COMPUTER FUNDAMENTALS

COMPUTER BASICS:

Number System... *We know that the computer is a high speed-computing device and is able to understand and store information we feed into it. The information can be represented and stored in different ways. Such different methods are based on number systems and are discussed in this chapter.*

A number can be represented in many ways and the value of any number is determined by the following factors.

Position of digit... Normally, the last significant digit is placed at extreme right of the number and the most significant digit is placed at the leftmost position.

Value of digit... We know that value of the number is based on each digit in it, for this, weights are Basically, there are two types of the values as follows.

(1) Face value, and
(2) Place value.

Face value. Face value of the number is simply the number

Place value. The place value of a number relates to weights attached to the position of the number.

Base of the systems... Base is a number associated with each system. It is mainly determined by the number of symbols in the system. For example, the base for the decimal system is 10, and it has the symbols 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. The base is usually denoted as a suffix of the number. If no suffix is mentioned, the base is assumed to be 10.

Commonly used number systems

System	Base	Symbols	Weight
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	10
Binary	2	0, 1	2
Octal	8	0, 1, 2, 3, 4, 5, 6, 7	8
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9 A, B, C, D, E, F	16

Decimal system… The number system that is normally used is decimal system. While assigning weights to each digit of a number, rightmost digit gets the weight of unity and the successive digits to its usually have weights 10^1, 10^2 10^3 and so on. Each digit is multiplied by its weight and the sum of product provides value of the number.

Illustrative Examples

1. Decimal number 7537_{10} may be expressed as

$7 \times 10^3 = 7000$

$5 \times 10^2 = 500$

$3 \times 10^1 = 30$

$7 \times 10^0 = 7$

Thus $7 \times 10^3 + 5 \times 10^2 + 3 \times 10^1 + 7 \times 10^0 = 5737_{10}$

It should be noted that the digit 7 has a face value and a place value $10^3 = 1000$. Thus, the digit represents the product 7×10^3.

2. Decimal number 65355_{10} may be expressed as

$6 \times 10^4 = 60000$

$5 \times 10^3 = 5000$

$3 \times 10^2 = 300$

$5 \times 10^1 = 50$

$5 \times 10^0 = 5$

Thus, $6 \times 10^4 + 5 \times 10^3 + 3 \times 10^2 + 5 \times 10^1 + 5 \times 10^0 = 65355_{10}$

Binary system... *The binary system may be defined as a number system, having two symbols 0 and 1.* The base of a binary number system is 2 and the symbols 0 and 1 are termed as binary digits or bits.

Illustrative Examples

3. 1101_2 is a binary number, to find the decimal value of the binary number, powers of two (2) are used as weights in a binary system and is as follows:

$1 \times 2^3 = 8$

$1 \times 2^2 = 4$

$0 \times 2^1 = 0$

$1 \times 2^0 = 1$

Thus the decimal value of 1101_2 is $1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13$.

Remark: While assigning weights to each digit of a number, rightmost digit gets the weight of unity and the successive digits to its left usually have weights 2^1, 2^2, 2^3 and so on. Each digit is multiplied by its weight and the sum of product provides the decimal value of the binary number 1101_2

4. 11101_2 is a binary number, the equivalent decimal value is obtained in the following manner:

$1 \times 2^4 = 16$

$1 \times 2^3 = 8$

$1 \times 2^2 = 4$

$0 \times 2^1 = 0$

$1 \times 2^0 = 1$

Thus the decimal value of 11101_2 is $1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 29$.

Octal system... The octal system is the number system consisting of eight symbols 0, 1, 2, 3, 4, 5, 6 and 7.

The base of an octal number system is 8. Thus the weight assigned to each digit in octal system is power of 8.

Illustrative Examples

5. The decimal value of the octal number 176 can be obtained in the following manner:

$1 \times 8^2 = 64$

$7 \times 8^1 = 56$

$6 \times 8^0 = 6$

Thus the decimal value of 11101_2 is $1 \times 8^2 + 7 \times 8^1 + 6 \times 8^0 = 126$.

6. The decimal value of the octal number 1116 can be obtained in the following manner:

$1 \times 8^3 = 512$

$1 \times 8^2 = 64$

$1 \times 8^1 = 8$

$6 \times 8^0 = 6$

Thus the decimal value of 1116 is $1 \times 8^3 + 1 \times 8^2 + 1 \times 8^1 + 6 \times 8^0 = 590$.

Hexadecimal system. *The number system having symbols such as 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F is called hexadecimal system. It has 16 as the basis.*

Illustrative Examples

7. The decimal value of the hexadecimal number 84A8 is obtained as follows:

$8 \times 16^3 = 32768$

$4 \times 16^2 = 1024$

$A \times 16^1 = 160$

$8 \times 16^0 = 8$

Thus the decimal value of 84A8 is $8 \times 16^3 + 4 \times 16^2 + A \times 16^1 + 8 \times 16^0 = 33960$.

8. The decimal value of hexadecimal number 21E3 is obtained as follows:

$2 \times 16^3 = 8192$

$1 \times 16^2 = 256$

$E \times 16^1 = 160$

$3 \times 16^0 = 3$

Thus the decimal value of 21E3 is $2 \times 16^3 + 1 \times 16^2 + E \times 16^1 + 3 \times 16^0 = 8675$.

COMPUTER MEMORY

Every piece of information that is stored within the computer's memory is encoded as some unique combination of zeros and ones. These zeros and ones are called bits (binary digits). Each bit is represented by an electronic device that is in some sense, either "off" (zero) or "on" (one).

Small Computers have memories that are obtained into 8-bit multiples called bytes. Normally 1 byte represents a single character viz. a letter, a single digit or a punctuation symbol. Further, an instruction may occupy 1, 2 and 3 bytes, and a single numerical quantity may occupy anywhere form 2 to 8 bytes, depending on the precision and type of number. The size of a computer memory is normally expressed as some multiple of 2^{10} = 1024 bytes. This is referred to as1 K. Small computers have memory from 64K to 1024K bytes.

In computer technology, a word is made up of certain number of bytes. The number of bytes making up a work depends on the computer. In addition, the word length may be defined as number of bits it contains. Thus computers are classified into 8-bit, 16-bit, 32-bit and so on.

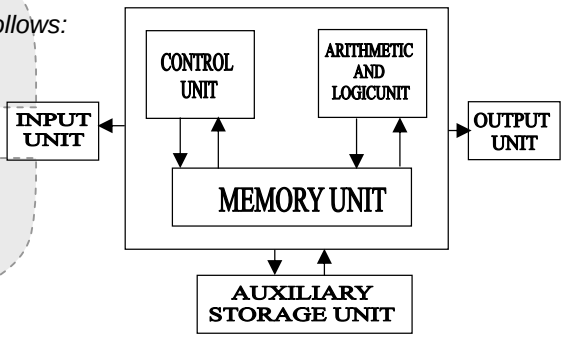
The most important coding formats are

Binary Coded Decimal (BCD)
Extended Binary Coded Decimal Interchange Code (EBCDIC)
American Standard Code for Information Interchange (ASCII)

COMPUTER ORGANIZATION

There are four primary units of a computer and are as follows:

(1) Input device
(2) Output device
(3) CPU (Central Processing Unit) and
(4) Auxiliary storage device



The block diagram shows the primary units or components of a computer.

Input devices: Input/output unit is the important tool to communicate between user and machine. An input unit accepts our instructions and passes them on to the Central Processing Unit (CPU) Some of the input devices are card reader, keyboard, mouse floppy reader and so on.

Output devices: An output is very essentially needed device to produce the processed results and data from the computer to microfilm.

Central Processing Unit (CPU): It has been already stated that, Central Processing Unit consists of primary memory, Control Unit, and Arithmetic and Logic Unit (ALU).

Primary memory stores program, data and processes results. Memory is actually made up of a large number of cells, each of which is capable of storing one binary digit. These cells can be arranged in the form of word. Each word has a unique address to identify and can store a sequence of bits. Its parts are discussed below:

1) A part of the primary memory is termed as **Read Only Memory (ROM).** ROM is chip containing special electronic circuit and is particularly designed to store instructions permanently. The programs, which are stored in ROM, can be executed by various commands. These programs are as permanent as hardware, though they perform the functions of software. Due to this reason, they are called Firmware.

   It is important to note that, contents of ROM can be erased as well as reconstructed, due to recent development in fabricating the computer memory. These memories are called **Erasable and Programmable (EPROM).**

2) The other part of the primary memory is called **Random Access Memory (RAM).** The retrieval and storage of a word in RAM is independent of the address of the word. The contents or RAM are volatile or temporary, since the contents of RAM are erased when computer power fails or switched off.

   *To overcome this problem, we have another secondary storage unit known as **Auxiliary Storage** and it stores the data and programs permanently. These are two common secondary storage devices. viz., magnetic tape drive and magnetic disk drive, e.g., hard disk, floppy disk, CD-ROM.*

Organization of memory

Computer memory stores the data, whatever we passed on to it. It does not know the difference between data and programs. Only the microprocessor can understand such differences.

We know that memory is made up of a large number of cells. A **memory cell** can be defined as a device, which can store a symbol selected from a set of symbols. A memory cell may also be referred to as a *memory location*. It is important to note that, unique addresses can be assigned for each memory cell using numeric label. Information can be stored and retrieved from memory location through its *address*. The difference between memory location and address is, memory location holds a byte of information whereas its address serves as a label to identify it.

Every personal computer has certain special circuits. These circuits are called registers. The address of memory location is placed in the register, known as **Memory Address Register (MAR).** This register has a circuit connected to it. Based on the information contents of MAR, whether the operation specified is read or write, the circuit initiates appropriate action.

Another storage space is called **Buffer**. It is a part of primary memory and holds the information on a temporary basis.

Another feature of memory organization is **Bus**. It is a set of wires that interlinks the various parts of the computer. It carries information from one part to another

ALGORITHMS AND FLOW CHARTING

*The term **Algorithm** may be defined as a sequence of instructions designed in such a way that if the instructions are executed in the specified sequence, the desired results can be obtained. Further, the instructions should be unambiguous and the result should be obtained after a finite number of executed steps, It should not repeat one or more instructions infinitely and algorithm must terminate.*

In other words… the algorithm represents the logic of the processing to be performed. The word algorithm originates from the word algorism, which means the process of doing arithmetic with Arabic numerals. Later, the word algorism combined with the word arithmetic to become algorithm.

Let us consider the cost of construction of a house. The cost depends on

1. Plinth area,
2. Quality and quality of material to be used, and
3. Labour required

These factors are design variables. As the plinth area increases, the cost of construction also increases. Restrictions on values of design variables are known as **constraints.** Within these constraints the cost of building can be evaluated.

Properties of Algorithms...

Finiteness: Whenever the algorithm is mechanically executed, it comes to an end finite number of steps. The step may consist of assignment, decision or even repetition of the above.

Definiteness: Each step is clearly defined also successive step to be executed are clearly defined.

There is no ambiguity in any of the above.

Effectiveness: All the operations used in the algorithms are basic and are such that those can be actually performed.

Generality: All the rules pertaining to algorithm are complete so that all the problems for a particular situation can be solved.

Input-Output…

An algorithm has always got some inputs or initial set of conditions or data. The output forms an intermediate step or final step.

Types of algorithms...

Algorithms are classified according to the order of execution and the logic included Based on the control logic involved in an algorithm, it is classified as…

1) Deterministic,
2) Non-Deterministic,
3) Random, and
4) Others.

Remark: An algorithm is said to be indirect, if the number of repetitions is unknown in the beginning

Example: To find out whether the number is prime or not.

Writing Algorithms…

In writing algorithms we will be using mnemonic variable names or identifiers such a **NUM** to denote *numerator*, **DEN** to denote *denominator*, **INT** to denote *interest* etc. In addition to mnemonic variable names we will use the assignment symbol **<--** and the following **constructs (pseudo codes):**

1) IF-THEN-ELSE
2) REPEAT-UNTIL
3) WHILE-DO
4) FOR-DO
5) CASE-OF

IF – ELSE CONSTRUCT... This construct is used to provide selection of actions. The general form of this construct is:

If condition

then Step I

else Step II

If the condition is true, step I is executed. If the condition is not true, step II is executed.

Ex. Write an algorithm to find the modulus of a real number.

Sol. Step I get x

Step II If $x \geq 0$

then required value is **x**

else required value is **–x**

Ex. Write an algorithm to find whether the square root of a real number is real or imaginary.

Sol. Step I get x

Step II If $x \geq 0$

then output "square root is real"

else output "square root is imaginary"

REPEAT – UNTIL CONSTRUCT... This construct is used when repetition of certain actions is required. All statements appearing between the repeat and until are executed until the condition mentioned after the word until is true. The general form of this construct is

Repeat

S

until condition

When this construct is executed, the portion of the algorithm appearing between the words repeat and until is executed and then the condition is tested. If the conditions is true, the portion of algorithm appearing after the word until is executed; If the condition is false, again S is executed and then the condition is tested, and so on. Every execution of S modifies some variables in the algorithm and eventually after some repetitions, the conditions become true. This completes the exaction of the repeat until construct and the execution proceeds to the portion of the algorithm appearing after the word until. Following example illustrates the use of this construct.

Ex. Write an algorithm to find the sum of the even natural numbers less than or equal to 20.

Sol. Step I Sum ← 0

Step II I ← 1

Step III Repeat

Sum ← Sum + 2I

I ← I + 1

Until I > 11

output Sum

WHILE – DO CONSTRUCT... Similar to repeat-until construct this construct is also used to provide repetition of instructions. The general form of this construct is…

While Condition

do S

Where S is a set of instructions

When this construct is executed, condition is tested first if the condition is true, the set S of instructions is executed, and then the condition is tested again until the condition becomes false. If the condition is false execution of S is skipped and the remaining portion of the algorithm appearing after S is executed. In every execution of S the values of some variables in the algorithm are modified and eventually after some repetitions, the conditions becomes false. This completes the execution of while-do construct and the execution proceeds to the portion of the algorithm appearing after S.

Remark…

1. As discussed, if the condition in while-do construct is not true the set S of instructions is not executed even once whereas in the use of repeat – until construct the portion of the algorithm between repeat and until is executed at least once.

2. The execution of while do construct is over when the condition therein is found false whereas Following example illustrates the use of while do construct.

Ex. Write an algorithm to find the sum of the even natural numbers less than or equal to 20.

Sol. Step I Sum ← 0

Step II I ← 1

Step III while I < 11

do Sum ← Sum + 2I

Step IV output Sum

FOR – DO CONSTRUCT... The general form of this construct is:

For identifiers = initial value **to** final value **by** increment

do S

Where S is the set of instructions

When this construct is executed the identifier takes initial value and the portion S of the algorithm gets executed. This identifier is then increased by the increment find value. As soon as the identifiers assumes a values greater than the final value the execution of S is not done and the execution proceeds to the portion of the algorithm appearing after the word do.

For I = 2 to 11 by 3

do S

I initially takes value 2 and the portion S of algorithm is executed. Now the identifier I takes the next value 5 and S is executed again. Similarly S is executed when I takes value 8 and 11. As soon as I takes the next value 14 the execution proceeds to the portion of the algorithm appearing after the word do.

All these constitute a language to present an algorithm, which is not a programming language, but it has desirable characteristic of a good programming language. This is called as **Pseudo Language**.

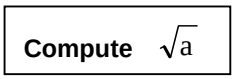
Flow Charts…

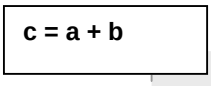
Structure of flow chart. A flow chart is a pictorial representation of an algorithm in which boxes of different shapes are used to denote different type of operations.

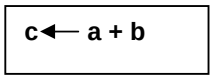
It is important to note that, for a beginner, it is recommended that a flow chart be drawn first in order to reduce the number of errors and omissions in the program. It is a good practice to have a flow chart, which may help during the testing of the program as well as while incorporating further modifications in the program.

Entry/Exit Boxes…

General processing Box...

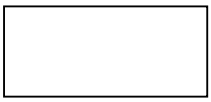




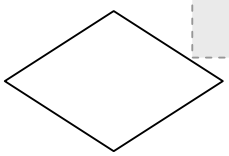


The arrow within box indicates that the contents or the values of the quantity at its head is replaced by the quantity at its tail a + b.

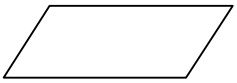
Assignment Symbol…



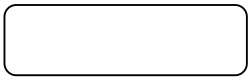
Decision Symbol…



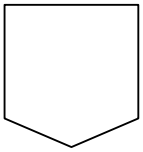
Input/output Symbol…



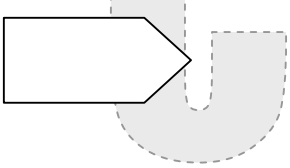
Terminal Symbol…



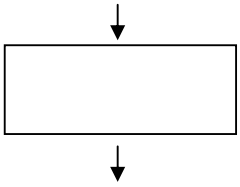
Off-page Connector Symbol…



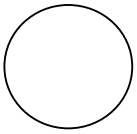
Modification Symbol…



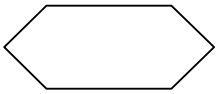
Module Symbol…

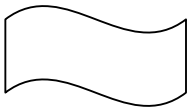


Connection Symbol…



Group Instruction Symbol...





Depending upon the activity represented, the flow charts may be classified as:

Macro Flow Charts **:** Those representing coarse level activity.

Micro Flow Charts **:** Those representing fine level of activity

Pica Flow Charts **:** Those representing super fine level of activity

LANGUAGES

Low level language. *Programming languages that normally translate from one source instruction to one object instruction. These languages are machine dependent.*

High-level language. A programming language whose structure is application oriented and is independent of the structure of the computer. Each statement of such a language is translated into many machine language statements.(e.g., FORTRAN, COBOL)

Assembly language. A low level programming language in which mnemonics are used to code operations and alphanumeric symbols are used for addresses. These languages lie between the high-level languages and machine language.

Machine language. A low level language that is directly understandable by the computer system. Each model of a computer has unique machine language. (The Computer understands only binary digits namely 0's and 1's)

CONVERSION OF NUMBERS

Decimal to Binary : To convert a decimal number to a binary system, each decimal value and the successive quotients are divided by two, till we get a zero as quotient. The remainder for each division is written from bottom to top, and is called as binary number equivalent to the decimal.

Illustrative Examples

9. The decimal 747 is converted into its binary form as follows:

Division	Quotient	Remainder
747/2	373	1
373/2	186	1
183/2	93	0
93/2	46	1
46/2	23	0
23/2	11	1
11/2	5	1
5/2	2	1
2/2	1	0
½	0	1

Thus the binary equivalent of 747 is 1011101011_2.

10. The decimal number 101 is converted into its binary form as follows :

Division	Quotient	Remainder
101/2	50	1
50/2	25	0
25/2	12	1
12/2	6	0
6/2	3	0
3/2	1	1
½	0	1

Thus the binary equivalent of 101 is 1100101_2.

Decimal to Octal. The octal equivalent of a decimal number is obtained by dividing the decimal number as well as successive quotients by 8, till a value of zero quotients is present. The remainders are written from last one to first one, giving the equivalent of the number is the octane form.

Illustrative Examples

11. The decimal number 747 equivalent octal can be obtained in the following manner :

Division	Quotient	Remainder
747/8	93	3
93/8	11	5
11/8	1	3
1/8	0	1

Thus the octal equivalent of 747 is 1353.

12. The decimal number 97 equivalent octal can be obtained in the following manner :

Dvision	Quotient	Remainder :
97/8	12	1
12/8	1	4
1/8	0	1

Thus the octal equivalent of 97 is 141.

Decimal to Hexadecimal: the hexadecimal equivalent of a decimal is found by dividing the decimal number and its successive quotients repeatedly by 16, till a zero quotient is obtained. The remainders written from the final one to first one, provide the equivalent in hexadecimal system.

Illustrative Examples

13. The decimal number 747 is converted to its hexadecimal form as given below :

Division	Quotient	Remainder
747/16	46	11(B)
46/16	2	14(E)
2/16	0	2

Thus the hexadecimal equivalent of 747 is 2EB.

14. The decimal number 137 is converted to its hexadecimal form as given below :

Division	Quotient	Remainder
137/16	8	9
8/16	0	8

Thus the hexadecimal of 137 is 89.

Binary to Decimal: To convert a binary system number to a decimal system, the digits are obtained first. The sum of the products of the value of the digit in the decimal system and the digit gives the decimal equivalent.

Illustrative Examples

15. The binary number 101101_2 is converted into its decimal equivalent as shown below :

Binary number	value of the digit (in decimal system)
1	2^5
0	2^4
1	2^3
1	2^2
0	2^1
1	2^0

Thus the decimal value of the binary number 101101_2 is

$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 45$

16. The binary number 111111_2 is converted into its decimal equivalent as shown below :

Binary number	Value of the digit (in decimal system)
1	2^5
1	2^4
1	2^3
1	2^2
1	2^1
1	2^0

Thus the decimal value of the binary number 101101_2 is

$1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 + 1 \times 2^1 + 1 \times 2^0 = 63$.

Binary to Octal: a number in the binary system is converted to its equivalent in octal system, by partitioning the binary number into 3 bit blocks, beginning with rightmost bit. Each block is replaced by its corresponding octal digit. The octal equivalent to binary system is obtained by writing the octal digit from left to right.

Illustrative Examples

17. The binary number 101001_2 is converted into its octal equivalent as follows :

$\underbrace{101}_{5}\ \underbrace{001}_{1}$ Thus the octal equivalent of the binary number 101001_2 is 51.

18. If the number of bits in the binary number is not a multiple of 3, then zeros may be introduced at the left of a binary number to get 3-bit blocks.

For example, the binary number 10101_2 is converted into its octal form as follows :

$\underbrace{010}_{2}\ \underbrace{101}_{5}$ Thus the octal equivalent of the binary number 10101_2 is 25.

Binary to hexadecimal: The hexadecimal equivalent to its binary number system is obtained by first partitioning the binary number digits into 4 – bit blocks beginning with rightmost bit. Each block is then introduced by the corresponding hexadecimal values, the hexadecimal values are then written from left one to right one to obtain the equivalent hexadecimal value.

Illustrative Examples

19. The binary number 10101001_2 is converted into its hexadecimal equivalent as follows :

$\underbrace{1010}_{A}$ $\underbrace{1001}_{9}$ Thus the hexadecimal equivalent of the binary number 10101001_2 is A9.

20. It is important to note that, in case of shortage of 4-bit blocks, zeros can be added at the left end to get 4-bit blocks.

For examples, the binary number 101001_2 is converted into its hexadecimal number in the $\underbrace{0010}_{2}$ $\underbrace{1001}_{9}$ manner. Thus the hexadecimal equivalent of the binary number 101001_2 is 29.

Octal to Decimal: To convert a number in octal system to its equivalent in the decimal system, the value of the digits are first obtained. The sum of products of the value of the digit in the decimal system and the digits gives the decimal equivalent.

Illustrative Examples

21. The octal number 2357 is converted into its decimal equivalent as shown below :

Octal number	**Value of the digit (in decimal system)**
2	8^3
3	8^2
5	8^1
7	8^0

Thus the decimal equivalent of the octal number 2357 is $2 \times 8^3 + 3 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 = 1263$.

22. The octal number 871 is converted into its decimal equivalent as shown below :

Octal number	**Value of the digit (in decimal system)**
8	8^2
7	8^1
1	8^0

Thus the decimal equivalent of the octal number 569 is $8 \times 8^2 + 7 \times 8^1 + 1 \times 8^0 = 569$.

Octal to Binary: The octal equivalent of a binary system is performed by first replacing all octal digits by 3-bit blocks are written from left to rightmost to get octal equivalent.

Illustrative Examples

23. The octal number 2563 is converted into the binary equivalent as described below:

$\overbrace{010}^{2}$ $\overbrace{101}^{5}$ $\overbrace{110}^{6}$ $\overbrace{011}^{3}$ Thus the binary number equivalent of 2663 is 1010110011_2 .

24. The octal number 1356 is converted into its binary equivalent as described below:

$\overbrace{001}^{1}$ $\overbrace{011}^{3}$ $\overbrace{101}^{5}$ $\overbrace{110}^{6}$ Thus the binary number equivalent of 1356 is 1011101110_2.

Octal to Hexadecimal: To convert a number in octal system to its equivalent hexadecimal, first each octal digit is replaced by its binary equivalent of the octal. The binary digits are then arranged by 4–bit blocks starting from rightmost to left.

In this case, we may end up with blocks less than 4-bits, and then zeros are introduced at the left to obtain 4-bit blocks. The hexadecimal equivalent of these 4-bit blocks are then obtained. These hexadecimal digits written from the leftmost to the right gives the hexadecimal equivalent of the octal number.

Illustrative Examples

25. The octal number 463 is converted into its equivalent hexadecimal in the following steps...

Step 1. Obtain binary equivalent of the octal number 463.

$\overbrace{100}^{4}\ \overbrace{110}^{6}\ \overbrace{011}^{3}$ Therefore, binary equivalent is 100110011_2.

Step 2. Arrange the binary equivalent 100110011_2 into 4-bits blocks (Introduce 3-zeros at the left, to get 4-bit blocks) and replace the 4-bit blocks by hexadecimal equivalent as shown below:

$\underbrace{0001}_{1}\ \underbrace{0011}_{3}\ \underbrace{0011}_{3}$ Thus the hexadecimal equivalent of the octal number 463 is 133.

26. The octal number 231 is converted into its equivalent hexadecimal in the following steps.

Step 1. Obtain binary equivalent of the octal number 231.

$\overbrace{010}^{2}\ \overbrace{011}^{3}\ \overbrace{001}^{1}$ Therefore, binary equivalent is 10011001_1.

Step 2. Arrange the binary equivalent 10011001_2 into 4-bit blocks and replace the 4-bit blocks by hexadecimal equivalent as indicated below:

$\underbrace{1001}_{9}\ \underbrace{1001}_{9}$ Thus the hexadecimal equivalent of the octal number 231 is 99.

Hexadecimal to Decimal: To convert a number in hexadecimal system to its equivalent decimal, first the values of the digits are obtained. The sum of the products of the value of the digits in the decimal system and the digits give the decimal equivalent.

Illustrative Examples

27. The hexadecimal number A59 is converted into its decimal equivalent as follows:

Hexadecimal number	Value of the digit (in decimal system)
A	16^2
5	16^1
9	16^0

Thus the hexadecimal equivalent of the decimal number

A59 is $10 \times 16^2 + 5 \times 16^1 + 9 \times 10^0 = 2649$

28. The hexadecimal number C71 is converted into its decimal equivalent as follows:

Hexadecimal number	Value of the digit (in decimal system)
C	16^2
7	16^1
1	16^0

Thus the hexadecimal equivalent of the decimal number
C71 is $12 \times 16^2 + 7 \times 16^1 + 1 \times 16^0 = 3185$

Hexadecimal to Binary: To convert a hexadecimal number system to its equivalent binary number, first, hexadecimal number is replaced by its 4-bit blocks. The combination of these 4-bit blocks is written from leftmost to right to obtain binary equivalent of a hexadecimal.

Illustrative Examples

29. The hexadecimal number B74 is converted into its binary equivalent as given below:

$\overbrace{1011}^{B}\ \overbrace{0111}^{7}\ \overbrace{0100}^{4}$ Thus the binary equivalent of the hexadecimal number is B74.

30. The hexadecimal number E93 is converted into its binary equivalent as given below $\overbrace{1110}^{E}\ \overbrace{1001}^{9}\ \overbrace{0011}^{3}$
Thus the binary equivalent of the hexadecimal number E93 is 111010010011_2.

Hexadecimal to Octal: To convert a number in hexadecimal system to it equivalent octal system, first hexadecimal number is replaced by its binary equivalent, the 4-bit blocks are then arranged into 3-bit blocks staring from rightmost bit to the leftmost bit. The octal equivalent of these 3-bit blocks are then replaced and written staring from leftmost to right providing the octal equivalent of a hexadecimal number.

Illustrative Examples

31. The hexadecimal number 4B7 is converted to its octal equivalent is the following way…

$\overbrace{0100}^{4}\ \overbrace{1011}^{B}\ \overbrace{0111}^{7}$Thus the binary equivalent of 4B7 is 010010110111_2. Arrange the 4-bit blocks into groups of 3-bit blocks like, $\underbrace{010}_{2}\ \underbrace{010}_{2}\ \underbrace{110}_{6}\ \underbrace{111}_{7}$. Hence the octal equivalent of the hexadecimal number 4B7 is 2267.

32. The hexadecimal number D5 is converted to its octal equivalent in the following way $\overbrace{1101}^{D}\ \overbrace{0101}^{5}$
Thus the binary equivalent of D5 is 11010101_2. Arrange the 4-bit blocks into groups of 3-bits blocks (introduce zero at the left side to make 3-bit blocks)
$\underbrace{011}_{3}\ \underbrace{010}_{2}\ \underbrace{101}_{5}$. Hence the octal equivalent of the hexadecimal number D5 is 325.

ARITHMETIC OPERATIONS ON BINARY

Laws:

Binary Addition

(1) 0 + 0 = 0 (2) 0 + 1 = 1 (3) 1 + 0 = 1

(4) 1 + 1 = 0, with a carry of 1 (5) 1 + 1 + 1 = 1, with a carry of 1.

Example…

Decimal	**Binary**
1	1
+ 1	+ 1
2	10_2

Binary Subtraction

(1) 0 – 0 = 0 (2) 1 – 0 = 1 (3) 1 – 1 = 0

(4) 0 – 1 = 1 with a borrow of 1

Example…

Decimal	**Binary**
25	11001
+ 14	+ 1110
11	1011_2

Binary Multiplication

(1) 0 × 0 = 0 (2) 0 × 1 = 0 (3) 1 × 0 = 0 (4) 1 × 1 = 1

Example …

Decimal	**Binary**
7	111
× 3	× 11
21	111
	1110
	10101_2

Binary Division

It employs the rules of binary subtraction and multiplication. The rules for binary division are:

(1) 0 – 1 = 0 (2) 1 – 1 = 0

Example …

Decimal	Binary
3 2) 6 6 --- 0 ---	11 10) 110 10 --- 010 10 --- 00_2

One's Complement

The 1's complement of a binary number is the number that results when we complement each bit of the number.

Example…

1's Complement of 1 0 1 0 is 0 1 0 1.

1 1 0 1 is 0 0 1 0.

Two's Complement

The 2's complement of a binary number is the number that results when we add 1 to the 1's complement.

For instance, 2's complement of 1111 is (one's complement of 1111) + 1 = 0000 + 1 = 0001.

Addition of 2 binary digits.

A	B	Sum (A + B)	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Subtraction of 2 binary digits.

A	B	Difference (A – B)	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

We have…

(1) 0 + 0 = 0 (2) 0 + 1 = 1 + 0 = 1 (3) 1 + 1 = 10 (4) 1 + 1 + 1 = 11

(5) 0 – 0 = 0 (6) 1 – 0 = 1 (7) 1 –1 = 0 (8) 10 – 1 = 1

When we want to subtract two binary numbers, it can be easily accomplished by taking the 1's complement of the subtrahend and adding it to the minuend.

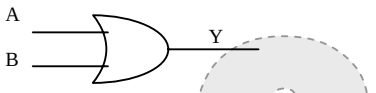
Gates, Logic Symbol and Truth Tables, Types of Gates

1) OR 2) AND 3) NOR 4) NOR 5) NAND 6) EXCLUSIVE–OR (XOR)

OR Gate:

Logical equation **Logical Symbol** **Truth table**

Y = A + B

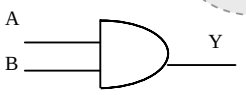


A	**B**	**Y**
0	0	0
0	1	1
1	0	1
1	1	1

AND Gate:

Logical equation **Logical Symbol** **Truth table**

Y = A . B

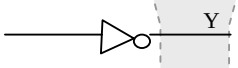


A	**B**	**Y**
0	0	0
0	1	0
1	0	0
1	1	1

NOT Gate :

Logical equation **Logic Symbol** **Truth table**

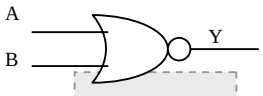
Y = Not A



A	**Y**
0	1
1	0

NOR Gate:

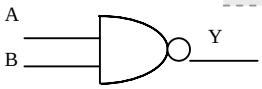
Logical equation **Logical Symbol** **Truth table**



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

NAND Gate:

Logical equation **Logic Symbol** **Truth table**

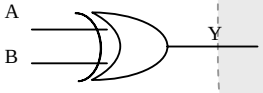


A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

Exclusive–OR Gate (XOR):

Logical equation **Logic symbol** **Truth table**

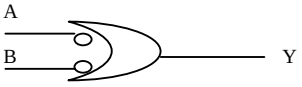
$Y = A \cdot B + A \cdot B$



A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

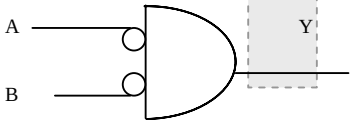
Bubbled–OR (Equivalent to NAND):

Logic symbol



Bubbled–AND (Equivalent to NOR):

Logic symbol



BOOLEAN LAWS

- A . 1 = A
- A . 0 = 0
- A + 1 = 0
- A + 0 = A
- A . Á = 0
- A + Á = 1
- A = A
- A + B = B + A
- A . B = B . A
- A + A. B = A
- A. (A + B) = A
- A + A. B = A + B
- A. (B + C) = A. B + A . C
- A + B . c = (A + B) . (A + C)
- A + (B + C) = (A + B) + C
- A . (B . C) = (A . B) . C
- A + B = A . B
- A . B = A + B De Morgan's laws.

ABBREVIATIONS, ACRONYMS AND EXPANSIONS

1.	ALGOL	- Algorithmic Language
2	ALU	- Arithmetic Logic Unit
3.	ANSI	- American National Standards Institute
4.	APL	- A programming Language
5.	ASCII	- American Standard code for Information Interchange
6.	BASIC	- Beginners All purpose Symbolic Instruction Code
7.	BCD	- Binary Coded Decimal
8.	BIOS	- Basic Input/Output system
9.	CAI	- Computer Aided Instruction
10.	CAM	- Computer Aided Manufacturing
11.	CGA	- Computer Graphics Adapter
12.	COBOL	- Common Business Oriented Language
13.	COM	- Computer Output Microfilm
14.	CP/M	- Control Program/Microprocessor
15.	CPU	- Control Processing Unit
16.	CRT	- Cathode Ray Tube
17.	DBMS	- Data Base Management System

18. DOS - Disk Operating System
19. EBCDIC - Extended Binary Coded Decimal Interchange Code
20. EDP - Electronic Data Processing
21. EGA - Enhanced Graphics Adapter
22. ENIAC - Electronic Numerical Integrator and Calculator
23. EPROM - Erasable Programmable Read Only Memory
24. FORTRAN - Formula Translation
25. GIGO - Garbage-In-Garbage-Out
26. IC - Integrated Circuit
27. 1 K - $2^{10} = 1024$
28. LAN - Local Area Network
29. LSI - Large Scale Integration
30. PL/I - Programming Language One
31. PROM - Programming Read Only Memory
32. RAM - Random Access Memory
33. ROM - Read Only Memory
34. UNIVAC - Universal Automatic Computer
35. VDU - Visual Display Unit