CHAPTER 1
SYSTEM SOFTWARE

Unit Structure

1.0 Objectives

1.1 Introduction

1.2 Operating System

1.2.1 Definition of operating system

1.2.2 Functions of Operating System

1.2.3 Operating System as User Interface

1.3 I/O System Management

1.4 Assembler

1.5 Compiler

1.6 Loader

1.7 History of Operating System

1.8 Summary

1.9 Model Question

1.0 Objectives

After going through this unit, you will be able to:

- Describe Basic Organization of Computer Systems
- Define Operating system, functions, history and Evolution
- Define assembler, linker, loader, compiler

1.1 Introduction

An operating system act as an intermediary between the user of a computer and computer hardware. The purpose of an operating system is to provide an environment in which a user can execute programs in a convenient and efficient manner.

An operating system is a software that manages the computer hardware. The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system and to prevent user programs from interfering with the proper operation of the system.

1.2 Operating System

1.2.1 Definition of Operating System:

- An Operating system is a program that controls the execution of application programs and acts as an interface between the user of a computer and the computer hardware.
- A more common definition is that the operating system is the one program running at all times on the computer (usually called the kernel), with all else being applications programs.
- An Operating system is concerned with the allocation of resources and services, such as memory, processors, devices and information. The Operating System

correspondingly includes programs to manage these resources, such as a traffic controller, a scheduler, memory management module, I/O programs, and a file system.

1.2.2 Functions of Operating System

Operating system performs three functions:

1. **Convenience:** An OS makes a computer more convenient to use.
2. **Efficiency:** An OS allows the computer system resources to be used in an efficient manner.
3. **Ability to Evolve:** An OS should be constructed in such a way as to permit the effective development, testing and introduction of new system functions without at the same time interfering with service.

1.2.3 Operating System as User Interface

- Every general purpose computer consists of the hardware, operating system, system programs, application programs. The hardware consists of memory, CPU, ALU, I/O devices, peripheral device and storage device. System program consists of compilers, loaders, editors, OS etc. The application program consists of business program, database program.
- The fig. 1.1 shows the conceptual view of a computer system

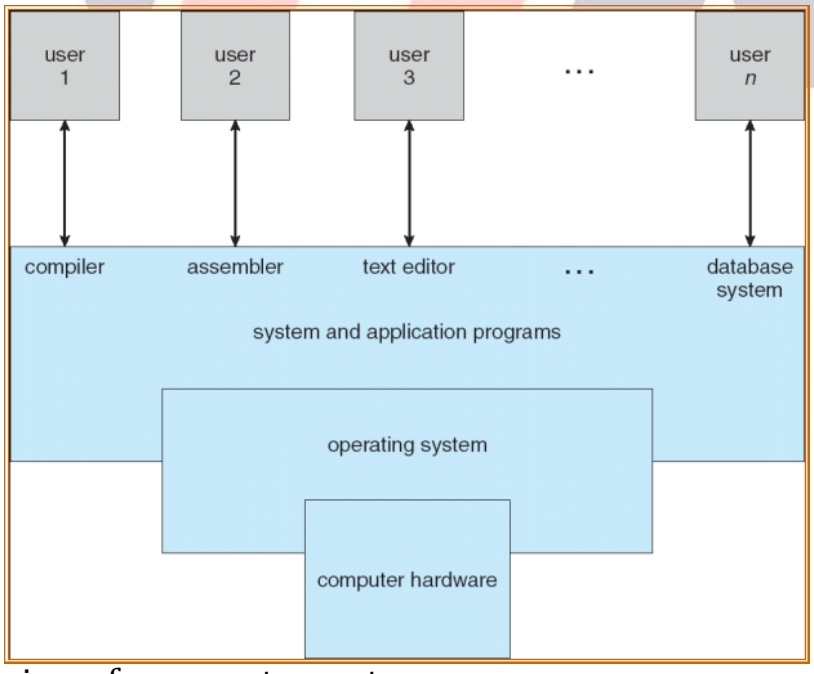

Fig 1.1 Conceptual view of a computer system

- Every computer must have an operating system to run other programs. The operating system and coordinates the use of the hardware among the various system programs and application program for a various users. It simply provides an environment within which other programs can do useful work.
- The operating system is a set of special programs that run on a computer system that allow it to work properly. It performs basic tasks such as recognizing input from the keyboard, keeping track of files and directories on the disk, sending output to the display screen and controlling a peripheral devices.

- OS is designed to serve two basic purposes :

1. It controls the allocation and use of the computing system's resources among the various user and tasks.
2. It provides an interface between the computer hardware and the programmer that simplifies and makes feasible for coding, creation, debugging of application programs.

- The operating system must support the following tasks. The tasks are :

1. Provides the facilities to create, modification of program and data files using and editor.
2. Access to the compiler for translating the user program from high level language to machine language.
3. Provide a loader program to move the compiled program code to the computer's memory for execution.
4. Provide routines that handle the details of I/O programming.

1.3 I/O System Management

I/O System Management

- The module that keeps track of the status of devices is called the I/O traffic controller. Each I/O device has a device handler that resides in a separate process associated with that device.
- The I/O subsystem consists of

1. A memory management component that includes buffering, caching and spooling.
2. A general device driver interface.

Drivers for specific hardware devices.

1.4 Assembler

Input to an assembler is an assembly language program. Output is an object program plus information that enables the loader to prepare the object program for execution. At one time, the computer programmer had at his disposal a basic machine that interpreted, through hardware, certain fundamental instructions. He would program this computer by writing a series of ones and zeros(machine language), place them into the memory of the machine.

1.5 Compiler

The high level languages – examples are FORTRAN, COBOL, ALGOL and PL/I – are processed by compilers and interpreters. A compilers is a program that accepts a source program in a ―high-level language‖ and produces a corresponding object program. An interpreter is a program that appears to execute a source program as if it was machine language. The same name (FORTRAN, COBOL etc) is often used to designate both a compiler and its associated language.

1.6 Loader

A loader is a routine that loads an object program and prepares it for execution. There are various loading schemes: absolute, relocating and direct-linking. In general, the loader must load, relocate, and link the object program. Loader is a program that

places programs into memory and prepares them for execution. In a simple loading scheme, the assembler outputs the machine language translation of a program on a secondary device and a loader is placed in core. The loader places into memory the machine language version of the user's program and transfers control to it. Since the loader program is much smaller than the assembler, thos makes more core available to user's program.

1.7 History of Operating System

- Operating systems have been evolving through the years. Following table shows the history of OS.

Generation	Year	Electronic devices used	Types of OS and devices
First	1945 – 55	Vacuum tubes	Plug boards
Second	1955 – 1965	Transistors	Batch system
Third	1965 – 1980	Integrated Circuit (IC)	Multiprogramming
Fourth	Since 1980	Large scale integration	PC

1.8 Summary

Operating Systems
An Operating system is concerned with the allocation of resources and services, such as memory, processors, devices and information. The Operating System correspondingly includes programs to manage these resources, such as a traffic controller, a scheduler, memory management module, I/O programs, and a file system.

Assembler
Input to an assembler is an assembly language program. Output is an object program plus information that enables the loader to prepare the object program for execution.
Loader
A loader is a routine that loads an object program and prepares it for execution. There

are various loading schemes: absolute, relocating and direct-linking. In general, the loader must load, relocate, and link the object program

Compilers
A compilers is a program that accepts a source program || in a high-level language|| and produces a corresponding object program.

1.9 Model Question

Model Question
Q. 1 Define Operating System?
Q. 2 Explain various function of operating system?
Q. 3 Explain I/O system Management?
Q. 4 Define & explain Assembler, Loader, Compiler?