

معماری کامپیوتر

دانشکده مهندسی کامپیوتر

دکتر اسدی
بهار ۱۴۰۳

مهدی علی نژاد، ۴۰۱۱۰۶۲۶۶



تمرین ۷

سوال ۱

۱. برنامه‌ی زیر را در نظر بگیرید و فرض کنید مقدار اولیه‌ی t_2 برابر ۱۰۰ و مقدار اولیه‌ی t_3 برابر ۰ است:

```
1  LOOP:  lw $t1, 0x100($t0)
2         addi $t1, $t1, 0x1
3         sw $t1, 0x100($t0)
4         addi $t0, $t0, 8
5         addi $t2, $t2, -1
6         bne $t2, $t3, LOOP
```

(آ) تمام وابستگی‌های داده^۱ در داخل یک تکرار^۲ حلقه را بنویسید.

(ب) فرض کنید که یک پردازنده‌ی MIPS داریم که هیچ سخت‌افزاری برای forwarding یا خواندن و نوشتن در دو لبه‌ی بالا و پایین چرخه ساعت ندارد. همچنین در نظر بگیرید که پردازنده فرض می‌کند که تمام پرش‌های not taken هستند و در صورتی که taken بودند، دو دستور بعدی موجود در خط‌لوله^۳ را از آن خارج می‌کند و دستور درست را fetch می‌کند. جدول زمان‌بندی اجرای این برنامه را بنویسید و CPI را محاسبه کنید.

(آ) وابستگی‌ها:

• خط ۲ به خط ۱، دلیل: مقدار t_1

• خط ۳ به خط ۲، دلیل: مقدار t_1

• خط ۶ به خط ۵، دلیل: مقدار t_2

(ب) جدول زمان‌بندی به ازای هر حلقه‌ی این برنامه به این صورت است.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1 I	1 R	1 A	1 M	1 W	2 R	2 A	2 M	2 W	3 R	3 A	3 M	3 W			6 R	6 A	6 M	6 W
	2 I				3 I				4 I	4 R	4 A	4 M	4 W					
									5 I	5 R	5 A	5 M	5 W					

از جدول به این نتیجه می‌رسیم که در هر ۱۷ کلاک سایکل، ۶ دستور اجرا می‌شود. (عملاً کلاک ۱۸ ام شروع حلقه‌ی بعد است پس CPI آن برابر است با $\frac{17 \times 99 + 19}{6}$)

۲. در یک سیستم دیجیتال، پردازش ورودی ۱۲ نانوثانیه زمان می برد. دو خط لوله مختلف A با ۶ طبقه و تأخیر طبقات (۱)، (۱، ۲، ۳، ۲، ۲) نانوثانیه و خط لوله B با ۴ طبقه و تأخیر طبقات (۲، ۳، ۴، ۳) برای این سیستم طراحی و ساخته شده اند (فرض کنید تأخیر بافر بین طبقات ناچیز است). اگر زمان پردازش n ورودی با خط لوله A را با TA_n و زمان پردازش n ورودی با خط لوله B را با TB_n نشان دهیم، نسبت $\frac{TA_\infty}{TB_\infty}$ را بدست آورید.

چون کلاک در هر خط لوله باید یکسان باشد، پس دوره تناوب آن باید برابر با بیشترین تأخیر بین طبقات مختلف پایپ لاین شود، پس برای A دوره ی تناوب کلاک برابر با ۳ns و برای B این مقدار برابر ۴ns می باشد. حال اگر n ورودی داشته باشیم، خواهیم داشت:

$$TA_n = 5 \times CT_A + n \times CT_A$$

$$TB_n = 3 \times CT_B + n \times CT_B$$

$$\lim_{n \rightarrow \infty} \frac{TA_n}{TB_n} = \lim_{n \rightarrow \infty} \frac{5 + 3 \times n}{3 + 4 \times n} = \frac{3}{4}$$

۳. درباره انواع وابستگی‌های داده (RAW, WAW, WAR) تحقیق کنید و به اختصار توضیح دهید که در چه صورتی رخ می‌دهند. همچنین یک مثال برای هرکدام بیاورید.

• Read After Write (RAW)

این وابستگی مربوط به زمانی است که قبل از اینکه داده‌ی درست برای یک رجیستر محاسبه شود، مقدار قبلی آن را برای یک دستور دیگر بخوانیم. برای مثال:

```
addi R2, R0, 1
add R4, R0, R2
```

• Write After Read (WAR)

این حالت زمانی اتفاق می‌افتد که دستوری که بعد از خواندن رجیستر در دستور قبل تر آمده، زودتر به پایان برسد و مقدار آن رجیستر را عوض کند. برای مثال:

```
add R4, R0, R2
addi R2, R0, 5
```

• Write After Write (WAW)

این وابستگی نیز مربوط به زمانی است که دو دستور که پشت سر هم آمده‌اند مقصد یکسان داشته باشند و در یک رجیستر مقدار بنویسند ولی نوشتن دوم زودتر از نوشتن اول صورت می‌گیرد. برای مثال:

```
add R4, R4, R2
add R4, R4, R3
```

۴. فرض کنید دستورات یک برنامه به شکل زیر هستند:

%35 r-type, %25 beq, %5 jmp, %30 lw, %5 sw

همچنین سه Branch Predictor داریم. اولین Branch Predictor همواره پرش شرطی را taken پیش‌بینی می‌کند، دومین Branch Predictor همواره not-taken پیش‌بینی می‌کند و سومی به صورت پویا است. در صورتی که درصد موفقیت اولی ۵۵ درصد، دومی ۴۵ درصد و برای سومی در حالت taken برابر ۷۰ درصد و برای حالات not-taken برابر ۸۰ درصد باشد، میزان cpi ناشی از توقف^۴ را در اجرای این برنامه برای هر Branch Predictor محاسبه کنید. (در این پردازنده آدرس پرش در ID محاسبه می‌شود، همچنین فرض کنید مخاطره داده نداریم)

برای محاسبه ی CPI از دست رفته به دلیل پرش نیاز است درصد برنامه را که توسط برنج تشکیل شده است را ضرب در حالتی که دچار اشتباه در پیش‌بینی شده ایم ضرب کنیم و در آخر نیز جواب را در ۲ ضرب کنیم. زیرا به ازای هر اشتباه نیاز است ۲ دستور بعدی پایپ لاین رو خالی کنیم و در واقع ۳ سیکل کلاک را از دست می‌دهیم.

برای اولین پردیکتور داریم:

$$0.25 \times 0.45 \times 2 = 0.225$$

برای پردیکتور دومی داریم:

$$0.25 \times 0.55 \times 2 = 0.275$$

برای سومین پردیکتور داریم:

$$0.25 \times (0.3 \times 0.55 + 0.2 \times 0.45) \times 2 = 0.125$$

۵. به هر یک از سوالات زیر پاسخ دهید:

(آ) دو پردازنده دارای خطلوله A و B داریم. پردازنده A دارای ۵ مرحله و تاخیر 2ns در هر مرحله است و پردازنده B دارای ۸ مرحله و تاخیر 1.5ns می باشد. اگر از تاخیر ثبات های میان مراحل صرفه نظر شود و تعداد زیادی دستورالعمل را بتوان بدون هیچ وقفه ای در این دو پردازنده اجرا کرد، این دو پردازنده را با هم مقایسه کنید.

(ب) اگر یک خطلوله سه مرحله ای را به خطلوله ۴ مرحله ای تبدیل کنیم دوره چرخه ساعت از T به 0.9T کاهش می یابد. فرض کنید ۳۰ درصد دستورات از نوع پرش هستند. همچنین تا وقتی که دستور پرش به پایان نرسد دستور بعدی وارد خطلوله نمی شود. زمان اجرای ۱۰۰ دستور در خطلوله سه مرحله ای نسبت به خطلوله چهار مرحله را محاسبه کنید.

(آ) اگر فرض کنیم هر دو پردازنده به حالت steady برسند و هیچ هازاردی باعث تخلیه ی pipeline نشود، پردازنده ی A در هر ۱۰ نانوثانیه، ۵ دستور اجرا می کند و پردازنده ی B نیز در هر ۱۲ نانوثانیه، ۸ دستور اجرا می کند. پس CPI پردازنده ی A برابر است با ۲ و برای پردازنده ی B برابر است با $\frac{3}{4}$ است، پس پردازنده ی B بهتر است.

(ب) برای مقایسه ی زمان اجرا، CPI را محاسبه کرده و سپس در دوره تناوب ضرب می کنیم تا زمان اجرا را بدست آوریم. در حالت اول داریم:

$$CPI_1 = \frac{30 * 3 + 2 + 70}{100} = 1/62$$

برای حالت دوم نیز به این صورت است:

$$CPI_2 = \frac{30 * 4 + 3 + 70}{100} = 1/93$$

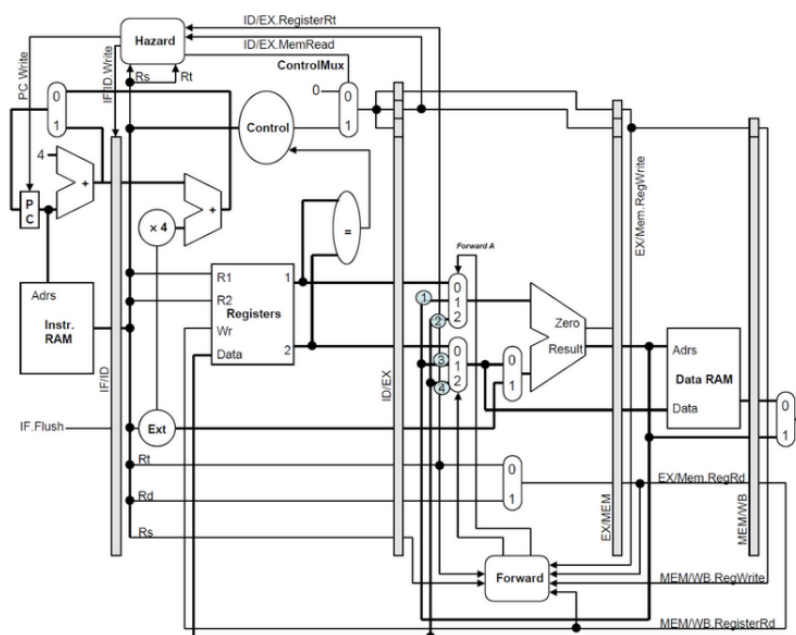
و سپس زمان اجرای آنها به این شکل است:

$$runtime_1 = 1/62 \times 100 \times T = 162T$$

$$runtime_2 = 1/93 \times 100 \times 0.9T = 173/9T$$

همانطور که مشاهده شد، زیاد کردن مرحله های خطلوله در اینجا به ضرر ماست.

۶. مسیره زیر برای پردازنده دارای خط لوله MIPS که شامل Forwarding می‌شود ولی Branch Prediction ندارد را در نظر بگیرید. همچنین کد C زیر را در نظر بگیرید.



for (int i = 0; i < max; ++i) ++A[i];
loop:

1. sub \$t3, \$t3, 1
2. add \$t0, \$t0, 4
3. add \$t1, \$a0, \$t0
4. lw \$t2, 0(\$t1)
5. add \$t2, \$t2, 1
6. sw \$t2, 0(\$t1)
7. bgt \$t3, \$zero, loop

(آ) همه وابستگی‌هایی بین بخش‌های مختلف کد در یک دور این حلقه را مشخص کنید. توجه کنید که مواردی که لزوماً نیاز به forwarding ندارند را هم مشخص کنید. منظور از یک دور حلقه، از ابتدای sub تا انتهای bgt است.

(ب) جدول زیر را برای گام‌های مختلف خط‌لوله تکمیل کنید. فرض کنید که قبل از این چندین بار این حلقه اجرا شده است و در اولین دور اجرای آن نیستیم. (اصطلاحاً در وضعیت steady state هستیم) توجه کنید که در خط‌لوله Forwarding وجود دارد ولی branch prediction نداریم. همچنین stall را می‌توانید با خط تیره یا S نمایش دهید.

Inst	iter	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2
sub	N	F	D	E	M	W																	
add	N		F																				
add	N																						
lw	N																						
add	N																						
sw	N																						
bgt	N																						
sub	N+1																						

(ج) شماره مسیر forwarding مورد استفاده برای هر یک از دستورات را مشخص کنید. شماره مسیرهای مختلف در شکل مشخص شده است.

(د) کد را طوری بازنویسی کنید که کارایی بهینه داشته باشد. سعی کنید که تعداد مسیرهای Forward مورد استفاده را نیز کمینه کنید.

(آ) وابستگی ها:

- خط ۳ به خط ۲ ، دلیل: مقدار t_0
- خط ۴ به خط ۳ ، دلیل: مقدار t_1
- خط ۵ به خط ۴ ، دلیل: مقدار t_2
- خط ۶ به خط ۵ و ۳ ، دلیل: مقدار t_1 و t_2
- خط ۷ به خط ۱ ، دلیل: مقدار t_3

(ب) .

Inst	Iter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
sub	N	F	D	E	M	W													
add	N		F	D	E	M	W												
add	N			F	D	E	M	W											
lw	N				F	D	E	M	W										
add	N					F	D	-	E	M	W								
sw	N						F	-	D	E	M	W							
bgt	N								F	D	E	M	W						
sub	N+1										F	D	E	M	W				

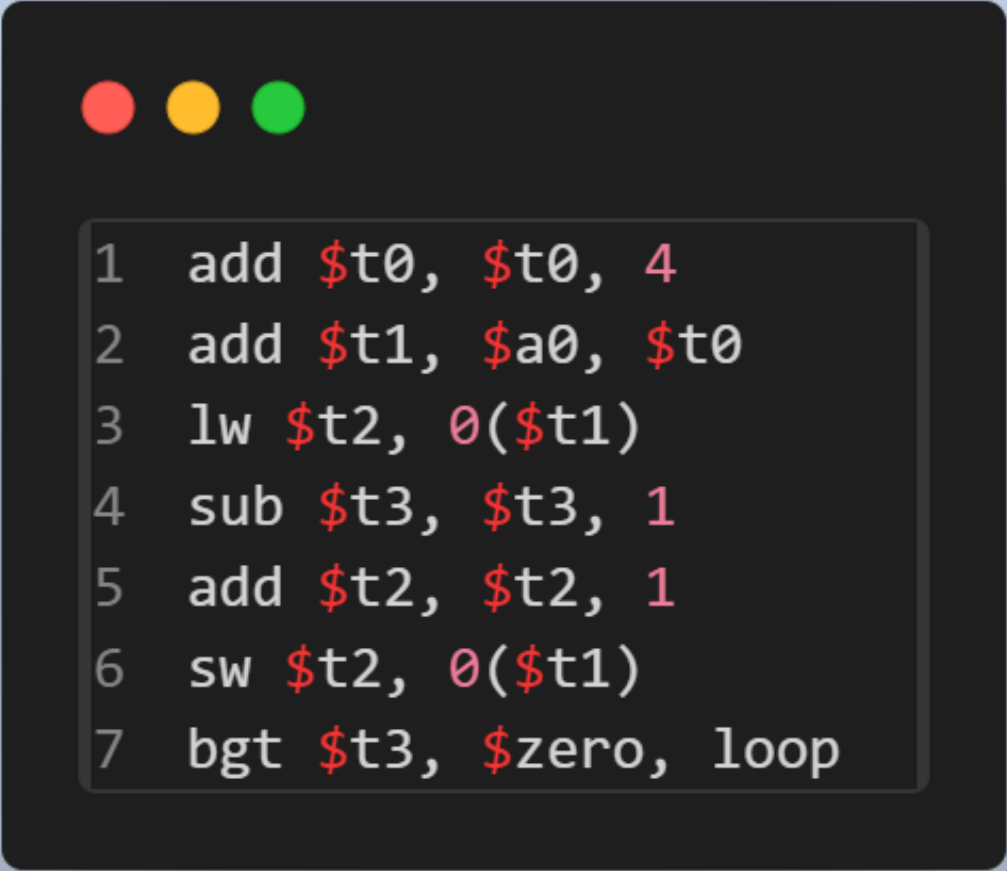
(ج) با فرض اینکه در دستورها به فرمت زیر باشند:

add rd, rs, rt
lw rt, imm(rs)
sw rt, imm(rs)

و rs همان R_1 و rt همان R_2 باشد داریم:

- وابستگی اول از مسیر ۳ داده را ارسال می کند.
- وابستگی دوم از مسیر ۱ استفاده می کند.
- وابستگی سوم از مسیر ۲ استفاده می کند.
- وابستگی چهارم از مسیر ۳.
- و وابستگی ۵ام نیازی به فوروارد کردن ندارد.

(د) .



```
1  add $t0, $t0, 4
2  add $t1, $a0, $t0
3  lw  $t2, 0($t1)
4  sub $t3, $t3, 1
5  add $t2, $t2, 1
6  sw  $t2, 0($t1)
7  bgt $t3, $zero, loop
```

با انجام این تغییرات کل فرایند در یک کلاک کمتر انجام می شود.