BOOTH'S MULTIPLICATION ALGORITHM

📓 Booth's Multiplication Algorithm — Tabular Method

Booth's Algorithm is used for multiplying **two signed binary numbers** using **2's complement**. It minimizes the number of additions and subtractions required in the process.

Key Concepts

- **Multiplicand (M)** and **Multiplier (Q)** are treated in 2's complement.
- An additional bit **Q-1** is used to track the previous LSB of the multiplier.
- **Registers Used**:
  - **A** (Accumulator)
  - **Q** (Multiplier)
  - **Q-1** (1-bit register)
  - **M** (Multiplicand)
- The result is in the form of **AQ** (Concatenation of A and Q).

General Steps

1. **Initialize**:
   - Set A = 0
   - Q = Multiplier
   - M = Multiplicand
   - Q-1 = 0
   - Count = number of bits
2. **Check Q_0 Q-1**:
   - 10 → A = A – M
   - 01 → A = A + M
   - 00 or 11 → Do nothing
3. **Arithmetic Right Shift** A, Q, and Q-1 together
4. **Repeat** for the number of bits

PTO

Example 1: (-79) × 13

Binary Representation (8-bit)

Number	Decimal	Binary
M (-79)	-79	10110001
Q (13)	+13	00001101

Booth's Table for -79 × 13

Step	A	Q	Q-1	Operation	Remarks
Initialization	00000000	00001101	0	-	-
1	00000000	00001101	0	01 → A = A + M	A = 10110001 (Add -79)
	11011000	00000110	1	Arithmetic Shift	
2	11011000	00000110	1	00 → No op	
	11101100	00000011	0	Arithmetic Shift	
3	11101100	00000011	0	10 → A = A - M	A = 11101100 - 10110001
	00111011	10000001	1	Arithmetic Shift	
4	00111011	10000001	1	00 → No op	
	00011101	11000000	0	Arithmetic Shift	
5	00011101	11000000	0	10 → A = A - M	A = 00011101 - 10110001
	01101100	11100000	1	Arithmetic Shift	
6	01101100	11100000	1	00 → No op	
	00110110	01110000	0	Arithmetic Shift	
7	00110110	01110000	0	10 → A = A - M	A = 00110110 - 10110001
	10000101	00111000	1	Arithmetic Shift	
8	10000101	00111000	1	00 → No op	Final

Final Result (A + Q) = 1000010100111000 = **-1027** **(Decimal)**

✓ **Check**: -79 × 13 = **-1027**

PTO

Example 2: (-25) × (-9)

Binary Representation (8-bit)

Number	Decimal	Binary
M (-25)	-25	11100111
Q (-9)	-9	11110111

Booth's Table for (-25) × (-9)

Step	A	Q	Q-1	Operation	Remarks
Initialization	00000000	11110111	0	-	-
1	00000000	11110111	0	01 → A = A + M	A = 11100111
	11110011	11111011	1	Arithmetic Shift	
2	11110011	11111011	1	00 → No op	
	11111001	11111101	0	Arithmetic Shift	
3	11111001	11111101	0	10 → A = A - M	A = 11111001 - 11100111
	00010010	11111110	1	Arithmetic Shift	
4	00010010	11111110	1	00 → No op	
	00001001	01111111	0	Arithmetic Shift	
5	00001001	01111111	0	10 → A = A - M	A = 00001001 - 11100111
	00100010	00111111	1	Arithmetic Shift	
6	00100010	00111111	1	00 → No op	
	00010001	00011111	0	Arithmetic Shift	
7	00010001	00011111	0	10 → A = A - M	A = 00010001 - 11100111
	00101010	00001111	1	Arithmetic Shift	
8	00101010	00001111	1	00 → No op	Final

Final Result (A + Q) = 0010101000001111 = **225** **(Decimal)**

✓ **Check**: (-25) × (-9) = **+225**