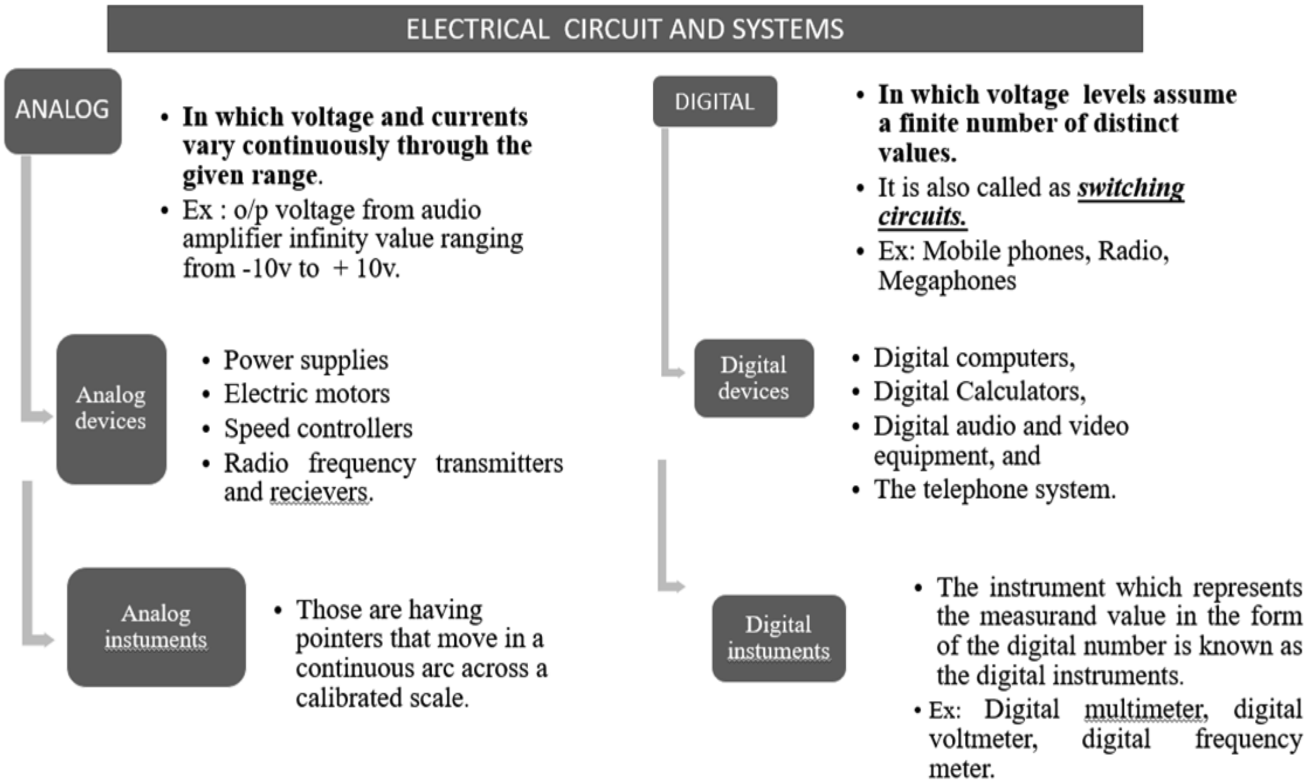
MODULE 1

Number systems& Binary codes:

- Number systems: Number Systems, Radix conversions, complement of numbers.
- Binary codes: Binary codes, Weighted and non-Weighted codes, BCD code, gray code, excess 3codes - Error detecting code, Error Correcting code, Hamming Code

INTRODUCTION



DIGITAL CIRCUIT:

- Digital circuit is one in which the voltage levels assume a finite number of distinct values.
- Each voltage level in a practical digital system can actually be a narrow band or range of voltages.
- Also called as switching circuits, the voltage levels in a digital circuit are assumed to be switched from one value to another value instantaneously, that is the transition time is assumed to be zero.
  - A) COMBINATIONAL SWITCHING CIRCUITS:
    - The output depends only on the present inputs.
    - They have no memory.
  - B) SEQUENTIAL SWITCHING CIRCUITS
    - The output depends on the present inputs as well as the present state of the circuit, i.e., on the past values also.
    - These are combinational circuits with memory.

❖ SEQUENTIAL SWITCHING CIRCUITS:

a) SYNCHRONOUS SEQUENTIAL CIRCUITS.: Digital sequential circuits in which the feedback to the input for next output generation is governed by clock signals.

b) ASYNCHRONOUS SEQUENTIAL CIRCUITS: Digital sequential circuits in which the feedback to the input for next output generation is not governed by clock signals.

DIGITAL CIRCUIT is also called as Binary signals or Logic signals.

- The digital signals are represented by two voltage bands, one band which is near a reference value (generally 0), and the other band lies near the supply voltage.

- This is similar to the values, ‘0’ and ‘1’ or ‘false’ and ‘true’ of the Boolean domain.
- This means that at any particular time, a digital signal can represent only one binary digit.
- The manner in which a logic circuit responds to an input as referred to as the circuit logic.

Application:

- Thermometer, photocopies, landline telephones, audiotape recorders, television, computers, laptops, mobile phones, wristwatches, wall clocks, are all becoming digital nowadays.
- It increases the accuracy of the message as well as makes it easy to read.

Advantages of Digital system or signals:

- Because of the digital nature, the signals in the digital systems can travel significantly faster over digital lines as compared to the Analog signals
- As compared to Analog signals, digital signals can transfer more data.
- The digital systems are less expensive, more reliable, easy to manipulate, and more flexible as compared to the Analog system.
- A digital system can be made compatible with other digital systems to which is not possible in the Analog system.

1. THE DECIMAL SYSTEM

The decimal number system comprises digits from 0-9 that are 0, 1, 2, 3, 4, 5, 6, 7, 8 & 9. The base or radix of the decimal number system is 10 because the total number of digits available in the decimal number system is 10. All the other digits can be expressed with the help of these 10 digit numbers.

number **345** represents:

$$= 3 * 10^2 + 4 * 10^1 + 5 * 10^0$$
$$= 3 * 100 + 4 * 10 + 5$$
$$= 300 + 40 + 5$$
$$= 34$$

the value **123.456** means:

$$= 1 * 10^2 + 2 * 10^1 + 3 * 10^0 + 4 * 10^{-1} + 5 * 10^{-2} + 6 * 10^{-3}$$
$$= 100 + 20 + 3 + 0.4 + 0.05 + 0.006$$

2. THE BINARY SYSTEM

Binary number system can be said to be the simplest one in the number system. It uses only two digits (0 and 1) to represent a number. Thus, as the ‘bi’ in its name suggests, the system uses 2 as a base. The entire number system can be represented through the binary system. For example, fractions, real numbers, as well as large numbers, can be represented through binary numbers.

BINARY TO DECIMAL CONVERSION

The binary numbering system works just like the decimal numbering system, with two exceptions:

- binary only allows the digits 0 and 1 (rather than 0–9), and
- binary uses powers of two rather than powers of ten.

Therefore, it is very easy to convert a binary number to decimal. For each "1" in the binary string, add 2n where "n" is the bit position in the binary string (0 to n–1 for n bit binary string).

For example, the binary value 1010_2 represents the decimal 10 which can be obtained through the procedure shown in the table 1:

Table 1

Binary No.	1	0	1	0
Bit Position (n)	3rd	2nd	1st	0th
Weight Factor (2n)	2^3	2^2	2^1	2^0
bit * 2n	$1*2^3$	$0*2^2$	$1*2^1$	$0*2^0$
Decimal Value	8	0	2	0

Decimal Number 8 + 0 + 2 + 0 = 10

All the steps in above procedure can be summarized in short as

$1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 = 8 + 0 + 2 + 0 = 1010$

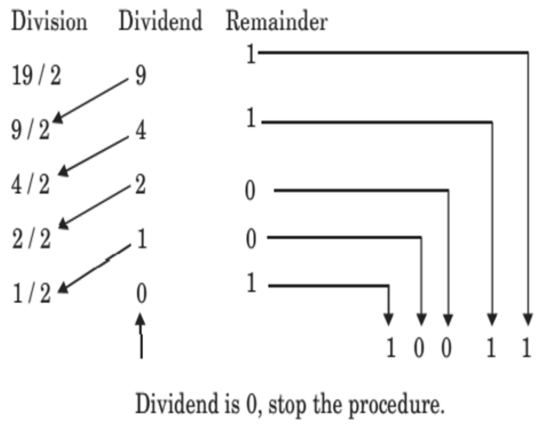
i.e.,

1. Multiply each digit of the binary number by its positional weight and then add up the result.

2. If any digit is 0, its positional weight is not to be taken into account.

DECIMAL TO BINARY CONVERSION

let us find out binary of 19_{10} (decimal 19).



1. The right most bit in a binary number is bit position zero.
2. Each bit to the left is given the next successive bit number.

- An eight-bit binary value uses bits zero through seven:
- X7 X6 X5 X4 X3 X2 X1 X0
- A 16-bit binary value uses bit positions zero through fifteen:

X15 X14 X13 X12 X11 X10 X9 X8 X7 X6 X5 X4 X3 X2 X1 X0

- Bit zero is usually referred to as the ***low order bit.***

or

Least significant bit (LSB).

- The left-most bit is typically called the ***high order bit***.

or

Most significant bit (msb)

3. OCTAL NUMBERING SYSTEM:

- The octal number system uses base 8 instead of base 10 or base 2.
- This is sometimes convenient since many computer operations are based on bytes (8 bits). In octal, we have 8 digits at our disposal, 0–7.

DECIMAL	OCTAL
0	0

1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	10
9	11
10	12
11	13
12	14
13	15
14	16
15	17
16	20

Octal to Decimal,

Converting octal to decimal is just like converting binary to decimal, except instead of powers of 2, we use powers of 8.

To convert 172 in octal to decimal:

$$\begin{array}{lcl} & & 1 \quad 7 \quad 2 \\ & & 8^2 \quad 8^1 \quad 8^0 \\ \text{Weight} & = & 1*8^2 + 7*8^1 + 2*8^0 \\ & = & 1*64 + 7*8 + 2*1 \\ & = & 122_{10} \end{array}$$

Decimal to Octal Conversion,

Converting decimal to octal is just like converting decimal to binary, except instead of dividing by 2, we divide by 8.

To convert 122 to octal:

$$\begin{array}{rcl} 122/8 & = & 15 \text{ remainder } 2 \\ 15/8 & = & 1 \text{ remainder } 7 \\ 1/8 & = & 0 \text{ remainder } 1 \\ & = & 172_8 \end{array}$$

Octal to binary

Convert $(145056)_8$ to binary.

To convert from octal to binary and vice versa we will need this conversion table.

value $(145056)_8$ can be converted to binary as $(001\ 100\ 101\ .\ 101\ 110)_2$

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

Binary to Octal:

We can use the same table to convert a binary number to octal number. And for that, we first have to group the binary number into a group of three bits and write the octal equivalent of it.

Convert the binary number $(11001111)_2$ to octal
The three bit group of binary numbers can be written as 011,001,111 because we have to add a zero before each number to complete the in the form of three binary digits. Therefore, the octal numbers will be 3, 1, 7 i.e., $(317)_8$

3. HEXADECIMAL NUMBERING SYSTEM

- Hexadecimal uses a base 16 numbering system. This means that we have 16 symbols to use for digits. Consequently, we must invent new digits beyond 9.
- The digits used in hex are the letters A, B, C, D, E, and F.

Hexa decimal to Decimal

Converting hex to decimal is just like converting binary to decimal, except instead of powers of 2, we use powers of 16.

To convert 15E in hex to decimal:

$$\begin{array}{ccc} 1 & 5 & E \\ 16^2 & 16^1 & 16^0 \end{array}$$

$$\begin{aligned} \text{Weight} &= 1*16^2 + 5*16^1 + 14*16^0 \\ &= 1*256 + 5*16 + 14*1 \\ &= 350_{10} \end{aligned}$$

Decimal to Hex Conversion

Converting decimal to hex is just like converting decimal to binary, except instead of dividing by 2, we divide by 16. To convert 350 to hex:

350/16 = 21 remainder 14 = E
21/16 = 1 remainder 5
1/16 = 0 remainder 1

Decimal	Hexa decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Hexa to Octal

A group of 4-bits represent a hexadecimal digit and a group of 3-bits represent an octal digit.

1. Convert the given hexadecimal number into binary.
2. Starting from right make groups of 3-bits and designate each group an octal digit.

. Convert $(1A3)_{16}$ into octal.

1. Converting hex to binary

$$(1\ A\ 3)_{16} = \underbrace{0001}_{1}\ \underbrace{1010}_{A}\ \underbrace{0011}_{3}$$

2. Grouping of 3-bits

$$(1A3)_{16} = \underbrace{000}_{0}\ \underbrace{110}_{6}\ \underbrace{100}_{4}\ \underbrace{011}_{3}$$

so $(1A3)_{16} = (0643)_8 \equiv (643)_8$

Octal to Hex Conversion

1. Convert the given octal number into binary.
2. Starting from right make groups of 4-bits and designate each group as a Hexadecimal digit.

Convert $(76)_8$ into hexadecimal.

Solution. 1. Converting octal to binary

$$(76)_8 = \underbrace{111}_{7}\ \underbrace{110}_{6}$$

2. Grouping of 4-bits

$$(76)_8 = \underbrace{11}_{3}\ \underbrace{1110}_{E} \equiv \underbrace{0011}_{3}\ \underbrace{1110}_{E}$$

$\therefore$ $(76)_8 = (3E)_{16}$

THE BINARY ARITHMETIC OPERATIONS

- Binary arithmetic's are simpler than decimal because they involve only two digits (bits) 1 and 0.
- Addition, subtraction, multiplication and division.

Binary Addition

Augend	*Addend*	*Sum*	*Carry*	*Result*
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	0	1	10

Binary subtraction

Minuend	*Subtrahend*	*Difference*	*Borrow*
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

(i) Add 1010 and 0011 (ii) Add 0101 and 1111

```
  111   ←Carry
  0101
+ 1111
 10100
 ↑
 Carry
```

```
  111   ←Carry
  0101
+ 1111
 10100
 ↑
 Carry
```

(i) Subtract 0100 from 1011 (ii) *Subtract 0110 from 1001*

1				←Borrow
1	0	1	1	←Minuend
−0	1	0	0	←Subtrahend
0	1	1	1	←Difference
↑	↑	↑	↑	
C_3	C_2	C_1	C_0	

1	1			←Borrow
1	0	0	1	←Minuend
−0	1	1	0	←Subtrahend
0	0	1	1	←Difference
↑	↑	↑	↑	
C_3	C_2	C_1	C_0	

The rules are still the same as in decimal, except that the borrow in a given significant position adds 2 to a minuend digit.

Binary multiplication

Multiplicand	Multiplier	product
0	0	0
1	0	0
1	1	1
0	1	0

(i) Multiply 1001 with 101

```
   1 0 0 1 ← MULTIPLICAND
     1 0 1 ← MULTIPLIER
   -------
   1 0 0 1 ← Partial Product when multiplier bit = 1
 0 0 0 0 x ← Partial Product when multiplier bit = 0
1 0 0 1 x x
-----------
1 0 1 1 0 1 ← FINAL PRODUCT
```

Binary division.

Divisor	dividend	Q
X	Y	IF X< Y, Q = 1
X	Y	IF X>Y, Q = 0

Binary division is also similar to decimal division

```
                      1 0 1
Divisor → 1 0 0 1 ) 1 0 1 1 0 1 ← Dividend
                    1 0 0 1
                    -------
                    x x 1 0 0 1
                        1 0 0 1
                        -------
                        x x x x
```

NEGATIVE NUMBERS AND THEIR ARITHMETIC

So far, we have discussed straight forward number representation which are nothing but positive number. The negative numbers have got two representation

- **Complement representation**. In digital computers to simplify the subtraction operation & for logical manipulation complements are used. There are two types of complements used in each radix system.
    - The radix complement or r's complement
    - The diminished radix complements or (r-1)'s complement

r's Complement and (r – 1)'s Complement

The r's and (r – 1)'s complements are generalized representation of the complements. r stands for radix or base of the number system; thus, r's complement is referred as radix complement and (r – 1)'s complement is referred as diminished radix complement. Examples of r's complements are 2's complement and 10's complement. Examples of (r – 1)'s complement are 1's complement and 9's

In a base-r system, the r's and $(r - 1)$'s complement of the number N having n digits, can be defined as:

$$(r - 1)\text{'s complement of N} = (r^n - 1) - N$$

and

$$r\text{'s complement of N} = r^n - N = (r - 1)\text{'s complement of N} + 1$$

The $(r - 1)$'s complement can also be obtained by subtracting each digit of N from $r-1$. Using the above methodology we can also define the 7's and 8's complement for octal system and 15's and 16's complement for hexadecimal system.

- **Sign magnitude representation** : Representation of signed no's binary arithmetic in computers: Two ways of representation of signed no's Sign Magnitude for Complemented form, Two complimented forms: 1's compliment form, & 2's compliment form

  1's and 2's Complement: These are the complements used for binary numbers. Their representation are very important as digital systems work on binary numbers only.

1's Complement

bit	Actual binary	complement

1's complement of a binary number is obtained simply by replacing each 1 by 0 and each 0 by 1. Alternately, 1's complement of a binary can be obtained by subtracting each bit from 1.

1's Complement	1	0
	0	1

EX: Find 1's complement of (i) 011001 (ii) 00100111

Sol: (i) Replace each 1 by 0 and each 0 by 1

0 1 1 0 0 1

↓ ↓ ↓ ↓ ↓ ↓

1 0 0 1 1 0

So, 1's complement of 011001 is 100110.

2's Complement: 2's complement of a binary number can be obtained by adding 1 to its 1's complement.

EX: Find 2's complement of (i) 011001 (ii) 010110016

Solution.

(i)

0	1	1	0	0	1		← Number
1	0	0	1	1	0		← 1's complement
				+	1		← Add 1 to 1's complement
1	0	0	1	1	1		← 2's complement

(ii)

0	1	0	1	1	0	0	← Number
1	0	1	0	0	1	1	← 1's complement
					+	1	← Add 1 to 1's complement
1	0	1	0	1	0	0	← 2's complement

Subtraction Using 1's and 2's Complement

Before using any complement method for subtraction equate the length of both minuend and subtrahend by introducing leading zeros.

1's complement subtraction following are the rules for subtraction using 1's complement.

1. To do the subtraction (M-S), represent the M&S in equal no. of digits.
2. Add 1's complement of subtrahend to minuend.
3. If a carry is produced by addition, then add this carry to the LSB of result. This is called as end around carry (EAC).
4. If carry is generated from MSB in step 2 then result is positive. If no carry generated result is negative, and is in 1's complement form.

EX: Perform binary subtraction for $(23)_{10}-(11)_{10}$

Sol: M= 23, 10111

S= 11, 1011

Step 1: represent the M&S in equal no. of digits. 10111=23

01011=11

Step 2: 1's complement of subtrahend (01011) = 10100

Add 1's complement of subtrahend to minuend 10111= M

+ 10100=1'S Comp of S

Carry 101011

Step3: If a carry is produced by addition, then add this carry to the LSB of result.

01011

+ 1

01100 =12

2's complement Subtraction:

Method of 2's complement is similar to 1's complement subtraction except the end around carry (EAC). The rules are listed below:

1. To do the subtraction (M-S), represent the M&S in equal no. of digits.
2. Take 2's complement of subtrahend. Add 2's complement of subtrahend to minuend.

3. If a carry is produced, then discard the carry and the result is positive. If no carry is produced result is negative and is in 2's compliment form.

EX: Perform binary subtraction for $(22)_{10}$-$(12)_{10}$ Using 2's complement

Sol: M= 22, 10110

S= 12, 1100

Step 1: represent the M&S in equal no. of digits. 10110=22

01100=12

Step 2: 2's complement of subtrahend (01100) = 10100

Add 2's complement of subtrahend to minuend 10110= M

+ 10100=1'S Comp of S

Carry 101010

(Neglected)

Step 3: If a carry is produced, then discard the carry and the result is positive = (01010) = $(10)_{10}$

Signed Binary Representation

Untill now we have discussed representation of unsigned (or positive) numbers, except one or two places. In computer systems sign (+ve or –ve) of a number should also be represented by binary bits.

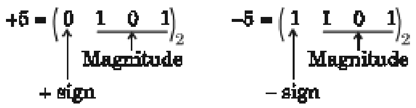
The accepted convention is to use 1 for negative sign and 0 for positive sign. In signed representation MSB of the given binary string represents the sign of the number, in all types of representation. We have two types of signed representation:

1. Signed Magnitude Representation

2. Signed Complement Representation

sign magnitude representation.

In a signed-magnitude representation, the MSB represent the sign and rest of the bits represent the magnitude. e.g.,



Note that positive number is represented similar to unsigned number. From the example it is also evident that out of 4-bits, only 3-bits are used to represent the magnitude.

What is sign magnitude of +5 and -7?

Sign bit	Actual binary number
+ is (0)	X
- Is (1)	X

Sol: actual number 5 is 0101 in binary number system. But to represent signed number in computer it has to represent in 8-bit binary number then

5 → 0000101 → 8 bit binary

+5 → 00000101 → signed magnitude for positive

-5 → 10000101 → signed magnitude for negative

Complement of signed magnitude representation

In a signed-complement representation the positive numbers are represented in true binary form with MSB as 0. Whereas the negative numbers are represented by taking appropriate complement of equivalent positive number, including the sign bit. Both 1's and 2's complements can be used for this purpose e.g.,

$+5 = (0101)_2$

$-5 = (1010)_2$ ←in 1's complement

$= (1011)_2$ ←in 2's complement

What is sign magnitude of +5 and -7 in 1's complement and 2's complement form

Sign bit	*1's complement* Of actualnumber	*2's complement* Of actual number
+ is (0)	X	X +1
- Is (1)	X	X+1

+5→ 00000101 → signed magnitude number

→ 11111010 → 1's complement of signed magnitude number

+1

→ 11111011 → 2's complement of signed magnitude number

9's and 10's Complement

9's and 10's complements are the methods used for the representation of decimal numbers. They are identical to the 1's and 2's complements used for binary numbers.

9's complement: 9's complement of a decimal number is defined as (10n – 1) – N, where n is no. of digits and N is given decimal numbers. Alternately, 9's complement of a decimal number can be obtained by subtracting each digit from 9.

9's complement of N = $(10^n - 1) - N$

EX: Find out the 9's complement of $(36)_{10}$.

Sol: By using $(10^n-1) - N$; n = 2. So, $(10^n-1) - N = (100 - 1) - 36 = 63$

By subtracting each digit from 9

$$\begin{array}{r} 9\ 9 \\ -3\ 6 \\ \hline 6\ 3 \end{array}$$

So, 9's complement of 36 is 63.

10's complement: 10's complement of a decimal number is defined as $10^n - N$. 10's complement of N = $10^n - N$ (or)

$10^n - N = (10^n - 1) - N + 1$ = 9's complement of N + 1. Thus, 10's complement of a decimal number can also be obtained by adding 1 to its 9's complement.

EX: Find out the 10's complement of $(36)_{10}$.

Solution. By adding 1 to 9's complement

9's complement of 36 = 99 – 36
= 63

Hence, 10's complement of 36 = 63 + 1
= 64

CODES

Coding and encoding is the process of assigning a group of binary digits, commonly referred to as 'bits', to represent, identify, or relate to a multivalued items of information. In short, a code is a symbolic representation of an information transform. The bit combination is referred to as 'CODEWORDS'.

In a broad sense we can classify the codes into five groups:

(i) Weighted Binary codes (ii) Non-weighted codes (iii) sequential codes(iv) Error–detecting codes (v) Error–correcting codes (vi) Alphanumeric codes

i) Weighted Binary Codes

In weighted binary codes, each position of a number represents a specific weight. The bits are multiplied by the weights indicated; and the sum of these weighted bits gives the equivalent decimal digit.

a) **Straight Binary coding:** is a method of representing a decimal number by its binary equivalent. A straight binary code representing decimal 0 through 7

Decimal	*Three bit straight Binary Code*	*Weights MOI* 2^2	2^1	2^0	*Sum*
0	000	0	0	0	0
1	001	0	0	1	1
2	010	0	2	0	2
3	011	0	2	1	3
4	100	4	0	0	4
5	101	4	0	1	5
6	110	4	2	0	6
7	111	4	2	1	7

b) **Binary Codes Decimal Codes (BCD codes).** In BCD codes, individual decimal digits are coded in binary notation and are operated upon singly. Thus, binary codes representing 0 to 9 decimal digits are allowed. Therefore, all BCD codes have at least four bits (∵ min. no. of bits required to encode to decimal digits = 4) For example, decimal 364 in BCD

3 → 0011

6 → 0110

4 → 0100

364 → 0011 0110 0100

However, we should realize that with 4 bits, total 16 combinations are possible (0000, 0001, ..., 1111) but only 10 are used (0 to 9). The remaining 6 combinations are invalid and commonly referred to as 'UNUSED CODES'

ii) Non weighted codes

Non weighted codes are codes that are not positionally weighted. That is, each position within the binary number is not assigned a fixed value. Ex: Excess-3 code, Gray code.

Excess-3 Code

Excess-3 is a non-weighted code used to express decimal numbers. The code derives its name from the fact that each binary code is the corresponding 8421 code plus 0011(3).

$[643]_{10}$ into XS3 code

Decimal	6	4	3
Add 3 to each	3	3	3
Sum	9	7	6

Converting the sum into BCD code we have

9	7	6
↓	↓	↓
1001	0111	0110

Hence, XS3 for $[643]_{10}$ = 1001 0111 0110

Gray Code

The Gray code belongs to a class of codes called minimum change codes, in which only one bit in the code changes when moving from one code to the next. The Gray code is non-weighted code, as the position of bit does not contain any weight. The Gray code is a reflective digital code which has the special property that any two subsequent numbers codes differ by only one bit. This is also called a unit-distance code. In digital Gray code has got a special place.

Gray codes*

Decimal Digit	*Three bit Gray code*	*Four bit Gray code*	*Decimal Digit*	*Three bit Gray code*	*Four bit Gray code*
0	0 0 0	0 0 0 0	8	–	1 1 0 0
1	0 0 1	0 0 0 1	9	–	1 1 0 1
2	0 1 1	0 0 1 1	10	–	1 1 1 1
3	0 1 0	0 0 1 0	11	–	1 1 1 0
4	1 1 0	0 1 1 0	12	–	1 0 1 0
5	1 1 1	0 1 1 1	13	–	1 0 1 1
6	1 0 1	0 1 0 1	14	–	1 0 0 1
7	1 0 0	0 1 0 0	15	–	1 0 0 0

Binary to Gray conversion:

N bit binary no is rep by $B_n\ B_{n-1}$ ------------ B_1
Gray code equivalent is by $G_n\ G_{n-1}$ ------------ G_1
B_n,, G_n are the MSB's then the gray code bits are obtaind from the binary code as

$G_n=B_n$	$G_{n-1}=B_n \oplus B_{n-1}$	$G_{n-2}=B_{n-1} \oplus B_{n-}$	-----------	$G_1=B_2 \oplus B1$	

$\oplus$ →EX-or symbol

Procedure: ex-or the bits of the binary no with those of the binary no shifted one position to the right . The LSB of the shifted no. is discarded & the MSB of the gray code no.is the same as the MSB of the original binaryno.

EX: 10001 $\oplus$ $\oplus$ $\oplus$

(a). Binary : 1 →0 →0 →1

Gray : 1 1 0 1

(b). Binary: 1 0 0 1
Shifted binary: 1 0 0 (1)

—— – – – – – – –

1 1 0 1→gray

Gray to Binary Conversion:

If an n bit gray no. is rep by $G_n\ G_{n-1}$ ------------ G_1

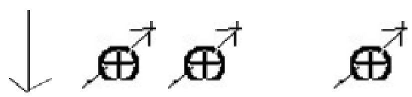
its binary equivalent by $B_n\ B_{n-1}$ ------------ B_1 then the binary bits are obtained from gray bits as

$B_n= G_n$	$Bn-1=B_n \oplus G_{n-1}$	$Bn-2= \oplus G_{n-2}$	-----------	$B1 =B_2 \oplus G_1$

To convert no. in any system into given no. first convert it into binary & then binary to gray. To convert gray no into binary no & convert binary no into require no system.

Ex:10110010(gray) = 11011100_2= DC_{16}=334_8=220_{10}
EX:1101

Gray: 1 1 0 1



Binary: 1 0 0 1

Ex: $3A7_{16}$=$0011,1010,0111_2$=1001110100(gray)
527_8=$101,011,011_2$=111110110(gray)
652_{10}=1010001100_2= 1111001010(gray)

XS-3 gray code:

In a normal gray code , the bit patterns for 0(0000) & 9(1101) do not have a unit distance between them i.e, they differ in more than one position.In xs-3 gray code , each decimal digit is encoded with gray code patter of the decimal digit that is greater by 3. It has a unit distance between the patterns for 0 & 9.

XS-3 gray code for decimal digits 0 through 9

Decimal digit	Xs-3 gray code	Decimal digit	Xs-3 gray code
0	0010	5	1100
1	0110	6	1101
2	0111	7	1111
3	0101	8	1110
4	0100	9	1010

iii) Sequential Codes

A code is said to be sequential when two subsequent codes, seen as numbers in binary representation, differ by one. This greatly aids mathematical manipulation of data. The 8421 and

Excess-3 codes are sequential, whereas the 2421 and 5211 codes are not.

<u>Binary coded decimal (bcd) and its arithmetic:</u>

The BCD is a group of four binary bits that represent a decimal digit. In this representation each digit of a decimal number is replaced by a 4-bit binary number (i.e., a nibble). Since a decimal digit is a number from 0 to 9, a nibble representing a number greater than 9 is invalid BCD. For example (1010)2 is invalid BCD as it represents a number greater than 9.

Decimal Number	*Binary Representation*	*BCD Representation*
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 0
3	0 0 1 1	0 0 1 1
4	0 1 0 0	0 1 0 0
5	0 1 0 1	0 1 0 1
6	0 1 1 0	0 1 1 0
7	0 1 1 1	0 1 1 1
8	1 0 0 0	1 0 0 0
9	1 0 0 1	1 0 0 1
10	1 0 1 0	0 0 0 1 0 0 0 0
11	1 0 1 1	0 0 0 1 0 0 0 1
12	1 1 0 0	0 0 0 1 0 0 1 0
13	1 1 0 1	0 0 0 1 0 0 1 1
14	1 1 1 0	0 0 0 1 0 1 0 0
15	1 1 1 1	0 0 0 1 0 1 0 1

BCD Addition: In many application it is required to add two BCD numbers. But the adder circuits used are simple binary adders, which does not take care of peculiarity of BCD representation. Thus one must verify the result for valid BCD by using following rules:

1. If Nibble (i.e., group of 4-bits) is less than or equal to 9, it is a valid BCD number.

2. If Nibble is greater than 9, it is invalid. Add 6 (0110) to the nibble, to make it valid. Or If a carry was generated from the nibble during the addition, it is invalid. Add 6 (0110) to the nibble, to make it valid.

3. If a carry is generated when 6 is added, add this carry to next nibble.

EX: Add the BCD numbers i)1000 and 0101 and ii) 00011001 and 00011000

Solution.

```
    1 0 0 0  →    8
  + 0 1 0 1  →  + 5
    1 1 0 1      13
```

Since, $(1101)_2 > (9)_{10}$ add 6 (0110) to it

So,

```
      1 1 0 1
      0 1 1 0
    1 0 0 1 1
    1    3
```

So, result = 00010011

(ii)

```
          1                 ← Carry generated from nibble
  0 0 0 1   1 0 0 1  →   19
  0 0 0 1   1 0 0 0  →  +18
  0 0 1 1   0 0 0 1      37
```

Since, a carry is generated from right most nibble we must add 6 (0110) to it.

So,

```
  0 0 1 1   0 0 0 1
            0 1 1 0
  0 0 1 1   0 1 1 1  → (37)10
```

So, result = 00110111

BCD Subtraction:

The best way to cary out the BCD subtraction is to use complements. e. Although any of the two complements can be used, we prefer 10's complement for subtraction. Following are the steps to be followed for BCD subtraction using 10's complement:

1. Add the 10's complement of subtrahend to minuend.
2. Apply the rules of BCD addition to verify that result of addition is valid BCD.
3. Apply the rules of 10's complement on the result obtained in step 2, to declare the final result i.e., to declare the result of subtraction.

Ex: Subtract 61 from 68 using BCD.

Solution. To illustrate the process first we perform the subtraction using 10's comple ment in decimal system. After that we go for BCD subtraction.

we have, D = 68 – 61

So, 10's complement of 61 = 99 – 61 + 1 = 39

So, 10's complement of 61 = 99 – 61 + 1 = 39

So,

```
    6 8
  + 3 9
  1 0 7
  ↑
  Carry
```

In 10's complement if an end carry is produced then it is discarded and result is declared positive. So,

D = +07

by using BCD

1.

```
                          1              ← Carry generated
BCD of 68 =     0 1 1 0   1 0 0 0          from nibble
BCD of 39 =  +  0 0 1 1   1 0 0 1
                1 0 1 0   0 0 0 1
```

2. Check for valid BCD– since a carry is generated from right most nibble, we must add 6 (0110) to it. Since the left most nibble is greater than 9, we must add 6(0110) to it.

Thus,

```
    1 0 1 0   0 0 0 1
  + 0 1 1 0   0 1 1 0
```

(a) 8421 BCD code, sometimes referred to as the Natural Binary Coded Decimal Code (NBCD);

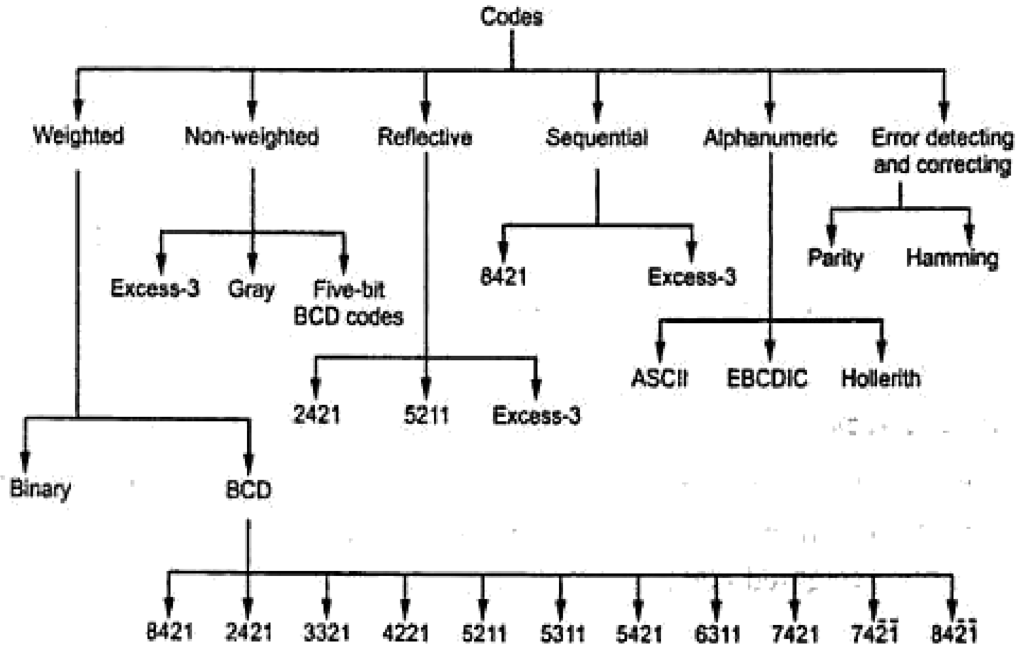
(b)* Excess-3 code (XS3); adding 3 to BCD gives the Excess -3 code.

(c)** 84 –2 –1 code (+8, +4, –2, –1);

(d) 2 4 2 1 code

Table BCD codes

Decimal Digit	*8421 (NBCD)*	*Excess-3 code (XS3)*	*84–2–1 code*	*2421 code*
0	0000	0011	0000	0000
1	0001	0100	0111	0001
2	0010	0101	0110	0010
3	0011	0110	0101	0011
4	0100	0111	0100	0100
5	0101	1000	1011	1011
6	0110	1001	1010	1100
7	0111	1010	1001	1101
8	1000	1011	1000	1110
9	1001	1100	1111	1111



Binary codes block diagram

Error – Detecting codes: When binary data is transmitted & processed,it is susceptible to noise that can alter or distort its contents. The 1‘s may get changed to 0‘s & 1‘s .because digital systems must be accurate to the digit, error can pose a problem. Several schemes have been devised to detect the occurrence of a single bit error in a binary word, so that whenever such an error occurs the concerned binary word can be corrected & retransmitted.

Parity: The simplest techniques for detecting errors is that of adding an extra bit known as parity bit to each word being transmitted.Two types of parity: Oddparity, evenparity forodd parity, the parity bit is set to a ‗0‘ or a ‗1‘ at the transmitter such that the total no. of 1 bit in the word including the parity bit is an odd no.For even parity, the parity bit is set to a ‗0‘ or a ‗1‘ at the transmitter such that the parity bit is an even no.

Decimal	8421 code	Odd parity	Even parity
0	0000 1		0
1	0001 0		1
2	0010 0		1
3	0011 1		0
4	0100 0		1
5	0100 1		0
6	0110 1		0
7	0111 0		1
8	1000 0		1
9	1001 1		0

When the digit data is received . a parity checking circuit generates an error signal if the total no of 1‘s is even in an odd parity system or odd in an even parity system. This parity check can always detect a single bit error but cannot detect 2 or more errors with in the same word.Odd parity is used more often than even parity does not detect the situation. Where all 0‘s are created by a short ckt or some other fault condition.

Ex: Even parity scheme
(a) 10101010 (b) 11110110 (c)10111001
Ans:

- **(a) No. of 1‘s in the word is even is 4 so there is no error**
- **(b) No. of 1‘s in the word is even is 6 so there is no error**
- **(c) No. of 1‘s in the word is odd is 5 so there is error**

Ex: odd parity
(a)10110111 (b) 10011010 (c)11101010

Ans:

- **(a) No. of 1‘s in the word is even is 6 so word has error**
- **(b) No. of 1‘s in the word is even is 4 so word has error**
- **(c) No. of 1‘s in the word is odd is 5 so there is no error**

Checksums:

Simple parity can‘t detect two errors within the same word. To overcome this, use a sort of 2 dimensional parity. As each word is transmitted, it is added to the sum of the previously transmitted words, and the sum retained at the transmitter end. At the end of transmission, the sum called the check sum. Up to that time sent to the receiver. The receiver can check its sum with the transmitted sum. If the two sums are the same, then no errors were detected at the receiver end. If there is an error, the receiving location can ask for retransmission of the entire data, used in teleprocessing systems.

Block parity:

Block of data shown is create the row & column parity bits for the data using odd parity. The parity bit 0 or 1 is added column wise & row wise such that the total no. of 1‘s in each column & row including the data bits & parity bit is odd as

Data	Parity bit
10110	0
10001	1
10101	0
00010	0
11000	1
00000	1
11010	0

data
10110
10001
10101
00010
11000
00000
11010

Error –Correcting Codes:

A code is said to be an error –correcting code, if the code word can always be deduced from an erroneous word. For a code to be a single bit error correcting code, the minimum distance of that code must be three. The minimum distance of that code is the smallest no. of bits by which any two code words must differ. A code with minimum distance of 3 can‘t only correct single bit errors but also detect (can‘t correct) two bit errors, The key to error correction is that it must be possible to detect & locate erroneous that it must be possible to detect & locate erroneous digits. If the location of an error has been determined. Then by complementing the erroneous digit, the message can be corrected , error correcting , code is the Hamming code , In this , to each group of m information or message or data bits, K parity checking bits denoted by P1,P2,----------pk located at positions 2^{k-1} from left are added to form an (m+k) bit code word.

To correct the error, k parity checks are performed on selected digits of each code word, & the position of the error bit is located by forming an error word, & the error bit is then complemented. The k bit error word is generated by putting a 0 or a 1 in the 2^{k-1}th position depending upon whether the check for parity involving the parity bit P_k is satisfied or not.Error positions & their corresponding values :

Error Position	For 15 bit code C_4 C_3 C_2 C_1	For 12 bit code C_4 C_3 C_2 C_1	For 7 bit code C_3 C_2 C_1
0	0000	0000	0 0 0
1	0001	0001	0 0 1
2	0010	0010	0 1 0
3	0011	0011	0 1 1
4	0100	0100	1 0 0
5	0101	0101	1 0 1
6	0 11 0	0 11 0	1 1 0
7	0 11 1	0 11 1	1 1 1
8	1 00 0	1 00 0	
9	1 00 1	1 00 1	
10	1 01 0	1 01 0	
11	1 01 1	1 01 1	
12	1 10 0	1 10 0	
13	1 10 1		
14	1 11 0		
15	1 11 1		

7- bit Hamming code:

To transmit four data bits, 3 parity bits located at positions 2^0 21&2^2 from left are added to make a 7 bit codeword which is then transmitted.

The word format

P_1	P_2	D_3	P_4	D_5	D_6	D_7

D—Data bits P-
Parity bits

Decimal Digit	For BCD P1P2D3P4D5D6D7	For Excess-3 P1P2D3P4D5D6D7
0	0 0 0 0 00 0	1 0 0 0 0 1 1
1	1 1 0 1 00 1	1 0 0 1 1 0 0
2	0 1 0 1 01 1	0 1 0 0 1 0 1
3	1 0 0 0 01 1	1 1 0 0 1 1 0
4	1 0 0 1 10 0	0 0 0 1 1 1 1
5	0 1 0 0 10 1	1 1 1 0 0 0 0
6	1 1 0 0 11 0	0 0 1 1 0 0 1
7	0 0 0 1 11 1	1 0 1 1 0 1 0

8	1 1 1 0 0 0 0	0 1 1 0 0 1 1
9	0 0 1 1 0 0 1	0 1 1 1 1 0 0

Ex: Encode the data bits 1101 into the 7 bit even parity Hamming Code
The bit pattern is
P1P2D3P4D5D6D7

1 1 0 1

Bits 1,3,5,7 (P_1 111) must have even parity, so P_1 =1
Bits 2, 3, 6, 7(P_2 101) must have even parity, so P_2 =0
Bits 4,5,6,7 (P_4 101)must have even parity, so P_4 =0
The final code is 1010101

EX: Code word is 1001001

Bits 1,3,5,7 (C_1 1001) →no error →put a 0 in the 1's position→C1=0
Bits 2, 3, 6, 7(C_2 0001)) → error →put a 1 in the 2's position→C2=1
Bits 4,5,6,7 (C_4 1001)) →no error →put a 0 in the 4's position→C3=0

15-bit Hamming Code: It transmit 11 data bits, 4 parity bits located 2^0 2^1 2^2 2^3 Word format is

P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D10	D11	D12	D13	D14	D15

12-Bit Hamming Code:It transmit 8 data bits, 4 parity bits located at position 2^0 2^1 2^2 2^3 Word format is

P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D10	D11	D12

Alphanumeric Codes:

These codes are used to encode the characteristics of alphabet in addition to the decimal digits. It is used for transmitting data between computers & its I/O device such as printers, keyboards & video display terminals.Popular modern alphanumeric codes are ASCII code & EBCDIC code.