COMPUTER ORGANISATION
BOOTH'S ALGORITHM

Garvita Jain (2018034)|Arnav Tandon(2018278)

Implemented using Java(11.0.3)

Booth's algorithm helps us to multiply and divide two signed binary integers in a better and efficient manner as it involves lesser number of additions and subtraction.

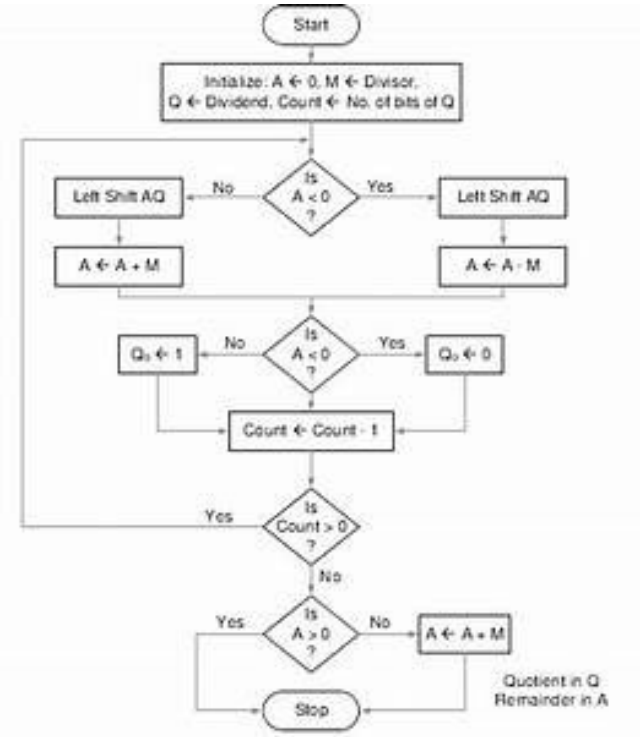
DIVISION:

Algorithm applied:

- Q = Dividend, M = Divisor, A = 0, n = number of bits in dividend)
- Check the sign bit of register A
- If it is 1 shift left AQ and perform A = A+M,if it is 0 shift left AQ and perform A = A-M (add 2's complement of M to A and store it to A)
- Again check the sign bit of register A
- If the sign bit is 1 Q[0]=0 otherwise Q[0]=1 (Q[0] means a least significant bit of register Q)
- Decrements value of N by 1
- If N is not equal to zero, start from Step 2 again otherwise go to next step
- If sign bit of A is 1 then perform A = A+M
- Register Q contain quotient and A contain remainder

Division Algorithm : FLOWCHART

- **Count is the length of the binary string that we are representing our numbers in the above flowchart**



Example:

N	M	A	Q	Action
4	00011	00000	1011	Start
		00001	011?	Shift Left AQ
		11110	011?	A=A-M
3		11110	0110	Q[0]=0
		11100	110?	Shift Left AQ
		11111	110?	A=A+M
2		11111	1100	Q[0]=0
		11111	100?	Shift Left AQ
		00010	100?	A=A+M
1		00010	1001	Q[0]=1
		00101	001?	Shift Left AQ

		00010	001?	A=A-M
0		00010	0011	Q[0]=1

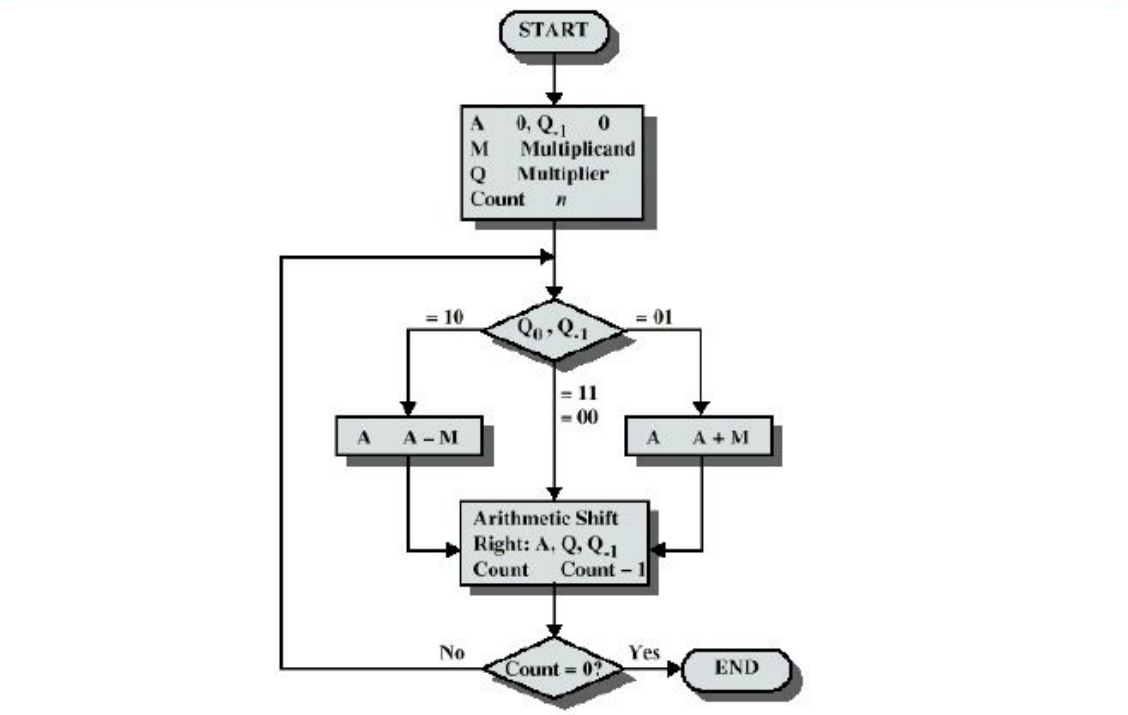
Complexity

The complexity of our code is O(n^2), where n denotes the length of our binary number. This is because
n-bit number runs through a loop that computes in n complexity and the bitshift works in linear time complexity thus complexity comes out to be 2(n^2) .

So, the calculated time complexity is O(n^2).

MULTIPLICATION:

Booth's Algorithm for Multiplication : FLOWCHART



Multiplication Algorithm:

- Determine the values of *A* and *B* and the initial value of *C*.
- Determine the two least significant bits of *C*.
    - If 01 find the value of *C+B*. Ignore any overflow.
    - If 10 find the value of *C-B*. Ignore any overflow.
    - If 00,Use *C* directly in the next step.
    - If 11,Use *C* directly in the next step.
- Arithmetically shift the value obtained in the 2nd step by a single place to the right. Let *C* now equal this new value.
- Repeat steps 2 and 3 until they have been done *i(the maximum number of bits = n)* times.
- Drop the least significant bit from *C*. This is the product of **A** and **B**.

Example :

OPERATION	AC	MR	QN+1	SC
	0000	1001	0	4
AC + MD' + 1	0101	1001	0	
ASHR	0010	1100	1	3
AC + MR	1101	1100	1	
ASHR	1110	1110	0	2
ASHR	1111	0111	0	1
AC + MD' + 1	0010	0011	1	0

(Source : https://www.geeksforgeeks.org/computer-organization-booths-algorithm/)

Test cases:

1) 8,11
2) 11,8
3) 8,8
4) -2,5
5) 5,-2

6) -2,-4
7) -5,-5
8) 35,11

Reference Used :

https://en.wikipedia.org/wiki/Booth%27s_multiplication_algorithm
https://www.geeksforgeeks.org/computer-organization-booths-algorithm/
https://en.wikipedia.org/wiki/Division_algorithm
http://www.pitt.edu/~juy9/142/slides/L5-Multiplication.pdf