Unit: 2

Number System and Codes

3.1 Decimal, Binary, Octal and Hexadecimal Number System and Conversion:

- **INTRODUCTIONTO NUMBER SYSTEM:**

In every step of life, we definitely find the use of numbers. If you need to buy something, you will have to pay a certain amount of money for which you will have to count the money. Likewise, the shopkeeper will also count the goods to give you and same for your changes. Therefore, number system can simply be defined as a way to represent numbers.

For example, a number system can be used to represent the number of players in a certain game like 11 players in a football team or for the number of audiences for a concert like 25000 concert-goers, etc. So, number system can also be viewed as a set of values that is used to represent different quantities.

- **TYPESOF NUMBER SYSTEM:**

1) Decimal Number System:

The decimal number system is the most commonly used number system in our daily life. This generally used number system is also known as the base 10 number system because it uses just the 10 symbols i.e. 0 to 9. It is also known as the denary number system because any numeric value there is, these system's digits can easily represent them. The decimal system is specially used in the computer interface. The weight and position of the digit dictate the value represented by it.

In this system, each number consists of the digits that are located at different positions. The positions of the 1^{st} and the 2^{nd} digits towards the right side of the decimal point are $^{-1}$ and $^{-2.}$ Similarly, the positions of the 1^{st} and 2^{nd} digits towards the left side of the decimal point are 0 and 1 respectively.

The value of the number is determined by adding the results out of the multiplication of the digits with the weight of their position. This method is called the expansion method. Under this method, the rightmost digit of the number is called the Least Significant Digit (LSD), as it has the lowest weight. Likewise, the leftmost digit of the number is called the Most Significant Digit (MSD), as it has the highest weight.

2) Binary Number System:

Binary Number System refers to the number system that uses only two symbols i.e. 0 and 1, that's 0why it is called a base 2 number system. It is also known as Binary Digit (BIT). This number system is especially used in the internal processing of computer system. When we count up from 0 in binary, symbols are much more frequently run out as this system only uses 0 and 1 and 2 do not exist. Therefore, we use 10 in this system because 10 is equal to 2 in decimal. The combination of binary numbers can be used to represent different quantities like 1001. In Binary, each digit's positional value is twice the face value or place value of the digit of its right side. Each position's weight is a power of 2.

3) Octal Number System:

Octal Number System is the base 8 system. Like the decimal number system, this system is also used in the internal processing of computer system. It is the system that consists of eight digits i.e. {0, 1, 2, 3, 4, 5, 6, and 7} which is used for the representation of long binary numbers short-handedly. In this system, each digit position represents a power of 8. The number 708 will not be valid in this system as 8 is not a valid digit.

4) Hexadecimal Number System:

Hexadecimal Number System is the number system that represents long binary numbers in shortcut method. It is a base 16 system as it consists of 16 digits i.e. {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F } where the alphabets represent the decimal numbers 10 to 15. This system is also used in the computer system, mainly in the memory management. As the name suggests, each digit's position represents a power of 16 in this system.

❖ <u>Conversion of the Number System</u>:

➢ CONVERSION OF BINARY

It consists of:

- conversion from binary to decimal number system
- conversion from binary to octal number system
- conversion from binary to hexadecimal number system

Conversion of binary to decimal (base 2 to base 10):

The rules for conversion from binary to decimal are given below:

1. Multiply each bit by corresponding power of 2 (base).
2. Sum each product term to get a decimal equivalent number.

Note: A power of 2 is 0 for a left bit of binary point (or for a right most bit for the number that does not contain fractional part) and increase the power by one for each bit towards left and decrease power by one towards the right of binary point.

Example 1: convert $(110011)_2$ to decimal.

Solution:

$(110011)_2 = 1\times2^5 + 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$

$= 32 + 16 + 0 + 0 + 2 + 1$

$= (51)_{10}$

Example 2: convert $(1011.101)_2$ into decimal.

$(1011.101)_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$

$= 8 + 0 + 2 + 1 + 0.5 + 0 + 0.125$

$= 11 + 0.5 + 0.125$

$= (11.625)_{10}$

Conversion of binary to octal (base 2 to base 8):

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

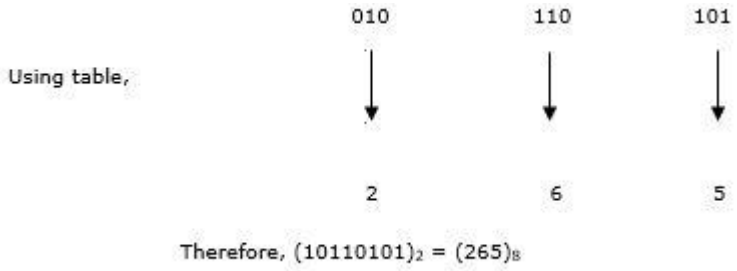
The rules for conversion from binary to decimal are given below:

1. Divide the given binary number into a group of three bits from binary point to left (or from right to left if no fractional part) and from binary point to the right. Append 0's at leading or trailing or trailing position if necessary, to make each group of 3 bits.
2. Substitute each group of three bits by octal equivalent from a table.
3. Collect octal digits to get an octal equivalent number.

Example: convert $(10110101)_2$ into octal.

Solution:

Dividing the given binary number $(10110101)_2$ into groups of 3 bits from right to left, and appending 0's at leading position, we have,



Therefore, $(10110101)_2 = (265)_8$

Conversion of binary to hexadecimal (base 2 to base 16):

Hexadecimal	Binary
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

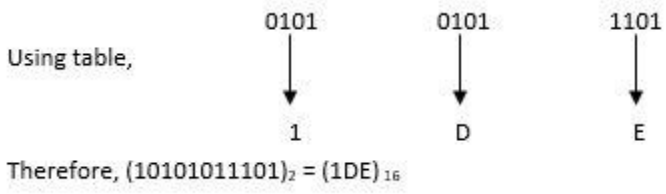
The rules for conversion from binary to hexadecimal are as given below:

1. Divide the given binary number into a group of four bits from binary point to left (or from right to left if no fractional part) and from binary point to the right. Append 0's at leading or trailing position if necessary to make each group of 4 bits.
2. Substitute each group of four bits by hexadecimal equivalent symbol (letter or digit) from the table.
3. Collect Hexadecimal symbols to get Hexadecimal equivalent number

Example: convert $(10101011101)_2$ into hexadecimal

Solution:

Dividing the given binary number $(10101011101)_2$ into groups of 4 bits from binary point to left and binary point right and appending 0's at leading or at trailing position to make each group of four bits, we have,



- **<u>Conversion of Decimal</u>**

It consists of conversion from decimal to binary, octal and hexadecimal number system which are described below:

- **Conversion of decimal to binary (base 10 to base 2):**

The rules for conversion from decimal to binary are as given below:

1. Divide the given number by 2 and note the remainder.
2. Repeatedly divide the quotient by two and note the remainder until quotient reduced to 0.
3. Collect the remainders, last obtained first and first obtained last to binary equivalent.

Example: convert $(51)_{10}$ into binary

Solution:

51÷2=25	remainder	= 1
25÷ 2=12	remainder	= 1
12÷ 2=6	remainder	= 0
6÷ 2=3	remainder	= 0
3÷ 2=1	remainder	= 1
1÷ 2=0	remainder	= 1

Therefore, $(51)_{10} = (110011)_2$

- **Conversion of decimal to octal (base 10 to base 8):**

The rules for conversion from decimal to binary are as given below:

1. Divide the given number by 8 and note the remainder.
2. Repeatedly divide the quotient by 8 and note the remainder until quotient reduced to 0.
3. Collect the remainders, last obtained first and first obtained last to get the octal equivalent.

Example: convert $(177)_{10}$ into octal

Solution:

177÷ 8 =22	remainder	= 1
22÷ 8 = 2	remainder	= 6
2÷ 8 = 0	remainder	= 2

Therefore, $(177)_{10} = (261)_8$

- **Conversion of decimal to hexadecimal (base 10 to base 16):**

Example: convert $(77)_{10}$ into hexadecimal

Solution:

77÷ 16=4 remainder = 13 which means D

4÷ 16 =0 remainder = 4

Therefore, $(77)_{10} = (4D)_{16}$

- **Conversion of octal:**

It consists of conversion from octal to decimal, binary and hexadecimal number system which are described below:

- **Conversion of octal to decimal (base 8 to base 10):**

Multiply each octal digit by corresponding power of 8 and sum each product term to get decimal equivalent.

Example: convert $(632)_8$ to decimal.

Solution:

$(632)_8 = 6\times8^2 + 3\times8^1 + 2\times8^0$

$= 384 + 24 + 2$

$= 410$

Therefore, $(632)_8 = (410)_{10}$

- **Conversion from octal to binary (base 8 to base 2):**

Substitute each octal digit by equivalent 3 bit binary from table and collect bits for each digit to get binary equivalent numbers.

Example: convert $(741)_8$ into binary.

Solution:

Using table,

7	4	1
↓	↓	↓
111	100	001

Therefore, $(741)_8 = (111100001)_2$

- **Conversion from octal to hexadecimal (base 8 to base 16):**

Actually, there is no direct method for converting from octal to hexadecimal so first, convert octal into binary or decimal and then convert binary or decimal to hexadecimal.

- **Conversion of hexadecimal**

It consists of conversion from hexadecimal to decimal, binary and octal number system which are described below:

- **Conversion of hexadecimal to decimal (base 16 to base 10):**

The rules for conversion from hexadecimal to decimal are as given below:

1. Multiply each digit by corresponding power of 16(base) as in decimal to binary.
2. Sum each product term to get decimal equivalent.

Example: convert $(F4C)_{16}$ into decimal.

Solution:

$(F4C)_{16} = F\times16^2 + 4\times16^1 + C\times16^0$

$= 15\times256 + 4\times16 + 12\times1$

$= 3840 + 64 + 12$

$= 3916$

Therefore, $(F4C)_{16} = (3916)_{10}$

- **Conversion of hexadecimal to binary (base 16 to base 2):**

Substitute each hexadecimal symbol by equivalent 4 bit binary from table and collect bits for each digit to get binary equivalent numbers.

Example: convert $(A1D)_{16}$ into binary.

Solution:

Using table,

A	1	D
↓	↓	↓
1010	0001	1101

Therefore, $(A1D)_{16} = (101000011101)_2$

- **Conversion of hexadecimal to octal (base 16 to base 8):**

There is no direct method for converting from hexadecimal to octal, so first convert hexadecimal into binary or decimal and then convert binary or decimal to octal.

3.2 (9's and 10's) Complement Decimal Subtraction:

The 9`s complement of decimal number can be obtained by subtracting each digit of the number from 9.

For example, the 9`s complement of 3 is 6 (9-3=6), and 234 is 765 (999-234 =765).

The 10`s complement of decimal number can be obtained by adding 1 to the least significant digit of 9`s complement of that number. For example, 10`s complement of 3 is 7 (9-3=6+1=7), and 123 is 877.

Subtraction of decimal number using 9`s complement

Here are the steps are given below:

1. Make the both numbers having the same number of digits.
2. Determine the 9`s complement of the number from which we subtracted (subtrahend).
3. Add the 9`s complement to the given number from which we subtract (minuend).
4. If there exists` any additional digit (carry) in the result after addition, remove it and add it to the complement of the result and prefix by a negative sign to get the final result.

E.g. Subtract $(123)_{10}$ from $(345)_{10}$

9`s complement of 123= (999 -123) =876

Adding the 9`s complement with 345, i.e. 345 + 876 = 1221

In the result, most significant digit 1 is the carryover. So add this carry over to remaining digits 221

i.e., 221 + 1 = 222

Hence, $(222)_{10}$ is the required result after subtracting $(123)_{10}$ from $(345)_{10}$.

Subtraction using 10`s complement:

Here are the steps are given below:

1. Make the both numbers having same numbers of digits.
2. Determine the 10`s complement of the number to be subtracted (subtrahend).
3. Add the 10`s complement to the given number from which we subtract (minuend).
4. If their exits' any additional digit (carry) in the result after addition, remove it from the result and the remaining digits form the final result.
5. If there exists` no any carry then determines the 10`s complement of the result and prefix by the negative sign to get the final result.

Example: Subtract $(123)_{10}$ from $(345)_{10}$

10`s Complement of 123 = (999 - 123) = 876 + 1 = 877

Adding the 10`s complement with 345, i.e. 345 + 877 = 1222

In this result, most significant digit 1 is the carry over. So, remove it to find the result.

Therefore, $(222)_{10}$ is the required result.

Binary Mathematics

1. Binary Addition

Rule for binary addition0+0=01+0=10+1=11+1=10 (0 with carry over 1)

Example: Binary addition101101+101111000100 sum

2. Binary Subtraction

Rule for binary subtraction 1-1=01-0=10-1=1 (with borrowing 1)0-0=0

Example: Binary addition101101 minuend-10111 subtrahend10110 difference

3. Binary Multiplication

Rule for binary multiplication1*1=11*0=00*1=00*0=0

Example: Binary multiplication1011 multiplicand *1011 multiplier10111011*0000** +1011***1111001 product

4. Binary Division

Rule for binary division1/1=11/0=not defined0/1=00/0=not defined

Example: Binary DivisionDivide 101011 by 110110) 101011 (111 quotient -110 1001 -110 111 -110 1 remainder

Some Basic Terms Related with Number System

- **MSB (Most Significant Bit)**

The left most bit of a number is called MSB.

Example:

1010 = MBS

- **LSB (Leas Significant Bit)**

The right most bit of a number is called LSB.

Example:

1010 = LSB

BIT: Single binary number either 0 or 1

Nibble: Combination of 4 binary bits e.g. 1001

Byte: Combination of 8 binary bits e.g. 1001 0111

3.3 Calculation in Binary – Addition, Subtraction, One's and Two's complement Methods of Binary Subtraction:

COMPLEMENT

In a computer system, subtraction is not performed directly as arithmetic subtraction. It is performed by the technique called complement. It is the process of repeated addition.

- There are two types of complement: r`s complement and (r-1)`s complement.

Where 'r' is the base of a number system.

In binary number system, there are two types of complement: 1`s complement and 2`s complement.

Similarly, decimal number system has 9`s and 10`s complement.

- **1`s Complement**

1`s complement of a binary number is obtained by subtracting each bit by 1. We can get 1`s complement by simply replacing 1 by 0 and 0 by 1.

Example: 1`s complement of 1011 = 0100

Subtraction of binary numbers using 1`s complement

Steps are here as below:

1. Make the both numbers having the same number of bits.
2. Determine the 1`s complement of the number to be subtracted (subtrahend).
3. Add the 1`s complement to the given number from which we subtract (minuend).
4. If there exists` any additional bit (carry) in the result after addition, remove and add it to the result else (i.e. if there exists` no any carry)
5. Determine the 1`s complement of the result and prefix by a negative sign to get the final result**.** **Example: Subtract 1110000 from 1100000**

1`s complement of 1110000 =0001111
Adding it with minuend (i.e.1100000) =0001111
+1100000
1101111

Since, there exists` no any additional bits (carry),
1`s complement of 1101111 =0010000
Hence, the difference will be -0010000.
Example: 111000 -110000
1`s complement of 110000 =001111
Adding it with 111000 =001111
+111000
1000111

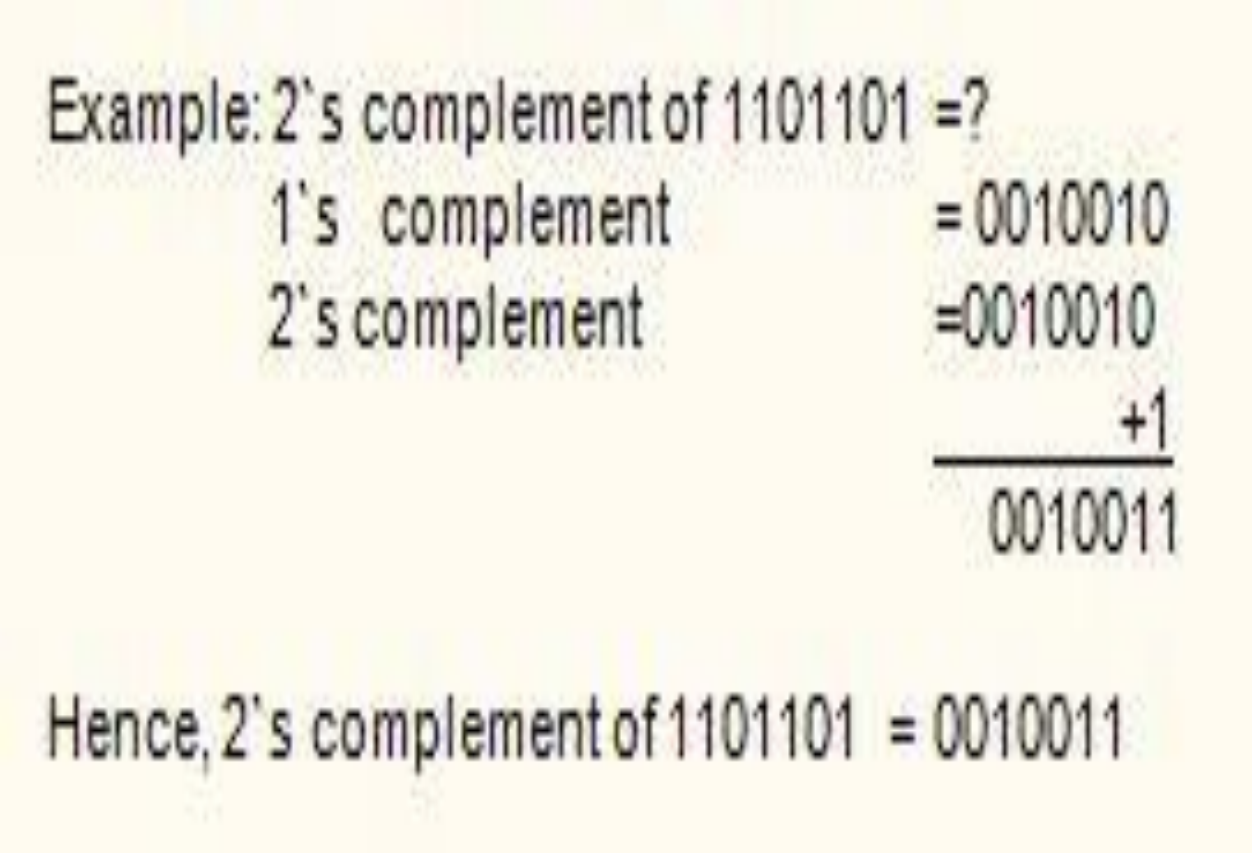
Since, there exists` one additional bit,
Difference
=000111
+1
001000

Hence, Difference will be 001000.

- **2`s complement:**

The 2`s complement of a binary number is obtained by adding binary 1 to the 1`s

Complement to the number.



Subtraction using 2`s complement:

Steps are here as below:

1. Make the both numbers having the same number of bits.
2. Determine the 2`s complement of the number to be subtracted (subtrahend).
3. Add the 2`s complement to the given number from which we subtract (minuend).
4. If there exists no carry determine the 2`s complement of the result and prefix by a negative sign to get a final result.

Example: Subtract 1110000 from 1100000

2`s complement of 1110000 =0001111
+1
0010000

Adding it with minuend (i.e.1100000) =0010000
+1100000
1110000

Since, there exists` no any additional bits (carry).
2`s complement of 1110000 =0001111
+1
0010000

Hence, the difference will be -0010000.
Example: 111000 -110000
2`s complement of 110000 =001111
+1
010000

Adding it with 111000 =010000
+111000
1001000

Since, there exists` one additional bit,
Difference =001000 (Neglect carry i.e. 1)
Hence, Difference will be 001000.

❖ Binary Coded Decimal (BCD):

The binary number system is the most natural system for a computer, but people are accustomed to the decimal system. So, to resolve this difference, computer uses decimals in coded form which the hardware understands. A binary code that distinguishes among 10 elements of decimal digits must contain at least four bits. Numerous different binary codes can be obtained by arranging four bits into 10 distinct combinations. The code most commonly used for the decimal digits is the straightforward binary assignment listed in the table below. This is called ***binary-coded decimal*** and is commonly referred to as **BCD**.

Example: A number with n decimal digits will require 4n bits in BCD. E.g. decimal 396 is represented in BCD with 12 bits as 0011 1001 011

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

❖ Error-Detection codes:

Binary information can be transmitted from one location to another by electric wires or other communication medium. Any external noise introduced into the physical communication medium may change some of the bits from 0 to 1 or vice versa.

The purpose of an error-detection code is to detect such bit-reversal errors. One of the most common ways to achieve error detection is by means of a parity bit. A parity bit is the extra bit included to make the total number of 1's in the resulting code word either even or odd. A message of 4-bits and a parity bit P are shown in the table below:

Error Checking Mechanism:

During the transmission of information from one location to another, an even parity bit is generated in the sending end for each message transmission.

The message, together with the parity bit, is transmitted to its destination. The parity of the received data is checked in the receiving end. If the parity of the received information is not even, it means that at least one bit has changed value during the transmission.

This method detects one, three, or any odd combination of errors in each message that is transmitted. An even combination of errors is undetected. Additional error detection schemes may be needed to take care of an even combination of errors.

Odd parity		Even parity	
Message	P	Message	P
0000	1	0000	0
0001	0	0001	1
0010	0	0010	1
0011	1	0011	0
0100	0	0100	1
0101	1	0101	0
0110	1	0110	0
0111	0	0111	1
1000	0	1000	1
1001	1	1001	0
1010	1	1010	0
1011	0	1011	1
1100	1	1100	0
1101	0	1101	1
1110	0	1110	1
1111	1	1111	0

- **Gray code (Reflected code)/Unit distance code/cyclic code:**

It is a binary coding scheme used to represent digits generated from a mechanical sensor that may be prone to error. Used in telegraphy in the late 1800s, and also known as "reflected binary code". Gray code was patented by Bell Labs researcher Frank Gray in 1947. In Gray code, there is only one-bit location different between two successive values, which makes mechanical transitions from one digit to the next less error prone.

The following chart shows normal binary representations from 0 to 15 and the corresponding Gray code. The Gray code is used in applications where the normal sequence of binary numbers may produce an error or ambiguity during the transition from one number to the next. If binary numbers are used, a change from 0111 to 1000 may produce an intermediate erroneous number 1001 if the rightmost bit takes more time to change than the other three bits. The Gray code eliminates this problem since only one-bit changes in value during any transition between two numbers.

Decimal digit	Binary code	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

❖ ***Alphanumeric codes:***

Alphanumeric character set is a set of elements that includes the 10 decimal digits, 26 letters of the alphabet and special characters such as $, %, + etc. It is necessary to formulate a binary code for this set to handle different data types. If only capital letters are included, we need a binary code of at least six bits, and if both uppercase letters and lowercase letters are included, we need a binary code of at least seven bits.

ASCII character code:

The standard binary code for the alphanumeric characters is called ASCII (American Standard Code for Information Interchange). It uses seven bits to code 128 characters as shown in the table below. The seven bits of the code are designated by B1 through B7 with B7 being the most significant bit.

American Standard Code for Information Interchange (ASCII)

	$B_7B_6B_5$							
$B_4B_3B_2B_1$	000	001	010	011	100	101	110	111
0000	NULL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

NULL	NULL
SOH	Start of heading
STX	Start of text
ETX	End of text
EOT	End of transmission
ENQ	Enquiry
ACK	Acknowledge
BEL	Bell
BS	Backspace
HT	Horizontal tab
LF	Line feed
VT	Vertical tab
FF	Form feed
CR	Carriage return
SO	Shift out
SI	Shift in
SP	Space

DLE	Data link escape
DC1	Device control 1
DC2	Device control 2
DC3	Device control 3
DC4	Device control 4
NAK	Negative acknowledge
SYN	Synchronous idle
ETB	End of transmission block
CAN	Cancel
EM	End of medium
SUB	Substitute
ESC	Escape
FS	File separator
GS	Group separator
RS	Record separator
US	Unit separator
DEL	Delete

Example: ASCII for each symbol is (B7B6B5B4B3B2B1)

EBCDIC character code:

EBCDIC (Extended Binary Coded Decimal Interchange Code) is another alphanumeric code used in IBM equipment. It uses eight bits for each character. EBCDIC has the same character symbols as ASCII, but the bit assignment for characters is different. As the name implies, the binary code for the letters and numerals is an extension of the binary-coded decimal (BCD) code. This means that the last four bits of the code range from 0000 through 1001 as in BCD.

- ***Excess-3 Code:***

The excess 3 code is defined as the binary code formed after adding the 3 i.e. 0011 to the decimal number converted to the BCD number.

Excess-3, also called XS3, is a non-weighted code used to express decimal number-s. It is another important binary code. It is particularly significant for arithmetic operations as it overcomes the shortcomings encountered while using the 8421 BCD code to add two decimal digits whose sum exceeds 9. This code is used in some old computers.

The Excess-3 code for a given decimal number is determined by adding '3' to each decimal digit in the given number and then replacing each digit of the newly found decimal number by its four-bit binary equivalent. The table gives is the Excess-3 code.

The key feature of the Excess-3 code is. that it is self-complementing. In other words, the l's complement of an Excess- 3 number is the Excess- 3 code for the 9's complement of the corresponding decimal number. For example, the Excess- 3 code for decimal 6 is 1001. The l's complement of 1001 is 0110, which is the Excess-3 code for decimal 3, and 3 is the 9's complement of 6.

Decimal	BCD 8 4 2 1	Excess-3 BCD + 0011
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

Example

Let us find the Excess 3 equivalent of the decimal number 786. This number has 3 decimal digits 7, 8 and 6. From the table, we can write the Excess 3 codes of 7, 8 and 6 are 1010, 1011 and 1001 respectively.
Therefore, the Excess 3 equivalent of the decimal number 786 is **101010111001**

Binary codes can be classified into two types.

- Weighted codes
- Unweighted codes

If the code has positional weights, then it is said to be **weighted code**. Otherwise, it is an unweighted code. Weighted codes can be further classified as positively weighted codes and negatively weighted codes.

❖ ***Binary Codes for Decimal digits:***

The following table shows the various binary codes for decimal digits 0 to 9.

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

We have 10 digits in decimal number system. To represent these 10 digits in binary, we require minimum of 4 bits. But, with 4 bits there will be 16 unique combinations of zeros and ones. Since, we have only 10 decimal digits, the other 6 combinations of zeros and ones are not required.

8 4 2 1 code

- The weights of this code are 8, 4, 2 and 1.
- This code has all positive weights. So, it is a **positively weighted code**.
- This code is also called as **natural BCD** BinaryCodedDecimalBinaryCodedDecimal **code**.

Example

Let us find the BCD equivalent of the decimal number 786. This number has 3 decimal digits 7, 8 and 6. From the table, we can write the BCD 84218421 codes of 7, 8 and 6 are 0111, 1000 and 0110 respectively.

$$\therefore 786786_{10} = 011110000110011110000110_{BCD}$$

There are 12 bits in BCD representation, since each BCD code of decimal digit has 4 bits.

❖ *2 4 2 1 code:*

- The weights of this code are 2, 4, 2 and 1.
- This code has all positive weights. So, it is a **positively weighted code**.
- It is an **unnatural BCD** code. Sum of weights of unnatural BCD codes is equal to 9.
- It is a **self-complementing** code. Self-complementing codes provide the 9's complement of a decimal number, just by interchanging 1's and 0's in its equivalent 2421 representation.

Example

Let us find the 2421 equivalent of the decimal number 786. This number has 3 decimal digits 7, 8 and 6. From the table, we can write the 2421 codes of 7, 8 and 6 are 1101, 1110 and 1100 respectively.

Therefore, the 2421 equivalent of the decimal number 786 is **110111101100**.

❖ *8 4 -2 -1 code:*

- The weights of this code are 8, 4, -2 and -1.
- This code has negative weights along with positive weights. So, it is a **negatively weighted code**.
- It is an **unnatural BCD** code.
- It is a **self-complementing** code.

Example

Let us find the 8 4-2-1 equivalent of the decimal number 786. This number has 3 decimal digits 7, 8 and 6. From the table, we can write the 8 4 -2 -1 codes of 7, 8 and 6 are 1001, 1000 and 1010 respectively.

Therefore, the 8 4 -2 -1 equivalent of the decimal number 786 is **100110001010**.

Binary code to Gray Code Conversion

Follow these steps for converting a binary code into its equivalent Gray code.

- Consider the given binary code and place a zero to the left of MSB.
- Compare the successive two bits starting from zero. If the 2 bits are same, then the output is zero. Otherwise, output is one.
- Repeat the above step till the LSB of Gray code is obtained.

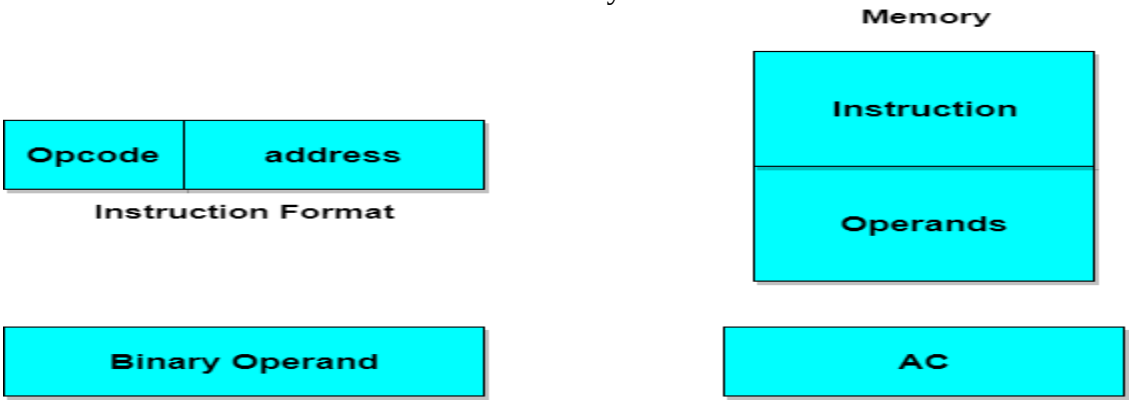
Example

From the table, we know that the Gray code corresponding to binary code 1000 is 1100. Now, let us verify it by using the above procedure.
Given, binary code is 1000.
Step 1 − By placing zero to the left of MSB, the binary code will be 01000.
Step 2 − By comparing successive two bits of new binary code, we will get the gray code as **1100**.

- ## Instruction Code:

  Instruction code is a group of bits that tells the computer to perform a specific operation part.The operation must be performed on the data stored in registers. An instruction code therefore specifies not only operations to be performed but also the registers where the operands(data) will be found as well as the registers where the result has to be stored.

Processor Register and instruction code with two parts. The first part specifies the operation to be performed and second specifies an address. The memory address tells where the operand in memory will be found.
Instructions are stored in one section of memory and data in another.



Computers with a single processor register is known as Accumulator (AC). The operation is performed with the memory operand and the content of A

❖ ***BCD Arithmetic and Excess-3 Arithmetic:***
Excess-3 Arithmetic:

Excess 3 Code Addition
The operation of addition can be done by very simple method we will illustrate the operation in a simple way using steps.
Step 1We have to convert the numbers (which are to be added) into excess 3 forms by adding 0011 with each of the four-bit groups them or simply increasing them by 3.
Step 2 Now the two numbers are added using the basic laws of binary addition, there is no exception for this method.
Step 3 Now which of the four groups have produced a carry we have to add 0011 with them and subtract 0011 from the groups which have not produced a carry during the addition.
Step 4 The result which we have obtained after this operation is in Excess 3 form and this is our desired result
Example To understand the Excess 3 Code Addition method better we can observe the method with the help of an example, Let us take two numbers which we will to add.0011 0101 0110 and 0101 0111 1001 are the two binary numbers. Now following the first step we take the excess 3 form of these two numbers which are 0110 1000 1001 and 1000 1010 1100, now these numbers

```
  0110 1000 1001
  1000 1010 1100
-----------------
  1111 0011 0101
```

are added following the **basic rules of addition**.
Now adding 0011 to the groups which produces a carry and subtracting 0011 from the groups which did not produced carry we get the result as 1100 0110 1000 is the result of the addition in excess 3 code and the BCD answer is 1001 0011 0101.

Excess 3 Code Subtraction
Similarly binary subtraction can be performed by Excess 3 Code Subtraction method. The operation is illustrated with the help of some steps.

Step 1 Like the previous method both the numbers have to be converted into excess 3 code
Step2 Following the basic methods of binary subtraction, subtraction is done
Step 3Subtract '0011' from each BCD four-bit group in the answer if the subtraction operation of the relevant four-bit groups required a borrow from the next higher adjacent four-bit group.
Step 4Add '0011' to the remaining four-bit groups, if any, in the result.
Step 5Finally we get the desired result in excess 3 code.

Example Again an example will make the understanding very easy for us. Let us take the numbers0001 1000 0101 and 0000 0000 1000 now the excess 3 equivalent of those numbers are 0100 1011 1000 and

```
0100 1011 1000
0011 0011 1011
---------------
0001 0111 1101
```

0011 0011 1011 Now performing the operation of binary subtraction we get Now in the above mentioned operation the least significant column which needed a borrow and the other two columns did not need borrow. Now we have to subtract 0011 from the result of this column and add 0011 to the other two columns, we get 0100 1010 1010. This is the result expressed in excess 3 codes. And the binary result is 0001 0111 011