# Assignment -3(Computer Architecture Lab)

**Group no.** 19

**Name and ID Student 1:** Tanvi Behera 2019B5A80989H

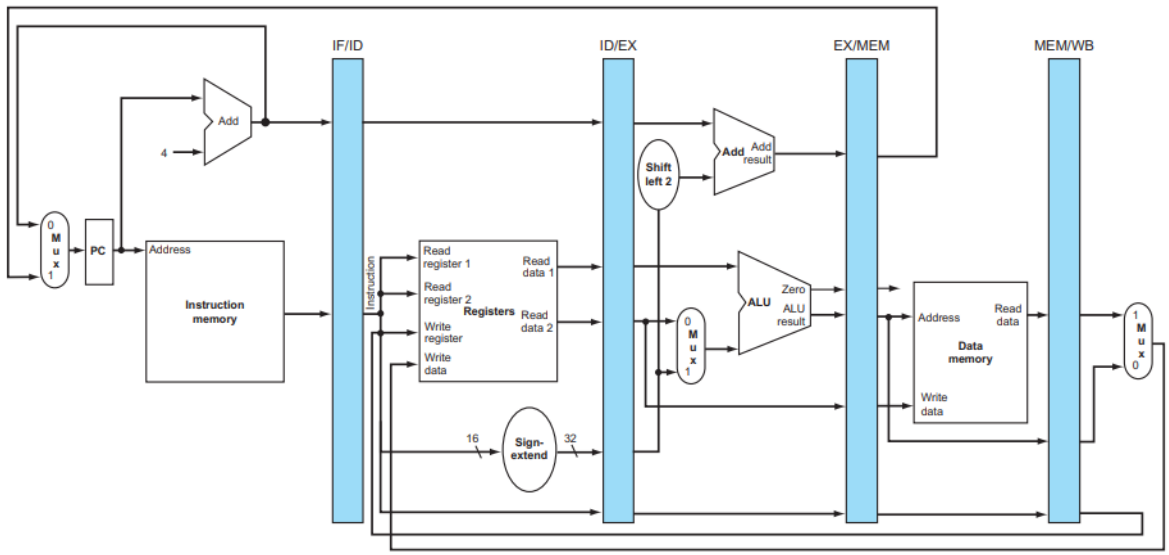
**Name and ID Student 2:** Adithya K 2019B4A80926H

**Name and ID Student 3:** Ayaz Hussain 2019B5A81108H

Implement a pipeline-based MIPS processor in Verilog that can execute at least 12 instructions, including R-type, I-type, and J-type (conditional and unconditional). The instructions should be chosen so that the three hazards, namely data, structural, and control hazards, should arise, and these issues should be resolved using techniques such as stalling, flushing, and forwarding based on the optimal solution.

**Part-A**

## Block Diagram




## Truth Table of Control Unit

| Instruction | Execution/address calculation stage control lines | | | | Memory access stage control lines | | | Write-back stage control lines | |
|---|---|---|---|---|---|---|---|---|---|
| | RegDst | ALUOp1 | ALUOp0 | ALUSrc | Branch | Mem-Read | Mem-Write | Reg-Write | Memto-Reg |
| R-format | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| lw | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| sw | X | 0 | 0 | 1 | 0 | 0 | 1 | 0 | X |
| beq | X | 0 | 1 | 0 | 1 | 0 | 0 | 0 | X |

| Instruction opcode | ALUOp | Instruction operation | Function code | Desired ALU action | ALU control input |
|---|---|---|---|---|---|
| LW | 00 | load word | XXXXXX | add | 0010 |
| SW | 00 | store word | XXXXXX | add | 0010 |
| Branch equal | 01 | branch equal | XXXXXX | subtract | 0110 |
| R-type | 10 | add | 100000 | add | 0010 |
| R-type | 10 | subtract | 100010 | subtract | 0110 |
| R-type | 10 | AND | 100100 | AND | 0000 |
| R-type | 10 | OR | 100101 | OR | 0001 |
| R-type | 10 | set on less than | 101010 | set on less than | 0111 |

# CODES

### Address_Cal

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 17:57:20
// Design Name:
// Module Name: Address_Cal
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Address_Cal(
input [31:0] Instruction_code,
  input IF_Flush,
  input  [31:0] PC,
  output reg [31:0] X,
  output reg [1:0] PCsrc
 );
 wire [19:0] Imm={Instruction_code[31],Instruction_code[19:12],Instruction_code[20],Instruction_code[30:21]};
    wire [19:0] Shft_2=Imm<<2;
    wire [31:0] J_addr={{12{Shft_2[19]}},Shft_2};
    always@(Instruction_code,IF_Flush,PC,J_addr)
    begin
    if (IF_Flush==0)
    PCsrc=2'b10;
    else if (Instruction_code[6:0]==7'b1101111)
    begin
    X<=J_addr+PC;
    PCsrc=2'b01;
    end
    else
    PCsrc=2'b0;
    end
    endmodule
```

---

## ALU

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 17:59:35
// Design Name:
// Module Name: ALU
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module ALU(
   input [31:0] A1,
   input [31:0] B2,
   input [2:0] ALU_Op,
   input [4:0] Temp1,
   output reg Zero,
   output reg [31:0] ALU_Result
   );
   reg [31:0] flag;
   wire [2:0] temp;
   assign temp=Temp1[2:0];
   always@(A1,B2,Temp1,ALU_Op)
   begin
   if(ALU_Op==3'b011)
   begin
   case(Temp1)
   5'b00000: ALU_Result=A1+B2;          //ADD
   5'b01000: ALU_Result=A1-B2;           //SUB
   5'b00110: ALU_Result=A1|B2;           //OR
   5'b00111: ALU_Result=A1&B2;           //AND
   5'b00010: begin
             flag=A1-B2;                 //SLT
             ALU_Result[31:1]=31'h0;
             if (flag[31]==1'b1)
             ALU_Result[0]=1'b1;
             else
             ALU_Result[0]=1'b0;
             end
    5'b00100: ALU_Result=A1^B2;
    5'b00001: ALU_Result=A1<<B2;  //shift left
    5'b00101: ALU_Result=A1>>B2;//shift right
   endcase
   end
```

```
    end
    else if(ALU_Op==3'b001)
    begin
    case(temp)
    3'b000: ALU_Result=A1+B2;          //ADD
    3'b110: ALU_Result=A1|B2;           //OR
    3'b111: ALU_Result=A1&B2;
    3'b001: ALU_Result=A1+B2;
   endcase
   end
   else if((ALU_Op==3'b000)||(ALU_Op==3'b010))
   begin
   ALU_Result=A1+B2;
   end
   end
    always@(A1,B2,ALU_Op,Temp1)
    begin
    if (ALU_Result==32'b0)
        Zero=1;
    else
        Zero=0;
    end


endmodule
```

---

**Branch_cal**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 18:01:17
// Design Name:
// Module Name: Branch_Cal
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

    module Branch_Cal(
   input [31:0] Out1,
   input [31:0] SgnEx,
   output reg [31:0] B_Addr
    );
    always@(Out1,SgnEx)
    begin
    B_Addr={Out1+SgnEx};
    end
endmodule
```

---

**Branch_Comparator**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 18:03:15
// Design Name:
// Module Name: Branch_Comparator
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Branch_Comparator(
    input [31:0] Read_Data1,
    input [31:0] Read_Data2,
    input [6:0] Opcode,
    output reg IF_Flush
    );

    always@(Read_Data1,Read_Data2,Opcode)
    begin
    if ((Read_Data1==Read_Data2)&&(Opcode==7'b1100011))
    begin
    IF_Flush=0;
    end
    else
    begin
    IF_Flush=1;
    end
    end
endmodule
```

---

**Data_Memory**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 18:04:33
// Design Name:
// Module Name: Data_Memory
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


 module Data_Memory(
  input Reset,
  input [31:0] Y1,
  input [31:0] Data3,
  input MemRead2,
  input MemWrite2,
  output reg [31:0] Read_Data
    );
         reg [7:0] Mem [48:0];

          always@(MemRead2,MemWrite2,Y1,Data3)
          begin
          if (MemRead2==1)
          Read_Data={Mem[Y1+3],Mem[Y1+2],Mem[Y1+1],Mem[Y1]};
          if (MemWrite2==1)
          {Mem[Y1+3],Mem[Y1+2],Mem[Y1+1],Mem[Y1]}=Data3;
          end

         always@(negedge Reset)
         begin
         if (Reset==0)
         begin
         // 12 Instructions
       Mem[0]=8'b00010000;Mem[1]=8'b00000001;Mem[2]=8'b0;Mem[3]=8'b0;
       Mem[4]=8'b00010001;Mem[5]=8'b00000001;Mem[6]=8'b0;Mem[7]=8'b0;
       Mem[8]=8'b00010010;Mem[9]=8'b00000001;Mem[10]=8'b0;Mem[11]=8'b0;
       Mem[12]=8'b00010011;Mem[13]=8'b00000001;Mem[14]=8'b0;Mem[15]=8'b0;
       Mem[16]=8'b00010100;Mem[17]=8'b00000001;Mem[18]=8'b0;Mem[19]=8'b0;
       Mem[20]=8'b00010101;Mem[21]=8'b00000001;Mem[22]=8'b0;Mem[23]=8'b0;
       Mem[24]=8'b00010110;Mem[25]=8'b00000001;Mem[26]=8'b0;Mem[27]=8'b0;
       Mem[28]=8'b00010111;Mem[29]=8'b00000001;Mem[30]=8'b0;Mem[31]=8'b0;
       Mem[32]=8'b00011000;Mem[33]=8'b00000001;Mem[34]=8'b0;Mem[35]=8'b0;
       Mem[36]=8'b00011001;Mem[37]=8'b00000001;Mem[38]=8'b0;Mem[39]=8'b0;
       Mem[40]=8'b00011010;Mem[41]=8'b00000001;Mem[42]=8'b0;Mem[43]=8'b0;
       Mem[44]=8'b00011011;Mem[45]=8'b00000001;Mem[46]=8'b0;Mem[47]=8'b0;
       Mem[48]=8'b00011100;Mem[49]=8'b00000001;Mem[50]=8'b0;Mem[51]=8'b0;
       Mem[52]=8'b00011101;Mem[53]=8'b00000001;Mem[54]=8'b0;Mem[55]=8'b0;
       Mem[56]=8'b00011110;Mem[57]=8'b00000001;Mem[58]=8'b0;Mem[59]=8'b0;
       Mem[60]=8'b00011111;Mem[61]=8'b00000001;Mem[62]=8'b0;Mem[63]=8'b0;
       end
       end
endmodule
```

---

**EX_Mem_Reg**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 18:08:46
// Design Name:
// Module Name: EX_Mem_Reg
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module EX_Mem_Reg(
    input Clk,
    input Reset,
    input [31:0] ALU_Result,
    input [4:0] Dest_Reg_Num1,
    input [2:0] ALU_Op,
    input RegWrite1,
    input [31:0] B1,
    input MemRead1,
    input MemWrite1,
    input Mem2Reg1,
    output reg [31:0] Y1,
    output reg [4:0] Dest_Reg_Num2,
    output reg [2:0] ALU_Op1,
    output reg RegWrite2,
    output reg [31:0] Data3,
    output reg MemRead2,
    output reg MemWrite2,
    output reg Mem2Reg2
    );

     always@(posedge Clk,negedge Reset)
      begin
           if (Reset==0)
           begin
           Y1<=0;
           Dest_Reg_Num2<=0;
           RegWrite2<=0;
           ALU_Op1<=0;
           Data3<=0;
           MemRead2<=0;
           MemWrite2<=0;
           Mem2Reg2<=0;
           end
```

```
            else
            begin
            Y1<=ALU_Result;
            Dest_Reg_Num2<=Dest_Reg_Num1;
            RegWrite2<=RegWrite1;
            ALU_Op1<=ALU_Op;
            Data3<=B1;
            MemRead2<=MemRead1;
            MemWrite2<=MemWrite1;
            Mem2Reg2<=Mem2Reg1;
            end
        end
endmodule

```

---

**Forward_Unit**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 18:10:28
// Design Name:
// Module Name: Forward_Unit
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module Forward_Unit(
    input [4:0] Write_Reg_Num,
    input [4:0] Dest_Reg_Num2,
    input RegWrite2,
    input RegWrite3,
    input [4:0] Read_Reg_Num1,
    input [4:0] Read_Reg_Num2,
    input [4:0] Num1,
    input [4:0] Num2,
    input [2:0] ALU_Opcode,
    input MemRead2,
    input MemRead3,
    output reg [1:0] Fwd_Rs,
    output reg [1:0] Fwd_Rd,
    output reg [1:0] Fwd_Rs1,
    output reg [1:0] Fwd_Rd1
    );

    always@(Dest_Reg_Num2,Write_Reg_Num,RegWrite2,RegWrite3,Read_Reg_Num1,Read_Reg_Num2,Num1,Num2,MemRead3,ALU_Opcode,MemRead2)
    begin
     if ((RegWrite2==1)&&(Dest_Reg_Num2==Num1))
        Fwd_Rs=2'b01;
     else if ((RegWrite3==1)&&(Write_Reg_Num==Num1))
        Fwd_Rs=2'b10;
     else
        Fwd_Rs=2'b0;

     if((RegWrite2==1)&&(Dest_Reg_Num2==Num2))
        Fwd_Rd=2'b01;
     else if ((RegWrite3==1)&&(Write_Reg_Num==Num2))
         Fwd_Rd=2'b10;
     else
         Fwd_Rd=2'b0;
```

```
56
57        if((MemRead3==1)&&(ALU_Opcode==3'b110)&&(Write_Reg_Num==Read_Reg_Num1))
58        Fwd_Rs1=2'b10;
59        else if((ALU_Opcode==3'b110)&&(RegWrite2==1)&&(Dest_Reg_Num2==Read_Reg_Num1))
60        Fwd_Rs1=2'b01;
61        else
62        Fwd_Rs1=2'b0;
63
64         if((MemRead3==1)&&(ALU_Opcode==3'b110)&&(Write_Reg_Num==Read_Reg_Num2))
65         Fwd_Rd1=2'b10;
66         else if((ALU_Opcode==3'b110)&&(RegWrite2==1)&&(Dest_Reg_Num2==Read_Reg_Num2))
67         Fwd_Rd1=2'b01;
68         else
69         Fwd_Rd1=2'b0;
70
71         end
72  endmodule
```

---

**ID_EX_Reg**

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   // Company:
4   // Engineer:
5   //
6   // Create Date: 17.04.2023 18:12:02
7   // Design Name:
8   // Module Name: ID_EX_Reg
9   // Project Name:
10  // Target Devices:
11  // Tool Versions:
12  // Description:
13  //
14  // Dependencies:
15  //
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////////////////////
21
22
23  module ID_EX_Reg(
24         input Clk,
25         input Reset,
26         input [31:0] Read_Data1,
27         input [31:0] Read_Data2,
28         input [4:0] Dest_Reg_Num,
29         input [31:0] SignEx,
30         input [4:0] Read_Reg_Num1,
31         input [4:0] Read_Reg_Num2,
32         input [2:0] ALU_Opcode,
33         input ALUsrc,
34         input RegWrite,
35         input [4:0] Temp,
36         input MemRead,
37         input MemWrite,
38         input Mem2Reg,
39         output reg [31:0] Data1,
40         output reg [31:0] Data2,
41         output reg [31:0] Sign_Ex1,
42         output reg [4:0] Dest_Reg_Num1,
43         output reg [2:0] ALU_Op,
44         output reg ALUsrc1,
45         output reg RegWrite1,
46         output reg [4:0] Num1,
47         output reg [4:0] Num2,
48         output reg [4:0] Temp1,
49         output reg MemRead1,
50         output reg MemWrite1,
51         output reg Mem2Reg1
52      );
```

```
53
54      always@(posedge Clk,negedge Reset)
55      begin
56           if (Reset==0)
57           begin
58           Data1=0;
59           Data2=0;
60           Dest_Reg_Num1=0;
61           Sign_Ex1=0;
62           ALU_Op=0;
63           ALUsrc1=0;
64           RegWrite1=0;
65           Num1=0;
66           Num2=0;
67           Temp1=0;
68           MemRead1=0;
69           MemWrite1=0;
70           Mem2Reg1=0;
71           end
72           else
73           begin
74           Data1<=Read_Data1;
75           Data2<=Read_Data2;
76           Dest_Reg_Num1<=Dest_Reg_Num;
77           Sign_Ex1<=SignEx;
78           ALU_Op<=ALU_Opcode;
79           ALUsrc1<=ALUsrc;
80           RegWrite1<=RegWrite;
81           Num1<=Read_Reg_Num1;
82           Num2<=Read_Reg_Num2;
83           Temp1<=Temp;
84           MemRead1<=MemRead;
85           MemWrite1<=MemWrite;
86           Mem2Reg1<=Mem2Reg;
87           end
88      end
89  endmodule
```

---

**IF_ID_Reg**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 18:14:16
// Design Name:
// Module Name: IF_ID_Reg
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module IF_ID_Reg(
        input Clk,
        input Reset,
        input [31:0] Instruction_code,
        input IF_Write,
        input [31:0] PC,
        input IF_Flush,
        output reg [31:0] Instruction_code1,
        output reg [4:0] Read_Reg_Num1,
        output reg [4:0] Read_Reg_Num2,
        output reg [4:0] Dest_Reg_Num,
        output reg [11:0] Immediate1,
        output reg [11:0] Immediate2,
        output reg [11:0] Immediate3,
        output reg [6:0] Opcode,
        output reg [2:0] F3,
        output reg [1:0] F7,
        output reg [4:0] Temp,
        output reg [31:0] Out1
    );
```

```
    always@(posedge Clk,negedge Reset,negedge IF_Write,negedge IF_Flush)
    begin
        if (Reset==0)
        begin
        Instruction_code1<=0;
        Read_Reg_Num1<=0;
        Read_Reg_Num2<=0;
        Dest_Reg_Num<=0;
        Immediate1<=0;
        Immediate2<=0;
        Immediate3<=0;
        Opcode<=0;
        F3<=0;
        F7<=0;
        Temp<=0;
        Out1<=0;
        end
        else
        begin
        if(IF_Write==0)
        begin
        Instruction_code1<=Instruction_code1;
        Opcode=Instruction_code1[6:0];
        end
        else if(IF_Flush==0)
        begin
        Instruction_code1<=32'h0;
        Opcode=Instruction_code1[6:0];
        end
        else
        begin
        Instruction_code1=Instruction_code;
        Read_Reg_Num1=Instruction_code1[19:15];
        Read_Reg_Num2=Instruction_code1[24:20];
        Dest_Reg_Num=Instruction_code1[11:7];
        Immediate1=Instruction_code1[31:20];
        Immediate2={Instruction_code1[31:25],Instruction_code1[11:7]};
        Immediate3={Instruction_code1[31],Instruction_code1[7],Instruction_code1[30:25],Instruction_code1[11:8]};
        Opcode=Instruction_code1[6:0];
        F3=Instruction_code1[14:12];
        F7=Instruction_code1[31:30];
        Temp={F7,F3};
        Out1=PC;
        end
        end
    end
endmodule
```

---

## Instruction_Memory

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 18:18:08
// Design Name:
// Module Name: Instruction_Memory
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Instruction_Memory(
   input [31:0] PC,
   input Reset,
   output [31:0] Instruction_code
    );

    reg [7:0] Mem [84:0];
    assign Instruction_code={Mem[PC+3],Mem[PC+2],Mem[PC+1],Mem[PC]}; //Little-Endian

    always@(Reset)
    begin
    if (Reset==0)
    begin
    //12 Instructions
      Mem[0]=8'hB3;Mem[1]=8'h0;Mem[2]=8'h31;Mem[3]=8'h0;         //ADD
      Mem[4]=8'h33;Mem[5]=8'h82;Mem[6]=8'h62;Mem[7]=8'h40;       //SUB
      Mem[8]=8'hB3;Mem[9]=8'h73;Mem[10]=8'h31;Mem[11]=8'h0;      //AND
      Mem[12]=8'h33;Mem[13]=8'h64;Mem[14]=8'h31;Mem[15]=8'h0;    //OR
      Mem[16]=8'hB3;Mem[17]=8'h14;Mem[18]=8'h31;Mem[19]=8'h0;    //SHIFT_LEFT
      Mem[20]=8'h33;Mem[21]=8'hD5;Mem[22]=8'h62;Mem[23]=8'h0;    //SHIFT_RIGHT
      Mem[24]=8'h13;Mem[25]=8'h06;Mem[26]=8'h51;Mem[27]=8'h0;    //Add Immediate
      Mem[28]=8'h83;Mem[29]=8'h21;Mem[30]=8'h81;Mem[31]=8'h0;    //Load
      Mem[32]=8'h23;Mem[33]=8'h22;Mem[34]=8'h71;Mem[35]=8'h0;    //Store
      Mem[36]=8'h63;Mem[37]=8'h04;Mem[38]=8'hB1;Mem[39]=8'h0;    //Beq
      Mem[40]=8'hB3;Mem[41]=8'h0;Mem[42]=8'h31;Mem[43]=8'h0;     //Add
      Mem[44]=8'h33;Mem[45]=8'h82;Mem[46]=8'h62;Mem[47]=8'h40;   //Sub
      Mem[48]=8'hB3;Mem[49]=8'h73;Mem[50]=8'h31;Mem[51]=8'h0;    //AND
      Mem[52]=8'h93;Mem[53]=8'h62;Mem[54]=8'hA1;Mem[55]=8'h0;    //OR Immediate
      Mem[56]=8'h6f;Mem[57]=8'h0;Mem[58]=8'h60;Mem[59]=8'h0;     //jal
      Mem[60]=8'hb3;Mem[61]=8'h0;Mem[62]=8'h31;Mem[63]=8'h0;     //add
      Mem[64]=8'h33;Mem[65]=8'h82;Mem[66]=8'h62;Mem[67]=8'h40;   //sub
      Mem[68]=8'hB3;Mem[69]=8'h0;Mem[70]=8'h31;Mem[71]=8'h0;     //Add
    end
    end
endmodule
```

---

**Main_Control**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 18:20:04
// Design Name:
// Module Name: Main_Control
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Main_Control(
    input [6:0] Opcode,
    input Cntsrc,
    output [2:0] ALU_Opcode,
    output reg ALUsrc,
    output reg RegWrite,
    output reg MemRead,
    output reg MemWrite,
    output reg Mem2Reg,
    output reg Immsrc
    );

    assign ALU_Opcode=Opcode[6:4];
    always@(Opcode,Cntsrc)
    begin
    if(Cntsrc==0)
    begin
    RegWrite<=1'b0;
    MemRead<=1'b0;
    MemWrite<=1'b0;
    Mem2Reg<=1'b0;
    Immsrc=0;
    ALUsrc=0;
    end
```

```
    else
    begin
    case(Opcode)
    7'b0: begin
          RegWrite<=1'b0;
          ALUsrc<=1'b0;
          MemRead<=1'b0;
          MemWrite<=1'b0;
          Mem2Reg<=1'b0;
          Immsrc=0;
          end

    7'b0110011: begin
                RegWrite<=1'b1;
                ALUsrc<=1'b0;
                MemRead<=1'b0;
                MemWrite<=1'b0;
                Mem2Reg<=1'b0;
                Immsrc<=0;
                end

    7'b0010011: begin
                RegWrite<=1'b1;
                ALUsrc<=1'b1;
                MemRead<=1'b0;
                MemWrite<=1'b0;
                Mem2Reg<=1'b0;
                Immsrc<=0;
                end

    7'b0000011: begin
                RegWrite<=1'b1;
                ALUsrc<=1'b1;
                MemRead<=1'b1;
                MemWrite<=1'b0;
                Mem2Reg<=1'b1;
                Immsrc<=0;
                end

    7'b0100011: begin
                RegWrite<=1'b0;
                ALUsrc<=1'b1;
                MemRead<=1'b0;
                MemWrite<=1'b1;
                Mem2Reg<=1'b0;
                Immsrc<=2'b01;
                end
     7'b0100011: begin
                 RegWrite<=1'b0;
                 ALUsrc<=1'b0;
                 MemRead<=1'b0;
                 MemWrite<=1'b0;
                 Mem2Reg<=1'b0;
                 Immsrc<=0;
                 end
```

```
103
104         7'b1101111: begin
105                         RegWrite<=1'b0;
106                         ALUsrc<=1'b0;
107                         MemRead<=1'b0;
108                         MemWrite<=1'b0;
109                         Mem2Reg<=1'b0;
110                         Immsrc<=0;
111                         end
112             7'b1100011: begin
113                         RegWrite<=1'b0;
114                         ALUsrc<=1'b0;
115                         MemRead<=1'b0;
116                         MemWrite<=1'b0;
117                         Mem2Reg<=1'b0;
118                         Immsrc<=0;
119                         end
120
121         endcase
122         end
123         end
124         endmodule
```

---

## Mem_WB_Reg

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   // Company:
4   // Engineer:
5   //
6   // Create Date: 17.04.2023 18:22:03
7   // Design Name:
8   // Module Name: Mem_WB_Reg
9   // Project Name:
10  // Target Devices:
11  // Tool Versions:
12  // Description:
13  //
14  // Dependencies:
15  //
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////////////////////
21
22
23  module Mem_WB_Reg(
24     input Clk,
25     input Reset,
26     input [31:0] Read_Data,
27     input [4:0] Dest_Reg_Num2,
28     input [31:0] Y1,
29     input Mem2Reg2,
30     input RegWrite2,
31     output reg [31:0] W,
32     output reg [4:0] Write_Reg_Num,
33     output reg [31:0] Y2,
34     output reg Mem2Reg3,
35     output reg RegWrite3
36      );
37      always@(posedge Clk,negedge Reset)
38      begin
39      if (Reset==0)
40      begin
41      Y2<=0;
42      Write_Reg_Num<=0;
43      W<=0;
44      Mem2Reg3<=0;
45      RegWrite3<=0;
46      end
47      else
48      begin
49      Y2<=Y1;
50      Write_Reg_Num<=Dest_Reg_Num2;
51      W<=Read_Data;
52      Mem2Reg3<=Mem2Reg2;
53      RegWrite3<=RegWrite2;
54      end
55      end
56  endmodule
```

Mux_2x1

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 18:23:28
// Design Name:
// Module Name: Mux_2x1
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Mux_2x1(
    input [31:0] B1,
    input [31:0] Sign_Ex1,
    input ALUsrc1,
    output reg [31:0] B2
    );

   always@(B1,Sign_Ex1,ALUsrc1)
   begin
   case (ALUsrc1)
   1'b0 : B2=B1;
   1'b1 : B2=Sign_Ex1;
   endcase
   end
endmodule
```

---

**Mux_Rd1_3x1**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 18:24:43
// Design Name:
// Module Name: Mux_Rd1_3x1
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module Mux_Rd1_3x1(
    input [31:0] Read_2,
    input [31:0] Y1,
    input [31:0] W,
    input [1:0] Fwd_Rd1,
    output reg [31:0] Read_Data2
    );

   always@(Read_2,Y1,Fwd_Rd1,W)
   begin
   case (Fwd_Rd1)
   2'b00: Read_Data2=Read_2;
   2'b01: Read_Data2=Y1;
   2'b10: Read_Data2=W;
   endcase
   end
endmodule
```

---

**Mux_Rs1_3x1**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 23:38:36
// Design Name:
// Module Name: Mux_Rs1_3x1
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Mux_Rs1_3x1(
   input [31:0] Read_1,
   input [31:0] Y1,
   input [31:0] W,
   input [1:0] Fwd_Rs1,
   output reg [31:0] Read_Data1
   );

  always@(Read_1,Y1,Fwd_Rs1,W)
  begin
  case (Fwd_Rs1)
  2'b00: Read_Data1=Read_1;
  2'b01: Read_Data1=Y1;
  2'b10: Read_Data1=W;
  endcase
  end
endmodule
```

---

**MuxA_3x1MuxA_3x1**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 23:49:39
// Design Name:
// Module Name: MuxA_3x1MuxA_3x1
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module MuxA_3x1(
    input [31:0] Data1,
    input [31:0] Y1,
    input [31:0] Write_Data,
    input [1:0] Fwd_Rs,
    output reg [31:0] A1
    );

   always@(Data1,Y1,Fwd_Rs,Write_Data)
   begin
   case (Fwd_Rs)
   2'b00: A1=Data1;
   2'b01: A1=Y1;
   2'b10: A1=Write_Data;
   endcase
   end
endmodule
```

---

**MuxB_3x1**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 23:51:29
// Design Name:
// Module Name: MuxB_3x1
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module MuxB_3x1(
   input [31:0] Data2,
   input [31:0] Y1,
   input [1:0] Fwd_Rd,
   input [31:0] Write_Data,
   output reg [31:0] B1
   );

  always@(Data2,Y1,Fwd_Rd,Write_Data)
  begin
  case (Fwd_Rd)
    2'b00: B1=Data2;
    2'b01: B1=Y1;
    2'b10: B1=Write_Data;
  endcase
  end
endmodule
```

---

**MuxImm_2x1**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 18.04.2023 00:13:21
// Design Name:
// Module Name: MuxImm_2x1
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module MuxImm_2x1(
    input [11:0] Immediate1,
    input [11:0] Immediate2,
    input Immsrc,
    output reg [11:0] Immediate
    );

   always@(Immediate1,Immediate2,Immsrc)
   begin
   case (Immsrc)
   1'b0 : Immediate=Immediate1;
   1'b1 : Immediate=Immediate2;
   endcase
   end

endmodule
```

---

**MuxMem_2x1**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 23:53:05
// Design Name:
// Module Name: MuxMem_2x1
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module MuxMem_2x1(
    input [31:0] W,
    input [31:0] Y2,
    input Mem2Reg3,
    output reg [31:0] Write_Data
    );

   always@(W,Y2,Mem2Reg3)
   begin
   case (Mem2Reg3)
   1'b0 : Write_Data=Y2;
   1'b1 : Write_Data=W;
   endcase
   end
   endmodule
```

---

**MUXPC_2x1**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 23:55:18
// Design Name:
// Module Name: MUXPC_2x1
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////



module MUXPC_2x1(
    input [31:0] Out,
    input [31:0] X,
    input [31:0] B_Addr,
    input [1:0] PCsrc,
    output reg [31:0] Z
    );

   always@(Out,X,B_Addr,PCsrc)
   begin
   case (PCsrc)

         2'b00 : Z=Out;
         2'b01 : Z=X;
         2'b10 : Z=B_Addr;

    endcase
    end
endmodule
```

---

**PC_Adder**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 23:58:25
// Design Name:
// Module Name: PC_Adder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module PC_Adder(
   input [31:0] PC,
   output [31:0] Out
   );
   assign Out=PC+4;
endmodule
```

---

**PC**

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 17.04.2023 23:57:12
7  // Design Name:
8  // Module Name: PC
9  // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////////////////////////////////////
21
22
23 module PC(
24     input [31:0] Z,
25     input Reset,
26     input Clk,
27     input PC_Write,
28     output reg [31:0] PC
29    );
30
31    always@(posedge Clk,negedge Reset,negedge PC_Write)
32    begin
33    if (Reset==0)
34    PC=0;
35    else
36    begin
37    if (PC_Write==0)
38    PC<=PC;
39    else
40    PC=Z;
41   end
42    end
43 endmodule
```

---

**pipeline**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 17:51:28
// Design Name:
// Module Name: pipeline
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module pipeline(
   input Clk,
   input Reset
   );
      wire [31:0] Instruction_code;
      wire [31:0] Instruction_code1;
      wire [4:0] Temp;
      wire [4:0] Temp1;
      wire [31:0] PC;
      wire [4:0] Read_Reg_Num1;
      wire [4:0] Read_Reg_Num2;
      wire [4:0] Write_Reg_Num;
      wire [31:0] Read_Data1;
      wire [31:0] Read_Data2;
      wire [31:0] Read_Data;
      wire [31:0] Data1;
      wire [31:0] Data2;
      wire [31:0] Data3;
      wire [31:0] W;
      wire [4:0] Num1;
      wire [4:0] Num2;
      wire [31:0] A1;
      wire [31:0] B1;
      wire [31:0] B2;
      wire [31:0] Write_Data;
      wire [4:0] Dest_Reg_Num;
      wire [4:0] Dest_Reg_Num1;
      wire [6:0] Opcode;
      wire [11:0] Immediate;
      wire [11:0] Immediate1;
      wire [11:0] Immediate2;
      wire [11:0] Immediate3;
      wire [31:0] SgnEx;
      wire [31:0] SignEx;
```

```
55      wire [31:0] SgnEx;
56      wire [31:0] SignEx;
57      wire [31:0] Sign_Ex1;
58      wire [2:0] ALU_Opcode;
59      wire [2:0] ALU_Op;
60      wire [2:0] ALU_Op1;
61      wire ALUsrc;
62      wire ALUsrc1;
63      wire [1:0] PCsrc;
64      wire RegWrite;
65      wire RegWrite1;
66      wire RegWrite2;
67      wire RegWrite3;
68      wire [1:0] Fwd_Rs;
69      wire [1:0] Fwd_Rd;
70      wire [1:0] Fwd_Rs1;
71      wire [1:0] Fwd_Rd1;
72      wire Zero;
73      wire [31:0] ALU_Result;
74      wire [31:0] Y;
75      wire [31:0] X;
76      wire [31:0] Out;
77      wire [31:0] Out1;
78      wire [31:0] Z;
79      wire [2:0] F3;
80      wire [1:0] F7;
81      wire [31:0] Y1;
82      wire [31:0] Y2;
83      wire [4:0] Dest_Reg_Num2;
84      wire MemRead;
85      wire MemWrite;
86      wire MemRead1;
87      wire MemWrite1;
88      wire MemRead2;
89      wire MemWrite2;
90      wire Mem2Reg;
91      wire Mem2Reg1;
92      wire Mem2Reg2;
93      wire Mem2Reg3;
94      wire Immsrc;
95      wire Cntsrc;
96      wire PC_Write;
97      wire IF_Write;
98      wire IF_Flush;
99      wire [11:0] Shift_2;
100     wire [31:0] B_Addr;
101     wire Flag;
102     wire [31:0] Read_1;
103     wire [31:0] Read_2;
104
105     MUXPC_2x1 M1(Out,X,B_Addr,PCsrc,Z);                          //Selecting Whether PC+4 or Branch Address or Jump Address
106     PC A15(Z,Reset,Clk,PC_Write,PC);                             //Updating of address of PC according to Instruction
107     Instruction_Memory A14(PC,Reset,Instruction_code);           //Gives Instruction_code as output for processor implementation
108     Address_Cal A13(Instruction_code,IF_Flush,PC,X,PCsrc);       //For giving value for PCsrc and Jump address
109     PC_Adder A16(PC,Out);                                        //Calculate next address of PC i.e PC+4
110     Stall_Unit A44(MemRead1,MemRead2,Dest_Reg_Num1,Dest_Reg_Num2,Read_Reg_Num1,Read_Reg_Num2,RegWrite1,ALU_Opcode,Cntsrc,PC_Write,IF_Write);
```

```
111   IF_ID_Reg F2(Clk,Reset,Instruction_code,IF_Write,PC,IF_Flush,Instruction_code1,Read_Reg_Num1,Read_Reg_Num2,Dest_Reg_Num,Immediate1,Immediate2,Immediate3,Opcode
112   Main_Control A3(Opcode,Cntsrc,ALU_Opcode,ALUsrc,RegWrite,MemRead,MemWrite,Mem2Reg,Immsrc);      //Control signals for MUX,ALU,Data Memory
113   MuxImm_2x1 A33(Immediate1,Immediate2,Immsrc,Immediate);
114   Sign_Ex A4(Immediate,Immediate3,SgnEx,SignEx);                          //For extending to 32-bit
115   Reg_File A5(Reset,Read_Reg_Num1,Read_Reg_Num2,Write_Reg_Num,Write_Data,RegWrite3,Read_1,Read_2);
116   Mux_Rs1_3x1 A68(Read_1,Y1,W,Fwd_Rs1,Read_Data1);
117   Mux_Rd1_3x1 A67(Read_2,Y1,W,Fwd_Rd1,Read_Data2);
118   ID_EX_Reg A6(Clk,Reset,Read_Data1,Read_Data2,Dest_Reg_Num,SignEx,Read_Reg_Num1,Read_Reg_Num2,ALU_Opcode,ALUsrc,RegWrite,Temp,MemRead,MemWrite,Mem2Reg,Data1,Dat
119   ALU A7(A1,B2,ALU_Op,Temp1,Zero,ALU_Result);
120   Mux_2x1 A8(B1,Sign_Ex1,ALUsrc1,B2);                                     // Select between Addition and Immediate Instruction
121   Forward_Unit A10(Write_Reg_Num,Dest_Reg_Num2,RegWrite2,RegWrite3,Read_Reg_Num1,Read_Reg_Num2,Num1,Num2,ALU_Opcode,MemRead2,MemRead3,Fwd_Rs,Fwd_Rd,Fwd_Rs1,Fwd_R
122   MuxA_3x1 A11(Data1,Y1,Write_Data,Fwd_Rs,A1);                            //Select whether forwarded Data or Nominal data
123   MuxB_3x1 A12(Data2,Y1,Fwd_Rd,Write_Data,B1);                            //Select whether forwarded Data or Nominal data
124   EX_Mem_Reg A20(Clk,Reset,ALU_Result,Dest_Reg_Num1,ALU_Op,RegWrite1,B1,MemRead1,MemWrite1,Mem2Reg1,Y1,Dest_Reg_Num2,ALU_Op1,RegWrite2,Data3,MemRead2,MemWrite2,M
125   Data_Memory A21(Reset,Y1,Data3,MemRead2,MemWrite2,Read_Data);
126   Mem_WB_Reg A22(Clk,Reset,Read_Data,Dest_Reg_Num2,Y1,Mem2Reg2,RegWrite2,W,Write_Reg_Num,Y2,Mem2Reg3,RegWrite3);
127   MuxMem_2x1 A23(W,Y2,Mem2Reg3,Write_Data);
128   Branch_Cal A66(Out1,SgnEx,B_Addr);
129   Branch_Comparator A77(Read_Data1,Read_Data2,Opcode,IF_Flush);
130 endmodule
```

---

**Reg_File**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.04.2023 17:54:59
// Design Name:
// Module Name: Reg_File
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Reg_File(
input Reset,
    input [4:0] Read_Reg_Num1,
    input [4:0] Read_Reg_Num2,
    input [4:0] Write_Reg_Num,
    input [31:0] Write_Data,
    input RegWrite3,
    output [31:0] Read_1,
    output [31:0] Read_2
 );
 reg [31:0] RegMem [31:0];
```

```
    assign Read_1=RegMem[Read_Reg_Num1];
    assign Read_2=RegMem[Read_Reg_Num2];

    always@(RegWrite3,Write_Data,Write_Reg_Num)
    begin
    if (RegWrite3==1)
    RegMem[Write_Reg_Num]=Write_Data;
    end

    always@(negedge Reset)
    begin
    if (Reset==0)
    begin
    RegMem[0]=32'h0;
    RegMem[1]=32'h00000001;
    RegMem[2]=32'h00000002;
    RegMem[3]=32'h00000003;
    RegMem[4]=32'h00000004;
    RegMem[5]=32'h00000005;
    RegMem[6]=32'h00000006;
    RegMem[7]=32'h00000007;
    RegMem[8]=32'h00000008;
    RegMem[9]=32'h00000009;
    RegMem[10]=32'h0000000A;
    RegMem[11]=32'h00000002;
    RegMem[12]=32'h0000000C;
    RegMem[13]=32'h0000000D;
    RegMem[14]=32'h0000000E;
    RegMem[15]=32'h0000000F;
    RegMem[16]=32'h00000010;
    RegMem[17]=32'h00000011;
    RegMem[18]=32'h00000012;
    RegMem[19]=32'h00000013;
    RegMem[20]=32'h00000014;
    RegMem[21]=32'h00000015;
    RegMem[22]=32'h00000016;
    RegMem[23]=32'h00000017;
    RegMem[24]=32'h00000018;
    RegMem[25]=32'h00000019;
    RegMem[26]=32'h0000001A;
    RegMem[27]=32'h0000001B;
    RegMem[28]=32'h0000001C;
    RegMem[29]=32'h0000001D;
    RegMem[30]=32'h0000001E;
    RegMem[31]=32'h0000001F;
    end
    end
  endmodule
```

---

**Sign_Ex**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 18.04.2023 00:10:00
// Design Name:
// Module Name: Sign_Ex
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module Sign_Ex(
    input [11:0] Immediate,
    input [11:0] Immediate3,
    output [31:0] SgnEx,
    output [31:0] SignEx
    );
    wire [11:0] Shift_2;
    assign Shift_2=Immediate3<<2;
    assign SgnEx={{20{Shift_2[11]}},Shift_2};
    assign SignEx={{20{Immediate[11]}},Immediate};

endmodule
```

---

## Stall_Unit

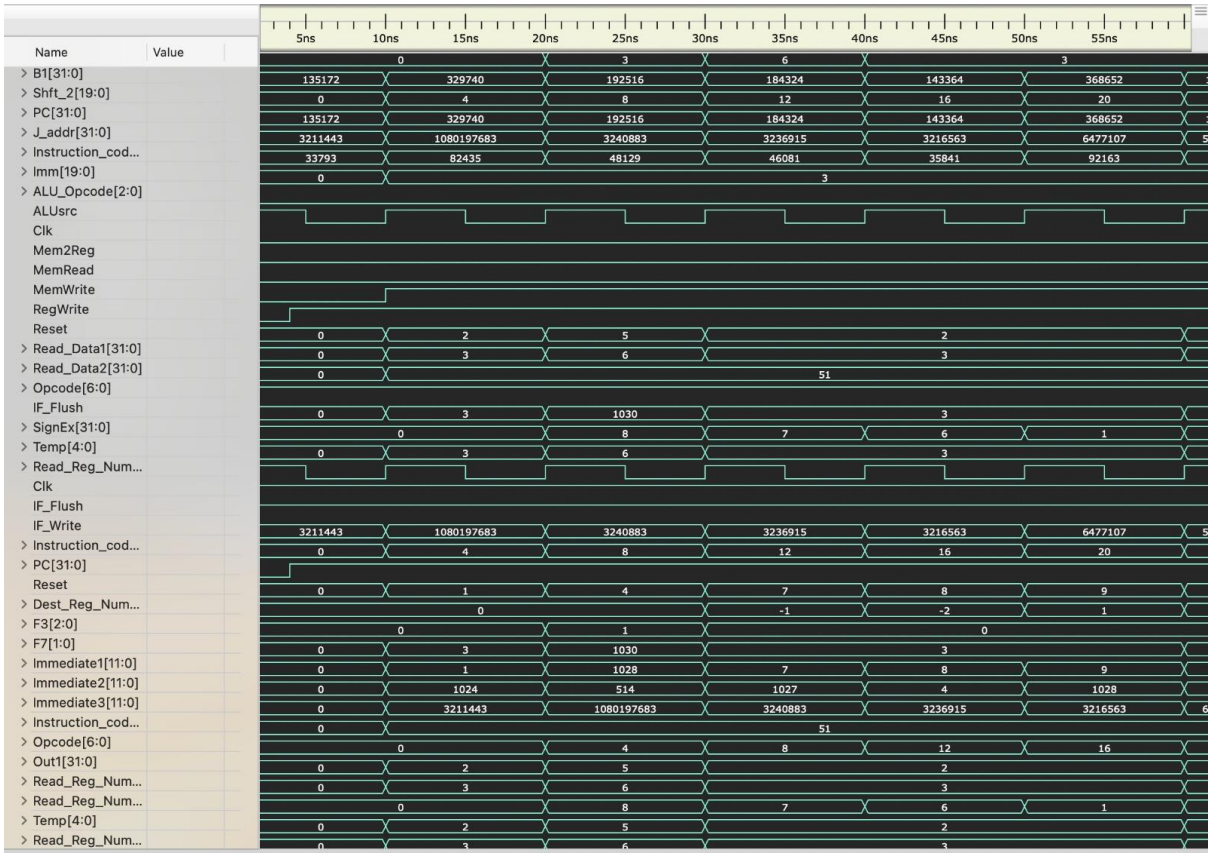
```
module Stall_Unit(
    input MemRead1,
    input MemRead2,
    input [4:0] Dest_Reg_Num1,
    input [4:0] Dest_Reg_Num2,
    input [4:0] Read_Reg_Num1,
    input [4:0] Read_Reg_Num2,
    input RegWrite1,
    input [2:0] ALU_Opcode,
    output reg Cntsrc,
    output reg PC_Write,
    output reg IF_Write
    );

    always@(MemRead1,MemRead2,ALU_Opcode,Dest_Reg_Num1,Dest_Reg_Num2,RegWrite1,Read_Reg_Num1,Read_Reg_Num2,RegWrite1)
    begin
    if ((MemRead1==1)&&((Dest_Reg_Num1==Read_Reg_Num1)||(Dest_Reg_Num1==Read_Reg_Num2)))
    begin
        Cntsrc=0;
        PC_Write=0;
        IF_Write=0;
    end
    else  if ((MemRead2==1)&&(ALU_Opcode==3'b110)&&((Dest_Reg_Num2==Read_Reg_Num1)||(Dest_Reg_Num2==Read_Reg_Num2)))
    begin
           Cntsrc=0;
           PC_Write=0;
           IF_Write=0;
    end
    else  if ((RegWrite1==1)&&(ALU_Opcode==3'b110)&&((Dest_Reg_Num1==Read_Reg_Num1)||(Dest_Reg_Num1==Read_Reg_Num2)))
    begin
          Cntsrc=0;
          PC_Write=0;
          IF_Write=0;
   end
    else
    begin
        Cntsrc=1;
        PC_Write=1;
        IF_Write=1;
    end
    end
endmodule
```

---

## OUTPUT WAVEFORMS

Here we are displaying the output waveform for the Register File, Data memory and the immediate and Jump address.




## Acknowledgements:

Our team would like to state that the project was made while learning from the internet and we also took help to write the code from our batchmate Anas Khan - 2019B5AA0896H.