Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

A.Y. 2022-2023

PROCESSOR ORGANIZATION AND ARCHITECTURE

AYUSH JAIN

COMPUTER ENGINEERING | TE – B2 | 60004200132

EXPERIMENT – 1

AIM: To implement Booth's multiplication algorithm.

THEORY:

The booth algorithm is a multiplication algorithm that allows us to multiply the two signed binary integers in 2's complement, respectively. It is also used to speed up the performance of the multiplication process. It is very efficient too. It works on the string bits 0's in the multiplier that requires no additional bit only shift the right-most string bits and a string of 1's in a multiplier bit weight 2^k to weight 2^m that can be considered as $\mathbf{2^{k+1} - 2^m}$.

FLOW CHART:

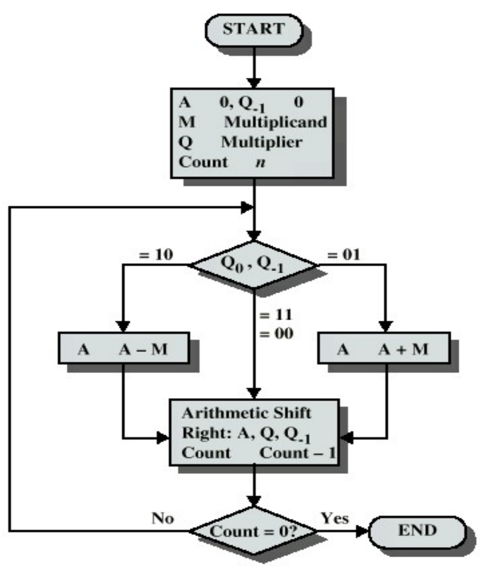

Figure 8.12 Booth's Algorithm for Twos Complement Multiplication

ALGORITHM:

- Multiplier and multiplicand are placed in the Q and M register respectively.
- Result for this will be stored in the AC and Q registers.
- Initially, AC and Q_{-1} register will be 0.
- Multiplication of a number is done in a cycle.
- A 1-bit register Q_{-1} is placed right of the least significant bit Q_0 of the register Q.
- In each of the cycle, Q_0 and Q_{-1} bits will be checked.
    - ✓ If Q_0 and Q_{-1} are 11 or 00 then the bits of AC, Q and Q_{-1} are shifted to the right by 1 bit.
    - ✓ If the value is shown 01 then multiplicand is added to AC. After addition, AC, Q_0, Q_{-1} register are shifted to the right by 1 bit.
    - ✓ If the value is shown 10 then multiplicand is subtracted from AC. After subtraction AC, Q_0, Q_{-1} register is shifted to the right by 1 bit.

CODE:

```
#include <iostream>
using namespace std;
void complement_2(int a[], int x[], int q);
void complement_1(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;
    }
    complement_2(a, x, n);
}
void complement_2(int a[], int x[], int n){
    int i, c = 0;
    for (i = 0; i < n; i++){
        a[i] = a[i] + x[i] + c;
        if (a[i] > 1){
            a[i] = a[i] % 2;
            c = 1;
        }
        else
            c = 0;
    }
}
void ashr(int ac[], int qr[], int &qn, int q){
    int temp, i;
    temp = ac[0];
    qn = qr[0];
    cout << "\t\tashr\t\t";
    for (i = 0; i < q - 1; i++){
        ac[i] = ac[i + 1];
        qr[i] = qr[i + 1];
```

```
   }
   qr[q - 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 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];
   complement_1(mt, brn);
   cout << "\nNo. of multiplier bit=";
   cin >> qrn;
   sc = qrn;
   cout << "Multiplier=";
   for (i = qrn - 1; i >= 0; i--)
      cin >> qr[i];
   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){
            complement_2(ac, mt, qrn);
            cout << "\t\tsubtracting BR\t";
            for (i = qrn - 1; i >= 0; i--)
               cout << ac[i];
            temp = 1;
         }
         else if (temp == 1){
            complement_2(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=4

multiplicand=1
0
1
0

No. of multiplier bit=4
Multiplier=0 0 1 0
qn      q[n+1]          BR              AC      QR              sc
                        initial         0000 0010               4
0       0               ashr            0000 0001               3
1       0               subtracting BR  0110
                        ashr            0011 0000               2
0       1               adding BR       1101
                        ashr            1110 1000               1
0       0               ashr            1111 0100               0
Result=1111 0100
PS C:\.vscode\college>
```

CONCLUSION: Thus, we have successfully implemented the Booth’s algorithm.