Unit -6 COMPUTER SOFTWARE

Prashant Gautam

M.Sc. CSIT

Software

- Software can be broadly classified in two categories:
  1. System Software, and
  2. Application Software

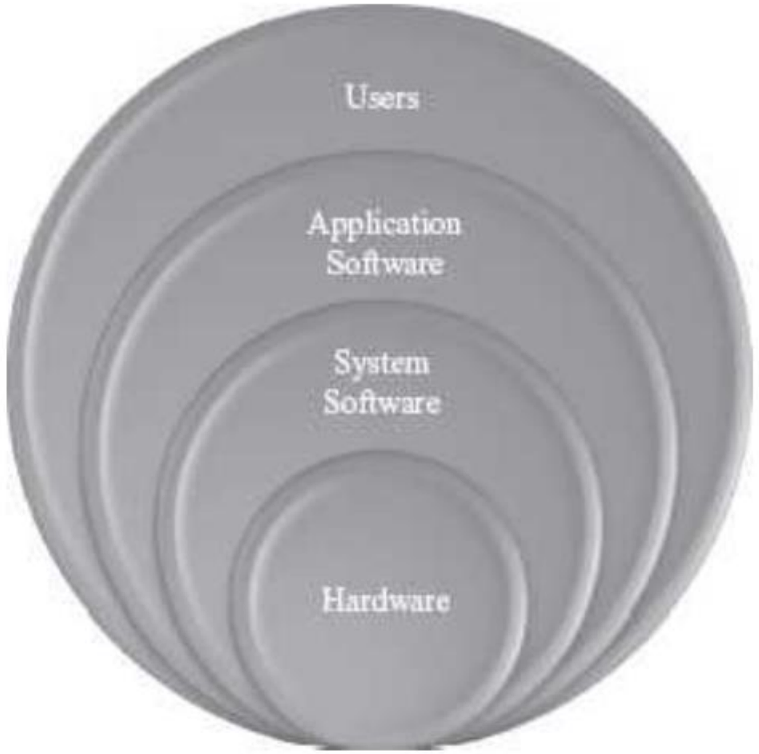


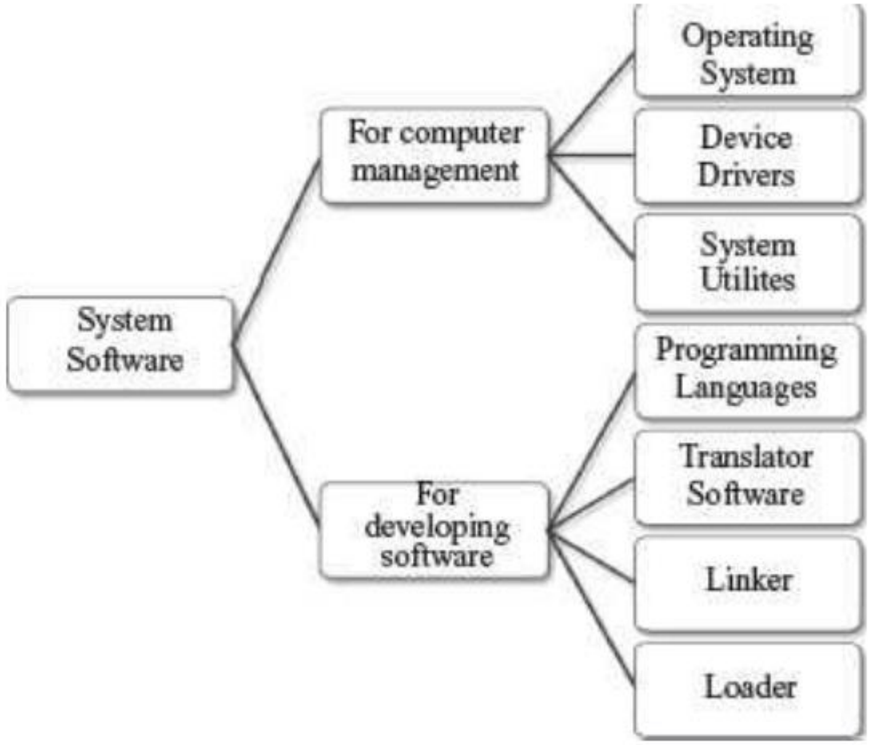
Figure 6.2 Software hierarchy

System Software

- System software provides basic functionality to the computer.
- System software is required for the working of computer itself.
- The user of computer does not need to be aware about the functioning of system software, while using the computer.
- For example, when you buy a computer, the system software would also include different device drivers.
- When you request for using any of the devices, the corresponding device driver software interacts with the hardware device to perform the specified request. If the appropriate device driver for any device, say a particular model of a printer, is installed on the computer, the user does not need to know about the device driver, while printing on this printer.

- The purposes of the system software are:
- To provide basic functionality to computer,
- To control computer hardware, and
- To act as an interface between user, application software and computer hardware.

System software



Operating System

- Operating System (OS) is an important part of a computer.
- OS intermediates between the user of a computer and the computer hardware.
- Different kinds of application software use specific hardware resources of a computer like CPU, I/O devices and memory, as needed by the application software.
- OS controls and coordinates the use of hardware among the different application software and the users.
- It provides an interface that is convenient for the user to use, and facilitates efficient operations of the computer system resources.

Functions of OS

- It provides an environment in which users and application software can do work.
- It manages different resources of the computer like the CPU time, memory space, file storage, I/O devices etc. During the use of computer by other programs or users, operating system manages various resources and allocates them whenever required, efficiently.
- It controls the execution of different programs to prevent occurrence of error.
- It provides a convenient interface to the user in the form of commands and graphical interface, which facilitates the use of computer.

What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

Operating System Definition

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

Operating System Definition (Cont.)

- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is a good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the **kernel**.
- Everything else is either
  - a system program (ships with the operating system) , or
  - an application program.

Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a ***passive entity***, process is an ***active entity***.
- Process needs resources to accomplish its task
    - CPU, memory, I/O, files
    - Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one **program counter** specifying location of next instruction to execute
    - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
    - Concurrency by multiplexing the CPUs among the processes / threads

Process Management Activities

The operating system is responsible for the following activities in connection with process management:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

Memory Management

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory.
- Memory management determines what is in memory and when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Storage Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - **file**
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and directories
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

Storage Management

- The OS manages storage through one of its **sub-modules**, the **File Manager**
- A **file system** is a collection of directories, subdirectories, and files organized in a **logical order**
- File manager **maintains an index** of the filenames & where they are located on the disk
- **File manager make it easy to find** the required file in a **logical and timely fashion**

Device Management

- controlling the Input/Output devices like disk, microphone, keyboard, printer, magnetic tape, USB ports, camcorder, scanner, other accessories, and supporting units like supporting units control channels.
- OS handles communication with the devices via their drivers. Functions:
  - It keeps track of data, status, location, uses, etc.
  - It enforces the pre-determined policies and decides which process receives the device when and for how long.
  - It improves the performance of specific devices.
  - It monitors the status of every device, including printers, storage drivers, and other devices.
  - It allocates and effectively deallocates the device.

Protection and Security

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
  - User identities (**user IDs**, security IDs) include name and associated number, one per user
  - User ID then associated with all files, processes of that user to determine access control
  - Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file
  - **Privilege escalation** allows user to change to effective ID with more rights

Types of OS'es

- Classification w.r.t. the **type of computers** they run on and the type of **applications** they support
  - **Real-Time** Operating System (RTOS)
  - **Single-User, Single Task**
  - **Single-User, Multi-Tasking**
  - **Multi-User**

RTOS (1)

- Used to run **computers embedded** in machinery, robots, scientific instruments and industrial systems
- Typically, it has **little user interaction capability**, and no end-user utilities, since the system will be a "**sealed box**" when delivered for use
- **Examples:** Wind River, QNX, Real-time Linux, Real-time Windows NT

RTOS (2)

- An important part of an RTOS is **managing the resources** of the computer so that a particular operation **executes in precisely the same amount of time** every time it occurs

- In a complex machine, **having a part move more quickly** just **because system resources are available** may be just as **CATASTROPHIC** as having it not move at all because the system was busy

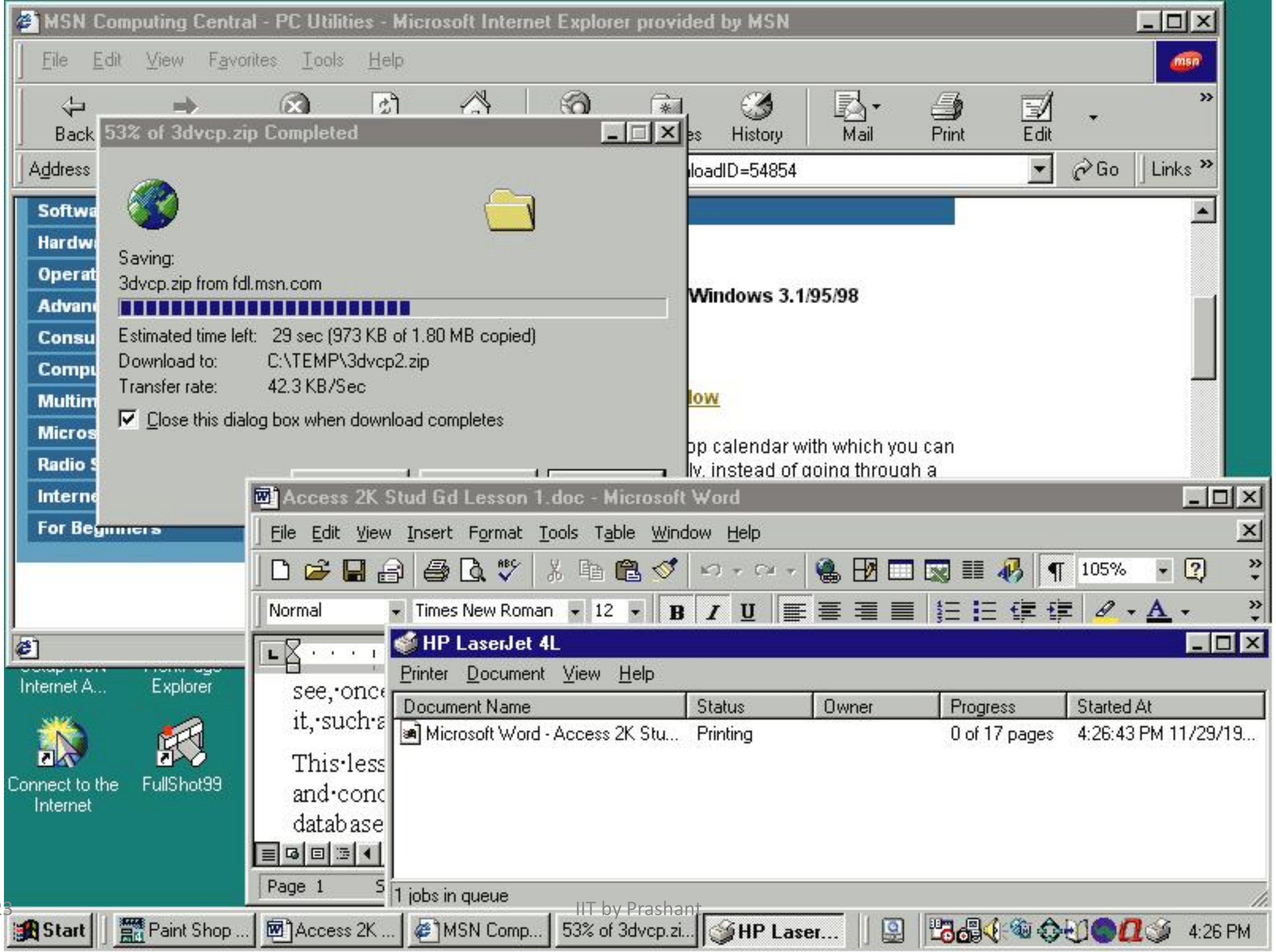
Single-User, Single Task

- OS’es designed to manage the computer so that **one user can effectively do one thing at a time**

- The **Palm OS** used in many palmtop computers (PDA’s) is an example of a single-user, single-task OS

Single User, Multi-Tasking

- Most **popular** OS
- Used by **most all PC's** and Laptops
- **Examples:** Windows, Mac OS, Linux
- Lets a single user **interact with several programs**, simultaneously



53% of 3dvcp.zip Completed

Saving:
3dvcp.zip from fdl.msn.com

Estimated time left: 29 sec (973 KB of 1.80 MB copied)
Download to: C:\TEMP\3dvcp2.zip
Transfer rate: 42.3 KB/Sec

☑ Close this dialog box when download completes

HP LaserJet 4L

Printer Document View Help

Document Name	Status	Owner	Progress	Started At
Microsoft Word - Access 2K Stu...	Printing		0 of 17 pages	4:26:43 PM 11/29/19...

1 jobs in queue

Multi-User

- A multi-user OS allows many users to take advantage of the **computer's resources, simultaneously**

- The OS must make sure that the **requirements of the various users are balanced**, and that the programs they are using each have sufficient and separate resources so that a **problem with one user doesn't affect any of the other users**

- **Examples:** Linux, Unix, VMS and mainframe OS'es, such as MVS

Another Way of Classifying

Uni-processor OS'es

Designed to schedule tasks on a single uP only

Example: DOS

Multi-processor OS'es

Can control computers having multiple uPs, at times 1000's of them

Example: Current versions of Windows, Mac OS, Linux, Solaris

How many different OS'es are there?

- 100's
- OS'es from the **Windows family** dominate the desktops and run on millions of PC's
- OS'es from the **Unix family** (Unix, Linux, etc) are quite popular on servers
- There are **hundreds more**. Some designed for **mainframes** only. Some for **embedded** applications only.

Comparing Popular OS'es

OS	HW	Stability	Cost	Apps.	Support	Security	Popularity
Windows (GUI)	PC	Poor	$300	Huge no.	OK	Poor	Amazing
Mac OS (Shell/GUI)	Mac	Good	$60	Many	OK	Good	Low
Linux (Shell/GUI)	Many	Good	Low	Many	Variable	Good	Low
Unix (Shell/GUI)	Many	Excellent	High	Many	Expensive	Excellent	Servers

Device Driver

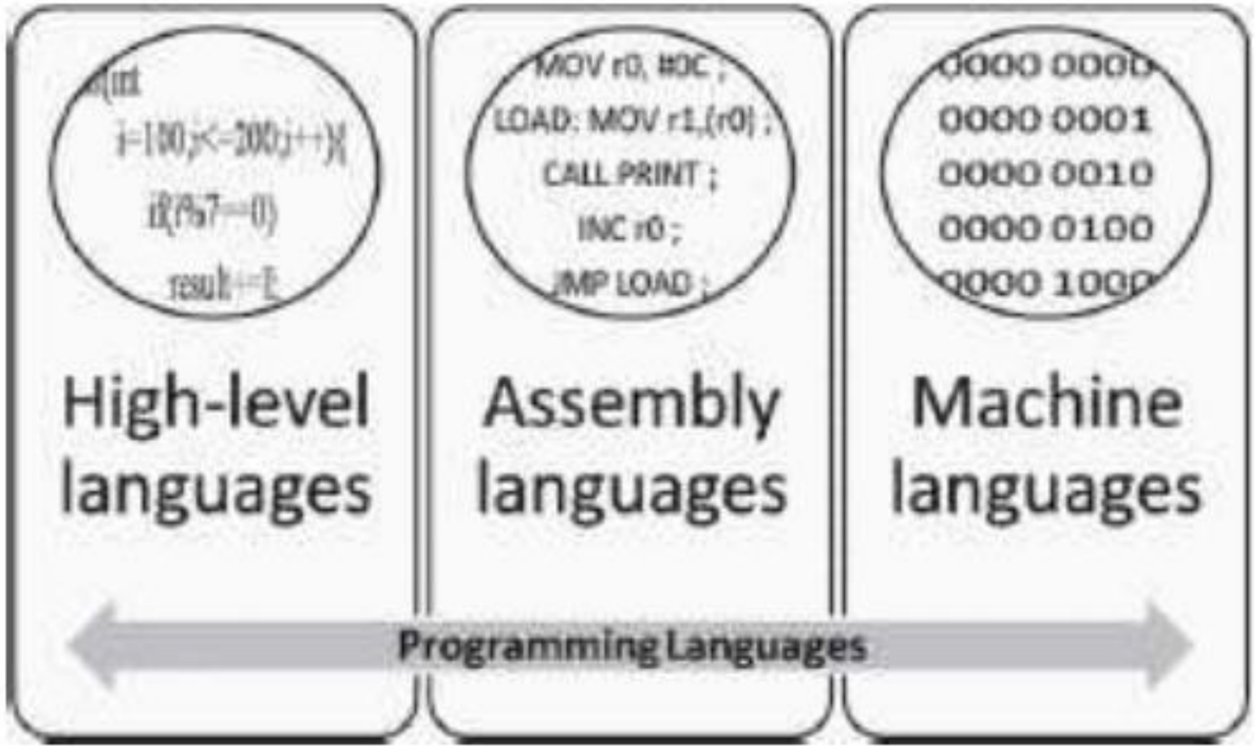
- A device driver acts as a translator between the hardware and the software that uses the devices.
- In other words, it intermediates between the device and the software, in order to use the device.

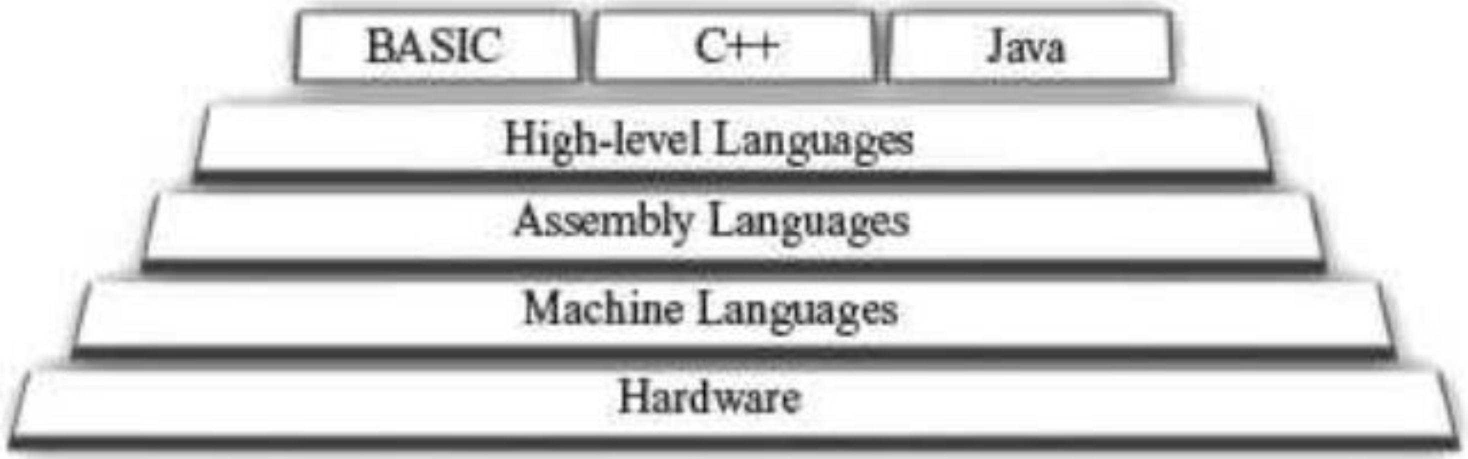
System Utilities

- System utility software is required for the maintenance of computer. System utilities are used for supporting and enhancing the programs and the data in computer.
- Some system utilities may come embedded with OS and others may be added later on. Some examples of system utilities are:
  - **Anti-virus utility** to scan computer for viruses.
  - **Data Compression utility** to compress the files.
  - **Cryptographic utility** to encrypt and decrypt files.
  - **Disk Compression utility** to compress contents of a disk for increasing the capacity of a disk.
  - **Disk Partitioning** to divide a single drive into multiple logical drives.
  - **Disk Cleaners** to find files that have not been used for a long time.
  - **Backup Utility** to make a copy of all information stored on the disk.
  - **System Profiling** Utility provides detailed information about the software installed on the computer and the hardware attached to it.
  - **Network Managers** to check the computer network and to log events

Programming Languages

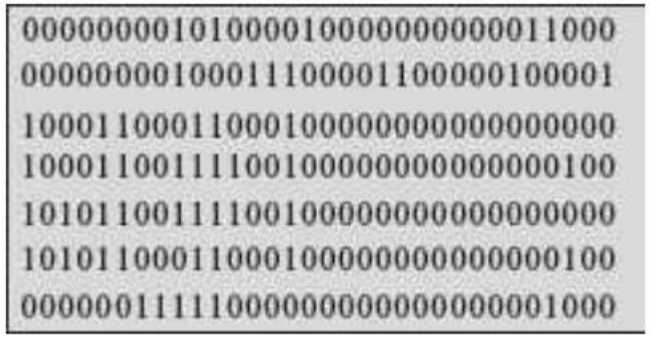
- A Programming Language consists of a set of vocabulary and grammatical rules, to express the computations and tasks that the computer has to perform.
- Programming languages are used to write a program, which controls the behavior of computer, codify the algorithms precisely, or enables the human-computer interface.




BASIC
C++
Java
High-level Languages
Assembly Languages
Machine Languages
Hardware

Machine Language

- A program written in machine language is a collection of binary digits or bits that the computer reads and interprets.
- It is a system of instructions and data executed directly by a computer's CPU.
- It is also referred to as machine code or object code. It is written as strings of 0's and 1's

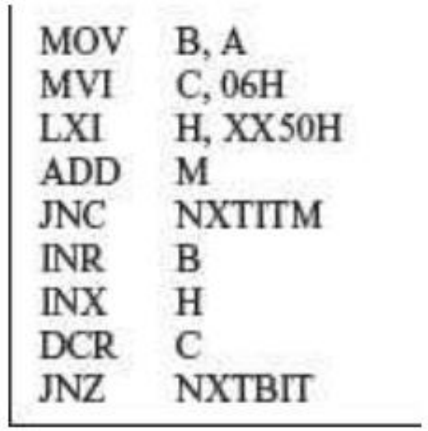


```
00000000101000010000000000011000
00000000100011100001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Figure 6.10 Machine language code

Assembly Language

- A program written in assembly language uses symbolic representation of machine codes needed to program a particular processor (CPU) or processor family.
- This representation is usually defined by the CPU manufacturer, and is based on abbreviations (called mnemonics) that help the programmer remember individual instructions, registers, etc.



```
MOV   B, A
MVI   C, 06H
LXI   H, XX50H
ADD   M
JNC   NXTITM
INR   B
INX   H
DCR   C
JNZ   NXTBIT
```

Figure 6.11 Assembly language code

High-level Language

- A program in a high-level language is written in English-like language. Such languages hide the details of CPU operations and are easily portable across computers.
- A high-level language isolates the execution semantics of computer architecture from the specification of the program, making the process of developing a program simpler and more understandable with respect to assembly and machine level languages.

Features of HLL

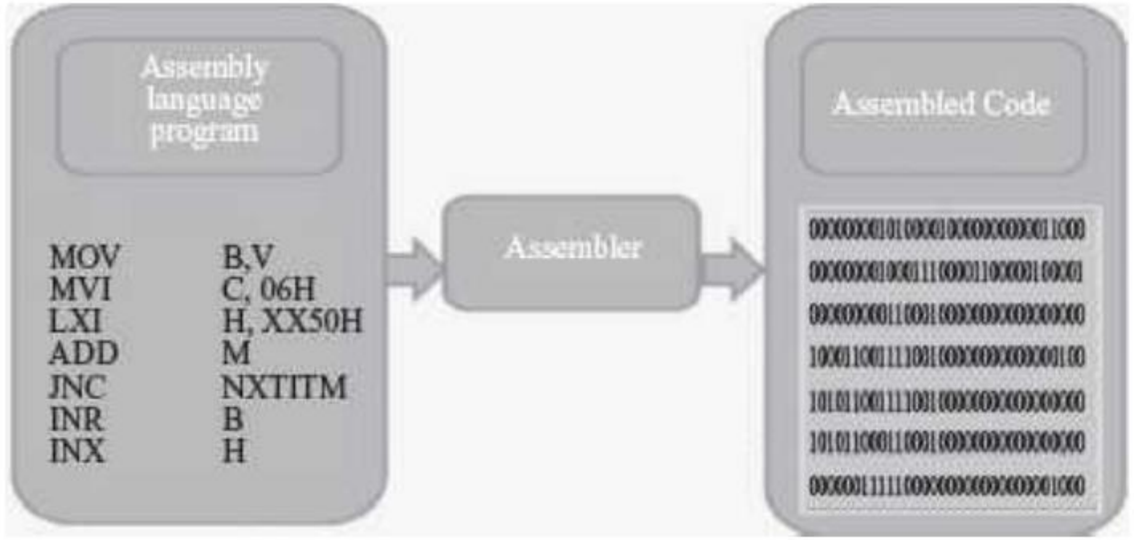
- easier to write, read or understand in high-level languages than in machine language or assembly language.
  - For example, a program written in C++ is easier to understand than a machine language program.
- Programs written in high-level languages is the source code which is converted into the object code (machine code) using translator software like interpreter or compiler.
- A line of code in high-level program may correspond to more than one line of machine code.
- Programs written in high-level languages are easily portable from one computer to another.

Translator Software

- Assembler,
- Compiler, and
- Interpreter

Assembler

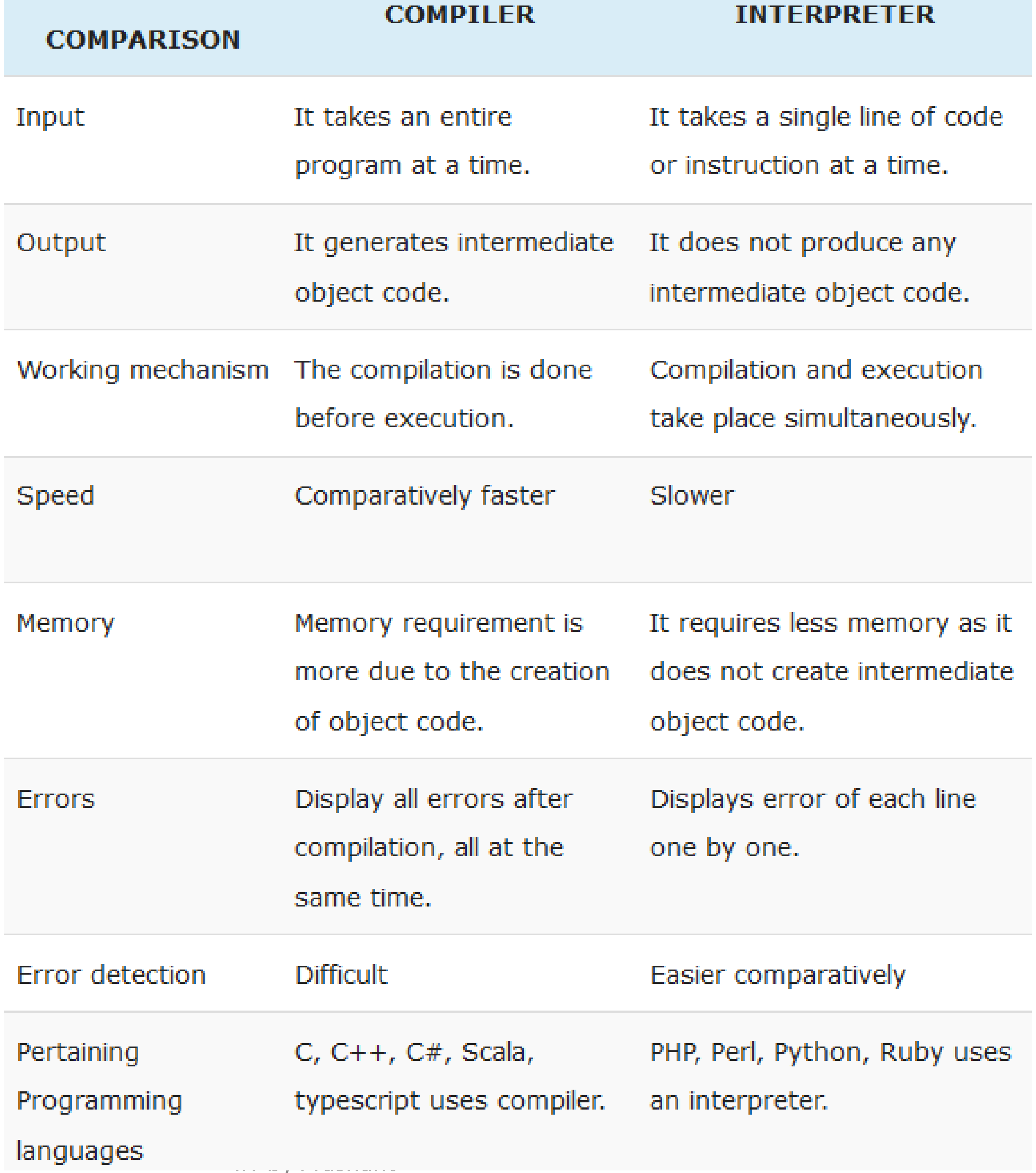
- Assembly language is also referred to as a symbolic representation of the machine code.
- Assembler is a software that converts a program written in assembly language into machine code



```
MOV     B,V
MVI     C, 06H
LXI     H, XX50H
ADD     M
JNC     NXTITM
INR     B
INX     H
```

Figure 6.14 Assembler

Compiler and Interpreter



COMPARISON	COMPILER	INTERPRETER
Input	It takes an entire program at a time.	It takes a single line of code or instruction at a time.
Output	It generates intermediate object code.	It does not produce any intermediate object code.
Working mechanism	The compilation is done before execution.	Compilation and execution take place simultaneously.
Speed	Comparatively faster	Slower
Memory	Memory requirement is more due to the creation of object code.	It requires less memory as it does not create intermediate object code.
Errors	Display all errors after compilation, all at the same time.	Displays error of each line one by one.
Error detection	Difficult	Easier comparatively
Pertaining Programming languages	C, C++, C#, Scala, typescript uses compiler.	PHP, Perl, Python, Ruby uses an interpreter.

How program executes ?

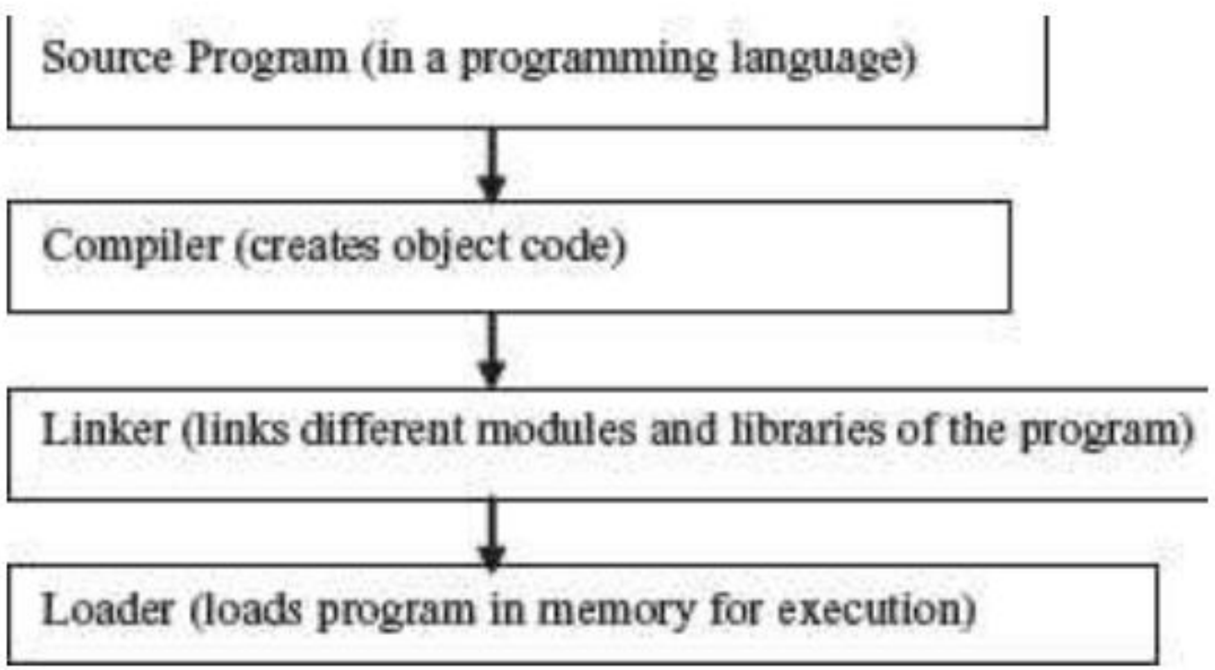

Source Program (in a programming language)
Compiler (creates object code)
Linker (links different modules and libraries of the program)
Loader (loads program in memory for execution)

Figure 6.16 Hierarchy of program execution