دانشگاه کردستان
University of Kurdistan
زانکۆی کوردستان

Department of Computer and IT Engineering
University of Kurdistan

Computer Architecture

MIPS ISA and Assembly

By: Dr. Alireza Abdollahpouri

مقدمه

- MIPS یکی از اولین و موفقترین پردازندهای RISC است که بصورت تجاری عرضه شده و در سال ۱۹۸٤ توسط تیمی دردانشگاه استانفورد طراحی شده است.
- پردازنده ای ساده ولی در عین حال قوی است.
- در تجهیزات مختلفی بصورت تعبیه شده (embedded) استفاده شده است:
  - Various routers from Cisco
  - Game machines like the Nintendo 64 and Sony Playstation 2

Most HP LaserJet workgroup printers are driven by MIPS-based™ 64-bit processors.

ویژگیها

- تعداد زیاد رجیسترهای همه منظوره
- مجموعه کوچک دستورات
- اندازه دستورات ثابت (٤ بایت) ولی فرمت آنها متغیر است
- دسترسی به حافظه محدود به دستورات load/store است
- مد های آدرس دهی محدود است

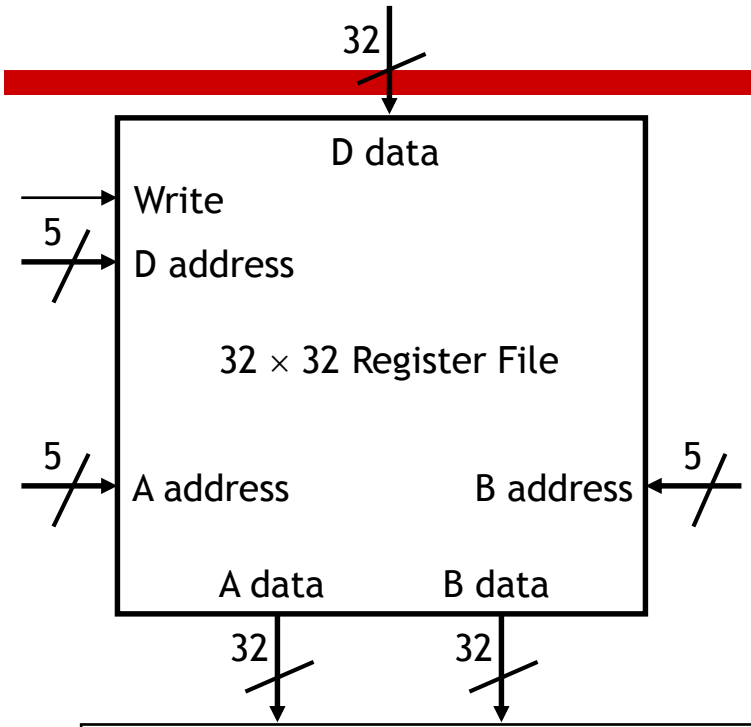
رجیسترها

- این پردازنده دارای ۳۲ رجیستر عمومی ۳۲ بیتی است:
  - R0 .. R31
  - رجیستر R0 بصورت سخت افزاری با مقدار صفر پر شده است یعنی همیشه برابر با صفر است
  - رجیستر R1 برای کار اسمبلر رزرو شده است
  - از بقیه رجیستر ها میشود در برنامه ها استفاده نمود.
- عملوند ها همیشه باید در یکی از رجیستر ها قرار داشته باشند.

$0 = $zero	$16 = $s0
$1 = $at	$17 = $s1
$2 = $v0	$18 = $s2
$3 = $v1	$19 = $s3
$4 = $a0	$20 = $s4
$5 = $a1	$21 = $s5
$6 = $a2	$22 = $s6
$7 = $a3	$23 = $s7
$8 = $t0	$24 = $t8
$9 = $t1	$25 = $t9
$10 = $t2	$26 = $k0
$11 = $t3	$27 = $k1
$12 = $t4	$28 = $gp
$13 = $t5	$29 = $sp
$14 = $t6	$30 = $fp
$15 = $t7	$31 = $ra

برای اینکه برنامه نویسی اسمبلی راحت تر باشد به هر رجیستر اسمی داده شده است

فایل رجیستر



- رجیسترها به صورت یک مجموعه به نام «فایل رجیستر» سازماندهی شده اند

If Write = 1, then D data is stored into D address.

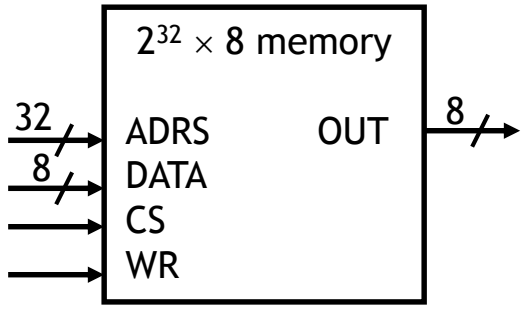
- You can read from two registers at once, by supplying the A address and B address inputs. The outputs appear as A data and B data.
- Registers are clocked, sequential devices.
  - We can read from the register file at any time.
  - Data is written only on the positive edge of the clock.

سایر رجیستر ها

- علاوه بر ر جیسترهای فوق MIPS دارای رجیسترهای دیگری نیز میباشد:

- PC (program counter) register
- Status register
- Floating point registers

حافظه MIPS

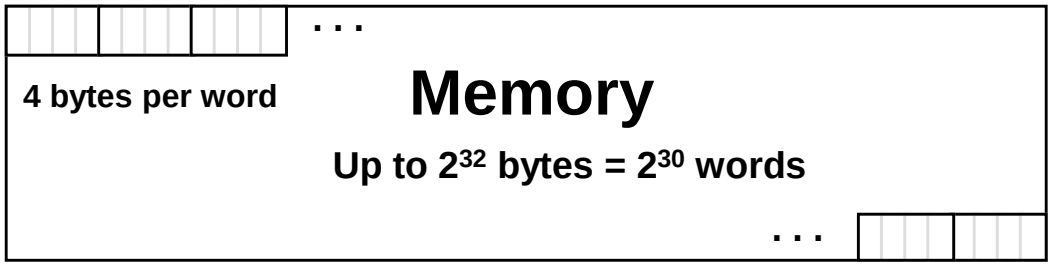


CS	WR	Operation
0	x	None
1	0	Read selected address
1	1	Write selected address

- MIPS دارای ۳۲ خط آدرس است. یعنی میتواند تا 2^{32}محل حافظه را آدرس دهی نماید. در هر محل حافظه یک بایت داده قرار میگیرند.
- **This results in a 2^{32} x 8 RAM, which would be 4 GB of memory.**

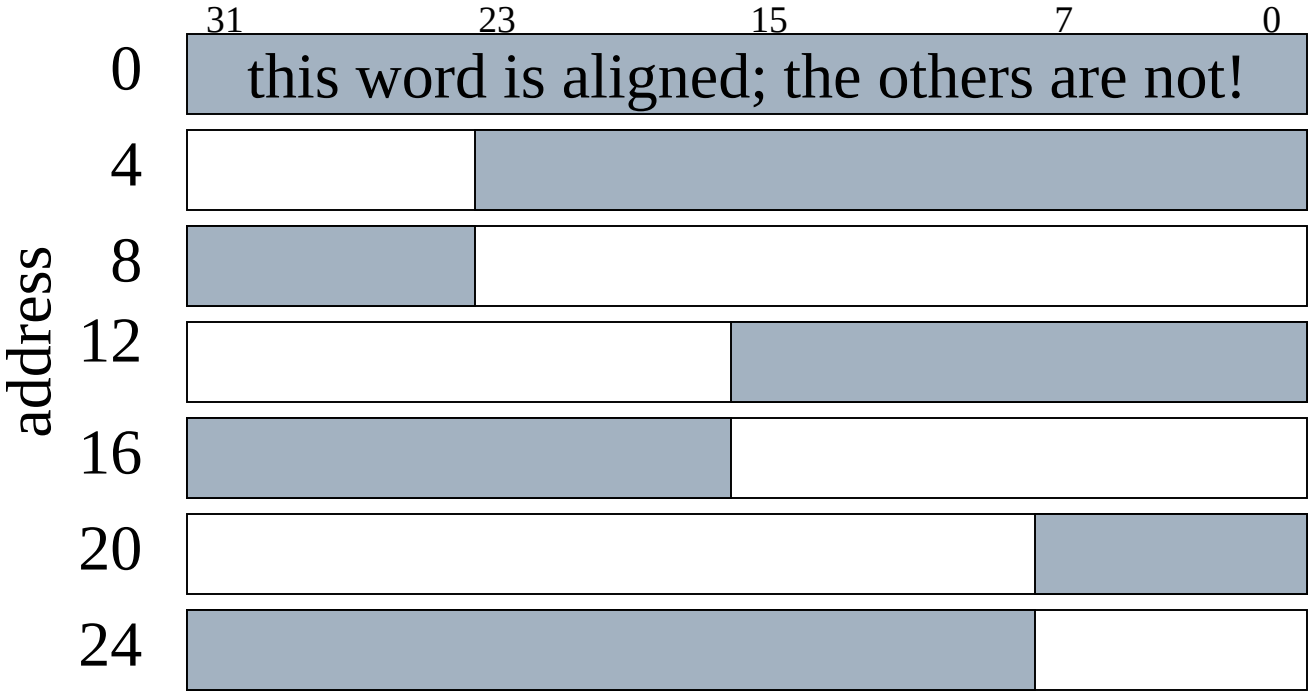
سازمان حافظه

- هر کلمه دارای ٤ بایت میباشد



- 2^{32} bytes with byte addresses from 0 to 2^{32}-1
- 2^{30} words with byte addresses 0, 4, 8, ... 2^{32}-4

Alignment : سازمان حافظه



در معماری MIPS کلمات باید بصورت تراز شده در حافظه قرار گیرند. یعنی یک کلمه ۳۲ بیتی باید در یک محلی از حافظه قرار گیرد که آدرس آن مضربی از ۴ باشد.

0, 4, 8 and 12 are valid word addresses

آرایه ای از کلمات

- باید هنگام کار با آرایه ها مراقب بود که اگر آرایه ای برای مثال از محل ۲۰۰۰ حافظه شرع شود، عضو اول آن در آدرس ۲۰۰۰ و عضو دوم آن در آدرس ۲۰۰٤ خواهد بود و نه در آدرس ۲۰۰۱
- برای مثال اگر رجیستر **$a0** دارای مقدار ۲۰۰۰ باشد:

```
lw $t0, 0($a0)
```

به اولین عضو اشاره میکند در حالیکه

```
lw $t0,8($a0)
```

به سومین عضو آرایه که در آدرس ۲۰۰۸ است دسترسی پیدا خواهد نمود.

انواع اصلی دستورالعملهای MIPS

- **Arithmetic and Logic**
  - Integer
  - Floating Point
- **Memory access instructions**
  - Load & Store
- **Control flow**
  - Jump
  - Conditional Branch
  - Call & Return

دستورات محاسباتی

- چهار دسته دستورات محاسباتی وجود دارند:
- Add
- Subtract
- Multiply
- Divide

دستورات محاسباتی

- تمامی دستورت ALU نظیر دستورات جمع و ضرب دارای ۳ عملوند هستند: یکی برای مقصد و دو تای دیگر برای مبدا داده ها. هر سه عملوند ها باید یکی از رجیستر های MIPS باشند. تمامی محاسبات ۳۲ بیتی هستند.

C code:
```
A = B + C;
E = F - A;
```

MIPS code:
```
add $t0, $s1, $s2
sub $s4, $s5, $s0
```

دستورات محاسباتی

اصول معماری در MIPS:

- تمامی محاسبات بر روی داده های رجیسترها انجام میشود. یعنی نمیتوان عددی را که در حافظه ذخیره شده است با یک رجیستر جمع کرد. برای اینکار ابتدا باید محتوی حافظه به رجیستر به منتقل شده و عملیات بر روی داده های رجیستر ها انجام شود.
- ترتیب اپراندها همیشه ثابت است: اول مقصد نوشته میشود.

دستورات محاسباتی

Add/Sub_Immediate instructions

یک عدد ۱۶ بیتی با علامت و یا بدون علامت را با یکی از رجیستر های ۱۶ بیتی جمع / تفریق مینماید.

Destination Reg = Source Register + Immediate

Example: A = A - 4

```
addi   $t0,  $t0,  -4      #  $t0 = $t0 –4
```

Signed/Unsigned Arithm: **addi, addiu**

مثال

- تبدیل یک دستور برنامه C به اسمبلی MIPS

```
a = b + c + d - e;
```

- Break into multiple instructions

```
add $t0, $s1, $s2 # temp = b + c
add $t0, $t0, $s3 # temp = temp + d
sub $s0, $t0, $s4 # a = temp - e
```

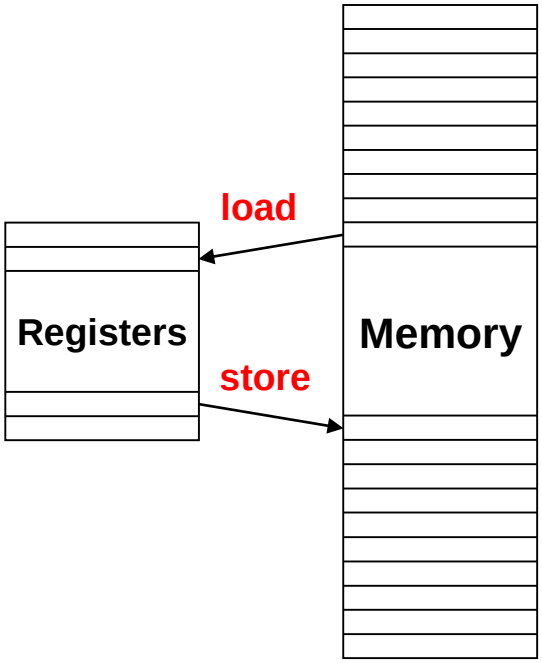
- Notice: A single line of C may break up into several lines of MIPS.

دستورات منطقی

- برخی ازدستورات منطقی موجود در **MIPS**
  - **AND**
    - bit-wise AND between registers
    - `and $t1, $s0, $s1`
  - **OR**
    - bit-wise OR between registers
    - `or $t1, $s0, $s1`
  - **NOR**
    - Bit-wise NOR between registers
    - `nor $t1, $s0, $s1`
    - `nor $t1, $t0, $0` `# $t1 = NOT($t0)`
  - Immediate modes
    - `andi` and `ori`

دستورات دسترسی به حافظه

Mnemonic	Instruction
LB	Load Byte
LBU	Load Byte Unsigned
LH	Load Halfword
LHU	Load Halfword Unsigned
LW	**Load Word**
SB	Store Byte
SW	**Store Word**
SH	Store Halfword



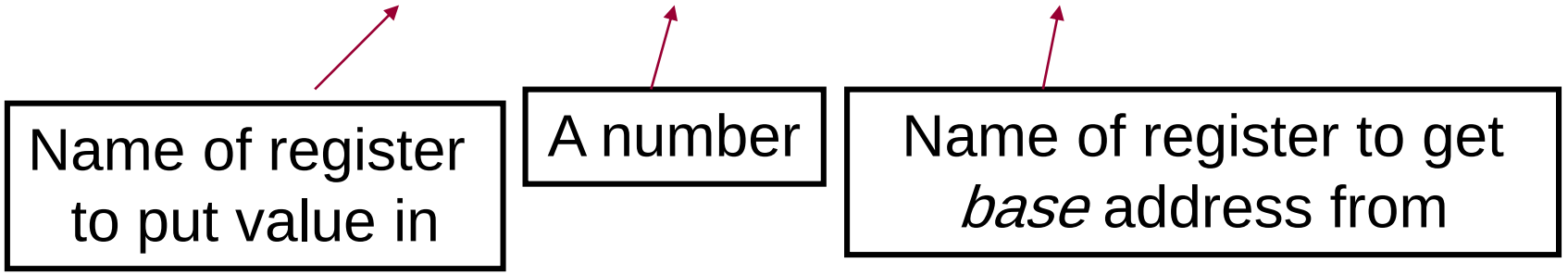
دستورات دسترسی به حافظه

- داده ها را بین حافظه و رجیسترها منتقل میکنند. دارای ۳ اپراند میباشند:
- LW/SW instruction:

Load/Store

Assembly language format(I-format):

label: operation dest_reg, offset (src_reg) # Comment



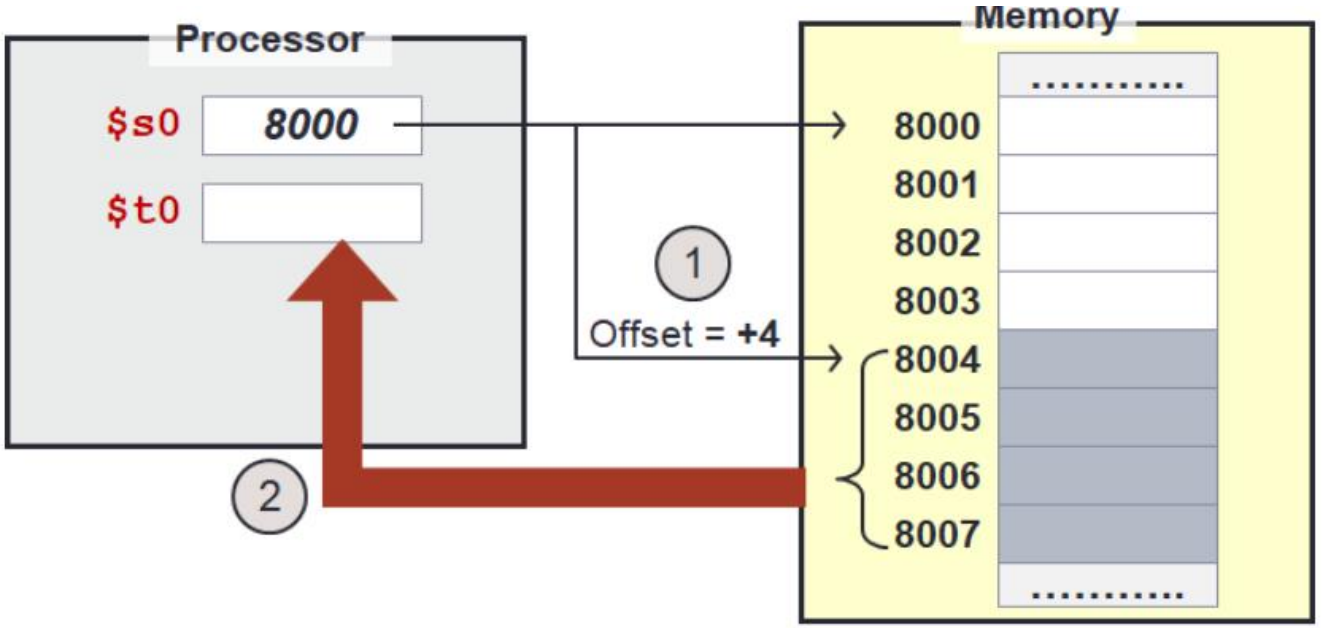
- آدرس داده در حافظه بصورت زیر محاسبه میشود:

Source Address = Source Base Address + Offset

Load Word :مثال

```
lw $t0, 4($s0)
```

If **$s0** has the value **8000**, this will copy the word at memory location **8004** to the register **$t0.**



Store Word :مثال

```
sw $t0, 4($s0)
```

If **$s0** has the value **8000**, this will copy the word in register **$t0** to memory location **8004**.

```
Memory[8004] <- $t0
```

دستورات کنترلی

این دستورات سیر اجرای برنامه را تغییر میدهند یعنی اینکه دستور بعدی که باید اجرا شود را تعیین میکنند.

انواع مختلف دستورات کنترلی

- Conditional branches
- Jumps (unconditional branch)
- Procedure calls
- Procedure returns

دستورات انشعاب شرطی

- دستورات پرش شرطی در MIPS عبارتند از:

```
bne $t0, $t1, Label
beq $t0, $t1, Label
```

- Example: if (i==j) h = i + j;

```
        bne $s0, $s1, Label
        add $s3, $s0, $s1
Label:     ....
```

- Note the reversal of the condition from equality to inequality!

دستورات پرش غیر شرطی

- دستورات انشعاب غیرشرطی در MIPS عبارتند از:

`j label` Unconditional jump

`jr $t0` “jump register”. Jump to the instruction specified in register $t0

- Example:

```
if (i!=j)                beq $s4, $s5, Lab1
    h=i+j;               add $s3, $s4, $s5
else                     j Lab2
    h=i-j;       Lab1:   sub $s3, $s4, $s5
                 Lab2:   ...
```

دستورات کنترلی دیگر

- اغلب در کنار دو دستور فوق از دستوردیگری نیز استفاده میشود:
-

slt and slti // set if less than (w/ and w/o an immediate)

- slt $t0, $s1, $s2

```
if  $s1 < $s2 then
            $t0 = 1
else
            $t0 = 0
```

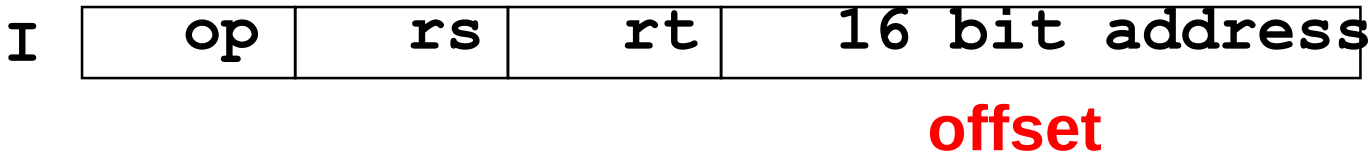
- از این دستور به همراه دستورات قبلی استفاده میشود:

```
slti $a1, $a0, 5    # $a1 = 1 if $a0 < 5
bne        $a1, $0, Label        #Branch if $a0 < 5
```

نحوه محاسبه آدرس در دستورات کنترلی

bne $t4,$t5,Label Next instruction is at Label if $t4 ≠ $t5

beq $t4,$t5,Label Next instruction is at Label if $t4 = $t5

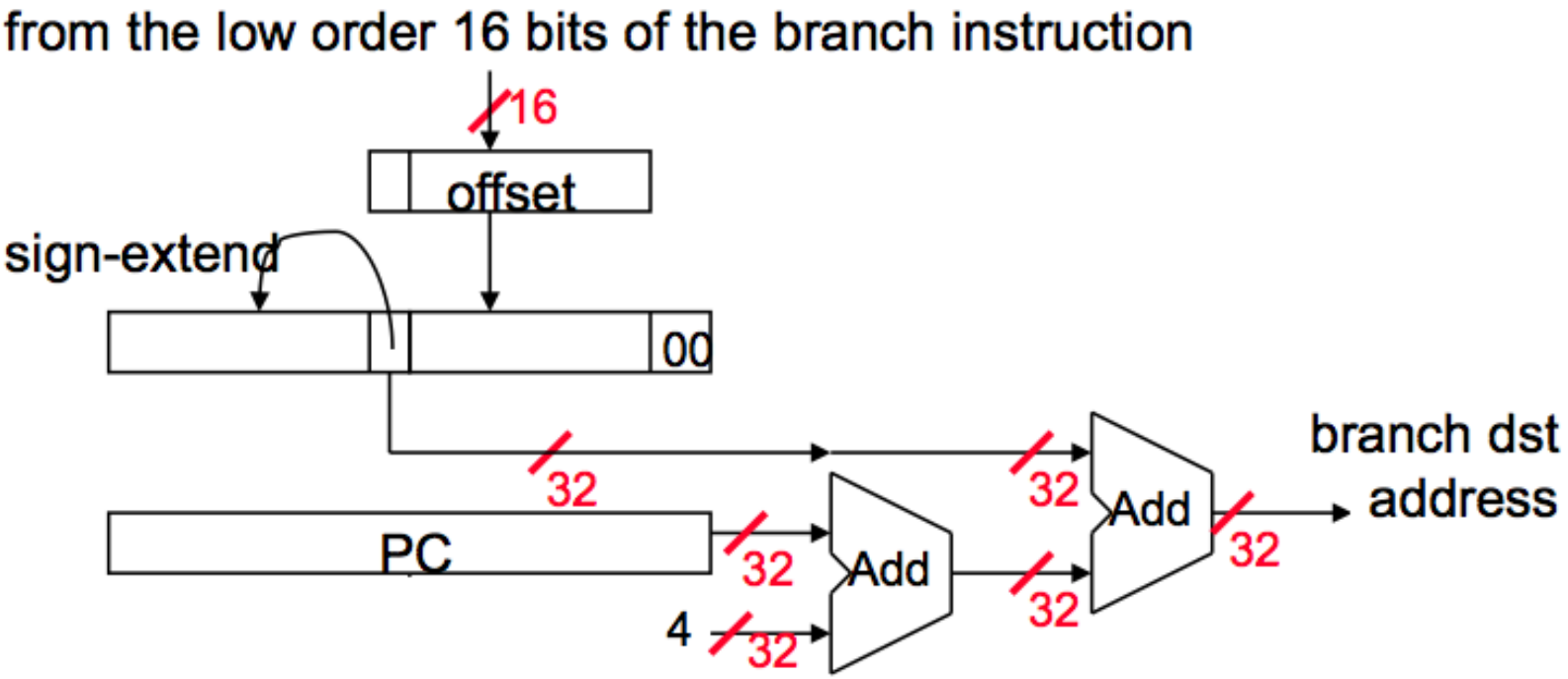


- PC-relative addressing

آفست بصورت یک مقدار ۱۶ بیتی علامت دار نوشته میشود. این مقدار پس از توسعه بیت علامت و تبدیل آن به ۳۲ بیت و سپس دوبیت شیفت به چپ به مقدار PC+4 اضافه میشود.

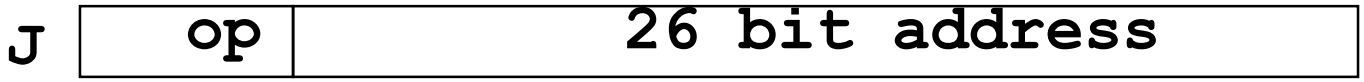
نحوه محاسبه آدرس در دستورات کنترلی

مقصد پرش ← PC+4 + SignExt (L)*4



نحوه محاسبه آدرس در دستورات کنترلی

- j Label Next instruction is at Label



- Pseudodirect addressing
  - در این دستور فقط از بیت های با ارزش PC استفاده میشود.

32-bit jump address = 4 Most Significant bits of PC concatenated with 26-bit word address (or 28- bit byte address)

Address boundaries of 256 MB

For larger distances: Jump register **jr** required.

مثال

```
LOOP:   mult   $9, $19, $10    # R9 = R19*R10
        lw     $8, 1000($9)    # R8 = @(R9+1000)
        bne    $8, $21, EXIT
        add    $19, $19, $20   #i = i + j
        j      LOOP
EXIT:   ......
```

- Assume LOOP is placed at location 80000

	op	rs	rt			
80000	0	19	10	9	0	24
80004	35	9	8	1000		
80008	5	8	21	8		
80012	0	19	20	19	0	32
80016	2	80000				
80020	...					

دستورات شیفت

MIPS Shift Instructions

- **Format: op_code dest reg src reg shift amount**

```
sll  $t4, $t0, 5          #shift left logical 5 bits (multiply by 32)
sra $t5, $t0, 2           #shift right arithmetic 2 bits (divide by 4),
                          #sign extended
srl $v0, $t0, 1           #shift right logical 1 bit. Sign bit is now 0
srlv $v0, $t0, $t1        #shift right logical, $t1 says how many bits
```

Type of shift	Left	Right
Logical	sll sllv	srl srlv
Arithmetic	(none: use sll)	sra srav
Rotate	rol	ror

مثالی از ترجمه یک برنامه C

```
swap(int v[], int k);
{ int temp;
   temp = v[k]
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

```
swap:
    muli  $2 , $5, 4
    add   $2 , $4, $2
    lw    $15, 0($2)
    lw    $16, 4($2)
    sw    $16, 0($2)
    sw    $15, 4($2)
    jr    $31
```

Explanation:

index k : $5

base address of v: $4

address of v[k] is $4 + 4.$5

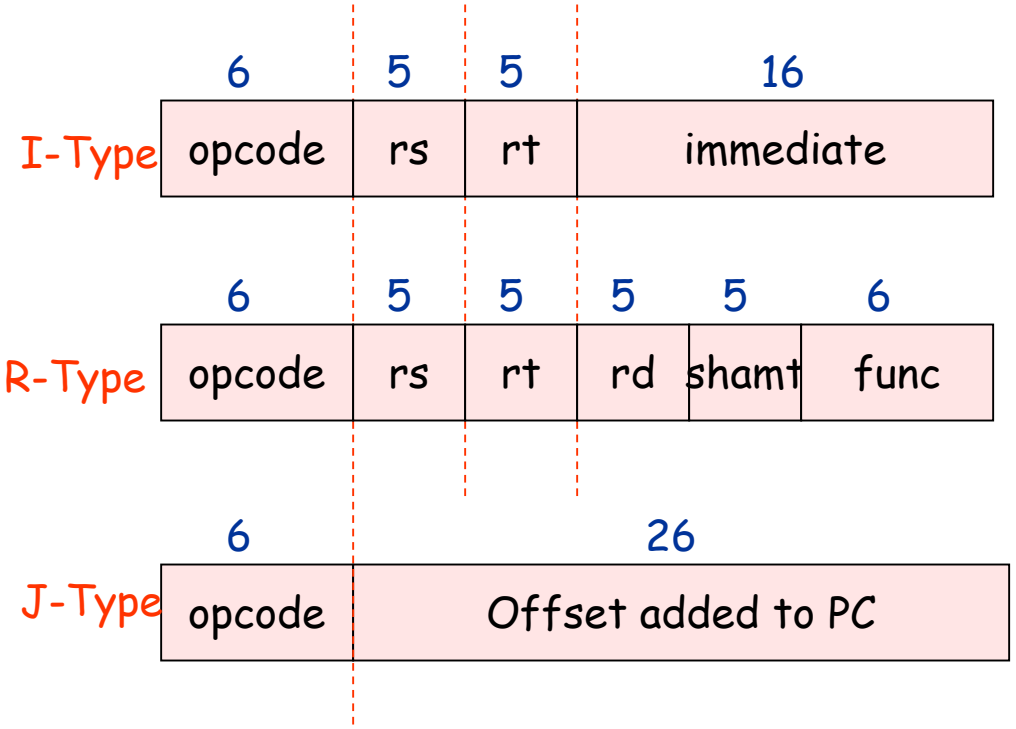
فرمت دستورات MIPS

- طول هر دستور ۳۲ بیت است.
- هر دستور به تعدادی Field تقسیم میشود. هر فیلد توضیحی در مورد دستور العمل ارائه میدهد.
- از آنجائیکه دستورات مختلف نیازمند ارائه توضیحات مختلفی هستند لذا در MIPS سه نوع فرمت مختلف (ولی با طول یکسان) برای دستورات در نظر گرفته شده است.

- **R-format** Register instructions are used for register based ALU operations.
- **I-format** Immediate instructions, can be either Load/Store operations, Branch operations, or Immediate ALU operations.
- **J-format** Jump instructions, devote all of the non-opcode space to a 26-bit jump destination field.

فرمت دستورات MIPS

۳ نوع فرمت مختلف بصورت زیر هستند:

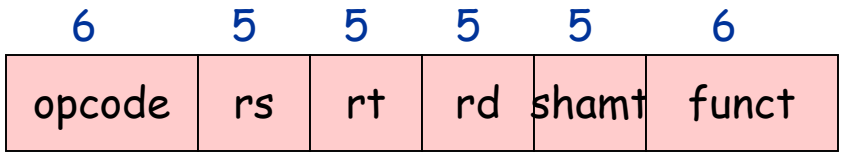


Note the regularity of instruction encoding. This is important for implementing an efficient pipelined CPU.

فیلد های مختلف دستورالعمل ها

- **op** operation of the instruction
- **rs** first register source operand
- **rt** second register source operand
- **rd** register destination operand
- **shamt** shift amount
- **funct** function (select type of ALU operation)
  - add = 32
  - sub = 34

دستورات R-Type



- rs: source register 1
- rt: source register 2
- rd: destination register
- shamt: shift amount
- funct: specific variant of opcode

- این دستورات دارای opcode=0 بوده و برای عملیات ALU استفاده میشوند.
- عمل مورد نظر توسط فیلد funct مشخص میشود:

add: 32

sub: 34

مثال

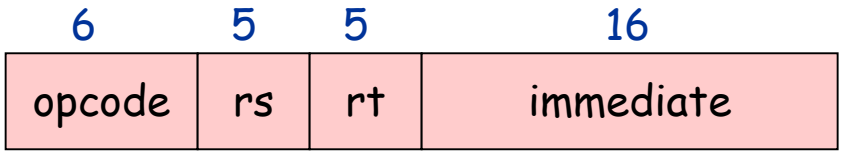
- Example: `add $t0, $s1, $s2`
- registers have numbers, `$t0=8, $s1=17, $s2=18`

- Instruction Format:

000000	10001	10010	01000	00000	100000

op	rs	rt	rd	shamt	funct

I-Type دستورات



I-Type Format

- Opcode
  - lw: 35 (100011)
  - sw: 43 (101011)

  } Similarity in opcode for lw & sw simplifies hardware
  - opcode value differentiates I- and R-Type instructions
- rs: base register
- rt:
  - destination register for lw
  - source register for store
- immediate: offset from base range: -2^{15} to $(2^{15}-1)$

مثال

```
lw $s0, 24($t1)
```

100011 01001 10000 0000000000011000

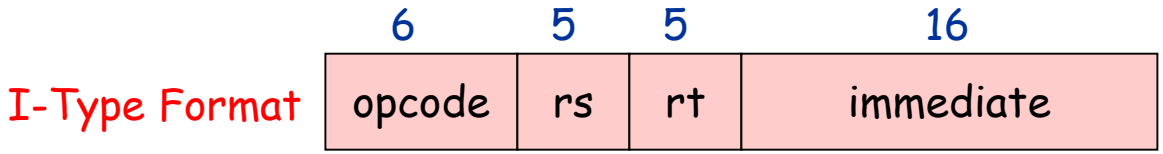
op	rs	rt	address

```
sw $s0, 24($t1)
```

101011 01001 10000 0000000000011000

op	rs	rt	address

سایر دستورات I-Type

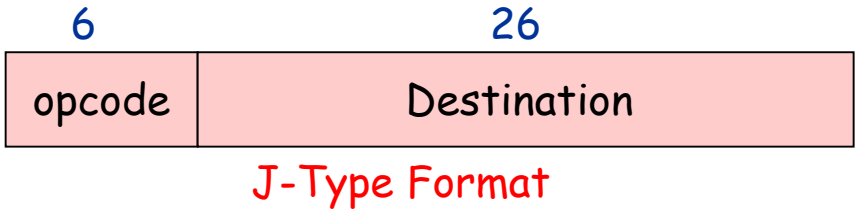


- برای دستورات ALU که نیاز به یک اپراند ثابت دارند استفاده میشود. (e.g., X=X+4)
- مقادیر ثابت بصورت یک عدد ۱۶ بیتی کد میشوند. لذا در رنج -2^{15} to $(2^{15}-1)$ خواهند بود.
- مثال:

addi R4, R8, 79

slti R1, R2, 56: sets Reg[R1]=1 if Reg[R2]<56 else Reg[R1]=0

دستورات J-Type



J-Type Format

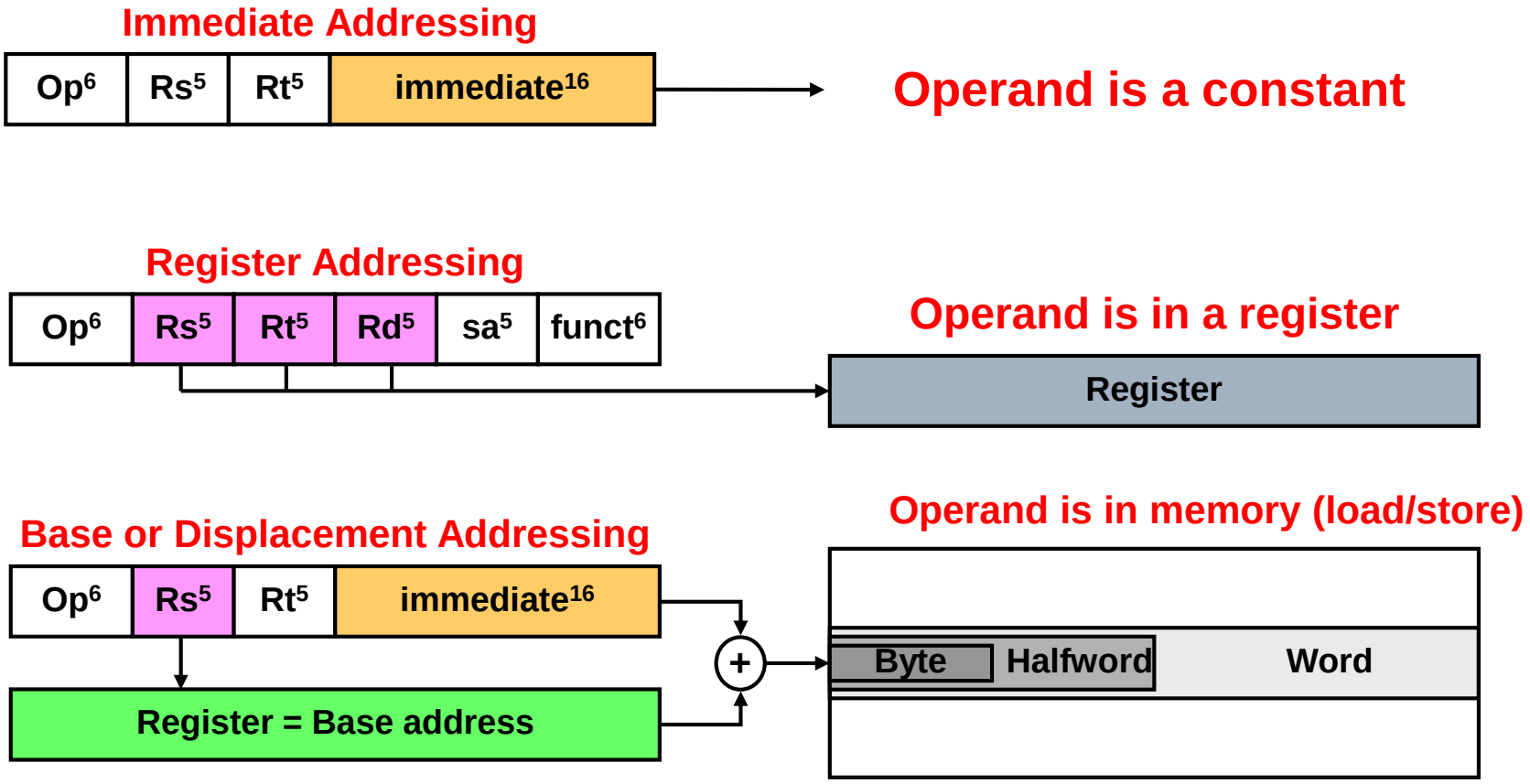
- دستورات پرش غیرشرطی به فرم J-Type کد میشوند.
- اپکد دستور J برابر با ۲ و اپکد دستور Jal برابر با ۳ می باشد
- آدرس مقصد پرش به صورت زیر به دست می آید:

چهار بیت پرارزش PC	۲۶ بیت فوق	00

مد های آدرس دهی MIPS

- Immediate
  - add R4,R3 #7 # Regs[R4]=Regs[R3]+7
  - 16-bit field for the constant
- Register
  - add R4, R3, R2 # Regs[R4]=Regs[R3]+Regs[R2]
- Displacement
  - lw R4, 100(R1) #regs[R4]=Regs[R4]+Mem[Regs[R1]+100]
  - 16-bits for displacement
- Special cases of displacement mode
  - indirect mode: displacement value=0
    - lw R4, 0(R1) #regs[R4]=Regs[R4]+Mem[Regs[R1]]
  - absolute addressing : R0 as base register (always stores 0)
    - lw R4, 8769(R0)

مد های آدرس دهی MIPS




Questions