5.TEMEL BİLGİSAYAR YAPISI VE DEVRELERİ

- Temel bilgisayarlar, iç yazaçlar, zamanlama ve denetim yapısı ve kullandığı komut (buyruk-instruction) kümesi ile tanımlanabilir.
- Bu bölümde biz daha anlaşılır olması bakımından temel bilgisayar üzerinde duracağız.
- Bir bilgisayarın iç yapısı; yazaçlardaki veriler üzerine işlem yapacak mikro işlemler sırasıyla tanımlanır.
- Genel amaçlı bilgisayar ise verilerle birçok mikro işlem yapabileceği gibi, bu mikroişlemlerin hangi sıraya göre işleneceğini de önceden programlanarak bilir (İşletim sistemi ile denilebilir.)

5.TEMEL BİLGİSAYAR YAPISI VE DEVRELERİ

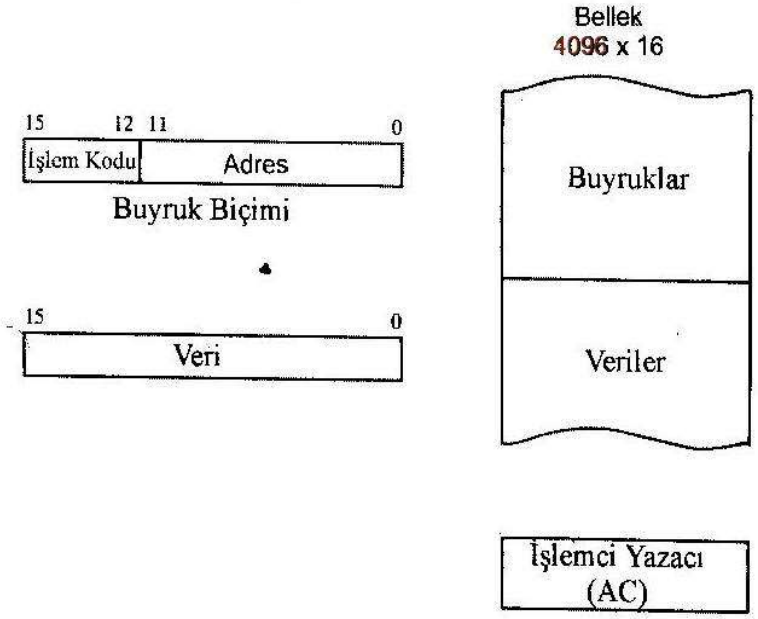
- Bir dizi mikro işlemleri tanımlayan binary kodlu söz dizimine komut (buyruk) diyoruz.
- Buyruklar verilerle beraber bellekte bulunurlar. Bilgisayar herbir buyruğu bellekten okuyarak bunu denetim yazacına (control register) yazar. Denetim bu kodu anlayarak onun gereğini bir dizi mikroişlemleri icra ederek yerine getirir.
- Buyruk (Instruction)kodu: Bilgisayara bellirli işler yapmasını söylerler.Birkaç önemli kısımdan oluşurlar. En önemli kısmı, işlem (operand-OPR) kısmıdır. Bu kısımdaki kombinasyonlar, dört işlemi ve mantıksal ve kaydırma işlemlerini tanımlarlar. OPR bitlerinin sayısı bilgisayardaki mevcut yapılabilecek işlem sayısını da ifade eder.
- Örnek: OPR bit sayısı 4 ise , 16 adet değişik işlem (eğer donanımda mevcutsa) yapılabilir. Örneğin 1101 (ADD) toplama işlemi olabilir.

5.TEMEL BİLGİSAYAR YAPISI VE DEVRELERİ

- Bilgisayar işlemi ile, mikro işlem arasındaki fark?
- Mikro işlem, bilgisayar belleğindeki buyruğun bir parçasıdır. (Özel bir hal olarak buyruğun OPR kısmındaki işlemi yerine getiren kısımdır denilebilir.)
- Buyruk ise, bilgisayara belli işleri yapmasını söyler. Denetim birimi buyruğu bellekten aldıktan sonra, işlem kodu (OPR) kısmındaki bitleri anlamlandırıp, yazaçlardaki bazı mikro işlemleri başlatmak üzere bir dizi işaretler üretir.Her işlem kodu için birtakım mikro işlemler icra edilir. Bunların icrası donanımla yapılır.
- Buyruk kodunun işlem kısmı (OPR) icra edilecek (yürütülecek) işlemi belirler. Oysa buyruk kodunun meramı, sadece işlemi değil, verilerin bulunduğu yazaç, bellek adresler ile sonucun yazılacağı yerin adreslerinin de sağlanmasıdır.
- Her bilgisayar kendine özel buyruk kod biçimine sahiptir.

5.Saklanmış program yapısı

- **Bilgisayar organizasyonunda en önemli unsur,bir işlemci yazacı alarak ve iki parçalı bir buyruk biçimi seçmektir. Birincisi icra edilecek işlemi, ikincisi adresi belirtmelidir. Bu adres kısmı da denetim birimi tarafından gereği değerlendirilerek veri bu adresten okunup, işlemci yazacı içindeki bilgi ile işleme tabi tutulur (12 bit adres uzunluğu).**
- **İşlem kodu: Buyruklar belleğin bir kısmında, veriler diğer kısmında bulunur. Denetim birimi 16 bitlik buyruğu, belleğin buyruklar kısmından okur, veriyi, buyruğun adres kısmını kullanarak belleğin veri kısmından 16 bit olarak okur. Daha sonra, buyruğun işlem kısmı tarafından belirlenen işlemi icra eder.**



Şekil 5.1 saklanmış program yapısı

5.Saklanmış program yapısı

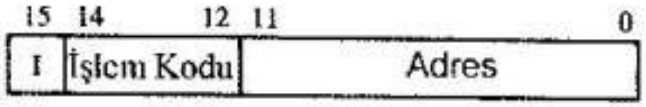
- İşlemci yazacı (AC-Accumulator-birikeç):İşlem bellekten alınan veri ile AC'nin içeriği arasında gerçekleştirilir.

- Eğer buyruk kodu içindeki işlem, bellekten bir veriye gerek duymuyorsa, kod içindeki adres bitleri başka maksatlar için kullanılabilir. Örneğin Ac'yü temizle, bir arttır v.b.

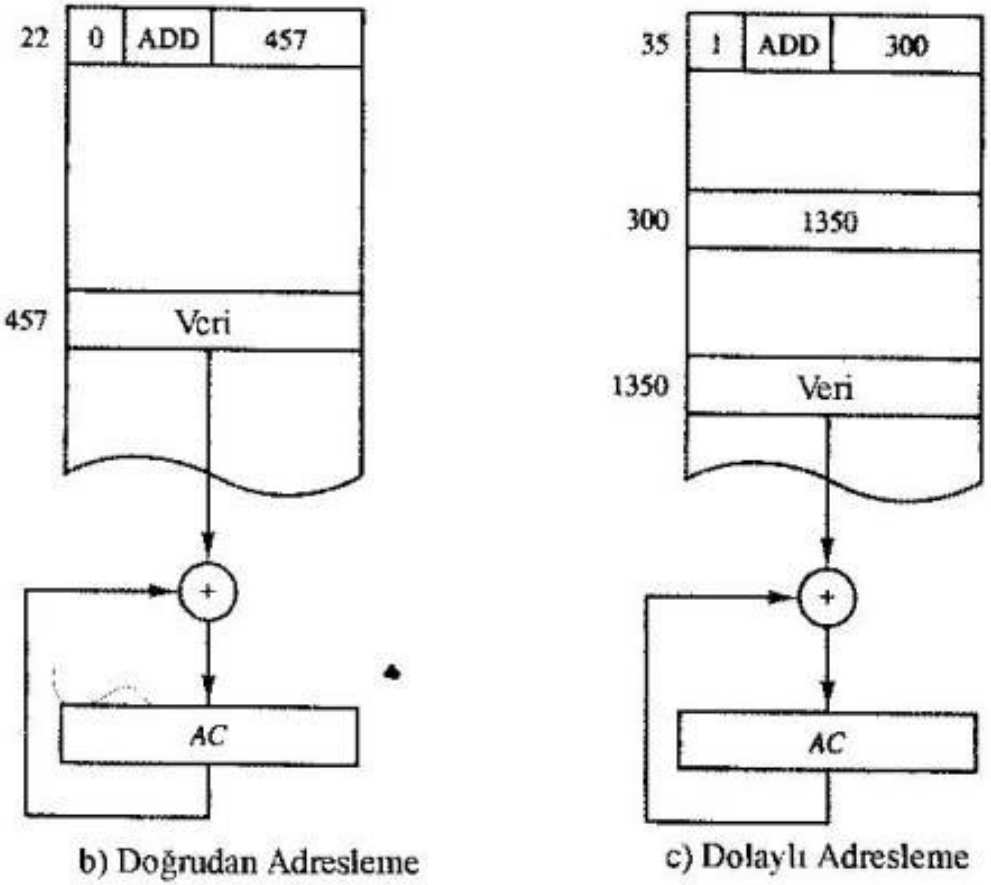
5.DOLAYLI ADRESLEME

- Ani buyruk (İmmediate operand):Bazen, buyruk kodunun adres bitleri yerine verinin kendiside yazılabilir. Bu tip buyruklara ani buyruk denir.
- Eğer buyruğun adres kısmında verinin adresi varsa buna doğrudan adresleme denir.
- Bir başka adresleme ise dolaylı adresleme'dir. Bu adreslemede, buyruk kodundaki adres, işlem görecek verinin adresinin saklandığı bellek gözesinin adresidir.
- Dolaylı ve dolaysız adreslemenin bildirilmesi için buyruk kodunun MSB biti kullanılır.

Dolaylı adresleme



a) Buyruk Biçimi



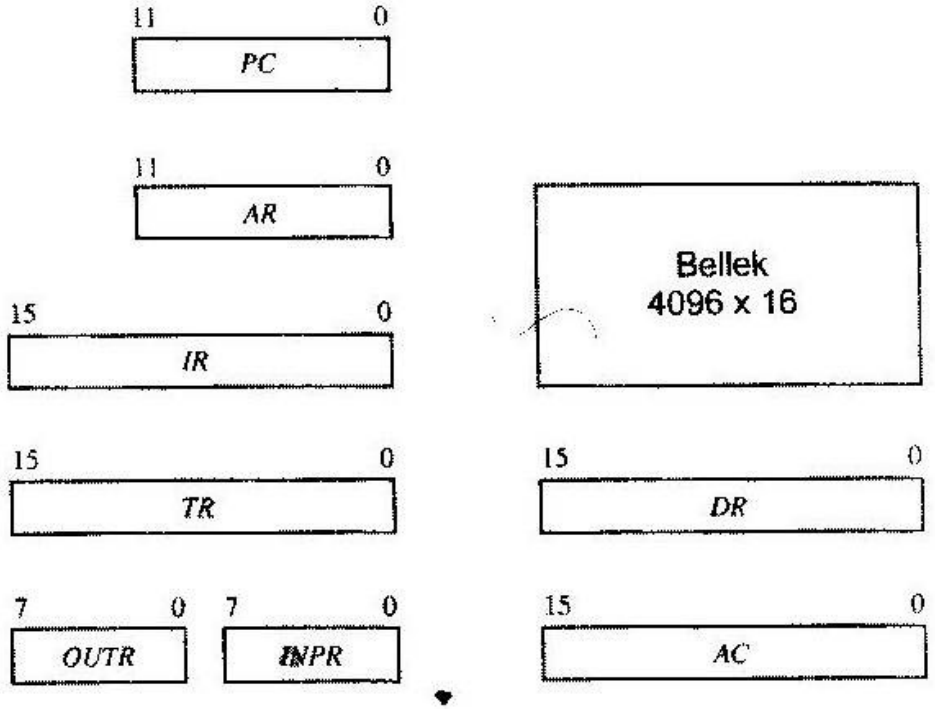
b) Doğrudan Adresleme

c) Dolaylı Adresleme

Şekil 5.2 Doğrudan ve dolaylı adresin gösterimi

5.2 Bilgisayar yazaçları

- Bilgisayarlar, ardışık olarak program bellek adresindeki okuyup, birim zamanda yalnızca bir tanesini işlerler.
- Denetim, belleğin belirili bölgesindeki buyruğu, daha sonra sonrakini okuyup icra eder.Bu ardarda buyrukların adreslerinin hesaplanması için bir sayaca ihtiyaç vardır.
- Ayrıca denetim biriminin içinde bellekten okunan buyruğun saklanması için bir yazaç bulunmalıdır.
- Yani bilgisayarın veriyi işlemek için işlemci yazaçlarına, bellek adreslerini tutmak için bir yazaca ihtiyaç vardır.
- Bu gereksinimlere göre yazaç düzeni Şekil .3'te verilmiştir.



Şekil 5.3 Temel bilgisayar yazaçları ve bellek

Çizelge 5.1 Temel bilgisayar için yazaçların listesi

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

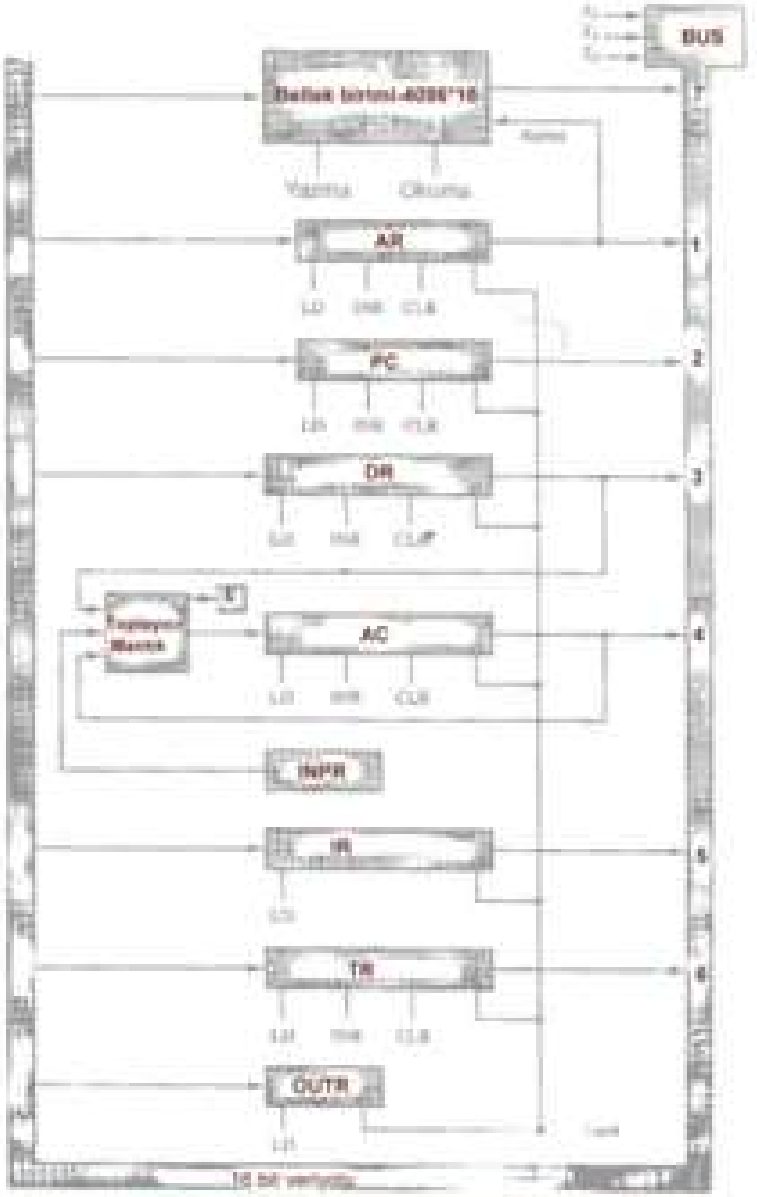
5.2 Bilgisayar yazaçları (Program sayacı)

- Bellek adres yazacı (AR) 12 bittir.(adres uzunluğu 12 bit uzunluğudadır). Program sayacı da bir sonraki buyruk adresini göstereceği için 12 bittir. PC'deki adres bir sonraki işlenecek komutun bellekteki adresidir.PC program çalışırken, sayarak, daha önceden belleğe yerleştirilmiş buyrukların sırasıyla okunmasını sağlar.

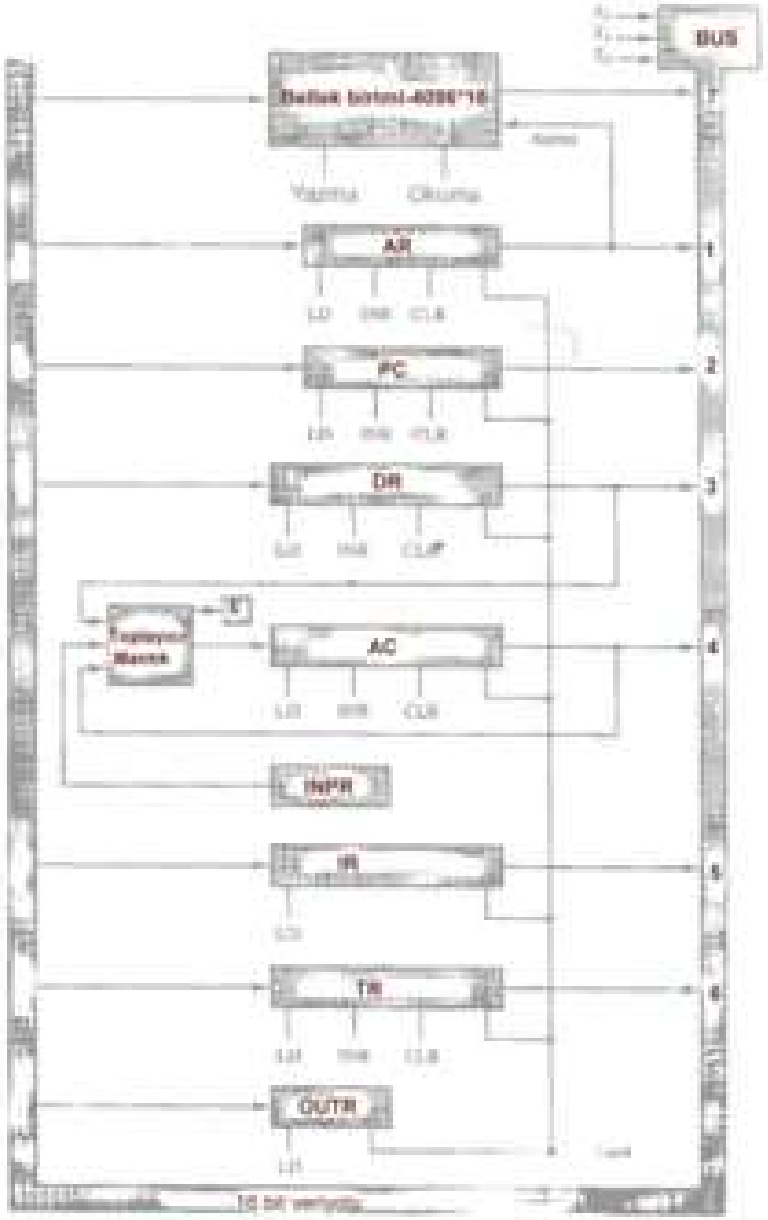
Ortak veriyolu Sistemi

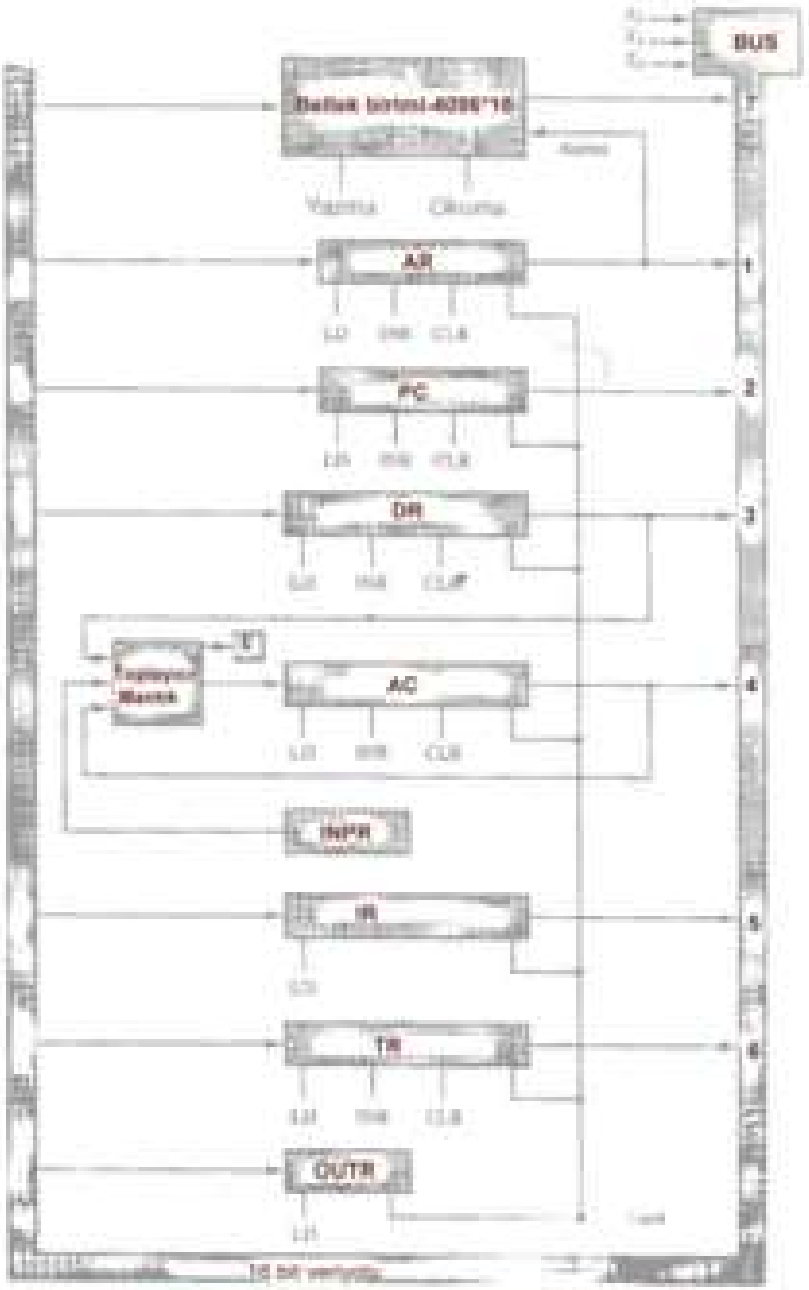
- Çalışacağımız temel bilgisayar devresi , yazaçlar, bellek, denetim birimi ve basit aritmetik-mantık biriminden oluşur.
- Bu birimlerin , özellikle yazaçlar ve bellek ve ALU biriminin biribirleriyle veri alışverişi yapması gerekir.
- Herbir üniteye giriş ve çıkış ayrı ayrı veriyollarıyla olursa işin içinden çıkılmaz.
- Önceki bölümlerde yazaçların ortak veriyolu için, MUX'lar veya 3 durumlu-bufferlı devreler kullanmıştık.
- Temel bilgisayar ortak veriyolu Şekil.5.4'de verilmiştir. burada, yedi yazaç ve bellek'in giriş ve çıkış uçları ortak veriyoluna bağlanmıştır. Ve bir zaman diliminde ünite seçimi için 8:1 mux kullanılmıştır.Mux'un seçme girişleri $S_2S_1S_0$'dır.

- **Yükle (Load-LD):** Her yazacın numarası seçme girişinin bir kombinasyonu olarak görülmektedir.
- DR, AC, IR, TR yazaçlarının ve belleğin kelime uzunluğu 16 bittir.
- Örnek olarak, $S_2S_1S_0$ 011 seçildiğinde DR nin 16 bitlik içeriği veri yoluna çıkar.
- AR ve PC yazacının veri uzunluğu 12 bittir. Bunların verileri veri yoluna aktarıldığında en ağırlıklı 4 biti 0 alınır. Veri yolundan aldıkları bilginin en önemsiz 12 bitini alırlar.
- Bir yazacın veri yolundan bilgi kaydedebilmesi için LD (Load) girişinin aktif edilmesi gerekir. Bu durumda bir sonraki clok kenarında veri yolundaki veriler, seçilmiş yazaca kaydolur.



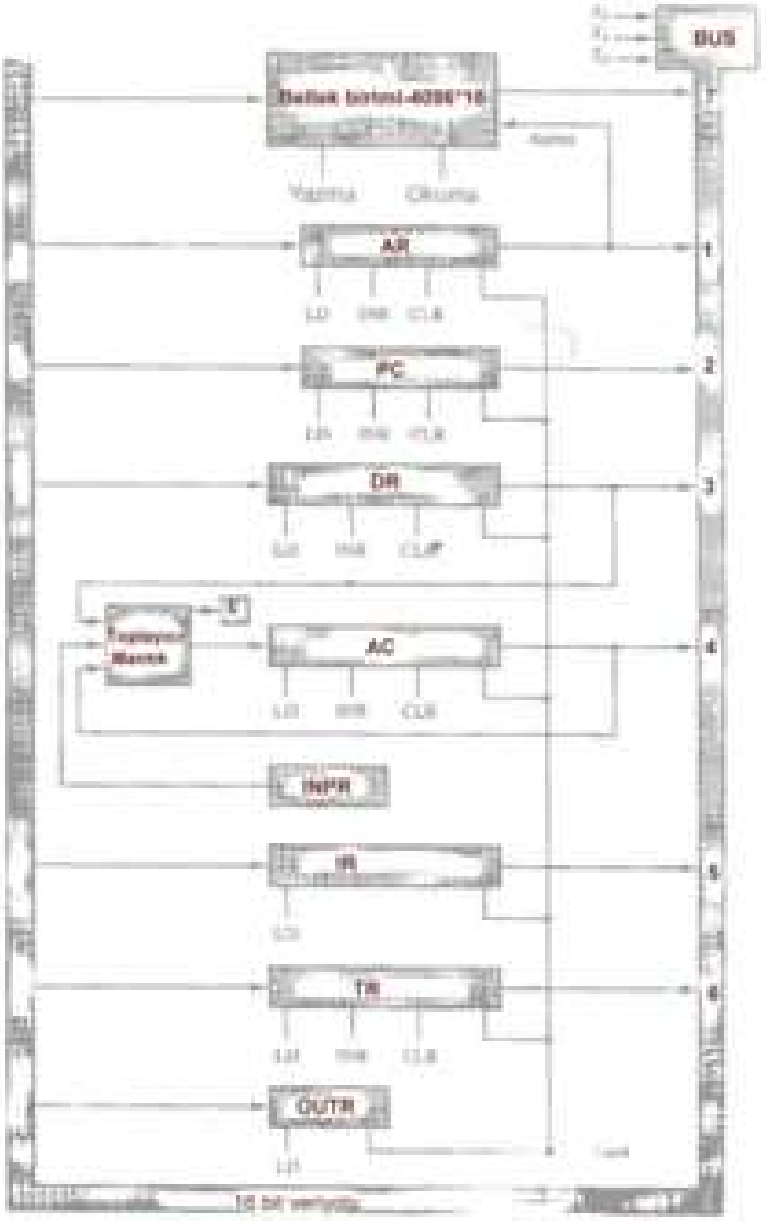
- Giriş ve çıkış yazaçları (INPR ve OUTR) 8 bitliktir. Veri yolunun önemsiz 8 biti aracılığıyla haberleşirler. INPR sadece veriyoluna veri verir (AC aracılığı ile). INTR dış çevre biriminden karakter bazında veriyi alır.
- OUTR sadece veri yolundan veri alır (AC den bir karakter alır ve dış çevre birimine gönderir). OUTR yazacından, başka yazaçlara aktarım yoktur.





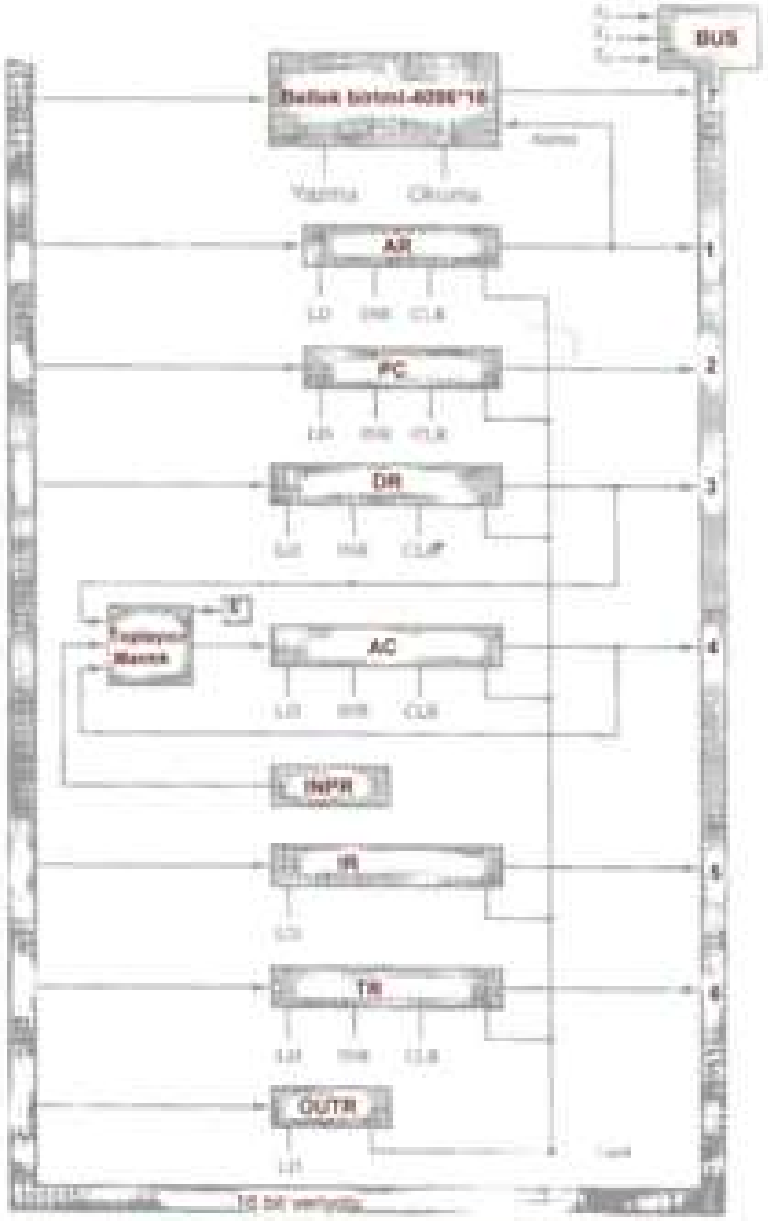
Şekil 5.4 : Temel Bilgisayar yazaçlarının bir ortak veri yoluna bağlanması

- 1,2,3,4,6 yazaçlarının, LD, INC ve CLR olarak , 3 tane denetim girişleri (load (LD), arttır (INC) ve sil(CLR) girişleri) vardır.
- IR ve OUTR yazaçlarının sadece LD girişi vardır. (Bu yazaçların iç yapısı Şekil2.7'de verilmiştir)

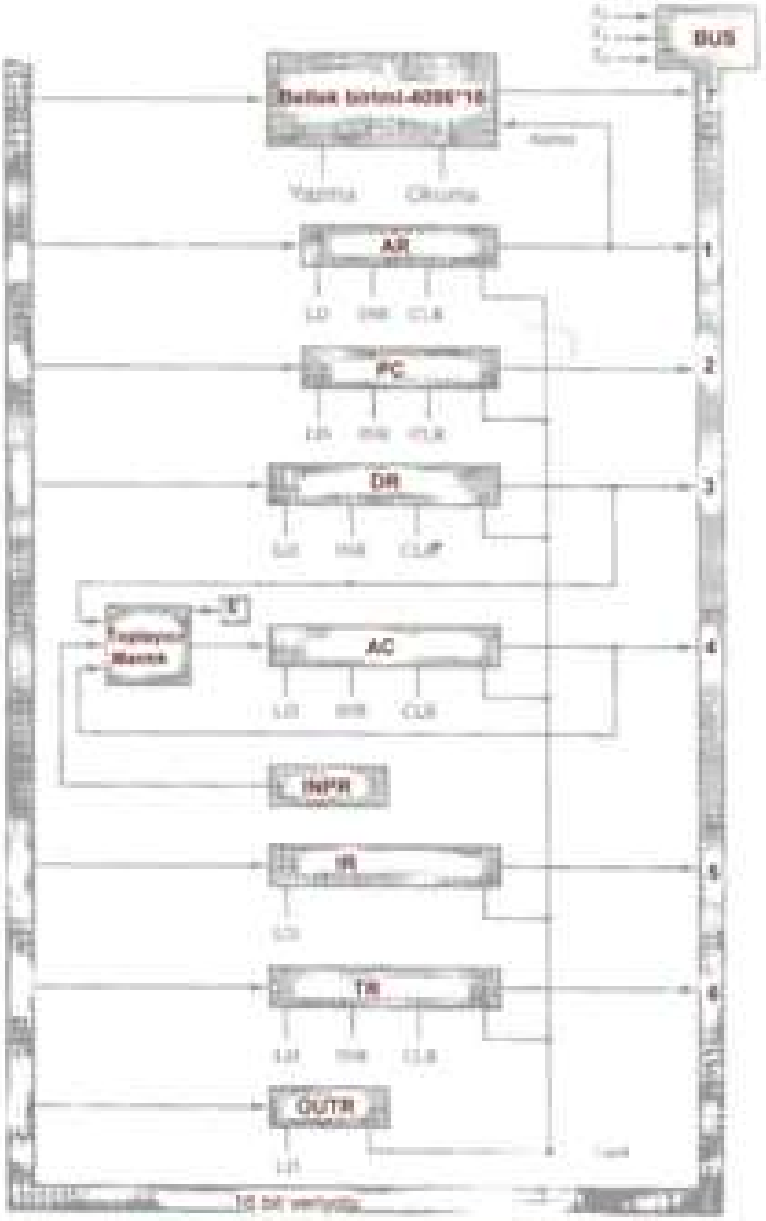


Şekil 5.4 : Temel Bilgisayar yazaçlarının bir ortak veri yoluna bağlanması

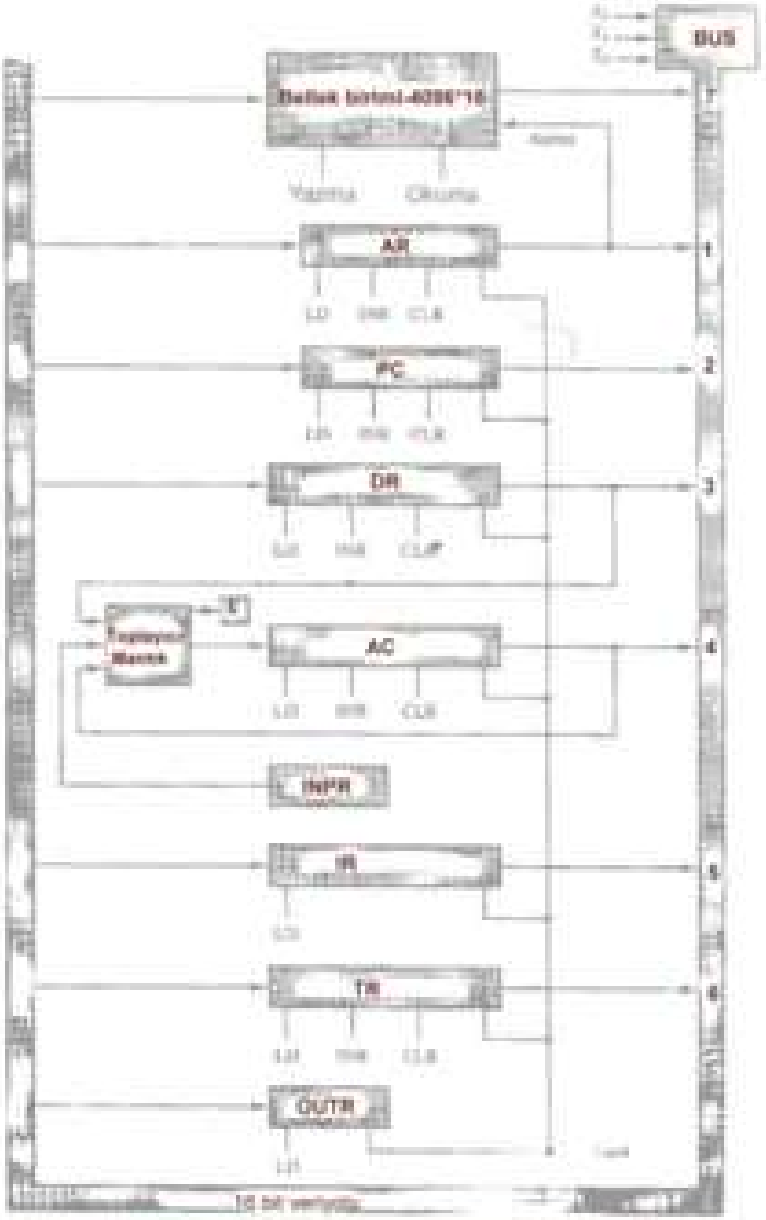
- 4096*16’lık belleğin giriş ve çıkışları ortak veriyoluna bağlanmıştır.Belleğin adres girişi ise AR registerine bağlıdır. Yani bellekteki adres bulma işleminden AR sorumludur.
- Tekbir AR kullanarak ayrı bir adres yolu gereksinimi ihtiyacından kurtulunmuş olur.
- Herhangibir yazacın içeriği bir yazma işlemi sırasında belleğe aktarılmış olur.
- AC hariç herhangi bir yazaç okuma işlemi ile bellekten bilgi alınabilir.



- **AC'nin 16 bitlik girişi, toplayıcı ve Mantık Devresinden gelir. AC'nin bir başka girişi de kendi üzerindeki bilgilerdir (Bunlar kendi üzerinde yapılan işlemler içindir tümleme, kaydırma v.b). AC'nin veri çıkışları aynı zamanda Toplayıcı ve mantık devresinin çıkış bitleride olabilir.**
- **AC' ye DR registerinden de giriş yapılabilir. DR ve AC'den gelen veriler Toplayıcı+mantık biriminde işlem görürler.**
- **ADD DR yi AC'ye ekle işlemi örnek verilebilir. Toplamanın sonucu AC'ye aktarılır. Taşma olursa E bitine yazılır.**
- **AC'ye gelebilen 3.veri kümesi de 8 bitlik İNPR registerindendir.**

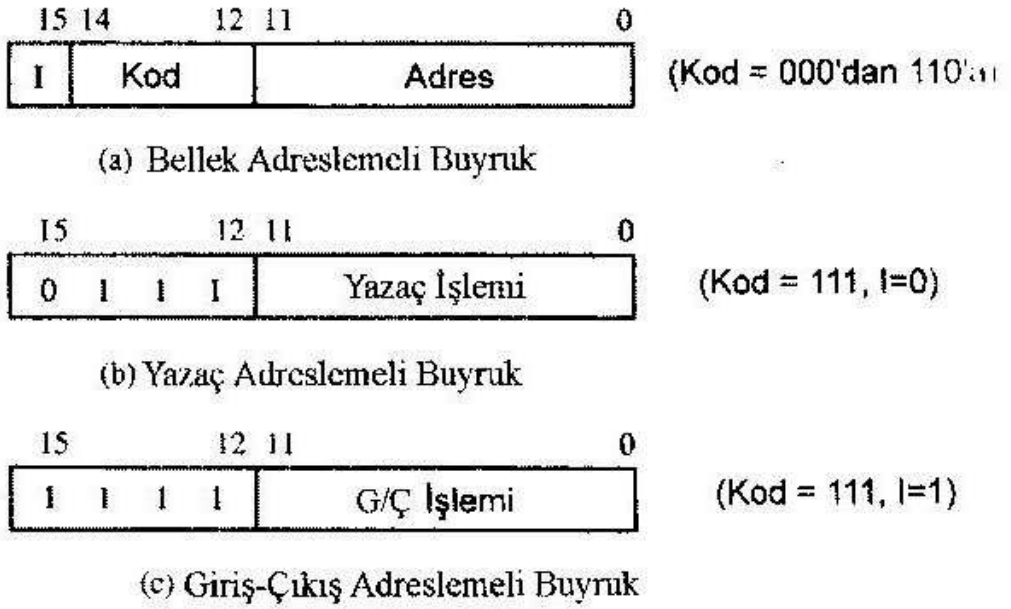


- **Bu şemada herhangibir yazacın içeriği ortak veri yoluna aktarılabilir ve toplayıcı+mantık devrede aynı clock periyodunda işlem görebilir. Clock periyodu sonunda bulunan sonuç veri yolundaki belirlenen hedef yazaca aktarılabilir.**
- **Toplayıcı+mantık biriminde elde edilen sonuç , AC'ye yazılır.**
- **Örneğin: DR←AC ve AC←DR İşlemleri aynı zamanda icra edilir. Bunun için AC'nin içeriğinin veri yoluna aktarılması için $S_2S_1S_0 = 100$ yapılması ve DR'nin LD(LOAD) girişinin aktif edilmesi gerekir. DR'nin içeriğinin de AC'ye aktarılması toplayıcı+mantık devresi üzerinden olur. Bunun için de AC'nin LD girişinin aktif edilmesi gerekir.**



5.3 Bilgisayar Buyrukları

- **Temel bilgisayarın üç buyruk kod biçimi mevcuttur. Her biçim 16 bit'tir. İşlem kodu 3 bittir (12,13,14. bitler).**
- **Bellek Adereslemeli buyruk: ilk 12 biti bellek adresidir, MSB biti =0 ise Doğrudan, 1 ise dolaylı bellek adresleme kipidir.**
- **Yazaç Adreslemeli buyruk: İşlem kodları 111 'dir.MSB biti 0'dır. Bu buyruk tipi AC üzerinde işlem yapmak için kullanılır. Bellekten veriye gerek yoktur. Yani 12 bitlik kısım işlem veya test için kullanılır.**
- **Giriş/Çıkış adreslemeli buyruk:İşlem bitleri 111'dir. MSB biti 1 dir. Kalan 12 bit I/O tipi veya test işlemi için kullanılır.**



Şekil 5.5 Temel bilgisayar buyruk biçimleri

5.3 Bilgisayar Buyrukları-2

- **Buyruk tipi, dnetim birimi tarafından, işlem bitleri ve MSB bitinin değerlendirilmesiyle belirlenir.**
- **İşlem bitleri 111 değilse bu bellek adreslemeli bir buyruktur. MSB biti adresleme kipini belirtir.**
- **İşlem bitleri (12,13,14.bitler) 111 ise, denetim MSB bitine bakarak, (0= ise yazaç adreslemeli, 1 ise G/Ç adereslemeli) anlar.**
- **İşlem kodunun 3 bitlik olması, temel bilgisayarda 8 işlemi belirtir. Ancak yazaç adreslemeli ve G/Ç buyruklarıda kalan 12 bitlik buyruk tanımında kullanılırsa işlem sayısı artar. Temel bilgsayarda 25 buyruk vardır.**

Şekil 5.5 Temel bilgisayar buyruk biçimleri

Temel Bilgisayar Buyrukları

Çizelge 5.2. Temel bilgisayar buyrukları

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

Buyruk kümesinin tamamlılığı

- Bilgisayarların buyruk kümelerinde aşağıdaki 3 tip buyruk alt kümesi mevcutsa küme TAM'dır denir.

1-Aritmetik, mantık, kaydırma buyrukları

2-Bilginin bellek ve yazaçlar arası aktarımını sağlayan buyruklar

3-Durum belirleyen ve denetim buyrukları

4- Giriş/Çıkış Buyrukları

Çizelge 5.2'de bu buyrukları sağlayan minimum buyruk kümesi görülmektedir.

Buyruk kümesinin tamamlılığı-1

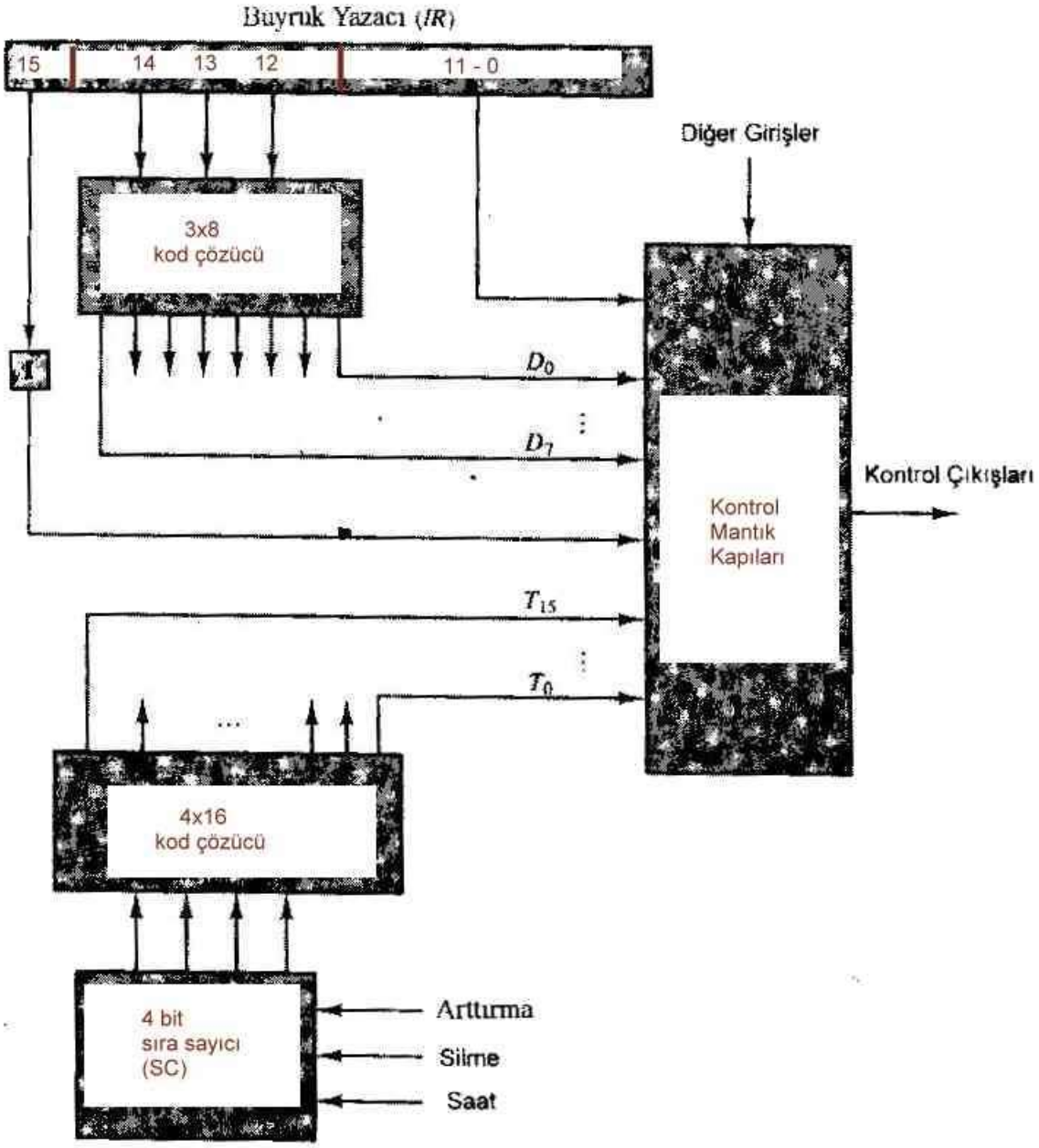
- Örnek, bir tek ADD aritmetik buyruğu ve CMA(ters alma) ve INC (1arttırma) ile toplama ve çıkarma işlemleri başarılabilir.
- CIL (dairesel sola kaydır-AC ve E'nin içeriğini), CIR (Dairesel sağa kaydır- AC ve E'nin içeriğini) kaydırma komutlarıdır.
- Çarpma ve bölme işlemi ise toplama, çıkarma, kaydırma komutları yardımıyla yapılabilir.
- AND, CMA (AC'yi tümle) ve CLA (temizle) komutlarıdır. Bunlarla diğer mantıksal işlemler yapılabilir.
- Bilginin aktarılması ise, LDA, STA ile olur.
- Dallanma buyrukları BUN, BSA, ISZ'dir.
- Giriş/Çıkış buyrukları ise INP ve OUT'tur.
- Daha gelişmiş bilgisayarlarda işlemlerin daha hızlı yapılabilmesi için çıkarma, çarpma, OR ve ex-OR gibi işlem komutları bulunur.

5.4 Zamanlama ve Denetim

- Temel bilgisayardaki bütün yazaçların zamanı bir ana clock tarafından kontrol edilmektedir. Sistemdeki clock işareti tüm FF ve yazaçlara uygulanmaktadır. Yazaçların içeriği, clock palslarını kenarında ve izinlendirildiği takdirde değişebilir.
- Denetim birimi tarafından üretilen denetim sinyalleri ortak veriyolundaki MUX girişlerine, işlemci yazaçlarına ve mikroişlem için SC (Sequence Counter-Sıra sayacına) gönderilir.
- Denetim işareti temelde CLOCK palsı ile senkronizedir.

Donanımsal denetim

- İki tip denetim birimi vardır.
- 1- DONANIMSAL DENETİM: yapısı mantık devreleri, FF'lar ve kod çözücülerle yapılır.
- MİKROPROGRAMLANMIŞ DENETİM: Mikroprogramın güncellenmesiyle oluşturulur.
- Burada donanımsal denetim ile ilgileneceğiz.
- Şekil 5.6'da bu denetim biriminin şeması verilmiştir.Bu ünite, 2 adet kod çözücü, sıra sayıcı ve mantık kapılarından oluşmuş bir üniteden ibarettir.
- Program belleğinden okunan buyruk, buyruk yazacında (IR) tutulur.
- Bu yazacın veri yoluyla bağlantısı Şekil 5.4'de verilmişti.

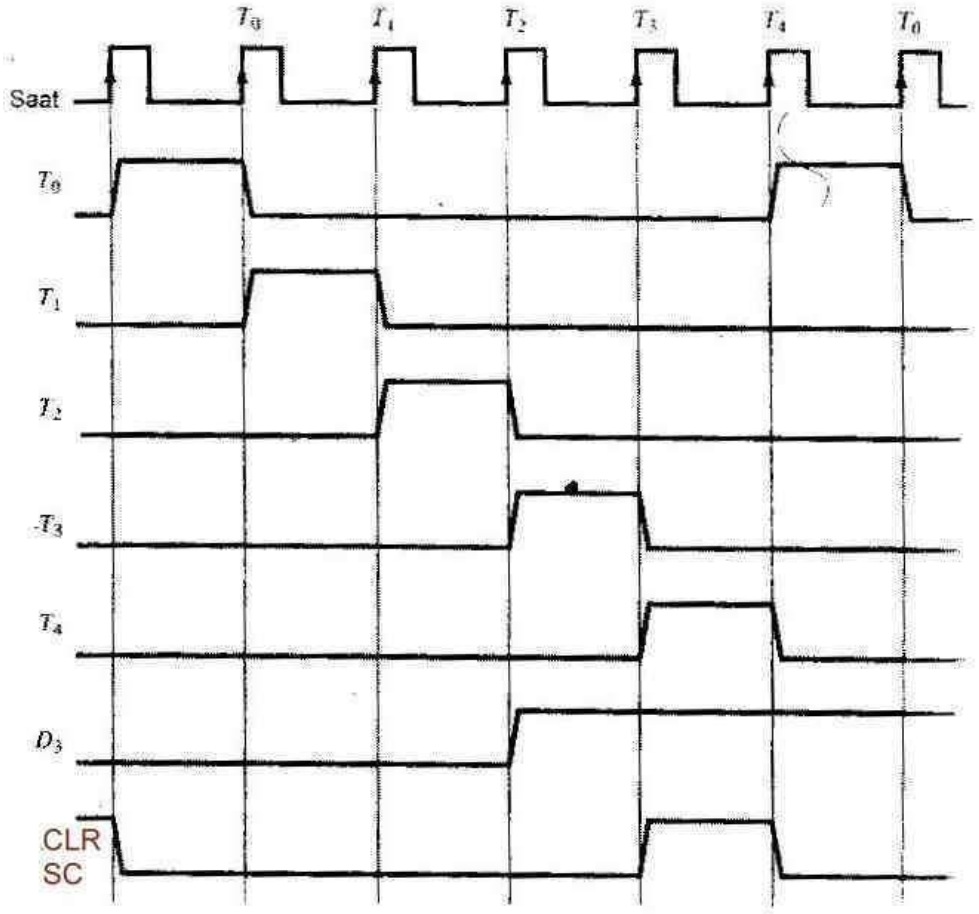


Şekil 5.6 Temel bilgisayarın denetim ünitesi

Zamanlama Sinyali

- **4 bitlik sıra sayacının çıkışları, 4x16'lık kod çözücünün çıkış seçme girişlerine uygulanır. Böylelikle Kod çözünün 16 adet çıkışı 16 tane zamanlama sinyali üretmiş olur.**
- **eğer sayıcı sıfırlanmış ise, buna karşılık T_0 sinyali gönderilir. Sıra sayacı arttırılarak T_0,T_1,T_2,T_3 ve T_4 zaman sinyalleri üretilmiş ve gönderilmiş olur.**
- **T4 zamanında D3 kod çözücü çıkış aktifse SC→ 0 olur.**

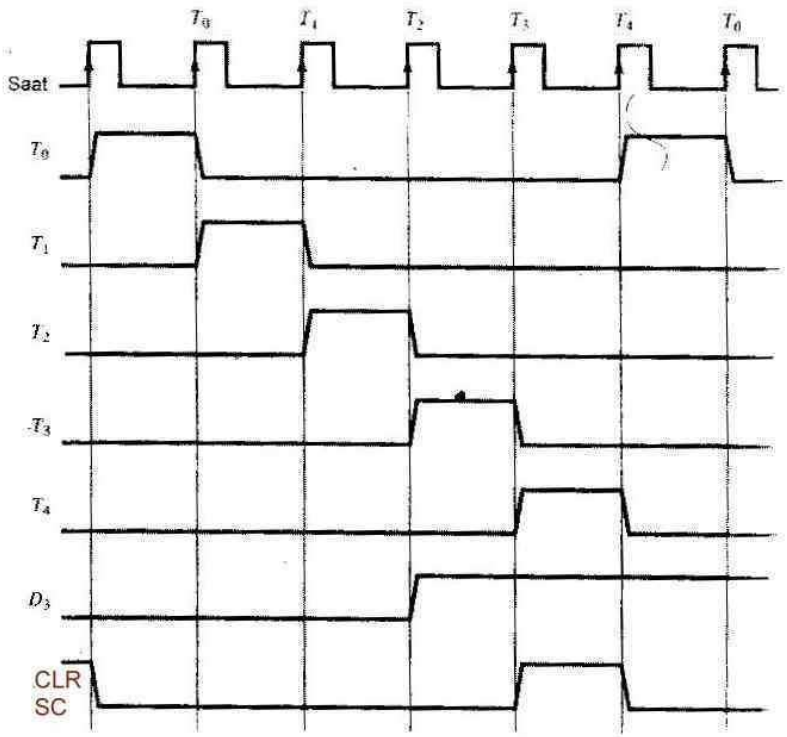
  D_3T_4 : SC←0
- **Şeklinde sembolize edilir.**



Şekil 5.7 Denetim zaman sinyal örneği

Zamanlama Sinyali-1

- **Üretim clock sinyalinin yükselen kenarlarında yapılır. SC'nin silme girişi aktiftir.**
- **SC'ye gelen clock işaretinin ilk yük.kenarı, SC'yi 0'lar. Bu duru T_0 zaman sinyalini başlatır.T_0 çıkışı zaman birimi süresince (2. cp'nın yükselen kenarına kadar) aktif kalır (Lojik 1).**
- **T0 sinyali, sadece denetim girişlerine bu sinyalin uygulandığı yazaçları etkiler.**
- **SC her clock palsının yük.kenarlarında birer ileriye sayar. Ancak silme girişi aktif olunca sıfırlanır.**
- **Bu durumda T0,T1,T2…. Zamanlama işaretleri devamlı olarak üretilmiş olur.**



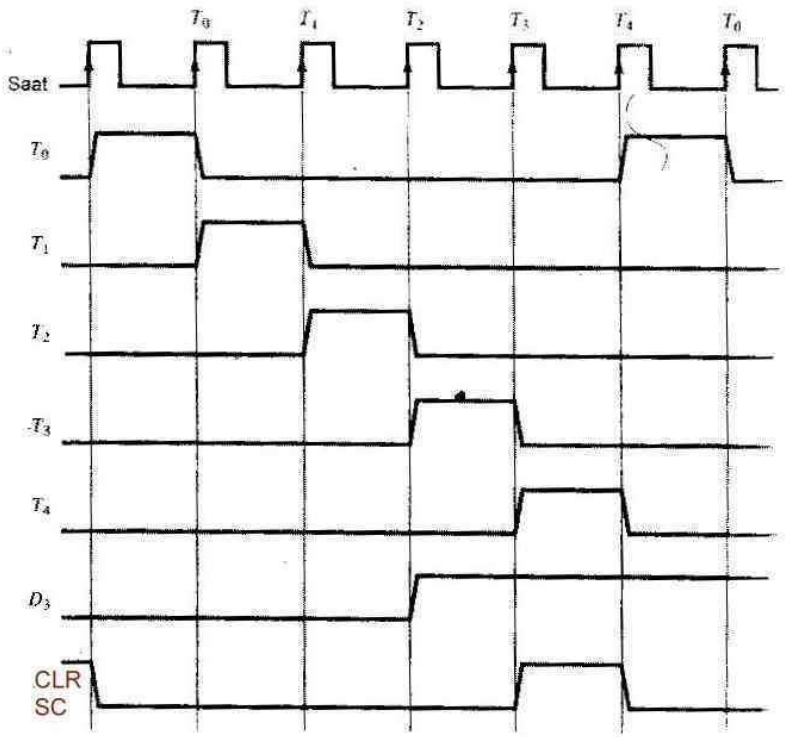
Şekil 5.7 Denetim zaman sinyal örneği

Zamanlama Sinyali-2

- **Ancak şekilden de görüldüğü gibi, sayıcını 15'e kadar sayması istenmiyorsa , herhangi bir zamanda SC belli bir zamanda sıfırlanabilir.(Son üç sinyale dikkat ediniz)**
- **D_3T_4=1 olduktan sonra, ilk cp'nın yükselen kenarında**

$$D_3T_4 : SC \leftarrow 0$$

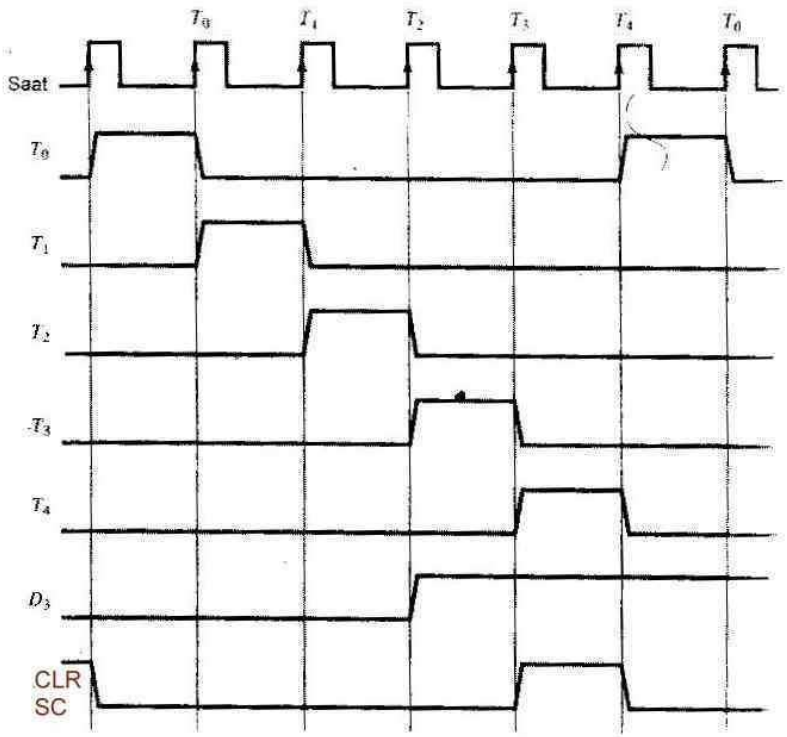
Sayıcı 0'lanır. Sonra tekrar T_0 sinyali ve diğerleri başlar.



Şekil 5.7 Denetim zaman sinyal örneği

Zamanlama Sinyali-3

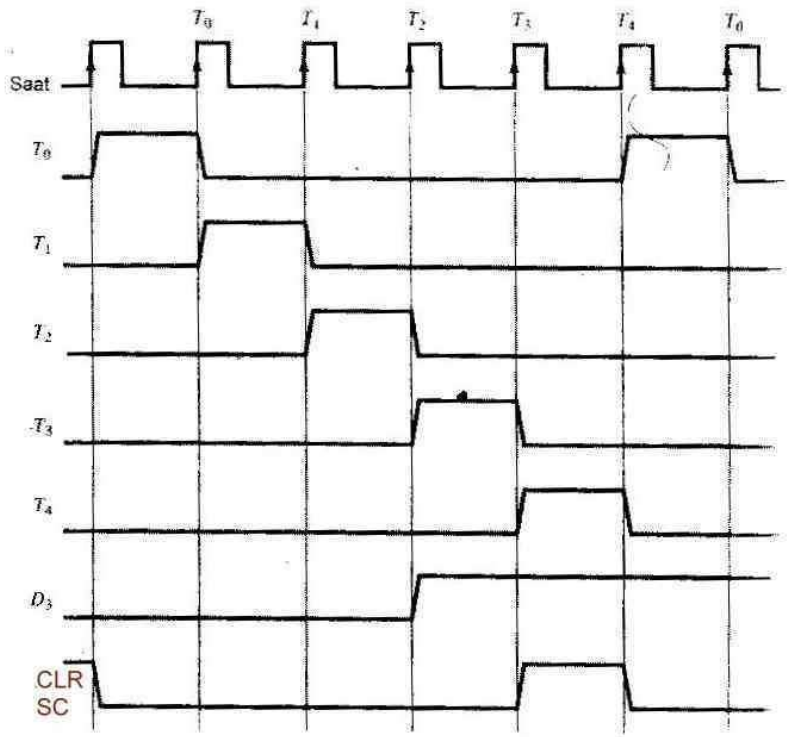
- **Bellekten okuma veya yazma sürecinin, zaman sinyali sürecinden kısa olduğu varsayılarak (bu herzaman doğru değildir),**
- **Okuma veya yazma süreci, zaman sinyalinin yükselen kenarında başlar ve düşen kenarına kadar tamamlanmalıdır.**
- **Saat süreci bundan sonra bellekten okunan bilginin yazaca yazılması için kullanılır.**
- **Bellekten okuma/yazma işlemi daha uzun sürdüğünden çoğu bilgisayarlrada bu işlem süresince *bekle* konumunda durur.**



Şekil 5.7 Denetim zaman sinyal örneği

Zamanlama Sinyali-4

- Temel bilgisayarda, bekle sürecinin, zaman sinyalinden küçük olduğu varsayılacaktır.
- Örnek: **T0: AR←PC**
- Yazaç aktarım deyimini ele alalım. PC'nin içeriğinin AR'ye T_0 sürecinde aktarılması işlemidir. T0 bu süreçte aktifdir. Bu süreçte, PC'nin içeriği veriyoluna yerleştirilir ($S_2S_1S_0 = 010$ ile). AR yazacının LD girişi aktiflenir. Aktarım işlemi bir sonraki CP'nın yükselen kenarında tamamlanır. Bu yükselen kenar ile, SC 0000'dan 0001 konumuna geçer. T_1 aktif olur, T_0 pasif duruma geçer.



Şekil 5.7 Denetim zaman sinyal örneği

5.5.Buyruk Süreci

- Bir program,sonuçta buyruklar kümesidir. Buyruklarında yerine getirilme süreçleri vardır.Her bir buyruk süreci birtakım alt süreçlerden oluşur.Bunlara alt süreçler denir.

Temel bilgisayarlarda her buyruk 4 alt süreçten oluşur.

1- Bir buyruğun bellekten alınıp getirilmesi. (Fetch)

2- Buyruk kodunun çözülmesi (Decode)

3- Buyruk dolaylı adreslemeli ise etkin adresin okunması

4- Buyruğun icrası

Dördüncü adımdan sonra, denetim 1.adım süreci yani al-getir evresine döner. Bu işlem HALT buyruğu icra edilene kadar sürer.

Al-Getir kodunu çöz

- PC(program sayıcı) ilk buyruğun adresi ile yüklenir. SC sıfırlanır. Böylece T_0 zaman sinyali elde edilir. Her bir clock vuruşunda SC bir arttırılır. Böylece zaman sinyalleri T_0, T_1, T_2….diye üretilir.
- Al-getir ve kod çöz süreçlerine ait mikroişlemler aşağıdaki yazaç buyrukları ile belirlenir.

  T0: AR←PC

  T1: IR←M[AR], PC←PC+1

  T2: D0,..D7←kodu çöz IR(12-14), AR←IR(0-11), I←IR(15)

Al –Getir Evresi için yazaç aktarımları

T0:AR←PC mikroişlemi

Belleğin adres girişleri sadece AR çıkışlarından beslenir. Dolayısıyla, önce PC'deki bilginin(PC'deki komutun bellekteki adresi) AR'ye aktarılması gerekir. Bu aktarım işlemi T_0 sürecinde olur.

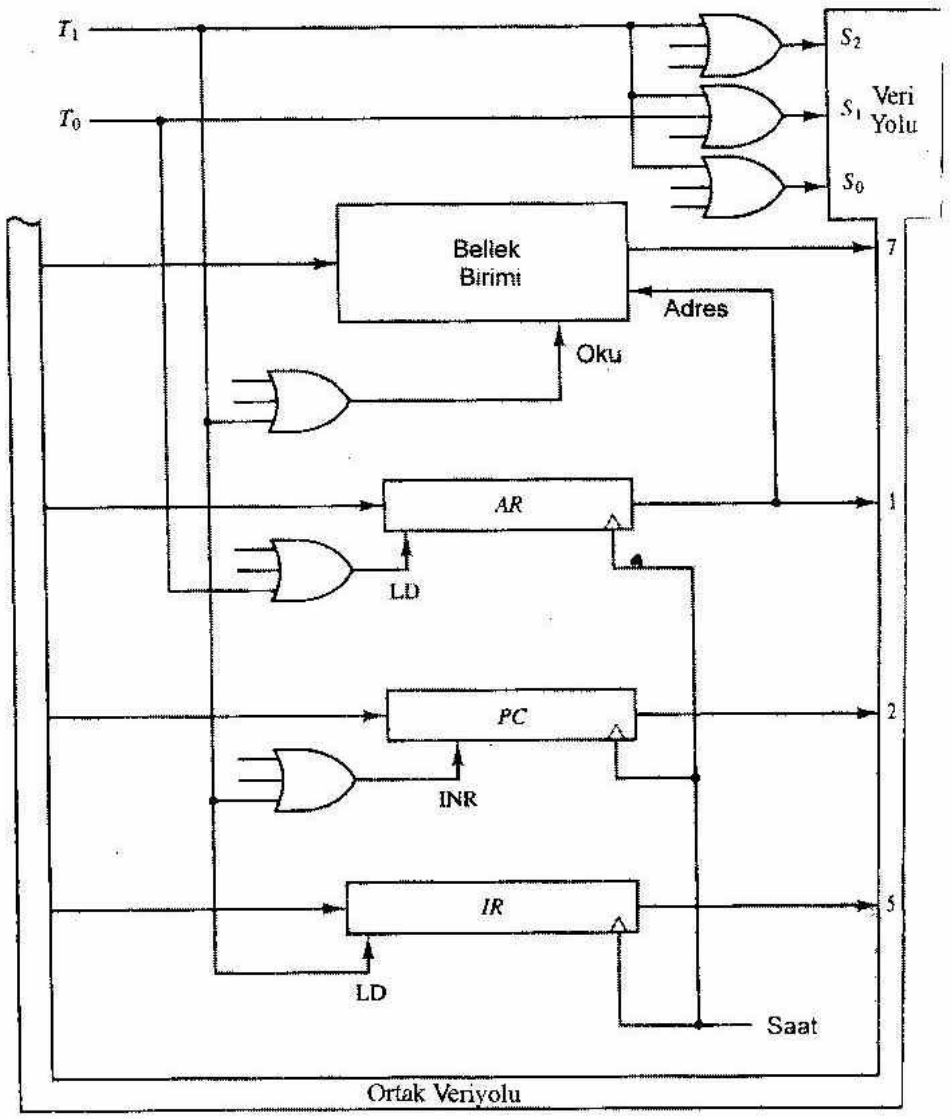
- **Nasıl olur ?**

  T0 sürecinde;
- **Veri yolu seçici $S_2S_1S_0$ = 010 ile PC'nin içeriği veri yoluna aktarılır.**
- **AR'nin LD girişi aktiflenir.**
- **Bundan sonraki clock palsının yükselen kenarında**
- **AR←PC dir.**
- ******: T0 ve T1 uçlarının nerelere bağlanmış olduğuna dikkat ediniz**

. T0: AR←PC

T1: IR←M[AR], PC←PC+1

T2: D0,..D7←kodu çöz IR(12-14), AR←IR(0-11), I←IR(15)



Şekil 5.8 Al-getir evresi için yazaç aktarımları

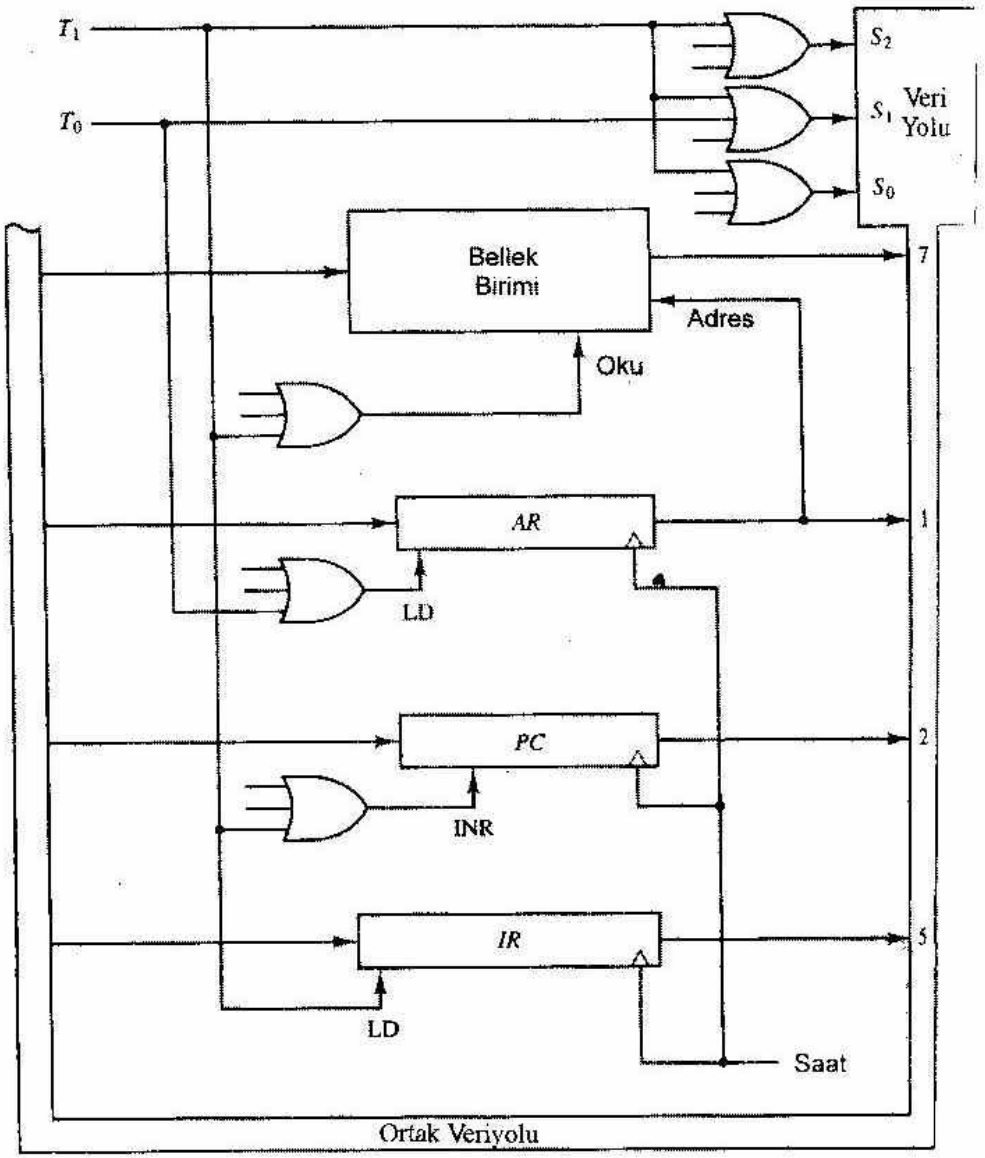
Al –Getir Evresi için yazaç aktarımları

T1:IR←M[AR], PC←PC+1 mikroişl.

- **Bellekten okunan buyruk IR'ye yerleştirilmelidir. Bu iş T_1 sürecinde olur (T_0 süreci sonunda bellekteki buyruğun adresi AR'ye aktarılmıştır). Aynı süreçte PC 1 arttırılarak bir sonraki buyruk adresi hazır hale getirilir.**
- **Nasıl?**

T1'in 1 olmasıyla $S_2S_1S_0$ = 111 olur,Bellek okuma girişi ve IR'nin LD girişi, PC'nin İncrement girişi aktiflenir. Bu durumda bellek adresindeki buyruk kodları veriyoluna çıkmıştır,sonraki clock palsında buyruk kodları IR yazmacına yazılmıştır.

FETCH işlemi bitti



Şekil 5.8 Al-getir evresi için yazaç aktarımları

Buyruk Tipinin Belirlenmesi

- T1 süreci sonunda T2 süreci başlar ve bu süreçte buyruk kodunun çözümlenmesi gerçekleşir.

  IR(12-14) ;deki İşlem Kodunun çözümü

  AR←IR(0-11), I←IR(15) ;Adres

Dolaylı Adresleme:

Eğer OPR kodu 111 ise kod çözücünün D7 çıkışı 1 olur.

D7=1 ise buyruk, Yazaç veya G/Ç adreslemeli bir buyruktur.

D7=0 (İşlem kodu 0-6 arası bir işlem ifade eder) ise bu buyruk bellek adreslemeli bir buyruktur. Şekil 5.9'da;

$D_7'IT_3$: AR←M[AR]

$D_7'I'T_3$: Hiçbirşey

$D_7I'T_3$: Yazaç adreslemeli buyruğu icra et

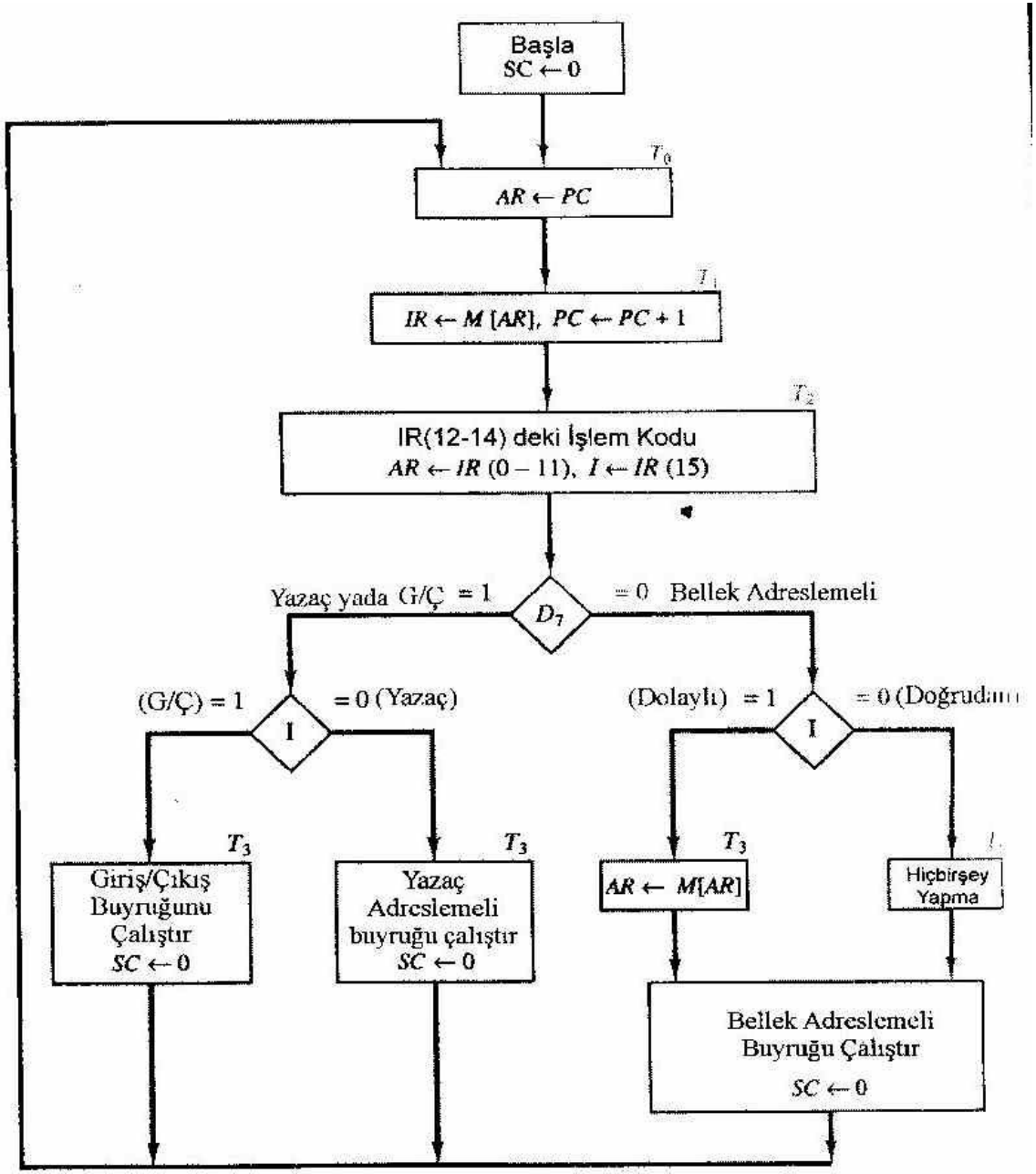
D_7IT_3: G/Ç buyruğunu icra et.

T0: AR←PC

T1: IR←M[AR],
PC←PC+1

T2: D0,..D7←kodu çöz
IR(12-14),
AR←IR(0-11), I←IR(15)

$D_7'IT_3$: AR←M[AR]
$D_7'I'T_3$: Hiçbirşey
$D_7I'T_3$: Yazaç adreslemeli buyruğu icra et
D_7IT_3: G/Ç buyruğunu icra et.



Şekil 5.9 Buyruk süreci için akış şeması

Yazaç Adreslemeli Buyruklar

- Yazaç adreslemeli buyruklar D_7=1 olması ile tanınır.I=0 'dır (Yani 16 bitlik buyruk kodu 0111(7)rakamı ile başlar). Bu buyruk, buyruk kelimesineki ilk 0-11.bilerin değişik kombinasyonuyla 12 farklı buyruk oluşturabilir. Bu 12 bit T1 sürecinde IR'ye aktarılmıştı, T2 sürecinde de IR'deki 12 bit (0-11) AR'ye aktarılır. Yazaç adreslemeli buyruğun icrası T_3 sürecinde gerçekleşir.
- Bu buyruklar için denetim fonksiyonu $D_7I'T_3$ şeklindedir.

Çizelge 5.3 Yazaç-adreslemeli buyruklarının çalışması

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

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

Bellek Adreslemeli Buyruklar

- **7 adet bellek adreslemeli buyruk çizelge 5.4'te verilmiştir. Çizelgedeki $D_0 \ldots D_6$ herbir buyruk için kod çözücünün çıkışlarıdır. Buyrukların etkin adresleri adres yazacı AR'dedir. Buyruğun veri yolu içerisinde gerçek icrası için bir dizi mikroişle gerekir. Çünkü bellekte bulunan veriler doğrudan işlenemezler. Bu buyrukları detaylı inceleyelim.**

Çizelge 5.4 Bellek-adreslemeli buyruklar

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

Bellek Adreslemeli buyruklar
AND: AC'yi VE'le

- Bu buyruk, AC'de bulunan bitlerle, etkin adresten alınan bellek kelimesinin bitlerini çift çift VE'ler ve sonucu AC'ye aktarır. Bu buyruğu icra eden mikroişlemler;
- D0T4: DR←M[AR]
- D0T5: AC←AC^DR, SC←0
- Bu buyruk için denetim fonksiyonu D_0 işlem kod çözücüsünü kullanır.Çünkü buyruk içerisindeki OPR 'de 000 işlem kod değeri ile AND işlemi buyruk içerisindedir.Buyruğun icrası için iki zaman sinyali gerekir.
- T4 zaman sürecinde veri DR'ye aktarılır.
- T5 zaman sürecinde de AC'nin içeriği ile DR'nin içeriiği VE'lenir ve AC'ye yazılır.SC=0 yapılır. Sonra T0 ile yeni bir buyruğun icra adımları başlar.

Bellek Adreslemeli buyruklar
ADD: AC'ye Ekle

- Etkin adresi verilen bellek kelimesinin içeriğini,ac'nin değeri ile toplar.Toplamı AC'ye yazar. AC'nin çıkış eldesi E'ye aktarılır.Bu buyruğun icrası için gerekli mikroişlemler;
- D_1T_4 : DR←M[AR]
- D_1T_5 : AC←AC+DR, E←$C_{çıkış}$, SC←0

- Dikkat: Aynı T4 ve T5 zamanlama sinyalleri kullanılmıştır. Fakat işlem kodu D_0 (AND) değil D_1 (ADD)'dir.

Bellek Adreslemeli buyruklar
LDA: AC'ye Yükle

- Etkin adresi verilen bellek kelimesinin içeriğinin AC'ye aktarılmasıdır.Bu buyruğun icrası için mikroişlemler,
- D_2T_4 : DR←M[AR]
- D_2T_5 : AC←DR, SC←0
- Şk.5.4'den görüldüğü gibi, veri yolundan AC'ye doğrudan yol olmadığından, önce bellek kelimesi DR'ye okunur, buradan AC'ye okunur.

Bellek Adreslemeli buyruklar
STA: AC'yi belleğe aktar

- AC'nin içeriğini etkin adresi verilen bellek kelimesine yazar. AC'nin çıkışları, veri yoluna ve veri yolunda belleğin veri girişlerine doğrudan bağlı olduğu için, bu komut tekbir mikroişlemle yapılabilir.

D_3T_4: M[AR]←AC, SC←0

Bellek Adreslemeli buyruklar
BUN: Şartsız dallan

- Bu buyruk, programı etkin adresle verilen adrese gönderir.PC bir sonraki adımda bellekten okunan buyruğun adresini tutar.PC, T1 zamanında 1 arttırılarak bir sonraki buyruğun adresini hazırlar.
- BUN buyruğu,kullanıcıya buyrukların sırasını şartsız olarak değiştirme yetkisi kazandırır. Buyruk tekbir mikroişlemle icra edilir.

D_4T_4: PC←AR, SC←0

Etkin adres AR'den ortak veriyolu aracılığı ile PC'ye aktarılır. SC'nin 0 yapımasıyla, denetim T0 zamanına geçer, ve PC'deki yeni adresteki buyruk alınıp getirilir.

Bellek Adreslemeli buyruklar

BSA:Dallan ve geri dönüş adresini sakla

Bu buyruk, programdan alt programa dallanması için uygundur.

Bu buyruk icra edilirse, birsonraki buyruğun adresini PC'den alarak bellekteki beliri bir adrese yazar. Sonra Alt programın başlangıç adresini PC'ye yazar ve a ilk adresi olarak çalıştırır.

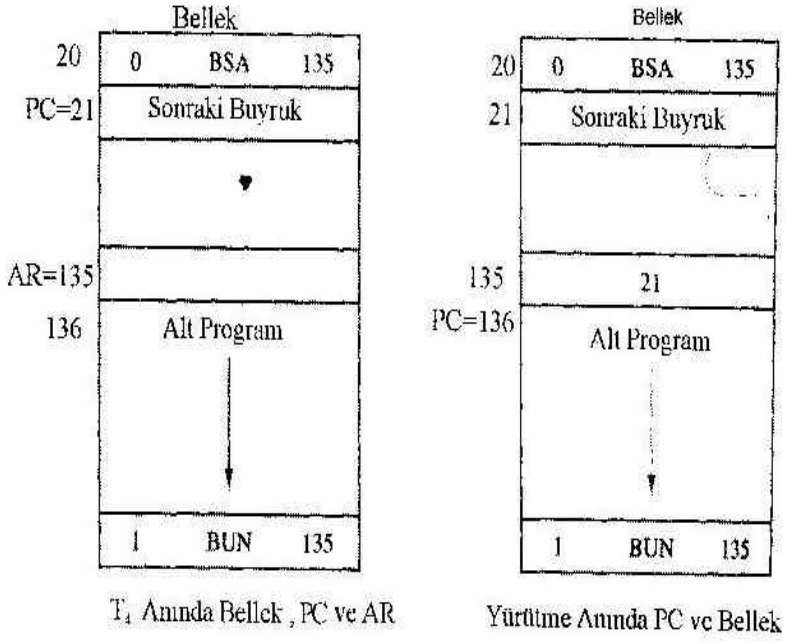
Bu işlem çizelge 5.4'deki yazaç aktarım işlemleriyle verilmiştir.

M[AR]←PC, PC←AR+1

Bellek Adreslemeli buyruklar

BSA:Dallan ve geri dönüş adresini sakla

- **Geri dönüş Adresi: Bu buyruğun bir alt programda nasıl kullanıldığı şekilde verilmiştir. BSA'nın bellek adresi 20 olsun. I biti 0'dır. Fetch ve kod çözme adımlarından sonra PC'de 21 adresi vardır.Bu programdaki bir sonraki buyruğun adresidir. AR'nin içinde 135 olsun (Şekil a). BSA buyruğu aşağıdaki işlemi yapar;**
- **M[135]←21, PC←135+1=136**
- **Bu işlemin sonucu şekil (b) dedir. Dönüş adresi 21, bellek adresi ise 135'te bulunmaktadır.**
- **Denetim programı işletmeye 136 adresinden devam eder.Alt programdan ana programa dönüş işlemi (21 adresi), dolaylı BUN buyruğu ile olur.**
- **BUN buyruğu alt programın son buyruğudur.Bu buyruk icra edilince 135adresinde bulunan etkin adres (21) vardır.Yani önceden kaldığı yerden program devam eder.**



T_4 Anında Bellek , PC ve AR — Yürütme Anında PC ve Bellek

Şekil 5.10 BSA buyruğunun icra örneği

Bellek Adreslemeli buyruklar

BSA:Dallan ve geri dönüş adresini sakla

- Alt program çağrısı: BSA buyruğu, genellikle alt program çağırma adı verilen fonksiyonu icra eder.Alt programın en son satırı ise BUN buyruğudur ve alt programdan dönüşü ifade eder.
- BSA buyruğu, 1 zaman sürecinde icra edilemez. Bellek ve veriyolunun uygun kullanımıyla ard arda iki süreçte icra edilir.

- D5T4: M[AR]←PC, AR←AR+1
- D5T5: PC←AR, SC←0

- T4 süreci; belleğe yazma işlemi ve PC içeriğinin veriyoluna verilmesi ve AR'nin 1 arttırılması için gerekir.clok palsıyla bu işlem ler gerçekleşir.
- T5 süreci; AR'nin PC'ye aktarılması işlemi için kullanılır.

Bellek Adreslemeli buyruklar
ISZ: Arttır ve Eğer Sıfır ise Atla

- Bu buyruk etkin adresteki kelimenin değerini arttırır. Eğer artırılan değer 0 ise, PC 1 arttırılır. Dolayısıyla bir sonraki gelen komut atlanmış olur.
- Bellekte doğrudan doğruya bir kelimenin arttırılması mümkün değildir. Bu sebepten; kelime DR'ye okunarak arttırma işlemi sağlanır.

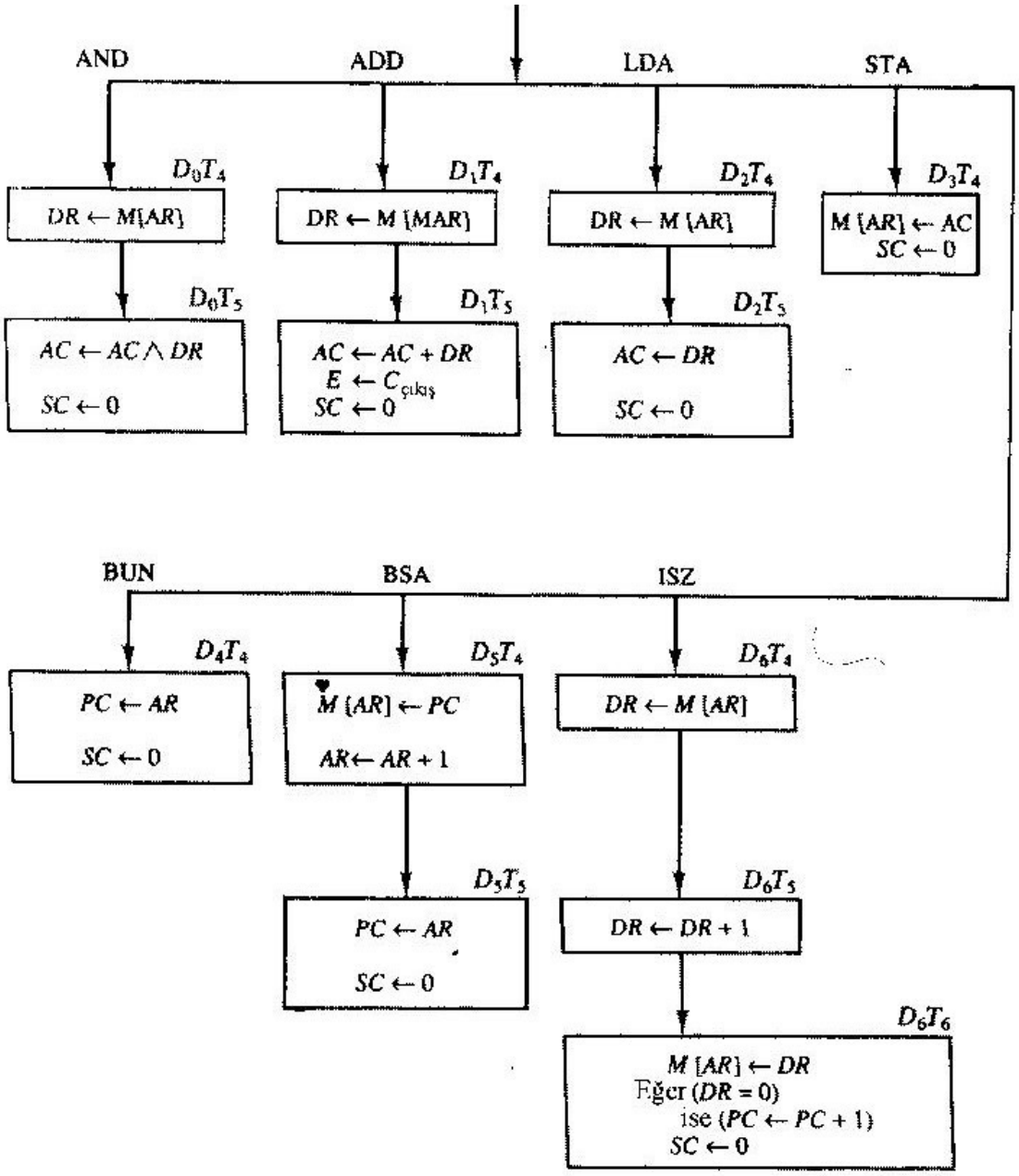
D6T4: DR←M[AR]

D6T5: DR←DR+1

D6T6: M[AR] ←DR, eğer (DR=0) ise (PC←PC+1), SC←0

Denetim Akış Şeması

7 adet bellek adreslemeli buyruğun icra edilmesi için gerekli mikroişlemler aşağıdaki şekilde verilmiştir.



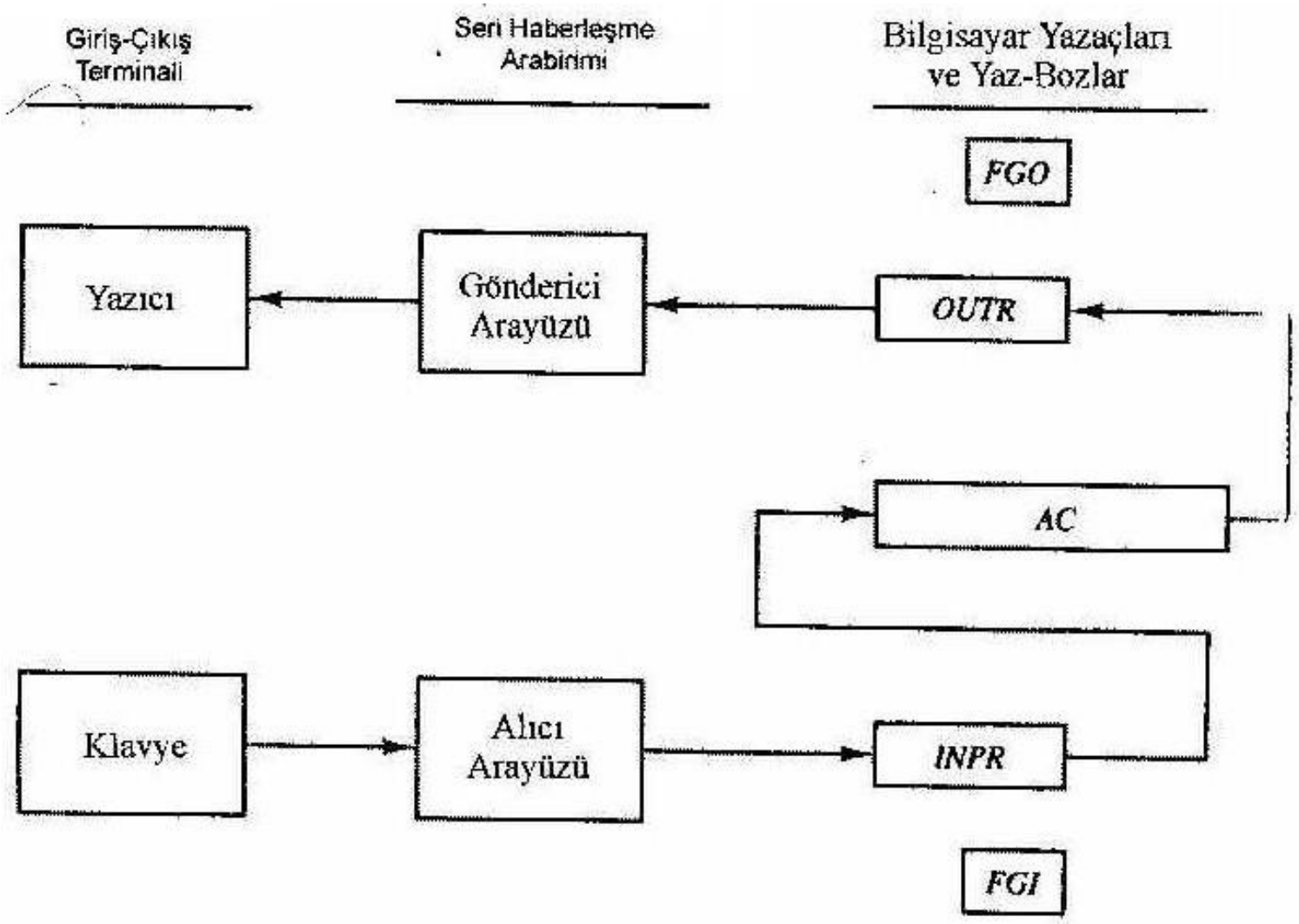
Şekil 5.11 Bellek adreslemeli buyruklar için akış şeması

Giriş/Çıkış Buyrukları ve kesmeler

- Dış çevre birimleriyle haberleşmek için kullanılır.Temel bilgisayarda , terminal,klavye ve yazıcıdan oluşan G/Ç birimleri olsun.Terminal seri bilgi alış verişi yapar (8 bitlik karakter bazında), klavyeden gelen 8 bit karakterler INPR giriş yazacına aktarılır.
- Yazıcı için seri bilgiler OUTR çıkış yazacında tutulur.Bu iki yazaç seri haberleşme arayüzü ile irtibatlıdır.Gönderici arayüzü ise seri bilgileri klavyeden alır ve INPR yazacına aktarır.alıcı arayüzü bilgiyi OUTR'den alır ve yazıcıya seri olarak gönderir.

Giriş/Çıkış Düzeni

FGO:Flag output, FGI:Flag ınput



Şekil 5.12 Giriş/Çıkış düzeni

Giriş/Çıkış Buyrukları

- G/Ç buyrukları bilginin AC'nin içine ve AC'den dışarıya aktarımı için kullanılır.Ayrıca bayrak bitleride eklenir.
- G/Ç buyruklarının işlem kodu,111 dir. D7=1 ve I=1 ile tanımlanır.Buyruğun kalan bitleri işlemin özel kodu için ayrılmıştır.
- Bu buyruklar T3 zaman sinyalini oluşturan clock geçişi ile icra edililer.

G/Ç buyrukları-I

- **G/Ç buyrukları verinin AC ün içine ve AC'den dışarıya aktarımı içi gereklidir Ayrıca bayrak bitleri ile denetlenir. Ayrıca kesmelerde, bu buyruklarla denetim yapılır. İşlem kodu 111 (7)'dir. D7=1 ve I=1 ile tanımlanır. Buyruğun kalan bitleri işlemin özel kodu için ayrılmıştır. Bu buyruklar, T3 zaman sinyalini oluşturan clock işaretinin yükselen kenarında oluşur. D7IT3 =1 denetim fonksiyonu ile icra edilir. SC sıfırlanır. INP buyruğu veriyi INPR yazacından, AC nin düşük 8 bitine aktarır.OUT buyruğu, AC'nin önemsiz 8 bitindeki veriyi OUTR yazacına aktarır. Ç.5.5'deki son iki buyruk IEN FF'unu 0 veya 1 yapmaktadır. IEN FF'nun amacı kesmeleri yönetmektir.**

Çizelge 5.5 Giriş /çıkış buyrukları

$D_7IT_3 = p$ (tüm giriş-çıkış buyruklarında ortak kullanılır).
$IR(i)$ = (IR(6-11) bitleri bunlar buyruklarla belli)

	p:	$SC \leftarrow 0$	Sil SC
INP	pB_{11}:	$AC(0\text{–}) \leftarrow INPR$, $FGI \leftarrow 0$	Giriş karakteri
OUT	pB_{10}:	$OUTR \leftarrow AC(0\text{–}7)$, $FGO \leftarrow 0$	Çıkış karakteri
SKI	pB_9:	Eğer ($FGI = 1$) ise ($PC \leftarrow PC + 1$)	Giriş bayrağını atla
SKO	pB_8:	Eğer ($FGI = 0$) ise ($PC \leftarrow PC + 1$)	Çıkış bayrağını atla
ION	pB_7:	$IEN \leftarrow 1$	Kesme aktif olacak
IOF	pB_6:	$IEN \leftarrow 0$	Kesme pasif olacak

Program Kesme

- G/Ç birimleriyle Bilgisayarın haberleşmesi,aktarımı 2 biçimde olabilir.

1- Programla aktarım denetimi: Bilgisayar, bayrak bitini devamlı olarak kontrol ederek, bayrak bitinin değiştiğini gördüğü zaman yaptığı işi bırakıp bilgi aktarımına başlar.

Örneğin, buyruk sürecinin 1 µsn olduğu bir bilgisayar, saniyede 10 karakter aktarabilen bir G/Ç ünitesine sahip olsun.Bu durumda 1 karakter 0.1sn=100 ms=100.000µsn'de taşınır.Her bir komut işlendikten sonra bayrağın kontrol edilemesi gerekeceğinden bir karakter aktarımı süresince 50.000 defa bayrağı kontrol etmekiçin zaman harcanmış olur.

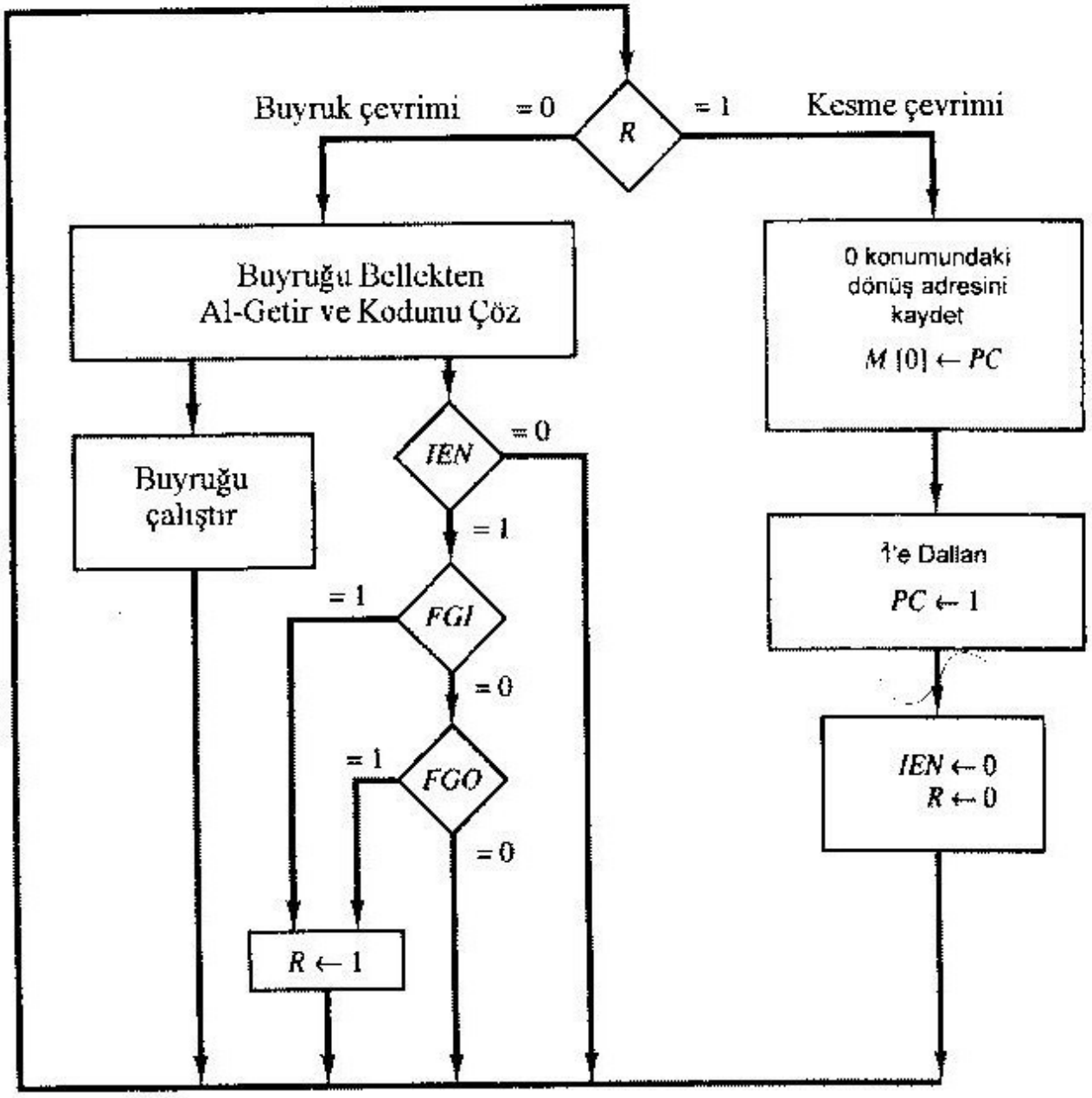
Oysa bu harcadığı süre zarfında başka işlerin yapılması gerekirdi.

Program kesme-2

- Kesme yöntemi :Programlama ile denetimin, zaman kaybınına yol açmasından ötürü,G/Ç ünitesinin aktarıma hazır olduğunu bilgisayara haber vermesi tekniği çok daha hızlıdır. Bu durumda bilgisayar başka işlerle uğraşabilir. Bilgisayar durum bayraklarını kontrol etmekle meşgul olmaz, bayrak bitleri durum değiştirdiğinde o anki icra ettiği programı durdurulup, bayrak bitinin 1 olduğu bilgisi oluşur,bilgisayar bu durumda G/ç aktarımı için gerekli önlemleri alır, G/Ç işlemi bitince kesmede programı kaldığı yerden çalışmaya devam eder.
- Kesme izin FF'u (İnterrupt enable -IEN) IOF ve ION buyruğu ile 0 veya 1 yapılabilir.Bu iki buyruk kullanıcıya, kesmeleri kullanıp kullanmama yetisi sağlar.

Kesme süresicinin işlemesi

R:kesme FF'u bilgisayara eklenmiştir, R=1 kesme süreci,R=0 buyruk süreci



Şekil 5.13 Kesme süreci için akış şeması

Kesme Sürecinin açıklaması

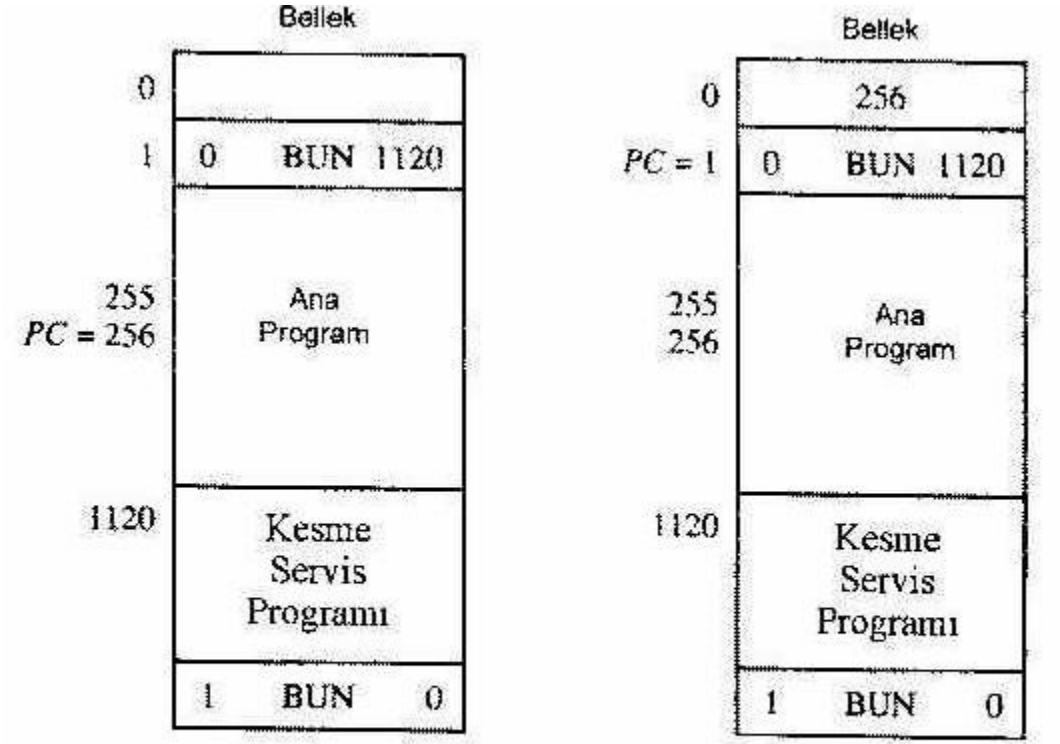
- 5.13'e göre, R FF'u 0 ise buyruk çevrimine devam edilir.Buyruğun fetch ce decode işlemi tamamlandıktan sonra, IEN devamlı olarak denetim birimi tarafından gözlenir. IEN 0 ise, programcı kesmeleri kullanmıyordur.Denetim bir sonraki buyruk sürecine gider. IEN 1 ise, denetim bayrak bitlerine (FGI, FGO) bakar.Eğer her ikiside 0 ise G/Ç yazaçlarının hazır olmadığı anlaşılır. Denetim bir sonraki buyruk sürecine geçer.
- IEN 1 ise,ve bayraklardan biri 1 ise R=1 yapılır. Kesme sürecine geçilir.

Kesme Süreci

- Bu süreç, dallanma ve geri dönüş işleminin donanımla yapılmış bir çeşididir.
- Bir kesme sürecinde yapılan işler:
- PC içindeki dönüş adresi, kolayca bulunabilecek bir adrese yerleştirilir.Bu yer bir işlemci yazacı veya bellekte bir adres olabilir. Burada 0 adresi, dönüş adresinin saklanığı yer olarak alınmıştır.
- Daha sonra, denetim PC'a 1 adresini yazar.R'yi ve IEN'i 0 yapar.Böylece bir kesme isteği bitirilene kadar başka kesmelerin olmasına izin verilmez.
- Şek.5.14 kesme sürecindeki işlemleri gösterir.

Kesme sürecinin başlangıcında neler oluyor?

- 255 adresindeki komut icra edilirken kesme isteği oluşsun (R=1).
- PC'nin içeriğinde 256 adresi kesmeden sonraki dönülecek adrestir. 1 adresinde BUN 1120 buyruğu vardır.
- Programcı kesme işleminde yapacağı işi 1120 adresinden başlatacağını belirtmiştir.
- T0 zaman den.sinyali geldiğinde R=1 'dir.Denetim kesme sürecini başlatır.
- PC'deki 256 adresi 0. belleğe yazılır. PC'a 1 yazılır. R=0 olur.
- Bir sonra işlenen buyruk, denetimi 1120 adresindeki G/Ç hizmet programına taşır. Gerekeni yapar.
- Esasa programın, kalındığı yere döndürülmesini sağlayan buyruk adres kısmında 0 bulunan dolaylı bir dallanma buyruğudur. Ve kesme hizmet programının sonunda bulunur.
- Bu buyruk bellekten okunur. I=1 olduğundan dolaylı buyruk olduğu anlaşılır.
- Etkin adres 0'dır. Burada da ana programın kaldığı buyruğun adresi yazılıdır.
- Dlaylı BUN buyruk icrasıyla PC'a 0 adresindeki dönüş adresi yaılmış olur.



Şekil 5.14 Kesme süreci gösterimi

KESME sürecinde neler oluyor?

- Kesme süreci, Kesme FF'unun 1 yapılmasından sonraki icra evresi bitince başlar.
- R FF'u IEN=1 ve bayraklardan birisi aktif olunca 1 olur. R'nin 1 yapılması herhangi bir zamanda (herhangibir T_i de - T_0, T_1, T_2 hariç) olabilir.
- R'nin 1 yapılma şartı: $T'_0 T'_1 T'_2$ (IEN)(FGI+FGO): R←1

Kesme'deki Al-Getir (FETCH) ve (Kod Çöz) Decode İşlemi nasıl olur?

Normal FETCH ve DECODE işlemleri T_0, T_1, T_2 zaman sinyalleriyle oluyordu. Kesme işlemi için FETCH ve DECODE işlemi $R'T_0$, $R'T_1, R'T_2$ denetim fonksiyonlarıyla oluşur. Sebebi, buyruğun icrasından sonra SC=0 yapılmasıdır. Denetim, FETC evresine R=0 ile gider, R=1 ise denetim kesme evresine gider.

Kesme süreci, PC içindeki dönüş adresi 0'a yerleştirilir. 1 bellek adresine dallanılır. IEN, R, ve SC temizlenir.

KESME sürecinde neler oluyor-2?

- Bu işlemler, aşağıdaki mikroişlemler ile olur,

RT0: AR ← 0, TR ← PC

RT1: M[AR] ← TR, PC ← 0

RT2: PC ← PC + 1, IEN ←0, R←0, SC ←0

1.zaman sinyalinde AR sıfırlanır. PC'nin içeriği TR yazacına aktarılır.

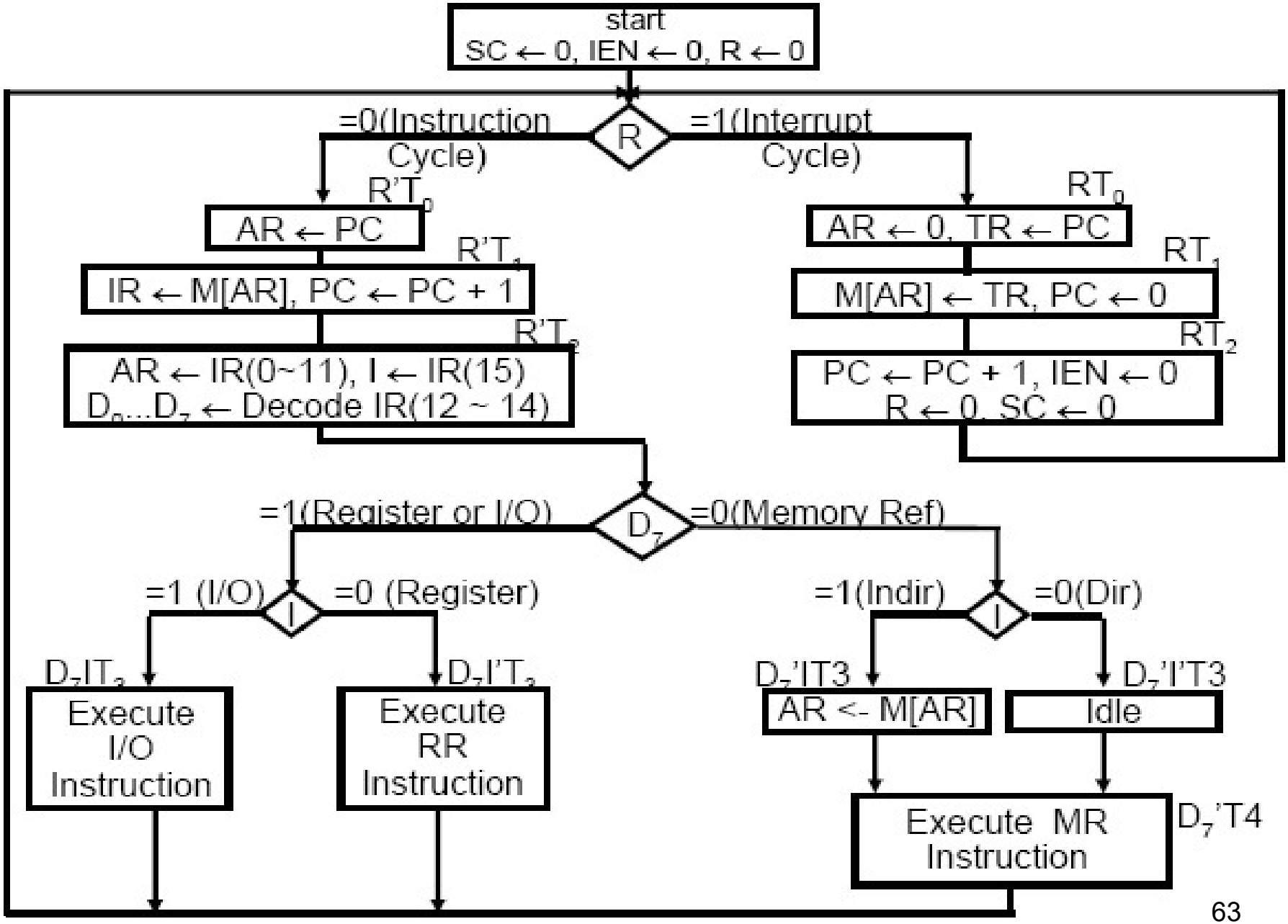
2. zaman sinyalinde dönüş adresi 0. belleğe yerleştirilir. PC sıfırlanır.

3. Zaman sinyalinde PC 1 arttırılır. R ve IEN sıfırlanır. Denetim tekrar T_0 zamanına döner. Çünkü SC sıfırlanır.

Bu buyruk sürecinin başında R'To şartı vardır. Ve PC'un içeriği 1'dir. Denetim buyruk sürecine girer. 1 adresindeki BUN buyruğunu icra eder.

Bilgisayarın tanımlanmış tanımı

- Kesme sürecini de içeren, buyruk süreçleri Şekil 5.15'teki akış şemasında verilmiştir.
- R , dolaylı veya icra evrelerinin herhangibir anında 1 yapılabilir.
- SC=0 ise, denetim T0 zaman sinyaline döner. Eğer R=1 ise, bilgisayar kesme sürecine, R=0 ise buyruk sürecine gider. Buyruk tipi belirlenir ve ilgili mikroişlemler ile buyruğun gereği yapılır.
- Tüm bilgisayar için deneyim fonksiyonları ve mikroişlemler Ç.5.6'da özetlenmiştir.
- Yazaç aktarım deyimleri bilgisayarın iç yapısını özetler. Bunlar aynı zamanda lojik develerin tasarım bilgilerini de verir
- Denetim fonksiyonları ve şartlı denetim deyimleri de, fonksiyon denklemlerinin oluşturulması için gerekir. Bu denklemler denetim birimini oluşturur.


start
SC ← 0, IEN ← 0, R ← 0
R
=0(Instruction Cycle)
=1(Interrupt Cycle)
R'T0
AR ← PC
R'T1
IR ← M[AR], PC ← PC + 1
R'T2
AR ← IR(0~11), I ← IR(15)
D0...D7 ← Decode IR(12 ~ 14)
RT0
AR ← 0, TR ← PC
RT1
M[AR] ← TR, PC ← 0
RT2
PC ← PC + 1, IEN ← 0
R ← 0, SC ← 0
D7
=1(Register or I/O)
=0(Memory Ref)
I
=1 (I/O)
=0 (Register)
I
=1(Indir)
=0(Dir)
D7IT3
Execute I/O Instruction
D7I'T3
Execute RR Instruction
D7'IT3
AR <- M[AR]
D7'I'T3
Idle
Execute MR Instruction
D7'T4

Çizelge 5.6 Temel bilgisayar için denetim fonksiyonu ve Mikro işlemler.

Al getir:	RT_0:	$AR \leftarrow PC$
	RT_1:	$IR \leftarrow M[AR],\ PC \leftarrow PC + 1$
Kodunu çöz	RT_2:	$D_0, \ldots, D_7 \leftarrow$ kodunu çöz IR (12–14) $AR \leftarrow IR$ 0–11), $I \leftarrow IR(15)$
Dolaylı	$D'7IT_3$:	$AR \leftarrow M[AR]$
Kesme:		
$T_0T_1T_2(IEN)\ (FGI + FGO)$:		$R \leftarrow 1$
	RT_0:	$AR \leftarrow 0,\ TR \leftarrow$ PC
	RT_1:	$M[AR] \leftarrow TR,\ PC \leftarrow 0$
	RT_2:	$PC \leftarrow PC + 1,\ IEN \leftarrow 0,\ R \leftarrow 0,\ SC \leftarrow 0$
Bellek adreslemeli buyruklar		
ADD	D_0T_4:	$DR \leftarrow M[AR]$
	D_0T_5:	$AC \leftarrow AC \wedge DR,\ SC \leftarrow 0$
	D_1T_4:	$DR \leftarrow M[AR]$
	D_1T_5:	$AC + DR,\ E \leftarrow C_{çıkış},\ SC \leftarrow 0$
LDA	D_2T_4:	$DR \leftarrow M[AR]$
	D_2T_5:	$AC \leftarrow DR,\ SC \leftarrow 0$
STA	D_3T_4:	$M[AR] \leftarrow AC,\ SC \leftarrow 0$
BUN	D_4T_4:	$PC \leftarrow AR,\ SC \leftarrow 0$
BSA	D_5T_4:	$M[AR] \leftarrow PC,\ AR \leftarrow AR + 1$
	D_5T_5:	$PC \leftarrow AR,\ SC \leftarrow 0$
ISZ	D_6T_4:	$DR \leftarrow M[AR]$
	D_6T_5:	$DR \leftarrow DR + 1$
	D_6T_6:	$M[AR] \leftarrow DR$, eğer $(DR = 0)$ ise $(PC \leftarrow PC + 1),\ SC \leftarrow 0$
Yazaç adreslemeli buyruklar		
	$D_7IT_3 = r$ (tüm yazaç adreslemeli buyruklarda ortak)	
	$IR(i) = B_i\ (i = 0, 1, 2, \ldots, 11)$	
	r:	$SC \leftarrow 0$
CLA	rB_{11}:	$AC \leftarrow 0$
CLE	rB_{10}:	$E \leftarrow 0$
CMA	rB_9:	$AC \leftarrow \overline{AC}$
CME	rB_8:	$E \leftarrow \overline{E}$
CIR	rB_7:	AC Shr $AC,\ AC(15) \leftarrow E,\ E \leftarrow AC(0)$
CIL	rB_6:	AC Shl $AC,\ AC(0) \leftarrow E,\ E \leftarrow AC(15)$
INC	rB_5:	$AC \leftarrow AC + 1$
SPA	rB_4:	Eğer $(AC(15) = 0)$ ise $(PC \leftarrow PC + 1)$
SNA	rB_3:	Eğer $(AC(15) = 1)$ ise $(PC \leftarrow PC + 1)$
SZA	rB_2:	Eğer $(AC = 0)$ ise $(PC \leftarrow PC + 1)$
SZE	rB_1:	Eğer $(E = 0)$ ise $(PC \leftarrow PC + 1)$
HLT	rB_0:	$S \leftarrow 0$
Giriş çıkış buyrukları		
	$D_7IT_3 = p$ (tüm giriş-çıkış buyruklarında ortak)	
	$IR(i) = B_i\ (i = 6, 7, 8, 9, 10, 11)$	
	p:	$SC \leftarrow 0$
INP	pB_{11}:	$AC(0–7) \leftarrow INPR,\ FGI \leftarrow 0$
OUT	pB_{10}:	$OUTR \leftarrow AC(0–7),\ FGO \leftarrow 0$
SKI	pB_9:	Eğer $(FGI = 1)$ ise $(PC \leftarrow PC + 1)$
SKO	pB_8:	Eğer $(FGO = 1)$ ise $(PC \leftarrow PC + 1)$
ION	pB_7:	$IEN \leftarrow 1$
IOF	pB_6:	$IEN \leftarrow 0$

TEMEL BİLGİSAYARIN TANIMI

- Donanım bileşenleri:

1- 16 bit 4096 kelimelik bellek

2- 9 yazac; AR,PC,DR,AC,IR,TR,OUTR,INPR,SC

3- 7 FF; I,S,E,R,IEN,FGI,FGO

4- 3x8 ve 4x16'lık(bu zamanlama için kullanılır) kod çözücü

5-16 bit ortak veri yolu

6-AC'nin girişine konan toplayıcı ve mantık devreleri.

7-Denetim mantık kapıları

Bunlardan ilk 6 sının yapılarını biliyoruz. Denetim mantık kapılarının tasarımlarını inceleyelim.

DENETİM MANTIK KAPILARI

- Blok şeması Ş.5.6'da verilmiştir.

Girişleri:

1-3x8 ve 4x16'lık kod çözücü çıkışları, I FF'u ve IR'nin 0-11 bitleri,

2-AC'nin 0-15. bitleri (AC'nin 0 olup olmadığına veya AC'nin işaret bitini AC(15) kontrol için)

3-0-15 arası DR bitleri (DR 'nin 0 olup olmadığına ve 7 FF'un değerlerine bakmak için)dir.

Denetim Biriminin çıkışları:

1-9 yazacın girişlerini kontrol eden sinyaller

2- Belleğin oku ve yaz girişlerini kontrol eden sinyaller

3-FF'ları 1,0 yapan ve tümleyen sinyaller

4- S2,S1,S0'ı kontrol ederek veri yolu için bir yazaç seçen sinyaller.

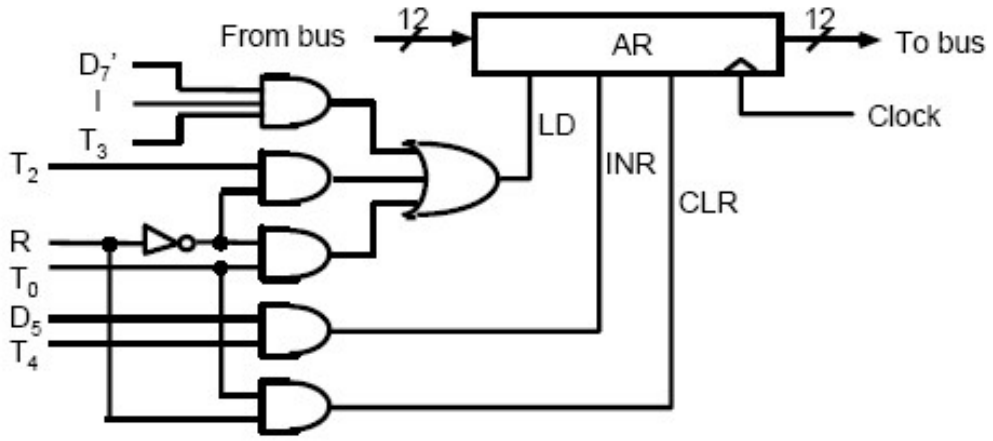
5- AC'nin toplayıcı ve mantık devrelerini kontrol eden sinyaller.

YAZACIN VE BELLEĞİN DENETİMİ
örnek: AR yazacı denetim girişleri tasarımı

- Çizelge 5.6'yı tarayarak AR'nin içeriğini değiştiren tüm deyimlere göre, denetim devresi tasarımı(Şk.5.16)

$R'T_0$:	$AR \leftarrow PC$	LD(AR)
$R'T_2$:	$AR \leftarrow IR(0\text{-}11)$	LD(AR)
$D_7'IT_3$:	$AR \leftarrow M[AR]$	LD(AR)
RT_0:	$AR \leftarrow 0$	CLR(AR)
D_5T_4:	$AR \leftarrow AR + 1$	INR(AR)

$$LD(AR) = R'T_0 + R'T_2 + D'_7IT_3$$
$$CLR(AR) = RT_0$$
$$INR(AR) = D_5T_4$$



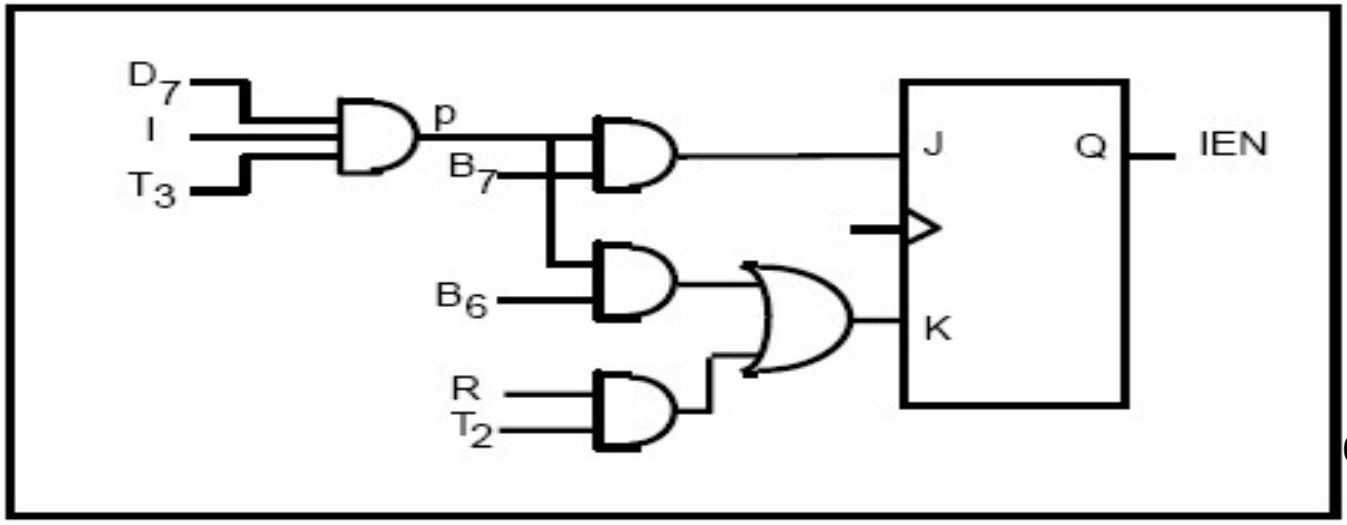
TEK FF'ların DENETİMİ

- 7 FF'nun denetim kapıları benzer biçimde tasarlanır. Örneğin Ç.5.6'da IEN'in ION ve IOF buyrukları ile değiştiği görülmektedir. B7 ve B6 IR'nin 6.ve 7.bitleridir. Buna göre kurulan devre (Şkl.5.17'dir)

IEN: Interrupt Enable Flag

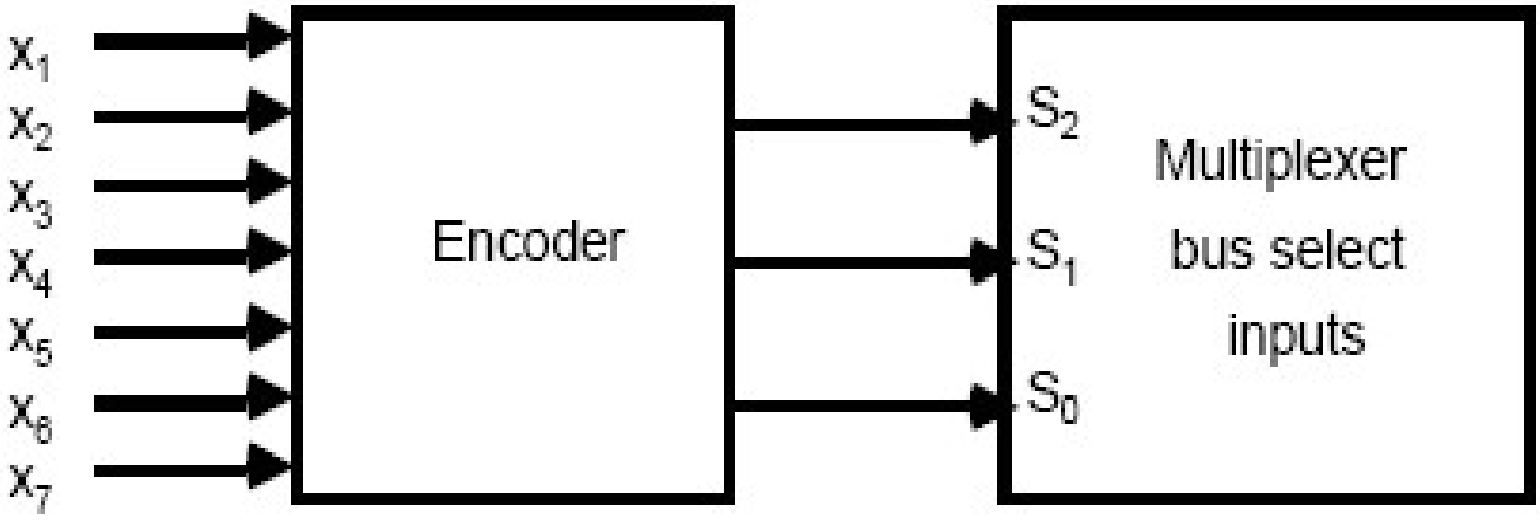
pB_7: IEN ← 1 (I/O Instruction)
pB_6: IEN ← 0 (I/O Instruction)
RT_2: IEN ← 0 (Interrupt)

$p = D_7IT_3$ (Input/Output Instruction)



ORTAK VERİ YOLUNUN DENETİMİ

- Şek.5.6'daki 16 bitlik ortak veri yolu S_2,S_1,S_0 seçme girişleriyle kontrol ediliyordu. Kodlayıcı girişindeki $x_1 \ldots x_7$ değişkenler veri yolu için yapılacak seçimde yazaç veya belleğin no'su olup, örneğin x_1=1 ise S2S1S0= 001 olarak AR registerini seçmelidir. (Şek.5.18)



ORTAK VERİ YOLUNUN DENETİMİ-2

- Çizelge 5.7 ikili kodlayıcı için doğruluk tablosunu verir.Kodlayıcının fonksiyon denklemi ise;
- $S_0 = x_1+x_3+x_5+x_7$
- $S_1 = x_2+x_3+x_6+x_7$
- $S_2 = x_4+x_5+x_6+x_7$

x_1	x_2	x_3	x_4	x_5	x_6	x_7	S_2	S_1	S_0	selected register
0	0	0	0	0	0	0	0	0	0	none
1	0	0	0	0	0	0	0	0	1	AR
0	1	0	0	0	0	0	0	1	0	PC
0	0	1	0	0	0	0	0	1	1	DR
0	0	0	1	0	0	0	1	0	0	AC
0	0	0	0	1	0	0	1	0	1	IR
0	0	0	0	0	1	0	1	1	0	TR
0	0	0	0	0	0	1	1	1	1	Memory

ORTAK VERİ YOLUNUN DENETİMİ-3

- Peki x girişleri nasıl belirlenecek?
- Bunun için istenen yazacı veriyoluna bağlamak için gerekli denetim fonksiyonlarını bulmak gerekir.Bu işlem için çizelge 5.6'daki yazaç ve bellek aktarım deyimleri taranarak ilgili denetim fonksiyonlarından bir fonksiyon denklemi elde edilmelidir.
- Örneğin X1=1 olması için (yani AR reg.ini veri yoluna bağlamak için) AR'yi kaynak alan şu denetim fonksiyonları bulunur.

D_4T_4: PC←AR

D_5T_5: PC←AR

$X_1 = D_4T_4 + D_5T_5$

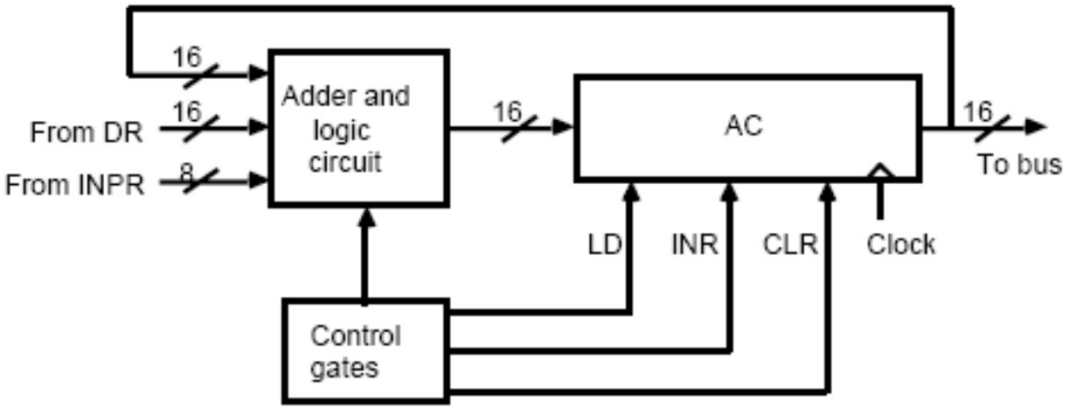
Örn: Bellekten veri okumak için x_7=1 $S_2S_1S_0$=1 olmalıdır. Buna göre

$X7 = R'T_1 + D'_7IT_3 + (D_0+D_1+D_2+D_6)T_4$

Fonk.denklemi, Ç.5.6 yardımıyla elde edilir.

İŞLEMCİ YAZACI MANTIK TASARIMI

AC ile ilgili mantığı kurmak için Çizelge5.6'daki yazaç aktarım deyimlerine bakılırsa

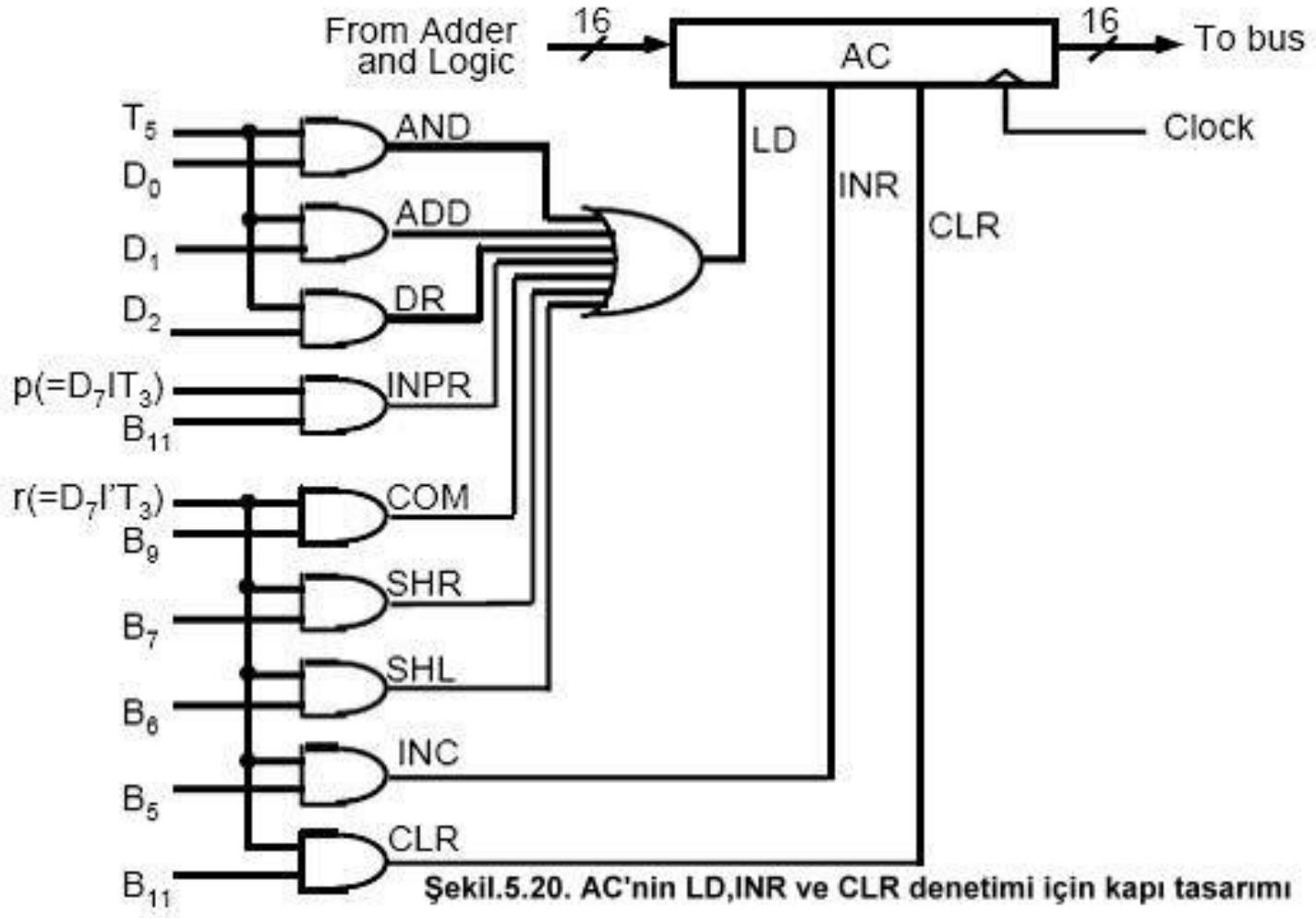


All the statements that change the content of AC :

D_0T_5:	AC ← AC ∧ DR	AND with DR
D_1T_5:	AC ← AC + DR	Add with DR
D_2T_5:	AC ← DR	Transfer from DR
pB_{11}:	AC(0-7) ← INPR	Transfer from INPR
rB_9:	AC ← AC′	Complement
rB_7 :	AC ← shr AC, AC(15) ← E	Shift right
rB_6 :	AC ← shl AC, AC(0) ← E	Shift left
rB_{11} :	AC ← 0	Clear
rB_5 :	AC ← AC + 1	Increment

AC yazacının denetimi

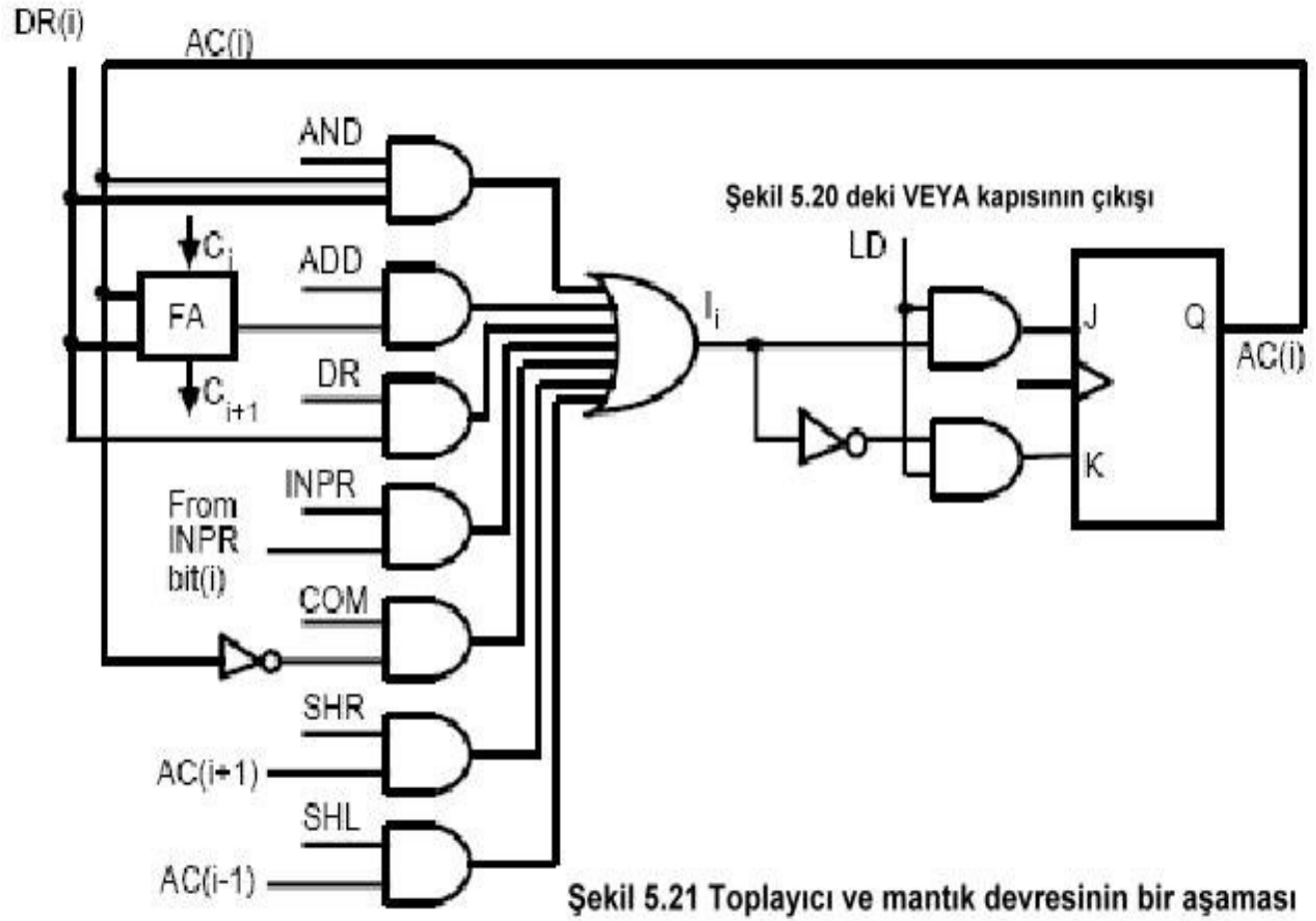
- **AC yazacının LD, INCR ve CLR girişlerini kontrol eden mantık devreleri , yukarıdaki denetim fonksiyonlarından türetilir.**



Şekil.5.20. AC'nin LD,INR ve CLR denetimi için kapı tasarımı

Toplayıcı ve mantık Devre

- **Toplayıcı ve mantık devresi 16 eşit alt parçaya ayrılır. Her bir parça AC'nin 1 bitine karşılık gelir.**
- **Toplayıcı mantık devresinin 1 parçası için mantık devresi aşağıdaki şekilde verilmiştir.**



Şekil 5.20 deki VEYA kapısının çıkışı

Şekil 5.21 Toplayıcı ve mantık devresinin bir aşaması