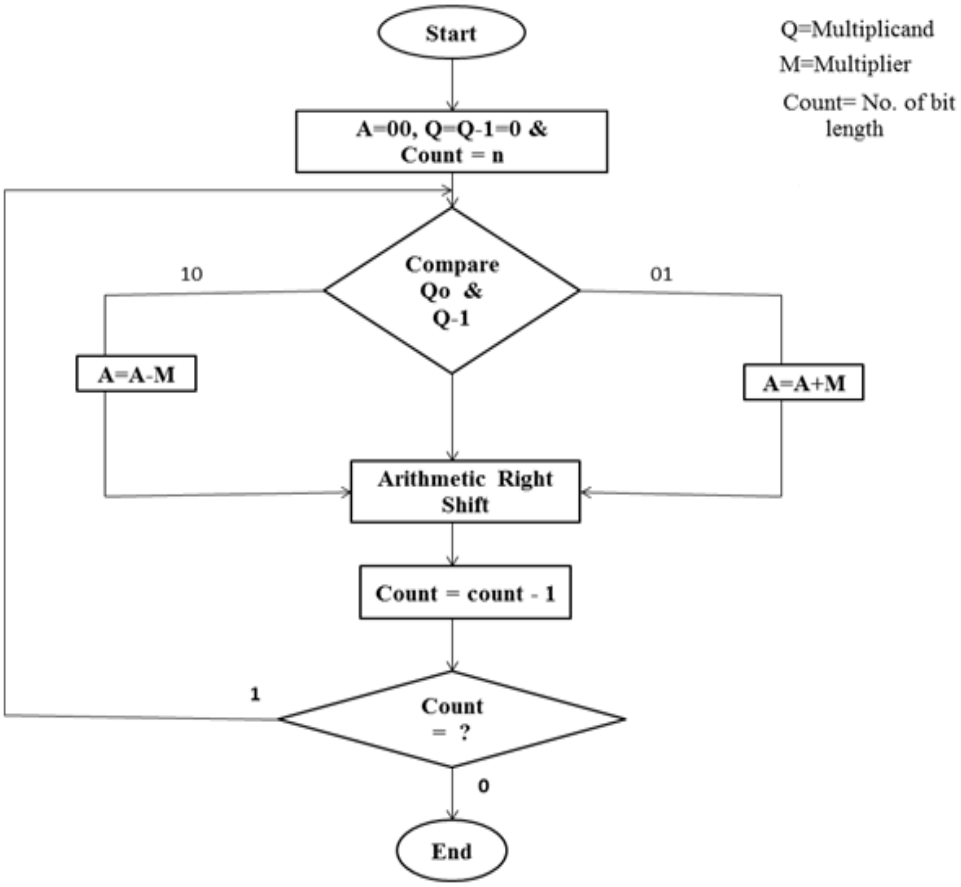
# Experiment 2

## (Booth's multiplication)

**Aim:** Implement Booth's multiplication algorithm.

**Theory:**

- Booth algorithms gives a procedure for multiplying binary integers in signed 2's complement representation in efficient way, i.e., a smaller number of additions/subtractions required. It operates on the fact that strings of 0's in the multiplier require no addition but just shifting and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as 2^(k+1 ) to 2^m.
- As in all multiplication schemes, booth algorithms require 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:
- The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier
- 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.
- The partial product does not change when the multiplier bit is identical to the previous multiplier bit.
- **Example –** A numerical example of booth's algorithm is shown below for n = 4. It shows the step by step multiplication of 7 and 5.



**Perform 7*5 using Booth's Algorithm**

| A | Q | Q-1 | M | | |
|---|---|---|---|---|---|
| 0000 | 0101 | 0 | 0111 | Initial value | |
| 1001 | 0101 | 0 | 0111 | A ← A-M | First cycle |
| 1100 | 1010 | 1 | 0111 | shift | |
| 0011 | 1010 | 1 | 0111 | A ← A+M | Second cycle |
| 0001 | 1101 | 0 | 0111 | shift | |
| 1010 | 1101 | 0 | 0111 | A ← A-M | Third cycle |
| 1101 | 0110 | 1 | 0111 | shift | |
| 0100 | 0110 | 1 | 0111 | A ← A+M | Fourth cycle |
| 0010 | 0011 | 0 | 0111 | shift | |

**Lab Assignments to complete in this session**

1. Perform binary multiplication of –7 and –3 using booths algorithm and register size=4 bits
2. Perform binary multiplication of –9 and 7 using booths algorithm and register size=5 bits
3. Perform binary multiplication of –13 and –6 using booths algorithm and register size=5 bits
4. Perform binary multiplication of –13 and –6 using booths algorithm and register size=4 bits.