CS 224

Fall 2023–2024

Lab 4 Preliminary Report

Hüseyin Uzun

21702559

Section 3

14.11.2023

Part 1-b)

Location	Machine Instruction	
8'h00	32'h20020005	ADDI $v0, $zero, 5
8'h04	32'h2003000c	ADDI $v1, $zero, 12
8'h08	32'h2067fff7	ADDI $a3, $v1, 0xFFF7
8'h0c	32'h00e22025	OR $a0, $a3, $v0
8'h10	32'h00642824	AND $a1, $v1, $a0
8'h14	32'h00a42820	ADD $a1, $a1, $a0
8'h18	32'h10a7000a	BEQ $a1, $a3, 40
8'h1c	32'h0064202a	SLT $a0, $v1, $a0
8'h20	32'h10800001	BEQ $a0, $zero, 4
8'h24	32'h20050000	ADDI $a1, $zero, 0
8'h28	32'h00e2202a	SLT $a0, $a3, $v0
8'h2c	32'h00853820	ADD $a3, $a0, $a1
8'h30	32'h00e23822	SUB $a3, $a3, $v0
8'h34	32'hac670044	SW $a3, 44($v1)
8'h38	32'h8c020050	LW $v0, 50($zero)
8'h3c	32'h08000011	J 44
8'h40	32'h20020001	ADDI $v0, $zero, 1
8'h44	32'hac020054	SW $v0, 54($zero)
8'h48	32'h08000012	J 48

Part 1-c)

beq:

IM[PC]

if(R[rs]>=R[rt])

PC = PC + 4 + BranchAddr

else

PC = PC + 4

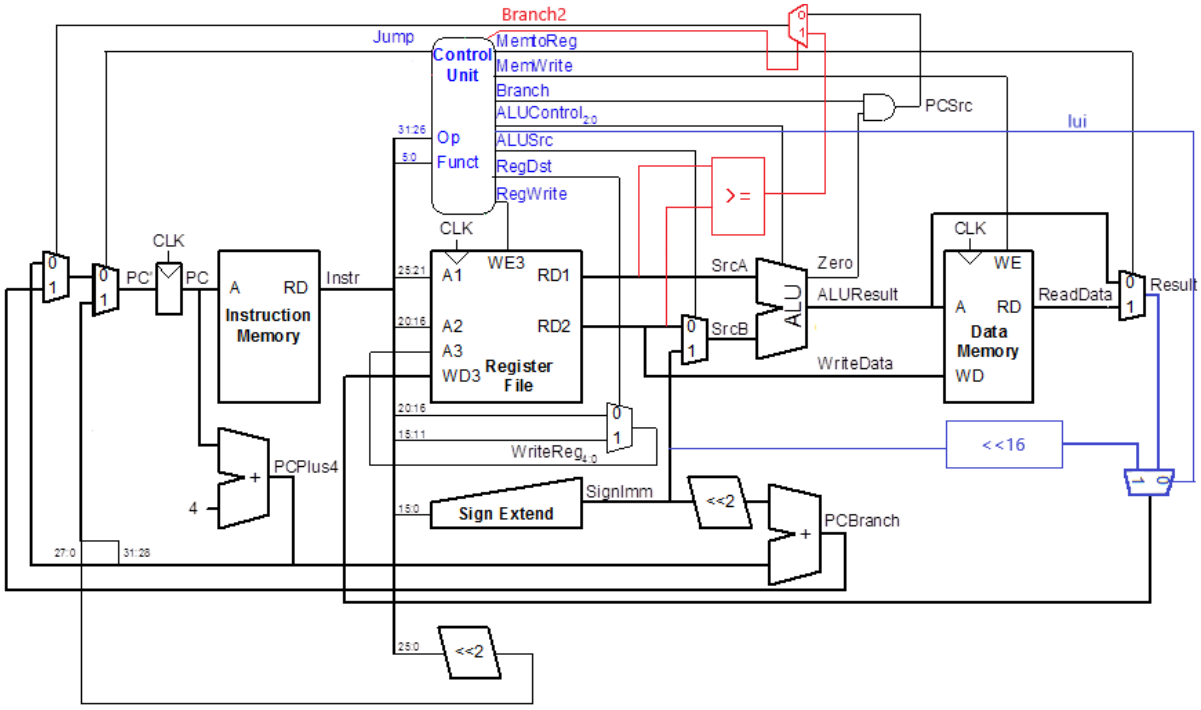
lui:

IM[PC]

R[rt] = {imm, 16'b0}

PC = PC + 4

Part 1-d)



Part 1-e)

Instruction	Op5:0	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUOp1:0	Branch2	lui
R-type	000000	1	1	0	0	0	0	10	0	0
lw	100011	1	0	1	0	0	1	00	0	0
sw	101011	0	X	1	0	1	X	00	0	0
beq	000100	0	X	0	1	0	X	01	0	0
bge	011110	1	X	0	X	0	0	XX	1	0
lui	001111	1	0	0	0	0	1	XX	0	1

Part 1-f)

addi $v0, $zero, 5

addi $s2, $zero, 2

slt $s2, $s2, $v0

addi $v1, $zero, 12

addi $a3, $v1, 1624

or $a0, $a3, $v0

and $a1, $v1, $a0

bge $a1, $a3, 48

lui $s3, 1024

sw $a3, 8($v1)

lw $v0, 8($zero)

sub $a1, $a3, $s2

beq $a1, $a3, 0

j 32

Part 1-g)

module imem

module mips

```
 logic        memtoreg, pcsrc, zero, alusrc, regdst, regwrite, jump, bge, lui;

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

 datapath dp (clk, reset, memtoreg, pcsrc, alusrc, regdst, regwrite, jump,
                      alucontrol, zero, pc, instr, aluout, writedata, readdata, bge, lui);

module controller(input  logic[5:0] op, funct,
                  input  logic     zero,
                  output logic     memtoreg, memwrite,
                  output logic     pcsrc, alusrc,
                  output logic     regdst, regwrite,
                  output logic     jump,
                  output logic[2:0] alucontrol
                     output logic bge, lui);
   maindec md (op, memtoreg, memwrite, branch, alusrc, regdst, regwrite,
                       jump, aluop, bge, lui);
```

```
module maindec (input logic[5:0] op,
                output logic memtoreg, memwrite, branch,
                output logic alusrc, regdst, regwrite, jump,
                output logic[1:0] aluop
                  output logic bge, lui);
  assign {regwrite, regdst, alusrc, branch, memwrite,
          memtoreg,  aluop, jump, bge, lui} = controls;
 always_comb
   case(op)
     6'b000000: controls <= 11'b11000010000; // R-type
     6'b100011: controls <= 11'b10100100000; // LW
     6'b101011: controls <= 11'b00101000000; // SW
     6'b000100: controls <= 11'b00010001000; // BEQ
     6'b001000: controls <= 11'b10100000000; // ADDI
     6'b000010: controls <= 11'b00000000100; // J
     6'b011110: controls <= 11'b10000000010; // BGE
     6'b001111: controls <= 11'b10000100001; // LUI
     default:   controls <= 11'bxxxxxxxxxxx; // illegal op
   endcase
endmodule

module datapath (input  logic clk, reset, memtoreg, pcsrc, alusrc, regdst,
           input  logic regwrite, jump,
                input  logic[2:0]  alucontrol,
           output logic zero,
                output logic[31:0] pc,
              input  logic[31:0] instr,
           output logic[31:0] aluout, writedata,
              input  logic[31:0] readdata
              input logic bge,lui);
```

```
  logic bgeout;
  // next PC logic
  bge #(32) bge_logic( srca, srcb, bgeout );
  mux2 #(32) branch2mux( pcsrc, bgeout, branch2, pcsrc2 );
module bge #(parameter WIDTH = 32)
                (input logic[WIDTH-1:0] c1, c2,
                 output logic bge);

    assign bge = (c1 >= c2);
endmodule
```