Bilgisayar Mimarisi

2023-2024 Bahar Dönemi

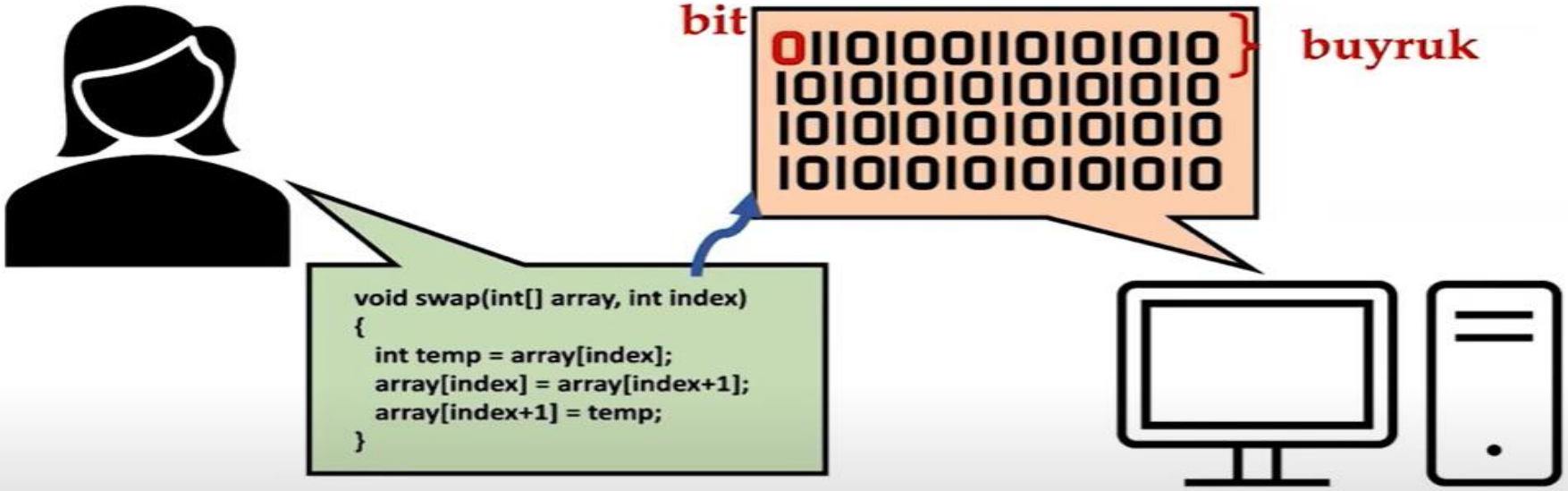
Hafta-4

Dr. Öğr. Üyesi Nurullah ÖZTÜRK

Bilgisayarın Dili

Bilgisayarların Dili

Bilgisayara hesaplama yaptırmak için anlayacağı dilde konuşmak gerekir.



Bilgisayarın Dili

Bilgisayar dili **sözlüğü (buyruk kümesi):**

Kelime (Buyruk)	Anlam
101010011	0 ve 3 sayılarını topla sonucu birinci yazmaca yaz.
111010011	0 ve 3 sayılarını çıkar sonucu birinci yazmaca yaz.

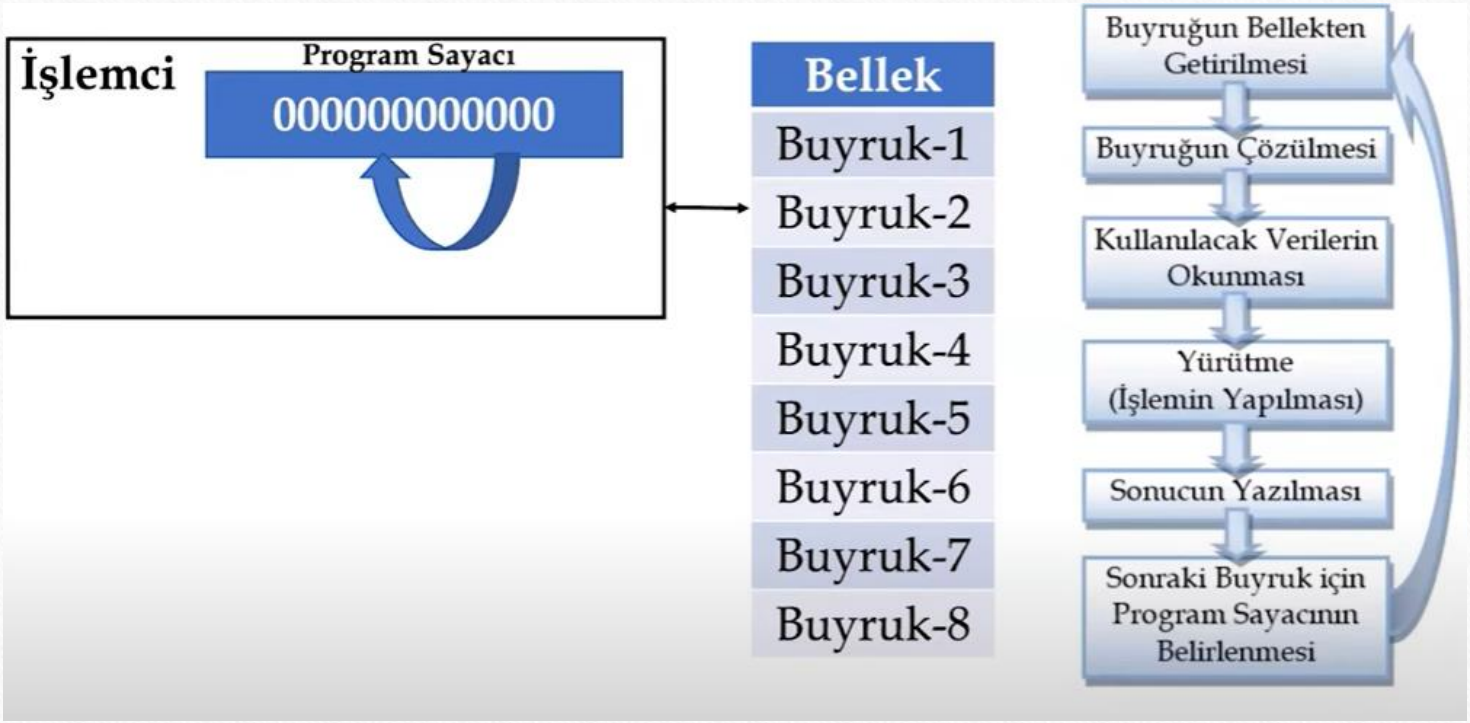
Yazılım ile donanım arasındaki arayüzü bu sözlük tanımlar.

"Bu işlemci hangi işleri yapabilir?"

Bilgisayarın Dili

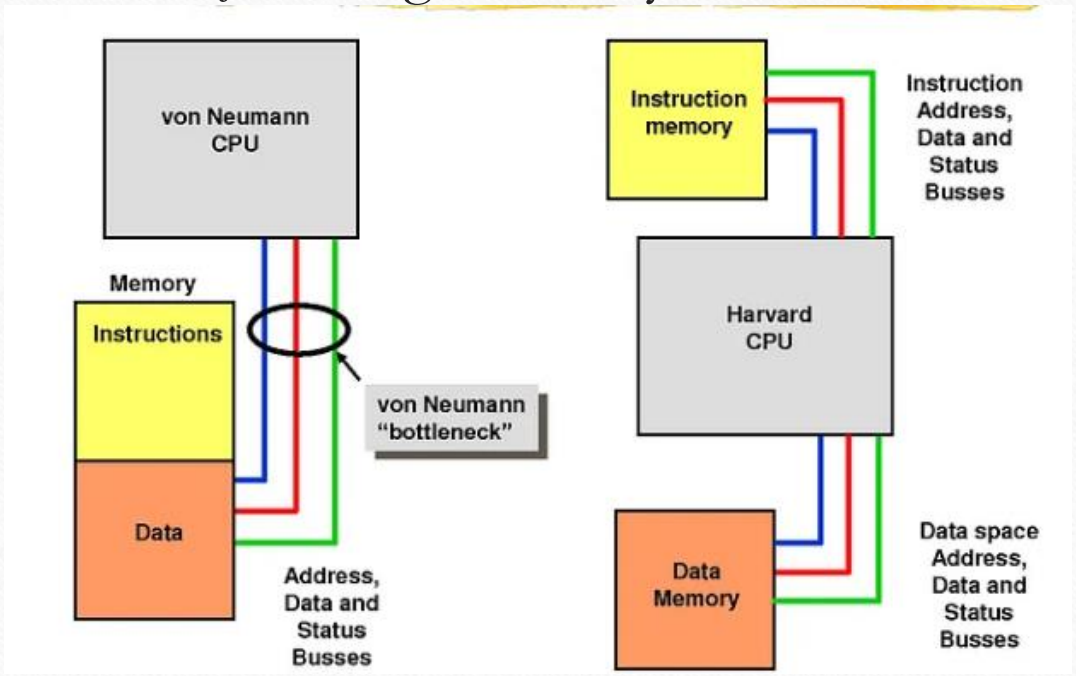
- Buyruk bit vektörüdür.
- Buyruk kümesi yazılım ile donanım arasındaki arayüz denebilir.
- Yazmaç, register ifade etmektedir.
- Register işlemci içindeki geçici saklama alanlarıdır.
- Yazmaç değişkenleri saklamak için kullanılmaktadır.

Buyrukların Yürütülme Döngüsü



Van Neumann Mimarisi & Harvard Mimarisi

- Bilgisayar sistemlerinde işlevselliği tanımlayan iki adet mimari bulunmaktadır.



Van Neumann Mimarisi & Harvard Mimarisi

VON NEUMANN ARCHITECTURE	HARVARD ARCHITECTURE
Komutların depolanması konseptine sahip eski bilgisayar mimarisidir.	Harvard Mark I röle tabanlı modele dayanan modern bilgisayar mimarisidir.
Veriler ve komutlar için aynı fiziksel bellek adresi kullanılır.	Veriler ve komutlar için farklı fiziksel bellek adresi kullanılır.
Veri ve komut aktarımı için ortak bir veriyolu vardır.	Veri aktarımı ve talimat için ayrı veri yolları kullanılır.
Bir komutun yürütülmesi için iki clock cycle kullanılır.	Bir komutun yürütülmesi için tek clock cycle kullanılır.
Maliyet olarak daha ucuzdur.	Van Neumann mimarisinden daha maliyetlidir.
CPU aynı anda sadece tek bir işlem gerçekleştirebilir çünkü tüm işlemler tek bir veriyolu üzerinden gerçekleştirilir.	Harvard mimarisine sahip bir bilgisayarda ise, CPU, aynı anda hem komut okuyabilir hem de bellek erişimi gerçekleştirebilir, bir önbellek olmasa bile.
Kişisel bilgisayarlarda ve küçük bilgisayarlarda kullanılır.	Mikrodenetleyicilerde ve sinyal işlemede kullanılır.

RISC VE CISC MİMARİSİ

- 2 farklı buvruk kümesi mimarisi vardır.

RISC: Reduced Instruction Set Computer

CISC: Complex Instruction Set Computer

Genel kıyaslama:

CISC	RISC
Çok sayıda karmaşık buyruk	Az sayıda basit buyruk
Buyruk boyutları değişken	Buyruk boyutları sabit
Karmaşık donanım	Karmaşık derleyici
Belleğe erişim herhangi bir buyruk tarafından gerçekleştirilebilir	Yalnızca LOAD ve STORE buyrukları belleğe erişim gerçekleştirir
Çok sayıda adresleme kipi	Az sayıda adresleme kipi

RISC-V İŞLEMLERİ

Tüm bilgisayarlar **aritmetik işlem** yapabilmelidir.

`add a, b, c` → b ve c *değişkenlerini* topla, sonucu a *değişkenine* yaz.

RISC-V Assembly Gösterimi

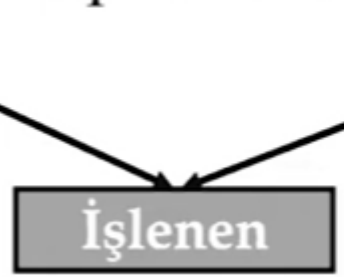
RISC-V aritmetik buyruklarında her zaman **üç işlenen** (a, b ve c) vardır.

- Sabit uzunlukta buyruklar.
  - \+ **Basit donanım.**
  - \- **Büyük kod. Daha fazla buyruk.**

RISC-V İşlenenleri

`add a, b, c` → b ve c *değişkenlerini* topla, sonucu a *değişkenine* yaz.

RISC-V Assembly Gösterimi



RISC-V aritmetik buyruklarının işlenenleri **yazmaçlar** veya **anlık değerlerdir**.

Yazılımda işlenenlerin (değişkenler) aksine donanımda işlenenler (yazmaçlar) **sınırlı sayıdadır**.

RISC-V Yazmaçları

RISC-V, RV64 mimarisinde:

- Yazmaçlar 64-bit genişliğindedir.
  - 64-bit = Çift-kelime (-ing. doubleword)
  - 32-bit = Kelime (-ing. word)
- 32 yazmaç vardır.
  - Neden 32'den fazla değil? → **Küçük olan hızlıdır.**

Kelime: Bilgisayarlarda alışılagelmiş bir erişim boyutu birimi. Genellikle 32-bit.

Not: RISC-V mimarisinde sıfırıncı yazmacın değeri daima 0'dır.

RISC-V Aritmetik Buyruklar

Buyruk	Örnek	Anlamı	Açıklama
Topla	add x5, x6, x7	x5 = x6 + x7	x6 ve x7 yazmaçlarını topla, x5'e yaz.
Çıkar	sub x5, x6, x7	x5 = x6 – x7	x6'dan x7'yi çıkar, x5'e yaz.
Anlıkla Topla	addi x5, x6, 20	x5 = x6 + 20	x6 yazmacını 20 **(anlık değer)** ile topla, x5'e yaz.

C kodu Derleme Örneği

Yazılımda işlenenlerin (değişkenler) aksine donanımda işlenenler (yazmaçlar) sınırlı sayıdadır.

- Derleyici bu değişkenleri yazmaçlara eşleştirmelidir.

```
f = (g + h) - (i + j);
```

Derleyici Değişken Eşleştirmesi
f → x19, g → x20, h → x21, i → x22, j → x23

```
add x5, x20, x21
add x6, x22, x23
sub x19, x5, x6
```

```
// x5 = g + h
// x6 = i + j
// f = (g + h) – (i + j)
```

Bellek

Programlama dilleri **birden çok veri elemanından** oluşan **dizi** benzeri yapıları barındırabilirler.

Bu **veri yapıları (Örn. ağaçlar, diziler)** bir bilgisayardaki **yazmaçlara sığamayacak** kadar büyük boyutta olabilir.

Bu tür veri yapıları **bellekte** tutulur.

Belleğe Erişim

RISC-V mimarisinde aritmetik işlemler yalnızca yazmaçlar üzerinde gerçekleştirilir.

Bellekte bulunan veri üzerinde hesaplama yapabilmek için **veri aktarma buyruklarını** (-ing. data transfer instructions) kullanırız.

Verinin bellekteki yerini **bellek adresleri** belirler.

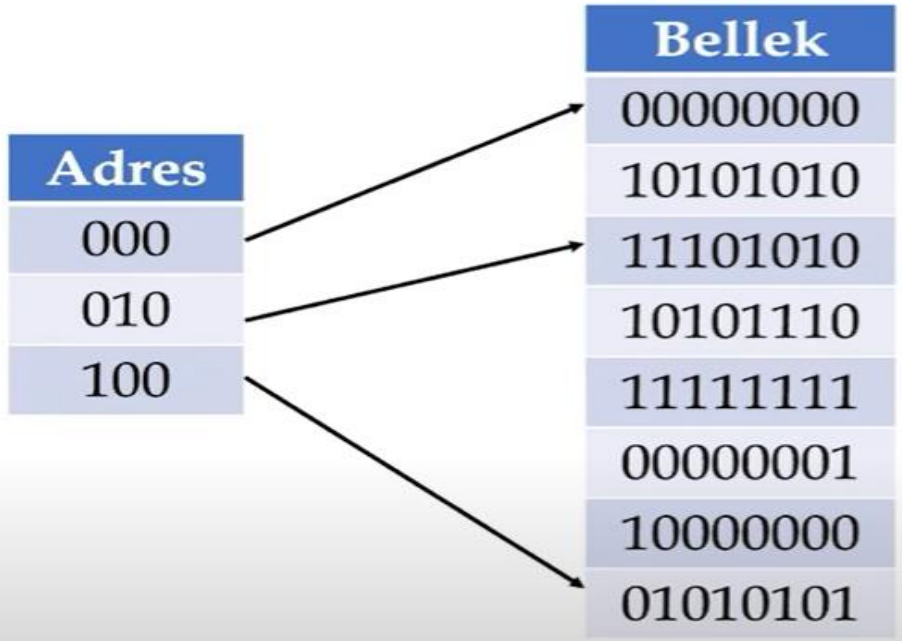
Temel RISC-V veri aktarma buyrukları:

Buyruk	Örnek	Anlamı	Açıklama
Yükle	ld x5, 8(x7) (Load Doubleword)	x5 = x7[8]	x7 başlangıç adresli dizinin 8. elemanını x5'e yaz. (bellekten okur)
Sakla	sd x5, 8(x7) (Store Doubleword)	x7[8] = x5	x5'i x7 başlangıç adresli dizinin 8. elemanına yaz. (belleğe yazar)

Bellek Adresleme

Bayt Adresleme

RISC-V mimarisinde **her bellek adresi** bellekte bir **bayta** işaret eder.



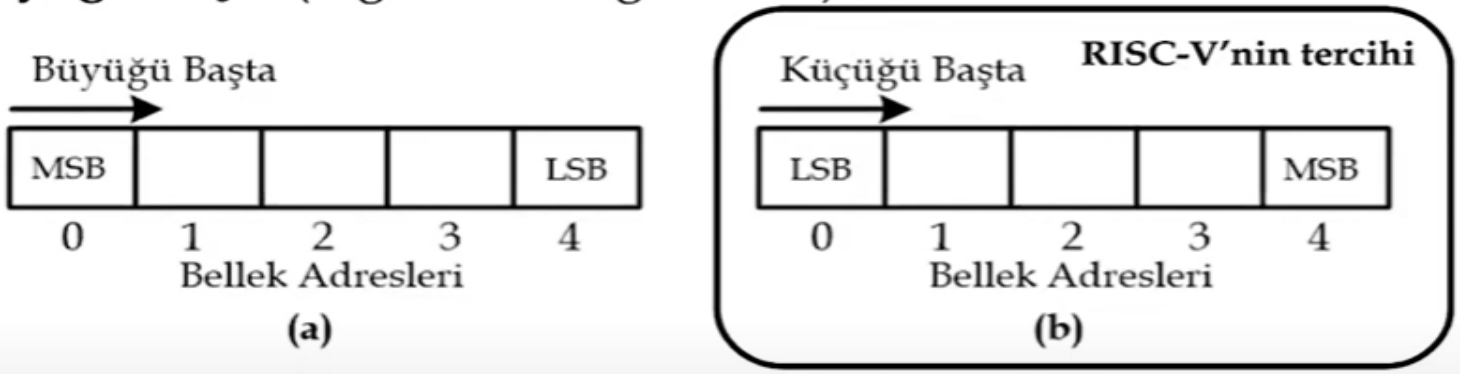
Buna göre **16-bit adresler** ile **kaç bit boyutunda** bellek adreslenebilir?

$2^{16} bayt = 2^{16} * 8\ bit = \mathbf{2^{19} bit}$

Küçüğü Başta - Büyüğü Başta

Verilerin hangi baytının önce yerleştirildiğine göre bilgisayarlar ikiye ayrılır:

Sayının **en küçük**, en anlamsız, en sağdaki baytının **en küçük adresli bellek konumuna** yazıldığı düzene **Küçüğü Başta** (İngilizcesi: Little Endian), sayının **en büyük**, en anlamlı, en soldaki baytının **en küçük adresli bellek konumuna** yazıldığı düzene ise **Büyüğü Başta** (İngilizcesi: Big Endian) denir.



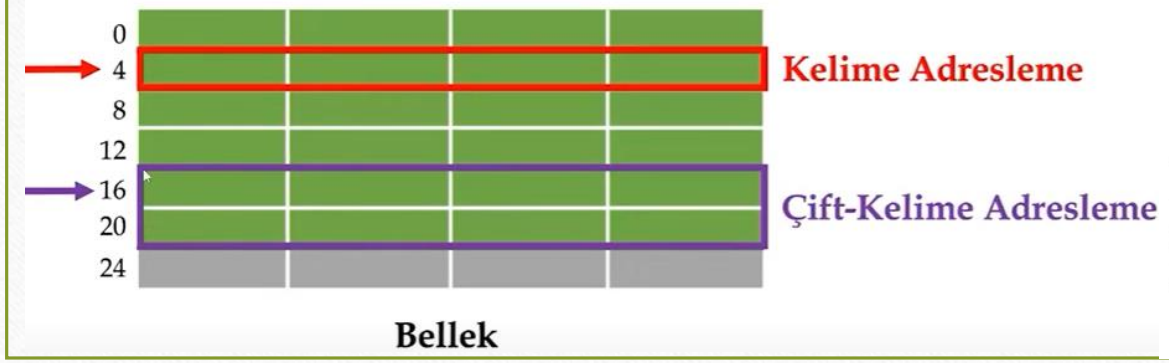
Şekil 2.8 (a) Büyüğü Başta, (b) Küçüğü Başta

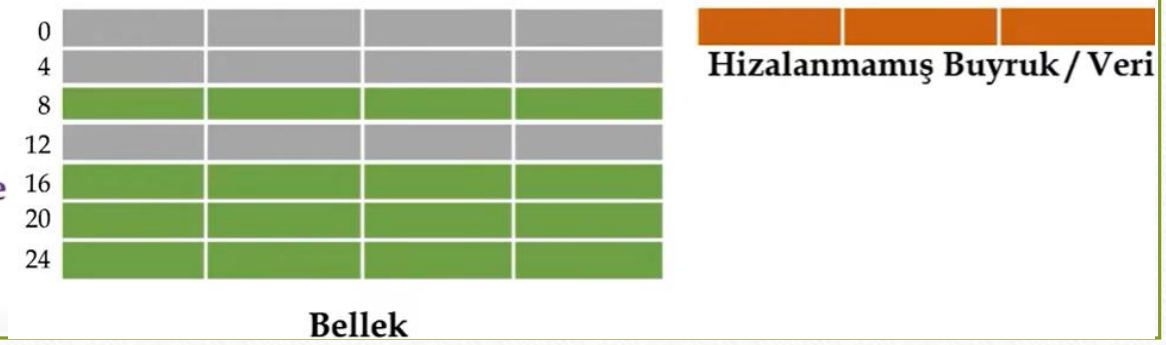
Bellek Adreslerinin Hizalanması

Başka mimarilerde (MIPS) de olduğu gibi RISC-V **veri transfer buyruklarının** adreslerinde bazı **hizalama** kısıtları vardır.

- Çift-kelime (doubleword) erişimlerinin adresleri **8'in** tam katı olmalıdır.
- Kelime (word) adresleri **4'ün** tam katı olmalıdır.

Bu kısıtlar **donanımın basit kalmasını** sağlar.





Bellek Adreslerinin Hizalanması

- Bellekte saklanan her veri(byte) ın adresi vardır. Bellege bir adres verdiğinde ise 1 byte okursun.
- Bir bite bir adres verirsen bir sürü adres okuması yapılmalıdır.
- RISC-V mimarisi byte byte okumaz. Yazmaçlar 32 bittir ve sözcük sözcük okuma yapar.
- Bu sayede hizalı erişim gerçekleşir.
- Hızalı erişimde sözcüğün tam katı şekilmelidir.

Sabit Boyutlu Buyruk Genişliği

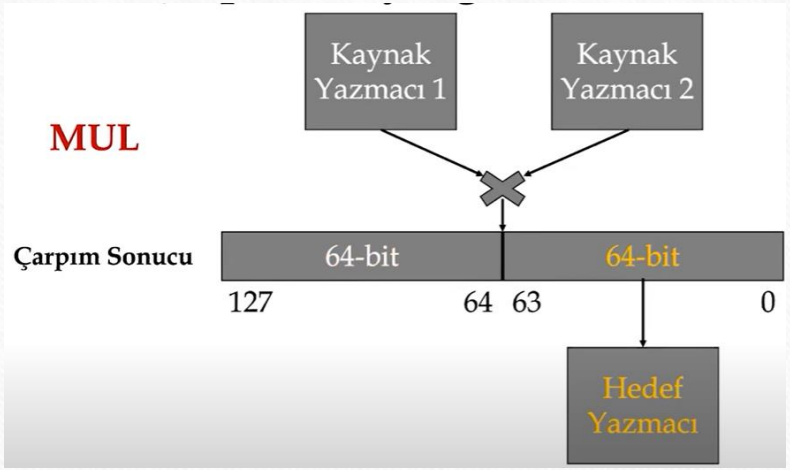
Değişken Boyutlu Buyruk Genişliği

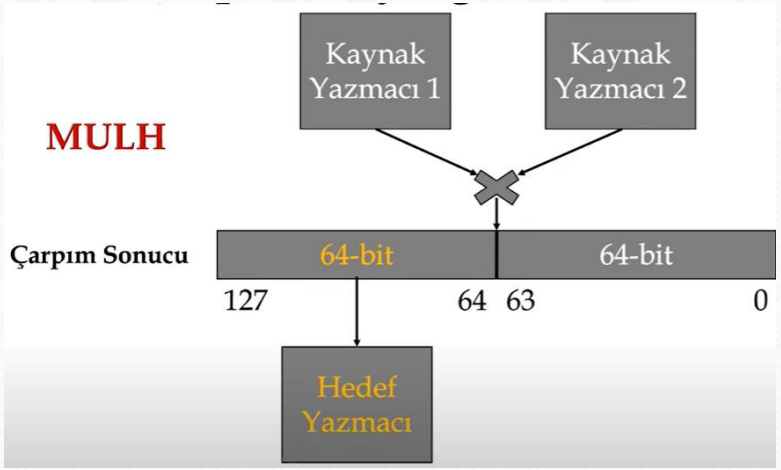
Değişken Boyutlu Buyruk Genişliği

- Bellek yapımın zorluğundan dolayı belleği verimli kullanmak için gereklidir.

Çarpma İşlemi

- X bitlik 2 değerin çarpımı 2X bit olur.
- 64 bitlik bir yazmaçta 64 bit iki değerin çarpımı 128 bit olmaktadır bu yüzden sonucu kaydetmek için iki hedef yamaç gereklidir.





Kodun Belleğe Saçılması

Çoğu programda yazmaçlardan fazla sayıda değişken vardır.

- Tüm değişkenler her zaman yazmaçlarda bulunamaz.

Derleyici **yakın zamanda kullanılmayacak** değişkenleri **bellekte** tutmaya çalışır. Buna kodun belleğe saçılması denir.

Yazmaçlara erişim belleğe erişimden hızlıdır. Programların **verimli** çalışması için:

- Donanımda **yeterince yazmaç** olması,
- **Derleyicilerin yazmaçları verimli kullanması** gerekir.

Buyrukların Kodlanması

Donanımda buyruklar elektrik sinyalleri ile iletilir.

- Sayısal devrelerdeki iki farklı gerilim değeri:
  - Toprak (0)
  - Kaynak (1)

Basitçe, bir buyruk sayı olarak gösterilen küçük parçalardan oluşur.

Alan: Buyruğun bilgi içeren bir parçası

`add x9, x20, x21` →

Onluk Taban:	0	21	20	0	9	51
İkilik Taban:	0000000	10101	10100	000	01001	0110011
	7 bit	5 bit	5 bit	3 bit	5 bit	7 bit

Toplama (add) — Kaynak Yazmacı (x21) — Kaynak Yazmacı (x20) — Hedef Yazmacı (x9)

Buyrukların Kodlanması

0000000	10101	10100	000	01001	0110011
7 bit	5 bit	5 bit	3 bit	5 bit	7 bit

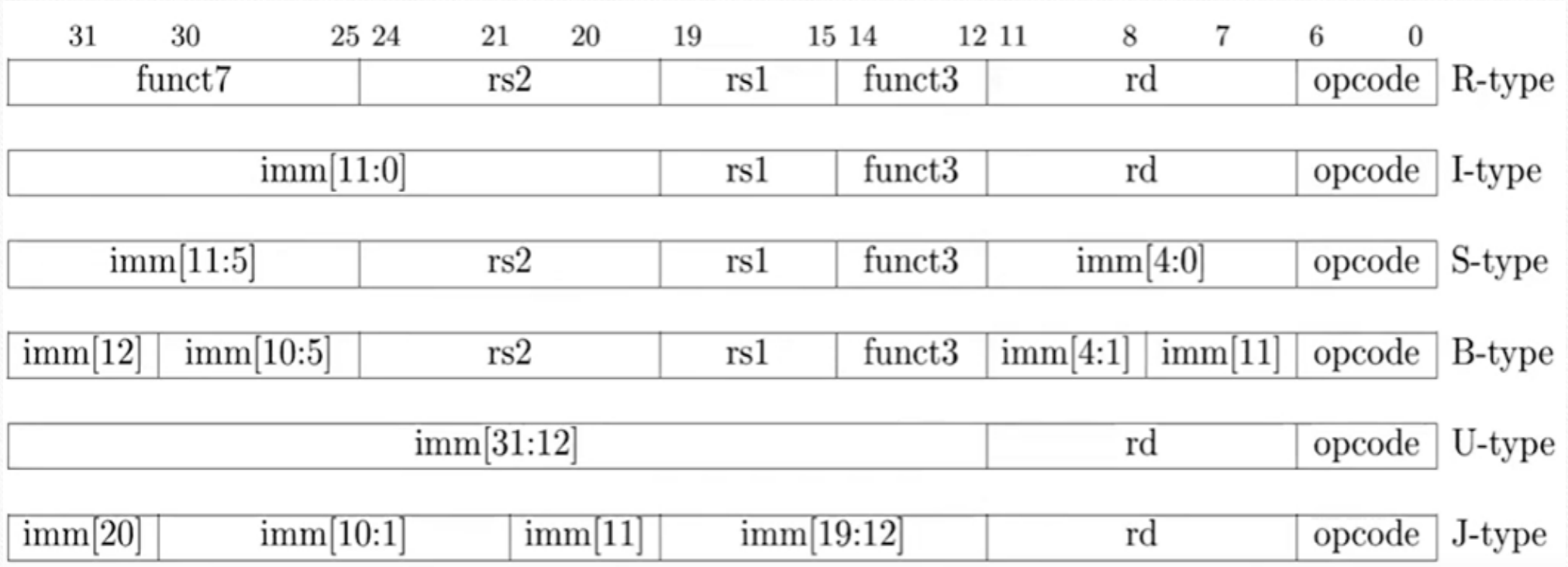
Yukarıdaki bir RISC-V **buyruk formatıdır**.

- Bütün RISC-V buyrukları **32-bit** genişliğindedir.
  - **Basit tasarım.**

Makine dili: Buyrukların sayısal gösterimi.

Makine kodu: Sayısal olarak gösterilen buyruklar dizisi.

Tüm RISC-V Buyruk Tipleri



RISC-V Buyruk Türleri

Kategori	Komut	Örnek	Anlamı
Aritmetik	topla	add $s1,$s2,$s3	$s1 = $s2 + $s3
	çıkar	sub $s1,$s2,$s3	$s1 = $s2 – $s3
	mutlak ekle	addi $s1,$s2,20	$s1 = $s2 + 20
Veri geçiş	kelime yükle	lw $s1,20($s2)	$s1 = Memory[$s2 + 20]
	kelime kaydet	sw $s1,20($s2)	Memory[$s2 + 20] = $s1
	yarım yükle	lh $s1,20($s2)	$s1 = Memory[$s2 + 20]
	işaretsiz yarım yükle	lhu $s1,20($s2)	$s1 = Memory[$s2 + 20]
	yarım kaydet	sh $s1,20($s2)	Memory[$s2 + 20] = $s1
	byte yükle	lb $s1,20($s2)	$s1 = Memory[$s2 + 20]
	işaretsiz byte yükle	lbu $s1,20($s2)	$s1 = Memory[$s2 + 20]
	byte kaydet	sb $s1,20($s2)	Memory[$s2 + 20] = $s1
	bağlı kelime yükle	ll $s1,20($s2)	$s1 = Memory[$s2 + 20]
	koşullu kelime kaydet	sc $s1,20($s2)	Memory[$s2 +20]=$s1;$s1=0 or 1
	üst mutlak yükle	lui $s1,20	$s1 = 20 * 2^{16}
Mantıksal	ve	and $s1,$s2,$s3	$s1 = $s2 & $s3
	veya	or $s1,$s2,$s3	$s1 = $s2 | $s3
	veya değil	nor $s1,$s2,$s3	$s1 = ~ ($s2 | $s3)
	ve mutlak	andi $s1,$s2,20	$s1 = $s2 & 20
	veya mutlak	ori $s1,$s2,20	$s1 = $s2 | 20
	mantıksal sola kaydır	sll $s1,$s2,10	$s1 = $s2 << 10
	mantıksal sağa kaydır	srl $s1,$s2,10	$s1 = $s2 >> 10
Koşullu dallanma	eşitse dallan	beq $s1,$s2,25	if ($s1 == $s2) go to PC + 4 + 100
	eşit değilse dallan	bne $s1,$s2,25	if ($s1 != $s2) go to PC + 4 + 100
	küçükse bir yap	slt $s1,$s2,$s3	if ($s2 < $s3) $s1 = 1; else $s1 = 0
	işaretsiz küçükse bir yap	sltu $s1,$s2,$s3	if ($s2 < $s3) $s1 = 1; else $s1 = 0
	mutlaktan küçükse bir yap	slti $s1,$s2,20	if ($s2 < 20) $s1 = 1; else $s1 = 0
	işaretsiz mutlaktan küçükse bir yap	sltiu $s1,$s2,20	if ($s2 < 20) $s1 = 1; else $s1 = 0
Koşulsuz atlama	atla	j 2500	go to 10000
	kaydediciye atla	jr $ra	go to $ra
	atla ve bağla	jal 2500	$ra = PC + 4; go to 10000