Name: Laiba Fatima Khan

**CA Lab 11: Single Cycle RISC Processor**

**RISCV Processor:**

module RISC\_V\_Processor

(

input clk, reset

);

wire[63:0] PC\_In\_wire;

wire[63:0] PC\_Out\_wire;

wire[31:0] Instruction\_Out\_wire;

wire [63:0] Adder1\_Out\_Wire;

wire [63:0] Adder2\_Out\_Wire;

wire [3:0] Operation\_Wire;

wire [63:0] MUX\_Data\_Out\_Wire;

wire [63:0] ALU\_Result\_Wire;

wire [63:0] Immediate\_Data\_Wire;

wire [6:0] opcode\_wire, funct7\_wire;

wire [4:0] rd\_wire, rs1\_wire, rs2\_wire;

wire [2:0] funct3\_wire;

wire [63:0]ReadData1\_wire, ReadData2\_wire;

wire Zero\_Wire;

wire [63:0] Register\_WriteData\_Wire;

wire [63:0] DataMemory\_Read\_Data;

wire Branch\_wire, MemRead\_wire, MemtoReg\_wire, MemWrite\_wire, ALUSrc\_wire, RegWrite\_wire;

wire [1:0]ALUOp\_wire;

Program\_Counter PC

(

.clk(clk),

.reset(reset),

.PC\_In(PC\_In\_wire),

.PC\_Out(PC\_Out\_wire)

);

Adder Adder1

(

.a(PC\_Out\_wire),

.b(64'd4),

.out(Adder1\_Out\_Wire)

);

Adder Adder2

(

.a(PC\_Out\_wire),

.b(Immediate\_Data\_Wire<<1),

.out(Adder2\_Out\_Wire)

);

multiplexer Mux1

(

.sel(Zero\_Wire & Branch\_wire),

.b(Adder1\_Out\_Wire),

.a(Adder2\_Out\_Wire),

.data\_out(PC\_In\_wire)

);

multiplexer Mux2

(

.sel(ALUSrc\_wire),

.b(ReadData2\_wire),

.a(Immediate\_Data\_Wire),

.data\_out(MUX\_Data\_Out\_Wire)

);

multiplexer Mux3

(

.sel(MemtoReg\_wire),

.b(ALU\_Result\_Wire),

.a(DataMemory\_Read\_Data),

.data\_out(Register\_WriteData\_Wire)

);

Instruction\_Memory InsMem

(

.Inst\_Address(PC\_Out\_wire),

.Instruction(Instruction\_Out\_wire)

);

instruction Ins

(

.instruction(Instruction\_Out\_wire),

.rs1(rs1\_wire),

.rs2(rs2\_wire),

.rd(rd\_wire),

.funct3(funct3\_wire),

.funct7(funct7\_wire),

.opcode(opcode\_wire)

);

Control\_Unit CtrlUnit

(

.Opcode(opcode\_wire),

.Branch(Branch\_wire),

.MemRead(MemRead\_wire),

.MemtoReg(MemtoReg\_wire),

.MemWrite(MemWrite\_wire),

.ALUSrc(ALUSrc\_wire),

.RegWrite(RegWrite\_wire),

.ALUOp(ALUOp\_wire)

);

registerFile RegFile

(

.RS1(rs1\_wire),

.RS2(rs2\_wire),

.RD(rd\_wire),

.ReadData1(ReadData1\_wire),

.ReadData2(ReadData2\_wire),

.WriteData(Register\_WriteData\_Wire),

.RegWrite(RegWrite\_wire),

.clk(clk),

.reset(reset)

);

immediate\_data\_extractor DataExtract

(

.instruction(Instruction\_Out\_wire),

.imm\_data(Immediate\_Data\_Wire)

);

ALU\_Control ALUCtrl

(

.ALUOp(ALUOp\_wire),

.Funct({Instruction\_Out\_wire[30], Instruction\_Out\_wire[14:12]}),

.Operation(Operation\_Wire)

);

ALU\_64\_bit ALU

(

.a(ReadData1\_wire),

.b(MUX\_Data\_Out\_Wire),

.ALUop(Operation\_Wire),

.Result(ALU\_Result\_Wire),

.ZERO(Zero\_Wire)

);

Data\_Memory DataMem

(

.Mem\_Addr(ALU\_Result\_Wire),

.Write\_Data(ReadData2\_wire),

.clk(clk),

.Mem\_Write(MemWrite\_wire),

.Mem\_Read(MemRead\_wire),

.Read\_Data(DataMemory\_Read\_Data)

);

endmodule

**EXERCISE**

**1: Testbench:**

module tb

(

);

reg clk;

reg reset;

RISC\_V\_Processor riscv

(

.clk(clk),

.reset(reset)

);

initial

begin

clk = 0;

reset = 1;

#10 reset = ~reset;

End

always

#5 clk = ~clk;

endmodule

**TEST:**

**Clk –** initial 0 then toggle ever 5ns (works)

**Reset -** high for first 10ns and low afterwards (works)

**PC Out –** increases by 4 with every new instruction (works)

**Instruction –** instruction outputs according to Instruction Memory. On PC\_In 0, instruction 0 and so on. (works)

**RD –** for first three instructions: 9 (x9), for last (sd): imm[4:0]=8 (offset 8) (works)

**RS1 –** in instructions sequence:10(x10), 21(x21), 9(x9), 10(x10) (works)

**RS2 -** in instructions sequence: 8(offset[4:0]) , 9(x9), 1 (imm[4:0], immediate value 1), 9(x9) (works)

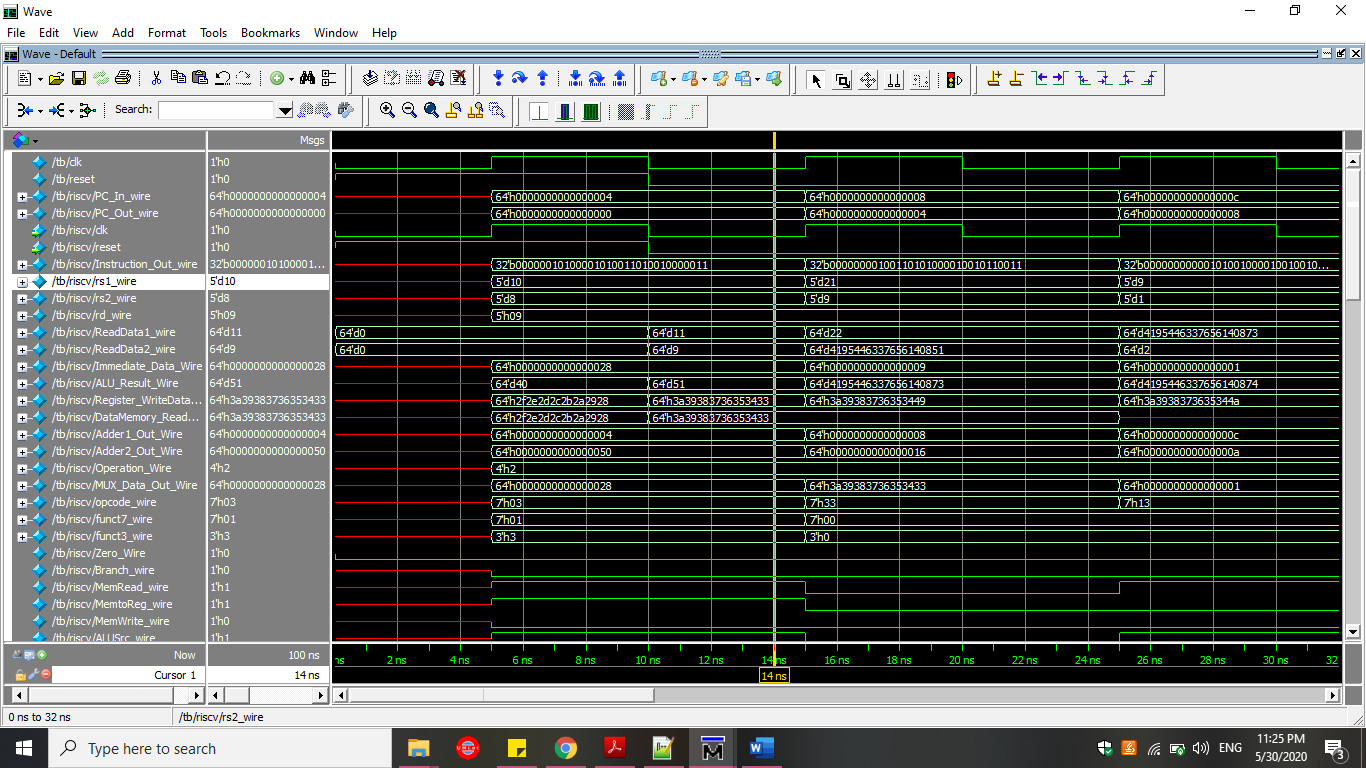
**ReadData1 –** when rs1 = x10 then ReadData1 = 11(screenshot attached), correct acc to RegisterFile (works)

**ReadData2 - –** when rs2 = x8 then ReadData1 = 9(screenshot attached), correct acc to RegisterFile (works)

**ALU Result –** 40+11(ReadData1) = 51(works)

**RegFile WriteDate -**

**DataMemory ReadData -**



**2: DECODE INSTRUCTIONS (Assembly Code)**

1. 00000010100001010011010010000011

lw x9, 40(x10)

1. 00000000100110101000010010110011

add x9, x21, x9

1. 00000000000101001000010010010011

addi x9, x9, 1

1. 00000010100101010011010000100011

sw x9, 8(x10)

**3: DECODE INSTRUCTIONS (C Code)**

1. 00000010100001010011010010000011

lw x9, 40(x10)

int A[]; //value at memory address stored in x10

// [0000000000000000000000000000000000000000000000000000000000001011]

**int h=A[10];** // 40/4, because we are considering int array

1. 00000000100110101000010010110011

add x9, x21, x9

int h; //value in x9

int temp=22; //x21=22

**h+=temp;**

1. 00000000000101001000010010010011

addi x9, x9, 1

int h; //value in x9

**h+=1;**

1. 00000010100101010011010000100011

sw x9, 8(x10)

int h; //value in x9

int A; //array at memory address stored in x10

**A[2]=h;**