CSE 112:		IIIT-Delhi, Winter 2023
Computer Organization	Quiz II Rubric	
Duration : 45 Minutes		
Date: May 12, 2023		Total: 25 Points

Guidelines

If you found any ambiguity in any of the questions or there appears to be a lack of information, then write an assumption on the answer sheet to explain your side interpretation of the problem and solve accordingly.

Note: *There can be some typos in the document. Please feel free to notify.*

Problem I: Instruction Set Architecture and Type of Instruction Classification **0.5x10 = 5 Points**

Note*: Students need to answer any 10 choices.*

Classify the below listed instructions in

a) Control Instructions.
b) Memory Instructions.
c) Arithmetic & Logical Instructions.
d) Miscellaneous instructions.

Instruction (Mnemonic, Operands)	Operation Performed	Type of Instruction
SUB R1, R2, R3	Subtract the content of R3 from R2 and store the result in R1.	Arithmetic & Logical Instruction
BNE R2, Address	Branch to the location specified by the address, if content of R2 is not equal to zero.	Control Instructions.
LD R1, Address	Load the content of the specified address in register R1.	Memory Instructions
ST R4, Address	Store the contents of register R4 to the specified memory address.	Memory Instructions
HALT	Halt(Stop) the program execution.	Miscellaneous instructions, Control Instruction
ISB	Instruction synchronization barrier(flushes the processor pipeline and all subsequent instructions will be fetched from cache).	Miscellaneous Instruction
LSL R1, #02	Left shift content of register R1 by 2-bits.	Arithmetic & Logical Instruction
NOP	Performs no operation.	Miscellaneous Instruction

Mention the ISA class followed by each instruction(e.g. Accumulator, Register-Memory, Register-Register).

Instruction (Mnemonic, Operands)	Operation performed	ISA class followed while designing Instruction.
ADD R7	Add the content of Accumulator, R7 and store the result in R1 (Accumulator = Accumulator + R7).	Accumulator Class
SUB R1, R2, R3	Subtract R3 from R2 and store the result in R1 (R1 = R2 - R3).	Register-Register class
ADD R1, R2, [R3]	Add the content pointed by register R3 with register R2 and store the result in R1.	Register-Memory class

Problem III: Assembly Program Encoding and ISA evaluation **4x2 + 2 = 10 Points**

Write two assembly programs while following two different ISA as mentioned-below and comment on the below specified evaluation parameter.

Algorithm of the program to be converted : Factorial Calculation of a given number.

```
- - - - - - - - - - - - - - - - - Program pseudo code begins - - - - - - - - - - - - - - - - -
temp_number = number_whose_factorial_is_to_be_calculated
multiplied_number = 1
while(temp_number != 0)
        multiplied_number = multiplied_number * temp_number
        // multiply the current number with the old multiplication result starting from factorial_number(to create series of multiplication)
        temp_number = temp_number - 1
        // decrement the factorial_number by 1
Store the result(multiplied_number) to any arbitrary memory location.
- - - - - - - - - - - - - - - - - Program pseudo code ends - - - - - - - - - - - - - - - - -
```

Note*: Students need to follow the above-mentioned factorial calculation algorithm to write the assembly program.*

Instruction Set A		Instruction Set B	
Instruction (Mnemonic, Operands)	Operation Performed	Instruction (Mnemonic, Operands)	Operation Performed
SUB R1, #Imm	Subtract the content of specified register(R1) by immediate.	DCR	Decrement the content of Accumulator by 1.
BNZ R1, Address	Branch to address if the content of specified register(R1) is not equal to zero.	JNZ Address	Jump to the specified address if the zero flag is not set.
MUL R1, R2	Multiply reg R1, R2 and store the result into the left_hand_side specified register(R1).	MVI #Imm	Copy the specified immediate value in the Accumulator.
MOV R1, R2	Copy the content of rigth_hand side specified register(R2) into left_hand_side specified register(R1).	MUL R1	Multiply accumulator with the specified register(R1) and store the result in Accumulator.
LD R1, Address	Load the content specified at address into specified register(R1).	MOV [MEM], A	Copy the content of Accumulator to the specified memory location(MEM: memory address).
ST R1, Address	Store the content of register R1 to the specified address.	MOV A, [MEM]	Copy the memory content into the accumulator (MEM: memory address).
MVI R1, #Imm	Copy the Immediate value into the specified register(R1).	CMP #Imm	Compare the immediate value with the accumulator content and set the zero flag if results are equal otherwise reset the zero flag.
		***MOV reg1, reg2**	Copy the content of specified(reg2) in specified(reg1).

Evaluation Parameter: Which Program needs more number of instructions for the specified program and why.
***Note**: There is no restriction on the number of general purpose registers available in both the ISA. HALT instruction is available in both ISA. And, the number whose factorial is to be calculated is stored at any arbitrary chosen memory location.*
*** Hint** : Specified register can be any general purpose register or Accumulator.

Solution:

S. No	Assembly Program Using Instruction Set A	S. No	Assembly Program Using Instruction Set B
0	Ld R1, Fact_No_Addr	0	MVI #1
1	MVI R2, #1	1	MOV R!, A
2	BNZ R1, loop	2	MOV A, [Fact_No_Addr]
3	MVI R1, #1	3	CMP #0
4	ST R1, Dest_Addr	4	JNZ loop
5: loop	MUL R2, R1	5	MVI A, #1
6	SUB R1, #1	6	MOV [Dest_Addr] A
7	BNZ R1, loop	7	HALT
8	ST R1, Dest_Addr	8: loop	MOV R2, A
9	HALT	9	MOV A, R1
		10	MUL R2
		11	MOV R1, A
		12	MOV A, R2
		13	DCR
		14	CMP #1
		15	JNZ loop
		16	MOV A, R1
		17	MOV [Dest_Addr] A
		18	HALT

Evaluation Note : The above mentioned program is just one possibility. A lot more possibilities are there. But while realizing the above-specified algorithm we will always find that instruction set B will have more instructions.This is due to that In Instruction set B only Accumulator/Working Register have the power to perform any arithmetic or logic operations.

<u>Problem II: Assembly Program Execution</u> **3x3 + 1 = 10 Points**

Instruction set specified for the below problem.

Instruction (Mnemonic, Operands)	Shorthand explanation	Operation performed
ADD R1, R2, R3	Add Operation.	Add the content of register R2, R3 and store the result into register R1.
MOV R1, #Imm	Move(copy) immediate instruction.	Move(copy) the specified immediate value into specified register R1.
BL Address	Branch and Link.	Branch to the mentioned address and store the return address(address of next instruction) in the Link register.
BGL R1, R2, Address	Branch if greater and link.	Branch to address and also store the return address in Link register, if content of register R1 is greater then content of register R2.
BNE R4, R5, Address	Branch if not equal.	Branch to the specified address if the content of registers R4, R5 are not equal.
HALT	Halt the program execution.	Stop the program execution immediately.
RET	Return from the current subroutine.	Jump to the memory address present in the Link register.

Note: *All of the general purpose registers are initialized with their register numbers. For example R0 contains "0", R9 contains "9". And all representations are in decimal format. Please note that there is no stack available to us.*

Question : Write the content of registers R6, R7 every time we enter into a subroutine and everytime we exit out from a subroutine.Also mention each content stored in Link register till the program encounters a "HALT" instruction. Also mention how many loop iterations are there.

Evaluation Note*: There is a pattern of the register content and for the correct pattern of each register(R6,R7,LR) you will be awarded 3-points.*

```
- - - - - - - - - - - - - - - - - Assembly program code begins - - - - - - - - - - - - - - - - -
BGL R7, R6, label1
BGL R6, R7, label2
HALT
label1:
BGL R5, R4, label3
ADD R6, R2, R3
BGL R7, R6, label4
BL label5
HALT
label4:
MOV R7, #27
MOV R6, #24
RET
label3:
MOV R1, #01
MOV R3, #00
MOV R2, #08
label6:
ADD R3, R1, R3
BNE R2, R3, label6
RET
label5:
MOV R7, #12
RET
label2:
MOV R6, #15
RET
HALT
- - - - - - - - - - - - - - - - - Assembly program code ends- - - - - - - - - - - - - - - - -
```

Solution:

Memory Address	Given Program	Execution Flow (Left to right)					
0	BGL R7, R6, label1	**BGL R7, R6, label1**					
1	BGL R6, R7, label2						
2	HALT						
3: label1	BGL R5, R4, label3		R4 = 4, R5 = 5 R6 = 6, R7 = 7 LR = 1. **BGL R5, R4, label3**				
4	ADD R6, R2, R3				R2 = 8, R3 = 8 R6 = 6, R7 = 7 LR = 4. **ADD R6, R2, R3**		
5	BGL R7, R6, label4				BGL R7, R6, label4		
6	BL label5				**BL label5** R2 = 8, R3 = 8 R6 = 16, R7 = 7 LR = 7.		
7	HALT						R2 = 8, R3 = 8 R6 = 16, R7 = 7 LR = 7. **HALT**
8: label4	MOV R7, #27						
9	MOV R6, #24						
10	RET						
11: label3	MOV R1, #01			R7 = 7, R6 = 6 LR = 4. **MOV R1, #01**			
12	MOV R3, #00			MOV R3, #00			
13	MOV R2, #08			MOV R2, #08			
14: label6	ADD R3, R1, R3			ADD R3, R1, R3			
15	BNE R2, R3, label6			BNE R2, R3, label6			
16	RET			**RET** R1 = 1, R2 = 8, R3 = 8, R6 = 6, R7 = 7, LR = 4.			
17: label5	MOV R7, #12					R2 = 8, R3 = 8 R6 = 16, R7 = 7 LR = 4. **MOV R7, #12**	
18	RET					RET	
19: label2	MOV R6, #15						
20	RET						
21	HALT						

Table Explanation : As the program is executing we are going from left to right. Each label as per their location in memory is assigned to a specific memory address. Whenever we enter into any new routine and leave a routine then the content of the concerned register is shown.

Loop Iterations: Iteration in a loop is counted when we get back to the same initial statement inside the loop from where we started. But if there is a break in that loop or we never start again from the loop's first statement then there is no iteration even if it is written as a loop.
In our example we encounter a loop when we jump to **"label3"**.And, the branch at Address **"15"** is responsible for that loop.

LOOP AT LABEL3 : This loop will iterate till the value of Register **R3** is not equal to register **R2**. And the loop has 8 iterations(As **R2** becomes **8** starting from **0**).