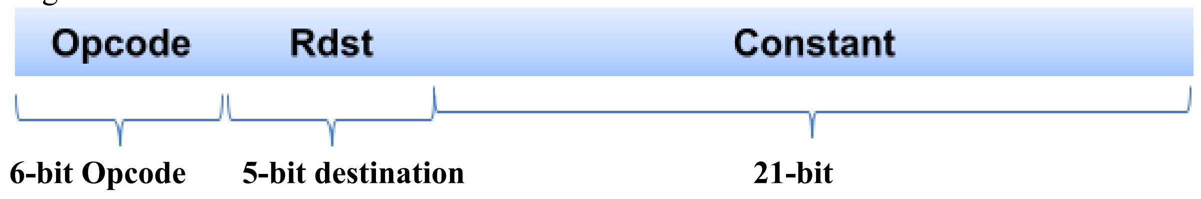
Lab 3 - Implementation of a MIPS like processor

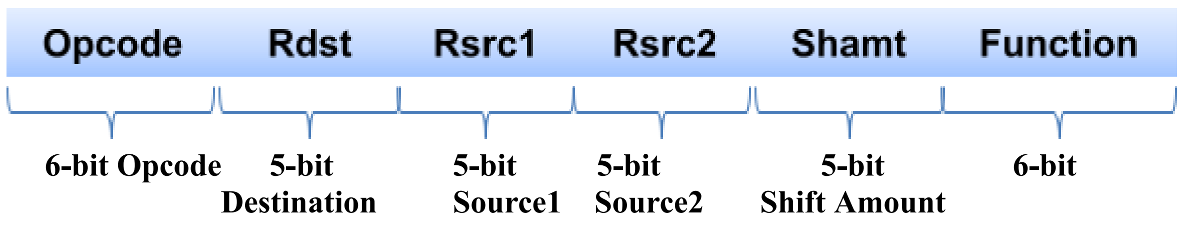
Yashi Malik
2021A3PS3056G

Design and implement (in Verilog) a Datapath and control unit for a single-cycle MIPS-like processor (including instruction memory) which has two classes of instructions. The two classes of instructions along with the example usage and instruction decoding to be used are as below

1. **Immediate Type**
   Example: li r1, constant loads immediate signed value specified in the instruction to the register R1



2. **Register Type (R-type)**
   Example: add r1, r2, r3 adds the contents of registers r2 and r3. The result of addition is written in to the register r1



Assume there are 32 32-bit general purpose registers indicated by r0, r1, r2…r31 and corresponding register numbers (00000), (00001)………..(11111).
Assume the Opcode for Immediate type and R-type instructions as below

Instruction Class	Opcode
Immediate type	111111
Register Type	000000

Additionally, R-type instructions have multiple variations defined by their function codes. The R-type instructions should include **add**, **sub**, **AND**, **OR**, **srl** (Shift right logical), **sll** (shift left logical). The different R-type instructions that the processor should support are tabulated below.

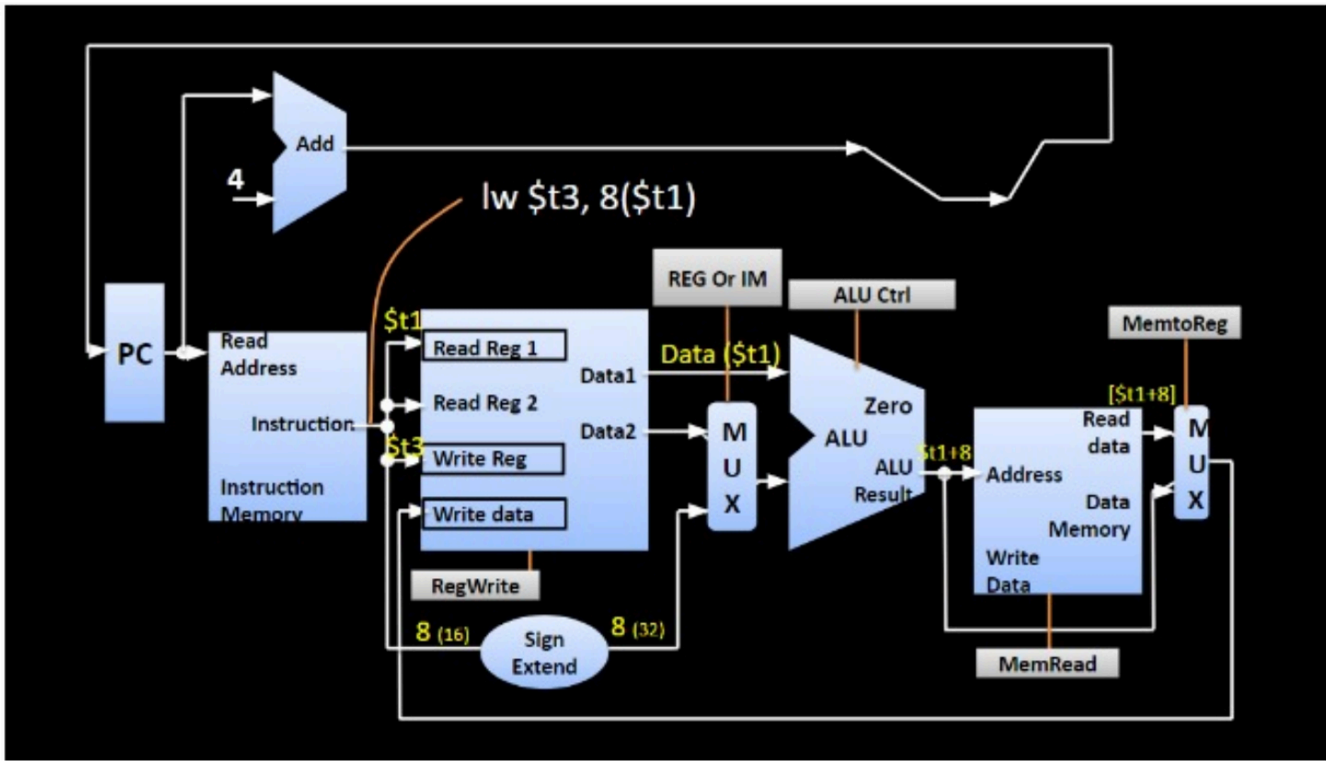
R-type Instruction	Example usage	Opcode	Rdst	Rsrc1	Rsrc2	shamt	Function
add	**add r0, r1, r2**	**000000**	**00000**	**00001**	**00010**	**00000**	**100000**
sub	**sub r4, r5, r6**	**000000**	**00100**	**00101**	**00110**	**00000**	**100010**
AND	**and r8, r9, r10**	**000000**	**01000**	**01001**	**01010**	**00000**	**100100**
OR	**and r9, r8, r10**	**000000**	**01001**	**01000**	**01010**	**00000**	**100101**
sll	**sll r11, r6, 6**	**000000**	**01011**	**00110**	**00000***	**00110**	**000000**
srl	**srl r13, r9, 10**	**000000**	**01101**	**01001**	**00000***	**01010**	**000010**

*Second source is not used for shift operations

The processor module should have only two inputs CLK and Reset. When Reset is activated the Processor starts executing instructions from the 0th location of instruction memory.

Q3.1. Draw the block-level design of the processor (datapath + control unit) for the above specifications. (you can modify the design given in the class ppts and copy the image of the final design here)

Answer:



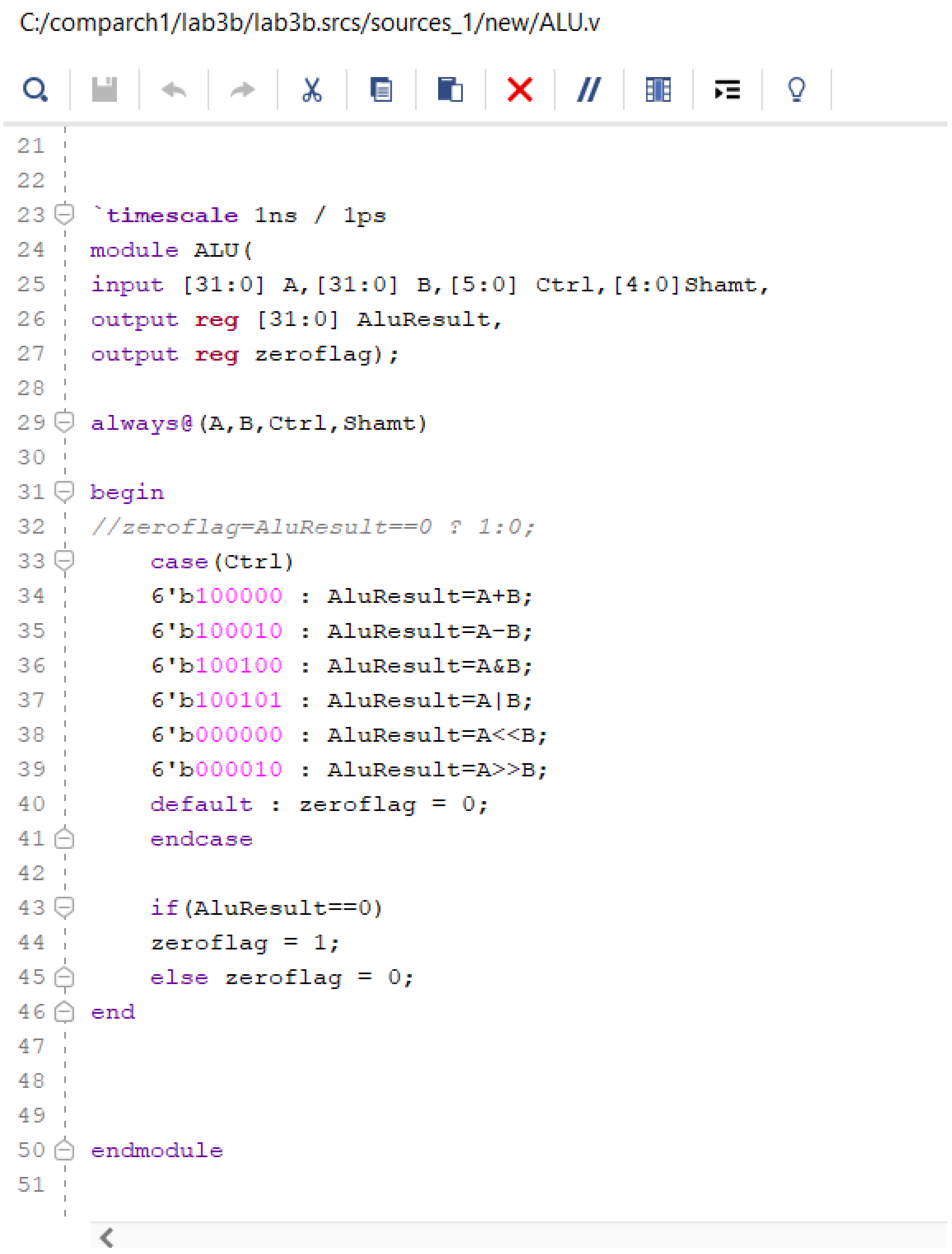
Q3.2. List the different blocks that will be required for the implementation of the datapath of the above processor.

Answer:

- Instruction fetch unit (PC+Adder+Instruction Memory)
- Register File
- Main Control Block(Main Control+control Signals)
- Sign Extension Unit
- 2:1 MUX

Q3.3. Most of the datapath blocks that are listed above have already been implemented as part of previous labs. Implement the blocks which have not been implemented in the previous labs and copy the <u>images</u> of those Verilog codes here.

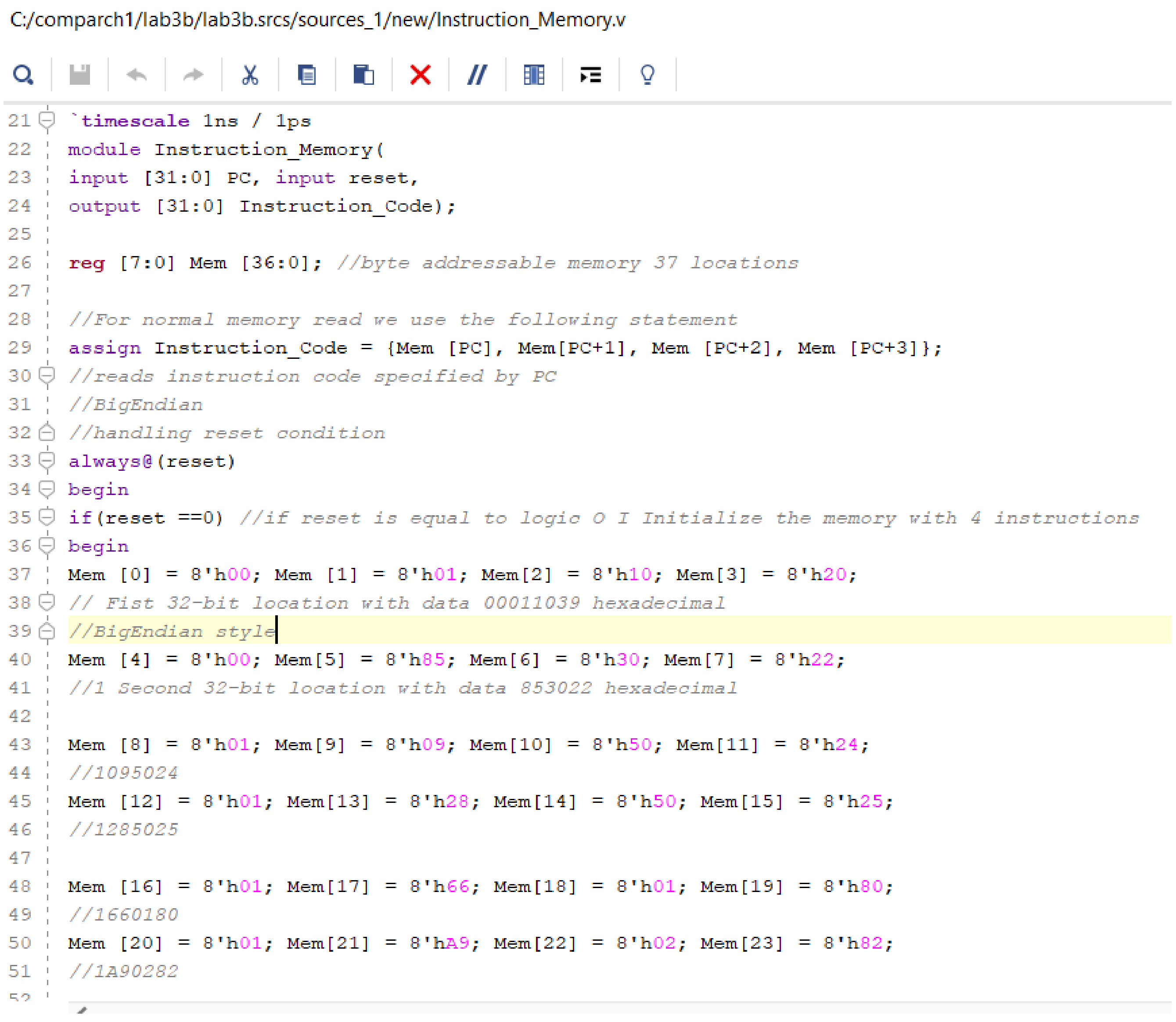
Answer: ALU



C:/comparch1/lab3b/lab3b.srcs/sources_1/new/ALU.v

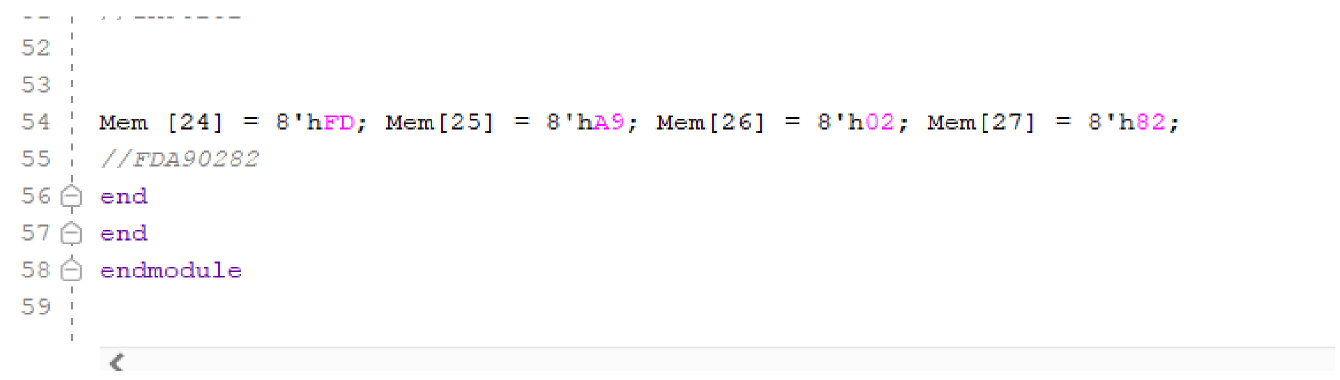
```


`timescale 1ns / 1ps
module ALU(
input [31:0] A,[31:0] B,[5:0] Ctrl,[4:0]Shamt,
output reg [31:0] AluResult,
output reg zeroflag);

always@(A,B,Ctrl,Shamt)

begin
//zeroflag=AluResult==0 ? 1:0;
    case(Ctrl)
    6'b100000 : AluResult=A+B;
    6'b100010 : AluResult=A-B;
    6'b100100 : AluResult=A&B;
    6'b100101 : AluResult=A|B;
    6'b000000 : AluResult=A<<B;
    6'b000010 : AluResult=A>>B;
    default : zeroflag = 0;
    endcase

    if(AluResult==0)
    zeroflag = 1;
    else zeroflag = 0;
end



endmodule

```

Instruction Memory



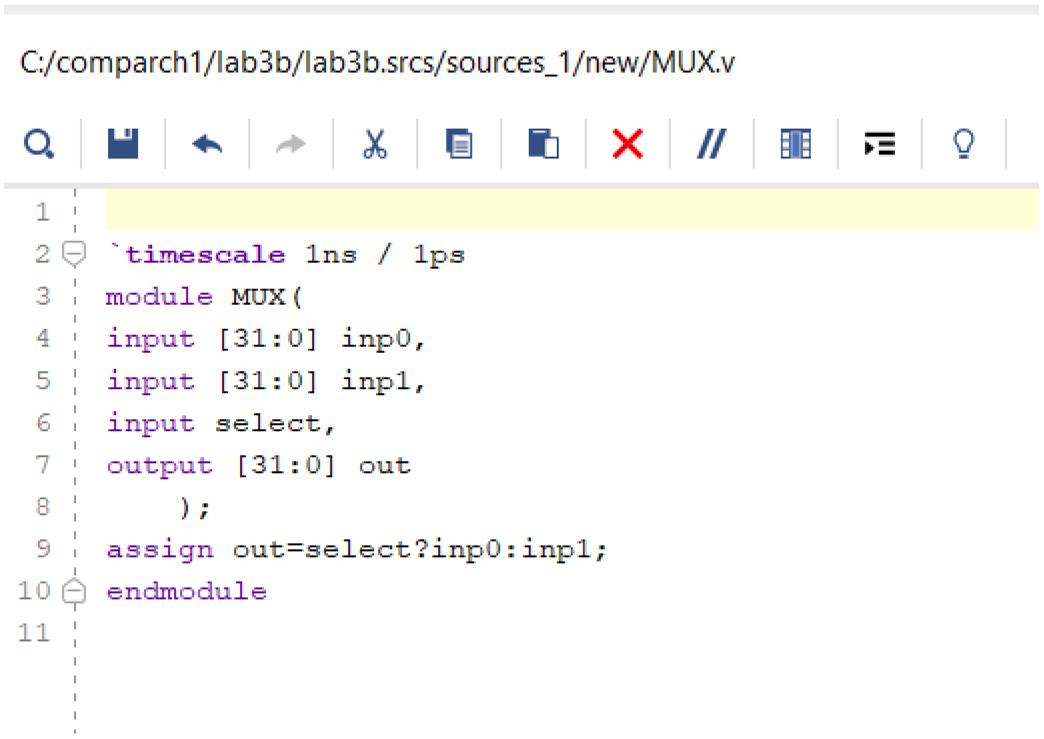
C:/comparch1/lab3b/lab3b.srcs/sources_1/new/Instruction_Memory.v

```
`timescale 1ns / 1ps
module Instruction_Memory(
input [31:0] PC, input reset,
output [31:0] Instruction_Code);

reg [7:0] Mem [36:0]; //byte addressable memory 37 locations

//For normal memory read we use the following statement
assign Instruction_Code = {Mem [PC], Mem[PC+1], Mem [PC+2], Mem [PC+3]};
//reads instruction code specified by PC
//BigEndian
//handling reset condition
always@(reset)
begin
if(reset ==0) //if reset is equal to logic 0 I Initialize the memory with 4 instructions
begin
Mem [0] = 8'h00; Mem [1] = 8'h01; Mem[2] = 8'h10; Mem[3] = 8'h20;
// Fist 32-bit location with data 00011039 hexadecimal
//BigEndian style
Mem [4] = 8'h00; Mem[5] = 8'h85; Mem[6] = 8'h30; Mem[7] = 8'h22;
//1 Second 32-bit location with data 853022 hexadecimal

Mem [8] = 8'h01; Mem[9] = 8'h09; Mem[10] = 8'h50; Mem[11] = 8'h24;
//1095024
Mem [12] = 8'h01; Mem[13] = 8'h28; Mem[14] = 8'h50; Mem[15] = 8'h25;
//1285025

Mem [16] = 8'h01; Mem[17] = 8'h66; Mem[18] = 8'h01; Mem[19] = 8'h80;
//1660180
Mem [20] = 8'h01; Mem[21] = 8'hA9; Mem[22] = 8'h02; Mem[23] = 8'h82;
//1A90282
```



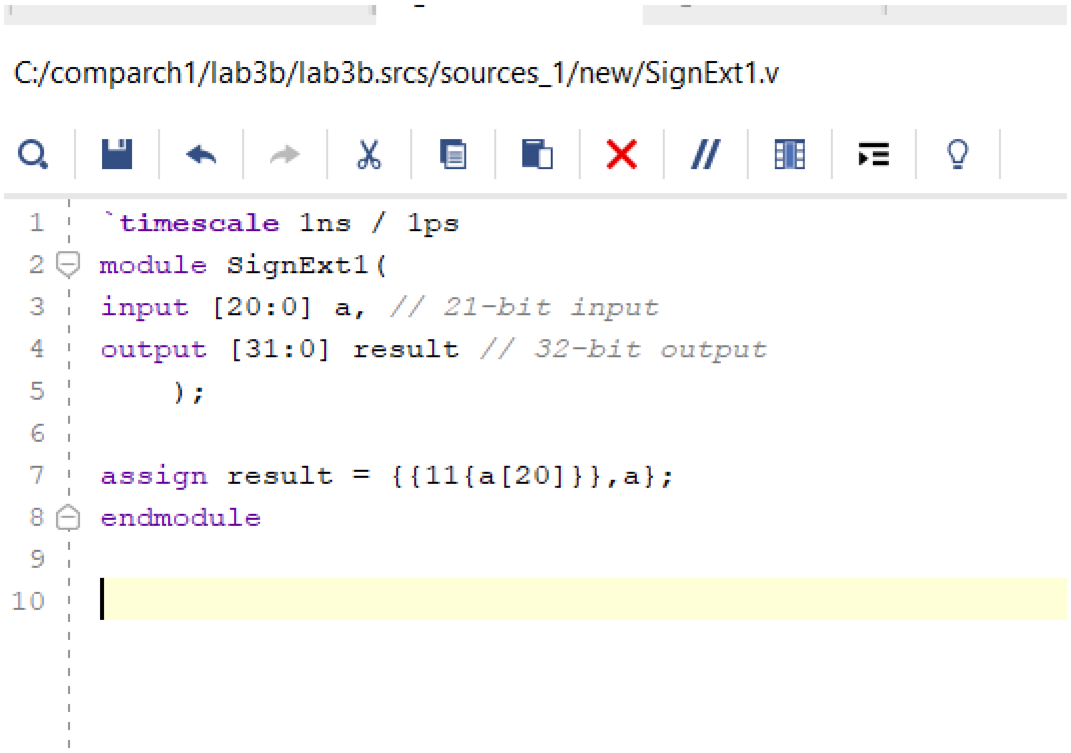
```


Mem [24] = 8'hFD; Mem[25] = 8'hA9; Mem[26] = 8'h02; Mem[27] = 8'h82;
//FDA90282
end
end
endmodule

```

MUX


C:/comparch1/lab3b/lab3b.srcs/sources_1/new/MUX.v

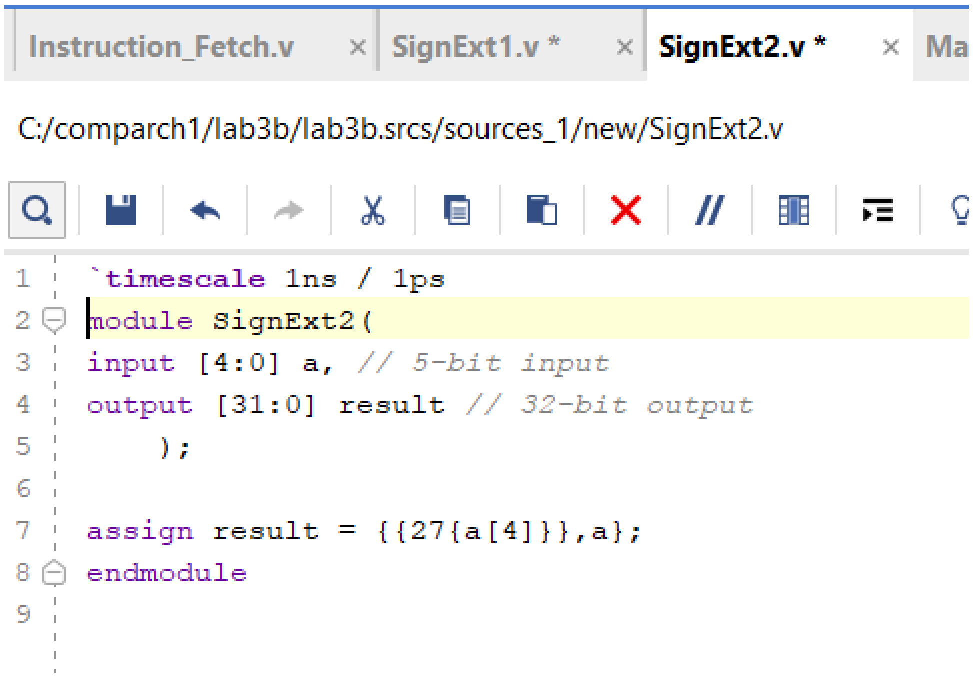
```verilog

`timescale 1ns / 1ps
module MUX(
input [31:0] inp0,
input [31:0] inp1,
input select,
output [31:0] out
    );
assign out=select?inp0:inp1;
endmodule

```

Sign extension unit -1


C:/comparch1/lab3b/lab3b.srcs/sources_1/new/SignExt1.v

```verilog
`timescale 1ns / 1ps
module SignExt1(
input [20:0] a, // 21-bit input
output [31:0] result // 32-bit output
    );

assign result = {{11{a[20]}},a};
endmodule


```

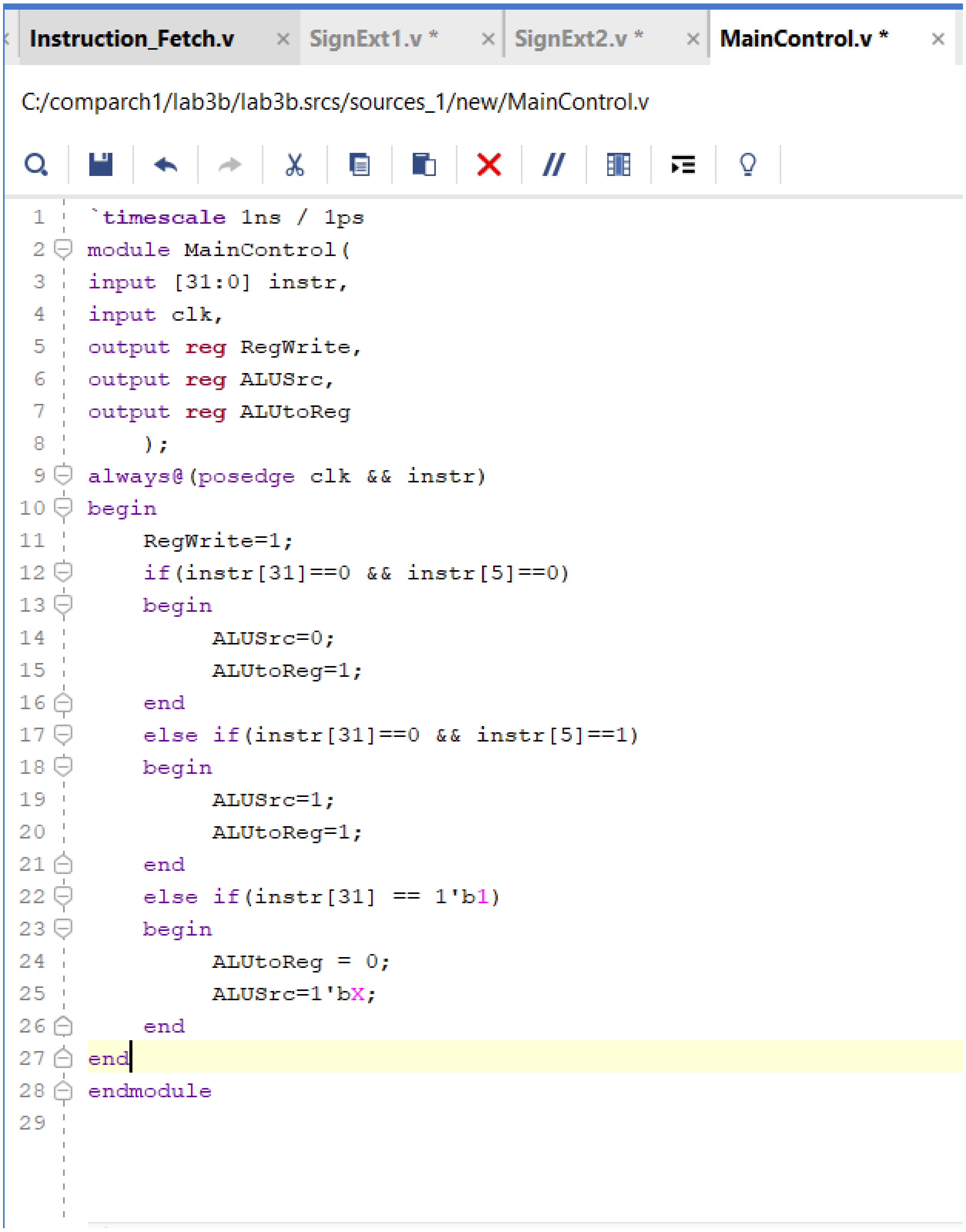
sign extension unit-2



C:/comparch1/lab3b/lab3b.srcs/sources_1/new/SignExt2.v

```verilog
`timescale 1ns / 1ps
module SignExt2(
input [4:0] a, // 5-bit input
output [31:0] result // 32-bit output
    );

assign result = {{27{a[4]}},a};
endmodule

```

Main control



C:/comparch1/lab3b/lab3b.srcs/sources_1/new/MainControl.v

```
`timescale 1ns / 1ps
module MainControl(
input [31:0] instr,
input clk,
output reg RegWrite,
output reg ALUSrc,
output reg ALUtoReg
    );
always@(posedge clk && instr)
begin
    RegWrite=1;
    if(instr[31]==0 && instr[5]==0)
    begin
        ALUSrc=0;
        ALUtoReg=1;
    end
    else if(instr[31]==0 && instr[5]==1)
    begin
        ALUSrc=1;
        ALUtoReg=1;
    end
    else if(instr[31] == 1'b1)
    begin
        ALUtoReg = 0;
        ALUSrc=1'bX;
    end
end
endmodule
```

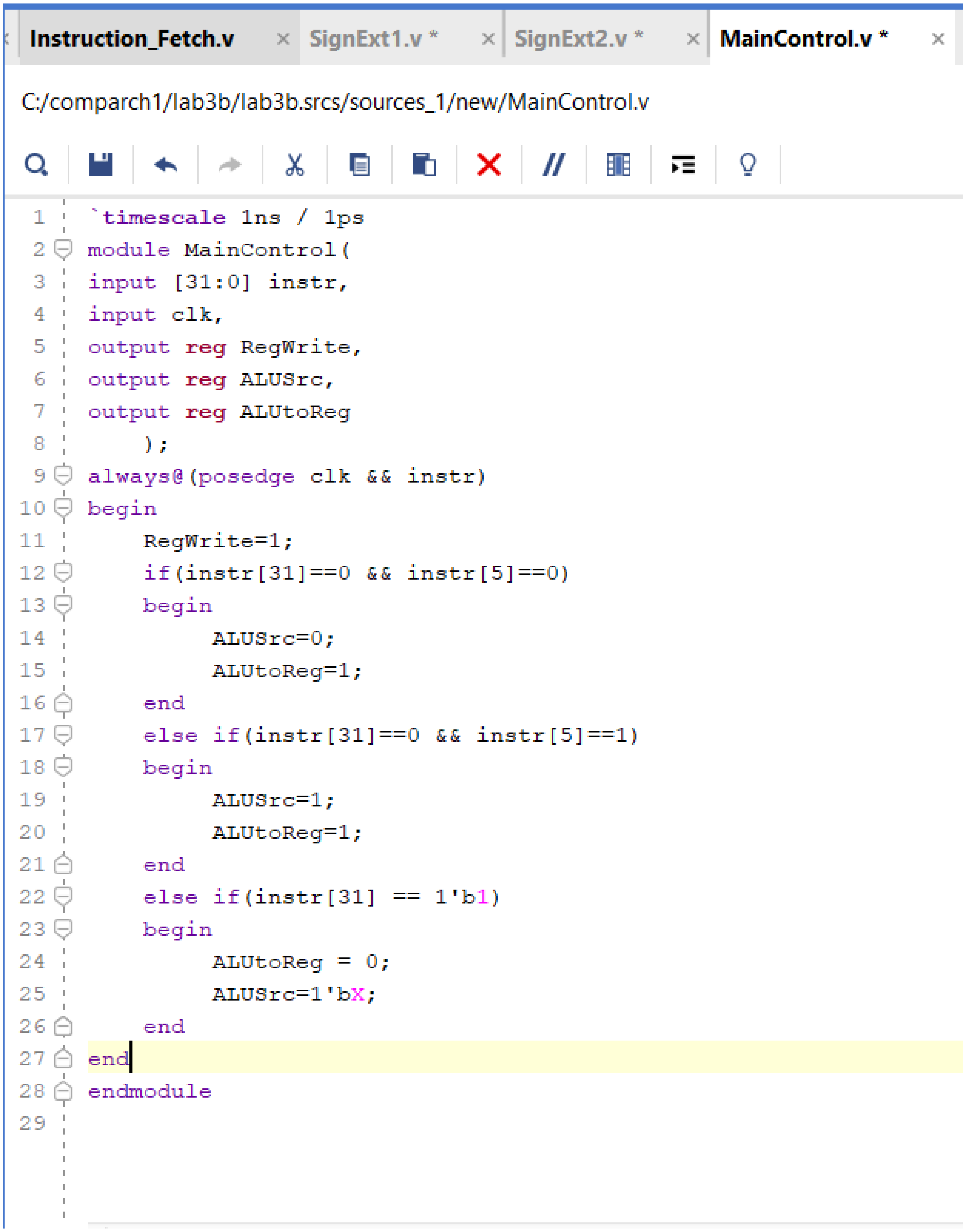
Q3.4. Assume the Main control unit generates all the control signals. List different control signals that will be required for the above processor. Also, specify the value of the control signals for different instructions.

Answer:

Control Signal Name	RegWrite.	ALUSrc	ALUtoReg
li r1, 8	1	X	0
add r0, r1, r2	1	1	1
sub r4, r5, r6	1	1	1
and r8, r9, r10	1	1	1
and r9, r8, r10	1	1	1
sll r11, r6, 6	1	0	1
srl r13, r9, 10	1	0	1

Q3.5. Implement the main control unit and copy the image of the Verilog code of the Main control unit here.

Answer:

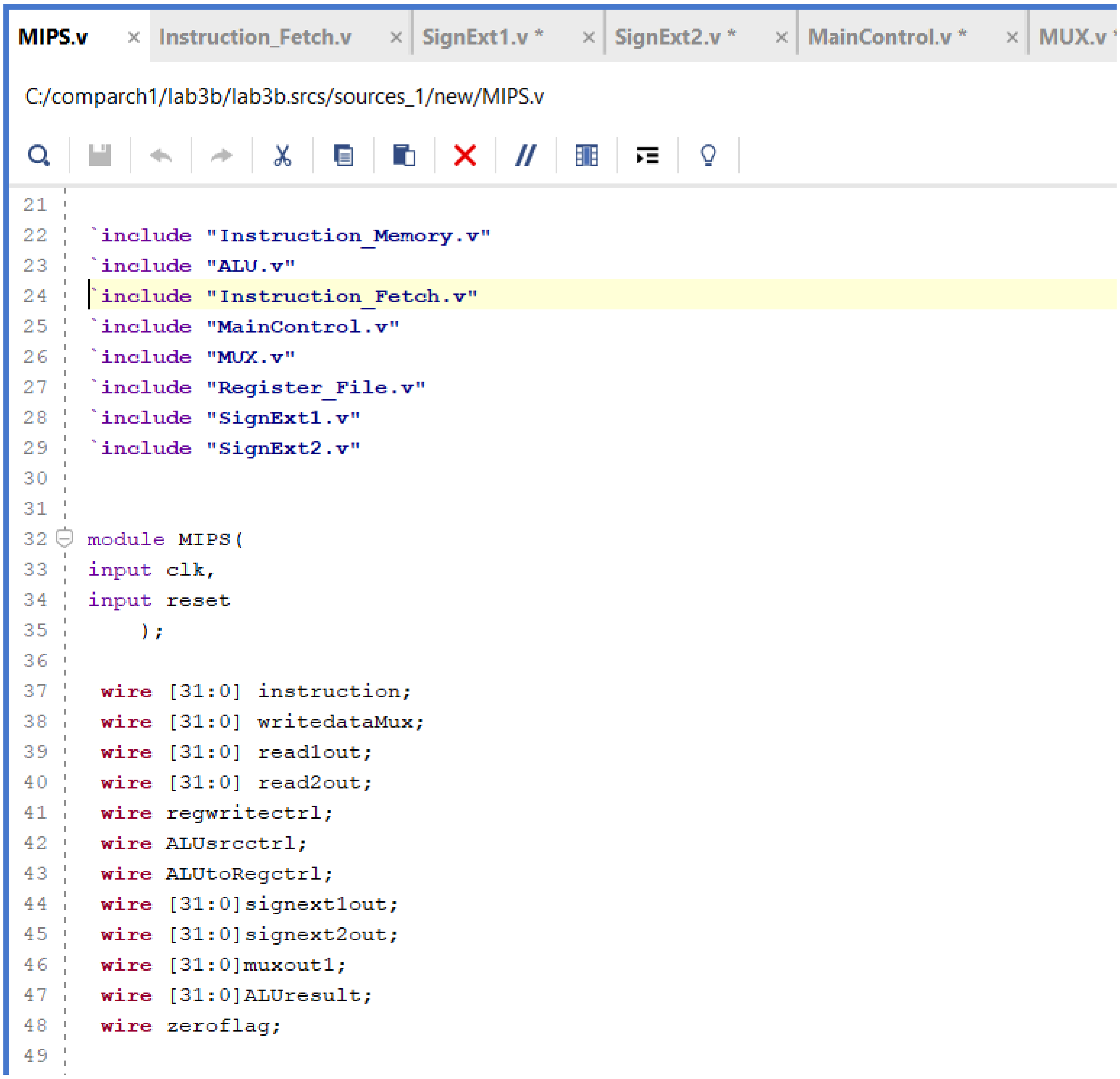


C:/comparch1/lab3b/lab3b.srcs/sources_1/new/MainControl.v

```verilog
`timescale 1ns / 1ps
module MainControl(
input [31:0] instr,
input clk,
output reg RegWrite,
output reg ALUSrc,
output reg ALUtoReg
    );
always@(posedge clk && instr)
begin
    RegWrite=1;
    if(instr[31]==0 && instr[5]==0)
    begin
         ALUSrc=0;
         ALUtoReg=1;
    end
    else if(instr[31]==0 && instr[5]==1)
    begin
         ALUSrc=1;
         ALUtoReg=1;
    end
    else if(instr[31] == 1'b1)
    begin
         ALUtoReg = 0;
         ALUSrc=1'bX;
    end
end
endmodule

```

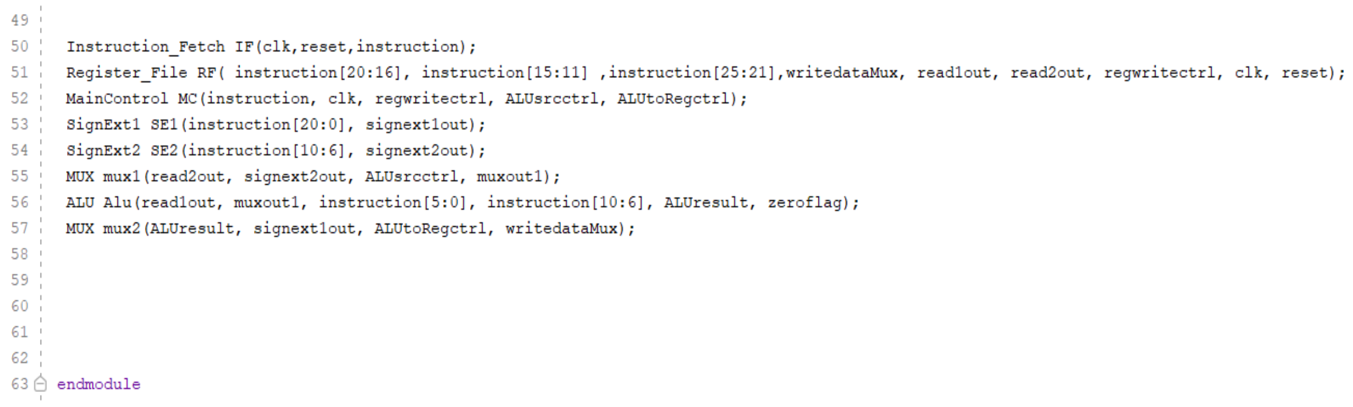
Q3.6. Implement a complete processor in Verilog (using all the datapath blocks and main control unit as modules). Copy the image of the Verilog code of the processor here.

Answer:



C:/comparch1/lab3b/lab3b.srcs/sources_1/new/MIPS.v

```verilog

 `include "Instruction_Memory.v"
 `include "ALU.v"
 `include "Instruction_Fetch.v"
 `include "MainControl.v"
 `include "MUX.v"
 `include "Register_File.v"
 `include "SignExt1.v"
 `include "SignExt2.v"


module MIPS(
input clk,
input reset
    );

 wire [31:0] instruction;
 wire [31:0] writedataMux;
 wire [31:0] read1out;
 wire [31:0] read2out;
 wire regwritectrl;
 wire ALUsrcctrl;
 wire ALUtoRegctrl;
 wire [31:0]signext1out;
 wire [31:0]signext2out;
 wire [31:0]muxout1;
 wire [31:0]ALUresult;
 wire zeroflag;

```



```
Instruction_Fetch IF(clk,reset,instruction);
Register_File RF( instruction[20:16], instruction[15:11] ,instruction[25:21],writedataMux, read1out, read2out, regwritectrl, clk, reset);
MainControl MC(instruction, clk, regwritectrl, ALUsrcctrl, ALUtoRegctrl);
SignExt1 SE1(instruction[20:0], signext1out);
SignExt2 SE2(instruction[10:6], signext2out);
MUX mux1(read2out, signext2out, ALUsrcctrl, muxout1);
ALU Alu(read1out, muxout1, instruction[5:0], instruction[10:6], ALUresult, zeroflag);
MUX mux2(ALUresult, signext1out, ALUtoRegctrl, writedataMux);





endmodule
```

Q3.7. Test the processor design by initializing the instruction memory with a set of instructions (at least 5 instructions). List below the instructions you have used to initialize the instruction memory. Verify if the register file is changing according to the instructions. (The register file contains unknowns, you can initialize the register file or you can load values into the register file using li instruction specified earlier).

Answer: **add r0, r1, r2 ,**

sub r4, r5, r6,

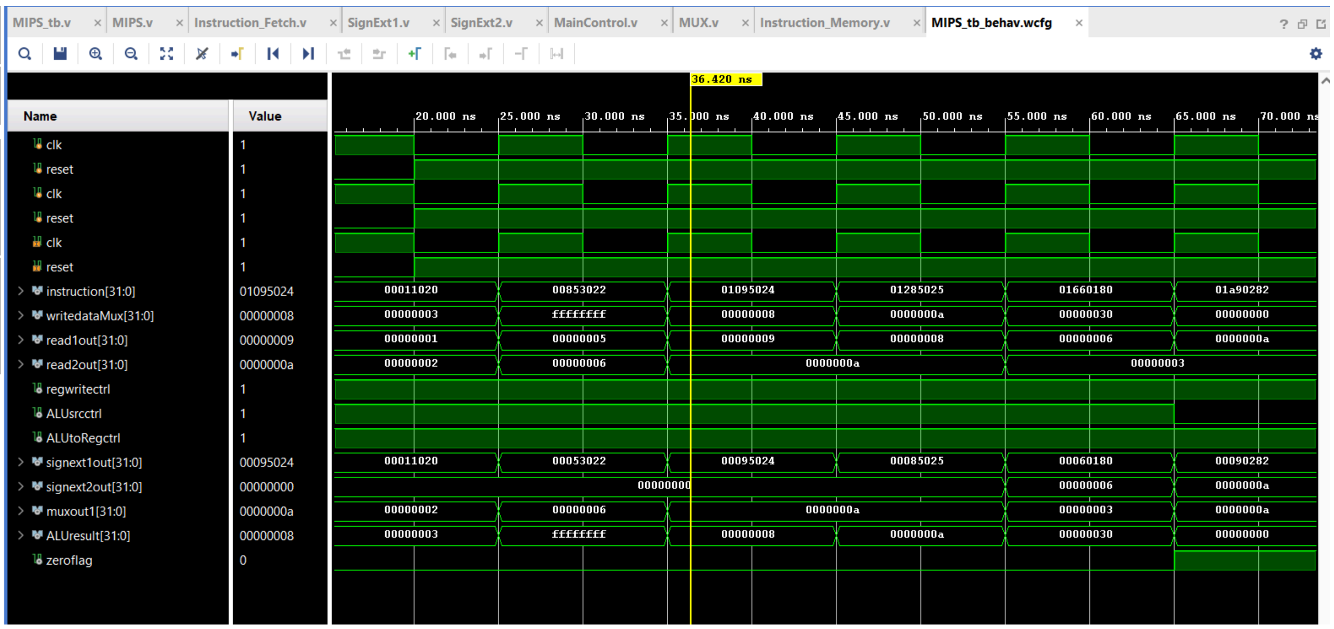
and r8, r9, r10,

sll r11, r6, 6,

srl r13, r9, 10

Q3.8. Once design, test, and verification are

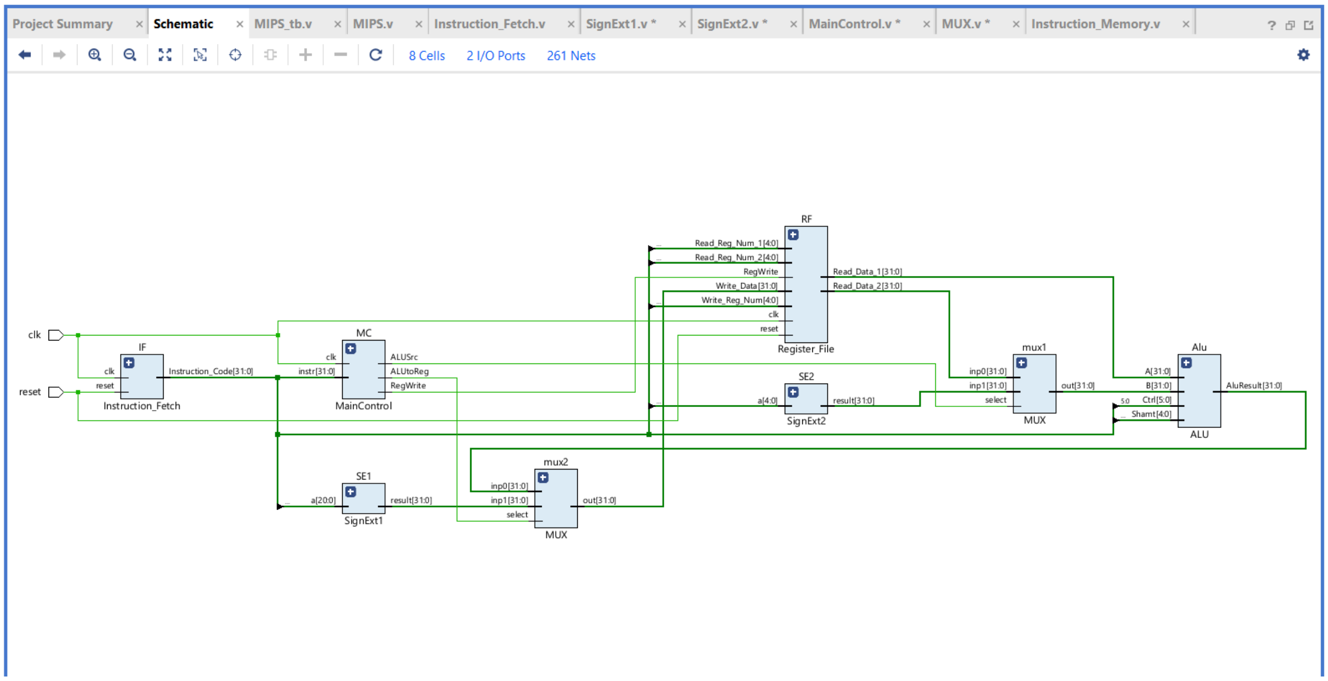
Copy verified **Register file** waveform here:



Name	Value
clk	1
reset	1
clk	1
reset	1
clk	1
reset	1
instruction[31:0]	01095024
writedataMux[31:0]	00000008
read1out[31:0]	00000009
read2out[31:0]	0000000a
regwritectrl	1
ALUsrcctrl	1
ALUtoRegctrl	1
signext1out[31:0]	00095024
signext2out[31:0]	00000000
muxout1[31:0]	0000000a
ALUresult[31:0]	00000008
zeroflag	0

```
instruction[31:0]    00011020  00853022  01095024  01285025  01660180  01a90282
writedataMux[31:0]   00000003  ffffffff  00000008  0000000a  00000030  00000000
read1out[31:0]       00000001  00000005  00000009  00000008  00000006  0000000a
read2out[31:0]       00000002  00000006  0000000a            00000003
signext1out[31:0]    00011020  00053022  00095024  00085025  00060180  00090282
signext2out[31:0]    00000000                                00000006  0000000a
muxout1[31:0]        00000002  00000006  0000000a            00000003  0000000a
ALUresult[31:0]      00000003  ffffffff  00000008  0000000a  00000030  00000000
```

Q3.9. Synthesise the processor designed and copy the RTL generated below


Project Summary
Schematic
MIPS_tb.v
MIPS.v
Instruction_Fetch.v
SignExt1.v *
SignExt2.v *
MainControl.v *
MUX.v *
Instruction_Memory.v
8 Cells
2 I/O Ports
261 Nets
clk
reset
IF
clk
reset
Instruction_Code[31:0]
Instruction_Fetch
MC
clk
instr[31:0]
ALUSrc
ALUtoReg
RegWrite
MainControl
RF
Read_Reg_Num_1[4:0]
Read_Reg_Num_2[4:0]
RegWrite
Write_Data[31:0]
Write_Reg_Num[4:0]
clk
reset
Read_Data_1[31:0]
Read_Data_2[31:0]
Register_File
SE2
a[4:0]
result[31:0]
SignExt2
mux1
inp0[31:0]
inp1[31:0]
select
out[31:0]
MUX
Alu
A[31:0]
B[31:0]
Ctrl[5:0]
Shamt[4:0]
AluResult[31:0]
ALU
SE1
a[20:0]
result[31:0]
SignExt1
mux2
inp0[31:0]
inp1[31:0]
select
out[31:0]
MUX