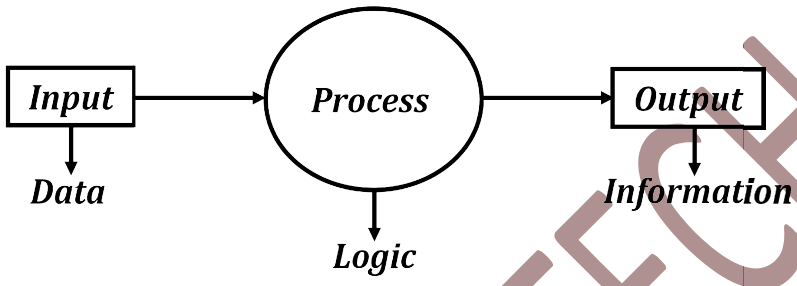
Computer

Computer is an electronic data processing machine which takes some input, process it and give us output as a result (information).



Block Diagram of Computer

Data

1. *Data is raw fact and figure which will be processed.*
2. *In case of computer data will in any form such as* ***Text, Audio, Video, Graphics, Number, etc.***

Information

When data is processed then it is convert into a meaningful terms known as information.

Processed Data = Information

Process

1. *Process is those which are done on input.*
2. *Process is a way in which through we describe how to achieve output.*
3. *Processing one use logic for converting input or data into in meaningful term.*

Logic

Logic is the statement or expression which is evaluated as true or false.

Instruction

Instructions are combination of operators, operands and statements.

Language Development Process

Language development process divides by two parts

- *Low level language*
- *High level language*

Low level language

The languages in this category are the machine level language and assembly language.

Machine level language

1. *Computers can understand only digital signals, which are in binary digits i.e. 0 and 1.*
2. *So the instructions given to the computer can be only in binary codes.*
3. *The machine language consists of instructions that are in binary 0 or 1.*
4. *Computers can understand only machine level language.*
5. *Writing a program in machine level language is a difficult task because it is not easy for programmers to write instructions in binary code.*
6. *A machine level language is error – phone and its maintenance are very difficult.*
7. *Further more machine language programs are not portable.*
8. *Every computer has its own machine instructions, so the programmers written for one computer are not valid for other computer.*

Advantage of machine – level language

1. *Machine level language makes efficient use of computer system resource like storage and register.*
2. *The instructions of a machine – level language program are directly executable so there is no need of translators.*
3. *Machine – level language instructions can be used to manipulate the individual bits in a computer system with a very high execution speed due to direct manipulation of memory and registers.*

Assembly language

1. *The difficult faced in machine level language were reduced to some extent by using a modified form of machine level language called assembly language.*
2. *In assembly language instructions are given in English like words, such as MOV, ADD, SUM, etc.*
3. *So it is easier to write and understand assembly programs.*
4. *Since a computer can understand only machine level language, hence assembly language program must be translated into machine level language.*
5. *The translator that is used for translating is called **"assembler"**.*
6. *Although writing programs in assembly language is a bit easier, but still the programmer has to know all the low level details related with the hardware of a computer.*
7. *In assembly language, data is stored in computer registers and each computer has different set of registers.*
8. *Hence the assembly language program is also not portable.*
9. *Since the low – level languages are related with the hardware, hence the execution of a low – level program is faster.*

Advantage of assembly language

1. *Easier to understand and use.*
2. *Easier to locate and correct errors.*
3. *Easier to modify.*
4. *No need to keep track of the addresses.*
5. *Easily re-allocatable.*
6. *Efficiency of machine – level language.*

Limitation of assembly language

1. *Machine dependence.*
2. *Knowledge of hardware required.*
3. *Machine level coding.*

High level language

1. *High – level language are designed keeping in mind the features of portability i.e. these languages are machine independent.*
2. *These are English like languages, so it is easy to write and understand the programs of high – level language.*
3. *While programming in a high – level language, the programmer is not concerned with the low – level language details, and so that whole attention can be paid to logic of the problem being solved.*
4. *For translating a high – level program into machine level language, compiler or interpreter is used.*
5. *Every language has its own compiler or interpreter.*
6. *Some languages in this category are – FORTRAN, COBOL, BASIC, Pascal, etc.*

Advantage of high – level languages

1. ***Machine independence***

   A program written in a high – level language can be executed on many different types of computers with very little or practically no effort of porting it on different computers.

2. ***Easier to learn and use***

   High – level languages are easier to learn because they are very similar to the natural languages used by us our day to day lives.

3. ***Fewer Errors***

   While programming in a high – level languages, a programmer need not worry about how and where to store the instructions and data of the program, and need not write machine – level instructions for the steps to be carried out by the computer. The programmer can concentrate more on the logic of the program which leads to fewer errors.

4. ***Better documentation***

   As the statement written programs in high – level languages are very similar to natural language, hence few or practically no separate comment statements are required in programs written in high – level languages.

5. ***Lower program preparation cost***

   Writing programs in high – level languages requires less time and effort, which ultimately leads to lower program preparation costs.

6. ***Easier to maintain***

   Program are easier to maintain because they are easier to understand and hence it is easier to locate, correct and modify instructions as when desires.

Translator

1. *Program which convert a source code in any other language mostly "machine code" are called translator programs.*
2. *Every high – level language and assembly uses a translator to translate its programs.*
3. *"A translator is a program that takes as input a program written in one programming language known as source language and produces as output a program in another language (machine – level language) knows target language".*

Compiler

1. *Compiler is programs which translate the high – level language programs into machine level instructions.*
2. *Compiler goes through a series of steps to examine and modify source code during the compilation process.*
3. *Each source code instruction typically gives a list to several machine level code instructions.*
4. *"Compiler is a program which takes as input a program of a high – level language and produces object program in machine code".*

Compilation Process

Interpreter

1. *An interpreter is the language translator for the third generation programming languages.*
2. *Rather than generating object code for the entire source code, this class of translators examines and executes source code on the line by line basis.*
3. *"Interpreter is smaller than a complex and facilities the implementation of complex programming language constructs by translating, interpreting and executing each line one by one".*

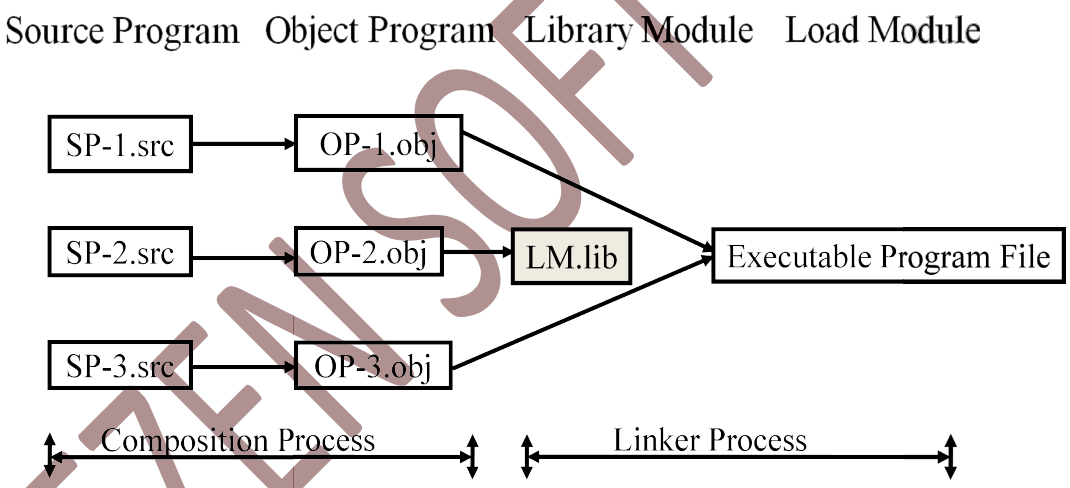
Interpreter	Compiler
It takes one statement of a high – level language program and translates it into a machine – a level instruction which is immediately executed.	*It is called so because it translate each high – level language instruction into a set of machine – level language instructions rather than a single machine – level language instruction.*
Translation and execution are alternate for each statement encountered in the high – level language program.	*This merely translates the entire source program into an object program and is not involved in its execution.*
The object code is not saved for future use because the translation and the execution processes alternate.	*The whole source program is translated into its equivalent machine – level language program. The object code obtained is permanently saved for future use and is used every time the program is to be executed.*
As instruction is interpreted and translated into machine – level language every time changes are incorporated in the program.	*Repeated compilation is not necessary for repeated execution of a program.*
Interpreters are slower than compilers.	*A compiled program runs much faster than an interpreted program.*

Assembler

An assembler takes a program written in assembly language as input (known as source program) and generates its equivalent machine language code (known as object program) as output.

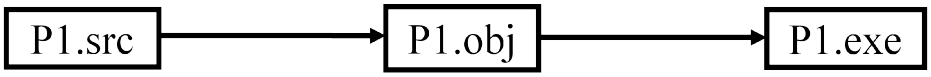
Linker

1. *It is not practically possible to store a large number of lines a single source program file.*
2. *For developing bulky software's, a modular approach is generally followed.*
3. *Which results in a number of source program files.*
4. *Each source program file can be handled independently of other source program files to create a corresponding object program file.*
5. *In this case, a program called LINKER is used to combine all the object program files of the software, and to convert them into the final executable program, which is referred to as a local module.*



Loader

Some systems use software called LODER to bypass the step of creating a loadable module and place the object module directly into memory for execution.



There are two types of error:

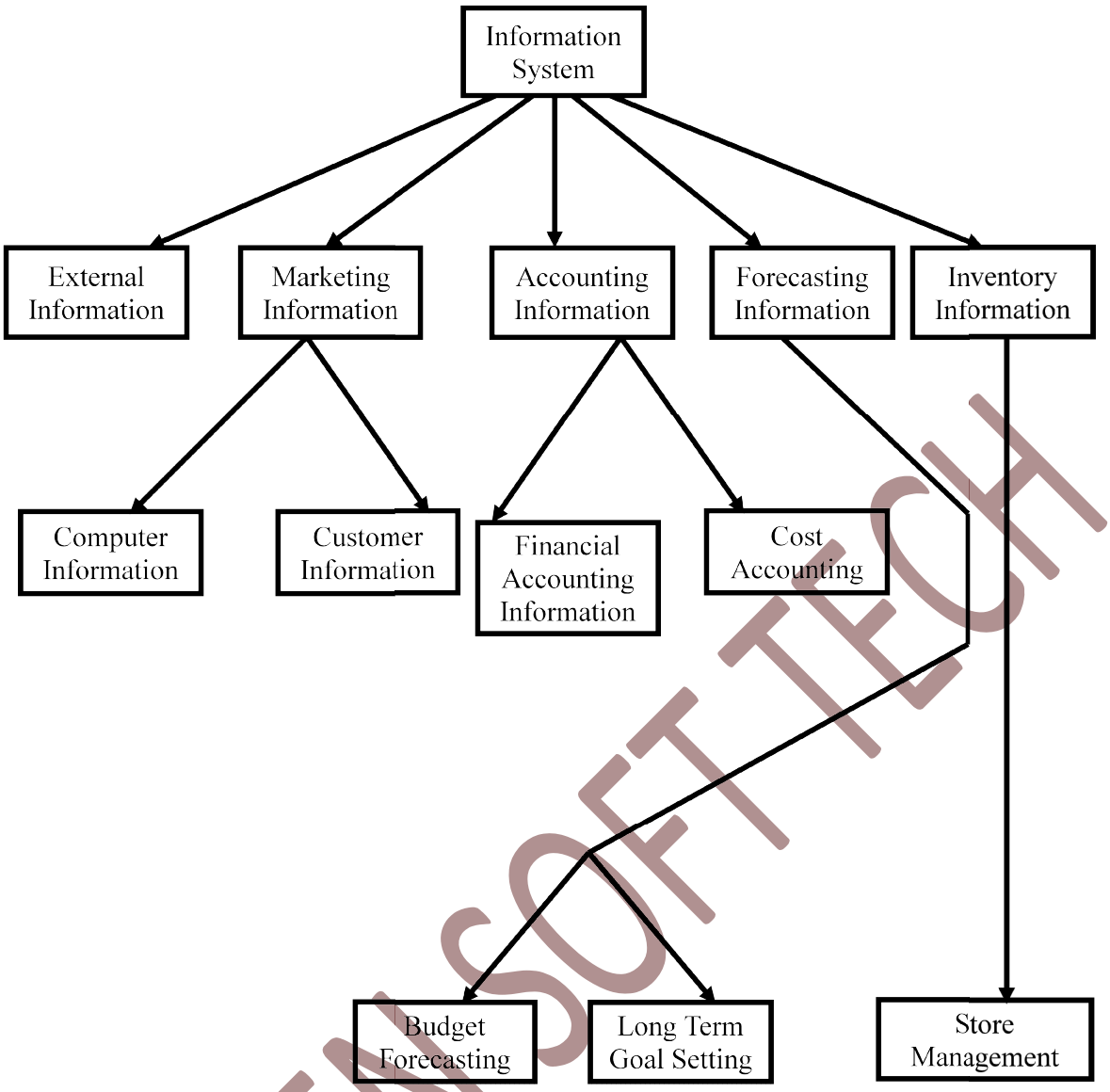
1. *Run time error*
2. *Logical error*

There are two types of *approach*:

1. *Top down approach*
2. *Bottom up approach*

Top down approach

1. *Most of the high level languages are producer oriented language.*
2. *A producer is a set of instructions which can be called anywhere in the program.*
3. *We can break a complex program into several less complex problems which can be further divided into smaller problems and so on.*
4. *Small and less complex tasks can be easily be handled in terms of making producers for them.*
5. *Which are collectively used to solve the main problem.*



Information System Hierarchy of an Organization

Software

Software is the set of programs that govern the operation of a computer system and make the hardware run.

Types of Software

- *Application Software*
- *System Software*

Application Software

Application software uses the computer system to perform useful work or provide entertainment functions beyond the basic operation of the computer itself.

Types of Application Software

- Standard Application Software
- Customized Application Software

System Software

System software is designed to operate the computer hardware, to provide basic functionality and to provide a platform for running application software. System software includes are operating system, device driver and utilities software.

Standard Application Software

This type of software is developed keeping in mind the standard requirements for carrying out a specific task.

Customized Application Software

This type of software is tailor – made software according to a user's requirements. The software is developed to meet all the requirements specified by the user.

This type of software cannot be directly installed at any other users workplace as the requirement of this user may differ from the first one and the software may not fit in the requirements of new users.

Operating System

1. *An operating system is a program which acts as an interface between a user and hardware i.e. all computer resources.*

2. *An operating system is an essential collection of computer programs that manages resources and provides common service for other software.*

3. *An operating system comes bundled with additional software (including application software) so that a user can potentially do some work with the computer that only has an operating system.*

4. *In other words, an operating system is the computer program that manages all other programs on the computer.*

5. *Supervisory programs, boot loaders, shells and window systems are parts of operating system.*

Device Driver

1. *Device driver is computer program that operates or controls a particular type of device that is attached to a computer.*
2. *Each device needs at least one corresponding device driver.*

Utilities Software

Utilities software designed to assist users in maintenance and care of their computers.

Object – Oriented Language:

The language should support several of the OOP concepts to claim that they are object – oriented depending upon the features they support.

They can be classified into the following two categories.

- ❖Object – Based Programming
- ❖Object – Oriented Programming

Object – Based Programming:

Object based programming is the style of programming that primarily supports encapsulation and object identity. Major feature that are required for object – based programming are–

- ★Data Encapsulation.
- ★Data Hiding and Access Mechanism.
- ★Automatic Initialization and Clear – Up of Objects.
- ★Operator Overloading

Languages that support programming with objects are said to be object – based programming languages. They do not support inheritance and dynamic binding. ***Ada*** *is a typical object – based programming language.*