# BCA 1ST Semester

BCA - 101: COMPUTER FUNDAMENTALS AND PROGRAMMING

## UNIT: 2

## What is Bit?

A bit is just a smaller unit of information than a byte, according to the most basic definition. That process is reflected in this symbol, which represents one unit of information representing either a zero (no charge) or a one (full charge) (a completed, charged circuit).

One byte of information is made up of eight bits of information. Bits (and their progressively bigger cousins, such as kilobits, megabits, and gigabits) are used to quantify data transmission speeds as an alternative and are more frequently employed in contemporary meanings than bygone generations. “Mbps” is one of the most often misunderstood abbreviations in all of the contemporary computing since it refers to “megabits,” not “megabytes,” per second, as the name implies.

## What is Byte?

A byte is an eight-bit representation of information, and it is the most frequently used word for referring to the quantity of information that may be kept in a computer’s memory. A computer system’s “eight bits” does not refer to “eight bits” in a broad, purely mathematical sense but rather to a collection of eight bits that function as a cohesive unit inside the computer system.

It was during the creation of the **IBM** Stretch computer that the byte was given its first official designation in 1956. A byte is a data unit that consists of eight bits of information. One byte may represent 28=256 different values, which is a very large number.

## Main Differences Between Bit and Byte

1. When it comes to computers, a bit is the smallest unit of data that can be represented, while a byte is eight bits.
2. A bit may be used to represent a maximum of two values at a time, whereas A byte may store up to 256 different values.
3. A bit is represented in lowercase b, whereas Byte is represented in uppercase B.
4. Bits are used to store just 1s and 0s in the computer’s memory, while bytes are used to store the whole alphabet plus any extra special characters.
5. A bit has different sizes such as s kilobit (kb megabit (Mb) gigabit (Gb) terabit (Tb) whereas Byte has kilobyte (kb) megabyte (MB)Gigabyte (GB) is terabyte (TB).

# NUMBER SYSTEMS

| Binary | Decimal | Octal | Hexadecimal |
|---|---|---|---|
| 0000 | 00 | 0 | 0 |
| 0001 | 01 | 1 | 1 |
| 0010 | 02 | 2 | 2 |
| 0011 | 03 | 3 | 3 |
| 0100 | 04 | 4 | 4 |
| 0101 | 05 | 5 | 5 |
| 0110 | 06 | 6 | 6 |
| 0111 | 07 | 7 | 7 |
| 1000 | 08 | 10 | 8 |
| 1001 | 09 | 11 | 9 |
| 1010 | 10 | 12 | A |
| 1011 | 11 | 13 | B |
| 1100 | 12 | 14 | C |
| 1101 | 13 | 15 | D |
| 1110 | 14 | 16 | E |
| 1111 | 15 | 17 | F |

## DECIMAL NUMBERS :

In the decimal number systems each of the ten digits, 0 through 9, represents a certain quantity. The position of each digit in a decimal number indicates the magnitude of the quantity represented and can be assigned a weight. The weights for whole numbers are positive powers of ten that increases from right to left, beginning with $10^o = 1$ that is $10^3$ $10^2$ $10^1$ $10^o$

For fractional numbers, the weights are negative powers of ten that decrease from left to right beginning with $10^{-1}$ that is $10^2\ 10^1\ 10^o.\ 10^{-1}\ 10^{-2}\ 10^{-3}$

The value of a decimal number is the sum of digits after each digit has been multiplied by its weights a sin following examples

**Express the decimal number 87 as a sum of the values of each.**

**digit.**

The digit 8 has a weight of10 which is 10 as indicated by its position. The digit 7 has a weight of 1 which is $10^o$ as indicated by its position.

$87 = (8 \times 10^1) + (7 \times 10^0)$

## BINARY NUMBERS

The binary system is less complicated than the decimal system because it has only two digits, it is a base two system. The two binary digits (bits) are 1 and 0. The position of a 1 or 0 in a binary number indicates its weight, or value within the number, just as the position of a decimal digit determines the value of that digit. The weights in a binary number are based on power of two as:

$$\ldots.\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0.\ 2^{-1}\ 2^{-2}\ldots.$$

With 4 digits position we can count from zero to 15.In general, with n bits we can count up to a number equal to Ḳ - 1. Largest decimal number = Ḳ - 1.A binary number is a weighted number. The right-most bit is the least significant bit (LSB) in a binary whole number and has a weight of $2^o$ =1. The weights increase from right to left by a power of two for each bit. The left-most bit is the most significant bit (MSB); its weight depends on the size of the binary number.

## BINARY – TO – DECIMAL CONVERSION

The decimal value of any binary number can be found by adding the weights of all bits that are 1 and discarding the weights of all bits that are 0

**Example**

Let‘s convert the binary whole number 101101 to decimal

**Binary no:** 1 0 1 1 0 1

**Weight:** $2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

**Value** → 32 0 8 4 0 1

**Sum** = 45

## HEXADECIMAL NUMBERS

The hexadecimal number system has sixteen digits and is used primarily as a compact way of displaying or writing binary numbers because it is very easy to convert between binary and hexadecimal.Long binary numbers are difficult to read and write because it is easy to drop or transpose a bit. Hexadecimal is widely used in computer and microprocessor applications. The hexadecimal system has a base of sixteen; it is composed of 16 digits and alphabetic characters. The maximum 3-digits hexadecimal number is FFF or decimal 4095 and maximum 4-digit hexadecimal number is FFFF or decimal 65.535.

## BINARY –TO – HEXADECIMAL CONVERSION

Simply break the binary number into 4-bit groups, starting at the right-most bit and replace each 4- bit group with the equivalent hexadecimal symbol as in the following

**example**

**Convert the binary number to hexadecimal:**

**1100101001010111**

**Solution:**

| **1100** | **1010** | **0101** | **0111** |
|---|---|---|---|
| **C** | **A** | **5** | **7** |

## HEXADECIMAL – TO – DECIMAL CONVERSION

One way to find the decimal equivalent of a hexadecimal number is to first convert the hexadecimal number to binary and then convert from binary to decimal. Convert the hexadecimal number 1C to decimal

1 C

01 1100 = $24+ 2^3 + 2^2 = 16 +8+4=28$

## OCTAL NUMBER

Like the hexadecimal system, the octal system provides a convenient way to express binary numbers and codes. However, it is used less frequently than hexadecimal in conjunction with computers and microprocessors to express binary quantities for input and output purposes.

The octal system is composed of eight digits, which are:0, 1, 2, 3, 4, 5, 6, 7

To count above 7, begin another column and start over: 10, 11, 12, 13, 14, 15, 16, 17, 20, 21 and so on. Counting in octal is similar to counting in decimal, except that the digits 8 and 9 are not used.

# Fixed and Floating Point :

## Fixed-Point Representation −

This representation has fixed number of bits for integer part and for fractional part. For example, if given fixed-point representation is IIII.FFFF, then you can store minimum value is 0000.0001 and maximum value is 9999.9999. There are three parts of a fixed-point number representation: the sign field, integer field, and fractional field.

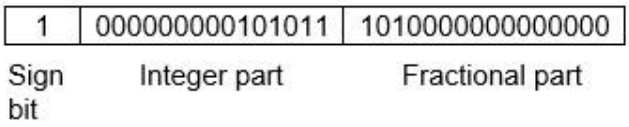

We can represent these numbers using:

- Signed representation: range from $-(2^{(k-1)}-1)$ to $(2^{(k-1)}-1)$, for k bits.
- 1's complement representation: range from $-(2^{(k-1)}-1)$ to $(2^{(k-1)}-1)$, for k bits.
- 2's complementation representation: range from $-(2^{(k-1)})$ to $(2^{(k-1)}-1)$, for k bits.

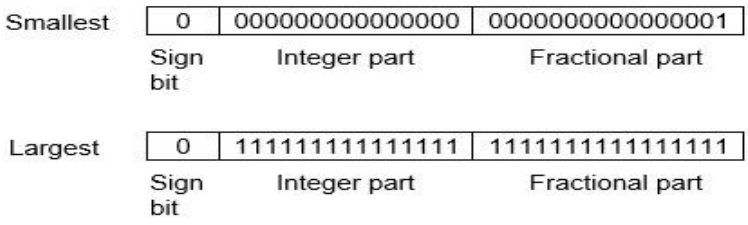
2's complementation representation is preferred in computer system because of unambiguous property and easier for arithmetic operations.

**Example −**Assume number is using 32-bit format which reserve 1 bit for the sign, 15 bits for the integer part and 16 bits for the fractional part.

Then, -43.625 is represented as following:




Where, 0 is used to represent + and 1 is used to represent. 000000000101011 is 15 bit binary value for decimal 43 and 1010000000000000 is 16 bit binary value for fractional 0.625.



These are above smallest positive number and largest positive number which can be store in 32-bit representation as given above format. Therefore, the smallest positive number is $2^{-16} \approx 0.000015$ approximate and the largest positive number is $(2^{15}-1)+(1-2^{-16})=2^{15}(1-2^{-16}) =32768$, and gap between these numbers is $2^{-16}$.

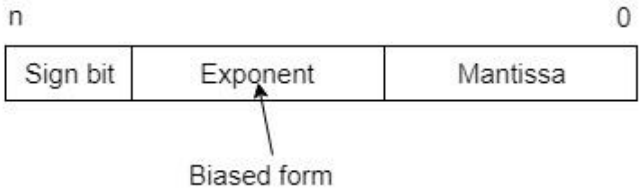
We can move the radix point either left or right with the help of only integer field is 1.

## Floating-Point Representation −

This representation does not reserve a specific number of bits for the integer part or the fractional part. Instead it reserves a certain number of bits for the number (called the mantissa or significand) and a certain number of bits to say where within that number the decimal place sits (called the exponent).

The floating number representation of a number has two part: the first part represents a signed fixed point number called mantissa. The second part of designates the position of the decimal (or binary) point and is called the exponent. The fixed point mantissa may be fraction or an integer. Floating -point is always interpreted to represent a number in the following form: $Mxr^e$.

Only the mantissa m and the exponent e are physically represented in the register (including their sign). A floating-point binary number is represented in a similar manner except that is uses base 2 for the exponent. A floating-point number is said to be normalized if the most significant digit of the mantissa is 1.



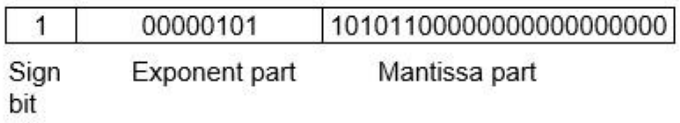

So, actual number is $(-1)^s(1+m)x2^{(e-Bias)}$, where *s* is the sign bit, *m* is the mantissa, *e* is the exponent value, and *Bias* is the bias number.

Note that signed integers and exponent are represented by either sign representation, or one's complement representation, or two's complement representation.

The floating point representation is more flexible. Any non-zero number can be represented in the normalized form of $\pm(1.b_1b_2b_3 ...)_2x2^n$ This is normalized form of a number x.

**Example −**Suppose number is using 32-bit format: the 1 bit sign bit, 8 bits for signed exponent, and 23 bits for the fractional part. The leading bit 1 is not stored (as it is always 1 for a normalized number) and is referred to as a *"hidden bit*".

Then −53.5 is normalized as $-53.5=(-110101.1)_2=(-1.101011)x2^5$ , which is represented as following below,



The precision of a floating-point format is the number of positions reserved for binary digits plus one (for the hidden bit). In the examples considered here the precision is 23+1=24.

The gap between 1 and the next normalized floating-point number is known as machine epsilon. the gap is $(1+2^{-23})-1=2^{-23}$for above example, but this is same as the smallest positive floating-point number because of non-uniform spacing unlike in the fixed-point scenario.

Note that non-terminating binary numbers can be represented in floating point representation, e.g., 1/3 = $(0.010101\ ...)_2$ cannot be a floating-point number as its binary representation is non-terminating.

# ASCII Code →

**ASCII**, stands for American Standard Code for Information Interchange. It's a 7-bit character code where every single bit represents a unique character. On this webpage you will find 8 bits, 256 characters, ASCII table according to Windows-1252 (code page 1252) which is a superset of ISO 8859-1 in terms of printable characters. In the range 128 to 159 (hex 80 to 9F), ISO/IEC 8859-1 has invisible control characters, while Windows-1252 has writable characters. Windows-1252 is probably the most-used 8-bit character encoding in the world.

| DEC | ASCII | DEC | ASCII | DEC | ASCII | DEC | ASCII | DEC | ASCII | DEC | ASCII | DEC | ASCII | DEC | ASCII |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ☺ | 32 | space | 64 | @ | 96 | ` | 128 | Ç | 160 | á | 192 | └ | 224 | Ó |
| 2 | ☻ | 33 | ! | 65 | A | 97 | a | 129 | ü | 161 | í | 193 | ┴ | 225 | ß |
| 3 | ♥ | 34 | " | 66 | B | 98 | b | 130 | è | 162 | ó | 194 | ┬ | 226 | Ô |
| 4 | ♦ | 35 | # | 67 | C | 99 | c | 131 | â | 163 | ú | 195 | ├ | 227 | Ò |
| 5 | ♣ | 36 | $ | 68 | D | 100 | d | 132 | ä | 164 | ñ | 196 | ─ | 228 | õ |
| 6 | ♠ | 37 | % | 69 | E | 101 | e | 133 | à | 165 | Ñ | 197 | ┼ | 229 | Õ |
| 7 | • | 38 | & | 70 | F | 102 | f | 134 | å | 166 | ª | 198 | ã | 230 | µ |
| 8 | ◘ | 39 | ' | 71 | G | 103 | g | 135 | ç | 167 | º | 199 | Ã | 231 | þ |
| 9 | ○ | 40 | ( | 72 | H | 104 | h | 136 | ê | 168 | ¿ | 200 | ╚ | 232 | Þ |
| 10 | ◙ | 41 | ) | 73 | I | 105 | i | 137 | ë | 169 | ® | 201 | ╔ | 233 | Ú |
| 11 | ♂ | 42 | * | 74 | J | 106 | j | 138 | è | 170 | ¬ | 202 | ╩ | 234 | Û |
| 12 | ♀ | 43 | + | 75 | K | 107 | k | 139 | ï | 171 | ½ | 203 | ╦ | 235 | Ù |
| 13 | ♪ | 44 | , | 76 | L | 108 | l | 140 | î | 172 | ¼ | 204 | ╠ | 236 | ý |
| 14 | ♫ | 45 | - | 77 | M | 109 | m | 141 | ì | 173 | ¡ | 205 | ═ | 237 | Ý |
| 15 | ☼ | 46 | . | 78 | N | 110 | n | 142 | Ä | 174 | « | 206 | ╬ | 238 | ¯ |
| 16 | ► | 47 | / | 79 | O | 111 | o | 143 | Å | 175 | » | 207 | ¤ | 239 | ´ |
| 17 | ◄ | 48 | 0 | 80 | P | 112 | p | 144 | È | 176 | ░ | 208 | ð | 240 | - |
| 18 | ↕ | 49 | 1 | 81 | Q | 113 | q | 145 | æ | 177 | ▒ | 209 | Ð | 241 | ± |
| 19 | ‼ | 50 | 2 | 82 | R | 114 | r | 146 | Æ | 178 | ▓ | 210 | Ê | 242 | ‗ |
| 20 | ¶ | 51 | 3 | 83 | S | 115 | s | 147 | ô | 179 | │ | 211 | Ë | 243 | ¾ |
| 21 | § | 52 | 4 | 84 | T | 116 | t | 148 | ö | 180 | ┤ | 212 | È | 244 | ¶ |
| 22 | ▬ | 53 | 5 | 85 | U | 117 | u | 149 | ò | 181 | Á | 213 | ı | 245 | § |
| 23 | ↨ | 54 | 6 | 86 | V | 118 | v | 150 | û | 182 | Â | 214 | Í | 246 | ÷ |
| 24 | ↑ | 55 | 7 | 87 | W | 119 | t | 151 | ù | 183 | À | 215 | Î | 247 | ¸ |
| 25 | ↓ | 56 | 8 | 88 | X | 120 | x | 152 | ÿ | 184 | © | 216 | Ï | 248 | ° |
| 26 | → | 57 | 9 | 89 | Y | 121 | y | 153 | Ö | 185 | ╣ | 217 | ┘ | 249 | ¨ |
| 27 | ← | 58 | : | 90 | Z | 122 | z | 154 | Ü | 186 | ║ | 218 | ┌ | 250 | · |
| 28 | ∟ | 59 | ; | 91 | [ | 123 | { | 155 | ø | 187 | ╗ | 219 | █ | 251 | ¹ |
| 29 | ↔ | 60 | < | 92 | \ | 124 | \| | 156 | £ | 188 | ╝ | 220 | ▄ | 252 | ³ |
| 30 | ▲ | 61 | = | 93 | ] | 125 | } | 157 | Ø | 189 | ¢ | 221 | ¦ | 253 | ² |
| 31 | ▼ | 62 | > | 94 | ^ | 126 | ~ | 158 | × | 190 | ¥ | 222 | Ì | 254 | ■ |
|  |  | 63 | ? | 95 | _ | 127 | ⌂ | 159 | ƒ | 191 | ┐ | 223 | ▀ | 255 | space |

# EBCDIC Code

Extended Binary Coded Decimal Interchange (EBCDIC) code is 8 bits code which means $2^8$ = 256 ways of representing digits, alphabets and special symbol.

Parity bit is also used under the system. EBCDIC is an 8 bit code, it can be divided into two 4 bit group. Each of these 4 bit group represented by hexadecimal number system is used as sort out notation for memory dump by computer, which is used EBCDIC for internal representation of character. This result in a one to four year reduction in the volume of memory dump.

## Computer Software

**DEFINITION-**

**Software is a set of programs, which is designed to perform a well defined function. A program is a sequence of instruction written to solve a particular problem.**

**1. Software (S/W):** - Software is the part of the computer system which enables the Hardware to operate. The term Software means collections of programs. A program is a set of logical instruction that is required to accomplish a particular task. The instruction may be given using any of the many computer language. (It is sets of instruction stored as programs that govern the operation of a computer system and it makes Hardware to do task.). It is 2 types:-

**(A) System Software:** - A system is a set of program that manages the resource of a computer system so that they are used in an optimal fashion, and provide routine service. It include the computer programs that run a computer system itself or that assist a computer in running application programs like operating system.

**System software** is the software used to manage and control the hardware components and which allow interaction between the hardware and the other types of software. The most obvious type of system software is the computer's operating system but device drivers are also included within this category.

**System software**-The system software is collection of programs designed to operate, control and extend the processing capabilities of the computer itself. System software is generally prepared by computer manufactures.

**Features of System Software :**

1. It is Close to system.
2. It is fast in speed.

3. It is Difficult to manipulate.

4. It is smaller in size.

5. It is Difficult to design.

6. It is Difficult to understand.

7. It is generally written in low level language.

## Types of System Software:

1. System control program

2. System support program

3. System development programs

**System control program-they control the execution of programs**. E.g. Operating system Device drivers-Device drivers are system programs, which are responsible for proper functioning of device. E.g. device like printer, a user must load the device driver of that particular printer. System support programs-They provide routine service function to other computer programs and users. E.g. utility programs.

**Some utility programs are-Text** editors, they are used to create and edit files. For exp-notepad is the text editor.

**Backup utilities**-These utility programs help us to backup of our important data. By using these programs files are backed up to floppies, CD and DVD.

**Data recovery software**-Sometimes an illegal operation may result in an accidental loss of data which was still to be needed then we used data recovery software. E.g. Recycle bin.

**Compression utilities**- Compression utility are used to compress large sized files so that they can be stored in storage of low capacity. Win-zip is a popular compression utility used in window based desktops.

**Anti virus utility-** Any program that affects the normal working of the other programs or affects the boot sector of the disk is a virus. The anti virus software detects the virus, identify and prevent it from spreading. Some examples of anti-virus software are Norton's anti virus, MCafee etc.

**System Development Programs-** They assists in the creation of computer programs. Examples of system development are – programming language, language translations.

(**B) Application Software:** - Application Software is the programs which on the other hand performs specific tasks for computer user. An appropriate program is designed to handle a particular task required by the end-user (Like, if an appropriate Software is designed to design images will only work for image designing.) e.g. Are MS-Excel, MS-Word, Photoshop etc.

**Applications software** (also known as 'apps') are designed to allow the user of the system complete a specific task or set of tasks. They include programs such as web browsers, office software, games and so on. They are usually the reason you bought the computer system in the first place and aren't concerned with the management or maintenance of the system itself. Any individual software package, whichever of the above types it falls into, can be either generic (or 'off-the-shelf') or it can be bespoke (custom-built). Generic software is mass produced with the intention that it will be used by a wide variety of different users in a range of different situations. Bespoke software is created for a specific purpose which will be used in a known environment.

**Application software-**Application software is the software that is designed to satisfy a particular need of a particular environment. All software prepared by us in the computer lab. Examples of application software are-student record software, railway reservation software, income tax software, word processors etc.

## Features of application software:-

1. It is close to user.

2. It is Slow in speed

3. It is Easy to understand.

4. It is Easy to manipulate

5. It is generally, written in high level language

6. It is easy to design.

## TYPES OF APPLICATION SOFTWARE

**General purpose application software**-General purpose application software are designed to satisfy common needs of various business. SOME GENERAL PURPOSE SOFTWARE ARE WORD PROCESSOR-word processor is the software used to word processing. There are many word processors available in the market. The common and the popular among are: Word Star, MS-WORD.

**ELECTRONIC SPREADSHEET**:-A spreadsheet contains greed of cells arranged in columns and rows. Data is entered into the cells to represent information. Examples of electronics spreadsheet are lotus 1, 2, 3 and excel.

**PRESENTATION SOFTWARE**-presentation software are the software which are used to present information to a large number of people. Microsoft PowerPoint is one of the most popular presentation software.

**DESKTOP PUBLISHING SOFTWARE**- desktop publishing software is used for type setting and designing purposes. Well know desktop publishing software is page marker and Coral Draw.

**WEB BROWSER SOFTWARE**-with an internet connection, this type of software enables a user to visit from one site to another by following, to search locations and view web documents .Examples are Netscape communicator, Microsoft internet explorer 6.

**SPECIAL PURPOSE APPLICATION SOFTWARE**-application software is created to satisfy specific needs of an organization. Example are payroll software, railway reservation software etc.

**Utility software** is software such as anti-virus software, firewalls, disk defragmenters and so on which helps to maintain and protect the computer system but does not directly interface with the hardware.

## Different between system Software and application Software

| | System Software | Application Software |
|---|---|---|
| 1 | System S/W include the computer program that run a computer system it self or that assists the computer in running application programs. | Application S/W is a program that runs over the operating system. |
| 2 | System S/W directly interacts with H/W. | Application S/W interacts with H/W through system S/W. |
| 3 | System S/W are common for various users. | Application S/W varies according to the user requirements. |
| 4 | System S/W helps the user to interact with computer. | Application S/W helps the user to solve their problems |
| 5 | eg :- O.S. , compiler, interpreter etc. | e.g.:- MS-Word, MS-EXCEL, TALLY, AUTOCAD, PHOTOSHOP etc |

# Computer Programming Languages :

## Programming languages :

A programming language is any set of rules that converts strings, or graphical program elements in the case of visual programming languages, to various kinds of machine code output. Programming languages are one kind of computer language, and are used in computer programming to implement algorithms.

## Classification of programming languages:

1. Machine Languages

2. Assembler Languages

3. High Level Languages

## Machine Languages

In computer programming, machine code is any low-level programming language, consisting of machine language instructions, which are used to control a computer's central processing unit (CPU). Each instruction causes the CPU to perform a very specific task, such as a load, a store, a jump, or an arithmetic logic unit (ALU) operation on one or more units of data in the CPU's registers or memory.

### Advantages of Machine Language

i) It makes fast and efficient use of the computer.

ii) It requires no translator to translate the code i.e. Directly understood by the computer

### Disadvantages of Machine Language:

i) All operation codes have to be remembered

ii) All memory addresses have to be remembered.

iii) It is hard to amend or find errors in a program written In the machine language

iv) These languages are machine dependent i.e. a particular Machine language can be used on only

one type of compute

### Assembly Languages

In computer programming, assembly language (or assembler language), sometimes abbreviated asm, is any low-level programming language in which there is a very strong correspondence between the instructions in the language and the architecture's machine code instructions. Assembly language usually has one statement per machine instruction (1:1), but constants, comments, assembler directives, symbolic labels of, e.g., memory locations, registers, and macros are generally also supported.

## Advantages of Assembly Language

i) It is easier to understand and use as compared to machine language.

ii) It is easy to locate and correct errors.

iii) It is modified easily.

### Disadvantages of Assembly Language

i) Like machine language it is also machine dependent.

ii) Since it is machine dependent therefore programmer should have the knowledge of the

hardware also.

## High-level Programming Language

In computer science, a high-level programming language is a programming language with strong abstraction from the details of the computer. In contrast to low-level programming languages, it may use natural language elements, be easier to use, or may automate (or even hide entirely) significant areas of computing systems (e.g. memory management), making the process of developing a program simpler and more understandable than when using a lower-level language. The amount of abstraction provided defines how "high-level" a programming language is.

### Advantages of High Level Language

Following are the advantages of a high level language:

- User-friendly
- Similar to English with vocabulary of words and symbols
- Therefore it is easier to learn.
- They require less time to write.
- They are easier to maintain.
- Problem oriented rather than 'machine' based.

- Program written in a high-level language can be translated into many machine language and therefore can run on any computer for which there exists an appropriate translator.
- It is independent of the machine on which it is used i.e. Programs developed in high level language can be run on any Computer.

**Disadvantages of High Level Language**

- A high-level language has to be translated into the machine language by a translator and thus a price in computer time is paid.
- The object code generated by a translator might be inefficient Compared to an assembly language program.

## Complier :

In computing, a compiler is a computer program that translates computer code written in one programming language (the source language) into another language (the target language). The name "compiler" is primarily used for programs that translate source code from a high-level programming language to a lower level language (e.g. assembly language, object code, or machine code) to create an executable program.