معماری مجموعه دستورالعمل

معماری کامپیوتر

۱۳۹۱-۱۱-۱۳

جلسه‌ی چهارم

دانشگاه شهید بهشتی

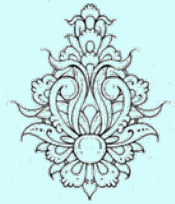
دانشکده‌ی مهندسی برق و کامپیوتر

زمستان ۱۳۹۱

احمد محمودی ازناوه

فهرست مطالب

- مجموعه دستورالعمل
  - اصول طراحی
- معماری MIPS-32

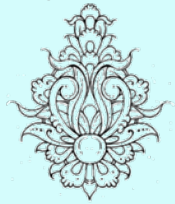


Instruction Set

واژگان مورد استفاده در زبان ماشین

مجموعه‌ی دستورها

- کامپیوترهای مختلف، مجموعه دستورات متفاوتی دارند.
  - با وجود تفاوت، در بسیاری وجوه مشابه هستند.
- نخستین کامپیوترها دارای مجموعه‌ی دستورات ساده‌ای بودند. در برخی از سیستم‌های امروزی نیز حفظ این سادگی ترجیح داده می‌شود.
- در این بخش با مجموعه‌ی دستورات MIPS آشنا خواهیم شد.



Microprocessor without Interlocked Pipeline Stages

مجموعه‌ی دستورها (ادامه...)

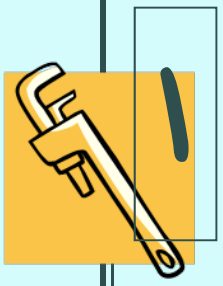
- **در ادامه نیم‌نگاهی خواهیم داشت به مجموعه‌ی دستورالعمل‌های ARM و Intel x86**

در سال ۲۰۰۸ بیش از سه میلیارد پردازنده‌ی **ARM** به فروش رفته و تقریبا در تمامی ۳۳۰ میلیون کامپیوتر شخصی یک پردازنده‌ی **Intel** تعبیه شده است.

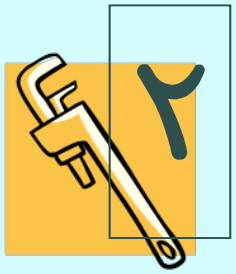
یکی از خصوصیات مهم در مورد مجموعه‌ی دستورات این است که سخت‌افزار مورد نیاز آن ساده باشد.

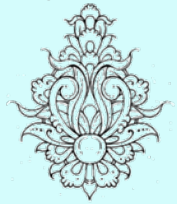
با توجه به آشنایی همه‌ی دوستان با زبان اسمبلی **x86** بیشتر این بخش در کلاس حل تمرین پوشش داده می‌شود

اصول طراحی



- نظم (Regularity) منجر به ارائه‌ی مداری ساده‌تر (Simplicity) می‌شود:
  - طول ثابت دستورات
  - تعداد کم فرمت‌ها
  - محل ثابت opcode
- طرح هرچه کوچک‌تر باشد، سریع‌تر خواهد بود:
  - تعداد کم ثبات‌ها
  - تعداد کم دستورالعمل‌ها
  - مودهای آدرس‌دهی محدود

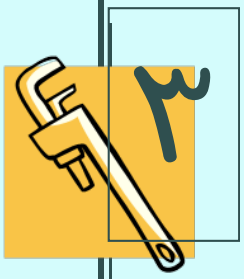




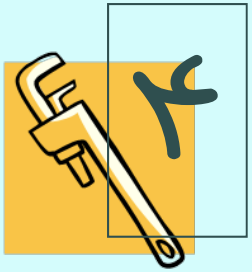
اصول طراحی (ادامه...)

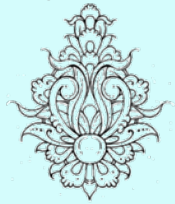
- **موارد پر استفاده، باید سریع‌تر باشند:**
  - دستورهای محاسباتی بر روی ثبات‌ها عمل می‌کنند.
  - دستورهایی که داده‌های بلاواسطه داشته باشند.

ماشین load-stor



- **در یک طراحی خوب، معادل حل یک مسأله‌ی بهینه‌سازی است.**
  - استفاده از قالب‌های متنوع طراحی را پیچیده می‌کند، در عوض طول دستورات ثابت مانده است.
    - با این حال، باید تا جایی که شدنی است، قالب‌ها مشابه باشند.





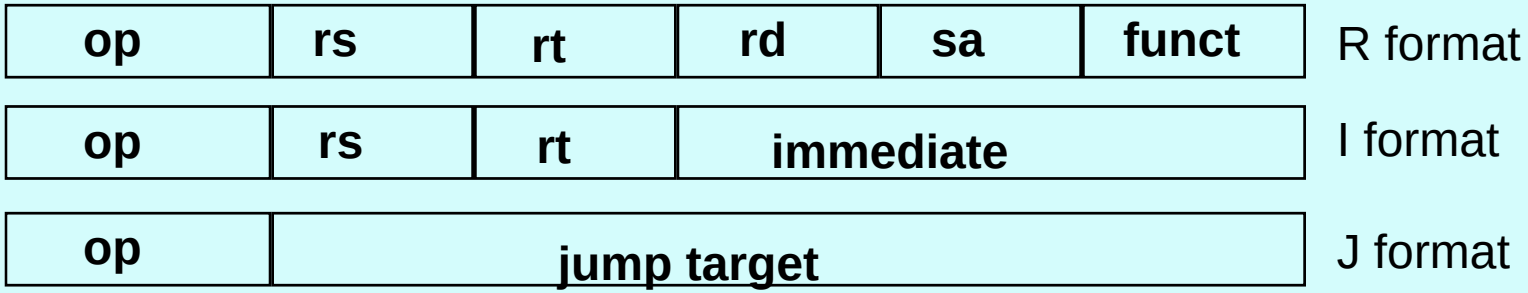
معماری MIPS-32

- انواع دستورالعمل
  - محاسباتی
  - خواندن/نوشتن
  - پرش و انشعاب
  - ممیز شناور
  - و..

ثبات‌ها

R0 - R31
PC
HI
LO

سه فرمت برای دستورالعمل‌ها وجود دارد



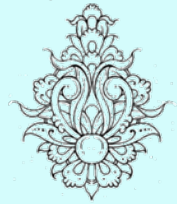
نمایش دستورات

machine code

- همان‌طور که داده‌ها با اعداد دودویی نمایش داده می‌شوند، دستورها هم با اعداد دودویی نشان داده می‌شوند.
- دستورات MIPS:
  - سی و دو بیتی هستند
  - بخشی از این سی و دو بیت به عملیات بستگی دارد.

opcode (operation code)

- به هر ثبات عددی نسبت داده می‌شود.
  - شماره‌ی ۸ تا ۱۵ برای $t0 – $t7
  - شماره‌های ۲۴ و ۲۵ برای $t8 – $t9
  - اعداد ۱۶ تا ۲۳ برای $s0 – $s7



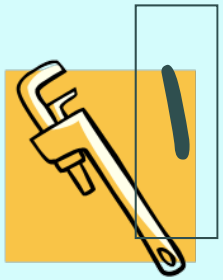
قالب دستورهای MIPS

- **فیلدهای دستور:**
  - op: دستور را مشخص می‌کند.
  - rs: ثبات منبع اول
  - rt: دومین ثبات منبع
  - rd: ثبات مقصد
  - shamt: میزان شیفت
  - funct: نوع خاصی از دستور

دستورهای حساب

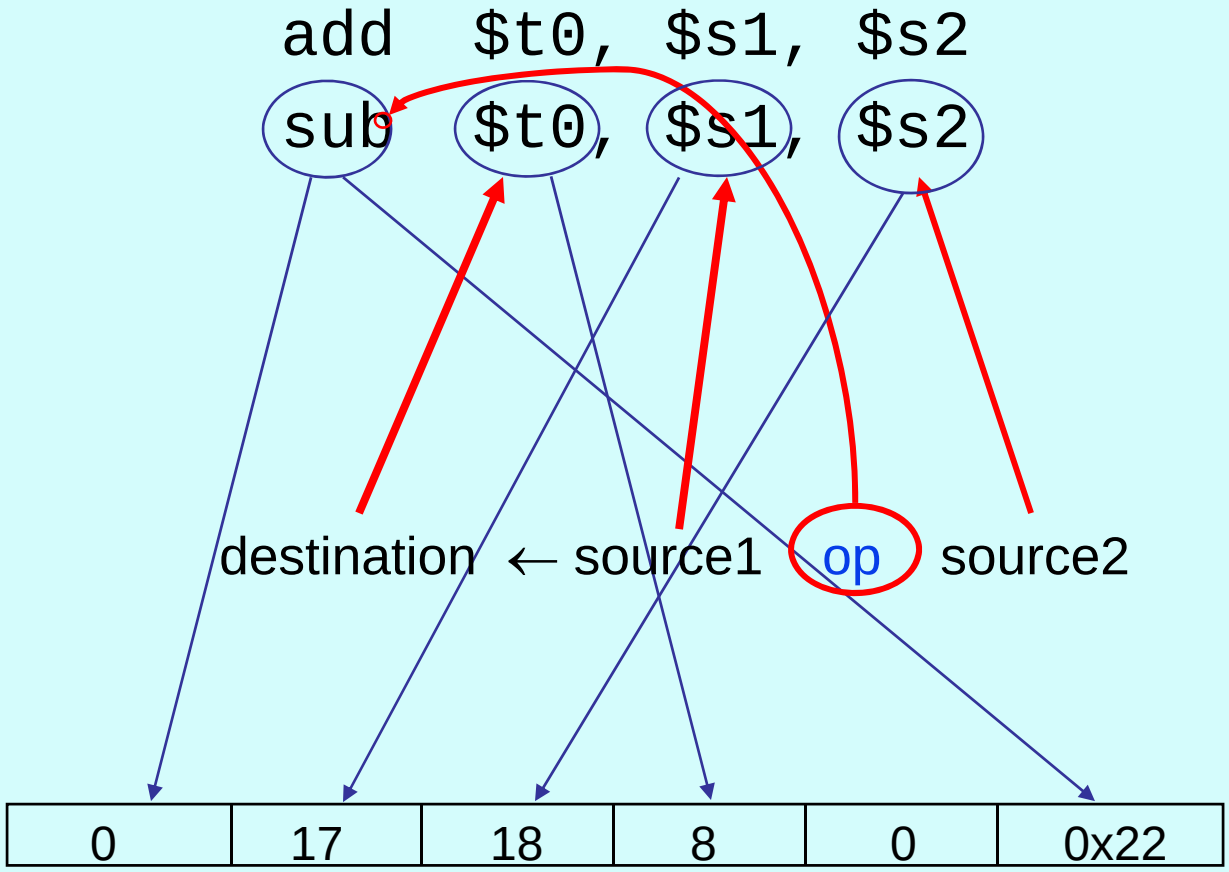
- جمع (دو منبع و یک حاصل)

```
add a, b, c  # a gets b + c
```



- تمام دستورات حسابی دارای سه عملوند هستند.
- نظم (Regularity) منجر به ارائه‌ی مداری ساده‌تر (Simplicity) می‌شود.
- مدار ساده‌تر معادل به دست آوردن مداری با کارایی بالا و قیمتی پایین خواهد شد.
- سخت‌افزار برای تعداد عملوند متغیر پیچیده‌تر است.

دستورهای محاسباتی



Irwin, PSU

مثال

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

add $t0, $s1, $s2

special	$s1	$s2	$t0	0	add
0	17	18	8	0	32
000000	10001	10010	01000	00000	100000

$00000010001100100100000000100000_2 = 02324020_{16}$

حافظه به عنوان عملوند

- ساختارهای پیچیده‌تر در حافظه ذخیره می‌شوند.
  - مانند آرایه‌ها، ساختارها و ...
- برای اعمال دستورهای حسابی
  - داده از حافظه به ثبات منتقل شده و پس از انجام محاسبات، حاصل در حافظه نوشته می‌شود.
- هر خانه‌ی حافظه به یک بایت اشاره می‌کند.
- کلمات در حافظه هم‌تراز شده‌اند، شروع هر کلمه مضربی از چهار است.

alignment restriction

- در MIPS، خانه‌ی حافظه با آدرس کوچک‌تر، حاوی بایت پرارزش‌تر است.

big-endian

بخش پرارزش‌تر در خانه‌ی اول قرار می‌گیرد

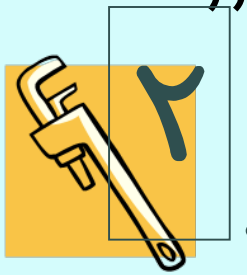
ثبات و حافظه

- دستیابی به محتوای ثبات‌ها بسیار سریع‌تر از محتوای حافظه می‌باشد.
- برای هر بار دستیابی به حافظه، اجرای دستورات lw و sw لازم است. یعنی تعداد دستورات بیشتر است.
- کامپایلر باید تا جایی که ممکن است از رجیسترها به عنوان متغیر استفاده کنند.
  - در صورت در اختیار نداشتن ثبات، از بین متغیرها، آن‌هایی که کم‌تر مورد استفاده قرار می‌گیرند، از ثبات خارج می‌شوند.
  - استفاده بهینه از فضای ثبات‌ها مهم است.

spilling register

ثبات‌ها در نقش عملوند

- در دستورات حسابی، محدود به استفاده از ثبات‌ها هستیم.
- MIPS دارای ۳۲ ثبات ۳۲ بیتی است.
  - که با شماره‌های 0 تا 31 مشخص می‌شوند.
  - داده‌هایی که بناست در پردازشگر مورد استفاده قرار گیرند، به این ثبات‌ها منتقل می‌شوند.
  - هر ۳۲ بیت را یک کلمه (Word) می‌نامند.
- در زبان اسمبلی ثبات‌ها به صورت زیر نام‌گذاری می‌شوند.
  - $s0, $s1, …, $s7
  - $t0, $t1, …, $t9 ثبات‌های موقت که در کامپایل مورد استفاده قرار می‌گیرند.
- طرح هرچه کوچک‌تر باشد، سریع‌تر خواهد بود.

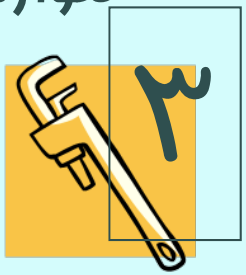


استفاده از اعداد ثابت

- در بسیاری موارد لازم است، از اعداد ثابت در برنامه‌ها استفاده کرد. چه راهی پیشنهاد می‌دهید؟
  - به عنوان مثال در صورتی که بخواهیم به $s3 چهار واحد اضافه کنیم؟

```
lw  $t0,AddrConstant4($s1) # $t0= constant 4
add $s3,$s3,$t0
```

  - با توجه به استفاده مکرر از چنین دستوراتی (بر اساس آزمون SPEC2006 نیمی از دستورات MIPS دارای عملوند ثابت هستند) و این اصل که
  - موارد پر استفاده، باید سریع‌تر باشند.




```
addi $s3, $s3, 4
```

دستورهای با عملوند ثابت

- برای دستورات lw، st و دستوراتی که نیاز به استفاده از ثابت‌ها دارند، قالب دیگری مطرح می‌شود.
  - rt: ثبات منبع یا مقصد
  - بدین ترتیب می‌توان ثابتی از -2^{15} تا $+2^{15}-1$ را در این گونه دستورها به کار برد.
  - همچنین، برای دستوراتی که با آدرس حافظه کار می‌کنند، بخش آخر دربردارنده‌ی آدرس می‌باشد.

Constant

- در یک طراحی خوب، معادل حل یک مسأله‌ی بهینه‌سازی است.
  - استفاده از قالب‌های متنوع طراحی را پیچیده می‌کند، در عوض طول دستورات ثابت مانده است.
  - با این حال، باید تا جایی که شدنی است، قالب‌ها مشابه باشند.

دستور شیفت

Shift amount

- shamt: میزان شیفت
- هنگام شیفت به چپ(راست)، پر(کم)ارزش‌ترین بیت با ’0‘ پر می‌شود.
- شیفت به چپ، معادل ضرب در دو است.
- شیفت به راست برای اعداد بدون‌علامت معادل تقسیم بر دوست.