به نام خدا

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

دانشکده مهندسی کامپیوتر

تمرین سری ششم

- 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 فشرده با فرمت HW1_401234567.zip آپلود کنید. فایل فشرده 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. فرض کنید پردازنده Multi-Cycle با نام P1 در اختیار داریم که دستورات load را در ۶ چرخه، دستورات store را در ۶ چرخه، دستورات Arithmetic را در ۲ چرخه و دستورات Branch را در ۲ چرخه انجام می‌دهد. برنامه A را در اختیار داریم که ۴۰% از دستورات آن را دستورات load، ۲۰% را دستورات sotre، ۳۰% را دستورات Arithmetic و ۱۰% باقی‌مانده را دستورات Branch تشکیل می‌دهد.

(آ) مقدار CPI برنامه A را زمانی که بر روی پردازنده P1 اجرا می‌شود، محاسبه کنید.

(ب) یک پردازنده جدید با نام P2 در اختیار داریم که فرکانس کلاک آن دو برابر فرکانس P1 است اما زمان اجرای همه دستورات به اندازه ۴ چرخه بیشتر شده است. اگر برنامه A را بر روی پردازنده P2 اجرا کنیم مقدار CPI چقدر می‌شود؟

(ج) استفاده از کدام پردازنده (P1 یا P2) بهینه‌تر است؟ میزان تسریع را محاسبه کنید.

(د) فرض کنید که می‌خواهیم پردازنده P1 را بهبود ببخشیم بدون اینکه فرکانس کلاک آن را تغییر دهیم. از میان دو مورد زیر فقط می‌توانیم یکی از آن‌ها را انتخاب کنیم:

۱. **ALU:** از یک ALU بهبود یافته استفاده کنیم که تعداد چرخه دستورات Arithmetic و Branch را نصف می‌کند.

۲. **LSU:** از یک واحد نامتقارن[1] برای load-store استفاده کنیم که تعداد چرخه دستورات load را نصف اما تعداد چرخه دستورات store را دو برابر می‌کند.

کدام یک از موارد بالا را برای بهبود پردازنده P1 به منظور اجرای برنامه A ترجیح می‌دهید؟ دلیل انتخاب خود را توضیح دهید.

۲. فرض کنید که میخواهیم، دستوری جدید به شکل max $rd, $rs, $rt را به مجموعه دستورات پردازنده Multi Cycle میپس بیفزاییم. عملکرد این دستور به این شکل است که بیشینه دو ثبات $rs و $rt را حساب میکند و نتیجه را در $rd ذخیره میکند. (در صورت تساوی، $rs را ذخیره میکند.)

(آ) برای افزودن این دستور، چه تغییراتی باید در datapath و واحد کنترل ایجاد شود.

(ب) اجرای این دستور را به صورت مرحله به مرحله (IF،ID،EX،MEM،WB) در معماری Multi Cycle پردازنده میپس توضیح دهید. در هر مرحله مقادیر سیگنال‌های کنترلی مورد نیاز را مشخص کنید.

(ج) فرض کنید که برنامه زیر روی یک پردازنده Multi Cycle اجرا میشود:

```
.data
array:  .word 8, 3
.text
main:
 la   $s0, array
 lw   $t0, 0($s0)
 lw   $t1, 4($s0)
 max  $t2, $t0, $t1
 sub  $t3, $t0, $t1
 beq  $t3, $zero,12
 add  $t2, $t2, $t3
 sw   $t2, 8($s0)
```

CPI متوسط را به دست بیاورید.

۳. با توجه به جدول زیر درباره تاخیرهای موجود در مسیر داده پردازنده MIPS به سوالات زیر پاسخ دهید:

[1] Asymmetric

Register Writeback	Memory	ALU	Register Read	Instruction Fetch
۵۰	۲۰۰	۱۰۰	۵۰	۲۰۰

(آ) برای هر یک از انواع دستور R-type، lw، sw، beq و J تعداد چرخه‌های ساعت و زمان مورد نیاز برای اجرا را در هر دو پردازنده single-cycle و multi-cycle بدست آورید.

(ب) فرض کنید که برنامه‌ای شامل ٪۲۵ دستور lw، ٪۱۰ دستور sw، ٪۵۲ دستورات ALU، ٪۱۱ دستور پرش شرطی و ٪۲ دستور پرش غیرشرطی است. مقدار CPI میانگین را در هر دو پردازنده single-cycle و multi-cycle بدست آورید.

(ج) میزان تسریع (Speedup) را در پردازنده multi-cycle نسبت به پردازنده single-cycle محاسبه کنید. آیا تسریعی حاصل می‌شود؟

۴. سه پردازنده P1 تا P3 داریم که همگی یک مجموعه دستورالعمل (ISA) مشترک را پیاده‌سازی می‌کنند. این ISA شامل سه دسته دستورالعمل اصلی است:

- **نوع X:** دستورالعمل‌های محاسباتی و منطقی (ALU operations)
- **نوع Y:** دستورالعمل‌های دسترسی به حافظه (Load/Store)
- **نوع Z:** دستورالعمل‌های کنترلی و پرش (Branch/Control Flow)

در طراحی هر سه پردازنده، یک **محدودیت زمانی پایه و ثابت** (T_{limit}) ناشی از مجموع زمان‌های آماده‌سازی (setup) و نگهداری (hold) در عناصر حیاتی مسیر بحرانی کلاک وجود داشته است که طراحان در هر سه نسل ملزم به رعایت آن بوده‌اند. افزایش سرعت کلاک در پردازنده‌های جدیدتر، از طریق بهینه‌سازی و کاهش تأخیر *سایر* بخش‌های مسیر داده حاصل شده است تا دوره تناوب کلی کلاک (که باید بزرگتر یا مساوی T_{limit} به علاوه‌ی تأخیر سایر بخش‌ها باشد) کاهش یابد.

جزئیات هر پردازنده به شرح زیر است:

۱. **پردازنده P۱ (طراحی اولیه):**

- فرکانس کاری: F مگاهرتز. (بالاترین فرکانس ممکن با توجه به T_{limit} و تأخیرهای مسیر در این طراحی).
- تعداد چرخه ساعت لازم برای اجرای دستور نوع X: 5 چرخه.
- تعداد چرخه ساعت لازم برای اجرای دستور نوع Y: 10 چرخه.
- تعداد چرخه ساعت لازم برای اجرای دستور نوع Z: 3 چرخه.

۲. **پردازنده P۲ (نسل بهبودیافته):**

- با بهینه‌سازی مسیر داده (ضمن ثابت ماندن T_{limit})، فرکانس کاری پایدار به $1.2F$ مگاهرتز افزایش یافته است (٪۲۰ سریعتر از P1).
- تعداد چرخه ساعت لازم برای اجرای دستور نوع X: 3 چرخه.
- تعداد چرخه ساعت لازم برای اجرای دستور نوع Y: 7 چرخه.
- تعداد چرخه ساعت لازم برای اجرای دستور نوع Z: 2 چرخه.

۳. **پردازنده P۳ (نسل پیشرفته):**

- با بازطراحی و بهینه‌سازی بیشتر (و همچنان با پایبندی به T_{limit})، فرکانس کاری به $1.5F$ مگاهرتز رسیده است (٪۵۰ سریعتر از P1).
- تعداد چرخه ساعت لازم برای اجرای دستور نوع X: 2 چرخه.
- تعداد چرخه ساعت لازم برای اجرای دستور نوع Y: 5 چرخه.
- تعداد چرخه ساعت لازم برای اجرای دستور نوع Z: 1 چرخه.

حال یک برنامه‌ی محک (benchmark) مشخص را روی این سه پردازنده اجرا می‌کنیم. نتایج اندازه‌گیری شده نشان می‌دهد که:

- زمان اجرای کل برنامه روی پردازنده P۲، به میزان 1.8 برابر کوتاه‌تر (یعنی 1.8 برابر سریعتر) از زمان اجرای آن روی پردازنده P۱ است.
- زمان اجرای کل برنامه روی پردازنده P۳، به میزان 1.875 برابر کوتاه‌تر (یعنی 1.875 برابر سریعتر) از زمان اجرای آن روی پردازنده P۲ است.

با توجه به این اطلاعات، ترکیب درصد دستورالعمل‌های مورد استفاده در این برنامه‌ی محک چگونه است؟ (چند درصد از کل دستورالعمل‌های اجرا شده در برنامه از نوع X، چند درصد از نوع Y و چند درصد از نوع Z هستند؟) همچنین یک کران بالا برای T_{limit} ارائه دهید.

۵. فرض کنید می خواهیم واحد کنترلی یک سیستم چند چرخه‌ای را پیاده سازی کنیم. FSM مربوط به حالات دستورات معماری این سیستم، دارای 12 FF است. تعداد بیت‌های Opcode برابر ۸ بیت است. تعداد سیگنال‌های کنترلی سیستم نیز ۳۰ عدد است.

(آ) فرض کنید برای پیاده‌سازی واحد کنترلی این سیستم، از یک ROM استفاده می‌کنیم. اندازه این ROM چقدر خواهد بود؟

(ب) اگر برای پیاده‌سازی واحد کنترلی آن از دو ROM مجزا از هم استفاده کنیم، اندازه هر کدام از این ROM ها چقدر خواهد بود؟

(ج) اگر دستورات به u-instruction ها تقسیم شده باشد و هر دستور حداکثر دارای ۸ u-instruction باشد، بخش الف و ب را مجددا حل کنید.

۶. MIPS یک معماری ثبات‌ـثبات است، به این معنی که منابع و مقصدهای محاسباتی باید حتماً ثبات باشند. اما فرض کنید می‌خواهیم یک دستور ثبات‌ـحافظه به مجموعه دستورات اضافه کنیم:

addm rd, rs, rt #rd = rs + Mem[rt]

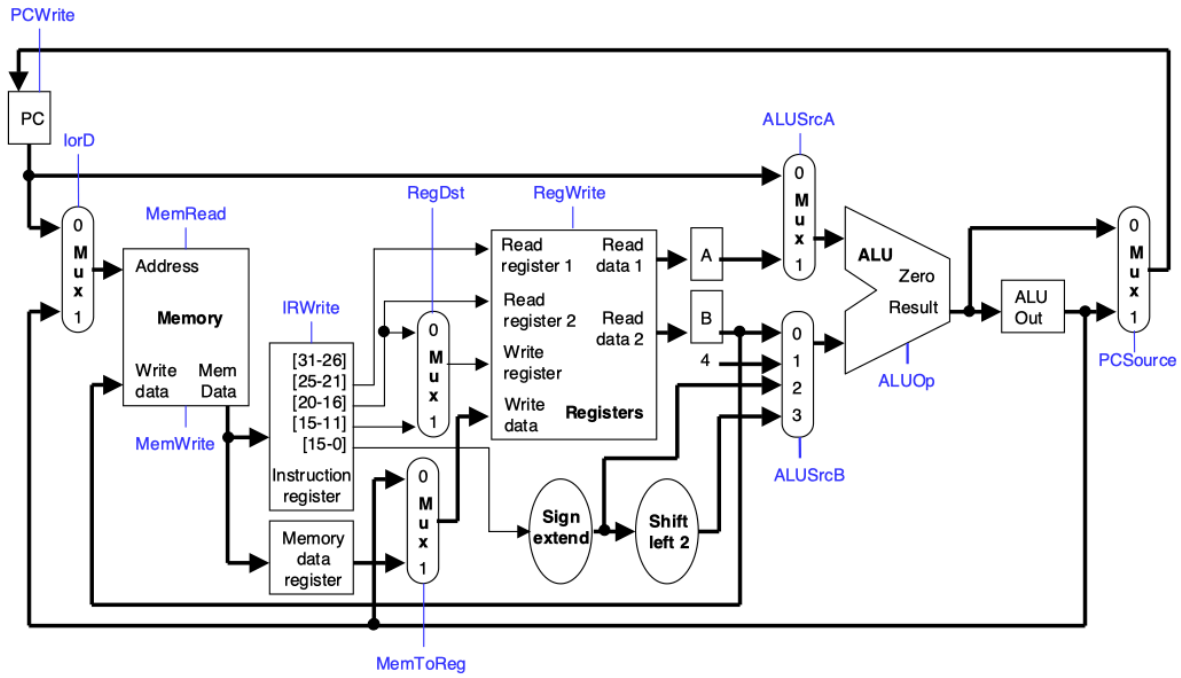
فرمت این دستور به صورت زیر است (توجه کنید که فیلدهای shamt و func استفاده نمی‌شوند):

Field	**op**	**rs**	**rt**	**rd**	**shamt**	**func**
Bits	۲۶–۳۱	۲۱–۲۵	۱۶–۲۰	۱۱–۱۵	۶–۱۰	۰–۵

جدول ۱: فرمت دستور addm

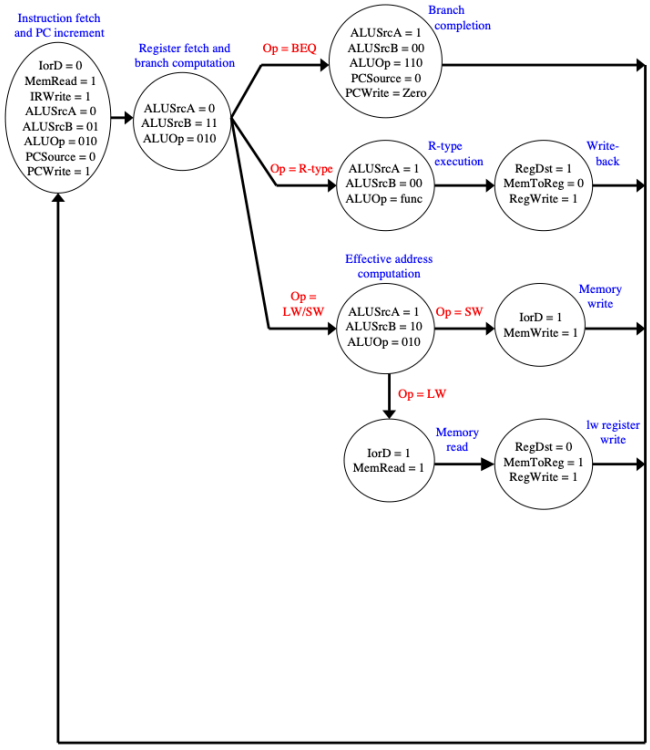
مسیر داده‌ای چندچرخه‌ای [۲] که در درس ارائه شده است در ادامه آورده شده است. فرض کنید که ALUOp = 010 منجر به انجام عملیات جمع می‌شود.

[۲] multicycle datapath



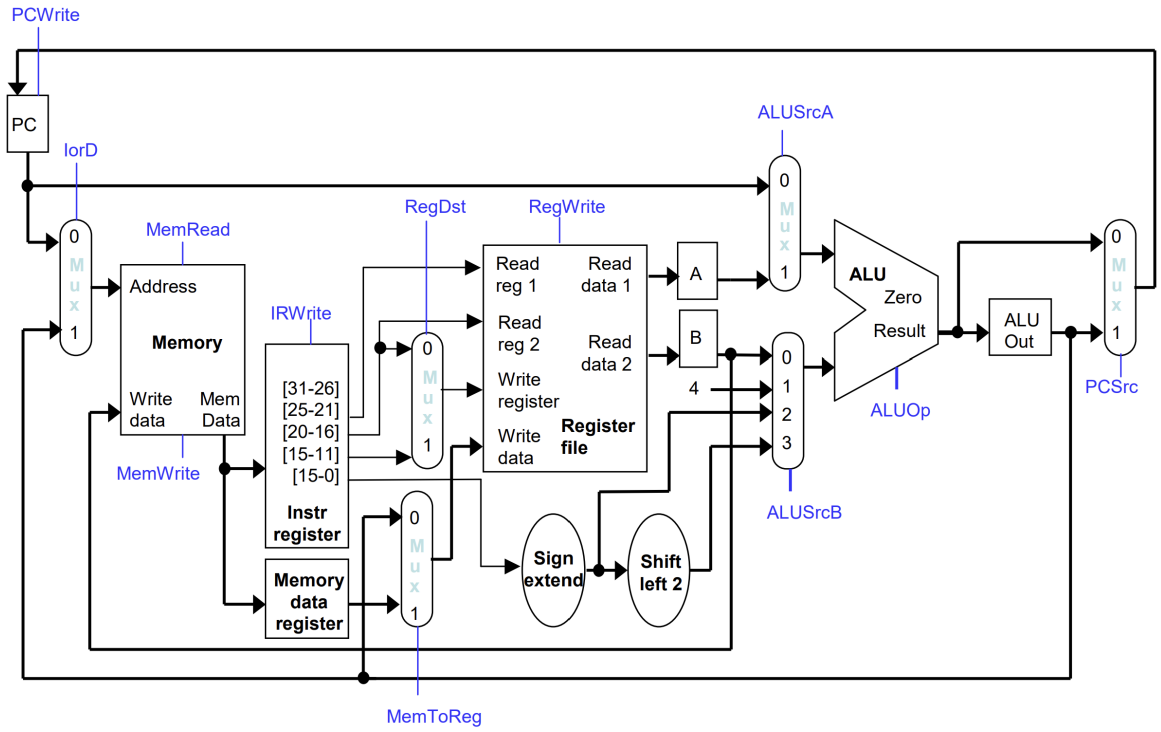
الف) تغییراتی که برای پشتیبانی از دستور addm در مسیر داده‌ای چندچرخه‌ای نیاز است را نشان دهید.

ب) ماشین حالت متناهی [۳] را برای دستور addm کامل کنید. حتماً هر سیگنال کنترلی جدیدی که اضافه کرده‌اید را نیز مشخص نمایید.



۷. **اختیاری** – با توجه به مسیر داده ای چندچرخه ای زیر به سوالات پاسخ دهید.

[۳] FSM



(آ) فرض کنید دستور زیر را اضافه می کنیم

```
jmem (rt), offset(rs)    # Memory[R[rs]+offset] = PC+4;
                         # PC = Memory[R[rt]]
```

که فرمت این دستور به صورت

Field	op	rs	rt	imm
Bits	31-26	25-21	20-16	15-0

می باشد، همچنین در نظر بگیرید که آدرس‌های R[rt] و (R[rs] + offset) مجزا و بدون همپوشانی هستند. تغییرات مورد نیاز را روی مسیر داده اعمال کرده و آن را رسم کنید.

(ب) فرض کنید که ALU می‌تواند عملیات max2 را انجام دهد (یعنی مقدار بزرگ‌تر از بین دو ورودی را بازگرداند). با در نظر گرفتن این ALU بهبودیافته، دستور max۴ را پیاده‌سازی کنید که مقدار بیشینه‌ی چهار ثبات را محاسبه کرده و در ثبات rd ذخیره می‌کند.

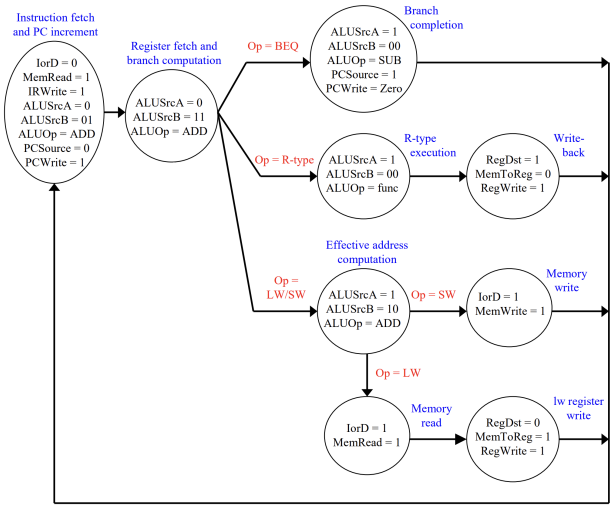
```
max4 rs, rt, rd, rm      # rd = max(rs, rt, rd, rm)
```

که فرمت این دستور به صورت

Field	op	rs	rt	rd	rm	func
Bits	31-26	25-21	20-16	15-11	10-6	5-0

می باشد. تغییرات مورد نیاز را روی مسیر داده اعمال کرده و آن را رسم کنید.

(ج) برای ۲ دستور گفته شده در قسمت های (الف) و (ب)، ماشین حالت متناهی زیر را برای آن کامل کنید.


Instruction fetch and PC increment
IorD = 0
MemRead = 1
IRWrite = 1
ALUSrcA = 0
ALUSrcB = 01
ALUOp = ADD
PCSource = 0
PCWrite = 1
Register fetch and branch computation
ALUSrcA = 0
ALUSrcB = 11
ALUOp = ADD
Op = BEQ
Branch completion
ALUSrcA = 1
ALUSrcB = 00
ALUOp = SUB
PCSource = 1
PCWrite = Zero
Op = R-type
R-type execution
ALUSrcA = 1
ALUSrcB = 00
ALUOp = func
Write-back
RegDst = 1
MemToReg = 0
RegWrite = 1
Effective address computation
Op = LW/SW
ALUSrcA = 1
ALUSrcB = 10
ALUOp = ADD
Op = SW
Memory write
IorD = 1
MemWrite = 1
Op = LW
Memory read
IorD = 1
MemRead = 1
lw register write
RegDst = 0
MemToReg = 1
RegWrite = 1

تمارین عملی

۱. در این تمرین قصد داریم پردازنده خود را به یک پردازنده چند چرخه تبدیل کنیم.
برای اینکار لازم است مراحل مختلف زیر هرکدام در کلاک جداگانه ای انجام شود :

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

همینطور باید از دستورات زیر پشتیبانی کنید :

```
add $1, $2, $3
addi $1, $2, $3
sub $1, $2, $3
and $1, $2, $3
or $1, $2, $3
xor $1, $2, $3
sll $1, $2, $3
srl $1, $2, $3
sra $1, $2, $3
nop
div $1, $2, $3
mul $1, $2, $3
mfhi $1
sll $1, $2, immediate
slti $1, $2, immediate
sw $1, n($2)
lw $1, n($2)
j label
bne $1, $2, label
jr $1
jal $1 label
```

همینطور توجه کنید که دستورات mul و jal و jr برای اولین بار افزوده شده‌اند.
در نهایت آزمون این پردازنده با کد زیر انجام می‌شود :

```
main:
addi sp, sp, -8
sw ra, (sp)
jal ra fact
addi a0, zero, 2
jal ra fact
add a0, zer0, v0
jal ra fact
add a0, zero, v0
lw ra, (sp)
addiu sp, sp, 8

fact:
addiu v0, zero, 1
beqz a0, done
```

```
nop
loop:
mul v0, a0, v0
addi a0, a0, -1
bnez a0, loop
nop
done:
jr ra
nop
```