William Stallings
Computer Organization and Architecture
6th Edition

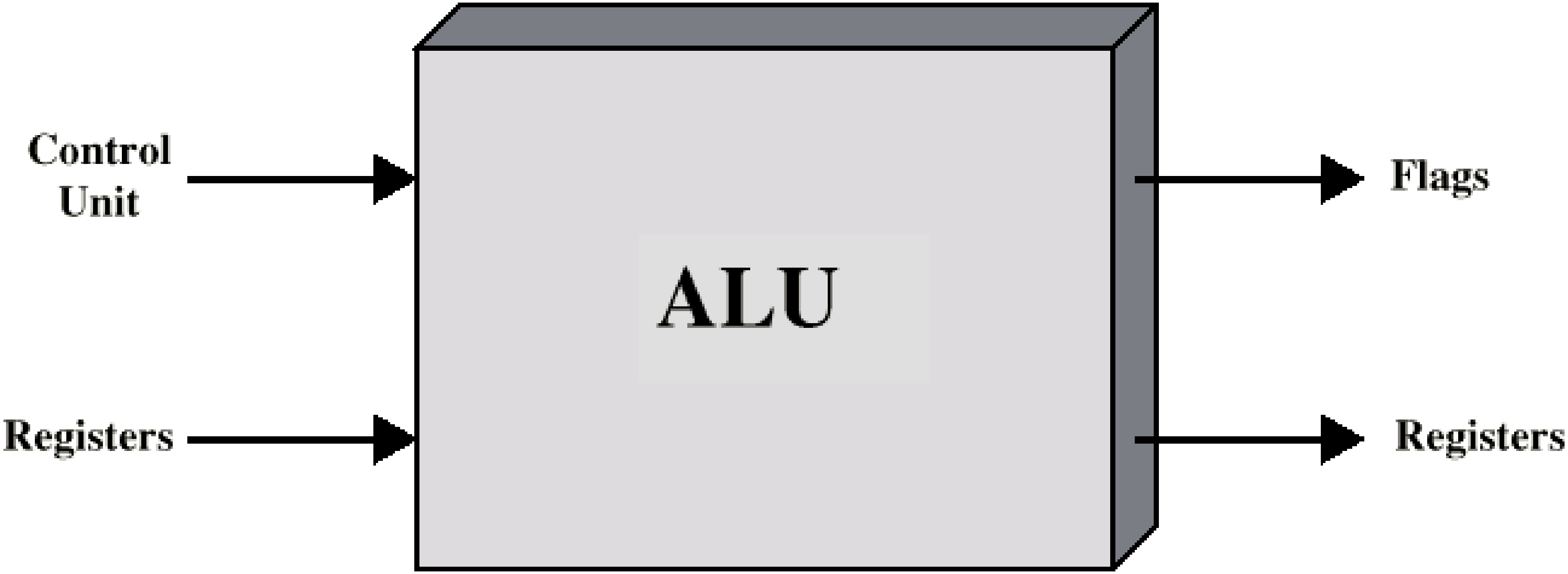
Chapter 9

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



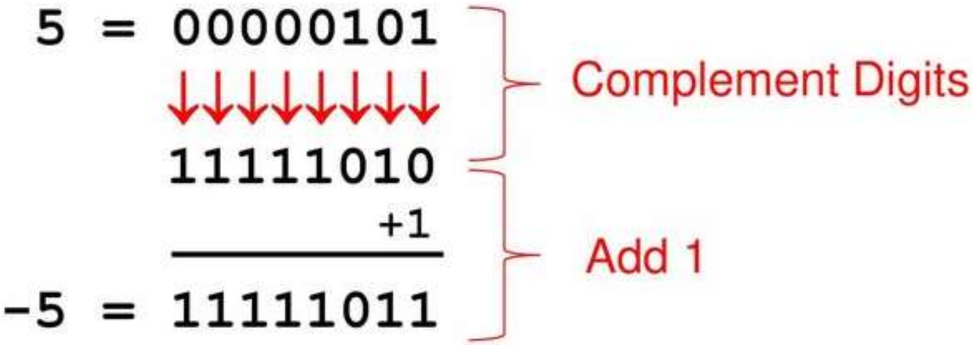
Addition and Subtraction

- 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

Example of 2’s Compliment



5 = 00000101

↓↓↓↓↓↓↓↓

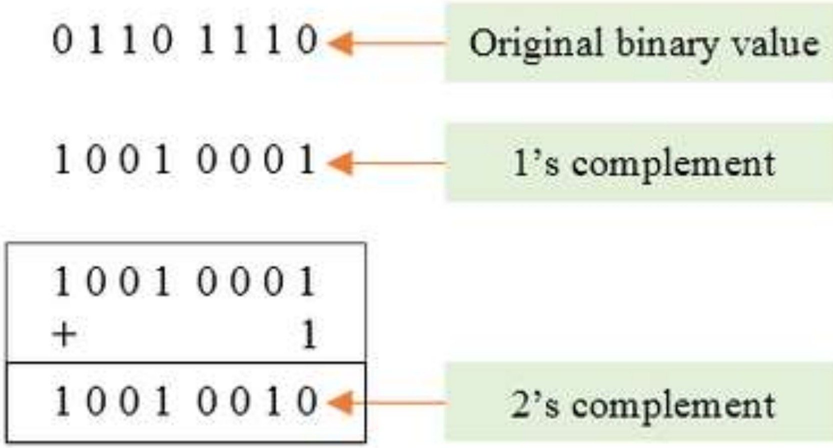
11111010

+1

-5 = 11111011

Complement Digits

Add 1



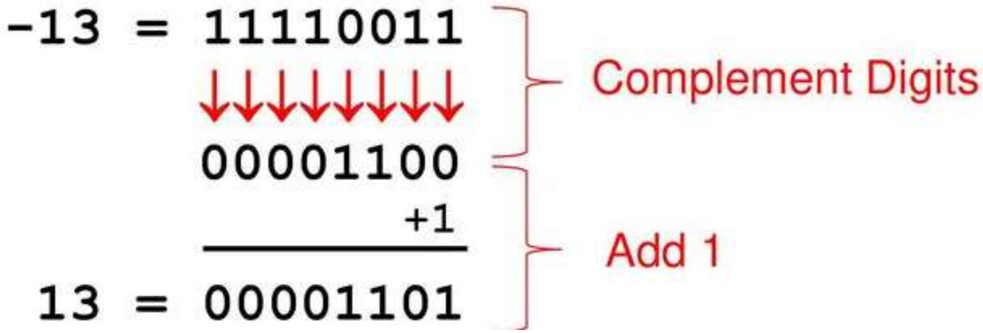
0 1 1 0 1 1 1 0 ← Original binary value

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



-13 = 11110011

↓↓↓↓↓↓↓↓

00001100

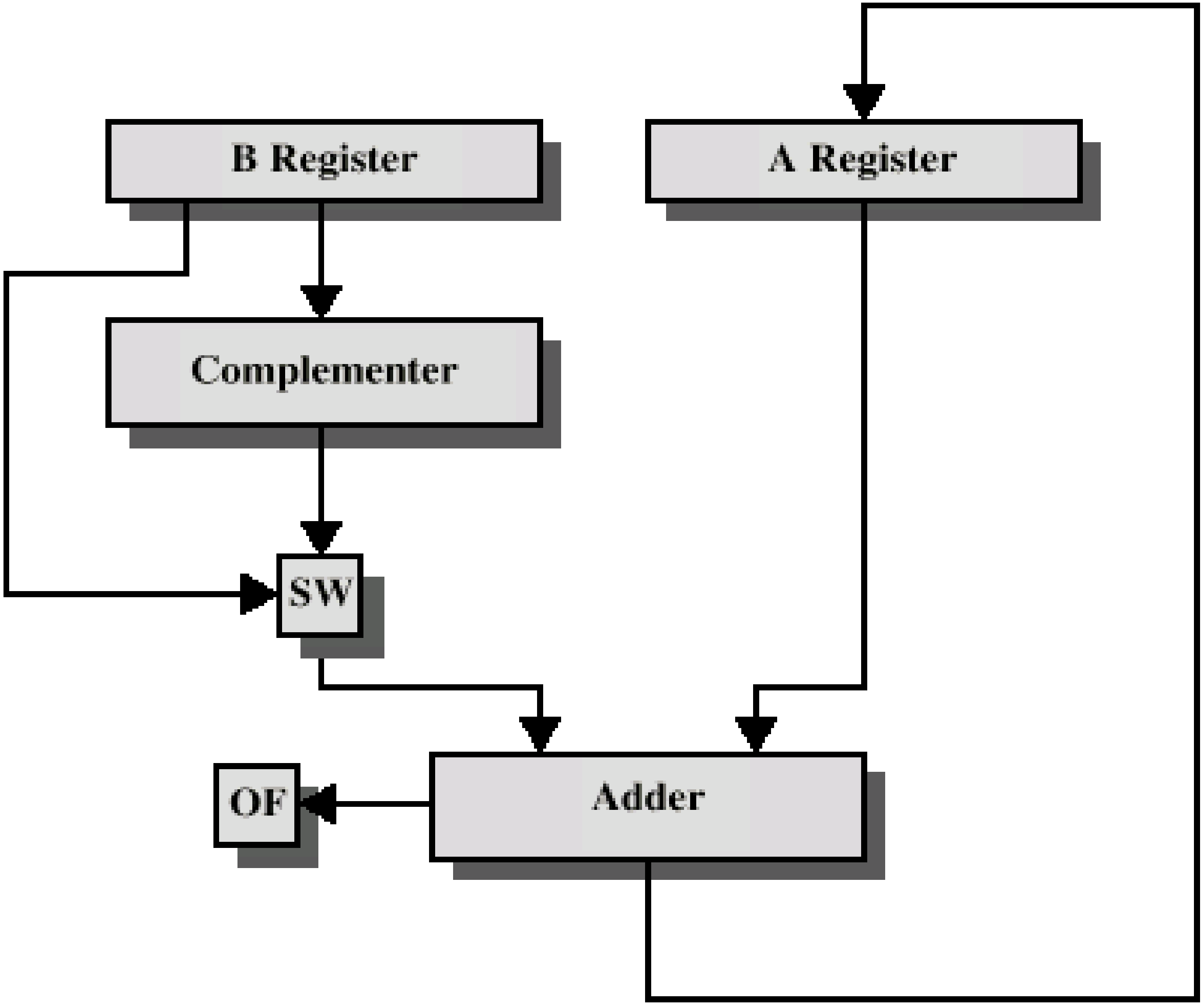
+1

13 = 00001101

Complement Digits

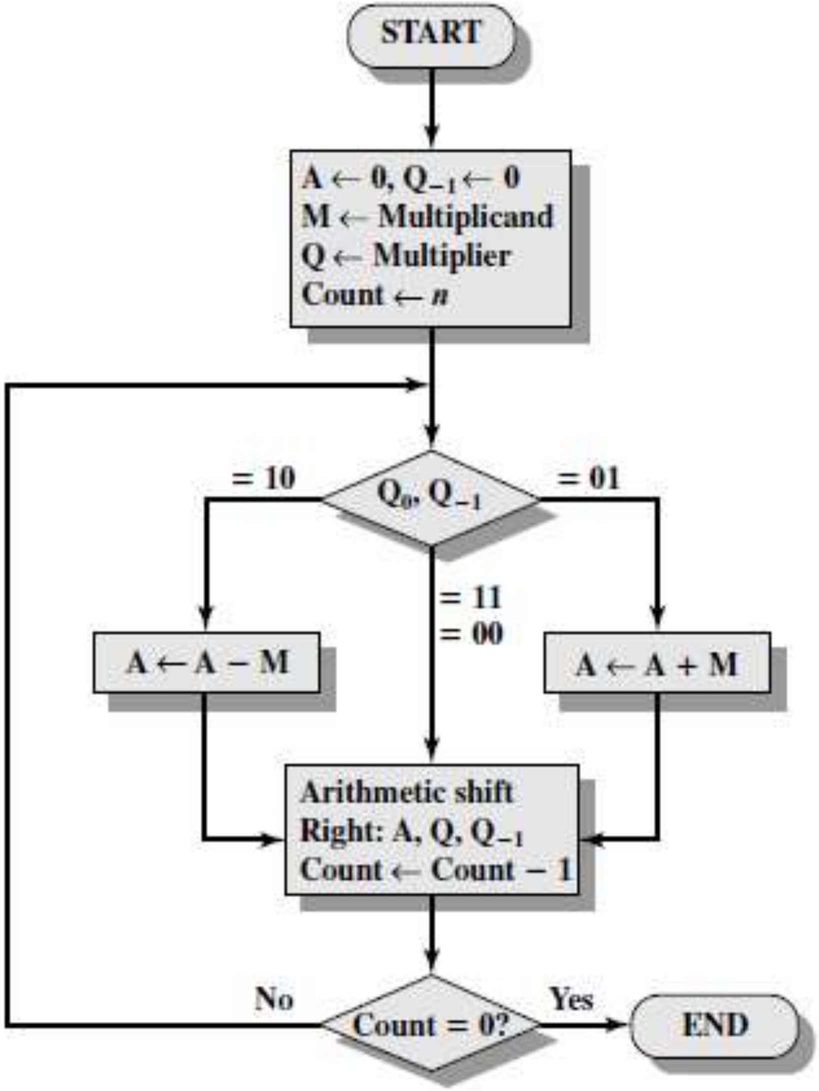
Add 1

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

$$\overset{M}{(7)} * \overset{Q}{(3)}$$

M ⇒ 0111

-M ⇒ 1001

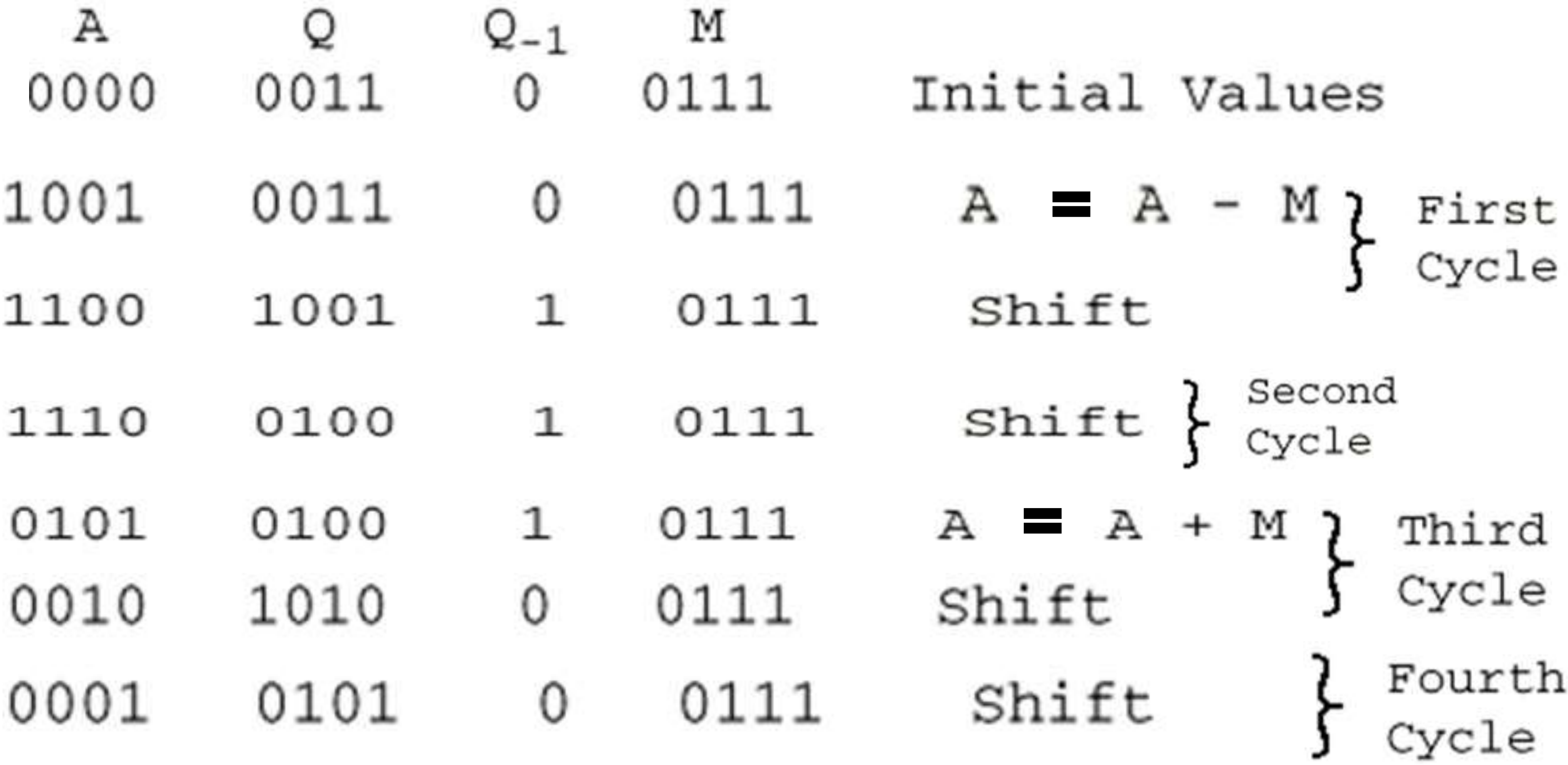
Q ⇒ 0011

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

$$\begin{array}{r} 1110 \\ 0111 \\ {\scriptstyle 1\,1} \\ \hline \boxed{1}0101 \end{array}$$

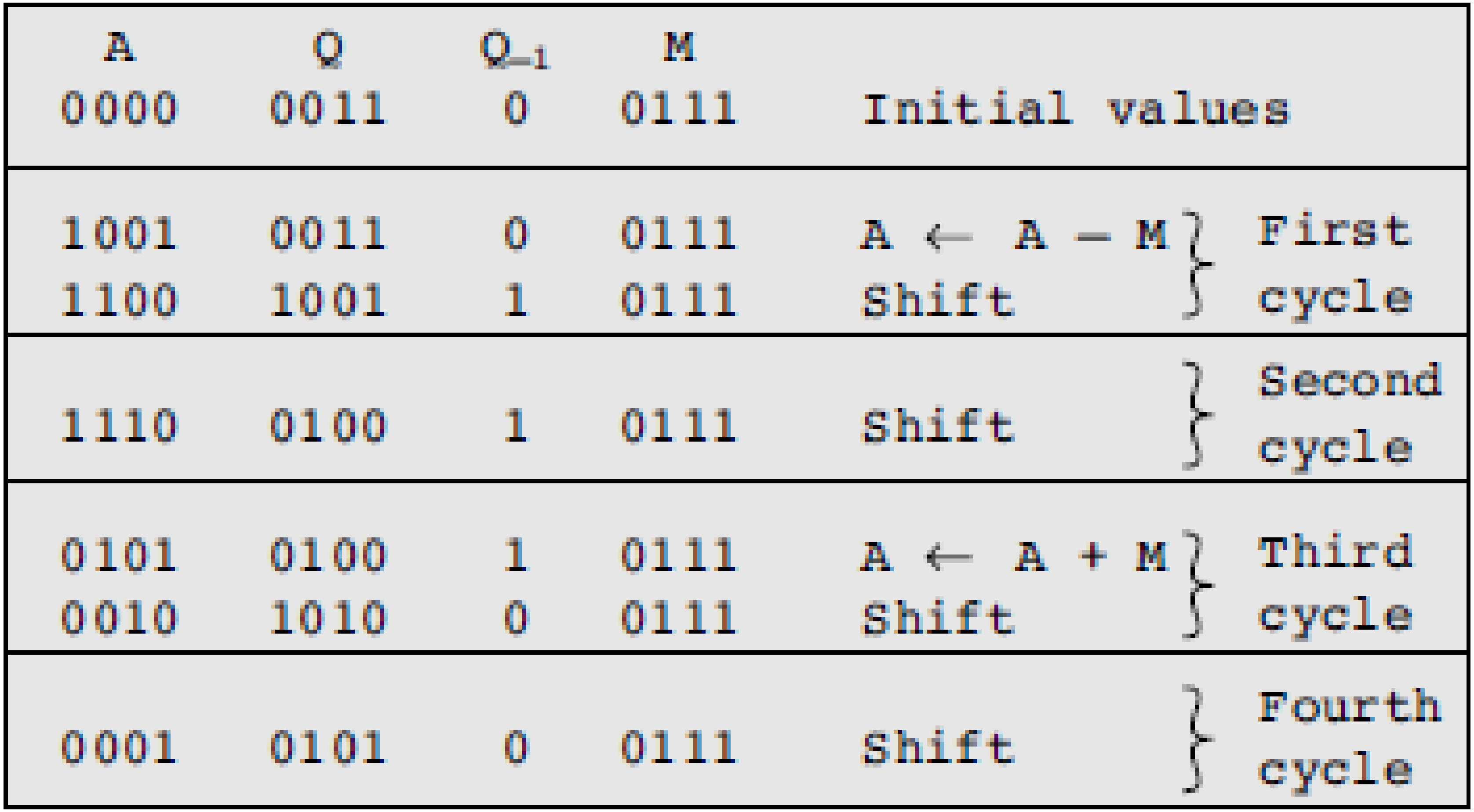
AQ ⇒ 0001 0101 ⇒ 21

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



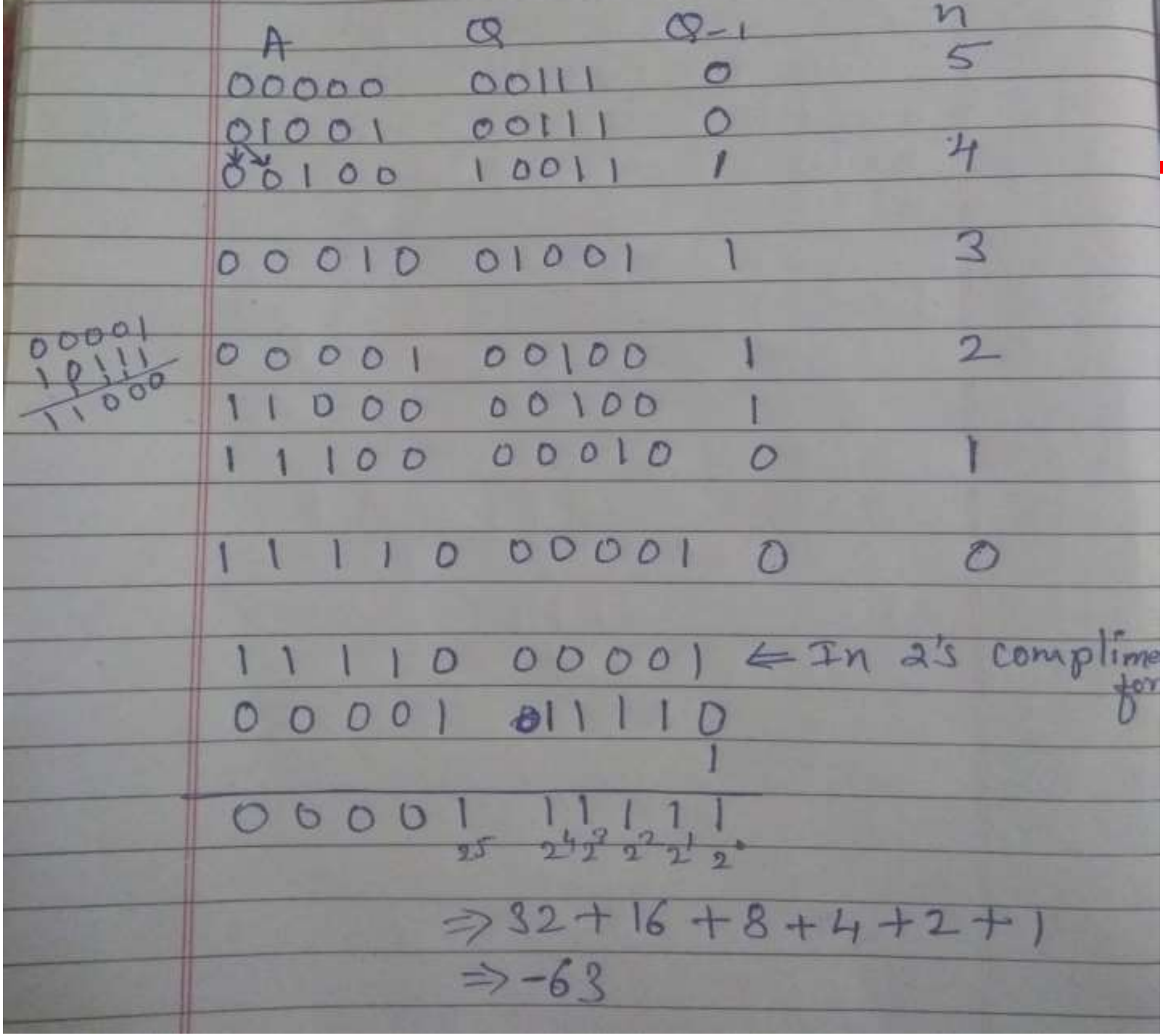
A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial Values	
1001	0011	0	0111	A = A - M	First Cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second Cycle
0101	0100	1	0111	A = A + M	Third Cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth Cycle

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)



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

00001
10111
11000

11110 00001 ⇐ In 2's complime
for

00001 011110
1

00001 11111

2^5 2^4 2^3 2^2 2^1 2^0

⇒ 32 + 16 + 8 + 4 + 2 + 1

⇒ −63

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

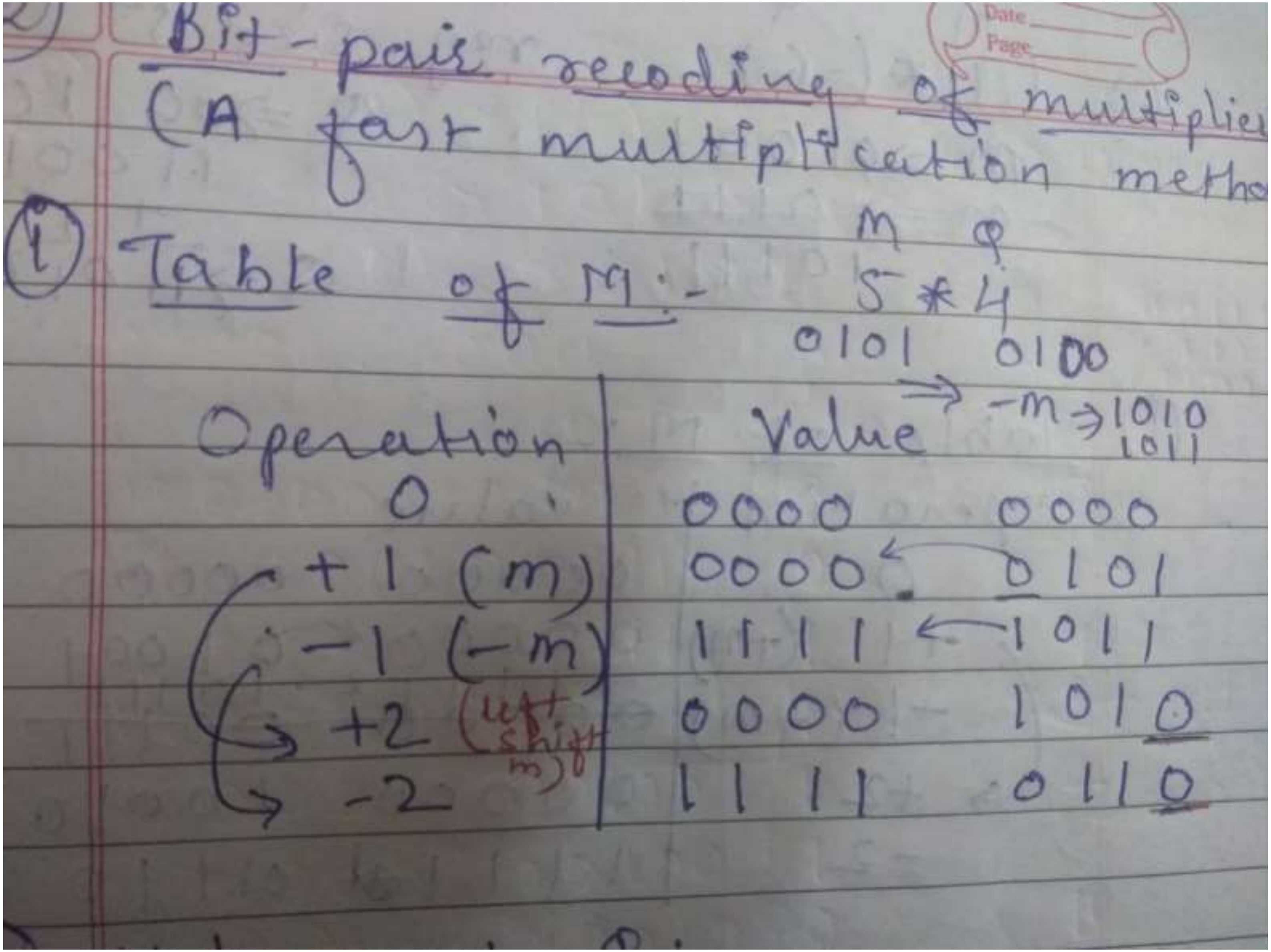
Booths Recoding / Bit pair recording

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



2) Bit-pair recoding of multiplier
(A fast multiplication metho

1) Table of M:-

M Q
5 * 4
0101 0100
⇒ -m ⇒ 1010
1011

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

↳ -2 m)0 | 1 1 1 1

(2) <u>Value</u> <u>of</u> <u>Q</u> :-

$$\begin{array}{ccccc} & & Q_2 & & Q_{-1} \\ 0 & 1 & 0 & 0 & 0 \end{array}$$

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

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

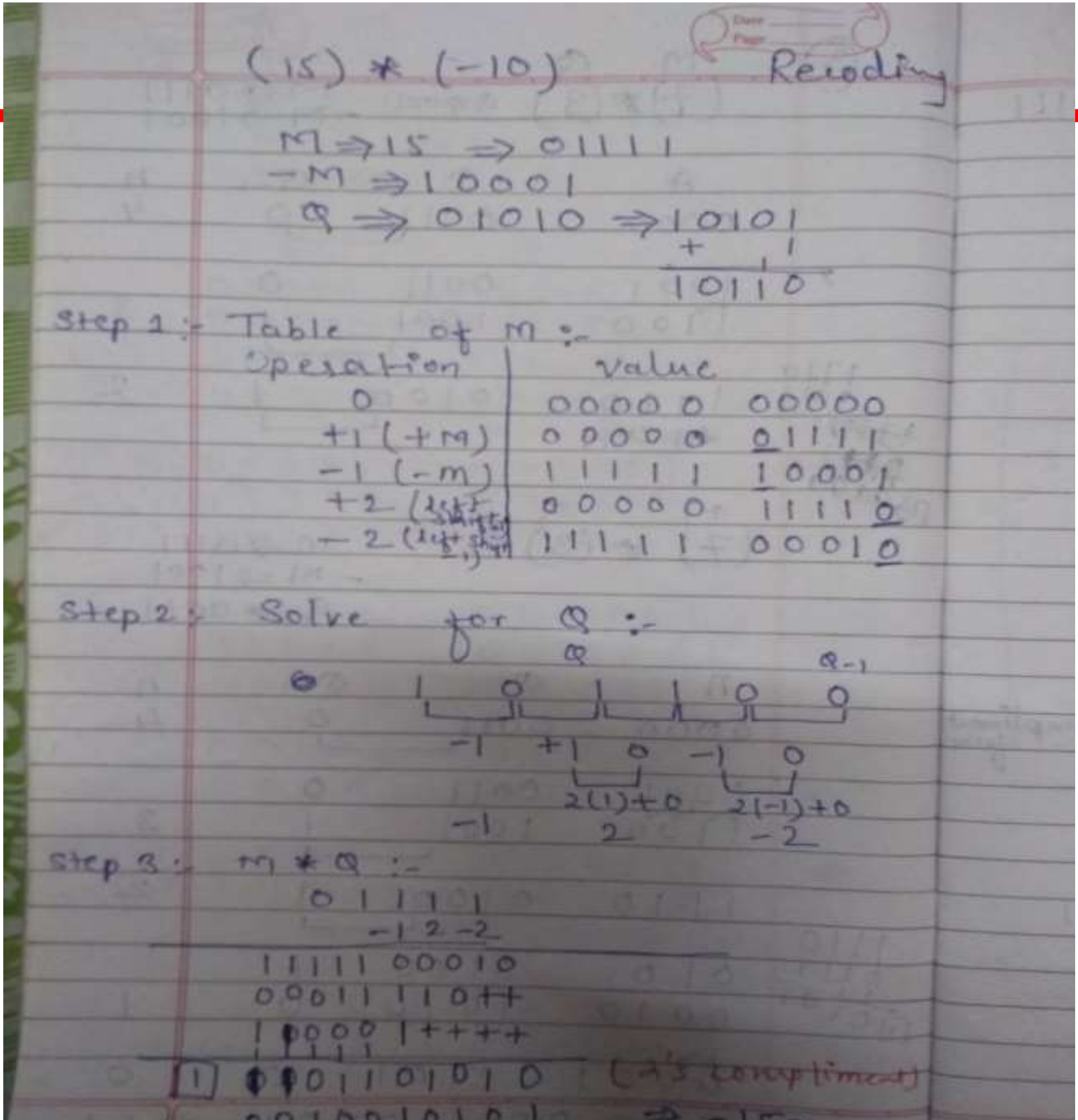
$$1 \qquad\qquad 0$$

1 0

③ <u>M * Q</u> :-

$$\begin{array}{cccccccc}
 & & & & 0 & 1 & 0 & 1 \\
 & & & & & & 1 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & \times & \times \\
\hline
0 & 0 & 0 & \underset{2^4}{1} & 0 & \underset{2^2}{1} & 0 & 0
\end{array}$$

$\Rightarrow 16 + 4 \Rightarrow 20$



(15) * (-10) Recoding

M ⇒ 15 ⇒ 01111

-M ⇒ 10001

Q ⇒ 01010 ⇒ 10101

+ 1

10110

Step 1 :- Table of M :-

Operation	Value	
0	00000	00000
+1 (+M)	00000	01111
-1 (-M)	11111	10001
+2 (left shifted)	00000	11110
-2 (left shift -1)	11111	00010

Step 2 :- Solve for Q :-

Q Q-1

1 0 1 1 0 0

-1 +1 0 -1 0

2(1)+0 2(-1)+0

-1 2 -2

Step 3 : M * Q :-

```
        0 1 1 1 1
          -1 2 -2
 -------------------
 1 1 1 1 1 0 0 0 1 0
 0 0 0 1 1 1 1 0 + +
 1 0 0 0 0 1 + + + +
 -------------------
[1] 0 0 0 1 1 0 1 0 1 0   (2's compliment)
```

Booths Recoding / Bit pair recoding

- Solve Booth’s Recoding algorithm

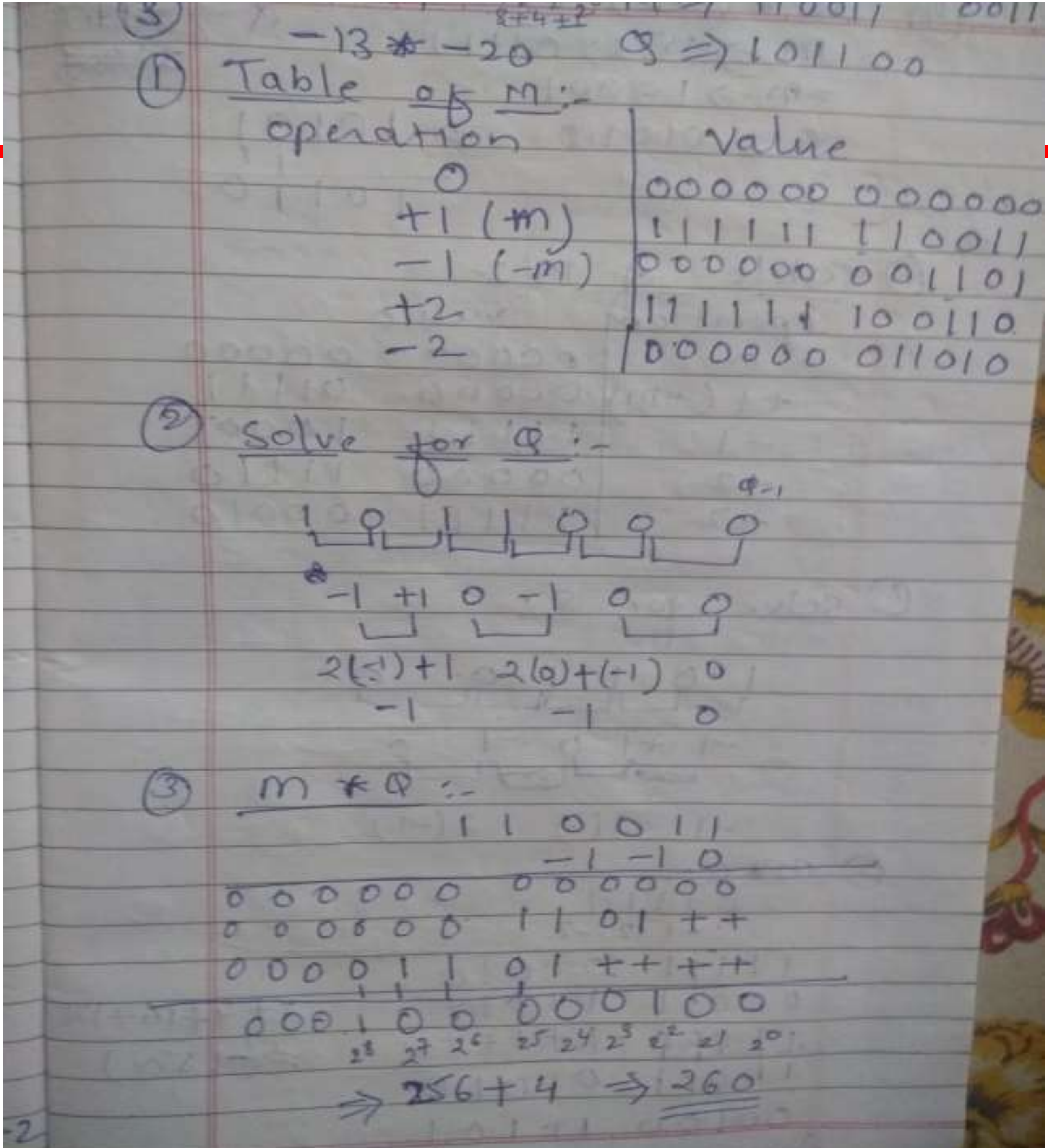
1. M = 9 , Q = -6   (take 5 bits)

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



8+4+2 110011 0011

−13 * −20   Q ⇒ 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
1   0   1   1   0   0          0

*-1  +1   0  -1    0    0

2(-1)+1   2(0)+(-1)   0
  -1         -1       0
```

③ M * Q :-

```
                    1 1 0 0 1 1
                      -1 -1 0
0 0 0 0 0 0   0 0 0 0 0 0
0 0 0 0 0 0   1 1 0 1 + +
0 0 0 0 1 1   0 1 + + + +
      1 1 1  1
 0 0 0 1 0 0   0 0 0 1 0 0
       2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
```

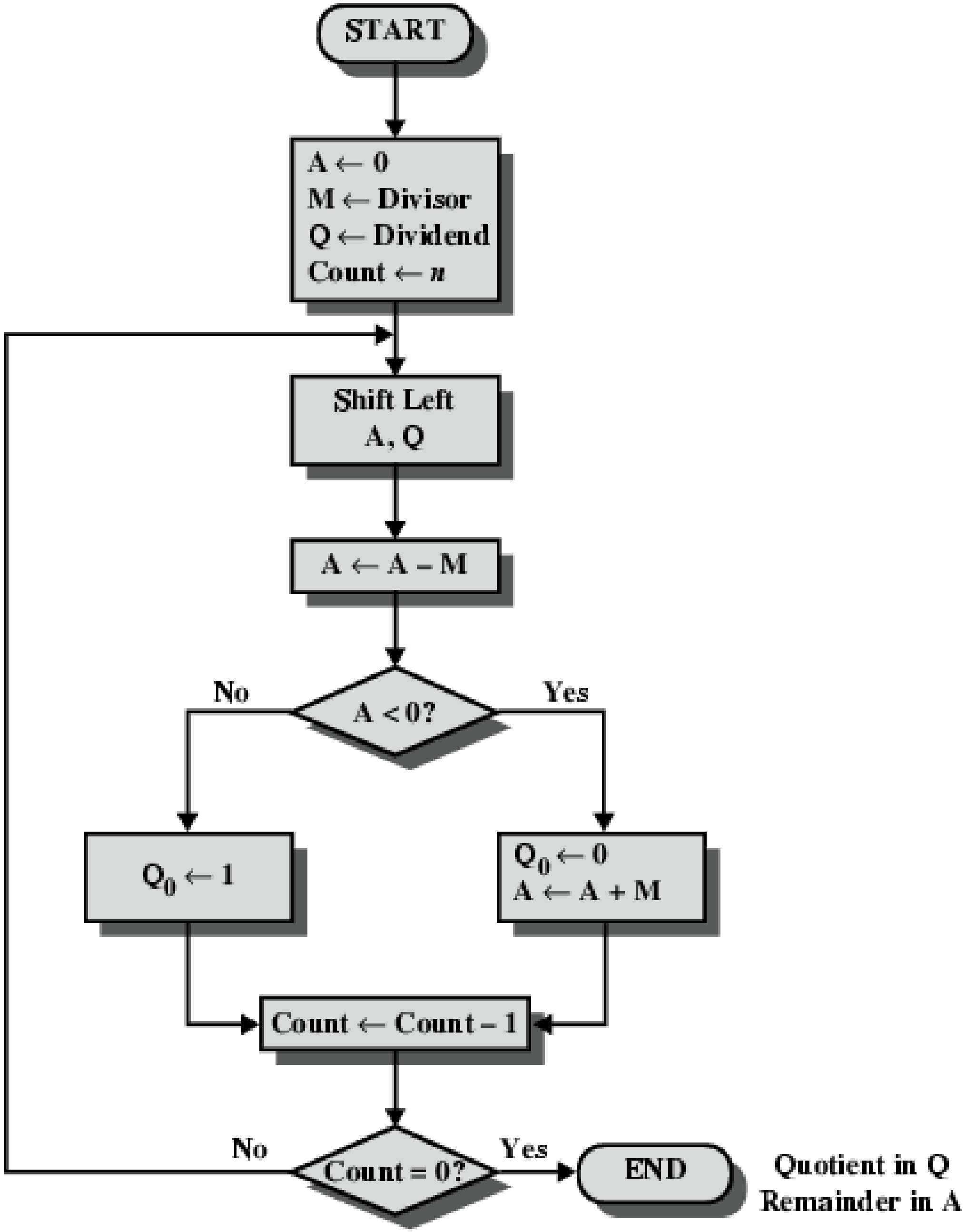
⇒ 256 + 4 ⇒ 260

-2

Division

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

Flowchart for Restoring Division



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

$M = 0011$

$-M = 1101$

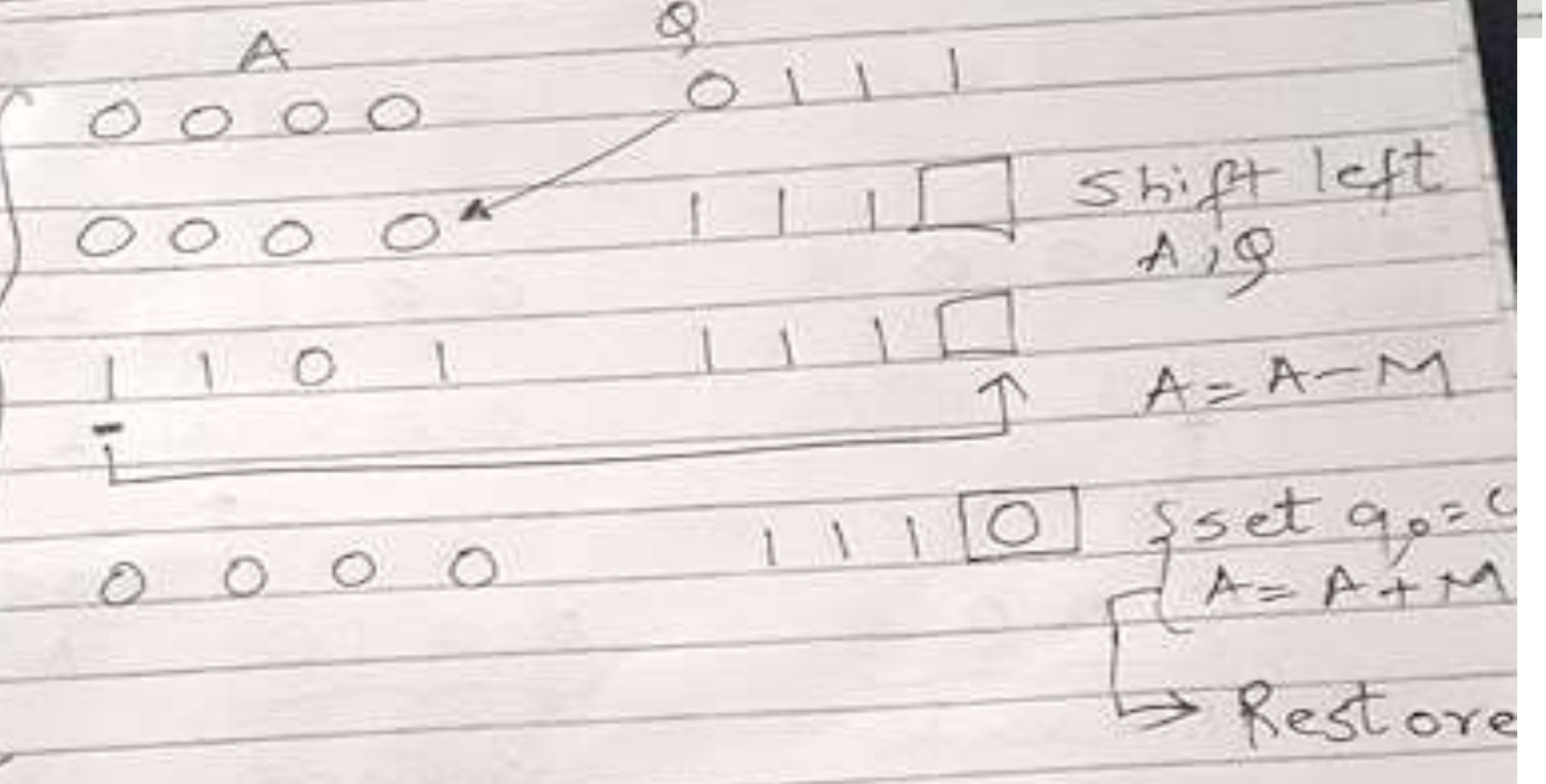
$Q = 0111$

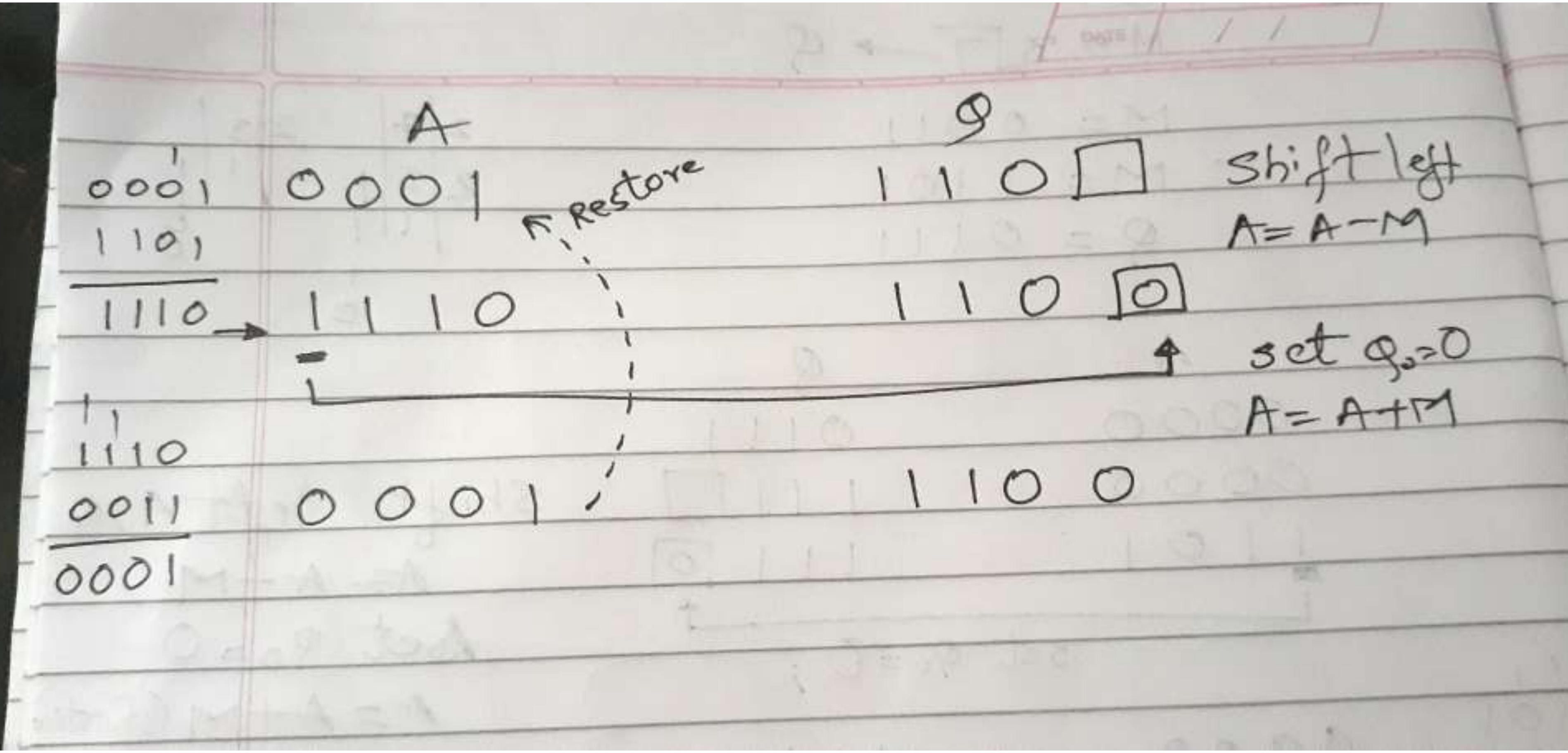
2 | 7 |
2 | 3 | 1
  | 1 | 1
      1

2 | 3 |
  | 1 | 1
      1
      0
      0

1
$A = 1101$
$M = 0011$
1 0000

cycle 1





	A	Q	
0001	0 0 0 1	1 1 0 □	shift left
1101			A=A−M
1110	1 1 1 0	1 1 0 0	
	−		set Q_0=0
1110			A=A+M
0011	0 0 0 1	1 1 0 0	
0001			

Restore

0001

111 0011	0 0 1 1	1 0 0 □	shift left
1101 0000	0 0 0 0	1 0 0 $\boxed{1}$	A = A − M
			set q₀ = 1
	0 0 0 0	1 0 0 1	
0001	0 0 0 1	0 0 1 □	shift left
1101 1110	1 1 1 0	0 0 1 $\boxed{0}$	A = A − M
			set Q₀ = 0 A = A + M
	0 0 0 1	0 0 1 0	

$\underbrace{0\ 0\ 0\ 1}_{\text{Remainder}}$ = 1

$\underbrace{0\ 0\ 1\ 0}_{\text{Quotient}}$ = 2

$M = 27$; $Q = 55$

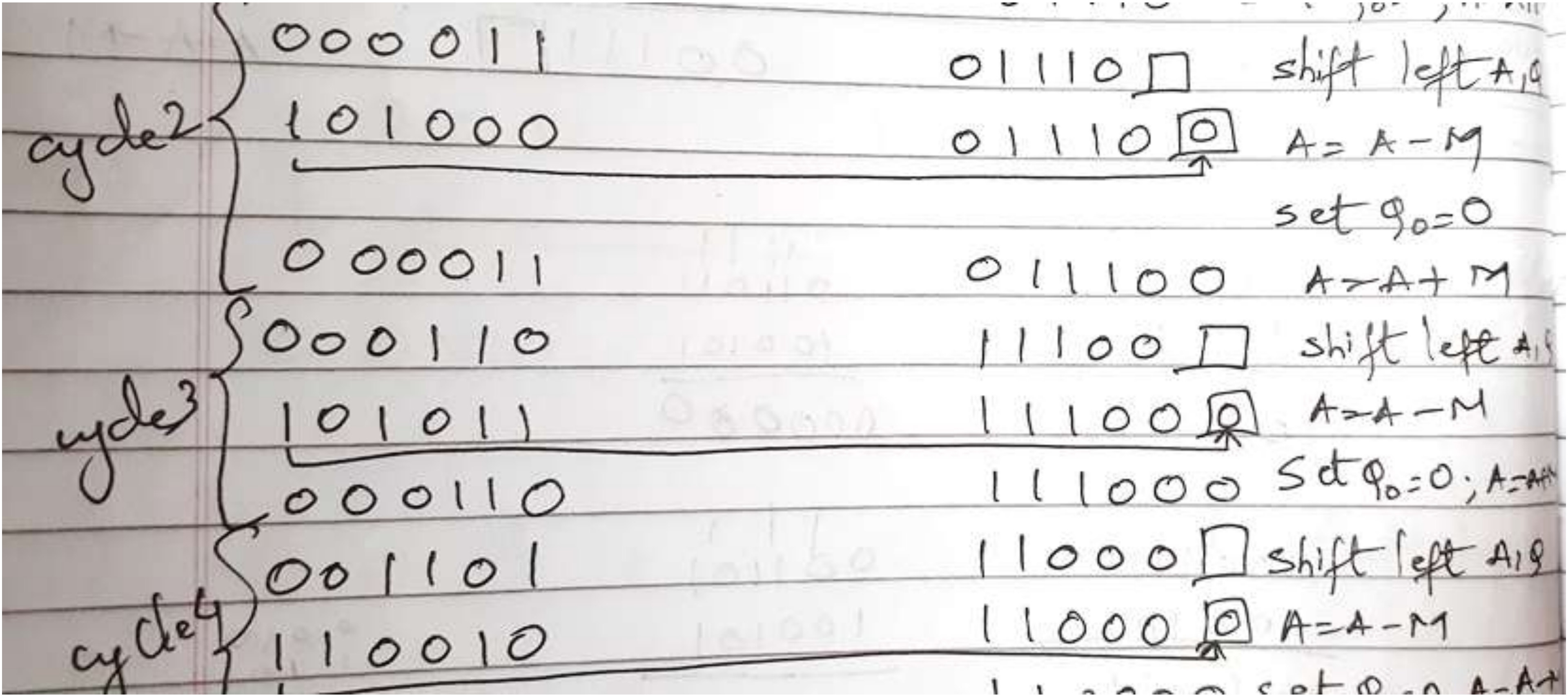
$M = 011011$

$-M = 100101$

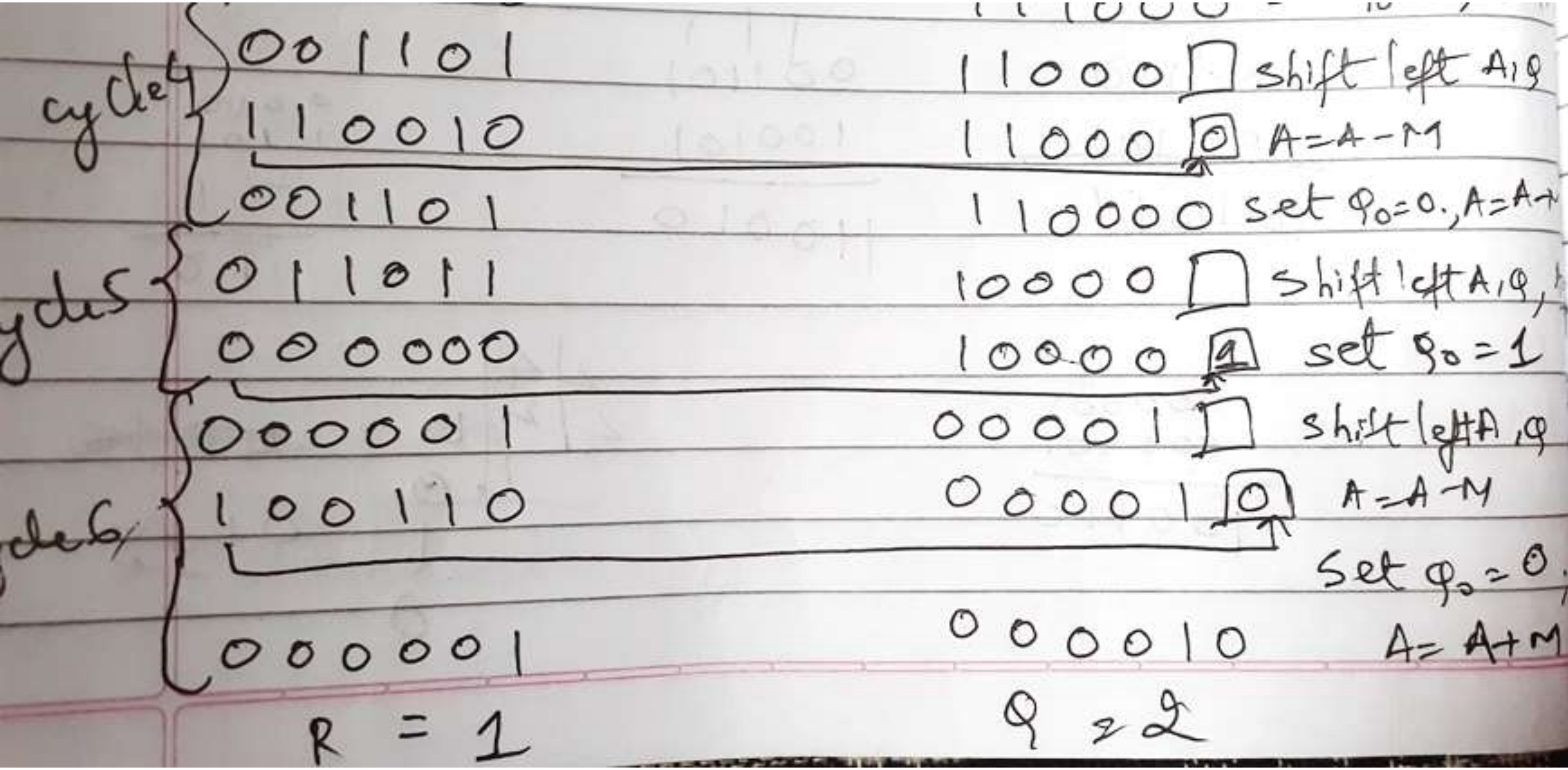
$Q = 110111$

de1

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

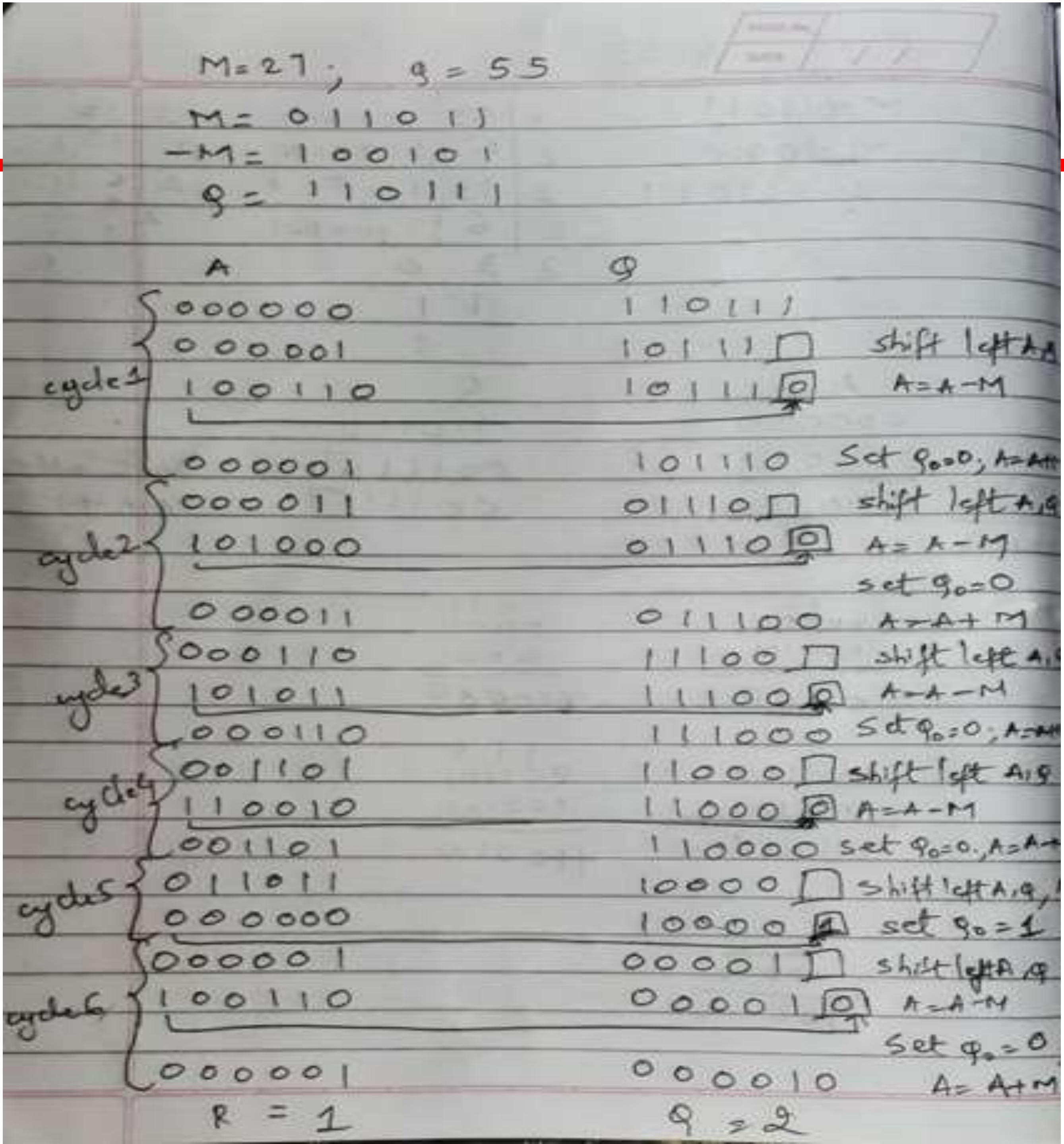


cycle2	000011	01110□	shift left A,Q
	101000	01110$\boxed{0}$	A= A−M
			set Q_0=0
	000011	011100	A=A+M
cycle3	000110	11100□	shift left A,Q
	101011	11100$\boxed{0}$	A=A−M
	000110	111000	Set Q_0=0; A=A+M
cycle4	001101	11000□	shift left A,Q
	110010	11000$\boxed{0}$	A=A−M
		110000	set Q_0=0 A=A+



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

	A	Q	
cycle1	000000	110111	
	000001	10111 □	shift left A,Q
	100110	10111 $\boxed{0}$	A=A-M
	000001	101110	Set q_0=0; A=A+
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+
cycle4	001101	11000 □	shift left A,Q
	110010	11000 $\boxed{0}$	A=A-M
	001101	110000	set q_0=0., A=A+
cycle5	011011	10000 □	shift left A,Q,
	000000	10000 $\boxed{1}$	set q_0=1
cycle6	000001	00001 □	shift left A,Q
	100110	00001 $\boxed{0}$	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

B. M = 12, Q = 26

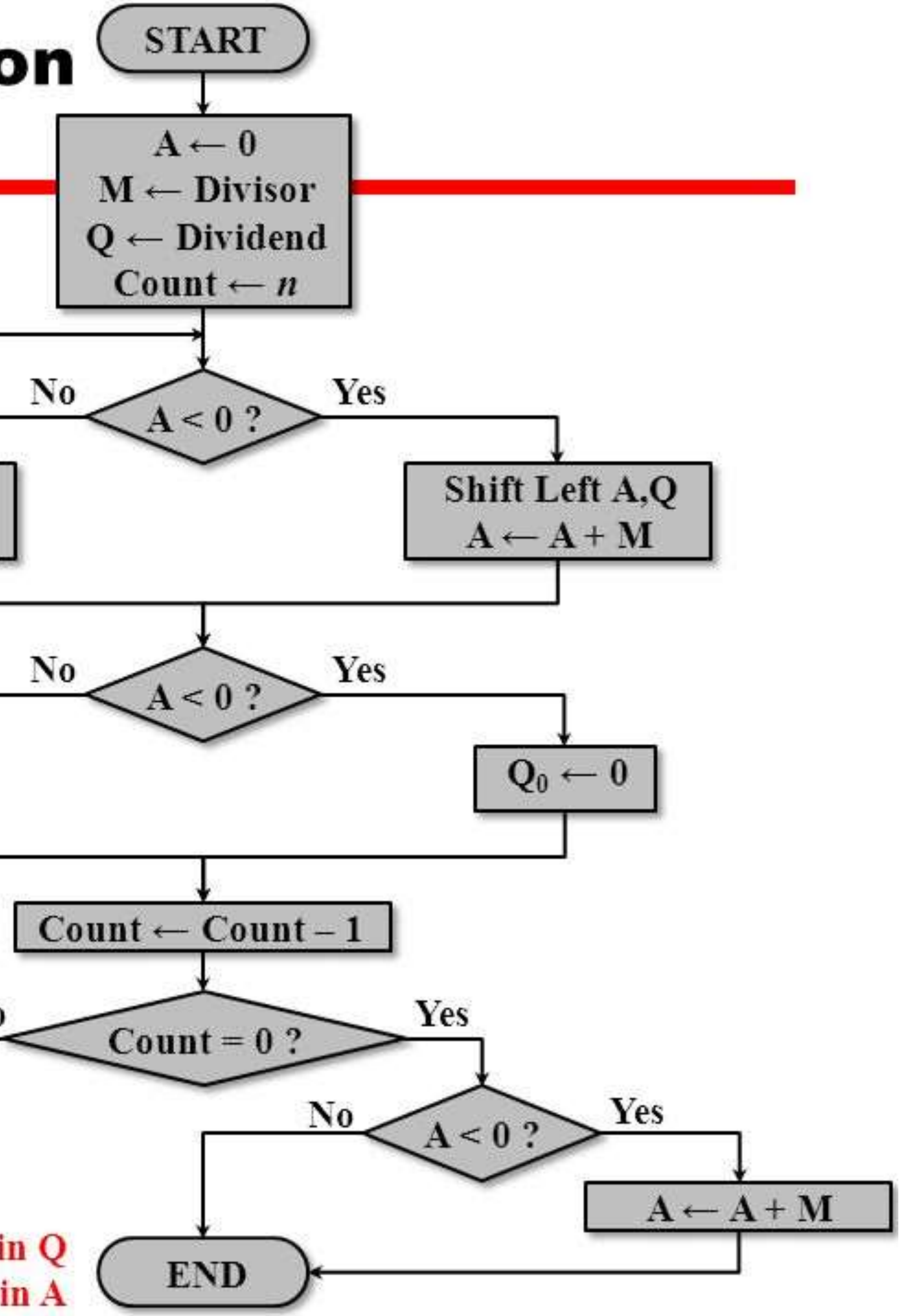
C. M = 9, Q = 19

D. M = 32 , Q = 59

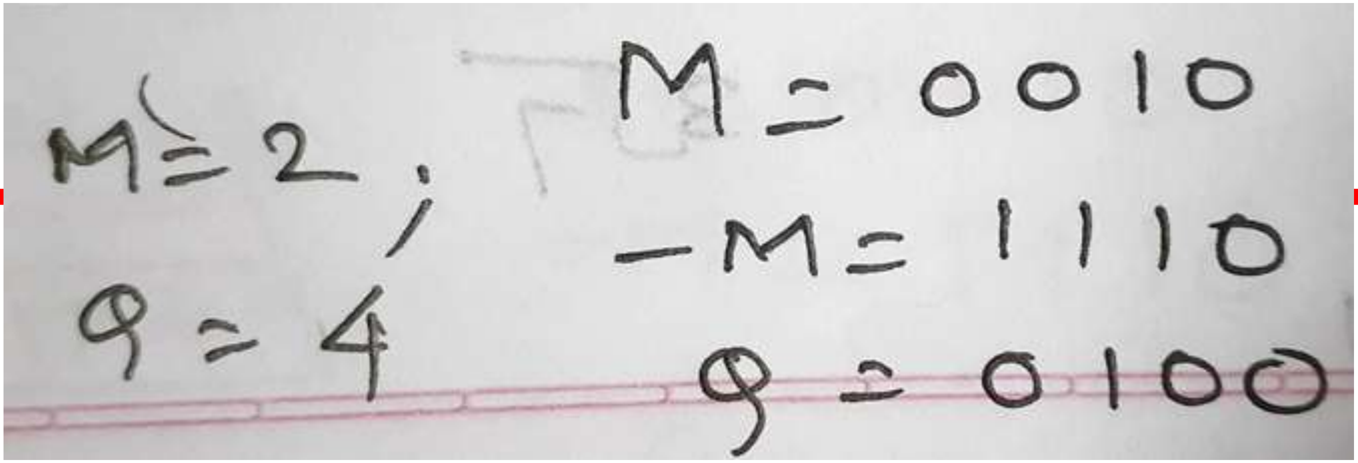
E. M=27 , Q=55

F. M = 17 , Q = 42

Non-Restoring Division (Positive Integers)



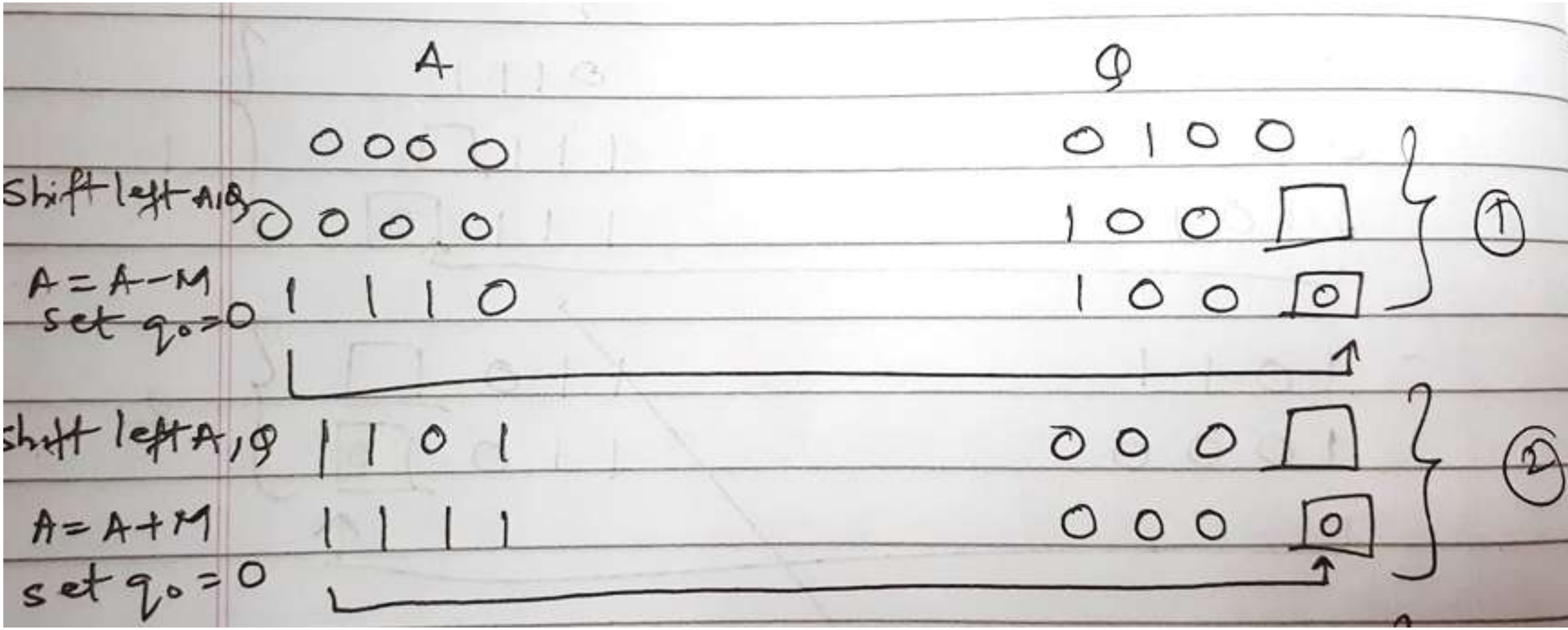
Quotient in Q
Remainder in A



$M = 2$; $M = 0010$

$Q = 4$ $\quad -M = 1110$

$Q = 0100$



	A	Q	
	0 0 0 0	0 1 0 0	①
Shift left A,Q	0 0 0 0	1 0 0 □	
A = A − M set $q_0 = 0$	1 1 1 0	1 0 0 $\boxed{0}$	
shift left A,Q	1 1 0 1	0 0 0 □	②
A = A + M set $q_0 = 0$	1 1 1 1	0 0 0 $\boxed{0}$	

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

```
  11
  1110 (A)
  0010 (M)
~~1~~ 0000
```

$\underbrace{0\ 0\ 0\ 0}_{}$ $\underbrace{0\ 0\ 1\ 0}_{}$

A = Remainder Q = Quotient

Solve using Non Restoring

A. M = 5 , Q = 5

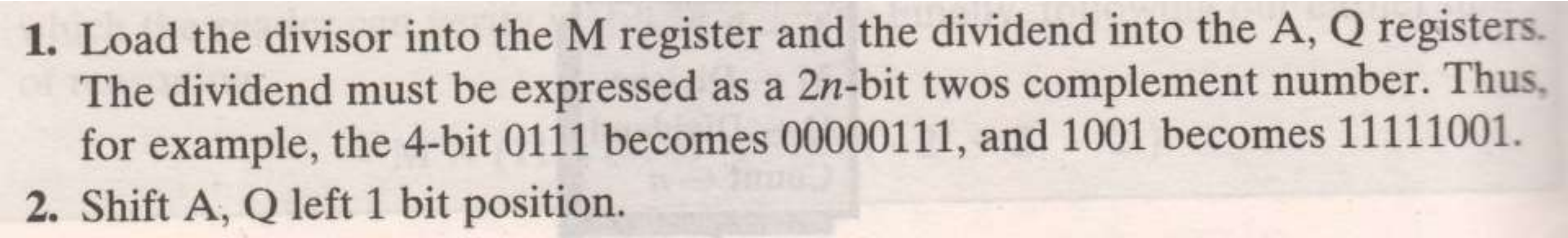
B. M = 12, Q = 26

C. M = 9, Q = 19

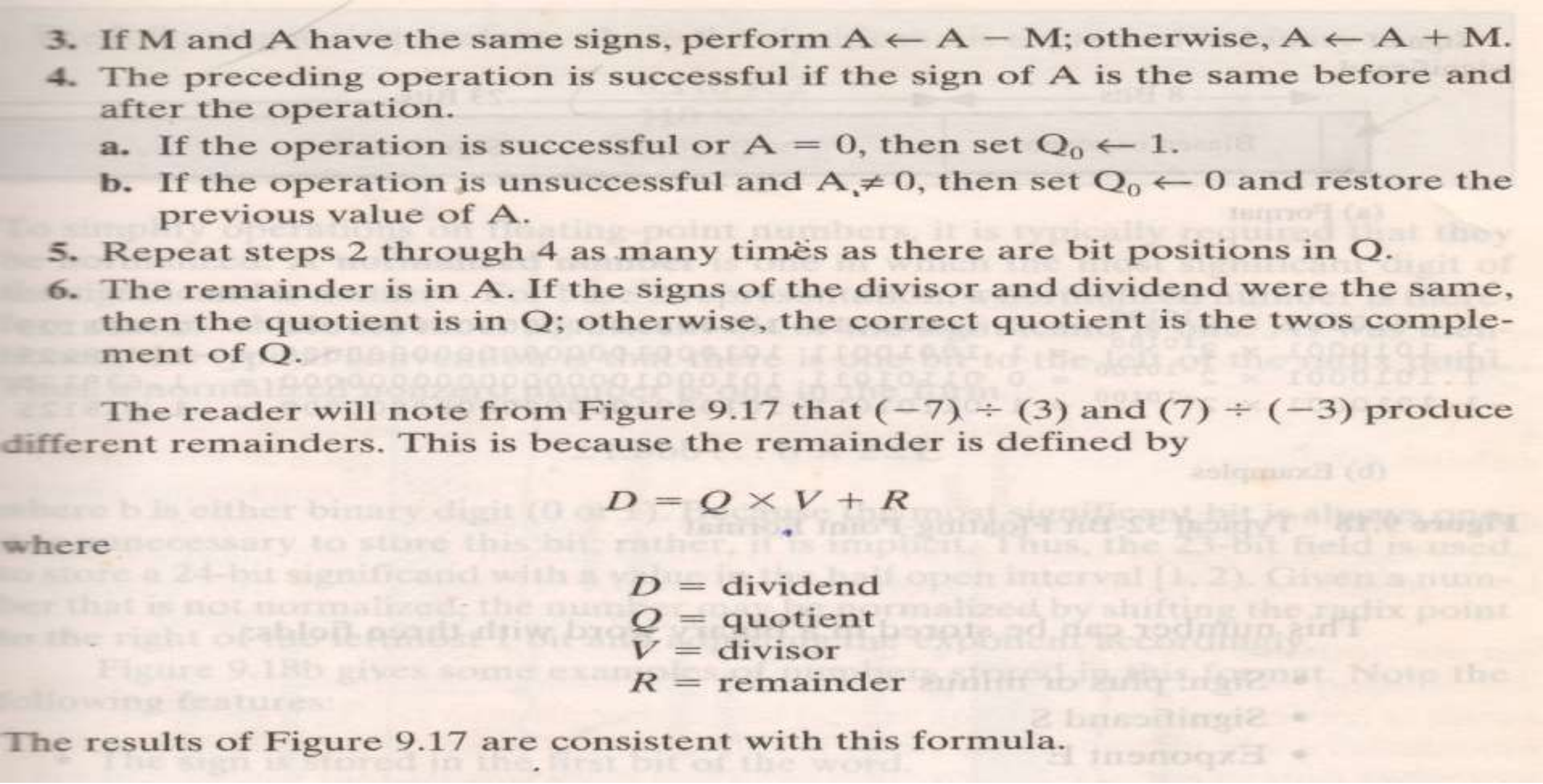
D. M = 32 , Q = 59

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 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.

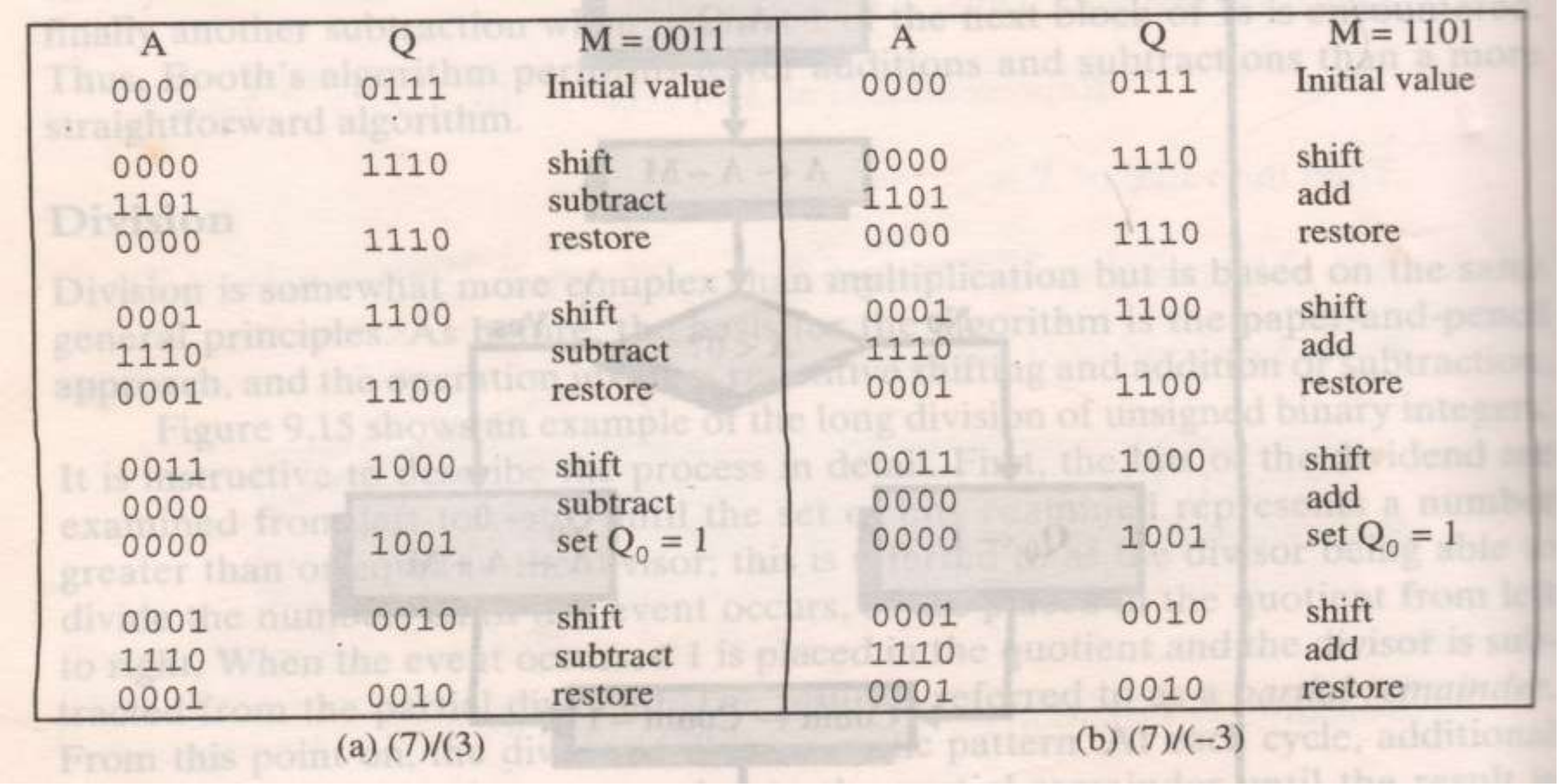
The reader will note from Figure 9.17 that $(-7) \div (3)$ and $(7) \div (-3)$ produce different remainders. This is because the remainder is defined by

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

where

D = dividend
Q = quotient
V = divisor
R = remainder

The results of Figure 9.17 are consistent with this formula.

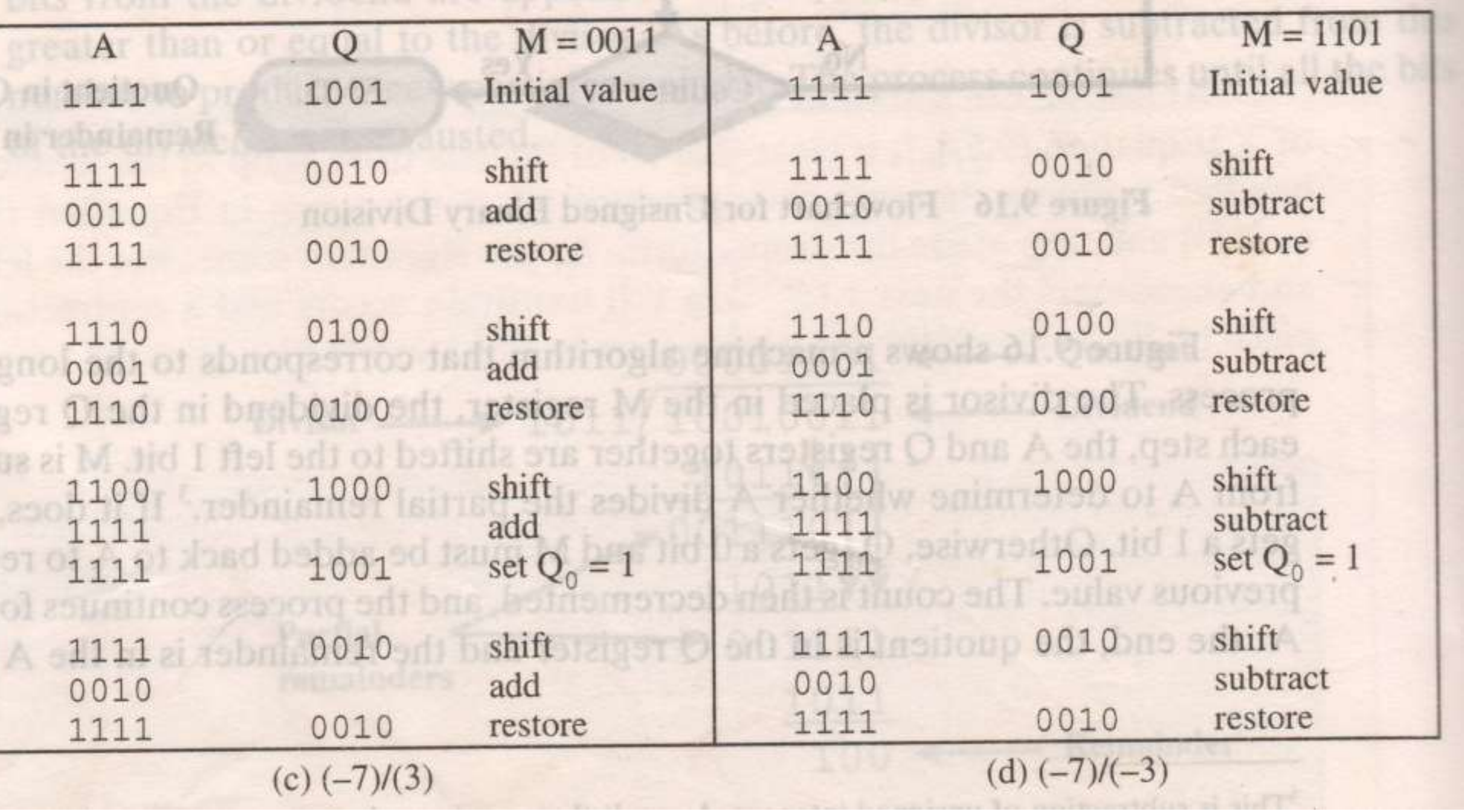


A	Q	M = 0011
0000	0111	Initial value
0000	1110	shift
1101		subtract
0000	1110	restore
0001	1100	shift
1110		subtract
0001	1100	restore
0011	1000	shift
0000		subtract
0000	1001	set $Q_0 = 1$
0001	0010	shift
1110		subtract
0001	0010	restore

(a) (7)/(3)

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	restore

(c) (–7)/(3)

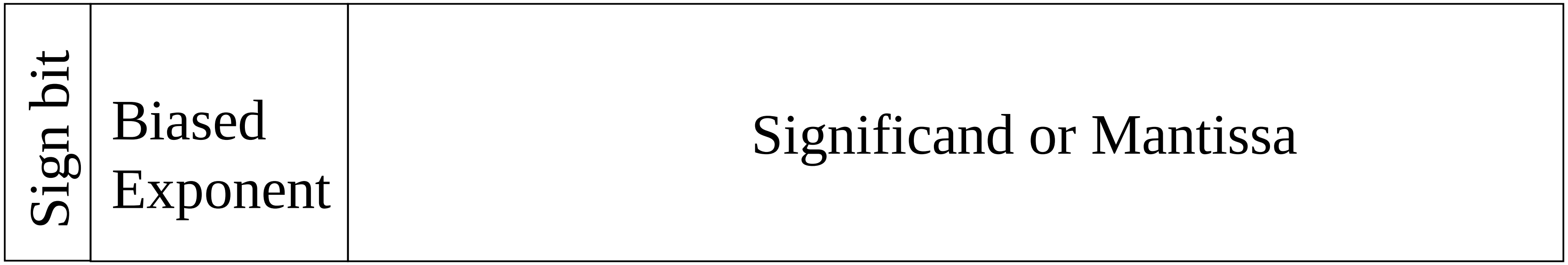
A	Q	M = 1101
1111	1001	Initial value
1111	0010	shift
0010		subtract
1111	0010	restore
1110	0100	shift
0001		subtract
1110	0100	restore
1100	1000	shift
1111		subtract
1111	1001	set $Q_0 = 1$
1111	0010	shift
0010		subtract
1111	0010	restore

(d) (–7)/(–3)

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



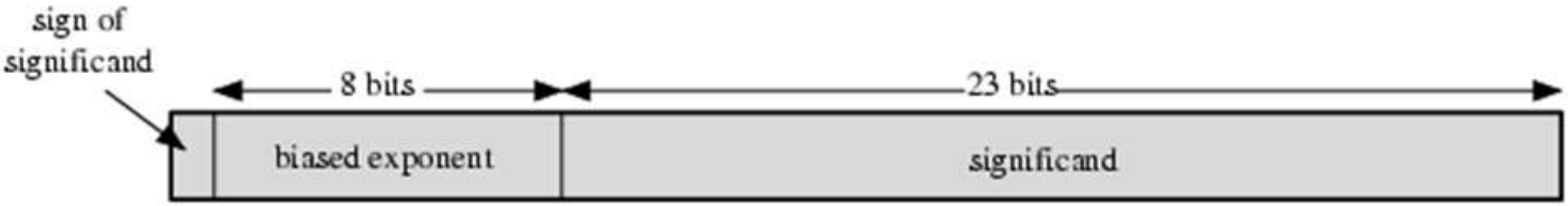
Sign bit

Biased Exponent

Significand or Mantissa

- +/- .significand x 2^{exponent}
- Point is actually fixed between sign bit and body of mantissa
- Exponent indicates place value (point position)
- E.g. 2.5*2^5

Floating Point Examples



(a) Format

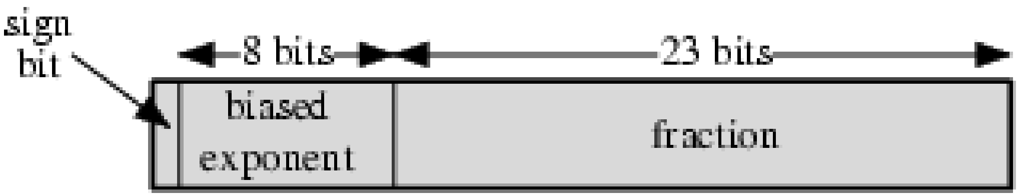
Signs for Floating Point

- 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

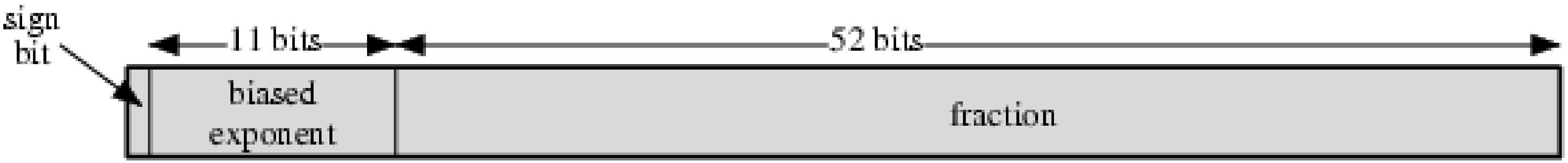
IEEE 754 Formats



(a) Single format

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

32 BIT



(b) Double format

64 BIT

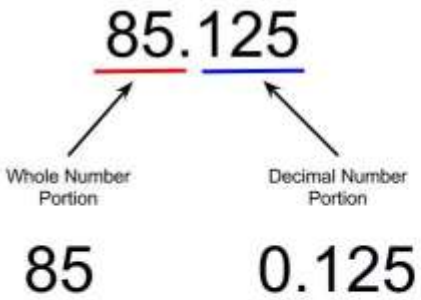
$(1.N)2^{E-1023}$

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



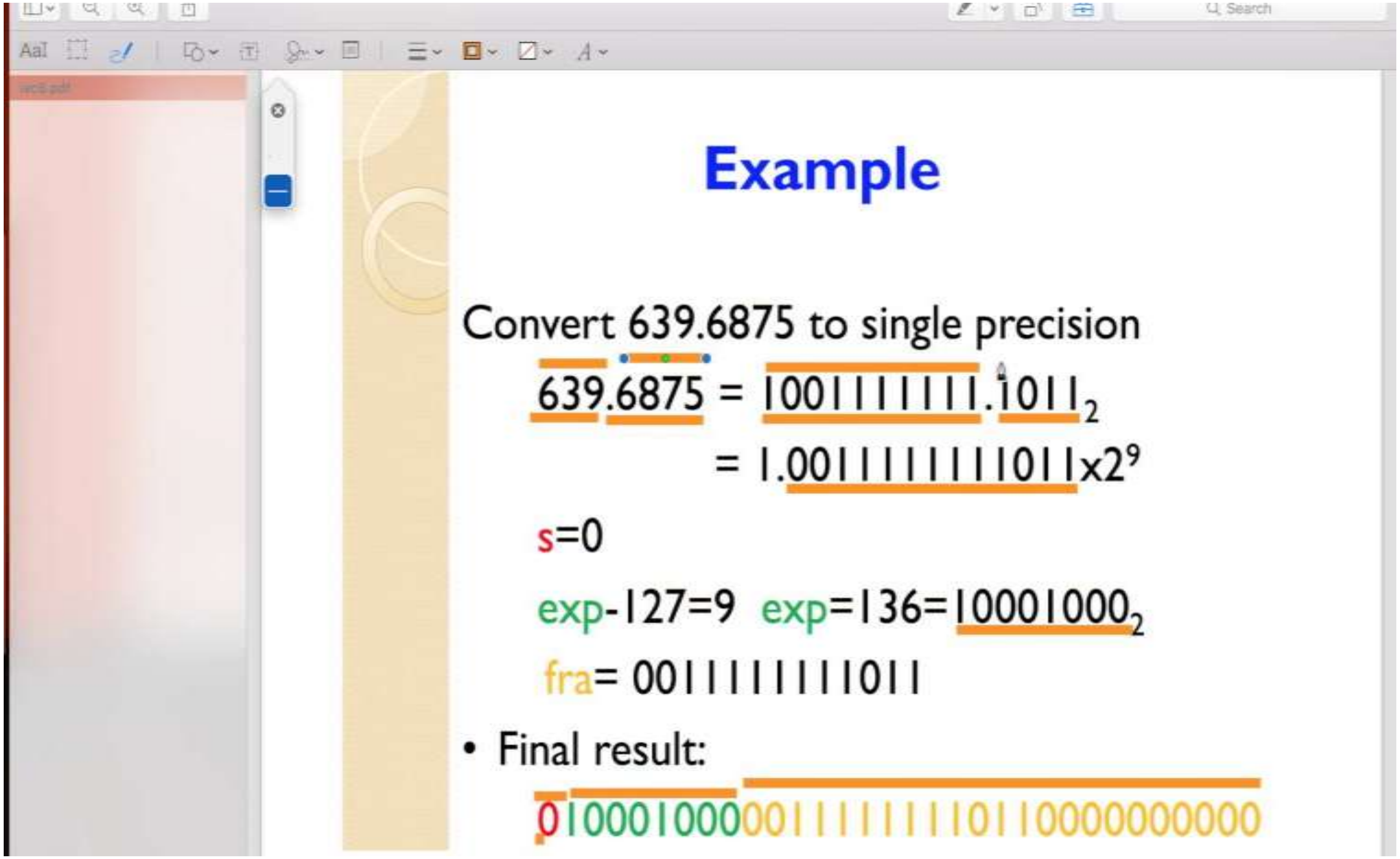
85.125

Whole Number Portion

Decimal Number Portion

85

0.125



Example

Convert 639.6875 to single precision

$639.6875 = 1001111111.1011_2$

$= 1.0011111111011 \text{x} 2^9$

s=0

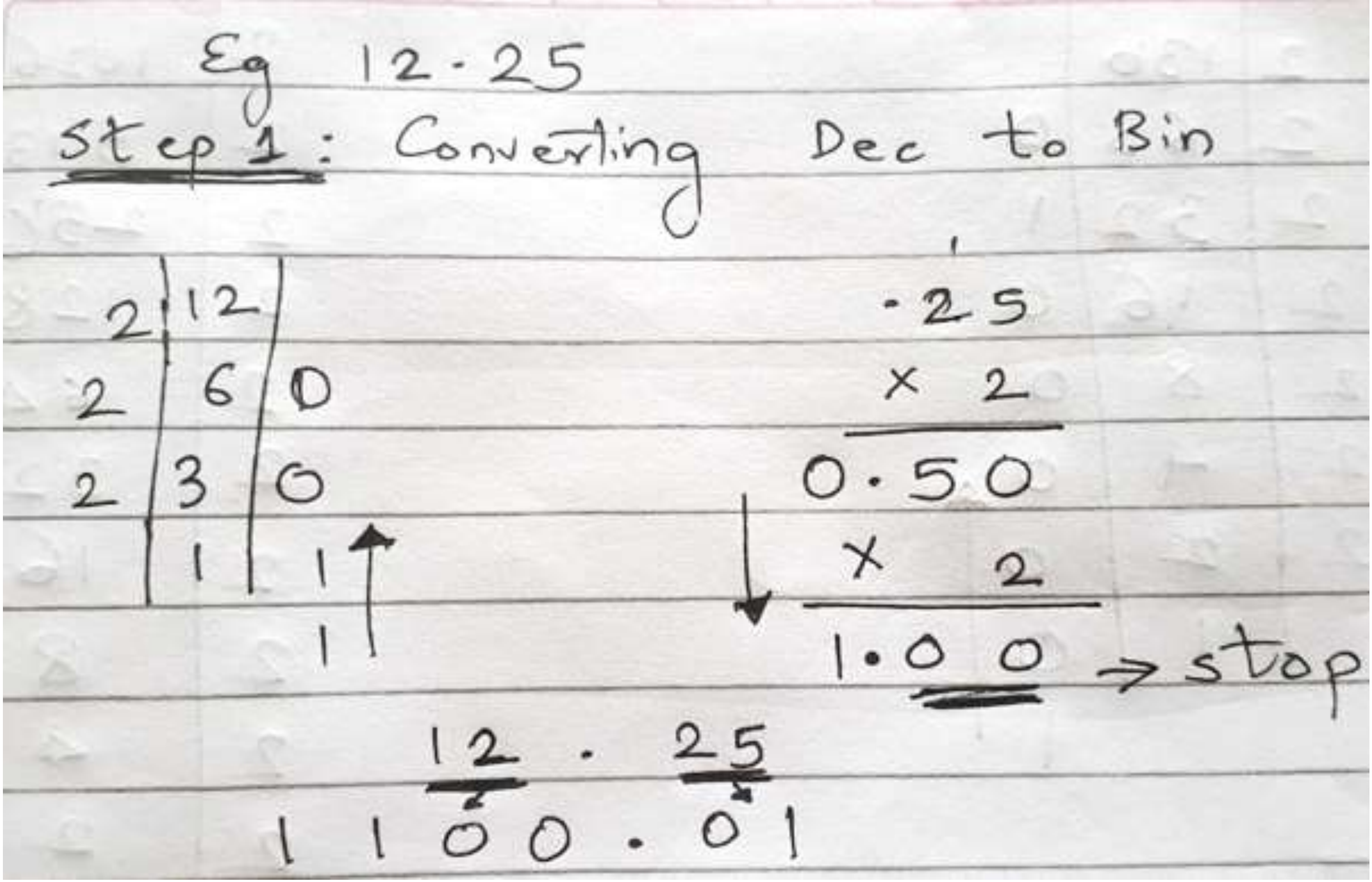
exp-127=9 exp=136=10001000_2

fra= 0011111111011

- Final result:

01000100000111111110110000000000

Solved Example



Eg 12.25

Step 1: Converting Dec to Bin

```
2 | 12 |
2 |  6 | 0
2 |  3 | 0
  |  1 | 1
         1
```

```
  .25
×   2
 0.50
×   2
 1.00 → stop
```

12 . 25

1100 . 01

Step 2: Normalization (1. N)

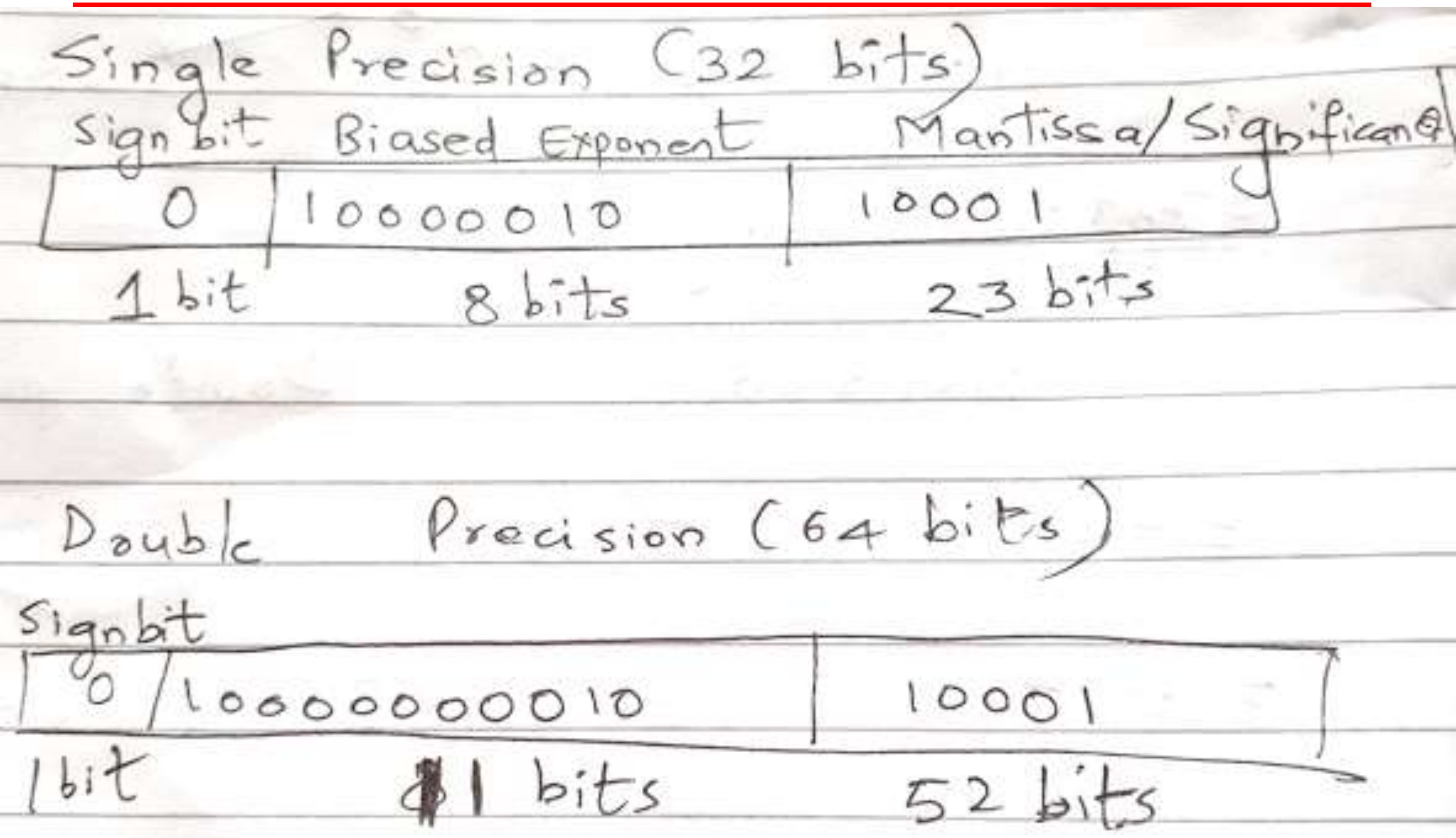
1.100001×2^{3} → Exponent

Step3: 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
1bit	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

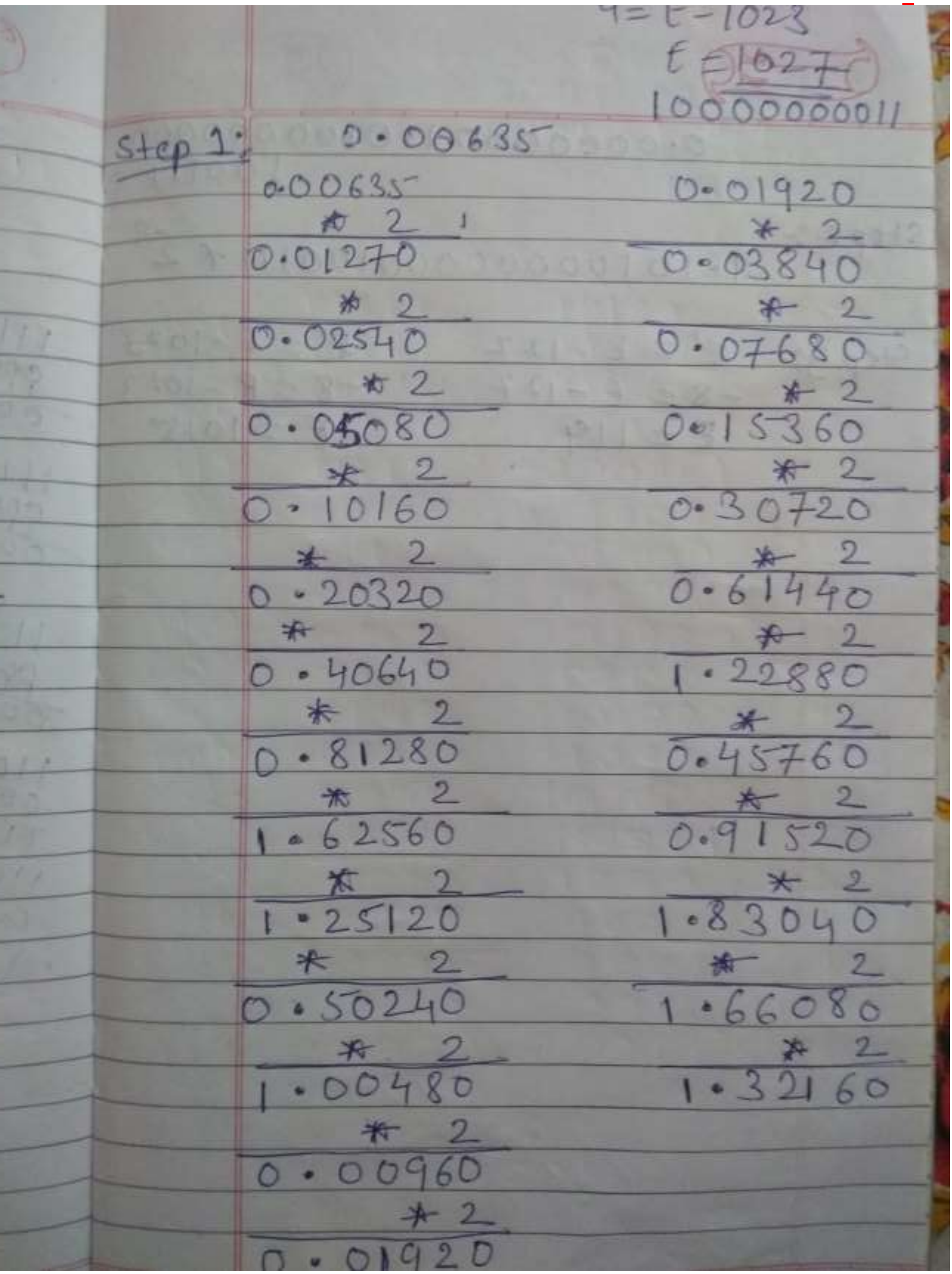
0.00635

-125.10

13.54   Single precision: 0 10000100 10110001010001111010111

Double precision: 0 10000000011 10110001010001111010111

Solve 0.00635



4=E-1023

E=1027

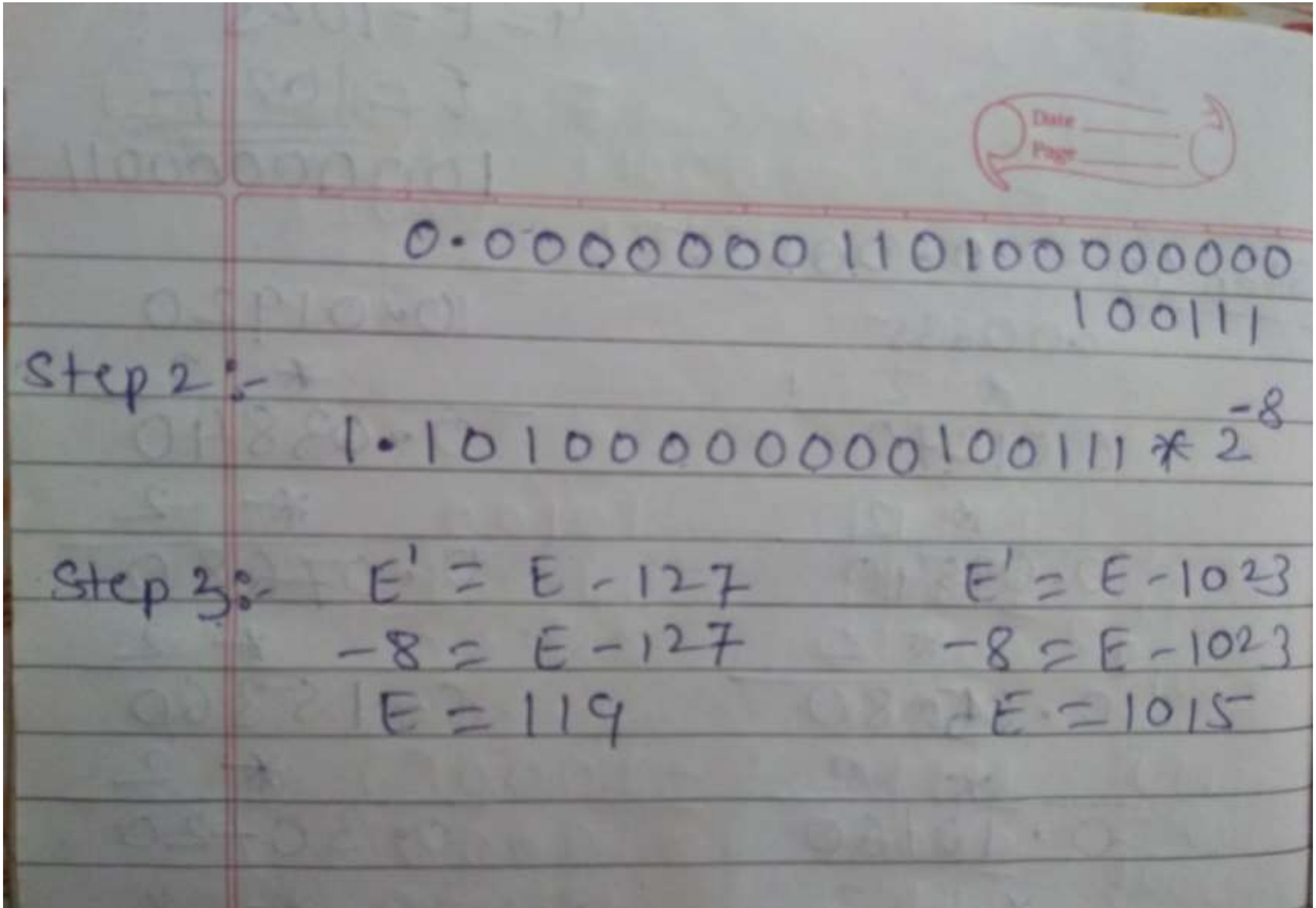
10000000011

Step 1: 0.00635

0.00635
* 2
0.01270
* 2
0.02540
* 2
0.05080
* 2
0.10160
* 2
0.20320
* 2
0.40640
* 2
0.81280
* 2
1.62560
* 2
1.25120
* 2
0.50240
* 2
1.00480
* 2
0.00960
* 2
0.01920

0.01920
* 2
0.03840
* 2
0.07680
* 2
0.15360
* 2
0.30720
* 2
0.61440
* 2
1.22880
* 2
0.45760
* 2
0.91520
* 2
1.83040
* 2
1.66080
* 2
1.32160

Solve 0.00635



Date
Page

$0.0000000\,11010000000$
100111

Step 2:-

$1.10100000000100111 * 2^{-8}$

Step 3:- $E' = E - 127$ $E' = E - 1023$

$-8 = E - 127$ $-8 = E - 1023$

$E = 119$ $E = 1015$

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:

8.70×10^{-1} with 9.95×10^{1}

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

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

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

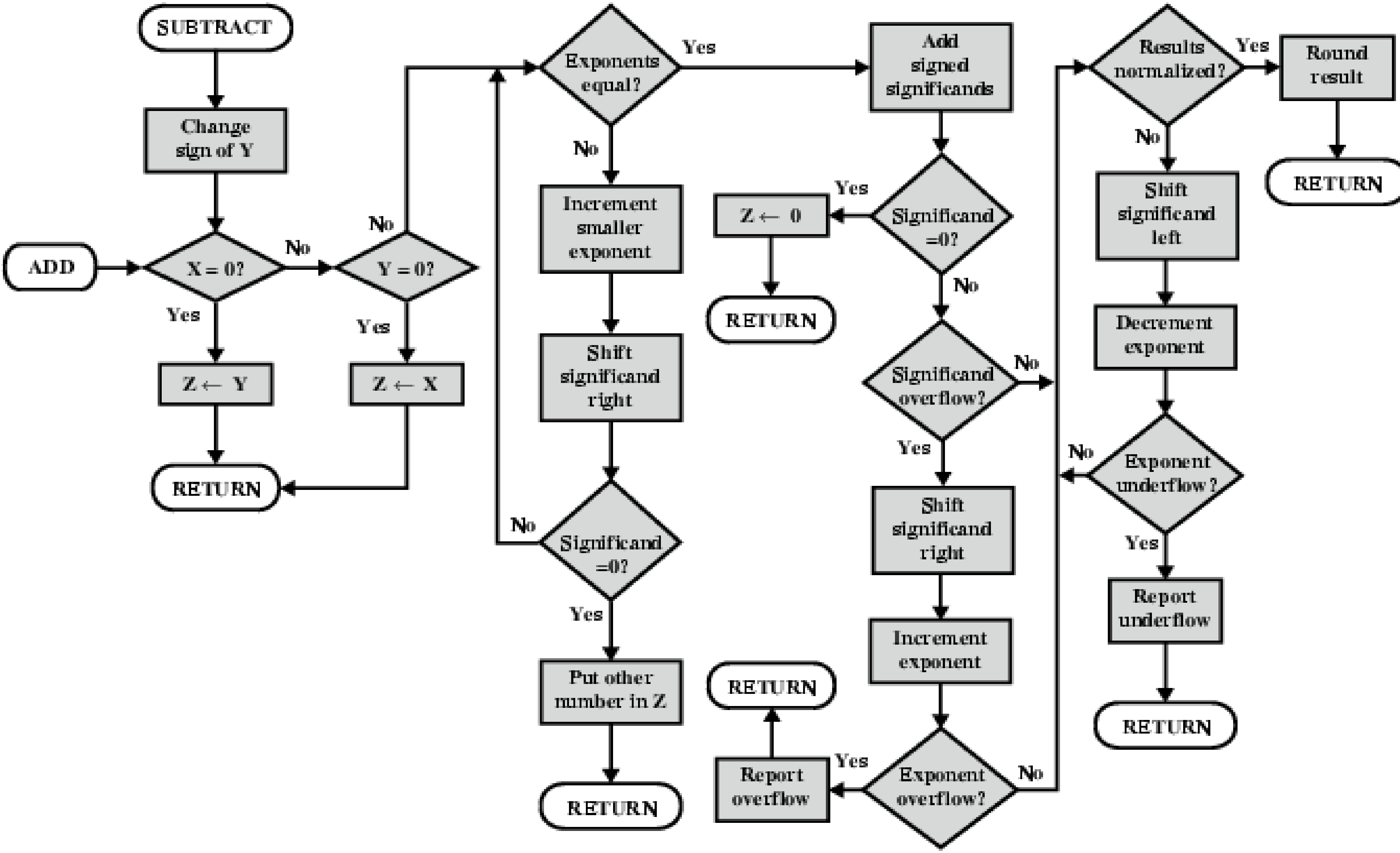
Round the result

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

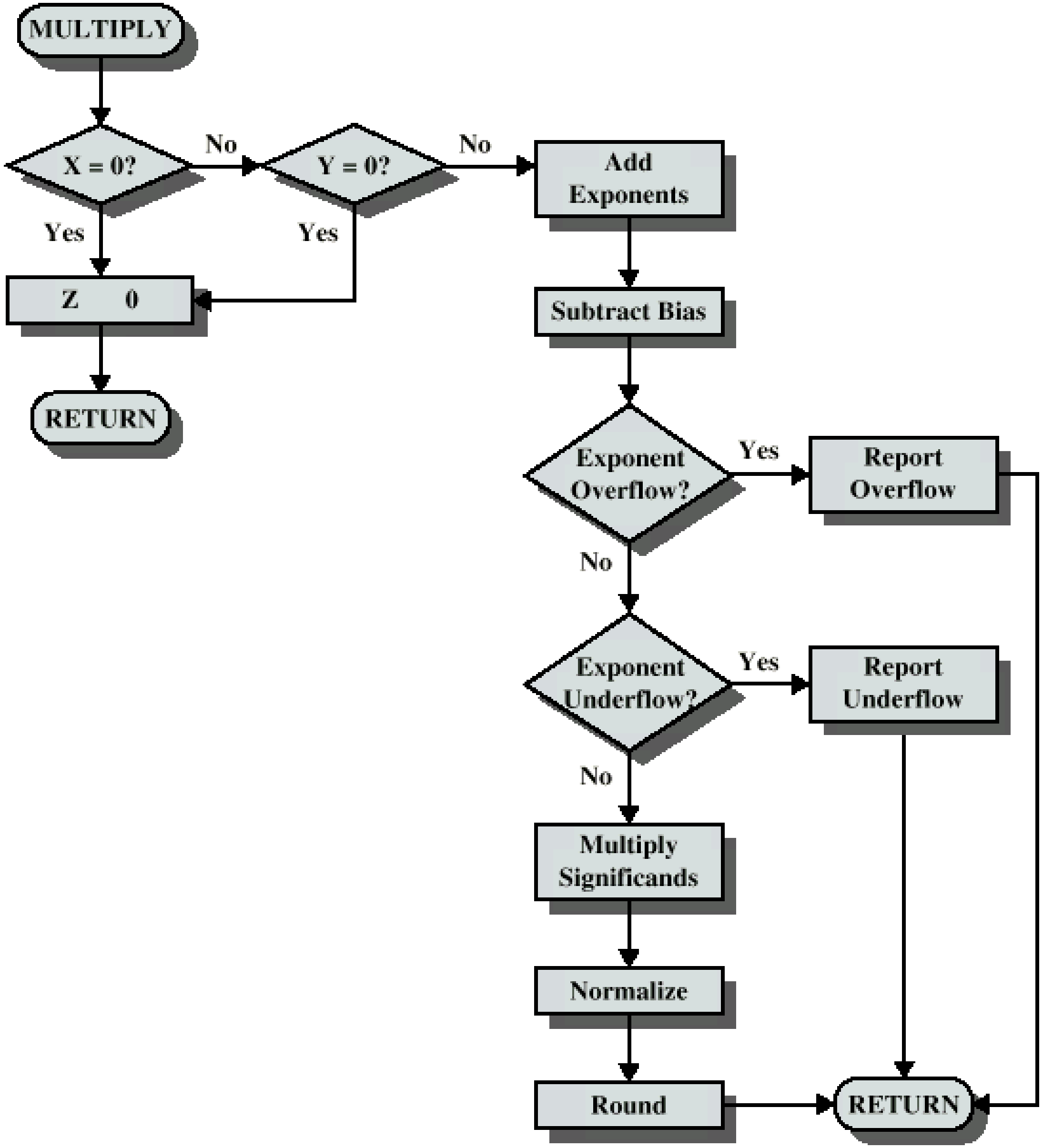
FP Addition & Subtraction Flowchart



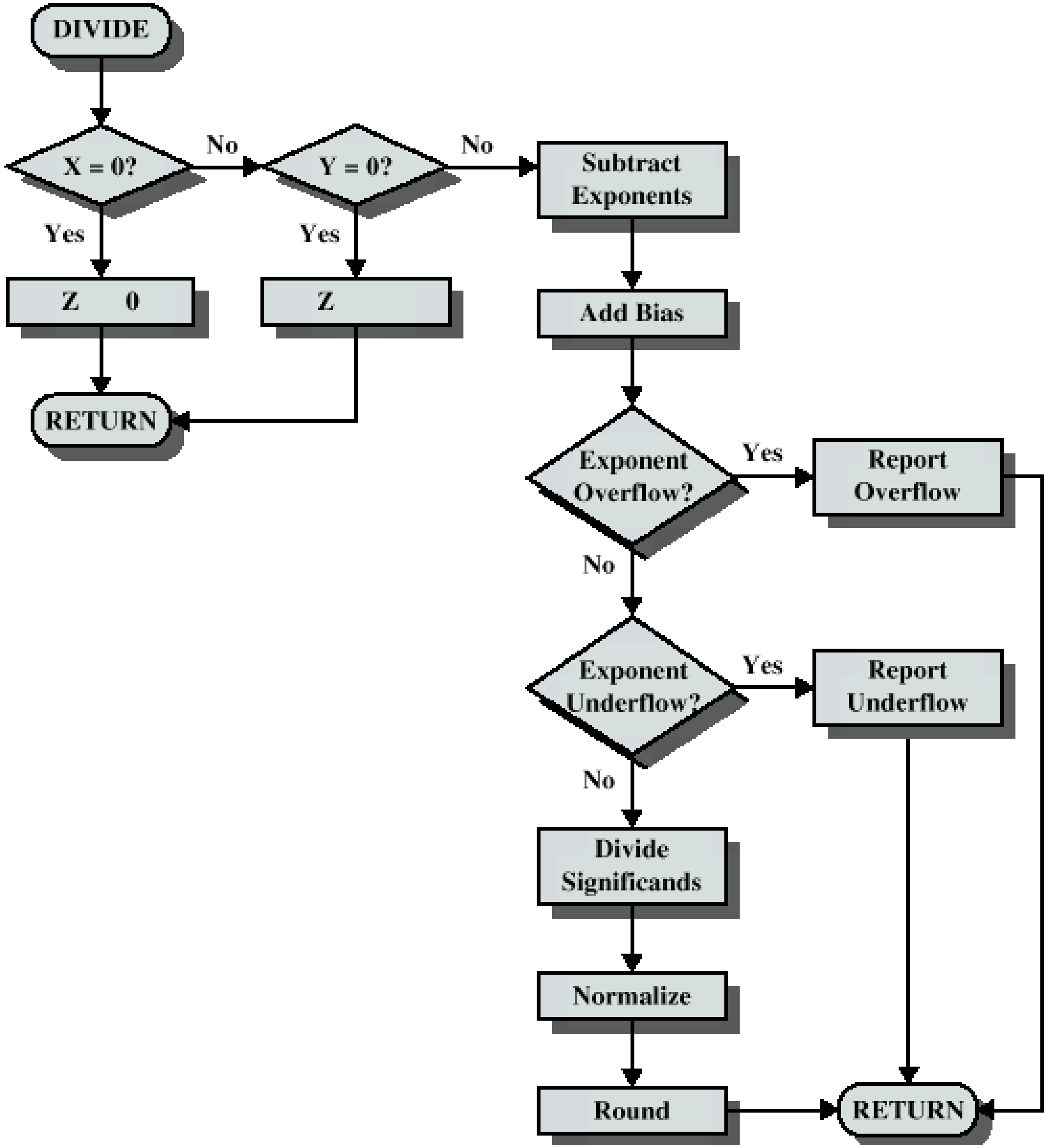
FP Arithmetic x/÷

- 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



Example addition in binary Perform 0.5 + (-0.4375)

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

$-0.4375 = -0.0111 \times 2^{0} = -1.110 \times 2^{-2}$ (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:

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

$-126 <= -4 <= 127$ ===> No overflow or underflow

Round the sum:

The sum fits in 4 bits so rounding is not required

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