EE/CS 480

INSTRUCTION SET AND IMPLEMENTING PROCESSOR ARCHITECTURE DEVELOPMENT AND VALIDATION PROJECT

SPRING SEMESTER 2012

INSTRUCTOR: J. ROBERT HEATH

Timothy Zaid Mullins

Caroline Jones

April 26, 2012

**Contents**

Overview……………………………………………………………………………………….………3

Goals…………………………………………………………………………………………..………..3

Objectives………………………………………………………………………………………..……..3

ISA……………………………………………………………………………………………………..4

Computer Organization/Architecture Schematics…………………………………………..………….6

System Flowchart and Controller State Diagram………………………………………….…………...9

Controller State Table..……………………………………………………………………………….14

Verilog Description of Architecture…………………………………………………………………..16

Test Programs…………………………………………………………………………………..……..67

Simulation Output……………………………………………………………………………………71

Program 1………………………………………………………………………………………..71

Program 2………………………………………………………………………………………..75

Program 3……………………………………………………………………………………......79

Program 4………………………………………………………………………………………..81

Conclusion…………………………………………………………………………………………….83

**Overview**

In this project we were to design a Memory-Register (MR) type Instruction Set Archetecture (ISA), model the underlying hardware to support that architecture with a hardware description language (HDL), and then verify its operation via post place and route simulation using the Xilinx development environment in conjunction with ModelSim SE simulation software. This ISA and resulting processor were to be our “own” in that we were to design each of them from the ground up, adding as many performance based features as we were able to do. The ISA was to use 16-bit instructions and operate on an 8-bit data path. The instructions that needed to be developed included ones for doing all Arethematic Logic Unit (ALU) functions (addition, subtraction, AND, OR, and complementing), conditional branches, unconditional branches, jumping to and returning from subroutines, shifts, moving register contents between each of them, loading and storing from/to memory, inputting and outputting to the main input and output busses, and enabling the Hardware Vector Priority Interrupt (HVPI) system. The HDL descriptions of the underlying hardware were to be done using the VERILOG HDL.

**Goals and Objectives**

1. Develop a MR type ISA
2. Develop HDL descriptions of all hardware units needed to implement the ISA
3. Develop a controller to control processor operation
4. Implement a fully functional direct mapped cache
5. Implement a 2-stage pipelined design
6. Develop a HVPI system
7. Write test programs to thoroughly test processor operation, including straight program flow, looping structures, subroutines, and interrupts.
8. Simulate post place and route functionality
9. Demonstrate functioning processor by running the test programs via post place and route simulation

**Achieved and Non-Achieved Goals and Objectives**

We were able to fully implement and test all of the planned features and acheieve all of the goals that we had for our project. We correctly implemented the hardware to execute our developed ISA with a 2-stage pipelined design that included a direct mapped cache and a HVPI system. Our processor was a true single bus machine, with all register inputs and outputs connected to the same bus. The functionality was verified using a post place and route simulation of our 4 tests programs where we confirmed that the correct operations were happening by examining both the instruction being executed and the data that was on the main data path bus.

**ISA**

Base instructions are 16 bits long.

* 8 registers (R0-R7)
* RAM: addresses $00-$FF
* Input Bus
* Output Bus

1. ADD/SUB/AND/OR

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| RD or M[RD] = RA + N | 0000 | RD | M | RA | M | N |
| = RA + M[N] | 0001 | RD | M | RA | M | N |
| = M[RA] + N | 0010 | RD | M | RA | M | N |
| = M[RA] + M[N] | 0011 | RD | M | RA | M | N |

M = 0 direct

M =1 indirect

1. COMP

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| RD or M[RD] = ~RA or ~M[RA] | 0100 | RD | M | RA | M |

1. Branch

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| BGT RA > RB | 0110 | 000 |  |  | RA | RB |
| BGE RA >= RB | 0110 | 001 |  |  | RA | RB |
| BLT RA < RB | 0110 | 010 |  |  | RA | RB |
| BLE RA <= RB | 0110 | 011 |  |  | RA | RB |
| BEQ RA == RB | 0110 | 100 |  |  | RA | RB |
| BNQ RA != RB | 0110 | 101 |  |  | RA | RB |

1. Shift

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SLL | 0101 | R |  | 00 | C | N |
| SLR | 0101 | R |  | 01 | C | N |
| SAL | 0101 | R |  | 10 | C | N |
| SAR | 0101 | R |  | 11 | C | N |

N = # bits to shift

C = 1 clear all bits

1. JMP

|  |  |  |
| --- | --- | --- |
| 0111 |  | addr |

1. JSR

|  |  |  |
| --- | --- | --- |
| 1000 |  | addr |

PC = R7

addr = PC

1. RTS

|  |  |
| --- | --- |
| 1001 |  |

R7 = PC

1. MV

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RD = RA | 1010 | RD |  | RA |

1. Store

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| M[addr] = RD | 1100 | RD | 0 | addr | |
| M[RA] = RD | 1100 | RD | 1 |  | RA |

1. INPUT

|  |  |  |
| --- | --- | --- |
| 1101 |  | addr |

M[addr] = in bus

1. OUTPUT

|  |  |  |
| --- | --- | --- |
| 1110 |  | addr |

out bus = M[addr]

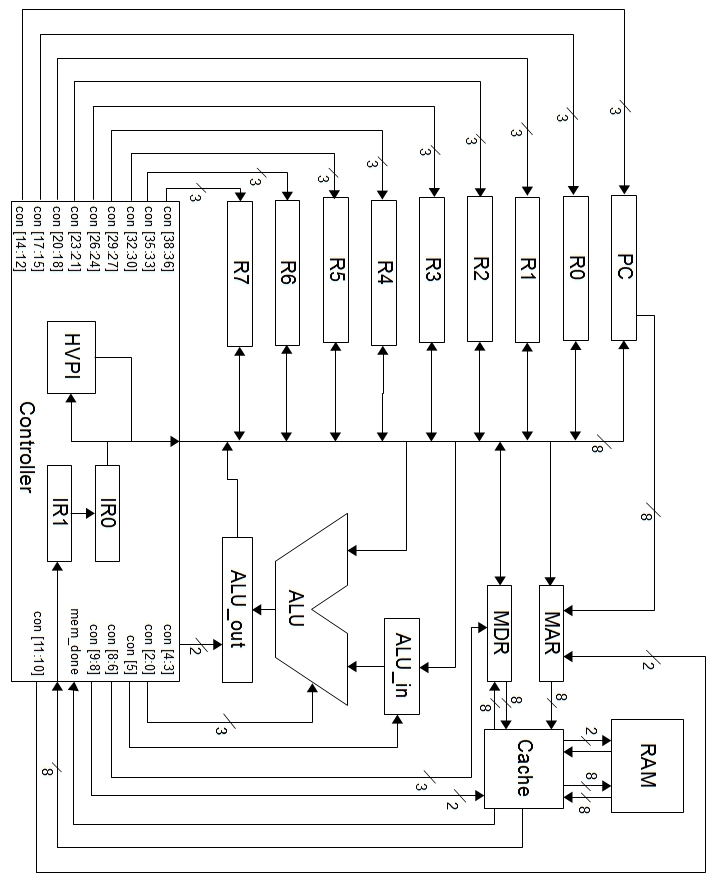
1. HVPI

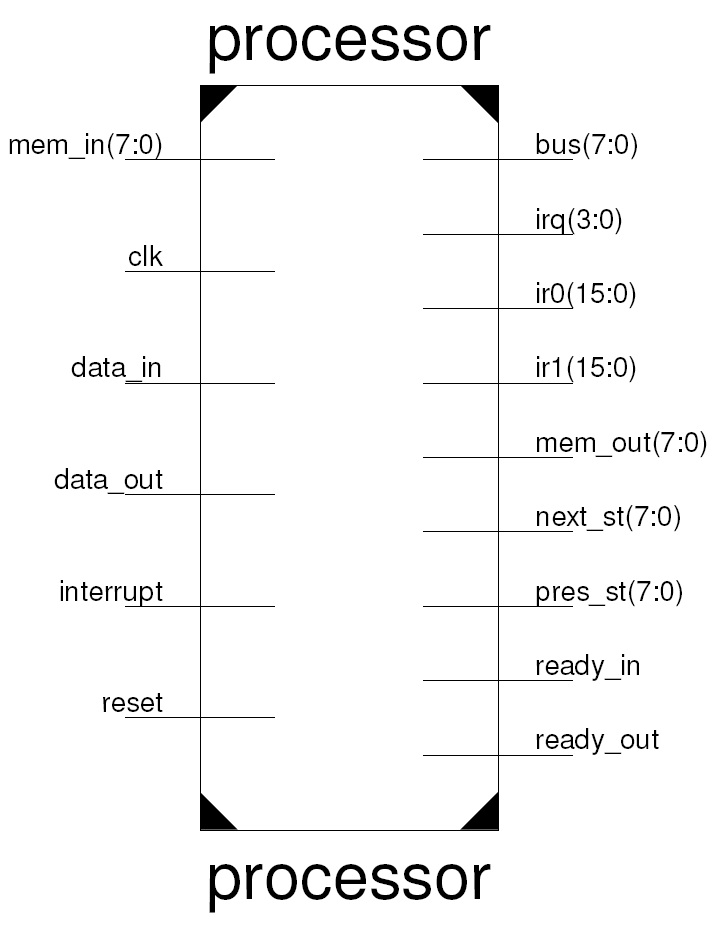
|  |  |  |  |
| --- | --- | --- | --- |
| 1111 |  | EN | MASK |

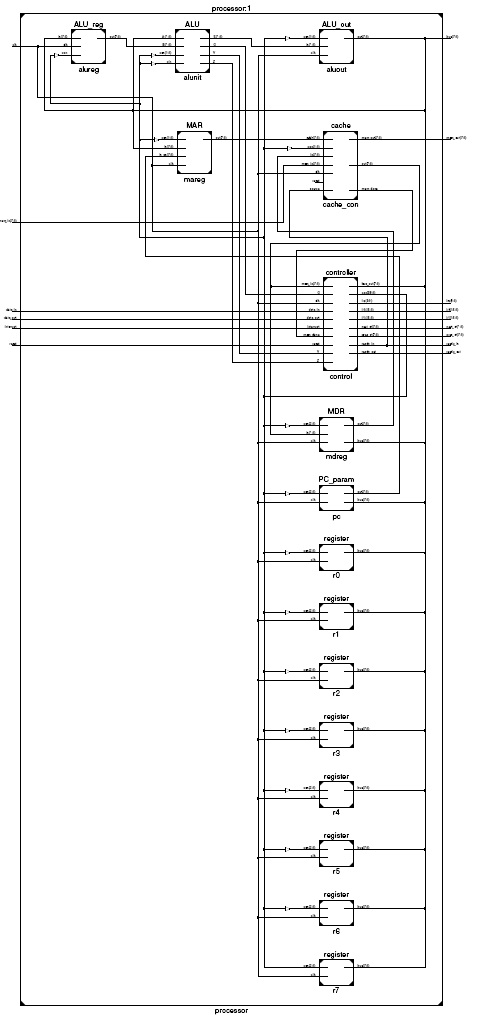
1. LOAD

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RD = N | 1011 | RD | 0 | N |
| RD = M[N] | 1011 | RD | 1 | N |

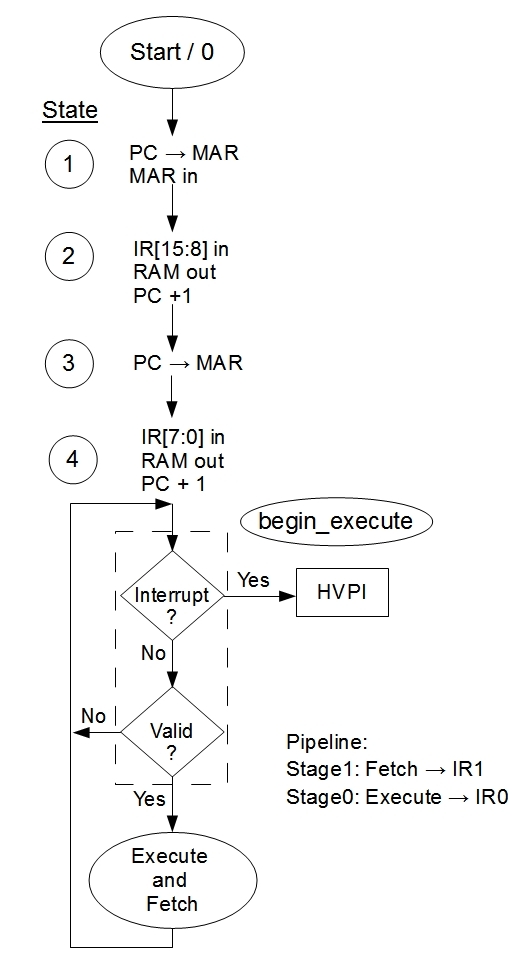
**Computer Organization/Architecture Schematics**

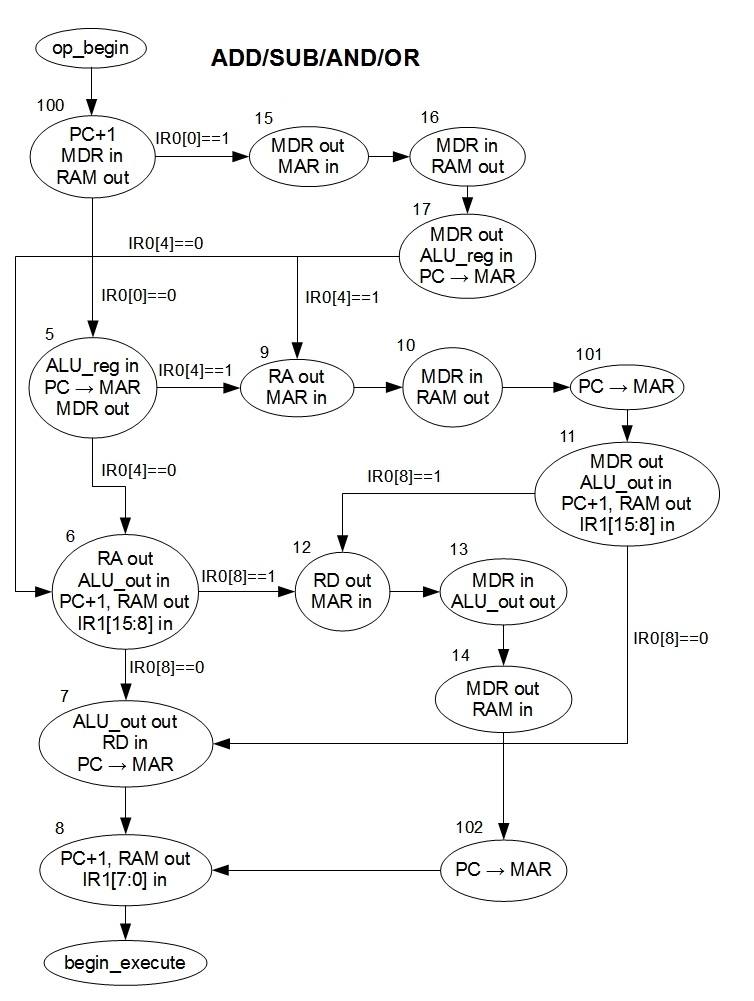
****

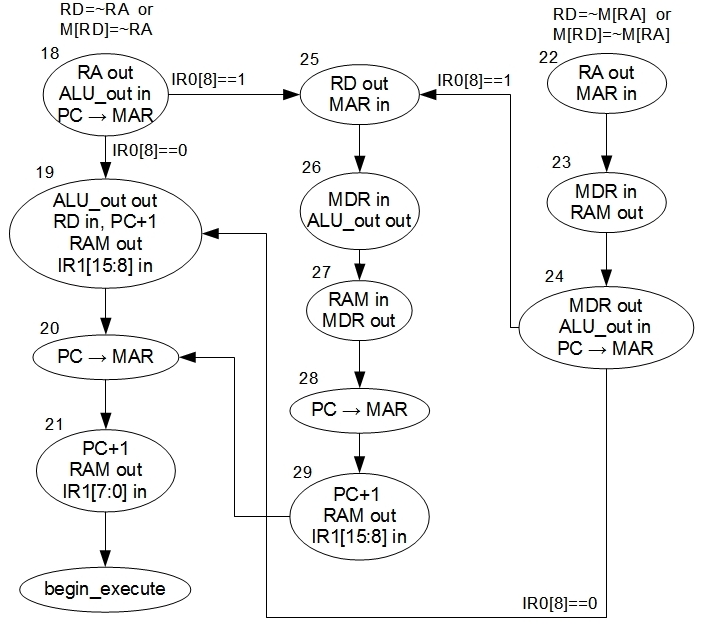
****

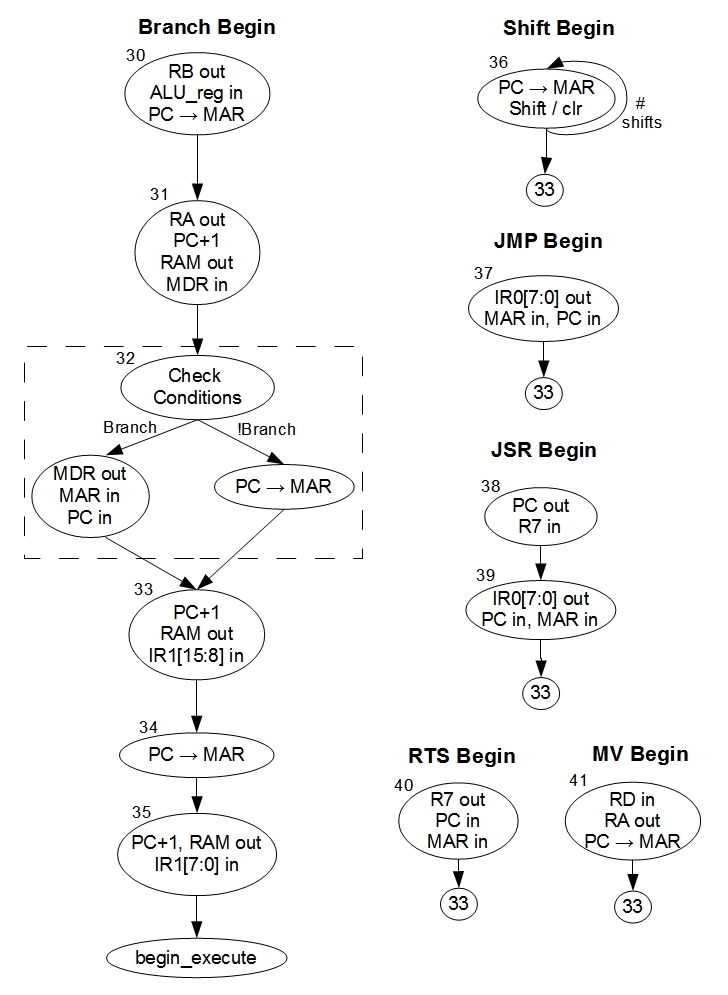
****

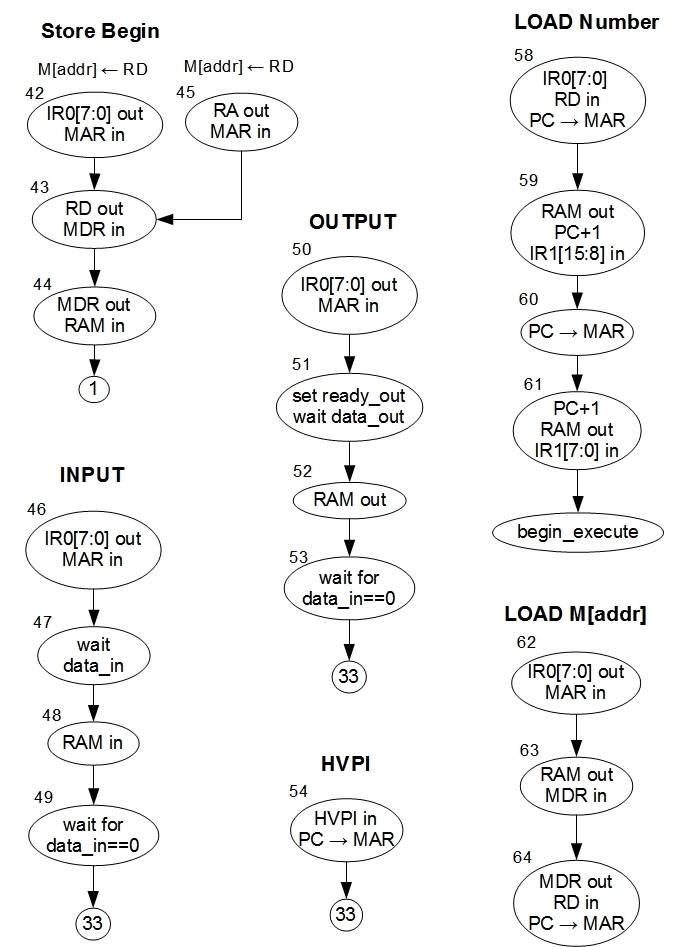
**System Flowchart and Controller State Diagram**

****

****

****

****

****

**Controller State Table**

**Verilog Description of Architecture**

`timescale 1ns / 1ps

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

// Company: University of Kentucky

// Engineer: Zaid Mullins and Caroline Jones

//

// Create Date: 21:58:51 04/21/2012

// Module Name: processor.v

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

module processor(bus, reset, clk, ir0, ir1, pres\_st, next\_st, mem\_in, mem\_out, data\_in, data\_out, ready\_in, ready\_out, interrupt, irq);

input reset, clk, data\_in, data\_out, interrupt;

input [7:0] mem\_in;

output [7:0] bus, pres\_st, next\_st, mem\_out;

output [15:0] ir0, ir1;

output ready\_in, ready\_out;

output [3:0] irq;

wire [38:0] con;

wire [7:0] mdr\_in, mdr\_out, ALU\_reg\_out, ALU\_out\_in, pc\_mar, mar\_out;

wire mem\_done, mem\_ready\_out, out\_ready, mem\_ready\_in, in\_ready, C, V, Z;

//module controller(mem\_in, mem\_done, C, V, Z, reset, clk, bus\_out, con, ir0, ir1, pres\_st, next\_st, ready\_in, ready\_out, data\_in, data\_out);

controller control(mdr\_in, mem\_done, C, V, Z, reset, clk, bus, con, ir0, ir1, pres\_st, next\_st, ready\_in, ready\_out, data\_in, data\_out, interrupt, irq);

//module ALU(A, B, cin, con, S, C, V, Z);

//module ALU\_reg(in, con, clk, out);

//module ALU\_out(in, con, clk, out);

ALU alunit(bus, ALU\_reg\_out, con[0], con[2:1], ALU\_out\_in, C, V, Z);

ALU\_reg alureg(bus, con[5], clk, ALU\_reg\_out);

ALU\_out aluout(ALU\_out\_in, con[4:3], clk, bus);

//module MDR(bus, in, con, clk, out);

MDR mdreg(bus, mdr\_in, con[8:6], clk, mdr\_out);

//module PC\_param(bus, clk, con, out);

PC\_param pc(bus, clk, con[14:12], pc\_mar);

//module MAR(in, in\_pc, con, clk, out);

MAR mareg(bus, pc\_mar, con[11:10], clk, mar\_out);

//cache(in, addr, con, clk, mem\_done, out, mem\_in, mem\_out, source, reset);

cache cache\_con(mdr\_out, mar\_out, con[9:8], clk, mem\_done, mdr\_in, mem\_in, mem\_out, ready\_in, 1'b0);

//module RAM(in, addr, con, clk, mem\_done, out);

//RAM rama(mdr\_out, mar\_out, con[9:8], clk, mem\_done, mdr\_in, mem\_in, mem\_out, ready\_in);

//module register(bus, con, clk);

register r0(bus, con[17:15], clk);

register r1(bus, con[20:18], clk);

register r2(bus, con[23:21], clk);

register r3(bus, con[26:24], clk);

register r4(bus, con[29:27], clk);

register r5(bus, con[32:30], clk);

register r6(bus, con[35:33], clk);

register r7(bus, con[38:36], clk);

endmodule

module ALU(A, B, cin, con, S, C, V, Z);

parameter n=8;

input [n-1:0] A, B;

input [1:0] con;

input cin;

output [n-1:0] S;

output C, V, Z;

wire [n:0] temp\_cin;

wire [n-1:0] B\_temp;

genvar i;

assign B\_temp = (cin)?~B:B;

assign temp\_cin[0]=cin;

assign C = S[n-1];

assign Z = (S==0)?1'b1:1'b0;

xor(V,temp\_cin[n],temp\_cin[n-1]);

generate

for(i=0;i<n;i=i+1) begin:ALUNIT

alu\_slice alunit(A[i],B\_temp[i],temp\_cin[i], con, S[i], temp\_cin[i+1]);

end

endgenerate

endmodule

module alu\_slice(a, b, cin, con, s, cout);

input a, b, cin;

input [1:0] con;

output reg s, cout;

always @ (a, b, cin, con)

begin

case (con)

0:begin {cout,s} = a + b + cin; end

1:begin s = a & b; cout = cin; end

2:begin s = a | b; cout = cin; end

3:begin s = (!cin)?~a:b; cout = cin; end

default:begin end

endcase

end

endmodule

module ALU\_reg(in, con, clk, out);

parameter n=8;

input [n-1:0] in;

input con, clk;

output [n-1:0] out;

genvar i;

generate

for(i=0;i < n;i=i+1) begin:register

reg\_alu\_in reg\_n(in[i], con, clk, out[i]);

end

endgenerate

endmodule

module reg\_alu\_in(in, con, clk, out);

input in, con, clk;

output out;

dff flip(mux\_out, 1'b0, 1'b0, clk, out);

mux\_2x1 mux({in, out}, con, mux\_out);

endmodule

module ALU\_out(in, con, clk, out);

parameter n=8;

input [n-1:0] in;

input [1:0] con;

input clk;

output [n-1:0] out;

genvar i;

wire [n-1:0] temp\_out;

assign out = (con[1] && ~con[0])?temp\_out:8'hz;

generate

for(i=0;i<8;i=i+1) begin:alu\_reg

reg\_alu\_out alu\_n(in[i], con, clk, temp\_out[i]);

end

endgenerate

endmodule

module reg\_alu\_out(in, con, clk, out);

input in, clk;

input [1:0] con;

output out;

dff flip(mux\_out, con[1]&&con[0], 1'b0, clk, out);

mux\_2x1 mux({in, out}, ~con[1]&&con[0], mux\_out);

endmodule

module dff(d, clr, set, clk, q);

input d, clr, set, clk;

output reg q;

always @ (negedge(clk))

begin

if(clr) q = 0;

else if (set) q = 1;

else q = d;

end

endmodule

module mux\_2x1(in, s, out);

input [1:0] in;

input s;

output reg out;

always @ (in, s)

begin

out=in[s];

end

endmodule

module MDR(bus, in, con, clk, out);

parameter n=8;

inout [n-1:0] bus;

input [n-1:0] in;

input [2:0] con;

input clk;

output [n-1:0] out;

wire [n-1:0] temp\_in;

genvar i;

assign bus = (con[1] && ~con[0])?out:8'hz;

assign temp\_in = (con[2] && ~con[1] && con[0])?in:bus;

generate

for(i=0;i<8;i=i+1) begin:alu\_reg

reg\_mdr alu\_n(temp\_in[i], con, clk, out[i]);

end

endgenerate

endmodule

module reg\_mdr(in, con, clk, out);

input in, clk;

input [2:0] con;

output out;

dff flip(mux\_out, ~con[2]&&con[1]&&con[0], 1'b0, clk, out);

mux\_2x1 mux({in, out}, ~con[1]&&con[0], mux\_out);

endmodule

module PC\_param(bus, clk, con, out);

parameter n=8;

inout [n-1:0] bus;

input clk;

input [2:0] con;

output reg [n-1:0] out;

reg check=0;

assign bus=(~con[2] && con[1] && ~con[0])?out:'hz;

always @ (negedge clk)

begin

if(con==0) begin

check=0;

out=out;

end else if (con==1) begin

check=0;

out=bus;

end else if (con==2) begin

out=out;

check=0;

end else if (con==3) begin

out=0;

check=0;

end else if (con==4 && !check) begin

check=1;

out=out+1;

end else if (con==4 && check) begin

out=out;

end else begin

check=0;

out=out;

end

end

endmodule

module register(bus, con, clk);

parameter n = 8;

inout [n-1:0] bus;

input [2:0] con;

input clk;

wire [n-1:0] out;

wire shift;

genvar i;

assign bus=(!con[2] && con[1] && !con[0])?out:8'hz;

assign shift=(con[1])?out[7]:1'b0;

//register\_slice(in, left, right, con, clk, out);

register\_slice zero(bus[0],out[1],1'b0,con,clk,out[0]);

register\_slice nth(bus[n-1],shift,out[n-2],con,clk,out[n-1]);

generate

for (i=1; i <= n-2; i = i+1) begin:register

register\_slice n(bus[i],out[i+1],out[i-1],con,clk,out[i]);

end

endgenerate

endmodule

module register\_slice(in, left, right, con, clk, out);

input in, clk, left, right;

input [2:0] con;

output out;

wire q;

wire [1:0] temp\_con;

assign temp\_con[1] = con[2];

assign temp\_con[0] = (con[2])?con[0]:(~con[1] && con[0]);

dff flip(ff\_in, ~con[2]&con[1]&con[0], 1'b0, clk, out);

mux\_4x1 mux({left, right, in, out},temp\_con,ff\_in);

endmodule

module mux\_4x1(in, s, out);

input [1:0] s;

input [3:0] in;

output reg out;

always @ (in, s)

begin

out = in[s];

end

endmodule

module MAR(in, in\_pc, con, clk, out);

parameter n=8;

input [n-1:0] in, in\_pc;

input [1:0] con;

input clk;

output [n-1:0] out;

wire [n-1:0] bit\_in;

genvar i;

assign bit\_in = (con[0])?in:in\_pc;

generate

for(i=0;i < n;i=i+1) begin:register

mar\_bit reg\_n(bit\_in[i], con, clk, out[i]);

end

endgenerate

endmodule

module mar\_bit(in, con, clk, out);

input [1:0] con;

input in, clk;

wire bit\_con;

output out;

xor(bit\_con, con[0], con[1]);

dff flip(mux\_out, con[1] && con[0], 1'b0, clk, out);

mux\_2x1 mux({in, out}, bit\_con, mux\_out);

endmodule

`timescale 1ns / 1ps

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

// Company: University of Kentucky

// Engineer: Zaid Mullins and Caroline Jones

//

// Create Date: 11:20:52 04/21/2012

// Module Name: controller.v

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

module controller(mem\_in, mem\_done, C, V, Z, reset, clk, bus\_out, con, ir0, ir1, pres\_st, next\_st, ready\_in, ready\_out, data\_in, data\_out, interrupt, irq);

input [7:0] mem\_in;

input mem\_done, C, V, Z, reset, clk, data\_in, data\_out, interrupt;

output reg [7:0] bus\_out;

output reg [38:0] con;

output reg [15:0] ir0, ir1;

output reg [7:0] pres\_st, next\_st;

output reg ready\_in=0, ready\_out=0;

reg [3:0] int\_mask=0;

output reg [3:0] irq=0;

reg [2:0] code, condition=0;

reg int\_en=0;

integer count, illegal\_ops=0;

parameter begin\_execute=8'b01111111, op\_begin=8'b01111110;

//registers

parameter r0=3'b000, r1=3'b001, r2=3'b010, r3=3'b011, r4=3'b100, r5=3'b101, r6=3'b110, r7=3'b111;

parameter r\_str=3'b000, r\_ld=3'b001, r\_asrt=3'b010, r\_clr=3'b011, r\_sll=3'b100, r\_slr=3'b101, r\_sal=3'b110, r\_sar=3'b111;

always @ (pres\_st, mem\_in, mem\_done, data\_in, data\_out, C, V, Z, int\_en, count)

begin

condition=condition;

code=code;

int\_mask=int\_mask;

irq=irq;

ready\_in=ready\_in;

ready\_out=ready\_out;

ir0=ir0;

ir1=ir1;

int\_en=int\_en;

case (pres\_st)

//start

0:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b11; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b011; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b11; //MAR

con[14:12]=3'b011;//PC

con[17:15]=r\_clr;

con[20:18]=r\_clr;

con[23:21]=r\_clr;

con[26:24]=r\_clr;

con[29:27]=r\_clr;

con[32:30]=r\_clr;

con[35:33]=r\_clr;

con[38:36]=r\_clr;

next\_st=1;

bus\_out='hz;

ready\_in=0;

ready\_out=0;

end

//pc->mar

1:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_in=0;

ready\_out=0;

next\_st=2;

end

//ram out, pc+1, ir0[15:8] load

2:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b100; //MDR

con[9]=2'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b100;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(mem\_done) begin

next\_st=3;

ir1[15:8]=mem\_in;

end else next\_st=2;

end

//pc->mar

3:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=4;

end

//ram out, pc+1, ir0[7:0], validate, interrupt check

4:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b100; //MDR

con[9]=2'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b100;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_in=0;

ready\_out=0;

if(mem\_done) begin

next\_st=begin\_execute;

ir1[7:0]=mem\_in;

end

else next\_st=4;

end

begin\_execute:begin

irq[0]=interrupt;

ir0=ir1;

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(int\_en && irq!=0) begin

if(irq[3] && int\_mask[3]) next\_st=65;

else if (irq[2] && int\_mask[2]) next\_st=66;

else if (irq[1] && int\_mask[1]) next\_st=67;

else if (irq[0] && int\_mask[0]) next\_st=69;

end

else begin

case (ir0[15:12])

0:begin

if(ir0[3:1]==0) begin

next\_st=op\_begin;

code=3'b000;

end

else begin

next\_st=1;

irq[3]=1'b1;

end

end

1:begin

if(ir0[3:1]==0) begin

next\_st=op\_begin;

code=3'b001;

end

else begin

next\_st=1;

irq[3]=1'b1;

end

end

2:begin

if(ir0[3:1]==0) begin

next\_st=op\_begin;

code=3'b010;

end

else begin

next\_st=1;

irq[3]=1'b1;

end

end

3:begin

if(ir0[3:1]==0) begin

next\_st=op\_begin;

code=3'b100;

end

else begin

next\_st=1;

irq[3]=1'b1;

end

end

4:begin

if(ir0[3:0]==0) begin

if(ir0[4]==0) next\_st=18;

else next\_st=22;

end

else begin

next\_st=1;

irq[3]=1'b1;

end

end

5:begin

if(ir0[8]==0) begin

next\_st=36;

end

else begin

next\_st=1;

irq[3]=1'b1;

end

end

6:begin

if(ir0[8:6]==0) next\_st=30;

else begin

next\_st=1;

irq[3]=1'b1;

end

end

7:begin

if(ir0[11:8]==0) next\_st=37;

else begin

next\_st=1;

irq[3]=1'b1;

end

end

8:begin

if(ir0[11:8]==0) next\_st=38;

else begin

next\_st=1;

irq[3]=1'b1;

end

end

9:begin

if(ir0[11:0]==0) next\_st=40;

else begin

next\_st=1;

irq[3]=1'b1;

end

end

10:begin

if(ir0[8:3]==0) begin

if(ir0[11:9]!=ir0[2:0]) next\_st=41;

else begin

next\_st=1;

irq[3]=1'b1;

end

end else begin

next\_st=1;

irq[3]=1'b1;

end

end

11:begin

if(ir0[8]==0) next\_st=58;

else next\_st=62;

end

12:begin

if(ir0[8]==0) next\_st=42;

else next\_st=45;

end

13:begin

if(ir0[11:8]==0) next\_st=46;

else begin

next\_st=1;

irq[3]=1'b1;

end

end

14:begin

if(ir0[11:8]==0) next\_st=50;

else begin

next\_st=1;

irq[3]=1'b1;

end

end

15:begin

if(ir0[11:5]==0) next\_st=54;

else begin

next\_st=1;

irq[3]=1'b1;

end

end

default: begin

next\_st=1;

irq[3]=1'b1;

end

endcase

end

end

//pc out, MAR in

op\_begin:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_in=0;

ready\_out=0;

next\_st=100;

end

//pc+1, MDR in, RAM out

100:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b101; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b100;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(mem\_done && ir0[0]==0) next\_st=5;

else if(mem\_done) next\_st=15;

else next\_st=100;

end

//ALU

//ALU\_reg in, pc->mar, MDR out

5:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b1; //ALU\_reg

con[8:6]=3'b010; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(ir0[4]==0) next\_st=6;

else next\_st=9;

end

//RA out, ALU\_out in, PC+1, RAM\_out, ir1[15:8]

6:begin

con[2:0]=code; //alu

con[4:3]=2'b01; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b100; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b100;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

con\_reg\_assign(ir0[7:5],r\_asrt);

bus\_out='hz;

if(mem\_done) begin

if(ir0[8]==0) next\_st=7;

else next\_st=12;

ir1[15:8]=mem\_in;

condition={V, Z, C};

if(V==1) irq[2]=1'b1;

if(Z==1) irq[1]=1'b1;

end

else next\_st=6;

end

//ALU\_out out, RD in, pc->mar

7:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b10; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

con\_reg\_assign(ir0[11:9],r\_ld);

next\_st=8;

bus\_out='hz;

end

//PC+1, RAM out, ir1[7:0]

8:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b100; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b100;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(mem\_done) begin

next\_st=begin\_execute;

ir1[7:0]=mem\_in;

end

else next\_st=8;

end

//RA out, MAR in

9:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b01; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

con\_reg\_assign(ir0[7:5],r\_asrt);

next\_st=10;

end

//MDR in, RAM out

10:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b101; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(mem\_done) next\_st=101;

else next\_st=10;

end

//pc->mar

101:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b101; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=11;

end

//MDR out, ALU\_out in, pc+1, RAM out, ir1[15:8]

11:begin

con[2:0]=code; //alu

con[4:3]=2'b01; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b110; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b100;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(mem\_done) begin

if(ir0[8]==0) next\_st=7;

else next\_st=12;

ir1[15:8]=mem\_in;

condition={V, Z, C};

if(V==1) irq[2]=1'b1;

if(Z==1) irq[1]=1'b1;

end

else next\_st=11;

end

//RD out, MAR in

12:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b01; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

con\_reg\_assign(ir0[11:9],r\_asrt);

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=13;

end

//MDR in, ALU\_out out

13:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b10; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b001; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=14;

end

//MDR out, RAM in

14:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b1; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_in=0;

ready\_out=0;

if(mem\_done) next\_st=102;

else next\_st=14;

end

//pc->mar

102:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=8;

end

//MDR out, MAR in

15:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b010; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b01; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=16;

end

//MDR in, RAM out

16:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b101; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(mem\_done) next\_st=17;

else next\_st=16;

end

//MDR out, ALU\_reg in, pc->mar

17:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b1; //ALU\_reg

con[8:6]=3'b010; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(ir0[4]==0) next\_st=6;

else next\_st=9;

end

//RD=~RA start

//RA out, ALU\_out in, pc->mar

18:begin

con[2:0]=3'b110; //alu

con[4:3]=2'b01; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

con\_reg\_assign(ir0[7:5],r\_asrt);

if(ir0[8]==0) next\_st=19;

else next\_st=25;

end

//ALU\_out out, RD in, pc+1, RAM out, ir1[15:8]

19:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b10; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b100; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b100;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_in=0;

ready\_out=0;

con\_reg\_assign(ir0[11:9],r\_ld);

if(mem\_done) begin

ir1[15:8]=mem\_in;

next\_st=20;

end

else next\_st=19;

end

//pc->mar

20:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=21;

end

//pc+1, RAM out, ir1[7:0]

21:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b100; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b100;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(mem\_done) begin

ir1[7:0]=mem\_in;

next\_st=begin\_execute;

end

else next\_st=21;

end

//RD=~M[RA] begin

//RA out, MAR in

22:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b01; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

con\_reg\_assign(ir0[7:5],r\_asrt);

next\_st=23;

end

//MDR in, RAM out

23:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b101; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(mem\_done) begin

next\_st=24;

end

else next\_st=23;

end

//MDR out, ALU\_out in, ALU control, pc->mar

24:begin

con[2:0]=3'b110; //alu

con[4:3]=2'b01; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b010; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(ir0[8]==0) begin

next\_st=19;

end

else next\_st=25;

end

//RD out, MAR in

25:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b01; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_in=0;

ready\_out=0;

con\_reg\_assign(ir0[11:9],r\_asrt);

next\_st=26;

end

//MDR in, ALU\_out out

26:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b10; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b001; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=27;

end

//RAM in, MDR out

27:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b1; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(mem\_done) next\_st=28;

else next\_st=27;

end

//pc->mar

28:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=29;

end

//pc+1, RAM out, ir1[15:8]

29:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b100; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b100;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(mem\_done) begin

next\_st=20;

ir1[15:8]=mem\_in;

end else next\_st=29;

end

//BRANCH begin

//RB out, ALU\_reg in, pc->mar

30:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b1; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

con\_reg\_assign(ir0[2:0],r\_asrt);

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=31;

end

//RA out, pc+1, RAM out, MDR in

31:begin

con[2:0]=3'b001; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b101; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b100;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

con\_reg\_assign(ir0[5:3],r\_asrt);

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(mem\_done) begin

condition={V, Z, C};

if(V==1) irq[2]=1'b1;

if(Z==1) irq[1]=1'b1;

next\_st=32;

end

else next\_st=31;

end

32:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[9]=1'b0; //RAM

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_in=0;

ready\_out=0;

next\_st=33;

case (ir0[11:9])

0:begin

//branch, MDR out, MAR in, PC in

if (condition[0]==0) begin

con[8:6]=3'b010; //MDR

con[11:10]=2'b01; //MAR

con[14:12]=3'b001;//PC

end else begin //don't branch, pc->mar

con[8:6]=3'b000; //MDR

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

end

end

1:begin

if (condition[0]==0 || condition[1]==1) begin

con[8:6]=3'b010; //MDR

con[11:10]=2'b01; //MAR

con[14:12]=3'b001;//PC

end else begin //don't branch, pc->mar

con[8:6]=3'b000; //MDR

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

end

end

2:begin

if (condition[0]==1) begin

con[8:6]=3'b010; //MDR

con[11:10]=2'b01; //MAR

con[14:12]=3'b001;//PC

end else begin //don't branch, pc->mar

con[8:6]=3'b000; //MDR

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

end

end

3:begin

if (condition[0]==1 || condition[1]==1) begin

con[8:6]=3'b010; //MDR

con[11:10]=2'b01; //MAR

con[14:12]=3'b001;//PC

end else begin //don't branch, pc->mar

con[8:6]=3'b000; //MDR

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

end

end

4:begin

if (condition[1]==1) begin

con[8:6]=3'b010; //MDR

con[11:10]=2'b01; //MAR

con[14:12]=3'b001;//PC

end else begin //don't branch, pc->mar

con[8:6]=3'b000; //MDR

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

end

end

5:begin

if (condition[1]==0) begin

con[8:6]=3'b010; //MDR

con[11:10]=2'b01; //MAR

con[14:12]=3'b001;//PC

end else begin //don't branch, pc->mar

con[8:6]=3'b000; //MDR

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

end

end

default:begin //don't branch, pc->mar

con[8:6]=3'b000; //MDR

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

end

endcase

end

//pc+1, RAM out, ir1[15:8]

33:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b100; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b100;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_in=0;

ready\_out=0;

if(mem\_done) begin

next\_st=34;

ir1[15:8]=mem\_in;

end else next\_st=33;

end

//pc->mar

34:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=35;

end

//pc+1, RAM out, ir1[7:0]

35:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b100; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b100;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(mem\_done) begin

ir1[7:0]=mem\_in;

next\_st=begin\_execute;

end

else next\_st=35;

end

//SHIFT begin

//pc->mar, shift 1

36:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(ir0[3]==1) begin

con\_reg\_assign(ir0[11:9],r\_clr);

next\_st=33;

end else if (count <= ir0[2:0]) begin

con\_reg\_assign(ir0[11:9],{1'b1,ir0[5:4]});

next\_st=36;

end else begin

next\_st=33;

end

end

//JMP begin

//ir0[7:0] out, MAR in, PC in

37:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b01; //MAR

con[14:12]=3'b001;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out=ir0[7:0];

next\_st=33;

end

//JSR begin

//PC out, r7 in

38:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b010;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

con\_reg\_assign(r7,r\_ld);

bus\_out='hz;

next\_st=39;

end

//ir0[7:0] out, PC in, MAR in

39:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b01; //MAR

con[14:12]=3'b001;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out=ir0[7:0];

next\_st=33;

end

//RTS begin

//r7 out, PC in, MAR in

40:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b01; //MAR

con[14:12]=3'b001;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

con\_reg\_assign(r7,r\_asrt);

bus\_out='hz;

next\_st=33;

end

//MV begin

//RD in, RA out, pc->mar

41:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

con\_reg\_assign(ir0[11:9],r\_ld);

con\_reg\_assign(ir0[2:0],r\_asrt);

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=33;

end

//STORE begin

//ir0[7:0] out, MAR in

42:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b01; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out=ir0[7:0];

ready\_in=0;

ready\_out=0;

next\_st=43;

end

//RD out, MDR in

43:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b001; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

con\_reg\_assign(ir0[11:9],r\_asrt);

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=44;

end

//MDR out, RAM in

44:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b1; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(mem\_done) next\_st=1;

else next\_st=44;

end

//RA out, MAR in

45:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b01; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

con\_reg\_assign(ir0[2:0],r\_asrt);

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=43;

end

//INPUT begin

//ir0[7:0] out, MAR in

46:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b01; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out=ir0[7:0];

ready\_in=0;

ready\_out=0;

next\_st=47;

end

//Set ready\_in, wait data\_in

47:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_in=1;

ready\_out=0;

if(data\_in) next\_st=48;

else next\_st=47;

end

//RAM in

48:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b1; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_in=1;

ready\_out=0;

if(mem\_done) begin

next\_st=49;

ready\_in=0;

end

else next\_st=48;

end

//wait for data\_in == 0

49:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_out=0;

ready\_in=0;

if(data\_in==0) next\_st=33;

else next\_st=49;

end

//OUTPUT begin

//ir0[7:0] out, MAR in

50:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b01; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out=ir0[7:0];

ready\_in=0;

ready\_out=0;

next\_st=51;

end

//Set ready\_out, wait data\_out

51:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_out=1;

ready\_in=0;

if(data\_out) next\_st=52;

else next\_st=51;

end

//RAM in

52:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b100; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_out=1;

ready\_in=0;

if(mem\_done) begin

next\_st=53;

end

else next\_st=52;

end

//wait for data\_out == 0

53:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_in=0;

ready\_out=0;

if(data\_out==0) next\_st=33;

else next\_st=53;

end

//HVPI enable begin

//HVPI in, pc->mar

54:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_in=0;

ready\_out=0;

next\_st=33;

if(ir0[4]==1) begin

int\_en=1;

int\_mask=ir0[3:0];

irq=0;

end

else begin

int\_en=0;

end

end

//LOAD begin

//ir0[7:0] out, RD in, pc->mar

58:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

con\_reg\_assign(ir0[11:9],r\_ld);

bus\_out=ir0[7:0];

ready\_in=0;

ready\_out=0;

next\_st=59;

end

//RAM out, pc+1, ir1[15:8]

59:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b100; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b100;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_in=0;

ready\_out=0;

if(mem\_done) begin

next\_st=60;

ir1[15:8]=mem\_in;

end

else next\_st=59;

end

//pc->mar

60:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_in=0;

ready\_out=0;

next\_st=61;

end

//pc+1, RAM out, ir1[7:0]

61:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b100; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b100;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(mem\_done) begin

next\_st=begin\_execute;

ir1[7:0]=mem\_in;

end

else next\_st=61;

end

//ir0[7:0] out, MAR in

62:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b01; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out=ir0[7:0];

next\_st=63;

end

//RAM out, MDR in

63:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b101; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

if(mem\_done) next\_st=64;

else next\_st=63;

end

//MDR out, RD in, pc->mar

64:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b010; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

con\_reg\_assign(ir0[11:9],r\_ld);

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=59;

end

//Interrupt Handler

65:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=0;

illegal\_ops=illegal\_ops+1;

irq[3]=1'b0;

if(illegal\_ops==1) next\_st=66;

else next\_st=1;

end

//stall machine

66:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=66;

end

//zero interrupt, add 1 to result

67:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b10; //ALU\_out

con[5]=1'b1; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

irq[1]=1'b0;

condition[1]=1'b0;

next\_st=68;

end

68:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b01; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out=8'b00000001;

next\_st=begin\_execute;

end

//outside interrupt, input to ram

69:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b01; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out=8'b00011111;

ready\_in=0;

ready\_out=0;

next\_st=70;

end

70:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_in=1;

ready\_out=0;

if(data\_in) next\_st=71;

else next\_st=70;

end

71:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b1; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_in=1;

ready\_out=0;

if(mem\_done) begin

next\_st=72;

ready\_in=0;

end

else next\_st=71;

end

72:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b10; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

bus\_out='hz;

ready\_out=0;

ready\_in=0;

irq[0]=1'b0;

if(data\_in==0) next\_st=begin\_execute;

else next\_st=72;

end

default:begin

con[2:0]=3'b000; //alu

con[4:3]=2'b00; //ALU\_out

con[5]=1'b0; //ALU\_reg

con[8:6]=3'b000; //MDR

con[9]=1'b0; //RAM

con[11:10]=2'b00; //MAR

con[14:12]=3'b000;//PC

con[17:15]=r\_str;

con[20:18]=r\_str;

con[23:21]=r\_str;

con[26:24]=r\_str;

con[29:27]=r\_str;

con[32:30]=r\_str;

con[35:33]=r\_str;

con[38:36]=r\_str;

ready\_in=0;

ready\_out=0;

bus\_out='hz;

next\_st=0;

end

endcase

end

always @ (posedge clk)

begin

if(reset) pres\_st=0;

else pres\_st = next\_st;

if (pres\_st==36) count=count+1;

else count=0;

end

task con\_reg\_assign;

input [3:0] register;

input [2:0] fnc;

begin

case (register)

r0:con[17:15]=fnc;

r1:con[20:18]=fnc;

r2:con[23:21]=fnc;

r3:con[26:24]=fnc;

r4:con[29:27]=fnc;

r5:con[32:30]=fnc;

r6:con[35:33]=fnc;

r7:con[38:36]=fnc;

endcase

end

endtask

endmodule

`timescale 1ns / 1ps

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

// Company: University of Kentucky

// Engineer: Zaid Mullins

//

// Create Date: 16:31:29 04/23/2012

// Module Name: cache.v

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

module cache(in, addr, con, clk, mem\_done, out, mem\_in, mem\_out, source, reset);

parameter m=8;

input [m-1:0] in, mem\_in;

input [m-1:0] addr;

input [1:0] con;

input clk, source, reset;

output [m-1:0] mem\_out;

output reg [m-1:0] out;

output reg mem\_done;

wire [m-1:0] s\_in;

integer i, test=0, num=-1;

reg [3:0] pres\_st;

reg [3:0] next\_st;

reg [7:0] addr\_reg, ram\_in, ram\_addr;

reg hit;

reg [2:0] cache\_addr;

wire ram\_done;

reg [1:0] ram\_con;

wire [7:0] ram\_out;

reg [m-1:0] cache\_array [7:0];

reg [m-1:0] address\_array [7:0];

reg [m-1:0] count [7:0];

assign s\_in = (source)?mem\_in:in;

assign mem\_out = out;

//RAM(in, addr, con, clk, mem\_done, out);

RAM memory(ram\_in, ram\_addr, ram\_con, clk, ram\_done, ram\_out);

always @ (con, mem\_in, source, pres\_st, hit, ram\_done)

begin

case (pres\_st)

0:begin

mem\_done=0;

hit=0;

cache\_addr=0;

ram\_con=0;

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

cache\_array[i]=0;

address\_array[i]=127;

count[i]=0;

end

if (con==2'b00) next\_st=1;

else if (con==2'b01) next\_st=2;

else if (con==2'b10) next\_st=7;

else if (con==2'b11) next\_st=1;

end

1:begin

mem\_done=0;

hit=0;

cache\_addr=0;

ram\_con=0;

if (con==2'b01) next\_st=2;

else if (con==2'b10) next\_st=7;

else next\_st=1;

end

2:begin

ram\_con=0;

mem\_done=0;

hit=0;

cache\_addr=0;

for(i=0;(i<m) && !hit;i=i+1) begin

if (address\_array[i]==addr\_reg) begin

hit=1;

cache\_addr=i;

end

end

if (hit) next\_st=3;

else next\_st=4;

end

3:begin

ram\_con=0;

mem\_done=1;

hit=0;

cache\_addr=cache\_addr;

out=cache\_array[cache\_addr];

count[cache\_addr]=count[cache\_addr]+1'b1;

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

if (count[i]!=0 && i!=cache\_addr) count[i]=count[i]-1;

end

next\_st=1;

end

4:begin

ram\_con=1;

mem\_done=0;

hit=0;

cache\_addr=0;

ram\_addr=addr\_reg;

if(ram\_done) next\_st=5;

else next\_st=4;

end

5:begin

ram\_con=1;

mem\_done=1;

hit=0;

cache\_addr=0;

out=ram\_out;

ram\_addr=addr\_reg;

if(num<8) begin

cache\_addr=num;

end else begin

test=count[0];

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

if (test>count[i]) begin

test=count[i];

cache\_addr=i;

end

end

end

next\_st=6;

end

6:begin

ram\_con=1;

mem\_done=0;

hit=0;

ram\_addr=addr\_reg;

cache\_addr=cache\_addr;

cache\_array[cache\_addr]=ram\_out;

address\_array[cache\_addr]=addr\_reg;

next\_st=1;

end

7:begin

ram\_con=0;

mem\_done=0;

hit=0;

cache\_addr=0;

for(i=0;(i<m) && !hit;i=i+1) begin

if (address\_array[i]==addr\_reg) begin

hit=1;

cache\_addr=i;

end

end

if (hit) next\_st=8;

else if(num<8) begin

cache\_addr=num;

next\_st=9;

end else begin

test=count[0];

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

if (test>count[i]) begin

test=count[i];

cache\_addr=i;

end

end

next\_st=10;

end

end

8:begin

ram\_con=0;

mem\_done=1;

hit=0;

cache\_addr=cache\_addr;

cache\_array[cache\_addr]=s\_in;

count[cache\_addr]=count[cache\_addr]+1'b1;

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

if (count[i]!=0 && i!=cache\_addr) count[i]=count[i]-1;

end

next\_st=1;

end

9:begin

ram\_con=0;

mem\_done=1;

hit=0;

cache\_addr=cache\_addr;

count[cache\_addr]=count[cache\_addr]+1'b1;

cache\_array[cache\_addr]=s\_in;

address\_array[cache\_addr]=addr\_reg;

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

if (count[i]!=0 && i!=cache\_addr) count[i]=count[i]-1;

end

next\_st=1;

end

10:begin

ram\_con=2;

mem\_done=0;

cache\_addr=cache\_addr;

ram\_in=cache\_array[cache\_addr];

ram\_addr=address\_array[cache\_addr];

if(ram\_done) next\_st=11;

else next\_st=10;

end

11:begin

ram\_con=0;

mem\_done=1;

cache\_addr=cache\_addr;

count[cache\_addr]=count[cache\_addr]+1'b1;

cache\_array[cache\_addr]=s\_in;

address\_array[cache\_addr]=addr\_reg;

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

if (count[i]!=0 && i!=cache\_addr) count[i]=count[i]-1;

end

next\_st=1;

end

default:begin

next\_st=0;

end

endcase

end

always @ (posedge clk)

begin

if(reset || con==2'b11) pres\_st=0;

else pres\_st=next\_st;

if (pres\_st==1) addr\_reg=addr;

if ((pres\_st==5 || pres\_st==7) && num<8) num=num+1;

end

endmodule

`timescale 1ns / 1ps

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

// Company: University of Kentucky

// Engineer: Zaid Mullins and Caroline Jones

//

// Create Date: 22:28:34 04/21/2012

// Module Name: processor\_test.v

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

module processor\_test;

reg reset, data\_in, data\_out, interrupt;

reg clk;

reg [7:0] mem\_in;

wire [7:0] bus;

wire [15:0] ir0;

wire [15:0] ir1;

wire [7:0] pres\_st;

wire [7:0] next\_st, mem\_out;

wire [3:0] irq;

wire ready\_in, ready\_out;

processor uut (

.bus(bus),

.reset(reset),

.clk(clk),

.ir0(ir0),

.ir1(ir1),

.pres\_st(pres\_st),

.next\_st(next\_st),

.ready\_in(ready\_in),

.ready\_out(ready\_out),

.data\_in(data\_in),

.data\_out(data\_out),

.mem\_in(mem\_in),

.mem\_out(mem\_out),

.interrupt(interrupt),

.irq(irq)

);

initial begin

reset = 0;

clk = 0;

data\_in=0;

data\_out=0;

interrupt=0;

// Wait 100 ns for global reset to finish

#100;

reset=1;

#20 reset=0;

mem\_in=-67;

data\_in=0;

data\_out=0;

//#200 interrupt=1;

// Add stimulus here

end

always begin #10 clk = !clk; end

always @ (ready\_in, ready\_out) begin

#20 data\_out = !data\_out;

#20 data\_in = !data\_in;

end

endmodule

`timescale 1ns / 1ps

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

// Company: University of Kentucky

// Engineer: Zaid Mullins and Caroline Jones

//

// Create Date: 12:36:01 04/22/2012

// Module Name: RAM.v

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

module RAM(in, addr, con, clk, mem\_done, out);

parameter n=6;

parameter m=8;

input [m-1:0] in;

input [n-1:0] addr;

input [1:0] con;

input clk;

output reg [m-1:0] out;

output reg mem\_done;

integer i, j=9;

reg [m-1:0] ram\_array [((2\*\*n)-1):0];

initial begin

//LOAD r0 6

ram\_array[0]<=8'b10110000;

ram\_array[1]<=8'b00000110;

//ADD r1 r0 15

ram\_array[2]<=8'b00000010;

ram\_array[3]<=8'b00000000;

ram\_array[4]<=8'b00001111;

//LOAD r2 M[49]

ram\_array[5]<=8'b10110101;

ram\_array[6]<=8'b00110001;

//AND r3 M[r2] M[50]

ram\_array[7]<=8'b00100110;

ram\_array[8]<=8'b01010001;

ram\_array[9]<=8'b00110010;

//OR r4 M[r2] -17

ram\_array[10]<=8'b00111000;

ram\_array[11]<=8'b01010000;

ram\_array[12]<=8'b11101111;

//LOAD r5 50

ram\_array[13]<=8'b10111010;

ram\_array[14]<=8'b00110010;

//SUB M[r5] r0 -37

ram\_array[15]<=8'b00011011;

ram\_array[16]<=8'b00000000;

ram\_array[17]<=8'b11011011;

//COMP r6 r3

ram\_array[18]<=8'b01001100;

ram\_array[19]<=8'b01100000;

//COMP r6 M[r5]

ram\_array[20]<=8'b01001100;

ram\_array[21]<=8'b10110000;

//COMP M[r5] r3

ram\_array[22]<=8'b01001011;

ram\_array[23]<=8'b01100000;

//COMP M[r5] M[r2]

ram\_array[24]<=8'b01001011;

ram\_array[25]<=8'b01010000;

//SLL(2)r1

ram\_array[26]<=8'b01010010;

ram\_array[27]<=8'b00000010;

//SLR(4)r0

ram\_array[28]<=8'b01010000;

ram\_array[29]<=8'b00010100;

//SAR(1)r3

ram\_array[30]<=8'b01010110;

ram\_array[31]<=8'b00100001;

//SAL(3)r5

ram\_array[32]<=8'b01011010;

ram\_array[33]<=8'b00110011;

//MV r0 r1

ram\_array[34]<=8'b10100000;

ram\_array[35]<=8'b00000001;

//STORE r3 0

ram\_array[36]<=8'b11000110;

ram\_array[37]<=8'b00000000;

//LOAD r1 1

ram\_array[38]<=8'b10110010;

ram\_array[39]<=8'b00000001;

//STORE r0 M[r1]

ram\_array[40]<=8'b11001011;

ram\_array[41]<=8'b00000001;

//INPUT 2

ram\_array[42]<=8'b11010000;

ram\_array[43]<=8'b00000010;

//OUTPUT 3

ram\_array[44]<=8'b11100000;

ram\_array[45]<=8'b00000001;

ram\_array[48]<=8'b00101010;

ram\_array[49]<=8'b00110000;

ram\_array[50]<=8'b00101001;

end

always @ (posedge clk)

begin

if(con==2'b11) begin

mem\_done=0;

j=9;

//for(i=0;i<2\*\*n;i=i+1) begin

// ram\_array[i]<=0;

//end

end else if (con==2'b10) begin

if (j==0) begin

j=9;

mem\_done=1;

ram\_array[addr]<=in;

end else begin

mem\_done=0;

j=j-1;

end

end else if (con==2'b01) begin

if (j==0) begin

j=9;

mem\_done=1;

out<=ram\_array[addr];

end else begin

mem\_done=0;

j=j-1;

end

end else begin

mem\_done=0;

out<='hx;

j=9;

end

end

endmodule

**Test Programs**

**Program 1:**

LOAD r0 6

ADD r1 r0 15

LOAD r2 M[49]

AND r3 M[r2] M[50]

OR r4 M[r2] -17

LOAD r5 50

SUB M[r5] r0 -37

COMP r6 r3

COMP r6 M[r5]

COMP M[r5] r3

COMP M[r5] M[r2]

SLL(2) r1

SLR(4) r0

SAR(1) r3

SAL(3) r5

MV r0 r1

STORE r3 0

LOAD r1 1

STORE r0 M[r1]

INPUT 2

OUTPUT 3

10110000

00000110

00000010

00000000

00001111

10110101

00110001

00100110

01010001

00110010

00111000

01010000

11101111

10111010

00110010

00011011

00000000

11011011

01001100

01100000

01001100

10110000

01001011

01100000

01001011

01010000

01010010

00000010

01010000

00010100

01010110

00100001

01011010

00110011

10100000

00000001

11000110

00000000

10110010

00000001

11001011

00000001

11010000

00000010

11100000

00000001

**Program 2:**

LOAD r0 6

LOAD r0 1

SUB r0 1

BGT r0 r1 2

ADD r0 1

BEQ r0 r1

ADD r0 1

SUB r0 1

BNQ r0 r1

ADD r0 1

SUB r0 1

BGE r0 r1

10110010

00000000

10110000

00000001

00010000

00000000

00000001

01100000

00000001

00000100

00000000

00000000

00000001

01101000

00000001

00001010

00000000

00000000

00000001

00010000

00000000

00000001

01101010

00000001

00010011

00000000

00000000

00000001

00010000

00000000

00000001

01100010

00000001

00011100

**Program 3:**

LOAD r0 5

JSR 1

ADD r0 10

//Subroutine 1

ADD r0 20

MV r6 r7

JSR 2

MV r7 r6

RTS

//Subroutine 2

ADD r0 15

RTS

10110000

00000101

10000000

00001010

00000000

00000000

00001010

00000000

00000000

00010100

10101100

00000111

10000000

00010110

10101110

00000110

10010000

00000000

00000000

00000000

00001111

10010000

00000000

**Program 4:**

HVPI #F

LOAD r0 10

SUB r1 r0 10

LOAD r0 100

ADD r1 r0 50

11110000

00011111

10110000

00001010

00010010

00000000

00001010

10110000

01100100

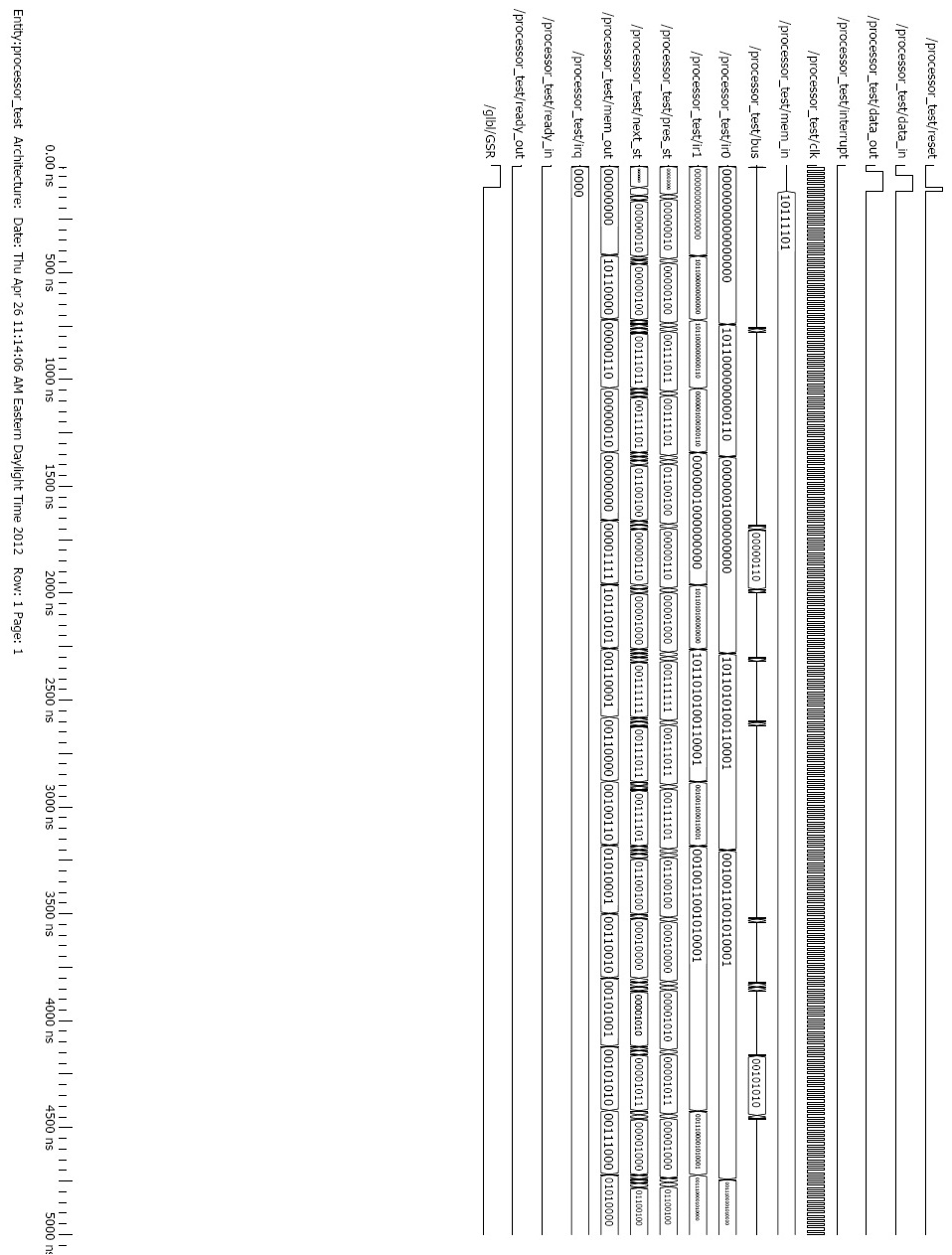
00000010

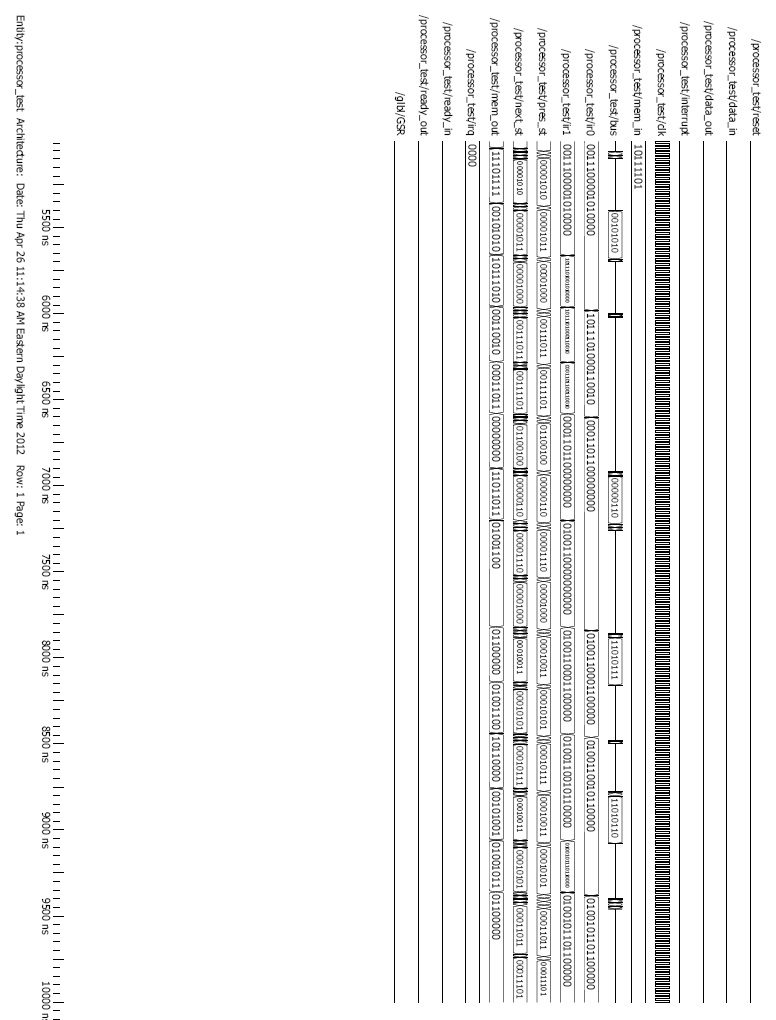
00000000

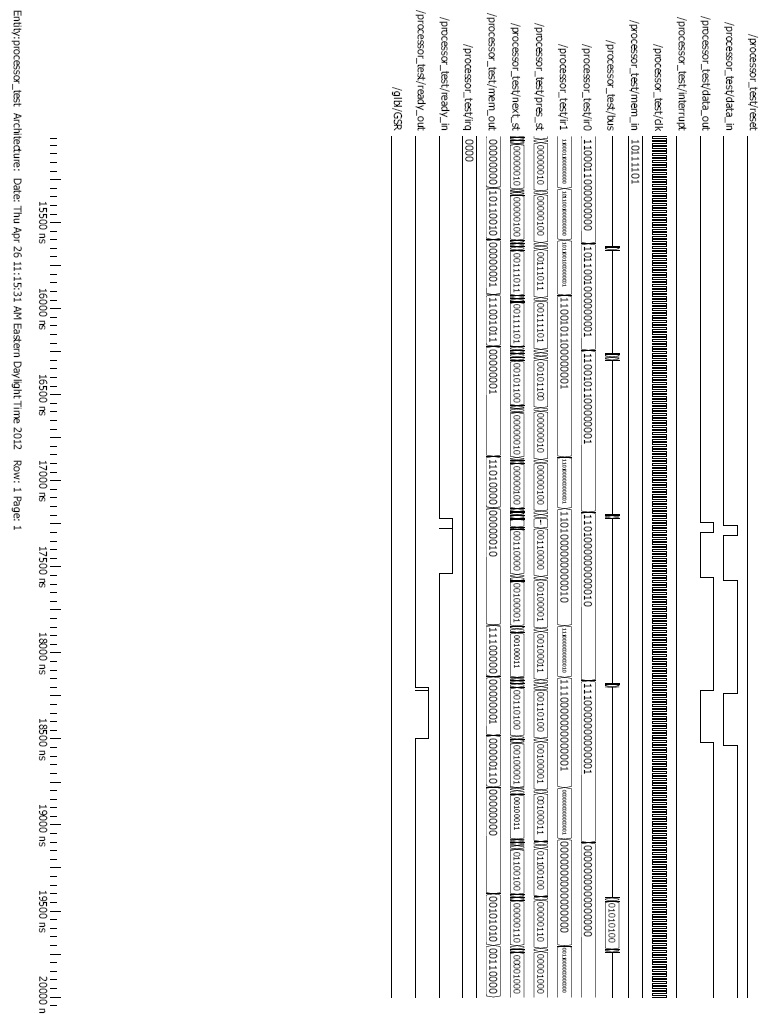
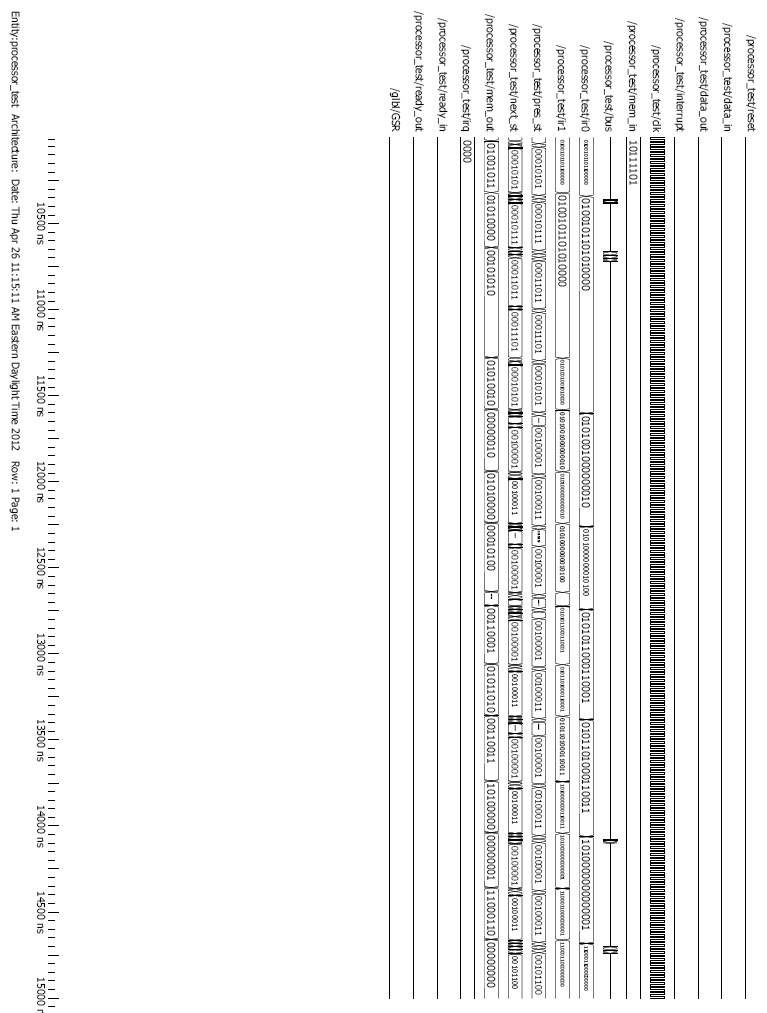
00110010

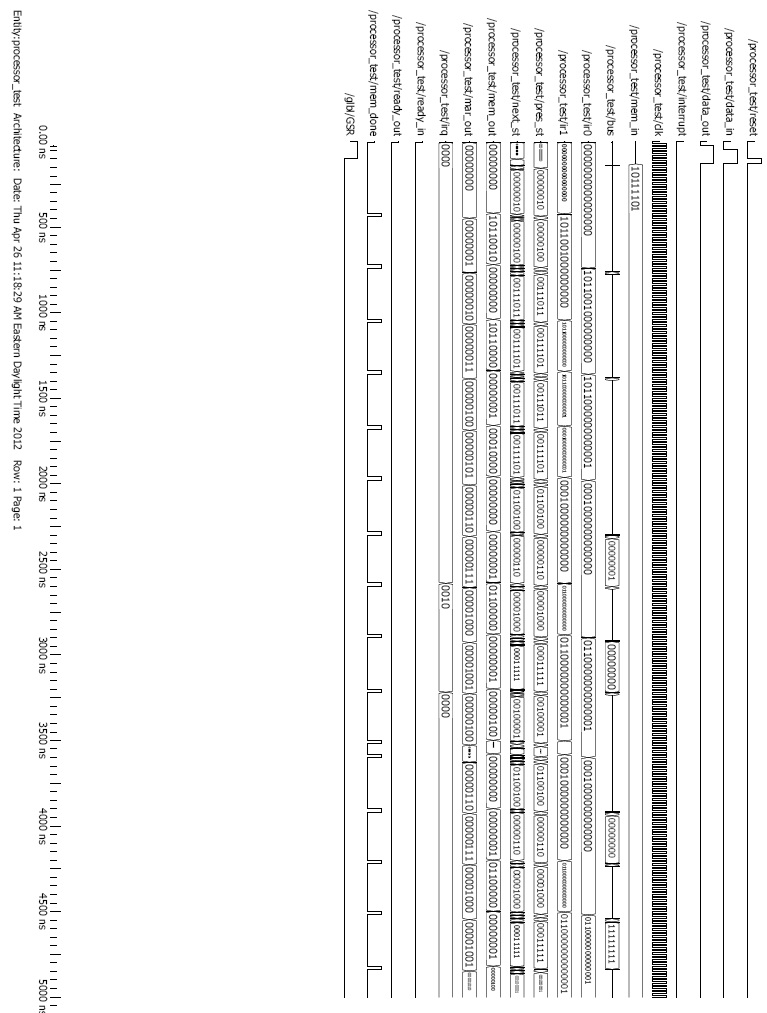
**Simulation Results**

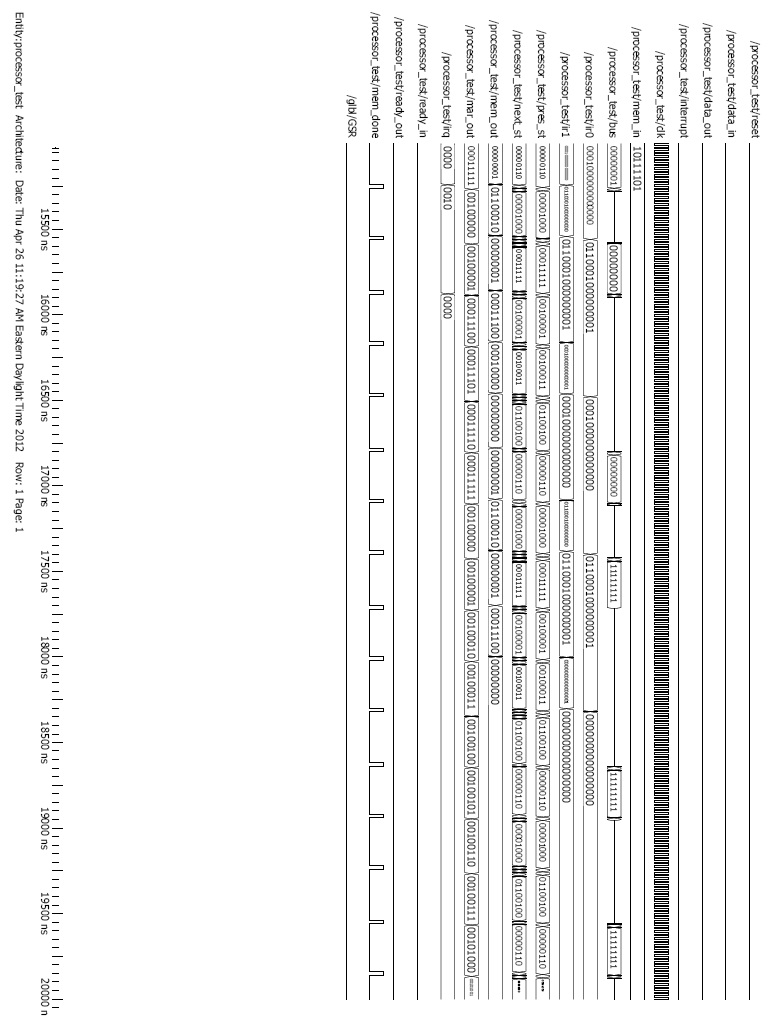
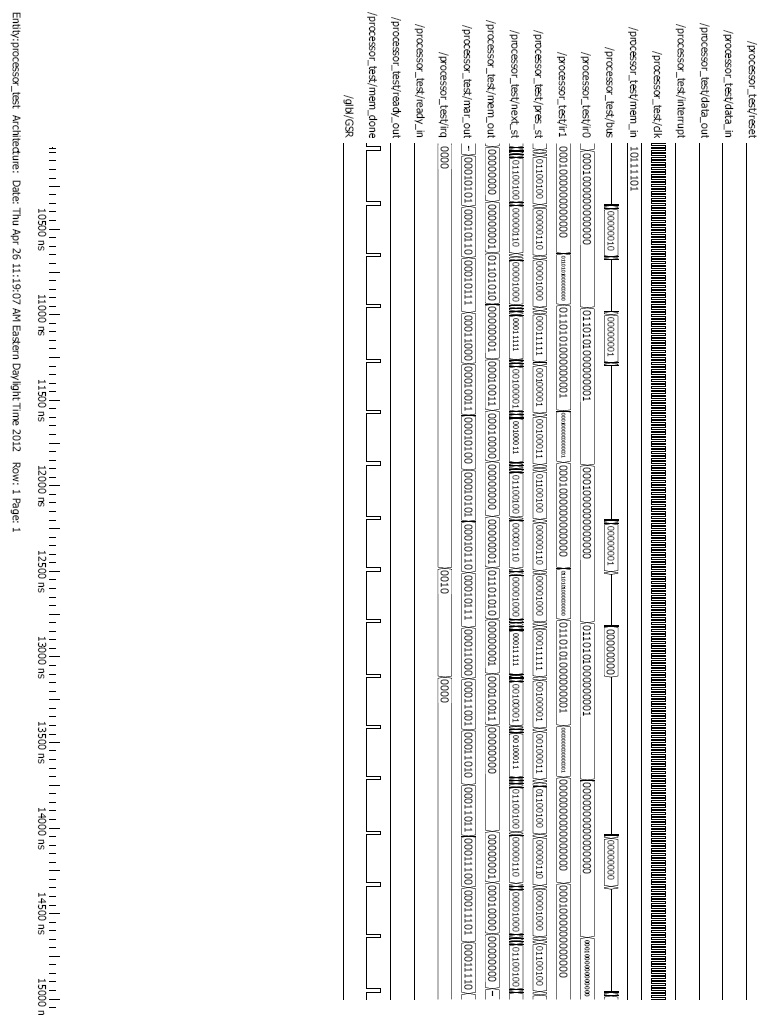
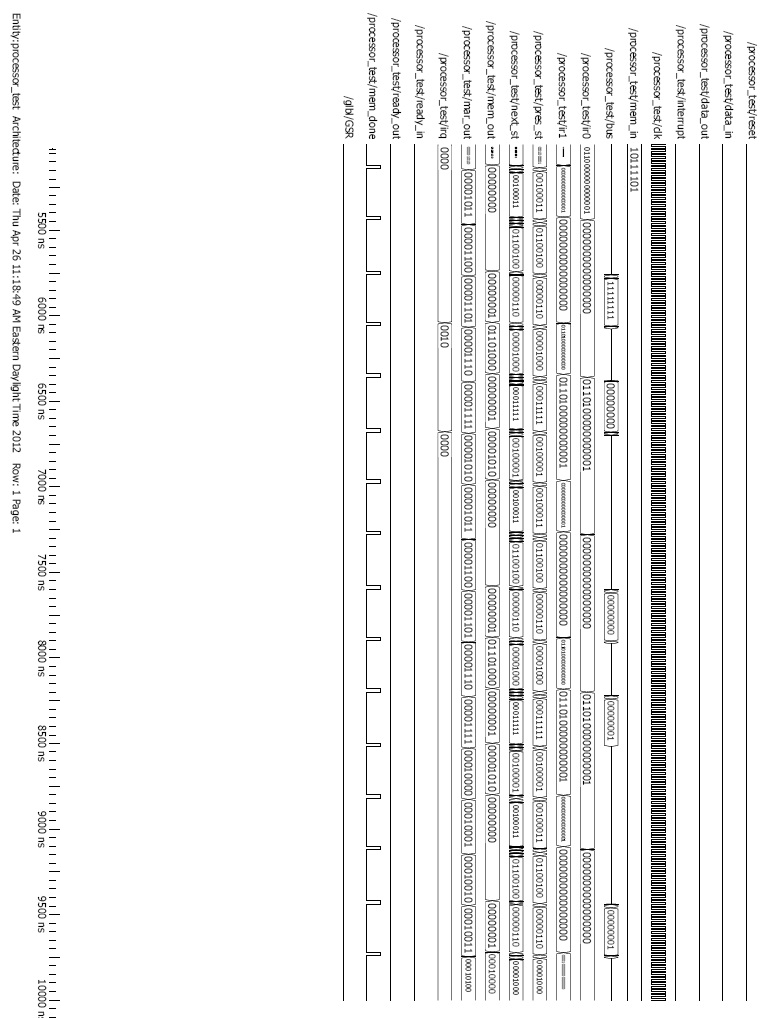
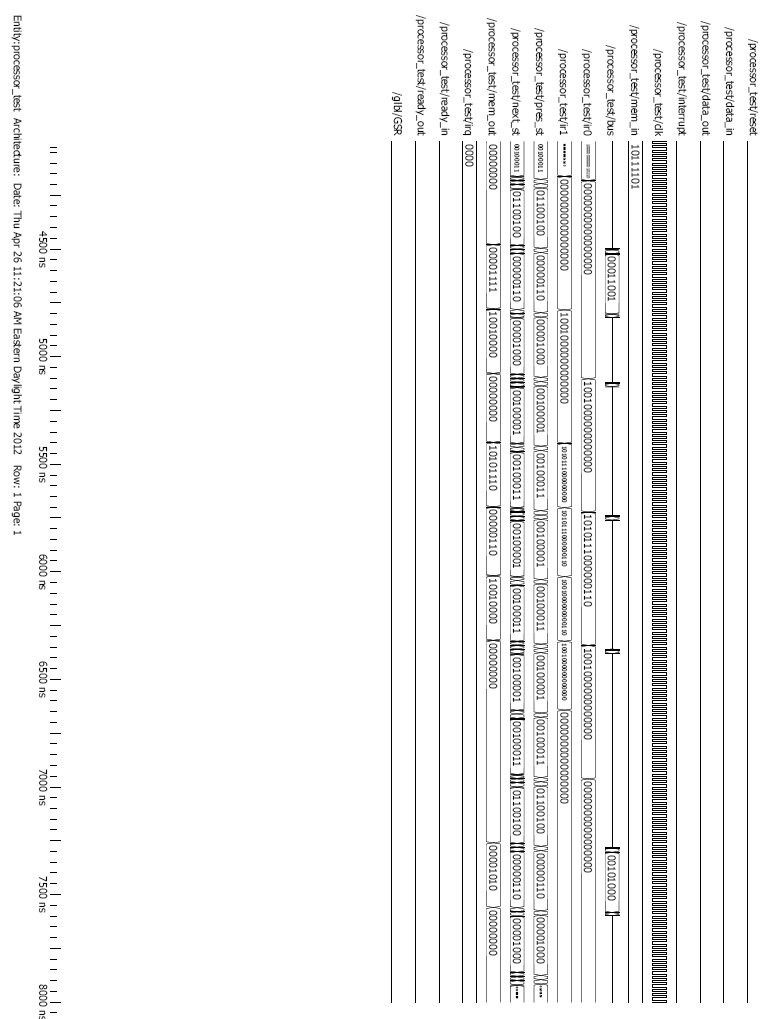
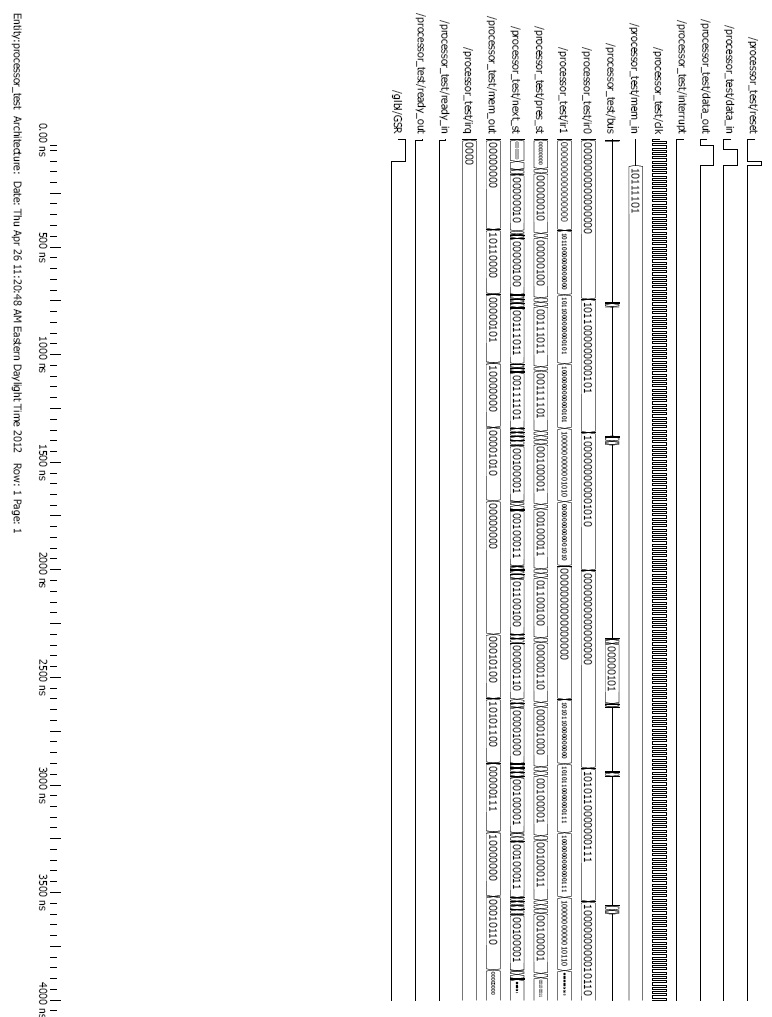
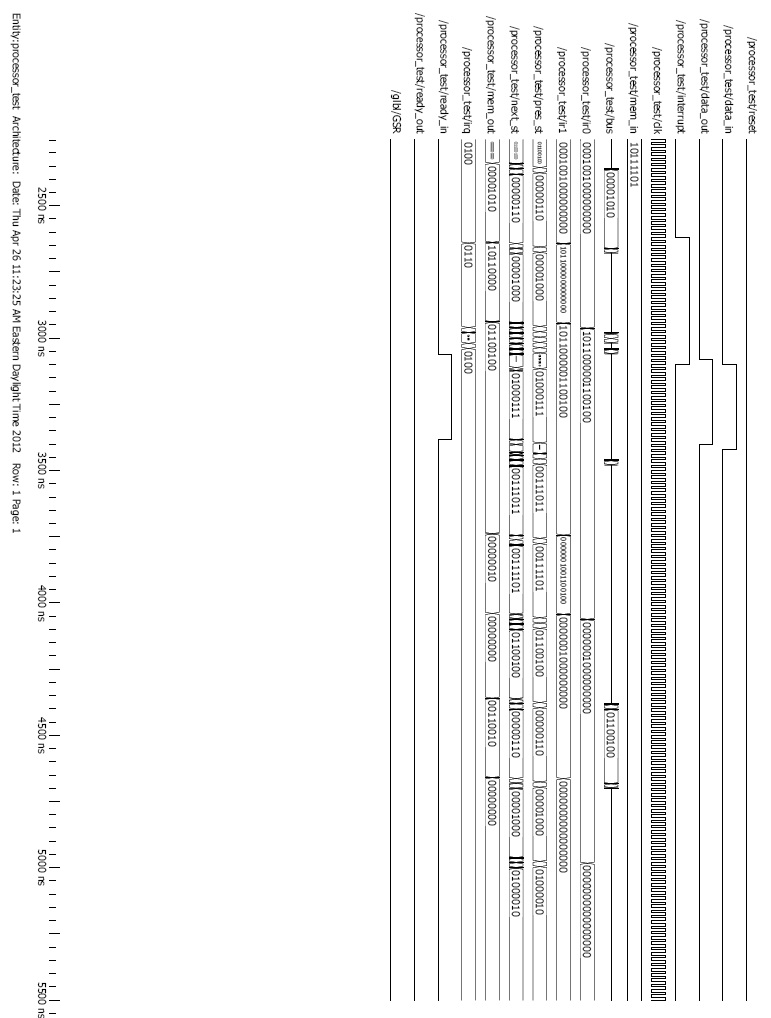
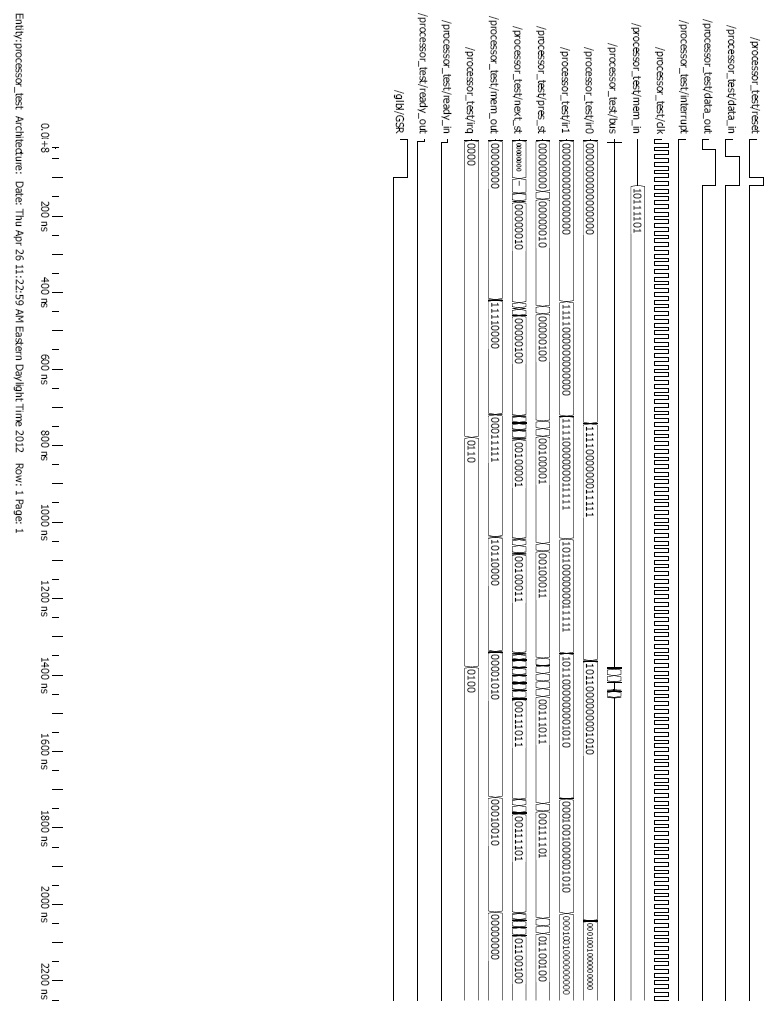
Program 1:

****

****

****Program 2:

****

****Program 3:****Program 4:****

**Conclusion**

In this project we designed an MR type ISA, developed the HDL descriptions of the hardware needed to implement that ISA, and then verified correct implementation by simulation. The processor we developed was a true single bus machine, had a direct mapped cache system, and a priority interrupt system. We were able to simulate the system “post place and route” and run each of our 4 test programs. All of the work put forth in this document was done by Zaid Mullins and Caroline Jones alone.