2. Computer System Operation

A modern general-purpose C/S consists of a CPU and a number of device controllers that are connected through a common bus that provide access to shared memory. See figure (2.1).

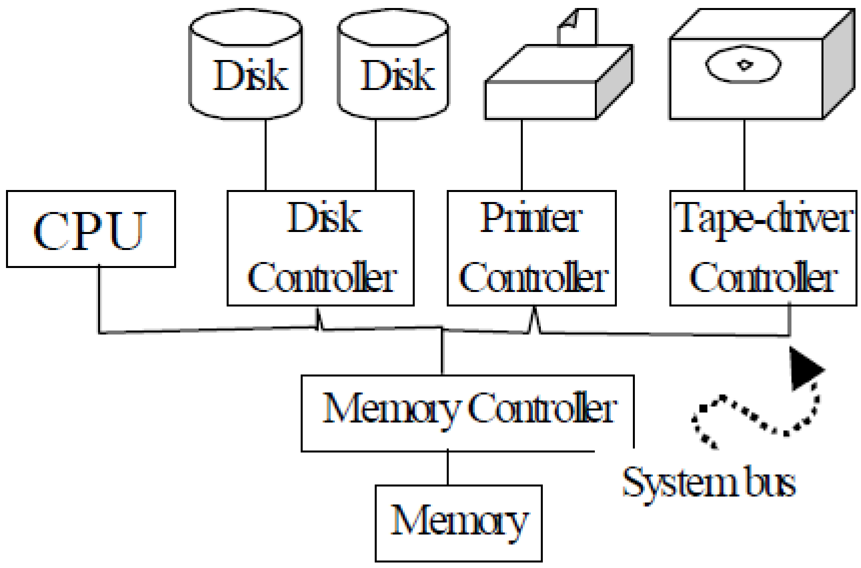

Figure (2.1) A modern computer system

When the computer to start running, it is powered up or rebooted, it needs to have an initial program to run. This initial program, or bootstrap program tends to be simple. It initializes all aspects of the system from CPU registers to device controllers to memory contents. The bootstrap program must know how to load the O.S. and how to start executing that system. The bootstrap program must locate and load into memory, the O.S kernel. The O.S then start executing the first process such as "init" and wait for some event to occur. The occurrence of an event is usually signaled by an interrupt from either H/W or S/W. The Hardware may trigger an interrupt at any time by sending a signal to the CPU by way of the system bus. Software may trigger an interrupt by executing a special operation called a system call.

2.1 I/O Structure:

Each I/O device connected to the c/s through its controller. A device controller maintains some local buffer storage and a set of special purpose registers. It is responsible for moving the data between the peripheral devices that is controls and its local buffer storage.

2.2 I/O Interrupts:

To start an I/O operation the CPU loads the appropriate registers within the device controller. The controller examines .the contents of these registers to determine what action to take. For example if it finds a read request the controller will start the transfer of data from the device to its local buffer. Once the transfer of data is complete the device controller informs the CPU that it has finished its operation.

2.3 DMA Structure:

A high-Speed device such as a tape, disk, or communication network may be able to transmit information at close to memory speeds; the CPU would need 2 microseconds to respond to each interrupt. That would not leave much time for process execution.

To solve this problem Direct Memory Access (DMA) is used for high-speed I/O devices. After setting up buffer, pointers, and counters for I/O device, the device controller transfers an entire block of data directly to or from its own buffer storage to memory with no intervention by the CPU. Only one interrupt is generated per block rather than one interrupt per byte (or word) generated for low-speed devices. A DMA controller interrupts the CPU when the transfer has been completed.

2.4 Storage Structure:

The programs must be in main memory to be executed. Main memory is the only large storage area that the processor can access directly. Each word in memory has its own address. Interaction is achieved through a sequence of load or store instruction to specific memory addresses. The load instruction moves a word from main memory to an internal register within the CPU where as the store instruction moves the content of a register to main memory.

We want the programs and data to reside in main memory permanently. This arrangement is not possible for the following two reasons:

a. Main memory is usually too small to store all needed programs and data permanently.

b. Main memory is a volatile storage device that loses its contents when power is turned off or otherwise lost.

Therefore most C/S provides secondary storage as an extension of main memory. It be able to hold large quantities of data permanently. The most common secondary-storage device is a magnetic disk, which provides storage of both programs and data. There are other many media such as floppy disks, CD-ROM's, and DVD's.

2.5 Storage Hierarchy

The variety of storage systems in a C/S can be organized in a hierarchy according to speed and their cost figure 2.2. The higher levels are expensive, but are fast.

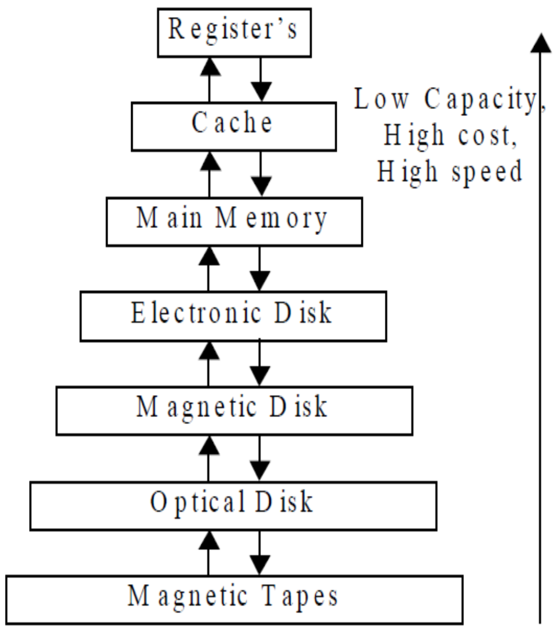

Figure (2.2) : Storage-device Hierarchy

2.6 Hardware Protection

To improve system utilization, the O.S began to share system resources among several programs simultaneously. Multi programming put several programs in memory at the same time. This sharing created both improved utilization and increased problems. When the system was run without sharing an error in a program could cause problems for only the one program that was running. With sharing many process could be affected by a bug in one program.

2.6.1 Dual-Mode Operation:

To ensure proper operation we must protect the O.S and all programs and their data from any malfunctioning program. Protection is needed for any shared resource. The approach taken is to H/W support to allow as differentiating among various modes of executions.

Therefore we need two separate modes of operation: user mode and monitor mode (also called supervisor mode, system mode, or privileged mode). A bit called mode bit is added to H/W to indicate the current mode; monitor (0) or user (1). With the mode bit we are able to distinguish between an execution that is done on behalf of the O.S, and one that is done on behalf of the user.

The dual mode of operation provides us with the means for protecting the O.S from errant users and errant users from one another. The H/W allows privileged instructions to be executed in only monitor mode.

2.6.2 I/O Protection:

To prevent a user from performing illegal I/O we define all I/O instructions to be privileged instructions. Thus user cannot issue I/O instructions directly they must do it through the O.S. For I/O protection to be complete we must be sure that a user program can never gain control of the Computer in monitor mode.

2.6.3 Memory Protection:

To ensure correct operation we must protect the interrupt vector from modification by a user program. Also we must protect the interrupt service routines in the O.S from modification. What we need to separate each program's memory space is an ability to determine the range of legal addresses that the program may access, and to protect the memory outside that space. We can provide this protection by using two registers usually a base and a limit as illustrated in figure 2.3.

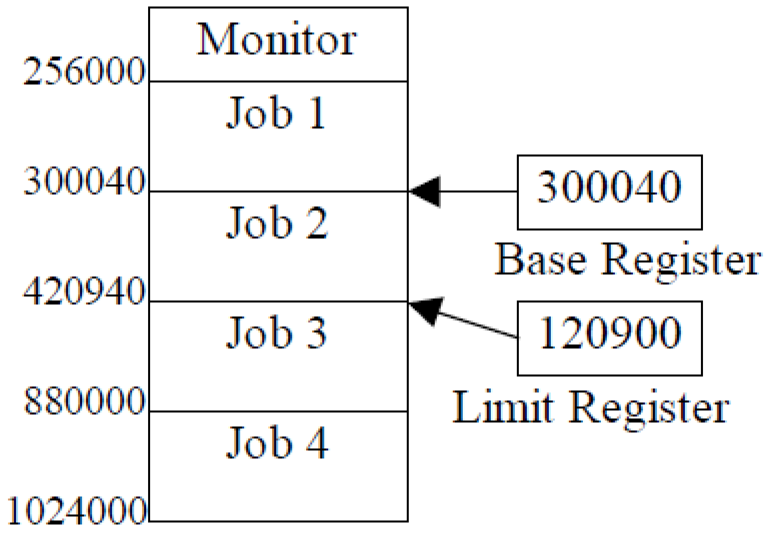

Figure 2.3 A base and Limit register

The base register holds the smallest legal physical memory address; the limit register contains the size of the range. For example if the base register holds 300040 and limit register is 120900 then the program can legally access all addresses from 300040 through 420940 inclusive.

The CPU H/W comparing every address generated in user mode with registers accomplishes this protection. Any attempt by a program executing in user mode to access monitor memory or other user's memory or other user’s memory results in a trap to the monitor which treats the attempt as a fatal error As explained in the figure (2.4).

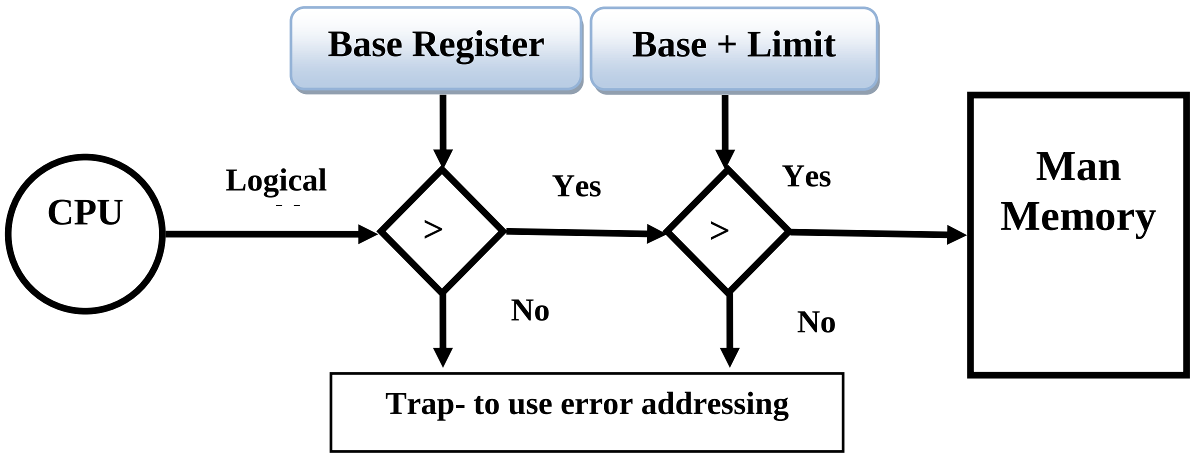

Figure 2.4 H/W address protection with base and limit registers

This scheme prevents the user program from modifying the code or data structures of either the O.S or other users. The base and limit registers can be loaded by only the O.S which uses a special privileged instruction. Since privileged instructions can be executed in only monitor mode therefore only O.S can load the base and limit registers. This scheme allows the monitor to change the value of the registers but prevents user programs from changing the registers contents.

2.6.4 CPU Protection:

The third piece of the protection is ensuring that the O.S maintains control, we must prevent a user program from an infinite loop, and never returning control to the O.S. To achieve this goal we can use a timer.

A timer can be set to interrupt the computer after a specified period. The period may be fixed (1/60 second) or variable (from 1 msecond to 1 second). To control the timer the O.S sets the counter, according to fixed-rate clock. Every time that the clock ticks the counter is decremented. When the counter reaches (0) on interrupt occurs, and control transfers automatically to the O.S, which may treat the interrupt as a fatal error or may give the program more time.

2.7 Operating System Services

• One set of operating-system services provides functions that are helpful to the user:

1. **User interface:** almost all operating systems have a user interface (UI) Varies between Command-Line (CLI), Graphics User Interface (GUI), Batch.
2. **Program execution:** the system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error).
3. **I/O operations :** a running program may require I/O, which may involve a file or an I/O device.
4. File-system manipulation: the file system is of particular interest. Obviously, programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.
5. **Communications-Processes:** may exchange information, on the same computer or between computers over a network

- Communications may be via shared memory or through message passing (packets moved by the OS)

6. Error detection: OS needs to be constantly aware of possible errors

- May occur in the CPU and memory hardware, in I/O devices, in user program.
- For each type of error, OS should take the appropriate action to ensure correct and consistent computing.
- Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system.

• Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing:

1. Resource allocation: when multiple users or multiple jobs running concurrently, resources must be allocated to each of them

- Many types of resources. Some (such as CPU cycles, main memory, and file storage) may have special allocation code, others (such as I/O devices) may have general request and release code.

2. Accounting: to keep track of which users use how much and what kinds of computer resources

3. Protection and security: the owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other

- **Protection** involves ensuring that all access to system resources is controlled
- **Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts
- If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link.

2.8 The User View

There are two methods of providing services :(System calls & System programs).

2.8.1 System Calls:

System call provides the interface between a running program and O.S, and it grouped into the major categories as follows:

a. ***Process and job control:*** End, abort, load, execute, create, terminate, wait for time, wait event, allocate and free memory. :

b. ***File manipulation:*** Create and delete file, open and close file, get file attributes and set file attributes.

c. ***Device manipulation:*** Request and release device, read, write reposition, get device attributes and set device attributes, logically attach or detach devices.

d. Information maintenance:

- Get time or date and set time or date.
- Get system data and set system data.
- Get or set process, file or device attributes.

e. Communication: Create, delete, communication, connection, send, receive messages, transfer status information, attach or detach remote devices.

2.8.2 System Programs:

Most modem O.S supply a large collection of system programs to solve common problems, and it can be divided into the following categories:

a. File manipulation. **b.** Status information. **c.** File modification.

d. Programming languages support. **e.** Program loading and execution

f. Communications

2.9 The O.S View

- O.S is event-driven programs. If there are no jobs to execute, no I/O device to serves, and no users to respond to, and O.S will sit quietly waiting for something to happen.
- Events are always signaled by the occurrence of an interrupt or a trap. Thus an O.S is interrupt driven. When an interrupt (or trap) occurs the H/W transfer control to O.S.
- The figure 2.8 shows the O.S general flow.

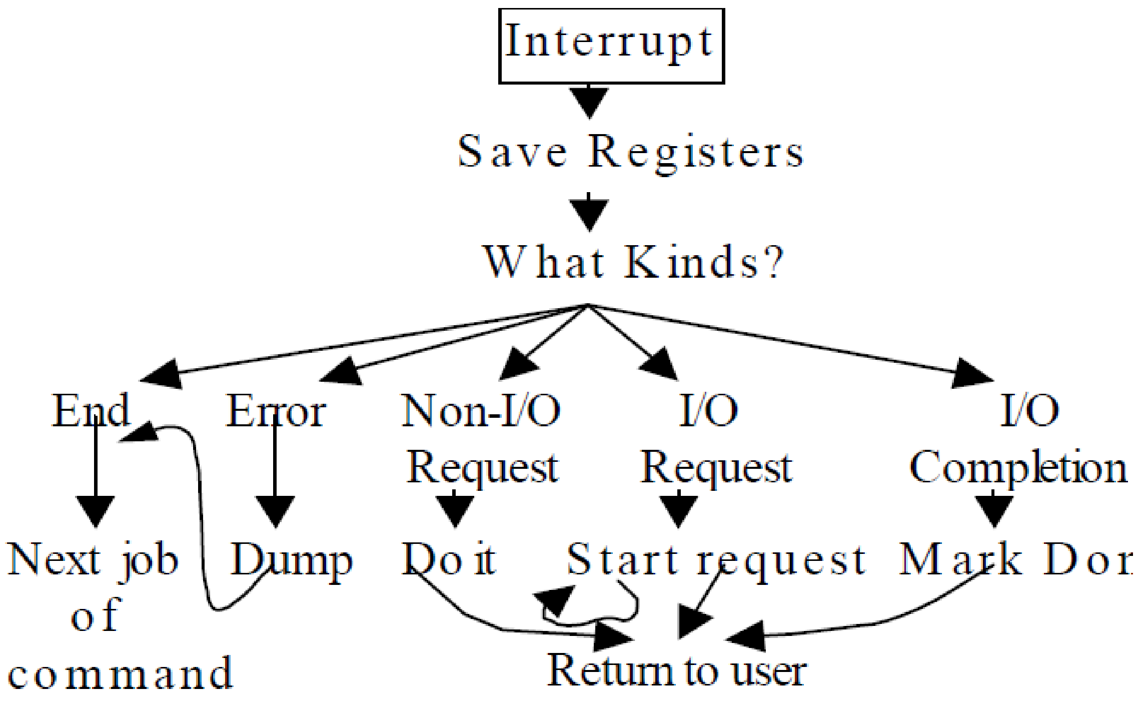

Figure (2.8) The O.S general flow