پردازنده

معماری کامپیوتر

۱۳۹۵-۱۱-۱۳

جلسه‌ی دوازدهم

دانشگاه شهید بهشتی

دانشکده‌ی مهندسی برق و کامپیوتر

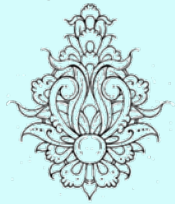
بهار ۱۳۹۵

احمد محمودی ازناوه

http://faculties.sbu.ac.ir/~a_mahmoudi/

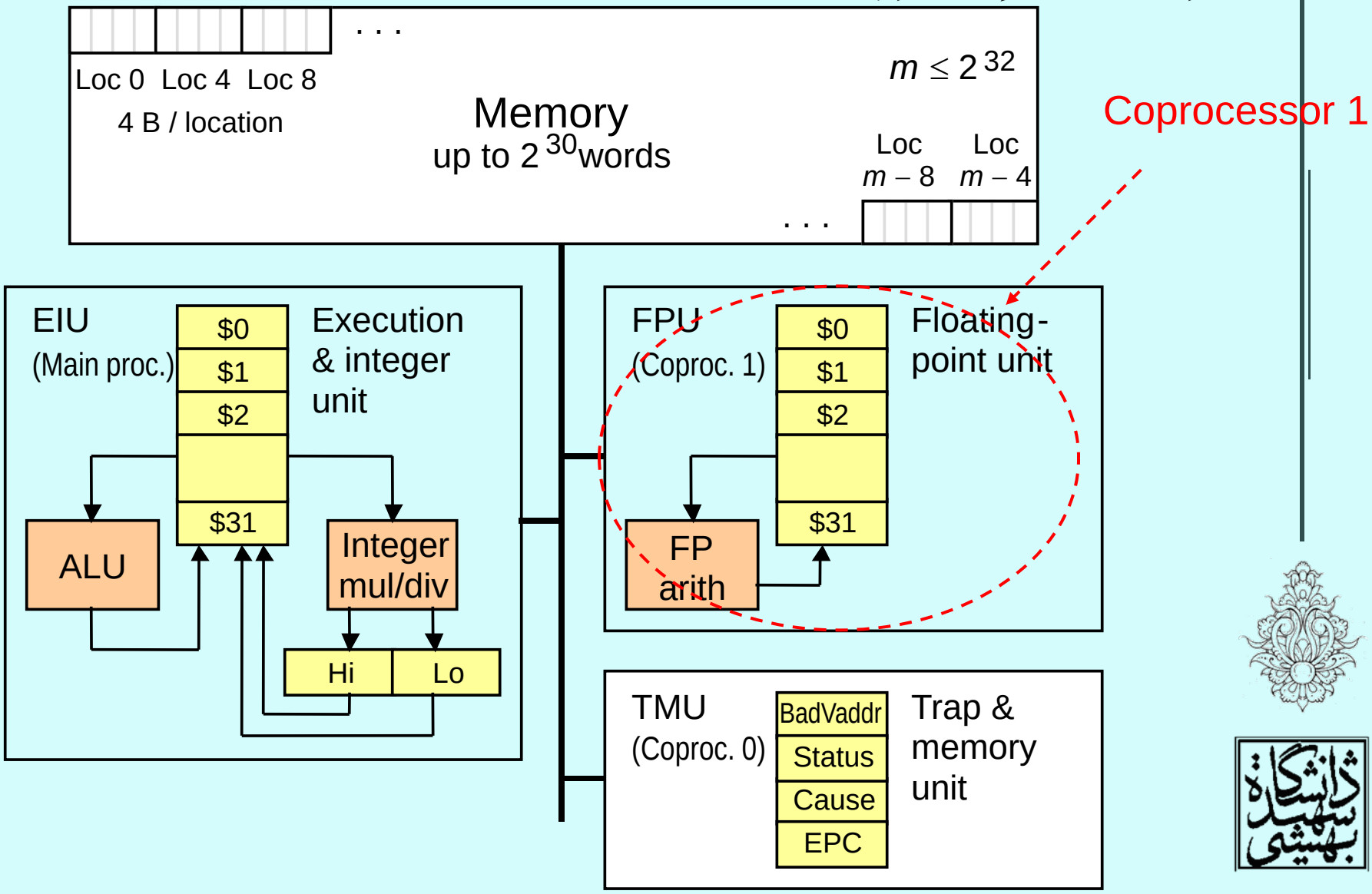
فهرست مطالب

- واحد ممیز شناور
- پردازنده
  - مسیر گذار داده



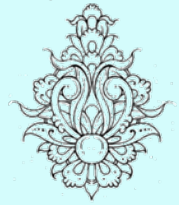
واحد ممیز شناور

- واحد محاسبات ممیز شناور معمولا اعمال جمع، تفریق، ضرب، تقسیم، معکوس‌سازی و تبدیل به صحیح را انجام می‌دهد.
- عملیات ممیز شناور به چند سیکل برای اجرا نیاز دارد.
- به صورت خط لوله نیز قابل استفاده می‌باشد.


Loc 0 Loc 4 Loc 8
4 B / location
. . .
Memory
up to 2 30 words
m ≤ 2 32
Loc m – 8
Loc m – 4
. . .
Coprocessor 1
EIU
(Main proc.)
$0
$1
$2
$31
Execution & integer unit
ALU
Integer mul/div
Hi
Lo
FPU
(Coproc. 1)
$0
$1
$2
$31
Floating-point unit
FP arith
TMU
(Coproc. 0)
BadVaddr
Status
Cause
EPC
Trap & memory unit

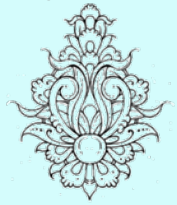
دستورالعمل‌های ممیز شناور در MIPS

- سی و دو ثبات جداگانه برای عملیات ممیز شناور وجود دارد:
  - $f0, $f1, … $f31
  - در صورت استفاده از دقت مضاعف این ثبات‌ها به صورت دو تایی مورد استفاده قرار می‌گیرند:
    - $f0/$f1, $f2/$f3
  - نسخه‌ی MIPS ۲، سی‌ودو ثبات شصت و چهار بیتی دارد.
- دستورات ممیز شناور تنها بر روی ثبات‌های ممیز شناور عمل می‌کنند.



دستورالعمل‌های ممیز شناور در MIPS (ادامه...)

- دستورات خواندن و نوشتن
  - lwc1, ldc1, swc1, sdc1
    - `ldc1 $f8, 32($sp)`
- محاسبات با دقت معمولی
- add.s, sub.s, mul.s, div.s
  - `add.s $f0, $f1, $f6`
- محاسبات با دقت مضاعف
- add.d, sub.d, mul.d, div.d
  - e.g., `mul.d $f4, $f4, $f6`



دستورالعمل‌های ممیز شناور در MIPS (ادامه...)

- دستورات مقایسه

– `c.`*xx*`.s`, `c.`*xx*`.d` (*xx* is `eq`, `lt`, `le`, ...)

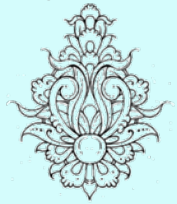
– Sets or clears FP condition-code bit

  - e.g. `c.lt.s $f3, $f4`

- دستورات پرش

– `bc1t`, `bc1f`

  - e.g., `bc1t TargetLabel`



خلاصه‌ای از دستورات ممیز شناور

	Instruction	Usage
Copy	Move s/d registers	`mov.* fd,fs`
	Move fm coprocessor 1	`mfc1 rt,rd`
	Move to coprocessor 1	`mtc1 rd,rt`
Arithmetic	Add single/double	`add.* fd,fs,ft`
	Subtract single/double	`sub.* fd,fs,ft`
	Multiply single/double	`mul.* fd,fs,ft`
	Divide single/double	`div.* fd,fs,ft`
	Negate single/double	`neg.* fd,fs`
	Compare equal s/d	`c.eq.* fs,ft`
	Compare less s/d	`c.lt.* fs,ft`
	Compare less or eq s/d	`c.le.* fs,ft`
Conversions	Convert integer to single	`cvt.s.w fd,fs`
	Convert integer to double	`cvt.d.w fd,fs`
	Convert single to double	`cvt.d.s fd,fs`
	Convert double to single	`cvt.s.d fd,fs`
	Convert single to integer	`cvt.w.s fd,fs`
	Convert double to integer	`cvt.w.d fd,fs`
Memory access	Load word coprocessor 1	`lwc1 ft,imm(rs)`
	Store word coprocessor 1	`swc1 ft,imm(rs)`
Control transfer	Branch coproc 1 true	`bc1t L`
	Branch coproc 1 false	`bc1f L`

* s/d for single/double

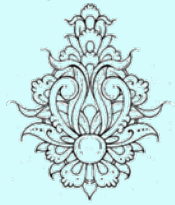
مثال تبدیل فارنهایت به سلسیوس

- کد به زبان C

```
float f2c (float fahr) {
  return ((5.0/9.0)*(fahr - 32.0));
}
```

- کد C کامپایل شده

```
f2c: lwc1  $f16, const5($gp)
     lwc2  $f18, const9($gp)
     div.s  $f16, $f16, $f18
     lwc1  $f18, const32($gp)
     sub.s $f18, $f12, $f18
     mul.s $f0,  $f16, $f18
     jr    $ra
```



پردازنده ...

Performance = 1 / Execution time simplified to 1 / CPU execution time

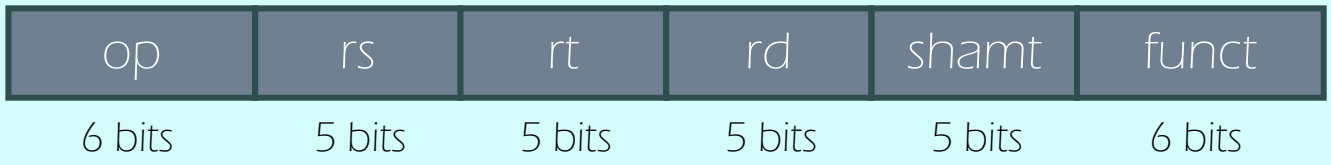
CPU execution time = Instructions × CPI / (Clock rate)

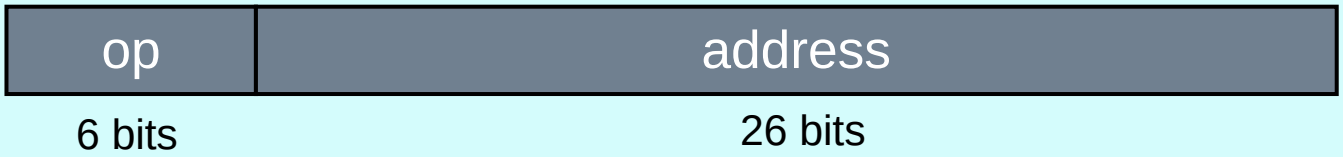
Performance = Clock rate / (Instructions × CPI)

- کارایی یک برنامه توسط موارد زیر تعیین می‌شوند:
  - Instruction count
  - CPI and Cycle time
  - کامپایلر و ISA موارد تأثیرگذار بر روی مورد نخست بودند که پیش‌ار این مورد بررسی قرار گرفتند.
  - سخت‌افزار طراحی شده برای CPU تعداد سیکل به ازای هر دستور و طول سیکل را مشخص می‌کند.

پیش‌گفتار (ادامه...)

- در این بخش یک پیاده‌سازی ساده‌سازی شده از پردازنده‌های MIPS ارائه خواهد شد. که شامل دستورات زیر می‌باشد:
  - Memory reference: `lw`, `sw`
  - Arithmetic/logical: `add`, `sub`, `and`, `or`, `slt`
  - Control transfer: `beq`, `j`

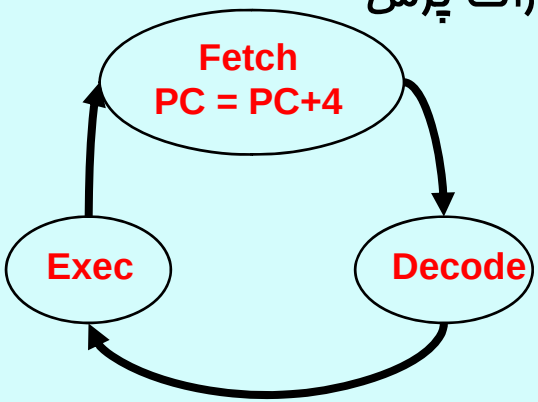




نحوه‌ی اجرای یک دستورالعمل

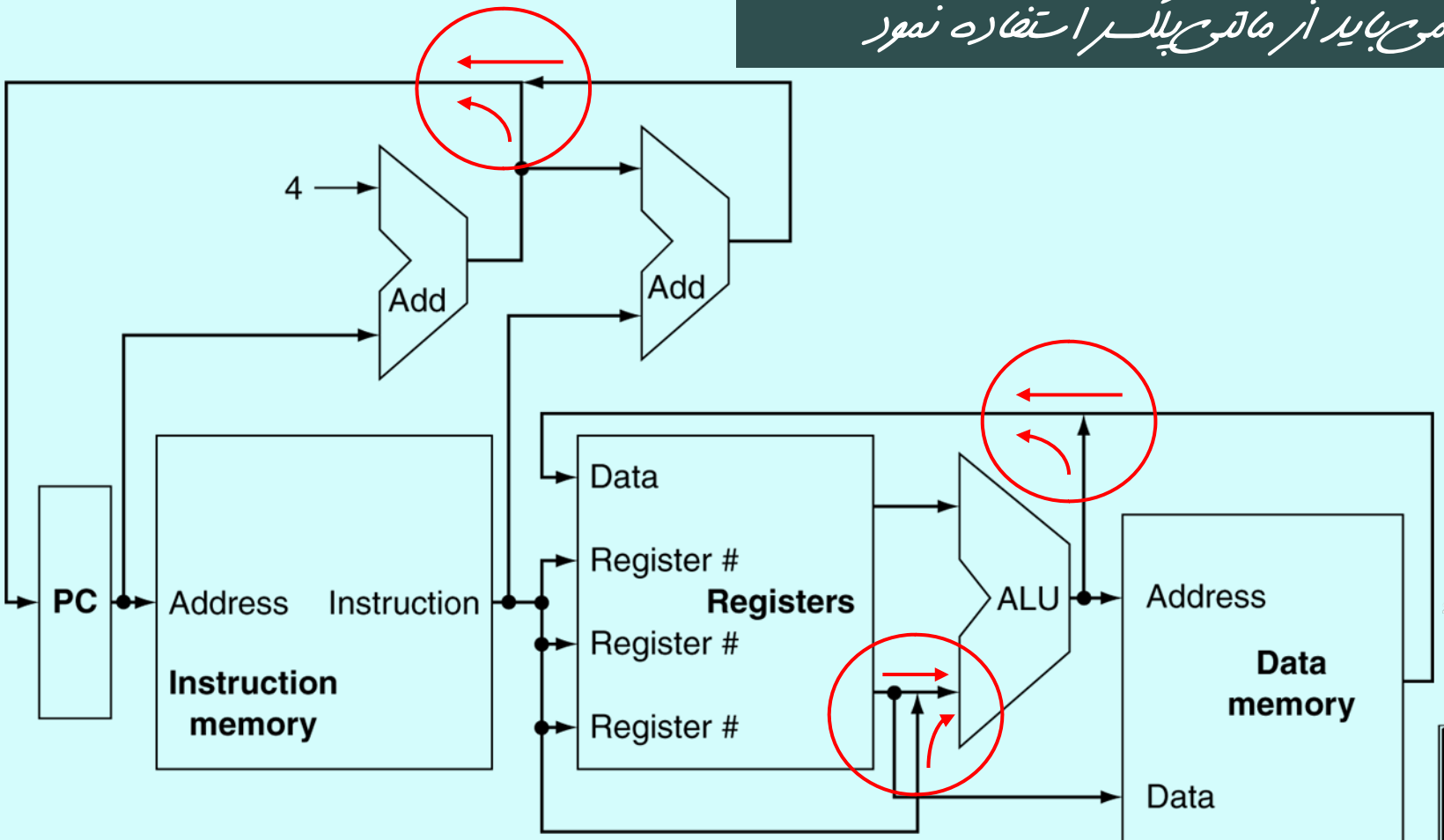
Fetch

- PC به آدرس خانه‌ای از حافظه اشاره می‌کند که می‌باید اجرا شود. دستور مزبور «واکشی» می‌شود.
- بسته به نوع دستورالعمل، عملوندها آماده می‌شوند، به عنوان مثال محتوای ثبات‌های مورد نظر خوانده می‌شود.
- بسته به نوع دستورالعمل
  - ALU برای اهداف زیر مورد استفاده قرار می‌گیرد:
    - به دست آوردن نتیجه‌ی محاسبات
    - محاسبه‌ی آدرس حافظه
    - به دست آوردن آدرس دستور بعدی در دستورات پرش
  - خواندن/نوشتن در حافظه
  - قرار دادن آدرس دستور بعدی در PC



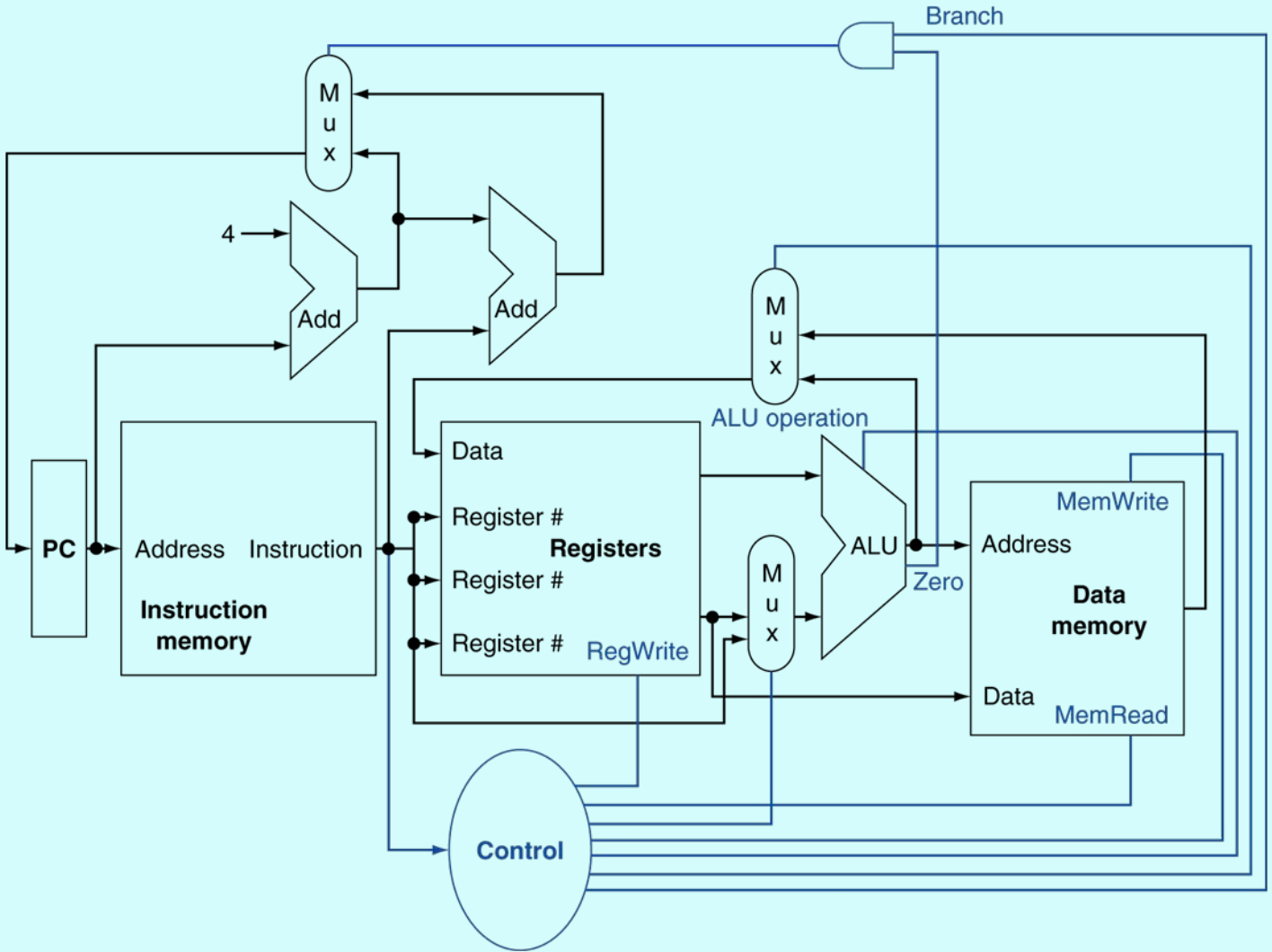
نمایی انتزاعی از CPU

اتصال سیم‌ها به یکدیگر درست نیست
می‌باید از مالتی پلکسر استفاده نمود



واحد کنترل

- سیگنال‌های کنترلی، بر خلاف سیگنال‌های داده که شامل اطلاعاتی هستند که می‌باید مورد پردازش قرار گیرد، برای هدایت سخت‌افزارها مورد استفاده قرار می‌گیرد.

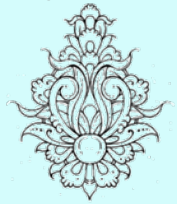


مبانی طراحی دیجیتال

- داده‌های به صورت دودویی کد می‌شوند
- به ازای هر بیت، یک سیم استفاده می‌شود.
- برای داده‌های چند بیتی از یک دسته سیم که گذرگاه نامیده می‌شود، استفاده می‌شود.

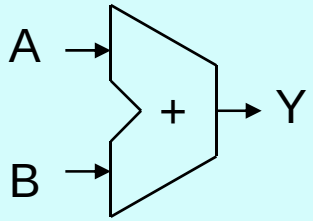
multi-wire buses

- مدارها به دو دسته تقسیم می‌شوند:
  - مدارهای ترکیبی
  - مدارهای ترتیبی

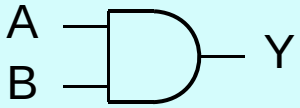


المان‌های ترکیبی

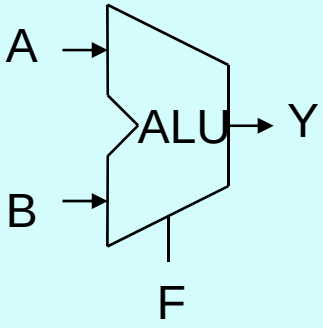
- جمع‌کننده
  - Y = A + B



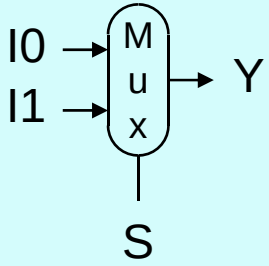
- گیت AND
  - Y = A & B



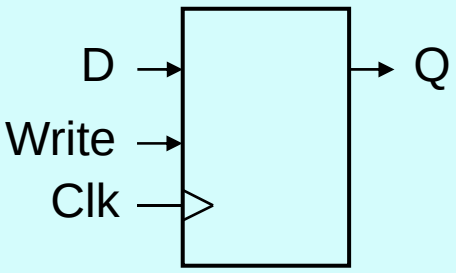
- Arithmetic/Logic Unit
  - Y = F(A, B)

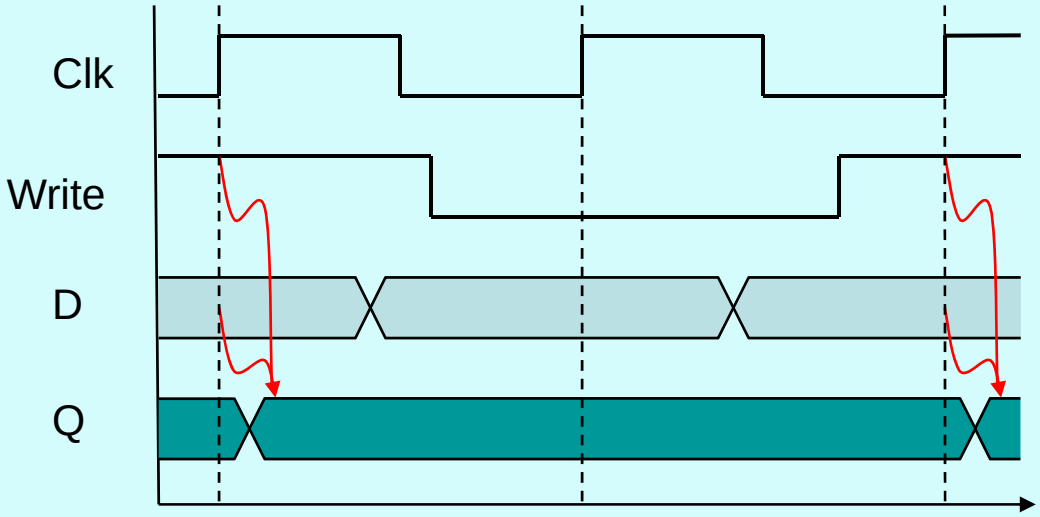


- مالتی‌پلکسر
  - Y = S ? I1 : I0

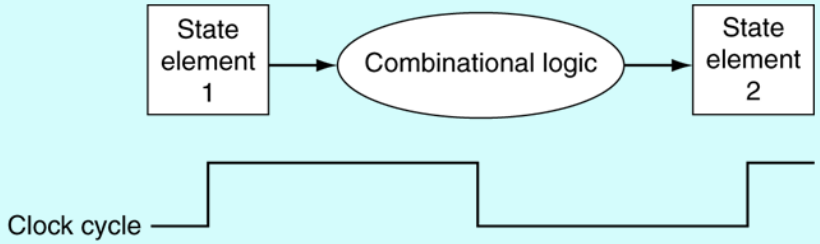


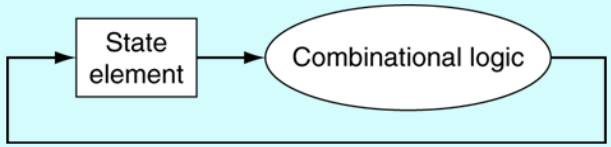
المان‌های ترتیبی





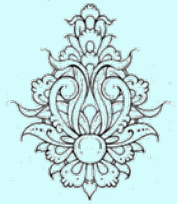
- چرا از پالس ساعت حساس به لبه استفاده می‌شود؟





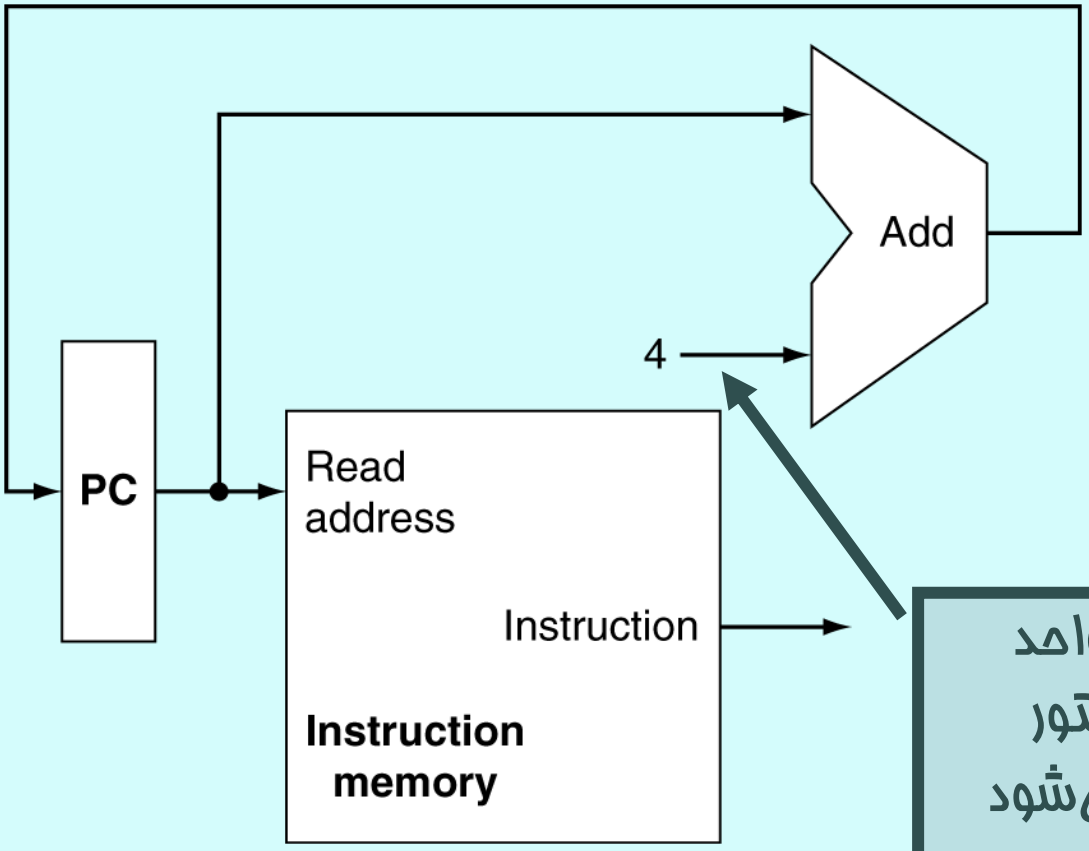
طراحی مسیر گذار داده (محاسبات)

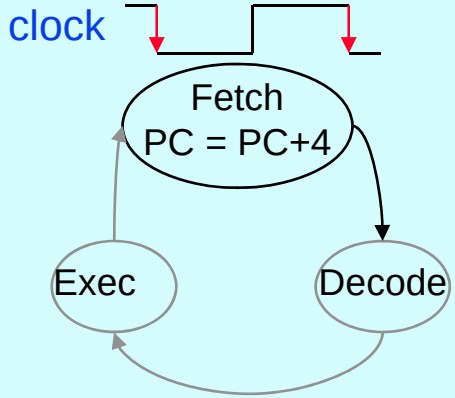
- Datapath در برخی منابع فارسی به «مسیر گذار داده» ترجمه شده است.
- کار این واحد، پردازش، انتقال و ذخیره‌ی داده‌های CPU است و شامل ALU، جمع‌کننده‌ها، مالتی‌پلکسرها، ثبات‌ها و گذرگاه‌داده می‌باشد.
- در ادامه با واحد محاسباتی MIPS به تدریج آشنا خواهیم شد.



Instruction Fetch

واکشی دستورات



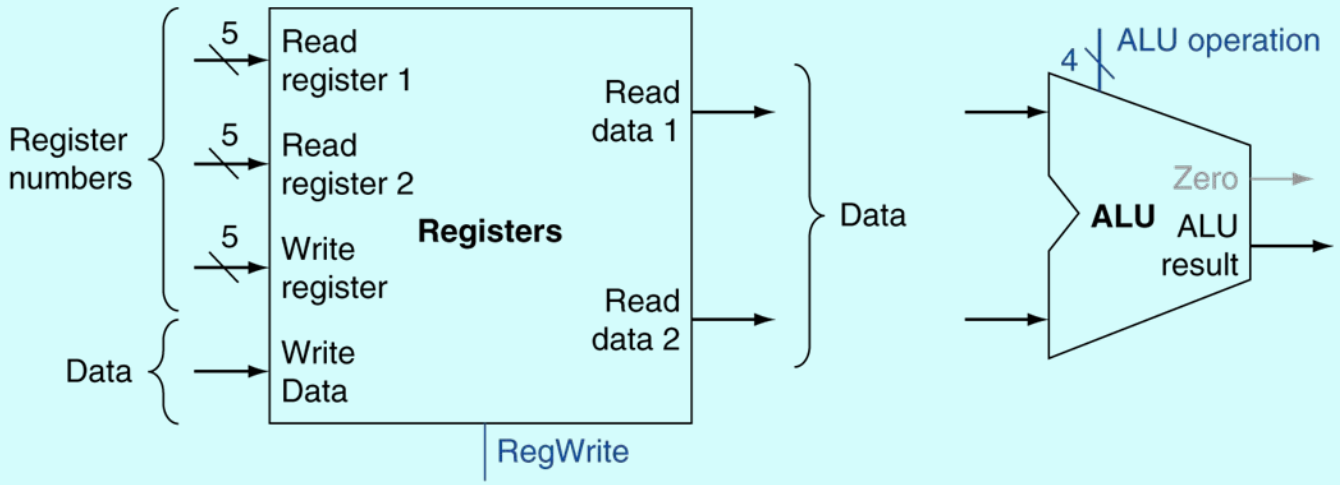


با افزودن چهار واحد
برای واکشی دستور
بعدی آماده می‌شود

Arithmetic-logical instruction

دستورهای نوع R

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



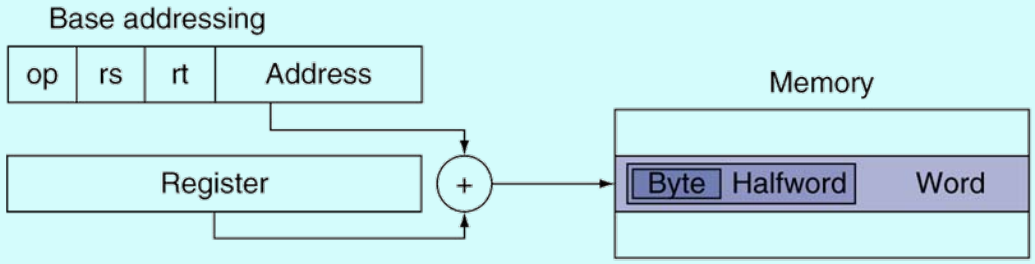
a. Registers

b. ALU

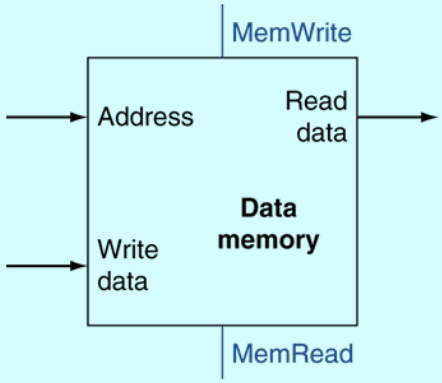
register file

بایگانی ثبات:
واحدی شامل همه‌ی ثبات‌ها که با انتخاب شماره‌ی ثبات، می‌توان در ثباتی خاص داده‌ی مورد نظر را خواند و یا نوشت

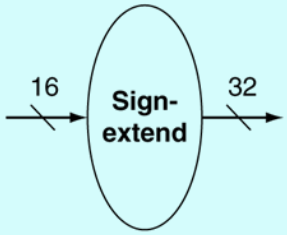
دستورات خواندن و نوشتن در حافظه



- با افزودن رجیستر پایه به بخش ثابت آدرس خانه‌ی مورد نظر در حافظه به دست می‌آید.
- پیش از افزودن عدد ثابت به ثبات پایه لازم است بیت علامت گسترش یابد



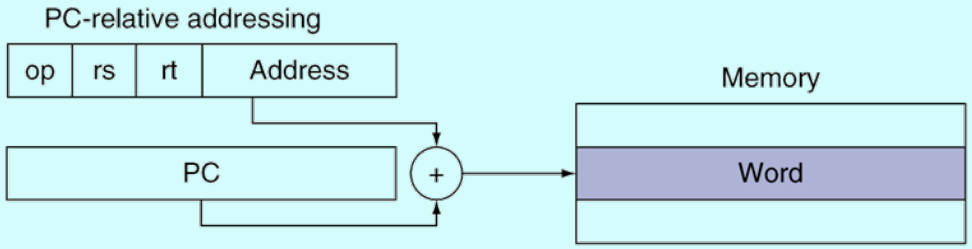
a. Data memory unit



b. Sign extension unit

دستورات پرش شرطی

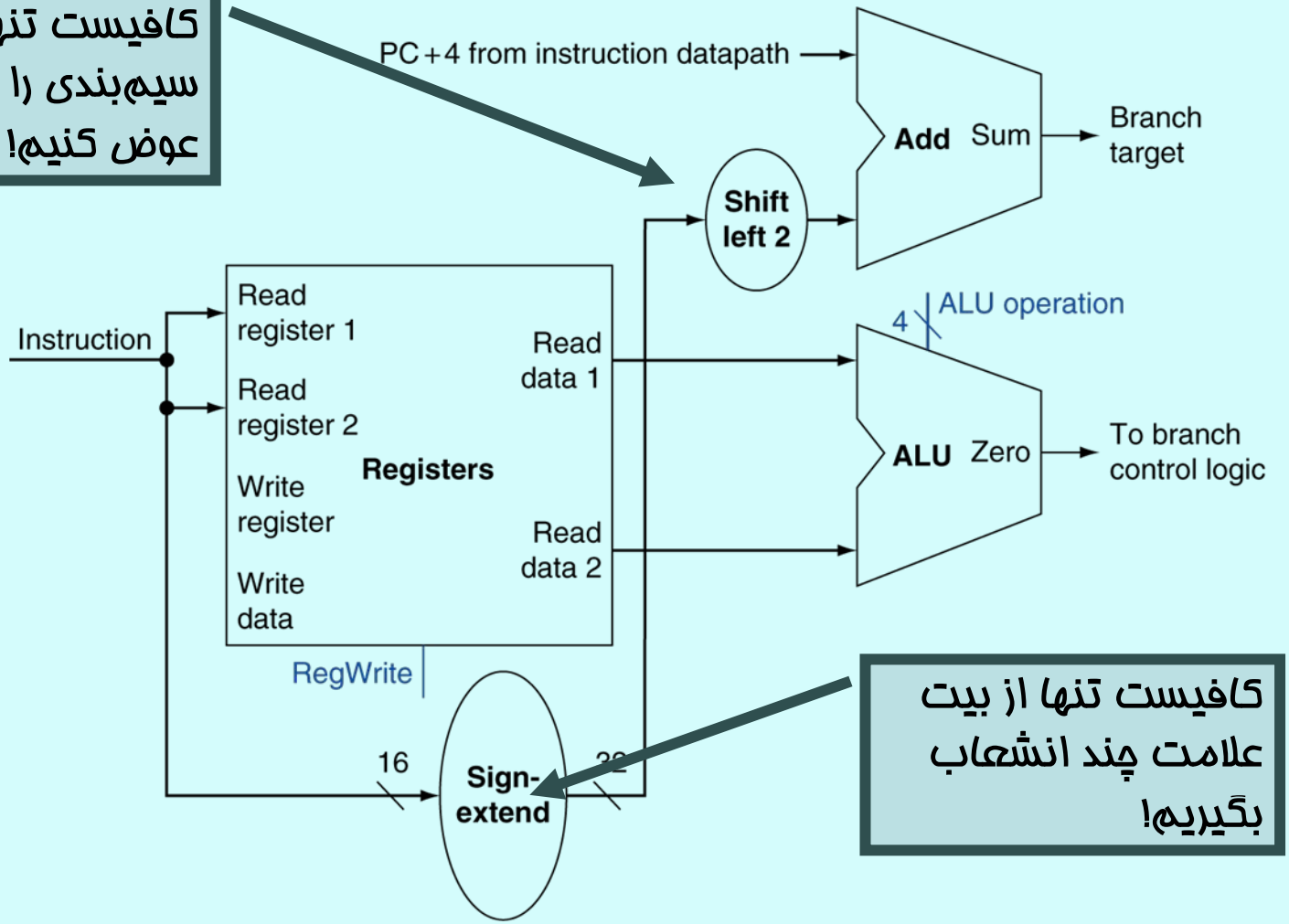
- محتوای ثبات‌ها را می‌خواند
- مقایسه می‌کند
  - با استفاده از ALU و خروجی صفر
- آدرس مقصد را به دست می‌آورد
  - علامت آدرس جابجایی را گسترش می‌دهد
  - دو واحد به سمت چپ شیفت می‌دهد
  - حاصل را به PC+4 اضافه می‌کند



دستورات پرش شرطی (ادامه...)

چگونه می‌توان ساخت افزار گسترش علامت و شیفت را طراحی کرد؟

کافیست تنها سیم‌بندی را عوض کنیم!

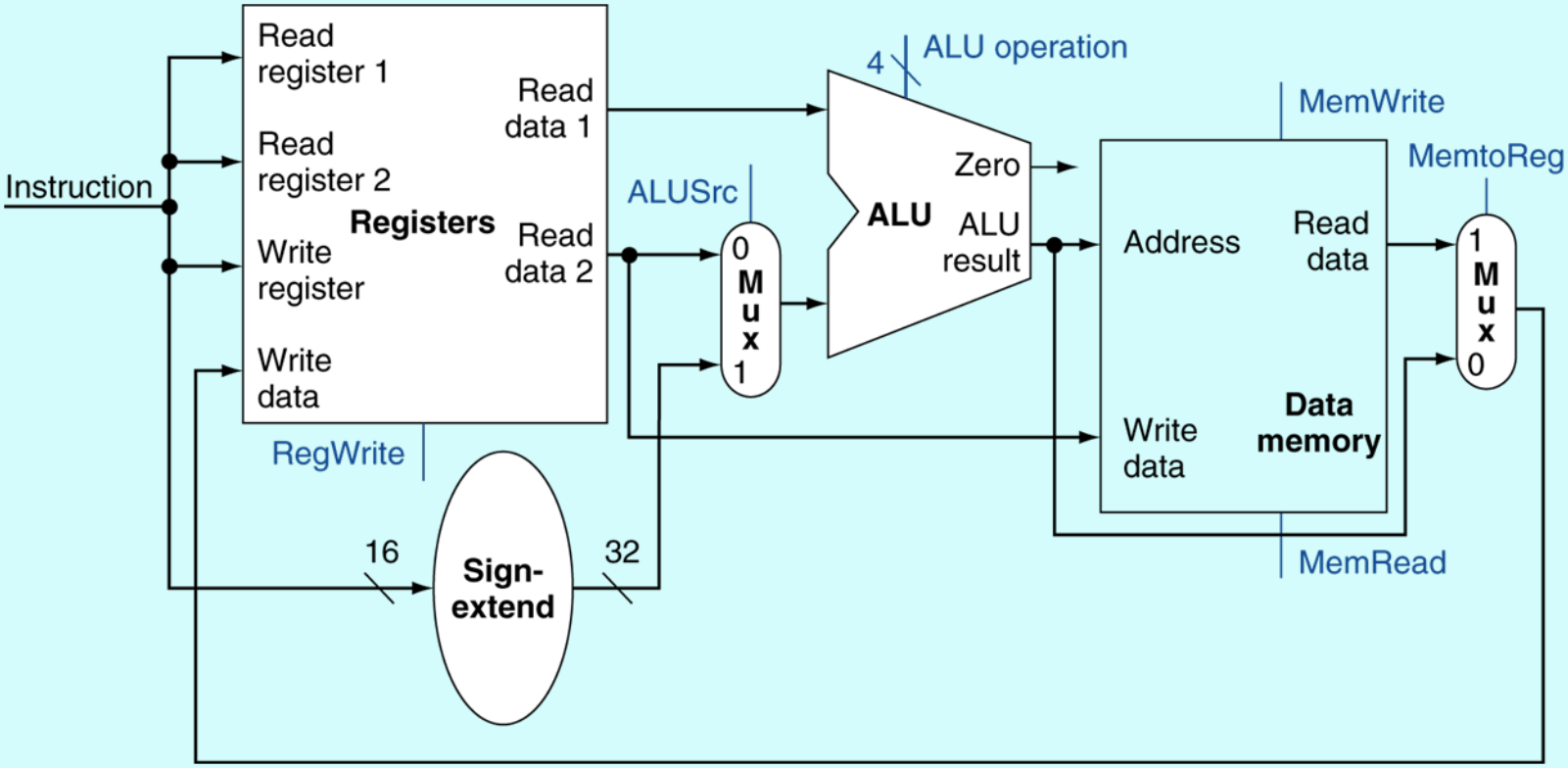


کافیست تنها از بیت علامت چند انشعاب بگیریم!

ترکیب اجزا

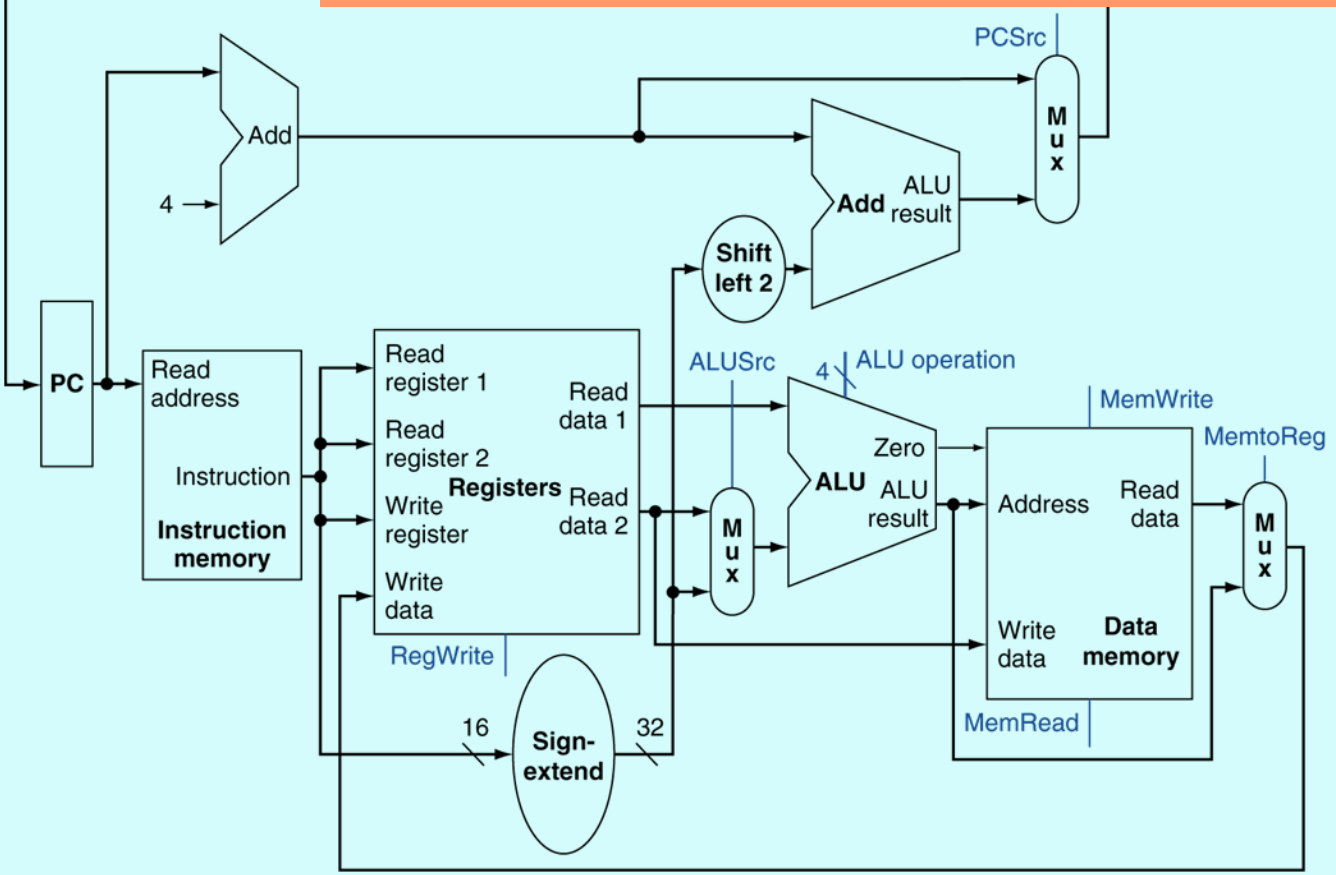
- با ترکیب اجزای مختلف، که برای دستورهای متنوع لازم هستند، یک داده‌گذر ساخته خواهد شد، که برای دستورهای مختلف توسط واحد کنترل هدایت می‌شود.
- ساده‌ترین داده گذر تمام دستورات را در یک سیکل اجرا خواهد کرد، در این صورت هیچ منبعی را نمی‌توان دوبار استفاده کرد.
- در این حالت باید از برخی منابع چند نسخه قرار داد.
- جاهایی که بیش از یک ورودی داریم، برای انتخاب از مالتی‌پلکسر استفاده می‌کنیم.

ترکیب داده‌گذر دستورات حسابی و منطقی و دستورات حافظه



داده‌گذر کامل شده

سخت‌افزار مورد نیاز برای اجرای دستورالعمل‌ها آماده است، تنها بخشی که باقی می‌ماند، آماده کردن سیگنال‌های کنترلی است



سؤال ۱: هر کدام از این سیگنال‌ها، برای چه دستوراتی می‌باید فعال باشند؟

سؤال ۲: چرا باید از دو حافظه‌ی مستقل استفاده کنیم؟؟