پاسخ‌نامه تمرین چهار **مهلت تحویل ساعت ۲۴ روز چهارشنبه یک آذر**

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.

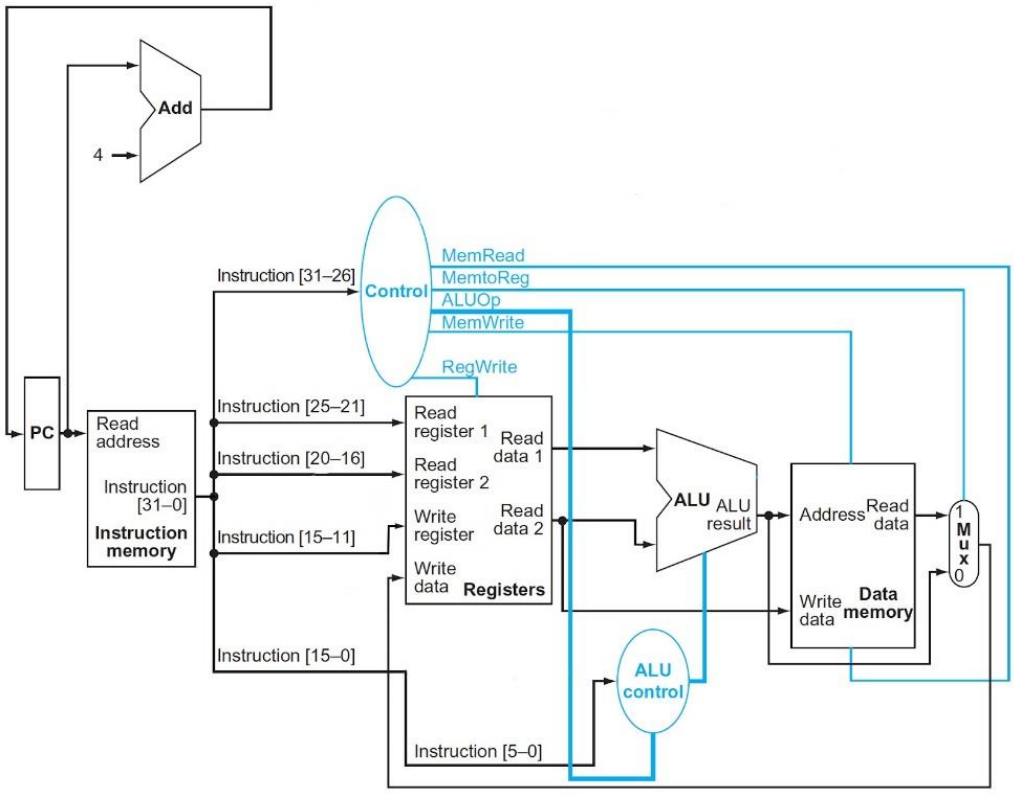
۱- (۱۰ نمره) فرض کنید یک پردازنده MIPS با مسیر دادهٔ زیر داشته باشیم.

الف- آیا دستور srl روی پردازنده زیر قابل اجرا است؟ چرا؟

ب- آیا دستور beq روی این پردازنده قابل اجرا است؟ چرا؟

ج- کدام دسته از دستورات MIPS می‌توانند روی این سخت‌افزار اجرا شوند؟ (R-type, I-type, J-type) با ذکر دلیل بیان کنید.

د- آیا می توانید یک کد میپس قابل اجرا روی این پردازنده بنویسید که حاصل جمع دو ثبات t1 و t2 را در حافظه به آدرس x05 ذخیره کند؟ فرض کنید مقادیر اولیه ثبات‌های t1 و t2 به ترتیب ۱ و ۲ باشد. اگر این کار ممکن نیست بگویید چرا نمی‌توانیم چنین دستوری اجرا کنیم.



پاسخ:

لف- خیر چون به مقدار shamt که در بیت های ۶ تا ۱۰ است دسترسی نداریم.

ب- خیر چون دستورات پرشی مقدار PC را تغییر می‌دهند در حالی که در این مسیر داده فقط می‌توانیم مقدار PC را ۴ تا افزایش دهیم و تغییر دیگری نمی‌توانیم در مقدار آن اعمال کنیم.

ج- بیشتر R-type ها به جز مواردی از قبیل دستورات شیفت (مانند مورد الف) قابلیت اجرایی روی این پردازنده را دارند. دستورات J-type نمی‌توانند اجرا شوند چون نمی‌توانیم PC را تغییر دهیم. دستورات I-type هم به دلیل تغییر PC (دستورات شرطی) و داشتن بخش immediate روی این پردازنده قابل اجرا نیستند.

د- بله، این کار با ترفندی شدنی است. در یک پردازنده معمولی در ALU مقدار immediate با ثبات s جمع شده و آدرس حافظه را می‌سازد. اما در پردازندهٔ ما امکان ورود مقدار immediate به ALU وجود ندارد و آدرس حافظهٔ مقصد از جمع دو ثبات s و t به دست می‌آید. با استفاده از همین نکته می‌توان مقدار ثبات t را در حافظه به آدرس s+t ذخیره کرد.

کد زیر حاصل‌جمع دو ثبات t1 و t2 را در خانه‌ای از حافظه به آدرس t2 + t0 (که برابر ۵ است) ذخیره می‌کند.

```
add $t0,$t1,$t2
sw $t0,0($t2)
```

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.

۲- (۲۰ نمره)

الف- برای اضافه کردن دستور jal چه تغییراتی باید به شکل و جداول بدهیم؟

دستور jal از نوع J-format است، opcode آن ۳ است و کاری که انجام می‌دهد این است که ابتدا آدرس بازگشت را در $ra (ثبات شماره ۳۱) می‌ریزد و سپس مثل دستور j رفتار می‌کند و به شکل زیر استفاده می‌شود:

```
jal JumpAddress
```

پاسخ:

برای اجرای دستور jal باید مقدار فعلی PC را روی $ra بنویسیم و سپس مقدار PC را برابر با آدرس جدید قرار دهیم. بنابراین باید یک MUX در ورودی write-data فایل ثبات قرار دهیم که ورودی اول آن خروجی MUX بعد از حافظهٔ داده است و ورودی دوم آن مقدار PC+4 است. سیگنال کنترلی این MUX را RegWriteSrc قرار می‌دهیم. همچنین MUX قبل از write-register را باید از حالت دوتایی به حالت ۳ یا ۴ تایی افزایش داد و ورودی سوم و چهارم آن را عدد ۳۱ (معادل شمارهٔ ثبات $ra) قرار داد. سیگنال کنترلی دومی که باید به این MUX اضافه شود را RegDst2 می‌نامیم. مدار داخلی Control هم باید طوری تغییر کند که خروجی Jump در زمان اجرای دستور jal هم یک شود.

با توجه به مطالب گفته شده سه سیگنال کنترلی به جدول دوم اضافه می‌شود. (چون Jump هم در این جدول نیست، آن را هم باید اضافه کنیم.)

ب- برای اضافه کردن دستور jr چه تغییراتی باید به شکل و جداول بدهیم؟
دستور jr از نوع R-format است، مقدار funct آن ۸ است و کاری که انجام می‌دهد این است که PC را برابر مقدار موجود در ثبات rs قرار می‌دهد و به این شکل استفاده می‌شود:

```
jr rs
```

پاسخ:
برای اینکه دستور jr را بتوانیم اجرا کنیم، باید این امکان را فراهم کنیم که محتوای یک ثبات خوانده شود و مستقیما وارد PC شود. می‌توانیم یک MUX بعد از MUXای که برای دستور j به شکل اضافه شده بود، قرار بدهیم و یک ورودی را به خروجی MUX مربوط به j اختصاص دهیم و ورودی دیگر را از بانک ثبات بگیریم (Read Data1) و ورودی select را هم در مدار ALU Control تولید کنیم. این ورودی را نمی‌توانیم در Control تولید کنیم، چون در دستور jr مثل بقیه دستورات R-format همه بیت‌های Op صفر هستند و دستور توسط بیت‌های function مشخص می‌شود که اصلا وارد واحد کنترل نمی‌شود. بنابراین جدول ۱ تغییر خواهد کرد، ضمن اینکه این بار باید سیگنال کنترلی مورد نظر از ترکیب خروجی‌های ALU Control تولید شود.
البته حالا که تعداد MUXهای پرش زیاد شده، می‌توانیم همه را با هم ترکیب کنیم اما برای اینکه فهم تغییرات ساده‌تر باشد، اضافه کردن یک MUX جدید هم اشکالی ندارد.

opcode	ALUOp	Operation	funct	ALU function	ALU control
lw	00	load word	XXXXXX	add	0010
sw	00	store word	XXXXXX	add	0010
beq	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		set-on-less-than	101010	set-on-less-than	0111
		jr	001000	One of the unused combinations of ALU control	

ج- برای اضافه کردن دستورهای فرضی push و pop چه تغییراتی باید به شکل و جداول بدهیم؟ این دستورها از نوع R-format هستند و مقدار funct آن‌ها به ترتیب برابر XXXXX و YYYYY است. کاری که push انجام می‌دهد این است که $sp را ۴ تا کم می‌کند و سپس مقدار $rs را در خانه $sp از حافظه ذخیره می‌کند، و pop ابتدا محتوای خانهٔ $sp از حافظه را در $rs می‌ریزد و سپس مقدار $sp را ۴ تا می‌افزاید و به این شکل استفاده می‌شوند:

```
push    rs
pop     rs
```

پاسخ:
چون در سوال فرمت خاصی برای این دو دستور داده نشده، می‌توانیم فرض کنیم آنها از نوع i-type هستند و همچنین برای ساده‌تر کردن مسیر داده می‌توانیم فرض کنیم داده‌ای که قرار است push یا pop شود در $rt

قرار دارد (در سوال گفته شده که داده در $rs قرار دارد. در آن صورت هم می‌توانیم دستور را بسازیم اما مسیر داده کمی پیچیده‌تر خواهد شد). بنابراین فرمت این دو دستور را به این صورت تعریف می‌کنیم:

push	sp	rt	-4

pop	sp	rt	0

علاوه بر این فرض می‌کنیم فایل ثبات دو ورودی غیر از ورودی‌های فعلی دارد که اگر اولی فعال شود، ۴ واحد از $sp کم می‌کند و اگر دومی فعال شود، ۴ واحد به $sp اضافه می‌کند.

اجرای دستور push به این شکل انجام می‌شود که محتوای $sp با ۴- جمع شده و آدرس حافظه را می‌سازد و محتوای $rt در حافظه ذخیره می‌شود (مشابه اجرای دستور sw $rt, -4($sp)) و در عین حال خطی که ۴ واحد از $sp کم می‌کند هم فعال می‌شود، بنابراین مقدار $sp هم تغییر خواهد کرد.

اجرای دستور pop به این شکل انجام می‌شود که محتوای $sp با صفر جمع شده و آدرس حافظه را می‌سازد و دادهٔ خوانده‌شده از حافظه روی $rt نوشته می‌شود (مشابه اجرای دستور lw $rt, 0($sp)) و در عین حال خطی که ۴ واحد به $sp اضافه می‌کند هم فعال می‌شود، بنابراین مقدار $sp هم به درستی تغییر خواهد کرد.

د- پس از تغییراتی که در شکل و جداول دادید، اگر از پردازنده صرفا واکشی دستورات متوالی و اجرای دستورات jal و jr انتظار برود حداقل طول هر چرخه ساعت باید چه مقدار باشد؟ زمان تاخیر هر یک از بلوک‌های شکل ۱ در جدول زیر آمده است. اگر بلوکی اضافه کردید که در جدول نبود، فرض کنید گیت‌های and و or هر یک ۱۰ پیکوثانیه، گیت‌های not هر یک ۸ پیکوثانیه و گیت‌های xor هر یک ۱۲ پیکوثانیه تاخیر دارند.

I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-extend	Shift-left-2
350ps	90ps	30ps	120ps	180ps	320ps	10ps	2ps

پاسخ:

برای واکشی به بلوک‌های I-Mem و Add نیاز داریم که همزمان کار می‌کنند، بنابراین به اندازه ماگزیمم این دو زمان می‌برند (۳۵۰ پیکوثانیه). برای رمزگشایی jal دستور کار خاصی نباید انجام بدهیم به جز تولید سیگنال‌های کنترلی در واحد Control، اما رمزگشایی jr نیاز به خواندن از فایل ثبات دارد، یعنی زمان Reg (۱۸۰ پیکوثانیه) هیچکدام از دو دستور به ALU و Mux پیش از آن و به D-Mem نیاز ندارند، اما دستور jal باید مقدار PC+4 را در $ra ذخیره کند، بنابراین به دو Mux و Reg نیاز دارد (۲۴۰ پیکوثانیه) و دستور jr باید $ra را در PC قرار بدهد که به این منظور باید از یک Mux عبور کند (۳۰ پیکوثانیه)، ماکزیمم این دو مقدار همان ۲۱۰ پیکوثانیه است. پس زمان اجرای هر کدام از دو دستور این طور حساب می‌شود:

jal: $350 + 240 = 590\ ps$

jr: $350 + 180 + 30 = 560\ ps$

می‌بینیم که jal بیشتر طول می‌کشد، پس حداقل چرخه ساعت باید ۵۹۰ پیکوثانیه باشد.

ه- فرض کنید پردازنده دستورات beq و push و pop را هم پشتیبانی می‌کند، مجددا حساب کنید حداقل طول هر چرخه ساعت باید چقدر باشد.

پاسخ:

بین این سه دستور pop از بقیه زمان بیشتری نیاز دارد، چون تقریبا از همه بخش‌ها باید استفاده کند:

fetch: max(I-Mem,Add)=350, decode: Reg=180, execute: Mux+ALU=150,
mem: D-Mem=320, write-back: Mux+Reg=210

pop:$350 + 180 + 150 + 320 + 210 = 750\ ps$

اگر برای اجرای pop و push از روش‌های دیگری استفاده شده باشد، این اعداد ممکن است متفاوت باشند.

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

```
0x1120000C
0x01294826
0x112000F8
```

الف- دستورات داده‌شده را رمزگشایی (decode) کنید.

ب- مسیر دادهٔ اجرای هر دستور را مشخص کنید.

ج- مقادیر سیگنال‌های کنترلی را در حین اجرای هر کدام از دستورات تعیین کنید. (اگر چند حالت دارد حالت‌بندی کنید یا در جدول عبارت بولی بنویسید)

د- این قطعه‌کد چه کاری انجام می‌دهد؟ مقادیر ثبات‌هایی که ممکن است تغییر کنند را پس از اجرای آن مشخص کنید. (اگر چند حالت دارد حالت‌بندی کنید)

پاسخ:

الف- دستورات معادل عبارتند از: (به جای نام ثبات‌ها می‌توانید شمارهٔ آنها را هم بنویسید.)

```
beq $t1, $0, 12
xor $t1, $t1, $t1
beq $t1, $0, 248
```

ب- برای اجرای دستورات beq مقدار دو ثبات از فایل ثبات‌ها خوانده شده و در ALU از هم کم می‌شود. همزمان سیگنال Branch فعال می‌شود. اگر مقدار دو ثبات مساوی باشد، خروجی Zero هم فعال می‌شود و در این صورت ۱۶ بیت کم‌ارزش دستور بعد از sign extend شدن به مقدار PC+4 اضافه می‌شود که به معنای انجام پرش است. اگر مقدار دو ثبات مساوی نباشند، اتفاق خاصی نمی‌افتد و دستور بعدی اجرا خواهد شد.

برای اجرای دستور XOR مقدار دو ثبات خوانده شده و در ALU با هم XOR می‌شود و نتیجه در $t1 نوشته می‌شود.

ج- سیگنال های کنترلی مطابق جدول زیر خواهند بود:

	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp
beq	x	0	x	0	0	0	1	01
xor	1	0	0	1	0	0	0	10

د- قطعه کد داده شده ابتدا بررسی می‌کند که ثبات $t1 صفر است یا خیر. اگر صفر بود به ۱۲ دستور بعد پرش می‌کند. اگر $t1 صفر نباشد، $t1 را با خودش xor می‌کند که نتیجه‌اش صفر می‌شود. صفر را در $t1 ذخیره می‌کند. سپس با اجرای دستور سوم به ۲۴۸ دستور بعد پرش می‌کند. تنها ثباتی که احتمال تغییرش وجود دارد $t1 است که این هم تنها در حالتی است که صفر نباشد. اگر $t1 صفر باشد، مقدارش تغییر نخواهد کرد.

۴- (۱۰ نمره) در پردازنده MIPS ما دستوری به نام sw را داریم. این دستور مقدار موجود در یک ثبات را در حافظه می‌نویسد. اکنون می‌خواهیم دستوری طراحی کنیم که یک مقدار immediate شانزده بیتی را ابتدا sign-extend کند و سپس در محلی از حافظه که آدرس آن در Rt است، بنویسد. برای این کار لازم است چه تغییراتی در data path و جدول‌های زیر بدهیم؟ فرض کنید op-code دستور جدید 111111 است.

```
swi Rt,i      # M[Rt]←i
```

پاسخ:

این دستور باید مقدار sign-extend شدهٔ بیت‌های ۰ تا ۱۵ یعنی همان مقدار immediate را در حافظه به آدرس Rt بنویسد. در دستورات i-type ثبات Rt با بیت‌های ۱۶ تا ۲۰ مشخص می‌شود. لذا یک روش این است که یک MUX در ورودی write-data حافظهٔ داده، قرار دهیم، که ورودی اول آن data-read-2 از فایل ثبات و ورودی دوم آن مقدار sign-extend شده immediate باشد. نام سیگنال کنترلی این MUX را هم WriteSrc می‌گذاریم. آدرس از خروجی ALU می‌آید. می‌توانیم سیگنال کنترلی ALUSrc را برابر صفر قرار دهیم که محتوای Rt با محتوای $zero جمع شود (چون بیت‌های ۲۱ تا ۲۵ صفر است). بنابراین سیگنال‌های کنترلی باید به این صورت باشند:

	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp	WriteSrc
swi	x	0	x	0	0	1	0	00	1

۵- (۱۰ نمره) می‌خواهیم دستور زیر را به مجموعهٔ دستوراتِ پردازنده MIPS اضافه کنیم. این دستور از نوع i-format است و حاصل‌جمع دو ثبات s و t را در حافظه به آدرس sign-extend شدهٔ i ذخیره می‌کند. تضمین می‌کنیم i همواره مثبت است. چه تغییری در مسیر داده و جدول‌های زیر لازم است اعمال کنیم تا این دستور قابلیت اجرا را داشته باشد؟ فرض کنید op-code دستور جدید ۱۱۱۱۱۱ است.

```
addm s,t,i    # M[i] = R[s] + R[t]
```

پاسخ:

برای حل این سوال کافی است یک MUX در ورودی آدرس حافظهٔ داده قرار دهیم که ورودی اولش خروجی ALU باشد و ورودی دومش مقدار sign-extend بیت‌های صفر تا ۱۵ است. سیگنال کنترلی این MUX را AddrSrc می‌نامیم. همچنین باید یک MUX دیگر در ورودی Write-data حافظهٔ داده باشد که ورودی اول آن data-read-2 فایل ثبات و ورودی دوم آن خروجی ALU باشد. سیگنال کنترلی این MUX را نیز WriteSrc می نامیم. بنا به موارد گفته شده سیگنال‌های کنترلی باید به صورت زیر باشند:

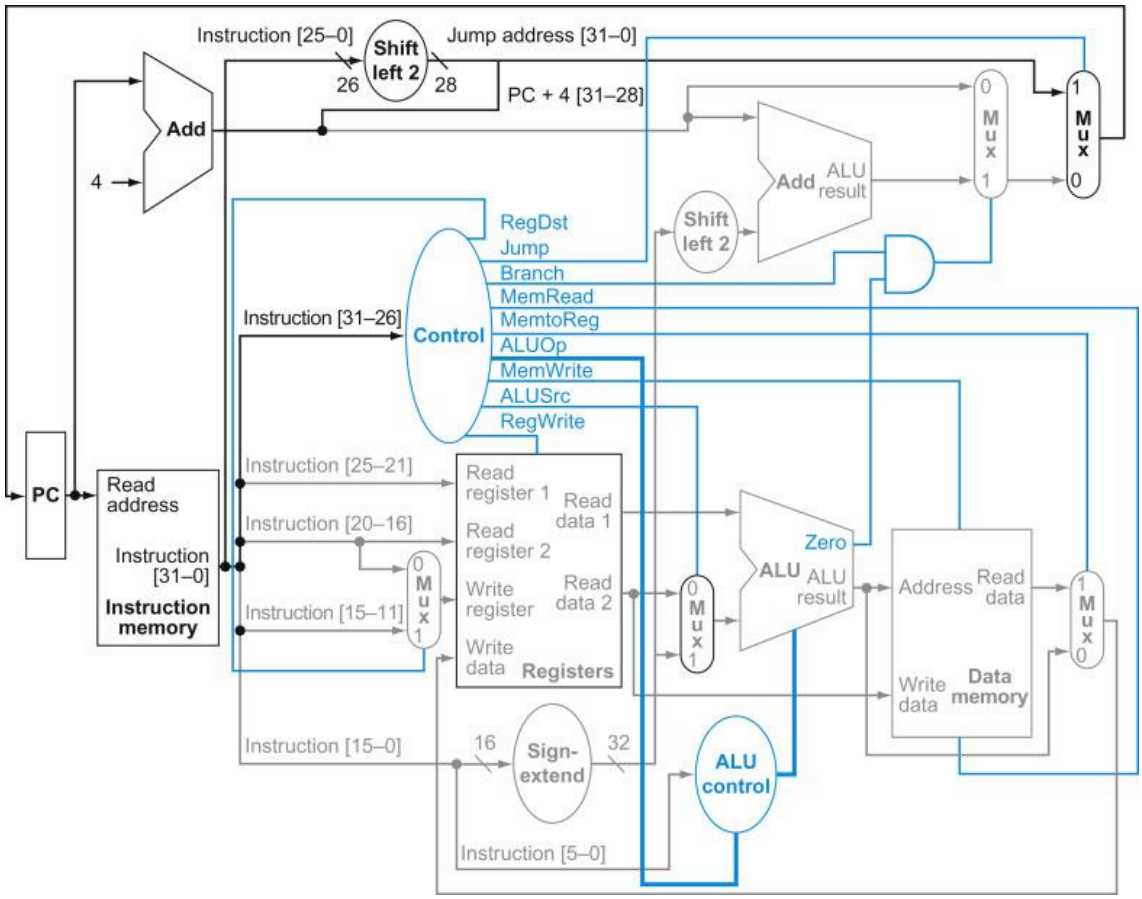
	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp	WriteSrc	AddrSrc
addm	x	0	x	0	0	1	0	00	1	1

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



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