تمرین سری پنجم

- تمرین‌های خود را در قالب فایل تایپ‌شده و ذخیره شده به فرمت *PDF* به صورت یک فایل *Zip* با نام (*1STDID_HW*) که *STDID* شماره‌ی دانشجویی شما است، در صفحه‌ی درس در *CourseWare (CW)* بارگذاری کنید.
- دانشجویان مجاز هستند دو تمرین را حداکثر با دو روز تاخیر ارسال کنند.
- سؤالات خود را صرفاً در فروم مربوطه در *CW* بپرسید.

تمرین سری پنجم

1- طراحی می‌خواهد ریز معماری یک پردازنده تک سیکلی (*Single Cycle*) را به خط لوله‌ای تبدیل کند. سیکل ساعت طراحی اول 7 نانوثانیه می‌باشد. پس از تقسیم ریزمعماری به چند قسمت، تاخیر هر بخش بدین ترتیب محاسبه می‌شود: $WB = 1.5ns, MEM = 2ns, EX = 1ns, ID = 1.5ns, IF = 1ns$ تاخیر ثبات‌های میانی خط لوله برابر 0,1 نانوثانیه می‌باشد. با فرض اینکه به ازای هر 4 دستور، یک *stall* در خط لوله رخ دهد، تسریع ریز معماری خط لوله‌ای، نسبت به ریز معماری تک سیکلی چند است؟

2- یک بسته نرم‌افزاری روی تک‌پردازنده A نیاز به T ثانیه برای اجرا دارد. بخشی از این نرم‌افزار به روش موازی نوشته شده‌است و این بخش می‌تواند از امکانات کامپیوتری که از 4 پردازنده نوع A ساخته شده استفاده کند و با سرعت 4 برابر نسبت به قبل اجرا شود. چند درصد از برنامه باید از نوع موازی باشد تا وقتی کل برنامه روی کامپیوتر 4 پردازنده اجرا کنیم نسبت به قبل افزایش سرعتی برابر با 2 داشته باشیم؟

3- چارت عملیاتی در یک سیستم دیجیتال دارای 100 جعبه انتقال و 14 جعبه شرطی متفاوت است. اگر تعداد سیگنال‌های کنترلی سیستم 50 عدد باشد و ریزبرنامه واحد کنترل شامل 180 ریزدستور باشد، استفاده از حافظه نانو حداقل چند بیت صرفه جویی در حجم ریز برنامه به وجود می‌آورد؟

4- در یک سیستم دیجیتال تعداد سیگنال‌های کنترل 200 عدد است. بررسی الگوی فعالیت سیگنال‌ها در ریزدستورات داخل ریزحافظه، نشان می‌دهد که امکان افراز این سیگنال‌ها به یک دسته 120 تایی، یک دسته 60 تایی و یک دسته 20 تایی به طوری که در هر دسته فقط یک سیگنال فعال باشد، وجود دارد. اگر حجم ریزحافظه 1M ریز دستور باشد، با استفاده از ریزدستورات عمودی چند بیت در ریزحافظه صرفه‌جویی می‌شود؟

5- یک پردازنده *MIPS* دارای 5 مرحله برای خط لوله است که عبارتند از:

Fetch, Decode, Execute, Memory, WriteBack

با در نظر گرفتن *full forwarding* به سوالات زیر پاسخ دهید:

الف) کد زیر را در نظر بگیرید:

```
lw $6, 0($7)
add $8, $9, $10
sub $11, $6, $8
```

عملوندهایی که در دستور تفریق در مرحله *execute* مورد استفاده قرار میگیرند در کدام مرحله تولید می‌شوند؟ با رسم جدول نشان دهید.

6- کد *MIPS* زیر را در نظر بگیرید:

```
sub $a3, $a2, $a1
and $v1, $a3, $v4
or $v2, $v5, $a3
lw $v4, 100($a3)
sw $v2, 100($a3)
```

الف) مخاطرات ممکن و نوع آن‌ها را مشخص کنید.

ب)در صورتی که متوقف (*stall*) کردن تنها راه حل برای برطرف کردن مخاطرات باشند اجرای دستورات چند *clock cycle* بیشتر به طول می‌انجامد؟

7- جدول زیر را در نظر بگیرید.

IF	ID	EX	MEM	WB
400ps	200ps	300ps	500ps	200ps

الف) اگر معماری ما *single-cycle* باشد دوره کلاک چقدر است؟

ب) اگر معماری *multi-cycle* باشد دوره کلاک چقدر می‌تواند باشد؟

8- با توجه به ترکیب زیر از دستورات و تاخیرات داده‌شده به سوالات پاسخ دهید. (تعداد کل دستورات 1 میلیون است)

IF	ID	EX	MEM	WB
400ps	200ps	300ps	500ps	200ps

ALU	Branches	Loads	Stores
50%	15%	25%	10%

الف) با معماری *single-cycle* بگوئید اجرای دستورات *load* چند کلاک طول می‌کشد؟ چند ثانیه؟

ب) برای معماری *multi-cycle* بگوئید اجرای دستورات *load* چند کلاک طول می‌کشد؟ چند ثانیه؟

9- یک پردازنده *Pipelined* را در نظر بگیرید که قطعه کد *MIPS* روبرو را در آن اجرا می‌کنیم. فرض کنید دستور *la* یک دستور واقعی است و از دو دستور فرعی تشکیل نشده است. مقدار ثبات مقصد در این دستور در انتهای مرحله *EX* در دسترس خواهد بود.

الف: اگر تنها یک حافظه برای داده و دستورات داشته باشیم و *forwarding*ای در کار نباشد *hazard* ها را کامل بنویسید و نوع آنها که ساختاری، داده‌ای و یا کنترلی هستند را مشخص کنید.

ب: حالا با فرض داشتن دو حافظه جدا برای دستورات و داده‌ها و وجود *forwarding* با کشیدن نمودار نشان دهید که دستورات چگونه اجرا می‌شوند. فرض کنید *branch* در برنامه *not taken* خواهد بود. *Forwarding* های رخداده را نشان دهید.

سوال عملی:

در این تمرین هدف طراحی و پیاده‌سازی یک پردازنده کامل است. به نکات زیر توجه نمایید:

- در این تمرین باید از ماژول‌هایی که در تمرین‌های قبلی ساخته‌اید استفاده نمایید. در صورتیکه تمرین‌های قبلی را تحویل نداده‌اید و یا آن‌ها را ناقص تحویل داده‌اید، می‌توانید آن‌ها را تکمیل کرده و استفاده کنید، یا از افرادی که تمرین را انجام داده‌اند فایل را بگیرید. دقت کنید فقط ماژول‌هایی که در تمرینات قبلی طراحی شده است را می‌توانید از بقیه دریافت کنید.
- برای استفاده از ماژول‌های قبلی در این تمرین، نیاز به تغییرات جزئی در آن‌ها خواهد بود.
- برای این تمرین باید یک گزارش کامل شامل تمام جزئیات مورد نظر برای ساخت پردازنده، تایپ کرده و ضمیمه‌ی تمرین نمایید. گزارش بخشی از نمره‌ی تمرین شما را در بر می‌گیرد.
- داخل گزارش خود، باید برای تمامی دستورات استفاده شده، RTL آن‌ها را دقیقا مشخص نمایید.
- استفاده از هر تعداد ثبات با هر اندازه‌ای در این تمرین مجاز است. دقت کنید مانند تمارین قبل تمامی ثبات‌ها باید توسط خودتان طراحی شود. همچنان استفاده از ماژول‌های آماده‌ی lmp، یا استفاده از ثبات‌های ساخته شده توسط دیگر دانشجویان مجاز نیست.
- تمامی ثبات‌های پردازنده، به همراه بانک ثبات اصلی باید دارای یک درگاه reset سنکرون باشند که به سیگنال بازنشانی پردازنده وصل شده است و با یک شدن آن، در لبه‌ی کلاک تمامی ثبات‌ها مقدار صفر به خود می‌گیرند.
- در این تمرین، برای حافظه، یک ماژول حافظه به همراه یک فایل برای مقداردهی اولیه‌ی آن در اختیار شما قرار گرفته است. این دو فایل را به پروژه‌ی خود اضافه نموده و از این ماژول استفاده نمایید.
- مشخصات ماژول حافظه:
  - حافظه‌های داده و دستور یکی هستند و درکل پردازنده یک حافظه وجود دارد.
  - این ماژول دارای سه درگاه آدرس تست است که با مقداردهی آن‌ها، محتویات سه آدرس مشخص شده بدون تاخیر در سه درگاه خروجی مربوط به آن‌ها ظاهر خواهد شد. دقت کنید درگاه‌های تست، صرفا برای تست پردازنده هستند و استفاده از آن‌ها در پردازنده نهایی مجاز نیست.
  - این حافظه دارای یک درگاه آدرس اصلی به نام address به طول 8 بیت و یک درگاه اصلی ورودی داده به نام data-in و به طول 16 بیت است. همچنین یک درگاه بیتی با نام rwn نیز مشخص کننده‌ی عمل خواندن (rwn=1) و یا (rwn=0) خواهد بود.

- به طور کلی تاخیر این ماژول مشخص نیست و پس از پایان انجام هر عملیات، سیگنال ready از این حافظه 1 خواهد شد که بعد از ان می‌توان درخواست بعدی را به این حافظه فرستاد.
- این مازول دارای 256 ردیف 16 بیتی است. بدین ترتیب آدرس‌دهی به حافظه با 8 بیت امکان‌پذیر است و پهنای هر خط از حافظه 16 بیت است. دقت نمایید با توجه به اینکه طول دستورات 32 بیت است، عملیات خواندن یک دستور در دو مرحله‌ی پشت سر هم انجام می‌شود. هر دستور 32 بیتی در حافظه در دو خط جا می‌گیرد؛ خط اول 16 پر ارزش شامل opcode([31:16]) و خط دوم شامل 16 بیت دوم یعنی عملوندها ([15:0]) خواهد بود.
- برای مقداردهی اولیه به این حافظه، فایل memory.v را مشاهده نموده و داده‌ی مربوط آدرس مورد نظر را تغییر دهید.
- درگاه reset حافظه به صورت آسنکرون و حساس به لبه‌ی مثبت است.

- برای این تمرین، علاوه بر واحدهایی که در تمرین‌های قبل طراحی شده، نیاز به طراحی یک واحد کنترل نیز خواهد بود که باید به تشخیص خود هر تعداد سیگنال مورد نیاز را به آن افزوده و واحد محاسبه‌ی آن‌ها را اضافه نمایید.
- مانند تمرین‌های قبل، طراحی ماژولار و قابل فهم بخش قابل توجهی از نمره‌ی شما را در بر می‌گیرد. همچنین یک تست‌بنچ که به خوبی صحت پیاده‌سازی را نشان دهد می‌بایست در نظر بگیرید.
- سیگنال‌های پرچم که در تمرین ساخت ALU طراحی شدند، همچنان در این پردازنده، همچنان در این پردازنده به همان حالت حضور داشته و استفاده می‌شوند.
- فایل ثبات استفاده شده در این پردازنده، مشابه تمرین‌های قبلی شما 32 ثبات 32 بیتی است.
- **نکته مهم:** برای پیاده‌سازی این پردازنده، نیاز به یک شمارنده داخلی خواهید داشتکه با هر لبه‌ی مثبت کلاک یکی اضافه می‌شود، در انتهای انجام هر دستور بازنشانی می‌شود، و در حین اجرای هر دستور مشخص می‌کند اجرای دستور فعلی در چه مرحله‌ای قرار دارد. بدین ترتیب این پردازنده خصلتی ما بین Single Cycle و Multi Cycle خواهد داشت. از این شمارنده برای معین کردن RTL مربوط به هر دستور استفاده نمایید و سپس با استفاده از لیست RTL به طراحی پردازنده بپردازید.

دستورات مورد استفاده در این پردازنده در ادامه به طور کامل توضیح داده شده است.

دستورات نوع 1:

Opcode					Reg1					Reg2					Reg3					Reg4					Shift Amount					Unused	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Opcode	Operation	Description
00001	Add	R1 = R2 + R3
00010	Subtract	R1 = R2 – R3
00011	AND (Bitwise)	R1 = R2 & R3
00100	OR (Bitwise)	R1 = R2 | R3
00101	Shift Left Logical	R1 = R2 << Shift Amount
00110	Shift Right Logical	R1 = R2 >> Shift Amount
00111	Max	R1 = Max (R2 , R3)
01000	Set on Less Than	if (R2 < R3) then R1 = 1 , else R1 = 0
01001	Multiply	{R1 , R2} = R3 × R4
01010	Move	R1 = R2
01011	Add Indirect	R1 = Mem[R2] + Mem[R3]

- در دستوراتی که از ثبات برای آدرس‌دهی به حافظه استفاده شده است، مانند Mem[R1]، 8 بیت کم‌ارزش این ثبات (R1[7:0]) لحاظ می‌شود.

دستورات نوع 2:

Opcode					Reg1					Reg2					Immediate														Unused
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Opcode	Operation	Description
01100	Load Upper Immediate	R1[31:16] = Imm , R1[15:0] = 0x0000
01101	Add Immediate	R1 = R2 + SE(Imm)
01110	OR Immediate	R1 = R2 | Imm
01111	Set on Less Than Immediate	if (R2 < Imm) then R1 = 1 , else R1 = 0

- منظور از عبارت SE که در دستور Add Immediate استفاده شده است Sign Extend است. دقت شود اعداد به شیوه‌ی two's complement نمایش داده می‌شوند، بنابراین عملیات Sign Extend با توجه به بیت پر ارزش عملوند انجام خواهد گرفت.
- در دستور Or Immediate، 16 بیت پر ارزش ثبات مقصد برابر با 16 بیت پر ارزش ثبات مبدا می‌شود و 16 بیت کم‌ارزش، حاصل OR 16 بیت کم‌ارزش ثبات مبدا و عدد Immediate خواهد شد.

دستورات نوع 3:

Opcode					Reg1					Reg2					Address									Unused								
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Opcode	Operation	Description
10000	Branch if Equal	if (R1 == R2) then PC ← Address
10001	Branch if Not Equal	if (R1 != R2) then PC ← Address
10010	Jump Register	PC ← R1[7:0]
10011	Jump	PC ← Address
10100	Load Word	R1 = Mem[Address]
10101	Add Memory	R1 = R2 + Mem[Address]