# **FR. Conceicao Rodrigues College of Engineering Department of Computer Engineering**

**7. Write a program for Binary Multiplication.(Booth’s Algorithm)**

# **Course, Subject & Experiment Details**

| **Academic Year** | **2023-24** | **Estimated Time** | **Experiment No. 7.– 02 Hours** |
| --- | --- | --- | --- |
| **Course & Semester** | **S.E. (Computers) – Sem. III** | **Subject Name** | **Digital Logic & Computer Organization and Architecture** |
| **Chapter No.** | **2** | **Chapter Title** | **Data Representation and Arithmetic algorithms** |
| **Experiment Type** | **Software** | **Subject Code** | **CSC304** |

**Rubrics**

| Roll No | Date of Performance  4/09/23 | Timeline  (2) | Practical Skill & Applied Knowledge  **(4)** | Output  (4) | Total  (10) |
| --- | --- | --- | --- | --- | --- |
| 9914 | Date of Submission  9/09/2023 |  |  |  |  |

# **Aim & Objective of Experiment**

* + Learn to implement multiplication by using addition and shifts (Booth's algorithm).
  + To study and implement n bit Binary Multiplication using C/Java/ Python

# **Problem Statement**

Write a C/ Java/ Python program to implement Booth’s Algorithm for Multiplication..

.

# **Brief Theoretical Description**

With unsigned multiplication there is no need to take the sign of the number into consideration. However in signed multiplication the same process cannot be applied because the signed number is in a 2’s compliment form which would yield an incorrect result if multiplied in a similar fashion to unsigned multiplication. That’s where Booth’s algorithm comes in. Booth’s algorithm preserves the sign of the result.

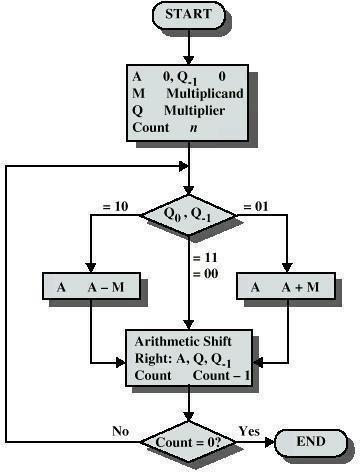
## ***Algorithm:***

1. Multiplier and multiplicand are placed in the Q and M registers.
2. One bit register(Q-1) placed logically to the right of least significant bit of the register Q0.
3. The result of multiplication will appear in A and Q registers.
4. A and Q-1 are initialized to zero.
5. If the combination of two bits is same(0-0 or 1-1), then all the bits of A

, Q and Q-1 registers are shifted to right by 1 bit.

1. If two bits differ , then the multiplicand is added to or subtracted from register A depending upon whether two bits are 0-1 or 1-0.
2. Following the addition or subtraction, the right shift occurs.
3. In either case, the right shift is such that the leftmost bit of A, namely A n-1, not only is shifted into A n-2, but also remains in A n-1. This is called **Arithmetic Shift,** because it preserves the sign bit.

## ***Flowchart of Booth’s Algorithm:***



* 1. ***Attach the program***

#include <stdio.h>

int boothsMultiplication(int *M*, int *Q*)

{

int result = 0; *// Final result*

int A = 0; *// Accumulator*

int QBit, prevQbit = 0; *// QBit: Current bit in Q, prevQbit: Previous bit in Q*

int n = sizeof(int) \* 8; *// Number of steps (bits in an integer)*

*// Loop n times ( i.e. for each bit)*

for (int i = 0; i < n; i++)

{

QBit = *Q* & 1; *// Get the least significant bit of Q*

*// Check if QBit and prevQbit indicate an addition or subtraction*

if (QBit == 1 && prevQbit == 0)

{

A -= *M*; *// Subtract M from the accumulator A*

}

else if (QBit == 0 && prevQbit == 1)

{

A += *M*; *// Add M to the accumulator A*

}

*// Check if the least significant bit of A is 1*

if ((A & 1) == 1)

{

result |= (1 << i); *// Set the corresponding bit in the result*

}

A >>= 1; *// Right shift the accumulator A*

prevQbit = QBit; *// Update prevQbit for the next iteration*

*Q* >>= 1; *// Right shift Q for the next iteration*

}

return result;

}

*// Start of the main function*

int main()

{

int Q, M;

printf("Enter multiplicand:");

scanf("%d", &M);

printf("Enter multiplier:");

scanf("%d", &Q);

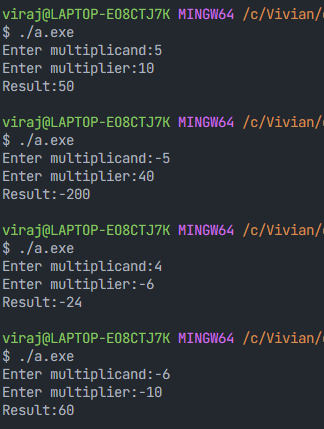
int result = boothsMultiplication(M, Q);

printf("Result:%d\n", result);

return 0;

}

**OUTPUT:**

****

# **Conclusion:**

In conclusion, Booth's Algorithm is a powerful technique for binary multiplication, offering advantages such as reduced operations, faster computation, and efficient hardware implementation. It excels in handling signed numbers, has parallel processing potential, and contributes to reduced power consumption in modern processors.

* 1. **Post-lab:**
     1. Write advantages of Booth’s algorithm.

⇒Some of the advantages of the booth’s algorithm are:

1. Fewer Operations: Booth's algorithm reduces arithmetic operations, saving precious time.

2.Faster Multiplication. It multiplies numbers faster, ideal for hardware and processors.

3. Hardware Efficiency: Perfect for efficient hardware circuits like binary multipliers.

4. Handling Signed Numbers: Handles positive and negative numbers with ease.

5. Parallel Processing Potential: Can be tweaked for parallel processing, cranking up the speed.

6. Reduced Power Consumption: Cuts power usage in modern processors, great for batteries and data centers.