CS 224
Lab - 4
Section: 1
Yusuf Güllüce
22202808

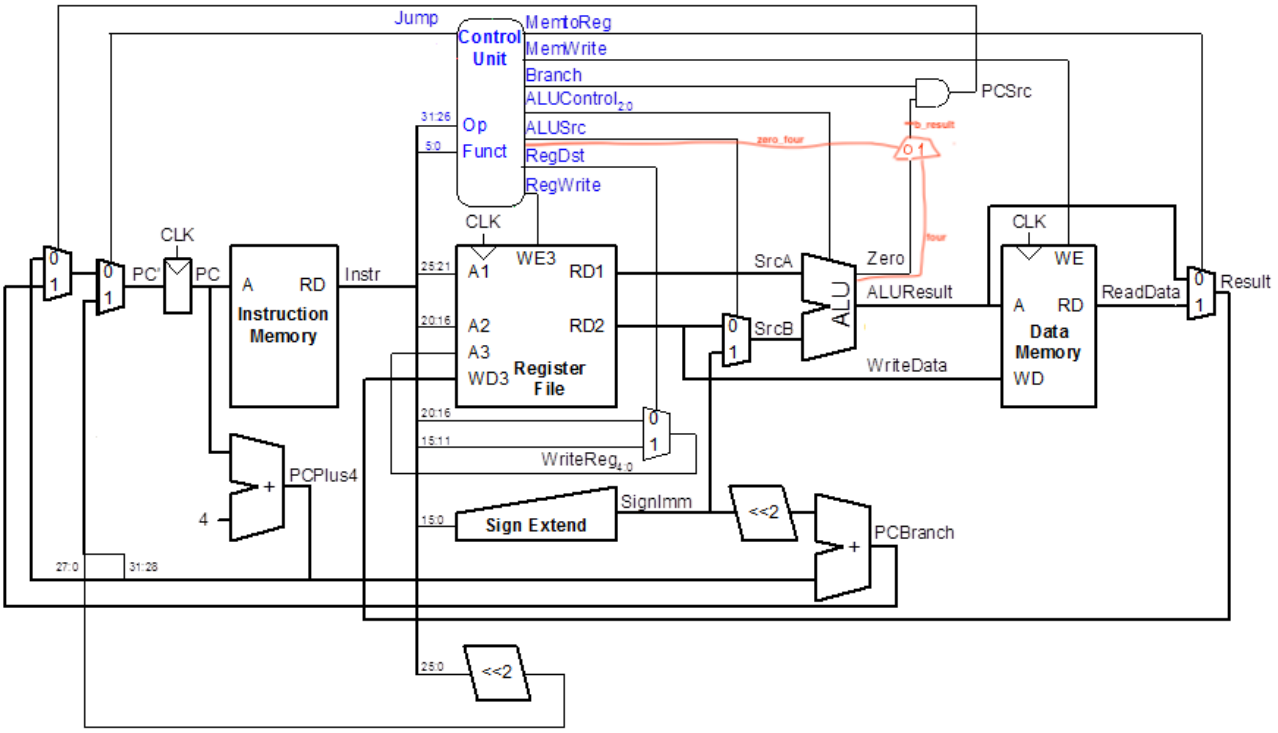
Location (hex)	Machine Instruction (hex)	Assembly Language
00	20020005	addi $v0, $zero, 5
04	2003000C	addi $v1, $zero, 12
08	2067FFF7	addi $a3, $v1, -9
0C	00E22025	or $a0, $a3, $v0
10	00642824	and $a1, $v1, $a0
14	00A42820	add $a1, $a1, $a0
18	10A7000A	beq $a1, $a3, 10
1C	0064202A	slt $a0, $v1, $a0
20	10800001	beq $a0, $zero, 1
24	20050000	addi $a1, $zero, 0
28	00E2202A	slt $a0, $a3, $v0
2C	00852820	add $a1, $a0, $a1
30	00E23822	sub $a3, $3, $v0
34	AC670044	sw $a3, 68($v1)
38	8C020050	lw $v0, 80($zero)
3C	08000011	j 17
40	20020001	addi $v0, $zero, 1
44	AC020054	sw $v0, 84($zero)
48	08000012	j 18

for bcon:

if(register([rt] - register[rs] == 4) PC <- PC + sign-extend(I constant) * 4 else PC <- PC + 4
IR <- M[PC]

For neg:

PC <- PC + 4
ALUOutput <- -register[rs]
register[rd] <- ALUOutput
IR <- M[PC]


Jump
Control Unit
MemtoReg
MemWrite
Branch
ALUControl2:0
ALUSrc
RegDst
RegWrite
Op
Funct
31:26
5:0
PCSrc
zero_four
to_result
CLK
PC'
PC
A
RD
Instruction Memory
Instr
WE3
A1
A2
A3
WD3
RD1
RD2
Register File
25:21
20:16
SrcA
SrcB
ALU
Zero
ALUResult
WriteData
WE
Data Memory
WD
ReadData
Result
20:16
15:11
WriteReg4:0
PCPlus4
4
15:0
Sign Extend
SignImm
<<2
PCBranch
27:0
31:28
25:0
<<2

Instruction	Opcode	RegWrite	RegDst	ALUSrc	Branch	MemWrite	MemToReg	ALUOp	Jump	zero_four
bcon	000001	0	x	1	1	0	x	01	0	1
neg	000000	1	1	0	0	0	0	10	0	0
x	x	x	x	x	x	x	x	x	x	0

ALUOp	Funct	ALUControl
1x	100110 (negative)	100

```
.text

//addi  $s0, $zero, 5
8'h4c: instr = 32’h2004005;

//neg   $a0, $s0
8'h50: instr = 32'h00802826;

//addi  $a0, $zero, 8
8'h54: instr = 32’h2004008;

//addi  $a1, $zero, 12
8'h58: instr = 32’h200500c;

//bcon $s0, $a1, -7
8'h5c: instr = 32'h14a4fff9;
```

changes:
- module mips
- module controller
- module maindec
- module aludec
- module datapath
- module alu

module mips:
…
logic jump;
logic four
…
~~controller c (instr[31:26], instr[5:0], zero, memtoreg, memwrite, pcsrc, alusrc, regdst, regwrite, jump, alucontrol);~~

controller c (instr[31:26], instr[5:0], zero, four, memtoreg, memwrite, pcsrc, alusrc, regdst, regwrite, jump, alucontrol);
…

controller:
…
~~input logic zero,~~
input logic zero, four,
…
logic branch;
logic zero_four;

~~maindec md (op, memtoreg, memwrite, branch, alusrc, regdst, regwrite, jump, aluop);~~
maindec md (op, memtoreg, memwrite, branch, alusrc, regdst, regwrite, jump, zero_four, aluop);
…
~~assign pcsrc = branch & zero;~~
assign pcsrc = branch & (zero_four ? zero : four);
…

module maindec:
…
output logic alusrc, regdst, regwrite, jump,
output logic zero_four,
output logic[1:0] aluop);

~~logic [8:0] controls~~
logic [9:0] controls

~~assign {regwrite, regdst, alusrc, branch, memwrite, memtoreg, aluop, jump} = controls;~~
assign {regwrite, regdst, alusrc, branch, memwrite, memtoreg, aluop, jump, zero_four} = controls;
…
~~6'bxxxxxx: controls <= 9'bxxxxxxxxx~~
6'bxxxxxx: controls <= 10'bxxxxxxxxx0

6'b000101: controls <= 10'b0011000101
…

module aludec:
…

6'b101010: alucontrol = 3'b111; // SLT
6’b100110: alucontrol = 3’b100 // making it negative

…

datapath:
…
~~output logic zero,~~
output logic zero, four,
…
~~alu alu (srca, srcb, alucontrol, aluout, zero);~~
alu alu (srca, srcb, alucontrol, aluout, zero, four);
…

alu:
…
~~output logic zero~~
output logic zero, four
…
3'b111: result = (a < b) ? 1 : 0;
3'b100: result = -a;
…
assign zero = (result == 0) ? 1'b1 : 1'b0;
assign four = (result == 4) ? 1'b1 : 1’b0;
…