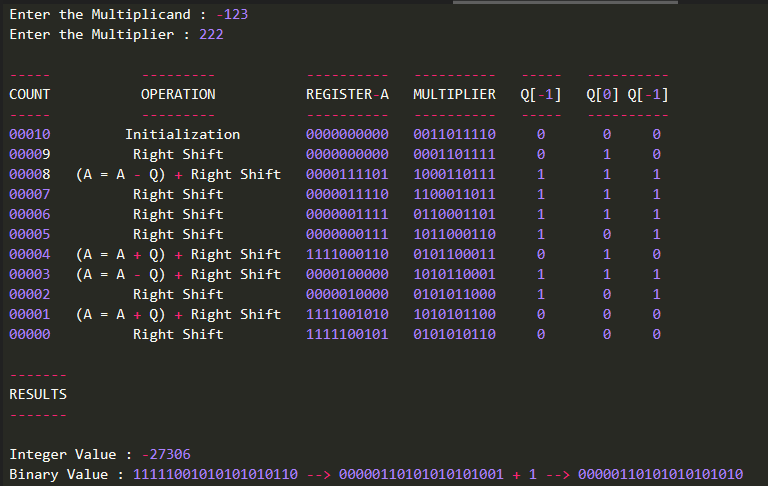
**Booth’s Algorithm**

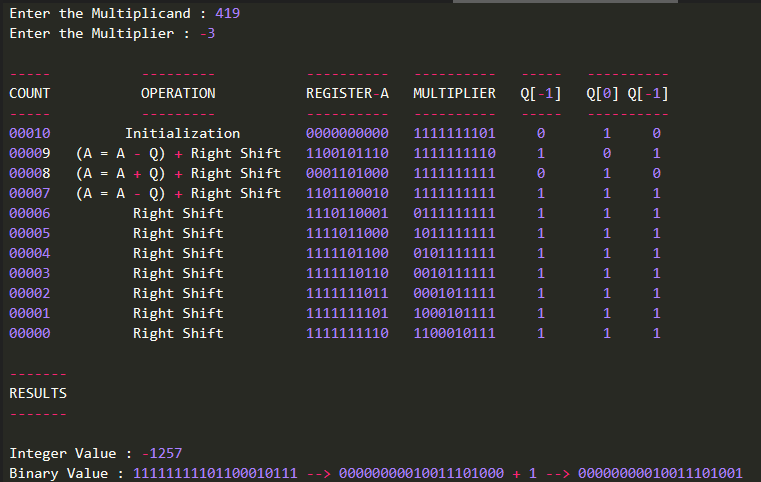
Booth’s algorithm is an efficient method to multiply two signed binary numbers. The method requires a multiplier Q (n bits), a multiplicand M (n bits), a register A (2n bits), and an additional register Q-1 (1 bit). The 2n-bits register and the 1-bit register are initially initialized to zero. During the calculation, Q-1 stores the last bit of Q in the previous step, and additions and subtractions are performed over A.

The following are the three conditions needed for calculating the correct result.

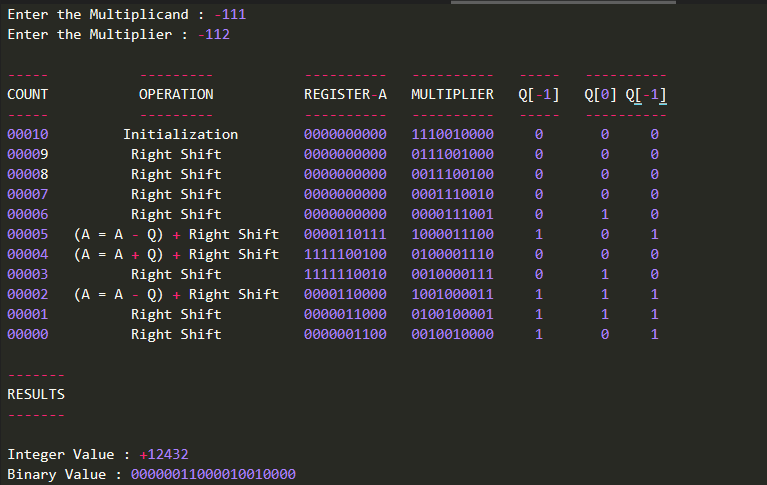
1. If Q0 Q-1 is equal to “10”, then perform A = A – M and the right shift.
2. If Q0 Q-1 is equal to “01”, then perform A = A + M and the right shift.
3. If Q0 Q-1 is equal to “11” or “00”, then just perform the right shift.
   1. Explanation of Code
4. The code includes a class **Operations** that has the following functions.
5. **Add ()** : This function accepts two string parameters. The two parameters are binary strings. This function adds the two binary strings and returns the result.
6. **BinaryValue ()** : This function accepts a positive integer value and converts the integer into its binary form.
7. **Binary ()** : This function accepts an integer and calls the BinaryValue () function to get the binary form of the positive integer. Further, depending upon the sign, the integer is converted into its final binary form. If it is a negative integer, then this function returns the 2’s complement of the number. If it is a positive integer or zero, then it returns the output of the function BinaryValue ().
8. The code includes some essential variables, which are as follows.
9. **Bits** : It stores the value of n, which is 10 for this program.
10. **A** : A n-bit register.
11. **Multiplicand** : It stores the binary value of the Multiplicand, M.
12. **NegativeMultiplicand** : It stores the binary value of the Negative of Multiplicand, -M.
13. **Multiplier** : It stores the binary value of the Multiplier, Q.
14. **Q0** : Stores the last bit of the current Multiplier.
15. **Q** : Stores the last bit of the previous value of Multiplier.
16. **String** : It stores a string of 2n + 1 – bits that includes the values of A, Multiplier, and Q (Q-1).
17. The code outputs the integer value and the binary value of the product and a table which shows the calculation process.
18. The table has the following 6 table headers.
19. **Count** : It displays the step numbers
20. **Operation** : It displays the operation performed. (Right Shift, (A = A - Q) + Right Shift, (A = A + Q) + Right Shift, Initialization)
21. **Register – A** : It displays the value of variable A.
22. **Multiplier** : It displays the value of the variable Multiplier.
23. **Q[-1]** : It displays the value of variable Q.
24. **Q[0] Q[-1]** : It displays the value of variables Q0 and Q .
    1. Constraints and Assumptions
25. The value of n (number of bits) considered in the program is 10.
26. The Multiplier and the Multiplicand should be strictly in the range (-512, 512).
27. The product obtained will be of 20 bits (2n).
    1. Examples
28. -123 x 222



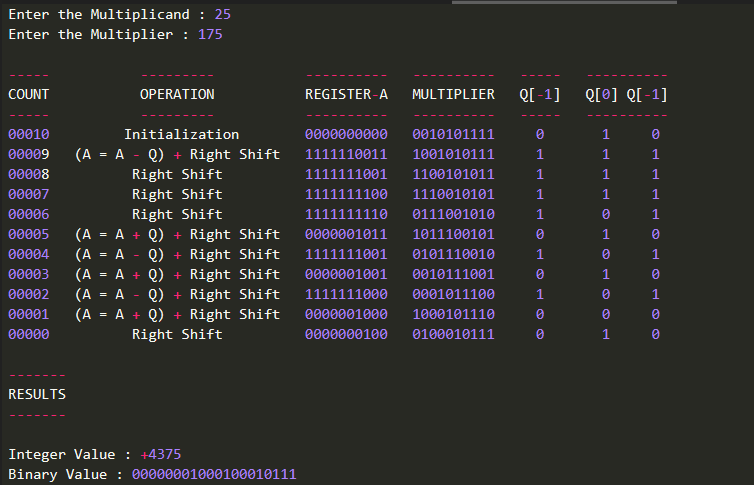
1. 419 x -3



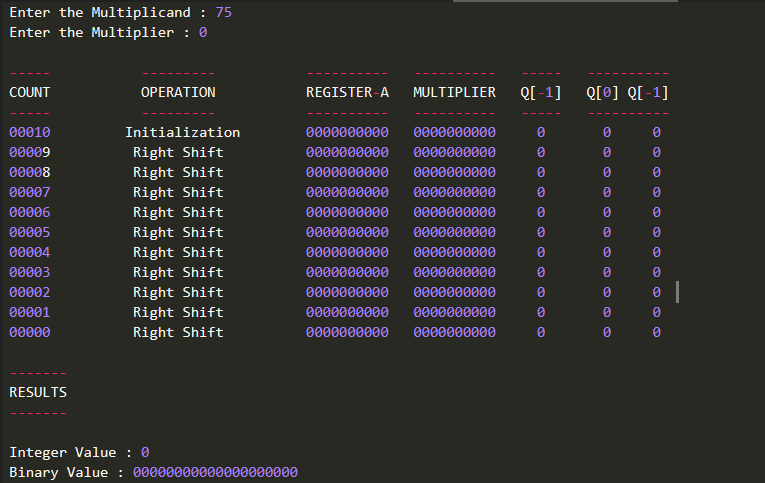
1. -111 x -112



1. 25 x 175



1. 75 x 0



1. 0 x -411

