

# Bilgisayar Mimarisi

## *Bölüm 5*

### *Temel Bilgisayar Tasarımı*

Dr. Emre Ünsal

Cumhuriyet Üniversitesi

Yazılım Mühendisliği Bölümü

# İçerik

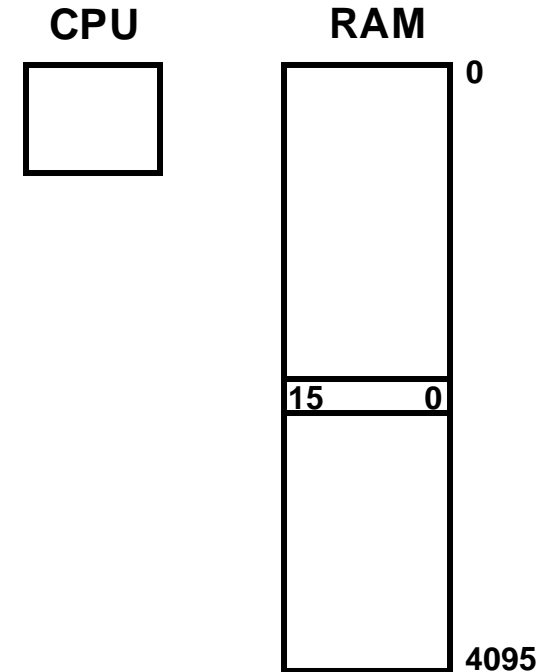
- Giriş
- Temel Bilgisayar
- Bilgisayar Buyrukları
- Adresleme Modları
- Bilgisayar Yazaçları
- Ortak Veri Yolu
- Bilgisayar Buyrukları
- Buyruk Süreçleri
- Kesme Döngüleri
- Temel Bilgisayarın Tamamlanmış Tanımı

# Giriş

- Bu bölümde temel bilgisayar tanıtılacak ve işleyişi yazaç aktarım dili ile gösterilecektir.
- Bilgisayarın tasarımı, iç yazaçları, zamanlama ve denetimi kullanılan buyruk kümesi ile tanımlanacaktır.
- Her farklı işlemci tipi kendi tasarımına sahiptir (Yazaçlar, veriyolları ve buyruklar farklılık gösterir.)
- İşlemcilerin nasıl çalıştığını anlamak için basitleştirilmiş bir işlemci modeliyle başlayacağız. Bu modele M. Morris Mano'nun **Temel Bilgisayarı** adını vereceğiz.
- Modern işlemciler ise oldukça karmaşıklardır. İçerisinde:
  - Çok sayıda yazaç
  - Farklı türlerde Aritmetik Lojik Birimler ve hesaplayıcı devreler barındırmaktadır.

# Temel Bilgisayar

- Temel Bilgisayarın iki bileşeni vardır; işlemci ve bellek.
  - Hafızanın içerisinde **4096** satır (word) bulunmaktadır.
  - $4096 = 2^{12}$  Hafızadaki her bir adrese erişebilmek için **12 bitlik bir adres** satırına ihtiyaç vardır.
  - Her bir hafıza satırı ise **16 bit** uzunluğundadır.

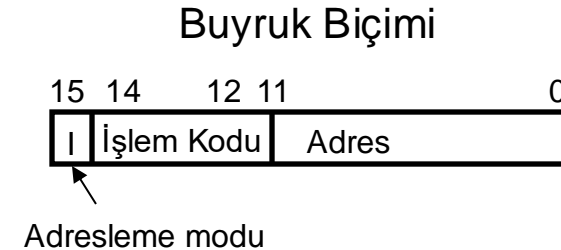


# Bilgisayar Buyrukları

- Program
  - Ardışık olarak çalışan bilgisayar buyruklarından oluşmaktadır.
- Program buyrukları gerekli olan tüm veriyle birlikte hafızada saklanır.
- İşlemci sonraki buyruğu hafızadan okuyacaktır.
  - Hafızadan okunan büyük **Instruction Register (IR)** içerisinde tutulur.
- Kontrol ünitesindeki kontrol devresi daha sonra komutu uygulamak için gerekli olan mikro işlemleri sırasıyla çalıştırır.

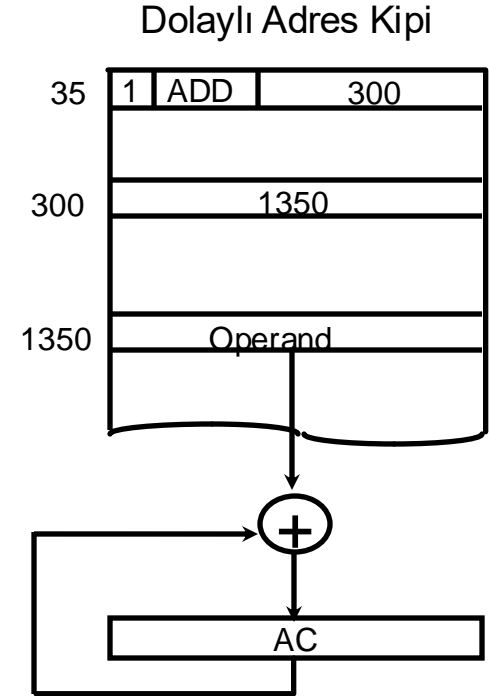
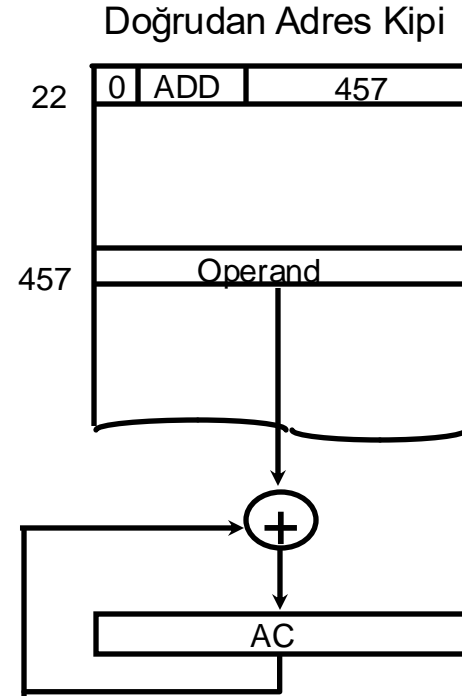
# Buyruk Biçimi

- Bir bilgisayar buyruğu genellikle iki bölüme ayrılır:
  - 4 Bitlik İşlem Kodu (Opcode)
  - 12 bitlik Adres kısmı.
  - Toplamda 16 bitten oluşmaktadır.
- Temel Bilgisayarda 15. bit ise Adresleme kipi için kullanılmaktadır.
  - **I=0** Doğrudan adresleme
  - **I=1** Dolaylı adresleme kipi
- **Ani buyruk** ise IR adres kısımda direk olarak veri bulunmaktadır. Bu yapıya Derhal yada Ani Buyruk adı verilir.



# Adresleme Modları

- Buyruk Yazacı (IR) içerisindeki adres kısmı
  - **Doğrudan Adres** (Direct Address): Kullanılacak verinin hafızadaki adresi buyruk adres satırlarında saklanır
  - .
  - **Dolaylı Adres** (Indirect Address): Verinin bulunduğu hafıza adresinin, hafıza adresi buyruk adres satırlarında göstermektedir.
  - **Etkin Adres** (Effective Address - EA): verinin bulunduğu gerçek adresi temsil eder.
    - Doğrudan Adresleme kipi için EA=457
    - Dolaylı Adresleme için EA=1350



# Bilgisayar Yazacıları

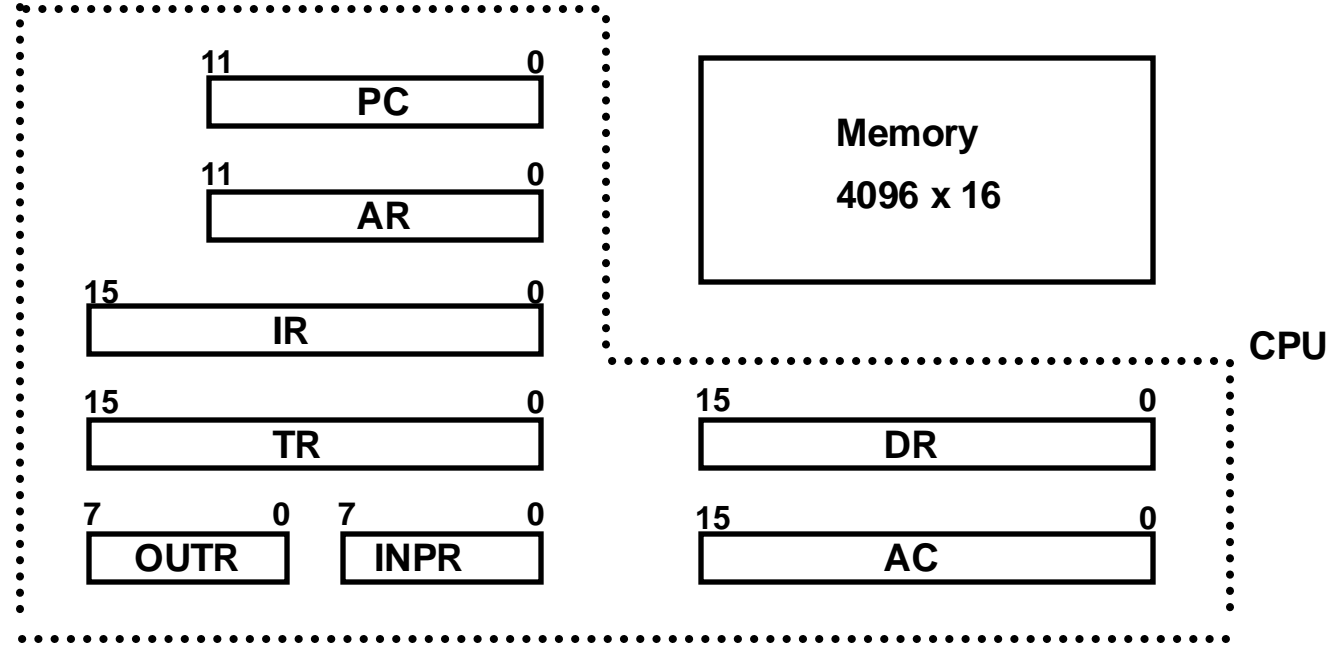
- Bir işlemcinin buyrukları, adresleri, verileri vb. saklamak için bir çok yazaca ihtiyacı vardır.
- **Program Sayıcı** (Program Counter - PC): İşlemci tarafından işletilecek bir sonraki buyruğun hafıza adresini tutmakla görevlidir.
  - Hafıza boyutu 4096 bit olduğu için PC sadece 12 bite ihtiyacı vardır.
- **Adres Yazacı** (Address Register): Adresleme kipine bağlı olarak hafıza adreslerinin saklandığı yazacıdır.
- **Veri Yazacı** (Data Register): Adres Yazacının gösterdiği adres satırlarındaki verinin tutulduğu yazacıdır. İşlemci daha sonra bu veriyi işlemek için kullanacaktır.
- **Akümülatör** (Accumulator - AC): Temel Bilgisayarda kullanılan tek genel amaçlı yazacıdır.



# Bilgisayar Yazacıları

- **Geçici Yazac** (Temporary Register - TR): Bazı buyruk işlemlerinde geçici veriyi saklamak için kullanılır.
- **Giriş Yazacı** (Input Register - INPR): İşlemciye dışarıdan (I/O) gelen 8 bitlik karakteri saklamak için kullanılır.
- **Çıkış Yazacı** (Output Register - OUTR): İşlemcinin çıkışındaki 8 bitlik karakteri saklamak için kullanılır.

# Temel Bilgisayar Yazacıları

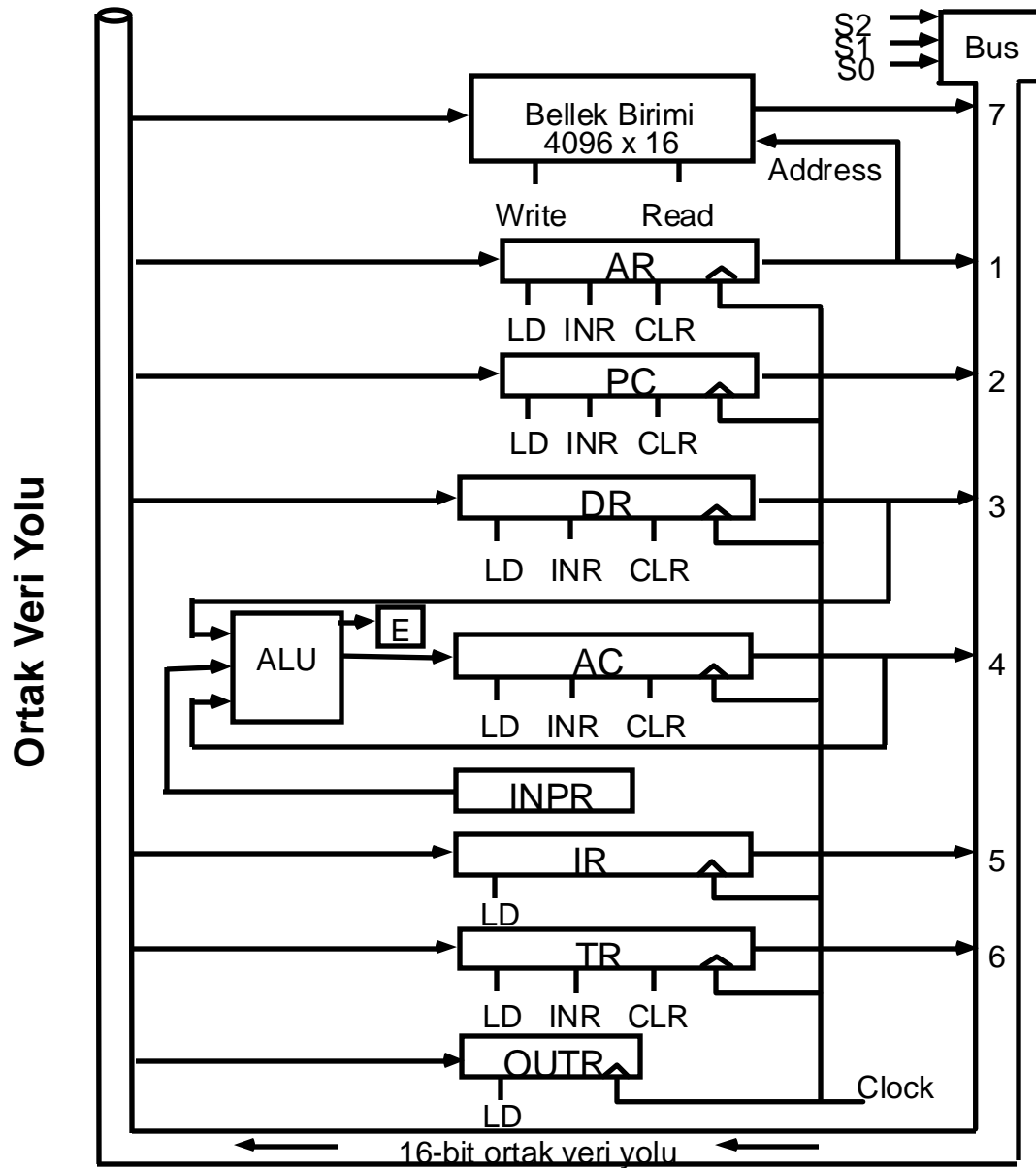


## List of BC Registers

DR	16	Veri Yazacı	Bellek verisini tutar
AR	12	Adres Yazacı	Bellek adresini tutar
AC	16	Akümülatör	İşlemci Yazacı
IR	16	Buyruk Yazacı	Buyruk Kodunu tutar
PC	12	Program Sayıcı	Buyruğun adresini tutar
TR	16	Geçici Yazac	Geçici veriyi tutar
INPR	8	Giriş Yazacı	Giriş Karakterini tutar
OUTR	8	Çıkış Yazacı	Çıkış Karakterini tutar

# Ortak Veri Yolu Sistemi

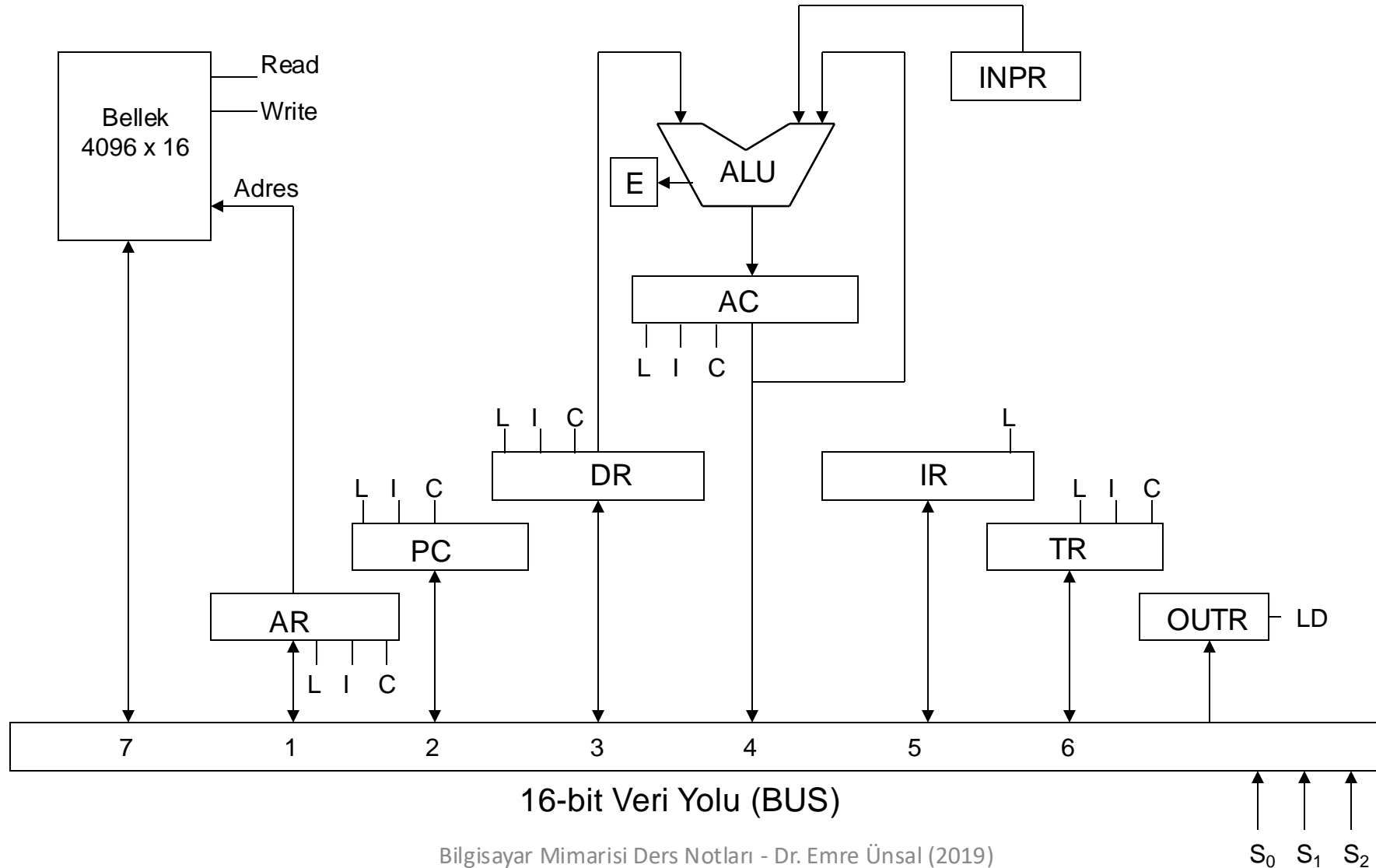
- Temel bilgisayarımızın 8 adet yazacı vardır.
- Yazaçlar arası ve yazaçlar ile bellek arasında verileri aktarabileceğimiz ortak yollara ***Veri Yolu Sistemi*** adı verilir.
- INPR direk olarak Artimetik ve Mantık birimine girdi olarak bağlanır. Bu nedenle Ortak veri yoluna 7 adet yazaç ve 1 adet bellek erişimi vardır.
- 8 adet birimi indekslenmek ve ortak veri yolunun erişimini kontrol etmek için 3 adet seçici  $S_2S_1S_0$  seçicileri kullanılmaktadır.



### Yazaç Kontrol Sinyalleri

- LD (Load): Yükleme
- INR (Increment): Arttırma
- CLR (Clear): Silme

# Ortak Veri Yolu



# Veri Yolu Seçimi

- Bellek yada yazaçların hangisinin veri yolunu kullanacağı 3 adet seçim biti ile kontrol edilir. ( $S_2S_1S_0$ )

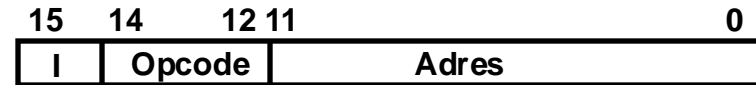
$S_2$	$S_1$	$S_0$	Yazaç
0	0	0	x
0	0	1	AR
0	1	0	PC
0	1	1	DR
1	0	0	AC
1	0	1	IR
1	1	0	TR
1	1	1	Memory

- 12 bitlik, AR ve PC yazaçları veri yoluna yüklenilen (16-bit) en yüksek 4 biti 0 olarak doldurulur.
- 8 bit OUTF yazacının bitleri Ortak veri yolunun ilk 8 biti ile yüklenir.

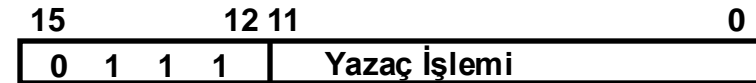
# Bilgisayar Buyrukları

- Temel Bilgisayar Buyruk Biçimleri

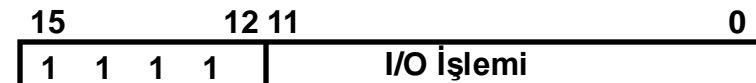
## Bellek Adreslemeli Buyruk (OP-code = 000 ~ 110)



## Yazaç Adreslemeli Buyruk (OP-code = 111, I = 0)



## Giriş-Çıkış Buyruğu (OP-code = 111, I = 1)



# Temel Bilgisayar Buyrukları

Sembol	Hex Kodu		Tanımlama
	I = 0	I = 1	
AND	0xxx	8xxx	Bellek kelimesini AC ile VE leme
ADD	1xxx	9xxx	Bellek kelimesini ile AC toplama
LDA	2xxx	Axxx	Bellek kelimesini AC ye yükle
STA	3xxx	Bxxx	AC yi belleğe yaz
BUN	4xxx	Cxxx	Şartsız Dallan
BSA	5xxx	Dxxx	Dallan ve geri dönüş adresini sakla
ISZ	6xxx	Exxx	Arttır, Eğer sıfır ise atla
CLA	7800		AC yi Sil
CLE	7400		E yi Sil
CMA	7200		AC nin tümleyenini al
CME	7100		E nin tümleyenini al
CIR	7080		AC ve E yi dairesel sağa kaydır
CIL	7040		AC ve E yi dairesel sola kaydır
INC	7020		AC yi 1 arttır
SPA	7010		AC pozitif ise sonraki buyruğu atla
SNA	7008		AC negatif ise sonraki buyruğu atla
SZA	7004		AC sıfır ise sonraki buyruğu atla
SZE	7002		E biti 0 ise sonraki buyruğu atla
HLT	7001		Programı durdur
INP	F800		Giriş karakterini AC ye al
OUT	F400		AC den çıkış karakterini al
SKI	F200		Giriş bayrağını atla
SKO	F100		Çıkış bayrağını atla
ION	F080		Kesmeyi aktif yap
IOF	F040		Kesmeyi pasif yap



# Buruk K mesinin Tamamlılığı

- Bir bilgisayarda bir buyruk k mesi bulunur. Bu buyruklar yardımı ile kullanıcı makine dili (assembly) kodları ile  şlenebilir fonksiyonlar yazabilir.
- Bilgisayarın buyruk k mesinde a ağıdaki sınıflarada yeterince buyruk varsa TAM denir.

- 1. Aritmetik, Mantık ve Kaydırma buyrukları**

- ADD, CMA, INC,CIR, AND, CLA, ...

- 2. Transfer (Bellek ve Yaza lar i in) Buyrukları**

- LDA, STA, ...

- 3. Durum Kontrol Buyrukları**

- BUN, BSA, ISZ, ...

- 4. Giri / ıkı  (G/ ) Buyrukları**

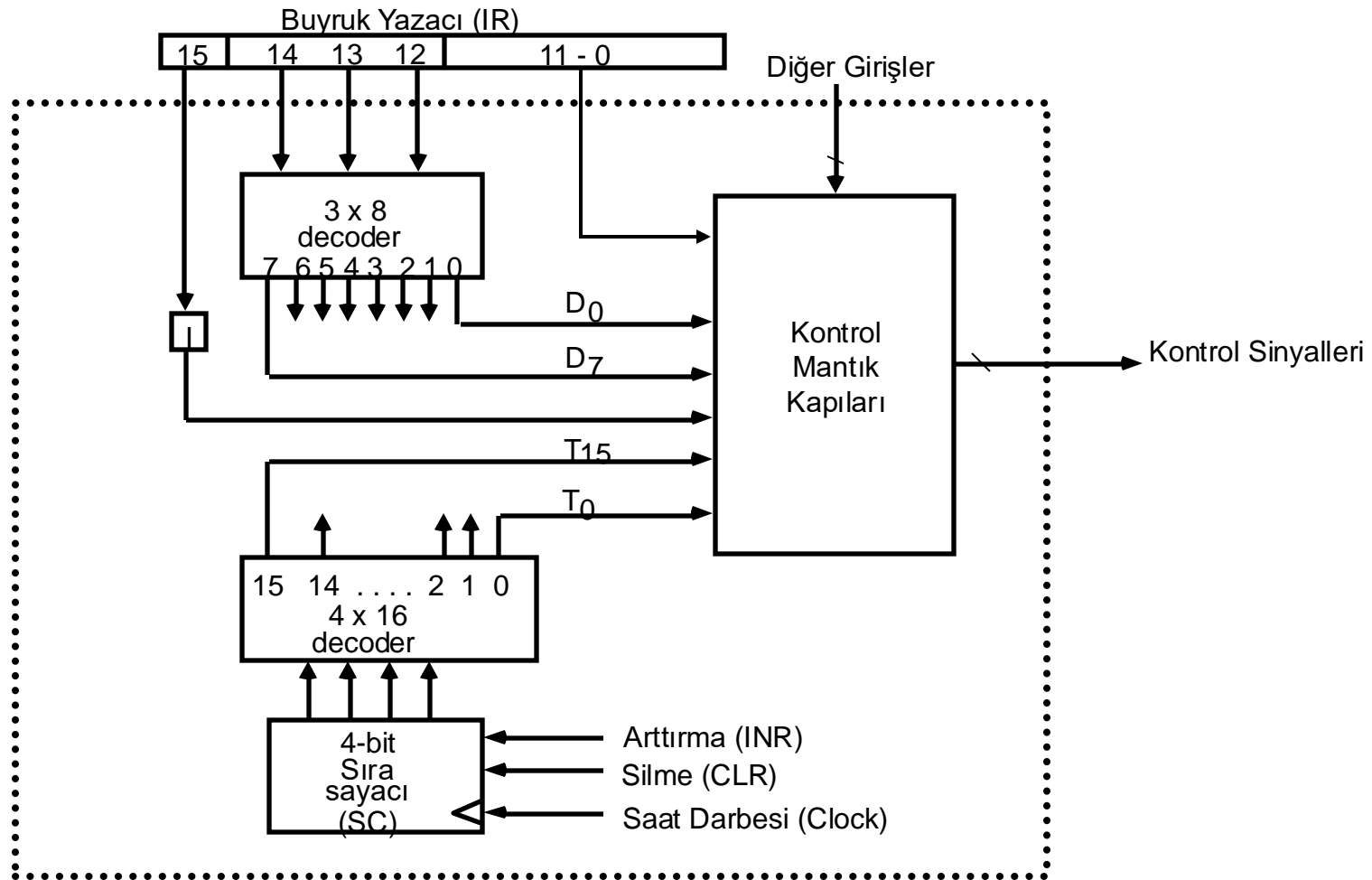
- INP, OUT, ...

# Kontrol Ünitesi

- Bir işlemcinin kontrol ünitesi (Control Unit - CU), bilgisayar buyruklarından onları uygulayan mikro işlemler için kontrol sinyalleri üretir.
- Kontrol üniteleri iki şekilde gerçekleştirilebilir.
  - Donanımsal Kontrol
    - Bu yöntemde kontrol ünitesi birleşik lojik devreler yardımı ile donanım seviyesinde oluşturulur.
  - Yazılımsal Kontrol
    - İşlemcideki bir kontrol belleği, gerekli kontrol sinyallerini aktive eden mikro programları içerir.
- Temel bilgisayar için Kontrol Ünitesini donanımsal olarak tasarlayacağız.

# Zamanlama ve Denetim

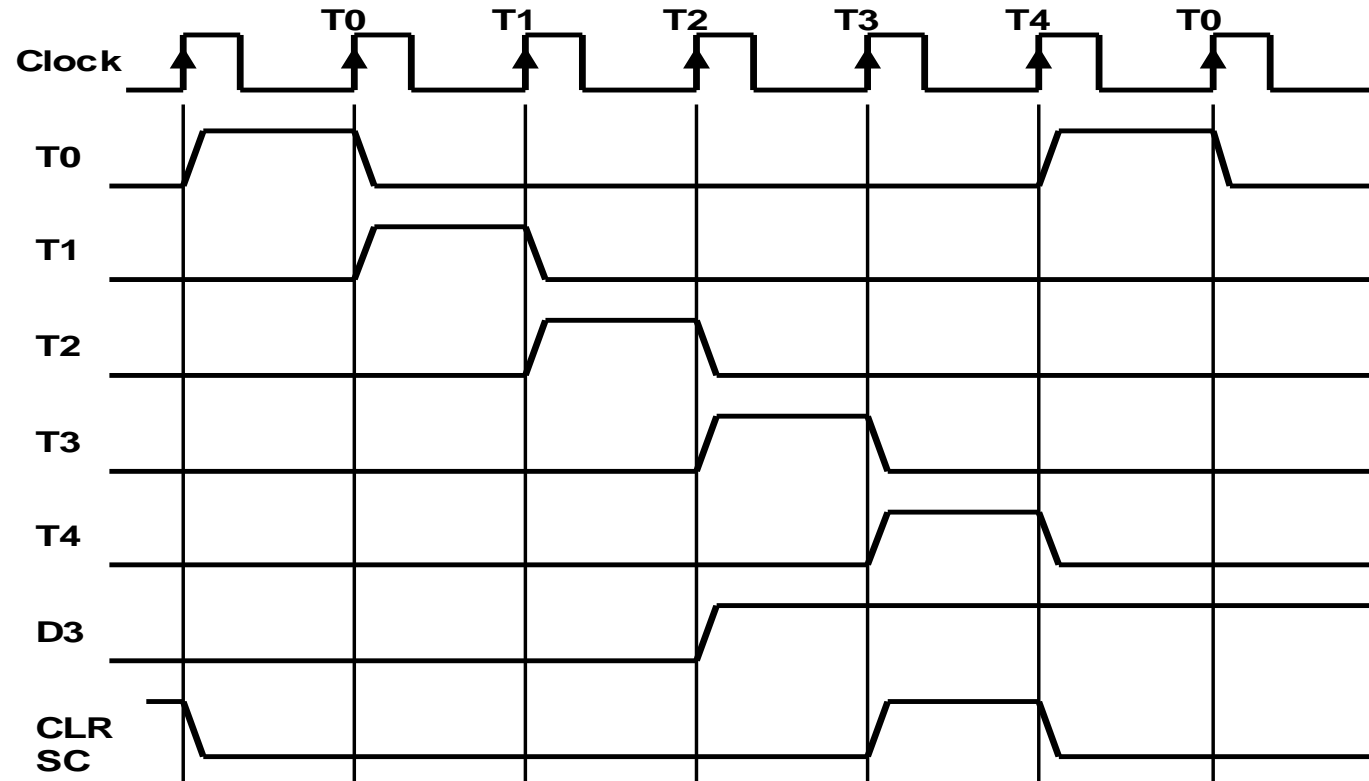
## Temel Bilgisayar Kontrol Ünitesi



# Denetim Zaman Sinyal Örneği

- $T_0, T_1, T_2, T_3, T_4, \dots, T_{15}$  Sinyalleri Sıra Sayacı (SC - Sequence Counter) tarafından üretilir.
- ÖRN:  $D_3$  sinyali aktif iken  $T_4$  anında SC içeriği sıfırlandığını düşünelim. Bu durumda:

**$D_3 T_4: SC \leftarrow 0$**

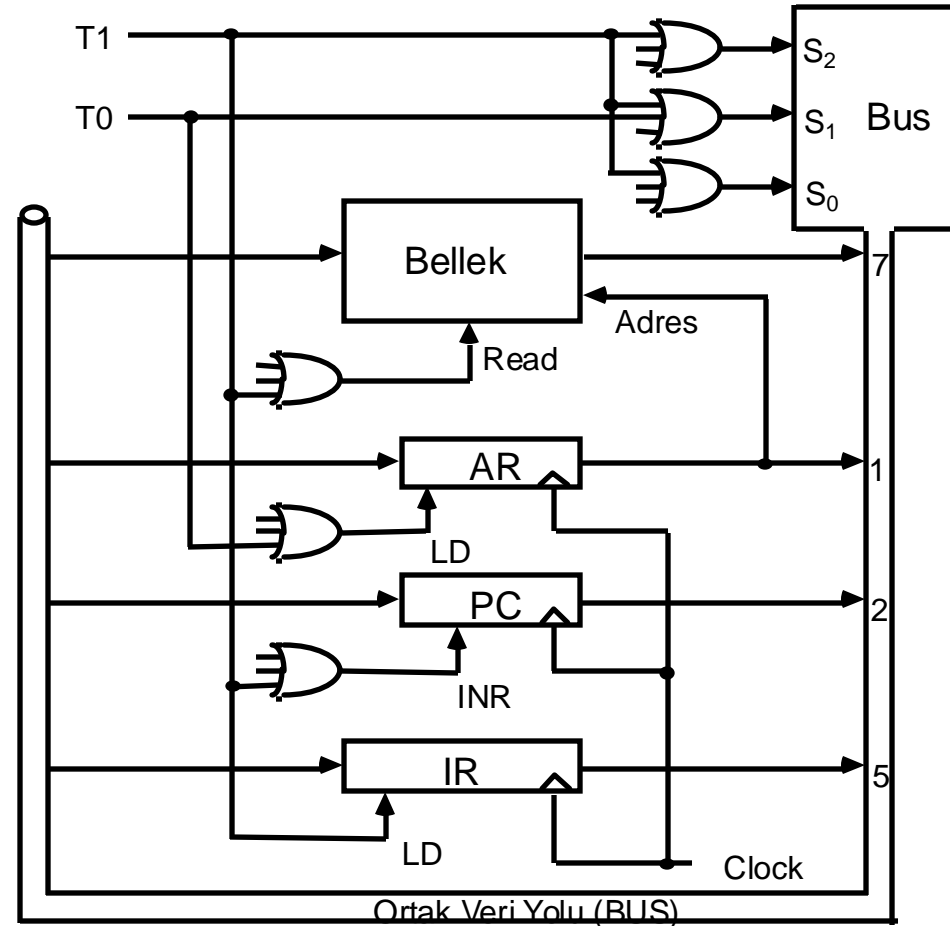


# Buyruk Süreci

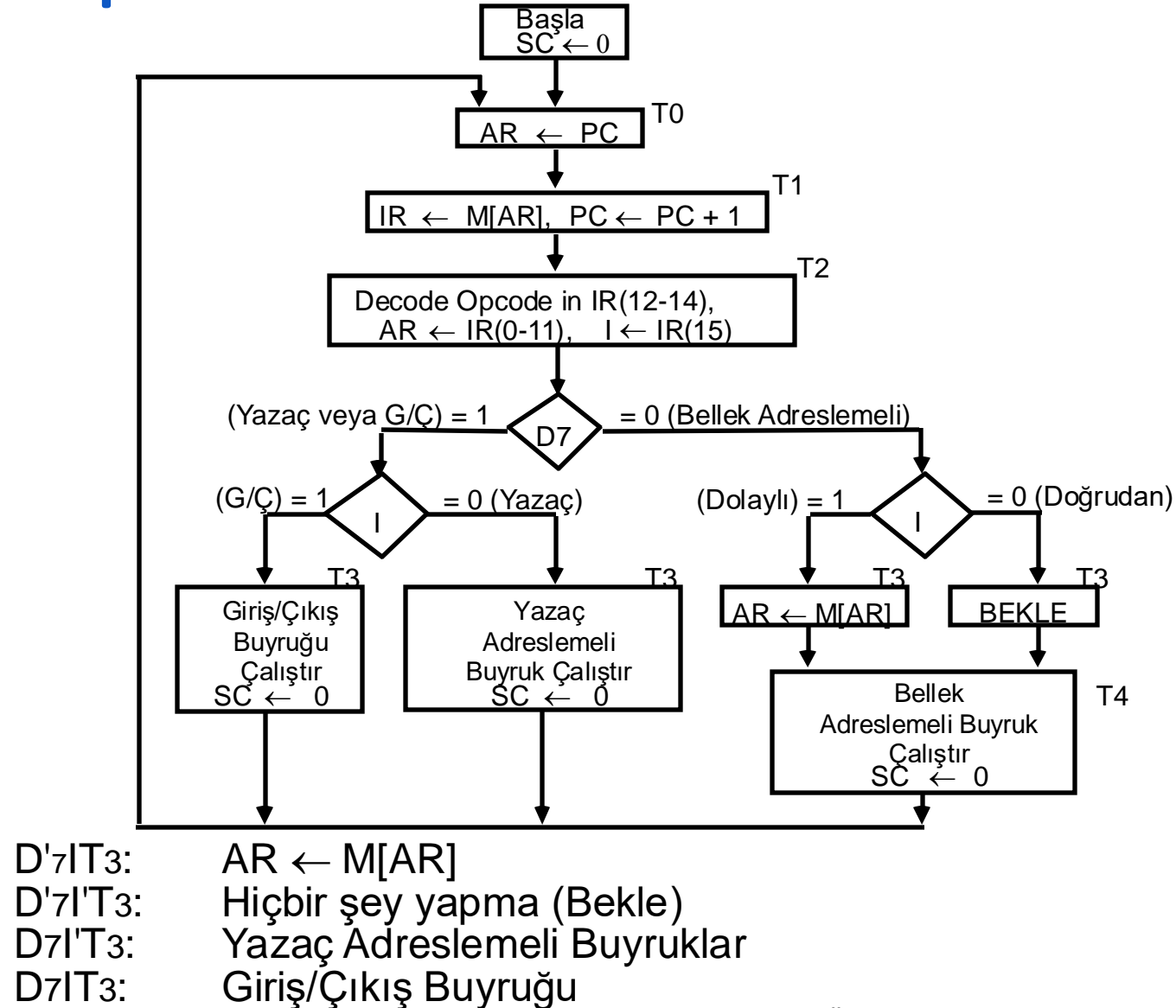
- Temel Bilgisayar'da, bir buyruk aşağıdaki şekilde yürütülür:
  - Bir buyruğun bellekten alınıp getirilmesi (FETCH)
  - Buyruk kodunun çözülmesi (DECODE)
  - Eğer buyruk dolaylı adreslemeli ise etkin adresin okunması (FETCH OPERAND)
  - Buyruğun çalıştırılması (EXECUTE)
- Bir buyruk yürütüldükten sonra, bir sonraki buyruk için döngü 1. adımdan tekrar başlar.
- Not: Her farklı işlemcinin kendi (farklı) buyruk döngüsü vardır.

# Al-getir ve Kod Çöz

T0:  $AR \leftarrow PC$  ( $S_0S_1S_2=010$ ,  $T0=1$ )  
T1:  $IR \leftarrow M[AR]$ ,  $PC \leftarrow PC + 1$  ( $S_0S_1S_2=111$ ,  $T1=1$ )  
T2:  $D0, \dots, D7 \leftarrow \text{Decode } IR(12-14)$ ,  $AR \leftarrow IR(0-11)$ ,  $I \leftarrow IR(15)$



# Buyruk Tipinin Belirlenmesi



# Yazaç Adreslemeli Buyruklar

- Yazaç adreslemeli buyruklar aşağıdaki gibi tanımlanır.
  - $D_7 = 1, I = 0$
  - Yazaç adreslemeli buyruklar IR(0 - 11) yazacının bitleri arasında tanımlanır ( $B_0 \sim B_{11}$ )
  - Buyruk  $T_3$  zamanında gerçekleşir.

$r = D_7 I' T_3 \Rightarrow$  Yazaç Adreslemeli Buyruk  
 $B_i = IR(i), i=0,1,2,...,11$

	r:	$SC \leftarrow 0$
CLA	$rB_{11}$ :	$AC \leftarrow 0$
CLE	$rB_{10}$ :	$E \leftarrow 0$
CMA	$rB_9$ :	$AC \leftarrow AC'$
CME	$rB_8$ :	$E \leftarrow E'$
CIR	$rB_7$ :	$AC \leftarrow shr AC, AC(15) \leftarrow E, E \leftarrow AC(0)$
CIL	$rB_6$ :	$AC \leftarrow shl AC, AC(0) \leftarrow E, E \leftarrow AC(15)$
INC	$rB_5$ :	$AC \leftarrow AC + 1$
SPA	$rB_4$ :	if ( $AC(15) = 0$ ) then ( $PC \leftarrow PC+1$ )
SNA	$rB_3$ :	if ( $AC(15) = 1$ ) then ( $PC \leftarrow PC+1$ )
SZA	$rB_2$ :	if ( $AC = 0$ ) then ( $PC \leftarrow PC+1$ )
SZE	$rB_1$ :	if ( $E = 0$ ) then ( $PC \leftarrow PC+1$ )
HLT	$rB_0$ :	$S \leftarrow 0$ (S bir başlangıç flip-flop 'dur.)



# Bellek Adreslemeli Buyruklar(\*1)

Sembol	Kod Çözücü	Senbolik Tanımlama
AND	D <sub>0</sub>	$AC \leftarrow AC \wedge M[AR]$
ADD	D <sub>1</sub>	$AC \leftarrow AC + M[AR], E \leftarrow C_{out}$
LDA	D <sub>2</sub>	$AC \leftarrow M[AR]$
STA	D <sub>3</sub>	$M[AR] \leftarrow AC$
BUN	D <sub>4</sub>	$PC \leftarrow AR$
BSA	D <sub>5</sub>	$M[AR] \leftarrow PC, PC \leftarrow AR + 1$
ISZ	D <sub>6</sub>	$M[AR] \leftarrow M[AR] + 1, \text{ if } M[AR] + 1 = 0 \text{ then } PC \leftarrow PC+1$

- Etkin Bellek Adresi I biti (I=0) iken T<sub>2</sub> zamanında AR yazacına aktarılır.
- Eğer I biti (I=1) ise Etkin adres T<sub>3</sub> anında AR yazacına aktarılır.
- Bellek Adreslemeli yazaç işlemleri T<sub>4</sub> anında icra edilmeye başlar.

# Bellek Adreslemeli Buyruklar(\*2)

- **AND : AC'yi VElemek**

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

Etkin Adresteki Veriyi Oku  
DR yi AC ile VE'le

- **ADD AC'ye Ekle**

- $D_1T_4$ :  $DR \leftarrow M[AR]$
- $D_1T_5$ :  $AC \leftarrow AC + DR, E \leftarrow C_{out}, SC \leftarrow 0$

Etkin Adresteki Veriyi Oku  
DR ile AC'yi topla Eldeyi E bitine yaz

- **LDA: Load to AC**

- $D_2T_4$ :  $DR \leftarrow M[AR]$
- $D_2T_5$ :  $AC \leftarrow DR, SC \leftarrow 0$

Etkin Adresteki Veriyi Oku  
DR içindeki veriyi AC'ye yükle

- **STA: Store AC**

- $D_3T_4$ :  $M[AR] \leftarrow AC, SC \leftarrow 0$

AC içeriğini Belleğe yükle

# Bellek Adreslemeli Buyruklar(\*3)

- **BUN: Şartsız Dalkan**


- $D_4T_4$ :  $PC \leftarrow AR, SC \leftarrow 0$

- **BSA: Dalkan ve Geri Dönüş Adresini Sakla**

- $D_5T_4$ :  $M[AR] \leftarrow PC, AR \leftarrow AR + 1$

- $D_5T_5$ :  $PC \leftarrow AR, SC \leftarrow 0$

Bellek, PC, AR, T4 Zamanında

20	0	BSA	135
PC = 21	Bir sonraki Buyruk		
AR = 135			
	136		
	Alt Program		
			
	1	BUN	135

Bellek

Bellek ve PC buyruktan sonra

20	0	BSA	135
21	Bir sonraki Buyruk		
135	21		
PC = 136	Alt Program		
	↓		
	1	BUN	135

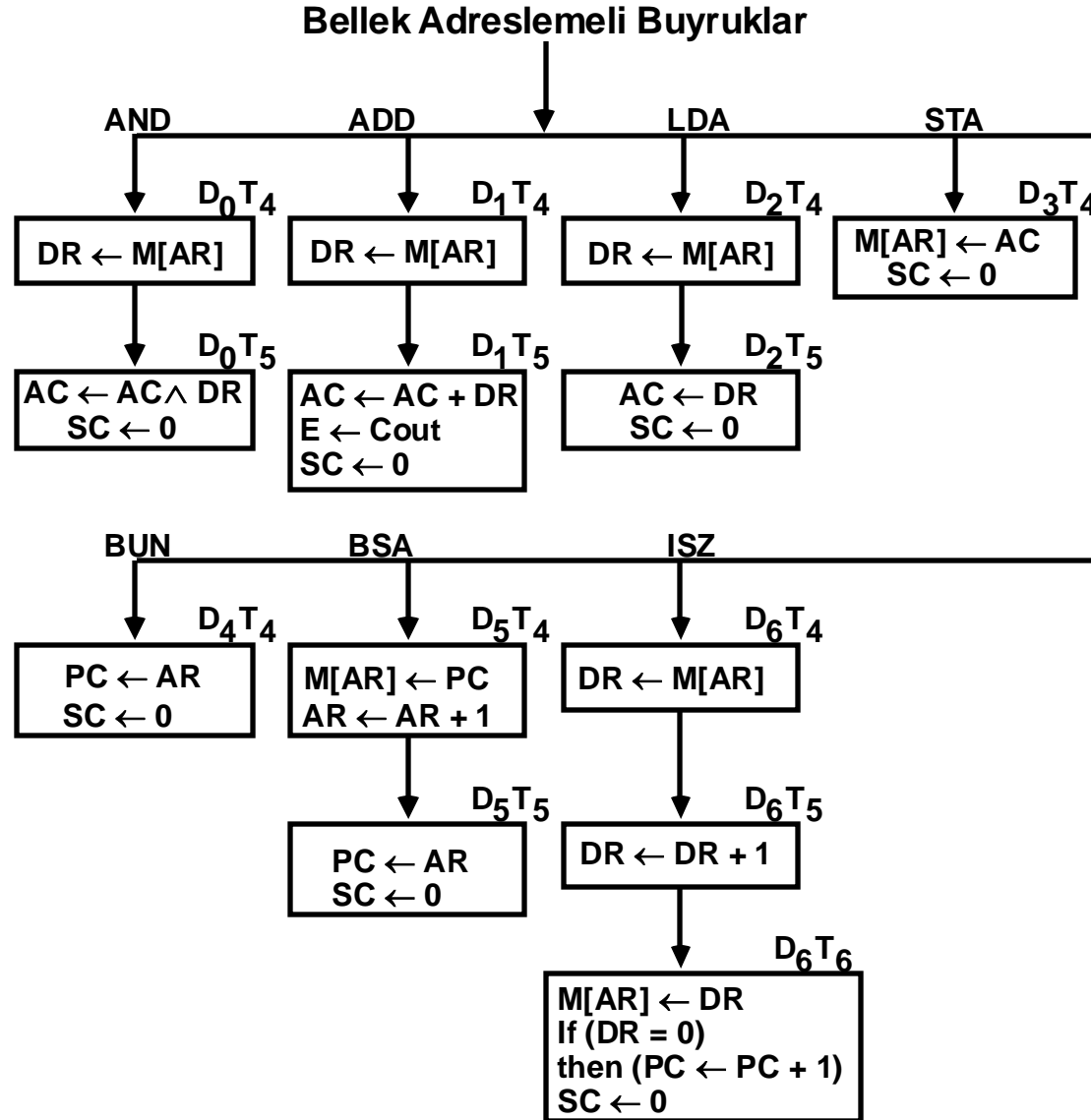
Bellek

# Bellek Adreslemeli Buyruklar(\*4)

- **ISZ: Arttır ve Eğer Sıfır İse Atla**

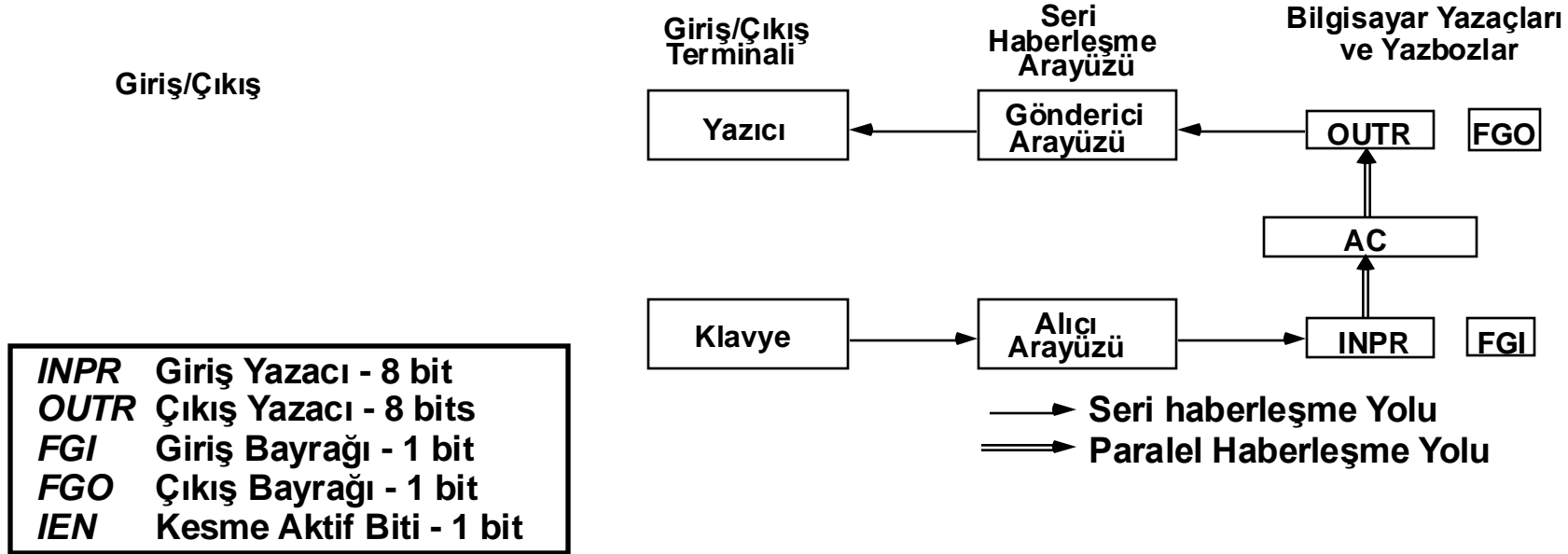
- $D_6T_4$ :  $DR \leftarrow M[AR]$
- $D_6T_5$ :  $DR \leftarrow DR + 1$
- $D_6T_6$ :  $M[AR] \leftarrow DR$ , if  $(DR = 0)$  then  $(PC \leftarrow PC + 1)$ ,  $SC \leftarrow 0$

# Bellek Adreslemeli Buyruklar için Akış Şeması



# Giriş/Çıkış Buyrukları ve Kesmeler

## Giriş/Çıkış Düzeni



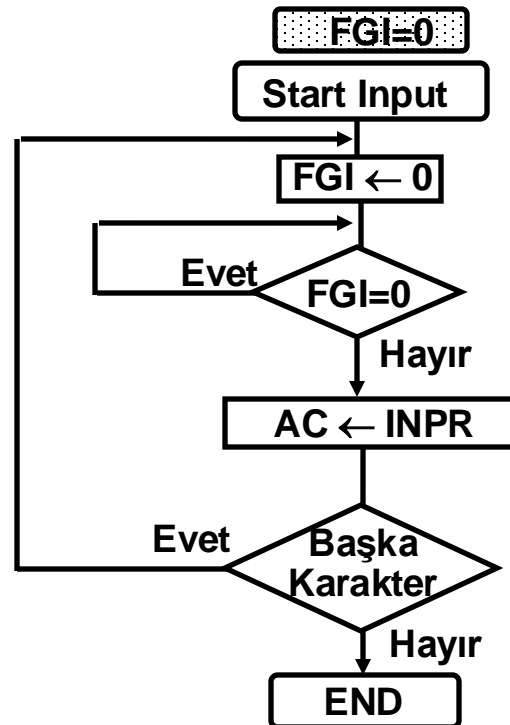
- Terminal Seri bilgi gönderir ve alır.
- Klavyeden ulaşan Seri bilgi INPR Yazacına shift ederek aktarılır.
- Yazıcıya gönderilecek seri bilgi OUTR Yazacında saklanır.
- INPR ve OUTR Terminal cihazlar ile iletişim kurmamızı sağlar.
- G / Ç aygıtı ve bilgisayar arasındaki zamanlama farkını **senkronize** etmek için bayraklar gerekir.

# Program Kontrollü Veri Aktarımı

-- İşlemci --

```
/* Giriş */      /* Initially FGI = 0 */  
loop: If FGI = 0 goto loop  
      AC ← INPR, FGI ← 0
```

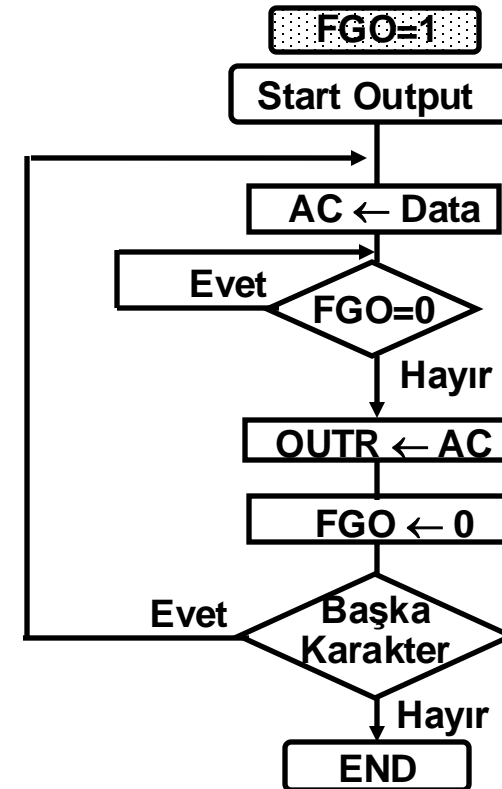
```
/* Çıkış */      /* Initially FGO = 1 */  
loop: If FGO = 0 goto loop  
      OUTR ← AC, FGO ← 0
```



-- G/Ç Aygıtı --

```
loop: If FGI = 1 goto loop  
      INPR ← new data, FGI ← 1
```

```
loop: If FGO = 1 goto loop  
      consume OUTR, FGO ← 1
```



# Giriş/Çıkış Buyrukları

$D_7IT_3 = p$  (tüm giriş çıkış buyruklarında ortak)  
 $IR(i) = B_i, i = 6, \dots, 11$  (Buyruk kontrol sinyali)

INP	p: $SC \leftarrow 0$	Sil SC
OUT	$pB_{11}: AC(0-7) \leftarrow INPR, FGI \leftarrow 0$	Giriş Karakteri oku (AC)
SKI	$pB_{10}: OUTR \leftarrow AC(0-7), FGO \leftarrow 0$	AC'den çıkış Karakteri al
SKO	$pB_9: \text{if}(FGI = 1) \text{ then } (PC \leftarrow PC + 1)$	Giriş Bayrağını Atla
ION	$pB_8: \text{if}(FGO = 1) \text{ then } (PC \leftarrow PC + 1)$	Çıkış Bayrağını Atla
IOF	$pB_7: IEN \leftarrow 1$	Keme (Interrupt) Aktif
	$pB_6: IEN \leftarrow 0$	Keme (Interrupt) Pasif



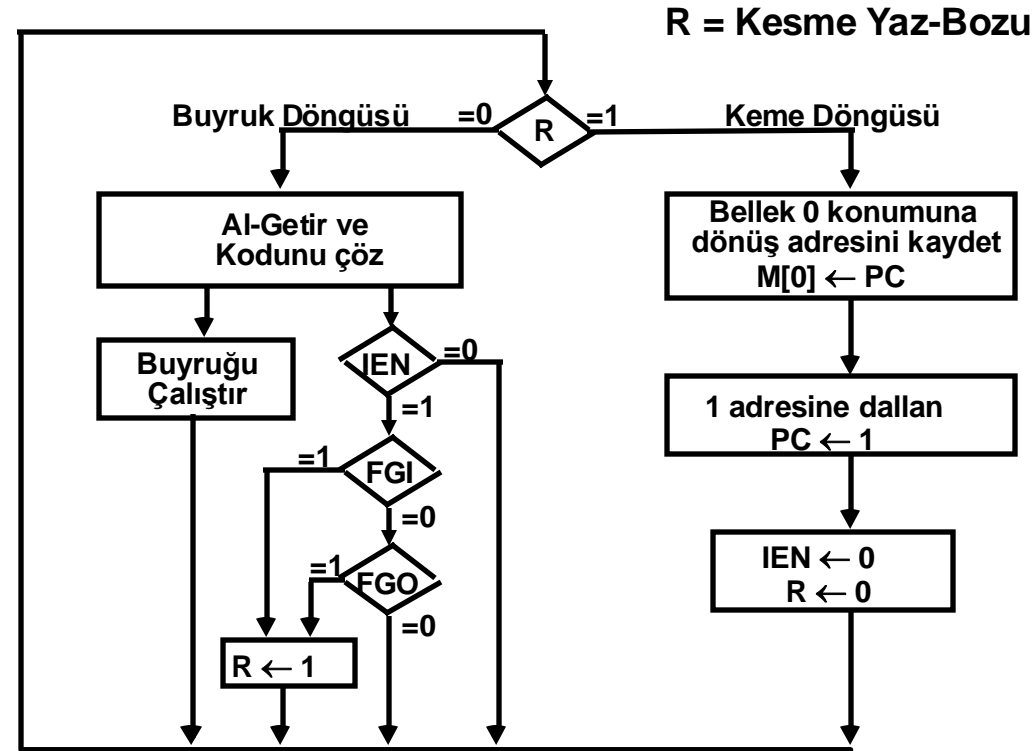
# Program Kesme

- Temel Bilgisayarda kesme denetimi için G/Ç aygıtının kesme işlemine hazır olduğunu bilgisayara haber vermesi gerekir.
- Bu işlem program kontrollü kesme işlemi olarak adlandırılır.
- Bu arada bilgisayar başka işler ile uğraşabilir.
- G/Ç aygıtı bir kesme isteği yapmak için kesme bayrağını (R) BİR (R=1) yapar.
- Bayrak (R=1) yapıldığında bilgisayar yapmakta olduğu işi anında bırakır ve G/Ç işlemlerine başlar.
- Eğer Kesme izin yaz-bozu (IEN=0) ise kullanıcı kesme yapmak istemiyordur.
- Eğer (IEN=1) iken Giriş (FGI) yada Çıkış(FGO) bayraklarından biri 1 ise R yazbozu (R=1) yapılarak Kesme Çevrimi başlatılır.
- R=1 ise buyruk sürecine geçilmez.

# Kesme Süreci

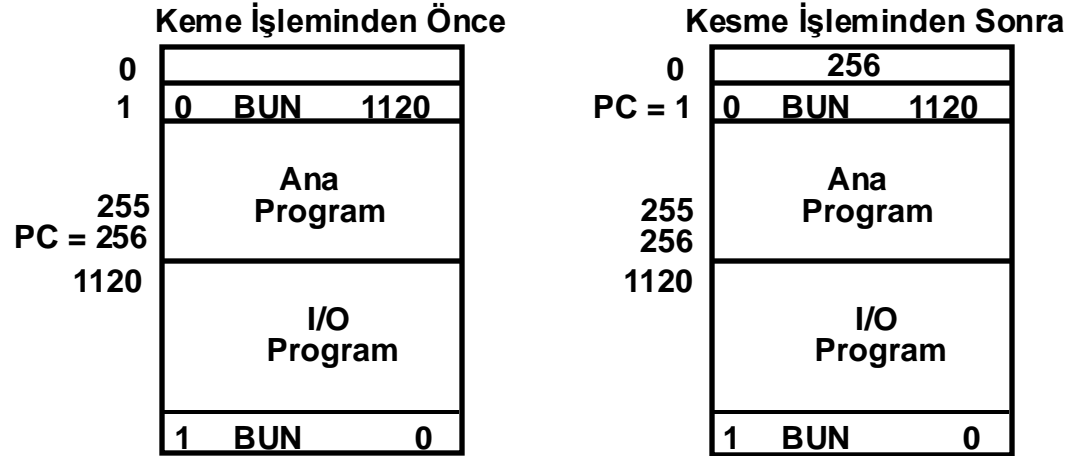
- Kesme süreci dallanma ve geri dönüş işleminin donanımla yapılmış bir biçimidir.
- PC içindeki dönüş adresi, daha sonra kolayca bulunabilecek bir adrese kaydedilir.
- Kesme işlemi için gerekli olan servis programı çalıştırılır.
- Kesme işlemi sona erdiğinde geri dönüş adresi tekrar PC geri yüklenerek program kaldığı yerden çalışmaya devam eder.
- Temel Bilgisayarda Belleğin Sıfırinci ( $AR=0$ ) adresi dönüş adresini yazmak için kullanılır.
- Bundan sonra denetim  $PC=1$  adresini yazar ve  $IEN=0$  yaparak Kesme alt programı çalıştırılmış olur.

# Kesme Süreci Akış Şeması



# Kesme Süreci Gösterimi

## Bellek



Kesme Servisinin çalıştırılması için R yazbozunun R=1 yapılma şartı :

$$\begin{aligned} & - R \text{ F/F} \leftarrow 1 \quad \text{if } IEN (FGI + FGO) T_0' T_1' T_2' \\ & \Leftrightarrow T_0' T_1' T_2' (IEN)(FGI + FGO): R \leftarrow 1 \end{aligned}$$

- Kesme Çevrimi Başladığında  $T_0, T_1, T_2$  adımları  $R'T_0, R'T_1, R'T_2$  adımları ile yer değiştir.
- Değiştirilmiş Al-Getir (Kesme) Döngüsü :

$RT_0: AR \leftarrow 0, TR \leftarrow PC$

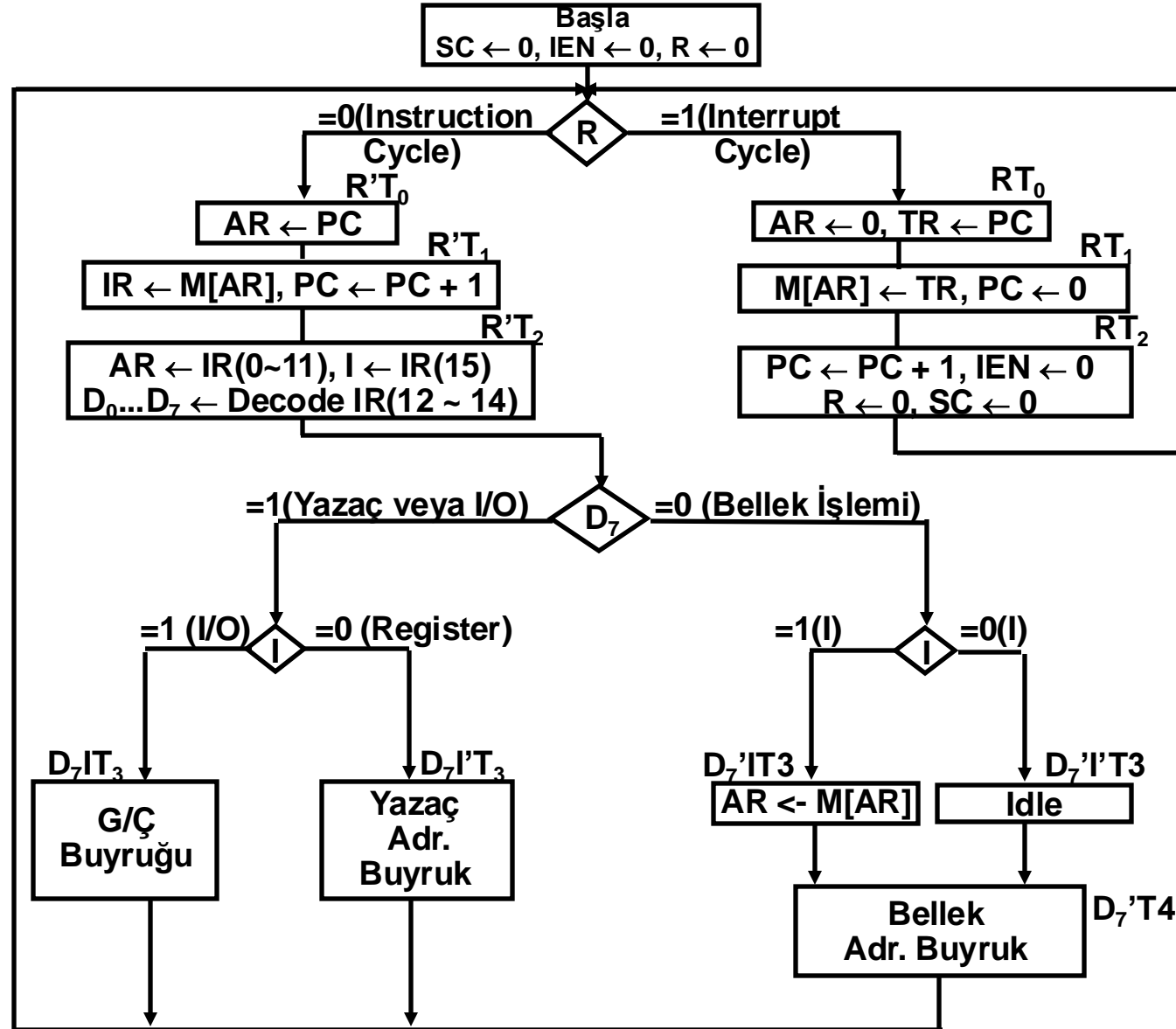
$RT_1: M[AR] \leftarrow TR, PC \leftarrow 0$

$RT_2: PC \leftarrow PC + 1, IEN \leftarrow 0, R \leftarrow 0, SC \leftarrow 0$

# Temel Bilgisayarın Tamamlanmış Tanımı

- Aşağıdaki bileşenlerden oluşur:
  1. 16 bit 4096 kelimelik bellek
  2. 9 Yazaç: AR, PC, DR, AC, IR, TR, OUTR, INPR VE SC
  3. 7 yaz-boz I,S,E,R IEN, FGI ve FGO

# Bilgisayar İşlemleri için Akış Şeması



# Tablo 5.6

Fetch	R'T <sub>0</sub> :	AR ← PC
	R'T <sub>1</sub> :	IR ← M[AR], PC ← PC + 1
Decode	R'T <sub>2</sub> :	D0, ..., D7 ← Decode IR(12 ~ 14), AR ← IR(0 ~ 11), I ← IR(15)
Indirect Interrupt	D <sub>7</sub> 'IT <sub>3</sub> :	AR ← M[AR]
	T <sub>0</sub> 'T <sub>1</sub> 'T <sub>2</sub> '(IEN)(FGI + FGO):	R ← 1
	RT <sub>0</sub> :	AR ← 0, TR ← PC
	RT <sub>1</sub> :	M[AR] ← TR, PC ← 0
	RT <sub>2</sub> :	PC ← PC + 1, IEN ← 0, R ← 0, SC ← 0
Memory-Reference		
AND	D <sub>0</sub> T <sub>4</sub> :	DR ← M[AR]
	D <sub>0</sub> T <sub>5</sub> :	AC ← AC ∧ DR, SC ← 0
ADD	D <sub>1</sub> T <sub>4</sub> :	DR ← M[AR]
	D <sub>1</sub> T <sub>5</sub> :	AC ← AC + DR, E ← C <sub>out</sub> , SC ← 0
LDA	D <sub>2</sub> T <sub>4</sub> :	DR ← M[AR]
	D <sub>2</sub> T <sub>5</sub> :	AC ← DR, SC ← 0
STA	D <sub>3</sub> T <sub>4</sub> :	M[AR] ← AC, SC ← 0
BUN	D <sub>4</sub> T <sub>4</sub> :	PC ← AR, SC ← 0
BSA	D <sub>5</sub> T <sub>4</sub> :	M[AR] ← PC, AR ← AR + 1
	D <sub>5</sub> T <sub>5</sub> :	PC ← AR, SC ← 0
ISZ	D <sub>6</sub> T <sub>4</sub> :	DR ← M[AR]
	D <sub>6</sub> T <sub>5</sub> :	DR ← DR + 1
	D <sub>6</sub> T <sub>6</sub> :	M[AR] ← DR, if(DR=0) then (PC ← PC + 1), SC ← 0

# Tablo 5.6

Register-Reference		
	$D_7I'T_3 = r$ $IR(i) = B_i$ $r:$	(Common to all register-reference instr) ( $i = 0,1,2, \dots, 11$ ) $SC \leftarrow 0$
CLA	$rB_{11}:$	$AC \leftarrow 0$
CLE	$rB_{10}:$	$E \leftarrow 0$
CMA	$rB_9:$	$AC \leftarrow AC'$
CME	$rB_8:$	$E \leftarrow E'$
CIR	$rB_7:$	$AC \leftarrow shr\ AC, AC(15) \leftarrow E, E \leftarrow AC(0)$
CIL	$rB_6:$	$AC \leftarrow shl\ AC, AC(0) \leftarrow E, E \leftarrow AC(15)$
INC	$rB_5:$	$AC \leftarrow AC + 1$
SPA	$rB_4:$	If( $AC(15) = 0$ ) then ( $PC \leftarrow PC + 1$ )
SNA	$rB_3:$	If( $AC(15) = 1$ ) then ( $PC \leftarrow PC + 1$ )
SZA	$rB_2:$	If( $AC = 0$ ) then ( $PC \leftarrow PC + 1$ )
SZE	$rB_1:$	If( $E=0$ ) then ( $PC \leftarrow PC + 1$ )
HLT	$rB_0:$	$S \leftarrow 0$
Input-Output		
	$D_7IT_3 = p$ $IR(i) = B_i$ $p:$	(Common to all input-output instructions) ( $i = 6,7,8,9,10,11$ ) $SC \leftarrow 0$
INP	$pB_{11}:$	$AC(0-7) \leftarrow INPR, FGI \leftarrow 0$
OUT	$pB_{10}:$	$OUTR \leftarrow AC(0-7), FGO \leftarrow 0$
SKI	$pB_9:$	If( $FGI=1$ ) then ( $PC \leftarrow PC + 1$ )
SKO	$pB_8:$	If( $FGO=1$ ) then ( $PC \leftarrow PC + 1$ )
ION	$pB_7:$	$IEN \leftarrow 1$
IOF	$pB_6:$	$IEN \leftarrow 0$



# Yazacın ve Belleğin Denetimi

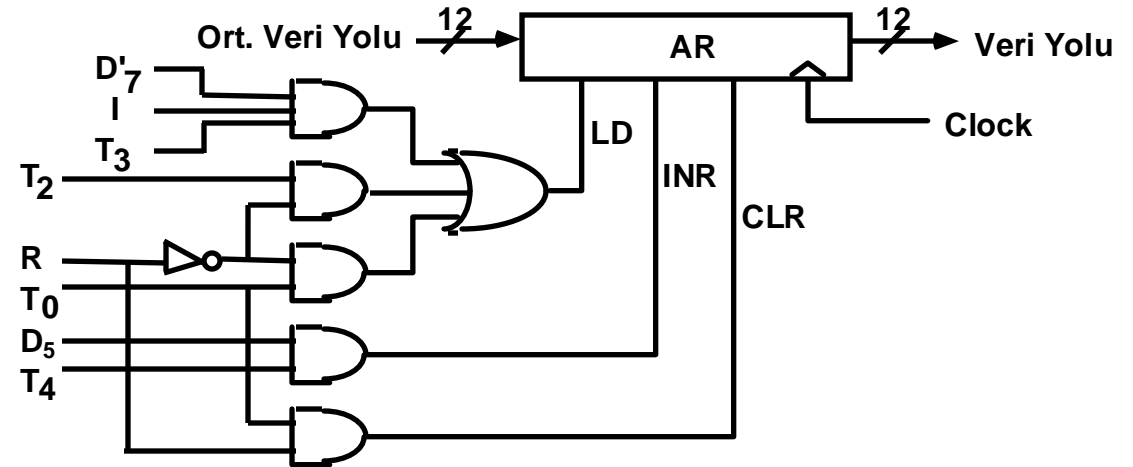
## Address Register; AR

Tablo 5.6 içerisinde AR ile ilgili bütün mikro işlemler taranır:

$R'T_0$ :	$AR \leftarrow PC$	$LD(AR)$
$R'T_2$ :	$AR \leftarrow IR(0-11)$	$LD(AR)$
$D'_7IT_3$ :	$AR \leftarrow M[AR]$	$LD(AR)$
$RT_0$ :	$AR \leftarrow 0$	$CLR(AR)$
$D_5T_4$ :	$AR \leftarrow AR + 1$	$INR(AR)$



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



# Tek Yaz-Boz (Bayrak) Denetimi

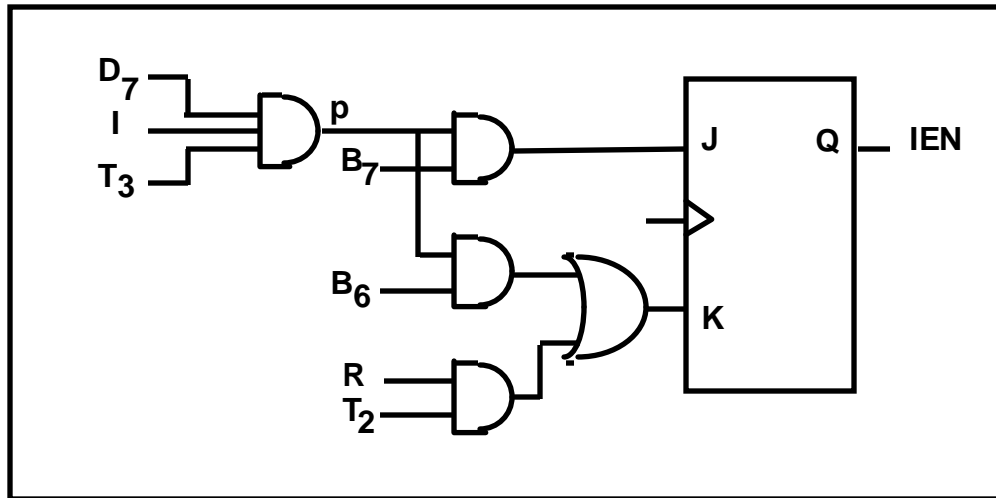
**IEN: Interrupt Enable Flag**

**pB<sub>7</sub>: IEN  $\leftarrow$  1 (I/O Instruction)**

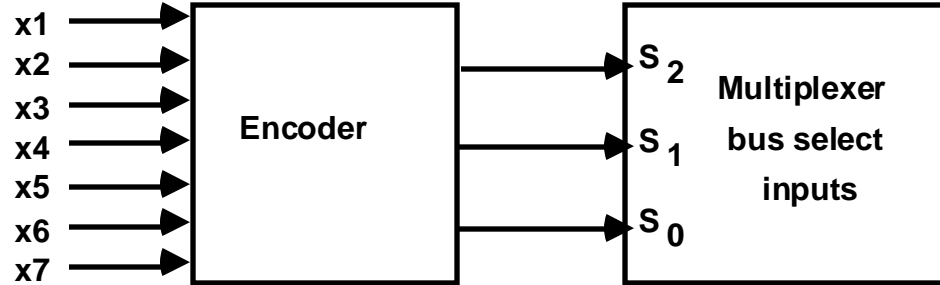
**pB<sub>6</sub>: IEN  $\leftarrow$  0 (I/O Instruction)**

**RT<sub>2</sub>: IEN  $\leftarrow$  0 (Interrupt)**

**p = D<sub>7</sub>IT<sub>3</sub> (Input/Output Instruction)**

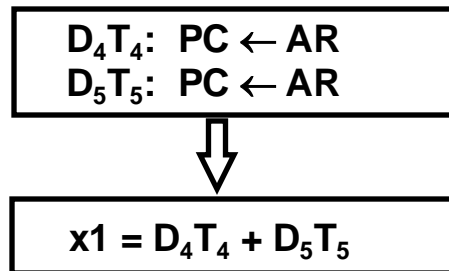


# Ortak Veri Yolu Denetimi



x1	x2	x3	x4	x5	x6	x7	S2	S1	S0	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	Bellek

AR Yazacı için:



$$S_0 = x1 + x3 + x5 + x7$$

$$S_1 = x2 + x3 + x6 + x7$$

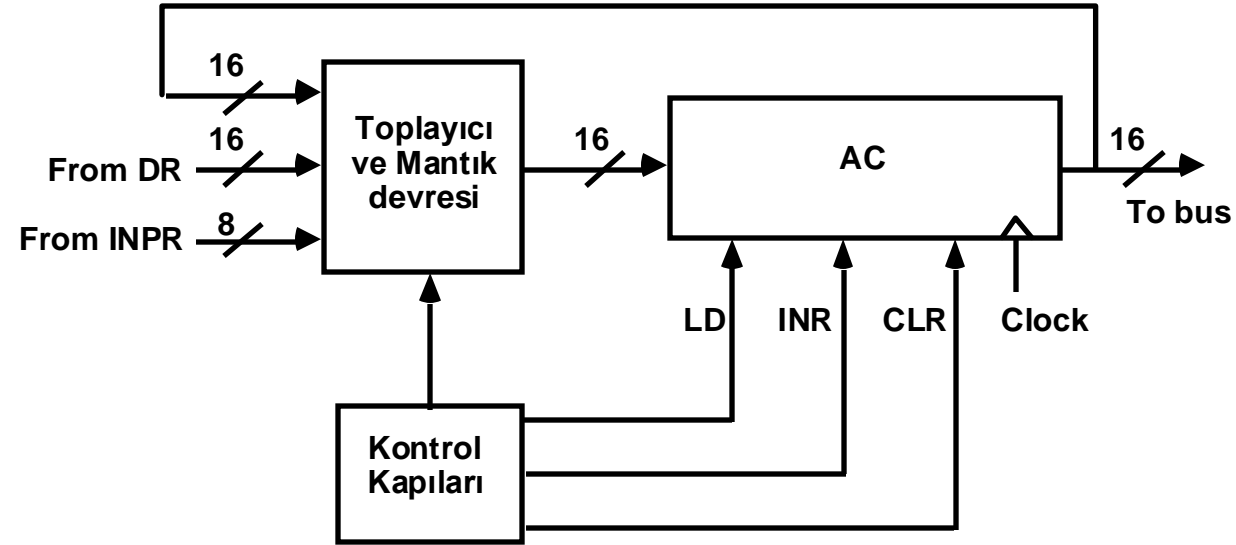
$$S_2 = x4 + x5 + x6 + x7$$

Bellek için:

$$x7 = R'T_1 + D_7'IT_3 + (D_0 + D_1 + D_2 + D_6)T_4$$

# İşlemci Yazacı Mantık Tasarımı

## AC ile ilişkili devreler

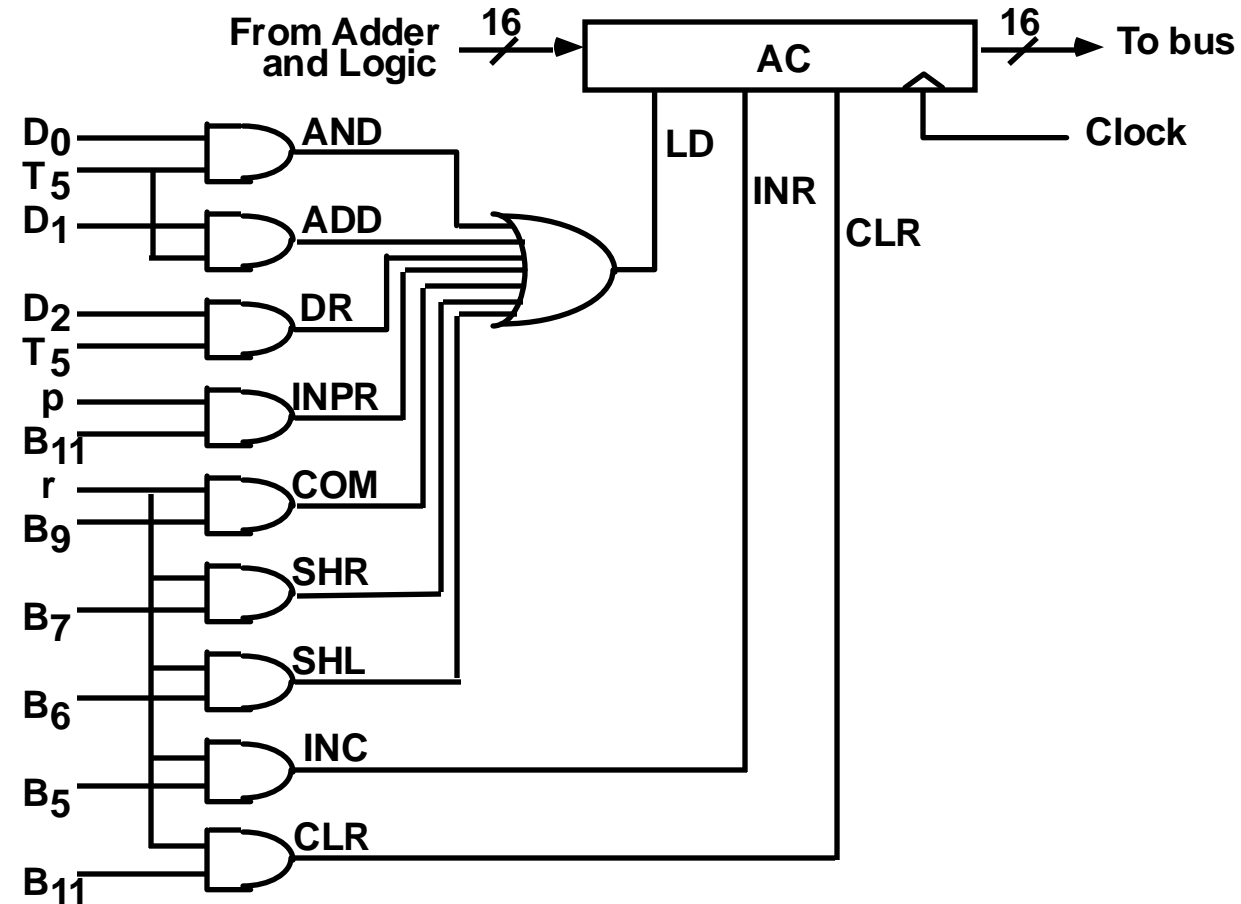


## AC'nin içeriğini değiştiren deyimler

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

# AC'nin Denetimi

## AC'nin LD, INR ve CLR denetimi için kapı tasarımı



# Sorularınız?