CS 224

Lab 04

Section 03

Emre Karataş

22001641

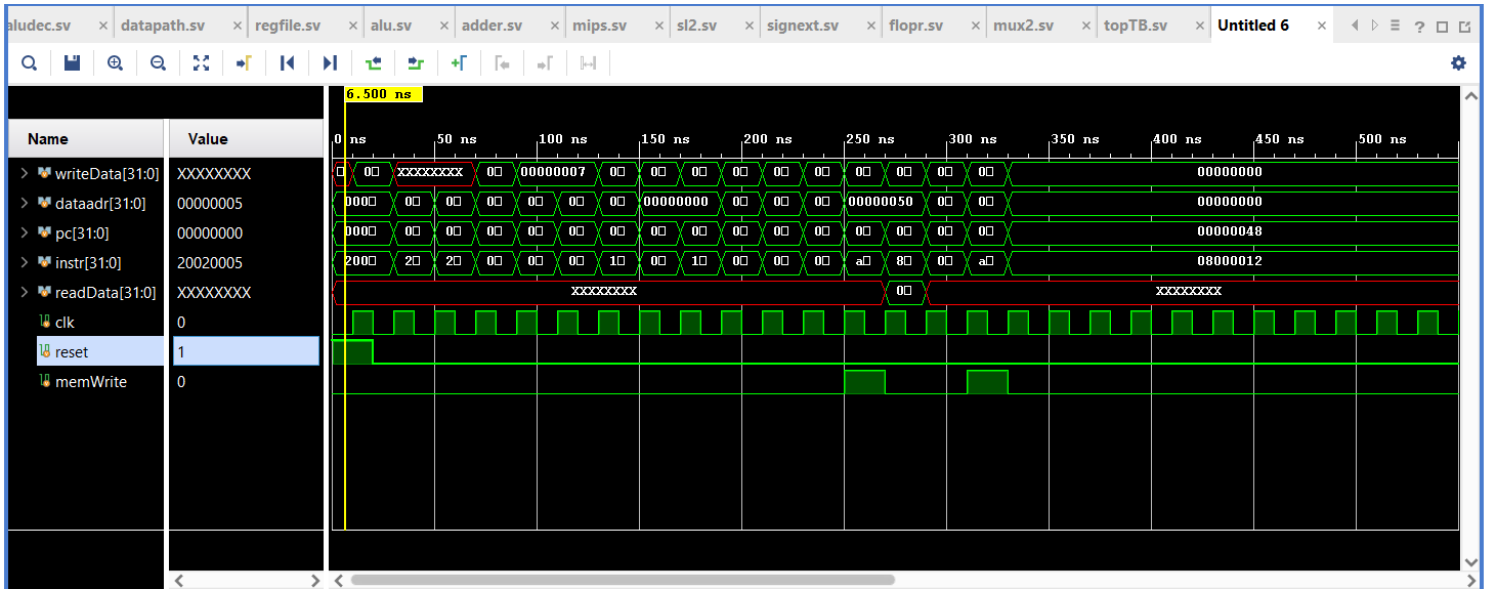
17.11.2022

PRELIMINARY REPORT

QUESTION 1.A:

Location (hex)	Machine Instruction (hex)	Assembly Language Equivalent
00	20020005	addi $v0, $0, 0x0005
04	2003000c	addi $v1, $0, 0x000c
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, $a1, $a0
1C	0064202a	slt $a0, $v1, $a0
20	10800001	beq $a0, $0, 0x0001
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, 0x0044, $v1
38	8c020050	lw $v0, 0x0050, $0
3C	08000011	j 0x00000011
40	20020001	addi $v0, $0, 0x0001
44	ac020054	sw $v0, 0x0054, $0
48	08000012	j 0x00000012

QUESTION 1.D:



QUESTION 1.E:

1. In R-type instructions, writedata corresponds rt value coming from register RD2. Note that for R type instructions, aluSrc is always 0.
2. Writedata is used for sw. Therefore it is undefined.
3. Since for the R type instructions, ALUResult bypasses Data memory since there is nothing to write data.
4. It corresponds to aluOut (aluresult).
5. For SW, memwrite becomes 1.

QUESTION 1.F:

```
module alu(input  logic [31:0] a, b,
           input  logic [2:0]  alucont,
           output logic [31:0] result,
           output logic zero);


  always_comb
    case(alucont)
      3'b010: result = a + b;
      3'b110: result = a - b;
      3'b000: result = a & b;
```

```
        3'b001: result = a | b;
         3'b011: result = a << b; // shift left operation
        3'b111: result = (a < b) ? 1 : 0;
        default: result = {32{1'bx}};
      endcase

  assign zero = (result == 0) ? 1'b1 : 1'b0;
endmodule
```

PART 2:

QUESTION 2.A:

Jm:

IM[PC]

PC <- DM[RF[rs]+SignExt(immed)]

Subi:

IM[PC]

RF[rt] <- RF[rs] - SignExt(immed)

PC <- PC + 4

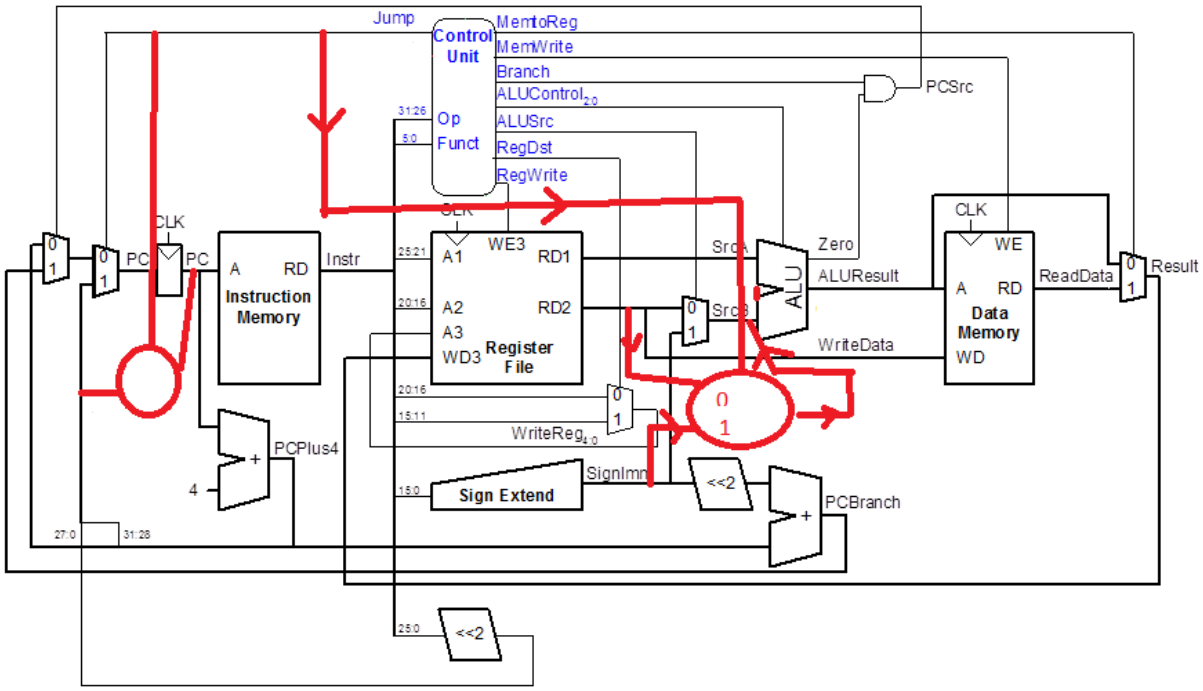
QUESTION 2.B:

Subi:

There is no need to change the hardware, since in subi we do subtraction instead of addition. Everything will be same except ALUControl [2:0], which is set to 010. It should be changed to 110 to perform substraction.

Jm:

There should be new 2 muxes for this instruction. The first one should select whether signExt(immed) value or rt. Since this operation is a jump operation, jump should be 1 so we can select signExt(immed). We make an addition to add rs and signExt value, ALUcontrol should be 010. There should be another mux, to assign result value to pc. That mux will be controlled by jump again, since this operation is a jump operation. In the coming page, new datapath will be seen. Jm instructions are labeled with red color.



QUESTION 2.C:

Instruction	Opcode	RegWrite	RegDst	ALUSrc	Branch	MemWrite	MemToReg	ALUOp	Jump
R-type	000000	1	1	0	0	0	0	10	0
lw	100011	1	0	1	0	0	1	00	0
sw	101011	0	X	1	0	1	X	00	0
beq	000100	0	X	0	1	0	X	01	0
addi	001000	1	0	1	0	0	0	00	0
j	000010	0	X	X	X	0	X	XX	1
jm	001111*	1	X	1	X	0	1	00	1
subi	001011*	1	0	1	0	0	0	01	0

***Note that opcode values are assigned randomly. They are not conflicting with other opcodes.**

Also note that this main decoder table is not conflicting with ALU decoder, therefore there is no need for changing the ALU decoder.