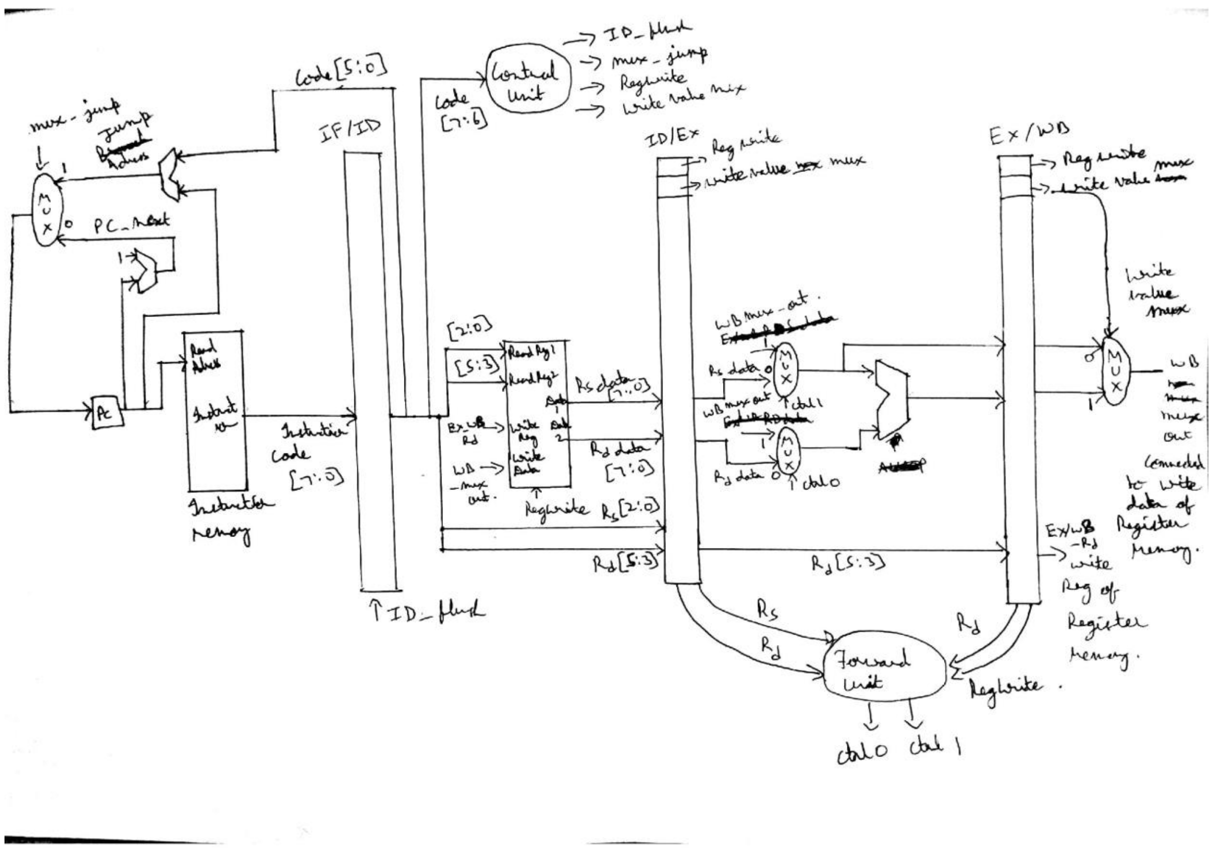
COMPUTER ARCHITECTURE ASSIGNMENT 2

Name: P.SURYACHANDRA PRASAD ID.NO: 2014B2A30949G

1)BLOCK DIAGRAM:



Assumptions:

No AlU ctrl needed as it only does shifting.

Pc+1 is not stored in ID/IF pipeline register as pc can be directly used to calculate the jump address as it will be pc+1 when the control signal for pc are generated , so they can be added .

Tested by changing jump to different locations and introducing reset in middle,worked fine without any wrong output.

2)LIST CONTROL SIGNALS USED AND ALSO THE VALUES OF CONTROL SIGNALS FOR DIFFERENT INSTRUCTIONS.

ANS:

	mux_jump,	ID_flush	regWrite	write_value_mux
MOV	0	0	1	0
SLL	0	0	1	0
J	1	1	0	X

3)IMPLEMENT THE INSTRUCTION FETCH BLOCK.COPY THE IMAGE OF VERILOG CODE OF INSTRUCTION FETCH.

VERILOG CODE:

```
`timescale 1ns / 1ps
module Instruction_Fetch(input clk,input reset,

  input mux_jump, input [5:0] jump_adress,
  output [7:0] Instruction_Code
  );

wire [7:0] jump_extend,jump_adder;
reg [7:0] pc,pc_nex;
```

```
Instruction_Memory ins(pc,reset,Instruction_Code);
assign jump_extend = {jump_adress[5],jump_adress[5],jump_adress};
assign jump_adder = jump_extend + pc;
always @(posedge clk,negedge reset)
begin
if(reset ==0)
  begin
    pc =0;
    pc_nex = pc +1;
  end
else
  begin
    pc = (mux_jump) ? jump_adder : pc_nex;
    pc_nex = pc +1;
  end
end
endmodule
```

4) IMPLEMENT THE REGISTER FILE AND COPY THE IMAGE OF VERILOG CODEOF REGISTER FILE UNIT HERE.

ANS:

```
module Register_file(input [2:0] Read_Reg_Num_1,input [2:0] Read_Reg_Num_2,input [2:0]
Write_Reg_Num,
  input [7:0] Write_Data,
  output  [7:0] Read_Data_1,
  output  [7:0] Read_Data_2,
  input RegWrite,
  input reset);
  reg  [7:0] RegMemory[0:7];
  assign  Read_Data_1=RegMemory[Read_Reg_Num_1];
  assign  Read_Data_2=RegMemory[Read_Reg_Num_2];

  always@(RegWrite,Write_Data,Write_Reg_Num)
  begin
    if(RegWrite)
      RegMemory[Write_Reg_Num]<=Write_Data;
  end
  always@(negedge reset)
  begin
  $readmemh("register.mem",RegMemory);
  end
endmodule
```

5)DETERMINE THE CONDITIONS THAT CAN BE USED TO DETECT DATA HAZARD

ANS:1) REGWRITE OF EX/WB PIPELINE REGISTER == 1

AND EX_WB_Rd_out == EX_ID_Rs_out, (EX/WB DESTINATION REGISTER IS EX/ID SOURCE REGISTER)

OR

2)) REGWRITE OF EX/WB PIPELINE REGISTER == 1

And (EX_WB_Rd_out == EX_ID_Rd_out)

6) IMPLEMENT THE FORWARD UNIT AND COPY THE IMAGE OF VERILOG CODE HERE.

ANS:

<u>VERILOG CODE FOR FORWARD UNIT:</u>

```
module forward_unit(input [2:0] EX_ID_Rs_out,EX_ID_Rd_out,EX_WB_Rd_out,
  input EX_WB_regWrite,output reg [1:0] ctrl);
```

```
 always@(EX_WB_regWrite,EX_WB_Rd_out,EX_ID_Rs_out,EX_ID_Rd_out)
 begin
  if ( EX_WB_regWrite==1)
    begin
      ctrl[0] = (EX_WB_Rd_out == EX_ID_Rs_out) ? 1 : 0;
      ctrl[1] = (EX_WB_Rd_out == EX_ID_Rd_out) ? 1 : 0;
    end
  else
    ctrl = 2'b0;
 end
endmodule
```

7) Implement complete processor in Verilog and copy the image here.

```
module datapath(input clock,reset);

 wire [7:0] Instruction_Code;
 wire mux_jump,ID_flush;
 wire [7:0] IF_ID_inst_code_out;
 wire [2:0] write_reg_num;
 wire [7:0] WB_mux_out,EX_ID_Rsdata_in,EX_ID_Rddata_in;
 wire writeRegWB_path,write_value_mux,regWrite;
 wire [7:0] EX_ID_Rsdata_out,EX_ID_Rddata_out;
 wire [2:0] EX_ID_Rs_out,EX_ID_Rd_out;
 wire EX_ID_write_mux_out,EX_ID_regWrite_out;
 wire [1:0] ctrl;
 wire [7:0] ALU_out,EX_WB_Rsdata_in;
 wire [2:0] EX_WB_Rd_in;
 wire EX_WB_write_mux_in,EX_WB_regWrite_in;
 wire [2:0] EX_WB_Rd_out;
 wire EX_WB_regWrite_out,EX_WB_write_mux_out;
 wire [7:0] ALU_WB_out,EX_WB_Rsdata_out;


 Instruction_Fetch X1(clock,reset,mux_jump,IF_ID_inst_code_out[5:0],Instruction_Code);


 IF_ID X3(clock,reset,Instruction_Code,ID_flush,IF_ID_inst_code_out);


 Register_file X4(IF_ID_inst_code_out[2:0],IF_ID_inst_code_out[5:3],write_reg_num,
 WB_mux_out,EX_ID_Rsdata_in,EX_ID_Rddata_in,writeRegWB_path,reset);


 ID_EX X5(clock,reset,EX_ID_Rsdata_in,EX_ID_Rddata_in,IF_ID_inst_code_out[2:0],
IF_ID_inst_code_out[5:3],write_value_mux,regWrite,EX_ID_Rsdata_out,EX_ID_Rddata_out,
EX_ID_Rs_out,EX_ID_Rd_out,EX_ID_write_mux_out,EX_ID_regWrite_out);
 execution_unit X6(EX_ID_Rsdata_out,EX_ID_Rddata_out,EX_ID_Rs_out,EX_ID_Rd_out,
EX_ID_write_mux_out,EX_ID_regWrite_out,ctrl,WB_mux_out,ALU_out,EX_WB_Rsdata_in,
EX_WB_Rd_in,EX_WB_write_mux_in,EX_WB_regWrite_in);


forward_unit X7(EX_ID_Rs_out,EX_ID_Rd_out,EX_WB_Rd_out,EX_WB_regWrite_out,ctrl);
control X8(IF_ID_inst_code_out[7:6],mux_jump,regWrite,write_value_mux,ID_flush);
EX_WB X9(clock,reset,ALU_out,EX_WB_Rsdata_in,EX_WB_Rd_in,EX_WB_write_mux_in,
EX_WB_regWrite_in,ALU_WB_out,EX_WB_Rsdata_out,EX_WB_Rd_out,EX_WB_write_mux_
out,EX_WB_regWrite_out);



 WB_path X10(ALU_WB_out,EX_WB_Rsdata_out, EX_WB_Rd_out,
                  EX_WB_write_mux_out,EX_WB_regWrite_out,
WB_mux_out,writeRegWB_path,write_reg_num);


endmodule
```

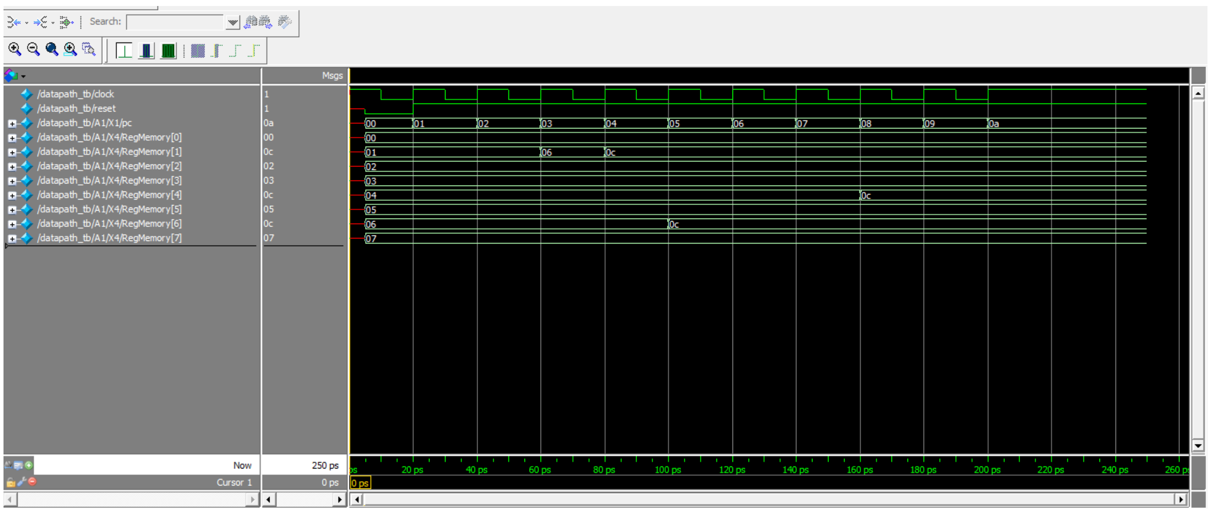
8) Test the design by generating appropriate clock and reset. Copy test bench image here.

Verilog code for test bench:

```
module datapath_tb;
  reg clock;
  reg reset;
  datapath A1(clock,reset);
  initial
  begin
    #5 reset = 0;
    #15 reset =1;
  end
  initial
  begin
    clock =1;
    repeat(20)
    #10 clock = ~clock;
  end
endmodule
```

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

X = 1, y =6,z=4



```
/datapath_tb/clock                  1
/datapath_tb/reset                  1
/datapath_tb/A1/X1/pc               0a
/datapath_tb/A1/X4/RegMemory[0]     00
/datapath_tb/A1/X4/RegMemory[1]     0c
/datapath_tb/A1/X4/RegMemory[2]     02
/datapath_tb/A1/X4/RegMemory[3]     03
/datapath_tb/A1/X4/RegMemory[4]     0c
/datapath_tb/A1/X4/RegMemory[5]     05
/datapath_tb/A1/X4/RegMemory[6]     0c
/datapath_tb/A1/X4/RegMemory[7]     07
Now      250 ps
Cursor 1   0 ps
```

Unrelated question:

What were the problems faced?

No problems faced

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?

Implemented on my own, no help taken.

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: P.SURYACHANDRA PRASAD DATE:20/4/18

ID.NO:2014B2A30949G