Batch: A 3 Roll No.: 16010121051 Type your text

Experiment / assignment / tutorial No. 2

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

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)

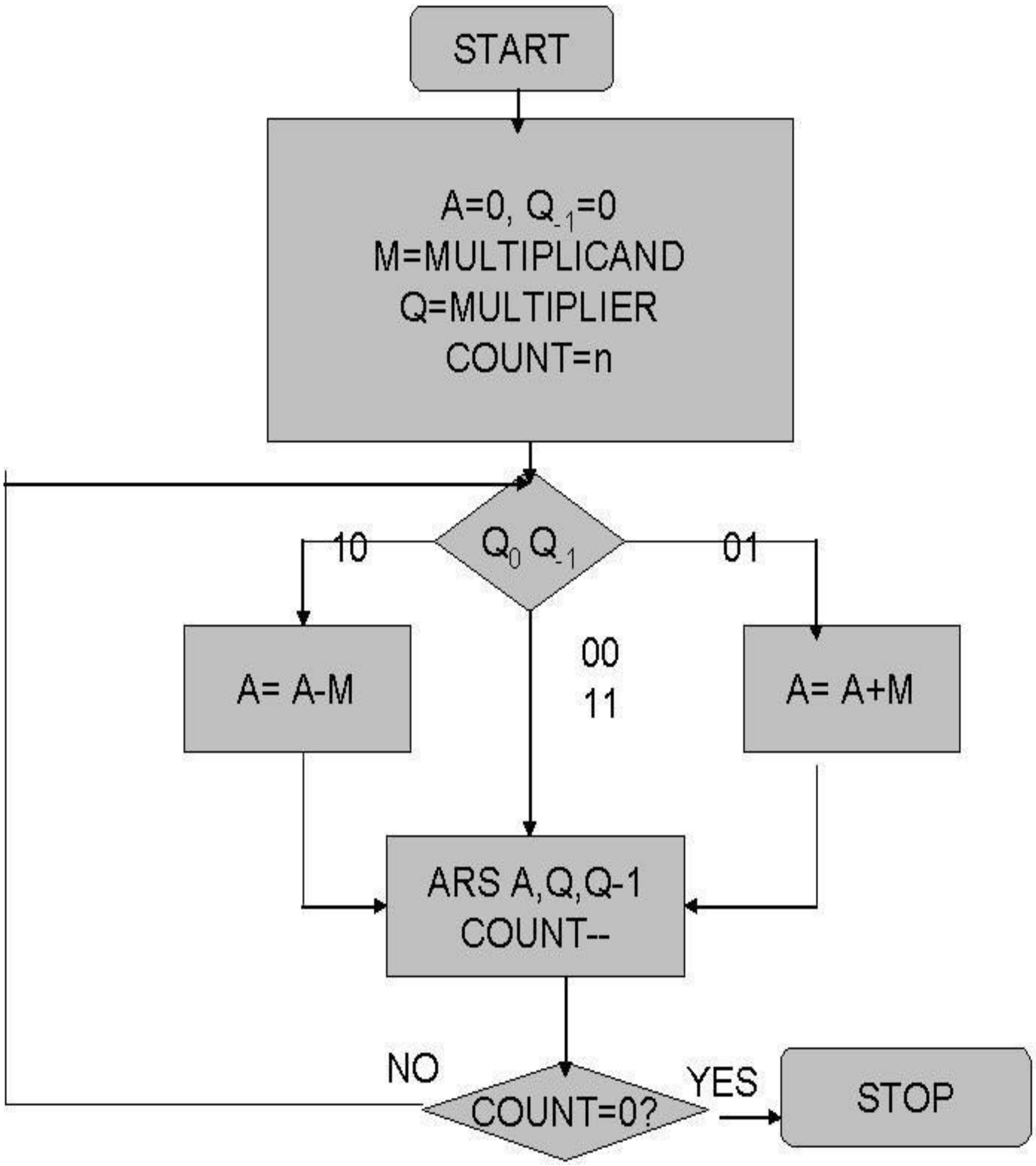
Books/ Journals/ Websites referred:

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:


START
A=0, Q-1=0
M=MULTIPLICAND
Q=MULTIPLIER
COUNT=n
Q0 Q-1
10
01
00
11
A= A-M
A= A+M
ARS A,Q,Q-1
COUNT--
NO
COUNT=0?
YES
STOP

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

Code:

```
#include<iostream>

using namespace std;

void add(int a[], int x[], int q);

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 q) {
  int i, c = 0;
  for (i = 0; i < q; 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 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 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 :-";
  cout<<"\n Pls give space beteen bits while inputting";
  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(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) {
          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);

  return 0;
}
```

Example: (Handwritten solved problem needs to be uploaded)

0011 1100 0010

1+8+16

Q. M=5 Q=5

M= 0101 -M = 1011

Q = 0101

A	Q	Q-1	
0000	0101	0	initial values
1011	0101	0	A = A-M } first cycle
1101	1010	1	shift
0010	1010	1	A = A+M } 2nd cycle
0001	0101	0	shift
1100	0101	0	A = A-M } 3rd cycle.
1110	0010	1	shift
0011	0010	1	A = A+M } 4th cycle.
0001	1001	0	shift

Ans is in A & Q = 00011001 = 25

Q. ~~M=12 Q=11~~

eg. (Done Roughly)

M = 7 , Q = 3

M = 0111 -M = 1001

Q = 0011

	A	Q	Q_{-1}	
	0000	0011	0	
	1001	0011	0	} ①
	1100	1001	1	
+M	1110	0100	1	} ②
0	0101	0100	1	} ③
1	0010	1010	0	
01	0001	0101	0	} ④

Ans → 0001 0101

= 21

Outputs :

```
--Enter the multiplicand and multipier :-
 Pls give space beteen bits while inputting
 Number of multiplicand bit=4

multiplicand=0 1 1 1

No. of multiplier bit=4
Multiplier=0 0 1 1
qn      q[n+1]          BR              AC      QR              sc
                        initial         0000 0011               4
1       0               subtracting BR  1001
                        ashr            1100 1001               3
1       1               ashr            1110 0100               2
0       1               adding BR       0101
                        ashr            0010 1010               1
0       0               ashr            0001 0101               0
Result=0001 0101
--------------------------------
Process exited after 20.18 seconds with return value 0
Press any key to continue . . .
```

```
--Enter the multiplicand and multipier :-
 Pls give space beteen bits while inputting
 Number of multiplicand bit=4

multiplicand=0 1 0 1

No. of multiplier bit=4
Multiplier=0 1 0 1
qn      q[n+1]          BR              AC      QR              sc
                        initial         0000 0101               4
1       0               subtracting BR  1011
                        ashr            1101 1010               3
0       1               adding BR       0010
                        ashr            0001 0101               2
1       0               subtracting BR  1100
                        ashr            1110 0010               1
0       1               adding BR       0011
                        ashr            0001 1001               0
Result=0001 1001
--------------------------------
Process exited after 12.26 seconds with return value 0
Press any key to continue . . .
```

Conclusion: Through this experiment, the method to multiply two binary numbers was learnt. New operations like right shift were also acquired. In the research for Post Lab Question I understood Booth's Recoding Method and its advantages.

Post Lab Descriptive Questions

Question: Explain advantages and disadvantages of Booth's algorithm.

Answer:

Advantages of booth's multiplication:

- Easy calculation of multiplication problem.
- Consecutive additions will be replaced.
- Less complex and ease scaling.

Disadvantages of booth's multiplication:

- This algorithm will not work for isolated 1's.
- It is time consuming.
- If digital gates are more, chip area would be large.

Question: Is Booth's recoding better than Booth's algorithm? Justify

Answer:

Advantage of Booth's recoding is that it reduces the number of 1's and increases the number of 0's in a binary number. Having more number of 0's is advantageous for easier calculation.

For Example: (01111)2 is equivalent to (+1 0 0 0 -1) in Booth Recoding.

Date: 16/09/2022 **Signature of faculty in-charge**