بنام خداوند جان و خرد

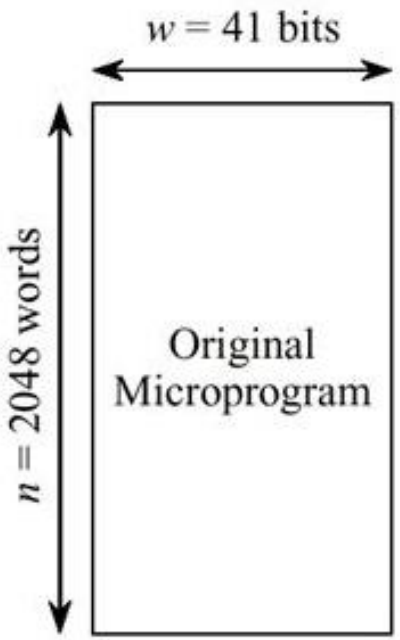
معماری کامپیوتر

جلسه بیست‌وششم: واحد کنترل نانو- خط‌لوله

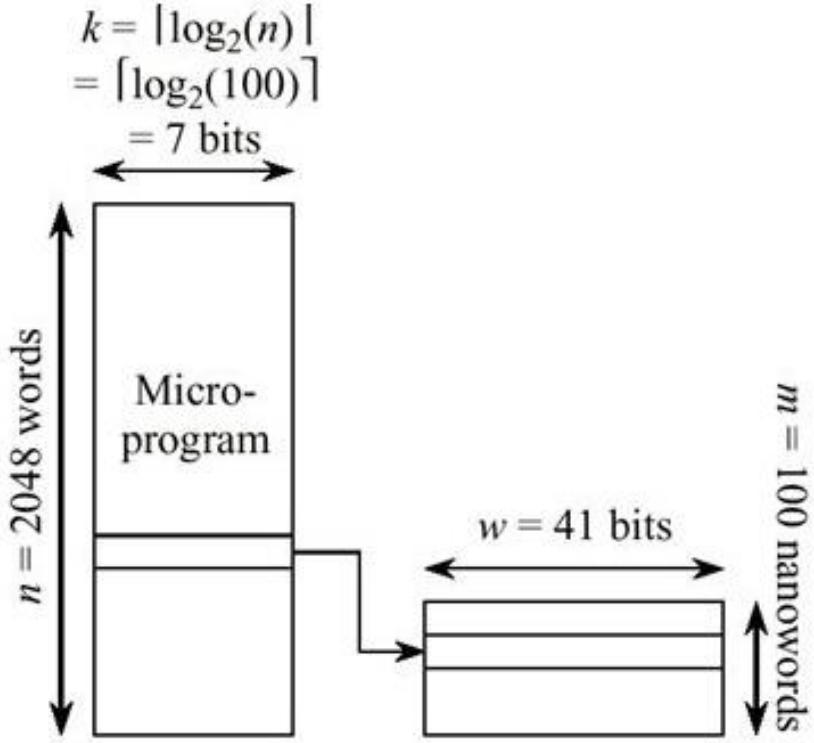
واحد کنترل Nano-programmed

- روش ریزبرنامه‌پذیر انعطاف بالا و تاخیر زیاد داشت
  - با پیچیده شدن دستورات تعداد بیت پیاده‌سازی ریزعملگرها و حجم CM زیاد می‌شود
- راهکار کاهش دادن تاخیر: استفاده از شیوه طراحی Nano-programmed
  - یک حافظه در داخل CM تعریف می‌کند که شامل ریزعملگرهای تعریف شده باشد
  - حافظه جدید فقط شامل ریزعملگرهایی است که استفاده می‌شوند پس تعداد بیت کمتری دارد (Nano-ROM)
    - عرض بیشتر و تعداد خطوط کمتر نسبت به حافظه میکرو
  - حافظه کنترلی کوچک شده و فقط شامل ایندکس اشاره به مکان ریزعملگر در حافظه جدید است (Micro-ROM)
    - عرض کمتر و تعداد خطوط بیشتر نسبت به حافظه نانو

Nano-programmed واحد کنترل



Micro-Programmed طراحی



Nano-Programmed طراحی

واحد کنترل Nano-programmed

- مثال: ریزبرنامه با ۲۰۴۸ ریزدستورالعمل ۲۰۰ بیتی تعریف شده است. اگر از این ۲۰۴۸ ریزدستورالعمل، ۲۵۶ تایشان متمایز باشد، حجم حافظه کنترلی در حالت ریزبرنامه‌ریزی شده و نانو برنامه‌ریزی شده چقدر است؟

Nano-programmed واحد کنترل

- حل:
  - حالت ریز برنامه‌ریزی شده

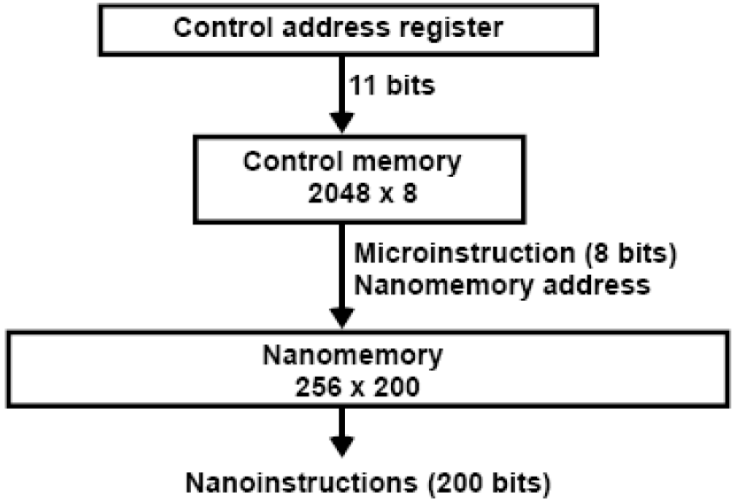
$CM_1 = 2048 * 200 = 409600$ bits

  - حالت نانوبرنامه‌ریزی شده

NCM (level 2) = 256 * 200 = 51200 bits

MCM (level 1) = 2048 * log(256) = 2048 * 8 = 16384 bits

Total $CM_2 = 51200 + 16384 = 67584$ bits



خط لوله – Pipeline

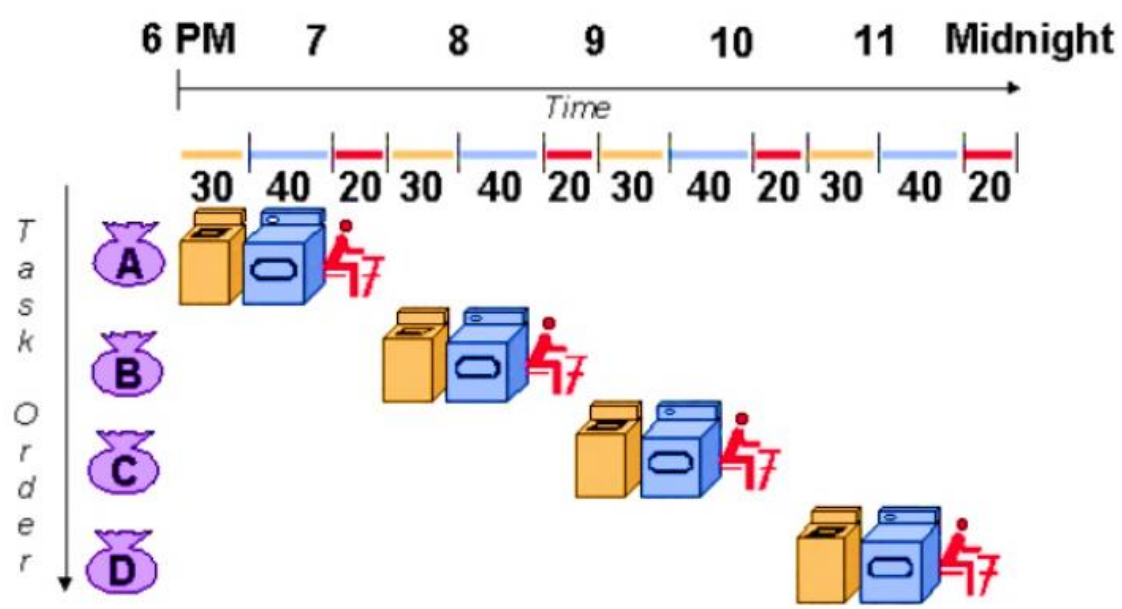
- تا به‌اینجا با طراحی اجزای مختلف پردازنده آشنا شدیم
  - پس ازتکمیل طراحی و تحقق الزامات کارکردی به بهبود کارایی و کیفیت عملکردی طرح خود می‌پردازیم
- کارایی و هزینه مهم‌ترین اهداف در طراحی سیستم کامپیوتری
  - **مهم‌ترین مانع:** اجرای ترتیبی و وابستگی مراحل عملیات
- تکنیک‌های تسریع برای افزایش سرعت و بهبود کارایی: **پردازش موازی**
  - **خط‌لوله:** تکنیک تفکیک دنباله‌ای از عملیات ترتیبی به چندین مرحله جزئی قابل اجرای موازی
    - جدا کردن اجزای یک مدار بزرگ به بخش‌های کوچک و شکاندن یک وظیفه بزرگ با هدف اجرای هم‌زمان

خط لوله – Pipeline

- مثال کاربردی: فرض کنید ۴ سرویس مجزای خشکشویی داریم که می‌بایست شسته، خشک و بسته‌بندی شوند
  - اگر شستن ۳۰ دقیقه، خشک کردن ۴۰ دقیقه و بسته‌بندی ۲۰ دقیقه طول بکشد
  - یک ماشین لباسشویی، یک ماشین خشک‌کن و یک بخش بسته‌بندی داشته باشیم
  - موثرترین روش برای مدیریت این چهار سرویس به چه نحو است؟

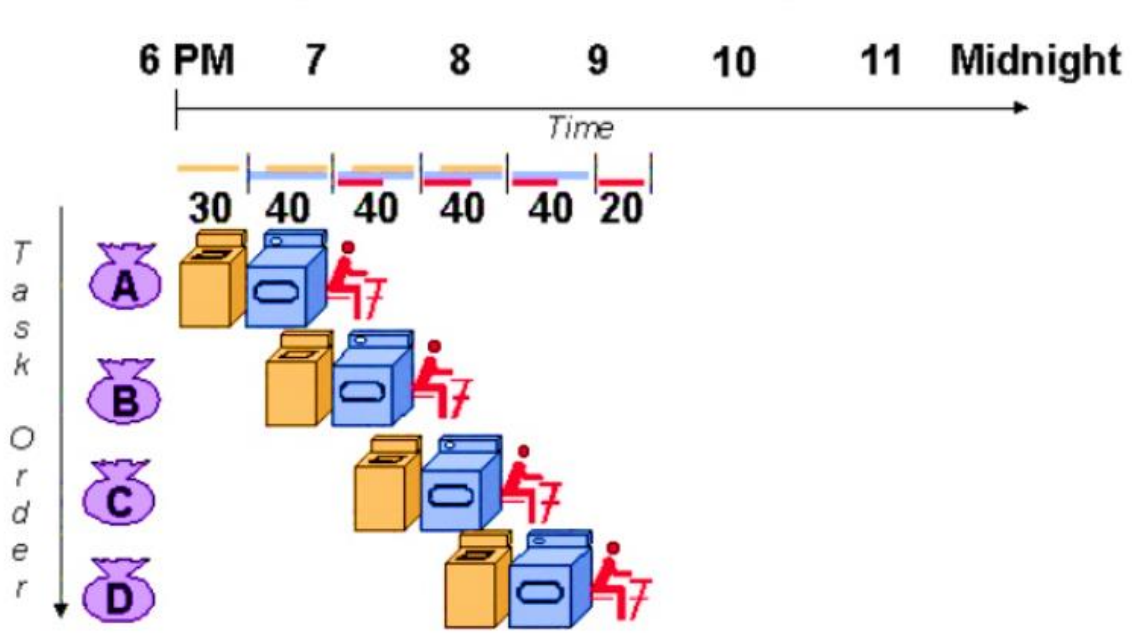
Pipeline – خط لوله

- مثال کاربردی: خشکشویی چهار سرویس
  - انجام عملیات به‌صورت ترتیبی
  - ۶ ساعت طول می‌کشد



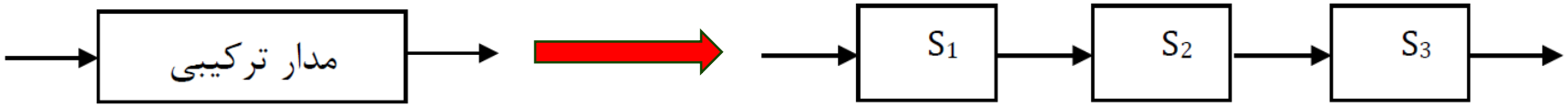
Pipeline – خط لوله

- مثال کاربردی: خشکشویی چهار سرویس
  - انجام عملیات به‌صورت خط‌لوله
    - ۳/۵ ساعت طول می‌کشد



خط لوله در طراحی پردازنده

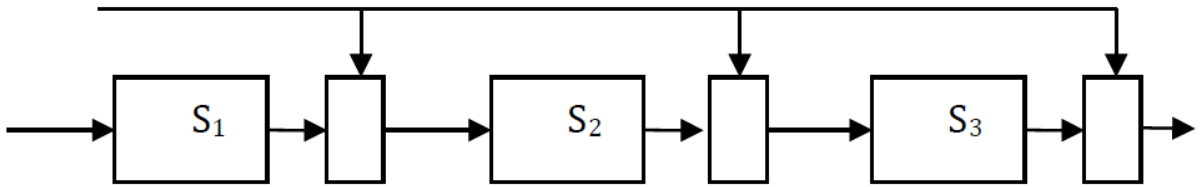
- استفاده از ایده شرح داده شده در مثال خشک‌شویی در بخش‌های مختلف طراحی پردازنده
  - **شرط:** جدا کردن اجزای یک مدار بزرگ به بخش‌های کوچک و شکاندن یک وظیفه بزرگ



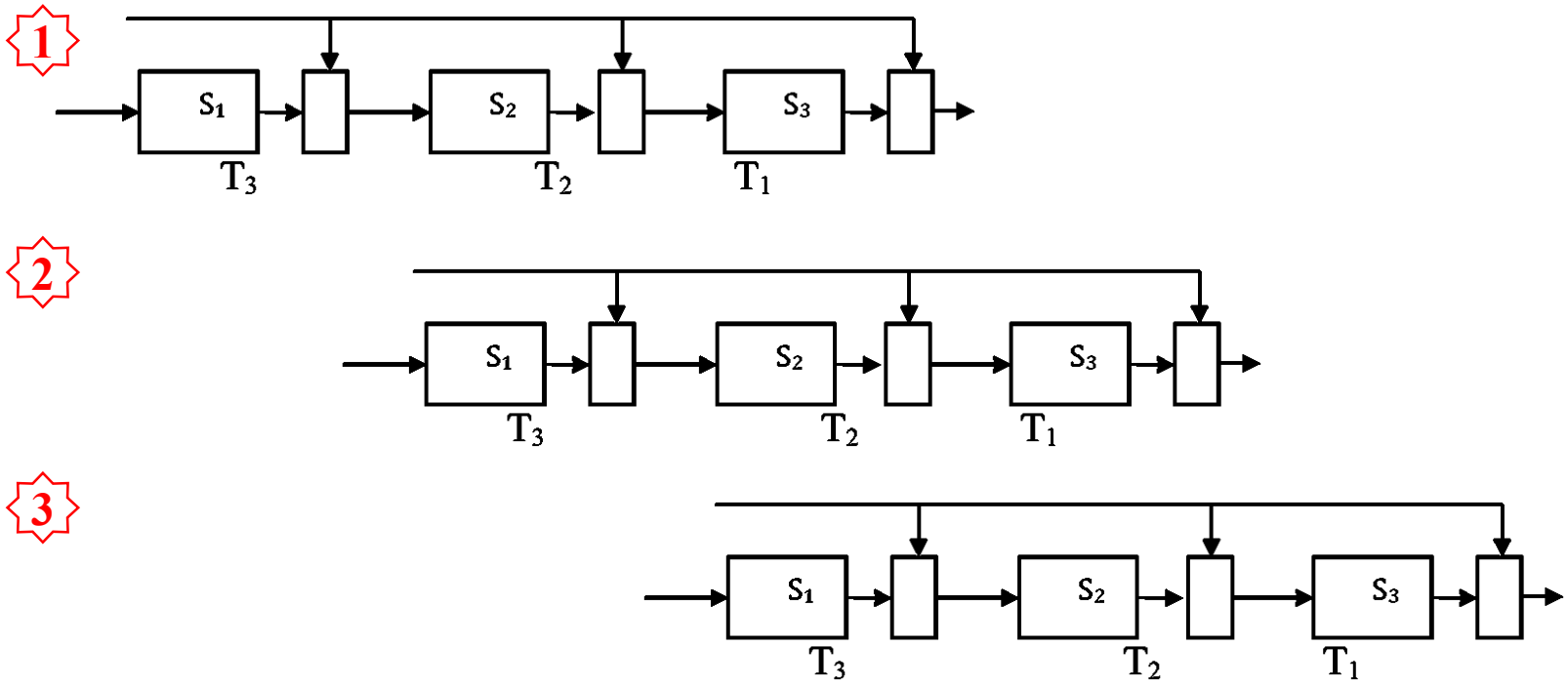
  - **خط لوله حسابی:** پیاده‌سازی سریع اعمال حسابی مانند ضرب و تقسیم ممیز شناور
  - **خط لوله دستورالعمل:** افزایش کارایی چرخه دستورالعمل

خط لوله در طراحی پردازنده

- در طراحی خطلوله پس از هر مرحله (stage) یک ثبات (لچ) داریم
  - بدین ترتیب با ورود داده از بخش اول به ثبات، می‌توان داده جدیدی به بخش اول وارد کرد
- طول کلاک خطلوله برابر بیشینه زمان پایان کار هریک از قطعات کوچک
  - $T_{clock} = Max(T_1, T_2, T_3)$



محاسبه تاخیر خط‌لوله



- اجرای اول: 3T
- اجرای دوم: T
- اجرای سوم: T
- و ...

Pipeline – تسریع خط لوله

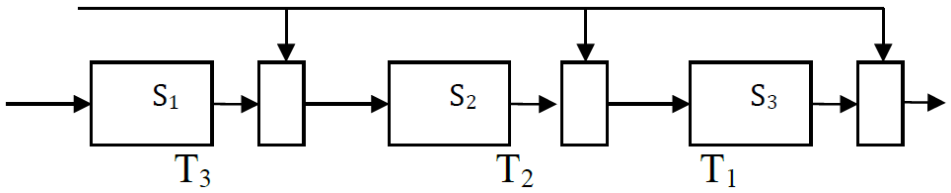
- اگر فرض کنیم n تا اجرا داریم که برای هریک k تا ریزماژول داریم و تاخیر هر یک T است

$$\textbf{Speed up} = \frac{\textit{Old Execution Time}}{\textit{New Execution Time}} = \frac{nkT}{kT_{clock}+(n-1)T_{clock}} = \frac{nk}{k+n-1}$$

- با میل دادن n به سمت بی‌نهایت، نرخ تسریع برابر k می‌شود
- درنتیجه نرخ تسریع به تعداد تقسیم‌های سیستم توسط طراح وابسته است
- امکان تجزیه مدار ترکیبی محدود است و تجزیه برحسب زمان‌های نزدیک به‌هم خوب است
- با هر تجزیه نیاز به یک لچ است که هزینه سخت‌افزاری داریم

تسریع روش خط لوله – Pipeline

- **مثال:** اگر در سیستم زیر، هزار دستور اجرا کنیم و $T_1=10$، $T_2=5$ و $T_3=2$ باشد، میزان تسریع روش خطلوله چقدر است؟



- **حل:**

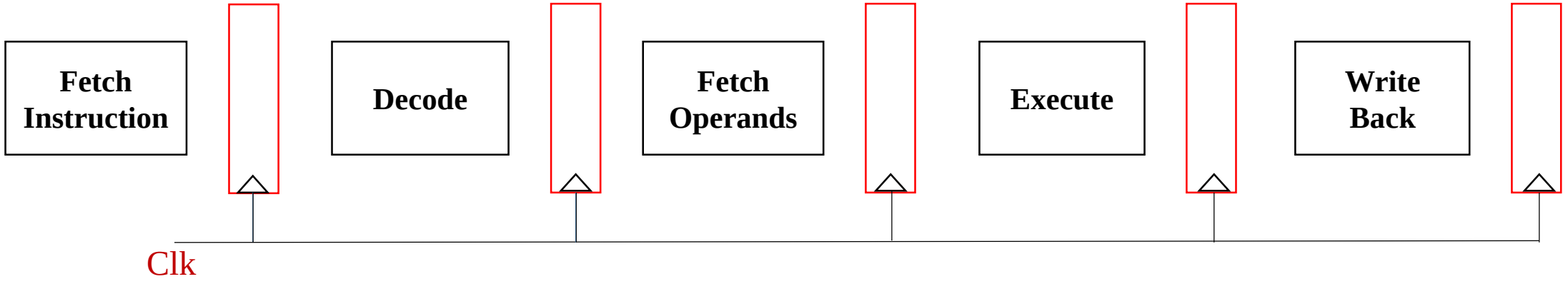
T_{clock} = Max (T1,T2,T3) = 10

Speed up = 1000 * (T1+T2+T3) / 3*T_{clock}+999*T_{clock} =

Speed up = 17000 / 10020 ~ 1.7

خط لوله دستورالعمل

- اجرای ایده خطلوله در روال ترتیبی چرخه دستورالعمل
  - حداکثر تسریع؟



خط لوله دستورالعمل

- نمودار زمانی خط‌لوله اجرای دستورالعمل ۵ مرحله‌ای

	1	2	3	4	5	6	7	8	9	10	11	12
Instruction 1	FI	D	FO	E	WB							
Instruction 2		FI	D	FO	E	WB						
Instruction 3			FI	D	FO	E	WB					
Instruction 4				FI	D	FO	E	WB				
Instruction 5					FI	D	FO	E	WB			
Instruction 6						FI	D	FO	E	WB		
Instruction 7							FI	D	FO	E	WB	
Instruction 8								FI	D	FO	E	WB

مشکلات خط‌لوله

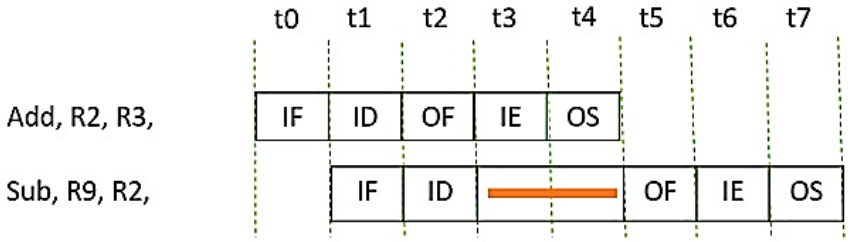
- روش خط‌لوله در کنار مزایا، می‌تواند مشکلاتی هم ایجاد کند: Pipeline Hazards
- راهکار حل مشکلات: صبر کردن و متوقف کردن اجرای موازی (Stall) در خط‌لوله (ایجاد حباب در اجرا)

1. **مخاطرات ساختاری(Structural Hazards)**: ناشی از وابستگی فیزیکی و تداخل منابع است
   - دو دستور موازی در حین اجرا نیاز به ALU دارند در نتیجه تداخل منابع رخ می‌دهد
   - راه‌حل۱: تخصیص منبع مشترک به اولین دستور و متوقف کردن دومین دستور و کل pipeline تا آزاد شده منبع
     - ایجاد وقفه در خط‌لوله (حباب در اجرا)
   - راه‌حل ۲: افزودن هزینه و سربار به سیستم و درنظر گرفتن چندین ALU

مشکلات خط‌لوله

2. مخاطرات داده‌ای (Data Hazards): دسترسی به داده پیش از آماده بودن

- وابستگی بین داده‌های چندین دستور وجود داشته باشد مانند:



- دستور دومی پس از تمام شدن اولی باید اجرا شود تا مقدار R2 به‌روز شود. (stall)
- **راه‌حل ۱:** کامپایلر تا حد ممکن وابستگی در کد را حذف کند یا بین دستورات وابسته تاخیر ایجاد کند
- **راه‌حل ۲:** افزودن سخت‌افزار جهت شناسایی وابستگی و تسریع جریان داده به دستور بعدی
  - مثلا خروجی ALU اگر در دستور بعدی مورد نیاز باشد، به‌جای انتقال به ثبات، به دستور بعد برود

مشکلات خط‌لوله

3. مخاطرات کنترلی (Control Hazards):

- ناشی از بروز یک تصمیم و شرط در روند اجرای برنامه است مانند پرش شرطی
- در پرش‌های شرطی، دستور بعدی برحسب برآورده شدن یا نشدن شرط دستور قبلی مشخص می‌شود
  - تا تمام شدن کامل دستور جاری نمی‌توان دستور بعدی را دانست و آن را شروع کرد
  - تا تمام شدن دستور جاری و مشخص شدن مقصد پرش، خط‌لوله stall می‌شود

#cycle	1	2	3	4	5	6	7	8	9	10	11
IF	I1	I2	I3	I4(JZ)	stall	stall	stall	stall	I5	I6	I7
D		I1	I2	I3	I4	stall	stall	stall	stall	I5	I6
FO			I1	I2	I3	I4	stall	stall	stall	stall	I5
E				I1	I2	I3	I4	stall	stall	stall	stall
WB					I1	I2	I3	I4	stall	stall	stall

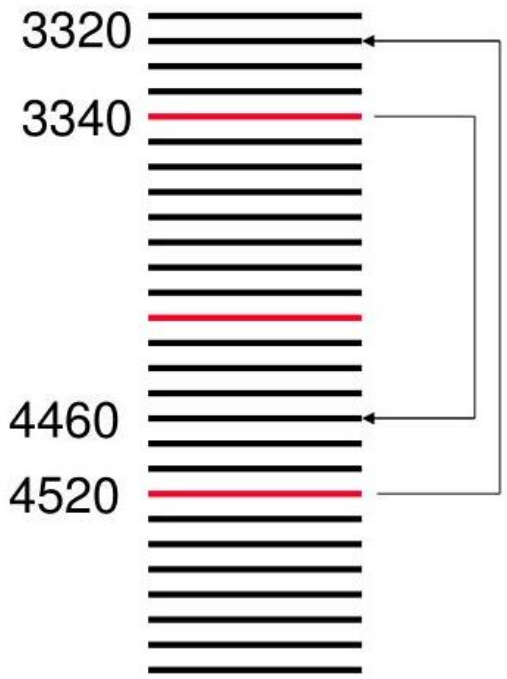
مشکلات خط‌لوله

3. مخاطرات کنترلی (Control Hazards):

- **راه‌حل ۱:** ملزم کردن کامپایلر به عدم استفاده از پرش‌های شرطی
- **راه‌حل ۲:** پیش‌بینی انشعاب Branch Prediction
  - یکی از دو حالتی که در شرط پرش آمده را برحسب تاریخچه اجرای دستور فعلی طی می‌کنیم
    - اگر درست بود که اجرا تسریع شده و از خط‌لوله خارج نشده‌ایم
    - اگر غلط بود، نتیجه را پاک کرده و دوباره انجام می‌دهیم در نتیجه مثل حالتی است که stall داشتیم
    - بهبود این پیش‌بینی براساس مکانیزمی مشابه cache
      - ذخیره نتیجه پرش‌ها در جدول و استفاده از آن (BTB: Branch Target Buffer)

جدول پیش‌بینی پرش (BTB)

- مشابه حافظه نهان ظرفیت این جدول هم محدود است



PC	Target PC	Prediction
3340	4460	1(T)
4520	3320	1(T)