UNIT-02

COMPUTER ARITHMETIC OPERATIONS

DATE : __/__/__
PAGE : _____

* Logic Gates :-

↳ Logic gates are main structural part of digital system.
↳ Logic gates are hardware that produces signals of binary, i.e. '0', '1'.
↳ Each gates has distinct graphical symbol.
↳ There are seven gates in digital system :-
AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR

Basic gate — Universal / Derived gate

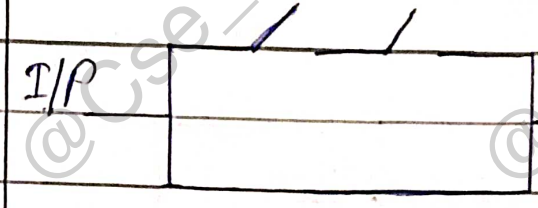
* AND Gate (.) :-

↳ Output of NAND gate is low or zero or false whenever one of the input is zero / low / false.
↳ Output of NAND gate is high or 1 or true whenever all the inputs are high / 1 / true.

Truth Table

A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

Circuit Diagram



DATE : __/__/__
PAGE : _____

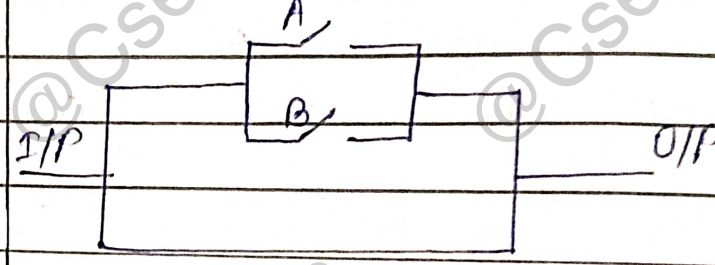
* OR gate (+) :- A, B → A+B

Output of the OR gate is 0 whenever all the inputs are zero.
Output of the OR gate is 1 whenever any one of the input is 1.

Truth Table

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Circuit diagram



* NOT Gate (¯) :- [I/P] A → $\bar{A}$ / A' [O/P]

↳ It is also called inverter.
↳ It perform 'unary' operations, i.e, if we provide input as 0 then it will produce O/P as 1.
↳ If we provide input as 1 then it produce O/P as 0.

Truth Table

A	$\bar{A}$
0	1
1	0

DATE : __/__/____
PAGE : _____

* NAND Gate :-

↳ The O/P of NAND gate is 0, iff both the input is 1.

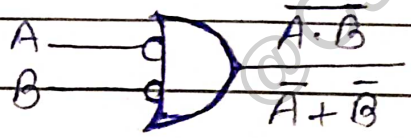
Truth Table

A	B	A·B	$\overline{A \cdot B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

$\overline{A \cdot B} = \bar{A} + \bar{B}$

AND + NOT

NOT + AND



Bubbled OR Gate

* NOR Gate :-

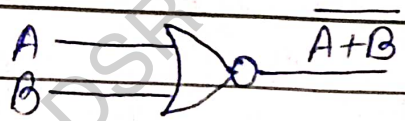
↳ The O/P of NOR Gate is 1, iff both the I/P is 0.

Truth Table

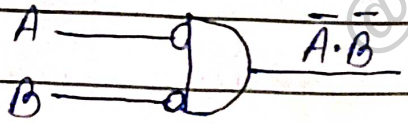
A	B	A+B	$\overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

$\overline{A+B} = \bar{A} \cdot \bar{B}$

OR + NOT

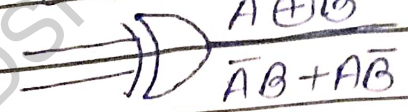


NOT + AND



Bubbled AND gate

* EX-OR / XOR (Exclusive OR) :—



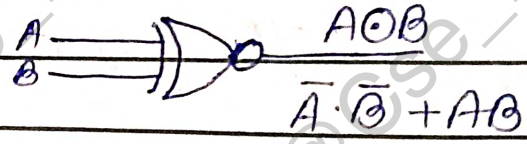
Truth Table

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

↳ O/P of XOR gate is 1, if one of its I/P is 1 but not both.

↳ It is also called Odd function gate because result is 1 whenever odd number of variables are 1.

* EX-NOR / XNOR :—



↳ It is inverse of XOR.

↳ The O/P of XNOR gate is 0, if one of its I/P is 1 but not both.

↳ It is also called equality checking gate.

Truth Table

A	B	A ⊙ B
0	0	1
0	1	0
1	0	0
1	1	1

DATE : __/__/____
PAGE : _____

Boolean Algebra

↳ It is used to analize and simplify the digital circuit.
It uses only '0' and '1'.
It is also called binary algebra or logical algebra.
It is invented by "George Boole" in '1854'.

Laws :-

① Commutative :-

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

(b) $A \cdot B = B \cdot A$

② Associative :-

(a) $A \cdot (B \cdot C) = (A \cdot B) \cdot C$

(b) $A + (B+C) = (A+B) + C$

③ Distributive :-

(a) $A \cdot (B+C) = A \cdot B + A \cdot C$

(b) $A + (B \cdot C) = (A+B) \cdot (A+C)$

④ AND :-

(a) $A \cdot 0 = 0$

(b) $A \cdot 1 = A$

(c) $A \cdot A = A$

(d) $A \cdot \bar{A} = 0$

⑤ OR :—

(a) $A + 0 = A$

(b) $A + 1 = 1$

(c) $A + A = A$

(d) $A + \bar{A} = 1$

⑥ Inversion :—

$\bar{\bar{A}} = A$

⑦ De Morgan's law :—

→ The complement result of the AND operation is equal to OR operation of the complement of that variable.

$\overline{A \cdot B} = \bar{A} + \bar{B}$

Ex :—

$\overline{0 \cdot 1} = \bar{0} + \bar{1}$

$\bar{0} \quad 1 + 0$

$1 = 1$

→ The complement result of OR operation of the AND operation of the complement of that variable.

$\overline{A + B} = \bar{A} \cdot \bar{B}$

Ex :—

$\overline{0 + 1} = \bar{0} \cdot \bar{1}$

$0 = 0$

DATE : __/__/____
PAGE : _____

* Data Representation :-

↳ Data representation refers to diff. format of data.

Number System

↳ For computer, everything is a number, i.e, alphabet, picture, video etc.
↳ It is categories in four types :-

i) Binary :-

↳ It contains only two variables that is '0' and '1'.
↳ It's base is 2.

ii) Octal

↳ It represents 8 values that is (0, 1, 2 ---- 7).
↳ It's base is 8.

iii) Decimal

↳ It represents 10 values.
↳ It's base is 10.

iv) Hexadecimal

↳ It represents 16 values. It's base is 16.

	System	Base	Digits
i)	Binary	2	0, 1
ii)	Octal	8	0, 1, 2, 3, 4, 5, 6, 7
iii)	Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
iv)	Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F ↓ ↓ ↓ ↓ ↓ ↓ 10 11 12 13 14 15

Data Format / Representation

- Fixed Point data
  - Magnitude format
    - Unsigned (only +ve)
    - Signed (+ve & -ve)
  - Complement format
    - 1's complement (+ve & -ve)
    - 2's complement (+ve & -ve)
- Floating Point data
  - Single Precision (34-bit data)
  - Double Precision (64-bit data)

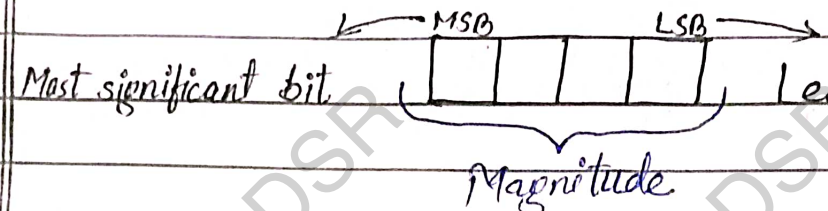
a) Unsigned

↳ It represents only +ve data.

↳ An unsigned data type simply means that the

DATE : __/__/____
PAGE : _____

↳ data type will only hold +ve values.
-ve values are not allowed to be stored in this data type.

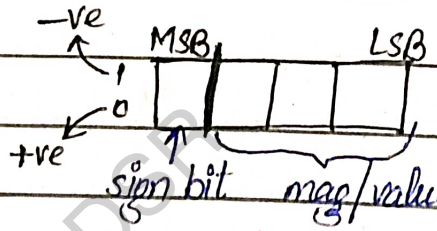


n-bit Range :- $\{0 \text{ to } 2^n - 1\}$

Ex:- 4-bit range - {0 to 15}
5-bit range - {0 to 31}

b) Signed (+ve & -ve)

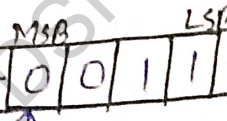
↳ When we want to store both +ve and -ve numbers, then we use signed data format.



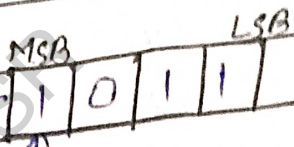
↳ When an integer binary number is +ve, the sign is represented by 0. and the magnitude by a +ve binary number.

↳ When the number is -ve, the sign is represented by 1.

4-bit :-



→ +3



→ -3

n-bit range :- $\{-(2^{n-1}-1)$ to $+(2^{n-1}-1)\}$

Ex:-

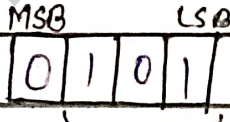
4-bit range :- $\{-7$ to $+7\}$

* 1's complement :-

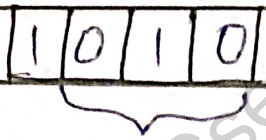
↳ +ve numbers are represented in same way as they are represented in sign magnitude.

↳ -ve numbers are represented using 1's complement

4-bit →



+ve → +5



-ve → -5

n-bit Range :- $\{-(2^{n-1}-1)$ to $+(2^{n-1}-1)\}$

4-bit " :- {-7 to +7}

(15)

Note :-

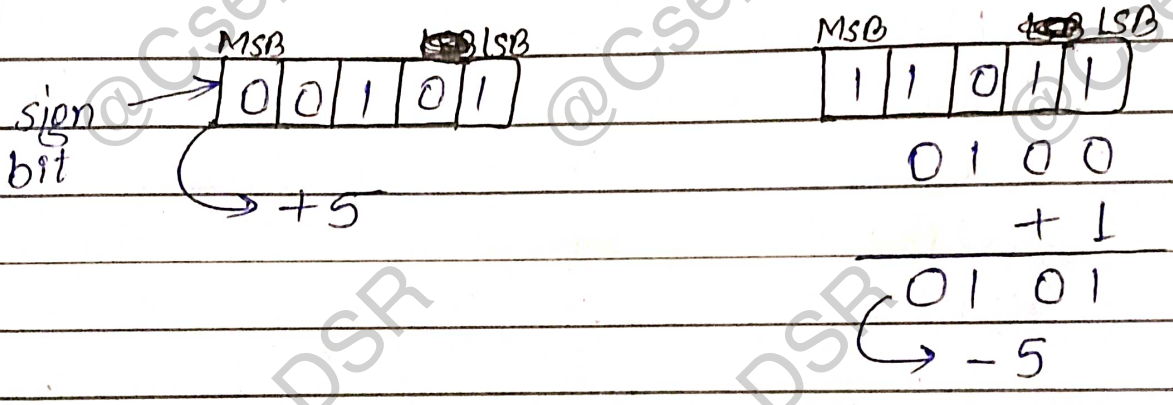
MSB is always 1 in case of -ve number.

DATE : __/__/____
PAGE : _____

* 2's complement :-

↳ If number is +ve then sign bit is 0 and remaining bits are magnitude of the number same as signed magnitude format.

↳ If number is -ve then sign bit is 1 and 2's complement of remaining bits represents magnitude.



Ex:- 0 1 0 1 0 1 0 → 1's complement

↳ -ve 1 0 1 0 1 0 1

+ 1

-ve 1 0 1 0 1 1 0

↳ -86

n-bit Range :- $\{-(2^{n-1}) \text{ to } +(2^{n-1}-1)\}$

4-bit " :- {-8 to +7}

(16)

DATE : __/__/____
PAGE : _____

4-bit binary	Unsigned	Signed	1's	2's
0000	0	+0	+0	+0
0001	1	+1	+1	+1
0010	2	+2	+2	+2
0011	3	+3	+3	+3
0100	4	+4	+4	+4
0101	5	+5	+5	+5
0110	6	+6	+6	+6
0111	7	+7	+7	+7
1000	8	-0	-7	-8
1001	9	-1	-6	-7
1010	10	-2	-5	-6
1011	11	-3	-4	-5
1100	12	-4	-3	-4
1101	13	-5	-2	-3
1110	14	-6	-1	-2
1111	15	-7	-0	-1

* Floating Point Representation :-

↳ This representation is used for very large and very small data.

↳ Very large no. can't be represented using fixed point represented nor can very small fractions.

↳ To represent very large and very small

DATE : __/__/__
PAGE : _____

fraction of data within less storage, floating point representation is use.

↳ This representation is different from fixed point representation.

Ex:- i) 0.00000000891
ii) 3876452897007.001

↳ The description of binary numbers in exponential format is called floating point representation.

↳ It has three part :-
(a) Mantissa
(b) Base
(c) Exponent

↳ General format (or) scientific notation :-

$$\pm M * B^{\pm E}$$

Here, $\pm$ → sign, M → Mantissa
B → Base, E → Exponent

Ex:-

Number	Mantissa	Base	Exponent
-9×10^{8}	-9	10	8
110×2^{7}	110	2	7
4364.784	4364784	10	-3

↳ 4364784×10^{-3}

DATE : / /
PAGE :

* IEEE Floating Point Representation

↳ The IEEE standard (IEEE-754) is a technical standard for floating point representation which was established in 1985 by I.E.E.E (Institute of Electrical and Electronics Engineering).

↳ IEEE-754 has three basic components or fields.

ⓐ sign
ⓑ Baised exponent
ⓒ Normalised Mantissa

ⓐ Sign :- In sign field, if it is 0 then it represent +ve number and if it is 1 then it represent -ve.

ⓑ Baised exponent :- It represents both +ve and -ve exponents.

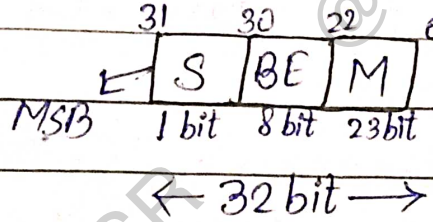
ⓒ Normalised mantissa :- The mantissa part is a part of number in scientific notation or floating point number consisting its significant digit / bit.

↳ According to IEEE standard -754

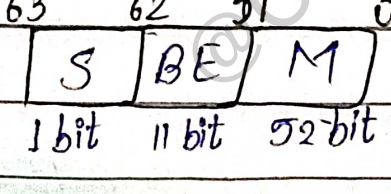
DATE : __/__/__
PAGE : _____

the floating point number is represented in two ways.
(i) single precision
(ii) double precision

(i) Single precision :-
In this 32 bits are used to represent floating point number.
↳ Out of 32 bit, 1 bit is used for sign bit which is MSB, 8 bits are used for exponent and 23 bits are for mantissa.



(ii) Double precision :-
In this 64 bits are used to represent floating point number.
↳ Out of 64 bits, 1 bit is used for sign bit or MSB, 11 bits are used for exponents and 52 bits are for mantissa.



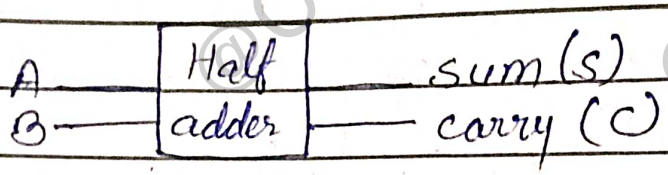
DATE : __/__/__
PAGE : _____

* Arithmatic Circuit :-

1) Adders

2) Half adder :- The half adder is a combinational circuit and an arithmatic circuit used to perform addition of two single bits.

Block diagram :-



Let, A and B are two input variables and sum (S) and carry (C) are two output variables.

I/P		O/P	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Simplified boolean expression for sum and carry.

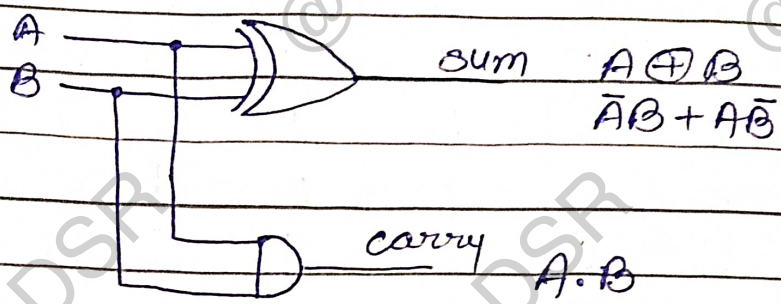
$$Sum = \bar{A}B + A\bar{B}$$

$$\boxed{Sum = A \oplus B}$$

DATE : __/__/____
PAGE : _____

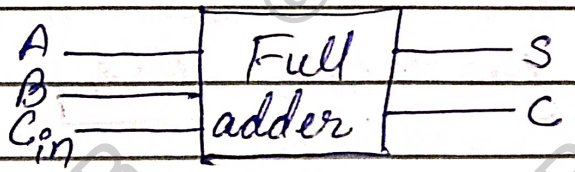
carry = AB

Logic diagram for half adder :-



b) Full adder :- The full adder is also an arithmetic and combinational circuit, that performs the sum of three input bits.

Block diagram :-



DATE : __/__/__
PAGE : _____

Truth Table

I/P			O/P	
A	B	C_{in}	S	C
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

Boolean expression :—

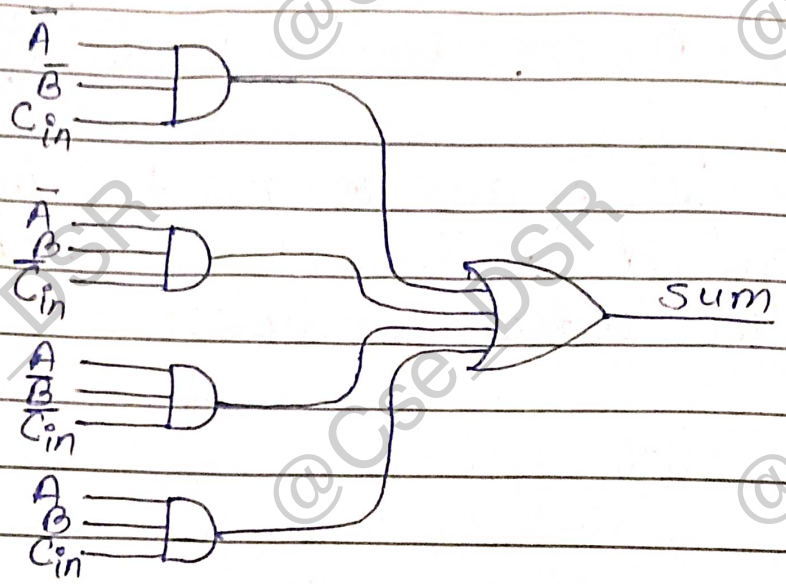
$$sum = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$$

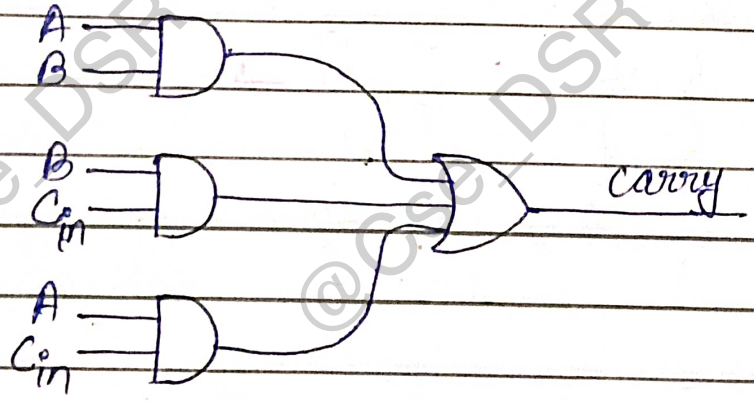
$$\boxed{sum = A \oplus B \oplus C_{in}}$$

$$\boxed{carry = AB + BC_{in} + AC_{in}}$$

DATE : __/__/____
PAGE : _____

Logic diagram for full adder :-



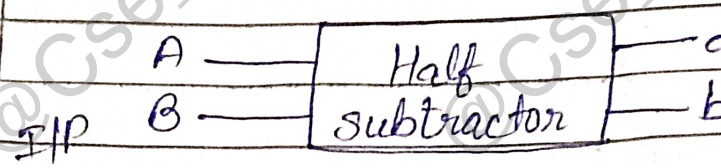


ii) Subtractor

a) Half subtractor :- The half subtractor is an arithmatic and a combinational circuit that subtract one-bit from the other bit and produces differences (d) & borrow (b).

DATE : __/__/____
PAGE : _____

Block diagram :-



Let, A and B are two input variables and d (difference) and b (borrow) are two output variables.

Truth Table

I/P		O/P	
A	B	d	b
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Simplified boolean expression :-

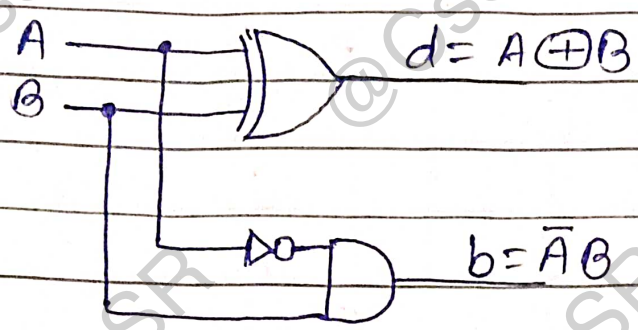
$$d = \bar{A}B + A\bar{B}$$

$$\boxed{d = A \oplus B}$$

$$\boxed{b = \bar{A} \cdot B}$$

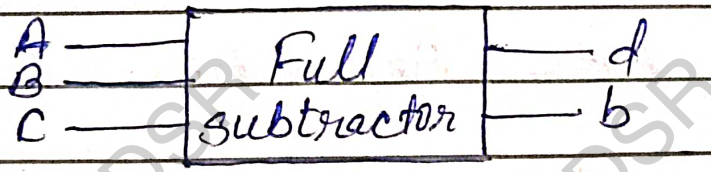
DATE : __/__/____
PAGE : _____

Logic diagram for half subtractor :—



b) **Full subtractor :—** The full subtractor is an arithmetic and a logic circuit, which performs subtraction of three 1-bit numbers. As half subtractor is used only for two numbers, to overcome this issue full subtractor is used.

Block diagram :—



Here, A, B and C are three input variables and d (diff.) and b (borrow) are two output variables.

DATE : __/__/____
PAGE : _____

Truth Table

I/P			O/P	
A	B	C	d	b
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Simplified boolean expression :-

$$d = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$\boxed{d = A \oplus B \oplus C}$$

$$b = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + ABC$$

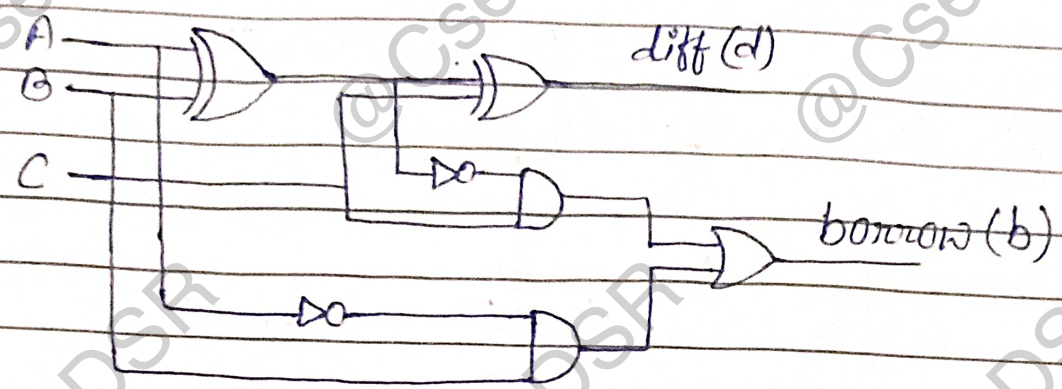
$$b = C(\bar{A}\bar{B} + AB) + \bar{A}B(\bar{C} \overset{1}{\neq} C)$$

$$b = C(A \odot B) + \bar{A}B$$

$$\boxed{b = \bar{A}B + \bar{A}C + BC}$$

DATE : __/__/__
PAGE :

Logic diagram for full subtraction :-



* Algorithm of addition and subtraction with sign magnitude.

We have two methods :-

i) 1's complement method
ii) 2's complement method

i) 1's complement method :-

Algorithm :-

Step 1: Convert number to be subtracted to its 1's complement form.
Step 2: Perform addition.
Step 3: If the final carry is 1, then add it to the result obtained in step 2. If final carry is 0, result obtained in step 2 is -ve and in the 1's complement format.

DATE : __/__/____
PAGE : _____

$A - B$
$= A + (-B)$
↑ 1's complement

Ex 1:→ $(1100)_2 - (0101)_2$ {12 − 5}

```
  1100
+ 1010   <- 1's complement
------
  1011 0
    -> +1
------
  0111  = +7
```

Ex 2:→ 5 − 12 = (−7) ?

5 = 0101
12 = 1100
1's complement
−12 = 0011

```
  0101
+ 0011
------
 [0]1000
```

1's complement

−ve 0111 = −7 //

DATE: __/__/____
PAGE: _____

Q. $(1011)_2 - (0100)_2$

1's complement

```
  1011
+ 1011
-------
 10110
   + 1
-------
  0111 = 7
```

Q. $(0110)_2 - (1011)_2$

1's complement

```
  0110
+ 0100
-------
 01010
```

1's complement

-ve 0101 = −5

H.W

Q. $(11001)_2 - (11110)_2$

1's comp.

```
  11001
+ 00001
-------
 011010
```

1's complement

-ve 00101 = −5

Q. $(10000)_2 - (11101)_2$

1's comp.

```
  10000
+ 00010
-------
 100010
```

-ve [illegible] = −

01101 = −13

DATE: __/__/__
PAGE: _____

ii) 2's complement :-

Algorithm :-

Step 1 : Find 2's complement of the number to be subtracted.

Step 2 : Perform addition.

Step 3 : If final carry is 1, then the result is +ve and in the true format or original format.

If final carry is 0, then the result is -ve and in 2's complement form.

Ex. 1 :- $(1001)_2 - (0100)_2$ → 2's complement

1001 = 9

1100 = (-4)

10101 = 5

discard

1011 = (+4)

+1

1100 = (-4)

Ex. 2 :- $(0110)_2 - (1011)_2$ → 2's complement

0110

0101

01011

0100 → 1's complement

+1

-ve 0101 = -5

0100

+1

0101

DATE : __/__/____
PAGE : _____

Note :- The final carry bit acts as a sign bit for the answer. If final carry is 1, then answer is +ve and if final carry is 0, then the answer is -ve.

H.W

(i) $(0110)_2 - (0100)_2$

2's complement of 0100 → 1011 + 1 = 1100

```
  0110
+ 1100
-------
 [1]0010 = 2
```

discard

(ii) $(0111)_2 - (1110)_2$

2's complement of 1110 → 0001 + 1 = 0010

```
  0111
+ 0010
-------
 [0]1001
    |
    v
   0110
   + 1
-------
-ve 0111 = -7
```

(iii) $(10110)_2 - (1111)_2$

2's complement of 01111 → 10000 + 1 = 10001

```
  10110
+ 10001
---------
[1]00111 = +7
```

discard

PAGE :

Multiplication of binary numbers :—

A × B

↑ ↑

Multiplicand Multiplier

Algorithm :—

Step 1 : Generation of partial product.
Step 2 : Addition of partial product.

Ex :— 22 × 15 = 330

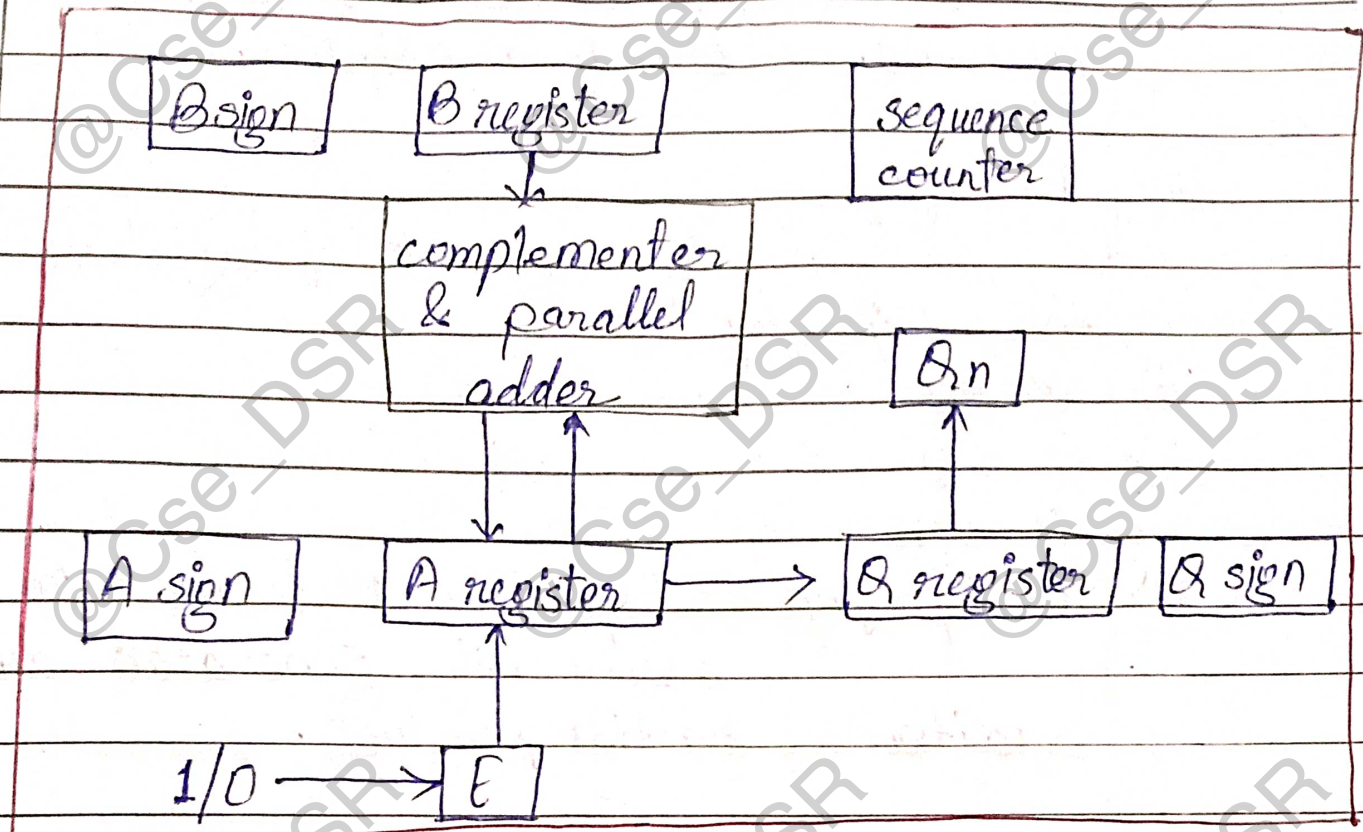
22 ⟶ 10110
15 ⟶ 01111

```
  10110
x 01111
-------
  10110
 101 10x
1011 0xx
10110xxx
00000xxxx
---------
0101001010  = 330
```

(256) 128 (64) 32 (16) (8) 4 (2) 1

DATE : __/__/____
PAGE : _____

Hardware implementation for multiplication :-



H/W implementation for multiplication

i) Register :-

a) B → B is used to store multiplicand.

b) Q → Q is used to store multiplier.

c) A → A is used to store partial product during multiplication.

ii) Sequence Counter :-
Sequence counter is used to store no. of bits in multiplier.

iii) Flip-flop :-
To store sign bit of registers require three flip-flop :-
a) B sign
b) Q sign
c) A sign
d) E flip-flop :- Flip-flop E is used to store carry bit, generated during addition of partial product.

iv) Complementer & parallel adder :-
This H/W is used in the calculation of partial products and addition of obtained partial products.

* Booth's Multiplication Algorithm :-

→ Booth's multiplication is invented by "Andrew Donald Booth" in 1950.
→ It is a multiplication algorithm that allows to multiply the two sign binary number in 2's complement.
→ It is also used to speed up the performance

DATE : __/__/__
PAGE : _____

of multiplication process.
In Booth's multiplication less no. of addition/subtraction required, so it is very efficient way.

* Advantages of Booth's multiplication :-

a) It reduces the no. of partial products.
b) Easy calculation of multiplication.
c) Consecutive additions will be replaced.
d) Less complex.

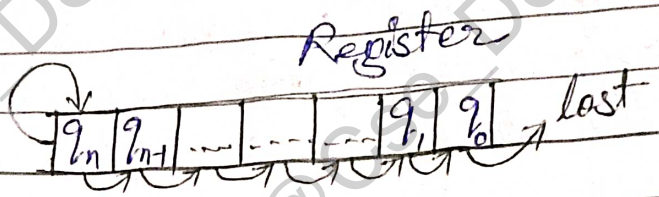
* Disadvantages of Booth's multiplication :-

a) This algorithm will not work for 1's complement.
b) It might be time consuming process.

Note :- Booth's multiplication algorithm uses the concept of ASR (Arithmetic Shift Right)

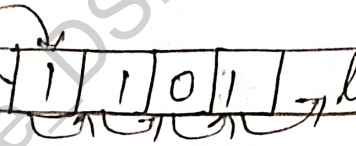
* ASR (Arithmetic Shift Right)

→ In this left most bit is not only shifted by one bit, but it also remains in the same/original position and the remaining bits are normally shifted to right by one bit and the right most bit will be lost.



After ASR :—

Ex:—



After ASR :— 1 1 1 0

* Algorithm for Booth's multiplication :—

Step 1: Multiplier and multiplicand are placed in Q & M registers respectively.
Step 2: Result for this will be stored in A and Q register.
Step 3: Initially A and Q_{-1} registers will be 0.
Step 4: Multiplication of numbers is done in cycle.
Step 5: 1-bit register Q_{-1} is placed right of the LSB bit (Q_0) of register Q.
Step 6: In each cycle, Q_0 and Q_{-1} bits will be checked.
(i) If Q_0 and Q_{-1} are 11 or 00

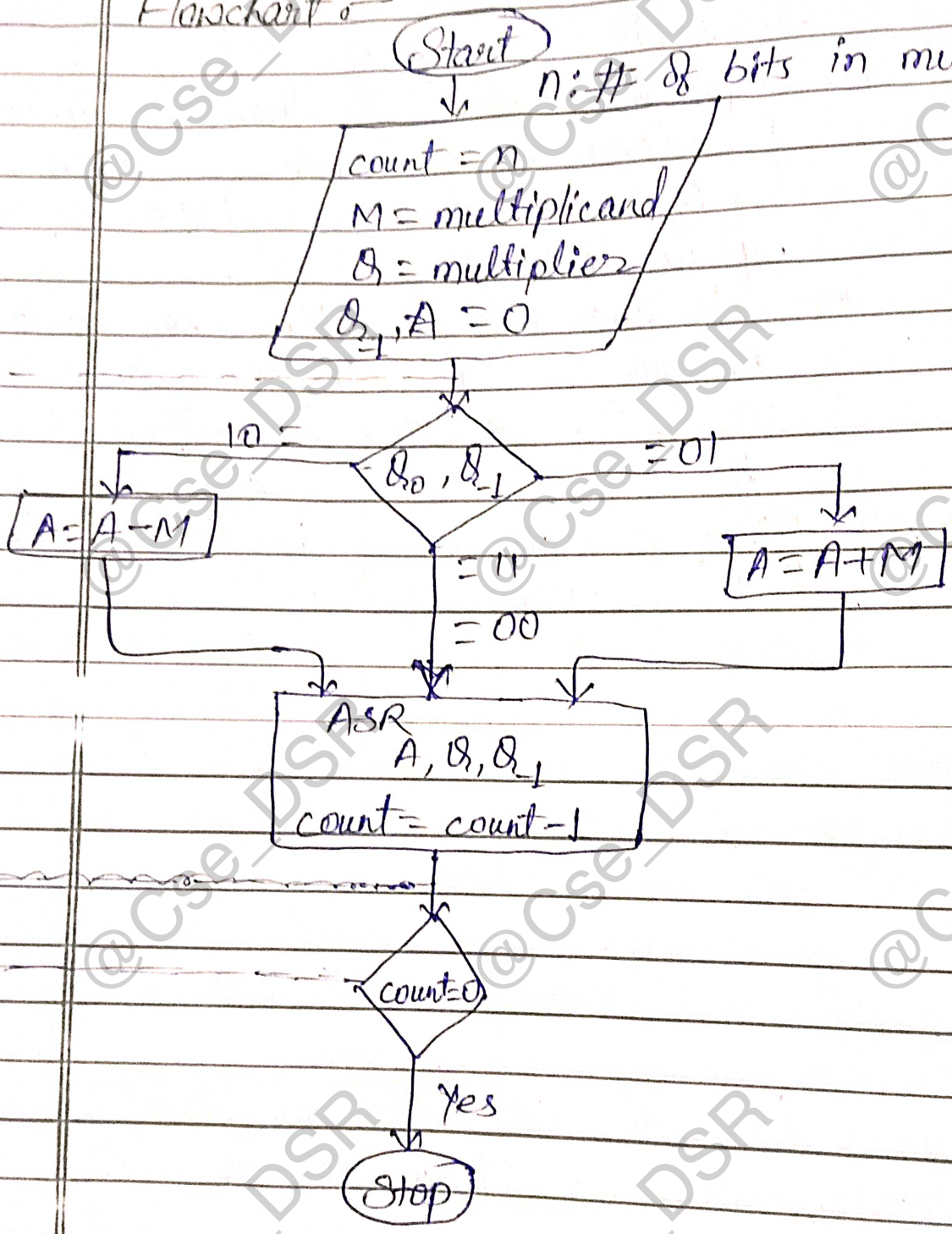
then the bits of A, Q, Q_{-1} are shifted by 1-bit using ASR.

(ii) If the value '01' then multiplicand is added to A. After addition, A, Q, Q_{-1} registers are shifted to right by one-bit using ASR.

(iii) If the value is '10', then multiplicand is subtracted from A. After that A, Q, Q_{-1} register is shifted to right by one-bit using ASR.

Step 7: Final product will be taken in "AQ" register pair.

Flowchart :-



Final product is in AQ register pair.

DATE : __/__/____
PAGE : _____

Ex:- $(-7) * (-3) = +21$

$+7 = 0111$

$-7 = 1001$ (by 2's complement)

$+3 = 0011$

$-3 = 1101$ (by 2's complement)

$M = 1001$, $Q = 1101$ ($Q_3 Q_2 Q_1 Q_0$), $A = 0000$, $Q_{-1} = 0$

A	Q	Q_{-1}		
				count = 4
0000	1101	0		
0111	1101	0	A = A - M	
0011	1110	1	ASR	count = 3
1100	1110	1	A = A + M	
1110	0111	0	ASR	count = 2
0101	0111	0	A = A - M	
0010	1011	1	ASR	count = 1
0001	0101	1		count = 0

A-Q pair

$+21$