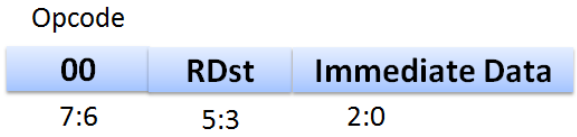
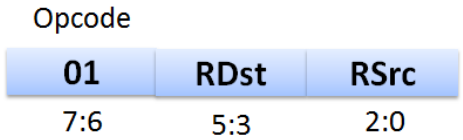
Birla Institute of Technology and Science – Pilani , Hyderabad Campus
Second Semester 2019-20
CS F342: Computer Architecture Assignment (20 Marks)

1. (a) Implement 4-stage pipelined processor in Verilog. This processor supports load immediate (li), addition (add) 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)



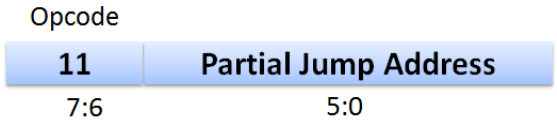
Example usage: li R3, 4 (4 = 100 signextension will result in 1111100. This data moves in to R3.

add DestinationReg, SourceReg (adds data in register specified by register number in Rsrc field to data in register specified by register number in RDst field. Result is stored in register specified by register number in RDst field. Opcode for add is 01)



Example usage: add R2, R0 (R2←R2+ R0)

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)



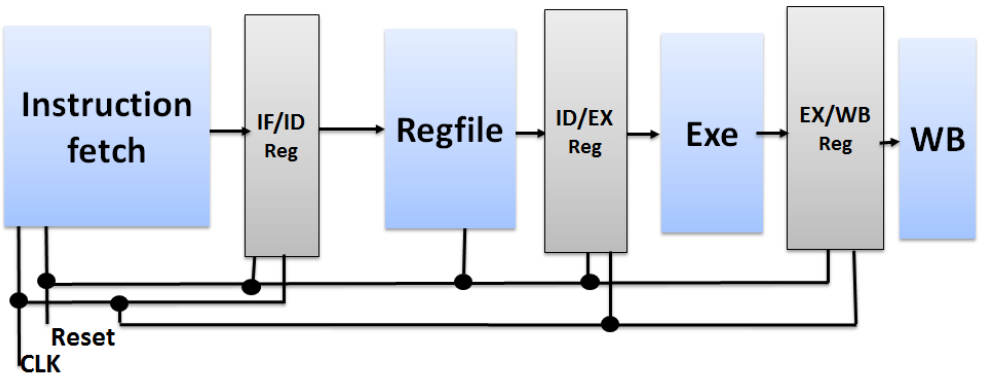
Example usage: j L1 (Jump address is calculated using Pseudo direct addressing)

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
        add Ry, Rx
        add Rz, Ry
        j L1
        li Rz, 4
L1:     add Rx, Rz
```

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

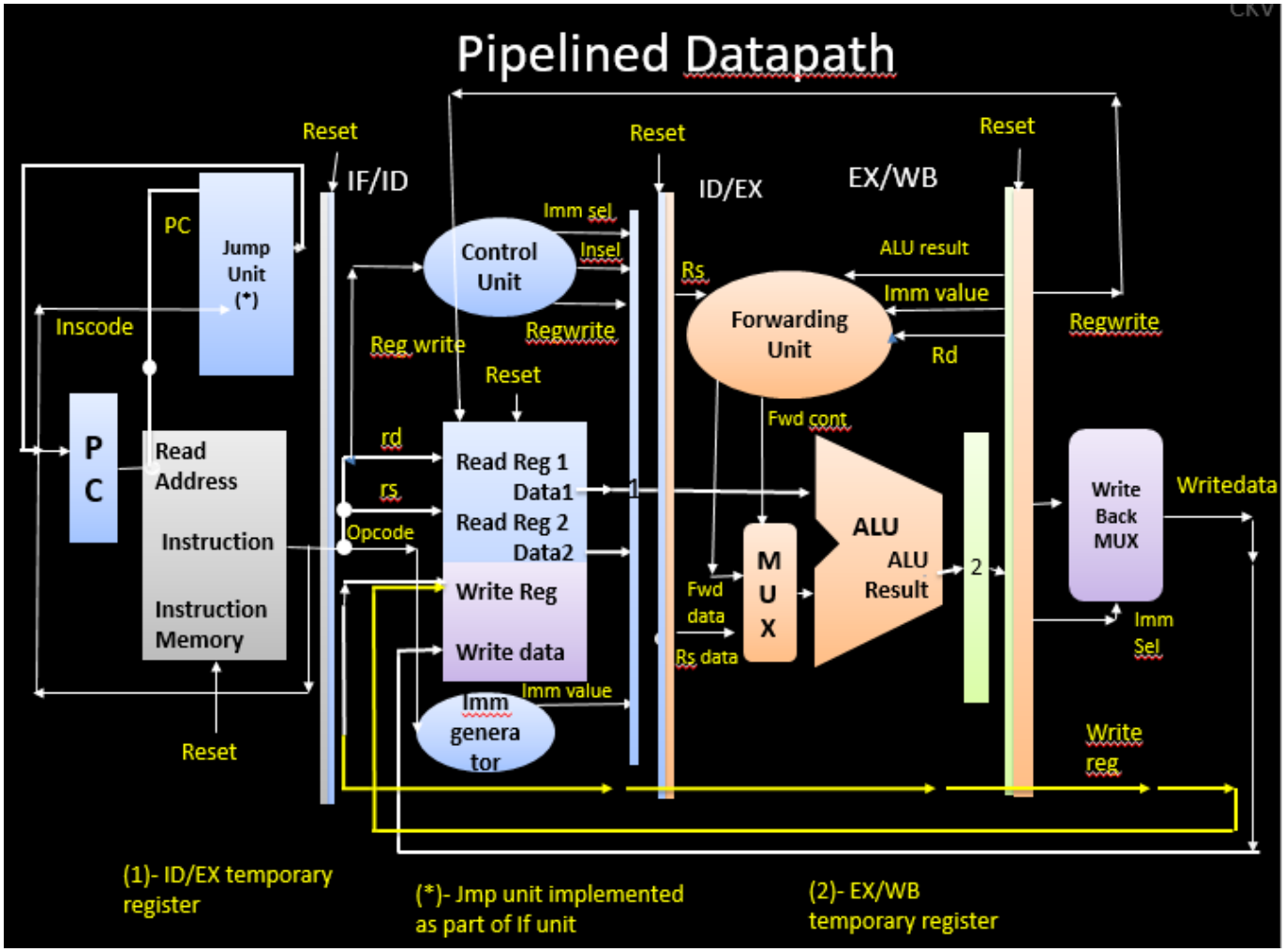
The due date for submission is 25-April-2020, 5:00 PM.

Name: SHASHWAT KAKKAR **IDNo:** 2017AAPS0269H

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**					
	Immsel	Insel	**Regwrite**			
li	1	00	1			
add	0	01	1			
j	X	11	X			

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

Answer:

```
module Insfetch(
    input clk,
    input reset,
    input [7:0] jmp,
    input [7:0] pcjmp,
    output reg [7:0] pc,
    output [7:0] inscode
    );
reg [7:0] temp;
insmem il(.pc(temp),.inscode(inscode));
always @(posedge clk)
begin
if(reset==1)
begin
temp <= 0;
pc <= 0;
end
else
begin
if(jmp[7:6]==2'b11)
begin
temp = {pcjmp[7:6],jmp[5:0]};
pc <= temp;
end
else
begin
temp = temp+1;
pc <= temp;
end
end
end
endmodule
```

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

Answer:

```
21  module regfile(
22      input [2:0] readreg1,
23      input [2:0] readreg2,
24      input reset,
25      input regwrite,
26      input [2:0] writereg,
27      input [7:0] writedata,
28      output reg [7:0] data1,
29      output reg [7:0] data2
30      );
31  reg [7:0] fixreg [7:0];
32  always @(readreg1 or readreg2 or writedata)
33  begin
34  if(reset==1)
35  begin
36  fixreg[0] = 'd0; fixreg[1] = 'd1; fixreg[2] = 'd2;
37  fixreg[3] = 'd3; fixreg[4] = 'd4; fixreg[5] = 'd5; fixreg[6] = 'd6; fixreg[7] = 'd7;
38  data1 <= fixreg[readreg1];
39  data2 <= fixreg[readreg2];
40  if(regwrite==1)
41  begin
42  fixreg[writereg] <= writedata;
43  end
44  end
45  else
46  begin
47  if(writereg==readreg1)
48  begin
49  data1 <= writedata;
50  end
51  else
52  begin
53  data1 <= fixreg[readreg1];
54  end
55  if(writereg==readreg2)
56  begin
57  data2 <= writedata;
58  end
59  else
60  begin
61  data2 <= fixreg[readreg2];
62  end
63  if(regwrite==1)
64  begin
65  fixreg[writereg] <= writedata;
66  end
67  end
68  end
69  endmodule
```

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

Answer: **If** EX/WB.Rd = ID/EX.Rs1orRs2 , then forward.

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

Answer:

```
21  module forwardunit(
22      input [2:0] rd,
23      input [2:0] rs,
24      input [7:0] result,
25      input [7:0] immval,
26      input [1:0] opcode,
27      output reg [7:0] out,
28      output reg forward
29      );
30  always @(rd or rs)begin
31  if(rd==rs)
32  begin
33  forward = 1;
34  if(opcode==2'b01)
35  begin
36  out <= result;
37  end
38  if(opcode==2'b00)
39  begin
40  out <= immval;
41  end
42  end
43  else
44  begin
45  forward = 0;
46  end
47  end
48  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 processor(
    input clk,
    input reset
    );
wire [7:0] inscode; wire [7:0] pc; wire [7:0] jmpaddr;
wire [7:0] toreg; wire fwdcont;
wire [7:0] datal; wire [7:0] data2;wire [1:0] cont; wire immsel;wire regwrite;
wire [7:0] immgen; wire [7:0] fwddata; wire [7:0] toalu; wire [7:0] aluout;wire [7:0] wbdata;

//////
reg [27:0] idextemp;reg [7:0] exwbtemp;//used as temporary registers to hold the valuse of generated data at each stages
reg [7:0] If_Id; reg [33:0] Id_Ex; reg [25:0] Ex_Wb;  //pipeline registers
//Instantiated blocks
Insfetch If(.clk(clk),.reset(reset),.jmp(inscode),.pcjmp(pc),.inscode(inscode),.pc(pc));

regfile rg(.readreg1(If_Id[5:3]),.readreg2(If_Id[2:0]),.writereg(Ex_Wb[7:5]),.reset(reset),.regwrite(Ex_Wb[1]),
    .writedata(wbdata),.datal(datal), .data2(data2));

controlunit contt(.opcode(If_Id[7:6]),.insel(cont),.immsel(immsel),.regwrite(regwrite));

signextend imm(.imm(If_Id[5:0]),.insel(cont),.out(immgen));

mux forward(.a(Id_Ex[25:18]),.b(fwddata),.cont(fwdcont),.out(toalu));

alu ex(.datal(Id_Ex[33:26]),.data2(toalu),.result(aluout));

forwardunit fwd(.rd(Ex_Wb[7:5]),.rs(Id_Ex[2:0]),.result(Ex_Wb[23:16]),.immval(Ex_Wb[15:8]),.opcode(Ex_Wb[25:24]),
.out(fwddata),.forward(fwdcont));

mux wbmux(.a(Ex_Wb[23:16]),.b(Ex_Wb[15:8]),.cont(Ex_Wb[0]),.out(wbdata));
/// end of instantiation
always @(posedge clk)begin
if(reset==1)begin
If_Id <= 0;
Id_Ex <= 0;
Ex_Wb <= 0;
idextemp <= 0;
exwbtemp <= 0;
end
//Id_Ex pipeline reg contains datareg1 in [33:26],datareg2 in [25:18],imm value(for li)in [17:10], opcode in [9:8],
//regwrite signal in 7th bit and immsel used to select imm value in writeback stage in 6th bit
//rd address in [5:3],rs in [2:0]
//Ex_Wb pipeline reg contains opcode in [25:24],alu result in [23:16],imm value in [15:8],rd in [7:5],
// rs-[4:2],regwrite-1st bit, immsel in 0th bit
else begin
If_Id <= inscode; // contains instruction code (inscode)
Id_Ex <= {idextemp,If_Id[5:3],If_Id[2:0]};
Ex_Wb <= {Id_Ex[9:8],exwbtemp,Id_Ex[17:10],Id_Ex[5:3],Id_Ex[2:0],Id_Ex[7],Id_Ex[6]};
assign idextemp = {datal,data2,immgen,cont,regwrite,immsel}; //continuous assignment of temporary registers
assign exwbtemp = aluout;
end
end
endmodule
```

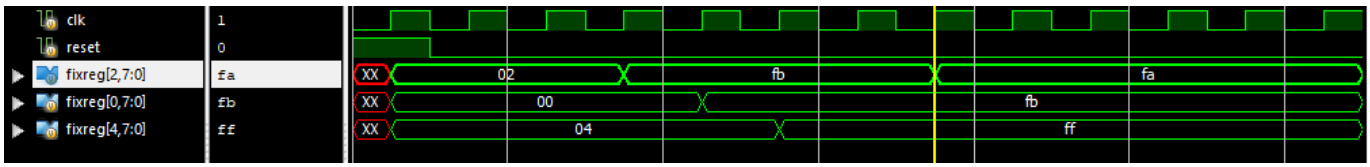
8. Test the processor design by generating the appropriate clock and reset. Copy the <u>image</u> of your testbench code here.

Answer:

```
24  module processortb;// Inputs
25     reg clk;
26     reg reset;
27  // Instantiate the Unit Under Test (UUT)
28     processor uut (
29        .clk(clk),
30        .reset(reset)
31     );
32
33     initial begin
34        // Initialize Inputs
35        clk = 0;
36        repeat(60)
37        #5 clk =~ clk;
38        $finish;
39     end
40        initial begin
41        reset = 1;
42        #10;
43        reset = 0;
44        #10;
45        reset = 0;
46        #10;
47        reset = 0;
48        #100;
49        $finish;
50     end
51  endmodule
```

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: Design related like implementing jump instructions, how pipeline registers will get updated with each clock cycle.

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 implemented 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: SHASHWAT KAKKAR **Date:** 26/04/2020
ID No.: 2017AAPS0269H