Batch: B2 **Roll No.:** 16010121110	
Experiment / assignment / tutorial No. 2	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

TITLE: To study and implement Booth's Multiplication Algorithm.

AIM: Booth's Algorithm for Multiplication

Expected OUTCOME of Experiment: (Mention CO/CO's attained here)

CO1- Describe and define the structure of a computer with buses structure and detail working of the arithmetic logic unit and its sub modules

---

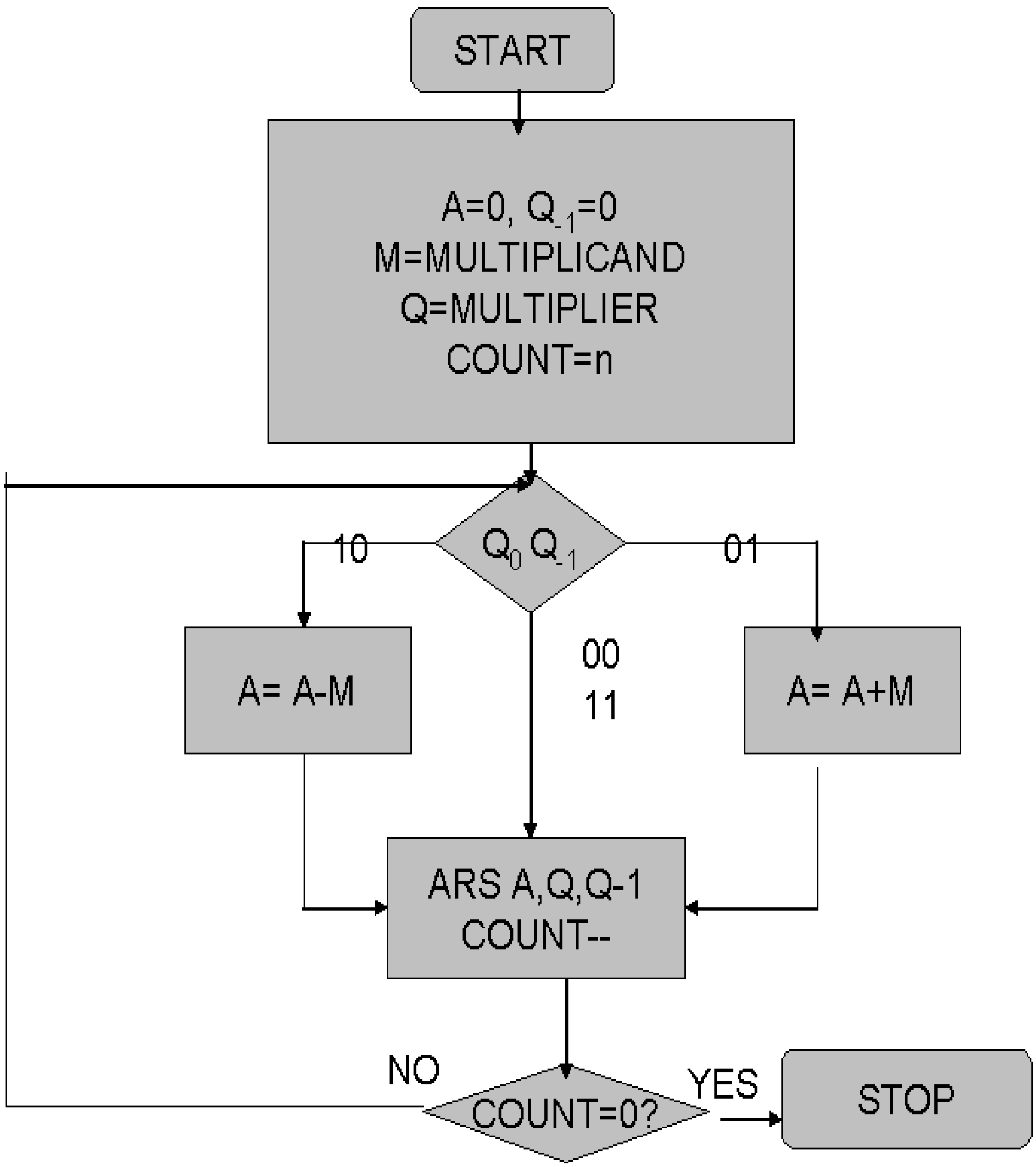
Books/ Journals/ Websites referred:

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, "Computer Organization", Fifth Edition, TataMcGraw-Hill.
2. William Stallings, "Computer Organization and Architecture: Designing for Performance", Eighth Edition, Pearson.
3. Dr. M. Usha, T. S. Srikanth, "Computer System Architecture and Organization", First Edition, Wiley-India.

Pre Lab/ Prior Concepts:

It is a powerful algorithm for signed number multiplication which generates a 2n bit product and treats both positive and negative numbers uniformly. Also the efficiency of the algorithm is good due to the fact that, block of 1's and 0's are skipped over and subtraction/addition is only done if pair contains 10 or 01

Flowchart:


START
A=0, Q-1=0
M=MULTIPLICAND
Q=MULTIPLIER
COUNT=n
Q0 Q-1
10
01
00
11
A= A-M
A= A+M
ARS A,Q,Q-1
COUNT--
NO
COUNT=0?
YES
STOP

Design Steps:

1. Start
2. Get the multiplicand (M) and Multiplier (Q) from the user
3. Initialize A= Q_{-1} =0
4. Convert M and Q into binar
5. Compare Q_0 and Q_{-1} and perform the respective operation.

Q_0 Q_{-1}	**Operation**
00/11	Arithmetic right shift
01	A+M and Arithmetic right shift
10	A-M and Arithmetic right shift

6. Repeat steps 5 till all bits are compared
7. Convert the result to decimal form and display
8. End

Example: (Handwritten solved problem needs to be uploaded)

6 x 3 using Booth Algo.

6 → 0 1 1 0

3 → 0 0 1 1

-3 → 1 1 0 1

4 step process.

(0) 0 0 0 0 0 1 1 0 0

(1) 0 0 Present, So No-OP

Shift right

0 0 0 0 0 0 1 1 0

(2) 1 0 Present, So subtract

$$\begin{array}{r} 0\,0\,0\,0 \\ +\ 1\,1\,0\,1 \\ \hline 1\,1\,0\,1 \end{array}$$

1 1 0 1 0 0 1 1 0

Shift right

1 1 1 0 1 0 0 1 1

(3) 11 , So No - OP

Shift right

1111 0100 1

(4) 01 So Add

```
  111
  1111
+ 0011
 10010
```

0010 0100 1

Shift right

0001 0010

10010 = 16 + 2 = 18

∴ 6 × 3 = 18

```
def convert_to_binary(num):
    arr=[]
    while (num!=1):
        arr.append(round(num%2))
        num=num-num%2
        num/=2
    arr.append(1)
    arr=arr[::-1]
    if(len(arr)<5):
        while(len(arr)!=5):
         arr.insert(0,0)
    return arr



def complement(x):
    num=[]
    one=[]#array  eg 0001
    for i in range(0,len(x)):
        one.append(0)
        if(x[i]==0):
            num.append(1)
        if(x[i]==1):
            num.append(0)
```

```
    one.pop()
    one.append(1)
    num=add(num,one)
    return num


def shift_right(c):
    c.pop()
    c.insert(0,c[0])
    return c
def add(x,y):
    carry=0
    z=[]
    x=x[::-1]
    y=y[::-1]
    for i in range(0,len(x)):
        temp=x[i]+y[i]+carry
        carry=0
        if(temp==2):
            carry=1
            temp=0
        if(temp==3):
            carry=1
            temp=1
        z.append(temp)
```

```
    z=z[::-1]
    return z

a_=convert_to_binary(6) #-6
print(a_)
a=complement(a_)#-6

b_=convert_to_binary(4) #4
b=complement(b_) #-4

c=[0,0,0,0,0]+b+[0]
print(c)
"""print(c)
c=shift_right(c)
print(add(a,b))"""
for i in range(0,5):
    if(c[-1]>c[-2]):
        c=add(c[0:5],a)+c[5:]
    if(c[-1]<c[-2]):
       c=add(c[0:5],a_)+c[5:]
    c=shift_right(c)
    print(c)
c=c[5:-1]
print(c)
```

Conclusion:

Thus we have successfully understood the implementation of Booth's multiplication algorithm. It is an algorithm which is used to perform binary multiplication by addition. We implemented it on paper and using code.

Post Lab Descriptive Questions

1. **Explain advantages and disadvantages of Booth's algorithm.**

One advantage of the Booth multiplier is, it reduce the number of partial product, thus make it extensively used in multiplier with long operands (>16 bits). The main disadvantage of Booth multiplier is the complexity of the circuit to generate a partial product bit in the Booth encoding

2. **Is Booth's recoding better than Booth's algorithm? Justify**

In some cases. While Booth's algorithm reduces the number of additions, the encoding that it performs may be counterproductive for multipliers such as 010101... The modified Booth's algorithm attempts to address this problem.