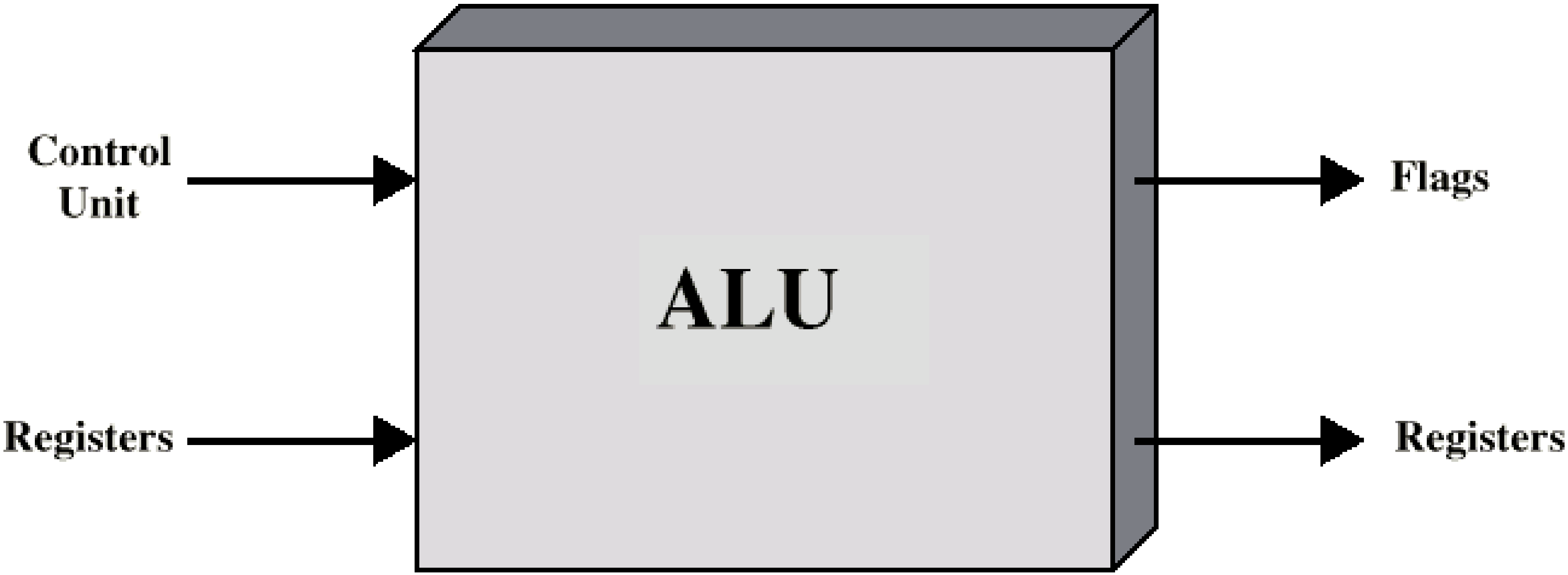
William Stallings
Computer Organization and Architecture
6th Edition

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 twos compliment of substahend and add to minuend
  - i.e. a - b = a + (-b)

- So we only need addition and complement circuits

A	B	Sum
0	0	0
0	1	1
1	0	1
1	1	0, Carry 1
1	1	1,Carry 1

Example of 2's Compliment

```
 5 = 00000101  ┐
     ↓↓↓↓↓↓↓↓  ├ Complement Digits
     11111010  ┘
           +1  ┐
     ────────  ├ Add 1
-5 = 11111011  ┘
```

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

```
-13 = 11110011  ┐
      ↓↓↓↓↓↓↓↓  ├ Complement Digits
      00001100  ┘
            +1  ┐
      ────────  ├ Add 1
 13 = 00001101  ┘
```

Find 2's compliment

1000

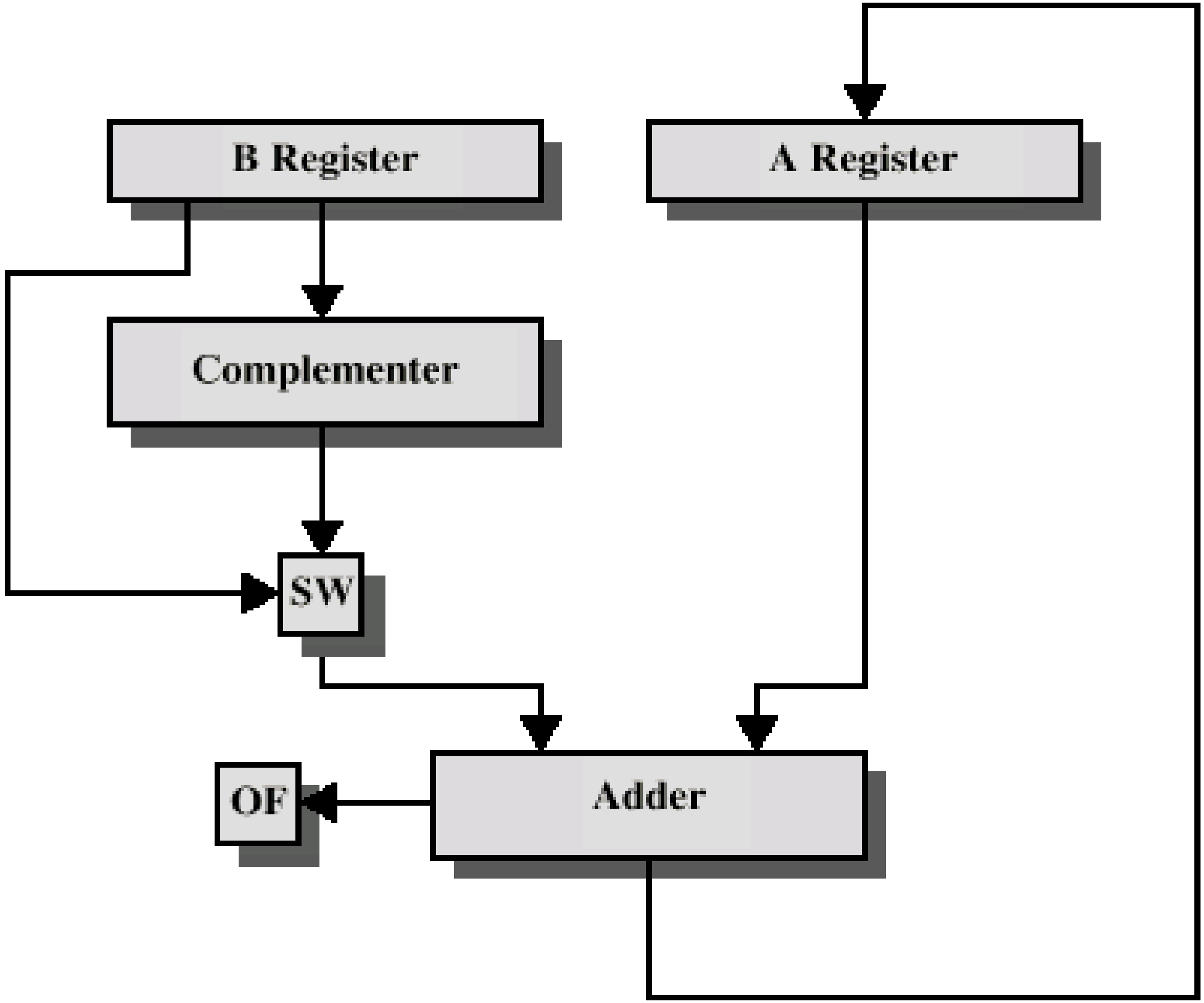
0100

111001

101010

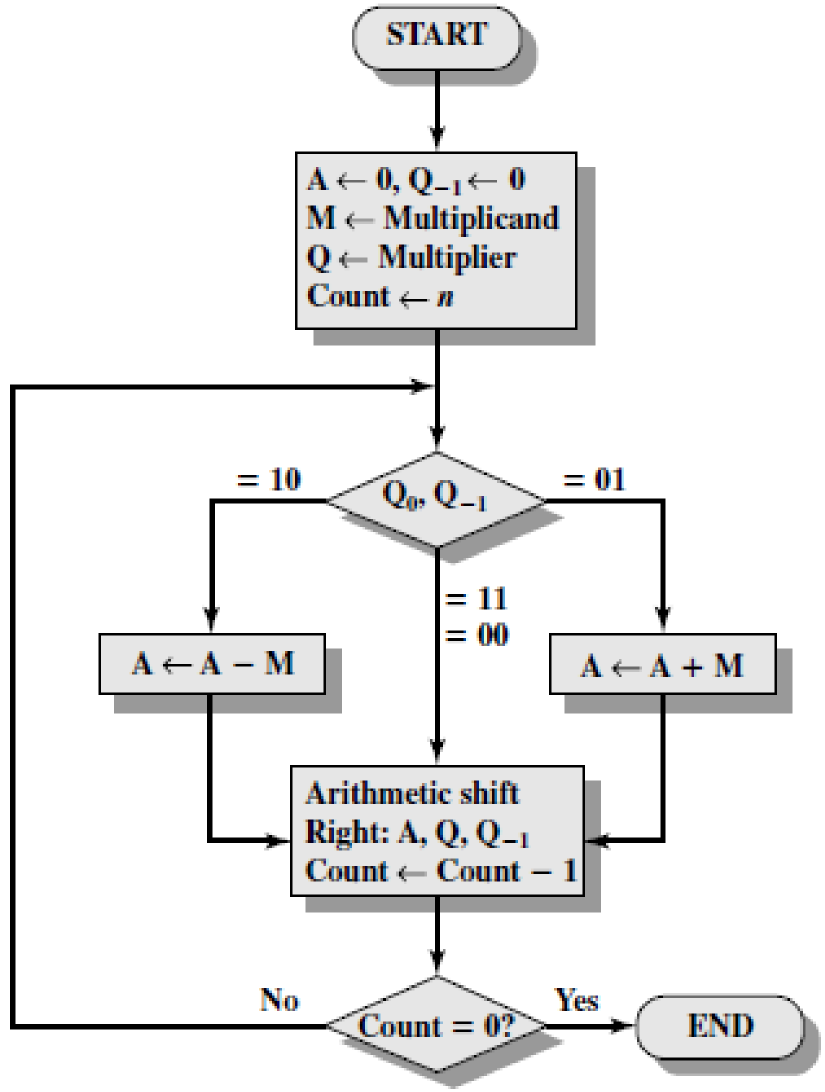
100000

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

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

A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial Values	
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_{-1}	M		
0000	0011	0	0111	Initial values	
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

Figure 9.13 Example of Booth's Algorithm (7 × 3)

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 (0011)** , Q = 6

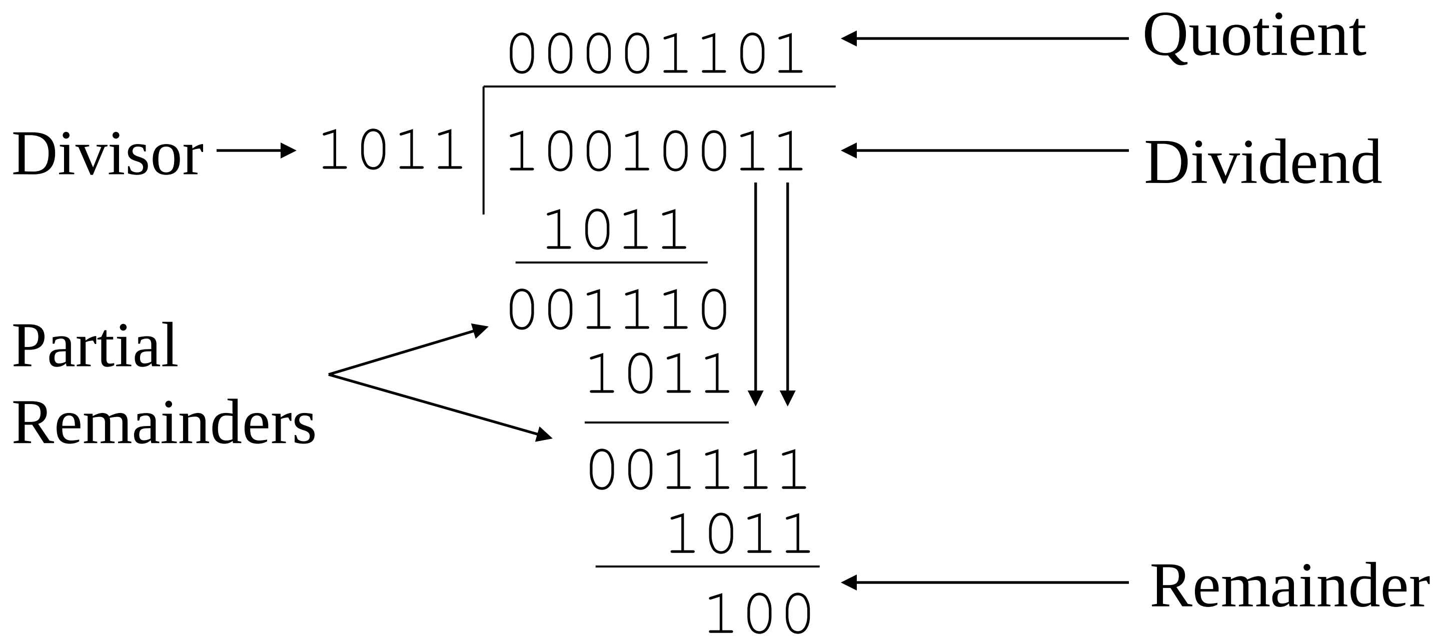
-M=13 (1101)

A. M = -19 , Q = -20

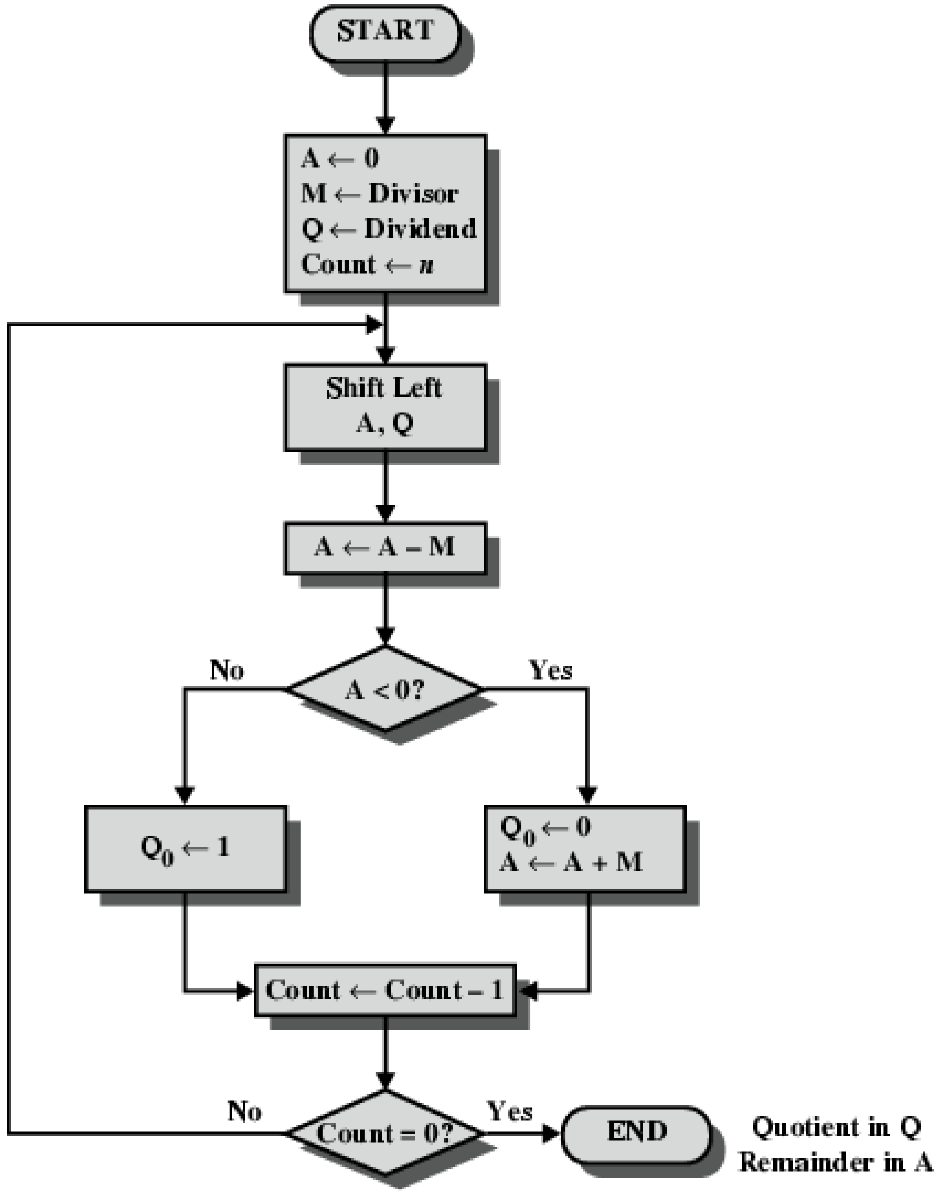
Division

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

Division of Unsigned Binary Integers



Flowchart for Restoring Division



DATE / /

$M \leftarrow 3\overline{)7} \rightarrow Q$

$M = 0011$

$-M = 1101$

$Q = 0111$

$$\begin{array}{r|l|l} 2 & 7 & \\ 2 & 3 & 1 \\ & 1 & 1 \end{array} \qquad \begin{array}{r|l|l} 2 & 3 & \\ & 1 & 1 \end{array}$$

1

1 1 0 0

0000

$A = 1101$

$M = 0011$

0000

cycle 1

A	Q	
0 0 0 0	0 1 1 1	
0 0 0 0	1 1 1 $\boxed{\ }$	Shift left A, Q
1 1 0 1	1 1 1 $\boxed{\ }$	$A = A - M$
0 0 0 0	1 1 1 $\boxed{0}$	set $q_0 = 0$, $A = A + M$ → Restore

	A	Q	
0001	0 0 0 1	1 1 0 $\square$	shift left
1101			A = A − M
1110	1 1 1 0	1 1 0 $\boxed{0}$	
			set $Q_0 = 0$
1110			A = A + M
0011	0 0 0 1	1 1 0 0	
0001			

Restore

0001

0011   1000 $\boxed{}$ Shift left

0011
1101
0000

0000   1000 $\boxed{1}$ A = A − M

set $q_0 = 1$

0000   1001

0001
1101
1110

0001   001 $\boxed{}$ shift left

1110   0010 $\boxed{0}$ A = A − M

set $q_0 = 0$
A = A + M

$\underbrace{0001}_{\text{Remainder} = 1}$ $\underbrace{0010}_{\text{Quotient} = 2}$

$M = 27$; $Q = 55$

$M = 011011$

$-M = 100101$

$Q = 110111$

	A	Q	
de1	000000	110111	
	000001	10111□	shift left A,Q
	100110	10111$\boxed{0}$	A=A-M
	000001	101110	Set $Q_0=0$; A=A+M

cycle2	000011	01110□	shift left A,Q
	101000	01110$\boxed{0}$	A = A − M
			set Q_0=0
	000011	011100	A = A + M
cycle3	000110	11100□	shift left A,Q
	101011	11100$\boxed{0}$	A = A − M
	000110	111000	Set Q_0=0; A=A+M
cycle4	001101	11000□	shift left A,Q
	110010	11000$\boxed{0}$	A = A − M

cycle4	001101	11000 $\square$	shift left A,Q
	110010	11000 $\boxed{0}$	A=A-M
	001101	110000	set $Q_0=0$., A=A+
cycle5	011011	10000 $\square$	shift left A,Q,
	000000	10000 $\boxed{1}$	set $Q_0=1$
cycle6	000001	00001 $\square$	shift left A,Q
	100110	00001 $\boxed{0}$	A=A-M
			Set $Q_0=0$,
	000001	000010	A= A+M

R = 1 Q = 2

M = 27 ; q = 55

M = 011011

−M = 100101

q = 110111

Cycle	A	Q	
	000000	110111	
cycle1	000001	10111□	shift left A,Q
	100110	101110	A = A − M
	000001	101110	Set $q_0 = 0$; A = A+M
cycle2	000011	01110□	shift left A,Q
	101000	011100	A = A − M
			set $q_0 = 0$
	000011	011100	A = A + M
cycle3	000110	11100□	shift left A,Q
	101011	111000	A = A − M
	000110	111000	Set $q_0 = 0$; A = A+M
cycle4	001101	11000□	shift left A,Q
	110010	110000	A = A − M
	001101	110000	set $q_0 = 0$, A = A+M
cycle5	011011	10000□	shift left A,Q,
	000000	100001	set $q_0 = 1$
cycle6	000001	00001□	shift left A,Q
	100110	000010	A = A − M
			Set $q_0 = 0$
	000001	000010	A = A + M

R = 1 Q = 2

Solve using Restoring Division

A. M = 5 , Q = 5 , A=0000 , Q=0010

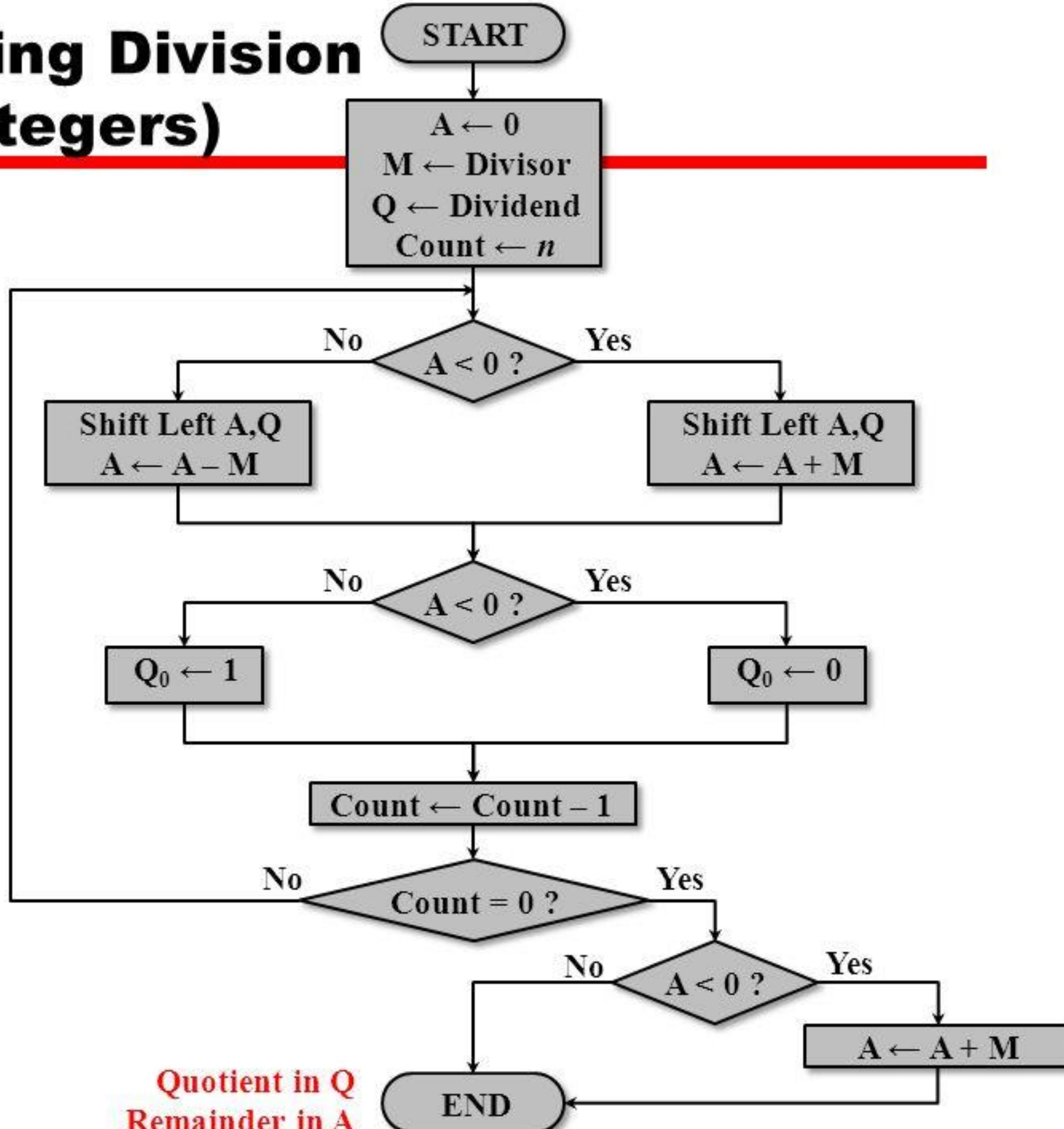
B. M = 12, Q = 26, A=00010 , Q= 00010

C. M = 9, Q = 19 , A=00001 , Q =00010

D. M = 32 , Q = 59, A=011011, Q=000001

E. M = 17 , Q = 42, A=001000 , Q=000010

Non-Restoring Division (Positive Integers)



$M = 2$; $M = 0010$

$-M = 1110$

$Q = 4$; $Q = 0100$

	A	Q	
	0000	0100	
Shift left A,Q	0000	100$\square$	①
A = A − M, set $q_0 = 0$	1110	100$\boxed{0}$	①
Shift left A,Q	1101	000$\square$	②
A = A + M, set $q_0 = 0$	1111	000$\boxed{0}$	②

shift left A,Q	1 1 1 0	0 0 0 □	③
A = A + M	0 0 0 0	0 0 0 $\boxed{1}$	
set $q_0 = 1$			
shift left A,Q	0 0 0 0	0 0 1 □	④
A = A − M	1 1 1 0	0 0 1 $\boxed{0}$	
set $q_0 = 0$			

count = 0

A = A + M

$$\begin{array}{rl} 1110 & (A) \\ 0010 & (M) \\ \hline \cancel{1}\ 0000 & \end{array}$$

$\underbrace{0\,0\,0\,0}$ $\underbrace{0\,0\,1\,0}$

A = Remainder Q = Quotient

Solve using Non Restoring

A. M = 5 ,   Q = 5,A=0000,Q=0001.

B. M = 12,   Q = 26,A=000010, Q=000010.

C. M = 9,   Q = 19,A=00001,Q=00010.

D. M = 32 , Q = 59,A=011011,Q=000001.

E. M = 17 , Q = 42,A=001000,Q=000010

Booths Recoding / Bit pair recording

STEPS

Booth's Recoding algorithm

5 × 3

M Q

M 0101 Q = 0011

−M 1011

Step 1 : Table for M

Operation	Value
0	0000 0000
+1	0000 0101
−1	1111 1011
+2	0000 1010
−2	1111 0110

left shift +1

left shift −1

Step 2 : Value of Q

Q_{-1}

$$0 \quad 0 \quad 1 \quad 1 \quad 0$$

$$0 \quad +1 \quad 0 \quad -1$$

2(1st nos) + 2nd nos

$$2(0) + 1 \qquad 2(0) + (-1)$$

$$1 \qquad -1$$

Step 3: M * Q

0 1 0 1

1 - 1

1 1 1

1 1 1 1 1 0 1 1

0 0 0 1 0 1 + +

0 0 0 0 1 1 1 1

Discard carry $\boxed{1}$

$2^3 \; 2^2 \; 2^1 \; 2^0$

$8 + 4 + 2 + 1 = 15$

Solve using Booths Recoding

1. M= 5 , Q = 4 (4 bits)= 00010100 (20)
2. M=9 , Q = -6 (5 bits)=11110 01010 (-54)
3. M=15 , Q=-10 (5 bits)=11011 01010(-150)
4. M= -13 ,Q = -20 (6 bits)=000100000100(260)

Sample mix problems-Kindly refrain referring to flowchart.

1. Booth's Algorithm = 000 100 000 100(260)

A= 110011 (Multiplicand)

B= 101100 (Multiplier)

2. Booth's Recoding = 0110 1010/11011 01010

M= (15)

Q= (-10)

3. Non Restoring Division

M=11 , Q= 21 , A= 01010 , Q= 00001

4. Restoring Division

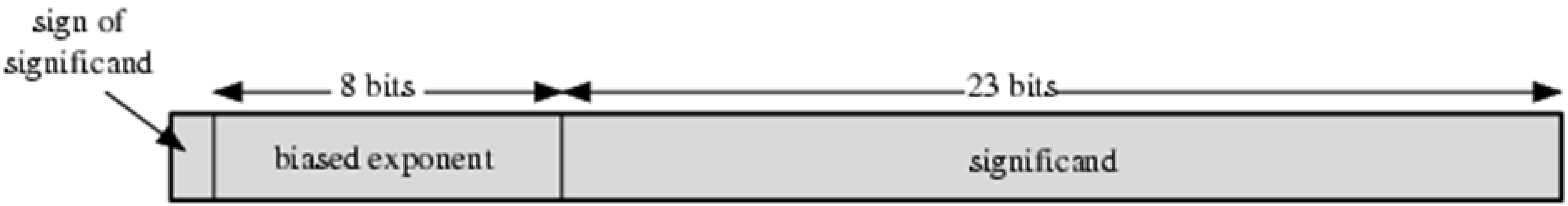
M=14 , Q= 15, A=00001 , Q = 00001

Floating Point

Sign bit	Biased Exponent	Significand or Mantissa

- +/- .significand x $2^{exponent}$
- Misnomer
- Point is actually fixed between sign bit and body of mantissa
- Exponent indicates place value (point position)

Floating Point Examples

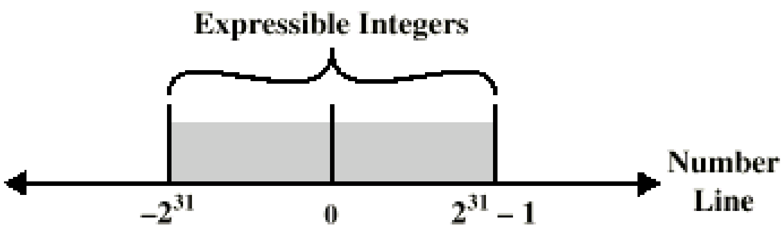


(a) Format

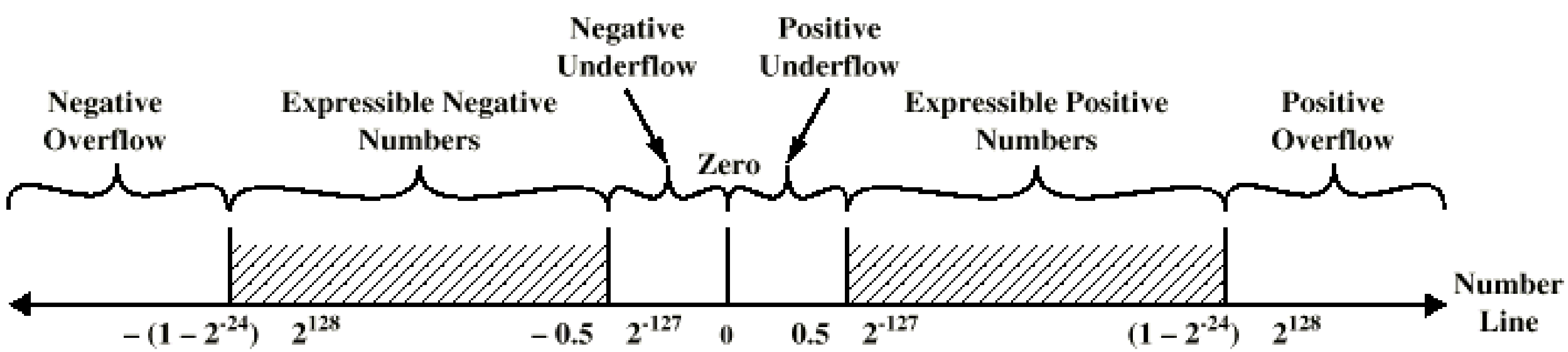
Signs for Floating Point

- 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

Expressible Numbers



(a) Twos Complement Integers

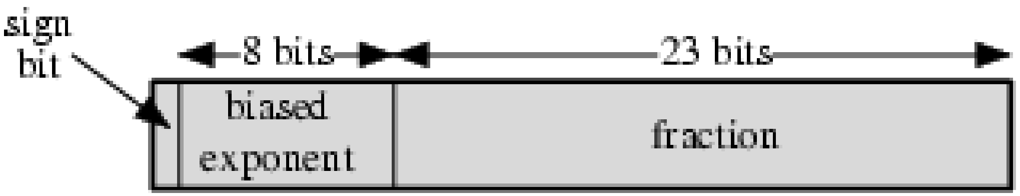


(b) Floating-Point Numbers

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

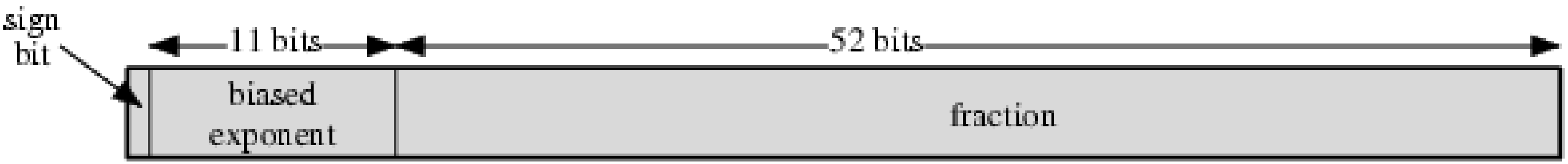
IEEE 754 Formats



(a) Single format

32 BIT

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



(b) Double format

64 BIT

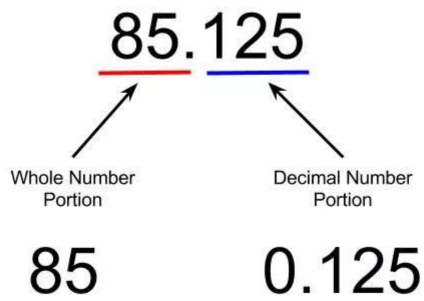
$(1.N)2^{E-1023}$

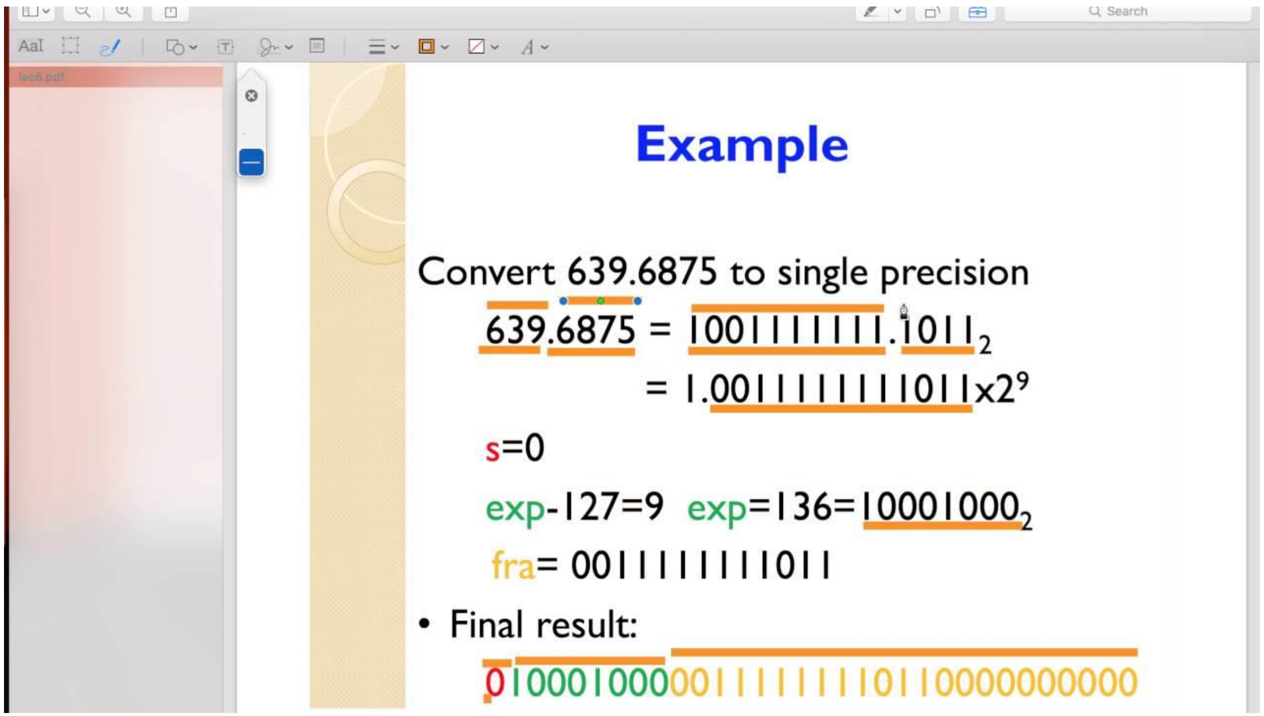
Steps

- 1. Convert Decimal to Binary
- 2. Normalization
  - Rewriting Step 1 into (1.N) form
  - Ex: 1 1 1 . 0 1 1 = **1 .** 1 1 0 1 1 x 2^{2}
  - Ex: 0 . 0 0 0 1 0 = 0 0 0 0 **1 .** 0 x 2^{-4}

Exponent

- 3.Biasing
  - Applying Single Precision (E – 1 2 7) & Double Precision (E – 1 0 2 3) on exponent from Step 2
- 4. Representation in Single (32 bit)and Double Precision (64 bit) Format


85.125
Whole Number Portion
Decimal Number Portion
85
0.125


Search
AaI
lec6.pdf
Example
Convert 639.6875 to single precision
639.6875 = 1001111111.1011₂
= 1.0011111111011x2⁹
s=0
exp-127=9 exp=136=10001000₂
fra= 0011111111011
• Final result:
01000100000111111111011 0000000000

Solved Example

Eg 12.25

Step 1: Converting Dec to Bin

2	12	
2	6	0
2	3	0
	1	1
		1

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

$$\underline{12} \;.\; \underline{25}$$

$$1100.01$$

Step 2: Normalization (1. N)

$1.100 01 \times 2^{3}$ → 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	130	
2	65	0
2	32	1
2	16	0
2	8	0
2	4	0
2	2	0
	1	0
		1

2	1026	
2	513	0
2	256	1
2	128	0
2	64	0
2	32	0
2	16	0
2	8	0
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)

Signbit		
0	10000000010	10001
1 bit	11 bits	52 bits

Solve

25.44	SP- 0|100000|1001 0111 0000 1010 0011 110
	DP- 0|10000000011|1001 0111 0000 1010 0011 110
0.00635	SP- 0|1110111|00000001101000...
	DP- 0|1111110111|00000001101000...
-125.10	SP- 1 | 10000101| 1111 010001
	DP- 1 | 10000000101| 1111 010001
-13.54	SP- 1|10000010|10110001010
	DP- 1|10000000010|10110001010

Sample Problems to Solve

1) 178.1875

SP 0|10000110|01100100011

DP 0|10000000110|

1) 309.175

SP 0|10000111|01011101001011

DP 0|10000000111|

1) 1259.125

SP 0|10001001|00111010110010000...(9 zeroes)

DP 0|10000001001|

1) 0.0625

SP 0|01111011|0000000....

DP 0|01111111|00000.....

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 twos 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 twos 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

$$D = \text{dividend}$$
$$Q = \text{quotient}$$
$$V = \text{divisor}$$
$$R = \text{remainder}$$

The results of Figure 9.17 are consistent with this formula.

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

(a) (7)/(3)

Solve

a) 7 / -3

b) -7 (Q) / 3 (M)

c) -7 (Q) / -3 (M)

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

(b) (7)/(–3)

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

(c) (–7)/(3)

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

(d) (–7)/(–3)

- Dividend negative → Remainder –ve

4 phases of 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:

8.70×10^{-1} with 9.95×10^{1}

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

$8.70 \times 10^{-1} = 0.087$ (Note !) $\times 10^{1}$

Add the mantissas

$9.95 + 0.087 = 10.037$ and

write the sum 10.037×10^1

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

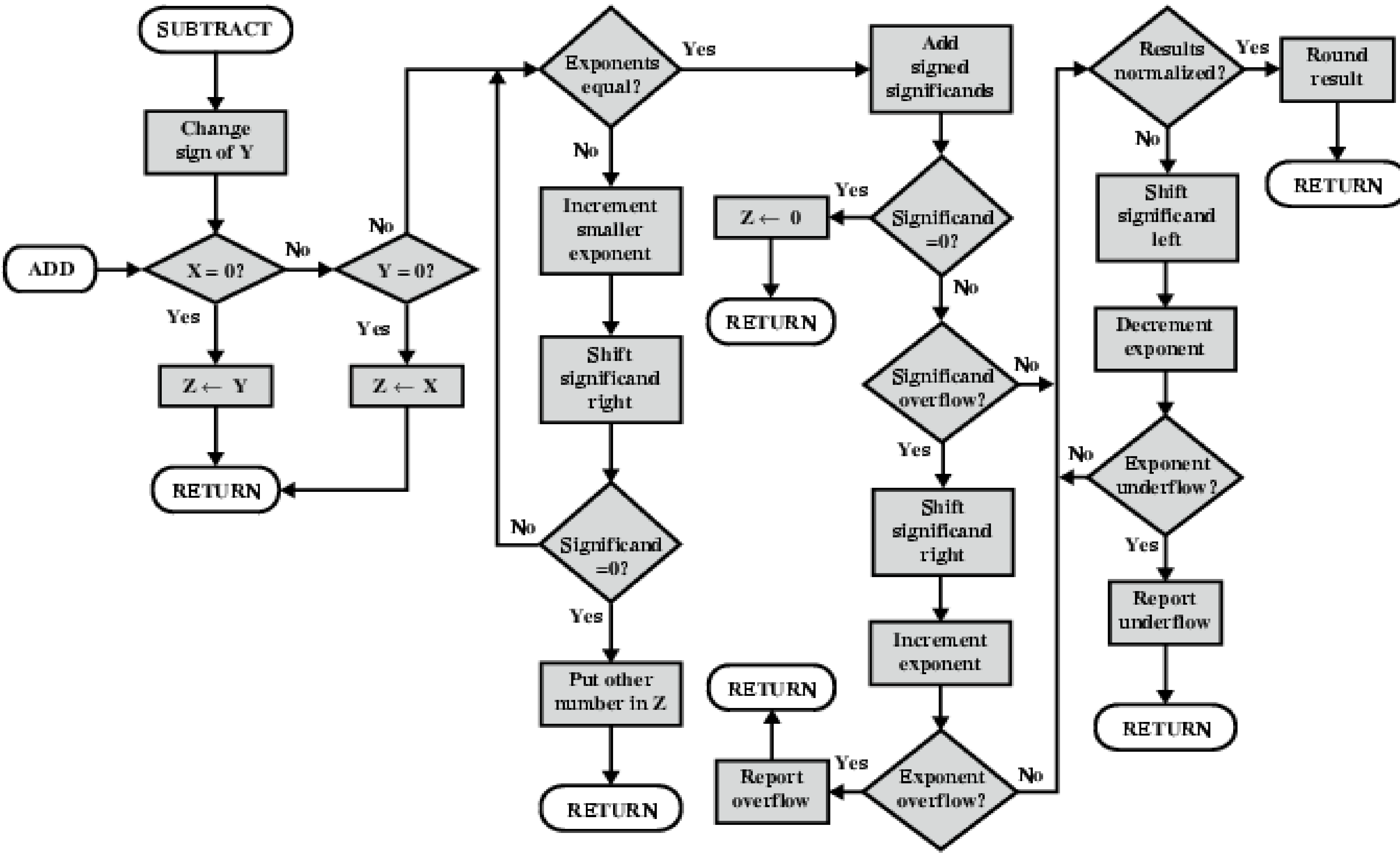
Round the result

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

1.0037×10^2 ===> 1.004×10^2

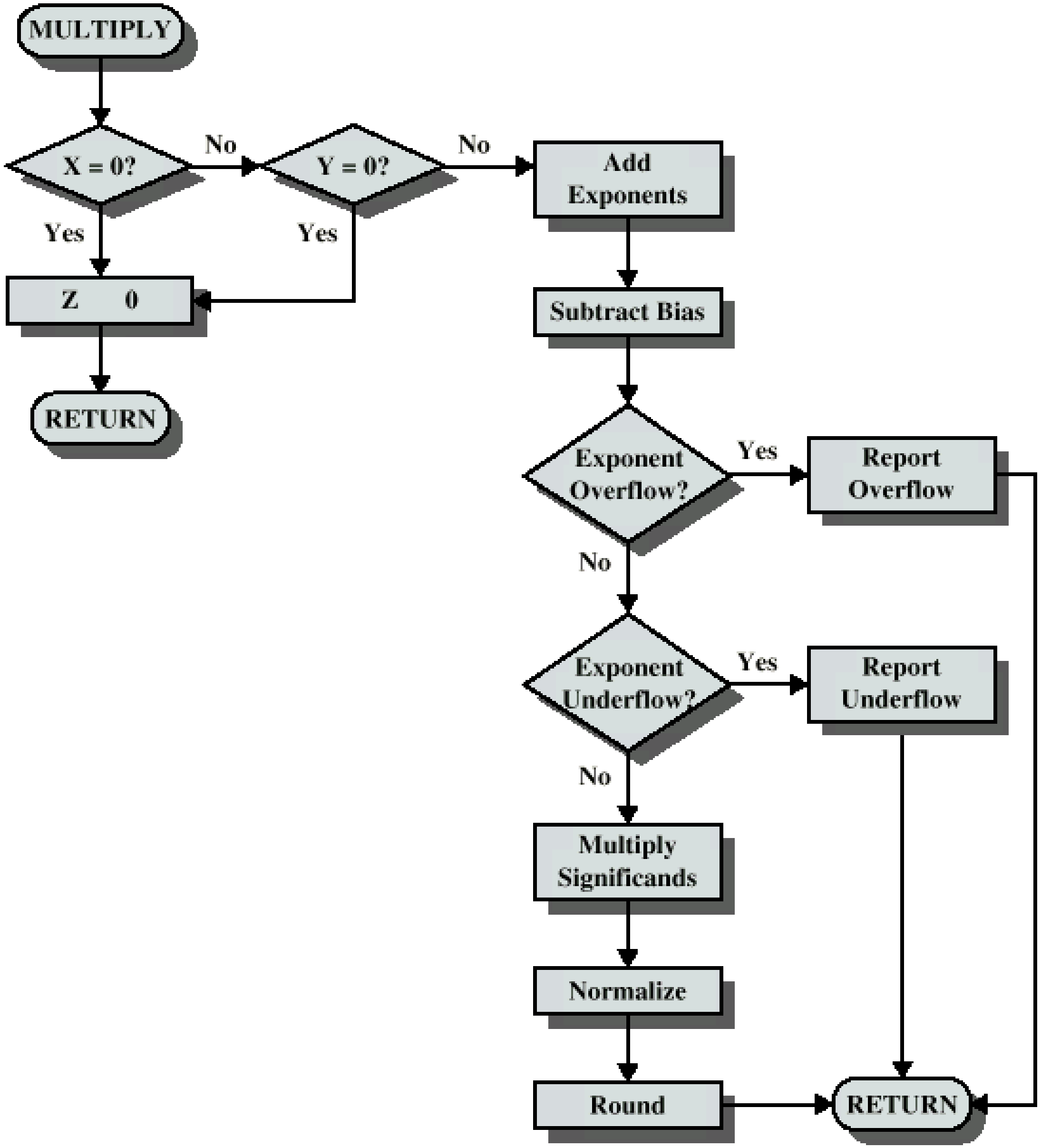
FP Addition & Subtraction Flowchart



FP Arithmetic x/÷

- 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