Batch: D3	**Roll No.: 16010123294**
Experiment / assignment / tutorial No. 02	

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)

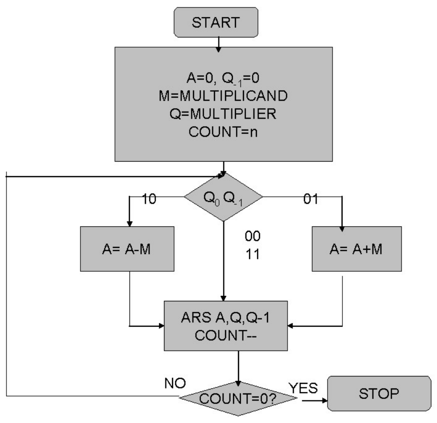
Books/ Journals/ Websites referred:

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:



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 binary
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)

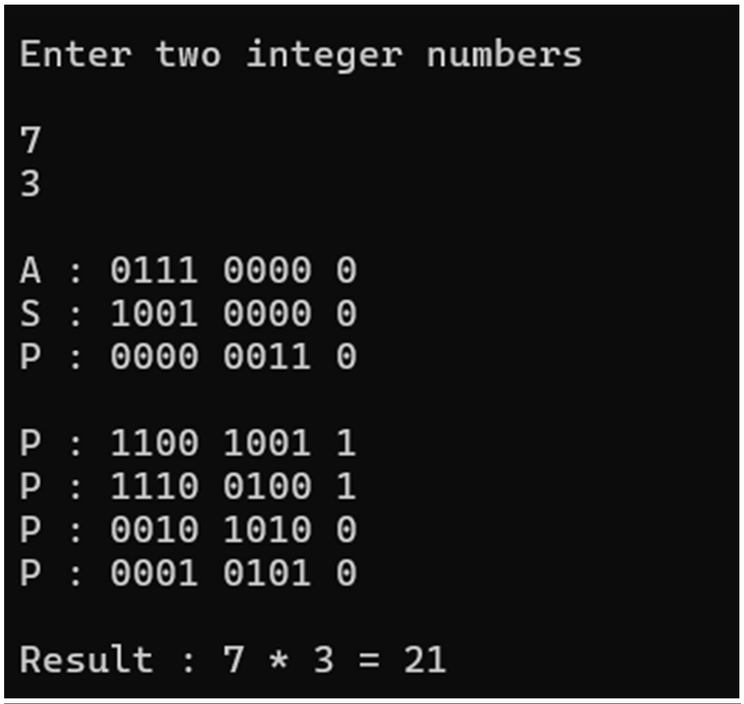
Implementation

```
import java.util.Scanner;

 public class Booth
 {
    public static Scanner s = new Scanner(System.in);
    public int multiply(int n1, int n2)
    {
       int[] m = binary(n1);
       int[] m1 = binary(-n1);
       int[] r = binary(n2);
```

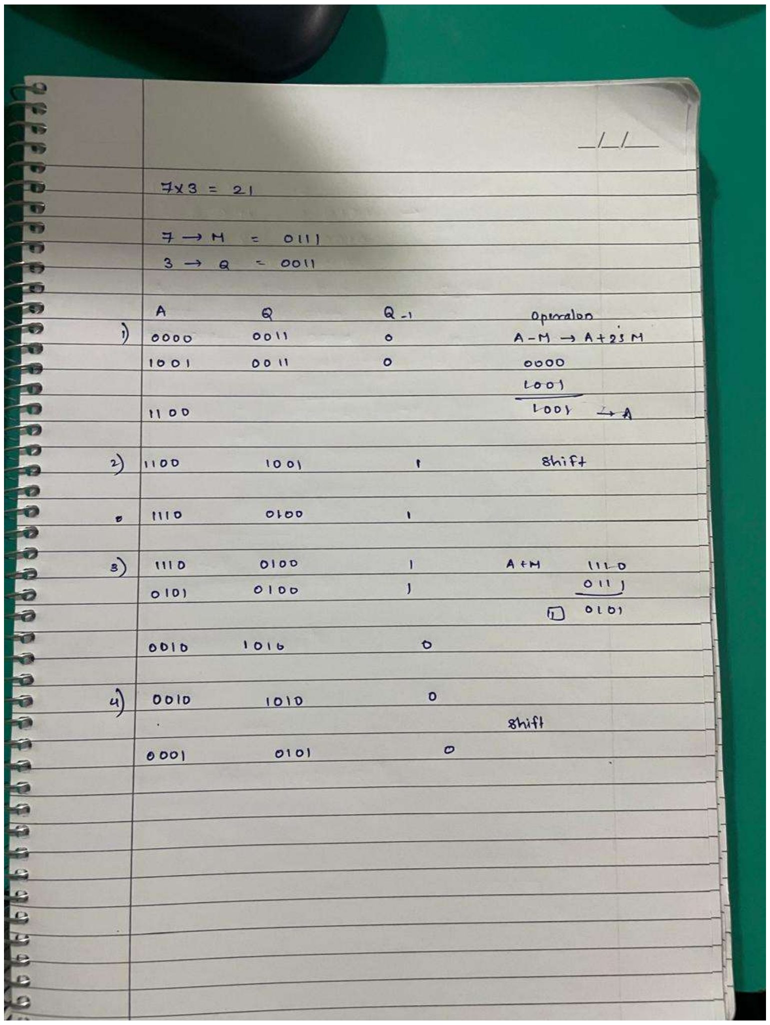
```
        int[] A = new int[9];
        int[] S = new int[9];
        int[] P = new int[9];
        for (int i = 0; i < 4; i++)
        {
            A[i] = m[i];
            S[i] = m1[i];
            P[i + 4] = r[i];
        }
        display(A, 'A');
        display(S, 'S');
        display(P, 'P');
        System.out.println();

        for (int i = 0; i < 4; i++)
        {
            if (P[7] == 0 && P[8] == 0);

            else if (P[7] == 1 && P[8] == 0)
                add(P, S);
            else if (P[7] == 0 && P[8] == 1)
                add(P, A);
            else if (P[7] == 1 && P[8] == 1);



            rightShift(P);
            display(P, 'P');
        }
        return getDecimal(P);
    }
```

```
public int getDecimal(int[] B)
{
    int p = 0;
    int t = 1;
    for (int i = 7; i >= 0; i--, t *= 2)
        p += (B[i] * t);
    if (p > 64)
        p = -(256 - p);
    return p;
}

public void rightShift(int[] A)
{
    for (int i = 8; i >= 1; i--)
        A[i] = A[i - 1];
}

public void add(int[] A, int[] B)
{
    int carry = 0;
    for (int i = 8; i >= 0; i--)
    {
        int temp = A[i] + B[i] + carry;
        A[i] = temp % 2;
        carry = temp / 2;
    }
}

public int[] binary(int n)
```

```
{
    int[] bin = new int[4];
    int ctr = 3;
    int num = n;

    if (n < 0)
        num = 16 + n;
    while (num != 0)
    {
        bin[ctr--] = num % 2;
        num /= 2;
    }
    return bin;
}
/** Function to print array **/
public void display(int[] P, char ch)
{
    System.out.print("\n"+ ch +" : ");
    for (int i = 0; i < P.length; i++)
    {
        if (i == 4)
            System.out.print(" ");
        if (i == 8)
            System.out.print(" ");
        System.out.print(P[i]);
    }
}

public static void main (String[] args)
{
        Scanner scan = new Scanner(System.in);
        System.out.println("Booth Algorithm Test\n");
        Booth b = new Booth();

        System.out.println("Enter two integer numbers\n");
        int n1 = scan.nextInt();
        int n2 = scan.nextInt();
        int result = b.multiply(n1, n2);
        System.out.println("\n\nResult : "+ n1 +" * "+ n2 +" = "+ result);
    }
}
```



```
Enter two integer numbers

7
3

A : 0111 0000 0
S : 1001 0000 0
P : 0000 0011 0

P : 1100 1001 1
P : 1110 0100 1
P : 0010 1010 0
P : 0001 0101 0

Result : 7 * 3 = 21
```



__/__/__

$7 \times 3 = 21$

$7 \rightarrow M = 0111$

$3 \rightarrow Q = 0011$

	A	Q	Q_{-1}	Operation
1)	0000	0011	0	$A - M \rightarrow A + 2\text{'s } M$
	1001	0011	0	0000
				$\underline{1001}$
	1100			$1001 \rightarrow A$
2)	1100	1001	1	shift
	1110	0100	1	
3)	1110	0100	1	A+M 1110
	0101	0100	1	$\underline{0111}$
				$\boxed{1}$ 0101
	0010	1010	0	
4)	0010	1010	0	
				shift
	0001	0101	0	

Conclusion:
learned about Booths Algorithm and its application with coding challenge.

<u>**Post Lab Descriptive Questions**</u>

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

Advantages:

Efficiency with Two's Complement: Booth's algorithm handles both positive and negative numbers in two's complement form, which is advantageous in many digital systems.

Reduces the Number of Operations: It minimizes the number of addition/subtraction operations by skipping over sequences of 1s, leading to potentially faster multiplication.

Handles Multiple Bits at Once: The algorithm can process multiple bits of the multiplier in one step, which can speed up the computation.

Disadvantages:

Complex Implementation: The algorithm is more complex to implement compared to simpler methods like the standard binary multiplication, requiring more control logic.

Variable Time: The execution time can vary depending on the bit pattern of the multiplier, making it less predictable.

Overhead for Special Cases: The algorithm can have overhead when dealing with certain patterns in the multiplicand or multiplier, which may not result in significant time savings.

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

Advantages of Booth's Recoding Over Booth's Algorithm:

Reduced Number of Operations: Booth's recoding optimizes the multiplication process by reducing the number of addition/subtraction operations required. It simplifies sequences of bits, leading to fewer operations.

Consistent Performance: Unlike Booth's original algorithm, where execution time can vary based on the bit pattern of the multiplier, Booth's recoding provides more consistent performance by standardizing the recoding process.

Simplified Hardware Implementation: Booth's recoding often results in simpler and more efficient hardware design because it deals with fewer and more predictable operations, making it easier to implement in circuits.

Justification:

Booth's recoding is often preferred in high-performance systems where consistent and optimized performance is crucial. It takes the original concept of Booth's algorithm and refines it to reduce complexity and enhance speed, especially in modern processors and digital systems.

Date: _____________