معماری کامپیوتر (۱۳۹۱-۱۱-۱۳)
جلسه‌ی نهم

دانشگاه شهید بهشتی
دانشکده‌ی مهندسی برق و کامپیوتر
بهار ۱۳۹۱
احمد محمودی ازناوه

فهرست مطالب

- ضرب
  - ضرب‌کننده‌های آرایه‌ای
  - الگوریتم Booth
- تقسیم

ضرب

مضروب

multiplicand → 1000

multiplier → × 1001

مضروب فیه (ضرب‌کننده)

1000

0000

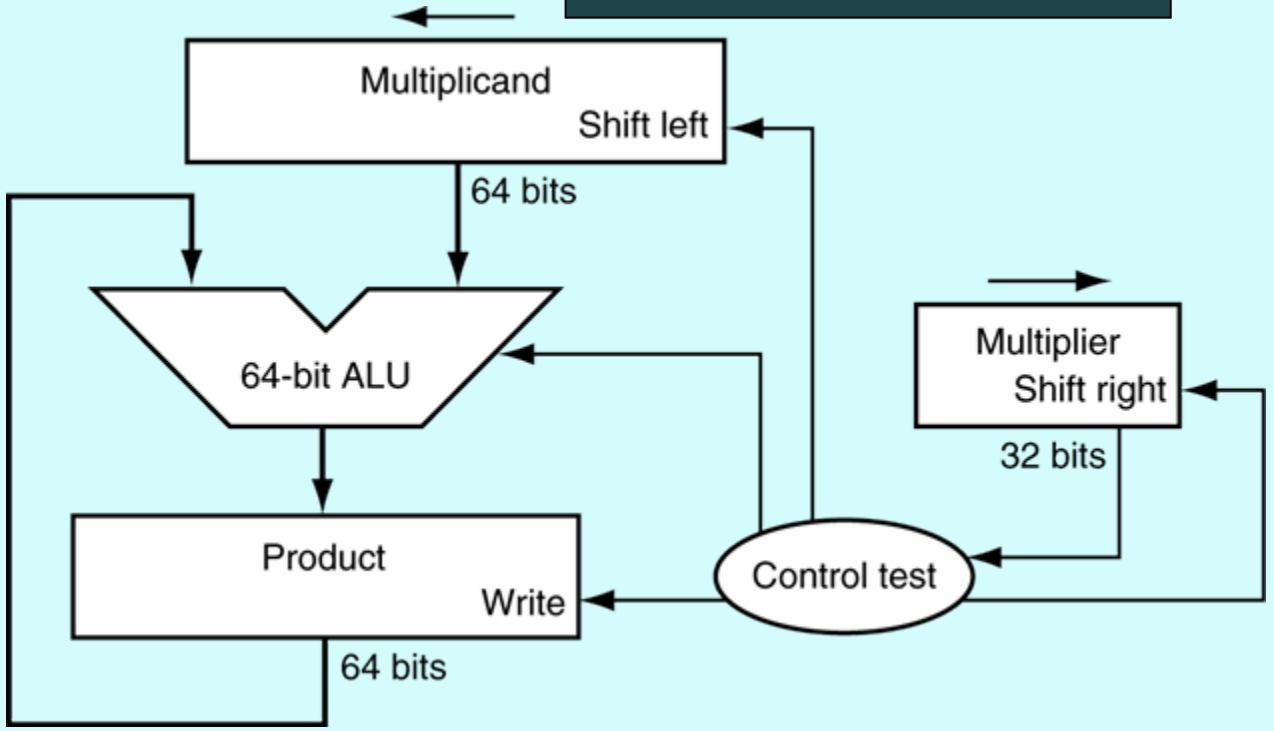
0000

1000

product → 1001000

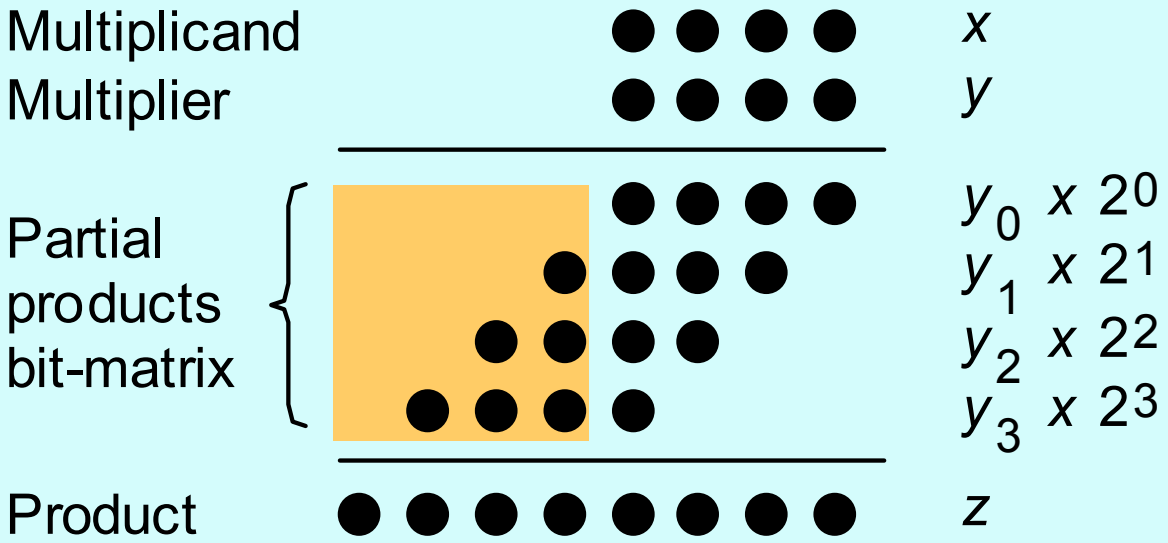
حاصل ضرب

طول حاصلضرب برابر است با جمع طول عملوندها



Dot Notation

ضرب (ادامه...)

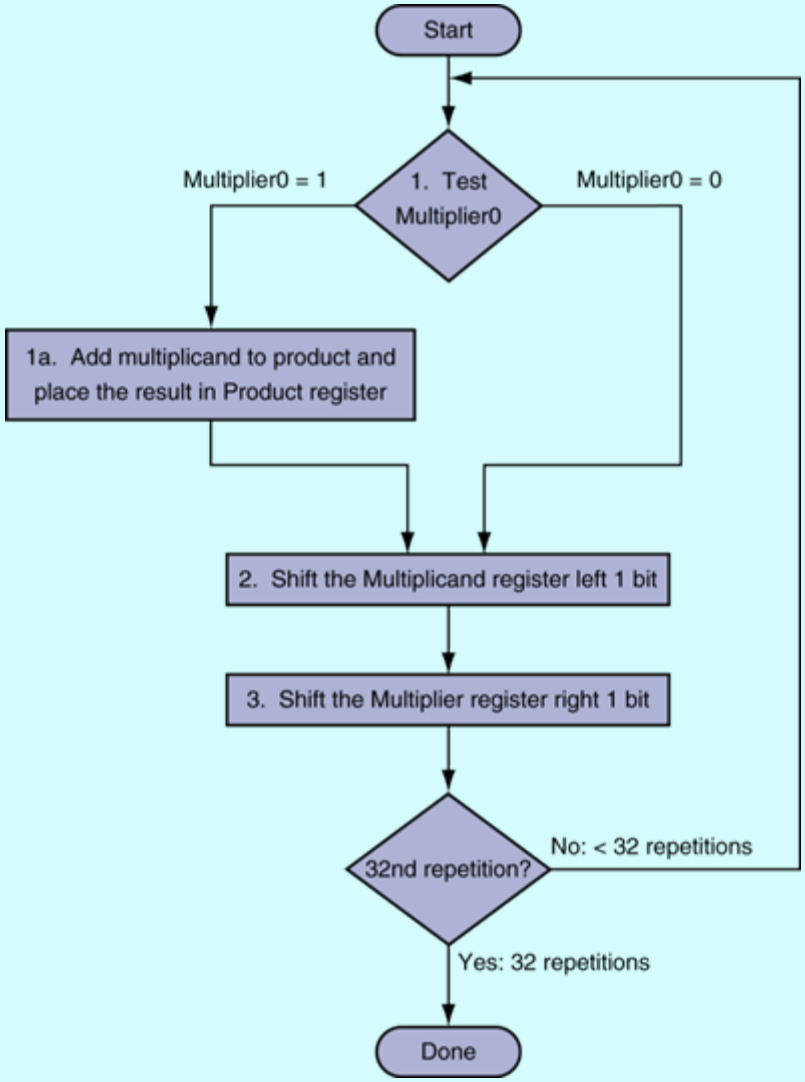


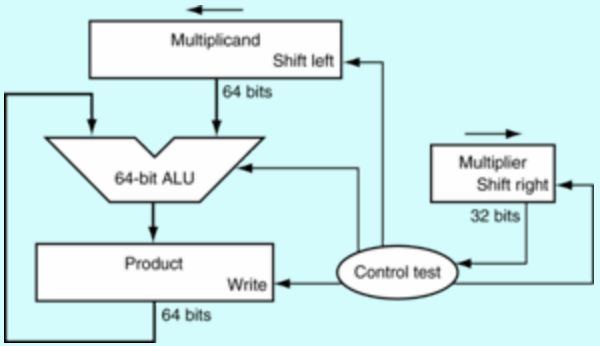
$$z^{(j+1)} = (z^{(j)} + y_j \times 2^k)\, 2^{-1} \quad \text{with } z^{(0)} = 0 \text{ and } z^{(k)} = z$$

$|$—— *add* ——$|$

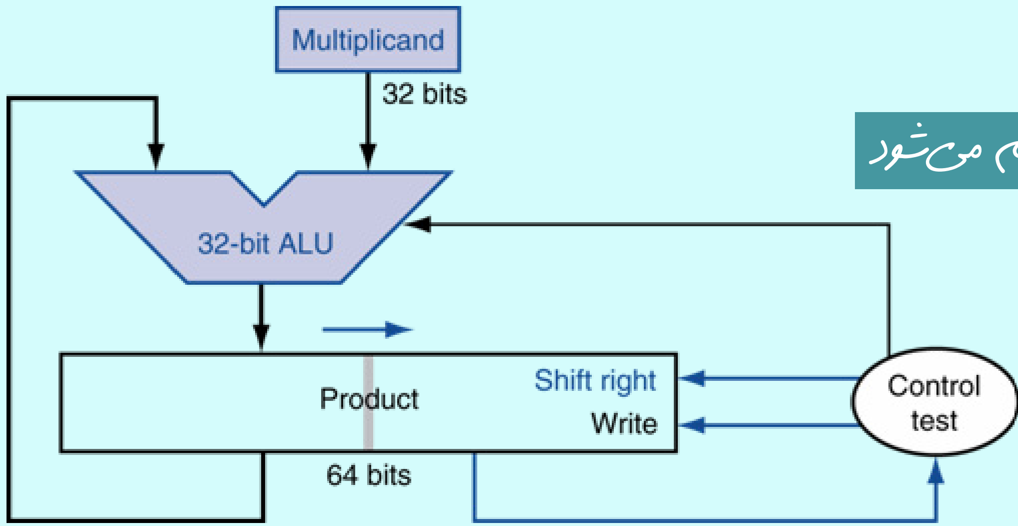
$|$— *shift right* —$|$

سخت افزار ضرب



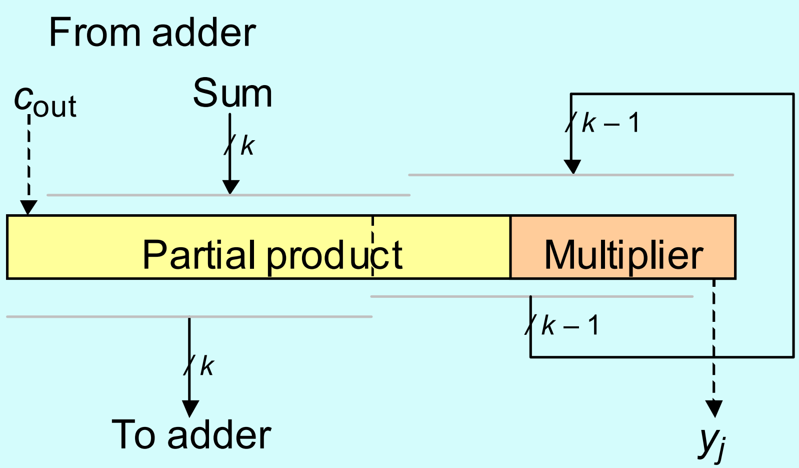


ضرب بهینه‌سازی‌شده

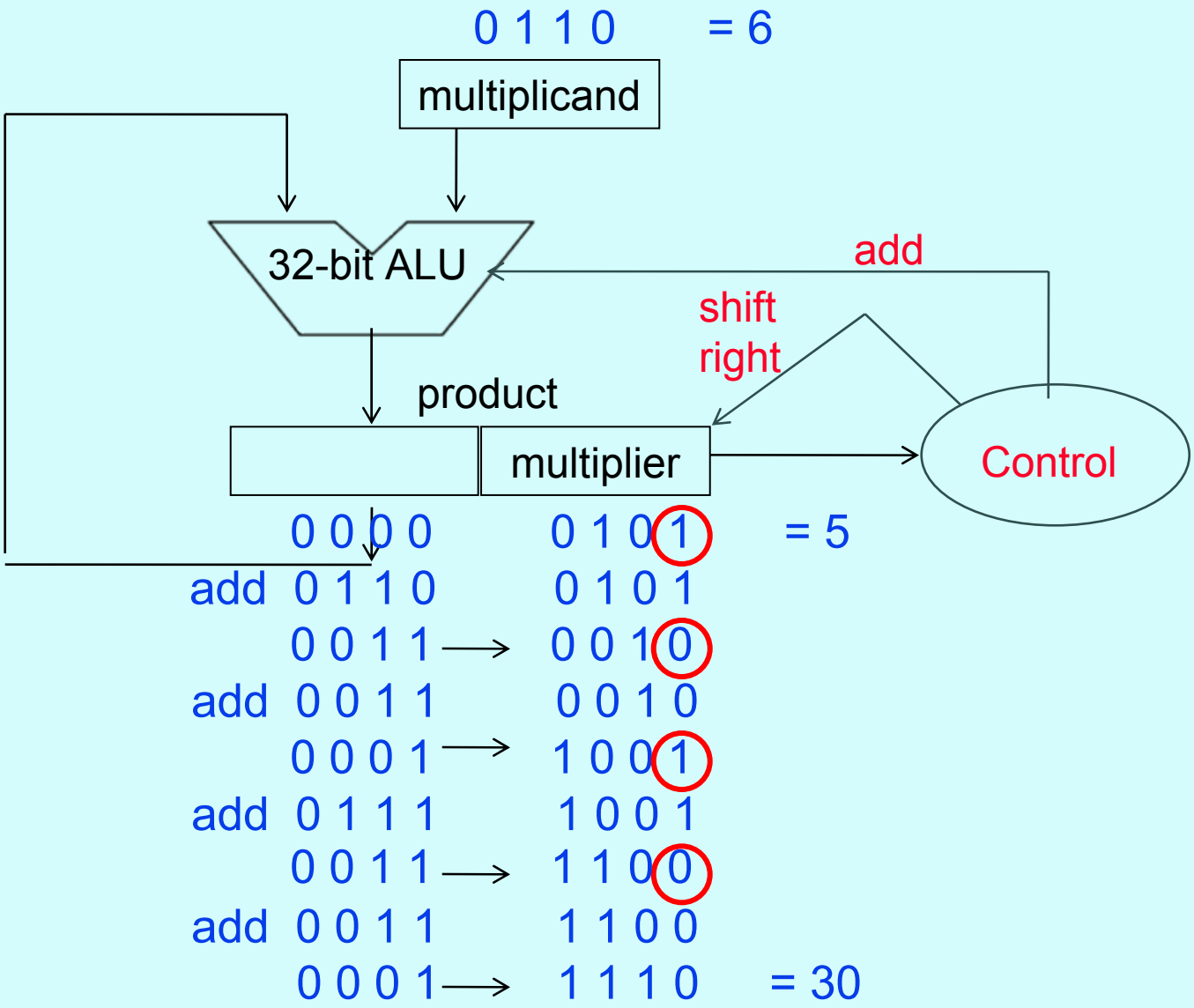


جمع و ضرب به صورت سری انجام می‌شود

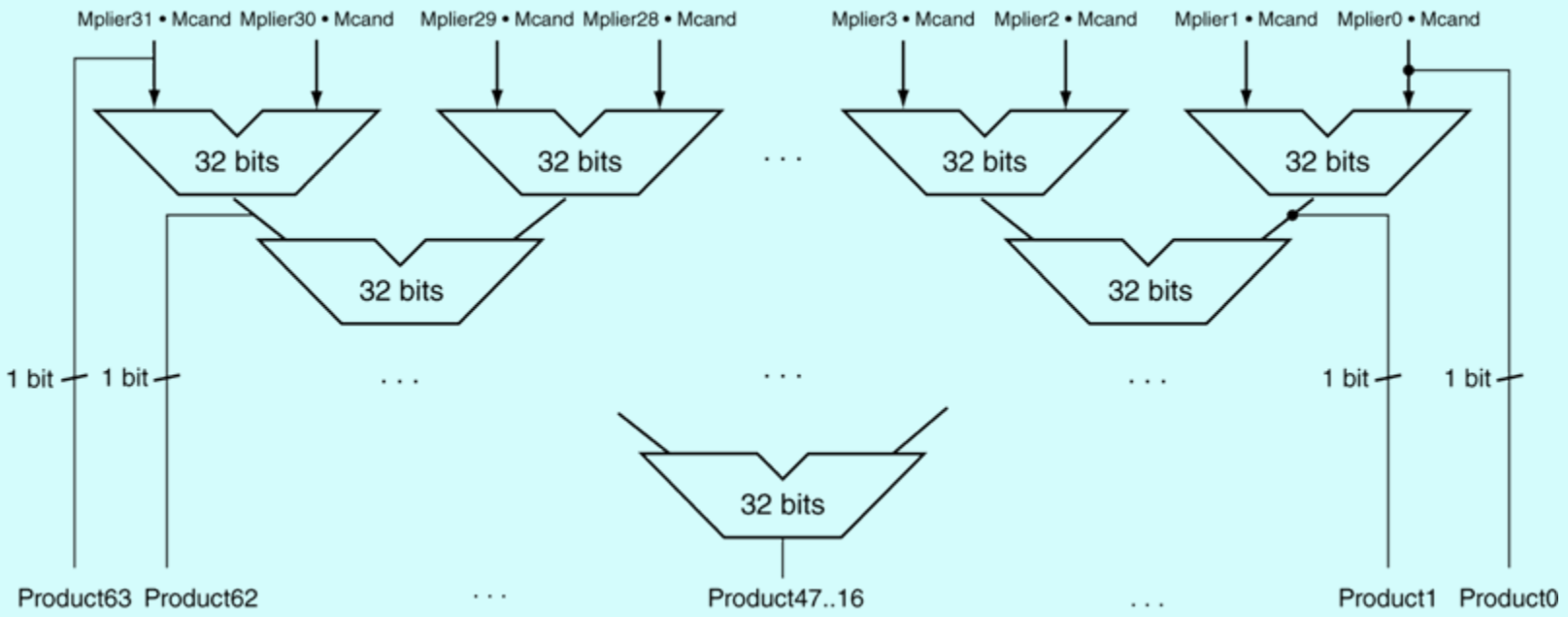
در صورتی که تعداد عملیات ضرب کم باشد، چنین مداری کفایت می‌کند.



مثال:

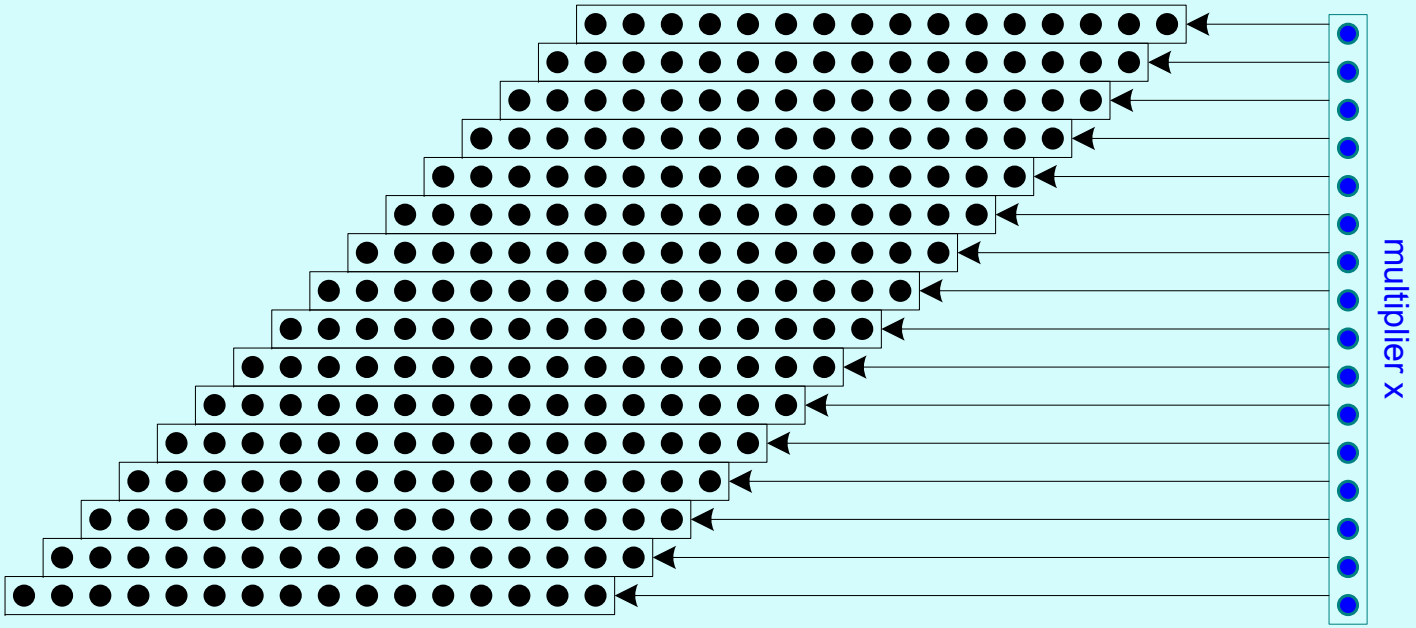

0 1 1 0 = 6
multiplicand
32-bit ALU
add
shift
right
product
multiplier
Control
0 0 0 0 0 1 0 1 = 5
add 0 1 1 0 0 1 0 1
0 0 1 1 → 0 0 1 0
add 0 0 1 1 0 0 1 0
0 0 0 1 → 1 0 0 1
add 0 1 1 1 1 0 0 1
0 0 1 1 → 1 1 0 0
add 0 0 1 1 1 1 0 0
0 0 0 1 → 1 1 1 0 = 30

ضرب‌کننده‌های سریع



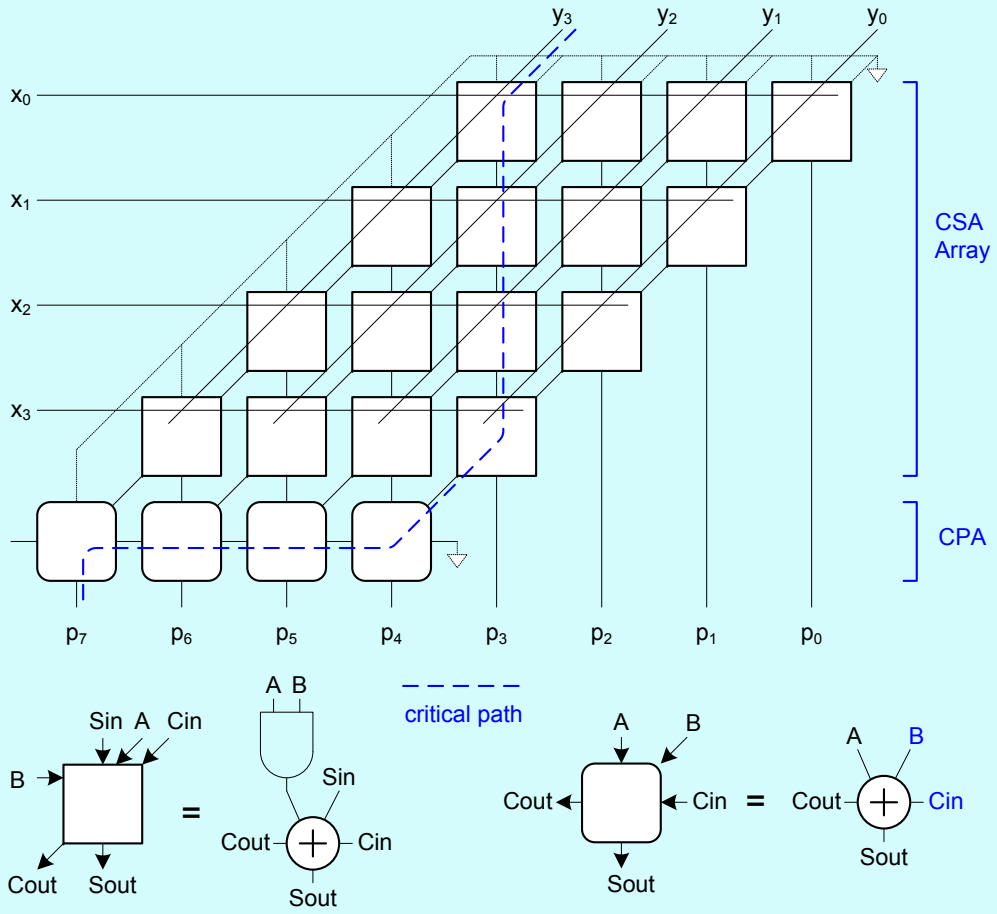
به صورت خط لوله قابل استفاده می‌باشد

ضرب‌کننده‌های سریع (ادامه...)

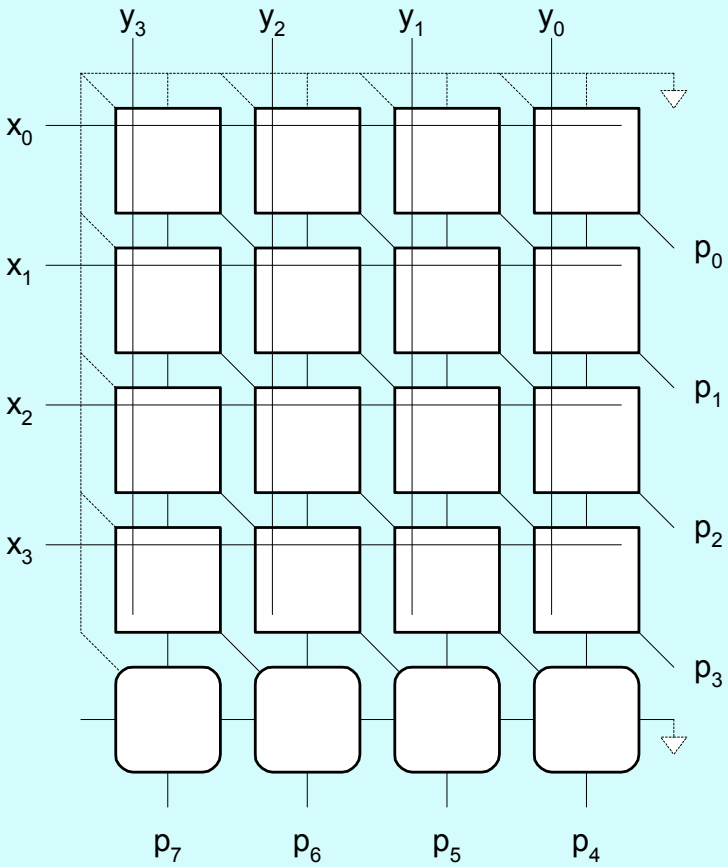


partial products

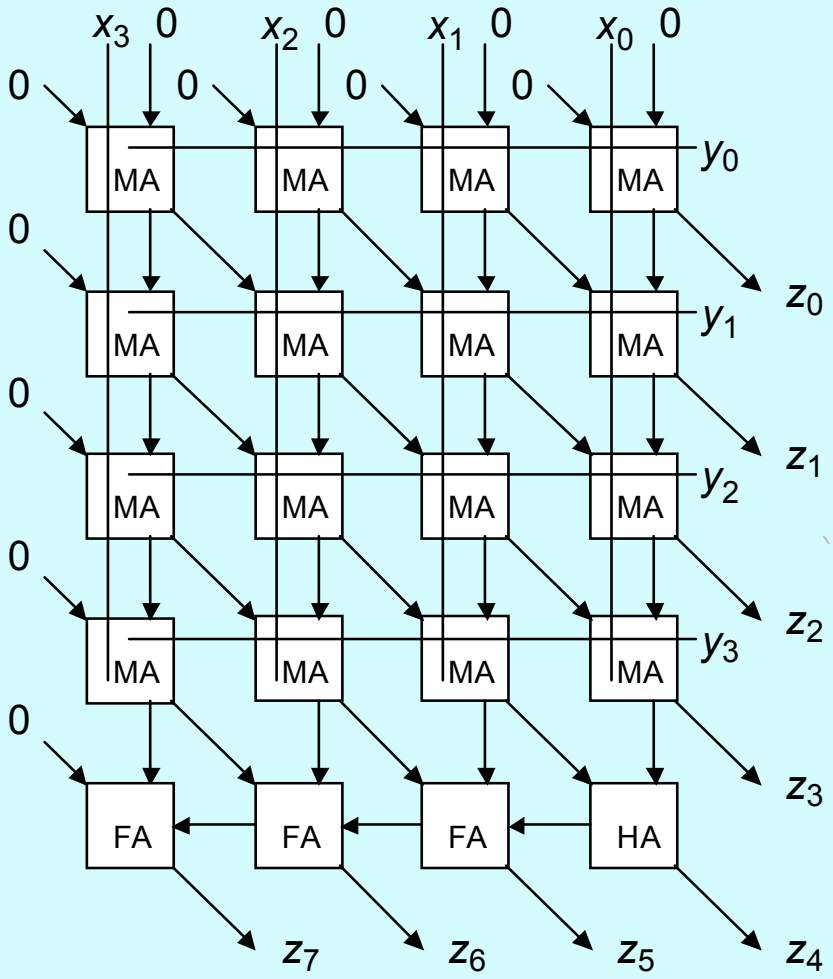
ضرب‌کننده‌ی آرایه‌ای

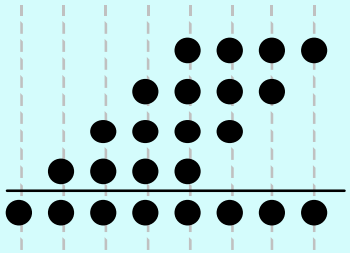


ضرب‌کننده‌ی آرایه‌ای (ادامه...)

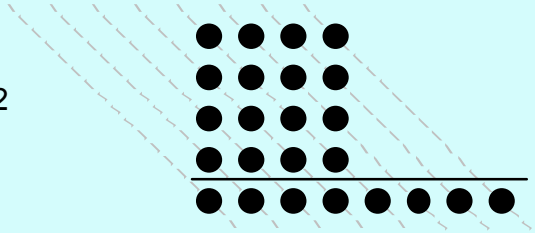


ضرب‌کننده‌های آرایه‌ای (ادامه...)





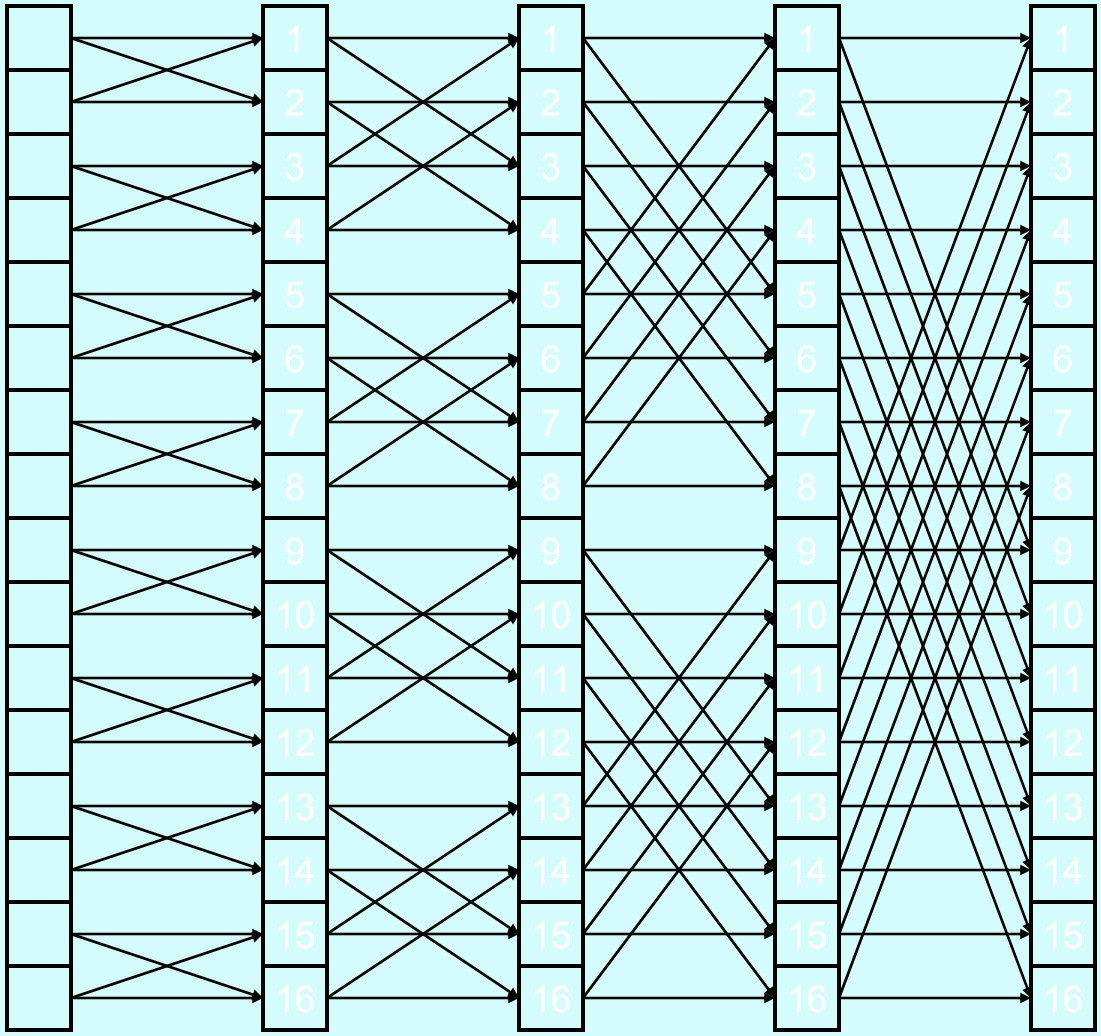
Our original dot-notation representing multiplication



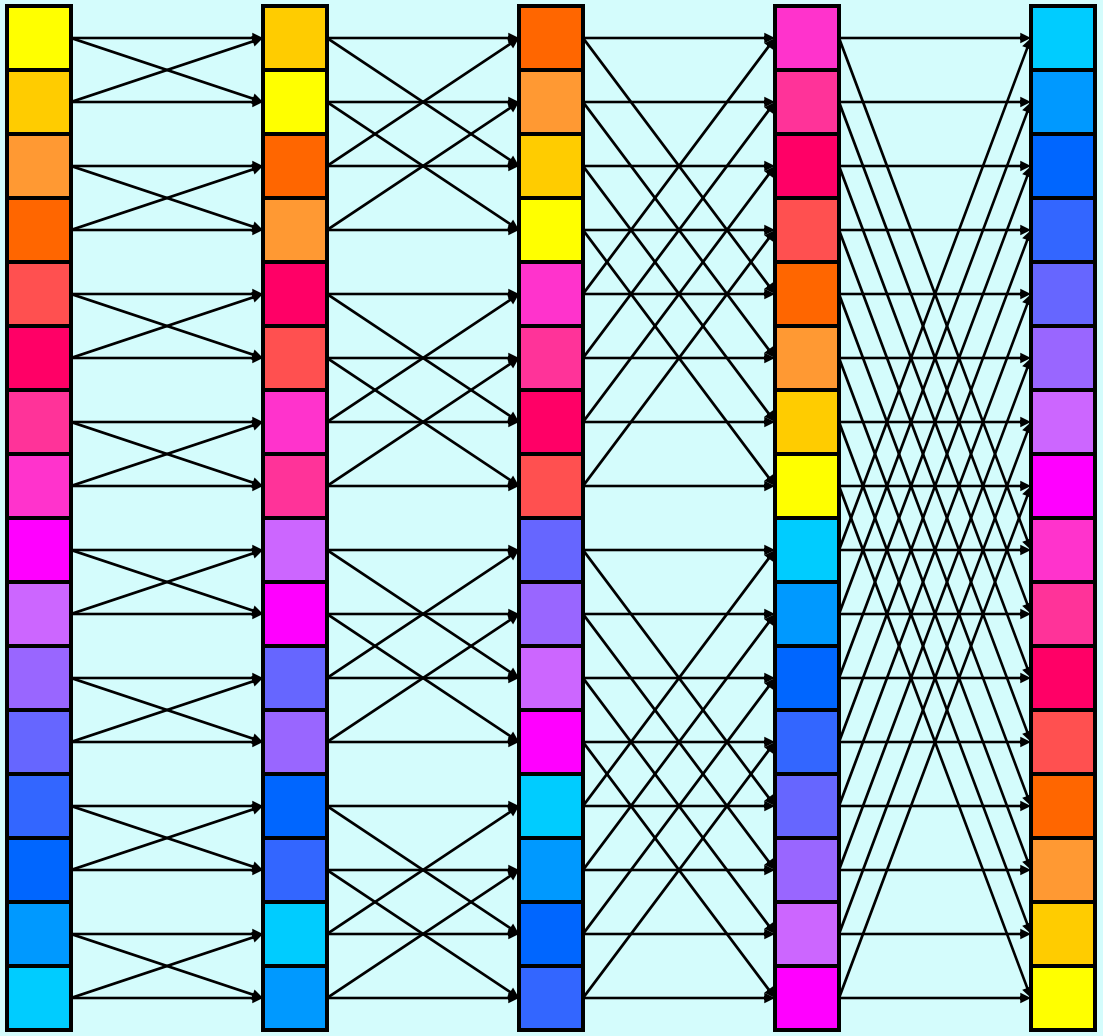
Straightened dots to depict array multiplier to the left

ضرب‌کننده‌ی آرایه‌ای

مدار محاسبه‌ی تبدیل فوریه

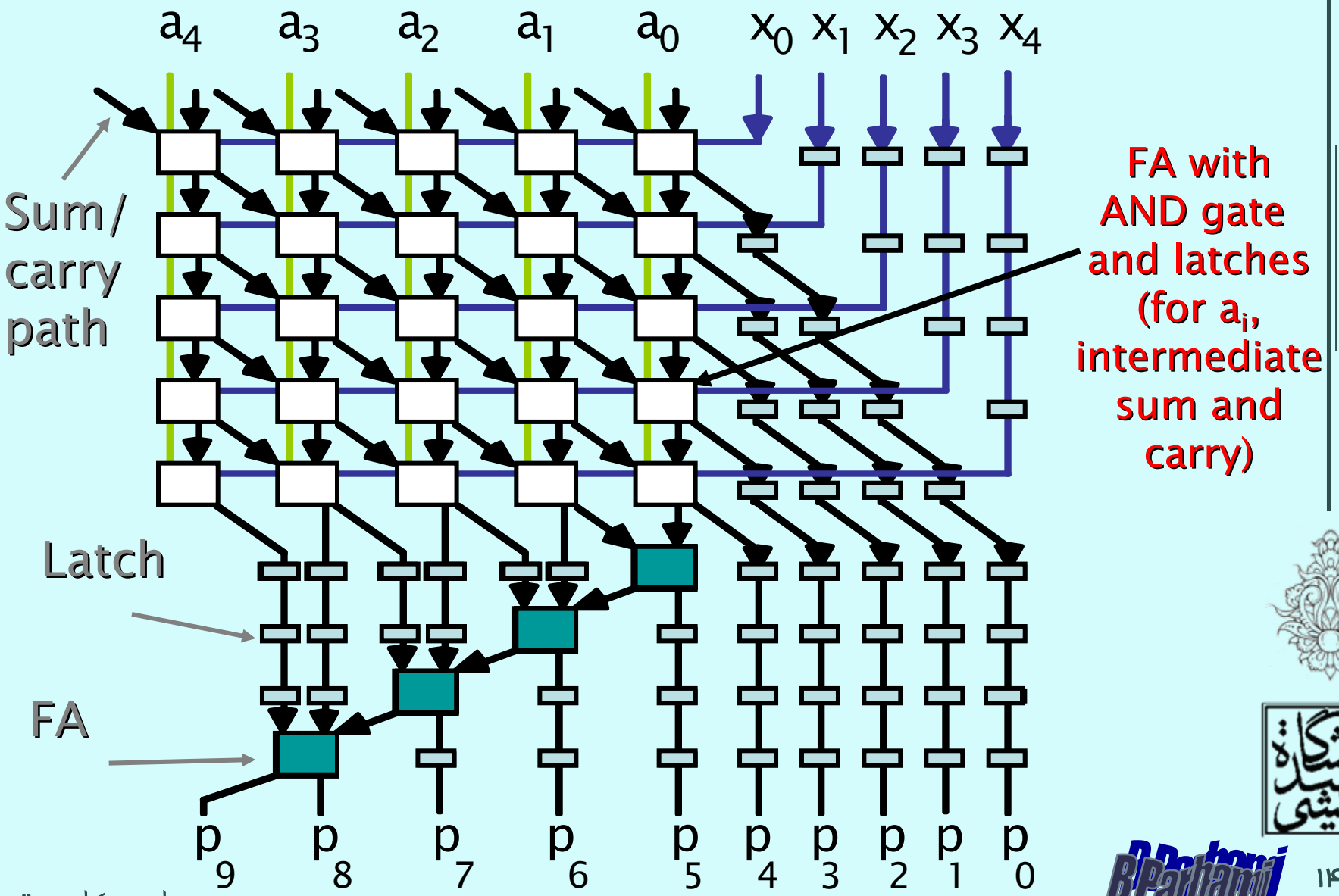


مثالی از خط لوله در اعمال حسابی (تبدیل فوریه)

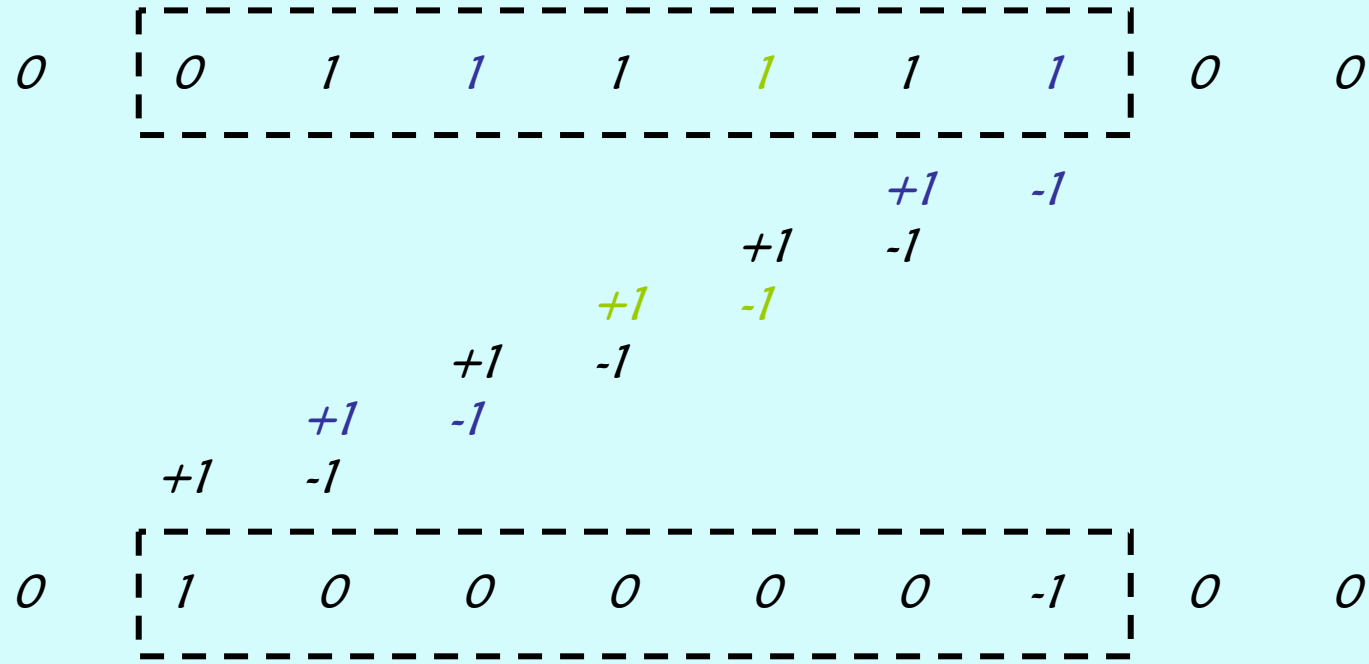


در فصل بعد با مفهوم خط لوله بیشتر آشنا خواهیم شد

خط لوله‌ی ضرب

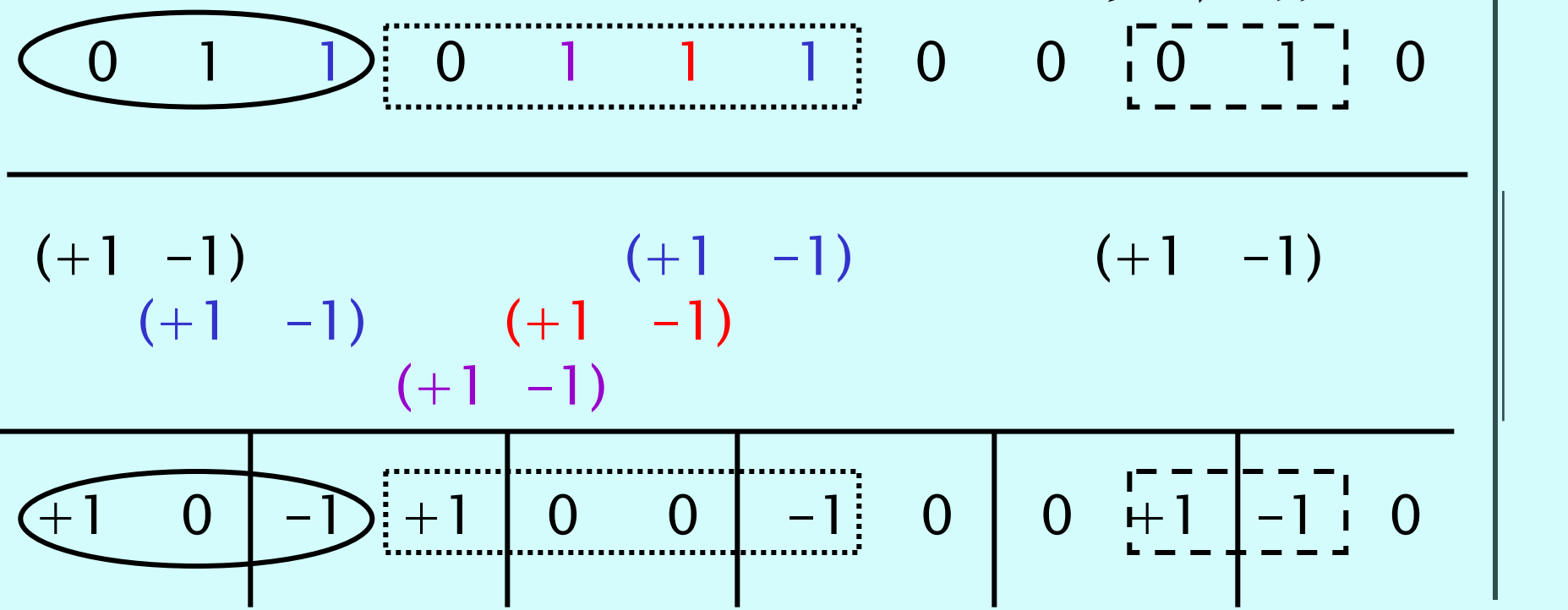


الگوریتم ضرب Booth



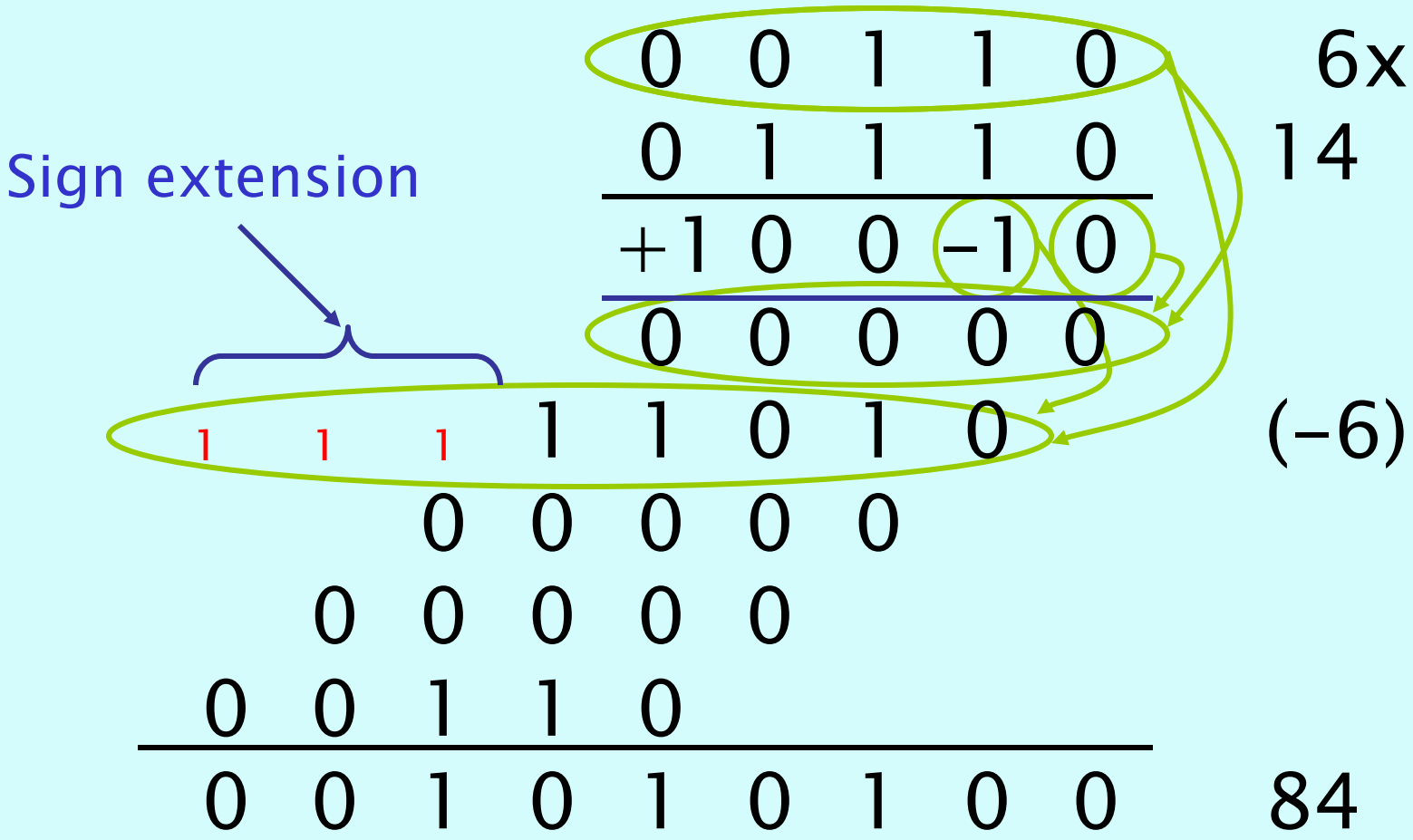
به جای ۱ قرار داده می‌شود ۱ -۱

الگوریتم ضرب Booth (ادامه...)



سطر نخست و سطر پایانی هر دو یک عدد را نشان می‌دهند

الگوریتم ضرب Booth (ادامه...)



ضرب در MIPS

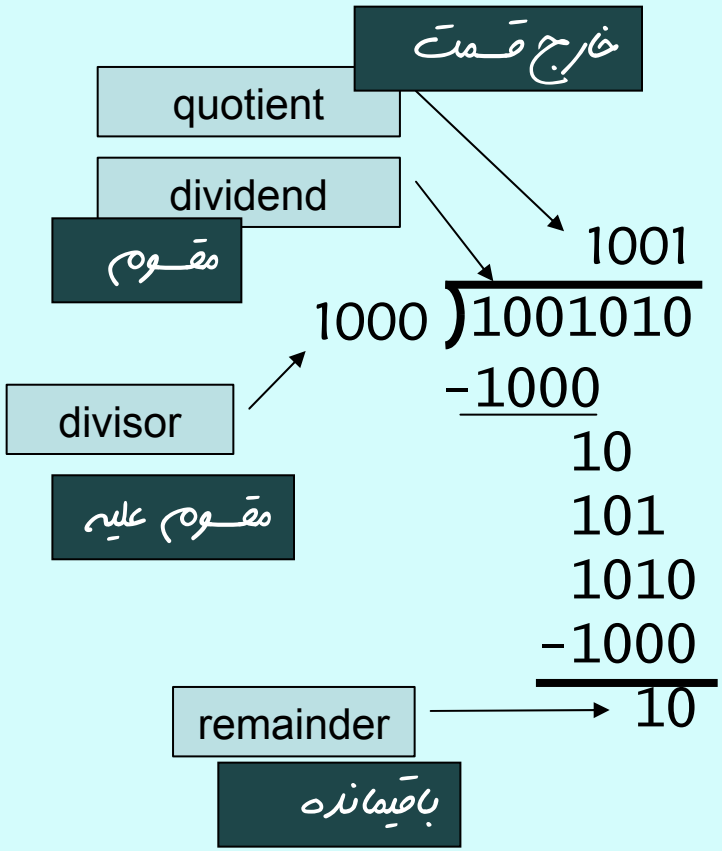
- دو ثبات سی‌ودو بیتی برای ضرب پیش‌بینی شده است:

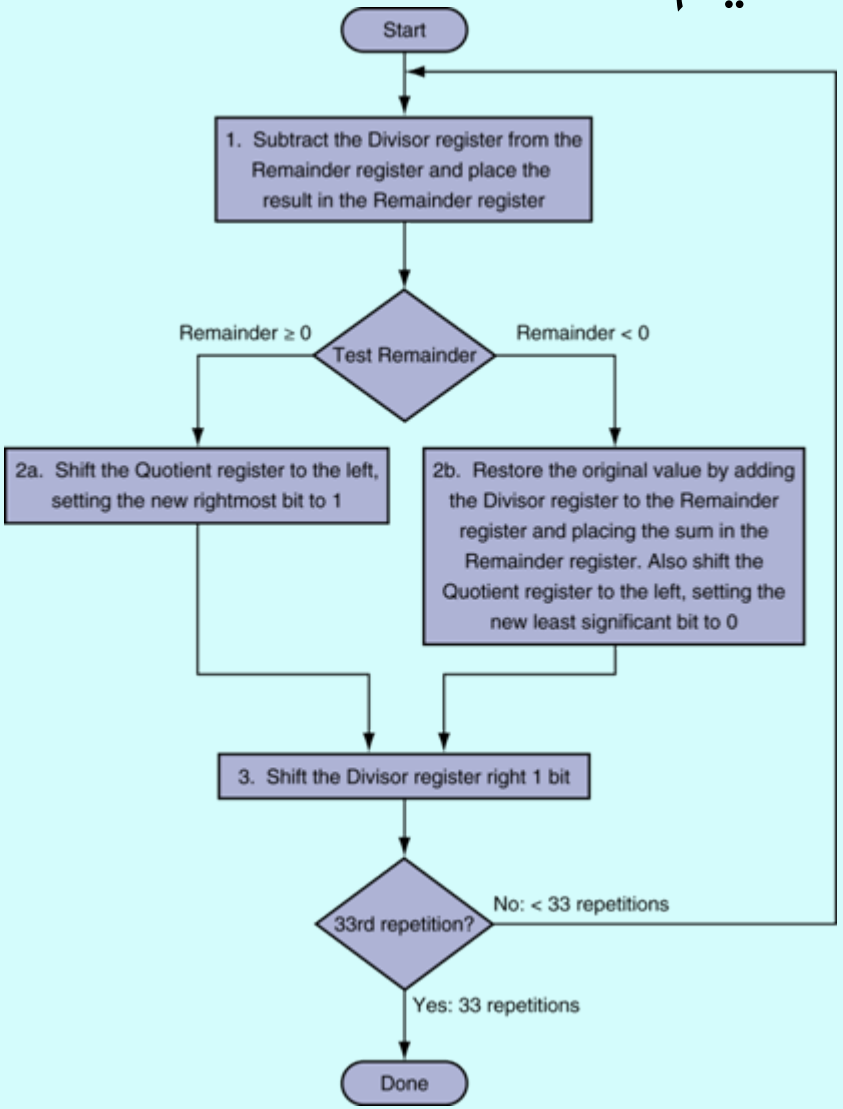
– HI: most-significant 32 bits
– LO: least-significant 32-bits

– دستورات

- `mult rs, rt /  multu rs, rt`
- انجام عملیات ضرب
- `mfhi rd  /  mflo rd`
- انتقال محتوای HI/LO به ثبات‌های چند منظوره
- `mul rd, rs, rt`
- انتقال قسمت کم‌ارزش به rd

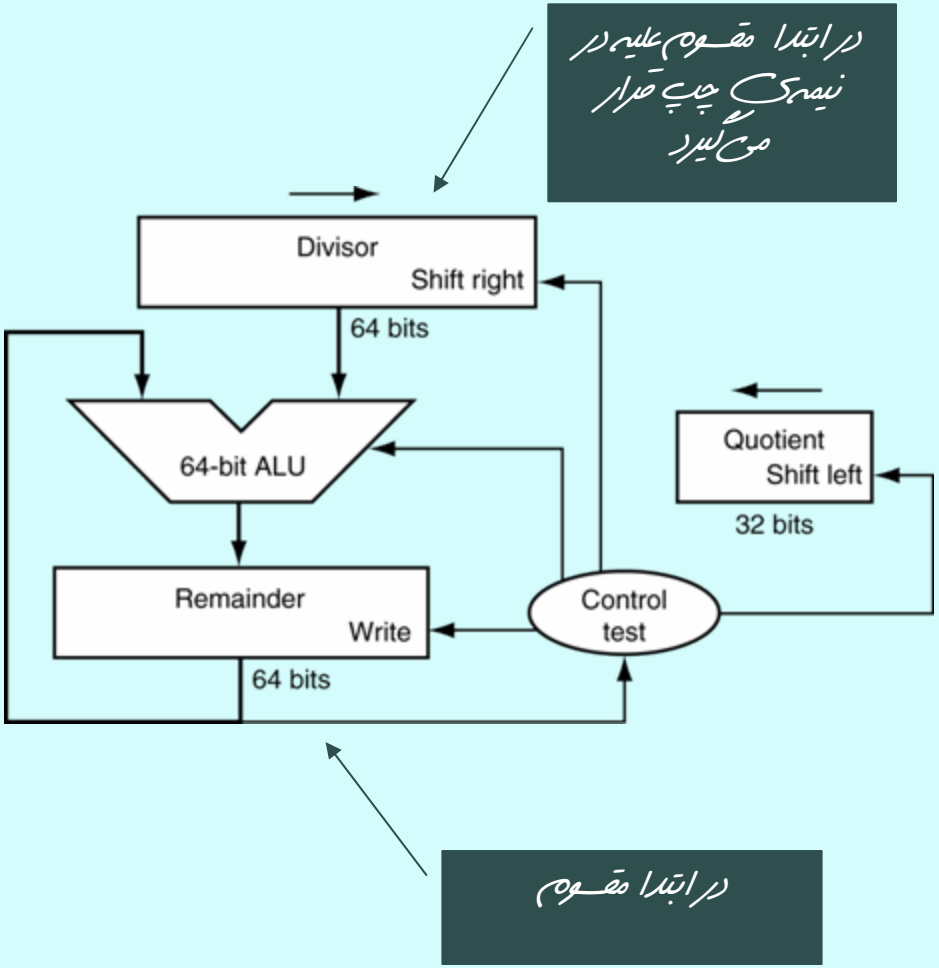
تقسیم



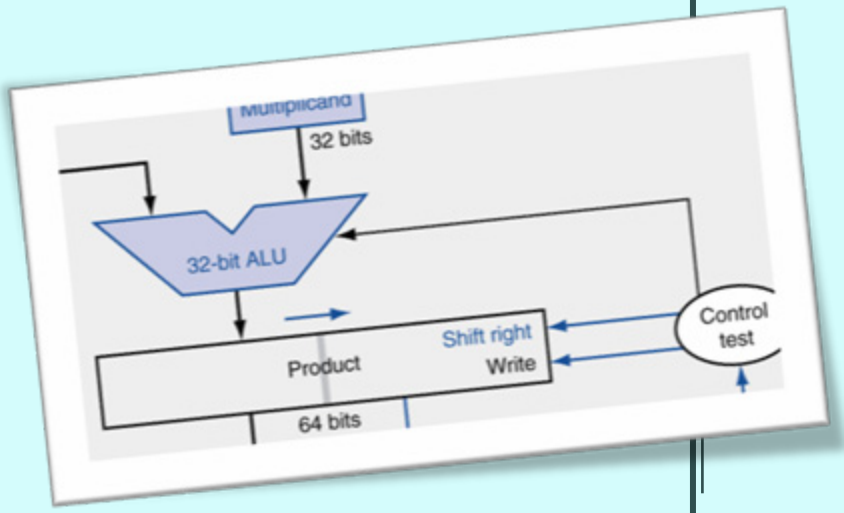


Dividend=Quotient X Divisor + Reminder

سخت‌افزار تقسیم



سخت‌افزار بهینه‌سازی شده



شیبه به مدار ضرب نیست؟!

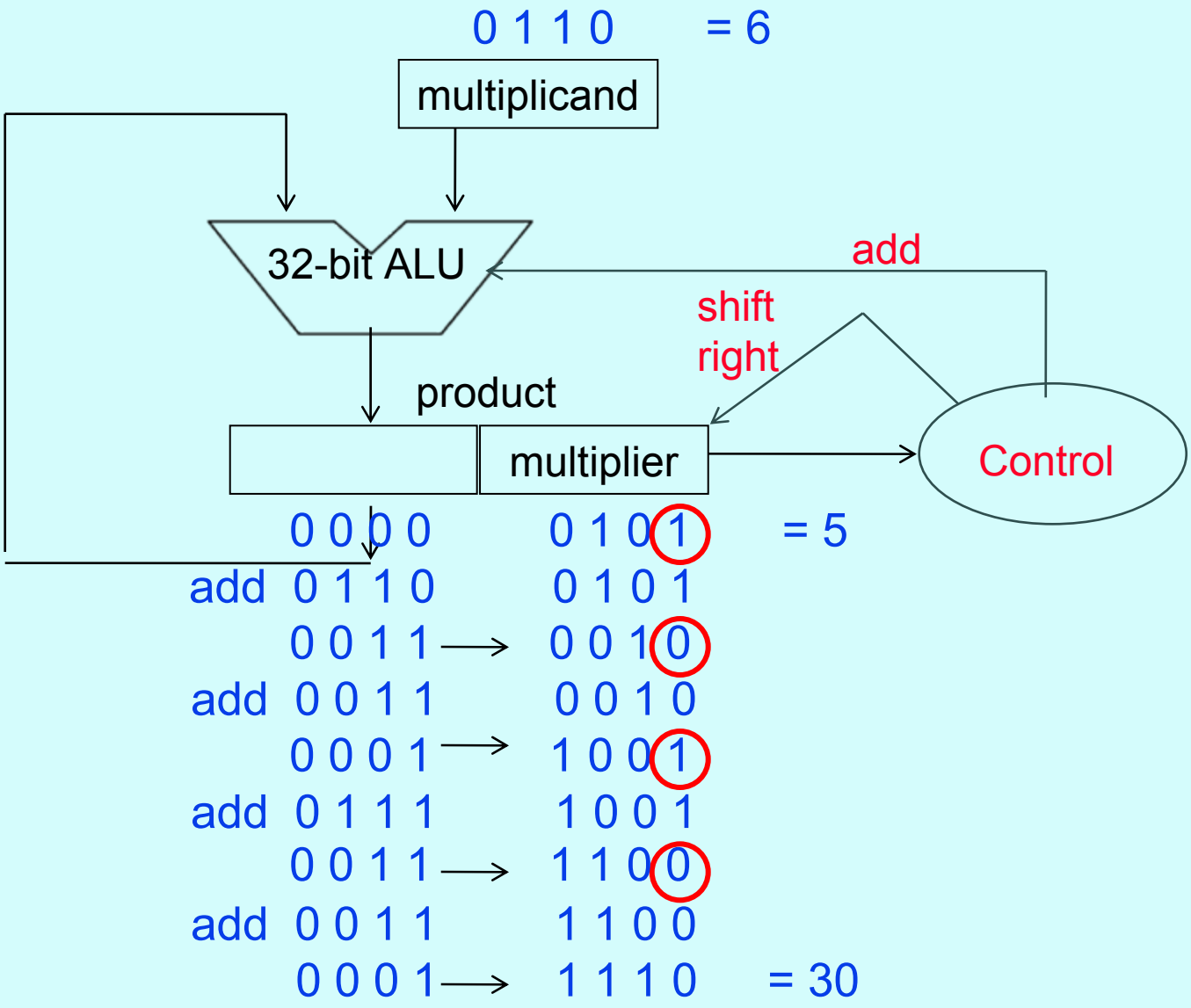
معماری کامپیوتر (۱۳۹۱-۱۱-۱۳)
جلسه‌ی دهم

دانشگاه شهید بهشتی
دانشکده‌ی مهندسی برق و کامپیوتر
بهار ۱۳۹۱
احمد محمودی ازناوه

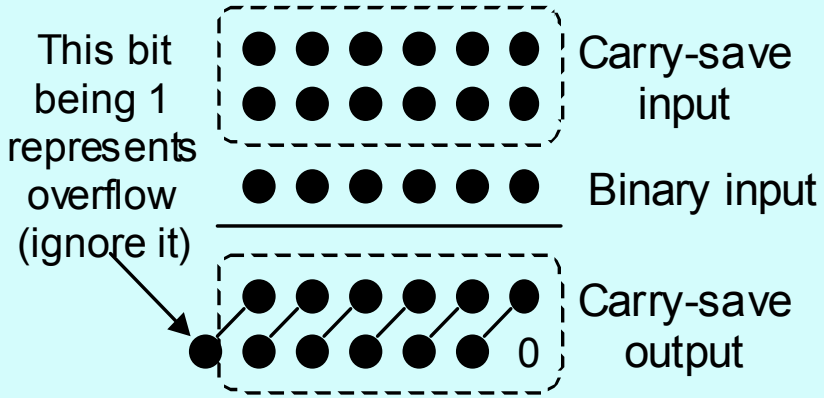
فهرست مطالب

- مروری بر جلسه‌ی پیش
- تقسیم
- ممیز شناور

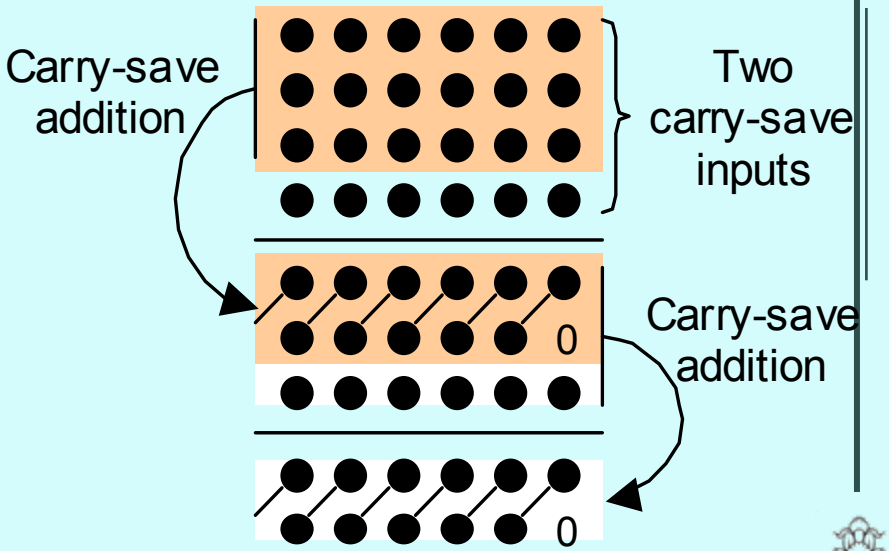
مثال:


0 1 1 0 = 6
multiplicand
32-bit ALU
add
shift
right
product
multiplier
Control
0 0 0 0 0 1 0 1 = 5
add 0 1 1 0 0 1 0 1
0 0 1 1 → 0 0 1 0
add 0 0 1 1 0 0 1 0
0 0 0 1 → 1 0 0 1
add 0 1 1 1 1 0 0 1
0 0 1 1 → 1 1 0 0
add 0 0 1 1 1 1 0 0
0 0 0 1 → 1 1 1 0 = 30

Carry save adder

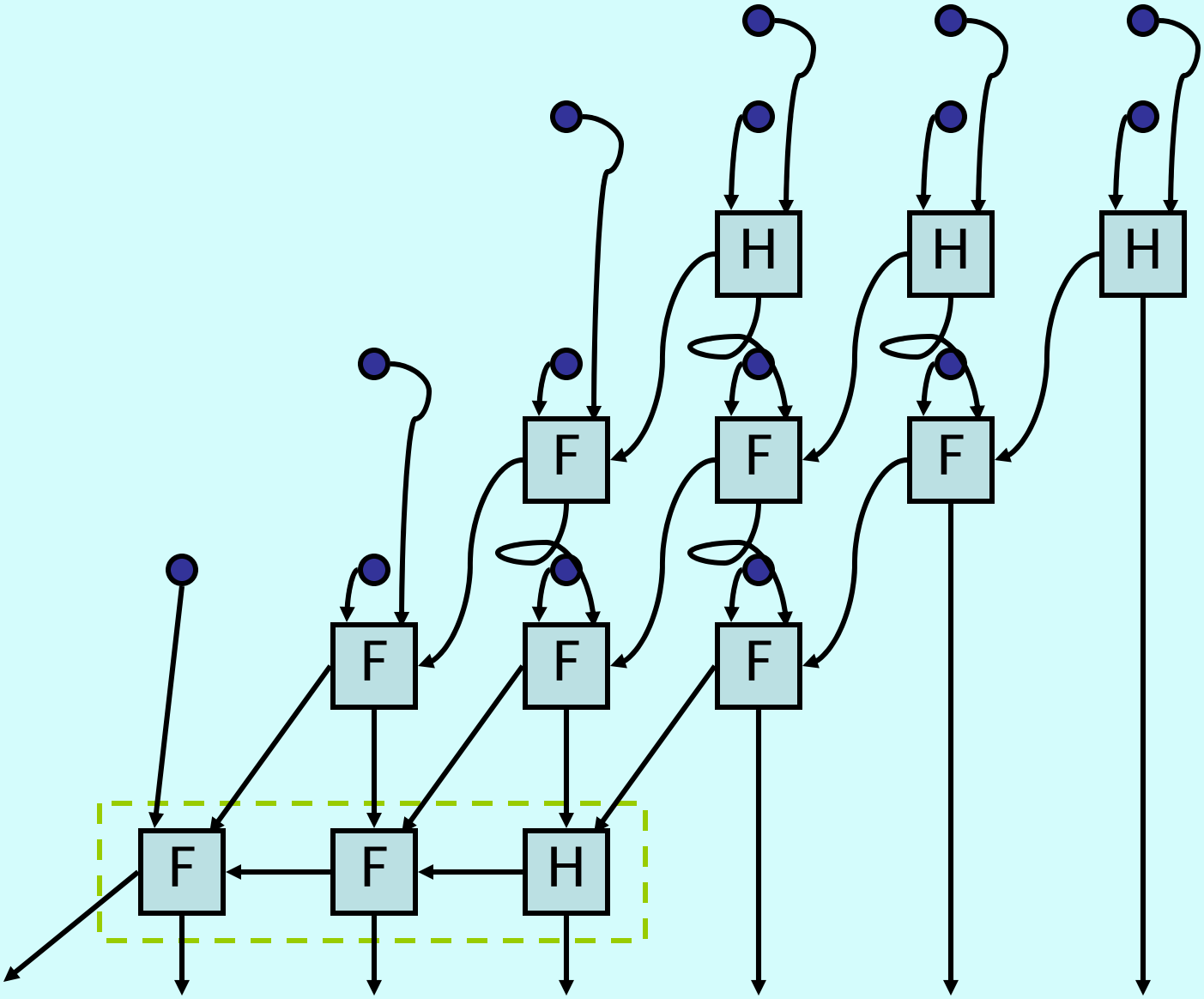


a. Carry-save addition.

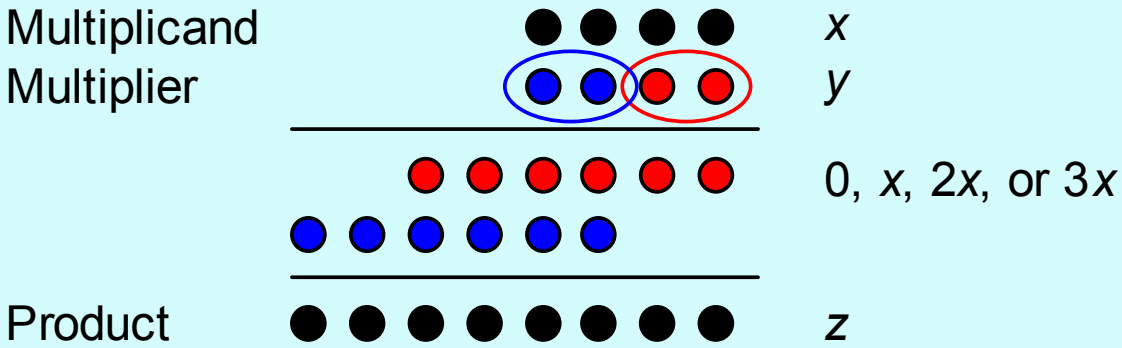


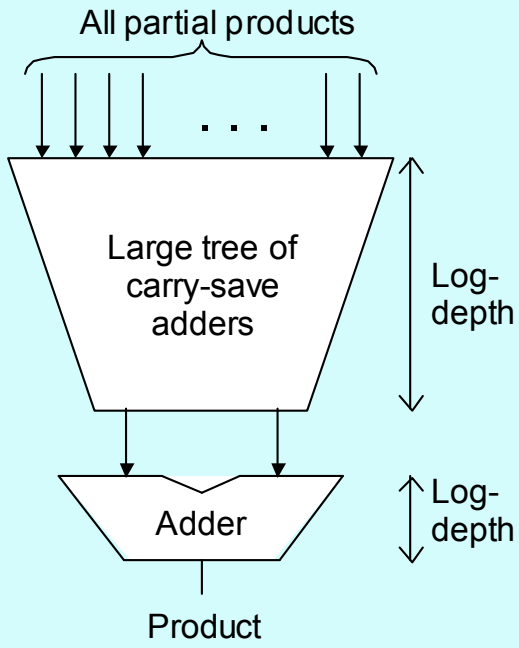
b. Adding two carry-save numbers.

Carry save adder

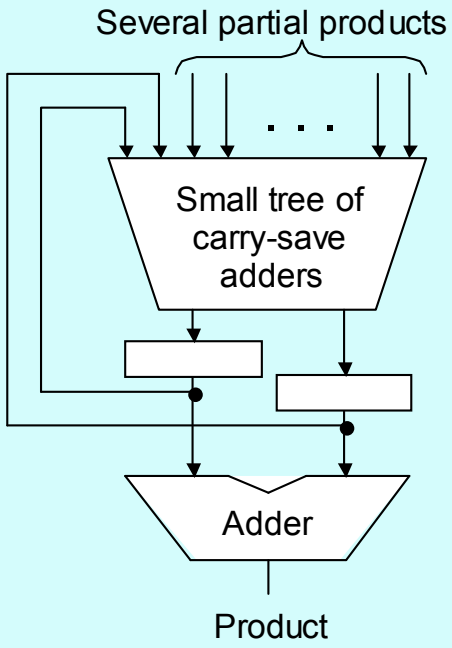


ضرب‌کننده‌های سریع (ادامه...)



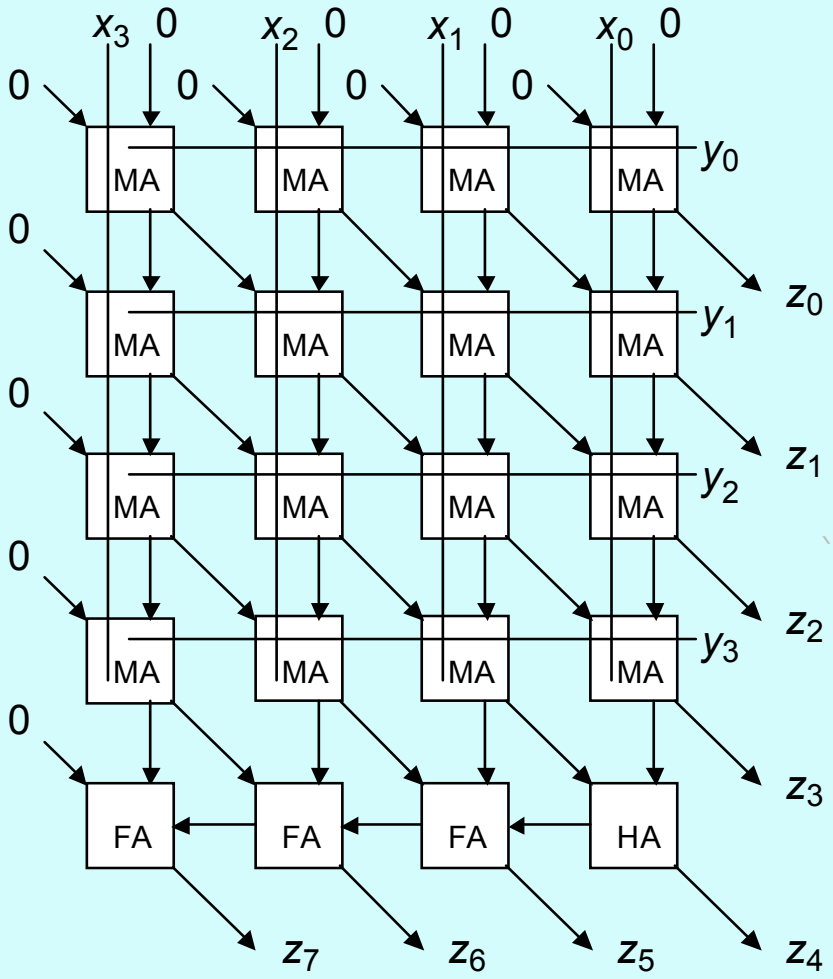


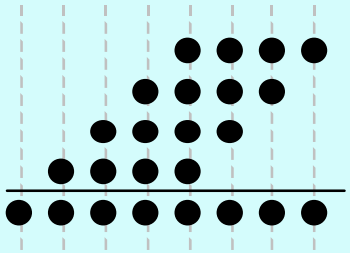
(a) Full-tree multiplier



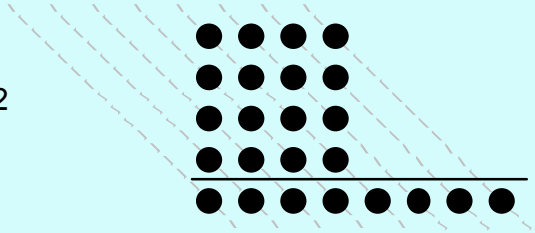
(b) Partial-tree multiplier

ضرب‌کننده‌های آرایه‌ای (ادامه...)





Our original dot-notation representing multiplication

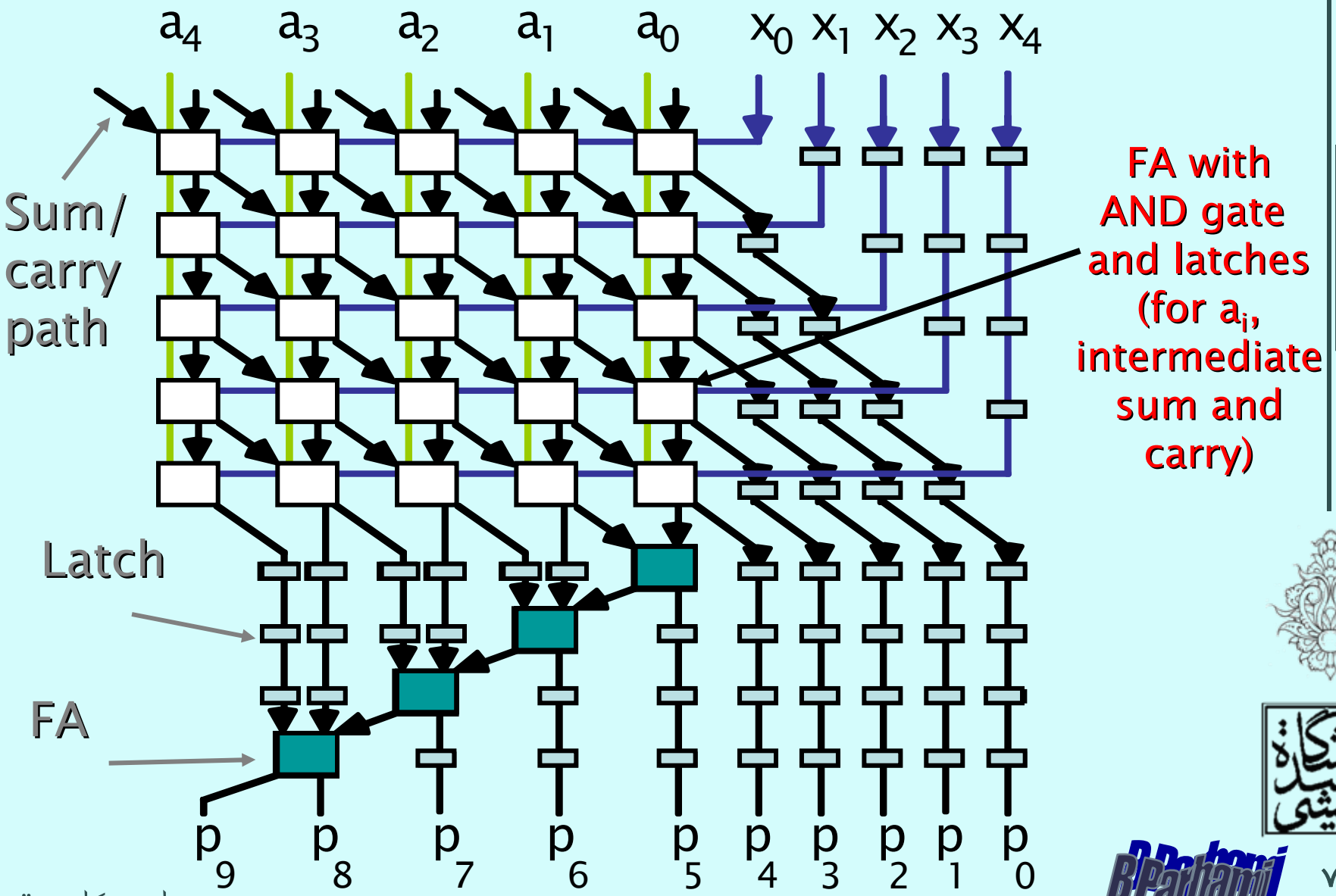


Straightened dots to depict array multiplier to the left

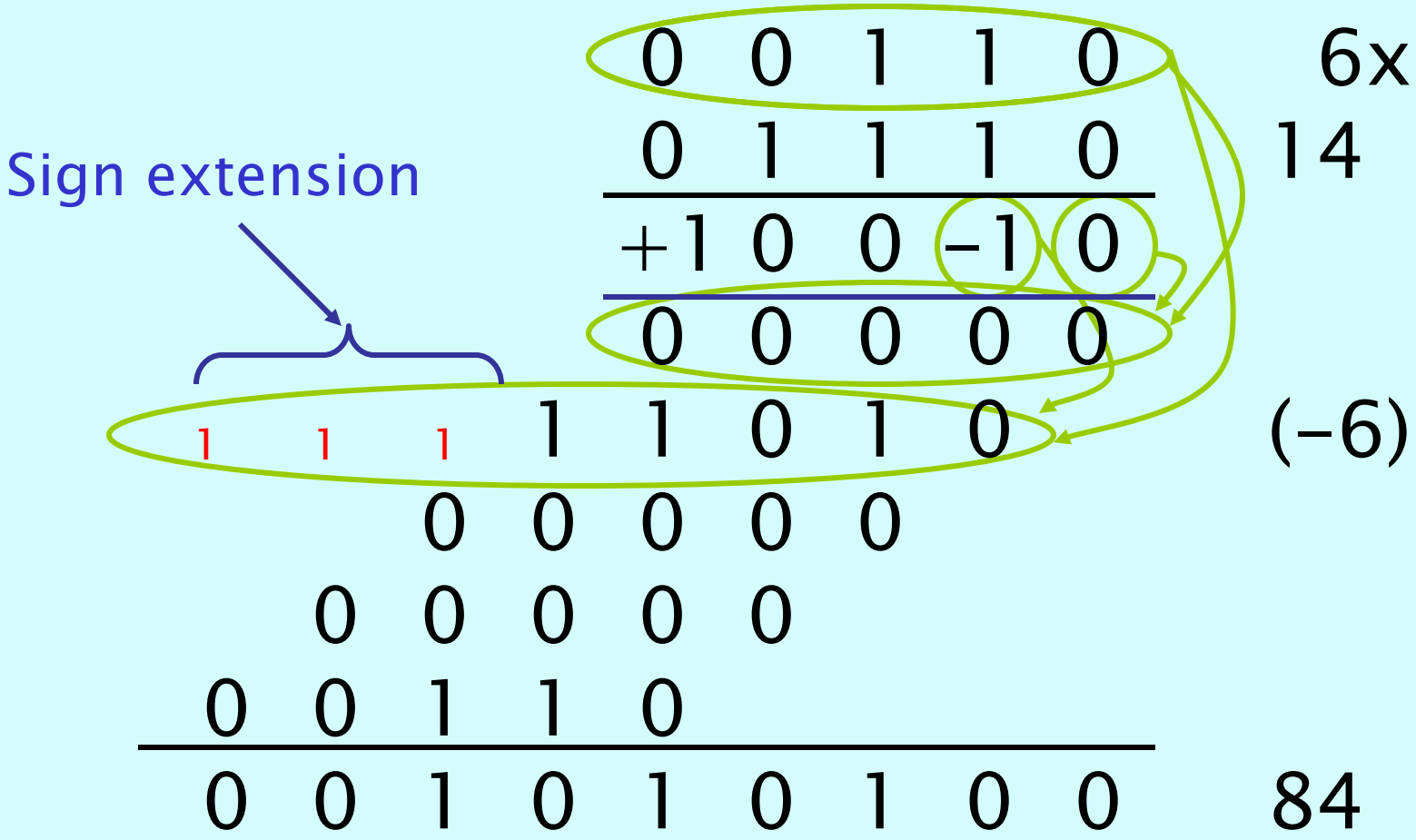
ضرب‌کننده‌ی آرایه‌ای

در فصل بعد با مفهوم خط لوله بیشتر آشنا خواهیم شد

خط لوله‌ی ضرب



الگوریتم ضرب Booth (ادامه...)



الگوریتم ضرب Booth (ادامه...)

x_i	x_{i-1}	Operation	Comments	y_i
0	0	shift only	string of zeros	0
1	1	shift only	string of ones	0
1	0	subtract and shift	beginning of a string of ones	$\bar{1}$
0	1	add and shift	end of a string of ones	1

-6 x 6 (1010 x 0110)= -36

1111 1010	x 0	0000 0000
1111 0100	x –1	0000 1100
1110 1000	x 0	0000 0000
1101 0000	x +1	1101 0000
	Final Sum:	1101 1100 (-36)

الگوریتم ضرب Booth (ادامه...)

-6 x -2 = 12

1010 x 1110

1111 1010	x 0	0000 0000
1111 0100	x –1	0000 1100
1110 1000	x 0	0000 0000
1101 0000	x 0	0000 0000
	Final Sum:	0000 1100 (12)

ضرب در MIPS

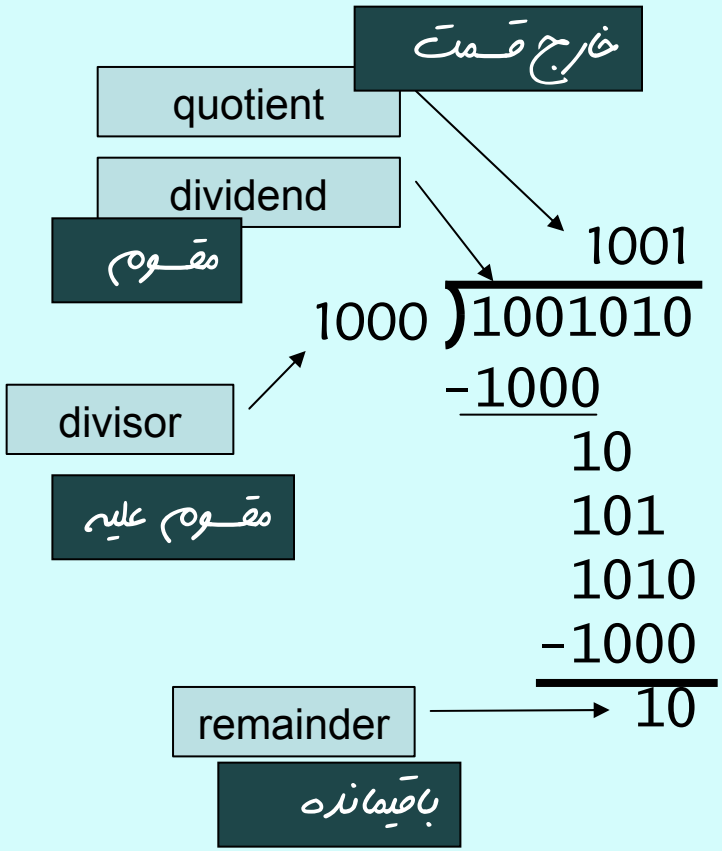
- دو ثبات سی‌ودو بیتی برای ضرب پیش‌بینی شده است:

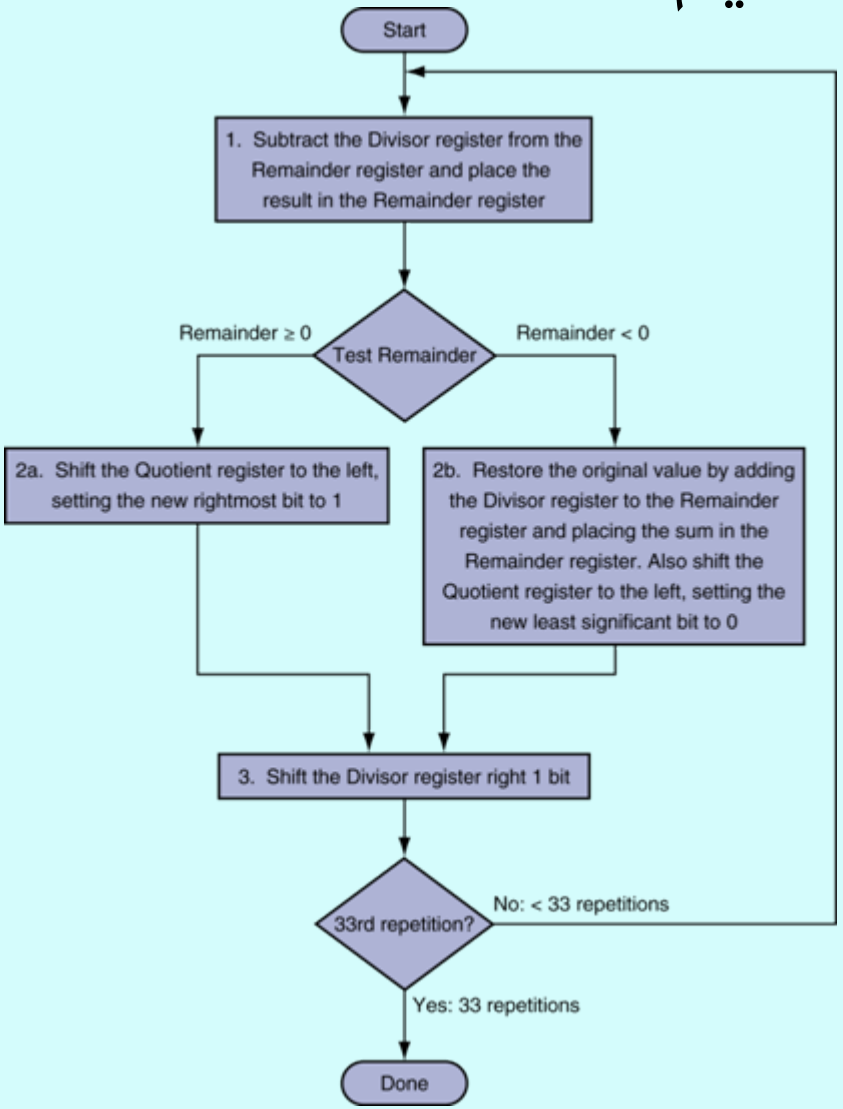
– HI: most-significant 32 bits
– LO: least-significant 32-bits

– دستورات

- `mult rs, rt /  multu rs, rt`
- انجام عملیات ضرب
- `mfhi rd  /  mflo rd`
- انتقال محتوای HI/LO به ثبات‌های چند منظوره
- `mul rd, rs, rt`
- انتقال قسمت کم‌ارزش به rd

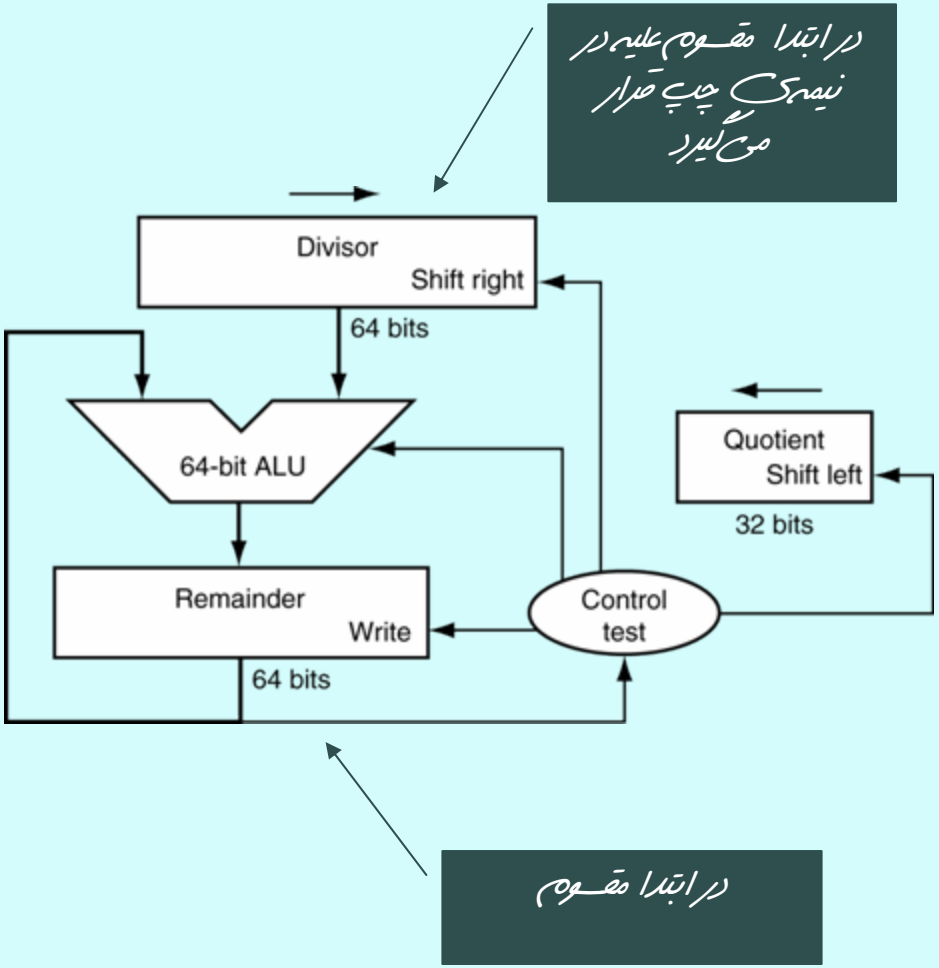
تقسیم



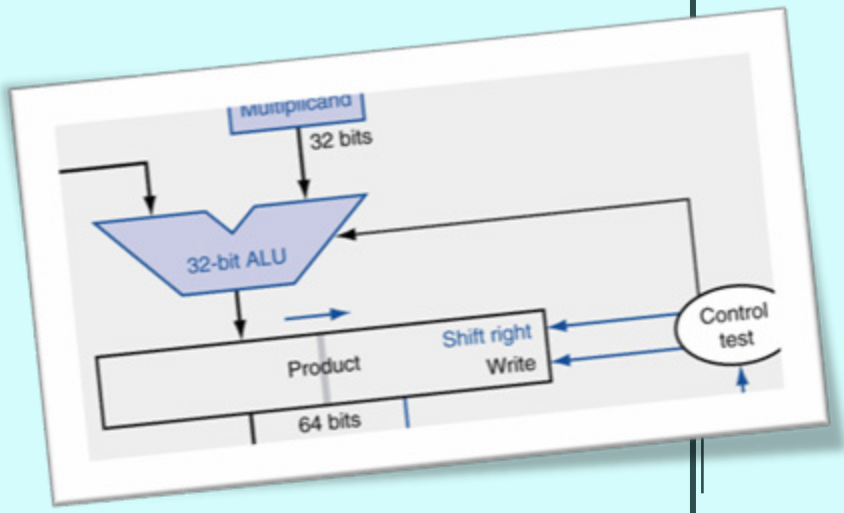


Dividend=Quotient X Divisor + Reminder

سخت‌افزار تقسیم



سخت‌افزار بهینه‌سازی شده



شبیه به مدار ضرب نیست؟!

مدارهای تقسیم سریع

- نمی‌توان تقسیم را به صورت موازی انجام داد.
  - تفریق به صورت مشروط انجام می‌شود.
- الگوریتم‌های سریع‌تر مانند SRT در هر مرحله چندین بیت خارج قسمت تولید می‌کنند.
  - باز هم الگوریتم در گام‌های متفاوت انجام می‌شود.

تقسیم در MIPS

- برای نتیجه‌ی تقسیم از ثبات‌های HI و و LO استفاده می‌شود.

– HI: 32-bit remainder

– LO: 32-bit quotient

– دستورات

– `div rs, rt  /  divu rs, rt`

– سرریز یا تقسیم بر صفر باید به صورت نرم‌افزاری چک شود.

– برای دسترسی به نتایج می‌توان از دستورات زیر استفاده کرد.

– `mfhi, mflo`

- برای نمایش اعداد اعشاری و اعداد بسیار بزرگ از سیستم عددی میز شناور استفاده می شود.
  - ۳,۱۴۱۵۹۲۶۵
  - ۲,۷۱۸۲۸
  - ۰۰۰۰۰۰۰۰۱ = ۰,۱× ۱۰$^{-۹}$

	IBM/370	DEC/VAX	Cyber 70
Word length (double)	32 (64) bits	32 (64) bits	60 bits
Significand+{hidden bit}	24 (56) bits	$23+1$ $(55+1)$ bits	48 bits
Exponent	7 bits	8 bits	11 bits
Bias	64	128	1024
Base	16	2	2
Range of M	$\frac{1}{16} \le M < 1$	$\frac{1}{2} \le M < 1$	$1 \le M < 2$
Representation of M	Signed-magnitude	Signed-magnitude	One's complement
Approximate range	$16^{63} \approx 7 \cdot 10^{75}$	$2^{127} \approx 1.9 \cdot 10^{38}$	$2^{1023} \approx 10^{307}$
Approximate resolution	$2^{-24} \approx 10^{-7}$ (10^{-17})	$2^{-24} \approx 10^{-7}$ (10^{-17})	$2^{-48} \approx 10^{-14}$

ممیز شناور (ادامه...)

- در سال ۱۹۸۵ استاندارد IEEE Std 754 مطرح شد.
- این استاندارد واگرایی شیوه‌های به کار رفته برای نمایش ممیز شناور را کاهش داد.
  - بدین‌ترتیب برنامه‌های نوشته شده برای مقاصد علمی قابل حمل شدند.
- بر طبق این استاندارد، اعداد به دو شیوه نشان داده می‌شود:

- single
- double

single: 8 bits
double: 11 bits

single: 23 bits
double: 52 bits

$$x = (-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

Single: Bias = 127; Double: Bias = 1023

ممیز شناور (ادامه...)

- در سال ۱۹۸۵ استاندارد IEEE Std 754 مطرح شد.
- این استاندارد واگرایی شیوه‌های به کار رفته برای نمایش ممیز شناور را کاهش داد.
  - بدین‌ترتیب برنامه‌های نوشته شده برای مقاصد علمی قابل حمل شدند.
- بر طبق این استاندارد، اعداد به دو شیوه نشان داده می‌شود:

- single
- double

single: 8 bits
double: 11 bits

single: 23 bits
double: 52 bits

$$x = (-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

Single: Bias = 127; Double: Bias = 1023

Denormal Numbers

اعداد نانرمال

Exponent = 000...0 $\Rightarrow$ hidden bit is 0

$$x = (-1)^S \times (0 + \text{Fraction}) \times 2^{-\text{Bias}}$$

- بدین ترتیب می‌توان اعداد کوچک‌تری را نیز نمایش داد.
- در صورتی که بخش کسری را برابر صفر قرار دهیم:

$$x = (-1)^S \times (0 + 0) \times 2^{-\text{Bias}} = \pm 0.0$$

بدین ترتیب دو نمایش برای 0 خواهیم داشت

Infinities and NaNs

ناعدد و بی‌نهایت

- Exponent = 111...1, Fraction = 000...0
  - $\pm\infty$
    - در محاسبات بعدی نیز قابل استفاده است.
- Exponent = 111...1, Fraction ≠ 000...0
    - ناعدد (Not-a-Number (NaN))
    - بیان‌گر محاسبات نادرست می‌باشد.
    - این اعداد نیز قابلیت استفاده در محاسبات بعدی را دارند.

Single precision		Double precision		Object represented
Exponent	Fraction	Exponent	Fraction	
0	0	0	0	0
0	Nonzero	0	Nonzero	± denormalized number
1–254	Anything	1–2046	Anything	± floating-point number
255	0	2047	0	± infinity
255	Nonzero	2047	Nonzero	NaN (Not a Number)

معماری کامپیوتر (۱۳۹۱-۱۱-۱۳)
جلسه‌ی یازدهم

دانشگاه شهید بهشتی
دانشکده‌ی مهندسی برق و کامپیوتر
بهار ۱۳۹۱
احمد محمودی ازناوه

فهرست مطالب

- مروری بر جلسه‌ی پیش
- ممیز شناور

ممیز شناور

- برای نمایش اعداد اعشاری و اعداد بسیار بزرگ از سیستم عددی میز شناور استفاده می شود.
  - ۳,۱۴۱۵۹۲۶۵
  - ۲,۷۱۸۲۸
  - ۰۰۰۰۰۰۰۰۱ = ۰,۱× ۱۰$^{-۹}$

	IBM/370	DEC/VAX	Cyber 70
Word length (double)	32 (64) bits	32 (64) bits	60 bits
Significand+{hidden bit}	24 (56) bits	$23+1$ $(55+1)$ bits	48 bits
Exponent	7 bits	8 bits	11 bits
Bias	64	128	1024
Base	16	2	2
Range of M	$\frac{1}{16} \le M < 1$	$\frac{1}{2} \le M < 1$	$1 \le M < 2$
Representation of M	Signed-magnitude	Signed-magnitude	One's complement
Approximate range	$16^{63} \approx 7 \cdot 10^{75}$	$2^{127} \approx 1.9 \cdot 10^{38}$	$2^{1023} \approx 10^{307}$
Approximate resolution	$2^{-24} \approx 10^{-7}$ (10^{-17})	$2^{-24} \approx 10^{-7}$ (10^{-17})	$2^{-48} \approx 10^{-14}$

Exponential Notation

- تمام اعداد زیر نمایش عدد 1234 می‌باشند.

$123{,}400.0 \times 10^{-2}$

$12{,}340.0 \times 10^{-1}$

$1{,}234.0 \times 10^{0}$

123.4×10^{1}

12.34×10^{2}

1.234×10^{3}

0.1234×10^{4}

با تغییر همزمان توان و جایگاه ممیز نمایش‌های متفاوتی برای یک عدد به دست می‌آید.

ممیز شناور (ادامه...)

- در سال ۱۹۸۵ استاندارد IEEE Std 754 مطرح شد.
- این استاندارد واگرایی شیوه‌های به کار رفته برای نمایش ممیز شناور را کاهش داد.
  - بدین‌ترتیب برنامه‌های نوشته شده برای مقاصد علمی قابل حمل شدند.
- بر طبق این استاندارد، اعداد به دو شیوه نشان داده می‌شود:

- single
- double

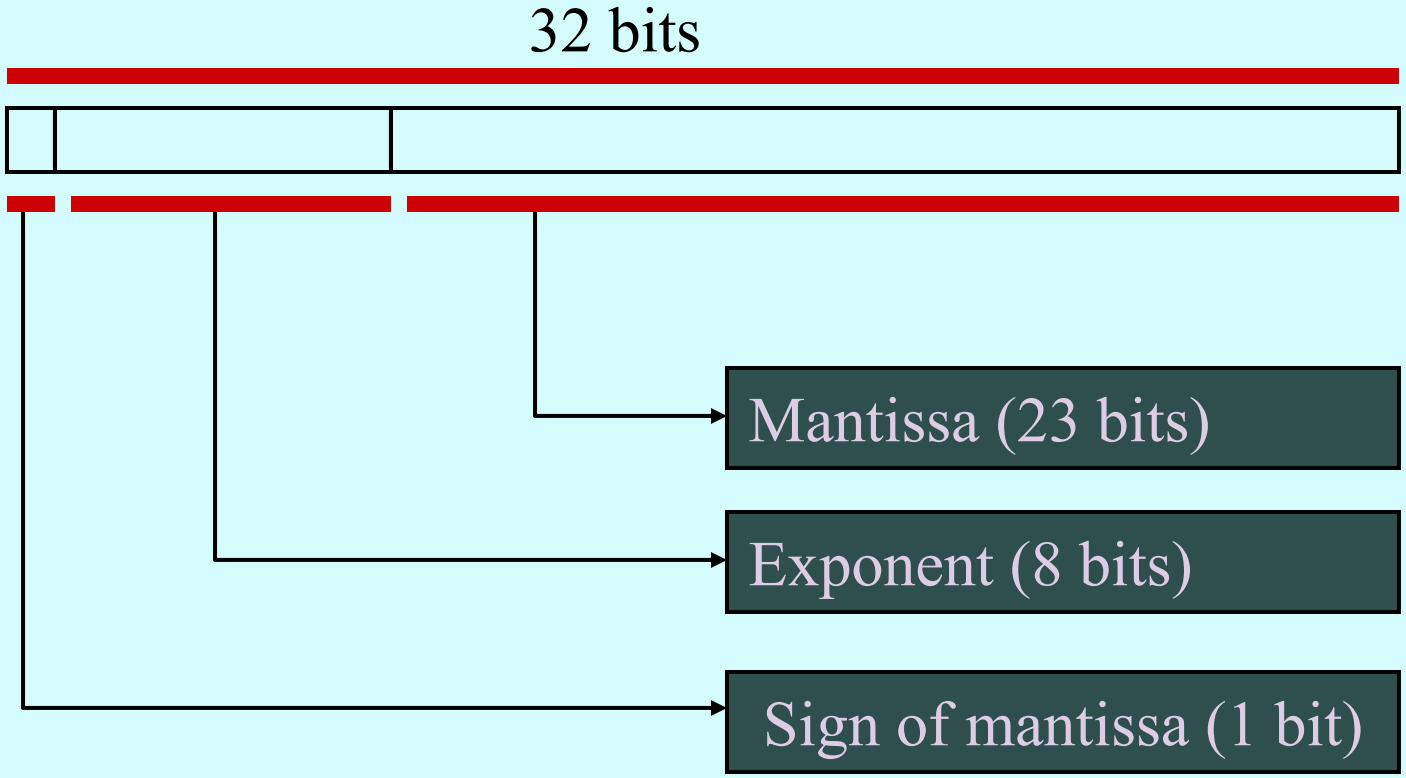
single: 8 bits
double: 11 bits

single: 23 bits
double: 52 bits

$$x = (-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

Single: Bias = 127; Double: Bias = 1023

Single Precision Format



$$Value = (-1)^S \; 1.F \times 2^{E-127}$$

Denormal Numbers

اعداد نانرمال

Exponent = 000...0 $\Rightarrow$ hidden bit is 0

$$x = (-1)^S \times (0 + \text{Fraction}) \times 2^{-\text{Bias}}$$

- بدین ترتیب می‌توان اعداد کوچک‌تری را نیز نمایش داد.
- در صورتی که بخش کسری را برابر صفر قرار دهیم:

$$x = (-1)^S \times (0 + 0) \times 2^{-\text{Bias}} = \pm 0.0$$

بدین ترتیب دو نمایش برای 0 خواهیم داشت

Infinities and NaNs

ناعدد و بی‌نهایت

- Exponent = 111...1, Fraction = 000...0
  - $\pm\infty$
    - در محاسبات بعدی نیز قابل استفاده است.
- Exponent = 111...1, Fraction ≠ 000...0
    - ناعدد (Not-a-Number (NaN))
    - بیان‌گر محاسبات نادرست می‌باشد.
    - این اعداد نیز قابلیت استفاده در محاسبات بعدی را دارند.

Single precision		Double precision		Object represented
Exponent	Fraction	Exponent	Fraction	
0	0	0	0	0
0	Nonzero	0	Nonzero	± denormalized number
1–254	Anything	1–2046	Anything	± floating-point number
255	0	2047	0	± infinity
255	Nonzero	2047	Nonzero	NaN (Not a Number)

محاسبات مقادیر خاص

- $(+0) + (+0) = (+0) - (-0) = +0$
- $(+0) \times (+5) = +0$
- $(+0) / (-5) = -0$
- $(+\infty) + (+\infty) = +\infty$
- $x - (+\infty) = -\infty$
- $(+\infty) \times x = \pm\infty$, depending on the sign of x
- $x / (+\infty) = \pm 0$, depending on the sign of x
- $\sqrt{(+\infty)} = +\infty$

نمایش در مبنای شانزده

- **نمایش یک عدد ممیز شناور در مبنای شانزده معمول است:**

0	10000011	01000000000000000000000
4	1	A	0	0	0	0	0

-15.6875

$C17B0000_{16} =$

$1\ 10000010\ 11110110000000000000000_2$

S E M

1 = negative
0 = positive

- محاسباتی که منجر به تولید ناعدد می‌شود:
  - $(\pm 0) / (\pm 0) = NaN$
  - $(+\infty) + (-\infty) = NaN$
  - $(\pm 0) \times (\pm \infty) = NaN$
  - $(\pm \infty) / (\pm \infty) = NaN$
- ناعدد در محاسبات و مقایسه‌ها
  - $NaN + x = NaN$
  - $NaN + NaN = NaN$
  - $NaN \times 0 = NaN$
  - $NaN \times NaN = NaN$

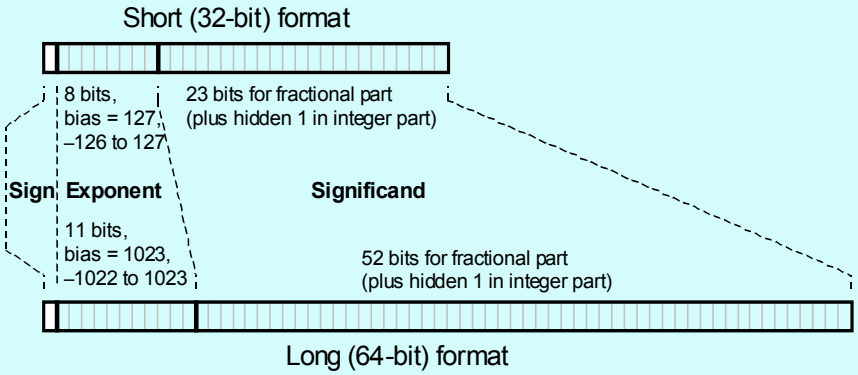
$NaN < 2 \rightarrow$ false

$NaN = Nan \rightarrow$ false

$NaN \neq (+\infty) \rightarrow$ true

$NaN \neq NaN \rightarrow$ true

پراکندگی داده‌ها در ممیز شناور

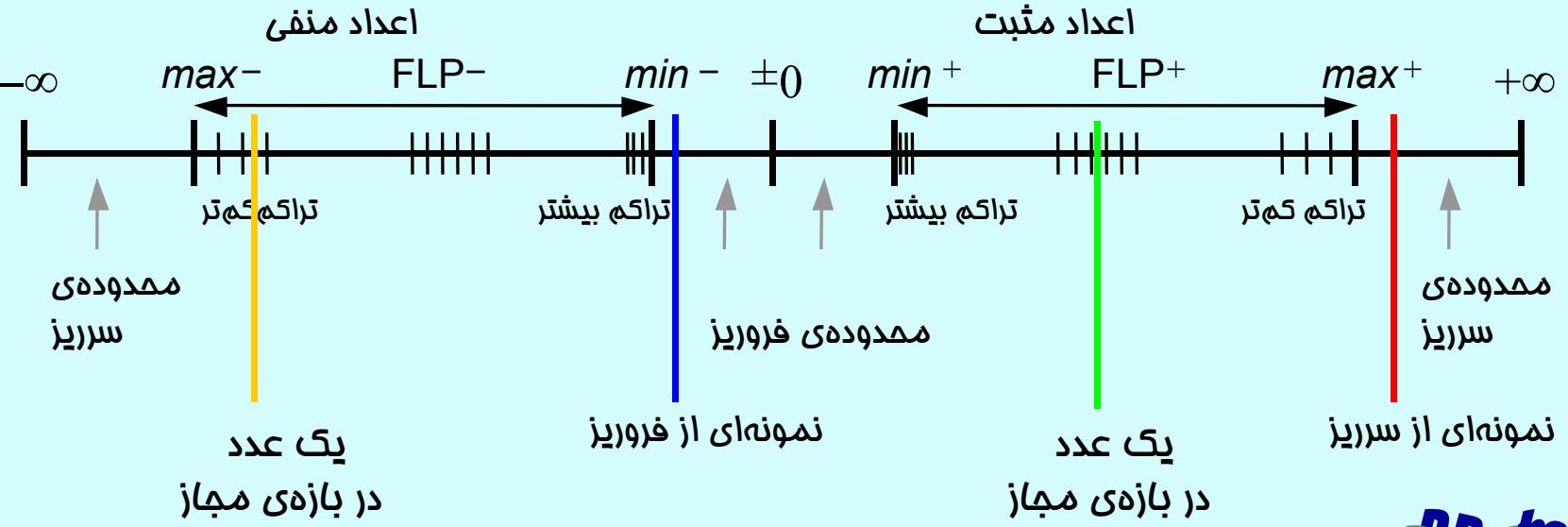


$\pm 0, \pm \infty, \text{NaN}$

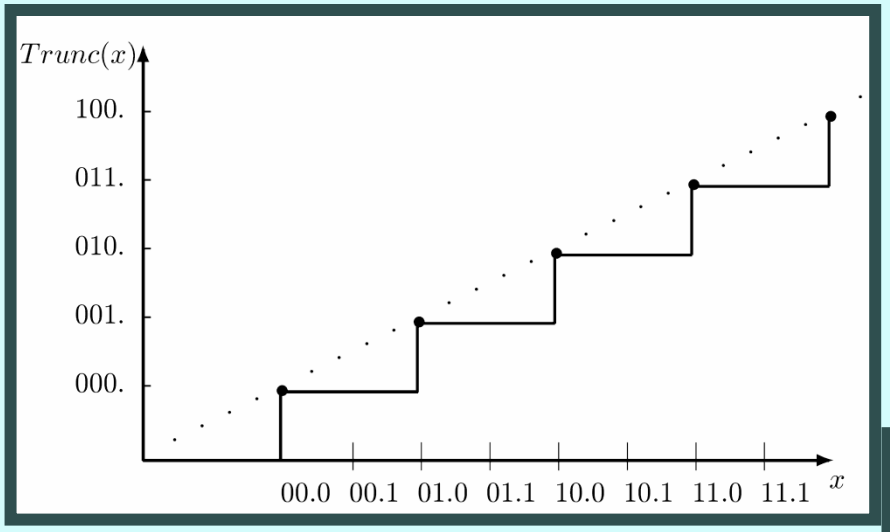
$1.f \times 2^e$

Denormals:

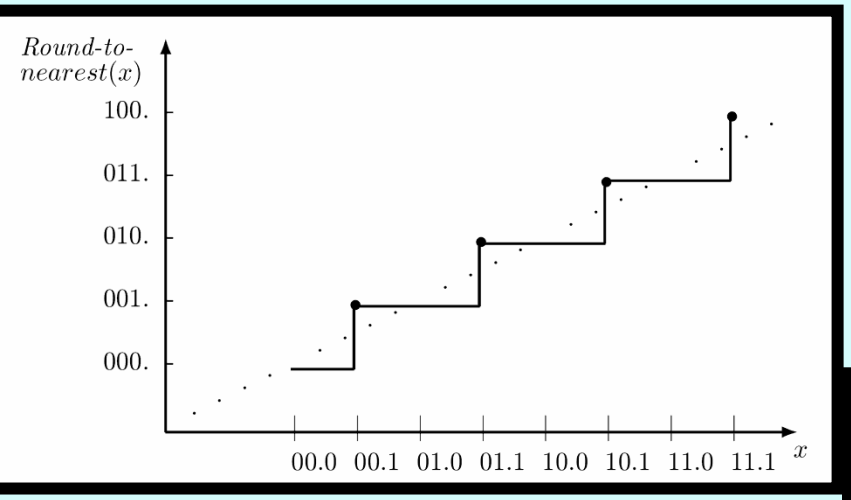
$0.f \times 2^{e_{min}}$



گرد کردن

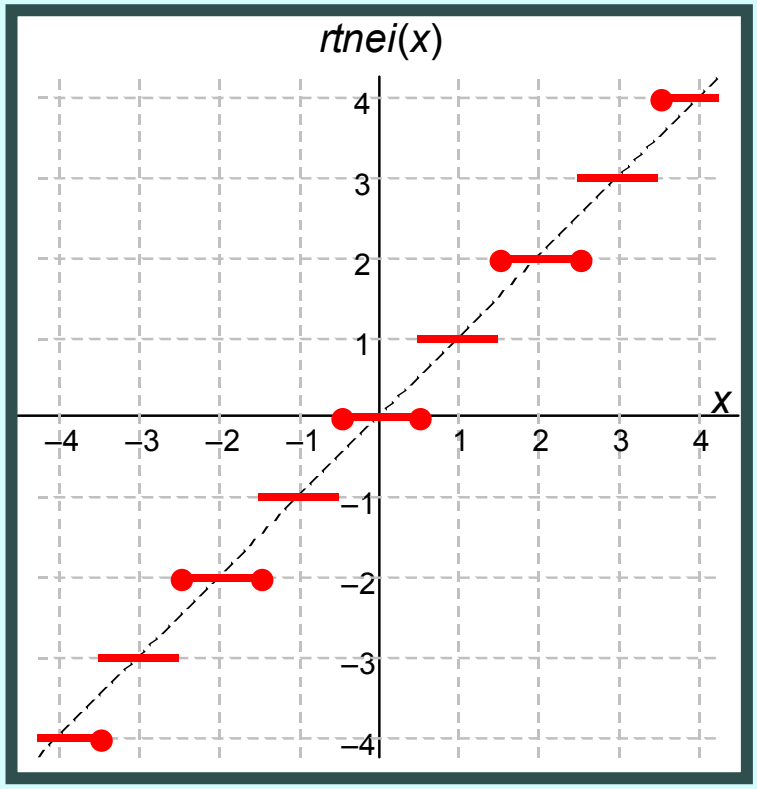


Number	$Trunc(x)$	Error
$X.00$	X	0
$X.01$	X	$-1/4$
$X.10$	X	$-1/2$
$X.11$	X	$-3/4$



Number	*Round-to-nearest(x)*	Error
$X.00$	X	0
$X.01$	X	$-1/4$
$X.10$	$X + 1$	$+1/2$
$X.11$	$X + 1$	$+1/4$

گرد کردن به نزدیک‌ترین مقدار زوج



Number	$Round(x)$	Error	Number	$Round(x)$	Error
$X0.00$	$X0.$	0	$X1.00$	$X1.$	0
$X0.01$	$X0.$	$-1/4$	$X1.01$	$X1.$	$-1/4$
$X0.10$	$X0.$	$-1/2$	$X1.10$	$X1.+1$	$+1/2$
$X0.11$	$X1.$	$+1/4$	$X1.11$	$X1.+1$	$+1/4$

شیوه‌های گرد کردن در استاندارد IEEE754

LSB	R	S	Operation	$\overline{Error}$
0	0	0	+ 0	0
0	0	1	+ 0	$-0.25\ ulp$
0	1	0	+ 0	$-0.50\ ulp$
0	1	1	$+0.5\ ulp$	$+0.25\ ulp$
1	0	0	+ 0	0
1	0	1	+ 0	$-0.25\ ulp$
1	1	0	$+0.5\ ulp$	$+0.50\ ulp$
1	1	1	$+0.5\ ulp$	$+0.25\ ulp$
			Total	0

(a) Round-to-nearest-even scheme

R	S	Operation	$\overline{Error}$
0	0	+ 0	0
0	1	+ 0	$-0.25\ ulp$
1	0	+ 0	$-0.50\ ulp$
1	1	+ 0	$-0.75\ ulp$
		Total	$-0.375\ ulp$

(b) Round-to-zero scheme

Sign	R	S	Operation
+	0	0	+ 0
+	0	1	$+1\ ulp$
+	1	0	$+1\ ulp$
+	1	1	$+1\ ulp$
−	0	0	+ 0
−	0	1	+ 0
−	1	0	+ 0
−	1	1	+ 0

(c) Round-to-plus-infinity scheme

Sign	R	S	Operation
−	0	0	+ 0
−	0	1	$+1\ ulp$
−	1	0	$+1\ ulp$
−	1	1	$+1\ ulp$
+	0	0	+ 0
+	0	1	+ 0
+	1	0	+ 0
+	1	1	+ 0

(d) Round-to-minus-infinity scheme

محدوده‌ی قابل نمایش

- **کوچک‌ترین مقدار ممکن**
  - Exponent: 00000001
    $\Rightarrow$ actual exponent = 1 – 127 = –126
  - Fraction: 000…00 $\Rightarrow$ significand = 1.0
  - $\pm 1.0 \times 2^{-126} \approx \pm 1.2 \times 10^{-38}$
- **بزرگ‌ترین مقدار ممکن**
  - exponent: 11111110
    $\Rightarrow$ actual exponent = 254 – 127 = +127
  - Fraction: 111…11 $\Rightarrow$ significand $\approx$ 2.0
  - $\pm 2.0 \times 2^{+127} \approx \pm 3.4 \times 10^{+38}$

محدوده‌ی قابل نمایش با دقت مضاعف

- کوچک‌ترین مقدار ممکن
  - Exponent: 00000000001
    $\Rightarrow$ actual exponent = 1 – 1023 = –1022
  - Fraction: 000…00 $\Rightarrow$ significand = 1.0
  - $\pm 1.0 \times 2^{-1022} \approx \pm 2.2 \times 10^{-308}$
- بزرگ‌ترین مقدار ممکن
  - Exponent: 11111111110
    $\Rightarrow$ actual exponent = 2046 – 1023 = +1023
  - Fraction: 111…11 $\Rightarrow$ significand $\approx$ 2.0
  - $\pm 2.0 \times 2^{+1023} \approx \pm 1.8 \times 10^{+308}$

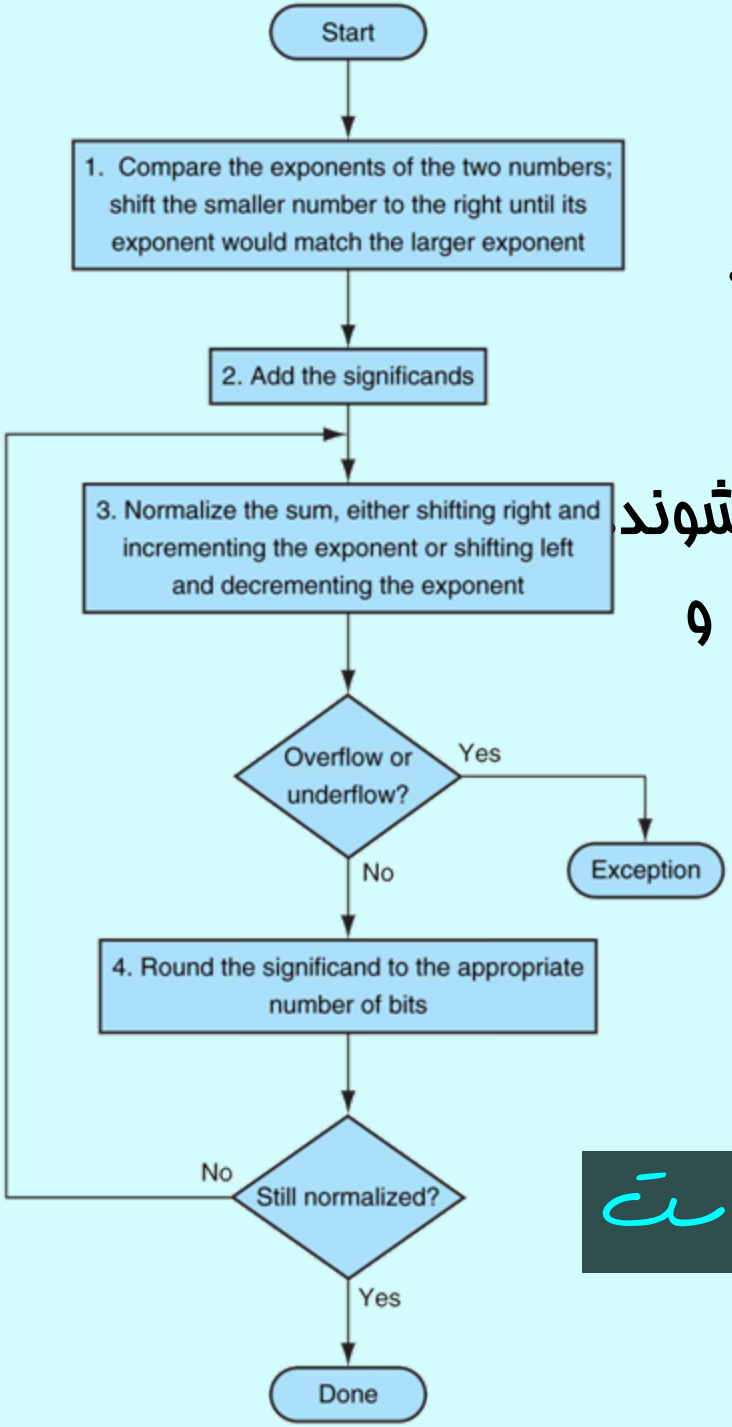
دقت در ممیز شناور

- Single: approx 2^{-23}
  - Equivalent to $23 \times \log_{10}2 \approx 23 \times 0.3 \approx 6$
  - برابر با شش رقم اعشار دقت
- Double: approx 2^{-52}
  - Equivalent to $52 \times \log_{10}2 \approx 52 \times 0.3 \approx 16$
  - برابر با شانزده رقم اعشار دقت

مشخصات اعداد ممیز شناور IEEE 754

Feature	Single/Short	Double/Long
Word width in bits	32	64
Significand in bits	23 + 1 hidden	52 + 1 hidden
Significand range	$[1, 2 - 2^{-23}]$	$[1, 2 - 2^{-52}]$
Exponent bits	8	11
Exponent bias	127	1023
Zero (± 0)	e + bias = 0, $f = 0$	e + bias = 0, $f = 0$
Denormal	e + bias = 0, $f \neq 0$ represents $\pm 0.f \times 2^{-126}$	e + bias = 0, $f \neq 0$ represents $\pm 0.f \times 2^{-1022}$
Infinity ($\pm\infty$)	e + bias = 255, $f = 0$	e + bias = 2047, $f = 0$
Not-a-number (NaN)	e + bias = 255, $f \neq 0$	e + bias = 2047, $f \neq 0$
Ordinary number	e + bias $\in [1, 254]$ $e \in [-126, 127]$ represents $1.f \times 2^e$	e + bias $\in [1, 2046]$ $e \in [-1022, 1023]$ represents $1.f \times 2^e$
min	$2^{-126} \cong 1.2 \times 10^{-38}$	$2^{-1022} \cong 2.2 \times 10^{-308}$
max	$\cong 2^{128} \cong 3.4 \times 10^{38}$	$\cong 2^{1024} \cong 1.8 \times 10^{308}$

جمع در ممیز شناور

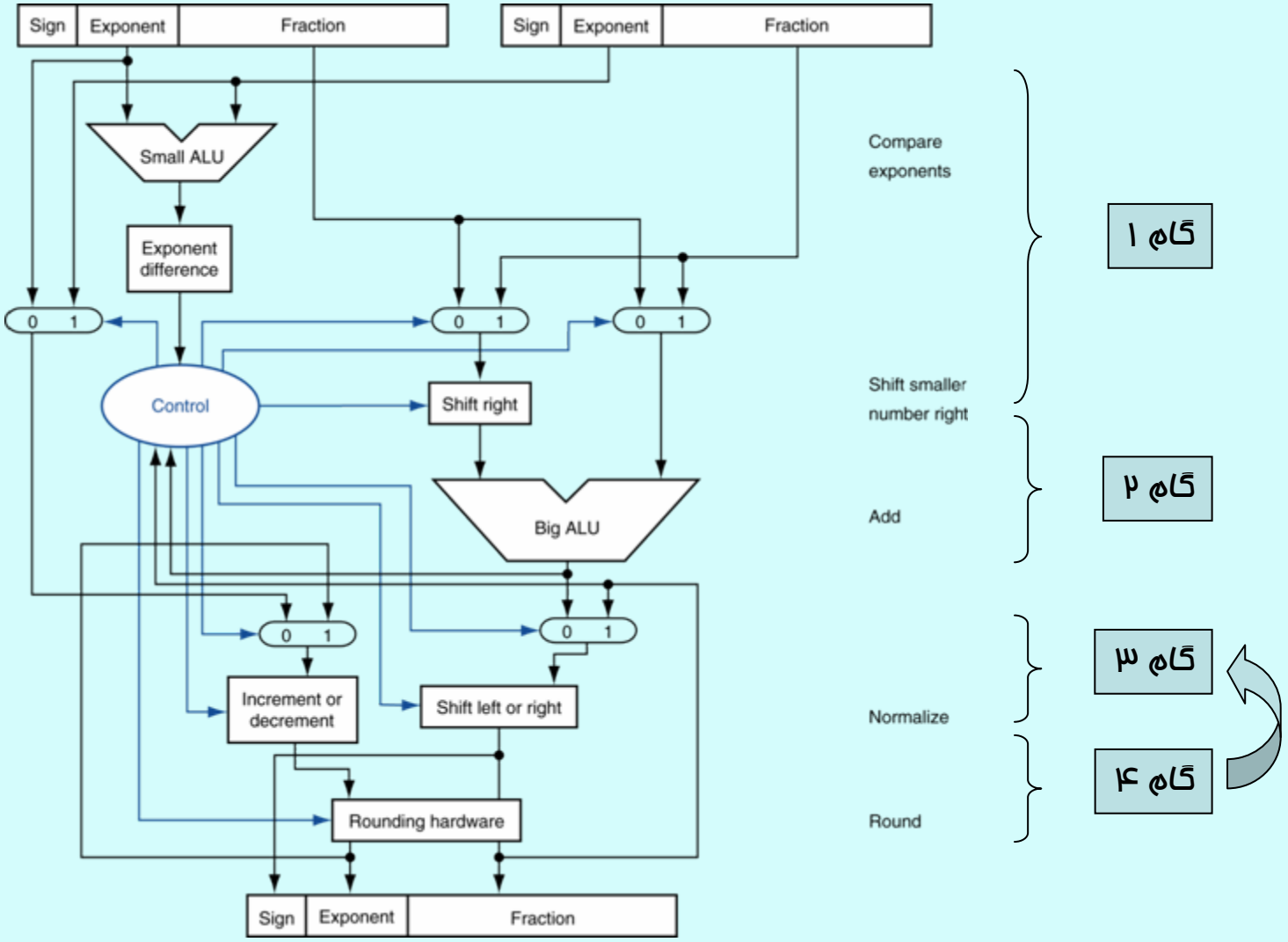


1. ابتدا توان‌ها را یکسان می‌کنیم.
   - این کار با شیفت عدد کوچک‌تر به راست انجام می‌شود.
2. مقادیر اعشاری با هم جمع می‌شوند.
3. حاصل بهنجار شده و وقوع سرریز و فروریز بررسی می‌شود.
4. حاصل گرد می‌شود.

- مجددا بهنجار بودن عدد بررسی می‌شود.

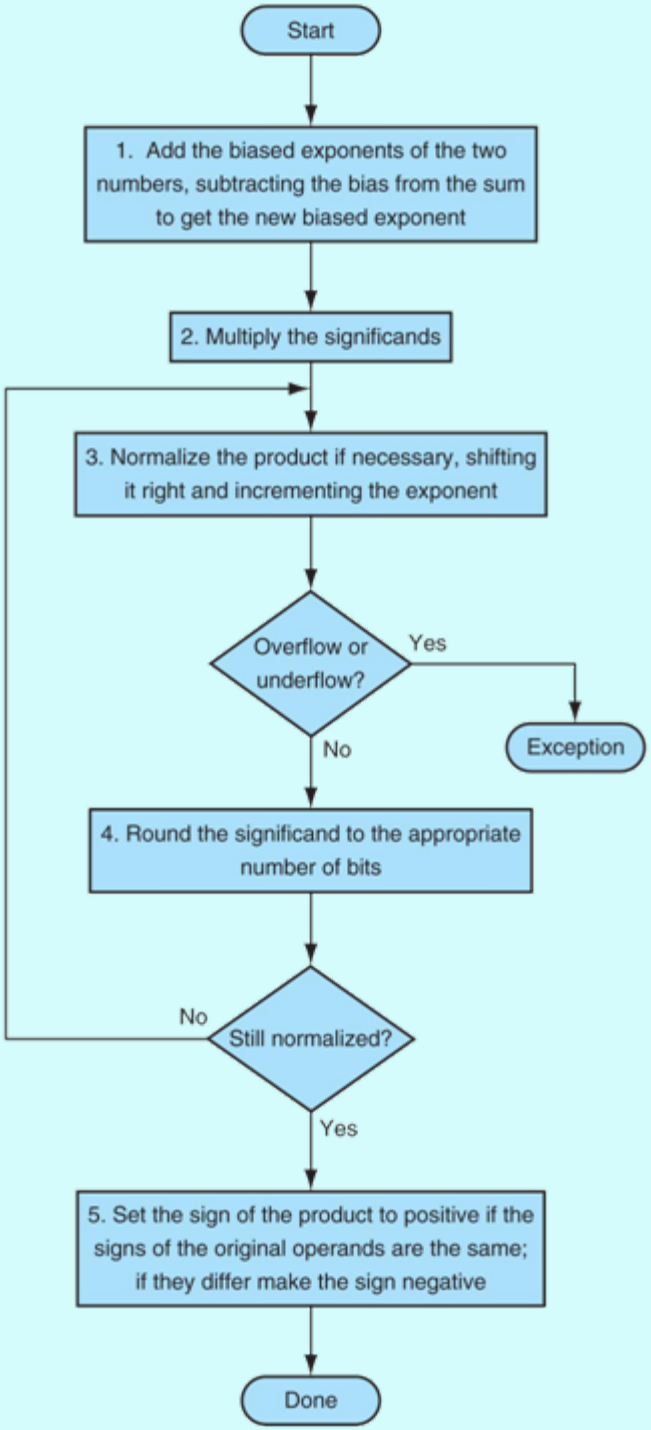
نسبت به اعداد صحیح پیچیده‌تر است

سخت‌افزار جمع ممیز شناور



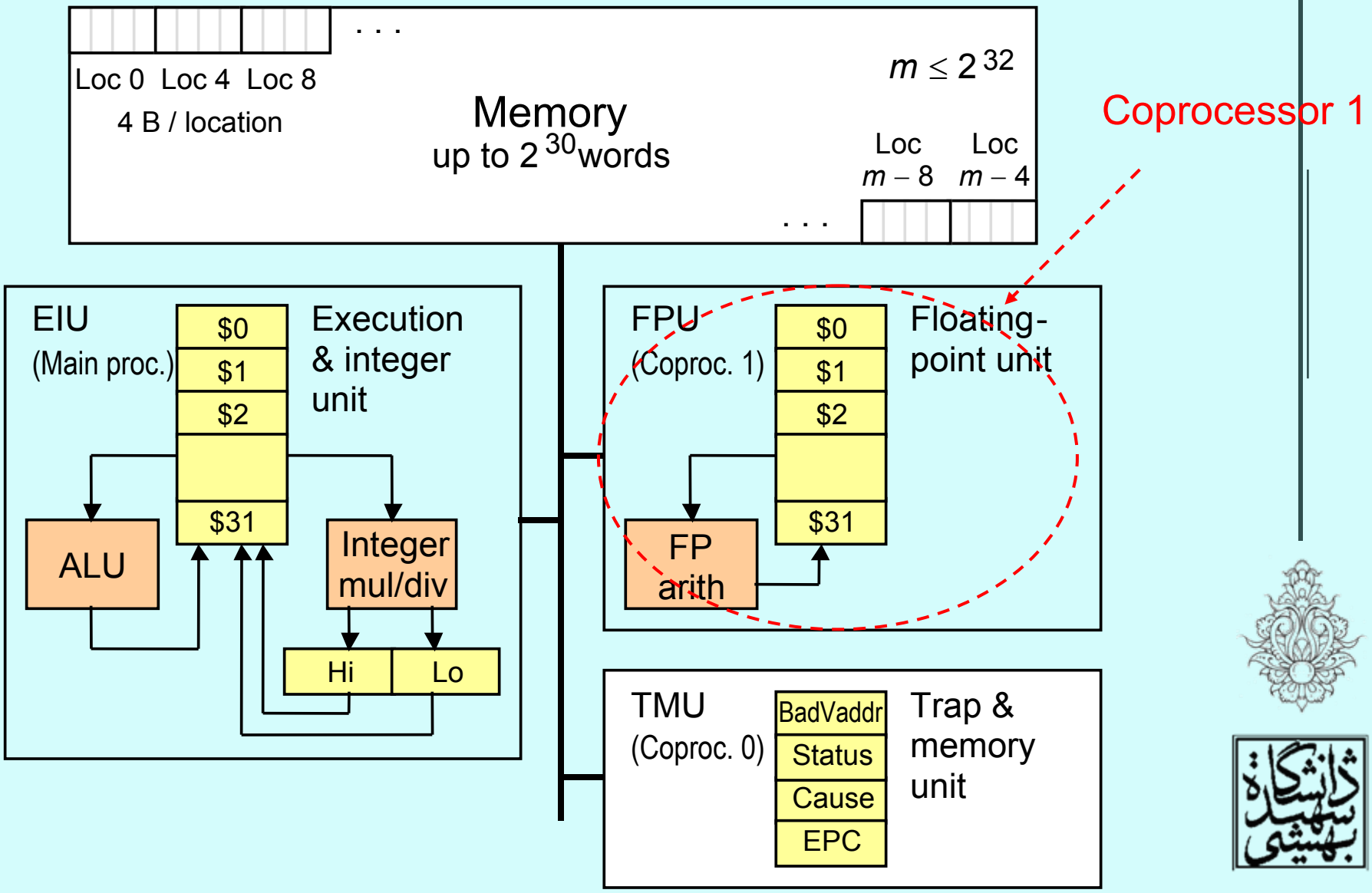
ضرب ممیز شناور

1. توان‌ها با هم جمع می‌شود
2. significandها در هم ضرب می‌شوند.
3. اعداد بهنجار شده و بروز سرریز یا فروریز چک می‌شود.
4. اعداد گرد می‌شوند و در صورت نیاز مجدد بهنجار می‌شوند.
5. علامت عدد تعیین می‌شود.



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 ۲ MIPS، 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
```

معماری کامپیوتر (۱۳۹۱-۱۱-۱۳)
جلسه‌ی دوازدهم

دانشگاه شهید بهشتی
دانشکده‌ی مهندسی برق و کامپیوتر
بهار ۱۳۹۱
احمد محمودی ازناوه

فهرست مطالب

- ما کجاییم؟
- پیش‌گفتار
- نحوه‌ی اجرای یک دستورالعمل
- مسیر گذار داده

ما کجاییم؟؟

Problems

Algorithms

Language (Program) — *Programmable*

Machine (ISA) Architecture — *Computer Specific*

Micro-architecture — *Manufacturer Specific*

Circuits

Devices

Partially from BRIGHAM YOUNG UNIVERSITY

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

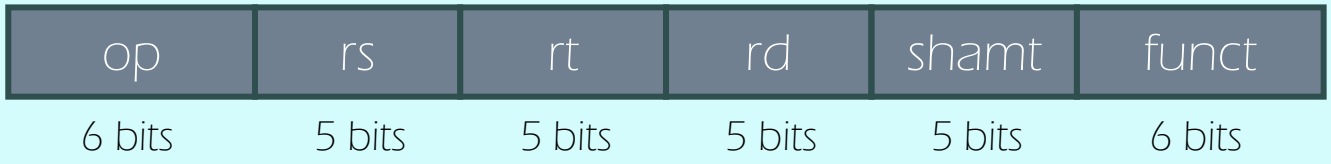
CPU execution time = Instructions × CPI / (Clock rate)

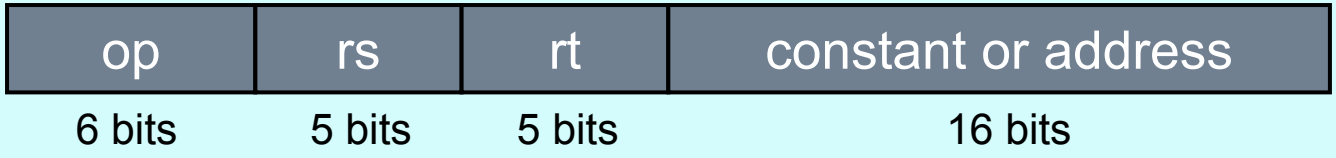
Performance = Clock rate / (Instructions × CPI)

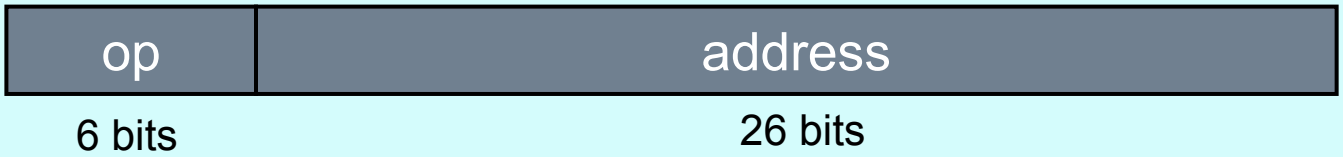
- کارایی یک برنامه توسط موارد زیر تعیین می‌شوند:
  - Instruction count
  - CPI and Cycle time
  - کامپایلر و ISA موارد تأثیرگذار بر روی مورد نخست بودند که پیش‌ار این مورد بررسی قرار گرفتند.
  - سخت‌افزار طراحی شده برای CPU تعداد سیکل به ازای هر دستور و طول سیکل را مشخص می‌کند.

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`







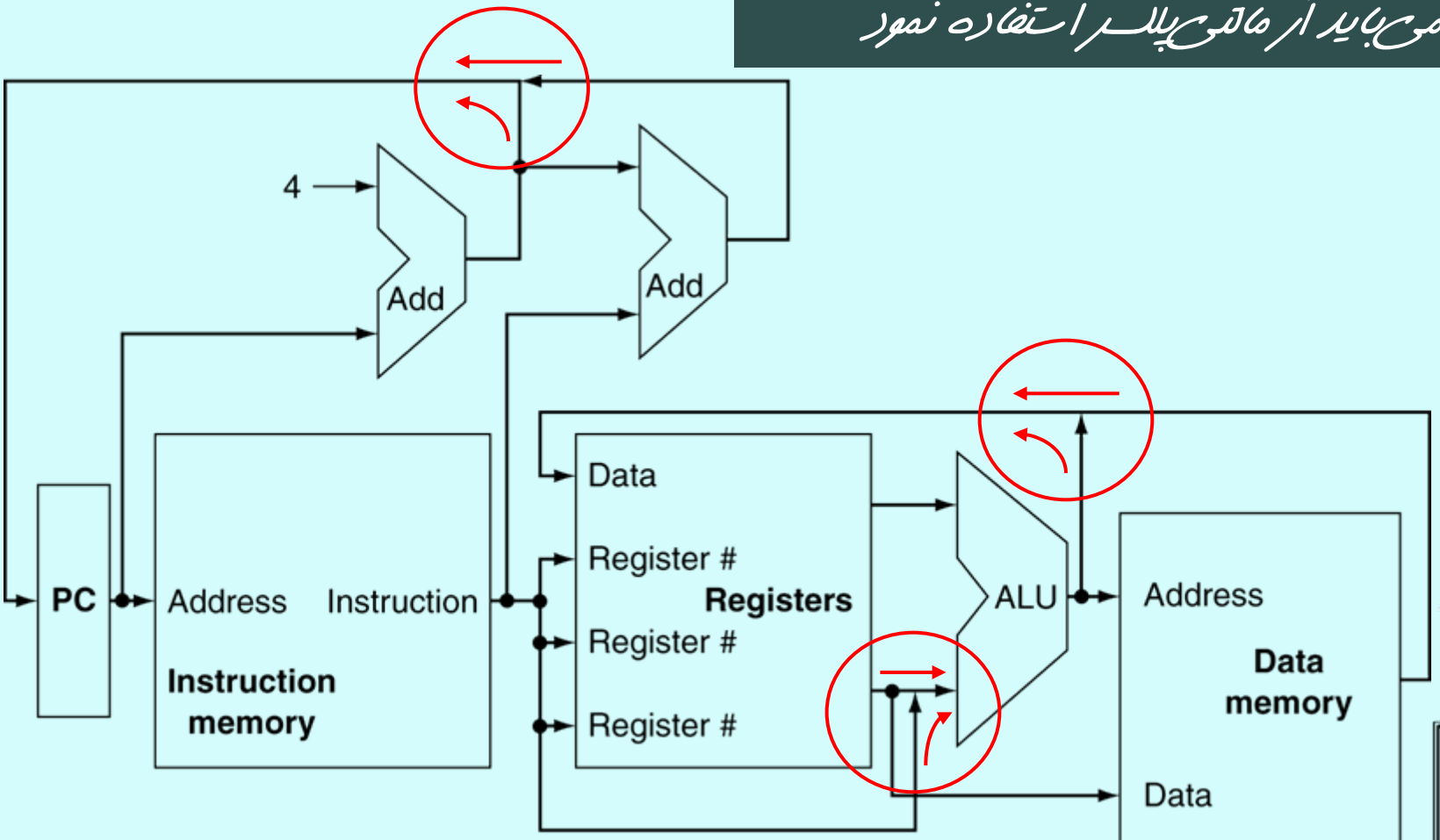
نحوه‌ی اجرای یک دستورالعمل

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

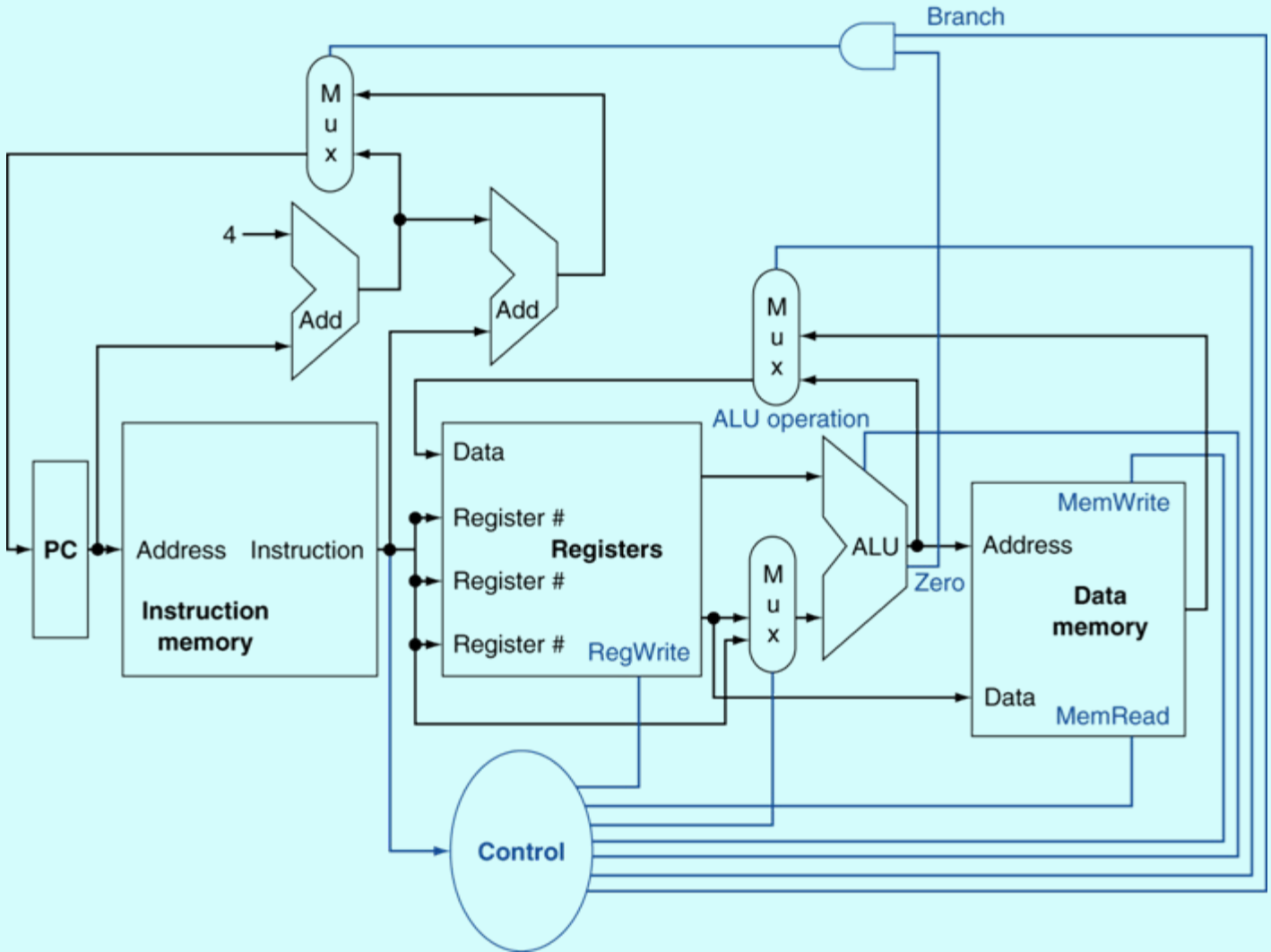
نمایی کلی از CPU

اتصال سیم‌ها به یکدیگر درست نیست
می‌باید از مالتی‌پلکسر 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

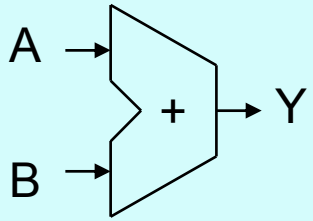
- داده‌های به صورت دودویی کد می‌شوند
- به ازای هر بیت، یک سیم استفاده می‌شود.
- برای داده‌های چند بیتی از یک دسته سیم که گذرگاه نامیده می‌شود، استفاده می‌شود.

multi-wire buses

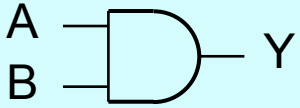
- مدارها به دو دسته تقسیم می‌شوند:
  - مدارهای ترکیبی
  - مدارهای ترتیبی

المان‌های ترکیبی

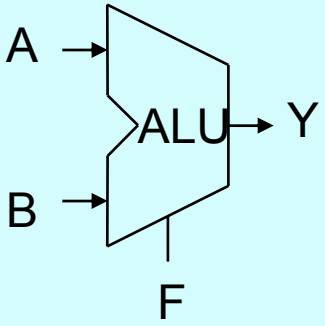
- جمع‌کننده
  - Y = A + B



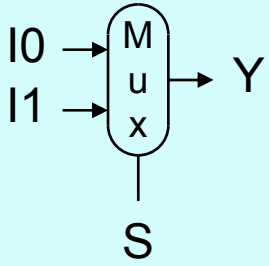
- گیت AND
  - Y = A & B



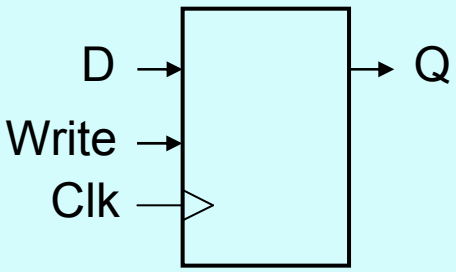
- Arithmetic/Logic Unit
  - Y = F(A, B)

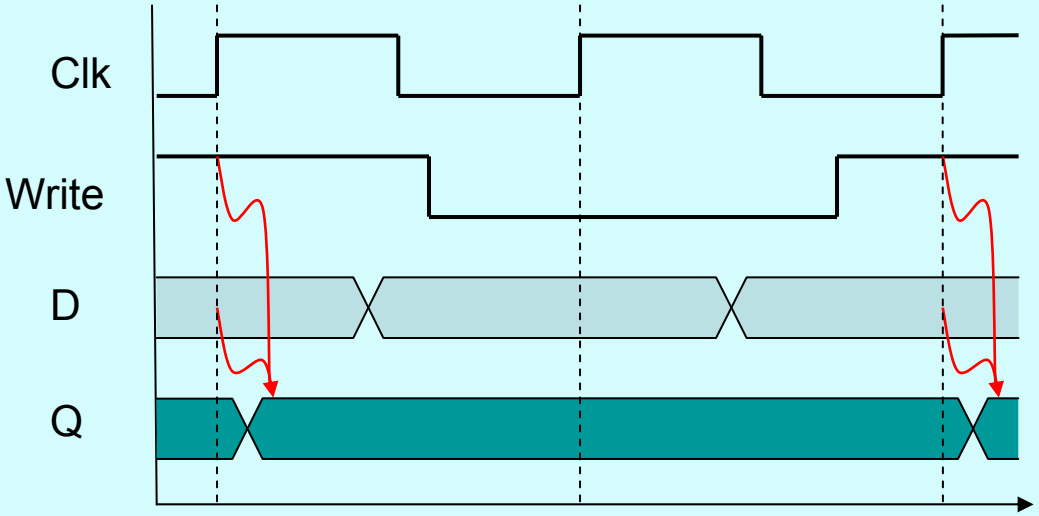


- مالتی‌پلکسر
  - Y = S ? I1 : I0

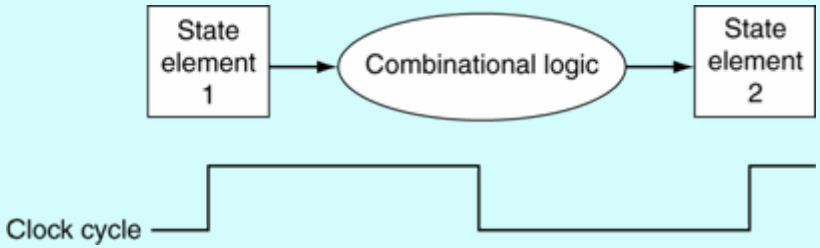


المان‌های ترتیبی





- چرا از پالس ساعت حساس به لبه استفاده می‌شود؟

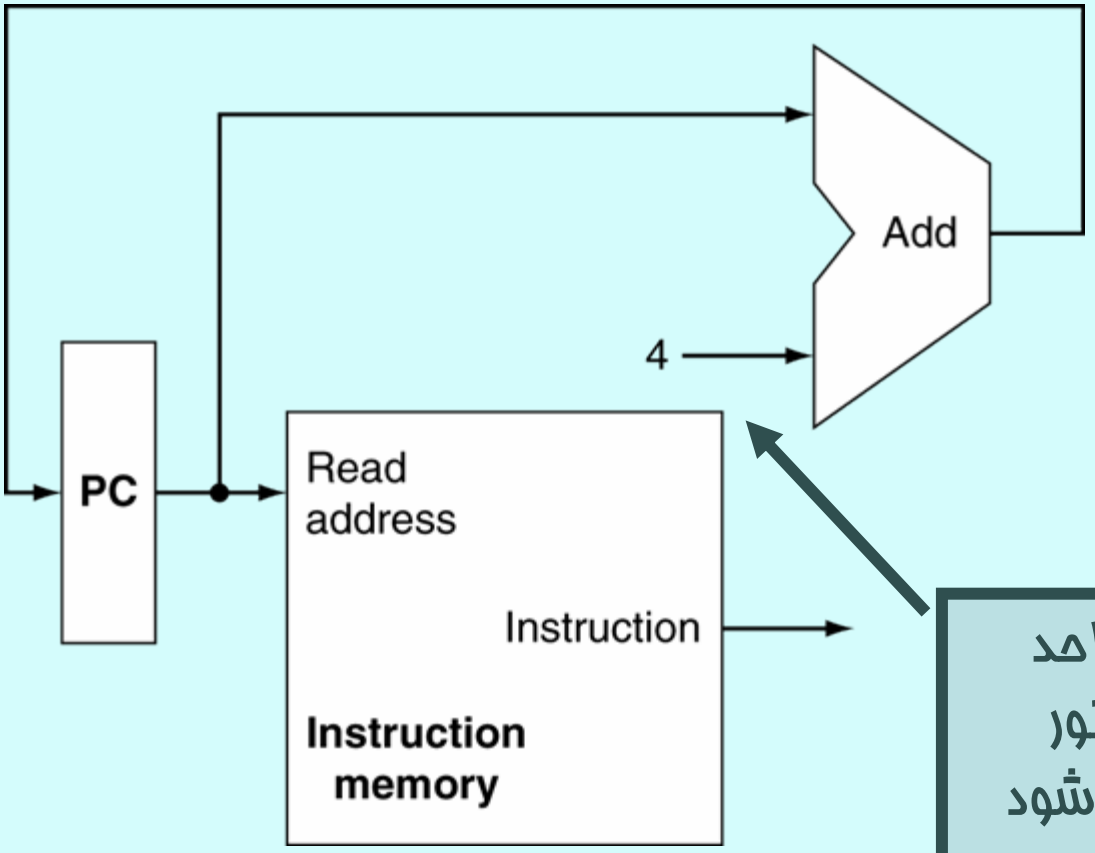


طراحی مسیر گذار داده (محاسبات)

- 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

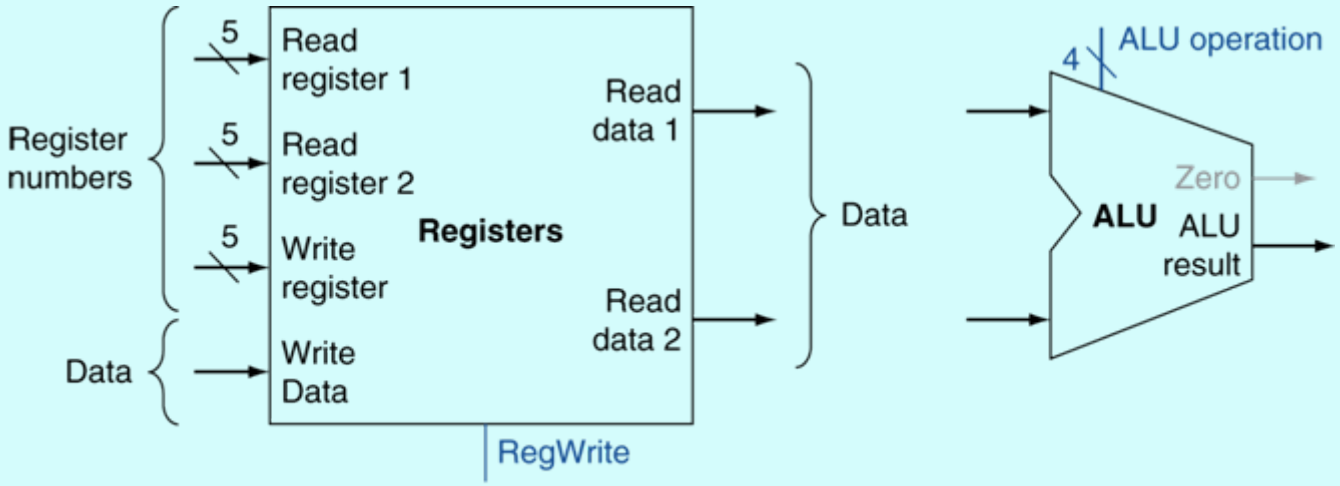


با افزودن چهار واحد
برای واکشی دستور
بعدی آماده می‌شود

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

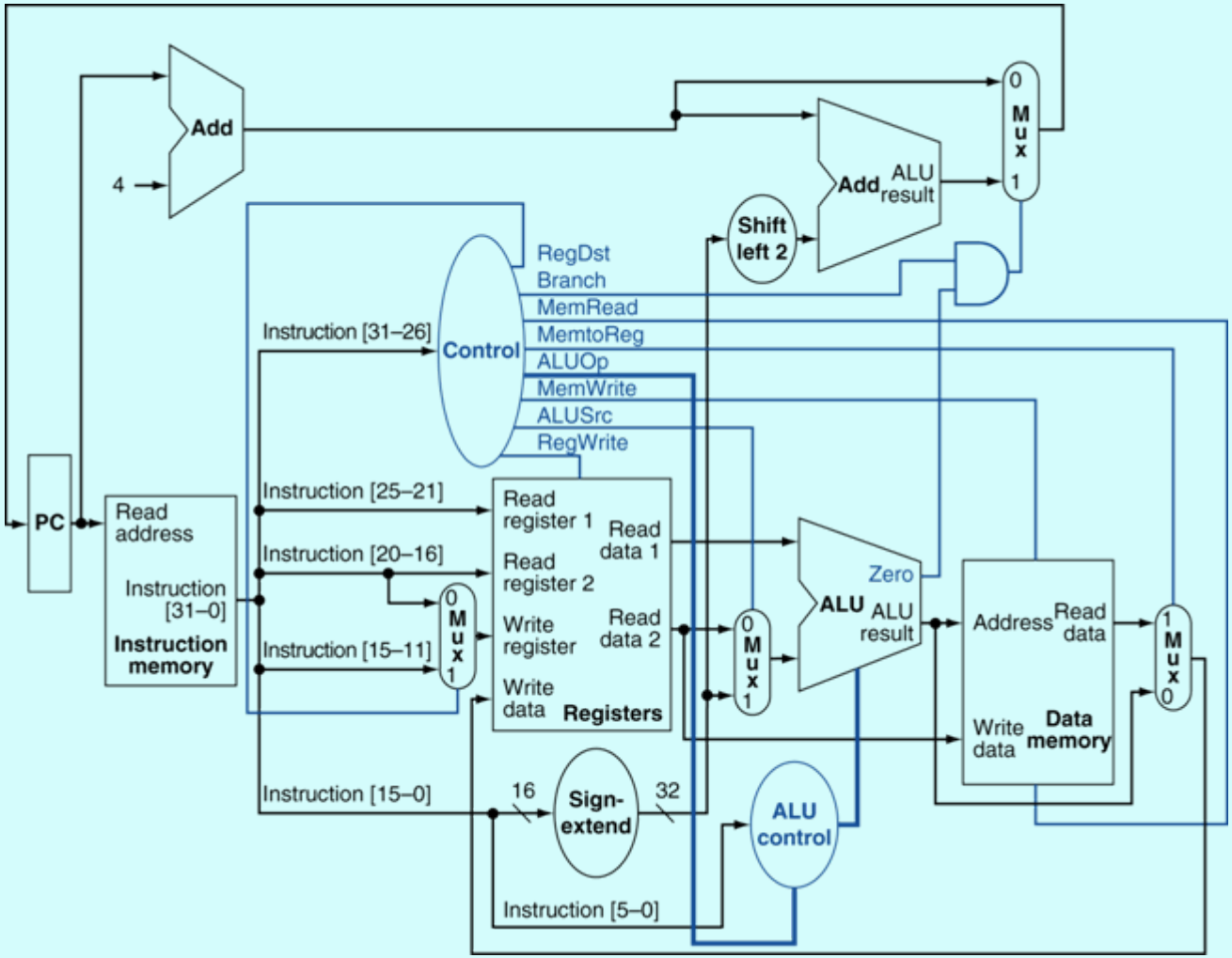
b. ALU

register file

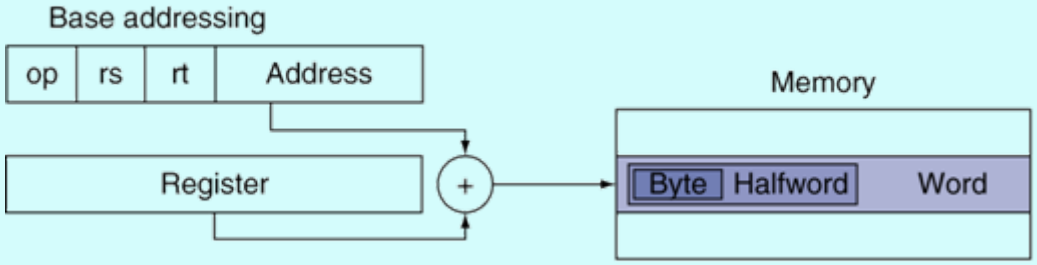
واحدی شامل همه‌ی ثبات‌ها که با انتخاب شماره‌ی ثبات، می‌توان در ثباتی خاص داده‌ی مورد نظر را خواند و یا نوشت

داده‌گذر همراه با واحد کنترل

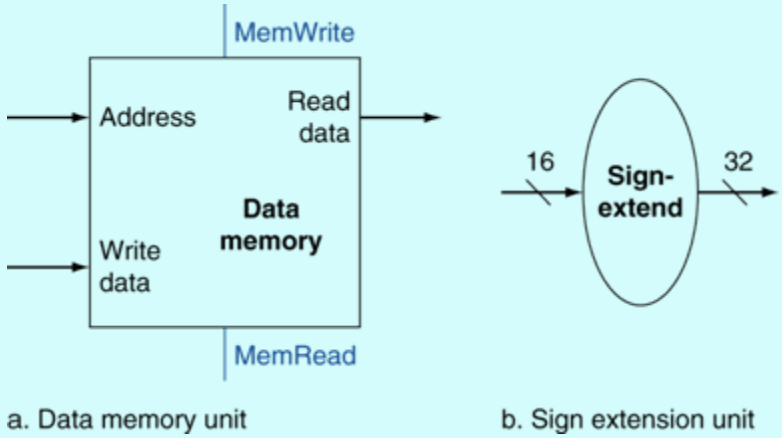
- با توجه به قالب دستور، می‌توان برچسب برخی سیگنال‌های داده و کنترلی را مشخص نمود:



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

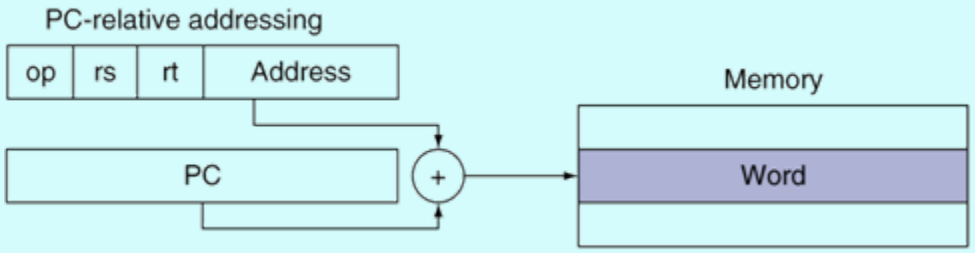


a. Data memory unit

b. Sign extension unit

دستورات پرش شرطی

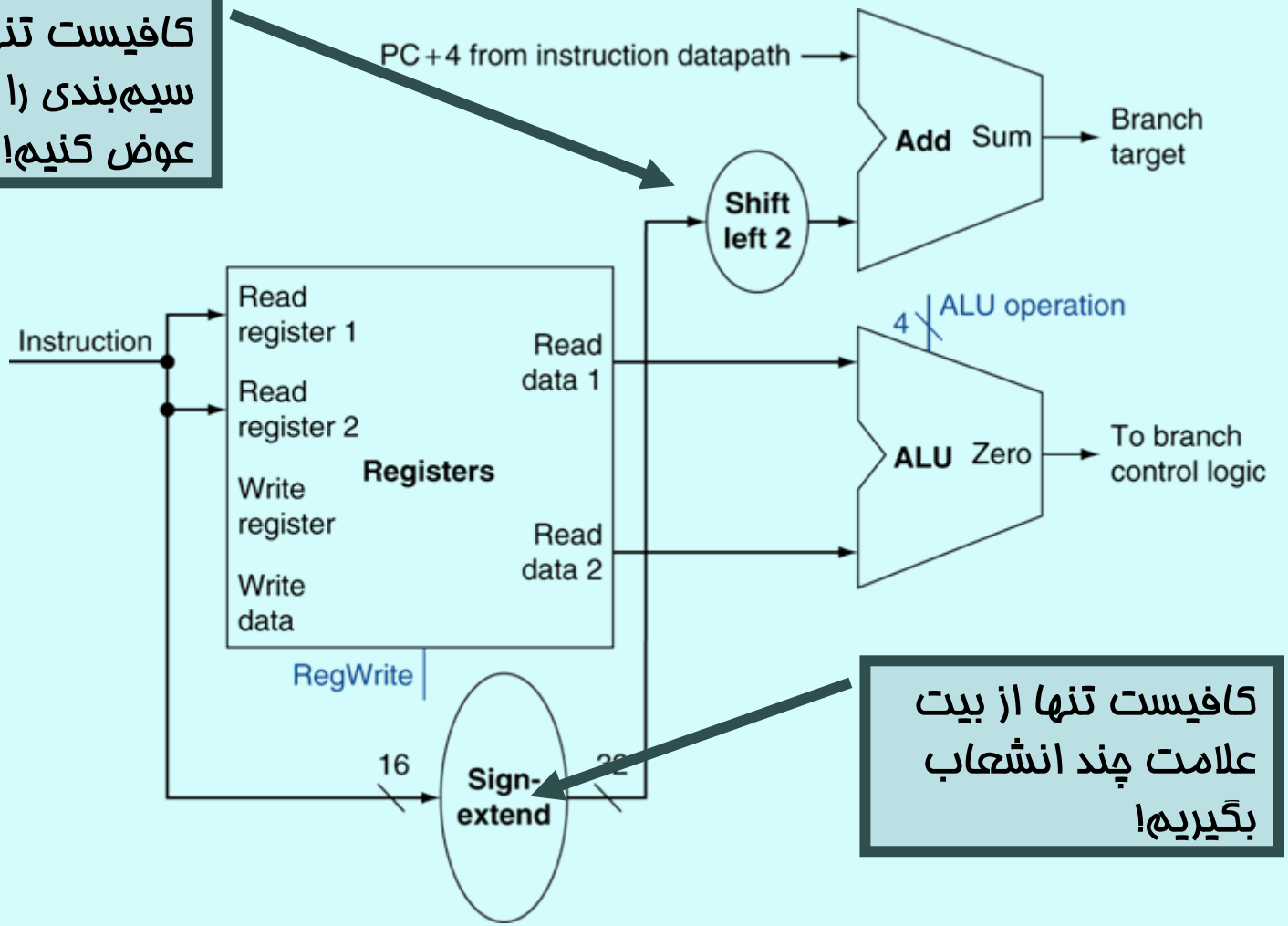
- محتوای ثبات‌ها را می‌خواند
- مقایسه می‌کند
  - با استفاده از ALU و خروجی صفر
- آدرس مقصد را به دست می‌آورد
  - علامت آدرس جابجایی را گسترش می‌دهد
  - دو واحد به سمت چپ شیفت می‌دهد
  - حاصل را به PC+4 اضافه می‌کند



دستورات پرش شرطی (ادامه...)

چگونه می‌توان سخت‌افزار گسترش علامت و شیفت را طراحی کرد؟

کافیست تنها سیم‌بندی را عوض کنیم!



کافیست تنها از بیت علامت چند انشعاب بگیریم!

معماری کامپیوتر (۱۳۹۱-۱۱-۱۳)

جلسه‌ی سیزدهم

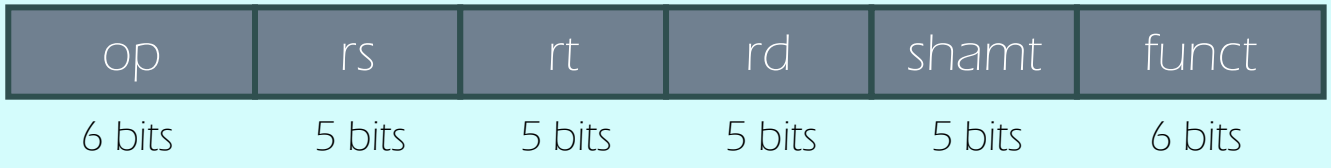
دانشگاه شهید بهشتی
دانشکده‌ی مهندسی برق و کامپیوتر
بهار ۱۳۹۱
احمد محمودی ازناوه

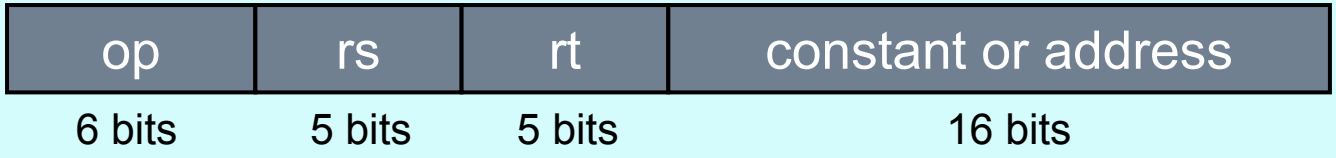
فهرست مطالب

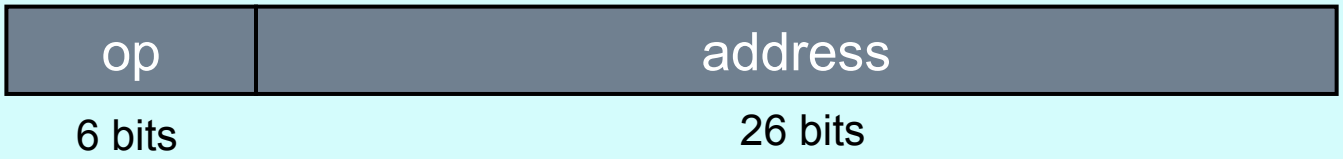
- مروری بر جلسه‌ی پیش
  - نحوه‌ی اجرای یک دستورالعمل
  - مسیر گذار داده
- واحد کنترل

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`

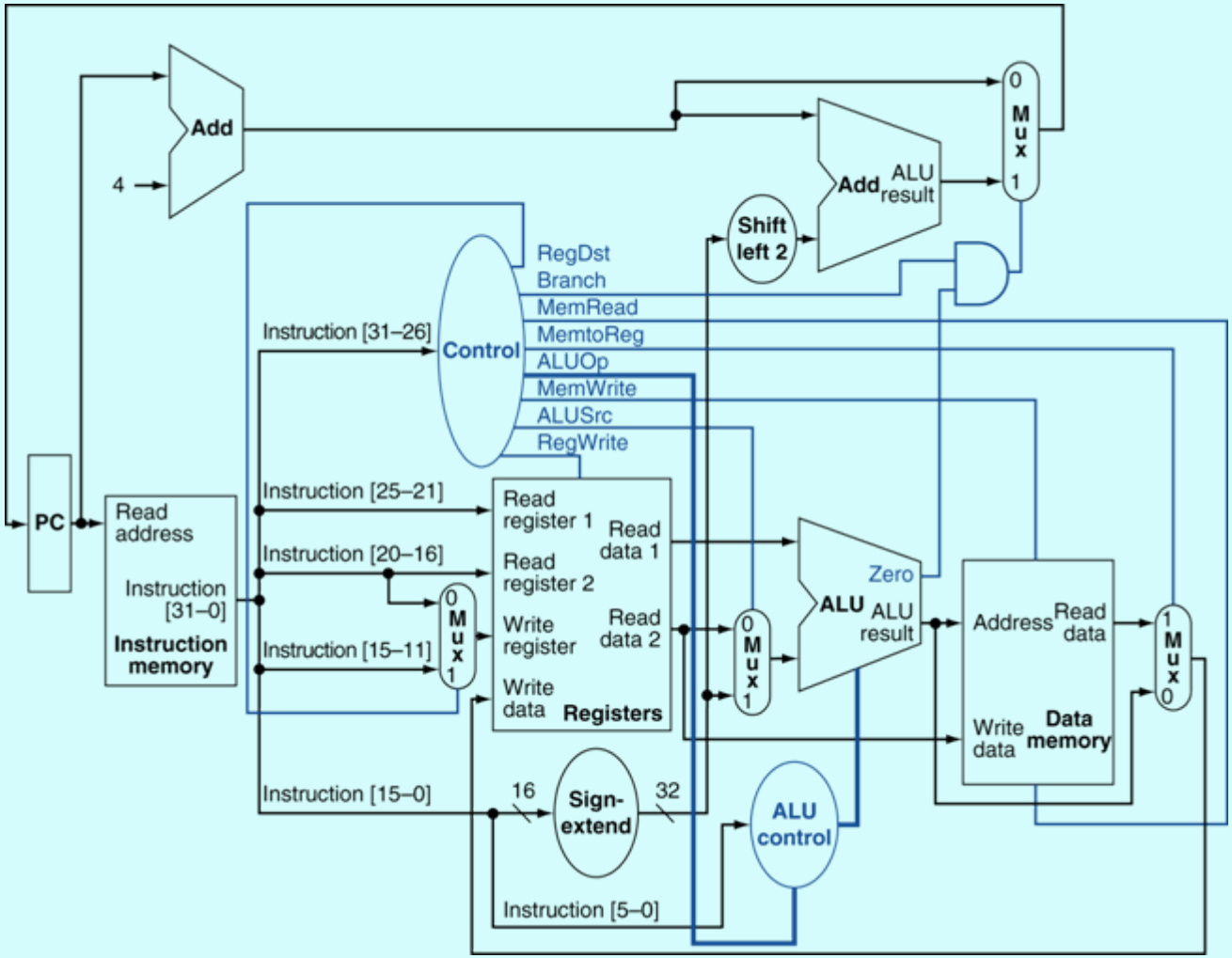






داده‌گذر همراه با واحد کنترل

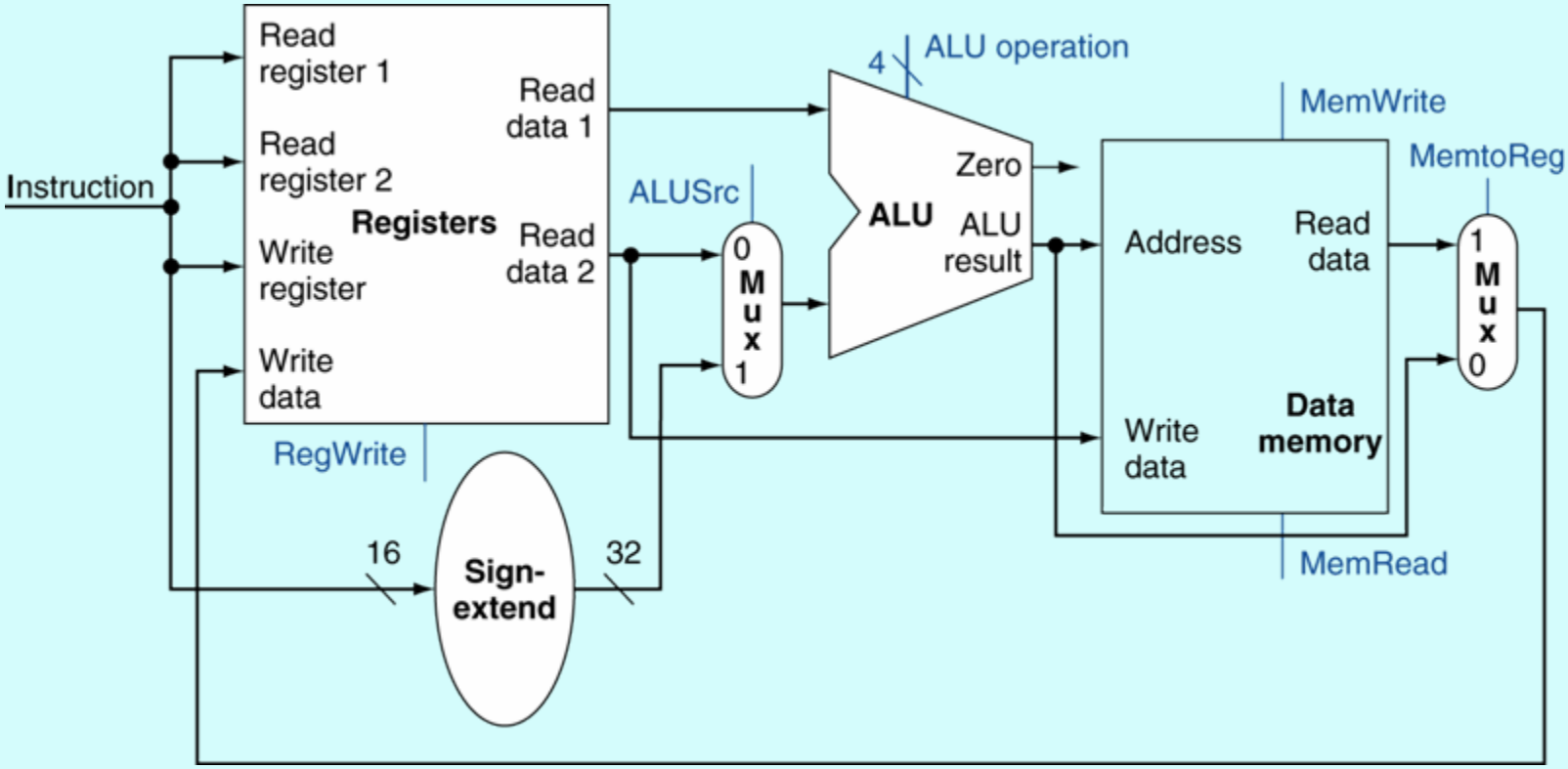
- با توجه به قالب دستور، می‌توان برچسب برخی سیگنال‌های داده و کنترلی را مشخص نمود:



ترکیب اجزا

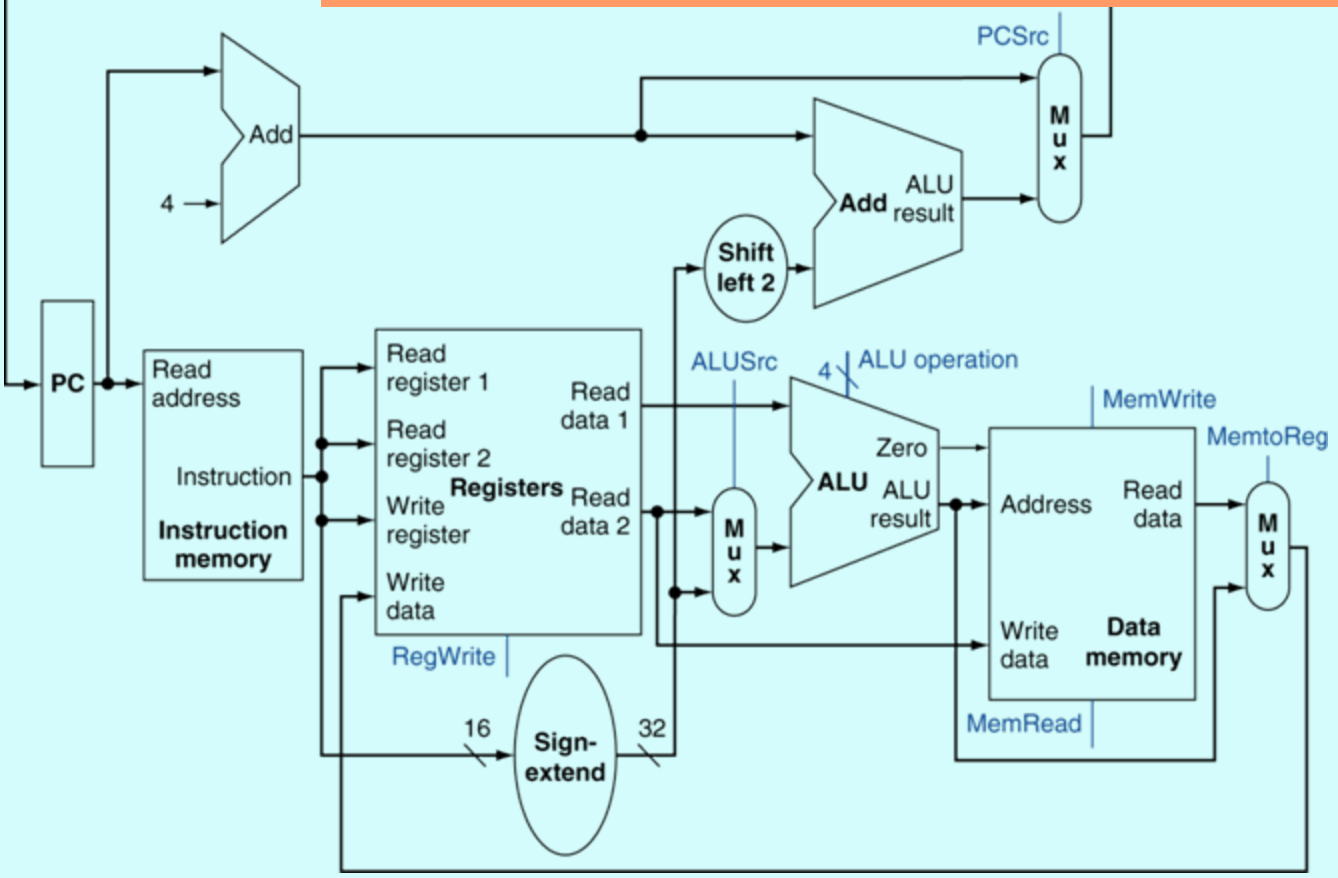
- با ترکیب اجزای مختلف، که برای دستورهای متنوع لازم هستند، یک «مسیر گذار داده» ساخته خواهد شد، که برای دستورهای مختلف توسط واحد کنترل هدایت می‌شود.
- ساده‌ترین داده گذر تمام دستورات را در یک سیکل اجرا خواهد کرد، در این صورت هیچ منبعی را نمی‌توان دوبار استفاده کرد.
- در این حالت باید از برخی منابع چند 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

ساخت‌افزار 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؟؟

واحد کنترل ALU

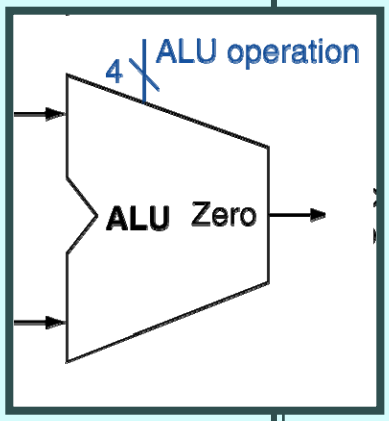
- در ادامه زیربخش کوچکی از دستورهای پردازنده‌ی MIPS را پیاده‌سازی خواهیم کرد.
- به داده‌گذر معرفی شده در بخش قبل، یک واحد کنترل برای اجرای دستورات زیر اضافه می‌کنیم:

- lw, sw
- beq
- arithmatic-logical instruction
  - add, sub, AND, OR, set on less than

- در ادامه، «دستور ل» را هم به این طرح ساده خواهیم افزود.

خطوط کنترلی ALU

ALU control	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set-on-less-than
1100	NOR



برای دستورات مختلف از خطوط کنترل ALU استفاده می‌کنیم:

- Load/Store: F = add
- Branch: F = subtract
- R-type: F depends on funct field

واحد کنترل ALU

- ورودی‌های واحد کنترل ALU:
  - دو بیت‌ورودی به نام ALUOp دارد
  - بخش funct دستورالعمل‌های نوع R

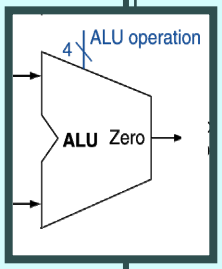
opcode	ALUOp	Operation	funct	ALU function	ALU control
lw	00	load word	XXXXXX	add	0010
sw	00	store word	XXXXXX	add	0010
beq	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		set-on-less-than	101010	set-on-less-than	0111

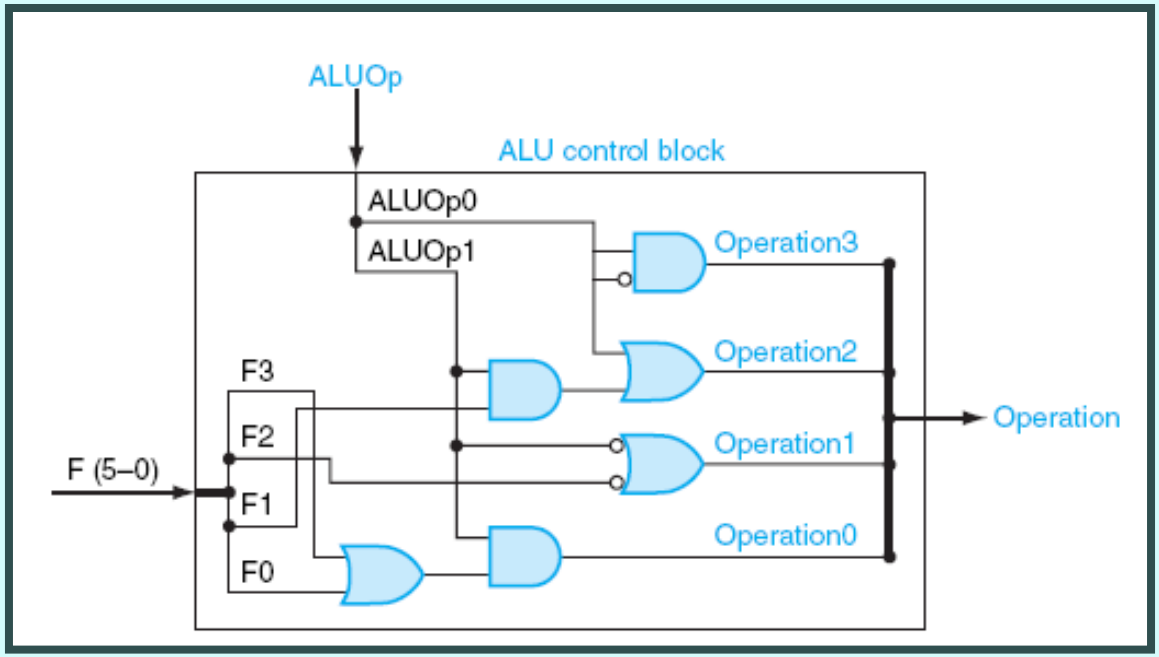
واحد کنترل ALU (ادامه...)

- استفاده از «واحد کنترل چندسطحی»، باعث کوچک شدن واحد کنترل اصلی می‌شود.
- چنین کاری می‌تواند باعث افزایش سرعت واحد کنترل شود.
  - سرعت واحد کنترل در تعیین سرعت پالس ساعت سیستم اهمیت دارد.
- در گام بعد، واحد کنترل ALU ساخته خواهد شد. به نظر شما بهترین شیوه‌ی پیاده‌سازی چیست؟

جدول درستی واحد کنترل ALU

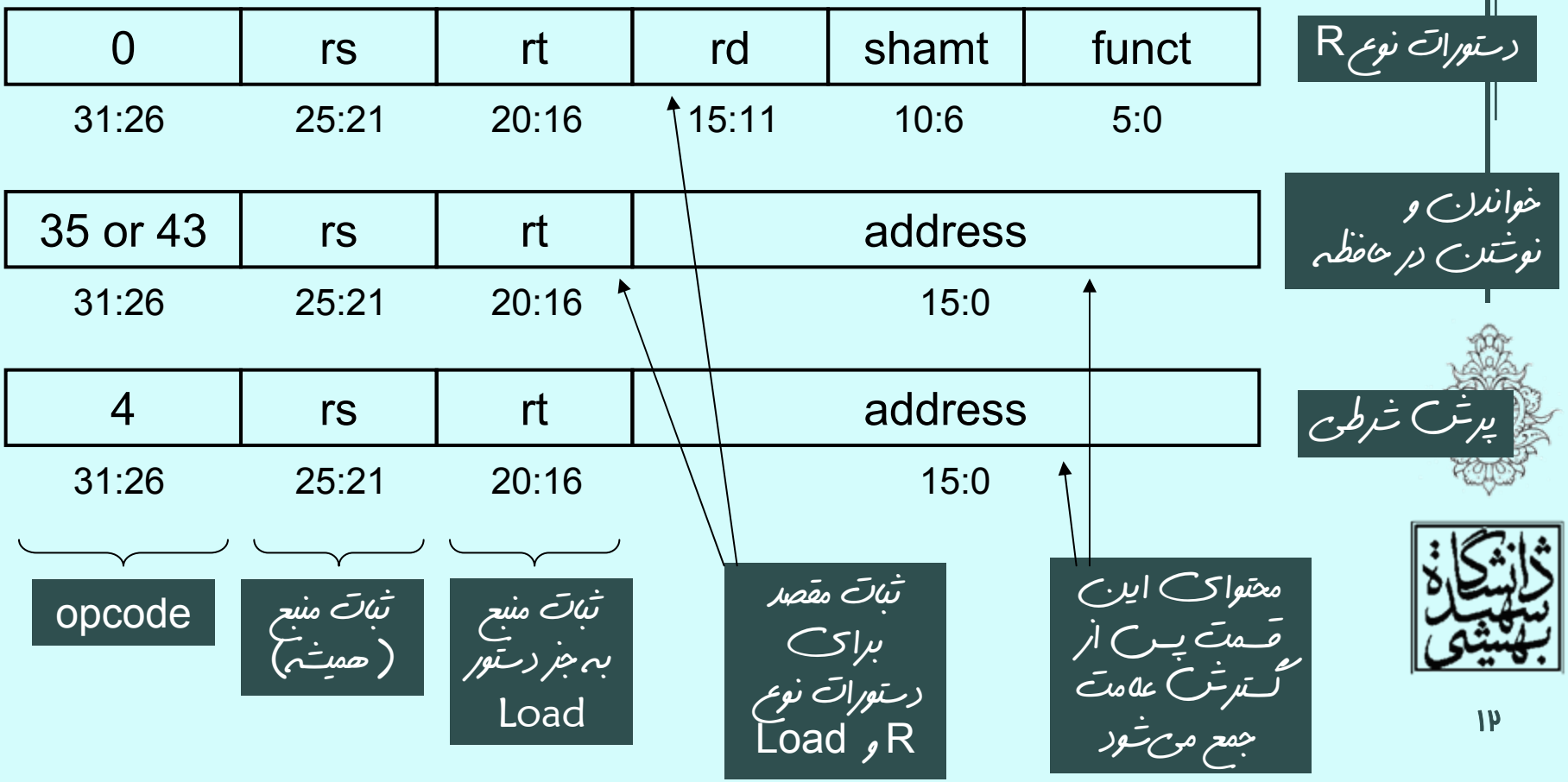
Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	AND	0000
R-type	10	OR	100101	OR	0001
R-type	10	set on less than	101010	set on less than	0111





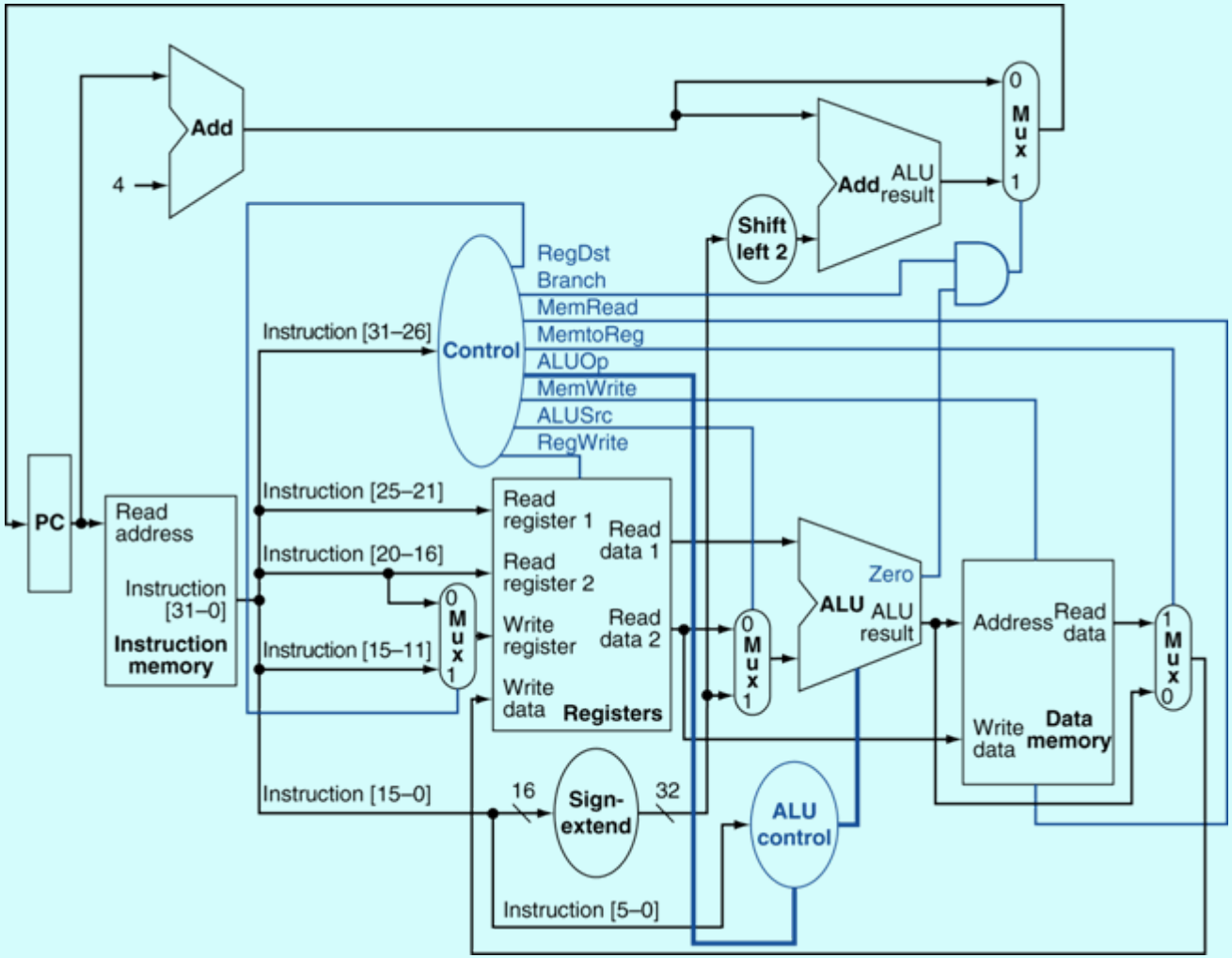
واحد کنترل اصلی

- در ادامه، واحد کنترل اصلی شرح داده خواهد شد.
- سیگنال‌های کنترلی که از دستورالعمل گرفته می‌شوند:



مسیر گذار داده همراه با واحد کنترل

- با توجه به قالب دستور، می‌توان برچسب برخی سیگنال‌های داده و کنترلی را مشخص نمود:

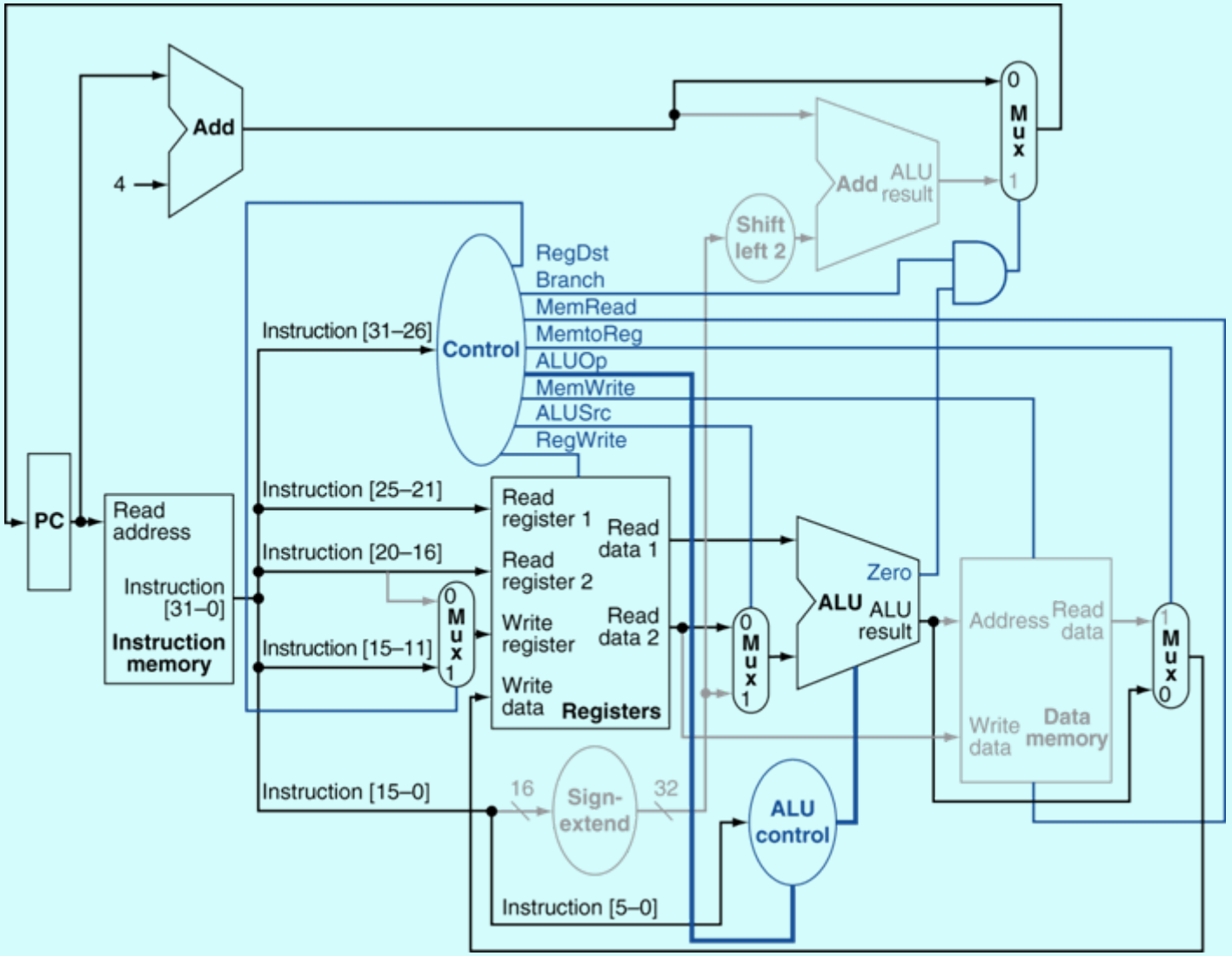


سیگنال‌های کنترلی

- سایر سیگنال‌های کنترلی، با توجه به opcode تعیین می‌شود.
- تنها PCSrc استثناست که به نتیجه‌ی ALU وابسته است.

Signal name	Effect when deasserted	Effect when asserted
RegDst	The register destination number for the Write register comes from the rt field (bits 20:16).	The register destination number for the Write register comes from the rd field (bits 15:11).
RegWrite	None.	The register on the Write register input is written with the value on the Write data input.
ALUSrc	The second ALU operand comes from the second register file output (Read data 2).	The second ALU operand is the sign-extended, lower 16 bits of the instruction.
PCSrc	The PC is replaced by the output of the adder that computes the value of PC + 4.	The PC is replaced by the output of the adder that computes the branch target.
MemRead	None.	Data memory contents designated by the address input are put on the Read data output.
MemWrite	None.	Data memory contents designated by the address input are replaced by the value on the Write data input.
MemtoReg	The value fed to the register Write data input comes from the ALU.	The value fed to the register Write data input comes from the data memory.

نحوه‌ی اجرای دستورات نوع R



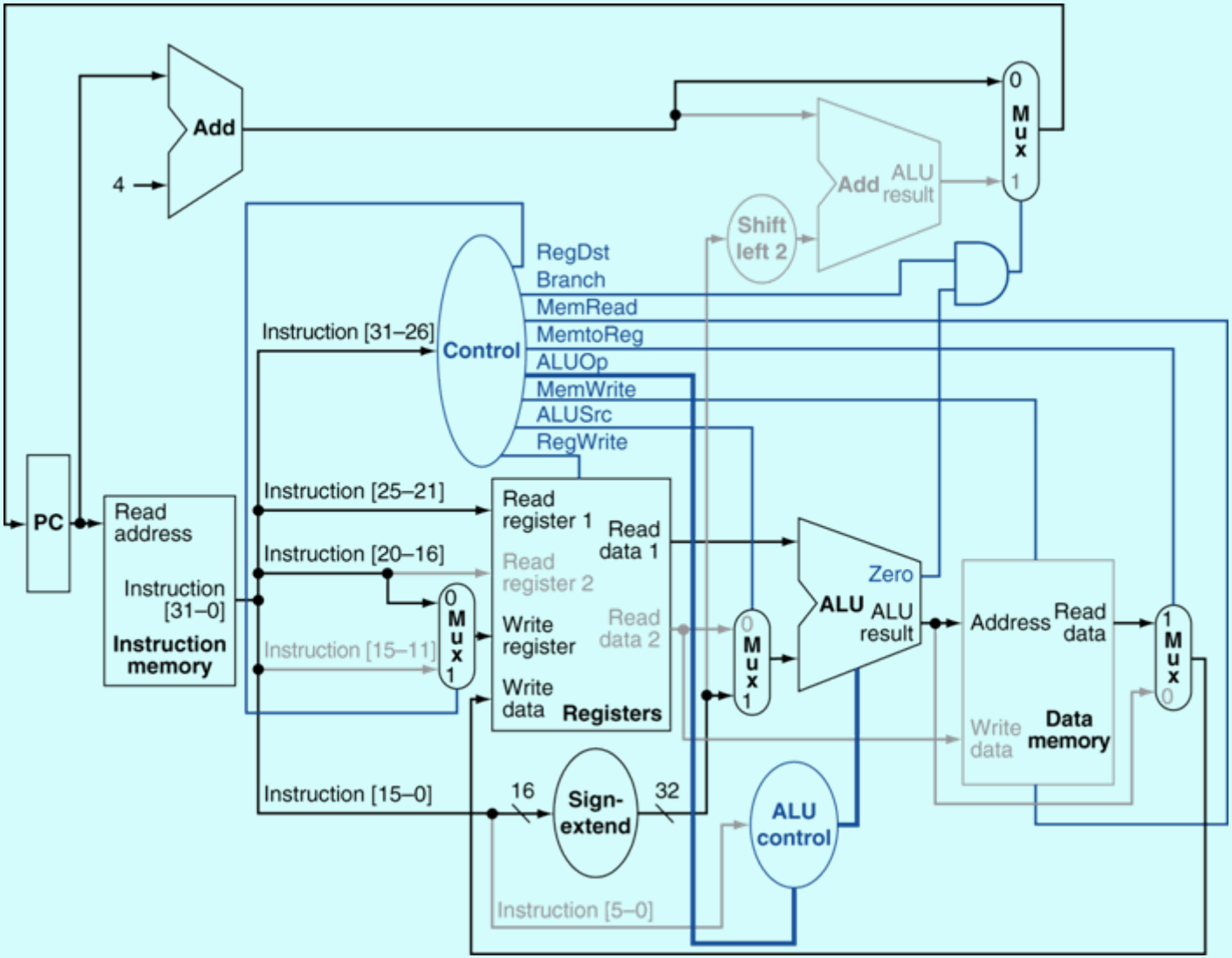
نحوه‌ی اجرای دستورات نوع R (ادامه...)

- هرچند همه‌ی دستورها در یک گام اجرا می‌شوند، می‌توان مراحل اجرای یک دستور را در چهار گام تصور نمود:

```
add $t1,$t2,$t3
```

- واکشی دستور و افزایش مقدار PC
- خواندن ثبات‌های $t2 و $t3 از بانک ثبات
- واحد کنترل ALU از روی بیت‌های funct خطوط کنترلی را برای انتخاب عمل مناسب انتخاب می‌کند.
- نتیجه‌ی محاسبات در ثبات مقصد ($t1) نوشته می‌شود.

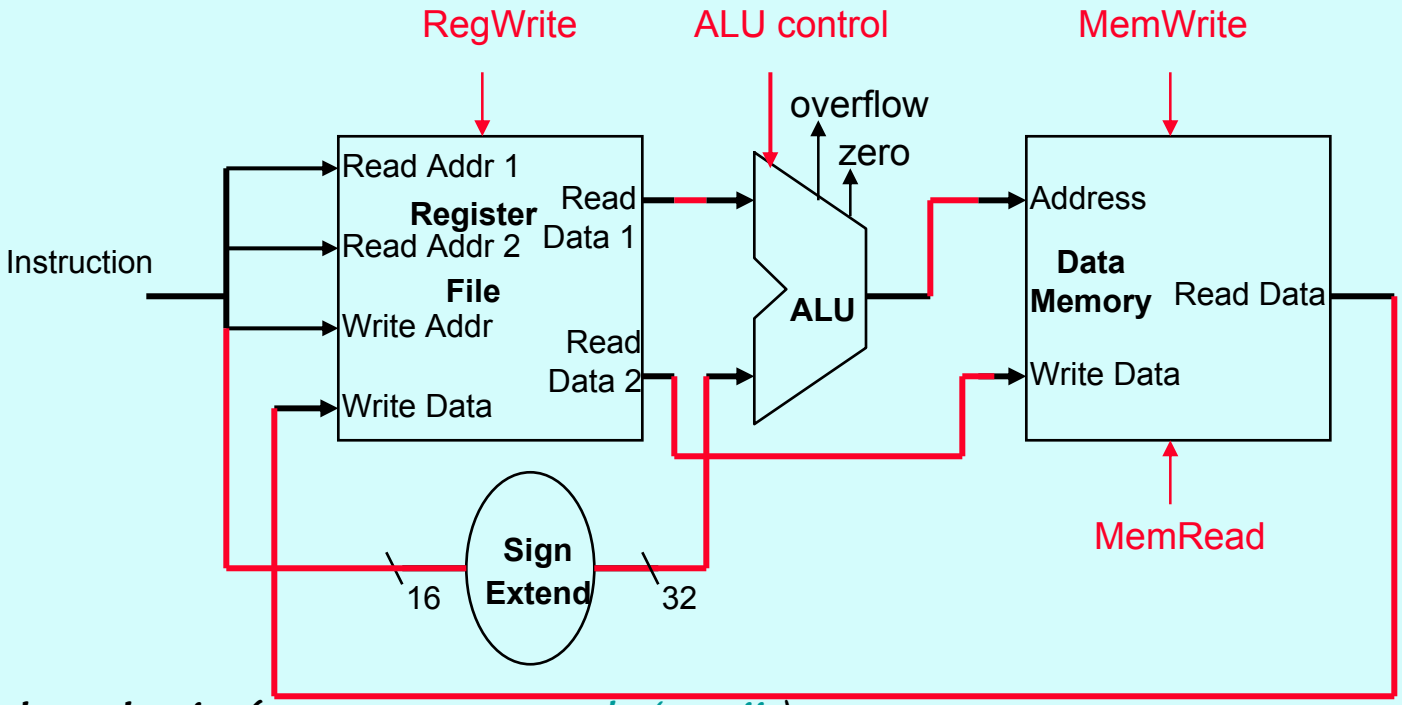
نحوه‌ی اجرای دستورات خواندن(نوشتن) از(در) حافظه



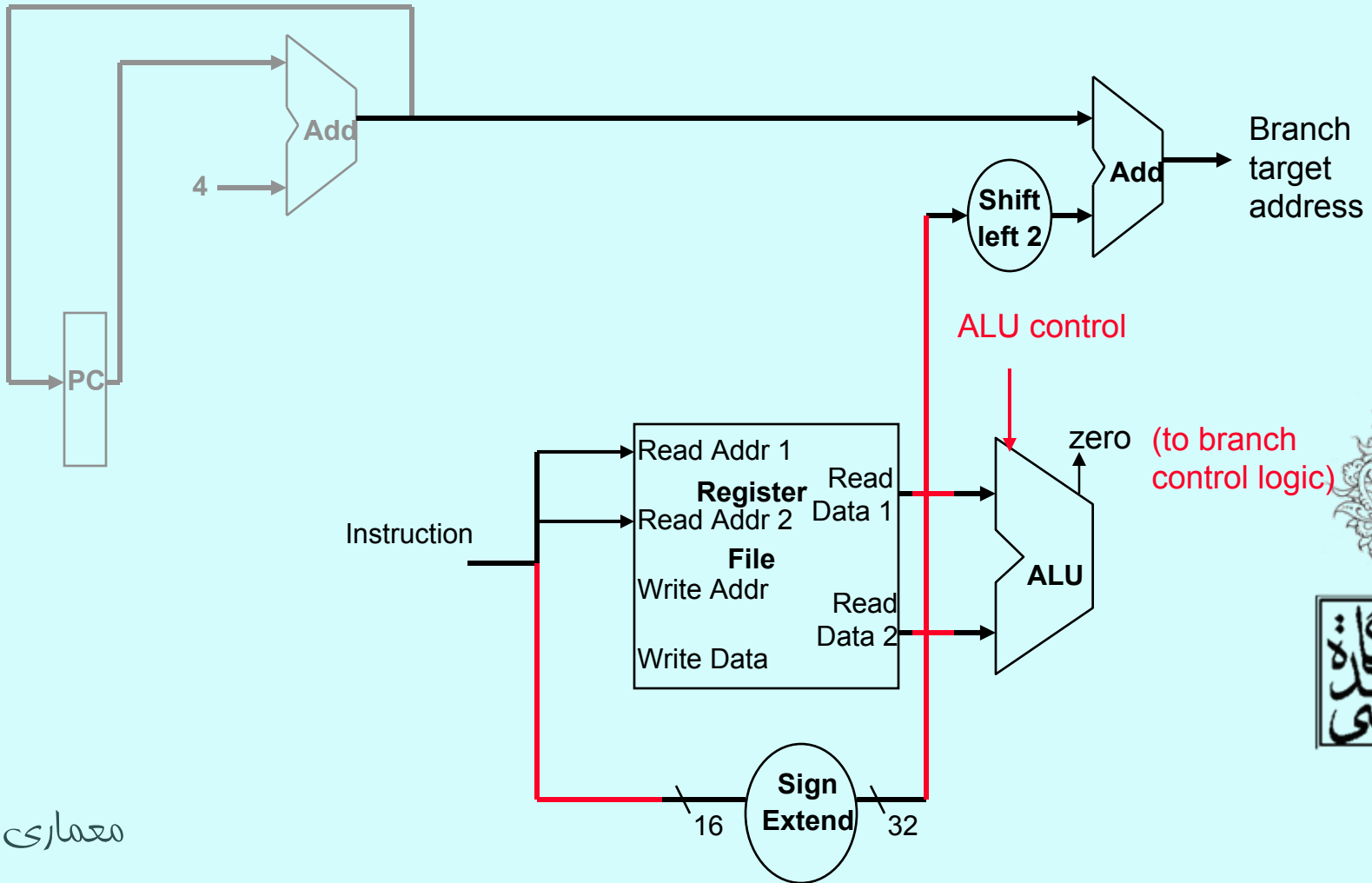
اجرای دستورات خواندن از حافظه (ادامه...)

```
lw $t1,offset($t2)
```

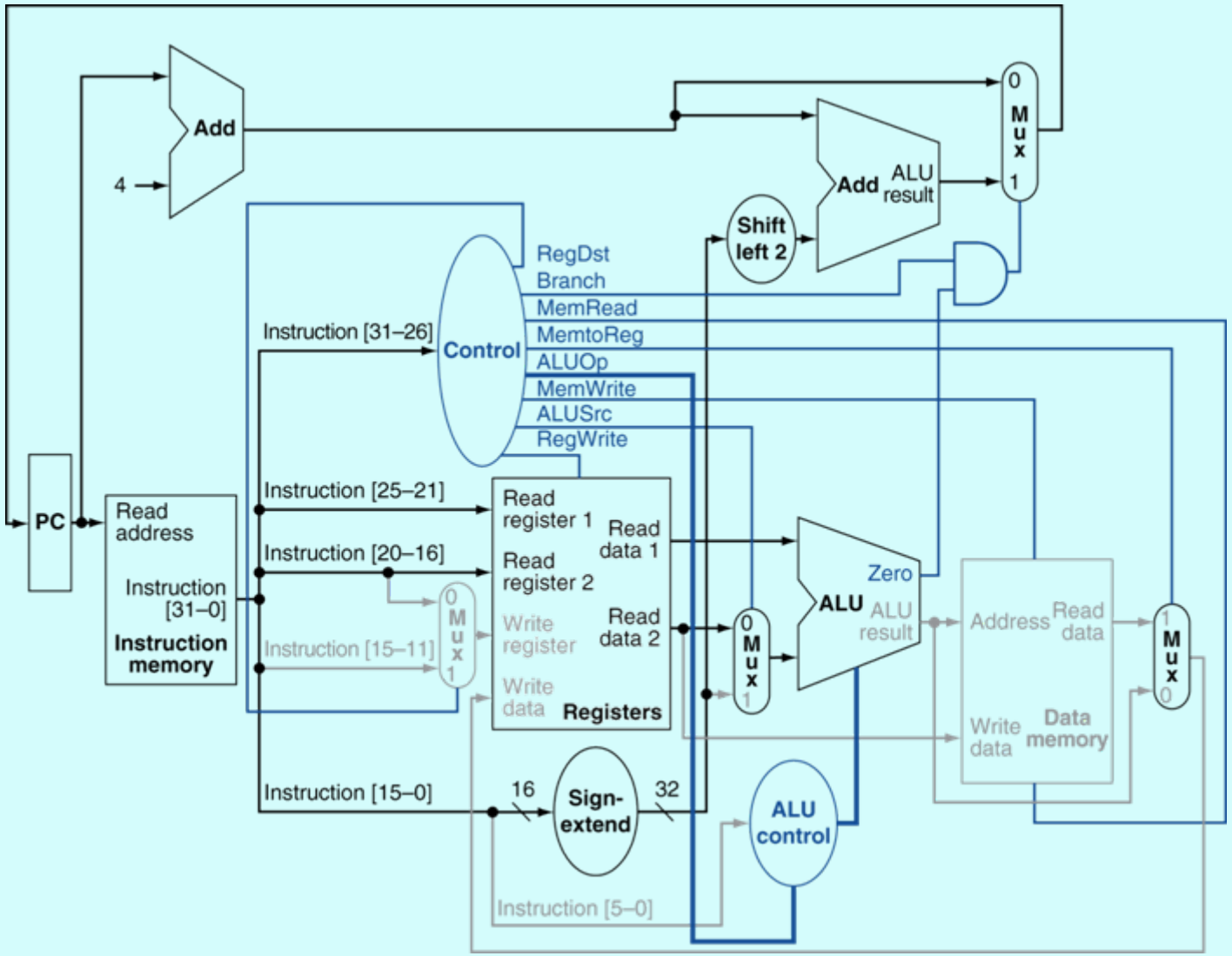
- واکشی دستور و افزایش مقدار PC
- خواندن ثبات $t2 از بانک ثبات
- ALU محتوای ثبات را با offset جمع می‌کند.
- حاصل به عنوان آدرس حافظه مورد استفاده قرار می‌گیرد.
- داده‌ی خوانده شد در حافظه در ثبات مقصد ($t1) نوشته می‌شود.



نحوه‌ی اجرای دستورات پرش شرطی



نحوه‌ی اجرای دستورات پرش شرطی (ادامه...)



نحوه‌ی اجرای دستورات پرش شرطی (ادامه...)

```
beq $t1,$t2,offset
```

- واکشی دستور و افزایش مقدار PC
- خواندن ثبات‌های $t1 و $t2 از بانک ثبات
- ALU عمل تفریق را انجام می‌دهد، بخش ثابت پس از گسترش علامت و شیفت به چپ به PC+4 اضافه می‌شود
- بر اساس خروجی zero در ALU در مورد این آدرس PC چه باشد، تصمیم‌گیری می‌شود.

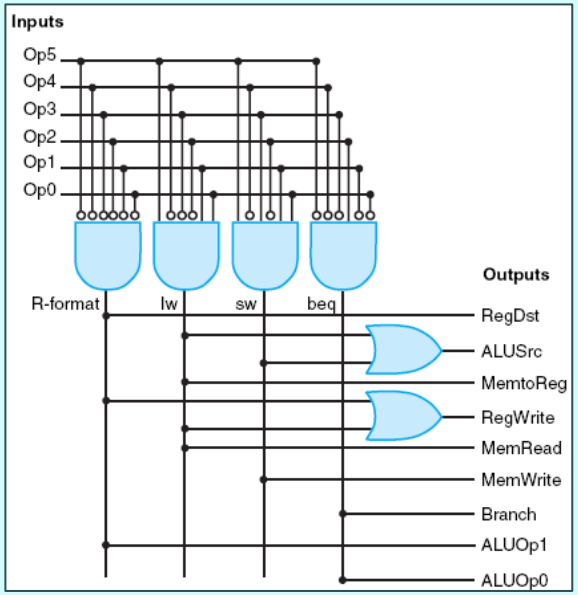
واحد کنترل اصلی

- خروجی واحد کنترل، سیگنال‌های کنترلی هستند.
- شش بیت opcode، ورودی واحد کنترل محسوب می‌شوند.

Instruction	RegDst	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

واحد کنترل اصلی (ادامه...)

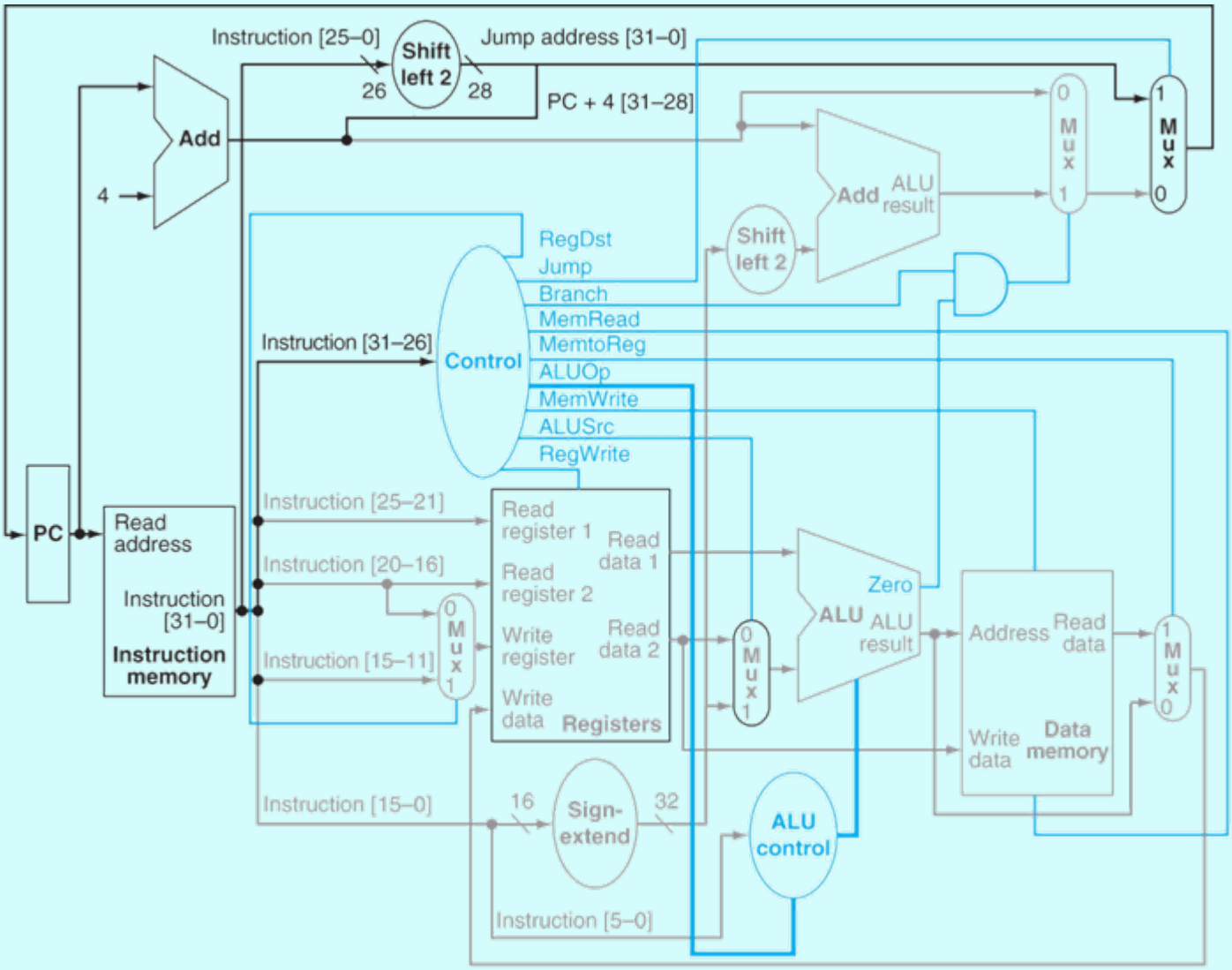
Input or output	Signal name	R-format	lw	sw	beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1



افزودن دستور پرش (Jump)

- چهار بیت پرارزش آدرس از PC+4 گرفته می‌شود.
- در دو بیت کم‌ارزش 00 قرار می‌گیرد.
- بیست و شش بیت دیگر از بخش آدرس دستورالعمل گرفته می‌شود.

افزودن دستور پرش (ادامه...)



معایب اجرای همه‌ی دستورها در یک سیکل

- کامپیوترهای اولیه از چنین ساختار تبعیت می‌کردند.
- طول سیکل ساعت باید با توجه به کندترین دستور انتخاب شود.

Making the common case fast

- این مسأله با اصول طراحی در تناقض است.
- در بخش بعدی شیوه‌ای دیگر، به نام خط لوله را بررسی خواهیم کرد.

معماری کامپیوتر (۱۳۹۱-۱۱-۱۳)

جلسه‌ی چهاردهم

دانشگاه شهید بهشتی

دانشکده‌ی مهندسی برق و کامپیوتر

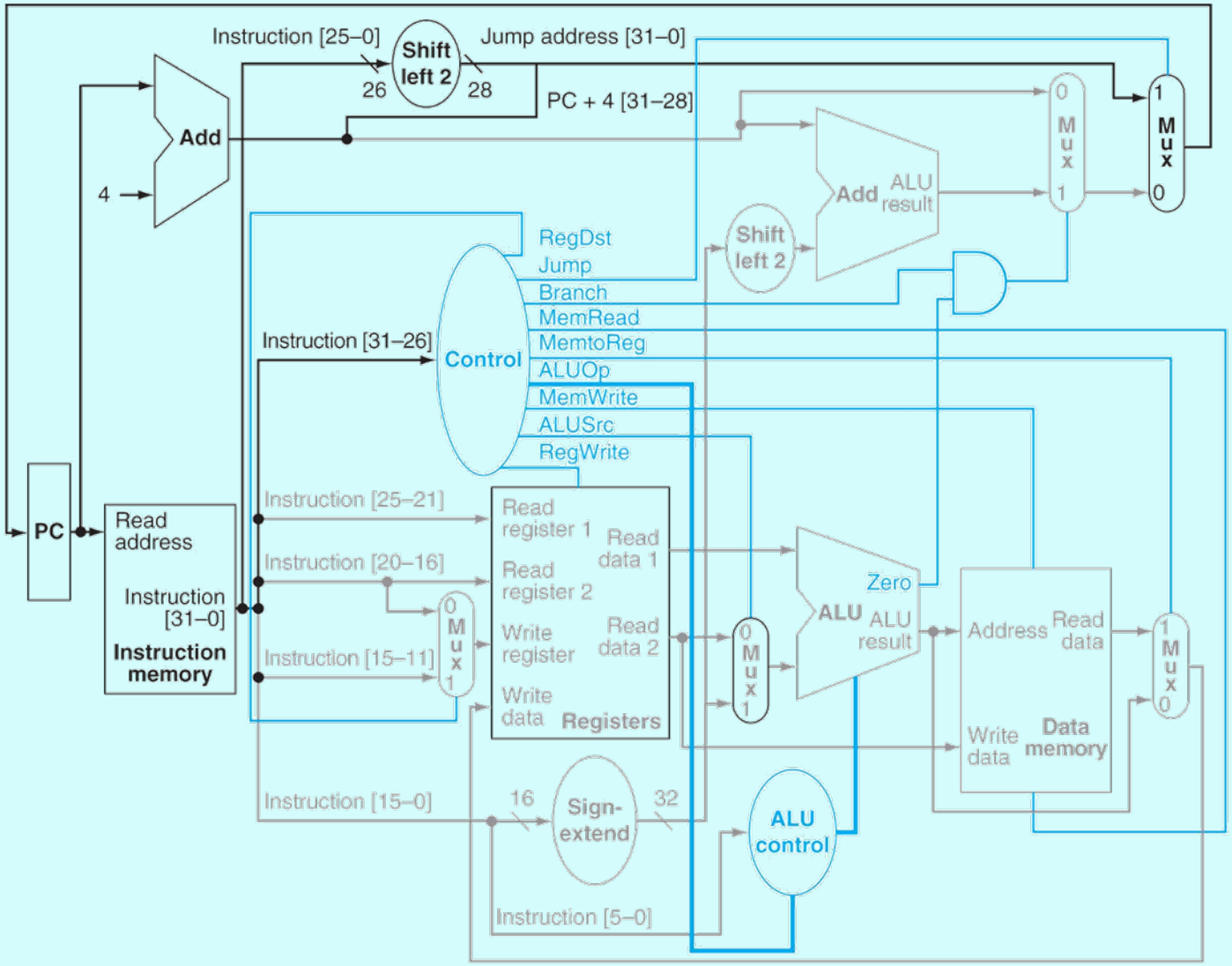
بهار ۱۳۹۱

احمد محمودی ازناوه

فهرست مطالب

- مروری بر جلسه‌ی پیش
- مراحل اجرای دستورالعمل‌ها
- اجرای دستور در چند سیکل
- خط لوله

مسیر گذار داده



مراحل اجرای دستورالعمل در MIPS

- در پردازنده‌های MIPS برای اجرای دستورات به صورت خط لوله، هر دستور در پنج گام انجام می‌شود.
  - واکشی دستورات
  - خواندن محتوای ثبات‌ها و کدگشایی دستورالعمل (قالب منظم MIPS چنین امکانی را مهیا می‌سازد)
  - اجرای دستورالعمل-محاسبه‌ی آدرس
  - دسترسی به حافظه
  - نوشتن پاسخ در ثبات

IF: Instruction fetch

ID: Instruction decode & register read

EX: Execute operation or calculate address

MEM: Access memory operand

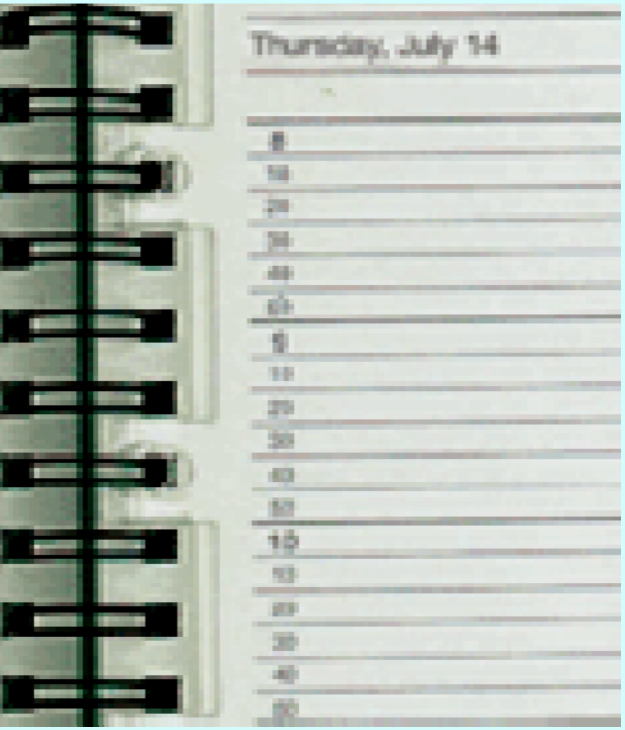
WB: Write result back to register

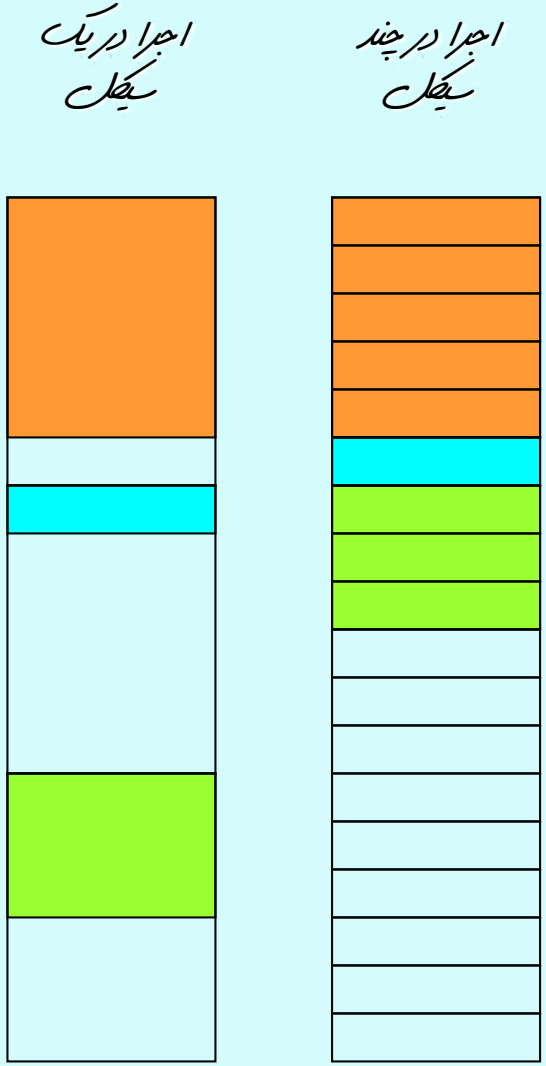
کارایی مسیر گذار داده‌ی تک سیکلی

- فرض می‌کنیم زمان مورد نیاز برای هر کدام از بخش‌های پردازنده به صورت زیر باشد:
  - برای نوشتن در ثبات 100ps
  - دستیابی به حافظه و محاسباتی و ... 200ps
- در معماری یک سیکلی، باید طول پالس را برابر با طول کندترین دستوالعمل در نظر گرفت.

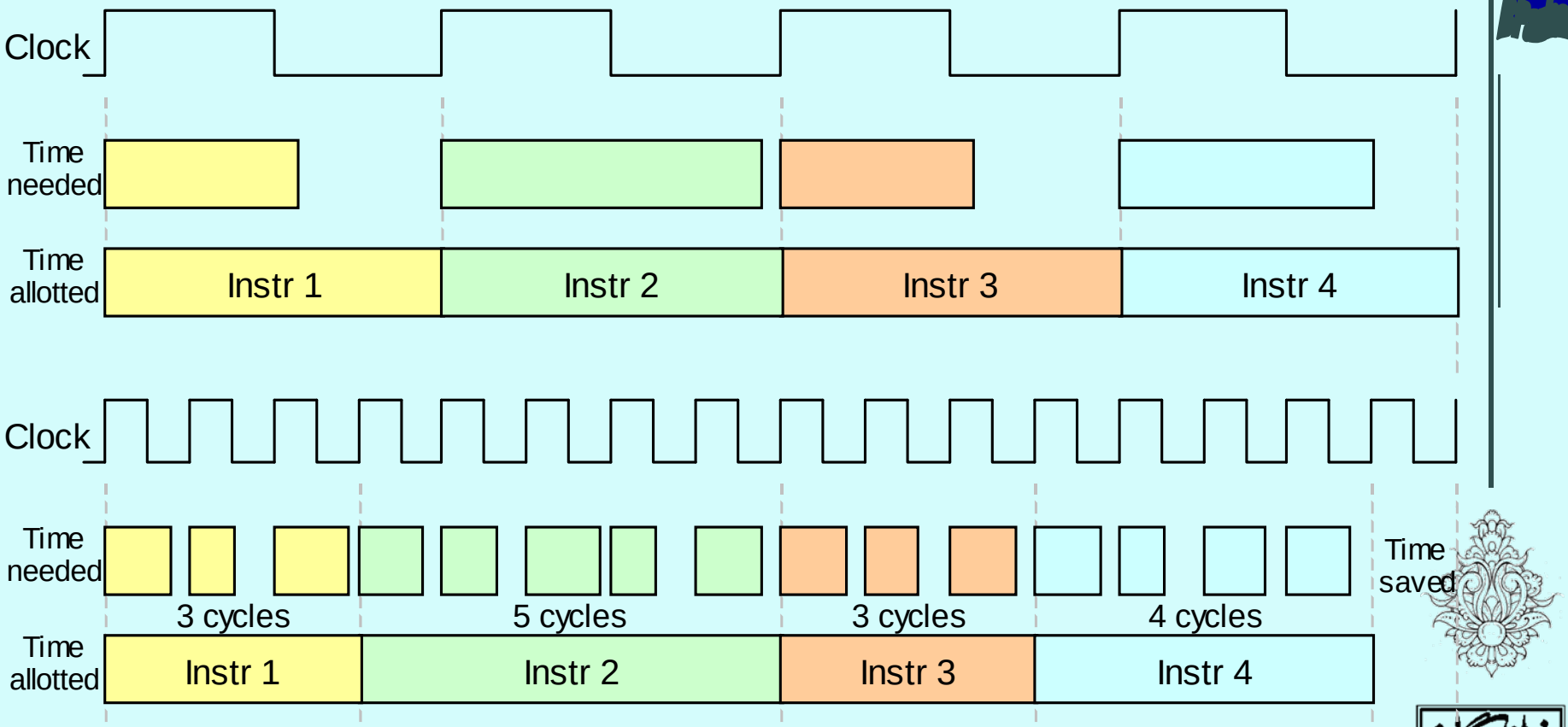
Instr	Instr fetch	Register read	ALU op	Memory access	Register write	Total time
lw	200ps	100 ps	200ps	200ps	100 ps	800ps
sw	200ps	100 ps	200ps	200ps		700ps
R-format	200ps	100 ps	200ps		100 ps	600ps
beq	200ps	100 ps	200ps			500ps

اجرای هر دستورالعمل در چند سیکل(در برابر اجرا در یک سیکل)

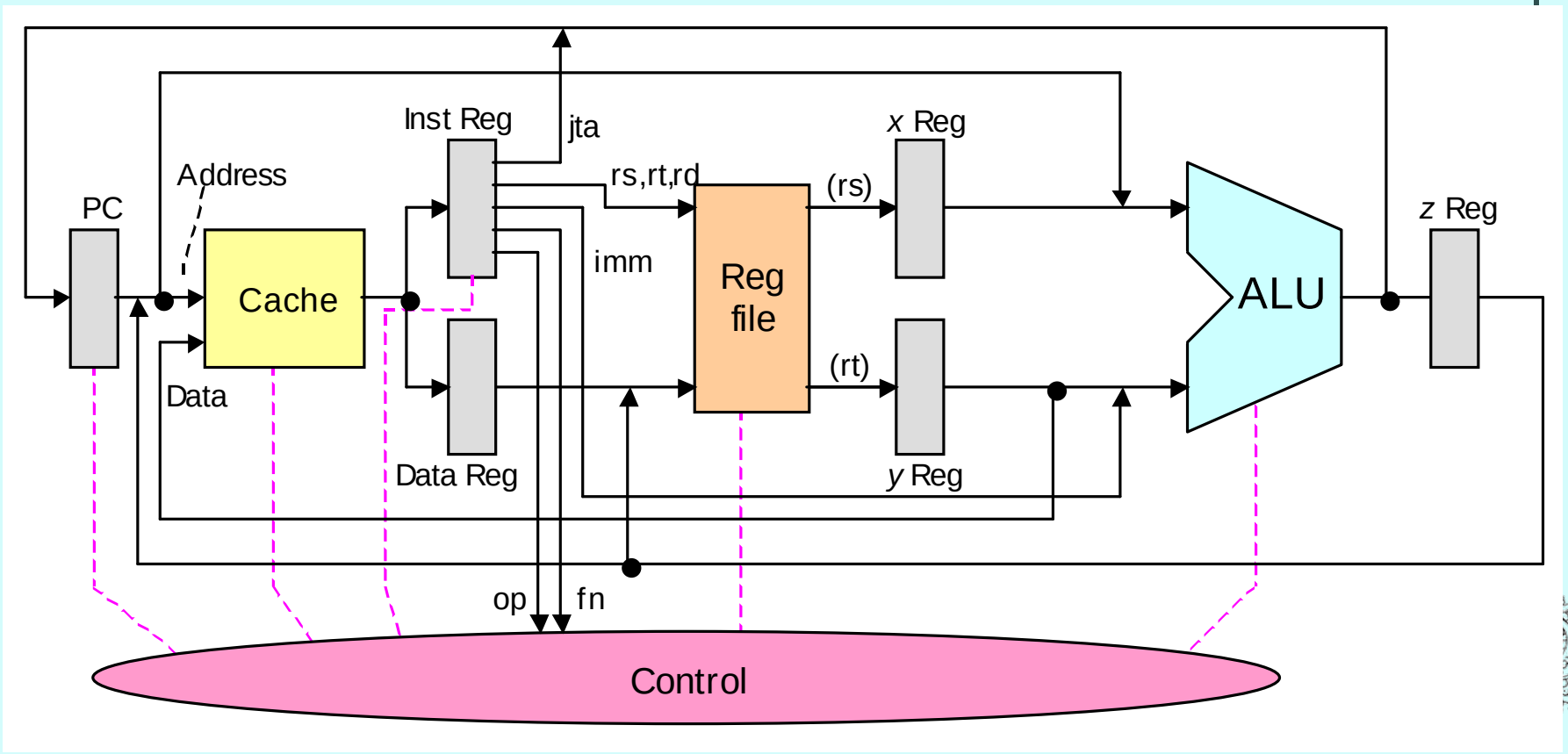




اجرای هر دستورالعمل در چند سیکل(در برابر اجرا در یک سیکل)



مسیر گذار داده‌ی چند سیکلی



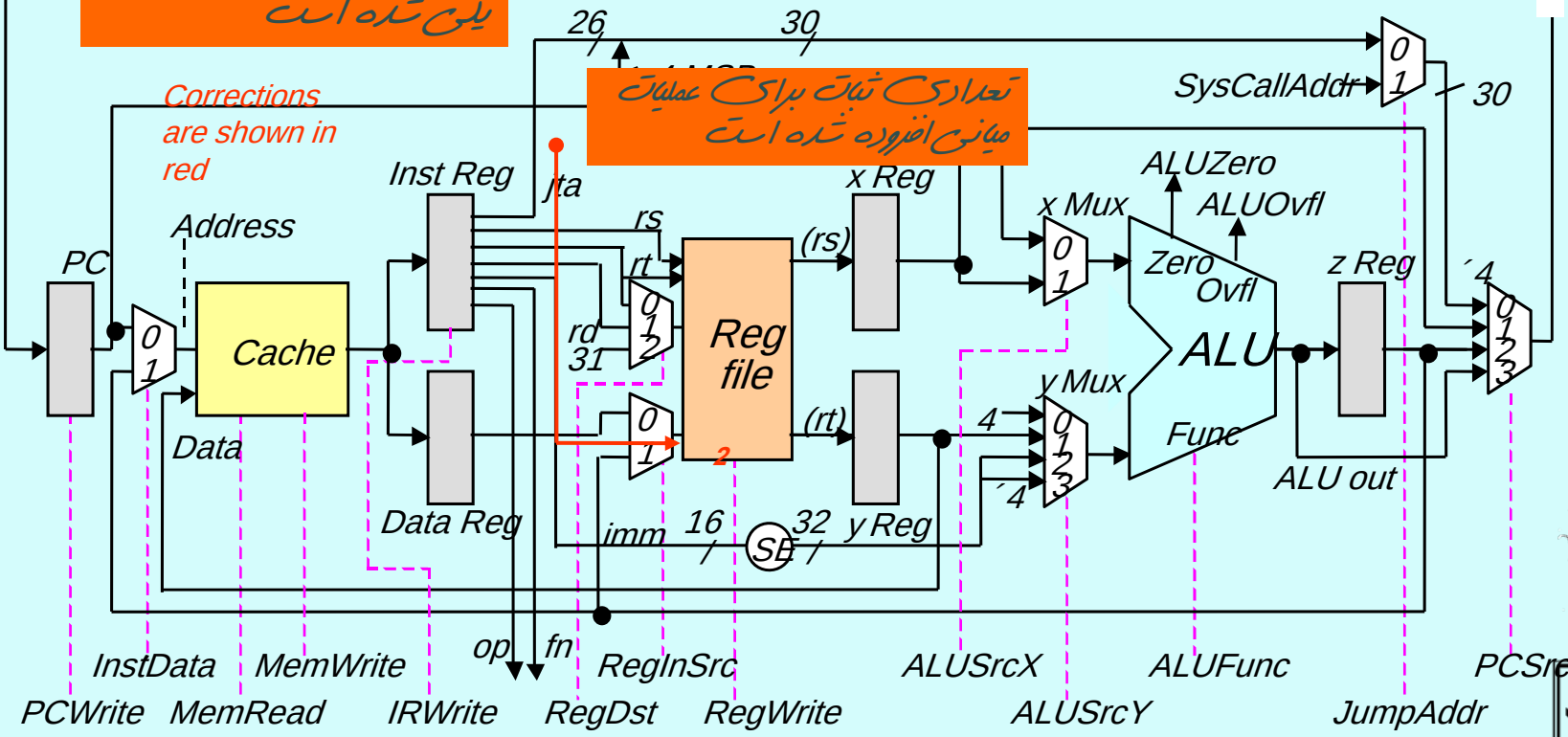
داده‌گذر در حالت چند سیکلی

سه تفاوت عمده با حالت تک سیکلی دارد

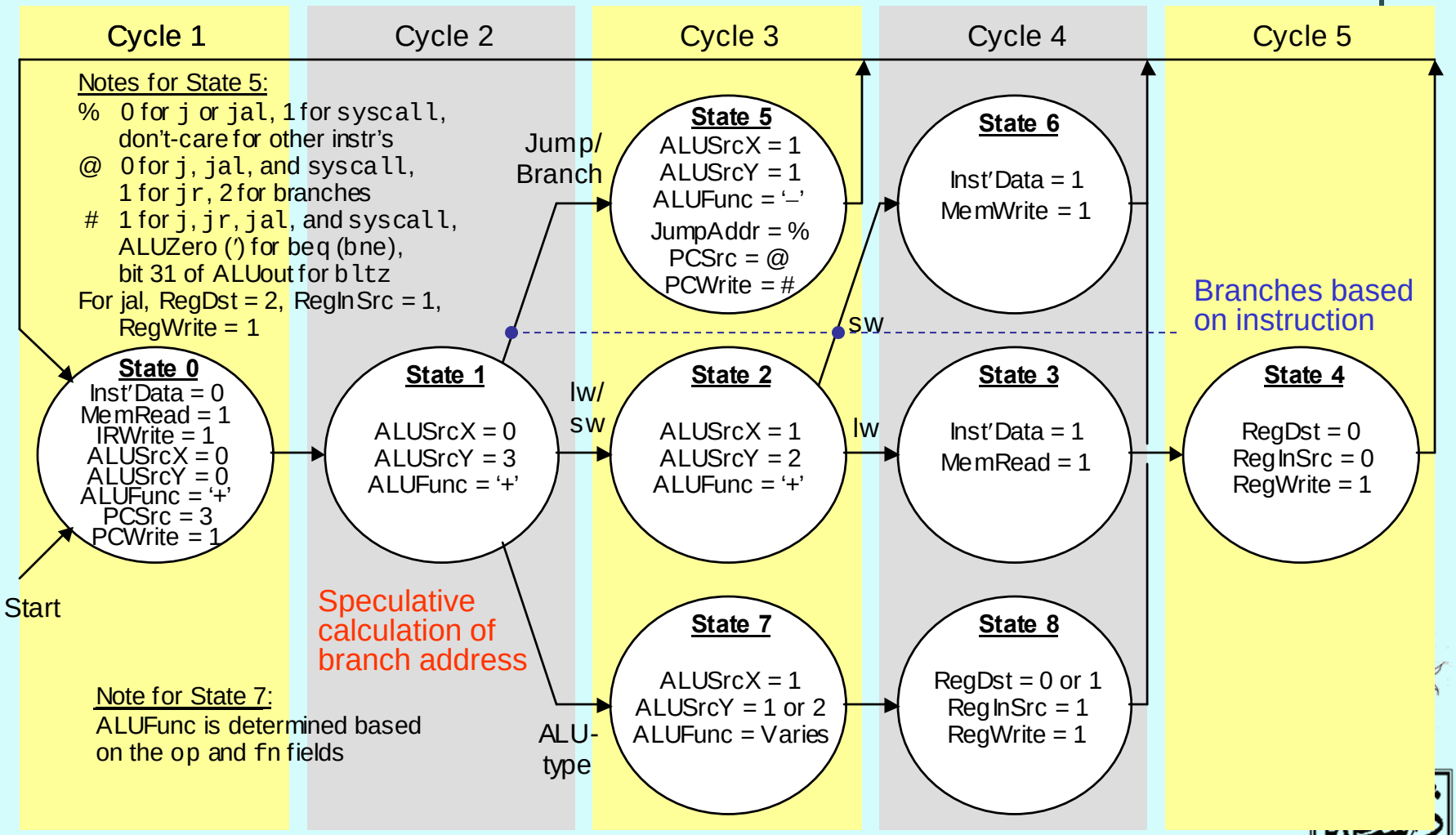
واحد ALU وظیفه‌ی محاسبه‌ی آدرس دستور بعدی را هم دارد

حافظه‌ی برنامه و داده یکی شده است

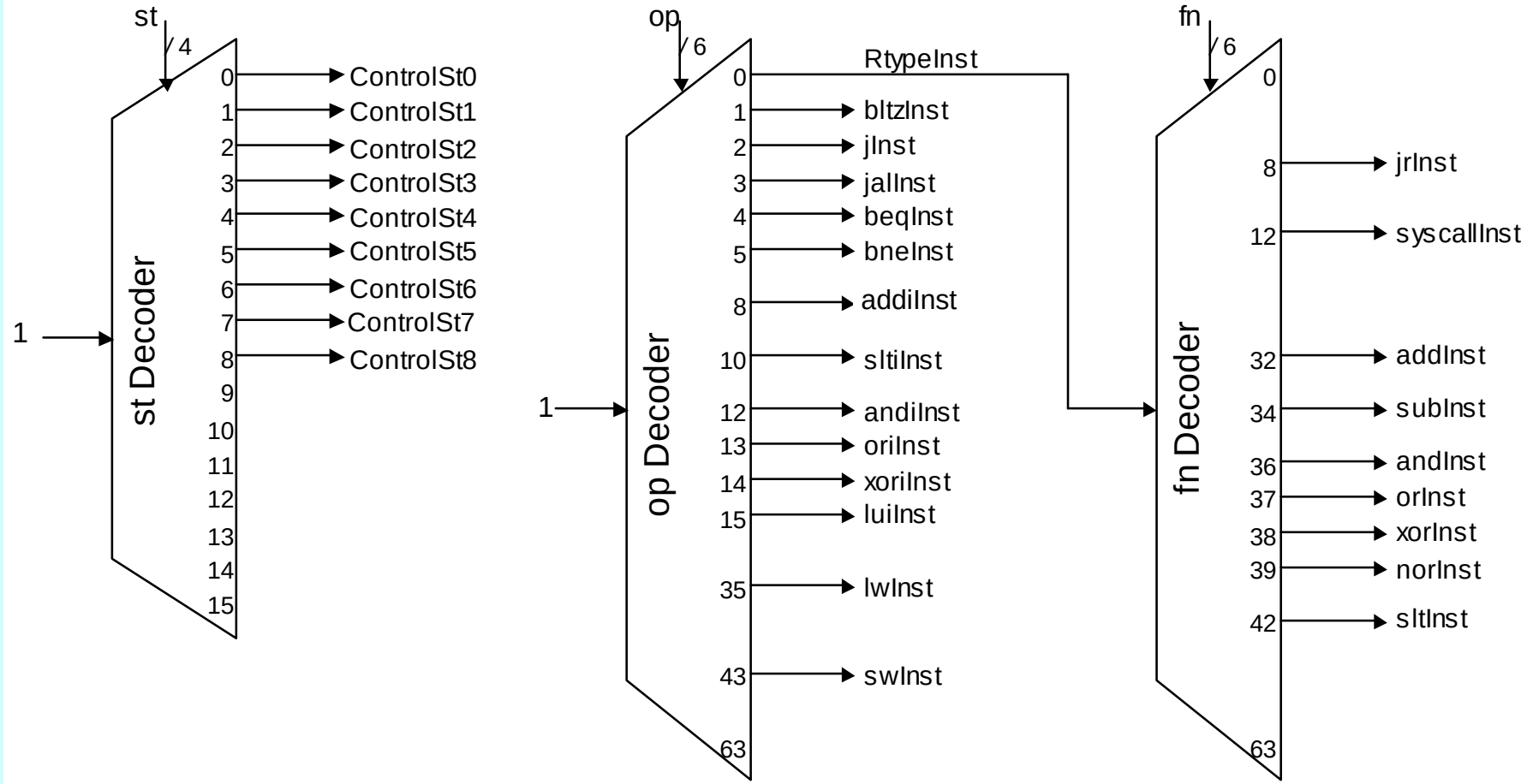
تعدادی ثبات برای عملیات میانی اضافه شده است



ماشین حالت واحد کنترل

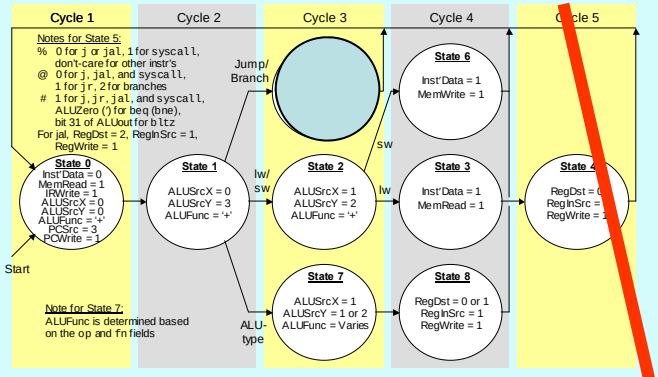


رمزگشایی دستورالعمل‌ها در گام‌های مختلف

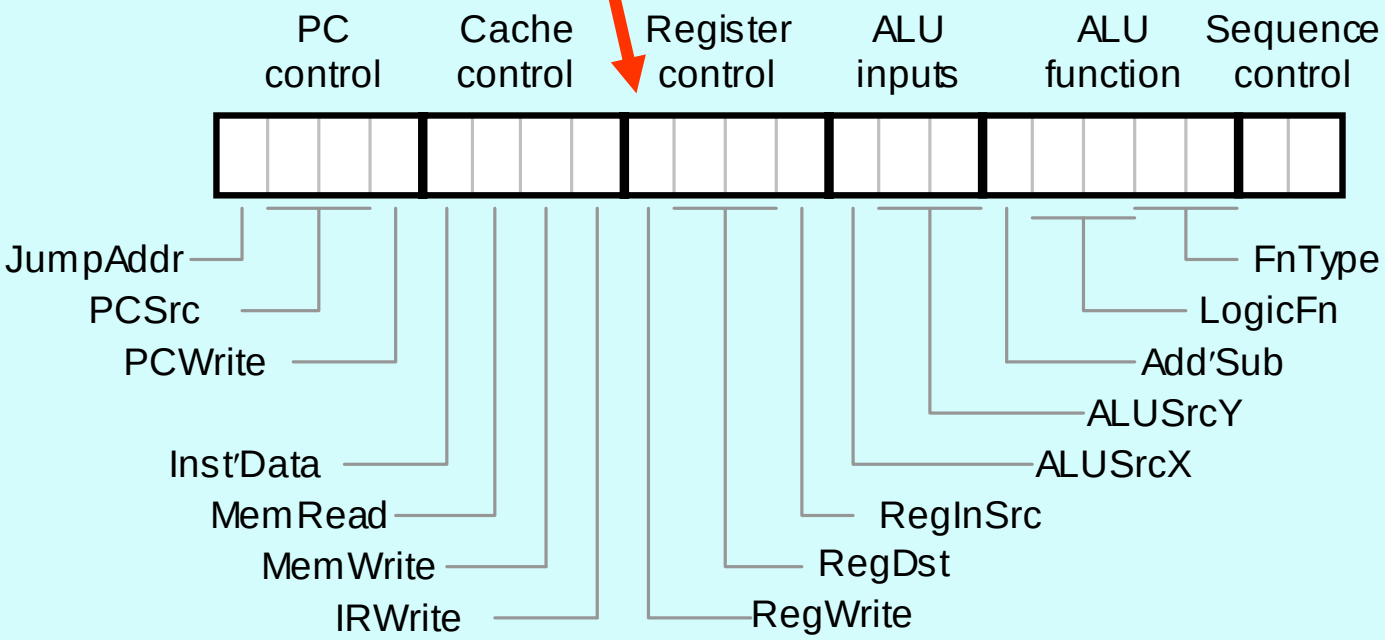


ریزبرنامه

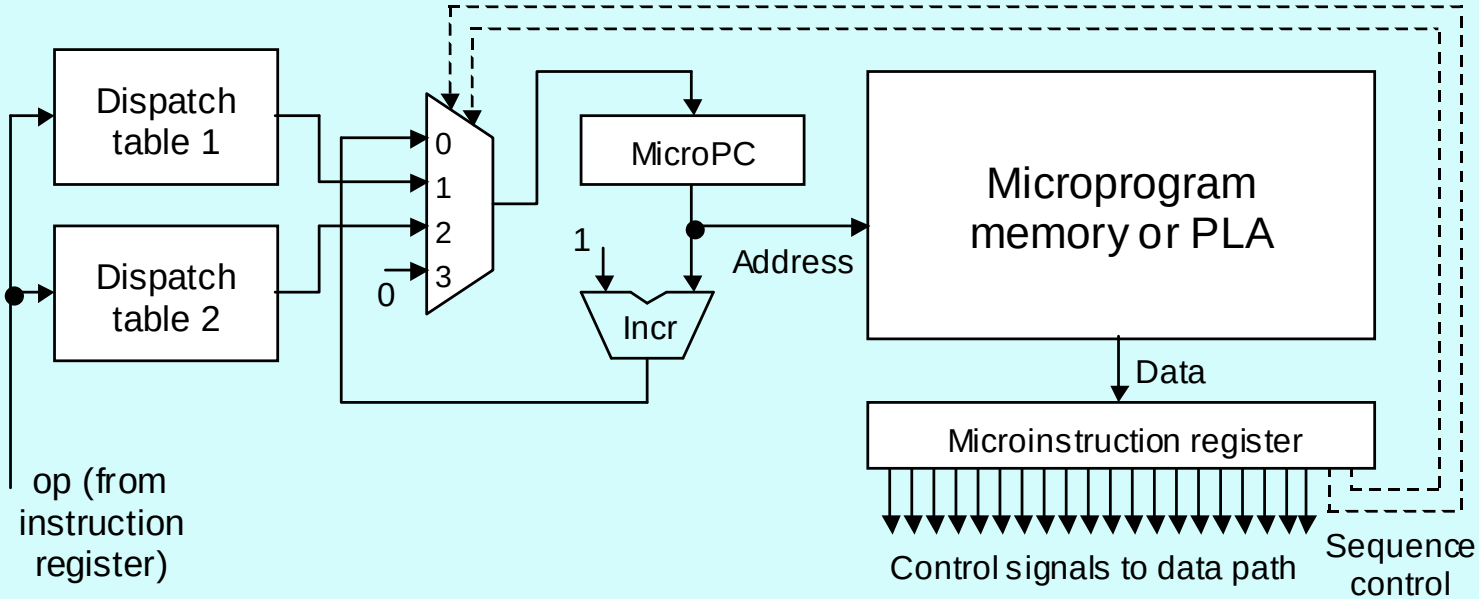
ریزبرنامه



ماشین حالت استفاده شده برای خطوط کنترلی، خود شبیه به یک برنامه است!



ریزبرنامه (ادامه...)



CISC در مقابل RISC

RISC

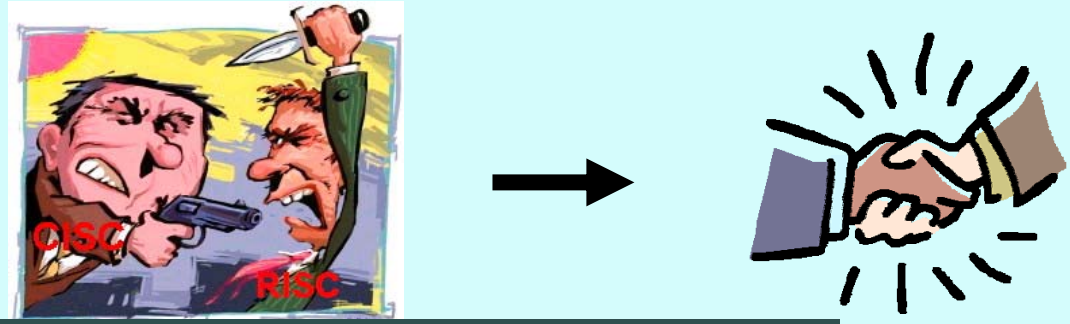
- تعداد دستورات کم
- طول دستورات ثابت
- زمان اجرای ثابت
- هزینه‌ی پایین
- تنها دستورات خواندن و نوشتن به حافظه دسترسی دارند
- همه‌ی عملوندها ثبات‌های پردازنده هستند
- مودهای آدرس محدود
- واحد کنترل به صورت سیم‌بندی

CISC

- دارای دستورات پیچیده و متنوع
  - حتی دستوراتی که کم‌تر به کار می‌روند
- طول دستورات متغیر
- میکروکدهای پیچیده
- بیشتر دستورات به حافظه دسترسی دارند
- مودهای آدرس‌دهی بسیار متنوع هستند

CISC در مقابل RISC (ادامه...)

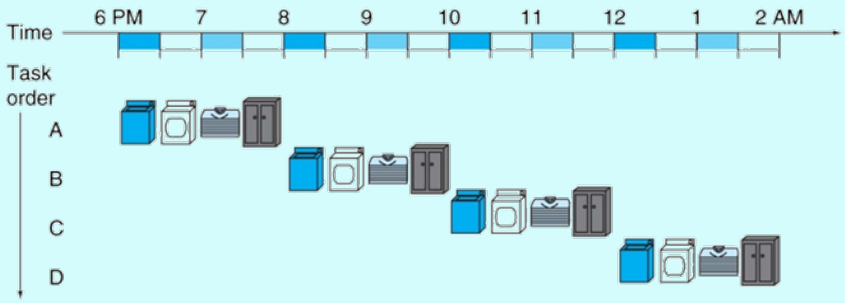
- در عمل مرزهای بین این دو در حال محو شدن هستند.
- پردازنده‌های جدید از خصوصیات هر دو بهره می‌گیرند.
- با این وجود، برای سیستم‌های درون‌کار (توکار) پردازنده‌ی RISC ترجیح داده می‌شوند.



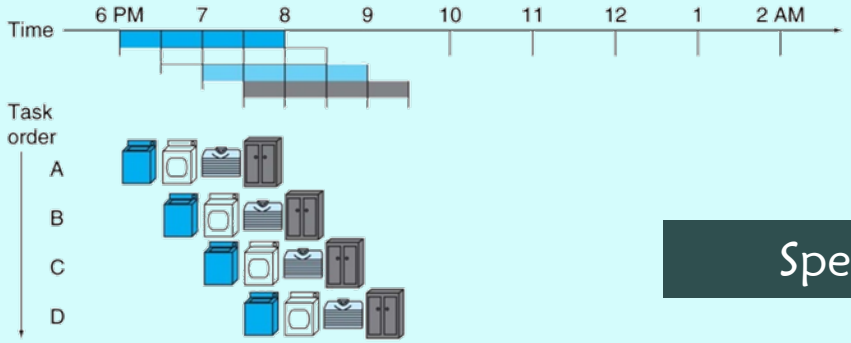
مروری بر خط لوله

- در یک سیستم خط لوله، اجرای چندین دستورالعمل دارای همپوشانی است.
- پایه‌ی خط لوله شبیه خط تولید کارخانه‌هاست.
- تقریبا در تمامی پردازنده‌های موجود از این تکنیک استفاده می‌شود.

مثالی از خط لوله (در رختشویخانه)

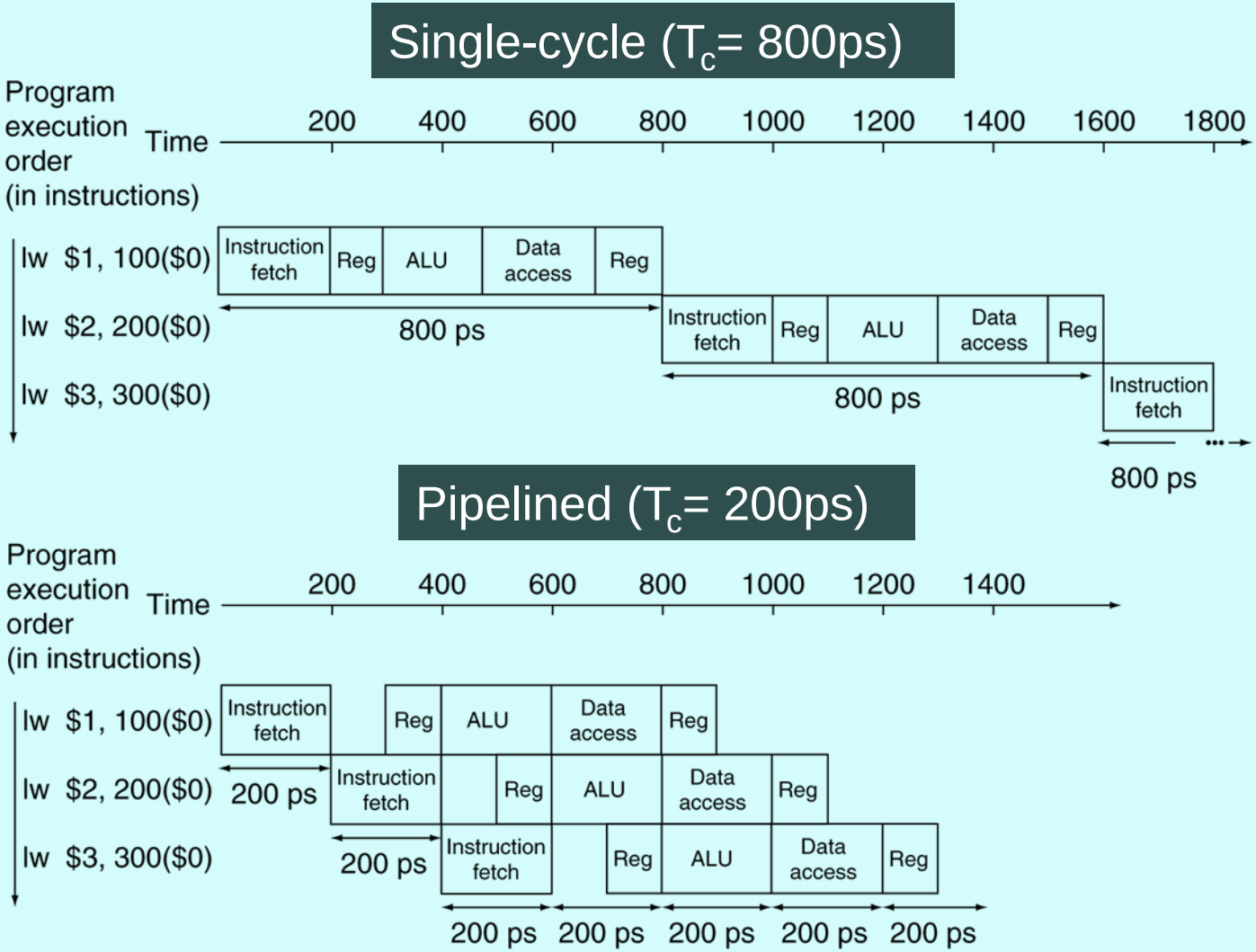


- در صورتی‌که کارها را با همپوشانی انجام دهیم، کارایی افزایش چشمگیری خواهد داشت



Speedup= 8/3.5 = 2.3

کارایی خط لوله (ادامه...)

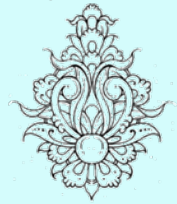


کارایی خط لوله (ادامه...)

- اگر تمامی مراحل متعادل (balanced) باشند؛ تمام مراحل زمان یکسانی صرف کنند

$$\text{Time between instructions}_{\text{pipelined}} = \frac{\text{Time between instructions}_{\text{nonpipelined}}}{\text{Number of stages}}$$

- در صورتی که خط لوله پر باشد، کارایی با تعداد گام‌ها خواهد بود؛ با یک خط لوله‌ی پنج مرحله‌ای سرعت پنج برابر می‌شود



که عشق آسان نمود اول، ولی افتاد مشکل‌ها

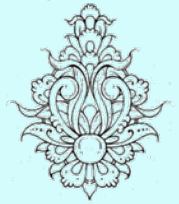
کارایی خط لوله (ادامه...)

- زمانی که هر واحد خط لوله نیاز دارد، یکسان نیست
- افزون بر این، استفاده از خط لوله به سیستم مقداری سربار هم تحمیل خواهد کرد.
- در مثال قبلی، زمان اجرای سه دستور به 1400ps رسید. اگر تعداد دستورات را ۱۰۰۰۰۰۳ در نظر بگیریم، افزایش سرعت تقریبا چهار برابر می‌شود.

1,000,000X800ps+2400

$$\frac{800002400\text{ps}}{200001400\text{ps}} \approx \frac{800\text{ps}}{200\text{ps}} \approx 4.00$$

1,000,000X200ps+1400ps

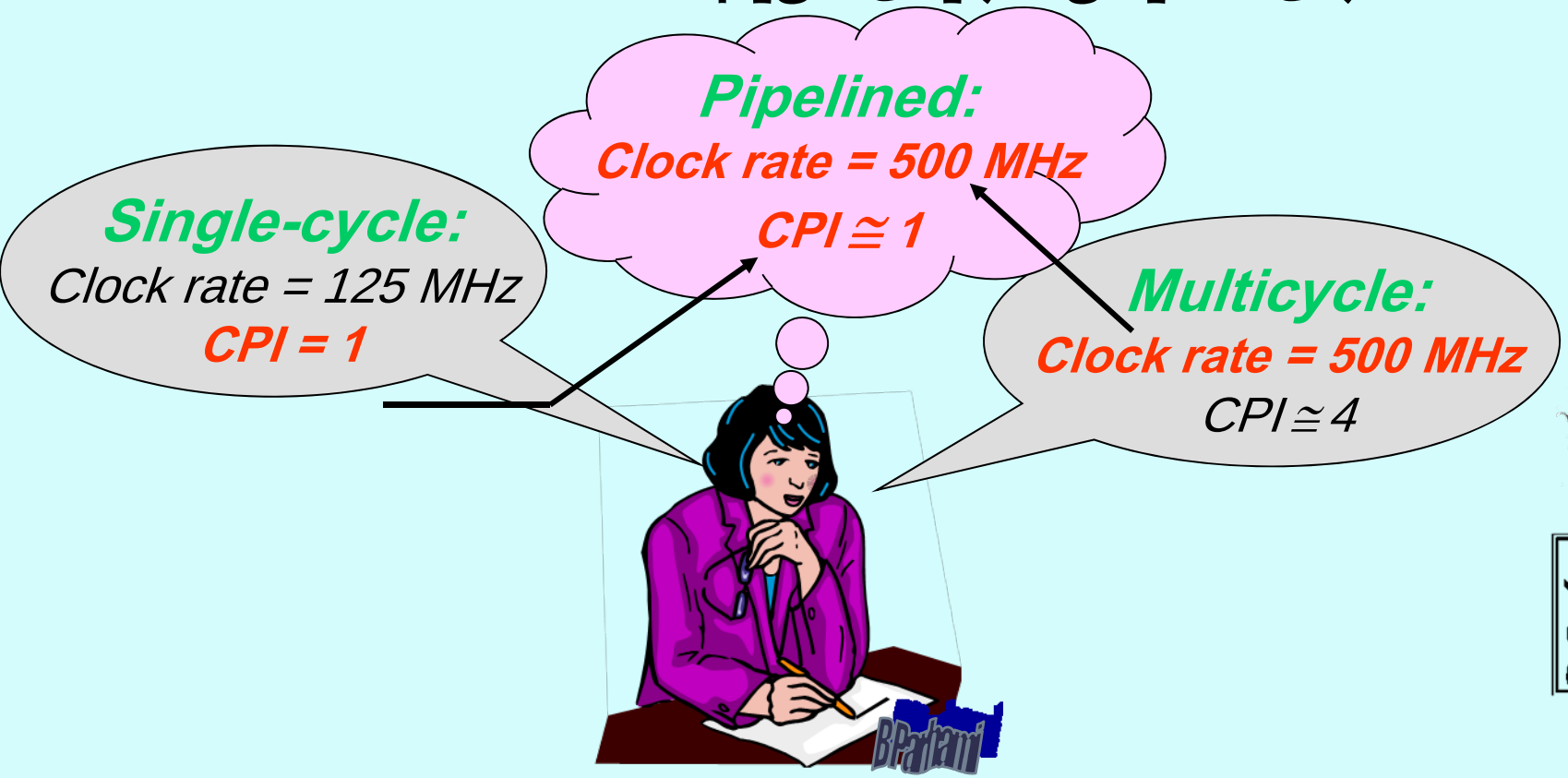


کارایی خط لوله (ادامه...)

- با استفاده از خط لوله،
  - زمان اجرای یک دستورالعمل افزایش می‌یابد.
  - توان عملیاتی افزایش می‌یابد.

Latency

throughput



معماری کامپیوتر (۱۳۹۱-۱۱-۱۳)

جلسه‌ی پانزدهم

دانشگاه شهید بهشتی

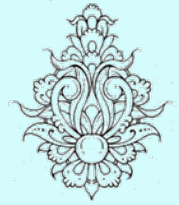
دانشکده‌ی مهندسی برق و کامپیوتر

بهار ۱۳۹۱

احمد محمودی ازناوه

فهرست مطالب

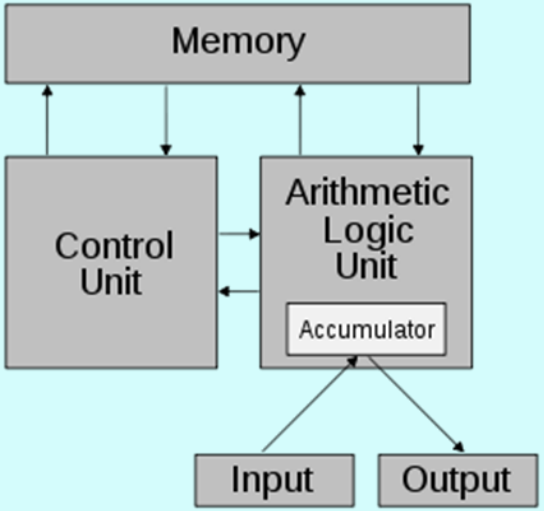
- مروری بر جلسه‌ی پیش
  - مراحل اجرای دستورالعمل‌ها
  - اجرای دستور در چند سیکل
- خط لوله
- مخاطرات خط لوله
  - انواع مخاطرات



معماری Von Neumann

- در این معماری دستورالعمل‌ها و داده‌ها در یک حافظه ذخیره می‌شوند.
- بدین‌ترتیب امکان خواندن همزمان داده و دستورالعمل از حافظه وجود ندارد، چنین مسأله‌ای به تنگنای معماری Von Neumann شهرت دارد.

Von Neumann bottleneck



معماری Harvard

- در معماری Harvard، برای داده‌ها و دستوالعمل‌ها، حافظه و گذرگاه‌داده‌ی جداگانه‌ای در نظر گرفته شده است. این نام در پی ساخت کامپیوتر Harvard Mark I که از حافظه‌های جداگانه برای داده‌ها و دستورالعمل‌ها استفاده می‌کرد، به این معماری اطلاق شده است.
- در اغلب کامپیوترهای امروزه از معماری تعدیل شده‌ی هاروارد (Modified Harvard Architecture) استفاده می‌شود.
- در این شیوه حافظه‌ی نهان مربوط به دستوالعمل و داده‌ها جداگانه هستند، به نوعی می‌توان این شیوه را ترکیبی از دو نوع معماری فوق دانست، چنین شیوه‌ای در پردازنده‌های x86، ARM، PowePc و MIPS مورد استفاده قرار می‌گیرد.

اجرای دستورات به صورت سری

Cycles \ Stages	S1	S2	S3	S4	S5	S6
1	I-1					
2		I-1				
3			I-1			
4				I-1		
5					I-1	
6						I-1
7	I-2					
8		I-2				
9			I-2			
10				I-2		
11					I-2	
12						I-2

- تا پیش از پردازنده‌های ۸۰۴۸۶، اجرای دستورات به صورت ترتیبی در شش مرحله صورت می‌پذیرفت. در پردازنده‌های ۸۰۴۸۶ این شش مرحله به صورت خط لوله در آمد.

خط لوله (pipeline)

- «خط لوله یک شگرد پیاده‌سازی است که در آن چندین دستورالعمل به طور هم‌پوشان (overlapped) به اجرا در می‌آید. در پردازنده‌های خانواده‌ی X86 نخستین بار در ۸۰۴۸۶ از خط لوله استفاده شد.

Stages

Cycles	S1	S2	S3	S4	S5	S6
1	I-1					
2	I-2	I-1				
3		I-2	I-1			
4			I-2	I-1		
5				I-2	I-1	
6					I-2	I-1
7						I-2

در یک خط لوله‌ی k مرحله‌ای، برای انجام n دستور چند سیکل ساعت زمان لازم است؟

k + (n – 1)

خط لوله (ادامه...)

- اگر زمانی که گام‌های مختلف خط لوله نیاز دارند یکسان نباشد، «ظرفیت گذردهی» چه تفاوتی خواهد کرد؟ فرض کنید گام چهارم به دو سیکل نیاز داشته باشد؟

Throughput

Stages

exe

Cycles

	S1	S2	S3	S4	S5	S6
1	I-1					
2	I-2	I-1				
3	I-3	I-2	I-1			
4		I-3	I-2	I-1		
5			I-3	I-1		
6				I-2	I-1	
7				I-2		I-1
8				I-3	I-2	
9				I-3		I-2
10					I-3	
11						I-3

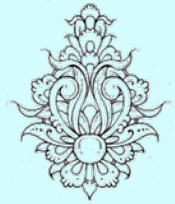
در چنین حالی، اجرای n دستور چند سیکل زمان لازم است؟

$k + (2n - 1)$

طراحی ISA برای خط لوله

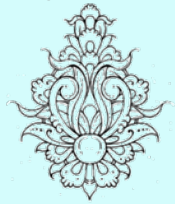
•تمام دستورات تمام دستورات MIPS طول یکسانی دارند. مرحله‌ی واکشی به سادگی انجام می‌شود.

•قالب دستورها مشابه هستند. بدین‌ترتیب کدگشایی ساده‌تر خواهد بود. کدگشایی دستور و خواندن محتوای ثبات در یک گام صورت می‌پذیرد. این که ثبات منبع جای مشخصی دارد، پیش از تمام شدن کدگشایی دستور محتوای ثبات خوانده می‌شود. وگرنه، گام‌های خطلوله به شش گام می‌رسید.



طراحی ISA برای خط لوله (ادامه...)

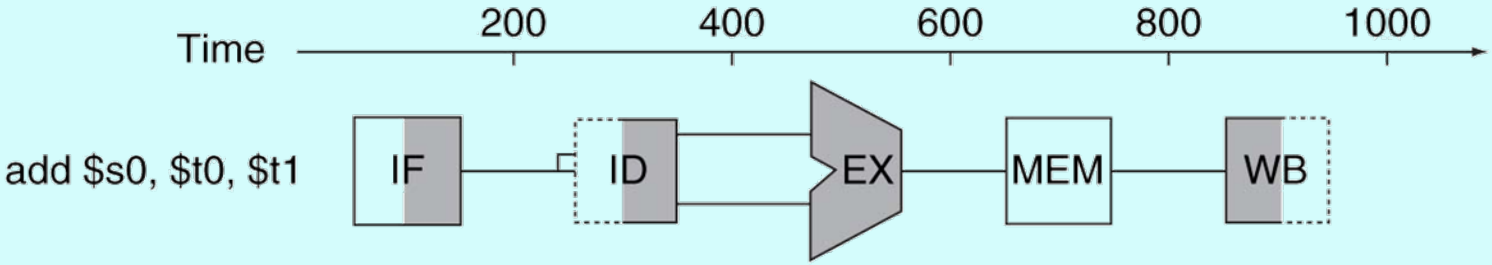
- در مجموعه دستورات MIPS، دسترسی به حافظه فقط در دستورهای lw/sw امکان‌پذیر می‌باشد. در صورتی که در دستورهای محاسباتی استفاده از عملوند حافظه مجاز بود، گام سوم و چهارم به گام‌های یافتن آدرس، خواندن حافظه و اجرای دستور گسترش می‌یافت.



مخاطره / مانع

به نقل از ترجمه‌ی آقای دکتر ملکیان

- گاهی اوقات در مسیر خط لوله شرایطی پیش می‌آید که دستورالعمل بعدی را نمی‌توان در سیکل ساعت بعدی اجرا کرد. به چنین رخدادهایی اصطلاحاً «خطرات ناخواسته» یا به عبارت بهتر «مخاطرات و موانع سدّ راه در خط لوله» گفته می‌شود و با سه نوع مختلف از آن مواجه خواهیم بود.



این شکل مراحل مختلف یک خط لوله، برای دستور add را نشان می‌دهد، شکل‌هایی که سایه نخورده‌اند، در این دستور مورد استفاده قرار نمی‌گیرند، بخش‌های که در سمت راست (چپ) سایه خورده‌اند، بیانگر خواندن (نوشتن) داده از (در) حافظه می‌باشد

مخاطرات خط لوله!!

Structural Hazard

– مخاطرات ساختاری

- یکی از منابع مورد نیاز مشغول است

– مخاطرات داده‌ای

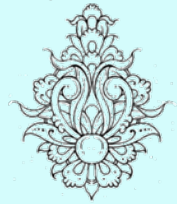
- به داده‌ای نیاز است که توسط دستور قبلی آماده نشده است

Data Hazard

– مخاطرات کنترلی

- چنان‌چه دستورات کنترلی بخواهند بر اساس اجرای دستور قبلی تصمیم‌گیری کنند.

Control Hazard



مخاطرات ساختاری

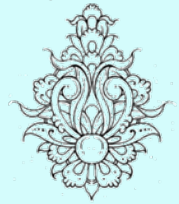
- اولین مانع در خط لوله به **مخاطرات ساختاری** شهرت دارد و بدین معناست که سخت‌افزار فی‌نفسه قادر به پشتیبانی ترکیبی خاص از دستورالعمل‌هایی که می‌خواهیم در یک سیکل واحد آنها را اجرا کنیم نیست.
- تعارض در هنگام استفاده از منابع مشترک
  - در MIPS با یک حافظه برای دستورالعمل‌ها و داده‌ها
    - دستورات خواندن و نوشتن به حافظه احتیاج دارند

stall

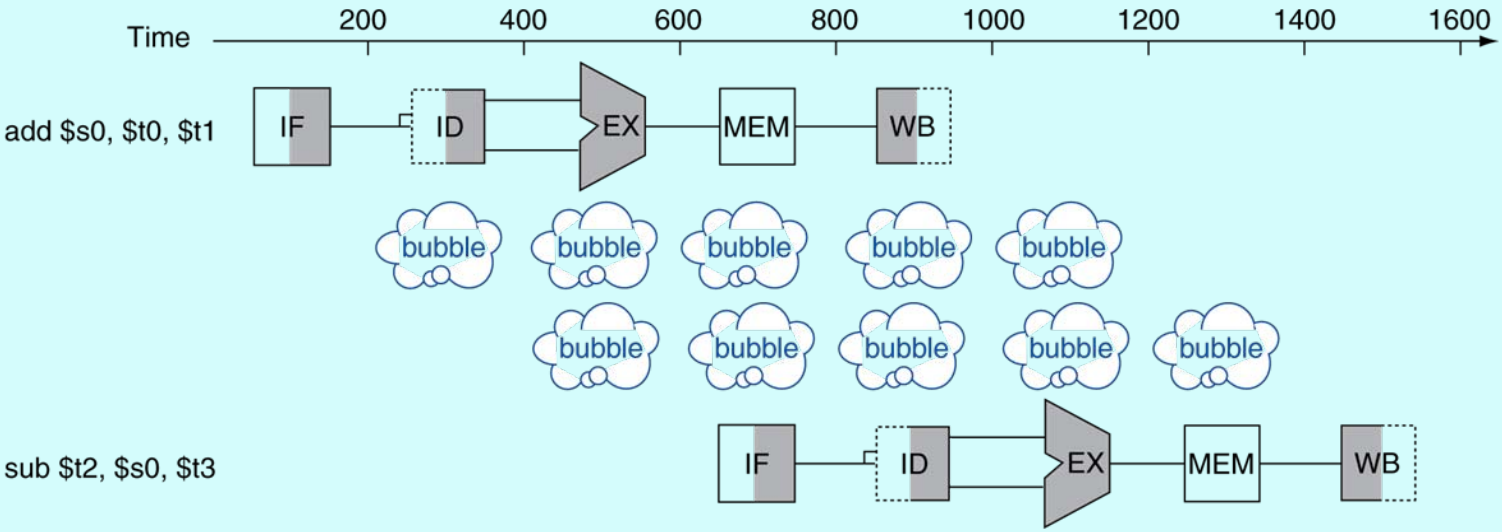
- در این صورت برای واکشی دستورات بعدی، تعلیق می‌شود.
  - در چنین حالاتی به جای دستورات بعدی حباب وارد خط لوله می‌شود.

Bubble

- خط لوله‌ی داده‌گذر (datapath) به دو حافظه‌ی داده و دستور نیاز دارد.



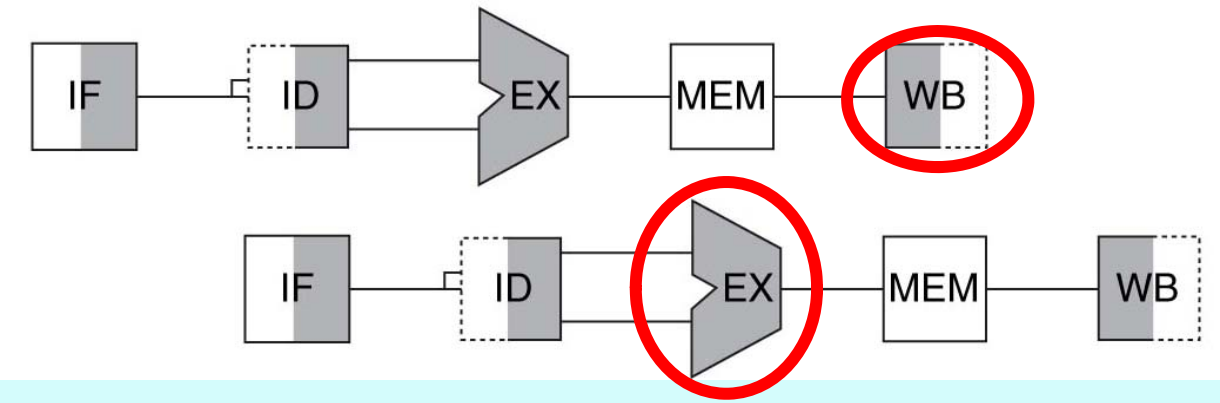
تعلیق



- این شکل یکی از مفاهیم بسیار با اهمیت در ایجاد خط لوله را به تصویر کشیده است که اگرچه عنوان رسمی «تعلیق خط لوله» دارد ولیکن اغلب اوقات از این پدیده با نام ساده‌تر «حباب» یاد می‌شود. در ادامه‌ی بحث به موانع منجر به «تعلیق» در بخش‌های دیگری از خط لوله خواهیم پرداخت.

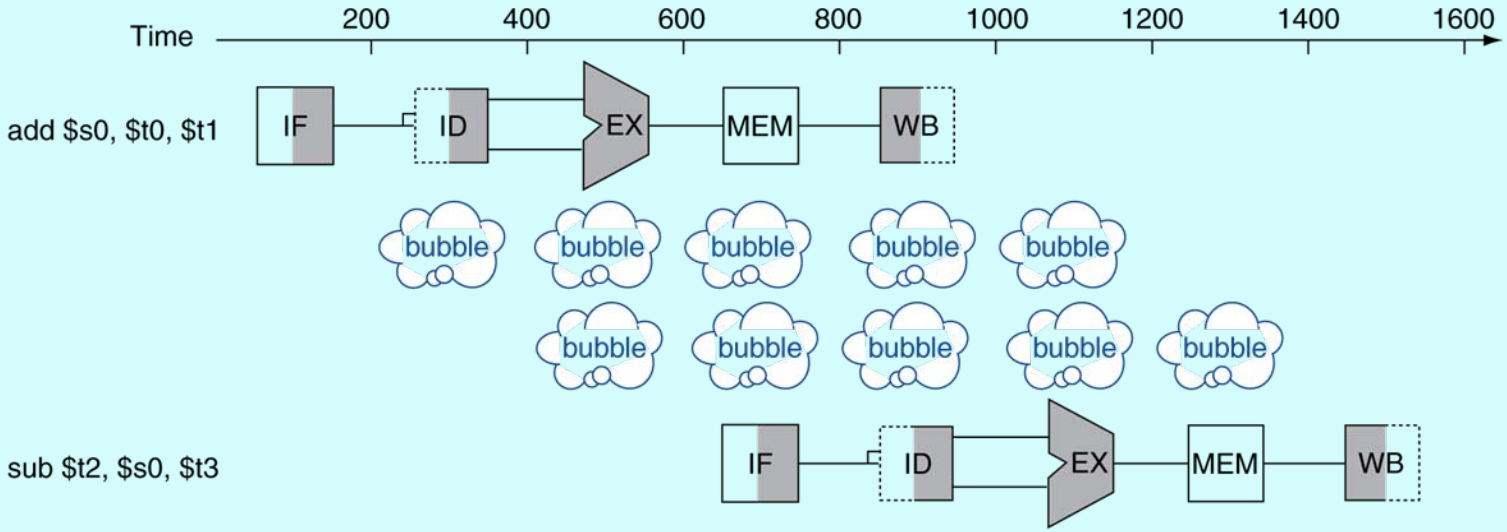
- مخاطرات و موانع ناشی از **داده** موقعی رخ خواهد داد که خط لوله باید در انتظار تکمیل یکی از مراحل قبلی از حرکت باز داشته شود

```
add $s0, $t0, $t1
sub $t2, $s0, $t3
```



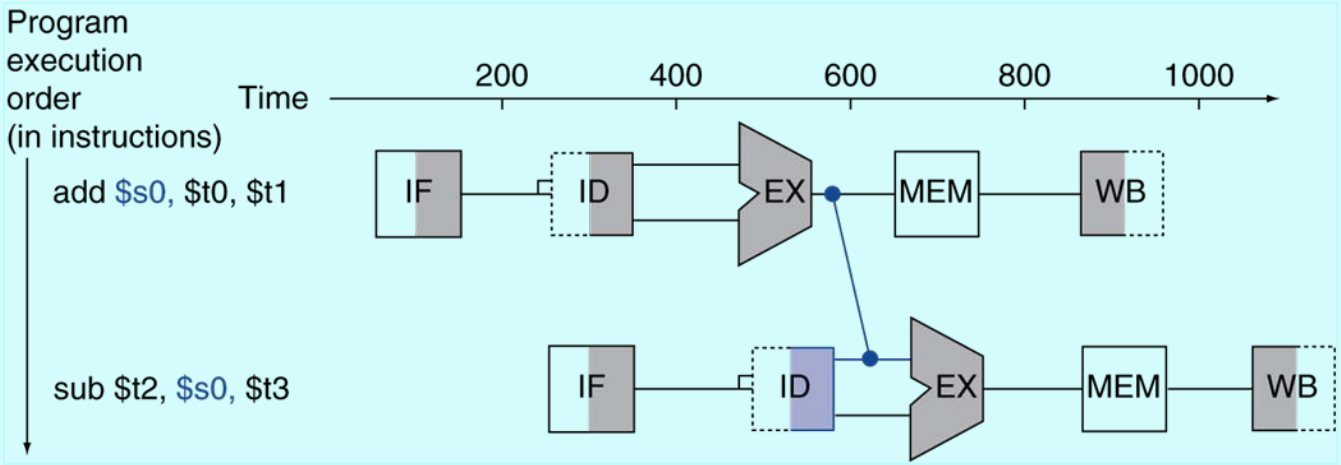
- یک دستورالعمل به داده‌ای نیاز دارد، که در دستور قبلی مشغول آماده کردن آن است

```
add $s0, $t0, $t1
sub $t2, $s0, $t3
```



پیش فرستادن

- در حالاتی که مخاطره‌ی داده رخ می‌دهد، پس از انجام دستورالعمل و آماده شدن داده، نتیجه به دست آمده پیش از ذخیره در ثبات، در دستور بعدی استفاده می‌شود.
- چنین حالتی، نیاز به اتصالات بیشتری در داده‌گذر (datapath) دارد.



معماری کامپیوتر (۱۳۹۱-۱۱-۱۳)

جلسه‌ی شانزدهم

دانشگاه شهید بهشتی

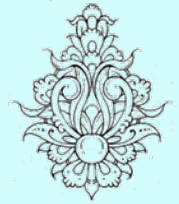
دانشکده‌ی مهندسی برق و کامپیوتر

بهار ۱۳۹۱

احمد محمودی ازناوه

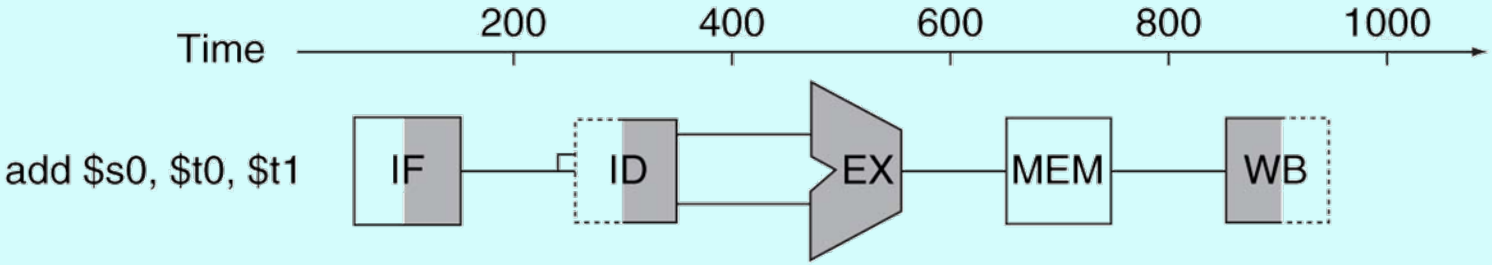
فهرست مطالب

- مروری بر جلسه‌ی پیش
  - مخاطرات خط لوله
  - تعلیق و حباب
- مخاطرات کنترلی
- مسیر گذار داده



مخاطره / مانع

- گاهی اوقات در مسیر خط لوله شرایطی پیش می‌آید که دستورالعمل بعدی را نمی‌توان در سیکل ساعت بعدی اجرا کرد. به چنین رخدادهایی اصطلاحاً «خطرات ناخواسته» یا به عبارت بهتر «مخاطرات و موانع سدّ راه در خط لوله» گفته می‌شود و با سه نوع مختلف از آن مواجه خواهیم بود.



این شکل مراحل مختلف یک خط لوله، برای دستور add را نشان می‌دهد، شکل‌هایی که سایه نخورده‌اند، در این دستور مورد استفاده قرار نمی‌گیرند، بخش‌های که در سمت راست (چپ) سایه خورده‌اند، بیانگر خواندن (نوشتن) داده از (در) حافظه می‌باشد

مخاطرات خط لوله!!

Structural Hazard

– مخاطرات ساختاری

- یکی از منابع مورد نیاز مشغول است

– مخاطرات داده‌ای

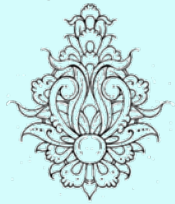
- به داده‌ای نیاز است که توسط دستور قبلی آماده نشده است

Data Hazard

– مخاطرات کنترلی

- چنان‌چه دستورات کنترلی بخواهند بر اساس اجرای دستور قبلی تصمیم‌گیری کنند.

Control Hazard



مخاطرات ساختاری

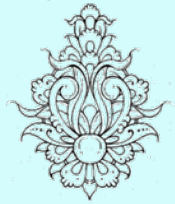
- اولین مانع در خط لوله به مخاطرات ساختاری شهرت دارد و بدین معناست که سخت‌افزار فی‌نفسه قادر به پشتیبانی ترکیبی خاص از دستورالعمل‌هایی که می‌خواهیم در یک سیکل واحد آنها را اجرا کنیم نیست.
- تعارض در هنگام استفاده از منابع مشترک
  - در MIPS با یک حافظه برای دستورالعمل‌ها و داده‌ها
    - دستورات خواندن و نوشتن به حافظه احتیاج دارند

stall

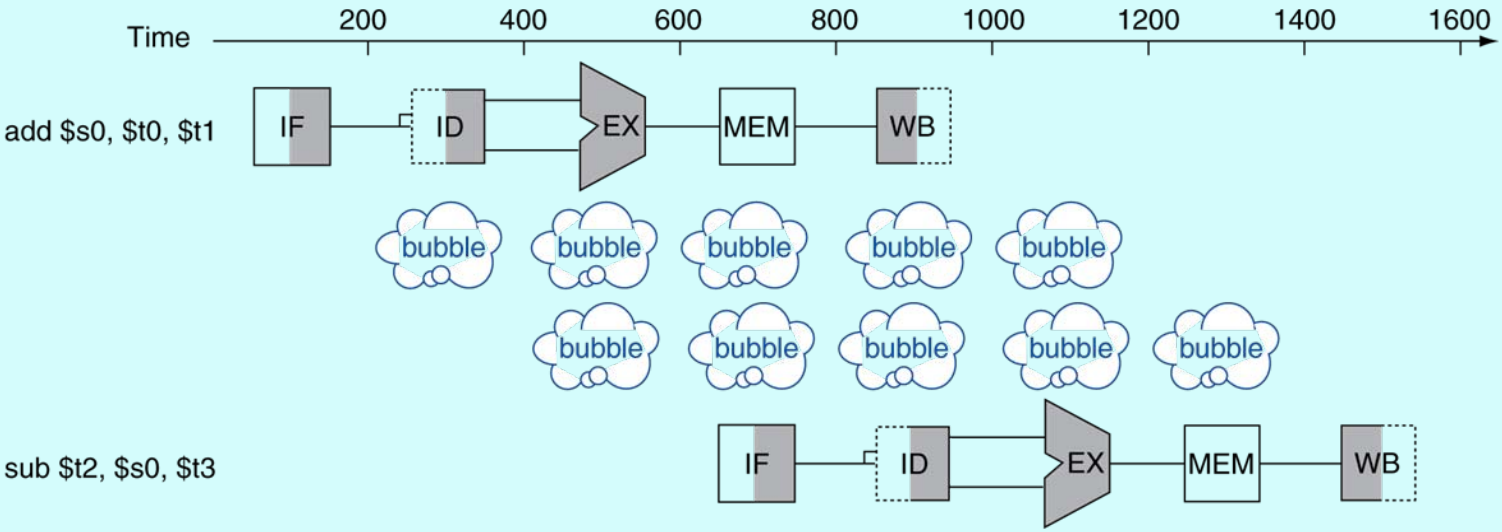
- در این صورت برای واکشی دستورات بعدی، تعلیق می‌شود.
  - در چنین حالاتی به جای دستورات بعدی حباب وارد خط لوله می‌شود.

Bubble

- خط لوله‌ی داده‌گذر (datapath) به دو حافظه‌ی داده و دستور نیاز دارد



تعلیق

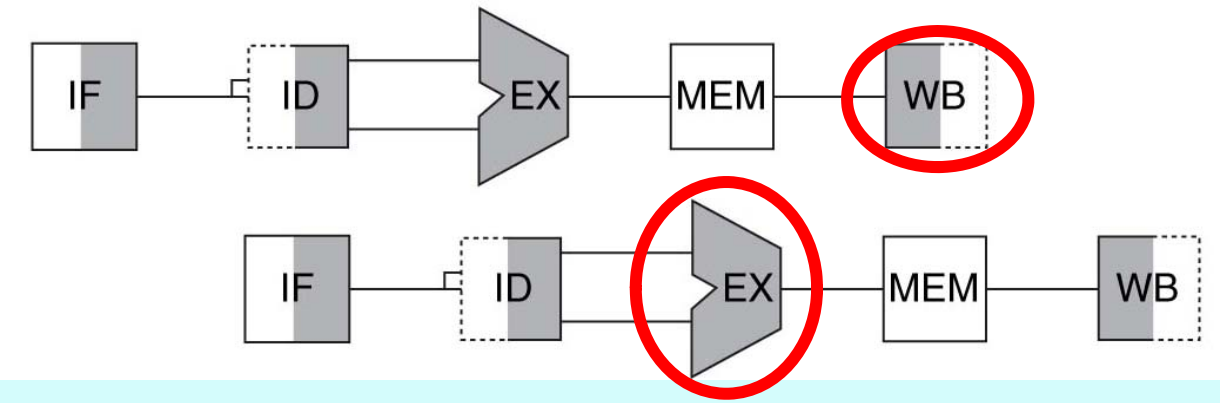


- این شکل یکی از مفاهیم بسیار با اهمیت در ایجاد خط لوله را به تصویر کشیده است که اگرچه عنوان رسمی «تعلیق خط لوله» دارد ولیکن اغلب اوقات از این پدیده با نام ساده‌تر «حباب» یاد می‌شود. در ادامه‌ی بحث به موانع منجر به «تعلیق» در بخش‌های دیگری از خط لوله خواهیم پرداخت.

مخاطرات داده‌ای

- مخاطرات و موانع ناشی از داده موقعی رخ خواهد داد که خط لوله باید در انتظار تکمیل یکی از مراحل قبلی از حرکت باز داشته شود.

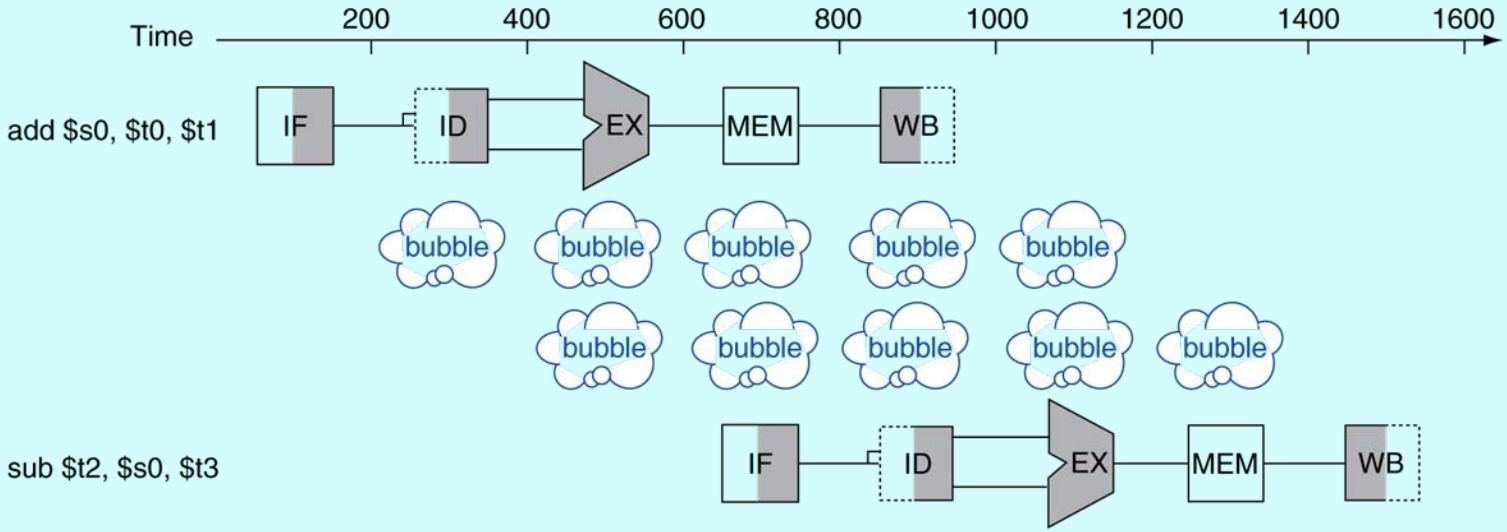
```
add $s0, $t0, $t1
sub $t2, $s0, $t3
```



مخاطرات داده‌ای

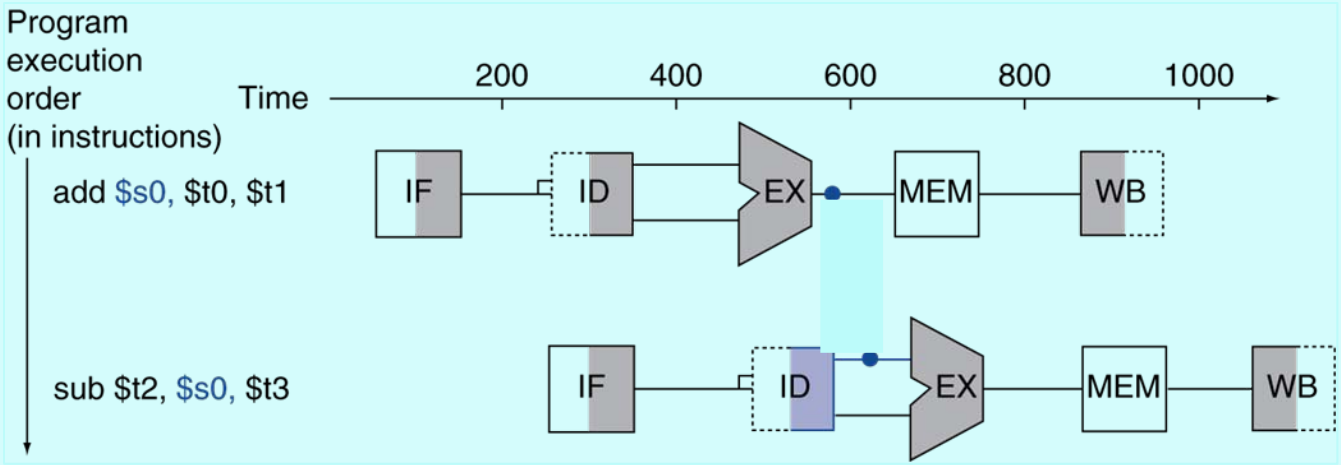
- یک دستورالعمل به داده‌ای نیاز دارد، که در دستور قبلی مشغول آماده کردن آن است

```
add $s0, $t0, $t1
sub $t2, $s0, $t3
```



شگرد پیش فرستادن (هدایت رو به جلو)

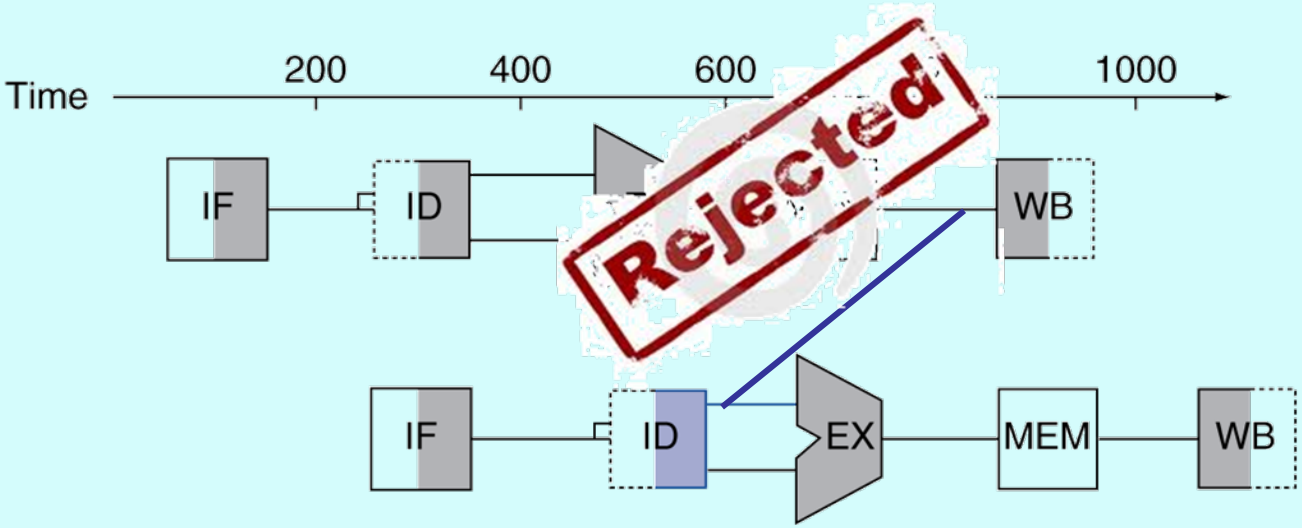
- در حالاتی که مخاطره‌ی داده رخ می‌دهد، پس از انجام دستورالعمل و آماده شدن داده، نتیجه به دست آمده پیش از ذخیره در ثبات، در دستور بعدی استفاده می‌شود.
- چنین حالتی، نیاز به اتصالات بیشتری در داده‌گذر (datapath) دارد.



استفاده از داده‌ی در حال بارگذاری

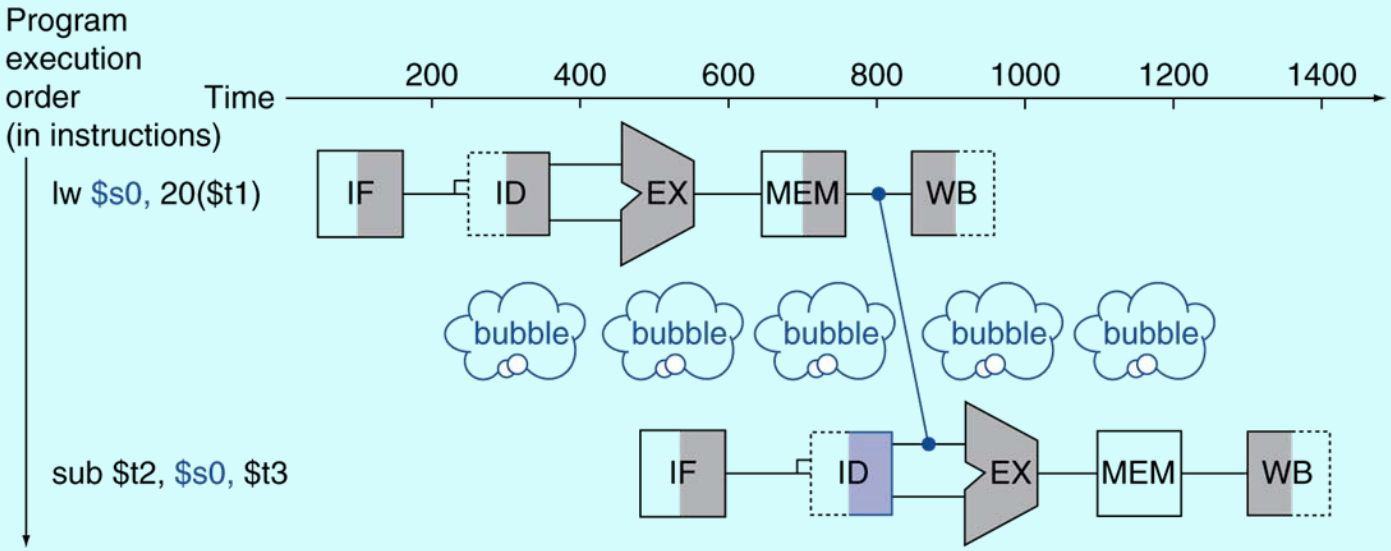
- نوع خاصی از مخاطره‌ی داده‌ای است که در آن داده‌ای در خواندن از حافظه است، هنوز برای استفاده در دستور بعدی آماده نیست.

```
lw  $s0, 20($t1)
sub $t2, $s0, $t3
```



استفاده از داده‌ی در حال بارگذاری (ادامه...)

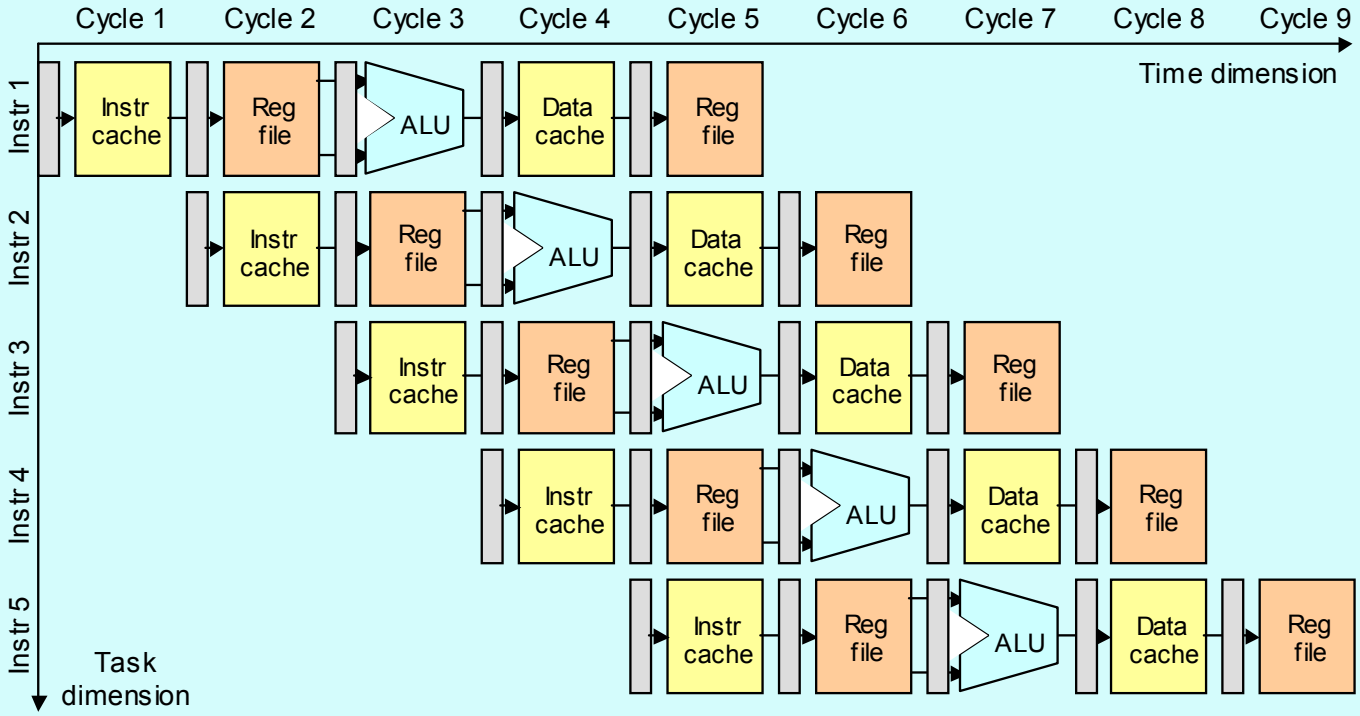
- با پیش‌فرستادن نمی‌توان از وقوع تعلیق اجتناب کرد.
- نمی‌توان در زمان سفر کرد !!



استفاده از داده‌ی در حال بارگذاری (ادامه...)

- در صورت عدم استفاده از پیش‌فرستادن چند حباب وارد خط لوله می‌شود؟

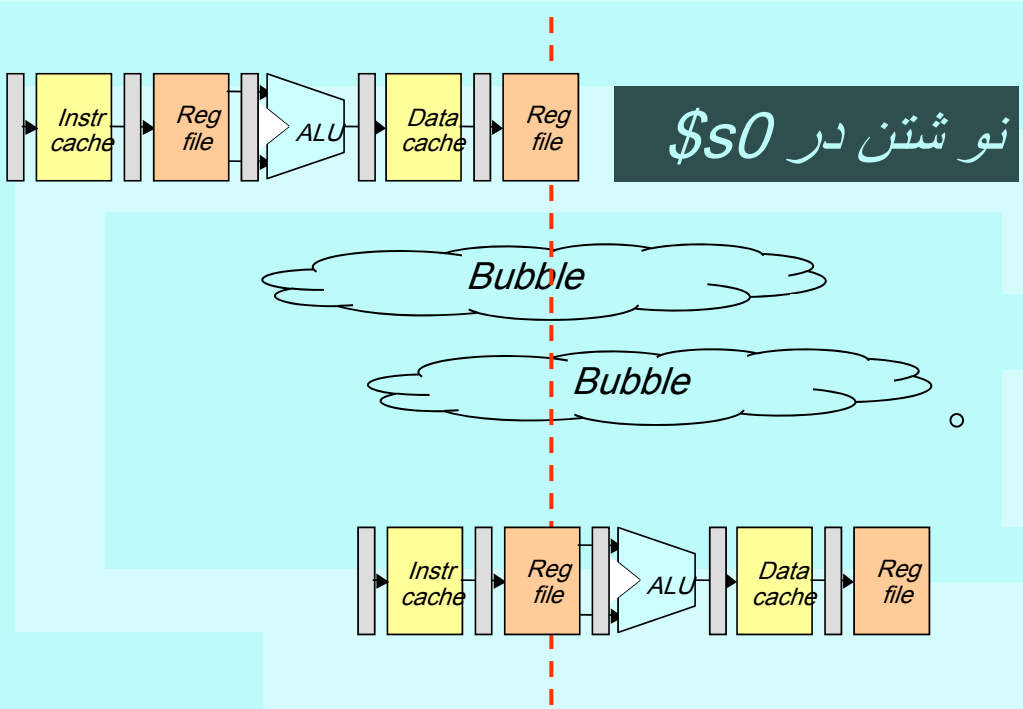
```
lw    $s0, 20($t1)
sub   $t2, $s0, $t3
```



استفاده از داده‌ی در حال بارگذاری (ادامه...)

- در صورت عدم استفاده از پیش‌فرستادن چند حباب وارد خط لوله می‌شود؟

```
lw    $s0, 20($t1)
sub   $t2, $s0, $t3
```



تغییر ترتیب برنامه

- با جابجا کردن کد می‌توان طول اجرای برنامه را کاهش داد.(از پیش‌فرستادن استفاده می‌شود.)

```
A = B + E;
C = B + F;
```

```
lw   $t1, 0($t0)
lw   $t2, 4($t0)
add  $t3, $t1, $t2
sw   $t3, 12($t0)
lw   $t4, 8($t0)
add  $t5, $t1, $t4
sw   $t5, 16($t0)
```

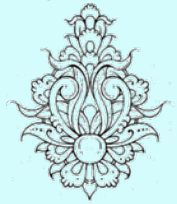
stall

stall

13 cycles

```
lw   $t1, 0($t0)
lw   $t2, 4($t0)
lw   $t4, 8($t0)
add  $t3, $t1, $t2
sw   $t3, 12($t0)
add  $t5, $t1, $t4
sw   $t5, 16($t0)
```

11 cycles



Control Hazard or Branch Hazard

مخاطره‌ی کنترلی

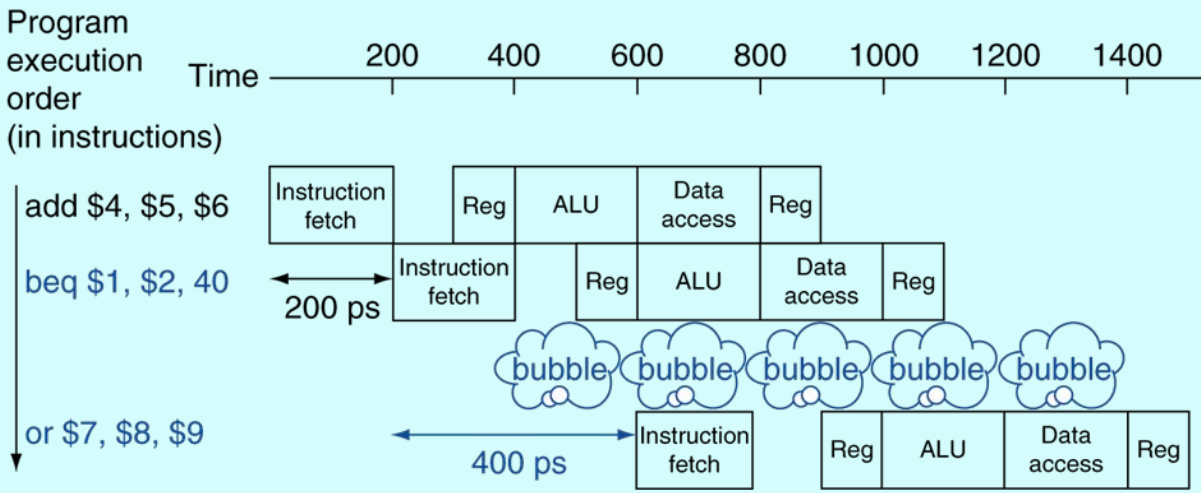
- در مواقعی که یک دستور پرش شرطی وجود دارد، ادامه‌ی دستورهایی که باید توسط پردازنده اجرا شوند وابسته به تصمیمی است که در این دستور گرفته می‌شود.

Stall on Branch

چه راه حلی پیشنهاد می‌دهید؟

- تعلیق؟؟

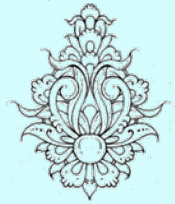
- در MIPS می‌توان با افزودن سخت‌افزار اضافی در گام ID، نتیجه‌ی شرط را زودتر به دست آورد.



مخاطره‌ی کنترلی (ادامه...)

- برای خط لوله‌های که دارای تعداد گام بالا هستند، دستیابی سریع‌تر به نتیجه‌ی به سادگی امکان‌پذیر نیست.
  - در چنین حالاتی زیان ناشی تعلیق، پذیرفتنی نیست.

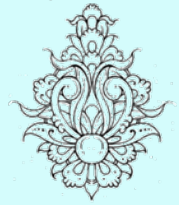
- در صورتی که از تعلیق برای رفع مشکل مخاطره‌ی کنترلی استفاده شود، تأثیر دستورات پرش شرطی بر CPI چه خواهد بود؟



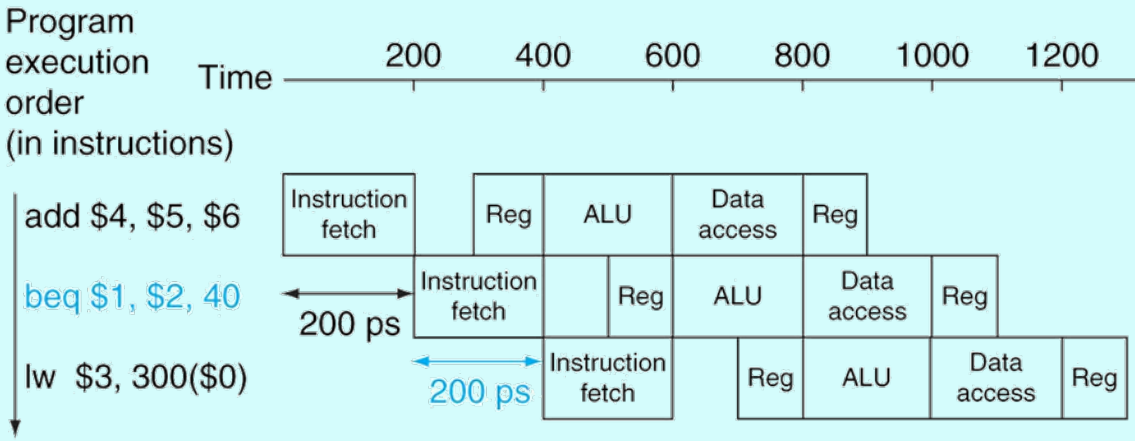
مخاطره‌ی کنترلی (ادامه...)

راه‌حل دیگر

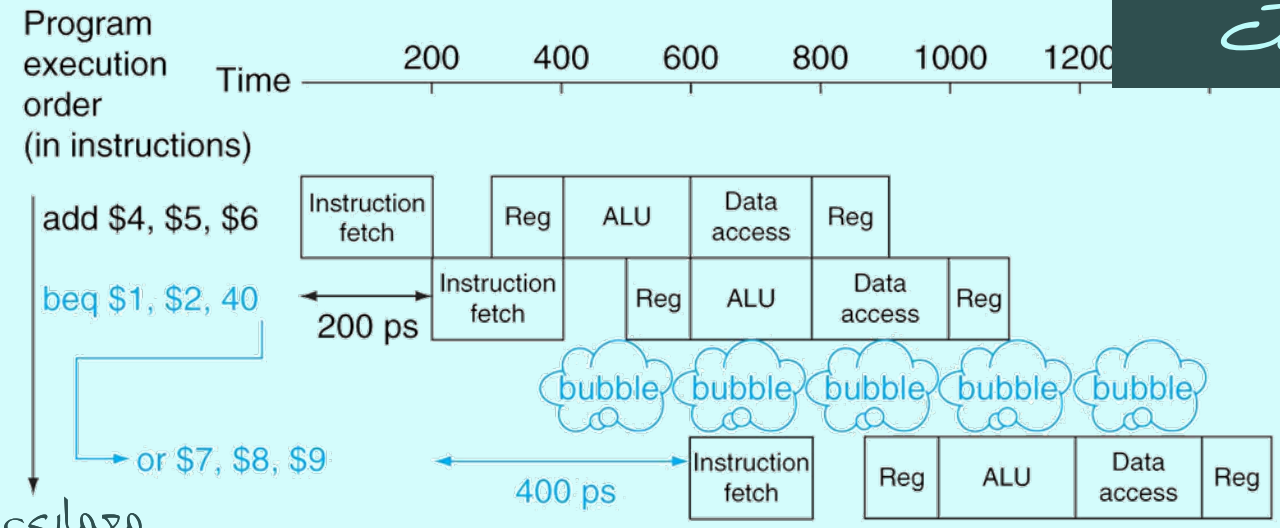
- استفاده از پیش‌بینی: یک راه ساده این است که همیشه فرض کنیم نتیجه شرط درست(نادرست) خواهد بود.
- در صورتی که پیش‌بینی ما درست باشد، خط لوله کار خود را به درستی انجام داده است.
- در غیر این صورت خط لوله دچار تعلیق می‌شود. ضمنا در چنین حالاتی باید مطمئن شویم که دستوراتی که به اشتباه وارد خط لوله شده‌اند، تأثیری از خود به جای نخواهند گذاشت



مخاطره‌ی کنترلی (ادامه...)



پیش‌بینی درست



پیش‌بینی نادرست

مخاطره‌ی کنترلی (ادامه...)

- یک راه پیچیده‌تر، استفاده از پیش‌بینی واقع‌گرایانه‌تر است.

Predict backward branches taken

  - مثلا در حلقه‌ها پیش‌بینی می‌شود، که شرط برقرار است.
  - در پیش‌بینی پویا، سخت‌افزاری برای پیش‌بینی عملکرد شرط در نظر گرفته می‌شود.

Dynamic branch prediction

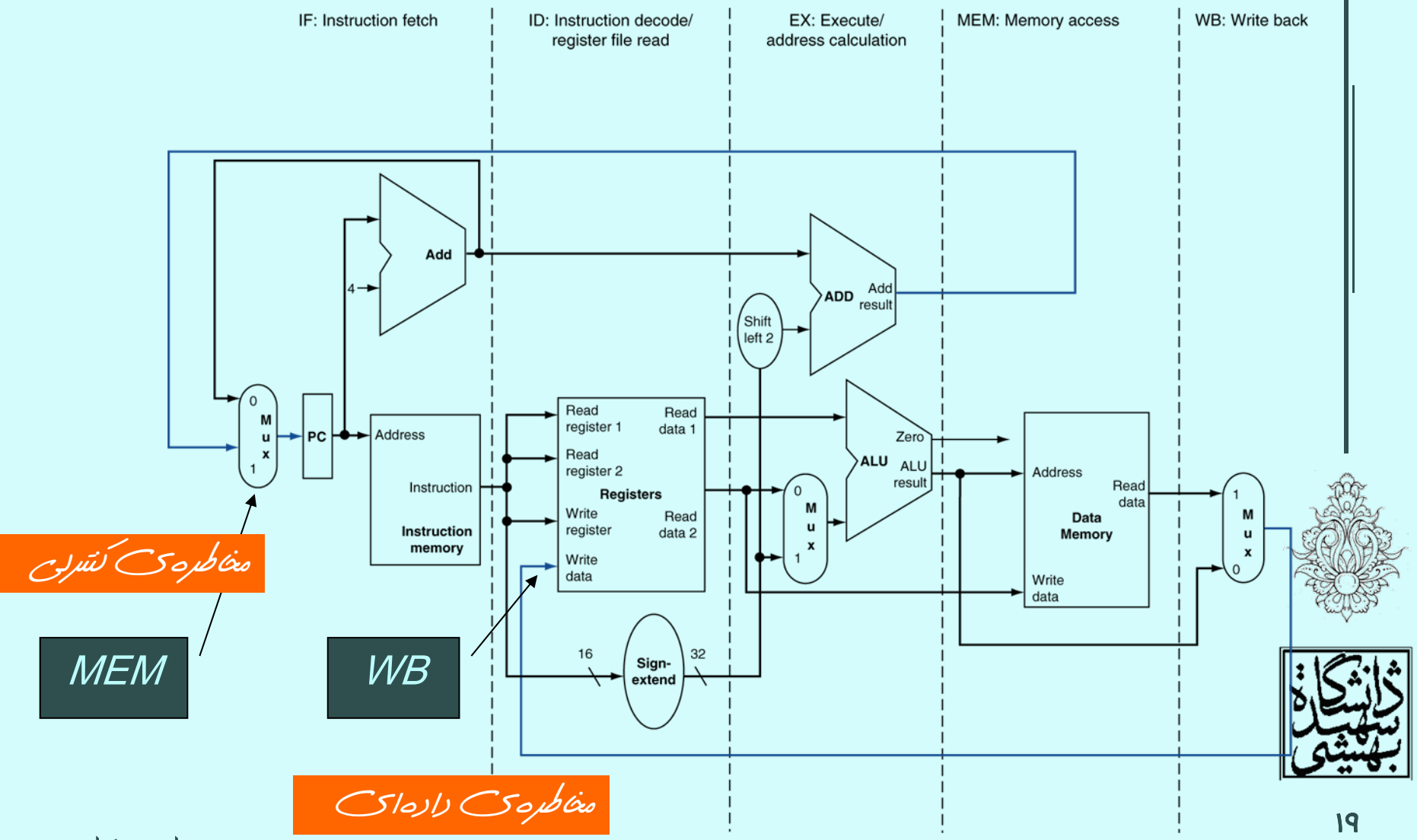
    - به عنوان مثال، می‌توان تاریخچه‌ای از رفتار شرط ذخیره نمود.

- راه سومی هم هست که در راه سومی هم هست که در MIPS مورد استفاده قرار می‌گیرد و آن این که اسمبلر دستوراتی را که در شرط مؤثر نیستند، به بعد از دستور پرش منتقل می‌کند.

Delayed decision

جریان داده از راست به چپ منجر به بروز مخاطره می شود

مسیر گذار داده‌ی مجهز به خطلوله



مخاطره‌ی کنترلی

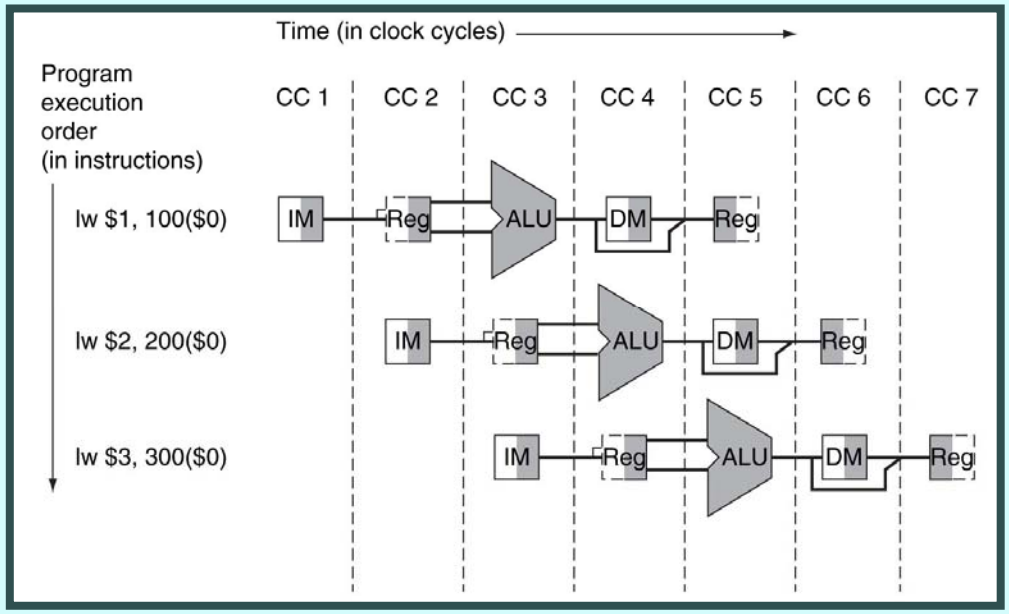
MEM

WB

مخاطره‌ی داده‌ای

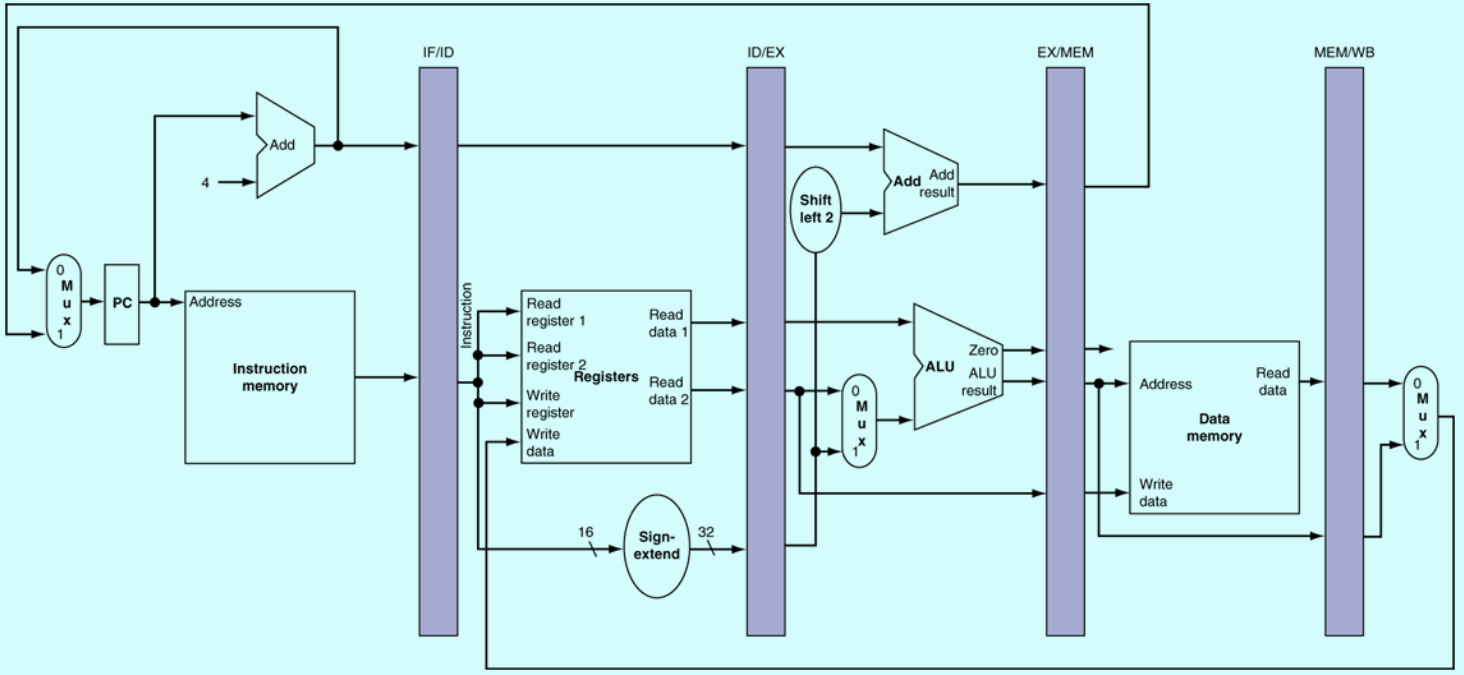
گذرداده‌ی مجهز به خطلوله (ادامه...)

- چنان‌چه ملاحظه شد، مخاطره‌ی خط لوله دو سرچشمه دارد، که در هر دو جریان داده از سمت راست به چپ می‌باشد:
  - مرحله‌ی WB (مخاطره‌ی داده‌ای)
  - انتخاب آدرس بعدی (مخاطره‌ی کنترلی)

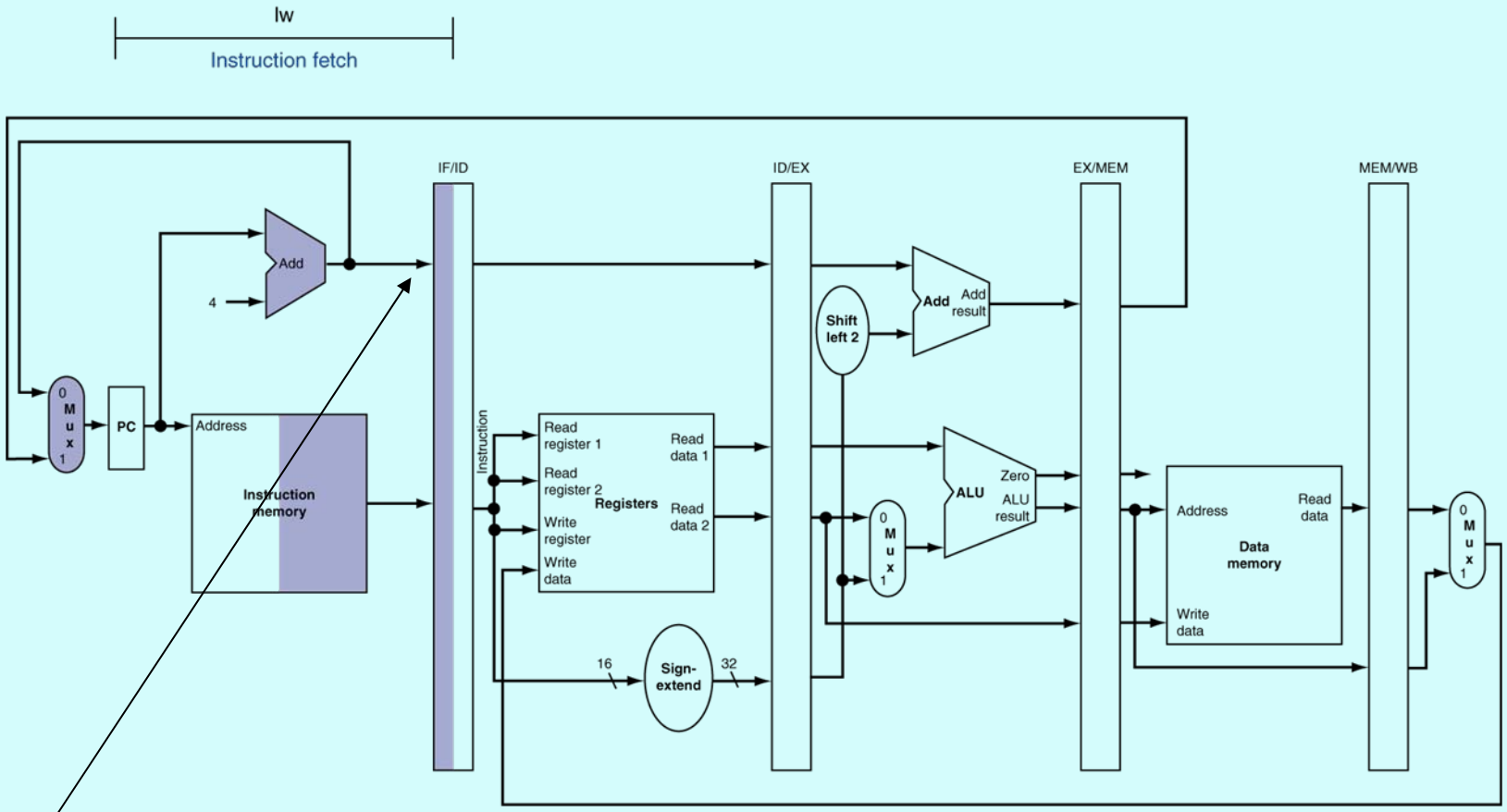


ثبات‌های خط لوله

- برای این که داده‌ی تولید شده در هر گام حفظ شود، می‌باید بین گام‌های مختلف خط لوله از ثبات استفاده شود.

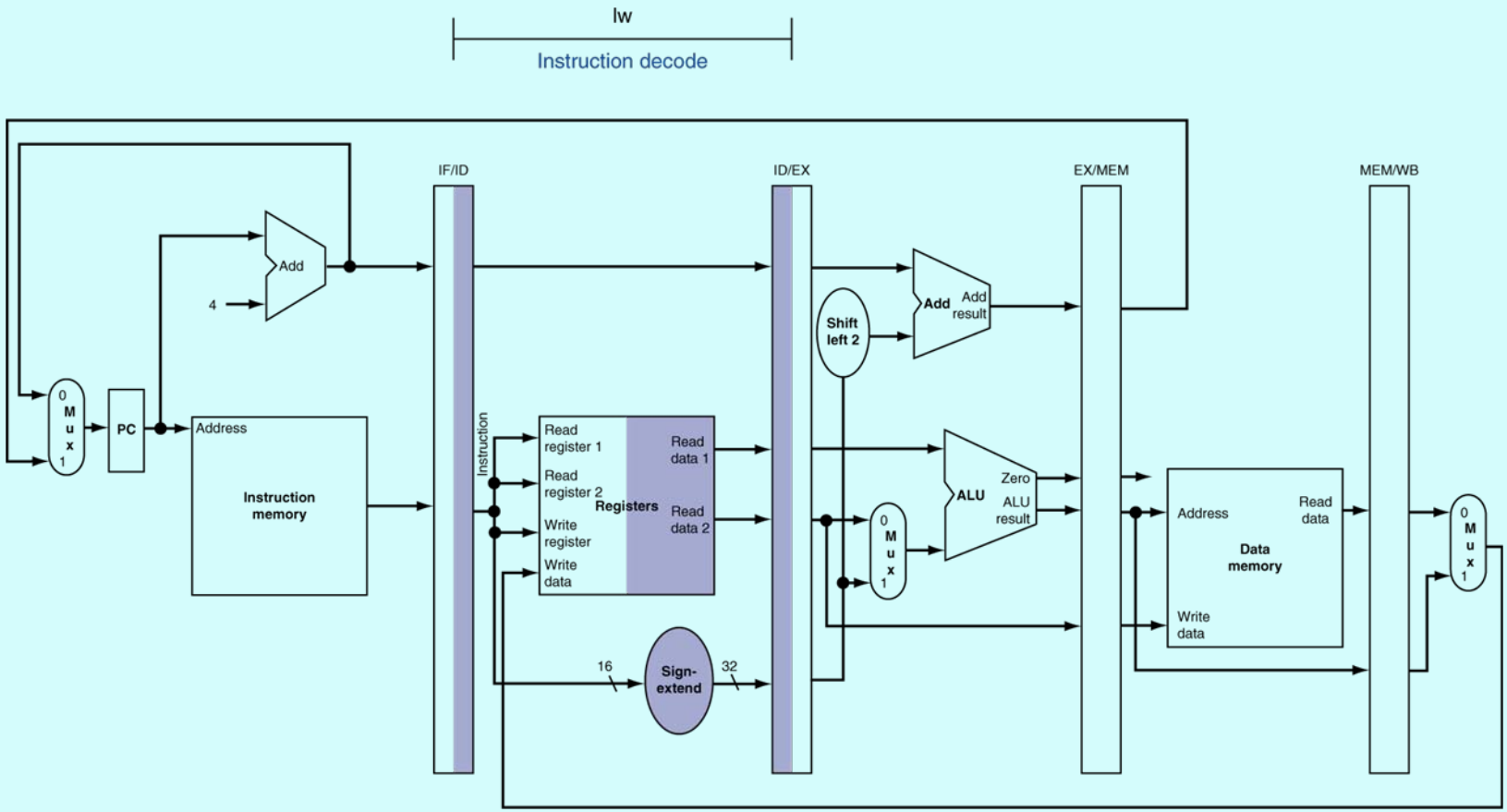


دستورات نوشتن و خواندن حافظه IF

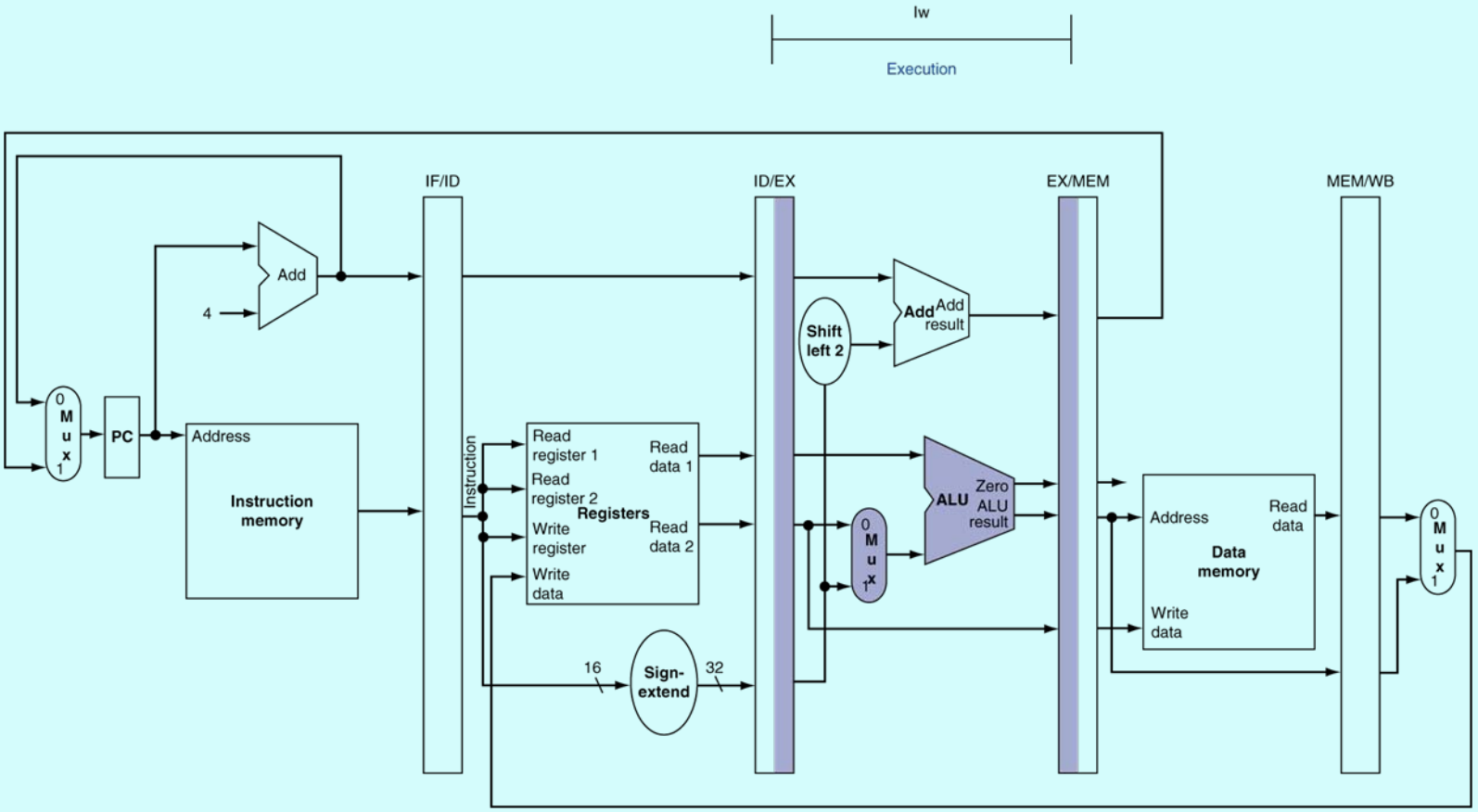


در صورتی که دستور پرش شرطی باشد، به این آدرس نیز خواهیم داشت

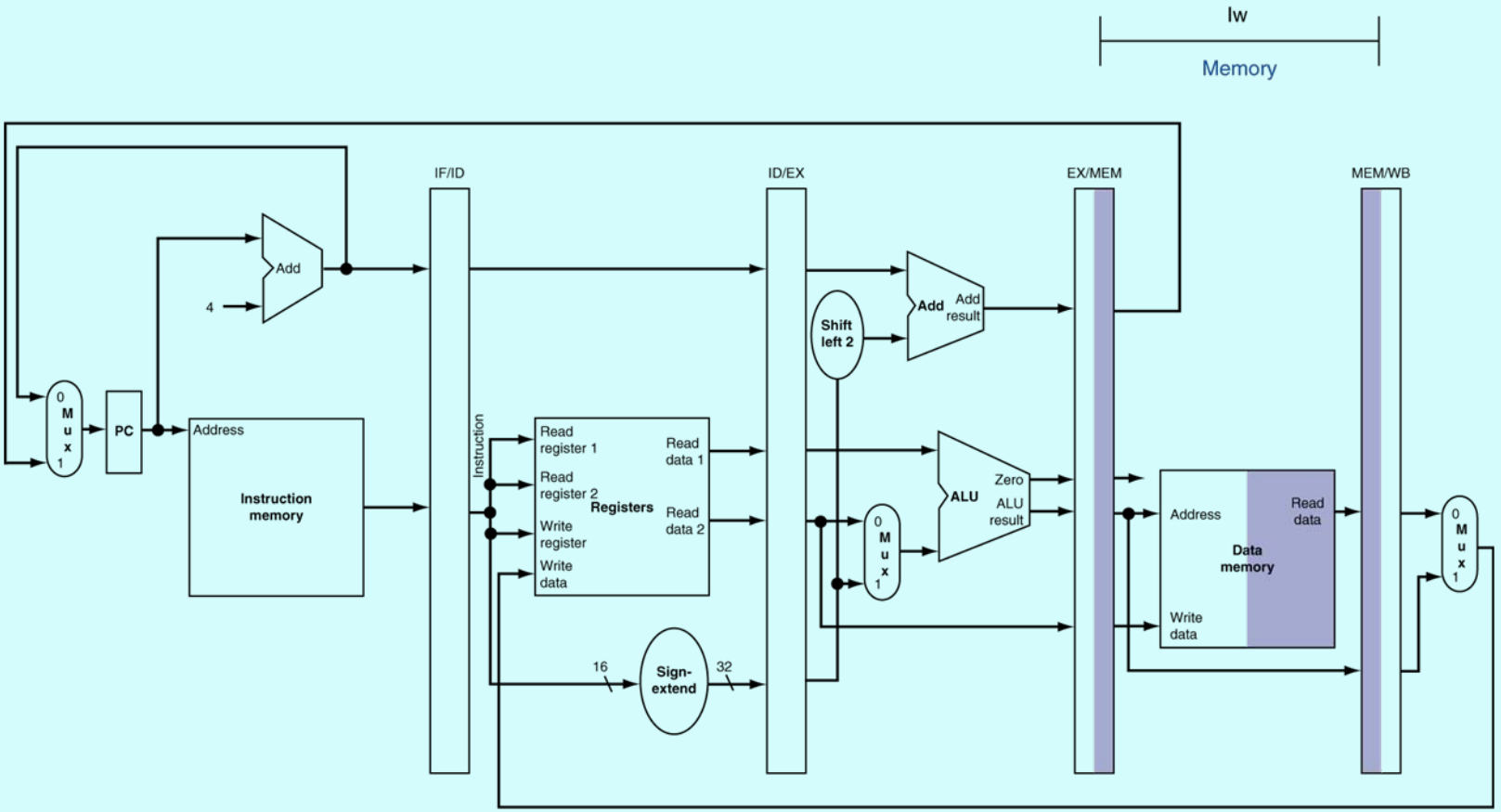
دستورات نوشتن و خواندن حافظه ID



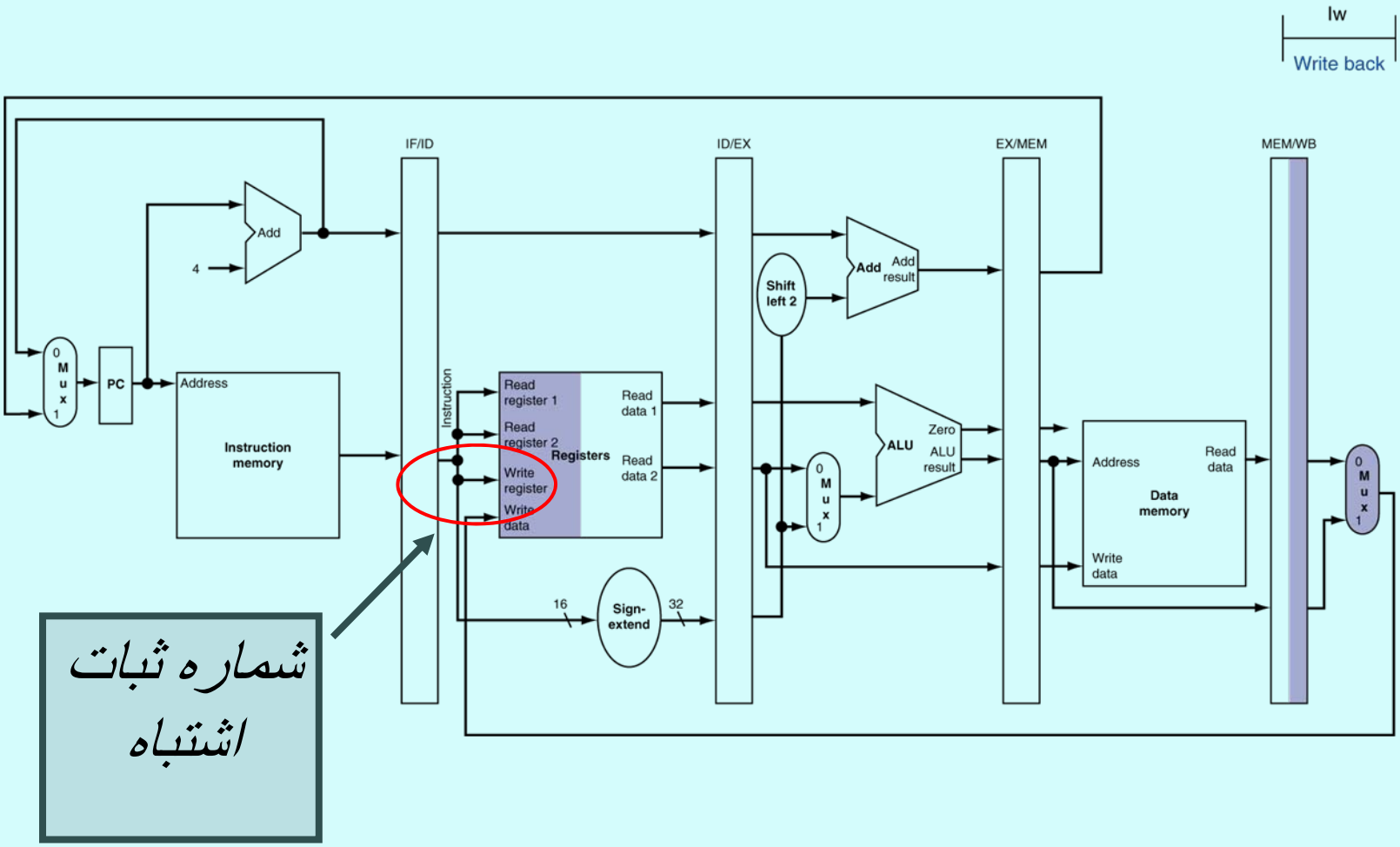
دستور خواندن حافظه EX



دستور خواندن حافظه MEM

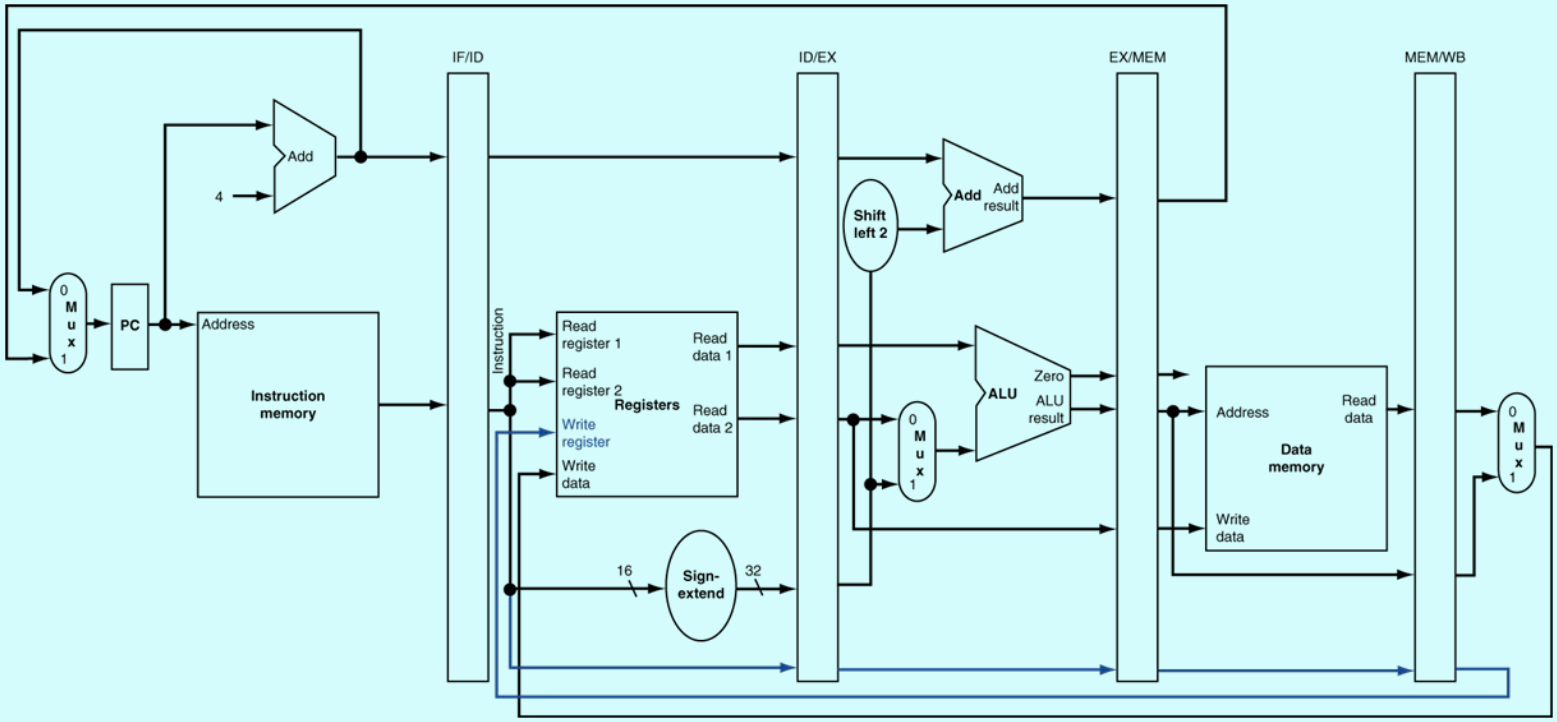


دستور خواندن حافظه WB

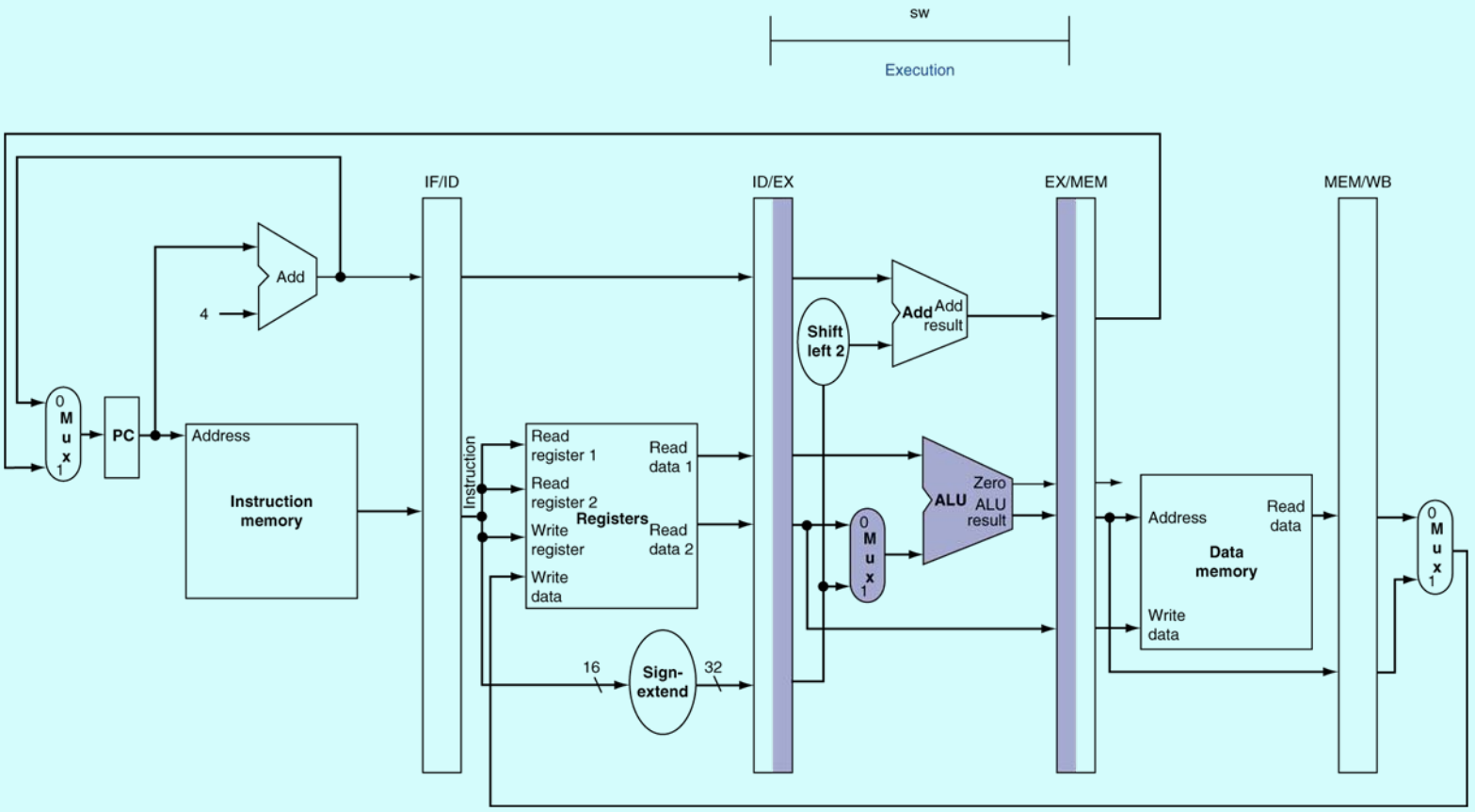


شماره ثبات اشتباه

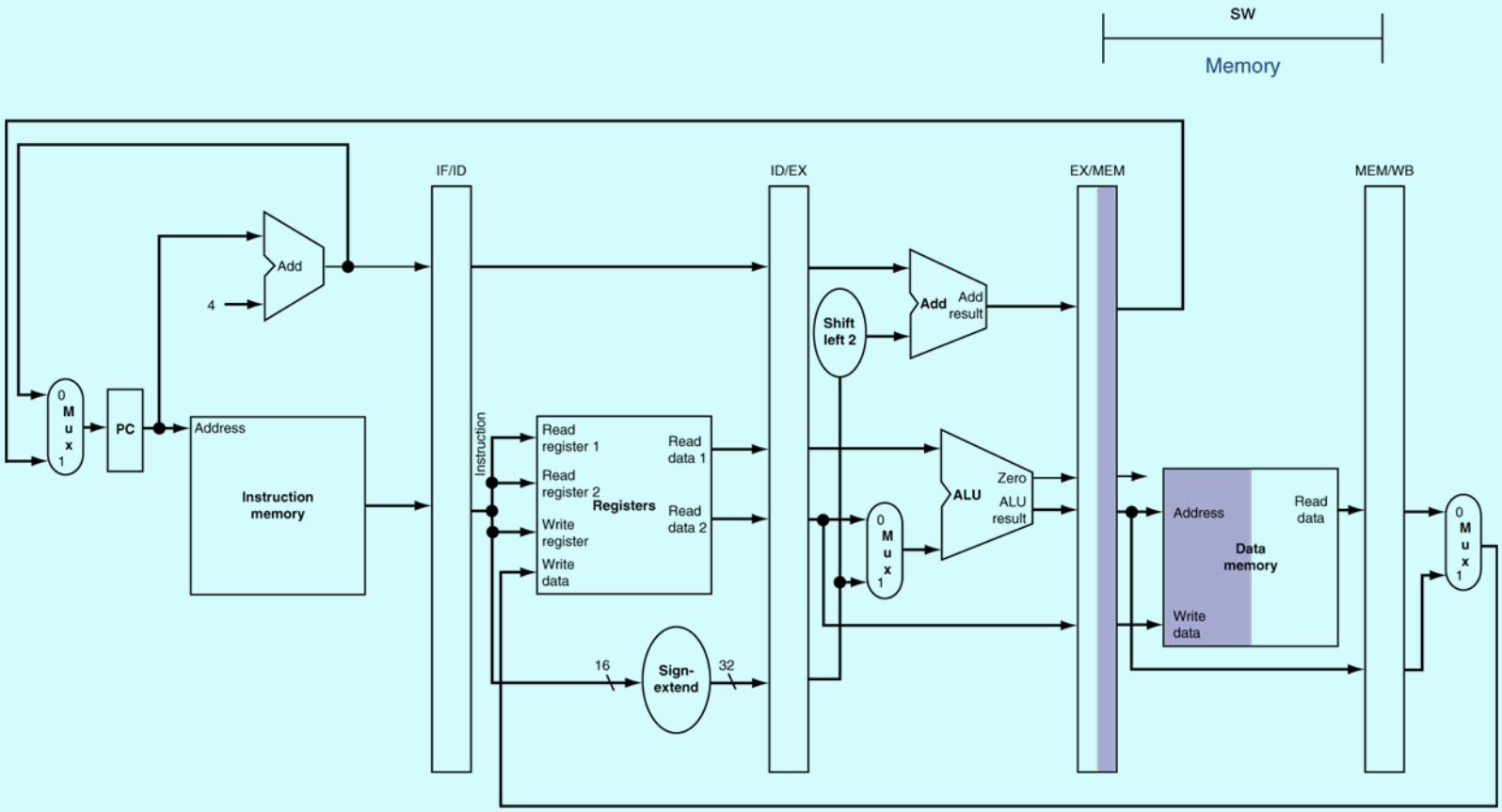
داده‌گذر اصلاح شده برای دستور خواندن



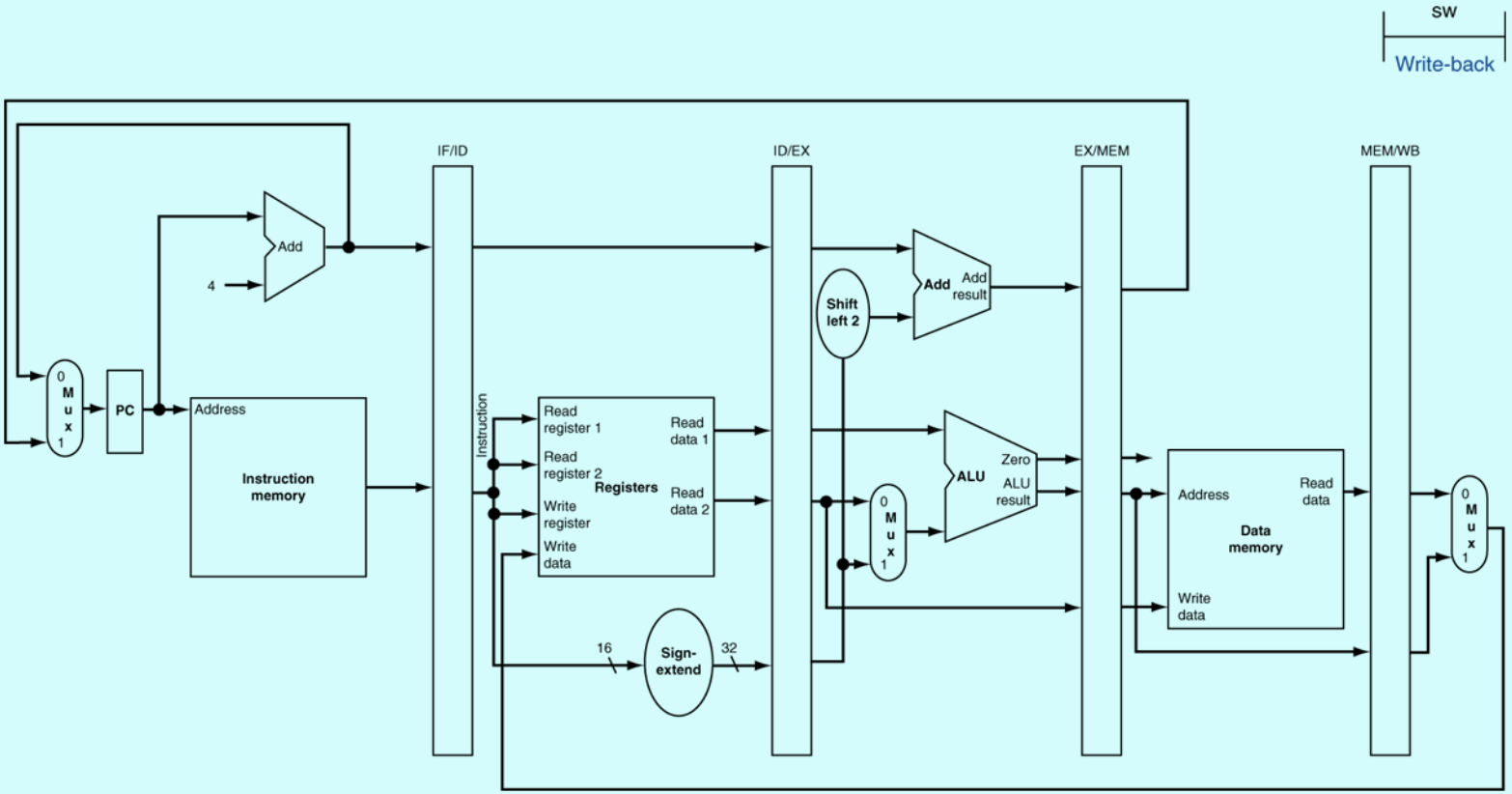
دستور نوشتن در حافظه EX



دستور نوشتن در حافظه MEM



دستور نوشتن در حافظه WB



نمودار خط لوله به صورت چندسیکلی

- در این شیوه به کارگیری منابع نشان داده شده است