CECS 341 Lab 5

Shawn Joseph

Ty Tanaka

Yiang Shen

To design the datapath, we used the module as a basis and modified any components with extra ports not in the original datapath so they had the correct amount of outputs. Afterwards, we added the new components to the datapath such as the muxes and the sign extension unit. Following that, we made each of the modules function by changing the code for each module so it would work properly.

We had to modify the control unit to send more signals to allow for I-type instructions. In addition, we had to remove the PCADD module, replacing it with a sign extender, Shift Left modifier, 2 add circuits, and some other circuits to allow for branching operations to be performed. In addition, we had to modify the datapath to add several new wires and connections.

Testbench:

`timescale 10ns / 1ps

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

// Company:

// Engineer:

//

// Create Date: 10/18/2020 05:30:29 PM

// Design Name:

// Module Name: Datapath\_tb

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module Datapath\_tb;

reg Reset;

reg clock;

wire [31:0]Dout;

reg [6:0] i;

Datapath uut (.clk(clock),.reset(Reset),.DataOut(Dout));

initial begin

$readmemh("imem.dat",uut.imem.imem);

$readmemh("regfile.dat",uut.rf32.regArray);

$readmemh("Datamem.dat",uut.d.dmem);

clock = 1;

Reset =1;

#1 clock = 0;

#1 clock = 1;

Reset = 0;

for(i = 0; i < 5'd60; i = i + 1) begin

#1 clock = ~clock;

if(~clock) begin

$display("Time: %t, Output: %h", $time, Dout);

end

end

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

$display("Register %d: %h", i, uut.rf32.regArray[i]);

end

$display("Memory values:");

for(i = 0; i < 5'd24; i = i + 1) begin

$display("%d: %h", i, uut.d.dmem[i]);

end

end

endmodule

`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create Date: 2020/09/22 08:23:26

// Design Name:

// Module Name: ALU

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module ALU(

input [31:0] A,

input [31:0] B,

input [3:0] ALUCntl,

output reg [31:0] ALUout,

output reg C,

output reg N,

output Z,

output reg V

);

reg signed [31:0]A\_s,B\_s;

assign Z = (ALUout == 32'b0)?1'b1:1'b0;

always@(\*)begin

A\_s = A;

B\_s = B;

case(ALUCntl)

4'b1010:begin //add

{C,ALUout} = A\_s + B\_s;

if((A\_s[31] & B\_s[31] & ~ALUout[31])||(~A\_s[31] & ~B\_s[31] & ALUout[31]))//If a is pos,b is pos and result is neg, Or vice visa

V = 1'b1;

else

V = 1'b0;

N = ALUout[31];

end

4'b1110:begin //Sub

{C,ALUout} = A\_s - B\_s;

if((A\_s[31] & ~B\_s[31] & ~ALUout[31])||(~A\_s[31] & B\_s[31] & ALUout[31]))

V = 1'b1;

else

V = 1'b0;

N = ALUout[31];

end

4'b0010:begin //add Unsigned

{C,ALUout} = A + B;

V = C;

N = 1'b0;

end

4'b0110:begin //Sub Unsigned

{C,ALUout} = A - B;

V = C;

N = 1'b0;

end

4'b0000:begin //And

{V,C,ALUout} = {2'bx,(A&B)};

N = ALUout[31];

end

4'b0001:begin //Or

{V,C,ALUout} = {2'bx,(A|B)};

N = ALUout[31];

end

4'b0011:begin //XOr

{V,C,ALUout} = {2'bx,(A^B)};

N = ALUout[31];

end

4'b1100:begin //Nor

{V,C,ALUout} = {2'bx,~(A|B)};

N = ALUout[31];

end

4'b0111:begin //Not

{V,C,ALUout} = {2'bx,~(A)};

N = ALUout[31];

end

4'b1101:begin //Not

{C,ALUout} = {(A)<<1};

V = 1'bx;

N = ALUout[31];

end

4'b1111:begin //SLT

C = 0;

V = 0;

N = 0;

if(A[31] & ~B[31])

ALUout = 0;

else if(B[31] & ~A[31])

ALUout = 1;

else if(~B[31] & ~A[31])

begin

if(A > B)

ALUout = -1;

else

ALUout = 0;

end

else

begin

if(A < B)

ALUout = -1;

else

ALUout = 0;

end

end

4'b0100: begin //SLTU

C = 0;

V = 0;

N = 0;

if(A > B)

ALUout = -1;

else

ALUout = 0;

end

default: begin

C = 0;

N = 0;

V = 0;

ALUout = 0;

end

endcase

end

endmodule

`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create Date: 10/15/2020 08:19:07 PM

// Design Name:

// Module Name: Control

// 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 RegDst, MemRead, MemToReg, MemWrite, ALUSrc, RegWrite,

output reg [1:0] Branch,

output reg [3:0] ALUCntl

);

always @(\*) begin

case(Op)

6'b0:begin

RegWrite <= 1;

RegDst <= 1;

Branch <= 0;

MemRead <= 0;

MemToReg <= 0;

MemWrite <= 0;

ALUSrc <= 0;

case(Func)

6'h20:begin //add

ALUCntl <= 4'b1010;

end

6'h21:begin //addu

ALUCntl <= 4'b0010;

end

6'h22:begin //sub

ALUCntl <= 4'b1110;

end

6'h23:begin //subu

ALUCntl <= 4'b0110;

end

6'h24:begin //and

ALUCntl <= 4'b0000;

end

6'h25:begin //or

ALUCntl <= 4'b0001;

end

6'h26:begin //xor

ALUCntl <= 4'b0011;

end

6'h27:begin //nor

ALUCntl <= 4'b1100;

end

6'h2A:begin //SLT

ALUCntl <= 4'b1111;

end

6'h2B:begin //SLTU

ALUCntl <= 4'b0100;

end

default:begin //invalid code

$display("Error! Check control file");

end

endcase

end

6'h08:begin //addi

RegWrite <= 1;

Branch <= 0;

MemWrite <= 0;

MemRead <= 0;

MemToReg <= 0;

RegDst <= 0;

ALUCntl <= 4'b1010;

ALUSrc <= 1;

end

6'h09:begin //addiu

Branch <= 0;

RegWrite <= 1;

MemWrite <= 0;

MemRead <= 0;

MemToReg <= 0;

RegDst <= 0;

ALUCntl <= 4'b0010;

ALUSrc <= 1;

end

6'h0C:begin //And immediate

RegWrite <= 1;

Branch <= 0;

MemWrite <= 0;

MemRead <= 0;

MemToReg <= 0;

RegDst <= 0;

ALUCntl <= 4'b0000;

ALUSrc <= 1;

end

6'h0D:begin //Ori

RegWrite <= 1;

Branch <= 0;

MemWrite <= 0;

MemRead <= 0;

MemToReg <= 0;

RegDst <= 0;

ALUSrc <= 1;

MemWrite <= 0;

ALUCntl <= 4'b0001;

end

6'h23:begin //lw

Branch <= 0;

RegDst <= 0;

RegWrite <= 1;

ALUSrc <= 1;

ALUCntl <= 4'b1010;

MemWrite <= 0;

MemRead <= 1;

MemToReg <= 1;

end

6'h2B:begin //sw

Branch <= 0;

RegDst <= 0;

RegWrite <= 0;

ALUSrc <= 1;

ALUCntl <= 4'b1010;

MemWrite <= 1;

MemRead <= 0;

MemToReg <= 0;

end

6'h04:begin //beq

RegDst <= 0;

Branch <= 2'b10;

RegWrite <= 0;

ALUCntl <= 4'b1110;

ALUSrc <= 0;

MemWrite <= 0;

MemRead <= 0;

MemToReg <= 0;

end

6'h05:begin //bne

RegDst <= 0;

Branch <= 2'b01;

RegWrite <= 0;

ALUCntl <= 4'b1110;

ALUSrc <= 0;

MemWrite <= 0;

MemRead <= 0;

MemToReg <= 0;

end

6'h0A:begin //slti

Branch <= 0;

RegWrite <= 1;

MemWrite <= 0;

MemRead <= 0;

MemToReg <= 0;

RegDst <= 0;

ALUCntl <= 4'b1111;

ALUSrc <= 1;

end

6'h0B:begin //sltiu

Branch <= 0;

RegWrite <= 1;

MemWrite <= 0;

MemRead <= 0;

MemToReg <= 0;

RegDst <= 0;

ALUCntl <= 4'b0100;

ALUSrc <= 1;

end

endcase

end

endmodule

`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create Date: 10/15/2020 09:26:41 PM

// Design Name:

// Module Name: Datapath

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module Datapath(input clk, input reset, output [31:0] DataOut);

wire [31:0] Dout;

wire [31:0]Cout;

wire [31:0]Cin;

wire [31:0]Inst\_out,S,T;

wire [3:0]ALUCntl;

wire RegWrite,C,N,Z,V;

wire [4:0] wr;

wire [1:0] br;

wire RegDst;

wire mr;

wire mtr;

wire mw;

wire ALUSrc;

wire [31:0] ALUin;

wire [31:0] SEval;

wire [31:0] rd;

wire [31:0] shl2;

wire [31:0] Co4;

ProgramCounter pc(.clk(clk),.reset(reset),.Din(Cin),.Dout(Cout));

assign Co4 = Cout + 4;

ShiftLeftTwo sll(.in(SEval), .out(shl2));

Mux32x1 adder(.zero(Co4), .one(shl2 + Co4), .ctrl(br[0] & Z | br[1] & ~Z), .out(Cin));

Instruction\_Memory imem(.Addr(Cout),.Inst\_out(Inst\_out));

Mux4x1 m(.zero(Inst\_out[20:16]), .one(Inst\_out[15:11]), .ctrl(RegDst), .out(wr));

regfile32 rf32(.clk(clk),.reset(reset),.D\_En(RegWrite),.D\_Addr(wr),

.S\_Addr(Inst\_out[25:21]),.T\_Addr(Inst\_out[20:16]), .D(DataOut),.S(S),.T(T) );

SignExtend se(.in(Inst\_out[15:0]), .out(SEval));

Control cntl(.Op(Inst\_out[31:26]),.Func(Inst\_out[5:0]), .ALUCntl(ALUCntl), .RegWrite(RegWrite),

.Branch(br), .MemRead(mr), .MemToReg(mtr), .MemWrite(mw), .ALUSrc(ALUSrc), .RegDst(RegDst));

Mux32x1 e(.zero(T), .one(SEval), .ctrl(ALUSrc), .out(ALUin));

ALU alu(.A(S),.B(ALUin),.ALUCntl(ALUCntl),.ALUout(Dout),.C(C),.N(N),.Z(Z),.V(V));

DataMem d(.clk(clk), .addr(Dout), .wr\_data(T), .mem\_wr(mw), .mem\_rd(mr), .rd\_data(rd));

Mux32x1 dMux(.one(rd), .zero(Dout), .ctrl(mtr), .out(DataOut));

Endmodule

`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create Date: 11/03/2020 09:20:34 AM

// Design Name:

// Module Name: Mux32x1

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module Mux32x1(

input [31:0] zero,

input [31:0] one,

input ctrl,

output reg [31:0] out

);

always @(\*) begin

if(ctrl)

out = one;

else

out = zero;

end

endmodule

`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create Date: 10/15/2020 08:19:07 PM

// Design Name:

// Module Name: ProgramCounter

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module ProgramCounter(

input clk,

input reset,

input [31:0] Din,

output reg[31:0] Dout

);

always @(posedge clk) begin

if(reset)

Dout = 0;

else

Dout = Din;

end

endmodule

`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create Date: 11/03/2020 12:30:21 AM

// Design Name:

// Module Name: ShiftLeftTwo

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module ShiftLeftTwo(

input [31:0] in,

output reg [31:0] out

);

reg [31:0] dout;

always @(\*) begin

dout[1:0] <= 2'b0;

dout[31:2] <= in[29:0];

out = dout;

end

endmodule

`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create Date: 11/03/2020 12:25:31 AM

// Design Name:

// Module Name: SignExtend

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module SignExtend(

input [15:0] in,

output reg [31:0] out

);

always @(\*) begin

if(in[15] == 1) begin

out[31:16] = 15'h7FFF;

end

if(in[15] == 0) begin

out[31:16] = 15'b0;

end

out[15:0] = in;

end

endmodule

`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create Date: 11/03/2020 08:43:18 AM

// Design Name:

// Module Name: Mux4x1

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module Mux4x1(

input [4:0] zero,

input [4:0] one,

input ctrl,

output reg [4:0] out

);

always @(\*) begin

if(ctrl == 0)

out = zero;

if(ctrl == 1)

out = one;

end

endmodule

RTL: A screenshot of a circuit board

Description automatically generated

Waveform:

A circuit board

Description automatically generated

Output:

Time: 30000, Output: ffffffff

Time: 50000, Output: 12345678

Time: 70000, Output: 00000008

Time: 90000, Output: 00000001

Time: 110000, Output: 00000003

Time: 130000, Output: 00000001

Time: 150000, Output: 00000000

Time: 170000, Output: 1234567f

Time: 190000, Output: 00000000

Time: 210000, Output: ffffffff

Time: 230000, Output: 00000002

Time: 250000, Output: 00000002

Time: 270000, Output: 00000018

Time: 290000, Output: 0000001c

Register 0: 00000000

Register 1: 00000000

Register 2: 00000000

Register 3: 00000000

Register 4: 00000000

Register 5: 00000000

Register 6: 00000000

Register 7: 00000000

Register 8: 00000001

Register 9: 00000000

Register 10: 1234567f

Register 11: 00000000

Register 12: ffffffff

Register 13: 00000002

Register 14: 0000000f

Register 15: 00000010

Register 16: 00000011

Register 17: 00000012

Register 18: 00000013

Register 19: 00000014

Register 20: 00000015

Register 21: 00000016

Register 22: 00000017

Register 23: 00000018

Register 24: 00000019

Register 25: 0000001a

Register 26: 00000000

Register 27: 00000000

Register 28: 00000000

Register 29: 00000000

Register 30: 00000000

Register 31: 00000000

Memory values:

0: 00

1: 00

2: 00

3: 00

4: ff

5: ff

6: ff

7: ff

8: 12

9: 34

10: 56

11: 78

12: 00

13: 00

14: 00

15: 08

16: 00

17: 00

18: 00

19: 01

20: 00

21: 00

22: 00

23: 03