Module -4

Number System and codes

1. **Introduction**
2. **Number Systems**
   - 2.1. Decimal number system
   - 2.2. Binary number system
   - 2.3. Octal number system
   - 2.4. Hexadecimal number system
   - 2.5. Radix-n number system
3. **Number base inter-conversions**
   - 3.1. Decimal to binary & binary to decimal
   - 3.2. Decimal to Octal & octal to decimal
   - 3.3. Octal to binary & binary to octal
   - 3.4. Decimal to Hexadecimal & Hexadecimal to Decimal
   - 3.5. Hexadecimal to binary & Binary to Hexadecimal
4. **Binary Codes**
   - 4.1. BCD code
   - 4.2. Weighted code
   - 4.3. Self complementing codes
   - 4.4. Excess-3 code
   - 4.5. Cyclic code
   - 4.6. Gray code
   - 4.7. ASCII code
5. **Summary**

Learning objectives

1. Know common characteristics of all number systems.
2. Determine the weighting factor for each digit position in different number systems.
3. Convert any number in one of the number systems (decimal, binary, octal or hexadecimal) to its equivalent value in any other number systems.
4. Describe the format and use of BCD, Excess-3, Parity, Gray and ASCII codes

1. Introduction

Number systems provide a way of conveying and quantifying information. The study of number systems is one of the important topics in digital electronics so as to understand how information is represented. Many of the applications of digital electronics like computers do not process the information as either alphabets or decimal numbers but are designed using circuits that process binary digits- only 0 and 1.

In this module, emphasis is given on different number systems commonly used in data representation. The discussion begins with basic decimal number system and its relation to binary number system. The octal and hexadecimal numbers are described with special reference to their correlation to binary number system.

For understanding how computers and other digital systems communicate with the outer world, number base inter-conversions are covered in brief. The module also introduces some important binary codes used digital electronics like BCD**, Gray, Excess-3, Parity, ASCII,** error detection and correction codes.

2. Number systems

Numbers are everywhere. Universally we use the numbers in day to day life. We use numbers for mainly for counting and expressing the measured physical parameters in quantitative manner. The easiest way in which quantity of items can be expressed is to write a slash (/), a one (1) or some other symbol for each item of the group. For example, five items might be represented as '11111'. This scheme is useful for small number of items, say less than ten, it becomes extremely difficult to represent large quantities. Almost two pages would be required to represent 10000 items.

To overcome this problem of representing compactly large quantities of items, the earliest mathematicians devised a scheme of notation in which.

1. Many different symbols are used, other than just a slash or a one.
2. The position of the symbol with respect to fixed reference (e.g. Decimal point) decides the weight of the symbol. Such a scheme is referred to as **number system.**

Number system is system of an ordered set of symbols used to represent a quantity. These numbers are defined for basic mathematical operations. All modern number systems use radix or base representation, which came to us from Hindus via Arabia. The format is based on following ideas:-

1. The number of different symbols is restricted to the base (or radix) quantity.
2. A special symbol is used to represent nothing ('0' – Zero).
3. Every symbol has positional value in a number.
4. The position of the symbol within the number indicates the multiplication by a relevant power of the base.

Many number systems are in used in real world. Most common are the decimal, binary, octal and hexadecimal system. Let us start the discussion with the popular decimal number system.

2.1 Decimal Number System

Decimal number system is the commonly used number system. It is known as base (or radix) ten systems because it has ten digits 0 to 9 that can be used to represent any number. The widespread use of decimal number system is related to human anatomy i.e. passion of 10 fingers and their utility in counting. Figure 1 shows the positional values or weights of decimal number system.

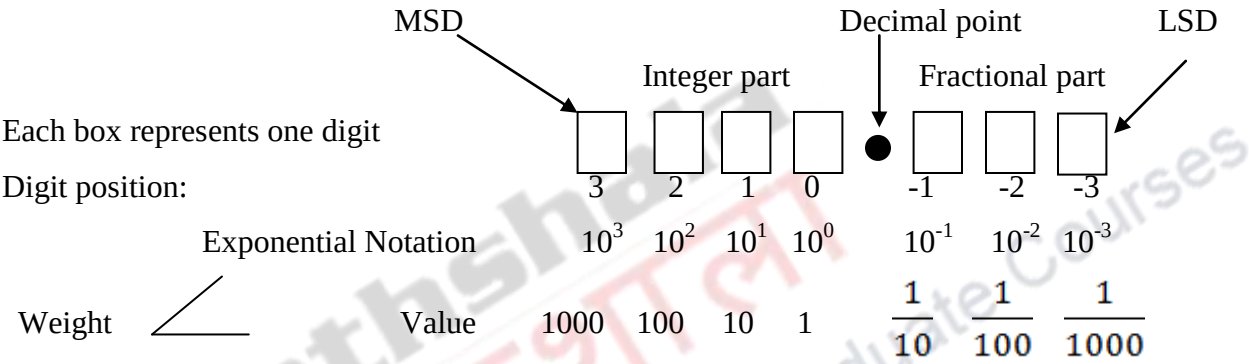


Figure 1 : Digit positions and their weights in decimal number system

A complete decimal number can be expressed in terms of Integer part and fractional with digit positions numbered with reference to decimal point. The weight of each digit increased by a factor of 10 for integer part and the weight of each digit decreases by a factor of 10. The weight of each digit is determined by its position. Thus an integer can be represented as the sum of weighted symbols. For example

$$234 = (2\times10^2) + (3\times10^1) + (4\times10^0)$$
$$= 200 + 30 + 4$$

Note that, the value of the left most digit is maximum, hence it is called as the Most Significant Digit (MSD). Similarly, the value of the right most digit has minimum value, hence it is referred to as Least Significant Digit (LSD).

For the representation of fractional part, it is necessary to represent using a reference point called decimal point. The fractional part is present to the right of the decimal point. Let us consider a representation of complete number

$$234.56 = (2\times10^2) + (3\times10^1) + (4\times10^0) + (5\times10^{-1}) + (6\times10^{-2})$$
$$= 200 + 30 + 4 + 0.5 + 0.06$$

The generalized polynomial notations for representation of decimal numbers are given by a sum of unspecified decimal coefficients weighted by position. Thus, in general any decimal number

'N' can be thought of as a sum of decimal coefficients (a_i) weighted by position (10^i) and is given by

$$N = \sum_{i=n}^{-m} a_i \, x \, 10^i$$

$$N = a_n \, x \, 10^n + a_{n-1} \, x \, 10^{n-1} \; \ldots + a_0 \, x \, 10^0 \; + a_{-1} \, x \, 10^{-1} + a_{-2} \, x \, 10^{-2} \; \ldots\ldots + a_{-m} \, x \, 10^{-m}$$

As there are ten symbols in the set of coefficients i.e. {0,1,2,3,4,5,6,7,8,9} , therefore the number 10 appears in the weighting factor. This number '10' is referred to as the '**radix'** or alternately as the **base,** of the decimal number system. The radix always tells us the number of unique symbols that may be used in a number system.

It is sometimes desirable, to utilize number systems that are other than radix-ten. In the next section, we shall discuss the importance and emergence of binary number system.

2.2 Binary Number System:-

Almost all computers and digital systems are based on binary (two-state) operation. The binary number system is a scheme of representing a number by using two symbols '0' and '1'. It is a radix-2 or the base 2 number system. A single binary digit is usually referred as a **bit (bi**nary digi**t).**

Just similar to decimal number system, binary number system is a positional number system. The position of bit in a binary number system indicates its weight or value within the number. The weight of each bit increases by a factor of 2 for each successive higher position and decreases by a factor of 2 with successive lower positions as shown in fig.2.

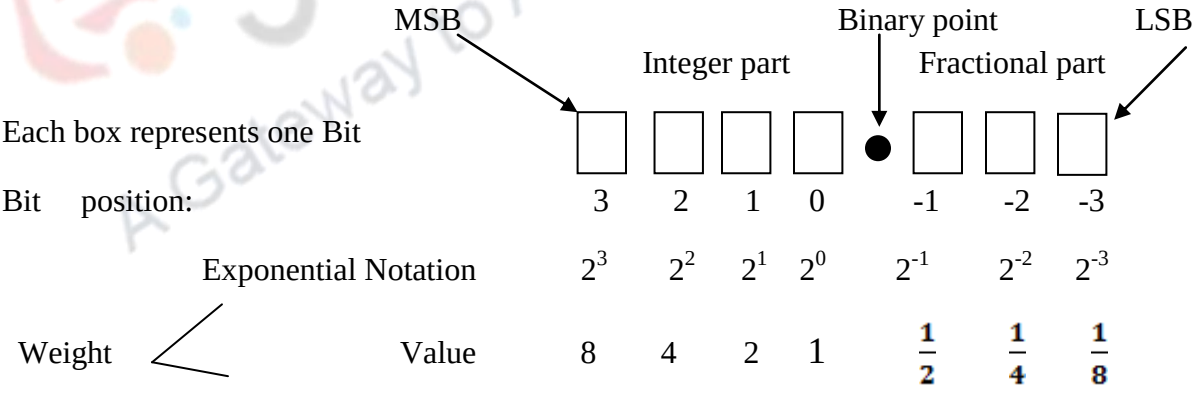


Figure 2: Bit positions and their weights in binary number system

Thus, in general any binary number 'N' can be thought of as a sum of binary coefficients (a_i) weighted by position (2^i) and is given by

$$N = a_n \, x \, 2^n + a_{n-1} \, x \, 2^{n-1} \; \ldots + a_0 \, x \, 2^0 \; + a_{-1} \, x \, 2^{-1} + a_{-2} \, x \, 2^{-2} \; \ldots\ldots + a_{-m} \, x \, 2^{-m}$$

For example,

$$(1101.11)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$$

Table 1 gives positive and negative powers of two.

Table -1: Positive and negative powers of two.

Positive power of 2	Negative power of 2
$2^0=1$	-
$2^1=2$	$2^{-1}= 0.5$
$2^2=4$	$2^{-2}= 0.25$
$2^3=8$	$2^{-3}= 0.125$
$2^4=16$	$2^{-4}= 0.0625$
$2^5=32$	$2^{-5}=0.03125$
$2^6=64$	$2^{-6}= 0.015625$
$2^7=128$	$2^{-7}= 0.0078125$
$2^8=256$	$2^{-8}=0.00390625$
$2^9=512$	$2^{-9}= 0.001953125$
$2^{10}=1024 \sim 1$ **Kilo**	$2^{-10} \approx 1$ **mili**
$2^{20} \sim 1$ **Mega**	$2^{-20} \approx 1$ **micro**
$2^{30} \sim 1$ **Giga**	$2^{-30} \approx 1$ **nano**
$2^{40} \sim 1$ **Tera**	$2^{-40} \approx 1$ **pico**

The leftmost bit has highest weight in binary number and hence it is known as Most Significant Bit (MSB). The rightmost bit has the lowest weight and is referred to as Least Significant Bit (LSB).

There are number systems other than binary and decimal, such as octal or Hexadecimal number systems. Usually, we find difficult to work with binary numbers because they are very long while representing larger decimal quantities. Therefore, Octal and hexadecimal, numbers are used to compress long strings of binary numbers. Let us discuss about these number systems.

2.3 Octal number system

The octal number system has eight unique symbols which are 0,1,2,3,4,5,6,and 7. As there are eight symbols, the number system has a radix of 8. It is also referred to as base 8 number system. Since, it has base $8=2^3$, every group of 3-bits is represented by an octal digit. For example,
$(5)_8 = (101)_2$ or $(7)_8 = (111)_2$ or $(27)_8 = (010111)_2$

In octal number system, the weight of each digit position increases by the factor of 8 (8, 16, 64,……) as we move towards right.

When a position digit crosses the value represented by symbols for that base, a carry into the next higher position is produced.

For example: $(234)_8 = (2\times8^2) + (3\times8^1) + (4\times8^0)$

2.4 Hexadecimal number system

The hexadecimal number system has sixteen symbols which are 0 to 9, A, B,C,D,E and F. As there are sixteen symbols, the number system has a radix of 16. It is also referred to as base 16 number system. Since, it has base $16=2^4$, **every group of 4-bits is represented by a hexadecimal digit**. For example,

$(6)_{16} = (0110)_2$

or $(E)_{16} = (1110)_2$

or $(7F)_{16} = (0111\ 1111)_2$

When a position digit crosses the value represented by symbols for that base, a carry into the next higher position is produced.

For example: $(234)_{16} = (2\times16^2) + (3\times16^1) + (4\times16^0)$

Many digital systems process binary data in groups that are multiples of four bits. Hexadecimal number system is widely used in microprocessors, since they are shorter than binary. This makes them easy to write and remember. Furthermore, you can convert them to binary whenever necessary. It is easy to write the hexadecimal number than binary number.

2.5 Radix-n number system

In radix-n number system, the numbers of symbols are restricted to value n. It can be any number other than decimal, binary, octal and hexadecimal.Table-2 indicates Radix and number of symbols.

Table-2: Radix and numbers of symbols

Radix (base)	Symbols
2 (Binary)	0,1
3	0,1,2
4	0,1,2,3
5	0,1,2,3,4
6	0,1,2,3,4,5
7	0,1,2,3,4,5,6
8 (Octal)	0,1,2,3,4,5,6,7
9	0,1,2,3,4,5,6,7,8
10 (Decimal)	0,1,2,3,4,5,6,7,8,9
……	
…….	
16 (Hexadecimal)	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

A number represented in Radix-n system, has n symbols in the system. Let us understand counting in the Radix-n system. Table-3 indicates counting in four popular number systems.

Table-3: Counting in different number systems.

Decimal	Binary	Octal	Hexadecimal	Radix-5	Radix-12
0	0000	0	0	0	0
1	0001	1	1	1	1
2	0010	2	2	2	2
3	0011	3	3	3	3
4	0100	4	4	4	4
5	0101	5	5	10	5
6	0110	6	6	11	6
7	0111	7	7	12	7
8	1000	10	8	13	8
9	1001	11	9	14	9
10	1010	12	A	20	A
11	1011	13	B	21	B
12	1100	14	C	22	10
13	1101	15	D	23	11
14	1110	16	E	24	12
15	1111	17	F	30	13

To learn to count in binary, let us first consider the problem of counting a quantity of items in decimal. Let us start at 0 and count up to 9 before we run out of digits. Then start with another digit position to the left and continue counting 10 through 19, then change the symbol at left and continue counting 20 through 29 and so on. Once we reach at 99, all the two digit combinations are exhausted, so a third digit is needed in order to count from 100 through 999.

A similar situation occurs while counting in binary, except that there are only two symbols. Let us begin counting 0,1. Let us include another digit position and continue 10, 11. With three digits, the count continues 100, 101, 110, and 111 as shown in Table-3. Similar counting process is repeated for Radix-n number system.

As the binary symbols 0 and 1 are exactly identical to first two decimal symbols, it is possible that confusion may result in writing of a number. For example, when a number 100 is written, are we expressing quantity four or hundred? Of course, that depends on the number system that is being used. To avoid confusion between different number systems, a number is enclosed in a parentheses and a subscript is written to indicate the radix.

e.g. $(100)_2$ = Four, $(100)_{10}$ = Hundred or $(1111)_2$ = $(15)_{10}$ etc.

3. Number Base Interconversions:-

Modern computers do not process decimal numbers. Instead they work with binary numbers, which use only the digit 0 and 1. Hence, we shall learn how to convert from decimal to binary, octal, hexadecimal and vice versa. People do not like working with binary numbers because they are very long when representing larger decimal quantities. Therefore, Octal, hexadecimal, numbers are used to compress long strings of binary numbers. Hence, we shall learn interconversions from one system to another as shown in fig. 3

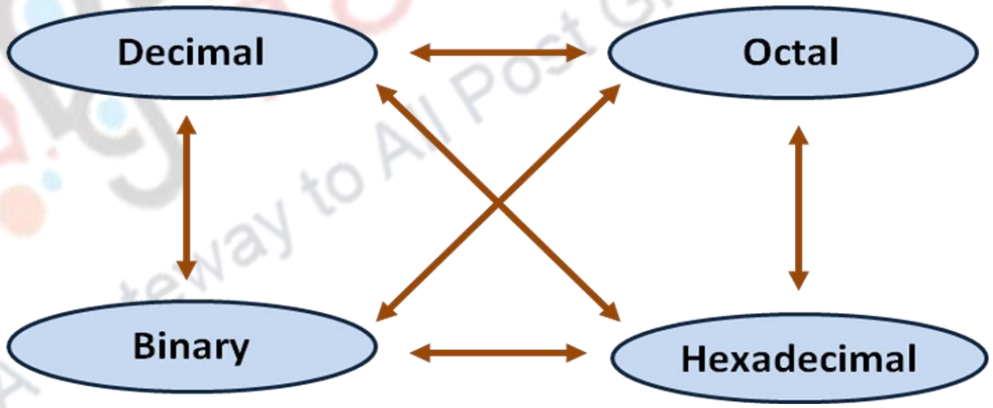


Figure-3: Possible number base interconversions

We will learn following interconversions

(1) DEC to BIN
(2) BIN to DEC
(3) DEC to OCT
(4) OCT to DEC
(5) OCT to BIN
(6) BIN to OCT
(7) DEC to HEX
(8) HEX to DEC
(9) HEX to BIN
(10) BIN to HEX

3.1 Decimal to binary conversions (DEC to BIN)

There are two ways of converting a decimal number into a binary number. (a) Sum-of-weights method and (b) Double-dabble method

(a) Sum-of- weights method

The following method can be adopted

(i) Get the decimal number
(ii) Write the weights in exponential form or/and its value.
(iii) Determine the set of binary weights whose sum is equal to decimal number.
(iv) Place 1 and a 0 at appropriate positions i.e. place a 1 if weight is taken during the sum and 0 otherwise.

Example: Convert decimal number 9 into its binary equivalent.

Solution : 8 4 2 1

$(9)_{10} = 8 + 0 + 0 + 1$

$(9)_{10} = (1001)_2$

Example: Convert decimal numbers 25, 58 and 255 into their binary equivalent.

Solution : $(25)_{10} = (11001)_2$

$(58)_{10} = (111010)_2$

$(255)_{10} = (1111\ 1111)_2$

(b) Double-dabble method :

It is a popular and most systematic way to convert decimal numbers to binary numbers. In this method a repeated division-by-2 for integers and repeated multiplication-by-2 for fractional part is used. The best way to understand the method is to follow the procedure step-by-step.

Double dabble procedure for integers –

(i) Get the decimal number.
(ii) Divide the decimal number by 2 and write quotient and remainder. This remainder is the LSB (Least significant bit).
(iii) Again divide the quotient obtained above by 2. This produces another quotient and remainder. The remainder is the next digit of the binary number.
(iv) Continue this process of division until the quotient becomes 0. The remainder obtained in the final division is the MSB (most significant bit).
(v) Write the remainders in reverse order i.e. first remainder is LSB and last reminder is MSB.

Example : Convert decimal number 19 into its binary equivalent.

Solution : Proceed as follows

		Quotient	Remainder	
2	19	9	1	**LSB**
2	9	4	1	
2	4	2	0	
2	2	1	0	
2	1	0	1	**MSB**

Thus $(19)_{10} = (10011)_2$

Double –dabble procedure for fractional part

(i) Get the fractional part

(ii) Multiply the decimal number by 2 and write product and carry. Note that product should be in fractions, integer part be transferred to carry. This carry is the MSB (most significant bit).

(iii) Multiply each fractional part of product by 2 until the fractional product is zero.

(iv) The carries generated by each multiplication form the binary number. The first carry produced is the MSB and the last one is the LSB.

Example : Convert decimal number 0.625 into its binary equivalent.

Solution :

	fractional product	Carry	
$0.625 \times 2 =$	1.25	1	MSB
$0.25 \times 2 =$	0.50	0	
$0.50 \times 2 =$	1.00	1	LSB

Thus, $(0.625)_{10} = (0.101)_2$

Note : If necessary repeated multiplication process can be continued further where greater accuracy is desired in case of nonzero product.

Binary to Decimal conversion (BIN to DEC) :

The decimal equivalent of binary number is obtained by using the fact that binary number is a weighted number. Hence we shall use once again sum- of –weights method, which is described below.

(i) Write the binary number.
(ii) Write the weights of each bit under the binary number.
(iii) If a zero appears in a bit position, delete the decimal equivalent.

Example :- Convert binary number 0101 into its decimal equivalent.

Solution : - 0 1 0 1

$\cancel{8} + 4 + 2 + \cancel{1} = 5$

$(0101)_2 = (5)_{10}$

3.2 Decimal to octal conversion (DEC to OCT) :

The repeated division-by-8 for integer and repeated multiplication-by-8 for fractions are used for decimal to octal conversion, which is similar to the method used in conversion of decimal numbers into binary numbers (also referred as octal dabble method).

Example : Convert decimal number 127 into its octal equivalent.

Solution :

	Quotient	Remainder	
$127 \div 8$	15	7	**LSB**
$15 \div 8$	1	7	
$1 \div 8$	0	1	**MSB**

Thus $(127)_{10} = (177)_8$

$(175)_{10} = (257)_8$

Octal to Decimal conversion (OCT to DEC)

Conversion from octal to decimal can be done using sum of weighted numerical symbols. Following procedure may be adopted:

(i) Write the octal number.
(ii) Write the weights of each symbol under that digit.
(iii) Multiply each digit by its positional weight and then add up to obtain equivalent.

Example : Convert octal number 23 to its decimal equivalent.

Solution : $(23)_8 = 2x8^1 + 3x8^0$

$= 2x8 + 3x1$

$= 16 + 3 \quad = (19)_{10}$

3.3 Octal to binary conversion (OCT to BIN) :

The prime application of octal numbers is in the representation of binary number. Since it takes only one digit to represent three bits, octal numbers are much easier to **'READ'** than binary numbers.

Following procedure may be adopted.

(i) Write the octal number.

(ii) For each octal digit write the corresponding 3-bit binary equivalent.

(iii) The binary number may be written without a gap.

Example : Convert octal number 23 to its binary equivalent.

Solution :

2 3

010 011

Thus $(23)_8 = (010011)_2$

Binary to octal conversion (BIN to OCT):

Converting a binary number to octal is a straightforward procedure, as described below,

(i) Beginning at a binary point, simply separate the binary number into groups of three bits towards both left as well as right.

(ii) To complete a group, add a bit 0 if necessary.

(iii) Replace each group of three bits with its octal equivalent.

Example : Convert binary number 11010 into its octal equivalent.

Solution :- 11010 $\cong$ 011 010 =32

Thus $(11010)_2 = (32)_8$

3.4 Decimal to Hexadecimal conversion (DEC to HEX)

One way to convert decimal number to Hexadecimal number is the use of Hex-dabble method. The idea is to divide integers successively by 16 and multiply fractions repetitively by 16.

Example : Convert decimal number 650 into its hexadecimal equivalent.

	Quotient	**Remainder**	
$650 \div 16$	40	10 (A)	**LSB**
$40 \div 16$	2	8	
$2 \div 16$	0	2	**MSB**

Thus $(650)_{10} = (28A)_{16}$

Hexadecimal to Decimal conversion (HEX to DEC)

In hexadecimal number system each digit position corresponds to appropriate power of 16. The weights of the digit positions in hexadecimal number are as follows:

$\ldots\ldots 16^3 \;\; 16^2 \;\; 16^1 \;\; 16^0 \;.\; 16^{-1} \;\; 16^{-2} \;\; 16^{-3}$

To convert Hexadecimal number to decimal number, multiply each hexadecimal digit by its weight and add the resulting product.

Example : Convert hexadecimal number 1C into its decimal equivalent.

Solution :- $1C = 1x16^1 + Cx16^0$

$= 1x\ 16 + 12x1$

$= 16 + 12 = (28)_{10}$

3.5 Hexadecimal to binary conversion (HEX to BIN) :

To convert a hexadecimal number to a binary number, first write each hexadecimal digit to its 4-bit equivalent using the binary codes, and then write the binary numbers without a gap.

Example : Convert $(9A)_{16}$ to binary equivalent.

Solution :- (9 A)

(1001 1010) $_2$

Thus $(9A)_{16} = (1001\ 1010)_2$

Binary to Hexadecimal conversion (BIN to HEX) :

Converting a binary to hexadecimal is a straight forward procedure. Simply break the binary number into four-bit groups starting at binary point, and replace each group with the equivalent hexadecimal symbol.

Example : Convert $(101101)_2$ to Hexadecimal equivalent.

Solution:- $\frac{10}{2} \;\; \frac{1101}{D}$

Thus $(10\ 1101)_2 = (2D)_{16}$

4. Binary codes

There are several binary codes like weighed, un-weighted, self-complementing, cyclic etc. In this section, let us begin with the simplest BCD code.

4.1 Binary Coded Decimal (BCD) Code:-

The binary coded decimal (BCD) code is a weighted code. This code is found very convenient for representing digits. Each group of four bits is used to represent one decimal digit. It is also called as 8421 code. This code consists of four bits which have the weights as 8 4 2 1 . The four bit combination that represents the decimal digits 0 to 9 are shown in table 1.4.

Table -4: The 8421 BCD code

Decimal digit	8421 (BCD) code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	0001 0000
11	0001 0001

It is realized that with four bits, sixteen different symbols (2^4) can be represented, but in 8421 BCD code only ten of these are used. The remaining six code combinations are invalid in the 8421 BCD code.

To express any decimal number in BCD code simply replace each decimal digit by the appropriate four-bit code. Let us consider the decimal number 23,

Thus $(23)_{10} = (0010\ 0011)_{BCD}$

Advantages of BCD code over binary are that it is very simple. We do not have to perform successive division by two and write the remainder. We simply have to know the codes for the ten basic decimal digits and then we can convert any decimal number into its equivalent BCD code. The disadvantages are that it requires large number bits than the binary equivalent. BCD arithmetic implementation is difficult as compared to binary.

The BCD code has both a code and a direct binary conversion as long as the decimal numbers are integers from 0 to 9. For number greater than 9, the conversion and coding are completely different. In the next section, we shall discuss other binary codes like excess code and Gray Code.

4.2 Weighted code

In the weighted codes, the weights or values are assigned to the binary bits as per their bit positions. The decimal value of the code is the algebraic sum of weighted digits. In other words, the decimal number in weighted code can be written as the sum of the products of weight and the number itself.

Most popular weighted codes are 8421, 5421, 7421, 2421, 5211 etc. Table – 5 shows the representation of decimal number in various weighted codes.

Table-5: Weighted codes

Decimal number	Weighted codes				
	8421	5421	7421	2421	5211
0	0000	0000	0000	0000	0000
1	0001	0001	0001	0001	0001
2	0010	0010	0010	0010	0011
3	0011	0011	0011	0011	0101
4	0100	0100	0100	0100	0111
5	0101	1000	0101	1011	1000
6	0110	1001	0110	1100	1010
7	0111	1010	1000	1101	1100
8	1000	1011	1001	1110	1110
9	1001	1100	1010	1111	1111

The 8421 code utilizes the natural weights for the representation of the binary numbers hence it is popularly known as natural binary coded decimal (NBCD).

4.3 Self-complementing code

A code is said to be self complementing if the binary representation of a decimal number (N) in that code is 1's complement of the decimal number (9-N). Let us consider N=6. The weighted code is said to be self complementing if N and (9-N) are 1's complement of each other. Here it is 9's complement. The 9's complement of 6 is 3.

Such codes have the property that the 9's**complement** of a decimal number is obtained directly by changing 1's to 0's and 0's to 1's (i.e., by **complementing** each bit in the pattern)

The weighted codes 2421 and 5211 are self complementing codes whereas 8421,5421,7421 are not self complementing code. This could be easily verified using the Table-5.

4.4 Excess-3 Code :

Excess-3 Code is a unweighted code, as no definite weights are assigned to bit positions. This code is sometimes used with BCD code. As the name suggests this code is obtained by adding 3 to each decimal digit and then converting it into binary. Table-6 illustrates the excess-3 code.

Table-6: Excess-3 Code

Decimal Digit	BCD Code	Excess-3 Code
0	0000	0011
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

For example, convert a decimal number 4 into Excess-3 code. BCD code is 0100 & excess-3 code is 0111.

The key feature of excess-3 code is that it is self-complementing. That is, the 1's complement of an excess-3 code is the excess-3 code for the 9's complement of the corresponding decimal number. The 9's complement of a decimal digit is obtained by subtracting that digit from number 9. Thus 9's complement of 3 is 6 and that of 2 is 7.

The excess-3 code finds applications in arithmetic operations as it greatly simplifies the process.

4.5 Cyclic code

While discussing cyclic codes, it is first necessary to understand the concept of Hamming distance. Hamming distance is defined as the number of places the binary bits differ in two consecutive number in a particular code. For example in 8421 code, hamming distance between 0 (0000) and 1 (0001) is one, as there is a change of one bit position at LSB from 0 to 1. Similarly the Hamming distance between 1 and 2 is 2. Similarly Hamming distance between 3 and 4 is 3. Thus, one can say that the Hamming distance between two successive codes of 8421 code is not constant.

All cyclic codes have a unit Hamming distance property. Gray code is an example of cyclic code. It is normally used in rotary encoders, ADCs etc.

4.6 Gray Code :

The Gray code is an unweighted code as no weights are assigned to bit positions. Hence, this code is not used in arithmetic operations. This code is often associated with optical encoders, a technique for converting the shaft angle into binary value. This is also referred to as **cyclic code.**

The Gray code is basically ***a unit distance code,*** where the distance between consecutive code words is constant. The Gray code exhibits only a one-bit change between two consecutive codes (unit Hamming distance). Table-7 gives a comparison of binary and Gray codes. Gray code can have any number of bits like binary.

Table-7: Four bit Gray Code

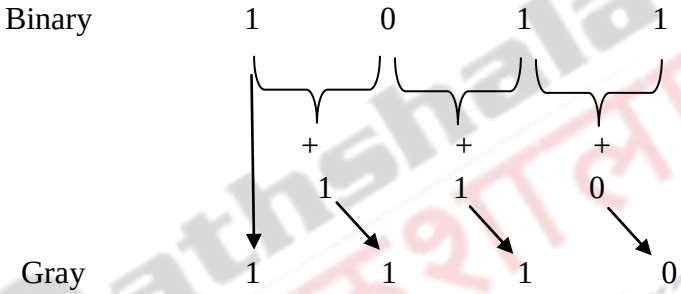
Decimal Digit	Binary	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110

12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Let us understand the procedure how Gray code is generated from binary numbers.

1. Write the binary number.
2. The MSB in Gray code is the same as in binary number.
3. Going from left to right, add each adjacent pair of binary digit to get the next Gray code digit. Discard the carries.

Let us consider the binary number 1011 to Gray code conversion,



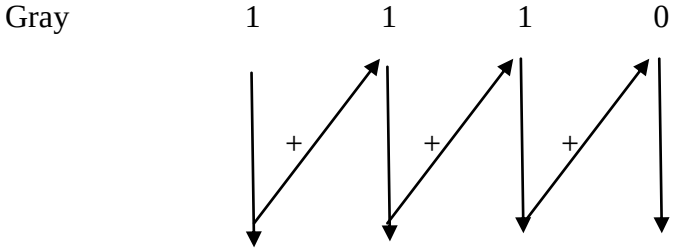
Thus binary number 1011 is converted to gray code as 1110.

Let us understand the procedure how the Gray code is converted to a binary numbers.

1. Write the Gray code.
2. The MSB of binary is the same as in Gray code.
3. Going from left to right, add the MSB bit of the result to the next adjacent bit of Gray code to get the next bit of binary. Discard the carries. The process is continued till the LSB is obtained.

Let us consider the binary number 1011 to Gray code conversion,

Let us convert Gray code to binary



Binary	1	0	1	1

Let us now discuss the popular ASCII code in next section.

4.7 Alphanumeric Codes :

All computers handle alphabets and special characters besides decimal numbers. These are normally the 26 English alphabets, the ten decimal digits and several special characters such as +,-,*,/, etc. In order to code these binary numbers, a string of binary digits is used. In modern digital computers, the number of characters has increased. Besides capital letters of the alphabet, the lower case letters are also used and several mathematical symbols such as <,> ,etc. have been introduced. In such system, new coding schemes use 7 or 8 bits to code a character.

7-bit ASCII Code

The American Standard Code for Information Interchange (ASCII) code, and is extensively used for digital computers and data communication. It is a 7-bit code. It can represent 2^7 i.e. 128 different characters. It represents decimal numbers 0 to 9 by 8421 BCD code preceded by 011. The letters of the alphabet in both uppercase as well as in lowercase and other symbols, special characters and instructions for communication (or commands) are also included. The 7-bit code for each symbol has two groups of bits-3 bit group ($X_6X_5X_4$) and 4 bit group ($X_3X_2X_1X_0$).

Definitions of Control Abbreviations

ACK	Acknowledge	BEL	BELL:
BS	Backspace	CAN	Cancel
CR	Carriage Return	DC_1-DC_4	Direct control
DEL	Delete idle	DLE	Data Link Escape
EM	End of Medium	ENQ	Enquiry
EOT	End of Text	ESC	Escape
ETB	End of Transmission Block	ETX	End Text
FF	Form Feed	FS	Form Separator
GS	Group Separator	HT	Horizontal Tab
LF	Line Feed	NAK	Negative Acknowledge
NUL	Null	RS	Record Separator
SI	Shift In	SO	Shift Out
SOH	Start of Heading	STX	Start Text
SUB	Substitute	SYN	Synchronous
US	Unit Separator	VT	Vertical Tab

For example, decimal number 5 is represented by 011 0101 (35) the letter A by 100 0001 (41) and the letter m by 110 1101 (60). Note the numbers in parentheses represents hexadecimal digits.

A parity-bit can be added to the 7-bit ASCII Character Code to generate 8-bit code. A group of 8-bits is known as byte. .

The 8-bit ASCII code is basically an extension of 7-bit ASCII code. The ASCII code for alphabets and numerals are just the same. In addition to 128 symbols of 7-bit ASCII code, there are several symbols used for generated Tables and graphics symbols along with special mathematical symbols. These are additional 128 symbols. There are total 256 symbols in this ASCII code.

5. Summary

- Numbers help us to quantify a given item with the help of different symbols.
- Numeric quantities occur naturally in analog form but must be converted to digital form to be used by computer or digital circuitry.
- Decimal number system is the oldest and most commonly used number system. It uses ten different symbols 0 to 9 and has radix or base 10.
- Binary number system has only two symbols 0 and 1. It has radix or base 2. This system is most suitable for digital systems and computers, because 1's and 0's are easily represented by ON and OFF transistors.
- Octal system has eight symbols 0 to 7 and has radix or base 8. It is used as three-bit binary code.
- Hexadecimal system has sixteen different symbols- number 0 to 9 and alphabets A to F. It is used in microprocessors and microcomputers as a substitute for binary numbers. It is convenient for us as long strings of binary 0's and 1's are difficult to handle.
- Any number in one system can be converted into another system provided the weights and symbols are properly known, BCD is a four-bit binary code used to represent decimal digits. Each decimal digit has a unique BCD equivalent. Writing a decimal number into its code is faster than converting it to binary.
- Excess-3 and Gray codes are other binary codes used to represent decimal numbers.
- Excess-3 code is the self-complementary code and used for arithmetic operations.
- Gray code is a cyclic code. Its main feature is the single-bit change going from one number in sequence to the next.
- The ASCII (American Standard Code for Information Interchange) is a seven-bit alphanumeric code. The code is most commonly used in data communications and computers, to represent all letters, numbers and symbols, in digital form