ELEC 374 CPU Project: Phase 2

Madeleine Asselin, 20154065

Julia Newcombe, 20146379

Aidan Hawthorne, 20171367

March 25, 2024

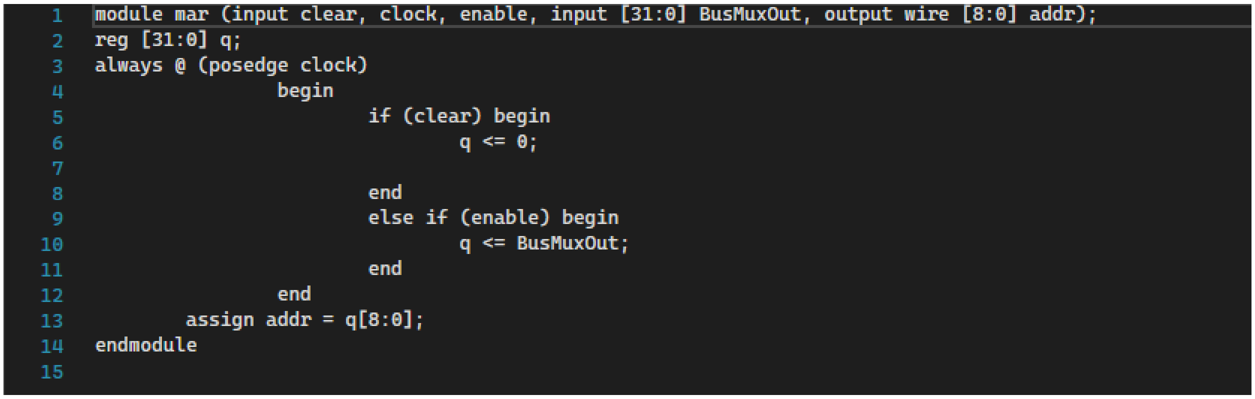
Academic Integrity Acknowledgement

We do hereby verify that this written lab report is our own work and contains our own original ideas, concepts, and designs. No portion of this report has been copied in whole or in part from another source, with the possible exception of properly referenced material.

New Code

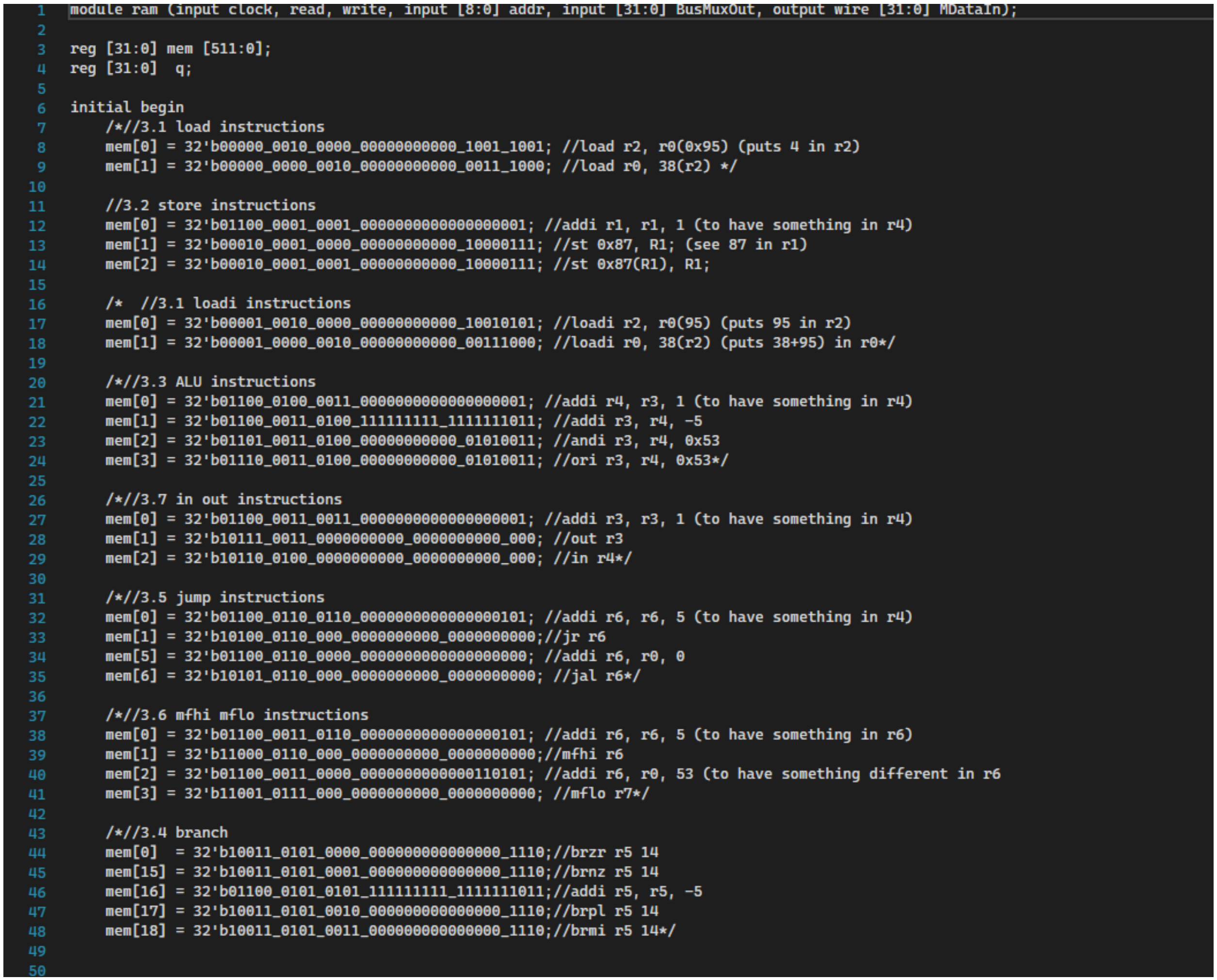
Memory Subsystem

Mar

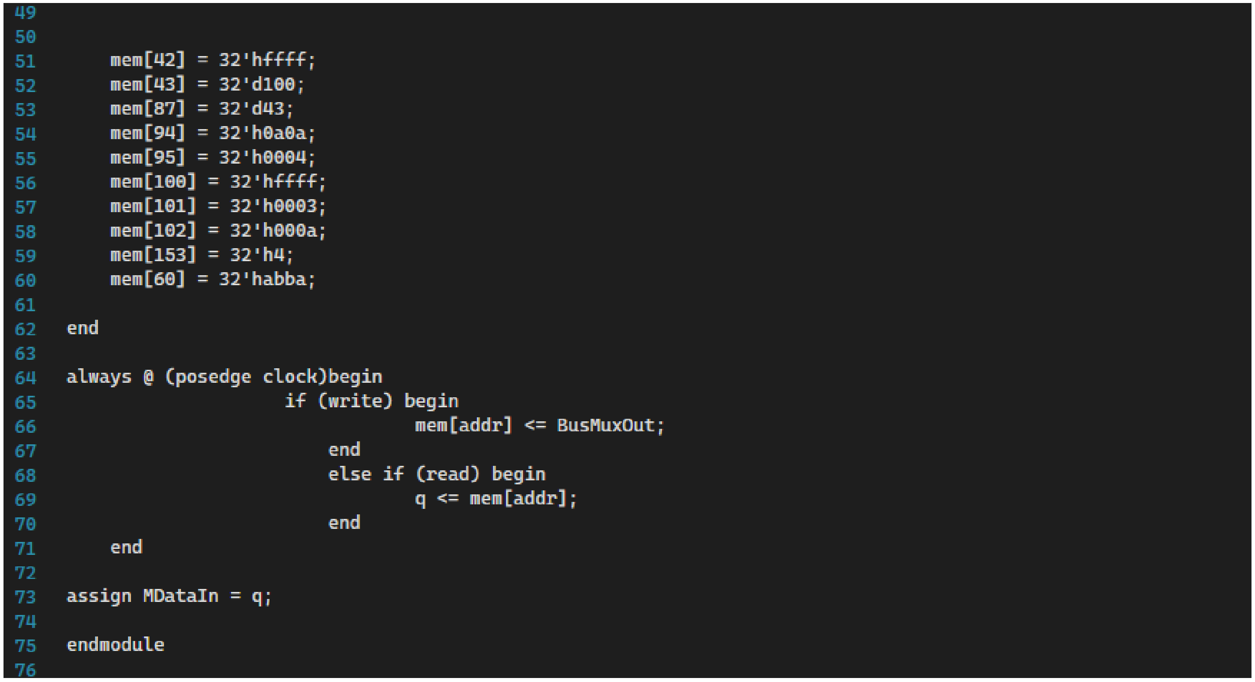


```
module mar (input clear, clock, enable, input [31:0] BusMuxOut, output wire [8:0] addr);
reg [31:0] q;
always @ (posedge clock)
                begin
                        if (clear) begin
                                q <= 0;

                        end
                        else if (enable) begin
                                q <= BusMuxOut;
                        end
                end
        assign addr = q[8:0];
endmodule

```

RAM

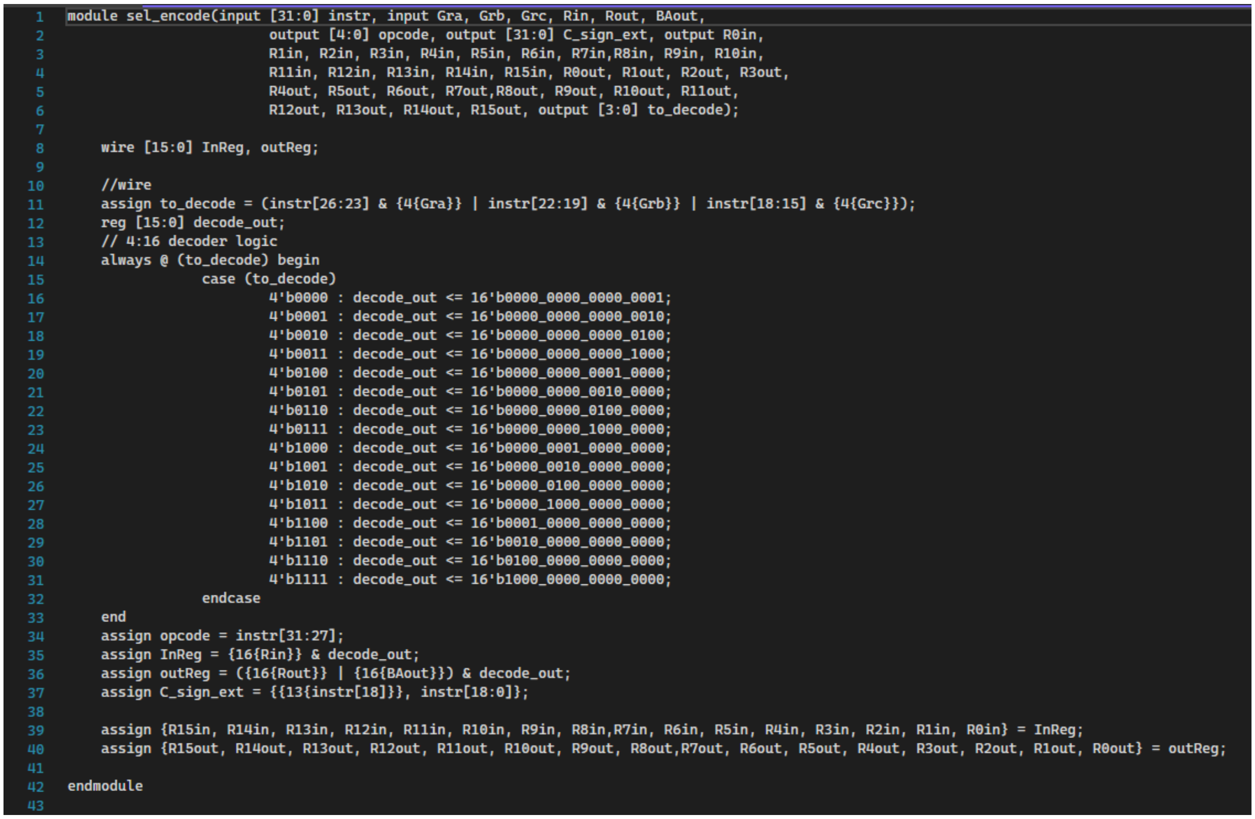


```
module ram (input clock, read, write, input [8:0] addr, input [31:0] BusMuxOut, output wire [31:0] MDataIn);

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

initial begin
    /*//3.1 load instructions
    mem[0] = 32'b00000_0010_0000_00000000000_1001_1001; //load r2, r0(0x95) (puts 4 in r2)
    mem[1] = 32'b00000_0000_0010_00000000000_0011_1000; //load r0, 38(r2) */

    //3.2 store instructions
    mem[0] = 32'b01100_0001_0001_0000000000000000001; //addi r1, r1, 1 (to have something in r4)
    mem[1] = 32'b00010_0001_0000_00000000000_10000111; //st 0x87, R1; (see 87 in r1)
    mem[2] = 32'b00010_0001_0001_00000000000_10000111; //st 0x87(R1), R1;

    /*  //3.1 loadi instructions
    mem[0] = 32'b00001_0010_0000_00000000000_10010101; //loadi r2, r0(95) (puts 95 in r2)
    mem[1] = 32'b00001_0000_0010_00000000000_00111000; //loadi r0, 38(r2) (puts 38+95) in r0*/

    /*//3.3 ALU instructions
    mem[0] = 32'b01100_0100_0011_0000000000000000001; //addi r4, r3, 1 (to have something in r4)
    mem[1] = 32'b01100_0011_0100_111111111_1111111011; //addi r3, r4, -5
    mem[2] = 32'b01101_0011_0100_00000000000_01010011; //andi r3, r4, 0x53
    mem[3] = 32'b01110_0011_0100_00000000000_01010011; //ori r3, r4, 0x53*/

    /*//3.7 in out instructions
    mem[0] = 32'b01100_0011_0011_0000000000000000001; //addi r3, r3, 1 (to have something in r4)
    mem[1] = 32'b10111_0011_0000000000_0000000000_000; //out r3
    mem[2] = 32'b10110_0100_0000000000_0000000000_000; //in r4*/

    /*//3.5 jump instructions
    mem[0] = 32'b01100_0110_0110_0000000000000000101; //addi r6, r6, 5 (to have something in r4)
    mem[1] = 32'b10100_0110_000_0000000000_0000000000;//jr r6
    mem[5] = 32'b01100_0110_0000_0000000000000000000; //addi r6, r0, 0
    mem[6] = 32'b10101_0110_000_0000000000_0000000000; //jal r6*/

    /*//3.6 mfhi mflo instructions
    mem[0] = 32'b01100_0011_0110_0000000000000000101; //addi r6, r6, 5 (to have something in r6)
    mem[1] = 32'b11000_0110_000_0000000000_0000000000;//mfhi r6
    mem[2] = 32'b01100_0011_0000_0000000000000110101; //addi r6, r0, 53 (to have something different in r6
    mem[3] = 32'b11001_0111_000_0000000000_0000000000; //mflo r7*/

    /*//3.4 branch
    mem[0]  = 32'b10011_0101_0000_000000000000000_1110;//brzr r5 14
    mem[15] = 32'b10011_0101_0001_000000000000000_1110;//brnz r5 14
    mem[16] = 32'b01100_0101_0101_111111111_1111111011;//addi r5, r5, -5
    mem[17] = 32'b10011_0101_0010_000000000000000_1110;//brpl r5 14
    mem[18] = 32'b10011_0101_0011_000000000000000_1110;//brmi r5 14*/


```



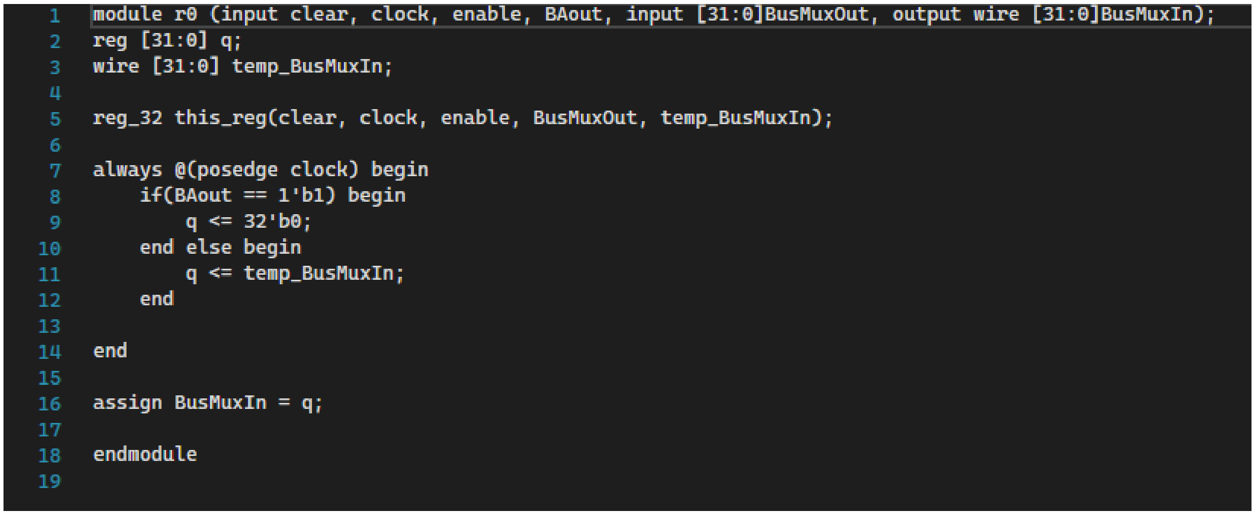
```


    mem[42] = 32'hffff;
    mem[43] = 32'd100;
    mem[87] = 32'd43;
    mem[94] = 32'h0a0a;
    mem[95] = 32'h0004;
    mem[100] = 32'hffff;
    mem[101] = 32'h0003;
    mem[102] = 32'h000a;
    mem[153] = 32'h4;
    mem[60] = 32'habba;

end

always @ (posedge clock)begin
                    if (write) begin
                                mem[addr] <= BusMuxOut;
                        end
                        else if (read) begin
                                q <= mem[addr];
                        end
    end

assign MDataIn = q;

endmodule

```

Select and Encode Logic



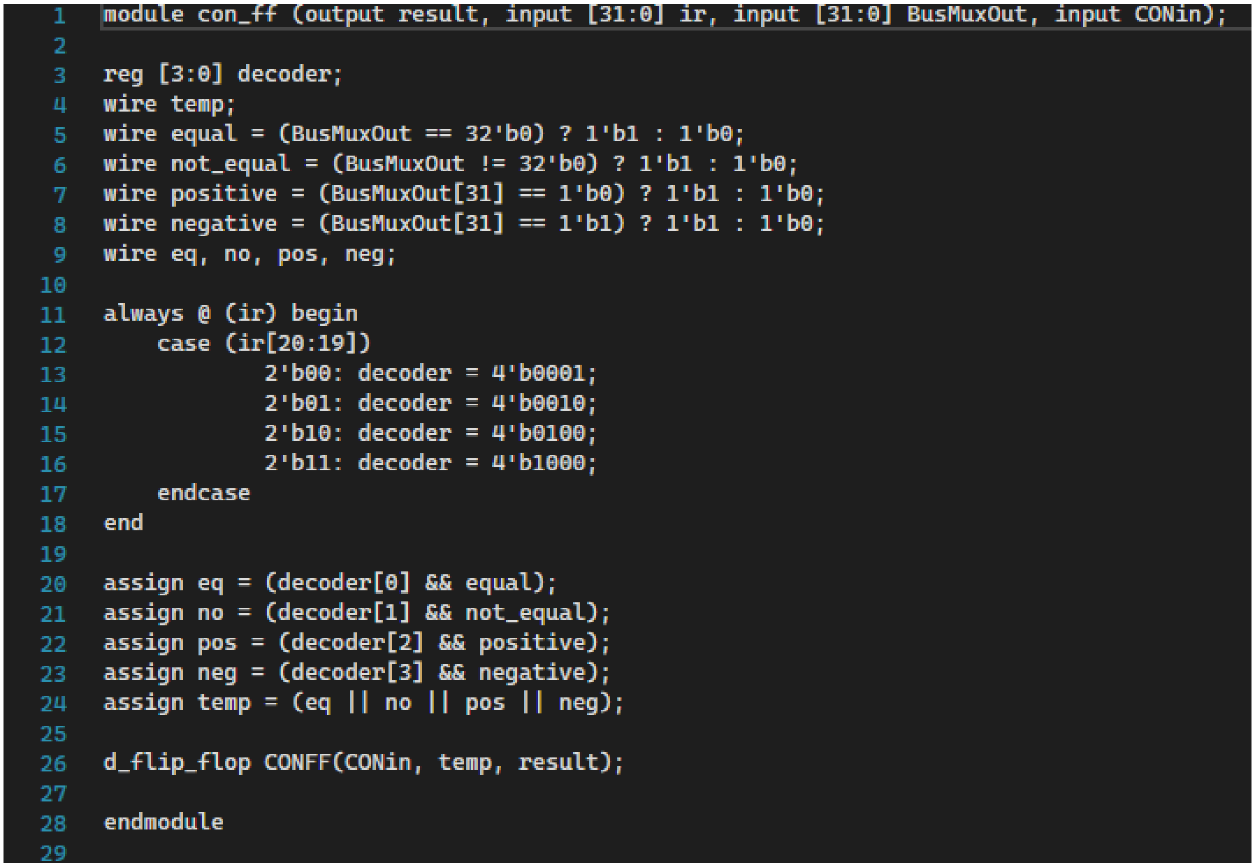
```
module sel_encode(input [31:0] instr, input Gra, Grb, Grc, Rin, Rout, BAout,
                        output [4:0] opcode, output [31:0] C_sign_ext, output R0in,
                        R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in,
                        R11in, R12in, R13in, R14in, R15in, R0out, R1out, R2out, R3out,
                        R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out,
                        R12out, R13out, R14out, R15out, output [3:0] to_decode);

    wire [15:0] InReg, outReg;

    //wire
    assign to_decode = (instr[26:23] & {4{Gra}} | instr[22:19] & {4{Grb}} | instr[18:15] & {4{Grc}});
    reg [15:0] decode_out;
    // 4:16 decoder logic
    always @ (to_decode) begin
                case (to_decode)
                        4'b0000 : decode_out <= 16'b0000_0000_0000_0001;
                        4'b0001 : decode_out <= 16'b0000_0000_0000_0010;
                        4'b0010 : decode_out <= 16'b0000_0000_0000_0100;
                        4'b0011 : decode_out <= 16'b0000_0000_0000_1000;
                        4'b0100 : decode_out <= 16'b0000_0000_0001_0000;
                        4'b0101 : decode_out <= 16'b0000_0000_0010_0000;
                        4'b0110 : decode_out <= 16'b0000_0000_0100_0000;
                        4'b0111 : decode_out <= 16'b0000_0000_1000_0000;
                        4'b1000 : decode_out <= 16'b0000_0001_0000_0000;
                        4'b1001 : decode_out <= 16'b0000_0010_0000_0000;
                        4'b1010 : decode_out <= 16'b0000_0100_0000_0000;
                        4'b1011 : decode_out <= 16'b0000_1000_0000_0000;
                        4'b1100 : decode_out <= 16'b0001_0000_0000_0000;
                        4'b1101 : decode_out <= 16'b0010_0000_0000_0000;
                        4'b1110 : decode_out <= 16'b0100_0000_0000_0000;
                        4'b1111 : decode_out <= 16'b1000_0000_0000_0000;
                endcase
    end
    assign opcode = instr[31:27];
    assign InReg = {16{Rin}} & decode_out;
    assign outReg = ({16{Rout}} | {16{BAout}}) & decode_out;
    assign C_sign_ext = {{13{instr[18]}}, instr[18:0]};

    assign {R15in, R14in, R13in, R12in, R11in, R10in, R9in, R8in,R7in, R6in, R5in, R4in, R3in, R2in, R1in, R0in} = InReg;
    assign {R15out, R14out, R13out, R12out, R11out, R10out, R9out, R8out,R7out, R6out, R5out, R4out, R3out, R2out, R1out, R0out} = outReg;

endmodule
```

Revision to R0



```
module r0 (input clear, clock, enable, BAout, input [31:0]BusMuxOut, output wire [31:0]BusMuxIn);
reg [31:0] q;
wire [31:0] temp_BusMuxIn;

reg_32 this_reg(clear, clock, enable, BusMuxOut, temp_BusMuxIn);

always @(posedge clock) begin
    if(BAout == 1'b1) begin
        q <= 32'b0;
    end else begin
        q <= temp_BusMuxIn;
    end

end

assign BusMuxIn = q;

endmodule
```

CON FF Logic

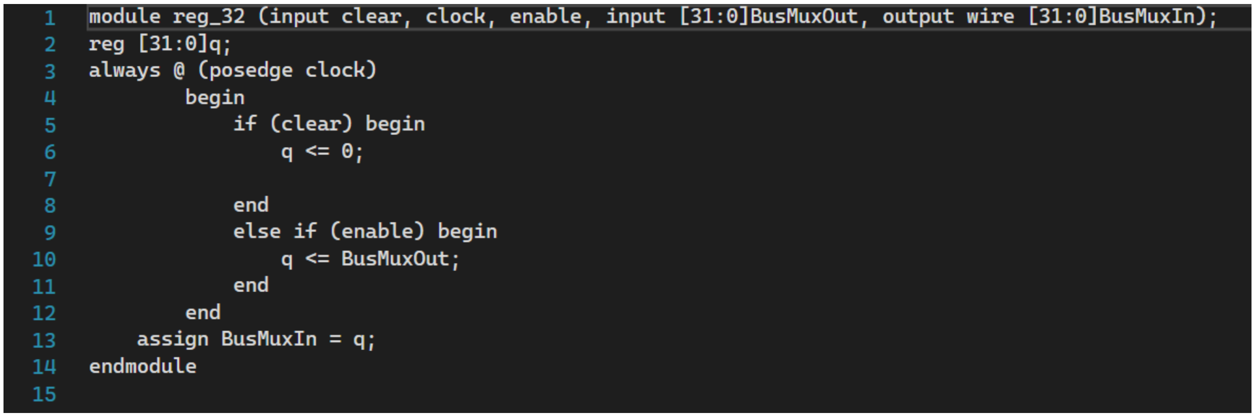


```
module con_ff (output result, input [31:0] ir, input [31:0] BusMuxOut, input CONin);

reg [3:0] decoder;
wire temp;
wire equal = (BusMuxOut == 32'b0) ? 1'b1 : 1'b0;
wire not_equal = (BusMuxOut != 32'b0) ? 1'b1 : 1'b0;
wire positive = (BusMuxOut[31] == 1'b0) ? 1'b1 : 1'b0;
wire negative = (BusMuxOut[31] == 1'b1) ? 1'b1 : 1'b0;
wire eq, no, pos, neg;

always @ (ir) begin
    case (ir[20:19])
            2'b00: decoder = 4'b0001;
            2'b01: decoder = 4'b0010;
            2'b10: decoder = 4'b0100;
            2'b11: decoder = 4'b1000;
    endcase
end

assign eq = (decoder[0] && equal);
assign no = (decoder[1] && not_equal);
assign pos = (decoder[2] && positive);
assign neg = (decoder[3] && negative);
assign temp = (eq || no || pos || neg);

d_flip_flop CONFF(CONin, temp, result);

endmodule
```

Input and Output Ports

Output Port

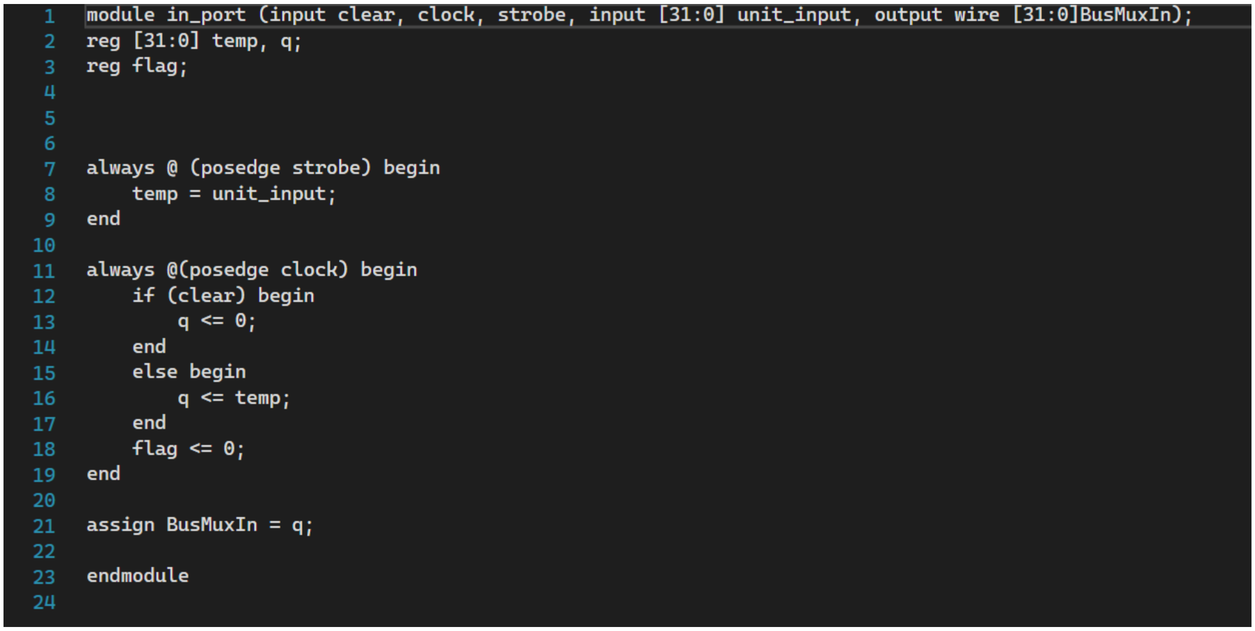
Note that an output port is implemented using a regular register.



```
module reg_32 (input clear, clock, enable, input [31:0]BusMuxOut, output wire [31:0]BusMuxIn);
reg [31:0]q;
always @ (posedge clock)
        begin
            if (clear) begin
                q <= 0;

            end
            else if (enable) begin
                q <= BusMuxOut;
            end
        end
    assign BusMuxIn = q;
endmodule
```

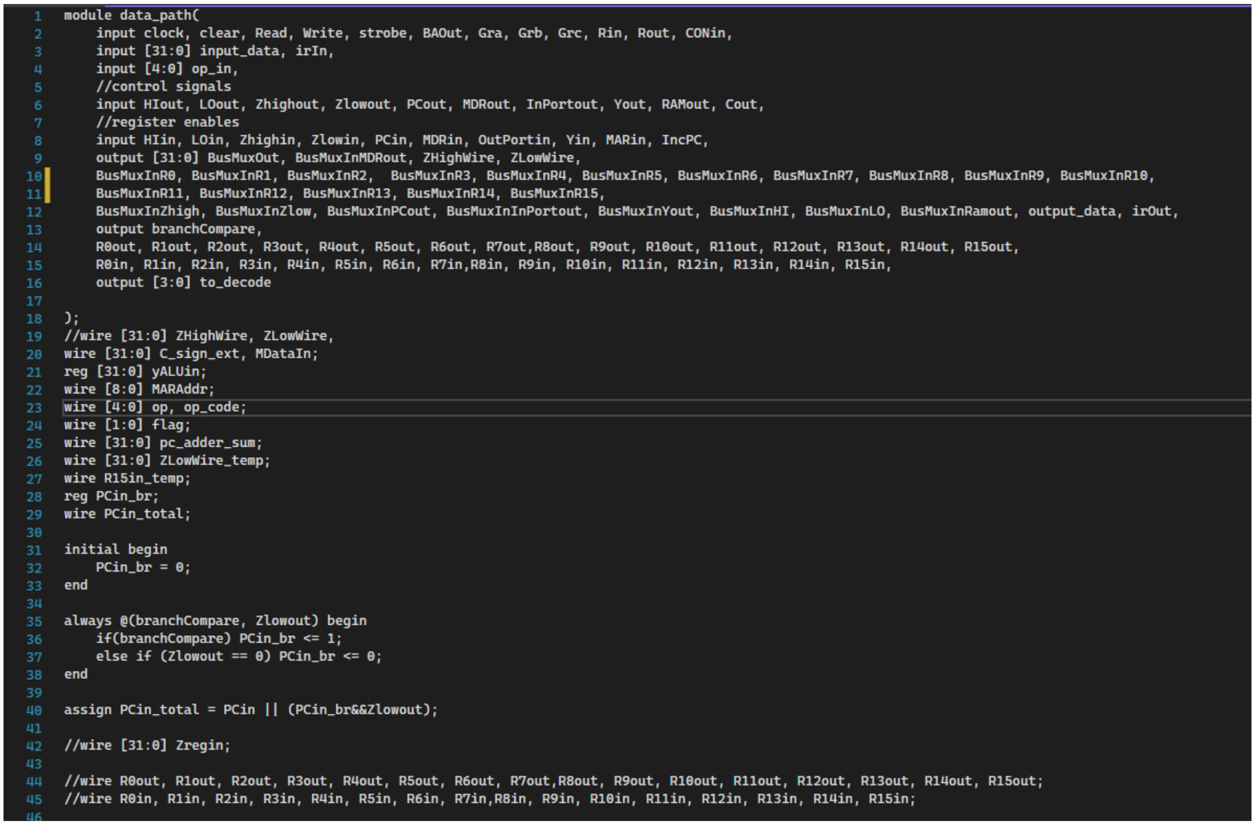
Input Port

Note that we implemented a strobe signal for the Input.

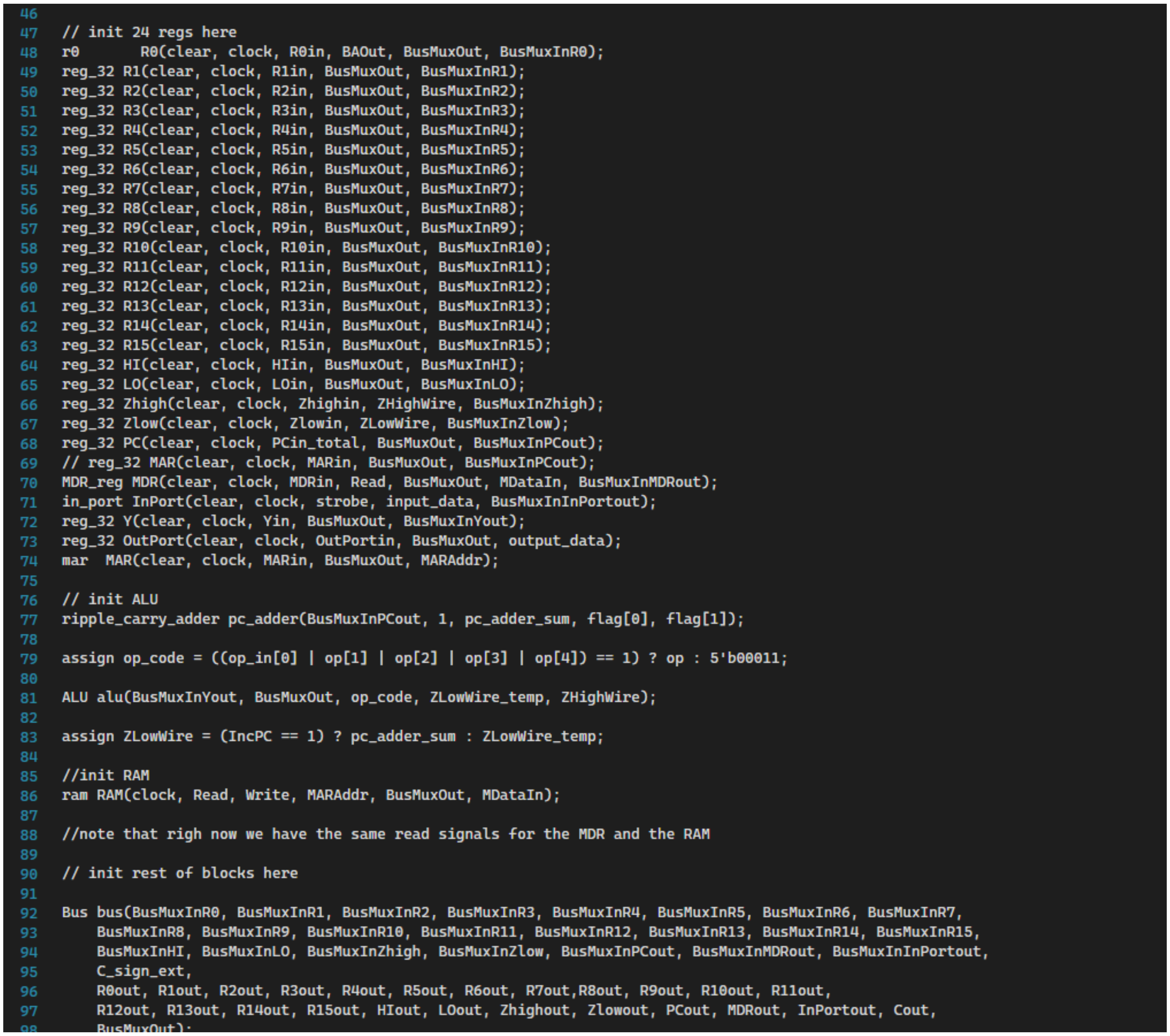


```
module in_port (input clear, clock, strobe, input [31:0] unit_input, output wire [31:0]BusMuxIn);
reg [31:0] temp, q;
reg flag;



always @ (posedge strobe) begin
    temp = unit_input;
end

always @(posedge clock) begin
    if (clear) begin
        q <= 0;
    end
    else begin
        q <= temp;
    end
    flag <= 0;
end

assign BusMuxIn = q;

endmodule

```

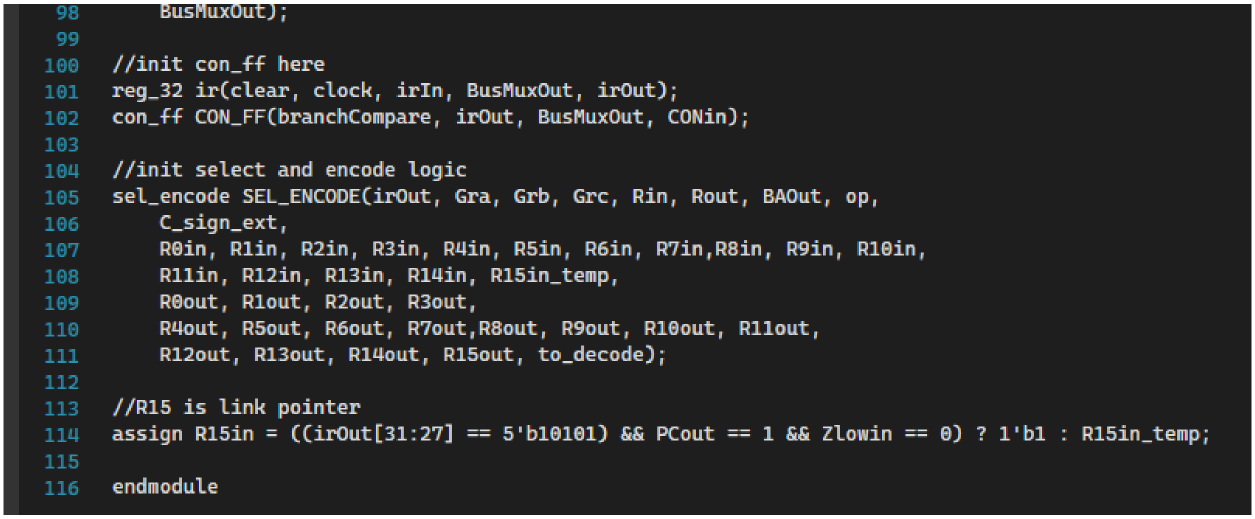
Updated Datapath



```
module data_path(
    input clock, clear, Read, Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, CONin,
    input [31:0] input_data, irIn,
    input [4:0] op_in,
    //control signals
    input HIout, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Yout, RAMout, Cout,
    //register enables
    input HIin, LOin, Zhighin, Zlowin, PCin, MDRin, OutPortin, Yin, MARin, IncPC,
    output [31:0] BusMuxOut, BusMuxInMDRout, ZHighWire, ZLowWire,
    BusMuxInR0, BusMuxInR1, BusMuxInR2,  BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7, BusMuxInR8, BusMuxInR9, BusMuxInR10,
    BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,
    BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output_data, irOut,
    output branchCompare,
    R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out,
    R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in,
    output [3:0] to_decode

);
//wire [31:0] ZHighWire, ZLowWire,
wire [31:0] C_sign_ext, MDataIn;
reg [31:0] yALUin;
wire [8:0] MARAddr;
wire [4:0] op, op_code;
wire [1:0] flag;
wire [31:0] pc_adder_sum;
wire [31:0] ZLowWire_temp;
wire R15in_temp;
reg PCin_br;
wire PCin_total;

initial begin
    PCin_br = 0;
end

always @(branchCompare, Zlowout) begin
    if(branchCompare) PCin_br <= 1;
    else if (Zlowout == 0) PCin_br <= 0;
end

assign PCin_total = PCin || (PCin_br&&Zlowout);

//wire [31:0] Zregin;

//wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out;
//wire R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;
```



```verilog

// init 24 regs here
r0      R0(clear, clock, R0in, BAOut, BusMuxOut, BusMuxInR0);
reg_32 R1(clear, clock, R1in, BusMuxOut, BusMuxInR1);
reg_32 R2(clear, clock, R2in, BusMuxOut, BusMuxInR2);
reg_32 R3(clear, clock, R3in, BusMuxOut, BusMuxInR3);
reg_32 R4(clear, clock, R4in, BusMuxOut, BusMuxInR4);
reg_32 R5(clear, clock, R5in, BusMuxOut, BusMuxInR5);
reg_32 R6(clear, clock, R6in, BusMuxOut, BusMuxInR6);
reg_32 R7(clear, clock, R7in, BusMuxOut, BusMuxInR7);
reg_32 R8(clear, clock, R8in, BusMuxOut, BusMuxInR8);
reg_32 R9(clear, clock, R9in, BusMuxOut, BusMuxInR9);
reg_32 R10(clear, clock, R10in, BusMuxOut, BusMuxInR10);
reg_32 R11(clear, clock, R11in, BusMuxOut, BusMuxInR11);
reg_32 R12(clear, clock, R12in, BusMuxOut, BusMuxInR12);
reg_32 R13(clear, clock, R13in, BusMuxOut, BusMuxInR13);
reg_32 R14(clear, clock, R14in, BusMuxOut, BusMuxInR14);
reg_32 R15(clear, clock, R15in, BusMuxOut, BusMuxInR15);
reg_32 HI(clear, clock, HIin, BusMuxOut, BusMuxInHI);
reg_32 LO(clear, clock, LOin, BusMuxOut, BusMuxInLO);
reg_32 Zhigh(clear, clock, Zhighin, ZHighWire, BusMuxInZhigh);
reg_32 Zlow(clear, clock, Zlowin, ZLowWire, BusMuxInZlow);
reg_32 PC(clear, clock, PCin_total, BusMuxOut, BusMuxInPCout);
// reg_32 MAR(clear, clock, MARin, BusMuxOut, BusMuxInPCout);
MDR_reg MDR(clear, clock, MDRin, Read, BusMuxOut, MDataIn, BusMuxInMDRout);
in_port InPort(clear, clock, strobe, input_data, BusMuxInInPortout);
reg_32 Y(clear, clock, Yin, BusMuxOut, BusMuxInYout);
reg_32 OutPort(clear, clock, OutPortin, BusMuxOut, output_data);
mar  MAR(clear, clock, MARin, BusMuxOut, MARAddr);

// init ALU
ripple_carry_adder pc_adder(BusMuxInPCout, 1, pc_adder_sum, flag[0], flag[1]);

assign op_code = ((op_in[0] | op[1] | op[2] | op[3] | op[4]) == 1) ? op : 5'b00011;

ALU alu(BusMuxInYout, BusMuxOut, op_code, ZLowWire_temp, ZHighWire);

assign ZLowWire = (IncPC == 1) ? pc_adder_sum : ZLowWire_temp;

//init RAM
ram RAM(clock, Read, Write, MARAddr, BusMuxOut, MDataIn);

//note that righ now we have the same read signals for the MDR and the RAM

// init rest of blocks here

Bus bus(BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,
    BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,
    BusMuxInHI, BusMuxInLO, BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInMDRout, BusMuxInInPortout,
    C_sign_ext,
    R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out,
    R12out, R13out, R14out, R15out, HIout, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Cout,
    BusMuxOut);
```



```verilog
    BusMuxOut);

//init con_ff here
reg_32 ir(clear, clock, irIn, BusMuxOut, irOut);
con_ff CON_FF(branchCompare, irOut, BusMuxOut, CONin);

//init select and encode logic
sel_encode SEL_ENCODE(irOut, Gra, Grb, Grc, Rin, Rout, BAOut, op,
    C_sign_ext,
    R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in,
    R11in, R12in, R13in, R14in, R15in_temp,
    R0out, R1out, R2out, R3out,
    R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out,
    R12out, R13out, R14out, R15out, to_decode);

//R15 is link pointer
assign R15in = ((irOut[31:27] == 5'b10101) && PCout == 1 && Zlowin == 0) ? 1'b1 : R15in_temp;

endmodule
```

Testing

1 Load Instructions

Load Testbench

```
`timescale 1ns/10ps

module ld_and_st_tb;

	reg PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout, RAMout, CONin;
	wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out, R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out;
	reg MARin, Zin, PCin, MDRin, IRin, OutPortin, Yin, IncPC, Read, Write, AND, HIin, Loin, ZHighin, Zlowin;
	wire R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in, R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;
	reg Clock, clear, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, Cout;
	wire branchCompare;
	wire [3:0] to_decode;
	reg [31:0] Mdatain, input_data;
	parameter Default = 4'b0000, s1 = 4'b0001, s2 = 4'b0010, s3 = 4'b0011,
					s4 = 4'b0100, s5 = 4'b0101, s6 = 4'b0110, s7 = 4'b0111, s8 = 4'b1000, s9 = 4'b1001, s10 = 4'b1010, s11 = 4'b1011, sclear = 4'b1100, delay = 4'b1101;
	reg [3:0] Present_state = Default;
	reg [4:0] op;
	wire [31:0] BusOut, mdrData, BusMuxInR0, BusMuxInR1, BusMuxInR2,  BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,
					BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,
					BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output_data, irOut,
					ZHighWire, ZLowWire;
	//wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out, R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;

	data_path DUT(Clock, clear, Read, Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, CONin,
	input_data,IRin,
	op,
	HIOut, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Yout, RAMout, Cout,
	HIin,  LOin,  ZHighin,  Zlowin,  PCin,  MDRin,  OutPortin, Yin, MARin, IncPC,
	BusOut, mdrData, ZHighWire, ZLowWire,
	BusMuxInR0, BusMuxInR1, BusMuxInR2,  BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7, BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,
	BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output_data, irOut,
	branchCompare,
	R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out,
	R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in,
	to_decode);


initial begin
	Clock = 1;
end
```

```
always #5 Clock = ~Clock;


always @(negedge Clock) begin// finite state machine; if clock falling-edge so as to be offset from reg clocking
    case (Present_state)
        Default : #40 Present_state = sclear;
        sclear : #40 Present_state = s1;
        s1 : #40 Present_state = s2;
        s2 : #40 Present_state = s3;
        s3 : #40 Present_state = s4;
        s4 : #40 Present_state = s5;
        s5 : #40 Present_state = s6;
        s6  : #40 Present_state = s7;
        s7  : #40 Present_state = s8;
        s8  : #40 Present_state = s1;
        s9 : #40 Present_state = s10;
        s10 : #40 Present_state = s11;
    endcase
end

always @(Present_state) begin // do the required job in each state
    case (Present_state) // assert the required signals in each clock cycle
        Default: begin
            /*PCout <= 0; Zlowout <= 0; MDRout <= 0; // initialize the signals
            R2out <= 0; R3out <= 0; MARin <= 0; Zin <= 0;
            PCin <=0; MDRin <= 0; IRin <= 0; Yin <= 0;
            IncPC <= 0; Read <= 0; AND <= 0;
            R1in <= 0; R2in <= 0; R3in <= 0; Mdatain <= 32'h00000000;*/
            {Write, strobe, BAOut,  Gra, Grb, Grc, Rin, Rout} <= 8'b0;
            {PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout} <= 8'b0;
            {MARin, Zin, PCin, MDRin, IRin, Yin, IncPC, Read, AND, HIin, InPortout, Loin, ZHighin,
Zlowin} <= 13'b0;
            clear<=0;
            Mdatain <= 32'h00000000;
            BAOut <= 0;
        end

        sclear : begin
            clear <= 1;
            #10 clear <= 0;
        end

        s1 : begin //t0
            PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus
            #10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;
        end

        delay: begin
          Read <= 1; MDRin <= 1;
            #10 strobe <= 0;
        end
```

```
s2 : begin //t1
        Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus
        #10 Read <= 1; MDRin <= 1;
        #20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;
end

s3 : begin //t2
        MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus
        #10 IRin <= 1;
        #20 MDRout <= 0; IRin <= 0;
end

s4 : begin //t3
        Grb <= 1; BAOut <= 1;
        #10 Yin <= 1; //see the value of the reg to be added
        #10 Grb <= 0; BAOut <= 0; Yin <= 0;
end

s5 : begin //t4 see C on the bus
        Cout<= 1; ZHighin <= 1; Zlowin <= 1;
        #10 op <= 5'b00011;
        #20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;
end

s6 : begin //t5
        Zlowout <= 1; MARin <= 1; // see the addition result on the bus
        #10 Zlowout <= 0; MARin <= 0;
end

s7 : begin //t6
        MDRin <= 1; Read <= 1;
        #10 MDRin <= 0; Read <= 0;
end


s8 : begin //t7
        Gra <= 1; Rin <= 1; MDRout <= 1; //see the contents of memory on the bus
        #10 Gra <= 0; Rin <= 0; MDRout <= 0;
end

s9 : begin
        Read <= 1; RAMout = 1; //read contents just written to RAM onto the bus
        #10 Read <= 0; RAMout = 0;
end

s10 : begin
        Mdatain <= 32'd50; //this is our compare value for the branch
        Read <= 1; MDRin <= 1;
        #10 Read <= 0; MDRin <= 0;
end

s11 : begin
        IRin <= {11'b0, 2'b10, 19'b0};//branch if positive
        MDRout <= 1; //put compare value on the bus
```

```
            #10 MDRout <= 0;//should see a positive branch compare value
        end

    endcase
end

endmodule
```

Load Immediate Testbench

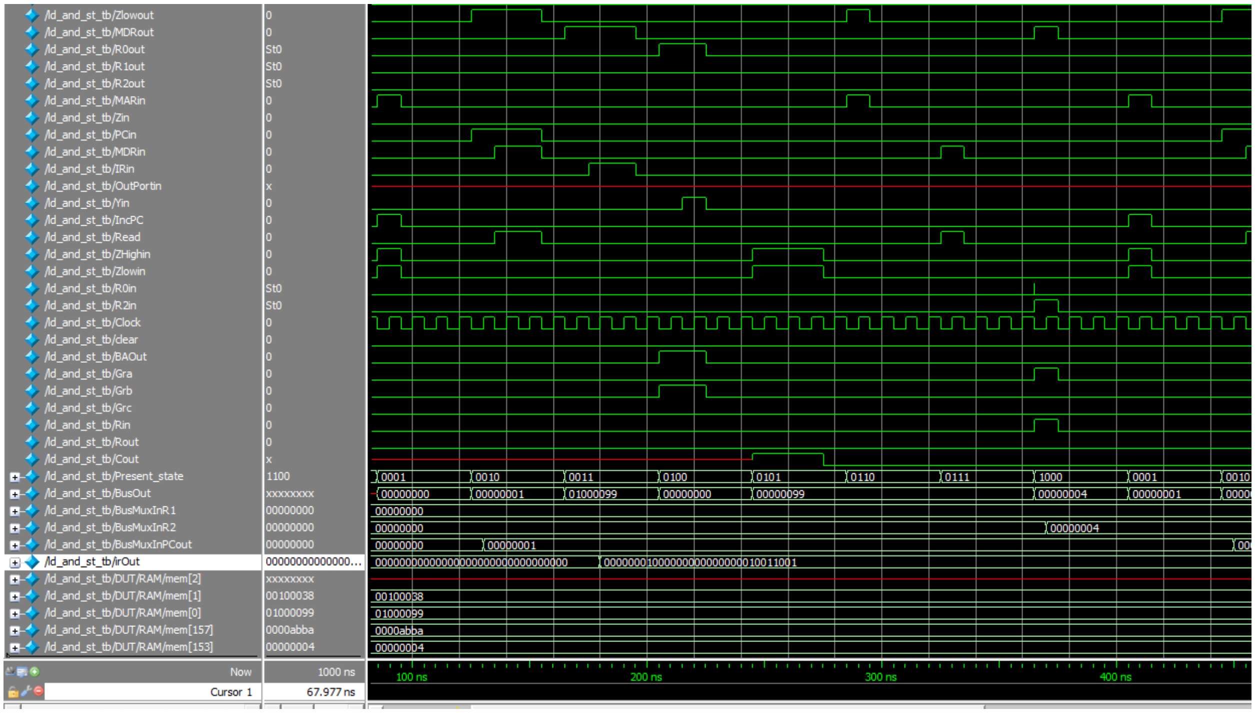
```
`timescale 1ns/10ps

module ldi_tb;

    reg PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout, RAMout, CONin;
    wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out, R8out, R9out, R10out, R11out, R12out, R13out,
R14out, R15out;
    reg MARin, Zin, PCin, MDRin, IRin, OutPortin, Yin, IncPC, Read, Write, AND, HIin, Loin, ZHighin, Zlowin;
    wire R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in, R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;
    reg Clock, clear, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, Cout;
    wire branchCompare;
    wire [3:0] to_decode;
    reg [31:0] Mdatain, input_data;
    parameter Default = 4'b0000, s1 = 4'b0001, s2 = 4'b0010, s3 = 4'b0011,
                s4 = 4'b0100, s5 = 4'b0101, s6 = 4'b0110, s7 = 4'b0111, s8 = 4'b1000, s9 =
4'b1001, s10 = 4'b1010, s11 = 4'b1011, sclear = 4'b1100, delay = 4'b1101;
    reg [3:0] Present_state = Default;
    reg [4:0] op;
    wire [31:0] BusOut, mdrData, BusMuxInR0, BusMuxInR1, BusMuxInR2,  BusMuxInR3, BusMuxInR4,
BusMuxInR5, BusMuxInR6, BusMuxInR7,
                    BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12,
BusMuxInR13, BusMuxInR14, BusMuxInR15,
                    BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout,
BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output_data, irOut,
                    ZHighWire, ZLowWire;
    //wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out,
R13out, R14out, R15out, R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in,
R15in;

    data_path DUT(Clock, clear, Read, Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, CONin,
    input_data,IRin,
    op,
    HIOut, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Yout, RAMout, Cout,
    HIin,  LOin,  ZHighin,  Zlowin,  PCin,  MDRin,  OutPortin, Yin, MARin, IncPC,
    BusOut, mdrData, ZHighWire, ZLowWire,
    BusMuxInR0, BusMuxInR1, BusMuxInR2,  BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,
BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,
    BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI,
BusMuxInLO, BusMuxInRamout, output_data, irOut,
    branchCompare,
    R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out,
R14out, R15out,
    R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in,
    to_decode);
```

```
initial begin
        Clock = 1;
end


always #5 Clock = ~Clock;


always @(negedge Clock) begin// finite state machine; if clock falling-edge so as to be offset from reg clocking
        case (Present_state)
                Default : #40 Present_state = sclear;
                sclear : #40 Present_state = s1;
                s1 : #40 Present_state = s2;
                s2 : #40 Present_state = s3;
                s3 : #40 Present_state = s4;
                s4 : #40 Present_state = s5;
                s5 : #40 Present_state = s6;
                s6      : #40 Present_state = s1;
                s7      : #40 Present_state = s8;
                s8      : #40 Present_state = s1;
                s9 : #40 Present_state = s10;
                s10 : #40 Present_state = s11;
        endcase
end

always @(Present_state) begin // do the required job in each state
        case (Present_state) // assert the required signals in each clock cycle
                Default: begin
                        /*PCout <= 0; Zlowout <= 0; MDRout <= 0; // initialize the signals
                        R2out <= 0; R3out <= 0; MARin <= 0; Zin <= 0;
                        PCin <=0; MDRin <= 0; IRin <= 0; Yin <= 0;
                        IncPC <= 0; Read <= 0; AND <= 0;
                        R1in <= 0; R2in <= 0; R3in <= 0; Mdatain <= 32'h00000000;*/
                        {Write, strobe, BAOut,  Gra, Grb, Grc, Rin, Rout} <= 8'b0;
                        {PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout} <= 8'b0;
                        {MARin, Zin, PCin, MDRin, IRin, Yin, IncPC, Read, AND, HIin, InPortout, Loin, ZHighin,
Zlowin} <= 13'b0;
                        clear<=0;
                        Mdatain <= 32'h00000000;
                        BAOut <= 0;
                end

                sclear : begin
                        clear <= 1;
                        #10 clear <= 0;
                end

                s1 : begin //t0
                        PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus
                        #10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;
                end

                delay: begin
```

```
  Read <= 1; MDRin <= 1;
            #10 strobe <= 0;
end

s2 : begin //t1
            Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus
            #10 Read <= 1; MDRin <= 1;
            #20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;
end

s3 : begin //t2
            MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus
            #10 IRin <= 1;
            #20 MDRout <= 0; IRin <= 0;
end

s4 : begin //t3
            Grb <= 1; BAOut <= 1;
            #10 Yin <= 1; //see the value of the reg to be added
            #10 Grb <= 0; BAOut <= 0; Yin <= 0;
end

s5 : begin //t4 see C on the bus
            Cout<= 1; ZHighin <= 1; Zlowin <= 1;
            #10 op <= 5'b00011;
            #20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;
end

s6 : begin //t5
            Zlowout <= 1; Gra <= 1; Rin <= 1;// see the addition result on the bus
            #10 Zlowout <= 0; Gra <= 0; Rin <= 0;
end

s7 : begin //t6
            MDRin <= 1; Read <= 1;
            #10 MDRin <= 0; Read <= 0;
end


s8 : begin //t7
            Gra <= 1; Rin <= 1; MDRout <= 1; //see the contents of memory on the bus
            #10 Gra <= 0; Rin <= 0; MDRout <= 0;
end

s9 : begin
            Read <= 1; RAMout = 1; //read contents just written to RAM onto the bus
            #10 Read <= 0; RAMout = 0;
end

s10 : begin
            Mdatain <= 32'd50; //this is our compare value for the branch
            Read <= 1; MDRin <= 1;
            #10 Read <= 0; MDRin <= 0;
end
```

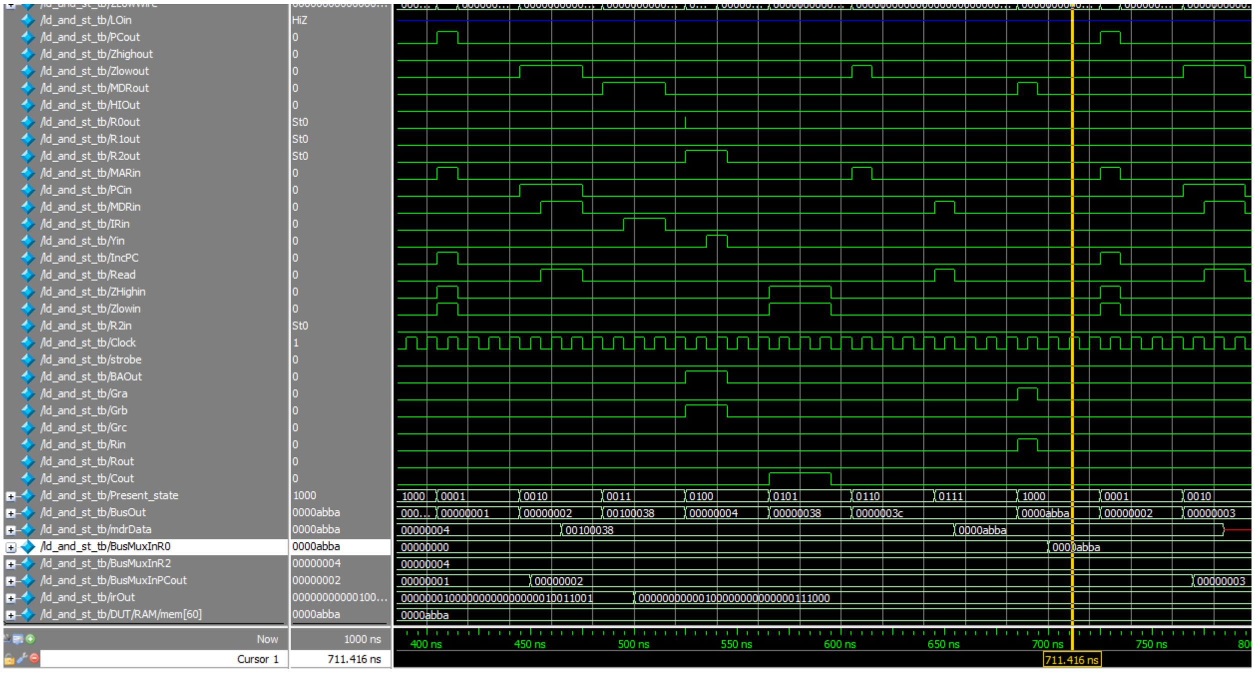
```
                s11 : begin
                        IRin <= {11'b0, 2'b10, 19'b0};//branch if positive
                        MDRout <= 1; //put compare value on the bus
                        #10 MDRout <= 0;//should see a positive branch compare value
                end

        endcase
end

endmodule
```

ld R2, 0x95



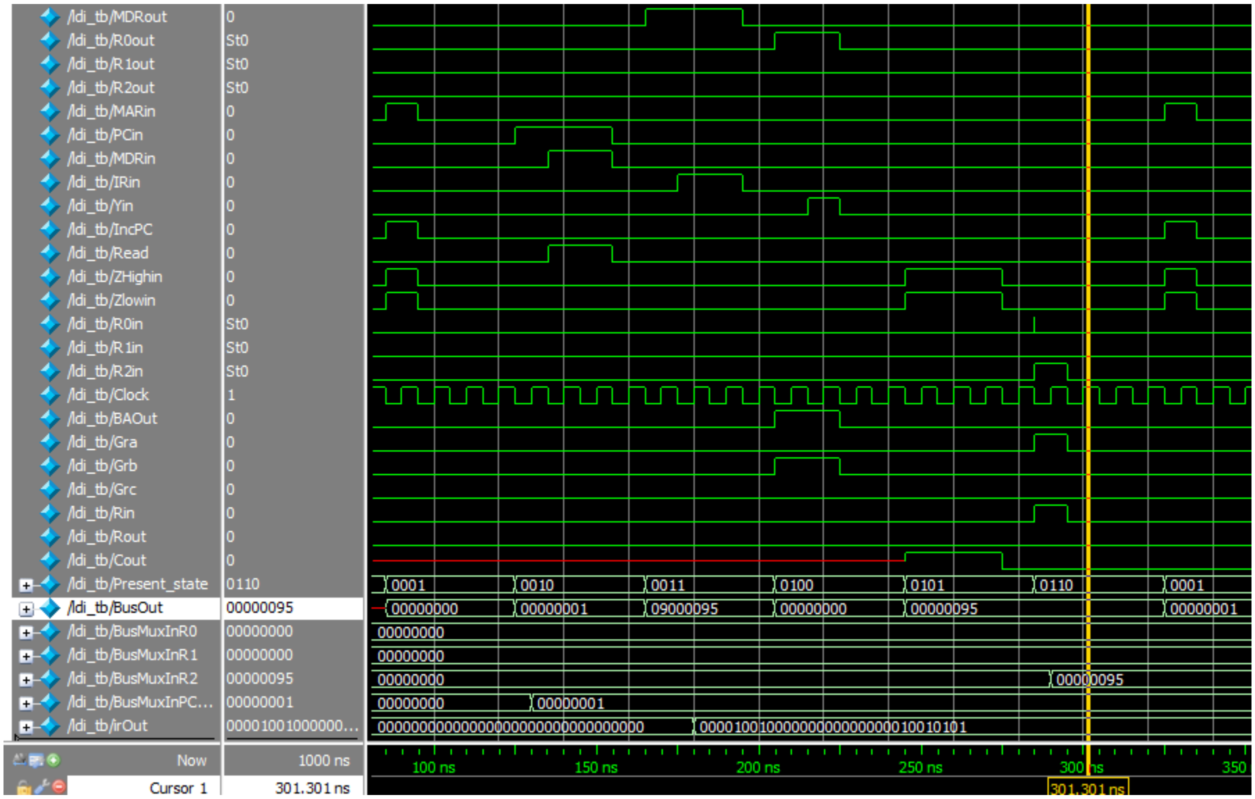
/ld_and_st_tb/Zlowout	0
/ld_and_st_tb/MDRout	0
/ld_and_st_tb/R0out	St0
/ld_and_st_tb/R1out	St0
/ld_and_st_tb/R2out	St0
/ld_and_st_tb/MARin	0
/ld_and_st_tb/Zin	0
/ld_and_st_tb/PCin	0
/ld_and_st_tb/MDRin	0
/ld_and_st_tb/IRin	0
/ld_and_st_tb/OutPortin	x
/ld_and_st_tb/Yin	0
/ld_and_st_tb/IncPC	0
/ld_and_st_tb/Read	0
/ld_and_st_tb/ZHighin	0
/ld_and_st_tb/Zlowin	0
/ld_and_st_tb/R0in	St0
/ld_and_st_tb/R2in	St0
/ld_and_st_tb/Clock	0
/ld_and_st_tb/clear	0
/ld_and_st_tb/BAOut	0
/ld_and_st_tb/Gra	0
/ld_and_st_tb/Grb	0
/ld_and_st_tb/Grc	0
/ld_and_st_tb/Rin	0
/ld_and_st_tb/Rout	0
/ld_and_st_tb/Cout	x
/ld_and_st_tb/Present_state	1100
/ld_and_st_tb/BusOut	xxxxxxxx
/ld_and_st_tb/BusMuxInR1	00000000
/ld_and_st_tb/BusMuxInR2	00000000
/ld_and_st_tb/BusMuxInPCout	00000000
/ld_and_st_tb/irOut	00000000000000...
/ld_and_st_tb/DUT/RAM/mem[2]	xxxxxxxx
/ld_and_st_tb/DUT/RAM/mem[1]	00100038
/ld_and_st_tb/DUT/RAM/mem[0]	01000099
/ld_and_st_tb/DUT/RAM/mem[157]	0000abba
/ld_and_st_tb/DUT/RAM/mem[153]	00000004
Now	1000 ns
Cursor 1	67.977 ns

ld R0, 0x38(R2)



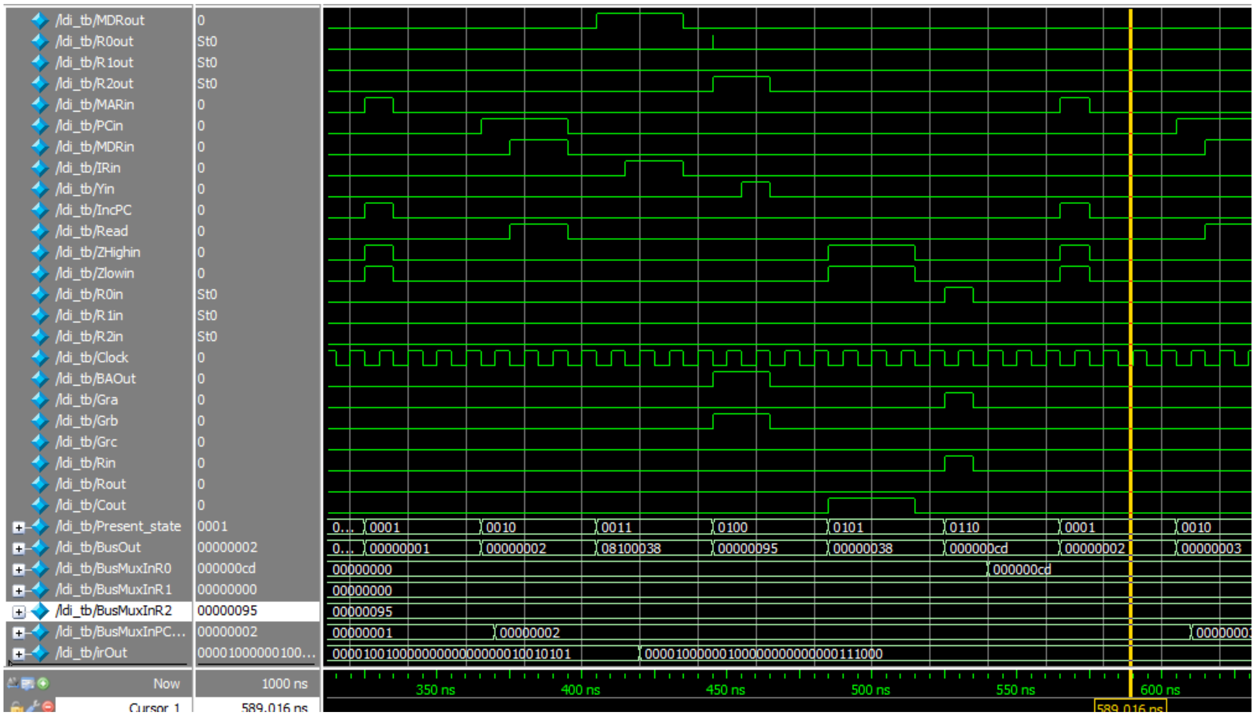
/ld_and_st_tb/LOin	HiZ
/ld_and_st_tb/PCout	0
/ld_and_st_tb/Zhighout	0
/ld_and_st_tb/Zlowout	0
/ld_and_st_tb/MDRout	0
/ld_and_st_tb/HIOut	0
/ld_and_st_tb/R0out	St0
/ld_and_st_tb/R1out	St0
/ld_and_st_tb/R2out	St0
/ld_and_st_tb/MARin	0
/ld_and_st_tb/PCin	0
/ld_and_st_tb/MDRin	0
/ld_and_st_tb/IRin	0
/ld_and_st_tb/Yin	0
/ld_and_st_tb/IncPC	0
/ld_and_st_tb/Read	0
/ld_and_st_tb/ZHighin	0
/ld_and_st_tb/Zlowin	0
/ld_and_st_tb/R2in	St0
/ld_and_st_tb/Clock	1
/ld_and_st_tb/strobe	0
/ld_and_st_tb/BAOut	0
/ld_and_st_tb/Gra	0
/ld_and_st_tb/Grb	0
/ld_and_st_tb/Grc	0
/ld_and_st_tb/Rin	0
/ld_and_st_tb/Rout	0
/ld_and_st_tb/Cout	0
/ld_and_st_tb/Present_state	1000
/ld_and_st_tb/BusOut	0000abba
/ld_and_st_tb/mdrData	0000abba
/ld_and_st_tb/BusMuxInR0	0000abba
/ld_and_st_tb/BusMuxInR2	00000004
/ld_and_st_tb/BusMuxInPCout	00000002
/ld_and_st_tb/irOut	00000000000100...
/ld_and_st_tb/DUT/RAM/mem[60]	0000abba
Now	1000 ns
Cursor 1	711.416 ns

ldi R2, 0x95



/ldi_tb/MDRout	0
/ldi_tb/R0out	St0
/ldi_tb/R1out	St0
/ldi_tb/R2out	St0
/ldi_tb/MARin	0
/ldi_tb/PCin	0
/ldi_tb/MDRin	0
/ldi_tb/IRin	0
/ldi_tb/Yin	0
/ldi_tb/IncPC	0
/ldi_tb/Read	0
/ldi_tb/ZHighin	0
/ldi_tb/Zlowin	0
/ldi_tb/R0in	St0
/ldi_tb/R1in	St0
/ldi_tb/R2in	St0
/ldi_tb/Clock	1
/ldi_tb/BAOut	0
/ldi_tb/Gra	0
/ldi_tb/Grb	0
/ldi_tb/Grc	0
/ldi_tb/Rin	0
/ldi_tb/Rout	0
/ldi_tb/Cout	0
/ldi_tb/Present_state	0110
/ldi_tb/BusOut	00000095
/ldi_tb/BusMuxInR0	00000000
/ldi_tb/BusMuxInR1	00000000
/ldi_tb/BusMuxInR2	00000095
/ldi_tb/BusMuxInPC...	00000001
/ldi_tb/irOut	0000100100000...
Now	1000 ns
Cursor 1	301.301 ns

ldi R0, 0x38(R2)



/ldi_tb/MDRout	0
/ldi_tb/R0out	St0
/ldi_tb/R1out	St0
/ldi_tb/R2out	St0
/ldi_tb/MARin	0
/ldi_tb/PCin	0
/ldi_tb/MDRin	0
/ldi_tb/IRin	0
/ldi_tb/Yin	0
/ldi_tb/IncPC	0
/ldi_tb/Read	0
/ldi_tb/ZHighin	0
/ldi_tb/Zlowin	0
/ldi_tb/R0in	St0
/ldi_tb/R1in	St0
/ldi_tb/R2in	St0
/ldi_tb/Clock	0
/ldi_tb/BAOut	0
/ldi_tb/Gra	0
/ldi_tb/Grb	0
/ldi_tb/Grc	0
/ldi_tb/Rin	0
/ldi_tb/Rout	0
/ldi_tb/Cout	0
/ldi_tb/Present_state	0001
/ldi_tb/BusOut	00000002
/ldi_tb/BusMuxInR0	000000cd
/ldi_tb/BusMuxInR1	00000000
/ldi_tb/BusMuxInR2	00000095
/ldi_tb/BusMuxInPC...	00000002
/ldi_tb/irOut	00001000000100...
Now	1000 ns
Cursor 1	589.016 ns

2 Store Instructions

Testbench

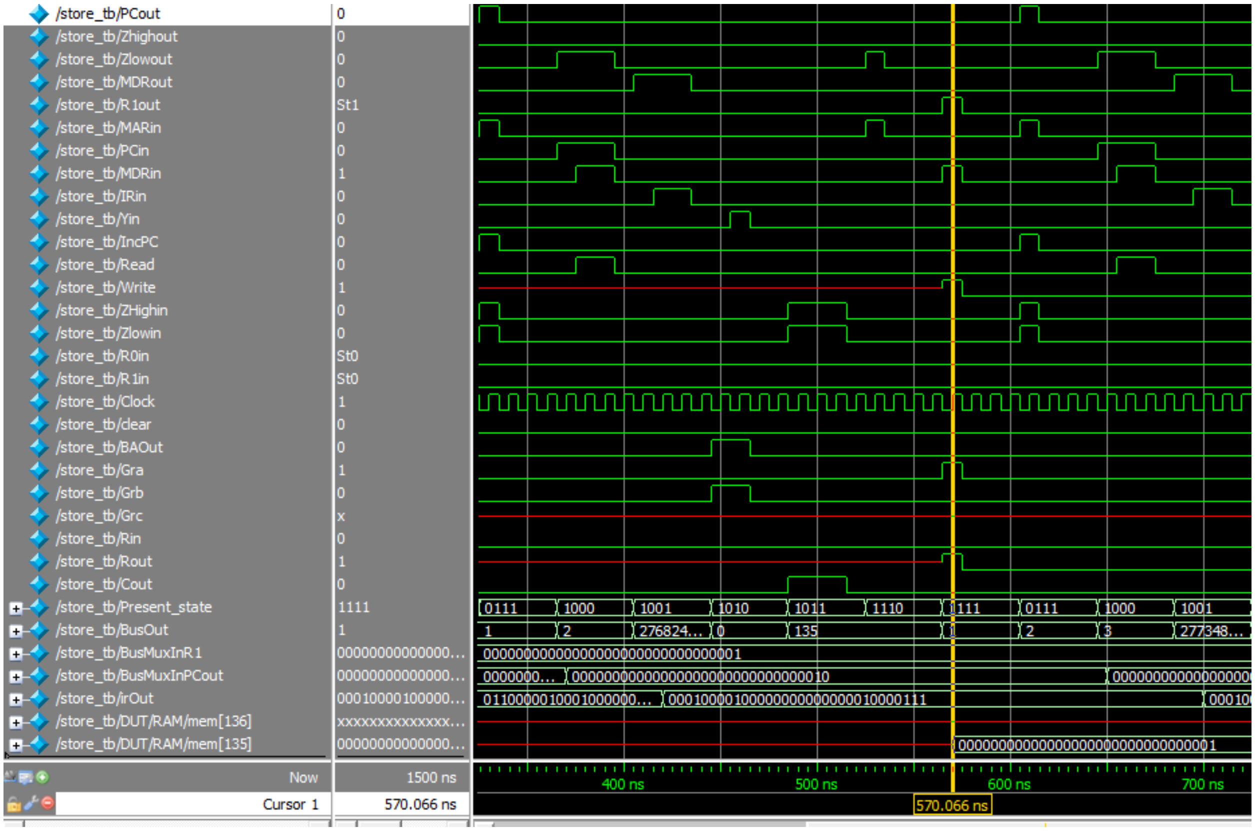
```
`timescale 1ns/10ps

module store_tb;

        reg PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout, RAMout, CONin;
        wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out, R8out, R9out, R10out, R11out, R12out, R13out, 
R14out, R15out;
        reg MARin, Zin, PCin, MDRin, IRin, OutPortin, Yin, IncPC, Read, Write, AND, HIin, Loin, ZHighin, Zlowin;
        wire R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in, R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;
        reg Clock, clear, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, Cout;
        wire branchCompare;
        wire [3:0] to_decode;
        reg [31:0] Mdatain, input_data;
        parameter Default = 4'b0000, s1 = 4'b0001, s2 = 4'b0010, s3 = 4'b0011,
                                s4 = 4'b0100, s5 = 4'b0101, s6 = 4'b0110, s7 = 4'b0111, s8 = 4'b1000, s9 = 
4'b1001, s10 = 4'b1010, s11 = 4'b1011, sclear = 4'b1100, delay = 4'b1101, s12 = 4'b1110, s13 = 4'b1111;
        reg [3:0] Present_state = Default;
        reg [4:0] op;
        wire [31:0] BusOut, mdrData, BusMuxInR0, BusMuxInR1, BusMuxInR2,  BusMuxInR3, BusMuxInR4, 
BusMuxInR5, BusMuxInR6, BusMuxInR7,
                                        BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, 
BusMuxInR13, BusMuxInR14, BusMuxInR15,
                                        BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, 
BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output_data, irOut,
                                        ZHighWire, ZLowWire;
```

```
    //wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out,
R13out, R14out, R15out, R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in,
R15in;

    data_path DUT(Clock, clear, Read, Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, CONin,
    input_data,IRin,
    op,
    HIOut, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Yout, RAMout, Cout,
    HIin,  LOin,  ZHighin,  Zlowin,  PCin,  MDRin,  OutPortin, Yin, MARin, IncPC,
    BusOut, mdrData, ZHighWire, ZLowWire,
    BusMuxInR0, BusMuxInR1, BusMuxInR2,  BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,
BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,
    BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI,
BusMuxInLO, BusMuxInRamout, output_data, irOut,
    branchCompare,
    R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out,
R14out, R15out,
    R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in,
    to_decode);


initial begin
    Clock = 1;
end


always #5 Clock = ~Clock;


always @(negedge Clock) begin// finite state machine; if clock falling-edge so as to be offset from reg clocking
    case (Present_state)
        Default : #40 Present_state = sclear;
        sclear : #40 Present_state = s1;
        s1 : #40 Present_state = s2;
        s2 : #40 Present_state = s3;
        s3 : #40 Present_state = s4;
        s4 : #40 Present_state = s5;
        s5 : #40 Present_state = s6;
        s6      : #40 Present_state = s7;
        s7      : #40 Present_state = s8;
        s8      : #40 Present_state = s9;
        s9 : #40 Present_state = s10;
        s10 : #40 Present_state = s11;
        s11 : #40 Present_state = s12;
        s12 : #40 Present_state = s13;
        s13 : #40 Present_state = s7;
    endcase
end

always @(Present_state) begin // do the required job in each state
    case (Present_state) // assert the required signals in each clock cycle
        Default: begin
            /*PCout <= 0; Zlowout <= 0; MDRout <= 0; // initialize the signals
            R2out <= 0; R3out <= 0; MARin <= 0; Zin <= 0;
```

```
                PCin <=0; MDRin <= 0; IRin <= 0; Yin <= 0;
                IncPC <= 0; Read <= 0; AND <= 0;
                R1in <= 0; R2in <= 0; R3in <= 0; Mdatain <= 32'h00000000;*/
                {Write, strobe, BAOut,  Gra, Grb, Grc, Rin, Rout} <= 8'b0;
                {PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout} <= 8'b0;
                {MARin, Zin, PCin, MDRin, IRin, Yin, IncPC, Read, AND, HIin, InPortout, Loin, ZHighin, Zlowin} <= 13'b0;
                clear<=0;
                Mdatain <= 32'h00000000;
                BAOut <= 0;
        end

        sclear : begin
                clear <= 1;
                #10 clear <= 0;
        end


        s1 : begin //t0
                PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus
                #10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;
        end

        s2 : begin //t1
                Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus
                #10 Read <= 1; MDRin <= 1;
                #20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;
        end

        s3 : begin //t2
                MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus
                #10 IRin <= 1;
                #20 MDRout <= 0; IRin <= 0;
        end

        s4 : begin //t3
                Grb <= 1; BAOut <= 1;
                #10 Yin <= 1; //see the value of the reg to be added
                #10 Grb <= 0; BAOut <= 0; Yin <= 0;
        end

        s5 : begin //t4 see C on the bus
                Cout<= 1; ZHighin <= 1; Zlowin <= 1;
                #10 op <= 5'b00011;
                #20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;
        end

        s6 : begin //t5
                Zlowout <= 1; Gra <= 1; Rin <= 1;// see the addition result on the bus
                #10 Zlowout <= 0; Gra <= 0; Rin <= 0;
        end

        s7 : begin //t0
                PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus
```

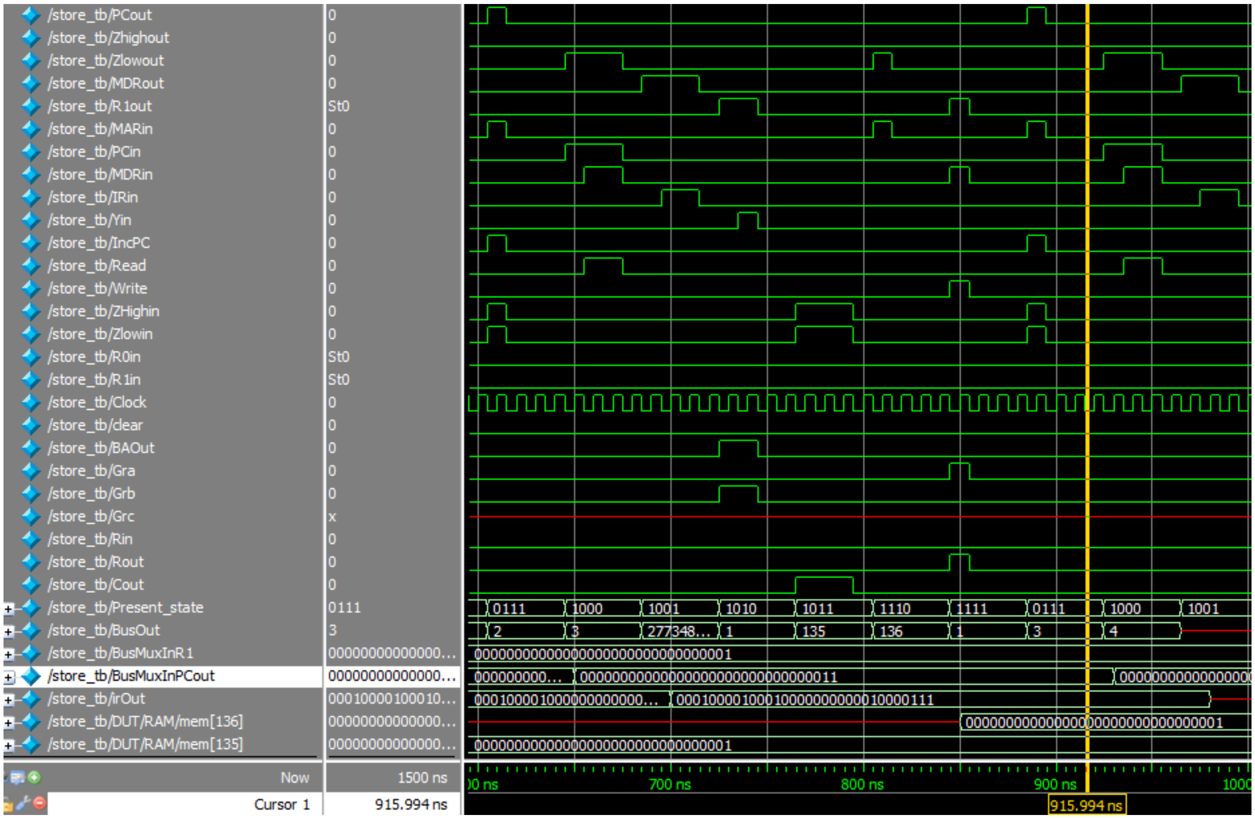
```
            #10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;
        end

        s8 : begin //t1
            Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus
            #10 Read <= 1; MDRin <= 1;
            #20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;
        end

        s9 : begin //t2
            MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus
            #10 IRin <= 1;
            #20 MDRout <= 0; IRin <= 0;
        end

        s10 : begin //t3
            Grb <= 1; BAOut <= 1;
            #10 Yin <= 1; //see the value of the reg to be added
            #10 Grb <= 0; BAOut <= 0; Yin <= 0;
        end

        s11 : begin //t4 see C on the bus
            Cout<= 1; ZHighin <= 1; Zlowin <= 1;
            #10 op <= 5'b00011;
            #20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;
        end

        s12 : begin //t5
            Zlowout <= 1; MARin <= 1; // see the addition result on the bus
            #10 Zlowout <= 0; MARin <= 0;
        end

        s13 : begin //t6
            MDRin <= 1; Write <= 1; Rout <= 1; Gra <= 1;
            #10 MDRin <= 0; Write <= 0; Rout <= 0; Gra <= 0;
        end


    endcase
end

endmodule
```

st 0x87, R1



```
/store_tb/PCout                 0
/store_tb/Zhighout              0
/store_tb/Zlowout               0
/store_tb/MDRout                0
/store_tb/R1out                 St1
/store_tb/MARin                 0
/store_tb/PCin                  0
/store_tb/MDRin                 1
/store_tb/IRin                  0
/store_tb/Yin                   0
/store_tb/IncPC                 0
/store_tb/Read                  0
/store_tb/Write                 1
/store_tb/ZHighin               0
/store_tb/Zlowin                0
/store_tb/R0in                  St0
/store_tb/R1in                  St0
/store_tb/Clock                 1
/store_tb/clear                 0
/store_tb/BAOut                 0
/store_tb/Gra                   1
/store_tb/Grb                   0
/store_tb/Grc                   x
/store_tb/Rin                   0
/store_tb/Rout                  1
/store_tb/Cout                  0
/store_tb/Present_state         1111
/store_tb/BusOut                1
/store_tb/BusMuxInR1            0000000000000...
/store_tb/BusMuxInPCout         0000000000000...
/store_tb/irOut                 000100001000000...
/store_tb/DUT/RAM/mem[136]      xxxxxxxxxxxxxx...
/store_tb/DUT/RAM/mem[135]      0000000000000...
Now                             1500 ns
Cursor 1                        570.066 ns
```

st 0x87(R1), R1

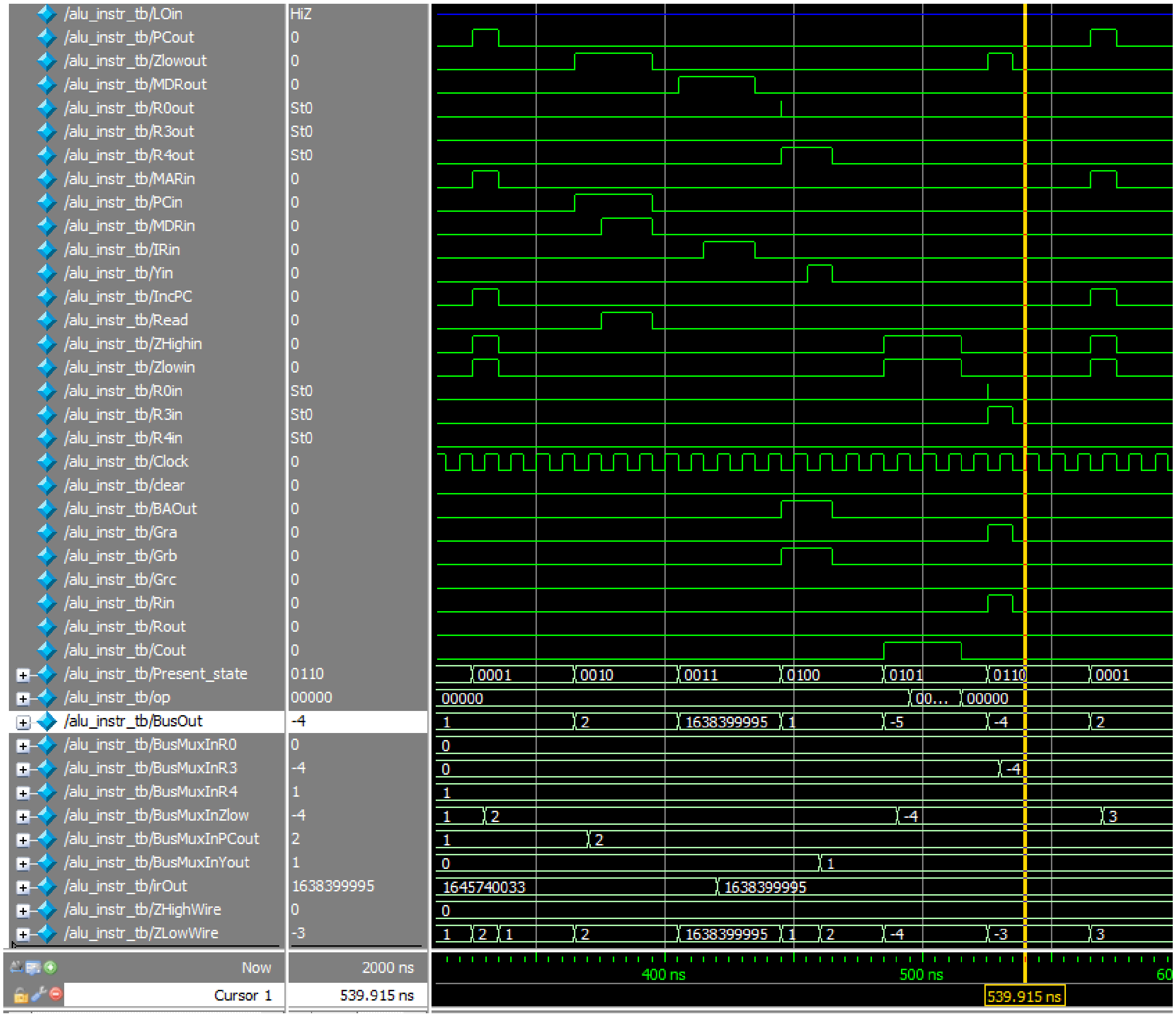


3 ALU Immediate Instructions

Testbench

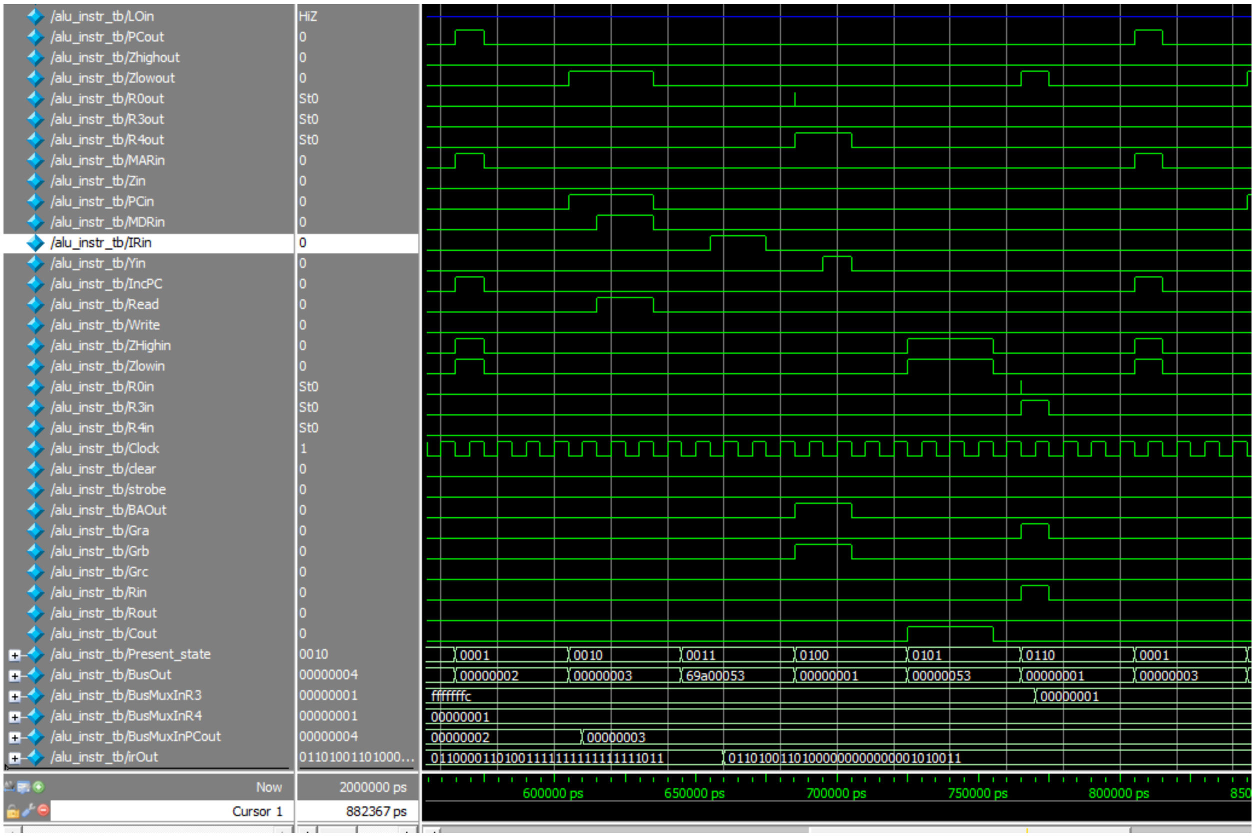
```
`timescale 1ns/10ps

module alu_instr_tb;

        reg PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout, RAMout, CONin;
        wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out, R8out, R9out, R10out, R11out, R12out, R13out,
R14out, R15out;
        reg MARin, Zin, PCin, MDRin, IRin, OutPortin, Yin, IncPC, Read, Write, AND, HIin, Loin, ZHighin, Zlowin;
        wire R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in, R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;
        reg Clock, clear, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, Cout;
        wire branchCompare;
        wire [3:0] to_decode;
        reg [31:0] Mdatain, input_data;
        parameter Default = 4'b0000, s1 = 4'b0001, s2 = 4'b0010, s3 = 4'b0011,
                                s4 = 4'b0100, s5 = 4'b0101, s6 = 4'b0110, s7 = 4'b0111, s8 = 4'b1000, s9 =
4'b1001, s10 = 4'b1010, s11 = 4'b1011, sclear = 4'b1100, delay = 4'b1101;
        reg [3:0] Present_state = Default;
        reg [4:0] op;
        wire [31:0] BusOut, mdrData, BusMuxInR0, BusMuxInR1, BusMuxInR2,  BusMuxInR3, BusMuxInR4,
BusMuxInR5, BusMuxInR6, BusMuxInR7,
                                BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12,
BusMuxInR13, BusMuxInR14, BusMuxInR15,
                                        BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout,
BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output_data, irOut,
                                        ZHighWire, ZLowWire;
        //wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out,
R13out, R14out, R15out, R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in,
R15in;

        data_path DUT(Clock, clear, Read, Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, CONin,
        input_data,IRin,
        op,
        HIOut, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Yout, RAMout, Cout,
        HIin,  LOin,  ZHighin,  Zlowin,  PCin,  MDRin,  OutPortin, Yin, MARin, IncPC,
        BusOut, mdrData, ZHighWire, ZLowWire,
        BusMuxInR0, BusMuxInR1, BusMuxInR2,  BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,
BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,
        BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI,
BusMuxInLO, BusMuxInRamout, output_data, irOut,
        branchCompare,
        R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out,
R14out, R15out,
        R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in,
        to_decode);


initial begin
        Clock = 1;
end


always #5 Clock = ~Clock;


always @(negedge Clock) begin// finite state machine; if clock falling-edge so as to be offset from reg clocking
        case (Present_state)
                Default : #40 Present_state = sclear;
                sclear : #40 Present_state = s1;
                s1 : #40 Present_state = s2;
                s2 : #40 Present_state = s3;
                s3 : #40 Present_state = s4;
                s4 : #40 Present_state = s5;
                s5 : #40 Present_state = s6;
                s6      : #40 Present_state = s1;
                s7      : #40 Present_state = s8;
                s8      : #40 Present_state = s1;
                s9 : #40 Present_state = s10;
                s10 : #40 Present_state = s11;
        endcase
end

always @(Present_state) begin // do the required job in each state
        case (Present_state) // assert the required signals in each clock cycle
                Default: begin
                        /*PCout <= 0; Zlowout <= 0; MDRout <= 0; // initialize the signals
                        R2out <= 0; R3out <= 0; MARin <= 0; Zin <= 0;
                        PCin <=0; MDRin <= 0; IRin <= 0; Yin <= 0;
                        IncPC <= 0; Read <= 0; AND <= 0;
                        R1in <= 0; R2in <= 0; R3in <= 0; Mdatain <= 32'h00000000;*/
                        {Write, strobe, BAOut,  Gra, Grb, Grc, Rin, Rout} <= 8'b0;
                        {PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout} <= 8'b0;
                        {MARin, Zin, PCin, MDRin, IRin, Yin, IncPC, Read, AND, HIin, InPortout, Loin, ZHighin, Zlowin} <= 13'b0;
                        clear<=0;
                        Mdatain <= 32'h00000000;
                        BAOut <= 0;
                end

                sclear : begin
                        clear <= 1;
                        #10 clear <= 0;
                end

                s1 : begin //t0
                        PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus
                        #10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;
                end

                delay: begin
                  Read <= 1; MDRin <= 1;
                        #10 strobe <= 0;
                end

                s2 : begin //t1
                        Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus
                        #10 Read <= 1; MDRin <= 1;
                        #20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;
                end

                s3 : begin //t2
                        MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus
                        #10 IRin <= 1;
                        #20 MDRout <= 0; IRin <= 0;
                end

                s4 : begin //t3
                        Grb <= 1; BAOut <= 1;
                        #10 Yin <= 1; //see the value of the reg to be added
                        #10 Grb <= 0; BAOut <= 0; Yin <= 0;
                end

                s5 : begin //t4 see C on the bus
                        Cout<= 1; ZHighin <= 1; Zlowin <= 1;
                        #10 op <= 5'b00011;
                        #20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;
                end

                s6 : begin //t5
                        Zlowout <= 1; Gra <= 1; Rin <= 1;// see the addition result on the bus
                        #10 Zlowout <= 0; Gra <= 0; Rin <= 0;
        end

        s7 : begin //t6
            MDRin <= 1; Read <= 1;
            #10 MDRin <= 0; Read <= 0;
        end


        s8 : begin //t7
            Gra <= 1; Rin <= 1; MDRout <= 1; //see the contents of memory on the bus
            #10 Gra <= 0; Rin <= 0; MDRout <= 0;
        end

        s9 : begin
            Read <= 1; RAMout = 1; //read contents just written to RAM onto the bus
            #10 Read <= 0; RAMout = 0;
        end

        s10 : begin
            Mdatain <= 32'd50; //this is our compare value for the branch
            Read <= 1; MDRin <= 1;
            #10 Read <= 0; MDRin <= 0;
        end

        s11 : begin
            IRin <= {11'b0, 2'b10, 19'b0};//branch if positive
            MDRout <= 1; //put compare value on the bus
            #10 MDRout <= 0;//should see a positive branch compare value
        end

    endcase
end

endmodule
```

addi R3, R4,-5



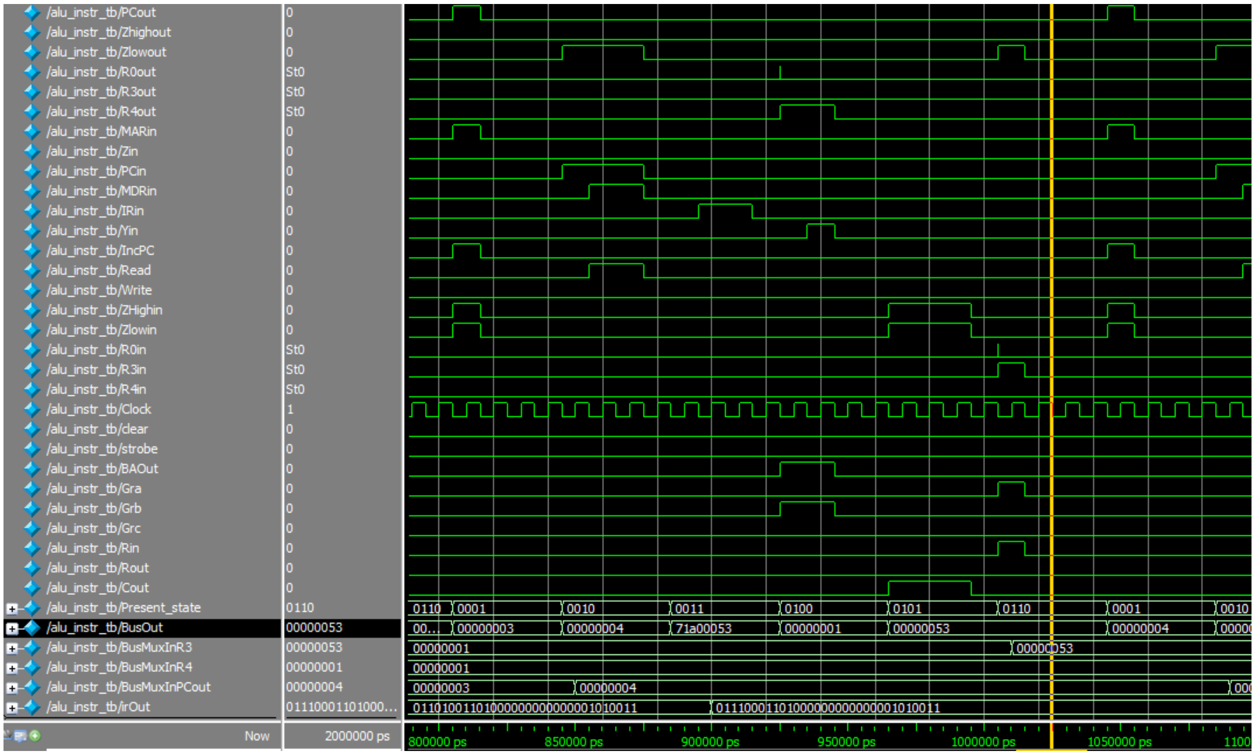
/alu_instr_tb/LOin	HiZ
/alu_instr_tb/PCout	0
/alu_instr_tb/Zlowout	0
/alu_instr_tb/MDRout	0
/alu_instr_tb/R0out	St0
/alu_instr_tb/R3out	St0
/alu_instr_tb/R4out	St0
/alu_instr_tb/MARin	0
/alu_instr_tb/PCin	0
/alu_instr_tb/MDRin	0
/alu_instr_tb/IRin	0
/alu_instr_tb/Yin	0
/alu_instr_tb/IncPC	0
/alu_instr_tb/Read	0
/alu_instr_tb/ZHighin	0
/alu_instr_tb/Zlowin	0
/alu_instr_tb/R0in	St0
/alu_instr_tb/R3in	St0
/alu_instr_tb/R4in	St0
/alu_instr_tb/Clock	0
/alu_instr_tb/clear	0
/alu_instr_tb/BAOut	0
/alu_instr_tb/Gra	0
/alu_instr_tb/Grb	0
/alu_instr_tb/Grc	0
/alu_instr_tb/Rin	0
/alu_instr_tb/Rout	0
/alu_instr_tb/Cout	0
/alu_instr_tb/Present_state	0110
/alu_instr_tb/op	00000
/alu_instr_tb/BusOut	-4
/alu_instr_tb/BusMuxInR0	0
/alu_instr_tb/BusMuxInR3	-4
/alu_instr_tb/BusMuxInR4	1
/alu_instr_tb/BusMuxInZlow	-4
/alu_instr_tb/BusMuxInPCout	2
/alu_instr_tb/BusMuxInYout	1
/alu_instr_tb/irOut	1638399995
/alu_instr_tb/ZHighWire	0
/alu_instr_tb/ZLowWire	-3
Now	2000 ns
Cursor 1	539.915 ns

andi R3, R4, 0x53



/alu_instr_tb/LOin	HiZ
/alu_instr_tb/PCout	0
/alu_instr_tb/Zhighout	0
/alu_instr_tb/Zlowout	0
/alu_instr_tb/R0out	St0
/alu_instr_tb/R3out	St0
/alu_instr_tb/R4out	St0
/alu_instr_tb/MARin	0
/alu_instr_tb/Zin	0
/alu_instr_tb/PCin	0
/alu_instr_tb/MDRin	0
/alu_instr_tb/IRin	0
/alu_instr_tb/Yin	0
/alu_instr_tb/IncPC	0
/alu_instr_tb/Read	0
/alu_instr_tb/Write	0
/alu_instr_tb/ZHighin	0
/alu_instr_tb/Zlowin	0
/alu_instr_tb/R0in	St0
/alu_instr_tb/R3in	St0
/alu_instr_tb/R4in	St0
/alu_instr_tb/Clock	1
/alu_instr_tb/clear	0
/alu_instr_tb/strobe	0
/alu_instr_tb/BAOut	0
/alu_instr_tb/Gra	0
/alu_instr_tb/Grb	0
/alu_instr_tb/Grc	0
/alu_instr_tb/Rin	0
/alu_instr_tb/Rout	0
/alu_instr_tb/Cout	0
/alu_instr_tb/Present_state	0010
/alu_instr_tb/BusOut	00000004
/alu_instr_tb/BusMuxInR3	00000001
/alu_instr_tb/BusMuxInR4	00000001
/alu_instr_tb/BusMuxInPCout	00000004
/alu_instr_tb/irOut	01101001101000...
Now	2000000 ps
Cursor 1	882367 ps

ori R3, R4, 0x53



/alu_instr_tb/PCout	0
/alu_instr_tb/Zhighout	0
/alu_instr_tb/Zlowout	0
/alu_instr_tb/R0out	St0
/alu_instr_tb/R3out	St0
/alu_instr_tb/R4out	St0
/alu_instr_tb/MARin	0
/alu_instr_tb/Zin	0
/alu_instr_tb/PCin	0
/alu_instr_tb/MDRin	0
/alu_instr_tb/IRin	0
/alu_instr_tb/Yin	0
/alu_instr_tb/IncPC	0
/alu_instr_tb/Read	0
/alu_instr_tb/Write	0
/alu_instr_tb/ZHighin	0
/alu_instr_tb/Zlowin	0
/alu_instr_tb/R0in	St0
/alu_instr_tb/R3in	St0
/alu_instr_tb/R4in	St0
/alu_instr_tb/Clock	1
/alu_instr_tb/clear	0
/alu_instr_tb/strobe	0
/alu_instr_tb/BAOut	0
/alu_instr_tb/Gra	0
/alu_instr_tb/Grb	0
/alu_instr_tb/Grc	0
/alu_instr_tb/Rin	0
/alu_instr_tb/Rout	0
/alu_instr_tb/Cout	0
/alu_instr_tb/Present_state	0110
/alu_instr_tb/BusOut	00000053
/alu_instr_tb/BusMuxInR3	00000053
/alu_instr_tb/BusMuxInR4	00000001
/alu_instr_tb/BusMuxInPCout	00000004
/alu_instr_tb/irOut	01110001101000...
Now	2000000 ps

4 Branch Instructions

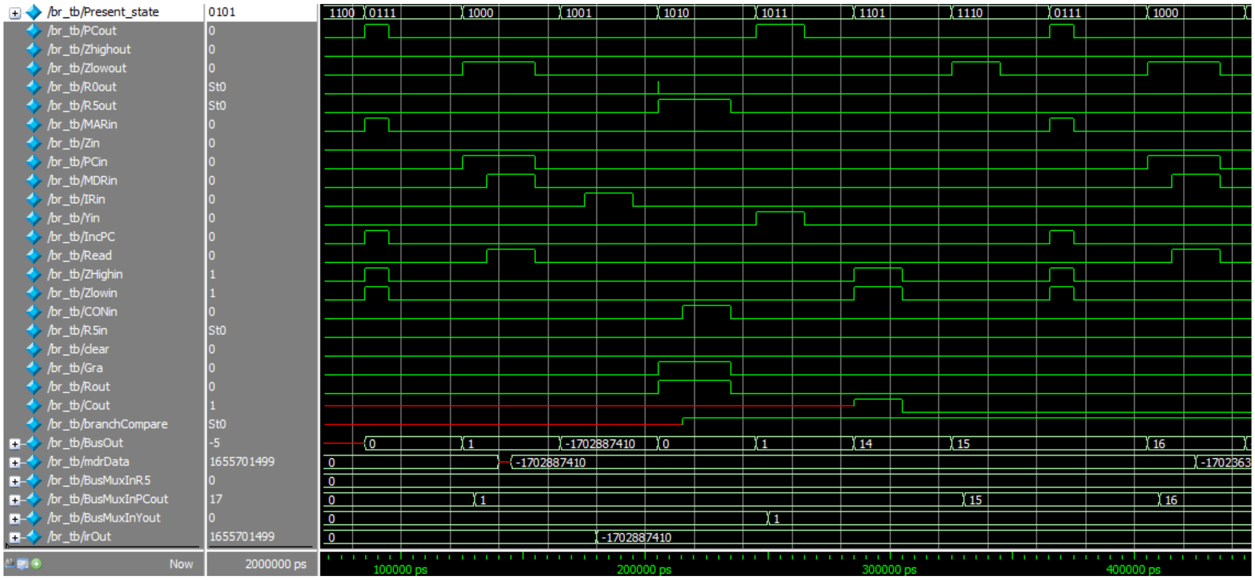
Testbench for Branch Instructions

```
`timescale 1ns/10ps

module br_tb;

        reg PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout, RAMout;
        wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out, R8out, R9out, R10out, R11out, R12out, R13out,
R14out, R15out;
        reg MARin, Zin, PCin, MDRin, IRin, OutPortin, Yin, IncPC, Read, Write, AND, HIin, LOin, ZHighin, Zlowin,
CONin;
        wire R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in, R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;
        reg Clock, clear, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, Cout;
        wire branchCompare;
        wire [3:0] to_decode;
        reg [31:0] Mdatain, input_data;
        parameter Default = 4'b0000, s1 = 4'b0001, s2 = 4'b0010, s3 = 4'b0011,
                                s4 = 4'b0100, s5 = 4'b0101, s6 = 4'b0110, s7 = 4'b0111, s8 = 4'b1000, s9 =
4'b1001, s10 = 4'b1010, s11 = 4'b1011, sclear = 4'b1100, s12 = 4'b1101, s13 = 4'b1110, s14 = 4'b1111;
        reg [3:0] Present_state = Default;
        reg [4:0] op;
        wire [31:0] BusOut, mdrData, BusMuxInR0, BusMuxInR1, BusMuxInR2,  BusMuxInR3, BusMuxInR4,
BusMuxInR5, BusMuxInR6, BusMuxInR7,
                                        BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12,
BusMuxInR13, BusMuxInR14, BusMuxInR15,
                                        BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout,
BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output_data, irOut,
                                        ZHighWire, ZLowWire;
        //wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out,
R13out, R14out, R15out, R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in,
R15in;

        data_path DUT(Clock, clear, Read, Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, CONin,
        input_data,IRin,
        op,
        HIOut, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Yout, RAMout, Cout,
        HIin,  LOin,  ZHighin,  Zlowin,  PCin,  MDRin,  OutPortin, Yin, MARin, IncPC,
        BusOut, mdrData, ZHighWire, ZLowWire,
        BusMuxInR0, BusMuxInR1, BusMuxInR2,  BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,
BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,
        BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI,
BusMuxInLO, BusMuxInRamout, output_data, irOut,
        branchCompare,
        R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out,
R14out, R15out,
        R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in,
        to_decode);
integer flag;

initial begin
        flag = 0;
        Clock = 1;
end


always #5 Clock = ~Clock;


always @(negedge Clock) begin// finite state machine; if clock falling-edge so as to be offset from reg clocking
        case (Present_state)
                Default : #40 Present_state = sclear;
                sclear : #40 Present_state = s7;
                s1 : #40 Present_state = s2;
                s2 : #40 Present_state = s3;
                s3 : #40 Present_state = s4;
                s4 : #40 Present_state = s5;
                s5 : #40 Present_state = s6;
                s6      : #40 Present_state = s7;
                s7      : #40 Present_state = s8;
                s8      : #40 Present_state = s9;
                s9 : #40 Present_state = s10;
                s10 : #40 Present_state = s11;
                s11 : #40 Present_state = s12;
                s12 : #40 Present_state = s13;
                s13 : #40 Present_state = (flag == 2) ? s1 : s7;
        endcase
end

always @(Present_state) begin // do the required job in each state
        case (Present_state) // assert the required signals in each clock cycle
                Default: begin
                        /*PCout <= 0; Zlowout <= 0; MDRout <= 0; // initialize the signals
                        R2out <= 0; R3out <= 0; MARin <= 0; Zin <= 0;
                        PCin <=0; MDRin <= 0; IRin <= 0; Yin <= 0;
                        IncPC <= 0; Read <= 0; AND <= 0;
                        R1in <= 0; R2in <= 0; R3in <= 0; Mdatain <= 32'h00000000;*/
                        {Write, strobe, BAOut,  Gra, Grb, Grc, Rin, Rout} <= 8'b0;
                        {PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout} <= 8'b0;
                        {MARin, Zin, PCin, MDRin, IRin, Yin, IncPC, Read, AND, HIin, InPortout, LOin, ZHighin,
Zlowin} <= 13'b0;
                        clear<=0;
                        Mdatain <= 32'h00000000;
                        BAOut <= 0;
                        CONin <= 0;
                end

                sclear : begin
                        clear <= 1;
                        #10 clear <= 0;
                end

                s1 : begin //t0
                        flag <= 0;
                        PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus
                        #10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;
```

```
end

s2 : begin //t1
        Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus
        #10 Read <= 1; MDRin <= 1;
        #20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;
end

s3 : begin //t2
        MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus
        #10 IRin <= 1;
        #20 MDRout <= 0; IRin <= 0;
end

s4 : begin //t3
        Grb <= 1; BAOut <= 1;
        #10 Yin <= 1; //see the value of the reg to be added
        #10 Grb <= 0; BAOut <= 0; Yin <= 0;
end

s5 : begin //t4 see C on the bus
        Cout<= 1; ZHighin <= 1; Zlowin <= 1;
        #10 op <= 5'b00011;
        #20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;
end

s6 : begin //t5
        Zlowout <= 1; Gra <= 1; Rin <= 1;// see the addition result on the bus
        #10 Zlowout <= 0; Gra <= 0; Rin <= 0;
end

s7 : begin //t0
        flag <= flag + 1;
        PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus
        #10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;
end

s8 : begin //t1
        Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus
        #10 Read <= 1; MDRin <= 1;
        #20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;
end

s9 : begin //t2
        MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus
        #10 IRin <= 1;
        #20 MDRout <= 0; IRin <= 0;
end

s10 : begin //t3
        Gra <= 1; Rout <= 1;
        #10 CONin <= 1;
        #20 Gra <= 0; Rout <= 0; CONin <= 0;
end
```

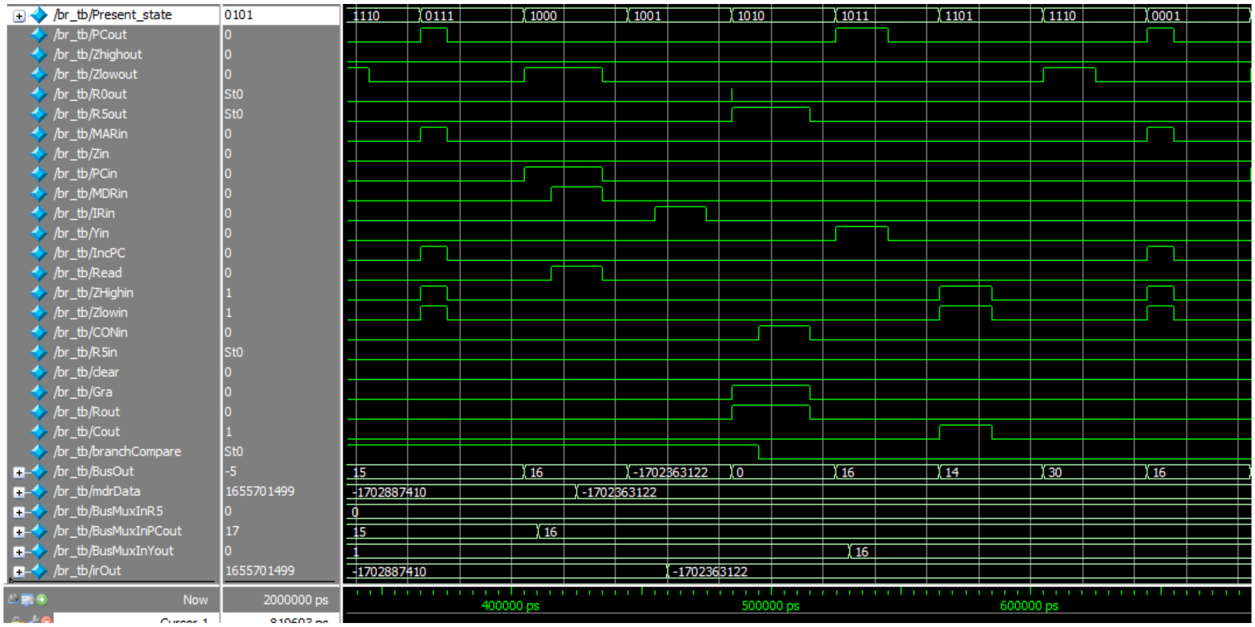
```
                    s11 : begin //t4
                              PCout <= 1; Yin <= 1;
                              #20 PCout <= 0; Yin <= 0;
                    end

                    s12 : begin //t5
                              Cout<= 1; ZHighin <= 1; Zlowin <= 1;
                              #20 Cout<= 0; ZHighin <= 0; Zlowin <= 0;
                    end

                    s13 : begin //t6
                              Zlowout <= 1;//This is where I need to use the Conin logic to figure out how to get this into PCin
                              #20 Zlowout <= 0;
                    end

          endcase
end

endmodule
```

brzr R5, 14



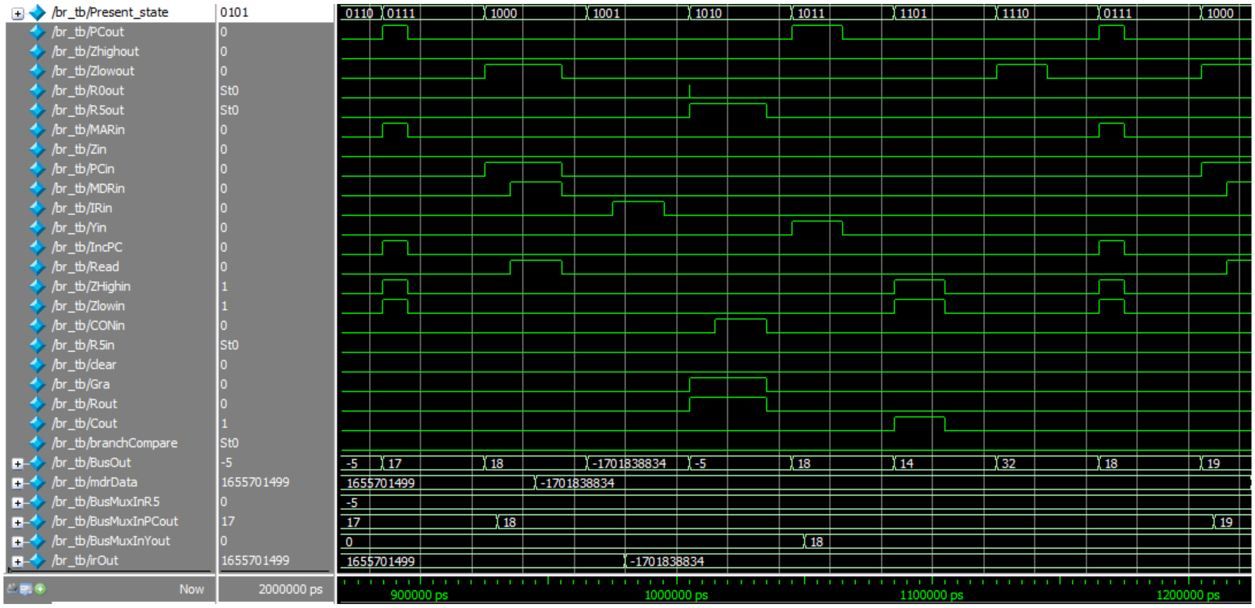
```
/br_tb/Present_state    0101
/br_tb/PCout            0
/br_tb/Zhighout         0
/br_tb/Zlowout          0
/br_tb/R0out            St0
/br_tb/R5out            St0
/br_tb/MARin            0
/br_tb/Zin              0
/br_tb/PCin             0
/br_tb/MDRin            0
/br_tb/IRin             0
/br_tb/Yin              0
/br_tb/IncPC            0
/br_tb/Read             0
/br_tb/ZHighin          1
/br_tb/Zlowin           1
/br_tb/CONin            0
/br_tb/R5in             St0
/br_tb/clear            0
/br_tb/Gra              0
/br_tb/Rout             0
/br_tb/Cout             1
/br_tb/branchCompare    St0
/br_tb/BusOut           -5
/br_tb/mdrData          1655701499
/br_tb/BusMuxInR5       0
/br_tb/BusMuxInPCout    17
/br_tb/BusMuxInYout     0
/br_tb/irOut            1655701499
Now                     2000000 ps
```

brnr R5, 14



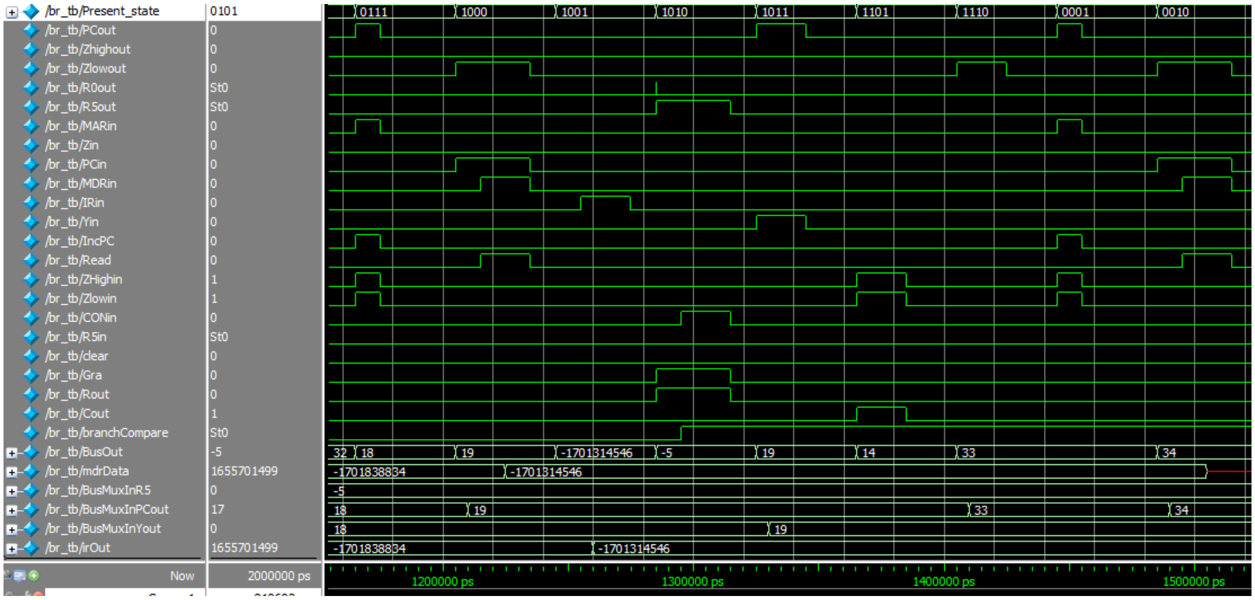
/br_tb/Present_state	0101
/br_tb/PCout	0
/br_tb/Zhighout	0
/br_tb/Zlowout	0
/br_tb/R0out	St0
/br_tb/R5out	St0
/br_tb/MARin	0
/br_tb/Zin	0
/br_tb/PCin	0
/br_tb/MDRin	0
/br_tb/IRin	0
/br_tb/Yin	0
/br_tb/IncPC	0
/br_tb/Read	0
/br_tb/ZHighin	1
/br_tb/Zlowin	1
/br_tb/CONin	0
/br_tb/R5in	St0
/br_tb/clear	0
/br_tb/Gra	0
/br_tb/Rout	0
/br_tb/Cout	1
/br_tb/branchCompare	St0
/br_tb/BusOut	-5
/br_tb/mdrData	1655701499
/br_tb/BusMuxInR5	0
/br_tb/BusMuxInPCout	17
/br_tb/BusMuxInYout	0
/br_tb/irOut	1655701499
Now	2000000 ps

brpl R5, 14



/br_tb/Present_state	0101
/br_tb/PCout	0
/br_tb/Zhighout	0
/br_tb/Zlowout	0
/br_tb/R0out	St0
/br_tb/R5out	St0
/br_tb/MARin	0
/br_tb/Zin	0
/br_tb/PCin	0
/br_tb/MDRin	0
/br_tb/IRin	0
/br_tb/Yin	0
/br_tb/IncPC	0
/br_tb/Read	0
/br_tb/ZHighin	1
/br_tb/Zlowin	1
/br_tb/CONin	0
/br_tb/R5in	St0
/br_tb/clear	0
/br_tb/Gra	0
/br_tb/Rout	0
/br_tb/Cout	1
/br_tb/branchCompare	St0
/br_tb/BusOut	-5
/br_tb/mdrData	1655701499
/br_tb/BusMuxInR5	0
/br_tb/BusMuxInPCout	17
/br_tb/BusMuxInYout	0
/br_tb/irOut	1655701499
Now	2000000 ps

brmi R5, 14



```
/br_tb/Present_state      0101
/br_tb/PCout              0
/br_tb/Zhighout           0
/br_tb/Zlowout            0
/br_tb/R0out              St0
/br_tb/R5out              St0
/br_tb/MARin              0
/br_tb/Zin                0
/br_tb/PCin               0
/br_tb/MDRin              0
/br_tb/IRin               0
/br_tb/Yin                0
/br_tb/IncPC              0
/br_tb/Read               0
/br_tb/ZHighin            1
/br_tb/Zlowin             1
/br_tb/CONin              0
/br_tb/R5in               St0
/br_tb/clear              0
/br_tb/Gra                0
/br_tb/Rout               0
/br_tb/Cout               1
/br_tb/branchCompare      St0
/br_tb/BusOut             -5
/br_tb/mdrData            1655701499
/br_tb/BusMuxInR5         0
/br_tb/BusMuxInPCout      17
/br_tb/BusMuxInYout       0
/br_tb/irOut              1655701499
Now                       2000000 ps
```

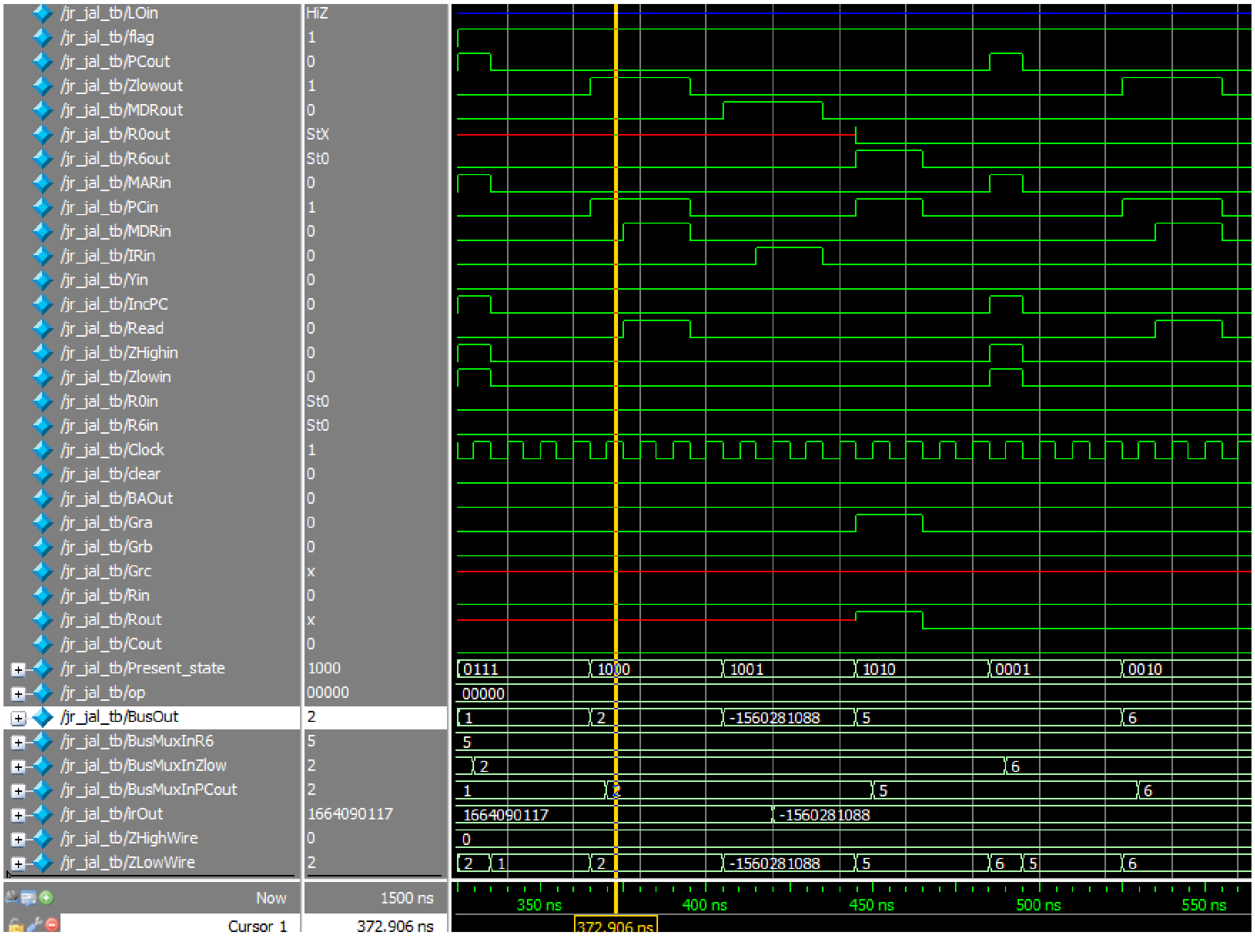
5 Jump Instructions

Testbench

```
`timescale 1ns/10ps

module jr_jal_tb;

	reg PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout, RAMout, CONin;
	wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out, R8out, R9out, R10out, R11out, R12out, R13out,
R14out, R15out;
	reg MARin, Zin, PCin, MDRin, IRin, OutPortin, Yin, IncPC, Read, Write, AND, HIin, LOin, ZHighin, Zlowin;
	wire R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in, R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;
	reg Clock, clear, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, Cout;
	wire branchCompare;
	wire [3:0] to_decode;
	reg [31:0] Mdatain, input_data;
	parameter Default = 4'b0000, s1 = 4'b0001, s2 = 4'b0010, s3 = 4'b0011,
					s4 = 4'b0100, s5 = 4'b0101, s6 = 4'b0110, s7 = 4'b0111, s8 = 4'b1000, s9 =
4'b1001, s10 = 4'b1010, s11 = 4'b1011, sclear = 4'b1100, s12 = 4'b1101, s13 = 4'b1110, s14 = 4'b1111, s15 =
5'b10000;
	reg [4:0] Present_state = Default;
	reg [4:0] op;
	wire [31:0] BusOut, mdrData, BusMuxInR0, BusMuxInR1, BusMuxInR2,  BusMuxInR3, BusMuxInR4,
BusMuxInR5, BusMuxInR6, BusMuxInR7,
						BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12,
BusMuxInR13, BusMuxInR14, BusMuxInR15,
						BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout,
BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output_data, irOut,
						ZHighWire, ZLowWire;
	//wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out,
R13out, R14out, R15out, R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in,
R15in;
```

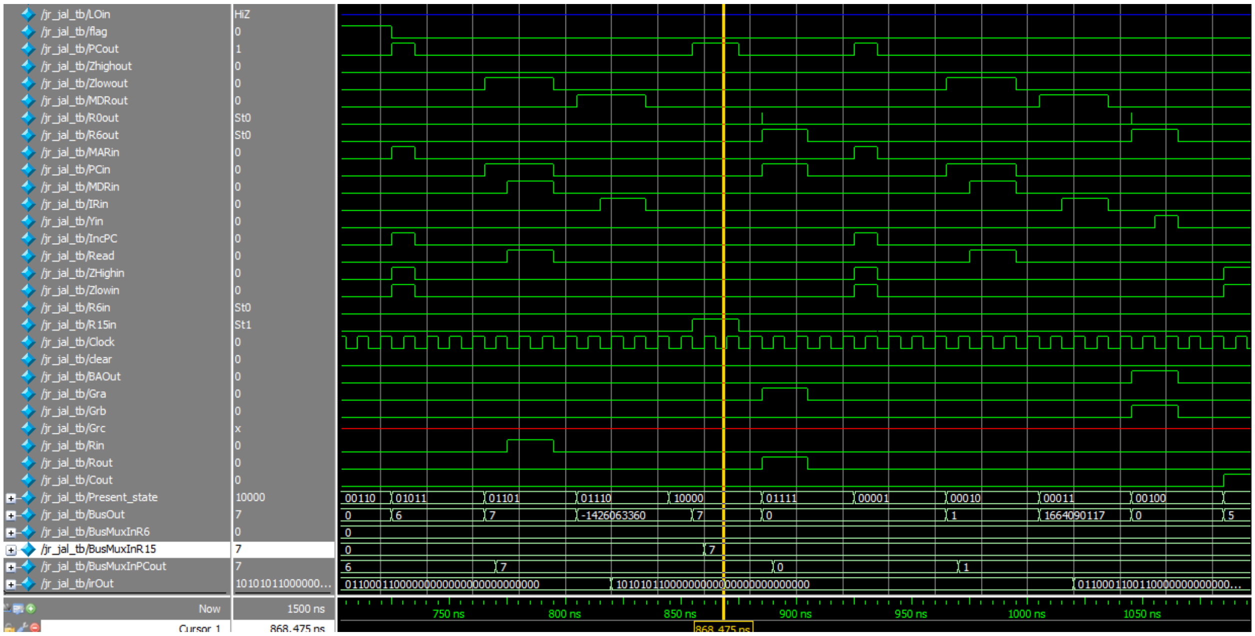
```
    data_path DUT(Clock, clear, Read, Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, CONin,
    input_data,IRin,
    op,
    HIOut, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Yout, RAMout, Cout,
    HIin,  LOin,  ZHighin,  Zlowin,  PCin,  MDRin,  OutPortin, Yin, MARin, IncPC,
    BusOut, mdrData, ZHighWire, ZLowWire,
    BusMuxInR0, BusMuxInR1, BusMuxInR2,  BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,
BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,
    BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI,
BusMuxInLO, BusMuxInRamout, output_data, irOut,
    branchCompare,
    R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out,
R14out, R15out,
    R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in,
    to_decode);
reg flag;

initial begin
    flag = 0;
    Clock = 1;
end


always #5 Clock = ~Clock;


always @(negedge Clock) begin// finite state machine; if clock falling-edge so as to be offset from reg clocking
    case (Present_state)
        Default : #40 Present_state = sclear;
        sclear : #40 Present_state = s1;
        s1 : #40 Present_state = s2;
        s2 : #40 Present_state = s3;
        s3 : #40 Present_state = s4;
        s4 : #40 Present_state = s5;
        s5 : #40 Present_state = s6;
        s6      : #40 Present_state = flag ? s11 : s7;
        s7      : #40 Present_state = s8;
        s8      : #40 Present_state = s9;
        s9 : #40 Present_state = s10;
        s10 : #40 Present_state = s1;
        s11     : #40 Present_state = s12;
        s12 : #40 Present_state = s13;
        s13 : #40 Present_state = s15;
        s15 : #40 Present_state = s14;
        s14 : #40 Present_state = s1;
    endcase
end

always @(Present_state) begin // do the required job in each state
    case (Present_state) // assert the required signals in each clock cycle
        Default: begin
            /*PCout <= 0; Zlowout <= 0; MDRout <= 0; // initialize the signals
            R2out <= 0; R3out <= 0; MARin <= 0; Zin <= 0;
            PCin <=0; MDRin <= 0; IRin <= 0; Yin <= 0;
            IncPC <= 0; Read <= 0; AND <= 0;
            R1in <= 0; R2in <= 0; R3in <= 0; Mdatain <= 32'h00000000;*/
            {Write, strobe, BAOut,  Gra, Grb, Grc, Rin, Rout} <= 8'b0;
            {PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout} <= 8'b0;
            {MARin, Zin, PCin, MDRin, IRin, Yin, IncPC, Read, AND, HIin, InPortout, Loin, ZHighin, Zlowin} <= 13'b0;
            clear<=0;
            Mdatain <= 32'h00000000;
            BAOut <= 0;
        end

        sclear : begin
            clear <= 1;
            #10 clear <= 0;
        end

        s1 : begin //t0
            PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus
            #10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;
        end

        s2 : begin //t1
            Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus
            #10 Read <= 1; MDRin <= 1;
            #20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;
        end

        s3 : begin //t2
            MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus
            #10 IRin <= 1;
            #20 MDRout <= 0; IRin <= 0;
        end

        s4 : begin //t3
            Grb <= 1; BAOut <= 1;
            #10 Yin <= 1; //see the value of the reg to be added
            #10 Grb <= 0; BAOut <= 0; Yin <= 0;
        end

        s5 : begin //t4 see C on the bus
            Cout<= 1; ZHighin <= 1; Zlowin <= 1;
            #10 op <= 5'b00011;
            #20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;
        end

        s6 : begin //t5
            Zlowout <= 1; Gra <= 1; Rin <= 1;// see the addition result on the bus
            #10 Zlowout <= 0; Gra <= 0; Rin <= 0;
        end

        s7 : begin //t0
            flag <= 1;
            PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus
            #10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;
        end

        s8 : begin //t1
            Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus
            #10 Read <= 1; MDRin <= 1;
            #20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;
        end

        s9 : begin //t2
            MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus
            #10 IRin <= 1;
            #20 MDRout <= 0; IRin <= 0;
        end

        s10 : begin //t3
            Gra <= 1; Rout <= 1; PCin <= 1;
            #20 Gra <= 0; Rout <= 0; PCin <= 0;
        end

        s11 : begin //t0
            flag <= 0;
            PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus
            #10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;
        end

        s12 : begin //t1
            Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus
            #10 Read <= 1; MDRin <= 1; Rin <= 1;
            #20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0; Rin <= 0;
        end

        s13 : begin //t2
            MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus
            #10 IRin <= 1;
            #20 MDRout <= 0; IRin <= 0;
        end

        s15 : begin
            #10 PCout <= 1;
            #20 PCout <= 0;
        end

        s14 : begin //t3
            Gra <= 1; Rout <= 1; PCin <= 1;
            #20 Gra <= 0; Rout <= 0; PCin <= 0;
        end


    endcase
end

endmodule
```

jr R6



```
/jr_jal_tb/LOin            HiZ
/jr_jal_tb/flag            1
/jr_jal_tb/PCout           0
/jr_jal_tb/Zlowout         1
/jr_jal_tb/MDRout          0
/jr_jal_tb/R0out           StX
/jr_jal_tb/R6out           St0
/jr_jal_tb/MARin           0
/jr_jal_tb/PCin            1
/jr_jal_tb/MDRin           0
/jr_jal_tb/IRin            0
/jr_jal_tb/Yin             0
/jr_jal_tb/IncPC           0
/jr_jal_tb/Read            0
/jr_jal_tb/ZHighin         0
/jr_jal_tb/Zlowin          0
/jr_jal_tb/R0in            St0
/jr_jal_tb/R6in            St0
/jr_jal_tb/Clock           1
/jr_jal_tb/clear           0
/jr_jal_tb/BAOut           0
/jr_jal_tb/Gra             0
/jr_jal_tb/Grb             0
/jr_jal_tb/Grc             x
/jr_jal_tb/Rin             0
/jr_jal_tb/Rout            x
/jr_jal_tb/Cout            0
/jr_jal_tb/Present_state   1000
/jr_jal_tb/op              00000
/jr_jal_tb/BusOut          2
/jr_jal_tb/BusMuxInR6      5
/jr_jal_tb/BusMuxInZlow    2
/jr_jal_tb/BusMuxInPCout   2
/jr_jal_tb/irOut           1664090117
/jr_jal_tb/ZHighWire       0
/jr_jal_tb/ZLowWire        2
Now                        1500 ns
Cursor 1                   372.906 ns
```

jal R6



```
/jr_jal_tb/LOin            HiZ
/jr_jal_tb/flag            0
/jr_jal_tb/PCout           1
/jr_jal_tb/Zhighout        0
/jr_jal_tb/Zlowout         0
/jr_jal_tb/MDRout          0
/jr_jal_tb/R0out           St0
/jr_jal_tb/R6out           St0
/jr_jal_tb/MARin           0
/jr_jal_tb/PCin            0
/jr_jal_tb/MDRin           0
/jr_jal_tb/IRin            0
/jr_jal_tb/Yin             0
/jr_jal_tb/IncPC           0
/jr_jal_tb/Read            0
/jr_jal_tb/ZHighin         0
/jr_jal_tb/Zlowin          0
/jr_jal_tb/R6in            St0
/jr_jal_tb/R15in           St1
/jr_jal_tb/Clock           0
/jr_jal_tb/clear           0
/jr_jal_tb/BAOut           0
/jr_jal_tb/Gra             0
/jr_jal_tb/Grb             0
/jr_jal_tb/Grc             x
/jr_jal_tb/Rin             0
/jr_jal_tb/Rout            0
/jr_jal_tb/Cout            0
/jr_jal_tb/Present_state   10000
/jr_jal_tb/BusOut          7
/jr_jal_tb/BusMuxInR6      0
/jr_jal_tb/BusMuxInR15     7
/jr_jal_tb/BusMuxInPCout   7
/jr_jal_tb/irOut           1010101100000000...
Now                        1500 ns
Cursor 1                   868.475 ns
```

6 Special Instructions

Testbench

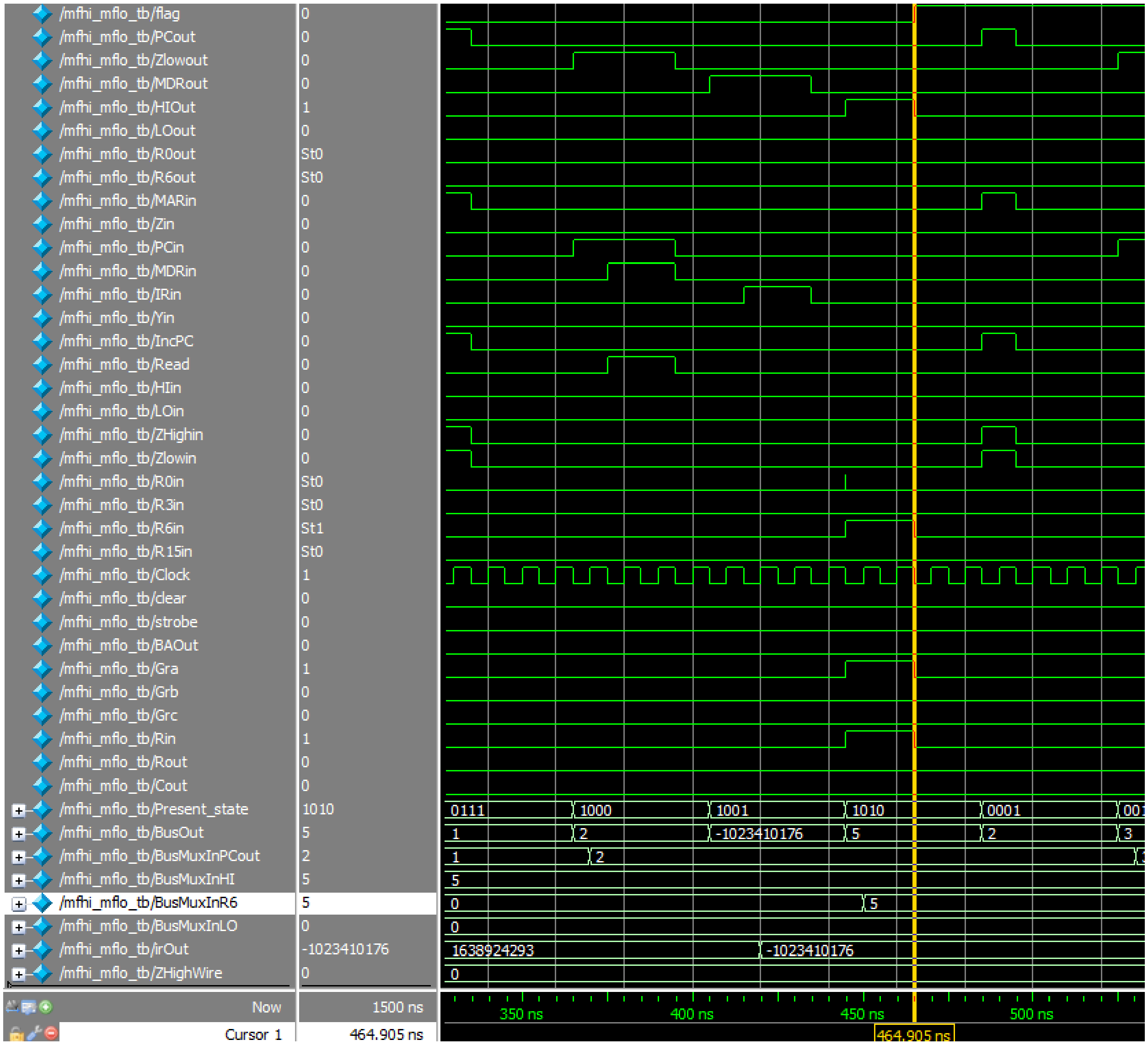
```
`timescale 1ns/10ps

module mfhi_mflo_tb;

	reg PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout, RAMout, CONin;
	wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out, R8out, R9out, R10out, R11out, R12out, R13out, 
R14out, R15out;
	reg MARin, Zin, PCin, MDRin, IRin, OutPortin, Yin, IncPC, Read, Write, AND, HIin, LOin, ZHighin, Zlowin;
	wire R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in, R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;
	reg Clock, clear, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, Cout;
	wire branchCompare;
	wire [3:0] to_decode;
	reg [31:0] Mdatain, input_data;
	parameter Default = 4'b0000, s1 = 4'b0001, s2 = 4'b0010, s3 = 4'b0011,
				s4 = 4'b0100, s5 = 4'b0101, s6 = 4'b0110, s7 = 4'b0111, s8 = 4'b1000, s9 = 
4'b1001, s10 = 4'b1010, s11 = 4'b1011, sclear = 4'b1100, s12 = 4'b1101, s13 = 4'b1110, s14 = 4'b1111;
	reg [3:0] Present_state = Default;
	reg [4:0] op;
	wire [31:0] BusOut, mdrData, BusMuxInR0, BusMuxInR1, BusMuxInR2,  BusMuxInR3, BusMuxInR4, 
BusMuxInR5, BusMuxInR6, BusMuxInR7,
					BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, 
BusMuxInR13, BusMuxInR14, BusMuxInR15,
					BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, 
BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output_data, irOut,
					ZHighWire, ZLowWire;
	//wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, 
R13out, R14out, R15out, R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, 
R15in;

	data_path DUT(Clock, clear, Read, Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, CONin,
	input_data,IRin,
	op,
	HIOut, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Yout, RAMout, Cout,
	HIin,  LOin,  ZHighin,  Zlowin,  PCin,  MDRin,  OutPortin, Yin, MARin, IncPC,
	BusOut, mdrData, ZHighWire, ZLowWire,
	BusMuxInR0, BusMuxInR1, BusMuxInR2,  BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7, 
BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,
	BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, 
BusMuxInLO, BusMuxInRamout, output_data, irOut,
	branchCompare,
	R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, 
R14out, R15out,
	R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in,
	to_decode);
reg flag;

initial begin
	flag = 0;
	Clock = 1;
end
```

```
always #5 Clock = ~Clock;


always @(negedge Clock) begin// finite state machine; if clock falling-edge so as to be offset from reg clocking
        case (Present_state)
                Default : #40 Present_state = sclear;
                sclear : #40 Present_state = s1;
                s1 : #40 Present_state = s2;
                s2 : #40 Present_state = s3;
                s3 : #40 Present_state = s4;
                s4 : #40 Present_state = s5;
                s5 : #40 Present_state = s6;
                s6      : #40 Present_state = s7;
                s7      : #40 Present_state = s8;
                s8      : #40 Present_state = s9;
                s9 : #40 Present_state = s10;
                s10 : #40 Present_state = s1;
                s11     : #40 Present_state = s12;
                s12 : #40 Present_state = s13;
                s13 : #40 Present_state = s14;
                s14 : #40 Present_state = s1;
        endcase
end

always @(Present_state) begin // do the required job in each state
        case (Present_state) // assert the required signals in each clock cycle
                Default: begin
                        /*PCout <= 0; Zlowout <= 0; MDRout <= 0; // initialize the signals
                        R2out <= 0; R3out <= 0; MARin <= 0; Zin <= 0;
                        PCin <=0; MDRin <= 0; IRin <= 0; Yin <= 0;
                        IncPC <= 0; Read <= 0; AND <= 0;
                        R1in <= 0; R2in <= 0; R3in <= 0; Mdatain <= 32'h00000000;*/
                        {Write, strobe, BAOut,  Gra, Grb, Grc, Rin, Rout} <= 8'b0;
                        {PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout} <= 8'b0;
                        {MARin, Zin, PCin, MDRin, IRin, Yin, IncPC, Read, AND, HIin, InPortout, LOin, ZHighin,
Zlowin} <= 13'b0;
                        clear<=0;
                        Mdatain <= 32'h00000000;
                        BAOut <= 0;
                end

                sclear : begin
                        clear <= 1;
                        #10 clear <= 0;
                end

                s1 : begin //t0
                        PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus
                        #10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;
                end

                s2 : begin //t1
                        Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus
```

```
    #10 Read <= 1; MDRin <= 1;
    #20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;
end

s3 : begin //t2
    MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus
    #10 IRin <= 1;
    #20 MDRout <= 0; IRin <= 0;
end

s4 : begin //t3
    Grb <= 1; BAOut <= 1;
    #10 Yin <= 1; //see the value of the reg to be added
    #10 Grb <= 0; BAOut <= 0; Yin <= 0;
end

s5 : begin //t4 see C on the bus
    Cout<= 1; ZHighin <= 1; Zlowin <= 1;
    #10 op <= 5'b00011;
    #20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;
end

s6 : begin //t5
    if(flag == 0) HIin <= 1;
    else  LOin <= 1;
    Zlowout <= 1; Gra <= 1; Rin <= 1;// see the addition result on the bus
    #10 Zlowout <= 0; Gra <= 0; Rin <= 0; HIin <= 0; LOin <= 0;
end

s7 : begin //t0
    PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus
    #10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;
end

s8 : begin //t1
    Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus
    #10 Read <= 1; MDRin <= 1;
    #20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;
end

s9 : begin //t2
    MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus
    #10 IRin <= 1;
    #20 MDRout <= 0; IRin <= 0;
end

s10 : begin //t3
    if (flag == 0) HIOut <= 1;
    else LOout <= 1;
    Gra <= 1; Rin <= 1;
    #20 Gra <= 0; Rin <= 0; HIOut <= 0; LOout <= 0;
    flag = 1;
end
```

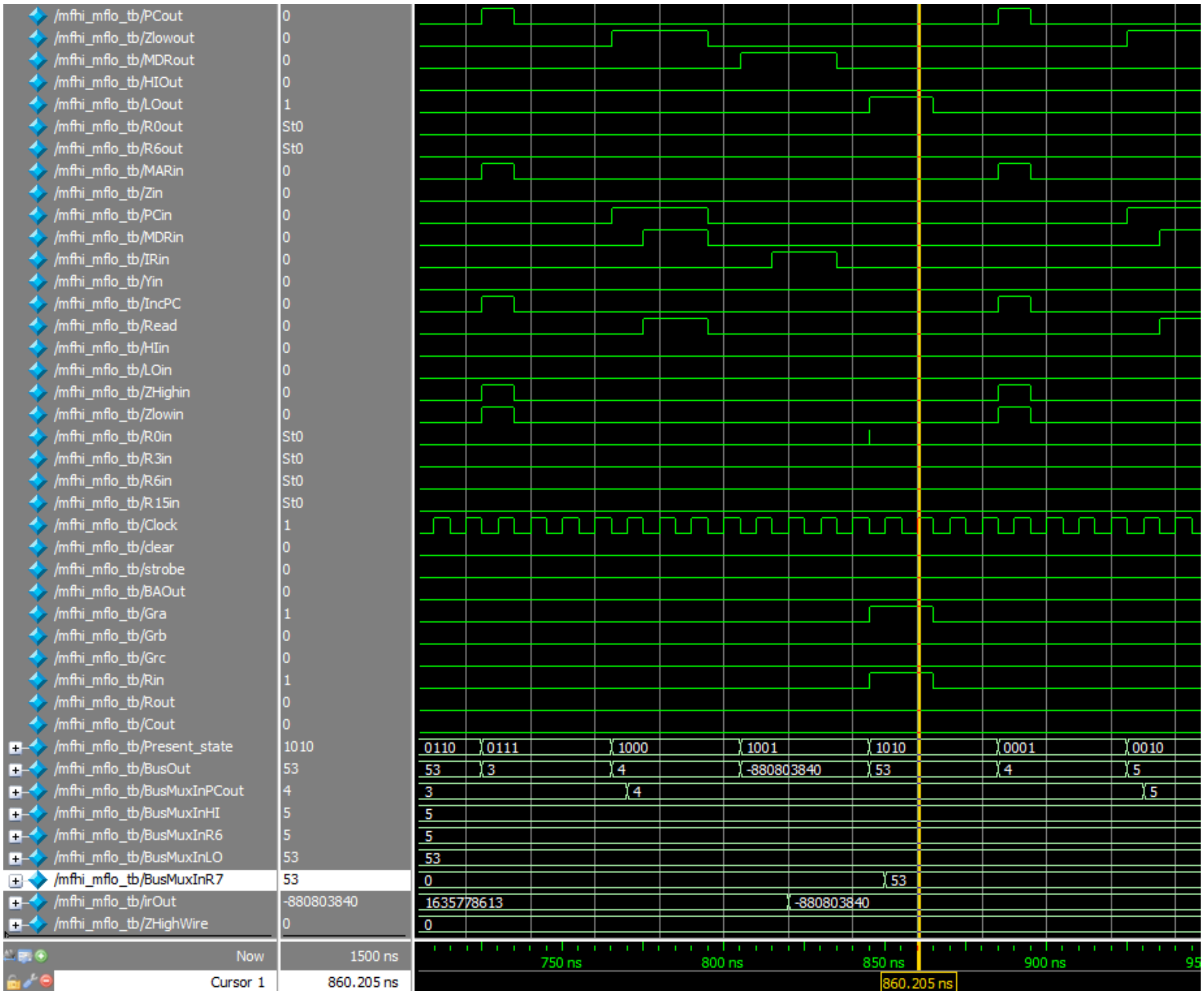
```
        endcase
end

endmodule
```

mfhi R6



/mfhi_mflo_tb/flag	0
/mfhi_mflo_tb/PCout	0
/mfhi_mflo_tb/Zlowout	0
/mfhi_mflo_tb/MDRout	0
/mfhi_mflo_tb/HIOut	1
/mfhi_mflo_tb/LOout	0
/mfhi_mflo_tb/R0out	St0
/mfhi_mflo_tb/R6out	St0
/mfhi_mflo_tb/MARin	0
/mfhi_mflo_tb/Zin	0
/mfhi_mflo_tb/PCin	0
/mfhi_mflo_tb/MDRin	0
/mfhi_mflo_tb/IRin	0
/mfhi_mflo_tb/Yin	0
/mfhi_mflo_tb/IncPC	0
/mfhi_mflo_tb/Read	0
/mfhi_mflo_tb/HIin	0
/mfhi_mflo_tb/LOin	0
/mfhi_mflo_tb/ZHighin	0
/mfhi_mflo_tb/Zlowin	0
/mfhi_mflo_tb/R0in	St0
/mfhi_mflo_tb/R3in	St0
/mfhi_mflo_tb/R6in	St1
/mfhi_mflo_tb/R15in	St0
/mfhi_mflo_tb/Clock	1
/mfhi_mflo_tb/clear	0
/mfhi_mflo_tb/strobe	0
/mfhi_mflo_tb/BAOut	0
/mfhi_mflo_tb/Gra	1
/mfhi_mflo_tb/Grb	0
/mfhi_mflo_tb/Grc	0
/mfhi_mflo_tb/Rin	1
/mfhi_mflo_tb/Rout	0
/mfhi_mflo_tb/Cout	0
/mfhi_mflo_tb/Present_state	1010
/mfhi_mflo_tb/BusOut	5
/mfhi_mflo_tb/BusMuxInPCout	2
/mfhi_mflo_tb/BusMuxInHI	5
/mfhi_mflo_tb/BusMuxInR6	5
/mfhi_mflo_tb/BusMuxInLO	0
/mfhi_mflo_tb/irOut	-1023410176
/mfhi_mflo_tb/ZHighWire	0
Now	1500 ns
Cursor 1	464.905 ns

mflo R7



7 Input/Output Instructions

Testbench

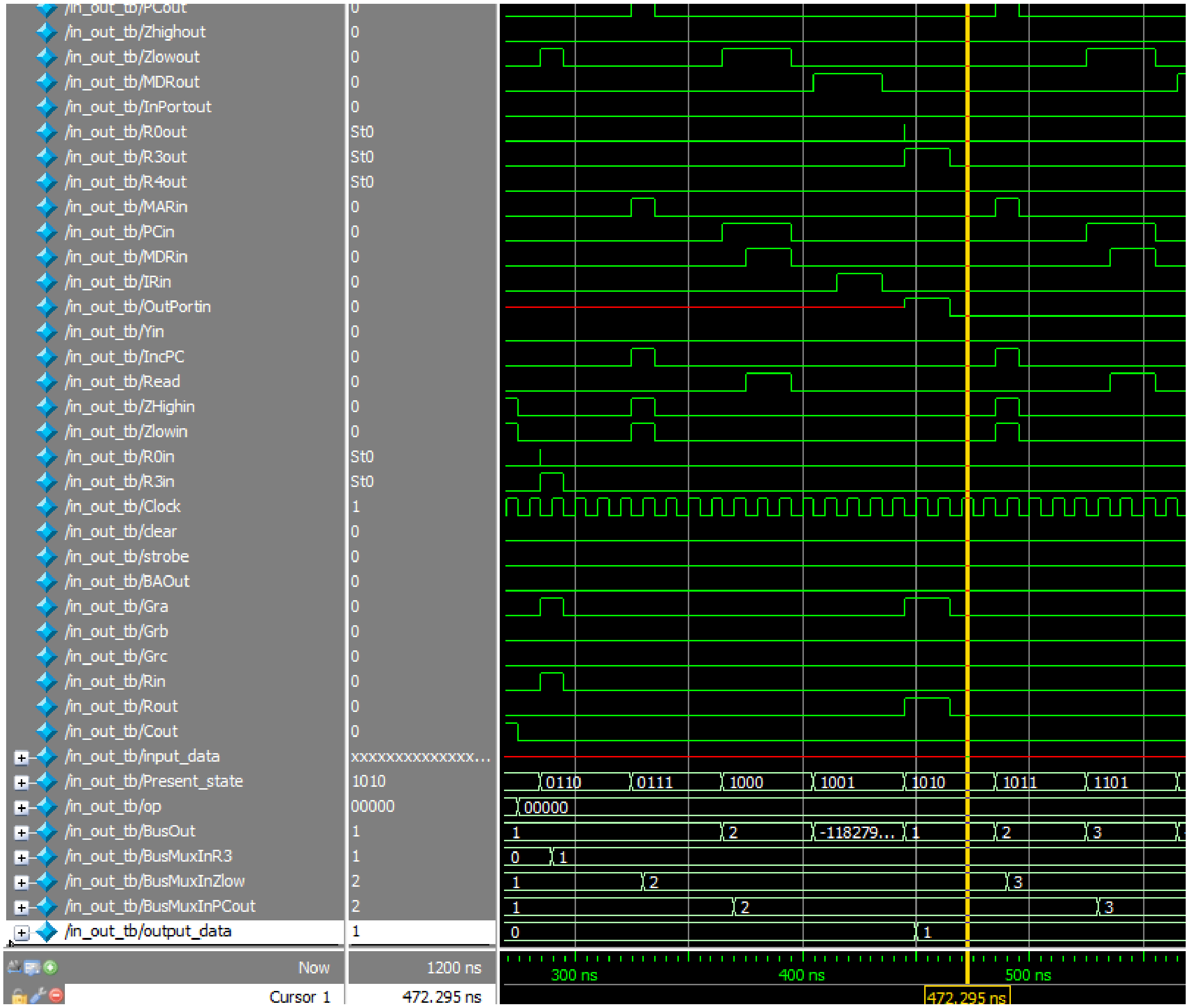
```
`timescale 1ns/10ps

module in_out_tb;

	reg PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout, RAMout, CONin;
	wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out, R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out;
	reg MARin, Zin, PCin, MDRin, IRin, OutPortin, Yin, IncPC, Read, Write, AND, HIin, Loin, ZHighin, Zlowin;
	wire R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in, R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;
	reg Clock, clear, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, Cout;
	wire branchCompare;
	wire [3:0] to_decode;
	reg [31:0] Mdatain, input_data;
	parameter Default = 4'b0000, s1 = 4'b0001, s2 = 4'b0010, s3 = 4'b0011,
                                        s4 = 4'b0100, s5 = 4'b0101, s6 = 4'b0110, s7 = 4'b0111, s8 = 4'b1000, s9 =
4'b1001, s10 = 4'b1010, s11 = 4'b1011, sclear = 4'b1100, s12 = 4'b1101, s13 = 4'b1110, s14 = 4'b1111;
          reg [3:0] Present_state = Default;
          reg [4:0] op;
          wire [31:0] BusOut, mdrData, BusMuxInR0, BusMuxInR1, BusMuxInR2,  BusMuxInR3, BusMuxInR4,
BusMuxInR5, BusMuxInR6, BusMuxInR7,
                                                  BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12,
BusMuxInR13, BusMuxInR14, BusMuxInR15,
                                                  BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout,
BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output_data, irOut,
                                                  ZHighWire, ZLowWire;
          //wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out,
R13out, R14out, R15out, R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in,
R15in;

          data_path DUT(Clock, clear, Read, Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, CONin,
          input_data,IRin,
          op,
          HIOut, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Yout, RAMout, Cout,
          HIin,  LOin,  ZHighin,  Zlowin,  PCin,  MDRin,  OutPortin, Yin, MARin, IncPC,
          BusOut, mdrData, ZHighWire, ZLowWire,
          BusMuxInR0, BusMuxInR1, BusMuxInR2,  BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,
BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,
          BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI,
BusMuxInLO, BusMuxInRamout, output_data, irOut,
          branchCompare,
          R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out,
R14out, R15out,
          R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in,
          to_decode);


initial begin
          Clock = 1;
end


always #5 Clock = ~Clock;


always @(negedge Clock) begin// finite state machine; if clock falling-edge so as to be offset from reg clocking
          case (Present_state)
                    Default : #40 Present_state = sclear;
                    sclear : #40 Present_state = s1;
                    s1 : #40 Present_state = s2;
                    s2 : #40 Present_state = s3;
                    s3 : #40 Present_state = s4;
                    s4 : #40 Present_state = s5;
                    s5 : #40 Present_state = s6;
                    s6          : #40 Present_state = s7;
                    s7          : #40 Present_state = s8;
                    s8          : #40 Present_state = s9;
                    s9 : #40 Present_state = s10;
                    s10 : #40 Present_state = s11;
```

```
                s11         : #40 Present_state = s12;
                s12 : #40 Present_state = s13;
                s13 : #40 Present_state = s14;
        endcase
end

always @(Present_state) begin // do the required job in each state
        case (Present_state) // assert the required signals in each clock cycle
                Default: begin
                        /*PCout <= 0; Zlowout <= 0; MDRout <= 0; // initialize the signals
                        R2out <= 0; R3out <= 0; MARin <= 0; Zin <= 0;
                        PCin <=0; MDRin <= 0; IRin <= 0; Yin <= 0;
                        IncPC <= 0; Read <= 0; AND <= 0;
                        R1in <= 0; R2in <= 0; R3in <= 0; Mdatain <= 32'h00000000;*/
                        {Write, strobe, BAOut,  Gra, Grb, Grc, Rin, Rout} <= 8'b0;
                        {PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout} <= 8'b0;
                        {MARin, Zin, PCin, MDRin, IRin, Yin, IncPC, Read, AND, HIin, InPortout, Loin, ZHighin,
Zlowin} <= 13'b0;
                        clear<=0;
                        Mdatain <= 32'h00000000;
                        BAOut <= 0;
                end

                sclear : begin
                        clear <= 1;
                        #10 clear <= 0;
                end

                s1 : begin //t0
                        PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus
                        #10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;
                end

                s2 : begin //t1
                        Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus
                        #10 Read <= 1; MDRin <= 1;
                        #20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;
                end

                s3 : begin //t2
                        MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus
                        #10 IRin <= 1;
                        #20 MDRout <= 0; IRin <= 0;
                end

                s4 : begin //t3
                        Grb <= 1; BAOut <= 1;
                        #10 Yin <= 1; //see the value of the reg to be added
                        #10 Grb <= 0; BAOut <= 0; Yin <= 0;
                end

                s5 : begin //t4 see C on the bus
                        Cout<= 1; ZHighin <= 1; Zlowin <= 1;
                        #10 op <= 5'b00011;
```

```
            #20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;
end

s6 : begin //t5
            Zlowout <= 1; Gra <= 1; Rin <= 1;// see the addition result on the bus
            #10 Zlowout <= 0; Gra <= 0; Rin <= 0;
end

s7 : begin //t0
            PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus
            #10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;
end

s8 : begin //t1
            Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus
            #10 Read <= 1; MDRin <= 1;
            #20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;
end

s9 : begin //t2
            MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus
            #10 IRin <= 1;
            #20 MDRout <= 0; IRin <= 0;
end

s10 : begin //t3
            Gra <= 1; Rout <= 1; OutPortin <= 1;
            #20 Gra <= 0; Rout <= 0; OutPortin <= 0;
end

s11 : begin //t0
            PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus
            #10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;
end

s12 : begin //t1
            Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus
            #10 Read <= 1; MDRin <= 1;
            #20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;
end

s13 : begin //t2
            MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus
            #10 IRin <= 1;
            #20 MDRout <= 0; IRin <= 0;
end

s14 : begin //t3
            input_data <= 32'habba;
            strobe <= 1;
            #5 strobe <= 0; Gra <= 1; Rin <= 1;  InPortout <= 1;
            #20 Gra <= 1; Rin <= 1; InPortout <= 0;
end
```

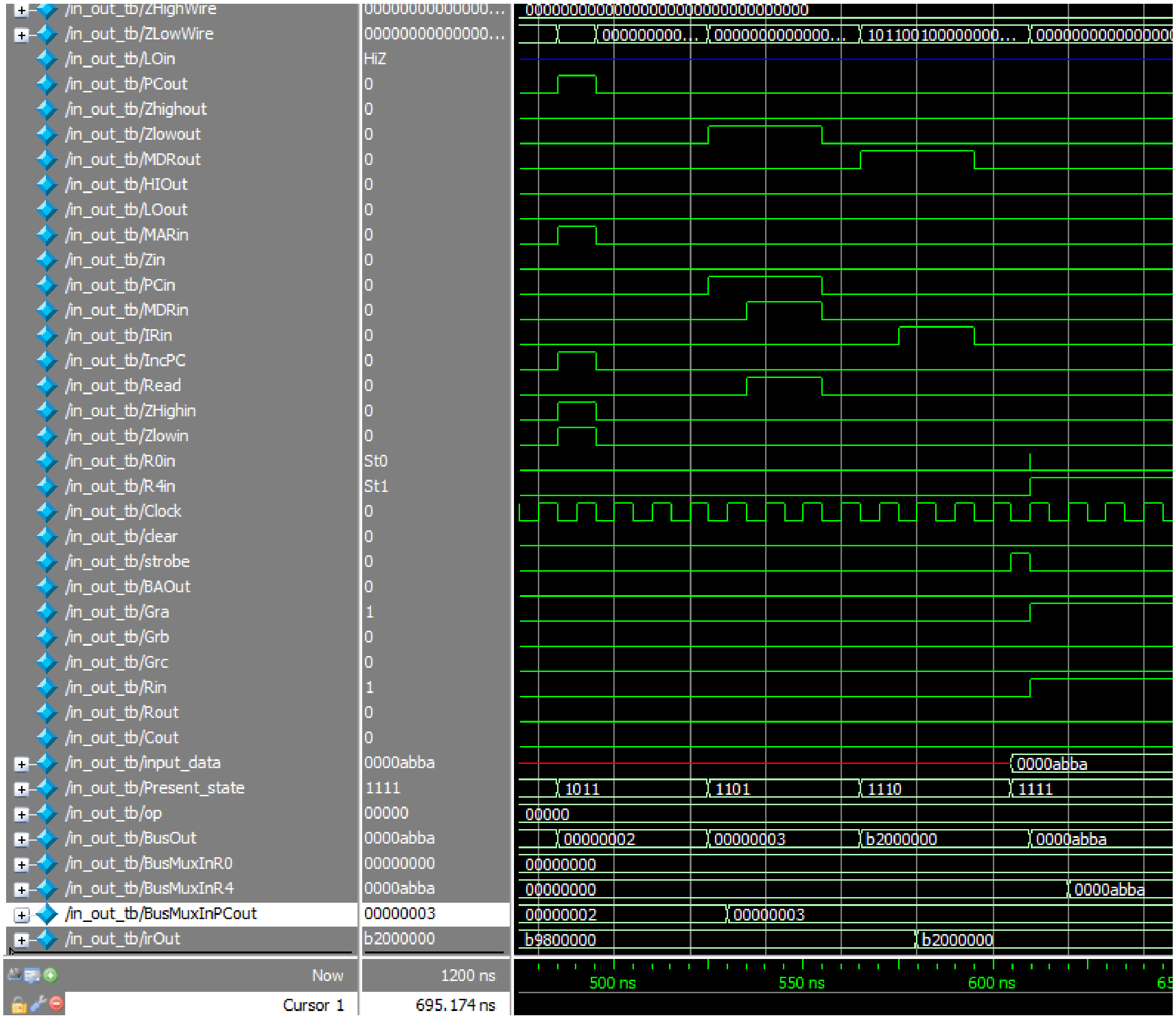
```
          endcase
end

endmodule
```

out R3



```
/in_out_tb/PCout          0
/in_out_tb/Zhighout       0
/in_out_tb/Zlowout        0
/in_out_tb/MDRout         0
/in_out_tb/InPortout      0
/in_out_tb/R0out          St0
/in_out_tb/R3out          St0
/in_out_tb/R4out          St0
/in_out_tb/MARin          0
/in_out_tb/PCin           0
/in_out_tb/MDRin          0
/in_out_tb/IRin           0
/in_out_tb/OutPortin      0
/in_out_tb/Yin            0
/in_out_tb/IncPC          0
/in_out_tb/Read           0
/in_out_tb/ZHighin        0
/in_out_tb/Zlowin         0
/in_out_tb/R0in           St0
/in_out_tb/R3in           St0
/in_out_tb/Clock          1
/in_out_tb/clear          0
/in_out_tb/strobe         0
/in_out_tb/BAOut          0
/in_out_tb/Gra            0
/in_out_tb/Grb            0
/in_out_tb/Grc            0
/in_out_tb/Rin            0
/in_out_tb/Rout           0
/in_out_tb/Cout           0
/in_out_tb/input_data     xxxxxxxxxxxxxx...
/in_out_tb/Present_state  1010
/in_out_tb/op             00000
/in_out_tb/BusOut         1
/in_out_tb/BusMuxInR3     1
/in_out_tb/BusMuxInZlow   2
/in_out_tb/BusMuxInPCout  2
/in_out_tb/output_data    1
Now                       1200 ns
Cursor 1                  472.295 ns
```

in R4



```
/in_out_tb/ZHighWire        00000000000000...
/in_out_tb/ZLowWire         00000000000000...
/in_out_tb/LOin             HiZ
/in_out_tb/PCout            0
/in_out_tb/Zhighout         0
/in_out_tb/Zlowout          0
/in_out_tb/MDRout           0
/in_out_tb/HIOut            0
/in_out_tb/LOout            0
/in_out_tb/MARin            0
/in_out_tb/Zin              0
/in_out_tb/PCin             0
/in_out_tb/MDRin            0
/in_out_tb/IRin             0
/in_out_tb/IncPC            0
/in_out_tb/Read             0
/in_out_tb/ZHighin          0
/in_out_tb/Zlowin           0
/in_out_tb/R0in             St0
/in_out_tb/R4in             St1
/in_out_tb/Clock            0
/in_out_tb/clear            0
/in_out_tb/strobe           0
/in_out_tb/BAOut            0
/in_out_tb/Gra              1
/in_out_tb/Grb              0
/in_out_tb/Grc              0
/in_out_tb/Rin              1
/in_out_tb/Rout             0
/in_out_tb/Cout             0
/in_out_tb/input_data       0000abba
/in_out_tb/Present_state    1111
/in_out_tb/op               00000
/in_out_tb/BusOut           0000abba
/in_out_tb/BusMuxInR0       00000000
/in_out_tb/BusMuxInR4       0000abba
/in_out_tb/BusMuxInPCout    00000003
/in_out_tb/irOut            b2000000

Now         1200 ns
Cursor 1    695.174 ns
```