

# BİLGİSAYAR ORGANİZASYONU ve TASARIMI

YRD. DOÇ. DR. FATİH KELEŞ

# **Temel Bilgisayar Yapısı ve Devreleri**

## **Temel Bilgisayar Organizasyonu ve Tasarımı**

- ▶ Komut Kodları
- ▶ Bilgisayar Saklayıcıları
- ▶ Bilgisayar Komutları
- ▶ Zamanlama ve Kontrol
- ▶ Komut Döngüsü
- ▶ Komut Türünün Belirlenmesi

# Temel Bilgisayar Yapısı ve Devreleri

- ▶ Bu kısımda temel bilgisayar tanıtılacak ve işleyişinin saklayıcı aktarım deyimleri (RTL) ile nasıl belirlenebileceği gösterilecektir. Ayrıca saklanmış program yapısı (stored program) gösterilecektir.
- ▶ Bilgisayarın iç saklayıcıları, zamanlama ve kontrol yapısı, kullandığı komut kümesi bilgisayarın organizasyonunu belirlemektedir.

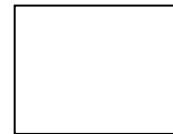
# Temel Bilgisayar Yapısı ve Devreleri

- ▶ Her farklı işlemci türü kendine özgü bir tasarıma sahiptir.  
(farklı saklayıcılar, ortak yol yapısı, mikroişlemler, makine komutları..vs..)
- ▶ Modern işlemcilerin oldukça kompleks bir yapısı bulunmaktadır.
- ▶ Bu işlemciler
  - Çok sayıda saklayıcı
  - Hem tamsayı hem kayan noktalı hesaplamalar yapabilen çoklu aritmetik işlem üniteleri
  - Program icra süresini azaltmak için ardarda gelen birkaç komutu pipeline yeteneği
  - .....
- ▶ Burada işlemcilerin nasıl çalıştığını anlamak için basitleştirilmiş bir model üzerinde çalışacağız.
- ▶ Buna Temel Bilgisayar Organizasyonu (TBO) adı vereceğiz.
- ▶ Bu işlemci ilk üretilen gerçek işlemcilere benzemektedir.
- ▶ TBO, bir işlemci organizasyonunu ve daha yüksek seviyeli bilgisayar işlemcisi için RTL modelini anlamada yararlı olacaktır.

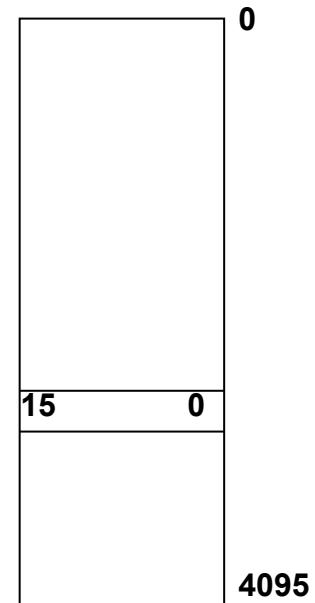
# Temel Bilgisayar

- ▶ Temel Bilgisayarın 2 ana bileşeni vardır:
  1. Bir ana işlemcisi CPU
  2. Bir ana RAM bellek
- ▶ Bellek 4096 kelime saklama kapasiteli
  - $4096 = 2^{12}$ , bu yüzden bellekteki bir kelimeyi seçmek için 12 bit gereklidir (adres uzunluğu)
  - Her kelime 16 bit uzunluğundadır

CPU



RAM



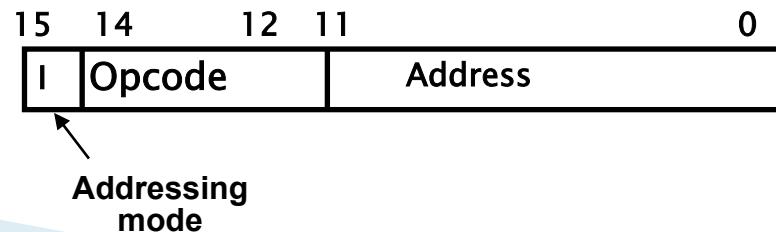
# Komutlar

- ▶ Program  
Makine komutları dizisi
- ▶ Makine Komutları
  - Belirli bir işlemi bilgisayara anlatmak üzere kullanılan bir grup bit (mikroişlem dizini)
- ▶ Program komutları ve gerekli datalar (operand) bellekte saklanır.
- ▶ CPU bellekten bir sonraki komutu okur.
- ▶ Komutu komut saklayıcısına (IR) yerleştirir.
- ▶ Kontrol ünitesinde bulunan kontrol devresi bu komutları yürütmek için yorumlayarak gereken mikroişlem dizinlerine dönüştürür.

# Komut Formatı

- ▶ Bir bilgisayar komutu iki parçaya bölünür:
  1. İşlem Alanı (Operation Code)  
komutun icra edeceği işlemleri belirler
  2. Adres alanı  
saklayıcıları ve/veya işlemi icra etmekte kullanmak için bellek gözleri
- ▶ TBO da, 4096 ( $= 2^{12}$ ) bellek kelimesi 12 bit ile adreslenir
- ▶ Komutun en anlamlı biti (15. bit) *adresleme modunu belirler.*  
(0: doğrudan adresleme, 1: dolaylı adresleme)
- ▶ Bellek kelimeleri ve komutlar 16 bit uzunluklu olduğundan komutun işlem alanına 3 bit ayrılmıştır.

Komut Formatı



# Saklanmış Program

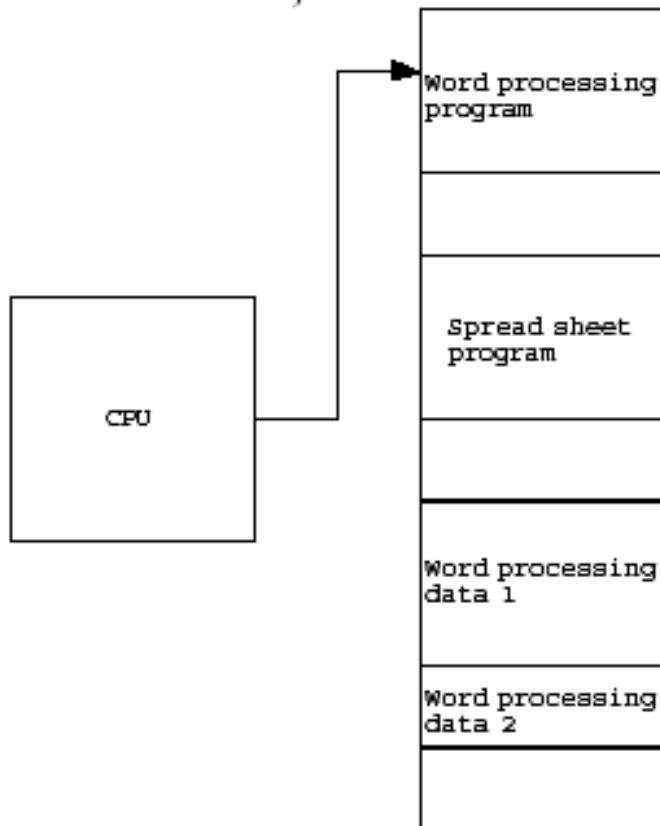


Figure: Stored program computer concept.

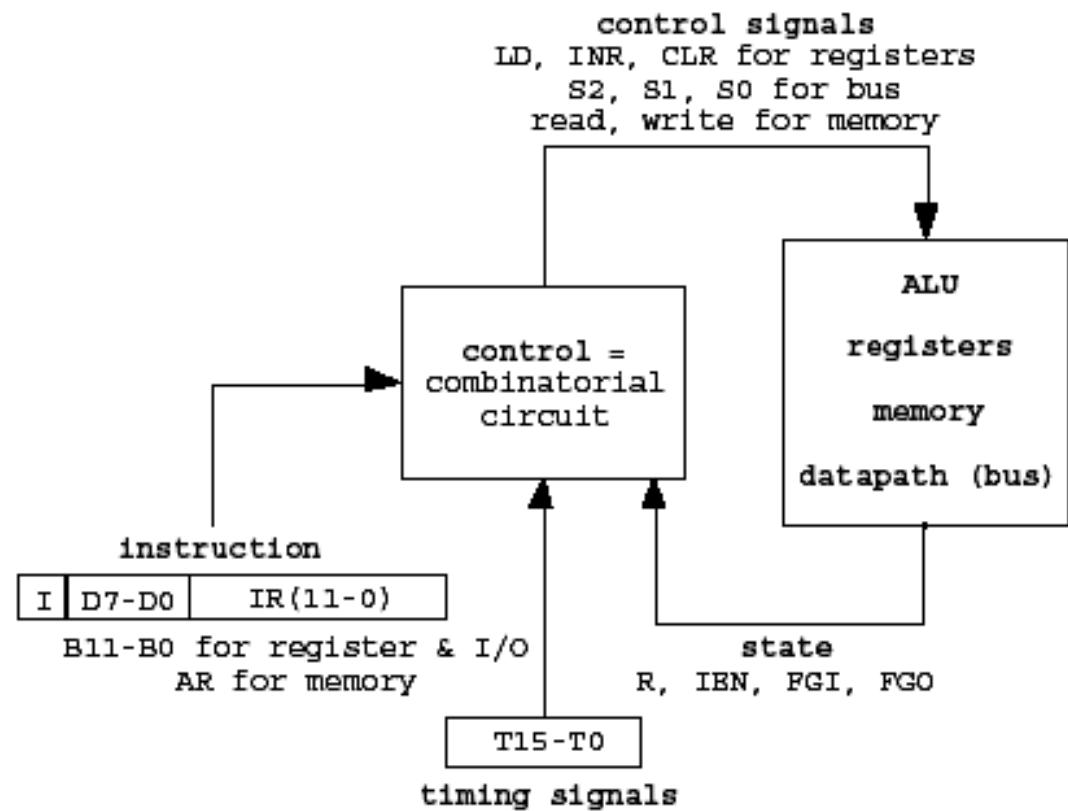


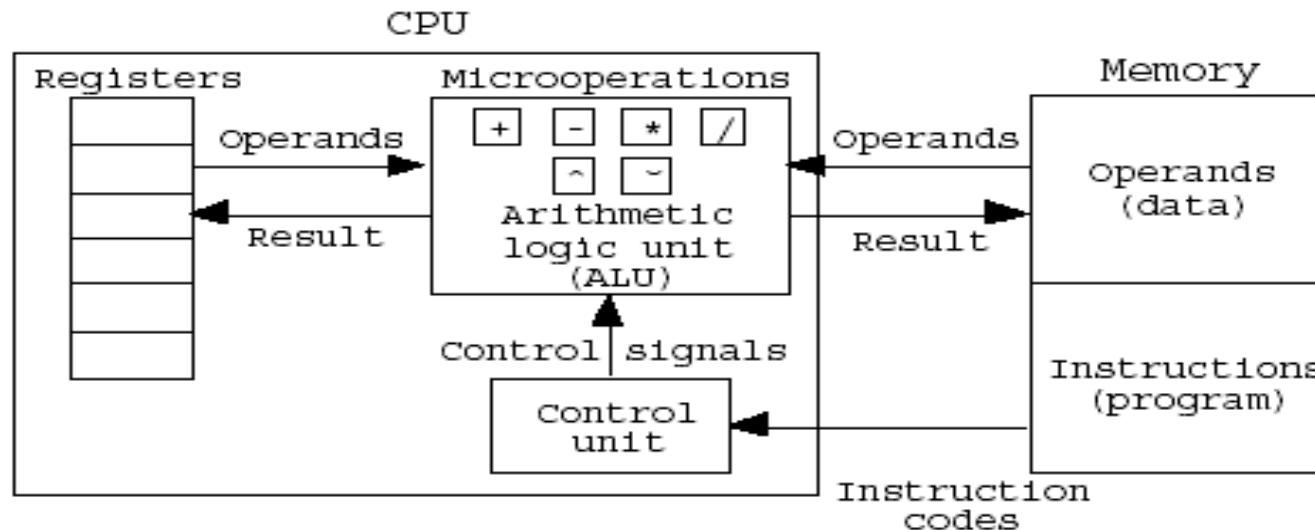
Figure: Stored program computer hardware.

# Saklanmış Program

- ▶ Bilgisayar için mikroişlemlerin sırasını belirleyen ikili koda bir bilgisayar komutu denir.
- ▶ Bilgisayar her komutu bellekten okur ve komutu ilgili saklayıcıya yerleştirir.
- ▶ Daha sonra kontrol, bu komutun ikili kodunu yorumlar ve mikroişlemleri yürüterek komutu çalıştırır.
- ▶ Her bilgisayarın kendine özgü bir komut kümesi bulunur.
- ▶ Her komutun saklanması ve sonra sırasıyla çalıştırılması (saklanmış program kavramı) genel amaçlı bilgisayarın en büyük özelliğiidir.

# Temel Kavramlar

- ▶ Bilgisayar donanımı = saklayıcılar + ALU + veriyolu + kontrol ünitesi.
- ▶ Bilgisayar komut döngüsüne (instruction cycle) girer:
  - Bellekten komutu al,
  - Kontrol sinyalleri sırasınca komutun kodunu çöz,
  - Kodu çözüleni mikroişlemler sırası olarak çalıştır.
- ▶ **Kontrol Ünitesi:** Mikroişlemleri tetiklemek için kullanılır.
- ▶ Girdi-çıktı işlemleri ise kesme döngüsünde (**interrupt cycle**) gerçekleştirilir.



# Komut Kodları

## ► Saklanmış Program Yapısı

- Bilgisayar organizasyonunun en kolay yolu.
  - Bir tane işlemci saklayıcısı (Accumulator=AC)
    - Bir işlem, bellekteki veri ile AC saklayıcısındaki veri üzerinde gerçekleştirilir.
    - İki bölümden oluşan komut kodu yapısı: Op.code + Adres
      - Op.code 16 olası farklı işlemi gösterir (4 bit).
      - Adres ise işlenenin bellek adresini gösterir (12 bit).
      - Eğer bir komut bellekten işlenen kullanmıyorsa, adres alanı farklı amaçlar için kullanılabilir.

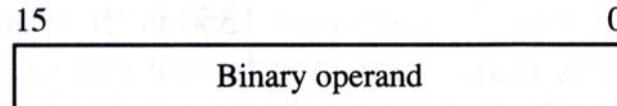
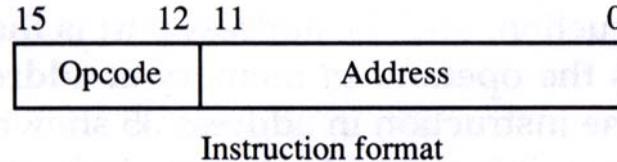
Clear AC,  
Complement  
AC

Bellek= veri + program için 12 bit = 4096 kelime

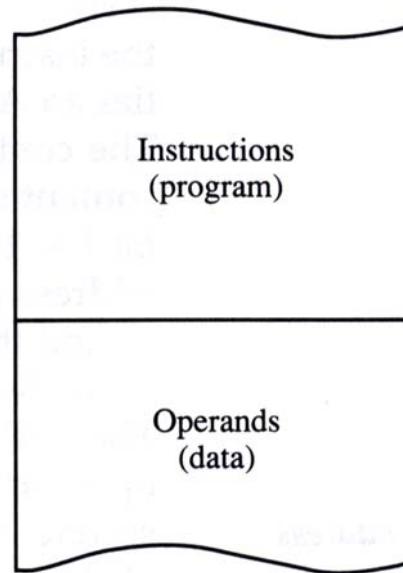
- Her komut kodu(program) ve işlenen(veri) 16 bitlik bir bellek kelimesinde saklanır.

# Komut Kodları

- ▶ Kontrol ünitesi 16-bit komutu belleğin program kısmından okur,
- ▶ Bu komut ile bellekte bulunan verinin adresinden 16-bit veri okunur,
- ▶ İşlem kodu ile işlem saklayıcısında belirtilen işlem yapılır.



Memory  
4096 × 16



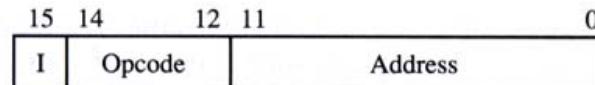
Processor register  
(accumulator or AC)

# Komut Kodları

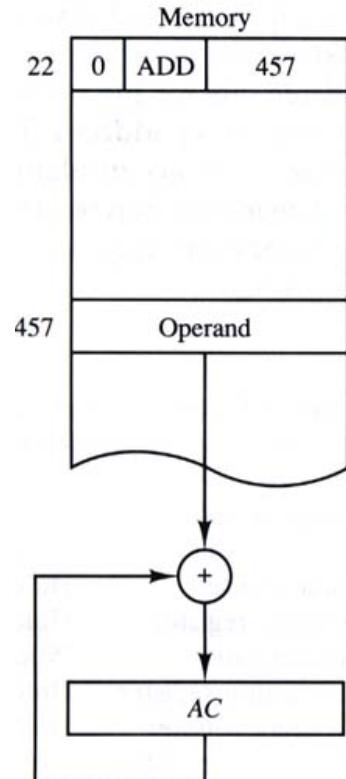
I=0 Doğrudan,  
I=1 Dolaylı

- ▶ Adresleme Modu:
  - Derhal Adresleme: Adres alanında veri bulunur.
  - Doğrudan Adresleme: İşlenenin bellek adresi, komutun adres alanında gösterilir.
  - Dolaylı Adresleme: İşlenenin adresinin işaretçisini tutar. (Komutun adres alanından, işlenenin adres bilgisi alınır.)
- ▶ **Efektif adres:** Fiziksel olarak işlenenin bellekteki bulunduğu adresdir.

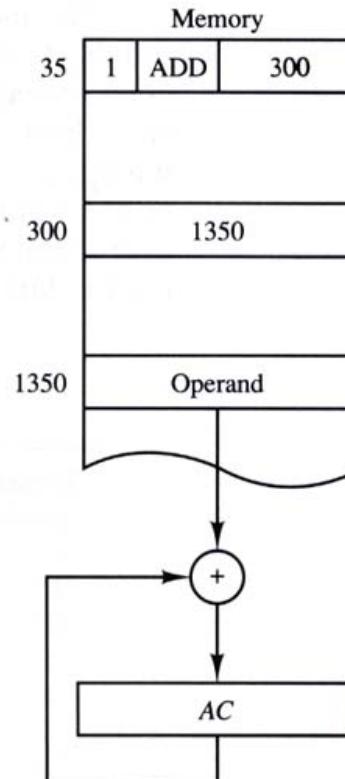
# Doğrudan ve Dolaylı Adresleme



(a) Instruction format



(b) Direct address

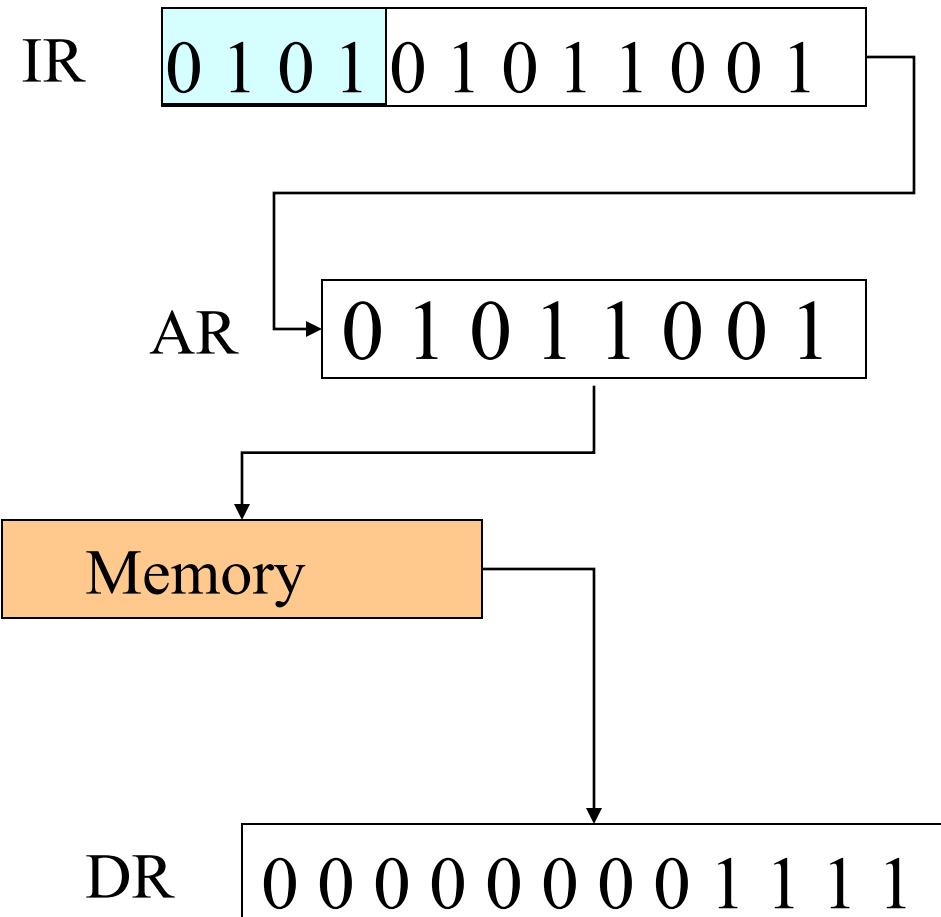


(c) Indirect address

# Doğrudan Adresleme

İhtiyaç duyulan verinin adresi, komutun adres alanında bulunduğu zaman kullanılır.

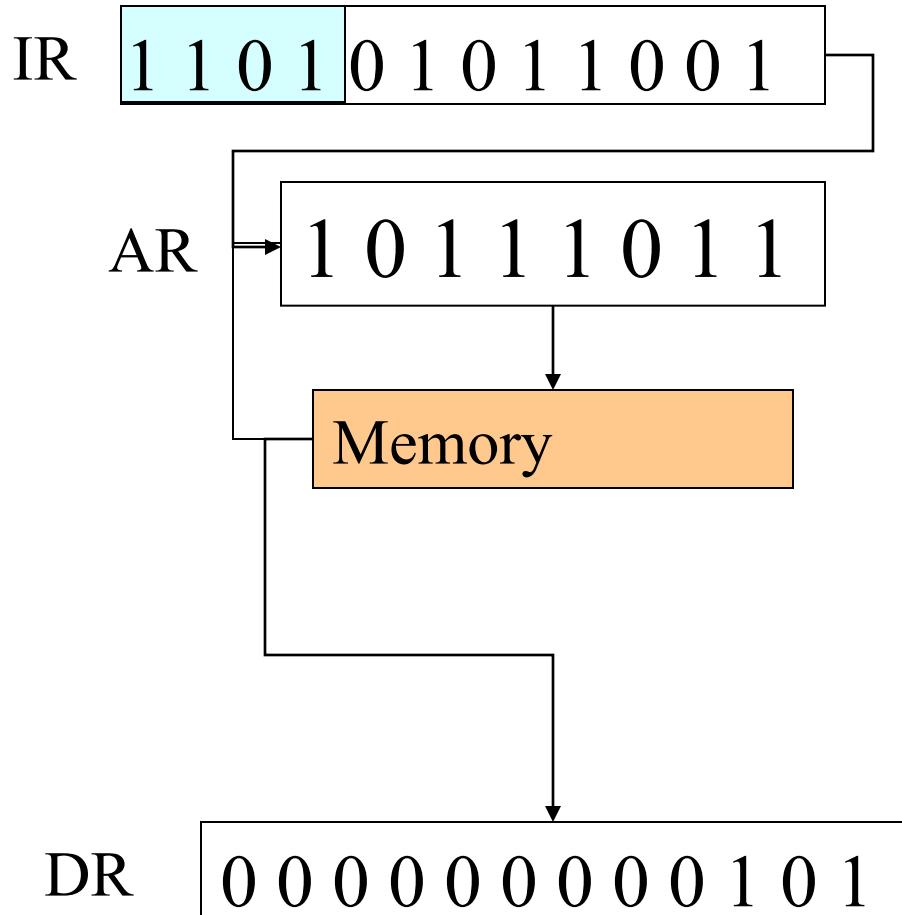
1. IR saklayıcısının adres alanı veriyoluna bırakılır ve AR saklayıcısına yüklenir.
2. Bellekteki adres seçilir ve veri yoluna veri bırakılır, DR saklayıcısına komut ile kullanılacak veri yüklenir.



# Dolaylı Adresleme

İhtiyaç duyulan verinin adresi, komutun adresinden alınıyorsa kullanılır.

1. IR saklayıcısındaki adres alanı veriyoluna bırakılır ve AR saklayıcısına yüklenir.
2. Bellekte Adres seçilir ve tekrar AR saklayıcısına yüklenir.
3. Yeni adres bellekte seçilir ve bellekteki veri veriyoluna bırakılır, DR saklayıcısına yüklenir.



# Bilgisayar Saklayıcıları

- Data Register(**DR**) : bellekten okunan işleneni (Data) tutar.
- Accumulator Register(**AC**) : genel amaçlı işlemci saklayıcısıdır.
- Instruction Register(**IR**) : bellekten okunan komutu tutar.
- Temporary Register(**TR**) : geçici veriyi tutar.
- Address Register(**AR**) : bellek adresini tutar, 12-bit genişliğindedir.
- Program Counter(**PC**) : komut adresini tutar, 12-bit genişliğindedir.
  - » Mevcut komut işletildikten sonra bir sonra okunacak komutun adresini tutar.
  - » Komut kelimeleri dallanma komutları ile karşılaşılmadığı sürece 1 artar.
  - » Bir dallanma komutu programın çalışmasını başka bir adrese atlatır.
  - » Dallanma komutunun adres alanı PC saklayıcısına bir sonra yürütülecek adres olarak atanır.
  - » Komutu okumak için bellek okuma döngüsü başlatılır ve PC 1 artırılır.

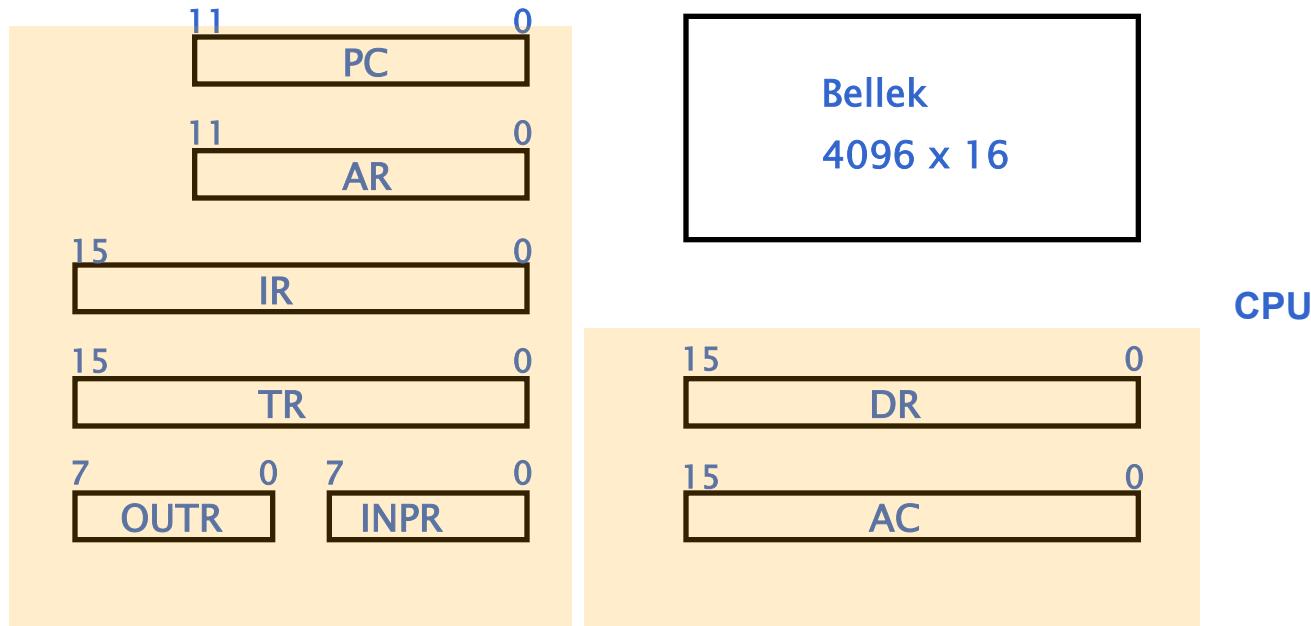
# Bilgisayar Saklayıcıları

- Input Register(**INPR**) : girdi cihazından 8-bit karakter veriyi alır.
- Output Register(**OUTR**) : çıktı cihazı için 8-bit karakter veriyi tutar.

## Temel Bilgisayar Saklayıcıları

Saklayıcı sembolu	Bit Sayısı	Saklayıcı Adı	Saklayıcının İşlevi-----
DR	16	Data register	Bellekteki işlenenleri tutar
AR	12	Address register	Bellek adresini tutar
AC	16	Accumulator	İşlemci saklayıcısı
IR	16	Instruction register	Komut kodunu tutar
PC	12	Program counter	Komutun adresini tutar
TR	16	Temporary register	Geçici veriyi tutar
INPR	8	Input register	Girdi karakterini tutar
OUTR	8	Output register	Çıktı karakterini tutar

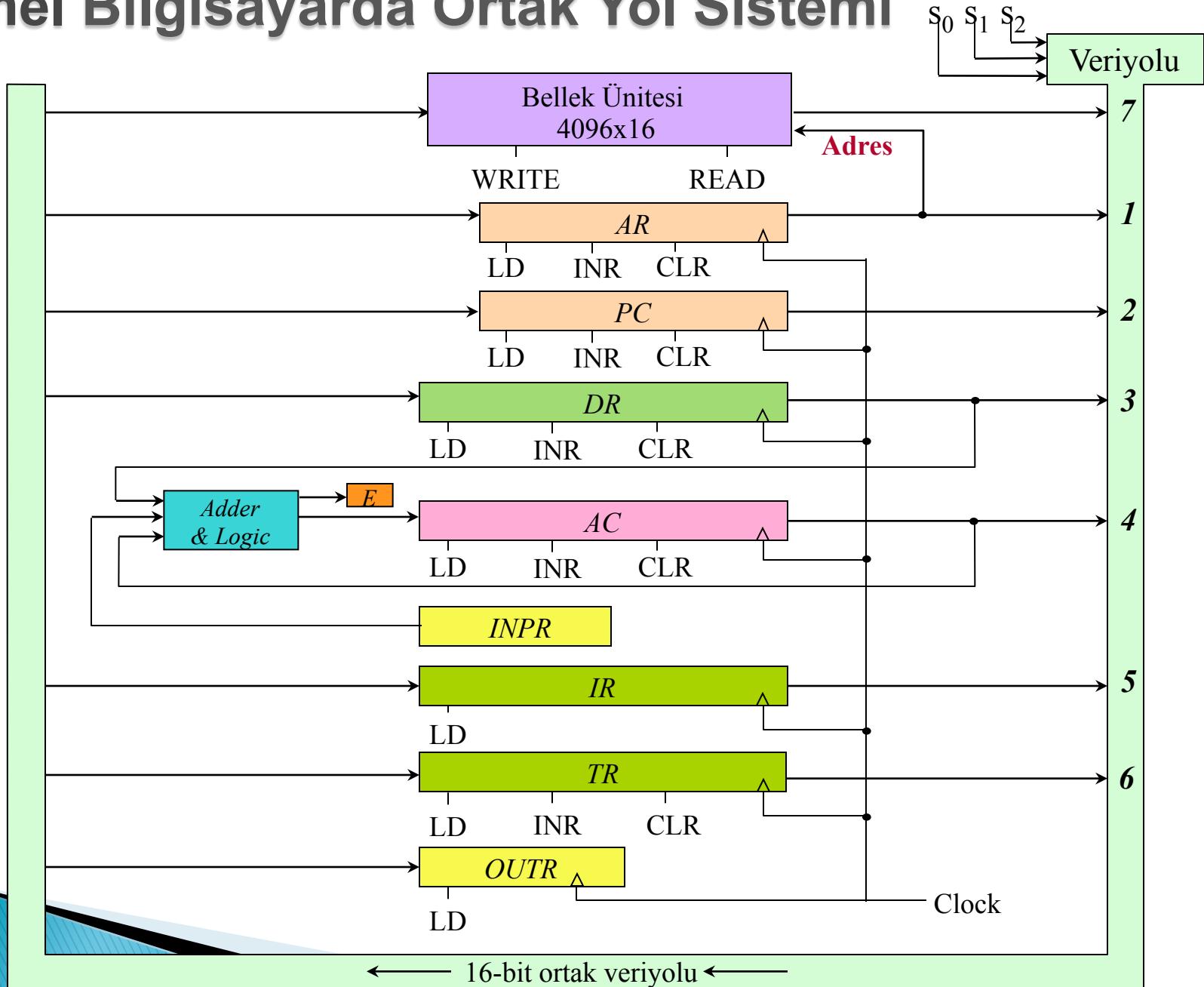
# Bilgisayar Saklayıcıları



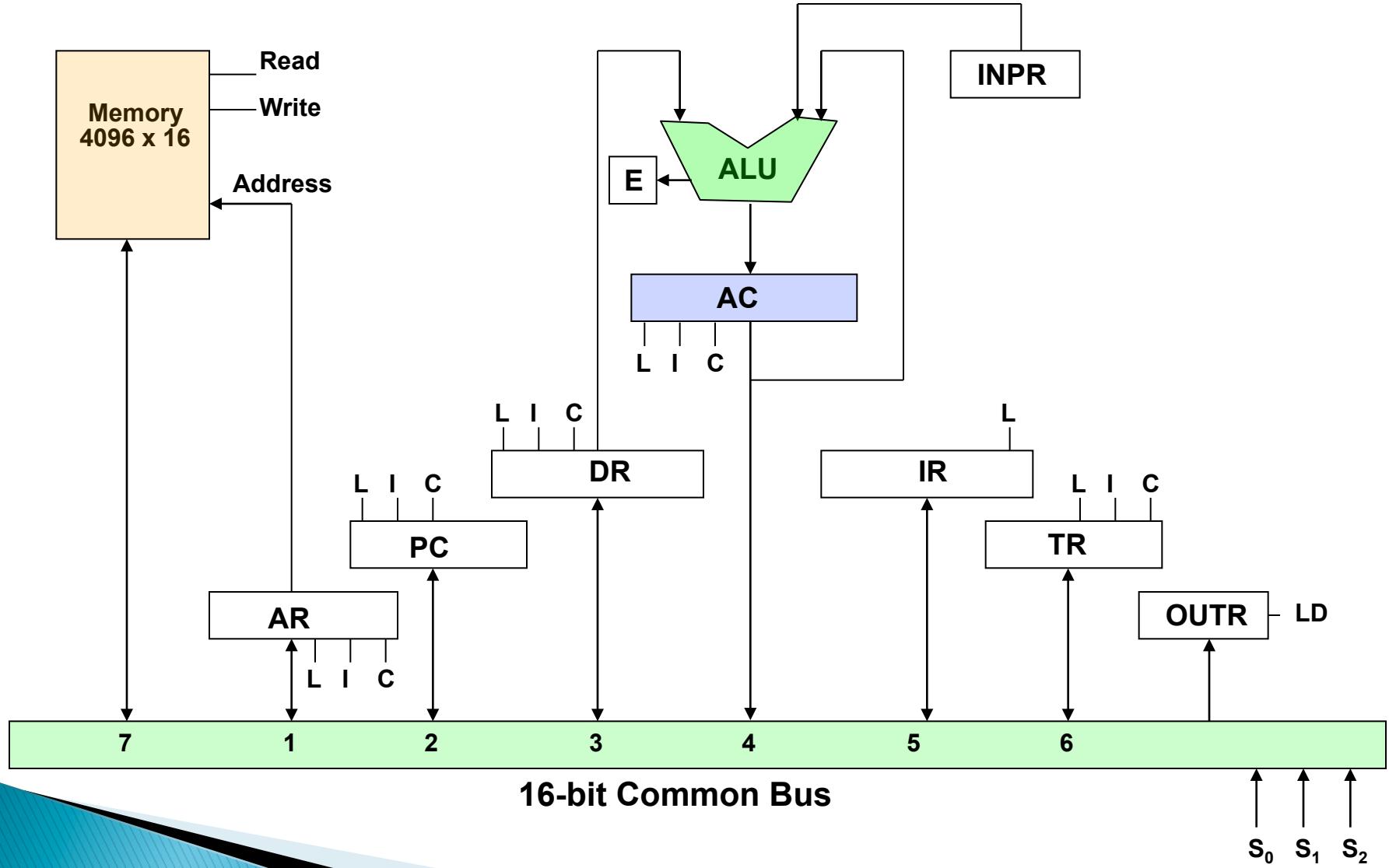
## Temel Bilgisayar Saklayıcıları

DR	16	Data Register	Bellek operandını tutar
AR	12	Address Register	Bellek için adres tutar.
AC	16	Accumulator	İşlemci saklayıcısı
IR	16	Instruction Register	Komut kodunu tutar.
PC	12	Program Counter	Bir komut adresi tutar
TR	16	Temporary Register	Geçici data tutar.
INPR	8	Input Register	Giriş karakterini tutar
OUTR	8	Output Register	Cıkış karakterini tutar.

# Temel Bilgisayarda Ortak Yol Sistemi



# Ortak Yol Sistemi



# Ortak Yol Sistemi

- 3 kontrol işaretini  $S_2$ ,  $S_1$ , ve  $S_0$  giriş olarak ortak yola hangi saklayıcının seçileceğini kontrol eder.

$S_2$	$S_1$	$S_0$	Register
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

- Saklayıcılarından birinin yükleme izni aktif olmalı veya bellek yaz işaretti aktif olmalıdır. (veri yolundan bilgi alma)
- 12-bit saklayıcılar olan AR ve PC, ortak yola transfer yaparken yüksek anlamlı 4 bitin hepsi 0 olarak yüklenirler.
- 8-bit saklayıcı olan OUTR ortak yolun düşük sekiz biti üzerinden transfer yapar.

# Ortak Veriyolu

- ▶ 6 saklayıcı ve belleğin çıktıları ortak veriyoluna bağlıdır.
  - İstenen çıktı Mux ile seçilir (S0, S1, S2) :
    - Memory(7), AR(1), PC(2), DR(3), AC(4), IR(5), TR(6)
      - LD (Load Input) sinyali aktifleştirildiğinde, veriyolundan seçilen saklayıcı bilgiyi alır.
      - Control Input : LD, INC, CLR, Write, Read
      - Increment input (INR): Saklayıcının içeriğini 1 artırır.
      - Clear input (CLR): Saklayıcının içeriğini temizler (0 atar).

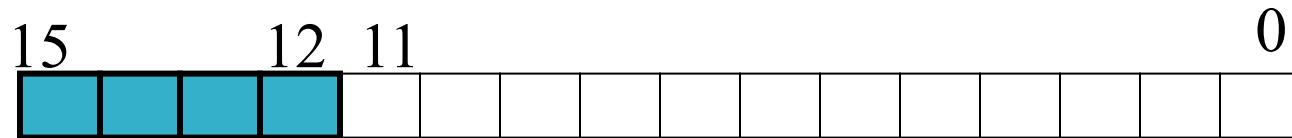
# Ortak Veriyolu

- ▶ PC ve AR saklayıcılarının içerikleri veriyoluna aktarıldığında, anlamlı 4 bit' e 0000 atanır. Bu saklayıcılara veriyolundan veri aktarılacağı zaman son 12 bit alınır.
- ▶ INPR ve OUTR saklayıcıları da son 8 biti kullanır.
- ▶ **Bellekteki adres hattına dikkat!**
- ▶ **AC saklayıcısına sadece DR, AC ve INPR saklayıcısından veri yüklenebilir!**

# Ortak Veriyolu

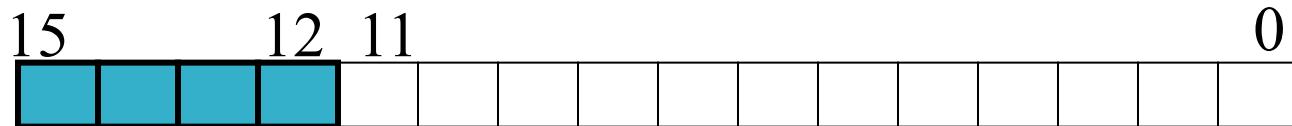
- ▶ DR $\leftarrow$ AC
  - s<sub>2</sub>s<sub>1</sub>s<sub>0</sub>=100 ve DR' nin LD sinyali aktif.
- ▶ DR $\leftarrow$ AC, AC $\leftarrow$ DR
  - Aynı zamanda gerçekleştirilir. s<sub>2</sub>s<sub>1</sub>s<sub>0</sub>=100 ve AC ve DR' nin LD sinyalleri aktif.

# Mano'nun Bilgisayarı: Bellek Kelimeleri



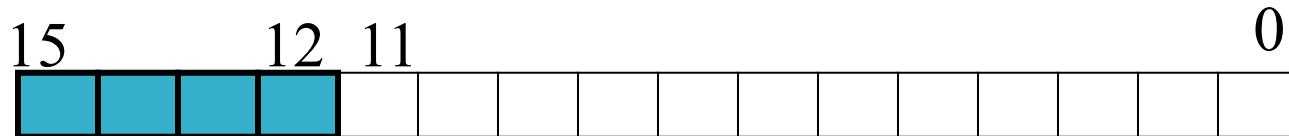
- ▶ 4-bit opcode 15-12. bitler
- ▶ Kaç tane komut vardır?
  - $2^4=16$
- ▶ Bu durum, 12 biti adres alanına bırakır.
  - Bellekte kaç tane kelime vardır?
  - $2^{12} = 2^2 \cdot 2^{10} = 4K = 4096$  tane 16-bit kelime

# Mano'nun Bilgisayarı: Bellek Kelimeleri



- 0111 numaralı opcode saklayıcı-referanslı komutlara ayrılmıştır.
- Kaç tane saklayıcı-referanslı komut vardır?
- $2^{12} = 2^2 \cdot 2^{10} = 4K = 4096$  r-r komutu  
(sadece 12 tanesi kullanılmaktadır.)

# Mano'nun Bilgisayarı: Bellek Kelimeleri



- ▶  $2^4 = 16$  komut
  - 1111 numaralı opcode giriş/çıkış işlemleri için ayrılmıştır.
- ▶  $2^4 = 16$  komut - 0111 (r-r) - 1111 (i/o) = 14 komut kalır.
  - Kalan bu komutlarla 7 tane komut hem doğrudan hem de dolaylı adresleme için oluşturulmuştur.

# Temel Bilgisayar Komutları

- Temel Bilgisayar Komut Formatı

Bellek–Referanslı Komutlar

İşlem Kodu (opcode) = 000 ~ 110



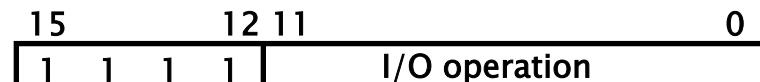
Saklayıcı–Referanslı Komutlar

İşlem Kodu (opcode) = 111, I = 0



I/O – Referanslı Komutlar

İşlem Kodu (opcode) = 111, I = 1



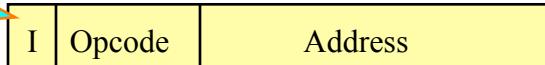
# Temel Bilgisayar Komutları

- 3 farklı komut formatı bulunmaktadır:

- Bellek-referanslı komutlar için

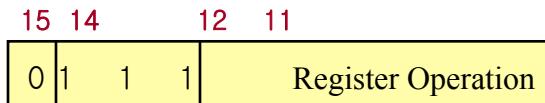
- Opcode = 000 ~ 110

I=0 : 0xxx ~ 6xxx, I=1: 8xxx ~Exxx



- Saklayıcı-referanslı komutlar için

- 7xxx (7800 ~ 7001) : CLA, CMA,



- Giriş-Çıkış komutları için

- Fxxx(F800 ~ F040) : INP, OUT, ION, SKI,



Symbol	Hex Code		Description
	I = 0	I = 1	
AND	0xxx	8xxx	And memory word to AC
ADD	1xxx	9xxx	Add memory word to AC
LDA	2xxx	Axxx	Load memory word to AC
STA	3xxx	Bxxx	Store content of AC in memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and Save return address
ISZ	6xxx	Exxx	Increment and skip if zero
CLA	7800		Clear AC
CLE	7400		Clear E
CMS	7200		Complement AC
CME	m	7100 e	Comp
CIR	7080		Circulate right AC and E
CIL	7040		Circulate left AC and E
INC	7020		Increment AC
SPA	7010		Skip next instruction if AC positive
SNA	7008		Skip next instruction if AC negative
SZA	7004		Skip next instruction if AC zero
SZE	7002		Skip next instruction if E is 0
HLT	7001		Halt computer
INP	F800		Input character to AC
OUT	F400		Output character from AC
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag
ION	F080		Interrup
IOF	F040		Inter

Tabloyu okumak için:

Tablodaki her kod 16 bit genişliğindedir. xxx dikkate almama durumunu gösterir (son 12 bit için herhangi bir veri). Örneğin 7002 onaltılık sistemdeki gösterimi 0111 0000 0000 0010. En soldaki bit 0 olduğu ve gerisi de 111 olduğu için saklayıcı-referanslı koddur.

# Temel Bilgisayar Komutları

- ▶  $7+12+6 = 25$  komut
  - 3 harfle gösterilen kısaltmalar programcı ve kullanıcılar için verilmiştir.
  - 16'lık kodlarla 16 biti 4 basamağa indirmīş oluruz.
- ▶ Örnekler:
  - 8XXX = 1000.xxxx.xxxx.xxxx (AND)
  - 7800 = 0111.1000.0000.0000 (CLA)
  - F800 = 1111.1000.0000.0000 (INP)

# Temel Bilgisayar Komutları

Symbol	Hexadecimal code		Description
	$I = 0$	$I = 1$	
AND	0xxx	8xxx	AND memory word to <i>AC</i>
ADD	1xxx	9xxx	Add memory word to <i>AC</i>
LDA	2xxx	Axxx	Load memory word to <i>AC</i>
STA	3xxx	Bxxx	Store content of <i>AC</i> in memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero
CLA	7800		Clear <i>AC</i>
CLE	7400		Clear <i>E</i>
CMA	7200		Complement <i>AC</i>
CME	7100		Complement <i>E</i>
CIR	7080		Circulate right <i>AC</i> and <i>E</i>
CIL	7040		Circulate left <i>AC</i> and <i>E</i>
INC	7020		Increment <i>AC</i>
SPA	7010		Skip next instruction if <i>AC</i> positive
SNA	7008		Skip next instruction if <i>AC</i> negative
SZA	7004		Skip next instruction if <i>AC</i> zero
SZE	7002		Skip next instruction if <i>E</i> is 0
HLT	7001		Halt computer
INP	F800		Input character to <i>AC</i>
OUT	F400		Output character from <i>AC</i>
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag
ION	F080		Interrupt on
IOF	F040		Interrupt off

# Komut Kümesinin Tamlığı

- ▶ Bir bilgisayarın komut kümesinde aşağıdaki komutlardan yeterince varsa, komut kümesi TAM'dır olarak adlandırılır:
  - Aritmetik, mantık ve kaydırma komutları,
  - Bilginin bellek ve saklayıcılar arasında aktarılmasını sağlayan komutlar,
  - Durumu belirleyen komutlarla program denetim komutları,
  - G/Ç komutları.
- ▶ Temel bilgisayarın komut kümesi TAM mıdır?

# Komut Kümesinin Tamliği

Bir bilgisayarın komut seti bütünlüğü, hesaplanabilir olarak bilinen herhangi bir fonksiyonu değerlendirmek için, kullanıcının makine dili komutları kullanarak bir program oluşturabilmesidir.

## • Komut Türleri

### Fonksiyonel Komutlar:

- Aritmetik, lojik, ve öteleme komutları
- ADD, CMA, INC, CIR, CIL, AND, CLA

### Transfer Komutları

- Bellek–işlemci saklayıcıları arasındaki data transferleri
- LDA, STA

### Kontrol komutları

- Program düzenleme ve kontrol
- BUN, BSA, ISZ

### Giriş/Çıkış Komutları

- Giriş ve çıkış
- INP, OUT

# Kontrol Birimi

- ▶ Bir işlemci kontrol ünitesi, makine komutlarından mikroişlemlerin gerçekleştirilmesi için üretilmesi gereken kontrol işaretlerine dönüşümü sağlar.
- ▶ Kontrol ünitesi 2 şekilde gerçekleşir:
- ▶ ***Donanımsal Kontrol***
  - Kontrol Ünitesi, kontrol işaretlerini üretmek için gereken ALD ve KLD devrelerinden oluşturulur.
- ▶ ***Yazılımsal (Microprogrammed) Kontrol***

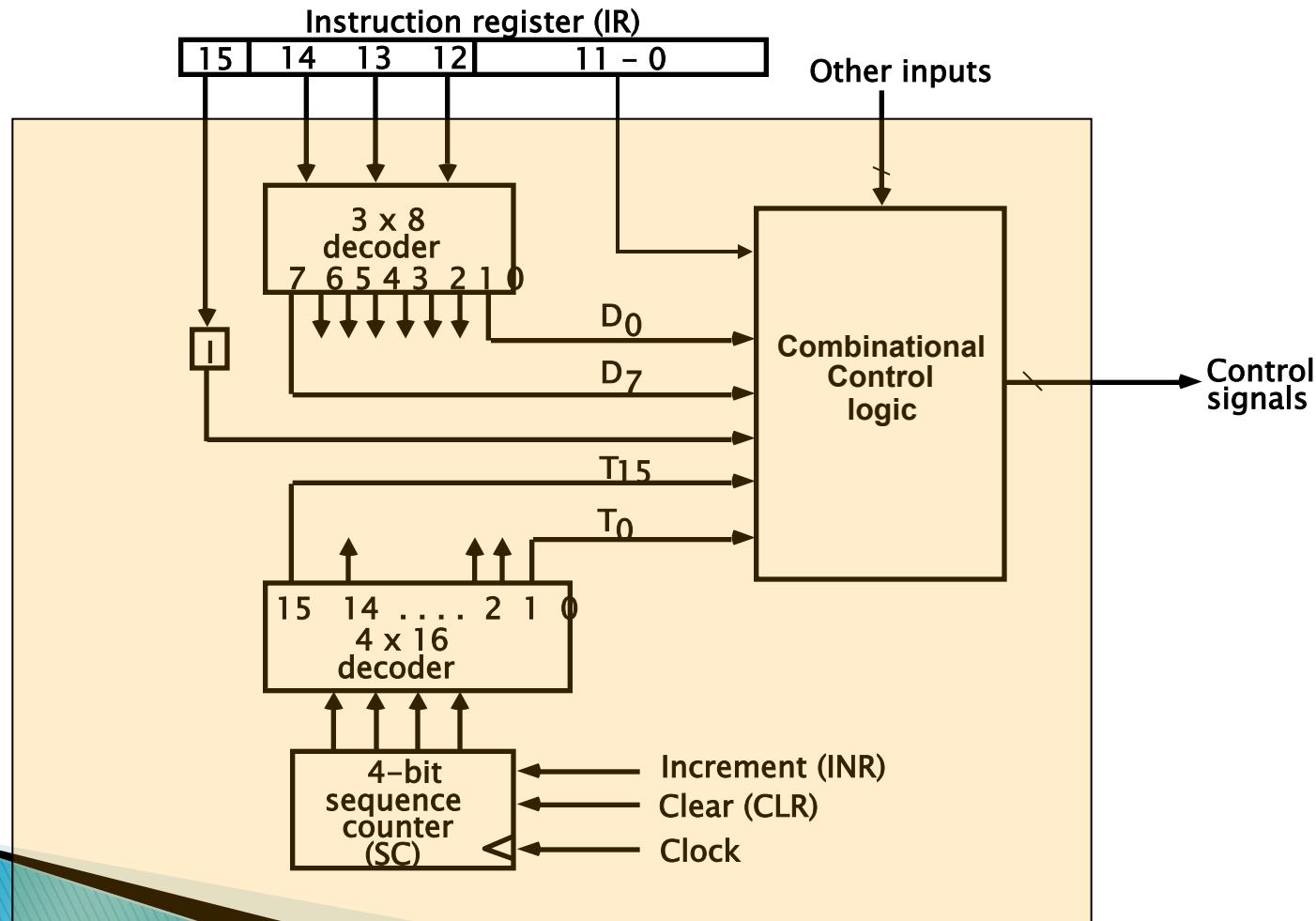
Bir kontrol belleğinde gerekli kontrol işaretlerini aktifleştirecek olan mikroprogramların yazılmasıdır.

  - Kontrol bilgisi kontrol belleğinde saklanır ve istenen mikroişlemlerin sırası başlatılır.
  - İstenen değişiklik kontrol belleğindeki mikroprogram değiştirilerek elde edilebilir.

**Burada, temel bilgisayarın kontrol ünitesi donanımsal olarak geliştirilecektir.**

# Zamanlama ve Kontrol

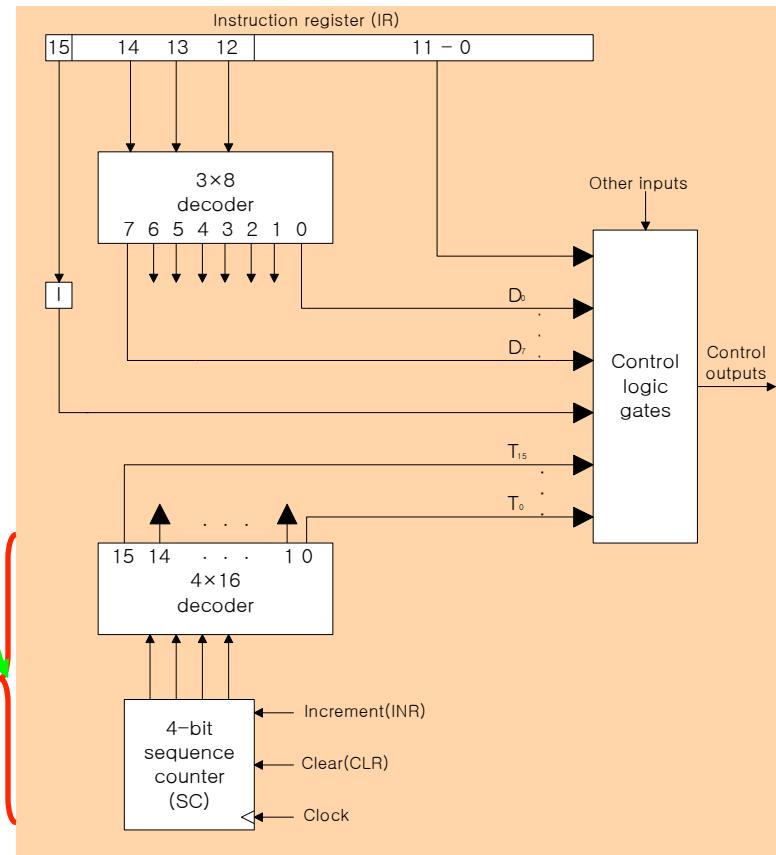
## Temel Bilgisayarın Kontrol Ünitesi



# Zamanlama ve Kontrol

## ■ Kontrol Birimi:

- Kontrol Birimi= Kontrol Lojik Kapıları + 3x8 Decoder + (Instruction Register) + Zamanlama Sinyali
- Zamanlama Sinyali = 4x16 Decoder + 4-bit Sequence Counter (Sıra Sayıcı)
- Örneğin, Kontrol zamanlamasında :  $D_3 T_4 : SC \leftarrow 0$ 
  - $D_3 T_4 = 1$  olduğunda Sıra Sayıcı temizlenir.

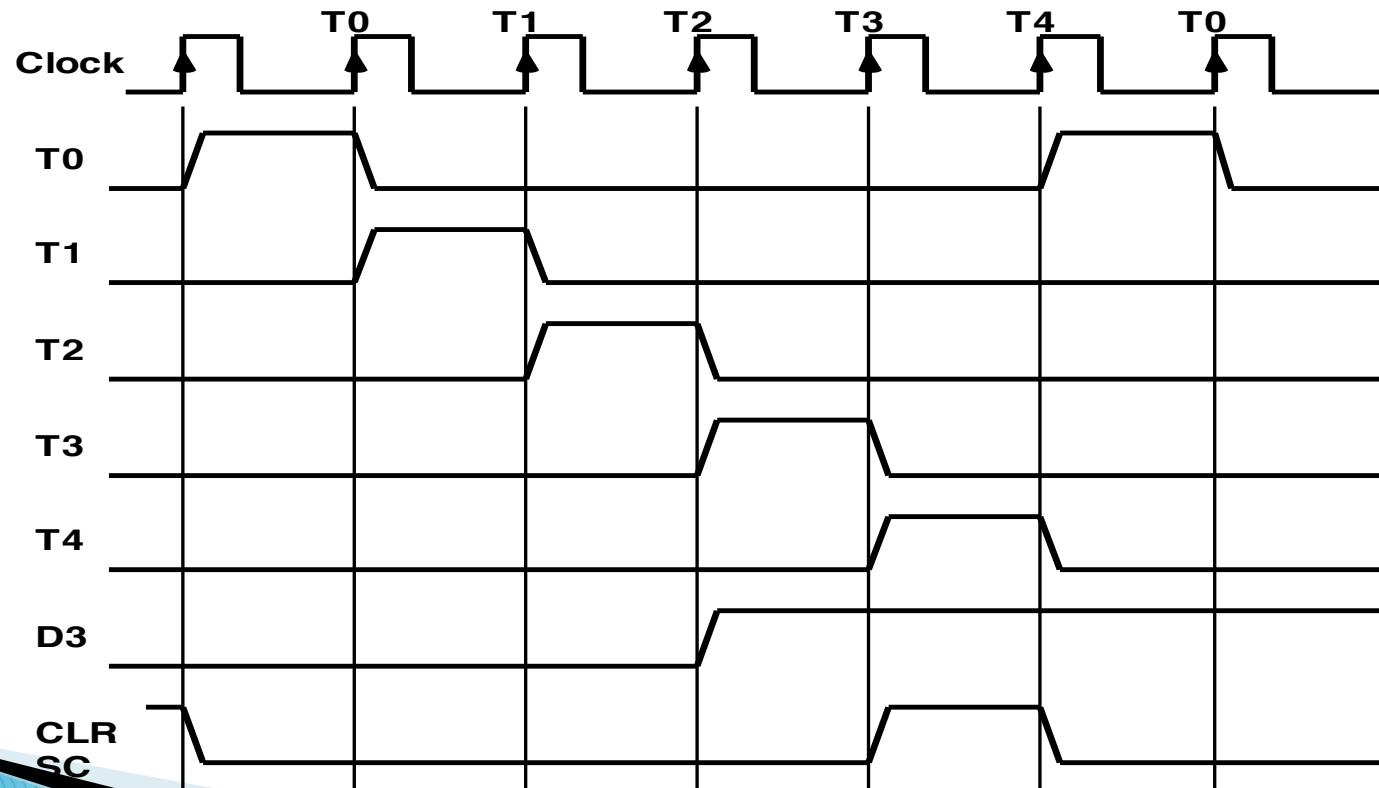


# Zamanlama İşaretleri

- 4-bit sıra sayıcı (SC) ve  $4 \times 16$  bir decoder (kod çözücü)
- SC içeriği 1 artırılabilir veya sıfırlanabilir.
- Örnek:  $T_0, T_1, T_2, T_3, T_4, T_0, T_1, \dots$

Kabul:  $T_4$  anında SC sıfırlanır (D3 decoder çıkışı aktifse)

$D_3 T_4: SC \leftarrow 0$



# Komut Döngüsü

- ▶ Temel Bilgisayarda, bir makine komutu sırasıyla aşağıdaki aşamalar halinde icra edilir. :
  1. Bellekten komut alma işlemi (fetching)
  2. Komutun kod çözümü (decoding)
  3. Komut dolaylı adresle kullanıyorsa, bellekten efektif adres okuma
  4. Komut icrası
- ▶ Bir komut icra edildikten sonra, bir sonraki komut için, 1. adımdaki çevrim tekrar başlar.
- ▶ *Not:* Her farklı işlemci, kendine özgü ayrı bir komut çevrimine sahiptir.

# Komut Döngüsü

- ▶ Komutun bellekten getirilip(fetch) ve çözülmesi(decode) aşamaları aşağıdaki mikroişlemlerden ve zamanlama sinyallerinden oluşmaktadır.

Zamanlama

Sinyali

T0:

T1:

T2:

Mikroişlemler

**$AR \leftarrow PC$**

**$IR \leftarrow M[AR], PC \leftarrow PC + 1$**

**$D0, \dots, D7 \leftarrow \text{Decode } IR(12-14),$**   
 **$AR \leftarrow IR(0-11), I \leftarrow IR(15)$**

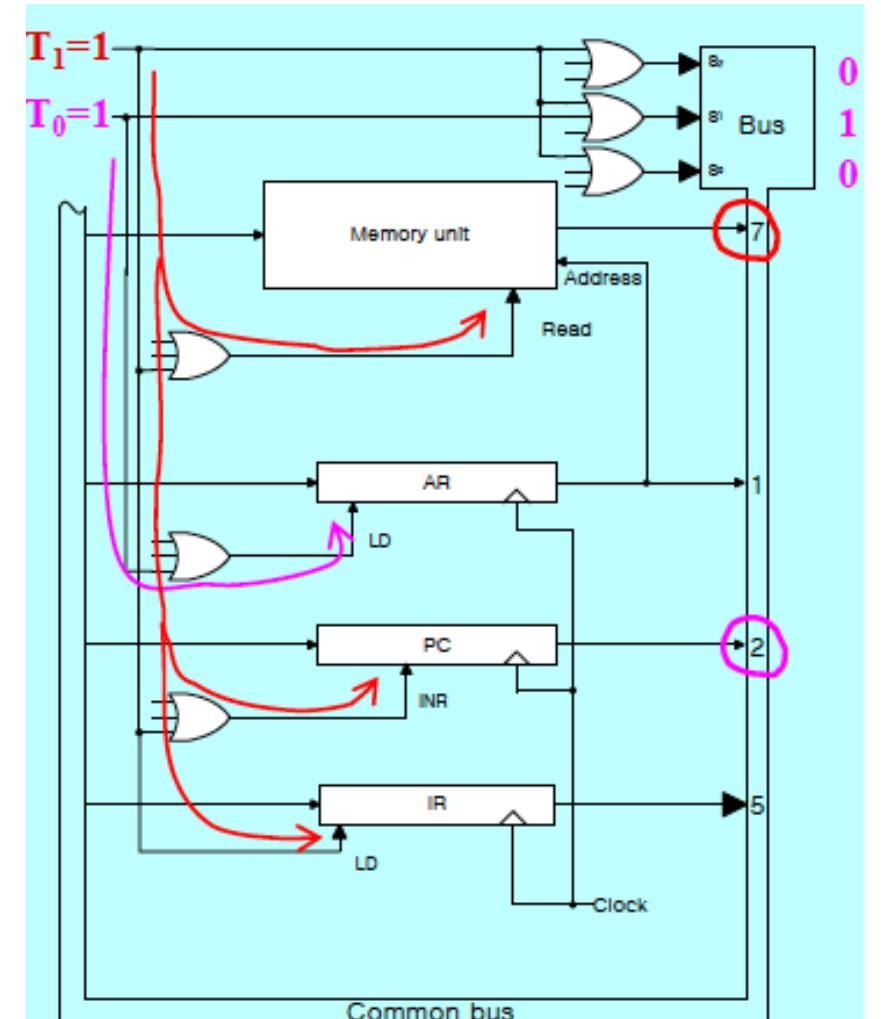
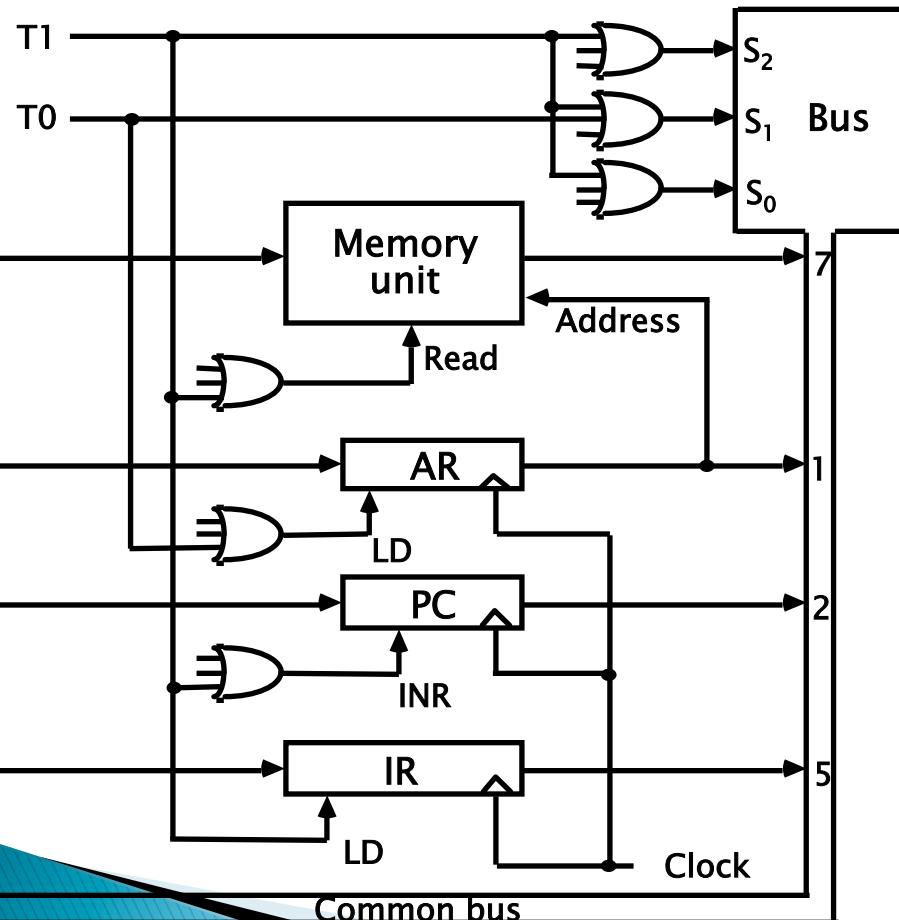
# Komut alma (fetch) ve kod çözümü (decode)

- Fetch ve Decode

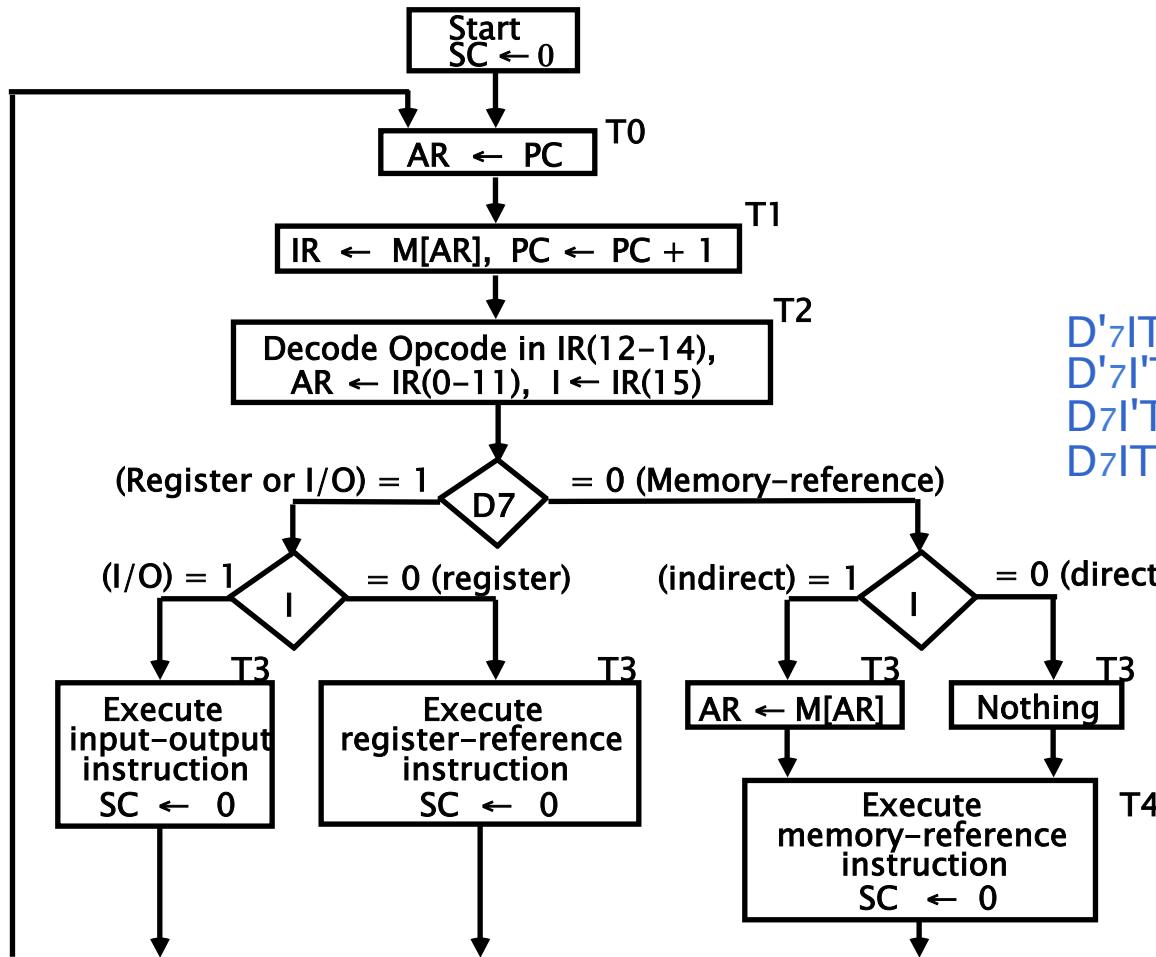
T0:  $AR \leftarrow PC$  ( $S_0S_1S_2=010$ ,  $T0=1$ )

T1:  $IR \leftarrow M[AR]$ ,  $PC \leftarrow PC + 1$  ( $S_0S_1S_2=111$ ,  $T1=1$ )

T2:  $D_0, \dots, D_7 \leftarrow \text{Decode IR}(12-14)$ ,  $AR \leftarrow IR(0-11)$ ,  $I \leftarrow IR(15)$



# Komut Türünün Belirlenmesi Akışı



D<sub>7</sub>'IT<sub>3</sub>: AR ← M[AR]

D<sub>7</sub>I'T<sub>3</sub>: Hiçbir şey

D<sub>7</sub>I'T<sub>3</sub>: Saklayıcı referanslı komut icrası

D<sub>7</sub>IT<sub>3</sub>: I/O referanslı komut icrası.

IR(12-14)  
= 111

$\left\{ \begin{array}{l} D_7=1 \\ \quad \left\{ \begin{array}{l} \text{Register}(I=0) \rightarrow D_7I'T_3(\text{Execute}) \\ \text{I/O} \quad (I=1) \rightarrow D_7IT_3 (\text{Execute}) \end{array} \right. \\ D_7=0 : \text{Memory Ref.} \end{array} \right.$

Read effective  
Address

$\left\{ \begin{array}{l} \text{Indirect}(I=1) \rightarrow D_7'IT_3(AR \leftarrow M[AR]) \\ \text{Direct } (I=0) \rightarrow \text{nothing in } T_3 \end{array} \right.$