ELEC 374 – Digital Systems Engineering

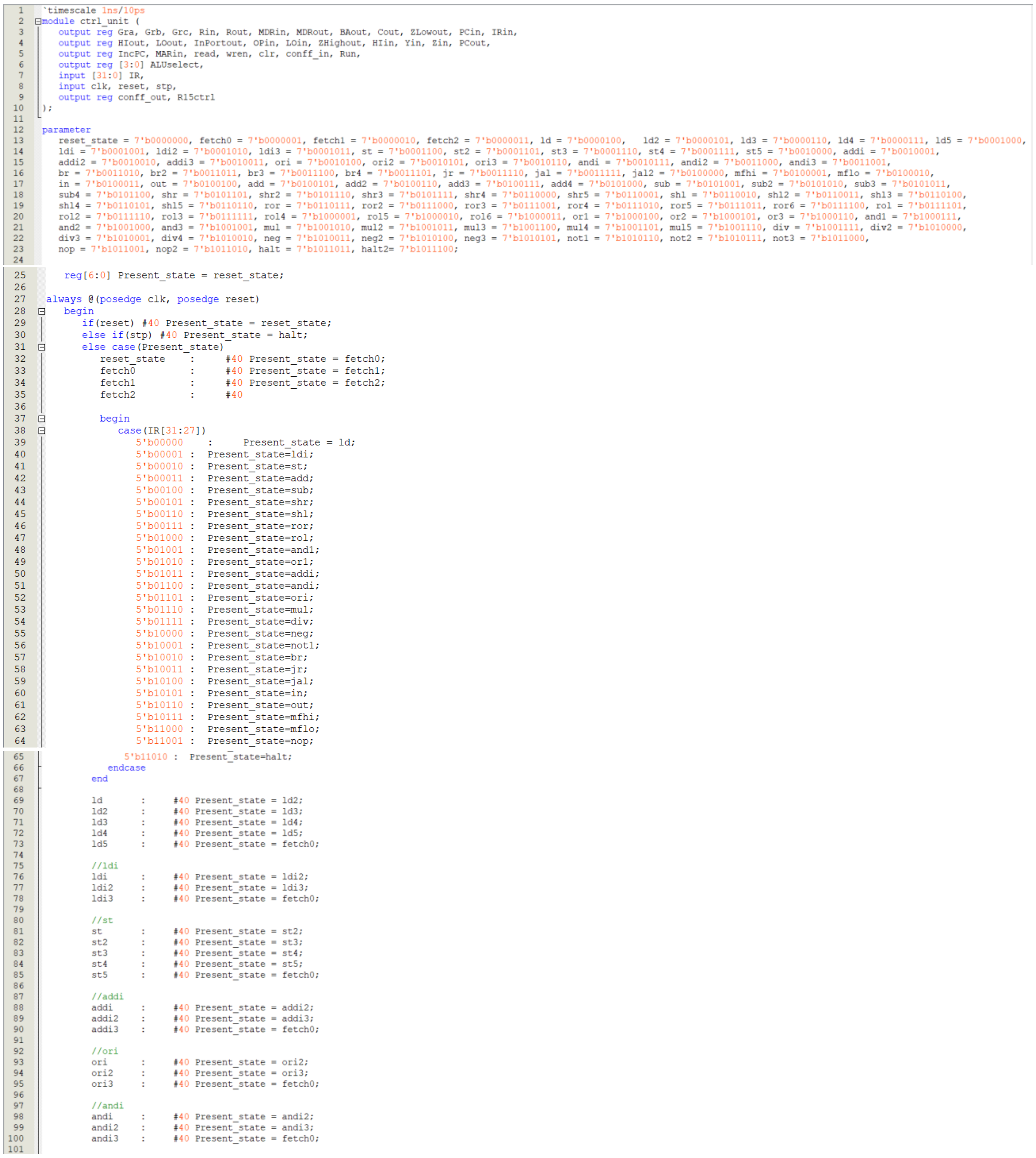
Phase 3 Report

Group 18

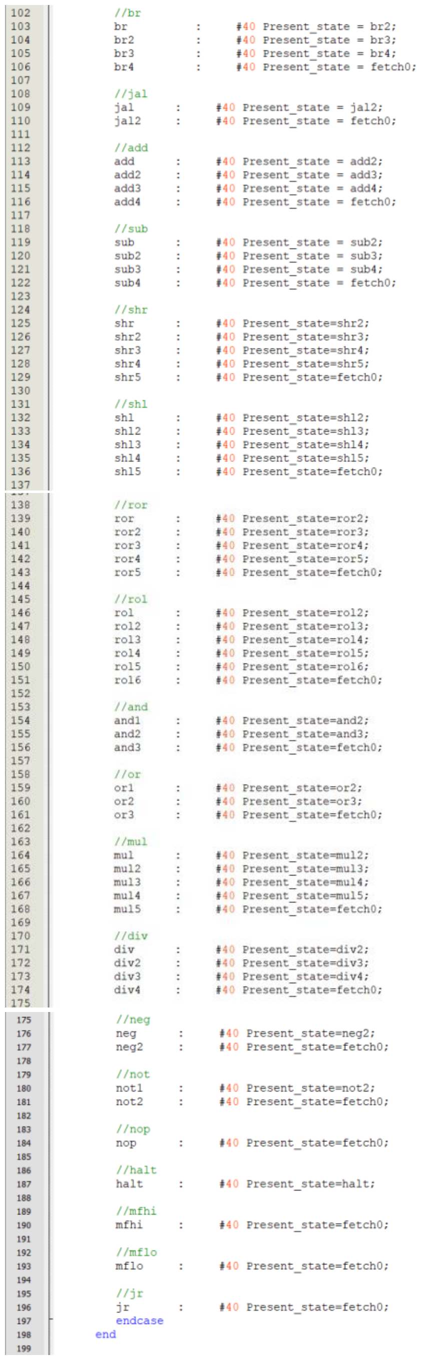
Divaydeep Singh	**20143859**
Lucas Austin	**20061953**

Code

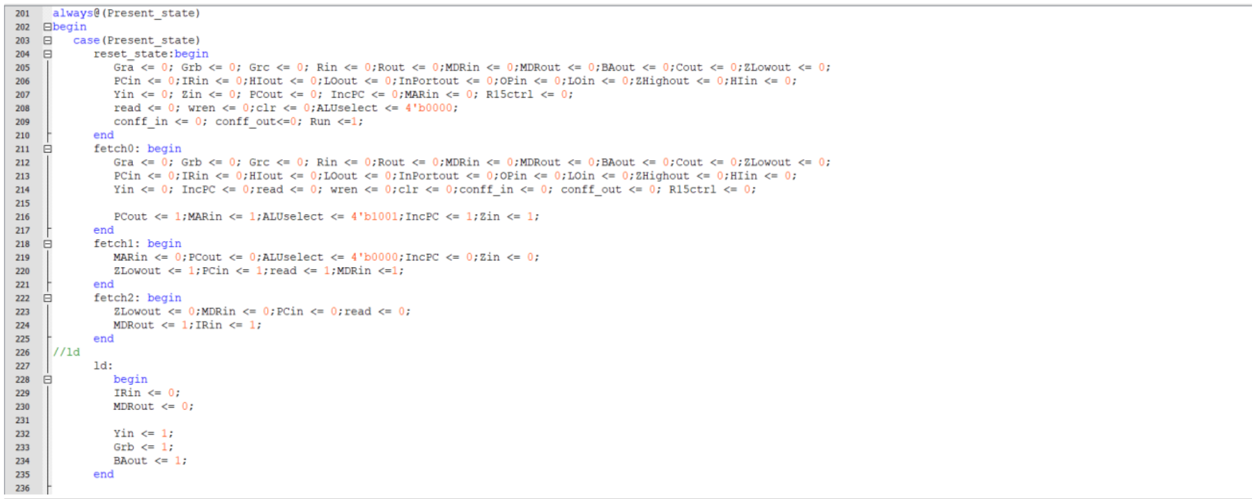
Ctrl_unit.v



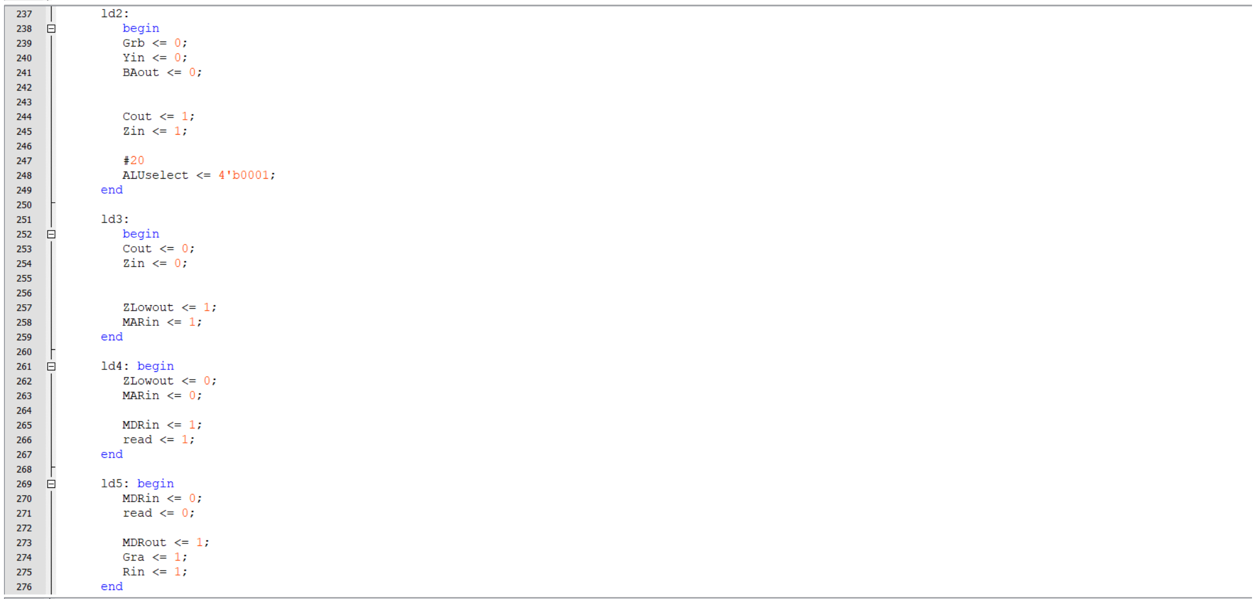
```verilog
`timescale 1ns/10ps
module ctrl_unit (
    output reg Gra, Grb, Grc, Rin, Rout, MDRin, MDRout, BAout, Cout, ZLowout, PCin, IRin,
    output reg HIout, LOout, InPortout, OPin, LOin, ZHighout, HIin, Yin, Zin, PCout,
    output reg IncPC, MARin, read, wren, clr, conff_in, Run,
    output reg [3:0] ALUselect,
    input [31:0] IR,
    input clk, reset, stp,
    output reg conff_out, R15ctrl
);

parameter
    reset_state = 7'b0000000, fetch0 = 7'b0000001, fetch1 = 7'b0000010, fetch2 = 7'b0000011, ld = 7'b0000100,   ld2 = 7'b0000101, ld3 = 7'b0000110, ld4 = 7'b0000111, ld5 = 7'b0001000,
    ldi = 7'b0001001, ldi2 = 7'b0001010, ldi3 = 7'b0001011, st = 7'b0001100, st2 = 7'b0001101, st3 = 7'b0001110, st4 = 7'b0001111, st5 = 7'b0010000, addi = 7'b0010001,
    addi2 = 7'b0010010, addi3 = 7'b0010011, ori = 7'b0010100, ori2 = 7'b0010101, ori3 = 7'b0010110, andi = 7'b0010111, andi2 = 7'b0011000, andi3 = 7'b0011001,
    br = 7'b0011010, br2 = 7'b0011011, br3 = 7'b0011100, br4 = 7'b0011101, jr = 7'b0011110, jal = 7'b0011111, jal2 = 7'b0100000, mfhi = 7'b0100001, mflo = 7'b0100010,
    in = 7'b0100011, out = 7'b0100100, add = 7'b0100101, add2 = 7'b0100110, add3 = 7'b0100111, add4 = 7'b0101000, sub = 7'b0101001, sub2 = 7'b0101010, sub3 = 7'b0101011,
    sub4 = 7'b0101100, shr = 7'b0101101, shr2 = 7'b0101110, shr3 = 7'b0101111, shr4 = 7'b0110000, shr5 = 7'b0110001, shl = 7'b0110010, shl2 = 7'b0110011, shl3 = 7'b0110100,
    shl4 = 7'b0110101, shl5 = 7'b0110110, ror = 7'b0110111, ror2 = 7'b0111000, ror3 = 7'b0111001, ror4 = 7'b0111010, ror5 = 7'b0111011, ror6 = 7'b0111100, rol = 7'b0111101,
    rol2 = 7'b0111110, rol3 = 7'b0111111, rol4 = 7'b1000001, rol5 = 7'b1000010, rol6 = 7'b1000011, or1 = 7'b1000100, or2 = 7'b1000101, or3 = 7'b1000110, and1 = 7'b1000111,
    and2 = 7'b1001000, and3 = 7'b1001001, mul = 7'b1001010, mul2 = 7'b1001011, mul3 = 7'b1001100, mul4 = 7'b1001101, mul5 = 7'b1001110, div = 7'b1001111, div2 = 7'b1010000,
    div3 = 7'b1010001, div4 = 7'b1010010, neg = 7'b1010011, neg2 = 7'b1010100, neg3 = 7'b1010101, not1 = 7'b1010110, not2 = 7'b1010111, not3 = 7'b1011000,
    nop = 7'b1011001, nop2 = 7'b1011010, halt = 7'b1011011, halt2= 7'b1011100;

    reg[6:0] Present_state = reset_state;

always @(posedge clk, posedge reset)
    begin
        if(reset) #40 Present_state = reset_state;
        else if(stp) #40 Present_state = halt;
        else case(Present_state)
            reset_state    :     #40 Present_state = fetch0;
            fetch0         :     #40 Present_state = fetch1;
            fetch1         :     #40 Present_state = fetch2;
            fetch2         :     #40

            begin
                case(IR[31:27])
                    5'b00000    :     Present_state = ld;
                    5'b00001 :  Present_state=ldi;
                    5'b00010 :  Present_state=st;
                    5'b00011 :  Present_state=add;
                    5'b00100 :  Present_state=sub;
                    5'b00101 :  Present_state=shr;
                    5'b00110 :  Present_state=shl;
                    5'b00111 :  Present_state=ror;
                    5'b01000 :  Present_state=rol;
                    5'b01001 :  Present_state=and1;
                    5'b01010 :  Present_state=or1;
                    5'b01011 :  Present_state=addi;
                    5'b01100 :  Present_state=andi;
                    5'b01101 :  Present_state=ori;
                    5'b01110 :  Present_state=mul;
                    5'b01111 :  Present_state=div;
                    5'b10000 :  Present_state=neg;
                    5'b10001 :  Present_state=not1;
                    5'b10010 :  Present_state=br;
                    5'b10011 :  Present_state=jr;
                    5'b10100 :  Present_state=jal;
                    5'b10101 :  Present_state=in;
                    5'b10110 :  Present_state=out;
                    5'b10111 :  Present_state=mfhi;
                    5'b11000 :  Present_state=mflo;
                    5'b11001 :  Present_state=nop;
                    5'b11010 :  Present_state=halt;
                endcase
            end

            ld       :     #40 Present_state = ld2;
            ld2      :     #40 Present_state = ld3;
            ld3      :     #40 Present_state = ld4;
            ld4      :     #40 Present_state = ld5;
            ld5      :     #40 Present_state = fetch0;

            //ldi
            ldi      :     #40 Present_state = ldi2;
            ldi2     :     #40 Present_state = ldi3;
            ldi3     :     #40 Present_state = fetch0;

            //st
            st       :     #40 Present_state = st2;
            st2      :     #40 Present_state = st3;
            st3      :     #40 Present_state = st4;
            st4      :     #40 Present_state = st5;
            st5      :     #40 Present_state = fetch0;

            //addi
            addi     :     #40 Present_state = addi2;
            addi2    :     #40 Present_state = addi3;
            addi3    :     #40 Present_state = fetch0;

            //ori
            ori      :     #40 Present_state = ori2;
            ori2     :     #40 Present_state = ori3;
            ori3     :     #40 Present_state = fetch0;

            //andi
            andi     :     #40 Present_state = andi2;
            andi2    :     #40 Present_state = andi3;
            andi3    :     #40 Present_state = fetch0;
```



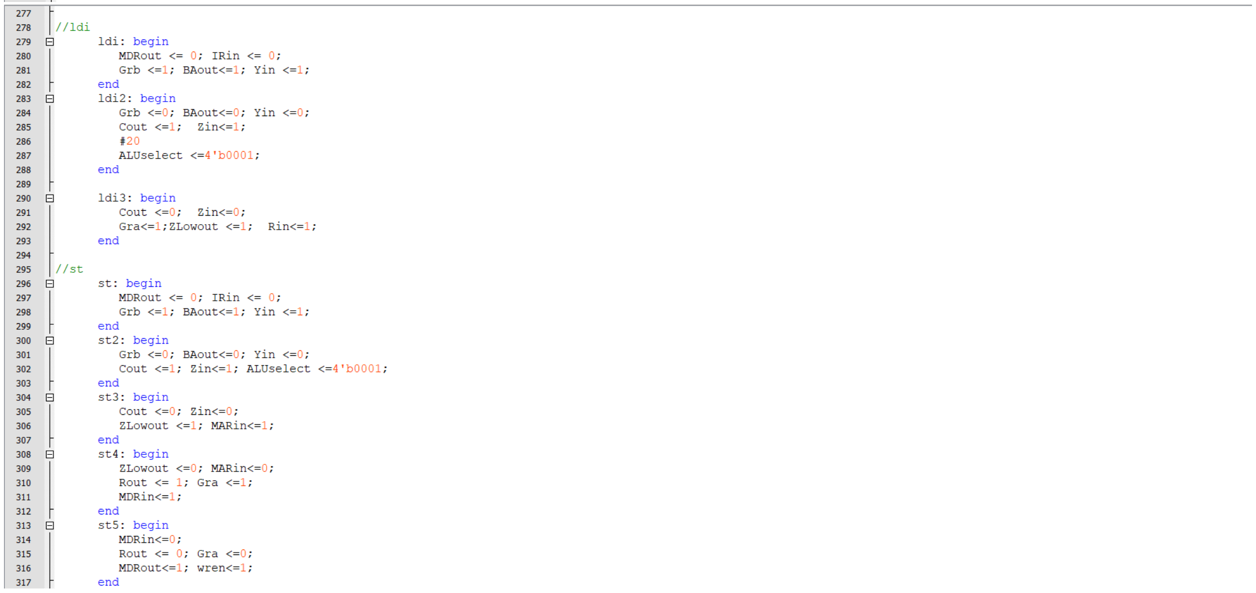
```
        //br
        br          :      #40 Present_state = br2;
        br2         :      #40 Present_state = br3;
        br3         :      #40 Present_state = br4;
        br4         :      #40 Present_state = fetch0;

        //jal
        jal      :     #40 Present_state = jal2;
        jal2     :     #40 Present_state = fetch0;

        //add
        add      :     #40 Present_state = add2;
        add2     :     #40 Present_state = add3;
        add3     :     #40 Present_state = add4;
        add4     :     #40 Present_state = fetch0;

        //sub
        sub      :     #40 Present_state = sub2;
        sub2     :     #40 Present_state = sub3;
        sub3     :     #40 Present_state = sub4;
        sub4     :     #40 Present_state = fetch0;

        //shr
        shr      :     #40 Present_state=shr2;
        shr2     :     #40 Present_state=shr3;
        shr3     :     #40 Present_state=shr4;
        shr4     :     #40 Present_state=shr5;
        shr5     :     #40 Present_state=fetch0;

        //shl
        shl      :     #40 Present_state=shl2;
        shl2     :     #40 Present_state=shl3;
        shl3     :     #40 Present_state=shl4;
        shl4     :     #40 Present_state=shl5;
        shl5     :     #40 Present_state=fetch0;

        //ror
        ror      :     #40 Present_state=ror2;
        ror2     :     #40 Present_state=ror3;
        ror3     :     #40 Present_state=ror4;
        ror4     :     #40 Present_state=ror5;
        ror5     :     #40 Present_state=fetch0;

        //rol
        rol      :     #40 Present_state=rol2;
        rol2     :     #40 Present_state=rol3;
        rol3     :     #40 Present_state=rol4;
        rol4     :     #40 Present_state=rol5;
        rol5     :     #40 Present_state=rol6;
        rol6     :     #40 Present_state=fetch0;

        //and
        and1     :     #40 Present_state=and2;
        and2     :     #40 Present_state=and3;
        and3     :     #40 Present_state=fetch0;

        //or
        or1      :     #40 Present_state=or2;
        or2      :     #40 Present_state=or3;
        or3      :     #40 Present_state=fetch0;

        //mul
        mul      :     #40 Present_state=mul2;
        mul2     :     #40 Present_state=mul3;
        mul3     :     #40 Present_state=mul4;
        mul4     :     #40 Present_state=mul5;
        mul5     :     #40 Present_state=fetch0;

        //div
        div      :     #40 Present_state=div2;
        div2     :     #40 Present_state=div3;
        div3     :     #40 Present_state=div4;
        div4     :     #40 Present_state=fetch0;

        //neg
        neg      :     #40 Present_state=neg2;
        neg2     :     #40 Present_state=fetch0;

        //not
        not1     :     #40 Present_state=not2;
        not2     :     #40 Present_state=fetch0;

        //nop
        nop      :     #40 Present_state=fetch0;

        //halt
        halt     :     #40 Present_state=halt;

        //mfhi
        mfhi     :     #40 Present_state=fetch0;

        //mflo
        mflo     :     #40 Present_state=fetch0;

        //jr
        jr       :     #40 Present_state=fetch0;
        endcase
    end
```



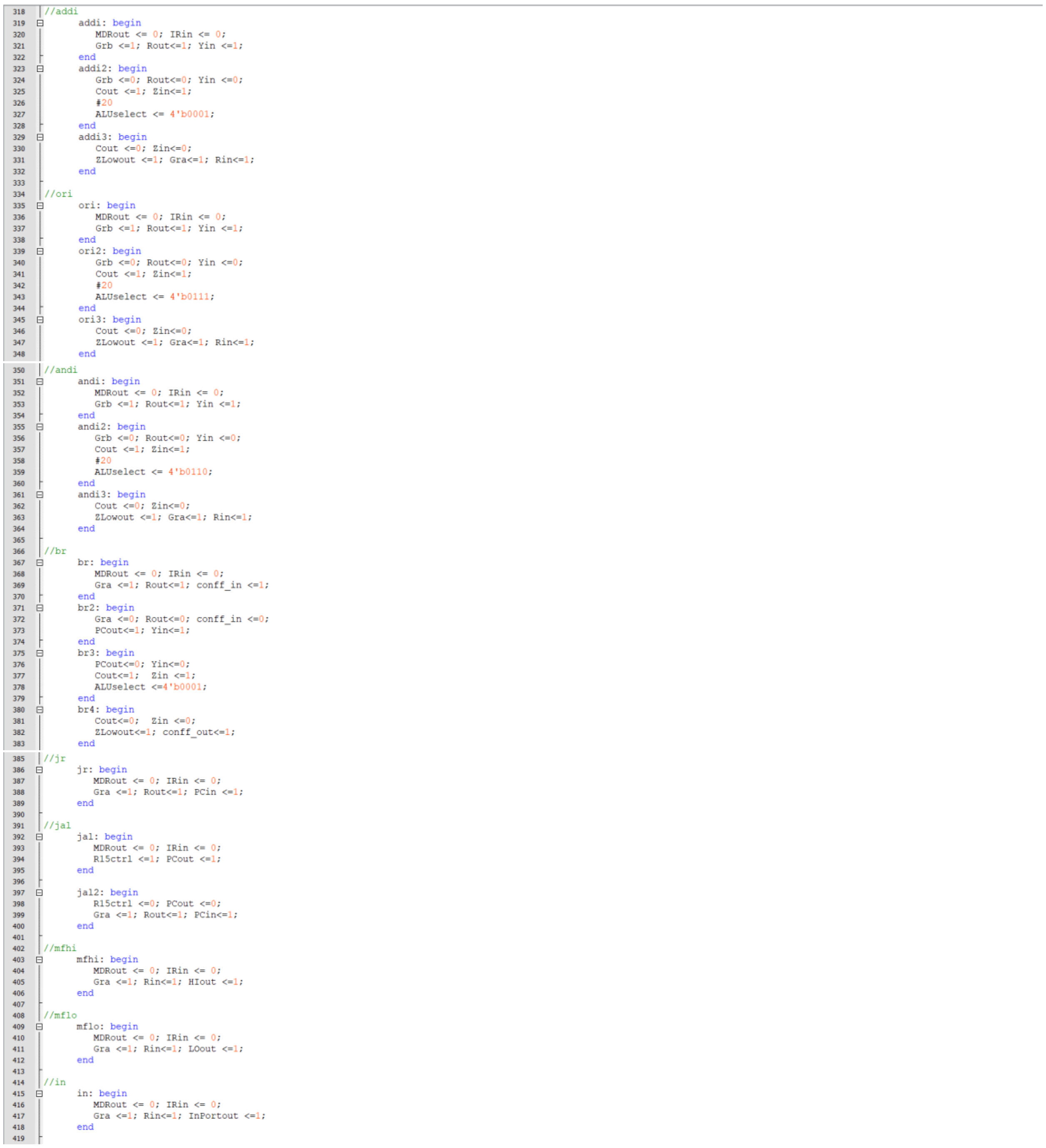
```verilog
always@(Present_state)
begin
   case(Present_state)
      reset_state:begin
         Gra <= 0; Grb <= 0; Grc <= 0; Rin <= 0;Rout <= 0;MDRin <= 0;MDRout <= 0;BAout <= 0;Cout <= 0;ZLowout <= 0;
         PCin <= 0;IRin <= 0;HIout <= 0;LOout <= 0;InPortout <= 0;OPin <= 0;LOin <= 0;ZHighout <= 0;HIin <= 0;
         Yin <= 0; Zin <= 0; PCout <= 0; IncPC <= 0;MARin <= 0; R15ctrl <= 0;
         read <= 0; wren <= 0;clr <= 0;ALUselect <= 4'b0000;
         conff_in <= 0; conff_out<=0; Run <=1;
      end
      fetch0: begin
         Gra <= 0; Grb <= 0; Grc <= 0; Rin <= 0;Rout <= 0;MDRin <= 0;MDRout <= 0;BAout <= 0;Cout <= 0;ZLowout <= 0;
         PCin <= 0;IRin <= 0;HIout <= 0;LOout <= 0;InPortout <= 0;OPin <= 0;LOin <= 0;ZHighout <= 0;HIin <= 0;
         Yin <= 0; IncPC <= 0;read <= 0; wren <= 0;clr <= 0;conff_in <= 0; conff_out <= 0; R15ctrl <= 0;

         PCout <= 1;MARin <= 1;ALUselect <= 4'b1001;IncPC <= 1;Zin <= 1;
      end
      fetch1: begin
         MARin <= 0;PCout <= 0;ALUselect <= 4'b0000;IncPC <= 0;Zin <= 0;
         ZLowout <= 1;PCin <= 1;read <= 1;MDRin <=1;
      end
      fetch2: begin
         ZLowout <= 0;MDRin <= 0;PCin <= 0;read <= 0;
         MDRout <= 1;IRin <= 1;
      end
//ld
      ld:
         begin
         IRin <= 0;
         MDRout <= 0;

         Yin <= 1;
         Grb <= 1;
         BAout <= 1;
      end

```



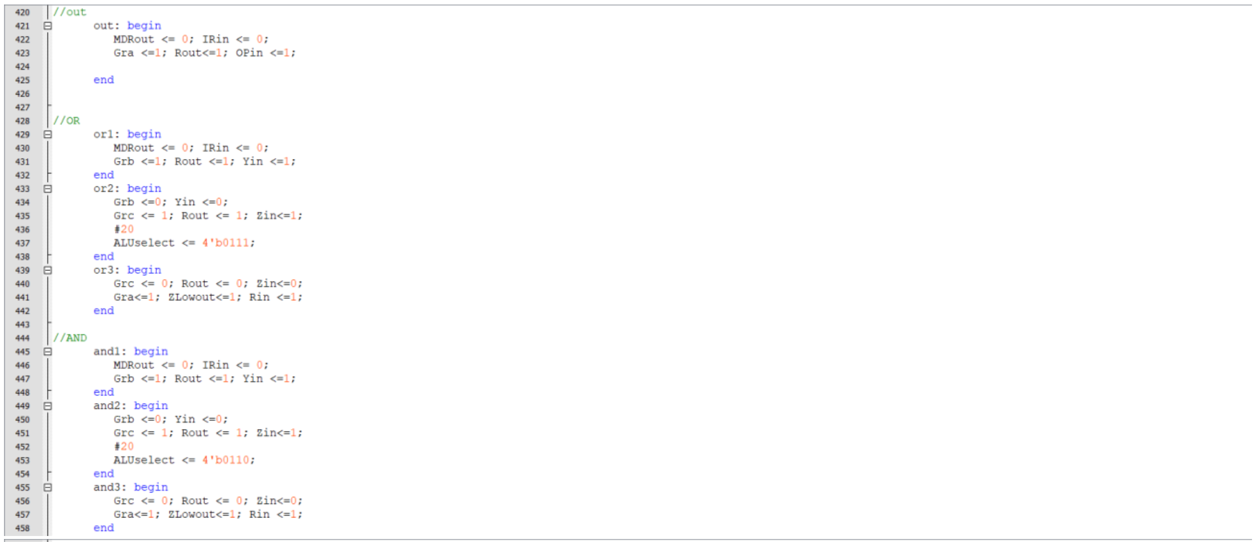
```verilog
      ld2:
         begin
         Grb <= 0;
         Yin <= 0;
         BAout <= 0;


         Cout <= 1;
         Zin <= 1;

         #20
         ALUselect <= 4'b0001;
      end

      ld3:
         begin
         Cout <= 0;
         Zin <= 0;


         ZLowout <= 1;
         MARin <= 1;
      end

      ld4: begin
         ZLowout <= 0;
         MARin <= 0;

         MDRin <= 1;
         read <= 1;
      end

      ld5: begin
         MDRin <= 0;
         read <= 0;

         MDRout <= 1;
         Gra <= 1;
         Rin <= 1;
      end
```



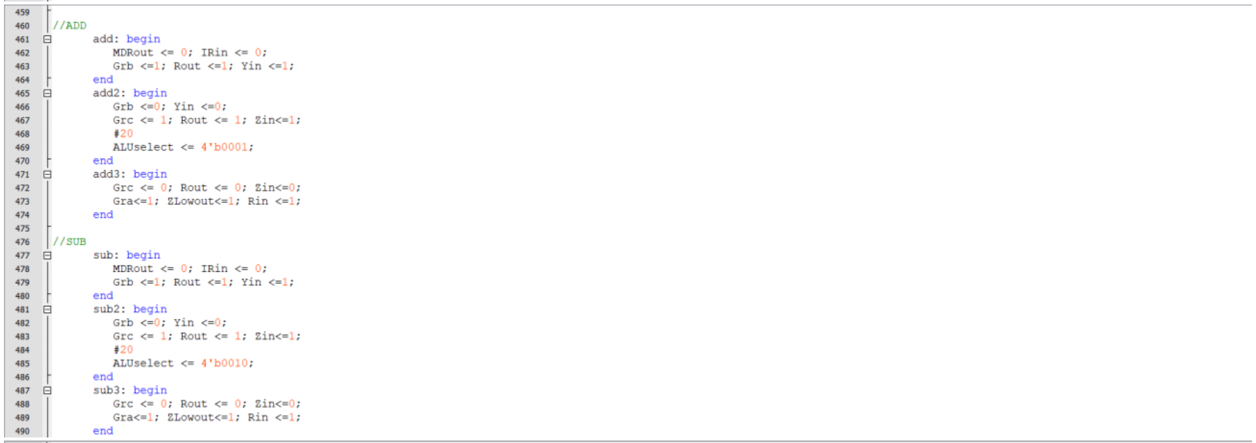
```verilog

//ldi
      ldi: begin
         MDRout <= 0; IRin <= 0;
         Grb <=1; BAout<=1; Yin <=1;
      end
      ldi2: begin
         Grb <=0; BAout<=0; Yin <=0;
         Cout <=1;  Zin<=1;
         #20
         ALUselect <=4'b0001;
      end

      ldi3: begin
         Cout <=0;  Zin<=0;
         Gra<=1;ZLowout <=1;  Rin<=1;
      end

//st
      st: begin
         MDRout <= 0; IRin <= 0;
         Grb <=1; BAout<=1; Yin <=1;
      end
      st2: begin
         Grb <=0; BAout<=0; Yin <=0;
         Cout <=1; Zin<=1; ALUselect <=4'b0001;
      end
      st3: begin
         Cout <=0; Zin<=0;
         ZLowout <=1; MARin<=1;
      end
      st4: begin
         ZLowout <=0; MARin<=0;
         Rout <= 1; Gra <=1;
         MDRin<=1;
      end
      st5: begin
         MDRin<=0;
         Rout <= 0; Gra <=0;
         MDRout<=1; wren<=1;
      end
```



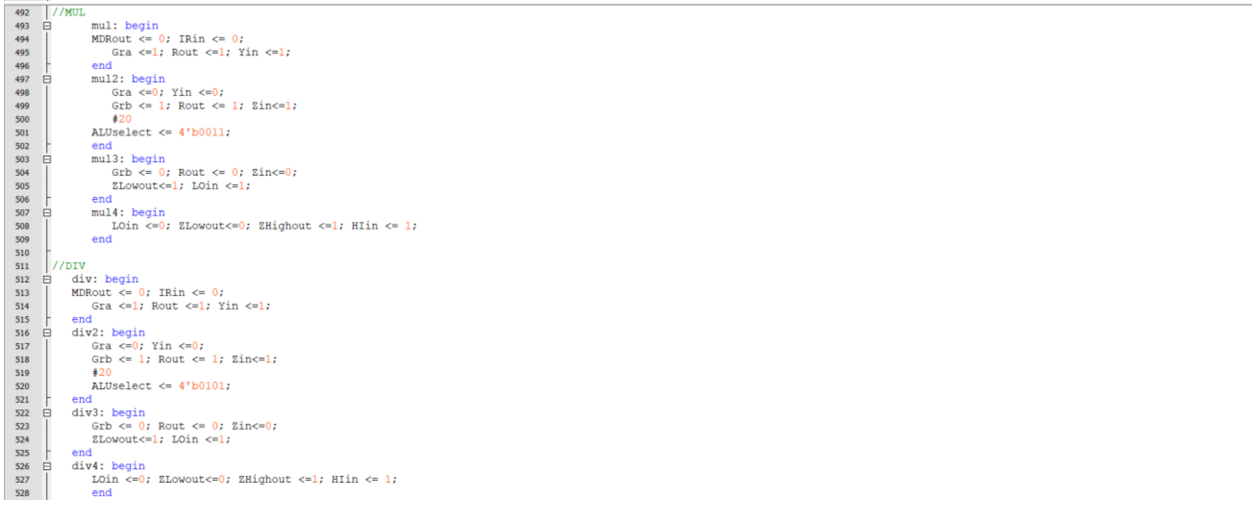
```
//addi
      addi: begin
         MDRout <= 0; IRin <= 0;
         Grb <=1; Rout<=1; Yin <=1;
      end
      addi2: begin
         Grb <=0; Rout<=0; Yin <=0;
         Cout <=1; Zin<=1;
         #20
         ALUselect <= 4'b0001;
      end
      addi3: begin
         Cout <=0; Zin<=0;
         ZLowout <=1; Gra<=1; Rin<=1;
      end

//ori
      ori: begin
         MDRout <= 0; IRin <= 0;
         Grb <=1; Rout<=1; Yin <=1;
      end
      ori2: begin
         Grb <=0; Rout<=0; Yin <=0;
         Cout <=1; Zin<=1;
         #20
         ALUselect <= 4'b0111;
      end
      ori3: begin
         Cout <=0; Zin<=0;
         ZLowout <=1; Gra<=1; Rin<=1;
      end
//andi
      andi: begin
         MDRout <= 0; IRin <= 0;
         Grb <=1; Rout<=1; Yin <=1;
      end
      andi2: begin
         Grb <=0; Rout<=0; Yin <=0;
         Cout <=1; Zin<=1;
         #20
         ALUselect <= 4'b0110;
      end
      andi3: begin
         Cout <=0; Zin<=0;
         ZLowout <=1; Gra<=1; Rin<=1;
      end

//br
      br: begin
         MDRout <= 0; IRin <= 0;
         Gra <=1; Rout<=1; conff_in <=1;
      end
      br2: begin
         Gra <=0; Rout<=0; conff_in <=0;
         PCout<=1; Yin<=1;
      end
      br3: begin
         PCout<=0; Yin<=0;
         Cout<=1;  Zin <=1;
         ALUselect <=4'b0001;
      end
      br4: begin
         Cout<=0;  Zin <=0;
         ZLowout<=1; conff_out<=1;
      end
//jr
      jr: begin
         MDRout <= 0; IRin <= 0;
         Gra <=1; Rout<=1; PCin <=1;
      end

//jal
      jal: begin
         MDRout <= 0; IRin <= 0;
         R15ctrl <=1; PCout <=1;
      end

      jal2: begin
         R15ctrl <=0; PCout <=0;
         Gra <=1; Rout<=1; PCin<=1;
      end

//mfhi
      mfhi: begin
         MDRout <= 0; IRin <= 0;
         Gra <=1; Rin<=1; HIout <=1;
      end

//mflo
      mflo: begin
         MDRout <= 0; IRin <= 0;
         Gra <=1; Rin<=1; LOout <=1;
      end

//in
      in: begin
         MDRout <= 0; IRin <= 0;
         Gra <=1; Rin<=1; InPortout <=1;
      end

```



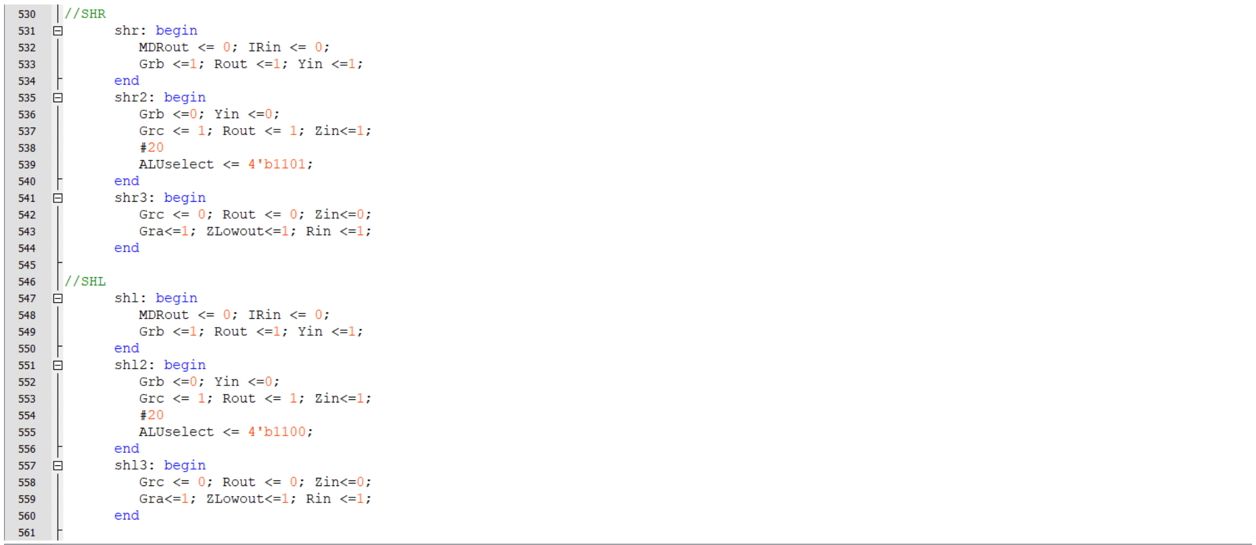
```
//out
      out: begin
         MDRout <= 0; IRin <= 0;
         Gra <=1; Rout<=1; OPin <=1;

      end


//OR
      or1: begin
         MDRout <= 0; IRin <= 0;
         Grb <=1; Rout <=1; Yin <=1;
      end
      or2: begin
         Grb <=0; Yin <=0;
         Grc <= 1; Rout <= 1; Zin<=1;
         #20
         ALUselect <= 4'b0111;
      end
      or3: begin
         Grc <= 0; Rout <= 0; Zin<=0;
         Gra<=1; ZLowout<=1; Rin <=1;
      end

//AND
      and1: begin
         MDRout <= 0; IRin <= 0;
         Grb <=1; Rout <=1; Yin <=1;
      end
      and2: begin
         Grb <=0; Yin <=0;
         Grc <= 1; Rout <= 1; Zin<=1;
         #20
         ALUselect <= 4'b0110;
      end
      and3: begin
         Grc <= 0; Rout <= 0; Zin<=0;
         Gra<=1; ZLowout<=1; Rin <=1;
      end
```



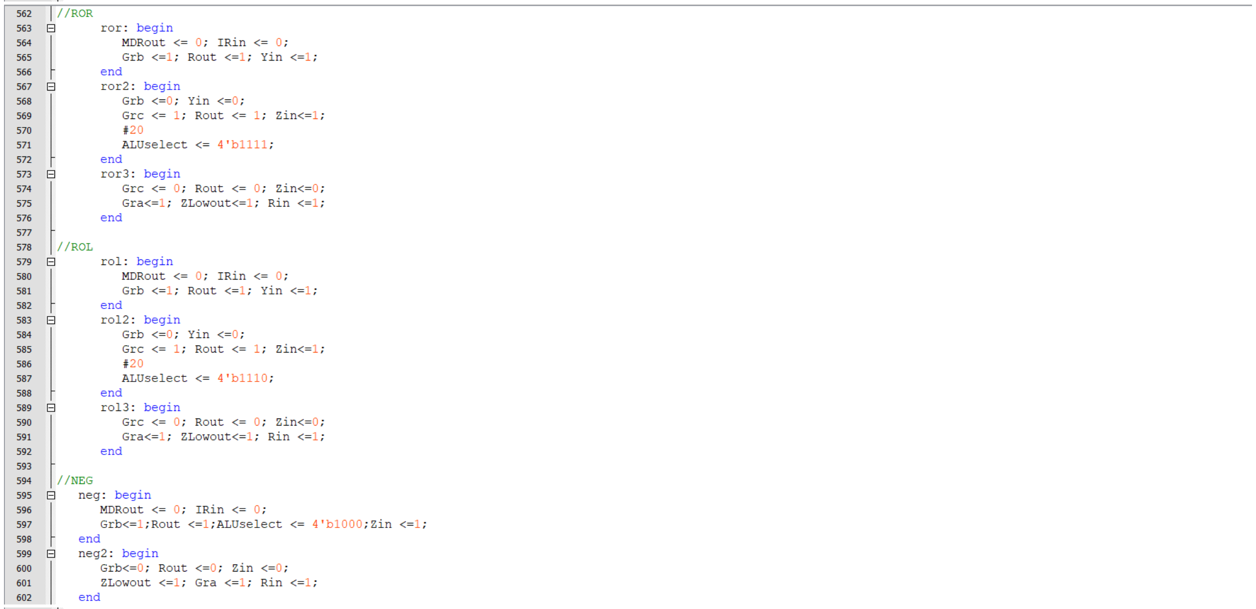
```

//ADD
      add: begin
         MDRout <= 0; IRin <= 0;
         Grb <=1; Rout <=1; Yin <=1;
      end
      add2: begin
         Grb <=0; Yin <=0;
         Grc <= 1; Rout <= 1; Zin<=1;
         #20
         ALUselect <= 4'b0001;
      end
      add3: begin
         Grc <= 0; Rout <= 0; Zin<=0;
         Gra<=1; ZLowout<=1; Rin <=1;
      end

//SUB
      sub: begin
         MDRout <= 0; IRin <= 0;
         Grb <=1; Rout <=1; Yin <=1;
      end
      sub2: begin
         Grb <=0; Yin <=0;
         Grc <= 1; Rout <= 1; Zin<=1;
         #20
         ALUselect <= 4'b0010;
      end
      sub3: begin
         Grc <= 0; Rout <= 0; Zin<=0;
         Gra<=1; ZLowout<=1; Rin <=1;
      end
```



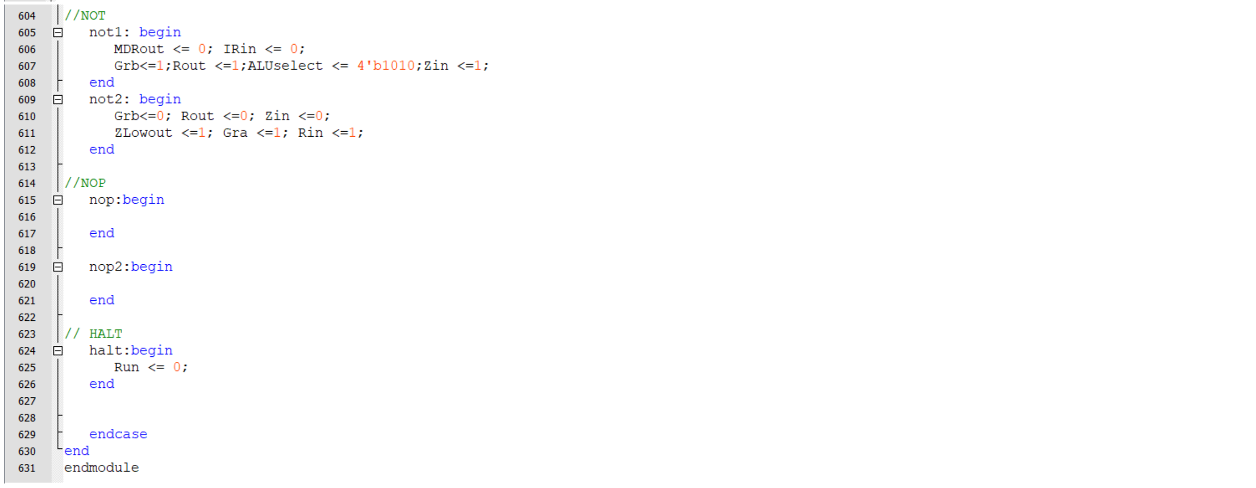
```
//MUL
      mul: begin
      MDRout <= 0; IRin <= 0;
         Gra <=1; Rout <=1; Yin <=1;
      end
      mul2: begin
         Gra <=0; Yin <=0;
         Grb <= 1; Rout <= 1; Zin<=1;
         #20
      ALUselect <= 4'b0011;
      end
      mul3: begin
         Grb <= 0; Rout <= 0; Zin<=0;
         ZLowout<=1; LOin <=1;
      end
      mul4: begin
         LOin <=0; ZLowout<=0; ZHighout <=1; HIin <= 1;
      end

//DIV
   div: begin
   MDRout <= 0; IRin <= 0;
      Gra <=1; Rout <=1; Yin <=1;
   end
   div2: begin
      Gra <=0; Yin <=0;
      Grb <= 1; Rout <= 1; Zin<=1;
      #20
      ALUselect <= 4'b0101;
   end
   div3: begin
      Grb <= 0; Rout <= 0; Zin<=0;
      ZLowout<=1; LOin <=1;
   end
   div4: begin
      LOin <=0; ZLowout<=0; ZHighout <=1; HIin <= 1;
      end
```



```
//SHR
      shr: begin
         MDRout <= 0; IRin <= 0;
         Grb <=1; Rout <=1; Yin <=1;
      end
      shr2: begin
         Grb <=0; Yin <=0;
         Grc <= 1; Rout <= 1; Zin<=1;
         #20
         ALUselect <= 4'b1101;
      end
      shr3: begin
         Grc <= 0; Rout <= 0; Zin<=0;
         Gra<=1; ZLowout<=1; Rin <=1;
      end

//SHL
      shl: begin
         MDRout <= 0; IRin <= 0;
         Grb <=1; Rout <=1; Yin <=1;
      end
      shl2: begin
         Grb <=0; Yin <=0;
         Grc <= 1; Rout <= 1; Zin<=1;
         #20
         ALUselect <= 4'b1100;
      end
      shl3: begin
         Grc <= 0; Rout <= 0; Zin<=0;
         Gra<=1; ZLowout<=1; Rin <=1;
      end

```

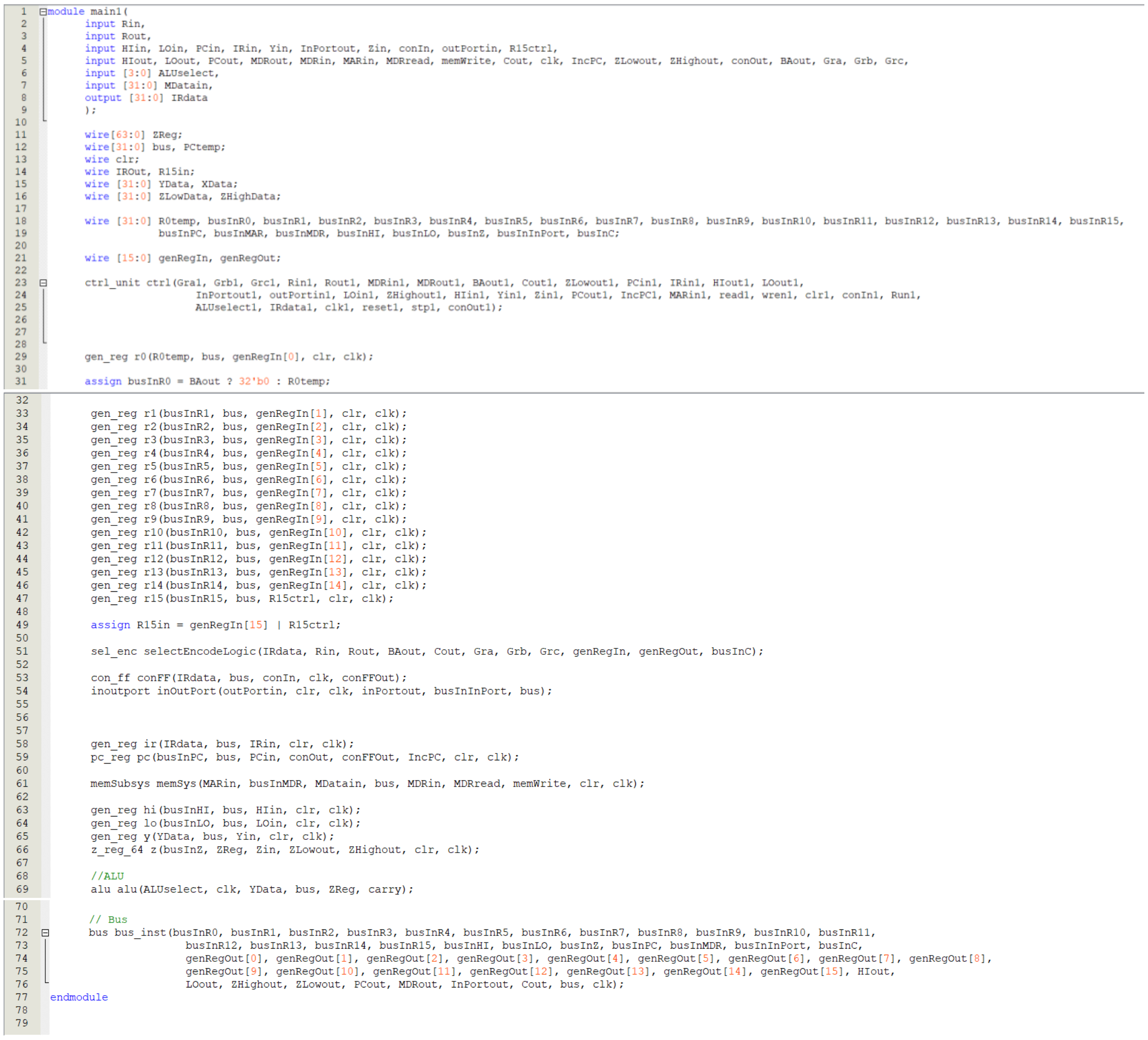


```
//ROR
      ror: begin
         MDRout <= 0; IRin <= 0;
         Grb <=1; Rout <=1; Yin <=1;
      end
      ror2: begin
         Grb <=0; Yin <=0;
         Grc <= 1; Rout <= 1; Zin<=1;
         #20
         ALUselect <= 4'b1111;
      end
      ror3: begin
         Grc <= 0; Rout <= 0; Zin<=0;
         Gra<=1; ZLowout<=1; Rin <=1;
      end

//ROL
      rol: begin
         MDRout <= 0; IRin <= 0;
         Grb <=1; Rout <=1; Yin <=1;
      end
      rol2: begin
         Grb <=0; Yin <=0;
         Grc <= 1; Rout <= 1; Zin<=1;
         #20
         ALUselect <= 4'b1110;
      end
      rol3: begin
         Grc <= 0; Rout <= 0; Zin<=0;
         Gra<=1; ZLowout<=1; Rin <=1;
      end

//NEG
   neg: begin
      MDRout <= 0; IRin <= 0;
      Grb<=1;Rout <=1;ALUselect <= 4'b1000;Zin <=1;
   end
   neg2: begin
      Grb<=0; Rout <=0; Zin <=0;
      ZLowout <=1; Gra <=1; Rin <=1;
   end
```



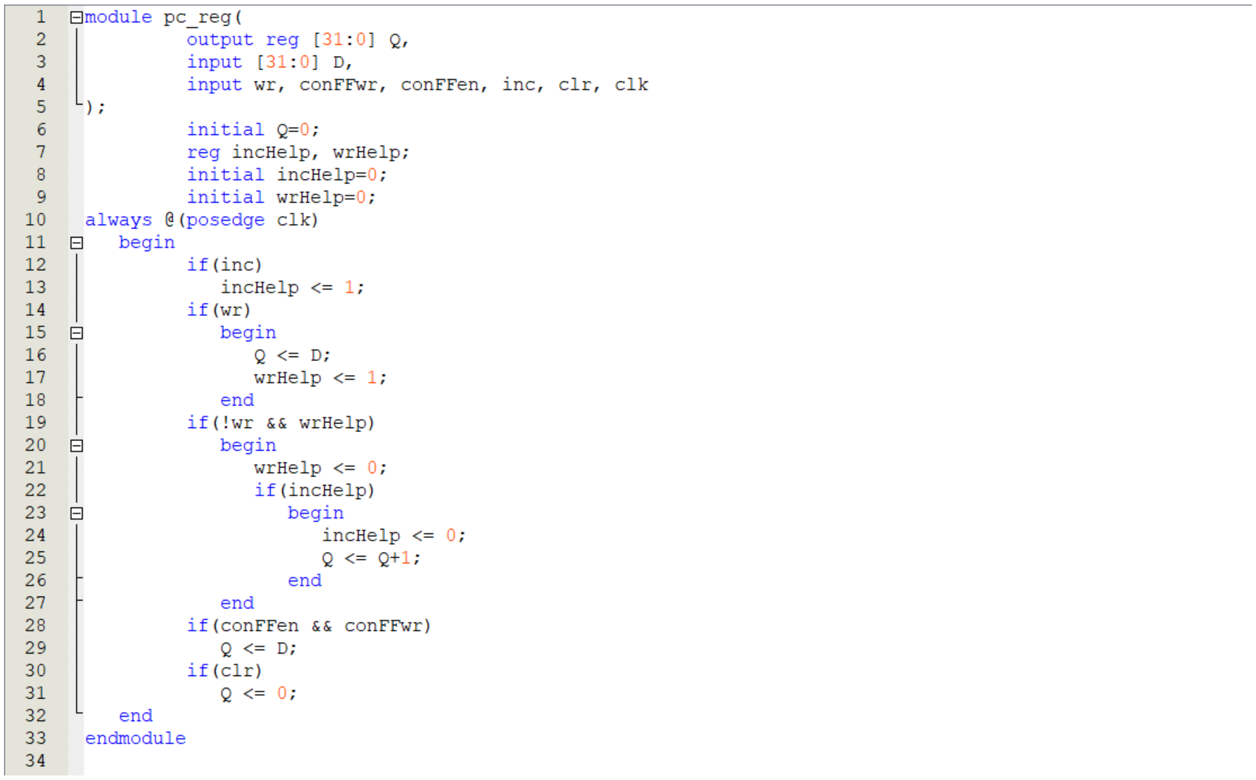
```
//NOT
   not1: begin
      MDRout <= 0; IRin <= 0;
      Grb<=1;Rout <=1;ALUselect <= 4'b1010;Zin <=1;
   end
   not2: begin
      Grb<=0; Rout <=0; Zin <=0;
      ZLowout <=1; Gra <=1; Rin <=1;
   end

//NOP
   nop:begin

   end

   nop2:begin

   end

// HALT
   halt:begin
      Run <= 0;
   end


   endcase
end
endmodule
```

main1.v



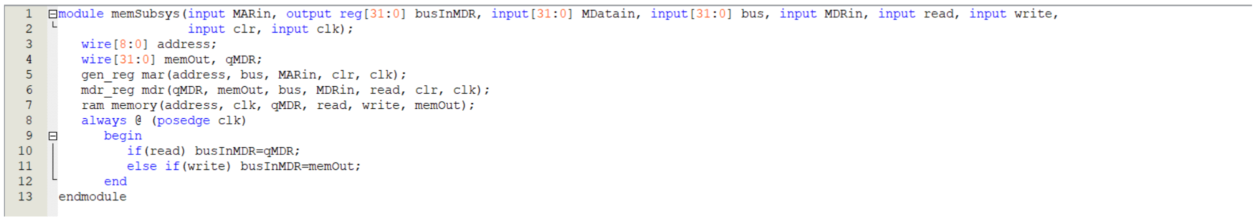
```verilog
module main1(
	input Rin,
	input Rout,
	input HIin, LOin, PCin, IRin, Yin, InPortout, Zin, conIn, outPortin, R15ctrl,
	input HIout, LOout, PCout, MDRout, MDRin, MARin, MDRread, memWrite, Cout, clk, IncPC, ZLowout, ZHighout, conOut, BAout, Gra, Grb, Grc,
	input [3:0] ALUselect,
	input [31:0] MDatain,
	output [31:0] IRdata
	);

	wire[63:0] ZReg;
	wire[31:0] bus, PCtemp;
	wire clr;
	wire IROut, R15in;
	wire [31:0] YData, XData;
	wire [31:0] ZLowData, ZHighData;

	wire [31:0] R0temp, busInR0, busInR1, busInR2, busInR3, busInR4, busInR5, busInR6, busInR7, busInR8, busInR9, busInR10, busInR11, busInR12, busInR13, busInR14, busInR15,
	            busInPC, busInMAR, busInMDR, busInHI, busInLO, busInZ, busInInPort, busInC;

	wire [15:0] genRegIn, genRegOut;

	ctrl_unit ctrl(Gra1, Grb1, Grc1, Rin1, Rout1, MDRin1, MDRout1, BAout1, Cout1, ZLowout1, PCin1, IRin1, HIout1, LOout1,
	                 InPortout1, outPortin1, LOin1, ZHighout1, HIin1, Yin1, Zin1, PCout1, IncPC1, MARin1, read1, wren1, clr1, conIn1, Run1,
	                 ALUselect1, IRdata1, clk1, reset1, stp1, conOut1);



	gen_reg r0(R0temp, bus, genRegIn[0], clr, clk);

	assign busInR0 = BAout ? 32'b0 : R0temp;

	 gen_reg r1(busInR1, bus, genRegIn[1], clr, clk);
	 gen_reg r2(busInR2, bus, genRegIn[2], clr, clk);
	 gen_reg r3(busInR3, bus, genRegIn[3], clr, clk);
	 gen_reg r4(busInR4, bus, genRegIn[4], clr, clk);
	 gen_reg r5(busInR5, bus, genRegIn[5], clr, clk);
	 gen_reg r6(busInR6, bus, genRegIn[6], clr, clk);
	 gen_reg r7(busInR7, bus, genRegIn[7], clr, clk);
	 gen_reg r8(busInR8, bus, genRegIn[8], clr, clk);
	 gen_reg r9(busInR9, bus, genRegIn[9], clr, clk);
	 gen_reg r10(busInR10, bus, genRegIn[10], clr, clk);
	 gen_reg r11(busInR11, bus, genRegIn[11], clr, clk);
	 gen_reg r12(busInR12, bus, genRegIn[12], clr, clk);
	 gen_reg r13(busInR13, bus, genRegIn[13], clr, clk);
	 gen_reg r14(busInR14, bus, genRegIn[14], clr, clk);
	 gen_reg r15(busInR15, bus, R15ctrl, clr, clk);

	 assign R15in = genRegIn[15] | R15ctrl;

	 sel_enc selectEncodeLogic(IRdata, Rin, Rout, BAout, Cout, Gra, Grb, Grc, genRegIn, genRegOut, busInC);

	 con_ff conFF(IRdata, bus, conIn, clk, conFFOut);
	 inoutport inOutPort(outPortin, clr, clk, inPortout, busInInPort, bus);



	 gen_reg ir(IRdata, bus, IRin, clr, clk);
	 pc_reg pc(busInPC, bus, PCin, conOut, conFFOut, IncPC, clr, clk);

	 memSubsys memSys(MARin, busInMDR, MDatain, bus, MDRin, MDRread, memWrite, clr, clk);

	 gen_reg hi(busInHI, bus, HIin, clr, clk);
	 gen_reg lo(busInLO, bus, LOin, clr, clk);
	 gen_reg y(YData, bus, Yin, clr, clk);
	 z_reg_64 z(busInZ, ZReg, Zin, ZLowout, ZHighout, clr, clk);

	 //ALU
	 alu alu(ALUselect, clk, YData, bus, ZReg, carry);

	// Bus
	bus bus_inst(busInR0, busInR1, busInR2, busInR3, busInR4, busInR5, busInR6, busInR7, busInR8, busInR9, busInR10, busInR11,
	               busInR12, busInR13, busInR14, busInR15, busInHI, busInLO, busInZ, busInPC, busInMDR, busInInPort, busInC,
	               genRegOut[0], genRegOut[1], genRegOut[2], genRegOut[3], genRegOut[4], genRegOut[5], genRegOut[6], genRegOut[7], genRegOut[8],
	               genRegOut[9], genRegOut[10], genRegOut[11], genRegOut[12], genRegOut[13], genRegOut[14], genRegOut[15], HIout,
	               LOout, ZHighout, ZLowout, PCout, MDRout, InPortout, Cout, bus, clk);
endmodule
```

pc_reg.v



```verilog
module pc_reg(
         output reg [31:0] Q,
         input [31:0] D,
         input wr, conFFwr, conFFen, inc, clr, clk
);
         initial Q=0;
         reg incHelp, wrHelp;
         initial incHelp=0;
         initial wrHelp=0;
always @(posedge clk)
   begin
         if(inc)
            incHelp <= 1;
         if(wr)
            begin
               Q <= D;
               wrHelp <= 1;
            end
         if(!wr && wrHelp)
            begin
               wrHelp <= 0;
               if(incHelp)
                  begin
                     incHelp <= 0;
                     Q <= Q+1;
                  end
            end
         if(conFFen && conFFwr)
            Q <= D;
         if(clr)
            Q <= 0;
   end
endmodule
```

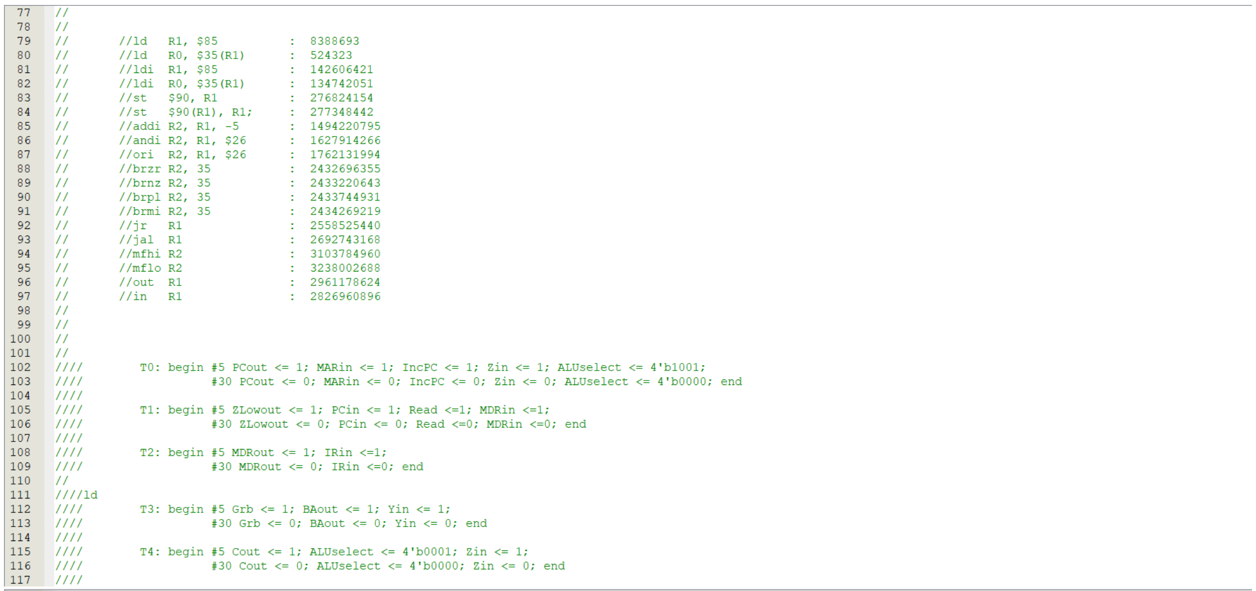
memSubsys.v



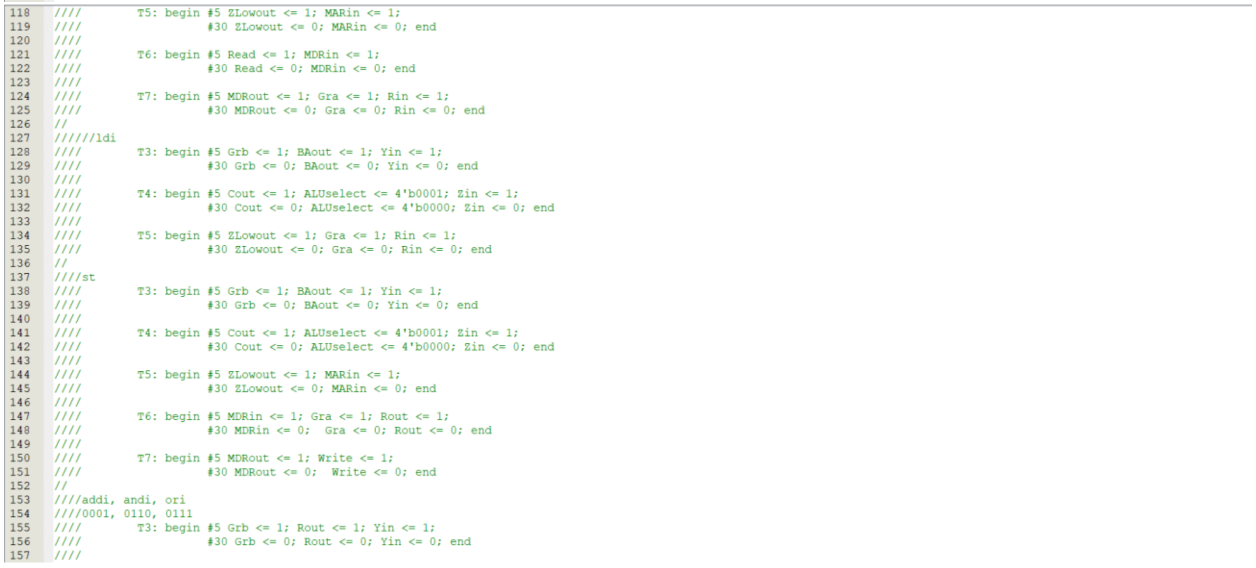
```verilog
module memSubsys(input MARin, output reg[31:0] busInMDR, input[31:0] MDatain, input[31:0] bus, input MDRin, input read, input write,
                 input clr, input clk);
   wire[8:0] address;
   wire[31:0] memOut, qMDR;
   gen_reg mar(address, bus, MARin, clr, clk);
   mdr_reg mdr(qMDR, memOut, bus, MDRin, read, clr, clk);
   ram memory(address, clk, qMDR, read, write, memOut);
   always @ (posedge clk)
      begin
         if(read) busInMDR=qMDR;
         else if(write) busInMDR=memOut;
      end
endmodule
```

datapath_tb.v

```verilog
`timescale 1ns/10ps
module datapath_tb;

   wire Rin;

   wire Rout;

   wire HIin, LOin, PCin, IRin, Yin, InPortout, Zin, conIn, outPortin, R15ctrl;

   wire HIout, LOout, PCout, MDRout, MDRin, MARin, MDRread, Cout, IncPC, ZLowout, ZHighout;

   reg clk;

   wire conOut, BAout, Gra, Grb, Grc;

   wire[3:0] ALUselect;
   reg[31:0] MDatain;
   wire Read, Write;

   wire [31:0]IRdata;

   parameter   Default = 4'b0000, Reg_load1a = 4'b0001, Reg_load1b = 4'b0010, Reg_load2a = 4'b0011 ,Reg_load2b = 4'b0100,
               Reg_load3a = 4'b0101, Reg_load3b = 4'b0110, T0 = 4'b0111, T1 = 4'b1000, T2 = 4'b1001, T3 = 4'b1010, T4 = 4'b1011,
               T5 = 4'b1100, T6 = 4'b1101, T7 = 4'b1110;

   reg[3:0] Present_state = Default;

   main1 DUT (.Rin(Rin), .Rout(Rout), .HIin(HIin), .LOin(LOin), .PCin(PCin), .IRin(IRin), .Yin(Yin), .InPortout(InPortout),
              .Zin(Zin), .conIn(conIn), .outPortin(outPortin), .R15ctrl(R15ctrl), .HIout(HIout), .LOout(LOout), .PCout(PCout), .MDRout(MDRout), .MDRin(MDRin),
              .MARin(MARin), .MDRread(Read), .memWrite(Write), .Cout(Cout), .clk(clk), .IncPC(IncPC), .ZLowout(ZLowout), .ZHighout(ZHighout),
              .conOut(conOut), .BAout(BAout), .Gra(Gra), .Grb(Grb), .Grc(Grc), .ALUselect(ALUselect), .MDatain(MDatain), .IRdata(IRdata));



   //control signals seem to work, but IR in ctrl_unit isn't recieving data in current form
   ctrl_unit ctrl(Gra, Grb, Grc, Rin, Rout, MDRin, MDRout, BAout, Cout, ZLowout, PCin, IRin, HIout, LOout,
                        InPortout, outPortin, LOin, ZHighout, HIin, Yin, Zin, PCout, IncPC, MARin, Read, Write, clr, conIn, Run,
                        ALUselect, IRdata, clk, reset, stp, conOut, R15ctrl);


initial
            begin
                  clk = 0;
                  forever #5 clk = ~clk;
            end


always @(posedge clk)
         begin
               case(Present_state)
                        Default     :  #40   Present_state = T0;
                        T0          :  #40   Present_state = T1;
                        T1          :  #40   Present_state = T2;
                        T2          :  #40   Present_state = T3;
                        T3          :  #40   Present_state = T4;
                        T4          :  #40   Present_state = T5;
                        T5          :  #40   Present_state = T6;
                        T6          :  #40   Present_state = T7;
               endcase
         end

//always @(Present_state)
//       begin
//             case(Present_state)
//                   Default:begin
//                               Rout <=0; Rin <=0;
//                               HIout <=0; LOout <=0; InPortout <=0; Cout <=0;
//                               LOin <=0; HIin <=0;
//                               PCout <=0; ZLowout <=0; ZHighout <=0; MDRout <=0;
//                               MARin <=0; Zin <=0;
//                               PCin <=0; MDRin <=0; IRin <=0; Yin <=0;
//                               IncPC <=0; Read <=0; Write <=0; ALUselect <=0;
//                               Gra <=0; Grb <=0; Grc <=0; BAout <=0; conOut <=0;
//                               MDatain<=32'h00000000;
//       end
//
```



```
//
//
//       //ld   R1, $85          :  8388693
//       //ld   R0, $35(R1)      :  524323
//       //ldi  R1, $85          :  142606421
//       //ldi  R0, $35(R1)      :  134742051
//       //st   $90, R1          :  276824154
//       //st   $90(R1), R1;     :  277348442
//       //addi R2, R1, -5       :  1494220795
//       //andi R2, R1, $26      :  1627914266
//       //ori  R2, R1, $26      :  1762131994
//       //brzr R2, 35           :  2432696355
//       //brnz R2, 35           :  2433220643
//       //brpl R2, 35           :  2433744931
//       //brmi R2, 35           :  2434269219
//       //jr   R1               :  2558525440
//       //jal  R1               :  2692743168
//       //mfhi R2               :  3103784960
//       //mflo R2               :  3238002688
//       //out  R1               :  2961178624
//       //in   R1               :  2826960896
//
//
//
//
////        T0: begin #5 PCout <= 1; MARin <= 1; IncPC <= 1; Zin <= 1; ALUselect <= 4'b1001;
////                  #30 PCout <= 0; MARin <= 0; IncPC <= 0; Zin <= 0; ALUselect <= 4'b0000; end
////
////        T1: begin #5 ZLowout <= 1; PCin <= 1; Read <=1; MDRin <=1;
////                  #30 ZLowout <= 0; PCin <= 0; Read <=0; MDRin <=0; end
////
////        T2: begin #5 MDRout <= 1; IRin <=1;
////                  #30 MDRout <= 0; IRin <=0; end
//
////ld
////        T3: begin #5 Grb <= 1; BAout <= 1; Yin <= 1;
////                  #30 Grb <= 0; BAout <= 0; Yin <= 0; end
////
////        T4: begin #5 Cout <= 1; ALUselect <= 4'b0001; Zin <= 1;
////                  #30 Cout <= 0; ALUselect <= 4'b0000; Zin <= 0; end
////
```

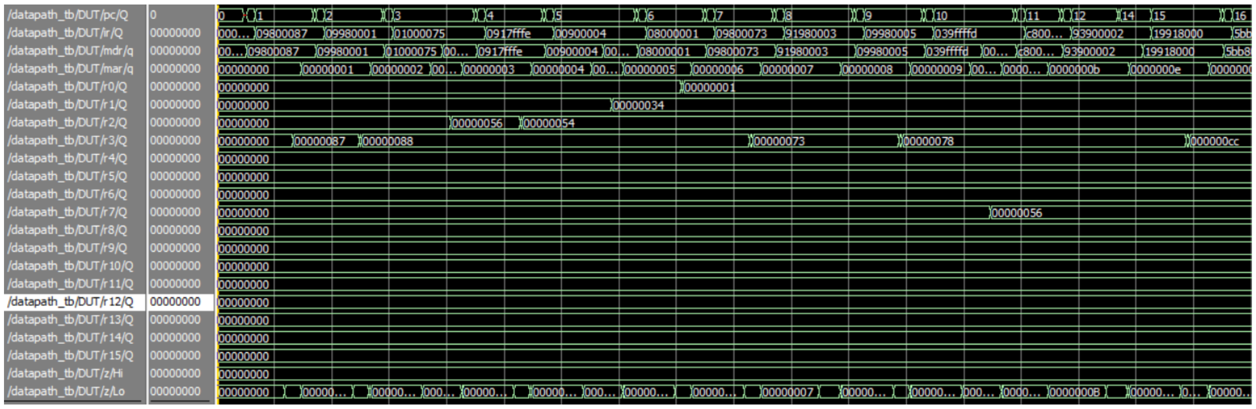


```
////        T5: begin #5 ZLowout <= 1; MARin <= 1;
////                  #30 ZLowout <= 0; MARin <= 0; end
////
////        T6: begin #5 Read <= 1; MDRin <= 1;
////                  #30 Read <= 0; MDRin <= 0; end
////
////        T7: begin #5 MDRout <= 1; Gra <= 1; Rin <= 1;
////                  #30 MDRout <= 0; Gra <= 0; Rin <= 0; end
//
//////ldi
////        T3: begin #5 Grb <= 1; BAout <= 1; Yin <= 1;
////                  #30 Grb <= 0; BAout <= 0; Yin <= 0; end
////
////        T4: begin #5 Cout <= 1; ALUselect <= 4'b0001; Zin <= 1;
////                  #30 Cout <= 0; ALUselect <= 4'b0000; Zin <= 0; end
////
////        T5: begin #5 ZLowout <= 1; Gra <= 1; Rin <= 1;
////                  #30 ZLowout <= 0; Gra <= 0; Rin <= 0; end
//
////st
////        T3: begin #5 Grb <= 1; BAout <= 1; Yin <= 1;
////                  #30 Grb <= 0; BAout <= 0; Yin <= 0; end
////
////        T4: begin #5 Cout <= 1; ALUselect <= 4'b0001; Zin <= 1;
////                  #30 Cout <= 0; ALUselect <= 4'b0000; Zin <= 0; end
////
////        T5: begin #5 ZLowout <= 1; MARin <= 1;
////                  #30 ZLowout <= 0; MARin <= 0; end
////
////        T6: begin #5 MDRin <= 1; Gra <= 1; Rout <= 1;
////                  #30 MDRin <= 0;  Gra <= 0; Rout <= 0; end
////
////        T7: begin #5 MDRout <= 1; Write <= 1;
////                  #30 MDRout <= 0;  Write <= 0; end
//
////addi, andi, ori
////0001, 0110, 0111
////        T3: begin #5 Grb <= 1; Rout <= 1; Yin <= 1;
////                  #30 Grb <= 0; Rout <= 0; Yin <= 0; end
////
```

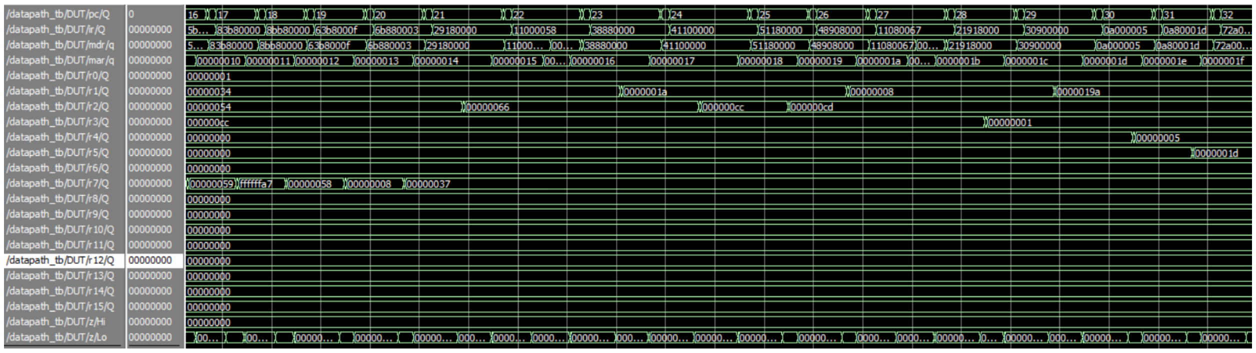
```
////        T4: begin #5 Cout <= 1; ALUselect <= 4'b0111; Zin <= 1;
////                  #30 Cout <= 0; ALUselect <= 4'b0000; Zin <= 0; end
////
////        T5: begin #5 ZLowout <= 1; Gra <= 1; Rin <= 1;
////                  #30 ZLowout <= 0; Gra <= 0; Rin <= 0; end
//
////brzr, brnz, brpl, brmi
//       T3: begin #5 Gra <= 1; Rout <= 1; conIn <= 1;
//                 #30 Gra <= 0; Rout <= 0; conIn <= 0; end
//
//       T4: begin #5 PCout <= 1; Yin <= 1;
//                 #30 PCout <= 0; Yin <= 0; end
//
//       T5: begin #5 Cout <= 1; ALUselect <= 4'b0001; Zin <= 1;
//                 #30 Cout <= 0; ALUselect <= 4'b0000; Zin <= 0; end
//
//       T6: begin #5 ZLowout <= 1; conOut <= 1;
//                 #30 ZLowout <= 0; conOut <= 0; end
//
////jr
////        T3: begin #5 Gra <= 1; Rout <= 1; PCin <= 1;
////                  #30 Gra <= 0; Rout <= 0; PCin <= 0; end
//
////jal
////        T3: begin #5 R15ctrl <= 1; PCout <= 1;
////                  #30 R15ctrl <= 0; PCout <= 0; end
////
////        T4: begin #5 Gra <= 1; Rout <= 1; PCin <= 1;
////                  #30 Gra <= 0; Rout <= 0; PCin <= 0; end
//
////mfhi
////        T3: begin #5 HIout <= 1; Gra <= 1; Rin <= 1;
////                  #30 HIout <= 0; Gra <= 0; Rin <= 0; end
//
////mflo
////        T3: begin #5 LOout <= 1; Gra <= 1; Rin <= 1;
////                  #30 LOout <= 0; Gra <= 0; Rin <= 0; end
//
////out
////        T3: begin #5 Gra <= 1; Rout <= 1; outPortin <= 1;
////                  #30 Gra <= 0; Rout <= 0; outPortin <= 0; end
//
////in
//       T3: begin #5 Gra <= 1; Rin <= 1; InPortout <= 1;
//                #30 Gra <= 0; Rin <= 0; InPortout <= 0; end
//
//
//       endcase
//    end
endmodule
```

Functional Simulations

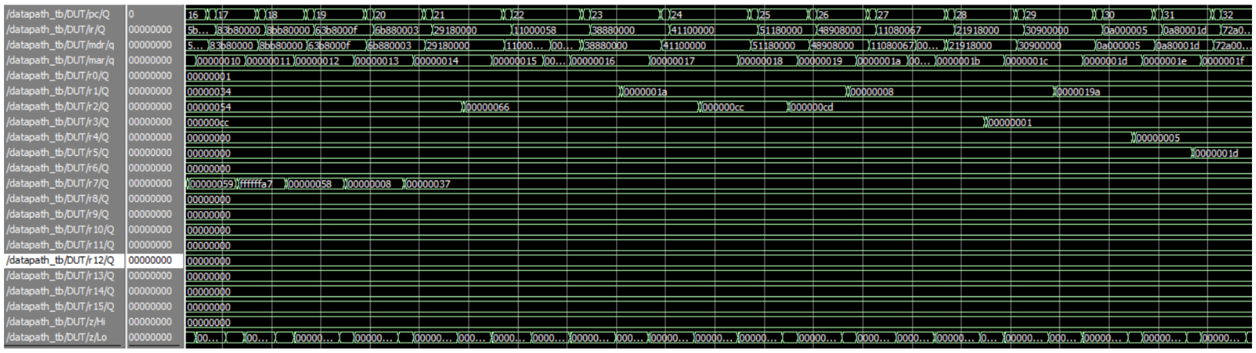
Instructions 0-15



Instructions 16-31

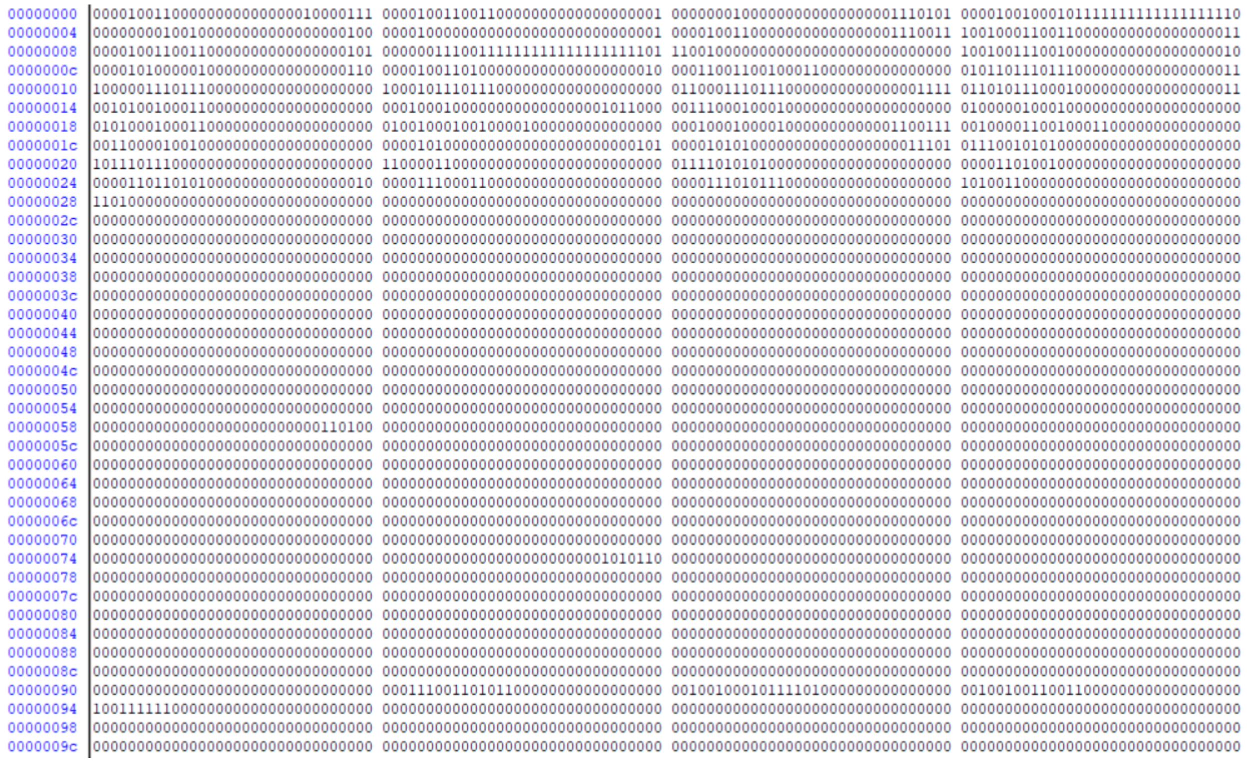


Instructions 32-end



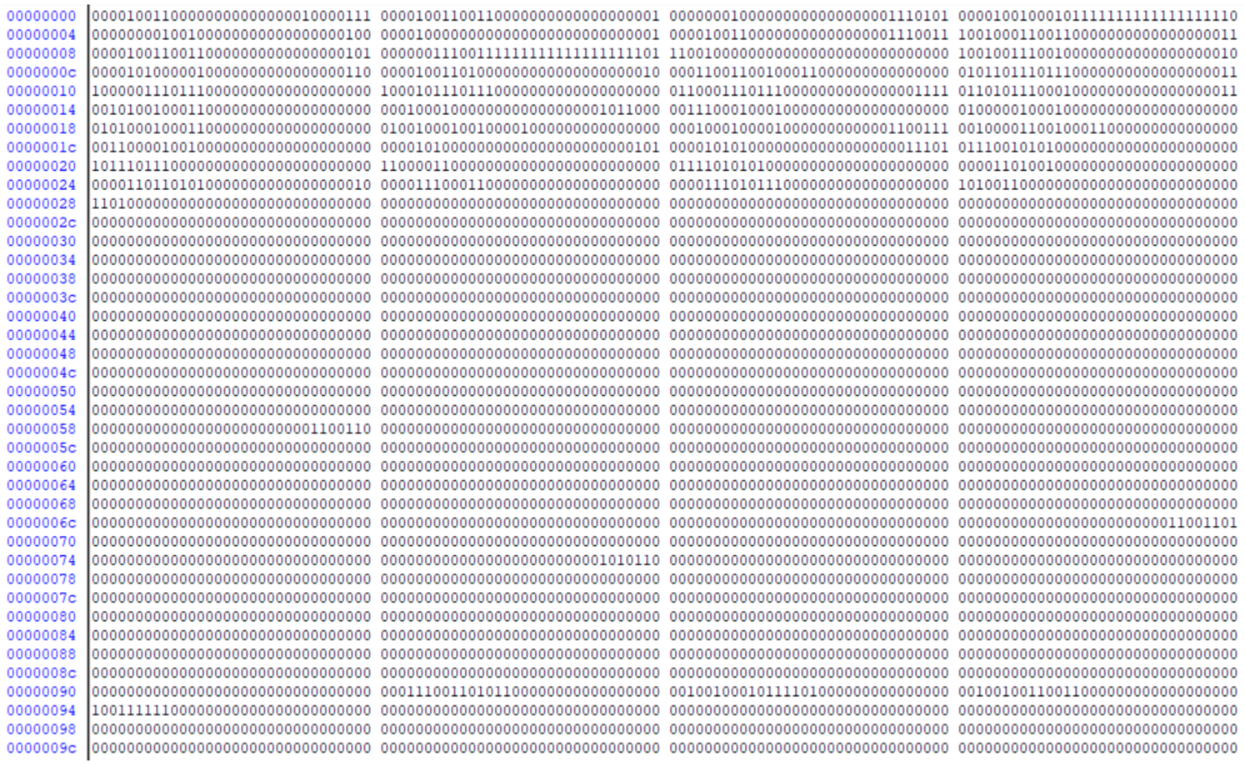
State of memory

Memory before run



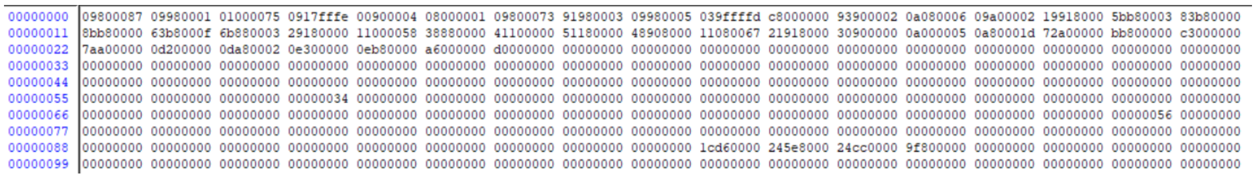
```
00000000 |00001001100000000000000010000111 00001001100110000000000000000001 00000001000000000000000001110101 00001001000101111111111111111110
00000004 |00000000100100000000000000000100 00001000000000000000000000000001 00001001100000000000000001110011 10010001100110000000000000000011
00000008 |00001001100110000000000000000101 00000011100111111111111111111101 11001000000000000000000000000000 10010011100100000000000000000010
0000000c |00001010000010000000000000000110 00001001101000000000000000000010 00011001100100011000000000000000 01011011101110000000000000000011
00000010 |10000011101110000000000000000000 10001011101110000000000000000000 01100011101110000000000000001111 01101011100010000000000000000011
00000014 |00101001000110000000000000000000 00010001000000000000000001011000 00111000100010000000000000000000 01000001000100000000000000000000
00000018 |01010001000110000000000000000000 01001000100100001000000000000000 00010001000010000000000001100111 00100001100100011000000000000000
0000001c |00110000100100000000000000000000 00001010000000000000000000000101 00001010100000000000000000011101 01110010101000000000000000000000
00000020 |10111011100000000000000000000000 11000011000000000000000000000000 01111010101000000000000000000000 00001101001000000000000000000000
00000024 |00001101101010000000000000000010 00001110001100000000000000000000 00001110101110000000000000000000 10100110000000000000000000000000
00000028 |11010000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000002c |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000030 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000034 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000038 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000003c |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000040 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000044 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000048 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000004c |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000050 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000054 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000058 |00000000000000000000000000110100 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000005c |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000060 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000064 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000068 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000006c |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000070 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000074 |00000000000000000000000000000000 00000000000000000000000001010110 00000000000000000000000000000000 00000000000000000000000000000000
00000078 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000007c |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000080 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000084 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000088 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000008c |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000090 |00000000000000000000000000000000 00011100110101100000000000000000 00100100010111101000000000000000 00100100110011000000000000000000
00000094 |10011111100000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000098 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000009c |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
```

Memory after run



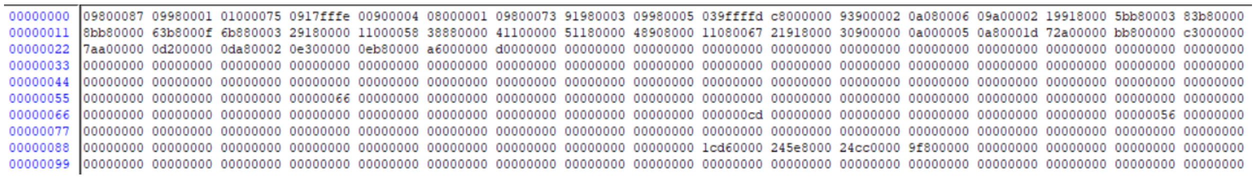
```
00000000 |00001001100000000000000010000111 00001001100110000000000000000001 00000001000000000000000001110101 00001001000101111111111111111110
00000004 |00000000100100000000000000000100 00001000000000000000000000000001 00001001100000000000000001110011 10010001100110000000000000000011
00000008 |00001001100110000000000000000101 00000011100111111111111111111101 11001000000000000000000000000000 10010011100100000000000000000010
0000000c |00001010000010000000000000000110 00001001101000000000000000000010 00011001100100011000000000000000 01011011101110000000000000000011
00000010 |10000011101110000000000000000000 10001011101110000000000000000000 01100011101110000000000000001111 01101011100010000000000000000011
00000014 |00101001000110000000000000000000 00010001000000000000000001011000 00111000100010000000000000000000 01000001000100000000000000000000
00000018 |01010001000110000000000000000000 01001000100100001000000000000000 00010001000010000000000001100111 00100001100100011000000000000000
0000001c |00110000100100000000000000000000 00001010000000000000000000000101 00001010100000000000000000011101 01110010101000000000000000000000
00000020 |10111011100000000000000000000000 11000011000000000000000000000000 01111010101000000000000000000000 00001101001000000000000000000000
00000024 |00001101101010000000000000000010 00001110001100000000000000000000 00001110101110000000000000000000 10100110000000000000000000000000
00000028 |11010000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000002c |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000030 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000034 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000038 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000003c |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000040 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000044 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000048 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000004c |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000050 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000054 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000058 |00000000000000000000000001100110 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000005c |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000060 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000064 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000068 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000006c |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000011001101
00000070 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000074 |00000000000000000000000000000000 00000000000000000000000001010110 00000000000000000000000000000000 00000000000000000000000000000000
00000078 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000007c |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000080 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000084 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000088 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000008c |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000090 |00000000000000000000000000000000 00011100110101100000000000000000 00100100010111101000000000000000 00100100110011000000000000000000
00000094 |10011111100000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
00000098 |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
0000009c |00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
```

Memory before run (Hex)



```
00000000 |09800087 09980001 01000075 0917fffe 00900004 08000001 09800073 91980003 09980005 039ffffd c8000000 93900002 0a080006 09a00002 19918000 5bb80003 83b80000
00000011 |8bb80000 63b8000f 6b880003 29180000 11000058 38880000 41100000 51180000 48908000 11080067 21918000 30900000 0a000005 0a80001d 72a00000 bb800000 c3000000
00000022 |7aa00000 0d200000 0da80002 0e300000 0eb80000 a6000000 d0000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000033 |00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000044 |00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000055 |00000000 00000000 00000000 00000034 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000066 |00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000056 00000000
00000077 |00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000088 |00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 1cd60000 245e8000 24cc0000 9f800000 00000000 00000000 00000000 00000000
00000099 |00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Memory after run (Hex)



```
00000000 |09800087 09980001 01000075 0917fffe 00900004 08000001 09800073 91980003 09980005 039ffffd c8000000 93900002 0a080006 09a00002 19918000 5bb80003 83b80000
00000011 |8bb80000 63b8000f 6b880003 29180000 11000058 38880000 41100000 51180000 48908000 11080067 21918000 30900000 0a000005 0a80001d 72a00000 bb800000 c3000000
00000022 |7aa00000 0d200000 0da80002 0e300000 0eb80000 a6000000 d0000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000033 |00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000044 |00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000055 |00000000 00000000 00000000 00000066 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000066 |00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000cd 00000000 00000000 00000000 00000000 00000000 00000056 00000000
00000077 |00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000088 |00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 1cd60000 245e8000 24cc0000 9f800000 00000000 00000000 00000000 00000000
00000099 |00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```