Experiment No. : 09

Aim: To implement Booth's algorithm.

Theory:

BOOTHS Algorithm in C -

A *multiplication algorithm* called ***Booth's algorithm*** is used to multiply two signed ***binary values***.
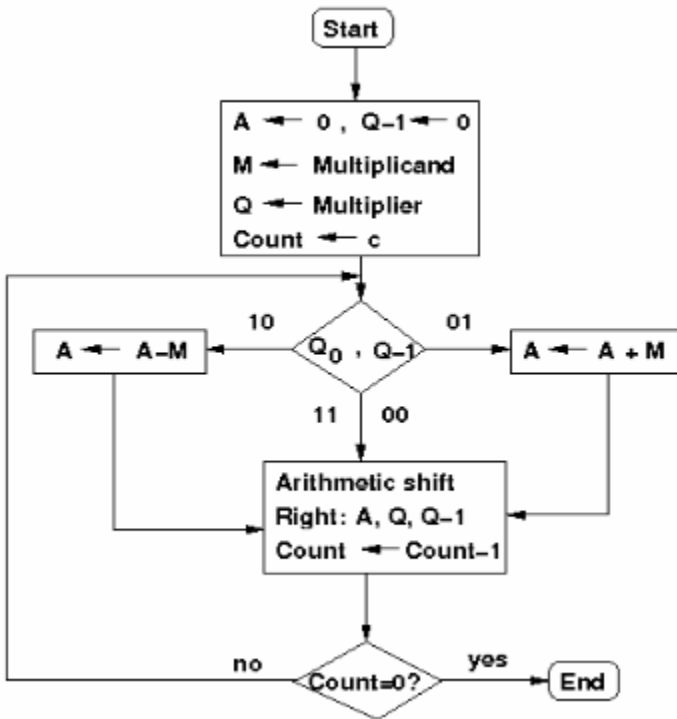
The technique increases processing efficiency by reducing the amount of addition operations needed for multiplication. It accomplishes this by performing a number of ***shifts*** and ***adds***, which are easily accomplished by straightforward hardware circuits.

Booth's Multipliers:

Booth's multiplication algorithm is an algorithm which multiplies 2 signed integers in 2's complement. The algorithm is depicted in the following figure with a brief description. This approach uses fewer additions and subtractions than more straightforward algorithms

- The multiplicand and multiplier are placed in the m and Q registers respectively.
- A 1 bit register is placed logically to the right of the LSB (least significant bit) Q0 of Q register. This is denoted by Q-1. A and Q-1 are initially set to 0.
- Control logic checks the two bits Q0 and Q-1. If the two bits are same (00 or 11) then all of the bits of A, Q, Q-1 are shifted 1 bit to the right.
- If they are not the same and if the combination is 10 then the multiplicand is subtracted from A and if the combination is 01 then the multiplicand is added with A.
- In both the cases results are stored in A, and after the addition or subtraction operation, A, Q, Q-1 are right shifted. The shifting is the arithmetic right shift operation where the left most bit namely, An-1 is not only shifted into An-2 but also remains in An-1.
- This is to preserve the sign of the number in A and Q. The result of the multiplication will appear in the A and Q

Flowchart:

Booth's Algorithm 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. 6. Repeat steps 5 till all bits are compared
7. 7. Convert the result to decimal form and display
8. 8. End

Constraints of this program:

• The number entered must be in between the range -16 < Number ≤ 16

• For number greater than 16 this algorithm fails.

• Range is calculated by the formula 2^k. So for 5 bit number it will be 32 bits, then by splitting the range we can say -16 to +16 and when 0 is considered we say it as -15 to +16

• Same steps are followed for 4 bit numbers

• -15 to 16 for 5 bits

• -7 to 8 for 4 bits

Output of few test cases:

Consider input of 12 and -2 the product is -24 so the binary format of the number would be 1111101000 which is the required result.

Conclusion: Booth's Multiplication algorithm has been successfully implemented in C.