Unit-I: Basics of Logic Design

1. **What do you mean by number system? Explain types of number system.**

- ✓ A **number system** defines a set of values used to represent a "quantity". **OR**
- ✓ A **number system** defines a set of values or symbols used to represent quantities or magnitudes.
- ✓ **Types of Number Systems**

Sl. No.	Number System	Radix (or Base) Value	Set of Digits	Example
1	Decimal Number System	r = 10	(0, 1, 2, 3, 4, 5, 6, 7, 8, and 9)	$(792)_{10}$
2	Binary Number System	r = 2	(0,1)	$(1001)_2$
3	Octal Number System	r = 8	(0, 1, 2, 3, 4, 5, 6, and 7)	$(214)_8$
4	Hexadecimal Number System	r = 16	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F)	$(12A)_{16}$

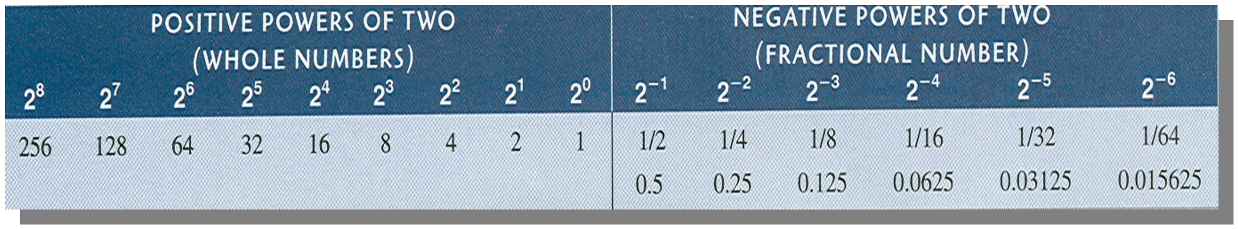
2. **What is the radix or base of the system? With the help of these systems, explain the various types of number system.**

- ✓ The **base or radix** of the number system tells about the number of symbols or digits used in the system.
- ✓ The base of a number system is indicated by a **subscript** (decimal number) and this will be followed by the value of the number

Types of Number Systems

Sl. No.	Number System	Radix (or Base) Value	Set of Digits	Example
1	Decimal Number System	r = 10	(0, 1, 2, 3, 4, 5, 6, 7, 8, and 9)	$(792)_{10}$
2	Binary Number System	r = 2	(0,1)	$(1001)_2$
3	Octal Number System	r = 8	(0, 1, 2, 3, 4, 5, 6, and 7)	$(214)_8$
4	Hexadecimal Number System	r = 16	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F)	$(12A)_{16}$

3. **What is binary number system and Explain?**
   - ✓ The binary number system has two (2) digits: 0 and 1
   - ✓ The base or radix value of binary numbering system is 2 and each position weighted by a factor of 2.
   - ✓ A binary number is a weighted number. The right-most bit is the **LSB** in a binary whole number and has a weight of $2^0 = 1$.
   - ✓ The weights for **whole numbers** are positive powers of 2 that increase from right to left, beginning with $\mathbf{2^0=1}$. The left-most bit is the **MSB.**
   - ✓ The **fractional** weights are decrease from left to right by a negative power of 2 for each bit.



POSITIVE POWERS OF TWO (WHOLE NUMBERS)									NEGATIVE POWERS OF TWO (FRACTIONAL NUMBER)					
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
256	128	64	32	16	8	4	2	1	1/2	1/4	1/8	1/16	1/32	1/64
									0.5	0.25	0.125	0.0625	0.03125	0.015625

4. **What is Decimal number system and Explain?**
- ✓ The decimal number system has ten digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.
- ✓ The decimal numbering system has a base of 10 with each position weighted by a factor of 10.
- ✓ The value of a digit is determined by its position in the number and has an assigned **weight**.
- ✓ The weights for **whole numbers** are positive powers of 10 that increase from right to left, beginning with $\mathbf{10^0=1}$.
- ✓ For **fractional numbers**, the weights are negative powers of 10 that decease from left to right beginning with $\mathbf{10^{-1}}$.

10^3	10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}
1000	100	10	1	•	1/10 =0.1	1/100 =0.01	1/1000 =0.001
Most Significant Digit **(MSB)**				Decimal point			Least Significant Digit **(LSB)**

- ✓ The value of a decimal number is the sum of the digits after each digit has been multiplied by its weight.

5. **What is octal number system and Explain?**

- ✓ The octal number system has Eight digits: 0, 1, 2, 3, 4, 5, 6 and 7
- ✓ The octal numbering system has a base of 8 with each position weighted by a factor of 8.
- ✓ An octal number is a weighted number. The right-most bit is the **LSB** in an octal whole number and has a weight of $\mathbf{8^0 = 1}$.
- ✓ The weights increase from right to left by a power of 8 for each bit. The left-most bit is the **MSB.**
- ✓ The **fractional** weights are decrease from left to right by a negative power of 8 for each bit.
- ✓ The weighted values for each position are determined as follows:

8^3	8^2	8^1	8^0		8^{-1}	8^{-2}	8^{-3}
512	64	8	1	.	1/8 = 0.125	1/64 = 0.0625	1/512 = 0.00390
Most Significant Digit **(MSB)**				Octal point			Least Significant Digit **(LSB)**

6. **What is hexadecimal number system and Explain?**

- ✓ Hexadecimal uses sixteen characters to represent numbers: the numbers **0** through **9** and the alphabetic characters **A** through **F**.
- ✓ The Hexadecimal Number System consists of digits **0 - 9** and **A -F**. The alphabets A to F represent decimal numbers from **10** to **15**.
- ✓ It has a base of 16; that is, it is composed of 16 **numeric** and alphabetic **characters**.
- ✓ Hexadecimal is a weighted number system. The column weights are powers of 16, which increase from right to left.
- ✓ As $\mathbf{16=2^4}$, hexadecimal number is represented by a group of four binary bits.
- ✓ **4** bits is a **nibble.**
- ✓ For example**,** 5 is represented by 0101.

16^3	16^2	16^1	16^0		16^{-1}	16^{-2}	16^{-3}
4096	256	16	1	.	1/16 = 0.0625	1/256 = 0.0039	1/4096 = 0.000244
Most Significant Digit **(MSB)**				Hexadecimal point			Least Significant Digit

1.2 Conversion from one number system to other

1.2.1 Binary- to –Decimal Conversion

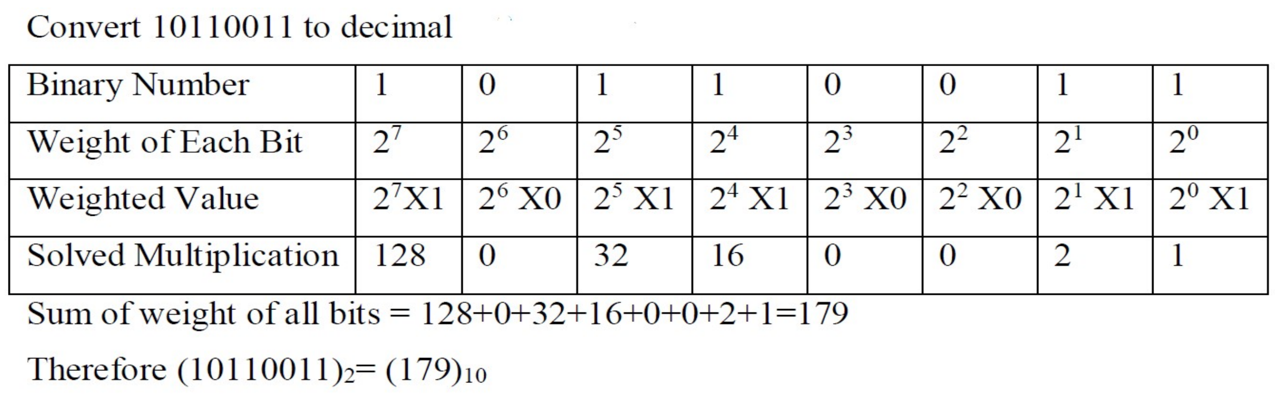
The decimal value of any binary number can be found by adding the weights of all bits that are 1 and discarding all of the bits that are 0.

1. Convert the Binary whole number 11010 to decimal

Weight of each bit	**2^4**	**2^3**	**2^2**	**2^1**	**2^0**
Binary Number	**1**	**1**	**0**	**1**	**1**
Decimal Value	**16**	**8**	**0**	**2**	**1**

Sum of weight of all bits = 16 + 8 + 0 + 2 + 0 =26

The decimal equivalent of $(11010)_2$ is $(26)_{10}$



Convert 10110011 to decimal

Binary Number	1	0	1	1	0	0	1	1
Weight of Each Bit	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Weighted Value	2^7X1	2^6 X0	2^5 X1	2^4 X1	2^3 X0	2^2 X0	2^1 X1	2^0 X1
Solved Multiplication	128	0	32	16	0	0	2	1

Sum of weight of all bits = 128+0+32+16+0+0+2+1=179

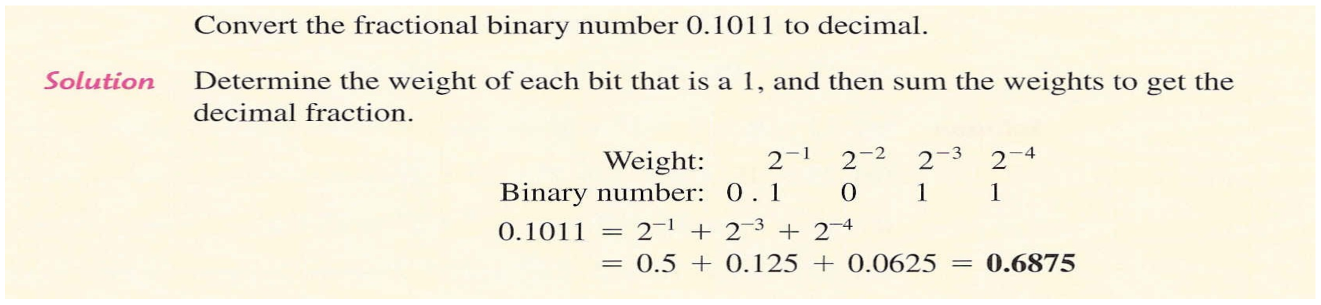
Therefore $(10110011)_2$= $(179)_{10}$

2. Convert the fractional binary number 0.01101 to decimal

Weight of each bit	**2^{-1}**	**2^{-2}**	**2^{-3}**	**2^{-4}**	**2^{-5}**
Binary Number	**0**	**1**	**1**	**0**	**1**
Decimal Value	**1/2 =0.5**	**1/4 =0.25**	**1/8 =0.125**	**1/16 =0.0625**	**1/32 =0.03125**

Sum of weight of all bits = 0 + 0.25 + 0.125 + 0 + 0.03125 = 0.40625

The decimal equivalent of $(0.01101)_2$ is $(0.40625)_{10}$



Convert the fractional binary number 0.1011 to decimal.

Solution Determine the weight of each bit that is a 1, and then sum the weights to get the decimal fraction.

Weight:	2^{-1}	2^{-2}	2^{-3}	2^{-4}
Binary number:	0 . 1	0	1	1

$$0.1011 = 2^{-1} + 2^{-3} + 2^{-4}$$
$$= 0.5 + 0.125 + 0.0625 = \mathbf{0.6875}$$

Example 1.3 Convert binary $(110100)_2$ to its decim

Solution.

$(110100)_2=1X2^5+1X2^4+0X2^3+1X2^2+0X2^1+0X2^0$

$=32+16+0+4+0+0$

$(110100)_2=(52)_{10}$

Example 1.4 Converting binary fraction (11101
equivalent decimal fraction.

Solution.

$(111011.101)_2=(1X2^5+1X2^4+1X2^3+0X2^2+1X2^1+1$
$^1+0X2^{-2}+1X2^{-3})$

$=(32+16+8+0+2+1)+(0.5+0+0.125$

$=(59.625)_{10}$

1.2.2 Decimal - to – Binary Conversion

A decimal number can be converted to binary number by using

- Sum-of-weights method or
- Repeated division-by-2 method

1 Sum-of-weights method determine the set of binary weights whose sum is equal to the decimal number.

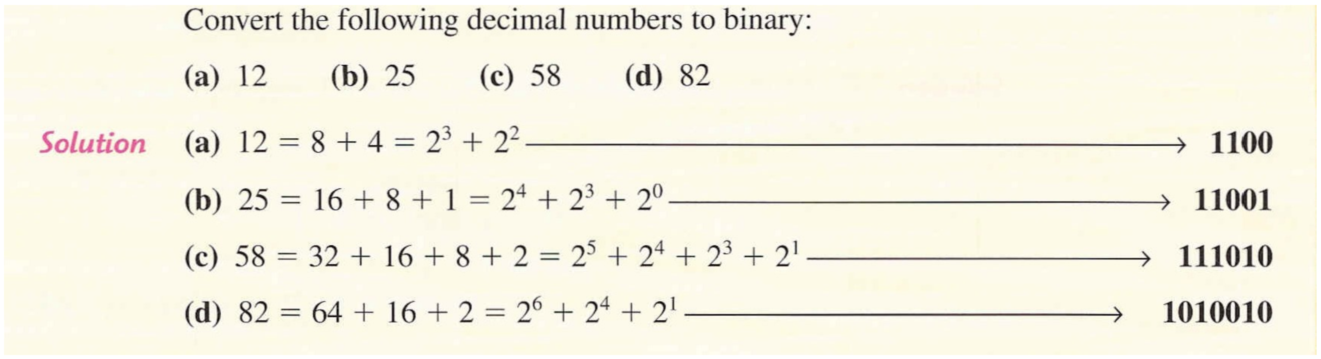
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1

The decimal number 9 can be expressed as the sum of binary weights as follows

$9 = 8 + 1$ or $9 = 2^3 + 2^0$

Placing 1s in the appropriate weight positions, 2^3 and 2^0, and 0s in the 2^2 and 2^1 positions determines the binary number for decimal 9.

2^3	2^2	2^1	2^0	
1	0	0	1	Binary number for decimal 9


Convert the following decimal numbers to binary:

(a) 12 **(b)** 25 **(c)** 58 **(d)** 82

Solution **(a)** $12 = 8 + 4 = 2^3 + 2^2 \longrightarrow$ **1100**

(b) $25 = 16 + 8 + 1 = 2^4 + 2^3 + 2^0 \longrightarrow$ **11001**

(c) $58 = 32 + 16 + 8 + 2 = 2^5 + 2^4 + 2^3 + 2^1 \longrightarrow$ **111010**

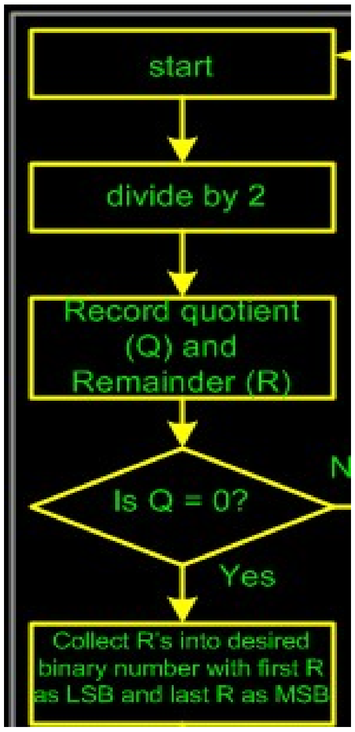
(d) $82 = 64 + 16 + 2 = 2^6 + 2^4 + 2^1 \longrightarrow$ **1010010**

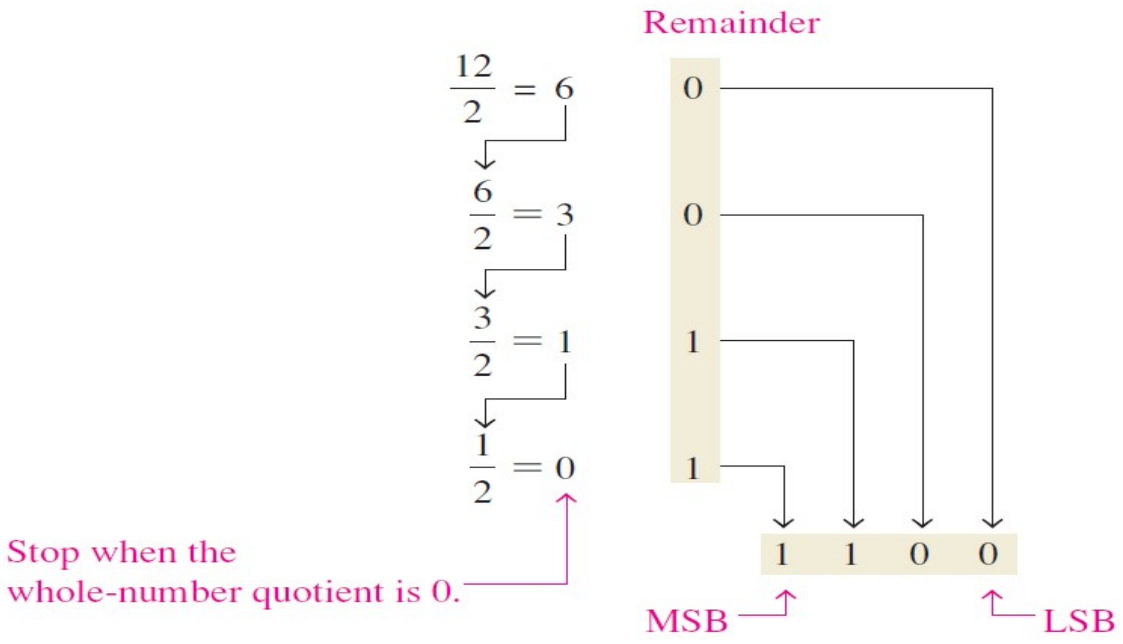
2 Repeated Division-by-2 Method

This method involves the following steps:

1. Divide the integer part of decimal number by 2.
2. Note the remainder separately.(remainders will be either 1 or 0)
3. Consider quotient as a new decimal number and repeat the process of dividing by until the quotient is 0 and keep writing the remainders after each step of division.
4. Write down the remainders in reverse order. The first remainder to be produced is the **LSB (least Significant Bit)** in the binary number and the last remainder is the **MSB (Most Significant Bit)**

The Flow chart for repeated-division method is as follows:

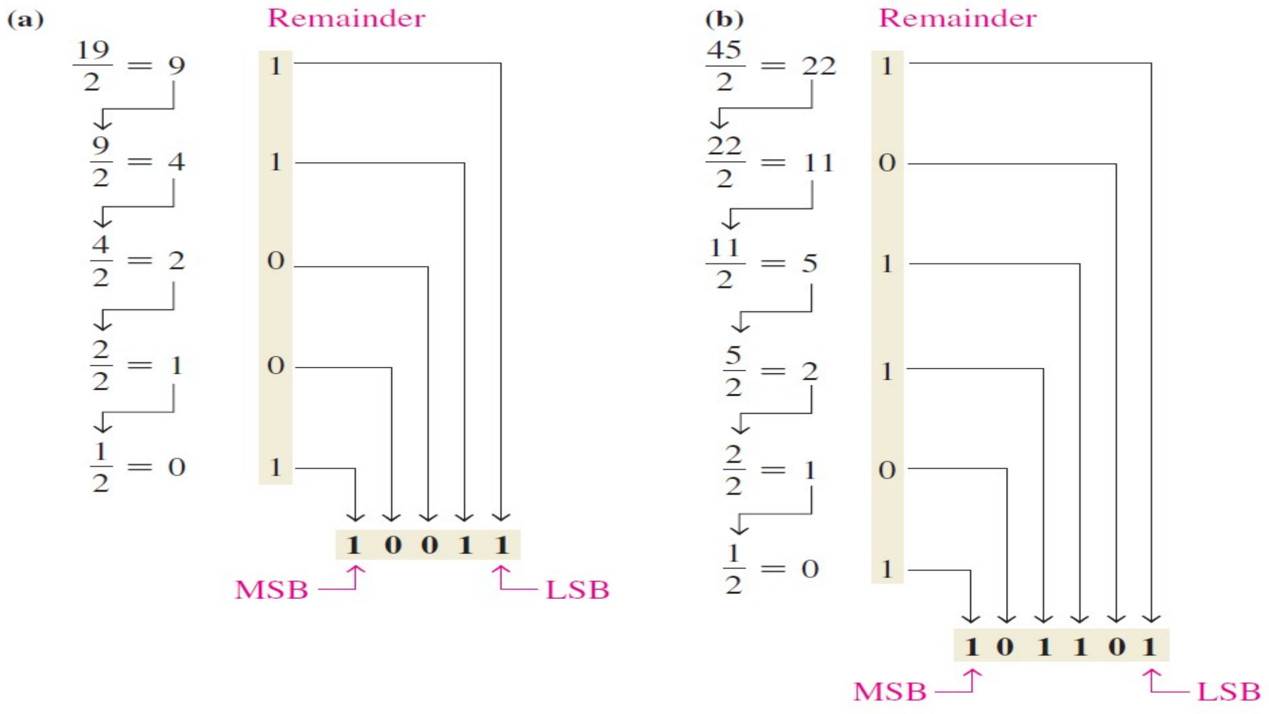

start
divide by 2
Record quotient (Q) and Remainder (R)
Is Q = 0?
N
Yes
Collect R's into desired binary number with first R as LSB and last R as MSB



Convert the following decimal numbers to binary:

(a) 19 **(b)** 45

Solution



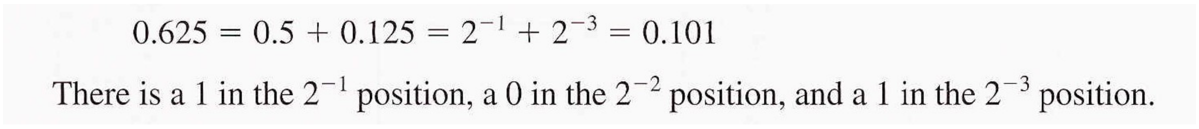
Convert 671 to binary

2	671	Remainder
2	335	1 LSB
2	167	1
2	83	1
2	41	1
2	20	1
2	10	0
2	5	0
2	2	1
2	1	0
	0	1 MSB

Taking remainders in reverse order 1010011111, therefore $(671)_{10}$ is equivalent to $(1010011111)_2$

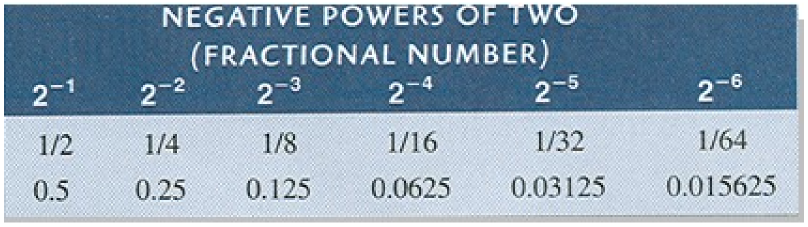
Converting Decimal Fractions to Binary

i) Sum of Weights: The sum of weights method can be applied to fractional decimal numbers as shown in the folloeing example



$$0.625 = 0.5 + 0.125 = 2^{-1} + 2^{-3} = 0.101$$

There is a 1 in the 2^{-1} position, a 0 in the 2^{-2} position, and a 1 in the 2^{-3} position.



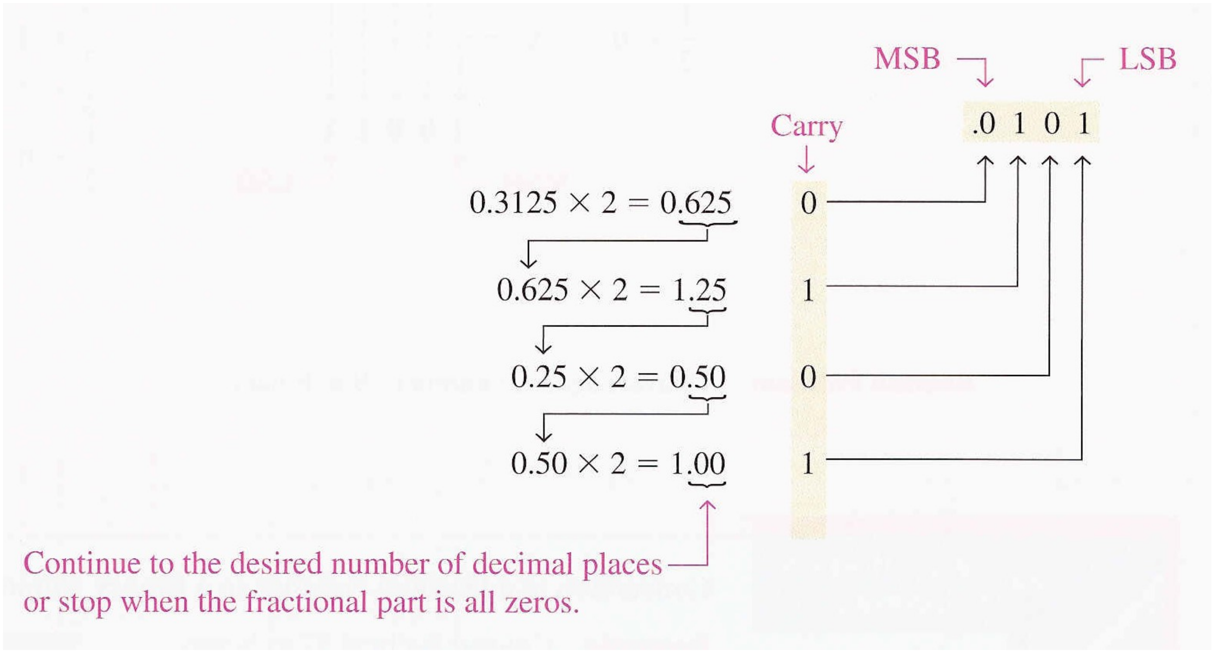
NEGATIVE POWERS OF TWO (FRACTIONAL NUMBER)

2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
1/2	1/4	1/8	1/16	1/32	1/64
0.5	0.25	0.125	0.0625	0.03125	0.015625

ii) Repeated Multiplication by 2

1. The **fractional part** of the given decimal number is multiplied repeatedly by 2.
2. Record the integer part of the product as carry and fractional part as new fractional part
3. Repeate steps 1 and 2 until fractional part of product becomes zero (0) or until the desired number of decimal places is reached.
4. Write down the carries in forward order to form the binary equivalent of thefraction decimal number. The first carry produced is the MSB, and the last carry is the LSB.

Convert the decimal fraction 0.3125 to binary



Example 1.6 Convert decimal fraction $(12.75)_{10}$ t
equivalent binary fraction.

```
2|12    Remainder            MSB   .75
 2|6       0                       X 2
 2|3       0                       1.50
  1        1                      X 2
                                   1.00
```

So, $(12)_{10}=(1100)_2$ and $(.75)_{10}=($

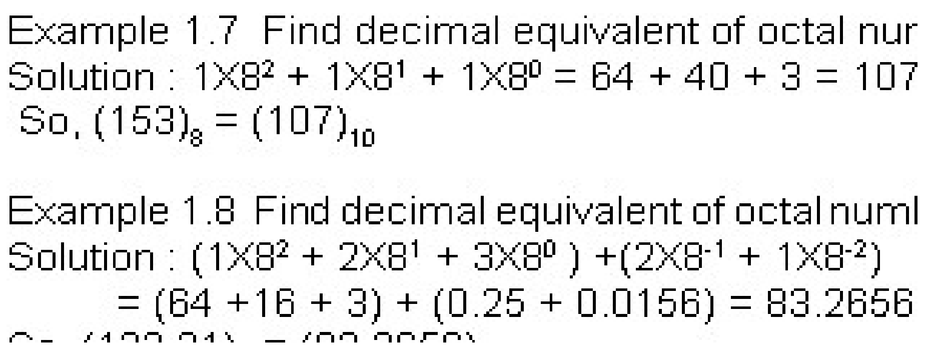
1.2.3 Octal to Decimal Conversion

When converting from octal to decimal, each digit must be multiplied by its weight and summing the products as illustrated here for 2374_8.

Weight: $8^3\ 8^2\ 8^1\ 8^0$

Octal number: 2 3 7 4

$$\begin{aligned}2374_8 &= (2\times 8^3) + (3\times 8^2) + (7\times 8^1) + (4\times 8^0)\\ &= (2\times 512) + (3\times 64) + (7\times 8) + (4\times 1)\\ &= 1024 + 192 + 56 + 4 = 1276_{10}\end{aligned}$$


Example 1.7 Find decimal equivalent of octal nur

Solution : $1\times8^2 + 1\times8^1 + 1\times8^0 = 64 + 40 + 3 = 107$

So, $(153)_8 = (107)_{10}$

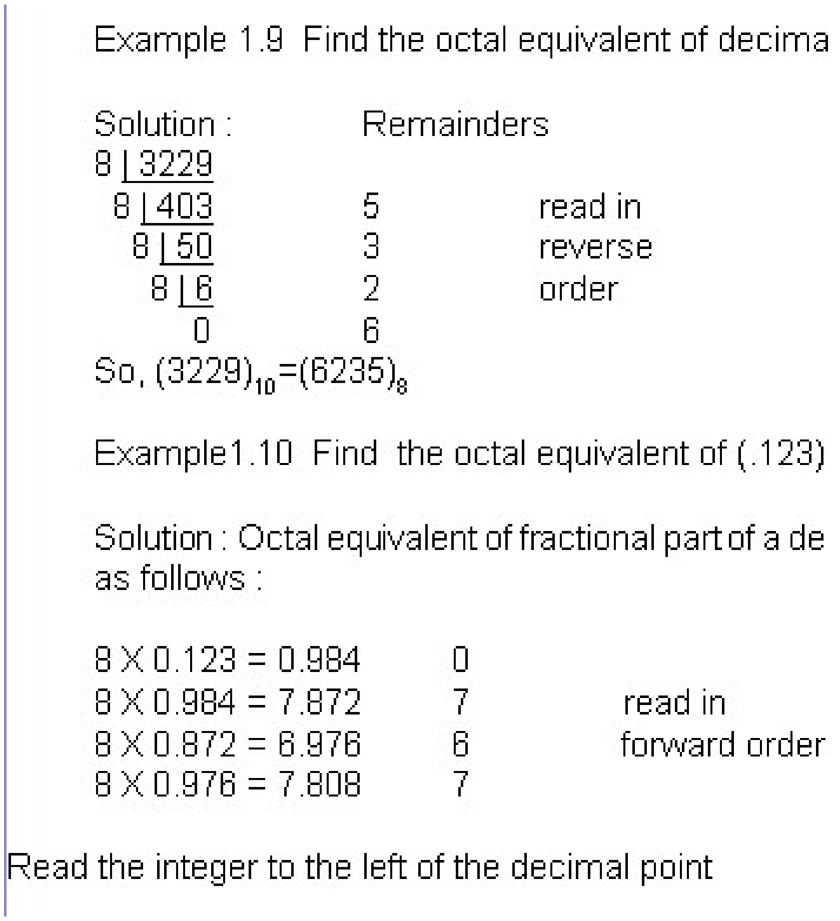
Example 1.8 Find decimal equivalent of octal numl

Solution : $(1\times8^2 + 2\times8^1 + 3\times8^0) + (2\times8^{-1} + 1\times8^{-2})$

$= (64 + 16 + 3) + (0.25 + 0.0156) = 83.2656$

1.2.4 Decimal to Octal Conversion

The procedure for conversion of decimal numbers to octal numbers is exactly similar to the conversion of decimal number to binary numbers except replacing 2 by 8.


Example 1.9 Find the octal equivalent of decima

```
Solution :          Remainders
8 | 3229
  8 | 403           5           read in
    8 | 50          3           reverse
      8 | 6         2           order
          0         6
```

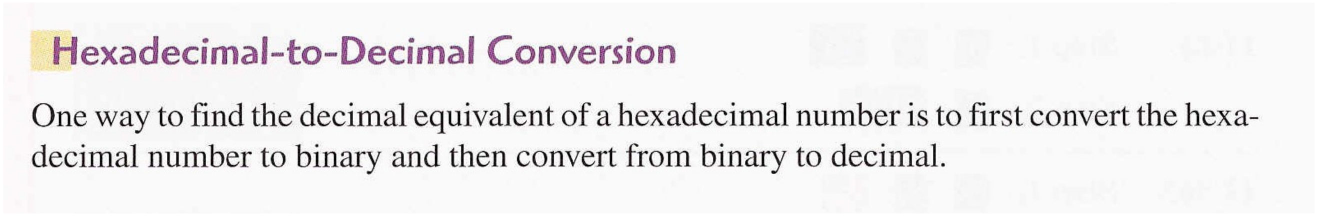
So, $(3229)_{10}=(6235)_8$

Example1.10 Find the octal equivalent of (.123)

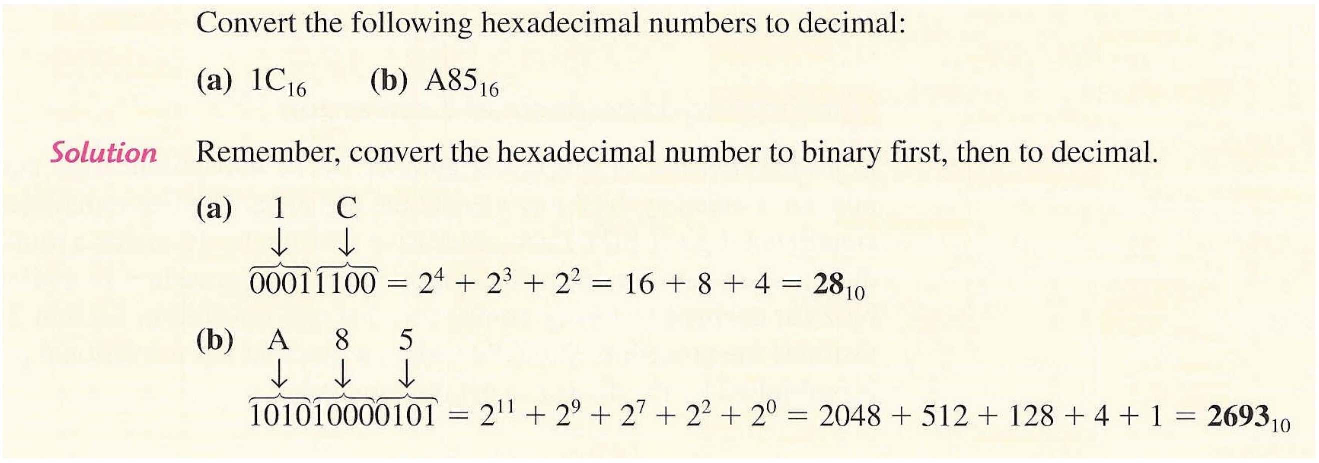
Solution : Octal equivalent of fractional part of a de
as follows :

```
8 X 0.123 = 0.984     0
8 X 0.984 = 7.872     7          read in
8 X 0.872 = 6.976     6          forward order
8 X 0.976 = 7.808     7
```

Read the integer to the left of the decimal point


Hexadecimal-to-Decimal Conversion

One way to find the decimal equivalent of a hexadecimal number is to first convert the hexadecimal number to binary and then convert from binary to decimal.



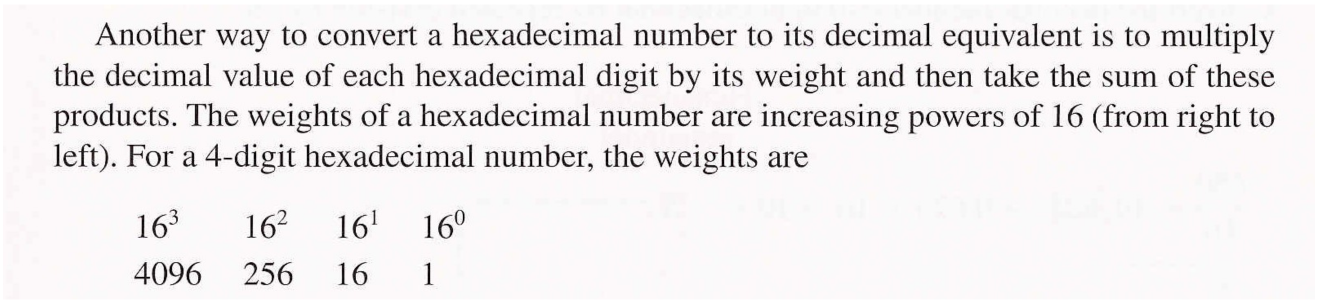
Convert the following hexadecimal numbers to decimal:

(a) $1C_{16}$ **(b)** $A85_{16}$

Solution Remember, convert the hexadecimal number to binary first, then to decimal.

(a)
$$\begin{array}{cc} 1 & C \\ \downarrow & \downarrow \end{array}$$
$$\underbrace{0001}\underbrace{1100} = 2^4 + 2^3 + 2^2 = 16 + 8 + 4 = \mathbf{28_{10}}$$

(b)
$$\begin{array}{ccc} A & 8 & 5 \\ \downarrow & \downarrow & \downarrow \end{array}$$
$$\underbrace{1010}\underbrace{1000}\underbrace{0101} = 2^{11} + 2^9 + 2^7 + 2^2 + 2^0 = 2048 + 512 + 128 + 4 + 1 = \mathbf{2693_{10}}$$

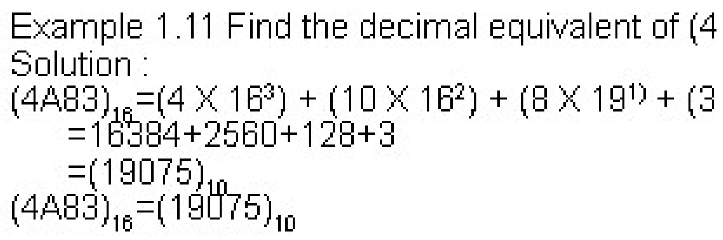


Another way to convert a hexadecimal number to its decimal equivalent is to multiply the decimal value of each hexadecimal digit by its weight and then take the sum of these products. The weights of a hexadecimal number are increasing powers of 16 (from right to left). For a 4-digit hexadecimal number, the weights are

16^3	16^2	16^1	16^0
4096	256	16	1

Example:

$2AF_{16} = 2 \times (16^2) + 10 \times (16^1) + 15 \times (16^0) = 687_{10}$



Example 1.11 Find the decimal equivalent of (4

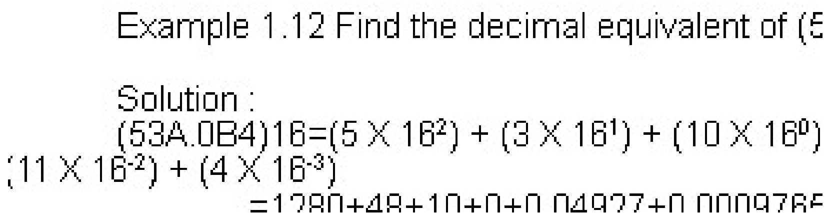
Solution :

$(4A83)_{16}$=(4 X 16^3) + (10 X 16^2) + (8 X 19^1) + (3

=16384+2560+128+3

=$(19075)_{10}$

$(4A83)_{16}$=$(19075)_{10}$



Example 1.12 Find the decimal equivalent of (5

Solution :

(53A.0B4)16=(5 X 16^2) + (3 X 16^1) + (10 X 16^0)

(11 X 16^{-2}) + (4 X 16^{-3})

=1280+48+10+0+0.04927+0.000976

1.2.6 Decimal to Hexadecimal Conversion

Integer Part Conversion

To convert a decimal integer number to hexadecimal, successively divide the given decimal number by 16 till the quotient is zero. The last remainder is the MSB (Most Significant Bit). The remainders read from bottom to top give the equivalent hexadecimal integer.

Fraction Part Conversion

To convert a decimal fraction to hexadecimal, successively multiply the given decimal fraction by 16, till the product is zero or till the required accuracy is obtained, and collect all the integers to the left of decimal point. The first integer is the MSB and the integers read from top to bottom give the hexadecimal fraction.

Example 1.13 Convert decimal $(1234.675)_{10}$ to
Solution :

1st consider $(1234)_{10}$

	Remainder Decimal	Remainder Hexade
$16\underline{\mid 1234}$	2	2
$16\underline{\mid 77}$	13	D
$16\underline{\mid 4}$	4	4

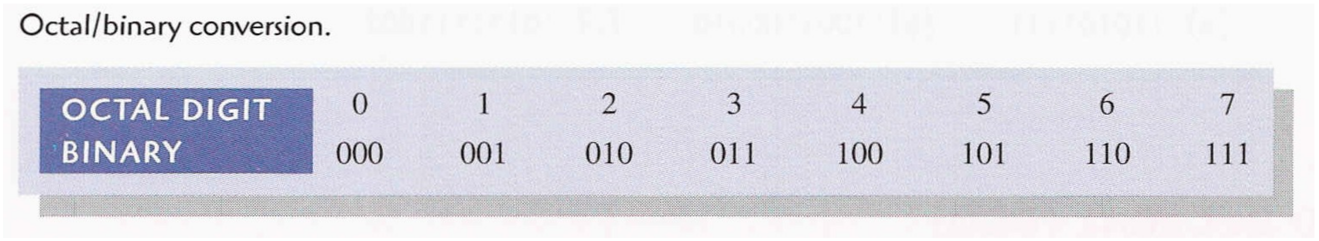
Hexadecimal conversion of fractional part of a decimal number as follows :

	Decimal	Hexa
$0.675 \times 16 = 10.8$	10	
$0.800 \times 16 = 12.8$	12	
$0.800 \times 16 = 12.8$	12	
$0.800 \times 16 = 12.8$	12	

$(0.675)_{10} = (0.ACC)_{16}$

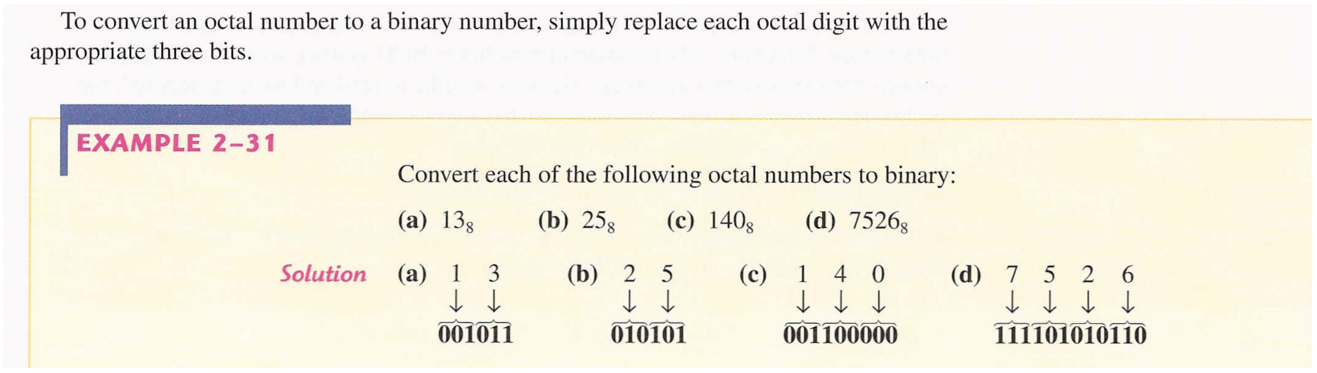
1.2.7 Octal- to- Binary Conversion

When converting from octal to binary, change each octal digit to its 3-bit binary equivalent as shown in following table. The octal number system is a convenient way to represent binary numbers.



Octal/binary conversion.

OCTAL DIGIT	0	1	2	3	4	5	6	7
BINARY	000	001	010	011	100	101	110	111


To convert an octal number to a binary number, simply replace each octal digit with the appropriate three bits.

EXAMPLE 2–31

Convert each of the following octal numbers to binary:

(a) 13_8 **(b)** 25_8 **(c)** 140_8 **(d)** 7526_8

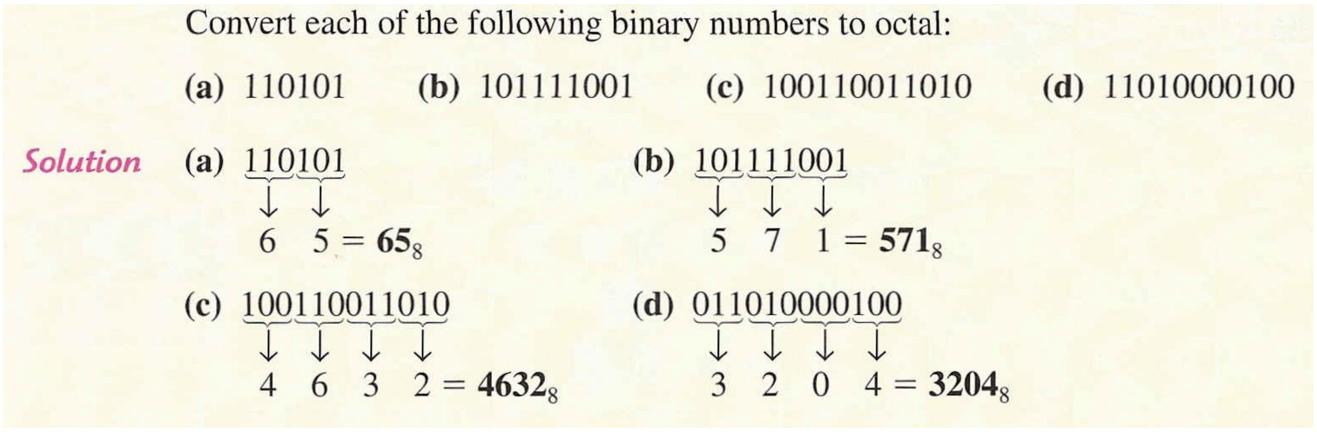
Solution

(a) $\begin{matrix}1&3\\ \downarrow&\downarrow\\ \overbrace{001}&\overbrace{011}\end{matrix}$ **(b)** $\begin{matrix}2&5\\ \downarrow&\downarrow\\ \overbrace{010}&\overbrace{101}\end{matrix}$ **(c)** $\begin{matrix}1&4&0\\ \downarrow&\downarrow&\downarrow\\ \overbrace{001}&\overbrace{100}&\overbrace{000}\end{matrix}$ **(d)** $\begin{matrix}7&5&2&6\\ \downarrow&\downarrow&\downarrow&\downarrow\\ \overbrace{111}&\overbrace{101}&\overbrace{010}&\overbrace{110}\end{matrix}$

1.2.8 Binary to octal Conversion

Procedure to convert Binary to Octal is as follows:

1. Start with the right-most group of tree bits and, moving from right to left, convert each 3-bit group to the equivalent octal digit.
2. If there are not three bits available for left-most group, add either one or two zeros to make a complete group. These leading zeros do not affect the value of the binary number.

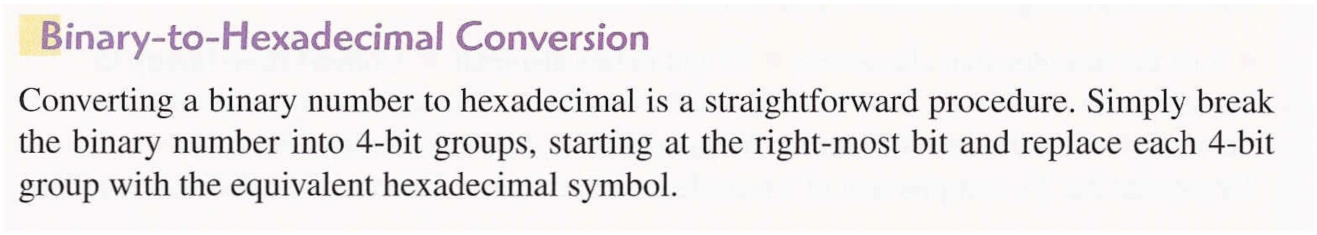

Convert each of the following binary numbers to octal:

(a) 110101 **(b)** 101111001 **(c)** 100110011010 **(d)** 11010000100

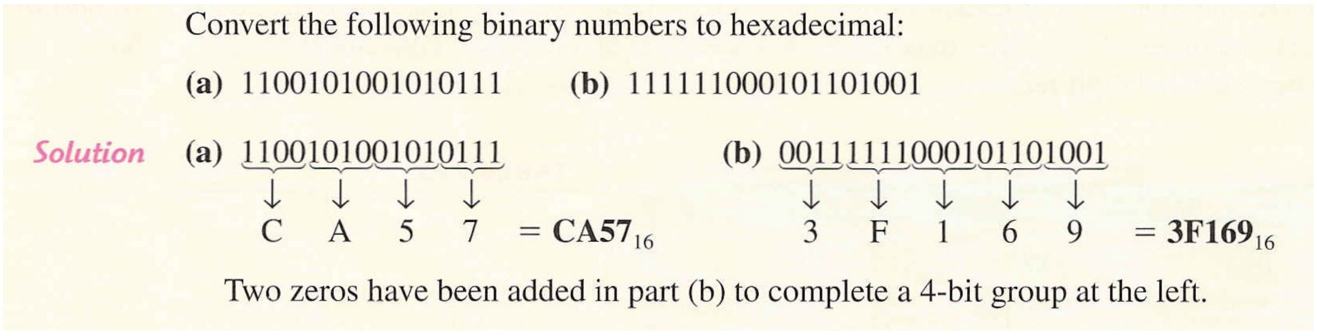
Solution

(a) $\begin{matrix}\underbrace{110}&\underbrace{101}\\ \downarrow&\downarrow\\ 6&5\end{matrix} = \mathbf{65_8}$ **(b)** $\begin{matrix}\underbrace{101}&\underbrace{111}&\underbrace{001}\\ \downarrow&\downarrow&\downarrow\\ 5&7&1\end{matrix} = \mathbf{571_8}$

(c) $\begin{matrix}\underbrace{100}&\underbrace{110}&\underbrace{011}&\underbrace{010}\\ \downarrow&\downarrow&\downarrow&\downarrow\\ 4&6&3&2\end{matrix} = \mathbf{4632_8}$ **(d)** $\begin{matrix}\underbrace{011}&\underbrace{010}&\underbrace{000}&\underbrace{100}\\ \downarrow&\downarrow&\downarrow&\downarrow\\ 3&2&0&4\end{matrix} = \mathbf{3204_8}$


Binary-to-Hexadecimal Conversion

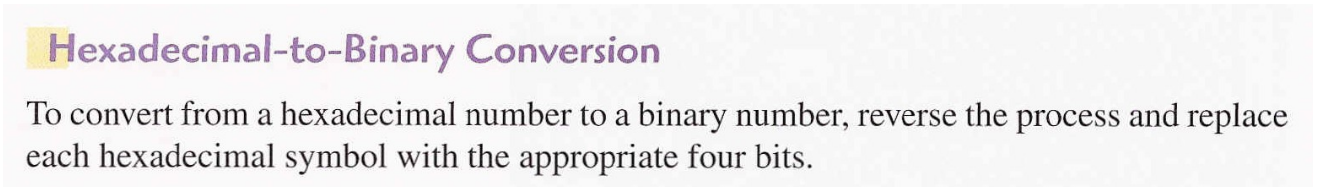
Converting a binary number to hexadecimal is a straightforward procedure. Simply break the binary number into 4-bit groups, starting at the right-most bit and replace each 4-bit group with the equivalent hexadecimal symbol.


Convert the following binary numbers to hexadecimal:

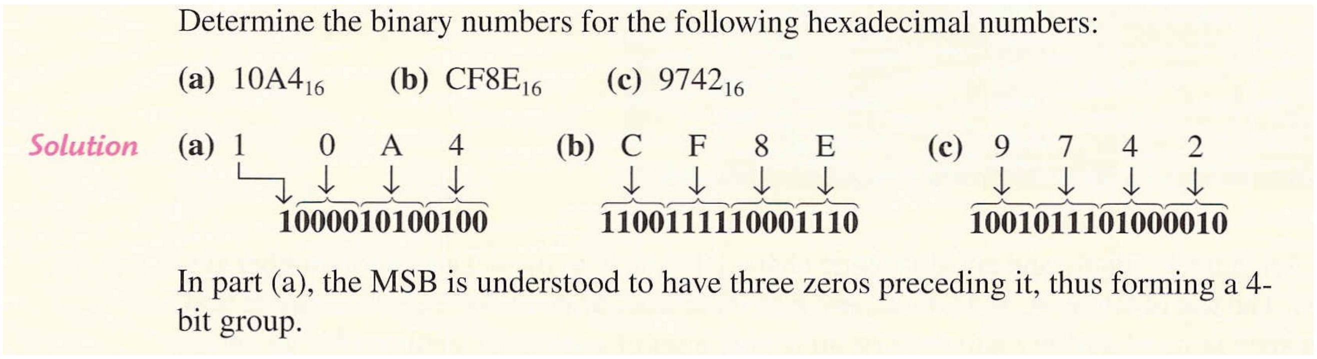
(a) 1100101001010111 **(b)** 111111000101101001

Solution **(a)** $\underbrace{1100}_{\downarrow \atop C}\underbrace{1010}_{\downarrow \atop A}\underbrace{0101}_{\downarrow \atop 5}\underbrace{0111}_{\downarrow \atop 7} = \mathbf{CA57}_{16}$ **(b)** $\underbrace{0011}_{\downarrow \atop 3}\underbrace{1111}_{\downarrow \atop F}\underbrace{0001}_{\downarrow \atop 1}\underbrace{0110}_{\downarrow \atop 6}\underbrace{1001}_{\downarrow \atop 9} = \mathbf{3F169}_{16}$

Two zeros have been added in part (b) to complete a 4-bit group at the left.


Hexadecimal-to-Binary Conversion

To convert from a hexadecimal number to a binary number, reverse the process and replace each hexadecimal symbol with the appropriate four bits.


Determine the binary numbers for the following hexadecimal numbers:

(a) $10A4_{16}$ **(b)** $CF8E_{16}$ **(c)** 9742_{16}

Solution **(a)** $\overbrace{1}^{1 \atop \downarrow}\overbrace{0000}^{0 \atop \downarrow}\overbrace{1010}^{A \atop \downarrow}\overbrace{0100}^{4 \atop \downarrow}$ **(b)** $\overbrace{1100}^{C \atop \downarrow}\overbrace{1111}^{F \atop \downarrow}\overbrace{1000}^{8 \atop \downarrow}\overbrace{1110}^{E \atop \downarrow}$ **(c)** $\overbrace{1001}^{9 \atop \downarrow}\overbrace{0111}^{7 \atop \downarrow}\overbrace{0100}^{4 \atop \downarrow}\overbrace{0010}^{2 \atop \downarrow}$

1000010100100 **1100111110001110** **1001011101000010**

In part (a), the MSB is understood to have three zeros preceding it, thus forming a 4-bit group.

1.2.11 Octal to Hexadecimal Conversion

Procedure

1. Convert each octal digit to 3-bit binary form

2. Combine all the 3-bits binary numbers.

3. Divide the binary numbers into the 4-bit binary form by starting the first number from the right bit to the first number from the left bit.

4. Finally, convert these 4-bit blocks into their respective hexadecimal symbols.

Example

Octal Number	2	3	2	7
Binary Coded Value	010	011	010	111

Combining three binary blocks, we have: 010011010111.

Dividing this into 4-bit binary blocks and converting theme to hexadecimal symbols:-

0100	1101	0111

4	**D**	**7**

Therefore $(2327)_8 = (4D7)_{16}$

1.2.12 Hexadecimal to Octal Conversion

Procedure

Same as Octal to hexadecimal except that each hexadecimal digit is converted into 4-bit binary form then after grouping of all 4-bit binary blocks, it is converted into 3-bit binary form.

Finally these 3-bit binary forms are converted into octal symbols.

Example

Hexadecimal Number	2	B	6
Binary Coded Value	0010	1011	0110

Combining all the 4-bit binary blocks 001010110110

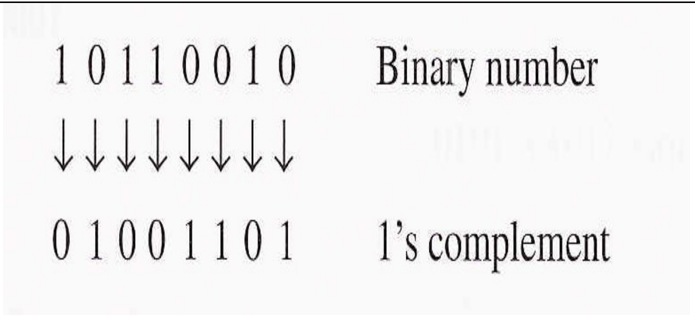
Dividing this into 3-bit binary blocks we have:

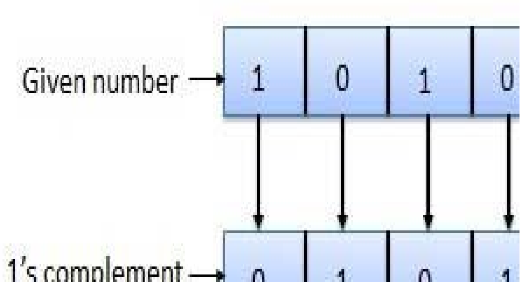
001	010	110	110
1	**2**	**6**	**6**

Therefore $(2B6)_{16} = (1266)_8$

7. **Explain the concepts of 1's complement and 2's complement.**

The **1's complement** of a binary number is found by changing all 1's to 0's and all 0's to 1's, as illustrated below

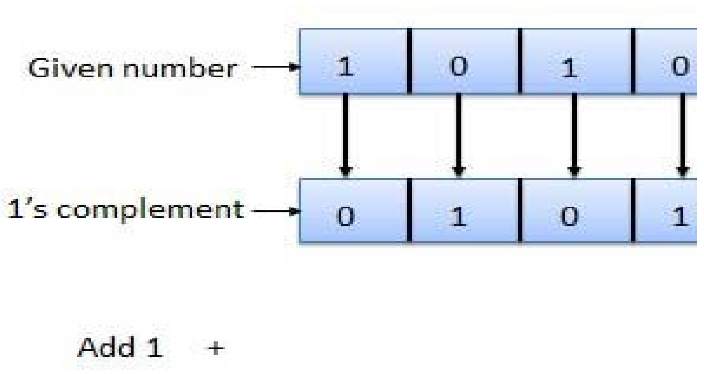


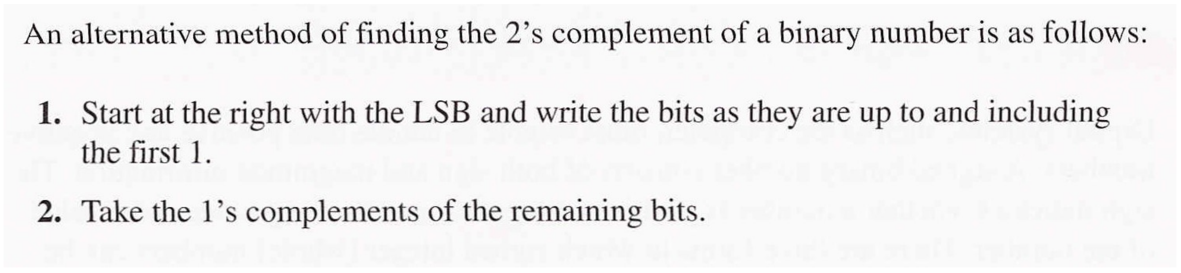


The **2's complement** of binary number is obtained by adding 1 to the Least Significant Bit (**LSB**) of 1's complement of the number.

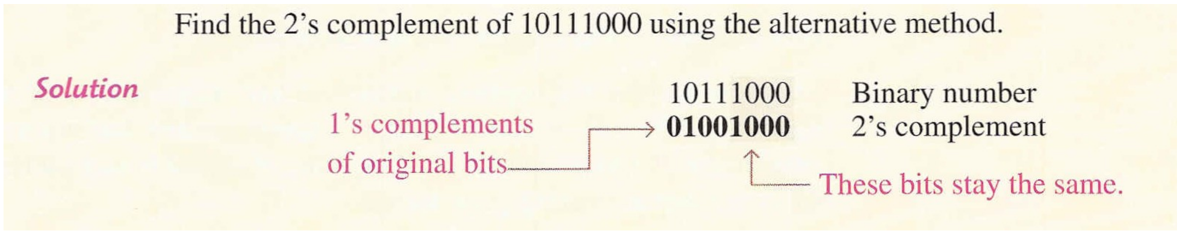
2's complement = (1's complement) + 1

Example of 2's Complement is as follows.




An alternative method of finding the 2's complement of a binary number is as follows:

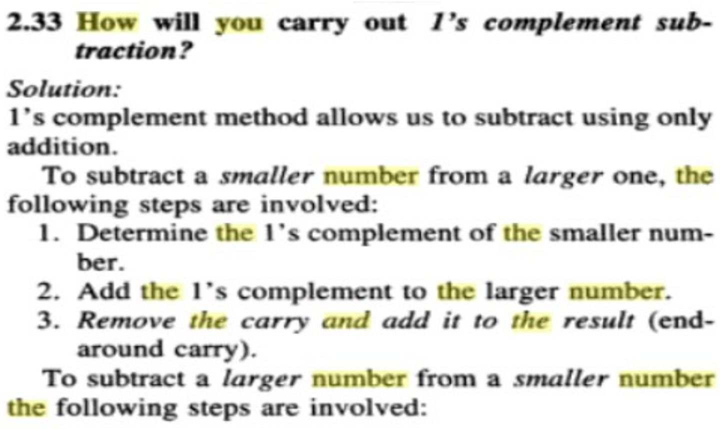
1. Start at the right with the LSB and write the bits as they are up to and including the first 1.
2. Take the 1's complements of the remaining bits.


Find the 2's complement of 10111000 using the alternative method.

Solution

	10111000	Binary number
1's complements of original bits →	**01001000**	2's complement
	↑ These bits stay the same.	

8. **How do you subtract a number in 1'compement?**


2.33 How will you carry out *1's complement subtraction?*

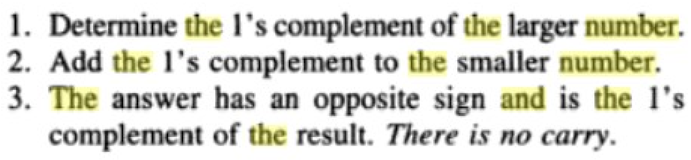
Solution:

1's complement method allows us to subtract using only addition.

To subtract a *smaller* number from a *larger* one, the following steps are involved:

1. Determine the 1's complement of the smaller number.
2. Add the 1's complement to the larger number.
3. *Remove the carry and add it to the result* (end-around carry).

To subtract a *larger* number from a *smaller* number the following steps are involved:


1. Determine the 1's complement of the larger number.
2. Add the 1's complement to the smaller number.
3. The answer has an opposite sign and is the 1's complement of the result. *There is no carry.*

Subtract 10011_2 from 11001_2 using the 1's complement method.

Solution:

$$\begin{array}{rl}
\text{1's complement} & \\
\text{subtraction} & \\
11001_2 & \\
+01100_2 & \text{1's complement} \\
\hline
\textcircled{1}00101_2 & \\
\hookrightarrow +1 & \text{Add end-around carry} \\
\hline
00110_2 & \\
\hline
\end{array}$$

Subtract 1101_2 from 1001_2 using the 1's complement method.

Solution:

$$\begin{array}{rl}
\text{1's complement} & \\
\text{subtraction} & \\
1001_2 & \\
+0010_2 & \text{1's complement} \\
\hline
1011_2 & \text{Answer in 1's complement form and opposite in sign} \\
\downarrow & \\
\hline
-0100 & \text{Final answer.} \\
\hline
\end{array}$$

9. **How do you subtract a number in 2'compement?**

2.39 How will you carry out *2's complement subtraction?*

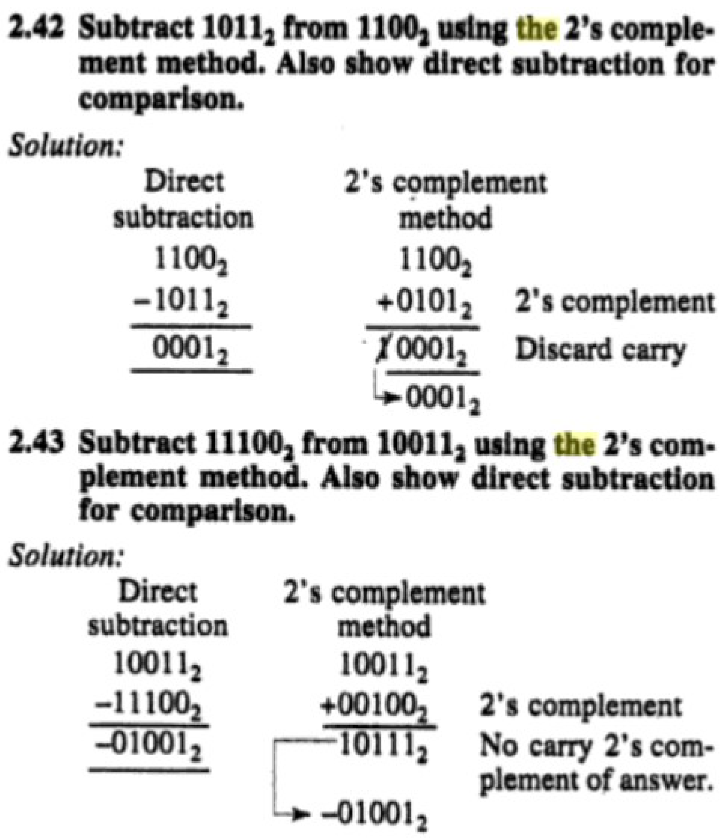
Solution:

To subtract *a smaller number from a larger number*, the following steps are involved:

1. Determine the 2's complement of the smaller number.
2. Add the 2's complement to the larger number.
3. Discard the carry (*there is always a carry in this case*).

To subtract *a larger number from a smaller number* the following steps are involved:

1. Determine the 2's complement of the larger number.
2. Add the 2's complement to the smaller number.
3. *There is no carry. The result is in 2's complement form and is negative.*
4. To get the answer in true form, take the 2's complement and change the sign.



2.42 Subtract 1011_2 from 1100_2 using the 2's complement method. Also show direct subtraction for comparison.

Solution:

Direct subtraction	2's complement method	
1100_2	1100_2	
-1011_2	$+0101_2$	2's complement
0001_2	$\cancel{1}0001_2$	Discard carry
	$\rightarrow 0001_2$	

2.43 Subtract 11100_2 from 10011_2 using the 2's complement method. Also show direct subtraction for comparison.

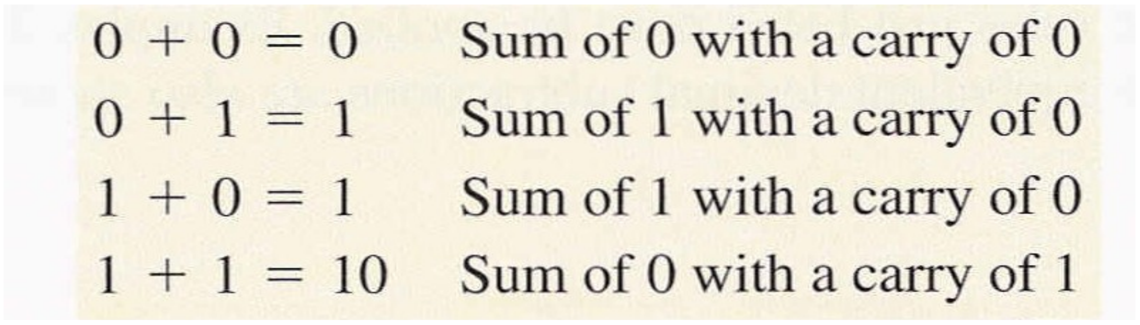
Solution:

Direct subtraction	2's complement method	
10011_2	10011_2	
-11100_2	$+00100_2$	2's complement
-01001_2	10111_2	No carry 2's complement of answer.
	$\rightarrow -01001_2$	

1.3 Binary Arithmetic

1.3.1 Binary addition

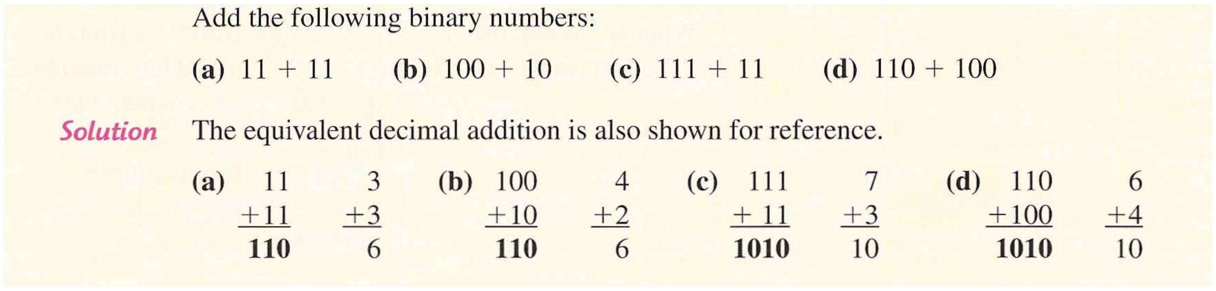
The four basic rules for adding binary digits (bits) are as follows



$0 + 0 = 0$	Sum of 0 with a carry of 0
$0 + 1 = 1$	Sum of 1 with a carry of 0
$1 + 0 = 1$	Sum of 1 with a carry of 0
$1 + 1 = 10$	Sum of 0 with a carry of 1

For example,

```
00011010 + 00001100 = 00100110      1 1                         carries
                                     0  0  0  1  1  0  1  0  =  26(base 10)
                                   + 0  0  0  0  1  1  0  0  =  12(base 10)
                                   ---------------------------
                                     0  0  1  0  0  1  1  0  =  38(base 10)
```

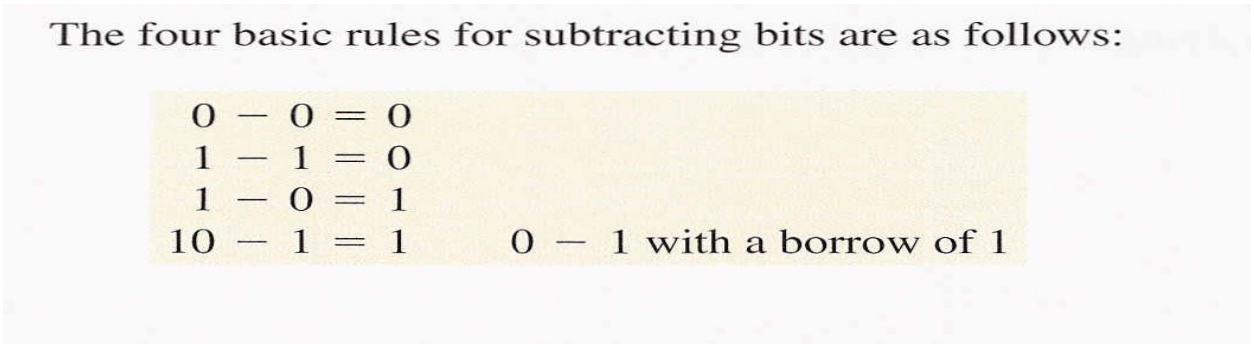

Add the following binary numbers:

(a) 11 + 11 **(b)** 100 + 10 **(c)** 111 + 11 **(d)** 110 + 100

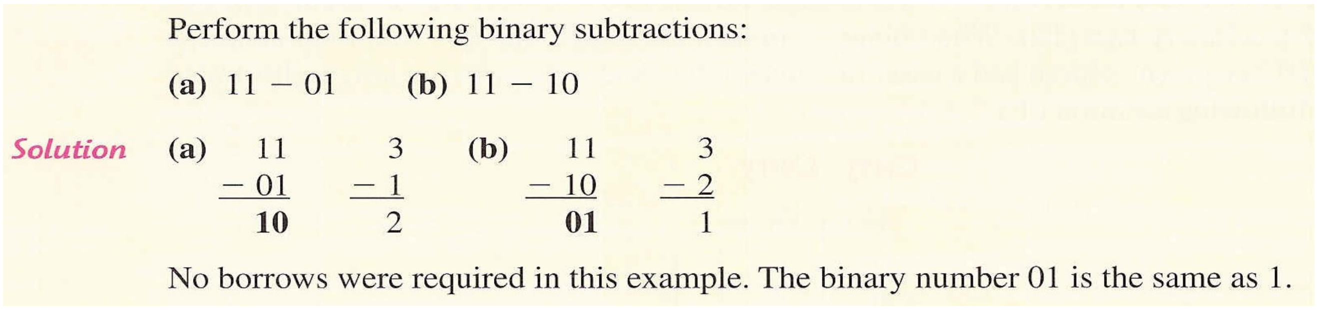
Solution The equivalent decimal addition is also shown for reference.

(a)	11	3	**(b)**	100	4	**(c)**	111	7	**(d)**	110	6
	+11	+3		+10	+2		+ 11	+3		+100	+4
	110	6		**110**	6		**1010**	10		**1010**	10

1.3.2 Binary subtraction


The four basic rules for subtracting bits are as follows:

$$0 - 0 = 0$$
$$1 - 1 = 0$$
$$1 - 0 = 1$$
$$10 - 1 = 1 \qquad 0 - 1 \text{ with a borrow of } 1$$


Perform the following binary subtractions:

(a) 11 − 01 **(b)** 11 − 10

Solution

(a)	11	3	**(b)**	11	3
	− 01	− 1		− 10	− 2
	10	2		**01**	1

No borrows were required in this example. The binary number 01 is the same as 1.

For example,

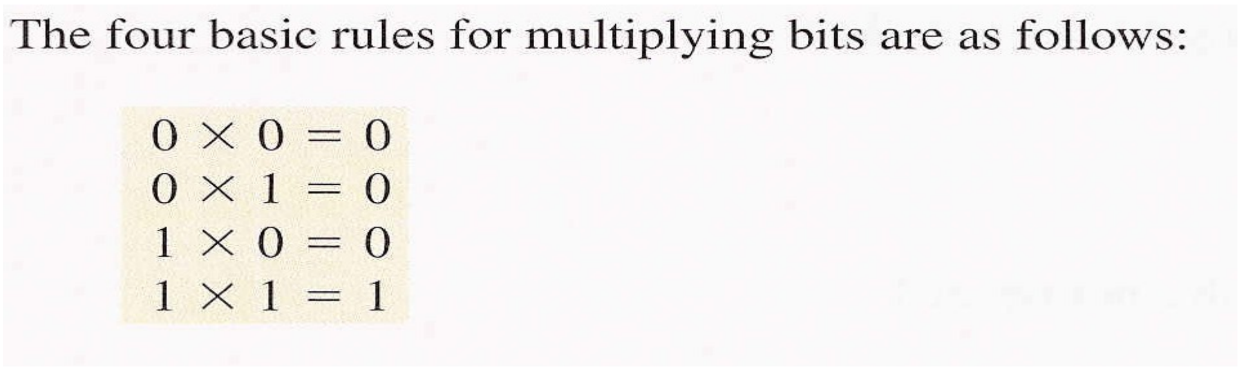
`00100101 - 00010001 = 00010100`

	0									*borrows*
	0	0	~~1~~	^{1}0	0	1	0	1	=	$37_{(\text{base } 10)}$
−	0	0	0	1	0	0	0	1	=	$17_{(\text{base } 10)}$
	0	0	0	1	0	1	0	0	=	$20_{(\text{base } 10)}$

`00110011 - 00010110 = 00011101`

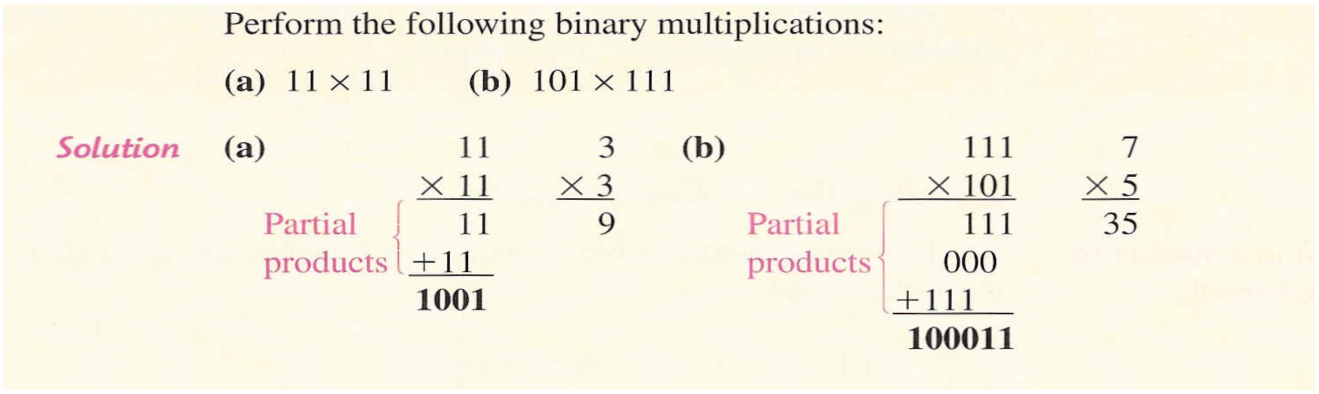
	0 ^{1}0 1									*borrows*
	0	0	~~1~~	~~1~~	~~0~~	^{1}0	1	1	=	$51_{(\text{base } 10)}$
−	0	0	0	1	0	1	1	0	=	$22_{(\text{base } 10)}$
	0	0	0	1	1	1	0	1	=	$29_{(\text{base } 10)}$

1.3.3 Binary multiplication


The four basic rules for multiplying bits are as follows:

$$\begin{aligned} 0 \times 0 &= 0 \\ 0 \times 1 &= 0 \\ 1 \times 0 &= 0 \\ 1 \times 1 &= 1 \end{aligned}$$

Multiplication is performed wirth binary numbers in the same manner as with decimal numbers.it involves forming partial products, shifting each succesive partial product left one place, and then adding all the partial products.


Perform the following binary multiplications:

(a) 11 × 11 **(b)** 101 × 111

Solution

$$\textbf{(a)}\quad \begin{array}{r} 11 \\ \underline{\times\, 11} \\ 11 \\ \underline{+11} \\ \textbf{1001} \end{array} \quad \begin{array}{r} 3 \\ \underline{\times\, 3} \\ 9 \end{array} \qquad \textbf{(b)}\quad \begin{array}{r} 111 \\ \underline{\times\, 101} \\ 111 \\ 000 \\ \underline{+111} \\ \textbf{100011} \end{array} \quad \begin{array}{r} 7 \\ \underline{\times\, 5} \\ 35 \end{array}$$

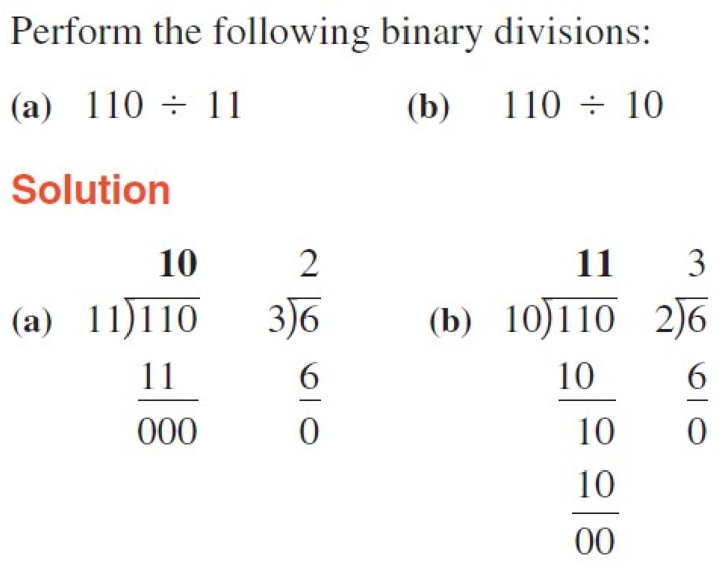
Partial products

Partial products

1.3.4 Binary Division

Dision in binary follows the same procedure as division in decimal.

Example :


Perform the following binary divisions:

(a) 110 ÷ 11 **(b)** 110 ÷ 10

Solution

$$\textbf{(a)}\quad \begin{array}{r} \textbf{10} \\ 11\overline{)110} \\ \underline{11} \\ 000 \end{array} \quad \begin{array}{r} 2 \\ 3\overline{)6} \\ \underline{6} \\ 0 \end{array} \qquad \textbf{(b)}\quad \begin{array}{r} \textbf{11} \\ 10\overline{)110} \\ \underline{10} \\ 10 \\ \underline{10} \\ 00 \end{array} \quad \begin{array}{r} 3 \\ 2\overline{)6} \\ \underline{6} \\ 0 \end{array}$$

10. **What is BCD and Explain?**

- ✓ Binary coded decimal (BCD) is a weighted code that is commonly used in digital systems such as digital clocks, digital thermometers, digital metes, and other devices with seven-segment displays.
- ✓ The decimal digits 0 through 9 are represented by a 4 – bit binary code.
- ✓ The weights used are 8-4-2-1.
- ✓ The 8421 combination indicates a binary weights of the four bits (nibbles)

$$\begin{matrix} 8 & 4 & 2 & 1 \\ (2^3 & 2^2 & 2^1 & 2^0) \end{matrix}$$

- ✓ With BCD, each digit of a number is converted into its binary equivalent rather than converting the entire decimal number to its binary form.
- ✓ BCD system was developed by the IBM (International Business Machines) corporation.
- ✓ BCD represents each decimal digit with a 4-bit code. The valid combinations of bits and their respective values are shown in table.

Decimal/BCD conversion.

Decimal Digit	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

- ✓ Notice that the codes 1010 through 1111 are not used in BCD.

Convert each of the following decimal numbers to BCD:

(a) 35 **(b)** 98 **(c)** 170 **(d)** 2469

Solution

(a) $$\overbrace{\mathbf{0011}}^{\overset{3}{\downarrow}}\overbrace{\mathbf{0101}}^{\overset{5}{\downarrow}}$$

(b) $$\overbrace{\mathbf{1001}}^{\overset{9}{\downarrow}}\overbrace{\mathbf{1000}}^{\overset{8}{\downarrow}}$$

(c) $$\overbrace{\mathbf{0001}}^{\overset{1}{\downarrow}}\overbrace{\mathbf{0111}}^{\overset{7}{\downarrow}}\overbrace{\mathbf{0000}}^{\overset{0}{\downarrow}}$$

(d) $$\overbrace{\mathbf{0010}}^{\overset{2}{\downarrow}}\overbrace{\mathbf{0100}}^{\overset{4}{\downarrow}}\overbrace{\mathbf{0110}}^{\overset{6}{\downarrow}}\overbrace{\mathbf{1001}}^{\overset{9}{\downarrow}}$$

Convert each of the following BCD codes to decimal:

(a) 10000110 **(b)** 001101010001 **(c)** 1001010001110000

Solution

(a) $$\underbrace{1000}_{\underset{\mathbf{8}}{\downarrow}}\underbrace{0110}_{\underset{\mathbf{6}}{\downarrow}}$$

(b) $$\underbrace{0011}_{\underset{\mathbf{3}}{\downarrow}}\underbrace{0101}_{\underset{\mathbf{5}}{\downarrow}}\underbrace{0001}_{\underset{\mathbf{1}}{\downarrow}}$$

(c) $$\underbrace{1001}_{\underset{\mathbf{9}}{\downarrow}}\underbrace{0100}_{\underset{\mathbf{4}}{\downarrow}}\underbrace{0111}_{\underset{\mathbf{7}}{\downarrow}}\underbrace{0000}_{\underset{\mathbf{0}}{\downarrow}}$$

11. **Explain the rules for BCD addition with the help of suitable examples.**

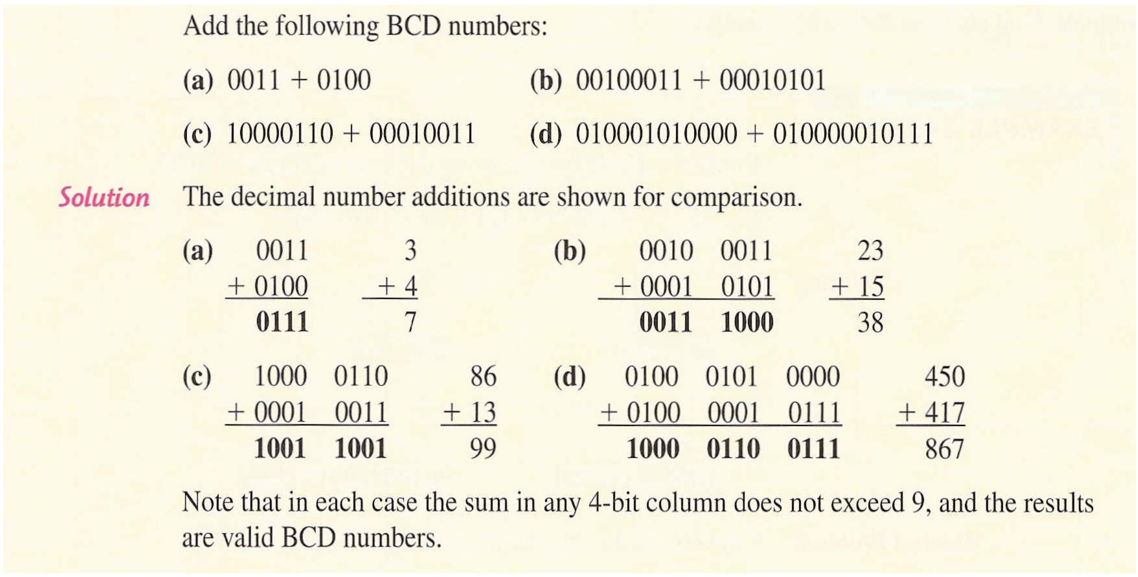
Step1: Add the two BCD numbers, using the rules for binary addition.

Step 2: If a 4-bit sum is equal to or less than **9**, it is a valid BCD number.

Step 3: If a 4-bit sum is greater than 9, or if a carry out of the 4-bit group is generated, it is an invalid result.

Add **6(0110)** to the 4-bit sum in order to skip the six invalid states and return the code to 8421.

If a carry results when 6 is added, simply add the carry to the next 4-bit group.



Add the following BCD numbers:

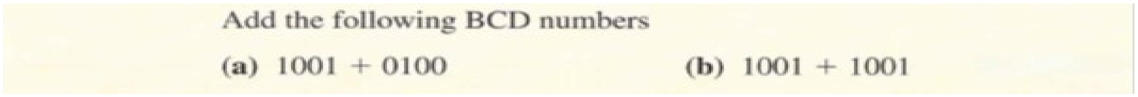
(a) 0011 + 0100 **(b)** 00100011 + 00010101

(c) 10000110 + 00010011 **(d)** 010001010000 + 010000010111

Solution The decimal number additions are shown for comparison.

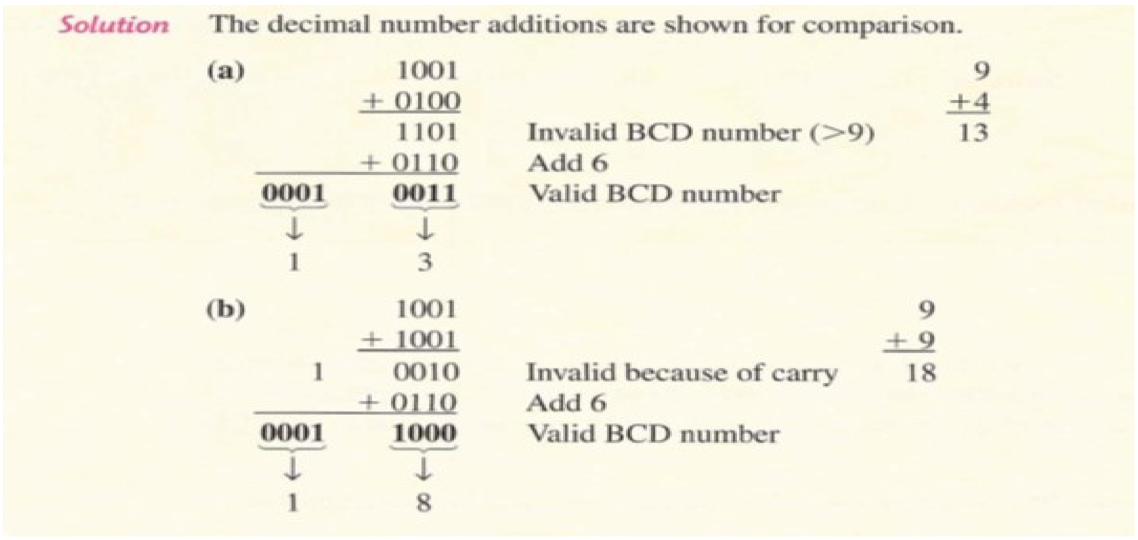
```
(a)   0011       3        (b)    0010  0011      23
    + 0100     + 4             + 0001  0101    + 15
      0111       7               0011  1000      38

(c)   1000  0110     86   (d)   0100  0101  0000      450
    + 0001  0011   + 13       + 0100  0001  0111    + 417
      1001  1001     99         1000  0110  0111      867
```

Note that in each case the sum in any 4-bit column does not exceed 9, and the results are valid BCD numbers.



Add the following BCD numbers

(a) 1001 + 0100 **(b)** 1001 + 1001



Solution The decimal number additions are shown for comparison.

```
(a)            1001                                   9
             + 0100                                  +4
               1101    Invalid BCD number (>9)       13
             + 0110    Add 6
       0001    0011    Valid BCD number
        ↓       ↓
        1       3

(b)            1001                                   9
             + 1001                                 + 9
          1    0010    Invalid because of carry      18
             + 0110    Add 6
       0001    1000    Valid BCD number
        ↓       ↓
        1       8
```

12. **What is ASCII code? Mention its applications.**

- ✓ ASCII stands for American Standard Code for Information Interchange.
- ✓ ASCII is a universally accepted alphanumeric code used in most of computers and other electronic equipment.
- ✓ This code uses 7 bit binary pattern.
- ✓ Therefore 128 (2^7) character codes (from 0 to 127) and symbols represented by a 7-bit binary code.

  Ex: 'A' = 1000001 (65) 'a' = 1100001 (97)

Applications:

1. The ASCII can be used to encode both the uppercase and lowercase characters of the alphabet (52 symbols) and some special symbols in addition to the 10 decimal digits.
2. It is used extensively for printers and terminals that interface with small computer systems.
3. A new version of ASCII code called Extended ASCII or ASCII-8. It uses 8-bit code.
4. ASCII is the standard alpha numeric code for keyboard data encoding, data transfer and other computer interface applications with printers and video terminals.
5. ASCII is mainly used in microcomputers.

13. Compare ASCII and EBCDIC.

ASCII	**EBCDIC**
American Standard Code For Information Interchange	Extended Binary Coded Decimal Interchange Code
It is an alphanumeric code which uses 7 bits for coding and the 8th bit for parity check	It is an alphanumeric code which uses 8 bits for coding, no parity check bit.
ASCII uses a linear ordering of letters.	EBCDIC doesn't use a linear ordering of letters.
Different versions of ASCII are compatible	EBCDIC's different versions are not compatible
ASCII is compatible with modern encodings.	EBCDIC isn't compatible with modern encodings
ASCII contained less characters than EBCDIC	EBCDIC contained more characters than ASCII
ASCII is mainly used in microcomputers.	EBCDIC is used by IBM and mainframes.

14. Explain the terms GRAY CODE, and EXCESS-3 CODE.

Gray code

- ✓ The Gay code is an unweighted 4 –bit code; that is there are no specific weights are assigned to the bit positions.
- ✓ The important feature of the Gray code is a single bit change from one code word to the next in sequence.
- ✓ Gray code is not an arithmetic code.
- ✓ Gray code not suited for arithmetic operations, but useful for input output devices, analog to digital converters, shaft position encoders etc.
- ✓ Gray code is also termed as " Unit Distance Code" or "Reflected Code"
- ✓ These codes are also called as **cyclic code**

  For Example;

  Gray code for decimal number 5 is 0 1 1 1 and

  Gray code for decimal number 6 is 0 1 0 1 .

  These two codes only differ by one bit position.
- ✓ Below table is a listing of the 4-bit Gray code for decimal numbers 0 through 15.

Four-bit Gray code.

Decimal	Binary	Gray Code	Decimal	Binary	Gray Code
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

Binary-to-Gray Code Conversion

The following rules explain how to convert from a binary number to a Gray code word:

1. The most significant bit (left-most) in the Gray code is the same as the corresponding MSB in the binary number.

2. Going from left to right, add each adjacent pair of binary code bits to get the next Gray code bit. Discard carries.

For example, the conversion of the binary number 10110 to Gray code is as follows:

$$\begin{tikzcd}
1 \arrow[r, "+"] \arrow[d] & 0 \arrow[r, "+"] \arrow[d] & 1 \arrow[r, "+"] \arrow[d] & 1 \arrow[r, "+"] \arrow[d] & 0 \arrow[d] & \text{Binary} \\
1 & 1 & 1 & 0 & 1 & \text{Gray}
\end{tikzcd}$$

The Gray code is 11101.

Gray-to-Binary Code Conversion

To convert from Gray code to binary, use a similar method; however, there are some differences. The following rules apply:

1. The most significant bit (left-most) in the binary code is the same as the corresponding bit in the Gray code.
2. Add each binary code bit generated to the Gray code bit in the next adjacent position. Discard carries.

For example, the conversion of the Gray code word 11011 to binary is as follows:

$$\begin{tikzcd}
1 \arrow[d] & 1 \arrow[d] & 0 \arrow[d] & 1 \arrow[d] & 1 \arrow[d] & \text{Gray} \\
1 \arrow[ur, "+"] & 0 \arrow[ur, "+"] & 0 \arrow[ur, "+"] & 1 \arrow[ur, "+"] & 0 & \text{Binary}
\end{tikzcd}$$

The binary number is 10010.

(a) Convert the binary number 11000110 to Gray code.

(b) Convert the Gray code 10101111 to binary.

Solution

(a) Binary to Gray code:

$$\begin{tikzcd}
1 \arrow[r, "+"] \arrow[d] & 1 \arrow[r, "+"] \arrow[d] & 0 \arrow[r, "+"] \arrow[d] & 0 \arrow[r, "+"] \arrow[d] & 0 \arrow[r, "+"] \arrow[d] & 1 \arrow[r, "+"] \arrow[d] & 1 \arrow[r, "+"] \arrow[d] & 0 \arrow[d] \\
\mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1}
\end{tikzcd}$$

(b) Gray code to binary:

$$\begin{tikzcd}
1 \arrow[d] & 0 \arrow[d] & 1 \arrow[d] & 0 \arrow[d] & 1 \arrow[d] & 1 \arrow[d] & 1 \arrow[d] & 1 \arrow[d] \\
\mathbf{1} \arrow[ur, "+"] & \mathbf{1} \arrow[ur, "+"] & \mathbf{0} \arrow[ur, "+"] & \mathbf{0} \arrow[ur, "+"] & \mathbf{1} \arrow[ur, "+"] & \mathbf{0} \arrow[ur, "+"] & \mathbf{1} \arrow[ur, "+"] & \mathbf{0}
\end{tikzcd}$$

EXCESS-3 CODE

- ✓ The Excess-3 is a digital code that is derived by adding 3 to each decimal digit and then converting the result to 4-bit BCD number.
- ✓ Excess-3 is an unweighted code.
- ✓ The key feature of Excess -3 codes is that it is self-complimenting code.
- ✓ For instance, the Excess-3 code for the- decimal 2 is 2 + 3 =5 as Excess-3 code

Decimal digit	BCD Code	Excess-3 Code
0	0000	0 011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

- ✓ Ten of a possible 16 code combinations are used in Excess-3 code. The six invalid combinations are 0000, 0001, 0010, 1101, 1110 and 1111.
- ✓ Excess-3 code Used in some arithmetic circuits for performing subtraction operations.

Decimal to Excess-3 code Conversion

The following steps are used to convert a decimal number to XS-3 code.

1) Add 3 to the each digit in the given decimal number.
2) Convert each sum to its equivalent binary code.

1) Convert 12 to Excess- 3 Code

Add 3 to each decimal digit

1		2
3		3
4		5

Convert into BCD Form

0100	**0101**

Excess -3 Code for 12 = 0100 0101

2) **Convert 29 to Excess- 3 Code**

Add 3 to each decimal digit

2		9
3		3
5		12

Convert into BCD Form

0101	**1100**

Excess -3 Code for 29 = 0101 1100

3) **Convert 159 to Excess- 3 Code**

Add 3 to each decimal digit

1		5		9
3		3		3
4		8		12

Convert into BCD Form

0100	**1000**	**1100**

Excess -3 Code for 159 = 0100 1000 1100

15. Convert the decimal number 73 and 4236 in to Excess -3 code.

a) **Convert 73 to Excess- 3 Code**

Add 3 to each decimal digit

7		3
3		3
10		6

Convert into BCD Form

1010	**0110**

Excess -3 Code for 73 = 1010 0110

b) Convert 4236 to Excess- 3 Code

Add 3 to each decimal digit

4		2		3		6
3		3		3		3
7		5		6		9

Convert into BCD Form

0111	**0101**	**0110**	**1001**

Excess -3 Code for 4236 = 0111 0101 0110 1001

Unicode

- ✓ Unicode provides the ability to encode all of the characters used for the written languages of the world by assigning each character a unique numeric value and name utilizing the universal character set (UCS).
- ✓ It is applicable in computer applications dealing with multilingual text, mathematical symbols, or other technical characters.
- ✓ Unicode has a wide array of characters, and their various encoding forms are used in many environments. While ASCII basically uses 7-bit codes, Unicode uses relatively abstract "code points"—non-negative integer numbers—that map sequences of one or more bytes, using different encoding forms and schemes.
- ✓ To permit compatibility, Unicode assigns the first 128 code points to the same characters as ASCII. One can, therefore, think of ASCII as a 7-bit encoding scheme for a very small subset of Unicode and of the UCS.
- ✓ Unicode consists of about 100,000 characters, a set of code charts for visual reference, an encoding methodology and set of standard character encodings, and an enumeration of character properties such as uppercase and lowercase.
- ✓ It also consists of a number of related items, such as character properties, rules for text normalization, decomposition, collation, rendering, and bidirectional display order (for the correct display of text containing both right-to-left scripts, such as Arabic or Hebrew, and left-to-right scripts).

16. With an example explain the conversion of octal number to hexadecimal number and vice versa.

Octal to Hexadecimal Conversion

Procedure

1. Convert each octal digit to 3-bit binary form

2. Combine all the 3-bits binary numbers.

3. Divide the binary numbers into the 4-bit binary form by starting the first number from the right bit to the first number from the left bit.

4. Finally, convert these 4-bit blocks into their respective hexadecimal symbols.

Example

Octal Number	2	3	2	7
Binary Coded Value	010	011	010	111

Combining three binary blocks, we have: 010011010111.

Dividing this into 4-bit binary blocks and converting theme to hexadecimal symbols:-

0100	1101	0111
4	**D**	**7**

Therefore $(2327)_8 = (4D7)_{16}$

Convert 536 from octal to hexadecimal number

Convert 536(octal) into its binary equivalent we get

$(536)_8 = (101)\ (011)\ (110)$

$= (101011110)_2$

Now forming the group of 4 binary bits to obtain its hexadecimal equivalent,

$(101011110)_2 = (0001)\ (0101)\ (1110)$

$= (1\ 5\ E)_{16}$

Procedure

Same as Octal to hexadecimal except that each hexadecimal digit is converted into 4-bit binary form then after grouping of all 4-bit binary blocks, it is converted into 3-bit binary form.

Finally these 3-bit binary forms are converted into octal symbols.

Example

Hexadecimal Number	2	B	6
Binary Coded Value	0010	1011	0110

Combining all the 4-bit binary blocks 001010110110

Dividing this into 3-bit binary blocks we have:

001	010	110	110
1	**2**	**6**	**6**

Therefore $(2B6)_{16} = (1266)_8$

Convert 1 5 E from hexadecimal to octal number

$(1\ 5\ E)_{16} = (0001)(0101)(1110)$

$= (000101011110)_2$

Now forming the group of 3 binary bits to obtain its octal equivalent,

$(000101011110)_2 = (000)(101)(011)(110)$

$= (0536)_8$

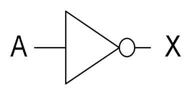
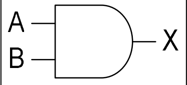
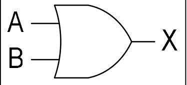
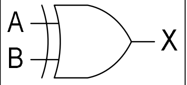
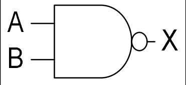
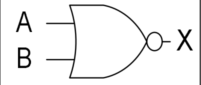
Logic gates

- ✓ A **logic gate** is a building block of a digital circuit that operates on one or more input signals to obtain the standard output signals.
- ✓ A **logic gate** is an elementary building block of a digital circuit. A simple logic gate has two inputs and one output.
- ✓ At any given moment, every terminal is in one of the two binary conditions LOW (0) or HIGH (1), represented by different voltage levels.
- ✓ In most logic gates, the LOW state is approximately zero volts (0 V), while the HIGH state is approximately five volts positive (+5 V).

Boolean Algebra	Boolean Logic	Voltage State
Logic “1”	True (T)	High (H)
Logic “0”	False (F)	Low (L)

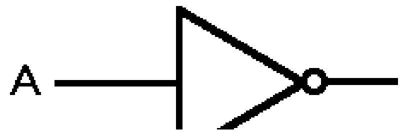
- ✓ A **Truth table** is a table showing the inputs and corresponding output(s) of a logic circuit
- ✓ There are seven basic logic gates: AND, OR, XOR, NOT, NAND, NOR, and XNOR.
- ✓ The ***NOT*** gate has only ***one input*** and ***one output*** and is the simplest gate. The remaining gates can all have ***one or many inputs***, but all have only ***one output***.

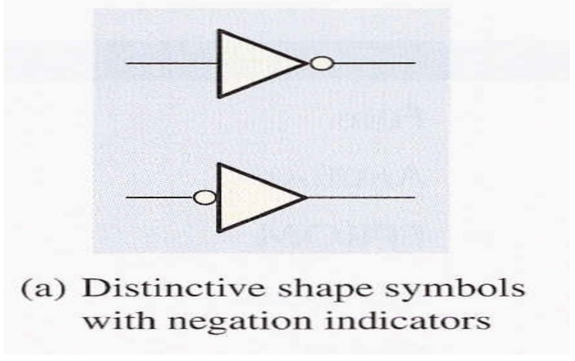
17. Explain Logic gates – NOT, OR, AND, NAND, NOR, XOR, and EX-NOR

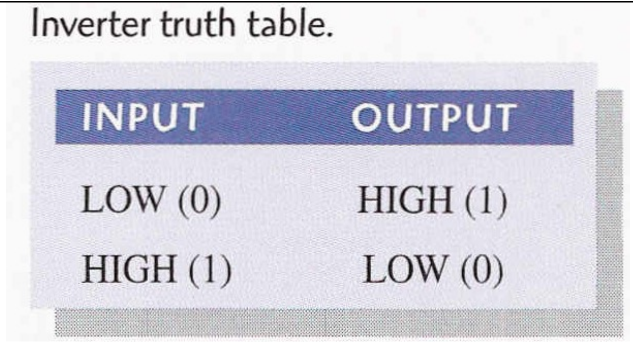
(i)	NOT	AND	OR	EX-OR	NAND	NOR
(ii)						
(iii)	$X = \overline{A}$	$X = A\,B$	$X = A + B$	$X = A \oplus B$	$X = \overline{A\,B}$	$X = \overline{A + B}$
(iv)	A | X 0 | 1 1 | 0	A B | X 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1	A B | X 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1	A B | X 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0	A B | X 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0	A B | X 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0

18. Write the truth table and logical symbol of NOT gate.

- ✓ The **NOT gate** (or **inverter)** performs a logic function called **inversion** or **complementation**.
- ✓ The **Inverter** changes one logic level (HIGH/LOW) to the opposite logic level (LOW/HIGH). In terms of bits, it changes a 1 to 0 and a 0 to 1
- ✓ The output is 1 (HIGH) if the input is 0 (LOW). The output is 0 (LOW) when input is 1(HIGH).




(a) Distinctive shape symbols with negation indicators

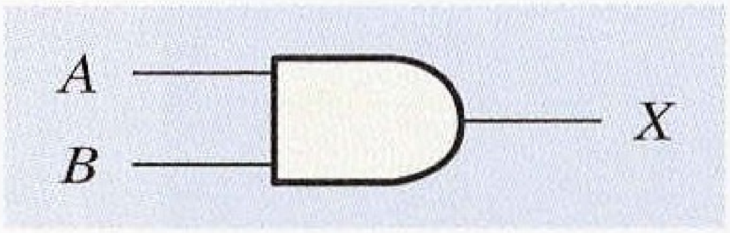

Inverter truth table.

INPUT	OUTPUT
LOW (0)	HIGH (1)
HIGH (1)	LOW (0)

- ✓ The **negation indicator** is a "bubble" (o) that indicates **inversion or complementation** when it appears on the input or output of any logic element as shown in above figure.

19. Write the truth table and logical symbol of AND gate.

- ✓ The AND gate is composed of two or more inputs and a single output and performs logical multiplication.
- ✓ **An AND gate** produces a HIGH output only when all of the inputs are HIGH. When any of the inputs is LOW, the output is LOW.
- ✓ The standard symbol and truth table for AND gate is shown below.



(a) Distinctive shape

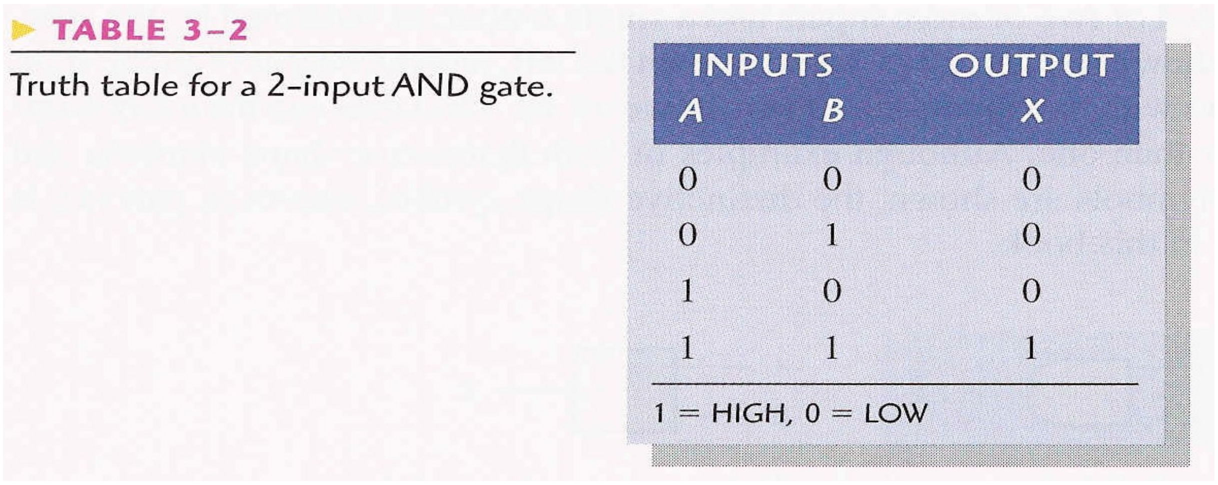


TABLE 3–2

Truth table for a 2-input AND gate.

INPUTS		OUTPUT
A	*B*	*X*
0	0	0
0	1	0
1	0	0
1	1	1

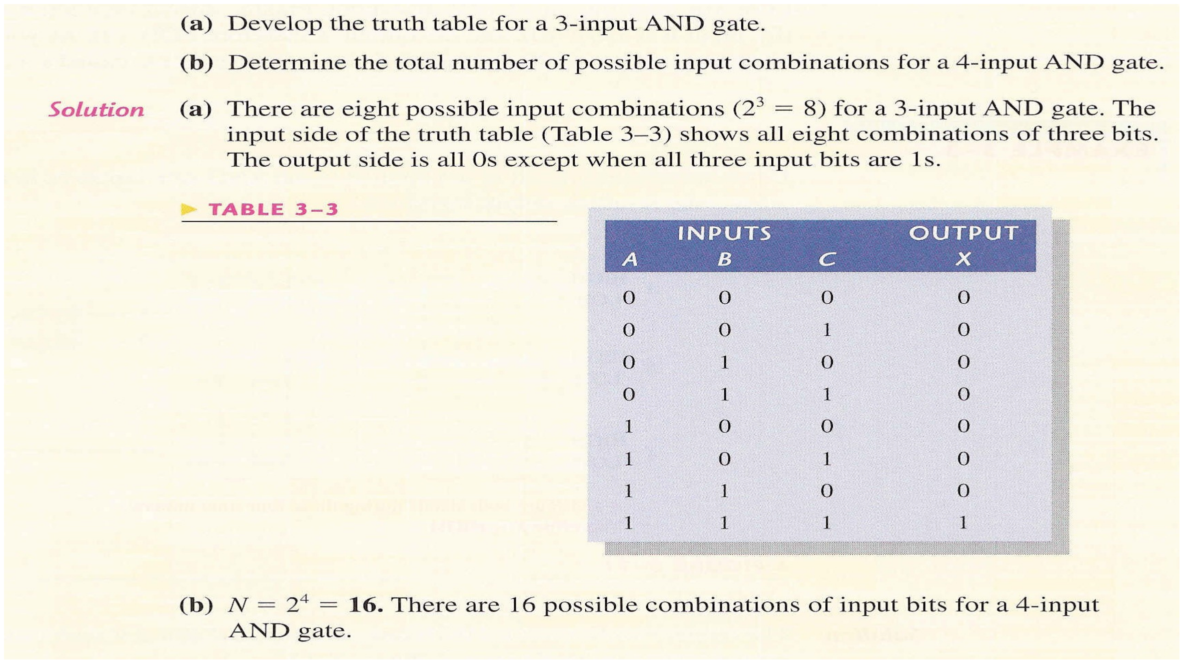
1 = HIGH, 0 = LOW

The total number of possible combinations of binary inputs to a gate is determined by the following formula:

$$N = 2^n$$

where N is the number of possible input combinations and n is the number of input variables. To illustrate,

For two input variables:	$N = 2^2 = 4$ combinations
For three input variables:	$N = 2^3 = 8$ combinations
For four input variables:	$N = 2^4 = 16$ combinations



(a) Develop the truth table for a 3-input AND gate.

(b) Determine the total number of possible input combinations for a 4-input AND gate.

Solution **(a)** There are eight possible input combinations ($2^3 = 8$) for a 3-input AND gate. The input side of the truth table (Table 3–3) shows all eight combinations of three bits. The output side is all 0s except when all three input bits are 1s.

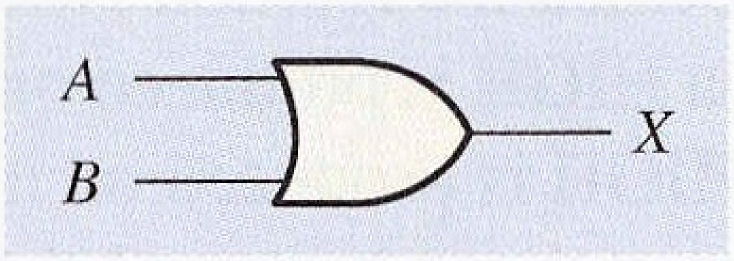
► TABLE 3–3

INPUTS			OUTPUT
A	*B*	*C*	*X*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

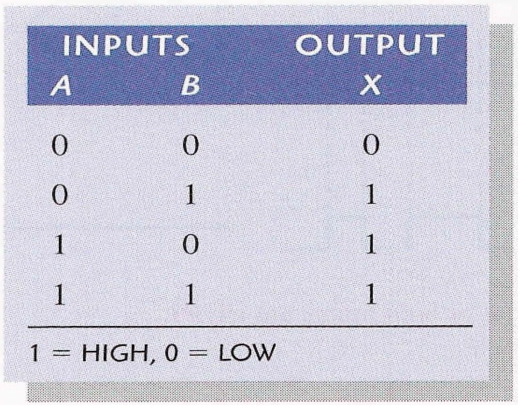
(b) $N = 2^4 = \mathbf{16}$. There are 16 possible combinations of input bits for a 4-input AND gate.

20. Write the truth table and logical symbol of OR gate.

- ✓ The **OR** gate is composed of two or more inputs and a single output and performs logical addition.
- ✓ The **OR gate** produces a HIGH output when any of the inputs is HIGH. The output is LOW only when all of the inputs are LOW.
- ✓ The standard symbol and truth table for OR gate is shown below.



(a) Distinctive shape



INPUTS		OUTPUT
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	1

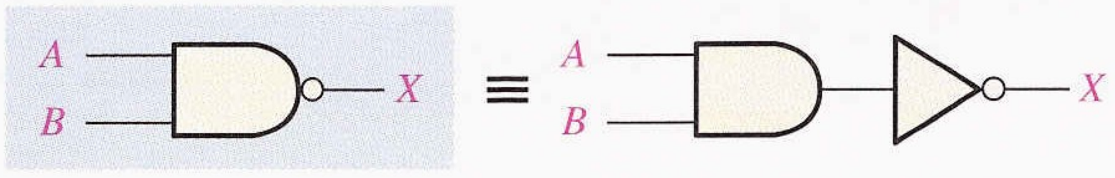
1 = HIGH, 0 = LOW

◄ TABLE 3–5

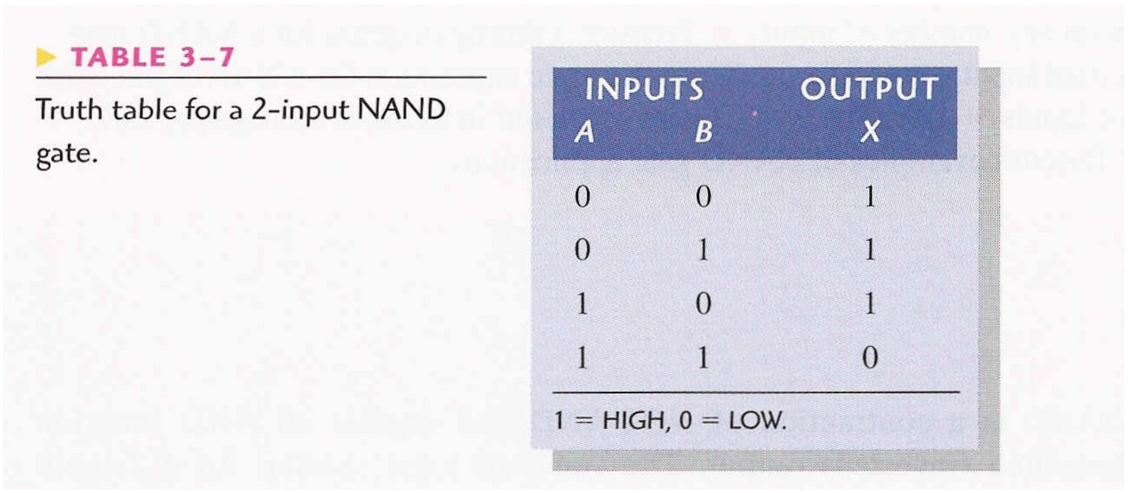
Truth table for a 2-input OR gate.

21. Write the truth table and logical symbol of NAND gate.

- ✓ The NAND gate is composed of two or more inputs and a single output.
- ✓ The NAND function is the complement of the AND function.
- ✓ **A NAND gate** produces a LOW output only when all the inputs are HIGH. When any of the inputs is LOW, the output will be HIGH.
- ✓ The standard symbol and truth table for NAND gate is shown below.



(a) Distinctive shape, 2-input NAND gate and its NOT/AND equivalent



▶ TABLE 3–7

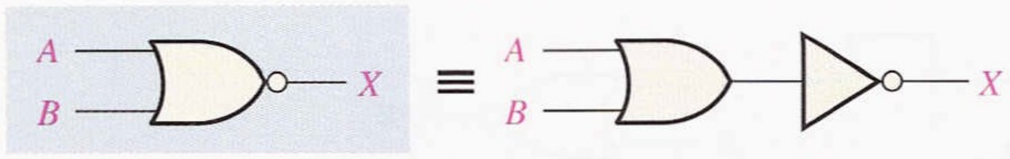
Truth table for a 2-input NAND gate.

INPUTS		OUTPUT
A	*B*	*X*
0	0	1
0	1	1
1	0	1
1	1	0

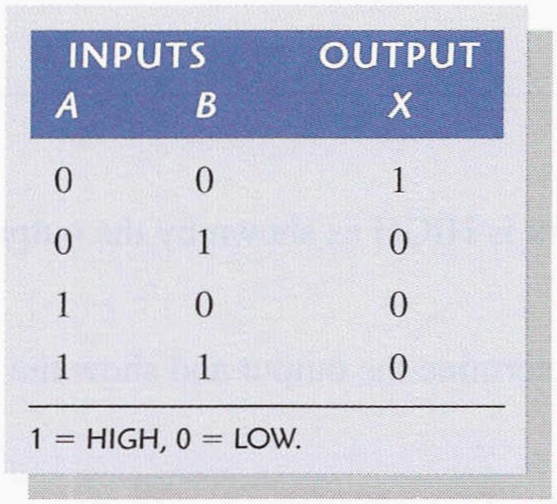
1 = HIGH, 0 = LOW.

22. Write the truth table and logical symbol of NOR gate.

- ✓ The NOR gate is composed of two or more inputs a single output.
- ✓ The NOR function is the complement of the OR function.
- ✓ **A NOR gate** produces a LOW output when any of its inputs is HIGH. Only when all of its inputs are LOW, is the output HIGH.
- ✓ The standard symbol and truth table for NAND gate is shown below.



(a) Distinctive shape, 2-input NOR gate and its NOT/OR equivalent



INPUTS		OUTPUT
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

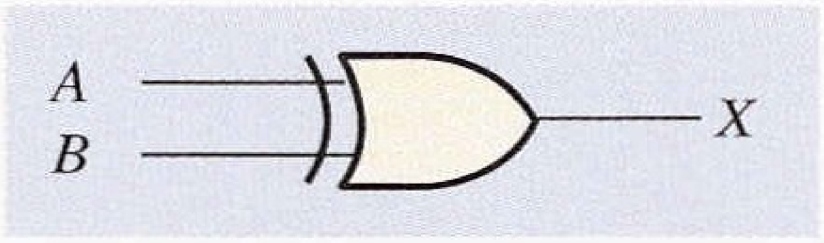
1 = HIGH, 0 = LOW.

◀ TABLE 3–9

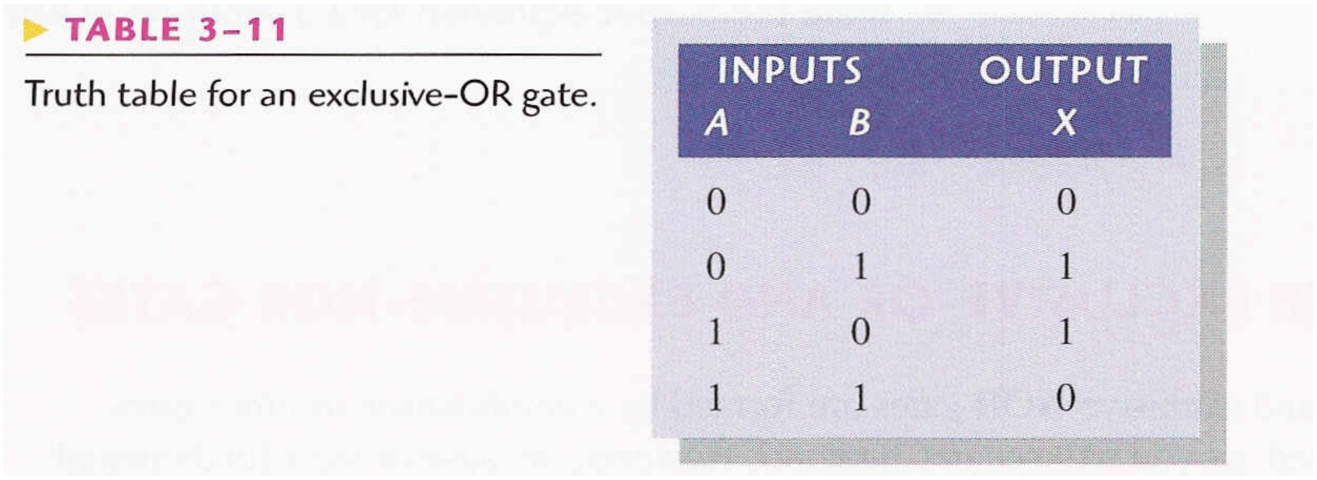
Truth table for a 2-input NOR gate.

23. Write the truth table and logical symbol of XOR gate

- ✓ The **Exclusive –OR (XOR)** has only two inputs and one output.
- ✓ The output of an **XOR gate** is HIGH only when the two inputs are at opposite logic levels.
- ✓ The standard symbol and truth table for Exclusive –OR (XOR) gate is shown below.



(a) Distinctive shape



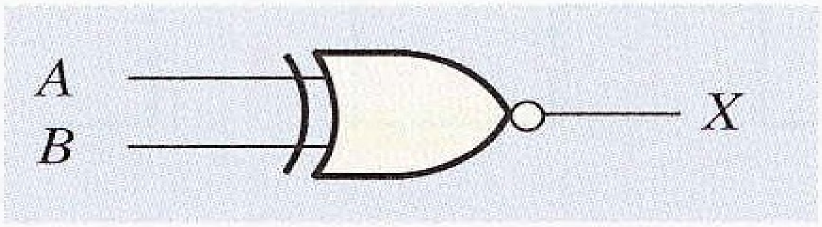
▶ TABLE 3–11

Truth table for an exclusive-OR gate.

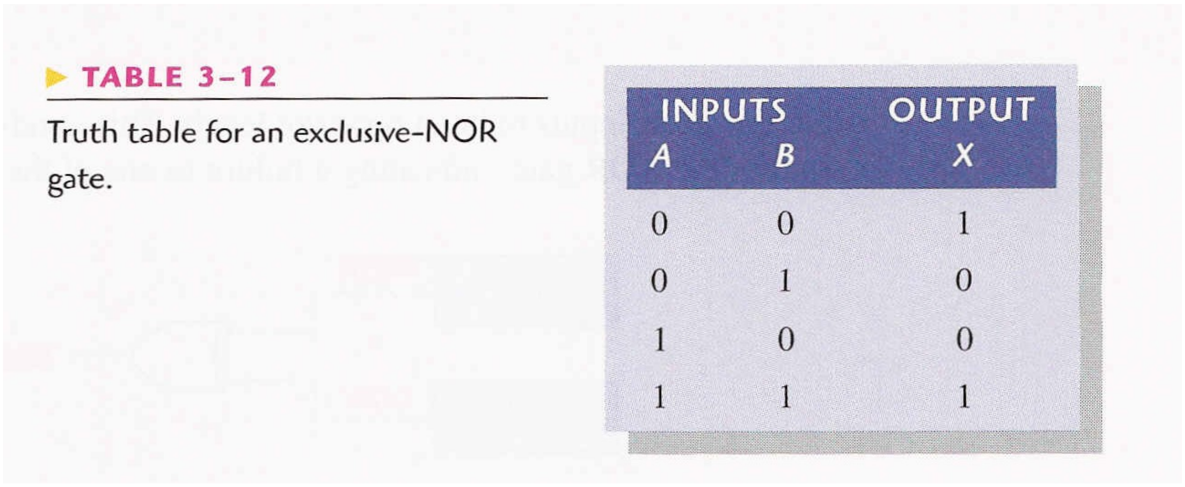
INPUTS		OUTPUT
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

24. Write the truth table and logical symbol of XOR gate

- ✓ The Exclusive –NOR (XNOR) has only two inputs and one output. It is logically equivalent to EX-OR gate followed by inverter.
- ✓ The **XNOR gate** produces a HIGH output only when both inputs are at the same logic level.
- ✓ The standard symbol and truth table for Exclusive –OR (XOR) gate is shown below.



(a) Distinctive shape



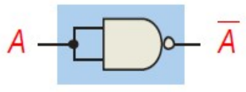
▶ TABLE 3–12

Truth table for an exclusive-NOR gate.

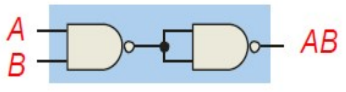
INPUTS		OUTPUT
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

NAND Gate as a Universal Gate

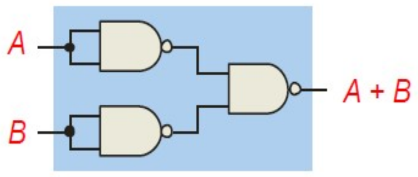
NAND gates are sometimes called **universal** gates because they can be used to produce the other basic Boolean functions.



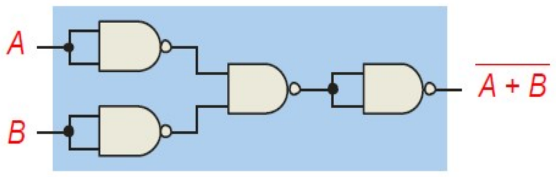
Inverter



AND gate



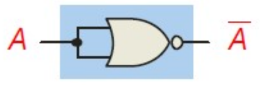
OR gate



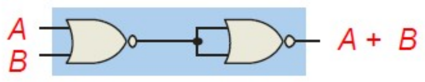
NOR gate

NOR Gate as a Universal Gate

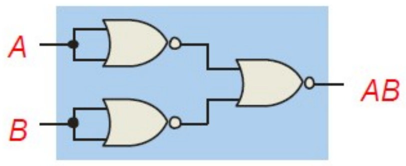
NOR gates are also **universal** gates and can form all of the basic gates.



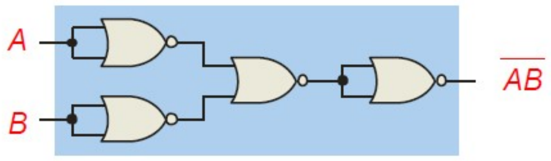
Inverter



OR gate

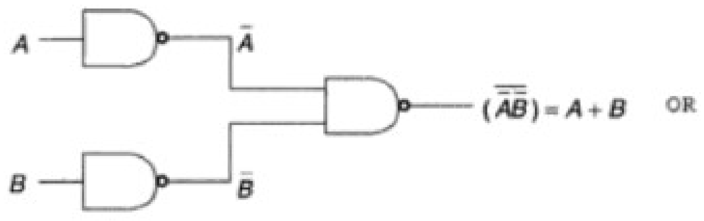


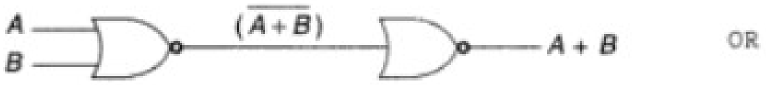
AND gate



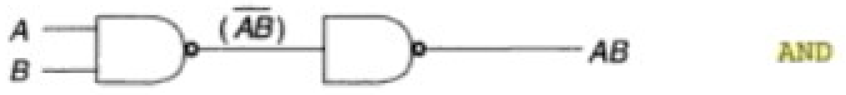
NAND gate

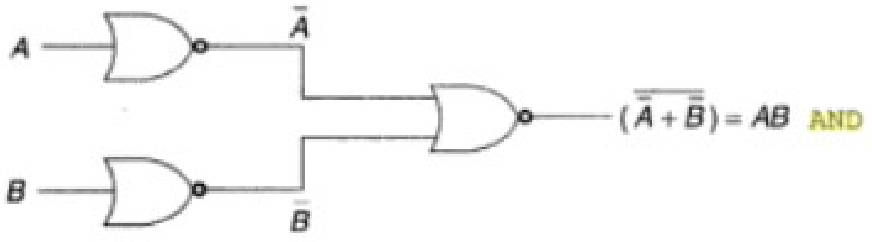
25. Construct the OR gate using NAND & NOR.



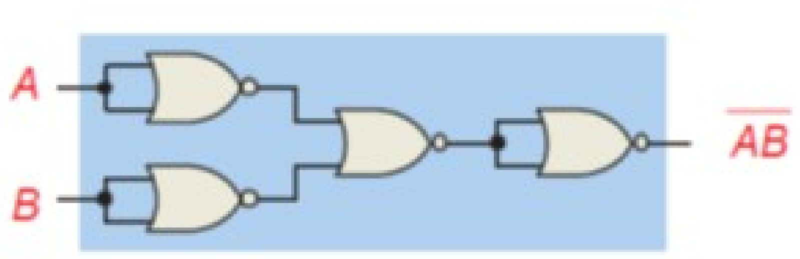


26. Construct the AND gate using NAND & NOR.





27. Realize NAND gate using NOR gate.



NAND gate

28. Write the Rules of Boolean algebra.

Basic rules of Boolean algebra.

1. $A + 0 = A$	**7.** $A \cdot A = A$
2. $A + 1 = 1$	**8.** $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	**9.** $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	**10.** $A + AB = A$
5. $A + A = A$	**11.** $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	**12.** $(A + B)(A + C) = A + BC$

A, *B*, or *C* can represent a single variable or a combination of variables.

29. Write the Laws of Boolean algebra.

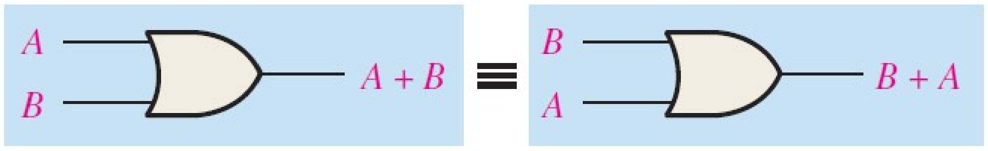
The basic laws of Boolean algebra—the **commutative laws** for addition and multiplication, the **associative laws** for addition and multiplication, and the **distributive law.**

Commutative Laws

The *commutative law of addition* for two variables is written as

A + B = B + A

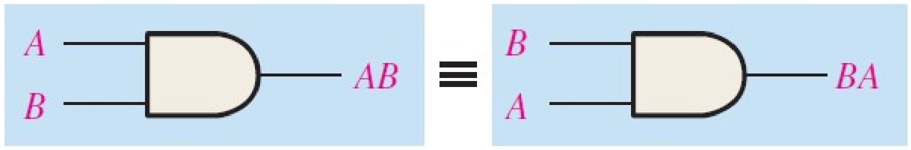
This states that the order in which the variables are ORed makes no difference in the output.



The *commutative law of multiplication* for two variables is

AB =BA

This states that the order in which the variables are ANDed makes no difference in the output.

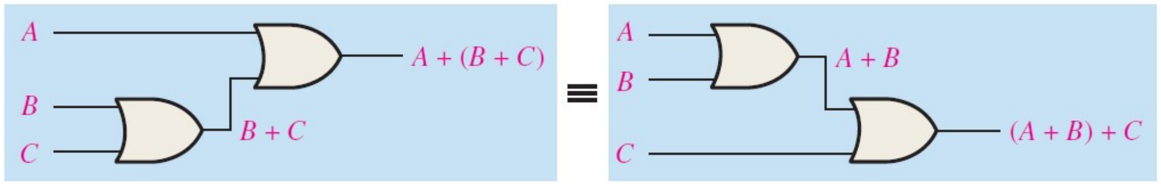


Associative Laws

The *associative law of addition* is written as follows for three variables:

A+(B+C) = (A+B)+C

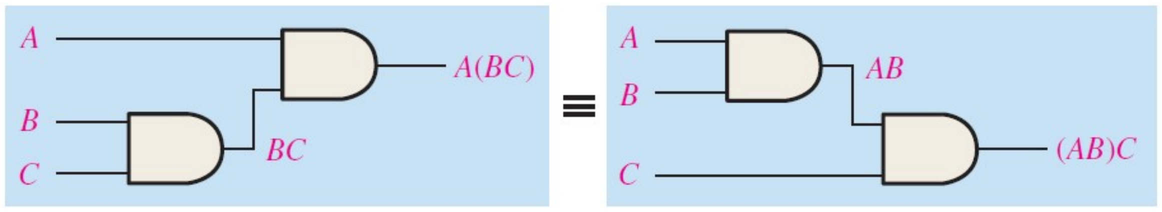
This law states that in ORing of several variables, the result is the same regardless of the grouping of the variables.



The *associative law of multiplication* is written as follows for three variables:

A .(B.C) = (A.B). C

This law states that when ORing more than two variables, the result is the same regardless of the grouping of the variables.

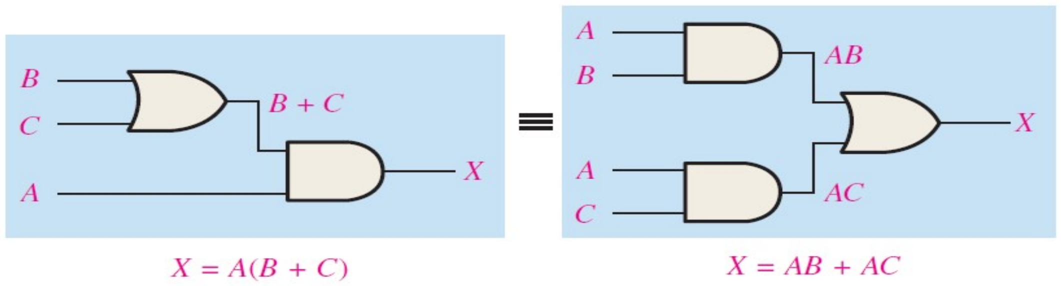


Distributive Law

The distributive law is written for three variables as follows:

A(B+C) = AB+AC

This law states that ORing two or more variables and then ANDing the result with a single variable is equivalent to ANDing the single variable with each of the two or more variables and then ORing the products.



30. State De-Morgan's theorem and illustrate the same with truth table.

1. DeMorgan's first theorem is stated as follows:

The complement of a product of variables is equal to the sum of the complements of the variables.

$$\overline{XY} = \overline{X} + \overline{Y}$$

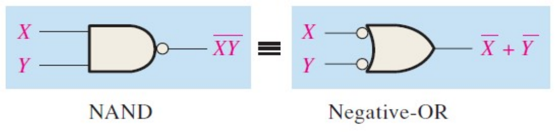
The complement of two or more ANDed variables is equivalent to the OR of the complements of the individual variables.

2. DeMorgan's second theorem is stated as follows:

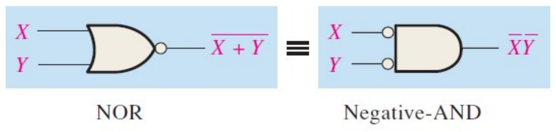
The complement of a sum of variables is equal to the product of the complements of the variables.

$$\overline{X + Y} = \overline{X}\,\overline{Y}$$

The complement of two or more ORed variables is equivalent to the AND of the complements of the individual variables.



Inputs		Output	
X	Y	$\overline{XY}$	$\overline{X}+\overline{Y}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0



Inputs		Output	
X	Y	$\overline{X+Y}$	$\overline{X}\,\overline{Y}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Apply DeMorgan's theorems to each of the following expressions:

(a) $\overline{(A + B + C)D}$

(b) $\overline{ABC + DEF}$

(c) $\overline{A\overline{B} + \overline{C}D + EF}$

Solution

(a) Let $A + B + C = X$ and $D = Y$. The expression $\overline{(A + B + C)D}$ is of the form $\overline{XY} = \overline{X} + \overline{Y}$ and can be rewritten as

$$\overline{(A + B + C)D} = \overline{A + B + C} + \overline{D}$$

Next, apply DeMorgan's theorem to the term $\overline{A + B + C}$.

$$\overline{A + B + C} + \overline{D} = \overline{A}\,\overline{B}\,\overline{C} + \overline{D}$$

(b) Let $ABC = X$ and $DEF = Y$. The expression $\overline{ABC + DEF}$ is of the form $\overline{X + Y} = \overline{X}\,\overline{Y}$ and can be rewritten as

$$\overline{ABC + DEF} = (\overline{ABC})(\overline{DEF})$$

Next, apply DeMorgan's theorem to each of the terms $\overline{ABC}$ and $\overline{DEF}$.

$$(\overline{ABC})(\overline{DEF}) = (\overline{A} + \overline{B} + \overline{C})(\overline{D} + \overline{E} + \overline{F})$$

(c) Let $A\overline{B} = X$, $\overline{C}D = Y$, and $EF = Z$. The expression $\overline{A\overline{B} + \overline{C}D + EF}$ is of the form $\overline{X + Y + Z} = \overline{X}\,\overline{Y}\,\overline{Z}$ and can be rewritten as

$$\overline{A\overline{B} + \overline{C}D + EF} = (\overline{A\overline{B}})(\overline{\overline{C}D})(\overline{EF})$$

Next, apply DeMorgan's theorem to each of the terms $\overline{A\overline{B}}$, $\overline{\overline{C}D}$, and $\overline{EF}$.

$$(\overline{A\overline{B}})(\overline{\overline{C}D})(\overline{EF}) = (\overline{A} + B)(C + \overline{D})(\overline{E} + \overline{F})$$

Apply DeMorgan's theorems to each expression:

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

(b) $\overline{(\overline{A}+B)+CD}$

(c) $\overline{(A+B)\overline{C}\,\overline{D}+E+\overline{F}}$

Solution

(a) $\overline{(\overline{A+B})+\overline{C}} = (\overline{\overline{A+B}})\overline{\overline{C}} = (A+B)C$

(b) $\overline{(\overline{A}+B)+CD} = (\overline{\overline{A}+B})\overline{CD} = (\overline{\overline{A}}\,\overline{B})(\overline{C}+\overline{D}) = A\overline{B}(\overline{C}+\overline{D})$

(c) $\overline{(A+B)\overline{C}\,\overline{D}+E+\overline{F}} = ((\overline{(A+B)\overline{C}\,\overline{D}})(\overline{E+\overline{F}}) = (\overline{A}\,\overline{B}+C+D)\overline{E}F$

1. $\mathbf{Q = A.B + B}$

$$Q = A.B + B$$
$$Q = B.(A+1)$$
$$Q = B.1$$
$$Q = B$$

2. $\mathbf{Q = C.(A + \overline{C})}$

$$Q = C.(A+\overline{C})$$
$$Q = A.C + C.\overline{C}$$
$$Q = A.C + 0$$
$$Q = A.C$$

3. $\mathbf{Q = A.B.\overline{C} + A.\overline{C} + \overline{A}.\overline{C}.D + \overline{A}.\overline{C}.\overline{D}}$

$$Q = A.B.\overline{C} + A.\overline{C} + \overline{A}.\overline{C}.D + \overline{A}.\overline{C}.\overline{D}$$
$$Q = A.\overline{C}.(B+1) + \overline{A}.\overline{C}.(D+\overline{D})$$
$$Q = A.\overline{C}.1 + \overline{A}.\overline{C}.1$$
$$Q = A.\overline{C} + \overline{A}.\overline{C}$$
$$Q = \overline{C}.(A+\overline{A})$$
$$Q = \overline{C}.1$$
$$Q = \overline{C}$$

4. $\mathbf{Q = A.B.(\overline{B}+C)+B.C+B}$

$$Q = A.B.(\overline{B}+C)+B.C+B$$
$$Q = A.B.\overline{B}+A.B.C+B.C+B$$
$$Q = A.0+A.B.C+B.(C+1)$$
$$Q = A.B.C+B.1$$
$$Q = A.B.C+B$$
$$Q = B.(A.C+1)$$
$$Q = B.1$$
$$Q = B$$

5. $\mathbf{Q = B.(A+\overline{C})+A+A.(\overline{A}+B)}$

$$Q = B.(A+\overline{C})+A+A.(\overline{A}+B)$$
$$Q = A.B+B.\overline{C}+A+A.\overline{A}+A.B$$
$$Q = A.B+B.\overline{C}+A$$
$$Q = A.(B+1)+B.\overline{C}$$
$$Q = A+B.\overline{C}$$

6. $\mathbf{Q = A.B.\overline{C}+B.C+\overline{A}.B.\overline{C}+A.B.\overline{B}}$

$$Q = A.B.\overline{C}+B.C+\overline{A}.B.\overline{C}+A.B.\overline{B}$$
$$Q = B.\overline{C}.(A+\overline{A})+B.C+A.0$$
$$Q = B.\overline{C}.1+B.C$$
$$Q = B.\overline{C}+B.C$$
$$Q = B.(\overline{C}+C)$$
$$Q = B.1$$

7. $\mathbf{Q = ABC + BCD + BC\overline{D} + B\overline{C}D + AB\overline{C} + \overline{A}B\overline{C}}$

$$Q = A.B.C + B.C.D + B.C.\overline{D} + B.\overline{C}.D + A.B.\overline{C} + \overline{A}.B.\overline{C}$$
$$Q = A.B.C + B.C.(D + \overline{D}) + B.\overline{C}.D + B.\overline{C}.(A + \overline{A})$$
$$Q = A.B.C + B.C.1 + B.\overline{C}.D + B.\overline{C}.1$$
$$Q = A.B.C + B.C + B.\overline{C}.D + B.\overline{C}$$
$$Q = B.C.(A + 1) + B.\overline{C}.(D + 1)$$
$$Q = B.C.1 + B.\overline{C}.1$$
$$Q = B.C + B.\overline{C}$$
$$Q = B.(C + \overline{C})$$
$$Q = B.1$$
$$Q = B$$

8. $\mathbf{Q = ABCD + ABD + A\overline{B}D + A\overline{B}CD + ACD + \overline{A}CD}$

$$Q = A.B.C.D + A.B.D + A.\overline{B}.D + A.\overline{B}.C.D + A.C.D + \overline{A}.C.D$$
$$Q = A.B.D.(C + 1) + A.\overline{B}.D + A.C.D.(\overline{B} + 1) + \overline{A}.C.D$$
$$Q = A.B.D.1 + A.\overline{B}.D + A.C.D.1 + \overline{A}.C.D$$
$$Q = A.B.D + A.\overline{B}.D + A.C.D + \overline{A}.C.D$$
$$Q = A.D.(B + \overline{B}) + C.D.(A + \overline{A})$$
$$Q = A.D.1 + C.D.1$$
$$Q = A.D + C.D$$
$$Q = D.(A + C)$$

2. Simplify the following expression by applying Rules of Boolean Algebra

$$\mathbf{Q = AB\overline{C} + ABC + A\overline{B}}$$

Simplification can begin by combining the first two terms only as follows:

$$\mathbf{Q = AB(\overline{C} + C) + A\overline{B}}$$

Using our sixth identity the term $\mathbf{C + \overline{C} = 1}$ so the expression now becomes:

$$\mathbf{Q = AB1 + A\overline{B}}$$

Using the first identity A.B.1 = A.B so the expression becomes

$$\mathbf{Q = AB + A\overline{B}}$$

Now we can remove the common term again to leave:

$$\mathbf{Q = A(B + \overline{B})}$$

Using our sixth identity the term $\mathbf{B + \overline{B} = 1}$ so the expression now becomes:

$$\mathbf{Q = A.1}$$

Using the first identity A.1 = A so the expression finally becomes

$$\mathbf{Q = A}$$

$$Q = \overline{A}.B + A.B.\overline{C} + A.B + C$$

Start by looking for possible common factors which will leave behind one of our two key identities inside the bracket, in this case A.B in terms 2 and 3. This gives the following simplification.

$$Q = \overline{A}.B + A.B.(\overline{C} + 1) + C$$

Using our fourth identity the term $\overline{C} + 1 = 1$ so the expression now becomes:

$$Q = \overline{A}.B + A.B.1 + C$$

Using the first identity A.B.1 = A.B so the expression becomes

$$Q = \overline{A}.B + A.B + C$$

Again we look for common terms, this time between the first two terms to give the following:

$$Q = B.(\overline{A} + A) + C$$

Using our sixth identity the term $A + \overline{A} = 1$ so the expression now becomes:

$$Q = B.1 + C$$

Using the first identity B.1 = B so the expression becomes

$$Q = B + C$$

$$Q = \overline{A}B + \overline{A}B.C + \overline{A}\overline{B}.\overline{C} + A\overline{B}.\overline{C}$$

Solution:

$$Q = \overline{A}.B + \overline{A}.B.C + \overline{A}.\overline{B}.\overline{C} + A.\overline{B}.\overline{C}$$
$$Q = \overline{A}.B + \overline{A}.B.C + \overline{B}.\overline{C}.(\overline{A} + A)$$
$$Q = \overline{A}.B + \overline{A}.B.C + \overline{B}.\overline{C}.1$$
$$Q = \overline{A}.B.(1 + C) + \overline{B}.\overline{C}$$
$$Q = \overline{A}.B.1 + \overline{B}.\overline{C}$$
$$Q = \overline{A}.B + \overline{B}.\overline{C}$$

4. Simplify the following expression.

$$Q = B.C.(\overline{C} + D) + C.D + C + \overline{A}$$
$$Q = B.C.(\overline{C} + D) + C.D + C + \overline{A}$$
$$Q = B.C.\overline{C} + B.C.D + C.D + C + \overline{A}$$
$$Q = B.0 + B.C.D + C.(D + 1) + \overline{A}$$
$$Q = B.C.D + C + \overline{A}$$
$$Q = C.(B.D + 1) + \overline{A}$$
$$Q = C.1 + \overline{A}$$
$$Q = C + \overline{A}$$

5. Simplify the expression: $Q = A.B.C + A\overline{C} + C.(D+\overline{C}) + A$

$$Q = A.B.C + A.\overline{C} + C.(D+\overline{C}) + A$$
$$Q = A.(B.C+\overline{C}) + C.D + C.\overline{C} + A$$
$$Q = A.(B+\overline{C}) + C.D + 0 + A$$
$$Q = A.B + A.\overline{C} + C.D + A$$
$$Q = A.(B+1) + A.\overline{C} + C.D$$
$$Q = A + A.\overline{C} + C.D$$
$$Q = A.(1+\overline{C}) + C.D$$
$$Q = A + C.D$$

6. Simplify the expression: $Q = A.B.\overline{C} + A.\overline{C} + \overline{A}.\overline{C}.D + \overline{A}.\overline{C}.\overline{D}$

$$Q = A.B.\overline{C} + A.\overline{C} + \overline{A}.\overline{C}.D + \overline{A}.\overline{C}.\overline{D}$$
$$Q = A.\overline{C}.(B+1) + \overline{A}.\overline{C}.(D+\overline{D})$$
$$Q = A.\overline{C}.1 + \overline{A}.\overline{C}.1$$
$$Q = A.\overline{C} + \overline{A}.\overline{C}$$
$$Q = \overline{C}.(A+\overline{A})$$
$$Q = \overline{C}.1$$
$$Q = \overline{C}$$

7.Simplify the expression: $Q = A.B.(\overline{B}+C) + B.C + B$

$$Q = A.B.(\overline{B}+C) + B.C + B$$
$$Q = A.B.\overline{B} + A.B.C + B.C + B$$
$$Q = A.0 + A.B.C + B.(C+1)$$
$$Q = A.B.C + B.1$$
$$Q = A.B.C + B$$
$$Q = B.(A.C+1)$$
$$Q = B.1$$
$$Q = B$$

8. Simplify the expression: $Q = B.(A+\overline{C}) + A + A.(\overline{A}+B)$

$$Q = B.(A+\overline{C}) + A + A.(\overline{A}+B)$$
$$Q = A.B + B.\overline{C} + A + A.\overline{A} + A.B$$
$$Q = A.B + B.\overline{C} + A$$
$$Q = A.(B+1) + B.\overline{C}$$
$$Q = A + B.\overline{C}$$

32. Simplify the expression $\boldsymbol{Y = A\overline{B}D + A\overline{B}\,\overline{D}}$

33. Simplify the expression $Y = (\overline{A} + B)(A + B)$

34. Simplify the expression $Y = ACD + \overline{A}BCD$

35. Simplify the expression $Y = \overline{AB + C}$

36. Simplify the expression $Y = \overline{(\overline{A} + C) \cdot (B + \overline{D})}$

37. Simplify the expression $Y = \overline{A + \overline{B} \cdot C}$