Computer System Operation:

A modern, general-purpose computer system consists of CPU and a number of device controllers that connected through a common bus that provides access to shared memory system, CPU other devices can execute concurrently competing for memory cycles.

Booting:

It is the operation of bringing operating system kernel from the secondary storage and put it in main storage to execute it in CPU. There is a program bootstrap which is performing this operation when computer is powered up or rebooted.
Bootstrap software: it is an initial program and simple it is stored in read-only memory (ROM) such as firmware or EEPROM within the computer hardware.
Jobs of Bootstrap program:

1- Initialize all the aspect of the system, from CPU registers to device controllers to memory contents.
2- Locate and load the operating system kernel into memory then the operating system starts executing the first process, such as "init" and waits for some event to occur.
The operating system then waits for some event to occur
Types of events are either software events (system call) or hardware events (signals from the hardware devices to the CPU through the system bus and known as an interrupt).
Note: all modern operating system are "interrupt driven".
Trap (exception): it is a software-generated interrupt caused either by an error (ex: division by zero or invalid memory access) or by a specific request from a user program that an operating system service be performed.
Interrupt vector (IV): it is a fixed locations (an array) in the low memory area (first 100 locations of RAM) of operating system when the interrupt occur the CPU stops what its doing and transfer execution to a fixed location (IV) contain starting address of the interrupt service routine(ISR), on completion the CPU resumes the interrupted computation.
Interrupt Service Routine: is it a routine provided to be responsible for dealing with the interrupt.

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]

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.

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
The DMA controller interrupts the CPU when the transfer has been completed

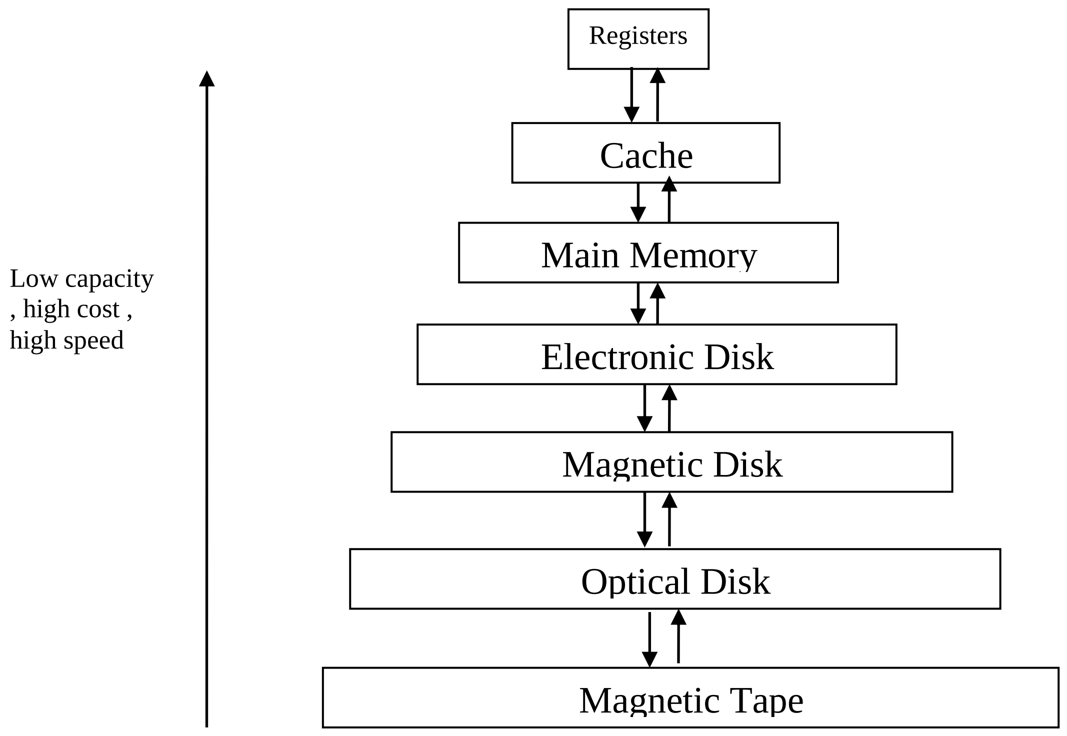
Storage Structure

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 register to main memory.
We want the programs and data to reside in 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 volatile storage device that loses its contents when power is turned off or otherwise lost.
  Therefore most C/S provide 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, ROMs , and DVDs.

Storage Hierarchy

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



Hardware protection:

when we have single user any error occur to the system then we could determined that this error must be caused by the user program ,but when we begin to dealing with spooling ,multiprogramming, and sharing disk to hold many users data this sharing both improved utilization and increase problems .
In multiprogramming system, where one erroneous program might modify the program or data of another program, or even the resident monitor itself. MS-DOS and the Macintosh OS both allow this kind of error.
A properly designed operating system must ensure that an incorrect (or malicious) program cannot cause other program to execute incorrectly.
Many programming error are detected by the hardware these error are normally handled by the operating system.

Dual-Mode Operation:

To ensure proper operation, we must protect the operating system and all other programs and their data from any malfunctioning program.
The approach taken by many operating systems provides hardware support that allows us to differentiate among various modes of execution.
A bit, called the mode bit is added to the hardware of the computer to indicates the current mode: monitor (0) or user (1) with mode bit we could distinguish between a task that is executed on behalf of the operating system , and one that is executed on behalf of the user.

I/O Operation Protection:

A use program may disrupt the normal operation of the system by issuing illegal I/O instruction we can use various mechanisms to ensure that such disruption can not take place in the system.
One of them is by defining all I/O instructions to be privileged instructions. Thus users cannot issue I/O instructions directly they must do it through the operating system, by execute a system call to request that the operating system performing I/O in its behalf. The operating system, executing in monitor mode, check that the request is valid, and (if the request is valid) does the I/O requested. The operating system then returns to the user.

Memory Protection:

To insure correct operation, we must protect the interrupt vector and interrupt service routine from modification by a user program. This protection must be provided by the hardware, we need the ability to determine the range of legal addresses that the program may access, and to protect the memory outside that space. We could provided the protection by using two registers a base register and limit register
Base register hold the smallest legal physical memory address.
Limit register: contains the size of the range.
This protection is accomplished by the CPU hardware comparing every address generated in user mode with the registers. Any attempt by a program executing in user mode to access monitor memory or other users' memory results in a trap to the monitor, which treats the attempts as a fatal error.

CPU Protection:

In addition to protecting I/O and memory we must insure that the operating system maintains control. We must prevent the user from getting stuck in an infinite loop or not calling system services, and never returning control to the operating system. To accomplish this goal, we can use a timer.
Timer can be set to interrupt the computer after a specified period. The period may be fixed (for example, 1/60 second) or variable (for example, from 1 millisecond to 1 second) A variable timer is generally implemented by a fixed rate clock and a counter.
We can use the timer to prevent a user program from running too long Simple technique is to initialize a counter with the mount of time that a program is allowed to run.
Amore common use of timer is to implement time sharing. In the most case, the timer could be set to interrupt every N millisecond, where N is the time slice that each user is allowed to execute before the next user get control of the CPU. The operating system is invoked to perform housekeeping tasks.
This procedure is known as a context switching, following a context switch, the next program continues with its execution from the point at which it left off.