**Computer Organization**

**Booth's Algorithm**

**Documentation**

The Booth's Algorithm is an effective way to multiply two **Integers**. It follows the approach that the if the integers are M1 and M2 then M1 is the **multiplicand** and M2 is the **multiplier**. Then we initialize **q0** that is initially **"0"** and a that is also "0" with **as many zeroes** as in the binary of the **maximum** among M1 and M2. n is initialized to no. the max of no. of bits in M1, M2. We initialize multiplier in binary and then by comparing the last bit (least significant bit) of M1 and bit of q0 we perform following operations:

1. If they are **"00","11"** in order, then we just perform the **right shift operation.**

2. If they are **"10"**, then we perform the operation **A=A-M1** followed by the **right shift operation**.

3. If they are **"01"**, then we perform **A=A+M1** followed by the **right shift operation.**

4. These are performed until the value of the **n** becomes **0**. Each of above operation is followed by **decrementing** value of n by **1**.

5. After n becomes 0 the value of A is checked and if It starts with **1** then we give the result as the **2's complement** of the binary number converted into **decimal** and preceded by a negative sign otherwise we give the result as the **decimal** of the **binary number.**

The list of **functions** used:

1. **XOR ()**

2. **AND ()**

3. **OR ()**

4. **sumbin ()**

5. **bincon ()**

6. **com ()**

The function **XOR ()** is used to calculate the **XOR** of two bits and return value **"1"** or **"0"** depending on the XOR of the two. The function **AND ()** is used to calculate the AND of two bits and return value **"1"** or **"0"** depending on the AND of the two.

The function **OR ()** is used to calculate the OR of two bits and return value **"1" or "0"** depending on the OR of the two. The function **sumbin ()** takes two binary numbers in form of a string input and then returns the **binary number** of their result in string form. The function **bincon ()** converts the input string into binary form of the number and returns the **binary form as a string output.** The function **com ()** takes string as an input and converts the binary no. String into respective **2’s complement** and return it in **string** form.

The integer **n1 and n2** are used to take **input** values from the user in Integer Form.

**n3 and n4** are used to store the **binary value** of Integers n1 and n2.

**n5 and n6** stores the length of numbers **n3 and n4** that are binary form as numbers stored in binary form act as a String.

**Multiplicand** is the number that is to be **multiplied.**

**Multiplicand1** is the 2’s complement of the **multiplicand**.

**Multiplier** is the number which multiplicand is **multiplied to.**

**Output Format:**

Output format shows the intermediate steps with the value of **n, a, q0** and the step or operation performed:

For example, the multiplication of 4x5=20 is shown below with the steps performed. **4 0000 0101 0 Initial 4 1100 0101 0 A=A-M 3 1110 0010 1 RightShift 3 0010 0010 1 A=A+M 2 0001 0001 0 RightShift 2 1101 0001 0 A=A-M 1 1110 1000 1 RightShift 1 0010 1000 1 A=A+M 0 0001 0100 0 RightShift 00010100 20**

The output shows the **binary form** of the number and the **Integer form.**

**Assumptions:**

The Numbers entered by the user will be only Integers and no decimal numbers. Example 2 is an acceptable input whereas 1.28, 3.8 etc. are unacceptable input.

**Working of The Algorithm:**

The algo first takes the input for the users and initializes the n, a, q and q0. Then initializes the numbers in binary numbers and then checks the value of the least significant bit of the q and value of q0 and performs the operation of the Booth’s Algorithm according to the conditions of the algo as provided above.