Lab 11 – Worksheet

Name: Breeha Qasim and Namel Shahid	ID:	Section:

Task 1.

Provide appropriately commented code for designed module, the code should contain meaningful variable naming.
**Add snippet or Copy paste the code written in the Editor window. Make sure the irrelevant area of the snip is cropped.*

```
`timescale 1ns / 1ps

module RISC_V_Processor
(
    input clk,
    input reset,
    output wire [63:0] PC_Out,
    output wire [31:0] Instruction,
    output wire [4:0] rs1,
    output wire [4:0] rs2,
    output wire [4:0] rd,
    output wire [63:0]WriteData,
    output wire [63:0]ReadData1,
    output wire [63:0]ReadData2,
    output wire [63:0]imm_data,
    output wire [63:0] dataout,
    output wire [63:0] Result,
    output wire ZERO,
    output wire [63:0] Read_Data,
    output wire [6:0] opcode,
    output wire Branch,
    output wire MemRead,
    output wire MemtoReg,
    output wire MemWrite,
    output wire ALUSrc,
    output wire RegWrite,
    output wire [1:0] ALUOp
);

wire [63:0] PC_In; //input to program counter it decides the next PC value
Program_Counter pc(clk,reset,PC_In,PC_Out);
```

```
wire [63:0] Y; //calculating PC+4
Adder add1(PC_Out, 4, Y);

Instruction_Memory instr_mem(PC_Out,Instruction); //fetching instruction from
instruction memory

wire [2:0] funct3;
wire [6:0] funct7;
InsParser ip(Instruction,opcode,rd, funct3, rs1,rs2,funct7);

//generates control signals based on opcode
Control_Unit cu(opcode,Branch, MemRead, MemtoReg, MemWrite, ALUSrc, RegWrite,
ALUOp);

//reads from/write to the register file
registerFile regf(WriteData,rs1,rs2,rd, RegWrite,clk, reset, ReadData1, ReadData2);

//generates immediate data from the instruction
ImmGen IG(Instruction, imm_data);

wire [3:0] Funct = {Instruction[30], funct3};
wire [3:0] Operation;
ALU_Control aluc(ALUOp,Funct,Operation); //determines ALUOp on basis of instruction
and control signals

Mux m1(ReadData2,imm_data ,ALUSrc,dataout); //to decide second ALU operand

ALU_64_bit alu(ReadData1, dataout,Operation, Result,ZERO); //performing alu
operaation

Data_Memory dm(Result,ReadData2,clk, MemWrite, MemRead,Read_Data);

Mux m2(Read_Data,Result,MemtoReg,WriteData); //decides data to be written back to the
register

//calculating branch target address
wire [63:0] b = imm_data << 1;
wire [63:0] out;
Adder add2(PC_Out,b,out);

//deciding next PC value based on branch condition
wire PC_Src = Branch&ZERO;
Mux m3(Y,out,PC_Src,PC_In);

endmodule
```

Task 2

*Dry run the processor and add values of each signal in the table in the respective clock cycle. Make sure to mention **bit size** for each signal. Use **hex** values for all signals **except** opcode, mention opcode in **binary** followed by instruction name in brackets for example 0110011(add).*

Cycle#	Bit Size	1	2	3	4
Inst_Addr	64	0x0	0x4	0x8	0xC
Instruction	32	0x02853483	0x009a84b3	0x00148493	0x02953423
Opcode	7	0000011	0110011	0010011	0100011
Rs1	5	0xA	0x15	0x09	0x0A
Rs2	5	0x08	0x09	0x01	0x09
Rd	5	0x09	0x09	0x09	0x08
WriteData (Reg)	64	0x28	0x0	0x0	0x33
Imm_data	64	0x0	0x09	0x01	0x28
ReadData1	64	0x0	0x16	0x00	0xB
ReadData2	64	0x0	0x28	0x02	0x0
Branch	1	0x0	0x0	0x0	0x0
MemWrite	1	0x0	0x0	0x0	0x1
MemRead	1	0x1	0x0	0x0	0x0
MemtoReg	1	0x1	0x0	0x0	X
ALUOp	2	0x0	0x2	0x0	0x0
ALUSrc	1	0x1	0x0	0x1	0x1
RegWrite	1	0x1	0x1	0x1	0x0
Mem_Address	64	0x28	0x3e	0x01	0x33
Read_Data	64	0x908	0x0	0x0	0x0
WriteData (Mem)	64	0x28	0x0	0x0	0x33

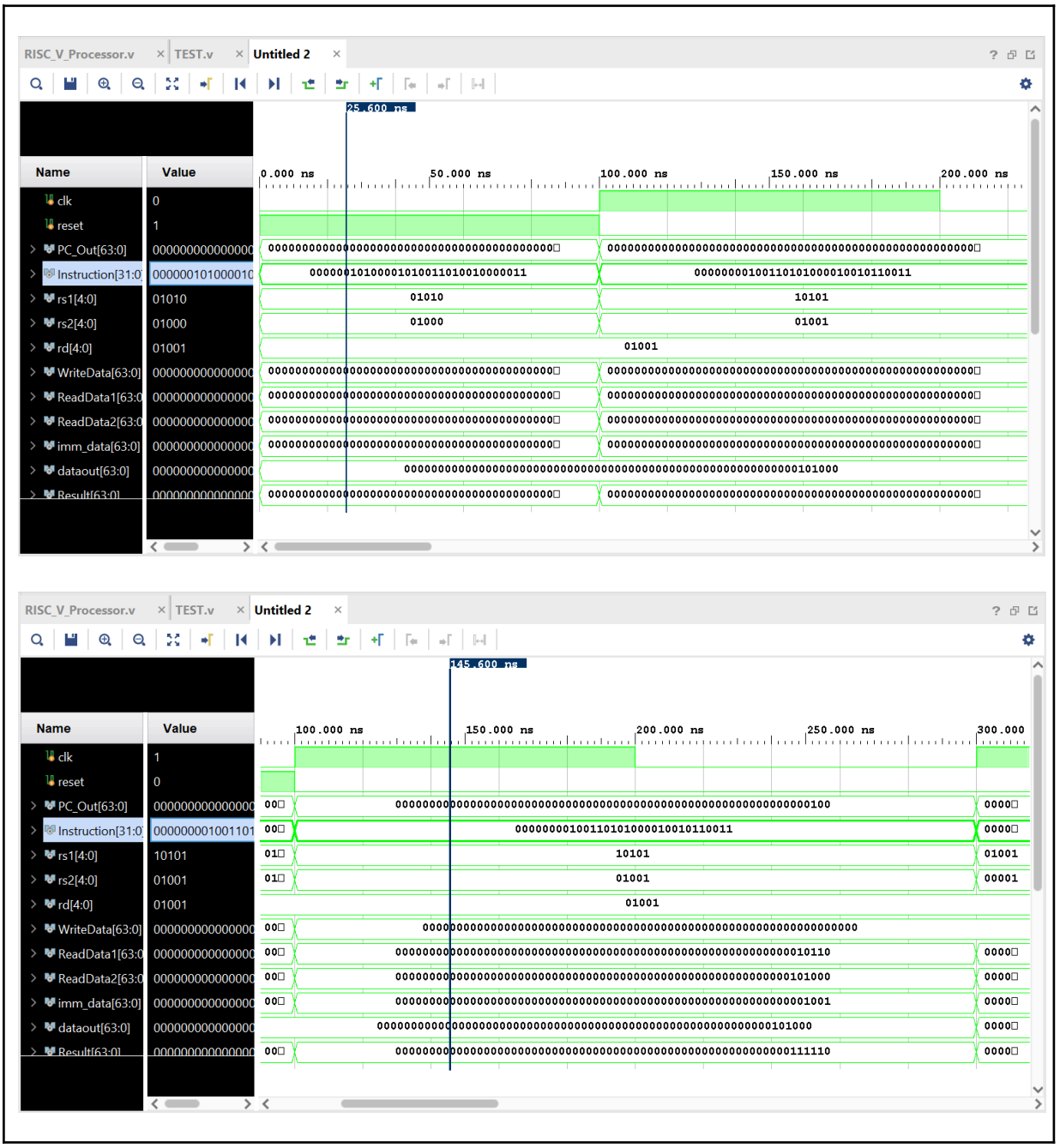
Mention the assembly code to each instruction:

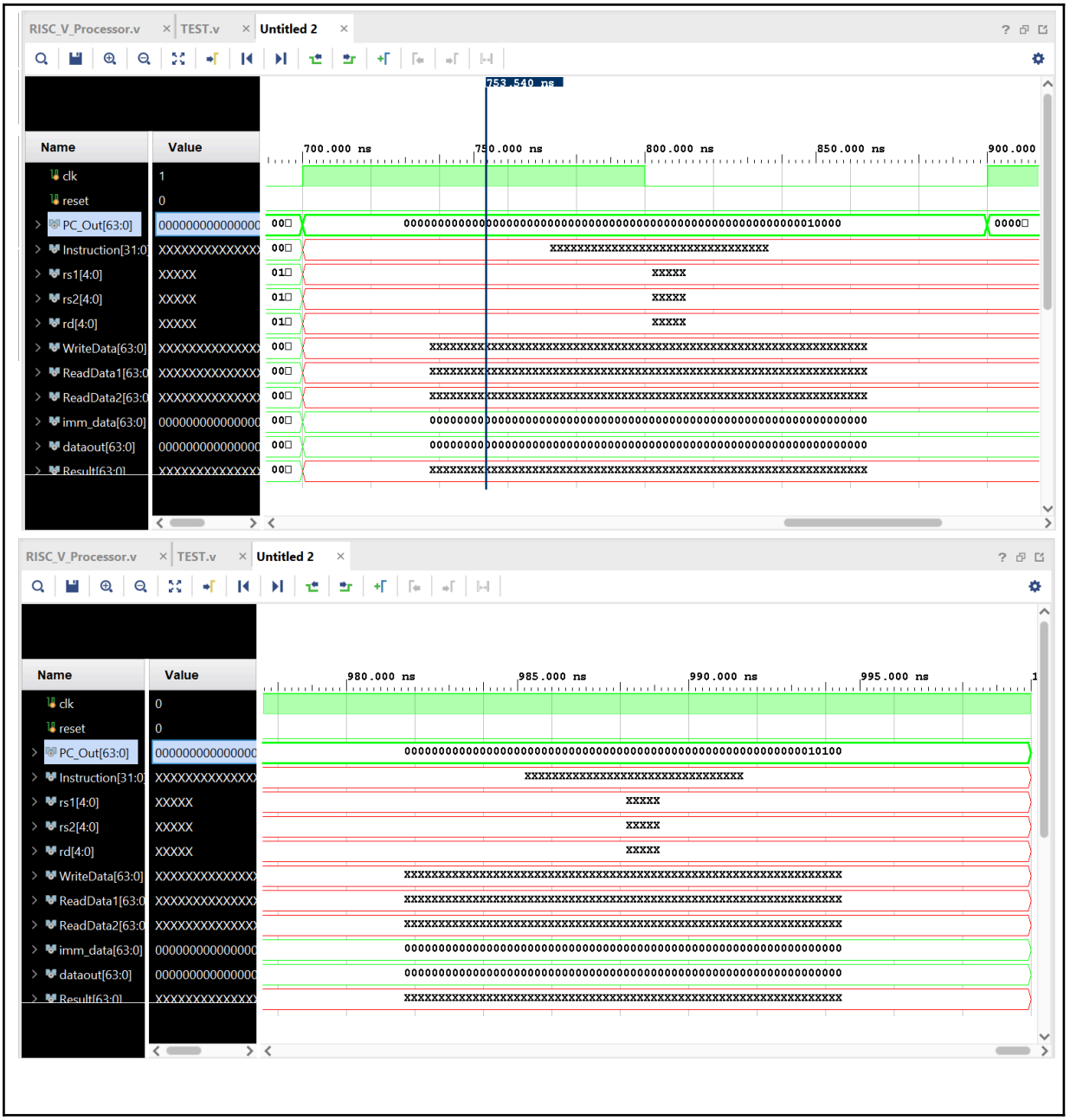
	Instruction in Binary	Assembly code
1	00000010100001010011010010000011	ld x9, 40(x10)
2	00000000100110101000010010110011	add x9, x21, x9
3	00000000000101001000010010010011	addi x9, x9, 1
4	00000010100101010011010000100011	sd x9, 40(x10)

Add your testbench here

```
`timescale 1ns / 1ps

module RISCV_testbench( );
reg clk;
reg reset;
wire [63:0] PC_Out;
wire [31:0] Instruction;
wire [4:0] rs1;
wire [4:0] rs2;
wire [4:0] rd;
wire [63:0]WriteData;
wire [63:0]ReadData1;
wire [63:0]ReadData2;
wire [63:0]imm_data;
wire [63:0] dataout;
wire [63:0] Result;
wire ZERO;
wire [63:0] Read_Data;
wire [6:0] opcode;
wire Branch;
wire MemRead;
wire MemtoReg;
wire MemWrite;
wire ALUSrc;
wire RegWrite;
wire [1:0] ALUOp;

RISC_V_Processor risc(clk, reset, PC_Out, Instruction,rs1,rs2,rd, WriteData,
    ReadData1, ReadData2, imm_data, dataout, Result, ZERO, Read_Data,
    opcode, Branch,MemRead,MemtoReg,MemWrite,ALUSrc,RegWrite,ALUOp);

initial
begin
clk = 0;
reset = 1;

#100
reset = 0;
end

always #100
clk = ~clk;

endmodule
```

**Add snip of relevant signals' waveforms. Make sure the irrelevant area of the snip is cropped.*




RISC_V_Processor.v
TEST.v
Untitled 2
753.540 ns
Name
Value
700.000 ns
750.000 ns
800.000 ns
850.000 ns
900.000
clk
1
reset
0
PC_Out[63:0]
Instruction[31:0]
rs1[4:0]
rs2[4:0]
rd[4:0]
WriteData[63:0]
ReadData1[63:0]
ReadData2[63:0]
imm_data[63:0]
dataout[63:0]
Result[63:0]
RISC_V_Processor.v
TEST.v
Untitled 2
Name
Value
980.000 ns
985.000 ns
990.000 ns
995.000 ns
clk
0
reset
0
PC_Out[63:0]
Instruction[31:0]
rs1[4:0]
rs2[4:0]
rd[4:0]
WriteData[63:0]
ReadData1[63:0]
ReadData2[63:0]
imm_data[63:0]
dataout[63:0]
Result[63:0]

Exercise

Write down Task 2's corresponding C code below. Assume that the relevant register used in this exercise is given a variable name "h" and the memory array is given a name "A".

```
//ld x9, 40(x10)
h = A[5];

// add x9, x21, x9
// x21 stored in variable k
h = h + k;;

// addi x9, x9, 1
h = h + 1;

// sd x9, 40(x10)
A[5] = h;
```

Lab 11- Design of a Single Cycle RISC V Processor
Assessment Rubric

Name:	Student ID:

Points Distribution

Task No.	LR2 Code	LR 5 Results	AR 7 ort Submission
Task 1 RISC V Processor	/30	-	/20
Task 2 Test Bench and Decoded Instructions	/15	/25	
Exercise (3) C Equivalent	/10	-	
Total Points	/100 Points		
CLO Mapped	CLO 1		

For description of different levels of the mapped rubrics, please refer to the provided Lab Evaluation Assessment Rubrics.

#	Assessment Elements	Level 1: Unsatisfactory Points 0-1	Level 2: Developing Points 2	Level 3: Good Points 3	Level 4: Exemplary Points 4
LR2	**Program/Code/ Simulation Model/ Network Model**	Program/code/simulation model/network model does not implement the required functionality and has several errors. The student is not able to utilize even the basic tools of the software.	Program/code/simulation model/network model has some errors and does not produce completely accurate results. Student has limited command on the basic tools of the software.	Program/code/simulation model/network model gives correct output but not efficiently implemented or implemented by computationally complex routine.	Program/code/simulation /Network model is efficiently implemented and gives correct output. Student has full command on the basic tools of the software.
LR5	**Results & Plots**	Figures/ graphs / tables are not developed or are poorly constructed with erroneous results. Titles, captions, units are not mentioned. Data is presented in an obscure manner.	Figures, graphs and tables are drawn but contain errors. Titles, captions, units are not accurate. Data presentation is not too clear.	All figures, graphs, tables are correctly drawn but contain minor errors or some of the details are missing.	Figures / graphs / tables are correctly drawn and appropriate titles/captions and proper units are mentioned. Data presentation is systematic.
AR7	**Report Content/Cod e Comments**	Most of the questions are not answered / figures are not labelled/ titles are not mentioned / units are not mentioned. No comments are present in the code.	Some of the questions are answered, figures are labelled, titles are mentioned and units are mentioned. Few comments are stated in the code.	Majority of the questions are answered, figures are labelled, titles are mentioned and units are mentioned. Comments are stated in the code.	All the questions are answered, figures are labelled, titles are mentioned and units are properly mentioned. Proper comments are stated in the code.

Exercise