خط لوله

معماری کامپیوتر

۱۳۹۵-۱۱-۱۳

جلسه‌ی پانزدهم

دانشگاه شهید بهشتی

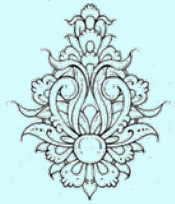
دانشکده‌ی مهندسی برق و کامپیوتر

بهار ۱۳۹۵

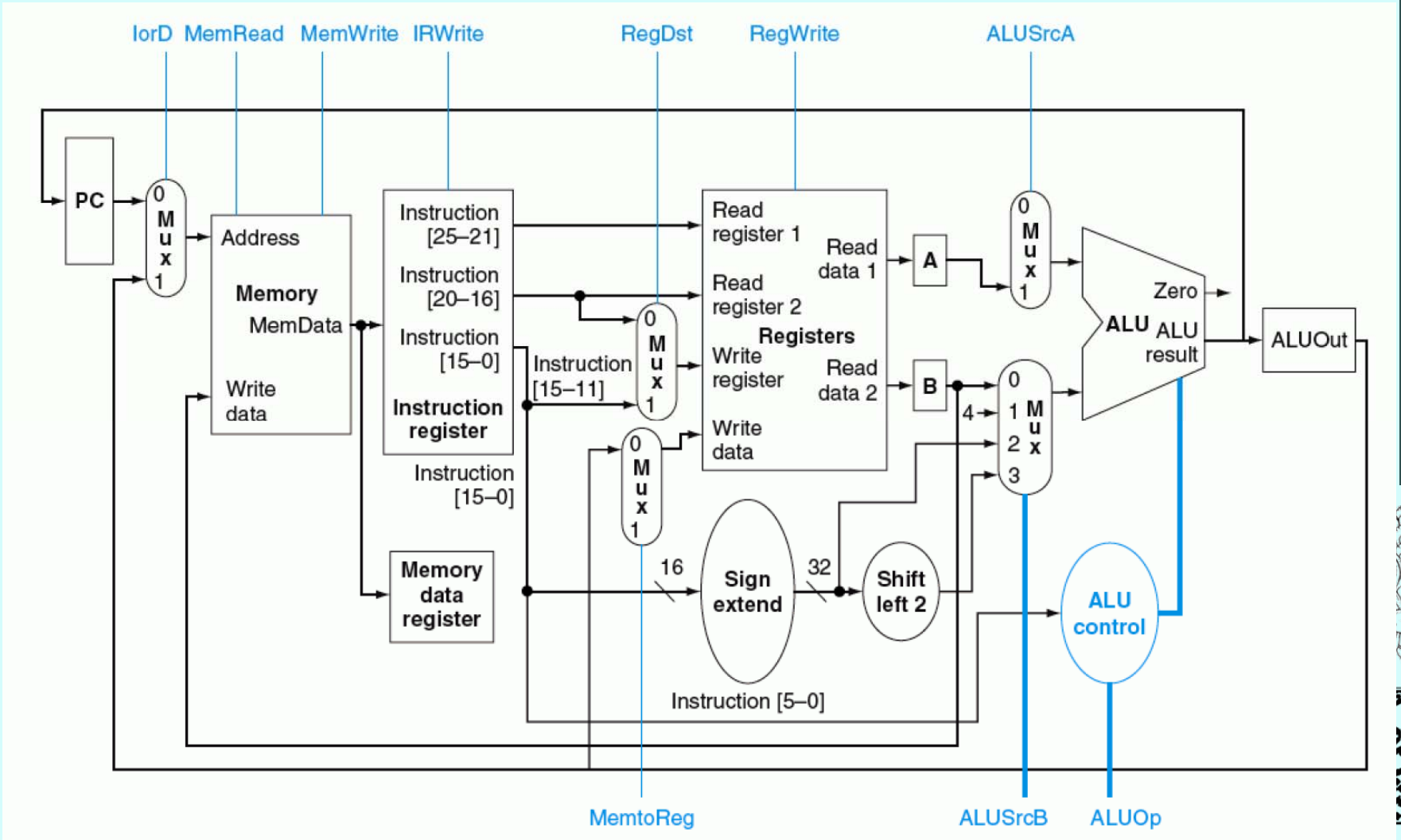
احمد محمودی ازناوه

فهرست مطالب

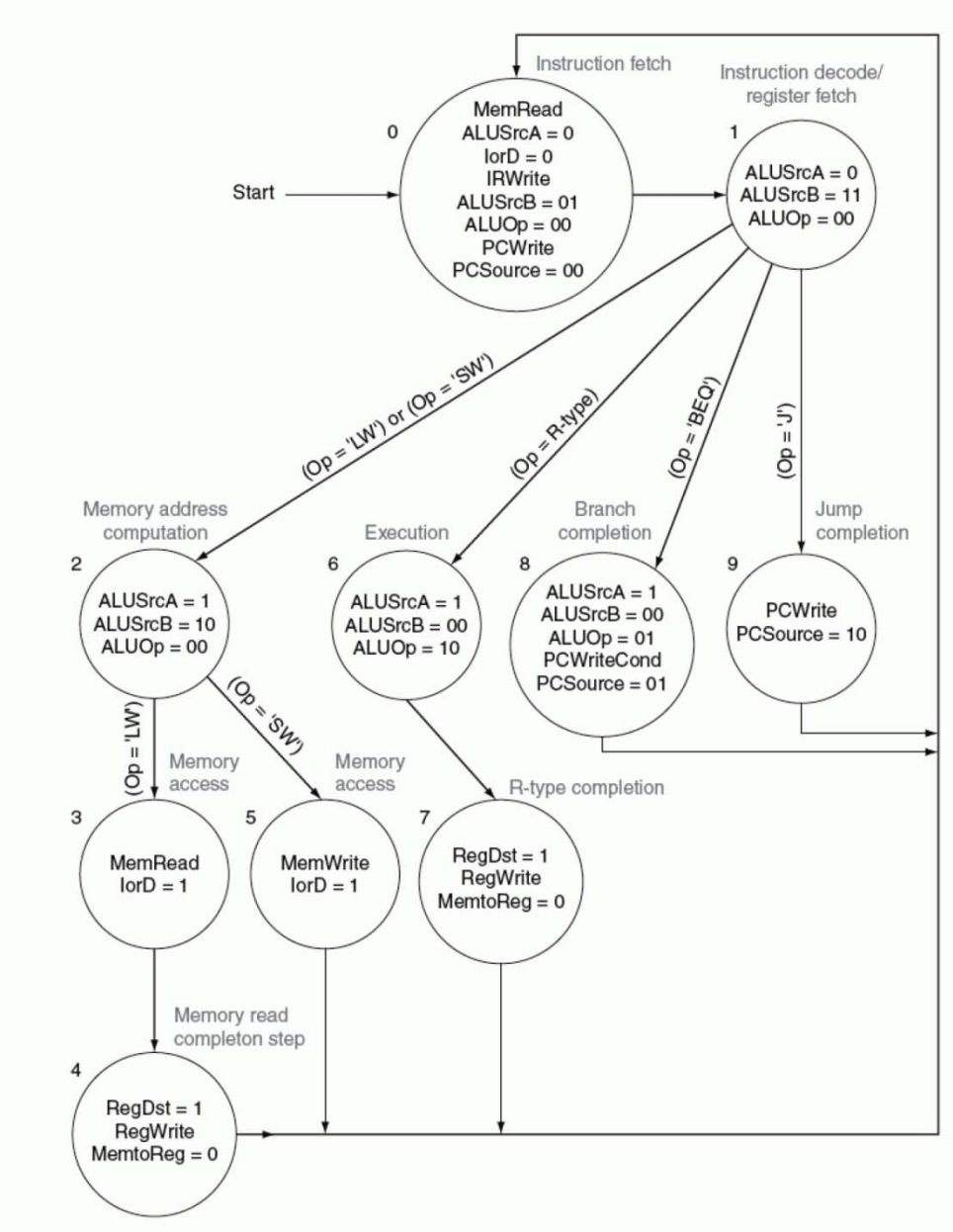
- مروری بر جلسه‌ی پیش
  - گذرداده‌ی چندسیکلی
- انواع واحد کنترل
- یادآوری خط لوله
- مسیر گذرداده‌ی خط لوله

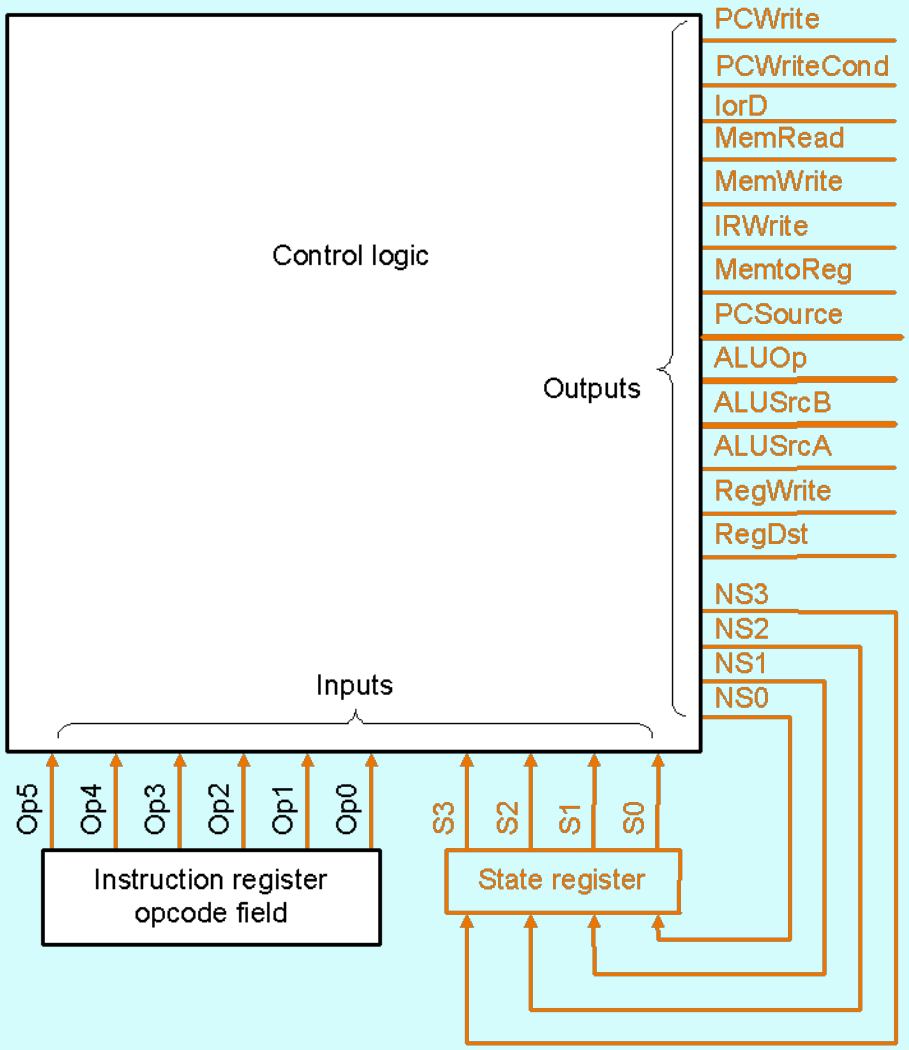


مسیر گذرداده‌ی چندسیکلی



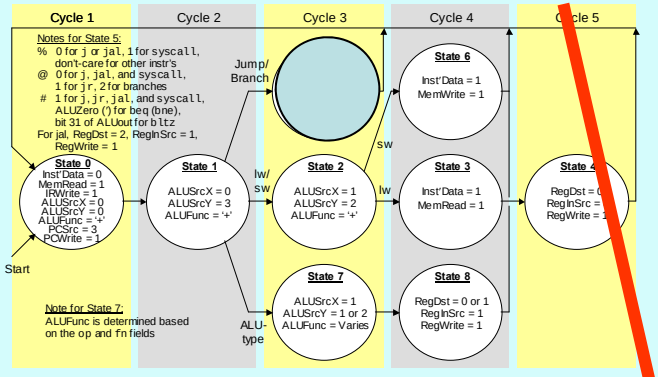
ماشین حالت واحد کنترل



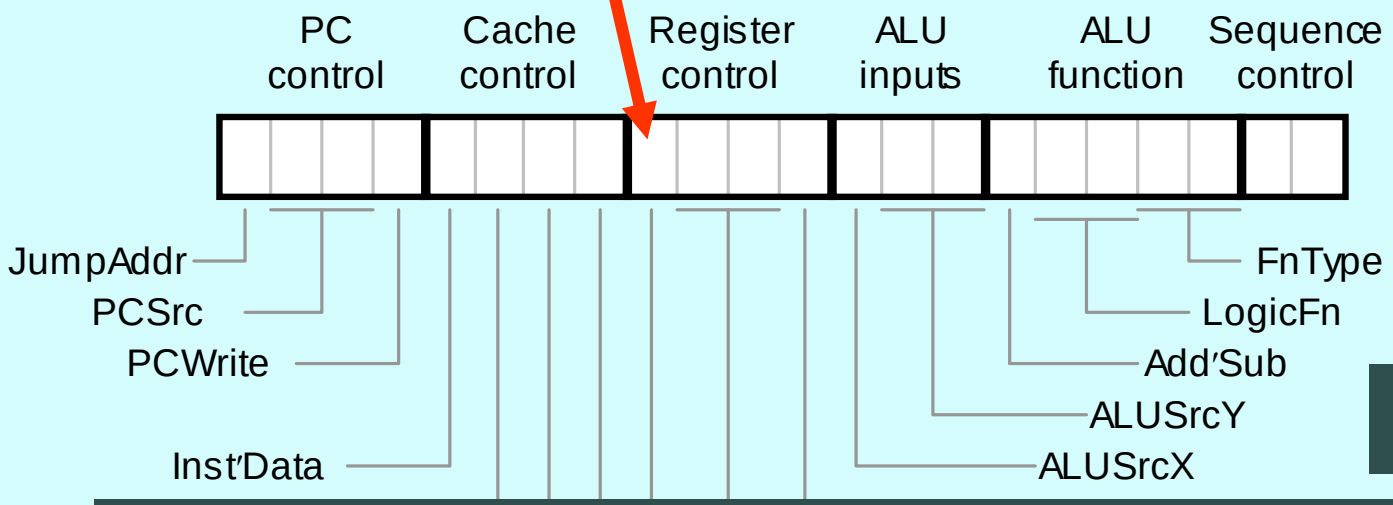

Control logic
Outputs
PCWrite
PCWriteCond
IorD
MemRead
MemWrite
IRWrite
MemtoReg
PCSource
ALUOp
ALUSrcB
ALUSrcA
RegWrite
RegDst
NS3
NS2
NS1
NS0
Inputs
Op5
Op4
Op3
Op2
Op1
Op0
S3
S2
S1
S0
Instruction register
opcode field
State register

Hard-wired

ریزبرنامه



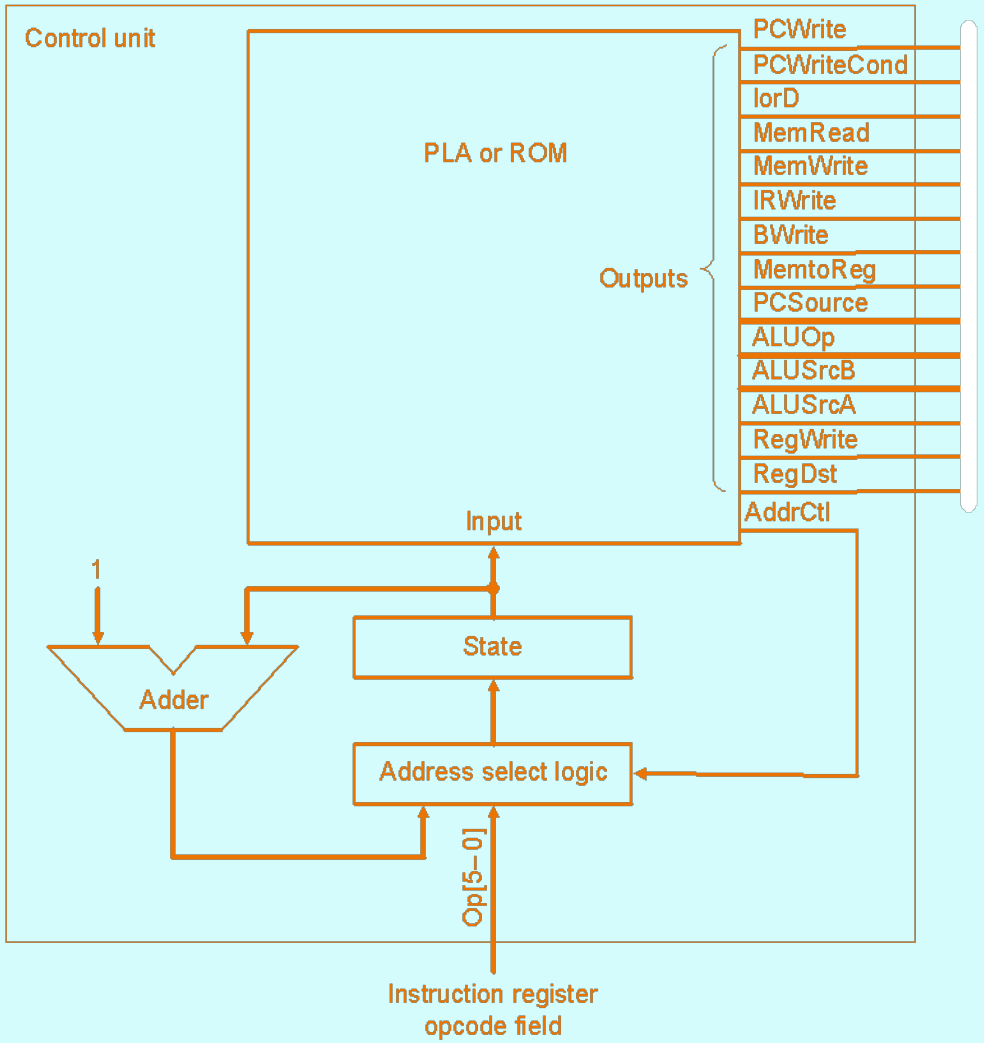
ماشین حالت استفاده شده برای خطوط کنترلی، خود شبیه به یک برنامه است!



microcode

«ریزدستور» گام‌های کوچک‌تری است که هر دستورالعمل طی آن انجام می‌شود.

پیاده‌سازی دیگری از واحد کنترل



microprogram

«ریزبرنامه» مجموعه‌ای از دستورها است که عملیات واحد پردازش مرکزی رایانه را کنترل می‌کند.

Hardwired vs Microprogrammed

Hard-wired

- در این شیوه برای پیاده‌سازی واحد کنترل از دروازه‌های منطقی استفاده می‌شود.
- طبعا کارایی بهتری دارد.
- برای به روزرسانی باید طراحی دوباره انجام شود، برای محصولاتی مانند تلفن همراه مناسب نیست.

Micro-programmed

- سیگنال‌های کنترلی به صورت داده‌های متوالی ذخیره می‌شوند.
- در هر سیکل از حافظه خوانده می‌شوند.
- با تغییر محتوای حافظه می‌توان به ISA بخش‌های جدیدی افزود.

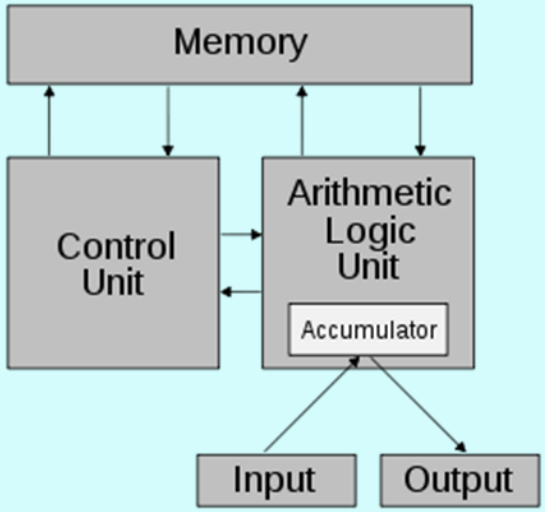
معماری Harvard

- 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.

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



CISC در مقابل RISC

RISC

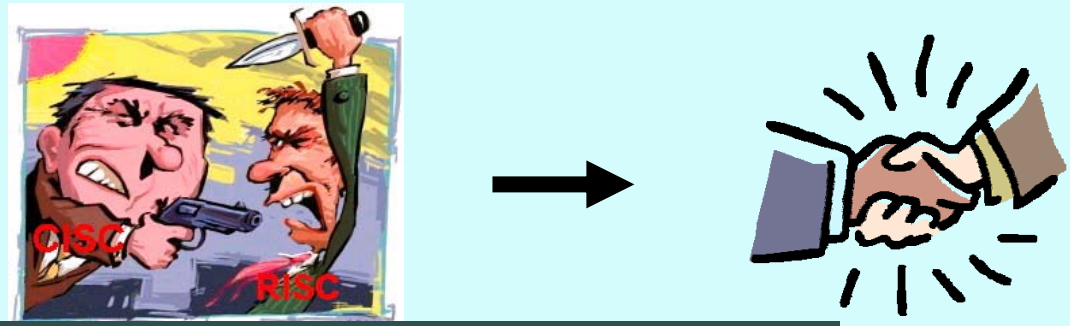
- تعداد دستورات کم
- طول دستورات ثابت
- هزینه‌ی پایین
- تنها دستورات خواندن و نوشتن به حافظه دسترسی دارند
- همه‌ی عملوندها ثبات‌های پردازنده هستند
- مودهای آدرس محدود
- واحد کنترل به صورت سیم‌بندی

- 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

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.



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

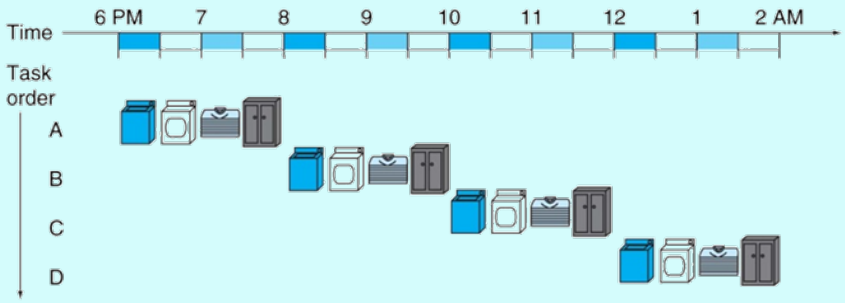
- تا پیش از پردازنده‌های ۸۰۴۸۶، اجرای دستورات به صورت ترتیبی در شش مرحله صورت می‌پذیرفت. در پردازنده‌های ۸۰۴۸۶ این شش مرحله به صورت خط لوله در آمد.

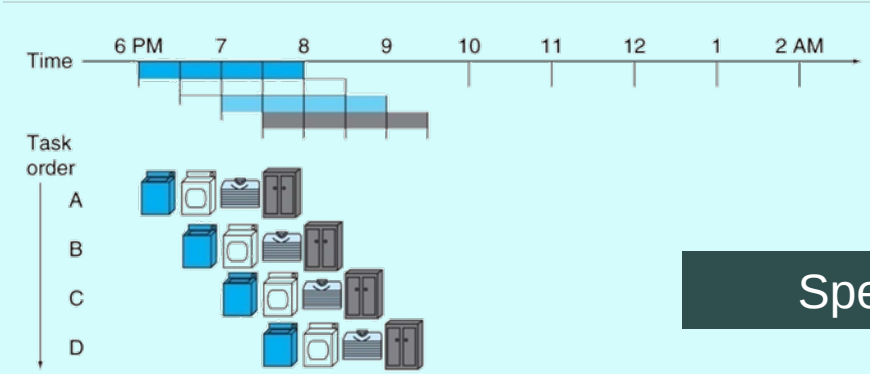
مروری بر خط لوله

- 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

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

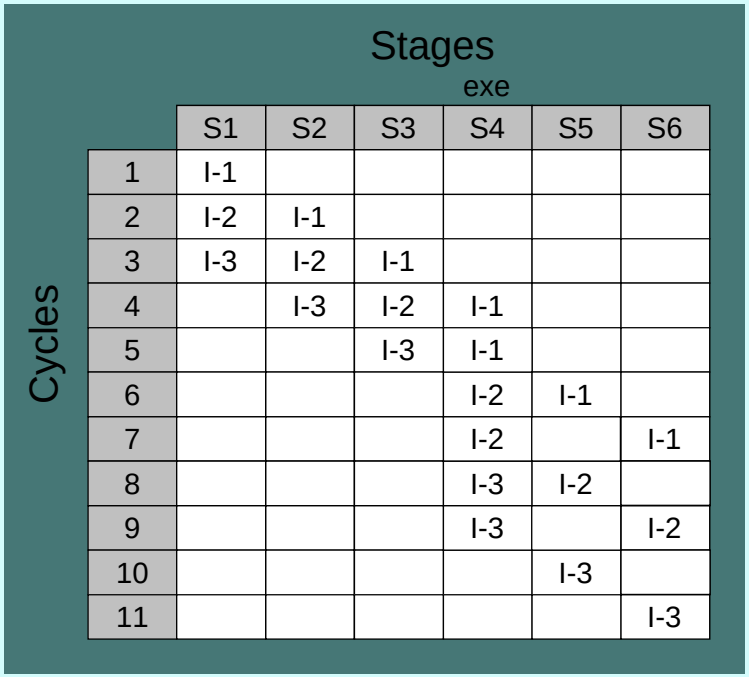
در یک خط لوله‌ی k مرحله‌ای، برای انجام n دستور چند سیکل ساعت زمان لازم است؟

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

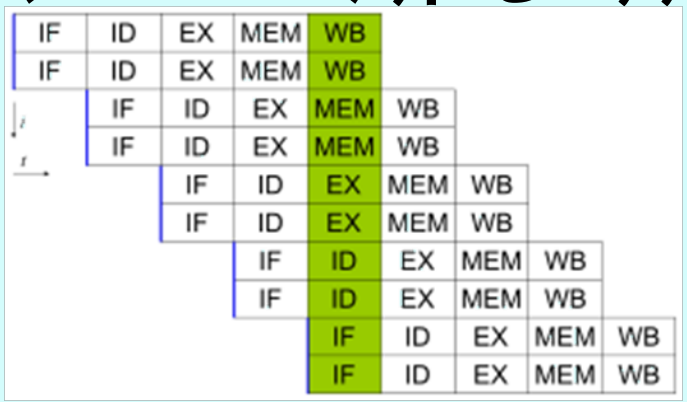


	S1	S2	S3	S4	S5	S6
1	I-1					
2	I-2	I-1				
3	I-3	I-2	I-1			
4		I-3	I-2	I-1		
5			I-3	I-1		
6				I-2	I-1	
7				I-2		I-1
8				I-3	I-2	
9				I-3		I-2
10					I-3	
11						I-3

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

k + (2n – 1)

- در سوپراسکالر چندین خط لوله برای اجرای دستورالعمل‌ها وجود دارد. بدین‌ترتیب چند دستور همزمان واکشی می‌شود. Pentium اولین پردازنده از خانواده‌ی X86 بود که مجهز به سوپراسکالر با دو خط لوله شد، پس از آن نوبت به Pentium Pro با سه خط لوله.
- وابستگی بین دستورالعمل ها به صورت سخت‌افزاری در زمان اجرا(run time)دائما چک می‌شود.



سوپراسکالر (ادامه...)

- در مثال قبل واحد S4 را به صورت افزونه در خط لوله قرار می‌دهیم، دستورات با شماره فرد وارد واحد u و دستوالعمل‌ها با شماره‌ی زوج وارد واحد v خواهند شد.

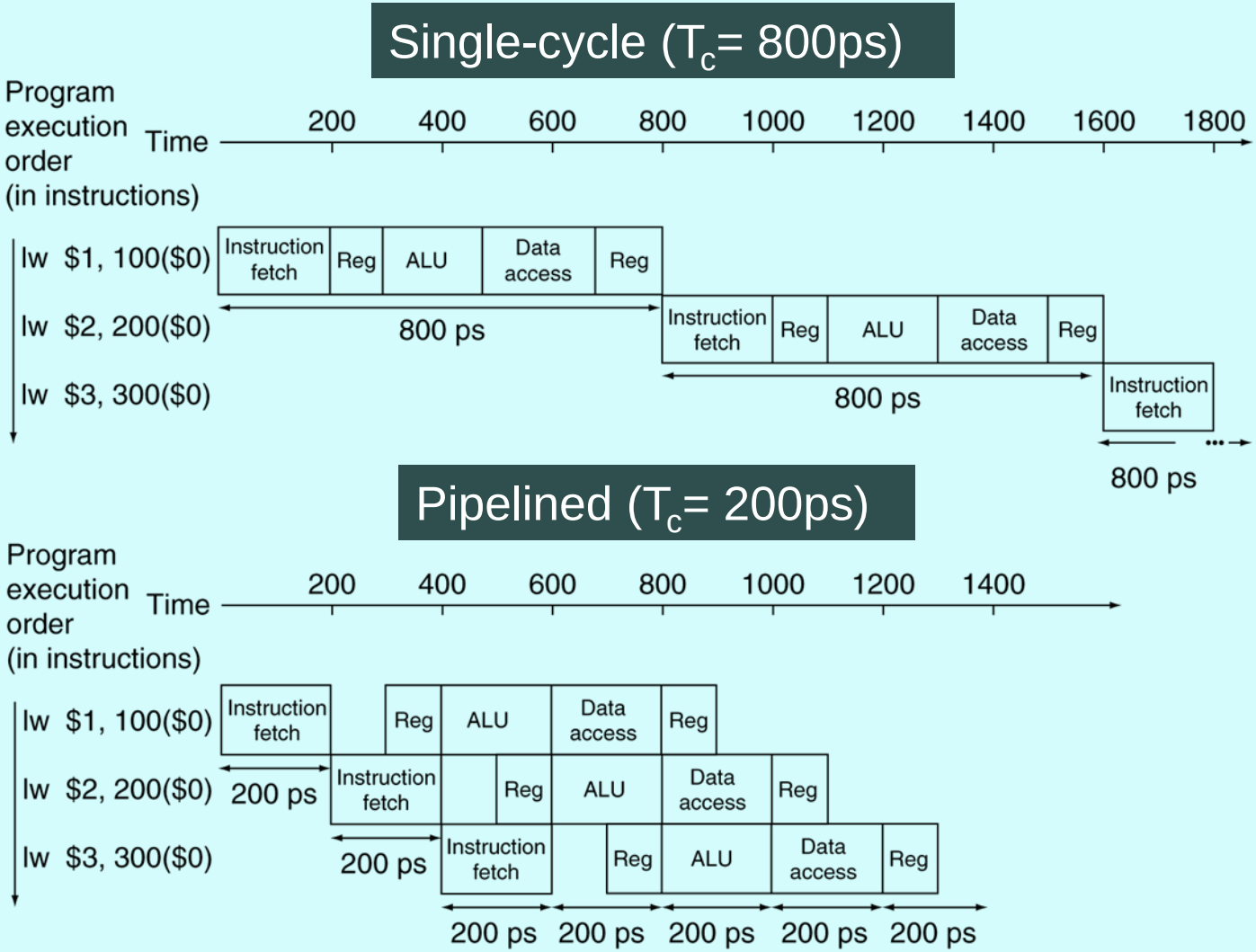
Stages

Cycles	S1	S2	S3	S4: u	S4: v	S5	S6
1	I-1						
2	I-2	I-1					
3	I-3	I-2	I-1				
4	I-4	I-3	I-2	I-1			
5		I-4	I-3	I-1	I-2		
6			I-4	I-3	I-2	I-1	
7				I-3	I-4	I-2	I-1
8					I-4	I-3	I-2
9						I-4	I-3
10							I-4

$k + n$

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

کارایی خط لوله (ادامه...)

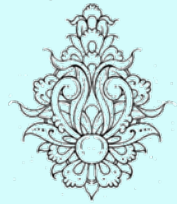


کارایی خط لوله (ادامه...)

- 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



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

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

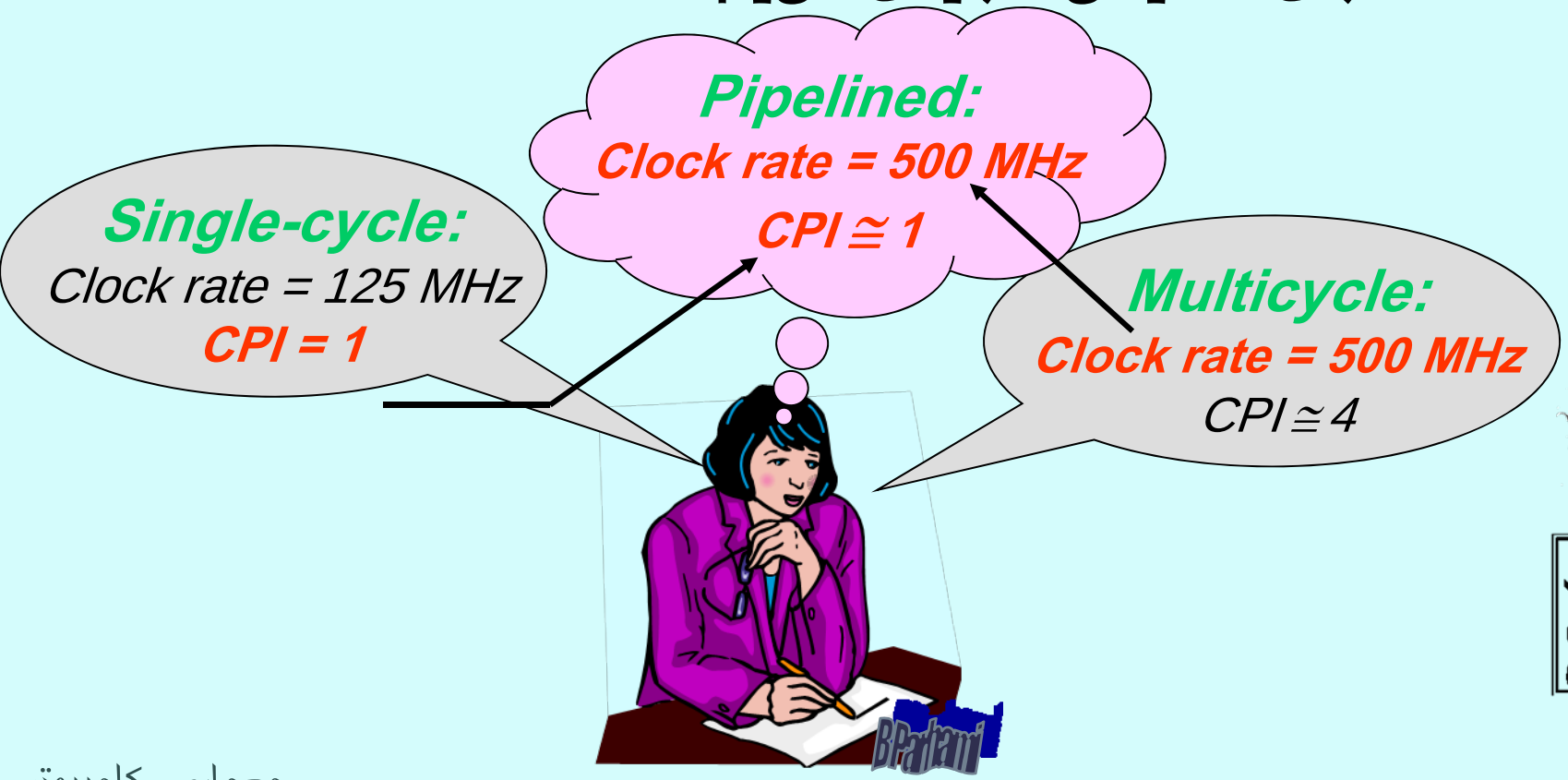
1,000,000X200ps+1400ps

کارایی خط لوله (ادامه...)

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

Latency

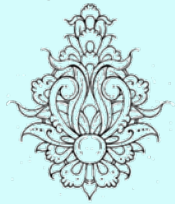
throughput



طراحی ISA برای خط لوله

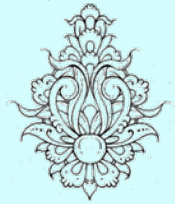
- تمام دستورات تمام دستورات MIPS طول یکسانی دارند. مرحله‌ی واکشی به سادگی انجام می‌شود.

- 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.



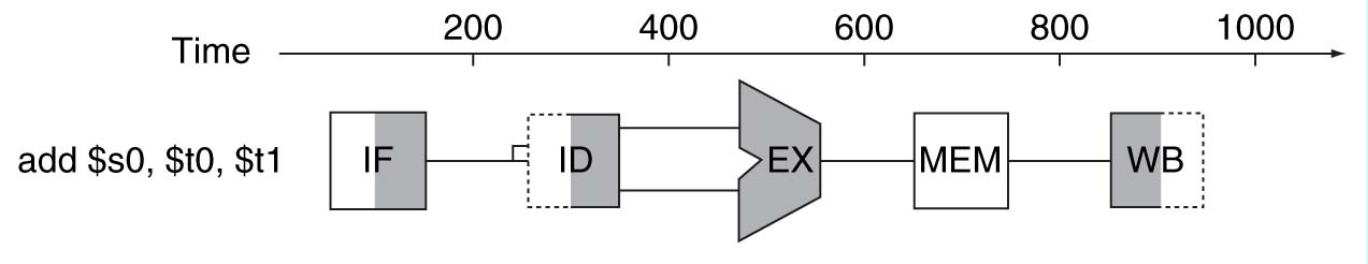
طراحی ISA برای خط لوله (ادامه...)

- 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

- گاهی اوقات در مسیر خط لوله شرایطی پیش می‌آید که دستورالعمل بعدی را نمی‌توان در سیکل ساعت بعدی اجرا کردو 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 بخش‌هایی که در سمت راست (چپ) سایه خورده‌اند، بیانگر خواندن (نوشتن) داده از (در) حافظه می‌باشند.

مخاطرات خط لوله!!

- در یک برنامه، ممکن است دستور بعدی قابل اجرا در پالس بعدی نباشد، چنین حوادثی را مخاطره می‌نامند.

Structural Hazard

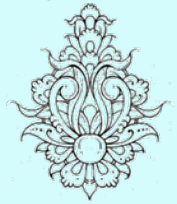
– مخاطرات ساختاری
  - یکی از منابع مورد نیاز مشغول است

Data Hazard

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

Control Hazard

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



مخاطرات ساختاری

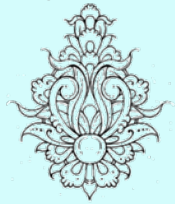
- اولین مانع در خط لوله به «مخاطرات ساختاری» شهرت دارد و بدین معناست که سخت‌افزار بی‌نفسه قادر به پشتیبانی ترکیبی خاص از دستورالعمل‌هایی که می‌خواهیم در یک سیکل واحد آن‌ها را اجرا کنیم، نیست.

stall

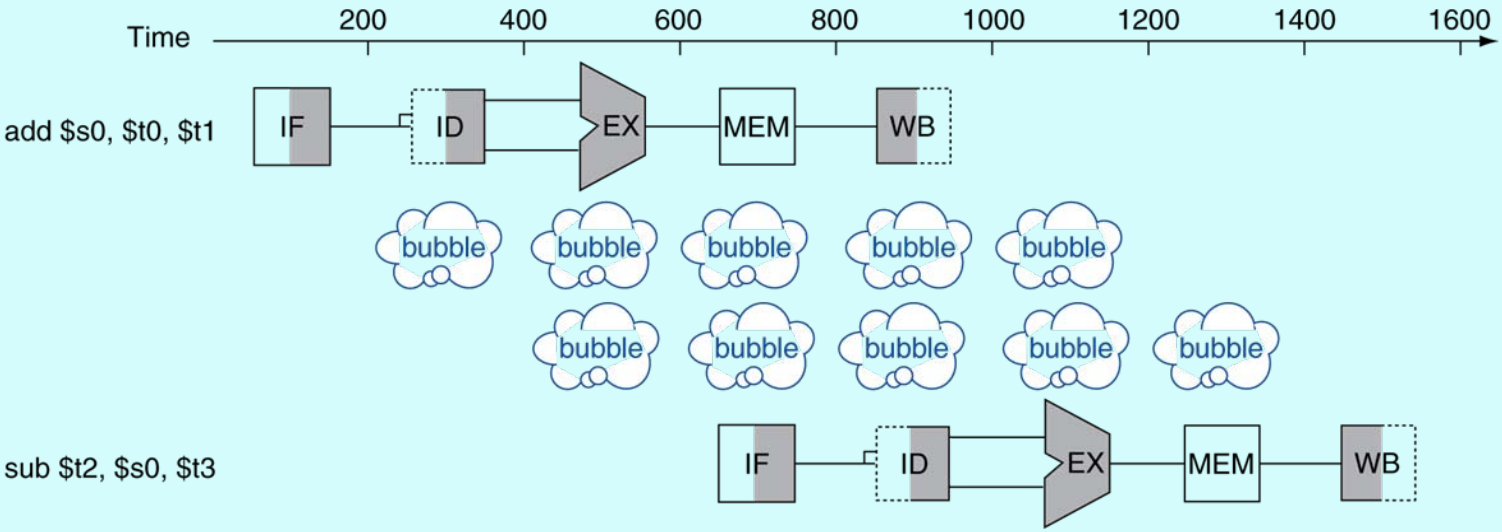
- تعارض در هنگام استفاده از منابع مشترک
  - در MIPS با یک حافظه برای دستورالعمل‌ها و داده‌ها
    - دستورات خواندن و نوشتن به حافظه احتیاج دارند
    - در این صورت برای واکشی دستورات بعدی، تعلیق می‌شود.
  - در چنین حالاتی به جای دستورات بعدی حباب وارد خط لوله می‌شود.

Bubble

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



تعلیق

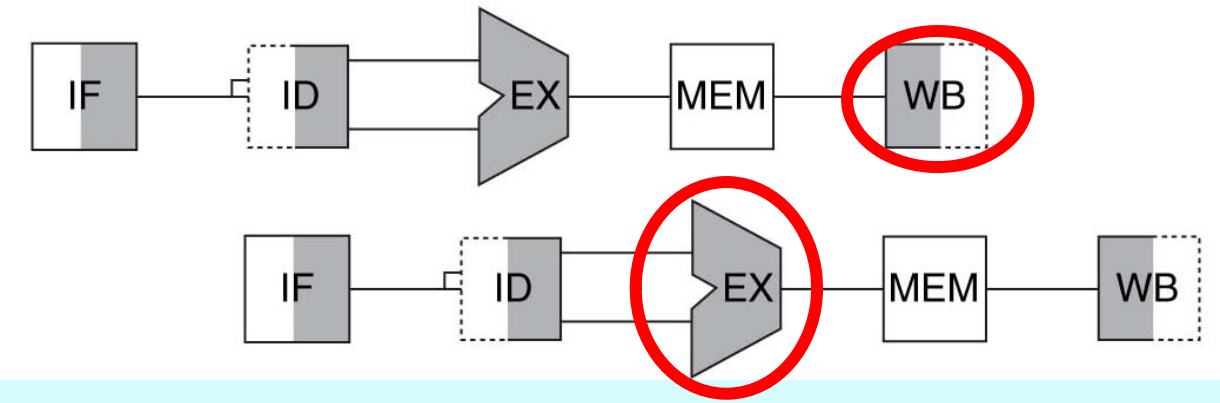


- این شکل یکی از مفاهیم بسیار با اهمیت در ایجاد خط لوله را به تصویر کشیده است که اگرچه عنوان رسمی «تعلیق خط لوله» دارد ولیکن اغلب اوقات از این پدیده با نام ساده‌تر «حباب» یاد می‌شود. در ادامه‌ی بحث به موانع منجر به «تعلیق» در بخش‌های دیگری از خط لوله خواهیم پرداخت.

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

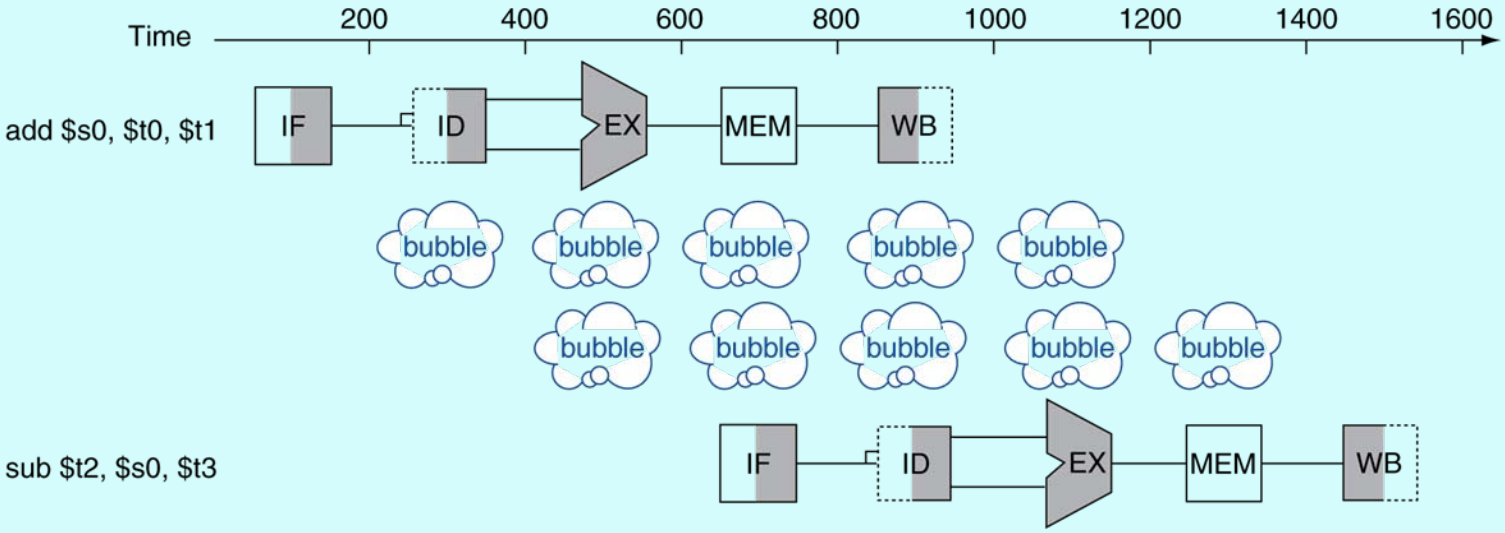
```
add $s0, $t0, $t1
sub $t2, $s0, $t3
```



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
```



پیش فرستادن

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