Ameya Barapatre
Roll No-06
SE-DS

Assignment -I

Q1 Represent $(-7.14)_{10}$ using IEEE 754 standards for both single and double precision format.

To represent the decimal number -7.14 in IEEE format, we need to convert it into both single precision (32-bit) and double precision (64-bit) formats. ~~Here is how to d~~

Convert to Binary →

Convert the Integer Part → The integer part of -7.14 is -7.

7 ~~is~~ in binary is 111.

Convert the Fractional Part : The fractional part is 0.14

To convert 0.14 ~~to~~ to binary, multiply by 2 and take the integer part:

$0.14 \times 2 = 0.28 \rightarrow 0$
$0.28 \times 2 = 0.56 \rightarrow 0$
$0.56 \times 2 = 1.12 \rightarrow 1$
$0.12 \times 2 = 0.24 \rightarrow 0$
$0.24 \times 2 = 0.48 \rightarrow 0$
$0.48 \times 2 = 0.96 \rightarrow 0$
$0.96 \times 2 = 1.92 \rightarrow 1$
$0.92 \times 2 = 1.84 \rightarrow 1$

This 0.14 approximately equals 0.00100011_2

Combining the Integer and fractional parts →
Therefore, -7.14 in binary is -111.001001_2

Normalize the Binary Number →
$-111.001001_2 = -1.11001001 \times 2^2$

Determine Sign, Exponent and Mantissa →

Sign Bit → Since the number is negative the sign bit is 1.

Exponent →

- In IEEE 754, the exponent is biased

for single precision, the bias is 127:
⇒ Exponent = 2 + 127 = $129_{10} = 10000001_2$

for double precision, the bias is 1023:
⇒ Exponent = 2 + 1023 = $1025_{10} = 10000000001_2$

Mantissa →
The mantissa in IEEE 754 does not include the leading 1:
From 1.11001001 we take 11001001 and fill to the right with zeros →

for single precision (23 bits): 11001001000000000000000

For double precision (52 bits):
1100100100000000000000000000000000000000000000000000

Assemble the final Representation:

Single Precision (32-bit): {[Sign | Exponent | Mantissa]}

=> 1 | 10000001 | 11001001100000000000000

=> | 1 | 10000001 | 11001001100000000000000 |

Final Representation -> 11000000111001001100000000000000

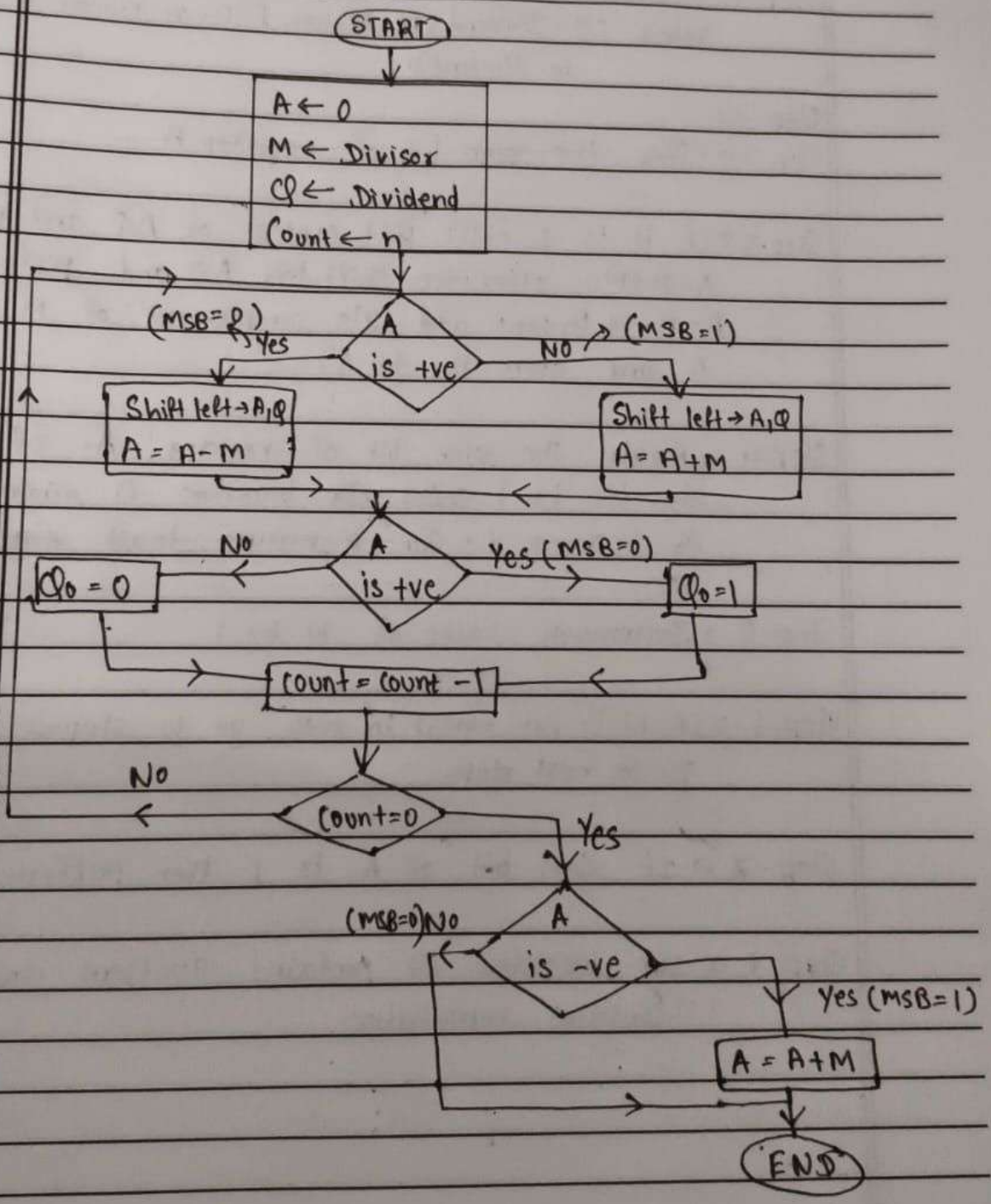
Double Precision (64-bit) ->

| 1 | 10000000001 | 1100100110000000000000000000000000000000000000000000 |

Final Representation ->
1100000000011100100110000000000000000000000000000000000000000000

These are the IEEE 754 representations of -7.14 in single and double precision formats.

Q2. Illustrate the flowchart of Non-Restoring division algorithm and perform 10/3.



Steps involved in Non-Restoring Division Algorithm→

Step-1 ⇒ First the registers are initialized with corresponding values (Q = Dividend, M = Divisor, A = 0, n = number of bits in dividend).

~~Step 2:~~

Step 2 ⇒ Check the sign bit of register A.

Step-3 ⇒ If it is 1 shift left content of AQ and perform A = A + M, otherwise shift left AQ and perform A = A − M (means add 2's complement of M to A and store it to A)

Step-4 → Again the sign bit of register A. If sign bit is 1 ~~Q[0]~~ Q_0 becomes 0 otherwise Q_0 becomes 1. ~~Q_0 : 0 means least significant bit~~

Step-5 → Decrements value of N by 1.

Step-6 → If N is not equal to zero go to step-2 otherwise go to next step.

Step-~~7~~ → If sign bit of A is 1 then perform A = A + M

Step 8 → ~~If~~ Register Q contains quotient and A contains remainder.

Non-Restoring division Algorithm for 10/3 →

Dividend = 10 = Q
Divisor = 3 = M
A = 0
n = 5

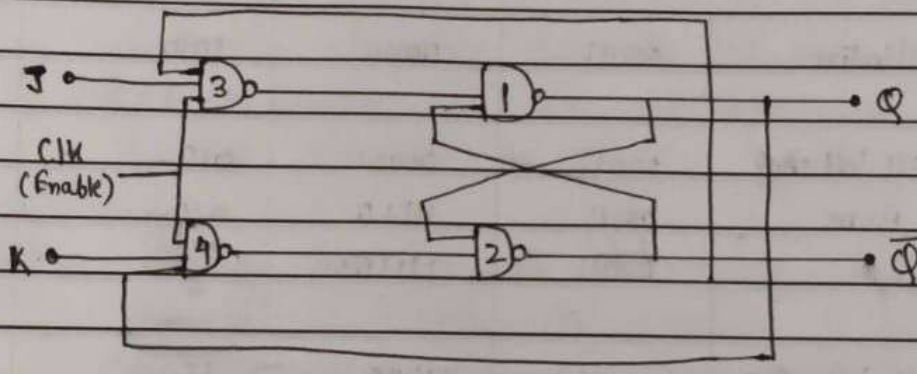
Operation	M	A	Q	n
Initialize	00011	00000	1010	4
Shift left → A,Q	00011	00001	010–	4
A = A - M	00011	11110	010–	
$Q_0 = 0$	00011	11110	0100	
Shift left → A,Q	00011	11100	100–	3
A = A + M		11111	100–	
$Q_0 = 0$		11111	1000	
Shift left → A,Q	00011	11111	000–	2
A = A + M		00010	000–	
$Q_0 = 1$		00010	0001	
Shift left → A,Q	00011	00100	001–	1
A = A - M		00001	001–	
$Q_0 = 1$		00001	0011	

Register Q contain the quotient 3 and register A contain remainder 1.

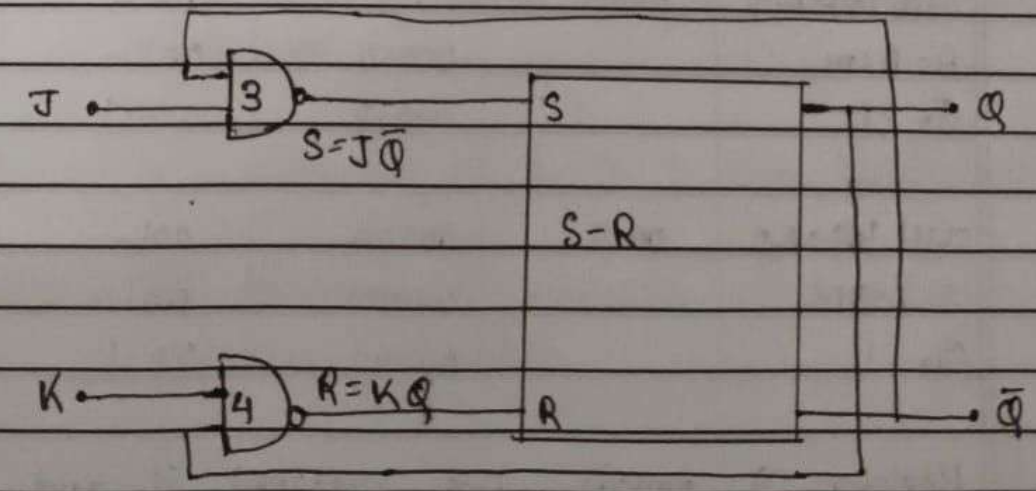
Q3. Brief the working principle of JK flip-flop with diagram and truth table.

J-K Flip Flop →

Consists of basic S-R latch and an enable input. It is also called as level triggered. A J-K flip-flop using NAND gate is shown in the figure given below →



The outputs Q and $\bar{Q}$ have been fed back and connected to the inputs of gate 4 and 3 respectively.



Case 1 → When J=0 and K=0 then G_3 and G_4 are disabled and there is no change in the output state of flipflop.

Case 2 → When J=1 and K=0 then G_3 will be enabled and G_4 is disabled. G_3 depends on $\overline{Q}$. If Q=1 and $\overline{Q}$=0, G_3 is disabled, so the output state of flipflop Q_{n+1} remains same as before. If Q=0 and $\overline{Q}$=1, G_3 is enabled and high level signal goes to S terminal which changes the output state Q_{n+1} to high Q_{n+1}=1.

Case 3→ When J=0 and K=1 the G_4 will be enabled and G_3 is disabled. G_4 depends on Q. If Q=0 and $\overline{Q}$=1 G_4 and G_3 are disabled so the output of flipflop Q_{n+1} remains same as before If Q=1, and $\overline{Q}$=0, G_4 is enabled and high high level signal Q_{n+1} to low, Q_{n+1}=0.

Case 4 → When J=1, K=1 then G_3 and G_4 are enabled and the output depends on flipflop before arrival of clock pulse. Suppose Q_n=1 then it makes G_4 enabled. It sends high level to input terminal R at the arrival of clock pulse and the flip flop gets reset. Output is same as $\overline{Q_n}$.

J	K	Q	$\bar{Q}$	$S = J\cdot\bar{Q}$	$R = KQ$	Q_{n+1}	
0	0	0	1	0	0	0	(Q_n)
0	0	1	0	0	0	1	
0	1	0	1	0	0	0	(Set)
0	1	1	0	0	1	0	
1	0	0	1	1	0	1	(Reset)
1	0	1	0	1	0	1	
1	1	0	1	1	0	1	($\bar{Q}_n$)
1	1	1	0	1	1	0	