حل تمرین چهار **مهلت تحویل ساعت ۲۴ روز جمعه اول اردیبهشت**

به موارد زیر توجه کنید:

۱- حتما نام و شماره دانشجویی خود را روی پاسخ‌نامه بنویسید.

۲- کل پاسخ تمرینات را در قالب یک فایل pdf با شماره دانشجویی خود نام‌گذاری کرده در سامانه CW بارگذاری کنید.

۳- این تمرین ۶۰ نمره دارد که معادل ۰٫۶ نمره از نمره کلی درس است.

۴- در صورت مشاهده هر گونه مشابهت نامتعارف هر دو (یا چند) نفر کل نمره این تمرین را از دست خواهند داد.

دیاگرام مسیر داده و کنترل پردازنده MIPS را در شکل ۱ مشاهده می‌کنید. عملیات کنترل در این شکل توسط تعدادی سیگنال کنترلی انجام می‌شود که در جدول‌های ۱ و ۲ آمده است. درباره این شکل و جدول‌های مرتبط با آن، به سوالات زیر پاسخ دهید.

۱- (۱ نمره) توجه دارید که جدول ۲ سیگنال سیگنال(های) کنترلی موردنیاز برای پیاده‌سازی دستور j را ندارد. این جدول را طوری کامل کنید که این سیگنال(های) کنترلی را نیز تولید کند.

پاسخ: سیگنالی که در سیگنال کنترلی که در جدول ۲ نیامده است، سیگنال jump است. بنابراین سطر و ستون زیر باید به آن جدول اضافه شود.

Input or Output	Signal Name	R-format	lw	sw	beq	j
Inputs	Op5	0	1	1	0	0
	Op4	0	0	0	0	0
	Op3	0	0	1	0	0
	Op2	0	0	0	1	0
	Op1	0	1	1	0	1
	Op0	0	1	1	0	0
Outputs	RegDst	1	0	X	X	X
	ALUSrc	0	1	1	0	X
	MemtoReg	0	1	X	X	X
	RegWrite	1	1	0	0	0
	MemRead	0	1	0	0	0
	MemWrite	0	0	1	0	0
	Branch	0	0	0	1	X
	ALUOp1	1	0	0	0	X
	ALUOp2	0	0	0	1	X
	Jump	0	0	0	0	1

۲- (۵ نمره) زمان تاخیر هر یک از بلوک‌های شکل ۱ در جدول زیر آمده است.

I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-extend	Shift-left-2
400ps	100ps	30ps	120ps	200ps	350ps	20ps	2ps

الف- اگر فرض کنیم تنها کاری که انتظار داریم این پردازنده انجام دهد واکشی دستورات متوالی باشد، حداقل هر چرخه ساعت چقدر باید باشد؟

ب- فرض کنید پردازنده فقط دستور پرش غیرشرطی نسبت به مقدار فعلی PC (PC Relative) را اجرا می‌کند. حداقل هر چرخه ساعت چقدر باید باشد؟

ج- این بار فرض کنید پردازنده فقط دستور پرش شرطی نسبت به مقدار فعلی PC را اجرا می‌کند، برای مثال دستور beq. در این صورت حداقل هر چرخه ساعت چقدر باید باشد؟

پاسخ:

الف- در این صورت به بلوک‌های I-Mem و Add نیاز داریم که همزمان کار می‌کنند، بنابراین هر چرخه ساعت باید به اندازه ماگزیمم این دو زمان باشد که یعنی ۴۰۰ پیکوثانیه.

ب-= برای اجرای دستورات پرش غیرشرطی به بلوک I-Mem برای واکشی و بلوک‌های Sign-extend، Shift-left-2 برای محاسبهٔ مقدار اضافه‌شده به PC و سپس به بلوک‌های Add و Mux برای به‌روزرسانی PC نیاز داریم. البته به یک بلوک Add هم برای افزایش ۴ واحد به مقدار فعلی PC نیاز داریم که همزمان با سه بلوک اول عمل می‌کند. در مجموع، حداقل چرخهٔ ساعت این طور محاسبه می‌شود:

$$400 + (20 + 2) + (100 + 30) = 552ps$$

ج- در اجرای دستور پرش شرطی، پس از واکشی (بلوک I-Mem)، دو مسیر موازی داریم که یکی شامل بلوک‌های Sign-extend، Shift-left-2 و Add است و دیگری مسیری برای برآورد شرط شامل Regs، Mux و ALU است و پس از آن به یک بلوک Mux نیاز داریم. بنابراین چرخهٔ ساعت این طور محاسبه می‌شود:

$$400 + max(20 + 2 + 100\ ,200 + 30 + 120) + 30 = 400 + 350 + 30 = 780ps$$

۳- (۱۰ نمره) فرض کنید دو دستور زیر در حافظه دستورالعمل پردازندهٔ شکل ۱ قرار دارد و PC به دستور اول اشاره می‌کند.

Address	Instruction
0x00400008	0x8e090000
0x0040000C	0x1120fffc

الف- هر دو دستور را رمزگشایی (decode) کنید.

ب- مسیر دادهٔ اجرای هر کدام از دو دستور را مشخص کنید.

ج- مقادیر سیگنال‌های کنترلی در حین اجرای هر کدام از دو دستور را تعیین کنید.

د- پس از اجرای متوالی این دو دستور مقدار کدامیک از ثبات‌ها تغییر می‌کند؟ چطور؟

پاسخ:

الف و ج:

Instruction		RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1-2
0x8e090000	lw $t1,0($s0)	0	1	1	1	1	0	0	00
0x1120fffc	beq $t1,$0,PC+4-16	x	0	x	0	0	0	1	01

ب- دستور اول: از حافظه واکشی می‌شود. آدرس Rs به فایل ثبات‌ها داده می‌شود و مقدار آن وارد ALU می‌شود. ۱۶ بیت کم‌ارزش دستور وارد بلوک sign-extend می‌شود و با مقدار Rs جمع می‌شود. خروجی ALU به عنوان آدرس به حافظهٔ داده وارد می‌شود و محتوای خوانده شده بر روی ثبات Rt نوشته می‌شود. همزمان مقدار PC هم با عدد ۴ جمع می‌شود.

دستور دوم: از حافظه واکشی می‌شود. آدرس‌های Rs و Rt به فایل ثبات‌ها داده می‌شود و مقادیر آنها وارد ALU شده و از هم کم می‌شوند و مقدار خروجی Zero تعیین می‌شود. همزمان ۱۶ بیت کم‌ارزش دستور وارد بلوک sign-extend می‌شود و دو بیت به چپ شیفت داده شده و با مقدار PC+4 جمع می‌شود و درنهایت محتوای ثبات PC بسته به مقدار Zero به‌روز می‌شود.

د- پس از اجرای متوالی این دو دستور فقط مقدار ثبات $t1 تغییر می‌کند.

۴- (۴ نمره) در فرایند ساخت تراشه‌های سیلیکونی، هر عیب و نقص در موارد اولیه و یا هر گونه خطا در یکی از مراحل تولید می‌تواند منجر به تحویل مدارهای معیوب شود. یکی از ایرادهای بسیار رایج در تراشه‌ها آن است که سیگنال منتشرشده روی یکی از سیم‌ها بر روی دیگری تاثیر می‌گذارد. این ایراد را خطای هم‌شنوایی (cross-talk fault) می‌نامند. ردهٔ خاصی از خطاهای هم‌شنوایی زمانی رخ می‌دهد که سیگنالی به سیمی متصل شده باشد که مقدار منطقی ثابتی دارد (برای مثال به سیم منبع تغذیه وصل شده باشد). در چنین حالتی خطایی موسوم به خطای سیگنال چسبیده به صفر (stuck-at-0) یا سیگنال چسبیده به یک (stuck-at-1) رخ می‌دهد و مقدار منطقی سینگال آلوده به این خطا همواره صفر یا یک خواهد بود. در سوالات زیر می‌خواهیم فرایندی برای آزمایش تراشه پس از خروج از خط تولید به منظور اطمینان از سالم بودن آن طراحی کنیم. فرض کنید در این فرایند محتوای PC، تمام ثبات‌ها، حافظهٔ داده و حافظهٔ دستور با مقادیری پر می‌شوند. (این مقادیر را می‌توانید به دلخواه انتخاب کنید) و سپس یک دستور واحد به اجرا درمی‌آید و در ادامه مقدار PC، حافظه و ثبات‌ها خوانده می‌شود.

الف- آزمون مناسبی طراحی کنید (یعنی مقادیر مناسب برای PC، حافظه‌ها و ثبات‌ها پیشنهاد دهید) که بتواند خطای چسبیدن به صفر بیت کم‌ارزش خطوط ورودی Write register را تشخیص دهد.

ب- بند الف را برای تشخیص خطای چسبیدن به یک همان بیت تکرار کنید. آیا امکانش هست که با یک آزمون واحد هر دو خطای چسبیده به صفر و چسبیده به یک را تشخیص داد؟ اگر پاسخ مثبت است چگونه و اگر منفی است چرا؟

ج- اگر مطمئن باشیم که سیگنال فوق در تراشهٔ تولید شده خطای چسبیده به یک دارد، آیا کماکان این پردازنده قابل استفاده است؟ برای این که چنین پردازنده‌ای قابل استفاده باشد باید قادر باشیم برنامه‌ای را که بر روی پردازنده معمولی MIPS اجرا می‌شود، به برنامه‌ای تبدیل کنیم که بر روی پردازنده معیوب هم اجرا شود. شما می‌توانید فرض کنید که به اندازهٔ کافی حافظهٔ داده و دستور در دسترس هست که بتوانید برنامهٔ موردنظر را طولانی‌تر کرده و یا داده‌های اضافه‌تری را ذخیره نمایید. (راهنمایی: یک پردازندهٔ معیوب زمانی قابل استفاده است که بتوان دستوری را که در اثر خطای حادث شده دیگر کار نمی‌کند با دنباله‌ای از دستوراتی که قطعا کار می‌کنند جایگزین کرد، طوری که برنامه همان نتایج مورد انتظار را بدهد.)

پاسخ:

الف- برای تست چسبیدن به صفر باید دستوری را اجرا کنیم که نیاز باشد آن بیت به خصوص مقدار یک داشته باشد. بنابراین در این آزمون باید عدد خاصی روی یکی از ثبات‌های شماره فرد بنویسیم و بررسی کنیم ببینیم آیا این عدد درست نوشته شده یا خیر. برای مثال می‌توانیم ابتدا مقدار دو ثبات R2 و R3 را برابر صفر و یک قرار

بدهیم. سپس دستور add R3,R2,R2 را اجرا کنیم. اگر بیت کم‌ارزش خطوط ورودی Write Register همیشه صفر باشد، مقدار R3 یک خواهد ماند، در حالی که اگر این خطا وجود نمی‌داشت، مقدار R3 باید صفر میشد.
ب- برای تست چسبیدن به یک همان بیت، باید روشی مشابه با بند الف به کار بریم، منتها جای ثبات‌های زوج و فرد را عوض کنیم. به این ترتیب که مقدار دو ثبات R2 و R3 را برابر یک و صفر قرار می‌دهیم و دستور add R2,R3,R3 را اجرا کرده و مقدار ثبات R2 را بررسی می‌کنیم. اگر مقدار آن صفر بود یعنی خطای موردنظر وجود دارد. طبعا این دو تست (بندهای الف و ب) را نمی‌توانیم با یک آزمون انجام بدهیم چون نمی‌توانیم همزمان ۰در یک چرخه ساعت) دو مقدار متفاوت روی یک سیم خاص قرار بدهیم.
ج- اگر بدانیم حتما خطای چسبیده به یک در خط کم‌ارزش Write Register وجود دارد، باید برنامه‌ها را طوری بازنویسی کنیم که فقط از ثبات‌هایی با شماره فرد استفاده کند.

در طراحی پردازندهٔ شکل ۱ فرض این بوده است که قرار است تنها دستورالعمل‌های add، sub، and، or، slt، lw، sw، beq و j اجرا شود. در ادامه، می‌خواهیم چند دستور دیگر از جمله دستورات شیفت را هم به مجموعه دستورات قابل اجرا اضافه کنیم. فرض کنید ALU یک ورودی ۵ بیتی دیگر دارد که تعداد بیت لازم برای شیفت را از این طریق دریافت می‌کند.

۵- (۱۵ نمره) برای اضافه کردن دستورات sll و srl (شیفت منطقی به چپ و راست) چه تغییراتی باید در شکل و جداول بدهیم؟ فعلا فرض کنید ALU قابلیت شیفت منطقی به چپ و راست را دارد و برای فعال کردن این قابلیت باید ALU control Input به ترتیب XXXX و YYYY باشد. توجه کنید این دو دستور از نوع r-format هستند و به شکل زیر استفاده می‌شوند:

```
sll/srl rd,rt,shamt
```

پاسخ: برای اجرای این دو دستور باید مقدار ثبات Rt و مقدار shamt به ALU داده شود. چون فرض کردیم ALU یک ورودی ۵بیتی جداگانه برای تعیین مقدار شیفت دارد، shamt را مستقیما از بیت‌های ۶ تا ۱۰ دستور گرفته و به ALU وصل می‌کنیم. محتوای Rt هم از طریق Read data2 وارد ALU می‌شود و نتیجهٔ شیفت هم باید در Rd نوشته شود، بنابراین باید Branch=Jump=0، ALUSrc=0، RegDst=1، RegWrite=1، MemtoReg=0، MemRead=MemWrite=0 باشد. از طرفی این دو دستور از نوع R-Type هستند بنابراین ALUOp=10 است و ALU Control باید طوری تغییر کند که اگر بیت‌های func (بیت‌های ۰ تا ۵ دستور) شیفت چپ یا راست بودند، خروجی‌های XXXX و YYYY را تولید کند. بنابراین جدول ۲ تغییری نمی‌کند، ولی دو ردیف به پایین جدول ۱ اضافه می‌شود.

۶- (۱۵ نمره) برای اضافه کردن دستورات sllv و srlv چه تغییراتی باید در شکل و جداول بدهیم؟ این دو دستور به شکل زیر استفاده می‌شوند و محتوای rt را به اندازهٔ rs بیت به صورت منطقی به چپ یا راست شیفت می‌دهند.

```
sllv/srlv rd,rt,rs
```

پاسخ: تفاوت این دو دستور با دو دستور سوال ۵ این است که مقدار شیفت در ثبات Rs قرار دارد. برای اجرای این دستور باید در ورودی مقدار شیفت ALU یک مالتی‌پلکسر قرار بدهیم که از دو جا مقدار می‌گیرد، از بیت‌های

shamt یا از محتوای Rs که از Read data1 قابل دسترس است. و بیت کنترلی که برای تمایز میان این دو ورودی استفاده می‌شود، ShamtSrc1 می‌نامیم، که اگر صفر باشد مقدار شیفت از بیت‌های shamt و اگر یک باشد مقدار شیفت از Read data1 می‌آید. این بیت کنترلی را ALU Control باید تولید کند، چون مقدار آن باید از روی بیت‌های func تعیین شود که به واحد Control وارد نمی‌شود. بنابراین باز هم نیازی به تغییر جدول ۲ نداریم، اما باید دو سطر و یک ستون به جدول ۱ اضافه کنیم.

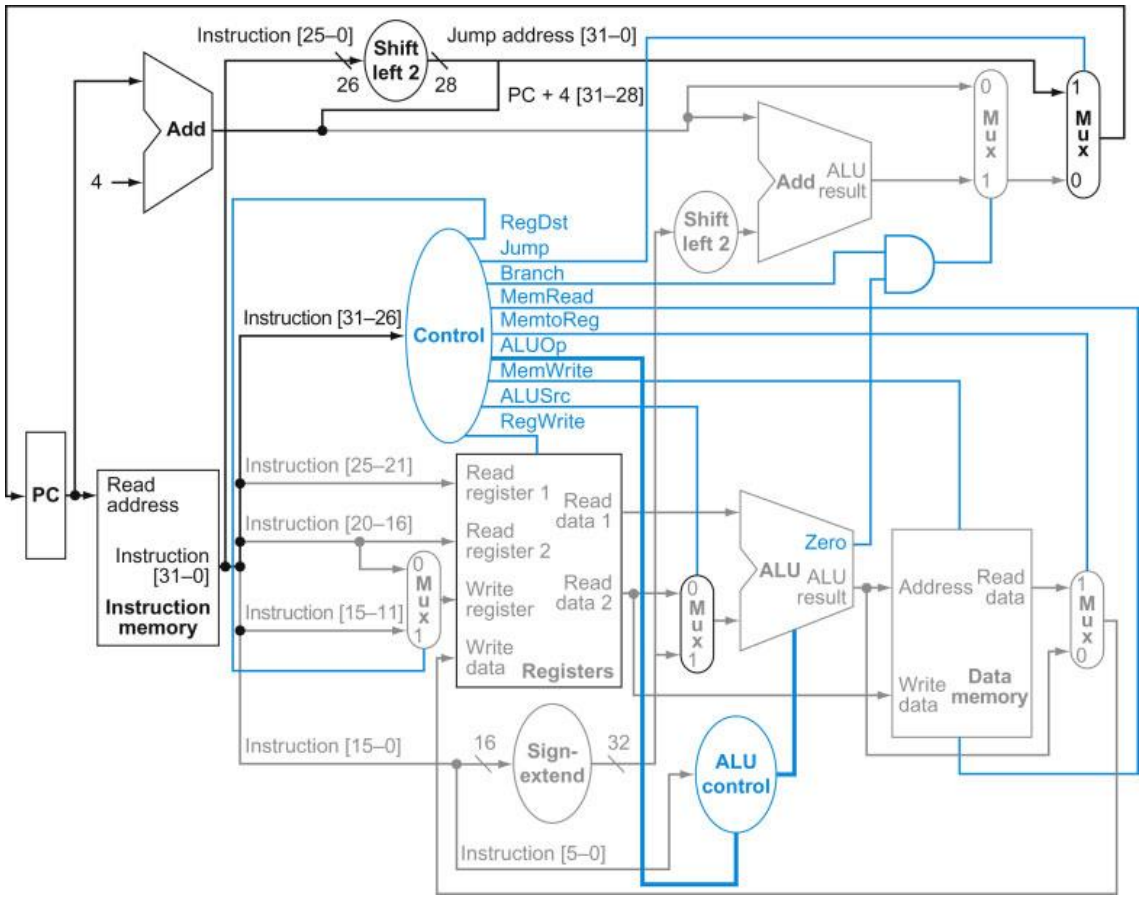
۷- (۱۰ نمره) برای اضافه کردن دستور lui چه تغییراتی باید در شکل و جداول بدهیم؟ این دستور به شکل زیر استفاده می‌شود و عدد ۱۶ بیتی imm را در ۱۶ بیت پرارزش rt قرار می‌دهد و ۱۶ بیت کم‌ارزش rt را صفر می‌کند.

```
lui rt,imm
```

پاسخ: در این دستور باید عدد ۱۶ بیتی imm را در ۱۶ بیت پرارزش ثبات Rt قرار بدهیم که معادل این است که این عدد را ۱۶ بیت به چپ شیفت بدهیم و در Rt بنویسیم. پس می‌توانیم مقدار این عدد را از طریق اپرند دوم به ALU بدهیم و عدد ۱۶ را به ورودی شیفت ALU بدهیم و از طریق ALU Control فرمان شیفت به چپ به ALU بدهیم و از طرفی RegDst را هم برابر یک قرار بدهیم که شماره ثبات مقصد از روی بیت‌های ۱۱ تا ۱۵ تعیین شود. این دستور از نوع دستورات I-type است، پس خود دستور توسط واحد Control شناسایی می‌شود. واحد کنترل می‌تواند با تشخیص این دستور کد ۱۱ برای ALU Control ارسال کند تا ALU Control کد XXXX برای ALU ارسال کند تا شیفت به چپ انجام شود. برای اجرای این دستور باید هر دو جدول ۱ و ۲ تغییر کند. ضمنا باید یک مالتی‌پلکسر دیگر هم به ورودی مقدار شیفت ALU اضافه شود که در زمان اجرای این دستور عدد ۱۶ را تولید کند. بیتی که این مالتی‌پلکسر را کنترل می‌کند در واحد Control تولید می‌شود و نام آن را ShamtSrc2 می‌نامیم که اگر صفر باشد، مقدار شیفت از مالتی‌پلکسری می‌آید که با ShamtSrc1 کنترل می‌شود و اگر یک باشد مقدار شیفت عدد ثابت ۱۶ است.
مقادیر نهایی دو جدول ۱ و ۲ در شکل زیر رسم شده است.

opcode	ALUOp	Operation	funct	ALU function	ALU control	ShamtSrc1
lw	00	load word	XXXXXX	add	0010	X
sw	00	store word	XXXXXX	add	0010	X
beq	01	branch equal	XXXXXX	subtract	0110	X
R-type	10	add	100000	add	0010	X
		subtract	100010	subtract	0110	X
		AND	100100	AND	0000	X
		OR	100101	OR	0001	X
		set-on-less-than	101010	set-on-less-than	0111	X
		sll	000000	shift left	XXXX	0
		srl	000010	shift right	YYYY	0
		sllv	000100	shift left	XXXX	1
		srlv	000110	shift right	YYYY	1
I-type	11	lui	XXXXXX	shift left	XXXX	X

Input or Output	Signal Name	R-format	lw	sw	beq	j	lui
Inputs	Op5	0	1	1	0	0	0
	Op4	0	0	0	0	0	0
	Op3	0	0	1	0	0	1
	Op2	0	0	0	1	0	1
	Op1	0	1	1	0	1	1
	Op0	0	1	1	0	0	1
Outputs	RegDst	1	0	X	X	X	1
	ALUSrc	0	1	1	0	X	0
	MemtoReg	0	1	X	X	X	0
	RegWrite	1	1	0	0	0	1
	MemRead	0	1	0	0	0	0
	MemWrite	0	0	1	0	0	0
	Branch	0	0	0	1	X	0
	ALUOp1	1	0	0	0	X	1
	ALUOp2	0	0	0	1	X	1
	Jump	0	0	0	0	1	0
	ShiftSrc2	0	X	X	X	X	1



شکل ۱- بلوک دیاگرام مسیر داده و کنترل پردازنده ساده MIPS

جدول ۱- شرح ارتباط سیگنال‌های واحد ALU Control در شکل ۱

Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	AND	0000
R-type	10	OR	100101	OR	0001
R-type	10	set on less than	101010	set on less than	0111

جدول ۲- شرح ارتباط سیگنال‌های واحد Control در شکل ۱

Input or output	Signal name	R-format	lw	sw	beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1