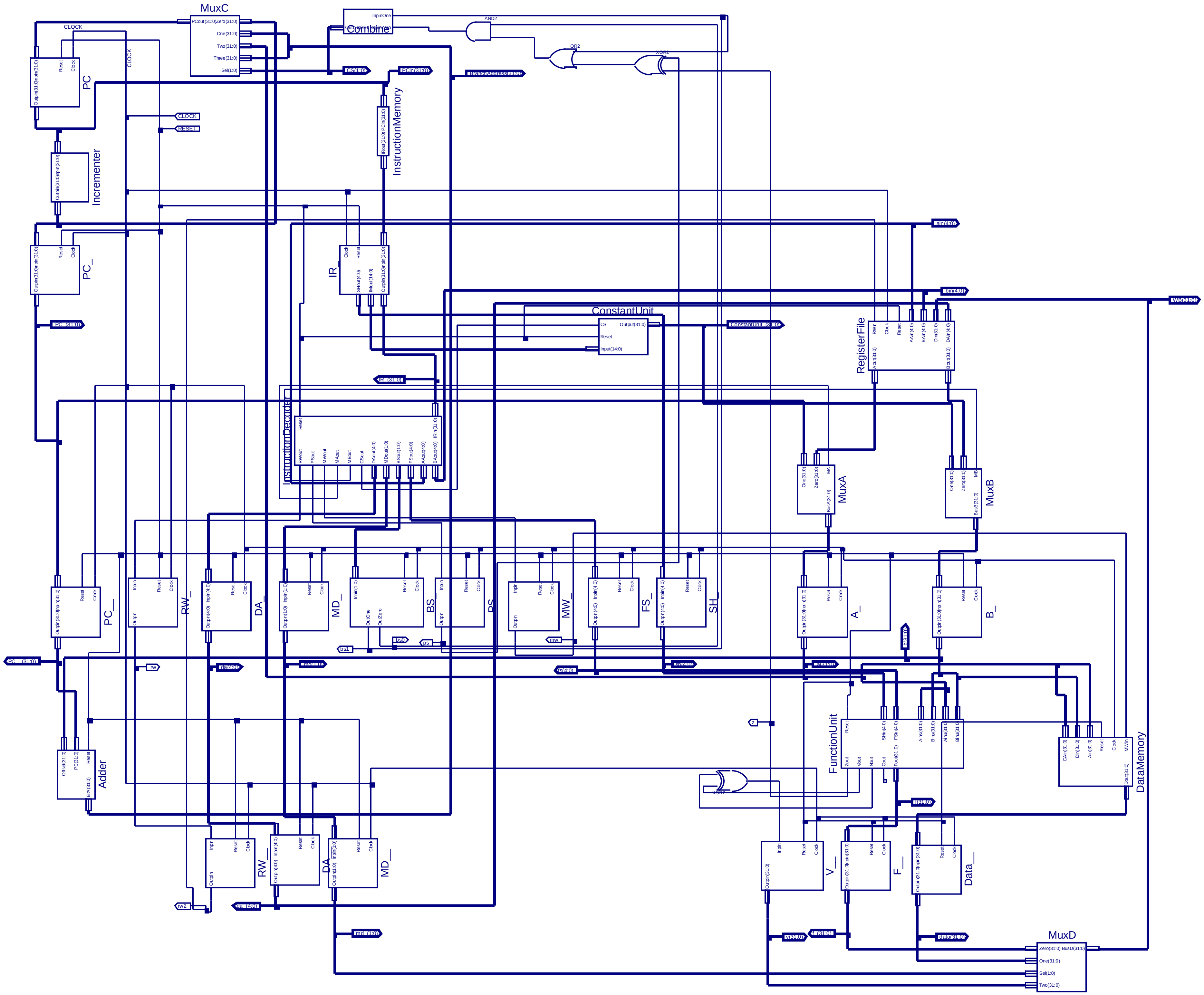

MuxC
PCout(31:0)
Zero(31:0)
One(31:0)
Two(31:0)
Three(31:0)
Sel(1:0)
Combine
InpinOne
AND2
OR2
XOR2
CS(1:0)
PCin(31:0)
BranchAddress(31:0)
CLOCK
RESET
PC
Incrementer
PC_
PC_(31:0)
InstructionMemory
IR_
IR_(31:0)
InstructionDecoder
ConstantUnit
ConstantUnit_(31:0)
RegisterFile
ain(4:0)
bin(4:0)
WB(31:0)
MuxA
MuxB
PC__
RW_
DA_
MD_
BS_
PS_
MW_
FS_
SH_
A_
B_
rw
da(4:0)
md(1:0)
bs
bs0
ps
mw
fs(4:0)
sh(4:0)
a(31:0)
b(31:0)
PC__(31:0)
Adder
FunctionUnit
z
XOR2
f(31:0)
DataMemory
RW__
DA__
MD__
rw2
da_(4:0)
md_(1:0)
V__
F__
Data__
v(31:0)
f_(31:0)
data(31:0)
MuxD
Zero(31:0)
One(31:0)
Sel(1:0)
Two(31:0)
BusD(31:0)

```verilog
// Verilog test fixture created from schematic C:\Users\Kyle\Xilinx\RISC\TOP.sch - Sun No
29 16:42:36 2009

`timescale 1ns / 1ps

module TOP_TOP_sch_tb();

// Inputs
   reg CLOCK;
   reg RESET;

// Output
   wire [31:0] PCin;
   wire [31:0] PC_;
   wire [31:0] PC__;
   wire [31:0] BranchAddress;
   wire [31:0] ConstantUnit_;
   wire [4:0] ain;
   wire [4:0] bin;
   wire [31:0] a;
    wire [31:0] b;
    wire [31:0] f;
    wire [31:0] f_;
     wire [31:0] WB;
   wire [4:0] sh;
   wire [4:0] fs;

   wire rw;
   wire [4:0] da;
   wire [1:0] md;
   wire rw2;
   wire [4:0] da_;

   wire [31:0] data;
   wire [1:0] md_;

   wire mw;
   wire ps;

   wire [31:0] IR_;
   wire [1:0] CS;
   wire bs1;
   wire bs0;

   wire [31:0] v;

   wire z;



// Bidirs

// Instantiate the UUT
   TOP UUT (
      .PCin(PCin),
      .PC_(PC_),
      .ConstantUnit_(ConstantUnit_),
      .a(a),
      .sh(sh),
      .fs(fs),
      .f(f),
			.rw(rw),
			.da(da),
			.md(md),
			.rw2(rw2),
			.da_(da_),
			.f_(f_),
			.data(data),
			.md_(md_),
			.WB(WB),
			.mw(mw),
			.ps(ps),
			.BranchAddress(BranchAddress),
			.IR_(IR_),
			.CS(CS),
			.bs1(bs1),
			.bs0(bs0),
			.CLOCK(CLOCK),
			.RESET(RESET),
			.b(b),
			.v(v),
			.PC__(PC__),
			.z(z),
			.ain(ain),
			.bin(bin)

		);
	initial begin
			// Initialize Inputs
			CLOCK = 0;
			RESET = 0;
			// Add stimulus here

		end

		always
			begin
				CLOCK = ~CLOCK;
				#50;
			end

		endmodule

```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: 
// Engineer: 
// 
// Create Date:    16:10:54 11/15/2009 
// Design Name: 
// Module Name:    AddressInstructionMemoryInstruction 
// Project Name: 
// Target Devices: 
// Tool versions: 
// Description: 
//
// Dependencies: 
//
// Revision: 
// Revision 0.01 - File Created
// Additional Comments: 
//
//////////////////////////////////////////////////////////////////////////////////
module InstructionMemory(
//Read ports
    input [31:0] PCin,
    output [31:0] IRout

    );

    `include "Instructions.txt"
    `include "Registers.txt"

    reg [31:0] IR;

    assign IRout = IR;


    reg [31:0] InstructionMemory  [31:0];   // 64 Registers of 32 bits

    reg [31:0] MemLoc;


    initial
      begin

      ///////////////////////////////////////////////
      //MOV R5, R0
      //Opcode
      MemLoc = 0;
      InstructionMemory[MemLoc][31:25] = MOV;
      //DR
      InstructionMemory[MemLoc][24:20] = R5;
      //SA
      InstructionMemory[MemLoc][19:15] = R0;
      //IM
      InstructionMemory[MemLoc][14:0] = 0;
      MemLoc = MemLoc + 1;
      ///////////////////////////////////////////////
      //NOP
      //Opcode
      InstructionMemory[MemLoc][31:25] = NOP;
      MemLoc = MemLoc + 1;
      ///////////////////////////////////////////////
```

```
                //AIU R5, 0||100000000000000
                //Opcode
                InstructionMemory[MemLoc][31:25] = AIU;
                //DR
                InstructionMemory[MemLoc][24:20] = R5;
                //SA
                InstructionMemory[MemLoc][19:15] = R5;
                //IM
                InstructionMemory[MemLoc][14:0] = 15'b100000000000000;
                MemLoc = MemLoc + 1;
                //////////////////////////////////////////////
                //NOP
                //Opcode
                InstructionMemory[MemLoc][31:25] = NOP;
                MemLoc = MemLoc + 1;
                //////////////////////////////////////////////
                //LSL R5, 1
                //Opcode
                InstructionMemory[MemLoc][31:25] = LSL;
                //DR
                InstructionMemory[MemLoc][24:20] = R5;
                //SA
                InstructionMemory[MemLoc][19:15] = R5;
                //IM
                InstructionMemory[MemLoc][14:0] = 1;
                MemLoc = MemLoc + 1;
                //////////////////////////////////////////////
                //MOV R10, R0
                //Opcode
                InstructionMemory[MemLoc][31:25] = MOV;
                //DR
                InstructionMemory[MemLoc][24:20] = R10;
                //SA
                InstructionMemory[MemLoc][19:15] = R0;
                //IM
                InstructionMemory[MemLoc][14:0] = 0;
                MemLoc = MemLoc + 1;
                //////////////////////////////////////////////
                //NOP
                //Opcode
                InstructionMemory[MemLoc][31:25] = NOP;
                MemLoc = MemLoc + 1;
                //////////////////////////////////////////////
                //AIU R10, R10, 0||000000000100000
                //Opcode
                InstructionMemory[MemLoc][31:25] = AIU;
                //DR
                InstructionMemory[MemLoc][24:20] = R10;
                //SA
                InstructionMemory[MemLoc][19:15] = R10;
                //IM
                InstructionMemory[MemLoc][14:0] = 15'b000000000100000;
                MemLoc = MemLoc + 1;
                //////////////////////////////////////////////
                //MOV R3, R0
                //Opcode
                InstructionMemory[MemLoc][31:25] = MOV;
                //DR
                InstructionMemory[MemLoc][24:20] = R3;
                //SA
                InstructionMemory[MemLoc][19:15] = R0;
                //IM
                InstructionMemory[MemLoc][14:0] = 0;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //ANI R8, R2, 0||000000000000001
                //Opcode
                InstructionMemory[MemLoc][31:25] = ANI;
                //DR
                InstructionMemory[MemLoc][24:20] = R8;
                //SA
                InstructionMemory[MemLoc][19:15] = R2;
                //IM
                InstructionMemory[MemLoc][14:0] = 15'b000000000000001;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //NOP
                //Opcode
                InstructionMemory[MemLoc][31:25] = NOP;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //BZ R8, 6
                //Opcode
                InstructionMemory[MemLoc][31:25] = BZ;
                //DR
                InstructionMemory[MemLoc][24:20] = R0;
                //SA
                InstructionMemory[MemLoc][19:15] = R8;
                //Offset
                InstructionMemory[MemLoc][14:0] = 6;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //NOP
                //Opcode
                InstructionMemory[MemLoc][31:25] = NOP;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //NOP
                //Opcode
                InstructionMemory[MemLoc][31:25] = NOP;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //AND R7, R3, R5
                //Opcode
                InstructionMemory[MemLoc][31:25] = AND;
                //DR
                InstructionMemory[MemLoc][24:20] = R7;
                //SA
                InstructionMemory[MemLoc][19:15] = R3;
                //IM
                InstructionMemory[MemLoc][14:10] = R5;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //ADD R3, R3, R1
                //Opcode
                InstructionMemory[MemLoc][31:25] = ADD;
                //DR
                InstructionMemory[MemLoc][24:20] = R3;
                //SA
                InstructionMemory[MemLoc][19:15] = R3;
                //IM
                InstructionMemory[MemLoc][14:10] = R1;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //NOP
                //Opcode
                InstructionMemory[MemLoc][31:25] = NOP;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //ANI R8, R3, 0||000000000000001
                //Opcode
                InstructionMemory[MemLoc][31:25] = ANI;
                //DR
                InstructionMemory[MemLoc][24:20] = R8;
                //SA
                InstructionMemory[MemLoc][19:15] = R3;
                //IM
                InstructionMemory[MemLoc][14:0] = 15'b000000000000001;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //NOP
                //Opcode
                InstructionMemory[MemLoc][31:25] = NOP;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //LSR R3, R3, 0||000000000000001
                //Opcode
                InstructionMemory[MemLoc][31:25] = LSR;
                //DR
                InstructionMemory[MemLoc][24:20] = R3;
                //SA
                InstructionMemory[MemLoc][19:15] = R3;
                //IM
                InstructionMemory[MemLoc][14:0] = 15'b000000000000001;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //NOP
                //Opcode
                InstructionMemory[MemLoc][31:25] = NOP;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //ADD R3, R3, R7
                //Opcode
                InstructionMemory[MemLoc][31:25] = ADD;
                //DR
                InstructionMemory[MemLoc][24:20] = R3;
                //SA
                InstructionMemory[MemLoc][19:15] = R3;
                //IM
                InstructionMemory[MemLoc][14:10] = R7;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //NOP
                //Opcode
                InstructionMemory[MemLoc][31:25] = NOP;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //LSR R2, R2, 0||000000000000001
                //Opcode
                InstructionMemory[MemLoc][31:25] = LSR;
                //DR
                InstructionMemory[MemLoc][24:20] = R2;
                //SA
```

```verilog
                InstructionMemory[MemLoc][19:15] = R2;
                //IM
                InstructionMemory[MemLoc][14:0] = 15'b000000000000001;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //NOP
                //Opcode
                InstructionMemory[MemLoc][31:25] = NOP;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //BZ R2, 6
                //Opcode
                InstructionMemory[MemLoc][31:25] = BZ;
                //DR
                InstructionMemory[MemLoc][24:20] = R0;
                //SA
                InstructionMemory[MemLoc][19:15] = R2;
                //Offset
                InstructionMemory[MemLoc][14:0] = 6;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //NOP
                //Opcode
                InstructionMemory[MemLoc][31:25] = NOP;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //NOP
                //Opcode
                InstructionMemory[MemLoc][31:25] = NOP;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //ADD R2, R2, R5
                //Opcode
                InstructionMemory[MemLoc][31:25] = ADD;
                //DR
                InstructionMemory[MemLoc][24:20] = R2;
                //SA
                InstructionMemory[MemLoc][19:15] = R2;
                //IM
                InstructionMemory[MemLoc][14:10] = R5;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //SIU R2, R2, R5
                //Opcode
                InstructionMemory[MemLoc][31:25] = SIU;
                //DR
                InstructionMemory[MemLoc][24:20] = R10;
                //SA
                InstructionMemory[MemLoc][19:15] = R10;
                //IM
                InstructionMemory[MemLoc][14:0] = 15'b000000000000001;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //NOP
                //Opcode
                InstructionMemory[MemLoc][31:25] = NOP;
                MemLoc = MemLoc + 1;
                ///////////////////////////////////////////////
                //BNZ R2, 6
                //Opcode
                InstructionMemory[MemLoc][31:25] = BZ;
```

```
            //DR
            InstructionMemory[MemLoc][24:20] = R0;
            //SA
            InstructionMemory[MemLoc][19:15] = R10;
            //Offset
            InstructionMemory[MemLoc][14:0] = 6;
            MemLoc = MemLoc + 1;
            ///////////////////////////////////////////////
            //NOP
            //Opcode
            InstructionMemory[MemLoc][31:25] = NOP;
            MemLoc = MemLoc + 1;
            ///////////////////////////////////////////////
            //NOP
            //Opcode
            InstructionMemory[MemLoc][31:25] = NOP;
            MemLoc = MemLoc + 1;
            ///////////////////////////////////////////////
            //MOV R3, R1
            //Opcode
            InstructionMemory[MemLoc][31:25] = MOV;
            //DR
            InstructionMemory[MemLoc][24:20] = R3;
            //SA
            InstructionMemory[MemLoc][19:15] = R1;
            //IM
            InstructionMemory[MemLoc][14:0] = 0;
            MemLoc = MemLoc + 1;

            //Start Program
            IR = InstructionMemory[0];
            end

            //Read
            always @ (PCin)
            begin
                  begin
                     IR = InstructionMemory[PCin];
                  end

            end

      endmodule

```

```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    16:33:39 11/15/2009
// Design Name:
// Module Name:    InstructionDecoder
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module InstructionDecoder(
    input [31:0] IRin,
    input Reset,
    output RWout,
    output [4:0] DAout,
    output [1:0] MDout,
    output [1:0] BSout,
    output PSout,
    output MWout,
    output [4:0] FSout,
    output MAout,
    output MBout,
    output [4:0] AAout,
    output [4:0] BAout  ,
    output CSout
    );

`include "Instructions.txt"

reg RWreg;
   assign RWout = RWreg;
reg [4:0] DAreg;
      assign DAout = DAreg;
reg [1:0] MDreg;
   assign MDout = MDreg;
reg [1:0] BSreg;
   assign BSout = BSreg;
reg PSreg;
   assign PSout = PSreg;
reg MWreg;
   assign MWout = MWreg;
reg [4:0] FSreg;
   assign FSout = FSreg;
reg MAreg;
   assign MAout = MAreg;
reg MBreg;
   assign MBout = MBreg;
reg [4:0] AAreg;
   assign AAout = AAreg;
reg [4:0] BAreg;
   assign BAout = BAreg;
```

```verilog
        reg CSreg;
           assign CSout = CSreg;

        //Clear outputs
        initial
           begin
              RWreg = 0;
              DAreg = 0;
              MDreg = 0;
              BSreg = 0;
              PSreg = 0;
              MWreg = 0;
              FSreg = 0;
           end

        //Hardware Reset Triggered
        always @ (posedge Reset)
           begin
               RWreg = 0;
               DAreg = 0;
               MDreg = 0;
               BSreg = 0;
               PSreg = 0;
               MWreg = 0;
               FSreg = 0;
           end

        //At each Instruction Change
        always @ (IRin)
            begin
              case (IRin[31:25])
              //////////////////////////////////////////
              NOP:
                 begin
                    RWreg = 0;
                    //MDreg = 2'bxx;
                    BSreg = 2'b00;
                    //PSreg = 1'bx;
                    MWreg = 0;
                    //FSreg = 5'bxxxxx;
                    //MBreg = 1'bx;
                    //MAreg = 1'bx;
                    //CSreg = 1'bx;

                    DAreg = IRin[24:20];
                    AAreg = IRin[19:15];
                    BAreg = IRin[14:10];


                 end
              //////////////////////////////////////////
              ADD:
                 begin
                    RWreg = 1;
                    MDreg = 2'b00;
                    BSreg = 2'b00;
                    //PSreg = 1'bx;
                    MWreg = 0;
                    FSreg = 5'b00010;
                    MBreg = 0;
                    MAreg = 0;
                    //CSreg = 1'bx;

                    DAreg = IRin[24:20];
                    AAreg = IRin[19:15];
                    BAreg = IRin[14:10];
                 end
              ///////////////////////////////////////////
              SUB:
                 begin
                    RWreg = 1;
                    MDreg = 2'b00;
                    BSreg = 2'b00;
                    //PSreg = 1'bx;
                    MWreg = 0;
                    FSreg = 5'b00100;
                    MBreg = 0;
                    MAreg = 0;
                    //CSreg = 1'bx;

                    DAreg = IRin[24:20];
                    AAreg = IRin[19:15];
                    BAreg = IRin[14:10];
                 end
              ///////////////////////////////////////////
              SLT:
                 begin
                    RWreg = 1;
                    MDreg = 2'b10;
                    BSreg = 2'b00;
                    //PSreg = 1'bx;
                    MWreg = 0;
                    FSreg = 5'b00101;
                    MBreg = 0;
                    MAreg = 0;
                    //CSreg = 1'bx;

                    DAreg = IRin[24:20];
                    AAreg = IRin[19:15];
                    BAreg = IRin[14:10];
                 end
                 ///////////////////////////////////////////
              AND:
                 begin
                    RWreg = 1;
                    MDreg = 2'b00;
                    BSreg = 2'b00;
                    //PSreg = 1'bx;
                    MWreg = 0;
                    FSreg = 5'b01000;
                    MBreg = 0;
                    MAreg = 0;
                    //CSreg = 1'bx;

                    DAreg = IRin[24:20];
                    AAreg = IRin[19:15];
                    BAreg = IRin[14:10];

                    DAreg = IRin[24:20];
                    AAreg = IRin[19:15];
                    BAreg = IRin[14:10];
                 end
```

```
            ////////////////////////////////////////////
        OR:
            begin
                RWreg = 1;
                MDreg = 2'b00;
                BSreg = 2'b00;
                //PSreg = 1'bx;
                MWreg = 0;
                FSreg = 5'b01010;
                MBreg = 0;
                MAreg = 0;
                //CSreg = 1'bx;

                DAreg = IRin[24:20];
                AAreg = IRin[19:15];
                BAreg = IRin[14:10];
            end
            ////////////////////////////////////////////
        XOR:
            begin
                RWreg = 1;
                MDreg = 2'b00;
                BSreg = 2'b00;
                //PSreg = 1'bx;
                MWreg = 0;
                FSreg = 5'b01100;
                MBreg = 0;
                MAreg = 0;
                //CSreg = 1'bx;

                DAreg = IRin[24:20];
                AAreg = IRin[19:15];
                BAreg = IRin[14:10];
            end
            ////////////////////////////////////////////
        ST:
            begin
                RWreg = 0;
                MDreg = 2'b00;
                BSreg = 2'b00;
                //PSreg = 1'bx;
                MWreg = 1;
                //FSreg = 5'bxxxxx;
                MBreg = 0;
                MAreg = 0;
                //CSreg = 1'bx;

                DAreg = IRin[24:20];
                AAreg = IRin[19:15];
                BAreg = IRin[14:10];
            end
            ////////////////////////////////////////////
        LD:
            begin
                RWreg = 1;
                MDreg = 2'b01;
                BSreg = 2'b00;
                //PSreg = 1'bx;
                MWreg = 0;
                //FSreg = 5'bxxxxx;
                //MBreg = 1'bx;
```

```
                    MAreg = 0;
                    //CSreg = 1'bx;

                    DAreg = IRin[24:20];
                    AAreg = IRin[19:15];
                    BAreg = IRin[14:10];
                 end
                 //////////////////////////////////////////
              ADI:
                 begin
                    RWreg = 1;
                    MDreg = 2'b00;
                    BSreg = 2'b00;
                    //PSreg = 1'bx;
                    MWreg = 0;
                    FSreg = 5'b00010;
                    MBreg = 1;
                    MAreg = 0;
                    CSreg = 1;

                    DAreg = IRin[24:20];
                    AAreg = IRin[19:15];

                 end
                 //////////////////////////////////////////
              SBI:
                 begin
                    RWreg = 1;
                    MDreg = 2'b00;
                    BSreg = 2'b00;
                    //PSreg = 1'bx;
                    MWreg = 0;
                    FSreg = 5'b00100;
                    MBreg = 1;
                    MAreg = 0;
                    CSreg = 1;

                    DAreg = IRin[24:20];
                    AAreg = IRin[19:15];
                 end
                 //////////////////////////////////////////
              NOT:
                 begin
                    RWreg = 1;
                    MDreg = 2'b00;
                    BSreg = 2'b00;
                    //PSreg = 1'bx;
                    MWreg = 0;
                    FSreg = 5'b01110;
                    //MBreg = 1'bx;
                    MAreg = 0;
                    //CSreg = 1'bx;

                    DAreg = IRin[24:20];
                    AAreg = IRin[19:15];
                 end
                 //////////////////////////////////////////
              ANI:
                 begin
                    RWreg = 1;
                    MDreg = 2'b00;
```

```
                        BSreg = 2'b00;
                        //PSreg = 1'bx;
                        MWreg = 0;
                        FSreg = 5'b01000;
                        MBreg = 1;
                        MAreg = 0;
                        CSreg = 0;

                        DAreg = IRin[24:20];
                        AAreg = IRin[19:15];
                     end
                     //////////////////////////////////////////
                  ORI:
                     begin
                        RWreg = 1;
                        MDreg = 2'b00;
                        BSreg = 2'b00;
                        //PSreg = 1'bx;
                        MWreg = 0;
                        FSreg = 5'b01010;
                        MBreg = 1;
                        MAreg = 0;
                        CSreg = 0;

                        DAreg = IRin[24:20];
                        AAreg = IRin[19:15];
                     end
                     //////////////////////////////////////////
                  XRI:
                     begin
                        RWreg = 1;
                        MDreg = 2'b00;
                        BSreg = 2'b00;
                        //PSreg = 1'bx;
                        MWreg = 0;
                        FSreg = 5'b01100;
                        MBreg = 1;
                        MAreg = 0;
                        CSreg = 0;

                        DAreg = IRin[24:20];
                        AAreg = IRin[19:15];
                     end
                     //////////////////////////////////////////
                  AIU:
                     begin
                        RWreg = 1;
                        MDreg = 2'b00;
                        BSreg = 2'b00;
                        //PSreg = 1'bx;
                        MWreg = 0;
                        FSreg = 5'b00010;
                        MBreg = 1;
                        MAreg = 0;
                        CSreg = 0;

                        DAreg = IRin[24:20];
                        AAreg = IRin[19:15];
                     end
                     //////////////////////////////////////////
                  SIU:
                     begin
                        RWreg = 1;
                        MDreg = 2'b00;
                        BSreg = 2'b00;
                        //PSreg = 1'bx;
                        MWreg = 0;
                        FSreg = 5'b00101;
                        MBreg = 1;
                        MAreg = 0;
                        CSreg = 0;

                        DAreg = IRin[24:20];
                        AAreg = IRin[19:15];
                     end
                     //////////////////////////////////////////
                  MOV:
                     begin
                        RWreg = 1;
                        MDreg = 2'b00;
                        BSreg = 2'b00;
                        //PSreg = 1'bx;
                        MWreg = 0;
                        FSreg = 5'b00000;
                        //MBreg = 1'bx;
                        MAreg = 0;
                        //CSreg = 1'bx;

                        DAreg = IRin[24:20];
                        AAreg = IRin[19:15];
                     end
                  //////////////////////////////////////////
                  LSL:
                     begin
                        RWreg = 1;
                        MDreg = 2'b00;
                        BSreg = 2'b00;
                        //PSreg = 1'bx;
                        MWreg = 0;
                        FSreg = 5'b10100;
                        //MBreg = 1'bx;
                        MAreg = 0;
                        //CSreg = 1'bx;

                        DAreg = IRin[24:20];
                        AAreg = IRin[19:15];
                     end
                     //////////////////////////////////////////
                  LSR:
                     begin
                        RWreg = 1;
                        MDreg = 2'b00;
                        BSreg = 2'b00;
                        //PSreg = 1'bx;
                        MWreg = 0;
                        FSreg = 5'b11000;
                        //MBreg = 1'bx;
                        MAreg = 0;
                        //CSreg = 1'bx;

                        DAreg = IRin[24:20];
                        AAreg = IRin[19:15];
```

```
                end
                //////////////////////////////////////////
             JMR:
                begin
                   RWreg = 0;
                   //MDreg = 2'bxx;
                   BSreg = 2'b10;
                   //PSreg = 1'bx;
                   MWreg = 0;
                   FSreg = 5'b00000;
                   //MBreg = 1'bx;
                   //MAreg = 1'bx;
                   //CSreg = 1'bx;

                   DAreg = IRin[24:20];
                   AAreg = IRin[19:15];
                end
                //////////////////////////////////////////
             BZ:
                begin
                   RWreg = 0;
                   //MDreg = 2'bxx;
                   BSreg = 2'b01;
                   PSreg = 0;
                   MWreg = 0;
                   FSreg = 5'b00000;
                   MBreg = 1;
                   MAreg = 0;
                   CSreg = 1;

                   DAreg = IRin[24:20];
                   AAreg = IRin[19:15];
                end
                //////////////////////////////////////////
             BNZ:
                begin
                   RWreg = 0;
                   //MDreg = 2'bxx;
                   BSreg = 2'b01;
                   PSreg = 1;
                   MWreg = 0;
                   FSreg = 5'b00000;
                   MBreg = 1;
                   MAreg = 0;
                   CSreg = 1;

                   DAreg = IRin[24:20];
                   AAreg = IRin[19:15];
                end
                //////////////////////////////////////////
             JMP:
                begin
                   RWreg = 0;
                   //MDreg = 2'bxx;
                   BSreg = 2'b11;
                   //PSreg = 1'bx;
                   MWreg = 0;
                   //FSreg = 5'bxxxxx;
                   MBreg = 1;
                   //MAreg = 1'bx;
                   CSreg = 1;
```

```

                    DAreg = IRin[24:20];
                    AAreg = IRin[19:15];
                 end
                 //////////////////////////////////////////
              JML:
                 begin
                    RWreg = 1;
                    MDreg = 2'b00;
                    BSreg = 2'b11;
                    //PSreg = 1'bx;
                    MWreg = 0;
                    FSreg = 5'b00111;
                    MBreg = 1;
                    MAreg = 1;
                    CSreg = 1;

                    DAreg = IRin[24:20];
                    AAreg = IRin[19:15];
                 end
              endcase
           end

        endmodule

```

```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    16:05:37 11/15/2009
// Design Name:
// Module Name:    FunctionUnit
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module FunctionUnit(
    input signed [31:0] Ains,
    input signed [31:0] Bins,
    input [31:0] Ainu,
    input [31:0] Binu,
    input [4:0] SHin,
    input [4:0] FSin,
    input Reset,
    output Zout,
    output Vout,
    output Nout,
    output Cout,
    output [31:0] Fout
    );

`include "FSCodes.txt"

reg Zreg, Vreg, Nreg, Creg;
reg [32:0] Freg;
reg signed [32:0] Fregs;
   assign Fout = Freg[31:0];

reg [31:0] ShiftBuf;
reg i;

assign Zout = Zreg;
assign Vout = Vreg;
assign Nout = Nreg;
assign Cout = Creg;

reg k;


initial
   begin
   ShiftBuf = 0;
      Zreg = 0;
      Vreg = 0;
      Nreg = 0;
```

```
              Creg = 0;
              Freg = 0;

           end

        always @ (negedge Reset)
           begin
              ShiftBuf = 0;
              Zreg = 0;
              Vreg = 0;
              Nreg = 0;
              Creg = 0;
              Freg = 0;
           end

        always @ *
           begin
              case (FSin)
                 ////////////////////////////////////////////////////
                 ADD:
                       begin
                         Freg = Ains + Bins;
                         Fregs = Freg;

                         Zreg = Freg==0;
                         Nreg = Freg[31];

                         if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
                         else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
                         else Vreg = 0;

                         Creg = Freg[32];
                       end
                       ////////////////////////////////////////////////////
                 SUB:
                       begin
                         Freg = Ains - Bins;
                         Fregs = Freg;
                         Zreg = Freg==0;
                         Nreg = Freg[31];
                         if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
                         else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
                         else Vreg = 0;
                         Creg = Freg[32];
                       end
                       ////////////////////////////////////////////////////
                 SLT:
                       begin
                          if(Ains < Bins)Freg = 1;
                          else Freg = 0;
                          Fregs = Freg;
                         Zreg = Freg==0;
                         Nreg = Freg[31];
                         if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
                         else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
                         else Vreg = 0;
                         Creg = Freg[32];
                       end
                       ////////////////////////////////////////////////////
                 AND:
                       begin
                         Freg = Ains & Bins;
                         Fregs = Freg;
                         Zreg = Freg==0;
                         Nreg = Freg[31];
                         if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
                         else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
                         else Vreg = 0;
                         Creg = Freg[32];

                       end
                       ////////////////////////////////////////////////////
                 OR:
                       begin

                         Freg = Ains | Bins;
                         Fregs = Freg;
                         Zreg = Freg==0;
                         Nreg = Freg[31];
                         if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
                         else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
                         else Vreg = 0;
                         Creg = Freg[32];

                       end
                       ////////////////////////////////////////////////////
                 XOR:
                       begin

                         Freg = Ains ^ Bins;
                         Fregs = Freg;
                         Zreg = Freg==0;
                         Nreg = Freg[31];
                         if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
                         else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
                         else Vreg = 0;
                         Creg = Freg[32];

                       end
                          ////////////////////////////////////////////////////
                 NOT:
                       begin

                         Freg = ~Ains;
                         Fregs = Freg;
                         Zreg = Freg==0;
                         Nreg = Freg[31];
                         if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
                         else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
                         else Vreg = 0;
                         Creg = Freg[32];

                       end
                       ////////////////////////////////////////////////////
                 AIU:
                       begin

                         Freg = Ainu + Binu;

                         Zreg = Freg==0;
                         Nreg = Freg[31];

```

```verilog
                    Vreg = Freg[32];

                    Creg = Freg[32];

                  end
                  /////////////////////////////////////////////////////
            SIU:
                  begin

                    Freg = Ainu - Binu;
                    Zreg = Freg==0;
                    Nreg = Freg[31];
                    Vreg = Freg[32];
                    Creg = Freg[32];

                  end
                  /////////////////////////////////////////////////////
            MOV:
                  begin

                    Freg = Ainu;
                    Zreg = Freg==0;
                    Nreg = Freg[31];
                    Vreg = Freg[32];
                    Creg = Freg[32];

                  end
                  /////////////////////////////////////////////////////
            LSL:
                  begin

                    Freg = Ains << SHin;
                    Fregs = Freg;
                    Zreg = Freg==0;
                    Nreg = Freg[31];
                    if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
                    else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
                    else Vreg = 0;
                    Creg = Freg[32];
                  end
                  /////////////////////////////////////////////////////
            LSR:
                  begin

                   Freg = Ains >> SHin;
                   Fregs = Freg;
                    Zreg = Freg==0;
                    Nreg = Freg[31];
                    if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
                    else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
                    else Vreg = 0;
                    Creg = Freg[32];

                  end


            JML:
                  begin
                     Freg = Ains;
                     Zreg = Freg==0;
                    Nreg = Freg[31];
```

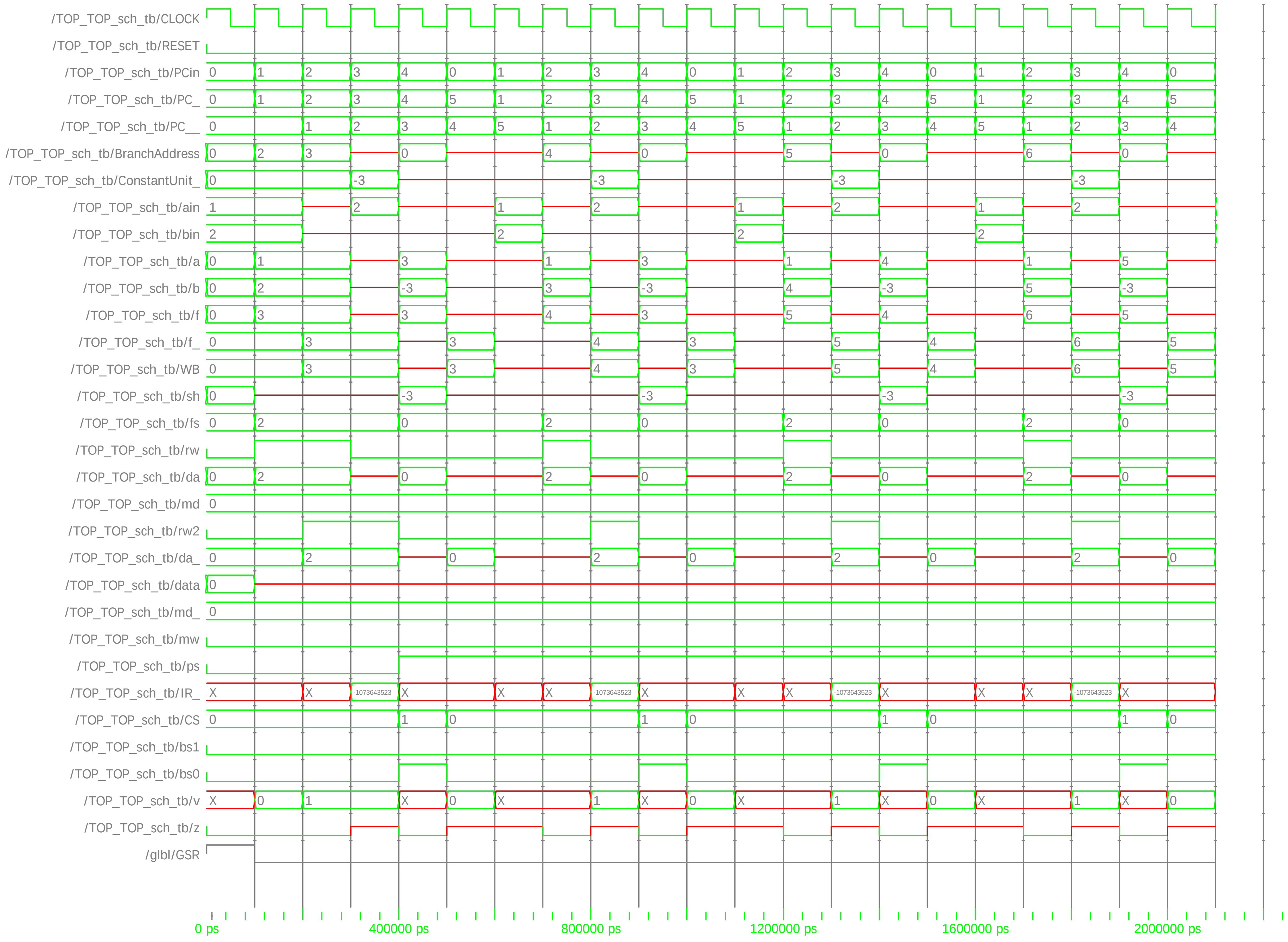
```verilog
                 if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
                 else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
                 else Vreg = 0;
                 Creg = Freg[32];

               end

      endcase

   end

endmodule

```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: 
// Engineer: 
// 
// Create Date:    16:12:33 11/15/2009 
// Design Name: 
// Module Name:    RegisterFile 
// Project Name: 
// Target Devices: 
// Tool versions: 
// Description: 
//
// Dependencies: 
//
// Revision: 
// Revision 0.01 - File Created
// Additional Comments: 
//
//////////////////////////////////////////////////////////////////////////////////
module RegisterFile(
   //Read ports
    input [4:0] AAin,
    input [4:0] BAin,
    output [31:0] Aout,
    output [31:0] Bout,

    //Write Ports
    input [31:0] Din,
    input RWin,
    input [4:0] DAin,

    //Control Signals
    input Clock,
    input Reset

    );

    `include "Instructions.txt"
    `include "Registers.txt"

    reg [31:0] Areg;
    reg [31:0] Breg;

    assign Aout = Areg;
    assign Bout = Breg;


    reg [31:0] RegisterMemory  [31:0];   // 32 Registers of 32 bits

    //Loop counter
    reg k;

    always @ (posedge Reset)
      begin
         Areg = 0;
         Breg = 0;

      end

    initial
```

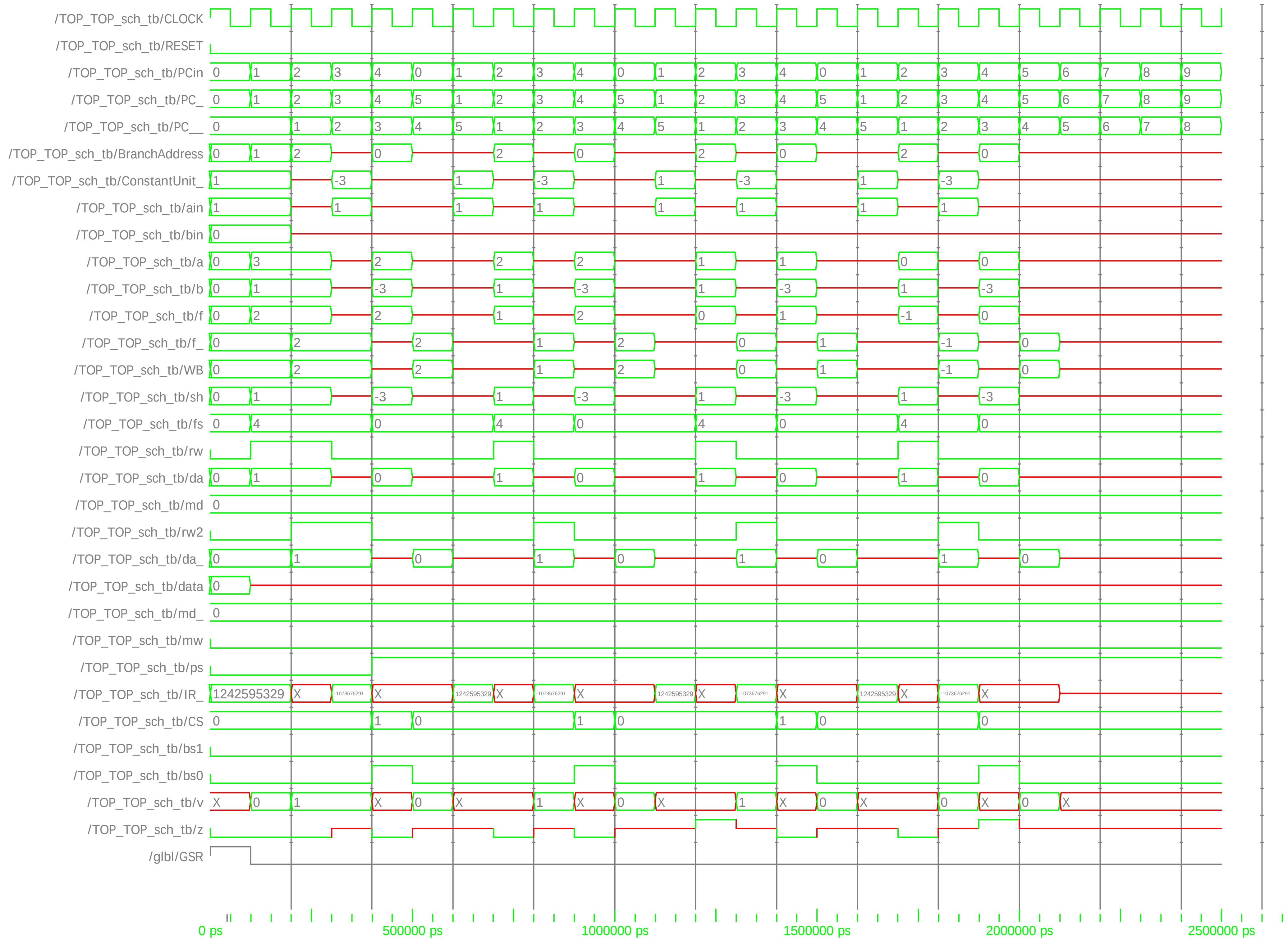
```
            begin
               Areg = 0;
               Breg = 0;

               RegisterMemory[R1] = 3;
               RegisterMemory[R2] = R2;
               RegisterMemory[R3] = R3;
               RegisterMemory[R4] = R4;
               RegisterMemory[R5] = R5;


            end



            //Read
            always @ (negedge Clock)
            begin
               if(AAin == 0)
                  begin
                     Areg = 0;
                  end
               else
                  begin
                     Areg = RegisterMemory[AAin];
                  end
               if(BAin == 0)
                  begin
                     Breg = 0;
                  end
               else
                  begin
                     Breg = RegisterMemory[BAin];
                  end
            end

            //Write
            always @ (posedge Clock)
            begin
            //If write is enabled
               if(RWin == 1)
                  begin
                  //If you are sending data to R0, throw data away
                  if(DAin == 0)
                     begin
                        RegisterMemory[0] = 0;
                     end
                     //Else if there is a valid Register address
                     else
                     begin
                        //Store the data in Register[DAin]
                        RegisterMemory[DAin] = Din;
                     end
                  end
               end
        endmodule

```

```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    12:47:21 11/18/2009
// Design Name:
// Module Name:    PC
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module PC(
   input Clock,
   input Reset,
    input [31:0] Inpin,
    output [31:0] Outpin
    );

reg [31:0] Buffer;
assign Outpin = Buffer;

initial
begin
   Buffer = 0;
end

always @ (Reset)
begin
   Buffer = 0;
end

always @ (posedge Clock)
begin
   Buffer = Inpin;
end


endmodule

```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: 
// Engineer: 
// 
// Create Date:    12:33:06 11/18/2009 
// Design Name: 
// Module Name:    MuxA 
// Project Name: 
// Target Devices: 
// Tool versions: 
// Description: 
//
// Dependencies: 
//
// Revision: 
// Revision 0.01 - File Created
// Additional Comments: 
//
//////////////////////////////////////////////////////////////////////////////////
module MuxA(
    input [31:0] Zero,
    input [31:0] One,
    output [31:0] BusA,
    input MA
    );

    reg [31:0] Selected;
    assign BusA = Selected;

initial
   begin
      Selected = 0;
   end

always @ *
   begin
      if(MA == 0)
      begin
         Selected = Zero;
      end
      else if(MA == 1)
      begin
         Selected = One;
      end
   end

endmodule
```



Entity:TOP_TOP_sch_tb Architecture: Date: Mon Nov 30 5:42:28 PM Central Standard Time 2009   Row: 1 Page: 1



Entity:TOP_TOP_sch_tb Architecture: Date: Mon Nov 30 5:57:20 PM Central Standard Time 2009   Row: 1 Page: 1