Sai Kathika saiprem@ucsb.edu
Pete Makrygiannis pmakrygiannis@ucsb.edu
Rahul Varghese rvarghese@ucsb.edu

ECE 154A Lab 4

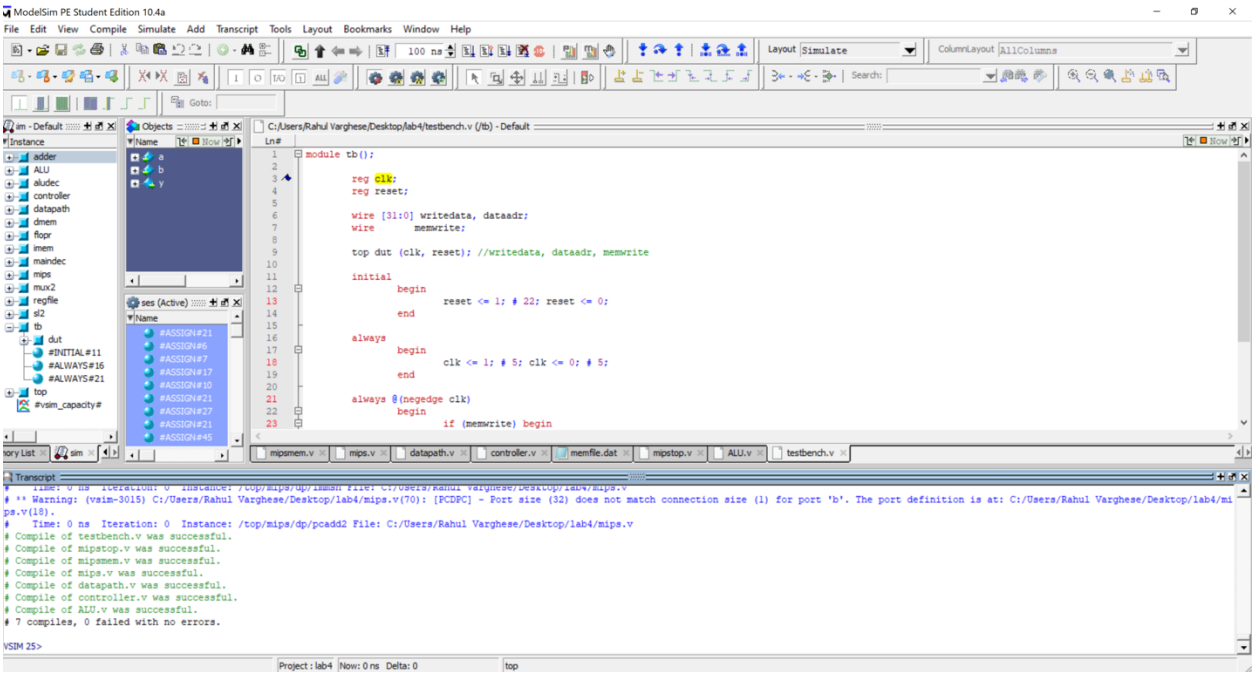
1. a. Please indicate how many hours you spent on this lab.
   Spent a total of 30 hours
   b. A completed version of Table 1.

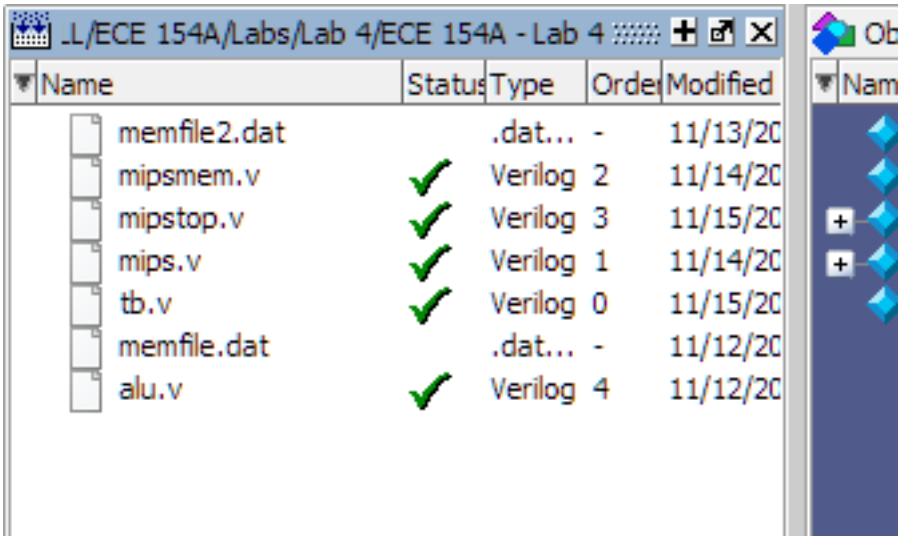
Cycle	reset	pc	Instr	branch	srca	srcb	aluresult	zero	pcsrc	Writedata	memwrite	read data
1	1	00	addi $2,$0,5 20020005	0	0	5	5	0	0	0	0	x
2	0	04	addi $3,$0,12 2003000c	0	0	c	c	0	0	0	0	x
3	0	08	addi $7,$3,-9 20067fff7	0	C	-9	3	0	0	0	0	x
4	0	0C	or $4, $7, $2 00e22025	0	3	5	7	0	0	5	0	x
5	0	10	and $5, $3, $4 00642824	0	C	7	4	0	0	7	0	x
6	0	14	add $5, $5, $4 00a42820	0	4	7	B	0	0	7	0	x
7	0	18	beq $5, $7, end 10a7000a	1	B	3	8	0	0	3	0	x
8	0	1C	slt $4, $3, $4 0064202a	0	C	7	0	1	0	7	0	x
9	0	20	beq $4, $0, around 10800001	1	0	0	0	1	1	0	0	x
10	0	28	slt $4, $7, $2 00e2202a	0	3	5	1	0	0	5	0	x
11	0	2C	add $7, $4, $5 00853820	0	1	B	C	0	0	B	0	x
12	0	30	sub $7, $7, $2 00e23822	0	C	5	7	0	0	5	0	x
13	0	34	sw $7, 68($3) ac670044	0	C	44	50	0	0	7	1	x
14	0	38	lw $2, 80($0) 8c020050	0	0	50	50	0	0	5	0	7
15	0	3C	j end 08000011	0	0	0	0	1	0	0	0	x
16	0	44	sw $2, 84($0) ac020054	0	0	54	54	0	0	7	1	x

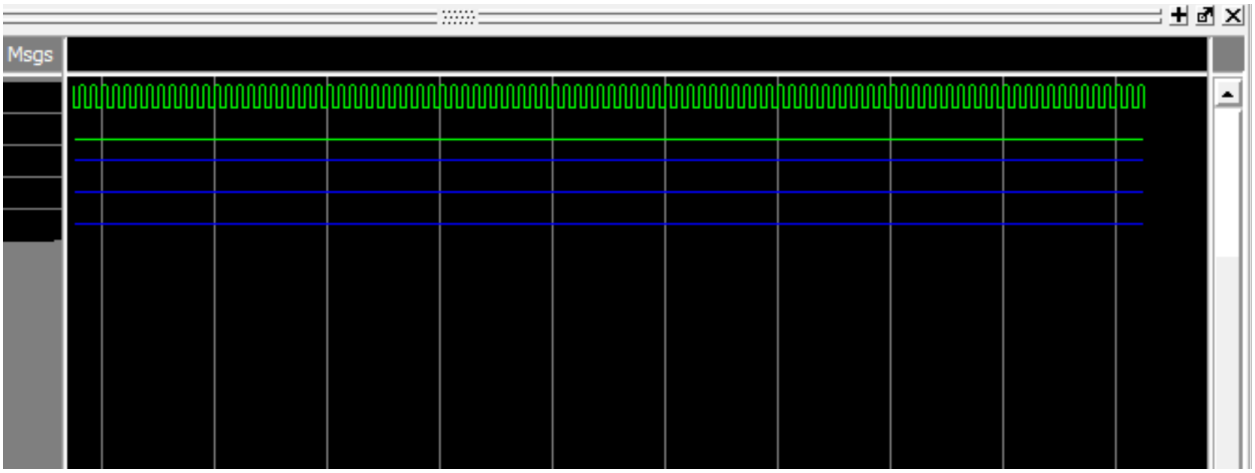
   c. An image of the simulation waveforms showing correct operation of the processor.

Our group spent a lot of time trying to get modelsim to output the waveforms, but we were unable to.

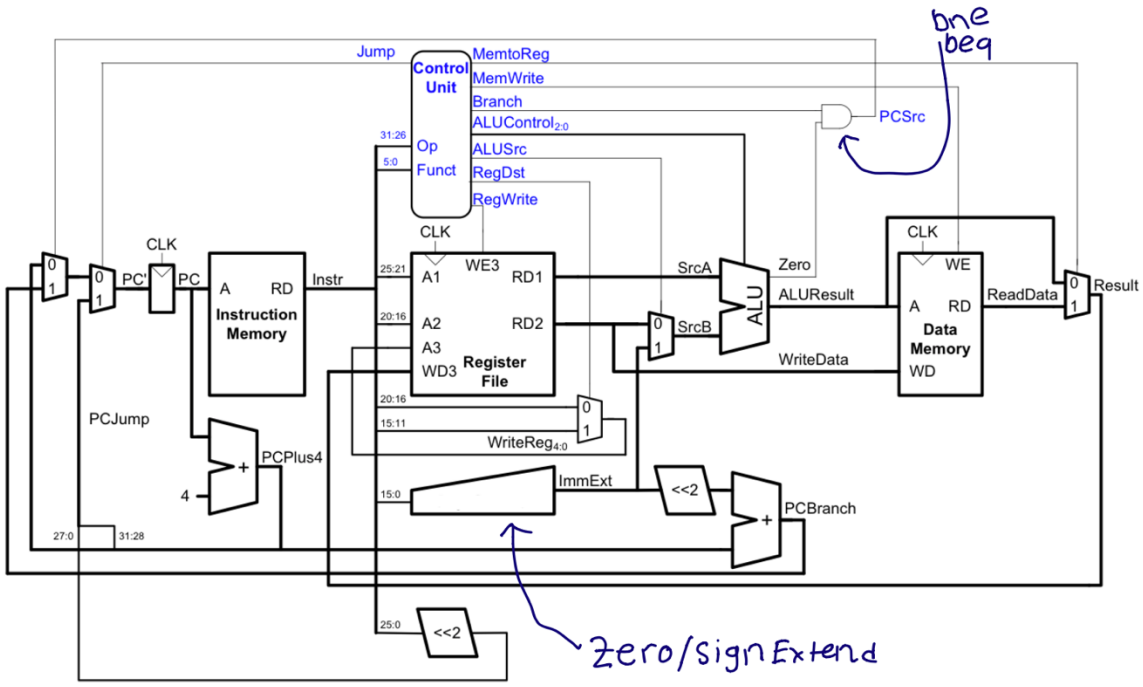
Everything compiled correctly without any errors but the waveforms were not what we anticipated from the code we wrote.


ModelSim PE Student Edition 10.4a
File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help
C:/Users/Rahul Varghese/Desktop/lab4/testbench.v (/tb) - Default
module tb();
reg clk;
reg reset;
wire [31:0] writedata, dataadr;
wire memwrite;
top dut (clk, reset); //writedata, dataadr, memwrite
initial
begin
reset <= 1; # 22; reset <= 0;
end
always
begin
clk <= 1; # 5; clk <= 0; # 5;
end
always @(negedge clk)
begin
if (memwrite) begin
mipsmem.v
mips.v
datapath.v
controller.v
memfile.dat
mipstop.v
ALU.v
testbench.v
Transcript
** Warning: (vsim-3015) C:/Users/Rahul Varghese/Desktop/lab4/mips.v(70): [PCDPC] - Port size (32) does not match connection size (1) for port 'b'. The port definition is at: C:/Users/Rahul Varghese/Desktop/lab4/mips.v(18).
Time: 0 ns Iteration: 0 Instance: /top/mips/dp/pcadd2 File: C:/Users/Rahul Varghese/Desktop/lab4/mips.v
Compile of testbench.v was successful.
Compile of mipstop.v was successful.
Compile of mipsmem.v was successful.
Compile of mips.v was successful.
Compile of datapath.v was successful.
Compile of controller.v was successful.
Compile of ALU.v was successful.
7 compiles, 0 failed with no errors.
VSIM 25>
Project : lab4
Now: 0 ns
Delta: 0
top


.L/ECE 154A/Labs/Lab 4/ECE 154A - Lab 4
Name
Status
Type
Order
Modified
memfile2.dat .dat... - 11/13/20
mipsmem.v Verilog 2 11/14/20
mipstop.v Verilog 3 11/15/20
mips.v Verilog 1 11/14/20
tb.v Verilog 0 11/15/20
memfile.dat .dat... - 11/12/20
alu.v Verilog 4 11/12/20



d. Marked up versions of the datapath schematic and decoder tables that add the ori and bne instructions.



Single-cycle MIPS processor

e. Your Verilog code for your modified MIPS processor

controller.v

```
always @*
  case(op)
      6'b000000: controls <= 9'b110000010; // RTYPE
      6'b100011: controls <= 9'b101001000; // LW
      6'b101011: controls <= 9'b001010000; // SW
      6'b000100: controls <= 9'b000100001; // BEQ
      6'b001000: controls <= 9'b101000000; // ADDI
      6'b000010: controls <= 9'b000000100; // J


      // CHANGES START
      // added functionallity of ORI and BNE to opcodes
      6'b001101: controls <= 9'b101000011; // ORI
      6'b000101: controls <= 9'b000100001; // BNE
      // CHANGES END

      default:   controls <= 9'bxxxxxxxxx;  // Not found, illegal
  endcase
endmodule
```

```
   always @*
     case(aluop)
         2'b00: alucontrol <= 3'b010;


         // CHANGES START
         // added beq and ori
         // included sub needed to beq
         2'b01: alucontrol <= 3'b110;
         2'b11: alucontrol <= 3'b001;
         // CHANGES END


         default: case(funct)          // R-type instructions
             6'b100000: alucontrol <= 3'b010; // add
             6'b100010: alucontrol <= 3'b110; // sub
             6'b100100: alucontrol <= 3'b000; // and
             6'b100101: alucontrol <= 3'b001; // or
             6'b101010: alucontrol <= 3'b111; // slt
             default:   alucontrol <= 3'bxxx; // ???
         endcase
     endcase
```

Datapath.v

```
// CHANGES START
// make changes to include functionality for both
// 0 and sign extend
module extend  (input  [15:0]  a,
                input  [2:0]   alucontrol,
                input          regdst,
                output [31:0]  y);
  assign y = (alucontrol == 3'b001 & regdst == 0) ? {{16'b0}, a} : {{16{a[15]}}, a};
endmodule
// CHANGES END
```

Mips.v

```
  // CHANGES START
  // add bne functionality
  assign pcsrc = (op == 6'b000101) ? branch & ~zero : branch & zero;
  // CHANGES END: here
```

```
// **PUT YOUR CODE HERE**
  wire [4:0] writereg;
  wire [31:0] pcnext, pcnextbr, pcplus4, pcbranch;

  // CHANGES START
  wire [31:0] imm, signimmsh;
  // CHANGES END
```

```
  //CHANGES START
  extend     ext(instr[15:0], alucontrol[2:0], regdst, imm);

  mux2 #(32)  srcbmux(writedata, imm, alusrc, srcb); // replaced signimm with imm
  //CHANGES END
```

Mipsmem.v

```

      // CHANGES STARt
      $readmemh("memfile2.dat",RAM);
      // CHANGES END
```

Mipstop.v

```
// CHANGES START
module top     (input clk, reset,
        output [31:0] writedata, dataadr,
        output memwrite);

  wire [31:0] pc, instr, readdata;

  // CHANGES END
```

Memfile2.dat

```
34088000
20098000
350a8001
11090005
0128582a
15600001
08000001
01485022
350800ff
016a5820
01484022
ad680052
```

f. Completed version of Tables 2 and 3 for the modified MIPS processor.

Table 2. Extended functionality for the main decoder:

Instruction	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	$ALUOp_{1:0}$	Jump
R-type	000000	1	1	0	0	0	0	10	0
`lw`	100011	1	0	1	0	0	1	00	0
`sw`	101011	0	X	1	0	1	X	00	0
`beq`	000100	0	X	0	1	0	X	01	0
`addi`	001000	1	0	1	0	0	0	00	0
`j`	000010	0	X	X	X	0	X	XX	1
`ori`	001101	1	0	1	0	0	0	00	0
`bne`	000101	0	X	0	1	0	X	01	0

Table 3. Extended functionality for the ALU decoder:

$ALUOp_{1:0}$	Meaning
00	Add
01	Subtract
10	Look at `funct` field
11	Ori

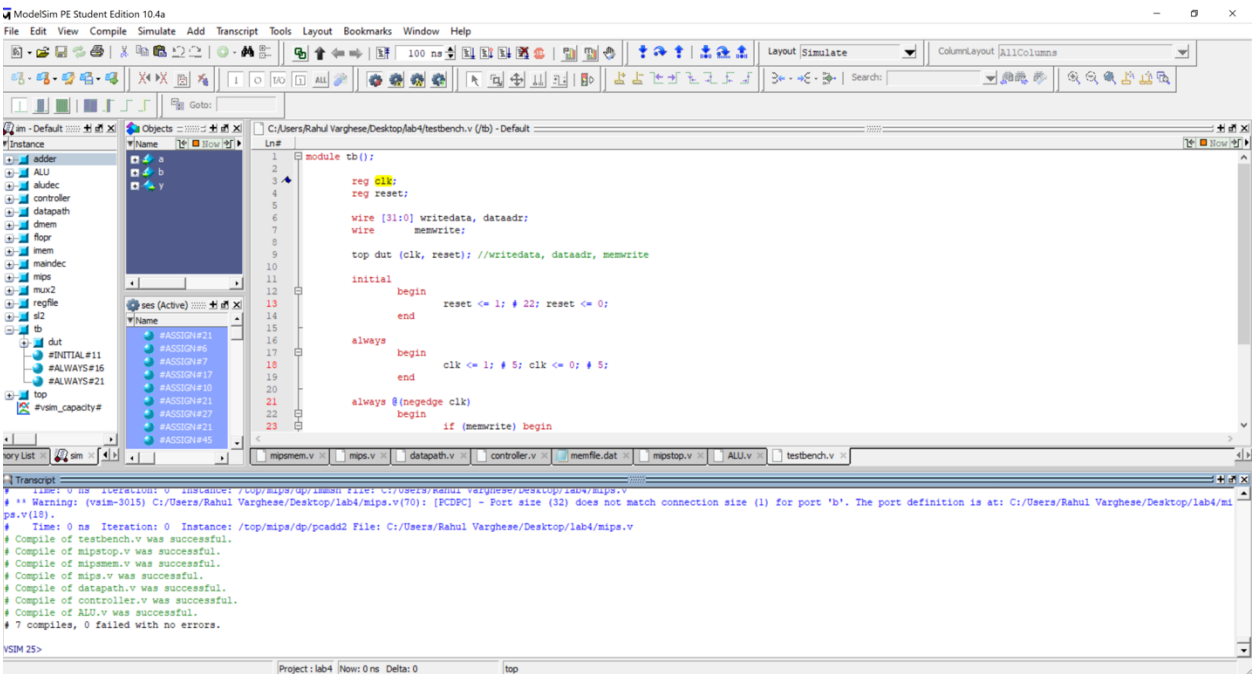
g. The contents of your memfile2.dat containing your test2 machine language code.

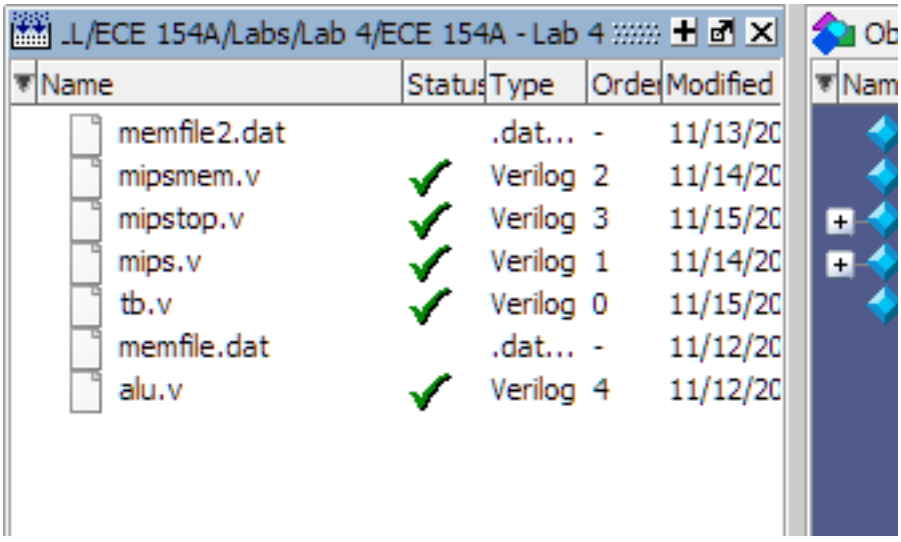
```
1   34088000
2   20098000
3   350a8001
4   11090005
5   0128582a
6   15600001
7   08000001
8   01485022
9   350800ff
10  016a5820
11  01484022
12  ad680052
```

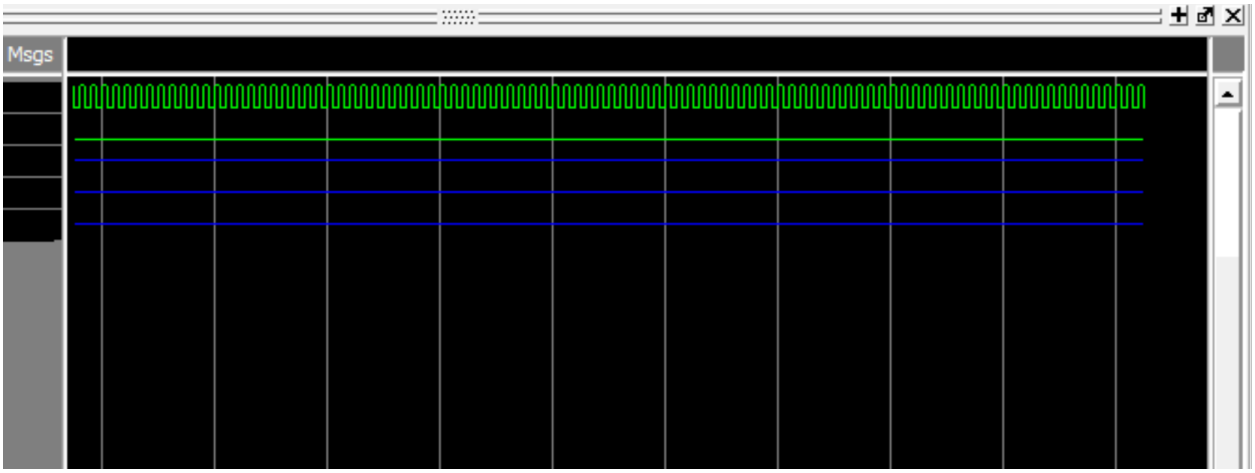
h. An image of the simulation waveforms showing correct operation of your modified processor on the new program. What address and data value are written by the sw instruction?

Our group spent a lot of time trying to get modelsim to output the waveforms, but we were unable to.

Everything compiled correctly without any errors but the waveforms were not what we anticipated from the code we wrote.







BOX Code

There is folder before changes which is the code before the changes implemented

There is folder after changes which is the code after the changes implemented

https://ucsb.box.com/s/2bfy4169vd3xvosai0kqle5qbghw9h84