به نام خدا

دانشکده مهندسی کامپیوتر

درس معماری کامپیوتر
نیم‌سال دوم ۰۴-۰۳
استاد: دکتر اسدی

تمرین سری هفتم

- پرسش‌های خود را در سامانه CW و تالار مربوط به تمرین مطرح نمایید.
- پاسخ سوالات را تایپ نمایید.
- پاسخ تمرین را به صورت یک فایل زیپ با فرمت HW1_401234567.zip آپلود کنید. فایل زیپ باید به صورتی باشد که پس از باز کردن آن بدون هیچ پوشه‌ای فایل‌های زیر با ساختار زیر قرار گرفته باشند:

```
.
|-- HW1T_401234567.pdf
|-- practical
    |-- HW1P_401234567_401234568.pdf
    |-- schematic.circ
    |-- verilog
        |-- circuit
        |   |-- main.v
        |   |-- ...
        |-- gates
        |   |-- ...
        |-- toplevel
            |-- ...
```

- در صورت عدم تطابق فایل آپلود شده با فرمت بالا، تمرین شما تصحیح **نخواهد شد**.
- پاسخ سوالات تئوری و گزارش تمرین‌های عملی باید به فرمت pdf باشد.
- هر دانشجو می‌تواند حداکثر سه تمرین را با دو روز تأخیر بدون کاهش نمره ارسال نماید.
- تمرینات عملی به صورت گروه‌های دو نفر تحویل داده شود.
- هر دو عضو گروه موظف هستند تمرینات خود را بارگذاری کنند.
- عواقب عدم تطابق بین پاسخ دو عضو گروه برعهده خودشان است.
- تحویل تمرین به صورت انگلیسی مجاز نیست. در صورت تحویل تمرین به صورت انگلیسی (حتی بخشی از تمرین) نمره تمرین موردنظر صفر در نظر گرفته می‌شود.
- در صورت مشاهده تقلب برای بار اول نمره هر دو طرف صفر می‌شود. در صورت تکرار نمره کل تمرینات صفر خواهد شد.
- استفاده از ابزارهایی مانند ChatGPT به منظور ابزار کمک آموزشی مجاز است به شرط آن که به خروجی آن اکتفا نشود.
- توجه شود که پروژه نهایی درس در گروه‌های چهار نفر تحویل گرفته می‌شود.
- سوالات با عنوان **اختیاری** نمره‌ای ندارند اما جواب دادن به آن‌ها کمک به‌سزایی در یادگیری درس می‌کند.

تمارین تئوری

۱. قطعه کد اسمبلی زیر را در نظر بگیرید. جدول ۱، خط زمانی اجرای دستورات این کد را در یک پردازنده خط لوله‌ای[1] نشان می‌دهد.

```
    MOVI R1, X      # R1 <- X
    MOVI R2, Y      # R2 <- Y
L1:
    MUL  R4, R1, R1     # R4 <- R1 × R1
    MUL  R1, R1, R2     # R1 <- R1 × R2
    ADD  R4, R5, R6     # R4 <- R5 + R6
    ADD  R5, R2, R4     # R5 <- R2 + R4
    SUBI R3, R1, 2048   # R3 <- R1 - 2048, set condition flags
    JNZ  L1             # Jump to L1 if zero flag is NOT set
    MUL  R1, R1, R2     # R1 <- R1 × R2
```

Instructions	**Cycles**															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
MOVI R1, X	F	D	E1	E2	E3	M	W									
MOVI R2, Y		F	D	E1	E2	E3	M	W								
MUL R4, R1, R1			F	D	-	E1	E2	E3	M	W						
MUL R1, R1, R2				F	-	D	E1	E2	E3	M	W					
ADD R4, R5, R6						F	D	E1	E2	E3	M	W				
ADD R5, R2, R4							F	D	-	-	E1	E2	E3	M	W	
SUBI R3, R1, 2048								F	-	-	D	E1	E2	E3	M	W
JNZ L1											F	D	-	-	E1	...
...	...															

Table 1: Execution timeline (F:Fetch, D:Decode, E:Execute, M:Memory, W:WriteBack)

(آ) مسیرهای مورد استفاده برای Data Forwarding بین مراحل مختلف خط لوله را مشخص کنید و دلیل آن را نیز بیان کنید. جواب شما باید به صورت زیر باشد.
میان E2 و E1 Data Forwarding وجود دارد چون در کلاک ششم به مقدار R5 نیاز داریم.

(ب) توضیح دهید که این پردازنده از interlocking به صورت نرم‌افزاری استفاده می‌کند یا سخت‌افزاری.

بر اساس داده‌های زیر به باقی موارد پاسخ دهید:

- $X = 4, Y = 2$
- Branch Predictor همیشه به درستی پیش‌بینی می‌کند.
- این پردازنده از interlocking سخت‌افزاری بهره می‌برد.

فرض کنید در چرخه T موارد زیر را داریم:

- مقدار ذخیره‌شده در ثبات R1 برابر با ۱۰۲۴ است.
- پردازنده دستور Nام را واکشی می‌کند که دستور MUL R4, R1, R1 است.

(ج) مقدار T را محاسبه کنید.

(د) مقدار N را محاسبه کنید. (تعداد دستورات واکشی‌شده تا چرخه Tام)

(ه) تعداد چرخه‌های لازم برای اجرا شدن کامل کد را محاسبه کنید.

[1] Pipelined Processor

۲. یک مسیر داده تک‌چرخه‌ای با طول زمانی چرخه $T = 20ns$ داریم. این معماری را با یک مسیر داده که شامل یک خط لوله n [۲] مرحله‌ای است جایگزین می‌کنیم. طول مدت هر مرحله در این مسیر داده $T_p = \frac{T}{n} + 0.05n$ خواهد بود.

با فرض این که دنباله طولانی از دستورات وارد این سخت‌افزار می‌شوند، حداکثر گذردهی‌ای [۳] که با تنظیم n می‌گیریم چقدر است؟

۳. با فرض اینکه دستورات اسمبلی زیر به ترتیبی که در جدول آمده‌اند اجرا می‌شوند، نموداری رسم کنید که مرحله اجرای هر دستور را در چرخه‌های ساعت بر اساس پایپ‌لاین استاندارد ۵ مرحله‌ای نشان دهد. اگر دستوری در هیچ مرحله‌ای در یک چرخه نیست، آن خانه را خالی بگذارید. مراحل پایپ‌لاین را با استفاده از حروف F، D، X، M و W نشان دهید. اگر به دلیل وجود خطر داده‌ای [۴] وقفه‌ای در مراحل وجود داشت، آن را با علامت *d نشان دهید. (فرض کنید از latch به جای FF استفاده می‌کنیم)

ابتدا، اجرای پایپ‌لاین را در حالتی که هیچ گونه دور زدن [۵] وجود ندارد، نشان دهید:

۲	۱	۰	۹	۸	۷	۶	۵	۴	۳	۲	۱	**Instructions**
												sub $2, $3, $1
												lw $5, 0($2)
												addi $4, $5, 1
												add $5, $3, $1

اکنون نشان دهید اگر پایپ‌لاین دارای **دور زدن کامل** باشد، چه اتفاقی می‌افتد:

۲	۱	۰	۹	۸	۷	۶	۵	۴	۳	۲	۱	**Instructions**
												sub $2, $3, $1
												lw $5, 0($2)
												addi $4, $5, 1
												add $5, $3, $1

در نهایت، عملکرد این قطعه کد در صورت وجود دور زدن چقدر بهبود یافته است؟ (به صورت درصدی بیان شود)

۴. فرض کنید قطعه کد $MIPS$ زیر بر روی یک پردازنده دارای خط لوله ۵ مرحله‌ای با $forwarding$ کامل و $branch\ prediction$، اجرا شود.

a.

```
Label1:  LW   R2,0(R2)
          BEQ  R2,R0,Label1    # Taken once, then not taken
          OR   R2,R2,R3
          SW   R2,0(R5)
```

b.

```
         LW   R2,0(R1)
Label1:  BEQ  R2,R0,Label2    # Not taken once, then taken
         LW   R3,0(R2)
         BEQ  R3,R0,Label1    # Taken
         ADD  R1,R3,R1
Label2:  SW   R1,0(R2)
```

[2] pipeline
[3] throughput
[4] data hazard
[5] bypassing

(آ) خط زمانی اجرای کدهای فوق را با فرض اینکه هیچ $delay\ slot$ وجود ندارد و پرش‌ها نیز در مرحله EX اجرا می‌شود، رسم کنید.

(ب) اکنون خط زمانی قسمت قبل را مجددا رسم کنید اما با این فرض که از $delay\ slot$ استفاده می‌شود. در این حالت، دستوری که بعد از پرش اجرا می‌شود یک $delay\ slot$ است.

۵. تابع زیر هر نویسه‌ی حروف کوچک (با کد ASCII بین ۹۷ و ۱۲۲) را که در رشته‌ی ورودی تهی‌پایان[6] وجود دارد، به حرف بزرگ تبدیل می‌کند.

```
toupper:
            lb $t2, 0($a0)
            beq $t2, $0, exit       # stop at end of string
            blt $t2, 97, next       # not lowercase
            bgt $t2, 122, next      # not lowercase
            sub $t2, $t2, 32        # convert to uppercase
            sb $t2, 0($a0)          # store in string
next:
            addi $a0, $a0, 1
            j toupper
exit:
            jr \$ra
```

فرض کنید این تابع با رشته‌ای فراخوانی شود که دقیقا شامل ۱۰۰ حرف کوچک باشد و پس از آن یک نویسه‌ی تهی[7] قرار داشته باشد.

(آ) چند دستور در هنگام فراخوانی این تابع اجرا خواهد شد؟

(ب) فرض کنید که یک پردازنده تک‌چرخه‌ای پیاده‌سازی کرده‌ایم، که زمان هر چرخه آن ۸ نانوثانیه است. چه مقدار زمان برای اجرای یک فراخوانی تابع نیاز است؟ مقدار CPI چقدر است؟

(ج) اکنون فرض کنید که پردازنده‌ی ما از یک خط لوله‌ی ۵ مرحله‌ای استفاده می‌کند، با ویژگی‌های زیر:

- هر مرحله یک چرخه ساعت زمان می‌برد.
- فایل ثبات می‌تواند در یک چرخه هم خوانده و هم نوشته شود.
- فرض کنید که ارسال مستقیم داده‌ها [8] هر زمان که ممکن باشد انجام می‌شود و در غیر این صورت وقفه [9] اضافه می‌شود.
- انشعابات [10] در مرحله‌ی ID حل می‌شوند و در ۹۰% مواقع به‌درستی پیش‌بینی می‌شوند.
- دستورالعمل‌های پرش به‌طور کامل در خط لوله قرار می‌گیرند، بنابراین نیازی به وقفه یا تخلیه [11] نیست.

با این فرضیات، چه تعداد چرخه به‌طور کلی برای اجرای فراخوانی تابع toupper نیاز است؟

(د) اگر زمان هر چرخه در طراحی خط‌لوله‌ای ۲ نانوثانیه باشد، عملکرد آن در مقایسه با پردازنده‌ی تک‌چرخه‌ای در بخش (ب) چگونه خواهد بود؟

۶. به سوالات زیر پاسخ دهید:

(آ) عملکرد پیشبینی‌کننده ۲ بیتی[12] را توضیح دهید.

[6] null-terminated
[7] null terminator
[8] Forwarding
[9] Stall
[10] Branches
[11] Flush
[12] 2-bit saturating counter

(ب) **اختیاری** - درباره انواع وابستگی داده زیر تحقیق کنید.

- WAR
- RAW
- WAW

(ج) **اختیاری** - راجع به Confidence-Based Prediction تحقیق کنید و تاثیر آن بر تعداد stall پردازنده را در حالت‌های مختلف پیشبینی درست یا غلط، و همچنین درصد تخمین درست پیشبینی‌کننده بررسی کنید.

(د) **اختیاری** - راجع به Branch target buffer تحقیق کنید و تاثیر آن بر تعداد stall پردازنده را بررسی کنید.

(ه) **اختیاری** - راجع به پیشبینی‌کننده‌های ترکیبی[۱۳] تحقیق کرده و دلیل برتری آن‌ها نسبت به پیشبینی‌کننده‌های ساده را بررسی کنید. (راهنمایی: حالتی را در نظر بگیرید که کد به صورت همزمان الگوهای کوتاه مدت و طولانی مدت پرش دارد.)

[13] Hybrid Branch Predictor

تمارین عملی

۱. در این تمرین قصد داریم پردازنده خود را به یک پردازنده خط لوله تبدیل کنیم. پردازنده شما باید به ۵ مرحله زیر تقسیم شود و نتایج حاصل از هر مرحله به مرحله بعدی منتقل کند:

- Instruction Fetch
- Instruction Decode
- Instruction Execution
- Memory Operation
- Write Back

در طراحی پردازنده به نکات زیر توجه کنید:

- پردازنده شما باید بتواند برنامه اجرا شده در تمرین ۶ را به درستی اجرا کند.
- همه دستورات در ۵ کلاک انجام می‌شوند (به جز دستورات محاسباتی سنگین مانند ضرب. در این حالت باید با قرار دادن حباب در خط لوله از بروز مشکل جلوگیری کنید).
- برای بررسی درستی عملکرد خط لوله، لازم است تا خروجی مرحله EXE (مقدار نهایی که به ثبات در این دستور نوشته خواهد شد) و مرحله DEC (مقدار ثبات های rs و rt خوانده شده) به عنوان خروجی مدار طراحی شده اضافه کنید تا سامانه داوری از آن برای بررسی درستی عملکرد خط لوله استفاده کند (فایل داوری را بررسی کنید).