

## Ek B: RISC (Reduced Instruction Set Computer) İşlemciler:

### RISC Özellikleri:

- Daha az sayıda komut vardır, komutların işlevleri basittir.
- Daha az sayıda , basit adresleme kipi
- Sabit uzunlukta komut yapısı (komut çözme işi kolaydır)
- Doğrudan bellek üzerinde işlem yapan komutlar yoktur, işlemler iç saklayıcılarda yapılır.
- Belleğe sadece yazma/okuma işlemleri için erişilir (*load-store architecture*).
- Tek çevrimde alınıp yürütülebilen komutlar (komut işhattı (*pipeline*) sayesinde)
- Devrelendirilmiş (*hardwired*) denetim birimi.

### Diğer Özellikler:

Aşağıdaki özelliklerin bazıları tüm RISC'lerde bulunmayabilir, bazıları ise CISC MIB'lerde de bulunabilir. Ancak bunlar RISC'ler için özellikle önemlidir.

- Çok sayıda saklayıcı (*register file*)
- Kesişimli (*overlapped register window*) saklayıcı penceresi
- Komutlar için optimize edilebilen iş hattı
- Harvard mimarisi
- Derleyici desteği

### Binişimli (Kesişimli) Saklayıcı Penceresi (*Overlapped Register Windows*):

Bu yapı, alt program çağrılarında yığına (bellek erişimine) gerek duymadan

- parametre aktarımını sağlamak ve
- yerel değişkenleri tutmak için kullanılır.

İşlemcinin çok sayıda saklayıcısı olmasına rağmen programcı belli bir anda bunlardan sadece belli bir adetini kullanabilir.

Bir anda kullanılabilen saklayıcıların oluşturduğu gruba **pencere** (*window*) denir.

Alt programa gidildikçe (ve geri döndükçe) pencere değişir.

Böylece programcı farklı saklayıcılara erişir.

İki pencere arasındaki ortak saklayıcılar parametre aktarımı için, ortak olmayanlar ise alt programların yerel değişkenleri için kullanılırlar.

Bir pencerede n saklayıcı varsa programlar yazılırken sadece R0 ve Rn-1 numaraları kullanılır.

Ancak pencere değiştikçe bu numaralar farklı fiziksel saklayıcılara denk düşerler.

Tüm RISC işlemciler bu yapıyı kullanmaz. Örneğin MIPS işlemcisinde yoktur.

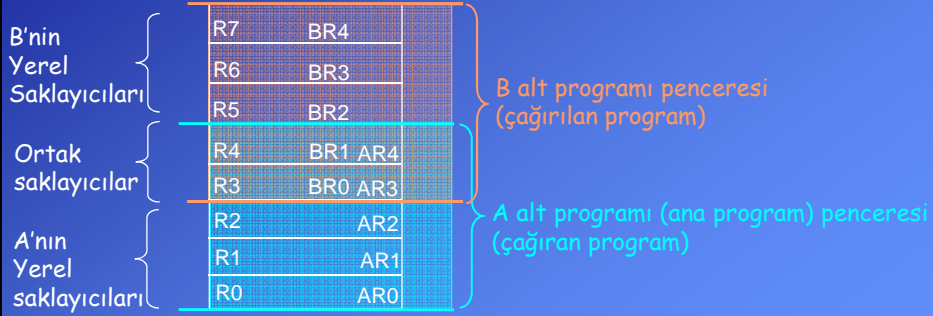
**Örnek:**

Aşağıdaki örnekte işlemcinin 8 adet saklayıcısı vardır. Ancak bir pencerede 5 saklayıcı olduğundan belli bir anda bunlardan sadece 5 tanesi kullanılabilir.

Programlarda sadece R0-R4 kullanılır ancak pencere değiştiğinde bunlar farklı saklayıcılara denk düşerler.

A'da programcı R0'a eriştiğinde işlemcinin R0'ına erişmiş olur.

B'de programcı R0'a eriştiğinde işlemcinin R3'üne erişmiş olur.



Tüm alt programların eriştiği ve numaraları değişmeyen global saklayıcılar da bulunur.

**Saklayıcı sayılarının belirlenmesi:**

G: Global saklayıcı sayısı

L: Yerel saklayıcı sayısı

C: İki pencere arasındaki ortak saklayıcı sayısı

W: Pencere Sayısı

Pencere boyu =  $L+2C+G$  ( $2 \cdot C$  çünkü hem alttaki hem de üstteki pencere ile ortak saklayıcılar vardır.)

Saklayıcı sayısı =  $(L+C)W + G$

Pencere sistemi çevrel (circular) olarak tasarlanır.

Eğer işlemcinin 4 adet penceresi varsa, iç içe 5nci alt program çağırıldığında en eski programın 1nci pencerede yer alan bilgileri belleğe yazılır.

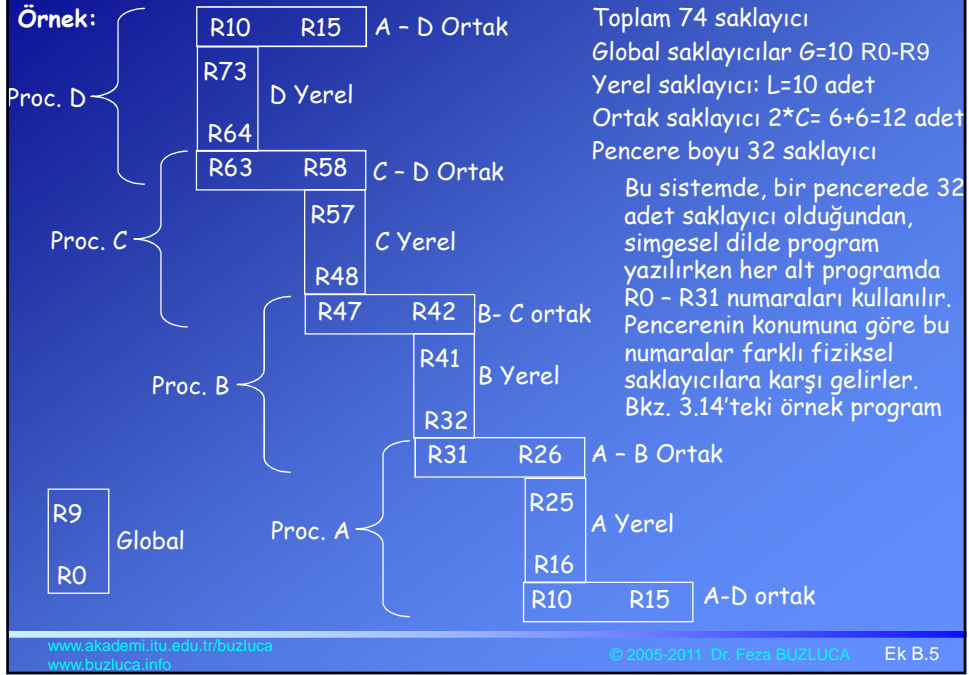
Bundan sonra 1nci pencere 5nci alt program tarafından kullanılır.

Geri dönüşte bellekteki bilgiler tekrar ilgili pencereye taşınır.

**Örnek:**

Sonraki örnekte, toplam 74 adet saklayıcısı bulunan, 32 saklayıcılı pencereye sahip ve 4 derinliğinde alt program çağırılmasına destek veren bir işlemcinin saklayıcı yapısı gösterilmiştir.

Bu örnekte alt programa gidildikçe pencerelerin artan numaralı saklayıcılara doğru ilerlediği var sayılmıştır. Gerçek işlemcilerde (RISC 1, SPARC) pencereler azalan adreslere doğru ilerlemektedir.



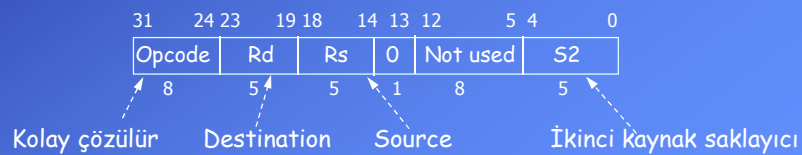
### RISC İşlemci Örneği: Berkeley RISC I

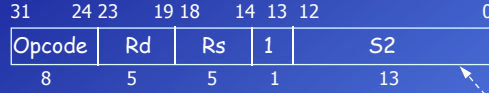
- 32 bit adres yolu
- von Neumann mimarisi: Komut ve veri belleği ortaktır.
- 8, 16, 32 bitlik veriler
- Komutlar sabit uzunlukta ve 32 bit
- Toplam 31 komutu var
- Toplam 138 saklayıcı (R0-R137),  
8 pencere her pencerede 32 saklayıcı, 10 adet global saklayıcı (R0-R9)  
Yemel saklayıcı: 10 adet, Ortak saklayıcı  $6+6=12$  adet
- 3 adet adresleme kipi: saklayıcı adresleme, ivedi adresleme, bağıl adresleme

#### Komut Yapısı:

##### 1. Saklayıcı kipi: (Register mode)

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



**2. Saklayıcı - ivedi (Register-immediate mode):**

Örnek: ADD R22, #150, R23

$R23 \leftarrow R22 + 150$

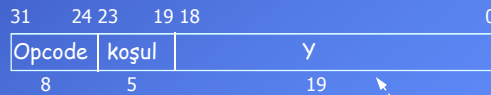
İvedi veri

Bellek erişimi komutlarında, Rs adres saklayıcısı (işaretçi), S2 ise öteleme miktarı olarak kullanılır. Rs'nin içindeki 32 bitlik adres + S2'nin gösterdiği yere erişilir.

Örnek: LDL (R10)#5,R5  $R5 \leftarrow M[R10 + 5]$  Load long: 32 bitlik veri aktarımı

**3. Bağlı (PC Relative mode):**

Örnek: JMPR EQ,Y



Öteleme (offset)

**Berkeley RISC I komutlarının kullanılması**

R0 her zaman sabit 0 (sıfır) değerini taşır.

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

ADD R0, #150, R22  $R22 \leftarrow 150$  (İvedi yükleme)

ADD R22, #1, R22  $R22 \leftarrow R22 + 1$  (Increment)

Bellek erişimi için Load/Store komutları kullanılır.

LDL (R22)#150,R5  $R5 \leftarrow M[R22 + 150]$  Load long: 32 bitlik veri aktarımı

LDL (R22)#0,R5  $R5 \leftarrow M[R22]$

LDL (R0)#500,R5  $R5 \leftarrow M[500]$

## Berkeley RISC I Komut Tablosu

Veri işleme komutları:

<b>Opcode</b>	<b>Operands</b>	<b>Register Transfer</b>
ADD	Rs,S2,Rd	$Rd \leftarrow Rs + S2$
ADDC	Rs,S2,Rd	$Rd \leftarrow Rs + S2 + \text{carry}$
SUB	Rs,S2,Rd	$Rd \leftarrow Rs - S2$
SUBC	Rs,S2,Rd	$Rd \leftarrow Rs - S2 - \text{carry}$
SUBR	Rs,S2,Rd	$Rd \leftarrow S2 - Rs$
SUBCR	Rs,S2,Rd	$Rd \leftarrow S2 - Rs - \text{carry}$
AND	Rs,S2,Rd	$Rd \leftarrow Rs \wedge S2$
OR	Rs,S2,Rd	$Rd \leftarrow Rs \vee S2$
XOR	Rs,S2,Rd	$Rd \leftarrow Rs \oplus S2$
SLL	Rs,S2,Rd	$Rd \leftarrow Rs$ shifted by S2
SRL	Rs,S2,Rd	$Rd \leftarrow Rs$ shifted by S2
SRA	Rs,S2,Rd	$Rd \leftarrow Rs$ shifted by S2

Veri aktarım komutları:

<b>Opcode</b>	<b>Operands</b>	<b>Register Transfer</b>	
LDL	(Rs)S2,Rd	$Rd \leftarrow M[Rs + S2]$	Long load
LDSU	(Rs)S2,Rd	$Rd \leftarrow M[Rs + S2]$	Short unsigned
LDSS	(Rs)S2,Rd	$Rd \leftarrow M[Rs + S2]$	Short signed
LDBU	(Rs)S2,Rd	$Rd \leftarrow M[Rs + S2]$	Byte unsigned
LDBS	(Rs)S2,Rd	$Rd \leftarrow M[Rs + S2]$	Byte signed
LDHI	Y,Rd	$Rd \leftarrow Y$	Immediate high
STL	(Rs)S2, Rm	$M[Rs + S2] \leftarrow Rm$	Store load
STS	(Rs)S2, Rm		
STB	(Rs)S2, Rm		
GETPSW	Rd	$Rd \leftarrow PSW$	Load status word
PUTPSW	Rd	$PSW \leftarrow Rd$	Set status word

## Program denetim komutları:

Opcode	Operands	Register Transfer	
JMP	COND,S2(Rs)	$PC \leftarrow Rs + S2$	Mutlak (doğrudan) adresleme
JMPR	COND,Y	$PC \leftarrow PC + Y$	Bağıl
CALL	S2(Rs),Rd	$Rd \leftarrow PC$ $PC \leftarrow Rs + S2$ $CWP \leftarrow CWP - 1$	Current window pointer
CALLR	Y,Rd	$Rd \leftarrow PC$ $PC \leftarrow PC + Y$ $CWP \leftarrow CWP - 1$	Bağıl
RET	(Rd)S2	$PC \leftarrow Rd + S2$ $CWP \leftarrow CWP + 1$	

Berkeley RISC I işlemcisinde altprograma gidildiğinde pencere işaretçisi (CWP) azaltıldığından daha küçük numaralı saklayıcılara doğru gidilir.

Buna göre ana program (A prosesi) en yüksek numaralı saklayıcıları (R116-R137) ve global saklayıcıları (R0-R9) kullanır.

## Örnek Program:

500 ve 504 numaralı bellek gözlerinde bulunan 32 bitlik iki işaretli sayının toplamını gerçekleştiren ve sonucu 508 numaralı bellek gözüne yazan programı Berkeley RISC-1 simgesel dili ile kesişimli saklayıcı pencereler üzerinde parametre aktarımı gerçekleştirerek yazınız.

Toplama alt programı, R1 numaralı saklayıcıdaki adresin 20 ilerisinden başlamaktadır.

Çözüm:	Program	Açıklama
	LDL (R0) #500, R10	$R10 \leftarrow M[500]$ (1. parametre)
	LDL (R0) #504, R11	$R11 \leftarrow M[504]$ (2. parametre)
	CALL (R1)#20, R15	$R15 \leftarrow PC$ $PC \leftarrow (R1)+20$ $CWP \leftarrow CWP-1$
	STL (R0) #508, R12	$M[508] \leftarrow R12$ (geri dönen değer)
	...	
	...	
[(R1)+20]	ADD R26, R27, R28	$R28 \leftarrow R27+R26$
	RET (R31)#0	$PC \leftarrow (R31)+0$ $CWP \leftarrow CWP+1$

**Not:** Bu program yazılırken 4. bölümde anlatılan iş hattında (pipeline) çıkan sorunlar dikkate alınmamıştır.