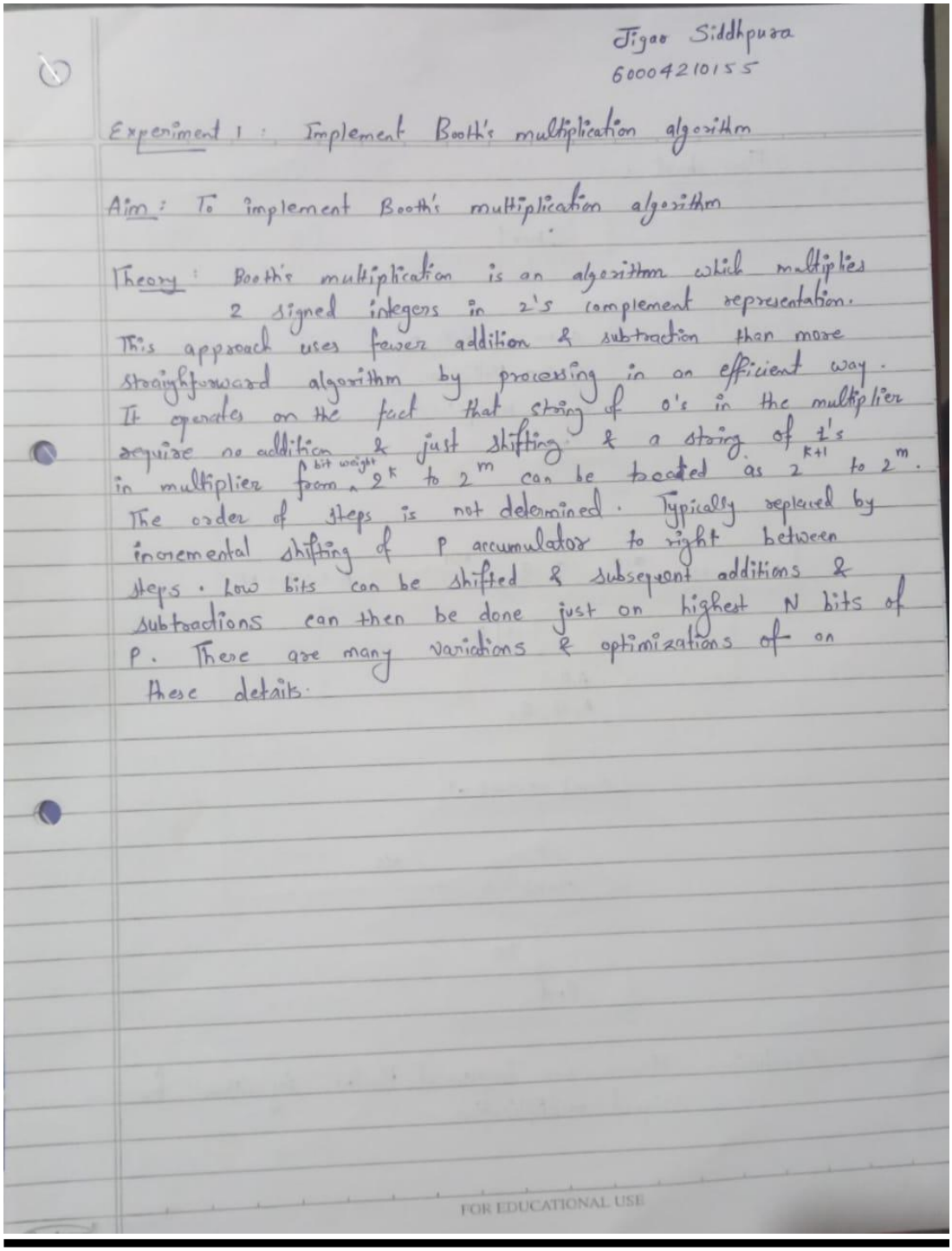
Name: Jigar Siddhpura **SAPID:** 60004200155

DIV: C/C2 **Branch:** Computer Engineering

POA EXPERIMENT 1



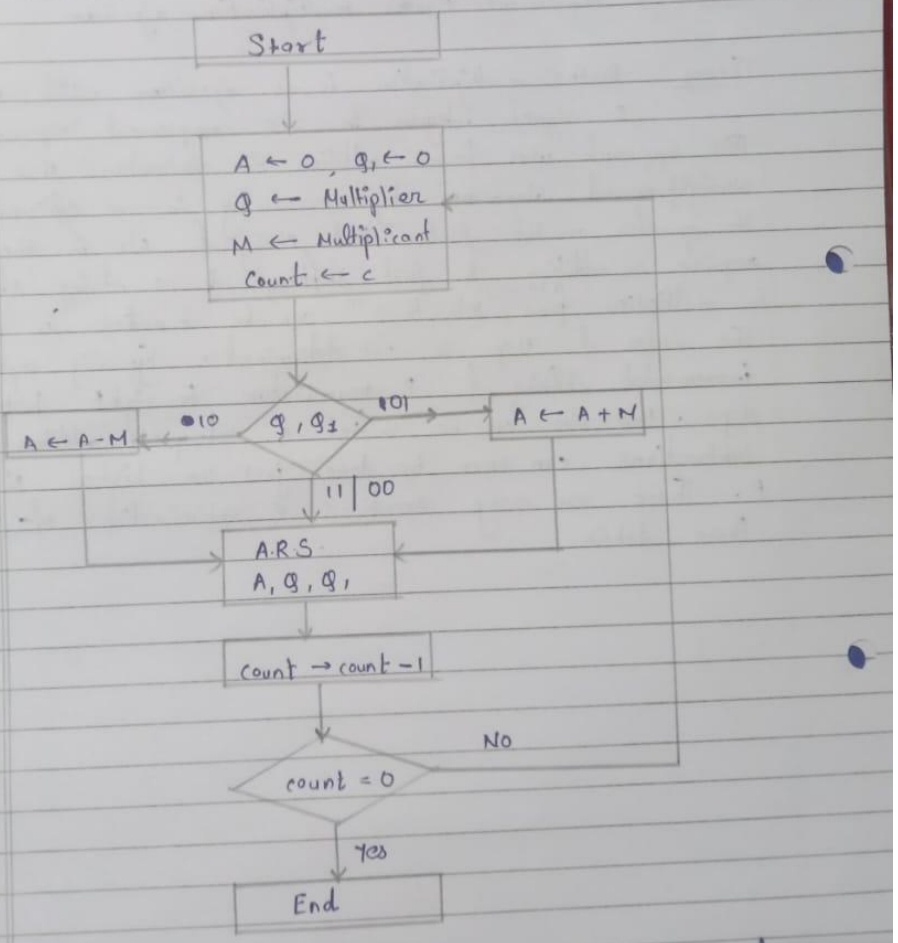
Jigar Siddhpura
60004210155

Experiment 1 : Implement Booth's multiplication algorithm

Aim : To implement Booth's multiplication algorithm

Theory : Booth's multiplication is an algorithm which multiplies 2 signed integers in 2's complement representation. This approach uses fewer addition & subtraction than more straightforward algorithm by processing in an efficient way. It operates on the fact that string of 0's in the multiplier require no addition & just shifting & a string of 1's in multiplier from bit weight 2^k to 2^m can be treated as 2^{k+1} to 2^m. The order of steps is not determined. Typically replaced by incremental shifting of P accumulator to right between steps. Low bits can be shifted & subsequent additions & subtractions can then be done just on highest N bits of P. There are many variations & optimizations of on these details.

Flow chart :



Conclusion : Hence we implement Booths algorithm for signed multiplication

Q - Signed Multiplication of (5 × -11)

Q = $(-11)_{10} = (10101)_2$ → -M = 11011

M = $(5)_{10} = (00101)_2$ → A = 0000

n = 4 → Q_1 = 0

-M = 1's comp (M) + 1
= ~~01010 + 1~~
= ~~+00~~
= 11010 + 1
= 11011

A	Q	Q_1	operation	count
0000	10101	0	(10) A ← A-M: 00000 + 11011 = 11011	5
11011	10101	0 → 1	A.R.S	
11101	11010			
11101	11010	1	(01) A ← A+M: 11101 + 00101 = 1 00010	4
00010	11010	1 → 0	A.R.S	
00001	01101	0		
00001	01101	0	(10) A ← A-M: 00001 + 11011 = 11100	3
11100	01101	0 → 1	A.R.S	
11110	00110	1	(01) A ← A+M: 11110 + 00101 = 1 00011	2
00011	00110	1 → 0	A.R.S.	
00001	10011	0		

00001	10011	0	A ← A − M
		(10)	00001 11011 11100 1
11100	10011	0	A.R.S.
11110	01001		

Now AQ → $(1111001001)_2 = -(55)_{10}$

∵ M.S.B = 1, thus answer is negative

To find the answer, we will take 2's complement

2's complement = 0000110110 + 1

$(0000110111)_2 = (55)_{10}$

∴ 5 × −11 = −55.

Code :

```
def binary_addition(a,b):
    """a,b are binary strings"""
    max_len = max(len(a), len(b))
    a = a.zfill(max_len)
    b = b.zfill(max_len)

    # Initialize the result
    result = ''

    # Initialize the carry
    carry = 0

    # Traverse the string
    for i in range(max_len - 1, -1, -1):
        r = carry
        r += 1 if a[i] == '1' else 0
        r += 1 if b[i] == '1' else 0
        result = ('1' if r % 2 == 1 else '0') + result

        # Compute the carry.
        carry = 0 if r < 2 else 1

    if carry != 0:
        result = '1' + result

    result = result.zfill(max_len)

    return result[-max_len:]

def complement(b):
    # 2's complement of binary number
    b_comp = ""
    for i in b:
        if i=="1": b_comp += "0"
        elif i=="0": b_comp += "1"
    b_comp = binary_addition(b_comp,"1")
    return b_comp

def binary_subtraction(a,b):
    """a,b are binary strings"""
    b_complement = complement(b)
    max_len = len(a)
    return binary_addition(a,b_complement.zfill(max_len))

def ARS(a,q):
    """all are binary strings"""
    q1 = q[-1]
    q = a[-1]+q[:-1]
    a = a[0]+a[:-1]
    return a,q,q1
```

```
def isNegative(bin_number):
    return bin_number[0] == "1"


def booth(multiplier,multiplicant,count,A="0000",Q1="0"):
    binary_multiplier  = bin(multiplier)[2:].zfill(count) if multiplier>0 else
bin(multiplier)[3:].zfill(count)
    binary_multiplicant = bin(multiplicant)[2:].zfill(count) if multiplicant>0
else bin(multiplier)[3:].zfill(count)
    print(f"Multiplier Q = {binary_multiplier}, Multiplicant M =
{binary_multiplicant}, A = {A}, count = {count}, Q1 = 0\n")
    print("Count\tA\tQ\tQ1")
    print("----------------------------------------")
    while(count!=0):
        print(f"{count}\t{A}\t{binary_multiplier}\t{Q1}")
        case = binary_multiplier[-1]+Q1
        print(f"A,Q <- {case}")
        if(case=="01"):
            A = binary_addition(A,binary_multiplicant)
            print(f"{count}\t{A}\t{binary_multiplier}\t{Q1}\tbefore ARS")
        elif(case=="10"):
            A = binary_subtraction(A,binary_multiplicant)
            print(f"{count}\t{A}\t{binary_multiplier}\t{Q1}\tbefore ARS")
        elif(case=="00" or case=="11"):
            pass
        A,binary_multiplier,Q1 = ARS(A,binary_multiplier)
        print(f"{count}\t{A}\t{binary_multiplier}\t{Q1}")
        count -= 1
        print("----------------------------------------")
    bin_ans = A+binary_multiplier
    ans = int(complement(bin_ans),base=2) if isNegative(bin_ans) else
int(bin_ans,base=2)
    print(f"Answer = {ans}")


booth(int(input("Enter Multiplier = ")),int(input("Enter Multiplicant = ")),4)
```

Output :

```
PS D:\SEM 5\POA\EXPERIMENTS> python -u "d:\SEM 5\POA\EXPERIMENTS\booth.py"
Enter Multiplier = 3
Enter Multiplicant = 7
Multiplier Q = 0011, Multiplicant M = 0111, A = 0000, count = 4, Q1 = 0

Count   A       Q       Q1
---------------------------------------
4       0000    0011    0
A,Q <- 10
4       1001    0011    0       before ARS
4       1100    1001    1
---------------------------------------
3       1100    1001    1
A,Q <- 11
3       1110    0100    1
---------------------------------------
2       1110    0100    1
A,Q <- 01
2       0101    0100    1       before ARS
2       0010    1010    0
---------------------------------------
1       0010    1010    0
A,Q <- 00
1       0001    0101    0
---------------------------------------
Answer = 21
PS D:\SEM 5\POA\EXPERIMENTS>
```