COMPUTER ORGANISATION

CS112

PROJECT-2

"IMPLEMENT THE BOOTH'S ALGORITHM FOR MULTIPLYING BINARY NUMBERS"

SUBMITTED BY

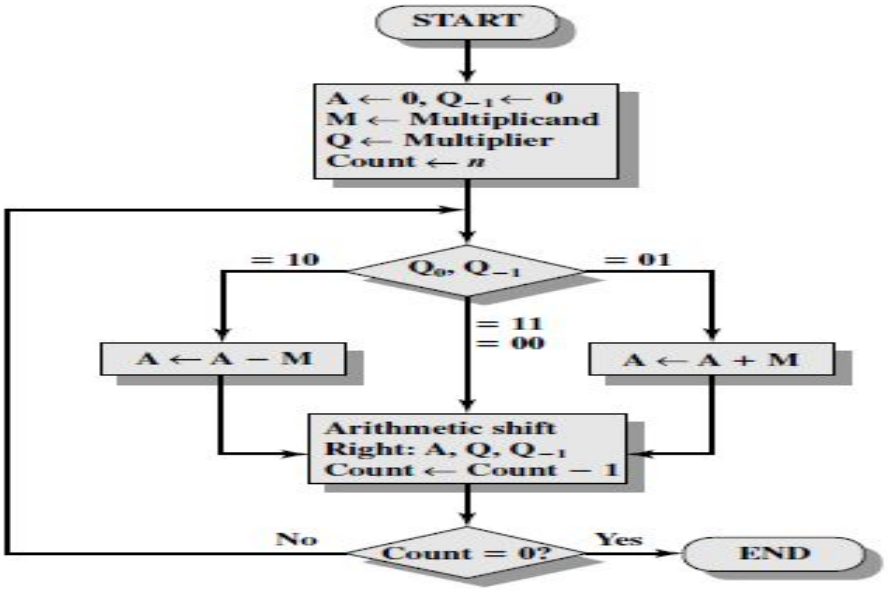
DOLLY SIDAR(2019304)

MULTIPLICATION

I followed Booth's Algorithm to multiply two numbers. Firstly it will check the inputs are in binary or not, if not then the numbers are converted to their respective signed binary strings(for negative numbers, I converted them to twos complement) and then I performed booth's algorithm to find the result.i made a various function to be performed like isBinary(), binary(),twosComplement(), ARM(), add(),boothMul() which is the main function for performing multiplication.

The operations carried out by functions are as follows-
1.isBinary()- check whether the given integer is binary or not.
2. binary()- Converts the number in binary format.
3. twoscomplement()- Converts the negative number
in twos complement form.
4. add()- Used to add two binary strings.
5. ARM()- Used to do an arithmetic right shift of the binary strings.
6. boothMul()- Two numbers (Multiplier and
Multiplicand) is passed and then it uses all above
functions to perform the booth's multiplication.

Flow Chart of the algorithm-



As in all multiplication schemes, the booth algorithm requires examination **of the multiplier bits** and shifting of the partial product. Prior to the shifting, the multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to the following rules:

1. The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier
2. The multiplicand is added to the partial product upon encountering the first 0 (provided that there was a previous '1') in a string of 0's in the multiplier.
3. The partial product does not change when the multiplier bit is identical to the previous multiplier bit.

Time Complexity of the Algorithm- As evident from flowchart we loop over this flowchart for n (count) times (which is the number of bits in binary). And in every loop we do a right shift operation which iterates over all the bits and shifts them one place to right. So, basically in every loop (outer loop), it performs another n operations (for shifting) which makes it's time complexity O(n 2). Or, in terms of decimal number m (i.e., multiplier/multiplicand) it becomes O((log 2 m) 2).

INPUT FORMAT

Enter inp1 then inp2 and also the num of register.

OUTPUT FORMAT

inp1:7

inp2:3

register size :4

ACC	Q	q	operation	step count
0000	0011	0	initial	4
1001	0011	0	A= A-M	
1100	1001	1	rightShif	3
1110	0100	1	rightShift	2
0101	0100	1	A= A+M	
0010	1010	0	rightShift	1
0001	0101	0	rightShift	0

In decimal: 21

In binary: 00010101

Source used:

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