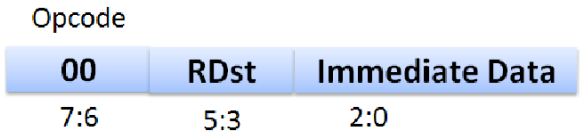
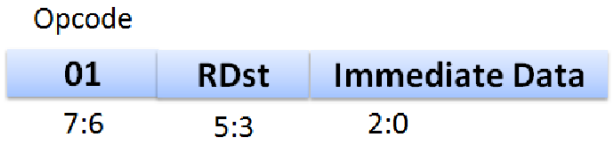
Birla Institute of Technology and Science – Pilani, Hyderabad Campus
Second Semester 2018-19
CS F342: Computer Architecture Assignment (20 Marks)

1. (a) Implement 4-stage pipelined processor in Verilog. This processor supports load immediate (li), constant addition (addi) 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 8 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 instruction and its **8-bit instruction format** 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)



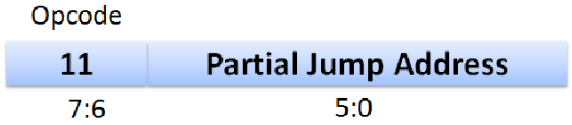
Example usage: li R3, 4 (4 = 100 signextension will result in 1111100. This data moves in to R3.

addi DestinationReg, immediateData (adds data in register specified by register number in RDst field to sign extended data in immediate data field (2:0). Result is stored in register specified by register number in RDst field. Opcode for addi is 01)



Example usage: addi R2, 3 3 gets sign extended to 8-bits 00000011 then is added to R2 (R2 ← R2+ 00000011)

j L1 (Jumps to an address generated by appending 2 MSB bits of PC+1 to the data specified in instruction field (5:0). Opcode for j is 11)



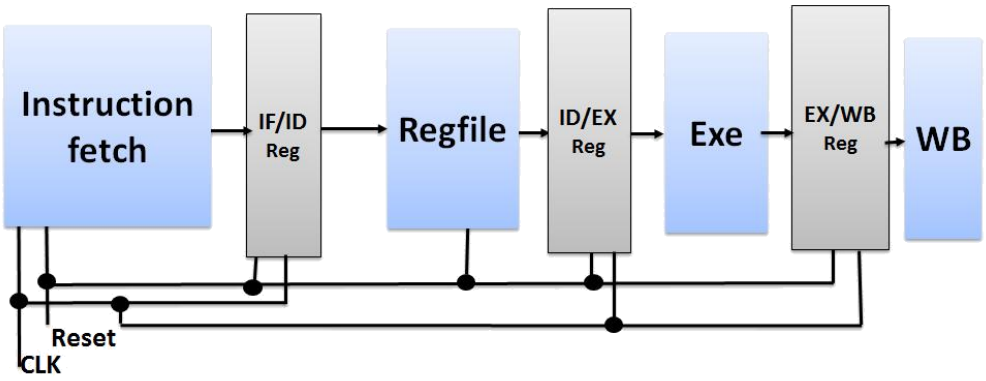
Example usage: j L1 (Jump address is calculated using Pseudo direct addressing)

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
        addi Rx, 2
        addi Ry,3
        j L1
        li Rz, 4
L1:     addi Rz, -3
```

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),

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.



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).

3. Design Verilog file for the main processor.

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

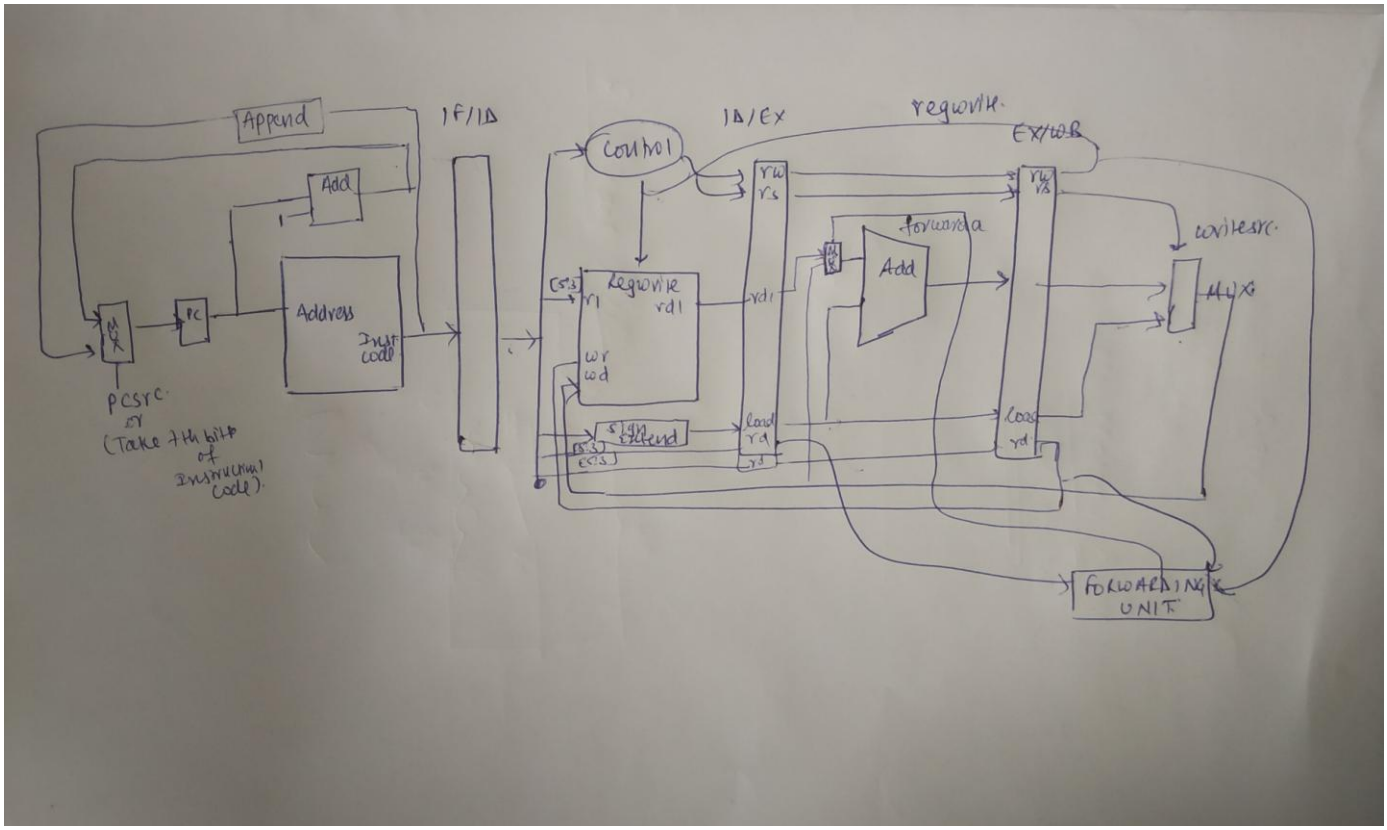
The due date for submission is 21-April-2019, 5:00 PM.

Name: Kunal Varshney **ID No:** 2016A8PS0422H

Questions Related to Assignment

1. **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:



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

Answer:

Instructions	**Control Signals**					
	opcode	regwrite	writesrc			
li	00	1	1			
addi	01	1	0			
j	11	0	0			

3. Implement the Instruction Fetch block. Copy the image of Verilog code of the Instruction fetch block here

```
module instfetch(
    input [7:0]PC,
   input clk,
    input rst,
    output [7:0]instruction_code
    );



    reg [7:0]mem[36:0];

    always@(rst)
    begin
    if(rst==1) $readmemh("Mem.mem",mem);
    end

    assign instruction_code = mem[PC];


endmodule
```

Answer:

Instruction Memory File

```
23
62
63
C5
2C
6D
```

4. Implement the Register File and copy the <u>image</u> of Verilog code of Register file unit here.

Answer:

```
module register(
    input [2:0]read_reg1,
    input [2:0]write_reg,
    input [7:0]write_data,
    input reset,
   input regwrite,
    output reg [7:0]read_data1
    );

    reg [7:0]regmem[7:0];

    always@(reset)
    begin
    if(reset==1) $readmemh("memreg.mem",regmem);
    end

    always@*
    begin
    read_data1 = regmem[read_reg1];
    end

    always@*
    begin
    if(regwrite==1) begin  regmem[write_reg] = write_data; end
    end


endmodule
```

REGISTER MEMORY FILE:

```
1  00
2  01
3  02
4  03
5  04
6  05
7  06
8  07
```

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

Answer: If the destination regsiter of source register of the n+1th instruction in ID/EX register is the same as the destination register of the nth instruction in EX/WB register while the regwrite signal is 1, we need to forward the write_data to either one of the alu inputs through muxes.

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

Answer:

```
module forwarding(
      input [2:0]rd,
      input osrc,
      input [2:0]writereg,
      input regwrite,
      input rst,
      output reg forwarda
    );

    always@(rst)
    begin
    if(rst==1)begin forwarda = 0;end
    end

    always@*
    begin
    if(regwrite==1 && rd == writereg)
    begin
    if(osrc == 0) begin forwarda = 1; end
    else if(osrc == 1) begin forwarda = 0; end
    end
    else begin forwarda = 0; end
    end

endmodule
```

7. Implement complete 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:

```
module pipe(
    input clk,
    input rst
    );

    wire writesrc;
    wire [7:0]inst_code;
    wire [7:0]inst_code1;
    wire [2:0]writereg;
    wire [2:0]writereg1;
    wire [7:0]write_data;
    wire regwrite;
    wire [7:0]r1;
    wire [7:0]r2;
    wire [7:0]s1;
    wire [7:0]o1;
    wire [7:0]o2;
    wire [7:0]os1;
    wire [2:0]rd;
    wire [2:0]rs;
    wire [7:0]out;
    wire [7:0]load;
    wire [7:0]alu;
    wire forwarda;
    wire forwardb;
    wire ow;
    wire osrc;
```

```
  wire regw;
  wire regsrc;
  wire [7:0]oul;
  wire [7:0]ou2;

  reg [7:0]PC;
  reg [7:0]PC_1;



    //PC module with jump
  always@(posedge rst,posedge clk)
  begin
  if(rst==1) PC = 0;
  else if(inst_code[7:6]==2'b11)
  begin
  PC_1 = PC + 1;
  PC = {{PC_1[7:6]},{inst_code[5:0]}};
  end
  else
  PC = PC+1;
  end

 // instruction fetch
  instfetch i1(PC,clk,rst,inst_code);
  //IF/ID Pipeline register
  IFID p1(inst_code,clk,rst,inst_code1);
  //Control Unit
 control c1(inst_code1[7:6],regwrite,writesrc);
 //Registers module
 register r3(inst_code1[5:3],writereg,write_data,rst,regw,r1);
 //Sign extension block
 signext s3(inst_code1[2:0],s1);
//ID/EX Pipeline register
 IDEX p2(clk,rst,regwrite,writesrc,inst_code1[5:3],r1,s1,inst_code1[5:3],o1,os1,rd,writeregl,ow,osrc);
 //forwarda mux
 mux m2(o1,write_data,forwarda,oul);
 //forwardb mux
 //alu (add)
 add a2(oul,os1,out);
 //EX/WB Pipeline register
 EXWB p3(clk,rst,ow,osrc,out,os1,writeregl,load,alu,writereg,regw,regsrc);
 //forwarding unit
 forwarding f1(rd,osrc,writereg,regw,rst,forwarda);
 //writesrc mux
 mux m1(alu,load,regsrc,write_data);
endmodule
```

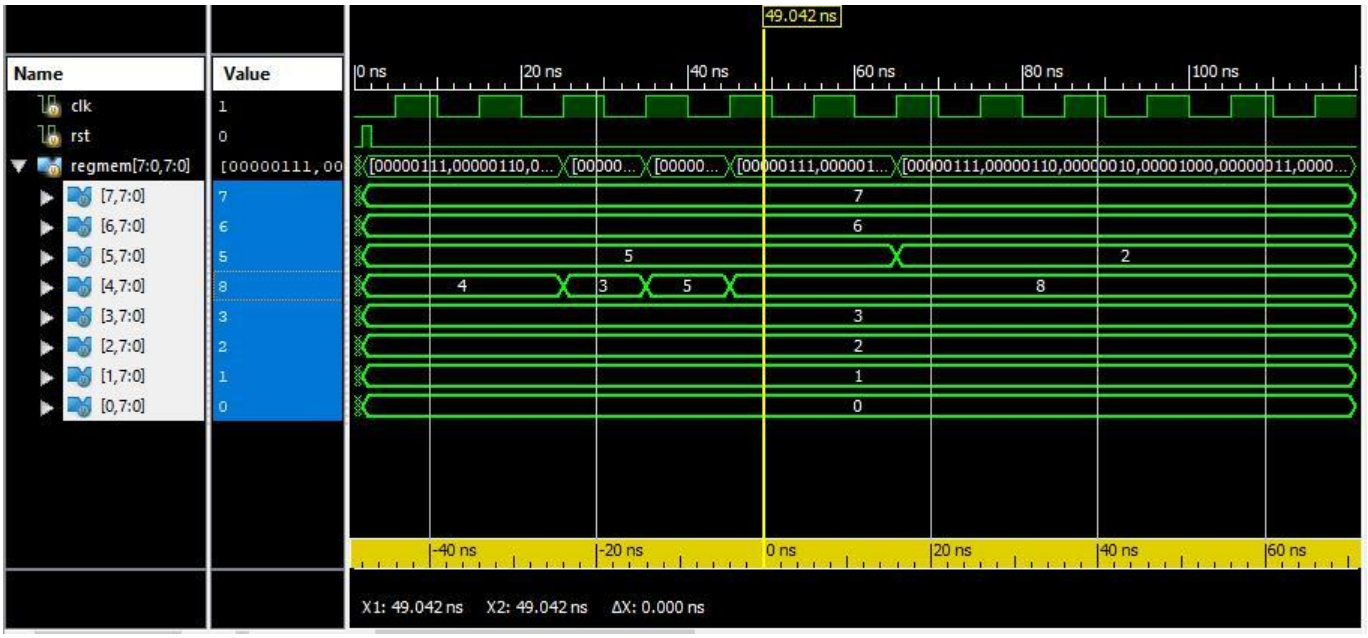
8. **Test the processor design by generating the appropriate clock and reset. Copy the <u>image</u> of your testbench code here.**

```
module test;

    // Inputs
    reg clk;
    reg rst;

    // Instantiate the Unit Under Test (UUT)
    pipe uut (
        .clk(clk),
        .rst(rst)
    );

    initial begin
    clk =0;
        repeat(24)
        #5 clk =~clk;
    end

    initial begin
    rst = 0; #1
    rst = 1; #1
    rst = 0; #118
    $finish;
    end

endmodule
```

Answer:

9. 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:

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, I IMPLEMENTED THE PROCESSOR ON MY OWN

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: KUNAL VARSHNEY **Date:** 23-04-2019
ID No.: 2016A8PS0422H