Department of Computer Science & Engineering
Introduction to Digital Electronics
Course: C.S.E.- 1211

Number System

Number System: The process of representing a number is called number system.
We use numbers:

- ✓ to communicate,
- ✓ to perform tasks,
- ✓ to quantify,
- ✓ to measure.

Types of Number System: There are two types of number systems which are given below:

1. Non-positional number system,
2. Positional number system.

1.Non-Positional Number System: A number in which each symbol represents the same value, regardless of its position in the number and the symbols are simply added to find out the value of a particular number. It is very difficult to perform arithmetic with such a number system.

Examples: Symbolic number system:

- ✓ uses Roman numerals (I=1, V=5, X=10, L= 50, C= 100, D= 500,M= 1000).

2.Positional Number System: A number system in which there are only a few symbols called digits, and these symbols represent different values, depending on the position they occupy in the number. The value of each digit in such a number is determined by the digit itself, the position of the digit in the number, and the base of the number system.

There are four types of positional number systems which are given below:

1.Decimal Number System: The number system, which we use in our day to day life is called decimal number system. In this system, the base is equal to 10, because there are altogether ten symbols or digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). In this system, the successive positions to the left of the decimal point represent units, tens, hundreds, thousands etc. Each position represents a specific power of the base 10.

Example: $(2586)_{10}$ = 2*10^3 + 5*10^2 + 8*10^1 + 6*10^0 = 2000 + 500 + 80 + 6 = $(2586)_{10}$

2.Binary Number System: The binary number system is exactly like the decimal number system, except that the base is 2, instead of 10. The largest single digit is 1. We have only two symbols or digits (0 or 1) which can be used in this number system. Each position in a binary number represents a power of the base 2. The binary digits are also known as ***bits.***

Example: $(10101)_2 = 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 16 + 0 + 4 + 0 + 1 = (21)_{10}$

3.Octal Number System: In octal number system, the base is 8. Hence, there are only eight symbols or digits: 0, 1, 2, 3, 4, 5, 6, 7. The largest single digit is 7. Each position in an octal number represents a power of the base 8. Advantage: (i)

Example: $(2057)_8 = 2*8^3 + 0*8^2 + 5*8^1 + 7*8^0 = 1024 + 0 + 40 + 7 = (1071)_{10}$

4. Hexadecimal Number System: The hexadecimal number system is one with a base of 16, having 16 single character digits or symbols. The first 10 digits are the digits of the decimal number system: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. The remaining six digits are denoted by the symbols A, B, C, D, E, and F representing the decimal values 10, 11, 12, 13, 14, 15 respectively. Hence, the largest single digits is F (15). Each position in the hexadecimal number system represents a power of the base 16. **Example:** $(1AF)_{16} = 1*16^2 + A*16^1 + F*16^0$

$= 1*256 + 10*16 + 15*1 = 256 + 160 + 15 = (431)_{10}$

Binary numbers to Octal Number

1. $(11011001)_2 = (011)\ (011)\ (001) = (331)_8$
2. $(101110)_2 = (101)\ (110) = (56)_8$
3. $(1101110)_2 = (001)(101)(110) = (156)_8$

Binary numbers to Decimal Number

1. $(11100110)_2 = (?)_{10}$

Answer: $= 1*2^7 + 1*2^6 + 1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0$

$= 128 + 64 + 32 + 0 + 0 + 4 + 2 + 0 = (230)_{10}$

2. $(11011011)_2 = (?)_{10}$

Answer: $= 1*2^7 + 1*2^6 + 0*2^5 + 1*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0$

$= 128 + 64 + 0 + 16 + 8 + 0 + 2 + 1 = (219)_{10}$

3. $(1101110)_2 = (?)_{10}$

Answer: $= 1*2^6 + 1*2^5 + 0*2^4 + 1*2^3 + 1*2^2 + 1*2^1 + 0*2^0$

$= 64 + 32 + 0 + 8 + 4 + 2 + 0 = (110)_{10}$

4. $(101.101)_2 = (?)_{10}$

Answer: $= 1*2^2 + 0*2^1 + 1*2^0 + 1*2^{-1} + 0*2^{-2} + 1*2^{-3}$

$= 4 + 0 + 1 + (1/2) + 0 + (18) = 5 + 0.5 + 0.125 = (5.625)_{10}$

Binary numbers to Hexadecimal Number

1. $(11011001)_2 = (1101)\ (1001) = (D9)_{16}$
2. $(11010011)_2 = (1101)\ (0011) = (D3)_{16}$
3. $(01011100111000)_2 = (0001)\ (0111)\ (0011)\ (1000) = (1738)_{16}$

Some features of Binary Numbers

3-bit groupings: Octal (radix 8) groups three binary digits. Digits will have one of the eight values: 0= 000, 1= 001, 2= 010, 3= 011, 4= 100, 5= 101, 6= 110, 7= 111.
4-digit groupings: Hexa-decimal (radix 16). Digits will have one of the sixteen values 0 through 15. Decimal values from 10 to 15 are designated as A (=10), B (=11), C (=12), D (=13), E (=14) and F (=15).
0= 0000, 1= 0001, 2= 0010, 3= 0011, 4= 0100, 5= 0101, 6= 0110 and 7= 0111, 8 = 1000, 9= 1001, A= 1010, B= 1011, C= 1100, D= 1101, E= 1110, F= 1111.

Octal numbers to Decimal Number

1. $(2057)_8 = 2*8^3 + 0*8^2 + 5*8^1 + 7*8^0 = 1024 + 0 + 40 + 7 = (1071)_{10}$
2. $(4706)_8 = 4*8^3 + 7*8^2 + 0*8^1 + 6*8^0 = 2048 + 448 + 0 + 6 = (2502)_{10}$
3. $(2115)_8 = 2*8^3 + 1*8^2 + 1*8^1 + 5*8^0 = 1024 + 64 + 8 + 5 = (1101)_{10}$
4. $(33.56)_8 = 3*8^1 + 3*8^0 + 5*8^{-1} + 6*8^{-2} = (27.69875)_{10}$

Hexadecimal numbers to Decimal Number

1. $(1AF)_{16} = 1*16^2+A*16^1+F*16^0 = 1*256+10*16+15*1 = 256+160+15 = (431)_{10}$
2. $(1AC)_{16} = 1*16^2+A*16^1+C*16^0 = 1*256+10*16+12*1 = 256+160+12 = (428)_{10}$
3. $(4FD)_{16} = 4*16^2+F*16^1+D*16^0 = 1024+15*16+14*1 = 1024+240+14 = (1278)_{10}$
4. $(E5.A)_{16} = 14*16^1 + 5*16^0 + 10*16^{-1} = (304.625)_{10}$

Decimal Number to Binary Numbers:

1

	Quotient	Remainder
156/ 2	= 78	0
78/2	= 39	0
39/2	= 19	1
19/2	= 9	1
9/2	= 4	1
4/2	= 2	0
2/2	= 1	0
1/2	= 0	1

$(156)_{10} = (10011100)_2$

2

	Quotient	Remainder
42/ 2	= 21	0
21/2	= 10	1
10/2	= 5	0
5/2	= 2	1
2/2	= 1	0
1/2	= 0	1

$(42)_{10} = (101010)_2$

3

	Quotient	Remainder
135/ 2	= 67	1
67/2	= 33	1
33/2	= 16	1
16/2	= 8	0
8/2	= 4	0
4/2	= 2	0
2/2	= 1	0
1/2	= 0	1

$(135)_{10} = (1000111)_2$

4 $(0.6875)_{10} = (?)_2$

$= 0.6875 \times 2 = 1.3750 \rightarrow 1$
$= 0.3750 \times 2 = 0.7500 \rightarrow 0$
$= 0.75 \times 2 = 1.50 \rightarrow 1$
$= 0.50 \times 2 = 1.00 \rightarrow 1$

$\therefore$ Result $= (.1011)_2$

Result: $(.4065177)_8$

(4) $(0.513)_{10} = (?)_8$

$= 0.513 \times 8 = 4.104 \rightarrow 4$
$= 0.104 \times 8 = 0.832 \rightarrow 0$
$= 0.832 \times 8 = 6.656 \rightarrow 6$
$= 0.656 \times 8 = 5.248 \rightarrow 5$
$= 0.248 \times 8 = 1.984 \rightarrow 1$
$= 0.984 \times 8 = 7.875 \rightarrow 7$
$= 0.875 \times 8 = 7 \rightarrow 7$

Decimal Number to Octal Numbers:

(1)

	Quotient	Remainder
952/8	= 119	0
119/8	= 14	7
14/8	= 1	6
1/8	= 0	1

$(952)_{10} = (1670)_8$

(2)

	Quotient	Remainder
678/ 8	= 84	6
84/ 8	=10	4
10/ 8	=1	2
1/ 8	=0	1

$(678)_{10} = (1246)_8$

(3)

	Quotient	Remainder
899/8	= 112	3
112/8	= 14	0
14/8	= 1	6
1/8	= 0	1

$(899)_{10} = (1603)_8$

Decimal Number to Hexadecimal Numbers:

	Quotient	Remainder
428/16	= 26	12(C)
26/16	= 1	10(A)
1/16	= 0	1

$(428)_{10} = (1AC)_{16}$

	Quotient	Remainder
678/16	= 42	6
42/16	= 2	10(A)
2/16	= 0	2

$(678)_{10} = (2A6)_{16}$

	Quotient	Remainder
3315.3/16	= 207	3
207/16	= 12	15(F)
12/16	= 0	12(C)
.3*16	= 4.8	4
.8*16	= 12.8	12(C)
.8*16	= 12.8	12(C)
.8*16	= 12.8	12(C)

$(3318.3)_{10} = (CF3.4CCC)_{16}$

⇒ 1. Find the 1's complement of 37_{10}

Rules: $(Base)^n - 1$ where, n = digit

Ans: Since the number has 2 digits and the value of base is 10.

$\therefore (Base)^n - 1 = (10)^2 - 1 = 100 - 1 = 99$

Now, $99 - 37 = 62$.

$\therefore$ The complement of $37_{10} = 62_{10}$

2. Find complement of 6_8.

$= (8)^1 - 1 = 8 - 1 = 7$

$\therefore$ result $= 7 - 6 = 1$

⇒ Subtract 56_{10} from 92_{10} using Complementary Method

Ans: step-1: complement of $56_{10} = (10)^2 - 1 - 56 = 99 - 56 = 43_{10}$

step 2:

$$\begin{array}{r} 92 \\ +43 \\ \hline 135 \\ +1 \\ \hline 36 \end{array}$$ (add carry 1)

$\therefore$ Result is: 36

⇒ Subtract 35_{10} from 18_{10}

step-1: Complement of $35_{10} = (10)^2 - 1 - 35 = 64_{10}$

step-2:

$$\begin{array}{r} 18 \\ +64 \\ \hline 82 \end{array}$$

step-3: There is no carry.

$\therefore$ Result $= (10)^2 - 1 - 82$
$= 99 - 82$
$= 17$

$\therefore$ Result $= -17$

Compare: **1's complement:** 1's complement of binary number is transforming the 0 bit to 1 and 1 to 0.
example: 1's of 7 (0111) is:– (1000) = 8.

2's: 1's of binary number is 1 added to the 1's complement of binary number. **example:** 2's of 7 (0111) is—

$$\begin{array}{r l} 1000 & \text{(1's of 0111)} \\ +1 & \\ \hline 1001 & \text{(2's of 7)} \end{array}$$

Decimal Number to Octal Numbers:

	Quotient	Remainder
952/8	= 119	0
119/8	= 14	7
14/8	= 1	6
1/8	= 0	1

$(952)_{10} = (1670)_8$

	Quotient	Remainder
678/8	= 84	6
84/8	= 10	4
10/8	= 1	2
1/8	= 0	1

$(678)_{10} = (1246)_8$

	Quotient	Remainder
899/8	= 112	3
112/8	= 14	0
14/8	= 1	6
1/8	= 0	1

$(899)_{10} = (1603)_8$

Decimal Number to Hexadecimal Numbers:

	Quotient	Remainder
428/16	= 26	12(C)
26/16	= 1	10(A)
1/16	= 0	1

$(428)_{10} = (1AC)_{16}$

	Quotient	Remainder
678/16	= 42	6
42/16	= 2	10(A)
2/16	= 0	2

$(678)_{10} = (2A6)_{16}$

	Quotient	Remainder
3315.3/16	= 207	3
207/16	= 12	15(F)
12/16	= 0	12(C)
.3*16	= 4.8	4
.8*16	= 12.8	12(C)
.8*16	= 12.8	12(C)
.8*16	= 12.8	12(C)

$(3318.3)_{10} = (CF3.4CCC)_{16}$

⇒ Subtract 01110_2 from 10101_2 using complementary method.

Ans: ~~complement of~~

$$\begin{array}{r l} 10101 & \\ +10001 & \text{(complement of } 01110_2) \\ \hline 100110 & \\ +1 & \text{(add carry 1)} \\ \hline 00111 & \end{array}$$

∴ Result: 00111_2

⇒ Subtract 1111_2 from 1100_2 using complementary method.

Ans:

$$\begin{array}{r l} 1100 & \\ +0000 & \text{(complement of 1111)} \\ \hline 1100 & \end{array}$$

Since there is no carry, then we complement again 1100_2:

∴ −0011 is the results.

4) Compare BCD Code and Binary Code.

Ans: **Binary Coded Decimal:** (i) In BCD, each digit of a decimal number is coded as a 4 bit binary number between 0 to 9.

Use: (i) Instruments (ii) Calculators

(ii) It is difficult to do calculations.

(iii) It is not efficient.

(iv) Example: 1248 : 0001 0010 0100 1000 in BCD.

Binary code: (i) In Binary, the number is converted to base 2, binary code.

(ii) It is easy to do calculations.

(iii) It is more efficient.

(iv) Example: 1248 : 10011100000 in binary.

⇒ find out the 2's Complement of : 10011.

Ans: 1's complement of 10011 is : 01100

∴ 2's complement of 10011 is :

$$\begin{array}{r} 01100 \\ +1 \\ \hline 01101 \end{array}$$

Fractional Problem: Decimal to Binary: ⟶

(i) $\cdot 375 = \cdot 375 \times 2 = 0.750 \rightarrow 0$

$= 0.75 \times 2 = 1.50 \rightarrow 1$

$= \cdot 50 \times 2 = 1.00 \rightarrow 1$

$\text{result} = (.011)_2$

(ii) **Decimal to Binary:** $(10.7)_{10} = (?)_2$.

$\text{result} = (1011.1011001$

$$\begin{array}{r|l} 2 & 10 \\ \hline 2 & 5 - 1 \\ \hline 2 & 2 - 1 \\ \hline 2 & 1 - 0 \end{array}$$

1011

$\cdot 7 \times 2 = 1.40 \rightarrow 1$

$\cdot 4 \times 2 = 0.80 \rightarrow 0$

$\cdot 80 \times 2 = 1.60 \rightarrow 1$

$\cdot 60 \times 2 = 1.20 \rightarrow 1$

$\cdot 20 \times 2 = \cdot 40 \rightarrow 0$

$\cdot 40 \times 2 = 0.80 \rightarrow 0$

$\cdot 80 \times 2 = 1.60 \rightarrow 1$

⋮

Decimal Number to Octal Numbers:

	Quotient	Remainder
952/8	= 119	0
119/8	= 14	7
14/8	= 1	6
1/8	= 0	1

$(952)_{10} = (1670)_8$

	Quotient	Remainder
678/ 8	= 84	6
84/ 8	=10	4
10/ 8	=1	2
1/ 8	=0	1

$(678)_{10} = (1246)_8$

	Quotient	Remainder
899/8	= 112	3
112/8	= 14	0
14/8	= 1	6
1/8	= 0	1

$(899)_{10} = (1603)_8$

Decimal Number to Hexadecimal Numbers:

	Quotient	Remainder
428/16	= 26	12(C)
26/16	= 1	10(A)
1/16	= 0	1

$(428)_{10} = (1AC)_{16}$

	Quotient	Remainder
678/16	= 42	6
42/16	= 2	10(A)
2/16	= 0	2

$(678)_{10} = (2A6)_{16}$

	Quotient	Remainder
3315.3/16	= 207	3
207/16	= 12	15(F)
12/16	= 0	12(C)
.3*16	= 4.8	4
.8*16	= 12.8	12(C)
.8*16	= 12.8	12(C)
.8*16	= 12.8	12(C)

$(3318.3)_{10} = (CF3.4CCC)_{16}$

⇒ **BCD Code:** Each decimal digit represent 4 bit binary code.

Example:

0 → 0000	10 → 0001 0000
1 → 0001	11 → 0001 0001
2 → 0010	12 → 0001 0010
⋮	⋮
9 → 1001	98 → 1001 1000

⇒ **Gray Code:** The reflected binary code is known as Gray code. Here, only one bit changes in the transition from one number to the next higher number. The gray code is used in shaft encoder which is to indicate the angular position of a shaft.

Uses: Gray Code reduces errors.

Function: ~~(ABCD) with (GXYZ)~~ → **Binary to Gray:**

$(G_3\, G_2\, G_1\, G_0)$ with $(B_3\, B_2\, B_1\, B_0) \rightarrow$

$G_3 = B_3$; $G_2 = B_3 \oplus B_2$; $G_1 = B_2 \oplus B_1$; $G_0 = B_1 \oplus B_0$

$\therefore$ ~~$G_3 = B_3$; $B_2 = G_3 \oplus G_2$, $B_1 = G_2 \oplus G_1$~~ , $B_0 = G_1 \oplus G_0$.

Example: Binary to Gray code $(0010)_2 = ?$

where, $B_3 = 0$, $B_2 = 0$, $B_1 = 1$, $B_0 = 0$

$\therefore$ $G_3 = B_3 = 0$; $G_2 = B_3 \oplus B_2 = 0 \oplus 0 = 0$; $G_1 = B_2 \oplus B_1 = 0 \oplus 1 = 1$; $G_0 = B_1 \oplus B_0 = 1 \oplus 0 = 1$

$\therefore$ Final Gray Code is: (0011).

Example: Gray to Binary code: $(G_3 G_2 G_1 G_0)$ & $(B_3 B_2 B_1 B_0)$

Function: $\therefore$ $B_3 = G_3$; $B_2 = B_3 \oplus G_2$; $B_1 = B_2 \oplus G_1$; $B_0 = B_1 \oplus G_0$

$\therefore$ $(1101)_{Gray} = (?)_2$. where, $G_3 = 1$, $G_2 = 1$, $G_1 = 0$, $G_0 = 1$.

$\therefore$ $B_3 = 1$ $= 1$; $B_2 = B_3 \oplus G_2 = 1 \oplus 1 = 0$; $B_1 = B_2 \oplus G_1 = 0 \oplus 0 = 0$; $B_0 = 0 \oplus 1 = 1$

$\therefore$ Final Binary code $=$ ~~$(1000)_2$~~ $(1001)_2$.

⇒ **ASCII Code:** American Standard Code for Information Interchange

Uses small computers, peripheral device (input/o device), instruments, communication devices etc.

It is a 7-bit code. Microcomputers having 8-bit ~~length~~ word length use 7-bits to represent the basic code. The 8th bit is used for parity or it may be kept permanently 1 or 0.

Character	Zone	ASCII Code	Hexa equivalent
0 →	011	0000	30
1 →	011	0001	31
A	100	0001	41

Definition: An error detection code can be used to detect errors during transmission.

Error Detection Codes:

Binary information may be transmitted through some communication medium, e.g. using wires or wireless media. A corrupted bit will have its value changed from 0 to 1 or vice versa. To be able to detect errors at the receiver end, the sender sends an extra bit (parity bit) with the original binary message. A parity bit is an extra bit included with the n-bit binary message to make the total number of 1's in this message (including the parity bit) either odd or even. If the parity bit makes the total number of 1's an odd (even) number, it is called odd (even) parity. The table shows the required odd (even) parity for a 3-bit message:

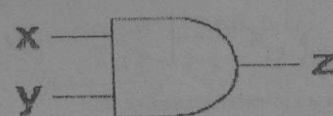
Three-Bit Message			Odd Parity Bit	Even Parity Bit
X	Y	Z	P	P
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

Basic Gate:

The basic building blocks of a computer are called logical gates or just gates. Gates are basic circuits that have at least one (and usually more) input and exactly one output. Input and output values are the logical values true and false. In computer architecture it is common to use 0 for false and 1 for true. Gates have no memory. The value of the output depends only on the current value of the inputs. A useful way of describing the relationship between the inputs of gates and their output is the truth table. In a truth table, the value of each output is tabulated for every possible combination of the input values. We usually consider three basic kinds of gates, and-gates, or-gates, and not-gates (or inverters).

- **The AND Gate:**

The AND gate implements the AND function. With the gate shown to the left, both inputs must have logic 1 signals applied to them in order for the output to be a logic 1. With either input at logic 0, the output will be held to logic 0.



The truth table for an and-gate with two inputs looks like this:

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

There is no limit to the number of inputs that may be applied to an AND function, so there is no functional limit to the number of inputs an AND gate may have. However, for practical reasons, commercial AND gates are most commonly manufactured with 2, 3, or 4 inputs. A standard Integrated Circuit (IC) package contains 14 or 16 pins, for practical size and handling. A standard 14-pin package can contain four 2-input gates, three 3-input gates, or two 4-input gates, and still have room for two pins for power supply connections.

- **The OR Gate:**

The OR gate is sort of the reverse of the AND gate. The OR function, like its verbal counterpart, allows the output to be true (logic 1) if any one or more of its inputs are true. Verbally, we might say, "If it is raining OR if I turn on the sprinkler, the lawn will be wet." Note that the lawn will still be wet if the sprinkler is on and it is also raining. This is correctly reflected by the basic OR function. In symbols, the OR function is designated with a plus sign (+). In logical diagrams, the symbol below designates the OR gate.

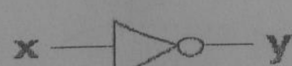
The truth table for an or-gate with two inputs looks like this:

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

As with the AND function, the OR function can have any number of inputs. However, practical commercial OR gates are mostly limited to 2, 3, and 4 inputs, as with AND gates.

- **The NOT Gate, or Inverter:**

The inverter is a little different from AND and OR gates in that it always has exactly one input as well as one output. Whatever logical state is applied to the input, the opposite state will appear at the output.



The truth table for an inverter looks like this:

x	y
0	1
1	0

The NOT function, as it is called, is necessary in many applications and highly useful in others. A practical verbal application might be:

The door is NOT locked = You may enter

In the inverter symbol, the triangle actually denotes only an amplifier, which in digital terms means that it "cleans up" the signal but does not change its logical sense. It is the circle at the output which denotes the logical inversion. The circle could have been placed at the input instead, and the logical meaning would still be the same.

Use of digital logic:

- Computing,
- Cell phones,
- Robotics, and
- Others electronics applications.

*** BCD to Decimal conversion:

BCD code: 4 bit represent.

Ex: $1 \rightarrow 0001$, $2 \rightarrow 0010$

$45 \rightarrow 0100\ 0101$

The weights of BCD is 8, 4, 2, 1.

(i) BCD to Decimal: $(0110)_2 = (?)_{10}$.

$\therefore = 0\times8 + 1\times4 + 1\times2 + 0\times1 = 0+4+2+0 = (6)_{10}$

(ii) $(1001)_2 = (?)_{10}$

$= 1\times8 + 0\times4 + 0\times2 + 1\times1 = 8+0+0+1 = 9$

*** BCD to Excess-3 Code:

Excess-3 Code is an unweighted code. Its code assignment is obtained from the corresponding value of BCD after the addition of 3.

Example: BCD-to-excess-3: (1) $(0000)_2$ (BCD) $= (?)_2$ (excess-3)

$\therefore$ We know: (BCD code + 3) = excess-3 code.

$\therefore$
```
  0000   (BCD code)
+ 0011   (3 add)
  ----
  0011
```

$\therefore (0000)_2$ (BCD) $= (0011)_2$ (ex-3)

(2) $(1001)_2$ (BCD) $= (\text{ex-3})_2$

$\therefore$
```
  1001   (BCD code)
+ 0011   (add 3 (0011))
  ----
  1100
```

$\therefore (1001)_2 = 1100$ (ex-3)

N.B.: 1. Weighted code = BCD code (8, 4, 2, 1)

2. Unweighted code = excess-3 code

Function of BCD to excess-3 Code Conversion:

Rule: $(ABCD)_{BCD}$ to $(wxyz)_{excess-3}$

$\therefore$

$$\begin{cases} W = A + BC + BD \\ X = B'D + B'C + BC'D' \\ Y = CD + C'D' \\ Z = D' \end{cases}$$

BCD code:

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

Example: Convert $(\underset{A\,B\,C\,D}{1001})_{BCD}$ to (W,X,Y,Z) excess-3 Code.

Ans:

$$\begin{aligned} W &= A + BC + BD \\ &= 1 + 0\cdot0 + 0\cdot1 \\ &= 1 + 0 + 0 \\ &= 1 \end{aligned}$$

$$\begin{aligned} X &= B'D + B'C + BC'D' \\ &= 1\cdot1 + 1\cdot0 + 0\cdot1\cdot0 \\ &= 1 + 0 + 0 \\ &= 1 \end{aligned}$$

$$\begin{aligned} Y &= CD + C'D' \\ &= 0\cdot1 + 1\cdot0 \\ &= 0 + 0 \\ &= 0 \end{aligned}$$

$$\begin{aligned} Z &= D' \\ &= 0 \end{aligned}$$

$\therefore$ Excess-3 Code of (1001) is : (1100) Ans