Birla Institute of Technology and Science – Pilani, Hyderabad Campus
Second Semester 2020-21
CS F342: Computer Architecture Assignment (30 Marks)

This assignment consists of two parts, Part A and Part B:

A. Implement a Single Cycle processor in Verilog. This processor supports load immediate (li), constant addition (addi) and unconditional jump (J) instructions only. The processor has Reset and CLK as inputs and no outputs. The processor has instruction fetch unit, registerfile (with eight 8-bit registers), Execution and Writeback unit. Read operation on the registerfile is independent of CLK whereas the write happens on the positive edge of the CLK. When reset is activated the PC register is initialized to 0, the instruction memory and register file get loaded by **predefined values.** (Assume 8-bit PC. Also Assume Address and Data size as 8-bits).

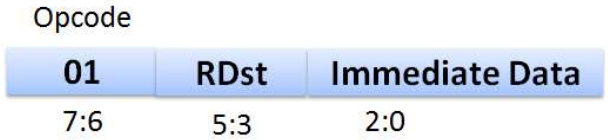
B. (a) Implement 4-stage pipelined processor in Verilog. This processor supports load immediate (li), constant addition (addi) and Unconditional Jump (J) instructions only. The processor should implement forwarding to resolve data hazards. The processor has Reset, CLK as inputs and no outputs. The processor 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 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)

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 adding PC+1 to the Signextended data specified in instruction field (5:0). Opcode for j is 11)

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

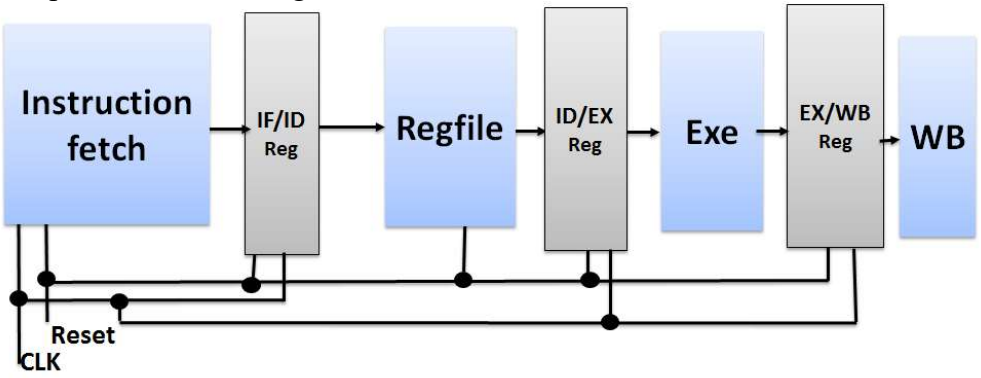
Assume the register file contains 8 registers (R0-R7) each register can hold 8-bit data. On reset register file should get initialized such that R0 = 0, R1 = 1, R2 = 2, R3 = 3 …etc. On reset assume that the instruction memory gets initialized with four instructions.

```
        li Rx, 3
        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),

Note for Pipelined Processor: 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 15-April-2021, 5:00 PM.

Name: Keshav Kabra **ID No:** 2018AAPS0527H

Common Questions applicable to both single cycle and pipelined processor

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

Answer:

```
module Instruction_Fetch(
    input Clk,
    input Reset,
    input PCSrc,
    input [7:0] Imm_Data,
    output [7:0] Instruction_Code
     );

    reg [7:0] PC;
    Instruction_Memory mem1(PC, Reset, Instruction_Code);

    always @(posedge Clk, posedge Reset)
    begin
        if(Reset == 1)
        begin
            PC <= 0;
        end

        else
        begin
            if(PCSrc == 0) // if the Instruction is not Jump
            begin
                PC <= PC + 1;
            end

            else // if the Instruction is Jump
            begin
                PC <= PC + 1 + Imm_Data;
            end
        end
    end

endmodule
```

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

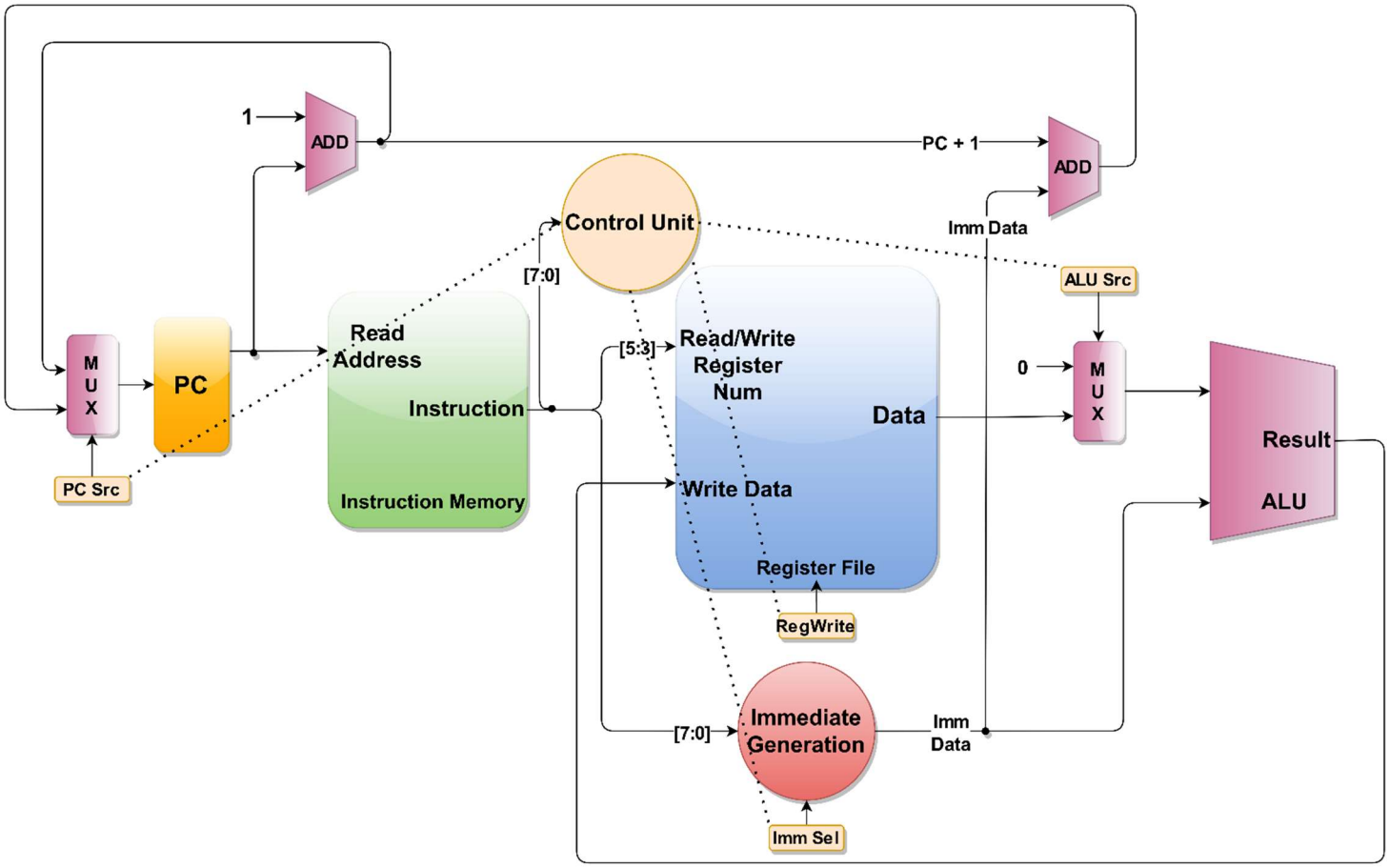
Answer:

Instructions	**Control Signals**					
	PCSrc	ImmSel	ALUSrc	RegWrite		
li	**0**	**0**	**0**	**1**		
addi	**0**	**0**	**1**	**1**		
j	**1**	**1**	**x**	**0**		

Questions applicable to single cycle processor

1. Draw the complete Datapath and show control signals of the single cycle processor.

Answer:



2. Implement complete single cycle 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 Main_Processor(
    input Clk,
    input Reset
    );

   wire PCSrc, RegWrite, ALUSrc, ImmSel; // Control Signals
   wire [7:0] Instruction_Code, Read_Data, Write_Data, Imm_Data;

   Control_Unit Ctrl(Instruction_Code[7:6], PCSrc, RegWrite, ALUSrc, ImmSel);
   // Control Unit Generates the Control Signals from the opcode.

   Instruction_Fetch IF(Clk, Reset,PCSrc, Imm_Data, Instruction_Code);
   // Instr Fetch unit generates the Instr Code, and updates the PC based on Imm_Data.

   Register_File RF(Clk, Reset, Instruction_Code[5:3],
                    Write_Data, Instruction_Code[5:0], RegWrite, ImmSel, Read_Data, Imm_Data);
   // Register File reads data from register and writes the data to the register.

   Execution_Writeback ADD(ALUSrc, Read_Data, Imm_Data, Write_Data);
   // ALU block generates the result to write back.
endmodule
```

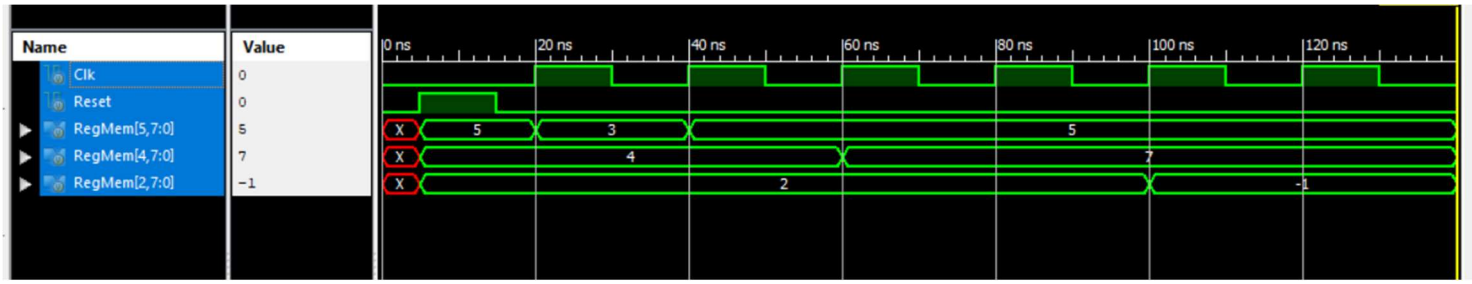
3. **Test the single cycle processor design by generating the appropriate clock and reset. Copy the <u>image</u> of your testbench code here.**

Answer:

```
module test_Main;

    // Inputs
    reg Clk;
    reg Reset;

    // Instantiate the Unit Under Test (UUT)
    Main_Processor uut (
        .Clk(Clk),
        .Reset(Reset)
    );

    initial begin
        Reset = 0;
        #5;
        Reset = 1;
        #10;
        Reset = 0;
    end

    initial begin
        Clk = 0;
        #10
        repeat (12)
        #10 Clk = ~ Clk;
        #10 $finish;
    end

endmodule
```

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

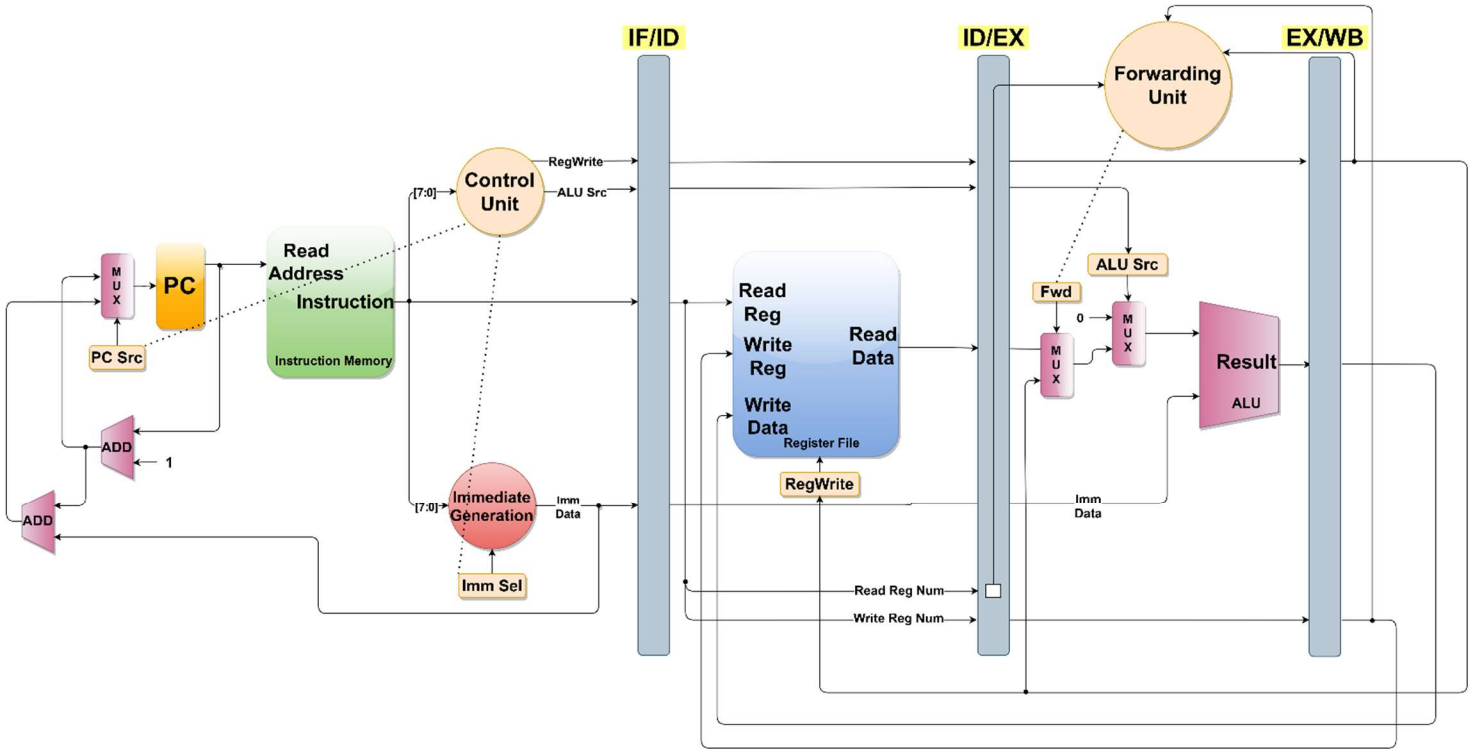
Copy verified **<u>Register file</u>** waveform here (show only the Registers that get updated, CLK, and RESET):



<u>Questions applicable to pipelined processor</u>

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

Answer:

EX/WB.Register_Rd == ID/EX.Register_Rs

&&

EX/WB.RegWrite == 1

i.e. Destination Register Num in EX/WB reg msut match with the Source Register Num in ID/EX reg and RegWrite of EX/WB must be high

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

Answer:

```
module Forwarding_Unit(
   input Clk,
   input [2:0] ID_EX_Read_Reg_Num,
   input [2:0] EX_WB_Write_Reg_Num,
   input EX_WB_RegWrite,

   output Fwd_signal

    );

   assign Fwd_signal = (ID_EX_Read_Reg_Num == EX_WB_Write_Reg_Num && EX_WB_RegWrite == 1) ? 1 : 0;
   // Forwarding signal is assigned with 1 or 0 based on the condition.

endmodule
```

4. **Implement complete pipelined 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 Main_Processor(
   input Clk,
   input Reset
    );

    wire PCSrc, ImmSel, ALUSrc, RegWrite
    wire IF_ID_RegWrite, IF_ID_ALUSrc, ID_EX_RegWrite, ID_EX_ALUSrc, EX_WB_RegWrite, Fwd_signal;
    wire [7:0] Instruction_Code, Imm_Data, IF_ID_Instruction_Code, IF_ID_Imm_Data, ID_EX_Imm_Data;
    wire [7:0] Read_Data, ID_EX_Read_Data;
    wire [7:0] Write_Data, EX_WB_Write_Data;
    wire [2:0] Write_Reg_Num, ID_EX_Write_Reg_Num, EX_WB_Write_Reg_Num, Read_Reg_Num, ID_EX_Read_Reg_Num;

    Instruction_Fetch IF(Clk, Reset, PCSrc, ImmSel, Instruction_Code, Imm_Data);
    // Instr Fetch unit generates the Instr Code, and updates the PC based on Imm_Data and PCSrc.

    Control_Unit CU(Instruction_Code, PCSrc, ImmSel, ALUSrc, RegWrite);
    // Control Unit Generates the Control Signals from the opcode.

    IF_ID_Reg IFID(Clk, Reset, RegWrite, ALUSrc, Instruction_Code, Imm_Data, IF_ID_RegWrite,
                   IF_ID_ALUSrc, IF_ID_Instruction_Code, IF_ID_Imm_Data );

    assign Read_Reg_Num = IF_ID_Instruction_Code [5:3];
    assign Write_Reg_Num = IF_ID_Instruction_Code [5:3];

    Register_File RF(Reset, Read_Reg_Num, EX_WB_Write_Reg_Num, EX_WB_Write_Data, EX_WB_RegWrite, Read_Data);
    // Register File reads data from register and writes the data to the register.

    ID_EX_Reg IDEX(Clk, Reset, IF_ID_RegWrite, IF_ID_ALUSrc, Read_Data, IF_ID_Imm_Data,
                   Read_Reg_Num, Write_Reg_Num, ID_EX_RegWrite, ID_EX_ALUSrc,
                      ID_EX_Read_Data, ID_EX_Imm_Data, ID_EX_Read_Reg_Num, ID_EX_Write_Reg_Num);
    // Read Reg Num and Write Reg Num are given to the ID/EX register

    ALU ADD(Clk, Reset, ID_EX_ALUSrc, ID_EX_Read_Data, ID_EX_Imm_Data, EX_WB_Write_Data, Fwd_signal, Write_Data);
    // ALU block generates the result to write back and forwards it to the EX/WB register

    EX_WB_Reg EXWB(Clk, Reset, ID_EX_RegWrite, Write_Data, ID_EX_Write_Reg_Num, EX_WB_RegWrite,
                   EX_WB_Write_Data, EX_WB_Write_Reg_Num);

    Forwarding_Unit Fwd(Clk, ID_EX_Read_Reg_Num, EX_WB_Write_Reg_Num, EX_WB_RegWrite, Fwd_signal );
    // Forwarding Unit generates the Fwd signal as an input to ALU block
endmodule
```

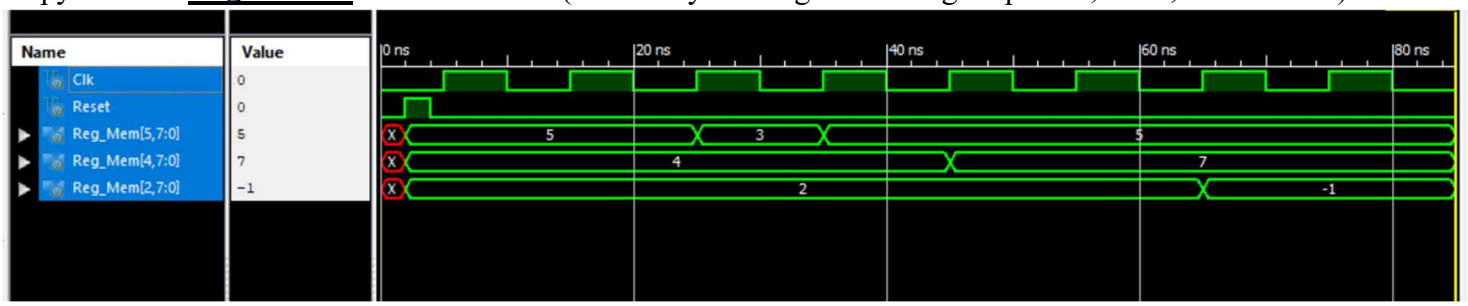
5. **Test the pipelined processor design by generating the appropriate clock and reset. Copy the image of your testbench code here.**

Answer:

```
module Test_Main;

   // Inputs
   reg Clk;
   reg Reset;

   // Instantiate the Unit Under Test (UUT)
   Main_Processor uut (
      .Clk(Clk),
      .Reset(Reset)
   );

   initial begin
      Reset = 0;
      #2;
      Reset = 1;
      #2;
      Reset = 0;
   end

   initial begin
      Clk = 0;
      repeat (16)
      #5 Clk = ~ Clk;
      #5 $finish;
   end

endmodule
```

6. 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: The main problem was to debug the Verilog modules to find the error. Other Problems included the wrong usage of always and assign blocks, which was giving extra Clock Cycle delays in the processing.

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: It was really interesting to read upon the slides and lectures again and again to clarify the concepts to implement the design. Yes, I implemented the Processor on my own with help from the Lab 8 and 9 assignments.

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: Keshav Kabra **Date:** 15-04-2021
ID No.: 2018AAPS0527H