REPORT

MULTIPLICATION AND DIVISION OF SIGNED BINARY NUMBERS

By- Amandeep Kaur (2018014)
Deepi Garg (2018389)

MULTIPLICATION:

Algorithm:

1. Initialise a variable (here, A) to 0 (With the same number of bits as the multiplicand and multiplier).
2. Initialize another variable (here, Q_{-1}) to 0.
3. Let the multiplicand be B and multiplier be Q.
4. Let count be the minimum number of bits required to represent B and Q as signed binary numbers.
5. Define Q_0 as the least significant bit of the multiplier(here, Q).
6. If $Q_0 = 0$ and $Q_{-1} = 1$, then add B to A.
7. If $Q_0 = 1$ and $Q_{-1} = 0$, then subtract B from A.
8. Perform arithmetic right shift on the binary number formed by appending Q, Q_{-1} to A. Eg- if A = 1101, Q = 0010, $Q_{-1} = 1$ then ARS results is A = 1110, Q = 1001, $Q_{-1} = 0$, dropping the previous value of Q_{-1}
9. Decrement count by 1.
10. If count > 0, jump to Step 6 , else no further computation is needed and the answer is A,Q i.e. Q appended to A.

Flowchart for Booth's Algorithm-

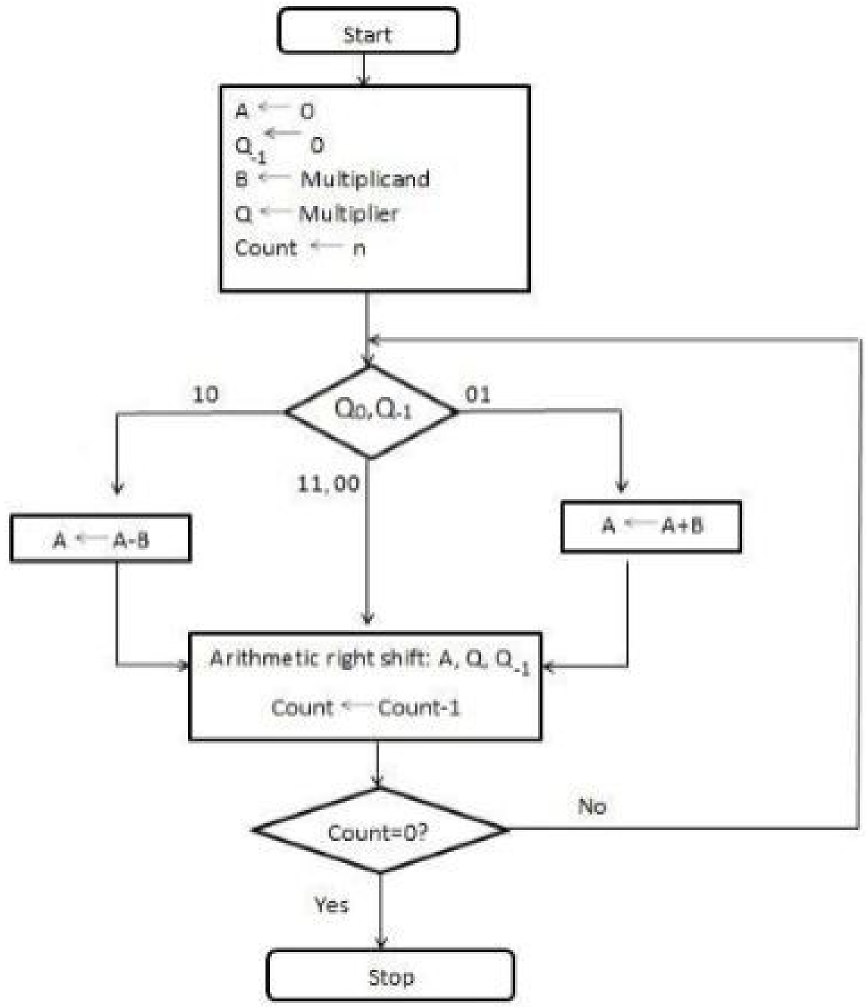

Image Src: https://www.ques10.com/p/8538/draw-flowchart-of-booths-algorithm/

Complexity Analysis of Booth's Algorithm-

If the number of bits is n, Complexity can be calculated as O(n)(number of iterations of the main loop)*(complexity of addition + Complexity of right shift(only when implemented in hardware)) = O(n*n)

Test Cases used

- 7, 2 → Binary: 0001110 Decimal: 14
- 4, -9 → Binary: 111011100 Decimal: -36
- -10, 7 → Binary: 110111010 Decimal: -70
- -126, -4 → Binary: 000000111111000 Decimal: -36

DIVISION:

Algorithm:

1. Initialise a variable (here, A) to 0.
2. Let the Divisor be M and the Dividend be Q.
3. Let count be the number of bits required to represent M and Q as signed binary numbers.
4. Perform left shift on the binary number formed by appending Q appended to A. Eg- if A=1001, Q=1100, then left shifting results in A=0011 Q=1000.
5. Subtract M from A (or add the 2's complement of M to A)
6. If A<0, then set the last bit of Q to 0, and add M to A.
7. If A>=0, then set the last bit of Q to 1.
8. Decrement count by 1.
9. Repeat steps 4 to 8 till count becomes 0.
10. The quotient is given by Q. The Remainder is given by A.

The above algorithm gives a result for the division of unsigned numbers which can be modified to be used for signed numbers in the following way:
Dividend = divisor * quotient + remainder
If dividend and divisor carry opposite signs, then the quotient is negative
If dividend is negative, then the remainder is also negative.

Flowchart for Division Algorithm-

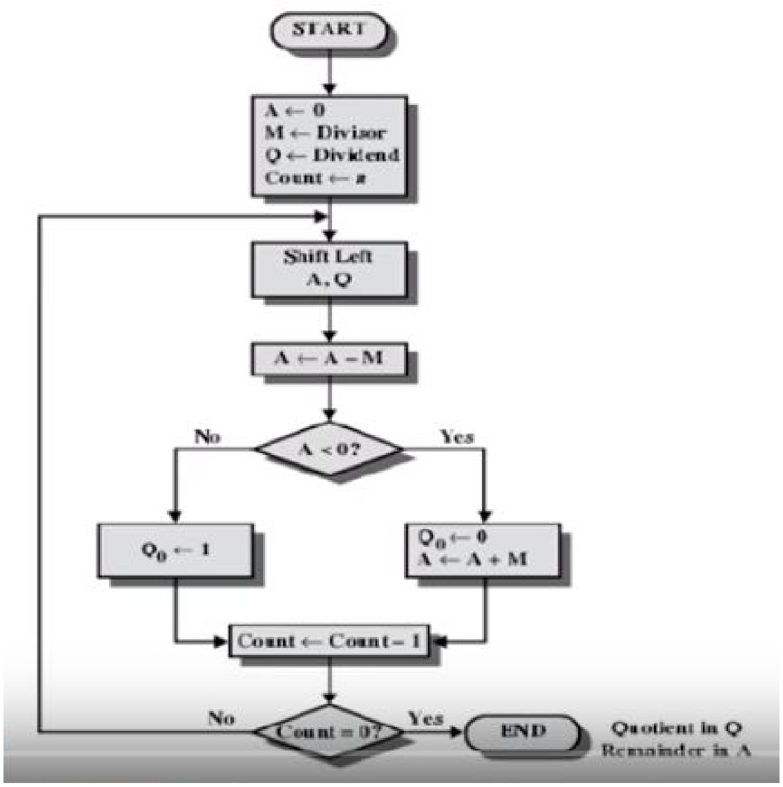

Image src: https://www.youtube.com/watch?v=G5rXmilRiZM

Complexity Analysis of the Algorithm-

If the number of bits is n, the complexity can be calculated as O(n)(number of iterations of the main loop)*(Complexity of Left shift(when implemented in hardware) + Complexity of Subtraction), which approximates to O(n*n)

Test Cases used

- 7, 2 →
  Quotient(Binary): 0011
  Remainder(Binary): 0001
  Quotient(Decimal): 3
  Remainder(Decimal): 1
- 4, -9 →
  Quotient(Binary): 00000
  Remainder(Binary): 00100
  Quotient(Decimal): 0
  Remainder(Decimal): 4
- -10, 7 →
  Quotient(Binary): 11111
  Remainder(Binary): 11101
  Quotient(Decimal): -1
  Remainder(Decimal): -3
- -126,-4 →
  Quotient(Binary): 00011111
  Remainder(Binary): 11111110
  Quotient(Decimal): 31
  Remainder(Decimal): -2