

CMPEN 331 Final Project

By: Aly Ghallab



May 4, 2023

The Pennsylvania State University

Professor Mohammed El Mekkawy

Design Code

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer: Aly Ghallab

//

// Create Date: 05/02/2023 06:21:45 PM

// Design Name:

// Module Name: datapath

// Project Name: CMPEN 331 Final Project

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module pc (newpc1, clk, pc1);

input [31:0] newpc1;

input clk;

output reg [31:0] pc1;

always @(posedge clk)

begin

pc1 <= newpc1;

end

initial

begin

pc1 <= 32'h00000000;

end

endmodule

module pc\_adder (pc, P4);

input [31:0] pc;

output reg [31:0] P4;

always @(\*)

begin

P4 <= pc + 4;

end

endmodule

module Ctrl (op, func, z, wreg, regrt, jal, m2reg, shift, alu\_imm, sext, alu\_c, wmem, pcsrc);

input [5:0] op;

input [5:0] func;

input z;

output reg wreg, regrt, jal, m2reg, shift, alu\_imm, sext, wmem;

output reg [3:0] alu\_c;

output reg [1:0] pcsrc;

always @(\*)

begin

case(op)

6'b000000 : begin

case(func)

6'b100000 : begin//add

wreg = 1'b1;

regrt = 1'b0;

jal = 1'b0;

m2reg = 1'b0;

shift = 1'b0;

alu\_imm = 1'b0;

sext = 1'bx;

alu\_c = 4'bx000;

wmem = 1'b0;

pcsrc = 2'b00;

end

6'b100010 : begin//sub

wreg = 1'b1;

regrt = 1'b0;

jal = 1'b0;

m2reg = 1'b0;

shift = 1'b0;

alu\_imm = 1'b0;

sext = 1'bx;

alu\_c = 4'bx100;

wmem = 1'b0;

pcsrc = 2'b00;

end

6'b100100 : begin//and

wreg = 1'b1;

regrt = 1'b0;

jal = 1'b0;

m2reg = 1'b0;

shift = 1'b0;

alu\_imm = 1'b0;

sext = 1'bx;

alu\_c = 4'bx001;

wmem = 1'b0;

pcsrc = 2'b00;

end

6'b100101 : begin//or

wreg = 1'b1;

regrt = 1'b0;

jal = 1'b0;

m2reg = 1'b0;

shift = 1'b0;

alu\_imm = 1'b0;

sext = 1'bx;

alu\_c = 4'bx101;

wmem = 1'b0;

pcsrc = 2'b00;

end

6'b100110 : begin//xor

wreg = 1'b1;

regrt = 1'b0;

jal = 1'b0;

m2reg = 1'b0;

shift = 1'b0;

alu\_imm = 1'b0;

sext = 1'bx;

alu\_c = 4'bx010;

wmem = 1'b0;

pcsrc = 2'b00;

end

6'b000000 : begin//sll

wreg = 1'b1;

regrt = 1'b0;

jal = 1'b0;

m2reg = 1'b0;

shift = 1'b1;

alu\_imm = 1'b0;

sext = 1'bx;

alu\_c = 4'b0011;

wmem = 1'b0;

pcsrc = 2'b00;

end

6'b000010 : begin//srl

wreg = 1'b1;

regrt = 1'b0;

jal = 1'b0;

m2reg = 1'b0;

shift = 1'b1;

alu\_imm = 1'b0;

sext = 1'bx;

alu\_c = 4'b0111;

wmem = 1'b0;

pcsrc = 2'b00;

end

6'b000011 : begin //sra

wreg = 1'b1;

regrt = 1'b0;

jal = 1'b0;

m2reg = 1'b0;

shift = 1'b1;

alu\_imm = 1'b0;

sext = 1'bx;

alu\_c = 4'b1111;

wmem = 1'b0;

pcsrc = 2'b00;

end

6'b001000: begin //jr

wreg = 1'b0;

regrt = 1'bx;

jal = 1'bx;

m2reg = 1'bx;

shift = 1'bx;

alu\_imm = 1'bx;

sext = 1'bx;

alu\_c = 4'bxxxx;

wmem = 1'b0;

pcsrc = 2'b10;

end

endcase

end

6'b001000: begin //addi

wreg = 1'b1;

regrt = 1'b1;

jal = 1'b0;

m2reg = 1'b0;

shift = 1'b0;

alu\_imm = 1'b1;

sext = 1'b1;

alu\_c = 4'bx000;

wmem = 1'b0;

pcsrc = 2'b00;

end

6'b001100: begin //andi

wreg = 1'b1;

regrt = 1'b1;

jal = 1'b0;

m2reg = 1'b0;

shift = 1'b0;

alu\_imm = 1'b1;

sext = 1'b0;

alu\_c = 4'bx001;

wmem = 1'b0;

pcsrc = 2'b00;

end

6'b001101: begin //ori

wreg = 1'b1;

regrt = 1'b1;

jal = 1'b0;

m2reg = 1'b0;

shift = 1'b0;

alu\_imm = 1'b1;

sext = 1'b0;

alu\_c = 4'bx101;

wmem = 1'b0;

pcsrc = 2'b00;

end

6'b001110: begin //xori

wreg = 1'b1;

regrt = 1'b1;

jal = 1'b0;

m2reg = 1'b0;

shift = 1'b0;

alu\_imm = 1'b1;

sext = 1'b1;

alu\_c = 4'bx010;

wmem = 1'b0;

pcsrc = 2'b00;

end

6'b100011: begin //lw

wreg = 1'b1;

regrt = 1'b1;

jal = 1'b0;

m2reg = 1'b1;

shift = 1'b0;

alu\_imm = 1'b1;

sext = 1'b1;

alu\_c = 4'bx000;

wmem = 1'b0;

pcsrc = 2'b00;

end

6'b101011: begin //sw

wreg = 1'b0;

regrt = 1'bx;

jal = 1'bx;

m2reg = 1'bx;

shift = 1'b0;

alu\_imm = 1'b1;

sext = 1'b1;

alu\_c = 4'bx000;

wmem = 1'b1;

pcsrc = 2'b00;

end

6'b000100: begin //beq

wreg = 1'b0;

regrt = 1'bx;

jal = 1'bx;

m2reg = 1'bx;

shift = 1'b0;

alu\_imm = 1'b0;

sext = 1'b1;

alu\_c = 4'bx010;

wmem = 1'b0;

pcsrc = 2'b00;

if (z == 1)

begin

wreg = 1'b0;

regrt = 1'bx;

jal = 1'bx;

m2reg = 1'bx;

shift = 1'b0;

alu\_imm = 1'b0;

sext = 1'b1;

alu\_c = 4'bx010;

wmem = 1'b0;

pcsrc = 2'b01;

end

end

6'b000101: begin //bne

if (z == 0) begin

wreg = 1'b0;

regrt = 1'bx;

jal = 1'bx;

m2reg = 1'bx;

shift = 1'b0;

alu\_imm = 1'b0;

sext = 1'b1;

alu\_c = 4'bx010;

wmem = 1'b0;

pcsrc = 2'b01;

end

if (z == 1) begin

wreg = 1'b0;

regrt = 1'bx;

jal = 1'bx;

m2reg = 1'bx;

shift = 1'b0;

alu\_imm = 1'b0;

sext = 1'b1;

alu\_c = 4'bx010;

wmem = 1'b0;

pcsrc = 2'b00;

end

end

6'b001111: begin //lui

wreg = 1'b1;

regrt = 1'b1;

jal = 1'b0;

m2reg = 1'b0;

shift = 1'bx;

alu\_imm = 1'b1;

sext = 1'bx;

alu\_c = 4'bx110;

wmem = 1'b0;

pcsrc = 2'b00;

end

6'b000010: begin //j

wreg = 1'b0;

regrt = 1'bx;

jal = 1'bx;

m2reg = 1'bx;

shift = 1'bx;

alu\_imm = 1'bx;

sext = 1'bx;

alu\_c = 4'bxxxx;

wmem = 1'b0;

pcsrc = 2'b11;

end

6'b000011: begin //jal

wreg = 1'b1;

regrt = 1'bx;

jal = 1'b1;

m2reg = 1'bx;

shift = 1'bx;

alu\_imm = 1'bx;

sext = 1'bx;

alu\_c = 4'bxxxx;

wmem = 1'b0;

pcsrc = 2'b11;

end

endcase

end

endmodule

module dst\_Mux (Ain, Bin, Sel, MUXout);

input [4:0] Ain, Bin;

input Sel;

output reg [4:0] MUXout;

always @(\*)

begin

MUXout <= Ain;

if (Sel == 1'b1)

begin

MUXout <= Bin;

end

end

endmodule

module f (jal, dst, dsel, wn);

input jal;

input [4:0] dst;

output reg dsel;

output reg [4:0] wn;

always @(\*)

begin

if (jal) begin

dsel = 1;

wn = 5'b11111;

end

else

begin

dsel = 0;

wn = dst;

end

end

endmodule

module MUX2to1 (ina, inb, sel, out);

input [31:0] ina, inb;

input sel;

output reg [31:0] out;

always @(\*)

begin

if (sel == 1)

begin

out <= inb;

end

else

out <= ina;

end

endmodule

module signextend (in16, sel, out32);

input [15:0] in16;

input sel;

output reg [31:0] out32;

always @(\*)

begin

if (sel == 1'b1)

out32 <= {{16{in16[15]}}, in16};

else

begin

out32 <= {16'b0, in16};

end

end

endmodule

module regfile (rna, rnb ,d ,wn ,we ,clk ,clrn ,qa ,qb); // 32x32 regfile

input [31:0] d; // data of write port

input [4:0] rna; // reg # of read port A

input [4:0] rnb; // reg # of read port B

input [4:0] wn; // reg # of write port

input we; // write enable

input clk, clrn; // clock and reset

output [31:0] qa, qb; // read ports A and B

reg [31:0] register [0:31]; // 31 32-bit registers

assign qa = (rna == 0)? 0 : register[rna]; // read port A

assign qb = (rnb == 0)? 0 : register[rnb]; // read port B

integer i;

always @(posedge clk or negedge clrn) // write port

if (!clrn)

for (i = 1; i < 32; i = i + 1)

register[i] <= 0; // reset

else

if ((wn != 0) && we) // not reg[0] & enabled

register[wn] <= d; // write d to reg[wn]

initial

begin

register[0] = 32'h00000000;

register[1] = 32'hA00000AA;

register[2] = 32'h10000011;

register[3] = 32'h20000022;

register[4] = 32'h30000033;

register[5] = 32'h40000044;

register[6] = 32'h50000055;

register[7] = 32'h60000066;

register[8] = 32'h70000077;

register[9] = 32'h80000088;

register[10] = 32'h90000099;

end

endmodule

module alu(INA, INB, alu\_c, alu\_result, zero);

input [31:0] INA, INB;

input [3:0] alu\_c;

output reg [31:0] alu\_result;

output reg zero;

always @(\*)

begin

case (alu\_c)

4'bx001: alu\_result = INA&INB;//and

4'bx101: alu\_result = INA|INB;//or

4'bx000: alu\_result = INA+INB;//add

4'bx100: alu\_result = INA-INB;//sub

4'bx010: alu\_result = INA^INB;

4'b0011: alu\_result = INB << INA;

default: alu\_result = 0;

endcase

if (INA == INB)

begin

zero = 1;

end

else

begin

zero = 0;

end

end

endmodule

module adder (A, B, S);

input [31:0] A, B;

output wire [31:0] S;

assign S = A+B;

endmodule

module pc\_mux (in0, in1, in2, in3, sel, out);

input [31:0] in0, in1, in2, in3;

input [1:0] sel;

output reg [31:0] out;

always @(\*)

begin

out <= in0;

if (sel == 2'b01)

begin

out <= in1;

end

else if (sel == 2'b10)

begin

out <= in2;

end

else if (sel == 2'b11)

begin

out <= in3;

end

end

endmodule

module Single\_Cycle\_CPU (instr, mem, clrn, clk, pc, alu, data, wmem);

input [31:0] instr, mem;

input clrn, clk;

output [31:0] pc, alu, data;

output wmem;

wire [31:0] new\_pc;

pc pc\_instr1 (new\_pc, clk, pc);

wire [31:0] p4;

pc\_adder pc\_adder\_instr1 (pc, p4);

wire z, wreg, regrt, jal, m2reg, shift, alu\_imm, sext;

wire [3:0] alu\_c;

wire [1:0] pcsrc;

Ctrl Ctrl\_instr1 (instr[31:26], instr[5:0], z, wreg, regrt, jal, m2reg, shift, alu\_imm, sext, alu\_c, wmem, pcsrc);

wire [4:0] dstout;

dst\_Mux dst\_Mux\_instr1 (instr[15:11], instr[20:16], regrt, dstout);

wire dsel;

wire [4:0] wn;

f f\_instr1 (jal, dstout, dsel, wn);

wire [31:0] dmux;

wire [31:0] dreg;

MUX2to1 MUX2to1\_inst2 (dmux, p4, dsel, dreg);

wire [31:0] imm;

signextend signextend\_instr1 (instr[15:0], sext, imm);

wire [31:0] qa, qb;

regfile regfile\_instr1 (instr[25:21], instr[20:16], dreg, wn, wreg, clk, clrn, qa, qb);

wire [31:0] alu\_a;

MUX2to1 MUX2to1\_inst3 (qa, {{27{1'b0}}, instr[10:6]}, shift, alu\_a);

wire [31:0] alu\_b;

MUX2to1 MUX2to1\_inst4 (qb, imm, alu\_imm, alu\_b);

alu alu\_instr1 (alu\_a, alu\_b, alu\_c, alu, z);

wire [31:0] p1;

adder adder\_instr1 (p4, {imm[29:0], 2'b00}, p1);

pc\_mux pc\_mux\_instr1 (p4, p1, qa, {p4[31:28], {instr[25:0], 2'b00}}, pcsrc, new\_pc);

MUX2to1 MUX2to1\_inst5 (alu, mem, m2reg, dmux);

endmodule

module Single\_Cycle\_Instr\_Mem (addr, instr);

input [31:0] addr;

output reg [31:0] instr;

reg [31:0] instr\_mem [0:32];

initial

begin

instr\_mem[0] <= 32'h3c010000;

instr\_mem[1] <= 32'h34240050;

instr\_mem[2] <= 32'h20050004;

instr\_mem[3] <= 32'h0c000018;

instr\_mem[4] <= 32'hac820000;

instr\_mem[5] <= 32'hac820000;

instr\_mem[6] <= 32'h01244022;

instr\_mem[7] <= 32'h20050003;

instr\_mem[8] <= 32'h20a5ffff;

instr\_mem[9] <= 32'h34a8ffff;

instr\_mem[10] <= 32'h39085555;

instr\_mem[11] <= 32'h2009ffff;

instr\_mem[12] <= 32'h312affff;

instr\_mem[13] <= 32'h01493025;

instr\_mem[14] <= 32'h01494026;

instr\_mem[15] <= 32'h01463824;

instr\_mem[16] <= 32'h10a00001;

instr\_mem[17] <= 32'h08000008;

instr\_mem[18] <= 32'h2005ffff;

instr\_mem[19] <= 32'h000543c0;

instr\_mem[20] <= 32'h00084400;

instr\_mem[21] <= 32'h00084403;

instr\_mem[22] <= 32'h000843c2;

instr\_mem[23] <= 32'h08000017;

instr\_mem[24] <= 32'h00004020;

instr\_mem[25] <= 32'h8c890000;

instr\_mem[26] <= 32'h20840004;

instr\_mem[27] <= 32'h01094020;

instr\_mem[28] <= 32'h20a5ffff;

instr\_mem[29] <= 32'h14a0fffb;

instr\_mem[30] <= 32'h00081000;

instr\_mem[31] <= 32'h03e00008;

end

always @(addr)

begin

instr = instr\_mem[addr/4];

end

endmodule

module Single\_Cycle\_Data\_Mem (clk, we, data\_in, addr, data\_out); // data memory, ram

input clk; // clock

input we; // write enable

input [31:0] data\_in; // data in (to memory)

input [31:0] addr; // ram address

output [31:0] data\_out; // data out (from memory)

reg [31:0] ram [0:31]; // ram cells: 32 words \* 32 bits

assign data\_out = ram[addr[6:2]]; // use word address to read ram

always @ (posedge clk)

if (we) ram[addr[6:2]] = data\_in; // use word address to write ram

integer i;

initial begin // initialize memory

for (i = 0; i < 32; i = i + 1)

ram[i] = 0;

// ram[word\_addr] = data // (byte\_addr)

ram[5'h14] = 32'h000000a3; // (50hex)

ram[5'h15] = 32'h00000027; // (54hex)

ram[5'h16] = 32'h00000079; // (58hex)

ram[5'h17] = 32'h00000115; // (5chex)

// ram[5'h18] should be 0x00000258, the sum stored by sw instruction

end

endmodule

module Datapath(clk, clrn, instr, pc, alu);

input clk, clrn;

output [31:0] instr, pc, alu;

wire wmem;

wire [31:0] data\_in;

wire [31:0] data\_out;

Single\_Cycle\_Data\_Mem Single\_Cycle\_Data\_Mem\_instr1 (clk, wmem, data\_in, alu, data\_out);

Single\_Cycle\_Instr\_Mem Single\_Cycle\_Instr\_Mem\_instr1 (pc, instr);

Single\_Cycle\_CPU Single\_Cycle\_CPU\_instr1 (instr, data\_out, clrn, clk, pc, alu, data\_in, wmem);

endmodule

Testbench

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer: Aly Ghallab

//

// Create Date: 05/04/2023 08:34:58 PM

// Design Name:

// Module Name: testbench

// Project Name: CMPEN 331 Final Project

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module testbench();

reg clk;

reg clrn;

wire [31:0] instr;

wire [31:0] pc;

wire [31:0] alu;

Datapath Datapath\_instr1 (clk, clrn, inst, pc, alu);

initial

begin

clk = 0;

clrn = 1;

end

always

begin

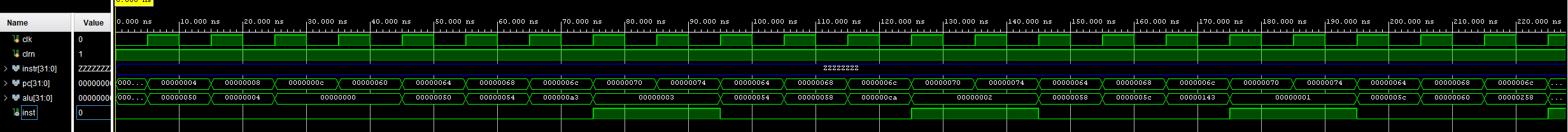
#5;

clk = ~clk;

end

endmodule

Waveforms



Schematics

A picture containing chart

Description automatically generated

Floor Planning

A picture containing text, electronics, display

Description automatically generated

IO Planning

A screenshot of a computer

Description automatically generated with medium confidence