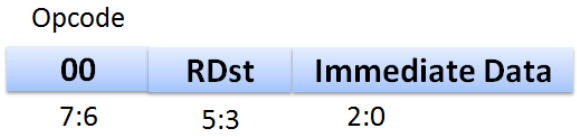
Birla Institute of Technology and Science – Pilani, Hyderabad Campus
Second Semester 2019-20
CS F342: Computer Architecture Assignment (20 Marks)

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

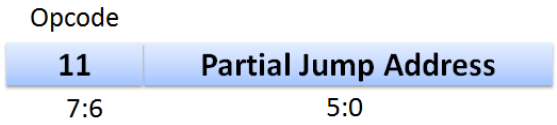


Example usage: li R3, 4 (4 = 100 sign extension will result in 1111100. This data moves in to R3.

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)

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



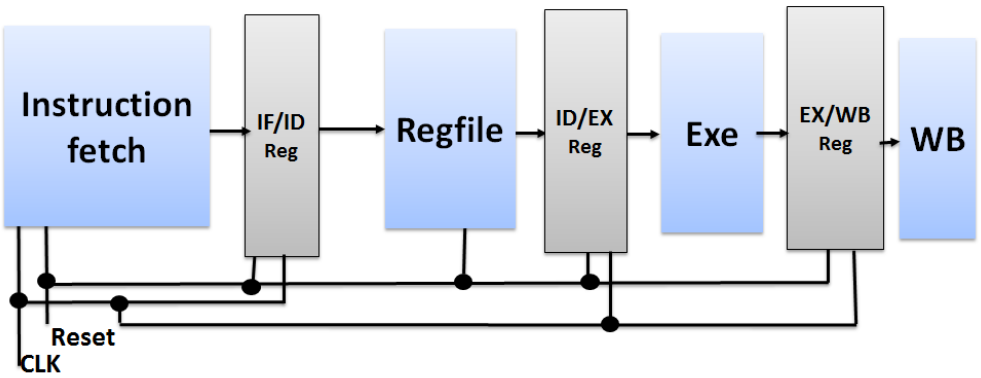
Example usage: j L1 (Jump address is calculated using Pseudo direct addressing)

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
        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),

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

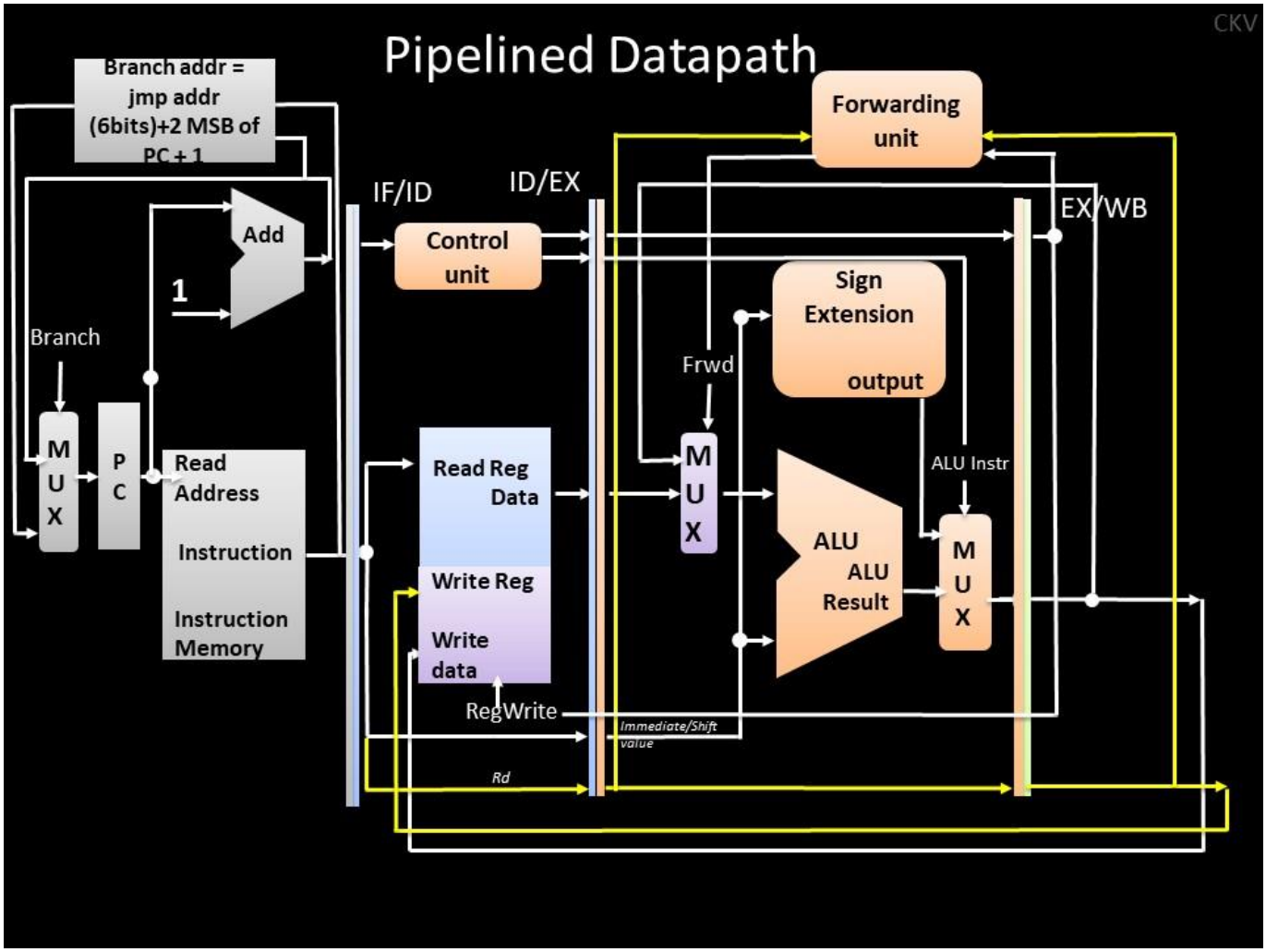
The due date for submission is 25-April-2020, 5:00 PM.

Name: Prashanth Reddy **ID No:** 2017AAPS0274H

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**					
	Reg_write	ALU_Instr	Branch			
li	1	0	0			
sll	1	1	0			
j	0	X	1			

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

Answer:

```
module instr_fetch_blck(
    input clk,
    input reset,
    output [7:0] instr_code,

    input [5:0]jmpaddr,
    input branch
    );

    reg [7:0] PC; wire [7:0] temp;

    assign temp = PC + 1;


    instr_mem_blck u1 (PC,reset,instr_code);

    always@(posedge clk,negedge reset)
    begin

    if(reset ==0)
    PC = 0;

    else if(branch)
     PC = {temp[7:6],jmpaddr};

    else
    PC = PC+1;

    end
endmodule
```

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

Answer:

```
module reg_file(
    input reset,

    input [2:0] read_reg_no,
   output [7:0] read_data,
   input [2:0]write_reg_no,
    input [7:0] write_data,
    input reg_write

    );

    reg [7:0] RegMem [7:0];


assign read_data = RegMem[read_reg_no];


    always@( reset)

    if(reset == 0 )
    begin
    RegMem[0]=0;
    RegMem[1]=1;
    RegMem[2]=2;
    RegMem[3]=3;
    RegMem[4]=4;
    RegMem[5]=5;
    RegMem[6]=6;
    RegMem[7]=7;

    end

always@(write_reg_no,write_data,reg_write)
 if(reg_write)
 RegMem[write_reg_no] =  write_data;

endmodule
```

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

Answer: EX/WB.RegWrite ==1 && EX/WB.Rd == ID/EX.Rs

6. Implement the forwarding unit and copy the image of Verilog code of forwarding unit here.

Answer:

```
module frwdng_unit(
input ex_wb_reg_write,
input [2:0] ex_wb_rd,
input [2:0] id_ex_rs,

output frwd
    );

    assign frwd = ex_wb_reg_write? (ex_wb_rd== id_ex_rs)? 1:0:0;
endmodule
```

7. Implement complete processor in Verilog (using all the Datapath blocks). Copy the image of Verilog code of the processor here. (Use comments to describe your Verilog implementation)

Answer:

```
module processor(
    input clk,
    input reset
    );
    wire [7:0] instr_code;
    wire [7:0]if_id_out;
    wire [7:0] read_data;
    wire reg_write,alu_instr;
    wire [15:0]id_ex_out;
    wire [7:0] alu_out,sgn_extnd_out,s8_out;
    wire [11:0]ex_wb_out;
    wire frwd;
    wire [7:0]sll_out;

 //Instruction fetch block
instr_fetch_blck s1 (clk,reset,instr_code,instr_code[5:0],instr_code[7]);  // branch = instr_code[7]
// IF/ID  pipeline register . Contains only 8-bit instruction code
if_id_pl_reg s2 (clk,reset,instr_code,if_id_out);
//Register file
reg_file s3 (reset,if_id_out[5:3],read_data,ex_wb_out[2:0],ex_wb_out[10:3],ex_wb_out[11]);
// Control unit
ctrl_unit s4 (reset,if_id_out[7:6],reg_write,alu_instr);
// ID/EX pipeline register. Contains the control signals,8-bit read data from register file, register no.,immediate/shift value
id_ex_pl_reg s5 (clk,reset,reg_write,alu_instr,read_data,if_id_out[5:3],if_id_out[2:0],id_ex_out);
// Execution unit. Gives the shifted ouput
ALU s6 (sll_out,id_ex_out[2:0],alu_out);
// Sign extension block. Gives the extended immediate value
sign_extnd_blck s7 (id_ex_out[2:0], sgn_extnd_out);
// Multiplexer that decides whther instruction is Shift or Load immediate instruction
mux_8_bit s8 (alu_out,sgn_extnd_out,id_ex_out[14],s8_out);
// EX/WB pipeline register. Contains Regwrite signal, 8-bit data to be stored in Reg file,register no.
ex_wb_pl_reg s9 (clk,reset,id_ex_out[15],s8_out,id_ex_out[5:3],ex_wb_out);
// Hazard Detection unit
frwdng_unit s10 (ex_wb_out[11],ex_wb_out[2:0],id_ex_out[5:3],frwd);
// Multiplexer thta selects if reg_file data or forwarded data should be sent to execution unit
mux_8_bit s11 (ex_wb_out[10:3],id_ex_out[13:6],frwd,sll_out);
endmodule

```

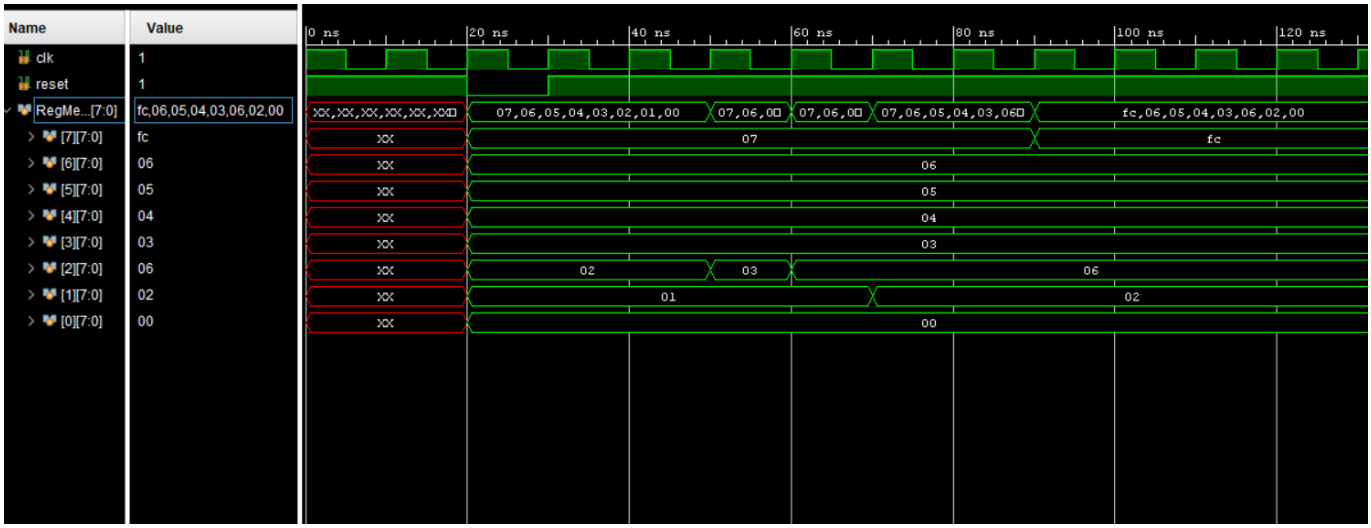
8. Test the processor design by generating the appropriate clock and reset. Copy the image of your testbench code here.

Answer:

```
22
23      module tb(
24
25          );
26
27          reg clk,reset;
28
29          processor uut (clk,reset);
30
31      initial begin
32
33      reset= 1'b1; #10;
34      reset = 1'b1; #10;
35       reset= 1'b0; #10;
36      reset = 1'b1; #104; $finish;
37      end
38
39      always begin
40      clk = 1'b1; #5;
41      clk = 1'b0; #5;
42      end
43
44      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: Had to figure out always block parameters for writing in reg file and which stage PC will be updated in case of the jump instruction

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: Prashanth Reddy **Date: 15/4/2020**
ID No.: 2017AAPS0274H