UNIT 1

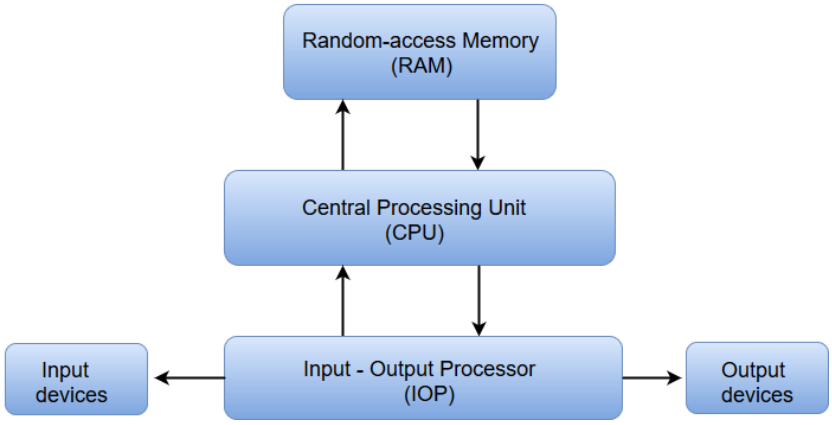
INTRODUCTION TO NUMBER SYSTEM AND CODES

Digital computers and digital systems

The digital computer is a digital system that performs various computational tasks. The word digital implies that the information in the computer is represented by variables that take a limited number of discrete values. These values are processed internally by components that can maintain a limited number of discrete states.

In practice, digital computers function more reliably if only two states are used. Digital computers use the binary number system, which has two digits: 0 and 1. A binary digit is called a bit. Information is represented in digital computers in groups of bits. By using various coding techniques, groups of bits can be made to represent not only binary numbers but also other discrete symbols, such as decimal digits or letters of the alphabet.

Block diagram of a digital computer:



- The Central Processing Unit (CPU) contains an arithmetic and logic unit for manipulating data, a number of registers for storing data, and a control circuit for fetching and executing instructions.
- The Random-Access Memory (RAM) for real-time processing of the data. The memory unit of a digital computer contains storage for instructions and data.
- The Input-Output devices for generating inputs from the user and displaying the final results to the user.
- The Input devices connected to the computer include the keyboard, mouse, microphone, scanner etc. The Output devices connected to the computer include the monitor, printer, speaker, projector etc.

NUMBER SYSTEM

Number system is simply the ways to count things. Aim of any number system is to deal with certain quantities which can be measured, monitored, recorded, manipulated arithmetically, observed and utilised. Each quantity has to be represented by its value as efficiently and accurately as is necessary for any application. The numerical value of a quantity can be basically expressed in either analog (continuous) or digital (step by step) method of representation. In digital computer system, data is represented in binary.

BINARY SYSTEM

All computers use the binary system. The following points provides an overview of the binary system:

- In the binary number system (base of 2), there are only two digits: 0 and 1. Binary digits are abbreviated as bits. For example, 1101 is a binary number of 4 bits (i.e.it is a binary number containing four binary digits.)
- A binary number may have any number of bits. Consider the number 11001.011. Note the binary point (counterpart of decimal point in decimal number system) in this number.
- Each digit is known as a bit and can take only two values 0 and 1. The left most bit is the highest-order bit and represents the most significant bit (MSB) while the lowest-order bit is the least significant bit (LSB). Some useful definitions are:
- Word is a binary number consisting of an arbitrary number of bits.
- Nibble is a 4-bit word (one hexadecimal digit) 16 values.
- Byte is an 8-bit word 256 values.

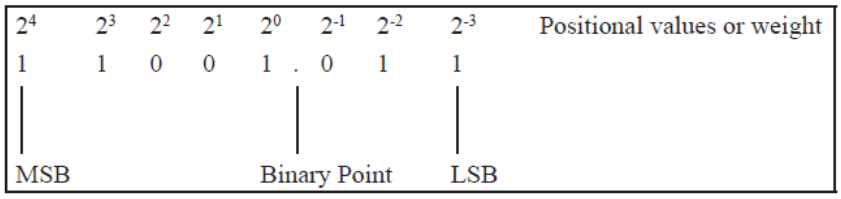


Fig. Positional value (weight) of each bit

Any number can be expressed in binary form in the usual way.

OCTAL NUMBER SYSTEM

The octal number system has base-8 that is, there arc 8 digits in this system. These digits are 0, 1, 2, 3, 4, 5, 6, and 7.

The weight of each octal digit is some power of 8 depending upon the position of the digit. Octal numbers showing positional values (weights) of each digit are as follows:

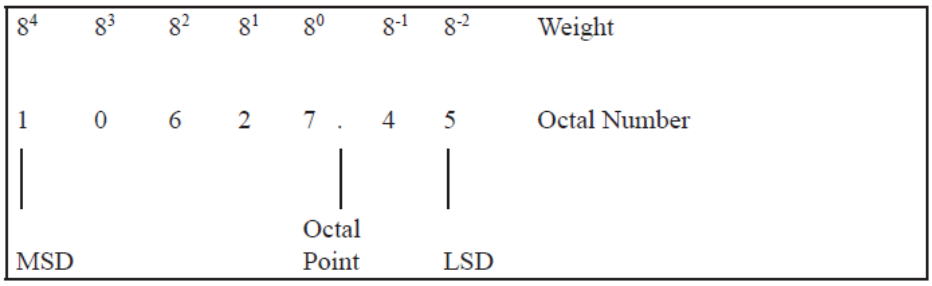


Fig. Octal number system

Octal number does not include the decimal digits 8 and 9. If any number includes decimal digits 8 and 9, then the number cannot be an octal number. Table below shows octal number:

Oct Digit	Octal Number	Decimal Number
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7

Table. Octal numbers

HEXADECIMAL NUMBER SYSTEM

The hexadecimal number system has base 16 that is it has 16 digits (Hexadecimal means'16'). These digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. The digits A, B, C, D, E and F have equivalent decimal values 10, 11, 12, 13, 14, and 15 respectively. Each Hex (Hexadecimal is popularly known as hex) digit in a hex number has a positional value that is some power of 16 depending upon its position in the number.

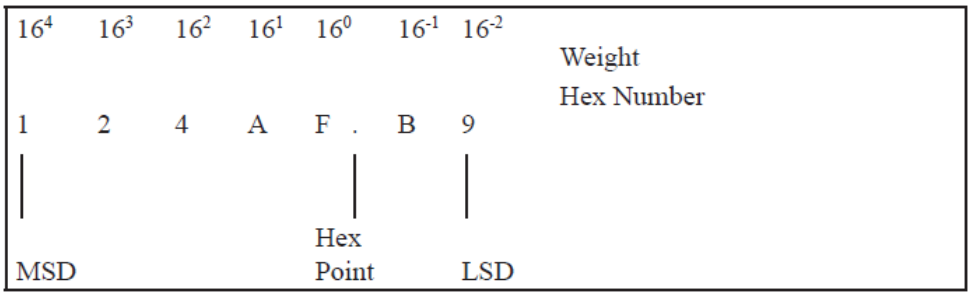


Fig. Hexadecimal number showing positional values (weight) of digits

Relationship of hex digits with decimal and binary numbers is given in Table below. Note that to represent the largest hex digit we require four binary bits. Therefore, the binary equivalent of all the hex digits has to be written in four bit numbers.

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

Table: decimal equivalent of each hex digit

NUMBER CONVERSION

DECIMAL TO BINARY CONVERSION

A decimal number is converted into its binary equivalent by its repeated divisions by 2. The division is continued till we get a quotient of 0. Then all the remainders are arranged sequentially with first remainder taking the position of LSB and the last one taking the position of MSB. Consider the conversion of 27 into its binary equivalent as follows.

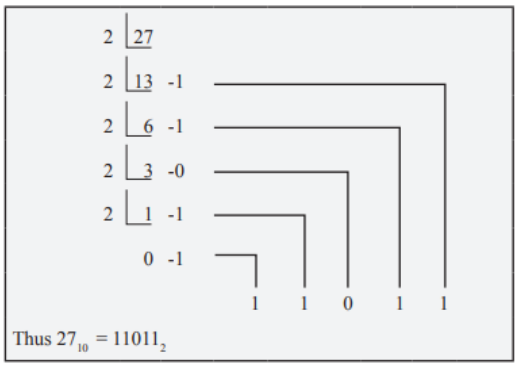


Fig. Decimal to binary conversion

If the number also has some figures on the right of the decimal point, then this part of the number is to be treated separately. Multiply this part repeatedly by two. After first multiplication by 2, either 1 or 0 will appear on the left of the decimal point. Keep this 1 or 0 separately and do not multiply it by 2 subsequently. This should be followed for every multiplication. Continue multiplication by 2 till you get all 0s after the decimal point or up to the level of the accuracy desired. Consider the conversion of 27.625_{10} into its binary equivalent. We have already converted 27 into its binary equivalent which is $(11011)_2$. Now for the conversion of 0.625, multiply it by 2 repeatedly as follows: Thus, $27.625_{10} = 11011.101$

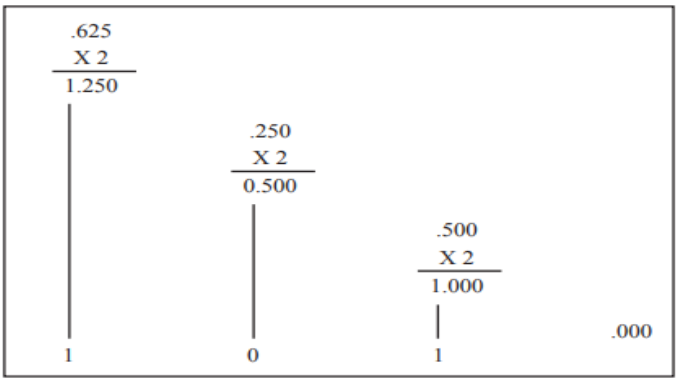


Fig. Decimal to binary conversion

Let us try another example, conversion of $(58.0725)_{10}$ into binary. Split this number in two parts, i.e. 58 and .0725 and convert them into binary separately as described above.

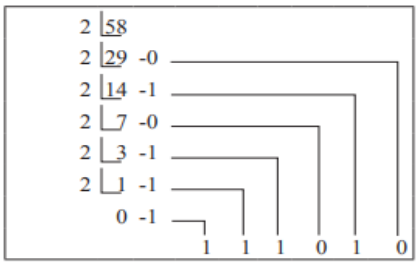


Fig. Decimal to binary conversion

Now let's look at the conversion of 0.725.

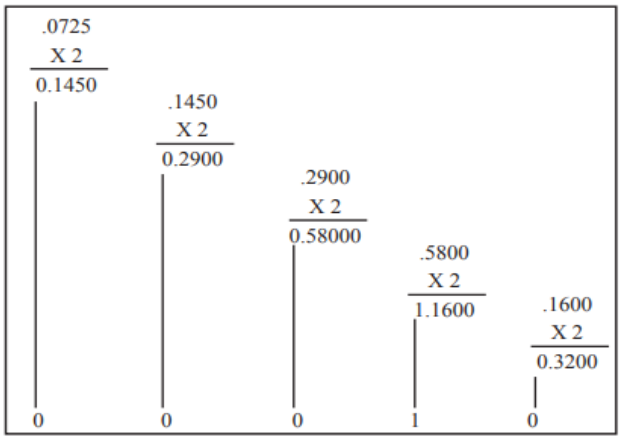


Fig: Decimal to binary conversion

Thus, $(58.0725)_{10} = 111010.00010$

Representing numbers in binary is very tedious, since binary numbers often consist of a large chain of 0's and 1's. Convenient shorthand forms for representing the binary numbers are developed such as octal system and hexadecimal system. With these number systems long strings of 0's and 1's can be reduced to a manageable form. The section below gives an overview of these systems.

Sample 1: Convert 200 to binary

200 % 2 = 100	remainder	0 (LSB)
100 % 2 = 50	remainder	0
50 % 2 = 25	remainder	0
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 (MSB)

Writing from MSB to LSB = 11001000

Thus $(200)_{10} = (11001000)_2$

Sample 2: Convert 0.375 to binary

0.375 x 2 = 0.750 integer 0 (MSB)

0.750 x 2 = 1.500 integer 1

0.500 x 2 = 1.000 integer 1 (LSB)

Writing from MSB to LSB = 011 i.e. writing from Top to bottom.

Therefore $(0.375)_{10} = (0.011)_2$

Sample 3: Convert 0.54545 to binary

0.54545 x 2 = 1.0909 integer 1 (MSB)

0.0909 x 2 = 0.1818 integer 0

0.1818 x 2 = 0.3636 integer 0

0.3636 x 2 = 0.7272 integer 0

0.7272 x 2 = 1.4545 integer 1 (LSB)

We can still continue to the desire number of decimal places after the decimal point.

Writing from MSB to LSB = 10001

Therefore $(0.54545)_{10} = (0.10001)_2$

Sample 4: Convert 16.512 to binary

Convert 16 separately

16%2 = 8 remainder 0

8%2 = 4 remainder 0

4%2 = 2 remainder 0

2%2 = 1 remainder 0

1%2 = 0 remainder 1

$(16)_{10} = (10000)_2$

Convert 0.512 separately
0.512x2 = 1.024 => 1
0.024x2 = 0.048 => 0
0.048x2 = 0.096 => 0
0.096x2 = 0.192 => 0
0.172x2 = 0.384 => 0
$(0.512)_{10} = (0.1000)_2$

Therefore, combining the results

$(16.512)_{10} = (10000.10000)_2$

Sample 5: Convert $(0.538)_{10}$ to binary equivalent

0.538 x 2 = 1.076 integral part 1

0.076 x 2 = 0.152 integral part 0

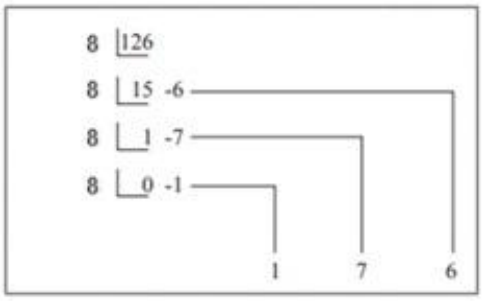
0.152 x 2 = 0.304 integral part 0

0.608 x 2 = 1.216 integral part 1

By writing the integral part from to to bottom $(0.538)_{10} = (0.1001)_2$

DECIMAL TO OCTAL CONVERSION

A decimal number can be converted by repeated division by 8 into equivalent octal number. This method is similar to that adopted in decimal to binary conversion. If the decimal number has some digits on the right of the decimal point, then this part of the number is converted into its octal equivalent by repeatedly multiplying it by 8. The process is same as has been followed in binary number system. Consider the conversion of 126.38_{10} into its decimal equivalent. Split it into two parts, that is 126 and .38



The conversion of .38 is as follows

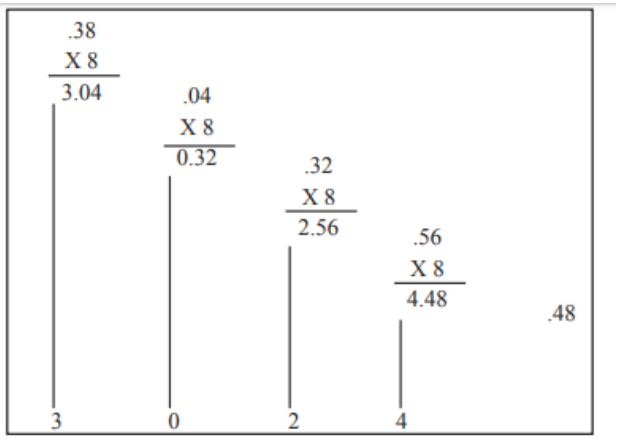


Fig. Decimal to octal conversion

Thus, $(126.38)_{10} = (176.3024)_8$

Example 1: Convert 83_{10} to octal

83 % 8 = 10	remainder	3 (LSB)
10 % 8 = 1	remainder	2
1 % 8 = 0	remainder	1 (MSB)

Writing from MSB to LSB = 123

Thus $(83)_{10} = (123)_8$

Example 2: Convert 668_{10} to octal

668 % 8 = 83	remainder	4 (LSB)
83 % 8 = 10	remainder	3
10 % 8 = 1	remainder	2
1 % 8 = 0	remainder	1 (MSB)

Writing from MSB to LSB = 1234, Thus $(668)_{10} = (1234)_8$

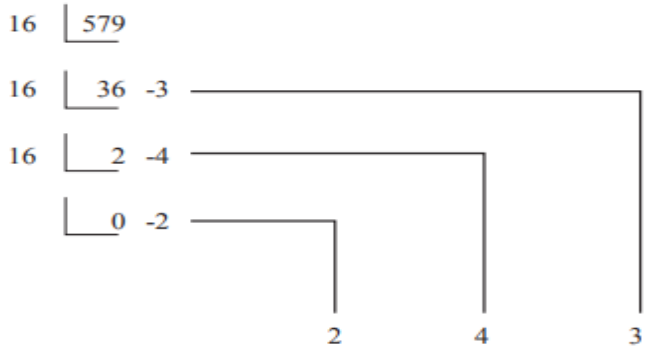
Example 3: Convert $(0.538)_{10}$ to octal equivalent

0.538 x 8 = 4.304	integral part 4
0.304 x 8 = 2.432	integral part 2
0.432 x 8 = 3.456	integral part 3
0.456 x 8 = 3.648	integral part 3

By writing the integral part from top to bottom $(0.538)_{10} = (0.4233)_8$

DECIMAL TO HEX CONVERSION

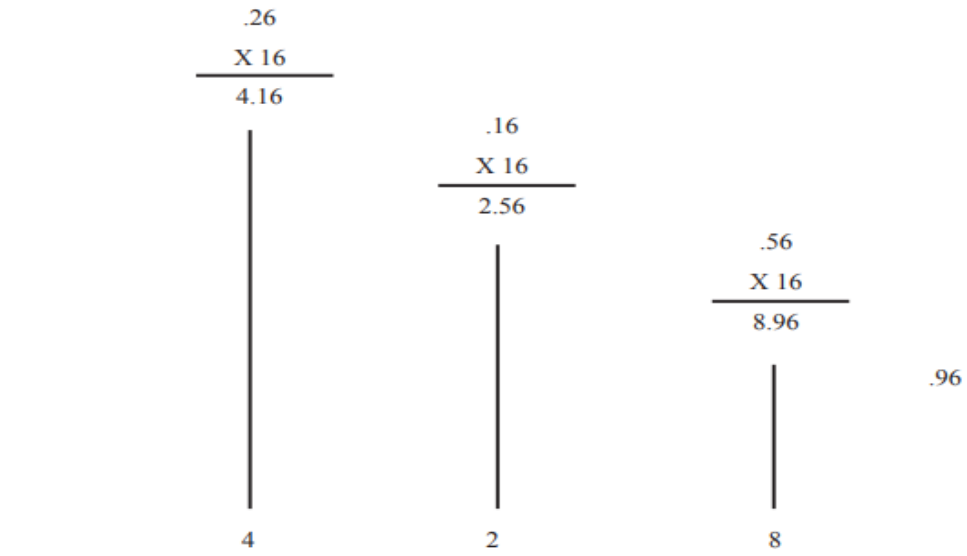
A decimal number is converted into hex number in the same way as a decimal number is converted into its equivalent binary and octal numbers. The part of the number on the left of the decimal point is to be divided repeatedly by 16 and the part an the right of the decimal point is to be repeatedly multiplied by 16. This will be clear from the examples of conversion of $(579.26)_{10}$ into hex equivalent. Split the number into two parts, 579 and .26.



Thus, $(579)_{10} = (243)_{16}$

Fig. Decimal to hex conversion

Now .26 is converted into hex number as follows:



Thus $579.26_{10} = 243.428_{16}$

Example 1: Convert 427 to hexadecimal

427 % 16 = 26 remainder 11(B) (LSB)

26 % 16 = 1 remainder 10(A)

1 % 16 = 0 remainder 1 (MSB)

Writing from MSB to LSB = 1AB

Thus $(427)_{10} = (1AB)_{16}$

Example 2: Convert 2748 to hex

2748 % 16 = 171 remainder 12 (C)

171 % 16 = 10 remainder 11 (B)

10 % 16 = 0 remainder 10(A) (MSB)

Writing from MSB to LSB = ABC

Thus $(2748)_{10} = (ABC)_{16}$

Example 3: Convert $(0.538)_{10}$ to hexadecimal equivalent

0.538 x16 = 8.608 integral part 8

0.608 x16 = 9.728 integral part 9

0.728 x16 = 11.648 integral 11 i.e. B

0.648 x16 = 10.368 integral 10 i.e. A

By writing the integral part from top to bottom $(0.538)_{10} = (0.89BA)_{16}$

BINARY TO DECIMAL CONVERSION

- Binary number can be converted into its decimal equivalent, by simply adding the weights of various positions in the binary number which have bit 1.

  Example 1:

  Find the decimal equivalent of the binary number $(11111)_2$ The equivalent decimal number is

  $=1x2^4+1x2^3+1x2^2+1x2^1+1x2^0$

  $=16+8+4+2+1$

  $= (31)_{10}$

- To differentiate between numbers represented in different number systems, either the corresponding number system may be specified along with the number or a small subscript at the end of the number may be added signifying the number system. Example $(1000)_2$ represents a binary number and is not one thousand.

- **Example 2:**

  Convert $(100011.101)_2$ to decimal

  $= 2^5+0+0+0+2^1+2^0+2^{-1}+0+2^{-3}$

  $=32+2+1+0.5+0.125$

  $= (35.625)_{10}$

  Consider the following examples.

  1111.00 = 15

  11110.0 = 30

  111100.0 = 60

  From these examples, it is clear that if the binary point is shifted towards right side, then the value of the number is doubled. Now consider the following examples.

  111.100 = 7.5

  11.1 100 = 3.75

  1.1 1100 = 1.875

  From these examples it is clear that if the binary point is shifted towards the left side, then the value of the number is halved.

OCTAL TO DECIMAL CONVERSION

As has been done in case of binary numbers, an octal number can be converted into its decimal equivalent by multiplying the octal digit by its positional value.

Example 1: Convert 203 into decimal number.

$203_8 = 2 \times 8^2 + 0 \times 8^1 + 3 \times 8^0$

$= 128 + 0 + 3$

$= (131)_{10}$

Example 2: Convert 36.48 into decimal number.

$36.4_8 = 3 \times 8^1 + 6 \times 8^0 + 4 \times 8^{-1}$

$= 24 + 6 + 0.5$

$= (30.5)_{10}$

HEX TO DECIMAL CONVERSION

Hex to decimal conversion is done in the same way as in the cases of binary and octal to decimal conversions. A hex number is converted into its equivalent decimal number by summing the products of the weights of each digit and their values. This is clear from the example of conversion of $514.AF_{16}$ into its decimal equivalent.

$514.\ AF_{16} = 5 \text{ X } 16^2 + 1 \text{ X } 16^1 + 4 \text{ X } 16^0 + 10 \text{ X } 16^{-1} + 15 \text{ X } 16^{-2}$

$= 1280+16+4+0.625+0.0586$

$= (1300.6836)_{10}$

Example 1: Convert BE to DECIMAL

BE Hexadecimal to DECIMAL

B x 16 + E x 16

$B \times 16^1 + E \times 16^0$

11 x 16 + 14 x 1

176 + 14

190

Therefore $(BE)_{16} = (190)_{10}$

Example 2: Convert BAD to DECIMAL

BAD Hexadecimal to Decimal

Bx16+Ax16+Dx16

$Bx16^2+Ax16^1+Dx16^0$

Bx256 + Ax16 + Dx1

11x256 + 10x16 + 13x1

2816 + 160 + 13

2989

Therefore $(BAD)_{16} = (2989)_{10}$

OCTAL TO BINARY CONVERSION

In the octal number system the highest octal digit i.e.7 can be expressed as a 3-bit binary number. Therefore, all the octal digits have to be represented by n 3-bit binary number. The binary equivalent of each octal digit is shown in Table 1.2. The main advantage of the octal number system is the easiness with which any octal number can be converted into its binary equivalent.

Octal digit	3-bit binary equivalent
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Table 1.2 Binary equivalent of octal digit

Using this conversion of octal digit into 3-bit binary number, any octal number can be converted into its binary equivalent by simply replacing each octal digit by a 3-bit binary number. For example, conversion of 567, into its binary equivalent is: 567 = 101 110 111

Example 1: Convert $(123)_8$ to binary

Convert each digit or number using three bits separately and then combine

For example, 123 can be converted as

1=001, 2=010 and 3=011

Thus $(123)_8 = (001010011)_2$

Delete zeroes $(123)_8 = (1010011)_2$

Example 2: Convert 407_8 to BINARY

4 = 100, 0 = 000 and 7 = 111

Therefore combining all terms

$(407)_8 = (100000111)_2$

BINARY TO OCTAL CONVERSION

A binary number can be converted into its octal equivalent by first making groups of 3-bits starting from the LSB side. If the MSB side does not have 3 bits, then add 0s to make the last group of 3 bits. Then by replacing each group of 3 bits by its octal equivalent, a binary number can be converted into its binary equivalent. For example, consider the conversion of 1100011001, into its octal equivalent as follows:

= 001 100 01 1 001 [As the MSB side does not have 3 bits, we have added two 0's to make the last group of 3 bits]

= 1 4 3 1

Thus, $(1100011001)_2 = (1431)_8$

Example: Convert 1101101101_2 to octal

Divide into 3 bits group:	1	101	101	101
Add zero to the left :	001	101	101	101
	1	5	5	5

Therefore combining all terms: $(1101101101)_2 = (1555)_8$

HEX TO BINARY CONVERSION

As in octal number system, a hex number is converted into its binary equivalent by replacing each hex digit by its equivalent 4-bit binary number. This is clear from the following example:

$$\begin{aligned}(BA6)_{16} &= B \quad A \quad 6 \\ &= 1011 \quad 1010 \quad 010 \\ &= (101110100110)_2\end{aligned}$$

Example 1: Convert $1AB_{16}$ to binary

1=0001, A=1010 and B=1011

Thus $(1AB)_{16} = (000110101011)_2$

$(1AB)_{16} = (110101011)_2$

Example 2: Convert $FACE_{16}$ to Binary

F	A	C	E
15	10	12	14
1111	1010	1100	1110

Therefore combining all terms

$(FACE)_{16}=(1111101011001110)_2$

BINARY TO HEX CONVERSION

By a process that is reverse of the process described in the above section, a binary number can be converted in to its hex equivalent. Starting from the LSB side, group the binary number bits into groups of lour bits. If towards the MSB side, the numbers of bits is less than four, then add zeros on the left of the MSB so that the group of four is complete. Replace each group by its equivalent hex digit. This is clear from the following example:

$$(1001101110)_2 = 0010 \quad 0110 \quad 1110$$
$$= 2 \quad 6 \quad E$$
$$= (26E)_{16}$$

Example 1: Convert $(11111010011)_2$ to HEX

Divide the binary numbers into 4 bits from right to left.

Zero can be added to the left

For example, 011111010011 can be divided as 01111101 0011

0111 = 7, 1101=D and 0011=3

Thus $(011111010011)_2 = (7D3)_{16}$

Example 2: Convert 1111101101 to hexadecimal

Divide into 4 bits group from right

11 1110 1101

Add zero to the left

0011	1110	1101
3	14(E)	13(D)

Therefore combining all terms

$(001111101101)_2 =(3ED)_{16}$

HEX TO OCTAL CONVERSION

Each digit of the hex number is first converted into its equivalent four bit binary number. Then the bits of the equivalent binary number are grouped into groups of three bits. Then each group is replaced by its equivalent octal digit to get the octal number.

For example:

$(5AF)_{16}$ = 0101 1010 1111
= 010110101111
=010 110 101 111
= 2 6 5 7
$=(2567)_8$

OCTAL TO HEX CONVERSION

For octal to hex conversion, just reverse the process described in the section above. This is clear from the following example:

$(5457)_8$ = 101 100 101 111
= 1011 0010 1111
= B 2 F
$= (B2F)_{16}$

1's complement

The 1's complement of a binary number is obtained just by changing each 0 to 1 and each 1 to 0.

Binary number	1	0	1	1	1	0	1	0
	↓	↓	↓	↓	↓	↓	↓	↓
1-complement	0	1	0	0	0	1	0	1

Fig. 1's complement

2's complement

The 2's complement of a binary number is obtained adding 1 to the 1's complement of this number:

2's complement = 1's complement+1
Binary number 1 0 1 1 1 0 1 0
1's complement 0 1 0 0 0 1 0 1
+ 1
2's complement 0 1 0 0 0 1 1 0

There is a simple method to obtain the 2's complement:

- Beginning with the LSB, just write down bits as they are moving to left till the first 1, including it. Substitute the rest of bits by their 1's complement.

Find the 1's complement and 2's complement of the following:

1. **10110**

   1's complement: 01001

   2's complement: 01001

   \+ 1

   01010

2. **110011**

   1's complement: 001100

   2's complement: 001100

   \+ 1

   001101

3. **0101010**

   1's complement: 1010101

   2's complement: 1010101

   \+ 1

   1010110

R'S COMPLEMENT

For a number system having its base/radix as **r**, we can define two types of complement for the corresponding number system which are as follows: r's complement and (r-1)'s complement

- Given a positive number N in base r with an integer part of n digits, the r's complement of N is defined as $\mathbf{r^n - N}$ where N is not Zero.
- For example, the 10's complement of $(52520)_{10}$ is

  $10^5 - 52520 = 100000 - 52520 = 47480$
- The 2's complement of $(101100)_2$ is

  $(2^6)_{10} - (101100)_2$

  $= (1000000 - 101100)_2 = 010100$

(R-1)'S COMPLEMENT

- Given a positive number N in base r with an integer part of n digits and a fraction part of m digits, the (r-1)'s complement of N is defined as $(\mathbf{r^n - 1) - N}$
- For example, the 9's complement of $(52520)_{10}$ is 10^5 -1 – 52520

  = 9999 – 52520 = 47479
- The 1's complement of $(101100)_2$ is

  $(2^6 - 1)_{10} - (101100)_2$

  $= (111111 - 101100)_2 = 010011$
- The 1's complement of a binary number is simpler to form: the 1's are changed to 0's and 0's are changed to 1's.

Find the 9's and 10's complement of a given number

1. **5321**

   9's complement: 9999-5321= 4678

   10's complement: 9's +1= 4678+1=4679
2. **88920**

   9's complement: 99999-88920= 11079

   10's complement: 9's +1= 11079+1= 11080
3. **3494**

   9's complement: 9999-3494= 6505

   10's complement: 9's +1= 6506

BCD Code

In BCD (BCD stands Binary coded decimal) code, each digit of a decimal number is converted in to its binary equivalent. The largest decimal digit is 9; therefore, the largest binary equivalent is 1001. This is illustrated as follows

$$951\ 10 = 1001\ \ 0101\ \ \ 0001$$
$$= (100101010001)_{BCD}$$

Decimal	Binay (BCD) 8 4 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

Table: BCD Table

ASCII CODE

The word ASCII is run acronym of American Standard Code for Information Interchange. This is the alphanumeric code most widely used in computers. The alphanumeric code is one that represents alphabets, numerical numbers, punctuation marks and other special characters recognized by a computer. The ASCII code is a 7-bit code representing 26 English alphabets, 0 through 9 digits, punctuation marks, etc. A 7-bit code has 27 = 128 possible code groups which arc quite sufficient. Table below shows ASCII table:

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

ERROR CORRECTION CODE

These codes are used to detect the errors present in the received data bitstream. These codes contain some bits, which are included appended to the original bit stream. These codes detect the error, if it is occurred during transmission of the original data bitstream. Example − Parity code, Hamming code.

PARITY CODE

It is easy to include append one parity bit either to the left of MSB or to the right of LSB of original bit stream. There are two types of parity codes, namely even parity code and odd parity code based on the type of parity being chosen.

EVEN PARITY CODE

The value of even parity bit should be zero, if even number of ones present in the binary code. Otherwise, it should be one. So that, even number of ones present in even parity code. Even parity code contains the data bits and even parity bit.

The following table shows the even parity codes corresponding to each 3-bit binary code. Here, the even parity bit is included to the right of LSB of binary code.

Binary Code	Even Parity bit	Even Parity Code
000	0	0000
001	1	0011
010	1	0101
011	0	0110
100	1	1001
101	0	1010
110	0	1100
111	1	1111

Here, the number of bits present in the even parity codes is 4. So, the possible even number of ones in these even parity codes are 0, 2 & 4.

- If the other system receives one of these even parity codes, then there is no error in the received data. The bits other than even parity bit are same as that of binary code.
- If the other system receives other than even parity codes, then there will be an errors in

the received data. In this case, we can't predict the original binary code because we don't know the bit positions of error.

Therefore, even parity bit is useful only for detection of error in the received parity code. But, it is not sufficient to correct the error.

ODD PARITY CODE

The value of odd parity bit should be zero, if odd number of ones present in the binary code. Otherwise, it should be one. So that, odd number of ones present in odd parity code. Odd parity code contains the data bits and odd parity bit.

The following table shows the odd parity codes corresponding to each 3-bit binary code. Here, the odd parity bit is included to the right of LSB of binary code.

Binary Code	Odd Parity bit	Odd Parity Code
000	1	0001
001	0	0010
010	0	0100
011	1	0111
100	0	1000
101	1	1011
110	1	1101
111	0	1110

Here, the number of bits present in the odd parity codes is 4. So, the possible odd number of ones in these odd parity codes is 1 & 3.

- If the other system receives one of these odd parity codes, then there is no error in the received data. The bits other than odd parity bit are same as that of binary code.
- If the other system receives other than odd parity codes, then there is an errors in the received data. In this case, we can't predict the original binary code because we don't know the bit positions of error.

Therefore, odd parity bit is useful only for detection of error in the received parity code. But, it is not sufficient to correct the error.

GRAY CODE

Gray Code is a form of binary that uses a different method of incrementing from one number to the next. With Gray Code, only one bit changes state from one position to another. This feature allows a system designer to perform some error checking (i.e., if more than one bit changes, the data must be incorrect).

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

Table. Gray code