AKTU

B. Tech II-Year

Gateway Classes

GATEWAY CLASSES

Semester -III

CS IT & CS Allied Branches

BCS302- COMPUTER ORGANIZATION AND ARCHITECTURE

UNIT-2 Arithmetic and logic unit

Hand Written Notes

AKTU

B. Tech II-Year

Gateway Classes

BCS302- COMPUTER ORGANIZATION AND ARCHITECTURE

Hand Written Notes

Unit-2

Arithmetic and logic unit

Syallbus

Arithmetic and logic unit: Look ahead carries adders. Multiplication: Signed operand multiplication, Booths algorithm and array multiplier. Division and logic operations. Floating point arithmetic operation, Arithmetic & logic unit design. IEEE Standard for Floating Point Numbers

UNIT-2

Arithmetic and Logic Unit

Decimal to Binary conversion

128	64	32	16	8	4	2	1	
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
0	0	0	0	0	0	0	0	→ 0
0	0	0	0	0	0	0	1	→ 1
0	0	0	0	0	0	1	0	→ 2
0	0	0	0	0	0	1	1	→ 3
0	0	0	1	0	0	1	1	→ 19
0	0	0	1	0	1	0	0	→ 20

1 byte = 8 bits

Ques

	Q	R
2	7	1
2	3	1
	1	

$(111)_2$

Ques

2	16	0
2	8	0
2	4	0
2	2	0
	1	

$(10000)_2$

Ques

2	17	1
2	8	0
2	4	0
2	2	0
	1	

$(10001)_2$

How Negative Number represented in Binary

- Number
  - Unsigned
    - +ve
  - Signed
    - +ve, -ve
      - +ve → Binary
      - -ve
        - 1's complement
        - 2's complement

Most preferable method to represent negative number

One's And Two's complement

- One's complement

conversion :

1 convert → 0

0 convert → 1

$(1101)_2 \xrightarrow{\text{1's complement}} (0010)_2$ 2's complement

2's complement :-

Step-1 : Get the Binary Number of Particular decimal number

Step-2 :- Calculate One's complement

Step-3 : After calculating One's complement then add 1 in it

```
      59 → Decimal
 0111011 → Binary
 1000100 → 1's complement
      +1
 -------
 1000101 → 2's complement
```

Ques $(-7)_{10} \rightarrow (\ )_2$

convert into Binary

first ignore Negative sign and just convert decimal into binary.

Find out 2's complement

```
     8   4   2   1
    2^3 2^2 2^1 2^0   7 Decimal
     0   1   1   1      Binary
     1   0   0   0      1's complement
                +1      2's complement
    -------------
MSB ← 1   0   0   1 → LSB
```

$\boxed{-8+1 \Rightarrow -7}$

Notes:-

LSB least significant bit

MSB Most significant bit

Ques convert decimal into Binary

$(-8)_{10} \longrightarrow (\ )_2$

16	8	4	2	1	
2^4	2^3	2^2	2^1	2^0	8 → Decimal
0	1	0	0	0	→ Binary
1	0	1	1	1	→ 1's complement
				+1	2's complement
1	1	0	0	0	

$-16+8 = -8$

MSB (Most significant bit)

LSB (Least significant bit)

Binary Addition

Rules:-

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 (with 1 as carry to the next column)

1+1+1 = 1 (with 1 as carry to the next column)

Examples:-

```
   1010          1000
 + 1110        + 0111
 ------        ------
 11000          1111
 (carry)
```

```
  0101  → (5)      0110
  0101  → (5)    + 0111
  ----            ------
  1010  → (10)     1101
```

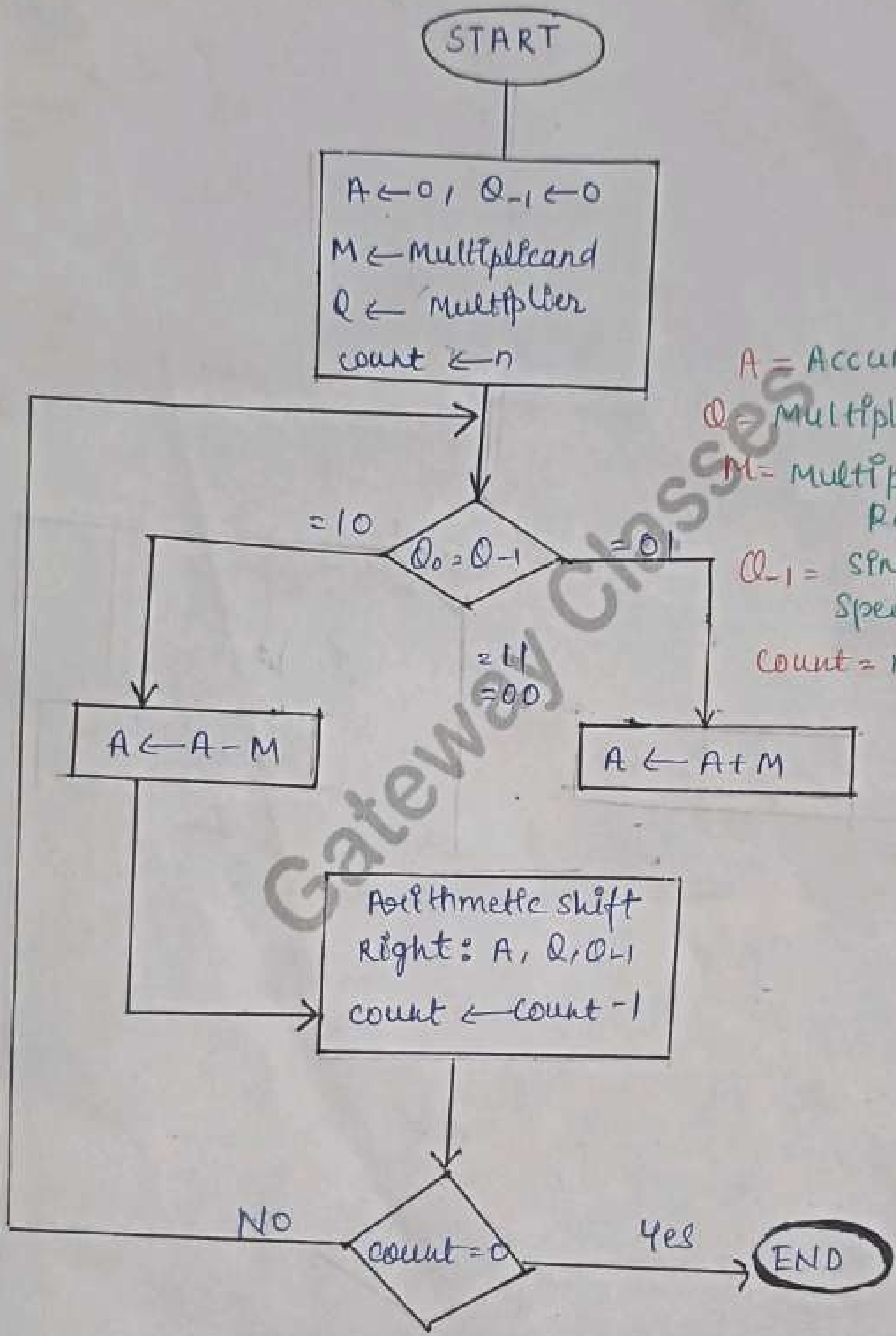
Booth's Algorithm

- It is an Algorithm used to Multiply two signed numbers.

Cases:-

* +ve * +ve
* −ve * −ve
* +ve * −ve
* −ve * +ve

BOOTH ALGORTHIM



A = Accumulator

Q = Multiple Register

M = Multiplicand Register

Q_{-1} = Single bit Special Register

Count = no of bit (n)

The steps in the booth's Algorithm are as follows:-

- Initialize A, Q_{-1} to 0 and count n.

- Based on the values of Q_0 and Q_{-1} do the following:-

(a) If $Q_0, Q_{-1} = 0,0$ then Right shift A, Q, Q_{-1} and finally decrement count by 1.

(b) If $Q_0, Q_{-1} = 0,1$ then Add A and M store in A, Right Shift A, Q, Q_{-1} and finally decrement count by 1.

(c) If $Q_0, Q_{-1} = 1,0$ then subtract A and M store in A, Right Shift A, Q, Q_{-1} and finally decrement count by 1.

(d) If $Q_0, Q_{-1} = 1,1$ the Right Shift A, Q, Q_{-1} and finally decrement count by 1.

- Repeat step 2 till count does not equal to 0.

Now, Question Practice based upon the above four cases:-

Q Multiply two Numbers (7*3) using Booth's Algorithm

Solution:-

A	Q	Q_{-1}	Operation	count
0000	0011	0	$Q_0 Q_{-1} = 10$ $A = A - M$ $A = A + (-M)$ $0000 + 1001$ $\begin{array}{r} 0000 \\ 1001 \\ \hline A = 1001 \\ \hline \end{array}$ ASR (A, Q, Q_{-1})	count = 4
1001	0011	0		count = 3
1100	1001	1	lost	
1100	1001	1	$Q_0 Q_{-1} = 11$	count = 3
1110	0100	1	ASR (A, Q, Q_{-1}) lost Automatic Shift Right	count = 2

A	Q	Q_{-1}	OPERATION	COUNT
1110	0100	1	$Q_0 Q_{-1} = 01$ $A = A + M$ $1110 + 0100$ $\begin{array}{r} 1110 \\ +0111 \\ \hline (1)0101 \end{array}$ Discard	count = 2
0101 0010	0100 1010	1 0	ASR(A, Q_0, Q_{-1}) lost	count = 1
0010 0001	1010 0101	0 0	$Q_0 Q_{-1} = 00$ ASR(A, Q_0, Q_{-1}) lost	count = 0

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	1	0	1	0	1

$\rightarrow 16 + 4 + 1 \Rightarrow 21$

Ques Multiply two numbers (-7 * 3) using Booth's Algorithm

solution

M = -7

+7 ⇒ 00111

1's ⇒ 11000

+1

11001

Check

$2^4\ 2^3\ 2^2\ 2^1\ 2^0$

11001

-16 + 8 + 1 = -7

M = -7 = 11001

Q = 3 = 00011

A = 00000

$Q_{-1} = 0$

-M = -(-7) = +7

-M = 00111

count = 5

Now, By Using the Booth's Algorithm

A	Q	Q_{-1}	OPERATION	COUNT
00000	00011	0	Q_0, Q_{-1} 1 0 $A = A - M$ $A = A + (-M)$ 00000 +00111 00111 ASR(A, Q, Q_{-1})	count=5
00111 00011	00011 10001	0 1	Lost	count=4
00011 00001	10001 11000	1 1	Q_0 Q_{-1} 1 1 ASR(A, Q, Q_{-1}) lost	count=4 count=3
00001	11000	1	Q_0 Q_{-1} 0 1 $A = A + M$ 00001 +11000 11001	

A	Q	Q_{-1}	OPERATION	COUNT
11010 11101	11000 01100	1 0	lost	COUNT = 2
11101 11110	01100 10110	0 0	$Q_0 Q_{-1} = 00$ ASR (A, Q, Q_{-1}) lost	count = 1
11110 11111	10110 01011	0 0	$Q_0 Q_{-1} = 00$ ASR (A, Q, Q_{-1}) lost	count = 1 count = 0

11111 01011 ⇒ −21

−(−21) = 21

CHECK ⇒

1111101011 → −21
0000010100 → 1's
+1 → 2's
0000010101 → +21

16 8 4 2 1
0000010101

$\boxed{16 + 4 + 1 \Rightarrow 21}$

Ques Multiply two numbers $(-7 * -3)$ using Booth's Algorithm.

Solution :- Case ⇒ 3 $-ve * -ve \Rightarrow +ve$

$M \Rightarrow -7$

$Q \Rightarrow -3$

$M \Rightarrow -7$

$+7 = 0111$

$1's \Rightarrow 1000$

$2's \Rightarrow +1$

1001

$M = -7 = 1001$

$-M = -(-7) \Rightarrow 7$

$\boxed{-M = 0111}$

$\boxed{A = 0000}$

$\boxed{Q_{-1} = 0}$

$Q \Rightarrow -3$

$+3 \Rightarrow 0011$

$1's \Rightarrow 1100$

$2's \Rightarrow +1$

1101

$\boxed{\text{count} = 4}$

$\boxed{Q = 1101}$

A	Q	Q_{-1}	OPERATION	
0000	1101	0	$Q_0 \; Q_{-1}$ 1 0 $A = A - M$ $A \Rightarrow A + (-M)$ 0000 +0011 0111 ASR(A, Q, Q_{-1})	count=4
0111 0011	1101 1110	0 1	Lost	count=3
0011	1110	1	$Q_0 \; Q_{-1}$ 0 1 $A \Rightarrow A + M$ 0011 +1001 1100	Count=3
1100 1110	1110 0111	1 0	ASR(A, Q, Q_{-1}) Lost	count=2

A	Q	Q_{-1}	OPERATION	COUNT
1110	0111	0	Q_0 Q_{-1} 1 0 A = A − M A = A + (−M) 1110 + 0111 (1)0101 Discard ASR (A, Q, Q_{-1})	count = 2
0101 0010	0111 1011	0 1	lost	count = 1
0010 0001	1011 0101	1 1	ASR (A, Q, Q_{-1}) lost	count = 1 Q_0 Q_{-1} 1 1 count = 0
	32 16 8 4 2 1 000101011 $\boxed{16 + 4 + 1 = 21}$			

Ques Show the Systematic process ((-15)*(-16)) using Booth's Algorithm

solution: M = -15

	32 16 8 4 2 1 / 2^5 2^4 2^3 2^2 2^1 2^0
+15	0 0 1 1 1 1
1's	1 1 0 0 0 0
2's	+ 1
	1 1 0 0 0 1

$\boxed{M = 110001}$ = -15

-M = -(-15)

-M = 001111 ⇒ +15

Q = -16 $Q_1 = 0$

	2^5 2^4 2^3 2^2 2^1 2^0
16	0 1 0 0 0 0
1's	0 0 1 1 1 1
	+ 1
2's	1 1 0 0 0 0

A = 000000

Count = 6

$\boxed{Q = 110000}$ ⇒ -16

A	Q	Q_{-1}	OPERATION	COUNT
000000 000000	110000 011000	0 0	$Q_0 Q_{-1} = 00$ ASR(A, Q, Q_{-1}) Lost	6 count = 5
000000 000000	011000 001100	0 0	$Q Q_{-1} = 00$ ASR (A, Q, Q_{-1}) lost	5 count = 4
000000 000000	001100 000110	0 0	$Q Q_{-1} = 00$ ASR (A, Q, Q_{-1}) Lost	4 count = 3
000000 000000	000110 000011	0 0	$Q Q_{-1} = 00$ ASR (A, Q, Q_{-1}) Lost	3 count = 2

A	Q	Q_{-1}	OPERATION	COUNT
000000	000011	0	$Q_0 Q_{-1} = 10$ $A = A - M \rightarrow$ $A = A + (-M)$ $\begin{array}{r} 000000 \\ +\ 001111 \\ \hline 001111 \end{array}$ ASR(A, Q, Q_{-1})	2
001111	000011	0		
000111	100001	1	Lost	count=1
000111	100001	1	$Q_0 Q_{-1} = 11$ ASR(A, Q, Q_{-1})	1
000011	110000	1	Lost	count=0

000011110000

$\Rightarrow$ 240

Ans

$128 - 2^7$
$64 - 2^6$
$32 - 2^5$
$16 - 2^4$

240

Ques: Multiply two numbers (13×-15) using the Booth's Algorithm.

$13 \times (-15) = -195$

Solution :-

$M = +13$

$-M = -(13) = -13$

$13 = 01101$

$1's = 10010$

$2's \quad +1$

10011

$\boxed{-M = 10011}$

$Q = -15$

$+15 = 01111$

$1's = 10000$

$2's = \quad +1$

10001

$\boxed{Q = 10001}$ -15

$\boxed{A = 00000}$

$\boxed{Q_1 = 0}$

$\boxed{\text{count} = 5}$

A	Q	Q-1	OPERATION	COUNT
00000	$Q_4 Q_3 Q_2 Q_1 Q_0$ 10001	0	$Q_0\ Q_{-1} \Rightarrow 10$ $A = A - M \rightarrow$ $A = A + (-M)$ $\begin{array}{r} 00000 \\ +\ 10011 \\ \hline 10011 \end{array}$ ASR (A, Q, Q_{-1})	5
10011	10001	0		
11001	11000	1	lost	count=4
11001	11000	1	$Q_0\ Q_{-1} = 01$ $A = A + M$ $\begin{array}{r} 11001 \\ +\ 01101 \\ \hline 1\,00110 \end{array}$ ASR (A, Q, Q_{-1})	4
00110	11000	1		
00011	01100	0	lost	count=3

A	Q	Q_{-1}	OPERATION	COUNT
00011 00001	01100 10110	0 0	$Q_0 Q_{-1} \Rightarrow 00$ ASR (A, Q, Q_{-1}) lost	3 count=2
00001 00000	10110 11011	0 0	$Q_0 Q_{-1} = 00$ ASR (A, Q, Q_{-1}) lost	2 count=1
00000 10011 11001	11011 11011 11101	0 0 1	$Q_0 Q_{-1} \Rightarrow 10$ $A = A - M$ 00000 + 10011 10011 ASR (A, Q, Q_{-1}) lost	1 count=0

1100111101 $\Rightarrow$ -195

or

Ans

1's $\rightarrow$ 0011000010

2's $\rightarrow$ +1

0011000011

$\hookrightarrow$ +195

Ques:- Multiply two numbers (-13 * +8) by using Booth's Algorithm.

Solution $M = -13$ $-13 * 8 \Rightarrow -104$

+13 $\Rightarrow$ 01101

1's $\Rightarrow$ 10010

2's $\Rightarrow$ +1

10011

M = 10011 $-M = -(-13) \Rightarrow 13$

$\boxed{-M = 01101}$

Q = 8

+8 $\Rightarrow$ 01000

$\boxed{A = 00000}$ $\boxed{Q_{-1} = 0}$

$\boxed{\text{count} = 5}$

A	Q	Q_{-1}	OPERATION	COUNT
00000 00000	01000 00100	0 0	$Q_0\ Q_{-1} \Rightarrow 00$ Arithmetic Shift Right (A, Q, Q_{-1}) lost	count ⇒ 5 count ⇒ 4
00000 00000	00100 00010	0 0	$Q_0\ Q_{-1} = 00$ ASR (A, Q, Q_{-1}) lost	4 count ⇒ 3
00000 00000	00010 00001	0 0	$Q_0\ Q_{-1} \Rightarrow 00$ ASR (A, Q, Q_{-1}) lost	count = 3 count ⇒ 2

A	Q	Q_{-1}	OPERATION	Count
00000	00001	0	$Q_0 Q_{-1} \Rightarrow 10$ $A = A - M$ $A \Rightarrow A + (-M)$ 00000 +01101 01101 ASR (A, Q, Q_{-1})	count ⇒ 2
01101 → 00110	00001 → 10000	0 → 1	lost	count ⇒ 1
00110	10000	1	$Q_0 Q_{-1} = 01$ $A = A + M$ 00110 + 10011 11001	count = 1
11001 → 11100	10000 → 11000	1 → 0	ASR (A, Q, Q_{-1}) lost	count = 0

Rechecking:- $-13 * 8 = -104$

1110011000 → (−104)

1110011000
0001100111 → 1's
+1 → 2's

$\underline{0001101000}$ ⇒ $\underline{+104}$

64+32+8 ⇒ 104

Ques:- Show the systematic process (+20)*(−19) using Booth's Algorithm.

Solution:- M ⇒ +20

M = 010100

−M = −(+20) = −20

+20 ⇒ 010100

1's ⇒ 101011

2's ⇒ +1

101100

$\boxed{-M = 101100}$ ⇒ −20

$\boxed{A = 000000}$

$\boxed{Q_{-1} = 0}$

$\boxed{\text{count} = 6}$

Q = −19

+19 ⇒ 010011

1's ⇒ 101100

2's ⇒ +1

Q ⇒ 101101 ⇒ −19

Gateway Classes

A	Q	Q_{-1}	OPERATION	COUNT
000000	101101	0	$Q_0 Q_{-1} \Rightarrow 10$ $A \Rightarrow A - M \rightarrow$ $A = A + (-M)$ 000000 + 101101 101100 ASR (A, Q, Q_{-1})	count = 6
101100 110110	101101 010110	0 1	Lost	Count = 5
110110	010110	1	$Q_0 Q_{-1} \Rightarrow 01$ $A = A + M$ 110110 + 010110 1 001010 Discard ASR (A, Q, Q_{-1})	count = 5
001010 000101	010110 001011	1 0	Lost	count = 4

A	Q	Q_{-1}	OPERATION	COUNT
000101	001011	0	$Q_0 Q_{-1} = 10$ $A = A - M \rightarrow$ $A = A + (-M)$ $\begin{array}{r} 000101 \\ +101100 \\ \hline 110001 \end{array}$	count=4
110001	001011	0	ASR (A, Q, Q_{-1})	
111000	100101	1	lost	count=3
111000	100101	1	$Q_0 Q_{-1} = 11$ ASR (A, Q, Q_{-1})	count=3
111100	010010	1	lost	count=2
111100	010010	1	$Q_0 Q_{-1} \Rightarrow 01$ $A = A + M \rightarrow$ $A = A + (-M)$	count=2

A	Q	Q-1	OPERATION	COUNT
010000 001000	010010 001001	1 0	111100 +010100 (1)010000 ASR (A, Q, Q-1) lost	count=1
001000 110100 111010	001001 001001 000100	0 0 1	$Q_0\ Q_{-1} = 10$ $A = A-M \rightarrow$ $A = A+(-M)$ 001000 +101100 110100 ASR (A, Q, Q-1) lost	count=1 count=0

$(20)*(-19) \Rightarrow -380$

$111010000100 \Rightarrow -380$ Ans

Question:- Show the content of register E, A, Q, & SC during the process of Multiplication of two binary number 11111 (multiplicand) 10101 (multiplier) the sign are not included.

Solution:-

M = 011111

1's ⇒ 100000
2's ⇒ +1
100001

$\boxed{-M = 100001}$

$\boxed{Q = 010101}$

$\boxed{A = 000000}$

$\boxed{E = 0}$ → 1 bit Special Register

$\boxed{SC = 6}$

Booth's Algorithm's Representation

A	Q	E	OPERATION	SC
000000	010101	0	$QE = 10$ $A = A - M \rightarrow$ $A = A + (-M)$ $\begin{array}{r} 000000 \\ + \; 100001 \\ \hline 100001 \end{array}$	SC=6
100001 110000	010101 101010	0 1	ASR(A, Q, E) lost	SC=5
110000	101010	1	$Q_0 E = 01$ $A = A + M$ $\begin{array}{r} 110000 \\ + \; 011111 \\ \hline 1\,001111 \end{array}$ Discard	SC=5
001111 000111	101010 110101	1 0	ASR(A, Q, E) lost	SC=4

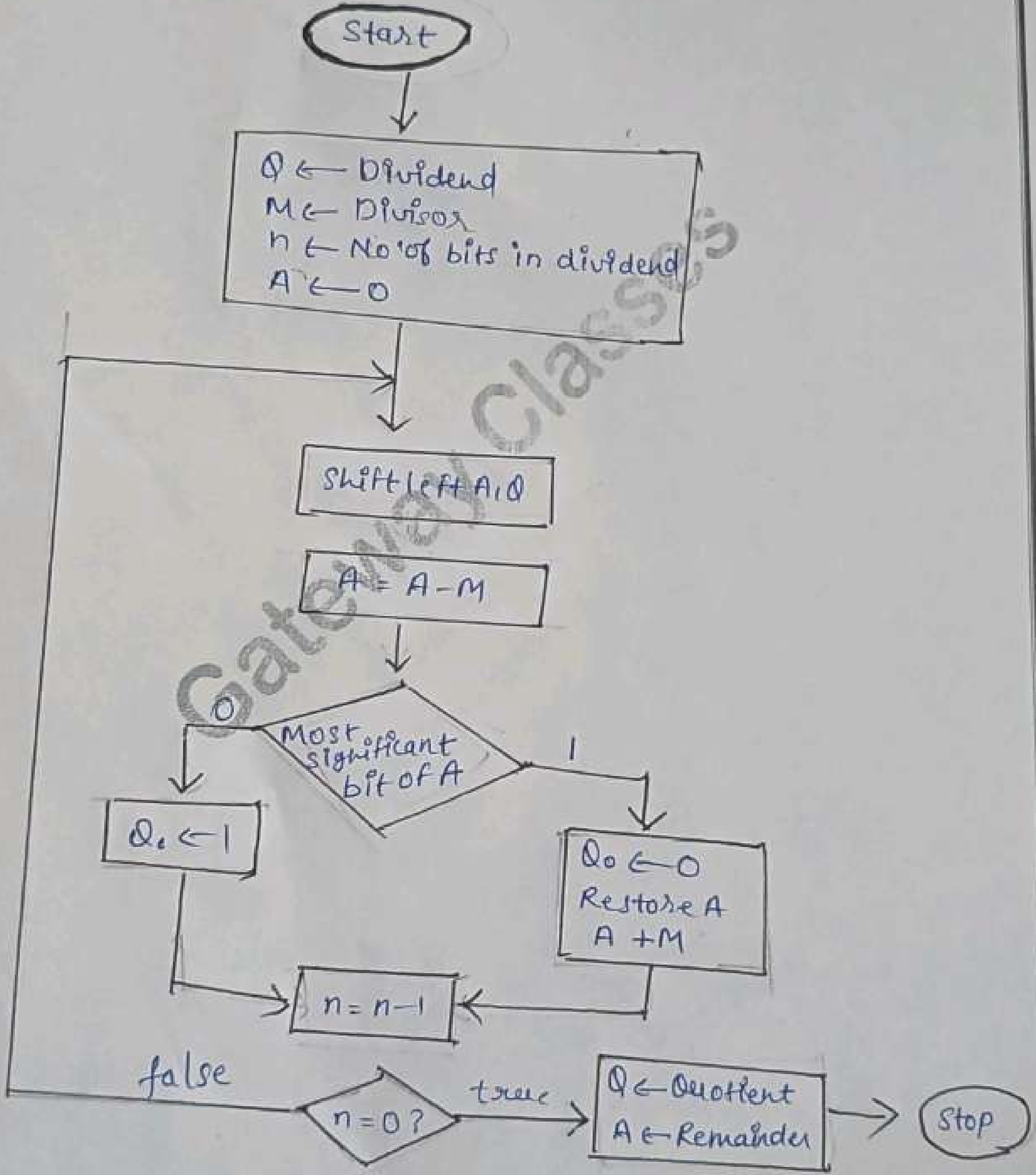
A	Q	E	OPERATION	Count
000111	110101	0	$Q_0E = 10$ $A = A+M$ 000111 + 100001 101000	SC = 4
101000	110101	0	ASR (A, Q, E)	
110100	011010	1	Lost	SC = 3
110100	011010	1	$Q_0E = 01$ $A = A+M$ 110100 + 011111 1 010011 Discard	
010011	011010	1	ASR (A, Q, E)	
001001	101101	0	Lost	SC = 2

A	Q	E	OPERATION	SC
001001	101101	0	$Q_0E = 10$ $A = A - M$ $A = A + (-M)$ 001001 + 100001 101010 ASR (A, Q, E)	SC = 2
101010 110101	101101 010110	0 1	lost	SC = 1
110101	010110	1	$Q_0E = 01$ $A = A + M$ 110101 + 011111 1 010100 Discard ASR (A, Q, E)	SC = 1
010100 001010	010110 001011	1 0	lost	SC = 0

001010001011 ⇒ 512 + 128 + 8 + 2 + 1 ⇒ 651

Ans

Restoring Division Algorithm:- Division for Unsigned Integer



Question:-

Restoring division Algorithm: for unsigned number

11 (DIVIDEND) / 3 (DIVISOR) = 3 (QUOTIENT) 2 (REMAINDER)

Solution:-

M = 3 (Divisor)

$\boxed{M \rightarrow 00011}$ → 3

+3 ⇒ 00011
1's ⇒ 11100
2's ⇒ +1
11101 → −M

$\boxed{-M = 11101}$

Q = 11 (Dividend)

11 ⇒ 1011

$\boxed{A \Rightarrow 00000}$

n = 4

$\boxed{\text{count} = 4}$

By Using Booth's Algorithm method:-

A	Q	OPERATION	COUNT

00000
00001
	1011
	011 $\boxed{?}$
		Shift left A, Q
		A = A-M → A = A+(-M)
		  00001
		+ 11101
		  11110
			4

11110	011 $\boxed{1}$	MSB [A] = 1 $Q_0 \leftarrow 0$	
00001	0110	Restore A	3

00001
00010
	0110
	110 $\boxed{?}$
		Shift left
		A = A - M
		  00010
		- 11101
		  11111
			3

11111	110 $\boxed{?}$	MSB [A] = 1 $Q_0 \leftarrow 0$	count = 2
00010	1100	Restore A	

A	Q	OPERATION	COUNT
00010	1100	Shift left A, Q	n=2
00101	100 $\boxed{?}$	$A = A - M \rightarrow$	
		$A = A + (-M)$	
		$\begin{array}{r} 00101 \\ +\ 11101 \\ \hline 1\ 00010 \end{array}$ Discard	
00010	100 $\boxed{?}$	Most significant bit of A	count=1
		$MSB[A] = 0$	
00010	1001	$Q_0 \leftarrow 1$	
00010	1001	Shift left A, Q	n=1
00101	001 $\boxed{?}$	$A = A - M \rightarrow A = A + (-M)$	
		$\begin{array}{r} 00101 \\ +\ 11101 \\ \hline 1\ 00010 \end{array}$ Discard	
00010	001 $\boxed{?}$	Most significant bit of A	Count=0
		$MSB[A] = 0$	
00010	0011	$Q_0 \leftarrow 1$	

A → 0010 → 2 → Remainder

Q → 0011 → 3 → Quotient

Ques:- Restoring Division Algorithm:- for unsigned number 00001111 by 0011.

Solution:-

Q = 00001111

n = 8
count = 8

A = 0000 0000 0

M = 00000 0011

$$
\begin{array}{lr}
M \Rightarrow & 00000\ 0011 \\
1'S \Rightarrow & 11111100 \\
2'S \Rightarrow & +1 \\
\hline
 & 11111101 \\
\end{array}
$$

−M = 11111101

Q = 15
M = 3

$$
\begin{array}{r}
5 \rightarrow \text{Quotient} \\
3\overline{)15} \\
15 \\
\hline
0 \rightarrow \text{Remainder}
\end{array}
$$

A	Q	OPERATION	COUNT
0000 00000 000000000	0000 1111 0001111$\boxed{\ }$	shift left A, Q	8
1111 11101	0001 111$\boxed{\ }$	$A = A - M \rightarrow$ $A = A + (-M)$ 0000 00000 + 1111 11101 1111 11101	
		Most significant Bit of A MSB[A] = 1	count = 7
0000 00000	0001 1110	$Q_0 \leftarrow 0$ Restore A	
0000 00000	0001 1110	shift left A, Q	7
		$A = A - M \rightarrow A = A + (-M)$ 0000 00000 1111 11101 1111 11101	
1111 11101	0011 110$\boxed{\ }$	MSB[A] = 1	
		$Q_0 \leftarrow 0$	count = 6
0000 00000	0011 1100	Restore A	

Gateway Classes

COA CS / IT / CS Allied / MCA

A	Q	OPERATION	count
000000000 000000000	0011 1100 0111100☐	Shift left A,Q	6
		A = A − M 000000000 11111101 11111101	
1111 1101	0111 100☐	MSB [A] = 1 $Q_0 \leftarrow 0$	
000000000	0111 1000	Restore A	count = 5
000000000 000000000	0111 1000 1111000☐	Shift left A,Q	5
		A = A − M A = A + (−M) 000000000 11111101 11111101	
1111 1101	1111000☐	MSB [A] = 1	
000000000	1110000	$Q_0 \leftarrow 0$ Restore A	count ⇒ 4

A	Q	OPERATION	count
# 000000000	11110000	Shift left A, Q	4
000000001	1110000$\boxed{?}$	A = A − M or A = A + (−M)	
		000000000 + 111111101 111111110	
111111110	1110000$\boxed{?}$	MSB [A] = 1 $Q_0 \leftarrow 0$	count = 3
000000001	1110 0000	Restore A	
# 0000 00001	1100000	Shift left A, Q	3
0000 00011	1100 000$\boxed{?}$	A = A + (−M)	
		0000 00011 + 1111 11101 1 0000 00000 Discard	
0000 00000	1100 000$\boxed{?}$	MSB [A] = 0	
000000000	1100 0001	$Q_0 \leftarrow 1$	count = 2

A	Q	OPERATION	count
000000000	1100 0001	Shift Left A, Q	2
0000 00001	100001[?]	A = A − M → A = A + (−M)	
		000000001 + 11111101 11111110	
1111 1110	100001[?]	MSB [A] = 1 $Q_0 \leftarrow 0$	
0000 00001	1000 0010	Restore A	count = 1
0000 00001	1000 0010	Shift Left A, Q	1
000000011	0000 010[?]	A = A − M → A = A + (−M)	
		0000 00011 + 11111101 1 0000 00000 Discard	
0000 00000	0000010[?]	MSB [A] = 0 $Q_0 \leftarrow 1$	
0000 00000 $(0)_{10}$ Remainder	00000101 $(5)_{10}$ Quotient		count = 0

Restoring Division Algorithm :- Division for Unsigned Integer.

Question for Practice Purpose:-

Question 0101 by 0011 dividend = 0101
divisor = 0011

Solution:-

Q = 0101

n = 4

count = 4

A = 00000

M = 0011

M ⇒ 00011
1's ⇒ 11100
2's ⇒ +1

11101

$\boxed{-M = 11101}$

Q ⇒ 0101 ⇒ $(5)_{10}$
M = 0011 ⇒ $(3)_{10}$

1 ← Quotient
3) 5
3
2
Remainder

A	Q	OPERATION	COUNT
00000 00000	0101 101 $\boxed{?}$	Shift left A, Q $A = A - M \rightarrow A = A + (-M)$	count = 4
11101	101 $\boxed{?}$	$\begin{array}{r} 00000 \\ +\ 11101 \\ \hline 11101 \end{array}$	
11101	1010	MSB [A] = 1 $Q_0 \leftarrow 0$ Restore A $A = A + M$	
00000	1010	$\begin{array}{r} 11101 \\ +\ 00011 \\ \hline 1\ 00000 \end{array}$ Discard	n = 3
00000 00001	1010 010 $\boxed{?}$	Shift left A, Q $A = A - M \rightarrow A = A + (-M)$	n = 3

A	Q	OPERATION	Count
		$\begin{array}{r} 00001 \\ +\ 11101 \\ \hline 11110 \end{array}$	
11110	010 $\boxed{?}$		
11110	0100	MSB [A] = 1 $Q_0 \leftarrow 0$ Restore A A = A+M	
		$\begin{array}{r} 11110 \\ +\ 00011 \\ \hline 1\,00001 \end{array}$	
00001	0100	Discard	n=2
00001	0100		
00010	100 $\boxed{?}$	Shift left A, Q	
		$A = A - M \rightarrow A = A + (-M)$	n=2
		$\begin{array}{r} 00010 \\ +\ 11101 \\ \hline 11111 \end{array}$	
11111	100 $\boxed{?}$		
		MSB [A] = 1 $Q \leftarrow 0$ Restore A A = A +M	

A	Q	OPERATION	Count
00010	1000	11111 + 00011 1 00010 Discard	n=1
00010 00101	1000 000?	Shift A, Q A = A − M 00101 + 11101 1 00010 Discard	n=1
00010	000?	MSB [A] ← 0	
0010	0001	$Q_0 \leftarrow 1$	n=0
Remainder	Quotient		
2	1		

Question :- Dividend = 1100 and division = 0011

Solution :- $Q = 1100$

$n = 4$

count = 4

$A = \boxed{00000}$

$M = 00011$

1's ⇒ 11100

2's ⇒ +1

11101

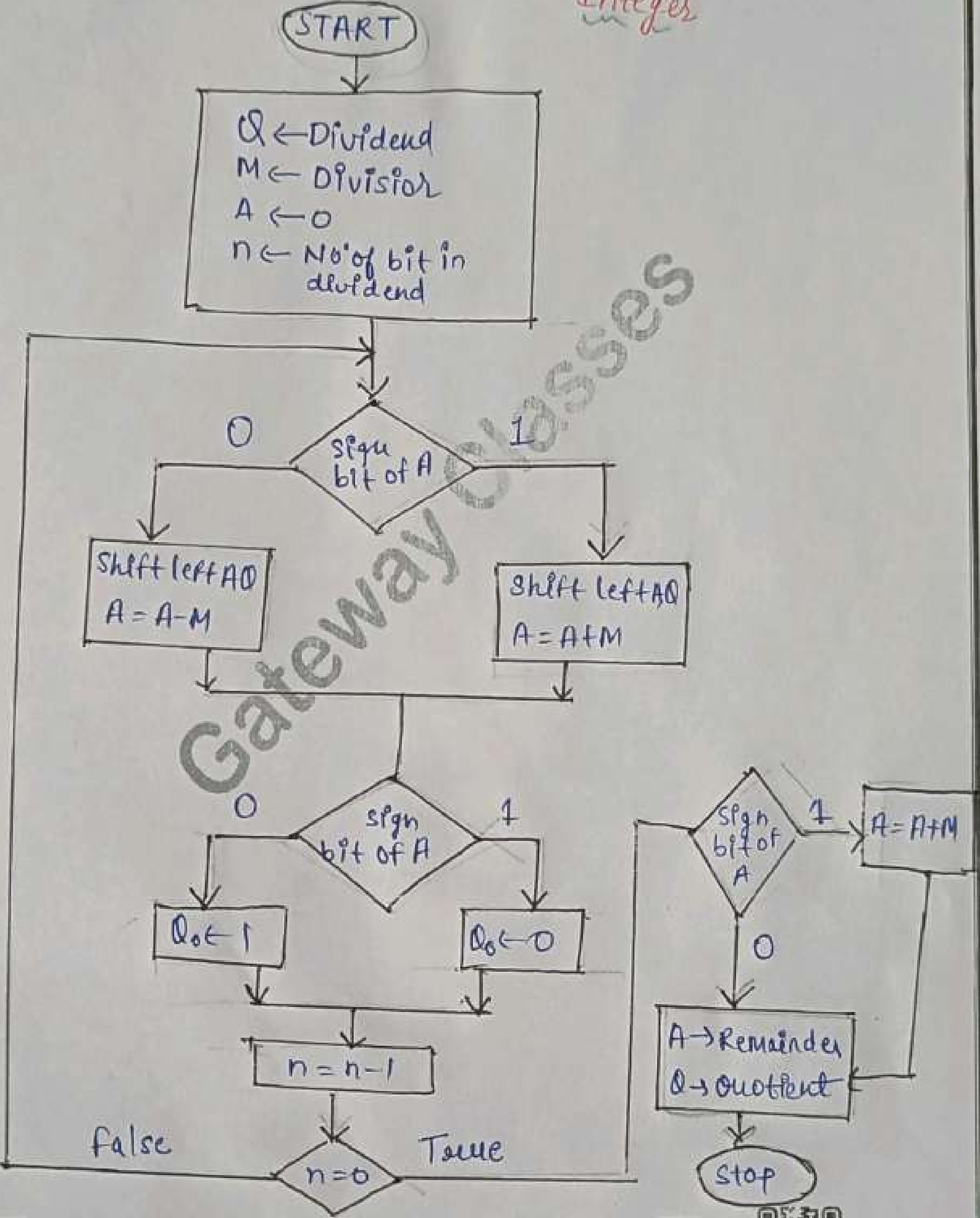
$\boxed{-M = 11101}$

A	Q	OPERATION	COUNT
00000 00000	0101 101 $\boxed{?}$	Shift left A,Q $A = A - M$ → $A = A + (-M)$ 00000 +11101 11101	$n = 4$
11101	101 $\boxed{?}$	MSB [A] = 1 $Q_0 \leftarrow 0$ Restore A $A = A + M$ 11101 00011 Discard (1) 00000	
00000	1010		count = 3

A	Q	OPERATION	Count
00000 → 00001	1010 → 010[?]	Shift left A,Q	n=3
		A = A−M	
		A = A+(−M) 00001 + 11101 = 11110	
		MSB[A]=1	
		$Q_0 \leftarrow 0$	
		Restore A = A+M	
11110	010[?]	11110 + 00011 = (1)00001	
11110	0100	Discard	
00001	0100		Count=2
00001 → 00010	0100 → 100[?]	Shift left A,Q	
		A = A−M →	n=2
		A = A+(−M)	
		00010 + 11101 = 11111	

A	Q	OPERATION	count
	0010	MSB[A] = 1 $Q_0 \leftarrow 0$ Restore A A = A+M	n = 2
00010	0010	11111 + 00011 1 00010 Discard	n = 1
00000 00000	0010 010 $\boxed{?}$	Shift left A, Q A = A - M A ⇒ A + (-M) 00000 11101 1 11101	n = 1
11101	010 $\boxed{?}$		
11101 00000 Remainder ⇒ 2	0100 0100 Quotient ⇒ 0	MSB [A] ← 1 $Q_0 \leftarrow 0$ Restore A A = A+M 11101 + 00011 1 00000 Discard	count = 0 n = 0

Non-Restoring Division Algoritm:- Division for Unsigned Integers



Question:- Non-Restoring Division Algorithm for Unsigned 11/3.

Solution:-

Binary of 11 ⇒ $(1011)_2$
Binary of 3 ⇒ $(0011)_2$

$Q \Rightarrow (11)_{10} \Rightarrow 1011$

$n = 4$

$\boxed{count = 4}$

$\boxed{A = 00000}$

$M = 00011$

$$\begin{array}{rr} M = & 00011 \\ 1's \Rightarrow & 11100 \\ 2's \Rightarrow & +1 \\ \hline & 11101 \\ \hline \end{array}$$

$\boxed{-M \Rightarrow 11101}$

Now, Booth's Algorithm Representation:-

Gateway Classes

COA | CS / IT / CS Allied / MCA

A	Q	OPERATION	COUNT
00000 00001	1011 011 $\boxed{?}$	Sign bit [A] = 0 Shift left A, Q A = A − M A → A + (−M) 00001 + 11101 11110	Count = 4
11110	011 $\boxed{?}$	Sign bit [A] = 1	
11110	0110	$Q_0 \leftarrow 0$	Count = 3
11110 11100	0110 110 $\boxed{?}$	Signed bit [A] = 1 Shift left A, Q A = A + M 11100 + 00011 11111	n = 3
11111	110 $\boxed{?}$	Signed bit [A] = 1	
11111	1100	$Q_0 \leftarrow 0$	Count = 2

A	Q	OPERATION	COUNT
11111 11111	1100 100□	Sign bit [A]=1 Shift left A,Q	n=2
		A = A+M $\begin{array}{r} 11111 \\ +\ 00011 \\ \hline 1\ 00010 \end{array}$ 1 → Discard	
00010	100□		
00010	1001	Signed bit [A]=0 $Q_0 \leftarrow 1$	count=1
00010 00101	1001 001□	Sign bit [A] = 0 Shift left A,Q A = A−M → A = A+(−M)	n=1
		$\begin{array}{r} 00101 \\ +\ 11101 \\ \hline 1\ 00010 \end{array}$ 1 → Discard	
00010	001□		
00010	0011	Sign bit [A]=0 $Q_0 \leftarrow 1$	Count=0
00010 Remainder	0011 Quotient	Sign bit [A] = 0	

Question:- Non Restoring Division Algorithm :- for for Unsigned number 1010 by 0101

Solution:-

$(1010)_2 \Rightarrow (10)_{10}$

$(0101)_2 \Rightarrow (5)_{10}$

→ Quotient

$$\begin{array}{r|l} 5 & 10 \;(2 \\ & 10 \\ \hline & 0 \end{array}$$

→ Remainder

Q = 1010

n = 4

$\boxed{count = 4}$

$\boxed{A = 00000}$

M = 00101

$$\begin{array}{rr} M = & 00101 \\ 1's \Rightarrow & 11010 \\ 2's \Rightarrow & +1 \\ \hline & 11011 \\ \hline \end{array}$$

$\boxed{-M = 11011}$

Now, using Booth's Algorithm for Division

A	Q	OPERATION	COUNT
00000	1010	Signed bit [A] = 0	4
00001	010$\boxed{?}$	Shift left A,Q	
		$A = A - M \rightarrow$	
		$A = A + (-M)$	
		00001 + 11011 = 11100	
11100	010$\boxed{?}$	Sign bit [A] = 1	count=3
11100	0100	$Q_0 \leftarrow 0$	
11100	0100	Signed bit [A] = 1	n = 3
11000	100$\boxed{?}$	Shift left A,Q	
		$A = A + M$	
		11000 + 00101 = 11101	
11101	100$\boxed{?}$	Sign bit [A] = 1	count=2
11101	1000	$Q_0 \leftarrow 0$	

A	Q	OPERATION	COUNT
11101	1000	Sign bit [A] = 1	n=2
11011	000[?]	Shift left A,Q	
		A = A+M	
		$\begin{array}{r} 11011 \\ +\ 00101 \\ \hline 1\ 00000 \end{array}$ Discard	
00000	000[?]	Sign bit [A] = 0	
		$Q_0 \leftarrow 1$	
00000	0001		Count=1
00000	0001	Signed bit [A] = 0	n = 1
00000	001[?]	Shift left A,Q	
		A = A−M → A = A+(−M)	
		$\begin{array}{r} 00000 \\ +\ 11011 \\ \hline 11011 \end{array}$	
11011	001[?]	Sign bit [A] = 1	Count=0
11011	0010	$Q_0 \leftarrow 0$	
		Sign bit [A] => 1	
Remainder	Quotient	A = A+M	
00000	0010	$\begin{array}{r} 11011 \\ +\ 00101 \\ \hline 1\ 00000 \end{array}$	n=0

Question:- Non Restoring Division Algorithm:- Division for Unsigned Integer

Dividend = 1010 , Divisor = 0011

Solution:-

$(1010)_2 \rightarrow (10)_{10}$

$(0011)_2 \rightarrow (3)_{10}$

3 → Quotient
3)10
-9
1 → Remainder

Q = 1010

n = 4

$\boxed{Count = 4}$

$\boxed{A = 00000}$

M = 00011

M = 00011
1's ⇒ 11100
2's ⇒ + 1
11101

$\boxed{-M = 11101}$

Now, using Booth's Algorthm:-

A	Q	OPERATION	COUNT
00000 00001	1010 010 $\boxed{?}$	Sign bit [A] = 0 Shift left A, Q A = A − M → A = A + (−M) $\begin{array}{r} 00001 \\ +\,11101 \\ \hline 11110 \end{array}$	n = 4
11110	010 $\boxed{?}$	Sign bit of A = 1 $Q_0 \leftarrow 0$	
11110	0100		count = 3
11110 11100	0100 100 $\boxed{?}$	Sign bit [A] = 1 Shift left A, Q A = A + M $\begin{array}{r} 11100 \\ +\,00011 \\ \hline 11111 \end{array}$	n = 3
11111	100 $\boxed{?}$	Sign bit of A = 1 $Q_0 \leftarrow 0$	
11111	1000		count = 2

A	Q	OPERATION	COUNT
11111	1000	Sign bit of A=1	n=2
11111	000[?]	Shift left A, Q	
		A = A+M	
		11111 +00011 (1)00010 ↳ Discard	
00010	000[?]		
		Sign bit [A]=0	count=1
00010	0001	$Q_0 \leftarrow 1$	
00010	0001	Sign bit [A] = 0	n=1
00100	001[?]	Shift left A, Q	
		A = A−M → A = A+(−M)	
		00100 +11101 (1)00001 ↳ Discard	
00001	001[?]		
		Sign bit [A] = 0	count=0
00001	0011	$Q_0 \leftarrow 1$	
Remainder $(1)_{10}$	Quotient $(3)_{10}$	Sign [A] = 0	n=0

Division for Unsigned Integer

Question:- Perform the division process of 00001111 by 0011 (used divided of 8 bits) [AKTU]

Solution

$(00001111)_2 \Rightarrow (15)_{10}$

$(0011)_2 \Rightarrow (3)_{10}$

$3 \overline{)15}$ 5 → Quotient
15
00 → Remainder

Q = 00001111

n = 8

count = 8

A = 0000 0000 0

M = 00000 0011

1's ⇒ 11111 1100
2's ⇒ +1

11111 1111

$\boxed{-M = 111111111}$

Now, Using Booth's Algorithm:-

A	Q	OPERATION	COUNT
11111101 11111010	01111000 1111000[?]	sign bit [A] = 1 Shift left A,Q A = A+M 11111010 + 00000011 11111101	n = 5
11111101	1111000☐	sign bit [A] = 1 $Q_0 \leftarrow 0$	
11111101	11110000		Count = 4
11111101 11111011	11110000 1110000[?]	sign bit [A] = 1 Shift left A,Q A = A+M 11111011 + 00000011 11111110	n = 4

A	Q	OPERATION	COUNT
1111 1101	0011 110[?]	Sign bit [A]=1 $Q_0 \leftarrow 0$	
11111101	00111100		count ⇒ 6
11111101	0011110	Sign bit [A] = 1	n = 6
11111010	011110[?]	Shift left A, Q A = A + M 11111010 + 00000011 11111101	
11111101	0111100[?]	Sign bit A = 1 $Q_0 \Rightarrow 0$	
11111101	01111000		count = 5

A	Q	OPERATION	COUNT
000000000 000000000	00001111 0001111$\boxed{?}$	Sign bit[A] = 0 Left shift A, Q A = A − M A → A + (−M) $\begin{array}{r} 000000000 \\ +\ 111111101 \\ \hline 111111101 \end{array}$	n = 8
111111101	0001111$\boxed{?}$	Sign bit [A] = 1 $Q_0 \leftarrow 0$	
111111101	00011110		Count = 7
111111101 111111010	00011110 0011110$\boxed{?}$	Sign bit[A] = 1 Shift left A, Q A = A + M $\begin{array}{r} 111111010 \\ +\ 000000011 \\ \hline 111111101 \end{array}$	n = 7

A	Q	OPERATION	COUNT
11111110	1110 000$\boxed{?}$	Sign bit [A] = 1 $Q_0 \leftarrow 0$	
11111110	1110000		count → 3
11111110 11111101	1110 0000 1100000$\boxed{?}$	Sign bit of A = 1 Shift left A, Q A = A+M 111111101 + 000000011 ① 000000000 → Discard	n = 3
000000000	1100 000$\boxed{\ }$	Sign bit of A = 0 $Q_0 \leftarrow 1$	
000000000	11000001		count = 2

A	Q	OPERATION	COUNT
000000000 0000 00001	1100 0001 1000 001$\boxed{?}$	Sign bit of A = 0 Shift left A,Q A = A - M A = A + (-M) 0000 00001 + 1111 11101 1111 11110 Sign bit of A = 1 $Q_0 \leftarrow 0$	n = 2
111111110 111111110	1000 001$\boxed{?}$ 1000 0010		count = 1
111111110 111111101	10000010 0000010$\boxed{\ }$	Sign bit of A = 1 Shift left A,Q A = A + M	n = 1

A	Q	OPERATION	COUNT
		1111 11101 + 0000 00011 (1)0000 00000 ↳ Discard	n=1
000000000	0000 010$\boxed{?}$	Signed bit of A=0 $Q_0 \leftarrow 1$	
000000000 Remainder $(0)_{10}$	00000101 Quotient $(5)_{10}$	Signed bit of A=0	count=0

Question:- An instruction is stored at location 400 with its address feild at location 401. The address-feild has the value 500. A Processor Register R_1 contain number 200. Evaluate the effective address if the addressing mode of the instruction is

(a) Direct
(b) Immediate
(c) Relative
(d) Register Indirect
(e) Index with register R_1 as Index Register

Solution:-

Addressing Mode	Effective Address
Direct	500
Immediate	401
Relative	402
Register Indirect	200
Index with register R_1 as Index Register	700

$\because$ Relative Effective Address $\Rightarrow$ PC + Address Part
402 + 500 = 902

$\because$ Index Register $\Rightarrow$ R_1 + Address Part
200 + 500 $\Rightarrow$ 700

IEEE Standard 754 Floating Point Numbers

The IEEE Standard for Floating - Point Arithmetic (IEEE 754) is a technical Standard for floating - point computation which was established in 1985 by the institute of Electrical and Electronics Engineer (IEEE)

1. The Sign of Mantissa :-

This is as simple as name 0 represents a Positive number while 1 represent a negative number

2. The Biased Exponent :

the exponent feild needs to represent both positive and Negative exponents. As bias is added to the actual exponent in order to get the Stored exponent.

3. Normalizing the Mantissa :-

It means adjusting a floating - point number so that the leading digit of the Mantissa is Non-zero (usually 1 in binary). This ensures the number is in Standard form, Maximizing Precision.

The Binary number 0.01101 is normalising to $1.101 \times 2^{(-2)}$ by shifting the decimal point and adjusting the exponent.

Single Precision IEEE 754 floating point standard

sign (1)	exponent (8)	Mantissa (23)

Total Bit ⇒ 32 bits

Double Precision IEEE 754 floating point standard

sign (1)	Exponent (11)	Mantissa (52)

Total Bits ⇒ 64 bits

Question for Practice

Ques:- Represent the floating point number using IEEE standard both single and double Precision.

① $(314.175)_{10}$

2	314	
2	157	0
2	78	1
2	39	0
2	19	1
2	9	1
2	4	1
2	2	0
2	1	0
		1

$0.175 \times 2 \Rightarrow 0.35 \rightarrow 0$
$0.35 \times 2 \Rightarrow 0.7 \rightarrow 0$
$0.7 \times 2 \Rightarrow 1.4 \rightarrow 1$
$0.4 \times 2 \Rightarrow 0.8 \rightarrow 0$
$0.8 \times 2 \Rightarrow 1.6 \rightarrow 1$
$0.6 \times 2 \Rightarrow 1.2 \rightarrow 1$
$0.2 \times 2 \Rightarrow 0.4 \rightarrow 0$
$0.4 \times 2 \Rightarrow 0.8 \rightarrow 0$
$0.8 \times 2 \Rightarrow 1.6 \rightarrow 1$
$0.6 \times 2 \Rightarrow 1.2 \rightarrow 1$
$0.2 \times 2 \Rightarrow 0.4 \rightarrow 0$

$(100111010.0010110)_2$

② Normalisation :-

$1.001110100010110 \times 2^8$

Single bit Precision

$1.N \times 2^{E-127}$

$E - 127 = 8$

$E = 127 + 8 = 135$

2	135	
2	67	1
2	33	1
2	16	1
2	8	0
2	4	0
2	2	0
2	1	0
		1

$(135)_{10}$

$(10000111)_2$

Question 2 :- $(-786.25)_{10}$

Binary of 786

2	786	
2	393	0
2	196	1
2	98	0
2	49	0
2	24	1
2	12	0
2	6	0
2	3	0
2	1	1
		1

$(1100000010)_2$

$.25 * 2 \Rightarrow 0.5 \rightarrow 0$
$.5 * 2 \Rightarrow 1.0 \rightarrow 1$
$.0 * 2 \Rightarrow 0$

$(786.25)_{10} \Rightarrow (1100000010.01)_2$

Normalisation :-
$1.10000001001 * 2^9$

Single bit Precision
$1.N * 2^{E-127}$
$E - 127 = 9$
$E = 127 + 9$
$E = 136$

2	136	0
2	68	0
2	34	0
2	17	1
2	8	0
2	4	0
2	2	0
2	1	1

$(10001000)_2$

← 1 bit →	← 8 bit →	← 23 →
Sign	Exponent	Mantissa
1	10001000	$\underbrace{10000001001}_{11}\ \underbrace{00\cdots0}_{12}$

Double Bit Precision :-

$1.N \times 2^{E-1023}$

$E - 1023 = 9$

$E = 9 + 1023$

$E = 1032$

2	1032	
2	516	0
2	258	0
2	129	0
2	64	1
2	32	0
2	16	0
2	8	0
2	4	0
2	2	0
2	1	0
		1

$(10000001000)_2$

← 64 bit →

← 1 bit →	← 11 bit →	← 52 bit →
Sign	Exponent	Mantissa
1	10000001000	$\underbrace{10000001001}_{11}\ \underbrace{00\cdots0}_{41}$

Ques $(12.21)_{10}$

Binary of 12 ⇒

2	12	
2	6	0
2	3	0
2	1	1
		1

$(1100)_2$

Binary of 0.21

$.21 * 2 = 0.42 \rightarrow 0$
$.42 * 2 = 0.84 \rightarrow 0$
$.84 * 2 = 1.68 \rightarrow 1$
$.68 * 2 = 1.36 \rightarrow 1$
$.36 * 2 = 0.72 \rightarrow 0$
$.72 * 2 = 1.44 \rightarrow 1$
$.44 * 2 = 0.88 \rightarrow 0$
$.88 * 2 = 1.76 \rightarrow 1$
$.76 * 2 = 1.52 \rightarrow 1$
$.52 * 2 = 1.04 \rightarrow 1$

$(0011010111)_2$

$(12.21)_{10} = (1100.0011010111)_2$

Normalisation of Number ⇒

$1.1000011010111 * 2^3$

For Single Bit Precision (32 bit)

$1.N * 2^{E-127}$

$E - 127 = 3$
$E = 127 + 3$
$E = 130$

2	130	
2	65	0
2	32	1
2	16	0
2	8	0
2	4	0
2	2	0
2	1	0
		1

⇒ $(10000010)_2$

← 1 bit →	← 8 bit →	← 23 bit →
Sign	Exponent	Mantissa
↓	↓	↓
0	10000010	$\underbrace{1000011010111}_{13\ \text{bit}}\ \underbrace{00\cdots0}_{10\ \text{bits}}$

Double Precision :- 64 bits

$$1.N * 2^{E-1023}$$

$$E-1023 = 3$$

$$E = 1023+3$$

$$E = 1026$$

$$(10000000010)_2$$

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
2	1	0
		1

← 1 bit →	← 11 bit →	← 52 bit →
Sign	Exponent	Mantissa
↓	↓	↓
0	10000000010	$\underbrace{1000011010111}_{13\ \text{bits}}\ \underbrace{000\cdots00}_{39\ \text{bits}}$

Question $(-456.1234)_{10}$

2	456	
2	228	0
2	114	0
2	57	0
2	28	1
2	14	0
2	7	0
2	3	1
2	1	1
		1

$.1234 * 2 \Rightarrow 0.2468 \rightarrow 0$

$.2468 * 2 = 0.4936 \rightarrow 0$

$.4936 * 2 = 0.9872 \rightarrow 0$

$.9872 * 2 = 1.9744 \rightarrow 1$

$.9744 * 2 = 1.9488 \rightarrow 1$

$.9488 * 2 = 1.8976 \rightarrow 1$

$(456.1234)_{10} \Rightarrow (111001000.000111)_2$

$$
\begin{array}{rl}
 & 111001000.000111 \\
1's \rightarrow & 000110111.111000 \\
2's \rightarrow & \underline{+1} \\
 & 000110111.111001
\end{array}
$$

$(-456.1234)_{10} \Rightarrow (000110111.111001)_2$

Single Precision :-

Normalisation of the Number

$0.00110111111001 * 2^8$

$1N * 2^{E-127}$

$E - 127 = 8$

$E = 127 + 8$

$E = 135$

2	135	
2	67	1
2	33	1
2	16	1
2	8	0
2	4	0
2	2	0
2	1	0
		1

$(135)_{10} \Rightarrow (10000111)_2$

←1 bit→	←8 bit→	←23 bit→
Sign	Exponent	Mantissa
↓	↓	↓
1	10000111	$\underbrace{110011001101}_{12 \text{ bits}}\ \underbrace{00\cdots00}_{11 \text{ bits}}$

Double Precision :- (64 bits)

$$1.N * 2^{E-1023}$$

$$E-1023 = 8$$

$$E = 1023+8$$

$$\boxed{E = 1031}$$

2	1031	
2	515	1
2	257	1
2	128	1
2	64	0
2	32	0
2	16	0
2	8	0
2	4	0
2	2	0
2	1	0
		1

$(10000000111)_2$

←1 bit→	←11 bits→	←52 bits→
Sign	Exponent	Mantissa
↓	↓	↓
1	10000000111	$\underbrace{110011001101}_{12 \text{ bits}}\ \underbrace{00\cdots00}_{40 \text{ bits}}$

Question :- $(-307.1875)_{10}$

2	307	
2	153	1
2	76	1
2	38	0
2	19	0
2	9	1
2	4	1
2	2	0
2	1	0
		1

$(100110011)_2$

$0.1875 * 2 \Rightarrow 0.3750 \rightarrow 0$

$0.3750 * 2 \Rightarrow 0.75 \rightarrow 0$

$0.75 * 2 \Rightarrow 1.5 \rightarrow 1$

$0.5 * 2 \Rightarrow 1.0 \rightarrow 1$

$0.0 * 2 \Rightarrow 0.0 \rightarrow 0$

$(0011)_2$

$$(307.1875)_{10} \Rightarrow (100110011.0011)_2$$

$$1's \Rightarrow 011001100.1100$$

$$2's \Rightarrow \quad +1$$

$$011001100.1101$$

$$(-3075.1875)_{10} \Rightarrow (011001100.1101)_2$$

Normalisation of the Number

$$0.11001100.1101 * 2^8$$

Single Bit Precision [32 bits]:

$$1N * 2^{E-127}$$

$$E-127 = 8$$

$$E = 127+8 = 135$$

2	135	
2	67	1
2	33	1
2	16	1
2	8	0
2	4	0
2	2	0
	1	0
		1

$(10000111)_2$

← 1 bit →	← 8 bit →	← 23 bit →
Sign	Exponent	Mantissa
1	10000111	$\underbrace{00110111111101}_{14\ bits}\ \underbrace{00...0}_{9\ bits}$

Double Precision:

$1N \times 2^{E-1023}$

$E - 1023 = 8$

$E = 1023 + 8$

$\boxed{E = 1031}$

← 1 bit →	← 11 bit →	← 52 bit →
Sign	Exponent	Mantissa
1	10000000111	$\underbrace{00110111111101}_{14\ bit}\ \underbrace{00...00}_{38\ bit}$

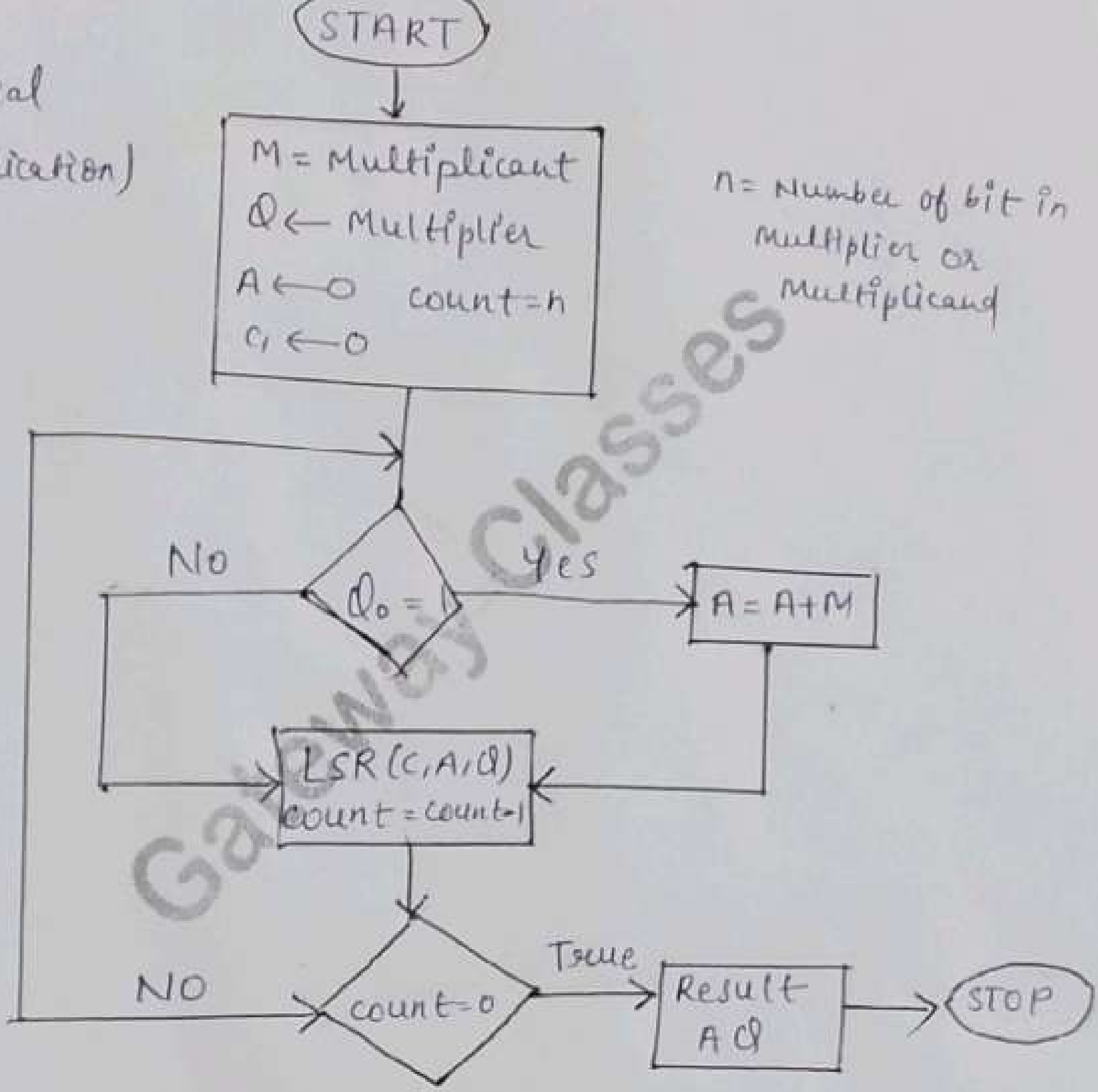
2	1031	
2	515	1
2	257	1
2	128	1
2	64	0
2	32	0
2	16	0
2	8	0
2	4	0
2	2	0
2	1	0
		1

$(10000000111)_2$

Multiplication for Unsigned Number:

(Sequential Multiplication)

n = Number of bit in Multiplier or Multiplicand



Question:- Multiply two unsigned number 1011 × 1101.

Solution:- M = 1011
Q = 1101
A = 0000
C = 0
Count = 4

C	A	Q	OPERATION	Count

0	0000	1101	$Q_0 = 1$
			A = A+M
			  0000
			+ 1011
			  1011
				count = 4

0	1011	1101	LSR (C, A, Q)	
0	0101	1110	lost	count = 3
0	0101	1110	$Q_0 = 0$ LSR (C, A, Q)	count = 3
0	0010	1111	lost	count = 2

C	A	Q	OPERATION	COUNT
0	0010	1111	$Q_0 = 1$ $A = A + M$ $\begin{array}{r} 0010 \\ +1011 \\ \hline 1101 \end{array}$	count = 2
0	1101	1111	LSR (C, A, Q)	
0	0110	1111	lost	count = 1
0	0110	1111	$Q_0 = 1$ $A = A + M$ $\begin{array}{r} 0110 \\ +1011 \\ \hline 10001 \end{array}$	count = 1
1	0001	1111	LSR (C, A, Q)	
0	1	1111	lost	

$100011111 \Rightarrow 128 + 8 + 4 + 2 + 1 = 143$ soln

Question:- Multiplication of two Unsigned Numbers

$6 * 14$

Solution :- $M = 0110$

$Q = 1110$

$A = 0000$

$C \leftarrow 0$

count = 4

$\therefore 6 * 14 = 84$

C	A	Q	OPERATION	COUNT
0	0000	1110	$Q_0 = 0$ LSR (C, A, Q)	count = 4
0	0000	0111	Lost	count = 3
0	0000	0111	$Q_0 = 1$, $A = A + M$ $0000 + 0110 = 0110$	count = 3
0	0110	0111	LSR (C, A, Q)	
0	0011	0011	Lost	count = 2

C	A	Q	OPERATION	COUNT
0	0011	0011	Q = 1 A = A + M 0011 + 0110 1001	count = 2
0	1001	0011	LSR (C, A, Q) lost	
0	0100	1001		count = 1
0	0100	1001	$Q_0 = 1$ A = A + M 0100 + 0110 1010	count = 1
0	1010	1001	LSR (C, A, Q) lost	
0	0101	0100		

001010100 ⇒ 64 + 16 + 4 = 84

HALF ADDER

- A half adder is a digital logic circuit that performs binary addition of two single-bit binary numbers. It has two inputs, A and B, and two outputs, SUM and CARRY. The SUM output is the (LSB) least significant Bit of the result, while the carry output is the most significant bit (MSB) of the result, indicating whether there was a carry-over from the addition of the two inputs. The Half adder can be implemented using basic Gates such as XOR and AND Gates.

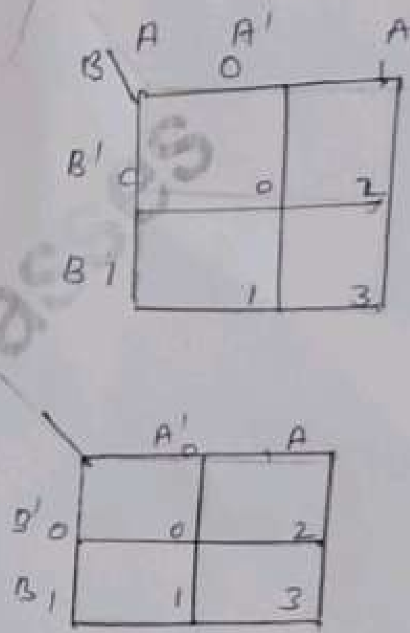
- Half adder is the simplest of all adder circuits. Half adder is a combinational arithmetic circuit that adds two numbers and produces a sum bits (S) and carry bit (c) both as output. The addition of 2 bits is done using a combinational circuit called Half adder. The input variables are augend and addened bits and output variables are sum and carry bits. A and B are the two input bits.

- Let us consider two input bits A and B, the sum bit (s) is the X-OR of A and B. It is evident from the function of a half adder that it requires one X-OR Gate and one And Gate for its construction.

For making a K-Map if n variable is there then we need 2^n cells.

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

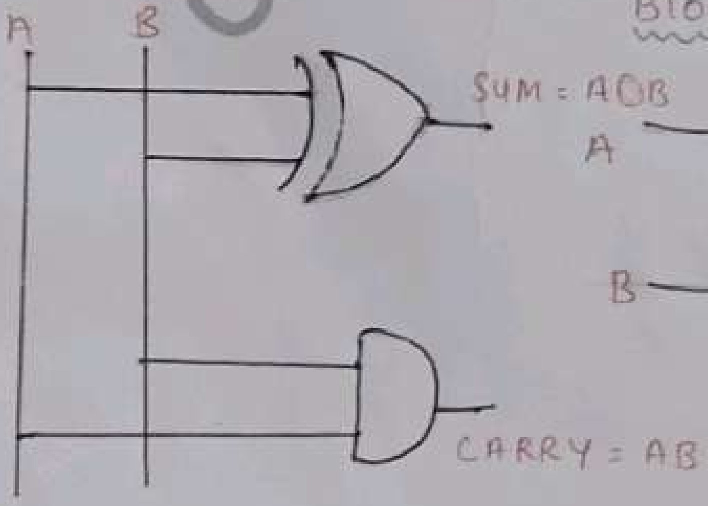
$\boxed{\text{K-MAP}}$



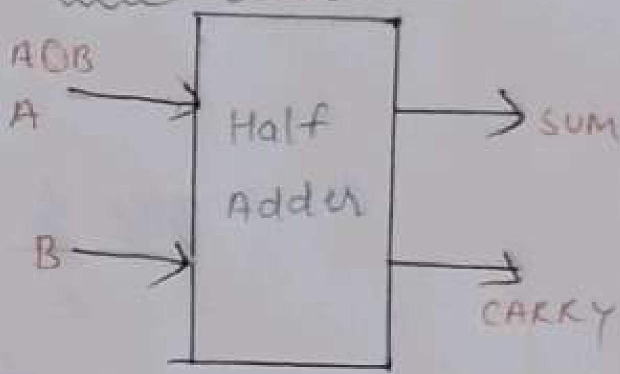
SUM = A'B + AB' (XOR GATE)

$\boxed{\text{CARRY} = AB}$

Implementation



Block Diagram



FULL ADDER

⇒ The Half adder is used to add only two numbers. To overcome this problem full adder was developed. The full adder is used to add three 1-bit binary-numbers A, B and carry C. The full Adder has three input states and two output states i.e. SUM and Carry.

INPUT			OUTPUT	
A	B	C	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

- It solve the problem of Half adder (not adding carry bit)
- Full adder is used to add three 1 bit number
- A, B, C in (carry).

Full Adder (for sum)

C_{in} \ AB	$\overline{A}\,\overline{B}$ 00	$\overline{A}B$ 01	AB 11	$A\overline{B}$ 10
$\overline{C}_{in}$ 0	0	1 (2)	(6)	1 (4)
C_{in} 1	1 (1)	(3)	1 (7)	(5)

$\overline{A}\,\overline{B}C_{in} + AB\overline{C}_{in} + ABC_{in}\ A\overline{B}\,\overline{C}_{in}$

$\overline{A}(\overline{B}C_{in} + B\overline{C}_{in}) + A(BC_{in} + \overline{B}\,\overline{C}_{in})$

$\overline{A}(B\oplus C) + A(\overline{B\oplus C})$

$\overline{A}X + A\overline{X}$

$A\oplus X \longrightarrow \boxed{A\oplus B\oplus C}$

$\because A\oplus B = A\overline{B} + \overline{A}B$

$\because \overline{(A\oplus B)} = AB + \overline{A}\,\overline{B}$

Full Adder (for carry)

C_{in} \ AB	$\overline{A}\,\overline{B}$ 00	$\overline{A}B$ 01	AB 11	$A\overline{B}$ 10
0	(0)	(2)	1 (6)	(4)
1	(1)	1 (3)	1 (7)	1 (5)

$AB\bar{C}_{in} + ABC_{in}$

$AB(\bar{C}_{in} + C_{in})$

$AB \cdot 1 = AB$

$\bar{A}BC_{in} + ABC_{in}$

$BC_{in}(\bar{A}+A)$

BC_{in}

Carry $\Rightarrow$ $AB + BC_{in} + AC_{in}$

$ABC_{in} + A\bar{B}C_{in}$

$AC_{in}(B+\bar{B})$

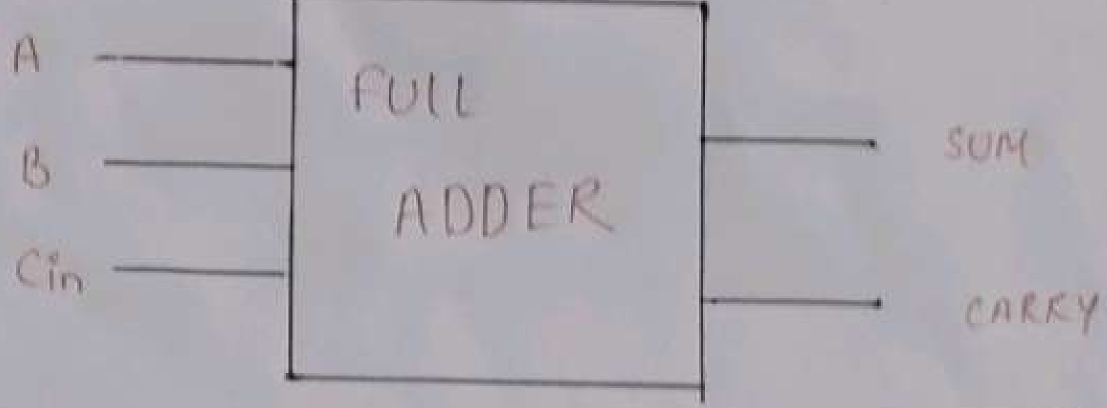
AC_{in}

$\bar{C}_{in} \rightarrow 0$

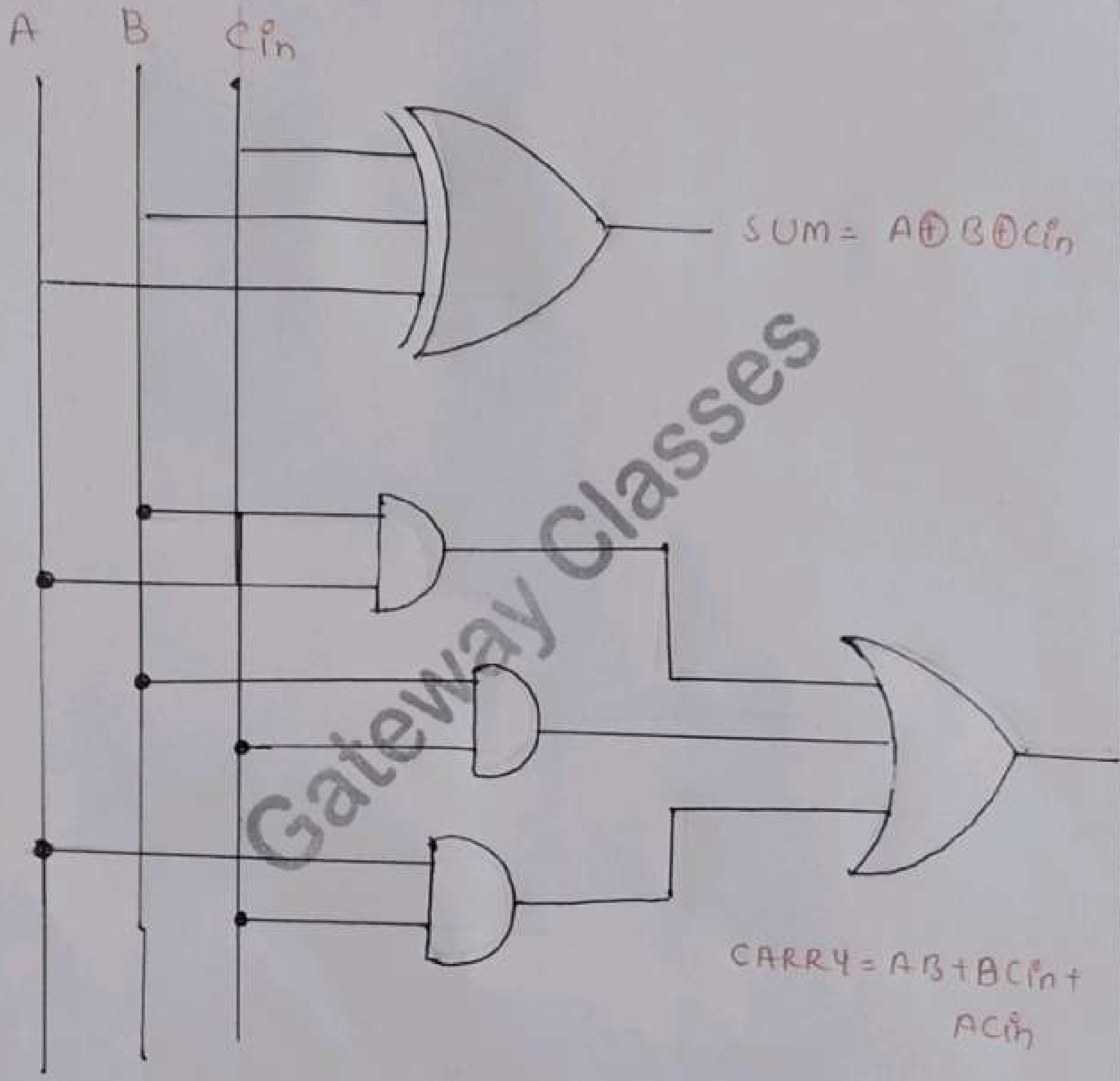
$C_{in} \rightarrow 1$

It's Implemention is on Next Page.

Block Diagram of FULL ADDER



Implementation of full Adder



→ In the above table [full Adder)

- 'A' and 'B' are the input variables. these variable represents the two significant bits which are going to be added.

- 'Cint' is the third input which represents the carry. from the previous lower significant position, the carry bit is fetched.

- the 'sum' and 'carry' are the output variables that defines the output values.

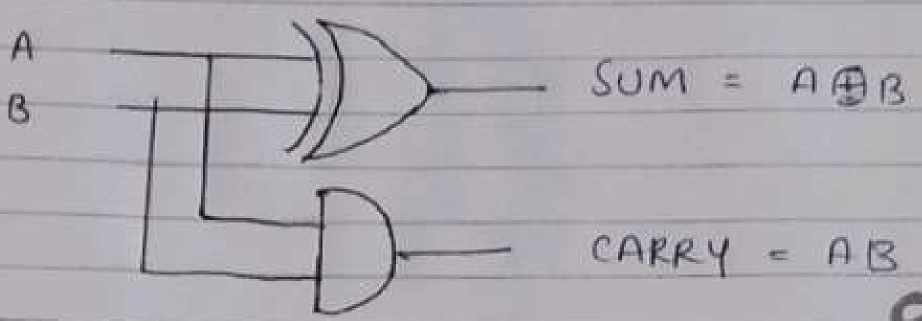
- the Eight rows under the input varibles designate all possible combinations of 0 and 1 that can occurs in these variables.

then finally, the ouput of this will implemented with the help of Gates.

Moreover here, the combination of AND Gate OR Gate and the XOR Gate is used to represent these Expression on the Paper and in the Practical Manner also.

Full Adder Using Half Adder

Half Adder :-

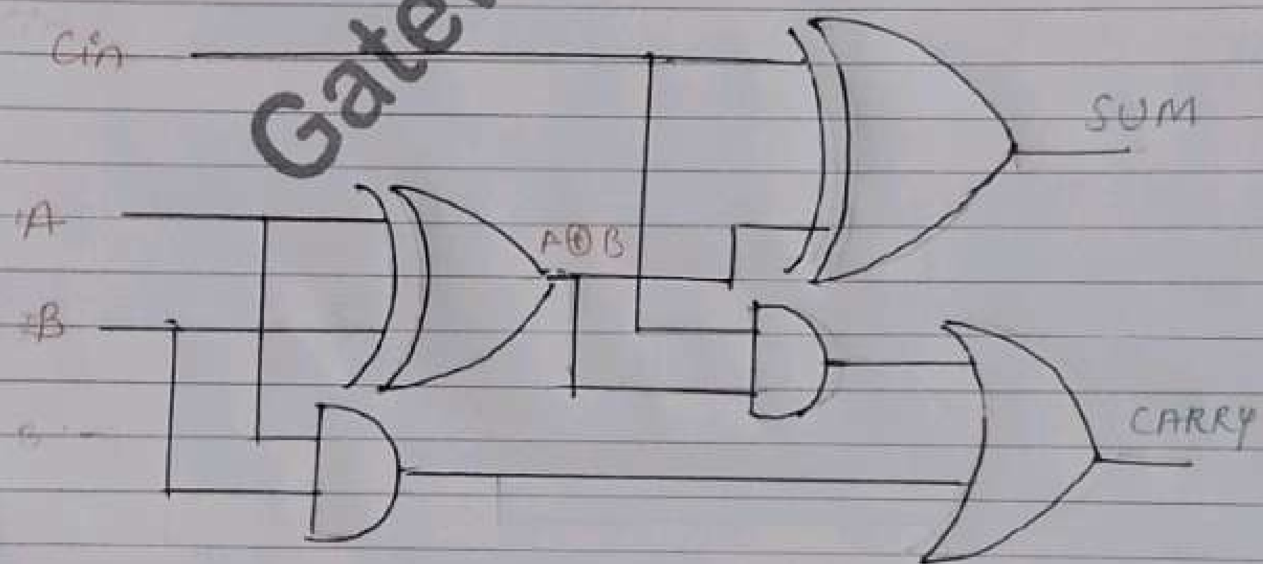


Full Adder :-

$$SUM = A \oplus B \oplus C$$

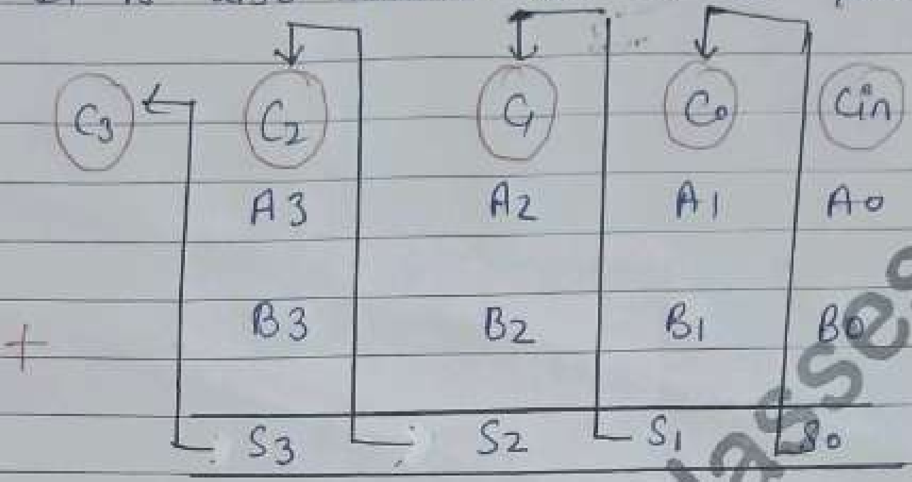
$$CARRY = AB + BC_{in} + AC_{in}$$

Implementation =)



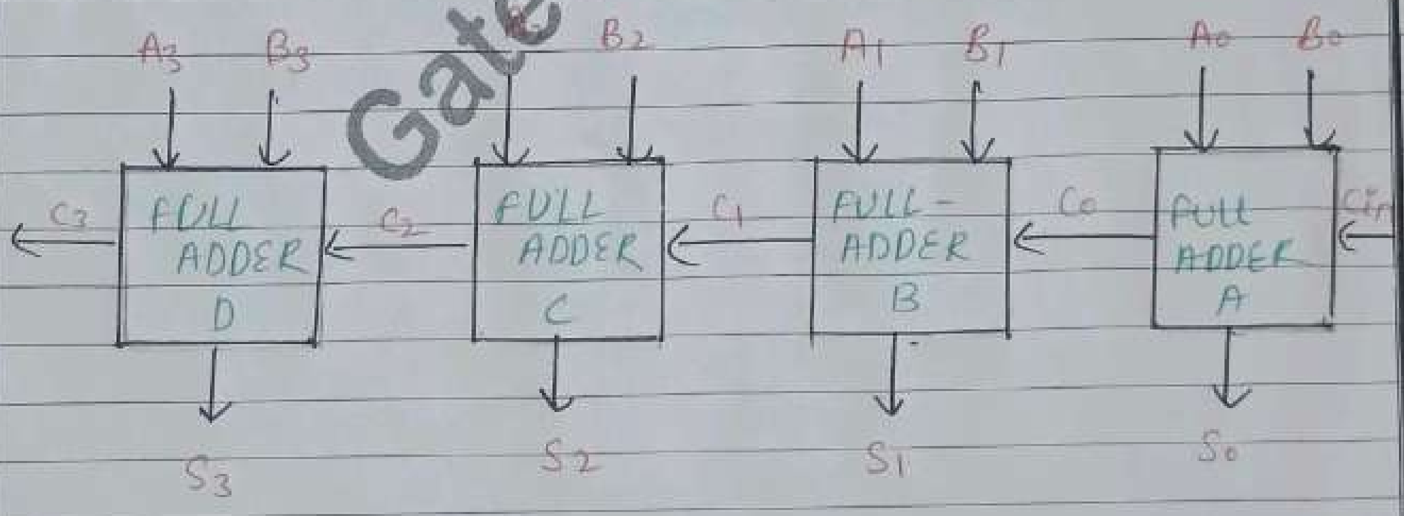
It requires n full adders in its circuit for adding two n bit numbers.

It is also known as n bit parallel Adder.



Addition two 4 digit Numbers

RIPPLE CARRY ADDER:



4- Bit Ripple Carry Adder

RIPPLE CARRY ADDER

RIPPLE CARRY ADDER works in different stages.

Each full adder takes the carry-in as input and produces carry out and sum bit as output.

the carry out produced by a full Adder serve as carry in for its adjust most significant full adder.

when carry-in becomes available to the full adder, it activates the full adder.

After full adder becomes activated, it comes into operation.

Disadvantages of Ripple Carry Adder:

Ripple carry Adder does not allow to use all the full adder simultaneously.

Each full adder has to necessarily wait until the carry bit becomes available from its adjacent full adder.

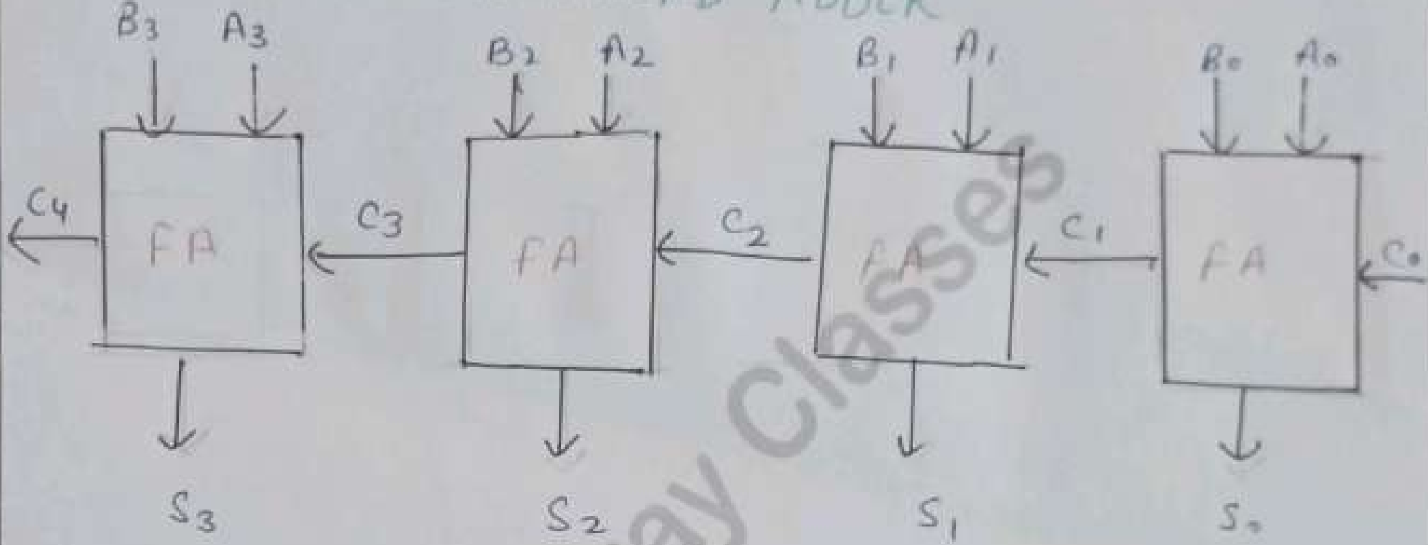
This increases the propagation time.

Due to this reason, ripple carry adder becomes

Extremely slow.

This is considered to be the biggest disadvantages of using ripple carry adder.

CARRY LOOK AHEAD ADDER



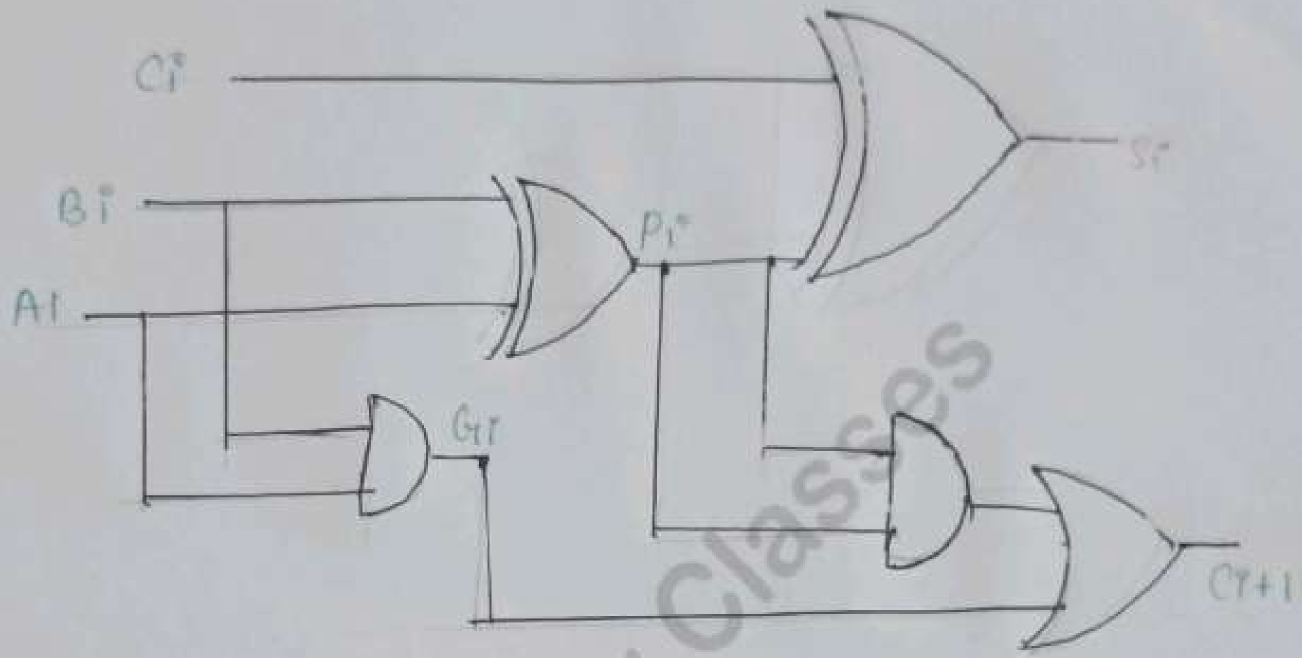
(For 4 bits)

Here, C_4 depends on C_3

C_3 depends on C_2

C_2 depends on C_1

C_1 depends on C_0



$P_i = A_i \oplus B_i \longrightarrow$ Carry Propogator

$G_i = A_i B_i \longrightarrow$ Carry Generator

$A_i \oplus B_i$

$\Rightarrow A\bar{B} + \bar{A}B$

$$\boxed{\begin{array}{l} S_i \Rightarrow P_i \oplus C_i \\ C_{i+1} \Rightarrow G_i + P_i C_i \end{array}}$$

Ai	Bi	Ci	Si	Ci+1
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

$$\boxed{C_1 = G_0 + P_0 C_0}$$

$$C_2 = G_1 + P_1 C_1$$

$$G_1 + P_1 (G_0 + P_0 C_0)$$

$$\boxed{C_2 = G_1 + P_1 G_0 + P_0 P_1 C_0}$$

$$C_3 = G_2 + P_2 C_2$$

$$G_2 + P_2 (G_1 + P_1 G_0 + P_0 P_1 C_0)$$

$$\boxed{C_3 = G_2 + P_2 G_1 + P_1 P_2 G_0 + P_0 P_1 P_2 C_0}$$

$$S_i = P_i \oplus C_i$$

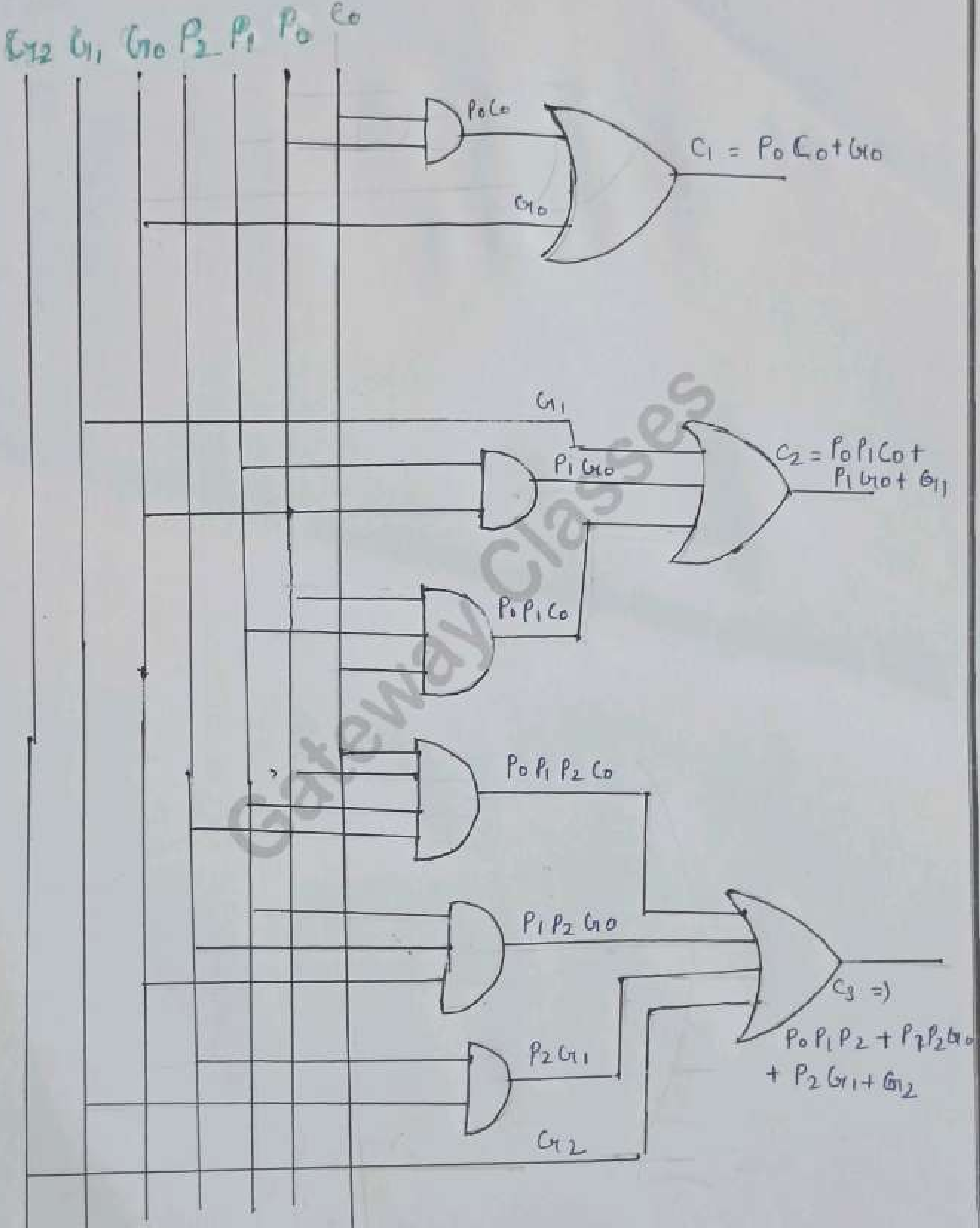
$$C_{i+1} = G_i + P_i C_i$$

$i = 0$ $\quad S_0 = P_0 \oplus C_0$ $\quad C_1 = G_0 + P_0 C_0$

$i = 1$ $\quad S_1 = P_1 \oplus C_1$ $\quad C_2 = G_1 + P_1 C_1$

$i = 2$ $\quad S_2 = P_2 \oplus C_2$ $\quad C_3 = G_2 + P_2 C_2$

$i = 3$ $\quad S_3 = P_3 \oplus C_3$ $\quad C_4 = G_3 + P_3 C_3$



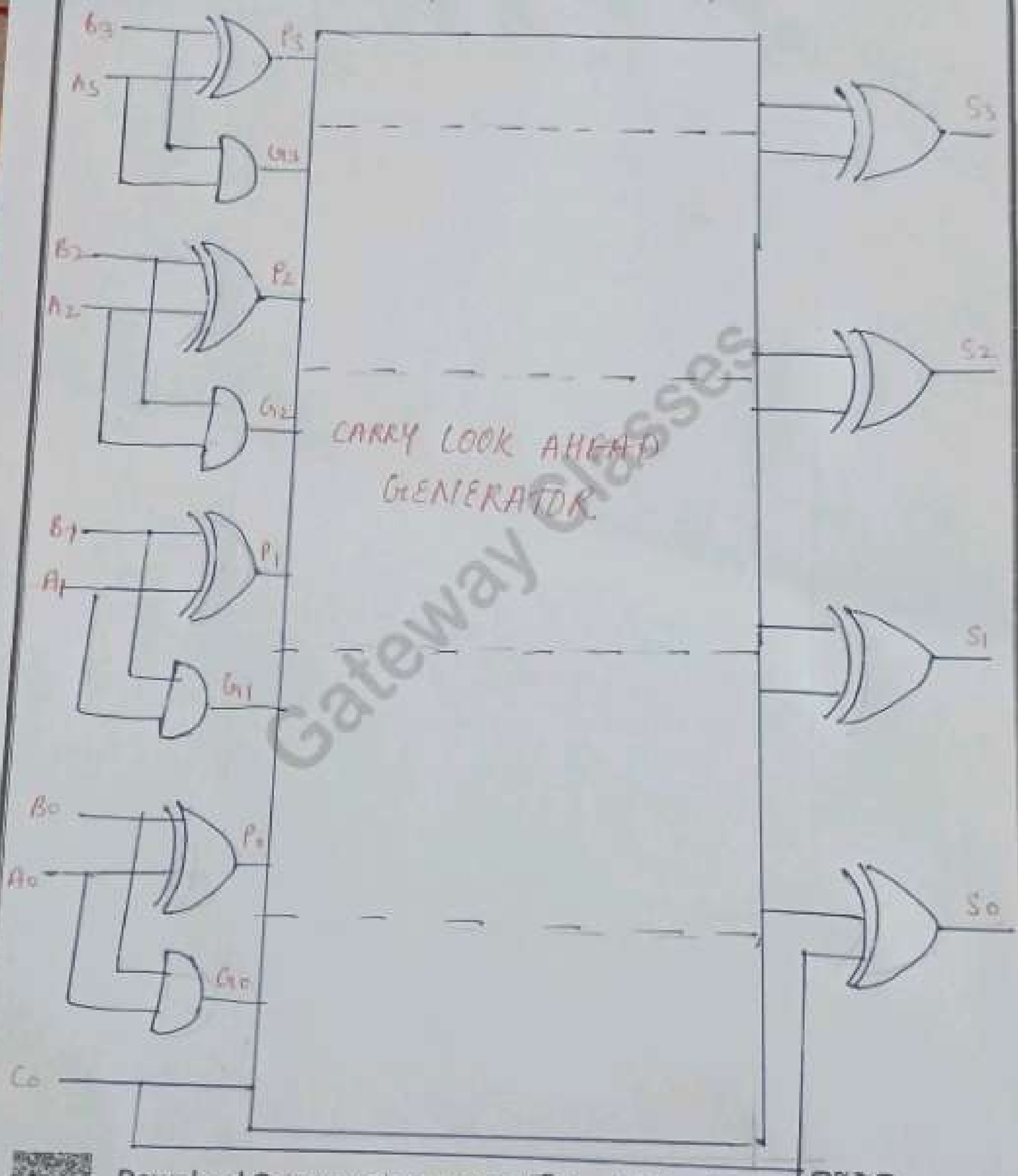
G2 G1 G0 P2 P1 P0 C0
P0C0
G0
C1 = P0C0+G0
G1
P1G0
C2 = P0P1C0 + P1G0 + G1
P0P1C0
P0P1P2C0
P1P2G0
C3 =)
P0P1P2 + P1P2G0 + P2G1 + G2
P2G1
G2

COA

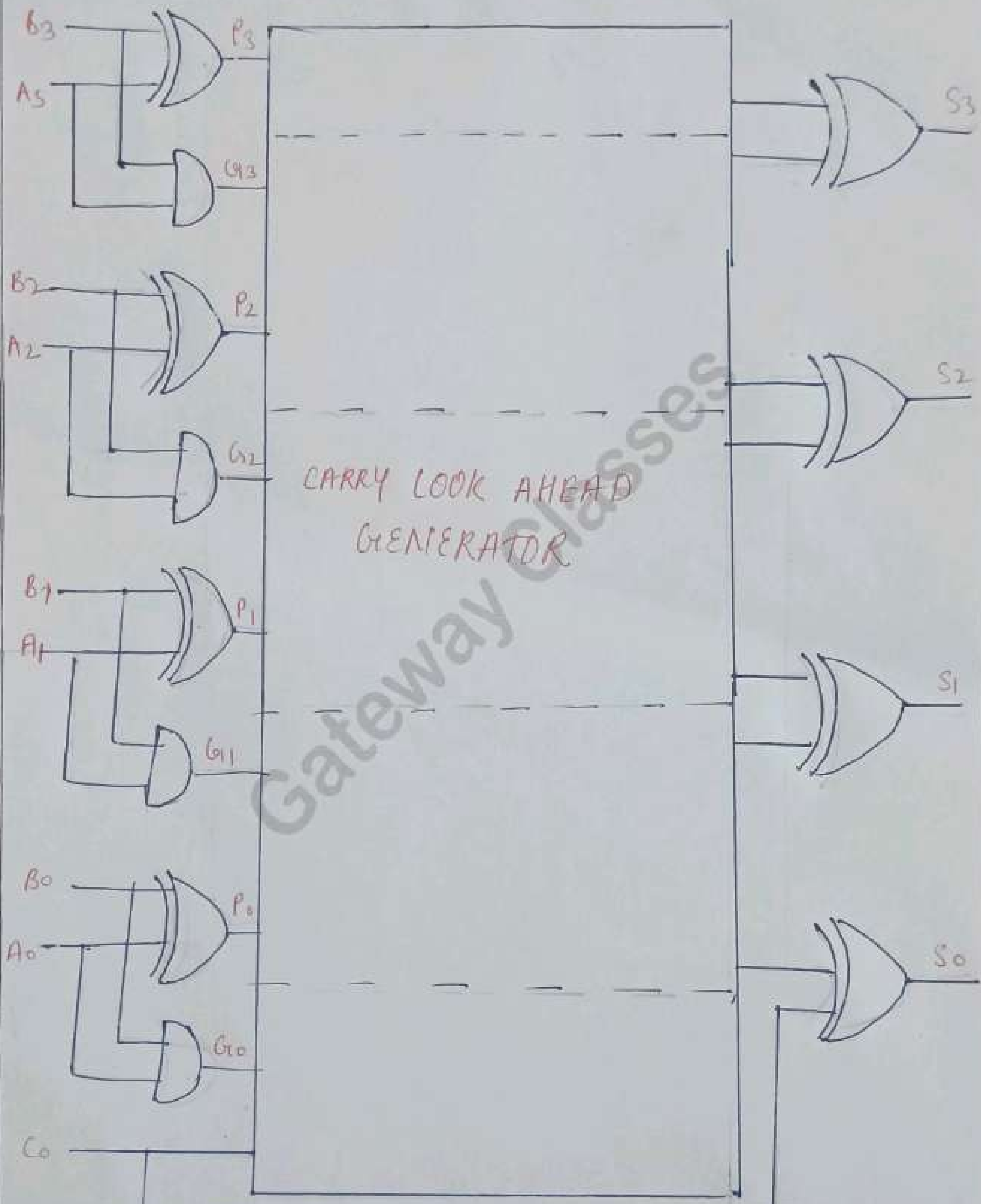
Gateway Classes

4- Bit Carry Look Ahead Adder

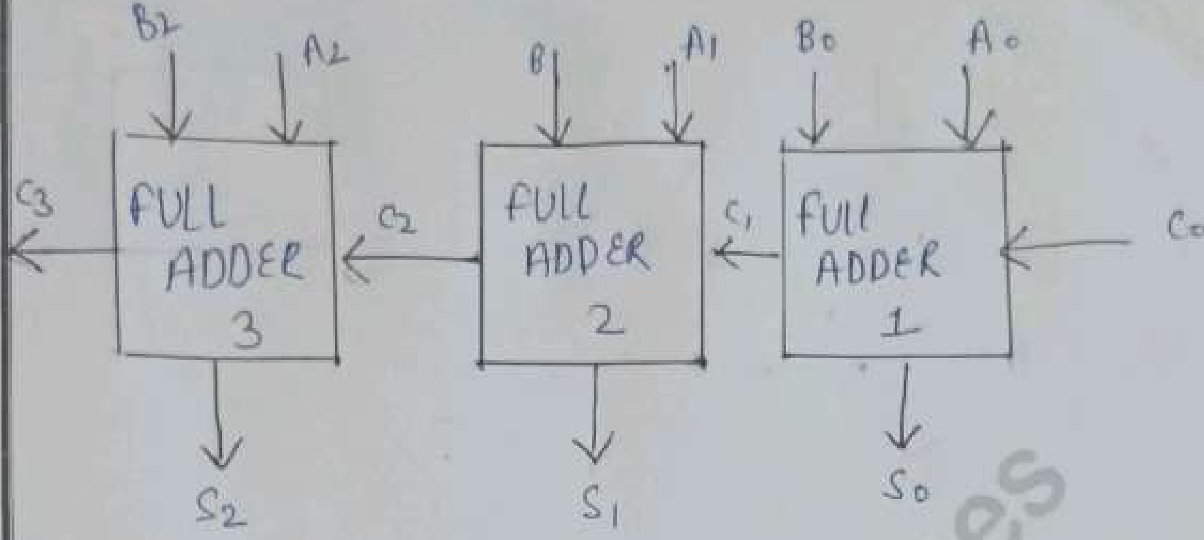
CS / IT / CS Allied / MCA

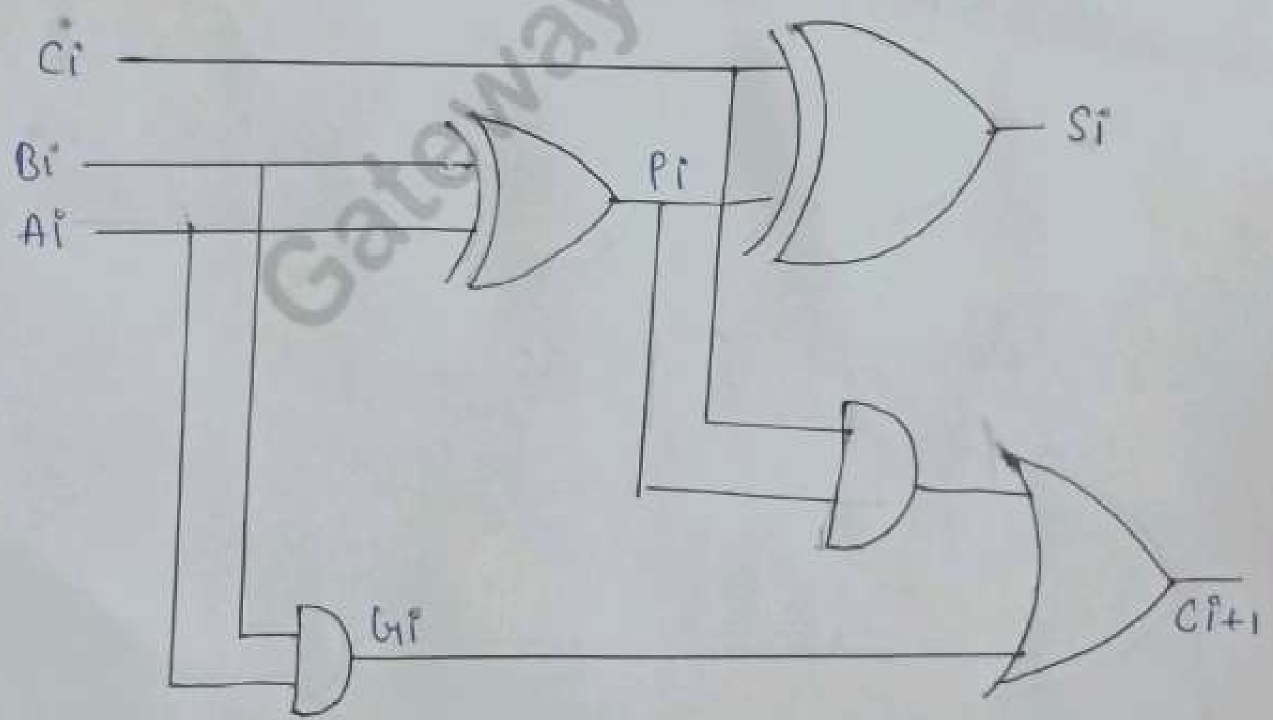


4- Bit Carry Look @head Adder



3 Bit Carry Look ahead adder





A_i	B_i	C_i	S_i	C_{i+1}
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

$P_i = A_i \oplus B_i$

$\overline{A_i}B_i + \overline{B_i}A_i$

$G_i = A_i B_i$

$S_i = P_i \oplus C_i$

$i=0 \quad S_0 = P_0 \oplus C_0$

$i=1 \quad S_1 = P_1 \oplus C_1$

$i=2 \quad S_2 = P_2 \oplus C_2$

$C_{i+1} = P_i C_i + G_i$

$i=0$

$\boxed{C_1 = P_0 C_0 + G_0}$

$i=1$

$C_2 = P_1 C_1 + G_1$

$P_1(P_0 C_0 + G_0) + G_1$

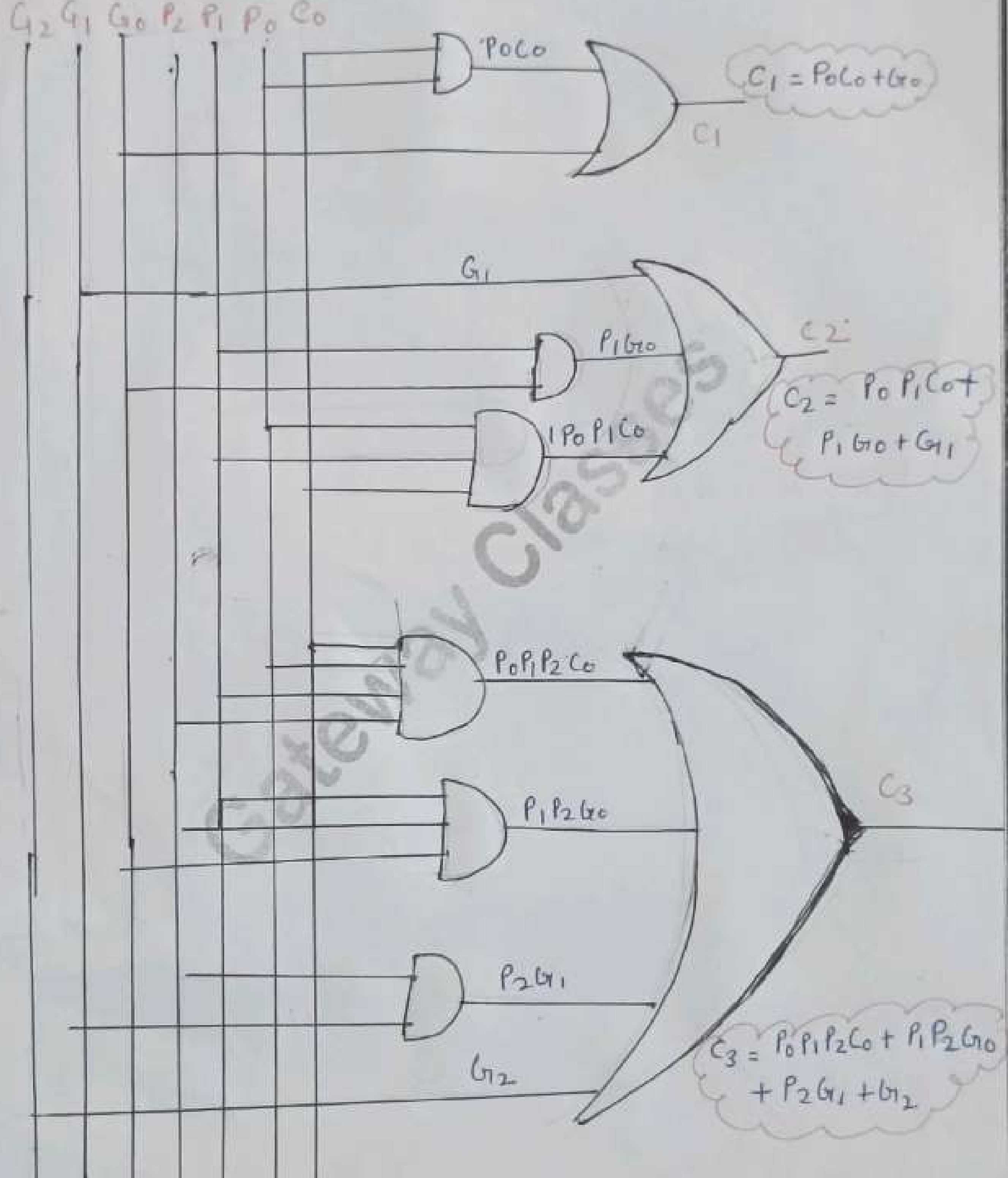
$\boxed{C_2 = P_0 P_1 C_0 + P_1 G_0 + G_1}$

$i=2 \quad C_3 = P_2 C_2 + G_2$

$P_2(P_0 P_1 C_0 + P_1 G_0 + G_1) + G_0$

$\boxed{C_3 = P_0 P_1 P_2 C_0 + P_1 P_2 G_0 + P_2 G_1 + G_2}$

Implementation :-



Carry look ahead Generator

4- Bit Carry Look @head Adder