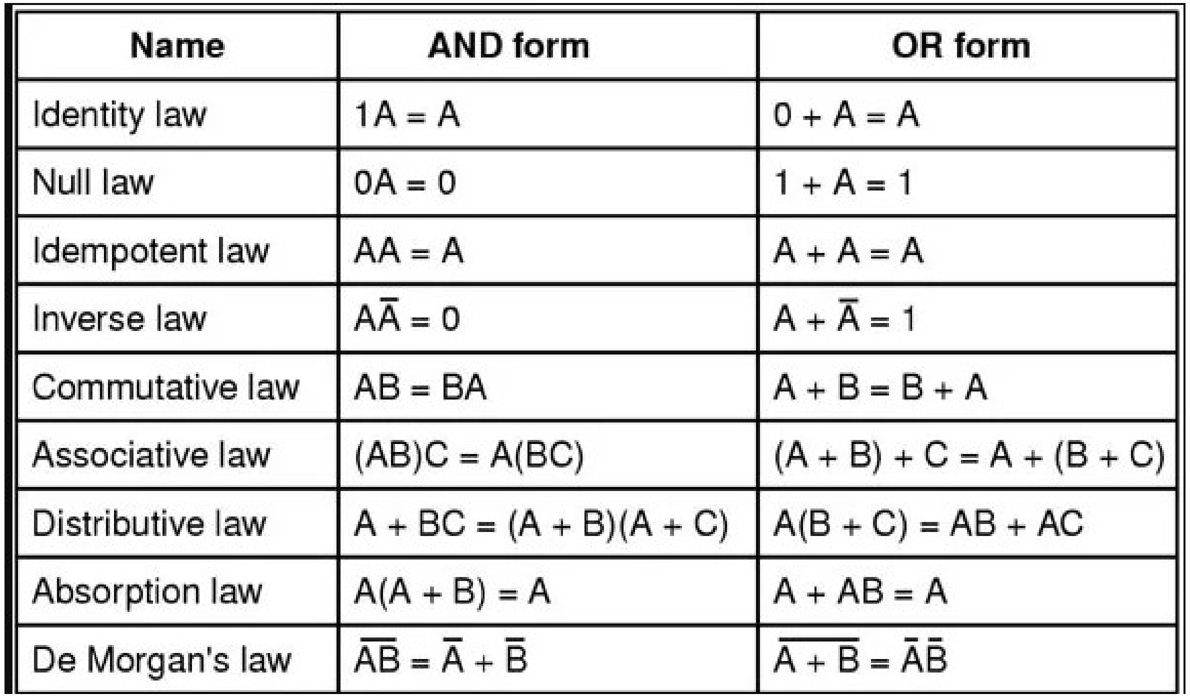
Minimization

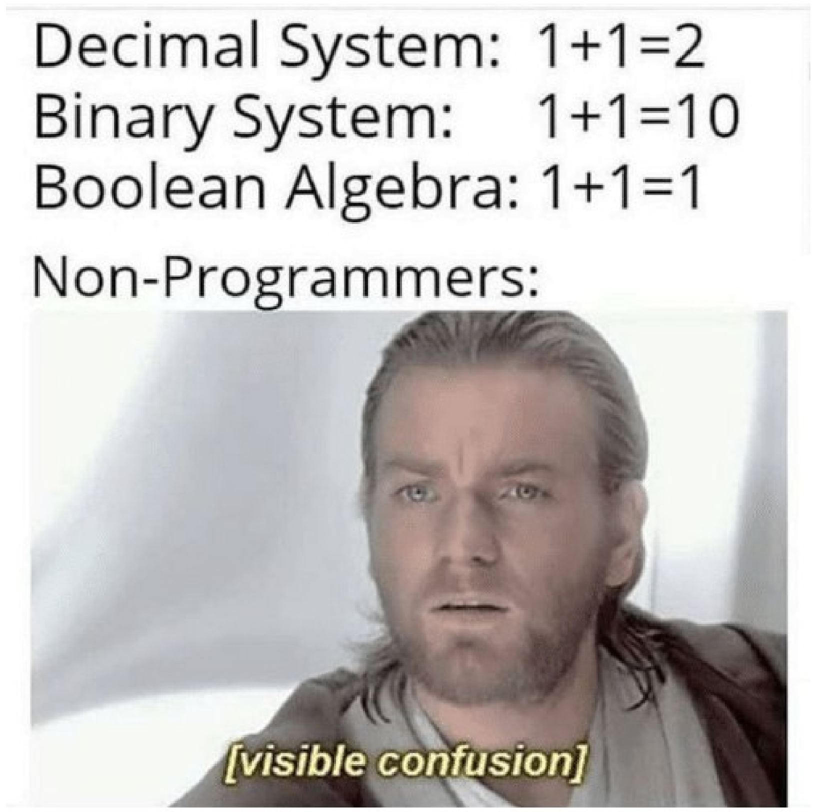
2 Methods

- Algebraic
- K - Map

Algebraic



Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\bar{A} = 0$	$A + \bar{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$



The first Boolean algebra must've been like:

Example

F = ABC’D’ + ABC’D + AB’C’D + ABCD + AB’CD + ABCD’ + AB’CD’

- Since this method is covered last class lets quickly gloss over the solution.

F = ABC’ (D’ + D) + AB’C’D + ACD (B + B’) + ACD’ (B + B’)

=> ABC’ + AB’C’D + ACD + ACD’ -- [A + A’ = 1]

=> ABC’ + AB’C’D + AC (D + D’)

=> ABC’ + AB’C’D + AC -- [A + A’ = 1]

=> A (BC’ + C) + AB’C’D

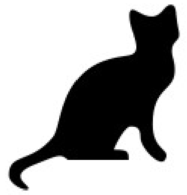
=> A (B +C) + AB’C’D -- [A + A’B = A + B]

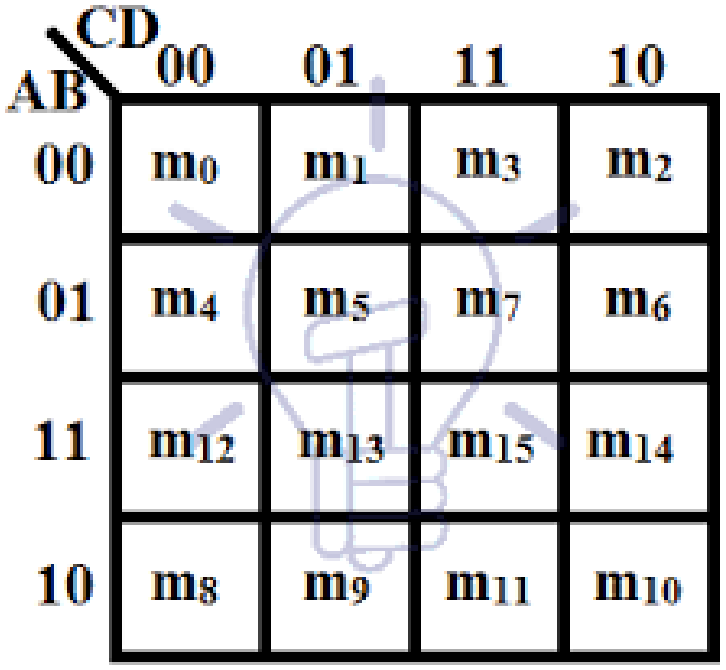
=> AB + AC + AB’C’D

=> AB + AC + AC’D -- [A + A’B = A + B]

=> AB + AC + AD -- [A + A’B = A + B]

K - Map

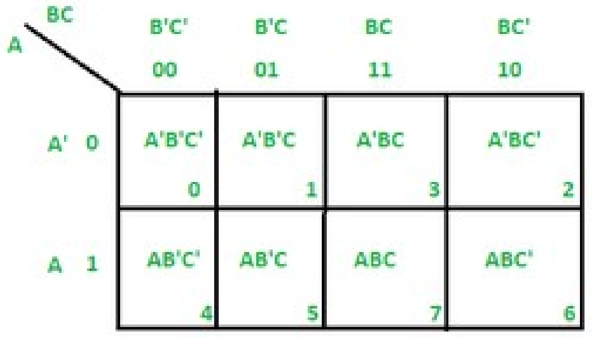
&&	!alive	alive
!dead		
dead		



AB \ CD	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

- Sum of Product (SOP)
- Product of Sum (POS)

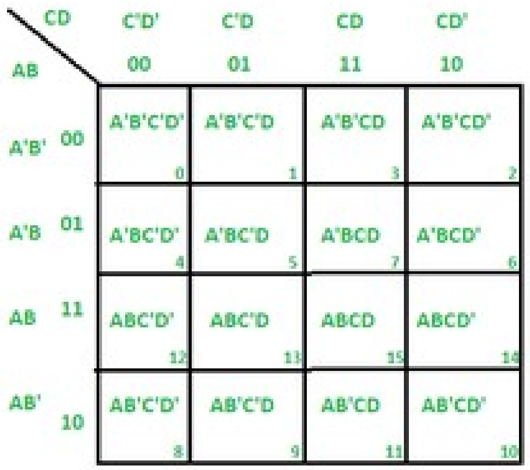
K-Map (SOP)



A \ BC	B'C' 00	B'C 01	BC 11	BC' 10
A' 0	A'B'C' 0	A'B'C 1	A'BC 3	A'BC' 2
A 1	AB'C' 4	AB'C 5	ABC 7	ABC' 6

SOP(MINTERMS)

8 Blocks = 1
4 Blocks = 1 variable term
2 Blocks = 2 variable term
1 Block = 3 variable term



AB \ CD	C'D' 00	C'D 01	CD 11	CD' 10
A'B' 00	A'B'C'D' 0	A'B'C'D 1	A'B'CD 3	A'B'CD' 2
A'B 01	A'BC'D' 4	A'BC'D 5	A'BCD 7	A'BCD' 6
AB 11	ABC'D' 12	ABC'D 13	ABCD 15	ABCD' 14
AB' 10	AB'C'D' 8	AB'C'D 9	AB'CD 11	AB'CD' 10

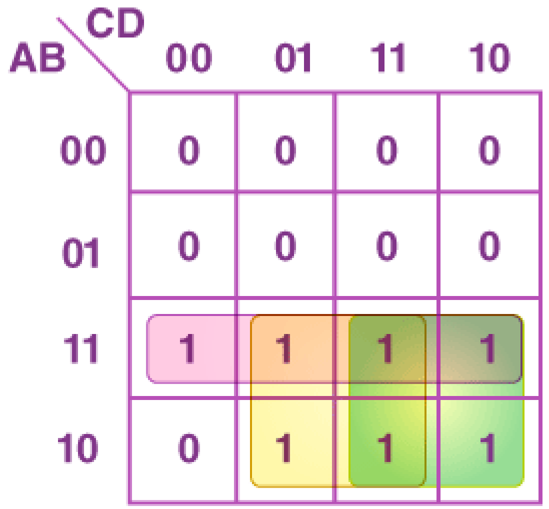
SOP(MINTERMS)

16 Blocks = 1
8 Blocks = 1 variable term
4 Blocks = 2 variable term
2 Blocks = 3 variable term
1 Block = 4 variable term

K-Map (SOP)

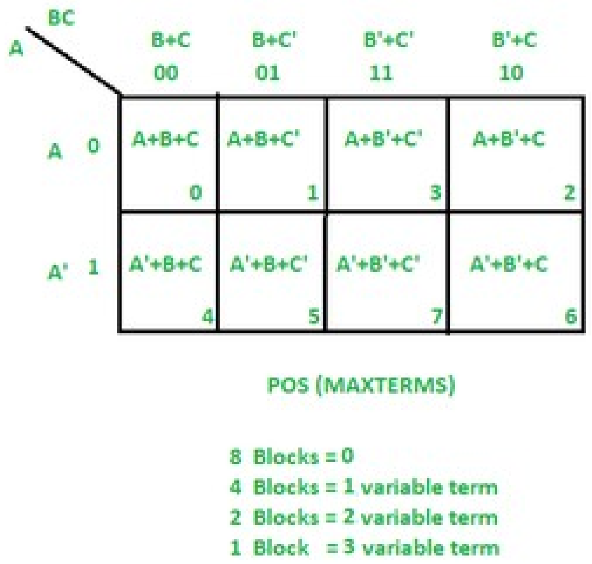
- F = ABC'D' + ABC'D + AB'C'D + ABCD + AB'CD + ABCD' + AB'CD'

- F = ABC’D’ + ABC’D + AB’C’D + ABCD + AB’CD + ABCD’ + AB’CD’



AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	1	1	1

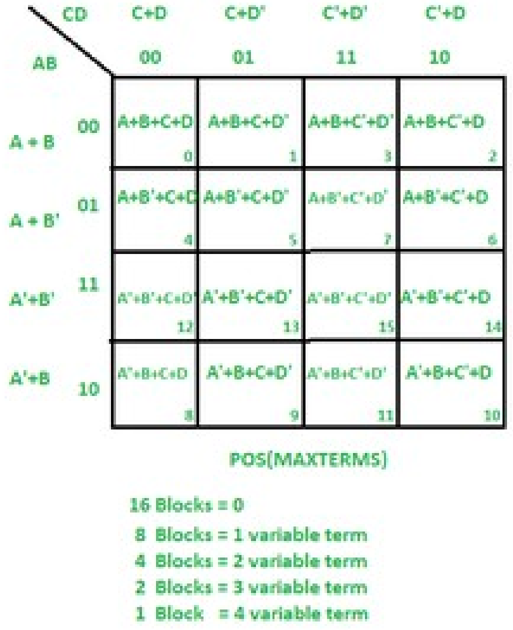
K – Map (POS)



A \ BC	B+C 00	B+C' 01	B'+C' 11	B'+C 10
A 0	A+B+C 0	A+B+C' 1	A+B'+C' 3	A+B'+C 2
A' 1	A'+B+C 4	A'+B+C' 5	A'+B'+C' 7	A'+B'+C 6

POS (MAXTERMS)

8 Blocks = 0
4 Blocks = 1 variable term
2 Blocks = 2 variable term
1 Block = 3 variable term



AB \ CD	C+D 00	C+D' 01	C'+D' 11	C'+D 10
A + B 00	A+B+C+D 0	A+B+C+D' 1	A+B+C'+D' 3	A+B+C'+D 2
A + B' 01	A+B'+C+D 4	A+B'+C+D' 5	A+B'+C'+D' 7	A+B'+C'+D 6
A'+B' 11	A'+B'+C+D 12	A'+B'+C+D' 13	A'+B'+C'+D' 15	A'+B'+C'+D 14
A'+B 10	A'+B+C+D 8	A'+B+C+D' 9	A'+B+C'+D' 11	A'+B+C'+D 10

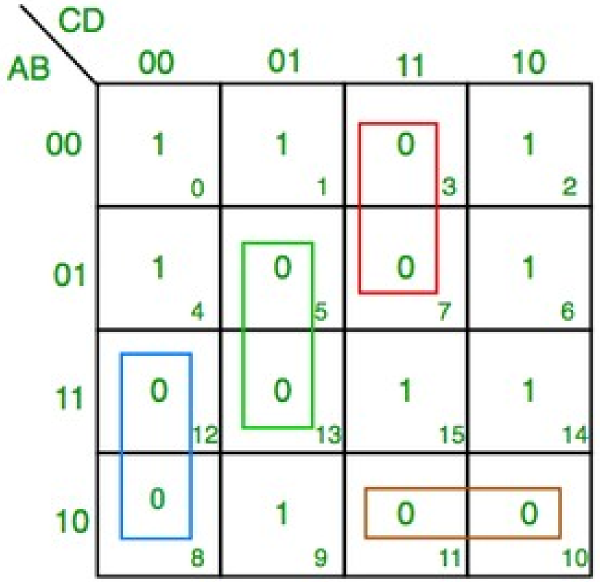
POS(MAXTERMS)

16 Blocks = 0
8 Blocks = 1 variable term
4 Blocks = 2 variable term
2 Blocks = 3 variable term
1 Block = 4 variable term

K Map POS

- $F(A,B,C,D)=\pi(3,5,7,8,10,11,12,13)$

- F(A,B,C,D)=π(3,5,7,8,10,11,12,13)



Lets Solve a couple of problems together

- $P'(P + QR) + (PC + Q'R)$.
- $A' (M' + A') (M + M'A)$
- $FT' + F' + T'F'$
- $F = \sum(0, 1, 5, 7, 15, 14, 10)$
- $F(J, K, L) = (J + KL') (JK' +L)$

Answer: https://byjus.com/gate/minimization
-of-boolean-functions-notes/

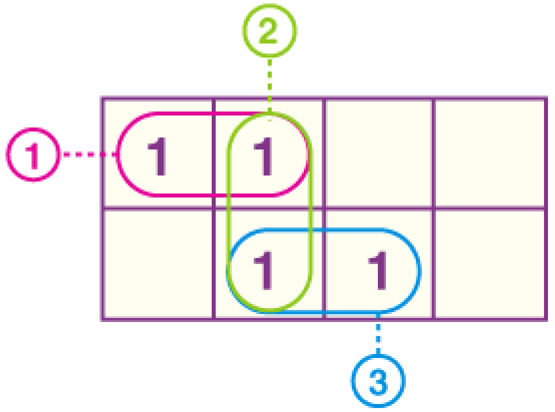
Implicants

- Individual Terms of a boolean function

Prime Implicants

- A Group of adjecent minterms (SOP) / maxterms (POS)

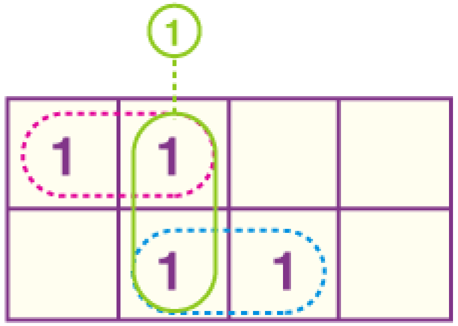
BYJU'S The Learning App



No. of Prime Implicants = 3

Redundant Prime Implicants

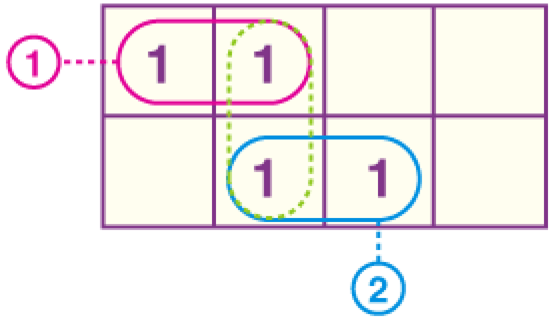
- Each minterm/maxterm is covered by other prime implicants – Will Never Appear in Simplified Function.



No. of Redundant Prime Implicants = 1

Essential Prime Implicants

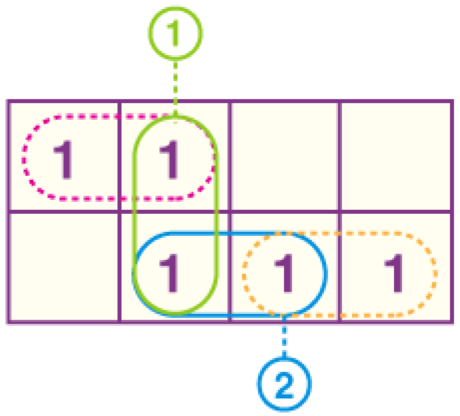
- Atleast one maxterm/minterm is not covered by any other prime implicants – Will Always Appear in Simplified Function



No. of Essential Prime Implicants = 2

Selective Prime Implicants

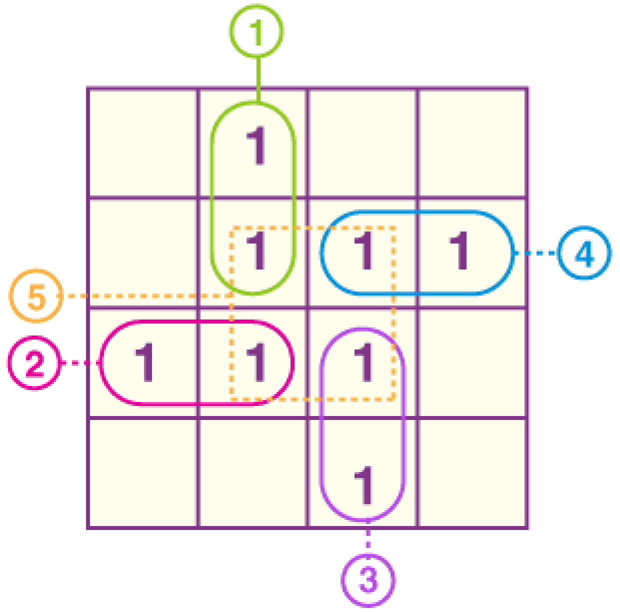
- Neither Essential nor Redundant – May or Maynot appear in the final simplified function



No. of Selective Prime Implicants = 2

Example

- Find the number of implicants, EPI, PI, RPI and SPI if $F = \sum(1, 5, 6, 7, 11, 12, 13, 15)$



$F = ① + ② + ③ + ④$

No. of Implicants = 8

PI = (1,2,3,4,5)

EPI = (1,2,3,4)

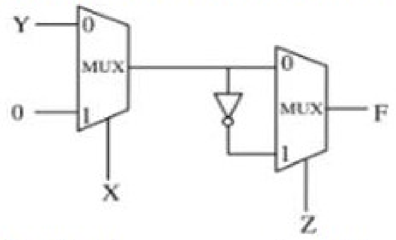
RPI = (5)

- Total number of Implicants = 8
- Total number of PI or Prime Implicants = 5
- Total number of EPI or Essential Prime Implicants = 4
- Total number of RPI or Redundant Prime Implicants = 1
- Total number of SPI or Selective Prime Implicants = 0

Gate Questions :)

Feel free to solve it in your free time

Consider the circuit shown in the figure.



The Boolean expression F implemented by the circuit is

A $\overline{XYZ} + XY + \overline{Y}Z$

B $\overline{X}Y\overline{Z} + XZ + \overline{Y}Z$

C $\overline{X}Y\overline{Z} + XY + \overline{Y}Z$

D $\overline{X}\,\overline{YZ} + XY + \overline{Y}Z$

Question 2

The Boolean expression $F(X,Y,Z) = \overline{X}Y\overline{Z} + X\overline{Y}\overline{Z} + XY\overline{Z} + XYZ$ converted into the canonical product of sum (POS) form is

A $(X+Y+Z)(X+Y+\overline{Z})(X+\overline{Y}+\overline{Z})(\overline{X}+Y+\overline{Z})$

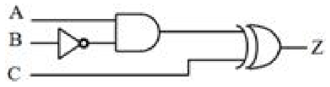
B $(X+\overline{Y}+Z)(\overline{X}+Y+\overline{Z})(\overline{X}+\overline{Y}+Z)(\overline{X}+\overline{Y}+\overline{Z})$

C $(X+Y+Z)(\overline{X}+Y+\overline{Z})(X+\overline{Y}+Z)(\overline{X}+\overline{Y}+\overline{Z})$

D $(X+\overline{Y}+\overline{Z})(\overline{X}+Y+Z)(\overline{X}+\overline{Y}+Z)(X+Y+Z)$

Question 3

All the logic gates shown in the figure have a propagation delay of 20 ns. Let A = C = 0 and B = 1 until time t = 0. At t = 0, all the inputs flip (i.e., A = C = 1 and B = 0) and remain in that state. For t > 0, output Z = 1 for a duration (in ns) of _________.



A Fill in the Blank Type Question

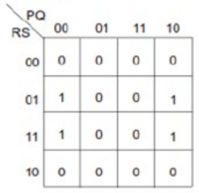
Question 4

A 3-input majority gate is defined by the logic function M(a, b,c) = ab + be + ca . Which one of the following gates is represented by the function $M(\overline{M(a,b,c)}, M(a,b,\bar{c}), c)$?

- A 3-input NAND gate
- B 3-input XOR gate
- C 3-input NOR gate
- D 3-input XNOR gate

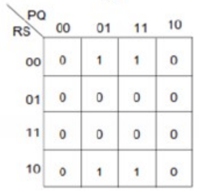
Question 5

Following is the K-map of a Boolean function of five variables P, Q, R, S and X. The minimum sum-of-product (SOP) expression for the function is



RS \ PQ	00	01	11	10
00	0	0	0	0
01	1	0	0	1
11	1	0	0	1
10	0	0	0	0

X=0



RS \ PQ	00	01	11	10
00	0	1	1	0
01	0	0	0	0
11	0	0	0	0
10	0	1	1	0

X=1

A $\bar{P}\bar{Q}S\bar{X}+P\bar{Q}S\bar{X}+Q\bar{R}\bar{S}X+QR\bar{S}X$

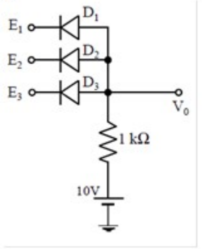
B $\bar{Q}S\bar{X}+Q\bar{S}X$

C $\bar{Q}SX+Q\bar{S}\bar{X}$

D $\bar{Q}S+Q\bar{S}$

Question 6

In the circuit shown diodes D_1, D_2 and D_3 are ideal, an the inputs E_1, E_2 and E_3 are “0 V” for logic ‘0’ and “10 V” for logic ‘1’. What logic gate does the circuit represent?



A 3-input OR gate

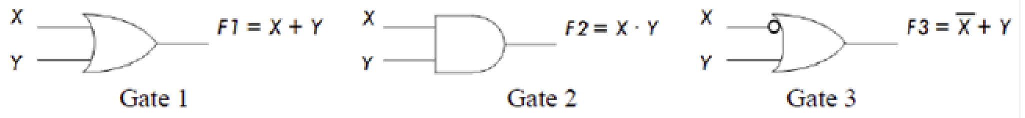
B 3-input NOR gate

C 3-input AND gate

D 3-input XOR gate

Question 7

A universal logic gate can implement nay Boolean function by connecting sufficient number of them appropriately. Three gates are shown.



Which one of the following statements is TRUE?

A Gate 1 is a universal gate

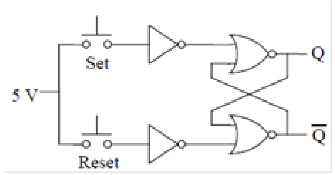
B Gate 2 is a universal gate

C Gate 3 is a universal gate

D None of the gates shown is a universal gate

Question 8

An SR latch is implemented using TTL gates as shown in the figure. The set and rest pulse inputs are provided using the push-button switches. It is observed that the circuit fails to work as desired. The SR latch can be made functional by changing



A NOR gates to NAND gates

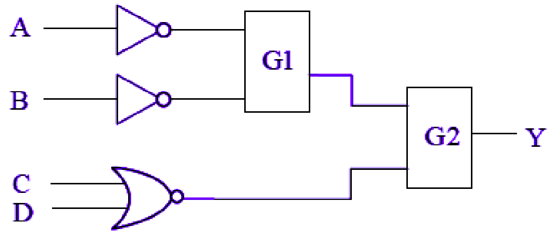
B Inverters to buffers

C NOR gates no NAND gates and inverters to buffers

D 5 V to ground

Question 9

In the figure shown, the output Y is required to be Y = AB + $\bar{C}\bar{D}$. The gates G1 and G2 must be, respectively,



A NOR, OR

B OR, NAND

C NAND, OR

D AND, NAND

A function of Boolean variables X, Y and Z is expressed in terms of the min-terms as
F(X, Y, Z) = Σ(1, 2, 5, 6, 7)
Which one of the product of sums given below is equal to the function F(X, Y, Z)?

A $(\bar{X}+\bar{Y}+\bar{Z}).(\bar{X}+Y+Z).(X+\bar{Y}+\bar{Z})$

B $(X+Y+Z).(X+\bar{Y}+\bar{Z}).(\bar{X}+Y+Z)$

C $(\bar{X}+\bar{Y}+Z).(X+Y+\bar{Z}.)(X+Y+\bar{Z}).(X+Y+Z)$

D $(X+Y+\bar{Z}).(\bar{X}+Y+Z).(\bar{X}+Y+\bar{Z}).(\bar{X}+\bar{Y}+Z).(\bar{X}+\bar{Y}+\bar{Z})$

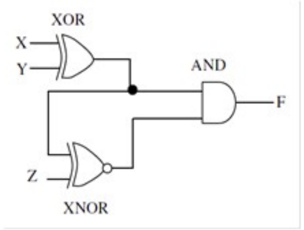
Question 11

The Boolean expression $(X+Y)(X+\overline{Y})+\overline{(X\overline{Y})+\overline{X}}$ simplifies to

- A X
- B Y
- C XY
- D X + Y

Question 12

The output F in the digital logic circuit shown in the figure is



A $F = \overline{X}YZ + X\overline{Y}Z$

B $F = \overline{X}Y\overline{Z} + X\overline{Y}Z$

C $F = \overline{X}\overline{Y}Z + X\overline{Y}Z$

D $F = \overline{XYZ} + XYZ$

Question 13

Consider the Boolean function, $F(w,x,y,z) = wy + xy + \overline{w}xyz + \overline{w}\,\overline{x}y + xz + \overline{x}\,\overline{y}\,\overline{z}$. which one of the following is the complete set of essential prime implicants?

(A) $w, y, xz, \overline{x}\,\overline{z}$ (B) w, y, xz (C) $y, \overline{x}\,\overline{y}\,\overline{z}$ (D) $y, xz, \overline{x}\,\overline{z}$

Which one of the following is the complete set of essential prime implicants ?

- A A
- B B
- C C
- D D

Question 14

For an n-variable Boolean function, the maximum number of prime implicants is

- A 2(n-1)
- B n/2
- C 2^n
- D $2^{(n-1)}$

Question 15

A bulb in a staircase has two switches, one switch being at the ground floor and the other one at the first floor. The bulb can be turned ON and also can be turned OFF by any one of the switches irrespective of the state of the other switch. The logic of switching of the bulb resembles

A an AND gate

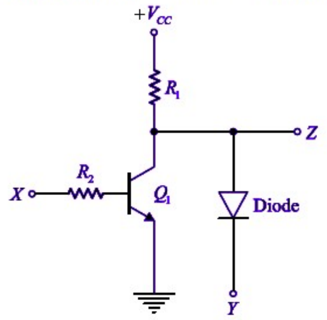
B an OR gate

C an XOR gate

D a NAND gate

Question 16

In the circuit shown below, Q_1 has negligible collector-to-emitter saturation voltage and the diode drops negligible voltage across it under forward bias. If V_{cc} is $+5\text{V}$, X and Y are digital signals with 0 V as logic 0 and V_{cc} as logic 1, the Boolean expansion for Z is



A XY

B $\overline{X}Y$

C $X\overline{Y}$

D $\overline{XY}$

Question 17

The output Y of a 2-bit comparator is logic 1 whenever the 2-bit input A is greater than the 2-bit input B. The number of combinations for which the output is logic 1, is

- A 4
- B 6
- C 8
- D 10

Question 18

In the sum of products function f(X, Y, Z) = ∑(2, 3, 4, 5), the prime implicants are

A $\overline{X}Y, X\overline{Y}$

B $\overline{X}Y, X\overline{Y}\overline{Z}, X\overline{Y}Z$

C $\overline{X}Y\overline{Z}, \overline{X}YZ, X\overline{Y}$

D $\overline{X}Y\overline{Z}, \overline{X}YZ, X\overline{Y}\overline{Z}, X\overline{Y}Z$

Question 19

Consider the following expression:

$AB CD + AB\overline{C}\ \overline{D} + ABC\overline{D} + AB\overline{C}D + ABCDE + AB\overline{C}\ \overline{D}\ \overline{E} + AB\overline{C}DE$

The simplification of this by using theorems of Boolean algebra will be

- **A** A + B
- **B** A ⊕ B
- **C** (A + B) (A . B)
- **D** A . B

Question 20

An electric power generating station supplies power to three loads A, B and C. Only a single generator is required when any one load is switched on. When more than one load is on, an auxiliary generator must be started. The Boolean equation for the control of switching of the auxiliary generator will be

A AA + BB + CC

B ABC + BCA + CAB

C AB + AC

D AB + AC + BC

Question 21

Which one of the following types of instructions will be used to copy from the source to the destination location?

A Arithmetic instruction

B Data transfer instructions

C Logical instructions

D Machine control instructions

Question 22

Simplify the Boolean expression F(w,x,y,z) = $\sum(0,1,2,4,5,6,8,9,12,13,14)$

A w+x+y+z

B y' + w'z'+xz'

C y +w'z'+xz

D x+z'w'y+x'

Integer Representation

Ritu Gain

106119106

Number System Conversion

Decimal to Other Base System

Steps

Step 1 − Divide the decimal number to be converted by the value of the new base.

Step 2 − Get the remainder from Step 1 as the rightmost digit (least significant digit) of new base number.

Step 3 − Divide the quotient of the previous divide by the new base.

Step 4 − Record the remainder from Step 3 as the next digit (to the left) of the new base number.

- epeat Steps 3 and 4, getting remainders from right to left, until the quotient becomes zero in Step 3.
- The last remainder thus obtained will be the Most Significant Digit (MSD) of the new base number

Example –

- Decimal Number: 29_{10}
- Calculating Binary Equivalent –
- As mentioned in Steps 2 and 4, the remainders have to be arranged in the reverse order so that the first remainder becomes the Least Significant Digit (LSD) and the last remainder becomes the Most Significant Digit (MSD).
- Decimal Number – 29_{10} = Binary Number – 11101_2.

Step	Operation	Result	Remainder
Step 1	29 / 2	14	1
Step 2	14 / 2	7	0
Step 3	7 / 2	3	1
Step 4	3 / 2	1	1
Step 5	1 / 2	0	1

Other Base System to Decimal System

Steps

- **Step 1** − Determine the column (positional) value of each digit (this depends on the position of the digit and the base of the number system).
- **Step 2** − Multiply the obtained column values (in Step 1) by the digits in the corresponding columns.
- **Step 3** − Sum the products calculated in Step 2. The total is the equivalent value in decimal

Example

- Binary Number – 11101_2
- Calculating Decimal Equivalent –
- Binary Number – 11101_2 = Decimal Number – 29_{10}

Step	Binary Number	Decimal Number
Step 1	11101_2	$((1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0))_{10}$
Step 2	11101_2	$(16 + 8 + 4 + 0 + 1)_{10}$
Step 3	11101_2	29_{10}

Other Base System to Non-Decimal System

- Steps
- **Step 1** − Convert the original number to a decimal number (base 10).
- **Step 2** − Convert the decimal number so obtained to the new base number.

Example

- Octal Number – 25_8
- Calculating Binary Equivalent –
- **Step 1 – Convert to Decimal**
- Octal Number – 25_8 = Decimal Number – 21_{10}

Step	Octal Number	Decimal Number
Step 1	25_8	$((2 \times 8^1) + (5 \times 8^0))_{10}$
Step 2	25_8	$(16 + 5)_{10}$
Step 3	25_8	21_{10}

Step 2 – Convert Decimal to Binary

- Decimal Number – 21_{10} = Binary Number – 10101_2
- Octal Number – 25_8 = Binary Number – 10101_2

Step	Operation	Result	Remainder
Step 1	21 / 2	10	1
Step 2	10 / 2	5	0
Step 3	5 / 2	2	1
Step 4	2 / 2	1	0
Step 5	1 / 2	0	1

Shortcut method - Binary to Octal

Steps

- **Step 1** − Divide the binary digits into groups of three (starting from the right).
- **Step 2** − Convert each group of three binary digits to one octal digit.

Example

- Binary Number – 10101_2
- Calculating Octal Equivalent –
- Binary Number – 10101_2 = Octal Number – 25_8

Step	Binary Number	Octal Number
Step 1	10101_2	010 101
Step 2	10101_2	2_8 5_8
Step 3	10101_2	25_8

Shortcut method - Octal to Binary

- Steps
- **Step 1** − Convert each octal digit to a 3 digit binary number (the octal digits may be treated as decimal for this conversion).
- **Step 2** − Combine all the resulting binary groups (of 3 digits each) into a single binary number.

Example

- Octal Number − 25_8
- Calculating Binary Equivalent −
- Octal Number − 25_8 = Binary Number − 10101_2

Step	Octal Number	Binary Number
Step 1	25_8	$2_{10}\ 5_{10}$
Step 2	25_8	$010_2\ 101_2$
Step 3	25_8	010101_2

Shortcut method - Binary to Hexadecimal

Steps

- **Step 1** – Divide the binary digits into groups of four (starting from the right).
- **Step 2** – Convert each group of four binary digits to one hexadecimal symbol.

Example

- Binary Number – 10101_2
- Calculating hexadecimal Equivalent –
- Binary Number – 10101_2 = Hexadecimal Number – 15_{16}

Step	Binary Number	Hexadecimal Number
Step 1	10101_2	0001 0101
Step 2	10101_2	1_{10} 5_{10}
Step 3	10101_2	15_{16}

Shortcut method - Hexadecimal to Binary

Steps

- **Step 1** − Convert each hexadecimal digit to a 4 digit binary number (the hexadecimal digits may be treated as decimal for this conversion).
- **Step 2** − Combine all the resulting binary groups (of 4 digits each) into a single binary number.

Example

- Hexadecimal Number – 15_{16}
- Calculating Binary Equivalent –
- Hexadecimal Number – 15_{16} = Binary Number – 10101_2

Step	Hexadecimal Number	Binary Number
Step 1	15_{16}	1_{10} 5_{10}
Step 2	15_{16}	0001_2 0101_2
Step 3	15_{16}	00010101_2

.

Unsigned Integers: Can represent zero and positive integers

Signed Integers: Can represent zero, positive and negative integers. Three representation schemes had been proposed for signed integers:

1. Sign-Magnitude representation
2. 1's Complement representation
3. 2's Complement representation

n-bit Unsigned Integers

Example 1: Suppose that n=8 and the binary pattern is 0100 0001B,

The value of this unsigned integer is 1×2^0 + 1×2^6 = 65D.

Example 2: Suppose that n=16 and the binary pattern is 0001 0000 0000 1000B,

The value of this unsigned integer is 1×2^3 + 1×2^12 = 4104D.

Example 3: Suppose that n=16 and the binary pattern is 0000 0000 0000 0000B,

The value of this unsigned integer is 0.

An **n-bit** pattern can represent 2^n distinct integers

An n-bit unsigned integer can represent integers from 0 to (2^n)-1, as tabulated below:

n	Minimum	Maximum
8	0	(2^8)-1 (=255)
16	0	(2^16)-1 (=65,535)
32	0	(2^32)-1 (=4,294,967,295) (9+ digits)
64	0	(2^64)-1 (=18,446,744,073,709,551,615) (19+ digits)

n-bit Sign Integers In Sign-magnitude Representation

In sign-magnitude representation:

- The most-significant bit (msb) is the sign bit, with value of 0 representing positive integer and 1 representing negative integer.
- The remaining n-1 bits represents the magnitude (absolute value) of the integer. The absolute value of the integer is interpreted as "the magnitude of the (n-1)-bit binary pattern".

Example 1: Suppose that n=8 and the binary representation is 0 100 0001B.

Sign bit is 0 ⇒ positive

Absolute value is 100 0001B = 65D

Hence, the integer is +65D

Example 2: Suppose that n=8 and the binary representation is 1 000 0001B.

Sign bit is 1 ⇒ negative

Absolute value is 000 0001B = 1D

Hence, the integer is -1D

Example 3: Suppose that n=8 and the binary representation is 0 000 0000B.

Sign bit is 0 ⇒ positive

Absolute value is 000 0000B = 0D

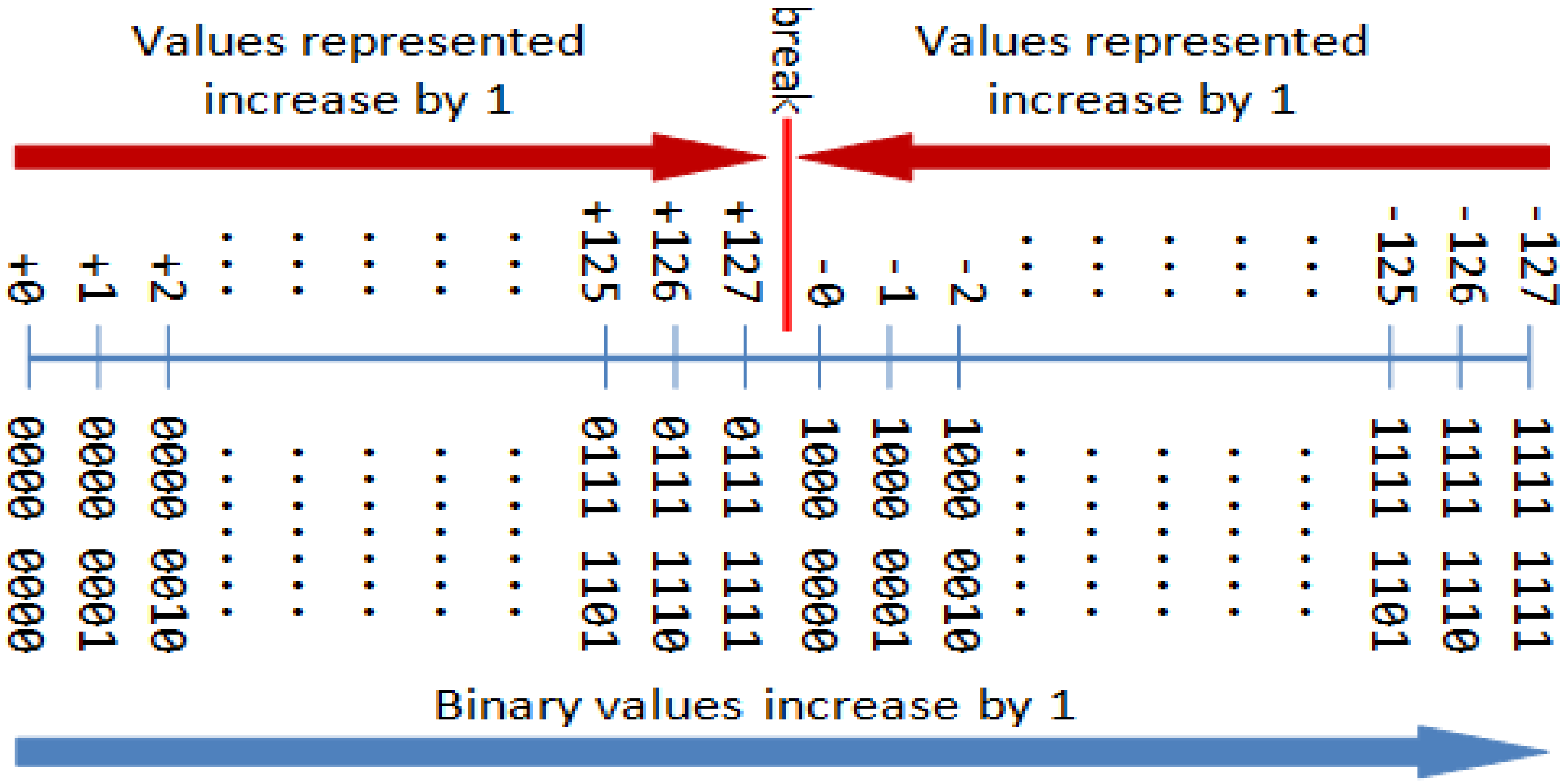
Hence, the integer is +0D

Example 4: Suppose that n=8 and the binary representation is 1 000 0000B.

Sign bit is 1 ⇒ negative

Absolute value is 000 0000B = 0D

Hence, the integer is -0D



Sign-Magnitude Representation

Drawbacks Of Sign-magnitude

- There are two representations (0000 0000B and 1000 0000B) for the number zero, which could lead to inefficiency and confusion.
- Positive and negative integers need to be processed separately.

n-bit Sign Integers in 1's Complement Representation

- In 1's complement representation:
- Again, the most significant bit (msb) is the *sign bit*, with value of 0 representing positive integers and 1 representing negative integers.
- The remaining *n*-1 bits represents the magnitude of the integer, as follows:
  - for positive integers, the absolute value of the integer is equal to "the magnitude of the (*n*-1)-bit binary pattern".
  - for negative integers, the absolute value of the integer is equal to "the magnitude of the *complement* (*inverse*) of the (*n*-1)-bit binary pattern" (hence called 1's complement).

Example 1: Suppose that n=8 and the binary representation 0 100 0001B.

Sign bit is 0 ⇒ positive

Absolute value is 100 0001B = 65D

Hence, the integer is +65D

Example 2: Suppose that n=8 and the binary representation 1 000 0001B.

Sign bit is 1 ⇒ negative

Absolute value is the complement of 000 0001B, i.e., 111 1110B = 126D

Hence, the integer is -126D

Example 3: Suppose that n=8 and the binary representation 0 000 0000B.

Sign bit is 0 ⇒ positive

Absolute value is 000 0000B = 0D

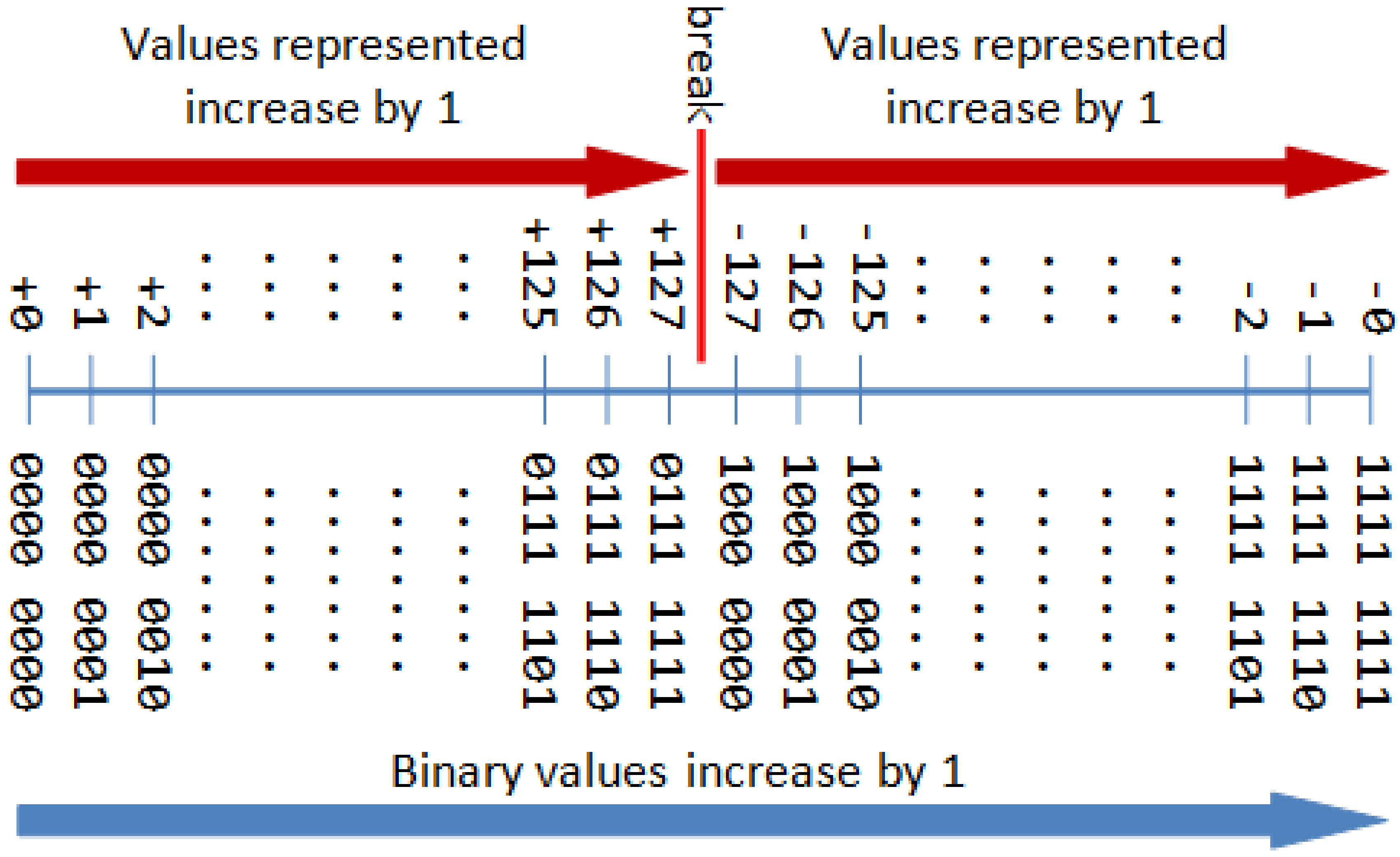
Hence, the integer is +0D

Example 4: Suppose that n=8 and the binary representation 1 111 1111B.

Sign bit is 1 ⇒ negative

Absolute value is the complement of 111 1111B, i.e., 000 0000B = 0D

Hence, the integer is -0D



1's Complement Representation

The Drawbacks Of Sign Integers In 1's Complement Representation

- There are two representations (0000 0000B and 1111 1111B) for zero.
- The positive integers and negative integers need to be processed separately.

n-bit Sign Integers in 2's Complement Representation

- In 2's complement representation:
- Again, the most significant bit (msb) is the *sign bit*, with value of 0 representing positive integers and 1 representing negative integers.
- The remaining *n*-1 bits represents the magnitude of the integer, as follows:
  - for positive integers, the absolute value of the integer is equal to "the magnitude of the (*n*-1)-bit binary pattern".
  - for negative integers, the absolute value of the integer is equal to "the magnitude of the *complement* of the (*n*-1)-bit binary pattern *plus one*" (hence called 2's complement).

Example 1: Suppose that n=8 and the binary representation 0 100 0001B.

Sign bit is 0 $\Rightarrow$ positive

Absolute value is 100 0001B = 65D

Hence, the integer is +65D

Example 2: Suppose that n=8 and the binary representation 1 000 0001B.

Sign bit is 1 $\Rightarrow$ negative

Absolute value is the complement of 000 0001B plus 1, i.e., 111 1110B + 1B = 127D

Hence, the integer is -127D

Example 3: Suppose that n=8 and the binary representation 0 000 0000B.

Sign bit is 0 ⇒ positive

Absolute value is 000 0000B = 0D

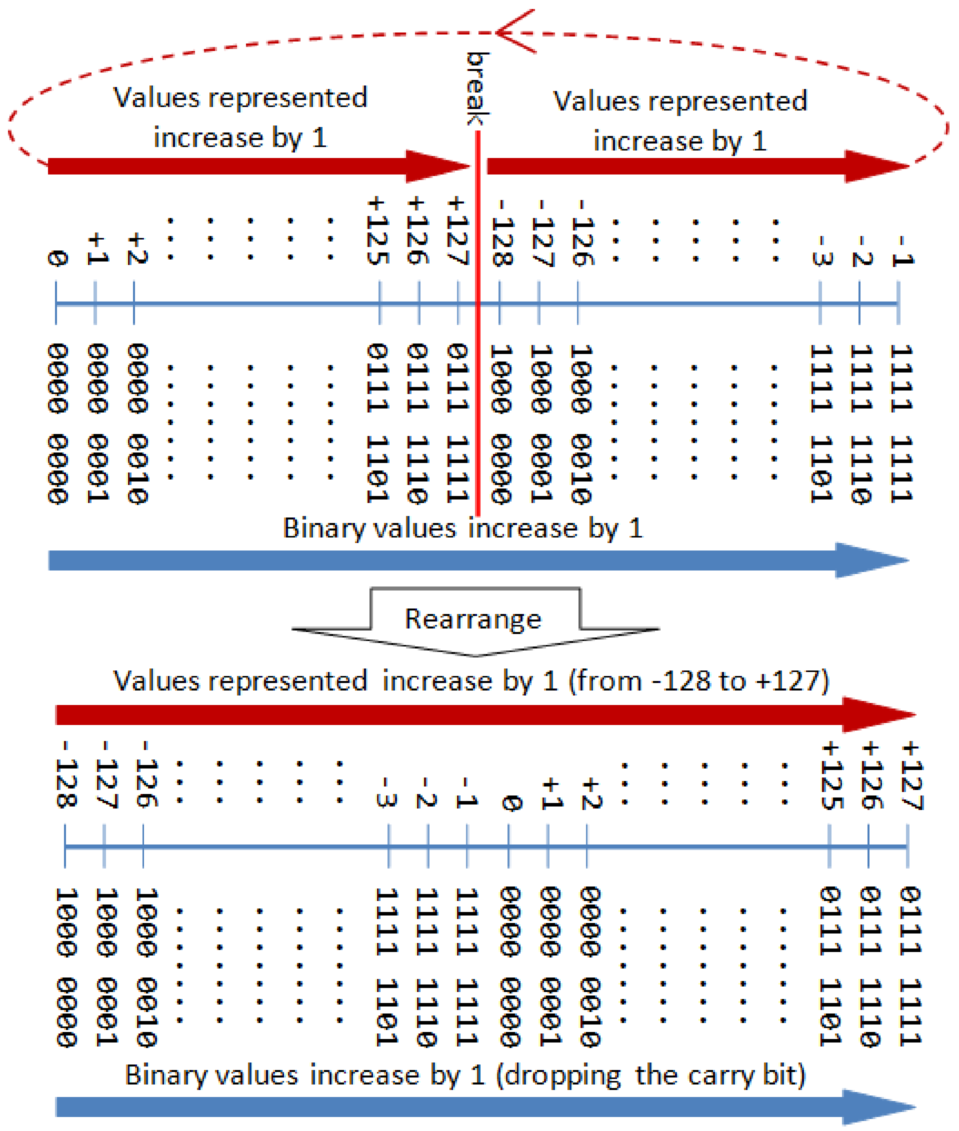
Hence, the integer is +0D

Example 4: Suppose that n=8 and the binary representation 1 111 1111B.

Sign bit is 1 ⇒ negative

Absolute value is the complement of 111 1111B plus 1, i.e., 000 0000B + 1B = 1D

Hence, the integer is -1D



2's Complement Representation

Ques **Let X be the number of distinct 16-bit integers in 2's complement representation. Let Y be the number of distinct 16-bit integers in sign magnitude representation. Then X –Y is ________**

GATE 2016 SET 2

Ans: 1

Solution: For 16- bit representation,

2’s Complement range X : (-2^{15}) to $(2^{15}-1) = -32768$ to $32767 = 65536$

signed Magnitude range Y : $-(2^{15}-1)$ to $(2^{15}-1) = -32767$ to $32767 = 65535$

$X - Y = 65536 - 65535 = 1$

Ques : The n-bit fixed-point representation of an unsigned real number X uses f bits for the fraction part. Let i = n – f. The range of decimal values for X in this representation is

GATE 2017 Set 1

(A) 2^{-f} to 2^{i}

(B) 2^{-f} to $(2^{i}-2^{-f})$

(C) 0 to 2^{i}

(D) 0 to $(2^{i}-2^{-f})$

Ans: (D) 0 to (2^i-2^{-f})

Solution: Suppose n=5

f=2

So, i=5-2=3

Min value : 000.00 = 0

Max Value: 111.11

= 7.75

= 8 – 0.25

=2^3 – 2^{-2}

Thank you

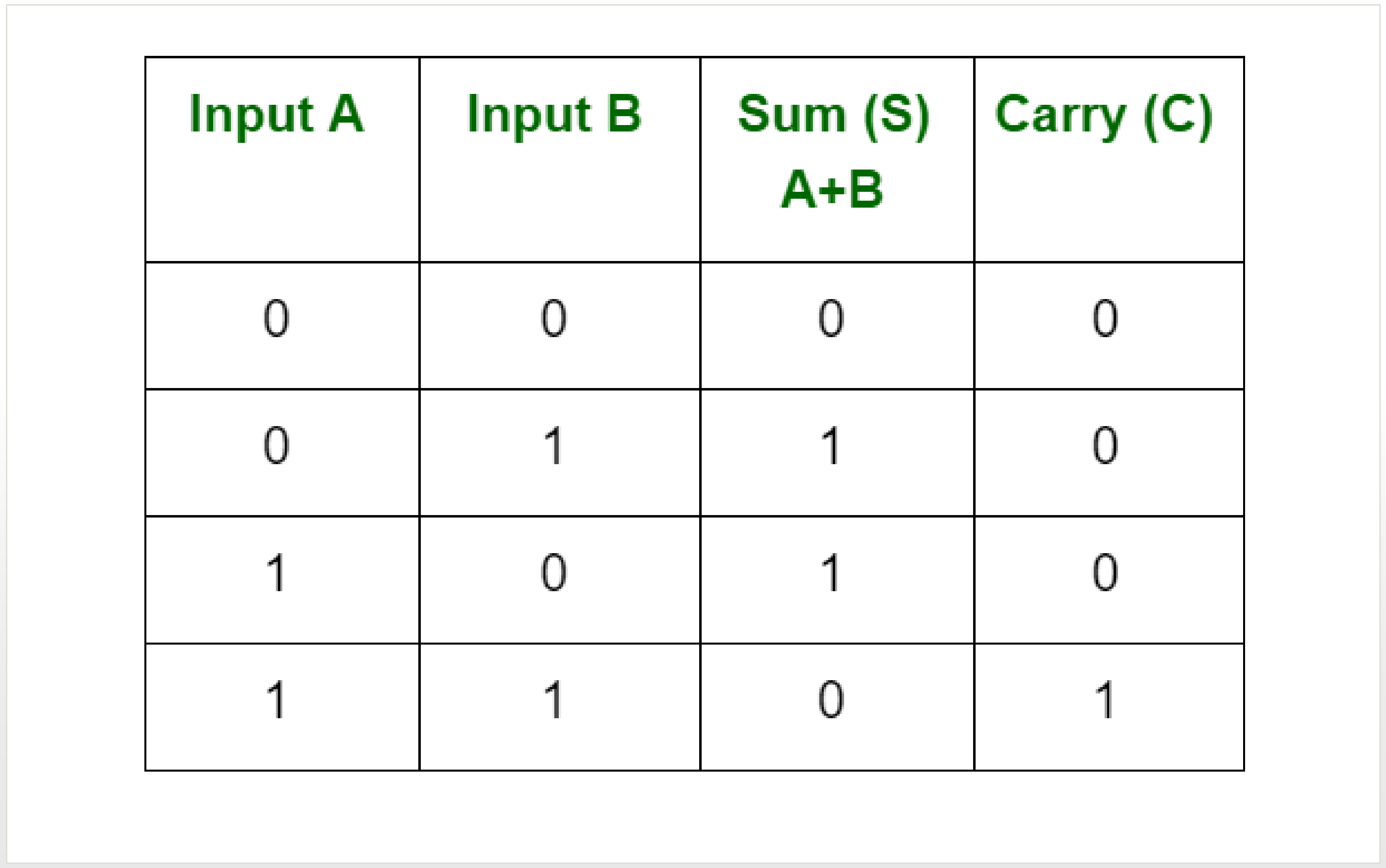
ARITHMETIC OPERATIONS IN BINARY

BY NITIN BENJAMIN DASIAH

FOUR MAIN OPERATIONS

- Addition
- Subtraction
- Multiplication
- Division

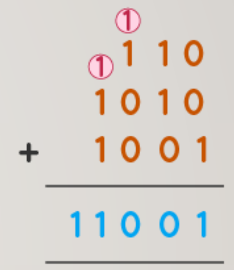
ADDITION

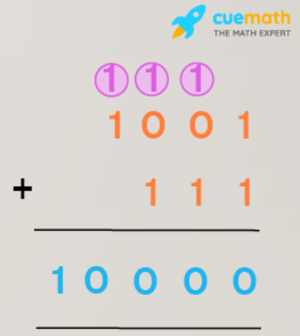


Input A	Input B	Sum (S) A+B	Carry (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

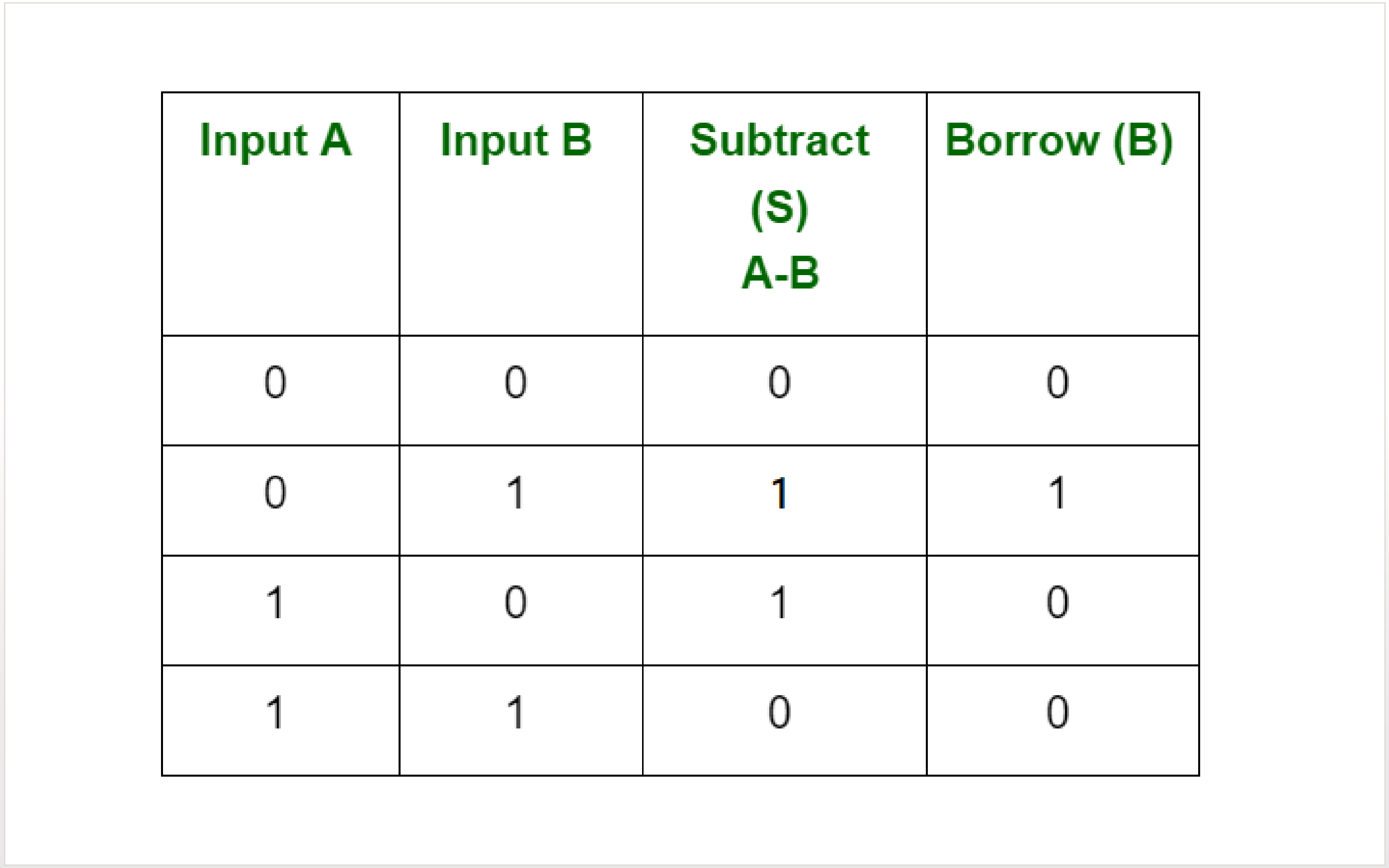
ADDITION

- Start addition from the right-most column
- Sum of a column is XOR of all the bits in the column
  $A \oplus B \oplus C_{i-1}$
- Carry for two numbers is the OR operation $AC_{i-1} + AB + BC_{i-1}$





SUBTRACTION



Input A	Input B	Subtract (S) A-B	Borrow (B)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

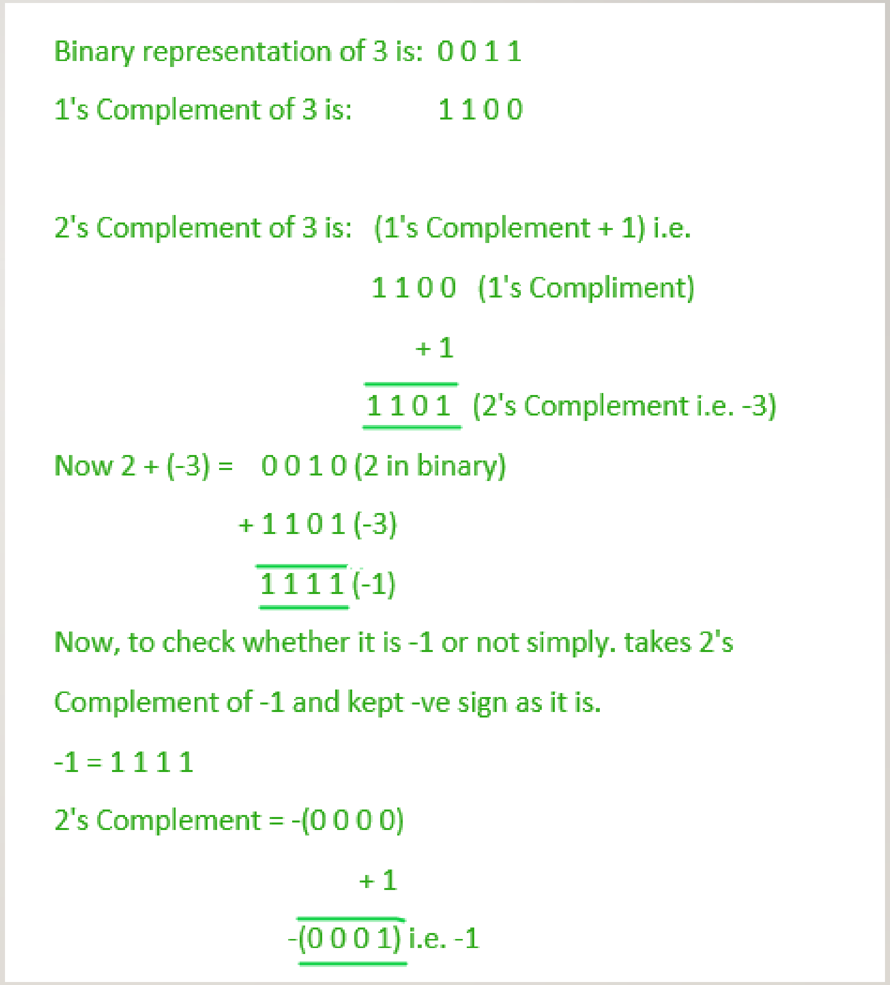
SUBTRACTION (SIGNED MAGNITUDE)

- Binary Subtraction with borrowing
- Start from the right-most column again
- If the bit in A is greater than or equal to B in the column, continue
- If bit in A is less than B in the column, borrow from the next left column

$$\begin{array}{rcccc} & 1 & 0 & 0 & 1 \\ - & & 1 & 0 & 1 \\ \hline & & 1 & 0 & 0 \\ \hline \end{array}$$

SUBTRACTION (TWO'S COMPLEMENT)

- Commonly used method in digital systems
- Take two's complement of B, (the subtrahend)
- Add A and two's complement of B
- Discard any carry



Binary representation of 3 is: 0 0 1 1

1's Complement of 3 is: 1 1 0 0

2's Complement of 3 is: (1's Complement + 1) i.e.

```
 1 1 0 0  (1's Compliment)
     + 1
 -------
 1 1 0 1  (2's Complement i.e. -3)
```

```
Now 2 + (-3) =   0 0 1 0 (2 in binary)
               + 1 1 0 1 (-3)
                 -------
                 1 1 1 1 (-1)
```

Now, to check whether it is -1 or not simply. takes 2's Complement of -1 and kept -ve sign as it is.

-1 = 1 1 1 1

```
2's Complement = -(0 0 0 0)
                       + 1
                 ---------
                 -(0 0 0 1) i.e. -1
```

GATE QUESTION

- We consider the addition of two 2's complement numbers $b_{n-1}b_{n-2}\ldots b_0$ and $a_{n-1}a_{n-2}\ldots a_0$. A binary adder for adding unsigned binary numbers is used to add the two numbers. The sum is denoted by $c_{n-1}c_{n-2}\ldots c_0$ and the carry-out by c_{out}. Which one of the following options correctly identifies the overflow condition?

(A) $c_{out}\left(\overline{a_{n-1} \oplus b_{n-1}}\right)$

(B) $a_{n-1}b_{n-1}\overline{c_{n-1}} + \overline{a_{n-1}b_{n-1}}c_{n-1}$

(C) $c_{out} \oplus c_{n-1}$

(D) $a_{n-1} \oplus b_{n-1} \oplus c_{n-1}$

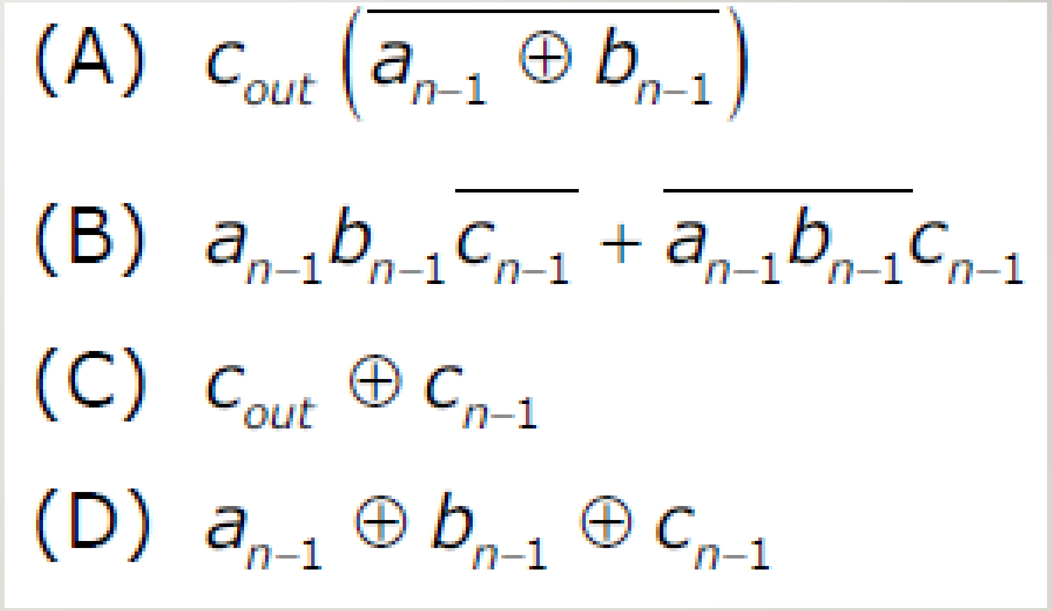
GATE QUESTION

- **Answer: (B)**

Explanation: Overflow occurs only when two same sign binary numbers added and result of these numbers is different sign in 2’s complement representation.
Otherwise overflow can not be occurred.
Counter example for given options

(A) 0111+0111=1110 has overflow, but given condition violates.
(C) 1001+0001=1010 has no overflow, but given condition violates.
(D) 1111+1111=1110 has no overflow, but given condition violates.

- Only option (B) is correct.



(A) $c_{out}\left(\overline{a_{n-1} \oplus b_{n-1}}\right)$

(B) $a_{n-1}b_{n-1}\overline{c_{n-1}} + \overline{a_{n-1}b_{n-1}}c_{n-1}$

(C) $c_{out} \oplus c_{n-1}$

(D) $a_{n-1} \oplus b_{n-1} \oplus c_{n-1}$

FLOATING POINT ADDITION

- We have to add **$1.1 * 10^3$** and **50**
- We cannot add these numbers directly. First, we need to align the exponent and then, we can add significant.
- After aligning exponent, we get **$50 = 0.05 * 10^3$**
- Now adding significant, **$0.05 + 1.1 = 1.15$**
- So, finally we get **$(1.1 * 10^3 + 50) = 1.15 * 10^3$**

FLOATING POINT ADDITION

- We follow these steps to add two numbers:

  1. Align the significant

  2. Add the significant

  3. Normalize the result

FLOATING POINT ADDITION

- **Let the two numbers be**
- x = 9.75
  y = 0.5625
- Converting them into 32-bit floating point representation,
- **9.75**'s representation in 32-bit format = **0 10000010 00111000000000000000000**
- **0.5625**'s representation in 32-bit format = **0 01111110 00100000000000000000000**
- Now we get the difference of exponents to know how much shifting is required.
- (**10000010 – 01111110**)2 = (**4**)10

FLOATING POINT ADDITION

- Now we get the difference of exponents to know how much shifting is required.
- (**10000010 – 01111110**)2 = (**4**)10
- Now, we shift the mantissa of lesser number right side by 4 units.
- Mantissa of **0.5625 = 1.00100000000000000000000**
- (note that 1 before decimal point is understood in 32-bit representation)
- Shifting right by **4** units, we get **0.00010010000000000000000**
- Mantissa of **9.75** = **1. 00111000000000000000000**
- Adding mantissa of both
- **0. 00010010000000000000000 + 1. 00111000000000000000000**

= 1. 01001010000000000000000

FLOATING POINT ADDITION

- In final answer, we take exponent of bigger number
- So, final answer consist of :
- Sign bit = **0**
- Exponent of bigger number = **10000010**
- Mantissa = **01001010000000000000000**
- 32 bit representation of answer = **x + y** = **0 10000010 01001010000000000000000**

FLOATING POINT SUBTRACTION

- Subtraction is similar to addition with some differences like we subtract mantissa unlike addition and in sign bit we put the sign of greater number.
- **Let the two numbers be**
- x = 9.75
  y = – 0.5625

FLOATING POINT SUBTRACTION

- Converting them into 32-bit floating point representation
- **9.75**’s representation in 32-bit format = **0 10000010 00111000000000000000000**
- **– 0.5625**’s representation in 32-bit format = **1 01111110 00100000000000000000000**
- Now, we find the difference of exponents to know how much shifting is required.
- (**10000010 – 01111110**)2 = (**4**)10
  Now, we shift the mantissa of lesser number right side by 4 units.
- Mantissa of **– 0.5625 = 1.00100000000000000000000**
- (note that 1 before decimal point is understood in 32-bit representation)

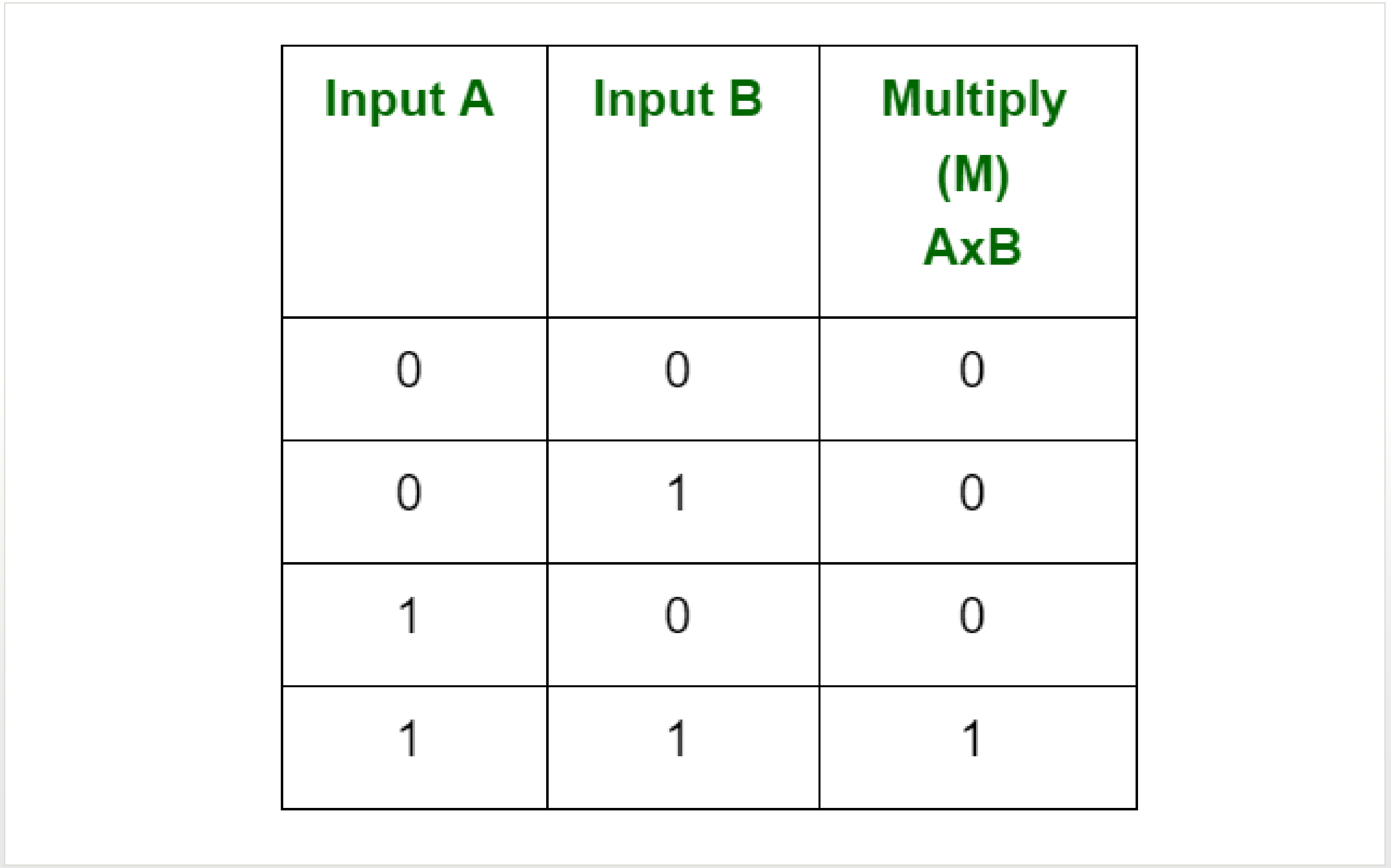
FLOATING POINT SUBTRACTION

- Shifting right by **4** units, **0.00010010000000000000000**
- Mantissa of **9.75**= **1. 00111000000000000000000**
- Subtracting mantissa of both
- **1. 00111000000000000000000**

– 0. 00010010000000000000000

- ________________________________
- **1. 00100110000000000000000**
- Sign bit of bigger number = **0**
- So, finally the answer = **x – y = 0 10000010 00100110000000000000000**

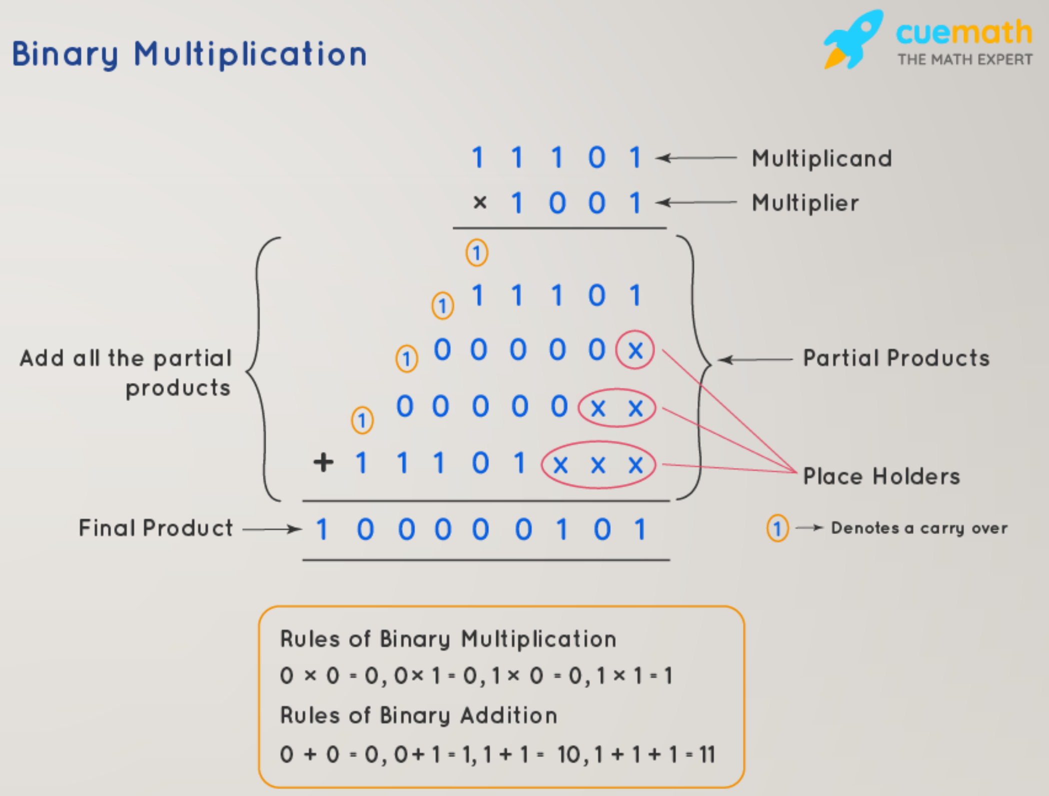
MULTIPLICATION



Input A	Input B	Multiply (M) AxB
0	0	0
0	1	0
1	0	0
1	1	1

MULTIPLICATION

- Similar to decimal multiplication
- Start from the right-most column of multiplier
- Multiply with the multiplicand after adding placeholders
- Add all partial products for final product



Rules of Binary Multiplication

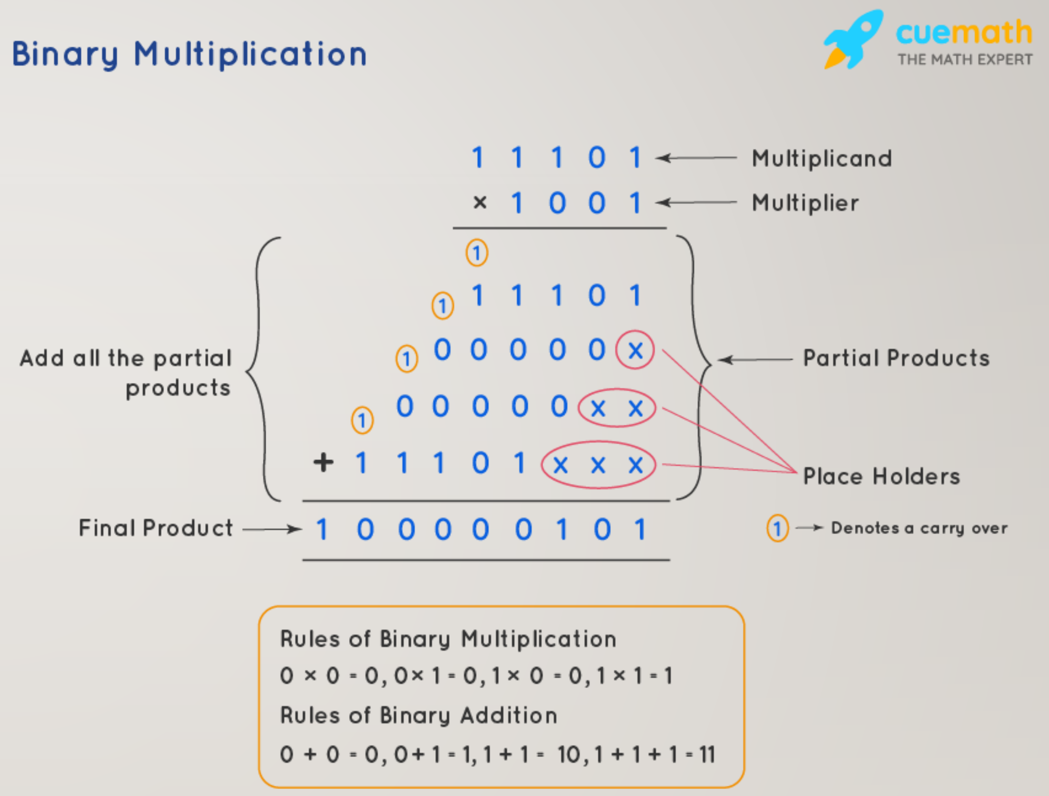
0 × 0 = 0, 0 × 1 = 0, 1 × 0 = 0, 1 × 1 = 1

Rules of Binary Addition

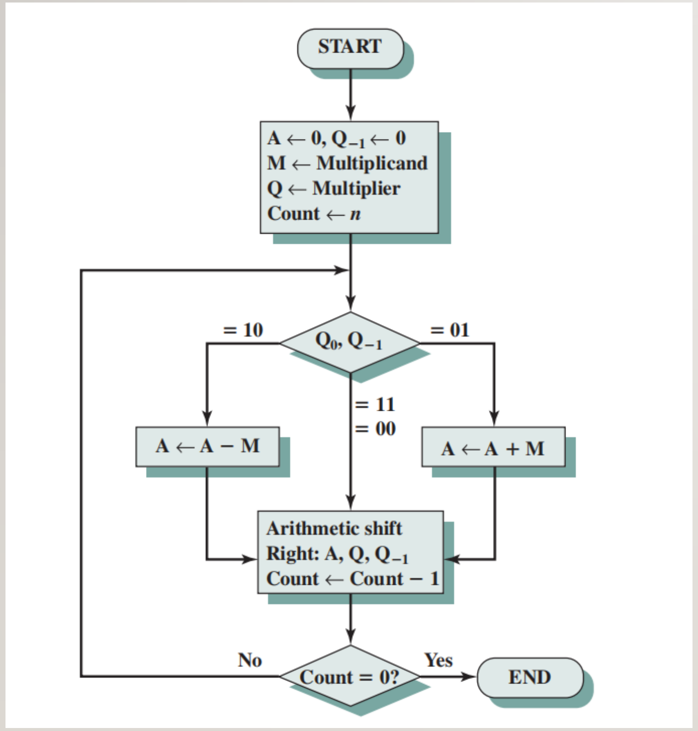
0 + 0 = 0, 0 + 1 = 1, 1 + 1 = 10, 1 + 1 + 1 = 11

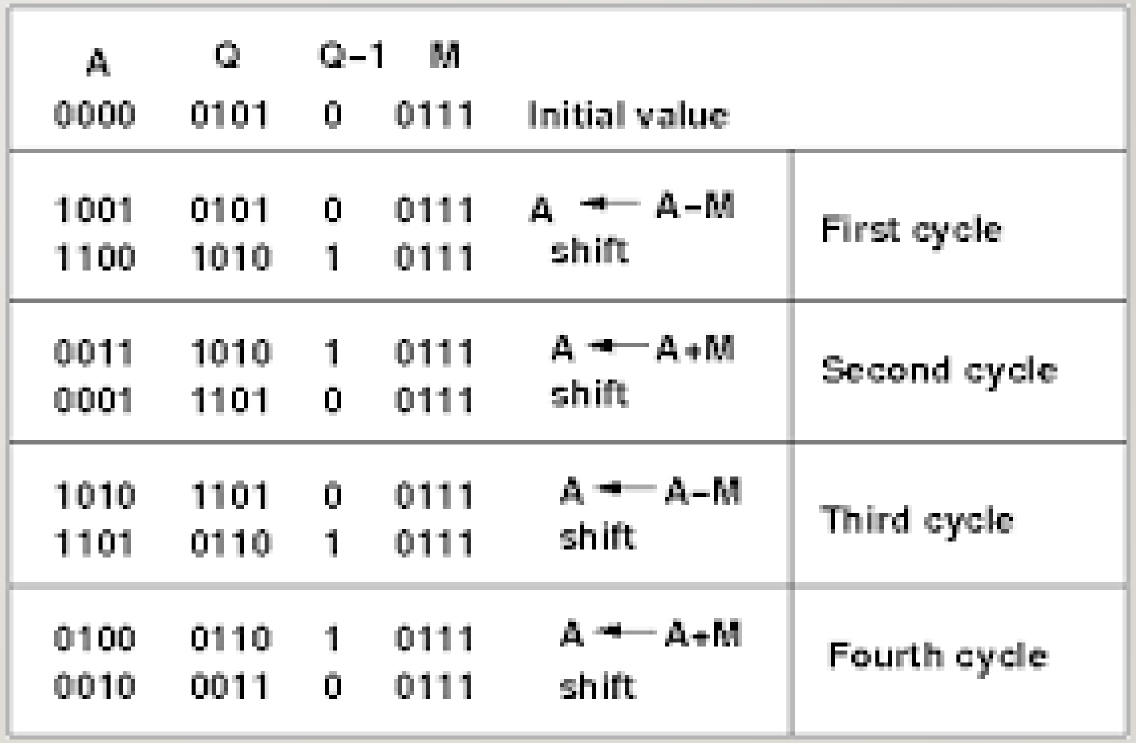
MULTIPLICATION

- When multiplying numbers in two's complement form, we must perform sign extension
- For two numbers of N bits, the resultant product will be of size 2N bits
- Sign extension increases the time complexity
- Booth's Algorithm solves the problem



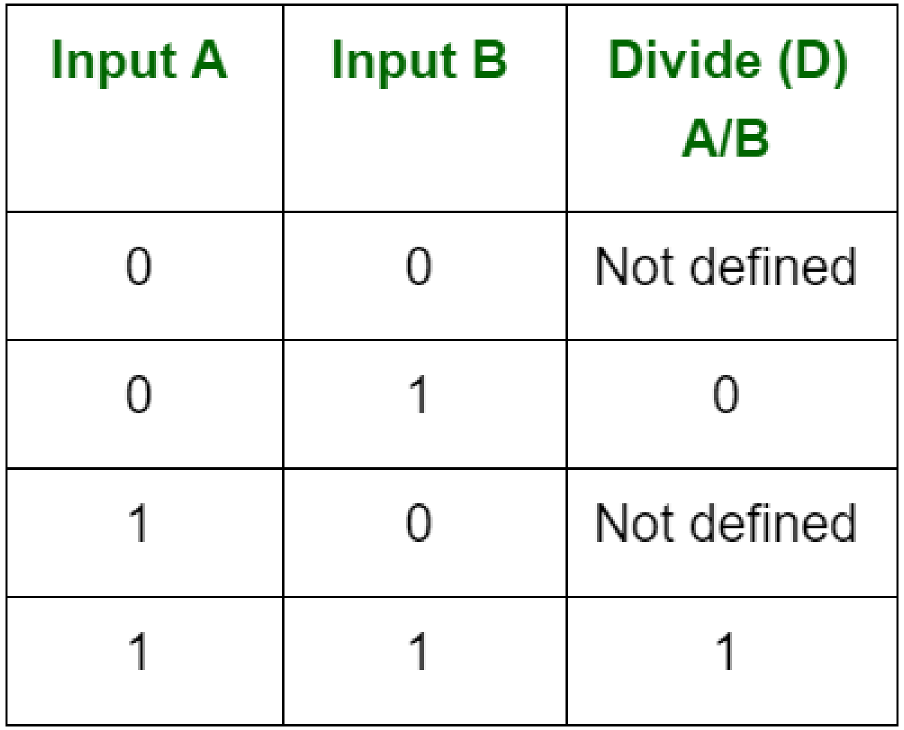
MULTIPLICATION USING BOOTH'S ALGORITHM





A	Q	Q-1	M		
0000	0101	0	0111	Initial value	
1001	0101	0	0111	A ← A−M	First cycle
1100	1010	1	0111	shift	
0011	1010	1	0111	A ← A+M	Second cycle
0001	1101	0	0111	shift	
1010	1101	0	0111	A ← A−M	Third cycle
1101	0110	1	0111	shift	
0100	0110	1	0111	A ← A+M	Fourth cycle
0010	0011	0	0111	shift	

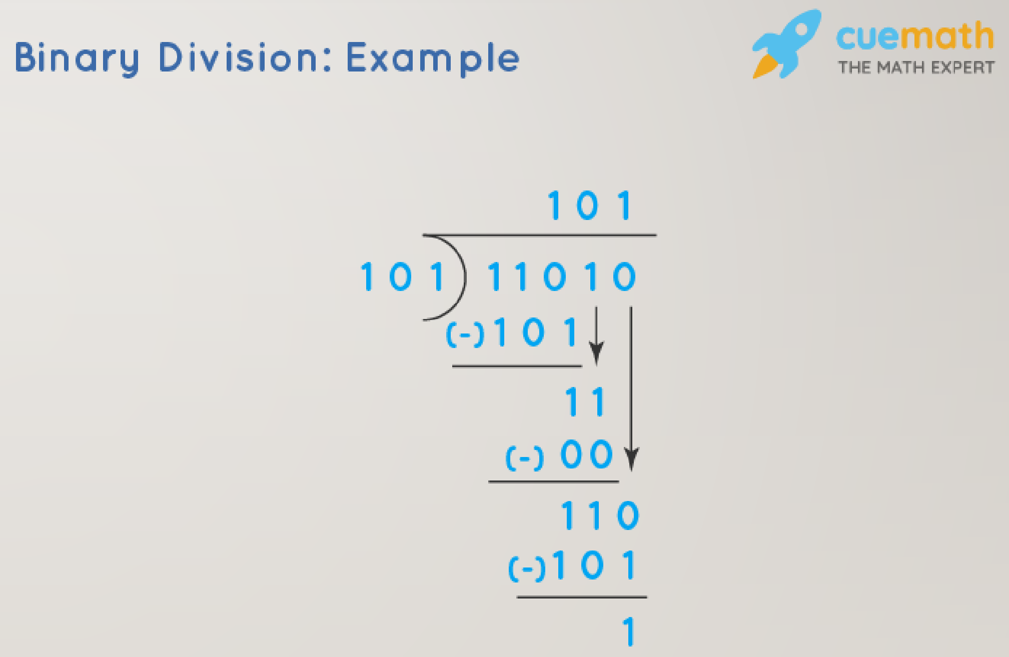
DIVISION



Input A	Input B	Divide (D) A/B
0	0	Not defined
0	1	0
1	0	Not defined
1	1	1

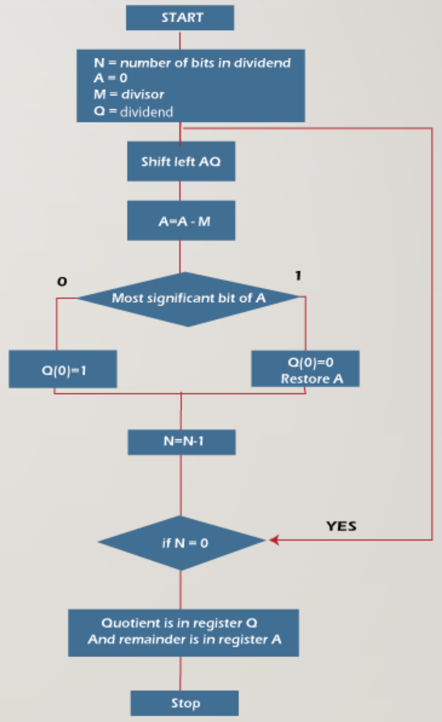
DIVISION

- Similar to decimal division
- We get quotient and remainder
- We have to follow binary subtraction rules when necessary



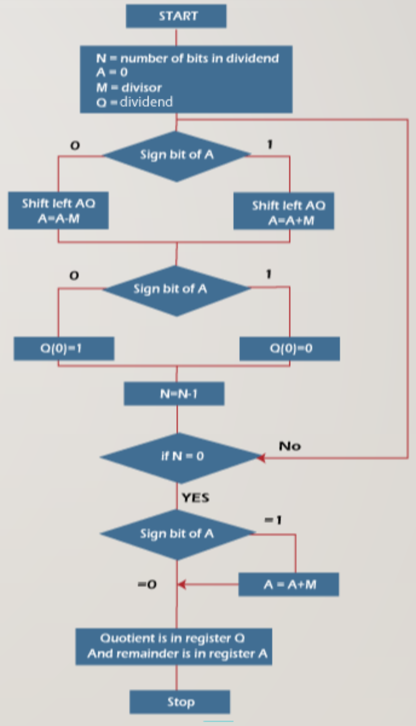
DIVISION USING RESTORING ALGORITHM

N	M	A	Q	Operation
4	00011	00000	1011	Initialize
	00011	00001	011_	Shift left AQ
	00011	11110	011_	A = A - M
	00011	00001	0110	Q[0] = 0 And restore A
3	00011	00010	110_	Shift left AQ
	00011	11111	110_	A = A - M
	00011	00010	1100	Q[0] = 0
2	00011	00101	100_	Shift left AQ
	00011	00010	100_	A = A - M
	00011	00010	1001	Q[0] = 1
1	00011	00101	001_	Shift left AQ
	00011	00010	001_	A = A - M
	00011	00010	0011	Q[0] = 1

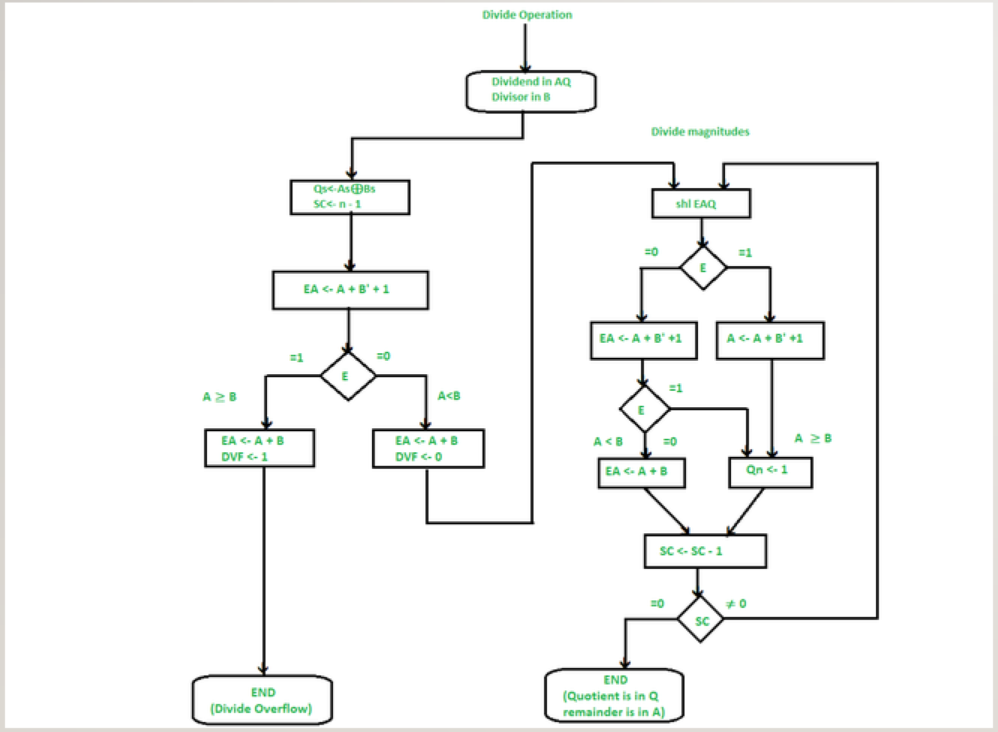

START
N = number of bits in dividend
A = 0
M = divisor
Q = dividend
Shift left AQ
A=A - M
0
1
Most significant bit of A
Q(0)=1
Q(0)=0
Restore A
N=N-1
YES
if N = 0
Quotient is in register Q
And remainder is in register A
Stop

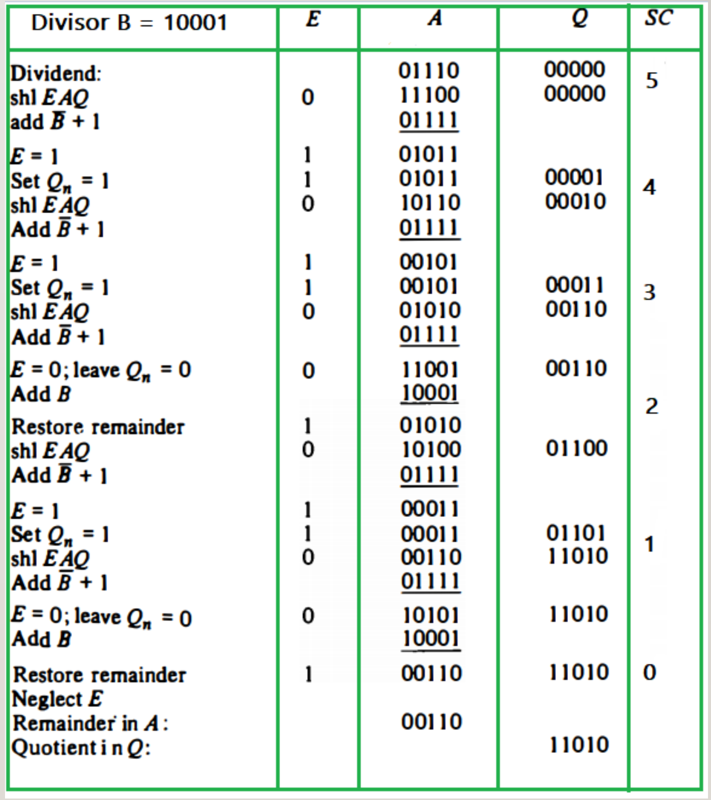
DIVISION USING NON-RESTORING ALGORITHM

N	M	A	Q	Action
4	00011	00000	1011	Begin
	00011	00001	011_	Shift left AQ
	00011	11110	011_	A = A - M
3	00011	11110	0110	Q[0] = 0
	00011	11100	110_	Shift left AQ
	00011	11111	110_	A = A + M
2	00011	11111	1100	Q[0] = 0
	00011	11111	100_	Shift left AQ
	00011	00010	100_	A = A +M
1	00011	00010	1001	Q[0] = 1
	00011	00101	001_	Shift left AQ
	00011	00010	001_	A = A - M
0	00011	00010	0011	Q[0] = 1



DIVISION FOR SIGNED INTEGERS





Divisor B = 10001	E	A	Q	SC
Dividend:		01110	00000	5
shl EAQ	0	11100	00000	
add $\overline{B}+1$		01111		
$E = 1$	1	01011		
Set $Q_n = 1$	1	01011	00001	4
shl EAQ	0	10110	00010	
Add $\overline{B}+1$		01111		
$E = 1$	1	00101		
Set $Q_n = 1$	1	00101	00011	3
shl EAQ	0	01010	00110	
Add $\overline{B}+1$		01111		
$E = 0$; leave $Q_n = 0$	0	11001	00110	
Add B		10001		
Restore remainder	1	01010		2
shl EAQ	0	10100	01100	
Add $\overline{B}+1$		01111		
$E = 1$	1	00011		
Set $Q_n = 1$	1	00011	01101	1
shl EAQ	0	00110	11010	
Add $\overline{B}+1$		01111		
$E = 0$; leave $Q_n = 0$	0	10101	11010	
Add B		10001		
Restore remainder	1	00110	11010	0
Neglect E				
Remainder in A:		00110		
Quotient i n Q:			11010	

FLOATING POINT MULTIPLICATION

- Convert these numbers in scientific notation, so that we can explicitly represent hidden 1.
- Let ‘a’ be the exponent of x and ‘b’ be the exponent of y.
- Assume resulting exponent c = a+b. It can be adjusted after the next step.
- Multiply mantissa of x to mantissa of y. Call this result m.
- If m does not have a single 1 left of radix point, then adjust radix point so it does, and adjust exponent c to compensate.
- Add sign bits, mod 2, to get sign of resulting multiplication.
- Convert back to one byte floating point representation, truncating bits if needed.

FLOATING POINT MULTIPLICATION

- Given, A = 1.11 x 2^0 and B = 1.01 x 2^2
- So, exponent c = a + b = 0 + 2 = 2 is the resulting exponent.
- Now, multiply 1.11 by 1.01, so result will be 10.0011
- We need to normalize 10.0011 to 1.00011 and adjust exponent 1 by 3 appropriately.
- Resulting sign bit 0 (XOR) 0 = 0, means positive.
- Now, truncate and normalize it 1.00011 x 2^3 to 1.000 x 2^3.

THANK YOU

IEEE Floating Point representation

Ragavi Vijayaragavan

106119098

Introduction

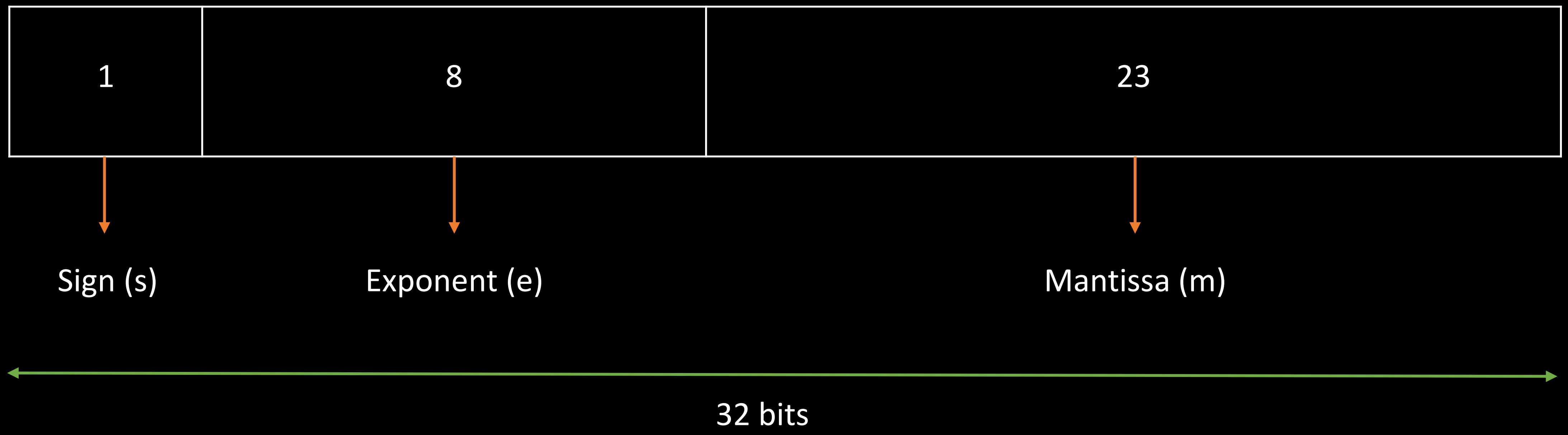
- The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation which was established in 1985 by the **Institute of Electrical and Electronics Engineers (IEEE)**.
- The standard addressed many problems found in the diverse floating point implementations that made them difficult to use reliably and reduced their portability.
- IEEE Standard 754 floating point is the most common representation today for real numbers on computers, including Intel-based PC’s, Macs, and most Unix platforms.

Components of IEEE 745 representation

- **The Sign of Mantissa –**
  This is as simple as the name. 0 represents a positive number while 1 represents a negative number.
- **The Biased exponent –**
  The exponent field needs to represent both positive and negative exponents. A bias is added to the actual exponent in order to get the stored exponent.
- **The Normalised Mantissa –**
  The mantissa is part of a number in scientific notation or a floating-point number, consisting of its significant digits. Here we have only 2 digits, i.e. O and 1. So a normalised mantissa is one with only one 1 to the left of the decimal.

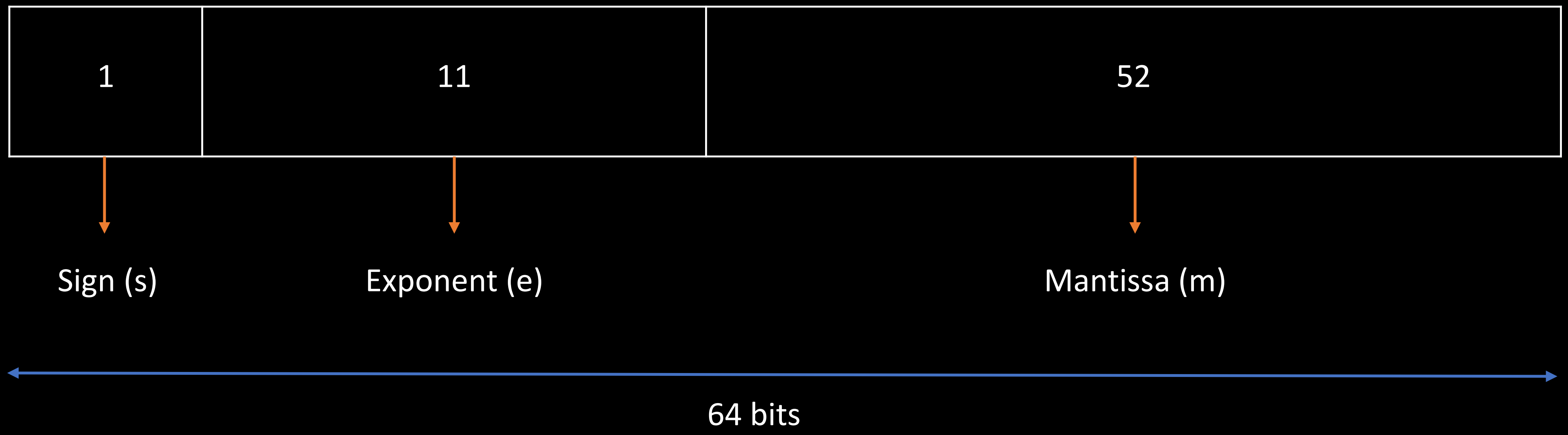
IEEE 754 numbers are divided into two based on the above three components: Single Precision and Double Precision.

Single Precision



Floating Point Calculation: $X_{FP32} = (-1)^s \times 2^{e-127} \times 1.m$

Double Precision



Floating Point Calculation: $X_{FP64} = (-1)^{s} \times 2^{e-1023} \times 1.m$

Special Values in IEEE 754

- **Zero –**
  Zero is a special value denoted with an exponent and mantissa of 0. -0 and +0 are distinct values, though they both are equal.
- **Deformalized –**
  If the exponent is all zeros, but the mantissa is not then the value is a deformalized number. This means this number does not have an assumed leading one before the binary point.

Special Values in IEEE 754

- **Infinity –**
  The values +infinity and -infinity are denoted with an exponent of all ones and a mantissa of all zeros. The sign bit distinguishes between negative infinity and positive infinity. Operations with infinite values are well defined in IEEE.
- **Not A Number (NAN) –**
  The value NAN is used to represent a value that is an error. This is represented when exponent field is all ones with a zero sign bit or a mantissa that is not 1 followed by 0s. This is a special value that miht be used to denote a variable that doesn’t yet hold a value.

Exception Values

e	**m**	**Inference**
255	0	NaN (Not a Number)
255	Not equal to 0	Infinite number
0	0	X $= (-1)^{S} \times 2^{-126} \times (0.m)$
0	Not equal to 0	0, $(-1)^{S} \times 0$ Again, +0 and -0 are possible

Converting to IEEE 754 format

Examples

What is the decimal equivalent of the floating point number 40400000H?

40400000H = 0|100 0000 0|100 0000 0000 0000 0000 0000

s=0 e=128 m=0.5

Floating Point representation,

$$X_{FP32} = (-1)^{s} \times 2^{e-127} \times 1.m$$
$$X_{FP32} = (-1)^{0} \times 2^{128-127} \times 1.5$$

= 2 x 1.5 = 3_{10}

GATE Questions

Express 10_{10} in IEEE 754 single precision floating point representation.

$$10 = (-1)^0 \times 2$$

But, $10 = 8 \times 1.25$

i.e., $10 = 2^3 \times 1.25$

$$10 = (-1)^0 \times 2^{130-127} \times 1.25$$

130 in binary form

$$130 = 128 + 2$$

$$130 = 2^7 \times 2^1$$

GATE Questions

130 = 0100 0001 0 in binary form

So 10_{10} in IEEE representation

10_{10} = 0 | 100 0001 | 0 010 0000 0000 0000 0000 0000

In Hexadecimal representation, 10_{10} = 41200000H

Converting from IEEE 754 format

GATE Questions

Evaluate the floating point number 40A00000H in decimal form

40A00000H = 0 | 100 0000 1 | 010 0000 0000 0000 0000 0000

s=0 e=129 m=0.25

$$X_{FP32} = (-1)^{s} \times 2^{e-127} \times 1.m$$
$$X_{FP32} = (-1)^{0} \times 2^{129-127} \times 1.25$$

= 4 x 125

= 5_{10}

Thank you