پروژه درسی

درس معماری کامپیوتر

نیم‌سال دوم ۹۸-۹۷

پروژه تعریف شده برای این درس شامل طراحی و پیاده‌سازی پردازنده‌ی ARM با روش ریزبرنامه‌ای است که در گروه‌های دو نفری 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 به عنوان نمره‌ی اضافه در نظر گرفته شده است.

توضیح

با مفهوم ریزعملیات[۱] آشنا شده‌اید. ریزبرنامه[۲] و ریزکد[۳] (یا ریزدستورالعمل[۴]) مفاهیمی مرتبط اما متفاوت با ریزعملیات هستند. در درس، روش پیاده‌سازی مستقیم ماشین حالت در سخت‌افزار برای پردازنده چندسیکل استفاده شد. دیدیم که در ساده‌ترین حالت، به عملیات کوچک‌تری که در هر گام انجام می‌شود یک ریزعملیات گفته می‌شود. روش دیگر برای پیاده‌سازی واحد کنترل پیاده‌سازی به‌صورت ریزبرنامه است که در این حالت گام‌های کوچک‌تری که برای اجرای یک دستورالعمل باید طی شوند، به‌جای پیاده‌سازی مستقیم سخت‌افزاری، در قالب یک سری ریزکد در یک حافظه معمولا فقط خواندنی[۵] درون واحد کنترل ذخیره می‌شوند. در این حالت، واحد کنترل برای اجرای هر دستورالعمل پردازنده، ریزکدها را یکی یکی از حافظه ROM خوانده تا دو عمل ترتیب ریزدستورالعمل‌ها[۶] و اجرای ریزدستورالعمل‌ها[۷] را انجام دهد. منظور از ترتیب ریز دستورالعمل‌ها، انتخاب ریزدستورالعمل بعد برای اجرا و منظور از اجرای ریز دستورالعمل، تولید سیگنال‌های کنترلی مورد نیاز برای مسیر داده است. **برای فهم بهتر مفهوم واحد کنترل ریزبرنامه‌ای، فصل ۲۱ مرجع استالینگز را مطالعه کنید.**

پردازنده شما باید مطابق طراحی انجام شده در کتاب درسی (شکل 7.30 مرجع هریس) باشد و دستورات ADD، SUB، AND و ORR (با عمل‌وندهای رجیستر و immediate ولی بدون شیفت)، LDR و STR (با آفست immediate مثبت) و B را اجرا کند.

پردازنده‌ی چند سیکل از واحدهای controller، datapath و mem تشکیل شده است. واحد mem هم دستورالعمل‌ها و هم داده را نگاه می‌دارد و واحد controller به‌جای طراحی سخت‌افزاری دیده شده در درس، به‌روش ریزبرنامه‌ای پیاده‌سازی می‌شود (مشابه شکل 21.4 مرجع استالینگز). در نظر داشته باشید که از رجیسترها تنها در صورتی استفاده کنید که زمان‌بندی مدار نسبت به پیاده‌سازی ماشین حالت تغییر نکند.

---

۱ Micro-operation

۲ Micro-program

۳ Micro-code

۴ Micro-instruction

۵ Read-only Memory (ROM)

۶ Micro-instruction sequencing

۷ Micro-instruction execution

فایل `arm_uprog.sv` در اختیار شما قرار گرفته است که در آن ماژول `arm` حاوی واحدهای `datapath` و `controller` است که در این پروژه آن‌ها را تکمیل خواهید کرد.

طراحی واحد کنترل

واحد کنترل بخش کلیدی پردازنده است و از ماژول‌های `condlogic` و `decode` که دربردارنده‌ی پیاده‌سازی ریزبرنامه‌ای واحد کنترل است تشکیل شده. به‌هنگام ریست، رجیستر کنترل آدرس از آدرس مربوط به اولین ریزدستورالعمل (`FETCH`) شروع می‌کند و طبق ترتیب ریزعملیات مشخص شده در کتاب درسی (شکل 7.41 مرجع هریس) پیش می‌رود.

ورودی‌ها، خروجی‌ها و بخشی از کد پیاده‌سازی ماژول‌های `controller`، `decode` و `condlogic` در فایل داده شده موجود است. کد SystemVerilog داده شده برای ماژول `controller` و اجزای تشکیل‌دهنده آن را تکمیل کنید. هر جا لازم بود، می‌توانید از کد پردازنده‌ی تک‌سیکل نیز استفاده کنید.

قالب کلی کلمه کنترلی ریزدستورالعمل

ریزدستورالعمل‌ها تحت قالب کلی ریزدستورالعمل‌های افقی (شکل 21.1 مرجع استالینگز) ساماندهی و در حافظه کنترلی ذخیره می‌شوند. با توجه به‌این‌که برای حافظه ROM کنترلی ۳۲ خط در نظر گرفته شده است، ۵ بیت برای آدرس ریزدستورالعمل بعدی اختصاص می‌یابد.

Control Signals													Branch Condition			Branch Target
FlagW	PCS	NextPC	RegW	MemW	IRWrite	AdrSrc	ResultSrc [1:0]	ALUSrcA [1:0]	ALUSrcB [1:0]	ImmSrc [1:0]	RegSrc [1:0]	ALUControl [1:0]	Op	Funct0	Funct5	*Micro-instruction* Address [4:0]

تولید سیگنال‌های کنترلی

پیش از آغاز به طراحی پردازنده‌ی چندسیکل ARM خود، باید سیگنال‌های کنترلی صحیح متناظر با هر ریزدستورالعمل را به‌دست آورید. در ریزدستورالعمل‌های افقی[۸]، این سیگنال‌ها به‌همراه آدرس و شرط پرش به ریزدستورالعمل بعد مشترکاً فرمت یک ریزدستورالعمل را که در هر خط حافظه کنترلی ذخیره می‌شود، تعیین می‌کنند. برخی خروجی‌هی واحد کنترل در جدول 1 رسم شده است. آن را تکمیل کنید. برای هر ریزدستورالعمل کلمه کنترلی را نیز در مبنای ۱۶ به‌دست آورید. در انجام این مرحله دقت کنید چرا که رفع عیب مدارات نادرست طراحی شده بسیار دشوار است.

آزمون واحد کنترل

یک testbench به‌نام `controllertest.sv` برای آزمون واحد کنترل خود طراحی کنید. هر یک از دستوراتی که پردازنده باید اجرا کند را مورد آزمون قرار دهید: دستورات ADD، SUB، AND و ORR (با عمل‌وندهای رجیستر و immediate ولی بدون شیفت)، LDR و STR (با آفست immediate مثبت) و B. مطمئن شوید که هم انشعاب[۹] انجام شده[۱۰] و انجام نشده[۱۱] را آزمون کنید.

ماژول testbench شما ورودی‌های واحد کنترل را اعمال می‌کند که شامل `clk`، `reset`، `Instr[31:12]` و `ALUFlags[3:0]` است (دقت کنید که `Instr[31:12]` شامل `Cond`، `Op`، `Funct` و `Rd` است). شکل موج خروجی‌ها را به‌صورت تصویری بررسی کنید و مطمئن شوید درست عمل می‌کنند. به ازای هر دستورالعمل یک شکل موج شبیه‌سازی ذخیره کرده و در گزارش خود درج کنید.

---

[۸] Horizontal micro-instruction

[۹] Branch

[۱۰] Taken

[۱۱] Not taken

جدول 1: برخی خروجی‌های واحد ریزبرنامه

Micro-instruction (Name)	NextPC	Branch	MemW	RegW	IRWrite	AdrSrc	ResultSrc$_{1:0}$	ALUSrcA$_{1:0}$		ALUSrcB$_{1:0}$	ALUOp	(Partial) Control Word
0 (Fetch)	1	0	0	0	1	0	10	0	1	10	0	0x114C
1 (Decode)	0	0	0	0	0	0	10	0	1	10	0	0x004C
2 (MemAdr)												
3 (MemRead)												
4 (MemWB)												
5 (MemWrite)												
6 (ExecuteR)												
7 (ExecuteI)												
8 (ALUWB)												
9 (Branch)												

طراحی مسیر داده

در بخش دوم پروژه اجزای مسیر داده را به پردازنده چندسیکل اضافه کرده و پردازنده نهایی را با یک برنامه نمونه آزمون می‌کنید.

برنامه نمونه

در این پروژه از برنامه نمونه‌ای که در تمرین‌ها برای آزمون پردازنده تک‌سیکل استفاده شده، بهره می‌بریم (شکل 7.60 کتاب درسی). در جدول 2 (که تا حدی تکمیل شده است) مراحل اجرای دستورالعمل‌ها را برای هر حالت مشخص کنید. حتما این کار را پیش از شبیه‌سازی انجام دهید تا از اشتباهات بعد جلوگیری شود.

دقت کنید که دستورالعمل (`Instr`) طی ریزدستورالعمل `FETCH` واکشی می‌شود و تا ریزدستورالعمل `DECODE` به‌هنگام نمی‌شود. هنگامی که `ALUResult` استفاده نمی‌شود (مثلا در مرحله Writeback دستورالعمل‌ها)، بهتر است به‌جای پیش‌بینی مقدار استفاده نشده پردازنده، از don't care (x) استفاده کنید.

جدول 2: پیش‌بینی مراحل اجرای دستورالعمل‌ها

Cycle	Reset	PC	Instr	uInstr.	SrcA	SrcB	ALUResult
1	1	00	0	FETCH	0	4	4
2	0	04	SUB E04F000F	DECODE	4	4	8
3	0	04		EXECUTER	8	8	0
4	0	04		ALUWB	x	x	x
5	0	04		FETCH	4	4	8
6	0	08	ADD E2802005	DECODE	8	4	C
7	0	08		EXECUTEI	0	5	5
8	0	08		ALUWB	x	x	x
9	0	08		FETCH	8	4	C
10	0						
11	0						
12	0						
13	0						
14	0						
15	0						
16	0						
17	0						
18	0						
19	0						
20	0						
21	0						
22	0						

23	0						
24	0						
25	0	18		FETCH	18	4	1C
26	0	1C	ADD E0855004	DECODE	1C	4	20
27	0	1C		EXECUTER	4	7	B
28	0	1C		ALUWB	x	x	x
29	0						
30	0						
31	0						
32	0						
33	0						
34	0						
35	0						
36	0						
37	0						
38	0						
39	0						
40	0						
41	0						
42	0						
43	0						
44	0						
45	0						
46	0						
47	0						
48	0						
49	0						
50	0						
51	0						
52	0						
53	0						
54	0						
55	0						
56	0						
57	0						
58	0						
59	0						
60	0						
61	0						
62	0						
63	0						
64	0						
65	0						
66	0						
67	0						
68	0						
69	0						
70	0						
71	0						
72	0						
73	0						
74	0						

طراحی و تکمیل اجزای مسیر داده

با بهره‌گیری از اجزایی که در درس معرفی شد (نظیر ALU، مالتی‌پلکسر، رجیستر و ...)، کد مسیر داده (datapath) را به زبان SystemVerilog تکمیل کنید.

تمام رجیسترها دارای ورودی Reset برای تنظیم حالت پیش‌فرض (۰) هستند. برخی رجیسترها نظیر IR و PC دارای ورودی Enable نیز هستند.

پردازنده تکمیل شده را با برنامه نمونه و با استفاده از testbenchی که در تمرین‌ها برای پردازنده تک‌سیکل استفاده کردید، شبیه‌سازی کنید. سیگنال Reset ابتدا مقدار ۱ دارد. سیگنال‌های `clk`، `reset`، `PC`، `Instr`، `state`، `srcA`، `srcB` و `ALUResult` را شکل موج‌های شبیه‌سازی اضافه کنید و نتیجه شبیه‌سازی پردازنده را با جدول 2 مقایسه کرده و خطاها را اصلاح کنید.

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

- مطمئن شوید که عمل‌کرد ریزپردازنده را کاملا 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 ناشی 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 آدرس‌دهی immediate 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 توصیه می‌کند، آن‌ها را بازنویسی کنید.
در صورتی که 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.
  - نسخه تکمیل شده جدول 1 و جدول 2.
  - فایل سورس اصلی پردازنده طراحی شده `arm_multi.sv`.
  - فایل سورس testbench بخش واحد کنترل (`controllertest.sv`).
  - شکل موج‌های خروجی شبیه‌سازی واحد کنترل شامل سیگنال‌های به‌ترتیب `CLK`، `Reset`، `Cond`، `OP`، `Funct`، `Rd`، `ALUFlags`، `ALUControl`، `ImmSrc`، `RegSrc`، `RegWrite`، `MemWrite`، `PCWrite`، `state` و کلمه کنترلی همه در مبنای ۱۶ برای تمام دستورالعمل‌ها.
  - شکل موج‌های خروجی شبیه‌سازی پردازنده شامل سیگنال‌های به‌ترتیب clk، reset، PC، Instr، state و ALUResult همه در مبنای ۱۶ هنگام اجرای برنامه نمونه.
  - مشخصات سیستم سنتز شده برای برد مشخص شده شامل سرعت و مساحت اشغال شده روی چیپ و خروجی RTL Viewer و State Machine Viewer.
  - توضیحات مربوط به بخش اختیاری (در صورت انجام).
- متن گزارش به صورت 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 مراجعه می‌کنند.

اعضای گروه در ابتدا یک گزارش شفاهی کوتاه (در حد ۱-۲ 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 به‌وضوح مراحل اجرای چند دستورالعمل را به طور صحیح نشان دهد.
در مرحله بعد در صورتی که گروه پیاده‌سازی سخت‌افزاری روی برد FPGA را نیز انجام داده باشد، آن را نمایش می‌دهند. نحوه ارائه این بخش به این ترتیب 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 به هنگام 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