TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A LAB REPORT
ON

Division of two unsigned integers by nonrestoring methods.

Lab No: 5
Experiments Date:
Submission Date:

Submitted By:
Name: Nabin Khanal
Group: (076 BCT 036)
Roll No: Group B

Submitted To:
Department of
Electronics and
Computer
Engineering

TITLE: DIVISION OF TWO UNSIGNED INTEGER BINARY NUMBERS

OBJECTIVE:

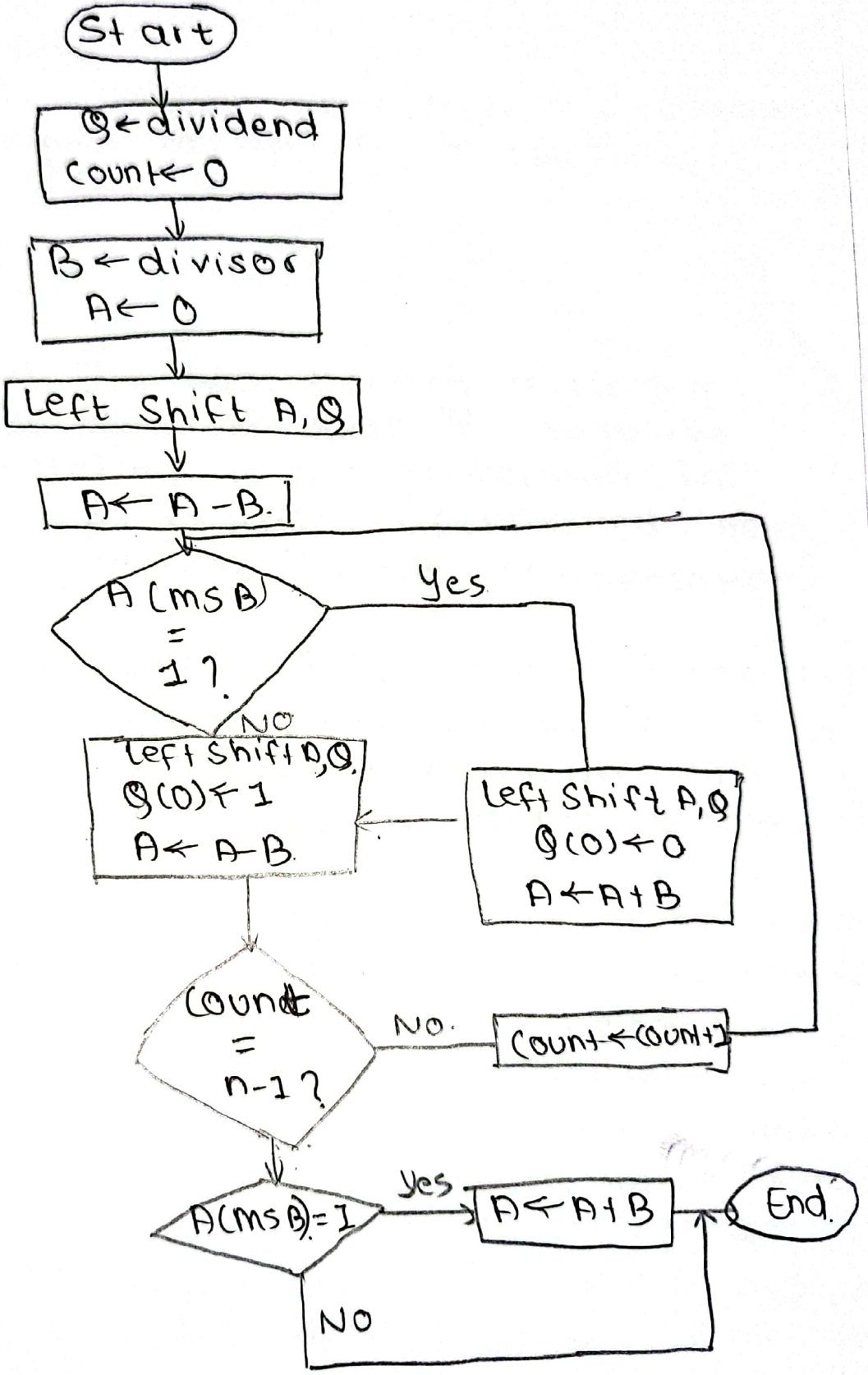
To implement non-restoring division algorithm in digital computer.

THEORY:

In the non-restoring algorithm, if the result of A-M is negative, the restoring is not performed. In next step addition is performed instead of subtraction for the compensation.

Let the number of bits stored in register Q is n Registers AQ is now shifted to the left with zero insertion into Q LSB. Initialize the counter to zero value. And divisor is subtracted by adding 2's complement value. If AMSB =1, set QLSB with value 0 then increment the counter by value 1. The partial remainder is shifted to the left and then B is added to the partial remainder. If AMSB =0, set QLSB with value 1 and then increment counter value by 1. Process is repeated until count = n-1 ;i.e all quotient bits are formed If the last bit of the quotient is 0, the partial remainder must be restored to obtain final correct answer.

The flow chart for non restoring division can be drawn as:



Example:

7 ÷ 3. $-3 \rightarrow 1101$

	A	Q	M.
Initially	0000	0111	0011
Shift	0000	111_	
Subtract	1101		
	1101	1110	
Shift	1011	110_	
Add	0011		
	1110	1100	
Shift	1101	100_	
Add	0011		
	0000	1001	
Shift	0001	001_	
Subtract	1101		
	1110		
Restore	0001	0010	

∴ Quotient: 0010 (2)
Remainder: 0001 (1)

Source Code:

```
from difference import subtract
from sum import add

def shift (A,Q)
    return A[1:] + Q[0], Q[1:] + '_'

def divide (dividend, divisor, n):
    should_add = false
    A = "".zfill(n)
    Q = dividend
    M = divisor

    for i in range(n):
        A, Q = shift(A, Q)
        if (should_add):
            A = add (A,M,n)
            if (len(A)>n):
                A=A[1:]

            else:
            else:
                A=subtract(A,M,n)

            if (A[0] == '1'):
                Q = Q [:len(Q)-1] + '0'
                should_add = True

            else:
                Q = Q [: len(Q)-1] + '1'
                should_add = False.

    return Q, A
```

```
def main():
    n= int (input(" Enter the no of bits: "))

    n1 = input(" Enter the first number: ")
    n2= input(" Enter the second number: ")

    n1= n1.zfill(n)
    n2= n2.zfill(n)

    print (divide(n1,n2,n))

main()
```

Output:

```
Enter the number of bits: 4
Enter the first number: 0111
Enter the second number: 0011

('0010','00011')
```

DISCUSSION AND CONCLUSION

Non Restoring is fast compared to restoring algorithm because it contains at max ~~only~~ one restoration. But it only works for unsigned numbers For ~~a~~ for signed numbers restoring algorithm must be applied

In this way we implemented and executed the non restoring algorithm for integer division of binary numbers