Midsem
CSE 112 - Computer Organization

INSTRUCTIONS:

Total Marks = 40
Time Duration = 60 mins solving + 15 mins uploading

1. **Duration of the exam is 60 mins, and 15 mins for scanning and uploading the solutions. No further extension of time will be given regarding this. Any late submission will be awarded 0 mark.**
2. **Question paper will be uploaded in the google classroom. Do not forget to turn in. Solutions submitted by any other means (email etc.) won't be considered for evaluation. Please upload your submission only on the classroom page of the section you are enrolled in. If you upload to the wrong section's page, you will be awarded 0 marks.**
3. **Students are required to switch on their cameras and mute themselves. Make sure you are sitting in a well-lit room so that we are able to see your faces clearly. If you are not clearly visible, you will be awarded 0 marks.**
4. **The answers should be in your own handwriting and submission should be in PDF format only.**
5. **Write any assumption clearly, if any. Needless to say, only reasonable assumptions will be considered if any ambiguity is found in the question.**
6. **During the exam if you have any query, write it in the meet chat box. It will be taken into notice by us. Don't unnecessarily unmute your mic for it as it creates disturbance to others.**
7. **Calculators are NOT allowed during the exam time. ONLY use pen and paper for writing the exam.**
8. **Students need to be present and visible for the whole exam duration (till the end of solution uploading time) even if they upload the solution before time.**
9. NAMING CONVENTION **-** <Name>_<Roll number>.pdf

GOOD LUCK !!

Q1: Consider the following computations:

a. 13 + (-10)
b. 9 + (-14)

i) Write the 2’s complement representation of both the operands.

[3 for part a + 3 for part b = 6 marks]

ii) Perform the computation in 2’s complement form. Show your computation.

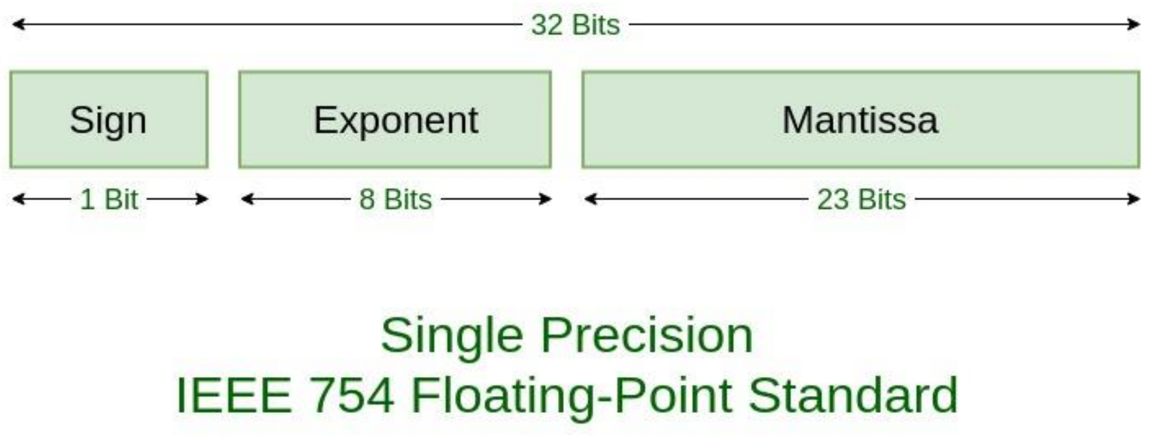
[3 for part a + 3 for part b = 6 marks]

Q2: Convert the following numbers represented in IEEE754 Single Precision Floating Point notation into decimal. (‘_’ are added for clarity)

a. 1000_0000_0000_0000_0000_0000_0000_0000
b. 1111_1111_1000_0000_0000_0000_0000_0000
c. 0100_0001_1011_1100_0000_0000_0000_0000

[3 x 2 = 6 marks]

The format of IEEE754 Single Precision Floating Point number is given below for your reference.



ISA for Q3 and Q4: (This is a modified version of the ISA which you encountered in tutorial 5)

The instructions supported by the ISA are mentioned in the table below. The ISA has **4 general purpose registers: R0 to R3**

Name	**Semantics**	**Syntax**
Add	Performs reg1 = reg2 + reg3	add reg1, reg2, reg3
Sub	Performs reg1 = reg2 - reg3	sub reg1, reg2, reg3
Branch if zero	Branch to addr if register value (reg) is zero	bz reg, addr
Branch if not zero	Branch to addr if register value (reg) is not zero	bnz reg, addr
Mov	Moves imm, an immediate value, to reg	mov reg, #imm
Multiply	Performs reg1 = reg2 x reg3	mul reg1, reg2, reg3

Assume that every register has undefined value at the beginning.

Q3: Write an assembly program, using the ISA described above. The program takes the value stored in the **R0** register, calculates its factorial and stores the value in **R3**.

You can assume that all multiplication and addition operations will never overflow.

Note that 0! = 1, and the factorial of a positive integer n is defined as n! = n x (n-1) x (n-2) x … x 2 x 1.

[10 marks]

Q4:

a.) Trace the execution of the following program by writing the content of all the 4 registers **after the current instruction is executed**. Write "**-**" if the value of a field is undefined. If the program ends, fill all the cells in the next row with **END**. Also fill in the address of the executed instruction. The instruction address is denoted in red for every instruction.

[10 marks]

Instruction Address Instruction

```
            1       mov R0, #0
            2       mov R1, #2
            3       mov R2, #10
            4       mov R3, #1
            5       loop:   add R0, R0, R2
            6               sub R1, R1, R3
            7               bnz R1, loop
                    loop_end:
```

Answer in a tabular fashion in the format given below. Add/Delete rows as needed. The content of the registers after executing the first instruction is already filled up for you as an example.

Instruction Address	**R0**	**R1**	**R2**	**R3**
1	0	-	-	-

b.) Write a new assembly program based on the program given in Q4a. Keep the first 4 instructions exactly the same. Replace instructions 5, 6 and 7 (the loop) by **a *single* instruction**, such that the new program stores the same value in R0 as the older program, at the end of the program execution. The value of registers R1, R2 and R3 can differ from the original program at the end of execution. The single instruction must not be a mov instruction.

[2 marks]