



پاسخ مسئلہ ۱.

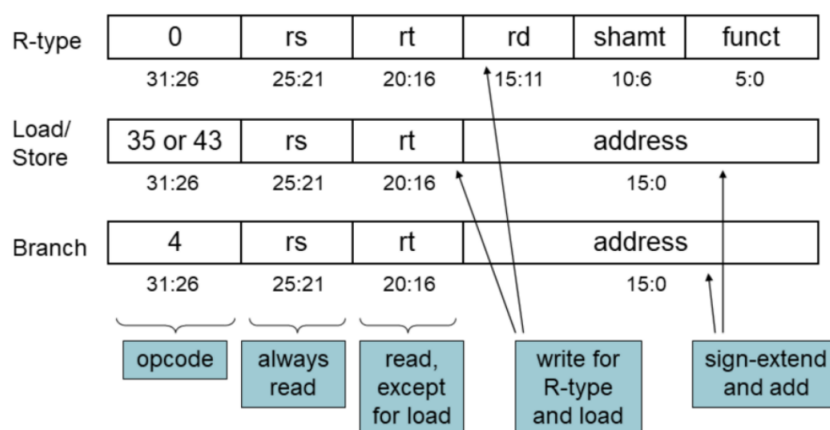
$$a : \cdot xAC\textcircled{2} \cdot \cdot 1\textcircled{4}$$
$$b : \cdot x \cdot \cdot \wedge \Upsilon \cdot \wedge \Upsilon A$$

الف

در این باید کد اسمبلی معادل با هر دستور را بنویسیم. برای این کار ابتدا اعداد را به باینری تبدیل می‌کنیم:

[illegible]
$$b = (\cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \setminus \cdot \cdot \cdot \cdot \cdot \cdot \setminus \cdot \cdot \cdot \cdot \cdot \cdot \setminus \cdot \cdot \cdot \cdot \cdot \cdot \setminus \cdot \setminus \cdot \setminus \cdot \setminus \cdot)_{\gamma}$$

حال از این تصویر استفاده کرده و دستورات را شناسایی می‌کنیم:



$$opcode_a = a[\mathfrak{A}1 : \mathfrak{A}6] = (1 \cdot 1 \cdot 11)_\gamma = (\mathfrak{A}3)_{\mathfrak{A}}, \longrightarrow store$$

$$\begin{cases} r_s = a[\mathfrak{Y}\mathfrak{D} : \mathfrak{Y}\mathfrak{I}] = (\bullet \bullet \bullet \mathfrak{I}\mathfrak{I})_{\mathfrak{Y}} = (\mathfrak{Y})_{\mathfrak{I}}, \longrightarrow v_{\mathfrak{I}} \\ r_t = a[\mathfrak{Y} \bullet : \mathfrak{I}\mathfrak{E}] = (\bullet \bullet \bullet \mathfrak{I} \bullet)_{\mathfrak{Y}} = (\mathfrak{Y})_{\mathfrak{I}}, \longrightarrow v, \longrightarrow sw \ \$v \bullet, \mathfrak{Y} \bullet (\$v_{\mathfrak{I}}) \\ immediate = a[\mathfrak{I}\mathfrak{D} : \bullet] = (\bullet \bullet \bullet \bullet \bullet \bullet \bullet \bullet \bullet \bullet \mathfrak{I} \bullet \mathfrak{I} \bullet \bullet)_{\mathfrak{Y}} = (\mathfrak{Y} \bullet)_{\mathfrak{I}}, \end{cases}$$

$$opcode_b = b[\mathfrak{V}\mathfrak{I} : \mathfrak{V}\mathfrak{E}] = (\cdot \cdot \cdot \cdot \cdot)_\mathfrak{I} = (\cdot)_\mathfrak{I} \cdot \longrightarrow R - Type$$

$$\begin{cases} r_s = b[\mathfrak{Y}\mathfrak{O} : \mathfrak{Y}\mathfrak{I}] = (\bullet\bullet\mathfrak{I}\bullet\bullet)_{\mathfrak{Y}} = (\mathfrak{Y})_{\mathfrak{I}}, \longrightarrow a, \\ r_t = b[\mathfrak{Y}\bullet : \mathfrak{I}\mathfrak{F}] = (\bullet\bullet\bullet\mathfrak{I}\bullet)_{\mathfrak{Y}} = (\mathfrak{Y})_{\mathfrak{I}}, \longrightarrow v, \\ r_d = b[\mathfrak{I}\mathfrak{O} : \mathfrak{I}\mathfrak{I}] = (\bullet\bullet\bullet\bullet\mathfrak{I})_{\mathfrak{Y}} = (\mathfrak{I})_{\mathfrak{I}}, \longrightarrow a_t \\ function = b[\mathfrak{O} : \bullet] = (\mathfrak{I}\bullet\mathfrak{I}\bullet\mathfrak{I}\bullet)_{\mathfrak{Y}} \longrightarrow set\ on\ less\ than \end{cases} \longrightarrow slt\ \$a_t, \$a., \$v.$$

ب

خروجی sign extend دستور a ، ۱۶ بیت اول دستور را به یک عدد ۳۲ بیتی تبدیل می‌کند با حفظ علامت:
 $output = (000000000000000010100)_2 = (14)_{16}$

خروجی واحد shift left که توسط دستور b اجرای می‌شود، خروجی واحد sign extend را می‌گیرد و ۲ بیت به چپ شیفت می‌دهد.

$output = (010000010000010000010101000)_2 = (20820A8)_{16}$

ج

در این بخش باید مقادیر واحد کنترل ALU را به ازای هر دستور محاسبه کنیم. از تصویر زیر کمک می‌گیریم:

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

$$a \rightarrow lw \rightarrow \begin{cases} ALUOp = 00 \\ funct = a[5:0] = \times \end{cases}$$

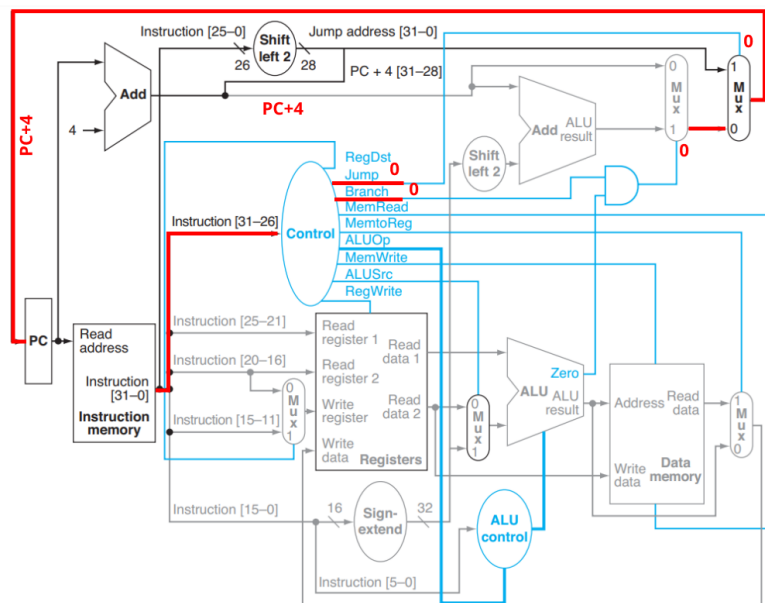
$$b \rightarrow RType \rightarrow \begin{cases} ALUOp = 10 \\ funct = b[5:0] = 101010 \end{cases}$$

د

با توجه به این‌که هیچ کدام از دستورات مربوط به دستورات jump و یا branch نیستند، پس در هر دو حالت مقدار برابر است با:

$$PC \leftarrow PC + 4$$

در شکل زیر مسیرهایی که از آن‌ها این مقادیر به دست می‌آیند را می‌بینید:

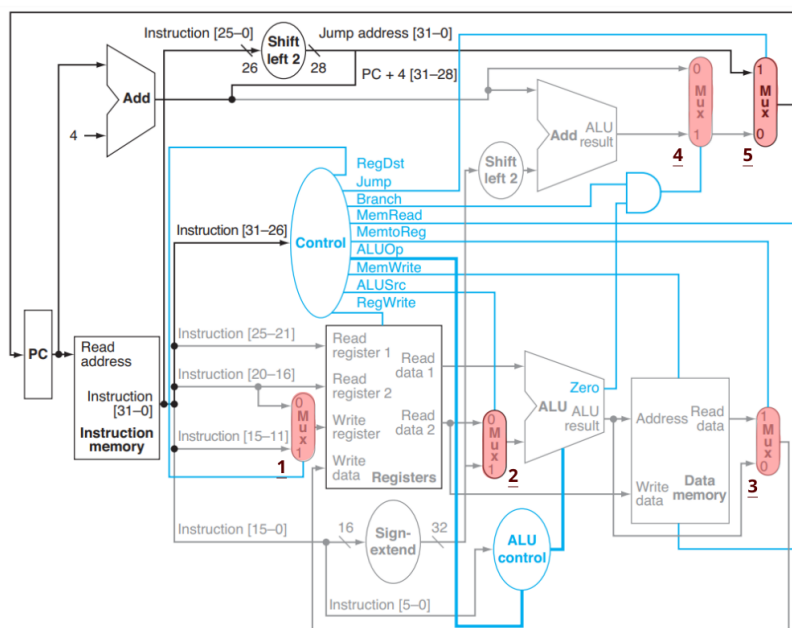


مقدار موجود در هر یک از ثبات‌ها به شرح زیر است:

	R0	R1	R2	R3	R4	R5	R6	R8	R12	R31
a	0	-1	2	-3	-4	10	6	8	2	-16
b	0	256	-128	19	-32	13	-6	-1	16	-2

۵

در این بخش باید مقادیر خروجی هر یک از MUX ها را مشخص کنیم. ابتدا به سراغ دستور a می‌رویم:



۱. چون دستور ما sw است و RegDst برابر صفر است، پس خروجی این مالتیپلکسر برابر است با:
 $MUX_1 = Instruction[20:16] = (00010)_2 = (2)_{10} \rightarrow \$R2$

۲. چون سیم مربوط به ALUSrc برابر یک است، پس حاصل واحد Sign Extend مقدار خروجی آن است:
 $MUX_2 = SignExtend(Instruction[15:0]) = (14)_{16} = (20)_{10} \rightarrow \$R20$

۳. خروجی این مالتیپلکسر برای ما فاقد اهمیت است.
 $MUX_3 = \times$

۴. سیم مربوط به Branch صفر است، پس خروجی این مالتیپلکسر برابر است با:
 $MUX_4 = PC + 4$

۵. چون این دستور مربوط به Jump نیست و سیم مربوط به آن صفر است، پس خروجی ما برابر است با:
 $MUX_5 = PC + 4$

حال به سراغ دستور b می‌رویم:

۱. مقدار سیم مربوط به RegDst برابر یک است، پس rd خروجی داده می‌شود:
 $MUX_1 = Instruction[15:11] = (00001)_2 = (1)_{10}$

۲. چون سیم مربوط به ALUSrc برابر صفر است، مقدار ReadData₂ خروجی داده می‌شود:
 $MUX_2 = Instruction[20:16] = \$rt = \$R2$

۳. چون سیم مربوط به MemtoReg قطع است، پس خروجی این مالتیپلکسر برابر است با:
 $MUX_3 = ALUResult = slt \ \$a_t, \$a_0, \$v_0 = 0$

۴. بیت کنترلی Branch صفر چون دستور ما مربوط به Branch یا Jump نیست، پس خروجی برابر است با:
 $MUX_4 = PC + 4$

۵. به همان دلیلی که در مورد قبل گفته شد، خروجی برابر است با:
 $MUX_5 = PC + 4$

و

در این بخش باید ورودی‌های ALU و واحدهای Add موجود در این معماری را محاسبه کنیم. ابتدا به سراغ دستور a می‌رویم:

۱. ورودی‌های واحد ALU معماری:

$$\begin{cases} Read_1 = Instruction[25 - 21] = rs = R_3 = (-3)_1 \\ Read_2 = Instruction[20 - 16] = address = (20)_1 \end{cases}$$

۲. ورودی‌های واحد Add سمت چپ معماری:

$$\begin{cases} input_1 = PC \\ input_2 = 4 \end{cases}$$

۳. ورودی‌های واحد Add سمت راست معماری:

$$\begin{cases} input_1 = PC + 4 \\ input_2 = 4 \times Address = 80 \end{cases}$$

حال به سراغ دستور b می‌رویم:

۱. ورودی‌های واحد ALU معماری:

$$\begin{cases} Read_1 = Instruction[25 - 21] = rs = R_2 = -128 \\ Read_2 = Instruction[20 - 16] = rt = R_4 = -32 \end{cases}$$

۲. ورودی‌های واحد Add سمت چپ معماری:

$$\begin{cases} input_1 = PC \\ input_2 = 4 \end{cases}$$

۳. ورودی‌های واحد Add سمت راست معماری:

$$\begin{cases} input_1 = PC + 4 \\ input_2 = 4 \times Address = 8360 \end{cases}$$

ز

در این بخش باید مقادیر مختلف Register File را به ازای هر دو دستور مشخص کنیم. به ازای دستور a مقادیر این ثبات‌ها برابر هستند با:

- $\text{ReadRegister}_1 = 3$
- $\text{ReadRegister}_2 = 2$
- $\text{WriteRegister} = 2$
- $\text{WriteData} = \times$
- $\text{RegWrite} = 0$

حال به سراغ دستور b می‌رویم:

- $\text{ReadRegister}_1 = 4$
- $\text{ReadRegister}_2 = 2$
- $\text{WriteRegister} = 1$
- $\text{WriteData} = 0$
- $\text{RegWrite} = 1$