EXPERIMENT 1
Booth's Algorithm

AIM: To implement Booth's Multiplication algorithm.

CODE:

```
# to ensure both numbers have the same number of bits
def conversion(a, count):
    q = ""
    current_n = len(a)
    temp = count - current_n
    if current_n != count:
        q = "0" * temp + a  # add 0 to its start
    return q


def add(x, y):
    max_len = max(len(x), len(y))
    result = ''
    carry = 0
    for i in range(max_len - 1, -1, -1):
        r = carry
        if x[i] == '1':
            r += 1
        if y[i] == '1':
            r += 1
        if r % 2 == 1:
            result = "1" + result  # concatenate 1 to the final answer string
        else:
            result = "0" + result  # concatenate 0 to the final answer string
        if r < 2:
```

```
            carry = 0  # generated for the next bit's addition
        else:
            carry = 1
    return result



def twoc(a):
    l = list(a)
    for i in range(len(l)):
        if l[i] == "1":
            l[i] = "0"
        else:
            l[i] = "1"
    b = "0" * (len(l) - 1) + "1"
    return add("".join(l), b)

def right_shift(ac, q, q1):
    a = ac[0]
    for i in range(1, len(ac)):
        a += ac[i - 1]
    b = ac[-1]
    for j in range(1, len(q)):
        b += q[j - 1]
    c = q[-1]  # Q gets A's last bit
    return a, b, c

a = "0100"
b = "-0010"

negative_a = 0
negative_b = 0
```

```
# if number is negative, in binary we add 1 to the start
if a[0] == "-":
    a = a.replace("-", "")
    negative_a = 1
if b[0] == "-":
    b = b.replace("-", "")
    negative_b = 1

if len(a) > len(b):
    count = len(a) + 1
else:
    count = len(b) + 1
count1 = count

firstP = conversion(a, count)
secondP = conversion(b, count)

# 2's complement
firstN = twoc(firstP)
secondN = twoc(secondP)

if negative_a == 0:
    M = firstP
    M2 = firstN
else:
    M = firstN
    M2 = firstP

if negative_b == 0:
    Q = secondP
```

```
else:
    Q = secondN

AC = conversion("0", count)
Q1 = "0"

print("Count" + " " * count1 + "AC" + " " * count1 + "Q" + " " * count1 + "Q1" + " " * count1 +
"Operation")
print(str(count) + " " * count1 + AC + " " * count1 + Q + " " * count1 + Q1 + " " * count1 + "initial")

while count > 0:
    compare = Q[-1] + Q1
    if compare[0] == compare[-1]:  # checking 00 or 11
        # only right shift is performed
        AC, Q, Q1 = right_shift(AC, Q, Q1)
        Op = "right shift"

    # if 10 then A = A+(-M) and right shift
    elif compare == "10":
        AC = add(AC, M2)
        AC, Q, Q1 = right_shift(AC, Q, Q1)
        Op = "AC=AC-M and right shift"

    # if 01 then A = A+M and right shift
    elif compare == "01":
        AC = add(AC, M)
        AC, Q, Q1 = right_shift(AC, Q, Q1)
        Op = "AC=AC+M and right shift"

    print(str(count) + " " * count1 + AC + " " * count1 + Q + " " * count1 + Q1 + " " * count1 + Op)
```

```
    count = count - 1  # decrementing count
answer = AC + Q

# if both numbers are negative/positive then the final answer is positive
if negative_a == negative_b:
    ans_d = str(int(answer, 2))

# if either of the numbers is negative then the final answer is negative
else:
    ans_d = "-" + str(int(twoc(answer), 2))



print("Product in binary is:" + answer)

print("a= 4")
print("b= -2")
print("In Decimal:" + ans_d)
print("Vishwa Jarsaniya C185")
```

OUTPUT:

```
Count     AC     Q     Q1     Operation
5     00000     11110     0     initial
5     00000     01111     0     right shift
4     11110     00111     1     AC=AC-M and right shift
3     11111     00011     1     right shift
2     11111     10001     1     right shift
1     11111     11000     1     right shift
Product in binary is:1111111000
a= 4
b= -2
In Decimal:-8
Vishwa Jarsaniya C185
```