

BİLGİSAYAR ORGANİZASYONU ve TASARIMI

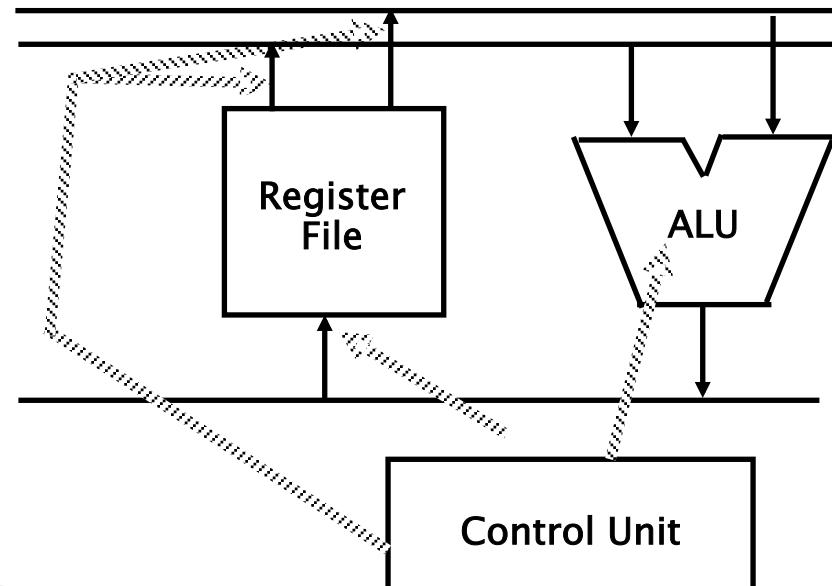
DR. FATİH KELEŞ

Merkezi İşlemci Birimi - MİB (CPU)

- ▶ Giriş
- ▶ Genel Saklayıcı Organizasyonu
- ▶ Yığın Organizasyonu
- ▶ Komut Formatları
- ▶ Adresleme Modları
- ▶ Data Transferi ve İşlemleri
- ▶ Program Kontrolü
- ▶ Azaltılmış Komut Kümeli Bilgisayar – RISC

CPU Temel Bileşenleri

- ▶ Bilgi Saklama Elemanları
 - ▶ Saklayıcılar (register)
 - ▶ Bayraklar (flip-flop)
- ▶ İşlem Elemanları
 - ▶ Aritmetik Lojik Ünite (ALU)
 - ▶ Aritmetik hesaplamalar, Lojik hesaplamalar, Öteleme/Döndürme
- ▶ Transfer Elemanı
 - ▶ Ortak Yol – Bus
- ▶ Kontrol Elemanı
 - ▶ Kontrol Birimi



CPU Komut İşleme Yapısı

► CPU :

- Komutun getirilmesi (Fetch instructions)
- Komutun yorumlanması (Interpret instructions)
- Verinin getirilmesi (Fetch data)
- Verinin işlenmesi (Process data)
- Verinin yazılması (Write data)

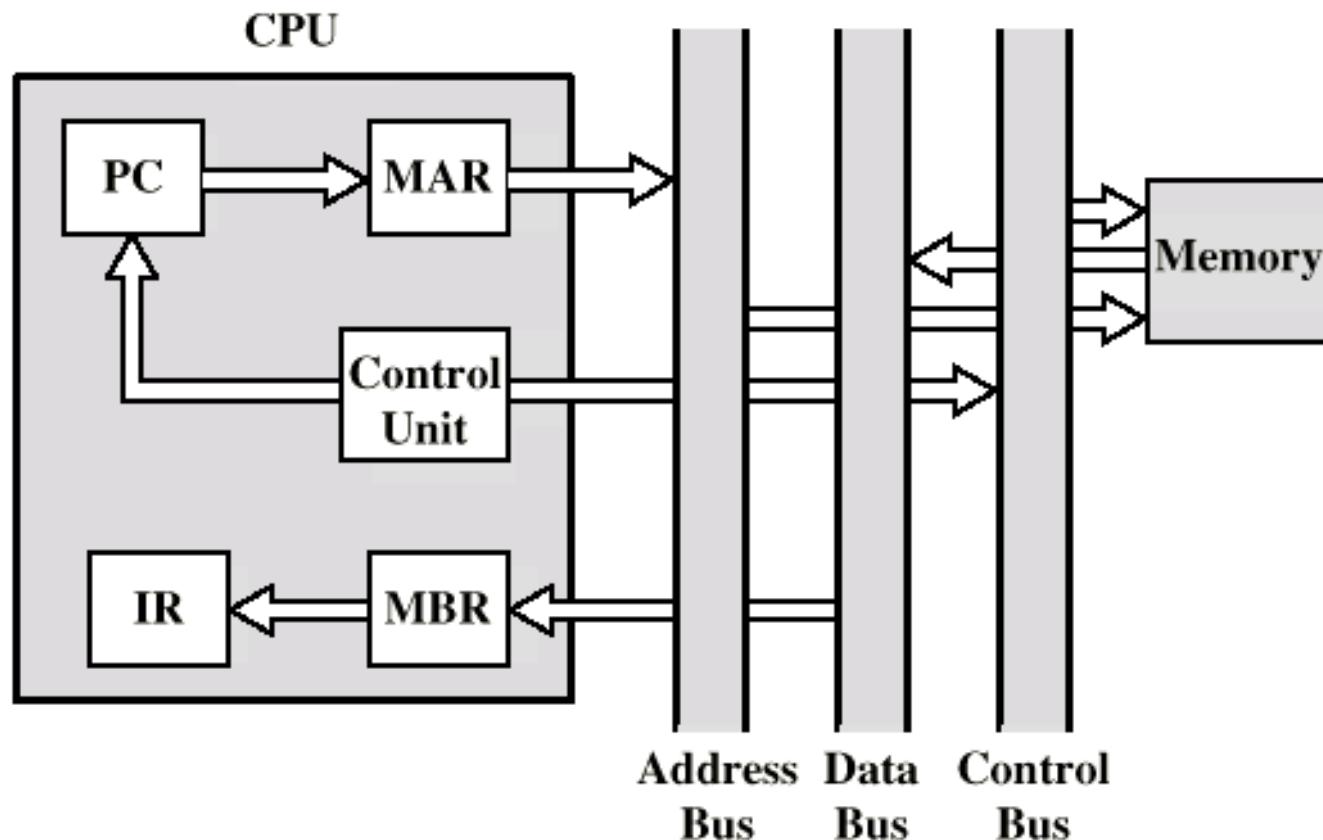
Veri Akışı (Instruction Fetch)

- ▶ CPU tasarımına bağlı olarak değişir
- ▶ Genelde:
 - Fetch işlemi
 - PC bir sonraki komut adresini taşıır
 - Adres MAR a transfer edilir
 - Adres Yolu üzerine adres çıkarılır
 - Kontrol Ünitesi Bellek Oku işaretini gönderir
 - Data bus üzerine data alınır, MBR bir kopyası transfer edilir ve IR komut saklayıcıya komut gönderilir
 - $PC = PC + 1$ yapılır..

Veri Akışı (Data Fetch)

- ▶ IR incelenir
- ▶ Dolaylı adreslemeli ise, dolaylı adres okunur
 - MBR, MAR a transfer edilir
 - Operand adresi MBR a iletılır.

Veri Akışı (Fetch Diagram)



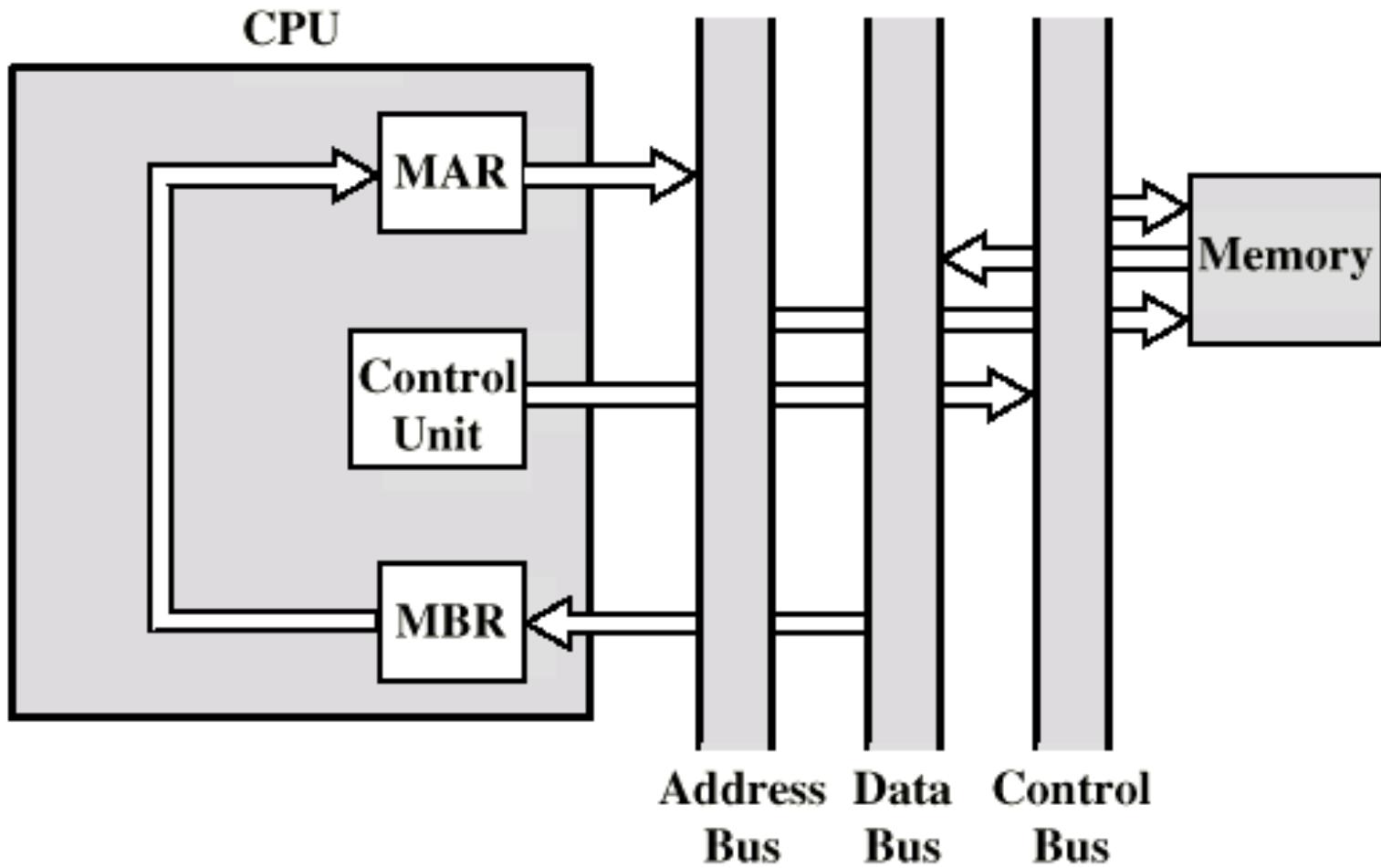
MBR = Memory buffer register

MAR = Memory address register

IR = Instruction register

PC = Program counter

Veri Akışı (Indirect Diagram)



İşlemci Organizasyonu

- ▶ **Tek Saklayıcı (Akümülatör) Organizasyonu**
 - TBO iyi bir örnektir. Akümülatör sadece tek genel amaçlı saklayıcı olarak kullanılır.
- ▶ **Genel Saklayıcı Organizasyonu**
 - ▶ Pek çok modern bilgisayar işlemcisi tarafından kullanılır.
 - ▶ Herhangi bir saklayıcı bilgisayar işlemleri için kaynak veya hedef olarak kullanılabilir.
- ▶ **Yığın Organizasyonu**
 - ▶ Bütün işlemler donanımsal yığın kullanılarak yapılır.
 - ▶ Örnek, OR komutu yığından üstteki 2 elemanı çeker, lojik OR işlemini gerçekler ve sonuçları yığına atar.

Saklayıcı

- ▶ CPU geçici bilgi tutmak için saklayıcı kullanır (temporary storage)
- ▶ İşlemci türüne göre sayıları ve fonksiyonları farklılık göstermektedir
- ▶ Saklayıcı kümесinin seçimi tasarımın önemli bir adımıdır.
- ▶ Bellek hiyerarşisi içinde de en üst seviyede bulunurlar.

Kullanılan Saklayıcı Türleri

- Genel Amaçlı (General Purpose)
- Veri (Data -Accumulator-)
- Adres (Address –segment register, AR, PC–)
- Durum Kodları (Condition Codes –status register–)
- Komut Çözme (Instruction Decoding Register)

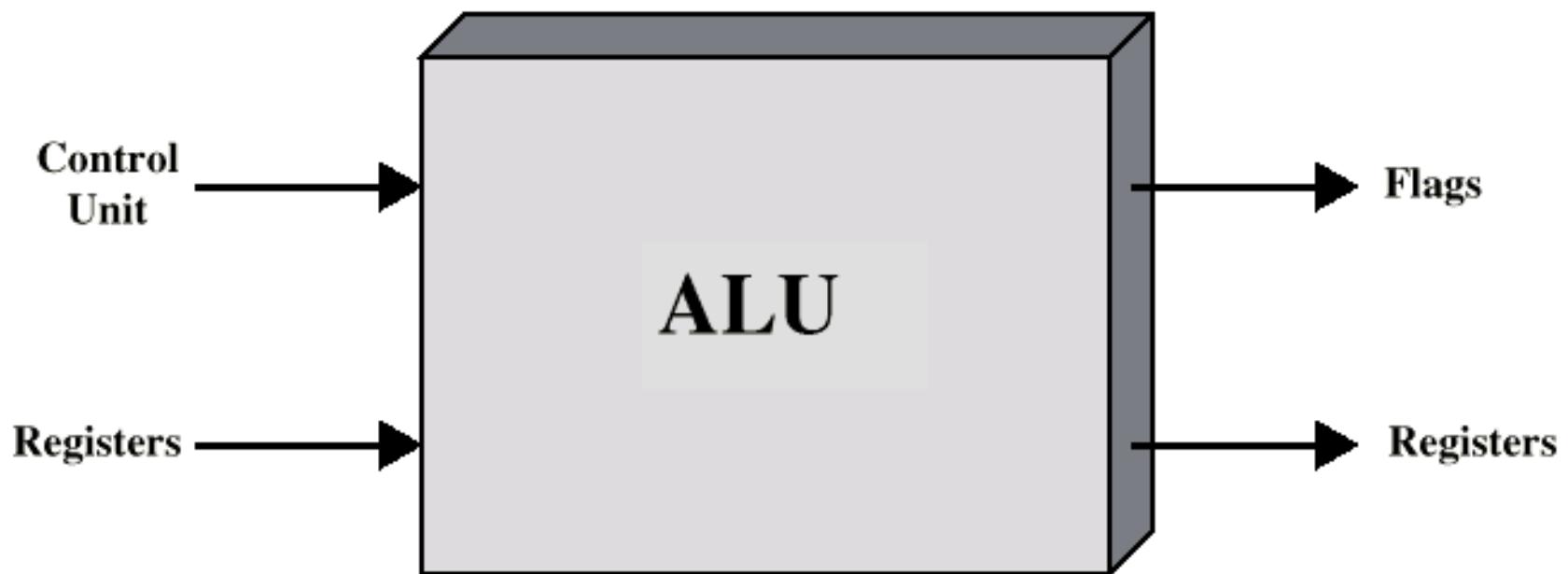
Program Durum Kelimesi (Status Word)

- ▶ Bit kümesi (A set of bits)
- ▶ Durum kodları (Includes Condition Codes)
- ▶ Son sonuç işaretleri (Sign of last result)
- ▶ Sıfır (Zero)
- ▶ Elde (Carry)
- ▶ Eşit (Equal)
- ▶ Taşma (Overflow)
- ▶ Kesme izni (Interrupt enable/disable)
- ▶ Yönetici (Supervisor)

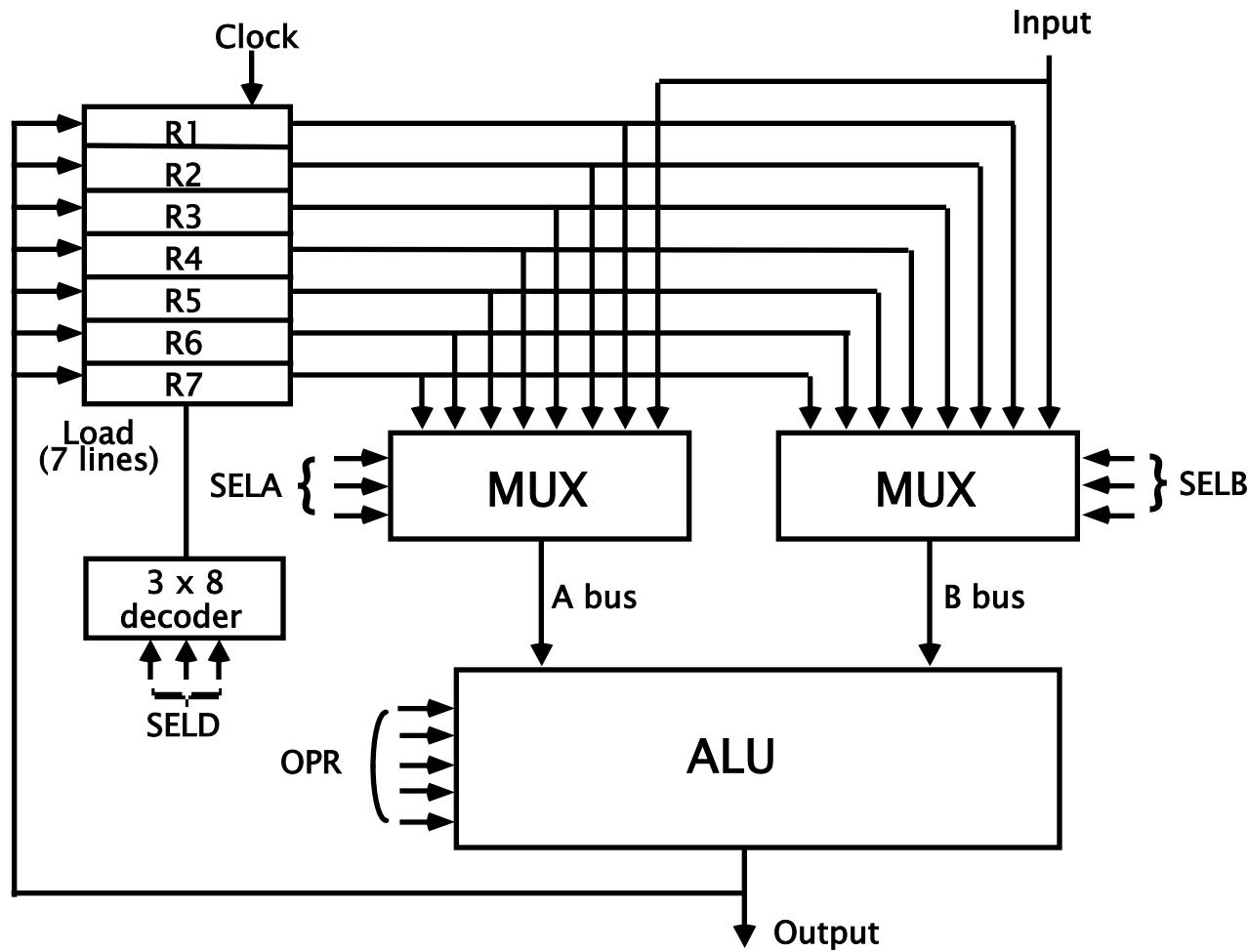
Aritmetik Lojik Birim (ALU)

- ▶ Bütün Hesaplamalar
- ▶ Tamsayı İşleme
- ▶ Kayan Noktalı Sayı İşleme (real numbers)
- ▶ Ayrı bir yardımcı işlemci FPU (maths co-processor)
- ▶ FPU (486DX +)

ALU Giriş ve Çıkışları



Genel Saklayıcı Organizasyonu



Kontrol Birimi

Kontrol Birimi

ALU ya bilgi akışını yönlendirir:

- Sistemdeki çeşitli elemanların seçimleri
- ALU fonksiyonlarının seçimi

Örnek: $R1 \leftarrow R2 + R3$

- | | |
|----------------------------|----------------------------------|
| [1] MUX A seçme girişi | (SEL A): BUS A $\leftarrow R2$ |
| [2] MUX B seçme girişi | (SEL B): BUS B $\leftarrow R3$ |
| [3] ALU işlem seçimi | (OPR): ALU to ADD |
| [4] Kodçözücü hedef seçimi | (SEL D): R1 \leftarrow Out Bus |

Kontrol Kelimesi

3	3	3	5
SEL A	SEL B	SEL D	OPR

Register Seçme Kodları

Binary Code	SEL A	SEL B	SEL D
000	Input	Input	None
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

ALU Kontrol

ALU işlem kodları

İşlem Seçim	İşlem	Sembol
00000	Transfer A	TSFA
00001	Increment A	INCA
00010	ADD A + B	ADD
00101	Subtract A – B	SUB
00110	Decrement A	DECA
01000	AND A and B	AND
01010	OR A and B	OR
01100	XOR A and B	XOR
01110	Complement A	COMA
10000	Shift right A	SHRA
11000	Shift left A	SHLA

ALU Örnek Mikroişlemleri

Mikroişlem	Sembolik Gösterim					Kontrol Kelimesi
	SELA	SELB	SELD	OPR		
R1 ← R2 – R3	R2	R3	R1	SUB	010 011 001	00101
R4 ← R4 v R5	R4	R5	R4	OR	100 101 100	01010
R6 ← R6 + 1	R6	–	R6	INCA	110 000 110	00001
R7 ← R1	R1	–	R7	TSFA	001 000 111	00000
Output ← R2	R2	–	None	TSFA	010 000 000	00000
Output ← Input	Input	–	None	TSFA	000 000 000	00000
R4 ← shl R4	R4	–	R4	SHLA	100 000 100	11000
R5 ← 0	R5	R5	R5	XOR	101 101 101	01100

Saklayıcı Yığın Organizasyonu

Yığın

- Altprogramda çok faydalı bir özellikleştir, Kesme Servis Rutinlerinde
- Aritmetik ifadelerin etkin değerlendirmesi
- LIFO türü yığın işlemi
- Yığın Göstergesi: SP
- PUSH ve POP işlemleri uygulanabilir

Register Stack

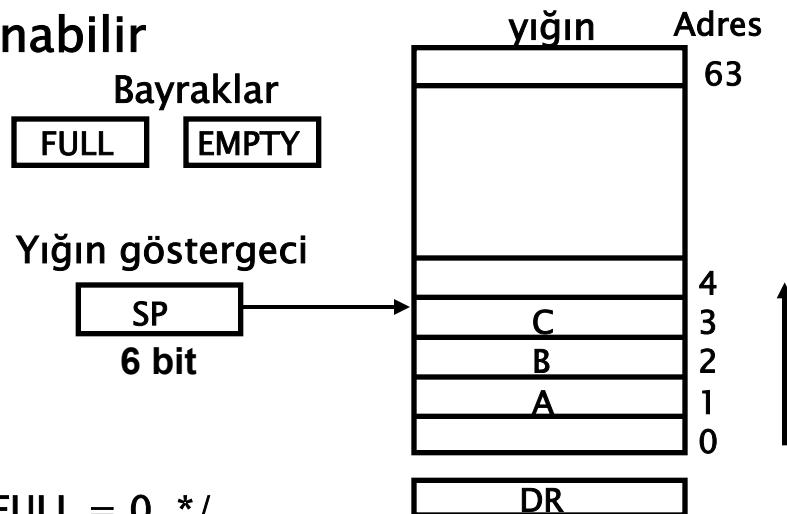
/* başlangıçta, SP = 0, EMPTY = 1, FULL = 0 */

PUSH

```
SP ← SP + 1  
M[SP] ← DR  
If (SP = 0) then (FULL ← 1)  
EMPTY ← 0
```

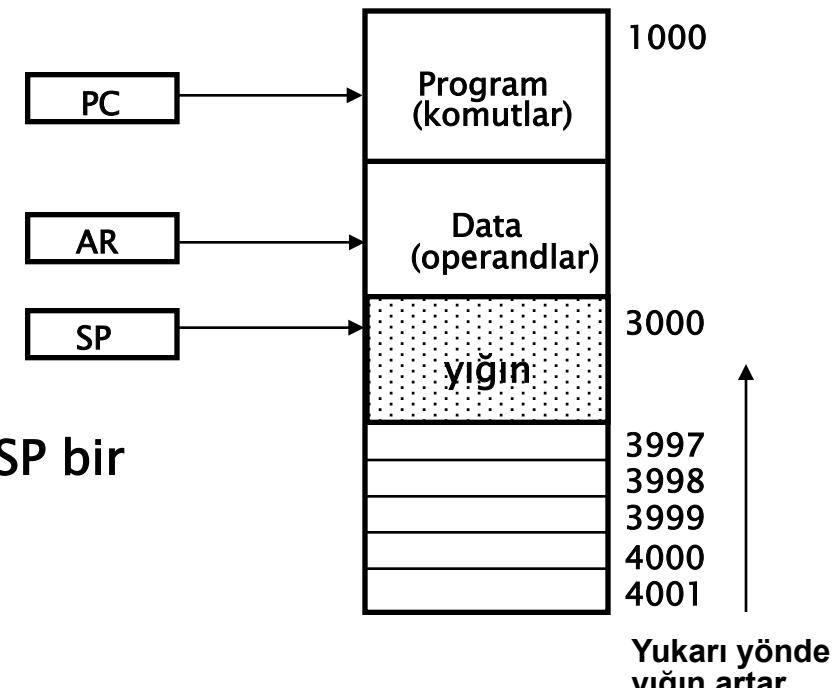
POP

```
DR ← M[SP]  
SP ← SP - 1  
If (SP = 0) then (EMPTY ← 1)  
FULL ← 0
```



Bellek Yiğin Organizasyonu

BELLEK:
Program, Data ve Yiğin Alanları



- Yiğin göstergeci olarak kullanılan SP bir işlemci saklayıcısı gibi kullanılır:
- PUSH: $SP \leftarrow SP - 1$
 $M[SP] \leftarrow DR$
- POP: $DR \leftarrow M[SP]$
 $SP \leftarrow SP + 1$
- Çoğu bilgisayarlar yiğin doluluk/böşlük kontrolü için donanım kullanmazlar → yazılımsal olarak bunu yaparlar.

Sonekli (Postfix - Zıt Polonyalı) Gösterim

- Aritmetik İfadeler: $A + B$

$A + B$ Infix gösterimi

$+ A B$ ÖNEKLİ gösterim (Polish)

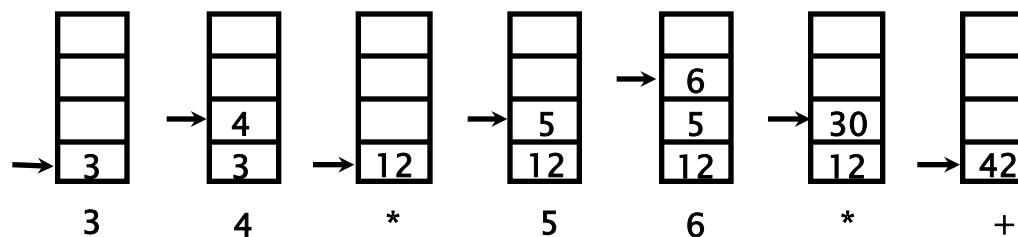
$A B +$ SONEKLİ gösterim (Reverse Polish)

-Sonekli gösterim yığın işlemleri için çok uygundur.

- Aritmetik İfadelerin değerlendirimi

Herhangi bir aritmetik ifade parantezden bağımsız olarak
Önekli veya Sonekli gösterimi ile değerlendirilebilir.

$$(3 * 4) + (5 * 6) \Rightarrow 3\ 4\ *\ 5\ 6\ *\ +$$



Örnek Saklayıcı Organizasyonları

Data Registers

D0	
D1	
D2	
D3	
D4	
D5	
D6	
D7	

Address Registers

A0	
A1	
A2	
A3	
A4	
A5	
A6	
A7	

Program Status

Program Counter

Status Register

(a) MC68000

General Registers

AX	Accumulator
BX	Base
CX	Count
DX	Data

Pointer & Index

SP	Stack Pointer
BP	Base Pointer
SI	Source Index
DI	Dest Index

Segment

CS	Code
DS	Data
SS	Stack
ES	Extra

Program Status

Instr Ptr
Flags

(b) 8086

General Registers

EAX	AX
EBX	BX
ECX	CX
EDX	DX

ESP	SP
EBP	BP
ESI	SI
EDI	DI

Program Status

FLAGS Register
Instruction Pointer

(c) 80386 - Pentium II

Komut Formatı

• Komut Alanları

- | | |
|-------------------------|--|
| İşlem Kod Alanı: | - İcra edilecek işlemi belirler. |
| Adres Alanı: | - Bellek adresini veya bir işlemci saklayıcı belirler. |
| Mod Alanı | - Adres alanının nasıl yorumlanacağını belirler
(efektif adres veya operand eldesinde) |

- Komut formatında yer alan adres alanları sayısı CPU iç organizasyonuna bağlıdır.

- En çok bilinen 3 adet CPU organizasyonu

Tek Akümülatör Organizasyonu:

Genel Register Organizasyonu :

ADD	R1, R2, R3	$\text{/* } R1 \leftarrow R2 + R3 \text{ */}$
ADD	R1, R2	$\text{/* } R1 \leftarrow R1 + R2 \text{ */}$
MOV	R1, R2	$\text{/* } R1 \leftarrow R2 \text{ */}$
ADD	R1, X	$\text{/* } R1 \leftarrow R1 + M[X] \text{ */}$

Yığın Organizasyonu:

PUSH X /* TOS ← M[X] */

APPENDIX

3 ve 2 Adresli Komutlar

- 3-Adresli Komutlar

$X = (A + B) * (C + D)$ hesaplamak için program :

ADD	R1, A, B	/* R1 $\leftarrow M[A] + M[B]$	*/
ADD	R2, C, D	/* R2 $\leftarrow M[C] + M[D]$	*/
MUL	X, R1, R2	/* M[X] $\leftarrow R1 * R2$	*/

- Kısa program yazılımı
- Komut bit uzunluğu büyük

- 2-Adresli Komutlar

$X = (A + B) * (C + D)$ hesaplamak için program :

MOV	R1, A	/* R1 $\leftarrow M[A]$	*/
ADD	R1, B	/* R1 $\leftarrow R1 + M[B]$	*/
MOV	R2, C	/* R2 $\leftarrow M[C]$	*/
ADD	R2, D	/* R2 $\leftarrow R2 + M[D]$	*/
MUL	R1, R2	/* R1 $\leftarrow R1 * R2$	*/
MOV	X, R1	/* M[X] $\leftarrow R1$	*/

1 ve 0 Adresli Komutlar

• 1-Adresli Komutlar

- Bütün data işlemleri için dolaylı AC register kullanılmıştır.
- $X = (A + B) * (C + D)$ hesaplamak için program:

LOAD	A	/* AC ← M[A] */
ADD	B	/* AC ← AC + M[B] */
STORE	T	/* M[T] ← AC */
LOAD	C	/* AC ← M[C] */
ADD	D	/* AC ← AC + M[D] */
MUL	T	/* AC ← AC * M[T] */
STORE	X	/* M[X] ← AC */

• 0-Adresli Komutlar

- Yığın organizasyonlu işlemcilerde olabilir
- $X = (A + B) * (C + D)$ hesaplamak için program:

PUSH	A	/* TOS ← A */
PUSH	B	/* TOS ← B */
ADD		/* TOS ← (A + B) */
PUSH	C	/* TOS ← C */
PUSH	D	/* TOS ← D */
ADD		/* TOS ← (C + D) */
MUL		/* TOS ← (C + D) * (A + B) */
POP	X	/* M[X] ← TOS */

Adresleme Modları



- ▶ Programın yürütülmesi sırasında datanın seçilme şekli, komutun adresleme moduna bağlıdır.
- ▶ Komutun adres alanını değiştirmek veya yorumlamak için bir kural belirler.
- ▶ Çok çeşitli adresleme modları bulunur:
 - ▶ Kullanıcıya programlama esnekliği verir.
 - ▶ Etkin olarak komutun adres alanındaki bitlerin kullanımını sağlar.

Adresleme Modları Türleri

- Direk Adresleme Modu (Implied)

Operand adresi komutun tanımlamasında doğrudan belirlidir.

- Komutta adres belirlemeye ihtiyaç yoktur.
- EA = AC, veya EA = Stack[SP]
- Örnekler: CLA, CME, INP

- İvedi Adresleme Modu (Immediate)

Operand adresini belirlemek yerine operandın kendisi belirlenir.

- Komutta adres belirlemeye ihtiyaç yoktur.
- Sadece operandın kendisi belirlenir.
- Bazen adresten daha fazla bit gerektirir.
- Bir operand işlemek için hızlı bir işlemidir.

Adresleme Modları Türleri

- Doğrudan Adresleme Modu (Direct Address)

Komut bellek adresini belirler

(belleğe erişim için doğrudan kullanılabilir)

- Diğer adresleme modlarından daha hızlı erişim

- Geniş bir fiziksel bellek alanını adresini belirlemek için çok fazla bit ihtiyacı bulunmaktadır.

- EA = IR(addr) (IR(addr): adres alanı, IR)

- Dolaylı Adresleme Modu (Indirect Address)

Bir komutun adres alanı operand adresini içeren bir bellek adresini taşıır.

- Kısıltılmış adres kullanılırsa, büyük fiziksel bellek daha az sayıda bit kullanarak adreslenebilir.

- İlave bellek erişimi nedeniyle, bir operandı işlemek yavaştır.

- EA = M[IR(adres)]

Adresleme Modları Türleri

- **Saklayıcı Adresleme Modu (Register Adr)**
Komutta belirlenen adres saklayıcı adresidir.
 - Saklayıcı içinde ihtiyaç olan operand tutulur.
 - Bellek adreslemeden daha kısa bir adresleme türüdür.
 - Daha hızlı operand işlenebilir. (Bellek Adreslemeye göre)
 - EA = IR(R) (IR(R): komut saklayıcı alanı)
- **Saklayıcı Dolaylı Adresleme Modu (Register Indirect)**
Komut operandın bellek adresini içeren bir saklayıcı belirler.
 - Daha az komut bit kullanımı
 - Saklayıcı Adresleme ve Bellek Adreslemeye göre bir operand işlemek için daha yavaştır.
 - EA = [IR(R)] ([x]: x içeriği)
- **Otomatik Artırımlı / Otomatik Azaltımlı Adresleme Modu
(Auto Increment / Auto Decrement Addressing Mode)**
 - Saklayıcı adresi belleğe erişim için kullanıldığında, saklayıcı değeri 1 artırılır veya 1 azaltılır.

Adresleme Modları Türleri

- **Bağıl Adresleme Modları**

- Bir komutun adres alanları, operandın adresini hesaplamak için belirli bir saklayıcı ile kullanılabilecek adresin kısmını belirler.
- Bir komutun adres alanı kısıdadır.
- Büyük fiziksel belleğe az sayıda adres bitleri ile erişilebilir.
- $EA = f(IR(adres), R)$,

R ye bağlı olarak 3 farklı bağıl adresleme modu;

- * **PC Bağıl Adresleme Modu ($R = PC$)**

- $EA = PC + IR(\text{adres})$

- * **Sıralı Adresleme Modu ($R = IX$, Burada IX: Indeks Register)**

- $EA = IX + IR(\text{adres})$

- * **Base Register Addressing Mode**

- $EA = BAR + IR(\text{address})$

($R = BAR$, burada BAR: Taban Adres Register)

Adresleme Modları -örnekler-

PC = 200
 R1 = 400
 XR = 100
 AC

Adresleme Modu	Efektif Adres	AC içerik
Ivedi Adr	-	/* AC \leftarrow 500 */ 500
Doğrudan Adr.	500	/* AC \leftarrow (500) */ 800
Dolaylı Adres.	800	/* AC \leftarrow ((500)) */ 300
Bağıl	702	/* AC \leftarrow (PC+500) */ 325
İndeks Adr.	600	/* AC \leftarrow (XR+500) */ 900
Saklayıcı	-	/* AC \leftarrow R1 */ 400
Saklayıcı Dolaylı	400	/* AC \leftarrow (R1) */ 700
Otom.Artırıım	400	/* AC \leftarrow (R1)+ */ 700
Otom.Azaltım	399	/* AC \leftarrow -(R1) */ 450

Adres	Bellek
200	Load AC Mod
201	Adress = 500
202	Bir Sonraki komut
399	450
400	700
500	800
600	900
702	325
800	300

Data Transfer Komutları

- Tipik Data Transfer Komutları

İsim	Sembol
Load	LD
Store	ST
Move	MOV
Exchange	XCH
Input	IN
Output	OUT
Push	PUSH
Pop	POP

- Farklı Adresleme Modlu Data Transfer Komutları

Mod	Assembler Gösterim	Register Transferi
Doğrudan	LD ADR	$AC \leftarrow M[ADR]$
Dolaylı	LD @ADR	$AC \leftarrow M[M[ADR]]$
Bağıl	LD \$ADR	$AC \leftarrow M[PC + ADR]$
İvedi	LD #NBR	$AC \leftarrow NBR$
İndeks	LD ADR(X)	$AC \leftarrow M[ADR + XR]$
Register	LD R1	$AC \leftarrow R1$
Register dolaylı	LD (R1)	$AC \leftarrow M[R1]$
Otom.artırımlı	LD (R1)+	$AC \leftarrow M[R1], R1 \leftarrow R1 + 1$
Otom,azaltım	LD -(R1)	$R1 \leftarrow R1 - 1, AC \leftarrow M[R1]$

Data İşleme Komutları

- 3 TEMEL KOMUT:
 - Aritmetik komutlar
 - Lojik ve bit işleme komutları
 - Öteleme komutları
- Aritmetik Komutlar

İsim	Sembol
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with Carry	ADDC
Subtract with Borrow	SUBB
Negate (2ye Tümleme)	NEG

- Lojik ve bit işleme komutları

İsim	Sembol
Clear	CLR
Complement	COM
AND	AND
OR	OR
Exclusive-OR	XOR
Clear carry	CLRC
Set carry	SETC
Complement carry	COMC
Enable interrupt	EI
Disable interrupt	DI

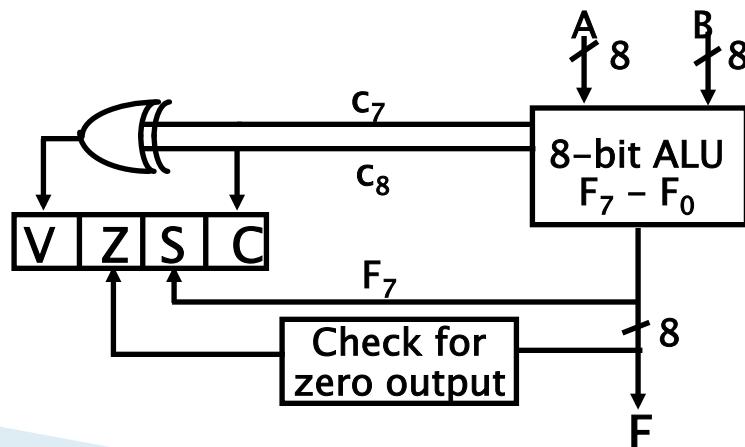
- Öteleme komutları

İsim	Sembol
Logical shift right	SHR
Logical shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right thru carry	RORC
Rotate left thru carry	ROLC

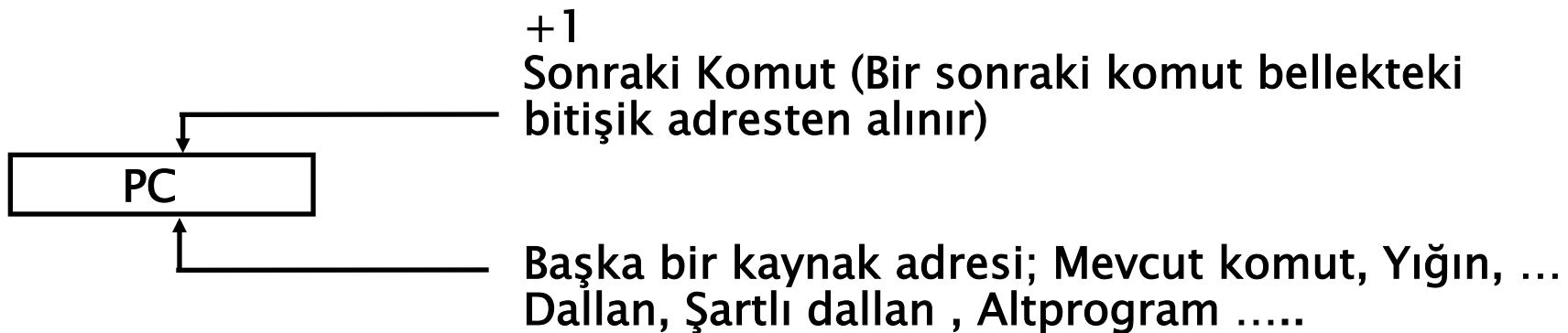
Bayraklar ve İşlemci Durum Kelimesi

- ▶ Temel Bilgisayarda, işlemcinin çeşitli (durum) bayrakları bulunur:
 - 1 bitlik bilgidir ve işlemcinin durumu hakkında bilgi verir.
 - E, FGI, FGO, I, IEN, R
- ▶ Bazı işlemcilerde, bayraklar sık sık bir saklayıcı içinde tutulur.
 - İşlemci durum saklayıcı (PSR); bazen bir durum kelimesi olarak da anılır (PSW).
- ▶ Bilinen bazı durum bitleri :
 - C (Elde-Carry): = 1 ALU elde çıkışı 1 ise,
 - S (İşaret - Sign): = 1 ALU çıkışının en anlamlı biti 1 ise,
 - Z (Sıfır - Zero): = 1 ALU çıkış içeriği 0 ise,
 - V (Taşma- Overflow): = 1 Taşma varsa.

Durum-Bayrak Devresi



Program Kontrol Komutları



- Program Kontrol Komutları

İsim	Sembol
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RTN
Compare(by -)	CMP
Test(by AND)	TST

* CMP ve TST komutları işlem sonuçlarını tutmazlar belirli bayrakları set/reset yaparlar.

Şartlı Dallanma Komutları

Sembol	Dallanma Şartı	Test Şartı
BZ	Branch if zero	$Z = 1$
BNZ	Branch if not zero	$Z = 0$
BC	Branch if carry	$C = 1$
BNC	Branch if no carry	$C = 0$
BP	Branch if plus	$S = 0$
BM	Branch if minus	$S = 1$
BV	Branch if overflow	$V = 1$
BNV	Branch if no overflow	$V = 0$
İşaretsiz Karşılaştırma Şartları ($A - B$)		
BHI	Branch if higher	$A > B$
BHE	Branch if higher or equal	$A \geq B$
BLO	Branch if lower	$A < B$
BLOE	Branch if lower or equal	$A \leq B$
BE	Branch if equal	$A = B$
BNE	Branch if not equal	$A \neq B$
İşaretli Karşılaştırma Şartları ($A - B$)		
BGT	Branch if greater than	$A > B$
BGE	Branch if greater or equal	$A \geq B$
BLT	Branch if less than	$A < B$
BLE	Branch if less or equal	$A \leq B$
BE	Branch if equal	$A = B$
BNE	Branch if not equal	$A \neq B$

Altprogram Çağırma ve Geri Dönüş

- **Subroutine Call** Altprogram çağrı - Call subroutine
Altprograma dallan - Jump to subroutine
Branch to subroutine
Dallan ve dönüş Adresini Sakla
Branch and save return address
- Altprogram başlangıcına dallanma
 - Dallan veya Şartlı Dallan komutları ile aynıdır.
 - * Çağırılan programa dönülünce kaldığı yerden icraya devam etmek için dönüş adresi tutulur.
- Dönüş adresini saklama

- Altprogramda sabit bir alanda (bellek)
- Bellekte sabit bir alan
- Bir işlemci register
- Bir bellek yiğini
 - en etkin yol

CALL
$SP \leftarrow SP - 1$
$M[SP] \leftarrow PC$
$PC \leftarrow EA$
RTN
$PC \leftarrow M[SP]$
$SP \leftarrow SP + 1$

Program Kesme Kavramı

Dış kesmeler (External interrupts)

CPU ve Bellek Dışından gelen dış kesme istekleri

- I/O Elemanları → Data transfer isteği veya Data transfer tamamlama
- Zamanlama Elemanı → Timeout
- Güç Problemi – Power Failure
- Operator

İç kesmeler (Internal interrupts– traps)

Mevcut halde çalışan programın neden olduğu kesmeler

- Saklayıcı Yığın Taşması – Sıfıra bölme – İşlem Kodu Hatası
- Koruma Hataları

Yazılımsal Kesmeler (Software Interrupts)

İç ve dış kesmeler bilgisayar donanımının sağladığı kesmelerdir.

Yazılım kesmeleri komut icrası ile başlatılır.

- Supervisor Call → Kullanıcı modundan bir yönetici moduna geçişini sağlar.
→ Kullanıcı modda izin verilmeyen belirli işlem sınıflarını icra etme iznini tanır.

Kesme Prosedürü

Kesme Prosedürü ve Altprogram Çağırma

- Yazılım kesmesi haricindeki diğer kesmelerde bir dış veya iç işaret ile kesme üretilir.
- Kesme Servis programı adresi bir komutun adres alanında ziyade donanımsal olarak belirlenir.
- Bir kesme prosedürü, genellikle CPU'nun durumunu tanımak için tüm gerekli bilgiyi saklar.

CPU'nun durumu şunlardan belirlenir:

PC'ın içeriği

Tüm işlemci saklayıcı içerikleri

Durum bitlerinin içerikleri

CPU mimarisine bağlı olarak CPU durumunu saklamanın yolları da değişmektedir.