IHSAN DOGRAMACI BILKENT UNIVERSITY

CS 224

Computer Organization

Preliminary Design Report

Lab-4

Kerem Ayöz

Section 03 / 21501569

14/11/2017

Part b

Memory Location	Machine Instruction	Assembly Equivalent
0x00000000	0x20020005	addi $v0 $zero 0x0005
0x00000004	0x2003000C	addi $v1 $zero 0x000C
0x00000008	0x2067FFF7	addi $a3 $v1 0xFFF7
0x0000000C	0x00E22025	or $a0 $a3 $v0
0x00000010	0x00642824	and $a1 $v1 $a0
0x00000014	0x00A42820	add $a1 $a1 $a0
0x00000018	0x10A7000A	beq $a1 $a3 0x000A
0x0000001C	0x0064202A	slt $a0 $v1 $a0
0x00000020	0x10800001	beq $a0 $zero 0x0001
0x00000024	0x20050000	addi $a1 $zero 0x0000
0x00000028	0x00E2202A	slt $a0 $a3 $v0
0x0000002C	0x00853820	add $a3 $a0 $a1
0x00000030	0x00E23822	sub $a3 $a3 $v0
0x00000034	0xAC670044	sw $a3 0x0044 $v1
0x00000038	0x8C020050	lw $v0 0x0050 $zero
0x0000003C	0x08000011	j 0x0000011
0x00000040	0x20020001	addi $v0 $zero 0x0001
0x00000044	0xAC020054	sw $v0 0x0054 $zero
0x00000048	0x08000012	j 0x0000012

Part c

- ## jalm

  IM[PC]

  RF[rt] ← PC + 4

  PC ← RF[rs] + SignExt(immed)

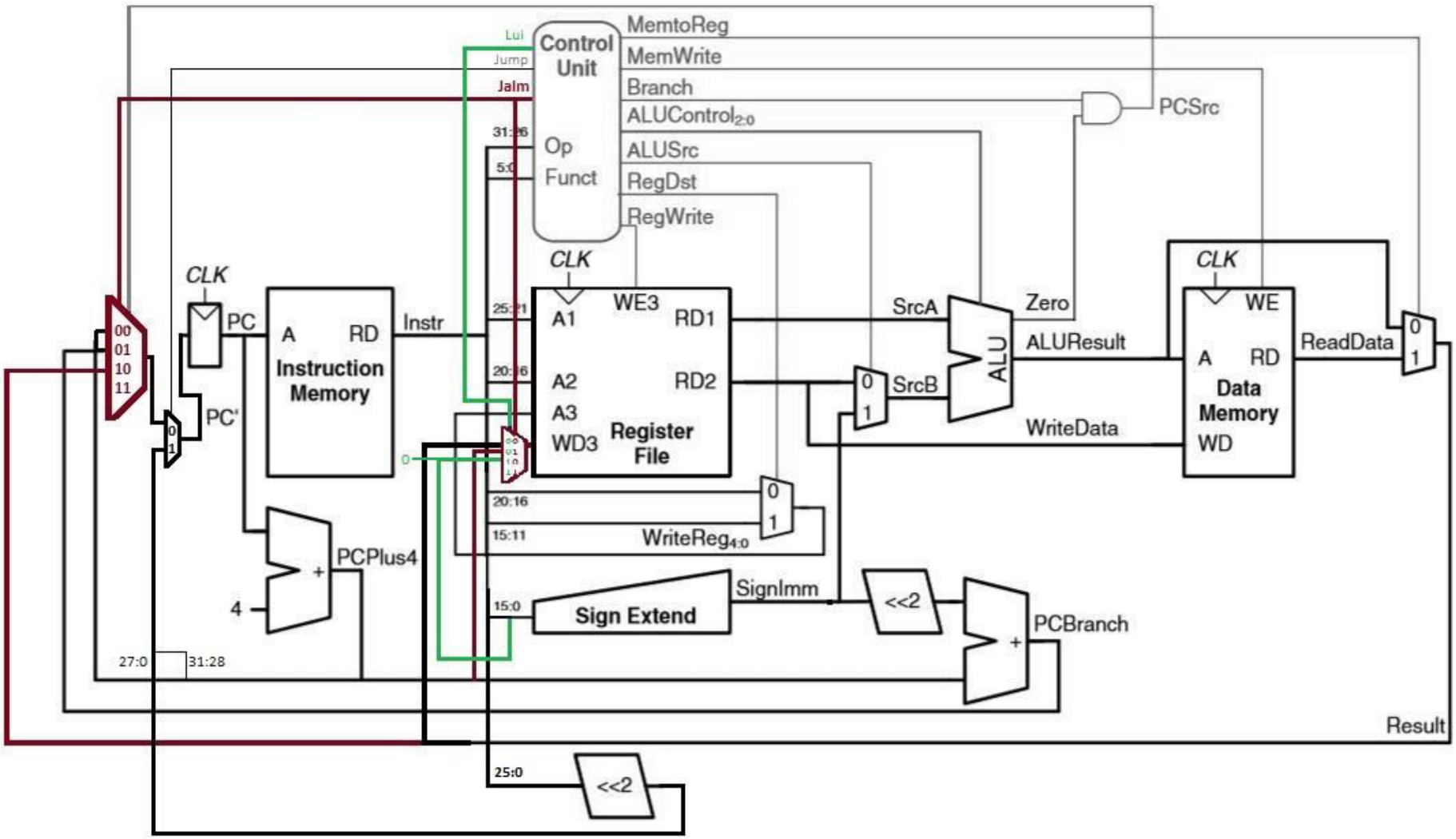
- ## lui

  IM[PC]

  RF[rt] ← {immed, 16'b{0}}

  PC ← PC + 4

Part d


Lui
Jump
Jalm
Control Unit
Op
Funct
MemtoReg
MemWrite
Branch
ALUControl2:0
ALUSrc
RegDst
RegWrite
PCSrc
CLK
PC
PC'
A
RD
Instruction Memory
Instr
WE3
A1
A2
A3
WD3
RD1
RD2
Register File
SrcA
SrcB
ALU
Zero
ALUResult
WriteData
WE
Data Memory
WD
ReadData
WriteReg4:0
PCPlus4
4
Sign Extend
SignImm
<<2
PCBranch
Result
00
01
10
11
31:26
5:0
25:21
20:16
15:11
15:0
27:0
31:28
25:0

Part e

Instruction	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	$ALUOp_{1:0}$	Jump	Jalm	Lui
R-type	000000	1	1	0	0	0	0	10	0	0	0
lw	100011	1	0	1	0	0	1	00	0	0	0
sw	101011	0	X	1	0	1	X	00	0	0	0
beq	000100	0	X	0	1	0	X	01	0	0	0
j	000010	0	X	X	X	0	X	XX	1	0	0
addi	001000	1	0	1	0	0	0	00	0	0	0
jalm	000011	1	0	1	0	0	1	00	0	1	0
lui	001111	1	0	X	0	0	0	XX	0	0	1

Part f

Simple MIPS test program

```
addi    $t1, $t1, 1
add     $t2, $t2, $t1
beq     $t0, $t1, 0x0001
j       0x2536000
jalm    $t1, 60($s3)
addi    $v0, $t2, $s4
and     $v0, $v0, $t1
addi    $a1 $zero 0x0000
slt     $a0 $a3 $v0
add     $a3 $a0 $a1
sub     $a3 $a3 $v0
lui     $v1, 0xABCD
ori     $v1, $v1, 0xEF00
```

Part g

Modules Need To Be Changed

Mips, controller, maindec, datapath, Mux41(newly added)

mips

```
module mips (input  logic       clk, reset,
             output logic[31:0]  pc,
             input  logic[31:0]  instr,
             output logic       memwrite,
             output logic[31:0]  aluout, writedata,
             input  logic[31:0]  readdata);

 logic       memtoreg, pcsrc, zero, alusrc, regdst, regwrite, jump, jalmSignal, luiSignal;
 logic [2:0]  alucontrol;

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

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

endmodule
```

controller

```
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     jalmSignal, luiSignal,
                  output logic[2:0] alucontrol);

  logic [1:0] aluop;
  logic       branch;

  maindec md (op, memtoreg, memwrite, branch, alusrc, regdst, regwrite,
                  jump, jalmSignal, luiSignal, aluop);

  aludec  ad (funct, aluop, alucontrol);

  assign pcsrc = branch & zero;

endmodule
```

maindec

```
module maindec (input logic[5:0] op,
                output logic memtoreg, memwrite, branch,
                output logic alusrc, regdst, regwrite, jump,
                output logic jalmSignal, luiSignal,
                output logic[1:0] aluop );
  logic [10:0] controls;

  assign {regwrite, regdst, alusrc, branch, memwrite,
          memtoreg,  aluop, jump, jalmSignal, luiSignal} = 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'b000011: controls <= 11'b10101100010; // Jalm
      6'b001111: controls <= 11'b10X000XX001; // Lui
      default:   controls <= 11'bxxxxxxxxxxx; // illegal op
    endcase
endmodule
```

datapath

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

 logic [4:0]  writereg;
 logic [31:0] pcnext, pcnextbr, pcplus4, pcbranch;
 logic [31:0] signimm, signimmsh, srca, srcb, result;

 // next PC logic
 flopr #(32) pcreg(clk, reset, pcnext, pc);
 adder       pcadd1(pc, 32'b100, pcplus4);
 sl2         immsh(signimm, signimmsh);
 adder       pcadd2(pcplus4, signimmsh, pcbranch);

 logic [31:0] useless;
 Mux41 jalmMux(pcplus4, pcbranch, result, useless, {jalmSignal,pcsrc}, pcnextbr );

 mux2 #(32)  pcmux(pcnextbr, {pcplus4[31:28],
               instr[25:0], 2'b00}, jump, pcnext);

// register file logic
  logic [31:0] outputt;
```

```
  Mux41 rtMux(result, pcplus4, {instr[15:0], 16'b0}, useless, {luiSignal, jalmSignal},outputt);
  regfile     rf (clk, regwrite, instr[25:21], instr[20:16], writereg,
                  outputt, srca, writedata);

  mux2 #(5)    wrmux (instr[20:16], instr[15:11], regdst, writereg);
  mux2 #(32)  resmux (aluout, readdata, memtoreg, result);
  signext        se (instr[15:0], signimm);

 // ALU logic
  mux2 #(32)  srcbmux (writedata, signimm, alusrc, srcb);
  alu        alu (srca, srcb, alucontrol, aluout, zero);

endmodule
```

Mux41

```
//Extra Module
module Mux41(
   input logic [31:0] d0,d1,d2,d3,
   input logic [1:0] S,
   output logic [31:0] Y
   );

   assign Y = S[1] ? (S[0] ? d3 : d2)
                   : (S[0] ? d1 : d0);

endmodule
```