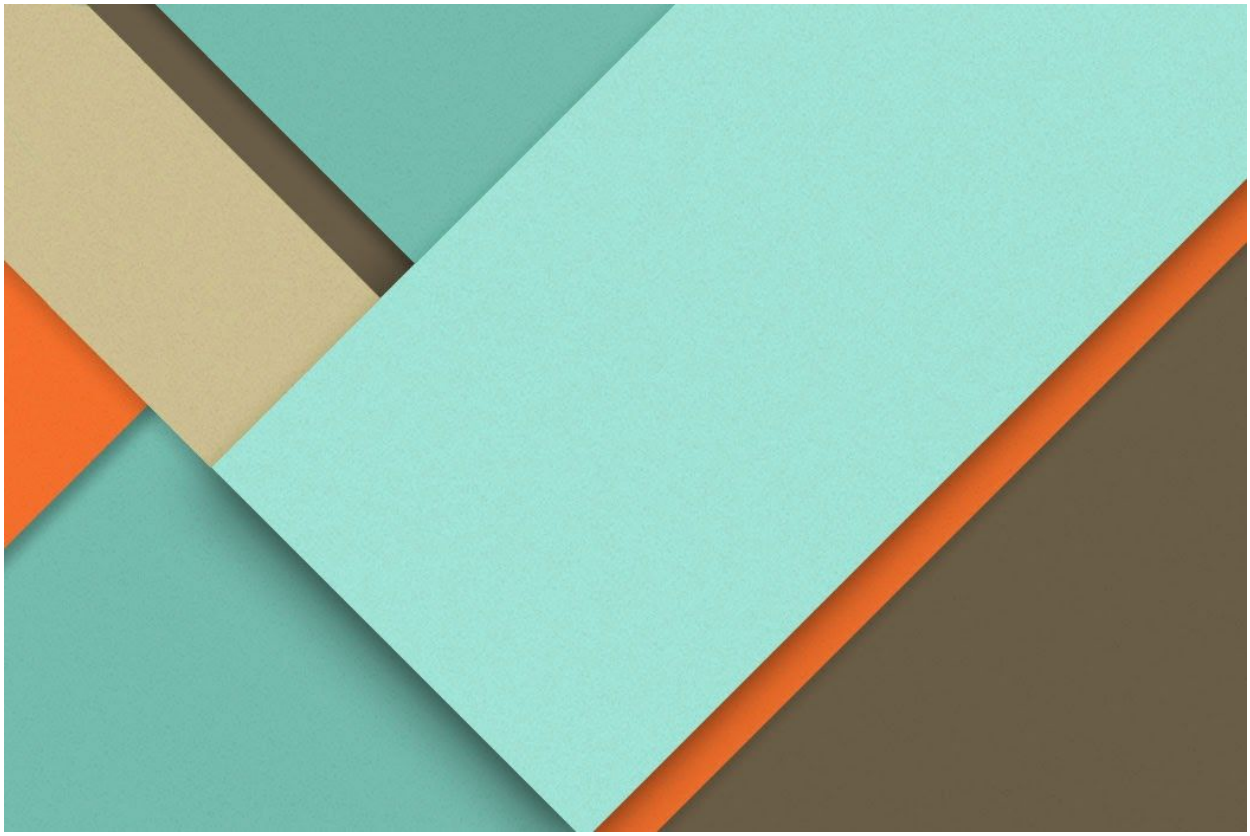


DOCUMENTATION

PROJECT 2 (BOOTH'S ALGORITHM)

2020

Abhishek Chaturvedi
2019401

WORKING OF THE ALGORITHM

1. The program implements Booth's algorithm in python.
2. It takes two integer inputs and returns the product of the numbers.
3. The inputs are converted into binary form for implementing booth's algorithm.
4. The multiplier is converted to the required product in binary form using the accumulator.
5. Initially, a zero bit(Q-1) is added at the end of the multiplier and the operations are performed based on the combination of values of the Q0 and Q-1 bit.
6. The three basic operations are performed:
   - Arithmetic right shift (ARS)
   - Add and Arithmetic right shift (ARS)
   - Subtract and Arithmetic right shift (ARS)
7. If the Q0 and the Q-1bit are both 0 or both are 1, then we use the operation ARS.
8. If the Q0 bit is 0 and Q-1 bit is 1, then we use Add and ARS; the multiplicand is added to the current value of the accumulator in binary form and then ARS is performed.
9. If the Q0 bit is 1 and Q-1 bit is 0, then we use Subtract and ARS; the multiplicand is subtracted from the current value of the accumulator in binary form and then ARS is performed.
10. One of the above three steps are performed for 16 steps before getting the final product in binary form.
11. For negative numbers the algorithm first takes its two's complement and then uses it to get the product using the above steps and if the product is negative the result is printed in two's complement form.
12. The program also prints the value of the product in decimal form.

WORKING OF THE CODE

- **The multiplication is carried out by the following functions:**

➢ **booths(): It is the main function which is responsible for calling other functions. It also prints the accumulator and multiplier at each step. Its parameters are numbers in string form and it returns the final product string in binary form. It mainly calls perform_operation() function.**

➢ **perform_operation(): It checks the last two bits of multiplier at each step and performs one of three operations(mentioned above) accordingly and it returns the product in string binary form and the operation which was performed also in string form. It takes three string parameters which are the accumulator, the multiplicand and the last two bits of the multiplier. It calls functions shift(), subtract() and Add().**

➢ **subtraction(): It is a simple function which subtracts two values in binary form. Its parameters are two string numbers and it returns the result in binary string form.**

➢ **Add(): It simply adds two values in binary form and returns the binary string form. It takes parameters which are two string numbers.**

➢ **shift(): This function simply shifts the binary string by one position to the right. Its return value and parameter are both strings in binary form.**

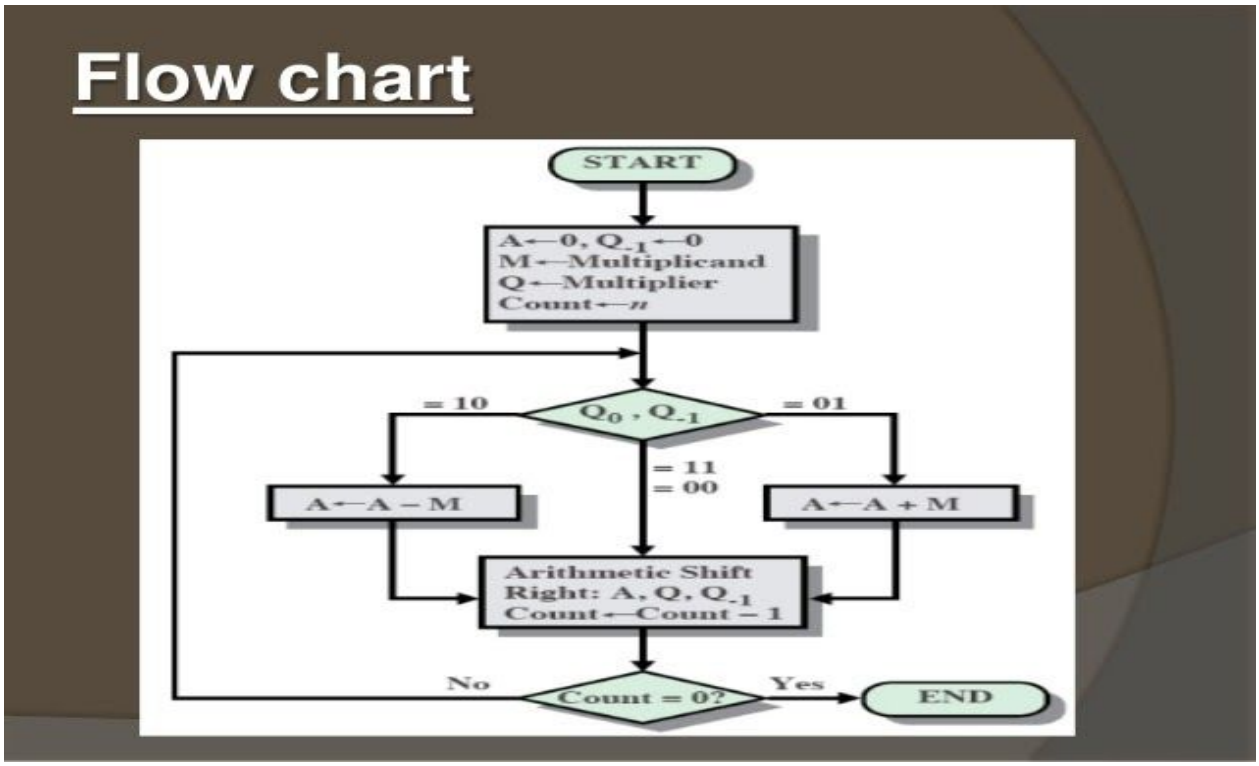
➢ **positiveBinaryConversion(): It converts a positive integer into binary form. It takes a non negative integer as parameter and returns a string in binary form.**

➢ **twoscompliment(): It converts a negative number into its two's complement form. It takes the negative integer as parameter and returns its two's complement string in binary form.**

➢ **binToDec(): It converts a binary string into its decimal value and returns the decimal value as string. It takes a string as a parameter which is in binary form.**

- **The code firstly checks that if the inputs are negative or positive and accordingly calls functions to convert it into binary form. The returned values are passed as parameters to the main function which is booths().**

ANALYSIS OF THE CODE



Time Complexity of the Algorithm

- As evident from flowchart we loop over this flowchart for n (count) times (which is 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((log2m)2).

CONSTRAINTS

- The product in binary form can be only upto 16 bits in length.
- The maximum value of product in decimal can be 65535.

ASSUMPTIONS

- If the value of the product is not expressible in 16 or less bits then the product is assumed to be 0 and the value 0 in decimal and binary form is printed.

References:

- https://en.wikipedia.org/wiki/Booth%27s_multiplication_algorithm
- Google images for diagram