Unit - 2

Number System

Number System

- Unsigned Numbers (+ve No)
  [5, 10, 23, 43, 46, etc]
- Signed Numbers (+ve/-ve No)
  [+5, -10, +23, -93, +46, etc]

* Magnitude (Absolute value of any number is the same number.

Signed Code | Binary Code

for +ve → 0

for -ve → 1

Operation	Value	Sign
(+A) + (+B)	(A + B)	+
(-A) + (+B)	(-A + B)	-
(+A) + (-B)	(A - B)	+
(-A) + (-B)	(A + B)	-

A > B

* There is actually no subtraction operation in computer. There are mainly 2 operations only. Addition & Multiplication.

	A	B	Sum	Carry
0 →	0	0	0	0
1 →	0	1	1	0
2 →	1	0	1	0
3 →	1	1	0	1

	A	B	C	Sum	Carry
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

* 1 + 1 + 1 = 1 1

Sum Carry.

Subtraction :-

A	B	Subtraction	Borrow.
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Not followed in computer.
for -ve No.
↓
Take use 2's complement.

* Whenever we see negative no. we first find 2's complement.

Eg. (-7)

$\Rightarrow$ 0 1 1 1 ← +7

Sign Code

1 0 0 0 ← 1's complement

\+ 1

1 0 0 1 ← 2's complement

So, (-7) = 1001

Sign Code

-7 (actual magnitude).

Addition & Subtraction of Signed No.

(1) Both No. +ve $(+A) + (+B) \quad +(A+B)$

(2) Greater No. +ve $(+A) + (-B) \quad +(A-B)$

(3) Greater No. -ve $(-A) + (+B) \quad -(A-B)$

(4) Both No. -ve $(-A) + (-B) \quad -(A+B)$

(1) Case I :- Both No. +ve

$$(+A) + (+B) = +(A+B)$$

Q Add $(+23)_{10} + (+15)_{10}$.

Sol →

$+23 = 0\ 10111$

$+15 = \boxed{0}\ 01111$

Extending the signed digit to make equal bit.
If sign digit is 1 then extend by putting 1 !

$$\begin{array}{rcl} +23 & = & 010111 \\ +\ +15 & = & +\ 001111 \\ \hline +38 & & 100110 \end{array}$$

Case II :- Greater No +ve

$$(+A) + (-B) = +(A-B).$$

Q Add $(+23)_{10} + (-15)_{10}$.

$+23 = 0\,1\,0\,1\,1\,1$

~~$+15 = 1\,0\,0\,0\,1$~~

$+15 = 0\,1\,1\,1\,1$

1's complement = 10000

2's Complement = 10001

So, $-15 = 10001$

$+23 = 0\,1\,0\,1\,1\,1$

$-15 = \boxed{1}\,1\,0\,0\,0\,1$

$+$

$\boxed{1}\,0\,0\,1\,0\,0\,0$

Neglected

* 1's complement → Carry is added in ans.
  2's Complement → Carry is discard in ans.

Case 3:- Greater No. is -ve :-

$$(-A) + (+B) = -(A-B)$$

Q Add $(-23) + (+15)$

Sol ⇒ $+23 = 010111$

1^S comp $= 101000$

2^S comp $= 101001$

So, $-23 = 101001$

$+15 = 001111$

$$\begin{array}{r} 101001 \\ +\ \boxed{0}01111 \\ \hline \boxed{1}11000 \end{array} \quad = -8$$

To Verify: $11000 \leftarrow -8$

$00111 \leftarrow 1^S$ Comp

$+\quad\quad 1 \leftarrow 2^S$ Comp

$01000 \leftarrow 8$

Case IV :- Both No -ve

$$(-A) + (-B) = -(A+B).$$

Q $(-23) + (-15)$

Sol→ $+23 = 010111$

101000

$+ \quad 1$

$-23 = 101001$

$+15 = 01111$

10000

$+ \quad 1$

$-15 = 10001$

$-23 = 101001$

$-15 = \boxed{1}10001$

$\boxed{1}011010 = -38$

Neglect

To verify $011010 \leftarrow -38$

100101

1

$100110 \leftarrow 38$

Q Subtract (+14) & (+20).

Sol → +14 = 01110

+20 = 010100

1 0 1 0 1 1

\+ 1

−20 = 101100

0 0 1 1 1 0

\+ 1 0 1 1 0 0

1 1 1 0 1 0 → −6

To verify :-

1 1 0 1 0 → −6

0 0 1 0 1 → 1's Comp

\+ 1 → 2's Comp.

0 0 1 1 0 → +C

Multiplication

+ve No

↓

Unsigned
Signed positive

Q. Multiply 13 x 11

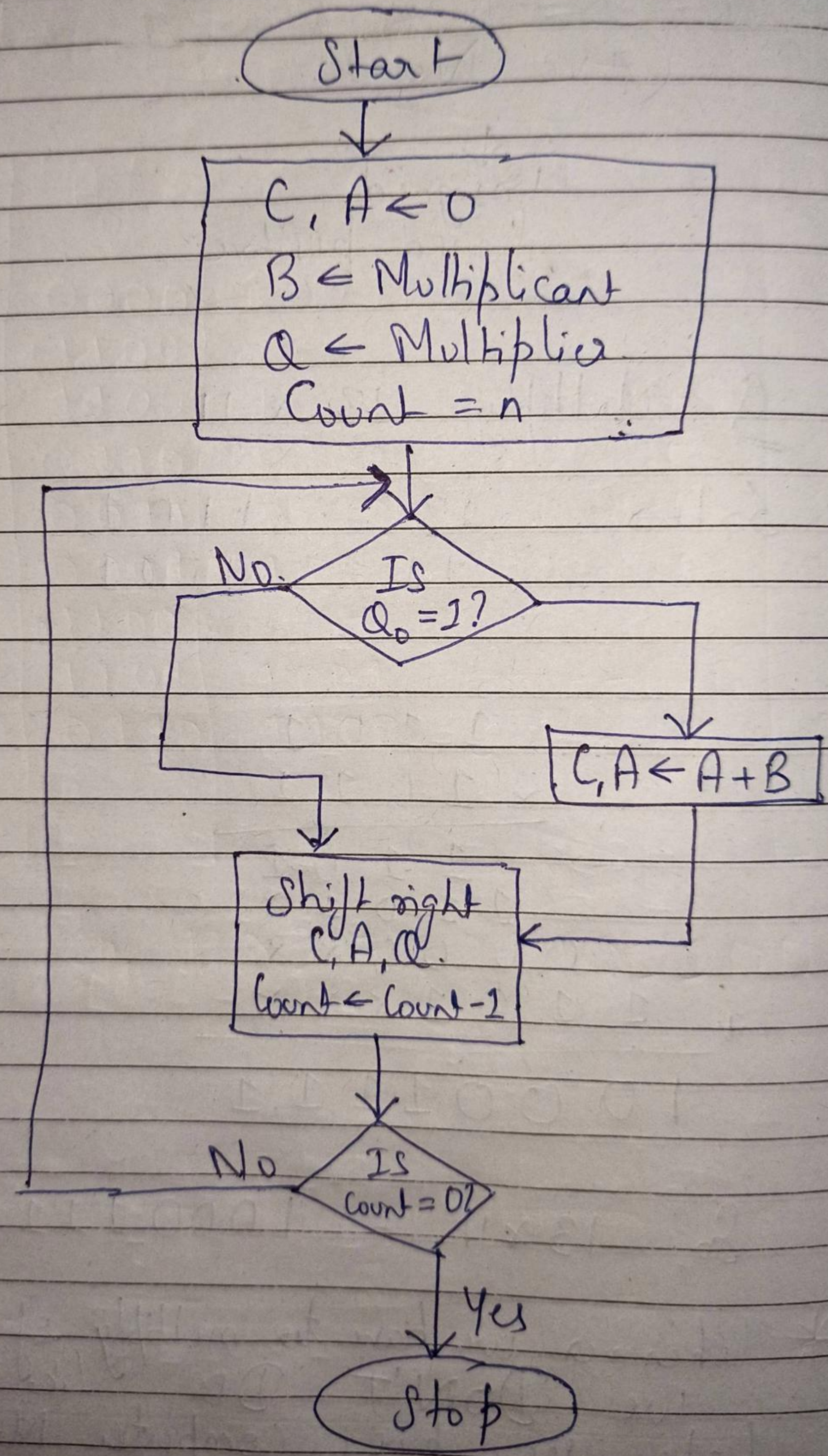
Sol → 13 = 1101
11 = 1011

```
    1 1 0 1
  x 1 0 1 1
  ---------
    1 1 0 1
  1 1 0 1 x
0 0 0 0 x x
1 1 0 1 x x x
---------------
1 0 0 0 1 1 1 1
```

So, 13 x 11 = 10001111

* Whenever we have to multiply of binary, we DON'T DO Like this.
Lets see how computer Multiply.

Flowchart of Multiplication



Q Multiply 13 x 11.

Sol → Multiplicand = 13 = B = 1101
Multiplier = 11 = Q = 1011

Carry	A	Q	operation	SC
0	0000	1011	Initial	4
	+1101			
0	1101	1011	Add A ← A+B.	
0	0110	1101	Shift right	3
	+1101			
1	0011	1101	Add A ← A+B.	
0	1001	1110	Shift right	2
	0100	1111	Shift right	1
	+1101			
1	0001	1111	Add A ← A+B.	
0	1000	1111	Shift right	0

So, $13 \times 11 = (10001111)_2$ Ans

Q Multiply 12 × 9.

Sol → 12 → Multiplicand = B = 1100
9 → Multiplier = Q = 1001

C	A	Q	Operation	SC.
0	0000	1001	Initial	4
	+1100			
0	1100	1001	Add A ← A+B	
0	0110	0100	Shift right	3
0	0011	0010	Shift right	2
	0001	1001	Shift right	1
	+1100			
0	1101	1001	Add A ← A+B	
0	0110	1100	Shift right	0

So, $12 \times 9 = (01101100)_2$ Ans

Q Multiply 14 × 11.

Sol → 14 → Multiplicant = B = 1110
11 → Multiplier = Q = 1011

C	A	Q	operation	SC
0	0000	1011	Initial	4
	1110			
0	1110	1011	Add A ← A + B.	
	0111	0101	Shift right	3
	1110			
1	0101	0101	Add A ← A + B	
0	1010	1010	Shift right	2
	0101	0101	Shift right	1
	1110			
1	0011	0101	Add A ← A + B	
	1001	1010	Shift right	0

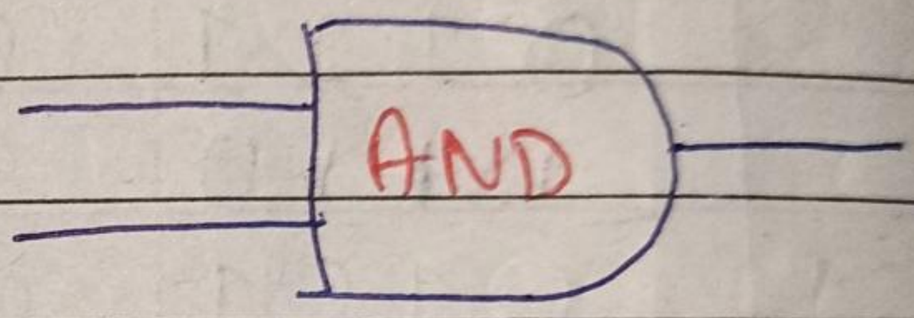
So, $14 \times 11 = (1001\ 1010)_2$ Ans

Adder

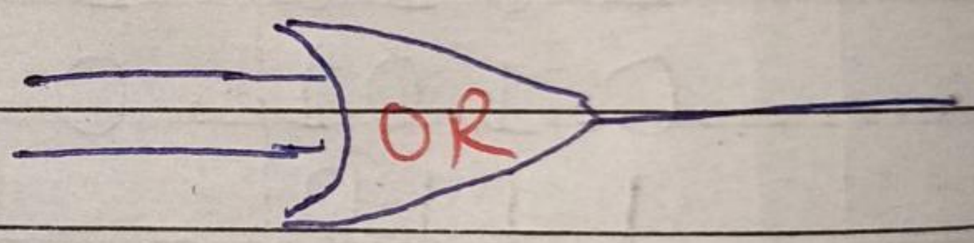
Half Adder Full adder.

Some basic logic Gate :-

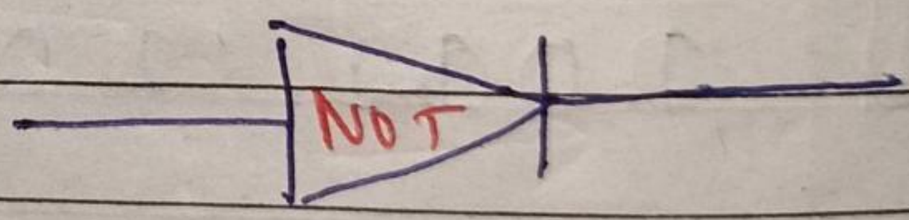
① AND :-



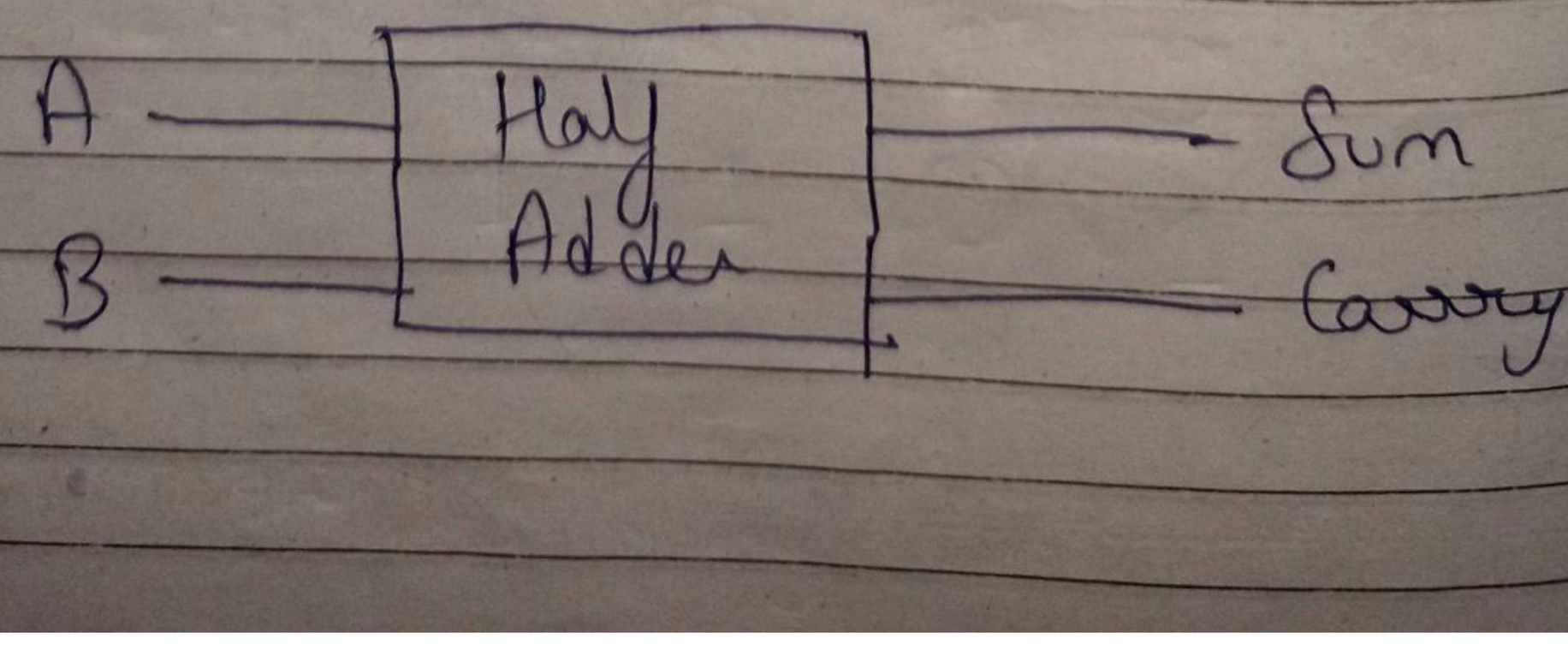
② OR :-



③ NOT :-



Half Adder :-



Truth Table for XOR

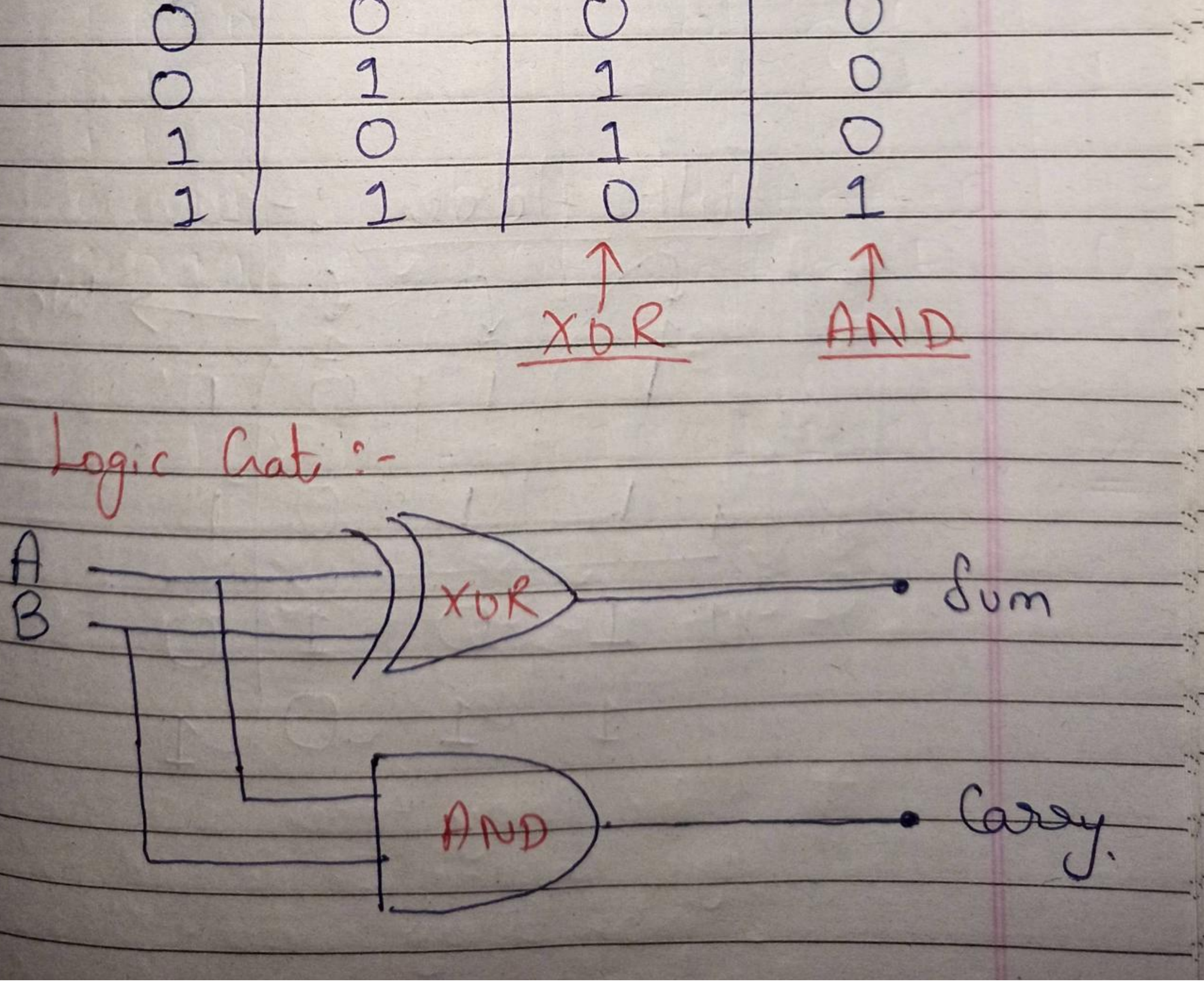
A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Truth Table for Half Adder :-

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

↑ XOR ↑ AND

Logic Gate :-



A_3 A_2 A_1 | A_0 → A

+ B_3 B_2 B_1 | B_0 → B

S_3 S_2 S_1 | S_0

C_3 C_2 C_1 | C_0

(*) Only Difference between Half Adder & Full adder is that in Half Adder we DO NOT TRANSFER CARRY.

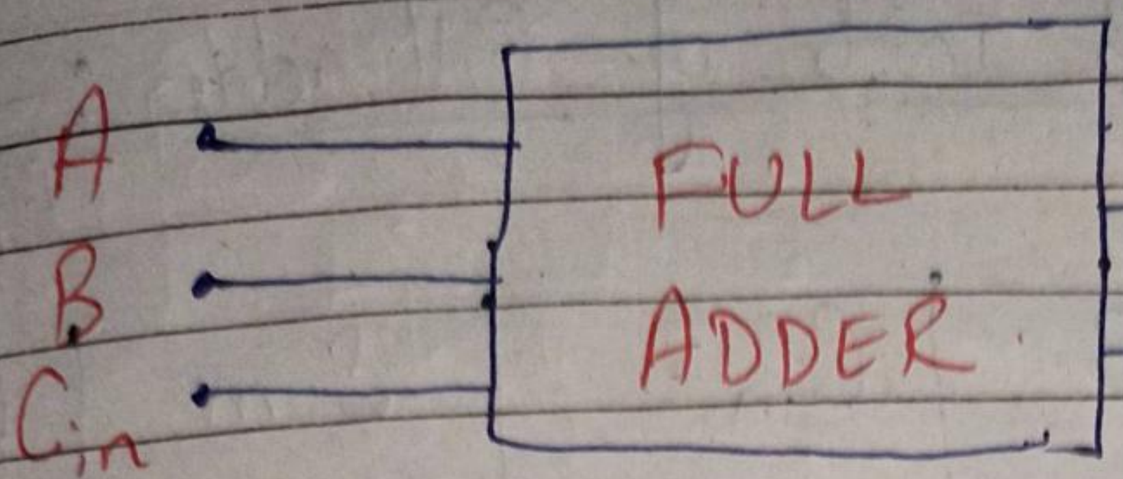
Full Adder :-

1 1 1 | 1 → We transfer Carry.

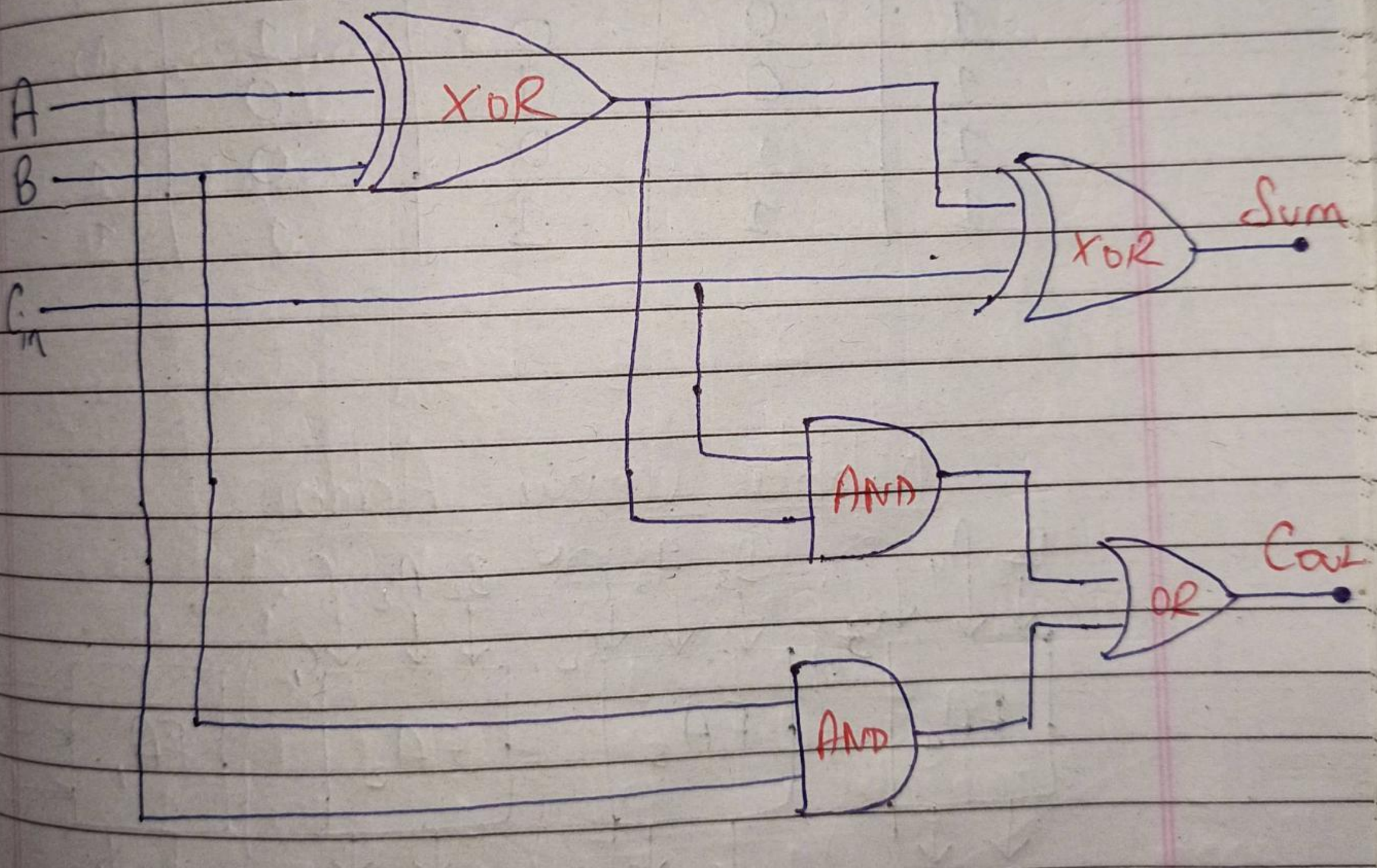
\+ 1 1 0 | 1

1 1 0 | 1 | 0

1 1 0 | 1



Logic Diagram :-



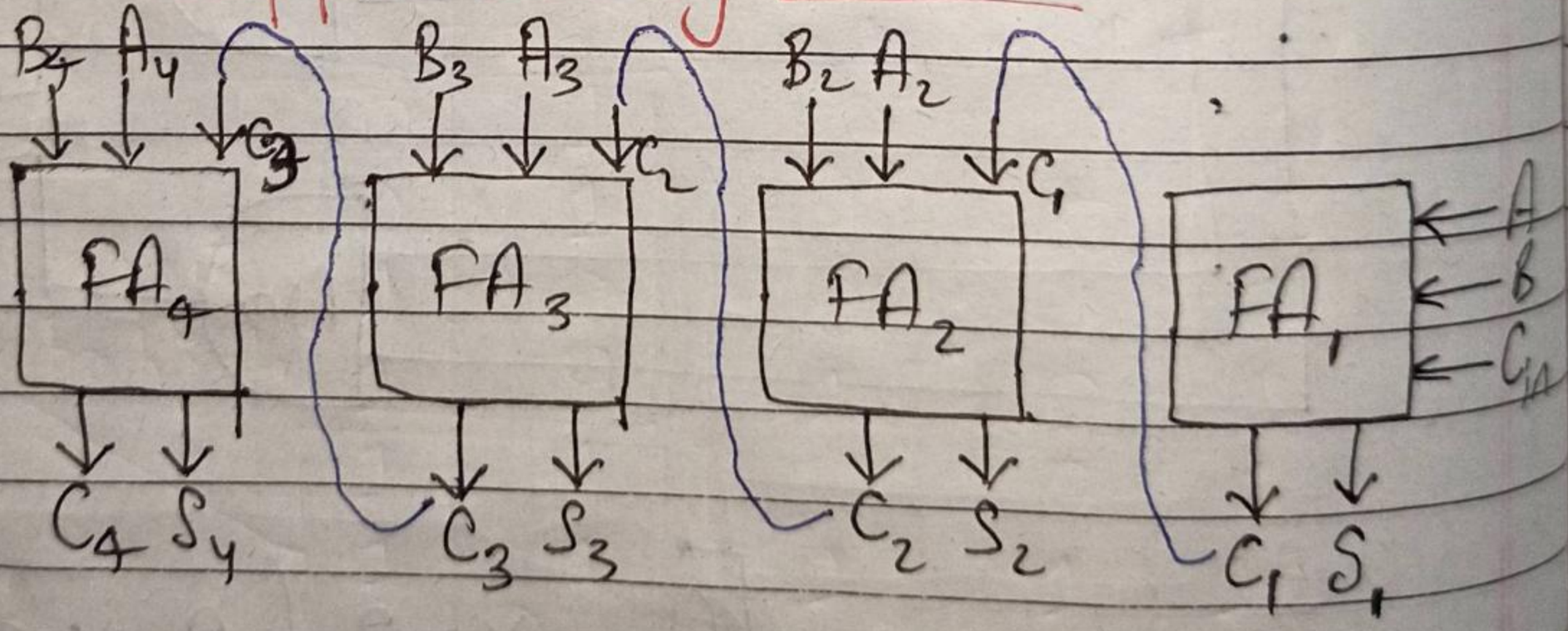
$$Sum \rightarrow [C_{in} \text{ XOR } (A \text{ XOR } B)]$$

$$Carry \rightarrow A \cdot B + C_{in} \cdot (A \text{ XOR } B)$$

Truth Table of Full adder :-

A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Ripple Carry Adder



* Ripple Carry Adder is a combinational logic circuit. It is used to add 2 n-bit binary number. It require n full adder in its circuit for adding

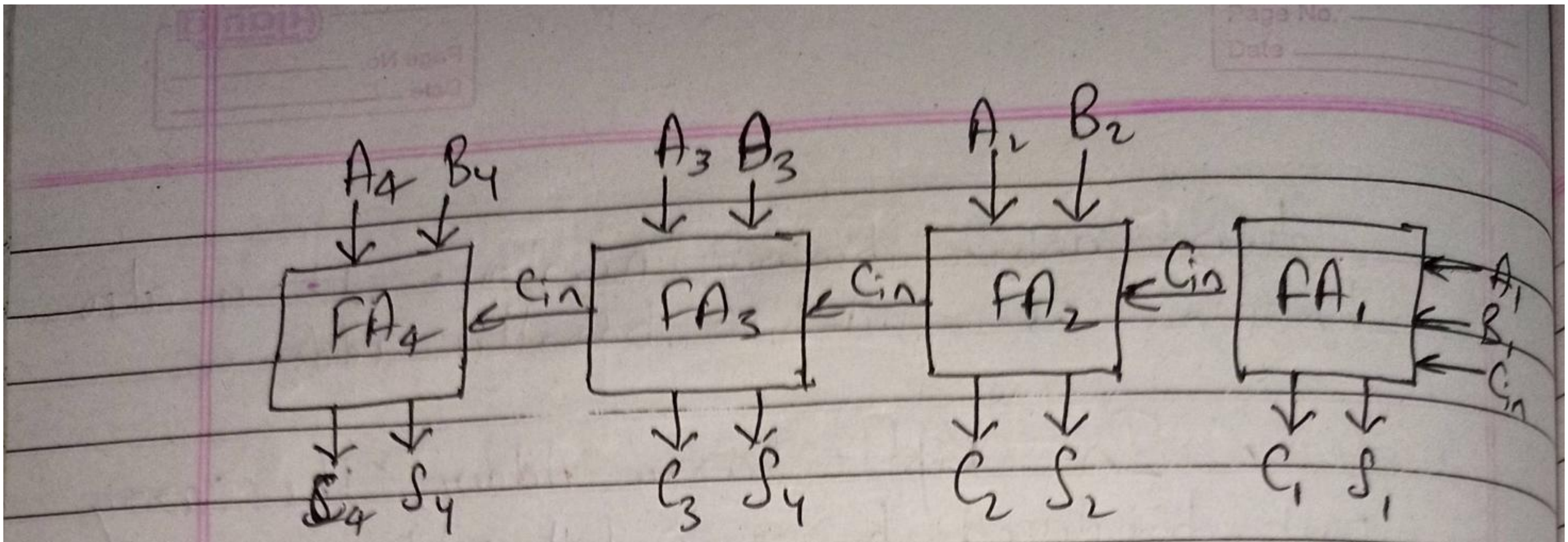
two n-bit binary number. It is also known as n-bit parallel adder.

In Ripple Carry Adder, The carry out produced by each full adder serves as carry-in for its adjacent most significant full adder. Each carry bit ripples or waves into next stage. That's why it is called Ripple Carry Adder.

* Ripple Carry Adder does not allow to use all the full adders simultaneously. Each full adder has to necessarily wait until the carry bit becomes available from its adjacent full adder. This increases propagation time. Due to this reason ripple carry adder becomes extremly slow. This is considered as biggest disadvantage of using ripple carry adder.

Carry look Ahead Adder :-

→ Used to save time as its propagation speed is more than that of Ripple Carry Adder. It generates Carry-in of each full adder simultaneously without causing any delay.



$$C_{out} = A \cdot B + (A \oplus B) \cdot C_{in}$$

$A \cdot B \rightarrow$ Carry Generator

$(A + B) \cdot C_{in} \rightarrow$ Carry propagator

A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Carry propagator

Carry Generator

$$C = G_1 + P \cdot C_{in}.$$

$$C_0 = G_{10} + P_0 \cdot C_{in}.$$

$$C_1 = G_{11} + P_1 \cdot C_0$$

$$= G_{11} + P_1 (G_{10} + P_0 \cdot C_{in}).$$

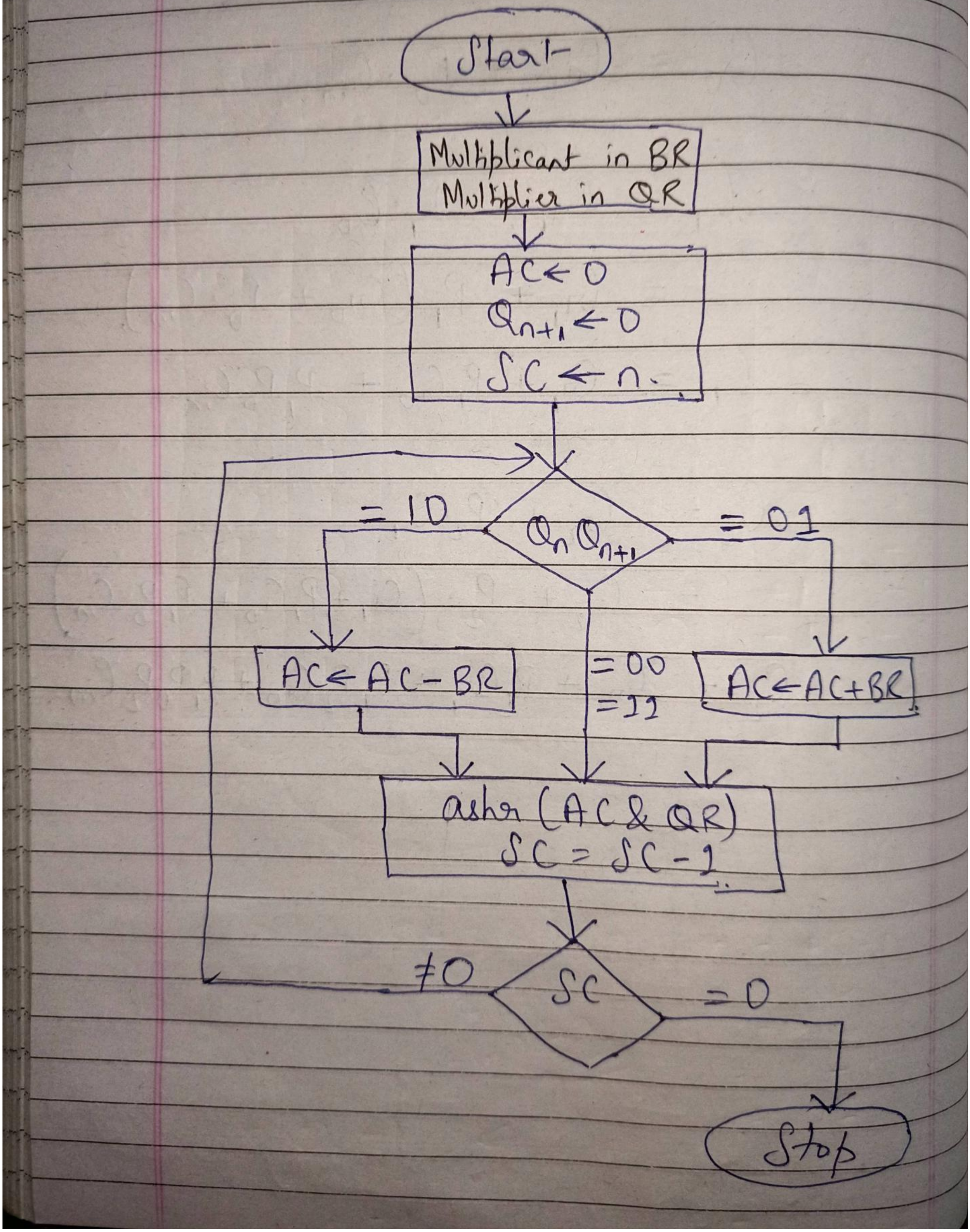
$$C_1 = G_{11} + P_1 G_0 + P_1 P_0 C_{in}$$

$$C_2 = G_{12} + P_2 \cdot C_1$$

$$= G_{12} + P_2 \cdot (G_1 + P_1 G_0 + P_1 P_0 C_{in})$$

$$C_2 = G_{12} + P_2 G_1 + P_1 P_2 G_0 + P_1 P_0 P_2 C_{in}.$$

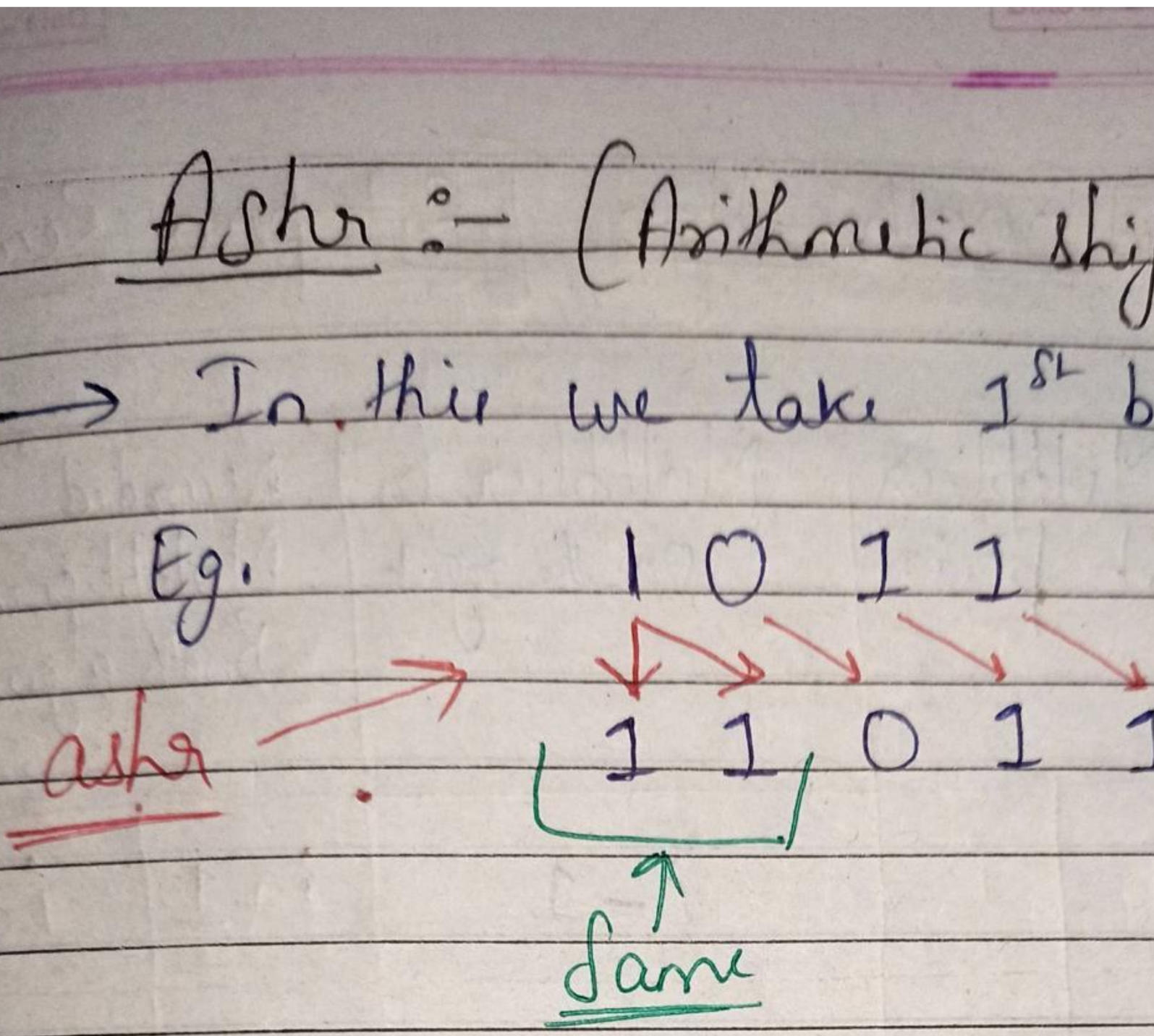
Booth Algorithm :-



Ashr :- (Arithmetic shift right)

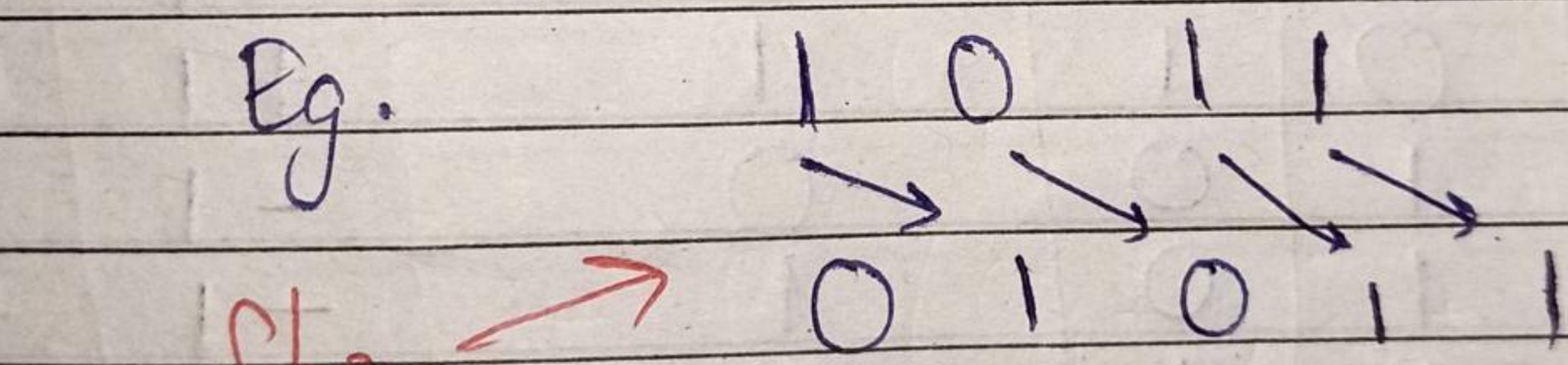
→ In this we take 1st bit as same

Eg. 1 0 1 1



In shift right we were not doing it.

Eg. 1 0 1 1



(*) Booth Algorithm is used for signed no. (+ve, -ve)

Q. Using Booth Algo
(a) Perform Multiplication of (-13) and (+6).
(b) Perform Multiplication of (+17) and (-12)
(c) Perform Multiplication of (-9) and (-13)

Ans → (a) Multiplicand = (-13) = 10011 → BR
Multiplier = (+6) = 00110 → QR.

operation	AC	QR	Q_{n+1}	SC
Initial	00000	00110	0	5
ashr	00000	00011	0	4
AC←AC-BR	+01101			
	01101	00011	0	
ashr	00110	10001	1	3
ashr	00011	01000	1	2
AC←AC+BR	+10011			
	10110	01000	1	
ashr	11011	00100	0	1
ashr	11101	10010	0	0

So, $(-13) \times (+6) = (11101110010)_2$ Ans

b) Multiplicant → (+17) = 010001 → BR
Multiplier → (-12) = 110100 → QR.
-BR → 101111

operation	AC	QR	Q_{n+1}	SC
Initial	000000	110100	0	6
ashr	000000	011010	0	5
ashr	000000	001101	0	4
AC←AC-BR	+101111			
	101111	001101		
ashr	110111	100110	1	3
AC←AC+BR	+010001			
	1001000	100110		
ashr	000100	010011	0	2
AC←AC-BR	+101111			
	110011	010011		
ashr	111001	101001	1	1
ashr	111100	110100	1	0

So, $(+17)\times(-12) = (111100110100)_2$

Ans

(c) Multiplicant = $-9 \rightarrow 10111 \rightarrow BR$

Multiplier = $-13 \rightarrow 10011 \rightarrow QR$

operation	AC	QR	Q_{n+1}	SC
Initial	00000	10011	0	5
AC ← AC-BR	+01001			
	01001	10011	0	
ashr	00100	11001	1	4
ashr	00010	01100	1	3
AC ← AC+BR	+10111			
	11001	01100	1	
ashr	11100	10110	0	2
ashr	11110	01011	0	1
AC ← AC-BR	+01001			
	10011	01011	0	
ashr	00011	10101	1	0

So, $(-9) \times (-13) = (000111010101)_2$ Ans

Modified Booth Algo [Bit pair Recording]

Multiplier bit pair		Multiplier b.4 on the right		Recorded Multiplier in Booth algo		Bit pair recorded multiplier bit at position i
i+1	i	i-1		i+1	i	
0	0	0		0	0	0
0	0	1		0	+1	+1
0	1	0		+1	-1	+1
0	1	1		+1	0	+2
1	0	0		-1	0	-2
1	0	1		-1	+1	-1
1	1	0		0	-1	-1
1	1	1		0	0	0

From i+1, i

Like :- +1, -1

$(+1)2^1 + (-1)2^0$

$= 2 - 1$

$= +1$

also, +1, 0

$1 \times 2^1 + 0 \times 2^0 = +2$

-1, 0

$= (-1)2^1 + 0$

$= -2$

0, -1

$= 0 \times 2^1 + (-1) \times 2^0$

$= -1$

Q Multiply (+13) × (-6).

Sol → +13 = 01101 → Multiplicand
-6 = 11010 → Multiplier.

Recorded Multiplier :-

1 1 1 0 1 0 [0] → append a zero at last

extra bit that we can get pair later on this

0 0 -1 1 -1 0

0 0 -1 1 -1 0

0 -1 -2

Extend 6-bit Now,

0 1 1 0 1
* 0 [x] -1 [x] -2

1 1 1 1 1 1 1 0 0 1 1 0

1 1 1 1 1 1 0 0 1 1 x x

+ 0 0 0 0 0 0 0 0 x x x x

1 1 1 1 1 1 0 1 1 0 0 1 0

Any

-2 = (-1)+(-1)

01101
-2

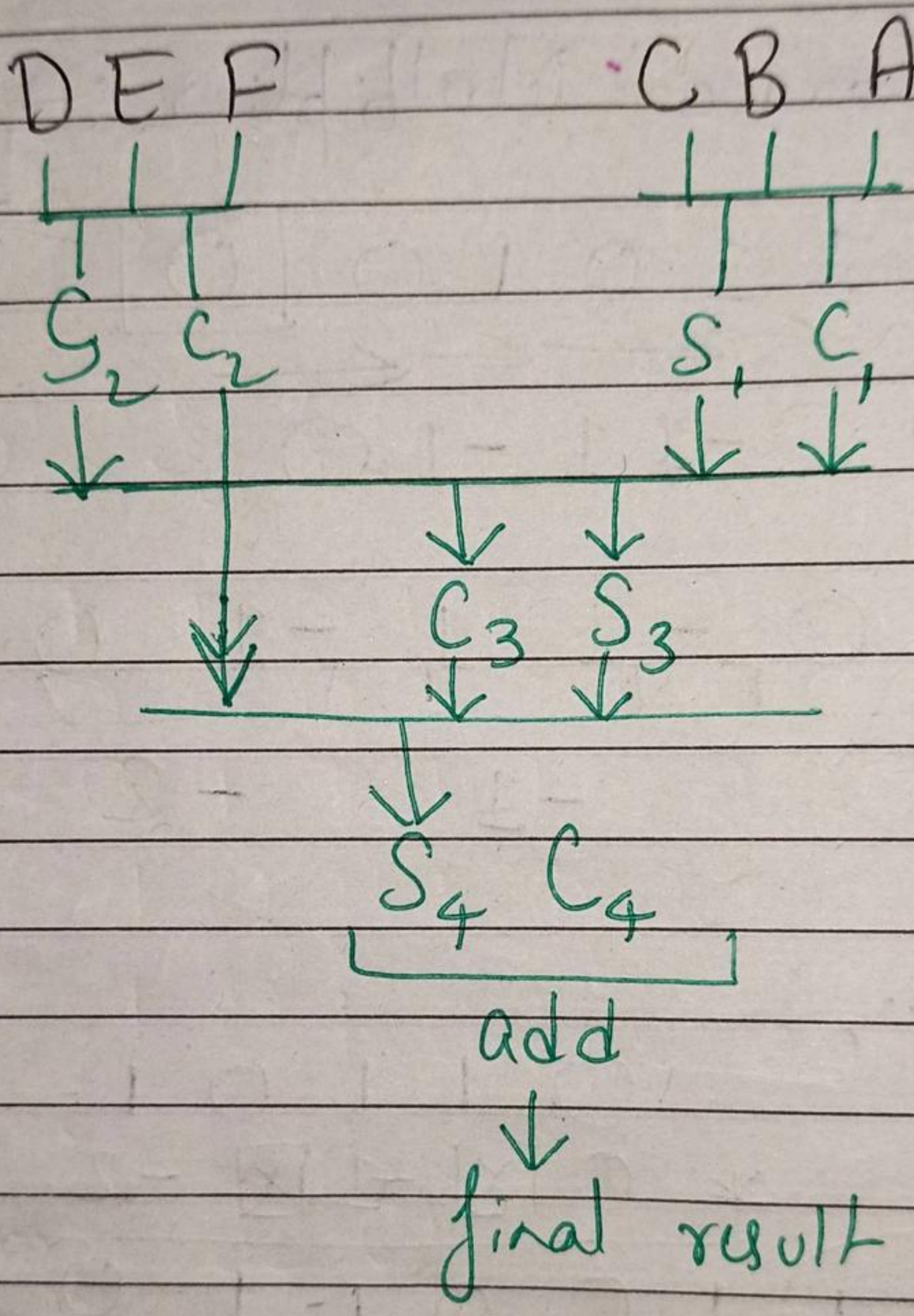
⇒ 01101
-1

= 10010
+ 1

10011 (-1)
+ 10011 (-1)
11 00110

Fast Multiplication.

Carry Save Addition Of Summands.



Q. Multiply 45 x 63.

Sol → 45 = 101101
63 = 111111

```
  1 0 1 1 0 1
x 1 1 1 1 1 1
-------------
  1 0 1 1 0 1      → A
  1 0 1 1 0 1 x    → B
  1 0 1 1 0 1 x x  → C
  1 0 1 1 0 1 x x x  → D
  1 0 1 1 0 1 x x x x  → E
  1 0 1 1 0 1 x x x x x  → F
-------------
```

```
    1 0 1 1 0 1      → A
    1 0 1 1 0 1 x    → B
+   1 0 1 1 0 1 x x  → C
-------------------
  1 1 0 0 0 0 1 1    → S1
  ↙ ↙ ↙ ↙ ↙ ↙ ↙
  0 1 1 1 1 0 0 x    → C1
```

Don't add Carry. Just put it here

```
  1 0 1 1 0 1 x x x  → D
  1 0 1 1 0 1 x x x x  → E
+ 1 0 1 1 0 1 x x x x x  → F
-------------------------
  1 1 0 0 0 0 1 1 0 0 0  → S2
  0 1 1 1 1 0 0 0 0 0 x  → C2
```

Now, $S_1 S_2 C_1$:-

$$
\begin{array}{l}
\quad\;\; 1\,1\,0\,0\,0\,0\,1\,1 \leftarrow S_1 \\
1\,1\,0\,0\,0\,0\,1\,1\,0\,0\,0 \leftarrow S_2 \\
+\quad\; 0\,1\,1\,1\,1\,0\,0\,x \leftarrow C_1 \\
\hline
1\,1\,0\,1\,0\,1\,0\,0\,0\,1\,1 \leftarrow S_3 \\
0\,0\,0\,1\,0\,1\,1\,0\,0\,0\,x \leftarrow C_3
\end{array}
$$

Now, $S_3 C_3 C_2$:-

$$
\begin{array}{l}
1\,1\,0\,1\,0\,1\,0\,0\,0\,1\,1 \leftarrow S_3 \\
0\,0\,0\,1\,0\,1\,1\,0\,0\,0\,x \leftarrow C_3 \\
0\,1\,1\,1\,1\,0\,0\,0\,0\,x \leftarrow C_2 \\
\hline
1\,0\,1\,1\,1\,0\,0\,1\,0\,0\,1\,1 \leftarrow S_4 \\
1\,0\,1\,0\,1\,0\,0\,0\,0\,0\,x \leftarrow C_4
\end{array}
$$

Now, $S_4 C_4$.

$$
\begin{array}{l}
1\,0\,1\,1\,1\,0\,1\,0\,0\,1\,1 \leftarrow S_4 \\
1\,0\,1\,0\,1\,0\,0\,0\,0\,0\,x \leftarrow C_4 \\
\hline
1\,0\,1\,1\,0\,0\,0\,1\,0\,0\,1\,1
\end{array}
$$

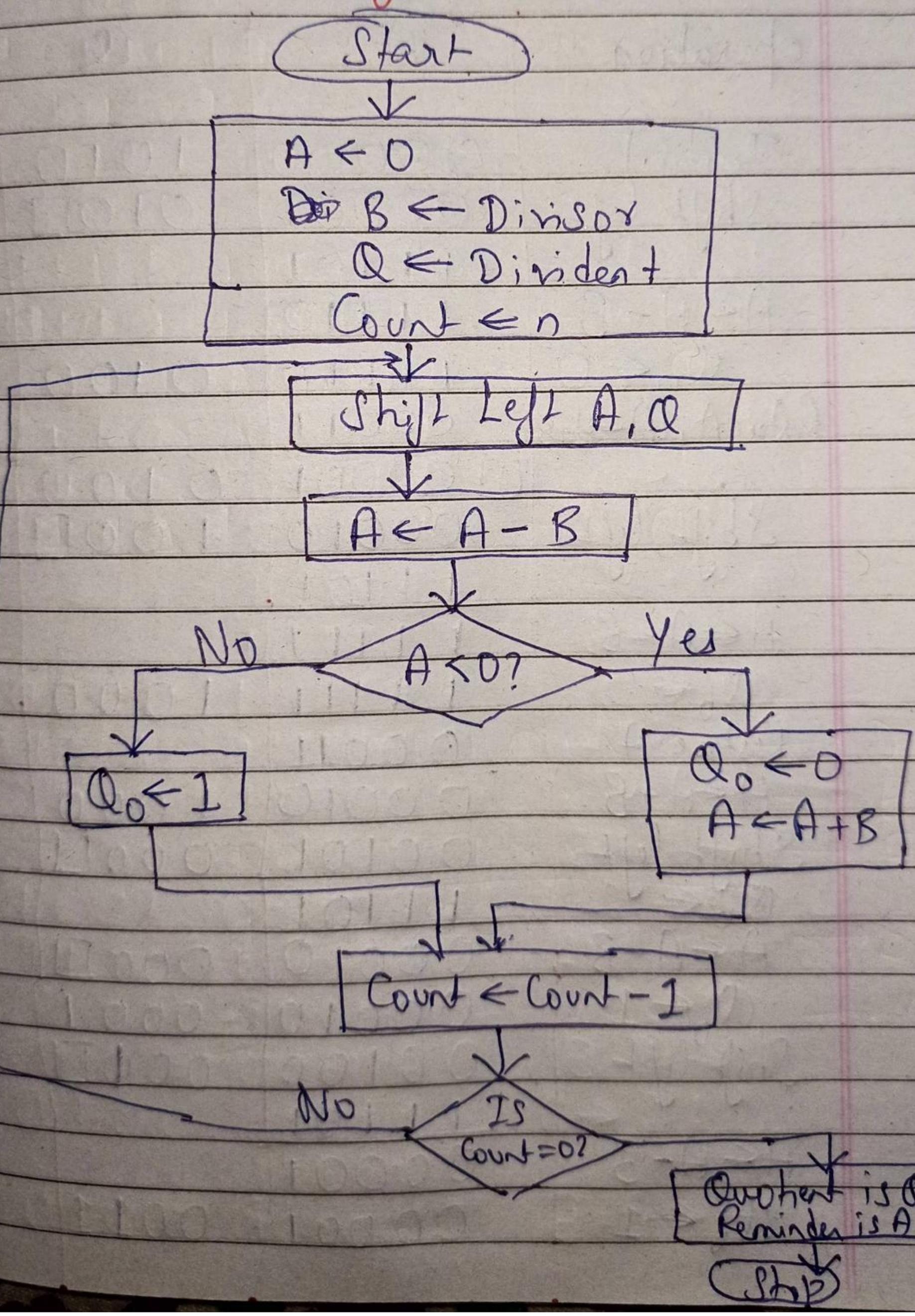
Normal add, in which carry will add

final result

Division

Restoring Division Non-Restoring Division

Restoring Division :-



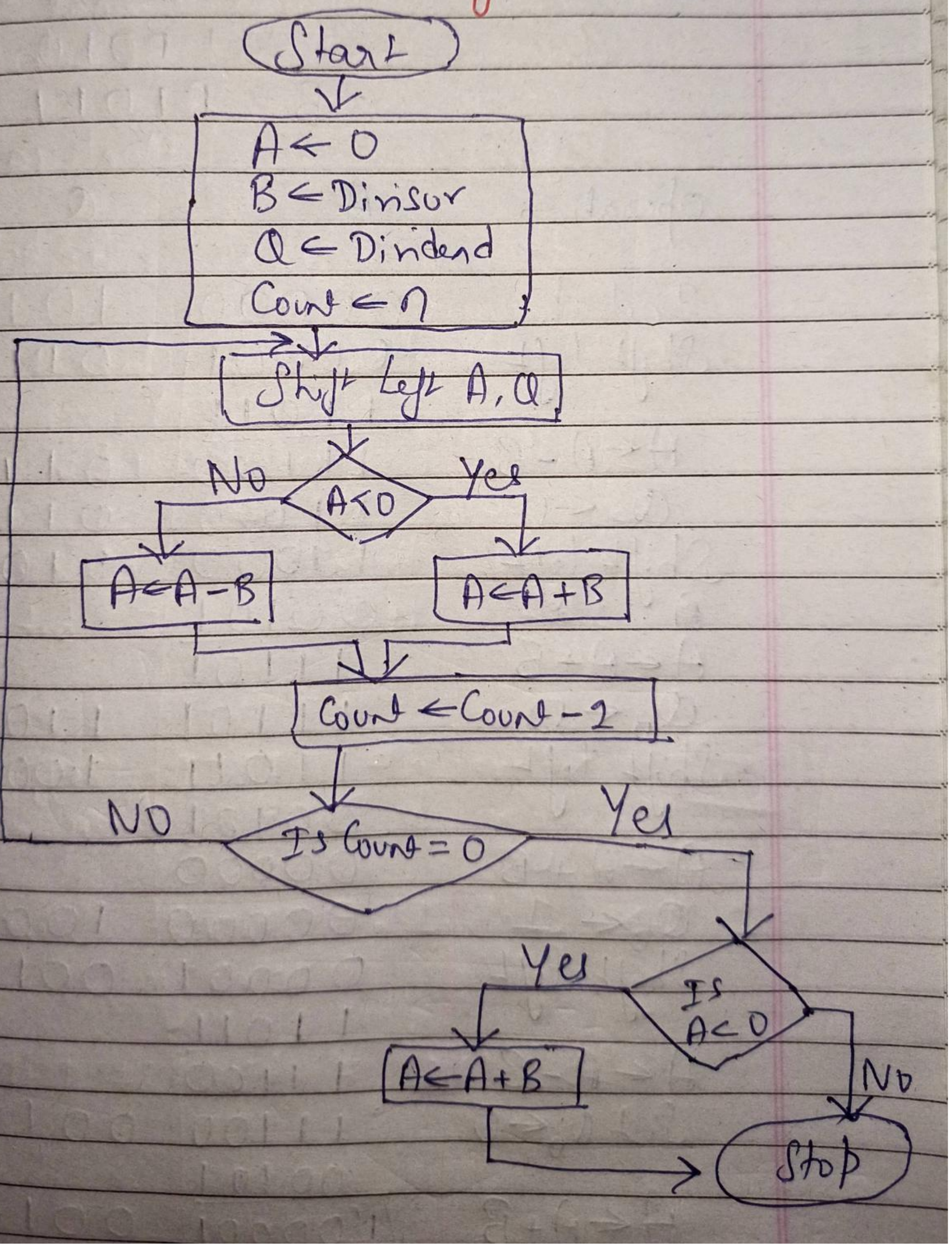
Q Divide when Dividend = 1010
Divisor = 00011

Sol → Q ← Dividend = 1010
B ← Divisor = 00011
−B = 11101

operation	A	Q	SC
Initial	00000	1010	4
Shift left	00001	010□	
	+ 11101		
A ← A − B	11110		
$Q_0 \leftarrow 0$	11110	0100	
Restore A, A ← A+B	00011		
	(1) 00001	0100	3
Shift left	00010	100□	
	11101		
A ← A − B	11111		
$Q_0 \leftarrow 0$	11111	1000	
Restore A	00011		
A ← A + B	00010	1000	2
Shift left	00101	000□	
	11101		
A ← A − B	00010	000□	
$Q_0 \leftarrow 1$	00010	0001	1
Shift left	00100	001□	
	11101		
A ← A − B	00001		
$Q_0 \leftarrow 1$	00001	0011	0

So, Reminder = 00001
Quotient = 0011 Ans

Non-Restoring Division



Q. Divide when Dividend = 1011
Divisor = 0101

Sol → Dividend → 1011 → Q.
Divisor → 0101 → B.

+B = 00101
11010
−B = 11011

Operation	A	Q	SC
Initial	00000	1011	4
Shift left	00001	011□	
	+11011		
A ← A − B	11100	011□	
Q_0 ← 0	11100	0110	3
Shift left	11000	110□	
	00101		
A ← A + B	11101		
Q_0 ← 0	11101	1100	2
Shift left	11011	100□	
	00101		
A ← A + B	00000		
Q_0 ← 1	00000	1001	1
Shift left	00001	001□	
	11011		
A ← A − B	11100		
Set Q_0 ← 0	11100	0010	0
	00101		
A ← A + B	(1)00001	0010	

So, Remiander = 00001

Quotient = 0010 Ans

Floating Point Representation :-

To represent floating point No. like

21.3

47.7845 :-

3 part of floating point representation :-

→ Mantissa

→ Radix / Base

→ Exponent

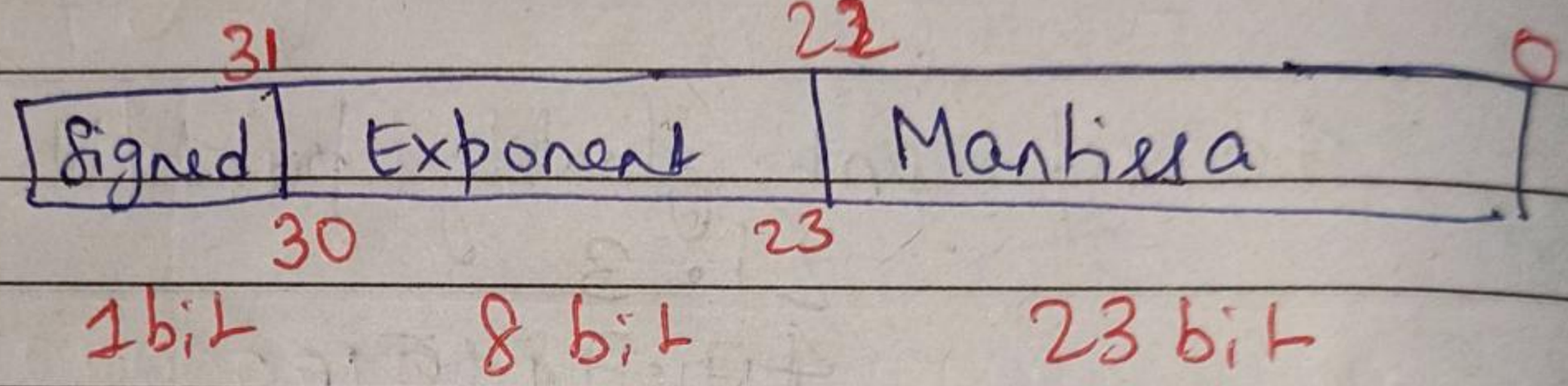
$\pm M \times B^{E}$ → Scientific Notation.

Number	Mantissa	Base	Exponent
9×10^{8}	9	10	8
127×10^{9}	127	10	9
284×10^{10}	284	10	10
123.782	123782	10	-3
1011×2^{3}	1011	2	3

Two format :-

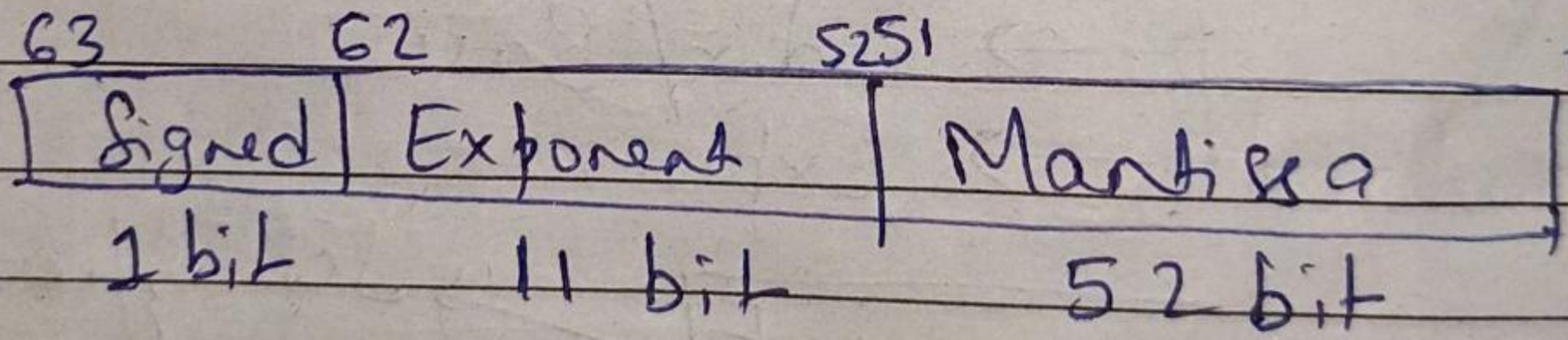
- Single precision format
- Double precision format

* Single Precision Format :-
In this There are 32 bit



* Double precision format :-

In this there are 64 bit



Q Represent $(1259.125)_{10}$ in Single & Double precision format.

Sol → Step 1 :- Convert Decimal to Binary.

$$(1259.125)_{10} = (10011101011.001)_2$$

Step 2:- Normalise the Number:-

Single P.F = $(1.N)\ 2^{E-127}$

Double P.F = $(1.N)\ 2^{E-1023}$.

N → Normalise Number.

$(1001110101 1.001)_2$

$= (1.0011101011001 \times 2^{10})$

↑ final Normalise No.

Now, for Single P.F :-

$(1.N)\ 2^{E-127}$ — (1)

$[(1.00111010 11001) \times 2^{10}]$ — (2)

On Comparing (1) & (2)

$E - 127 = 10$

$E = 137$.

Now, Binary of 137 = 10001001

So, Representation :-

0	10001001	00111010 11001000 - - -

For Double Precision.

$(1.N)\ 2^{E-1023}$ — — (1)

$(1.00111010111001)\ 2^{10}$ — — (2)

On Comparing (1) & (2).

$E = 1033$

Binary of 1033 = 10000001001

Representation :-

0	10000001001	0011101011100100 - -

Operation on floating point No.

Addition — Subtraction.

Q Add 0.5372400×10^{2}
0.1580000×10^{-1}

Sol → First we have to align the exponent

Let make 2nd No's exponent equal to 1st.

$\Rightarrow 0.1580000 \times 10^{-1} = 0.0001580 \times 10^{2}$

Now, on doing addition.

0.5372400×10^{2}
0.1580000×10^{-1}

$\Rightarrow$ 0.5372400×10^{2}
0.0001580×10^{2}
+
0.5373980×10^{2}

Q Subtract 0.56780×10^{5}
0.56430×10^{5}

Sol → 0.56780×10^{5}
$- 0.56430 \times 10^{5}$
0.00350×10^{5}

$\Rightarrow$ 0.350×10^{3}

↳ We did this because we always want integer after Point.