

**William Stallings**  
**Computer Organization**  
**and Architecture**  
**6<sup>th</sup> Edition**

---

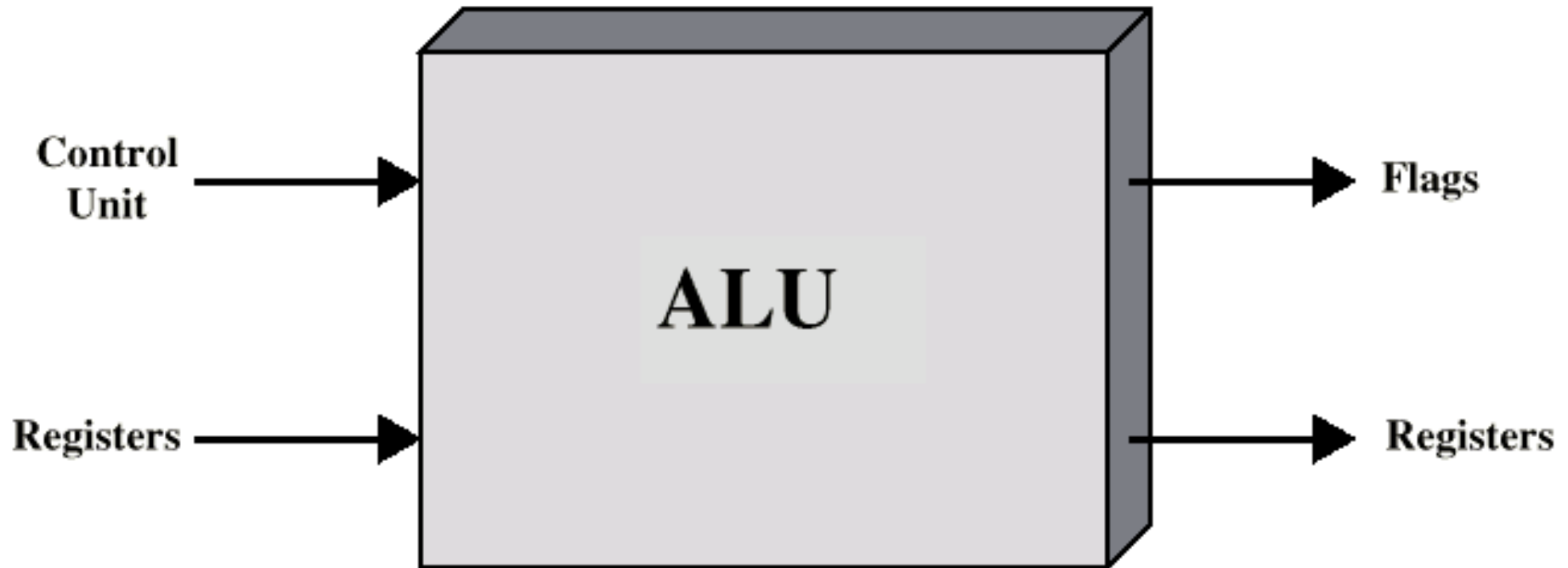
**Chapter 9**  
**Computer Arithmetic**

# Arithmetic & Logic Unit

---

- Does the calculations
- Everything else in the computer is there to service this unit
- Handles integers
- May handle floating point (real) numbers
- May be separate FPU (maths co-processor)

# **ALU Inputs and Outputs**



# Addition and Subtraction

---

- Normal binary addition
- Monitor sign bit for overflow
- Take two's complement of subtrahend and add to minuend
  - i.e.  $a - b = a + (-b)$
- So we only need addition and complement circuits

# Example of 2's Complement

**Example 1: Finding the 2's complement of 5**

5 = 00000101

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

11111010

+1

-----

-5 = 11111011

**Complement Digits**

**Add 1**

0 1 1 0 1 1 1 0 ← Original binary value

1 0 0 1 0 0 0 1 ← 1's complement

1 0 0 1 0 0 0 1
+ 1
1 0 0 1 0 0 1 0

2's complement

**Example 2: Finding the 2's complement of -13**

-13 = 11110011

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

00001100

+1

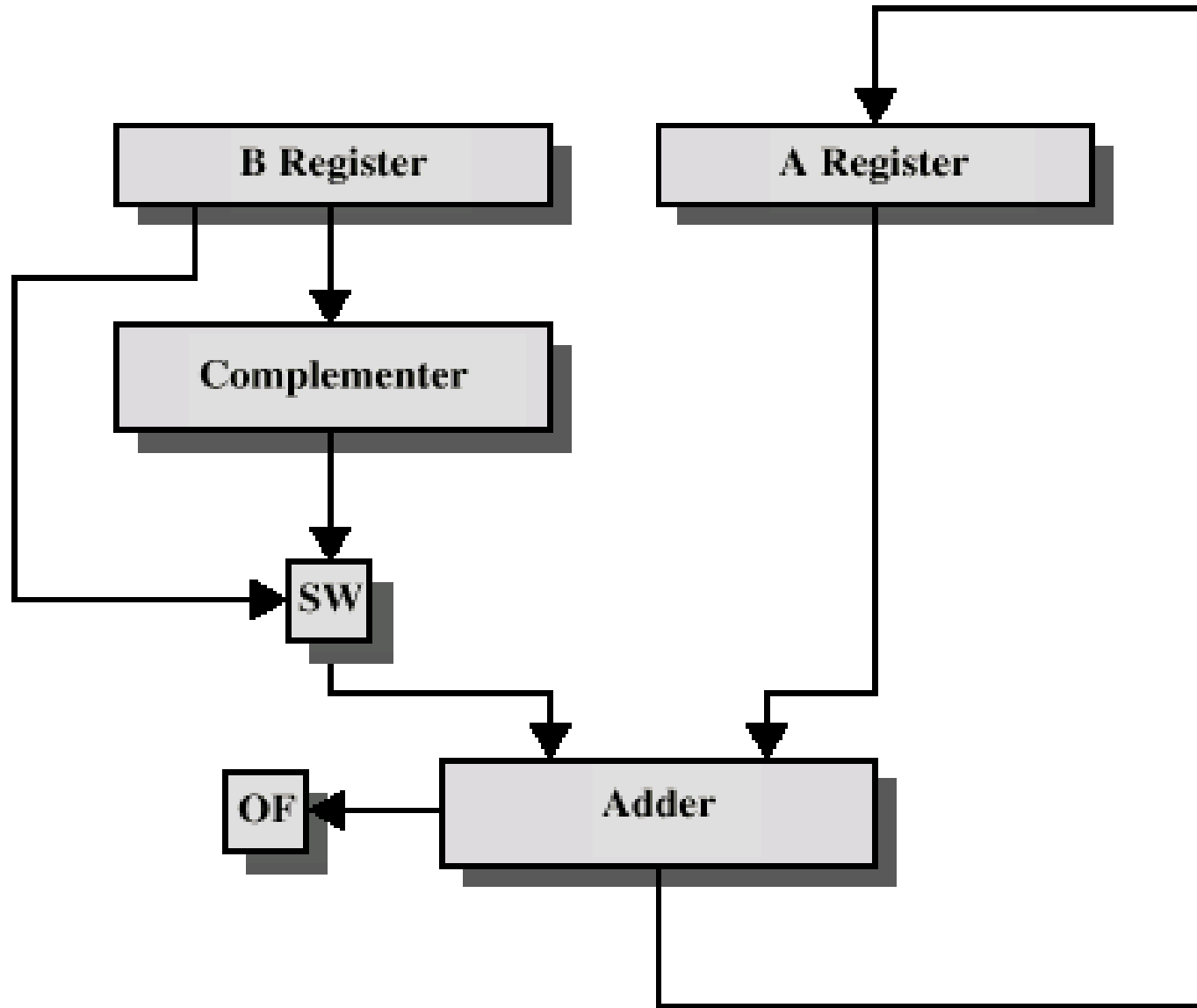
-----

13 = 00001101

**Complement Digits**

**Add 1**

# Hardware for Addition and Subtraction

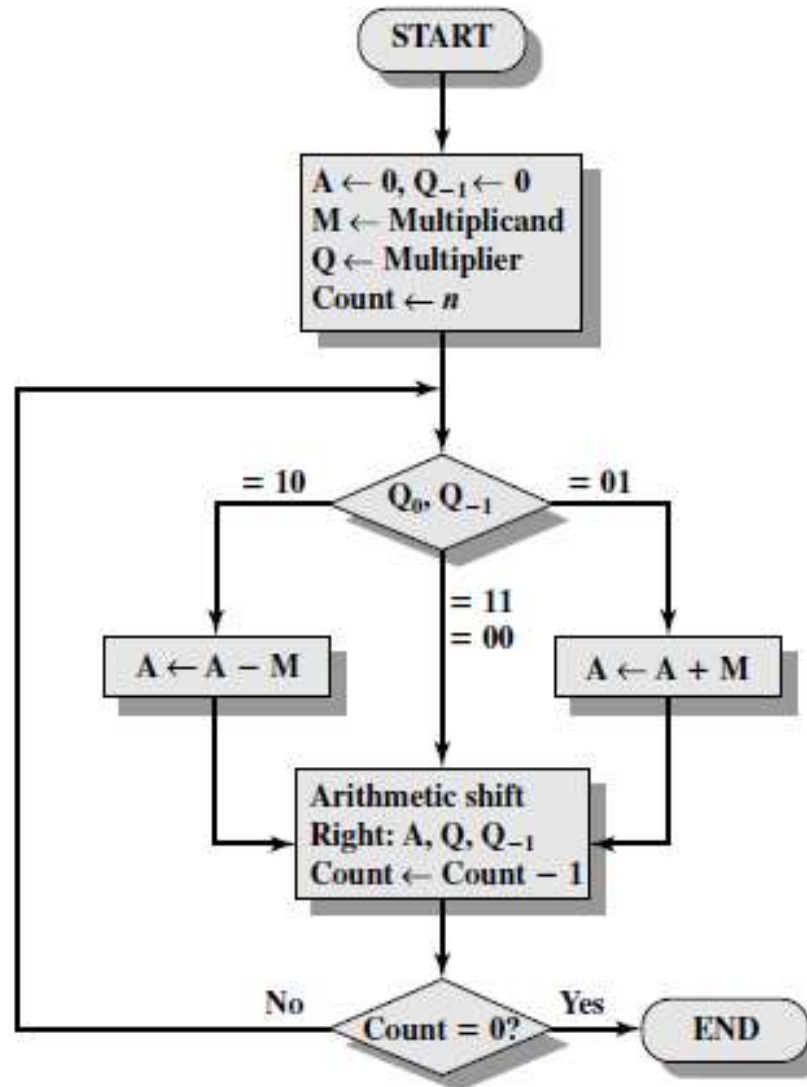


OF = overflow bit

SW = Switch (select addition or subtraction)

# Booth's Algorithm

---



---

Q0	Q-1	Result
0	0	Only shift
1	1	
0	1	A=A + M ,then shift
1	0	A= A – M , then shift

M =7

Q =3

M = 0 1 1 1

Q = 0 0 1 1

- M = 1 0 0 1



$$\begin{matrix} M & Q \\ (7) & * (3) \end{matrix}$$

$$\begin{aligned} M &\Rightarrow 0111 \\ -M &\Rightarrow 1001 \\ Q &\Rightarrow 0011 \end{aligned}$$

A	Q	Q-1	n
0000	0011	0	4
1001	0011	0	
1100	1001	1	3
1110	0100	1	2
0101	0100	1	
0010	1010	0	1
0001	0101	0	0

$$AQ \Rightarrow 0001 \ 0101 \Rightarrow 21$$

$$\begin{array}{r} 1110 \\ 0111 \\ \hline 1101 \\ \boxed{1} \end{array}$$

## Example of Booth's Algorithm: 7(M) \* 3(Q)

A	Q	Q <sub>-1</sub>	M	Initial Values	
0000	0011	0	0111		
1001	0011	0	0111	A = A - M	} First Cycle
1100	1001	1	0111		
				Shift	
1110	0100	1	0111	Shift	} Second Cycle
0101	0100	1	0111	A = A + M	} Third Cycle
0010	1010	0	0111		
				Shift	
0001	0101	0	0111	Shift	} Fourth Cycle

**Answer is in A and Q → 0001 0101 = 21**

A	Q	Q <sub>-1</sub>	M		
0000	0011	0	0111	Initial values	
1001	0011	0	0111	A ← A − M } Shift	First cycle
1100	1001	1	0111		
1110	0100	1	0111	Shift	} Second cycle
0101	0100	1	0111	A ← A + M }	
0010	1010	0	0111	Shift	} Third cycle
0001	0101	0	0111	Shift	
					} Fourth cycle

Figure 9.13 Example of Booth's Algorithm ( $7 \times 3$ )

A	Q	Q-1	n
00000	00111	0	5
01001	00111	0	
*00100	10011	1	4

00010 01001 1 3

00001 00100 1 2

11000 00100 1

11100 00010 0 1

11110 00001 0 0

11110 00001 ← In 2's complement  
for  
00001 011110  
1

00001 11111  
2<sup>5</sup> 2<sup>4</sup> 2<sup>3</sup> 2<sup>2</sup> 2<sup>1</sup> 2<sup>0</sup>

$$\Rightarrow 32 + 16 + 8 + 4 + 2 + 1$$

$$\Rightarrow -63$$

00001  
11111  
11000

## **Examples-size of n determines answer**

Solve using Booths Algorithm

A.  $M = 5$  ,  $Q = 5$

B.  $M = 12$  ,  $Q = 11$

C.  $M = 9$  ,  $Q = -3$

D.  $M = -13$  ,  $Q = 6$

E.  $M = -15$  ,  $Q = 15$

F.  $M = -19$  ,  $Q = -20$

G.  $M = -7$  ,  $Q = 3$

H.  $M = 15$  ,  $Q = -6$

I.  $M = -12$  ,  $Q = -18$

J.  $M = -7$  ,  $Q = 14$

# Booths Recoding / Bit pair recording

---

## Booths Recoding / Bit pair recording

- Derived from the Booth's algorithm.
  - Reduced number of steps
  - 3 steps
1. Calculate the table of M
  2. Solve for reduced value of Q  
for  $01 = +1$   
 $10 = -1$   
 $00/11 = 0$
  3. Perform  $M * Q$

# Bit-pair recoding of multiplier (A fast multiplication method)

① Table of M:-

$m \quad q$   
 $5 \times 4$   
 $0101 \quad 0100$

Operation	Value
0	0000 0000
+1 (m)	0000 ← 0101
-1 (-m)	1111 ← 1011
+2 (left shift m)	0000 1010
-2	1111 0110



②

Value of  $Q$  :-

$$\begin{array}{cccccc}
 & & & & & Q-1 \\
 & & & & & 0 \\
 & 0 & 1 & 0 & 0 & 0 \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
 & +1 & -1 & 0 & 0 & \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \\
 & 2(1) & -1 & 2(0) & +0 & \\
 & & & & & 0
 \end{array}$$



③

m \* q :-

0 1 0 |  
1 0

---

0 0 0 0 0 0 0 0 0

0 0 0 1 0 1 + + +

---

0 0 0 1 0 1 0 0

$2^4$

$2^2$

$\Rightarrow 16 + 4 \Rightarrow 20$

$$(15) * (-10)$$

Receding

$$M \Rightarrow 15 \Rightarrow 01111$$

$$-M \Rightarrow 10001$$

$$Q \Rightarrow 01010 \Rightarrow 10101$$

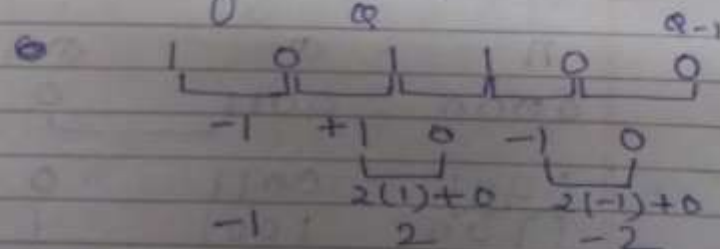
$$+$$

$$10110$$

Step 1: Table of M :-

Operation	value
0	00000 000000
+1 (+M)	00000 01111
-1 (-M)	11111 10001
+2 (shift)	00000 11110
-2 (shift)	11111 00010

Step 2: Solve for Q :-



Step 3: M \* Q :-

$$01111$$

$$-12-2$$

$$1111100010$$

$$0001111011$$

$$1000011111$$

$$11111$$

$$[1] 0001101010 \quad (2's \text{ complement})$$

# Booths Recoding / Bit pair recoding

---

- Solve Booth's Recoding algorithm

1.  $M = 9$  ,  $Q = -6$  (take 5 bits)

2. Implement multiplication of the following pair of signed 2's complement numbers using bit-pair recoding.

$M = 110011$

$Q = 101100$

i.e. numbers are  $-13 * -20$

③

$-13 * -20$

$\Rightarrow 101100$

① Table of m:  
operation

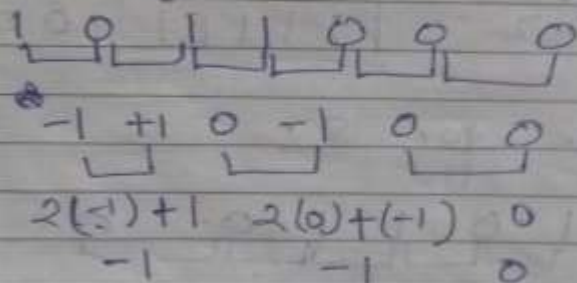
value

0	000000 000000
+1 (m)	111111 110011
-1 (-m)	000000 001101
+2	111111 100110
-2	000000 011010

②

Solve for Q:-

Q-1



③

m \* Q :-

$$\begin{array}{r}
 110011 \\
 -1-10 \\
 \hline
 000000 \quad 000000 \\
 000000 \quad 1101++ \\
 000011 \quad 01++++ \\
 \hline
 000100 \quad 000100 \\
 \hline
 \end{array}$$

2<sup>8</sup> 2<sup>7</sup> 2<sup>6</sup> 2<sup>5</sup> 2<sup>4</sup> 2<sup>3</sup> 2<sup>2</sup> 2<sup>1</sup> 2<sup>0</sup>

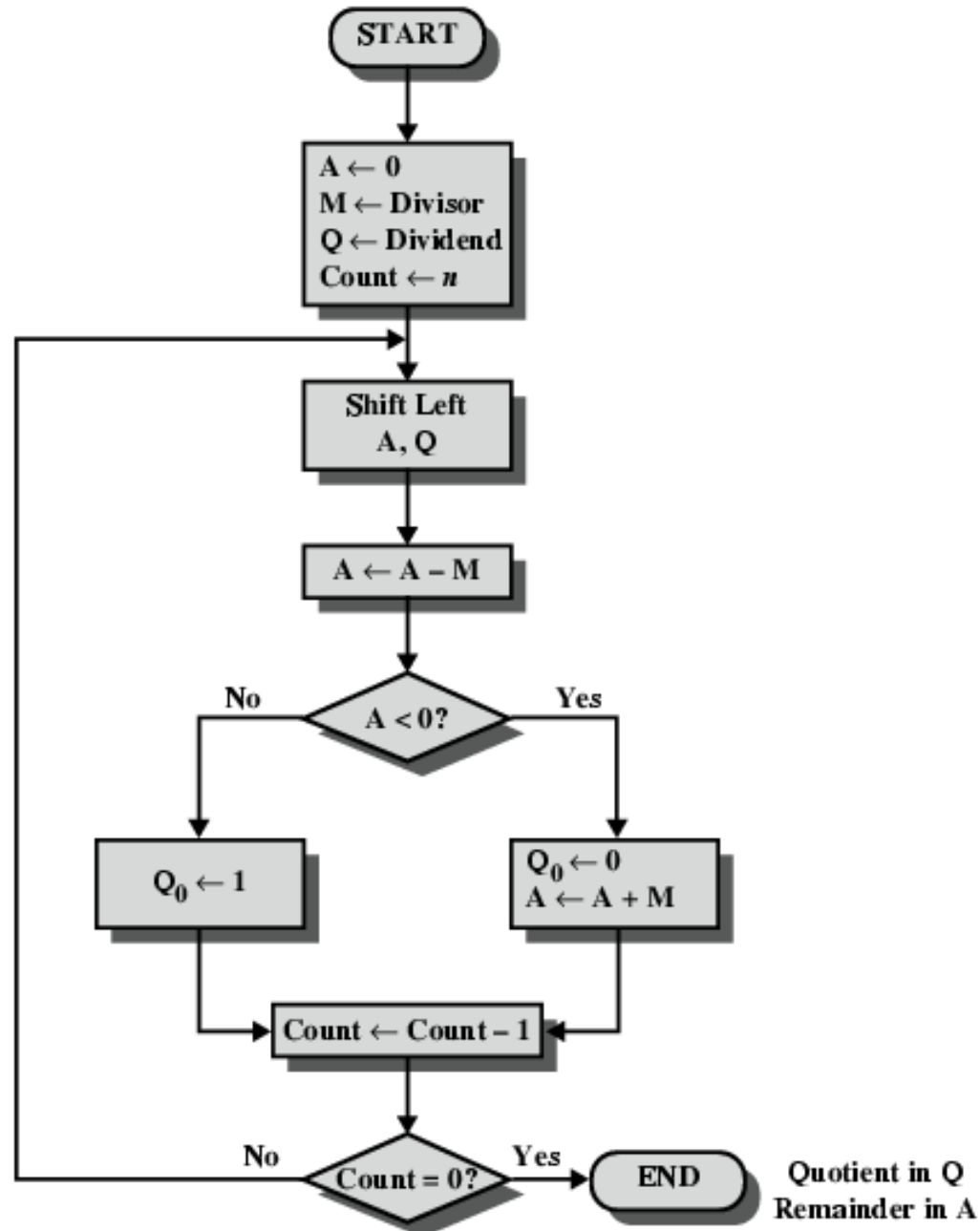
$\Rightarrow 256 + 4 \Rightarrow \underline{\underline{260}}$

# Division

---

- More complex than multiplication
- Negative numbers are really bad!
- Based on long division

# Flowchart for Restoring Division





$$M \leftarrow 3\sqrt{7} \rightarrow Q$$

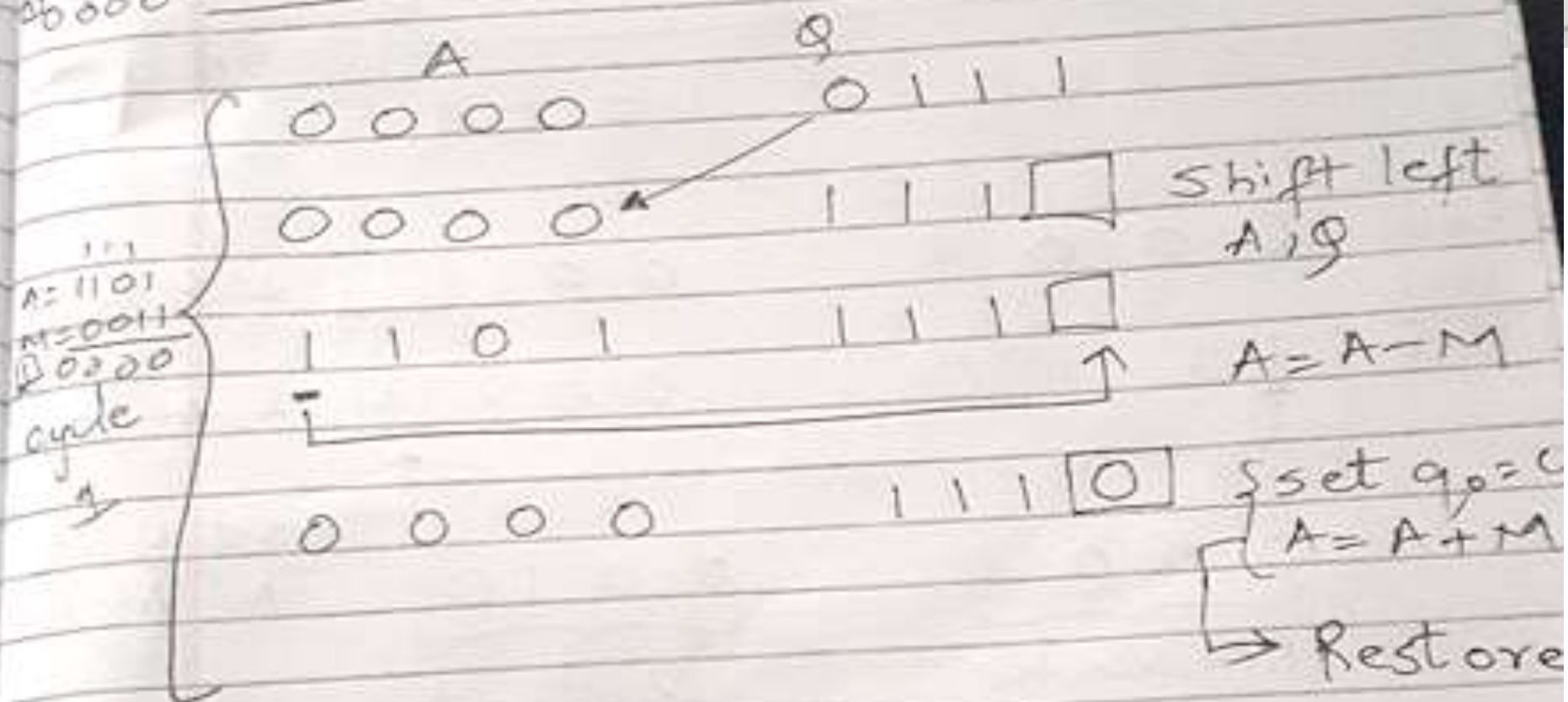
$$M = 0011$$

$$- M = 1101$$

$$Q = 0111$$

2	7		2	3	
2	3	1		1	1
	1	1			0
					0

0000



A		Q	
0001	0001	110	□
1101			
<hr/>			
1110	1110	110	□
1110			
0011	0001	1100	
<hr/>			
0001			

Restore (dashed arrow from 1110 to 0001)  
 Shift left  
 $A = A - M$   
 set  $Q_0 = 0$   
 $A = A + M$



0001

$$\begin{array}{r} 1111 \\ 0011 \\ \hline 1101 \\ 0000 \end{array}$$

0011

100   

shift left

0000

100 1

$A = A - M$

set  $q_0 = 1$

0000

1001

0001

0001

001   

shift left

1101

$$\begin{array}{r} 1101 \\ \hline 1110 \end{array}$$

1110

001 0

$A = A - M$

set  $q_0 = 0$

$A = A + M$

0001

Remainder

= 1

0010

Quotient

= 2

$$M = 27; \quad Q = 55$$

$$M = 011011$$

$$-M = 100101$$

$$Q = 110111$$

A

Q

000000

11

110111

000001

11

10111 ☐

shift left A

100110

11

10111 ☐

A = A - M

000001

11100

101110

Set  $Q_0 = 0$ ; A = A + M

de 1

cycle 2	000011	01110	□	shift left A, 9
	101000	01110	□	A = A - M
				set q <sub>0</sub> = 0
cycle 3	000011	011100		A = A + M
	000110	11100	□	shift left A, 9
	101011	11100	□	A = A - M
cycle 4	0000110	111000		set q <sub>0</sub> = 0; A = A + M
	001101	11000	□	shift left A, 9
	110010	11000	□	A = A - M

cycle 4	001101 110010	11000 <input type="checkbox"/> shift left A, q 11000 <input checked="" type="checkbox"/> $A = A - M$
	001101	110000 set $q_0 = 0, A = A +$
cycle 5	011011 000000	10000 <input type="checkbox"/> shift left A, q 10000 <input checked="" type="checkbox"/> set $q_0 = 1$
	000001	00001 <input type="checkbox"/> shift left A, q
cycle 6	100110	00001 <input checked="" type="checkbox"/> $A = A - M$
	000001	Set $q_0 = 0$
	$R = 1$	000010 $A = A + M$
		$Q = 2$



$$M = 27; \quad q = 55$$

$$M = 011011$$

$$-M = 100101$$

$$q = 110111$$

	A	Q	
cycle 1	000000	110111	
	000001	10111□	shift left A, q
	100110	10111□	A = A - M
cycle 2	000001	101110	Set $q_0 = 0$ ; A = A + M
	000011	01110□	shift left A, q
	101000	01110□	A = A - M
cycle 3	000011		set $q_0 = 0$
	000110	011100	A = A + M
	101011	11100□	shift left A, q
cycle 4	000110	11100□	A = A - M
	001101	111000	Set $q_0 = 0$ ; A = A + M
	110010	11000□	shift left A, q
cycle 5	001101	11000□	A = A - M
	011011	110000	set $q_0 = 0$ ; A = A + M
	000000	10000□	shift left A, q
cycle 6	000001	10000□	set $q_0 = 1$
	100110	00001□	shift left A, q
		00001□	A = A - M
	000001		Set $q_0 = 0$
		000010	A = A + M
	R = 1	Q = 2	

## **Solve using Restoring Division**

---

A.  $M = 5$  ,  $Q = 5$

B.  $M = 12$  ,  $Q = 26$

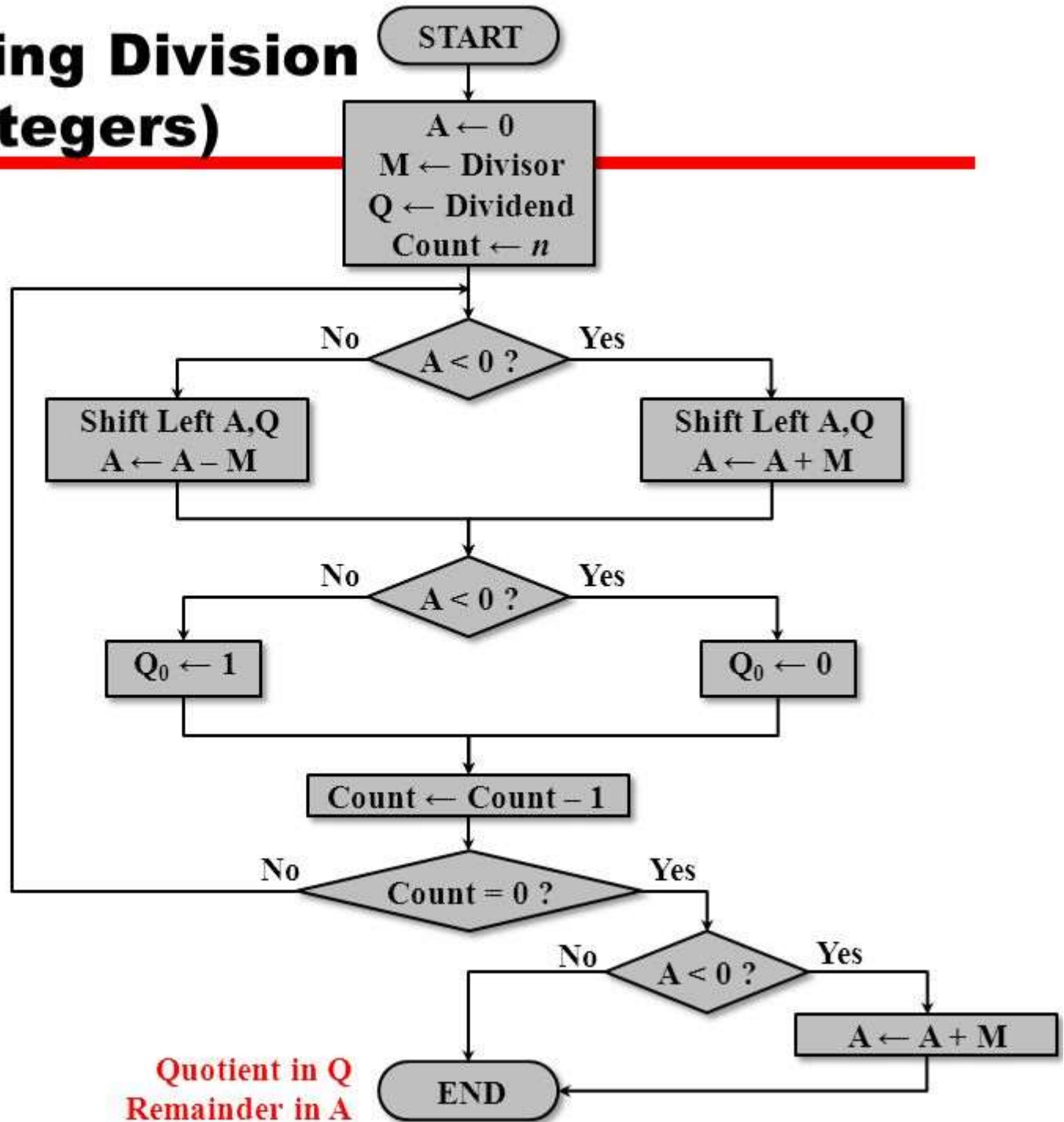
C.  $M = 9$  ,  $Q = 19$

D.  $M = 32$  ,  $Q = 59$

E.  $M = 27$  ,  $Q = 55$

F.  $M = 17$  ,  $Q = 42$

# Non-Restoring Division (Positive Integers)



$$M = 2;$$

$$q = 4$$

$$M = 0010$$

$$-M = 1110$$

$$q = 0100$$

A

0000

Shift left A, q

0000

A = A - M

set  $q_0 = 0$

1110

q

0100

1000

1000

①

Shift left A, q

1101

A = A + M

set  $q_0 = 0$

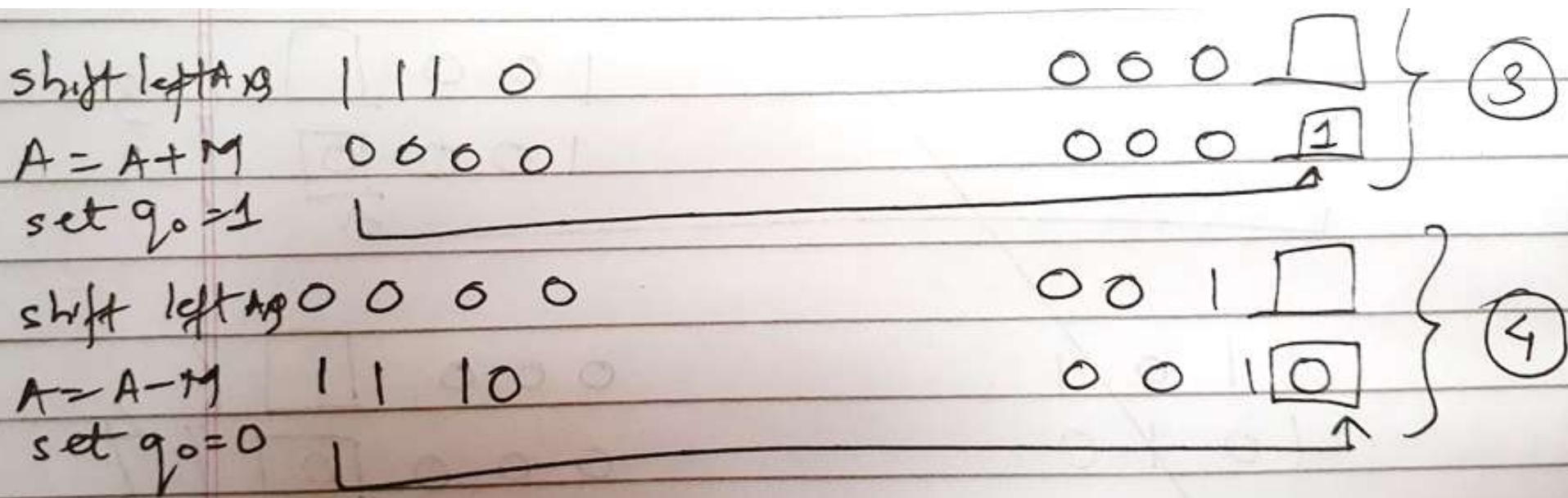
1111

0000

0000

②





count = 0

$A = A + M$

1 1	
1 1 1 0	(A)
x 0 0 1 0	(M)
0 0 0 0	

0 0 0 0

~~~~~

A = Remainder

0 0 1 0

~~~~~

q = Quotient

## **Solve using Non Restoring**

---

A.  $M = 5$  ,  $Q = 5$

B.  $M = 12$  ,  $Q = 26$

C.  $M = 9$  ,  $Q = 19$

D.  $M = 32$  ,  $Q = 59$

E.  $M = 17$  ,  $Q = 42$

# Division of signed numbers

1. Load the divisor into the M register and the dividend into the A, Q registers. The dividend must be expressed as a  $2n$ -bit two's complement number. Thus, for example, the 4-bit 0111 becomes 00000111, and 1001 becomes 11111001.
2. Shift A, Q left 1 bit position.
3. If M and A have the same signs, perform  $A \leftarrow A - M$ ; otherwise,  $A \leftarrow A + M$ .
4. The preceding operation is successful if the sign of A is the same before and after the operation.
  - a. If the operation is successful or  $A = 0$ , then set  $Q_0 \leftarrow 1$ .
  - b. If the operation is unsuccessful and  $A \neq 0$ , then set  $Q_0 \leftarrow 0$  and restore the previous value of A.
5. Repeat steps 2 through 4 as many times as there are bit positions in Q.
6. The remainder is in A. If the signs of the divisor and dividend were the same, then the quotient is in Q; otherwise, the correct quotient is the two's complement of Q.

The reader will note from Figure 9.17 that  $(-7) \div (3)$  and  $(7) \div (-3)$  produce different remainders. This is because the remainder is defined by

$$D = Q \times V + R$$

where

$D$  = dividend

$Q$  = quotient

$V$  = divisor

$R$  = remainder

The results of Figure 9.17 are consistent with this formula.

A	Q	M = 0011	A	Q	M = 1101
0000	0111	Initial value	0000	0111	Initial value
0000	1110	shift	0000	1110	shift
1101		subtract	1101		add
0000	1110	restore	0000	1110	restore
0001	1100	shift	0001	1100	shift
1110		subtract	1110		add
0001	1100	restore	0001	1100	restore
0011	1000	shift	0011	1000	shift
0000		subtract	0000		add
0000	1001	set $Q_0 = 1$	0000	1001	set $Q_0 = 1$
0001	0010	shift	0001	0010	shift
1110		subtract	1110		add
0001	0010	restore	0001	0010	restore

(a) (7)/(3)

(b) (7)/(-3)



A	Q	M = 0011	A	Q	M = 1101
1111	1001	Initial value	1111	1001	Initial value
1111	0010	shift	1111	0010	shift
0010		add	0010		subtract
1111	0010	restore	1111	0010	restore
1110	0100	shift	1110	0100	shift
0001		add	0001		subtract
1110	0100	restore	1110	0100	restore
1100	1000	shift	1100	1000	shift
1111		add	1111		subtract
1111	1001	set $Q_0 = 1$	1111	1001	set $Q_0 = 1$
1111	0010	shift	1111	0010	shift
0010		add	0010		subtract
1111	0010	restore	1111	0010	restore

(c)  $(-7)/(3)$

(d)  $(-7)/(-3)$

# Floating Point

---

- IEEE Standard 754 floating point is the **most common representation today for real numbers on computers**, including Intel-based PC's, Macs, and most Unix platforms.
- IEEE, stands for the **Institute of Electrical and Electronics Engineers**.

# Floating Point

---

Sign bit	Biased Exponent	Significand or Mantissa
----------	-----------------	-------------------------

- $\pm \text{.significand} \times 2^{\text{exponent}}$
- Point is actually fixed between sign bit and body of mantissa
- Exponent indicates place value (point position)
- E.g.  $2.5 \times 2^5$

# Floating Point Examples

---



(a) Format



# Signs for Floating Point

---

- Mantissa is stored in 2s compliment
- Exponent is in excess or biased notation
  - e.g. Excess (bias) 128 means
  - 8 bit exponent field
  - Pure value range 0-255
  - Subtract 128 to get correct value
  - Range -128 to +127

# IEEE 754

---

- Standard for floating point storage
- 32 and 64 bit standards
- 8 and 11 bit exponent respectively
- Extended formats (both mantissa and exponent) for intermediate results

# IEEE 754 Formats

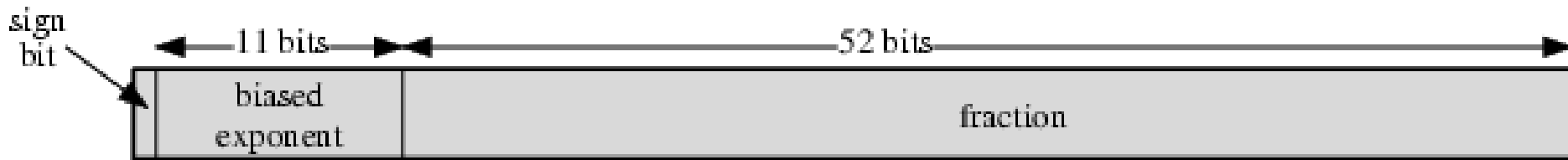
---



(a) Single format

32 BIT

$$(1.N)2^{E-127}$$



(b) Double format

64 BIT

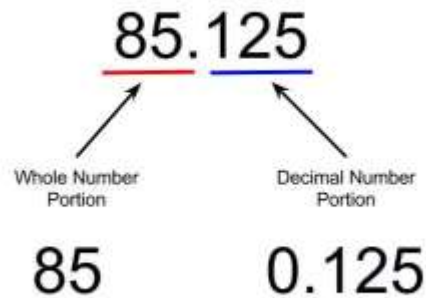
$$(1.N)2^{E-1023}$$

# Steps

---

- 1. Convert Decimal to Binary
- 2. Normalization
  - Rewriting Step 1 into (1.N) form
  - Ex:  $111.011 = \mathbf{1}.11011 \times 2^{\mathbf{2}}$
  - Ex:  $0.00010 = 0000\mathbf{1}.0 \times 2^{-\mathbf{4}}$
- 3. Biasing
  - Applying Single Precision (E – 127) & Double Precision (E – 1023) on exponent from Step 2
- 4. Representation in Single (32 bit )and Double Precision (64 bit ) Format

Exponent



## Example

Convert 639.6875 to single precision

$$\begin{aligned} 639.6875 &= 100111111.1011_2 \\ &= 1.001111111011 \times 2^9 \end{aligned}$$

$$s=0$$

$$\text{exp} - 127 = 9 \quad \text{exp} = 136 = 10001000_2$$

$$\text{fra} = 001111111011$$

• Final result:

$$010001000001111110110000000000$$

# Solved Example

Eg 12.25

Step 1: Converting Dec to Bin

2	12	
2	6	0
2	3	0
	1	1
		1

$$\begin{array}{r} .25 \\ \times 2 \\ \hline 0.50 \\ \times 2 \\ \hline 1.00 \Rightarrow \text{stop} \end{array}$$

$$\begin{array}{r} 12.25 \\ \hline 1100.01 \end{array}$$

Step 2: Normalization (1. N)

$$1.10001 \times 2^3 \rightarrow \text{Exponent}$$

Step 3: Biasing

Single Precision      Double precision

$$E - 127$$

$$E - 1023$$

$$3 = E - 127$$

$$3 = E - 1023$$

$$E = 127 + 3$$

$$E = 1023 + 3$$

$$= 130$$

$$= 1026$$

2	13	0		2	1026	
2	65	0		2	513	0
2	32	1		2	256	1
2	16	0		2	128	0
2	8	0		2	64	0
2	4	0		2	32	0
2	2	0		2	16	0
1	1	0		2	8	0
		1		2	4	0
				2	2	0
					1	0



## Single Precision (32 bits)

Sign bit	Biased Exponent	Mantissa/Significand
0	10000010	10001
1 bit	8 bits	23 bits

## Double Precision (64 bits)

Sign bit	Biased Exponent	Mantissa/Significand
0	100000000010	10001
1 bit	11 bits	52 bits

# Solve

---

25.44

178.1875 Single precision: 0 10000110 01100100011

Double precision: 0 10000000110 01100100011

-309.1875 Single precision: 1 10000111 001101010011

Double precision: 1 10000000111 001101010011

0.00635

-125.10

13.54 Single precision: 0 10000100 10110001010001111010111

Double precision: 0 10000000011 10110001010001111010111

# Solve 0.00635

Step 1:

$$0.00635$$

$$\begin{array}{r} 0.00635 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 0.01270 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 0.02540 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 0.05080 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 0.10160 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 0.20320 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 0.40640 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 0.81280 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 1.62560 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 3.25120 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 6.50240 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 13.00480 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 26.00960 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 52.01920 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 104.03840 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 208.07680 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 416.15360 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 832.30720 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 1664.61440 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 3329.22880 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 6658.45760 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 13316.91520 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 26633.83040 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 53267.66080 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 106535.32160 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 213070.64320 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 426141.28640 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 852282.57280 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 1704565.14560 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 3409130.29120 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 6818260.58240 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 13636521.16480 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 27273042.32960 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 54546084.65920 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 109092169.31840 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 218184338.63680 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 436368677.27360 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 872737354.54720 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 1745474709.09440 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 3490949418.18880 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 6981898836.37760 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 13963797672.75520 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 27927595345.51040 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 55855190691.02080 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 111710381382.04160 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 223420762764.08320 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 446841525528.16640 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 893683051056.33280 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 1787366102112.66560 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 3574732204225.33120 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 7149464408450.66240 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 14298928816901.32480 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 28597857633802.64960 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 57195715267605.29920 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 114391430535210.59840 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 228782861070421.19680 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 457565722140842.39360 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 915131444281684.78720 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 1830262888563369.57440 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 3660525777126739.14880 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 7321051554253478.29760 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 14642103108506956.59520 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 29284206217013913.19040 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 58568412434027826.38080 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 117136824868055652.76160 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 234273649736111305.52320 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 468547299472222611.04640 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 937094598944445222.09280 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 1874189197888890444.18560 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 3748378395777780888.37120 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 7496756791555561776.74240 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 14993513583111123553.48480 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 29987027166222247106.96960 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 59974054332444494213.93920 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 119948108664888988427.87840 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 239896217329777976855.75680 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 479792434659555953711.51360 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 959584869319111907423.02720 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 1919169738638223814846.05440 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 3838339477276447629692.10880 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 7676678954552895259384.21760 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 15353357909105790518768.43520 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 30706715818211581037536.87040 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 61413431636423162075073.74080 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 122826863272846324150147.48160 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 245653726545692648300294.96320 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 491307453091385296600589.92640 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 982614906182770593201179.85280 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 1965229812365541186402359.70560 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 3930459624731082372804719.41120 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 7860919249462164745609438.82240 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 15721838498924329491218877.64480 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 31443676997848658982437755.28960 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 62887353995697317964875510.57920 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 125774707991394635929751021.15840 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 251549415982789271859502042.31680 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 503098831965578543719004084.63360 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 1006197663931157087438008169.26720 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 2012395327862314174876016338.53440 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 4024790655724628349752032677.06880 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 8049581311449256699504065354.13760 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 16099162622898513399008130708.27520 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 32198325245797026798016261416.55040 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 64396650491594053596032522833.10080 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 128793300983188107192065045666.20160 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 257586601966376214384130091332.40320 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 515173203932752428768260182664.80640 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 1030346407865504857536520365329.61280 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 2060692815731009715073040730659.22560 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 4121385631462019430146081461318.45120 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 8242771262924038860292162922636.90240 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 16485542525848077720584325845273.80480 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 32971085051696155441168651690547.60960 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 65942170103392310882337303381095.21920 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 131884340206784621764674606762190.43840 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 263768680413569243529349213524380.87680 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 527537360827138487058698427048760.75360 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 1055074721654276974117396854097521.50720 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 2110149443308553948234793708195043.01440 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 4220298886617107896469587416390086.02880 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 8440597773234215792939174832780172.05760 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 16881195546468431585878349665560344.11520 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 33762391092936863171756699331120688.23040 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 67524782185873726343513398662241376.46080 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 135049564371747452687026797324482752.92160 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 270099128743494905374053594648965505.84320 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 540198257486989810748107189297931011.68640 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 1080396514973979621496214378595862023.37280 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 2160793029947959242992428757191724046.74560 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 4321586059895918485984857514383448093.49120 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 8643172119791836971969715028766896186.98240 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 17286344239583673943939430057533792373.96480 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 34572688479167347887878860115067584747.92960 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 69145376958334695775757720230135169495.85920 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 138290753916669391551515440460270338991.71840 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 276581507833338783103030880920540677983.43680 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} 553163015666677566206061761841081355966.87360 \\ \times 2 \\ \hline \end{array}$$

# Solve 0.00635

0.00000000 110100000000  
100111

Step 2:-

1.101000000000100111  $\times 2^{-8}$

Step 3:-  $E' = E - 127$

$$-8 = E - 127$$

$$E = 119$$

$E' = E - 1023$

$$-8 = E - 1023$$

$$E = 1015$$

## **FP Arithmetic +/-**

---

- Check for zeros
- Align significands (adjusting exponents)
- Add or subtract significands
- Normalize result

# Floating Point Addition

---

Add the following two decimal numbers in scientific notation:

$$8.70 \times 10^{-1} \text{ with } 9.95 \times 10^1$$

**Rewrite** the smaller number such that its exponent matches with the exponent of the larger number.

$$8.70 \times 10^{-1} = 0.087 \times 10^1$$

---

**Add** the mantissas

$$9.95 + 0.087 = 10.037 \text{ and}$$

write the sum  $10.037 \times 10^1$

Put the result in **Normalised Form**

$$10.037 \times 10^1 = 1.0037 \times 10^2$$

(shift mantissa, adjust exponent)

---

Check for overflow/underflow of the exponent after normalisation

- **Overflow**

The exponent is too *large* to be represented in the Exponent field

- **Underflow**

The number is too *small* to be represented in the Exponent field



---

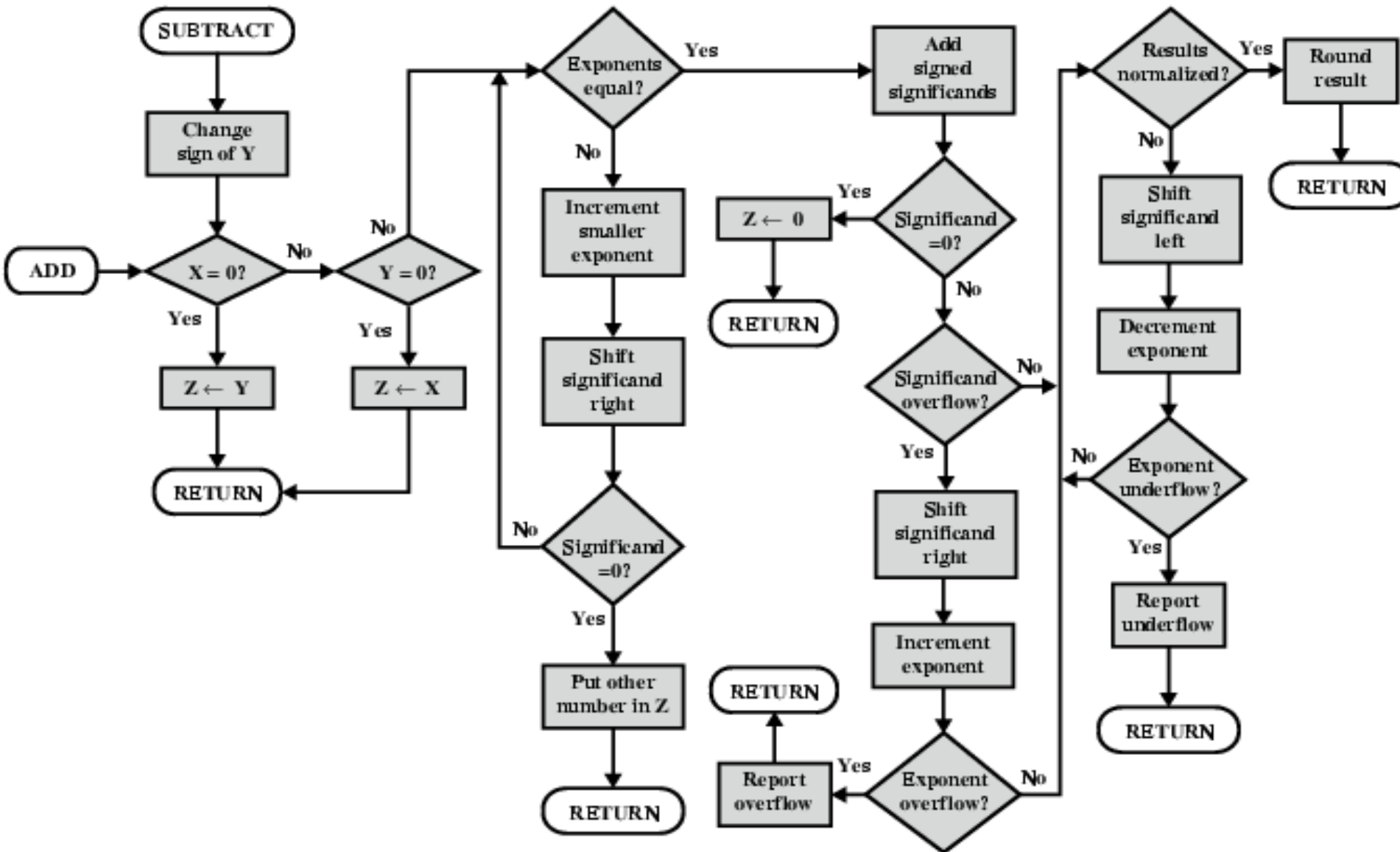
**Round** the result

If the mantissa does not fit in the space reserved for it, it has to be rounded off.

For Example: If only 4 digits are allowed for mantissa

$$1.0037 \times 10^2 \implies 1.004 \times 10^2$$

## FP Addition & Subtraction Flowchart

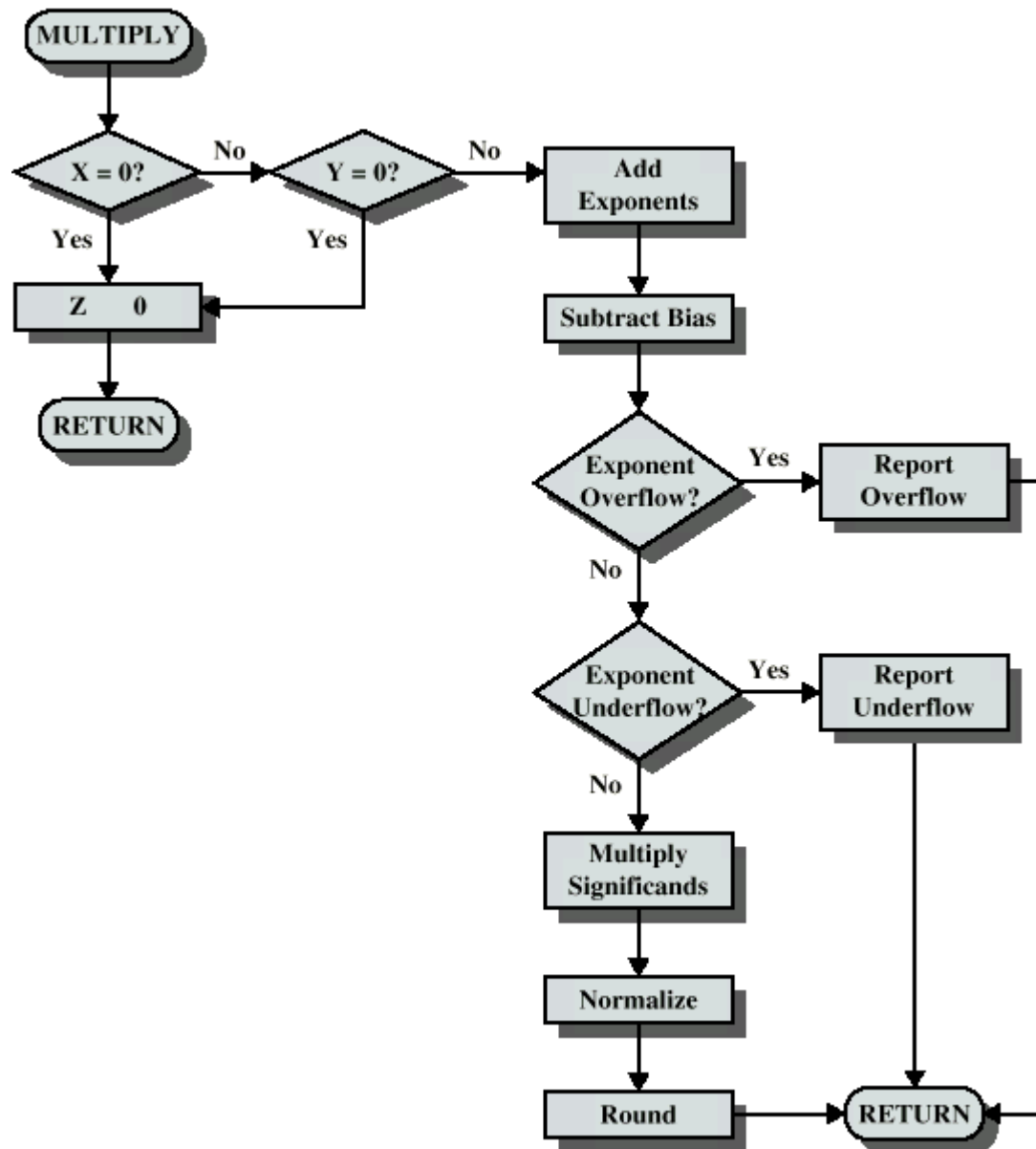


## **FP Arithmetic** $\times/\div$

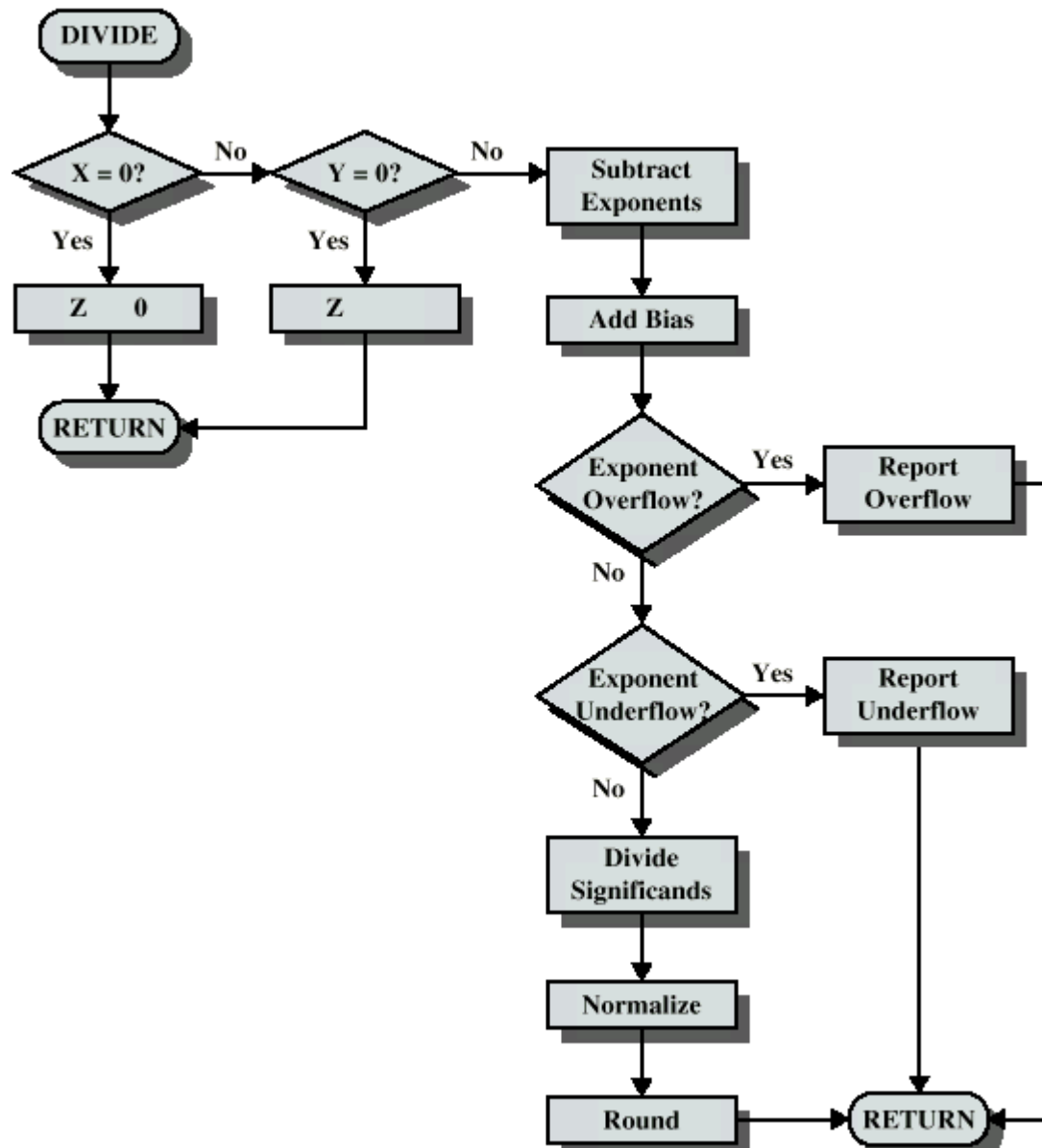
---

- Check for zero
- Add/subtract exponents
- Multiply/divide significands (watch sign)
- Normalize
- Round
- All intermediate results should be in double length storage

# Floating Point Multiplication



# Floating Point Division



## **Example addition in binary Perform 0.5 + (-0.4375)**

$$0.5 = 0.1 \times 2^0 = 1.000 \times 2^{-1} \text{ (normalised)}$$

$$-0.4375 = -0.0111 \times 2^0 = -1.110 \times 2^{-2} \text{ (normalised)}$$

Rewrite the smaller number such that its exponent matches with the exponent of the larger number.

$$-1.110 \times 2^{-2} = -0.1110 \times 2^{-1}$$

Add the mantissas:

$$1.000 \times 2^{-1} + -0.1110 \times 2^{-1} = 0.001 \times 2^{-1}$$

Normalise the sum, checking for overflow/underflow:

$$0.001 \times 2^{-1} = 1.000 \times 2^{-4}$$

$$-126 \leq -4 \leq 127 \implies \text{No overflow or underflow}$$

Round the sum:

The sum fits in 4 bits so rounding is not required

$$\text{Check: } 1.000 \times 2^{-4} = 0.0625 \text{ which is equal to } 0.5 - 0.4375$$