**Batch: A-2** **Roll No.: 16010122041**

**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)

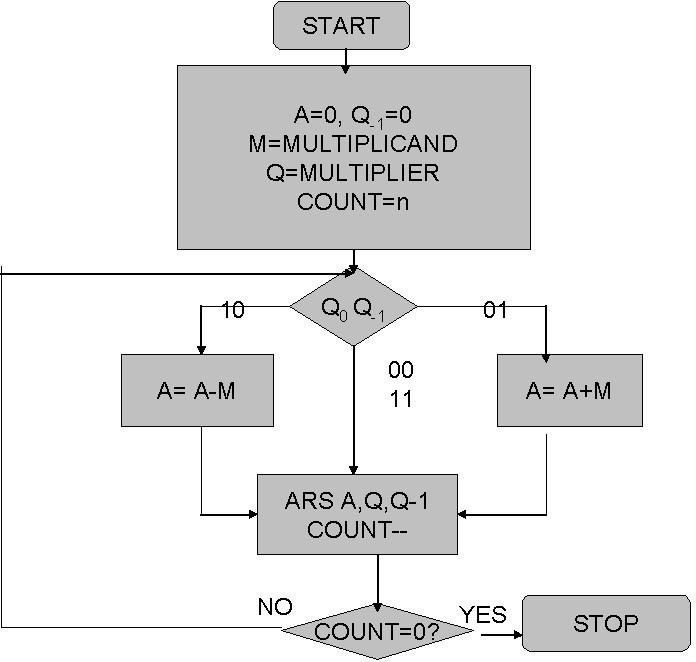
**Books/ Journals/ Websites referred:**

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:**



**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 Q0 and Q-1 and perform the respective operation.

|  |  |
| --- | --- |
| **Q0 Q-1** | **Operation** |
| 00/11 | Arithmetic right shift |
| 01 | A+M and Arithmetic right shift |
| 10 | A-M and Arithmetic right shift |

1. Repeat steps 5 till all bits are compared
2. Convert the result to decimal form and display
3. End

**Code:**

*public* *class* Main {

*public* *static* *int*[] binary\_m = {0*,* 0*,* 0*,* 1};

*public* *static* *int*[] binary\_m2 = {1*,* 1*,* 1*,* 1};

*public* *static* *int*[] binary\_q = {0*,* 0*,* 0*,* 1};

*public* *static* *int*[] binary\_a = {0*,* 0*,* 0*,* 1};

*public* *static* *int*[] binary3 = *new* *int*[4];

*public* *static* *int* q0 = 0;

*public* *static* *void* *add*(*int*[] binary\_m*,* *int*[] binary2*,* *int*[] binary3) {

*int* carry = 0*;*

*for* (*int* i = 3*;* i >= 0*;* i--) {

*int* sum = binary\_m[i] + binary2[i] + carry*;*

*if* (sum == 2) {

                carry = 1*;*

                sum = 0*;*

            } *else* *if* (sum == 3) {

                carry = 1*;*

                sum = 1*;*

            } *else* {

                carry = 0*;*

            }

            binary3[i] = sum*;*

        }

*int* temp*;*

*for* (*int* i = 0*;* i <= 1*;* i++) {

            temp = binary3[i]*;*

            binary3[i] = binary3[3 - i]*;*

            binary3[3 - i] = temp*;*

        }

    }

*public* *static* *int* *compare*(*int*[] binary\_q) {

*if* (binary\_q[3] == q0) {

*return* 0*;*

        } *else* *if* (binary\_q[3] == 1 && q0 == 0) {

*return* -1*;*

        } *else* *if* (binary\_q[3] == 0 && q0 == 1) {

*return* 1*;*

        }

*return* 0*;*

    }

*public* *static* *void* *shift\_right*(*int*[] binary\_a*,* *int*[] binary\_q) {

*int*[] full\_array = *new* *int*[8]*;*

*int*[] temp\_array = *new* *int*[8]*;*

*for* (*int* i = 0*;* i < 4*;* i++) {

            full\_array[i] = binary\_a[i]*;*

        }

*for* (*int* i = 4*;* i < 8*;* i++) {

            full\_array[i] = binary\_m[i - 4]*;*

        }

*for* (*int* i = 0*;* i < 7*;* i++) {

            temp\_array[i + 1] = full\_array[i]*;*

        }

        temp\_array[0] = full\_array[0]*;*

*for* (*int* i = 0*;* i < 4*;* i++) {

            binary\_a[i] = full\_array[i]*;*

            binary\_m[i] = full\_array[i + 4]*;*

        }

    }

*public* *static* *void* *main*(String[] args) {

        add(binary\_m*,* binary\_a*,* binary\_a)*;*

*for* (*int* i = 0*;* i < 4*;* i++) {

            System.out.print(binary\_a[i])*;*

        }

        System.out.println()*;*

        shift\_right(binary\_a*,* binary\_q)*;*

*for* (*int* i = 0*;* i < 4*;* i++) {

            System.out.print(binary\_a[i])*;*

        }

        System.out.println()*;*

*for* (*int* i = 0*;* i < 4*;* i++) {

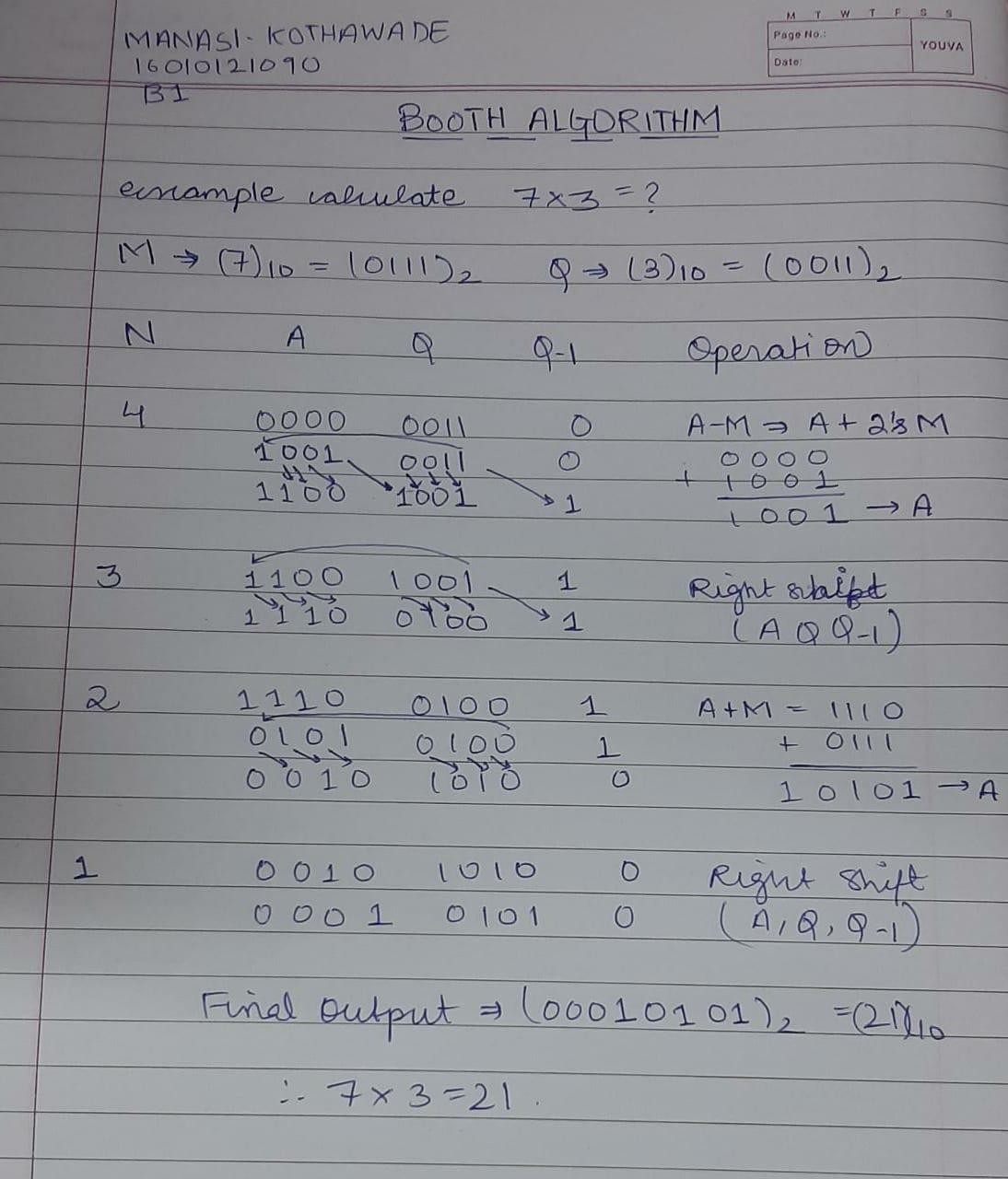
            System.out.print(binary\_m[i])*;*

        }

    }

}

**Example: (Handwritten solved problem needs to be uploaded)**



## 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: 12/08/2023 Signature of faculty in-charge