NAME: ALISTAIR SALDANHA
SAPID: 60009200024
BRANCH: DATA SCIENCE
DIV/BATCH: K / K1

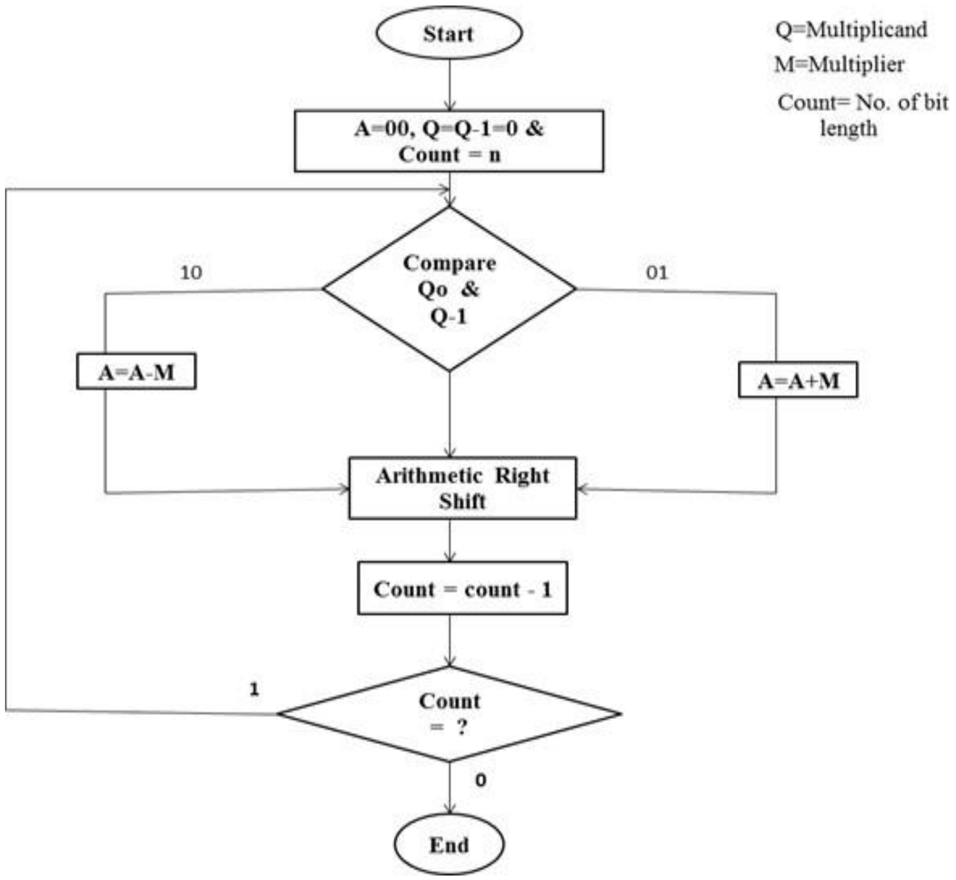
Experiment 2

(Booth's multiplication)

Aim: Implement Booth's multiplication algorithm.

Theory:

- Booth algorithms gives a procedure for multiplying binary integers in signed 2's complement representation in efficient way, i.e., a smaller number of additions/subtractions required. It operates on the fact that strings of 0's in the multiplier require no addition but just shifting and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as 2^(k+1) to 2^m.
- As in all multiplication schemes, booth algorithms require examination of the multiplier bits and shifting of the partial product. Prior to the shifting, the multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to the following rules:
- The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier
- The multiplicand is added to the partial product upon encountering the first 0 (provided that there was a previous '1') in a string of 0's in the multiplier.
- The partial product does not change when the multiplier bit is identical to the previous multiplier bit.
- **Example** – A numerical example of booth's algorithm is shown below for n = 4. It shows the step by step multiplication of 7 and 5.



NAME: ALISTAIR SALDANHA BRANCH: DATA SCIENCE
SAPID: 60009200024 DIV/BATCH: K / K1

Perform 7*5 using Booth's Algorithm

A	Q	Q-1	M		
0000	0101	0	0111	Initial value	
1001 1100	0101 1010	0 1	0111 0111	A ← A−M shift	First cycle
0011 0001	1010 1101	1 0	0111 0111	A ← A+M shift	Second cycle
1010 1101	1101 0110	0 1	0111 0111	A ← A−M shift	Third cycle
0100 0010	0110 0011	1 0	0111 0111	A ← A+M shift	Fourth cycle

Lab Assignments to complete in this session

1. Perform binary multiplication of –7 and –3 using booths algorithm and register size=4 bits

Output:

```
count    A      q     q_minus_1 Operation
  4     0000   0011      0      Initialisation
  4     1001   0011      0      A = A - M
  3     1100   1001      1      Shift Right
  2     1110   0100      1      Shift Right
  2     0101   0100      1      A = A + M
  1     0010   1010      0      Shift Right
  0     0001   0101      0      Shift Right
A : 0001 , q : 0101
Product of -7 and -3 is 21
```

2. Perform binary multiplication of –9 and 7 using booths algorithm and register size=5 bits

Output:

```
count    A      q      q_minus_1 Operation
  5     00000  00111      0       Initialisation
  5     10111  00111      0       A = A - M
  4     11011  10011      1       Shift Right
  3     11101  11001      1       Shift Right
  2     11110  11100      1       Shift Right
  2     00111  11100      1       A = A + M
  1     00011  11110      0       Shift Right
  0     00001  11111      0       Shift Right
A : 00001 , q : 11111
Product of -9 and 7 is -63
```

3. Perform binary multiplication of –13 and –6 using booths algorithm and register size=5 bits

Output:

```
count    A      q     q_minus_1 Operation
  5     00000   00110      0      Initialisation
  4     00000   00011      0      Shift Right
  4     01101   00011      0      A = A - M
  3     00110   10001      1      Shift Right
  2     00011   01000      1      Shift Right
  2     10110   01000      1      A = A + M
  1     11011   00100      0      Shift Right
  0     11101   10010      0      Shift Right
A : 11101 , q : 10010
Product of -13 and 6 is -78
```

4. Perform binary multiplication of –13 and –6 using booths algorithm and register size=4 bits.

Output:

```
count    A      q     q_minus_1 Operation
  5     00000   00110      0      Initialisation
  4     00000   00011      0      Shift Right
  4     01101   00011      0      A = A - M
  3     00110   10001      1      Shift Right
  2     00011   01000      1      Shift Right
  2     10110   01000      1      A = A + M
  1     11011   00100      0      Shift Right
  0     11101   10010      0      Shift Right
A : 11101 , q : 10010
Product of -13 and 6 is -78
```

NAME: ALISTAIR SALDANHA

SAPID: 60009200024

DIV / BATCH: K / K1

EXP-2 BOOTH'S ALGORITHM

Take the input

In [57]:

```
m = 6 # Multiplicand
q = 4  # Multiplier
n = 4  # Number of bits
```

BINARY CONVERSION FUNCTION

In [58]:

```
def to_binary(num):
  if(num >= 0):
    return(bin(num)[2:].zfill(n)) #[2:] - to remove 0b
  elif(num < 0):
    return (bin(abs(num))[2:].zfill(n))
# print(to_binary(9))
# print(to_binary(-2))
```

FUNCTION FOR ADDITION OF TWO BINARY NUMBERS

In [59]:

```
def add(x1, x2, n):
  result = ''
  carry = 0
  for i in range(n - 1, -1, -1):
      carry += 1 if x1[i] == '1' else 0
      carry += 1 if x2[i] == '1' else 0
      result = ('1' if carry % 2 == 1 else '0') + result
      carry = 0 if carry < 2 else 1
  return(result.zfill(n))
# add('0010','1010',4)
```

FUNCTION FOR TWO'S COMPLEMENT

In [60]:

```
def twos_comp(num,n):
  x = ''
  one_add = '0'*(n-1)+'1' # 0001
  one_c = map(lambda x: '0' if x=='1' else '1',num) # 0111 --> 1000
  for i in one_c:
    x += i
  two_c = add(x,one_add,n)
  return two_c
# print(twos_comp('1010',n))
```

FUNCTION FOR ARITHMETIC SHIFT RIGHT (msb is restored)

In [61]:

```
def ashr(bits):
    msb = bits[0] # for restoring in final answer
    shift_bits = bin(int(('0b' + bits),2) >> 1)[2:] # using the shift right operator (ad
vantage in string operations)
    if(msb == '0'): # when converted in int above if the first digit is 0 we lose 0 so to
retain it..
        bits = shift_bits.zfill(2*n+1) # zfill() is used to fill the starting places wit
h 0s
    else:
        bits = msb + shift_bits
    return bits
# print(ashr('001110011'))
# print(ashr('101110011'))
```

DISPLAY THE CALCULATION TABLE

In [62]:

```
def table(bits,count,oper,n): # Display Function
    A = bits[0:n]
    q_bin = bits[n:2*n]
    q_minus_1 = bits[2*n]
    print(f'  {count}     {A}   {q_bin}      {q_minus_1}      {oper}')
```

BOOTH'S ALGORITHM

In [63]:

```
def Booth_Algo(bits,count,n):
    A = bits[:n]
    q_bin = bits[n:2*n]
    q_minus_1 = bits[2*n]
    if(bits[-2] == '1' and bits[-1] == '0'):
        A = add(A,minus_M,n)
        bits = A + q_bin + q_minus_1
        table(bits,count,'A = A - M',n)
        bits = ashr(bits)
        count -= 1
        table(bits,count,'Shift Right',n)
    elif(bits[-2] == '0' and bits[-1] == '1'):
        A = add(A,plus_M,n)
        bits = A + q_bin + q_minus_1
        table(bits,count,'A = A + M',n)
        bits = ashr(bits)
        count -= 1
        table(bits,count,'Shift Right',n)
    elif(bits[-2] == '0' and bits[-1] == '0'):
        bits = ashr(bits)
        count -= 1
        table(bits,count,'Shift Right',n)
    elif(bits[-2] == '1' and bits[-1] == '1'):
        bits = ashr(bits)
        count -= 1
        table(bits,count,'Shift Right',n)
    if(count != 0):
        Booth_Algo(bits,count,n)
    else:
        A = bits[0:n]
        q_bin = bits[n:2*n]
        print(f'A : {A} , q : {q_bin}')
        x = A + q_bin
        if((m>0 and q>0) or (m<0 and q<0)):
            result = int(('0b' + x),2)
            print(f"Product of {m} and {q} is {result}")
        elif(m<0 or q<0):
            result = int(('0b' + twos_comp(x,2*n)),2)
            print(f"Product of {m} and {q} is {-result}")
```

In [64]:

```
# Check for m and q and accordingly get +M,-M,Q
if(m>0 and q>0):
    plus_M = to_binary(m)
    minus_M = twos_comp(plus_M,n)
    q_bin = to_binary(q)
elif(m<0 and q>0):
    plus_M = twos_comp(to_binary(m),n)
    minus_M = to_binary(m)
    q_bin = to_binary(q)
elif(m>0 and q<0):
    plus_M = to_binary(m)
    minus_M = twos_comp(plus_M,n)
    q_bin = twos_comp(to_binary(q),n)
elif(m<0 and q<0):
    plus_M = twos_comp(to_binary(m),n)
    minus_M = to_binary(m)
    q_bin = twos_comp(to_binary(q),n)

q_minus_1 = '0'
A = '0'*n
count=n

# Trace the table
print('count  ',' A  ','   q  ','  q_minus_1',    'Operation')

# Initialisation
bits = A + q_bin + q_minus_1
table(bits,count,'Initialisation',n)
Booth_Algo(bits,count,n)
```

```
count    A      q     q_minus_1 Operation
  4     0000   0100      0      Initialisation
  3     0000   0010      0      Shift Right
  2     0000   0001      0      Shift Right
  2     1010   0001      0      A = A - M
  1     1101   0000      1      Shift Right
  1     0011   0000      1      A = A + M
  0     0001   1000      0      Shift Right
A : 0001 , q : 1000
Product of 6 and 4 is 24
```