Assignment -2

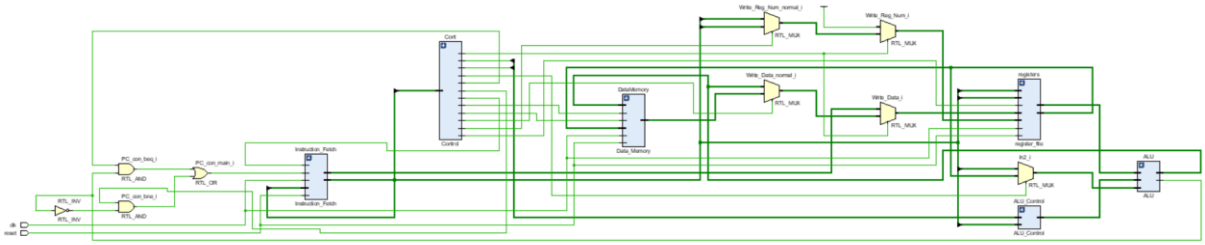
Computer Architecture Lab

Group no. 14

Name : Kaushik Kalakonda

ID Student : 2020A3PS0468H

The RTL Schematic of the processor showing the required data path and control path:



Truth Table of Main Control Unit:

Instruction Type	RegDst	ALUSource	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Branch1	ALUOp1	ALUOp0	Jump	AL
R	1	0	0	1	0	0	0	0	1	0	0	0
LW	0	1	1	1	1	0	0	0	0	0	0	0
SW	x	1	x	0	0	1	0	0	0	0	0	0
BEQ	x	0	x	0	0	0	1	0	0	1	0	0
BNE	x	0	x	0	0	0	0	1	0	1	0	0
J	x	x	x	0	0	0	x	x	x	x	1	0
JAL	x	x	x	1	0	0	x	x	x	x	1	1

Truth Table for the ALU Control Unit:

ALUOp		Function Code						ALU Control Input				Desired ALU Action
0	0	x	x	x	x	x	x	0	0	1	0	add
x	1	x	x	x	x	x	x	0	1	1	0	sub
1	x	x	x	0	0	0	0	0	0	1	0	add
1	x	x	x	0	0	0	0	0	1	1	0	sub
1	x	x	x	0	1	1	0	0	0	0	0	and
1	x	x	x	0	1	1	1	0	0	0	1	or
1	x	x	x	1	0	0	0	0	1	1	1	slt
1	x	x	x	0	1	1	0	1	0	0	0	xor
1	X	x	x	0	1	1	1	1	0	0	1	nor

Program loaded in the instruction memory:

Instruction Memory contains the following program:

Start: add $t0,$t1,$t2

sw $t2, 0($t3)

lw $t5, 0($t3)

and $t0,$t1,$t2

or $t0,$t1,$t2

j label //label=6

label: bne $t30, $t31

beq $t30, $t31

slt $t0,$t1,$t2

xor $t0,$t1,$t2

nor $t0,$t1,$t2

sub $t0,$t1,$t2

jal Start

Verilog Code for all the modules:

ALU.v

ALU.v

'KAUSHIK/Downloads/Comp_Arch_Lab/SCDataPath_Assignment_2/SCDataPath_Assig

```
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////////////////////
21
22  module ALU(ALUCtl, A,B, ALUOut, Zero);
23
24      input [3:0] ALUCtl;
25      input [31:0] A,B;
26      output reg [31:0] ALUOut;
27      output  Zero;
28
29      assign Zero= (ALUOut==0); // Zero is true if ALUOUT is 0
30
31      always@(ALUCtl,A,B) begin //reevaluate if these change
32          case (ALUCtl)
33              0: ALUOut = A&B;
34              1: ALUOut = A|B;
35              2: ALUOut = A+B;
36              6: ALUOut = A-B;
37              7: ALUOut = A<B;
38              8: ALUOut = A^B;
39              9: ALUOut = ~(A|B);
40              default: ALUOut = 0;
41          endcase
42      end
43
44  endmodule
45
```

SCDataPath.v

```
24
25  input clk, reset;
26
27  wire [31:0] ALU_output;
28  wire [31:0] Instruction_Code ;
29  wire [31:0] PCinc;
30  wire Jump,AL,PC_con_main,PC_con_beq,PC_con_bne;
31  wire [31:0] extend;
32  wire RegDst,MemRead,MemtoReg,ALUOp1,ALUOp0,MemWrite,ALUSource,RegWrite,Branch,Branch1;
33  wire [4:0] Read_Reg_Num_1 =Instruction_Code[25:21];
34  wire [4:0] Read_Reg_Num_2 =Instruction_Code[20:16];
35  wire [31:0] Read_Data_1,Read_Data_2;
36  wire [31:0] Write_Data;
37  wire [4:0] Write_Reg_Num;
38  wire [5:0] alpha = Instruction_Code[5:0];
39  wire [1:0] ALUOp = {ALUOp1,ALUOp0};
40  wire [3:0] ALU_Ctl;
41  wire [31:0] In2;
42  wire [31:0] ALU_Result;
43  wire zero;
44  wire [31:0] Read_Data;
45
46  wire [31:0] Write_Data_normal , Write_Data_al;
47  wire [4:0] Write_Reg_Num_al , Write_Reg_Num_normal;
48
49  Instruction_Fetch Instruction_Fetch(clk,reset,Instruction_Code,PC_con_main,Jump,extend,PCinc);
50
51  Control Cont(Instruction_Code[31:26],RegDst,MemRead,MemtoReg,ALUOp1,ALUOp0,MemWrite,ALUSource,RegWrite,Branch,Branch1,Jump,AL);
52
53  assign Write_Reg_Num_normal = RegDst ? Instruction_Code[15:11] : Instruction_Code[20:16];
54  assign Write_Reg_Num_al = 5'b11111; // $31
55
56  assign Write_Reg_Num = AL ? Write_Reg_Num_al : Write_Reg_Num_normal;
57
58  register_file registers(Read_Reg_Num_1,Read_Reg_Num_2,Write_Reg_Num,Write_Data,Read_Data_1,Read_Data_2,RegWrite,clk,reset);
59
60  ALU_Control ALU_Control(ALUOp, alpha, ALU_Ctl);
61
62  wire [15:0] msb = {16{Instruction_Code[15]}};
63  assign extend = {msb,Instruction_Code[15:0]};
64  assign In2 = ALUSource ? extend : Read_Data_2; // 1:0
65
66  ALU ALU(ALU_Ctl,Read_Data_1,In2,ALU_Result,zero);
67  Data_Memory DataMemory(ALU_Result,Read_Data_2,MemWrite,MemRead,Read_Data,clk,reset);
68
69  assign Write_Data_normal = !MemtoReg ? ALU_Result : Read_Data;
70  assign Write_Data_al = PCinc;
71
72  assign Write_Data = AL ? Write_Data_al : Write_Data_normal;
73
74
75  assign ALU_output= ALU_Result;
76
77  assign PC_con_main = PC_con_beq | PC_con_bne;
78  assign PC_con_beq = Branch & zero;
79  assign PC_con_bne = Branch1 & (!zero);
80  endmodule
```

Instruction_fetch.v

instruction_fetch.v

C:/Users/KAUSHIK/Downloads/Comp_Arch_Lab/SCDataPath_Assignment_2/SCDataPath_Assignment_2.srcs/sources_1/i

```
22
23  module Instruction_Fetch(
24      input clk,
25      input reset,
26      output [31:0] Instruction_Code,
27      input PC_con,
28      input Jump,
29      input [31:0] extend,
30      //output reg [31:0] PC,
31      output [31:0] Add_Source_1);
32
33       reg [31:0] PC;
34
35       wire [31:0] out_1,ALU_out,out,PC_jump;
36       wire [31:0] Add_Source_2; //Add_Source_1 = PC +4;
37       wire [27:0] Ins_part = Instruction_Code [25:0] <<<2;
38
39       assign PC_jump = {Add_Source_1[31:28],Ins_part};
40       assign Add_Source_2 = extend<<<2; // arithmetic left shift
41       assign Add_Source_1= PC +4;
42       assign ALU_out = Add_Source_2 + Add_Source_1;
43       assign out_1 = PC_con ? ALU_out : Add_Source_1;
44       assign out = Jump ? PC_jump : out_1;
45
46       Instruction_Memory a1(PC,reset,Instruction_Code);
47
48       always @(posedge clk, negedge reset)
49
50       begin
51
52       if (reset==0)
53       PC <=0;
54
55       else
56       PC<= out;
57
58       end
59
60  endmodule
```

Instruction_Memory.v

Instruction_Memory.v

C:/Users/KAUSHIK/Downloads/Comp_Arch_Lab/SCDataPath_Assignment_2/SCDataPath_Assignment_2.srcs/sources_1/imports/Comp_Arch_Lab/Instruction_Memory.v

```
module Instruction_Memory(
    input [31:0] PC,
    input reset,
    output [31:0] Instruction_Code
    );
    reg [7:0] Mem [55:0]; // Creating 56 memory locations each of 8 bit size.
assign Instruction_Code = {Mem[PC], Mem [PC +1], Mem[PC+2], Mem[PC+3]};
always @(reset)
begin
    if (reset==0)
        begin

        Mem[0] = 8'h00; Mem[1] = 8'h01; Mem[2] = 8'h10; Mem[3] = 8'h20;     //add $t0,$t1,$t2 000000 00000 00001 00010 00000100000 ;
                                                                              //0000 0000 0000 0001 0001 000000100000
        Mem[4] = 8'hac; Mem[5] = 8'h62; Mem[6] = 8'h00; Mem[7] = 8'h20;     // sw $t2, 0($t3)   1010|11 00|011 0|0010 0000 0000 0010 0000

        Mem[8] = 8'h8c; Mem[9] = 8'h65; Mem[10] = 8'h00; Mem[11] = 8'h00;   // lw $t5, 0($t3)   1000|11 00|011 0|0101 0000 0000 0000 0000

        Mem[12] = 8'h00; Mem[13] = 8'h01; Mem[14] = 8'h10; Mem[15] = 8'h24; //and $t0,$t1,$t2 000000 00000 00001 00010 00000100100 ;
                                                                              //0000 0000 0000 0001 0001 0000 0010 0100
        Mem[16] = 8'h00; Mem[17] = 8'h01; Mem[18] = 8'h10; Mem[19] = 8'h25; //or $t0,$t1,$t2 000000 00000 00001 00010 00000100101 ;
                                                                              //0000 0000 0000 0001 0001 0000 0010 0101
        Mem[20] = 8'h08; Mem[21] = 8'h00; Mem[22] = 8'h00; Mem[23] = 8'h06; //j label ; 000010 00000000000000000000000110
                                                                              //0000 1000 0000 0000 0000 0000 0000 0110
        Mem[24] = 8'h17; Mem[25] = 8'hdf; Mem[26] = 8'h00; Mem[27] = 8'h01; // bne $t30, $t31 000101 11110 11111 0000000000000001
                                                                              //0001 0111 1101 1111 0000 0000 0000 0001
        Mem[28] = 8'h13; Mem[29] = 8'hdf; Mem[30] = 8'h00; Mem[31] = 8'h01; // beq $t30, $t31 000100 11110 11111 0000000000000001
                                                                              //0001 0011 1101 1111 0000 0000 0000 0001
        Mem[36] = 8'h00; Mem[37] = 8'h01; Mem[38] = 8'h10; Mem[39] = 8'h2b; //slt $t0,$t1,$t2 000000 00000 00001 00010 00000101010 ;
                                                                              //0000 0000 0000 0001 0001 0000 0010 1010
        Mem[40] = 8'h00; Mem[41] = 8'h01; Mem[42] = 8'h10; Mem[43] = 8'h26; //xor $t0,$t1,$t2 000000 00000 00001 00010 00000100110 ;
                                                                              //0000 0000 0000 0001 0001 0000 0010 0110
        Mem[44] = 8'h00; Mem[45] = 8'h01; Mem[46] = 8'h10; Mem[47] = 8'h27; //nor $t0,$t1,$t2 000000 00000 00001 00010 00000100111 ;
                                                                              //0000 0000 0000 0001 0001 0000 0010 0111
        Mem[48] = 8'h00; Mem[49] = 8'h01; Mem[50] = 8'h10; Mem[51] = 8'h22; //sub $t0,$t1,$t2 000000 00000 00001 00010 00000100010 ;
                                                                              //0000 0000 0000 0001 0001 0000 0010 0010
        Mem[52] = 8'h0c; Mem[53] = 8'h00; Mem[54] = 8'h00; Mem[55] = 8'h00; //jal label ; 000011 00000000000000000000000000
                                                                              //0000 1100 0000 0000 0000 0000 0000 0000
        end
end
endmodule
```

Control.v

Control.v

C:/Users/KAUSHIK/Downloads/Comp_Arch_Lab/SCDataPath_Assignment_2/SCDataPath_Assignment_2.srcs/sources_1/imports/Comp_Arch_Lab/Control.v

```
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Control(Func_Code,RegDst,MemRead,MemtoReg,ALUOp1,ALUOp0,MemWrite,ALUSource,RegWrite,Branch,Branch1,Jump,AL);

input [5:0] Func_Code;
output wire RegDst,MemRead,MemtoReg,ALUOp1,ALUOp0,MemWrite,ALUSource,RegWrite,Branch,Branch1,Jump,AL;
reg [11:0] out;
wire [5:0] r=0;   //000000
wire [5:0] lw=35; //100011
wire [5:0] sw=43; //101011
wire [5:0] beq=4; //000100
wire [5:0] bne=5; //000101
wire [5:0] j=2;   //000010
wire [5:0] jal=3;//000011
always @(Func_Code)
begin
    case(Func_Code)
        r : out=12'b100100001000;
        lw: out=12'b011110000000;
        sw: out=12'bx1x001000000;
        beq:out=12'bx0x000100100;
        bne:out=12'bx0x000010100;
        j:  out=12'bxxx000xxxx10;
        jal:out=12'bxxx100xxxx11;// 11111 in write register, PC value in write data (32 bit)

    endcase
end
assign {RegDst,ALUSource,MemtoReg,RegWrite,MemRead,MemWrite,Branch,Branch1,ALUOp1,ALUOp0,Jump,AL} = out;
endmodule
```

Register_file.v

register_file.v

C:/Users/KAUSHIK/Downloads/Comp_Arch_Lab/SCDataPath_Assignment_2/SCDataPath_Assignment_2.srcs/sources_1/imports/Comp_Arch_Lab/register_file.v

```
22
23  module register_file(
24      input [4:0] Read_Reg_Num_1,
25       input [4:0] Read_Reg_Num_2,
26       input [4:0] Write_Reg_Num,
27       input [31:0] Write_Data,
28       output [31:0] Read_Data_1,
29       output [31:0] Read_Data_2,
30       input RegWrite,
31       input clk,
32       input reset
33       );
34
35  reg [31:0] RegMemory [31:0];
36
37  integer i;
38
39  assign Read_Data_1 = RegMemory[Read_Reg_Num_1];
40  assign Read_Data_2 = RegMemory[Read_Reg_Num_2];
41
42  always @(posedge clk or negedge reset)
43  begin
44  if (reset==0) // if reset is equal to 1 then register file will be refreshed
45  begin
46      for(i=0; i<30; i=i+1)
47      RegMemory[i] = 10*i+1;
48      RegMemory[30] = 10;
49      RegMemory[31] = 10;
50  end
51  end
52  always @(Write_Data)
53  begin
54  if (RegWrite ==1)
55      RegMemory[Write_Reg_Num] = Write_Data;
56  end
57  endmodule
58
```

ALU_Control.v

ALU_Control.v

:/Users/KAUSHIK/Downloads/Comp_Arch_Lab/SCDataPath_Assignment_2/SCDataPath_Assignm

```
19  //
20  //////////////////////////////////////////////////////////////////////////////////
21
22
23  module ALU_Control(ALUOp, FuncCode, ALUCtl);
24
25  input [1:0] ALUOp;
26  input [5:0] FuncCode;
27  output reg [3:0] ALUCtl;
28
29  wire [7:0] In= {ALUOp, FuncCode};
30
31  always @(*)
32  begin
33  casex(In)
34      8'b00xxxxxx : ALUCtl <= 4'b0010; //add
35      8'bx1xxxxxx : ALUCtl <= 4'b0110; //sub
36      8'b1xxx0000 : ALUCtl <= 4'b0010; //add //100000
37      8'b1xxx0010 : ALUCtl <= 4'b0110; //sub //100010
38      8'b1xxx0100 : ALUCtl <= 4'b0000; //and //100100
39      8'b1xxx0101 : ALUCtl <= 4'b0001; //or  //100101
40      8'b1xxx1010 : ALUCtl <= 4'b0111; //slt //101010
41      8'b1xxx0110 : ALUCtl <= 4'b1000; //xor //100110
42      8'b1xxx0111 : ALUCtl <= 4'b1001; //nor //100111
43      default: ALUCtl <= 4'b1001; //nor
44  endcase
45  end
46  endmodule
```

Data_Memory.v

Project Summary | Data_Memory.v

/Downloads/Comp_Arch_Lab/SCDataPath_Assignment_2/SCDataPath_Assignment_2.srcs/sources_1/imports/Comp_Arch_La

```
20  //////////////////////////////////////////////////////////////////////////////////
21
22
23  module Data_Memory(Address,Write_Data,MemWrite,MemRead,Read_Data,clk,reset);
24
25  input [31:0] Address;
26  input [31:0] Write_Data; //Data that is to be written into memory in case of store
27  input MemRead, MemWrite; // From control unit
28  output reg [31:0] Read_Data; // Data that is read and taken into register in case of load
29  input clk,reset;
30
31  integer i;
32
33  reg [31:0] mem [127:0];
34
35  always @(*)
36  begin
37  begin
38      for(i=0; i<128; i=i+1)
39      mem[i] = i+10;
40  end
41
42
43  if (MemWrite==1'b1 && MemRead ==1'b0)
44      mem[Address] = Write_Data;
45  else if(MemWrite==1'b0 && MemRead ==1'b1)
46      Read_Data = mem[Address];
47  end
48
49  endmodule
```

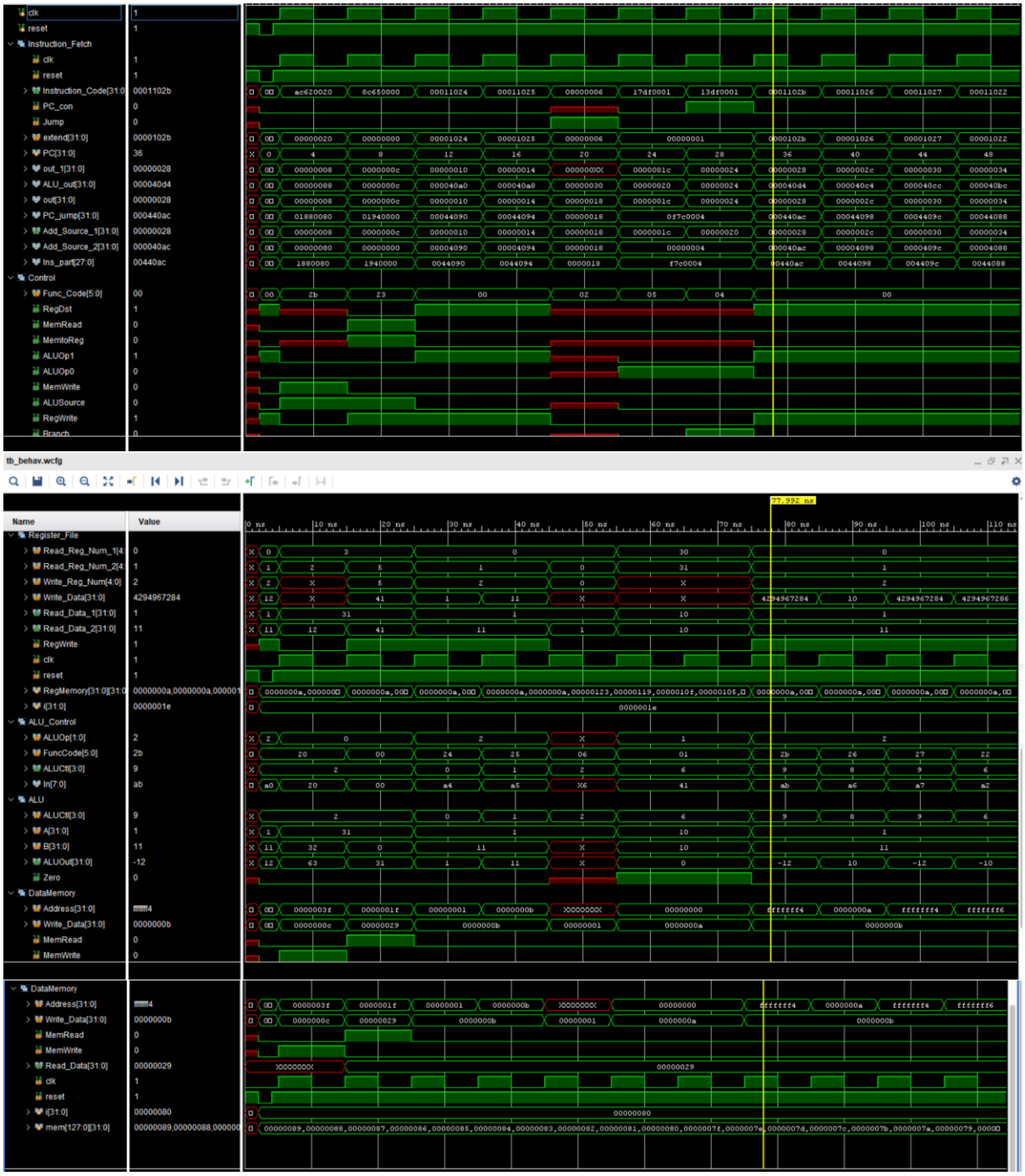
Testbench:

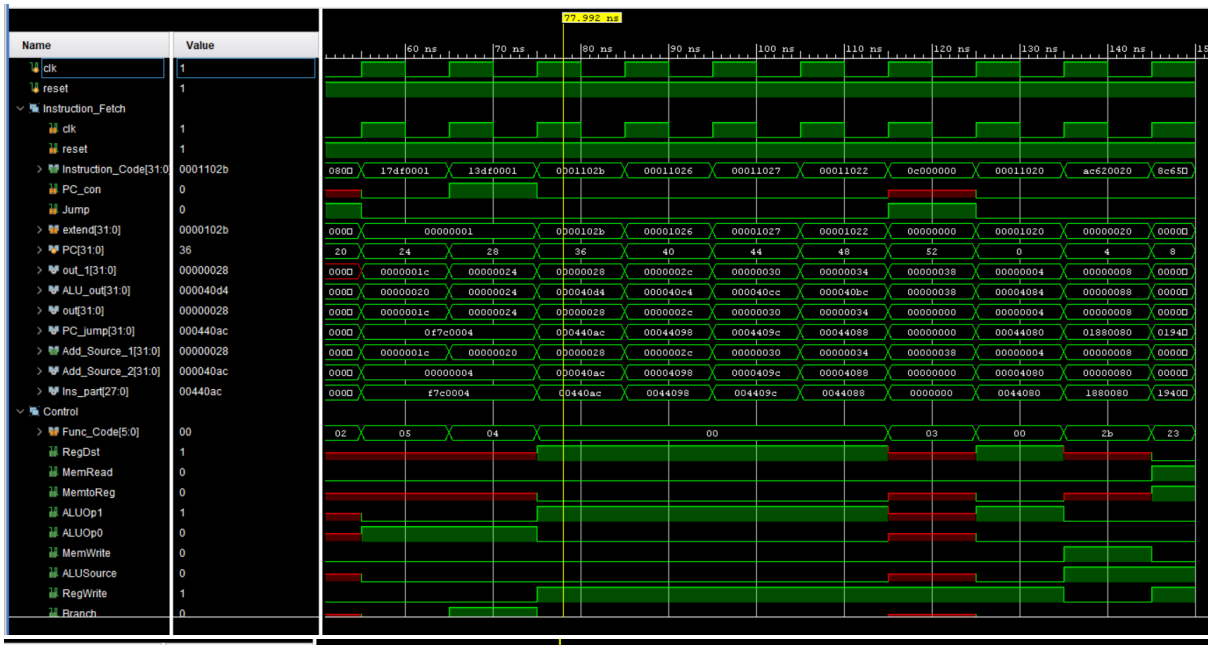
Project Summary | tb.v

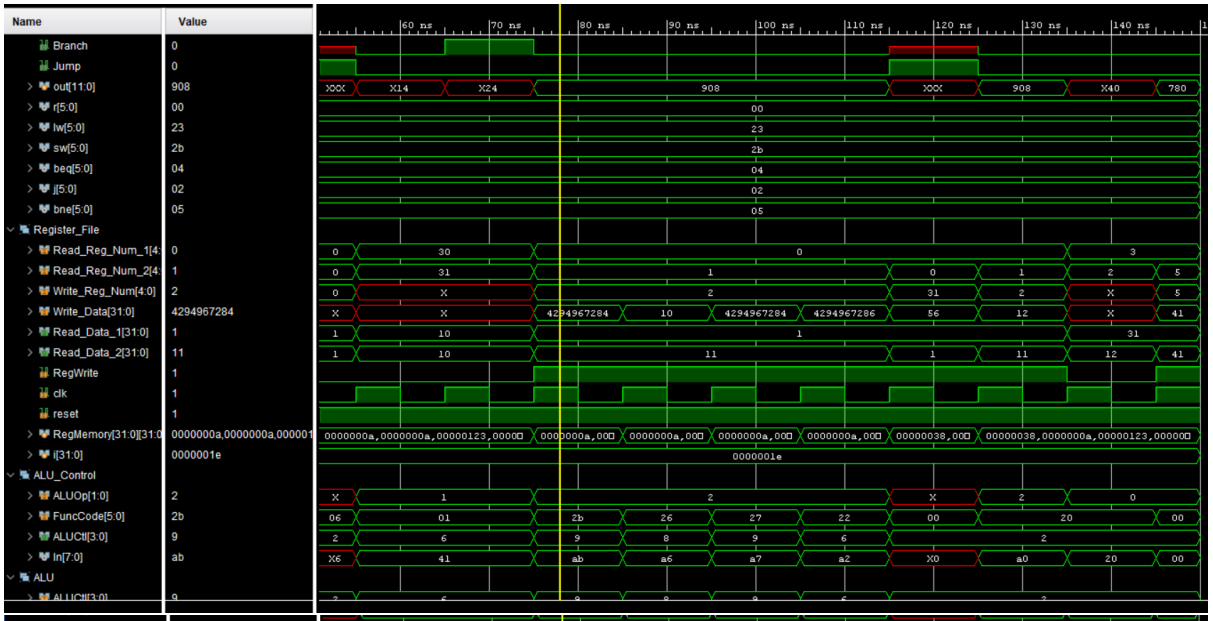
C:/Users/KAUSHIK/Downloads/Comp_Arch_Lab/SCDataPath_Assi

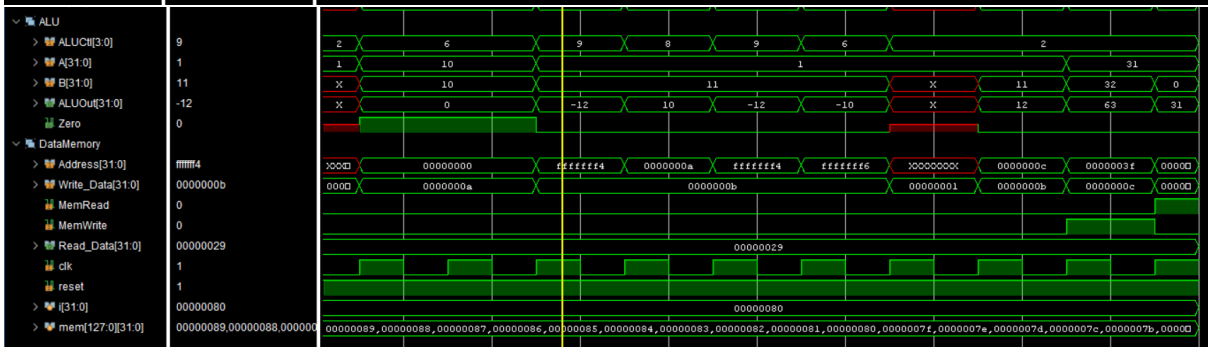
```
23  module tb();
24  reg clk, reset;
25  SCDataPath uut(clk, reset);
26
27  initial begin
28  clk = 0;
29  repeat(30)
30  #5 clk = ~clk;
31  $finish;
32  end
33
34  initial begin
35  reset = 1;
36  #2 reset = 0;
37  #2 reset = 1;
38  end
39
40  endmodule
```

Output Waveform:




27.992 ns
60 ns
70 ns
80 ns
90 ns
100 ns
110 ns
120 ns
130 ns
140 ns
Name
Value
clk 1
reset 1
Instruction_Fetch
clk 1
reset 1
Instruction_Code[31:0] 0001102b
PC_con 0
Jump 0
extend[31:0] 0000102b
PC[31:0] 36
out_1[31:0] 00000028
ALU_out[31:0] 000040d4
out[31:0] 00000028
PC_jump[31:0] 000440ac
Add_Source_1[31:0] 00000028
Add_Source_2[31:0] 000040ac
Ins_part[27:0] 00440ac
Control
Func_Code[5:0] 00
RegDst 1
MemRead 0
MemtoReg 0
ALUOp1 1
ALUOp0 0
MemWrite 0
ALUSource 0
RegWrite 1
Branch 0


Name
Value
Branch 0
Jump 0
out[11:0] 908
r[5:0] 00
lw[5:0] 23
sw[5:0] 2b
beq[5:0] 04
j[5:0] 02
bne[5:0] 05
Register_File
Read_Reg_Num_1[4:0] 0
Read_Reg_Num_2[4:0] 1
Write_Reg_Num[4:0] 2
Write_Data[31:0] 4294967284
Read_Data_1[31:0] 1
Read_Data_2[31:0] 11
RegWrite 1
clk 1
reset 1
RegMemory[31:0][31:0] 0000000a,0000000a,000001
i[31:0] 0000001e
ALU_Control
ALUOp[1:0] 2
FuncCode[5:0] 2b
ALUCtl[3:0] 9
In[7:0] ab
ALU


ALU
ALUCtl[3:0] 9
A[31:0] 1
B[31:0] 11
ALUOut[31:0] -12
Zero 0
DataMemory
Address[31:0] fffffff4
Write_Data[31:0] 0000000b
MemRead 0
MemWrite 0
Read_Data[31:0] 00000029
clk 1
reset 1
i[31:0] 00000080
mem[127:0][31:0] 00000089,00000088,000000