معماری مجموعه دستورالعمل ۳

معماری کامپیوتر

۱۳۹۵-۱۱-۱۳

جلسه‌ی ششم

دانشگاه شهید بهشتی

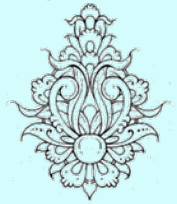
دانشکده‌ی مهندسی برق و کامپیوتر

زمستان ۱۳۹۵

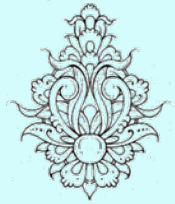
احمد محمودی ازناوه

فهرست مطالب

- مروری برجلسه‌ی پیش
- فراخوانی تابع
- سایر شیوه‌های آدرس‌دهی



- برای فراخوانی یک روال، مراحل زیر انجام می‌شود:
  - ارسال پارامترها به روال
  - انتقال کنترل به روال
  - تخصیص حافظه‌ی مورد نیاز
  - اجرای روال
  - انتقال نتیجه‌ی به دست آمده به برنامه‌ی اصلی
  - بازگرداندن کنترل به برنامه‌ی اصلی



ارسال پارامترها

- در MIPS، برای انتقال پارامترها از ثبات‌ها استفاده می‌شود.

`arguments`

  - $a0 – $a3:
    - برای پارامترهای ارسالی (ثبات شماره ۴ تا ۷)

`result values`

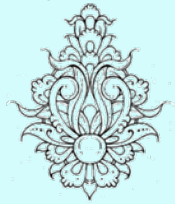
  - $v0, $v1:
    - برای مقادیری فرستاده شده(ثبات شماره ۲ تا ۳)

`return address`

  - $ra:
    - آدرس بازگشت در این ثبات ذخیره می‌شود. (ثبات شماره ۳۱)

سایر ثبات‌ها

- $t0 – $t9: temporaries
  - ثبات‌های موقت، رویه می‌تواند از این ثبات‌ها استفاده کند.
- $s0 – $s7: saved
  - رویه‌ی فراخوانی شده، نباید مقادیر این ثبات‌ها را تغییر دهد.
- $gp: global pointer
  - اشاره‌گر عمومی برای داده‌های ایستا (شماره ۲۸)
- $sp: stack pointer
  - اشاره‌گر به پشته (شماره ۲۹)
- $fp frame pointer
  - اشاره‌گر قاب (شماره ۳۰)



دستور فراخوانی تابع

```
jal ProcedureLabel
```

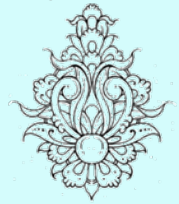
jump-and-link instruction

- با اجرای این دستور، افزون بر پرش به آدرس شروع رویه، آدرس بازگشت در $ra قرار می‌گیرد.
- برای بازگشت به برنامه کافیست از دستور پرشی که پیش از این با آن آشنا شدیم، استفاده کنیم.

```
jr $ra
```

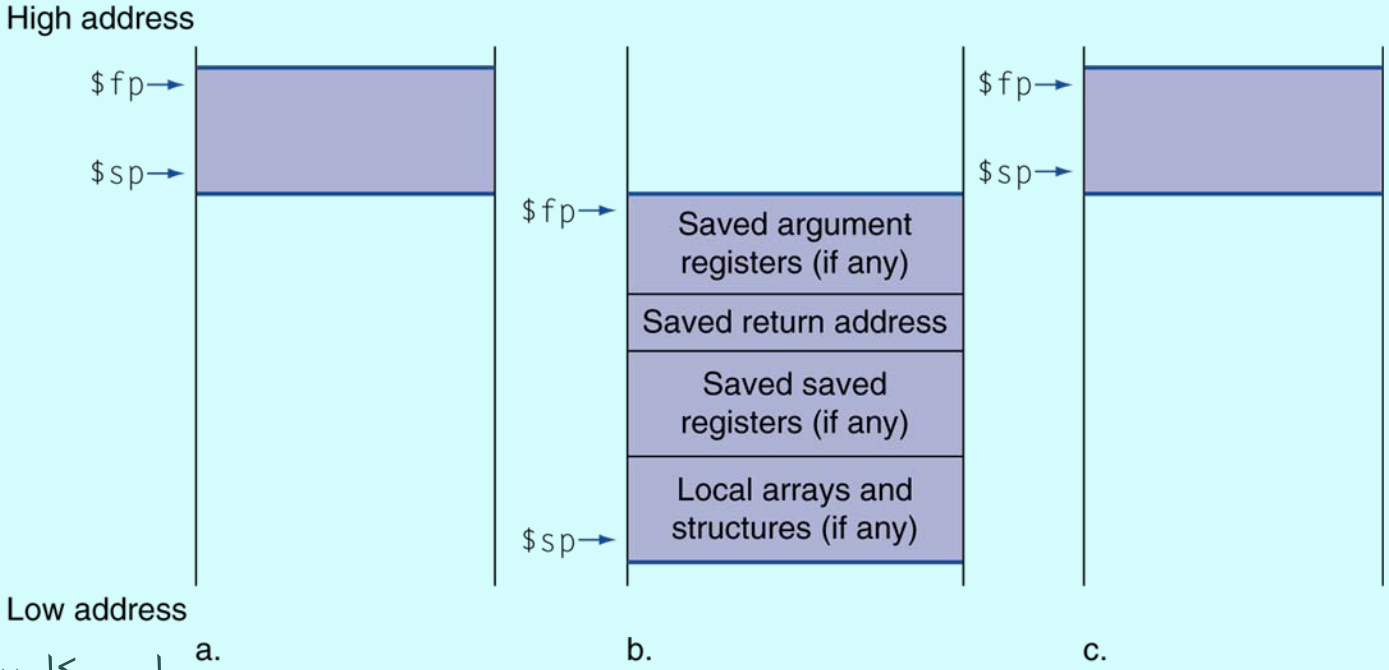
program counter (PC) or instruction address

نباتی که آدرس بخشی از برنامه را که بنا ست اجرا شود، نگه می‌دارد

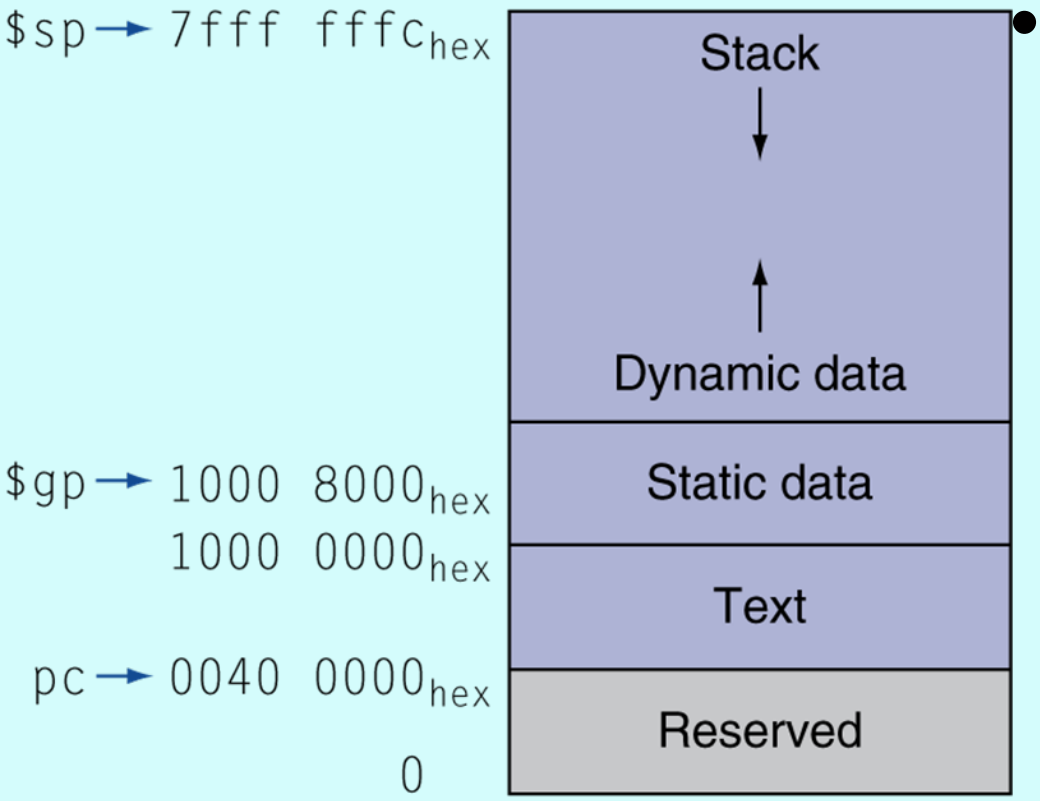


سایر ثبات‌ها (ادامه...)

- متغیرهای محلی یک تابع بر روی پشته ذخیره می‌شوند. به بخشی از پشته که حاوی اطلاعات روال مربوط است، procedure frame می‌گویند.
- با توجه به تغییر آدرس پشته، برای دسترسی به محتویات فریم از ثبات اشاره‌گر فریم $fp استفاده می‌شود.



شیوه‌ی تخصیص حافظه



- text segment:
  - حاوی برنامه
- بخش داده‌ی ایستا
  - دسترسی با استفاده از $gp
- بخش داده‌ی پویا
- پشته

ثبات‌ها در یک نگاه

Name	Register Number	Usage	Preserve on call?
$zero	0	constant 0 (hardware)	n.a.
$at	1	reserved for assembler	n.a.
$v0 - $v1	2-3	returned values	no
$a0 - $a3	4-7	arguments	yes
$t0 - $t7	8-15	temporaries	no
$s0 - $s7	16-23	saved values	yes
$t8 - $t9	24-25	temporaries	no
$gp	28	global pointer	yes
$sp	29	stack pointer	yes
$fp	30	frame pointer	yes
$ra	31	return addr (hardware)	yes

سبک‌های 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.

پیاده‌سازی پرش شرطی

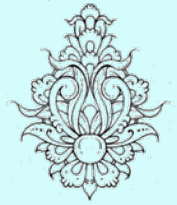
- در صورتی که نیاز به پرش شرطی داشتیم که فاصله‌ی نسبی آن با آدرس فعلی به بیش از شانزده بیت نیاز داشت،چه باید کرد؟

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

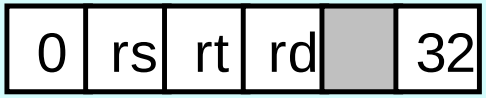
مثلاً آدرس L1 خیلی دور است!

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

البته زحمت انجام این کار به عهده‌ی اسمبلر است!

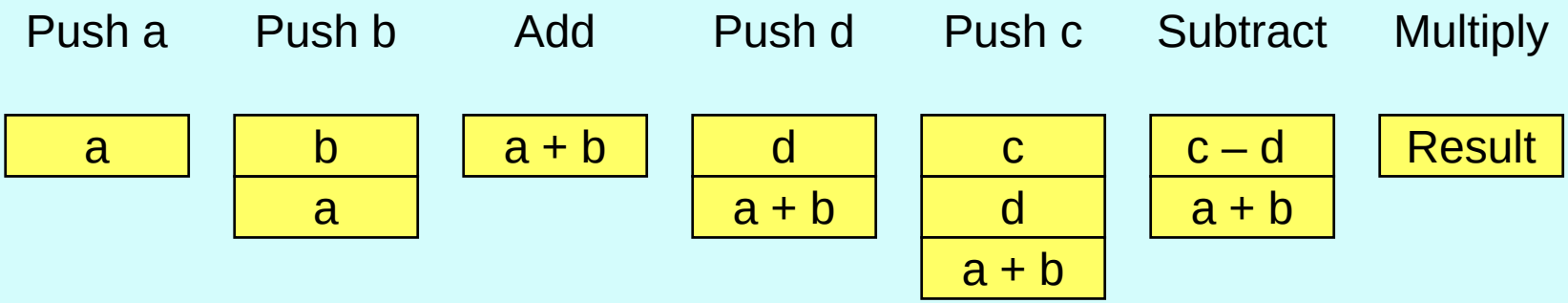


شکل‌های مختلف دستور

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

مثالی دستورات بدون عملوند

Example: Evaluating the expression (a + b) × (c – d)

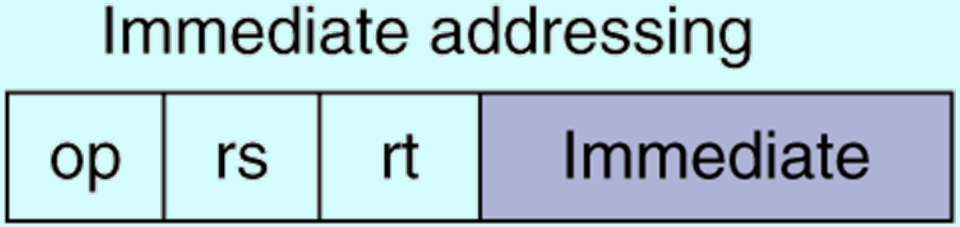


Reverse Polish string: a b + d c – ×

prefix notation

سبک‌های آدرس‌دهی در MIPS

- **آدرس‌دهی بی‌واسطه(برای ثابت‌ها):**
  - عملوند عدد ثابتی است که در دستور ذکر می‌گردد.



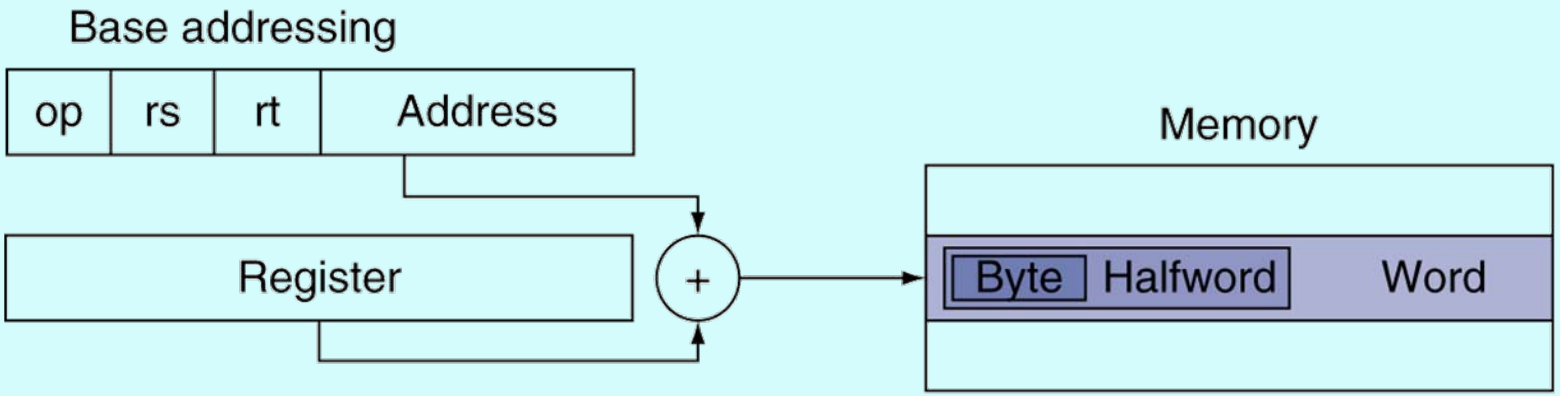
Immediate addressing

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

Register addressing

سبک‌های آدرس‌دهی در MIPS (ادامه...)

- **آدرس‌دهی بر اساس آدرس پایه:**
  - عملوند خانه‌ای از حافظه است که آدرس آن حاصل جمع یک آدرس پایه (در یک ثبات) و یک آدرس ثابت باشد.

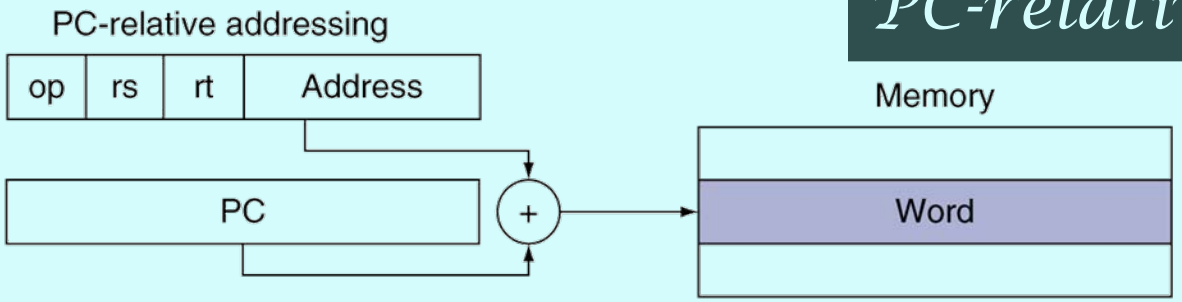


Base or displacement addressing

سبک‌های آدرس‌دهی در MIPS (ادامه...)

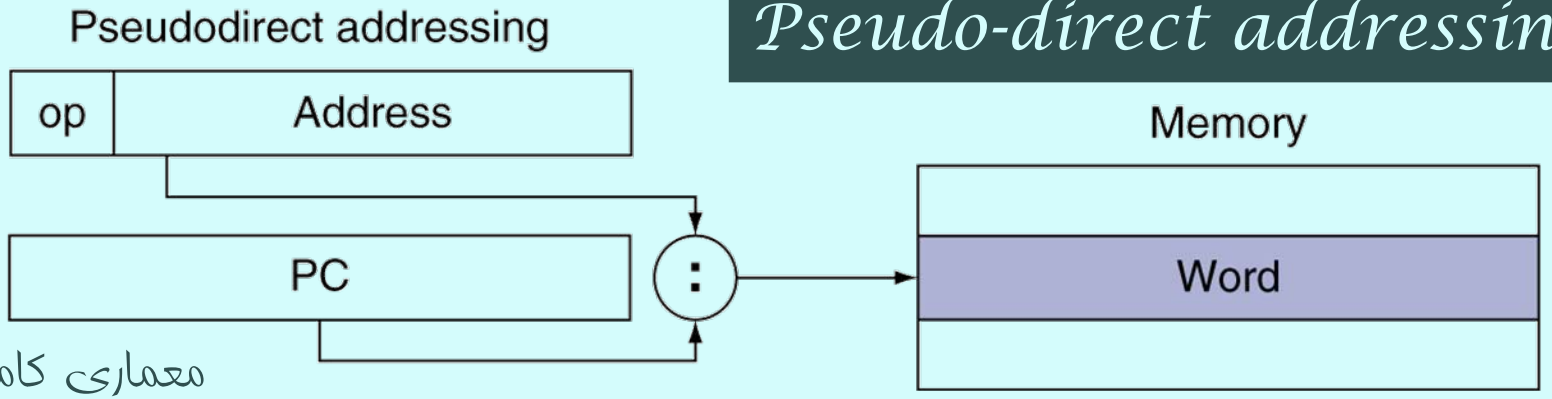
- آدرس دهی نسبی (نسبت به PC):
  - مانند آدرس‌های پرش شرطی که نسبت به آدرس دستور بعدی سنجیده می‌شوند

PC-relative addressing



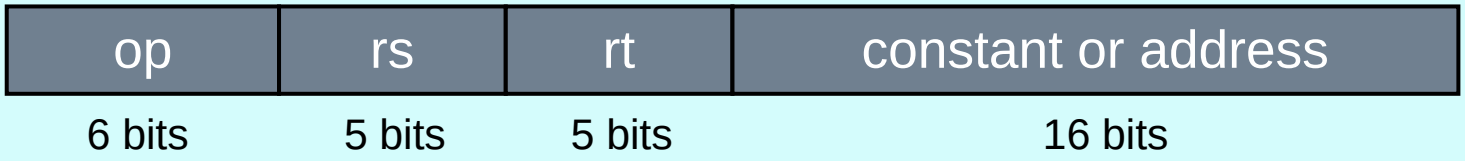
- آدرس‌دهی شبه مستقیم:
  - بخشی از بیت‌های آدرس از PC برداشته می‌شود.

Pseudo-direct addressing



آدرس‌دهی نسبی

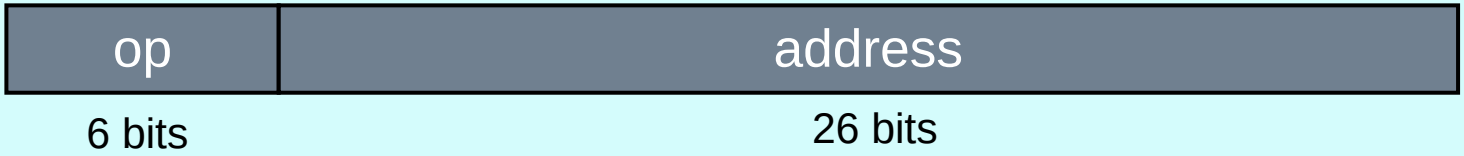
- Target address = PC + offset × 4



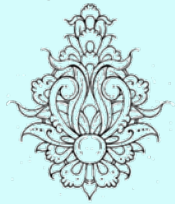
- بدین‌ترتیب، طول برنامه می‌تواند تا چهار گیگابایت افزایش یابد.
- هنگام اجرای هر دستور، PC به آدرس بعدی اشاره می‌کند.
- آدرسی که در دستورات پرش استفاده می‌شود، آدرس کلمه است، نه آدرس بایت

```
j L1
jal ProcedureLabel
```

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



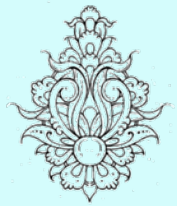
Target address = PC31…28 : (address × 4)



مثال

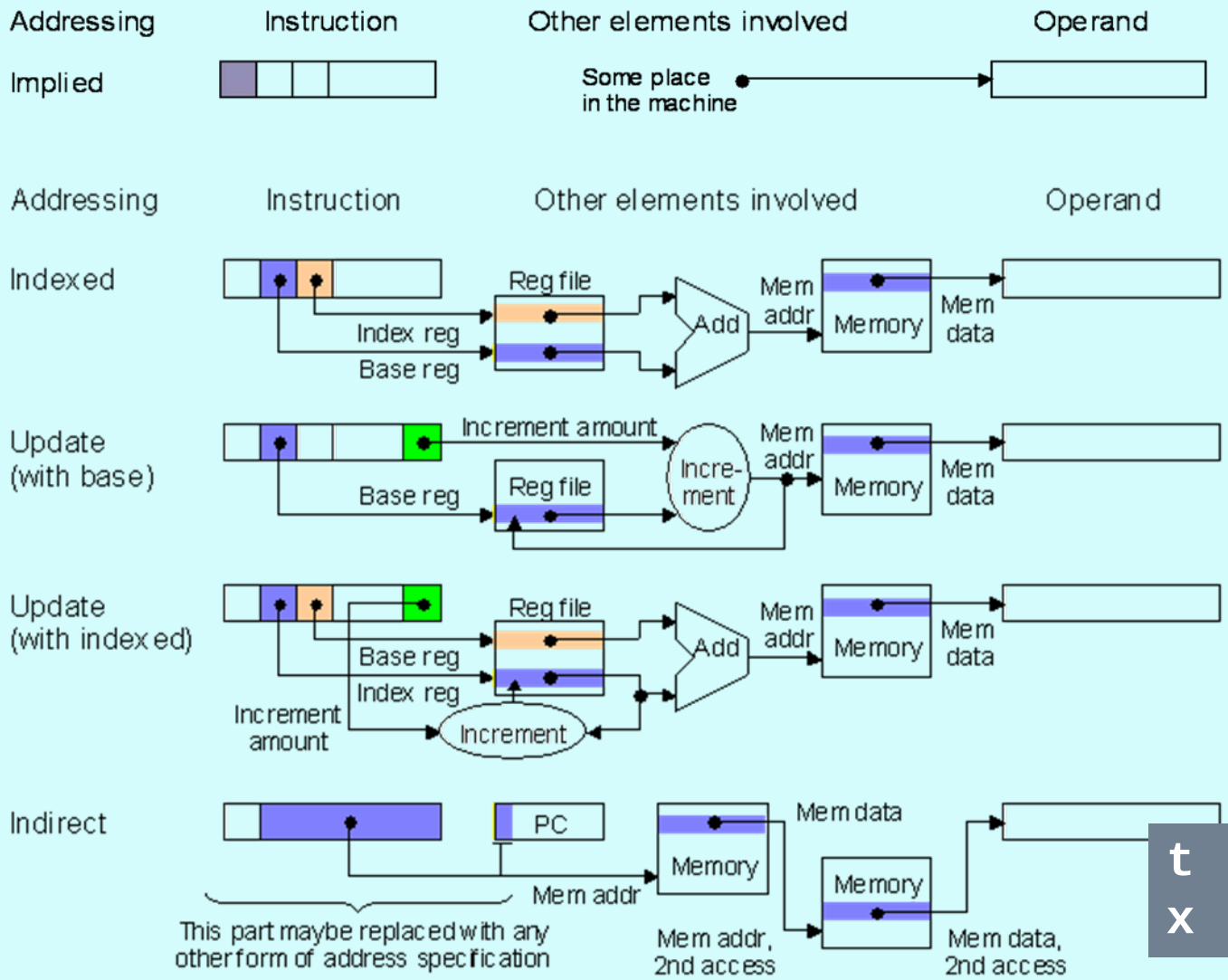
```
while (save[i] == k) i += 1;
```

Loop:	sll $t1, $s3, 2	80000	0	0	19	9	2	0
	add $t1, $t1, $s6	80004	0	9	22	9	0	32
	lw $t0, 0($t1)	80008	35	9	8	0		
	bne $t0, $s5, Exit	80012	5	8	21	2		
	addi $s3, $s3, 1	80016	8	19	19	1		
	j Loop	80020	2	20000				
Exit:	…	80024						



سایر انواع آدرس‌دهی

implicit addressing




```
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]]
```