CS224

Lab 4

Sec. 5

Suleyman Hanyyev

21901009

Part 1:

A)

Location(hex)	Machine Instruction(hex)	Assembly Equivalent
0	20020005	addi $v0, $zer0 5
8’h04	“2003000c”	addi $v1, $zero, 12
8'h08	“2067fff7”	addi $a3, $v1, -9
8'h0c	“00e22025”	or $a0, $a3, $v0
8'h10	“00642824”	and $a2, $v1, $a0
8'h14	“00a42820”	add $a1, $a0, $a1
8'h18	“10a7000a”	Beq $a1, $a3, target (pc+44)
8'h1c	“0064202a”	Slt $a0, $v1, $a0
8'h20	“10800001”	beq $a0, $zero, pc+ 8
8'h24	“20050000”	addi $a1, $zero, 0
8'h28	“00e2202a”	slt $a0,$a3,$v0
8'h2c	“00853820”	add $a3,$a1,$a2
8'h30	“00e23822”	sub $a3,$a3,$v0
8'h34	“ac670044”	sw $a3,68($v1)
8'h38	“8c020050”	lw $v0,80($zero)
8'h3c	“08000010”	j 0x80000040
8'h40	“001f6020”	add $t4,$zero, $ra
8'h44	“0c000012”	jal 0x80000048
8'h48	“ac020054”	sw $v0,84($0)
8'h4c	“00039042”	srl $s2,$v1,0x1
8'h50	“03E00008”	jr $ra

F)

i) it corresponds to RD2.

ii) I think it is because in one clock cycle it becomes AluSrcB and the actual writedata is no longer needed and therefore is undefined.

iii) Because it is only used in lw function.

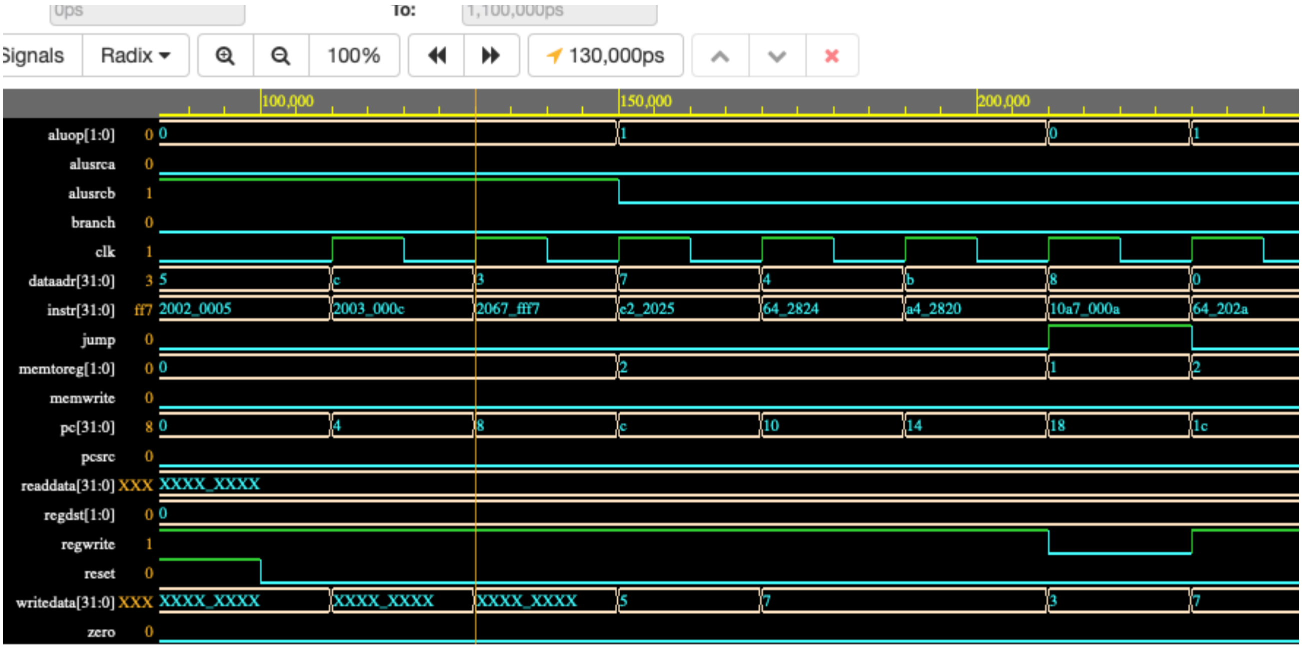
iv) To the result out of ALU.

v) it becomes aluout

G)

i) Yes, we will need to make alusrcA mux 4:1 not 2:1. This is because the mux has to know whether it should take shame RD1 or the rs.

ii) I don't think we need to modify anything. Part 2:


0ps
To:
1,100,000ps
Signals
Radix
100%
130,000ps
100,000
150,000
200,000
aluop[1:0] 0 0 1 0 1
alusrca 0
alusrcb 1
branch 0
clk 1
dataadr[31:0] 3 5 c 3 7 4 b 8 0
instr[31:0] ff7 2002_0005 2003_000c 2067_fff7 e2_2025 64_2824 a4_2820 10a7_000a 64_202a
jump 0
memtoreg[1:0] 0 0 2 1 2
memwrite 0
pc[31:0] 8 0 4 8 c 10 14 18 1c
pcsrc 0
readdata[31:0] XXX XXXX_XXXX
regdst[1:0] 0 0
regwrite 1
reset 0
writedata[31:0] XXX XXXX_XXXX XXXX_XXXX XXXX_XXXX 5 7 3 7
zero 0

revert to EPWave opening in a new browser window, set that option on your user page.

Part 2:

A) IM[PC] , RF[Instr(25:21)] RF[Instr(20:16)]

ALUResult = RF[rs] - RF[rt]

if(zero = 0) then PC <- PC+4 + BTA

B) I did not make any additional changes, simply because I didn’t think I needed them. I will use the a < b operation in the alu and if it is less, the flag zero will be 1 and I can use that for the BTA

C)

Instruction	Opcode	RegWrite	RegDst	ALUSrcA	ALUSrcB	Branch	MemWrite	MemToReg	ALUOp	Jump
lw	100011	1	00	0	1	0	0	01	00	00
sw	101011	0	X	0	1	0	1	XX	00	‘00
beq	000100	0	X	0	0	1	0	01	01	00
addi	001000	1	00	0	1	0	0	00	00	00
j	000010	0	X	X	X	X	0	XX	XX	01
jal	000011	1	10	X	X	X	0	10	XX	01
ble	‘010101	0	XX	0	0	1	0	XX	11	00
srl	000000	1	01	0	0	0	0	00	10	00
R	000000	1	‘01	0	0	0	0	00	10	00

ALU Decoder:

AluOP	Funct	ALUcontrol
00	X	010(add)
01	X	110(subtract)
11	X	111(slt)
XX	100000 (add)	010(add)
XX	100010 (sub)	110(subtract)
XX	100100 (and)	000(and)
XX	100101 (or)	001(or)
XX	101010 (slt)	111(slt)
XX	000010 (srl)	011(srl)