CS 224

Section.: 2

Spring 2019

Lab No.: 4

Zeynep Cankara/ 21703381

Question 1:

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 2:

nop:

```
Im[PC]
PC ← PC + 4
```

subi:

```
Im[PC]
RF[rt] ← RF[rs] - SignExt(immed)
PC ← PC + 4
```

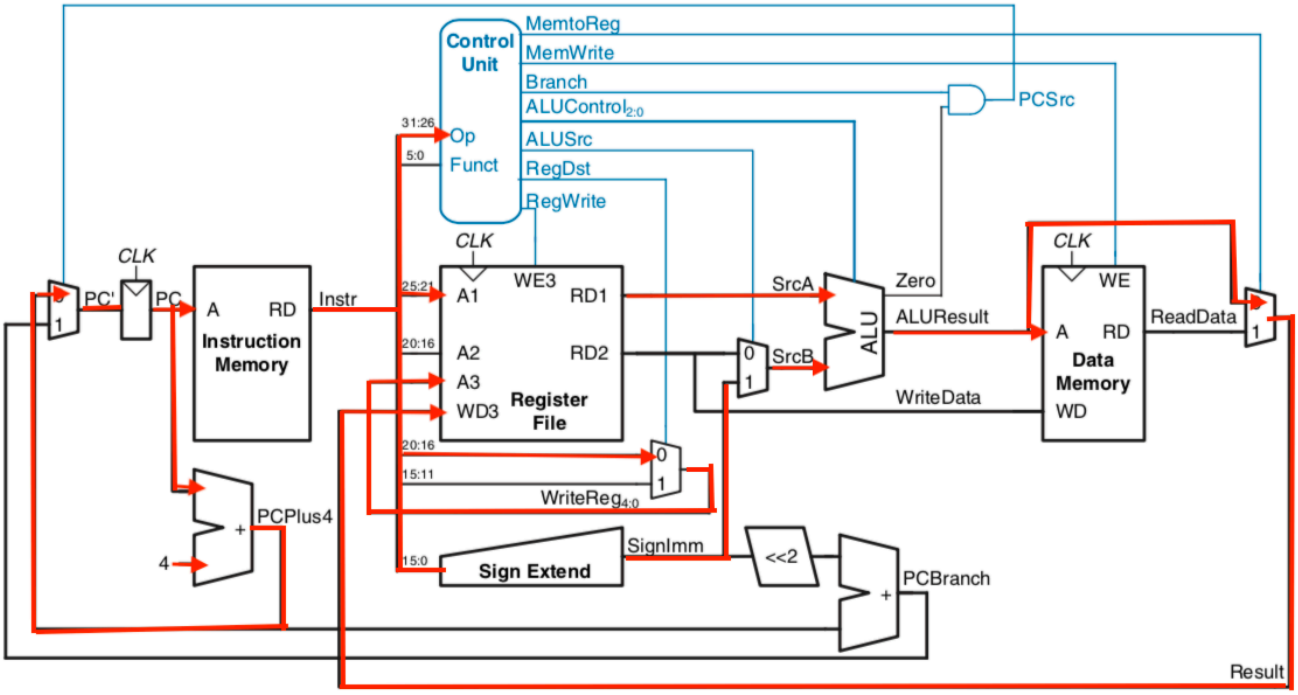
Question 3:

nop:

- No additional hardware changes done to datapath. In order to prevent changes made to the register file and memory; control signal must set. Thus, the I-type instruction addi $0, 0 being executed.

subi:

- Everything same as performing an addi instruction except control unit should perform a subtraction instead of addition. It fallows, $ALUControl_{2:0}$ should set to 110. Furthermore, no additional hardware changes done to datapath because of the reasons explained above.



Question 4:

Instruction	$Op_{5:0}$	RegWrite	RegDst	ALUSrc	Branch	MemWrite	MemToReg	$AluOp_{1:0}$	Jump
nop	001000	1	0	1	0	0	0	00	0
subi	001110	1	0	1	0	0	0	01	0

Implementation notes for Main Decoder :

ALUDecoder table same with MIPS-lite no change being done. Both nop and subi fallows the addi instruction's path in the datapath. Nop shares the same opcode with addi instruction since in order to perform nop, control signals must set by executing an addi instruction. Subi instruction requires a different opcode since ALUControl signal will used for subtraction operation. I decided to choose opcode $e_{(hex)}$ since it does not overlap with other instruction's opcodes in the core instruction set.

Question 5:

```
# addi test
addi $v1, $0, 3 # $v1(3)
# and test
and $a1, $v1, $a0 # $a1(0)
# or test
or $a0, $a1, $v1 # $a0(3)
# beq test
beq $a1, $a3, 0x0003 # jump 4 instructions forward
# slt test
slt $a2, $a1, $a0 # $a2(1) since a1(0) < a0(3)
# sw test
sw $a0, 0X10010044($0) # memory address 0X10010044 contains 3
# lw test
lw $a3, 0X10010044($0) # $a3(3)
# sub test
sub $a3, $a0, $a2 # $a3(2)
# add test
sub $a2, $a2, $a1 # $a2(1)
# subi test
subi $a3, $a0, 0x0010 # $a3(1)
# nop test
nop
# j test
j 0x00400000 # goto address 0x00400000
```

Question 6:

- Main decoder module should modified to support subi control signal.
- nop will compiled to 0x20000000. Machine code: 001000 00000 00000 0000 0000 0000 0000

```
module maindec (input logic[5:0] op,
                output logic memtoreg, memwrite, branch,
                output logic alusrc, regdst, regwrite, jump,
                output logic[1:0] aluop );
  logic [8:0] controls;

  assign {regwrite, regdst, alusrc, branch, memwrite,
                memtoreg,  aluop, jump} = controls;

  always_comb
    case(op)
      6'b000000: controls <= 9'b110000100; // R-type
      6'b100011: controls <= 9'b101001000; // LW
      6'b101011: controls <= 9'b001010000; // SW
      6'b000100: controls <= 9'b000100010; // BEQ
      6'b001000: controls <= 9'b101000000; // ADDI
      6'b000010: controls <= 9'b000000001; // J
      6'b001110: controls <= 9'b101000010; // SUBI (new)
      default:   controls <= 9'bxxxxxxxxx; // illegal op
    endcase
endmodule
```