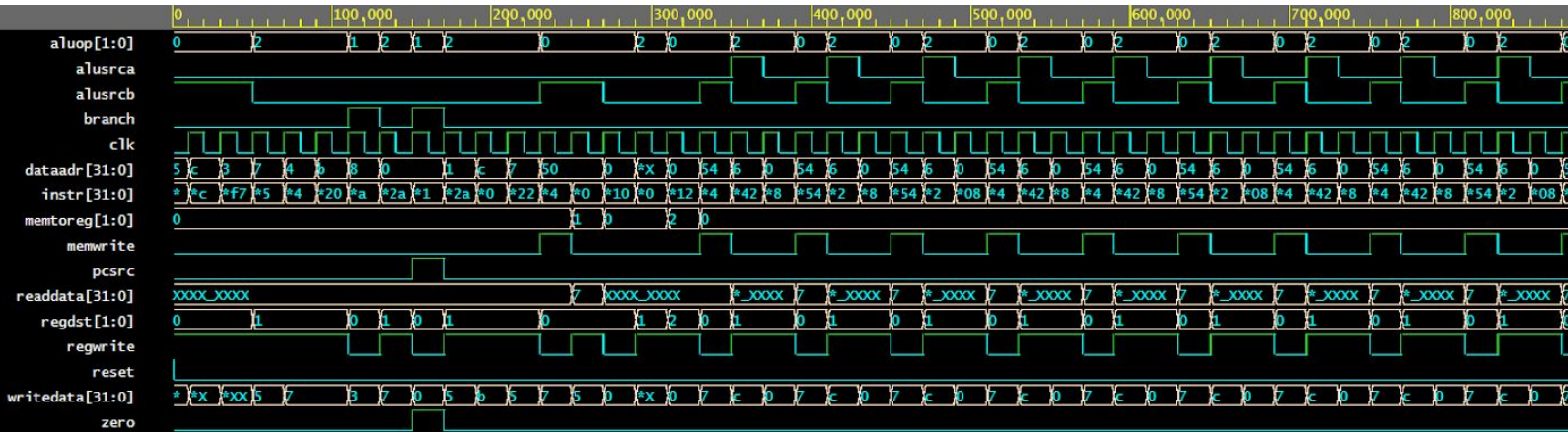
CS224
Lab 4
Section 1
Berk Saltuk Yılmaz
21903419

Part 1.

a)

Location	Machine Code(Hex)	Assembly Equivalent
8'h00	0x20020005	addi $v0, $zero, 5
8'h04	0x2003000c	addi $v1, $zero, 12
8'h08	0x2067fff7	addi $a3, $v1, 0xfff7
8'h0c	0x00e22025	or $a0, $a3, $v0
8'h10	0x00642824	and $a1, $v1, $a0
8'h14	0x00a42820	add $a1, $a1, $a0
8'h18	0x10a7000a	beq $a1, $a3, 0x000a
8'h1c	0x0064202a	slt $a0, $v1, $a0
8'h20	0x10800001	beq $a0, $zero, 0x0001
8'h24	0x20050000	addi $a1, $zero, 0x0000
8'h28	0x00e2202a	slt $a0, $a3, $v0
8'h2c	0x00853820	add $a3, $a0, $a1
8'h30	0x00e23822	sub $a3, $a3, $v0
8'h34	0xac670044	sw $a3, 0x44($v1)
8'h38	0x8c020050	lw $v0, 0x0050($zero)
8'h3c	0x08000010	j 0x0000010
8'h40	0x001f6020	add $t4, $zero, $ra
8'h44	0x0c000012	jal 0x0000012
8'h48	0xac020054	sw $v0, 0x54($zero)
8'h4c	0x00039042	srl $s2, $v1, 0x1
8'h50	0x03E00008	jr $ra

e)



f)

i) <u>In an R-type instruction what does writedata correspond to?</u>

This corresponds to the data value that will be written to the destination register.

ii) <u>Why is writedata undefined for some of the early instructions in the program?</u>

Because in early instructions, no value is written to the destination register (For example, no operations such as addition occurred and there were no resulting data to write).

iii) <u>Why is readdata most of the time undefined?</u>

Since there is no data reading or loading from the memory in the general case, readdata stays undefined.

iv) <u>In an R-type instruction what does dataadr correspond to?</u>

The address that corresponds to the address of data we are operating on, such as RF[$rs].

v) <u>Why does dataaddress become undefined at some point?</u>

Since the 16th instruction is a jump instruction and at that point, there is no operation on data values, the data address is not defined.

g)

i) <u>Do you need to make any changes to support srlv instruction?</u>

I would need to get a variable value from the register file in order to shift the amount of the given variable. I could add a new signal to the controller to choose the value from a 2x1 mux: one from the instruction part or one from the variable's value. Another alu code would be needed also.

ii) Which module would you modify to support sll instruction? How would you modify it?

I would add another alu code and funct code for a new case, in this case modifying the aludec is the right thing to do. The shift amount part would be the same but the direction of shifting would differ.

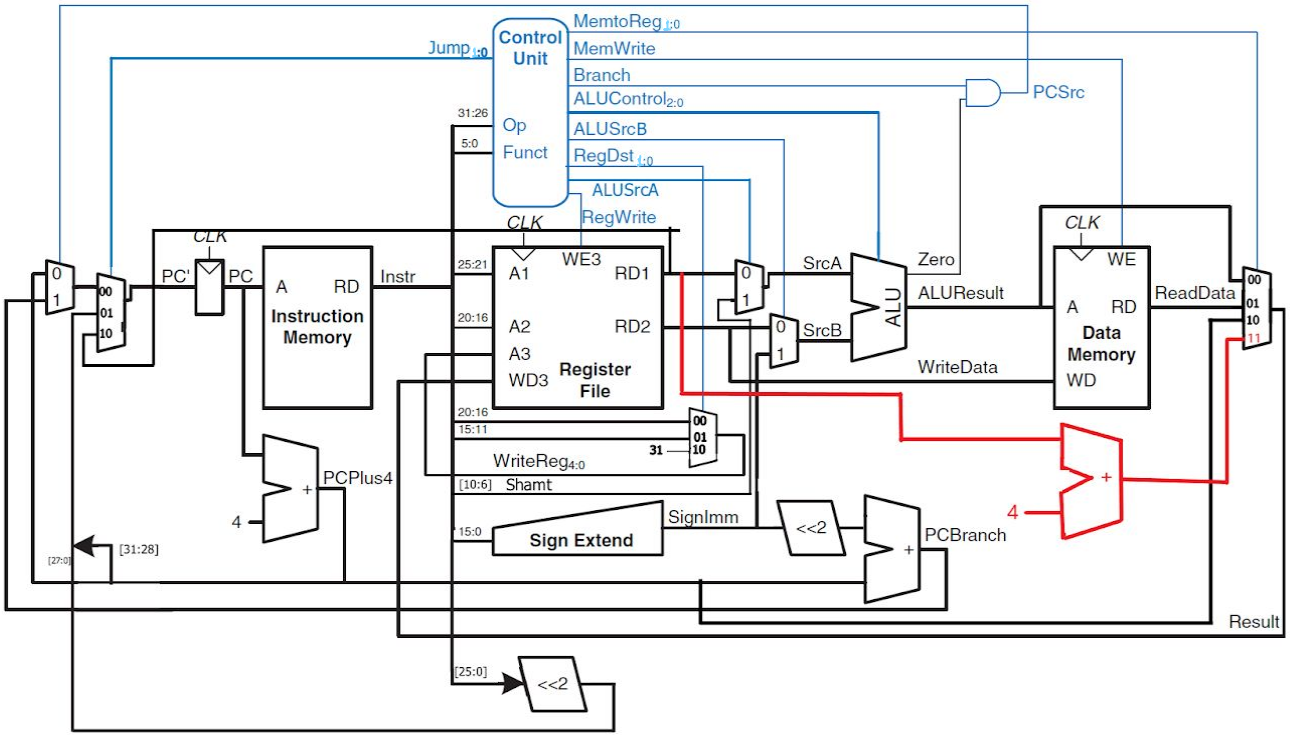
Part 2.

a) 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]. {Note: these kind of auto-increment instructions are useful when moving through an array of data words.} Example: sw+ $t0, 4($t1).

RTL Instruction for sw+:

1. IM[pc]
2. DM[RF[rs] + signExtend[imm]] <- RF[rt]
3. RF[rs] <- RF[rs] + 4
4. pc <- pc +4

b) Datapath which supports sw+ instruction:



c) Main Decoder Table with sw+:

Instruction	Opcode	RegWrite	RegDst	ALUSrcA	ALUSrcB	Branch	MemWrite	MemToReg	ALUOp	Jump
R-type	000000	1	01	0	0	0	0	00	10	00
srl	000000	1	01	1	0	0	0	00	10	00
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
jr	000000	1	01	0	0	0	0	00	10	10
sw+	101100*	1	X	0	1	0	1	11	00	00

* I made up the opcode it is sw but more (101011 + 1 = 101100)