پروژه درسی

درس معماری کامپیوتر

نیم‌سال دوم ۱۴۰۱–۱۴۰۰

پروژه تعریف شده برای این درس شامل طراحی و پیاده‌سازی پردازنده‌ی چند سیکل RISC-V و اجرای یک برنامه با زبان ماشین روی آن است که در گروه‌های دونفری انجام و تحویل داده می‌شود. در مراحل سنتز فرض بر این است که طراحی برای برد DE2-115 انجام می‌شود. بخشی از پروژه شامل سنتز و شبیه‌سازی و نوشتن گزارش اجباری بوده و مکمل نمره‌ی نهایی است ولی قسمت پیاده‌سازی بر روی یک برد FPGA دلخواه اختیاری بوده و به‌عنوان نمره‌ی اضافه در نظر گرفته شده است.

طراحی واحد کنترل

قبل از آغاز توسعه کنترلر، به نمودارها و جدول‌های زیر نگاهی بیندازید. جدول‌ها در انتهای این سند ارائه شده‌اند.

- شکل 7.28 مرجع هریس که بلوک دیاگرام کنترلر چند سیکل را نشان می‌دهد
- شکل 7.48 مرجع هریس که نمودار حالت FSM اصلی کنترل چند سیکل را نشان می‌دهد
- جدول ۲ منطق دیکدر ALU را تعریف می‌کنند
- جدول ۳ منطق دیکدر دستورالعمل‌ها را تعریف می‌کنند

مدل سلسله مراتبی کنترلر چند سیکل را در زبان SystemVerilog توصیف کنید. هنگامی که خروجی‌ها اهمیتی ندارند، آن‌ها را روی ۰ قرار دهید تا برای ساده کردن تست، مقدار مشخصی داشته باشند.

ماژول کنترلر باید مطابق ساختار زیر باشد و باید از سلسله مراتب گفته شده در درس پیروی کند. به یاد داشته باشید که `op`،`funct3` و`funcct7b5` فیلدهایی بیتی از Instr هستند و `zero` یک خروجی ALU است.

```
module controller(input logic       clk,
                  input logic       reset,
                  input logic [6:0] op,
                  input logic [2:0] funct3,
                  input logic       funct7b5,
                  input logic       zero,
                  output logic [1:0] immsrc,
                  output logic [1:0] alusrca, alusrcb,
                  output logic [1:0] resultsrc,
                  output logic       adrsrc,
                  output logic [2:0] alucontrol,
                  output logic       irwrite, pcwrite,
                  output logic       regwrite, memwrite);
```

ماشین حالت اصلی یک ماشین مور است که op[6:0] را به‌عنوان ورودی می‌گیرد و مجموعه‌ای از خروجی‌ها را تولید می‌کند. خروجی‌هایی که در هر حالت نام برده شده‌اند اما مقداری به آن‌ها داده نمی‌شود، به‌طور ضمنی تنظیم می‌شوند (یعنی برابر ۱) و به هر خروجی که در هر حالت فهرست نشده است، باید مقدار صفر ۰ داده شود.

آزمون واحد کنترل با Test Bench

تولید بردارهای تست خوب اغلب سخت‌تر از نوشتن کد تحت آزمون است. در این راستا، فایل‌های controller.sv و controller.tv جهت سهولت کار در اختیار شما قرار گرفته است. آن‌ها را خوانده گزارش کنید که چگونه عمل می‌کنند.
با به‌کارگیری Modelsim، کنترلر خود را کامپایل کنید و مورد آزمون قرار دهید. مطمئن شوید که زمان شبیه‌سازی به‌اندازه‌ای طولانی هست تا پیامی دریافت کنید که گزارش دهد تمام تست‌ها با ۰ خطا تکمیل شده‌اند. در صورت بروز خطا توصیف خود را عیب‌یابی کنید.

پردازنده چند سیکل

قبل از تکمیل پردازنده، به نمودارهای زیر نگاهی بیندازید.

- شکل 7.27 مرجع هریس که پردازنده کامل چند سیکل را نشان می‌دهد.
- شکل 7.63 مرجع هریس سلسله مراتب سطح بالای پردازنده تک‌سیکل شامل اتصالات بین کنترل‌کننده، مسیر داده، حافظه دستورالعمل و حافظه داده را نشان می‌دهد. تفاوت پردازنده چند سیکل در این است که یک حافظه یکپارچه دارد و سیگنال‌های کنترلی آن متفاوت است، بنابراین باید این اتصالات را تغییر دهید. نموداری شبیه به این شکل ترسیم کنید که کنترلر، مسیر داده و ماژول‌های حافظه و اتصال آن‌ها را نشان می‌دهد. یک کادر دور ماژول `riscv` بکشید که کنترلر و مسیر داده را در بر گیرد. سیگنال‌های عبوری بین بلوک‌ها را نام‌گذاری کنید.

یک توصیف سلسله‌مراتبی از پردازنده در زبان SystemVerilog بنویسید. پردازنده باید ساختار زیر را داشته باشد. سیگنال‌های حافظه برای سهولت تست بیرون آورده تا شنود شوند. از واحد کنترل طراحی شده و هر بلوک ساختاری Verilog که نیاز دارید (مانند mux، flop، adder، ALU، register file و غیره) از پردازنده تک سیکل استفاده کنید.

```
module top(input logic         clk, reset,
           output logic [31:0] WriteData, DataAdr,
           output logic        MemWrite);
```

آزمون کلی پردازنده با Test Bench

فایل riscv_testbench.sv و کدهای تست (در قالب اسمبلی .s و زبان ماشین .txt) را مشاهده کنید. تست بنچ را مطالعه کنید تا متوجه شوید که چگونه موفقیت یا شکست آزمون را گزارش می‌کند.
حافظه شما باید کد تست را از فایل حافظه در هنگام راه‌اندازی اولیه با خط زیر بخواند.

```
initial $readmemh("memfile.txt", RAM);
```

پیش از آغاز شبیه‌سازی، پیش‌بینی کنید که پردازنده در هنگام اجرای سه دستورالعمل اول چه کاری باید انجام دهد. جدول ۱ برای اولین دستورالعمل برای شما پر شده است.
شکل موج‌های شبیه‌سازی را حداقل برای سیگنال‌های clk، reset، PC، Instr، state، SrcA، SrcB، ALUResult، Adr، WriteData و MemWrite به‌صورت خوانا تولید کنید. برای سهولت در خواندن، سیگنال‌های ۳۲ بیتی را به‌صورت مبنای شانزده نمایش دهید (سیگنال‌ها را انتخاب کنید و کلیک راست کنید، سپس Radix را انتخاب کنید). خروجی را با مقادیر مورد انتظارتان مقایسه کنید. ممکن است لازم باشد سیگنال‌های دیگری را برای درک بهتر عملکرد مدل خود و کمک به عیب‌یابی به شبیه‌سازی اضافه کنید. تمام اشکالات را پیدا کرده و برطرف کنید تا زمانی که مدل شما برنامه نمونه را مطابق انتظار اجرا کند و testbench گزارش موفقیت دهد.

قبل از اشکال‌زدایی، همه هشدارهای (warning) مرتبط را از Quartus و Modelsim برطرف کنید. این کار در زمان شما صرفه‌جویی می‌کند تا با دقت پیش‌بینی کنید که هر یک از سیگنال‌های موجود در شکل موج شما باید در هر سیکل چه کاری انجام دهند. به‌طور نظام‌مند اشکال‌زدایی کنید: از اولین عدم تطابق پیدا شده آغاز شود و به سمت عقب حرکت کنید تا زمانی که ورودی‌های خوب و خروجی‌های بد داشته باشید تا اشکال ایزوله شود و سپس آن را رفع کنید.

اگر همه موارد را بررسی کرده‌اید و پردازنده شما هنوز کار نمی‌کند، سعی کنید تمام خروجی‌های کنترلر را به شبیه‌سازی اضافه کنید و مطمئن شوید که هیچ‌کدام شناور یا X نیستند. اگر هنوز مشکل را پیدا نکرده‌اید، به شکل موج‌های پیش‌بینی‌شده خود در جدول ۱ مراجعه کنید و بررسی کنید که پردازنده در هر مرحله درست کار می‌کند. اگر چند دستورالعمل اول درست باشد، ممکن است لازم باشد جدول را تکمیل کنید تا مراحل بعد را پیش‌بینی کنید و بدانید بقیه برنامه چه کاری باید انجام دهد. (زمانی که جدول را برای چند دستورالعمل دیگر پر کردید، ممکن است الگوی مورد نظر را به‌دست آورید؛ فقط ورودی‌هایی را پر کنید که جالب هستند..)

جدول 1: پیش‌بینی مراحل اجرای دستورالعمل‌ها

Step	PC	Instr	State	Result	Result Notes
3	00	n/a	S0: Fetch	4	PC+4
4	04	""	S1: Decode	X	OldPC+Immediate
5	04	""	S8: ExecuteI	X	ALUResult = x0 (0) + 5 = 5
6	04	""	S7: ALUWB	5	Result = ALUOUT
7	04	""	S0: Fetch	8	PC+4
8	08	00c00193	S1: Decode	X	OldPC+Immediate
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					
28					
29					
30					
31					
32					
33					
34					
35					
36					
37					

38					
39					
40					
41					
42					
43					

به‌هنگام رفع عیب طراحی خود نکات زیر را در نظر داشته باشید.

- مطمئن شوید که عمل‌کرد ریزپردازنده را کاملاً متوجه شده‌اید. چنین سیستمی پیچیده‌تر از آن است که با سعی و خطا عیب‌یابی شود. باید بتوانید پیش‌بینی کنید در هر مرحله هر یک از سیگنال‌ها چه مقداری باید داشته باشند.
- اشکالات را با یافتن اولین نقطه‌ای در شبیه‌سازی که در آن سیگنالی مقدار نادرست دارد ردیابی کنید. اشکالات بعدی ممکن است برآمده از اولین اشکال باشند. جزئی از مدار را که خروجی نادرست تولید می‌کند پیدا کرده و ورودی‌هایش را به شبیه‌سازی اضافه کنید. این کار را تا پیدا کردن مبدا خطا تکرار کنید.

توسعه مجموعه دستورالعمل‌ها (اختیاری)

با برنامه‌ریزی مجدد واحد حافظه ریزدستورالعمل‌ها و کمترین تغییرات در مسیر داده و واحد کنترل می‌توان مجموعه دستورالعمل‌های بیشتری را پیاده‌سازی کرد. به‌عنوان نمونه مد آدرس‌دهیimmediate با امکان شیفت / چرخش را برای دستورات داده پیاده‌سازی کنید.

پیاده‌سازی بر روی FPGA (اختیاری)

پس از اطمینان از صحت عمل‌کرد پردازنده طراحی شده خود می‌توانید آن را بر روی یک برد FPGA دلخواه خود پیاده‌سازی کنید. برای این منظور لازم است تمام اجزای پردازنده بر روی FPGA پیاده‌سازی شوند. به‌منظور سنتز بهینه اجزایی نظیر حافظه، ممکن است لازم باشد به‌گونه‌ای که شرکت سازنده FPGA پیشنهاد می‌کند، آن‌ها را بازنویسی کنید.
چنانچه این بخش اختیاری را انجام می‌دهید، بهتر است یک واحد IO ساده (memory mapped) طراحی و به پردازنده اضافه کنید و ورودی‌ها و خروجی‌های آن را به کلیدها و LEDهای برد متصل کنید.
با توجه به اختیاری بودن این بخش، میزان کار اضافه انجام شده نسبت به بخش اجباری نمره اضافه شما را تعیین می‌کند.

گزارش

- گزارش نهایی که توسط گروه‌ها تحویل داده می‌شود باید شامل موارد زیر باشد:
  - توضیح دقیق مراحل طراحی سیستم و چالش‌هایی که با آن برخورد داشته‌اید.
  - توصیف سلسله مراتبی SystemVerilog برای واحد کنترل مطابق ساختار داده شده.
  - نموداری که سلسله‌مراتب بلوک حافظه، riscv، مسیر داده و کنترلر و نام تمام سیگنال‌های بین آن‌ها را نشان می‌دهد.
  - توصیف سلسله مراتبی SystemVerilog برای ماژول پردازنده سطح بالای شما (و زیر ماژول‌های آن) مطابق با ساختار داده شده.
  - جدول ۱ شامل سیگنال‌های کلیدی برای دستورالعمل‌های برنامه
  - شکل‌های موج شبیه‌سازی (ذکر شده در بالا clk ، reset، PC، Instr، state، SrcA، SrcB، ALUResult، Adr، WriteDataو - MemWrite همه به‌صورت هگزادسیمال برای سهولت در خواندن نمایش داده می‌شوند) آیا سیستم شما از آزمون تست شما عبور می‌کند؟ امواجی را که نشان می‌دهد مقدار درست در آدرس صحیح نوشته شده است خط بکشید یا برجسته کنید و مطمئن شوید که خوانا است.

- مشخصات سیستم سنتز شده برای برد مشخص شده شامل سرعت و مساحت اشغال شده روی تراشه و خروجی RTL Viewer و State Machine Viewer.
- توضیحات مربوط به بخش اختیاری (در صورت انجام).

- متن گزارش به‌صورت یک فایل PDF است که به شکلی مناسب حروف‌چینی شده است و کدهای نوشته شده برای پروژه پیوست آن شده است. می‌توانید برای وضوح بیشتر از نگاتیو شکل موج‌ها استفاده کنید.
- گزارش روز پیش از تحویل پروژه باید ارسال شده باشد.

تحویل

در روز تحویل هر دو عضو گروه با به همراه داشتن یک نسخه از گزارش پروژه و همچنین نمونه سخت‌افزاری پیاده‌سازی شده (در صورت انجام بخش اختیاری) برای تحویل مجازی مراجعه می‌کنند.

اعضای گروه در ابتدا یک گزارش شفاهی کوتاه (در حد ۴-۳ دقیقه) در مورد پروژه ارائه می‌کنند که شامل نکات مهم، چالش‌ها، شیوه انجام کار و انتخاب پارامترها می‌باشد.

پس از آن گروه شبیه‌سازی سیستم را انجام خواهد داد و توضیحات لازم را ارائه خواهد نمود. شبیه‌سازی باید به‌روشنی مراحل اجرای چند دستورالعمل را به‌طور صحیح نشان دهد.

در مرحله بعد چنانچه گروه پیاده‌سازی سخت‌افزاری روی برد FPGA را نیز انجام داده باشد، آن را نمایش می‌دهند. نحوه ارائه این بخش بدین‌سان است که برد را برنامه‌ریزی کرده و اجرای یک برنامه کوتاه را روی آن نمایش دهد.

دقت کنید که وظیفه تک تک اعضای گروه است که کیفیت کار انجام شده و میزان مشارکت خود را به‌هنگام تحویل اثبات کنند. در صورت سکوت هر یک از اعضا هنگام جلسه تحویل طبیعی است که نمره‌ای به آن‌ها تعلق نخواهد گرفت.

موفق باشید

عطارزاده

ALUOp	funct3	op_5, $funct7_5$	Instruction	$ALUControl_{2:0}$
00	X	X	lw, sw	000 (add)
01	X	X	beq	001 (subtract)
10	000	00, 01, 10	add	000 (add)
	000	11	sub	001 (subtract)
	010	X	slt	101 (set less than)
	110	X	or	011 (or)
	111	X	and	010 (and)

جدول ۲: منطق دیکدر ALU

Instruction	Opcode (*op*)	$ImmSrc_{1:0}$
R-type	0110011	XX
I-type	0010011	00
lw	0000011	00
sw	0100011	01
beq	1100011	10
jal	1101111	11

جدول ۳: منطق دیکدر دستورالعمل‌ها برای ImmSrc