معماری کامپیوتر (۱۳۹۰-۱۱-۱۳)
جلسه‌ی پنجم

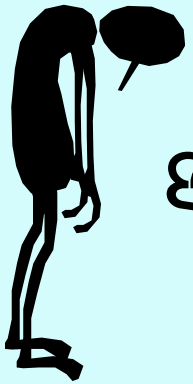
دانشگاه شهید بهشتی
دانشکده‌ی مهندسی برق و کامپیوتر
زمستان ۱۳۹۰
احمد محمودی ازناوه

فهرست مطالب

- شکل‌های مختلف دستور
- شیوه‌های آدرس‌دهی

شیوه‌های آدرس‌دهی

- حالتی را تصور کنید که به یک داده‌ی ثابت سی‌ودو بیتی نیاز داشته باشیم!
- یا بخواهیم به یک آدرس سی‌ودو بیتی دسترسی پیدا کنیم!
- چه راه حلی پیشنهاد می‌دهید؟
- برای نمونه بخواهیم عدد ۸۱۹۴۳۰۴ را در جمع ثابت استفاده کنیم.



0000 0000 0111 1101 0000 1001 0000 0000

ثابت‌های سی‌ودوبیتی

- بیشتر ثابت‌هایی که مورد استفاده قرار می‌گیرند، کوچک هستند و در شانزده بیت می‌گنجند.
- به ندرت مواردی پیش می‌آید که به ثابت‌هایی بزرگ نیاز داشته باشیم.
- در این موارد می‌توان داده‌ی مورد نیاز را در ثبات بارگذاری نمود و از دستورات که دارای عملوند ثبات هستند، استفاده کرد.
- برای این منظور دستور زیر پیش‌بینی شده است:

```
lui rt, constant
```

load upper immediate

این دستور، مقدار ثابت را در شانزده بیت پرارزش قرار می‌دهد و بخش کم ارزش را صفر می‌کند.

ثابت‌های سی‌ودوبیتی (ادامه...)

```
lui $s0, 61
```

0000 0000 0111 1101	0000 0000 0000 0000

برای قرار دادن بخش کم ارزش چه پیشنهادی دارید؟

نظر شما درباره‌ی دستور addi چیست؟

```
ori $s0, $s0, 2304
```

0000 0000 0111 1101	0000 1001 0000 0000

The machine language version of `lui $t0, 255` `# $t0 is register 8:`

001111	00000	01000	0000 0000 1111 1111

Contents of register `$t0` after executing `lui $t0, 255`:

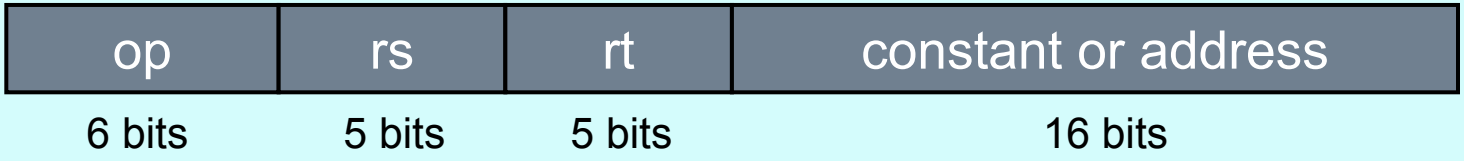
0000 0000 1111 1111	0000 0000 0000 0000

ثابت‌های سی‌ودوبیتی (ادامه...)

- ثابت‌های بزرگ توسط کامپایلر و یا اسمبلر به اعداد کوچک‌تر شکسته شده و در یک ثبات جمع‌آوری می‌شوند.
- کوچک بودن اندازه‌ی فیلد داده‌های ثابت، برای آدرس‌ها و به ویژه در دستورات lw و st ایجاد دشورای می‌کند.
- برای جمع‌آوری آدرس، اسمبلر به یک ثبات موقت نیاز دارد. ثبات $at برای این منظور در نظر گرفته می‌شود.

آدرس‌دهی در دستورات پرش شرطی

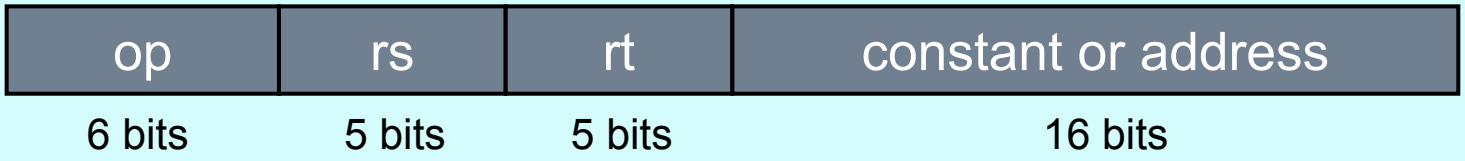
- دستورات پرش شرطی از نوع I هستند.
- آدرس شانزده‌بیتی میزان پرش را محدود می‌کند. در این حالت، طول برنامه برای کاربردهای امروزی معقول نیست.
- معمولاً محدوده‌ی مقصد پرش نزدیک محل دستور پرش است.



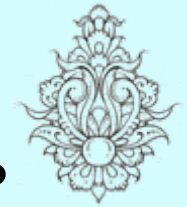
PC = a register + branch address

آدرس‌دهی نسبی بر مبنای PC

- Target address = PC + offset × 4



- بدین‌ترتیب، طول برنامه می‌تواند تا چهار گیگابایت افزایش یابد.
- هنگام اجرای هر دستور، PC به آدرس بعدی اشاره می‌کند.
- آدرسی که در دستورات پرش استفاده می‌شود، آدرس کلمه است، نه آدرس بایت

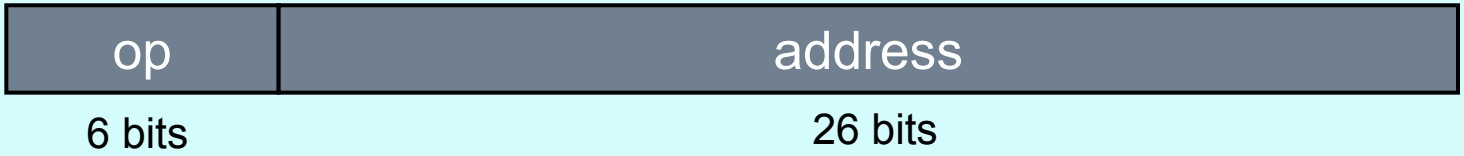


آدرس‌دهی شبه‌مستقیم

```
j L1
```

```
jal ProcedureLabel
```

- این دستورات از کدام نوع هستند؟
  - نوع I؟
- آخرین نوع دستورها در MIPS، نوع J می‌باشد:




```
Target address = PC31…28 : (address × 4)
```

مثال

```
while (save[i] == k) i += 1;
```

```
Loop: sll  $t1, $s3, 2
      add  $t1, $t1, $s6
      lw   $t0, 0($t1)
      bne  $t0, $s5, Exit
      addi $s3, $s3, 1
      j    Loop
Exit: …
```

80000	0	0	19	9	4	0
80004	0	9	22	9	0	32
80008	35	9	8	0		
80012	5	8	21	2		
80016	8	19	19	1		
80020	2	20000				
80024						

پیاده‌سازی پرش شرطی

- در صورتی که نیاز به پرش شرطی داشتیم که فاصله‌ی نسبی آن با آدرس فعلی به بیش از شانزده بیت نیاز داشت،چه باید کرد؟

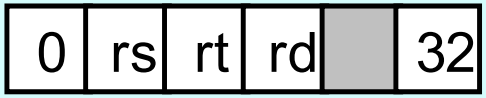
```
beq $s0,$s1, L1
```

مثلاً آدرس L1 خیلی دور است!

```
    bne $s0,$s1, L2
    j L1
L2:     …
```

البته زحمت انجام این کار به عهده‌ی اسمبلر است!

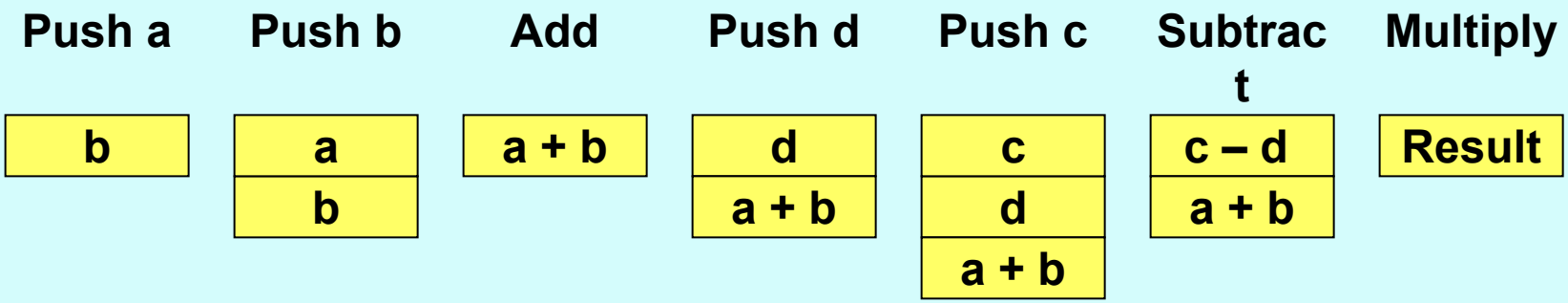
شکل‌های مختلف دستور

Category	Format	Opcode
0-address		`syscall`
1-address		`j`
2-address		`mult`
3-address		`add`

مثالی دستورات بدون عملوند

مثال: ارزیابی عبارت

$(a + b) \times (c - d)$



Reverse Polish string: b a + d c – ×

postfix notation

مثالی معماری تک عملوند

مثال: ارزیابی عبارت

$$(a + b) \times (c - d)$$

انباره ثباتی است که یکی از عملوندها را در خود نگه می‌دارد و همیشه پاسخ در آن ذخیره می‌شود.

```
Load      a
add       b
Store     t
load      c
subtract  d
multiply  t
```

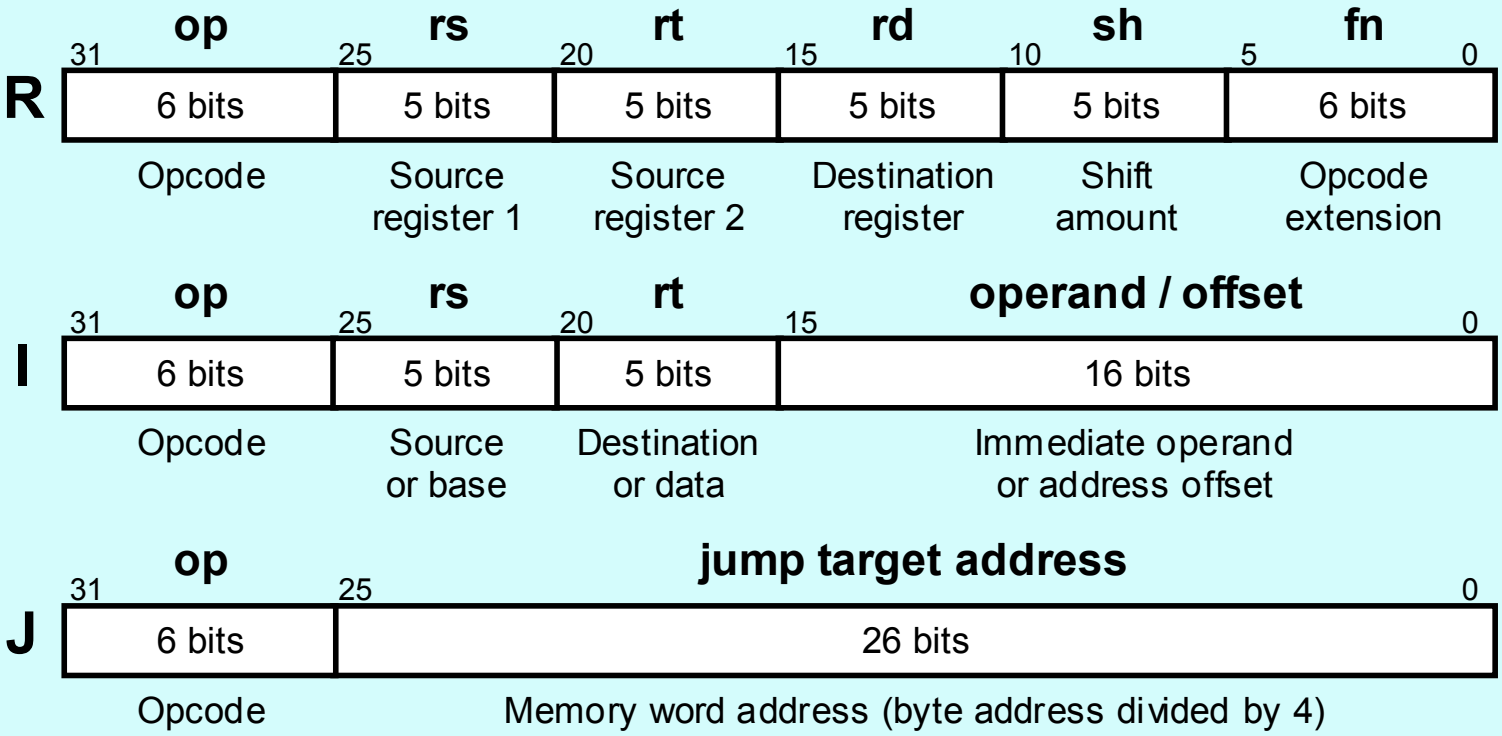
مثالی معماری دستورالعمل با دو عملوند

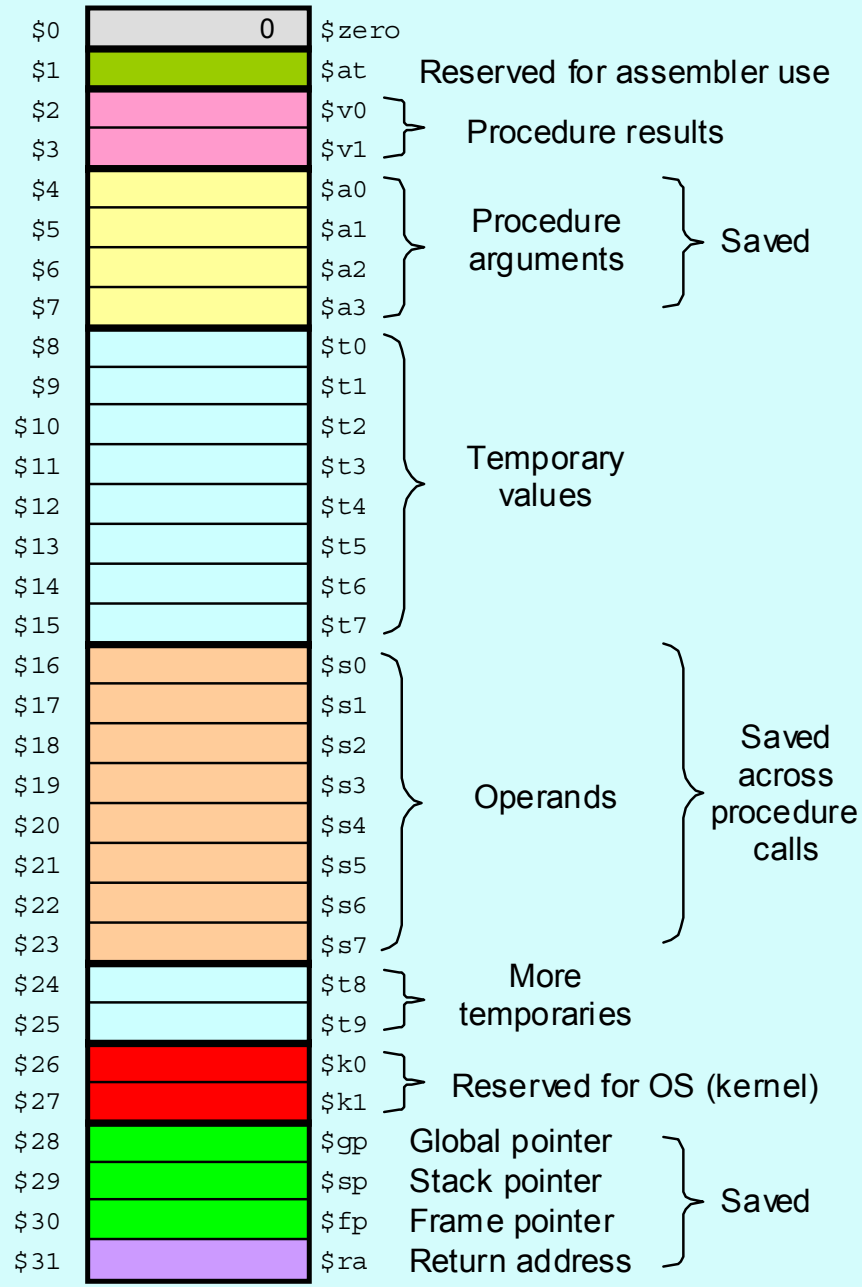
مثال: ارزیابی عبارت

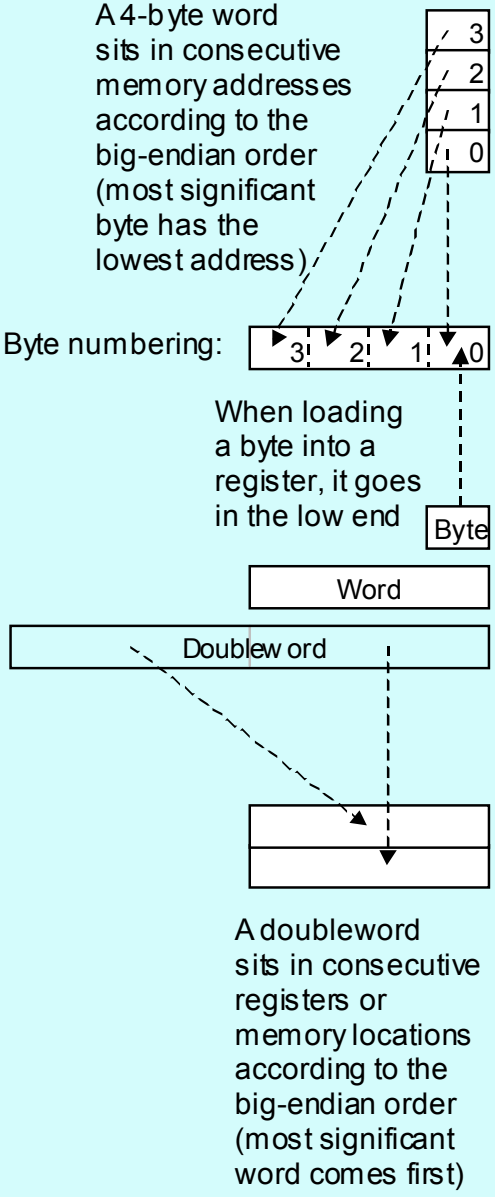
$$(a + b) \times (c - d)$$

```
load      $1,a
add       $1,b
load      $2,c
subtract  $2,d
multiply  $1,$2
```

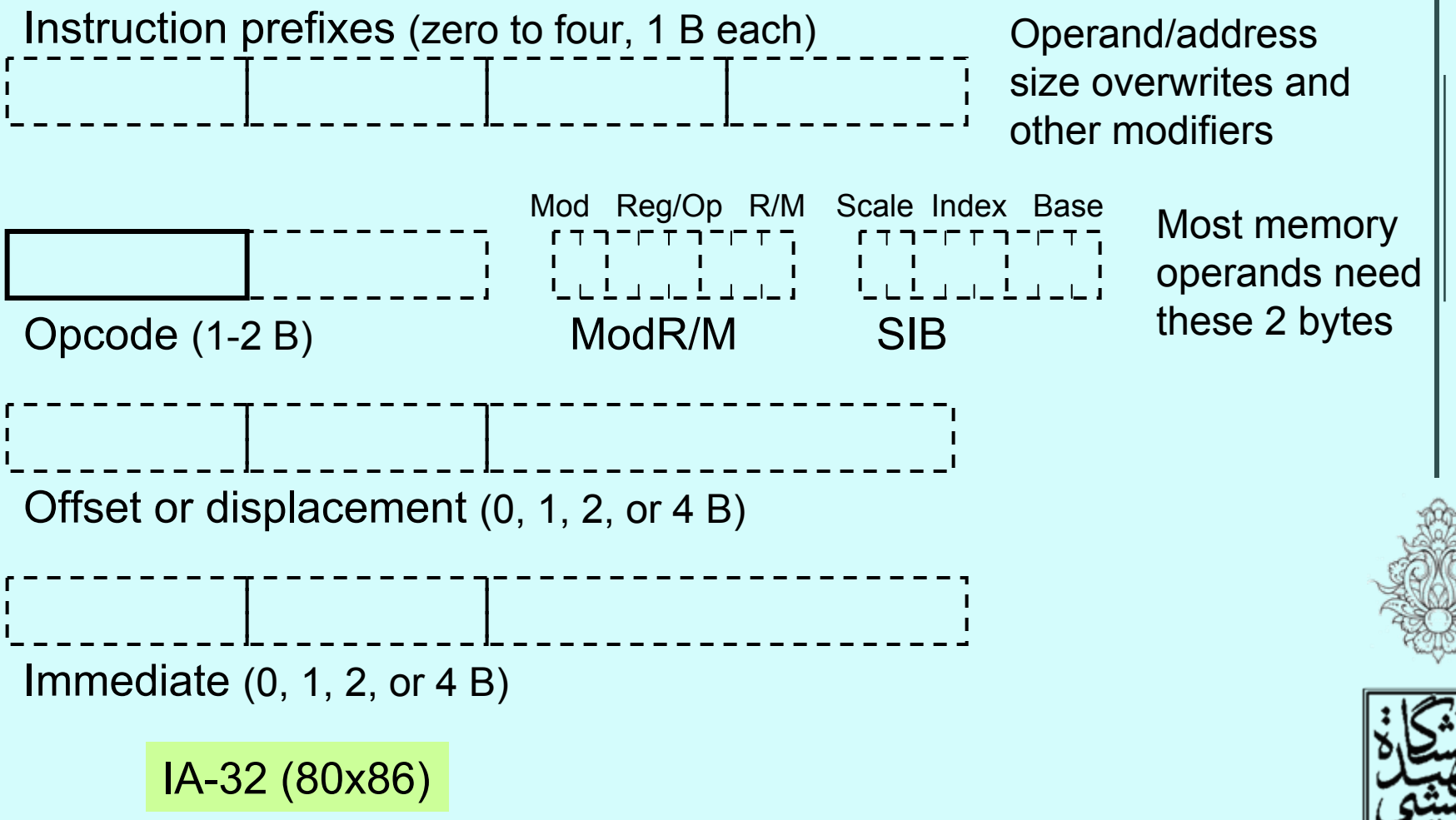
قالب‌های دستور در یک نگاه



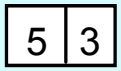
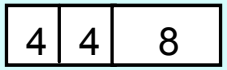
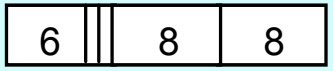
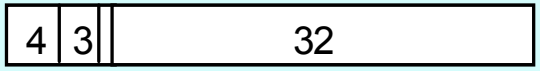

$0 0 $zero
$1 $at Reserved for assembler use
$2 $v0
$3 $v1 Procedure results
$4 $a0
$5 $a1
$6 $a2
$7 $a3 Procedure arguments
Saved
$8 $t0
$9 $t1
$10 $t2
$11 $t3
$12 $t4
$13 $t5
$14 $t6
$15 $t7 Temporary values
$16 $s0
$17 $s1
$18 $s2
$19 $s3
$20 $s4
$21 $s5
$22 $s6
$23 $s7 Operands
Saved across procedure calls
$24 $t8
$25 $t9 More temporaries
$26 $k0
$27 $k1 Reserved for OS (kernel)
$28 $gp Global pointer
$29 $sp Stack pointer
$30 $fp Frame pointer
$31 $ra Return address
Saved


A 4-byte word sits in consecutive memory addresses according to the big-endian order (most significant byte has the lowest address)
3
2
1
0
Byte numbering:
3 2 1 0
When loading a byte into a register, it goes in the low end
Byte
Word
Doubleword
A doubleword sits in consecutive registers or memory locations according to the big-endian order (most significant word comes first)

مثالی ازیک ساختار پیچیده

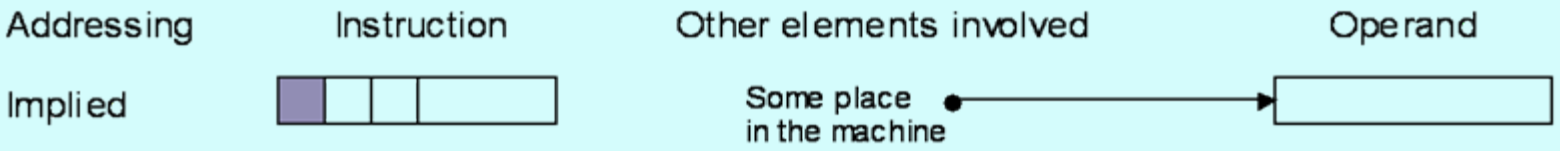


مثال

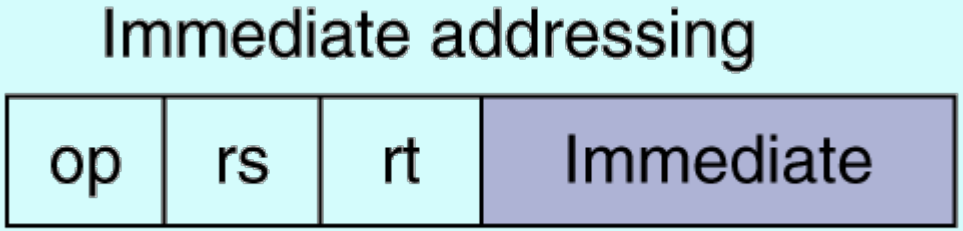
Type	Format (field widths shown)	Opcode	Description of operand(s)
1-byte		PUSH	3-bit register specification
2-byte		JE	4-bit condition, 8-bit jump offset
3-byte		MOV	8-bit register/mode, 8-bit offset
4-byte		XOR	8-bit register/mode, 8-bit base/index, 8-bit offset
5-byte		ADD	3-bit register spec, 32-bit immediate
6-byte		TEST	8-bit register/mode, 32-bit immediate

implied addressing

- **نشانی‌دهی ضمنی:**
  - عملوند آشکارا گفته نمی‌شود، هنگامی اجرا به صورت ضمنی برای پردازنده مشخص می‌شود.



- **نشانی‌دهی بی‌واسطه:**
  - داده‌ی ثابت به صورت مستقیم مورد استفاده قرار می‌گیرد.

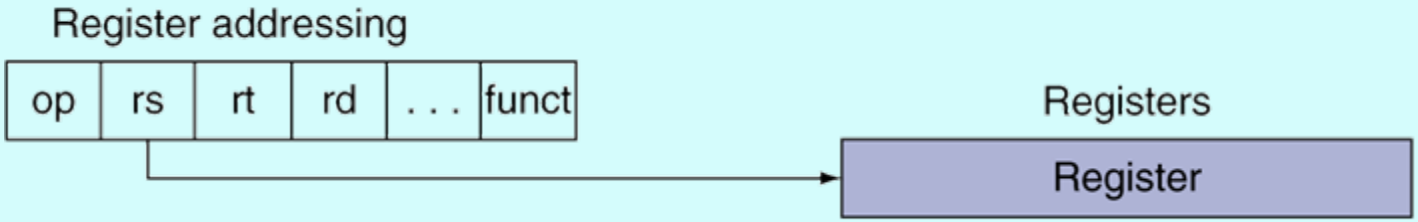


Operand=A

Immediate/literal/constant addressing

انواع نشانی‌دهی (ادامه...)

- **آدرس‌دهی ثبات:**
  - **در فیلد آدرس شماره ثبات مورد نظر آورده می‌شود.**



Register addressing

با توجه به محدودیت ثبات، فیلد آدرس کوچک خواهد بود
روش سریعی می‌باشد که پیاده‌سازی راحتی هم دارد

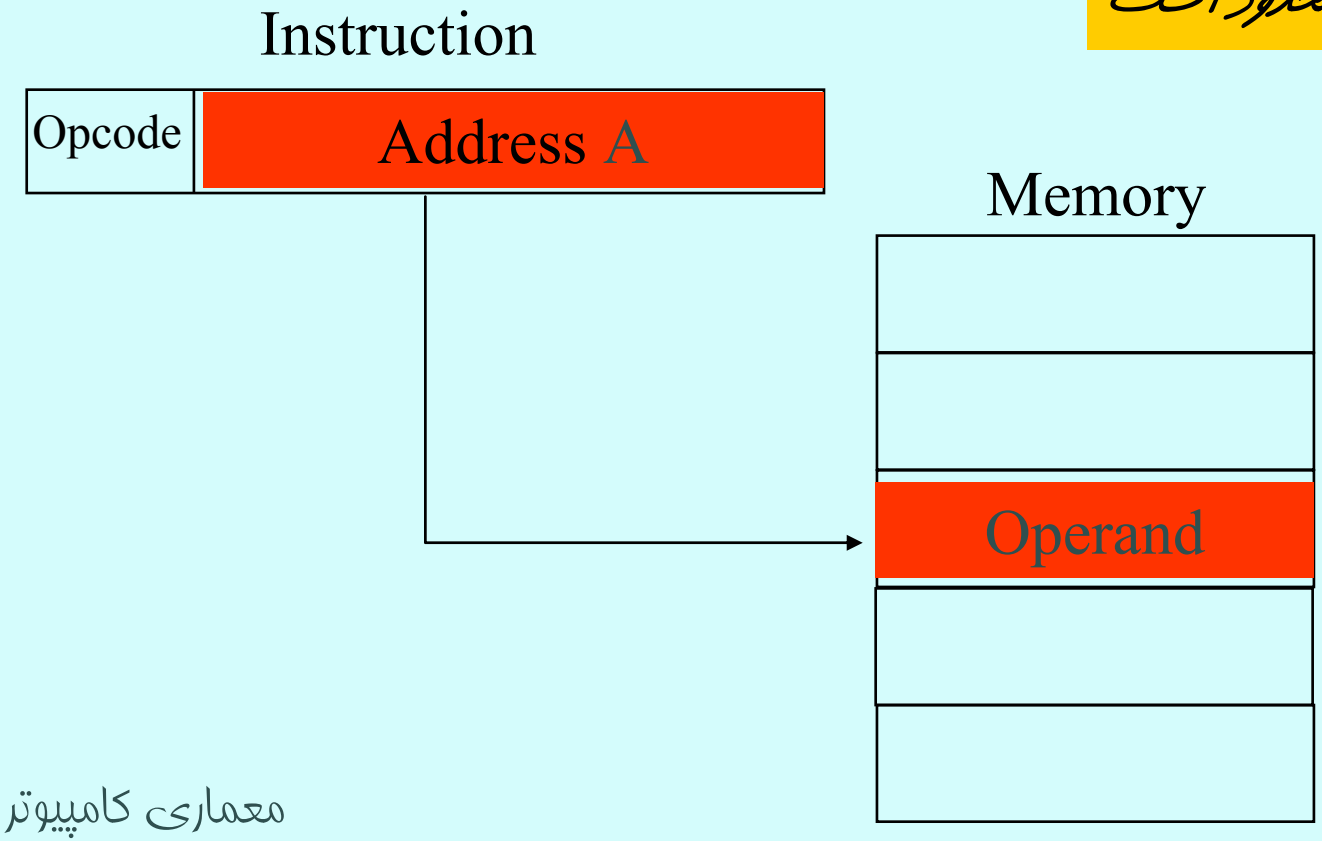
$EA = R$

Effective address

انواع نشانی‌دهی (ادامه...)

- **آدرس‌دهی مستقیم:**
  - در فیلد آدرس آدرس خانه‌ی حافظه آورده می‌شود.

فضای آدرس‌دهی محدود است

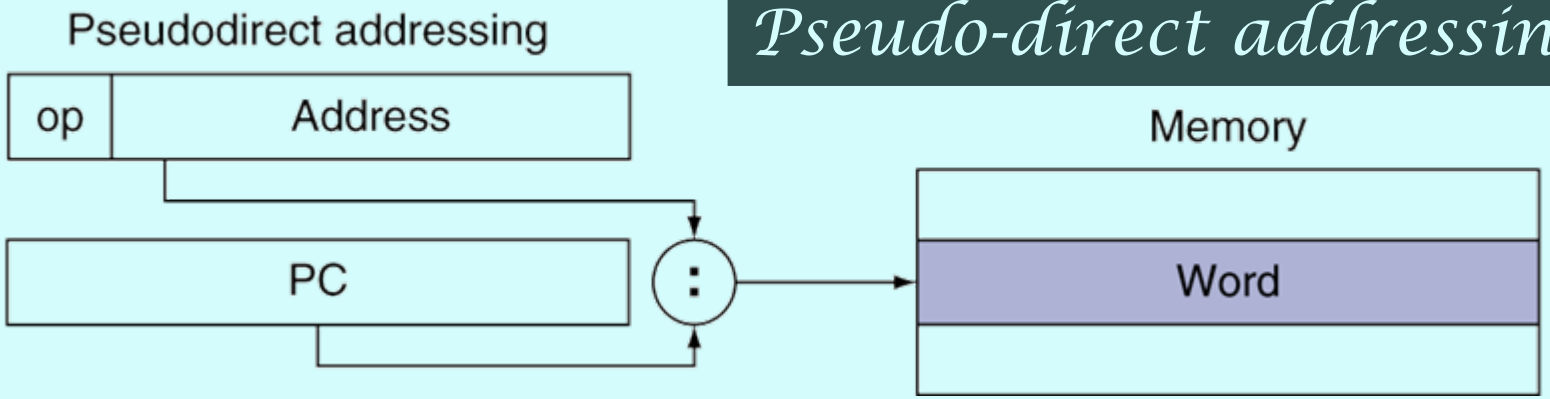


EA = A

انواع نشانی‌دهی (ادامه...)

- **آدرس‌دهی شبه مستقیم:**
  - بخشی از بیت‌های آدرس از PC برداشته می‌شود.

Pseudo-direct addressing

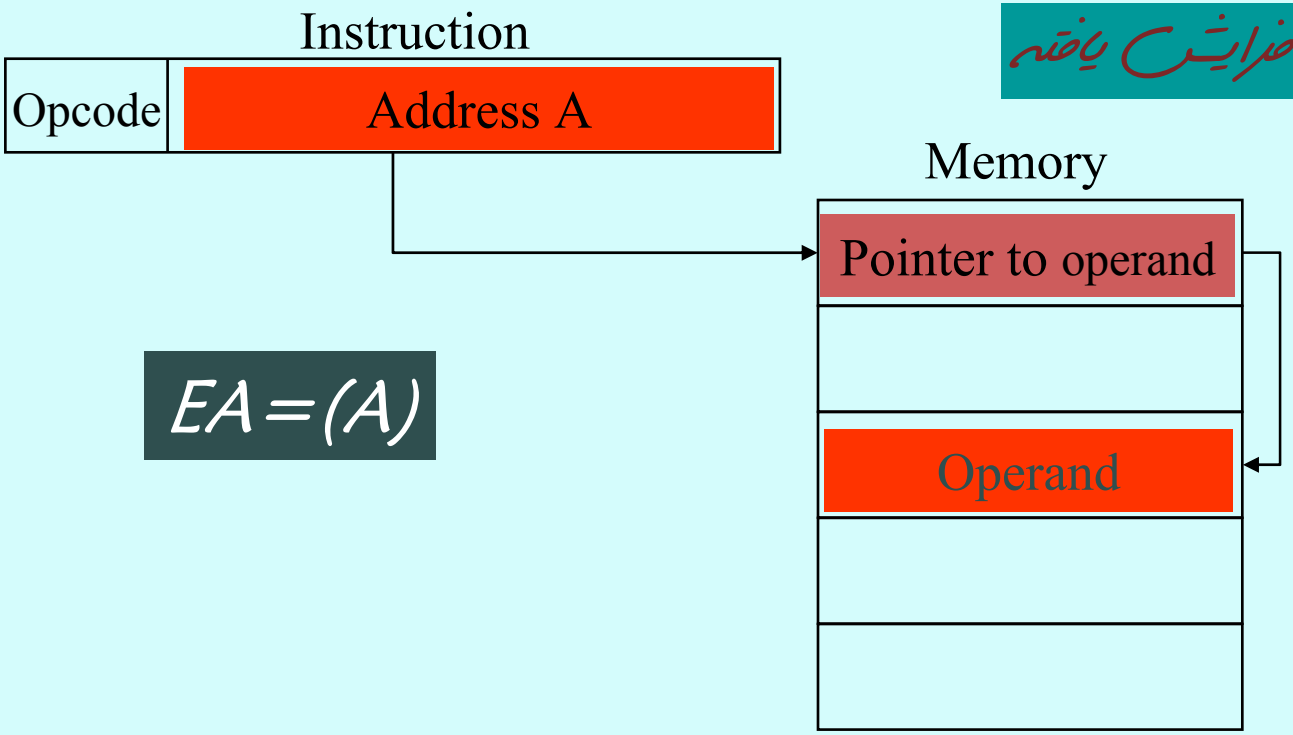


برخی منابع آن را augmented addressing هم می‌نامند.

انواع نشانی‌دهی (ادامه...)

- **آدرس‌دهی غیرمستقیم:**
  - در فیلد آدرس، آدرس خانه‌ای از حافظه آورده می‌شود که حاوی آدرس عملوند است.

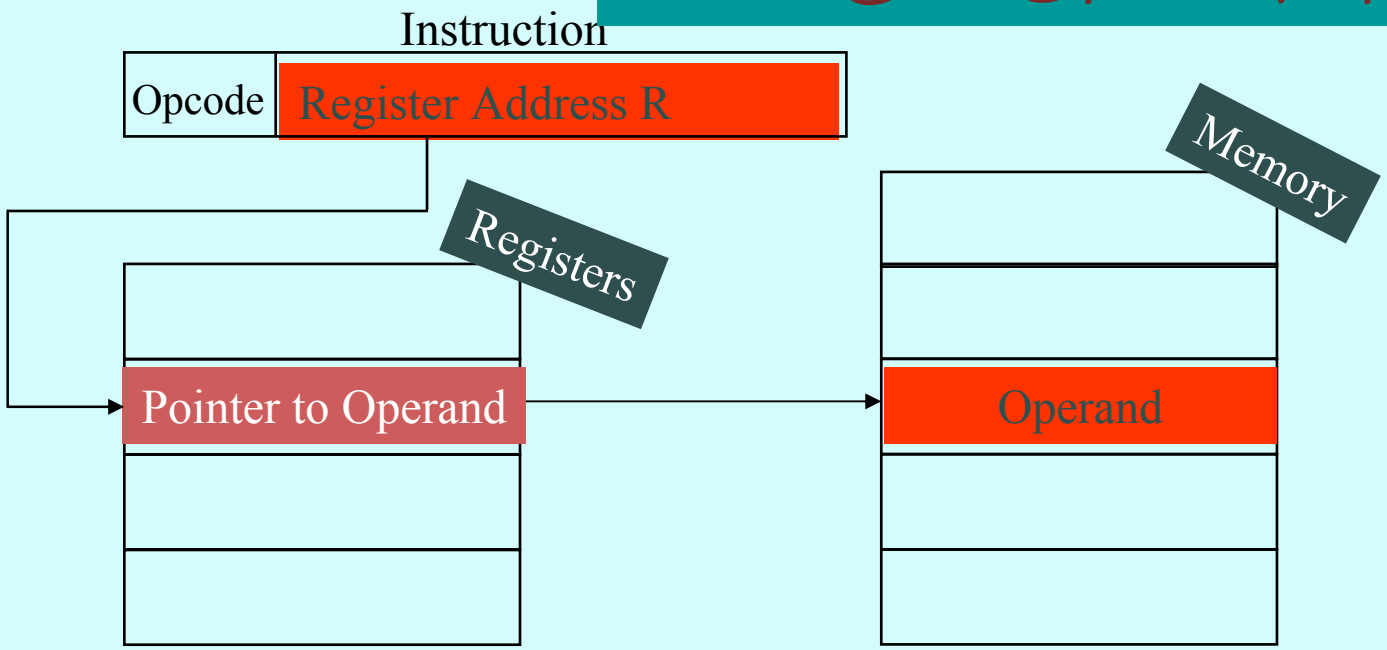
فضای آدرس‌دهی افزایش یافته



انواع نشانی‌دهی (ادامه...)

- آدرس‌دهی غیرمستقیم از طریق ثبات: در این شیوه آدرس خانه‌ی حافظه در یک ثبات قرار دارد.

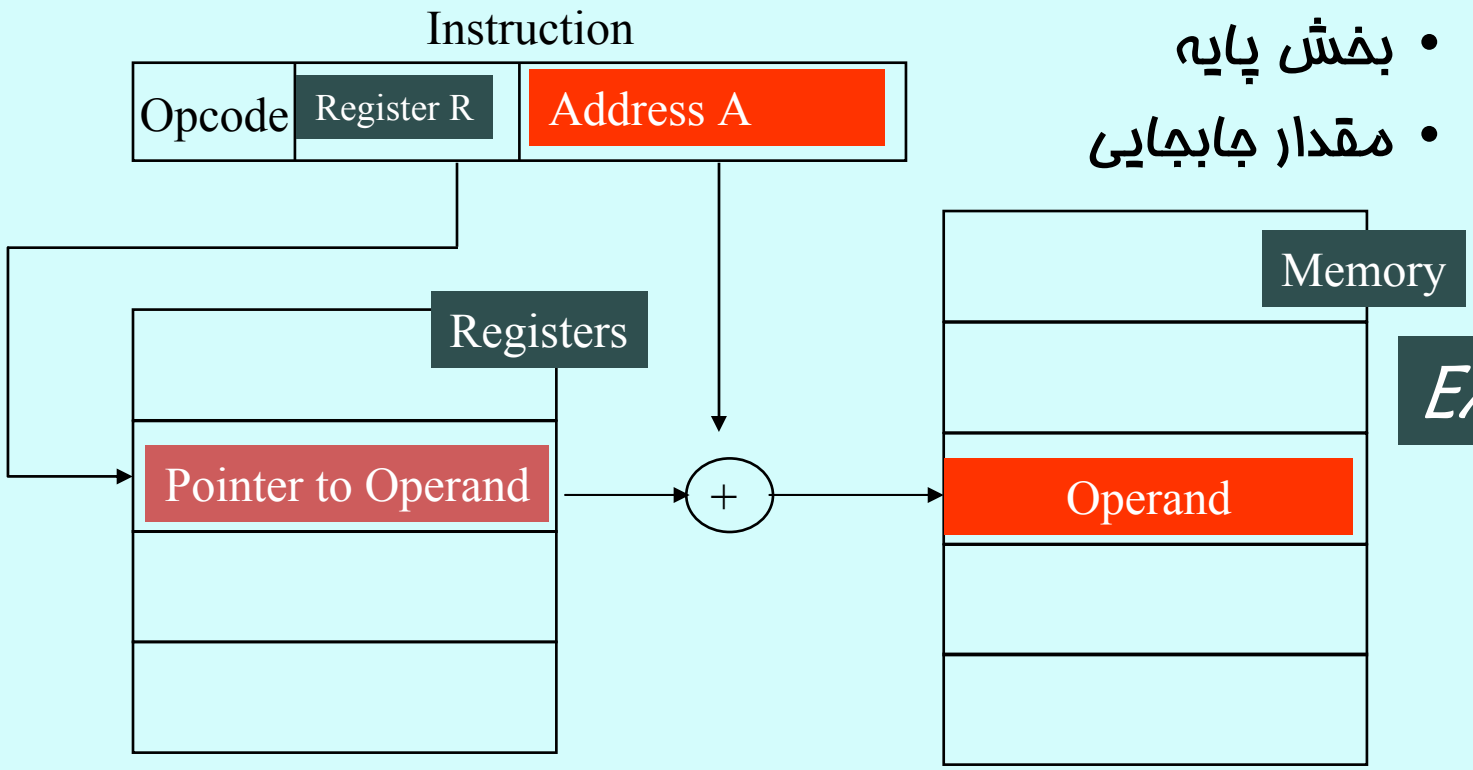
فضای آدرس‌دهی بزرگ است
مراجعه به حافظه نسبت به حالت قبل کاهش یافته است



$EA = (R)$

انواع نشانی‌دهی (ادامه...)

- آدرس‌دهی بر اساس آدرس پایه یا جانشین‌سازی:
  - آدرس از دو بخش تشکیل شده است:
    - بخش پایه
    - مقدار جابجایی

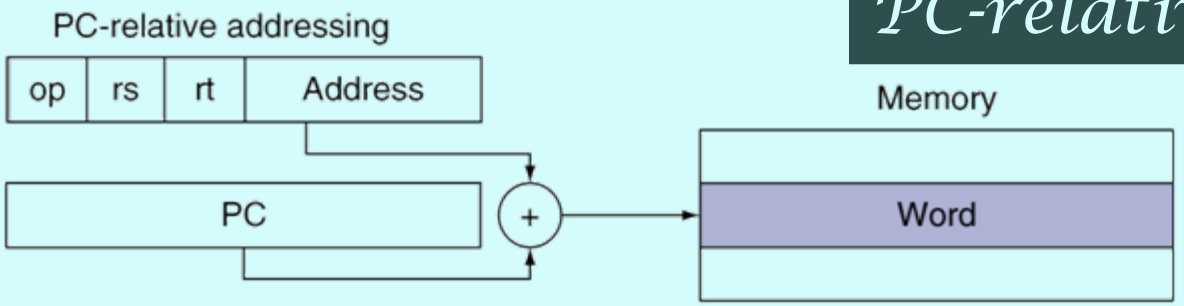


$EA = A + (R)$

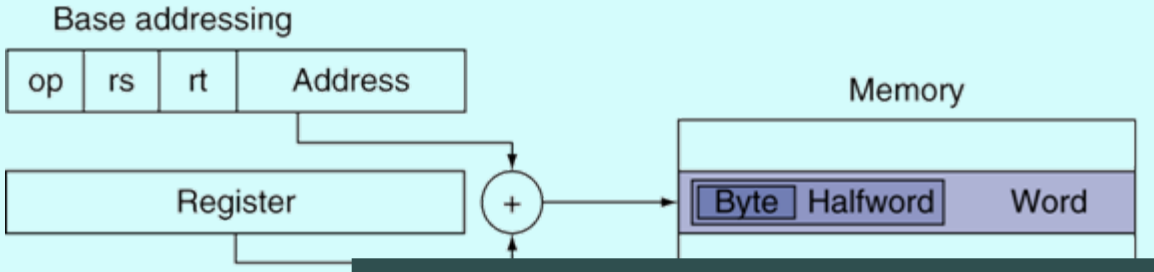
آدرس دهی نسبی (Relative) : نوع خاصی از این نوع آدرس دهی است که PC را به عنوان بخش پایه استفاده می‌کند.

انواع نشانی‌دهی (ادامه...)

- آدرس دهی نسبی (نسبت به PC):
  - مانند آدرس‌های پرش شرطی که نسبت به آدرس دستور بعدی سنجیده می‌شوند



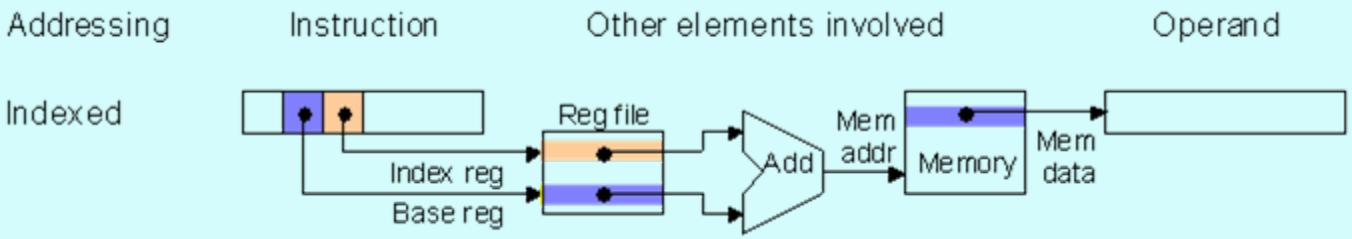
PC-relative addressing



Base or displacement addressing

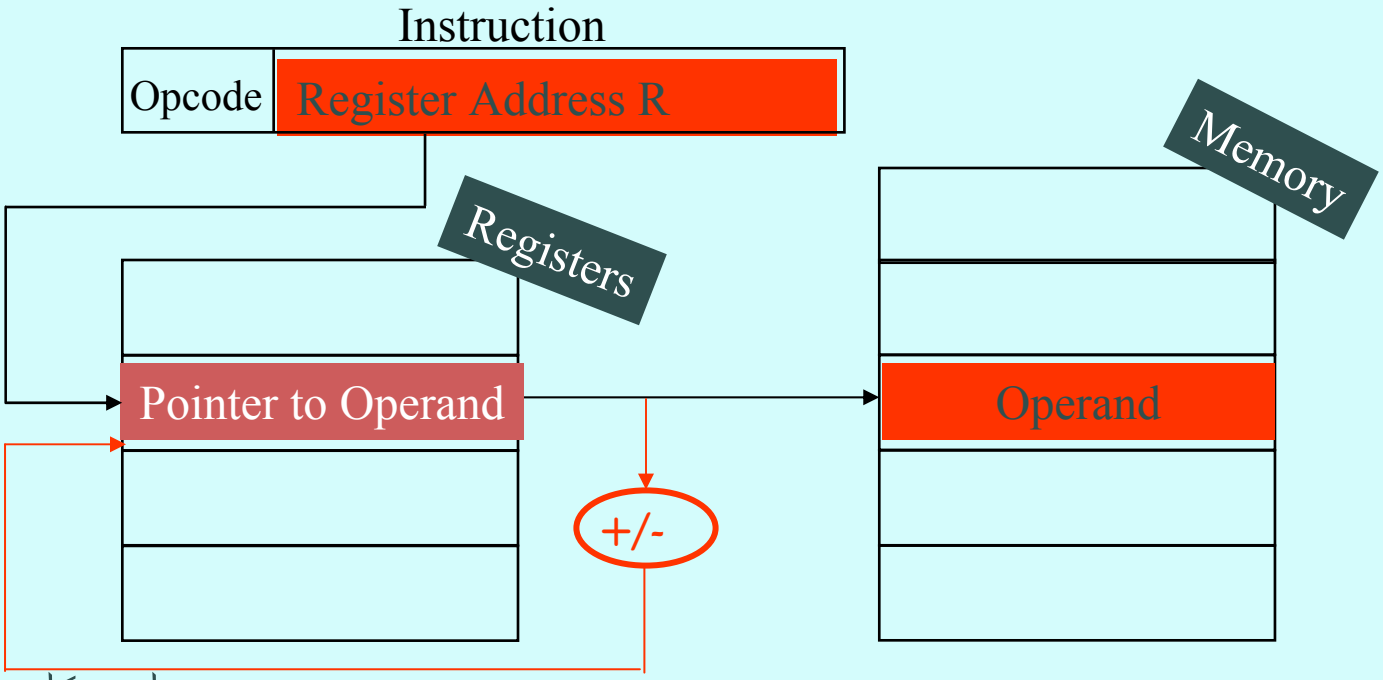
Indexed Addressing

- **آدرس‌دهی شاخص:**
  - شبیه‌آدرس‌دهی بر اساس آدرس پایه است، با این تفاوت که بخش آدرس، ابتدای آدرس بخشی از حافظه را نشان می‌دهد در حالی که بخش شاخص اختلاف از آن بخش را نشان می‌دهد.

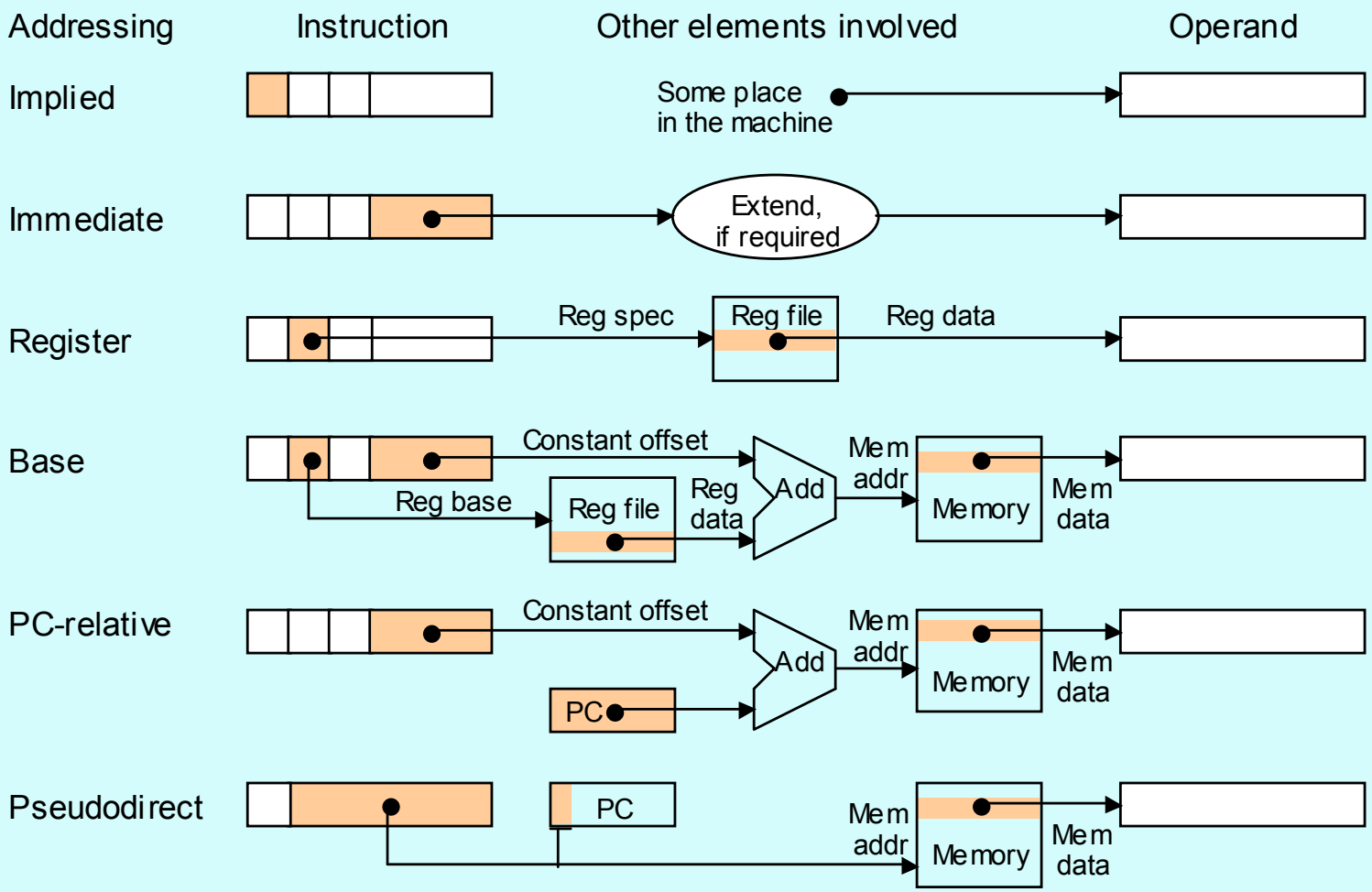


انواع نشانی‌دهی (ادامه...)

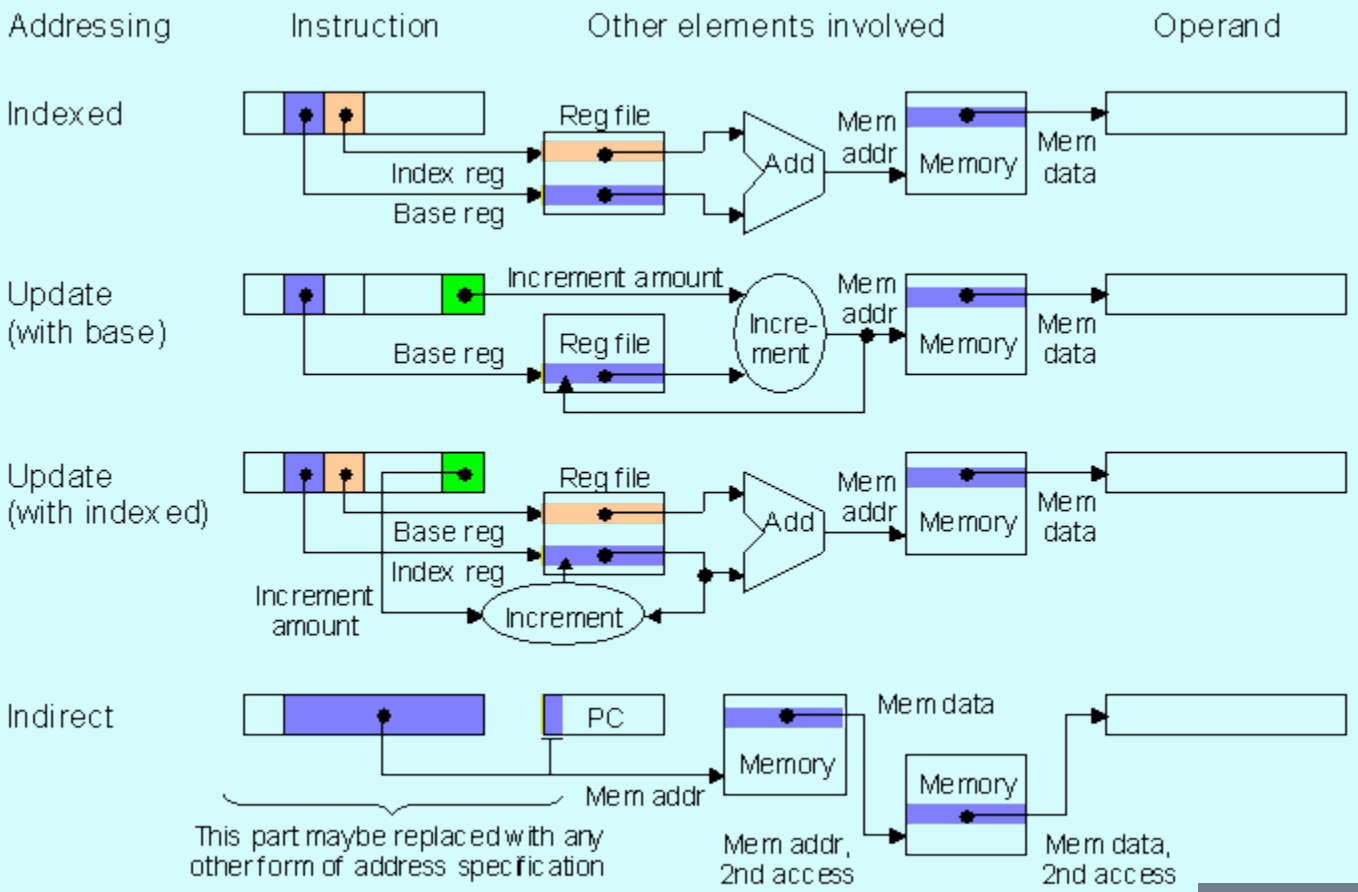
- در این شیوه‌ی محتوای ثبات هر بار یکی افزوده می‌شود



آدرس‌دهی در MIPS



سایر انواع آدرس‌دهی




```
x := B[i]
```

```
x := Mem[p]
p := p + 1
```

```
x := B[i]
i := i + 1
```

```
t := Mem[p]
x := Mem[t]
```

```
x := Mem[Mem[p]]
```