Bilkent University CS 224 Computer Organization

“Design Report” Lab 4 Section 2 | Defne Betül Çiftci | 21802635

15.04.2020

b) [15 points] Determine the assembly language equivalent of the machine codes given in the imem module in the "Complete MIPS model.txt" file posted on Unilica for this lab. In the given System Verilog module for imem, the hex values are the MIPS machine language instructions for a small test program. Dis-assemble these codes into the equivalent assembly language instructions and give a 3-column table for the program, with one line per instruction, containing its location, machine instruction (in hex) and its assembly language equivalent. [Note: you may dis-assembly by hand or use a program tool like the one in Unilica.]

Machine instruction	Assembly instruction	Location
20020005	addi $v0, $zero, 5	00
2003000c	addi $v1, $zero, 12	04
2067fff7	addi $a3, $v1, -9	08
00e22025	or $a0, $a3, $v0	0c
00642824	and $a1, $v1, $a0	10
00a42820	add $a1, $a1, $a0	14
10a7000a	beq $a1, $a3, 10	18
0064202a	slt $a0, $v1, $a0	1c
10800001	beq $a0, $zero, 1	20
20050000	addi $a1, $zero, 0	24
00e2202a	slt $a0, $a3, $v0	28
00853820	add $a3, $a0, $a1	2c
00e23822	sub $a3, $a3, $v0	30
ac670044	sw $a3, 68($v1)	34
8c020050	lw $v0, 80($zero)	38
08000011	j 17	3c
20020001	addi $v0, $zero, 1	40
ac020054	sw $v0, 84($zero)	44
08000012	j 18	48

c) [15points] Register Transfer Level (RTL) expressions for each of the new instructions that you are adding (see list below for your section), including the fetch and the updating of the PC.

nop: this I-type instruction does nothing, changes no values, takes one clock cycle. Except for the opcode, the I-type instruction field values are irrelevant.

IM[PC]

PC <- PC + 4

sw+: this I-type instruction does the normal store, as expected, plus an increment (by 4, since it is a word transfer) of the base address in RF[rs].

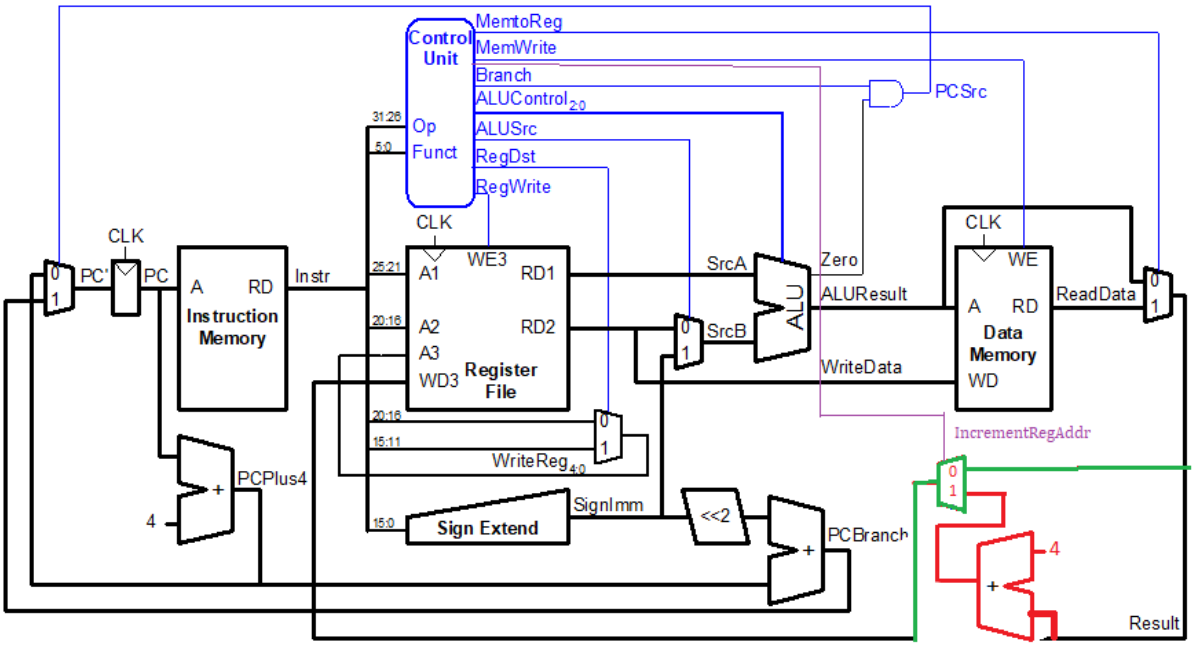
IM[PC]

DM[RF[Rs] + SignExt(immed)] <- RF[Rt]

RF[Rs] <- RF[Rs] + 4

PC <- PC + 4

d) [20 points] Make any additions or changes to the datapath which are needed in order to make the RTLs for the instructions possible. The base datapath should be in black, with changes marked in red and other colors (one color per new instruction).



e) [25points]

Instruction	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	$ALUOp_{1:0}$	jump	IncrementRegAddr
R-type	000000	1	1	0	0	0	0	10	0	0
`lw`	100011	1	0	1	0	0	1	00	0	0
`sw`	101011	0	X	1	0	1	X	00	0	0
`beq`	000100	0	X	0	1	0	X	01	0	0
`addi`	001000	1	0	1	0	0	0	00	0	0
`j`	000010	0	0	0	0	0	0	0	1	0
`nop`	010011*	0	X	X	0	0	X	0X	0	0
`sw+`	010100*	1	X	1	0	1	X	00	0	1

*: I decided these opcodes myself. They are not opcodes of existing instructions.