CMPEN 331 Lab 3

20220320

Shi Qiu

`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create I\_Outte: 2022/03/20 18:23:14

// Design Name:

// Module Name: ID

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module Control(

input [5:0] op,

input [5:0] func,

output reg wreg,

output reg m2reg,

output reg wmem,

output reg [3:0] aluc,

output reg aluImm,

output reg regrt

);

always @(\*) begin

wreg <= 1;

m2reg <= 1;

wmem <= 0;

aluc <= 4'b0010;

aluImm <= 1;

regrt <= 1;

end

endmodule

module regRTMux(

input [4:0] rt,

input [4:0] rd,

input regrt,

output reg [4:0] destReg

);

always @ (\*) begin

if (~regrt) destReg <= rd;

else if (regrt) destReg <= rt;

end

endmodule

module regFile(

input [4:0] rs,

input [4:0] rt,

output reg [31:0] qa,

output reg [31:0] qb

);

reg [31:0] registers [0:31];

integer i;

initial //Initialize RegFile

begin

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

registers[i] <= 32'd0;

end

end

always @ (\*) begin

qa = registers[rs];

qb = registers[rt];

end

endmodule

module ImmExt(

input [15:0] imm,

output reg [31:0] imm32

);

always @ (\*) begin

imm32 <= {imm[15], {16{1'b0}}, imm[14:0]};

end

endmodule

module IDEXEreg(

input wreg,

input m2reg,

input wmem,

input [3:0] aluc,

input aluimm,

input [4:0] destReg,

input [31:0] qa,

input [31:0] qb,

input [31:0] imm32,

input clk,

output reg ewreg,

output reg em2reg,

output reg ewmem,

output reg [3:0] ealuc,

output reg ealuimm,

output reg [4:0] edestReg,

output reg [31:0] eqa,

output reg [31:0] eqb,

output reg [31:0] eimm32

);

always @ (posedge clk) begin

ewreg <= wreg;

em2reg <= m2reg;

ewmem <= wmem;

ealuc <= aluc;

ealuimm <= aluimm;

edestReg <= destReg;

eqa <= qa;

eqb <= qb;

eimm32 <= imm32;

end

endmodule

module InstDecode(

input [31:0] dI\_Out,

input clk,

output ewreg,

output em2reg,

output ewmem,

output [3:0] ealuc,

output ealuimm,

output [4:0] edestReg,

output [31:0] eqa,

output [31:0] eqb,

output [31:0] eimm32

);

wire wwreg;

wire wm2reg;

wire wwmem;

wire [3:0] waluc;

wire waluimm;

wire [4:0] wdestReg;

wire [31:0] wqa;

wire [31:0] wqb;

wire [31:0] wimm32;

wire wrtMUX;

Control c(dI\_Out[31:26], dI\_Out[5:0], wwreg, wm2reg, wwmem, waluc, waluimm, wrtMUX);

regRTMux rtMux(dI\_Out[20:16], dI\_Out[15:11], wrtMUX, wdestReg);

regFile rf(dI\_Out[25:21], dI\_Out[20:16], wqa, wqb);

ImmExt extend(dI\_Out[15:0], wimm32);

IDEXEreg idexe1(wwreg, wm2reg, wwmem, waluc, waluimm, wdestReg, wqa, wqb, wimm32, clk, ewreg, em2reg, ewmem, ealuc, ealuimm, edestReg, eqa, eqb, eimm32);

Endmodule

`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create I\_Outte: 2022/03/20 18:23:14

// Design Name:

// Module Name: IF

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module pc(

input [31:0] nextPC ,

input clk,

output reg [31:0] pc

); always @(posedge clk) begin

pc <= nextPC;

end

endmodule

module instMem(

input [31:0] pc,

output reg [31:0] I\_Out

);

reg [31:0] mem [0:511];

initial begin // Temp Assigned value

mem[100] = {

6'b100011,

5'b00001,

5'b00010,

5'b00000,

5'b00000,

6'b000000

};

mem[104] = {

6'b100011,

5'b00001,

5'b00011,

5'b00000,

5'b00000,

6'b000100

};

end

always @ (\*) begin

I\_Out <= mem[pc[7:2]];

end

endmodule

module pcAdder(

input [31:0] pc,

//input [31:0] increment, reserved for j inst

output reg [31:0] nextPC

);

always @ (\*) begin

nextPC <= pc + 4;

end

endmodule

module IFIDreg(

input [31:0] I\_Out,

input clk,

output reg [31:0] dI\_Out,

output reg [5:0] op,

output reg [5:0] func,

output reg [4:0] rd,

output reg [4:0] rt,

output reg [4:0] rs,

output reg [15:0] imm

);

always @ (\*) begin

dI\_Out <= I\_Out;

op = I\_Out[31:26];

func = I\_Out[5:0];

rd = I\_Out[15:11];

rt = I\_Out[20:16];

rs = I\_Out[25:21];

imm = I\_Out[15:0];

end

endmodule

module InstFetch(

input clk,

output [31:0] dI\_Out

);

wire [31:0] wpc;

wire [31:0] wnextPC;

wire [31:0] wI\_Out;

pc pc1(wnextPC, clk, wpc);

pcAdder pcadd(wpc, wnextPC);

instMem inmem(wpc, wI\_Out);

IFIDreg IFID(wI\_Out, clk, dI\_Out);

Endmodule

`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create I\_Outte: 2022/03/20 18:28:49

// Design Name:

// Module Name: I\_OuttaPath

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module I\_OuttaPath(

input clk,

output ewreg,

output em2reg,

output ewmem,

output [3:0] ealuc,

output ealuimm,

output [4:0] edestReg,

output [31:0] eqa,

output [31:0] eqb,

output [31:0] eimm32

);

wire [31:0] inst;

InstFetch tInstFetch(clk, inst);

InstDecode tInstDecode(inst, clk, ewreg, em2reg, ewmem, ealuc, ealuimm, edestReg, eqa, eqb, eimm32);

endmodule

`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create I\_Outte: 2022/03/20 18:49:13

// Design Name:

// Module Name: TB

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module TB();

reg clk;

wire twreg;

wire tm2eg;

wire TB\_twmeml;

wire [3:0] taluc;

wire [5:0] talu;

wire [4:0] tdestReg;

wire [31:0] tqa;

wire [31:0] tqb;

wire [31:0] imm32;

I\_OuttaPath dp (clk, twreg, tm2eg, TB\_twmeml, talu, tdestReg, tqa, tqb, imm32);

initial begin

clk = 0;

#100 $finish;

end

always begin

#5;

clk = ~clk;

end

endmodule