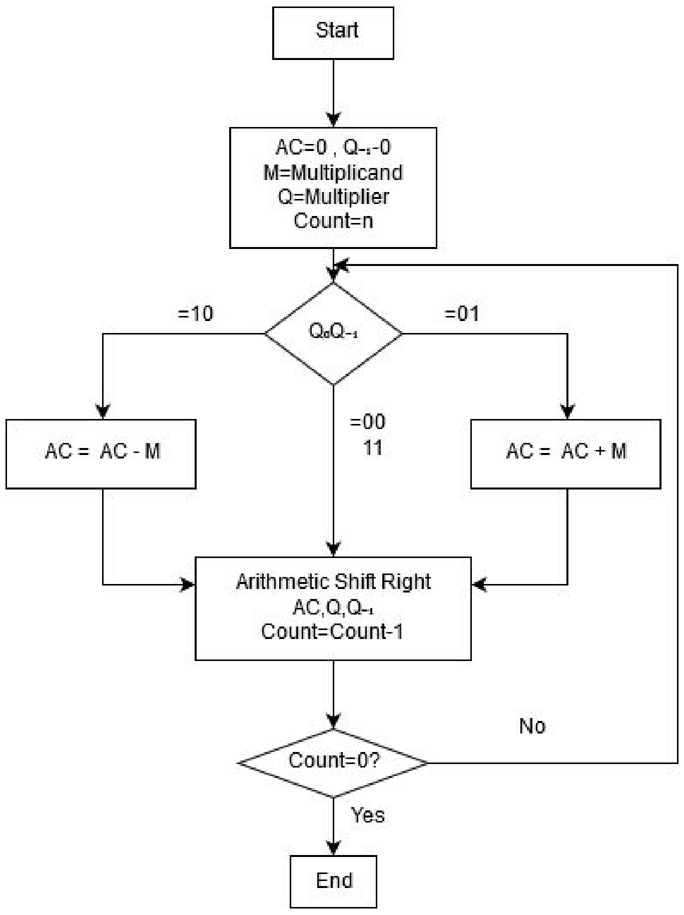
Computer Organisation Project 2

Navneet Agarwal - 2018348
Sarthak Arora - 2018307

Booth's Multiplication Algorithm :

1. Booth's Multiplication follow a simple and efficient method for multiplying binary integers. Four registers are used in this process, namely AC,Q and M and Q_{-1}.
The results of multiplication are obtained in A and Q registers.
2.
   - A and Q registers are initialized to zero.
   - Q and M are initialized to Multiplier and Multiplicand respectively.
   - Counter is initialized to the number of bits in the multiplier.
3. The two bits Q_0, last bit of Multiplier and Q_{-1} are checked.
4.
   - If Q_0Q_{-1} comes out to be 10, a special operation is done to AC, value of AC is updated to AC-M i.e. Multiplicand is subtracted from AC.
   - If Q_0Q_{-1} comes out to be 01, a special operation is done to AC, value of AC is updated to AC+M i.e. Multiplicand is added to AC.
   - No special operation is done if 00 or 11 are encountered.
5. After this, All three AC, Q and Q_{-1} are subjected to Arithmetic right shift and Counter is decremented.
6. This process continues until the Counter is encountered to be null.



7. After this, the answer of multiplication is obtained by joining the results of AC and Q registers.
8. For obtaining the value of the product in decimal, We simply convert the value obtained by joining AC and Q from binary to decimal in case of both positive or both negative numbers. But in case only one of the numbers is negative, we would have to convert the value obtained by joining AC and Q to its two's complement and then convert it to decimal with a negative sign to obtain the correct answer.

Test Cases :

Input	**Output**	**Output**
Multiplicand,Multiplier	**Product (Binary form)**	**Product (Decimal form)**
7,3	00010101	21
-7,3	11101011	-21

7,-3	11101011	-21
-7,-3	00010101	21
7,0	00000000	0

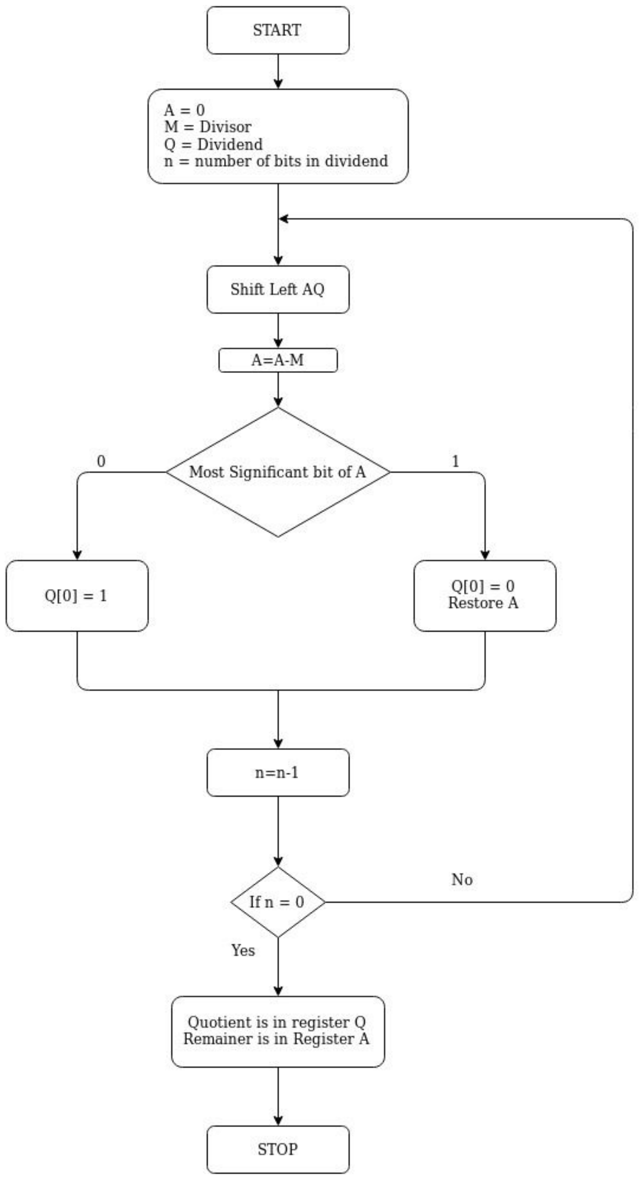
Complexity Analysis :

Major steps that account for the time complexity of Booth's Algorithm are Binary Addition in each iteration, till counter is null which gives the O(n^2) factor, the calculation of Two's Complement of the multiplicand which contributes O(n) and the Arithmetic right shift after each iteration which contributes O(n) to the Time Complexity. There are several other minor steps which contribute O(1) each and are neglected in the final time complexity.

Complexity of Booth's Multiplication() = O(5+2+2+2n+n+2+n*((n+4))+1)
= O(12+3n+(n^2+4n))
= O(n^2 + 7n + 12)
= O(n^2)

Division Algorithm

Since, we are dealing with signed numbers so we will follow a convention here Dividendide the two numbers by making them both positive and finally change answers according to the following convention. If both the Dividend and Divisor have the same sign then quotient will be positive.If the sign of Dividend is negative then sign of remainder will also be negative.So, we need an algorithm to divide unsigned integers. We have used **Restoring Division Algorithm for unsigned integers**.The flowchart for this algorithm is shown below.


START
A = 0
M = Divisor
Q = Dividend
n = number of bits in dividend
Shift Left AQ
A=A-M
Most Significant bit of A
0
1
Q[0] = 1
Q[0] = 0
Restore A
n=n-1
If n = 0
No
Yes
Quotient is in register Q
Remainer is in Register A
STOP

Restoring Division Algorithm for unsigned integers:

1. If the Divisor is zero ,ZeroDivisionError is thrown
2. Both the Divisor and Dividend are converted to positive numbers.
3. Initial values of the variables and registers are as follows:
   n = Number of bits in dividend
   Accumulator, A= 00….(n+1)..0
   Quotient Register, Q = Dividend
   Register M = Dividend
4. After this we get a while loop which is executed till n>0.The loop includes the following statements:
   i) A left shift operation is done on AQ leaving the least significant bit of Q empty for a while.
   ii) A=A-M operation is done by calculating 2's compliment of M and adding it to A.
   iii) If Most Significant bit of A is 0 then the Least Significant bit of Q is set to 1.

iv) If Most Significant bit of A is 1 then the Least Significant bit of Q is set to 0 and Accumulator value is restored to the value before A-M
v) n is decreased by 1.

5. After the loop has ended we get quotient in register Q and remainder in Accumulator A.
6. The values of Q and A are then changed according to sign of Divisor and Dividend and the 4 conditions described earlier.

Test Cases :

Input	**Output**	**Output**
Dividend ,Divisor	**Quotient ,Remainder (Binary form)**	**Quotient ,Remainder (Decimal form)**
32,3	001010 , 0000010	10,2
1000,-502	1111111111 , 00111110010	-1,498
-999 33	1111100010 , 11111110111	-30,-9
-878 -98	0000001000 , 11110100010	8,-94
913,0	ZeroDivisionError : Divisor cant be	zero

Complexity Analysis :

First we complete the register M and Accumulator A with n+1 bits, so worst case for this takes **(n+1) steps & (n+1) steps** respectively.The while loop runs for n iterations and inside it we have (12 +(7n +5) + (n+1)) steps. As function for binary addition has n iterations with 7 steps inside it and additional 5 steps outside the loop.It also completes the zero of LeftshiftA with n+1 iterations in the worst case.

So, the complexity of RestoringDivision() =O((n+1)+(n+1)+ n(12 +(7n +5) + (n+1))+24)
[Since,24 steps apart from the mentioned]

Complexity of RestoringDivision() = O((n+1)+(n+1)+ n(12 +(7n +5) + (n+1))+24)
= O(2n + 26 + n(8n +18)
= O(2n + 26 + 8n^2 + 18n)
= O(8n^2 + 20n +26)
= O(n^2)

References:

https://www.geeksforgeeks.org/restoring-division-algorithm-unsigned-integer/
http://homepages.cae.wisc.edu/~ece352/fall00/project/pj2.pdf
https://www.geeksforgeeks.org/computer-organization-booths-algorithm/
https://www.youtube.com/watch?v=DHhcnjEKEFo
https://cs.stackexchange.com/questions/86500/understanding-2s-complement-multiplication-using-booths-algorithm