Birla Institute of Technology and Science – Pilani, Hyderabad Campus
Second Semester 2021-22
CS F342: Computer Architecture Assignment (30 Marks)

A. Implement 4-stage pipelined processor in Verilog. This processor supports load immediate (li), shift left logical (sll) and Unconditional Jump (J) instructions only. The processor should implement forwarding to resolve data hazards. The processor has Reset, CLK as inputs and no outputs. The processor has instruction fetch unit, register file (with eight 8-bit registers), Execution and Writeback unit. Read and write operations on Register file can happen simultaneously and should be independent of CLK. The processor also contains three pipelined registers IF/ID, ID/EX and EX/WB. When reset is activated the PC, IF/ID, ID/EX, EX/WB registers are initialized to 0, the instruction memory and registerfile get loaded by **predefined values**. When the instruction unit starts fetching the first instruction the pipeline registers contain unknown values. When the second instruction is being fetched in IF unit, the IF/ID registers will hold the instruction code for first instruction. When the third instruction is being fetched by IF unit, the IF/ID register contains the instruction code of second instruction, ID/EX register contains information related to first instruction and so on. (Assume 8-bit PC. Also Assume Address and Data size as 8-bits).

The instructions and its **8-bit instruction format** for single cycle and pipeplined processor are shown below:

li DestinationReg, ImmediateData (Signextends data specified in instruction field (2:0) to 8-bits and stores it in register specified by register number in RDst field. Opcode for li is 00)

Opcode		
00	RDst	Immediate Data
7:6	5:3	2:0

Example usage: li R3, 4 (4 = 100 sign extension will result in 1111100. This data moves in to R3.

sll DestinationReg, shiftamount (Left shifts data in register specified by register number in RDst field by shift amount and moves back result to same register. Opcode for sll is 01)

Opcode		
01	RDst	Shamt
7:6	5:3	2:0

Example usage: sll R0, 4 shifts value in R0 by 4 times and store result back in R0.

j L1 (Jumps to an address generated by adding PC+1 to the Signextended data specified in instruction field (5:0). Opcode for j is 11)

Opcode	
11	Partial Jump Address
7:6	5:0

Example usage: j L1 (Jump address is calculated using PC relative addressing)

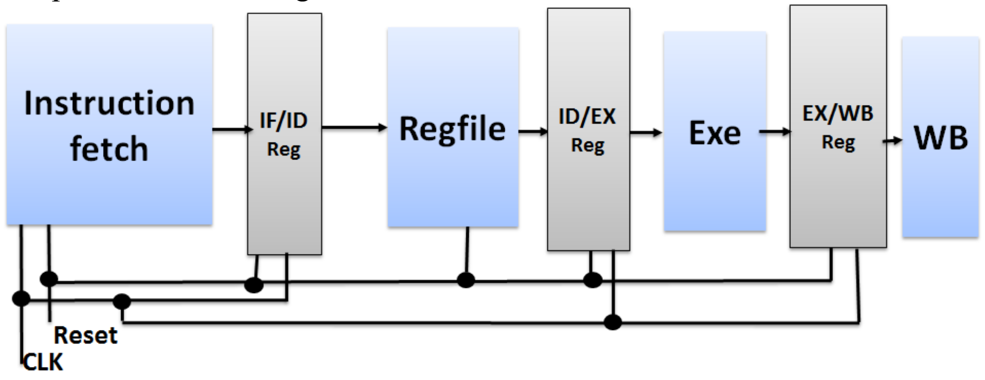
Assume the register file contains 8 registers (R0-R7) each register can hold 8-bit data. On reset register file should get initialized such that R0 = 0, R1 = 1, R2 = 2, R3 = 3 …etc. On reset assume that the instruction memory gets initialized with four instructions.

```
        li Rx, 3
        sll Rx, 1
        li Ry, 2
        j L1
        sll Ry, 3
L1:     li Rz, 4
```

Where x, y, z are related to last 3 digits of your ID No.
If ID number: 20XXXXXXABCH, then x =A mod 8 (A%8),

y= (B+2) mod 8 ((B+2)%8),
z= (C+3) mod 8 ((C+3)%8),

<u>Note for Pipelined Processor:</u> A partial block level representation of 4-stage pipelined processor is shown below. **Please note that for registerfile implementation, both read and write are independent of CLK**. Write operation depends on control signal.



Submission Procedure

As part of the assignment three files should be submitted in zipped folder.

1. PDF version of this Document with all the Questions below answered with file name as **IDNO_NAME.pdf**.

2. Design Verilog Files for all the Sub-modules (instruction fetch, Register file, forwarding unit, etc).

3. Design Verilog file for both single cycle and pipelined processor.

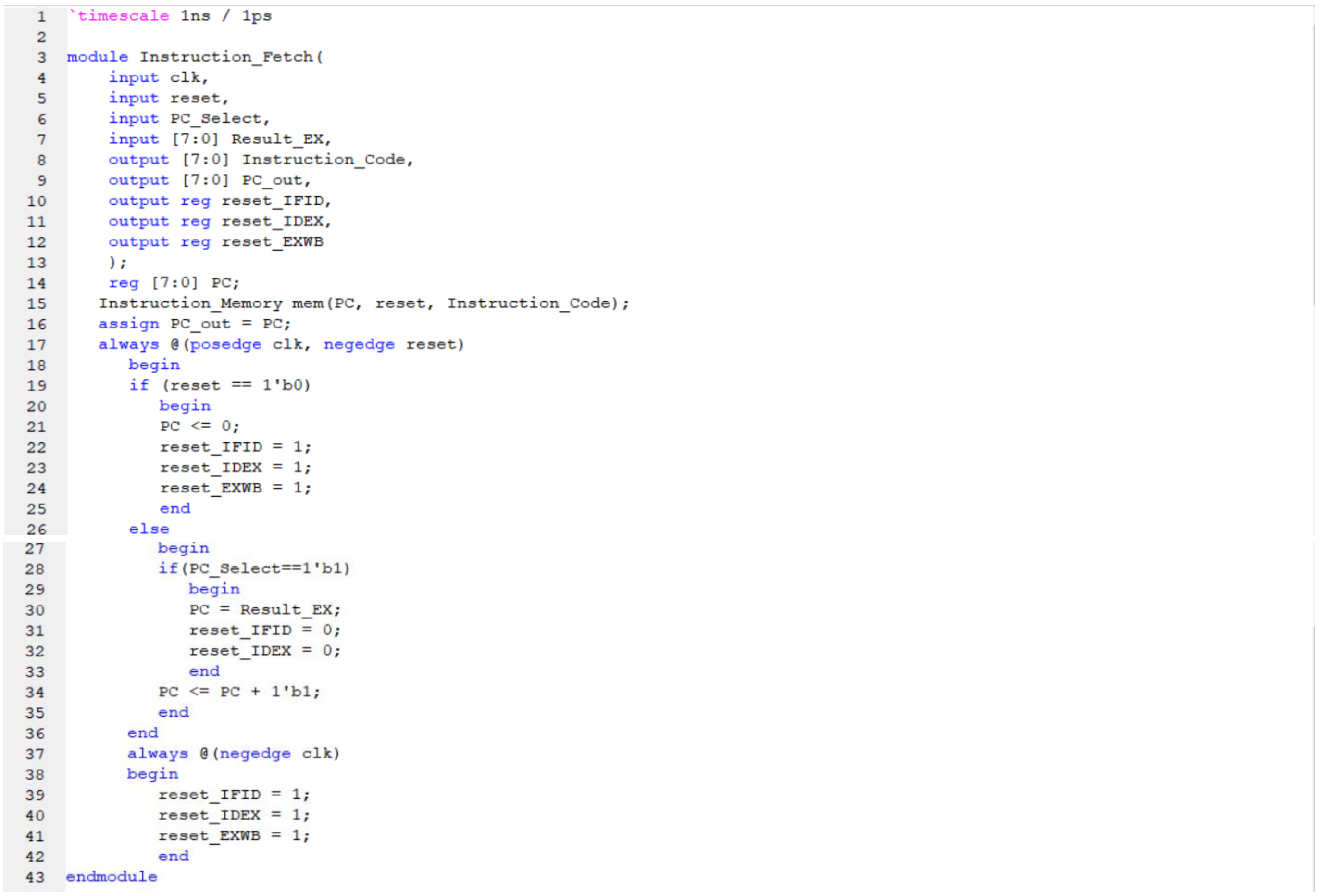
The name of the zipped folder should be in the format IDNO_NAME.zip

The due date for submission is 24-April-2022, 5:00 PM.

Name: Anantha Sai Satwik Vysyaraju **ID No:** 2019A3PS1323H

1. **Implement the Instruction Fetch block. Copy the <u>image</u> of Verilog code of the Instruction fetch block here**

Answer:



```verilog
`timescale 1ns / 1ps

module Instruction_Fetch(
    input clk,
    input reset,
    input PC_Select,
    input [7:0] Result_EX,
    output [7:0] Instruction_Code,
    output [7:0] PC_out,
    output reg reset_IFID,
    output reg reset_IDEX,
    output reg reset_EXWB
    );
    reg [7:0] PC;
   Instruction_Memory mem(PC, reset, Instruction_Code);
   assign PC_out = PC;
   always @(posedge clk, negedge reset)
      begin
      if (reset == 1'b0)
         begin
         PC <= 0;
         reset_IFID = 1;
         reset_IDEX = 1;
         reset_EXWB = 1;
         end
      else
         begin
         if(PC_Select==1'b1)
            begin
            PC = Result_EX;
            reset_IFID = 0;
            reset_IDEX = 0;
            end
         PC <= PC + 1'b1;
         end
      end
      always @(negedge clk)
      begin
         reset_IFID = 1;
         reset_IDEX = 1;
         reset_EXWB = 1;
         end
endmodule
```

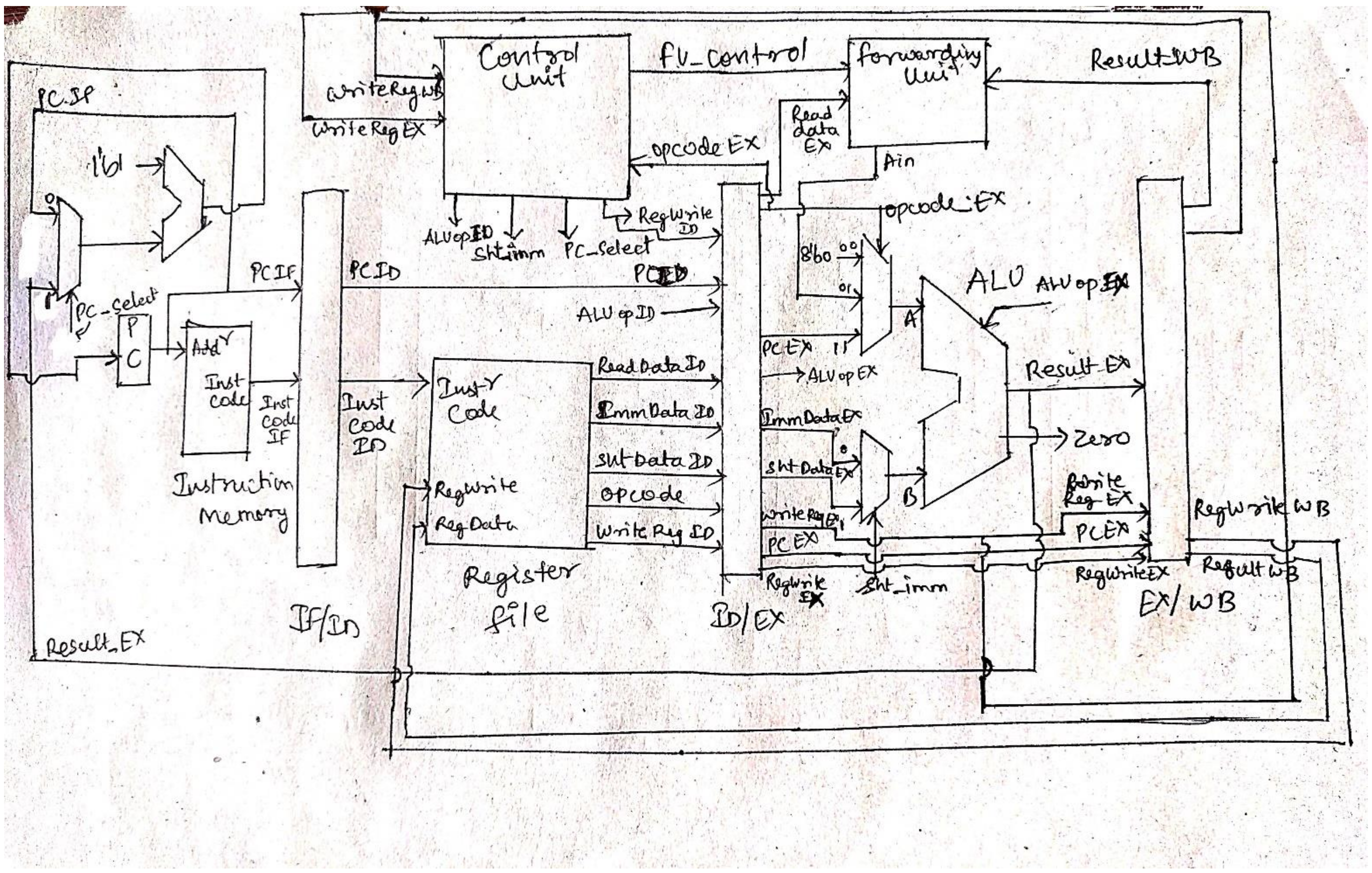
2. **List the control signals used and also the values of control signals for different instructions.**

Answer:

Instructions	**Control Signals**					
	RegWrite	**PC_Select**	**sht_imm**			
li	**1**	**0**	0			
sll	1	0	0			
j	0	1	1			

3. **Draw the complete Datapath and show control signals of the 4-stage pipelined processor. A sample Datapath for 5-stage pipelined MIPS processor has been discussed in class. A ppt named Assignmenthelp.ppt contains this 5-stage processor and is uploaded in CMS. You can modify this according to your specification.**

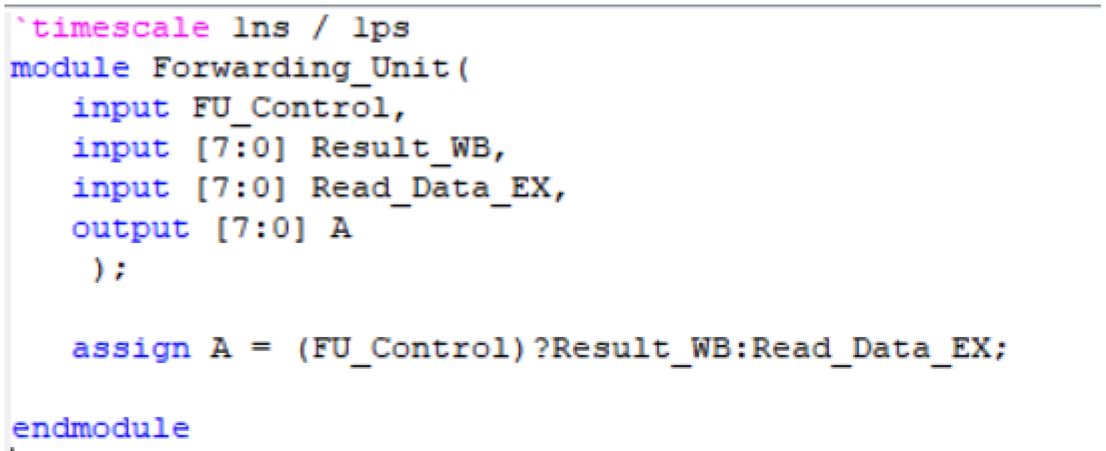
Answer:



4. **Determine the condition that can be used to detect data hazard?**

Answer: Write_Reg_EX == Write_Reg_WB

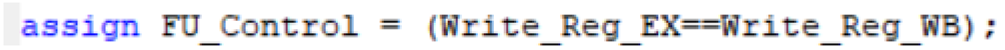
5. **Implement the forwarding unit and copy the <u>image</u> of Verilog code of forwarding unit here.**



```
`timescale 1ns / 1ps
module Forwarding_Unit(
   input FU_Control,
   input [7:0] Result_WB,
   input [7:0] Read_Data_EX,
   output [7:0] A
    );

   assign A = (FU_Control)?Result_WB:Read_Data_EX;

endmodule
```

Answer:

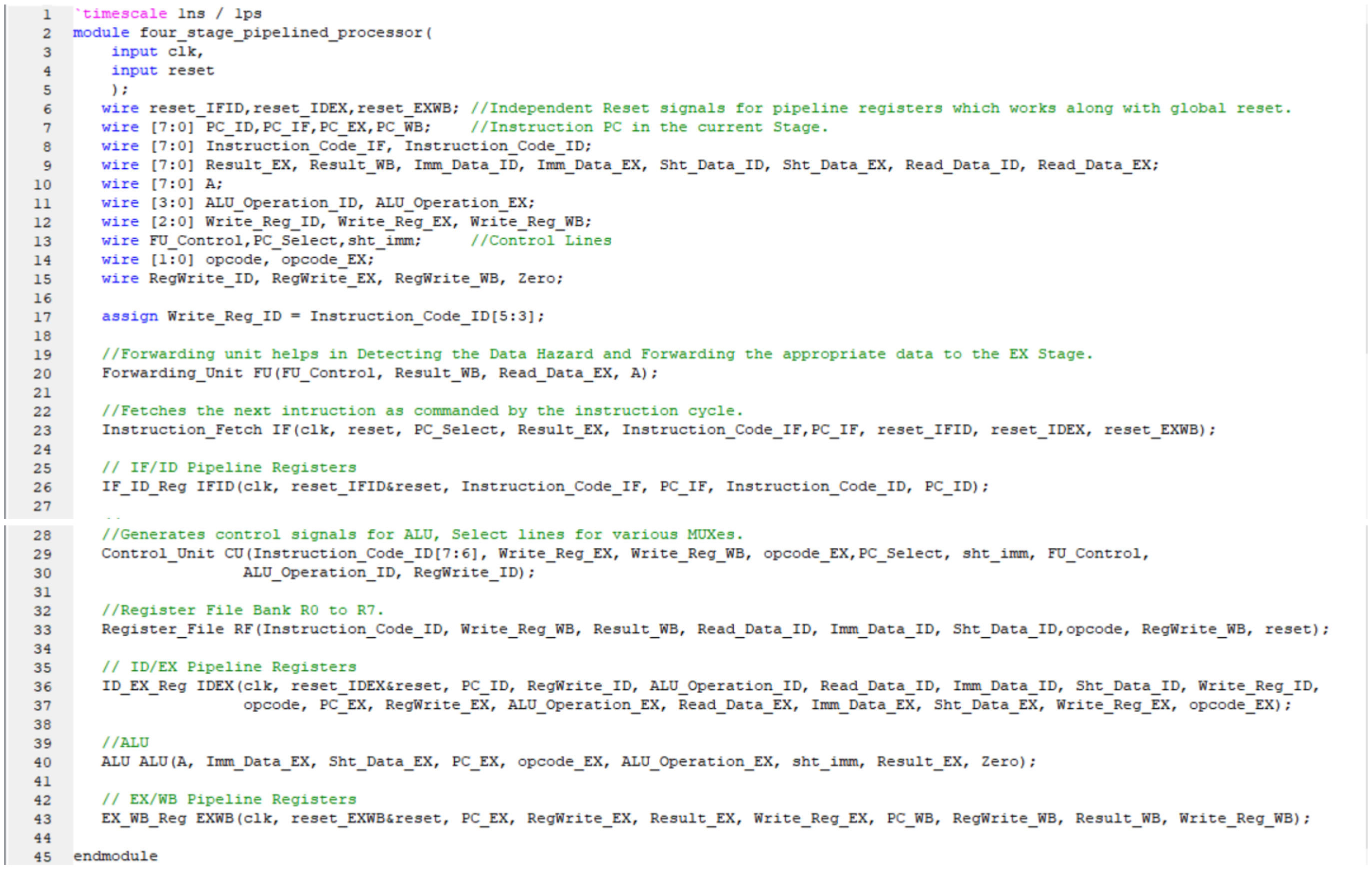
FU_Control is given by the control unit by comparing the Write_Reg_EX and Write_Reg_WB.



```
assign FU_Control = (Write_Reg_EX==Write_Reg_WB);
```

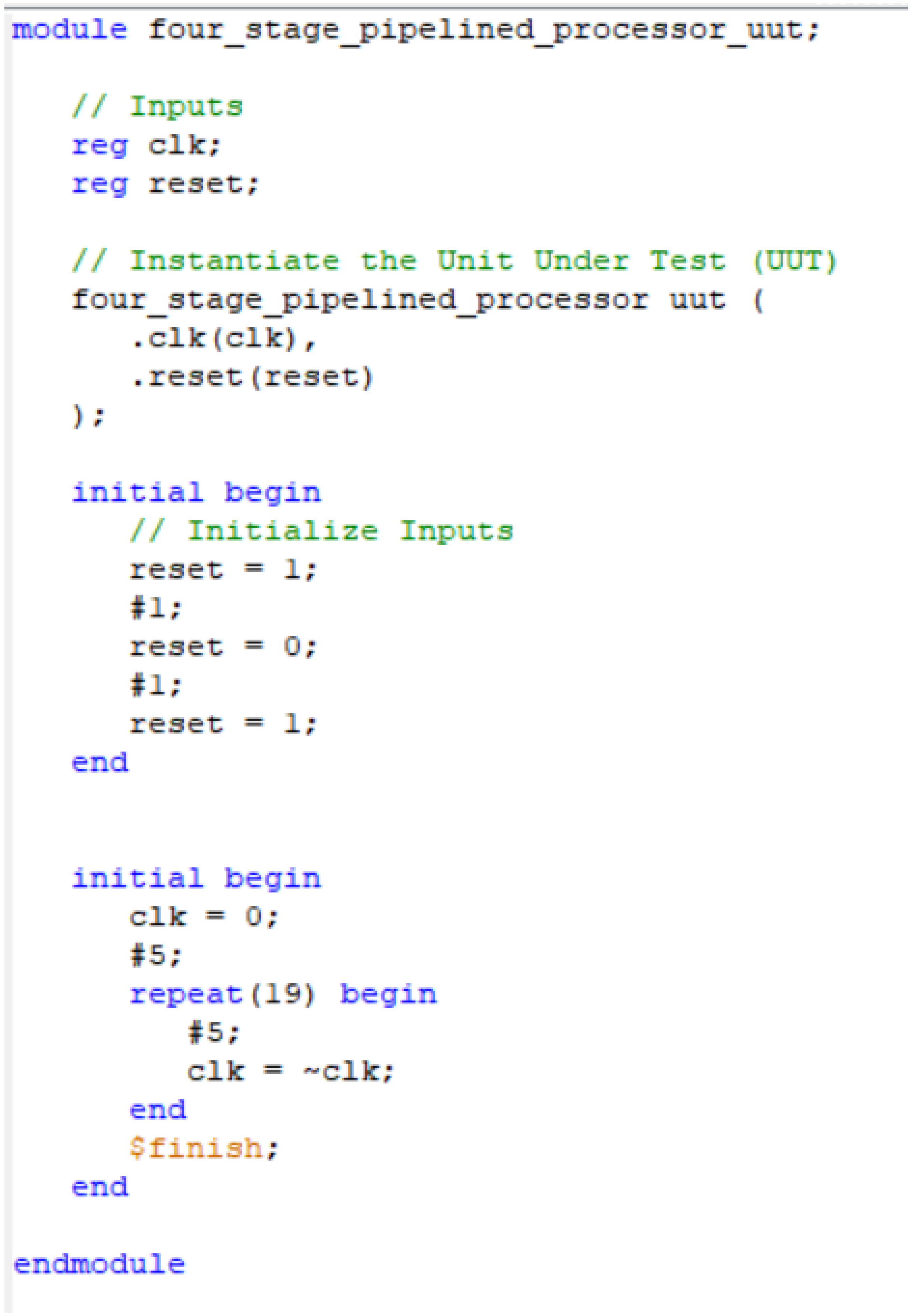
6. **Implement complete pipelined processor in Verilog (using all the Datapath blocks). Copy the <u>image</u> of Verilog code of the processor here. (Use comments to describe your Verilog implementation)**

Answer:



```verilog
`timescale 1ns / 1ps
module four_stage_pipelined_processor(
    input clk,
    input reset
    );
   wire reset_IFID,reset_IDEX,reset_EXWB; //Independent Reset signals for pipeline registers which works along with global reset.
   wire [7:0] PC_ID,PC_IF,PC_EX,PC_WB;    //Instruction PC in the current Stage.
   wire [7:0] Instruction_Code_IF, Instruction_Code_ID;
   wire [7:0] Result_EX, Result_WB, Imm_Data_ID, Imm_Data_EX, Sht_Data_ID, Sht_Data_EX, Read_Data_ID, Read_Data_EX;
   wire [7:0] A;
   wire [3:0] ALU_Operation_ID, ALU_Operation_EX;
   wire [2:0] Write_Reg_ID, Write_Reg_EX, Write_Reg_WB;
   wire FU_Control,PC_Select,sht_imm;     //Control Lines
   wire [1:0] opcode, opcode_EX;
   wire RegWrite_ID, RegWrite_EX, RegWrite_WB, Zero;

   assign Write_Reg_ID = Instruction_Code_ID[5:3];

   //Forwarding unit helps in Detecting the Data Hazard and Forwarding the appropriate data to the EX Stage.
   Forwarding_Unit FU(FU_Control, Result_WB, Read_Data_EX, A);

   //Fetches the next intruction as commanded by the instruction cycle.
   Instruction_Fetch IF(clk, reset, PC_Select, Result_EX, Instruction_Code_IF,PC_IF, reset_IFID, reset_IDEX, reset_EXWB);

   // IF/ID Pipeline Registers
   IF_ID_Reg IFID(clk, reset_IFID&reset, Instruction_Code_IF, PC_IF, Instruction_Code_ID, PC_ID);

   //Generates control signals for ALU, Select lines for various MUXes.
   Control_Unit CU(Instruction_Code_ID[7:6], Write_Reg_EX, Write_Reg_WB, opcode_EX,PC_Select, sht_imm, FU_Control,
                   ALU_Operation_ID, RegWrite_ID);

   //Register File Bank R0 to R7.
   Register_File RF(Instruction_Code_ID, Write_Reg_WB, Result_WB, Read_Data_ID, Imm_Data_ID, Sht_Data_ID,opcode, RegWrite_WB, reset);

   // ID/EX Pipeline Registers
   ID_EX_Reg IDEX(clk, reset_IDEX&reset, PC_ID, RegWrite_ID, ALU_Operation_ID, Read_Data_ID, Imm_Data_ID, Sht_Data_ID, Write_Reg_ID,
                   opcode, PC_EX, RegWrite_EX, ALU_Operation_EX, Read_Data_EX, Imm_Data_EX, Sht_Data_EX, Write_Reg_EX, opcode_EX);

   //ALU
   ALU ALU(A, Imm_Data_EX, Sht_Data_EX, PC_EX, opcode_EX, ALU_Operation_EX, sht_imm, Result_EX, Zero);

   // EX/WB Pipeline Registers
   EX_WB_Reg EXWB(clk, reset_EXWB&reset, PC_EX, RegWrite_EX, Result_EX, Write_Reg_EX, PC_WB, RegWrite_WB, Result_WB, Write_Reg_WB);

endmodule
```

7. **Test the pipelined processor design by generating the appropriate clock and reset. Copy the <u>image</u> of your testbench code here.**

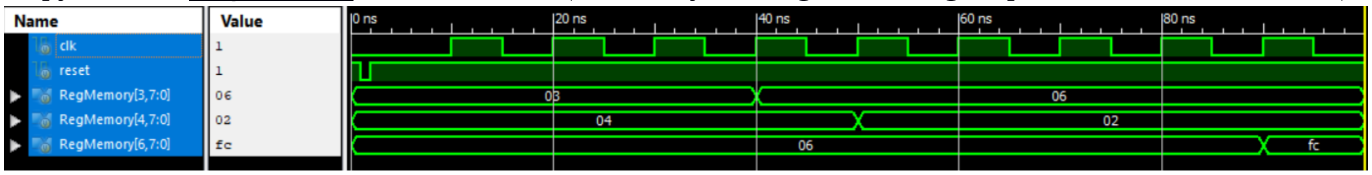


```
module four_stage_pipelined_processor_uut;

   // Inputs
   reg clk;
   reg reset;

   // Instantiate the Unit Under Test (UUT)
   four_stage_pipelined_processor uut (
      .clk(clk),
      .reset(reset)
   );

   initial begin
      // Initialize Inputs
      reset = 1;
      #1;
      reset = 0;
      #1;
      reset = 1;
   end


   initial begin
      clk = 0;
      #5;
      repeat(19) begin
         #5;
         clk = ~clk;
      end
      $finish;
   end

endmodule
```

Answer:

8. **Verify if the register file is getting updated according to the set of instructions (mentioned earlier).**

Copy verified **Register file** waveform here (show only the Registers that get updated, CLK, and RESET):



Unrelated Questions

What were the problems you faced during the implementation of the processor?

Answer: using individual reset for each and every pipeline register

Did you implement the processor on your own? If you took help from someone whose help did you take? Which part of the design did you take help for?

Answer: Yes

Honor Code Declaration by student:

- My answers to the above questions are my own work.
- I have not shared the codes/answers written by me with any other students. (I might have helped clear doubts of other students).
- I have not copied other's code/answers to improve my results. (I might have got some doubts cleared from other students).

Name: Anantha Sai Satwik Vysyaraju **Date:** 24-04-2022
ID No.: 2019A3PS1323H