i. Write a SV program for Instruction Decode Stage

```
module instr_decode (
   input logic [31:0] inst,
   output logic [31:0] rs1_data,
   output logic [31:0] rs2_data,
   output logic [31:0] imm32,
            output logic [6:0] func7,
            output logic [2:0] func3,
            input logic [31:0] wr_data,
            input logic reset
);
   wire [4:0] rs1_addr;
   wire [4:0] rs2_addr;
   wire [4:0] rd_addr;
   wire [11:0] imm12;
   wire regwrite;
   //wire reset;
   //wire [31:0] wr_data;

decoder g1(inst,rd_addr,rs1_addr,rs2_addr,func3,func7,imm12);
regfile g2(rs1_addr,rs2_addr,rd_addr,regwrite,wr_data,reset,rs1_data,rs2_data);
sign_exe g3(imm12,imm32);
endmodule
```

ii.Write a SV program for the Decoder

```
module decoder (input logic [31:0]inst, output logic [4:0]rd,output logic [4:0]rs1, output logic
[4:0]rs2,output logic [2:0]func3, output logic [6:0]func7,output logic [11:0]imm);
always_comb
begin
 rd = 5'b0;
```

```
        rs1 = 5'b0;
        rs2 = 5'b0;
        func3 = 3'b0;
        func7 = 7'b0;
        imm = 12'b0;
case(inst[6:0])
7'b0110011:
begin
 rd=inst[11:7];
 rs1=inst[19:15];
 rs2=inst[24:20];
 func3=inst[14:12];
 func7=inst[31:25];
 end

 7'b0010011:
 begin
 rd=inst[11:7];
 rs1=inst[19:15];
 func3=inst[14:7];
 imm=inst[11:0];
 end

 7'b1100011:
 begin
 rs1=inst[19:15];
 rs2=inst[24:20];
 func3=inst[14:12];
 imm={inst[31],inst[30:25],inst[11:8],inst[7]};
 end
```

```
7'b0100011:
begin
rs1=inst[19:15];
rs2=inst[24:20];
func3=inst[14:7];
imm={inst[31:25],inst[11:7]};
end

7'b0000011:
begin
rd=inst[11:7];
rs1=inst[19:15];
func3=inst[14:7];
imm=inst[11:0];
end

endcase
end
endmodule
```

3.Write a SV program for the Register File

```
module regfile(input logic [4:0]rs1_addr,rs2_addr,rd_addr,input logic reg_write,input logic
[31:0]wr_data, input logic reset,output logic [31:0]rs1_data,rs2_data);
logic [31:0]rfile[0:31];
always_comb
begin
        if(~reset)
        begin
                rs1_data=rfile[rs1_addr];
                rs2_data=rfile[rs2_addr];
```

```
            end
            else
            begin
                        rs1_data=32'b0;
                        rs2_data=32'b0;
            end
end
always_ff @(posedge reg_write or posedge reset) begin
      if (reset) begin
         for (int i = 0; i < 32; i++) begin
            rfile[i] = 32'b0; // Clear all registers
         end
      end
                        else if (reg_write) begin
         rfile[rd_addr] = wr_data; // Write data to the specified register
      end
                        end
endmodule
```

iv.Write a SV program for the Immediate Unit

```
module PC #(parameter N=32) (input logic clk,input logic rst,input logic [N-1:0] pc_next,output logic
[N-1:0] pc);
always_ff@(posedge clk)
begin
  if (rst)
    pc<= 32'b0 ;
  else
    pc<= pc_next ;
end
endmodule
```