Batch: E-2	**Roll No.: 16010123325**
Experiment / assignment / tutorial No.__2_	

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)

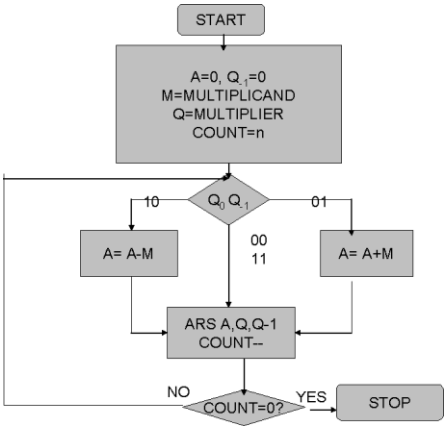
Books/ Journals/ Websites referred:

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:



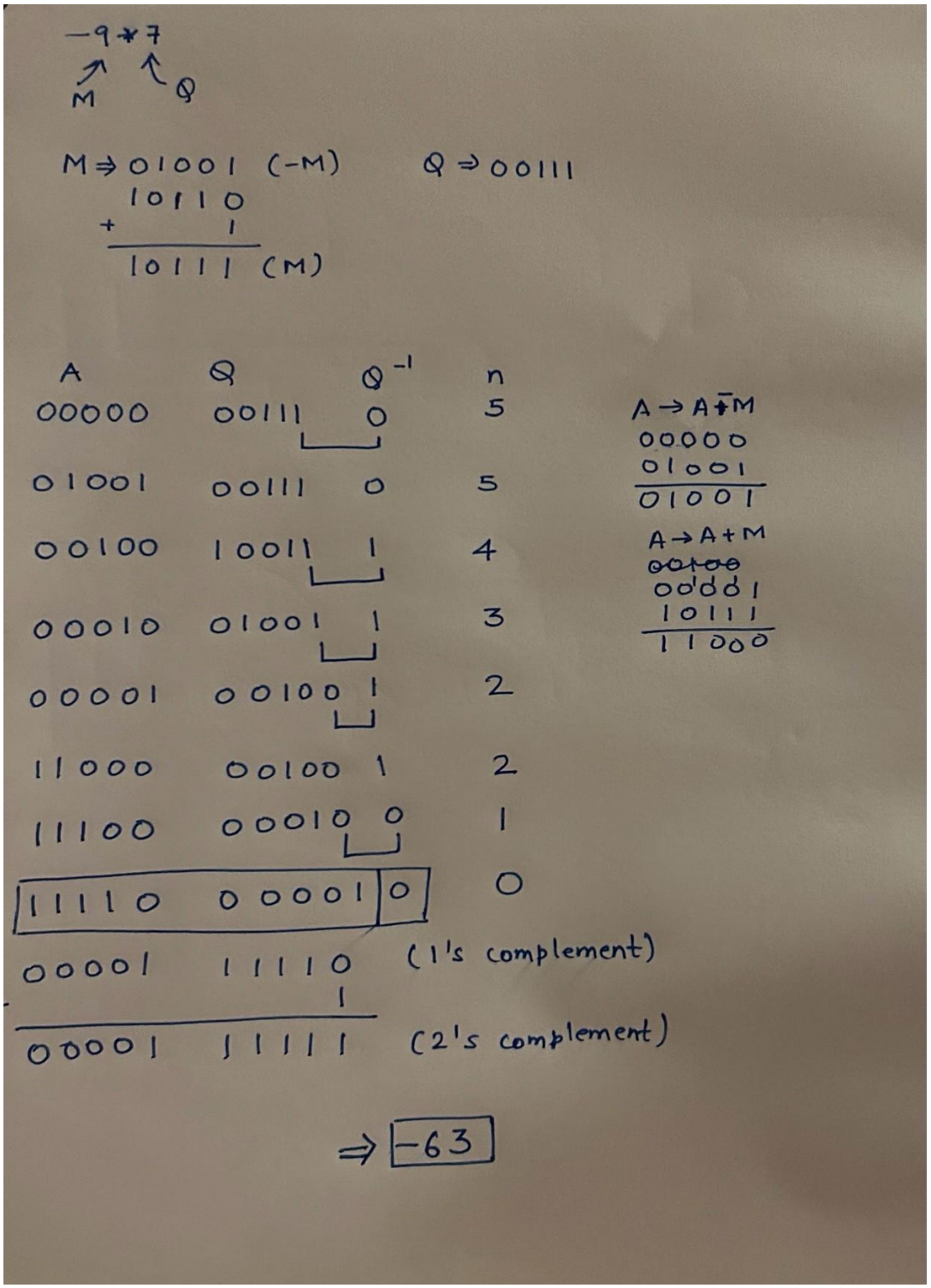
Design Steps:

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)



Code:

```cpp
#include <bits/stdc++.h>
using namespace std;

// Function to print binary representation
void printBinary(int n, int size) {
    for (int i = size - 1; i >= 0; i--) {
        cout << ((n >> i) & 1);
    }
    cout << '\n';
}

// Function to perform arithmetic right shift
void arithmeticRightShift(int& A, int& Q, int& Q_1, int size) {
    Q_1 = Q & 1;
    Q = (Q >> 1) | ((A & 1) << (size - 1));
    A = (A >> 1);
}

// Function to add two binary numbers
int binaryAdd(int a, int b) {
    return a + b;
}

// Function to perform Booth's Algorithm
void boothAlgorithm(int M, int Q, int size) {
    int A = 0;
    int Q_1 = 0;
    int count = size;

    cout << "Initial values:" << '\n';
    cout << "A: ";
    printBinary(A, size);
    cout << "Q: ";
    printBinary(Q, size);
    cout << "Q-1: " << Q_1 << '\n';
    cout << "M: ";
    printBinary(M, size);
    cout << '\n';

    while (count > 0) {
        int Q0 = Q & 1;

        // Apply Booth's Algorithm
        if (Q0 == 1 && Q_1 == 0) {
```

```
            A = binaryAdd(A, -M);
            cout << "A = A - M" << '\n';
        } else if (Q0 == 0 && Q_1 == 1) {
            A = binaryAdd(A, M);
            cout << "A = A + M" << '\n';
        }

        // Arithmetic right shift
        arithmeticRightShift(A, Q, Q_1, size);

        cout << "After shift:  A: ";
        printBinary(A, size);
        cout << "Q: ";
        printBinary(Q, size);
        cout << "Q-1: " << Q_1 << '\n';
        cout << '\n';

        count--;
    }

    int result = (A << size) | Q;
    cout << "Final result: ";
    printBinary(result, size * 2);
    cout << "Decimal result: " << result << '\n';
}

int main() {
    int M, Q, size;

    cout << "Enter the bit size: ";
    cin >> size;

    cout << "Enter multiplicand (M) in decimal: ";
    cin >> M;

    cout << "Enter multiplier (Q) in decimal: ";
    cin >> Q;

    boothAlgorithm(M, Q, size);

    return 0;
}
```

Output:

```
PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs> cd "c:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs\" ; if ($?) { g++ booth-m
ultiplication.cpp -o booth-multiplication } ; if ($?) { .\booth-multiplication }
Enter the bit size: 5
Enter multiplicand (M) in decimal: -9
Enter multiplier (Q) in decimal: 7
Initial values:
A: 00000
Q: 00111
Q-1: 0
M: 10111

A = A - M
After shift:  A: 00100
Q: 10011
Q-1: 1

After shift:  A: 00010
Q: 01001
Q-1: 1

After shift:  A: 00001
Q: 00100
Q-1: 1

A = A + M
After shift:  A: 11100
Q: 00010
Q-1: 0

After shift:  A: 11110
Q: 00001
Q-1: 0

Final result: 1111000001
Decimal result: -63
PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs>
```

```
PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA> cd "c:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs\" ; if ($?) { g++ booth-mul
tiplication.cpp -o booth-multiplication } ; if ($?) { .\booth-multiplication }
Enter the bit size: 4
Enter multiplicand (M) in decimal: 5
Enter multiplier (Q) in decimal: 4
Initial values:
A: 0000
Q: 0100
Q-1: 0
M: 0101

After shift:  A: 0000
Q: 0010
Q-1: 0

After shift:  A: 0000
Q: 0001
Q-1: 0

A = A - M
After shift:  A: 1101
Q: 1000
Q-1: 1

A = A + M
After shift:  A: 0001
Q: 0100
Q-1: 0

Final result: 00010100
Decimal result: 20
PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs>
```

Conclusion:

The above program highlights the implementation of Booth's Algorithm through code in Java.

Post Lab Descriptive Questions

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

Advantages

- **Efficient with Negative Numbers**: Handles both positive and negative binary numbers using two's complement.
- **Reduces Operations**: Skips unnecessary additions for consecutive 1s or 0s in the multiplier.
- **Simple Hardware Implementation**: Uses basic components like adders and shift registers.

Disadvantages

- **Variable Execution Time**: Performance depends on the bit pattern of the multiplier, leading to inconsistent execution times.
- **Complexity in Some Cases**: Can become inefficient with frequent bit changes in the multiplier, leading to more operations.
- **Difficult for Non-Binary Systems**: Primarily designed for binary multiplication, making it less adaptable to other numeral systems.

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

Booth's recoding is a technique used to optimize Booth's algorithm by reducing the number of additions needed during multiplication. It simplifies the process, especially with consecutive ones in the multiplier. Booth's algorithm incorporates this recoding to perform efficient binary multiplication, particularly for signed numbers. Thus, Booth's recoding improves Booth's algorithm by enhancing its efficiency and is better.

Date: