerçekleştirirler:

IN	A,(n)	DB	OUT	(n),A	D3

5 SEÇİLMİŞ MAKİNE DENETİM KOMUTLARI

İşlemci işlemleri

NOP (işlem yapma)	00
HALT	76

Yığın işlemleri

PUSH	AF	F5	POP	AF	F1
	BC	C5		BC	C1
	DE	D5		DE	D1
	HL	E5		HL	E1

Ek-2 : ISO/ASC II 7-bitlik veri kod tablosu

Karşı sayfadaki tabloda, kod takımının tümü verilmiştir. Harfle belirtilen gruplar, basılmayan denetim karakterleridir ve bunlar içinde önemli olanlar CR= Satır başı, LF = Satır atlatma, SP = Boşluk ve DEL = Silme karakterleridir. Buradaki eşlik biti çift eşliktir ve 7-bitlik kodla birlikte her bir karakter için toplam 8-bit üretir.

Çift eşlik biti	*7-bitlik sekizli kod*	*Karakter*	*Çift eşlik biti*	*7-bitlik sekizli kod*	*Karakter*	*Çift eşlik biti*	*7-bitlik sekizli kod*	*Karakter*
0	000	NUL	0	053	+	0	126	V
1	001	SOH	1	054	,	1	127	W
1	002	STX	0	055	-	1	130	X
0	003	ETX	0	056	.	0	131	Y
1	004	EOT	1	057	/	0	132	Z
0	005	ENQ	0	060	Ø	1	133	[
0	006	ACK	1	061	1	0	134	\
1	007	BEL	1	062	2	1	135	]
1	010	BS	0	063	3	1	136	↑
0	011	HT	1	064	4	0	137	←
0	012	LF	0	065	5	0	140	`
1	013	VT	0	066	6	1	141	a
0	014	FF	1	067	7	1	142	b
1	015	CR	1	070	8	0	143	c
1	016	SO	0	071	9	1	144	d
0	017	SI	0	073	:	0	145	e
1	020	DLE	1	073	;	0	146	f
0	021	DCI	0	074	<	1	147	g
0	022	DC2	1	075	=	1	150	h
1	023	DC3	1	076	>	0	151	i
0	024	DC4	0	077	?	0	152	j
1	025	NAK	1	100	@	1	153	k
1	026	SYN	0	101	A	0	154	l
0	027	ETB	0	102	B	1	155	m
0	030	CAN	1	103	C	1	156	n
1	031	EM	0	104	D	0	157	o
1	032	SUB	1	105	E	1	160	p
0	033	ESC	1	106	F	0	161	q
1	034	FS	0	107	G	0	162	r
0	035	GS	0	110	H	1	163	s
0	036	RS	1	111	I	0	164	t
1	037	US	1	112	J	1	165	u
1	040	SP	0	113	K	1	166	v
0	041	!	1	114	L	0	167	w
0	042	"	0	115	M	0	170	x
1	043	#	0	116	N	1	171	y
0	044	$	1	117	O	1	172	z
1	045	%	0	120	P	0	173	{
1	046	&	1	121	Q	1	174	|
0	047	'	1	122	R	0	175	}
0	050	(	0	123	S	0	176	~
1	051	)	1	124	T	1	177	DEL
1	052	*	0	125	U			

Soruların cevapları

1.2 27= 00011011
63= 00111111
226=11100010

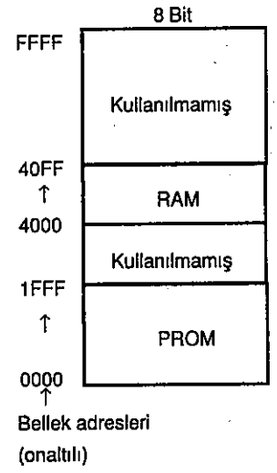
1.3 00101101 = 45
10101010 = 170
11011011 = 219

1.4 101 011 = 53_8
111 001 110 = 716_8
100 010 110 000 = 4260_8

1.5 C2 = 1100 0010
8E1 = 1000 1110 0001
3B2F = 0011 1011 0010 1111

2.1 On bitin 2^{10}, yani 1024 birleşimi; iki bitin 2^2, yani 4 birleşimi vardır. Böylece 12 bit, 4K bellek adresleyebilir.

2.2 256 konum için 8 bit gerekir, 00 →FF (onaltılı).
1024 (1K) konum için 10 bit gerekir, 000 →3FF (onaltılı).
8192 (8K) konum için 13 bit gerekir, 0000→1FFF (onaltılı).
Ayrıca Şekil A'ya bakınız.



Şekil A

2.3 ROM için, 0000 →07FF = 11 bit = 2 K (2048) konum.
RAM için, 2000 →23FF = 10 bit = 1 K (1024) konum.

3.4 (A) = D6 (onaltılı); (B) = 28;
(C) = 00; (D) = 28;
(E) = FF; (H) = 28;
(L) = 00.

3.5

Sembolik komutlar	*İşlem*
LD A,FF	(A) ← FF (onaltılı)
LD B,86	(B) ← 86 (onaltılı)
LD D,A	(D) ← FF (onaltılı)
LD E,B	(E) ← 86 (onaltılı)
LD HL,28A6	(HL) ← 28A6 (onaltılı)
EX DE,HL	(DE) ← 28A6 (onaltılı)
	(HL) ← FF86 (onaltılı)

3.6

Sembolik komutlar	*İşlem*
LD A,AA	(A) ← AA (onaltılı)
LD (2800),A	(2800) ← AA (onaltılı)
LD A,00	(A) ← 00 (onaltılı)
LD HL,2800	(HL) ← 2800 (onaltılı)
LD A,(HL)	(A) ← AA (onaltılı)

3.7 (A) = EE (onaltılı); (B) = EE (onaltılı); (C) = FF (onaltılı). Böylece:
(2800) = (A) = EE
(2801) = (C) = FF bulunur.

3.8

Onaltılı kod	*Sembolik komutlar*
11 00 28	LD DE,2800
3E D6	LD A,D6
42	LD B,D
4B	LD C,E
21 FF 28	LD HL, 28FF
EB	EX DE,HL
06 EE	LD B,EE
0E FF	LD C,FF
21 01 28	LD HL,2801
78	LD A,B
32 00 28	LD (2800),A
71	LD (HL),C

4.1
+ 63 = 0 0111111
+112 = 0 1110000
+ 37 = 0 0100101
+111 = 0 1101111 , böylece -111 = 1 0010001
+ 27 = 0 0011011 , böylece - 27 = 1 1100101
+ 78 = 0 1001110 , böylece - 78 = 1 0110010

4.2
0 1011011 = + 91
0 0010111 = + 23
0 1111111 = + 127
1 1110111 = -(0 0001001) = - 9
1 0000101 = -(0 1111011) = -123
1 1010101 = -(0 0101011) = - 43

4.3
65 = 0 1000001 94 = 0 1011110
+24 = +0 0011000 -36 = -0 0100100
89 = 0 1011001 58 = 0 0111010

+28 = 0 0011100 } - 28 = 1 1100100
- 28 = 1 1100100 } +81 = +0 1010001
81 = 0 1010001 } +53 = 0 0110101

+53 = 0 0110101 } - 53 = 1 1001011
- 53 = 1 1001011 } - 23 = - 0 0010111
23 = 0 0010111 } - 76 = 1 0110100

+63 = 0 0111111 } -63 = 1 1000001
- 63 = 1 1000001 } +(-56) = +1 1001000
+56 = 0 0111000 } -119 = 1 0001001
- 56 = 1 1001000 }

+68 = 0 1000100 } - 68 = 1 0111100
- 68 = 1 0111100 } -(-84) = - 1 0101100
+84 = 0 1010100 } +16 = 0 0010000
-84 = 1 0101100 }

4.4

0110 0111 = 67
+0010 0100 = +24
Normal ikili toplam = 1000 1011 EL = 0, YE = 0
Bu nedenle düzeltme = +0000 0110
Düzeltilmiş BCD toplamı = 1001 0001 = 91

1000 1001 = 89
- 000 1000 = +08
Normal ikili toplam = 1001 0001 EL = 0, YE = 1
Bu nedenle düzeltme = +0000 0110
Düzeltilmiş BCD toplamı = 1001 0111 = 97

0111 0100 = 74
-0100 1000 = -48
Normal ikili fark = 0010 1100 EL = 0, YE = 1
Bu nedenle düzeltme = +1111 1010
Düzeltilmiş BCD farkı = 0010 0110 = 26

1001 1000 = 98
-0011 1001 = -39
Normal ikili fark = 0010 1100 EL = 0, YE = 1
Bu nedenle düzeltme = +1111 1010
Düzeltilmiş BCD farkı = 0101 1001 = 59

4.5

Sembolik komutlar	*İşlem*
LD A,27	(A) ← 27 (onaltılı)
ADD A,56	(A) ← 7D (onaltılı)
DAA	(A) ← 83 (onaltılı)
SUB 38	(A) ← 4B (onaltılı)
DAA	(A) ← 45 (onaltılı)

4.6

Sembolik komutlar	*İşlem*
LD B,26	(A) ← 26 (onaltılı)
LD C,69	(A) ← 69 (onaltılı)
LD D,56	(A) ← 56 (onaltılı)
LD A,B	(A) ← 26 (onaltılı)
ADD A,C	(A) ← 8F (onaltılı)
DAA	(A) ← 95 (onaltılı)
SUB D	(A) ← 3F (onaltılı)
DAA	(A) ← 39 (onaltılı)

4.7

A, B	A VE B	A VEYA B	A ÖZEL VEYA B
A = 1010 0111 B = 1011 1000	1010 0000	1011 1111	0001 1111
A = 1110 0010 B = 1101 1100	1100 0000	1111 1110	0011 1110
A = 0011 1100 B = 0111 0011	0011 0000	0111 1111	0100 1111
A = 0101 1010 B = 0110 1011	0100 1010	0111 1011	0011 0001

4.8

Sembolik komutlar	*İşlem*		
LD A,6F	(A) ← 0110 1111	CY,P	etkilenmez
LD B,A4	(B) ← 1010 0100	CY,P	etkilenmez
AND B	(A) ← 0010 0100	CY ← 0, P ← 1	
OR 44	(A) ← 0110 0100	CY ← 0, P ← 0	
RLCA	(A) ← 1100 1000	CY ← 0, P etkilenmez	
RLCA	(A) ← 1001 0000	CY ← 1, P etkilenmez	

5.1

Adres (onaltılı)	*Onaltılı kod*	*Sembolik komutlar*
2000	32 E7	DONGU :LD A,E7
2002	3C	INC A
2003	C3 00 20	JP DONGU
2006	–	–

5.2 Program, B'nin değeri sıfır olana kadar aşağıdaki komutlar üzerinde döngülenecek ve daha sonra yürütüme normal sırasında devam edecektir:

```
SIFIRDGL:  DEC  B
           JP NZ, SIFIRDGL
```

5.3

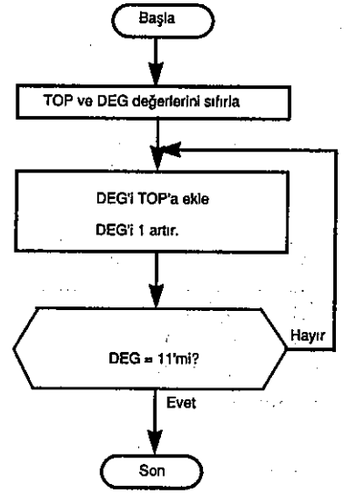
Adres	*Onaltılı kod*	*Sembolik komutlar*
2020	06 FF	LD B,FF
2022	05	SIFIRDGL:DEC B
2023	C2 22 20	JP NZ,SIFIRDGL
2026	---	---

5.4 Program:

```
DONGU : INC A
        JP  NC,DONGU
```

komutları üzerinde A, önce FF'e kadar artırılana, sonra da 00'a taşana kadar döngülenecektir. Böylece elde biti 1 olacak ve program normal işleyişine normal sırasında devam edecektir. Dolayısıyla 256 kere döngüleyecektir.

5.5 B, devreden (ve sonuçtaki TOP) toplamı, C de eklenecek bir sonraki değeri (DEG) tutsun. Akış diyagramı Şekil B'de, program kodu da aşağıda gösterilmiştir:

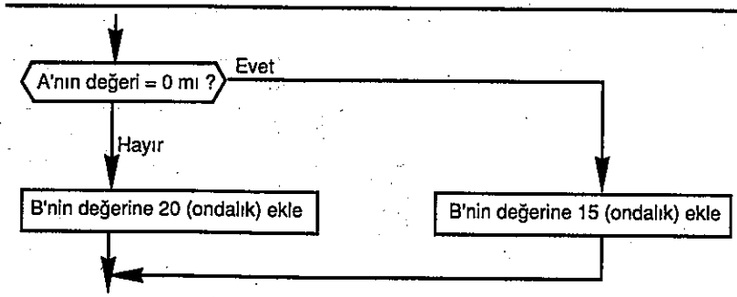


Şekil B

Sembolik komutlar	*İşlem*
LD B,00	TOP ve DEG değerlerini sıfırla
LD C,00	
DONGU: LD A, B	DEG'i TOP'a ekle
ADD C	
LD B, A	
INC C	DEG'i 1 artır
LD A, 0B	DEG = 11 (ondalık) mı ?
CP C	
JP NZ, DONGU	Hayır: geriye döngü yap
HALT	Evet : son (TOP B'de)

5.6 Akış diyagramı Şekil C'de, program kodu da aşağıda gösterilmiştir :

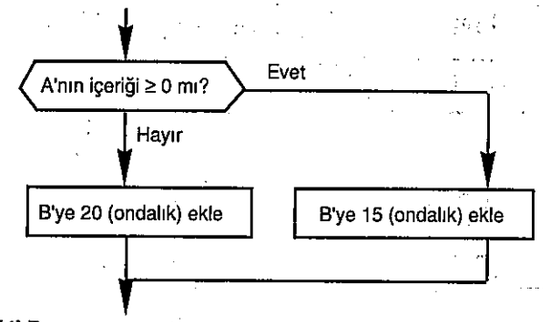
Sembolik komutlar		*İşlem*
	.	
	.	
	.	
	CP 00	A'nin değeri = 0 mı ?
	JP Z, ONBES	
	ADD A, 14	Hayır : 20 (ondalık) ekle
	JP DEVM	Devam etmek için dallan
ONBES:	ADD A, OF	Evet : 15 (ondalık) ekle
DEVM:	-	Devam



Şekil C

5.7 Akış diyagramı Şekil D'de, program kodu da aşağıda gösterilmiştir :

Sembolik komutlar		*İşlem*
	.	
	.	
	.	
	CP 00	A'nın içeriği sıfıra eşit veya sıfırdan küçük mü ?
	JP Z, ONBES	
	JP M, ONBES	
	ADD A, 14	Hayır : 20 (ondalık) ekle
	JP DEVM	
ONBES:	ADD A, OF	Evet : 15 (ondalık) ekle
DEVM:	-	
	.	
	.	
	.	



Şekil D

5.8 Sayı aralığı, bellek adresi 2800'da (onaltılı), hesaplanan toplam da bellek adresi 2801'de (onaltılı) tutuluyor olsun. Toplamın tek bir bayta gereksinim duyduğu varsayılmıştır. Altyordamın adı TOP olsun. Program kodu aşağıdaki gibidir :

Sembolik komutlar			*İşlem*
	LD	HL, 2800	Bellek işaretçisini 2800'e (onaltılı) kur
	LD	(HL), aralık+1	Sayı aralığı + 1'i belleğe yükle
	CALL	TOP	Altyordamı çağır
	LD	B, (HL)	Sayıların toplamı B'de
	.		
	.		
	.		
TOP:	LD	B, 00	TOP ve DEG'i
	LD	C, 00	sıfırla
DONGU:	LD	A, 00	DEG'İ TOP'a ekle
	ADD	C	
	LD	B, A	
	INC	C	DEG'i 1 artır
	LD	A, (HL)	DEG = sayı aralığı mı ?
	CP	C	
	JP	NZ, DONGU	Hayır : geriye döngü yap
	INC	HL	Evet : TOP'u bellek blokuna
			sakla ve geriye dön
	LD	(HL), B	
	RET		

5.9

	Sembolik komutlar		*İşlem*
	LD	C, XX	İstenen gecikme parametresini C'ye yükle
GECIK:	LD	A, C	C'nin değeri 0 mı ?
	CP	00	
	JP	Z, ZAMAN	
	LD	A, 64	Hayır : 100 kere döngü yap
DONGU:	DEC	A	
	JP	NZ, DONGU	
	DEC	C	Gecikme parametresini azalt
	JP	GECIK	ve geriye döngü yap
ZAMAN:			Evet : zaman gecikmesi hesaplandı

5.10 XX = 00 (onaltılı) olduğunda en az bilgisayar gecikmesi
= 3 komut veya 12 mikrosaniye
XX = FF (onaltılı) olduğunda en çok bilgisayar gecikmesi
= 256 x (6+2x100) +3
= 256 x 206 +3
= 52,739 komut
= 210.956 µsn.

5.11

	Sembolik komutlar		*İşlem*
	LD	D, aralık+1	Sayı aralığı + 1'i D'ye yükle
	PUSH	DE	ve yığında sakla
	CALL	TOP	
	POP	BC	Sayıların toplamı B'de
	.		
	.		
	.		
TOP:	POP	HL	Dönüş adresini HL'ye sakla
	LD	B, 00	TOP ve DEG'i
	LD	C, 00	sıfırla
DONGU:	LD	A, 00	DEG'i TOP'a ekle
	ADD	C	
	LD	B, A	
	INC	C	DEG'i 1 artır

	POP	AF	DEG = sayı aralığı mı ?
	CP	C	
	JP	NZ, DONGU	Hayır : geriye döngü yap
	PUSH	BC	Evet : TOP'u yığına sakla
	PUSH	HL	Dönüş adresini tekrar yığına sakla
	RET		Dönüş

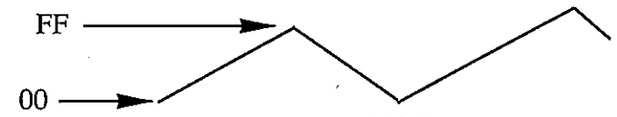
6.1

	Sembolik komutlar			*İşlem*
(a)	DONGU:	IN	A, (01)	01 portundan anahtarların durumunu gir
		XOR	FF	A'yı tümle
		OUT	(02),A	A'yı 02 portuna çık
		JP	DONGU	Döngü yap
(b)	DONGU:	IN	A, (01)	Anahtarların durumunu gir
		XOR	FF	A'yı tümle
		CP	FF	Tümlenmiş durum = FF (onaltılı) mı?
		JP	NZ, DONGU	Nayır: döngü yap
		HALT		Evet: son

6.2

	Sembolik komutlar		*İşlem*
TEKRAR:	LD	A, 00	Sayımı 00'a sıfırla
YUK:	OUT	02), A	Sayımı çık
	INC	A	Sayımı artır
	CP	00	Sayım 00'a eşit değilse
	JP	NZ, YUK	döngü yap
	LD	A, FF	Sayımı FF'e hazırla
ASAGI:	OUT	(02), A	Sayımı çık
	DEC	A	Sayımı azalt
	CP	FF	Sayım FF'e eşit değilse
	JP	NZ,ASAGI	döngü yap
	JP	TEKRAR	Tekrarla

Şekil E'de görülen dalga biçimine testeredişi veya üçgen dalga denir.



Şekil E

6.3

Sembolik komutlar		*İşlem*
	LD HL, 2800	Tablo değerlerini
	LD (HL), 15	2800 (onaltılı) adresinden
	INC HL	başlayarak
	LD (HL), 2B	bellekte sakla
	INC HL	
	LD (HL), 3F	
	INC HL	
	LD (HL), 49	
	INC HL	
	LD (HL), 54	
	INC HL	
	LD (HL), C2	
	INC HL	
	LD (HL), 00	
	LD HL, 2800	HL'yi tablonun ilk kaydına işaretle
TEKRAR:	LD A, (HL)	Tablodan bir sonraki durumu oku
	OUT (02), A	Bir sonraki durumu 02 portuna çık
	CALL GECIK	Gecikme kadar bekle
	INC HL	Tablo işaretçisini 1 artır
	LD A, L	Tüm durumların çıkışı
	CP 07	yapıldı mı?
	JP NZ, TEKRAR	Hayır: tekrarla
	HALT	Evet: son
GECIK:	-	Gecikme alyordamı
	-	
	.	
	.	
	.	
	RET	

6.4

Sembolik komutlar		*İşlem*
	LD HL, 2800	Tablonun içeriği bellekte
	LD (HL), 15	2800'den adresinden
	INC HL	başlayan bitişik
	LD (HL), 01	konumlarında saklıdır
	INC HL	
	LD (HL), 2B	

	INC HL	
	LD (HL), 14	
	INC HL	
	LD (HL), 3F	
	INC HL	
	LD (HL), C8	
	INC HL	
	.	
	.	
	.	
	LD (HL), 00	
	LD HL, 2800	Tablo işaretçisini Şekil 6.8'deki
TEKRAR:	-	gibi hazırla
	.	
	.	
	.	
	RET	

6.5

Sembolik komutlar		*İşlem*
	LD HL, 2800	Tablonun içeriği
	LD (HL), 3C	2800 (onaltılı) adresinden
	INC HL	başlayan bitişik bellek
	LD (HL), 2C	konumlarında saklıdır
	.	
	.	
	.	
	LD (HL), 00	
	LD HL, 2800	Tablo işaretçisini Şekil 6.9'daki
TEKRAR:	-	gibi hazırla
	.	
	.	
	.	
	HALT	

6.6

Sembolik komutlar			*İşlem*
BKDAVK:	IN	A, (01)	Basamağı gir
	RLCA		DAV hattı 1'e kurulu mu?
	JP	NC, BKDAVK	Hayır : döngü yap
	RRCA		Evet : basamağı ve DACC'ı

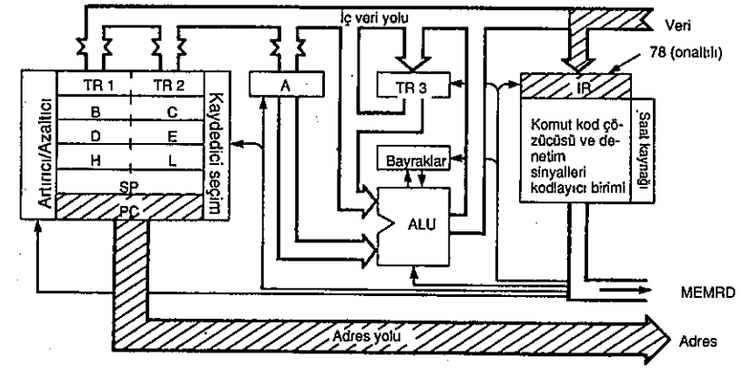
```
                                  eski durumuna getir
        OUT     (02), A           Basamağı ve DACC'ı LED'lere
                                  çık
BKDAVS: IN      A,(01)            Basamağı gir
        RLCA                      DAV hattı sıfırlanmış mı ?

        JP      C, BKDAVS         Hayır : döngü yap
        RRCA                      Evet : basamağı ve DACC'ı
                                  eski durumuna getir
        OUT     (02), A           Basamağı ve DACC'ı
                                  LED'lere çık
```

6.7 B, çıkacak değeri; C ise çıkan bit sayısını tutsun. Çıkış hattının, 02 portunun 0. biti olduğunu varsayın.

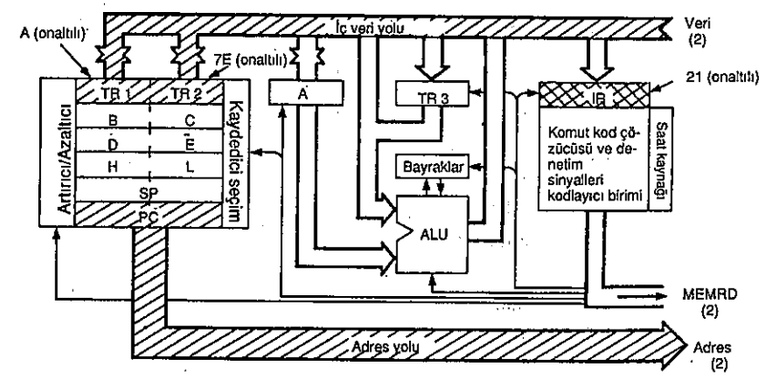
```
Sembolik komutlar       İşlem

        LD      B, değer        Çıkacak değeri
                                B'ye yükle
BIRBIT: LD      C, 00           Bit sayımı sıfırla
        LD      A, B            İstenmeyen bitleri maskele
        AND     01
        OUT     (02), A         Biti 02 portuna çık
        CALL    BIRSAN          1 saniye gecikme için bekle
        LD      A, B            Bir sonraki biti e.k.d.
        RRCA                    basmağa kaydır
        LD      B, A
        INC     C               Bit sayımını artır
        LD      A, 08           Tüm bitlerin çıkışı
        CP      C               yapıldı mı ?
        JP      NZ, BIRBIT      Hayır : bir başka bit çık
BIRSAN: HALT                    Evet : son
        -                       1 saniye gecikme
        .                       altyordamı
        .
        .
        RET
```


7.1 Şekil F'ye bakınız.

a) LD, A,B = 78 (onaltılı) getirme

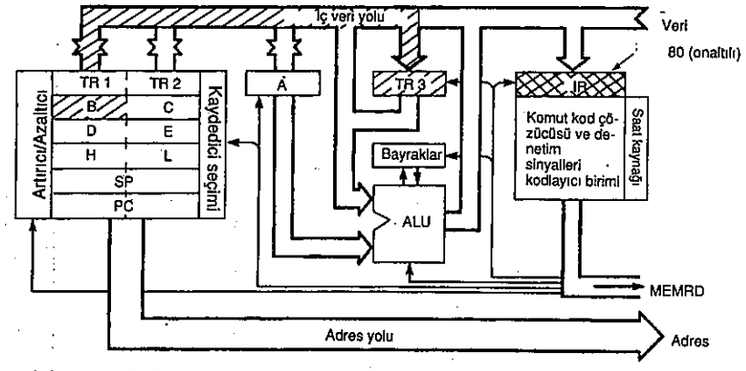


(b) LD HL, 2A7E = 21 7E 2A (onaltılı) getirme

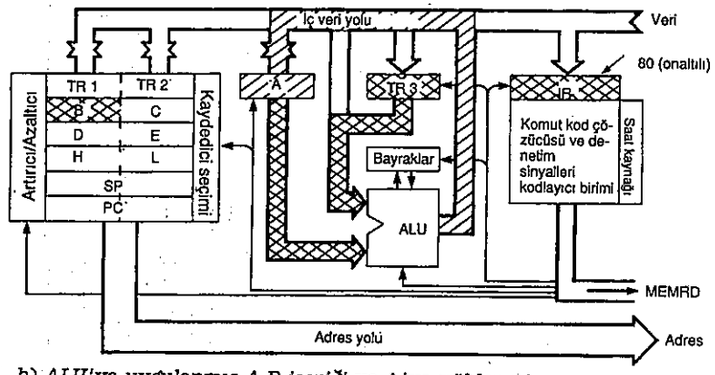


Şekil F

7.2 Şekil G'ye bakınız.



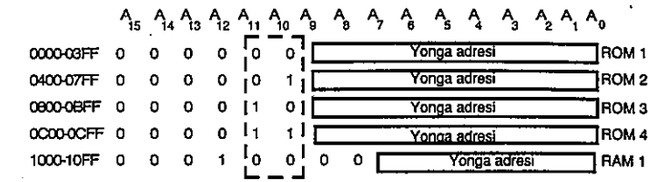
a) TR3'e yüklenmiş B içeriği

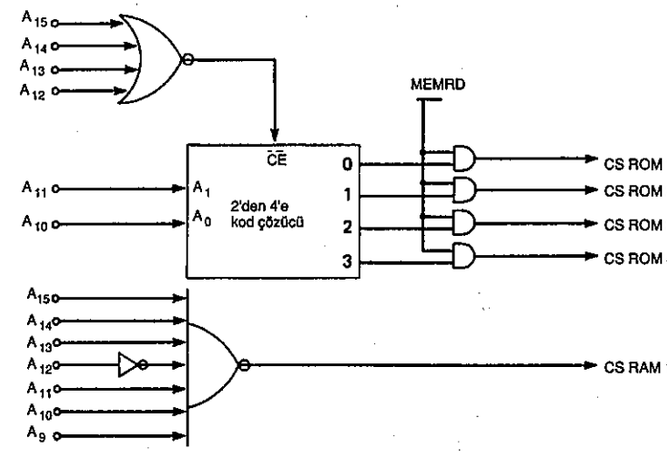


b) ALU'ya uygulanmış A-B içeriği ve A'ya yüklenmiş sonuç

Şekil G

7.9 Şekil H'ye bakınız. 1K = 10 bit, 4K = 12 bit olduğundan, ROM adres aralığı 0000-OFFF'tir. Ayrıca, 256 = 8 bit olduğundan, RAM adres aralığı 1000 - 10FF'tir.





Şekil H

İndeks

TERİMLER SÖZLÜĞÜ

Adres yolu (Address Bus): Üzerinden adresi 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.

Altyordam (Subroutine): bir programın içinde yer alan ve programda tekrarlanan belirli işlemleri yerine getirdikten sonra denitimi 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 elektroni devre.

Arabirim (Interface): Birimlerarası bağlantıyı ve bu bağlantıdaki uyumu sağlayan birim.

ASCII (American Standard Code for Information Interchange: Bilgi Değişimi için Amerikan Standart Kodu).

Assembly dili (Assembly language): Genellikle her bir komutu bir makine koduna karşılık gelen donanım-bağımlı sembolik dil.

Baud: Bid/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 - Auxilitary Carry (Yardımcı Elde)
PO - Parity Odd (Tek Eşlik)
PE - Parity Even (Çift Eşlik)
P - Plus (Pizitf)
M - Minus (Negatif)

BCD (Binary Coded Decimal: İkili kodlanmış ondalık): Ondalık tabandaki sayılaran, 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-ışınlı tüp): Elektron tabancasından çıkan elektron hüzmesinin, 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 kulanılabilecek biçimde (örn. 6800'den 8080'e) çeviren çevirici program.

Çevrebirimi (Peripheral): Bilgisayarın kendi parçası olmayan, ancak bilgisayara arabirinm 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 alışverişini yöneten donanım.

DIL paketi (Daul-In Line package): Çok kısa aralıklarla iki sıra bağlantı yapılabilmesini sağlayan entegre devre paketi.

Dinamik RAM (Dynamic RAM): İçerdiği bilgileri zaman içindekaybedebilen 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 tanışan program.

Emülasyon (Emulation): Bir bilgisayar sisteminin, bar başka bilgisayar sistemi için hazırlanmış veri ve programları kullanabilmesine ve aynı sonuçları alabilmesine olanak tanışan 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 çeft olmasına göre 0 veya 1 değeri alarak 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 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.

İşkodu (Opcode): Bkz: İşlem kodu

İşlem kodu (Operation code): Makinenin belli bir işlemi yapmasını sağlayan ve bir veya birkaç bayt uzulukta olabilen makine düzeymi 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 üreteci (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, çevirilmesi 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 sisltem.

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 bellekten 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 devrelirini içinde bulunduran bölümü.

MODEM (MOdulator+DEModulator): Asenkron seri haberleşmede, bilgisayarın 0 ve +5 V düzeyleri şeklindeki mantıksal sinyallerini, iletim hatlarından iletibilecek

biçimde değişen frekanstakı alternatif sinyallere; alışta da, bu alternatif sinyalleri tekrar mantıksal 0 ve 1 sinyallerine dönüştüren cihaz.

MOS: Metal-oksit yarı iletken anlanmında 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 eşzamanlı biçimde çalışabilmesini sağlamak üzere bir üreteçten üretilen periyodik sinyaller.

Saat üreteci (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 baytlıktır.

Sembolik adres (Symbolic address): Bir program içinde, bir sözcüğü, bir işlevi ya da bir bilgiyi, ilginin 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 ilişsisinde 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 sıyadı 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 emetörlü, 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 bağla 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 İngiliz'lerin kullandığı bir kısaltma. Eş anlamı. CRT-Display.

Veri yolu (Data Bus): Verilerin, MİB ile bellek ya da çevrebirimleri 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 (Işık, gerilim) uygulandığında, içerdiği valans elektronların yörüngelerin- den 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-- basışlı, mürekkep-püskürtmeli, tambur, vb. adlar alırlar.

Yazılım (Software): Bilgisayarın kaynaklarını kullanmak için gereken programlar, yordamlar, yardımcı programlar ve kütüphaneler gibi donanımın dışında kalan bölümü.

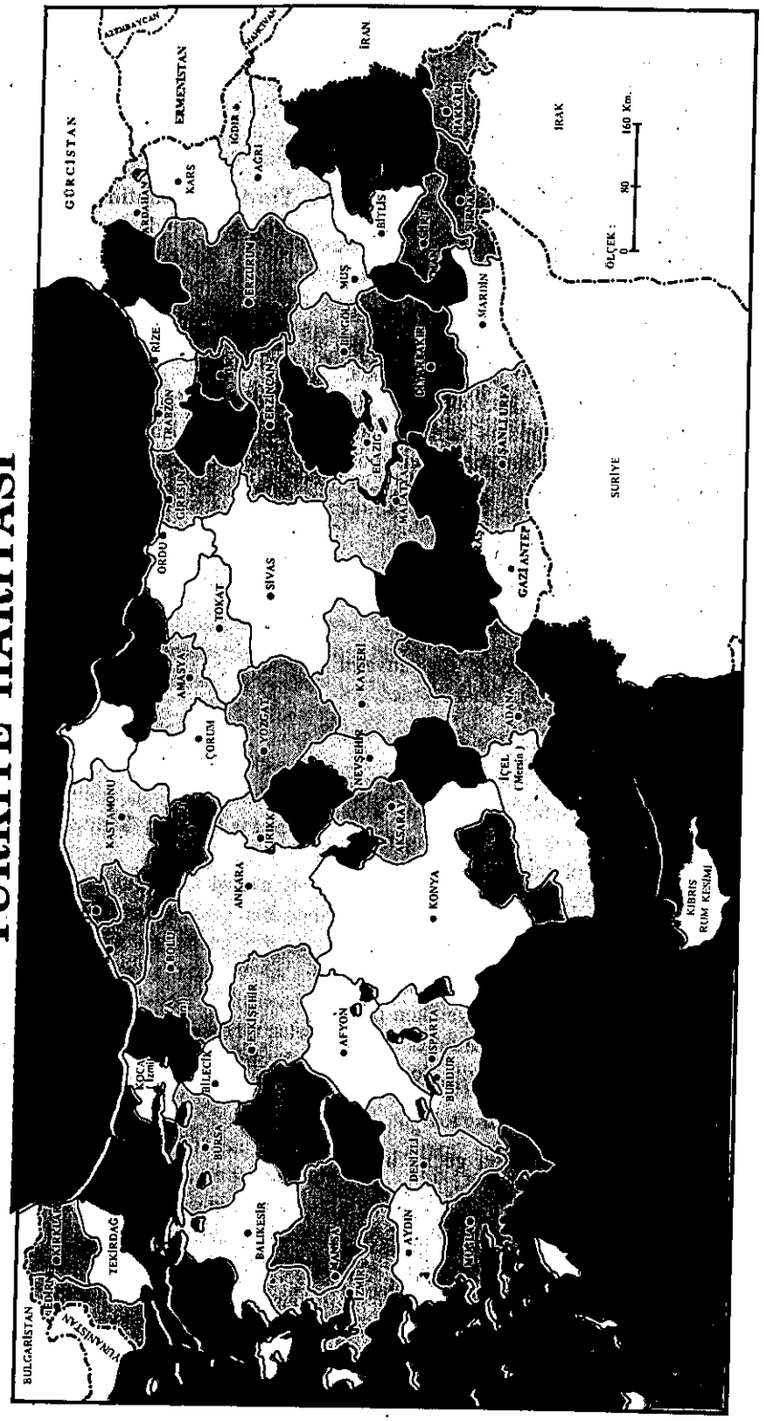
Yığın (Stack): Verileri geçici bir süreyle -örneğin, çıkışta geriye almak üzere bir altyordama girerken- saklandığı ve bilgiler, belirli bir sırada giriş-çıkış yapacak şekilde düzenlenmiş bellek alanı.

Yonga (Chip): Milimetrik boyutlarda yüzbinlerce elektronik devre elemanından omuşmuş 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 (Sever-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 arkadaş!
Öğren, öğret hakkı halka, gürle coş;
Durma durma koş.

Şanlı yurdum, her bucağın şanla dolsun;
Yurdum, seni yüceltmeye antlar olsun.

İsmail Hikmet ERTAYLAN