Name & ID: Laiba Fatima Khan Lk04067

**CA Lab 07 Report**

**RegisterFile:**

module registerFile

(

input [63:0]WriteData,

input [4:0]RS1,

input [4:0]RS2,

input [4:0]RD,

input RegWrite, clk, reset,

output reg [63:0]ReadData1,

output reg [63:0]ReadData2

);

reg[63:0] Registers [31:0];

//initialize Registers with random values (using 'initial' block)

initial

begin

Registers[0] = 64'd1;

Registers[1] = 64'd2;

Registers[2] = 64'd3;

Registers[3] = 64'd4;

Registers[4] = 64'd5;

Registers[5] = 64'd6;

Registers[6] = 64'd7;

Registers[7] = 64'd8;

Registers[8] = 64'd9;

Registers[9] = 64'd10;

Registers[10] = 64'd11;

Registers[11] = 64'd12;

Registers[12] = 64'd13;

Registers[13] = 64'd14;

Registers[14] = 64'd15;

Registers[15] = 64'd16;

Registers[16] = 64'd17;

Registers[17] = 64'd18;

Registers[18] = 64'd19;

Registers[19] = 64'd20;

Registers[20] = 64'd21;

Registers[21] = 64'd22;

Registers[22] = 64'd23;

Registers[23] = 64'd24;

Registers[24] = 64'd25;

Registers[25] = 64'd26;

Registers[26] = 64'd27;

Registers[27] = 64'd28;

Registers[28] = 64'd29;

Registers[29] = 64'd30;

Registers[30] = 64'd31;

Registers[31] = 64'd32;

end

//operation of writing data into a Register should always be done when

//positive edge of clock and RegWrite signal is asserted (or set, i.e. High)

always @ (posedge clk or RegWrite)

begin

case(RegWrite)

1'b1 : Registers[RD] <= WriteData;

endcase

end

always @ (reset or RS1 or RS2)

begin

if(reset)

begin

ReadData1 = 0;

ReadData2 = 0;

end

else

begin

ReadData1 = Registers[RS1];

ReadData2 = Registers[RS2];

end

end

endmodule

**TestBench:**

module tb

(

);

reg [63:0]WriteData;

reg [4:0]RS1;

reg [4:0]RS2;

reg [4:0]RD;

reg RegWrite, clk, reset;

wire [63:0]ReadData1;

wire [63:0]ReadData2;

registerFile regFile

(

.WriteData(WriteData),

.RS1(RS1),

.RS2(RS2),

.RD(RD),

.RegWrite(RegWrite),

.clk(clk),

.reset(reset),

.ReadData1(ReadData1),

.ReadData2(ReadData2)

);

initial

begin

clk = 1;

RegWrite = 0;

reset = 1;

#100

reset = 0;

RS1 = 10; //ReadData1 reads register 11

RS2 = 15; //ReadData2 reads register 16

WriteData = 31;

RD = 20; //register 21 set as RD to write 31

#100 RegWrite = 1;//allow write in register 21

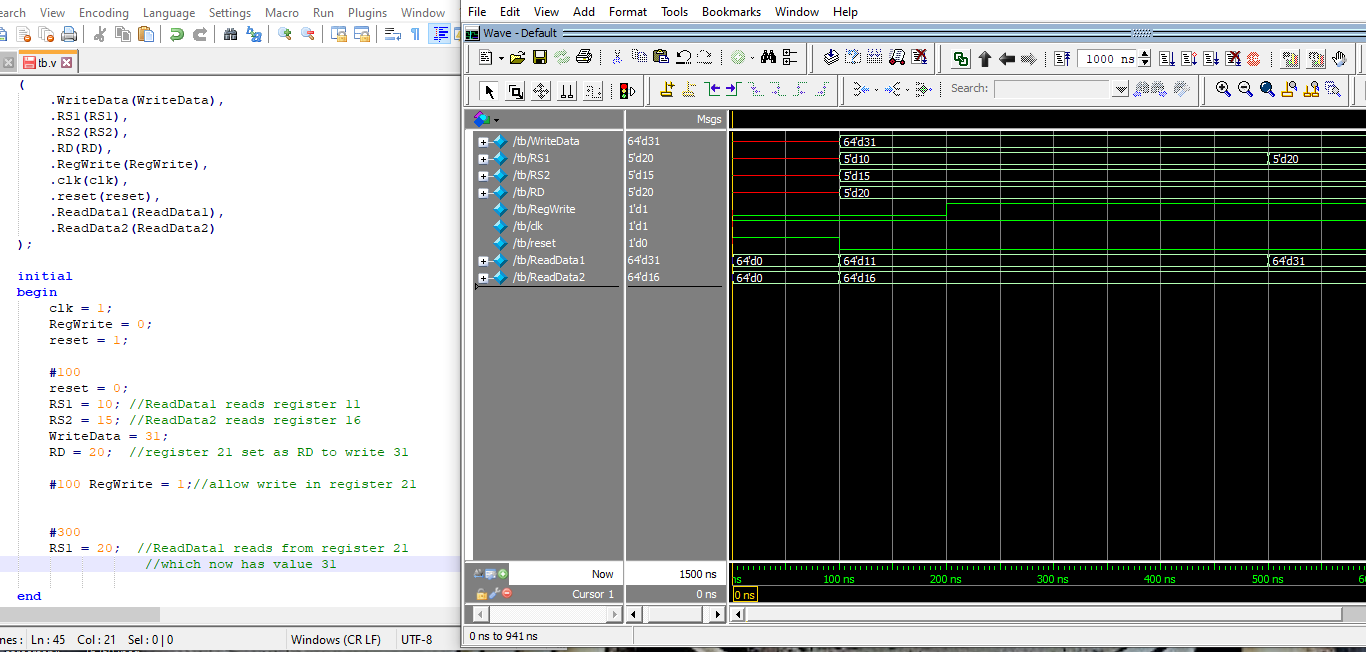
#300

RS1 = 20; //ReadData1 reads from register 21

//which now has value 31

end

endmodule



**EXERCISE:**

**Instruction Parser:**

module instruction

(

input [31:0] instruction,

output reg [6:0] opcode, funct7, [4:0] rd, rs1, rs2, [2:0] funct3

);

always @(instruction)

begin

opcode = instruction[6:0];

rd = instruction[11:7];

funct3 = instruction[14:12];

rs1 = instruction[19:15];

rs2 = instruction[24:16];

funct7 = instruction[31:25];

end

endmodule

**Top:**

module top

(

input [31:0]instruction,

output [63:0]ReadData1,

output [63:0]ReadData2

);

wire[4:0] R1;

wire[4:0] R2;

wire[4:0] RDwire;

instruction instr

(

.instruction(instruction),

.opcode(),

.funct7(),

.rd(RDwire),

.rs1(R1),

.rs2(R2),

.funct3()

);

registerFile regFi1e

(

.WriteData(100),

.RegWrite(0),

.clk(1),

.reset(1),

.RS1(R1-1),

.RS2(R2-1),

.RD(RDwire-1),

.ReadData1(ReadData1),

.ReadData2(ReadData2)

);

endmodule

**TestBench:**

module tb

(

);

reg[31:0] instruction;

wire[63:0] ReadData1;

wire[63:0] ReadData2;

top tp

(

.instruction(instruction),

.ReadData1(ReadData1),

.ReadData2(ReadData2)

);

initial

begin

instruction = 32'b00000000001000001000001010110011 //add x5, x1, x2

end

endmodule

