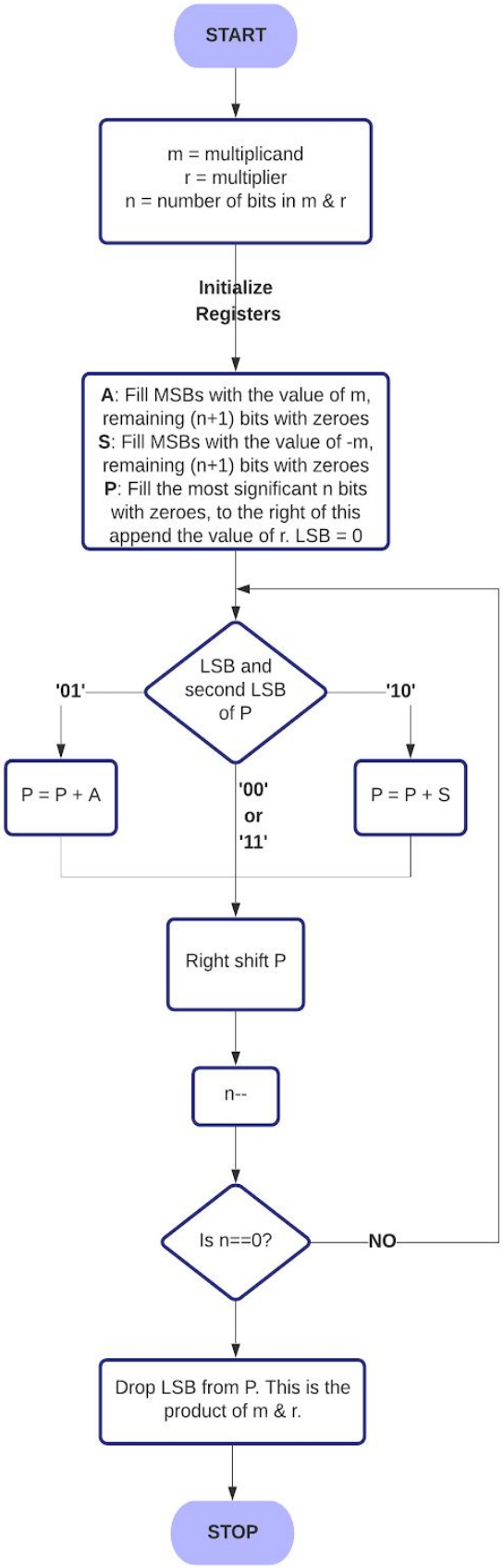
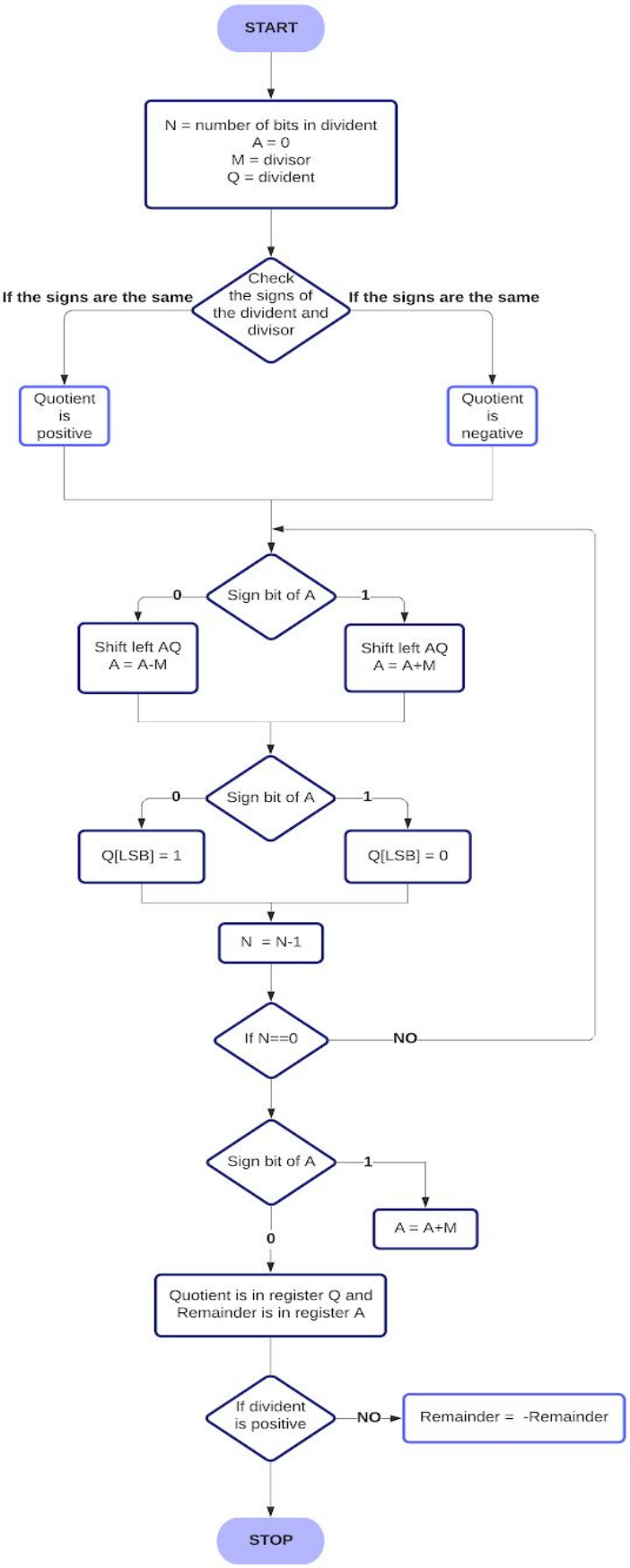
BOOTH'S MULTIPLICATION AND NON-RESTORING DIVISION

BOOTH'S MULTIPLICATION FOR SIGNED INTEGERS



NON-RESTORING DIVISION FOR SIGNED INTEGERS



Algorithms

Both the algorithms use and follow 2's complement representation of a binary number.

Booth's multiplication algorithm

1. A=Fill n MSBs with m(multiplicand), remaining n+1 bits with 0s, S=Fill n MSBs with -m, remaining n+1 with 0s. P=Fill n MSBs with 0s, to the right append r(multiplier), LSB=0.
2. If LSB & 2nd LSB of P == '11' or '00', proceed. If '01', P=P+A, else if '10', P = P+S.
3. Right shift P, decrement n(number of bits in m & r).
4. If n is not 0, return to step 2, otherwise
5. Drop LSB from P. This is the product of m & r.

Signed non-restoring division algorithm

1. Initialise registers (Q = Dividend, M = Divisor, A = 0, n = number of bits in dividend)
2. If dividend & divisor have the same sign, the quotient is positive, otherwise negative.
3. Left shift AQ, if sign bit of A is 1, A = A+M, else, A = A-M
4. If sign bit of A is 1 Q[LSB] = 0, otherwise Q[LSB] = 1
5. Decrement, N by 1, if N is not equal to zero go to step 2, otherwise, proceed.
6. If sign bit of A is 1, perform A = A+M
7. Register Q contain quotient. If the dividend is positive, A contains remainder, otherwise remainder = -remainder.

Complexity Analysis

Both the multiplication and division algorithms use the same addition function which has O(n) complexity. All the other helper functions have a maximum complexity of O(n). The main algorithm logic is executed n times in a loop (at most). Therefore, **O(** n^2 **)** is the tightest bound for Booth's multiplication algorithm as well as for the signed non-restoring division. *n* is the size of the operand registers or the number of bits of the operands, taken by us to be 12.

Testing

`256 and -51` Multipln -13056(111111111100110100000000), Div R=1(000000000001), Q=-5(111111111011)

`-100 and -89`Multipln 8900(000000000010001011000100),Div R=-11(111111110101), Q=1(000000000001)

`712 and 0` Multipln 0(000000000000000000000000),Div:divisor is zero, algorithm does not proceed further

`0 and -138` Multipln 0(000000000000000000000000), Division: Remainder = 0, Quotient = 0

`0 and 0` Multipln 0(000000000000000000000000), Div:divisor is zero, algorithm does not proceed further

Automated tests were used to verify the algorithms for all possible multiplications and divisions from -1000 to +1000. These can be run using `$ java Main test`

References

1. https://en.wikipedia.org/wiki/Booth%27s_multiplication_algorithm
2. https://www.geeksforgeeks.org/non-restoring-division-unsigned-integer/
3. Digital Fundamentals by Thomas L. Floyd, Chapter 2