طراحی واحد کنترل ALU

مدار ALU در MIPS با این ساختار کار می کند

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

عملکردهای ALU با توجه به حالات مختلف آن

ALU OP	Description
00	بارگذاری – ذخیره
01	تفریق برای پرش شرطی
10	برحسب مقدار FUNC در بطن دستور عملکرد ALU تعریف می شود

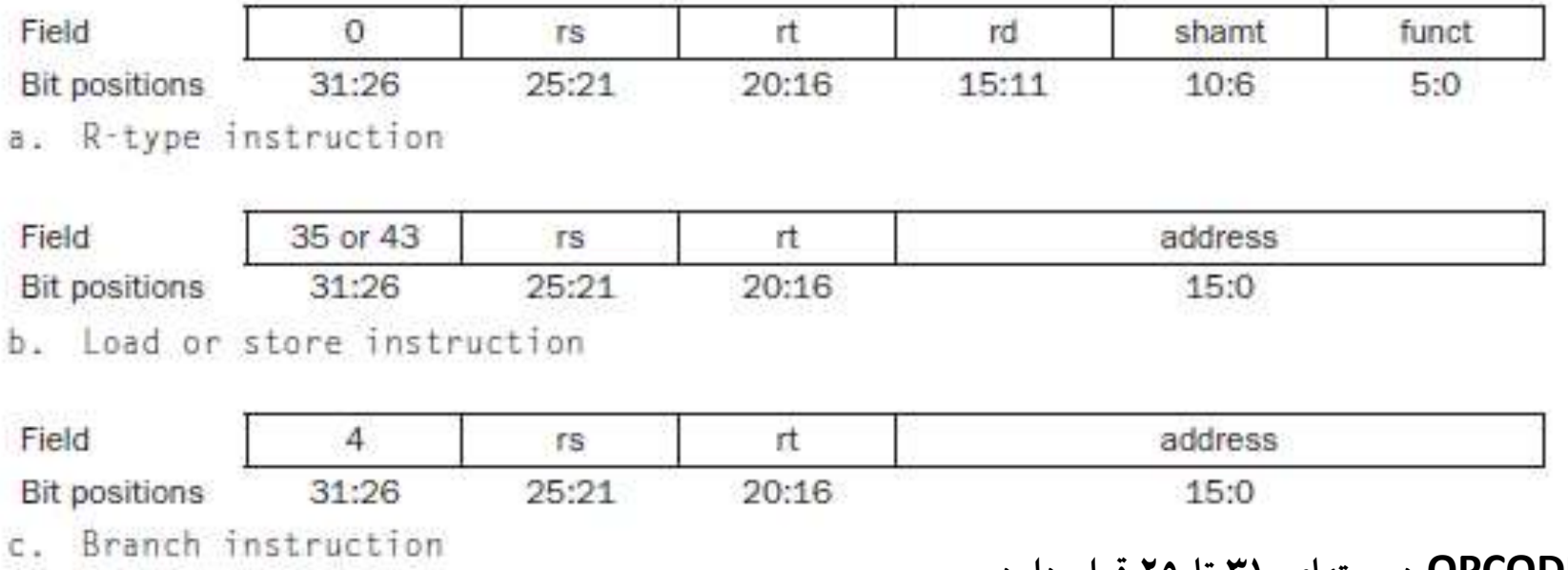
با توجه به ماهیت دستور، عملکردهای ALU باید تنظیم شوند

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

جدول صحت ورودی های ALU به صورت تابعی از
ALU OP و F5-F0

ALUOp		Funct field						
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	Operation
0	0	X	X	X	X	X	X	0010
0	1	X	X	X	X	X	X	0110
1	0	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	0	X	X	0	1	0	0	0000
1	0	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

طراحی واحد کنترل مرکزی



❖**OPCODE[5:0]** در بیتهای ۳۱ تا ۲۵ قرار دارد.

opcode The field that denotes the operation and format of an instruction.

❖ثبات ورودی اول در موقعیت ۲۱ تا ۲۵

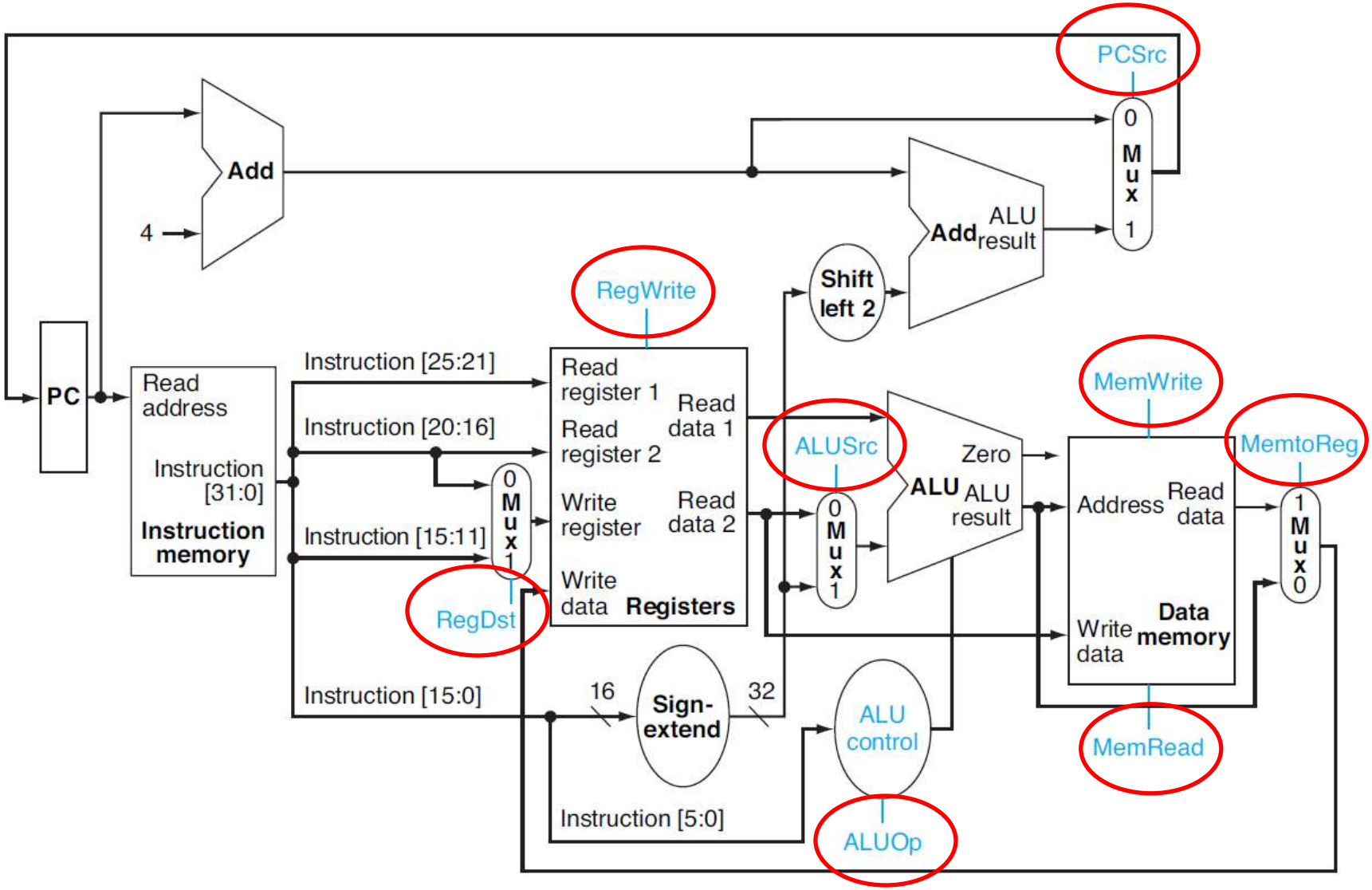
❖ثبات ورودی دوم در صورت وجود در ۱۶ تا ۲۰ قرار دارند.

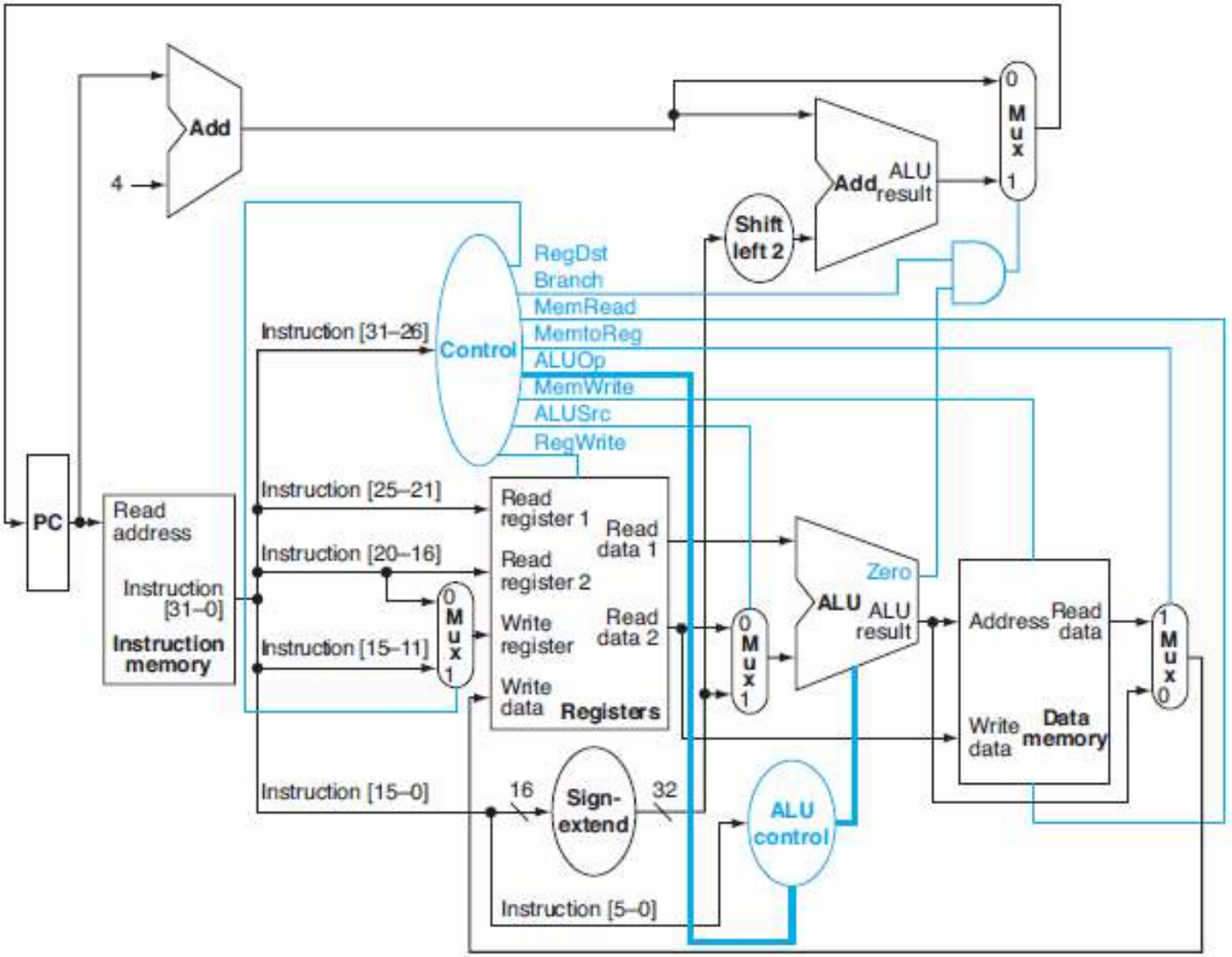
❖شانزده بیت افست در دستورهای **beq** و **lw**و **sw** در بیتهای ۰ تا ۱۵ قرار دارند

❖ثبات پایه در دستورات **lw** و **sw** در بیتهای ۲۱ تا ۲۵ قرار دارند

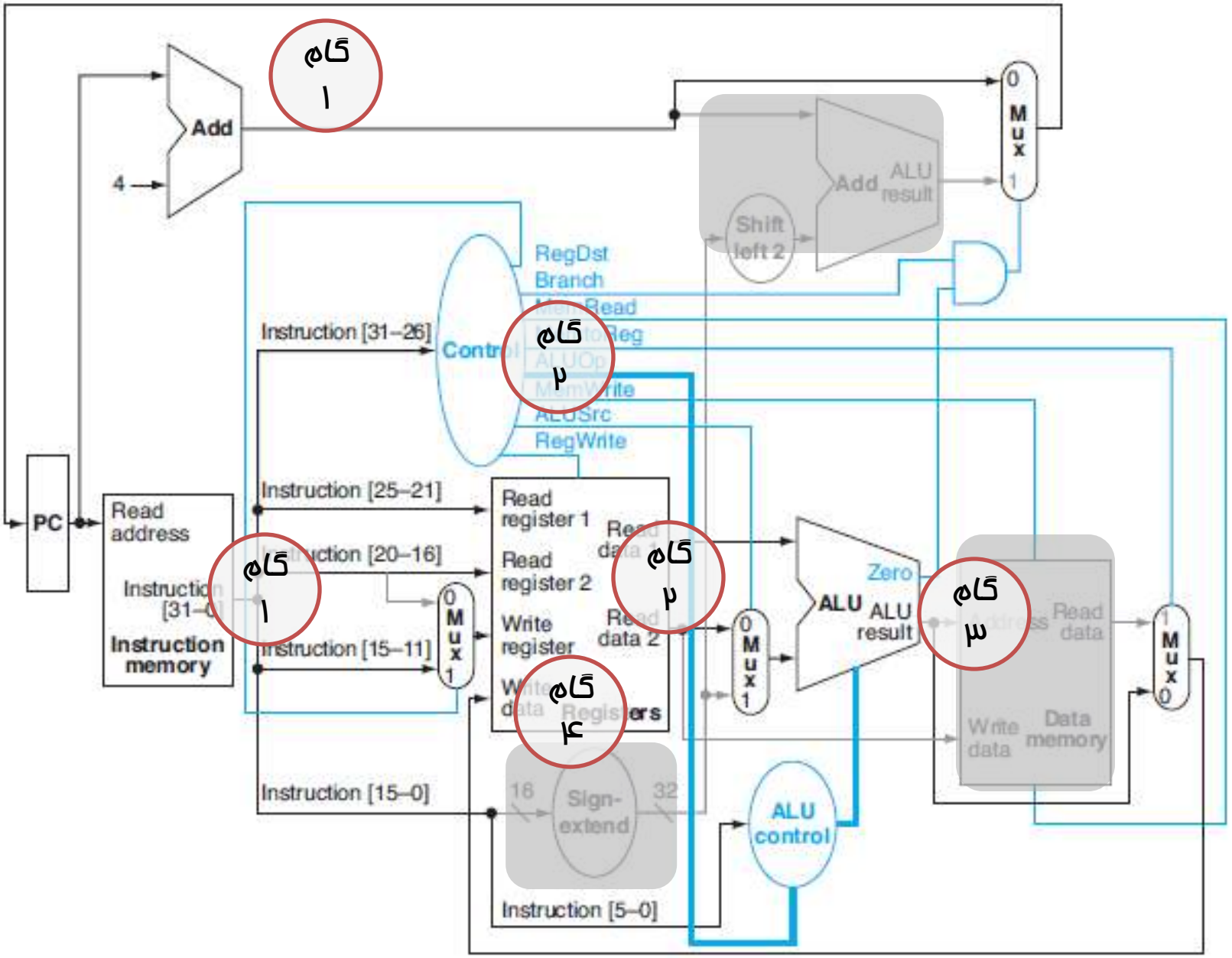
❖ثبات مقصد بسته به نوع دستور در بیتهای ۱۶–۲۰(ثبات **rt** در دستور **load**) و یا ۱۱–۱۵ (ثبات **rd** دستورات **R**)قرار دارند.

با توجه به این اطلاعات، ساختار datapath طراحی شده را کامل تر می کنیم

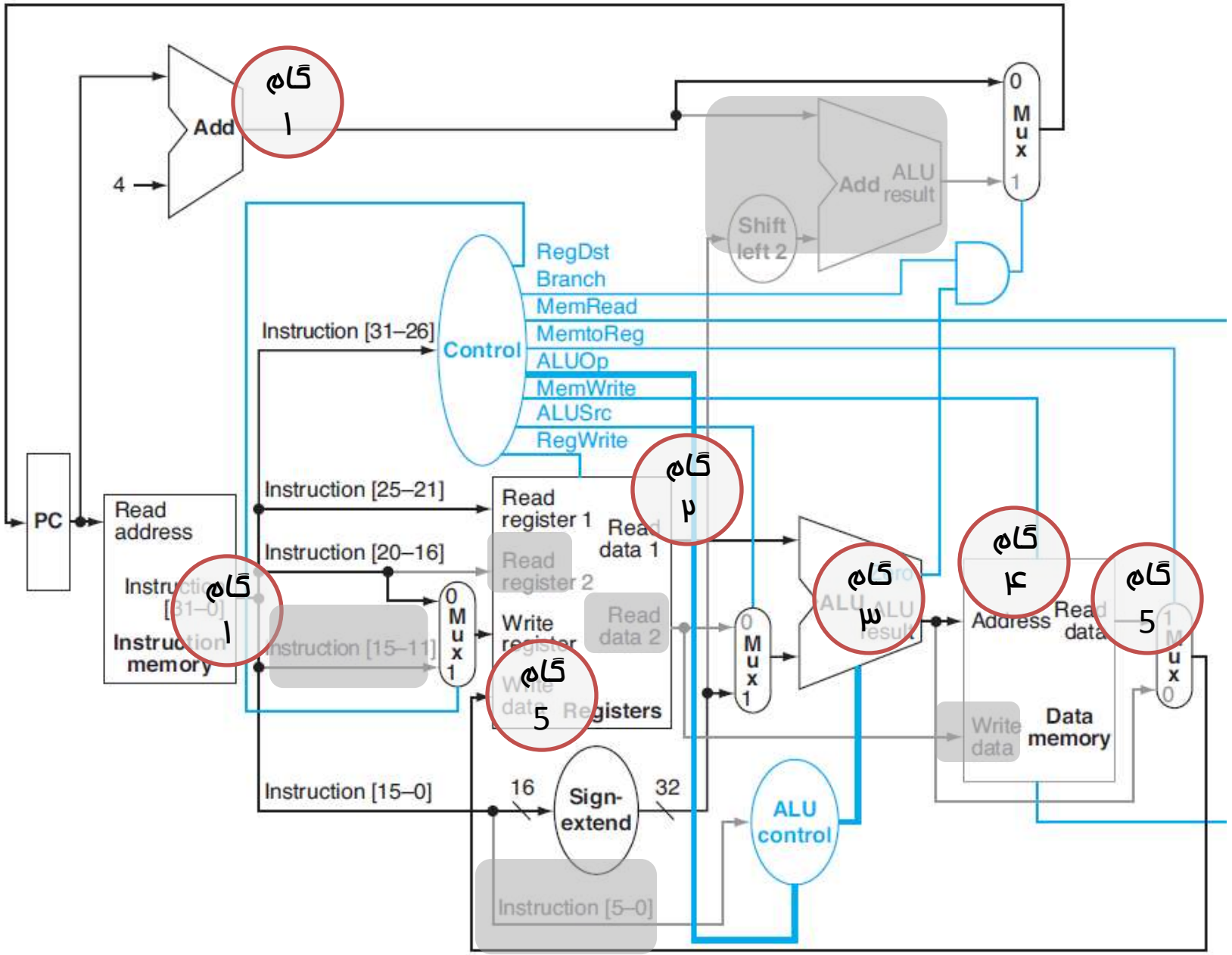




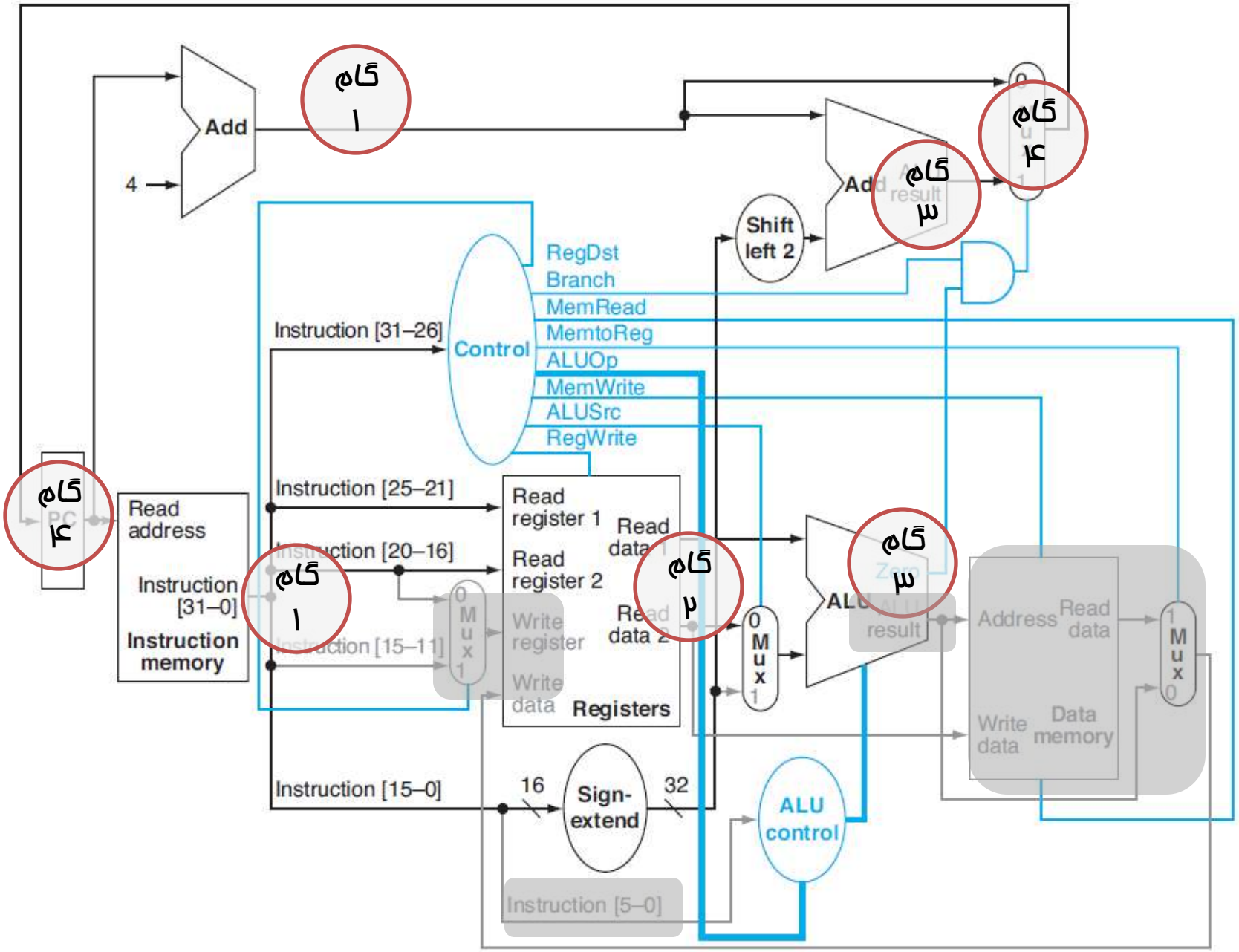
Instruction	RegDst	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1



The datapath in operation for an R-type instruction, such as **add $t1,$t2,$t3**



The datapath in operation for a load instruction

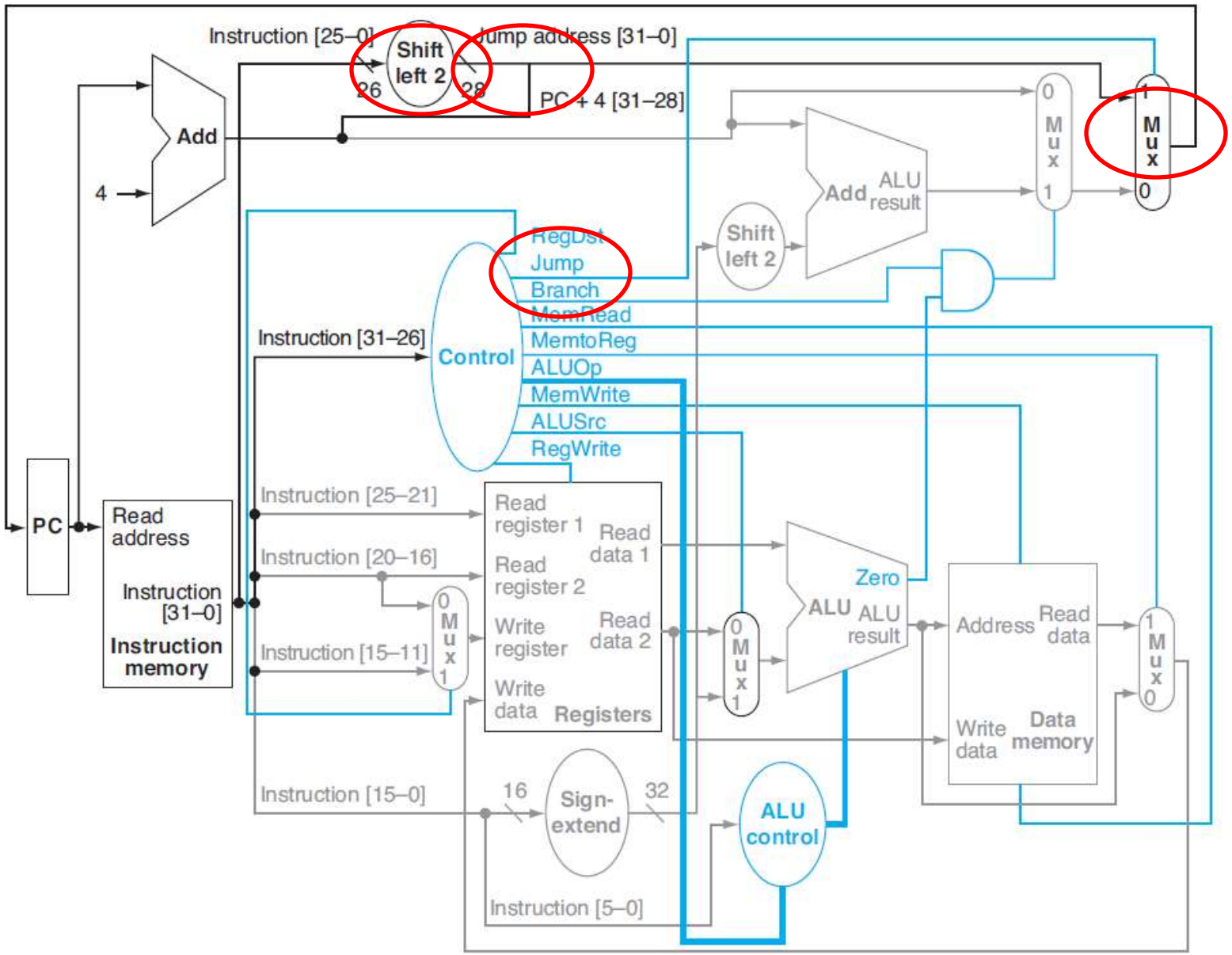


The datapath in operation for a branch-on-equal instruction.

با توجه به توجه به حالات مختلف دستورات و opcode می توان سیگنالهای هر دستور را محاسبه نمود و جدول صحتی به شرح زیر استخراج می شود

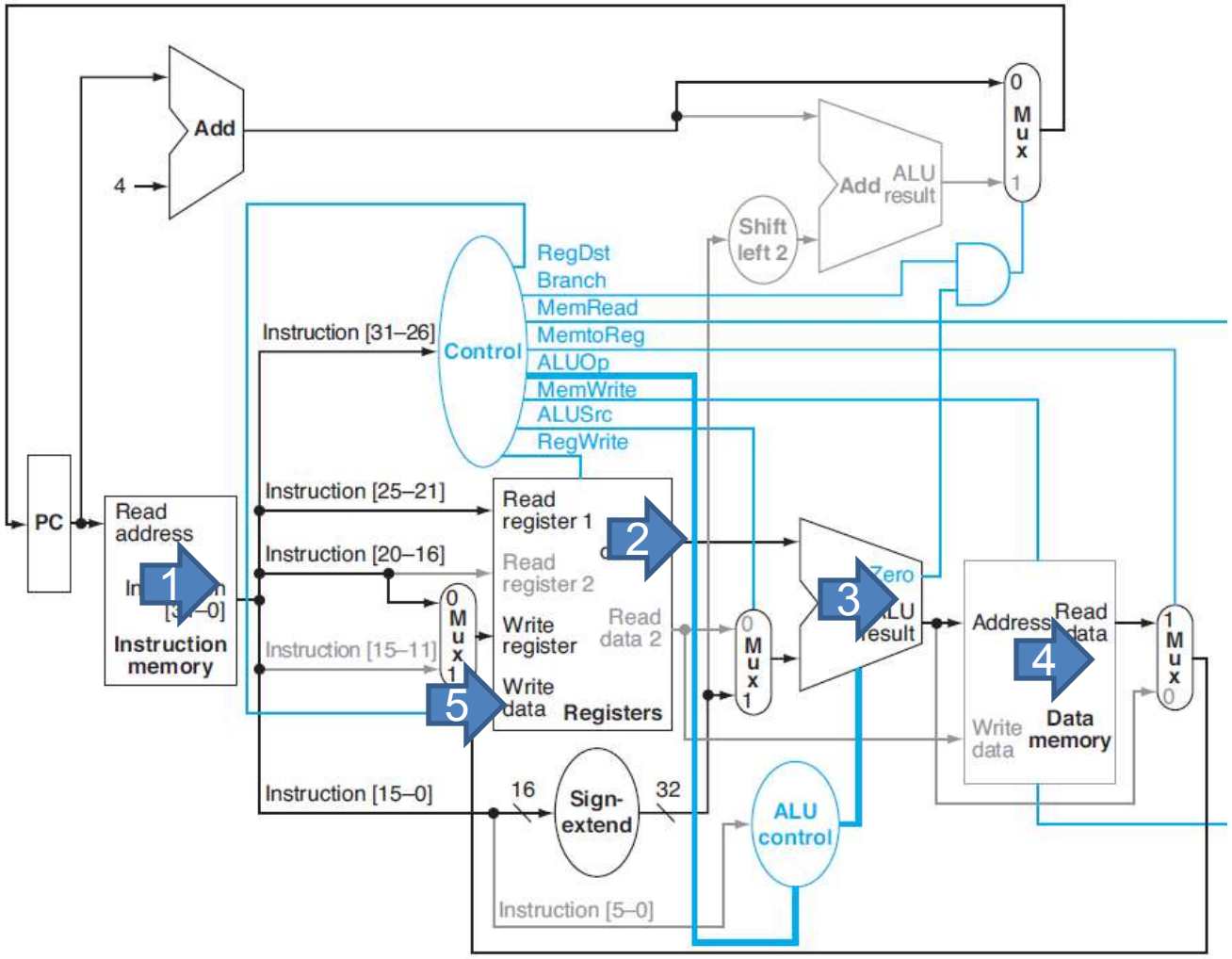
Input or output	Signal name	R-format	lw	sw	beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

اضافه کردن دستور دستور پرش غیر شرطی unconditional jump به گذرگاه داده



اشکالات پیاده سازی تک سیکلی

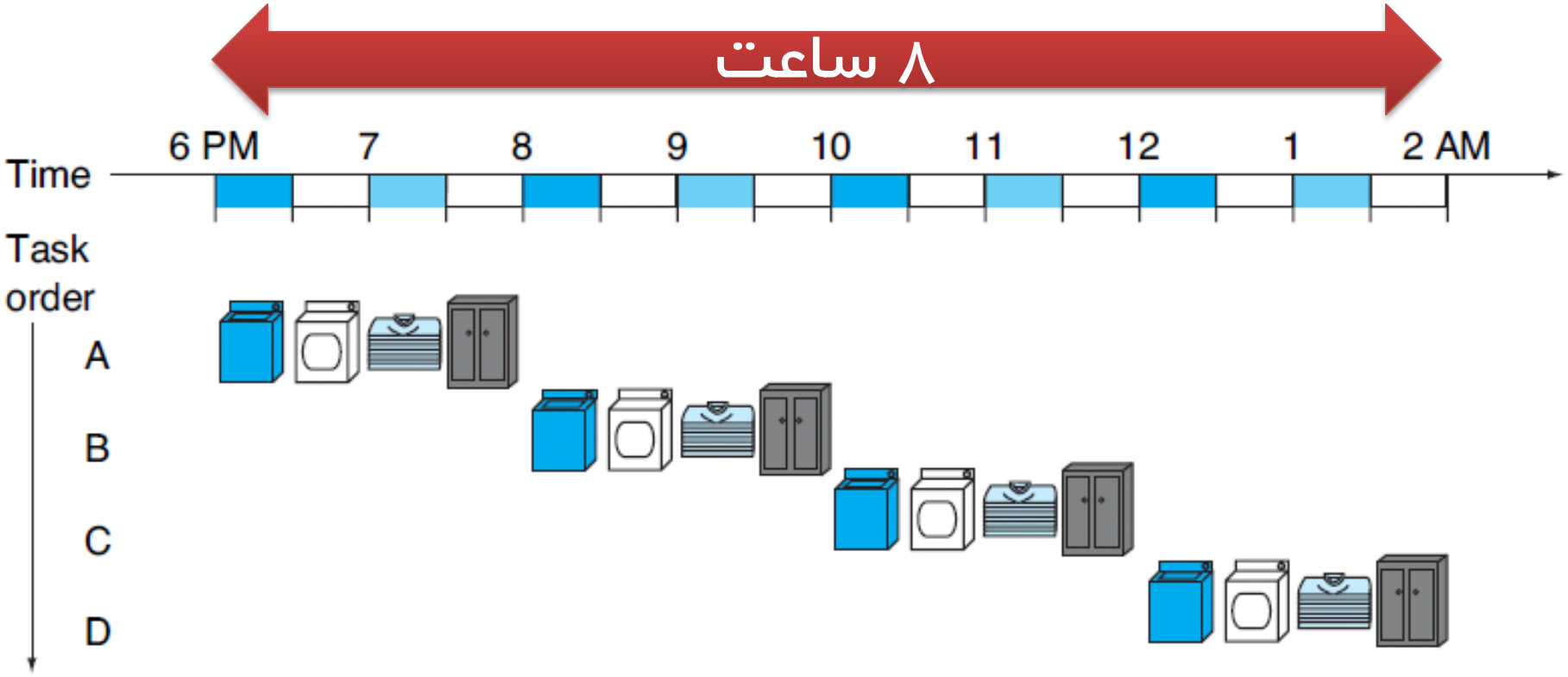
- حداقل زمان سیکل = زمان طولانی ترین دستور یا طول بلندترین مسیر اجرایی
- بلندترین مسیر در طراحی ما مربوط به دستور **SW** است که از ۵ واحد عملیاتی عبور می کند



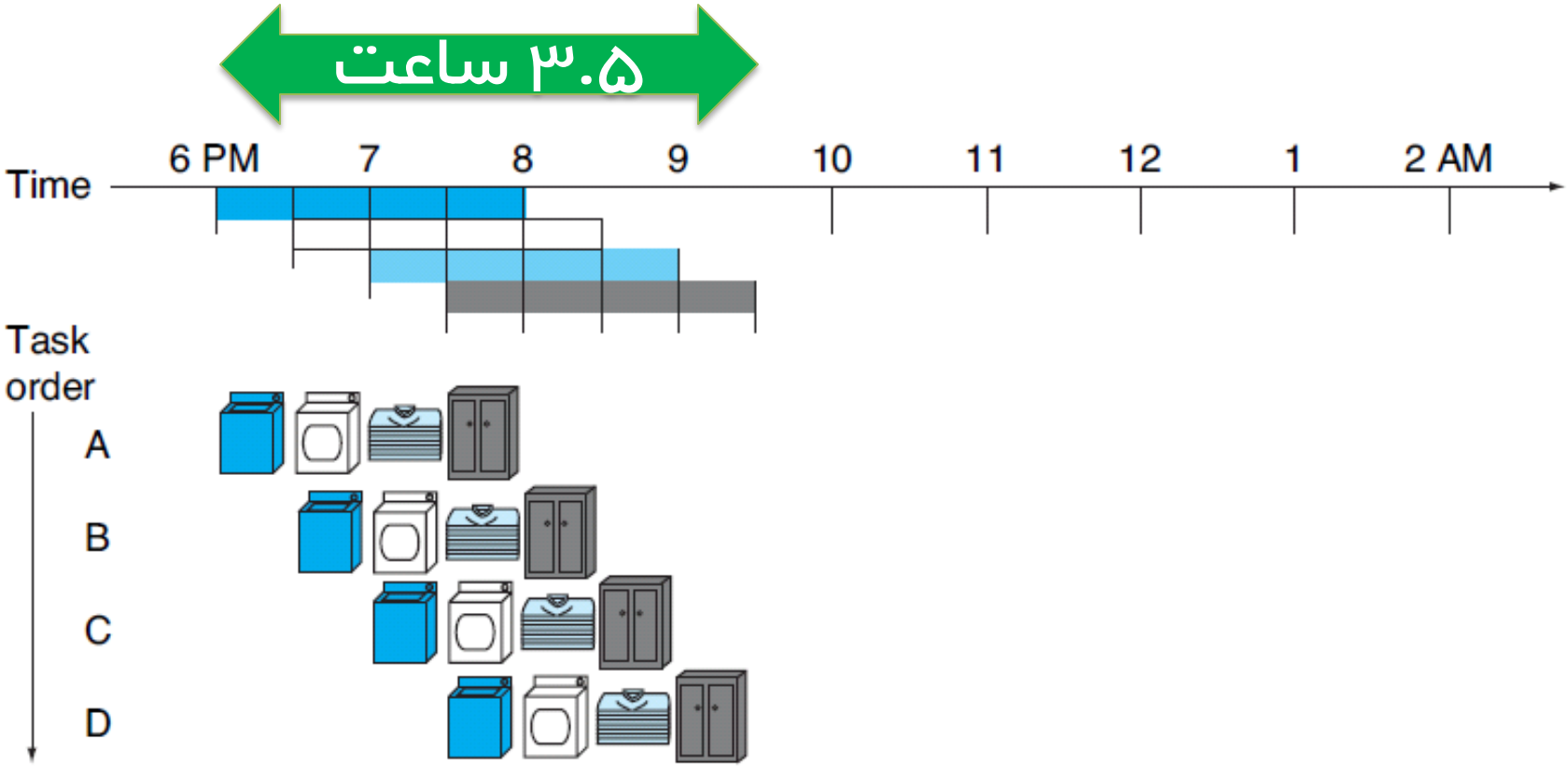
اشکالات پیاده سازی تک سیکلی

- باید بدترین دستور را بهبود دهیم.
- چون سرعت پردازنده محدود به سرعت بدترین دستور می شود، اصل دوم طراحی ما نقض می شود
  - ~~مؤلفه های پرکاربرد را بهبود ببخشید~~

اجرا بدون همپوشانی زمانی



اجرا با همپوشانی زمانی



نکات مهم

- زمان اجرای هر **task** به طور کامل در هر دو حالت برابر است. یعنی در هر دو حالت ۲ ساعت زمان صرف شده است تا یک **task** تکمیل شود. چون مراحل لازم در هر دو حالت ثابت و زمان آنها یکسان است.
- عامل افزایش سرعت، کاهش زمان لازم برای اجرای هر عمل نیست، بلکه افزایش برون دهی است که منجر به کاهش زمان اجرای کل کارها می گردد.

- اگر فرض کنیم زمان لازم برای هر کدام از ۴ مرحله برابر باشد (در این مثال نیم ساعت)، با این سیستم ضریب افزایش سرعت حداکثر ۴ خواهد بود.
- چرا؟ تا پیش از این در هر ۲ ساعت یک task خاتمه می یافته است اما با روش جدید هر نیم ساعت یک task خاتمه می یابد (همچنان زمان اجرای هر task دو ساعت است)
- Speed Up Factor = number of stages

خط لوله در پردازنده ها

- عموماً دستورات در پنج گام تکمیل می شوند:

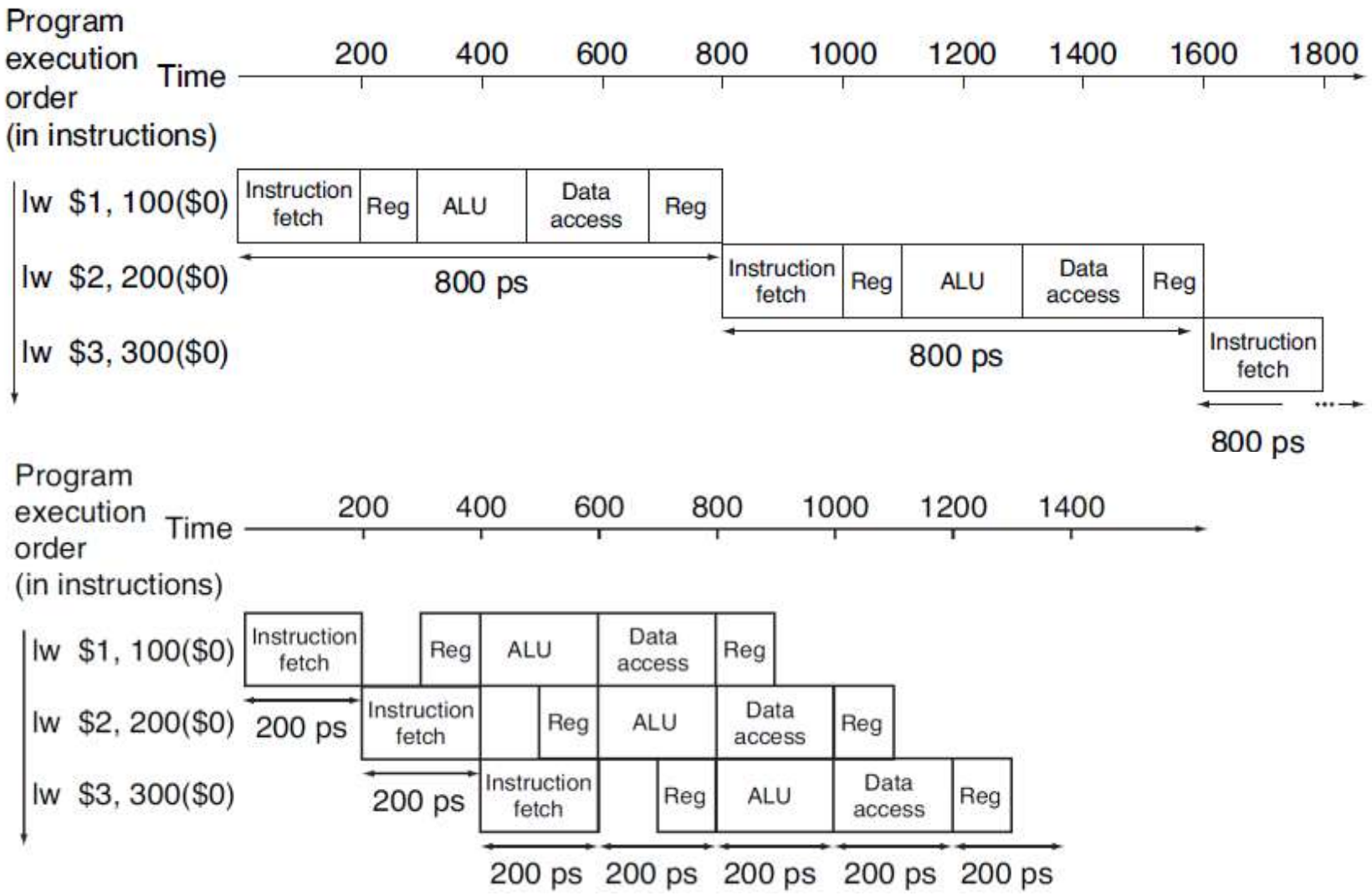
1. واکشی دستور (FETCH)
2. خواندن ثباتهای (Register Read)
3. اجرا دستور یا محاسبه آدرس (ALU Operation)
4. مراجعه به عملوندی در حافظه داده (data access)
5. نوشتن نتیجه در یک ثبات (register write)

زمان اجرای دستور در حالت تک سیکلی

Instruction class	Instruction fetch	Register read	ALU operation	Data access	Register write	Total time
Load word (lw)	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
Store word (sw)	200 ps	100 ps	200 ps	200 ps		700 ps
R-format (add, sub, AND, OR, slt)	200 ps	100 ps	200 ps		100 ps	600 ps
Branch (beq)	200 ps	100 ps	200 ps			500 ps

برخی تأخیرهای معادل صفر در نظر گرفته شده اند مثل تأخیر ناشی از مالتی پلکسرها، واحد کنترل، دسترسی به PC

مقایسه زمان اجرا



ضریب افزایش سرعت (Speed Up)

- تعریف کلی **Speed UP** در خط لوله:

$$Speed\ Up = \frac{(execution\ time)_{non-pipelined}}{(execution\ time)_{pipelined}}$$

ضریب افزایش سرعت (Speed Up)

- در حالت ایده آل که تمام مراحل متوازن **(Balanced)** باشند و تعداد دستورات به مقدار کافی زیاد باشد:

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

$$Speed\ Up = \frac{time_{non-pipelined}}{time_{pipelined}} = number\ of\ stages$$

ضریب افزایش سرعت (Speed Up)

- در مثال ما با ۳ دستور LW و ۵ ایستگاه نا متوازن

$$Speed\ Up = \frac{time_{non-pipelined}}{time_{pipelined}} = \frac{2400}{1400} \cong 1.7\ !!!$$

- دلیل: تأخیر ناشی از خالی ماندن خط لوله، نا متوازن بودن دستورات
- اگر دستورات پشت سر هم اجرا شود و تعداد آنها به 1 میلیون و 3 عدد افزایش یابد:

$$Speed\ Up = \frac{time_{non-pipelined}}{time_{pipelined}} = \frac{800{,}002{,}400}{200{,}001{,}400} \cong \frac{800}{200} = 4$$

- دلیل: نا متوازن بودن سرعت اجرا در ایستگاه ها

افزایش سرعت
در خط لوله با ایستگاه های نامتوازن

زمان اجرای دستور در حالت تک سیکلی (غیر خط لوله ای)

$$Speed\ Up = \frac{time_{non-pipelined}}{time_{pipelined}} = \frac{800,002,400}{200,001,400} \cong \frac{800}{200} = 4$$

زمان لازم برای تکمیل عملیات کندترین ایستگاه در حالت خط لوله ای

فرض مهم: تعداد دستورات به مقدار کافی زیاد است که تأخیر ناشی از خالی بودن خط لوله را می توان ناچیز در نظر گرفت

ملاحظات اعمال شده در MIPS

1. تمام دستورات طول یکسانی دارند (یک کلمه ۳۲ بیتی)
   - در گام اول واکشی دستور تکمیل و در گام دوم کدگشایی و تولید سیگنالها انجام می شود
   - در X86 به دلیل تنوع طول دستورات چنین نیست
2. قالب دستور العمل ها در MIPS محدود است
   - محل ثباتهای مبدأ در دستور ثابت و معلوم است لذا قبل از دیکد شدن OPCODE می توان شروع به خواندن ثباتها کرد

ملاحظات اعمال شده در MIPS

3. عملوندهای حافظه تنها در دو نوع دستور وجود دارند (دستورات SW و LW)
   - محاسبات ریاضی در این دستورات صرفاً محاسبه آدرس است پس گام اجرا در این دستورات را با گام محاسبه آدرس می توان جایگزین کرد (به عبارتی پس از محاسبه آدرس، دیگر نیازی به انجام محاسبه برای اجرای دستور وجود ندارد)
   - اگر مانند X86 روی حافظه عملیات ریاضی به صورت مستقیم قابل انجام بود، دو گام اجرا و دسترسی حافظه به ۳ گام محاسبه آدرس، دسترسی حافظه و اجرا تبدیل می شد.
4. به دلیل تراز بودن حافظه، دسترسی به حافظه در MIPS در یک مرتبه رجوع به حافظه تکمیل می شود.

مخاطرات (Hazard)در خط لوله

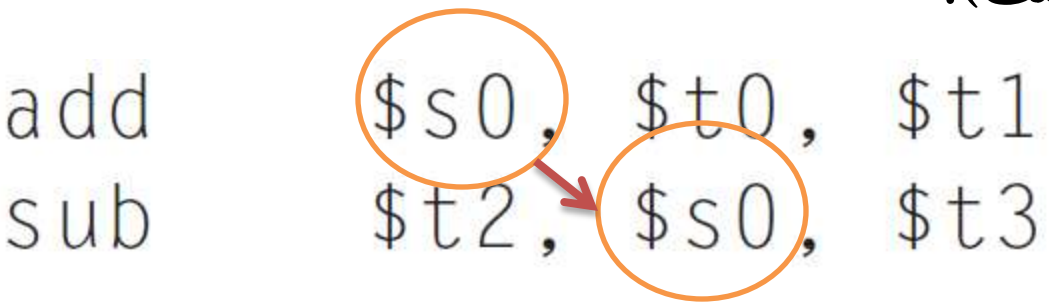
- در برخی شرایط دستورات را نمی توان پشت سرهم و بلافاصله اجرا نمود. به رخدادهای از این دست Hazard گفته می شود:
  - Structural Hazards (مخاطرات ساختاری)
  - Data Hazards (مخاطرات ناشی از داده)
  - Control Hazards (مخاطرات کنترلی)

مخاطرات ساختاری

- به دلیل ترکیب خاصی در ساختار پردازنده، امکان اجرای دو گام همزمان وجود نداشته باشد.
- مثال: اگر حافظه داده و حافظه دستور از هم مجزا نباشند، گامهای واکشی دستور (گام ۱) و گام رجوع به حافظه (گام ٤) همزمان با هم قابل اجرا نیست.

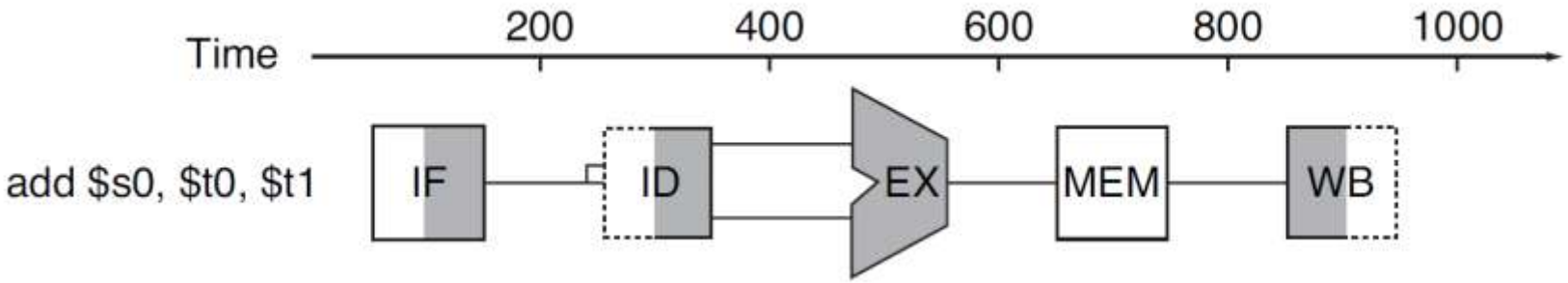
مخاطرات ناشی از داده

- زمانی رخ می دهد که یک دستور سیکل ساعت برنامه ریزی شده نتواند اجرا شود به این دلیل که داده ای که برای اجرا نیاز دارد، هنوز آماده نشده (**available** نیست).

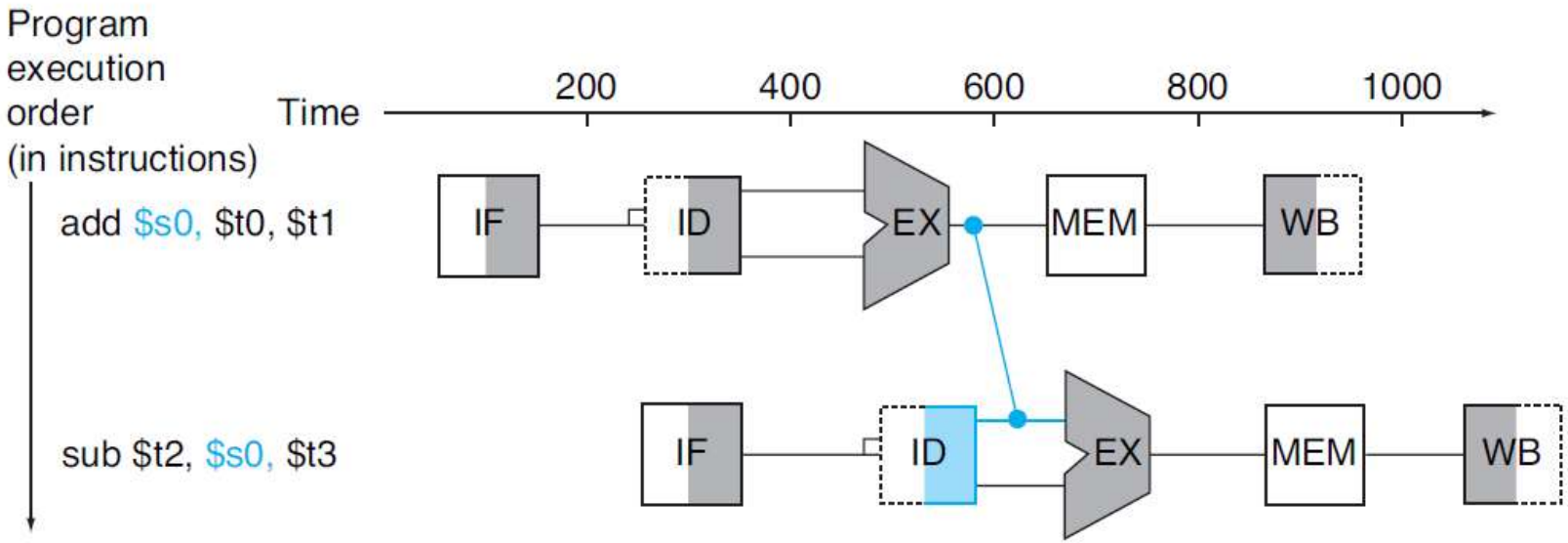


- دستور **SUB** در گام ۲، باید منتظر پایان گام ۵ دستور **add** شود و ۳ سیکل ساعت هدر می رود!
- تا حد مختصری با کمک کامپایلر قابل بهبود است اما نه به طور کامل
- استفاده از روشهای سخت افزاری مثل **forwarding** که نتیجه را از **ALU** به صورت مستقیم در اختیار دستور معلق قرار می دهد (هزینه سخت افزاری دارد)

نمایش بصری اجرای دستور



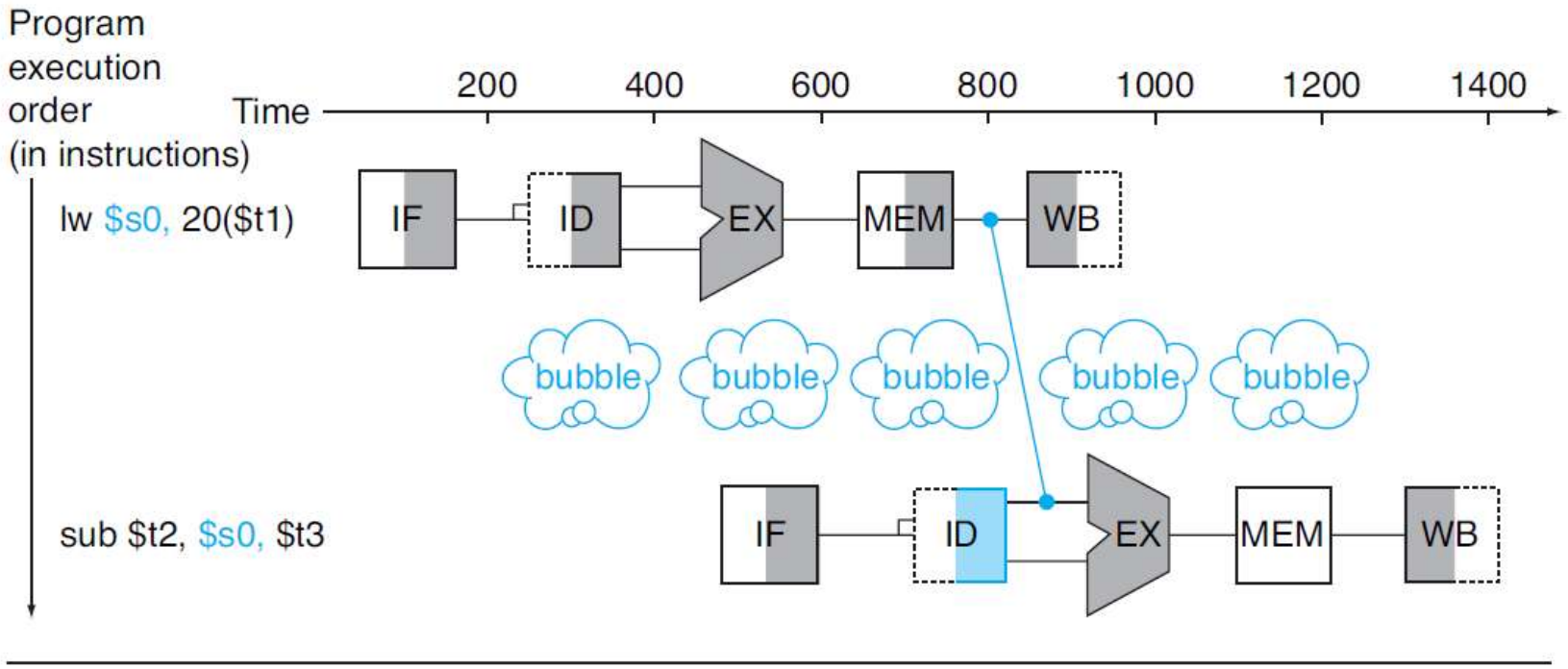
Forwarding/Bypassing



Forwarding زمانی قابل استفاده خواهد بود که مرحله مقصد دیرتر از مرحله مبدأ اجرا شود (عقب تر باشد)

همیشه این روش کارساز نیست مثلاً اگر به جای دستور Add دستور LW $S0 را داشتیم تا پایان مرحله MEM کاری نمی توان کرد و باز هم یک سیکل خط لوله توقف نیاز دارد

تعلیق در خط لوله



نمونه از بهبود کدها توسط کامپایلر

```
a = b + e;
c = b + f;
```

فرض: تمام عملوندها در حافظه هستند

```
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)
```

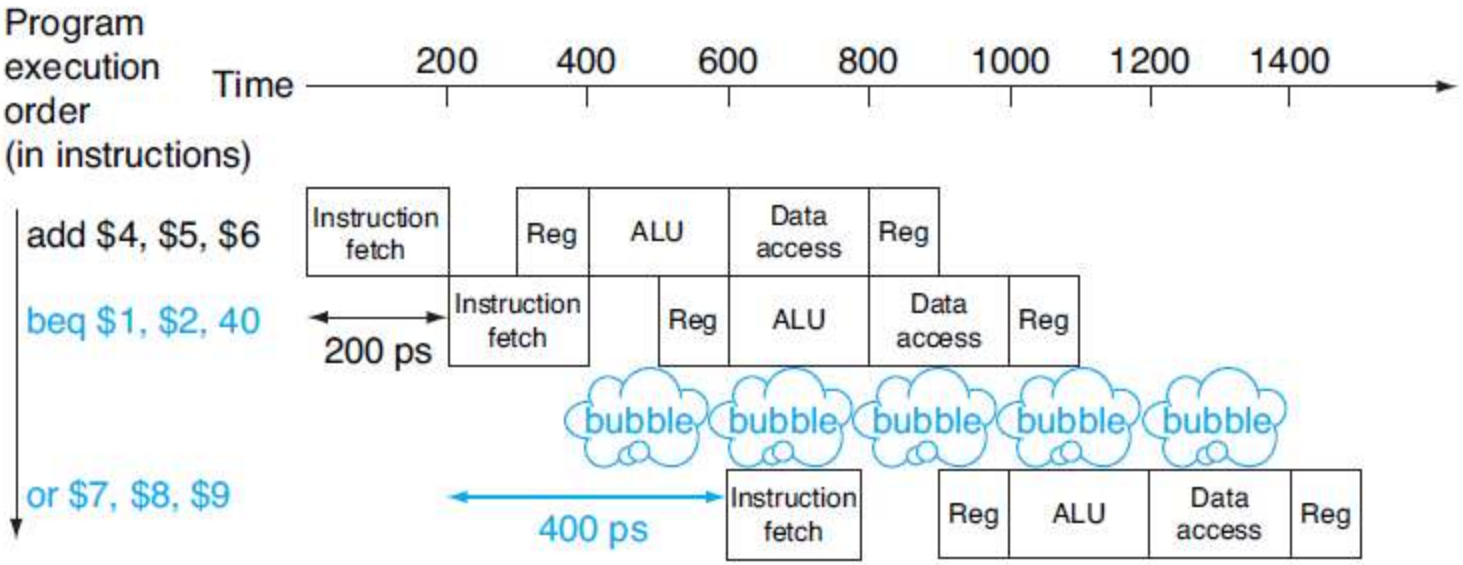
```
lw    $t1, 0($t0)
lw    $t2, 4($t0)
lw    $t4, 8($t0)
add   $t3, $t1,$t2
sw    $t3, 12($t0)
add   $t5, $t1,$t4
sw    $t5, 16($t0)
```

Control Hazard/Branch Hazard

- **Control Hazard**: رخدادی که یک دستور در سیکل مقرر اجرا نشود به این دلیل که دستوری که واکشی شده است، دستوری که قرار است اجرا شود، نیست!
- دستوراتی که بعد از پرشهای شرطی، باید اجرا شوند، تا زمان محاسبه آدرس مؤثر پرش شرطی قابل واکشی به صورت دقیق نیستند چون معلوم نیست که چه دستوراتی باید واکشی شوند.

راه حلها

- **Stall** یا تعلیق: واکشی دستورات بعدی را تا زمانی که آدرس مؤثر دستور پرش شرطی معلوم نشده است، به تعویق می اندازیم.
- فرض می کنیم با روشهایی، می توانیم این آدرس را بعد سیکل دوم یعنی **Instruction Decode**محاسبه کنیم.
- به ازای هر پرش شرطی یک تعلیق در خط لوله ایجاد می کنیم.



- با توجه به این که ۱۷٪ دستورات SPEC2006، پرش شرطی هستند و با توجه به این CPI سایر دستورات 1است، CPI به مقدار ۱.۱۷ افزایش می یابد
- اگر نتوانیم آدرس پرش شرطی را تا سیکل دوم تشخیص دهیم، تأخیر بیشتر هم می شود

Branch Prediction

- روش دیگر تخمین آدرس احتمالی پرش قبل از محاسبه مقدار قطعی آن است.
- با استفاده از این آدرس احتمالی، واکشی دستورات را ادامه می دهیم.
- اگر تخمین درست بود به کار خود ادامه می دهیم
- اگر اشتباه بود، باید اثر چند دستوری که اشتباه اجرا شده را از بین برده و دستورات جدید را واکشی کنیم

روشهای تخمین آدرس پرش

- فرض می کنیم پرش رخ نمی دهد و برنامه به روند عادی خود ادامه می دهد
- روش دیگر تخمین پرش تخمین نسبت به رفتار کلیشه ای و متداول پرش است (مانند آنچه که درپرشهای حلقه های تکرار رخ می دهد) – اشکل این است که در یک بار اجرای پرش توفیق چندانی ندارد.
- راهکارهای دینامیک سخت افزاری: سرنوشت دستور بعدی پرش با توجه به وضعیت فعلی و دقیقاً همین یک دستور، حدس زده می شود. این حدسیات به مرور زمان در برنامه ممکن است تغییر کند.
- یک نمونه از این روش ثبت سوابق پرشها قبلی و استفاده از آن در تخمین است که هزینه سخت افزاری زیادی دارد اما تا ۹۰٪ دقت را می تواند ایجاد کند.
- طولانی شدن طول خط لوله، باعث کاهش دقت در تخمینها و افزایش hazard می شود

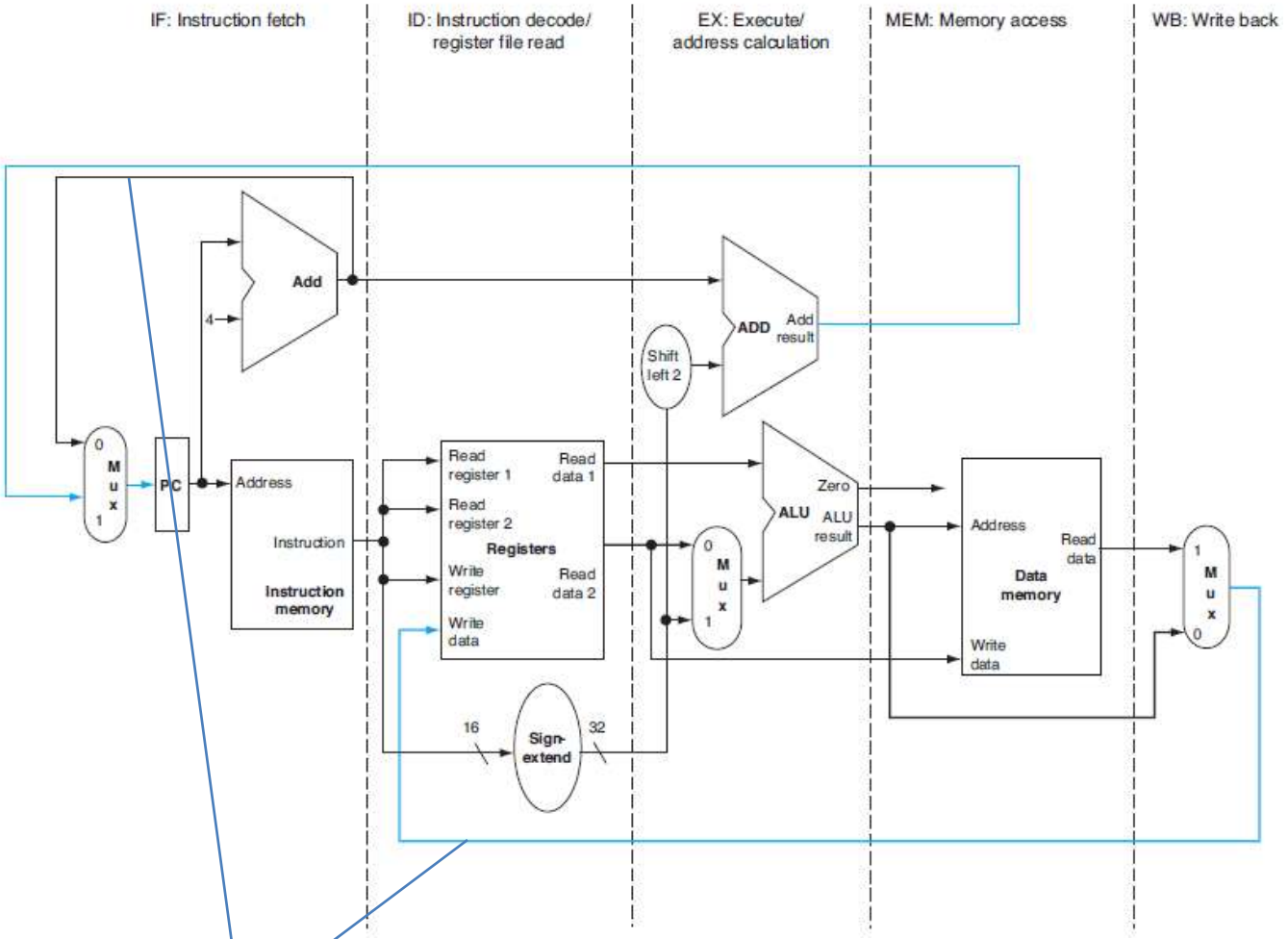
Delayed Decision

- روش سوم در برخورد با **Control Hazard** این است که دستورات را به نحوی جابجا کنیم که در روند برنامه خللی ایجاد نشود و بعد از دستور پرش شرطی دستوراتی قرار بگیرد که خط لوله خالی نماند.
- به عبارتی برخی از دستورات قبل پرش شرطی به بعد آن منتقل می شوند که خط لوله تا هنگام محاسبه آدرس مؤثر خالی نماند (این دستورات در محاسبه نتیجه پرش نباید تأثیری داشته باشند)
- برای پرشهای کوتاه مفید است و در پرشهای بلند، از روشهای تخمین آدرس پرش استفاده می شود.

نگاه کلی به خط لوله

- به زمان اجرای هر دستور به صورت مستقل، **Latency** می گویند.
- ایجاد خط لوله باعث کاهش **Latency** نمی شود، بلکه باعث افزایش **Throughput** (برون دهی) می شود.
- **Structural Hazard**: در برنامه های ممیز شناور بیشتر رخ می دهد چون ذاتاً عملیات ممیز شناور سخت تر به صورت خط لوله ای پیاده سازی می شوند.
- **Control Hazard**: در برنامه های با عملیات صحیح احتمال رخداد آنها بیشتر است چون ذاتاً عملکرد آنها سخت‌تر قابل پیش بینی هستند.
- **Data Hazard**: در همه نوع برنامه ای رخ می دهند اما چون رفتار و عملکرد برنامه های ممیز شناور قابل پیش بینی تر هستند، کامپایلرها می توانند ترتیب اجرای دستورات را بهتر تخمین بزنند و دستور العملها را به نحوی زمان بندی **(Schedule)** کنند، که این مخاطرات کمتر رخ دهد.

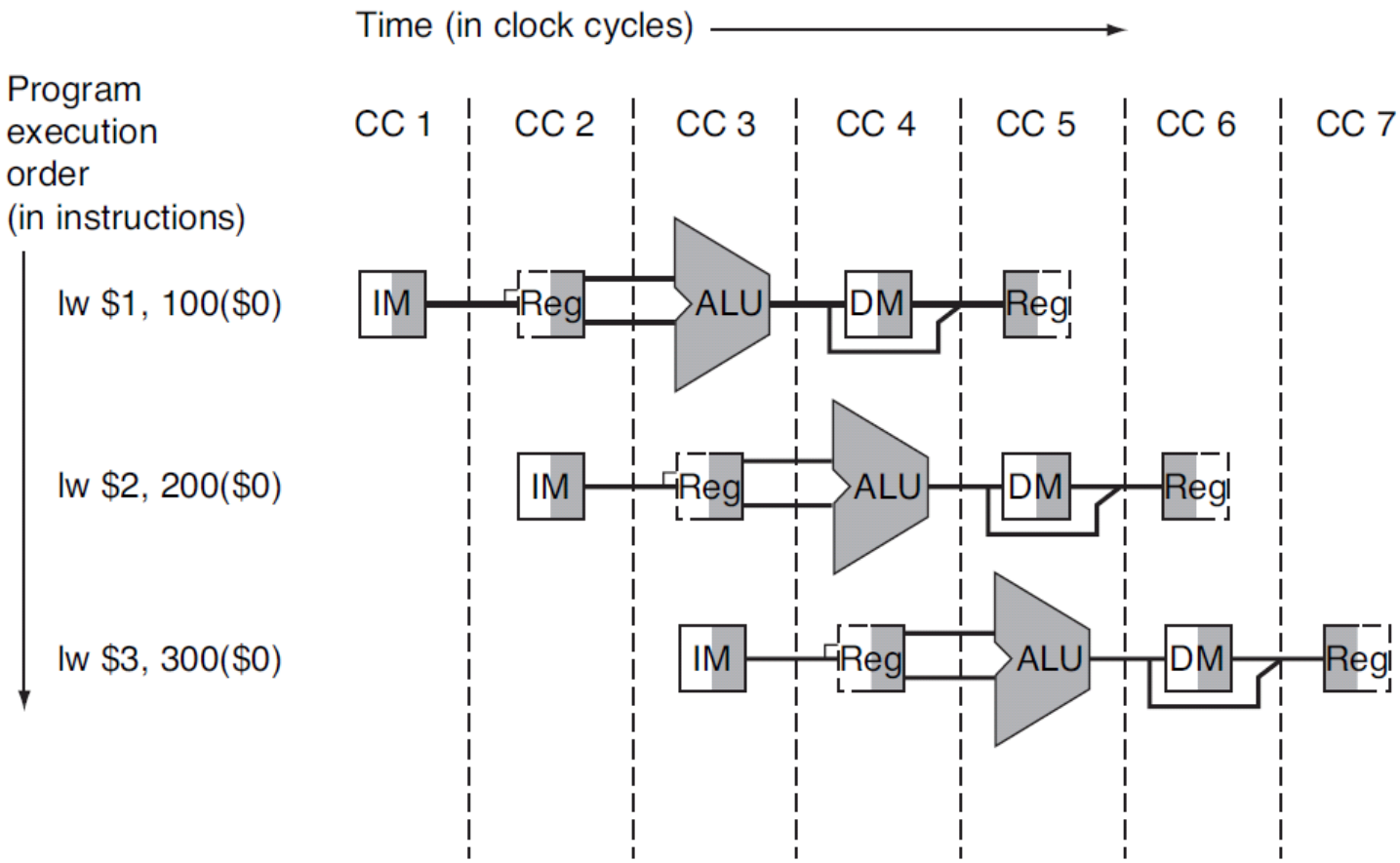
تفکیک مسیر گذر داده جهت اجرای خط لوله ای



•دو مسیر در مسیر گذر داده از راست به چپ وجود دارد، وجود آنها باعث تأثیر در دستورات بعدی در خط لوله می شود

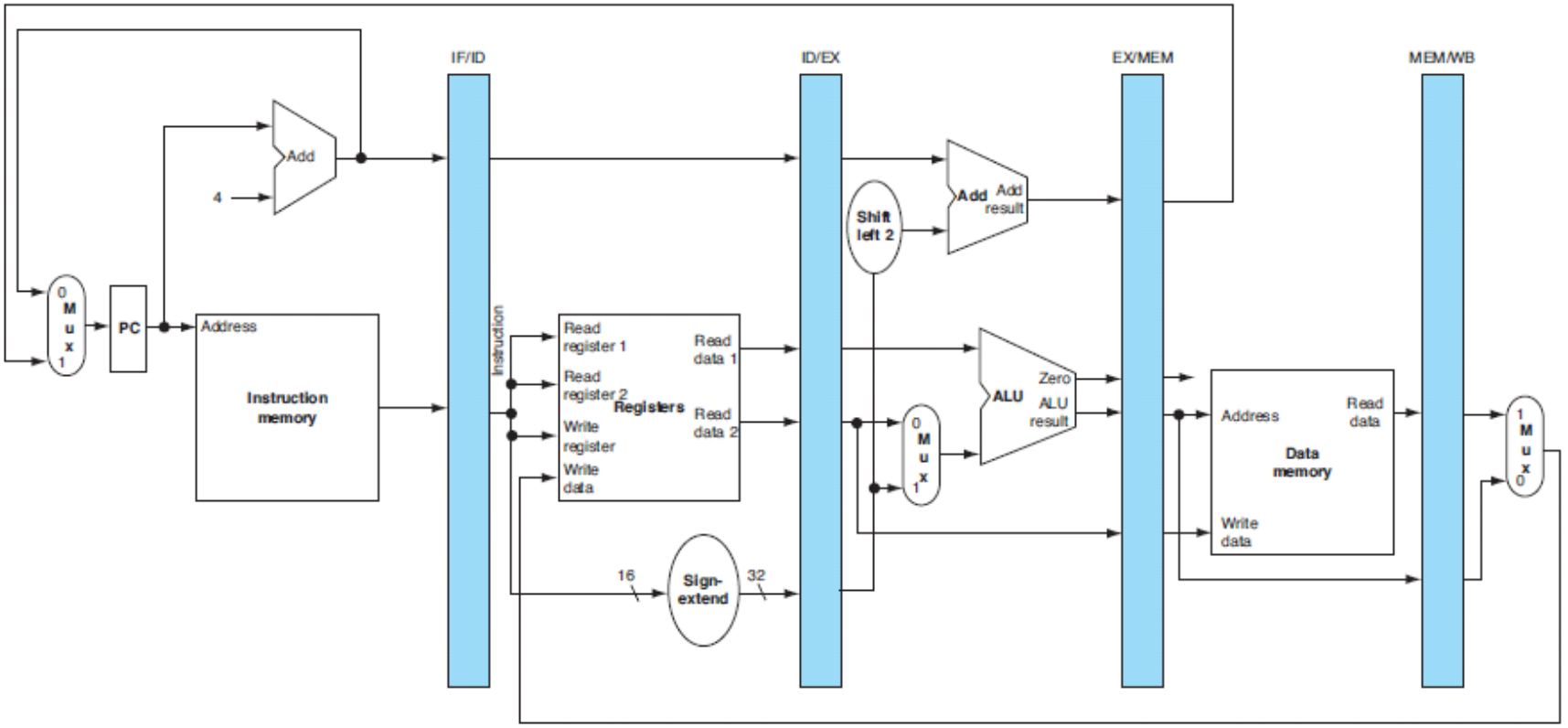
•یکی باعث ایجاد مخاطره داده و دیگری باعث ایجاد مخاطره کنترل می گردد.

نمایش اجرای خط لوله ای دستورات متوالی در خط لوله



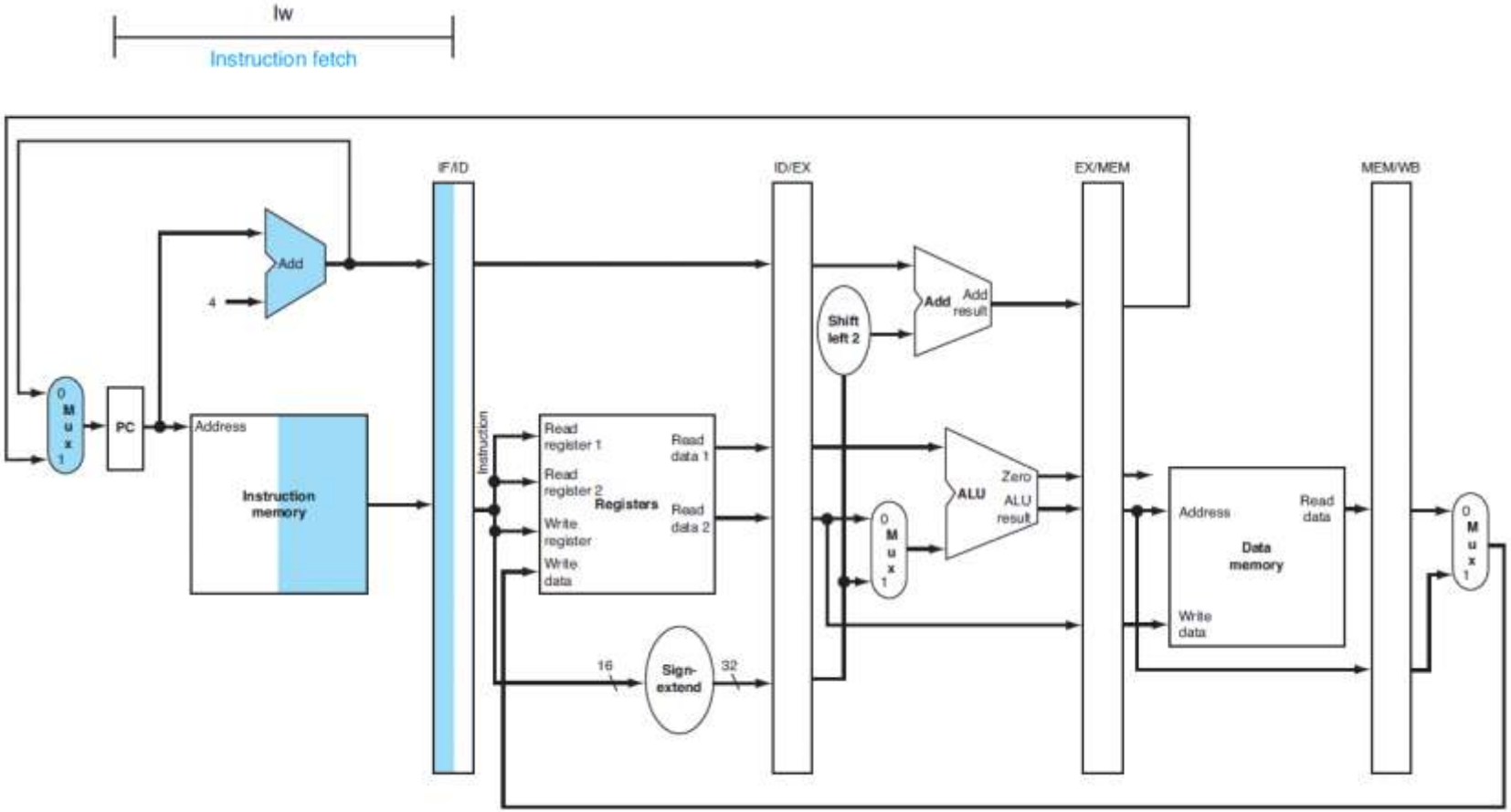
- ظاهر شکل طوری ترسیم شده که به نظر می رسد هر دستور مسیر داده مستقل خودش را دارد.
- عنوان روی هر مرحله، عنوان واحد عملیاتی در گیر است.
- برای جابجایی داده بین مراحل و حفظ نتیجه بین مراحل باید از ثبات استفاده کنیم.
- امکان نوشتن/خواندن توأم در ثباتها و بانک ثبات، با نوشتن در نیمه اول سیکل ساعت و خواندن آن در نیمه دوم سیکل میسر است.

استفاده از ثباتهای میانی

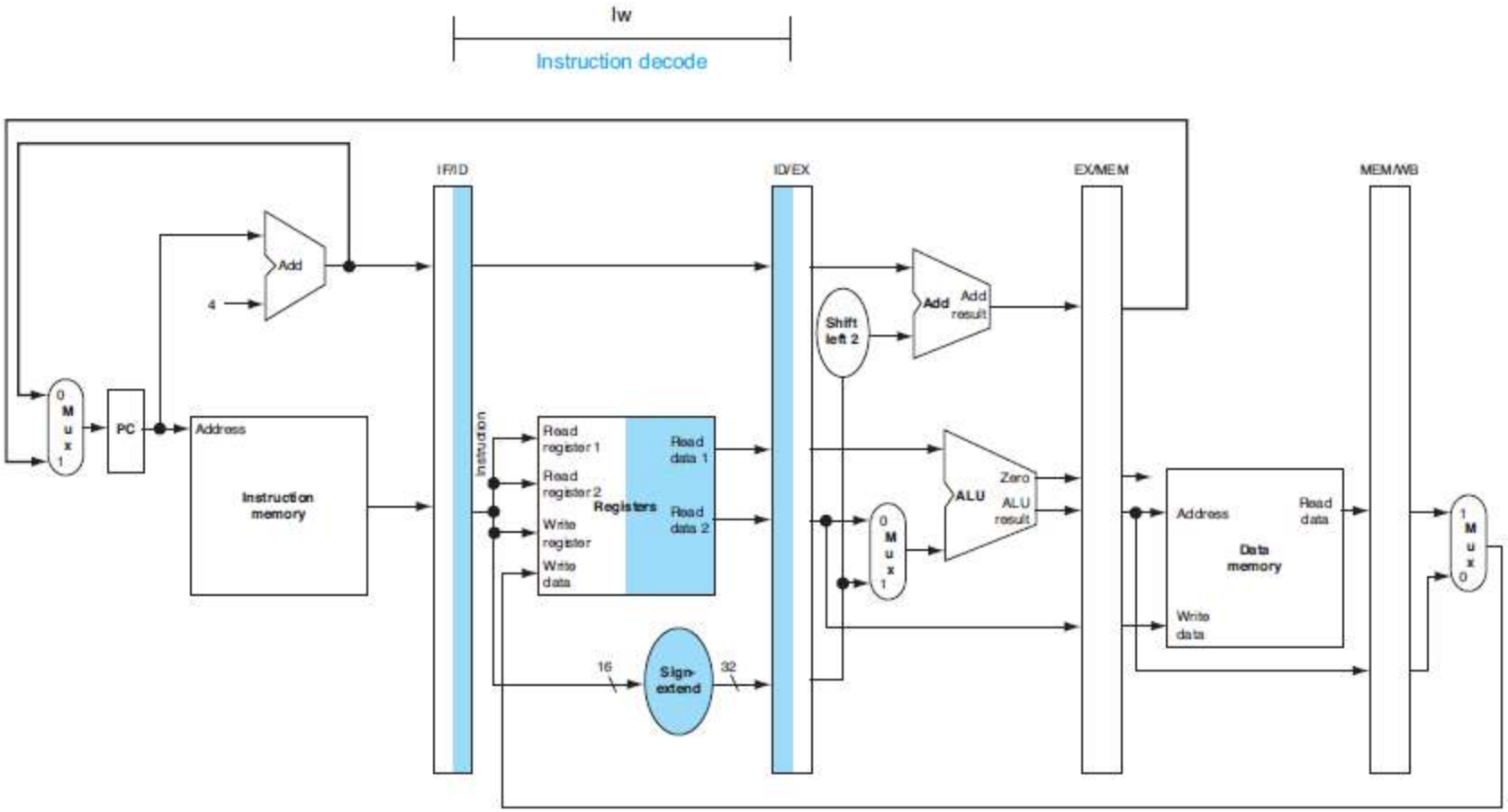


ثباتهای بین مرحله ای، به عنوان واسط برای جابجایی اطلاعات بین ایستگاه های خط لوله استفاده می شوند.

فرآیند اجرای دستور LW در مسیر داده

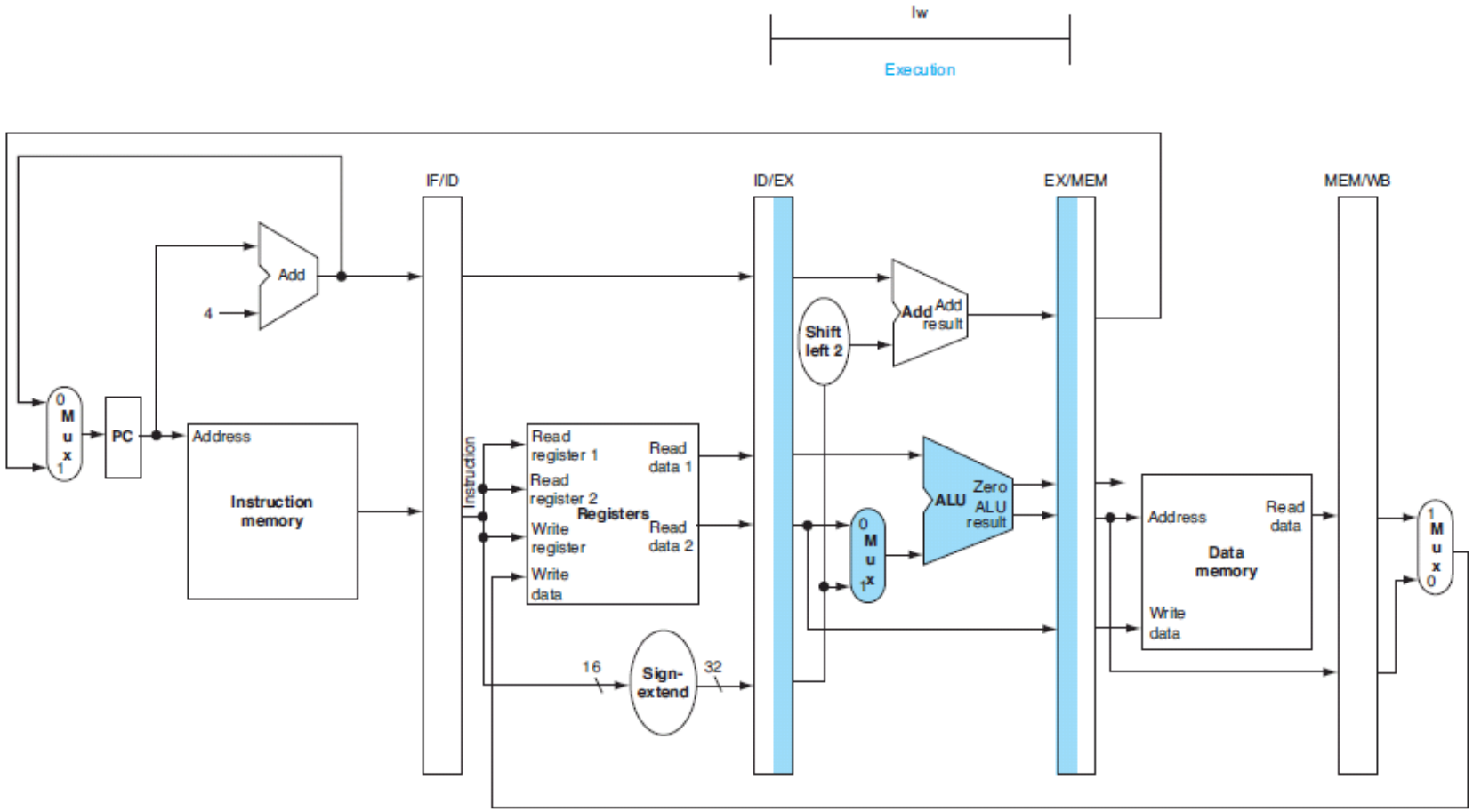


فرآیند اجرای دستور LW در مسیر داده

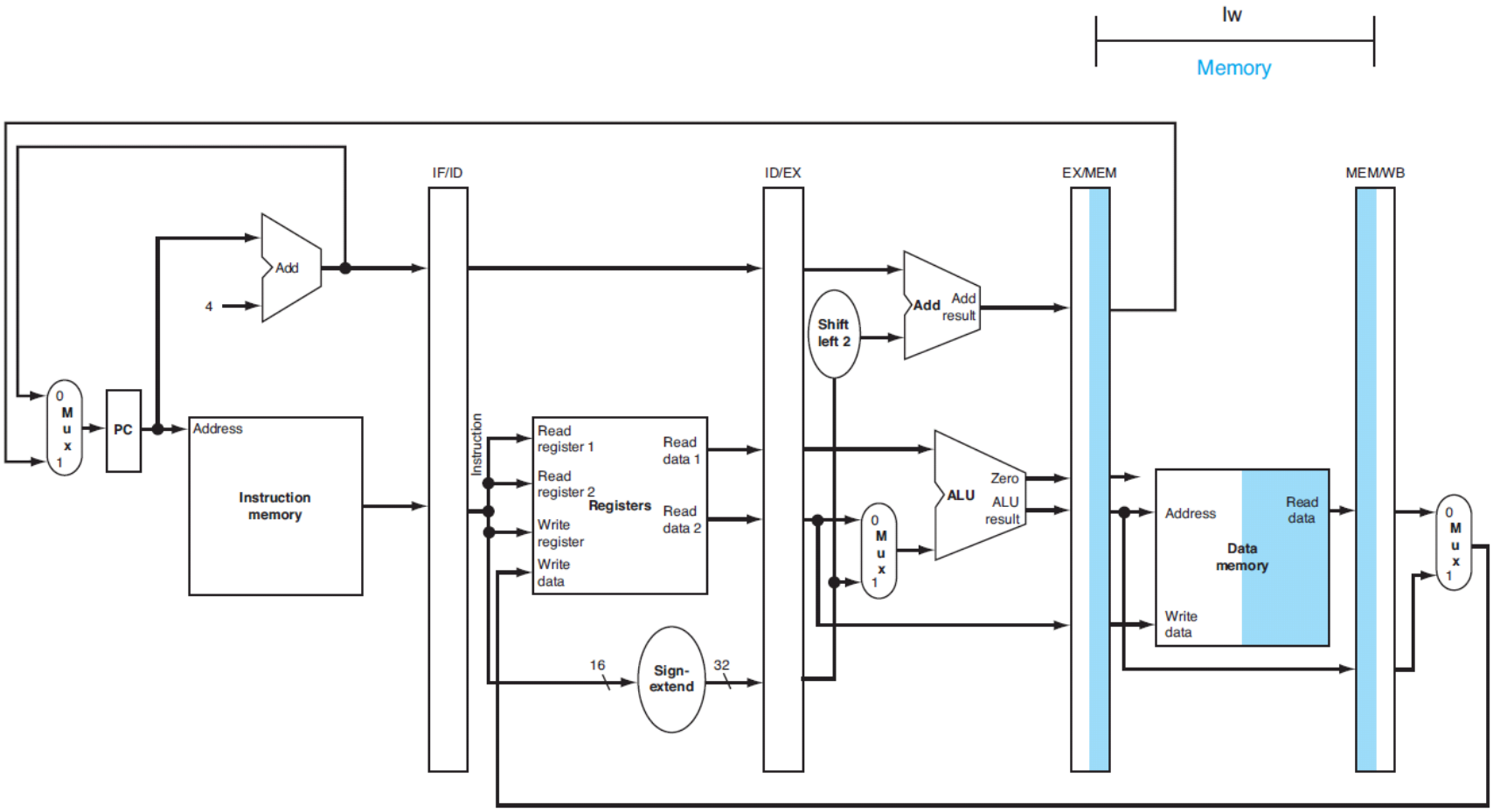


فرض شده تا شروع مرحله **Execute**(مرحله ۳)، نمی دانیم دستور چیست. لذا دو ثبات خوانده شده از بانک ثباتها، در ثباتهای میانی **ID/EX** ذخیره می شوند و در گام سوم مشخص می شود که لازم هستند و یا نه.

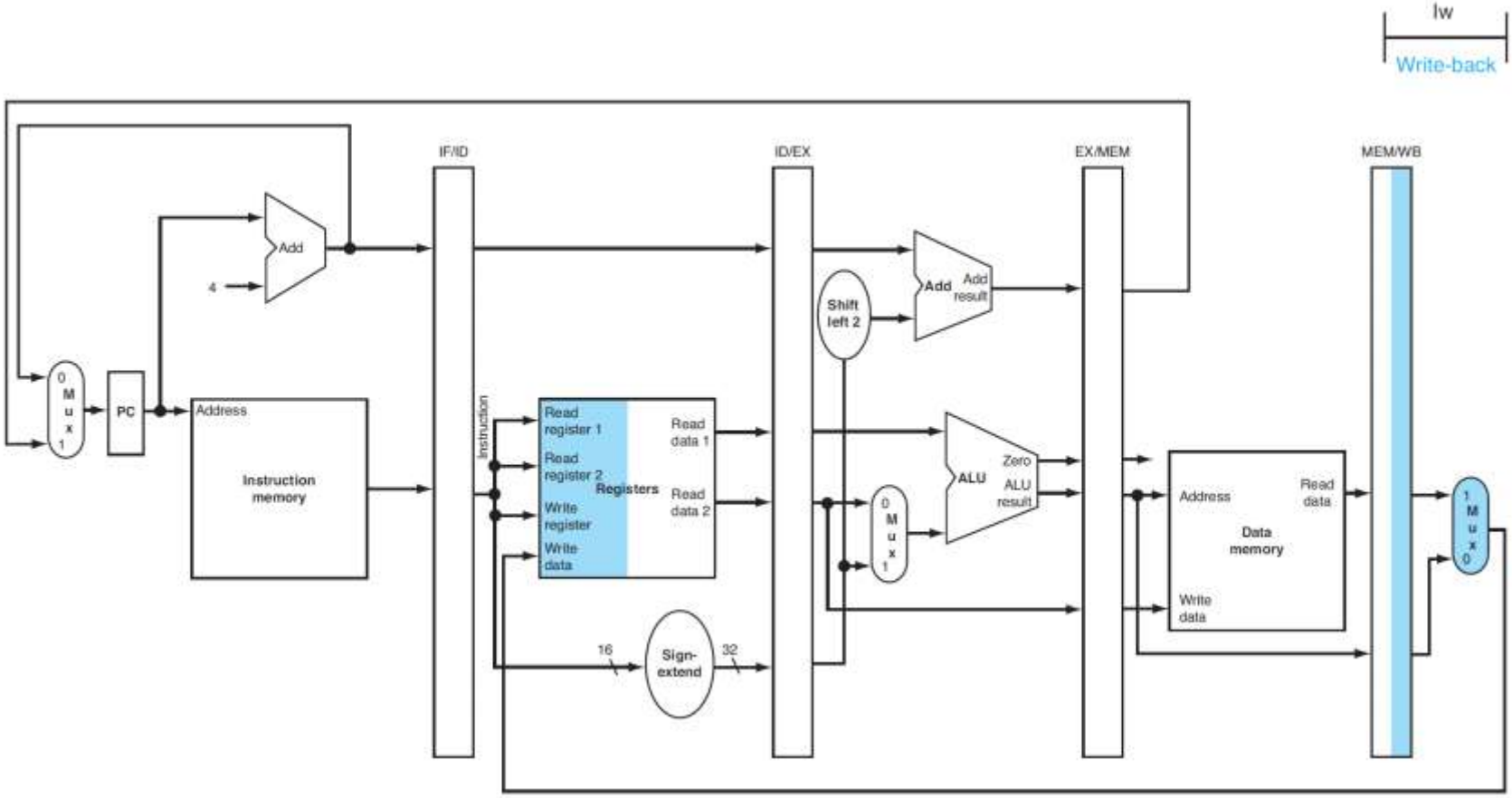
فرآیند اجرای دستور LW در مسیر داده



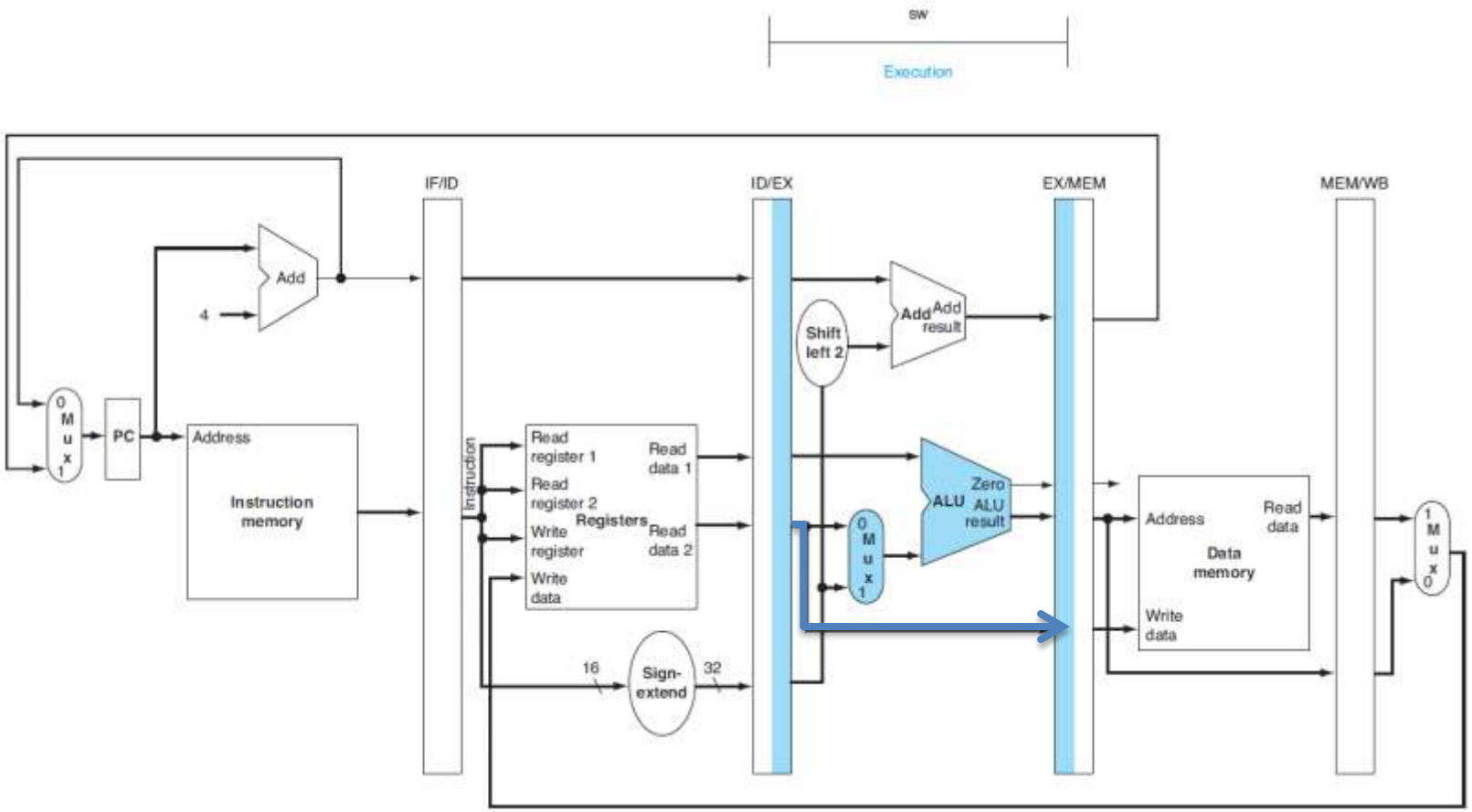
فرآیند اجرای دستور LW در مسیر داده



فرآیند اجرای دستور LW در مسیر داده



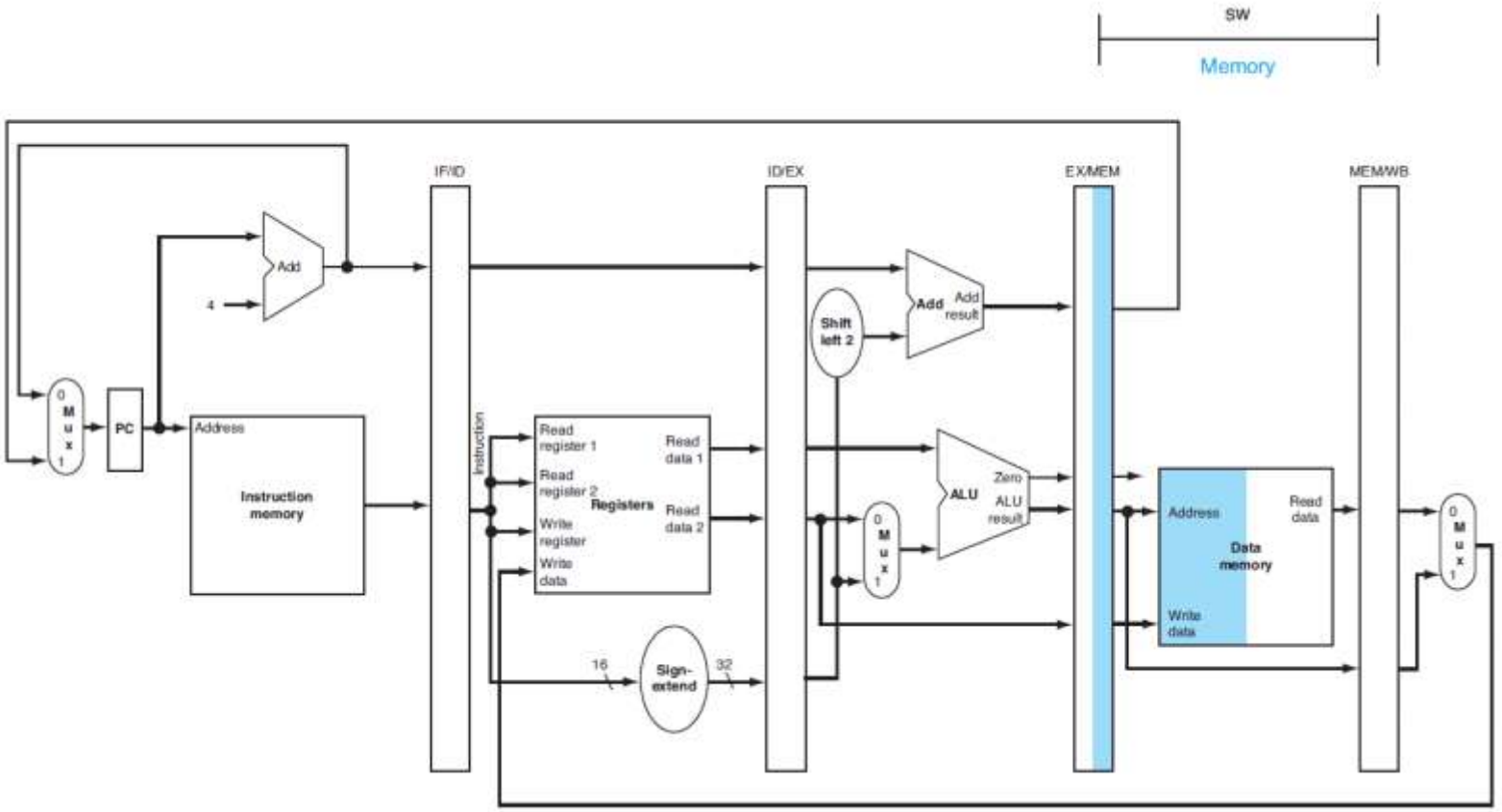
فرآیند اجرای دستور SW در مسیر داده



گامهای یک و دو (IF , ID) در دستور LW و SW مشابه هستند.

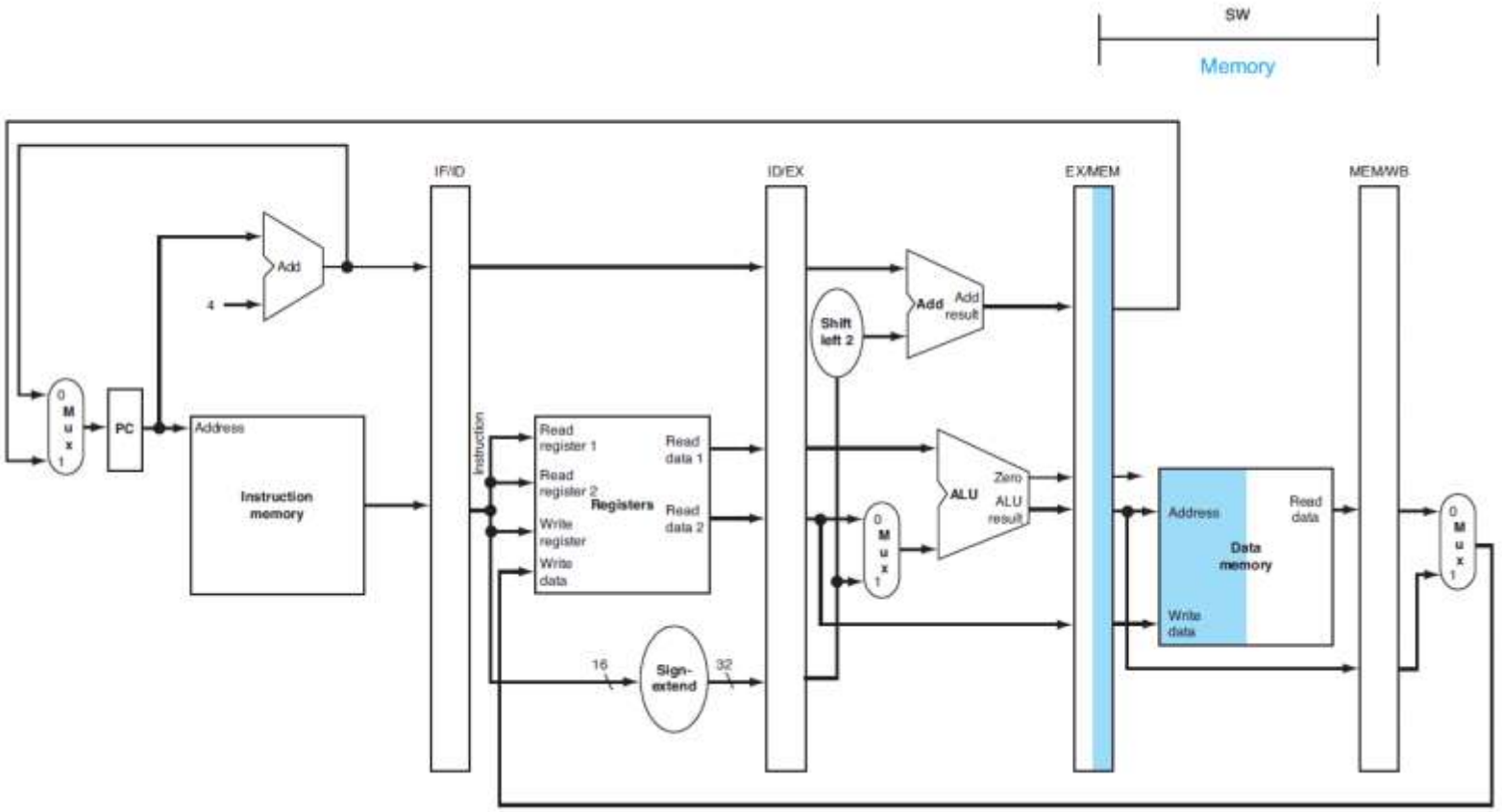
در گام سوم، علاوه بر نتیجه ALU جهت استفاده به عنوان آدرس مؤثر مقدار ثبات دوم خوانده شده در ثبات EX/MEM ذخیره می شود (برخلاف دستور LW) تا در گام بعد در حافظه نوشته شود

فرآیند اجرای دستور SW در مسیر داده



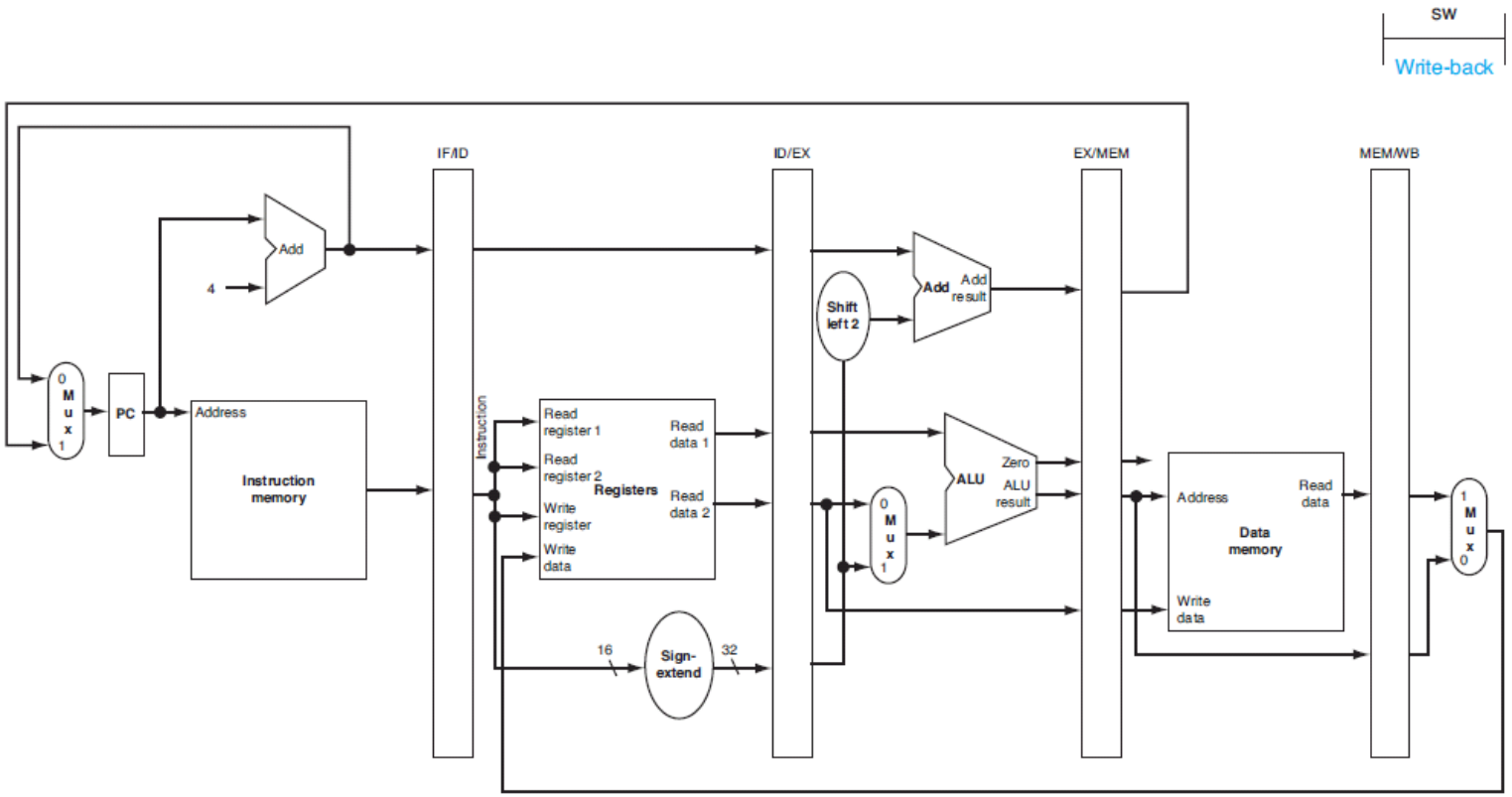
نوشتن ثبات در حافظه داده

فرآیند اجرای دستور SW در مسیر داده



نوشتن ثبات در حافظه داده

فرآیند اجرای دستور SW در مسیر داده



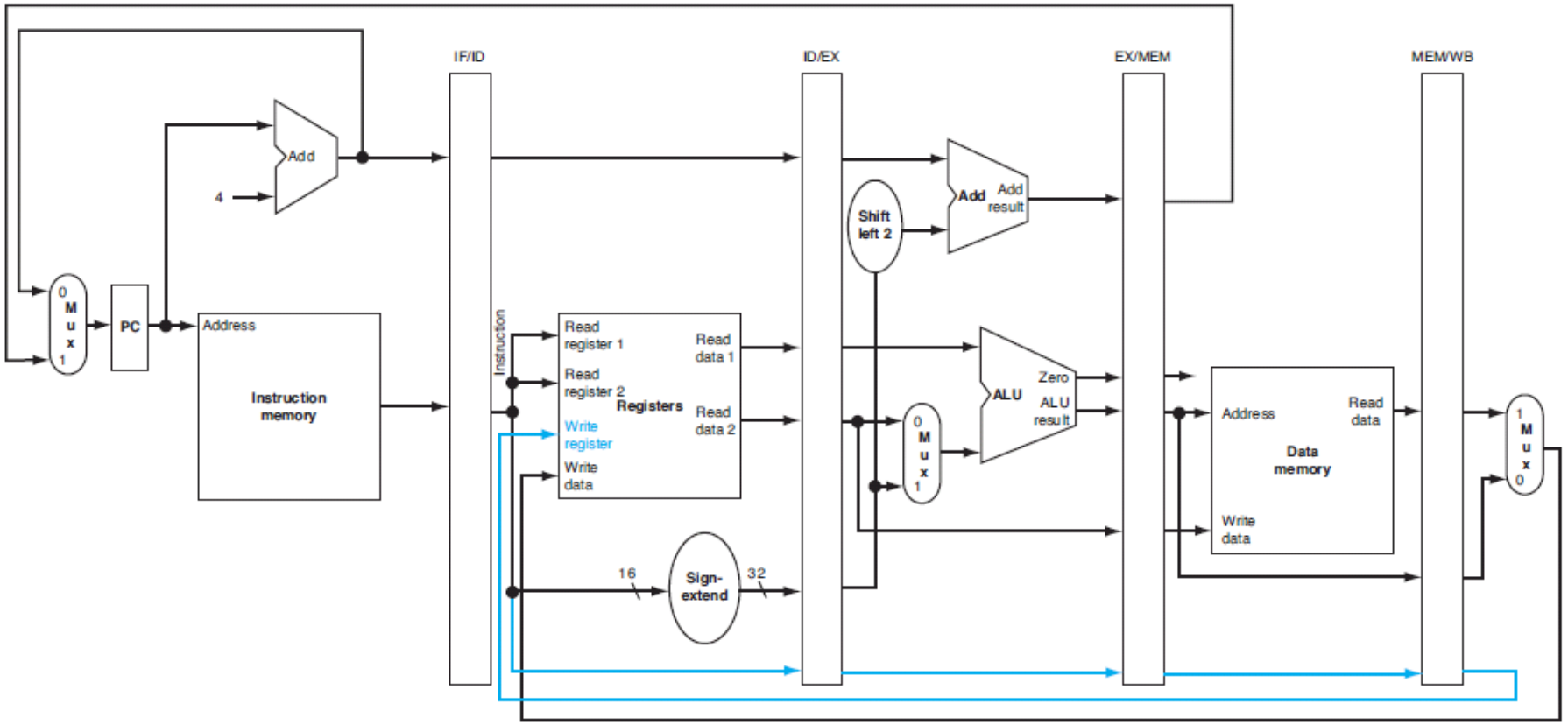
در این گام با توجه به خاتمه اجرای دستور **SW** و عدم نیاز به نوشتن مقداری در ثبات، هیچ عمل مؤثری رخ نمی دهد.

- در صورت عدم وجود ثباتهای میانی، با رفتن به سیکل ساعت بعد، اطلاعات مرحله قبل از بین می رود و واحد عملیاتی اطلاعاتی درست را دریافت نمی کند.
- هر یک از مؤلفه ها و مدارات منطقی و همچنین خطوط کنترلی آنها باید متناظر با یک مرحله از خط لوله باشند و عملیاتشان الزاماً باید در یک سیکل ساعت تکمیل شود. در غیر اینصورت در خلال اجرای دستورات با مخاطره ساختاری (Structural Hazard) مواجه خواهیم شد.

نکته!

- شماره ثباتی که در مرحله **WB** دستور **LW** در آن نوشته می شود چگونه تنظیم می شود؟
- در طراحی قبل این مقدار از مرحله ای در خط لوله تأمین می شود که در حال حاضر اطلاعاتش در ثباتهای **IF/ID** ذخیره شده است (به عبارتی ثباتی که در سیکل قبل **Fetch** شده است) نه دستور **LW** ای که الان اطلاعاتش در حال **WB** است!
- مقدار ثبات مقصد باید در طول مراحل نگهداری شود و در نهایت در هنگام **WB** مورد استفاده قرار گیرد.

اصلاح مسیر داده



در سه مرحله انتهایی خط لوله به ثباتهای خط لوله ۵ بیت اضافه می شود تا شماره ثبات مقصد را برای استفاده در ایستگاه **WB** حفظ کنند.