به نام خدا

معماری کامپیوتر تمرین 6

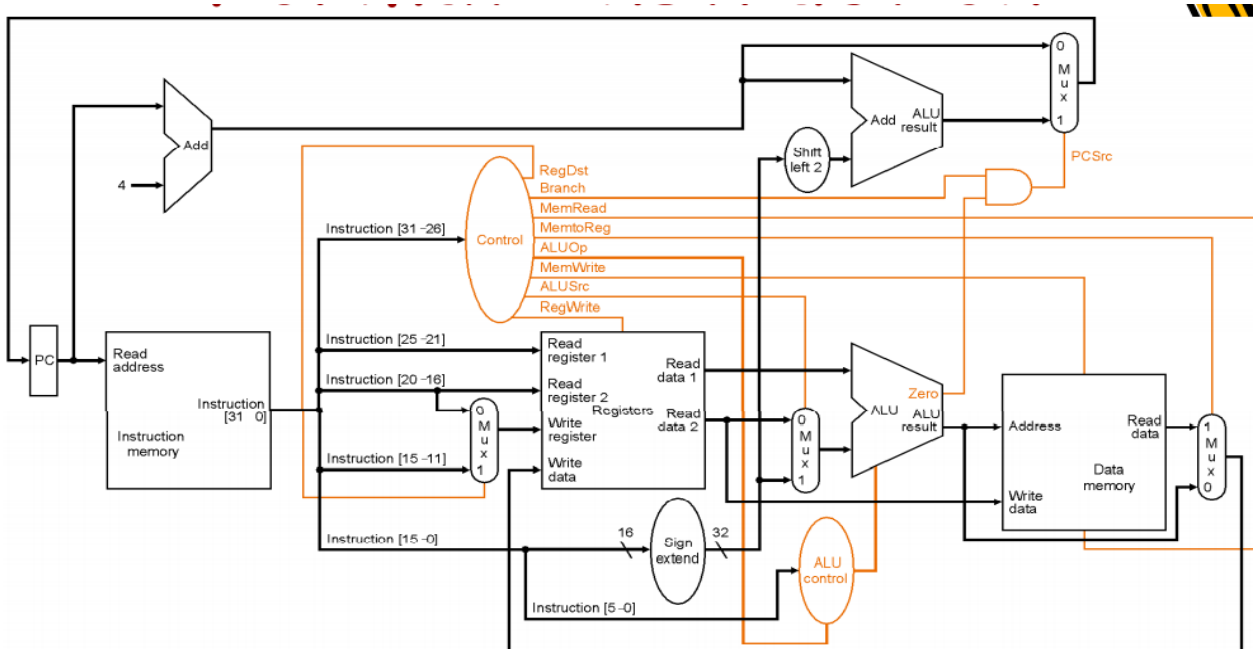
جناب آقای دکتر اسدی

سارا آذرنوش

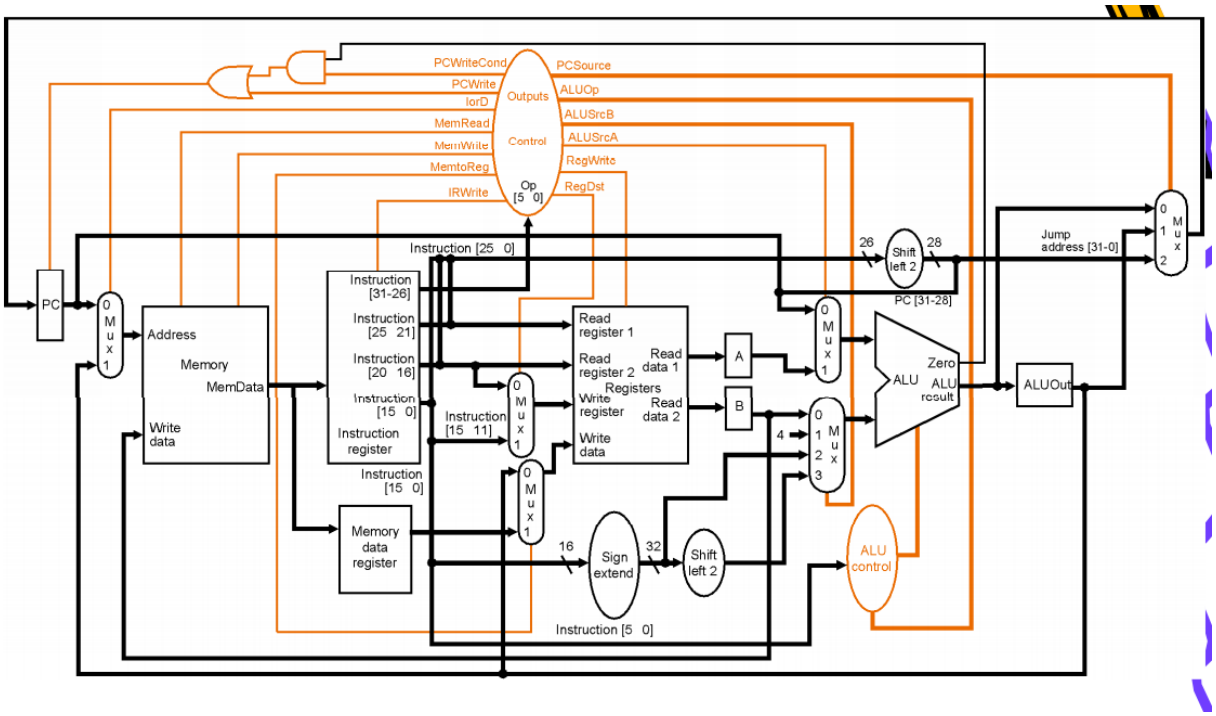
98170668

(1

Single cycle



Multi cycle



1) رجیسترهای اضافه برای ذخیره کردن مقادیر در مولتی ساسیکل نیاز است.

2) در مولتی ساسیکل به یک alu به جای 2 alu و یک adder در سینگل سایکل نیاز است.

3)در سینگل سایکل به instruction memory و datamemory جداگانه نیاز است اما در مولتی سایکل یک memory نیاز است.

2)

اگر در یک گیر کند دستوراتی که با 0 انجام میشوند انجام نمیشوند و بالعکس و مقدار هر سیگنال در دستورات و کلاک های متفاوت به شکل زیر است.

T0: instruction fetch

T1: id/ register fetch

T2: execution, address computation,branch/jump

T3: memory access or R-type

T4: memory read

Ex = execute wb= write back m= memory

T	IorD	Mem Read	Mem Write	IR Write	Reg Dst	Mem toReg	Reg Write	Ext Op	ALU SrcA	ALU SrcB	ALU Op	PC Src	PC WrCd	PC Wr	
T0	0	1	0	1	x	x	0	x	0	1	add	0	0	1	IF
T1	x	0	0	0	x	x	0	1	1	3	add	x	0	0	ID
T2	x	0	0	0	x	x	0	x	1	0	fun	x	0	0	Ex R-T
T3	x	0	0	0	1	0	1	x	x	x	x	x	0	0	Wb R-T
T2	x	0	0	0	x	x	0	0	1	2	or	x	0	0	Ex ORI
T3	x	0	0	0	0	0	1	x	x	x	x	x	0	0	Wb ORI
T2	x	0	0	0	x	x	0	1	1	2	add	x	0	0	Ex LW
T3	1	1	0	0	x	x	0	x	x	x	x	x	0	0	M LW
T4	x	0	0	0	0	1	1	x	x	x	x	x	0	0	Wb LW
T2	x	0	0	0	x	x	0	1	1	2	add	x	0	0	Ex SW
T3	1	0	1	0	x	x	0	x	x	x	x	x	0	0	M SW
T2	x	0	0	0	x	x	0	x	x	x	sub	1	1	0	Ex BEQ
T2	x	0	0	0	x	x	0	x	x	x	x	2	0	1	Ex J

	1	0
Regwrite:	بقیه به جز 3 حالت	T3 wb r-t/ t4 wb lw/ t3 wb ori
Memread:	بقیه به جز 2 حالت	T0 if/ t3 m lw
Memwrite:	حالت1بقیه به جز	T3 m sw
Irwrite:	بقیه به جز 1 حالت	T0 if
Pcwrite:	بقیه به جز 1 حالت	T0 if /t2 exj
Pcwritecond:	بقیه به جز 1 حالت	T2 ex beq

(3

	I-cache	Decode, R-Read	ALU	PC update	D-cache	R-Write	Total
R-type	1	1	.9	-	-	.8	3.7
Load	1	1	.9	-	1	.8	4.7
Store	1	1	.9	-	1	-	3.9
beq	1	1	.9	.1	-	-	3.0

(setup time = hold time = 0.15 ns)

	Instruction-cache	Register-Read	ALU	PC update	Data-cache	Register-Write
Delay	3 ns	2 ns	1.5 ns	0.5 ns	3 ns	2.5 ns

Setup + hold = .3

الف) کلاک به اندازه بیشترین زمان دستور است که lw بیشترین زمان را دارد =>

Cct => lw = 3 + 2 + 1.5 + 3 + 2.5 = 12 => +.3 = 12.3

ب)

	Instruction-cache	Register-Read	ALU	PC update	Data-cache	Register-Write
Delay	3 ns	2 ns	1.5 ns	0.5 ns	3 ns	2.5 ns

Cct = 3 + .3 = 3.3

r-type = total =3 + 2 + 1.5 + 2.5 = 9

clk = 4

4 * 3.3 = 13.2

lw = 3 + 2 + 1.5 + 3 + 2.5 = 12

clk = 5

5 * 3.3 = 16.5

sw = 3 + 2 + 1.5 + 3 = 9.5

clk = 4

4 * 3.3 = 13.2

beq = 3 + 2 + 1.5 + .5= 7

clk =3

3 * 3.3 = 9.9

ج)

خیر بسته به نوع و تعداد دستورات و زمان ان ها میتواند هر یک بهتر باشد برای مثال اگر cpi را مانند کلاس 4.1 بگیریم و از کلاک سایکل های بدست آمده استفاده کنیم single cycle بهتر خواهد بود.

Time/inst = 1 *12.3 = 12.3

Cpi = 4.1

Time /inst = 4.1 * 3.3 = 13.53

4)

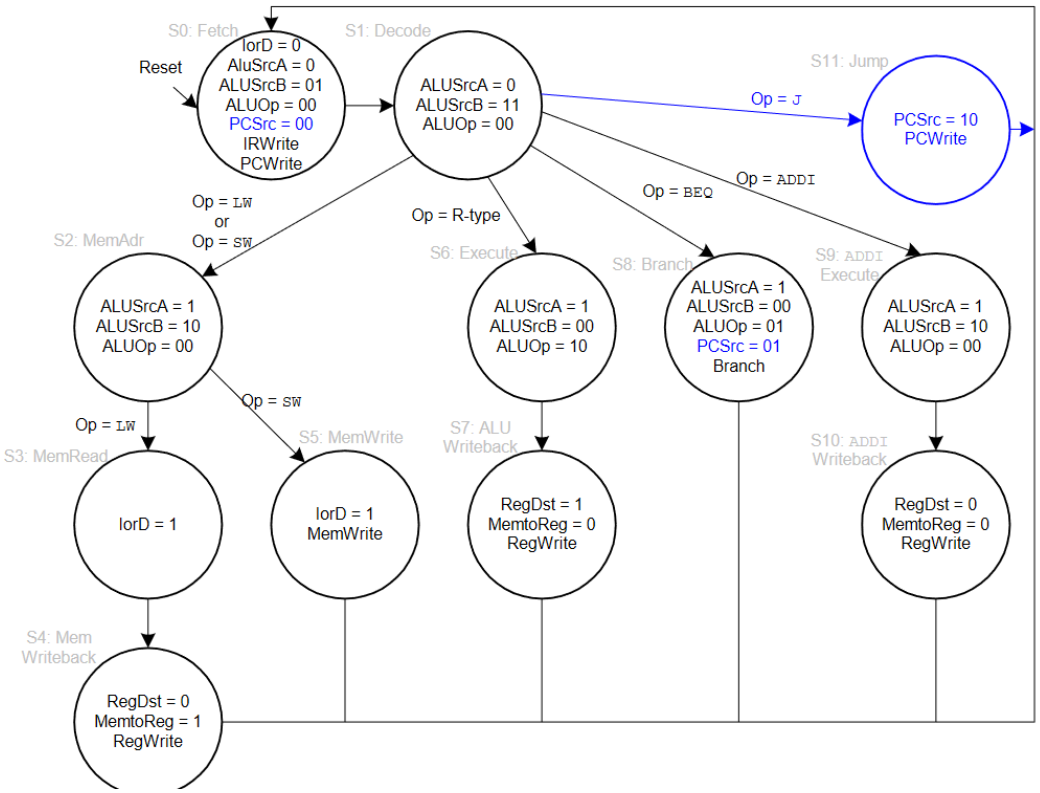
.با توجه به اعمالی که باید صورت بگیرد سیگنال های کنترلی به شکل زیر خواهند بود

2 کلاک ابتدایی برای همه مشترک و کلاک بعدی دستورات به صورت زیر خواهند بود.

Addi => 3: aluSrcA =1 , aluSrcb = 100 , aluOp =00

4: regDest =0 , memtoReg =0 regwrite

Jump => pcSrc =10 pcWrite



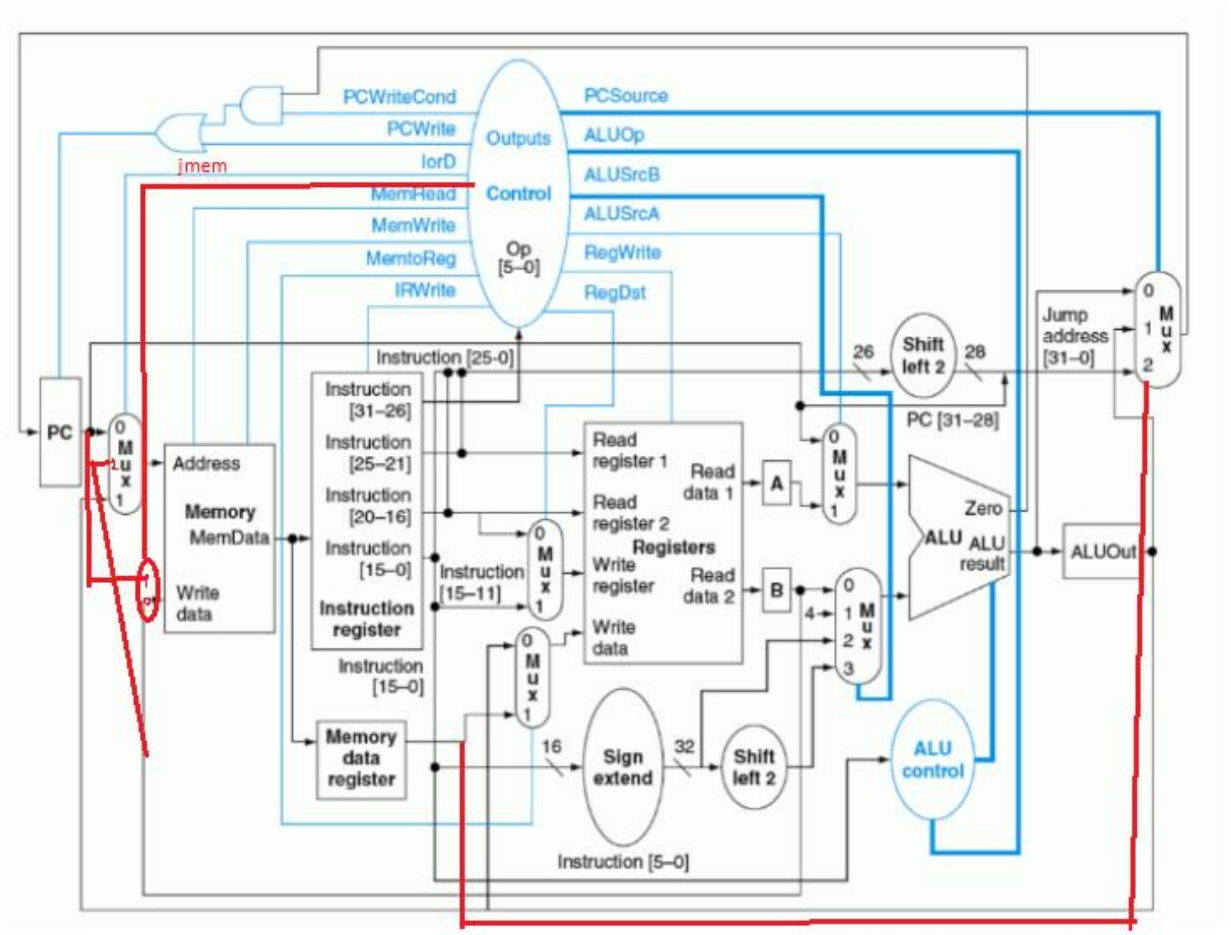
(5

Jmem rt, offset(rs)

Memory[R[rs]+offset] = PC+4;

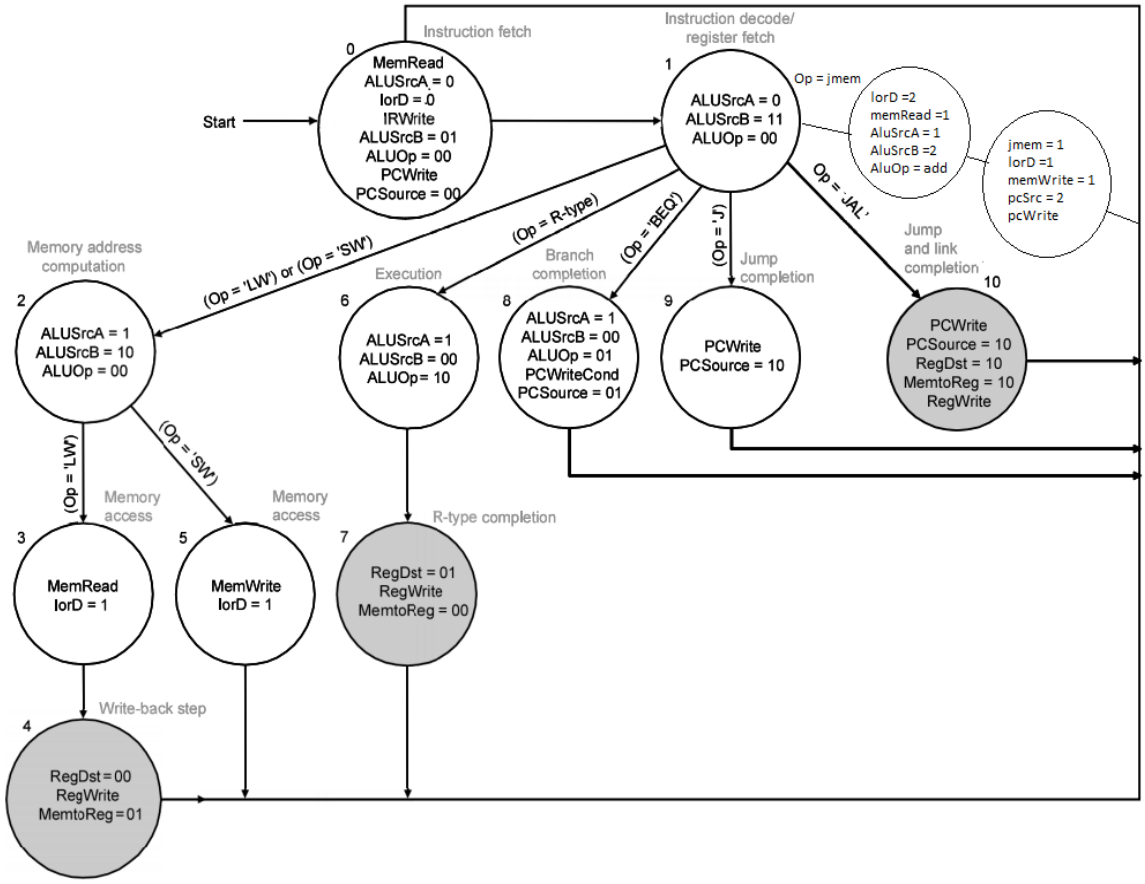
PC = Memory[R[rt]];

یک سیم باید از مقدار خوانده شده از رجیستر b که شامل r[rs] + offset است باید به ماکس متصل به ادرس نوشتن memdata وجود داشته باشد و یک سیم از مقدار pc که به قسمت نوشتن ان متصل است ایجاد میکنیم ویک سیم که محتویات نوشته شده درmemory dataregister را به مقدار جدید pc بدهد



در چرخه 3 مقدار خوانده شده در rf+4 شده و مقدار b در memory نوشته میشود بنابراین سیگنال های مرتبط به آن فعال هستند.

در چرخه 4 مقدار pc در ماکس جدید در آدرسaluout نوشته میشود و مقدار mdr در pc میرود و آن سیگنال ها فعال هستند.



(6

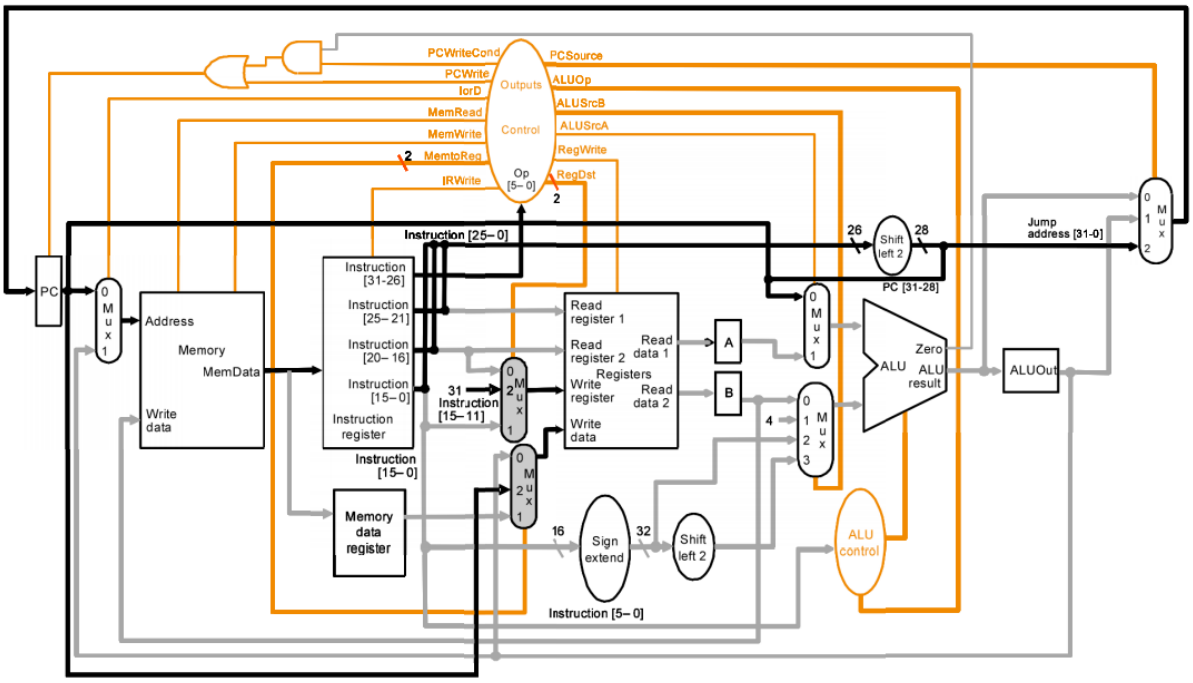
Jal => jump and link

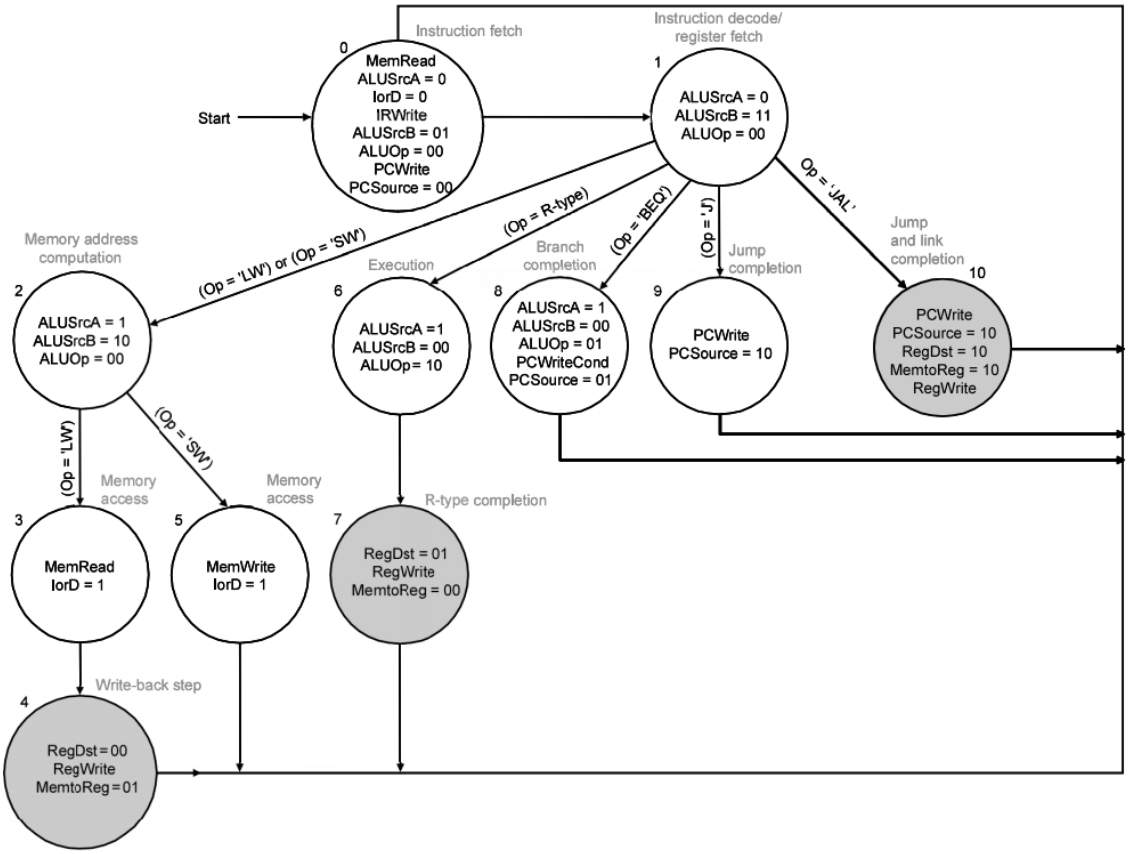
Reg[31] = pc +4

Pc = pc[31:28] ||(ir[25:0] << 2)

به 3 سیکل نیاز است دیکود، اینکود و پرش و لینک.

در اینجا لینک نیز انجام میشود بنابراین مقدار به عنوان ادرس وارد rf میشود و نیاز است 3 mux تایی شود.


PCWriteCond
PCWrite
IorD
MemRead
MemWrite
MemtoReg
IRWrite
Outputs
Control
Op [5–0]
PCSource
ALUOp
ALUSrcB
ALUSrcA
RegWrite
RegDst
Instruction [25–0]
26
Shift left 2
28
PC [31-28]
Jump address [31-0]
PC
Address
Memory
MemData
Write data
Instruction [31-26]
Instruction [25–21]
Instruction [20–16]
Instruction [15–0]
Instruction register
31
Instruction [15–11]
Instruction [15–0]
Memory data register
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
A
B
4
ALU
Zero
ALU result
ALUOut
16
Sign extend
32
Shift left 2
ALU control
Instruction [5–0]


Instruction fetch
Instruction decode/ register fetch
Start
0
MemRead
ALUSrcA = 0
IorD = 0
IRWrite
ALUSrcB = 01
ALUOp = 00
PCWrite
PCSource = 00
1
ALUSrcA = 0
ALUSrcB = 11
ALUOp = 00
Op = 'JAL'
(Op = 'LW') or (Op = 'SW')
(Op = R-type)
(Op = 'BEQ')
(Op = 'J')
Memory address computation
Execution
Branch completion
Jump completion
Jump and link completion
2
ALUSrcA = 1
ALUSrcB = 10
ALUOp = 00
6
ALUSrcA =1
ALUSrcB = 00
ALUOp= 10
8
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 01
PCWriteCond
PCSource = 01
9
PCWrite
PCSource = 10
10
PCWrite
PCSource = 10
RegDst = 10
MemtoReg = 10
RegWrite
(Op = 'LW')
(Op = 'SW')
Memory access
Memory access
R-type completion
3
MemRead
IorD = 1
5
MemWrite
IorD = 1
7
RegDst = 01
RegWrite
MemtoReg = 00
Write-back step
4
RegDst = 00
RegWrite
MemtoReg = 01

7)

$r0 <- MEM[MEM[(Base,IMM)]]

Ldi مقدار را لود کرده و سپس همان را به عنوان ادرس داده و لود میکند.

Lw در پیاده سازی موجود است و یکبار مقدار را لود و در رجیستر مینویسد. پس برای خواندن مجدد از آدرس خوانده شده یه دستور lw یک سیکل برای خواندن ان ادرس جدید از memory پیش از نوشتن در رجیستر و پس از خواندن از ادرس(که در lw نیز موجود است) اضافه میکنیم تا پس از خواندن اولیه دوباره با آن ادرس مقدار را بخواند و ذخیره کند .

برای اینکار mdr را به ماکس متصل به memdata وصل کرده و 3 ورودی و با دوبیت سیگنال میشود در واقع (lorD) نیز 2 بیتی میشود و به شکل زیر با مقادیر زیر در می آید.

T0 : IR <- M[PC]

PC <- PC + 4

T1:A <- RF[rs]

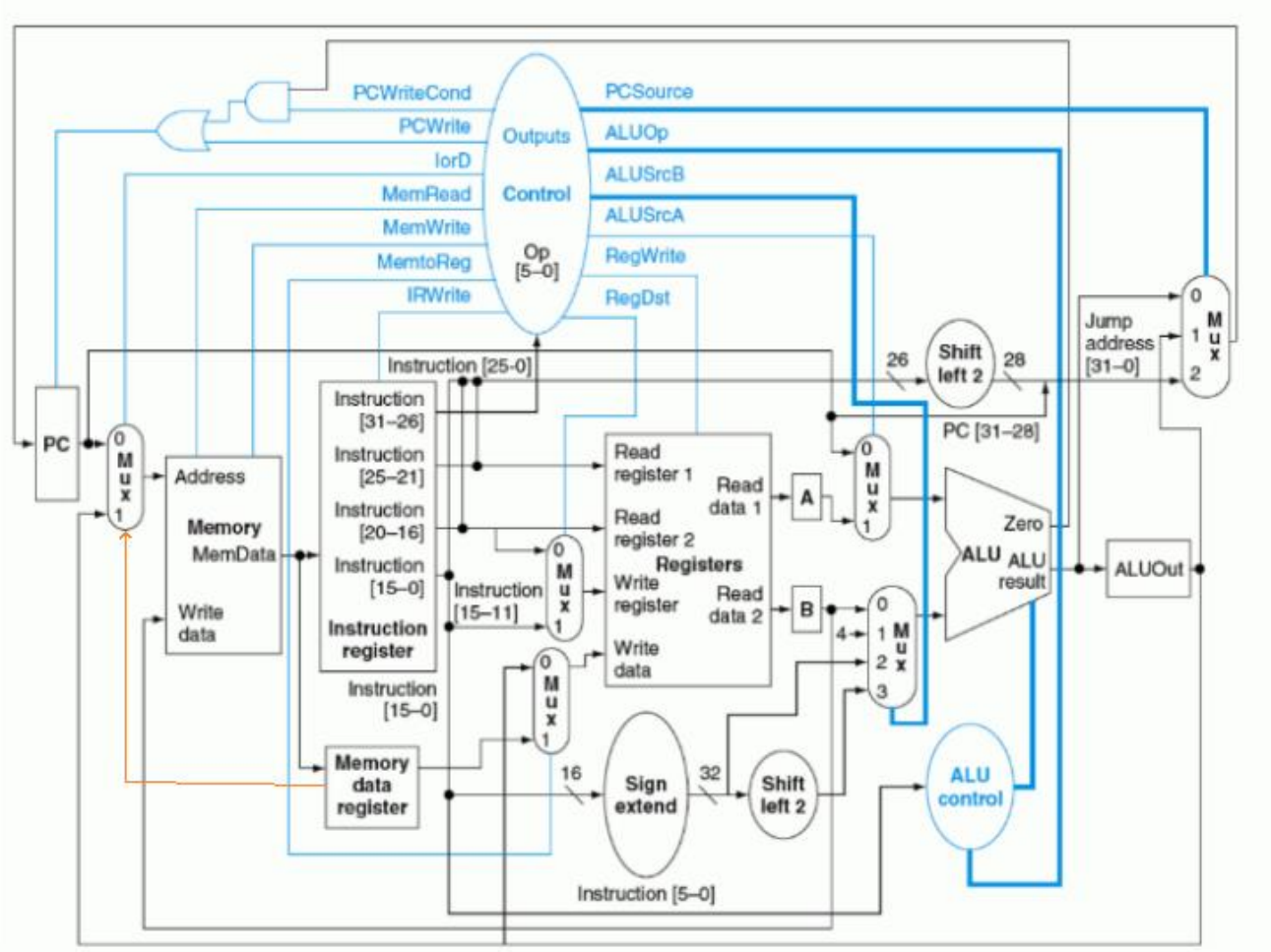
B <- RF[rt]

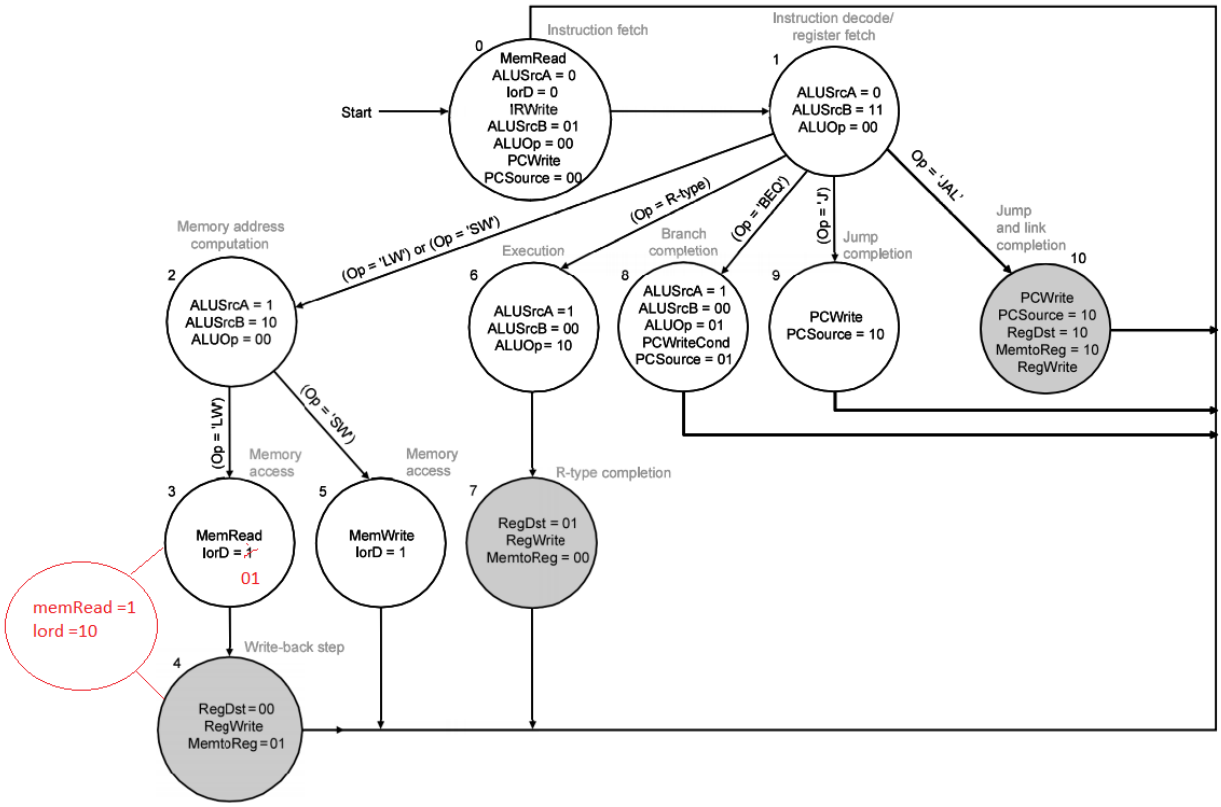
T2: ALUOut <- A + Ext(imm)

T3 : MDR <- M[ALUOut]

T4: MDR <- M[MDR]

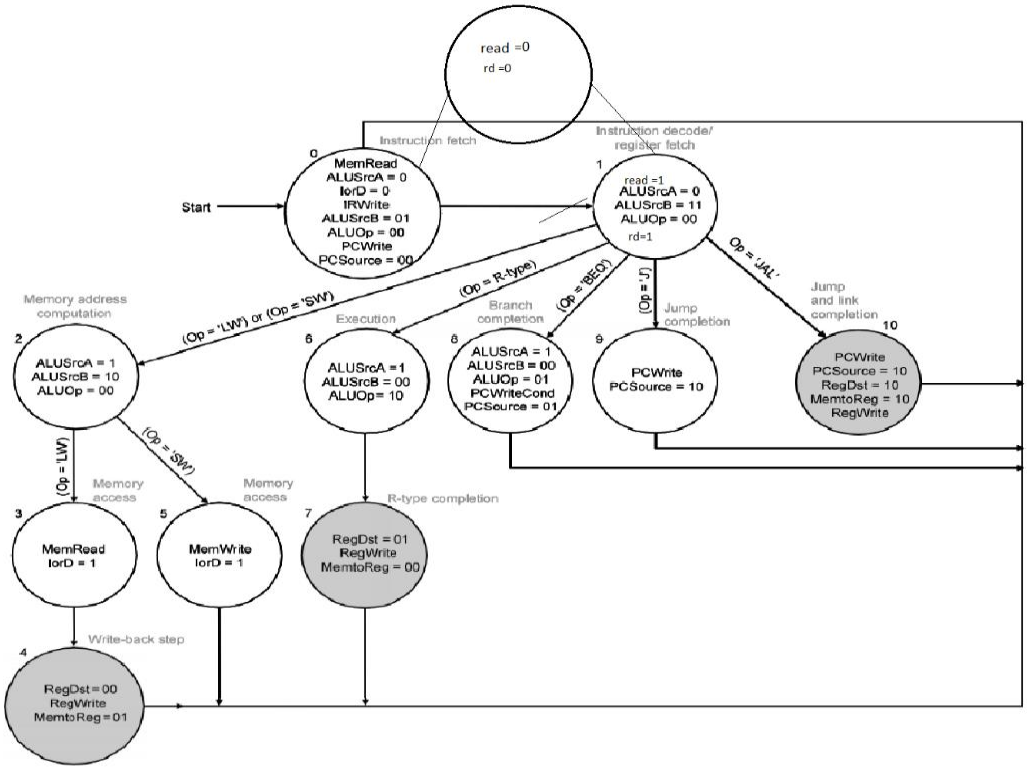
T5: RF[rt] <- MDR

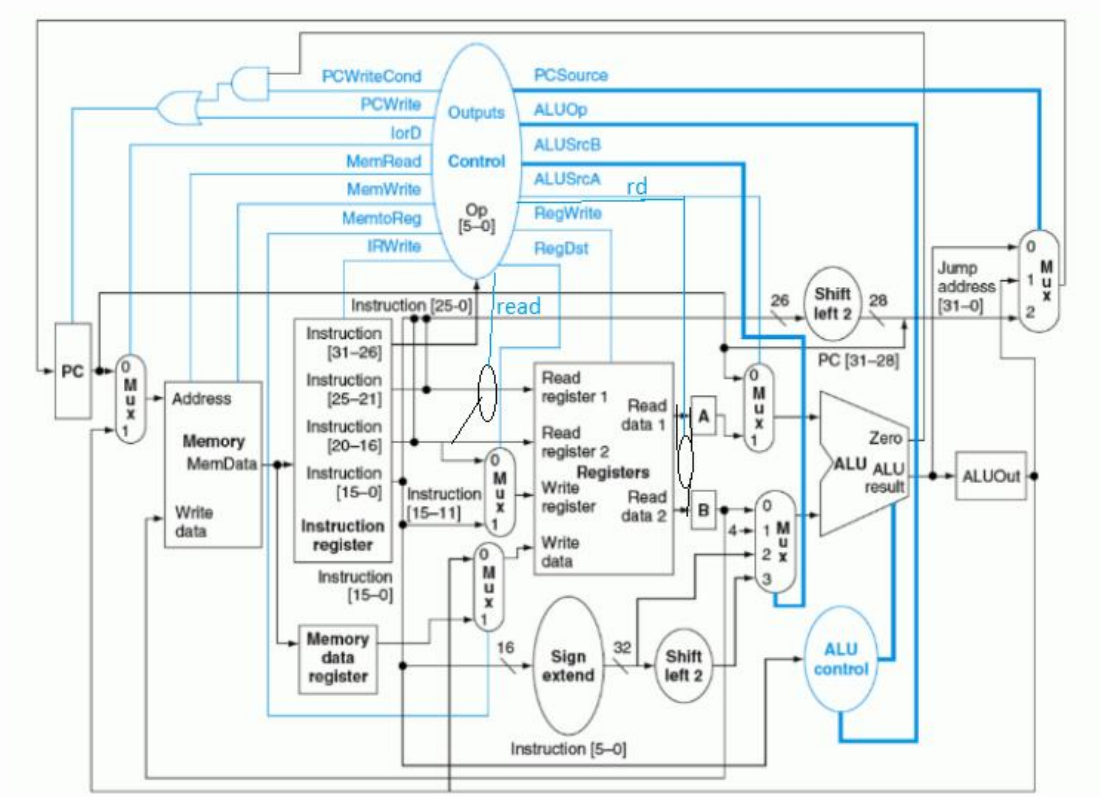

PCWriteCond
PCWrite
IorD
MemRead
MemWrite
MemtoReg
IRWrite
Outputs
Control
Op
[5–0]
PCSource
ALUOp
ALUSrcB
ALUSrcA
RegWrite
RegDst
Instruction [25-0]
26
Shift
left 2
28
PC [31–28]
Jump
address
[31–0]
M
u
x
0
1
2
PC
Address
Memory
MemData
Write
data
Instruction
[31–26]
Instruction
[25–21]
Instruction
[20–16]
Instruction
[15–0]
Instruction
register
Instruction
[15–0]
Instruction
[15–11]
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
A
B
4
Zero
ALU
ALU
result
ALUOut
Memory
data
register
16
Sign
extend
32
Shift
left 2
ALU
control
Instruction [5–0]



(8

اگر به درستی خواسته مسئله را متوجه شده باشم در واقع یک ورودی برای نوشتن داریم اگر 2 تا باشد ممکن است ادرس ها برابر باشد و از یک رجیستر بخواند و توانایی همزمان خواندن از یک ثبات ندارد بنابراین در کلاک دوم که دیکود است و در هر حالتی که دو مقدار وارد میشوند خواندن را یک کلاک بین کلاک 1 و 2 اضافه کنیم و یک ماکس در ورودی گذاشته تا مقدار صحیح را انتخاب کند ویک دیکودر در خروجی که متناسب ورودی به رجیستر دهد.


read =0
rd =0
Instruction fetch
Instruction decode/
register fetch
0
MemRead
ALUSrcA = 0
IorD = 0
IRWrite
ALUSrcB = 01
ALUOp = 00
PCWrite
PCSource = 00
Start
1
read =1
ALUSrcA = 0
ALUSrcB = 11
ALUOp = 00
rd=1
(Op = 'LW') or (Op = 'SW')
(Op = R-type)
(Op = 'BEQ')
(Op = 'J')
Op = 'JAL'
Memory address
computation
Execution
Branch
completion
Jump
completion
Jump
and link
completion
2
ALUSrcA = 1
ALUSrcB = 10
ALUOp = 00
6
ALUSrcA = 1
ALUSrcB = 00
ALUOp= 10
8
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 01
PCWriteCond
PCSource = 01
9
PCWrite
PCSource = 10
10
PCWrite
PCSource = 10
RegDst = 10
MemtoReg = 10
RegWrite
(Op = 'LW')
(Op = 'SW')
Memory
access
Memory
access
R-type completion
3
MemRead
IorD = 1
5
MemWrite
IorD = 1
7
RegDst = 01
RegWrite
MemtoReg = 00
Write-back step
4
RegDst = 00
RegWrite
MemtoReg = 01


PCWriteCond
PCWrite
IorD
MemRead
MemWrite
MemtoReg
IRWrite
Outputs
Control
Op
[5–0]
PCSource
ALUOp
ALUSrcB
ALUSrcA
RegWrite
RegDst
rd
read
Instruction [25-0]
26
Shift
left 2
28
PC [31–28]
Jump
address
[31–0]
PC
Address
Memory
MemData
Write
data
Instruction
[31–26]
Instruction
[25–21]
Instruction
[20–16]
Instruction
[15–0]
Instruction
register
Instruction
[15–0]
Instruction
[15–11]
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
A
B
4
Zero
ALU
ALU
result
ALUOut
Memory
data
register
16
Sign
extend
32
Shift
left 2
ALU
control
Instruction [5–0]
M
u
x