به نام خدا

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

دانشکده مهندسی کامپیوتر

تمرین سری اول

- 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. می‌خواهیم دو کامپیوتر C1 و C2 را با یکدیگر از لحاظ عملکردی مقایسه کنیم. کامپیوتر C1 در سطح سخت‌افزار، از دستورات ممیز شناور[1] پشتیبانی می‌کند اما کامپیوتر C2 این قابلیت را نداشته و این دستورات را در سطح نرم‌افزار با سلسله دستورات دیگری پشتیبانی می‌کند. پردازنده هر دو کامپیوتر فرکانس ساعت[2] 400 MHz دارند. در هر دو پردازنده می‌خواهیم یک برنامه اجرا کنیم که دستورات زیر را شامل می‌شود:

نوع دستور	درصد استفاده دستورات در برنامه	زمان اجرای دستورات (تعداد ساعت)	
		C1	C2
جمع ممیز شناور	۱۶%	۶	۲۰
ضرب ممیز شناور	۱۰%	۸	۳۲
تقسیم ممیز شناور	۸%	۱۰	۶۶
سایر دستورات	۶۶%	۳	۳

(آ) مقدار CPI میانگین را برای این برنامه، در هر دو کامپیوتر محاسبه کنید.

(ب) مقدار MIPS را برای هر دو کامپیوتر C1 و C2 محاسبه کنید.

(ج) زمان اجرا شدن برنامه‌ای که شامل ۱۲۰۰۰ دستور است را در هر دو کامپیوتر محاسبه کنید.

(د) اگر زمان اجرای دستور جمع ممیز شناور، ضرب ممیز شناور و تقسیم ممیز شناور در کامپیوتر C2 به ترتیب $\frac{1}{2}$، $\frac{1}{4}$ و $\frac{1}{3}$ برابر شود، میزان تسریع را محاسبه کنید.

۲. برای کامپایل یک برنامه ثابت، از دو کامپایلر استفاده شده است. جدول زیر زمان اجرا و همچنین تعداد دستورات کد اسمبلی ایجاد شده توسط هرکدام از کامپایلرها را نشان می‌دهد.

کامپایلر A		کامپایلر B	
تعداد دستورها	زمان اجرا	تعداد دستورها	زمان اجرا
1×10^9	1s	1.2×10^9	1.4s

(آ) با فرض اینکه زمان چرخه ساعت[3] پردازنده ۱ نانوثانیه باشد، CPI متوسط را برای هر کدام از کامپایلرها محاسبه کنید.

(ب) CPIهای میانگین به دست آمده در بخش قبل را در نظر بگیرید، اما فرض کنید برنامه‌های کامپایل‌شده روی دو پردازنده مختلف اجرا می‌شوند. اگر زمان اجرا بر روی دو پردازنده یکسان باشد، پردازنده‌ای که کد کامپایلر A را اجرا می‌کند چقدر سریع‌تر از پردازنده‌ای است که کد کامپایلر B را اجرا می‌کند؟

(ج) کامپایلر جدیدی توسعه داده شده است که کد اسمبلی تولید شده توسط آن برای این برنامه دارای ۶۰۰ میلیون دستور با CPI میانگین ۱.۱ است. میزان تسریع با استفاده از این کامپایلر جدید در مقایسه با کامپایلر A و B در پردازنده اولیه در بخش الف چقدر است؟

۳. (آ) معماری‌های SIMD و MIMD را تشریح کرده و تفاوت‌های آن‌ها را توضیح دهید. از هرکدام از این معماری‌ها یک مثال بزنید که در اکثر کامپیوترهای شخصی استفاده می‌شود. برتری‌های این دو معماری را نسبت به یکدیگر بررسی کنید.

(ب) تحقیق کنید که out-of-order execution چیست و چه تاثیری بر عملکرد پردازنده دارد.

[1] Floating-point
[2] Clock
[3] clock cycle time

۴. یک پردازنده دارای ترکیب دستورالعمل‌های زیر است:

- دستورالعمل‌های Integer با CPI برابر ۳ که ٪۳۵ از کل دستورات را تشکیل می‌دهند.
- دستورالعمل‌های Load با CPI برابر ۵ که ٪۲۰ از کل دستورات را شامل می‌شوند.
- دستورالعمل‌های Store با CPI برابر ۴ که ٪۱۵ از دستورات را به خود اختصاص می‌دهند.
- دستورالعمل‌های Branch با CPI برابر ۲ که ٪۲۵ از دستورات هستند.
- دستورالعمل‌های Floating Point با CPI برابر ۱۰ که تنها ٪۵ از دستورات را تشکیل می‌دهند.

دو روش برای بهبود عملکرد پیشنهاد شده است:

۱. بهبود X: CPI دستورالعمل‌های Floating Point را ٪۵۰ کاهش می‌دهد.
۲. بهبود Y: دستورالعمل‌های Store را به دو نوع تقسیم می‌کند:
   - Fast Store با CPI برابر ۲ که ٪۶۰ از دستورات Store را شامل می‌شود.
   - Slow Store با CPI برابر ۵ که ٪۴۰ باقی‌مانده از دستورات Store را تشکیل می‌دهد.

(آ) CPI کلی پردازنده را در حالت اصلی محاسبه کنید.

(ب) CPI کلی را پس از اعمال هر دو بهبود X و Y را جداگانه محاسبه کنید.

(ج) کدام بهبود عملکرد بهتری دارد و این بهبود به چند درصد می‌رسد؟

۵. یک برنامه از سه کلاس دستورالعمل استفاده می‌کند:

کلاس	CPI	تعداد
A	۱	۲۰,۰۰۰
B	۲	۱۵,۰۰۰
C	۳	۵,۰۰۰

(آ) CPI متوسط و زمان کل CPU را محاسبه کنید (زمان چرخه کلاک = 0.5 نانوثانیه). اگر یک دستورالعمل جدید (کلاس D، $CPI = 0.5$) جایگزین ٪۳۰ از دستورالعمل‌های کلاس B شود، عملکرد چگونه تغییر می‌کند؟

(ب) یک بهینه‌سازی کامپایلر تعداد دستورالعمل‌ها را ٪۲۰ کاهش می‌دهد اما CPI را ٪۱۰ افزایش می‌دهد. برنامه اصلی دارای $IC = 100000$ و $CPI = 2.0$ است. آیا این بهینه‌سازی سودمند است؟ (فرکانس کلاک ثابت فرض شود.)

۶. یک پردازنده را در نظر بگیرید که ٪۴۰ از زمان اجرای آن صرف یک تابع خاص F می‌شود. یک بهبود[4] معرفی شده که F را ۵ برابر سریع‌تر می‌کند. اکنون به سوالات زیر پاسخ دهید:

(آ) Speedup نهایی سیستم را محاسبه نمایید.

(ب) اگر به جای افزایش سرعت به مقدار ۵ برابر، این مقدار x برابر شود، حد Speedup نهایی سیستم را زمانی که $x \to \infty$ در نظر بگیریم، محاسبه نمایید.

(ج) اکنون فرض کنید که دو بهبود مستقل از یکدیگر اعمال می‌شود:
- اولین بهبود، ٪۳۰ از زمان اجرا را ۴ برابر سریع‌تر می‌کند.
- دومین بهبود، ٪۲۵ زمان اجرا (بخشی مجزا) را ۳ برابر سریع‌تر می‌کند.

Speedup نهایی سیستم را بر اساس این دو بهبود بدست آورید.

۷. هدف این سوال مرور زبان اسمبلی MIPS است که در ادامه برای سوالات عملی از آن استفاده خواهید کرد. **دقت کنید که در هر بخش راه حل خود را مرحله به مرحله شرح دهید و فقط به جواب نهایی اکتفا نکنید.**

[4] Enhancement

(آ) آرایه زیر داده شده است، محتوای ثبات‌های[5] $v0 و $v1 را بعد از اجرای کامل کد تعیین کنید و مراحل حل خود را توضیح دهید.

```
array: .word 15, -19, 17, 20, -10, 12, 100, -5
        la $a0, array       # $a0 = 0x10010000
        addi $a1, $a0, 28
        move $v0, $a0
        lw $v1, 0($v0)
        move $t0, $a0
loop:   addi $t0, $t0, 4
        lw $t1, 0($t0)
        bge $t1, $v1, skip
        move $v0, $t0
        move $v1, $t1
skip:   bne $t0, $a1, loop
```

(ب) آرایه زیر داده شده است. محتوای آرایه را بعد از اجرای کد زیر را مشخص کنید.

```
array: .half 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
        la $a0, array
        li $a1, 6
        move $t0, $a0
        addi $t1, $a0, 12
loop:   lh $t3, ($t0)
        lh $t4, ($t1)
        sh $t3, ($t1)
        sh $t4, ($t0)
        addi $t0, $t0, 2
        addi $t1, $t1, 2
        addi $a1, $a1, -1
        bne $a1, $zero, loop
```

(ج) یک تابع به زبان MIPS بنویسید که تعداد 1 های موجود در ثبات $a0 را بشمارد و این مقدار را در داخل ثبات $v0 ذخیره کند.

[5] Register

تمارین عملی

۱. در این تمرین قصد داریم با ابزار Logisim-evolution و نحوه داوری آن در تمرینات آینده آشنا شویم. با دنبال کردن گام‌های زیر یک جمع/تفریق‌کننده سازگار با سیستم داوری طراحی خواهید کرد.

(آ) فایل jar نرم‌افزار Logisim-evolution را از این لینک دریافت کنید. سپس آن را به واسطه دستور زیر اجرا کنید. (برای اجرای نرم‌افزار به نسخه ۲۱ یا نسخه‌های بالاتر از java نیاز دارید)

```
java -jar logisim-evolution-3.9.0-all.jar
```

(ب) پروژه خود را با نام schematic.circ ذخیره کنید.

(ج) در هر تمرین ورودی و خروجی‌های مدار برای شما مشخص می‌شود. به طور معمول ورودی‌ها و خروجی‌ها را به واسطه پین‌ها نشان می‌دهیم. برای اضافه کردن یک پین به مدار، باید از پوشه Wiring در سمت راست، آن را به مدار اضافه کنید. به نکات زیر در رابطه با پین‌ها توجه کنید.

- اسکریپت‌ها و فایل‌های مورد نیاز برای داوری در این لینک قرار دارد.
- لازم است تمام پین‌های ورودی و خروجی با توجه به دستور تمرین، label **دقیقا** مطابق با label مشخص شده داشته باشند.
- در صورتی که پین مشخص شده خروجی است، باید از طریق منوی Properties آن را مشخص کرد.
- در صورتی پین چند بیتی است، لازم است تعداد بیت‌های آن از طریق منوی Properties مشخص بشود.

ورودی‌ها و خروجی‌های مدار به صورت زیر هستند:

Inputs:

- a (4 bits)
- b (4 bits)
- sub_notadd

Outputs:

- s (4 bits)
- cout

(د) یک مدار جمع/تفریق‌کننده مطابق با پین‌های مشخص شده طراحی کنید. توجه کنید که طراحی خود را به گونه‌ای انجام دهید که در تمام اجزای استفاده شده، ویژگی FPGA Supported فعال باشد. همچنین در صورتی که label یک قطعه به صورت Required for HDL است، یک label مناسب برای آن اختصاص دهید.

(ه) از منوی FPGA در بالا، گزینه ...Synthesize & Download را انتخاب کنید. در منوی باز شده، Setting را انتخاب کنید و در آنجا زبان سنتز را از VHDL به Verilog تغییر دهید. با استفاده از Execute و سپس انتخاب Done کد وریلاگ طراحی خود را تولید کنید. در صورت اجرای بدون مشکل، کد وریلاگ طراحی شما در آدرس زیر قرار خواهد گرفت. (توجه کنید که ماژول top level حتما برابر main باشد)

```
HOME/logisim_evolution_workspace/schematic/main/verilog
```

(و) ابزار Icarus Verilog را نصب کنید.

(ز) فایل‌های tb0.v و validate.sh را از فایل‌های ضمیمه CW دریافت کنید و به صورت زیر درستی عملکرد طراحی خود را بررسی کنید.

```
./validate.sh HW1/tb0.v HOME/logisim\_evolution\_workspace/schematic/
```

در صورتی که طراحی شما بدون مشکل باشد، در خروجی باید عبارت ACCEPTED را مشاهده کنید.

گزارشی کوتاه در حد یک صفحه، از گام‌های انجام تمرین به همراه تصویر از نحوه انجام آن، بنویسید.

نکته: برای تولید کدهای وریلاگ و تست مستقیم از اسکریپت‌های زیر نیز می‌توانید استفاده کنید:

```
./synthesize.sh schematic.circ
```

```
./synth_valid.sh bench.circ HW1/tb0.v
```