معماری مجموعه دستورالعمل ۲

... معماری کامپیوتر

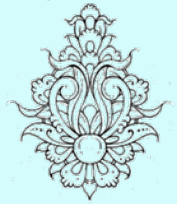
۱۳۹۰-۱۱-۱۳

جلسه‌ی پنجم

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

فهرست مطالب

- مروری برجلسه‌ی پیش
  - مجموعه دستورالعمل
  - معماری MIPS-32
- دستورات شرطی
- فراخوانی تابع



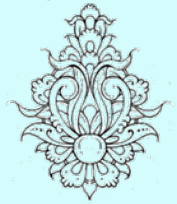
استفاده از اعداد ثابت (ادامه...)

- مثال: برای این که یک واحد از ثبات $s1 کم کنیم، چه پیشنهادی دارید؟

```
addi $s2, $s1, -1
```

- بدین ترتیب با استفاده از این دستورات، تعداد دستورات کاهش می‌یابد.
- یکی از ثابت‌های پراستفاده، صفر است.
- رجیستر $zero، حاوی ثابت 0 است.
  - این ثبات قابل تغییر نیست! برای خیلی کاربردها مفید است، به عنوان مثال برای جابه‌جایی بین رجیسترها

```
add $t2, $s1, $zero
```

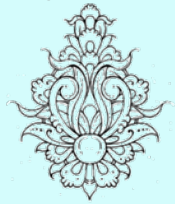


مثال:

A[300]=h+A[300]

```
lw $t0, 1200($t1)
add $t0, $s2, $t0
sw $t0, 1200($t1)
```

op	rs	rt	rd	address /shamt	funct
35	9	8	1200		
0	18	8	8	0	32
43	9	8	1200		



یک کامپیوتر چه تفاوتی با ماشین حساب دارد؟!

- پرش به آدرس دستور دیگر اگر شرطی برقرار باشد، وگرنه روال عادی ادامه پیدا کند.

```
beq register1, register2, L1
```

Branch if equal

```
bnq register1, register2, L1
```

Branch if not equal

```
j L1
```

unconditional jump

دیگر دستورات شرطی

```
slt rd, rs, rt
```

set on less than

```
if (rs < rt) rd = 1; else rd = 0;
```

```
slti rd, rs, constant
```

set on less than

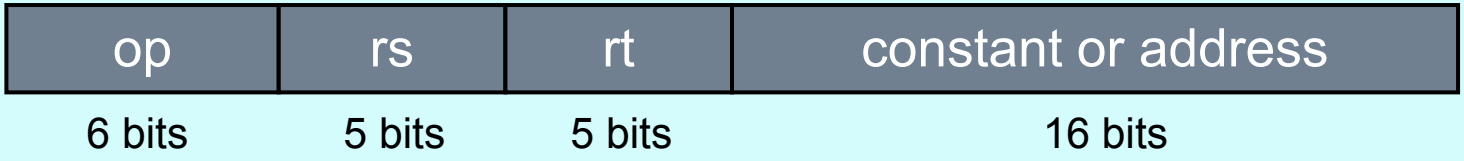
```
if (rs < constant) rd = 1; else rd = 0;
```

- به صورت ترکیبی با سایر دستورات شرطی مورد استفاده قرار می‌گیرد.

```
slt $t0, $s1, $s2  # if ($s1 < $s2)
bne $t0, $zero, L  #   branch to L
```

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.

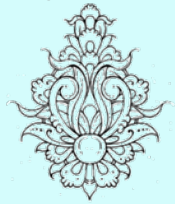


PC = PC + branch address

case/switch

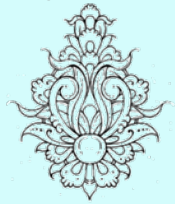
- می‌توان با یک سری دستورات شرطی، چنین ساختاری را به زبان اسمبلی/ماشین تبدیل کرد.
- یک راه مناسب‌تر استفاده از جدول آدرس‌های پرش است.
- جدول مزبور آرایه‌ای از آدرس‌هاست.
- براساس نتایج به دست آمده، یک اندیس از جدول انتخاب شده و محتوای آن در یک ثبات بارگذاری می‌شود.
- سپس از jr برای اجرای کد مورد نظر استفاده می‌شود.

```
jr register
```



طراحی دستورات شرطی

- دستورهای blt و bge
  - سخت‌افزار مدارهای > و یا ≤ نسبت به = یا ≠ کندتر هستند.
  - هنگامی که با پرش همراه شوند، به زمان بیشتری نیاز دارند و در نتیجه پالس‌ساعت کندتر خواهد شد.
  - بدین ترتیب تمام دستورها کند می‌شوند.
- در MIPS دستور پرش در حالتی رجیستری از دیگری کوچک‌تر باشد، تعبیه نشده است. چنین دستوری پیچیده است. استفاده از دو دستور ساده ترجیح داده شده است.



مقایسه و علامت

- دستورات slt و slti در مقایسه، اعداد را به صورت مکمل ۲ در نظر می‌گیرند، برای مقایسه‌ی بدون علامت از دستورهای sltu و sltui استفاده می‌شود.

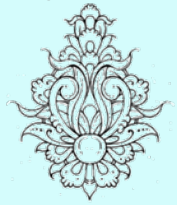
```
$s0 = 1111 1111 1111 1111 1111 1111 1111 1111
$s1 = 0000 0000 0000 0000 0000 0000 0000 0001
```

```
slt  $t0, $s0, $s1  # signed
```

–1 < +1 ⇒ $t0 = 1

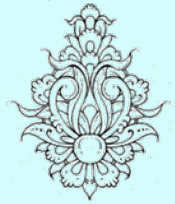
```
sltu $t0, $s0, $s1  # unsigned
```

+4,294,967,295 > +1 ⇒ $t0 = 0



فراخوانی روال

- 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`

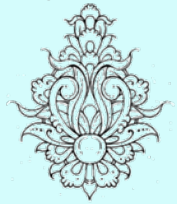
  - $a0 – $a3:
    - برای پارامترهای ارسالی (ثبات شماره ۴ تا ۷)

`result values`

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

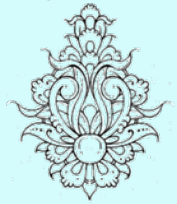
`return address`

  - $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 اشنا 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