به نام خدا

دانشکده مهندسی کامپیوتر

درس معماری کامپیوتر
نیم‌سال دوم ۰۲-۰۳
استاد: دکتر حسین اسدی

تمرین سری هفتم

- پرسش‌های خود را در صفحه quera مربوط به تمرین مطرح نمایید.
- سوالات نظری را حتماً به صورت انفرادی و سوالات عملی را می‌توانید در گروه‌های دو نفر تحویل دهید.
- پاسخ‌ها را به صورت تایپی بنویسید.
- اسکرین‌شات‌ها، عکس‌ها و فایل‌های مربوط به سوال عملی را در فایل فشرده مربوطه در cw و quera قرار دهید. هر گونه عدم تطابق بین دو تمرین آپلود شده در دو سایت منجر به از دست رفتن نمره تمرین مربوطه می‌شود.
- پی دی اف قسمت تئوری را در سامانه cw و quera بارگذاری کنید.
- هر دانشجو می‌تواند حداکثر سه تمرین را با دو روز تأخیر بدون کاهش نمره ارسال نماید.

تمارین تئوری

۱. برنامه‌ی زیر را در نظر بگیرید و فرض کنید مقدار اولیه‌ی $t2$ برابر ۱۰۰ و مقدار اولیه‌ی $t3$ برابر ۰ است:

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

آ) تمام وابستگی‌های داده[1] در داخل یک تکرار[2] 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 درصورتی‌که taken بودند، دو دستور بعدی موجود در خط‌لوله[3] را از آن خارج می‌کند و دستور درست را fetch می‌کند. جدول زمان‌بندی اجرای این برنامه را بنویسید و CPI را محاسبه کنید.

۲. در یک سیستم دیجیتال، پردازش ورودی ۱۲ نانوثانیه زمان می‌برد. دو خط لوله مختلف A با ۶ طبقه و تأخیر طبقات (۱، ۲، ۲، ۳، ۲، ۱) نانوثانیه و خط لوله B با ۴ طبقه و تأخیر طبقات (۲، ۳، ۴، ۳) برای این سیستم طراحی و ساخته شده‌اند (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.

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.

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[4] را در اجرای این برنامه برای هر Branch Predictor محاسبه کنید. (در این پردازنده آدرس پرش در ID محاسبه می شود، همچنین فرض کنید مخاطره داده نداریم)

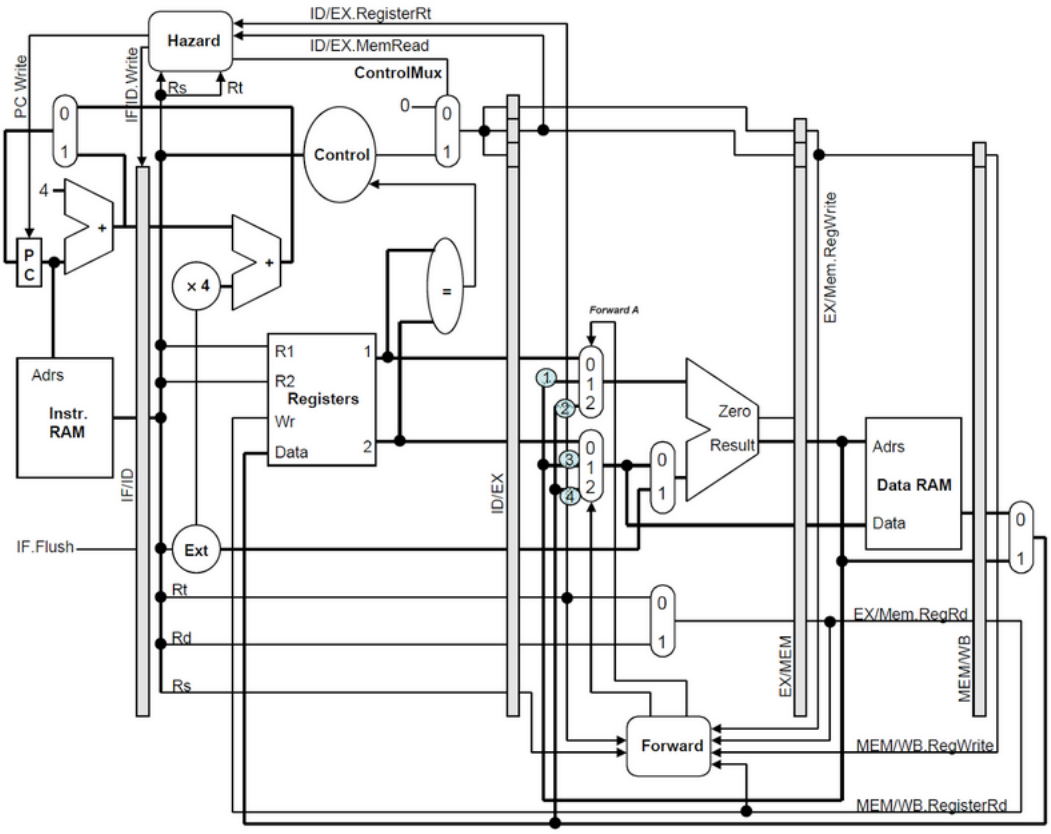
۵. به هر یک از سوالات زیر پاسخ دهید:

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 می‌باشد. اگر از تاخیر ثبات‌های میان مراحل صرفه نظر شود و تعداد زیادی دستورالعمل را بتوان بدون هیچ وقفه‌ای در این دو پردازنده اجرا کرد، این دو پردازنده را با هم مقایسه کنید.

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

[1] data dependency
[2] loop iteration
[3] pipeline
[4] stall

۶. مسیرداده 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 لزوما 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. (اصطلاحا 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.

تمارین عملی

یادآوری

در تمرین‌های گذشته یک پردازنده‌ی MIPS ساده شده را طراحی کردیم. مانند پردازنده‌ی MIPS این پردازنده نیز سه نوع کدگذاری دستور دارد: Register, Immediate, Jump. تعریف هر کدام از این دستورات در زیر آمده است:

- **Register Instructions**: این دستورات همان طور که از اسم آن‌ها پیدا است برای زمانی استفاده می‌شوند که قرار است به کمک محتوای دو ثبات، یک ثبات دیگر را مقداردهی کنیم. این دستورات دارای فرمت زیر هستند:

opcode	rs	rt	rd	funct
4 bits	3 bits	3 bits	3 bits	3 bits

دقیقا مثل پردازنده‌ی MIPS در تمامی دستورات این نوع، ثبات opcode آن‌ها برابر صفر است و بر اساس مقدار سیگنال funct می‌توان نوع عملیات را تعیین کرد. جدول عملیات در زیر آمده است:

Mnemonic	Operation	funct
ADD	rd ← rs + rt	000
SUB	rd ← rs - rt	001
AND	rd ← rs & rt	010
OR	rd ← rs | rt	011
MULT	rd ← rs * rt	100
XOR	rd ← rs ^ rt	101
JR	PC ← rs	111

- **Immediate Instructions**: این دستورات خود سه نوع هستند:(۱) یا مسئول یک پرش شرطی[5] هستند، (۲) یا برای load و store استفاده می‌شوند (۳) یا اینکه برای انجام دادن یک عملیات با یک مقدار ثابت و ثبات هستند. فرمت این دستورات مانند شکل زیر است:

opcode	rs	rt	immediate
4 bits	3 bits	3 bits	6 bits

لیست این دستورات و opcodeهای آن در زیر آمده است:

Mnemonic	Operation	opcode
ADDi	rt ← rs + SIGN_EXTEND(immediate)	0010
SUBi	rt ← rs - SIGN_EXTEND(immediate)	0011
ANDi	rt ← rs & immediate	0100
ORi	rt ← rs | immediate	0101
SB	MEM[rs + SIGN_EXTEND(immediate)] ← rt	0110
LB	rt ← MEM[rs + SIGN_EXTEND(immediate)]	0111
BEQ	`if` (rt == rs): PC ← PC + SIGN_EXTEND(immediate << 1)	1000
BNQ	`if` (rt != rs): PC ← PC + SIGN_EXTEND(immediate << 1)	1000

در رابطه با این نوع دستورات به نکات زیر توجه کنید:

- دقت کنید که مقدار immediate در دستورات `ANDi` و `ORi` به هیچ وجه sign extend نمی‌شوند.
- اگر به یاد داشته باشید زمانی که در پردازنده میپس پرش نسبی [6] داشتیم به اندازه‌ی ۲ بیت مقدار immediate را شیفت می‌دادیم چرا که دستورات 4 byte aligned بودند. اما در اینجا از آنجا که دستورات 2 byte aligned

[5] conditional branch
[6] relative jump

هستند باید یک واحد مقدار immediate را شیفت دهید. در دستورات نیز علامت >> به معنای شیفت دادن است.

- **Jump Instructions**: این دستورات برای پرش استفاده می‌شوند. شکل این دستورات به صورت زیر است:

opcode	NOT USED	address
4 bits	5 bits	7 bits

همان طور که مشاهده می‌کنید در اینجا طول آدرس ۷ بیت است چرا که با توجه به طراحی ما و اندازه‌ی حافظه این پردازنده می‌تواند حداکثر ۱۲۸ دستور را در خود ذخیره و اجرا کند. همچنین همان طور که از اسم آن پیدا است بیت‌های NOT USED صرفا بدون استفاده هستند و به عبارتی dont care در نظر گرفته می‌شوند. این سری از دستورات شامل دو دستور زیر هستند:

Mnemonic	Operation	opcode
J	$PC \leftarrow address << 1$	1110
JAL	$R[7] \leftarrow PC + 2, PC \leftarrow address << 1$	1111

در نهایت نیز به این موضوع توجه کنید که تمامی مقادیر opcode و funct که نوشته شده‌اند پیشنهادی هستند و در صورت نیاز می‌توانید آن‌ها را تغییر دهید. اما در صورتی که آن‌ها را تغییر دهید حتما در گزارش خود ذکر کنید که دستورات شما چه opcode یا functهایی دارند.

صورت تمرین

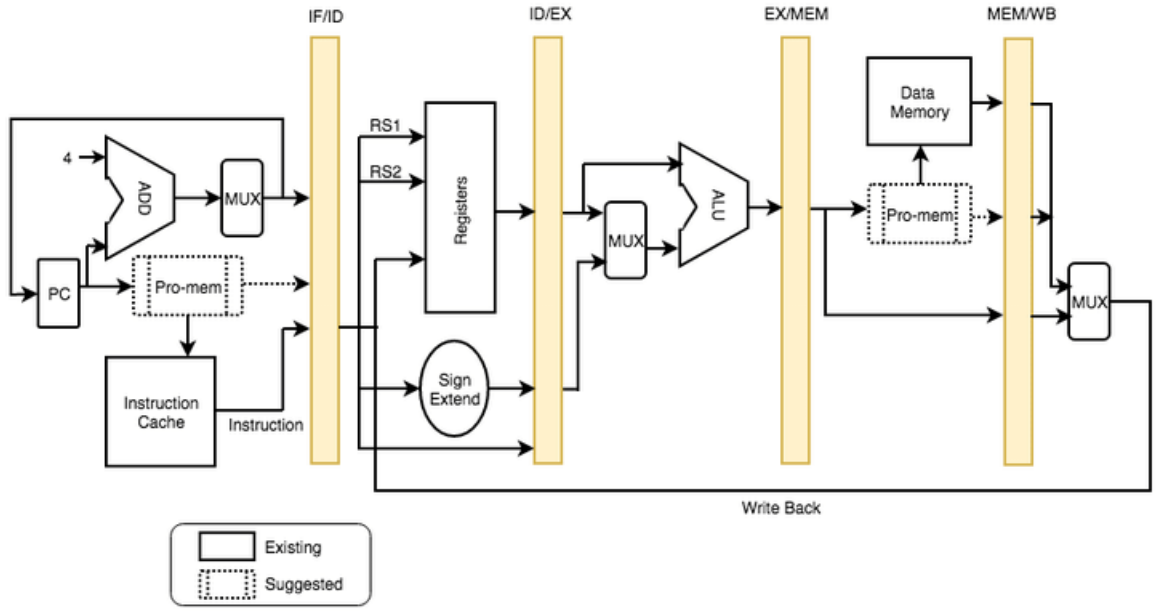
در این تمرین می‌خواهیم به پردازنده‌ای که در دو تمرین قبل ساخته‌ایم، یک خط‌لوله را اضافه کنیم به این صورت که این خط‌لوله ۵ مرحله به ترتیب زیر دارد:

1. IF : Instruction Fetch
2. ID : Instruction decode/register file read
3. EX : Execute/address calculation
4. MEM : Memory acess
5. WB : Write back

اضافه کردن این خط‌لوله قرار است منجر به تسریع پردازنده ما شود. می‌توانید به صورت ساده عکس زیر را برای خط‌لوله خود در نظر بگیرید.

گام‌های پیشنهادی

پیشنهاد می‌کنیم که خط‌لوله را مرحله به مرحله طراحی و پیاده‌سازی و تست کنید چرا که همانطور که می‌دانید با اضافه شدن خط‌لوله به مدار، تعداد سیم‌ها زیاد شده و در صورتی که قسمت به قسمت تست خود را انجام ندهید در دیباگ کردن آن به مشکل خواهید خورد. همچنین خط‌لوله شما باید Multi-Cycle را نیز پشتیبانی کند.



نحوه‌ی ارزیابی

برای ارزیابی یکی از برنامه‌هایی را که در گذشته نوشته بودید ابتدا بدون خطلوله بررسی کنید و wave form آن را نیز برای محاسبه تعداد چرخه‌ها بررسی کنید. سپس خطلوله را اضافه کرده و برنامه قبلی را اجرا کنید و wave form آن را نیز بررسی کنید. از تفاوت تعداد کلاک‌ها چه چیز را متوجه می‌شوید؟ از نظر زمانی آیا برنامه شما تغییری کرده است؟ آیا در صورت پیاده‌سازی فیزیکی این مدار (نه شبیه‌سازی مانند الان) در سرعت اجرای برنامه تسریعی خواهید دید؟

گزارش مورد نیاز

در گزارش خود باید نحوه پیاده‌سازی را ذکر کنید و همچنین ورودی و خروجی متناظر را نیز نشان دهید. همچنین جواب سوالات قسمت قبل را نیز باید در گزارش خود ذکر کنید. دقت نمایید که در صورتی که این فاز را به درستی انجام ندهید یا در آن اشکالی وجود داشته باشد قطعا در تمارین بعدی با مشکلاتی روبه‌رو می‌شوید.