COMPUTER ARITHMETIC

Slides prepared by:
Dr. Zubair Ahmad Shah
Department of Computer Science and Engineering
Islamic University of Science and Technology, Awantipora

DATA REPRESENTATION OF FIXED-POINT BINARY DATA

- When an integer binary number is positive, the sign is represented by 0 and the magnitude by a positive binary number.

  e.g. +5 = 0 0000101

- When an integer binary number is negative, it is represented in one of the following three possible ways:

  I. Signed-magnitude Representation

  II. Signed-1’s Complement Representation

  III. Signed-2’s Complement Representation

Data Representation cont.

I. **Signed-magnitude Representation:**

-5 = Complement sign bit of +5

= 1 0000101

II. **Signed-1's Complement Representation:**

-5 = Complement all bits of +5 including sign bit

= 1 1111010

III. **Signed-2's Complement Representation:**

-5 = 2's complement of +5 including the sign bit

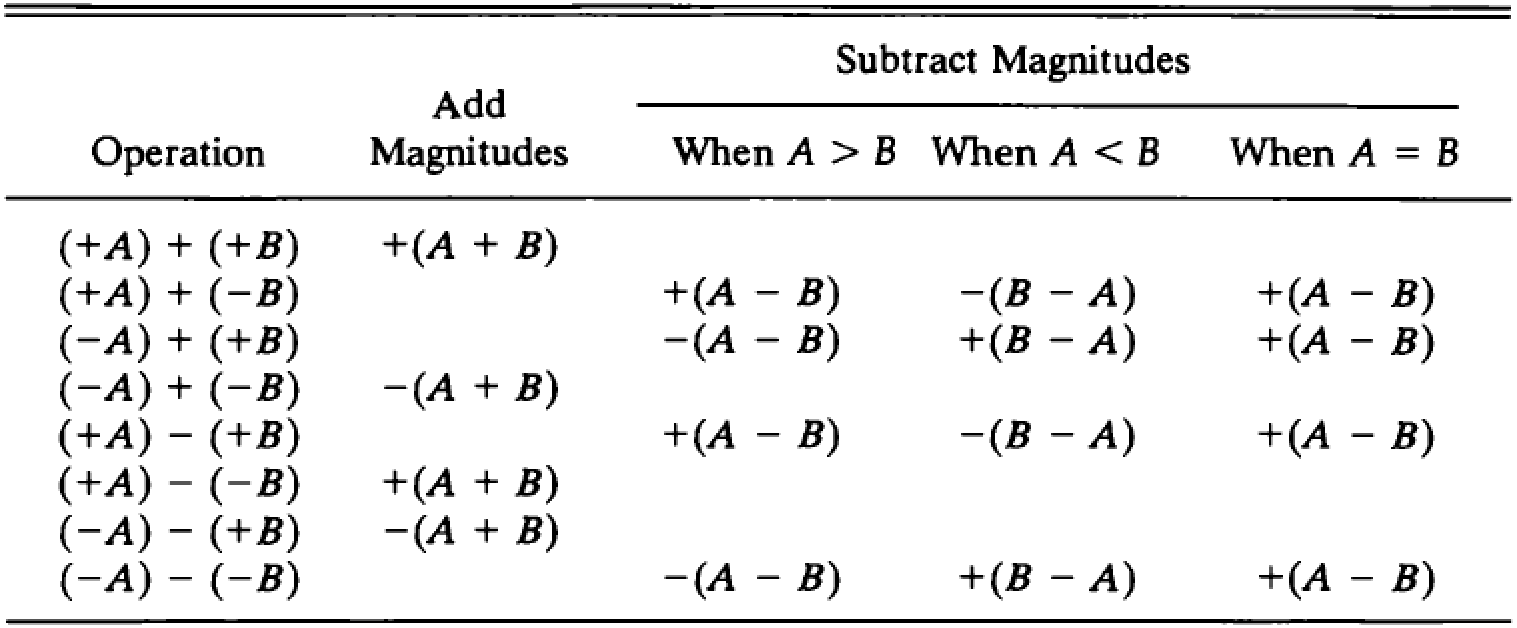
= 1 1111011

[+5 = 0 0000101]

ADDITION AND SUBTRACTION OF FIXED-POINT BINARY NUMBERS

Addition and Subtraction of Signed-Magnitude Numbers:

Eight different conditions to consider:



		Subtract Magnitudes		
Operation	Add Magnitudes	When $A > B$	When $A < B$	When $A = B$
$(+A) + (+B)$	$+(A + B)$			
$(+A) + (-B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(-A) + (+B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$
$(-A) + (-B)$	$-(A + B)$			
$(+A) - (+B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(+A) - (-B)$	$+(A + B)$			
$(-A) - (+B)$	$-(A + B)$			
$(-A) - (-B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$

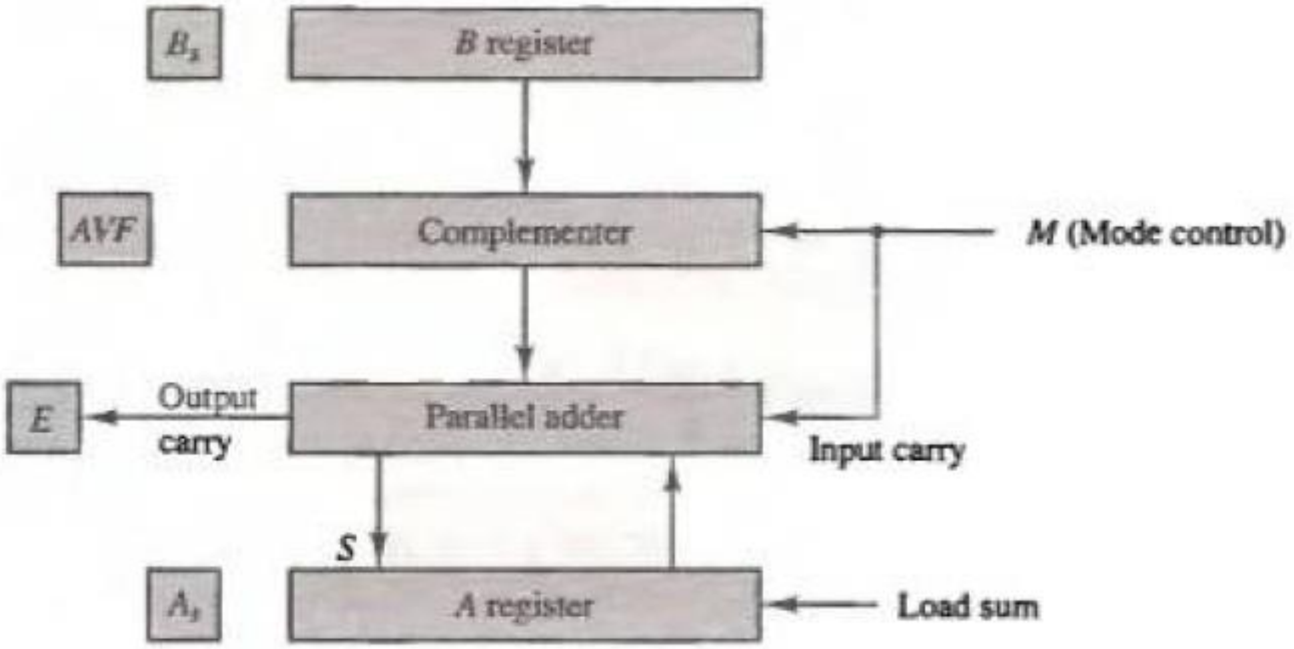
A and B are magnitudes of the two numbers

Addition and Subtraction cont.

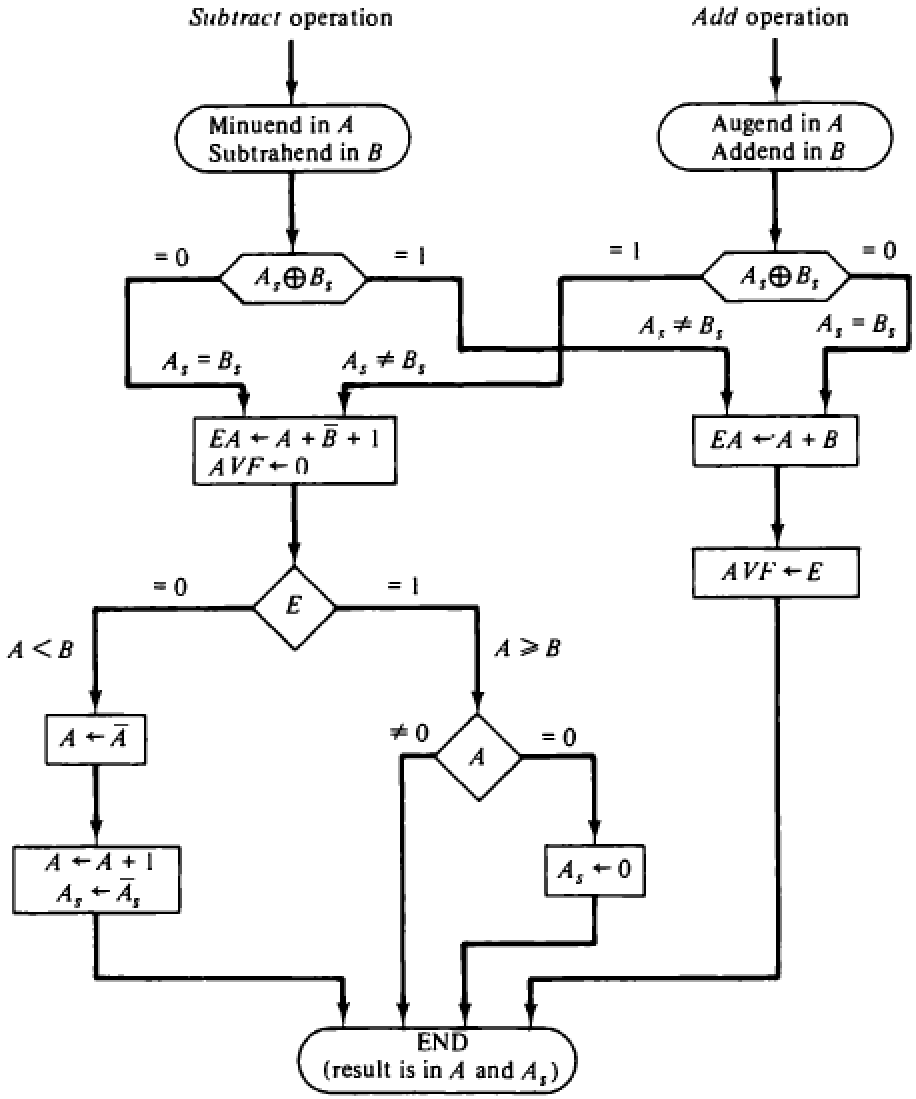
- Hardware Required:
  - 2 registers (for A and B)
  - 3 flip-flops (one for sign bit of A, second for sign bit of B and third for output carry say E)
  - 1 adder
  - 1 comparator
  - 2 subtractors

Addition and Subtraction cont.

- A different procedure can be found that requires less hardware:



Hardware for signed-magnitude addition and subtraction.



Flowchart for add and subtract operations.

Addition and Subtraction cont.

- **Example 1:**

  X = +5 and Y = -3, perform operation X+Y

  Magnitude of X (A) = 5 = 0000101

  Magnitude of Y (B) = 3 = 0000011

  Sign bit of X (A_s) = 0

  Sign bit of Y (B_s) = 1

  EA = A + B' + 1 = 0000101 + 1111100 + 1 = **1**0000010

  => E = 1 and A = 0000010

  Result = A_sA = 0 0000010 = +2

Addition and Subtraction cont.

- **Example 2:**

  X = +3 and Y = -5, perform operation X+Y

  Magnitude of X (A) = 3 = 0000011

  Magnitude of Y (B) = 5 = 0000101

  Sign bit of X (A_s) = 0

  Sign bit of Y (B_s) = 1

  EA = A + B’ + 1 = 0000011 + 1111010 + 1 = 1111110 (no carry)

  => E = 0 and A = 1111110

  A = A’ = 0000001

  A = A + 1 = 0000010, $A_s = A_s' = 1$

  Result = A_sA = 1 0000010 = -2

Addition and Subtraction cont.

- **Example 3:**

  X = -3 and Y = +5, perform operation X-Y

  Magnitude of X (A) = 3 = 0000011

  Magnitude of Y (B) = 5 = 0000101

  Sign bit of X (A_s) = 1

  Sign bit of Y (B_s) = 0

  EA = A + B = 0000011 + 0000101 = 0001000 (no carry)

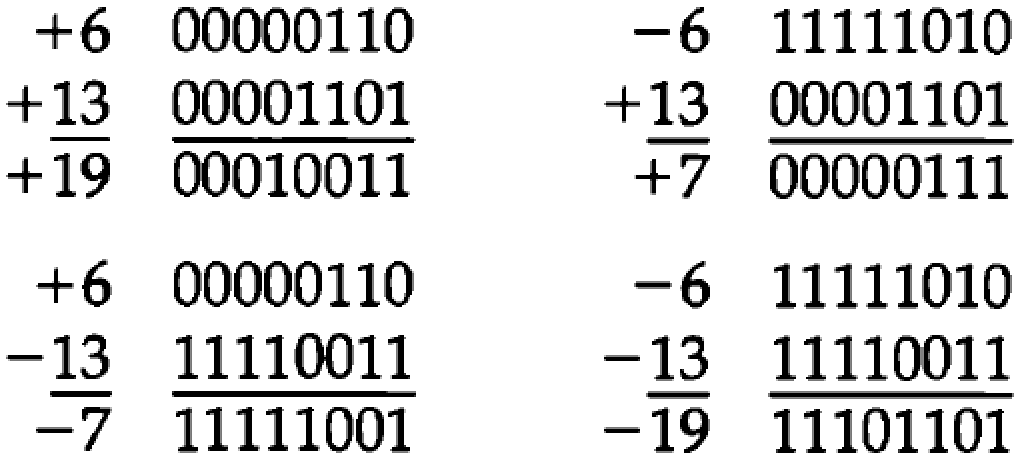
  => E = 0 and A = 0001000

  Result = A_sA = 1 0001000 = -8

Addition and Subtraction cont.

Addition and Subtraction of Signed Numbers (with negative numbers represented in 2’s complement):

- **Addition:**

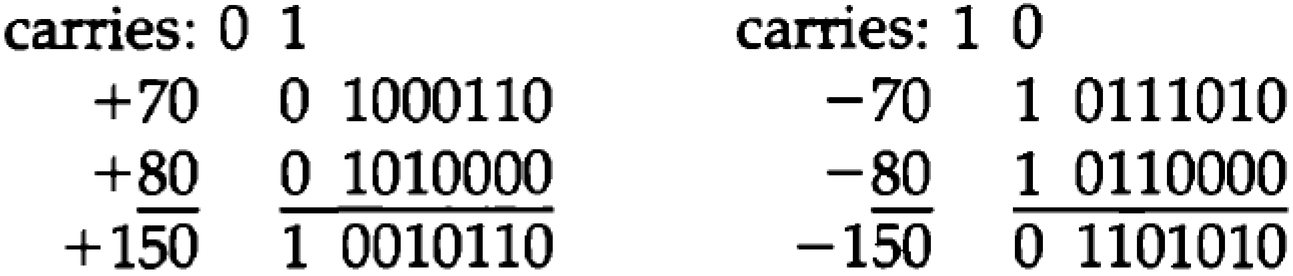


+6	00000110	−6	11111010
+13	00001101	+13	00001101
+19	00010011	+7	00000111
+6	00000110	−6	11111010
−13	11110011	−13	11110011
−7	11111001	−19	11101101

Addition and Subtraction cont.

- **Note:**
  - **An overflow cannot occur after an addition if one number is positive and the other is negative.**
  - **An overflow may occur after an addition if both numbers are positive or both are negative.**
    - **In such cases, overflow is detected by XORing the *carry into the sign bit position* and the *carry out of the sign bit position.* If 1 overflow has occurred (result invalid), if 0 no overflow has occurred (result valid).**

e.g.



```
carries: 0  1                      carries: 1  0
   +70     0  1000110                 −70     1  0111010
   +80     0  1010000                 −80     1  0110000
  +150     1  0010110                −150     0  1101010
```

Addition and Subtraction cont.

– **Subtraction:**

(+A) – (+B) = (+A) + (–B)
(+A) – (–B) = (+A) + (+B)
(–A) – (+B) = (–A) + (–B)
(–A) – (–B) = (–A) + (+B)

e.g.

(–6) – (–13) = 11111010 – 11110011
= 11111010 + 00001101
= 1 00000111
= 00000111 (Discarding end carry)
= +7

Addition and Subtraction cont.

(+6) – (+13) = 00000110 – 00001101

= 00000110 + 11110011

= 11111001

= -7 (in 2’s complement)

(+6) – (–13) = 00000110 – 11110011

= 00000110 + 00001101

= 00010011

= + 19

(– 6) – (+13) = 11111010 – 00001101

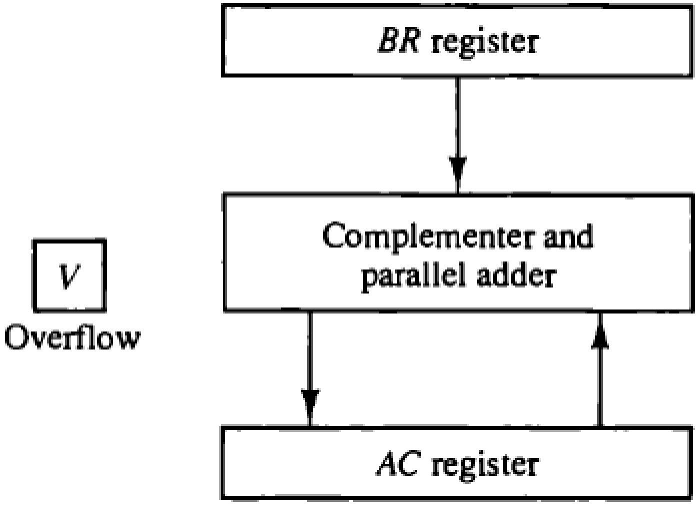
= 11111010 + 11110011

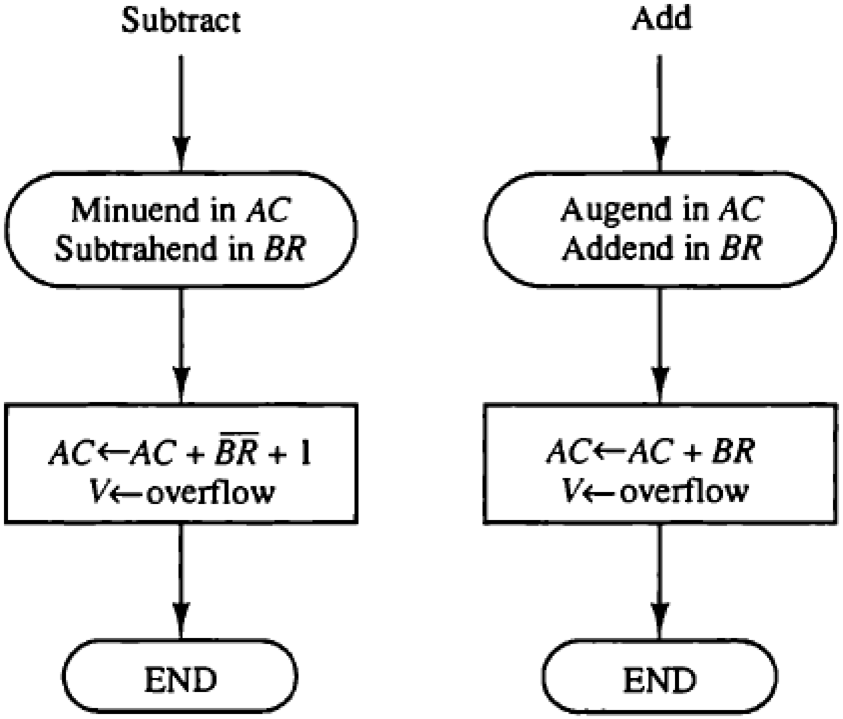
= 1 11101101

= 11101101 (Discarding end carry)

= – 19 (in 2’s complement)

Hardware for signed-2's complement addition and subtraction.

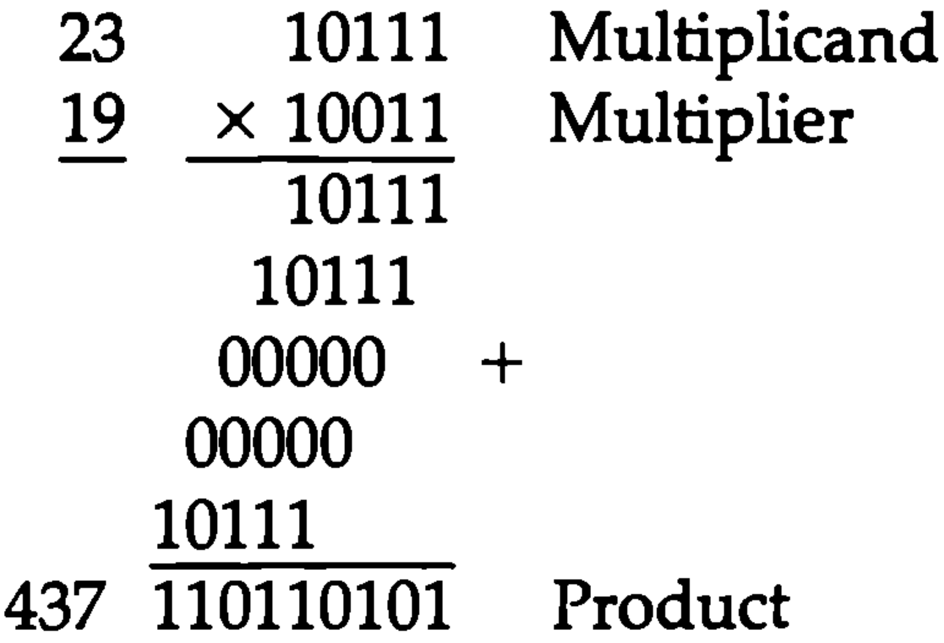




Algorithm for adding and subtracting numbers in signed-2's complement representation.

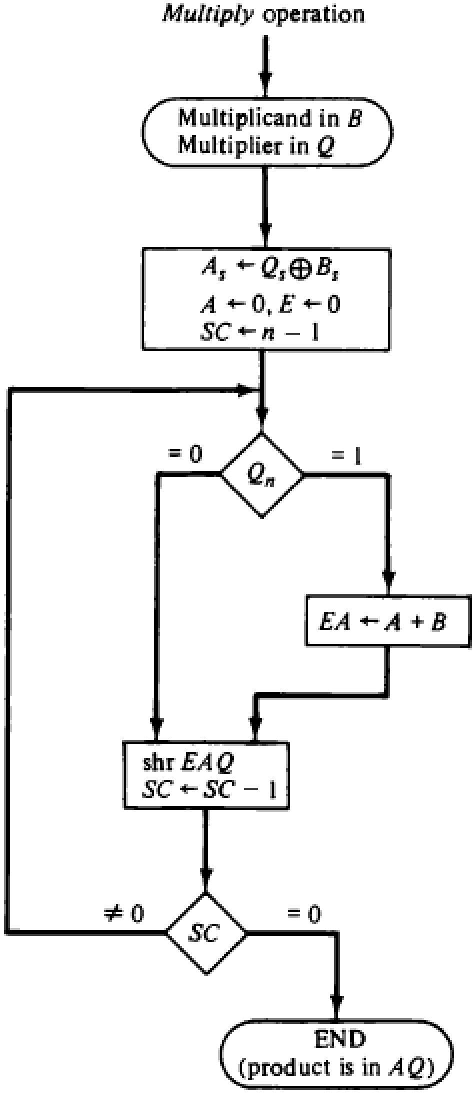
MULTIPLICATION OF FIXED-POINT BINARY NUMBERS

- **Multiplication of Signed-Magnitude Numbers:**

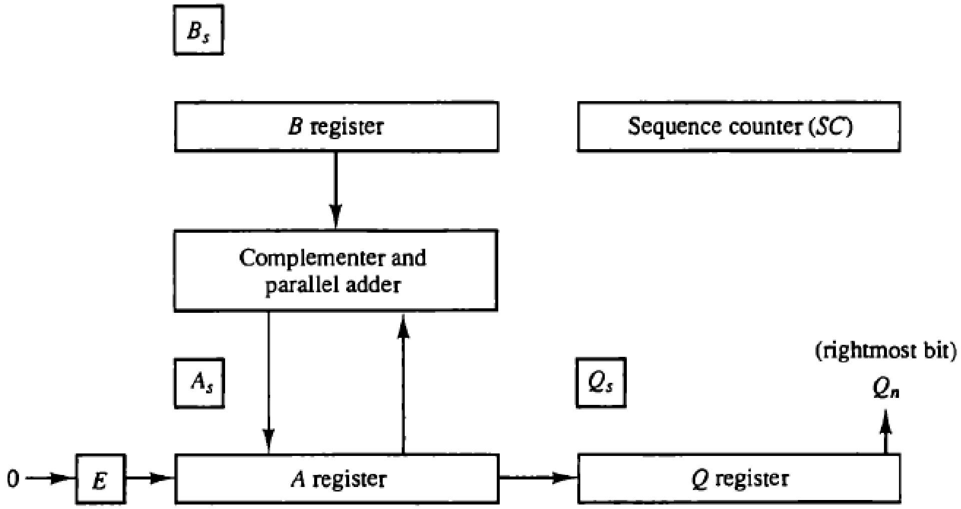


```
 23        10111    Multiplicand
 19      × 10011    Multiplier
 --     --------
           10111
          10111
         00000     +
        00000
       10111
       ---------
437    110110101    Product
```

Flowchart for multiply operation.



Hardware for multiply operation.

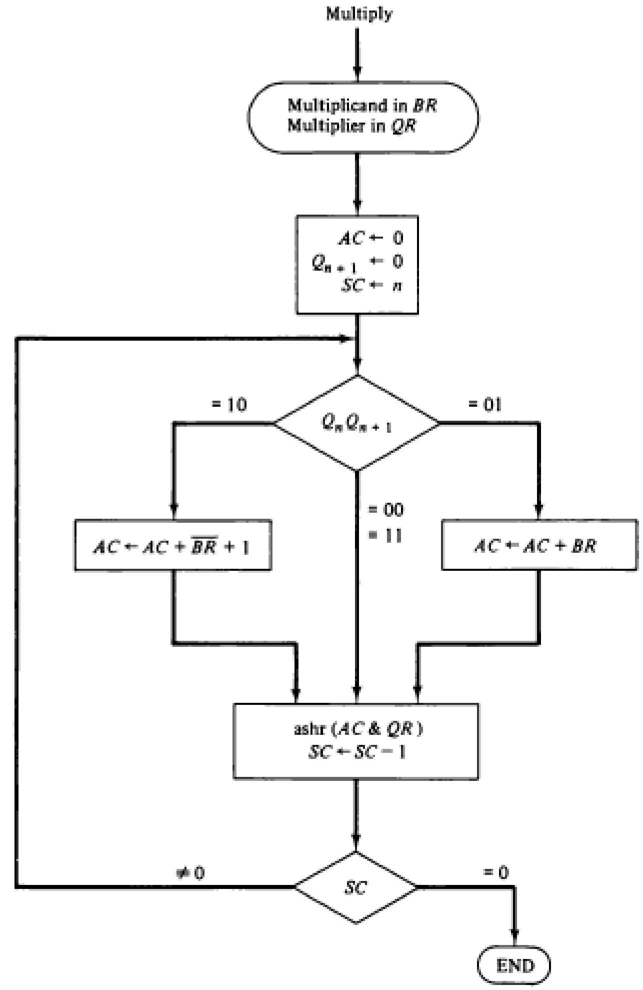


Numerical Example for Binary Multiplier

Multiplicand B = 10111	E	A	Q	SC
Multiplier in Q	0	00000	10011	101
Q_n = 1; add B		10111		
First partial product	0	10111		
Shift right EAQ	0	01011	11001	100
Q_n = 1; add B		10111		
Second partial product	1	00010		
Shift right EAQ	0	10001	01100	011
Q_n = 0; shift right EAQ	0	01000	10110	010
Q_n = 0; shift right EAQ	0	00100	01011	001
Q_n = 1; add B		10111		
Fifth partial product	0	11011		
Shift right EAQ	0	01101	10101	000
Final product in AQ = 0110110101				

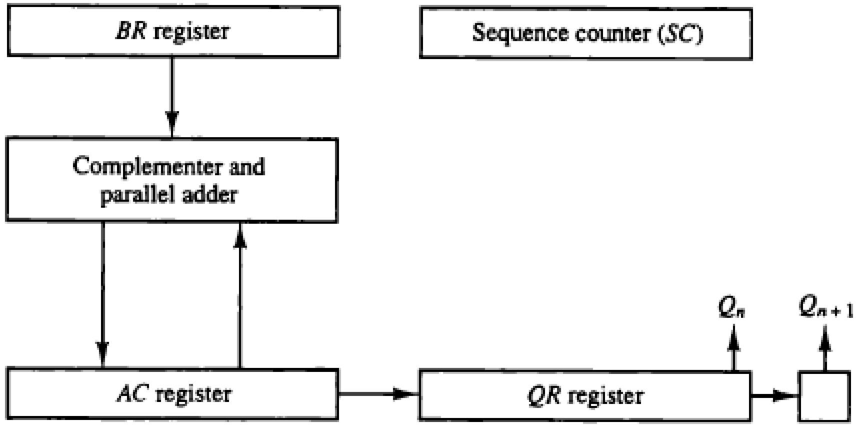
Multiplication Continued

- **Multiplication of Signed-2’s Complement Numbers:**
  - **Booth Multiplication Algorithm**



Booth algorithm for multiplication of signed-2's complement numbers.

Hardware for Booth algorithm.



Example of Multiplication with Booth Algorithm **(-9) x (-13)**

$Q_n Q_{n+1}$	$BR = 10111$ $\overline{BR} + 1 = 01001$	AC	QR	Q_{n+1}	SC
	Initial	00000	10011	0	101
1 0	Subtract *BR*	01001			
		01001			
	ashr	00100	11001	1	100
1 1	ashr	00010	01100	1	011
0 1	Add *BR*	10111			
		11001			
	ashr	11100	10110	0	010
0 0	ashr	11110	01011	0	001
1 0	Subtract *BR*	01001			
		00111			
	ashr	00011	10101	1	000

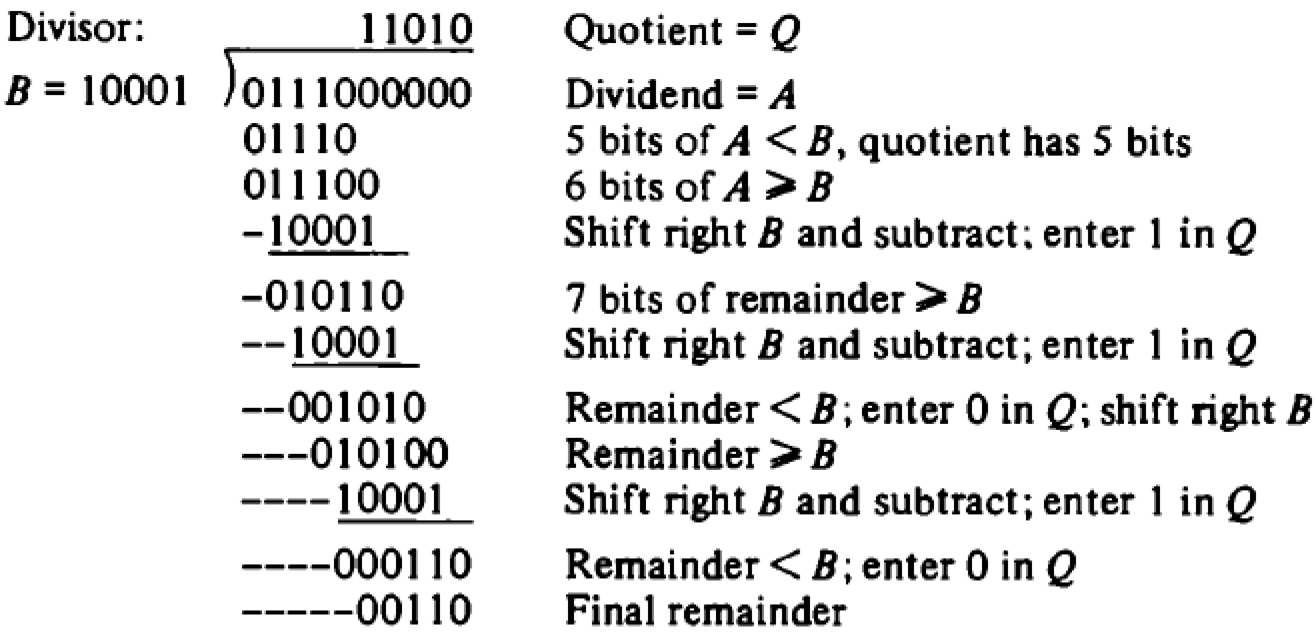
-9 in 2's complement = 10111
-13 in 2's complement = 10011
Result in AC & QR = 0001110101 = +117

DIVISION OF FIXED-POINT BINARY NUMBERS

Division of Signed-Magnitude Numbers:

Example:

Divisor = +17 = 0 10001
Dividend = +448 = 0 0111000000



```
Divisor:             11010        Quotient = Q
B = 10001      )0111000000        Dividend = A
                01110             5 bits of A < B, quotient has 5 bits
                011100            6 bits of A ≥ B
                -10001            Shift right B and subtract; enter 1 in Q
                -010110           7 bits of remainder ≥ B
                --10001           Shift right B and subtract; enter 1 in Q
                --001010          Remainder < B; enter 0 in Q; shift right B
                ---010100         Remainder ≥ B
                ----10001         Shift right B and subtract; enter 1 in Q
                ----000110        Remainder < B; enter 0 in Q
                -----00110        Final remainder
```

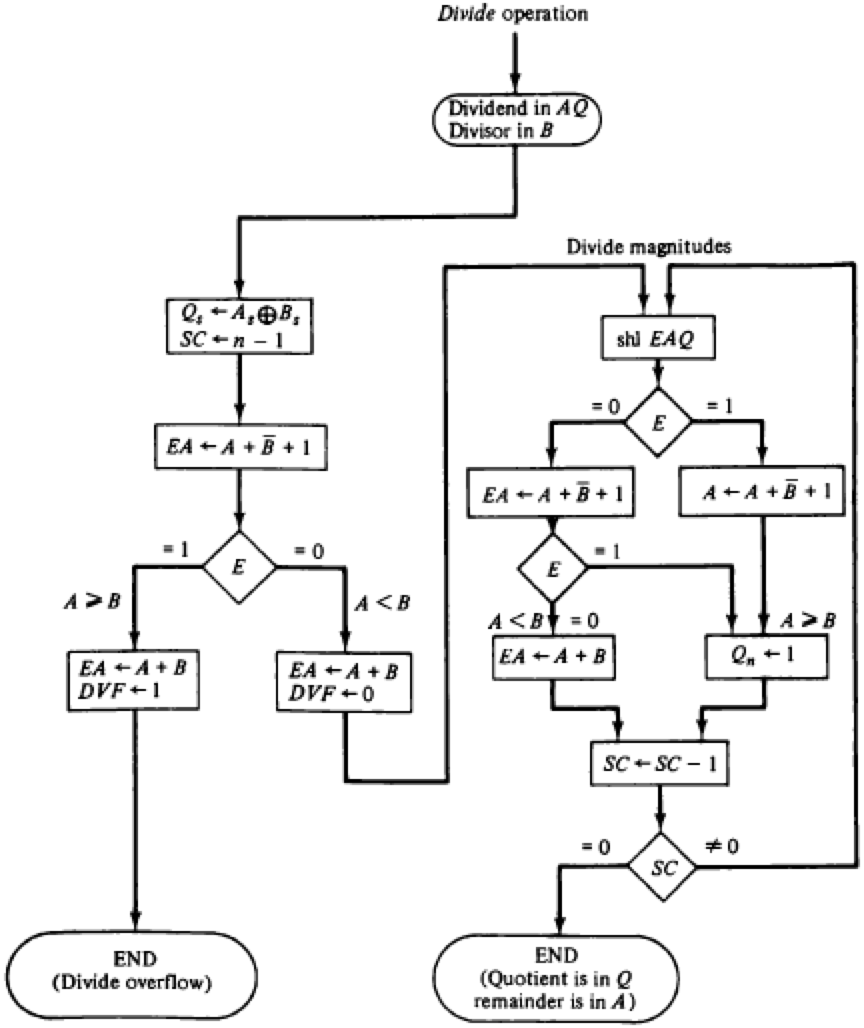
Quotient = 11010 = 26
Remainder = 00110 = 6

Divisor $B = 10001$, $\overline{B} + 1 = 01111$

	E	A	Q	SC
Dividend:		01110	00000	5
shl EAQ	0	11100	00000	
add $\overline{B} + 1$		01111		
$E = 1$	1	01011		
Set $Q_n = 1$	1	01011	00001	4
shl EAQ	0	10110	00010	
Add $\overline{B} + 1$		01111		
$E = 1$	1	00101		
Set $Q_n = 1$	1	00101	00011	3
shl EAQ	0	01010	00110	
Add $\overline{B} + 1$		01111		
$E = 0$; leave $Q_n = 0$	0	11001	00110	
Add B		10001		
Restore remainder	1	01010		2
shl EAQ	0	10100	01100	
Add $\overline{B} + 1$		01111		
$E = 1$	1	00011		
Set $Q_n = 1$	1	00011	01101	1
shl EAQ	0	00110	11010	
Add $\overline{B} + 1$		01111		
$E = 0$; leave $Q_n = 0$	0	10101	11010	
Add B		10001		
Restore remainder	1	00110	11010	0
Neglect E				
Remainder in A:		00110		
Quotient in Q:			11010	

Example of binary division

Flowchart for divide operation.



REFERENCE

Morris Mano, Computer System Architecture, Prentice-Hall of India.