Lab 7: Write a C Program to Implement Booth's Algorithm for Multiplication

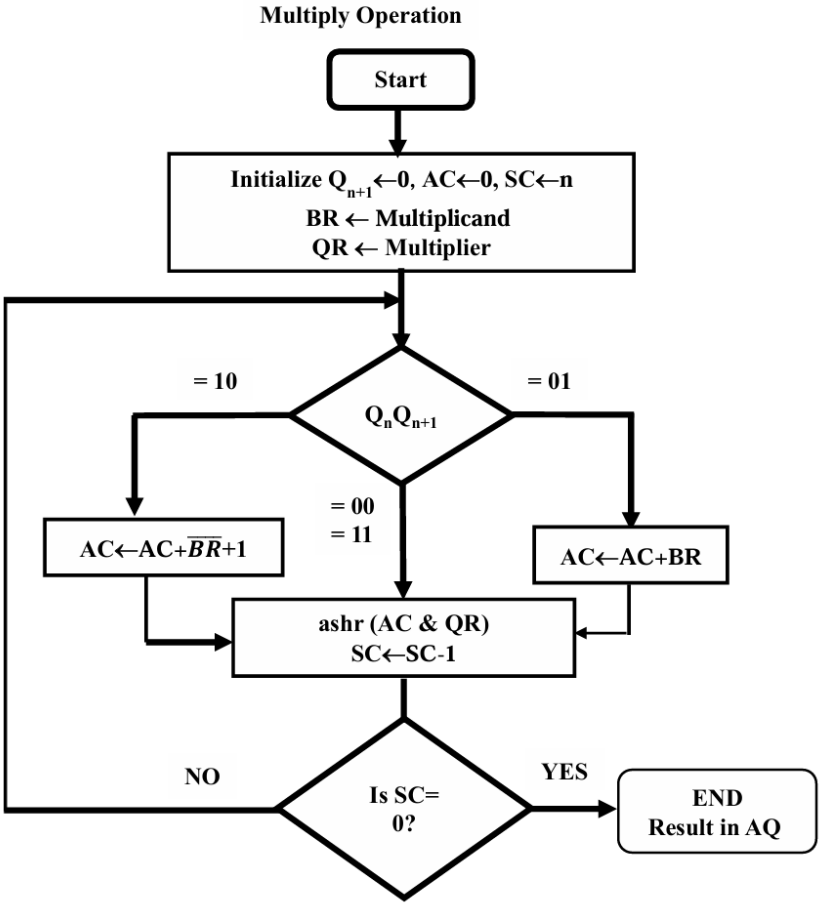
Objective:

Write a C Program to Implement Booth's Algorithm for Multiplication.

Theory:

- Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed Binary numbers in notation.
- Booth's algorithm serves two purposes:
  a) Fast multiplication (when there is consecutive 0's or 1's in the multiplier).
  b) And Signed multiplication.
- Booth's algorithm is a powerful direct algorithm to perform signed-number multiplication.
- The algorithm is based on the fact that any binary number can be represented by the sum and difference of other binary numbers.
- Booth's algorithm examines adjacent pairs of bits of the N-bit multiplier Q in signed two's complement representation, including an implicit bit below the least significant bit, Qn+1.

Flowchart:



Algorithm:

1. Start
2. Initialize, ***BR←Multiplicand and QR←Multiplier***.
3. ***AC←0, Q_{n+1}←0 and SC←n*** (no of bits in multiplier).
4. Inspect ***Q_n, Qn+1***,
5. If ***(Q_nQ_{n+1}=10)***, first 1 in a string of 1' s has been encountered, subtraction required i.e., ***AC←AC+$\overline{BR}$+1***
6. If ***(Q_nQ_{n+1}=01),*** first 0 in a string of 0' s has been encountered, addition required i.e., ***AC←AC+BR***
7. If (***Q_nQ_{n+1}=00 or 11(equal)),*** do nothing
8. Shift right the partial product and the multiplier (including bit Q_{n+1}).
9. **SC←SC-1**. If (SC=0) stop else continue and go to step 3.
10. Result in AC and QR
11. End

C++ Code:

```
#include <iostream>
#include <conio.h>

using namespace std;

void add(int a[], int x[], int qrn);
void complement(int a[], int n)
{
    int i;

    int x[8] = {NULL};
    x[0] = 1;
    for (i = 0; i < n; i++)
    {
        a[i] = (a[i] + 1) % 2;
    }
    add(a, x, n);
}
```

```
void add(int ac[], int x[], int qrn)
{
    int i, c = 0;
    for (i = 0; i < qrn; i++)
    {
        ac[i] = ac[i] + x[i] + c;
        if (ac[i] > 1)
        {
            ac[i] = ac[i] % 2;
            c = 1;
        }
        else
            c = 0;
    }
}

void ashr(int ac[], int qr[], int &qn, int qrn)
{
    int temp, i;

    temp = ac[0];
    qn = qr[0];
    cout << "\t\tashr\t\t";
    for (i = 0; i < qrn - 1; i++)
    {
        ac[i] = ac[i + 1];
        qr[i] = qr[i + 1];
    }
    qr[qrn - 1] = temp;
}

void display(int ac[], int qr[], int qrn)
{
    int i;

    for (i = qrn - 1; i >= 0; i--)
        cout << ac[i];
    cout << " ";
    for (i = qrn - 1; i >= 0; i--)
        cout << qr[i];
}
```

```
int main(int argc, char **argv)
{
    int mt[10], br[10], qr[10], sc, ac[10] = {0};
    int brn, qrn, i, qn, temp;
    cout
        << "\n--Enter the multiplicand and multipier in signed 2's complement
form if negative--";

    cout << "\n Number of multiplicand bit=";
    cin >> brn;
    cout << "\nmultiplicand=";

    for (i = brn - 1; i >= 0; i--)
        cin >> br[i]; // multiplicand

    for (i = brn - 1; i >= 0; i--)
        mt[i] = br[i]; // copy multipier to temp array mt[]

    complement(mt, brn);

    cout << "\nNo. of multiplier bit=";
    cin >> qrn;

    sc = qrn; // sequence counter

    cout << "Multiplier=";
    for (i = qrn - 1; i >= 0; i--)
        cin >> qr[i]; // multiplier

    qn = 0;
    temp = 0;

    cout << "qn\tq[n+1]\t\tBR\t\tAC\tQR\t\tsc\n";
    cout << "\t\t\tinitial\t\t";
    display(ac, qr, qrn);
    cout << "\t\t" << sc << "\n";

    while (sc != 0)
    {
        cout << qr[0] << "\t" << qn;
        if ((qn + qr[0]) == 1)
        {
            if (temp == 0)
            {
                add(ac, mt, qrn);
```

```
                cout << "\t\tsubtracting BR\t";
                for (i = qrn - 1; i >= 0; i--)
                    cout << ac[i];
                temp = 1;
            }
            else if (temp == 1)
            {
                add(ac, br, qrn);
                cout << "\t\tadding BR\t";
                for (i = qrn - 1; i >= 0; i--)
                    cout << ac[i];
                temp = 0;
            }
            cout << "\n\t";
            ashr(ac, qr, qn, qrn);
        }
        else if (qn - qr[0] == 0)
            ashr(ac, qr, qn, qrn);

        display(ac, qr, qrn);
        cout << "\t";

        sc--;
        cout << "\t" << sc << "\n";
    }
    cout << "Result=";
    display(ac, qr, qrn);
}
```

Output:

```
--Enter the multiplicand and multipier in signed 2's complement form if negative--
 Number of multiplicand bit=5

multiplicand=1 0 1 1 1

No. of multiplier bit=5
Multiplier=1 0 0 1 1
qn      q[n+1]          BR              AC      QR              sc
                        initial         00000 10011             5
1       0               subtracting BR  01001
                        ashr            00100 11001             4
1       1               ashr            00010 01100             3
0       1               adding BR       11001
                        ashr            11100 10110             2
0       0               ashr            11110 01011             1
1       0               subtracting BR  00111
                        ashr            00011 10101             0
Result=00011 10101
 ■  ...  CA  main ?2 ~2  21:48  28.44s
```