PRELIMINARY REPORT

CS-224

# Section No: 5

# Spring 2018

# Lab No: 5

# Elif Gülşah Kaşdoğan/ 21601183

**Asuman Aydın/21502604**

lui:

IM[PC]

RF[rt] < - *immed16* || 16b’0

PC < - PC+4

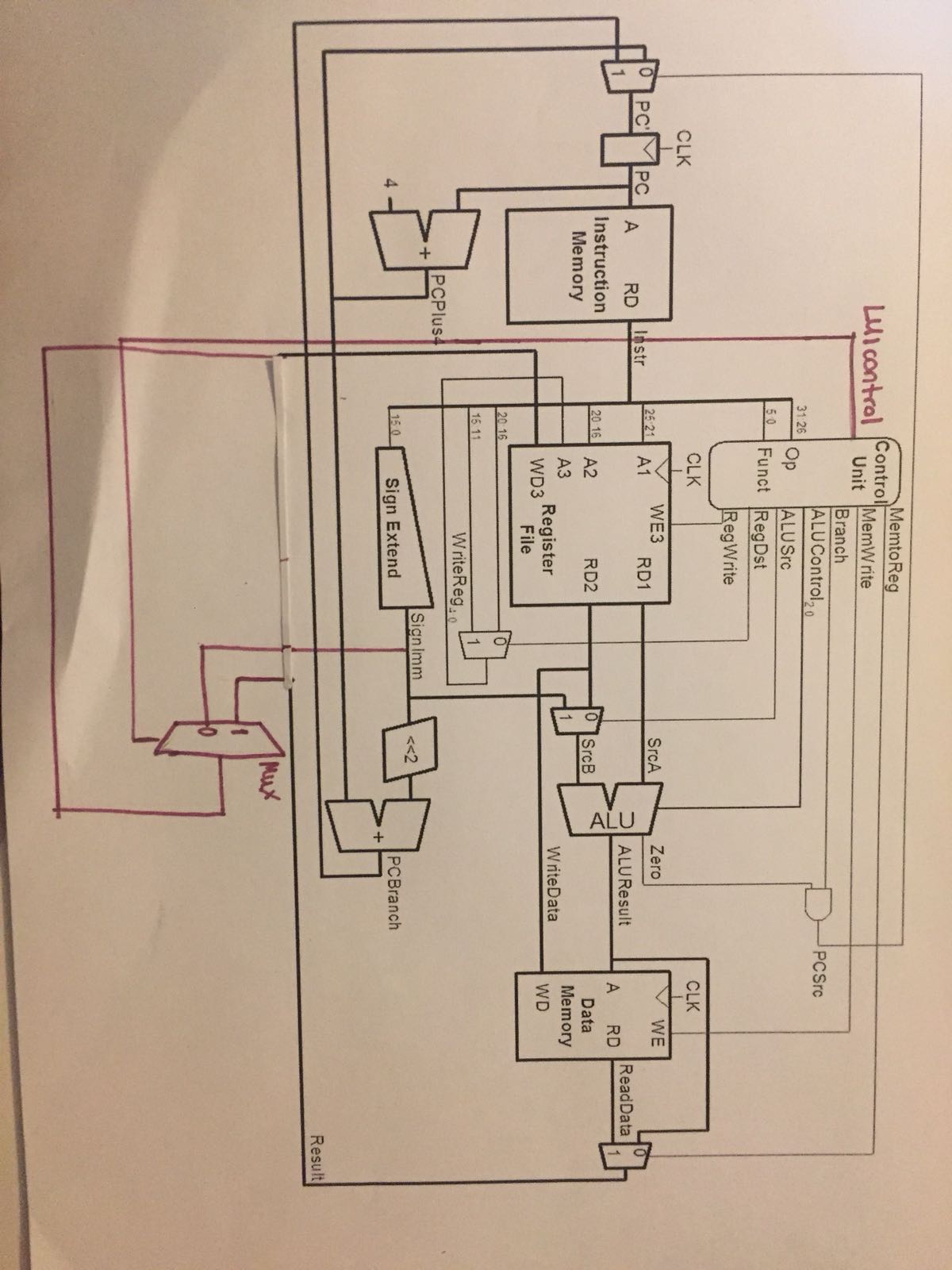
jalm:

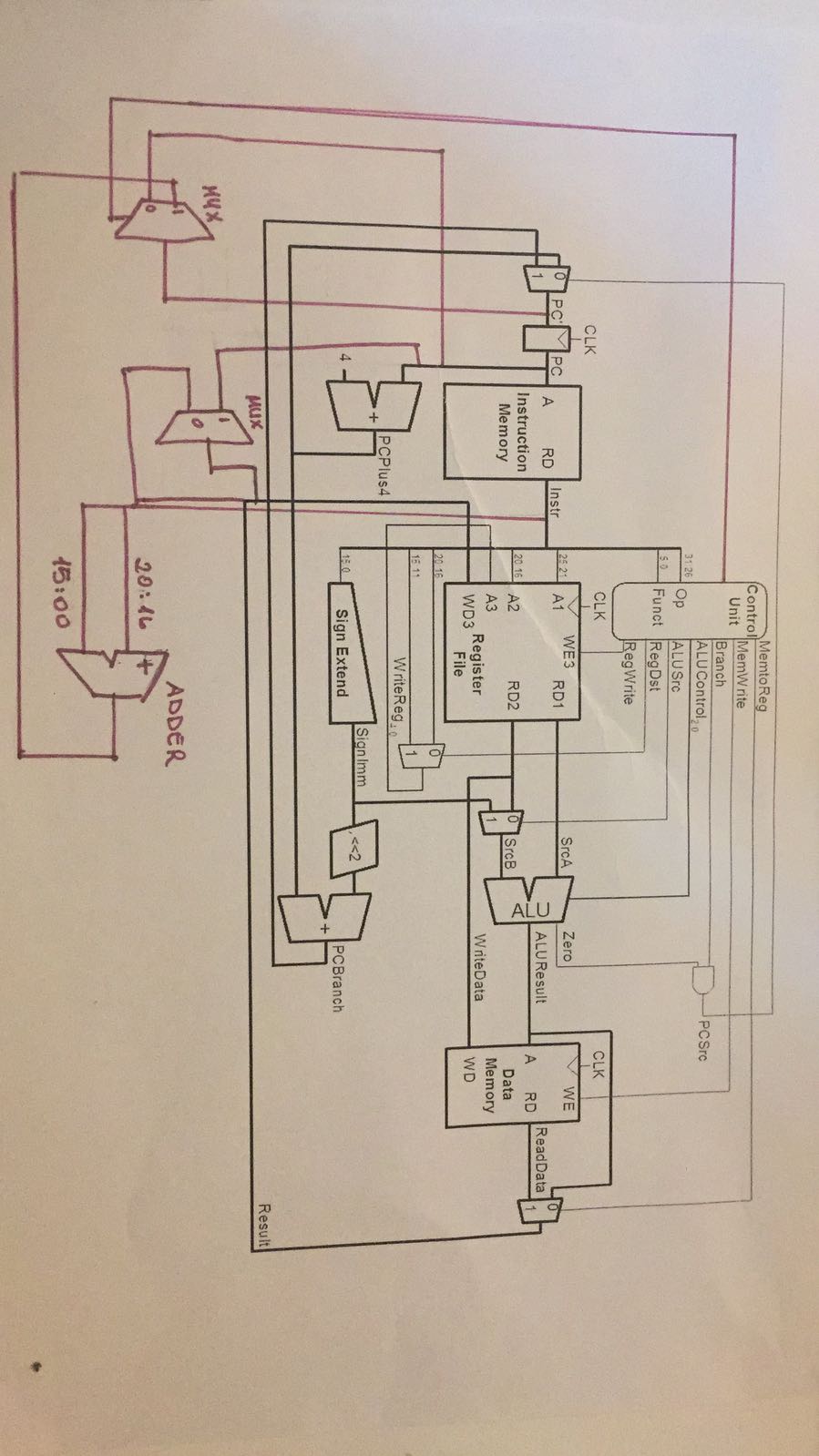
IM[PC]

RF[rt]<-*instr address*

PC<-RF[rs]+*immed*

**2.**

**** **LUI INSTRUCTION**

****

**JALM INSTRUCTION**

**3**.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | MemtoReg | MemWrite | branch | alusrc | regdst | regwrite | jum | ALUop | LUIcontrol\*\* |
| R | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 10 | 0 |
| lw | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 00 | 0 |
| sw | x | 1 | 0 | 1 | x | 0 | 0 | 00 | 0 |
| beq | x | 0 | 1 | 0 | x | 0 | 0 | 01 | 0 |
| addi | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 00 | 0 |
| j | x | 0 | 0 | x | 0 | 1 | 0 | xx | 0 |
| lui | x | 0 | 0 | x | 0 | 1 | 0 | xx | 1 |
| jalm | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 00 | 0 |

\*\*: LUIcontrol will be select signal for a new mux, for other instructions (Original10) LUIcontrol will be zero.

**4**.

module imem ( input logic [5:0] addr, output logic [31:0] instr);

// imem is modeled as a lookup table, a stored-program byte-addressable ROM

always\_comb

case ({addr,2'b00}) // word-aligned fetch

// address instruction

// ------- -----------

8'h00: instr = 32'h20020005; // disassemble, by hand

8'h04: instr = 32'h2003000c; // or with a program,

8'h08: instr = 32'h2067fff7; // to find out what

8'h0c: instr = 32'h00e22025; // this program does!

8'h10: instr = 32'h00642824;

8'h14: instr = 32'h00a42820;

8'h18: instr = 32'h10a7000a;

8'h1c: instr = 32'h0064202a;

8'h20: instr = 32'h10800001;

8'h24: instr = 32'h20050000;

Original 10instructions, our older imem used to indicate that datapath stayed unchanged and Original 10 can still be implemented without a problem

8'h28: instr = 32'h00e2202a;

8'h2c: instr = 32'h00853820;

8'h30: instr = 32'h00e23822;

8'h34: instr = 32'hac670044;

8'h38: instr = 32'h8c020050;

8'h3c: instr = 32'h08000011;

8'h40: instr = 32'h20020001;

8'h44: instr = 32'hac020054;

8'h48: instr = 32'h08000012; // j 48, so it will loop here

New instructions lui & jalm

8’h4c: instr=32’h5db30040;

8’h50: inst=32’h3c0ddead;

default: instr = {32{1'bx}}; // unknown address

endcase

endmodule

**5**.

Datapath, control unit, maindec and mips internally changed for these instructions, additional multiplexers and one adder added to datapath and used.

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 LUIcontrol,

output logic[2:0] alucontrol);

logic [1:0] aluop;

logic branch;

maindec md (op, memtoreg, memwrite, branch, alusrc, regdst, regwrite,

jump,LUIcontrol,aluop);

aludec ad (funct, aluop, alucontrol);

assign pcsrc = branch & zero;

endmodule

module maindec (input logic[5:0] op,

output logic memtoreg, memwrite, branch,

output logic alusrc, regdst, regwrite, jump,

output logic LUIcontrol,

output logic[1:0] aluop );

logic [9:0] controls;

assign {regwrite, regdst, alusrc, branch, memwrite,

memtoreg, aluop, jump, LUIcontrol} = controls;

always\_comb

case(op)

6'b000000: controls = 10'b1100001000; //Rtype

6'b100011: controls = 10'b1010010000; //LW

6'b101011: controls = 10'b0010100000; //SW

6'b000100: controls = 10'b0001000010; //BEQ

6'b001000: controls = 10'b1010000000; //ADDI

6'b000010: controls = 10'b0000001000; //J

6'b001111: controls = 10'bx00x010xx1;//LUI

default: controls = 10'bxxxxxxxxxx; //illegal op

endcase

endmodule

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, LUIcontrol;

logic [2:0] alucontrol;

controller c (instr[31:26], instr[5:0], zero, memtoreg, memwrite, pcsrc,

alusrc, regdst, regwrite, jump, LUIcontrol, alucontrol);

datapath dp (clk, reset, memtoreg, pcsrc, alusrc, regdst, regwrite, jump, LUIcontrol,

alucontrol, zero, pc, instr, aluout, writedata, readdata);

endmodule