

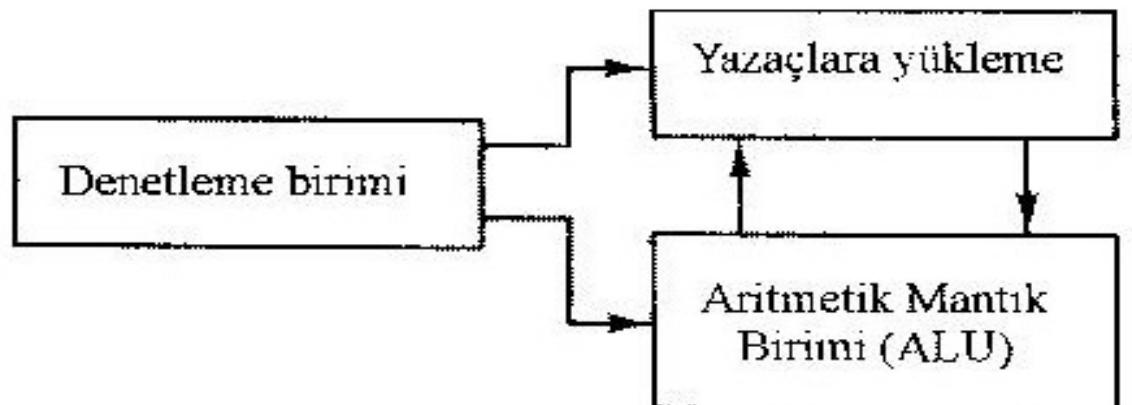
# BÖLÜM 8

# MERKEZİ İŞLEM BİRİMİ

GENEL İŞLEMLER ÜZERİNE  
CİRU

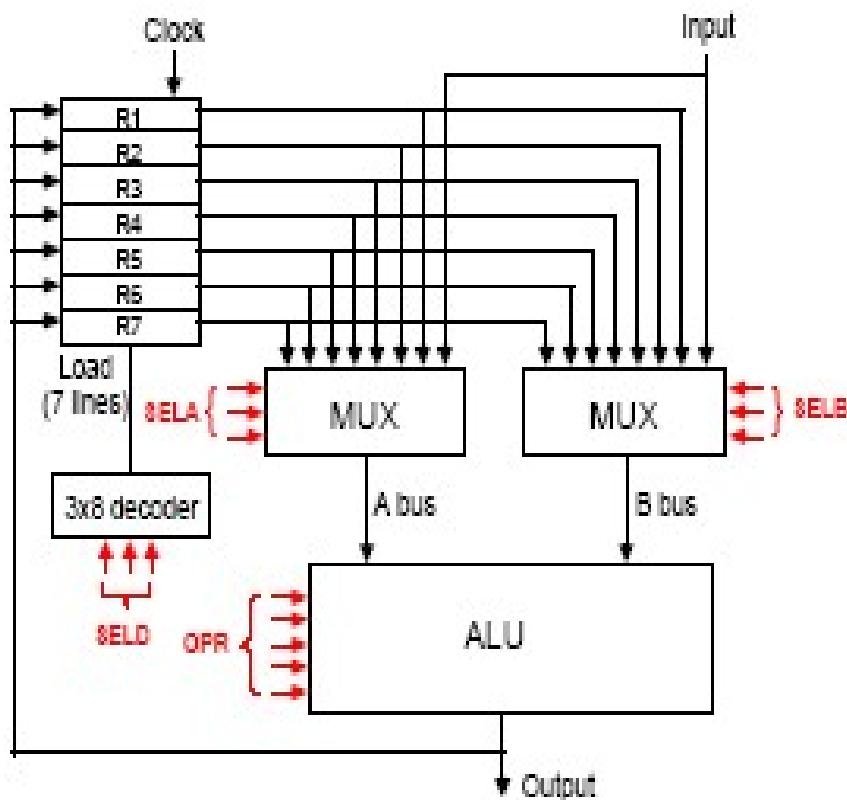
# Merkezi İşlem Birim (MİB)

- Veri işleme işlemlerini yerine getiren Merkezi işlem Birimi, bilgisayarların en önemli parçasıdır. Ve şekilde de görüldüğü gibi 3 ana kısımdan oluşur.
- Bilgisayar mimarisi, bilgisayar donanımı ve makine dili programlarının kullanıldığı programlardan oluşur.
- Denetim birimi, yazaçlardaki bilginin transferinden ve ALU buyruklarının denetiminden sorumludur.
- ALU, buyrukların çalışması için gerekli mikroişlemleri icra eder.
- Yazaçlar, buyrukların çalışması boyunca kullanılan ara veri depolama ünitesidir.



**Şekil 8.1** MİB nin başlıca elemanları

# Genel Yazaç Kurulumu



- Geçici işlemlerin, dönüş adreslerinin, kısmi çarpımların bellekte saklanması ve tekrar okunması uzun zaman alacağından, bu işlemler için MİB'de yazaçlarda yapılması daha uygundur. Yani yazaçların veri aktarımında değil, mikro işlemlerin icrasında da kullanılması daha uygundur.
- Bunun için düzenlenmiş yapı ve denetim kelimesi, şekillerde görülmektedir.

Control Word



MİB'i çalıştırın denetim birimi, bilgileri değişik bileşenler yardımıyla ALU'ya gönderir. ALU'da işlenen işlem gerekli yere yazılır. Örnek olarak;

$$R_1 \leftarrow R_1 + R_2$$

İşlemini ele alalım. Burada denetim birimi bunun için binary seçim değişkenlerini bulmalı ve aşağıdaki seçim girişlerine uygulamalıdır.

- 1- MUX A seçici (SEL<sub>A</sub>): R<sub>2</sub>'nin içeriğini A veriyoluna
- 2- MUX B seçici (SEL<sub>B</sub>): R<sub>3</sub>'ün içeriğini B veriyoluna
- 3- ALU işlem seçici (OPR) : A+B işlemi için.
- 4- Kod çözücü ile hedef yazaç seçici (SEL<sub>D</sub>): Toplamı R<sub>1</sub>'e akt.

Bu dört denetim fonksiyonu aynı anda üretilmeli, bir clock pulsında hazır olmalıdır.

# DENETİM KELİMESİ

- Denetim kelimesi 14 bit 4 bölümden oluşur. 3'er bitlik bölümler, ALU'nun girişlerini ve ALU'dan çıkan bilginin yazılacağı yeri seçer. 5 bitlik OPR kısmı ise ALU'da yapılacak işlemleri seçer.

**Çizelge 8.1** Yazaç seçim alanlarının kodlanması

İkili kod	SELA	SELB	SELD
000	Giriş	Giriş	Bir şey yok
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 İşlemlerinin Açıklaması

**Çizeğe 8.2 ALU işlemlerinin açıklanması**

OPR seçimi	İşlem	Sembol
00000	$A$ nın aktarımı	TSFA
00001	$A$ yı 1 artırma	INCA
00010	$A + B$ işlemi	ADD
00101	$A - B$ işlemi	SUB
00110	$A$ yı 1 azaltma	DECA
01000	$A$ ve $B$ yi VE ile yi VEYA la	AND
01010	$y$ i ÖZEL-VEYA la	OR
01100	$y$ i ÖZEL-VEYA la	XOR
01110	$A$ nın tümleyenini al	COMA
10000	$A$ yı sağa kaydır	SHRA
11000	$A$ yı sola kaydır	SHLA

ALU işlemleri için kod çözümü MİB için yukarıda tanımlanmıştır. OPR alanı 5 bitten oluşmakta ve her bir işlemin sembolik adı bulunmaktadır.

# Mikroişlem Örnekleri

14 bitlik denetim kelimesinde MİB için bir mikroişlem vardır. Mikroişlem için bir denetim kelimesi seçim girişlerinden elde edilir. Örneğin

$$R_1 \leftarrow R_2 - R_3$$

Çıkarma işlemi için ALU'nun A girişi  $R_2$ , B girişi  $R_3$ , hedef yazaç'ta  $R_1$  dir. ALU'nun işlemi A-B dir.

Çıkarma işlemi için denetim kelimesi;

ALAN:	SELA	SELB	SELD	OPR
SEMBOL:	$R_2$	$R_3$	$R_1$	SUB
DENETİM KELİMESİ	010	011	001	00101

# MİB Mikroişlem Örnekleri

Çizelge 8.3 MİB mikro işlem örnekleri

Mikro İşlem	Sembolik Gösterim				Denetim Kelimesi
	SELA	SELB	SELD	OPR	
$R1 \leftarrow R2 - R3$	$R2$	$R3$	$R1$	SUB	010 011 001 00101
$R4 \leftarrow R4 \vee R5$	$R4$	$R5$	$R4$	OR	100 101 100 01010
$R6 \leftarrow R6 + 1$	$R6$	—	$R6$	INCA	110 000 110 00001
$R7 \leftarrow R1$	$R1$	—	$R7$	TSFA	001 000 111 00000
Çıkış $\leftarrow R2$	$R2$	—	None	TSFA	010 000 000 00000
Çıkış $\leftarrow$ Giriş	giriş	—	None	TSFA	000 000 000 00000
$R4 \leftarrow ShlR4$	$R4$	—	$R4$	SHLA	100 000 100 11000
$R5 \leftarrow 0$	$R5$	$R5$	$R5$	XOR	101 101 101 01100

- Arttırma ve transfer işlemleri ALU'nun B girişini kullanmaz. Bu durum – ile gösterilir. Birçok mikroişlemde MİB içinde gerçekleştirilebilir. Örneğin; x(XOR)x işlemi ile A'yı sıfırlamak mümkündür.
- Denetim kelimelerinin depolandığı birime denetim belleği denir. Bellekten denetim kelimelerinin ardarda okunmasıyla MİB'deki mikroişlemlerin sırayla gerçekleşmesi mümkün olur.

# YİĞİT KURULUMU

- MİB içindeki çok faydalı bir kurulumdayıgit veya LIFO (Least input First Output)'dur.
- Yığıt bir bellek parçasıdır. Son depolanan bilgi ilk geri dönen bilgi olur.Yığittaki en son depolanan bilgi en üsttedir.
- Say.Bilgisayarlarda yığıt temel bir bellek birimidir.Bir adres yazacı vardır. Bu sayıç sadece sayabilir. Bunlara yığıt göstergesi denir. Yığittaki son adresi gösterir.
- Yığittaki yazaçlar okuma ve yazmaya devamlı hazırlıdır.
- Yığıtla ilgili işlem ekleme ve silmedir. Ekleme işlemi itme, tersi ise silmedir.

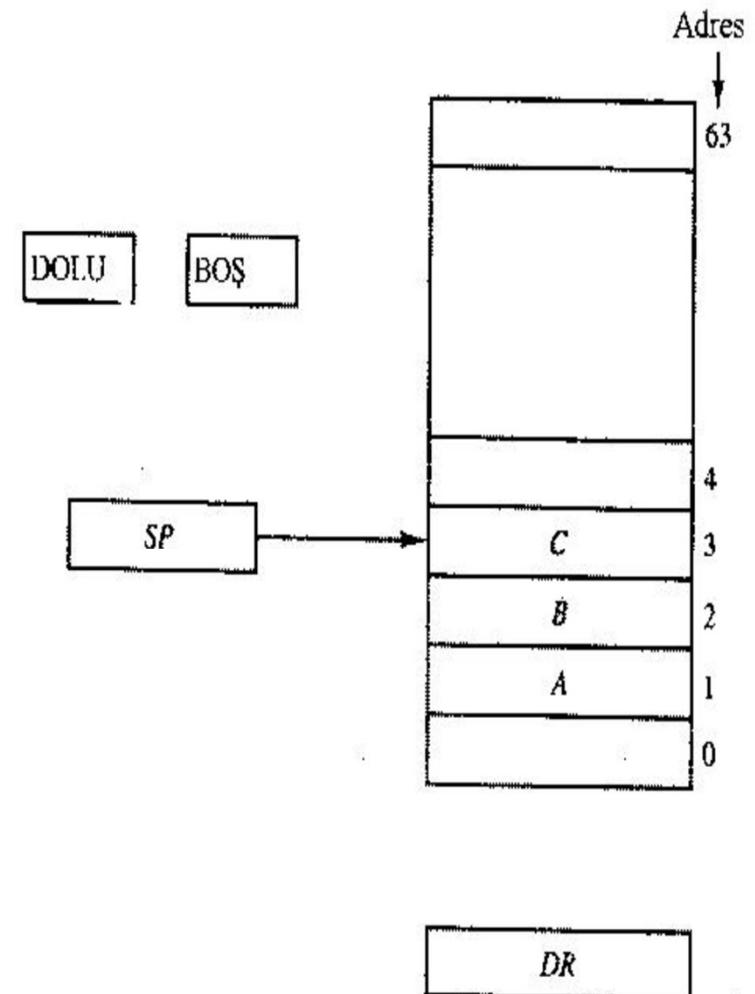
# YAZAÇ YIĞITI

Bir yiğit, belleğin bir bölümünde yer alır, veya yiğit sonlu sayıda bellek kelimeleri veya yazaçlardan düzenlenir. S.8.3 bir 64 kelimelik yazaçyiğit kurulumudur. SP yiğit göstergesidir ve yiğitin en üstündeki Kelimenin adresinin binary değerini barındırır.

Su anda yiğitta 3 bilgi var ve en tepedeki C'dir. SP'deki adres C'nin adresidir.

En üstteki bilgiyi silmek için; 3 adresindeki bilgi okunur ve SP'nin içeriği 1 azaltılır. Bu durumda B (2 adresi) yiğitin en üstündedir.

Yığita yeni bir şey yazmak için, SP bir arttırılır ve yeni adrese yeni bilgi yazılır. Yani C'ye yeni bilgi yazılır. Dolayısıyla C'de daha önce okunmuş olan bilgi silinmiş olur.



Şekil 8.3 Bir 64 kelimelik yiğitin blok şeması

# Yazaç Yığıtı-1

- 64 kelimelik yığıt için SP 6 bit'tir. değeri 63 ten büyük olamaz. 63 bir arttırılırsa ( $111111+1$ ) SP (1)oooooo yı yani başlangıç adresini gösterir. Benzer olarak bu durum  $000000-1=111111$  ile de 63'ü elde edriz.
- 1 bitlik DOLU FF'u yığıt dolu ise 1 yapılır. Eğer yazaç boş ise 1 bitlik BOŞ Ff'u 1 yapılır.
- Veri yazacı DR yığıta yazılan veya yığittan okunan bilgileri tutar.
- Başlangıçta SP o'lanır. BOŞ=1, DOLU=o yapılır. (Bu durum yazaç göstergesi bomboş işlemidir)

## **Yazaç Yığıtı-2**

**İtme (yığıta yazma) işleminin mikroişlemleri;**

$SP \leftarrow SP + 1$        $SP$ 'yi 1 arttır

$M[SP] \leftarrow DR$        $DR$ 'deki bilgiyi yığıtin en üstüne yaz

$If(SP=0)$  then ( $DOLU \leftarrow 1$ ) yığıtin doluluğunu kontrol et.

$BOŞ \leftarrow 0$       Yığıtin boş olmadığını işaret et.

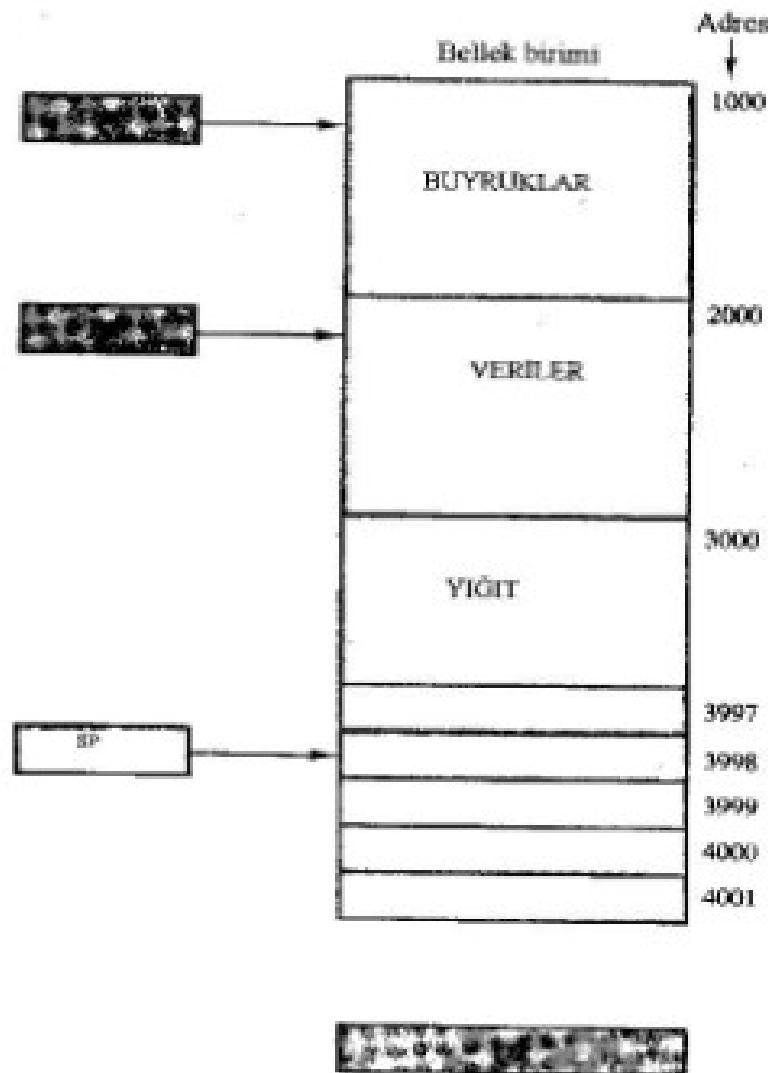
**Çekme (yığıtdan okuma) işleminin mikroişlemleri;**

$DR \leftarrow M[SP]$       Yığıttan bilgiyi oku

$SP \leftarrow SP - 1$        $SP$ 'yi 1 azalt

$If(SP=0)$  then ( $BOŞ \leftarrow 1$ ) yığıtin boşluluğunu kontrol et.

$DOLU \leftarrow 0$       Yığıtin dolu olmadığını işaret et.



Şekil 8.4 Program, veri ve yiğit kesimleri ile bilgisayar hafızası

# Zıt Polonyalı Gösterimi

- Zıt polonyalı gösteriminde karşılık gelen art ek gösterimde verilerden sonra operatörler yer alır . Üç gösterim için örnek verilirse:

$A + B$  Matematikte

$+ AB$  Ön ek veya polish gösterimi

$AB +$  Art ek veya zıt polonyalı gösterimi

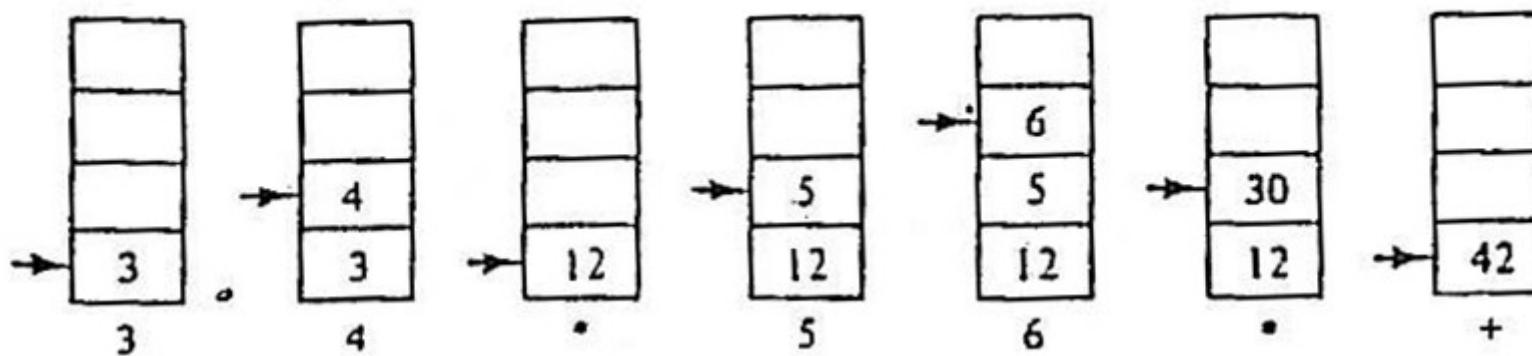
Zıt polonyalı yöntemi, yiğit işletimi için uygun bir yöntemdir.

$$A * B + C * D$$

İfadesi zıt polonyalı gösteriminde

$$AB * CD * +$$

## Aritmetik ifadelerin Degerlendirilmesi



Şekil 8.5  $3 * 4 + 5 * 6$  için yığıt işlemleri

Aşağıdaki sayısal örneğin bu işlemleri açıklamaktadır.

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

Zıt polonyalı yönteminde bu ifade: **34\*56\*+**

# Buyruk Biçimleri

- Bir buyrugun biçimi dikdörtgen içinde verilir. Dikdörtgen buyrugun bitlerini gösterir. Bellek kelimesi ve denetim yazacı böyle gösterilir. Bir buyrugun bitleri alanlar adı verilen gruptara ayrıılır. Bir buyruk biçiminde bulunan alanlar;
  1. İcra edilecek işlemi gösteren OPR kod alanı
  2. Bir adres alanı: bellek alanını veya bir işlemci yazacını gösterir.
  3. Bir kip alanı: verini yolunu veya etkin adresin bulunduğu gösterir.

## Üç Adresli Buyruklar

Üç adres buyruklu bilgisayarlarda bu adresler bellek adresleri veya işlemci yazacık adresleri olabilir.  $X = (A + B) * (C + D)$  yi yapan birleştirici dil programı.

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

# Buyruk Biçimleri

## İki Adresli Buyruklar

Ticari bilgisayarlarda en çok bunlar vardır. Adresler bellek veya işlemci yazacı adresi olabilir.  $X = (A + B) * (C + D)$  işlemini yapan birleştirici dil programı.

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	$M[X] \leftarrow R1$

Birinci adresin hem kaynak, hem de hedef yazacı olduğu varsayılıyor.

# Buyruk Biçimleri

## Bir Adresli Buyruklar

Bir adresli buyruklar tüm veri işlemleri için  $AC$  yi kullanırlar. Çarpma ve bölme işlemleri için ikinci bir yazaca ihtiyaç vardır. Buna rağmen ikinci bir yazacık ihmal edilip bütün işlemler  $AC$  de yapılır ve sonuçlar  $AC$  de saklanır.

LOAD	A	$AC \leftarrow M[A]$
ADD	B	$AC \leftarrow A + M[B]$
STORE	T	$M[T] \leftarrow AC$
LOAD	C	$AC \leftarrow M[C]$
ADD	D	$AC \leftarrow AC + M[D]$
MUL	T	$AC \leftarrow AC * M[T]$
STORE	X	$M[X] \leftarrow AC$

Bütün işlemler  $AC$  yazacı ile bellekteki veriler arasındadır.  $T$  bellek adresi ara değerinin saklanması içindir.

# Buyruk Biçimleri

## Sıfır Adresli Buyruklar

Yığıt kurulumlu bir bilgisayarda buyrukların adrese gereksinimi yoktur. ADD, MUL, PUSH ve POP adresizdir, fakat yığıtla bellek haberleşmesi için bellek adreslerine gerek vardır. ( $TOS = \text{yığıt göstergesi}$ ).  $X = (A + B) * (C + D)$  işleminin yığıt kurulumlu bilgisayarda birleştirici dil programı.

PUSH	A	$TOS \leftarrow A$
PUSH	B	$TOS \leftarrow B$
ADD		$TOS \leftarrow (A + B)$
PUSH	C	$TOS \leftarrow C$
PUSH	D	$TOS \leftarrow D$
ADD		$TOS \leftarrow (C + D)$
MUL		$TOS \leftarrow (C + D) * (A + B)$
POP	X	$M[X] \leftarrow TOS$

Aritmetik işlemlerin yığıt kurulumlu bir bilgisayarda yapılabilmesi için Polish gösterimine çevrilmesi gerekir.

# Buyruk Biçimleri

## RISC Buyrukları

RISC makinesinin yararları 8.8 de anlatılacaktır. Tipik bir RISC buyruk kümlesi, hellekle MİB arasındaki aktarım için LOAD ve STORE buyrukları ile kısıtlıdır. Diğer bütün buyruklar MİB içinde icra edilir. Yani buyruklar içinde bellek adreslemeli buyruk yoktur. RISC için yazılmış bir programda LOAD ve STORE buyrukları bir bellek ve bir yazaç adresi içerir. Hesaplama tipi buyruklar ise 3 adresli buyruk olup, adreslerin hepsi yazaç adresleridir. Aşağıdaki program  $X = (A + B) * (C + D)$  yi bulur.

LOAD	R1, A	R1 $\leftarrow M[A]$
LOAD	R2, B	R2 $\leftarrow M[B]$
LOAD	R3, C	R3 $\leftarrow M[C]$
LOAD	R4, D	R4 $\leftarrow M[D]$
ADD	R1, R1, R2	R1 $\leftarrow R1 + R2$
ADD	R3, R3, R4	R3 $\leftarrow R3 + R4$
MUL	R1, R1, R3	R1 $\leftarrow R1 * R3$
STORE	X, R1	$M[X] \leftarrow R1$

Yükleme buyruğu verileri bellekten MİB yazaçlarına aktarır. Toplama ve çarpma buyrukları veriler üzerinde belirtilen matematiksel işlemleri yapar ve depolama buyruğu hesaplanan sonucu hafızada saklama yazacına aktarır.

# ADRESLEME KİPLERİ İÇİN SAYISAL ÖRNEKLER

PC = 200  
RI = 400  
XR = 100  
AC

Adres	Bellek
200	AC'ye yönlendirme Mod
201	Adres=500
202	Sonraki Buyruk
399	450
400	700
500	800
600	900
702	325
800	300

Şeldil 8.7 Adresleme kipleri için sayısal örnek

# NUMERİK ÖRNEK LİSTESİ

**Çizelge 8.4 Nümerik örnek listesi**

Adres kipi	Etkin adres	AC nın içeriği
Düzenli adres	500	800
Derhal veri	201	500
Dolaylı adres	800	300
Göreceli adres	702	325
İndirimmiş adres	600	900
Yazaç	—	400
Yazaç dolaylı	400	700
Otomatik artma	400	700
Otomatik azalma	399	450

# VERİ AKTARIM BUYRUKLARI

Çizelge 8.5 Veri aktarımı buyrukları

İsim	Birleşirici dil sembollerı
Yükle	LD
Aktar	ST
Taşı	MOV
Yer değiştir	XCH
Giriş	IN
Cıkış	OUT
İtme	PUSH
Çekme	POP

- Yükle buyruğu genellikle bellekteki işlemci yazacına (genellikle AC ye) aktarımı için kullanılır. Aktar buyruğu işlemci yazacından belleğe aktarımı için kullanılır. Taşı buyruğu bir yazaçtan başka yazaca aktarımı sağlar. Bu buyruk aynı zamanda bellek ile MIB yazacı arasındaki veya iki bellek kelimesi arasındaki aktarımı içinde kullanılabilir.

Yer değiştir buyruğu iki yazaç veya bir yazaç ile bir bellek kelimesinin içeriklerini yer değiştirmiştir. Giriş ve çıkış buyrukları işlemci yazacı ile giriş veya çıkış uçları arasında veri aktarımı yapar. İtme ve çekme buyrukları işlemci yazacı ile bellek vienisi arasında veri aktarımı yapar.

# YÜKLEME BUYRUK İÇİN SEKİZ ADRESLEME KİPİ

Çizelge 8.6 Yüklemeye buyruk için sekiz adresleme kipi

Kip	Kabul edilen hırleştirmeli dil	Yazac aktarımı
Doğrudan adres	LD ADR	$AC \leftarrow M[ADR]$
Dolaylı adres	LD @ADR	$AC \leftarrow M[M[ADR]]$
Göreceli adres	LD SADR	$AC \leftarrow M[PC + ADR]$
Derhal Veri	LD #NBR	$AC \leftarrow NBR$
İndislenmiş adres	LD ADR(X)	$AC \leftarrow M[ADR + XR]$
Yazac	LD R1	$AC \leftarrow R1$
Yazac dolaylı	LD (R1)	$AC \leftarrow M[R1]$
Otomatik artma	LD (R1)+	$AC \leftarrow M[R1], R1 \leftarrow R1 + 1$

Çizelgede gerçekten meydana gelen aktarımalar da yazılmıştır. *ADR* ile adres gösterilmiş, *NBR* yerinin numarası, *X* indis yazacı, *R1* işlemci yazıcı, *AC* işlemci yazıcısıdır. @ karakteri dolaylı adresstir. \$ işaretli adresi *PC* ye göre göreceli yapar. Veriden önce gelen # işaretti doğrudan kipi gösterir. İndisli adresleme kipi, sembolik adresden sonra parantez içine yazılan yazac ile belliidir. Yazac kipi, yazacın adının yazılmasıyla anlaşıılır. Dolaylı yazac kipinde yazac adı parantez içine yazılır. Otomatik artma kipi de parantez içinde yazac adının yanına yazılıp + işaretinden anlaşıılır.

# TİPİK ARİTMETİK BUYRUKLAR

**Çizelge 8.7 Tipik aritmetik buyruklar**

<u>İsim</u>	<u>Birleştirici dil sembollerı</u>
Aşırı	INC
Azalt	DEC
Topla	ADD
Çıkar	SUB
Carp	MUL
Böl	DIV
Elide ile topla	ADD C
Borc ile çıkar	SUB B
2 ye göre türmeyen at	NEG

# MANTIKSAL VE BİT İŞLEME BUYUKALARI

**Çizelge 8.8 Mantıksal ve bit işleme buyrukları**

<u>İsim</u>	<u>Birleştirici dil sembolleri</u>
Silme	CLR
Tümleme	COM
VE	AND
- VEYA	OR
ÖZEL-VEYA	XOR
Elde yi silme	CLRC
Elde yi l yapma	SETC
Elde yi ümleme	COMC
Kesmeyi açma	EI
Kesmeyi kapatma	DI

# KAYDIRMA BUYRUKLARI

Çizelge 8.9 Kaydırma buyrukları

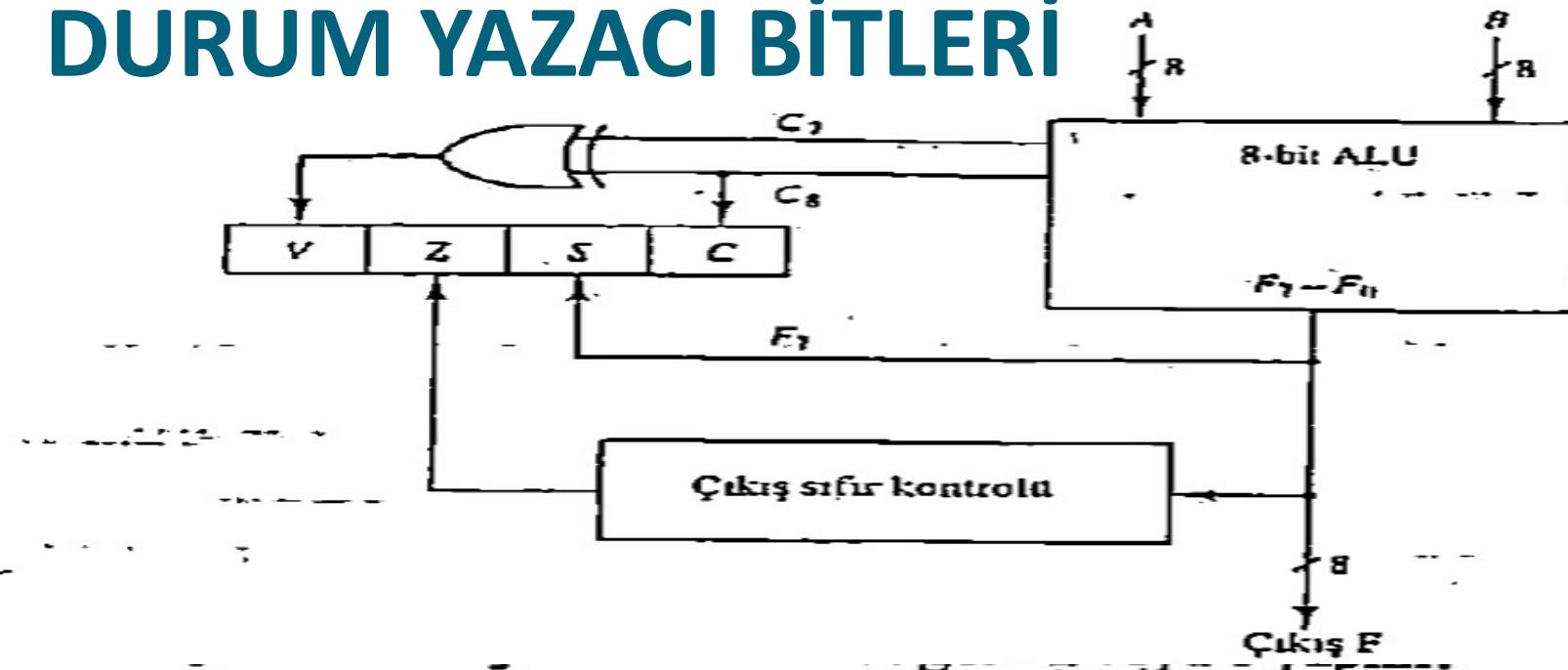
İsim	Birleştirici dil sembollerı
Manüksal sağa kaydırma	SHR
Mantıksal sola kaydırma	SHL
Aritmetik sağa kaydırma	SHRA
Aritmetik sola kaydırma	SHLA
Dairesel sağa kaydırma	ROR
Dairesel sola ka ydirmia	RÖL
Elde ile dairesel sağa kaydırma	RORC
Elde ile dairesel sola kaydırma	ROLC

# PROGRAM DENETİM BUYRUKLARI

**Çizeğe 8.10 - Program denetim buyrukları**

İsim	Birleştirici dil sembollerini
Dallan	BR
Aşla	JMP
Aşla	.SKP .
Çağır	CALL.
Geri dön	RET
Çıkarma ile kıyaslama	CMP
VE işlemi ile kontrol	TST

# DURUM YAZACI BİTLERİ



1.  $C(\text{elde}) = 1$  yapılır. Eğer  $C_8$  yani son elde 1 ise. Eğer  $C_8 = 0$  ise  $C = 0$  yapılır.
2. Eğer en yüksek mertebeli bit  $F_7 = 1$  ise  $S$  ( işaret ) biti 1 yapılır.  $F_7 = 0$  ise  $S = 0$  dır.
3. ALU nun bütün bitleri 0 ise (ALU nua bütün çıkışları = 0)  $Z = 1$  yapılır. Eğer çıkış sıfır değil ise ve  $Z = 0$  dır.
4. Son iki elde nin ÖZEL-VEYA sı 1 ise  $V(\text{taşma}) = 1$  yapılır. Yani taşıma vardır. Aksi halde 0 dır. Eğer negatif sayılar  $2^n$  nin tümleyenini biçiminde verilmişse bu taşıma şartıdır (Bölüm 3.3). 8 bit ALU için çıkış +127 den büyük veya -128 den küçük olunca  $V = 1$  olur.

**Çizelge 8.11 Şartlı dallanma buyrukları**

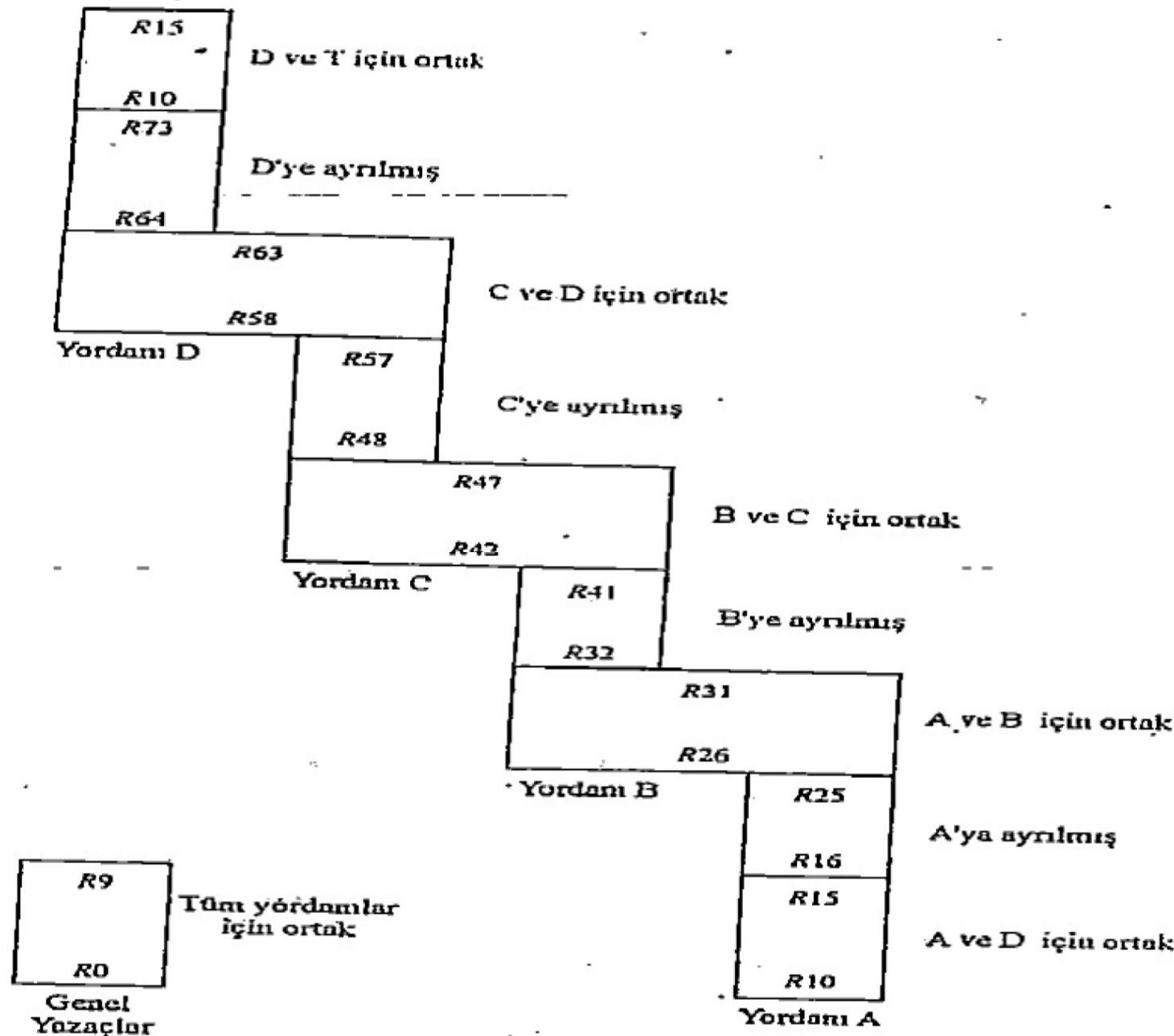
Birleştirici dil sembollerleri	Dallanma Şartları	Tesi Şartları
BZ	sıfır ise dallan	$Z = 1$
BNC	sıfır değilse dallan	$Z \neq 0$
BC	elde varsa dallan	$C = 1$
BNC	elde yoksa dallan	$C = 0$
BP	sayı pozitifse dallan	$S = 0$
BM	sayı negatifse dallan	$S = 1$
BV	ışıkta varsa dallan	$V = 1$
BNV	ışıkta yoksa dallan	$V = 0$

**İşaretsiz karşılaştırma şartları ( $A - B$ )**

BHT	yüksek ise dallan	$A > B$
BHE	yüksek veya eşit ise dallan	$A \geq B$
BLO	düşük ise dallan	$A < B$
BLOE	düşük veya eşit ise dallan	$A \leq B$
BE	eşit ise dallan	$A = B$
BNE	eşit değil ise dallan	$A \neq B$

**İşaretli karşılaştırma şartları ( $A - B$ )**

BGT	büyük ise dallan	$A > B$
BGE	büyük veya eşit ise dallan	$A \geq B$
BLT	küçük ise dallan	$A < B$
BLE	küçük veya eşit ise dallan	$A \leq B$
BE	eşit ise dallan	$A = B$
BNE	eşit değil ise dallan	$A \neq B$



Şekil 8.9 Üst üste binen yazac pencereleri

# BERKELEY RICK-I

31	24 23	19 18	14 13	12	5 4	0
İşlem Kodu	Rd	Rs	0	Kullanılmışınır	S2	
8	5	5	1	8	5	

(a) Yazıcı Mod (S2 yazıcı belirtir)

31	24 23	19 18	14 13	12	0
İşlem Kodu	Rd	Rs	1		S2
8	5	5	1		13

(b) Yazıcı-derhal Mod (S2 işlem belirtir).

31	24 23	19 18	0
İşlem Kodu	COND		Y
8	5		19

(c) PC İlgili Mod

## BERKELEY RICK-I

ADD R22, R21, R23       $R23 \leftarrow R22 + R21$

ADD R22, #150, R23       $R23 \leftarrow R22 + 150$

ADD R0, R21, R22       $R22 \leftarrow R21$  (taşıma)

ADD R0, #150, R22       $R22 \leftarrow 150$  (veri aktarımı)

ADD R2, #1, R22       $R22 \leftarrow R22+1$  (arttırma)

muşlardır. Veri işleme buyrukları, aritmetik, mantıksal ve kaydutma işlemleri için yapar. İşlem kodu ve veri sütunlarındaki kodlar birleştirici dil programı yazarken kullanılır. Yazıcı transfer ve tanımlama sütunları buyruğu yazıcı aktarım diliyle açıklıyor. Bütün buyruklar 3 adreslidir. S2 bir yazıcı veya derhal veri olarak kullanılabilir. Bu olay # işaretti ile gösterilmiştir. ADD buyruğu ile bu olayın nasıl olduğuna bakılırsa

**Çizelge 8.12 Berkeley RISC I buyruk kümesi**

İşlem kodu	Veriler	Yazılım akışları	Tanımlama
<b>Veri İşleme Buyrukları</b>			
ADD	Rs, S2, Rd	$Rd \leftarrow Rs + S2$	tamsayı toplama
ADDC	Rs, S2, Rd	$Rd \leftarrow Rs + S2 + \text{elde}$	elde ile toplama
SUB	Rs, S2, Rd	$Rd \leftarrow Rs - S2$	tamsayı çıkarma
SUBC	Rs, S2, Rd	$Rd \leftarrow Rs - S2 - \text{elde}$	elde ile çıkarma
SUBR	Rs, S2, Rd	$Rd \leftarrow S2 - Rs$	yer değiştirilmiş çıkışma
SUBCR	Rs, S2, Rd	$Rd \leftarrow S2 - Rs - \text{elde}$	elde ile çıkışma
AND	Rs, S2, Rd	$Rd \leftarrow Rs \wedge S2$	VE
OR	Rs, S2, Rd	$Rd \leftarrow Rs \vee S2$	VEYA
XOR	Rs, S2, Rd	$Rd \leftarrow Rs \oplus S2$	ÖZEL-VEYA
SLL	Rs, S2, Rd	$Rd \leftarrow Rs S2 \text{ ile kaydırma}$	sola kaydırma
ŠRL	Rs, S2, Rd	$Rd \leftarrow Rs S2 \text{ ile kaydırma}$	mantıksal sağa kaydırma
SRA	Rs, S2, Rd	$Rd \leftarrow Rs S2 \text{ ile kaydırma}$	aritmetik sağa kaydırma
<b>Veri Aktarım Buyrukları</b>			
LDL	(Rs)S2, Rd	$Rd \leftarrow M[Rs + S2]$	uzun kelime yükle
LDSU	(Rs)S2, Rd	$Rd \leftarrow M[Rs + S2]$	kısa işaretsız kelime yükle
LDBU	(Rs)S2, Rd	$Rd \leftarrow M[Rs + S2]$	kısa işaretli kelime yükle
LDBS	(Rs)S2, Rd	$Rd \leftarrow M[Rs + S2]$	İşaretsız byte yükle
LDHI	Rd, Y	$Rd \leftarrow Y$	derhal yüksek veri yükle
STL	Rd, (Rs)S2	$M[Rs + S2] \leftarrow Rd$	uzun kelime sakla
STS	Rd, (Rs)S2	$M[Rs + S2] \leftarrow Rd$	kısa kelime sakla
STB	Rd, (Rs)S2	$M[Rs + S2] \leftarrow Rd$	byte sakla
GETPSW	Rd	$Rd \leftarrow PSW$	durum kelimesini yükle
PUTPSW	Rd	$PSW \leftarrow Rd$	durum kelimesini sakla
<b>Program Denetim Buyrukları</b>			
JMP	COND, S2(Rs)	$PC \leftarrow Rs + S2$	şartlı atlama
JMPR	COND, Y	$PC \leftarrow PC + Y$	göreceli atlama
CALL	Rd, S2(Rs)	$Rd \leftarrow PC$ $PC \leftarrow Rs + S2$	alt program çağır ve
CALLR	Rd, Y	$CWP \leftarrow CWP - 1$ $Rd \leftarrow PC$ $PC \leftarrow PC + Y$	pencereyi değiştir göreceli çağır ve
RET	Rd, S2	$CWP \leftarrow CWP - 1$ $PC \leftarrow Rs + S2$	pencereyi değiştir geri dön ve
CALLINT	Rd	$CWP \leftarrow CWP - 1$ $Rd \leftarrow PC$	pencereyi değiştir kesmeleri kapat
RETINT	Rd, S2	$CWP \leftarrow CWP - 1$ $PC \leftarrow Rd + S2$	kesmeleri aç
GTLPC	:	$CWP \leftarrow CWP - 1$ $Rd \leftarrow PC$	