YILDIZ TEKNİK ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

Z80 Mikroişlemcisi ile Sistem Kontrol

Beyhan Yıldız

Yüksek Lisans Tezi

YILDIZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Z80 MİKROİŞLEMCİSİ İLE SİSTEM KONTROLÜ

YÜKSEK LİSANS TEZİ

Elektrik Müh. Beyhan Yıldız

İSTANBUL-1988

ÖNSÖZ

Günümüzde, mühendisliğin her dalında,endüstrede,tıpta ve evlerde insan'a hizmet veren gelişmiş bilgisayarlara gelinceye dek, bilgisayar teknolojisinin geçirdiği gelişmeler üç ana kuşağa ayrılabilir.Elektronik öncesinde hesap makinalarının gelişmesiyle başlayan bu kuşağın bilinen en eski örneği,1642 yılında Pascal'ın gerçekleştirdiği yalnız toplama ve çıkarma yapan hesap makinasıdır.1822 yılında İngiliz Charles Babbage, farklara dayanarak çalışan bir hesap makinası önerdi.Bu öneri, 1937 yılında IBM firması tarafından gerçeklendi.

1940'larda, elektronik kuşağında,ilk elektronik bilgisayar ENIAC(electronic numerical integrator and calculator) onluk düzenle çalışıyordu.1947 yılında Maurice Wilkes, ilk kez bellekteki programla çalışan bilgisayarı gerçekleştirdi.1955 ve sonrasında, transistörler ve tüm devrelerle bilgisayarın gelişmesinde büyük bir hıza tanık olunmuştur.

1970'lerde, mikroişlemci kuşağında,ilerleyen yarı iletken teknolojisiyle,merkezi işlem birimi tek bir tümdevreye sığdırıldı ve ilk mikroişlemci I4004 1971 yılında Intel firması tarafından piyasaya sürüldü.1980'lerde üç mikroişlemci I 8085, M6802 ve Z 80 endüstri standartı olarak piyasaya sürüldü.Şimdilerde 32 ve 64 bitlik mikroişlemciler üretilmeye başlanmıştır.

Bu kadar hızla gelişen bir konuda, bana tez hazırlama imkanı veren ve beni teşvik eden değerli hocam sayın Doç.Dr.Halit PASTACI'ya teşekkürü borç bilirim.

Ayrıca, beni yönlendiren değerli hocam, sayın Yard.Doç.Dr. Galip CANSEVER'e teşekkür etmek isterim.

Mikroişlemciler konusunda, bilgilerini bana bıkmadan aktaran değerli arkadaşım,araştırma görevlisi sayın Nizamettin AYDIN'a içten teşekkür ederim.

Beyhan YILDIZ

1988-Haziran

İÇİNDEKİLER

ÖZET

İlk bölümde mikroişlemciler hakkında genel bilgi verilmiştir.Kanımca mikroişlemci kavramını yakalamak,belirli bir işlemciyianlamak için bir anahtardır. Bu amaçla, Basit bir mikrobilgisayarın çalışması, bellek adresleme, giriş-çıkış işlemleri, adres çözme ve birimlerin iç yapıları açıklanmıştır.

İkinci bölümde, Z 80 CPU ve çevre birimlerinden CTC ile PIO iç yapıları ve programlanmaları açıklanmıştır.Ve bir uygulama örneği olarak, Z80 CPU komutları yardımıyla yazılım kontrollu bir saat tasarlanmıştır.

SUMMARY

The subject of this study is about in general microprocessors and especially the Z80.In the first chapter, it is intended as an introduction to microprocessors and as a companion to basic learning efforts employing specific devices, for which it will provide the enhancement of generalized coceptual framework.For this reason, the operation of a simple microcomputer,memory addressing, address decoding.input-output operations and internal architecture was discussed.

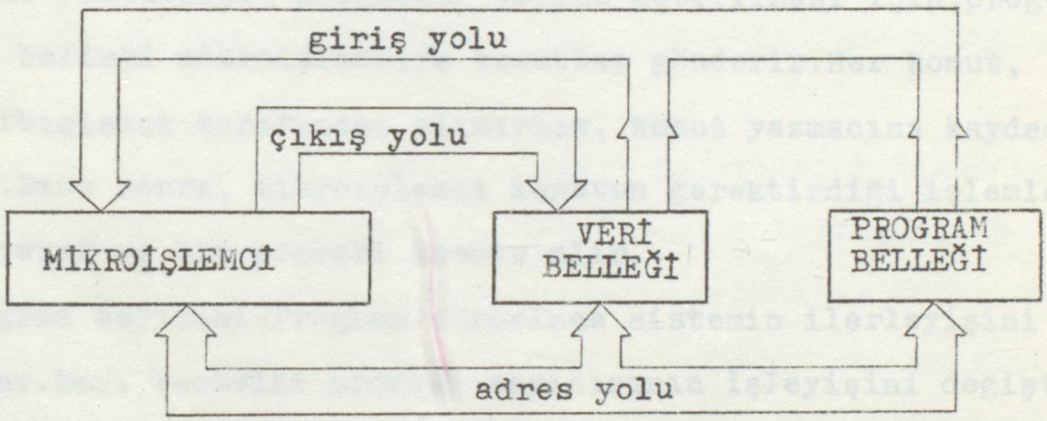
The second chapter deals with Z80 CPU and peripheral devices (Z80 CTC and Z80PIO).İnternal architecture and programming of CTC and PIO are explained.Then, as an example of application a clock using software with Z80 CPU instructions was designed.

BİRİNCİ BÖLÜM

I-GENEL MİKROİŞLEMCİLER

I-1 BASİT BİR BİLGİSAYARIN ÇALIŞMASI

Bir mikro bilgisayar, bilgi-işlem birimi oluşturmak amacıyla biraraya getirilmiş dijital devre elemanlarından oluşmuştur.Bu birim, üç temel elemandan meydana gelmiştir.(şekil I-1)



ŞekilI-1 Basit bir mikro bilgisayarın blok diagramı

Sistemin yürüteceği program, program belleğinde,o programda kullanılacak sayılar da veri belleğinde depolanır.Mikroişlemci de programın belirlediği sıraya göre işlem yapar.Mikroişlemci, sistemin odaksal bir elemanıdır, ve diğerlerinden yalıtılmış değildir.Program, mikroişlemcinin yapacağı çeşitli işlem basamakların göstermek için kodlanmış binary karakter takımı olarak program belleğinde depolanır.Mikroişlemci, bu kodları çözebilecek ve program basamaklarını yerine getirebilecek devrelere sahiptir.

Mikroişlemci içinde binary sayıları depolamak için kullanılan yazmaçlar(registers) vardır.Bunlar:
akümülatör:Tüm verilerin işlenmesinde odaksal elemandır.Sayılar akümülatöre eklenir veya akümülatörden çıkarılır. Kaydırma gibi belli işlemler, yalnızca akümülatör tarafından yapılabilir.
index yazmaç:Özellikle önemli veri adreslerini oluşturmak ve depolamak için kullanılır.
komut yazmacı:Bir programın yerine getirilmesi için,program belleği mikroişlemciye komutlar gönderir.Her komut, mikroişlemci tarafından alınırken, komut yazmacına kaydedilir.Daha sonra, mikroişlemci komutun gerektirdiği işlemleri yapar ve bir sonraki komutu alır.
program sayıcısı:Program süresince sistemin ilerleyişini izler.Bazı komutlar program sayıcısının işleyişini değiştirebilirler.

Mikrobilgisayar sisteminin önemli bir özelliği, üç temel elemanın; mikroişlemci, program belleği ve veri belleğinin bir "yol" vasıtası ile bağlantılarıdır.YOL, paralel binary bilginin iletilebildiği bağlantı grubudur. Paralel bilginin yalnızca bir bölümü (genellikle bayt denir.) aynı anda bir yolda olabilir.Örneğin, şekil I-1'de giriş yoluyla hem program belleğinin, hem de veri belleğinin aynı anda mikroişlemciye bilgi göndermesine izin verilmeyebilir.Sistem elemanlarından biri yol kontrolünü üzerine almalıdır ki, bu görev de mikroişlemciye düşer.Yol kontrolü, mikroişlemci tarafından üretilen özel işaretler

vasıtası ile yapılır ve çeşitli kontrol hatları ile gönderilir.Bazan bu kontrol işaretlerine, kontrol yol'u da denir.Karışıklığa yol açmamak için, kontrol yolu şekil I-1 de gösterilmemiştir.ŞekilI-1'deki sistemde giriş ve çıkış yolu 8 bitlik, adres yolu da 16 bitliktir.

I-1.2 TEMEL KOMUT PERYODU

CPU tarafından her komutun tam olarak işlenmesi bir mikroprogram oluşturur.Bu program, FETCH(gidip getirme) ve EXECUTE(yürütme) olmak üzere iki ana bölümden oluşur.

Getirme peryodu: Yerine getirilecek komut bellekte depolanmış olduğundan, mikroprogramın ilk bölümü komut baytını getirip, uygun yazmaçda tutmaktır.Komutun OPCODE(işlem kodu) parçası komut yazmacında tutulur.Bu, yerine getirilecek komut tipine uygun kontrol işaretlerini diğer devrelere gönderecek kontrol ünitesini"enable" durumuna getirir.Komut iki ya da daha fazla bayttan oluşuyorsa, 2.ve 3. baytlar da bellekten getirilmelidir.Tüm bu işlemler gidip getirme peryodu olarak anılır.

Yürütme peryodu: Bu aşamada CPU içinde tutulan komutun gereği yerine getirilir.

Komut, seçilen bir bellek bölgesi içeriğinin akümülatöre yüklenmesini istesin.Bu üç baytlık bir komut olacaktır.İlk bayt işlem kodunu, 2. ve3. baytlar da verinin bulunduğu bellek bölgesinin adresini içerecektir.

Komutun hex kodunun 3A ve akümülatöre yüklenecek değerin de 38H olduğunu varsayalım.Komutun bellekte yerleşimi aşağıdaki gibi olacaktır.

bellek adresi(H)	adreste bulunanlar	yorum
0000	3A	; komutun işlem kodu
0001	AA	
0002	00	; verinin bulunduğu yerin adresi
⋮	⋮	
00AA	38	; veri

Getirme peryodu sonunda CPU yazmaçlarının durumu şöyledir:

1-Komut yazmacı binary formda işlem kodunu tutuyor.(3AH)

2-Bellek adres yazmacı akümülatöre yüklenecek veri adresini tutuyor.(00AAH)

3-PC(program sayıcısı), bir sonra getirilecek komutu gösterir durumda hazır bekliyor.(0003H)

Getirme süreci tamamlandı.CPU, şimdi otomatik olarak yürütme sürecine girecektir.

1-Bellek adres yazmacı içeriği adres yoluna iletilir.(00AA)

2-00AA adresindeki veri veri yoluna konur. (38H)

3-Veri yolu ile gelen veri akümülatöre yerleştirilir.

CPU şimdi, eğer bir atlama komutu yoksa, 0003 adresinde bulunan komutu getirmeye başlayacaktır.

I-1-3 KOMUT TÜRLERİ

Mikroişlemcilerde kullanılan dört temel komut türü vardır:

1-Aritmetik ve lojik komutlar (toplama,çıkarma, VE,VEYA vb.)

2-Bellekle çalışan komutlar (veri veya program belleği)

3-Atlama (jump) komutları

4-Giriş ve çıkış komutları (Sistemin dış dünya ile bağlantısını sağlarlar.)

I-1-3.1 ARİTMETİK VE LOJİK KOMUTLAR

Bu komutlar ALU'yu kullanarak bir veri ile çalışırlar.İşlem(sözgelimi toplama) için iki "operand" gereklidir."Operand"ın birinin kaynağı akümülatör, diğerinin ki ise, belleğin herhangi bir bölgesi veya yazmaçlardan biridir.ALU'da bayrak(flag) denilen flip-flop'lar vardır. Bu bayraklar genellikle durum bayrakları (status flags) olarak anılır, bu durumları oluşturan binary kelimelere de durum kelimeleri denir.

Durum bayrakları işlemciden işlemciye göre değişir.En temel olanları:

-Elde bayrağı

-Yarı elde bayrağı

ELDE bayrağı, aritmetik işlemin elde'si varsa set edilir. Yarı-elde bayrağı, BCD işlemlerde en az ağırlıklı 4 bitten en çok ağırlıklı 4 bite elde varsa set edilir.Bu bayrağın asıl işlevi, "decimal adjust accumulator" komutunun yürütülmesinde ortaya çıkar.Ancak diğer amaçlar için de kulla-

nılabilir.Diğer bayraklar akümülatördeki sayıyı bağlı olarak, işaret, sıfır veya "parity" bayrağı olabilir.Bir de aritmetik işlemin sonucu olarak bir overflow (taşma)gösteren taşma bayrağı olabilir.Taşma ve elde bayrakları arasındaki fark önemlidir: Bilgisayarda sayıların gösteriliminde signed 2's complemet (işaretli 2'ye tümleyen) yöntemi kullanılır.Bu yöntemi kullanın mikroişlemciler için, n yazmaçdaki bit sayısı 8 olmak üzere; $2^7-1=127$ ile $-2^7=-128$ arasındaki sayılar gösterilebilir.Bu gösterimde, makina, en soldaki biti işaret biti (0 ise pozitif, 1 ise negatif) olarak algılayacağından, aynı işaretli sayılar toplanırken bu sınırlar aşıldığında işaret bitinin değeri değişeceğinden sonuç yanlış çıkacaktır.Bu yanlışlık, taşma bayrağı set edilerek bildirilecektir.İşaret bitine giren elde ile işaret bitinden çıkan elde aynı değilse, bunların bir exOR kapısından geçirilmeleri halinde çıkış 1 olacak, bu da taşma bayrağını set edecektir.

Aritmetik komutlara örnek olarak, toplama, çıkarma, sağa kaydırma(2'ye bölmek), sola kaydırma(2 ile çarpmak) ve lojik komutlara örnek olarak da, ve, veya, evrik alma, akümülatörü sıfırlama, artırma, azaltma ve karşılaştırma komutlarını verebiliriz.

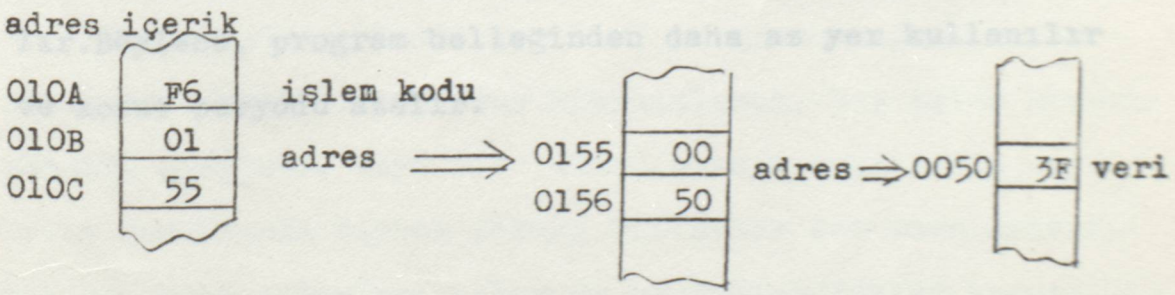
I-1-3.2 BELLEKLE ÇALIŞAN KOMUTLAR

Bellekte depolanmış bilgi ile çalışan komutlar "memory reference instructions" olarak anılırlar.Mikroişlemcinin bellekte depolanmış belli bir bilgiye veya kelimeye ulaşma hızı ve kolaylığı, programın yürütülme hızını saptar. Bir komuttaki işlem kodu, mikroişlemcinin veri ile hangi işlemi yapacağını saptar.Verinin bellekten "nasıl" alınacağını adreslemenin türü belirler.En temel olanları şunlardır:

-İvedi adresleme(immediate mode):Bellekte işlem kodunu izleyen bayt içinde, o işlemde kullanılacak veri bulunur.

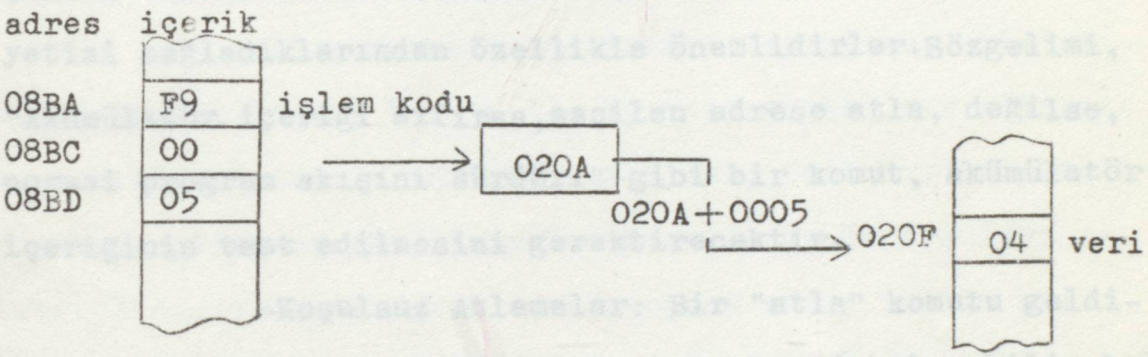
-Doğrudan adresleme(direct mode):Bellekte işlem kodunu izleyen bölgede, işlemde kullanılacak verinin bir veya iki baytlık adresi bulunur.

-Dolaylı adresleme (indirect mode):İşlem kodunu izleyen bellek bölgesi, işlemde kullanılacak verinin bulunduğu bölgenin adresinin adresidir.(şekil I-2)



Şekil I-2 Dolaylı adresleme

-Sıralı adresleme (indexed mode): Genellikle merkezi işlem birimlerinin içinde index yazmacı ya da index yazmaç olarak kullanılan genel amaçlı bir yazmaç vardır.İndex yazmaç bir adresle yüklenebilir ve verinin bulunabileceği bölge; işlem kodunu izleyen bölge içeriği, index yazmaç içeriğine eklenerek saptanır.(şekil I-3)



Şekil I-3 Sıralı adresleme

Belleği referans almayan komutlar da olabilir. Bu durumda, CPU yazmaçlarından birinin içeriği işlemde kullanılacağından,tek başına işlem kodu, komut olarak algılanır.İşlemde kullanılacak yazmaç adresi, işlem kodunun içinde belirtilir.Bu yöntem, "yazmaç adresleme" olarak da anılır.Böylece, program belleğinden daha az yer kullanılır ve komut peryodu azalır.

I-1-3.3 ATLAMA KOMUTLARI

Atlama komutları PC'nin programdaki normal akış değeri yerine, yeni bir değerle yüklenmesine neden olurlar.Bu, önemli bir noktadır çünki, PC'yi değiştirebilecek komutlar oldukça çoktur.Bu atlamalar koşullu ya da koşulsuz olabilirler.Koşullu atlamalar, programa karar verme yetisi sağladıklarından özellikle önemlidirler.Sözgelimi, "Akümülatör içeriği sıfırsa,seçilen adrese atla, değilse, normal program akışını sürdür." gibi bir komut, akümülatör içeriğinin test edilmesini gerektirecektir.

-Koşulsuz Atlamalar: Bir "atla" komutu geldiğinde ana programın yürütülmesi durur ve adreslenebilir bellek bölgesinin herhangi bir yerinde bulunabilen seçilmiş bir adresten yeniden başlar.Bunu sağlamak için de PC, atlanacak yeni adres ile yüklenmelidir.Tipik olarak "atla" komutları bir adresi doğrudan seçebilirler ya da bir CPU yazmacındaki adrese göre dallanırlar.

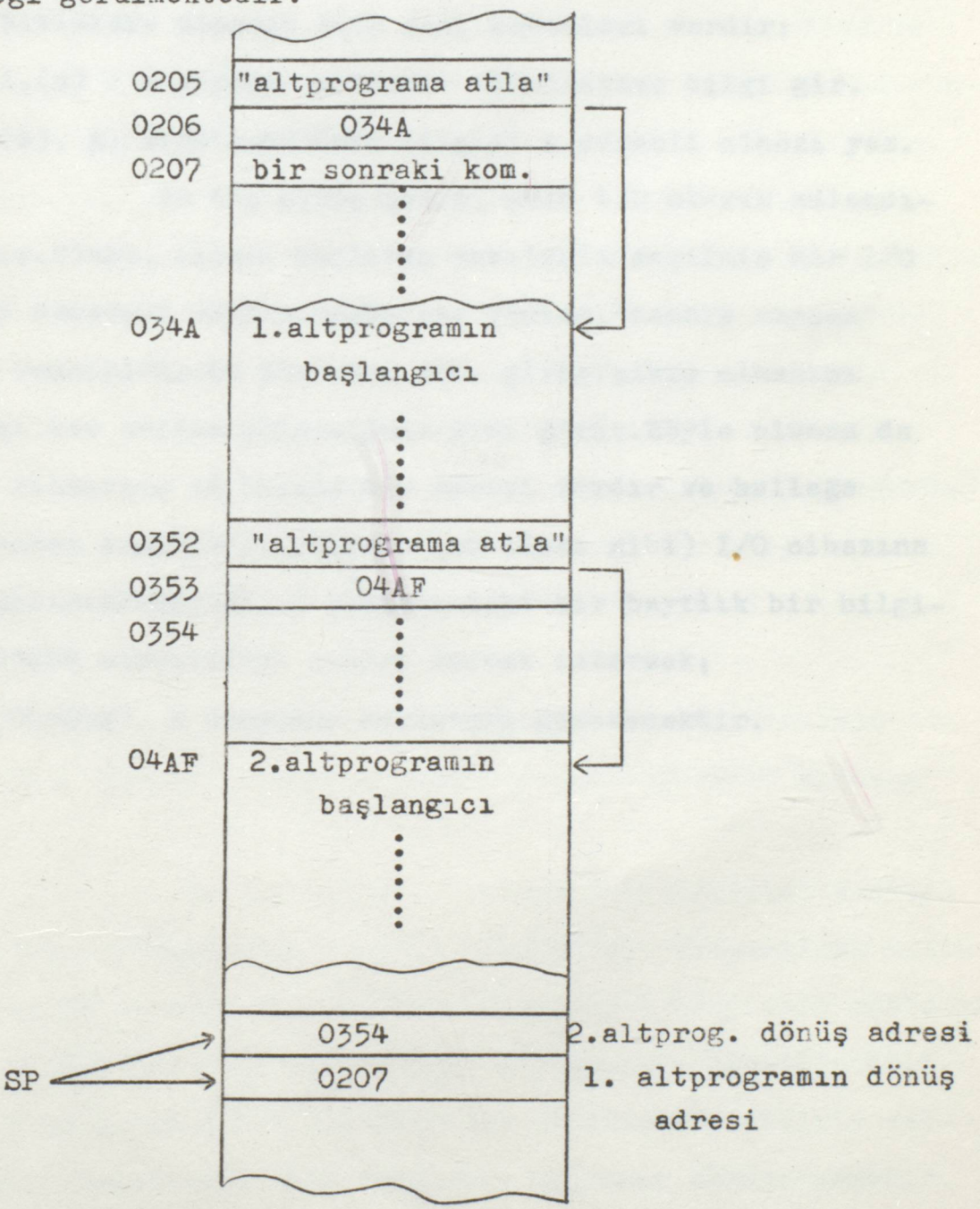
-Koşullu Atlamalar:Atlamadan önce bir koşulun yerine gelmiş olması gerekir; program akışına karar yetkisini veren bu koşuldur.Mikroişlemci, bir işlem sonucu negatif olduğunda veya bir "elde" ortaya çıktığında veya başka durumlarda bayrak yazmaç bitlerine set eden aritmetik ve lojik işlemleri izler.Bu bayraklar "atla" komutuna karar vermek için kullanılır.Başka bir yaklaşım da program sayıcısına, o andaki içeriği baz alınarak atlayacağı miktarı eklemektir.Atlama miktarı PC'ye bağlı olduğun-

dan bu yöntem, "bağıl adresleme" olarak da anılır.8 bitlik veri yolu ve 16 bitlik adres yolu olan mikroişlemcilerde bağıl adresleme kullanılarak atlama komutları üç bayttan iki bayta indirilebilir.Tek bir bayt ile koşullu atlamayı sağlayan özel bir komut tipi "skip" komutudur."xyz koşulu sağlanmışsa, daha sonra gelen n adet komutu atla" gibi. Böylece, programda n tane komut yürütülmeden atlanır.

-Altprogramlara Atlama:"Altprograma atla" komutu, normal atlama komutu ile aynıdır.Ancak, bir altprograma atlandığında PC içeriği kaybolmaz, özel bir bölgede depolanır.Altprograma atlamalar koşullu ya da koşulsuz olabilirler.Bir altprogram, ana program akışı içinde bir program kesitinin sıkça tekrarlandığı durumlarda kullanılır. Altprogram bellekte bir yere depolanır.Ana program akışı içinde, altprograma atlanıldığında işlemler bitince ana programa tekrar dönülür.Bu, PC içeriğini bellekte saklamayı ve tekrar almayı gerektirir.Bellekte bu geçici depolama için ayrılan bölge yığın(stack) olarak adlandırılır ve SP(stack pointer reg.) içindeki adres ile belirlenir.Her bilgi yığın'a depolanırken SP tarafından adreslenir, SP içeriği 1 azaltılır.Yığın'a depolama işlemi "push"ve yığından depolananları geri alma "pop" olarak adlandırılır.

Bir altprogram yürütülürken, bir başka alt program çağrısı gelirse, PC içeriği yeniden yığın'a itilir. Bu bir sorun yaratmayacaktır çünki; SP, ikinci altprogram tamamlandıktan sonra dönülecek ilk altprogram adresini tutmak üzere 1 azaltılacaktır.Her altprogram, "ana programa

dön" komutu ile sona ermelidir.Altprogramların iç içe geçmesinde sınırlayıcı faktör, bellekte yığın'a ayrılan bölgenin büyüklüğüdür.Şekil I-4'de iç içe geçmiş bir altprogram örneği görülmektedir.



Şekil I-4 İç içe geçmiş altprogramlar

I-1-3.4 GİRİŞ/ÇIKIŞ KOMUTLARI

Bir mikrokompüter sisteminde bir "keyboard" giriş ve "display" bir çıkış cihazıdır.Mikroişlemcinin bu birimlere ulaşmak için özel komutları vardır:

IN A,(n) : n adresli cihazdan akümülatöre bilgi gir.

OUT(n), A: Akümülatördeki bilgiyi n adresli cihazı yaz.

Bu tip giriş/çıkış, port I/O olarak adlandırılır.Çünki, işlemi başlatan yazılımla seçilmiş bir I/O port numarası vardır.Başka bir yöntem,"memory mapped" I/O tekniğidir.Bu yöntemde CPU, giriş/çıkış cihazını sanki bir bellek bölgesiymiş gibi görür.Böyle olunca da I/O cihazının 16 bitlik bir adresi vardır ve belleğe erişilen komutlarla(yükleme komutları gibi) I/O cihazına erişilir.Sözgelimi, B yazmacındaki bir baytlık bir bilgiyi C325H adresindeki cihaza yazmak istersek;

LD (0C325H), B komutunu kullanmak gerekecektir.

I-1-4 KOMUT NOTASYON YÖNTEMLERI

Bir bilgisayar donanım ile vardır ancak, yazılım ile çalışır.Bu da, CPU'nun bellekte depolanmış programı yürütmesi ile olur.Bir programı oluşturan komutlar ve bunların binary kodları, CPU donanımı ile ilintili olduğundan,her CPU'nun kendine özgü bir komut kümesi vardır.Sözgelimi,toplama işleminin kodu 70H ise, ALU bu sayıyı gördüğünde donanımı gereği toplama yapacaktır.Bir bilgisayarın programlanması CPU'nun kendine özgü dili ile yapılıyorsa, buna "makina dili"nde programlama denir.Makina dilinde program yazmak, programlama dillerinin en zorudur,çünki;

- Makina dili, CPU mimarisine bağlı olduğundan CPU türlerine göre farklıdır.Bu nedenle, bir bilgisayarın dilini öğrenmek için, o bilgisayarın mimarisini bilmek gerekir.

- Makina dilinde komutlar binary yazılmış sayılardır.Yüzlerce komutu binary olarak öğrenmek oldukça zordur.

Makina dilinin zorluğunu giderebilek amacıyla geliştirilmiş olan ilk dil "assembler" programlama dilidir.Bu dilde komutlar, kısaltılmış olarak harflerle gösterilir.Bu da, CPU mimarisine bağlı olduğundan, genelliştirildeğildir.Kolay kullanılabilir bir programlama dilinin bilgisayar yapısından olabildiğince, bağımsız olması gerekir. Bu amaçla geliştirilmiş programlama dillerine "üst düzey diller" denir.Fortran,basic, Pascal bu türdendir.

I-2 CPU İÇ DONANIMI

Bir CPU, aşağıdaki bağlaşık fonksiyonel birimlerden oluşmuştur:

-Yazmaçlar

-Aritmetik-lojik ünite

-Kontrol devresi

I-2-1 YAZMAÇLAR

CPU içinde geçici depolama birimleridir.Bazı yazmaçların(program sayıcısı ve komut yazmacı gibi)özel görevleri vardır.Diğerleri genel amaçlı kullanım içindir.

AKÜMÜLATÖR:Genellikle ALU'nun kullanacağı "operand"lardan birini depolar.Akümülatördeki veri ile işlem yapıldığında,sonuç yine akümülatöre depolanır.Genellikle, CPU içinde ara veri veya "operand"ları depolayabilen birkaç genel amaçlı yazmaç vardır.Böylece, ara sonuçların bellek ve akümülatör arasında gidip gelmesi sırasında karışıklık önlenerek, işlem hızı ve verimi artar.

PROGRAM SAYICISI(PC):CPU, bir programı yürütürken bir sonraki komutun nerde olduğunu bilmelidir.Bu adres bir program sayıcısında tutulur.İşlemci, bellekten her komut getirişinde sayıcıyı "1" artırır ki, sayıcı bir sonraki komutu gösterir durumda bulunsun.Bu nedenle, programcı komutlarını, en küçük adres ilk komutu içerecek şekilde ardışıl adreslere depolar.Bu ardışıl kuralın bozulduğu zamanlar,atlama komutlarının olduğu zamanlardır.

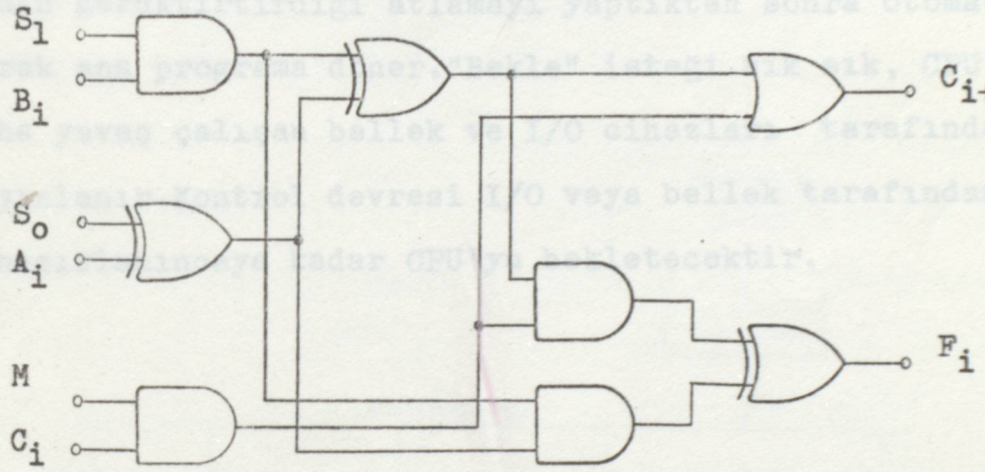
KOMUT YAZMACI VE KOD ÇÖZÜCÜ: Her bilgisayar, karakteristik olarak bir kelime uzunluğuna sahiptir.Kelime uzunluğu, bilgisayarın iç depolama elemanları ve yollarına bağlı olarak saptanır.Sögelimi, bir bilgisayar 8 bitlik bilgiyi depolayıp, iletebiliyorsa, kelime uzunluğu 8 bittir ve 8 bitlik paralel işlemci olarak anılır.Veriler ve komutlar, bellekte 8 bitlik binary sayı halinde depolanır veya 8 bitin integral çarpanları olarak(16,24 gibi) depolanır.8 bitlik işlemcilerin $2^8=256$ farklı komut kodu sözkonusu olabilir.

Mikroişlemci bir komutu iki farklı işlemle getirir: Önce, işlemci program sayıcısındaki adresi belleğe iletir.Bellek adreslenmiş baytı CPU'ya gönderir.CPU bu komut baytını, komut yazmacında tutar ve komutu buna göre yürütür.Komut yazmacındaki 8 bit çözülebilir ve çıkış hatlarından birin aktif hale getirmek için kullanılabilir.Her hat, belli bir komut kodunun yürütülmesi ile birleştirilen aktiflik seti gösterir.

ADRES YAZMAÇLARI:CPU veri almak için ulaşılacak bellek bölgesinin adresini tutan bir yazmaç ya da yazmaç çifti kullanır.Adres yazmacı programlanabilir ise, yani, programlayıcının yazmaç içeriğini değiştirebileceği komutlar varsa, program adres yazmacında adres üretilebilir.

I-2-2 ALU

Tüm aritmetik-lojik işlemlerin yapıldığı ünitedir.Bellekten veya girişlerden aldığı bilgiyi kullanarak işlem yapar.Akümülatördeki işlemden etkilenen bayrak bitleri vardır.Şekil I-5'de bir bitlik aritmetik-lojik ünite ve fonksiyon tablosu görülmektedir.

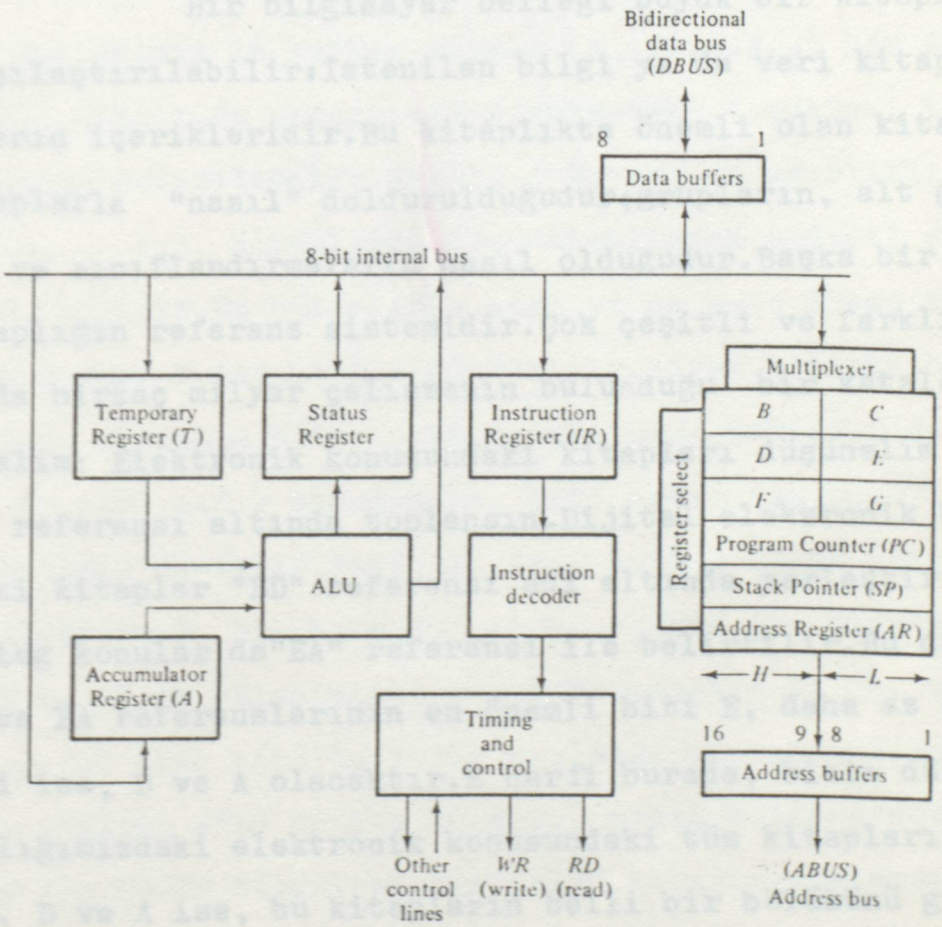


S_1	S_o	M=0 F_i	M=1 , C_i=0 F_i	M=1 , C_i=1	C_{i+1}
0	0	A_i	A_i	$\overline{A_i}$	0
0	1	$\overline{A_i}$	$\overline{A_i}$	A_i	0
1	0	$A_i \oplus B_i$	$A_i \oplus B_i$	$\overline{A_i} \oplus B_i$	$A_i B_i$
1	1	$\overline{A_i} \oplus B_i$	$\overline{A_i} \oplus B_i$	$A_i \oplus B_i$	$\overline{A_i} B_i$

Şekil I-5 Bir bitlik ALU

I-2-3 KONTROL DEVRESİ

Darbe girişlerini kullanarak, herhangi bir işlemin yürütülmesini sağlar.Bir komut getirilip çözüldükten sonra, kontrol devresi,işlemin başlaması için uygun işaretleri üretir.Kontrol devresi kesinti ve bekleme gibi dışsal işaretlere cevap verebilir.Bir kesinti, kontrol devresinin ana programı yürütmesini geçici olarak durdurur, kesintinin gerektirtirdiği atlamayı yaptıktan sonra otomatik olarak ana programa döner."Bekle" isteği sık sık, CPU'dan daha yavaş çalışan bellek ve I/O cihazları tarafından yayınlanır.Kontrol devresi I/O veya bellek tarafından veri hazırlanıncaya kadar CPU'yu bekletecektir.



Şekil I-6 CPU blok diyagramı

I-3 ADRES ÇÖZME

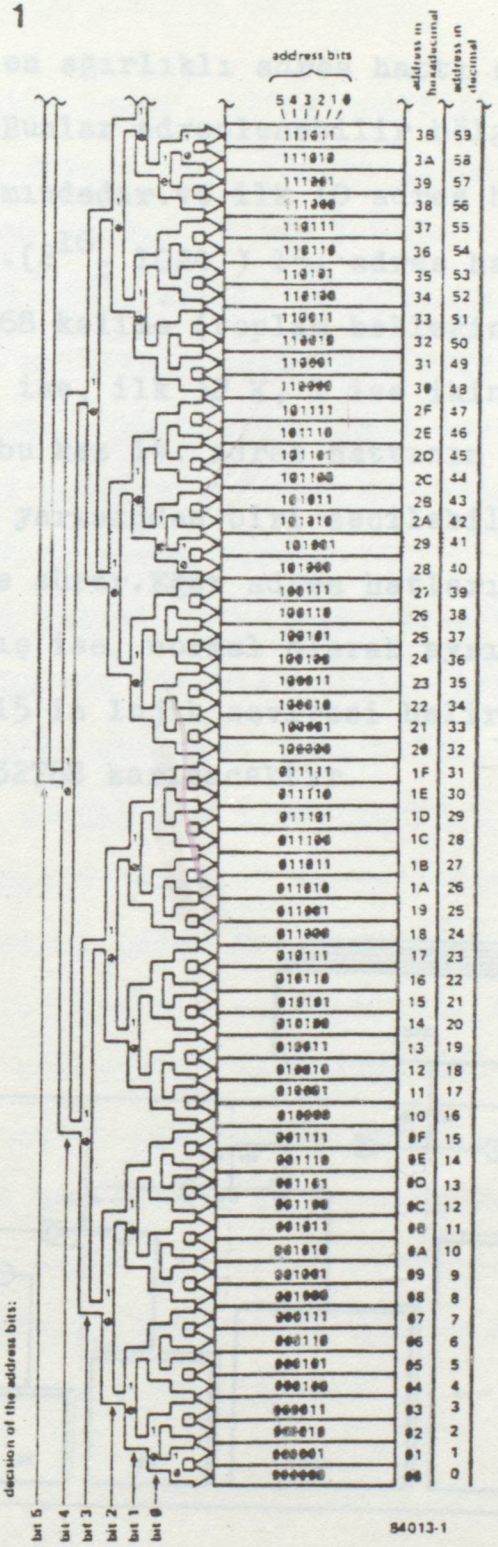
Bir bilgisayarda bellek, tamamen yazılımdan bağımsız olan lojik elemanların donanım kombinasyonudur fakat, yazılım da gözönüne alınmalıdır.Adreslenebilir bölgenin düzenlenmesi ve yapısı oldukça önemlidir.Bu, bir bilgisayarın en az anlaşılan karekteristiklerindendir ve makinanın çalışmasında esas rol oynar.

I-3-1 ADRESLENEBİLİR BÖLGENİN DÜZENLENİŞİ

Bir bilgisayar belleği büyük bir kitaplık ile karşılaştırılabilir:İstenilen bilgi ya da veri kitaplar ve onların içerikleridir.Bu kitaplıkta önemli olan kitaplığın kitaplarla "nasıl" doldurulduğudur;grupların, alt grupların ve sınıflandırmaların nasıl olduğudur.Başka bir deyişle kitaplığın referans sistemidir.Çok çeşitli ve farklı konularda birkaç milyar çalışmanın bulunduğu bir katalog varsayalım: Elektronik konusundaki kitapları düşünelim.Bunlar "E" referansı altında toplansın.Dijital elektronik konusunda ki kitaplar "ED" referansı adı altında yerleştirilsin. Analog konular da"EA" referansı ile belirtilir.Bu durumda ED ve EA referanslarının en önemli biti E, daha az önemli biti ise, D ve A olacaktır.E harfi burada, bizim düşsel kitaplığımızdaki elektronik konusundaki tüm kitapları göste rir, D ve A ise, bu kitapların belli bir bölümünü gösterir. Daha ayrıntıya girersek, bir sonraki karakter(ki yukardaki iki karakterden daha az ağırlıklı) İngilizce olan ve

olmayan kitapların ayrılması olabilir.Böylece "EDE" başlığı altındaki kitaplar İngilizcedir ve dijital elektronikle ilgilidir.

Bilgisayar belleğini bir apartman olarak düşünebiliriz.Her kat 8 birim içerir ve bu ayrık birimlere (bitlere) ayrı ayrı erişilemez; bayt olarak adlandırılan sekiz bitlik bir kelime oluştururlar.Bu sekiz bitin lojik değeri de bir VERİ meydana getirir.Bu kelimeler bir VERİ YOLU ile sistemin içine iletilebilirler.Her bir hat (D_0.....D_7) bir veri bitini iletir.Mikroişlemci, bellekteki kelimelere 16 hattan oluşan (A_0.....A_{15}) bir ADRES YOLUile ulaşır.Bu organizasyon yukardaki örnekteki kitaplık ile karşılaştırılabilir.Şekil I-7'de en az ağırlıklı altı bit (A_0....A_5), kitaplıkta düşünülebilecek ardışıl altı bölüm olabilir.Bu bölmelerin sağa ya da sola dönme kararı lojik H veya L seviye ile gösterilir.Adres bitinin binary ağırlığı arttıkça bölge daha önemli olur.ŞekilI-7'de bit5 ve bit4 sıfır olduğundan, bit3 sıfır ise,00-07 bölgesi seçilmiş demektir.Bit3 '1' ise, bu kez 08-0F bölgesine erişilebilir.5. bit hala '0' iken, 4. bit '1' olursa bit3 10-17 ve 18-1F bölgelerini seçer.Varsayalım ki, 4. ve 5. bitler '0' iken bit3'ün lojik seviyesi tanımlanmasın; o zaman yukarda sözü edilen bölgeler ayırt edilemeyecektir. 00-07 ile 08-0F bölgeleri birbirine karıştırılacaktır.Bu durum, "çift adresleme" olarak adlandırılır.Tanımlanmamış bitin binary ağırlığına bağlı olarak çift adresleme bölgesi daha az veya çok olabilir.

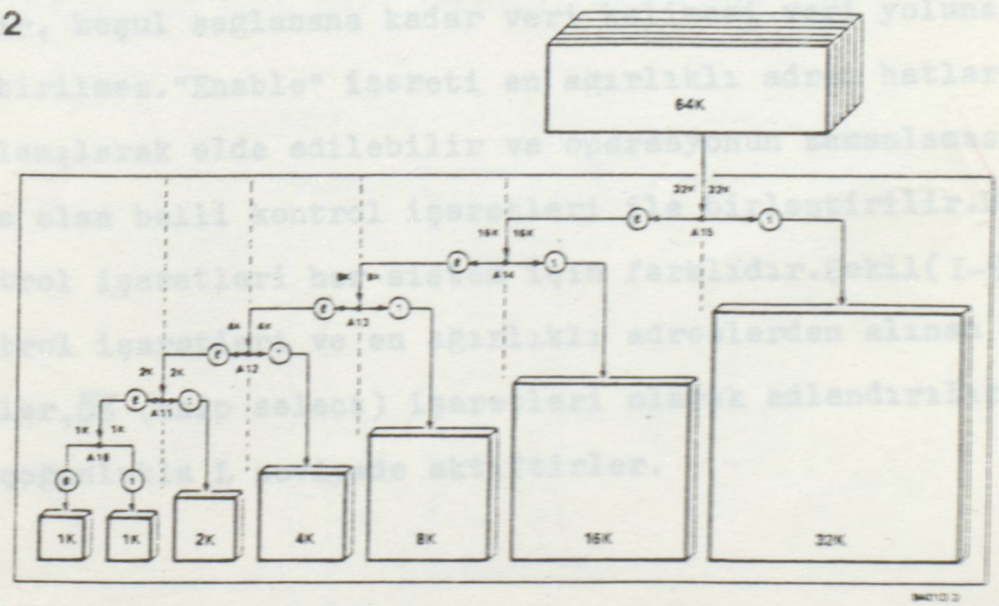


Şekil I-7 Bitlerin binary ağırlıklarına bağlı olarak adreslenen bölgeleri gösteren binary "ağaç"

$2^{16} = 65536$

6 en ağırlıklı adres hattı şekil I-8'de gösterilmiştir.(Bunlar adreslenebilir bölgeleri ayırırlar.) K,1024 bayt anlamındadır.ve ilk 10 adres hattı ($A_0 \ldots A_9$) ile erişilebilir.($2^{16} = 1024$) 15. adres hattının 1 veya 0 oluşuna göre 32768 kelime (toplam belleğin yarısı) seçilebilir.(15. bit 0 ise, ilk 32 K, 1 ise ikinci 32 K)Bu her iki blok içinde bu kez 14. adres hattının 1 veya 0 oluşuna göre, bloğun iki yarısından biri seçilebilir.Bu durum, 10. hatta kadar böyle sürer.Eğer adres hatlarının lojik seviyeleri tanımlanmamış ise, normal olarak ayrı iki blok karışacaktır.Örneğin A15'in lojik seviyesi belirlenmemiş ise, adres 0 ile adres 32768 karışacaktır.

2

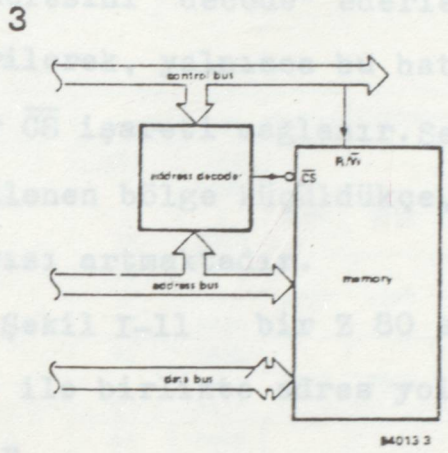


Şekil I-8 En ağırlıklı bitlerin lojik seviyesinin adreslenebilir bölgeyi bloklara ayırması

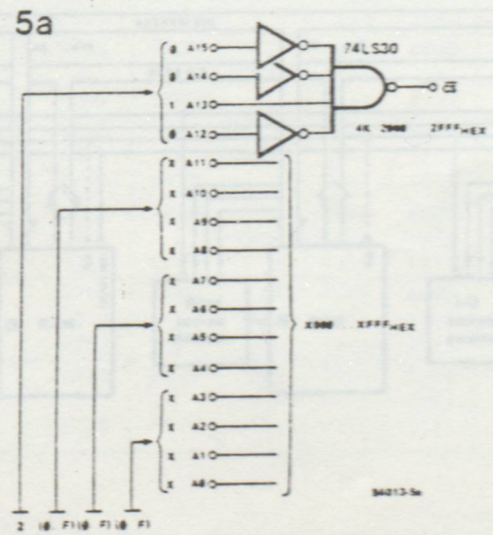
I-3-2 "ENABLE" İŞARETLERİNİN ÜRETİLMESİ

Şimdiye kadar, adreslemeye bir topografi sorunu olarak bakıldı.Entegre devrelerin çoğunluğunda 16 değil daha az adres hattı vardır. Şekil I-8'de görüleceği gibi 4 K'lık bir bellek (2732 EPROM) 12 adres hattı gerektirir. (A_0 ....A_{11}) 4096 kelimenin herbirini adreslemek entegre devre içindeki adres çözücüler(address decoders) ile gerçekleştirilir.Aynı şekilde 2 K'lık ber bellek(6116 RAM) için 11 adres hattına gerek vardır.

Her bellek entegresi adres hatlarının yanısıra,bir ya da daha çok "enable" girişe sahiptir.Bu girişler chip'i aktif hale getirmek için belli bir lojik seviyede olmalıdır.(Bu, genellikle L seviyedir.) Bu demektir ki, iç adresleme, yalnızca "enable" işareti olduğunda yapılabilir, koşul sağlanana kadar veri kelimesi veri yoluna yerleştirilmez."Enable" işareti en ağırlıklı adres hatları kullanılarak elde edilebilir ve operasyonun zamanlaması için esas olan belli kontrol işaretleri ile birleştirilir.Bu kontrol işaretleri her sistem için farklıdır.Şekil(I-9) Kontrol işaretleri ve en ağırlıklı adreslerden alınan işaretler,$\overline{CS}$ (chip select) işaretleri olarak adlandırılırlar ve çoğunlukla L seviyede aktiftirler.



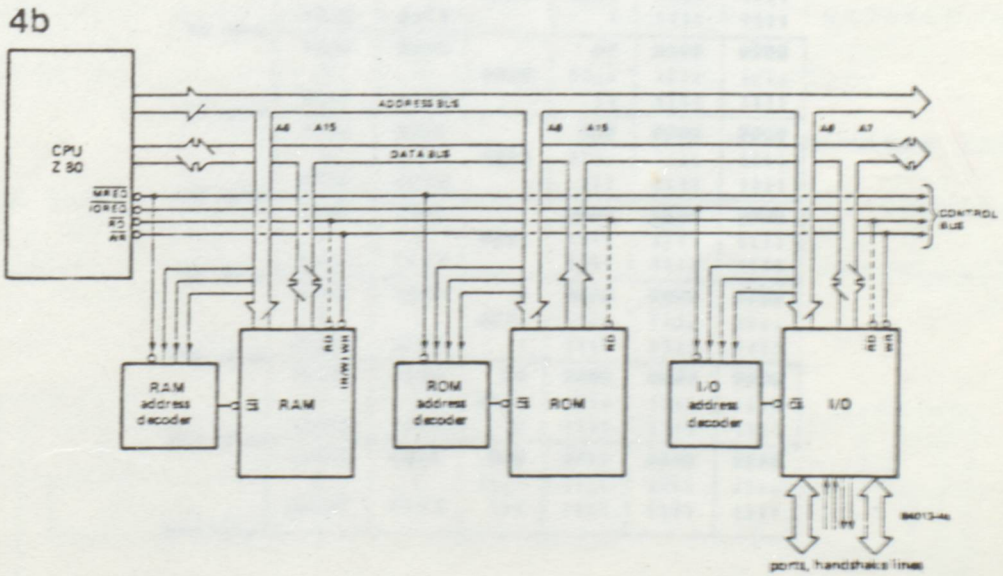
Şekil I-9 Veri ve adres yolları belleği adreslemek için yeterli değildir; okuma ve yazmayı doğru zamanlayabilmek için bazı kontrol işaretleri gerekir.



Şekil I-10 4 K'lık bir bellek bölgesi için adres çözücü devre

Belleğin 2000-2FFF bölgeleri arasını adresleyebilecek lojik kombinasyon şekilI-10'da görülmektedir. (2000-2FFF:4 K) $A_{11}....A_o$ hatları X000-XFFF arasındaki 4098 bellek hücresini "decode" ederler.A_{15} ve A_{12} şekildeki gibi birleştirilerek, yalnızca bu hatlarda 0010_2 olduğunda aktif olan bir $\overline{CS}$ işareti sağlanır.Şekilden de anlaşılacağı gibi, adreslenen bölge küçüldükçe, kombine edilen adres işaretleri sayısı artmaktadır.

Şekil I-11 bir Z 80 sisteminin özel kontrol işaretleri ile birlikte adres yolu bağlantılarını görüntülemektedir.



Şekil I-11 Z80 yol sistemi

	ADRESSES		15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
	DEC.	HEX				
	0 ↓ 15	0000 ↓ 000F	0000	0000	0000	0000 ↓↓↓↓ 1111
	16 ↓ 31	0010 ↓ 001F	0000	0000	0001	0000 ↓↓↓↓ 1111
	32 ↓ 63	0020 ↓ 003F	0000	0000	0 001↓ 1	0000 ↓↓↓↓ 1111
	64 ↓ 127	0040 ↓ 007F	0000	0000	00 01↓↓ 11	0000 ↓↓↓↓ 1111
	128 ↓ 255	0080 ↓ 00FF	0000	0000	000 1↓↓↓ 111	0000 ↓↓↓↓ 1111
	256 ↓ 511	0100 ↓ 01FF	0000	0001	0000 ↓↓↓↓ 1111	0000 ↓↓↓↓ 1111
	512 ↓ 1023	0200 ↓ 03FF	0000	0 001↓ 1	0000 ↓↓↓↓ 1111	0000 ↓↓↓↓ 1111
1K ↔	1024 ↓ 2047	0400 ↓ 07FF	0000	00 01↓↓ 11	0000 ↓↓↓↓ 1111	0000 ↓↓↓↓ 1111
2K ↔	2048 ↓ 4095	0800 ↓ 0FFF	0000	000 1↓↓↓ 111	0000 ↓↓↓↓ 1111	0000 ↓↓↓↓ 1111
4K ↔	4096 ↓ 8191	1000 ↓ 1FFF	0001	0000 ↓↓↓↓ 1111	0000 ↓↓↓↓ 1111	0000 ↓↓↓↓ 1111
8K ↔	8192 ↓ 16383	2000 ↓ 3FFF	0 001↓ 1	0000 ↓↓↓↓ 1111	0000 ↓↓↓↓ 1111	0000 ↓↓↓↓ 1111
16K ↔	16384 ↓ 32767	4000 ↓ 7FFF	00 01↓↓ 11	0000 ↓↓↓↓ 1111	0000 ↓↓↓↓ 1111	0000 ↓↓↓↓ 1111
32K ↔	32768 ↓ 65535	8000 ↓ FFFF	000 1↓↓↓ 111	0000 ↓↓↓↓ 1111	0000 ↓↓↓↓ 1111	0000 ↓↓↓↓ 1111
64K ↔						

Şekil I-12 16 adres hattı ile 65536 kelimenin adreslenmesi

I-4 YOL YAPISI

Mikroişlemci, YOL olarak adlandırılan işaret hattı grupları ile bilgi göndererek mikrokomputer sistemin diğer bölümleriyle iletişim kurar.Sistemin çeşitli elemanları arasında iletilen bilgi türleri şunlardır:

-Program belleği adresleri

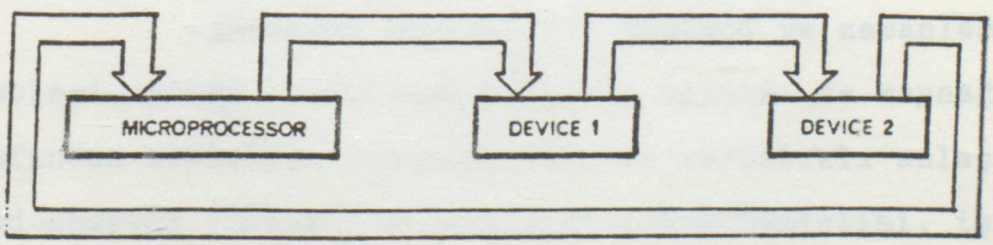
-Komut kodları

-Veri bellek adresleri

-Giriş-çıkış cihaz adresleri

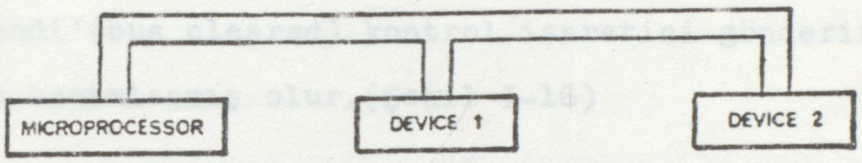
Büyük bir kompüter sisteminde, her tip bilgi kendi için ayrılan yol ile iletilir.Bir mikrokompüterde ise, birkaç tip bilginin aynı yolu kullanması yaygındır. Ortaklaşa yol kullanılmasının nedeni, standart bir tümdevre bacaklarının sınırlı sayıda olmasıdır.Ayrıca iç bağlantı sayısını azaltmak için "data multiplexing" gerekir.Ortaklaşa yol sistemine dayalı iki tür yol vardır:

-Daisy-chain yol: Doğru cihaz bulunana kadar bilgi tek yollu bir çevrim halinde iletilir.Yol üstündeki her cihaz, bilgi kaynağı veya alıcısı olarak davranabilir.



Şekil I-13 Daisy-chain yol yapısı

-"Party-line" yol:Her cihaz, doğrudan tekbir yola bağlanır.Yalnızca tek bir cihaz bilgi kaynağı olarak davrandığı zaman, yol tek yönlüdür.



Şekil I- 14 Party-line yol yapısı

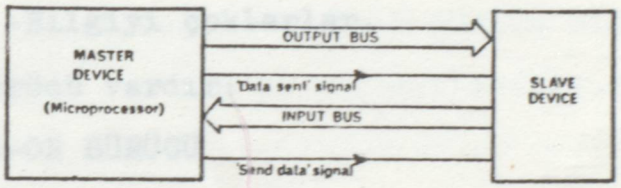
I-4-1 YOL KONTROL İŞARETLERİ

Yoldaki bilgi akış hızını kontrol etmenin yanısıra, çift yönlü bir yolda, bilgi akış yönünü, bilgi türünü,birden çok cihazın aynı yola bağlanması durumunda alıcı ve vericiyi saptamak için yol kontrol işaretlerine gerek vardır.Bunun için iki yöntem vardır:

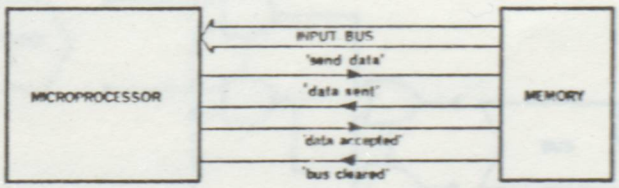
-Senkron Kontrol:Tüm yol kontrol ve zamanlama işaretlerine mikroişlemci üretir.Sistemdeki diğer elemanlar veri yolundaki bilgiyi iletir veya alırken, bu işaretlere eşzamanlanır.(Şekil I-15)

-Asenkron Kontrol:Yol kontrol ve zamanlama işaretleri, yolda iletilecek bilginin alıcısı ve kaynağı tarafından ortaklaşa üretilir.Bunu da karşılıklı anlaşarak (hand-shaking yöntemi) gerçekleştirirler.Sözgelimi, işlemci bellekten veri isteyecekse, işleme 'veriyi gönder'(send data) kontrol işareti ile başlar ve daha sonra bellekten

bir yanıt alana kadar "halt"durumunda bekler.Veri yola yerleştirildikten sonra, işlemciye 'veri gönderildi'(data sent) işareti gelir.Bu işareti olan işlemci, yoldan veriyi alır ve'veri alındı'(data accepted) işaretini yayar.Bunu alan bellek veriyi yoldan kaldırır ve 'yol temizlendi'(bus cleared) kontrol işaretini gönderir ve işlem tamamlanmış olur.(Şekil I-16)



Şekil I-15 Senkron yol kontrolü



Şekil I-16 Asenkron yol kontrolü

I-4-2 BİRDEN ÇOK KAYNAĞIN AYNI YOLA BAĞLANMASI

Bir yola bilgi, birden çok kaynaktan sağlanıyorsa, karışıklığı önlemek için kaynak çıkışlarını kontrol etmek ve özel yol sürücü devreler kullanmak gerekir. Yol sürücüler üç temel işlevi yerine getirirler:

-Yola bağlanan kaynaklar arasında elektriksel uygunluk sağlarlar.

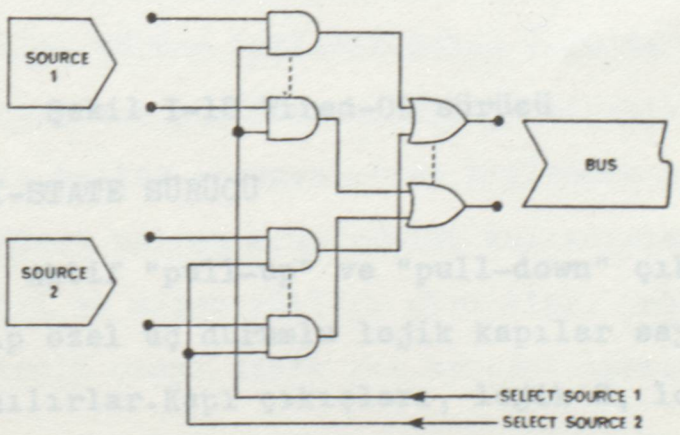
-Kaynağın yola bağlanmasını ve bağlanmamasını denetlerler.

-Bilgiyi çoklarlar.

Üç tip yol sürücü vardır:

I-4-2.1 LOJİK-OR SÜRÜCÜ

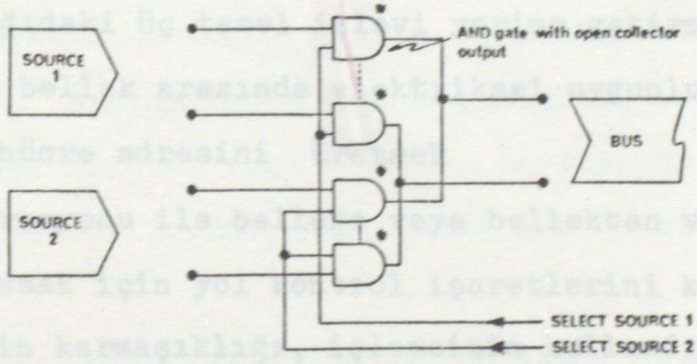
Kaynaklardan birinin ya da diğerinin daima yola bağlı olmasından dolayı, çift yönlü yol ile bu sürücü kullanılamaz.Ayrıca, uzak mesafelerde kaynak içeren sistemlerde elverişsizdir, yola kaynak eklenmesi gerektiğinde değiştirilmesi zordur.(Şekil I-17)



Şekil I-17 Lojik-OR sürücü

I-4-2.2 WIRED-OR SÜRÜCÜ

Her bilgi kaynağı çıkışında açık-kollektörlü lojik kapılar kullanılarak çoğullama sağlanır.Çıkış kapıları doğrudan yola bağlanır.Açık-kollektörlü kapıların özelliği, bir çıkış transistörünün iletimde olması, kesimde olan diğerlerine üstün geleceğinden, ortak çıkış hattının durumunun otomatikman saptanmasıdır.Seçilmeyen kaynakların çıkış transistörleri kesimde olduğundan, seçilen kaynağın çıkış transistörü yola konacak veriyi saptar.Kaynakların hiçbirinin seçilmemesi de mümkün olduğundan, bu sürücü çift yönlü yol ile de kullanılabilir.(Şekil I-18)

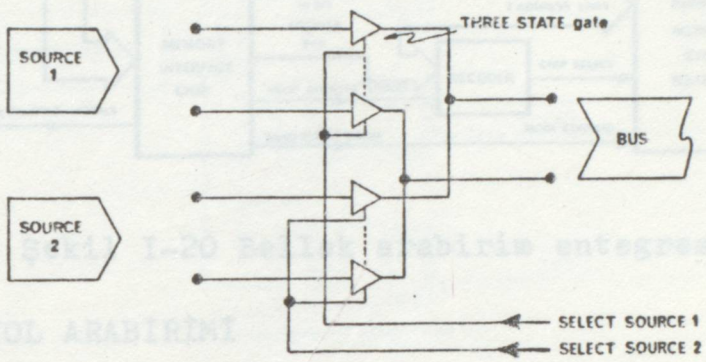


Şekil I-18 Wired-OR sürücü

I-4-2.3 TRI-STATE SÜRÜCÜ

Aktif "pull-up" ve "pull-down" çıkış durumlarına sahip özel üç durumlu lojik kapılar sayısal çoklamada kullanılırlar.Kapı çıkışları, lojik 0, lojik 1 veya yüksek empedans durumundadır.Kapı "enable" edildiğinde, çıkışı, giriş saptar.Kapı "disable" edildiğinde çıkış yüksek empedans durumundadır; ne kaynak, ne alıcı gibi dav-

ranır.Özel bir yolda kullanılacak sürücünün tipi mikroişlemciye bağlıdır.Yol, sürücüsünün adıyla anılır.



Şekil I-19 Tri-state sürücü

I-4-3 BELLEK/YOL ARABİRİMİ

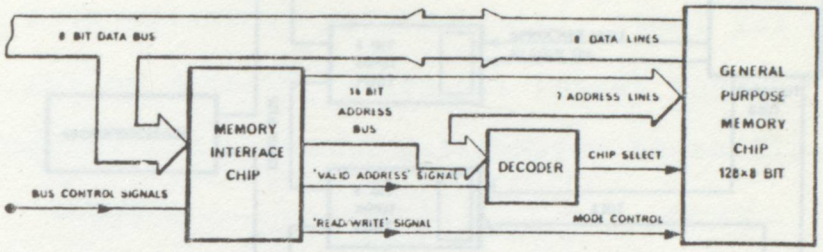
Mikroişlemci ile bellek arasındaki bir arabirim, aşağıdaki üç temel işlevi yerine getirmelidir:

-Yol ve bellek arasında elektriksel uygunluğu sağlamak

-Bellek hücre adresini üretmek

-Yol operasyonu ile belleğe veya bellekten veri transferini eşzamanlamak için yol kontrol işaretlerini kullanmak

Arabirimin karmaşıklığı, işlemcinin kullandığı yol sistemine ve bellek türüne bağlıdır.Şekil I-20'de bir bellek arabirim entegresi görülmektedir.Çoğu mikroişlemciler, genel amaçlı bellek entegrelerine bağlanmak için tasarlanmış bu standart bellek arabirimini kullanırlar.Bu tür sistemlerde veri ve adres yolları ayrı olup, "chip select" ve mod kontrolü olmak üzere iki kontrol işareti kullanılır.

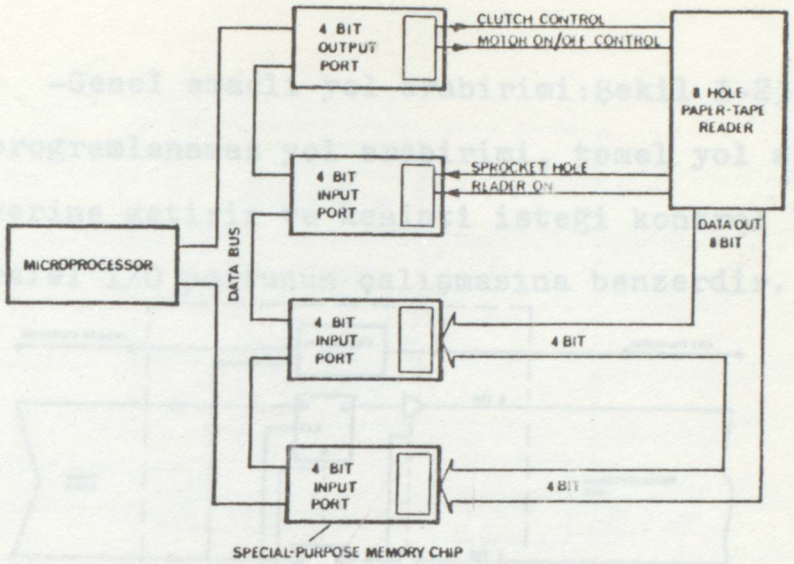


Şekil I-20 Bellek arabirim entegresi

I-4-4 I/O YOL ARABİRİMİ

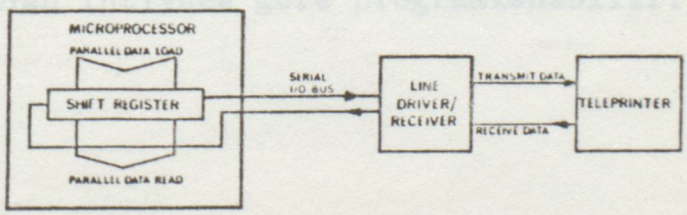
I/O yol arabirimi, yol-bellek arabirimine benzer ancak, daha karmaşık işleve sahiptir.Çünki, I/O cihazları oldukça yavaş çalışırlar ve veri akışının arabirim vasıtası ile kontrol edilmesi gerekir.Ve çoğu I/O cihazları elektro-mekaniktir, yol uygunluğu ve işaret standardizasyonu sağlayamazlar.Bunun yanısıra, cihazdan veya cihaza veri iletimi için arabirim devresinde kod dönüştürmek gerekebilir.Dört farklı tip I/O arabirimi tanımlanabilir:

-Paralel I/O portu:En temel türdür, tek veya çift yönlü olabilir.Çıkışlarda açık-kollektörlü veya tri-state"latch" vardır.Port'lar genellikle, 4 veya 8 bit genişliğindedir, ve normal olarak mikroişlemci chip'i üzerine veya özel amaçlı bellek chip'ine yerleştirilir. Port vasıtasıyla veri iletimi işlemci ile senkronizedir ve tek yönlü port için yol kontrol işaretlerine gerek yoktur.Şekil I-21'de dört portun kağıt-teyp okuyucu arabirimi düzenlemesinde kullanılışı görülmektedir.



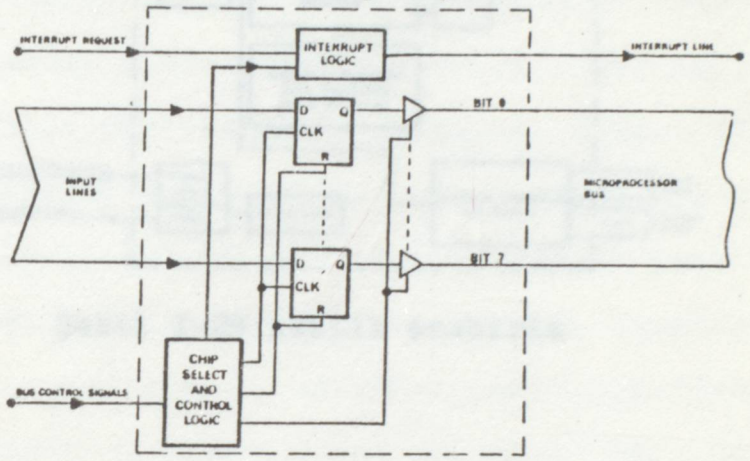
Şekil I-21 Çoklu I/O portları kullanılarak kağıt-teyp okuyucu arabiriminin gerçekleştirilmesi

-Seri I/O portu:Port, 8 veya 16 bitlik bir "shift register"ın seri giriş ve seri çıkışına bağlı iki tane tek bitlik veri yolundan oluşmuştur.Genellikle, mikroişlemci chip'i üstüne yerleştirilen bu yazmaç, paralel giriş ya da çıkış hatları ile yüklenebilir ya da okunabilir.Program kontrolü altında bir kerede bir bit kaydırılır.Şekil I-22'de görüldüğü gibi seri I/O portu çoğunlukla, bir teleprinter için asenkron veri iletimi arabirimi olarak kullanılır."Shift register" yazılım kontrolü ile paralelden seriye, seriden paralele kod dönüşümü yapar.



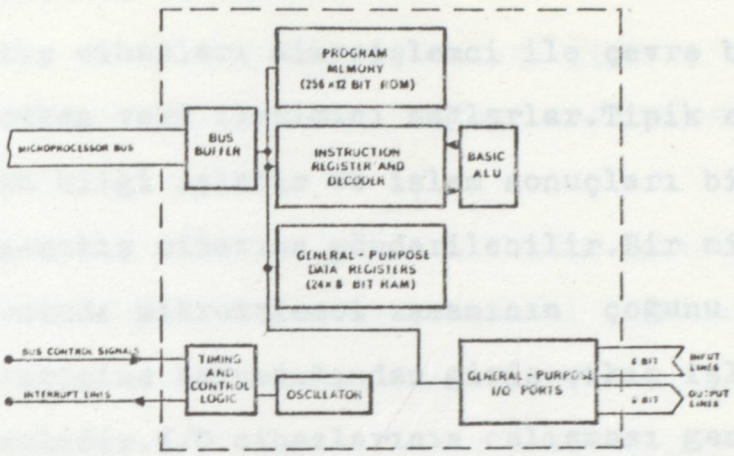
Şekil I-22 Teleprinter arabiriminde kullanılan seri I/O portu

-Genel amaçlı yol arabirimi:Şekil I-23'de gösterilen, programlanamaz yol arabirimi, temel yol arabirim işlevini yerine getirir ve kesinti isteği kontrol lojiği içerir.Paralel I/O portunun çalışmasına benzerdir.



Şekil I-23 Programlanamaz giriş arabirimi

-Akıllı yol arabirimi:Şekil I-24'de gösterilen akıllı arabirim, I/O kontrolü ve arabirim işlemlerini sınırlı sayıdaki komutlarla gerçekleştirmek üzere özel olarak düzenlenmiş, tek chip üzerinde basit bir mikrokompüterdir.Bu chip üzerinde sisteme uygun yol, I/O kolaylıkları ve küçük bir veri ve program belleği vardır.Özel bir I/O cihazı için gerekn arabirim işlevini yerine getirmek üzere, sistem, üretim sırasında programlanır.Genel amaçlı akıllı arabirimlerde, chip üzerindeki ROM bellek kullanıcı tarafından ihtiyaca göre programlanabilir.



Şekil I-24 Akıllı arabirim

I-5 GİRİŞ/ÇIKIŞ İŞLEMLERİ

Giriş-çıkış cihazları mikroişlemci ile çevre birimleri arasında gereken veri iletimini sağlarlar.Tipik olarak,girişten alınan bilgi işlenir ve işlem sonuçları bir ya da daha çok giriş-çıkış cihazına gönderilebilir.Bir mikrobilgisayar sisteminde mikroişlemci zamanının çoğunu I/O cihazları ile iletişime harcadığından giriş-çıkış işlemleri özellikle önemlidir.I/O cihazlarının çalışması genellikle mikroişlemciden bağımsızdır ve veri iletimi sırasındaki işlem ile program yürütülmesi eşzamanlı olmalıdır.Veri iletiminin eşzamanlanması ve denetlenmesi yöntemine göre üç temel tip giriş/çıkış vardır:

-Program denetimli I/O

-Kesinti denetimli I/O

-Doğrudan bellek erişimli I/O

Özel bir uygulamada kullanılan I/O tipi üç ana etkene bağlıdır:

1-Verinin hangi hızda iletileceği,

2-I/O cihazının veri iletmeye hazır olduğu an ile iletimin gerçekleştiği an arasındaki maksimum gecikme zamanı,

3-Giriş-çıkış bağlantısına ve diğer mikroişlemci operasyonlarına

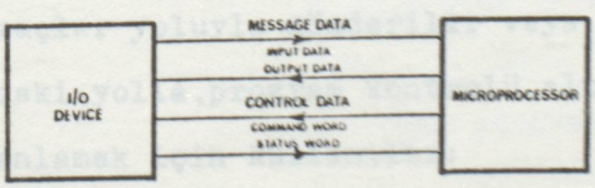
I-5-1 PROGRAM DENETİMLİ I/O

Şekil I-25'de görüleceği gibi, I/O ile mikroişlemci arasında iki temel tip bilgi iletilir:Mesaj ve denetim verisi.

Denetim verisi, mesaj verisi iletilmeden önce program yürütülmesi ile I/O cihazını eşzamanlamak için kullanılır. Giriş denetim verisi cihazın durum kelimesi olarak adlandırılır.Çıkış denetim verisi cihaz komut kelimesi olarak adlandırılır.Program denetimli I/O ile giriş/çıkış komutları, her tip veri iletimini denetlemek ve başlatmak için kullanılır.

Durum kelimesi cihazın o andaki durumunu saptamak için işlemciye okunur.Durum kelimesinin her biti, iletim hatası, cihaz kullanılamaz durumda, cihaz meşgul, mesaj verisi iletime hazır gibi özel cihaz durumunu gösterir.

Komut kelimesi cihazın çalışmasını denetlemek için mikroişlemciden gönderilir.Komut kelimesinin her biti, motoru durdur, beslemeyi artır, iletim hızını değiştir gibi özel bir işleve sahiptir.



Şekil I-25 I/O cihazı ile işlemci arasındaki bilgi akışı

Giriş-çıkış komutları aşağıdaki üç yoldan biri ile organiedilebilir:

1- Her bir I/O cihazını tanımlamak için tek bir cihaz adresi kullanarak her tür I/O veri iletimi tek bir komutla sağlanır.

- Veriyi oku
-Veriyi yaz
-Komutu gönder
-Durumu al

2-Biri giriş, biri çıkış için olmak üzere ,mesaj ve denetim verilerin iletmek için iki giriş-çıkış komutu kullanılır.Özel bir I/O cihazı için mesaj ya da denetim verisi iletimini ayırt etmek için iki cihaz adresi kullanılır.

3-I/O komutları arasında bir fark yoktur.Bellek veri iletim komutları, cihaz adresi olarak kullanılmamış bir bellek adres bloğunu atayarak I/O cihazları ile iletişimi sağlamak için de kullanılır.Bu yaklaşım"memory-mapped I/O" olarak adlandırılır.Kullanılabilir bellek adres alanı küçültülmesine rağmen bu yaklaşım hem program depolama gereğini ve program yürütme zamanını azaltır.Çoğu mikroişlemcilerde hem mesaj, hem denetim verisi akümülatör veya bazı diğer yazmaçlar yoluyla gönderilir veya alınır.Denetim verisi aşağıdaki yolla,program kontrolü altında veri iletimini eşzamanlamak için kullanılır:

1-Bir komut kelimesi mesaj verisinin iletilmesini istemek için I/O cihazına yazılır.

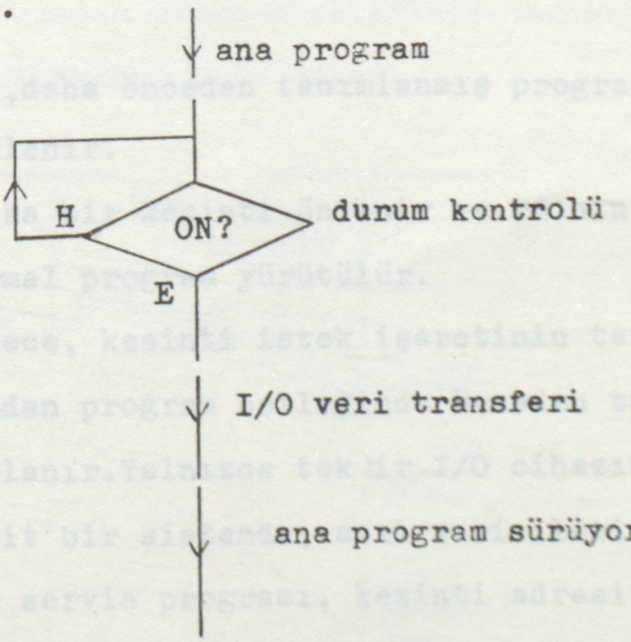
2-I/O'dan durum kelimesi okunur.

3-Mesaj verisinin iletilip iletilmeyeceğini test etmek için uygun durum biti kontrol edilir.

4-I/O iletime hazır değilsi hazır olana kadar 2ve3. adımlar tekrarlanır.

5-Mesaj verisi I/O'ya yazılır veya I/O'dan okunur.

Veri iletimin başlatmak için karar I/O'nun kendisine aitse, ilk adım çoğu uygulamalarda ihmal edilir. Bu durumda cihaz uygun durum bitini set ederek veri iletimi arzusunu gösterir.I/O iletime hazır olana kadar durum kontrolü tekrarlandığı için, bu çevrim pogram yürütülmesini durdurur ve işlem zamınının israfına neden olur.Bu sorun, birkaç I/O cihazı ile bağlantısı olan mikroişlemcilerde özellikle önemlidir.Bazı mikroişlemcilerde cihaz durumunu kontrol için harcanan zaman tek bir test hattı kullanılarak azaltılabilir.Bu hat tüm I/O cihazları tarafından ortaklaşa kullanılır.



Şekil I-26 durum kontrol çevrimi

I-5-2 KESİNTİ KONTROLLU I/O

Program kontrollu I/O'nun temel dezavantajı ana program yürütülmesinin periyodik olarak kesintiye uğraması ve veri iletimi için herhangi bir cihazın hazır olup olmadığının kontrol edilmesidir.Bu kontrol işlemi gereksiz zaman tüketimine neden olur.Kesinti kontrollu I/O ile bu sorun çözülmüştür.Ana programın yürütülmesi yalnız ve yalnız veri iletimi hazır olduğu zaman I/O tarafından kesilir.

En basit tipik bir kesinti sisteminde yalnızca bir I/O cihazı tek bir kesinti istek hattına bağlanır. Bu hatta işaret olması mikroişlemcinin otomatik olarak aşağıdaki minimum işlem dizisini yapmasına neden olur:

1-İşaret geldiği anda yerine getirilmekte olan komut tamamlanır.

2-Program sayıcısının o andaki içeriği depolanır.

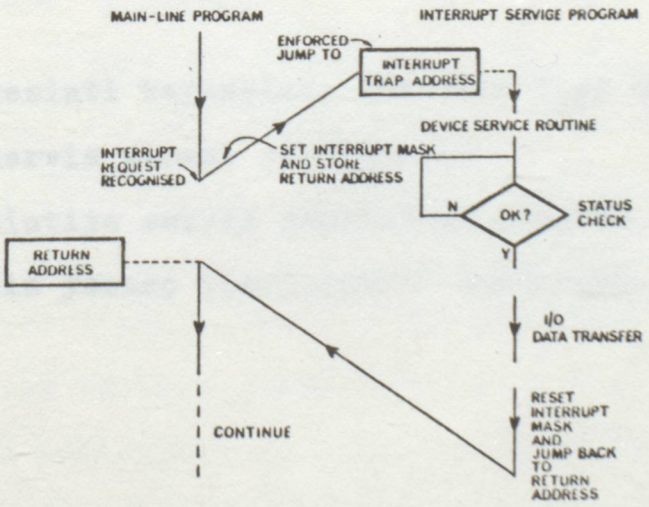
3-PC ,daha önceden tanımlanmış programın bellek adresi ile yüklenir.

4-Başka bir kesinti önlenir ve PC'nin yeni içeriğine göre normal program yürütülür.

Böylece, kesinti istek işaretinin tanınması ana program hattından program belleğinde önceden tanımlanan bir bölgeye atlanır.Yalnızca tek bir I/O cihazının kesinti ürettiği basit bir sistemde, asıl veri iletimini kontrol eden cihaz servis programı, kesinti adresi ile yüklenir.Şekil I-27'de görüldüğü gibi,cihaza servis programı tamamlandıktan sonra PC'nin program kesildiği anda

depolanan içeriği, bu programın dönüş adresi olur.Ve ana program kesildiği yerden devam eder.Cihaz servis programı yürütülmeye başlandıktan sonra aynı kesinti istek hattı ile çoklu kesintiyi önlemek için gelebilecek diğer kesintiler otomatik olarak önlenir.Bu da mikroişlemci içindeki kesinti engelleme biti set edilerek gerçekleştirilir.Çoğu sistemlerde bu bit durum yazmacının bir parçasıdır ve yazılım ile set veya reset edilebilir. Engelleme(mask)biti kesilmeksizin yürütülmesi gereken program kesitleri için kullanılır.

Çoğu mikroişlemciler, cihaz servis programına bağlanmak için dolaylı adreslemeden yararlanan kesinti sistemlerini kullanırlar.Kesinti adresi komutun kendini değil servis programının ilk komutunun adresini içerir.Kesinti donanımı otomatik olarak bu adresi PC'yi yükler.Dolaylı adresleme servis programının programbelliğinde herhangi bir keyfi yere yerleştirilmesine izin verir.



Şekil I-27 Kesinti kontrollü I/O

I-5-2.1 ÇOKLU KESİNTİLİ SİSTEMDE KESİNTİ SERVİSİ

Önceki bölümde tanımlanan kesinti servis işlemi tek bir I/O cihazı olan sistemlerde uygundur.Çoğu sistemlerde birden çok I/O ve kesinti türü vardır.Üç temel tip kesinti tanımlanabilir:

1-Dış kaynaklı kesintiler:Bir ya da daha çok I/O cihazı tarafından gönderilebilir.

2-İç kaynaklı kesintiler:Özel bir durum ya da hata olduğunu göstermek için mikroişlemcinin kendisi tarafından üretilir. (Sözgelimi, güç yetersizliği, iletim hatası gibi durumlar)

3-Simüle edilmiş kesintiler:Kesinti servis testine ya da program hata bulucuya yardımcı olmak amacıyla yazılımla üretilebilir.

Farklı kesinti kaynakları farklı servis gerektirecektir.Kimileri hemen hizmet isteyecek, kimileri de halihazırdaki işlemin tamamlanması için bir gecikme kabul edebilecektir.Böylece kesinti servis işlemi aşağıdaki gibi olmalıdır:

1-Çeşitli kesinti kaynakları arasında fark olmalı.

2-Kesinti servis sırası saptanmalı.

3-Çoklu kesintiye servis yapılırken program sürekliliğini sağlamak için yazmaç içeriklerini depolamak.

I-5-2.2 KESİNTİ İSTEYEN KAYNAĞIN TEŞHİSİ

Bazı mikroişlemcilerde birkaç kesinti istek hattı ve her birinin kendine has kesinti adresi vardır.Kesinti isteyen kaynağın tanınması sorunu, her hatta tek bir kesinti kaynağı atayarak çözülebilir.Bu yaklaşım, iç kaynaklı, dış kaynaklı ve simüle edilmiş kesintilerin ayırt edilmesinde yaygın olarak kullanılır.Çoğu mikroişlemcilerde birkaç I/O cihazı aynı kesinti istek hattına bağlanmak zorundadır.Bu sistemlerde kesinti kaynağını tanımak için iki yöntem yaygın olarak kullanılır:

1-Cihaz seçimi: Kesinti, bir kesinti adresi ile kesinti servis programına atlamaya neden olur.Servis programının başlangıç kesiti , hangi I/O cihazının kesinti istediğini saptamak için tüm I/O cihazlarının durum kelimesini kontrol eder.Cihaz durum kelimesi mikroişlemci durum yazmacına okunur ve eğer ilgili bit set edilmişse , ilgili cihaz servis programına atlanır.Bu yöntemde cihazın tanınması yazılımla sağlanır.

2-Vektörlü Kesintiler: Bu sistemde, mikroişlemci içindeki kesinti kontrol lojiği I/O kesintisini tanır Her I/O cihazının tekbir kesinti adresi vardır.(Cihaz adresi ile karıştırılmamalıdır.)Kesinti kontrol lojiği kesintiyi tanıyınca cihazın kesinti adresini mikroişlemciye bildirir.Bu adres tanınan cihaz için,tek bir kesinti adresi üretmek için kullanılır.Bu adresler genellikle program belleğinde ardışıl bölgelere yerleştirilir ve kesinti vektörü olarak adlandırılırlar.

Vektördeki her bölge bir cihaz servis programının başlama adresini içerir.Kesinti vektöründe tanımlanan özel kesinti adreslerinin içeriği PC'ye yüklenir ve program kontrolü seçilen cihaz servis programına geçer.Bu işlem, bazı mikroişlemcilerde cihaz kesinti adresi olarak kesinti adreslerinin kullanılması ile basitleştirilir.

Kimi mikroişlemcilerde bir adres iletmek yerine, I/O cihazının kesinti isteği kabul edildikten sonra tek bir baytlık komut mikroişlemciye gönderilir.Kesinti kontrol lojiği otomatik olarak komut kodunu komut yazmacına yükler ve bundan sonra komut , diğer işlemlerde olduğu gibi yürütülür.Kesinti vektörü ile cihazın tanınması donanım ile gerçekleştirilir.

I-5-2.3 KESİNTİ ÖNCELİĞİ

Birkaç kesinti kaynağının olması durumunda ilk gelen kesinti programı yürütülürken bir yada daha çok kesinti istenme olasılığı her zaman vardır.Basit sistemlerde, ilk kesinti isteği tanındığı zaman, kesinti engelleme biti hemen set edilir.İlk programın gereği yapıldıktan sonra,sırayla diğerleri de hizmet verilecektir.Sıradaki kesinti isteğinin tanınma sırası,servisten önceki zaman gecikmesini saptamada kritik faktördür.Sıra ya da öncelik yazılım veya donanımla düzenlenebilir.

YAZILIM ÖNCELİĞİ:Kesinti isteğinin tanınmasından sonra, servis programı, her cihazın kesinti önceliğini saptayan bir sırada I/O cihazlarını kaydeder.

Böylece en önceliğe sahip olan en önce kaydedilir.

DONANIM ÖNCELİĞİ:Mikroişlemcinin kesinti kontrol lojiği her I/O cihazına kesinti isteği olup olmadığını anlamak için bir işaret gönderir.Kesinti mask bitinin durumunu yansıtan kontrol işareti sırayla her cihazı gider.(Daisy-chain yapı) Mask biti set edilmişse,işaret diğer cihazlaların kesinti isteği üretmesini engelleyecektir.Eğer mask biti reset durumunda ise,işaret bir sonraki cihaza devam edecektir.İşaret,kesinti servisi bekleyen bir cihaza geldiği zaman, cihazdaki kesinti lojiği işaretin bir sonraki cihaza geçmesini önleyerek bir kesinti isteği üretir.Kontrol hattında cihazın pozisyonu kesinti önceliğin saptar. Çünki, birden fazla cihaz kesinti servisi bekliyorsa, kontrol işaretini önce alana önce hizmet verilecektir.Çoğu mikroişlemcilerde iki kesinti istek hattı vardır. Biri yazılım kontrollü mask bitine sahip, öteki ise, daimi olarak "enabled" durumundadır.Bu, "non-maskable" kesinti istek hattı en yüksek kesinti önceliğine sahiptir ve her durumda, her zaman derhal servis gerektiren uygulamalarda kullanılır."simulated" kesintilerde mask biti yoktur ancak, kesinti bir program komutu tarafından üretildiğinden en düşük kesinti önceliğine sahiptir.

Kimi vektörlü kesinti sistemlerinde kesinti öncelikleri otomatik olarak tanımlanmış ve mikroişlemcinin kesinti kontrol lojiği ile kontrol edilmiştir.Bir kesinti isteği olduktan ve cihaz kesinti adresi mikroişmeciye

iletildikten sonra, bu adres, bir çok bitli kesinti"enable-mask" ile karşılaştırılır.Cihaz kesinti adresi, bu mask'a eşit veya küçükse, kesinti isteği kabul edilir ve mask cihaz kesinti adresinden bir küçük değeri almaya zorlanır. Cihaz adresi bu değerden büyükse, kesinti isteği sıraya sokulur.Böylece, cihaz kesinti adresi cihaza kesinti önceliği sağlar.(Daha küçük adrese daha öncelik tanınır.)

I-5-2.4 ÇOKLU KESİNTİ SERVİSİ SIRASINDA PROGRAM SÜREKLİLİĞİNİN SAĞLANMASI

Genellikle, çoklu kesinti servisi bulunan sistemde programın sürekliliğini sağlama sorunu, tek kesintinin bulunduğu durumdakine benzer ve fakat, daha karmaşıktır.Özellikle, bir kesintiden diğerine, ondan da bir diğerine geçildiği durumlarda, halihazırda hizmet edilen her bir farklı kesinti isteği için PC dönüş adresi ve mikroişlemcinin tüm önemli yazmaç içerikleri saklanmalıdır. Her kesinti servis kesiti, yalnızca kendine ait olan depolama bölgelerini gerektirecektir.Bilginin saklanarak tekelde edilmesi aşağıdaki üç temel yoldan biri ile yapılır:

1-Program kontrollu saklama ve tekrar elde etme:

Daha öncelikli kesinti isteğine tanınma olanağı vermek için kesinti mask'ı reset edilmeden önce, her kesinti servis programının başlangıcında bir "non-interruptable" kesit içinde ve program kontrolü altında bilgi,belleğin tek bir bölgesine saklanır.Bu bilgi, her servis programı sonunda "non-interruptable" kesittekine benzer olarak

tekrar kullanılmak üzere alınır. Kimi mikroişlemcilerde kesinti isteği tanındığı zaman SP otomatik olarak ilerletilir.Ve program, yığını depolama bölgesi olarak kullanmakla basitleştirilebilir.

2-Otomatik yığma

Herhangi bir kesinti isteği tanındığında kesinti kontrol lojiği otomatik olarak saklanması gereken bilgiyi yığına iter ve SP'yi artırır.Kesinti servisi tamamlandığında,özel amaçlı "kesintiden dön" komutu bilgiyi geri alır ve SP'yi uygun olarak azaltır.

3-Özel amaçlı donanım

Mikroişlemcinin birkaç grup yazmacı ve gösterici yazmacı vardır.Gösterici yazmaç, kesinti isteği tanındığında da azaltılır.Bazı mikroişlecilerde bu yazmaç grupları mikroişlemci entegresinin içindedir.Diğerlerinde ise, bu yazmaçlar yerine bellekte farklı bölgeler kullanılır. Bir kesinti isteğine cevap verme hızı, bir kere istek kabul edildikten sonra, temel olarak halihazırdaki bilgiyi saklama işlemlerine harcanan zamanla saptanır.

I-5-3 DOĞRUDAN BELLEK ERİŞİMLİ I/O

Kimi giriş-çıkış cihazları, mikroişlemci tarafından denetlenen herhangi bir türdeki giriş-çıkış sağlamak için çok hızlı veri iletimi isteyebilirler.Bu durumlarda, bilgi mikroişlecinin müdahalesi olmaksızın I/O cihazı ile bellek arasında doğrudan taşınır. Bu teknik kısaca DMA(direct-memory access) olarak adlandırılır.Veri iletimi mikroişleciden daha yüksek hızda çalışabilen yüksek hızlı bir lojik devre (DMA kontrolörü) ile denetlenir.DMA veri iletimi sırasında,mikroişlemci,kendi belleğinin kontrolünü DMA kontrolörünün üstüne almasına izin vermelidir.Bunun için de birkaç yol vardır:

1-Mikroişlemcinin durdurulması

Dışsal bir kontrol hattı, mikroişlemcinin o anda yerine getirmekte olduğu komutu tamamlamasından sonra, durdurmayı başlatır.Mikroişlemcinin bellek kontrol işaretleri "halt" durumunda zayıflatıldığından DMA kontrolörü bellek denetimini üstüne alarak, veri iletimini başlatır.DMA giriş-çıkışı tamamladıktan sonra "halt" kontrol hattını reset eder ve mikroişlemci normal çalışmasını sürdürerek bir sonraki komutu yerine getirmeye başlar.

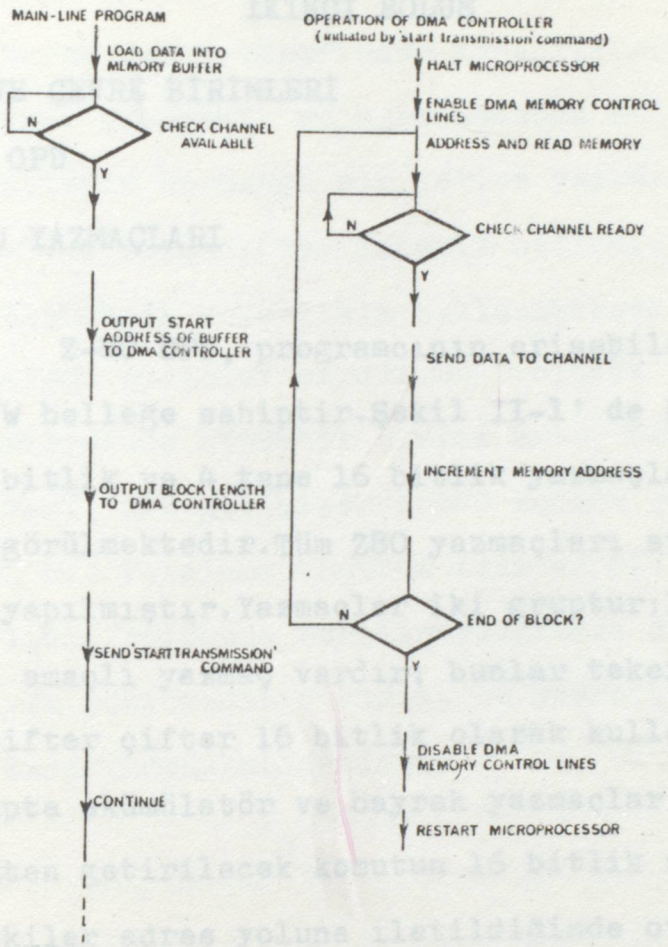
2-Cycle-Steal

Dış kontrol işaretleri komut peryodu içindeki komutun yürütülmesini erteleyerek, mikroişlemcide bir duraklama başlatır.Mikroişlemci saati durdurulur ve bellek kontrol hattı zayıflatılır.

DMA kontrolörü, kontrolü üstüne alır ve veri iletimini gerçekleştirmek için birkaç makina peryodunu 'çalar'.Veri iletişimi tamamlanınca kontrol hattı reset edilir, saat çalışmaya başlar.Komut peryodu yarıda kalan komutu yürütmeyi sürdürür.Komut peryodunu bölmenin tek sonucu, komut yürütme süresinin uzamasıdır.Dinamik bellek kullanın mikroişlemcilerde, çalınabilecek makina peryodu sınırlıdır.Uzun veri blokları giriş-çıkışı, birkaç ayrı makina peryodu daha gerektirecektir.

3-Belleğin ortaklaşa kullanılması

Temel makina peryodu sırasında belli anlarda mikroişlemci belleğe erişebilir.Diğer zamanlarda, diğer cihazlar kullanabilir.DMA kontrolörünün mikroişlemci saati ile senkronize edilmesiyle, DMA veri iletimi, temel makina peryodu içinde normal mikroişlemci-bellek veri iletişimi aynı anda sürdürülebilir.Bu durumda, DMA mikroişlemcinin çalışma hızını etkilemez.



Şekil I-28 DMA veri transferi

İKİNCİ BÖLÜM

II- Z-80 VE ÇEVRE BİRİMLERİ

II-1 Z 80 CPU

II-1.1 CPU YAZMAÇLARI

Z-80 CPU, programcının erişebileceği 208 bitlik bir R/W belleğe sahiptir.Şekil II-1' de bu belleğin, 18 tane 8 bitlik ve 4 tane 16 bitlik yazmaçlar şeklinde düzenlenişi görülmektedir.Tüm Z80 yazmaçları statik RAM kullanılarak yapılmıştır.Yazmaçlar iki gruptur:İlk grupta 6 tane genel amaçlı yazmaç vardır; bunlar teker teker 8 bitlik veya çifter çifter 16 bitlik olarak kullanılabilirler. İkinci grupta akümülatör ve bayrak yazmaçlar vardır.

PC: Bellekten getirilecek komutun 16 bitlik adresini tutar. PC, içindekiler adres yoluna iletildiğinde otomatik olarak artırılır.Atlama olduğunda yeni adres otomatik olarak PC'ye yerleştirilir.

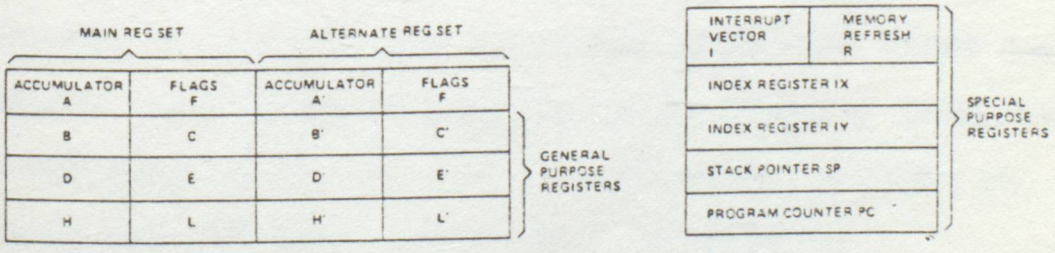
SP: Harici RAM bellekte yığın bölgesinin en üstünün 16 bitlik adresini tutar.Bu yığın, son giren, ilk çıkacak şekilde düzenlemiştir.Veri, PUSH ya da POP komutları ile yığına itilebilir veya yığından çakilebilir.

IX-IY: Bu iki bağımsız index yazmaç, sıralı adresleme modunda kullanılacak 16 bitlik temel adresi tutar.

Kesinti sayfa adres yazmacı I: Z80 CPU ile kesinti isteğini cevaplamak amacıyla, belleğin herhangi bir bölgesine dallanmak olasıdır.I yazmacı dallanmak istenilen adresin

en ağırlıklı 8 bitini tutar, daha az ağırlıklı 8 bit te kesinti isteyen cihaz tarafından gönderilir.Bu özellikle kesinti hizmet programı, en kısa zamanda erişilebilecek şekilde, belleğin herhangi bir yerine yerleştirilebilir.

Bellek Tazeleme Yazmacı(R): Dinamik bellekleri, statik bellekler ölçüsünde kolaylıkla kullanabilmek amacıyla,bir bellek tazeleme sayıcısı vardır.Bu 7 bitlik yazmaç, her komut getirilişinde otomatik olarak artar.Tazeleme sayıcısındaki veri, CPU getirilen komut kodunu çözerken ve yürütürken, bir tazeleme kontrol işareti ile adres yolunun en az ağırlıklı bölümüne verilir.Programcı R yazmacını test amacıyla yükleyebilir ancak, bu yazmaç, normal olarak programcı tarafından kullanılmaz.



Şekil II-1 Z80-CPU yazmaçları

Akümülatör ve Bayrak yazmaçlar:İki tane bağımsız 8 bitlik akümülatör, aritmetik ve lojik işlem sonuçlarını tutar. Her iki bayrak yazmaç da CPU operasyonlarına bağlı olarak set ya da reset edilen 6 bitlik bilgi içerir.Bunlardan 4'ü test edilebilir yani,koşullu atlama, CALL, RETURN komutları için kullanılır.Test edilebilen 4 bit şunlardır:

ELDE BİTİ(C): Bir toplama yapılırken, sonuçta bir elde varsa, C bayrağı set edilecektir.("1" değerini alacaktır.) Çıkarma yapılırken bir ödünç varsa, bu bayrak yine set edilecektir.Ayrıca döndürme ve kaydırma komutları da bu bayrağı etkileyecektir.

SIFIR BİTİ(Z): İşlem sonucunda akümülatördeki değer sıfır ise, bu bayrak 1 değerini alacaktır.Aksi halde, 0 olacaktır.

İŞARET BİTİ(S): İşaretli sayılar için tasarlanmıştır, işlem sonucu negatifse set edilir.Bir sayının 7. biti işaretini gösterdiği için, bu bayrak akümülatördeki sayının 7. biti ile yüklenir.

PARİTY/OVERFLOW(P/V):Bu dual amaçlı bayrak,lojik işlemler yapılırken (AND A B gibi) akümülatördeki sonucun PARİTY durumunu ve işaretli 2's complement aritmetik işlem yapıldığında taşma durumunu gösterir.Lojik işlemler için, akümülatördeki binary sayıdaki 1'lerin sayısı çiftse, bu bayrak set edilecek, tekse reset edilecektir.

HALF CARRY:Toplama veya çıkarma işlemlerinde en az ağırlıklı 4 bitten gelen BCD elde veya ödünçtür.Doğru sonucu bulmak için DAA komutu kullanılabilir.

TOPLAMA/ÇIKARMA BİTİ (N): BCD sistemde toplama veya çıkarma için kullanılan algoritma farklı olduğundan, bu bayrak yapılan en son işlemin ne olduğunu göstermek için kullanılır.Böylece DAA komutu doğru olarak kullanılır.

CPU işlemlerinin bayrak yazmacın her bir bitini nasıl etkilediği komutların ayrıntılı açıklamasında belirtilir.Burada "." işareti, komutun ilgili biti değiştirmediğini, "X" , bayrağın belirsiz durumda olduğunu "0" reset, "1" set olduğunu ve ↕ sembolü de bir önceki sonuca göre set veya reset olabileceğini gösterir.Şekil II-2 de bayrak yazmacı formatı görülmektedir.

D_7							D_0
S	Z	X	H	X	P/V	N	C

Şekil II-2 Z80 bayrak yazmacı formatı

Genel amaçlı yazmaç takımı, 6 tane 8 bitlik B,C,D,E,H,L ve ikinci takım B',C',D',E',H',L' dür.16 bitlik işlemler için bunlar ikili kullanılabilirler.

II-1-2 ALU

CPU'nun 8 bitlik aritmetik-lojik komutları bu birimde yürütülür.ALU, iç ve dış veri yolu ve yazmaçlar ile ilintilidir.Programcı, doğrudan bu birime erişemez. ALU'nun yaptığı işlemler:
Toplama/çıkarma/sağa veya sola kaydırma/karşılaştırma/ VE/VEYA/ExOR/artırma/azaltma/bit testi

II-1-3 Z80 CPU KOMUTLARI

Z80 CPU, 78'i 8080A mikroişlamcisinde de bulunan 158 farklı komutu yerine getirebilir.Bu komutlar aşadaki bölümlere ayrılır:

-Yükleme ve değiştirme
-Blok transferi ve araması
-Aritmetik ve lojik
-Döndürme ve kaydırma
-Bit denetimi(set, reset, test)
-Giriş/Çıkış
-Temel CPU kontrolü

Yükleme komutları, veriyi CPU yazmaçları arasında veya yazmaçlar ile dış bellek arasında taşır.Komutlarda bilginin alınacağı kaynak ile gideceği yer belirtilir.Kaynak bölge, yükleme komutu ile değiştirilemez.Yükleme komutları genel amaçlı yazmaçlar arasında bilgi iletişimini sağlar.CPU yazmaçlarına veya dış bellek arasında doğrudan yükleme de sözkonusudur.Diğer yükleme komutları CPU yazmaçları ile bellek alanı arasında veri iletişimini sağlar.

Z80'de tek bir komutla, belleğin istenilen bir bloğu, başka bir yere taşınabilir.Bu işlem, özellikle büyük veri dizileri ile işlem yapan programlarda önemlidir. Yalnızca tek bir komutla, istenilen boyuttaki bir bellek bölgesi içinde 8 bitlik bir karakter aranabilir.Aranan karakter bulunduğunda veya bloğun sonu geldiğinde komut

kendiliğinden sona erer.Hem blok transferi ve hem de blok arama komutları yürütülürken kesintiye uğrayabilirler.

Aritmetik-lojik komutlar, akümülatördeki genel amaçlı yazmaçlardaki veya da dış bellekteki veri ile işlem yaparlar.İşlem sonuçları akümülatöre yerleştirilir ve sonuca göre uygun bayraklar set edilir.Bu grup, 16 bitlik CPU yazmaçları arasında toplama ve çıkarmayı da içerir.

Bit yönetim komutları, tek bir komut ile, herhangi bir genel amaçlı yazmaçtaki, akümülatördeki veya belleğin herhangi bir yerindeki bir biti, set, reset veya test ederler.

Atlama, çağırma ve geri dönüş komutları, kullanıcının program içinde değişik alanlar arasında hareketini sağlarlar.Bu komut grubu, özel bir dış bellek bölgesinden program sayıcısının yeni adresini almak için birkaç farklı teknik kullanır.Atlama komutunun bir türü RESTART komutudur.Bu komut, 8 bitlik işlem kodunun bir parçası olarak yeni adresi içerir.Program atlaması, HL,IX ve IY içeriklerini doğrudan PC'ye yükleyerek de sağlanabilir.

Giriş/çıkış komutları, CPU'nun genel amaçlı yazmaçları ile dış bellek veya I/O cihazları arasında veri bildirişimini sağlar.I/O işlemleri sırasında,PORT numarası, adres yolunun alt 8 biti ile sağlanır.Seçilecek PORT numarası komutun ikinci baytında verilir.Diğer komutlarda, bu numara C yazmacı içeriği tarafından saptanır.I/O cihazını seçmek için C yazmacını kullanmanın temel avantajı, farklı

I/O cihazlarının ortak yazılım sürücü programını paylaşmalarını sağlamaktır.Bu komutların diğer bir özelliği, giriş verisinin durumunu saptamak için (örneğin, "parity" durumu) ek işlemlere gerek kalmaksızın bayrak yazmacını otomatik olarak set etmeleridir.Tek bir komutla veri blokları otomatik olarak herhangi bir I/O port'undan, herhangi bir bellek bölgesine (tersi de olabilir.) aktarılabilir.Genel amaçlı yazmaçların ikili birleşimlerinde bu komutlar I/O blok transfer hızını artırırlar.

Son olarak, temel CPU kontrol komutları, çeşitli mod ve opsiyonları içerirler.Bu gruptaki komutlar ile kesinti "enable" flip-flop'u sıfırlanabilir veya kesinti cevap modu set edilebilir.

II-1-4 KESİNTİ CEVABI

Kesintinin amacı, CPU işlemlerin durdurarak, CPU'nun bir çevre birimine servis vermesini sağlamaktır. Bu servis programı genellikle, CPU ve çevre birimi arasında veri, durum ve bilgi alışverişi ile ilgilidir.Servis programı tamamlanınca, CPU çalışmasına kaldığı yerden devam eder.Z80'de iki tane kesinti girişi vardır; yazılımla engellenebilir ve yazılımla engellenemez girişler. Bir"non-maskable" kesinti, CPU tarafından her zaman kabul edilecektir.Bu durmda CPU, yürütmesi gereken bir sonraki komutu görmezlikten gelir ve onun yerine, belleğin 0066H bölgesine atlar.Böylece, CPU tam anlamıyla bir RESTART komutu almış gibi davranır fakat, bellekte gittiği bölge,

8 RESTART bölgesinden biri değildir.Bir RESTART, bellekte 0. sayfadaki özel bir adrese atlamadır.

CPU, "maskable" kesintiye üç moddan biri ile cevap vermek üzere programlanabilir:

MOD 0: Bu modla, kesinti isteyen cihaz veri yoluna herhangi bir komut yerleştirebilir,CPU bunu yürütecektir.Kesinti isteyen cihaz, yalnızca tek bir baytlık komut sağlayacağından bu, genellikle, bir RESTART komutu olacaktır.Bu komutu yerine getirmek için gerekli saat peryot sayısı, normal sayıdan 2 fazladır.Bu fazlalık,CPU nun "daisy chain" yapıda öncelik kontrolü yapabilmek amacıyla, yeterli zaman sağlamak için kesinti cevap peryoduna otomatik olarak 2 "bekle" durumu eklemesi nedeniyle meydana gelir.

MOD 1: Bu mod, programcı tarafından seçildiği zaman, CPU belleğin 0038H bölgesine dallanan RESTART komutunu yürüterek cevap verecektir.Bu cevap,"non-mask able" kesinti cevabına benzer ancak, dallanma 0066H bölgesine değildir.

MOD 2: Bu en güçlü kesinti cevap modudur. Kullanıcı, yalnızca 8 bitlik bir komut ile belleğin herhangi bir bölgesine dallanabilir.Programcı, her kesinti servis programı için 16 bitlik bir başlangıç adres tablosu sağlar.Bu tablo, bellekte herhangi bir yere yerleştirilebilir.Kesinti kabul edildiğinde, tablodan istenen servis programının başlangıç adresini elde etmek için, 16 bitlik bir göstericiden oluşturulmalıdır.Bu göstericinin

üst 8 biti I yazmacı içeriğidir.I yazmacı, daha önceden programcı tarafından istenen değerle yüklenmelidir.(LD I,A) Göstericinin alt 8 biti kesinti isteyen cihaz tarafından sağlanır.Aslında, kesinti isteyen cihazın göndermesi gereken sadece 7 bittir; en az ağırlıklı olan bit 0 olmak zorundadır. (Adresler daima çift numaralı bölgelerden başlarlar ve 16 bitlik gösterici iki komşu baytı kullanır.)

Tablodaki ilk bayt, adresin alt 8 bitidir; dolayısıyla programcı, bir kesintiden önce bu tabloyu istenen adreslerle doldurmalıdır.Bu tablo herhangi bir anda programcı tarafından değiştirilebilir.(Eğer, R/W belleğe depolanmışsa)Böylece, farklı çevre birimlerine, farklı programlarla servis verilebilir.

Kesinti isteyen cihaz adresin alt 8 bitini gönderdikten sonra, CPU otomatik olarak PC içeriğini yığına iter ve tablodan başlama adresini alarak, bu adrese atlar. Bu cevap modu, 19 saat peryodu gerektirir.(kesinti isteyen cihazdan alt 8 biti almak için 7, PC içeriğini saklamak için 6, atlama adresini elde etmek için 6)

Z80 çevre birimleri, kesinti kabulü sırasında CPU'ya programlanmış vektörü otomatik olarak sağlayan bir "daisy-chain" kesinti öncelik yapısına sahiptir.

II-1-5 Z80 CPU BACAK TANIMLARI

Şekil II-3'de Z80 CPU bacak bağlantısı görülmektedir.Her bacağın işlevi aşağıda verilmiştir.

A_0-A_{15}(adres yolu): Tri-state çıkış, aktif H.Bu 16 bitlik adres yolu, 64 Kbayta kadar bellek ve I/O cihazları arasında veri bildirişimini sağlar.I/O adresleme, alt 8 biti kullanır.

D_0-D_7(veri yolu): Tri-state giriş/çıkış, aktif H. Çift yönlüdür.

$\overline{M1}$(makina peryodu): Çıkış, aktif L.İşlemcinin o anda bir komut okuğunu gösterir.

$\overline{MREQ}$(memory request): Tri-state çıkış, aktif L.Bu işaret, belleğe yazmak veya bellekten okumak için adres yolunun geçerli bir adres tuttuğunu gösterir.

$\overline{IORQ}$(I/O request): Tri-state çıkış,aktif L.İşlemci bu işaretle, I/O okuma veya yazma işlemleri için, adres yolumun alt yarısının geçerli bir I/O adresi tuttuğunu gösterir. Bir kesinti kabul edilirken $\overline{M1}$ işareti ile birlikte aktiftir.

$\overline{RD}$(read): Tri-state çıkış, aktif L.İşlemcinin bellekten veya bir I/O cihazından veri okamak isteğini belirtir. Adreslenmiş bellek bölgesi ya da I/O cihazı,veri yoluna veri koymak için bu işareti kullanır.

$\overline{WR}$(write):Tri-state çıkış,aktif L.İşlemcinin belleğe veya bir I/O cihazına veri yazmak isteğini belirtir.

$\overline{RFSH}$(refresh):Çıkış, aktif L.Adres yolunun alt 7 bitinin dinamik bellekler için bir refresh(tazeleme)adresi tuttuğunu gösterir.

$\overline{HALT}$: Çıkış aktif L.İşlemcinin halt(duraklama) durumunda olduğunu ve yeniden işleme başlamadan önce,bir "non-maskable" veya "maskable"kesinti beklediğini gösterir.

$\overline{WAIT}$(bekleme): Giriş, aktif L.Bu işaret işlemciye, adreslenmiş bellek bölgesinin veya I/O cihazının veri iletimine hazır olmadığını gösterir.CPU,bu işaret aktif oluncaya kadar beklemeye devam eder.Böylece, farklı hızlarda çalı- I/O cihazı ve bellek ile CPU eşzamanlanır.

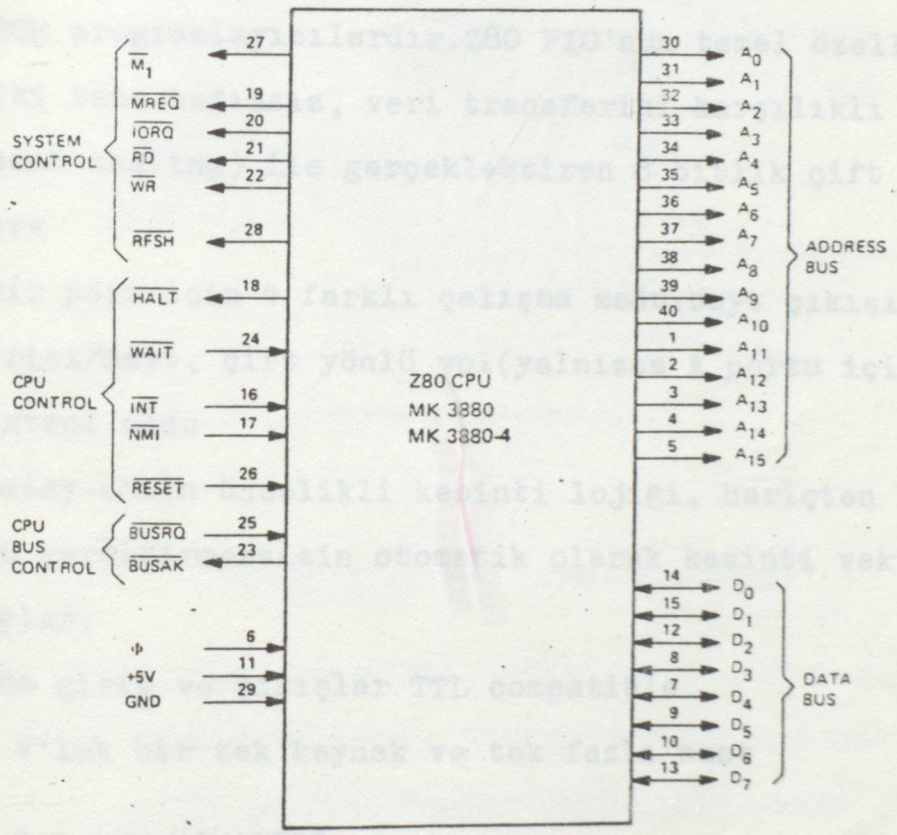
$\overline{INT}$(kesinti): Giriş, aktif L.I/O cihazları tarafından üretilir.BUSRQ işareti aktif değilse ve yazılım kontrollü kesinti flip-flop'u "enable" durumda ise,halihazırdaki komutun tamamlanmasından sonra, bu işaret kabul edilecektir.

$\overline{NMI}$: Giriş, negatif kenar tetiklemeli.En öncelikli kesintidir,kabulü kesinti "enable" flip-flop'unun durumundan bağımsızdır.

$\overline{RESET}$:Giriş,aktif L.Bu işaret, PC,I ve R yazmaçlarını sıfırlar.

$\overline{BUSRQ}$: Giriş,aktif L.Çevre birimlerince gönderilir.CPU-dan adres ve veri yollarını yüksek empedans durumuna getirmesi istenir.

$\overline{BUSAK}$(bus acknowledge):Çıkış, aktif L.Kesinti isteyen cihaza, adres, veri yolu ve tri-state yol kontrol işaretlerinin yüksek empedans durumuna getirildiğini gösterir.



Şekil II-3 Z80 CPU bacak bağlantısı

II-2 Z80 PIO

Paralel I/O devresi, CPU ile çevre birimleri arasında TTL "compatible" arabirim sağlayan, iki portlu, programlanabilir cihazdır.Z80 PIO ile kullanılabilen çevre birimleri, tuş takımları, okuyucular, yazıcılır ve PROM programlayıcılardır.Z80 PIO'nun temel özellikleri:

-İki tane bağımsız, veri transferini karşılıklı anlaşma (hand-shaking) ile gerçekleştiren 8 bitlik çift yönlü port

-Bir port için 4 farklı çalışma modu:bayt çıkışı/bayt girişi/Bayt, çift yönlü yol(yalnızca A portu için)/Bit kontrol modu

-Daisy-chain öncelikli kesinti lojiği, hariçten bir lojik gerektirmeksizin otomatik olarak kesinti vektörünü sağlar.

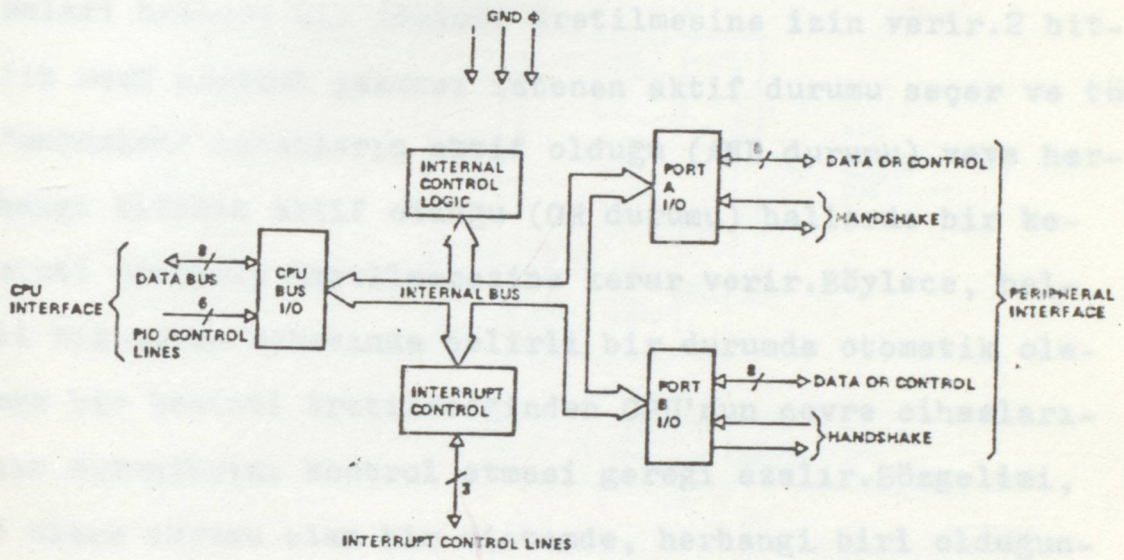
-Tüm giriş ve çıkışlar TTL compatible

-5 V'luk bir tek kaynak ve tek fazlı saat

II-2-1 PIO MİMARİSİ

Z80 PIO blok diyagramı şekil II-4'de görülmektedir.İç yapı, bir Z80 CPU yol arabirimi,iç kontrol lojiği,port A, port B ve kesinti kontrol lojiğinden ibarettir. CPU yol arabirim lojiği, PIO'nun doğrudan CPU'ya bağlanmasına izin verir.Ancak, daha büyük sistemler için adres "decoder"ve/veya hat"buffer"ları gerekebilir.İç kontrol lojiği, CPU veri yolu ile portA ve portB'yi eşzamanlar.

A ve B port'ları tamamen aynıdır ve çevre birimlerine doğrudan bağlanabilir.

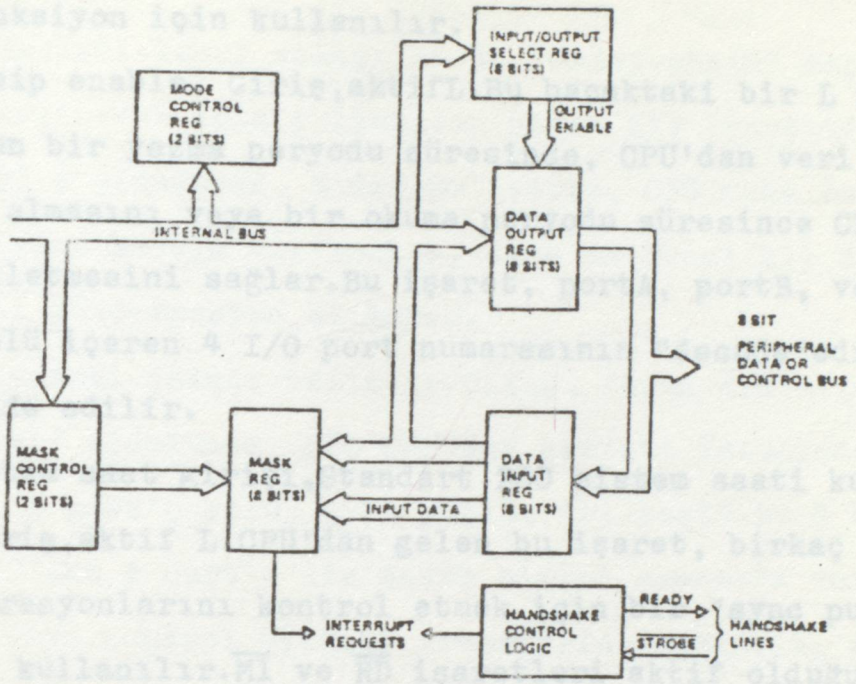


Şekil II-4 PIO blok diyagramı

Port I/O lojiği,karşılıklı anlaşma kontrol lojiği ve 6 yazmaçtan oluşmuştur.(şekil II-5) 2 bitlik mod kontrol yazmacı, istenen çalışma modunu(bayt çıkış, bayt girişi, bayt çift yönlü yol, bit kontrolü)seçmek için CPU tarafından yüklenir.CPU ve çevre birimleri arasındaki tüm veri iletişimi, veri giriş ve veri çıkış yazmaçlarıyla sağlanır.Karşılıklı bildirişim hatları PIO ve çevre birimi arasında veri iletimini kontrol etmek için kullanılır.
8 bitlik "mask"yazmacı ve 8 bitlik giriş/çıkış seçim yazmacı yalnızca bit kontrol modunda kullanılır.Bu modda veri yolu veya kontrol yolu bacakları "select"yazmacın seçimine göre giriş veya çıkış olarak programlanabilir.

Mask yazmacı bu modda belli bir kesinti özelliğine göre kullanılır.Bu özellik,"unmasked" bacakların hepsinin veya herhangi birinin belli bir değere(H veya L) erişmeleri halinde bir kesinti üretilmesine izin verir.2 bitlik mask kontrol yazmacı istenen aktif durumu seçer ve tüm "unmasked" bacakların aktif olduğu (AND durumu) veya herhangi birinin aktif olduğu (OR durumu) hallerde bir kesinti üretilip üretilmemesine karar verir.Böylece, belli bir çevre cihazında belirli bir durumda otomatik olarak bir kesinti üretileceğinden CPU'nun çevre cihazlarının durumlarını kontrol etmesi gereği azalır.Sözgelimi, 3 alarm durumu olan bir sistemde, herhangi biri olduğunda veya üçü birden olduğunda bir kesinti üretilecektir.

Kesinti kontrol lojiği bölümü, CPU'nun kesinti protokolünü kullanır.PIO içinde, portA, portB'ye göre kesinti önceliğine sahiptir.Bit kontrol modunda çevre biriminin durumu, programlanmış bir değere uyduğunda bir kesinti üretilebilir.PIO iç içe geçmiş kesintilerde tam kontrolü sağlar.Yani, daha az öncelikli birim CPU ISR'yi(interrupt service routine) tamamlayana kadar daha çok öncelikli cihazları kesemez.Mod2 ile,CPU bir kesintiyi kabul ettiğinde, kesinti isteyen cihaz 8 bitlik kesinti vektörünü sağlar.A ve B port'ları bağımsız kesinti vektörlerine sahiptir.PIO, CPU veri yolundan aldığı RETI(return from int.) komutun çözer; böylece, PIO, CPU ile başka bir bildirişime gerek kalmaksızın CPU'nun bir ISR yürütüp yürütmediğini bilir.



Şekil II-5 port I/O blok diyagramı

II-2-2 PIO BACAK TANIMLARI

Şekil II-6'da PIO bacak bağlantısı görülmektedir.

D_0-D_7: Çift yönlü, tri-state veri yolu.CPU ile PIO arasındaki tüm veri ve komut iletimleri için kullanılır.

B/$\bar{A}$ Sel:Giriş, aktif H.Port A ve B'yi seçmek için kullanılır.L seviye A port'unu, H seviye B port'unu seçer.Bu seçim için A_0 kullanılır.

C/$\bar{D}$ Sel: Giriş,aktif H.Kontrol veya veri seçimi yapılır. CPU, PIO'ya yazarken bu bacaktaki H seviye, Z80 veri yolundaki bilginin seçilen port tarafından komut olarak algılanmasına neden olur.L seviye de PIO ile CPU arasında veri transfer edildiğini gösterir.Adres yolunun A_1 biti

bu fonksiyon için kullanılır.

$\overline{CE}$: Chip enable. Giriş,aktifL.Bu bacaktaki bir L seviye PIO'nun bir yazma peryodu süresince, CPU'dan veri ya da komut almasını veya bir okuma peryodu süresince CPU'ya veri iletmesini sağlar.Bu işaret, portA, portB, veri ve kontrolü içeren 4 I/O port numarasının "decode"edilmesi ile elde edilir.

ϕ: Sistem saat girişi,Standart Z80 sistem saati kullanı

$\overline{M1}$: Giriş,aktif L.CPU'dan gelen bu işaret, birkaç PIO iç operasyonlarını kontrol etmek için bir "sync pulse" olarak kullanılır.$\overline{M1}$ ve $\overline{RD}$ işaretleri aktif olduğunda Z80 CPU bellekten bir komut getiriyordur.$\overline{M1}$ ile $\overline{IORQ}$ aktif olduğunda, CPU bir kesinti bilgisi alıyordur.

$\overline{M1}$ işaretinin PIO içinde diğer iki işlevi:$\overline{M1}$, PIO kesinti lojiğini senkronize eder ve $\overline{RD}$ ve $\overline{IORQ}$ aktif olmadığı zaman, yalnızca $\overline{M1}$ ile PIO lojiği reset durumuna girer.

IORQ: Giriş,aktif L.CPU'dan gelen giriş/çıkış isteğidir. Bu işaret, PIO ve CPU arasındaki transferler için, $\overline{CE}$, $\overline{RD}$, C/$\overline{D}$,B/$\overline{A}$ işaretleri ile birlikte kullanılır.

RD: Giriş, aktif L.Z80 CPU'dan okuma peryodu durumu.$\overline{RD}$ aktifse, bir bellek veya I/O okuma işlemi yapılıyordur. $\overline{RD}$ işareti, PIO'dan CPU'ya veri iletmek için B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$ ve $\overline{IORQ}$ işaretleri ile birlikte kullanılır.

IEI(int. enable in.) Kesinti kabul girişidir.Giriş aktif H'dır.Birden çok kesinti isteyebilecek cihaz olduğunda kesinti önceliği oluşturmak için kullanılır.H seviye CPU-nun daha öncelikli bir birime kesinti servisi vermediği-

ni gösterir.

IEO: Kesinti kabul çıkışı.Bu da "daisy-chain" yapıda öncelik oluşturmak için kullanılan diğer bir işarettir.Yalnızca, IEI H olduğunda, H'dır.Bu işaret, CPU daha öncelikli bir birime servis veriyorken, daha az öncelikli cihazları kesinti göndermekten alıkoyar.

INT: Kesinti isteği belirtir.Çıkış, "open drain",aktif L. INT aktif olduğunda, PIO CPU'dan kesinti isteğinde bulunur.

A_0-A_7: A port'u yolu, çift yönlü, tri-state.Bu 8 bitlik yol PIO'nun A port'u ile çevre birimi arasında veri ve/veya durum veya kontrol bilgisi transferini sağlar.A_0 en az ağırlıklı bittir.

$\overline{\text{A STB}}$: Bu işaretin anlamı portA tarafından seçilen işlem moduna bağlıdır:

-giriş modu:Çevre birimi, portA giriş yazmacına bilgi yüklemek istediğinde çevre birimi tarafından yayımlanır.Aktif olduğunda PIO'ya veri yüklenir.

-çıkış modu: Bu işaret,bilgiyi alan çevre birimi tarafından bilginin alındığına dair PIO'ya hönderilir.

-çift yönlü mod:İşaret aktif olduğunda, port A çıkış yazmaçlarına gelen veri, çift yönlü veri yoluna konur."Strobe" pozitif kenarı veri alındığını gösterir.

-kontrol modu:"Strobe"yasaklanır:

A RDY: (A yazmacı hazır)Çıkış,aktif H.İşaretin anlamı A port'unun çalışma moduna bağlıdır:

-çıkış modu:Bu işaret, A port'u çıkış yazmacının yüklendiğini ve çevre birimi veri yolunun iletime

hazır olduğunu göstermek amacıyla aktif olur.

-giriş modu:PortA giriş yazmacının boş olduğunu ve çevre biriminden veri almaya hazır olduğunu göstermek amacıyla aktif olur.

-çift yönlü mod:PortA çıkış yazmacındaki verinin çevre birimine iletilmek üzere kullanılabilir durumda olduğunu göstermek için aktif olur.Bu modda, A STB aktif olmadıkça, veri portA'ya yerleştirilmez.

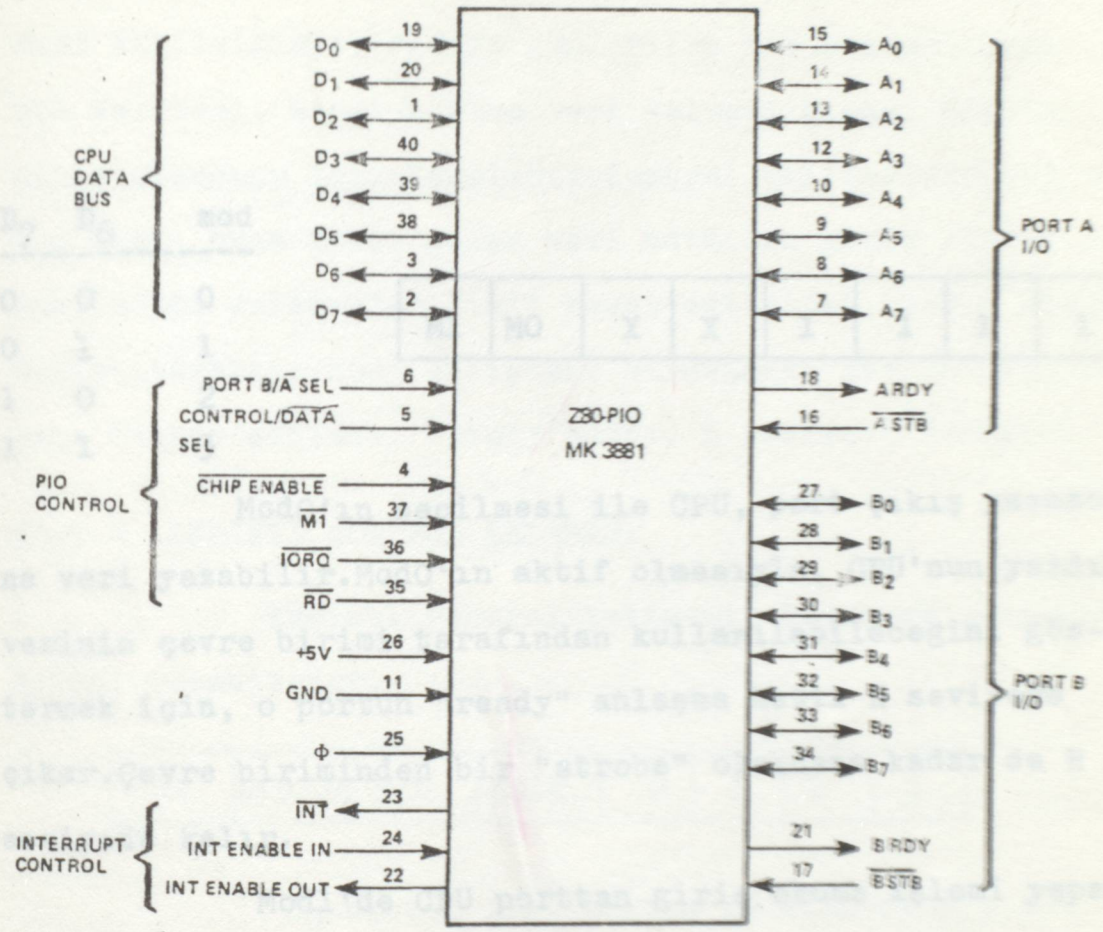
-kontrol modu:Etkisiz hale getirilmiştir.

L seviyede tutulur.

B_0-B_7: Çift yönlü, tri-state, portB yolu.Çevre birimi ve port arasında veri ve/veya durum veya kontrol bilgisi iletmek için kullanılır.

B STB: Çevre biriminden gelir.Giriş,aktif L.Aşağıdaki durum hariç, işlevi A STB ile aynıdır.PortA çift yönlü modda çalıştığı zaman bu işaret, bir çevre biriminden portA giriş yazmacına gelen veriyi "strobe" eder.

B RDY: B yazmacı hazır.Çıkış,aktif H.A RDY'ye benzer, bir farkla; portA çift yönlü olduğu zaman, portA giriş yazmacı boş ve bir çevre biriminden veri almaya hazır olduğunda bu işaret H seviyededir.



Şekil II-6 PIO bacak bağlantısı

II-2-3 ÇALIŞMA MODUNUN SEÇİMİ

A port'u dört farklı modda çalıştırılabilir: Mod0(çıkış modu),Mod1(giriş modu),Mod2(çift yönlü mod), Mod3(kontrol modu).B port'u mod2 hariç diğer modlarda çalışabilir.PIO'ya aşağıdaki formatta bir kontrol kelimesi yazılarak, operasyon modu oluşturulur.

D_7	D_6	mod
0	0	0
0	1	1
1	0	2
1	1	3

M1	MO	X	X	1	1	1	1

Mod0'ın seçilmesi ile CPU, port çıkış yazmacına veri yazabilir.Mod0'ın aktif olmasıyla, CPU'nun yazdığı verinin çevre birimi tarafından kullanılabileceğini göstermek için, o portun "ready" anlaşma hattı H seviyeye çıkar.Çevre biriminden bir "strobe" olaınana kadar da H seviyede kalır.

Mod1'de CPU porttan giriş okuma işlemi yapar. Giriş yazmacına veri yazılabileceğini çevre birimine belirtmek için "ready" hattı aktif kılınır.Çevre birimi, "strobe" hattını kullanarak, portun giriş yazmacına veri- "strobe" eder."Strobe" işaretinin yükselen kenarı(0→1) eğer INT "enable" durumda ise, bir kesinti isteğine neden olur, "ready" hattı pasif hale getirilir.

Mod2, tüm 4 anlaşma hattını kullanan, çift yönlü veri transfer modudur.Mod2, portA'nın anlaşma işaretlerini çıkış kontrolü, portB anlaşma işaretlerini de giriş kontrolü için kullanır.Böyece, A RDY ve B RDY aynı anda aktif olabilirler.

Mod3, anlaşma işaretlerini kullanmaz.

Bu mod, durum ve kontrol uygulamaları için kullanılır. Mod3 seçildiğinde, PIO'ya gönderilen bir sonraki kontrol kelimesi, hangi portun veri yolunun çıkış, hangisinin giriş olduğunu tanımlamalıdır.Kontrol kelimesinin set edilen biti, buna karşı gelen veri hattının giriş olarak kullanılacağı anlamındadır.Bit reset edilmişse, çıkış olarak kullanılacaktır.Mod3 çalışması sırasında "strobe" işaretine dikkat edilmez, "Ready"hattı, L seviyede tutulur.

II-2-4 KESİNTİ KONTROL KELİMESİ

Kesinti kontrol kelimesi, her port için aşağıdaki formattadır.

EI	AND/OR	H/L	MASK	0	1	1	1

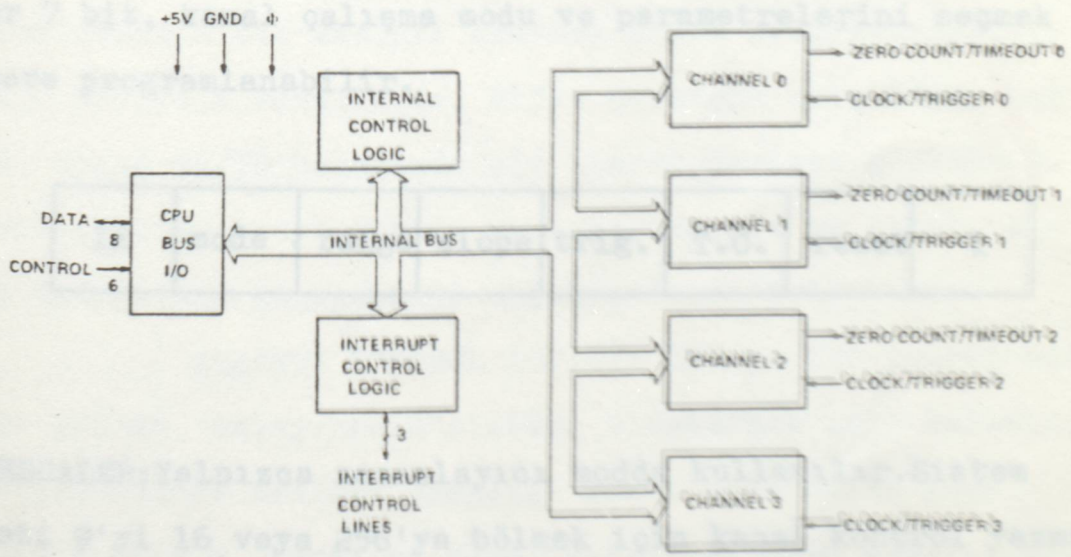
D_7 biti 1 ise, portun IE flip-flop'u set edilmiştir ve port bir kesinti üretecektir.D_7 biti 0 iken bir kesinti olursa, PIO tarafından içerde kilitlenecek, PIO kesintiyi yeniden "enable" durumuna getirdiğinde CPU ya iletecektir.D_6 biti 1 ise AND değilse, OR fonksiyonu seçilecektir.D_5 biti hangi lojik seviyenin aktif kabul edileceğini gösterir.D_4 biti lojik 1 seviyede ise ve mod3'te çalışılıyorsa, bir sonraki kontrol baytı PIO tarafından "mask" baytı olarak değerlendirilecektir. D_4 0 ise, kesinti isteği belirtilecektir.D_3-D_0 bitlerinin 0111 şeklinde olması halinde,PIO,kontrol baytını kesinti kontrol kelimesi olarak algılar.

II-3 Z80 CTC

Z80-CTC, Z80-CPU tabanlı mikrokompüter sistemleri için "zamanlama" ve "sayma" fonksiyonlarını yerine getiren, 4 bağımsız kanallı programlanabilen bir bileşendir.

II-3-1 CTC İÇ YAPISI

Z80-CTC blok diyagramı şekil II-7'de görülmektedir.İç yapısında, Z80-CPU yol arabirimi, iç kontrol lojiği, dört sayıcı/zamanlayıcı kanal ve kesinti kontrol lojiği vardır.4 bağımsız sayıcı/zamanlayıcı kanal 0'dan 3'e ardışıl sayılarla belirtilir.CTC her bir kanal için bir kesinti vektörü üretme yeteneğine sahiptir.0 numaralı kanal en fazla önceliğe sahiptir.CPU yol arabirim lojiği CTC'nin doğrudan CPU'ya bağlanmasını sağlar.



Şekil II-7 Z80-CTC blok diyagramı

KANAL LOJİK YAPISI:

Bir kanalın iç yapısı şekil II-8'de görülmektedir.2 yazmaç, 2 sayıcı ve kontrol lojikten ibarettir.Sayıcılar 8 bitlik(CPU tarafından okunabilir.)geriye sayıcı ve "prescaler"dir.8 bitlik kanal kontrol yazmaç lojiği, modları ve kanal parametrelerini seçmek üzere CPU tarafından yazılır.CTC entegresi içinde, her bir kanal için böyle 4 yazmaç vardır.CPU'nun A_0 ve A_1 adres yolu kullanılarak, CSO ve CS1 girişleriyle bu kanallar seçilir.

CSO	CS1	Kanal
0	0	0
1	0	1
0	1	2
1	1	3

Her kanal kontrol yazmacını programlamak için yazılan kontrol kelimesinde 0. bit daima set edilir ve diğer 7 bit, kanal çalışma modu ve parametrelerini seçmek üzere programlanabilir.

IE	mode	range	slope	trig.	T.C.	reset	1

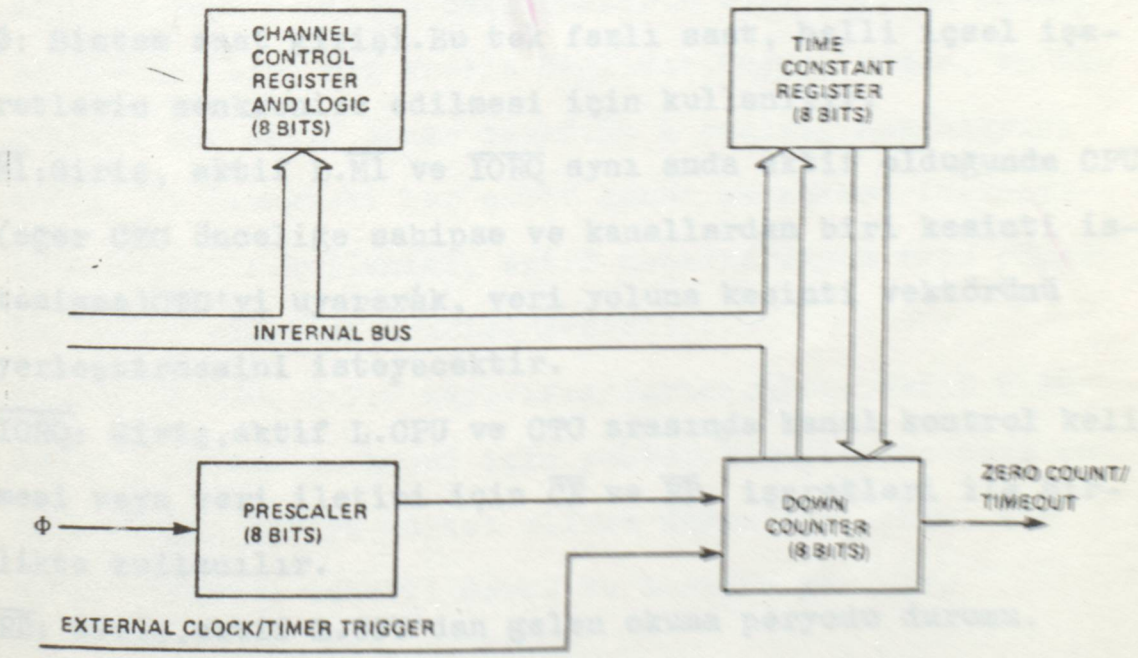
PRESCALER:Yalnızca zamanlayıcı modda kullanılır.Sistem saati ϕ'yi 16 veya 256'ya bölmek için kanal kontrol yazmacı vasıtasıyla CPU tarafından programlanabilir.8 bitliktir.

ZAMAN SABİTİ YAZMACI:Hem zamanlayıcı, hem sayıcı modda kullanılan 8 bitlik bir yazmaçtır.CPU tarafından programlanabilir.Kanal kontrol kelimesi 1'den 256'ya kadar herhangi bir zaman sabiti değerini alabilir.Zaman sabiti yazmacı, programlanmiş bu değeri CTC ilk çalışmaya başladığında geriye sayıcıya yükler.Geriye sayıcı 0 değerine aldığında, programlanmış olduğu değerle otomatik olarak yeniden yüklenir.Kanal, sayma veya zamanlama modunda çalışıyorken, zaman sabiti yazmacına yeni bir zaman sabiti yüklenirse, bu değerin geriye sayıcıya yüklenmesi için bir önceki sayının sıfır olması beklenir.

GERİYE SAYICI: 8 bitliktir, her iki modda da kullanılır ve zaman sabiti yazmacı tarafından yüklenir.Geriye sayıcı her iki modda, Prescaler çıkış saat darbesinin her kenarında bir azaltılır.CPU basit bir I/O işlemi ile bu yazmacın içeriğine erişebilir.Herhang bir CTC kanalı, her sıfıra erişilmesinde bir kesinti üretmek üzere programlanabilir. 0,1,2 numaralı kanallarda, sıfır durumuna erişildiği her kez, uygun ZC/TO bacağında bir işaret darbesi görünür.3. kanal için bu bacak yoktur ve dolayısı ile bu kanal, çıkış darbesinin gerekmediği uygulamalarda kullanılır.

KESİNTİ KONTROL LOJİĞİ: IEI ve IEO işaret hatları CTC'de "daisy-chain"sistemi oluşturmak için kullanılır.CTC içinde kesinti önceliği kanal numarasına göre saptanmıştır.0. kanal en önceliklidir.Bir CTC kanalı, geriye sayıcının her sıfırı sayışında bir kesinti istemek üzere programlanabilir.(Bu özelliği kullanmak için CPU

kesinti modu 2'ye göre programlanmalıdır.)Kesinti kabul işaretinden sonra, CTC kesinti kontrol lojiği de CTC içinde en yüksek öncelikli kanalı saptayacaktır.Daha sonra eğer CTC'nin IEI girişi aktifse(ki bu, en öncelikli durumda kendisinin olduğunu gösterir.)8 bitlik kesinti vektörü veri yoluna yerleştirilecektir.Bu vektörün en ağırlıklı 5 biti daha önceden CTC'ye yazılır.1 ve 2 numaralı bitlerde ise, kanalların kesinti önceliğini belirten binary kod bulunur. Vektörün sıfırıncı biti sıfır olmak zorundadır.Kesinti vektörü, bellekte ISR'nin depolandığı bölgeyi göstermek için kullanılır.CPU, ISR'nin depolandığı bölgenin 16 bitlik başlangıç adresinin üst sekiz bitini I yazmacını okuyarak anlar, alt sekiz bitini ise, bu vektör sağlar.



Şekil II-8 Kanal blok diyagramı

II-3-2 CTC BACAK TANIMLARI

CTC bacak bağlantısı şekil II-9'da görülmektedir.Tanımları:

D_0-D_7: Z80 CPU veri yolu, çift yönlü, tri-state.

CS0-CS1:Giriş,aktif H.I/O ve R/W işlemi için dört bağımsız kanalı seçmek için kullanılır.

$\overline{CE}$: (Chip enable)Giriş,aktif L.Bu bacaktaki bir L seviye CTC'nin kontrol kelimelerini kabul etmesini sağlar.Bir I/O yazma peryodu süresince CPU'dan kesinti vektörleri, zaman sabitini veren veri kelimeleri alınır veya bir okuma peryodunda, işlemciye geriye sayıcı içeriği gönderilir.Çoğu uygulamarda bu işaret, adres yolunun 8 biti çözülerek elde edilir.

ϕ: Sistem saat girişi.Bu tek fazlı saat, belli içsel işaretlerin senkronize edilmesi için kullanılır.

$\overline{M1}$:Giriş, aktif L.$\overline{M1}$ ve $\overline{IORQ}$ aynı anda aktif olduğunda CPU-(eğer CTC önceliğe sahipse ve kanallardan biri kesinti istemişse)CTC'yi uyararák, veri yoluna kesinti vektörünü yerleştirmesini isteyecektir.

$\overline{IORQ}$: Giriş,aktif L.CPU ve CTC arasında kanal kontrol kelimesi veya veri iletimi için $\overline{CE}$ ve $\overline{RD}$ işaretleri ile birlikte kullanılır.

$\overline{RD}$: Giriş,aktif L.CPU'dan gelen okuma peryodu durumu.

IEI:Giriş,aktif H.Bu bacaktaki bir H seviye, CPU'nun CTC den daha öncelikli bir birime servis vermediğini gösterir.

IEO: Çıkış,aktif H.Yalnızca, IEI H olduğunda H'dır.

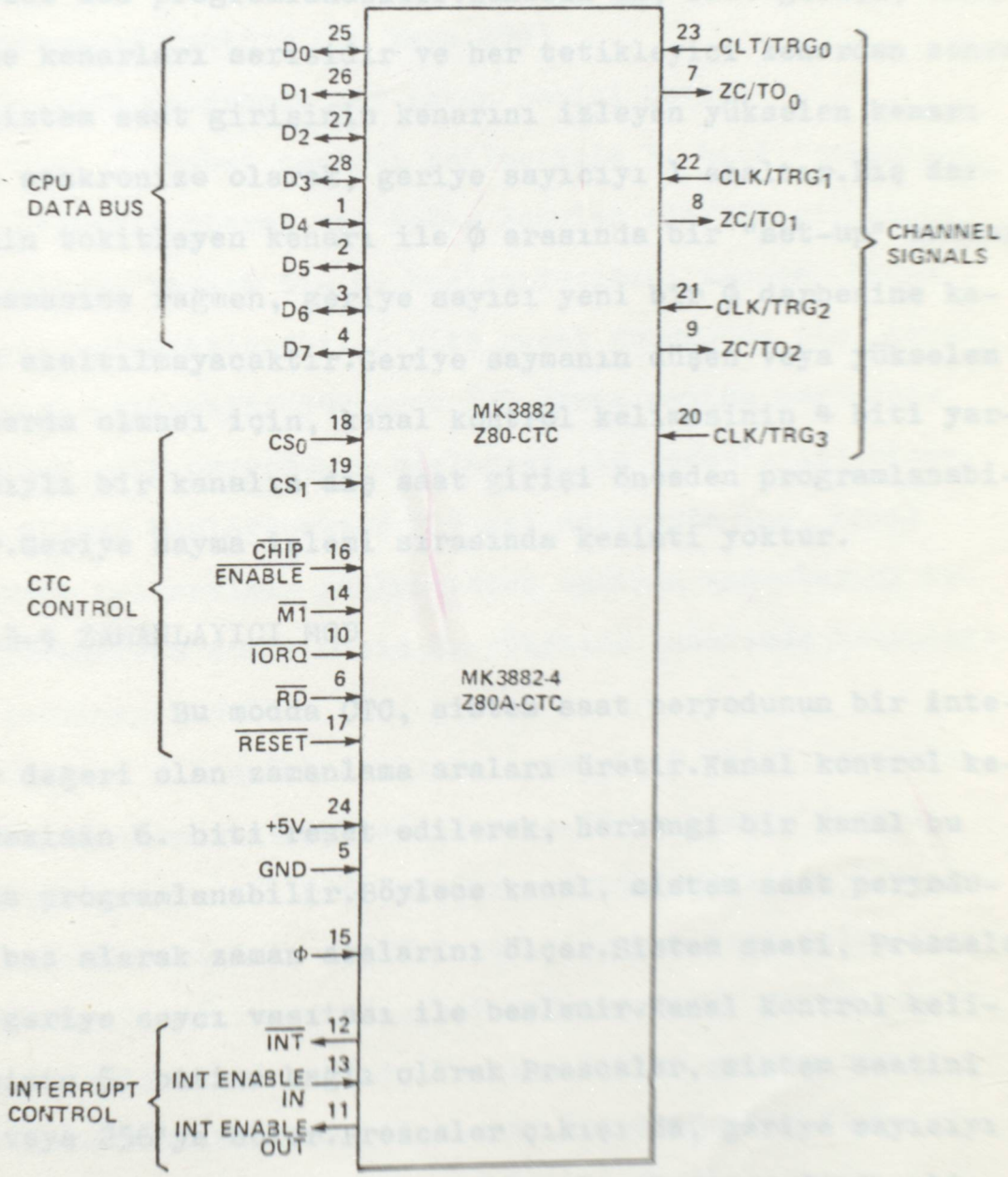
Bu işaret, CPU daha çok öncelikli bir birimin ISR programını yürütüyorken, daha az öncelikli bir birimi kesinti istemekten alıkoyar.

$\overline{\text{INT}}$: Çıkış,"open drain", aktif L.CTC'nin herhangi bir kanalıgeriye sayıcıdaki sıfır-sayı durumu için kesinti istemek üzere programlanmışsa, bu işaret kullanılır.

$\overline{\text{RESET}}$:Giriş,aktif L.Tüm kontrol yazmaçlarındaki kanal kesinti "enable" bitlerini reset eder,tüm kanalların sayma işlemini durdurur.Böylece, CTC'nin üreteceği kesintiler "disable" edilir.ZC/TO ve INT çıkışları aktif olmazlar, IEO, IEI'yı yansıtır ve CTC veri yolu çıkış sürücüsü yüksek empedans durumuna girer.

CLK/TRG3-CLK/TRG0: Harici "clock/timer" tetiklemesi.Giriş aktif H veya L olarak seçilebilir.4 tane CLK/TRG bacağı vardır.Her biri bir kanala bağlıdır.Sayıcı modda, bu bacaktaki her aktif kenar sayıcıyı 1 azaltır.Zamanlayıcı modda, bu bacaktaki her aktif kenar zamanlama fonksiyonunu başlatır.Kullanıcı, aktif kenarın düşen veya yükselen kenar olmasını kendisi seçebilir.

ZC/TO2-ZC/TO0: Sıfır sayı/zamanlayıcı.Giriş,aktif H.Bu bacak yalnızca 3. kanal için yoktur.Zamanlayıcı veya sayıcı modda, geriye sayıcı sıfıra doğru azalırken, bir aktif yükselen kenarlı darbe bu bacakta görünür.



Şekil II-9 Z80 CTC bacak bağlantısı

II-3-3 SAYICI MODU

Bu modda CTC,"clock/trigger" girişi kenarlarını sayar.Kanal kontrol kelimesinin 6. biti set edilerek sayıcı mod programlanabilir.Kanalın dış saat girişi, tetikleme kenarları serisidir ve her tetikleyici kenardan sonra Ø sistem saat girişinin kenarını izleyen yükselen kenarı ile senkronize olarak, geriye sayıcıyı 1 azaltır.Dış darbenin tekitleyen kenarı ile Ø arasında bir "set-up" zamanı olmamasına rağmen, geriye sayıcı yeni bir Ø darbesine kadar azaltılmayacaktır.Geriye saymanın düşen veya yükselem kenarda olması için, kanal kontrol kelimesinin 4 biti yardımıylı bir kanalın dış saat girişi önceden programlanabilir.Geriye sayma işlemi sırasında kesinti yoktur.

II-3-4 ZAMANLAYICI MOD

Bu modda CTC, sistem saat peryodunun bir integer değeri olan zamanlama araları üretir.Kanal kontrol kelimesinin 6. biti reset edilerek, herhangi bir kanal bu moda programlanabilir.Böylece kanal, sistem saat peryodunu baz olarak zaman aralarını ölçer.Sistem saati, Prescaler ve geriye saycı vasıtası ile beslenir.Kanal kontrol kelimesinin 5. bitine bağlı olarak Prescaler, sistem saatini 16 veya 256'ya böler.Prescaler çıkışı da, geriye sayıcıyı saydırmak için saat olarak kullanılır.Sayma modunda olduğu gibi, sıfıra ulaşılınca, başlangıç değeri geriye sayıcıya otomatik olarak yeniden yüklenir.Sıfır sayısında kanalın ZC/TO çıkışı (ki bu, geriye sayıcının çıkışıdır)

tetiklenir.Bu da;

$$t_c \times P \times TC$$

çarpımını sağlayan peryotta darbe katarı meydana getirir.

t_c:ϕ peryodu

P:Prescaler faktörü (16 veya 256)

TC:Önceden programlanmış zaman sabiti değeri

Kanal kontrol kelimesinin 3. biti, zamanlamanın otomatik olarak mı, yoksa kanalın CLK/TRG girişinin tetikleyen kenarı ile mi başlatılacağını seçmek üzere önceden programlanabilir.Bit3 reset durumanda ise, kanala kontrol kelimesini yükleyen I/O yazma makina peryodunu takiben, CPU peryodunun başlaması ile zamanlayıcı otomatik olarak başlayacaktır.Bi3 set durumunda ise, kanal kontrol kelimesinin yüklenmesini takiben zamanlayıcı tetiklemesinden sonra O'nin 2. yükselen kenarında zamanlayıcı çalışmaya başlar.

Kanal kontrol kelimesinin 4. biti, zamanlayıcı tetiklemesinin yükselen veya düşen kenara duyarlı olmasını sağlamak için önceden programlanabilir. Zamanlayıcı tetiklemesinin aktif kenarı ile peşisıra gelen O'nin yükselen kenarı arasında "set-up" zamanına gerek yoktur. Kanal kontrol kelimesinin 7. biti 1 ise, geriye sayıcı O'ı sayarken, kanalın "time-out" bacağında bir darbe görünmesi yanısıra, bir kesinti isteği üretilecektir.

II-4 YAZILIM KONTROLLÜ SAAT

Program yapılırken tüm zamanlama, sistem saati üzerine temellendirilmiştir.(1.789772MHz) Çevrimlerdeki toplam peryot sayısı hesaplanmıştır:

CLEAR: 1041 T

MSG : 4956 T

GÖSTER: 1747 T

SCANl : 28898 T

ANDEĞ: 258 T

Toplam sayı 1800610 olup; 0.56X1800610=1.008 s

Kullanılan altprogramların işlevleri:

MSG: Giriş buffer'da depolanan ASCII kodları display pattern'lerine dönüştürür ve bir CR(carriage return) tuşuna basılana kadar pattern'leri display buffer'larına yerleştirir.Giriş buffer'ı için gösterici olarak HL çifti kullanılır.

CLR : Display buffer'ı temizler ve display buffer ve giriş buffer göstericisini başlangıç adreslerine set eder. Yani display buffer içerikleri FF değerine set edilir.

READLN: Bir CR'e rastlayıncaya kadar karakterleri okur.

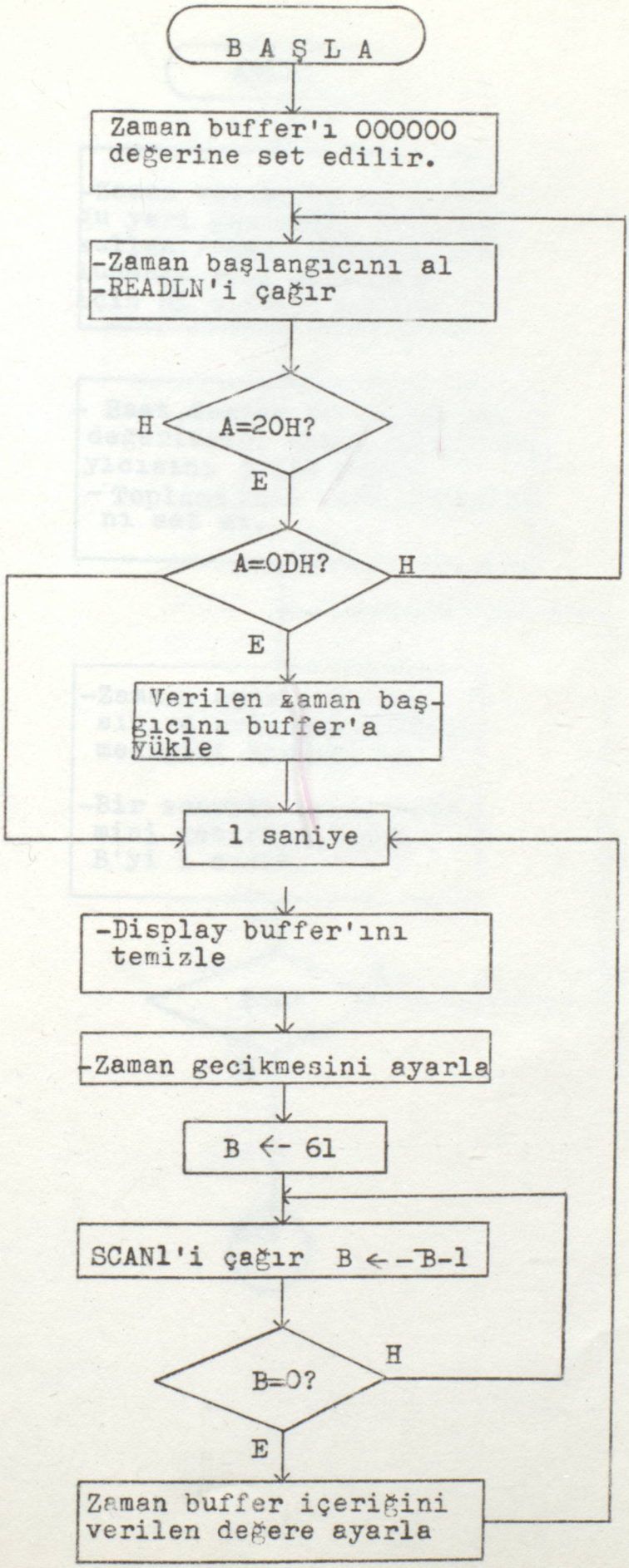
GETHL: HL çiftini gösterici olarak kullanarak ASCII kodları hex'e çevirir ve HL çiftinde depolar.

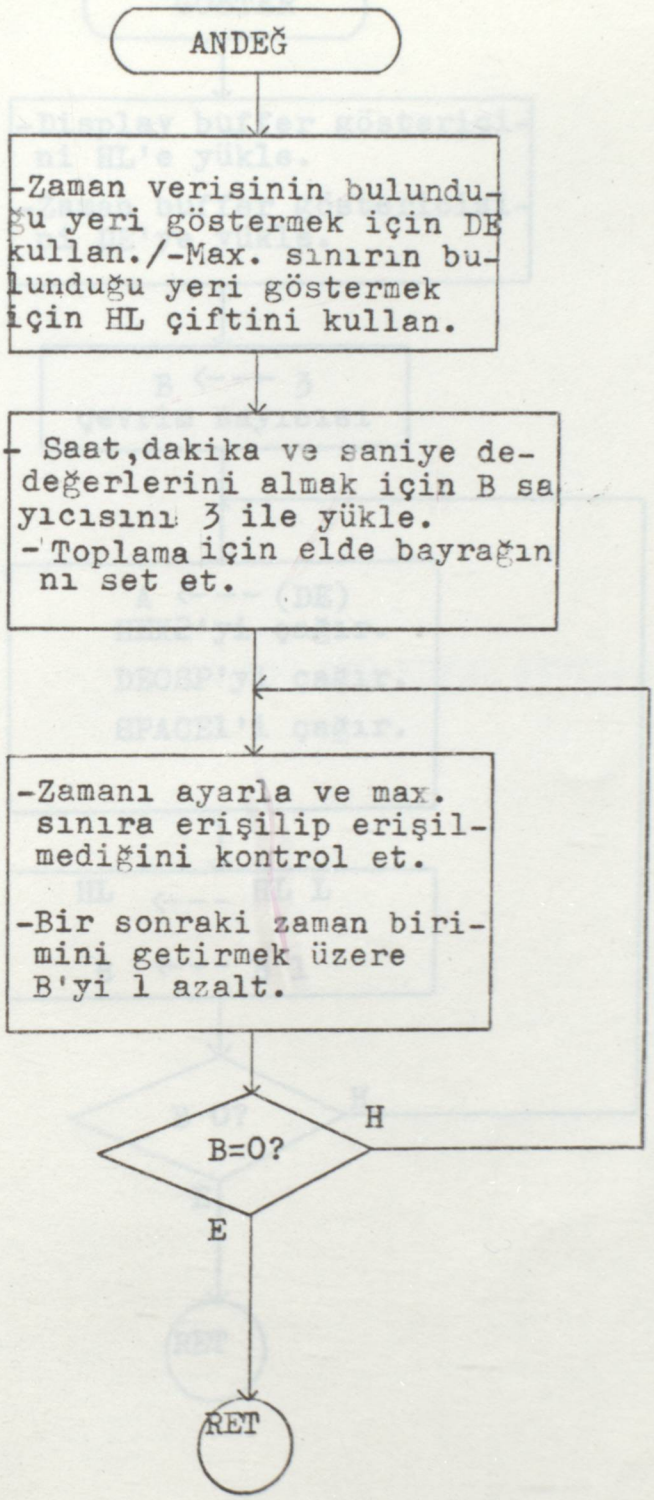
SCANl : Keyboard ve display'i tarar.Bir tarama sırasında tuşa basılmamışsa, CF=1, basılmışsa CF=O'dır.

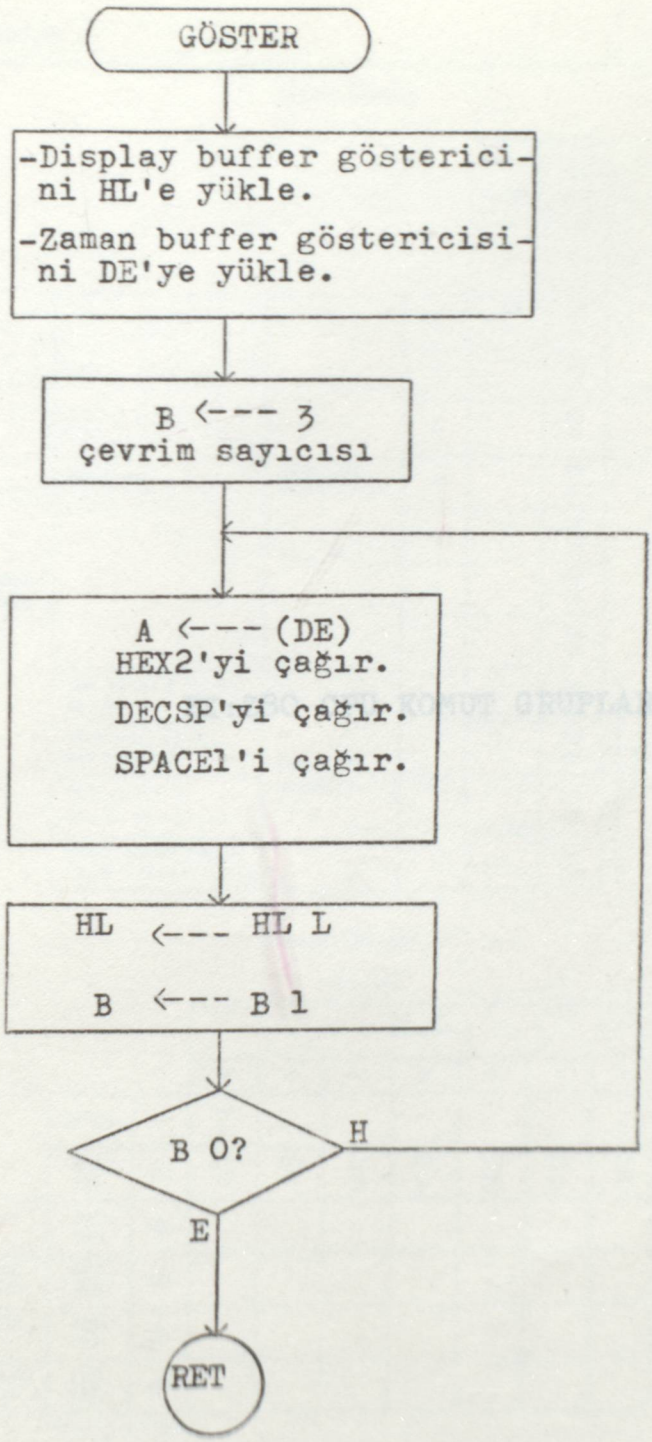
YAZILIM KONTROLLÜ SAAT PROGRAMI

```
         CALL    CLEAR
         LD      B,3
ÇEV1     LD      HL,SAAT
         LD      (HL),0          ;zaman buffer başlan-
         INC     HL               gıç değeri
         DJNZ    ÇEV1
         LD      HL,FORMAT
         CALL    MSG
         CALL    READLN          ;Zaman set değerinin
         JR      Z,HESAP          alınması.Girişte CR
         CALL    CHKHEX           varsa,HESAP'a atla,
         LD      B,3              yoksa, saati, dakikayı
         LD      HL,SAAT          saniyeyi al.
ÇEV2     PUSH    HL
         CALL    GETHL
         POP     HL
         LD      (HL),A
         INC     HL
         DJNZ    ÇEV2
HESAP    CALL    CLEAR
         CALL    GÖSTER
         LD      B,62
         LD      IX,DISPBF
ÇEV3     CALL    SCAN1
         DJNZ    ÇEV3
         CALL    ANDEĞ
         JR      HESAP
ANDEĞ    LD      HL,SINIR+2
         LD      DE,SAN
         PUSH    BC
         LD      B,3
         SCF                     ;Elde bayrağı set edilir.
ARTMA    LD      A,(DE)
         ADC     A,0
         DAA
         LD      (DE),A
         SUB     (HL)            ; Sonuç SINIR'daki
         JR      C,COMPL          veri ile karşılaş-
         LD      (DE),A           tırılır; ondan kü-
COMPL    CCF                      çük ise, sonraki
         DEC     HL               çevrimin anlamı kal-
         DEC     DE               mayacaktır.
         DJNZ    ARTMA
         POP     BC
         RET
GÖSTER   LD      HL,DISPBF+12
         LD      (DISP),HL
         LD      DE,TMBF
         LD      B,3
ÇEV4     LD      A,(DE)
```

```
        CALL    HEX2
        CALL    DECSP
        CALL    SPACE1
        INC     DE
        DJNZ    ÇEV4
        CALL    DECSP
        RET
TMBF:
SAAT    DEFS    1
DAK     DEFS    1
SAN     DEFS    1
SINIR   DEFB    24H
        DEFB    60H
        DEFB    60H
FORMAT  DEFM    "ZAMAN BAŞ "
        DEFB    0DH
CLEAR   EQU     09B9H
CURSOR  EQU     0A79H
DISP    EQU     0FF84H
MSG     EQU     09CAH
DECSP   EQU     0399H
CHKHEX  EQU     08DFH
DISPBF  EQU     0FF2CH
HEX2    EQU     0A9AH
READLN  EQU     09D4H
GETHL   EQU     08E5H
SCAN1   EQU     029BH
SPACE1  EQU     0A95H
```


B A Ş L A
Zaman buffer'ı 000000 değerine set edilir.
-Zaman başlangıcını al
-READLN'i çağır
A=20H?
H
E
A=0DH?
H
E
Verilen zaman başlangıcını buffer'a yükle
1 saniye
-Display buffer'ını temizle
-Zaman gecikmesini ayarla
B ← 61
SCAN1'i çağır B ← -B-1
B=0?
H
E
Zaman buffer içeriğini verilen değere ayarla


ANDEĞ
-Zaman verisinin bulunduğu yeri göstermek için DE kullan./-Max. sınırın bulunduğu yeri göstermek için HL çiftini kullan.
- Saat,dakika ve saniye değerlerini almak için B sayıcısını 3 ile yükle.
- Toplama için elde bayrağını set et.
-Zamanı ayarla ve max. sınıra erişilip erişilmediğini kontrol et.
-Bir sonraki zaman birimini getirmek üzere B'yi 1 azalt.
B=0?
H
E
RET


GÖSTER
-Display buffer göstericini HL'e yükle.
-Zaman buffer göstericisini DE'ye yükle.
B <--- 3
çevrim sayıcısı
A <--- (DE)
HEX2'yi çağır.
DECSP'yi çağır.
SPACE1'i çağır.
HL <--- HL L
B <--- B 1
B 0?
H
E
RET

EK:Z80 CPU KOMUT GRUPLARI

Input and Output Groups

Input Group

INPUT DESTINATION				PORT ADDRESS: IMMED. n	PORT ADDRESS: REG. INDIR. (C)
INPUT 'IN'	REGISTER ADDRESSING	A		DB n	ED 78
		B			ED 40
		C			ED 48
		D			ED 50
		E			ED 58
		H			ED 60
		L			ED 68
'INI'-INPUT & Inc HL, Dec B	REGISTER INDIRECT	(HL)			ED A2
'INIR'-INP, Inc HL, Dec B, REPEAT IF B ≠ 0					ED B2
'IND'-INPUT & Dec HL, Dec B					ED AA
'INDR'-INPUT, Dec HL Dec B, REPEAT IF B ≠ 0					ED BA

BLOCK INPUT COMMANDS

Output Group

	PORT DESTINATION ADDRESS: IMMED.	n	SOURCE: REGISTER A	B	C	D	E	H	L	REG. IND. (HL)
'OUT	IMMED.	n	D3 n							
	REG. IND.	(C)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69	
'OUTI'-OUTPUT Inc HL Dec b	REG. IND.	(C)								ED A3
'OTIR'-OUTPUT, Inc HL, Dec B, REPEAT IF B ≠ 0	REG. IND.	(C)								ED B3
'OUTD'-OUTPUT Dec HL Dec B	REG. IND.	(C)								ED AB
'OTDR'-OUTPUT, Dec HL Dec B, REPEAT IF B ≠ 0	REG. IND.	(C)								ED BB

BLOCK OUTPUT COMMANDS

Call and Return Groups and Restart

Call and Return Group

			CONDITION									
			UN COND.	CARRY	NON CARRY	ZERO	NON ZERO	PARITY EVEN	PARITY ODD	SIGN NEG.	SIGN POS.	REG. B ≠ 0
'CALL'	IMMEDIATE EXTENSION	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
RETURN 'RET'	REGISTER INDIRECT	(SP) (SP + 1)	C9	D8	D0	C8	C0	E8	E0	F8	F0	
RETURN FROM INT 'RETI'	REGISTER INDIRECT	(SP) (SP + 1)	ED 4D									
RETURN FROM NON MASKABLE INT 'RETN'	REGISTER INDIRECT	(SP) (SP + 1)	ED 45									

Note: Certain flags have more than one purpose.
Refer to the Z80 CPU Technical Manual for details.

Restart Group

		OP CODE	
CALL ADDRESS	0000_H	C7	'RST 0'
	0008_H	CF	'RST 8'
	0010_H	D7	'RST 16'
	0018_H	DF	'RST 24'
	0020_H	E7	'RST 32'
	0028_H	EF	'RST 40'
	0030_H	F7	'RST 48'
	0038_H	FF	'RST 56'

Mnemonic	Symbolic Operation	Flags S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
CALL nn	(SP – 1) ← PC_H (SP – 2) ← PC_L PC ← nn	•	•	X	X	•	•	•	11 001 101 – n – – n –	CD	3	5	17	
CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn	•	•	X	X	•	•	•	11 cc 100 – n – – n –		3	3	10	If cc is false
											3	5	17	If cc is true
RET	PC_L ← (SP) PC_H ← (SP + 1)	•	•	X	X	•	•	•	11 001 001	C9	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	X	X	•	•	•	11 cc 000		1	1	5	If cc is false
											1	3	11	If cc is true
RETI	Return from interrupt	•	•	X	X	•	•	•	11 101 101 01 001 101	ED 4D	2	4	14	
RETN[1]	Return from non-maskable interrupt	•	•	X	X	•	•	•	11 101 101 01 000 101	ED 45	2	4	14	
RST p	(SP – 1) ← PC_H (SP – 2) ← PC_L PC_H ← 0 PC_L ← p	•	•	X	X	•	•	•	11 t 111		1	3	11	

cc: 000 NZ non-zero; 001 Z zero; 010 NC non-carry; 011 C carry; 100 PO parity odd; 101 PE parity even; 110 P sign positive; 111 M sign negative

t: 000 00H; 001 08H; 010 10H; 011 18H; 100 20H; 101 28H; 110 30H; 111 38H

NOTE: RETN loads IFF_2 → IFF_1

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown, ↕ = flag is affected according to the result of the operation

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
BIT b, r	$Z \leftarrow \overline{r_b}$	X	↕	1	X	X	0	• 11 001 011 01 b r	CB	2	2	8	r: Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
BIT b, (HL)	$Z \leftarrow \overline{(HL)_b}$	X	↕	1	X	X	0	• 11 001 011 01 b 110	CB	2	3	12	
BIT b, (IX + d)	$Z \leftarrow \overline{(IX + d)_b}$	X	↕	1	X	X	0	• 11 011 101 11 001 011 – d – 01 b 110	DD CB	4	5	20	
BIT b, (IY + d)	$Z \leftarrow \overline{(IY + d)_b}$	X	↕	1	X	X	0	• 11 111 101 11 001 011 – d – 01 b 110	FD CB	4	5	20	b: Bit Tested 000 0 001 1 010 2 011 3 100 4 101 5 110 6 111 7
SET b, r	$r_b \leftarrow 1$	•	•	X	•	X	•	• 11 001 011 11 b r	CB	2	2	8	
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	X	•	X	•	• 11 001 011 11 b 110	CB	2	4	15	
SET b, (IX + d)	$(IX + d)_b \leftarrow 1$	•	•	X	•	X	•	• 11 011 101 11 001 011 – d – 11 b 110	DD CB	4	6	23	
SET b, (IY + d)	$(IY + d)_b \leftarrow 1$	•	•	X	•	X	•	• 11 111 101 11 001 011 – d – 11 b 110	FD CB	4	6	23	
RES b, m	$m_b \leftarrow 0$ m ≡ r, (HL), (IX + d), (IY + d)	•	•	X	•	X	•	• 10					To form new opcode replace 11 of SET b, s with 10. Flags and time states for SET instruction.

NOTES: The notation m_b indicates bit b (0 to 7) of location m.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

			CONDITION									
			UN COND	CARRY	NON CARRY	ZERO	NON ZERO	PARITY EVEN	PARITY ODD	SIGN NEG	SIGN POS	REG B ≠ 0
JUMP 'JP'	IMMEDIATE EXTENSION	nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JUMP 'JP'	RELATIVE	PC + e	18 e - 2	38 e - 2	30 e - 2	28 e - 2	20 e - 2					
JUMP 'JP'	REGISTER INDIRECT	(HL)	E9									
JUMP 'JR'		(IX)	DD E9									
JUMP 'JP'		(IY)	FD E9									
DECREMENT B, JUMP IF NON ZERO 'DJNZ'	RELATIVE	PC e										10 e - 2

Mnemonic	Symbolic Operation	Flags S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
JP nn	PC ← nn	•	•	X	•	X	•	•	•	11 000 011 ← n → ← n →	C3	3	3	10	
JP cc, nn	If condition cc is true PC ← nn, otherwise continue	•	•	X	•	X	•	•	•	11 cc 010 ← n → ← n →		3	3	10	cc Condition 000 NZ non zero 001 Z zero 010 NC non carry 011 C carry 100 PO parity odd 101 PE parity even 110 P sign positive 111 M sign negative
JR e	PC ← PC + e	•	•	X	•	X	•	•	•	00 011 000 ← e - 2 →	18	2	3	12	
JR C, e	If C = 0, continue	•	•	X	•	X	•	•	•	00 111 000 ← e - 2 →	38	2	2	7	If condition not met
	If C = 1, PC ← PC + e											2	3	12	If condition is met
JR NC, e	If C = 1, continue	•	•	X	•	X	•	•	•	00 110 000 ← e - 2 →	30	2	2	7	If condition not met
	If C = 0, PC ← PC + e											2	3	12	If condition is met
JP Z, e	If Z = 0 continue	•	•	X	•	X	•	•	•	00 101 000 ← e - 2 →	28	2	2	7	If condition not met
	If Z = 1 PC ← PC + e											2	3	12	If condition is met
JR NZ, e	If Z = 1, continue	•	•	X	•	X	•	•	•	00 100 000 ← e - 2 →	20	2	2	7	If condition not met
	If Z = 0, PC ← PC + e											2	3	12	If condition is met
JP (HL)	PC ← HL	•	•	X	•	X	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC ← IX	•	•	X	•	X	•	•	•	11 011 101 11 101 001	DD E9	2	2	8	
JP (IY)	PC ← IY	•	•	X	•	X	•	•	•	11 111 101 11 101 001	FD E9	2	2	8	
DJNZ, e	B ← B - 1 If B = 0, continue	•	•	X	•	X	•	•	•	00 010 000 ← e - 2 →	10	2	2	8	If B = 0
	If B ≠ 0, PC ← PC + e											2	3	13	If B ≠ 0

NOTES e represents the extension in the relative addressing mode
e is a signed two's complement number in the range < - 126, 129 >
e - 2 in the opcode provides an effective address of pc + e as PC is incremented by 2 prior to the addition of e

Flag Notation • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown
↕ = flag is affected according to the result of the operation

16-Bit Load Group

SOURCE

DESTINATION			REGISTER							IMM. EXT.	EXT. ADDR.	REG. INDIR.
			AF	BC	DE	HL	SP	IX	IY	nn	(nn)	(SP)
	REGISTER	AF										F1
		BC								01 n n	ED 4B n n	C1
		DE								11 n n	ED 5B n n	D1
		HL								21 n n	2A n n	E1
		SP				F9		DD F9	FD F9	31 n n	ED 7B n n	
		IX								DD 21 n n	DD 2A n n	DD E1
		IY								FD 21 n n	FD 2A n n	FD E1
	EXTERNAL ADDRESS	(nn)		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			
PUSH INSTRUCTIONS	REGISTER IND.	(SP)	F5	C5	D5	E5		DD E5	FD E5			

NOTE: The Push & Pop Instructions adjust the SP after every execution.

8-Bit Load Group

DESTINATION		SOURCE																
		IMPLIED		REGISTER							REGISTER INDIRECT			INDEXED		EXT. ADDR.	IMME.
		I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX + d)	(IY + d)	(nn)	n
REGISTER	A	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A n n	3E n
	B			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 n
	C			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E n
	D			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 n
	E			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		1E n
	H			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		26 n
	L			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		2E n
REGISTER INDIRECT	(HL)			77	70	71	72	73	74	75							36 n
	(BC)			02													
	(DE)			12													
INDEXED	(IX + d)			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
	(IY + d)			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
EXTERNAL ADDRESS	(nn)			32 n n													
IMPLIED	I			ED 47													
	R			ED 4F													

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
EX DE, HL	DE ↔ HL	•	•	X	•	X	•	•	•	11 101 011	EB	1	1	4	
EX AF, AF'	AF ↔ AF'	•	•	X	•	X	•	•	•	00 001 000	08	1	1	4	
EXX	BC ↔ BC' DE ↔ DE' HL ↔ HL'	•	•	X	•	X	•	•	•	11 011 001	D9	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H ↔ (SP + 1) L ↔ (SP)	•	•	X	•	X	•	•	•	11 100 011	E3	1	5	19	
EX (SP), IX	IX_H ↔ (SP + 1) IX_L ↔ (SP)	•	•	X	•	X	•	•	•	11 011 101 11 100 011	DD E3	2	6	23	
EX (SP), IY	IY_H ↔ (SP + 1) IY_L ↔ (SP)	•	•	X	•	X	•	•	•	11 111 101 11 100 011	FD E3	2	6	23	
LDI	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1	•	•	X	0	X	① I	0	•	11 101 101 10 100 000	ED A0	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1 Repeat until BC = 0	•	•	X	0	X	① 0	0	•	11 101 101 10 110 000	ED B0	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
LDD	(DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1	•	•	X	0	X	① I	0	•	11 101 101 10 101 000	ED A8	2	4	16	
LDDR	(DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1 Repeat until BC = 0	•	•	X	0	X	0	0	•	11 101 101 10 111 000	ED B8	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
CPI	A − (HL) HL ← HL + 1 BC ← BC − 1	② I	② I	X	I	X	① I	1	•	11 101 101 10 100 001	ED A1	2	4	16	
CPIR	A − (HL) HL ← HL + 1 BC ← BC − 1 Repeat until A = (HL) or BC = 0	② I	② I	X	I	X	① I	1	•	11 101 101 10 110 001	ED B1	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A − (HL) HL ← HL − 1 BC ← BC − 1	② I	② I	X	I	X	① I	1	•	11 101 101 10 101 001	ED A9	2	4	16	
CPDR	A − (HL) HL ← HL − 1 BC ← BC − 1 Repeat until A = (HL) or BC = 0	② I	② I	X	I	X	① I	1	•	11 101 101 10 111 001	ED B9	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

NOTES: ① P/V flag is 0 if the result of BC − 1 = 0, otherwise P/V = 1

② Z flag is 1 if A = (HL), otherwise Z = 0

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown, I = flag is affected according to the result of the operation

General-Purpose Arithmetic

Decimal Adjust Acc, 'DAA'	27
Complement Acc, 'CPL'	2F
Negate Acc, 'NEG' (2's complement)	ED 44
Complement Carry Flag, 'CCF'	3F
Set Carry Flag, 'SCF'	37

Miscellaneous CPU Control

'NOP'	00	
'HALT'	76	
DISABLE INT '(DI)'	F3	
ENABLE INT '(EI)'	FB	
SET INT MODE 0 'IM 0'	ED 46	8080A MODE
SET INT MODE 1 'IM 1'	ED 56	RESTART TO LOCATION 0038_H
SET INT MODE 2 'IM 2'	ED 5E	INDIRECT CALL USING REGISTER 1 AND 8 BITS FROM INTERRUPTING DEVICE AS A POINTER.

Mnemonic	Symbolic Operation	Flags S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
DAA	Converts acc. content into packed BCD following add or subtract with packed BCD operands	↕	↕	X	↕	X	P	•	↕	00 100 111	27	1	1	4	Decimal adjust accumulator
CPL	$A \leftarrow \overline{A}$	•	•	X	1	X	•	1	•	00 101 111	2F	1	1	4	Complement accumulator (one's complement)
NEG	$A \leftarrow 0 - A$	↕	↕	X	↕	X	V	1	↕	11 101 101 01 000 100	ED 44	2	2	8	Negate acc. (two's complement)
CCF	$CY \leftarrow \overline{CY}$	•	•	X	X	X	•	0	↕	00 111 111	3F	1	1	4	Complement carry flag
SCF	CY ← 1	•	•	X	0	X	•	0	1	00 110 111	37	1	1	4	Set carry flag
NOP	No operation	•	•	X	•	X	•	•	•	00 000 000	00	1	1	4	
HALT	CPU halted	•	•	X	•	X	•	•	•	01 110 110	76	1	1	4	
DI *	IFF ← 0	•	•	X	•	X	•	•	•	11 110 011	F3	1	1	4	
EI *	IFF ← 1	•	•	X	•	X	•	•	•	11 111 011	FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	•	11 101 101 01 000 110	ED 46	2	2	8	
IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	•	11 101 101 01 010 110	ED 56	2	2	8	
IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	•	11 101 101 01 011 110	ED 5E	2	2	8	

NOTES: IFF indicates the interrupt enable flip-flop
CY indicates the carry flip-flop
* indicates interrupts are not sampled at the end of EI or DI

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown
↕ = flag is affected according to the result of the operation

8-Bit Arithmetic and Logical Group

	SOURCE										
	REGISTER ADDRESSING							REG. INDIR.	INDEXED		IMMED.
	A	B	C	D	E	H	L	(HL)	(IX + d)	(IY + d)	n
'ADD'	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
ADD w CARRY 'ADC'	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUBTRACT 'SUB'	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SUB w CARRY 'SBC'	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
'AND'	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
'XOR'	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
'OR'	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
COMPARE 'CP'	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INCREMENT 'INC'	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DECREMENT 'DEC'	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
ADD A, r	A ← A + r	↕	↕	X	↕	X	V	0	↕	10 [000] r		1	1	4	r Reg.
ADD A, n	A ← A + n	↕	↕	X	↕	X	V	0	↕	11 [000] 110		2	2	7	000 B
										← n →					001 C
															010 D
ADD A, (HL)	A ← A + (HL)	↕	↕	X	↕	X	V	0	↕	10 [000] 110		1	2	7	011 E
ADD A, (IX + d)	A ← A + (IX + d)	↕	↕	X	↕	X	V	0	↕	11 011 101	DD	3	5	19	100 H
										10 [000] 110					101 L
										← d →					111 A
ADD A, (IY + d)	A ← A + (IY + d)	↕	↕	X	↕	X	V	0	↕	11 111 101	FD	3	5	19	
										10 [000] 110					
										← d →					
ADC A, s	A ← A + s + CY	↕	↕	X	↕	X	V	0	↕	[001]					s is any of r, n, (HL), (IX + d), (IY + d) as shown for ADD instruction. The indicated bits replace the [000] in the ADD set above.
SUB s	A ← A − s	↕	↕	X	↕	X	V	1	↕	[010]					
SBC A, s	A ← A − s − CY	↕	↕	X	↕	X	V	1	↕	[011]					
AND s	A ← A ∧ s	↕	↕	X	1	X	P	0	0	[100]					
OR s	A ← A ∨ s	↕	↕	X	0	X	P	0	0	[110]					
XOR s	A ← A ⊕ s	↕	↕	X	0	X	P	0	0	[101]					
CP s	A − s	↕	↕	X	↕	X	V	1	↕	[111]					
INC r	r ← r + 1	↕	↕	X	↕	X	V	0	•	00 r [100]		1	1	4	
INC (HL)	(HL) ← (HL) + 1	↕	↕	X	↕	X	V	0	•	00 110 [100]		1	3	11	
INC (IX + d)	(IX + d) ← (IX + d) + 1	↕	↕	X	↕	X	V	0	•	11 011 101	DD	3	6	23	
										00 110 [100]					
										← d →					
INC (IY + d)	(IY + d) ← (IY + d) + 1	↕	↕	X	↕	X	V	0	•	11 111 101	FD	3	6	23	m is any of r, (HL), (IX + d), (IY + d) as shown for INC. DEC same format and states as INC. Replace [100] with [101] in opcode.
										00 110 [100]					
										← d →					
DEC m	m ← m − 1	↕	↕	X	↕	X	V	1	•	[101]					

NOTES The V symbol in the P/V flag column indicates that the P/V flag contains the overflow of the result of the operation. Similarly the P symbol indicates parity. V = 1 means overflow, V = 0 means not overflow. P = 1 means parity of the result is even, P = 0 means parity of the result is odd.

Flag Notation • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

Summary of Flag Operations

Instruction	D_7 S	Z		H		P/V	N	D_0 C	Comments
ADD A, s; ADC A, s	↕	↕	X	↕	X	V	0	↕	8 bit add or add with carry
SUB s; SBC A, s; CP s; NEG	↕	↕	X	↕	X	V	1	↕	8 bit subtract, subtract with carry, compare and negate accumulator
AND s	↕	↕	X	1	X	P	0	0	Logical operations
OR s; XOR s	↕	↕	X	0	X	P	0	0	Logical operations
INC s	↕	↕	X	↕	X	V	0	•	8 bit increment
DEC s	↕	↕	X	↕	X	V	1	•	8 bit decrement
ADD DD, ss	•	•	X	X	X	•	0	↕	16 bit add
ADC HL, ss	↕	↕	X	X	X	V	0	↕	16 bit add with carry
SBC HL, ss	↕	↕	X	X	X	V	1	↕	16 bit subtract with carry
RLA, RLCA, RRA, RRCA	•	•	X	0	X	•	0	↕	Rotate accumulator
RL m, RLC m, RR m, RRC m, SLA m, SRA m, SRL m	↕	↕	X	0	X	P	0	↕	Rotate and shift locations
RLD, RRD	↕	↕	X	0	X	P	0	•	Rotate digit left and right
DAA	↕	↕	X	↕	X	P	•	↕	Decimal adjust accumulator
CPL	•	•	X	1	X	•	1	•	Complement accumulator
SCF	•	•	X	0	X	•	0	1	Set carry
CCF	•	•	X	X	X	•	0	↕	Complement carry
IN r (C)	↕	↕	X	0	X	P	0	•	Input register indirect
INI, IND, OUTI, OUTD	X	↕	X	X	X	X	1	•	Block input and output Z = 0 if B ≠ 0 otherwise Z = 0
INIR, INDR, OTIR, OTDR	X	1	X	X	X	X	1	•	
LDI, LDD	X	X	X	0	X	↕	0	•	Block transfer instructions P/V = 1 if BC ≠ 0, otherwise P/V = 0
LDIR, LDDR	X	X	X	0	X	0	0	•	
CPI, CPIR, CPD, CPDR	X	↕	X	X	X	↕	1	•	Block search instructions Z = 1 if A = (HL), otherwise Z = 0 P/V = 1 if BC ≠ 0, otherwise P/V = 0
LD A, I; LD A, R	↕	↕	X	0	X	IFF	0	•	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b, s	X	↕	X	1	X	X	0	•	The state of bit b of location s is copied into the Z flag

Symbol	Operation
S	Sign flag. S = 1 if the MSB of the result is 1
Z	Zero flag. Z = 1 if the result of the operation is 0
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V = 1 if the result of the operation is even, P/V = 0 if result is odd. If P/V holds overflow, P/V = 1 if the result of the operation produced an overflow
H	Half-carry flag. H = 1 if the add or subtract operation produced a carry into or borrow from bit 4 of the accumulator
N	Add/Subtract flag. N = 1 if the previous operation was a subtract
H & N	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format
C	Carry/Link flag. C = 1 if the operation produced a carry from the MSB of the operand or result

Symbol	Operation
↕	The flag is affected according to the result of the operation
•	The flag is unchanged by the operation
0	The flag is reset by the operation
1	The flag is set by the operation
X	The flag is a "don't care"
V	P/V flag affected according to the overflow result of the operation
P	P/V flag affected according to the parity result of the operation
r	Any one of the CPU registers A, B, C, D, E, H, L
s	Any 8-bit location for all the addressing modes allowed for the particular instruction
ss	Any 16-bit location for all the addressing modes allowed for that instruction
ii	Any one of the two index registers IX or IY
R	Refresh counter
n	8 bit value in range < 0, 255 >
nn	16-bit value in range < 0, 65535 >

TERİMLER

ASSEMBLER: Bir makinanın kendine özgü komutlarının sözle ifadesini sağlayan sözcüklerin kısaltılmalarından oluşmuş sembolik dil

BINARY : İkil sayı

ENASLE : Muktedir kılmak

DISABLE : Etkisiz hale getirmek

LATCH : Binary bilgiyi değiştirmedn tutabilen (kilitleyebilen) ff

MEMORY MAPPED I/O : Giriş/çıkış cihazının CPU tarafından bellekte bir adresmiş gibi algılandığı giriş/çıkış işlemi

MULTİPLEXİNG: Çoğullama

NON-MASKABLE: Engellenemez

NON- INTERRUPTABLE: Kesintiye uğratılamaz

OPERAND:Aritmetik ve lojik işlemde kullanılan veri

PARITY :Binary sayıdaki '1' adedinin çift veya tek olması durumu

REGISTER: Binary bilginin yazıldığı, saklandığı ve işleme sokulduğu flip-flop'lar dizisi,saklayıcı

SHIFT REGISTER: İçindeki binary bilgiyi, bit bit sağa ya da sola kaydırabilen ff

SIMULATED : Benzeştirilmiş

SKIP : Bir programda, bir bölgenin işlem görmeden atlanması

TRI(THREE)-STATE: Üç durum

YARARLANILAN KAYNAKLAR

Dr.D.Burton, A.L.Dexter — "Microprocessors Systems Handbook"USA 1977

M.Moris Mano — "Digital Logiç and Computer Design" USA 1979

Malvino and Leach — "Digital Principles and Applications"Newyork 1982

D.E.Heffer,D.Keith, G.A.King — "Basic Principles and Practice of microprocessors" 1980

İNTEL 8080 — "Microcomputer Systems User's Manuel"Eylül 1975

Elektor — Eylül 1981

J.Nichols,E.Nichols, Peter Rony — "Z80 Microprocessor Programming and Interfacing" 1981, USA

P.Tedeschi, Robert Colon — "101 Projects for the Z80" USA, 1981

Steve Clarcia — "Build Your Own Z80 Computer" 1981 , USA

ÖZGEÇMİŞ

1964 yılında Yerköy'de doğdu.1981 yılında Antalya Lisesini bitirdi.Aynı yıl, Yıldız Üniversitesi Mühendislik Fakültesi Elektrik Bölümüne girdi.1985 yılında mezun olarak, aynı yıl Yıldız Üniversitesi Fen Bilimleri Enstitüsü Elektrik Bölümünde Yüksek Lisans eğitimine başladı.1987 yılında Yıldız Üniversitesi Elektrik Mühendisliği Elektrik Makinaları Anabilim dalı Elektronik Ölçme ve Standartları bilim dalında araştırma görevlisi olarak çalışmaya başladı, halen bu görevi sürdürmektedir.