تمرین پنج - خط لوله **مهلت تحویل ساعت ۲۴ روز یکشنبه ۳۰ اردیبهشت**

به موارد زیر توجه کنید:

۱- حتما نام و شماره دانشجویی خود را روی پاسخ‌نامه بنویسید.

۲- کل پاسخ تمرینات را در قالب یک فایل pdf با شماره دانشجویی خود نام‌گذاری کرده در سامانه CW بارگذاری کنید.

۳- این تمرین ۶۰ نمره دارد که معادل ۰٫۶ نمره از نمره کلی درس است.

۴- در صورت مشاهده هر گونه مشابهت نامتعارف هر دو (یا چند) نفر کل نمره این تمرین را از دست خواهند داد.

۱- (۲۰ نمره) پردازنده‌ای داریم که بدون استفاده از خط لوله با نرخ ساعت ۲٫۵ گیگاهرتز، هر دستور را به طور متوسط در ۵ چرخه اجرا می‌کند. همین پردازنده به یک پردازندهٔ خط لوله با پنج مرحله ارتقا یافته است، اما به دلیل تاخیر داخلی خط لوله، نرخ ساعت به ۲ گیگاهرتز کاهش یافته است.

الف- فرض کنید هیچ وابستگی بین دستورات وجود ندارد. تسریع حاصل از خط لوله چقدر است؟

پاسخ:

$$speedup = \frac{5/2.5}{1/2} = 4$$

ب- فرض کنید درصد دستوراتی که وابستگی داده دارند طبق جدول زیر است. این جدول شامل دستوراتی است که نتیجه‌ای که در یک مرحله از خط لوله تولید می‌شود در یک دستور پس از آن، یا در دو دستور پس از آن و یا در هر دو دستور پس از آن موردنیاز خواهند بود. اگر هیچ سازوکاری برای هدایت به جلو (forwarding) نداشته باشیم، تسریع حاصل از خط لوله چقدر خواهد بود؟

No. of stalls	EX to 1st Only	MEM to 1st only	Ex to 2nd Only	MEM to 2nd Only	EX to 1st and EX to 2nd
	5%	20%	10%	10%	5%
no for.	2	2	1	1	2
full for.	0	1	0	0	0

پاسخ:

با توجه به تعداد تعلیق‌های موردنیاز در حالتی که سازوکاری برای هدایت به جلو نداریم، میانگینِ زمانِ اجرای هر دستور از رابطهٔ زیر محاسبه می‌شود:

$$ExecTimeNofor = \frac{1}{2}[1 + 2 \times (0.05 + 0.20 + 0.05) + 1 \times (0.10 + 0.10)] = \frac{1}{2}(1 + 0.6 + 0.2) = 0.9\ ns$$

بنابراین، تسریع این طور حساب می‌شود:

$$speedup = \frac{5/2.5}{0.9} = 2.2$$

ج- اگر سازوکار هدایت به جلو به طور کامل استفاده شود (یعنی داده‌ها را در صورت امکان به صورت زودهنگام در اختیار بگیریم) به سوال بند ب دوباره پاسخ دهید.

$$ExeTimeFullfor = \frac{1}{2}[1 + 1 \times 0.20] = 0.6\ ns$$

$$speedup = \frac{5/2.5}{0.6} = 3.3$$

د- اگر ۱۰٪ از دستورات، دستورات پرش باشند و سخت‌افزار لازم برای محاسبهٔ شرط و مقصدِ پرش در مرحلهٔ دوم خط لوله وجود داشته باشد، با فرضیاتِ بند ج و بدون پیش‌بینی انجام شدن یا نشدنِ پرش، تسریع حاصل از خط لوله چقدر خواهد بود؟

پاسخ:

با توجه به فرض این بند، هر دستور پرش یک چرخه تعلیق ایجاد می‌کند، بنابراین زمان اجرا و تسریع به این صورت محاسبه می‌شود:

$$ExeTimeFullfor\&Branch = \frac{1}{2}[1 + 1 \times 0.20 + 1 \times 0.10] = 0.65\ ns$$

$$speedup = \frac{5/2.5}{0.65} = 3.08$$

ه- این بار خط لوله را طوری می‌سازیم که همه پرش‌ها را انجام‌نشده فرض کند. می‌دانیم که فقط ۲۰٪ از پرش‌ها انجام نمی‌شوند. در این صورت تسریعِ حاصل از خط لوله چقدر خواهد بود؟

پاسخ:

با توجه به فرض این بند، فقط پرش‌های انجام‌شده یک چرخه تعلیق ایجاد می‌کند، بنابراین زمان اجرا و تسریع به این صورت محاسبه می‌شود:

$$ExeTimeFullfor\&Branch = \frac{1}{2}[1 + 1 \times 0.20 + 1 \times 0.10 \times 0.80] = 0.64\ ns$$

$$speedup = \frac{5/2.5}{0.64} = 3.125$$

۲- (۱۰ نمره) یک پردازنده غیر خط لوله‌ای را در نظر بگیرید که با ساعت ۴ گیگاهرتز کار می‌کند. این پردازنده به ازای هر عملیات ALU

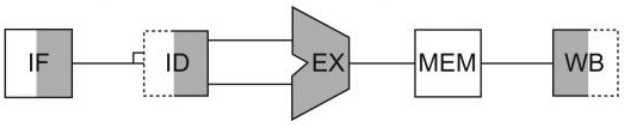
۳- یا Branch چهار چرخه و به ازای هر عملیات حافظه ۱۰ چرخه استفاده می‌کند. فرض کنید که فراوانی این دستورات به ترتیب ۴۰٪، ۳۰٪ و ۳۰٪ باشد. فرض کنید اضافه کردن خط لوله ۰٫۲ نانوثانیه سربار به چرخهٔ ساعت اضافه می‌کند. با صرف نظر کردن از هرگونه تاخیر، تسریعِ حاصل از اضافه کردن خط لوله را محاسبه کنید.

پاسخ:

$$CC\_noPipeline = \frac{1}{4} = 0.25\ ns \quad \Rightarrow \quad CC_{withPipeline} = 0.5 + 0.2 = 0.45\ ns$$

$$speedup = \frac{execTimeNoPipeline}{execTimeWithPipeline} = \frac{[4 \times (0.4 + 0.3) + 10 \times 0.3] \times 0.25}{1 \times 0.45} = \frac{1.45}{0.45} = 3.2$$

۴- (۱۵ نمره) یک پیاده‌سازی خط لوله‌ای از MIPS ISA را در نظر بگیرید. این ماشین یک ALU داشته و شامل یک خط لوله ۵ مرحله‌ای به شکل زیر است.



با فرض اینکه دستورات اسمبلی زیر به ترتیب وارد خط لوله می‌شوند، روند اجرای این دستورات را در جدول مشخص کنید. مراحل خط لوله را با حروف F/D/X/M/W مشخص کنید و اگر دستوری در هیچ مرحله‌ای از خط لوله نیست، آن خانه را خالی بگذارید. همچنین اگر بین مراحل به دلیل data hazard فاصله می‌افتد، آن خانه(ها) را با *d مشخص کنید. همچنین در تمام قسمت‌های زیر فرض کنید که فایل ثبات در نیمه اول clock نوشته شده و در نیمه دوم clock خوانده می‌شود.

الف- اجرای دستورات را در حالتی که هیچ forwarding (bypassing) ای وجود نداشته باشد نشان دهید.

ب- اجرای دستورات را در حالتی که full forwarding وجود داشته باشد نشان دهید.

ج- اجرای دستورات را در حالتی نشان دهید که فقط EX to EX forwarding داریم.

الف-

instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14
add $3,$1,$5	F	D	X	M	W									
sub $2,$1,$5		F	D	X	M	W								
lw $5,0($3)			F	d*	D	X	M	W						
addi $4,$5,1					F	d*	d*	D	X	M	W			
add $5,$4,$1								F	d*	d*	D	X	M	W

ب-

instructions	1	2	3	4	5	6	7	8	9	10
add $3,$1,$5	F	D	X	M	W					
sub $2,$1,$5		F	D	X	M	W				
lw $5,0($3)			F	D	X	M	W			
addi $4,$5,1				F	d*	D	X	M	W	
add $5,$4,$1						F	D	X	M	W

ج-

instructions	1	2	3	4	5	6	7	8	9	10	11	12
add $3,$1,$5	F	D	X	M	W							
sub $2,$1,$5		F	D	X	M	W						
lw $5,0($3)			F	d*	D	X	M	W				
addi $4,$5,1					F	d*	d*	D	X	M	W	
add $5,$4,$1								F	D	X	M	W

۵- (۱۵ نمره) می‌خواهیم برنامهٔ زیر را در یک پردازنده با یک خط لولهٔ ۵ مرحله‌ای (مشابه با بندِ ب سوالِ قبل) اجرا کنیم. فرض کنید مقدارِ اولیهٔ R4 برابر با ۱۰۰ است.

```
I1: lw    R1,0(R2)          ; R1 ← Memory[R2]
I2: addi  R1,R1,1           ; R1 ← R1+1
I3: sw    R1,0(R2)          ; Memory[R2] ← R1
I4: addi  R2,R2,8           ; R2 ← R2+8
I5: addi  R4,R4,-1          ; R4 ← R4-1
I6: bne   R4,R0,I1          ; branch if R4!=0
```

الف- در این خط لوله مدارهایی داریم که بتوانند شرط و مقصدِ پرش را در فازِ دوم محاسبه کنند، به این شرط که مقادیرِ ثبات‌های موردِ مقایسه آماده باشند. ترتیبِ دستورات را طوری تغییر دهید که این شرط برای دستورِ bne صدق کند.

پاسخ:

چون I6 به I5 وابسته است، باید I5 را ببریم بالاتر طوری که حداقل دو دستور بین I5 و I6 فاصله باشد. بنابراین می‌توانیم I5 را بین I2 و I3 ببریم. اما بهتر این است که I5 را بین I1 و I2 ببریم که اثر وابستگی I2 و I1 را هم از بین برده باشیم.

```
I1: lw    R1,0(R2)          ; R1 ← Memory[R2]
I5: addi  R4,R4,-1          ; R4 ← R4-1
I2: addi  R1,R1,1           ; R1 ← R1+1
I3: sw    R1,0(R2)          ; Memory[R2] ← R1
I4: addi  R2,R2,8           ; R2 ← R2+8
I6: bne   R4,R0,I1          ; branch if R4!=0
```

ب- اگر پردازنده از روشِ delayed branch استفاده کند، دوباره ترتیبِ دستورات را طوری تغییر دهید که دستورِ مناسب در delayed slot قرار بگیرد.

پاسخ:

این جا می‌توانیم I4 را به بعد از I6 منتقل کنیم، چون کاملا مستقل از I6 قابل اجرا است.

```
I1: lw    R1,0(R2)          ; R1 ← Memory[R2]
I5: addi  R4,R4,-1          ; R4 ← R4-1
I2: addi  R1,R1,1           ; R1 ← R1+1
I3: sw    R1,0(R2)          ; Memory[R2] ← R1
I6: bne   R4,R0,I1          ; branch if R4!=0
I4: addi  R2,R2,8           ; R2 ← R2+8
```

ج- با توجه به همهٔ تغییراتی که در بندهای پیش داده‌اید، با رسمِ جدولی مشابه با جدولِ سوال قبل برای یک دور اجرای حلقه، حساب کنید کلِ برنامه در چند چرخه اجرا می‌شود؟

پاسخ:

instructions	1	2	3	4	5	6	7	8	9	10	11
lw R1,0(R2)	F	D	X	M	W						
addi R4,R4,-1		F	D	X	M	W					
addi R1,R1,1			F	D	X	M	W				
sw R1,0(R2)				F	D	X	M	W			
bne R4,R0,I1					F	D	X	M	W		
addi R2,R2,8						F	D	X	M	W	
lw R1,0(R2)							F	D	X	M	W

حلقه ۱۰۰ بار تکرار می‌شود و در هر دور تکرار ۶ دستور اجرا می‌کند، بنابراین در کل ۶۰۰ دستور داریم، پس تعداد چرخه‌های موردنیاز $۵۹۹ \times ۱ + ۵ = ۶۰۴$ چرخه است.

به روش دیگری هم می‌توانیم تعداد چرخه‌ها را حساب کنیم. از روی جدول بالا مشخص است که اجرای هر دو از حلقه در ۶ چرخه تمام می‌شود، بنابراین همه دستورات (به جز دستور آخر) بعد از ۶۰۰ چرخه کامل می‌شوند. دستور آخر هم ۴ چرخهٔ بعد کامل می‌شود، پس در مجموع به ۶۰۴ چرخه نیاز داریم.