**CECS 225**

**Lab 6**

**Ronnie Casasola**

**012737398**

**11/17/16**

Section 1: **datapath** Verilog module source

`timescale 1ns / 1ps

module datapath(

input clk,

input reset,

input memtoreg,

input pcsrc,

input alusrc,

input regdst,

input regwrite,

input jump,

input [2:0] alucontrol,

input [31:0] instr,

input [31:0] readdata,

output zero,

output [31:0] pc,

output [31:0] aluresult,

output [31:0] writedata

);

wire [4:0] writereg;

wire [31:0] pcnext, pcnextbr, pcplus4, pcbranch;

wire [31:0] signimm, signimmsh;

wire [31:0] srca, srcb, result;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* next PC logic \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//program counter

flopr pcreg( .clk( clk ),

.reset( reset ),

.d( pcnext[7:0] ),

.q( pc[7:0] ));

//Adder to increment the PC value

Adder\_32 pcadd1( .augend(pc ),

.addend( 32'h4 ),

.sum( pcplus4 ));

// shift left the sign extended constant

sl2 immsh( .a( signimm ),

.y( signimmsh ));

//branch address adder

Adder\_32 pcadd2( .augend( signimmsh ),

.addend( pcplus4 ),

.sum( pcbranch ));

//branch mux

mux2\_32 pcbrmux( .in0( pcplus4 ),

.in1( pcbranch ),

.sel( pcsrc ),

.out( pcnextbr ));

//jump mux

mux2\_32 pcmux( .in0( pcnextbr ),

.in1( {pcplus4[31:28], instr[25:0], 2'b00}),

.sel( jump ),

.out( pcnext ));

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* register file logic\*\*\*\*\*\*\*\*\*\*\*\*/

// The register file

regfile rf( .clk( clk ),

.we3( regwrite ),

.ra1( instr[25:21] ),

.ra2( instr[20:16] ),

.wa3( writereg ),

.wd3( result ),

.rd1( srca ),

.rd2( writedata ));

//destination register selection mux

mux2\_5 wrmux( .in0( instr[20:16] ),

.in1( instr[15:11] ),

.sel( regdst ),

.out( writereg ));

//register data input selection mux

mux2\_32 resmux( .in0( aluresult ),

.in1( readdata ),

.sel( memtoreg ),

.out( result ));

// sign extender to make 32-bit constant

signext se( .a( instr[15:0] ),

.y( signimm ));

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\* ALU Logic \*\*\*\*\*\*\*\*\*\*/

//ALU input B selection mux

mux2\_32 srcbmux( .in0( writedata ),

.in1( signimm ),

.sel( alusrc ),

.out( srcb ));

//The arithmetic Logic unit

ALU alu( .ALUControl( alucontrol ),

.A( srca ),

.B( srcb ),

.Result( aluresult ),

.Zero\_Flag( zero ));

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Endmodule

Section 2**: datapath\_tester** Verilog Test Fixture

`timescale 1ns / 1ps

module datapath\_tester;

// Inputs

reg clk;

reg reset;

reg memtoreg;

reg pcsrc;

reg alusrc;

reg regdst;

reg regwrite;

reg jump;

reg [2:0] alucontrol;

reg [31:0] instr;

reg [31:0] readdata;

// Outputs

wire zero;

wire [31:0] pc;

wire [31:0] aluresult;

wire [31:0] writedata;

//Internal variables

reg [5:0] Opcode, func;

reg [4:0] rs, rt, rd;

reg [15:0] imm;

// Instantiate the Unit Under Test (UUT)

datapath uut (

.clk(clk),

.reset(reset),

.memtoreg(memtoreg),

.pcsrc(pcsrc),

.alusrc(alusrc),

.regdst(regdst),

.regwrite(regwrite),

.jump(jump),

.alucontrol(alucontrol),

.instr(instr),

.readdata(readdata),

.zero(zero),

.pc(pc),

.aluresult(aluresult),

.writedata(writedata)

);

assign pc[31:8] = 0; //upper 24 bits of program counter not used

always #5 clk = ~clk; //clock pulse generation

integer i;

initial begin

// initialize contents of registers

uut.rf.rf[0] = 0;

uut.rf.rf[11] = 50;

uut.rf.rf[12] = 100;

uut.rf.rf[13] = 150;

uut.rf.rf[14] = 200;

uut.rf.rf[15] = 250;

uut.rf.rf[31] = 32'hFFFFFFFF;

// Initialize Inputs

clk = 0;

reset = 1;

memtoreg = 0;

pcsrc = 0;

alusrc = 0;

regdst = 0;

regwrite = 0;

jump = 0;

alucontrol = 0;

instr = 0;

readdata = 0;

@(posedge clk);

#1 reset = 0;

@(negedge clk)#1;

$display("Begin MIPS datapath Test");

$display("");

$display("Show initial register contents");

regdump();

//initialize control signal values for R-type instructions

memtoreg = 0;

pcsrc = 0;

alusrc = 0;

regdst = 1;

regwrite = 1;

jump = 0;

readdata = 32'hX;

Opcode = 6'h00;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Test case 0: add $16, $14, $12

@(posedge clk)

alucontrol = 2;

rs = 14;

rt = 12;

rd = 16;

func = 6'h20;

instr = {Opcode, rs, rt, rd, 5'h0, func};

@(negedge clk) #1;

$display("Test case 0" );

$display("check result of add $16, $14, $12");

$display("Expected Result: $16 = %h", uut.rf.rf[14] + uut.rf.rf[12] );

$display("Actual Result: $%d = %h", rd, uut.rf.rf[16]);

testCaseOutcome( uut.rf.rf[14] + uut.rf.rf[12], uut.rf.rf[16]);

$display("");

//Test case 1: sub $17, $15, $11

@(posedge clk)

alucontrol = 6;

rs = 15;

rt = 11;

rd = 17;

func = 6'h22;

instr = {Opcode, rs, rt, rd, 5'h0, func};

@(negedge clk) #1;

$display("Test case 1" );

$display("check result of sub $17, $15, $11");

$display("Expected Result: $17 = %h", uut.rf.rf[15] - uut.rf.rf[11] );

$display("Actual Result: $%d = %h", rd, uut.rf.rf[rd]);

testCaseOutcome( uut.rf.rf[15] - uut.rf.rf[11], uut.rf.rf[rd]);

$display("");

//Test case 2: and $18, $31, $0

@(posedge clk)

alucontrol = 0;

rs = 31;

rt = 0;

rd = 18;

func = 6'h24;

instr = {Opcode, rs, rt, rd, 5'h0, func};

@(negedge clk) #1;

$display("Test case 2" );

$display("check result of and $18, $31, $0");

$display("Expected Result: $18 = %h", uut.rf.rf[31] & uut.rf.rf[0] );

$display("Actual Result: $%d = %h", rd, uut.rf.rf[rd]);

testCaseOutcome( uut.rf.rf[31] & uut.rf.rf[0], uut.rf.rf[rd]);

$display("");

//Test case 3: or $19, $17, $31

@(posedge clk)

alucontrol = 1;

rs = 17;

rt = 31;

rd = 19;

func = 6'h25;

instr = {Opcode, rs, rt, rd, 5'h0, func};

@(negedge clk) #1;

$display("Test case 3" );

$display("check result of or $19, $17, $31");

$display("Expected Result: $19 = %h", uut.rf.rf[19] | uut.rf.rf[31] );

$display("Actual Result: $%d = %h", rd, uut.rf.rf[rd]);

testCaseOutcome( uut.rf.rf[19] | uut.rf.rf[31], uut.rf.rf[rd]);

$display("");

//Test case 4: slt $20, $17, $16

@(posedge clk)

alucontrol = 7;

rs = 17;

rt = 16;

rd = 20;

func = 6'h2A;

instr = {Opcode, rs, rt, rd, 5'h0, func};

@(negedge clk) #1;

$display("Test case 4" );

$display("check result of slt $20, $17, $16");

$display("Expected Result: $20 = %h", {{31'b0},uut.rf.rf[17] < uut.rf.rf[16]} );

$display("Actual Result: $%d = %h", rd, uut.rf.rf[rd]);

testCaseOutcome( {{31'b0},uut.rf.rf[17] < uut.rf.rf[16]}, uut.rf.rf[rd]);

$display("");

//Test case 5: slt $21, $16, $17

@(posedge clk)

alucontrol = 7;

rs = 16;

rt = 17;

rd = 21;

func = 6'h2A;

instr = {Opcode, rs, rt, rd, 5'h0, func};

@(negedge clk) #1;

$display("Test case 5" );

$display("check result of slt $21, $16, $17");

$display("Expected Result: $21 = %h", {{31'b0},uut.rf.rf[16] < uut.rf.rf[17]} );

$display("Actual Result: $%d = %h", rd, uut.rf.rf[21]);

testCaseOutcome( {{31'b0},uut.rf.rf[16] < uut.rf.rf[17]}, uut.rf.rf[21]);

$display("");

$display("");

$display("Show final register contents");

regdump();

$display("End MIPS datapath Test");

@(negedge clk) #1;

$finish;

end

task regdump;

begin

$display("Scan registers for valid data");

$display("The following registers have data");

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

if( uut.rf.rf[i] || uut.rf.rf[i] == 32'h0)

$display("register%d = 32'h%h'" , i, uut.rf.rf[i]);

$display("");

end

endtask

task testCaseOutcome;

input [31:0] input1, input2;

if(input1 == input2)

$display("Test passed!");

else

begin

$display("Tests Failed");

checksignals();

end

endtask

task checksignals;

begin

$display("All signals for tested instructions");

$display("RegDst Mux Signals");

$display("\tData Inputs");

$display("\t\tIn0 = %h\tIn1 = %h", uut.instr[20:16], uut.instr[15:11]);

$display("\tControl Inputs");

$display("\t\tSel = %b", uut.regdst);

$display("\tOutputs");

$display("\t\tOut = %h", uut.writereg);

$display("Register File Signals");

$display("\tData Inputs");

$display("\t\trs = %h\trt = %h\tWriteReg = %h\tWriteData = %h",

uut.instr[25:21], uut.instr[20:16], uut.writereg, uut.writedata);

$display("\tControl Inputs");

$display("\t\tRegWrite = %b", uut.regwrite);

$display("\tOutputs");

$display("\t\tRD1 = %h\tRD2 = %h", uut.srca, uut.writedata);

//Sign extender

$display("Sign Extender Signals");

$display("\tData Inputs");

$display("\t\tInstr[15:0] = %h", uut.instr[15:0]);

$display("\tOutputs");

$display("\t\tSignImm = %h", uut.signimm);$display("");

//ALU Source B Selection mux

$display("ALUSrc Mux Signals");

$display("\tData Inputs");

$display("\t\tIn0 = %hIn1 = %h", uut.writedata, uut.signimm);

$display("\tControl Inputs");

$display("\t\tOut = %h", uut.srcb);

//ALU data signals

$display("ALU Signals");

$display("\tData Inputs");

$display("\t\tSrcA = %h\tSrcB = %h", uut.srca, uut.srcb);

$display("\tControl Inputs");

$display("\t\tALUControl = %h", uut.alucontrol);

$display("tOutputs");

$display("\t\tALUResult = %h\tZero = %h", uut.aluresult, uut.zero);

$display("");

$display("");

end

endtask

endmodule

Section 3: **datapath\_tester** Simulation console message

