CS224

Lab 4

Section 1

Efe Beydoğan

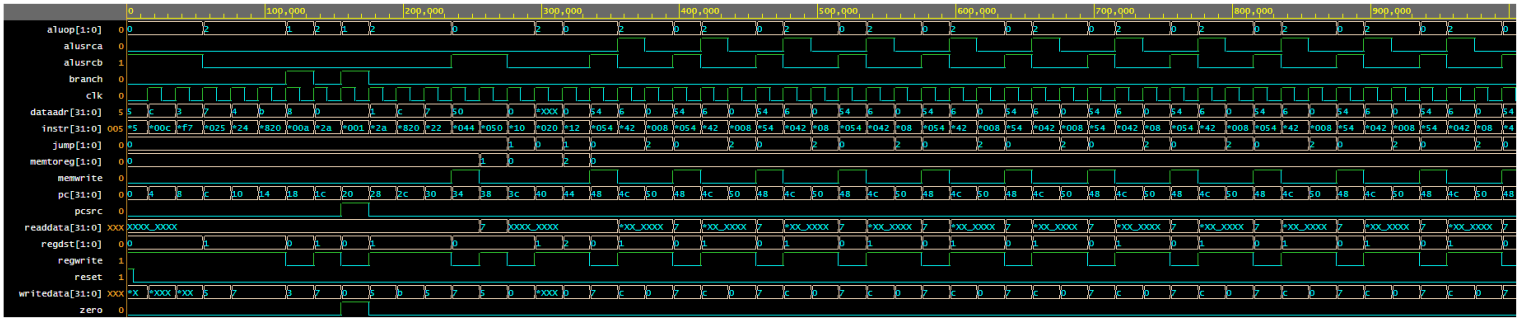
21901548

Part 1:

1.a)

Location (Hex)	Machine Instruction (Hex)	Assembly Equivalent
00	20020005	addi $v0, $0, 0x5
04	2003000c	addi $v1, $0, 0xc
08	2067fff7	addi $a3, $v1, 0Xfff7
0c	00e22025	or $a0, $a3, $v0
10	00642824	and $a1, $v1, $a0
14	00a42820	add $a1, $a1, $a0
18	10a7000a	beq $a1, $a3, 0x44
1c	0064202a	slt $a0, $v1, $a0
20	10800001	beq $a0, $0, 0x28
24	20050000	addi $a1, $0, 0x0000
28	00e2202a	slt $a0, $a3, $v0
2c	00853820	add $a3, $a0, $a1
30	00e23822	sub $a3, $a3, $v0
34	ac670044	sw $a3, 68($v1)
38	8c020050	lw $v0, 80($0)
3c	08000010	j 0x40
40	001f6020	add $t4, $0, $ra
44	0c000012	jal 0x48
48	ac020054	sw $v0, 84($0)
4c	00039042	srl $s2, $v1, 1
50	03E00008	jr $ra

1.e)



1.f)

i) In an R-type instruction, WriteData corresponds to RF[rt] but it's irrelevant for an R-type instruction because nothing is written in the data memory.

ii) Since the registers in the register file aren't initialized to any value by default, for the first 3 instructions when the registers are read, their values will be undefined. So writedata is undefined.

iii) ReadData is mostly undefined because the values in data memory are initially undefined, yet in every instruction an address of data memory is read even if it will be unused. So readdata becomes undefined until the first lw instruction, because in the previous sw instruction a value is stored into the 80th address of the data memory and the lw instruction reads the 80th address of the data memory as well.

iv) In an R-type instruction, dataadr corresponds to the value that comes of the ALU (ALUresult) which will be written in R[rd].

v) The instruction during which dataaddress is undefined corresponds to "add $t4, $0, $ra". During this instruction, the address "0" yields the result 0, however $ra is uninitialized, therefore the addition which corresponds to dataaddress becomes undefined as well.

1.g)

i) srlv operation is an R-type operation with funct field 000110. In order to add this into our implementation, we must make a new case for this function in the aludec module with a new alucontrol. The implementation of the single cycle processor already selects RF[rs] and RF[rt] as inputs coming to the ALU in R-type operations. The result of the ALU will also be written into RF[rd] when the shift operation is done, so if we implement the shifting operation in the alu module, it will work.

ii) sll is an R-type operation so if we add another case in the aludec module for its funct field and create a new alucontrol code and implement the sll operation in the alu module by creating a new case, it will work. Also, in the controller module we have to set alusrca to 1 if the funct for sll is entered.

Part 2:

2.a)

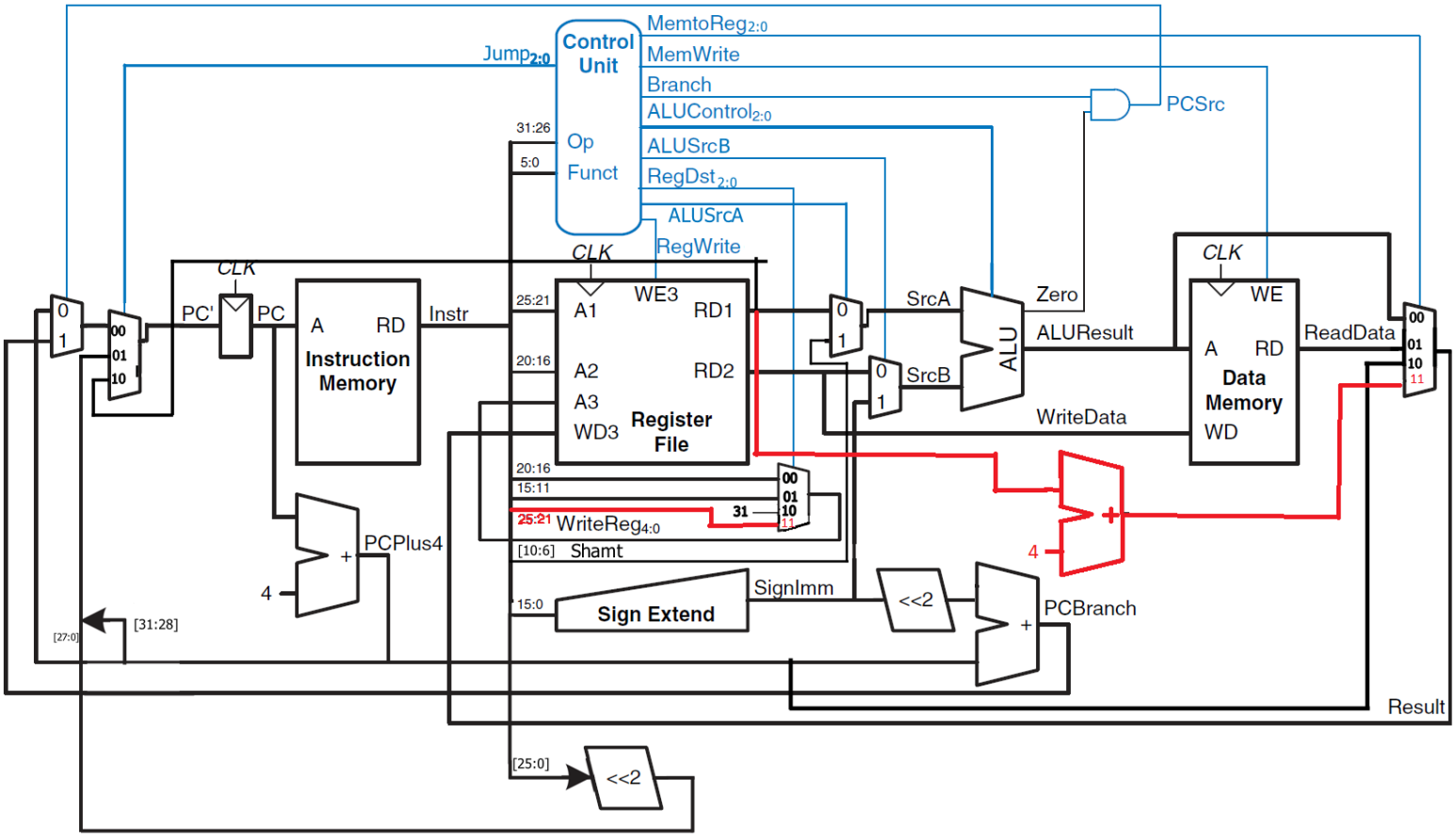
IM[PC]

DM[RF[rs] + SignExt(imm)] ← RF[rt]

RF[rs] ← RF[rs] + 4

PC ← PC + 4

2.b)



2.c)

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+	111011	1	11	0	1	0	1	11	00	00