Srinivasan College of Arts & Science

(Affiliated to Bharathidasan University,Tiruchirappalli)

Perambalur – 621 212

COURSR MATERIAL

OPERATING SYSTEMS

DEPARTMENT OF COMPUTER SCIENCE

& INFORMATION TECHNOLOGY


Operating
Systems
roseindia.net

CORE COURSE VIII
OPERATING SYSTEMS (16SCCCA6 / 16SCCCS8)
SEMESTER VI

Unit I

Introducing Operating Systems Introduction - What Is an Operating System-Operating System Software -A Brief History of Machine Hardware -Types of Operating Systems -Brief History of Operating System Development-Object-Oriented Design

Unit II

Memory Management Early Systems: Single-User Contiguous Scheme – Fixed Partitions-Dynamic Partitions-Best-Fit versus First-Fit Allocation - Deallocation - Relocatable Dynamic Partitions. Virtual Memory: Paged Memory Allocation-Demand Paging-Page Replacement Policies and Concepts -Segmented Memory Allocation-Segmented /Demand Paged Memory Allocation - Virtual Memory-Cache Memory

Unit III

Processor Management Overview-About Multi-Core Technologies-Job Scheduling Versus Process Scheduling-Process Scheduler-Process Scheduling Policies-Process Scheduling Algorithms -A Word About Interrupts-Deadlock-Seven Cases of Deadlock -Conditions for Deadlock-Modeling Deadlock-Strategies for Handling Deadlocks–Starvation- Concurrent Processes: What Is Parallel Processing-Evolution of Multiprocessors-Introduction to Multi-Core Processors-Typical Multiprocessing Configurations--Process Synchronization Software

Unit IV

Device Management Types of Devices-Sequential Access Storage Media-Direct Access Storage Devices-Magnetic Disk Drive Access Times- Components of the I/O Subsystem- Communication among Devices-Management of I/O Requests

Unit: V

File Management The File Manager -Interacting with the File Manager-File Organization - Physical Storage Allocation -Access Methods-Levels in a File Management System - Access Control Verification Module

Text Book:

1.Understanding Operating Systems, Ann McIver McHoes and Ida M. Flynn, Course Technology, Cengage Learning, 2011.

Reference Book:

1.OperatingSystems,AchyutGodbole and AtulKahate , McGraw Hill Publishing, 2010

OPERATING SYSTEMS

TWO MARK QUESTIONS

1. What is an Operating System?

An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

2. What is the function of an operating System?

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

3. Define Memory Management.

Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address.

Main memory provides a fast storage that can be accessed directly by the CPU. For a program to be executed, it must in the main memory.

4. What are the memory management functions?

- Keeps tracks of primary memory, i.e., what part of it are in use by whom, what parts are not in use?

- In multiprogramming, the OS decides which process will get memory when and how much.
- Allocates the memory when a process requests it to do so.
- De-allocates the memory when a process no longer needs it or has been terminated.

5. Define Processor Management

In multiprogramming environment, the OS decides which process gets the processor when and for how much time. This function is called **process scheduling**.

6. What are the processor management functions?

- Keeps tracks of processor and status of process. The program responsible for this task is known as **traffic controller**.

- Allocates the processor (CPU) to a process.
- De-allocates processor when a process is no longer required.

7. Define Device Management.

An Operating System manages device communication via their respective drivers.

8. What are the device management functions?

- Keeps tracks of all devices. Program responsible for this task is known as the **I/O controller**.

- Decides which process gets the device when and for how much time.
- Allocates the device in the efficient way.
- De-allocates devices.

9. Define File Management.

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.

10. What are the file management functions?

- Keeps track of information, location, uses, status etc. The collective facilities are often known as **file system**.
- Decides who gets the resources.
- Allocates the resources.

- De-allocates the resources.

11. What is meant by Spooling

Spooling is an acronym for simultaneous peripheral operations on line. Spooling refers to putting data of various I/O jobs in a buffer. This buffer is a special area in memory or hard disk which is accessible to I/O devices.

12. What is meant by Kernel?

The kernel is the heart of the operating system. It is the first program loaded when the computer starts up, it manages computer resources, and it handles requests from system programs and applications.

13. Define Paging.

Paging is a memory management technique in which process address space is broken into blocks of the same size called pages (size is power of 2, between 512 bytes and 8192 bytes). The size of the process is measured in the number of pages.

14. Define Page fault.

When the page (data) requested by a program is not available in the memory, it is called as a page fault.

15. Define Fragmentation.

The most efficient way to store a file is in a contiguous physical block. However, over time, as storage device reads and writes data, fewer blocks of free space are available. In some cases, it may be necessary to split a file into multiple areas of a storage device. This is called file fragmentation.

16. Define Segmentation.

Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program.

17. Define Cache Memory.

The Cache Memory is the Memory which is very nearest to the CPU , all the Recent Instructions are Stored into the Cache Memory.

18. What is the memory management in operating system?

Memory management is the process of controlling and coordinating computer memory, assigning portions called blocks to various running programs to optimize overall system performance. Memory management resides in hardware, in the OS (operating system), and in programs and applications.

19. What is meant by memory partitioning in OS?

Memory Partitioning- means partitioning the memory into small units or blocks & load the data or code into the blocks, & use this data & code when required by a process. Paging- is a techique to manage the memory in the form of equal soze of blocks called pages.

20. What do you mean by process address space?

Process Address Space. The process address space is the set of logical addresses that a process references in its code.

21. Differentiate logical and physical addresses.

The set of all logical addresses generated by a program is referred to as a logical address space. The set of all physical addresses corresponding to these logical addresses is referred to as a physical address space.

22. Write about types of Fragmentation.

External fragmentation

Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used

Internal fragmentation

Memory block assigned to process is bigger. Some portion of memory is left unused, as it cannot be used by another process.

23. What are frames in operating system?

Main memory is divided into small fixed-sized blocks of (physical) memory called frames and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.

24. Write the Advantages and Disadvantages of Paging.

A list of advantages and disadvantages of paging is given.

- Paging reduces external fragmentation, but still suffer from internal fragmentation.
- Paging is simple to implement and assumed as an efficient memory management technique.
- Due to equal size of the pages and frames, swapping becomes very easy.
- Page table requires extra memory space, so may not be good for a system having small RAM.

25. What is virtual memory?

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard disk that's set up to emulate the computer's RAM.

26. Define Process Scheduling.

Process scheduling is an essential part of Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

27. What is meant by deadlock?

A deadlock is a situation in which two computer programs sharing the same resource are effectively preventing each other from accessing the resource, resulting in both programs ceasing to function.

28. What are the conditions occur in deadlock?

- **Mutual Exclusion** : At least one unsharable resource - processes claim exclusive control of resources they need
- **Hold and Wait** : Process *holds* one resource while waiting for another
- **No Preemption** : Resources only released voluntarily - no interruption possible (i.e. cannot be forcefully withdrawn by another process)
- **Circular Wait** : Circular chain of processes - each waiting for a resource held by another

29. Define Starvation.

Starvation is a problem encountered in concurrent computing where a process is perpetually denied necessary resources to process its work. Starvation may be caused by errors in a scheduling or mutual exclusion algorithm.

30. Define Parallel processing.

Parallel processing is the processing of program instructions by dividing them among multiple processors with the objective of running a program in less time.

31. Define SAM.

Sequential access memory (SAM) is a class of data storage devices that read stored data in a sequence.

32. Define RAM.

Random Access Memory (RAM) where data can be accessed in any order.

33. Define Access control.

Access Control is the process of mediating the requests for accessing data in a system and determining whether the request should be granted or denied.

FIVE MARK QUESTIONS

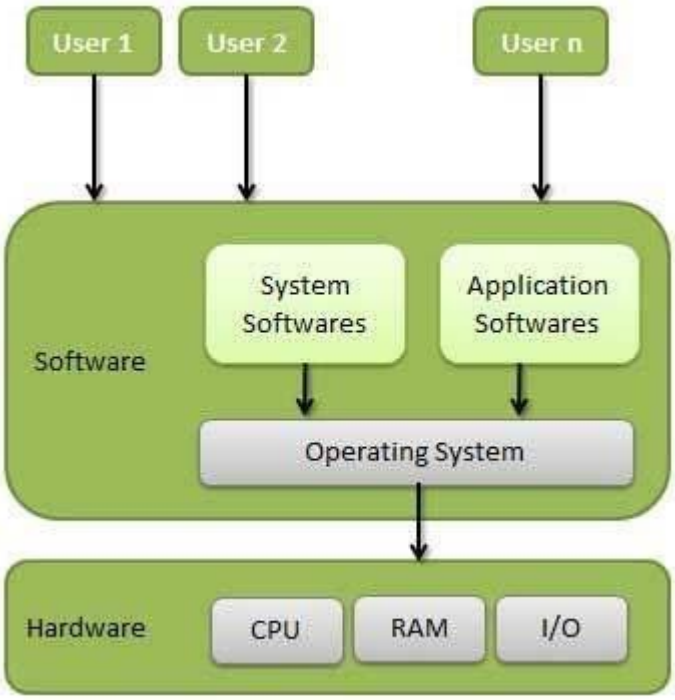
1. Explain Operating System Software.

An Operating System (OS) is an interface between computer user and computer hardware. An operating system is software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

Some popular Operating Systems include Linux, Windows, OS X, VMS, OS/400, AIX, z/OS, etc.

Definition

An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.



2. Explain a Brief History of Operating Systems

The first operating system was created by General Motors in 1956 to run a single IBM mainframe computer. Other IBM mainframe owners followed suit and created their own operating systems. As you can imagine, the earliest operating systems varied wildly from one computer to the next, and while they did make it easier to write programs, they did not allow programs to be used on more than one mainframe without a complete rewrite.

In the 1960s, IBM was the first computer manufacturer to take on the task of operating system development and began distributing operating systems with their computers. However, IBM wasn't the only vendor creating operating systems during this time. Control Data Corporation, Computer Sciences Corporation, Burroughs Corporation, GE, Digital Equipment Corporation, and Xerox all released mainframe operating systems in the 1960s as well.

In the late 1960s, the first version of the Unix operating system was developed. Written in C, and freely available during it's earliest years, Unix was easily ported to new systems and rapidly achieved broad acceptance. Many modern operating systems, including Apple OS X and all Linux flavors, trace their roots back to Unix.

Microsoft Windows was developed in response to a request from IBM for an operating system to run its range of personal computers. The first OS built by Microsoft wasn't called Windows, it was called MS-DOS and was built in 1981 by purchasing the 86-DOS operating system from Seattle Computer Products and modifying it to meet IBM's requirements. The name Windows was first used in 1985 when a graphical user interface was created and paired with MS-DOS.

Apple OS X, Microsoft Windows, and the various forms of Linux (including Android) now command the vast majority of the modern operating system market.

3. What are the Parts of an Operating System

Operating systems are built out of two main parts:

- The kernel;
- System programs.

The kernel is the heart of the operating system. It is the first program loaded when the computer starts up, it manages computer resources, and it handles requests from system programs and applications.

System programs run on top of the kernel. They aren't used to perform useful work, instead, they are the programs necessary to connect the kernel to user applications and peripheral devices. Device drivers, file systems, networking programs, and system utilities like disk defragmeters are all examples of system programs.

Application programs aren't part of the operating system and are the programs used to perform useful work. Word processing applications, browsers, and media player are common types of application programs. Application programs are managed and enabled by the kernel, and use system programs to access computer periphery devices and hardware.

4. What exactly is an Object Oriented Operating System?

- an operating system implemented using the techniques of object orientation (i.e. implemented in an Object Oriented Programming Language using inheritance, PolyMorphism, etc.)?
- an operating system designed according to the principles of object orientation (i.e. the basic abstractions provided by the operating system are represented internally and externally as objects with state, behavior, and identity, and can be acted upon and maybe even extended as one would act and extend objects e.g. in a Smalltalk Programming Environment)?
- Consider device drivers, interrupts, and file systems. Device drivers were some of the first polymorphic components. New drivers could be loaded or compiled into an operating system, without the OS having prior knowledge of the driver type. The driver only had to conform to a specific interface. Interrupts provided a very basic messaging system. Hardware or software could call an interrupt without know precisely what would handle it or even if it would be handled. Software and occasionally hardware would be installed to handle the interrupt and would need to register itself to the system to be the interrupt handler. File systems are really an extension of the device driver example, but a routine is really unconcerned where a file exists; it merely reads from it and writes to it. The details of whether the file is on a floppy disk, a hard drive, or a remote file server are hidden.

See e.g. Choices Object Oriented Operating System and Blue Abyss.

5. Explain Single-User Contiguous Scheme.

The main memory must accommodate both the operating system and the various user processes. We therefore need to allocate different parts of the main memory in the most efficient way possible.

The memory is usually divided into two partitions: one for the resident operating system, and one for the user processes. We may place the operating system in either low memory or high memory. With this approach each process is contained in a single contiguous section of memory.

One of the simplest methods for memory allocation is to divide memory into several fixed-sized partitions. Each partition may contain exactly one process. In this multiple-partition method, when a partition is free, a process is selected from the input queue and is loaded into the free partition. When the process terminates, the partition becomes available for another process. The operating system keeps a table indicating which parts of memory are available and which are occupied. Finally, when a process arrives and needs memory, a memory section large enough for this process is provided.

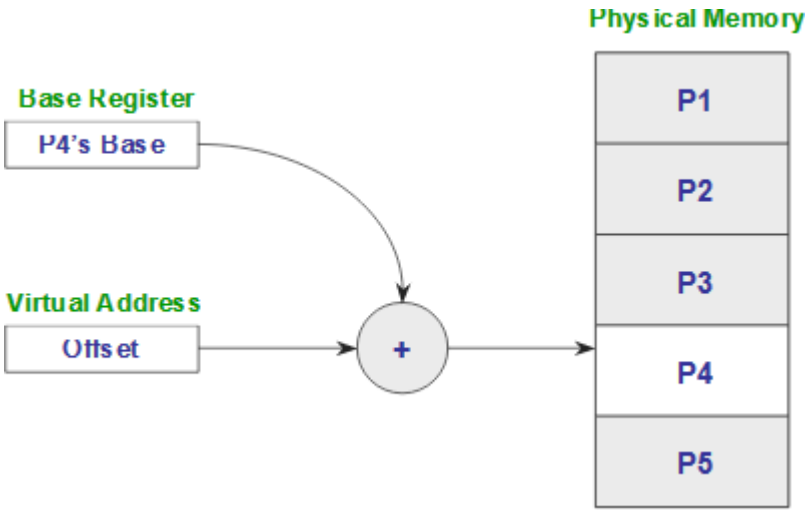
6. Write short note on Fixed Partitions.

Physical memory is broken up into fixed partitions
- Hardware requirements: base register
- Physical address = virtual address + base register
- Base register loaded by OS when it switches to a process
- Size of each partition is the same and fixed
- How do we provide protection?

Advantages
- Easy to implement, fast context switch

Problems
- Internal fragmentation : memory in a partition not used by a

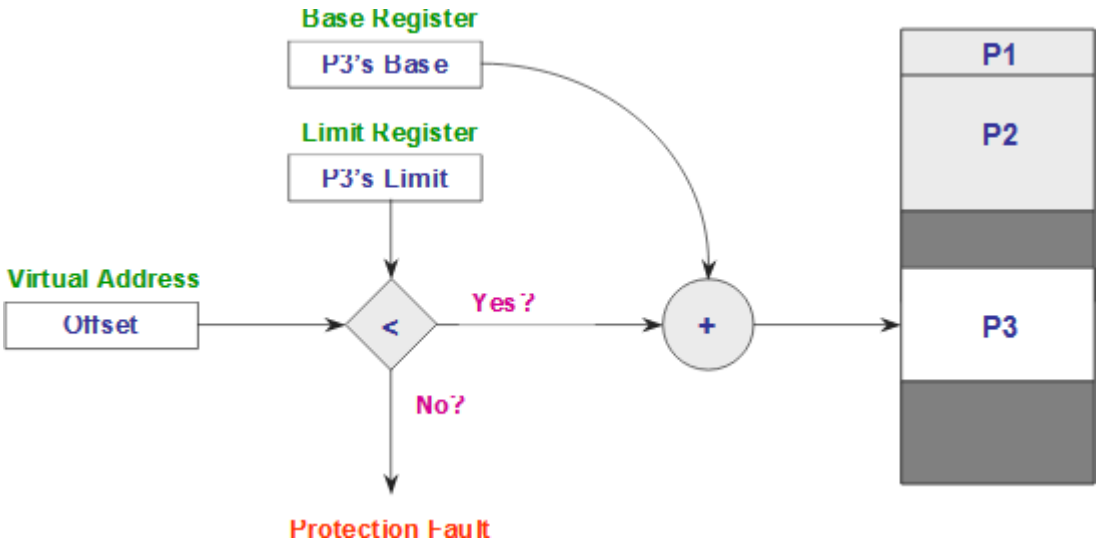
process is not available to other processes
◆Partition size : one size does not fit all (very large processes?)



7. Write short note on Dynamic Partitions.
Natural extension -- physical memory is broken up into variable sized partitions
◆Hardware requirements: base register and limit register
◆Physical address = virtual address + base register
◆Why do we need the limit register? Protection
◆If (physical address > base + limit) then exception fault

Advantages
◆No internal fragmentation : allocate just enough for process

Problems
◆External fragmentation : job loading and unloading produces empty holes scattered throughout memory



8. Write Short note on Segment memory management.

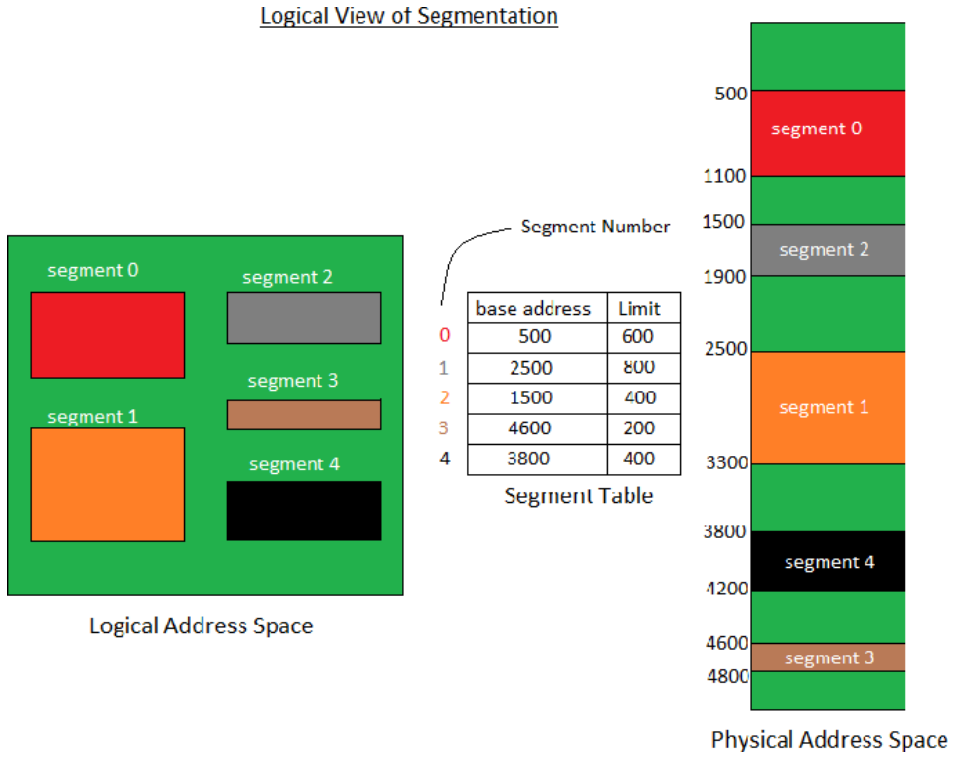
A Memory Management technique in which memory is divided into variable sized chunks which can be allocated to processes. Each chunk is called a **Segment**.

A table stores the information about all such segments and is called **Segment Table.**

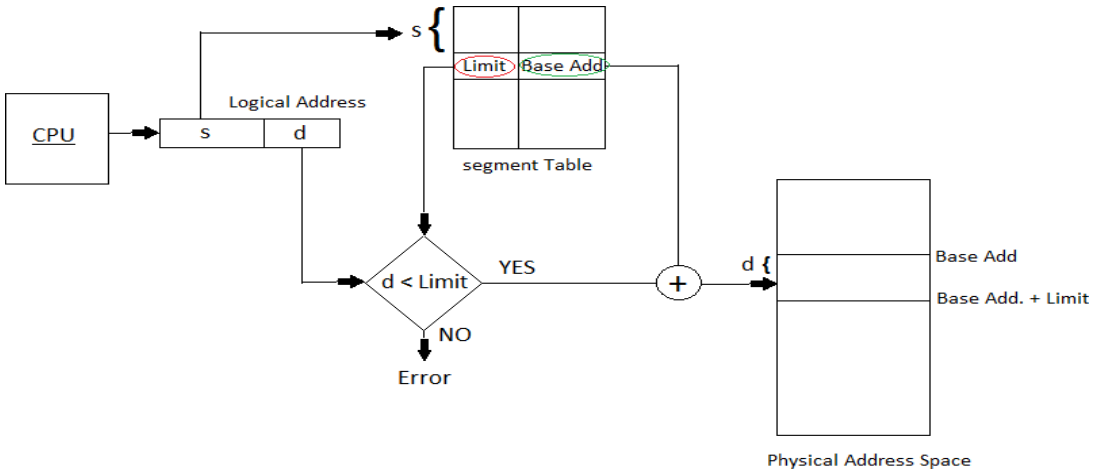
Segment Table: It maps two dimensional Logical address into one dimensional Physical address.

It's each table entry has

- **Base Address:** It contains the starting physical address where the segments reside in memory.
- **Limit:** It specifies the length of the segment.



Translation of Two dimensional Logical Address to one dimensional Physical Address.



Address generated by the CPU is divided into:

- Segment number (s): Number of bits required to represent the segment.
- Segment offset (d): Number of bits required to represent the size of the segment.

Advantages of Segmentation:

- No Internal fragmentation.
- Segment Table consumes less space in comparison to Page table in paging.

Disadvantage of Segmentation:

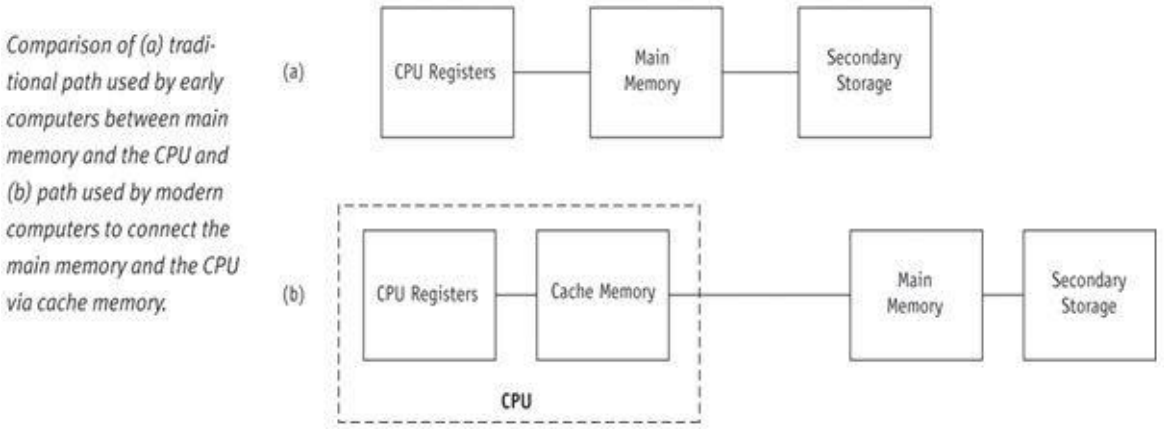
- As processes are loaded and removed from the memory, the free memory space is broken into little pieces, causing External fragmentation.

9. Explain Virtual Memory

- Allows program execution even if not stored entirely in memory
- Requires cooperation between memory manager and processor hardware
- Advantages
  - Job size not restricted to size of main memory
  - Memory used more efficiently
  - Allows an unlimited amount of multiprogramming
  - Eliminates external fragmentation and minimizes internal fragmentation
- Disadvantages
  - Increased processor hardware costs
  - Increased overhead for handling paging interrupts
  - Increased software complexity to prevent thrashing

10. Explain Cache Memory

- Small high-speed intermediate memory unit
- Performance of computer system increased
  - Memory access time significantly reduced
  - Faster processor access compared to main memory
  - Stores frequently used data and instructions
- Two levels of cache
  - L2: Connected to CPU; contains copy of bus data
  - L1: Pair built into CPU; stores instructions and data
- Data/instructions move from main memory to cache
  - Uses methods similar to paging algorithms



Comparison of (a) traditional path used by early computers between main memory and the CPU and (b) path used by modern computers to connect the main memory and the CPU via cache memory.

- Four cache memory design factors
  - Cache size, block size, block replacement algorithm, and rewrite policy
- An optimal selection of cache and replacement algorithm necessary
  - May lead to 80-90% of all requests in cache
- Efficiency measures
  - Cache hit ratio (h)
    - Percentage of total memory request found in cache
  - Miss ratio (1-h)
  - Average memory access time
    - AvgCacheAccessTime + (1-h) * AvgMemACCTime

11. Explain about Types of Process Scheduler

Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types −

- Long-Term Scheduler
- Short-Term Scheduler
- Medium-Term Scheduler

Long Term Scheduler

It is also called a **job scheduler**. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.

On some systems, the long-term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

Short Term Scheduler

It is also called as **CPU scheduler**. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.

Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

Medium Term Scheduler

Medium-term scheduling is a part of **swapping**. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.

A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called **swapping**, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

12. Explain about Process Scheduling Policies.

We will concentrate on scheduling at the level of selecting among a set of ready processes. Scheduler is invoked whenever the operating system must select a user-level process to execute:

•after process creation/termination

•a process blocks on I/O

•I/O interrupt occurs

•clock interrupt occurs (if preemptive)

Types of processes:

•interactive jobs

•low priority, cpu bound jobs that use excess processor capacity (e.g., calculating π to 101000000 decimal places)

•somewhere in between Distinguish between a short and long process. Based on the time a process runs when it gets the CPU. An I/O bound process is short and a CPU bound process is long.

Note: The idea of short vs. long is determined by how much of its time slice that a process uses, not the total amount of time it executes

13. Comparison among Scheduler.

S.N.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of	It provides lesser	It reduces the degree of

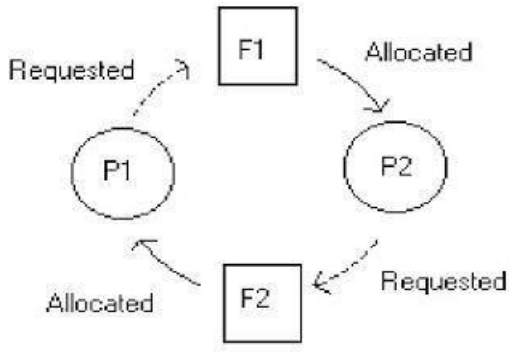
	multiprogramming	control over degree of multiprogramming	multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

14. Explain seven cases of deadlocks.

Case 1: Deadlocks on File Request

- If jobs can request and hold files for duration of their execution, deadlock can occur.Any other programs that require F1 or F2 are put on hold as long as this situation continues. Deadlock remains until a programs is withdrawn or forcibly removed and its file is released.

example:



Case 2: Deadlocks in Database

-Deadlock can occur if 2 processes access & lock records in database.

Three different levels of locking entire database for duration of request
a subsection of the database individual record until process is completed.

If don't use locks, can lead to a race condition.

Example:

1. P1 accesses R1 and locks it

2. P2 accesses R2 and locks it.

3. P1 requests R2, which is locked by P2.

4. P2 requests R1, which is locked by P1.

Case 3: Deadlocks in Dedicated Device Allocation

-Deadlock can occur when there is a limited number of dedicated devices.

E.g., printers, plotters or tape drives.

Example:

P1 requests tape drive 1 and gets it.

P2 requests tape drive 2 and gets it.

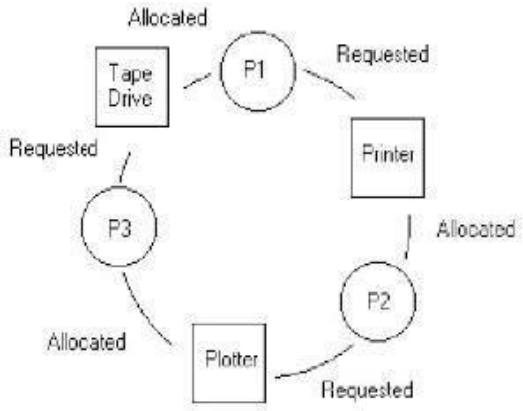
P1 requests tape drive 2 but is blocked.

P2 requests tape drive 1 but is blocked.

Case 4. Deadlocks in Multiple Device Allocation

-Deadlocks can happen when several processes request, and hold on to, dedicated devices while other processes act in a similar manner.

Example:



Case 5: Deadlocks in Spooling

-Most systems have transformed dedicated devices such as a printer into a sharable device by installing a high-speed device, a disk, between it and the CPU. Disk accepts output from several users and acts as a temporary storage area for all output until printer is ready to accept it (spooling).

Example:

If printer needs all of a job's output before it will begin printing, but spooling system fills available disk space with only partially completed output, then a deadlock can occur.

Case 6: Deadlocks in Network

-a network that's congested or has filled a large percentage of it's I/O buffer space can become deadlocked if it doesn't have protocols to control the flow of messages through the network.

Case 7: Deadlocks in Disk Sharing

-Disks are designed to be shared, so it’s not uncommon for 2 processes access different areas of same disk. Without controls to regulate use of disk drive, competing processes could send conflicting commands and deadlock the system.

15. Explain about deadlock modeling.

A computer scientist/programmer named Holt showed how the four conditions for deadlock (as described in the previous tutorial) can be modelled using directed graphs. These graphs shown below.

In the above resource allocation graphs, the figure A, B, and C represents:

Figure	Represents
(A)	Holding a resource
(B)	Requesting a resource
(C)	Deadlock

According to Hold, these graphs have the following two types of nodes:

Processes - shown in graph as circles

Resources - shown in graph as squares

Here, in this graph, an arc from a square (resource node) to a circle (process node) means that the resource has previously been requested by, granted to, and is currently held by that process.

As you can see from the figure given above, resource R is currently assigned to process A in figure A. An arc from a process to a resource means that the process is currently blocked waiting for that resource.

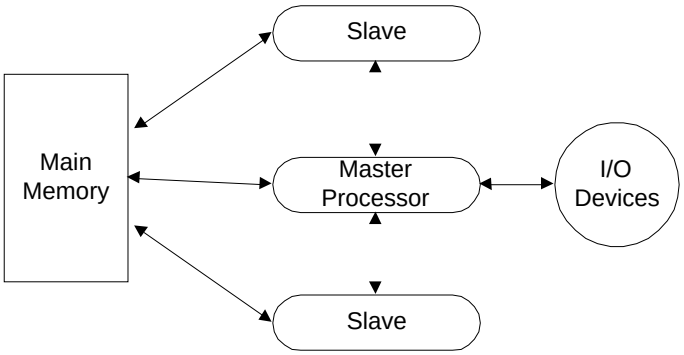
As you can see from the above figure, in the figure B, process B is waiting for resource S. And, in the figure C, we see a deadlock, that is, the process C is waiting for the resource T, which is currently held by the process D. Here, process D is not about the release the resource T because it is waiting for resource U, held by C. Here both processes will wait forever.

16. Discuss about Typical Multiprocessing Configurations.

- Master/slave
- Loosely coupled
- Symmetric

Master/Slave Multiprocessing Configuration

- Asymmetric configuration.
- Single-processor system with additional "slave" processors.
- Each slave, all files, all devices, and memory managed by primary "master" processor.
- Master processor maintains status of all processes in system, performs storage management activities, schedules work for the other processors, and executes all control programs.

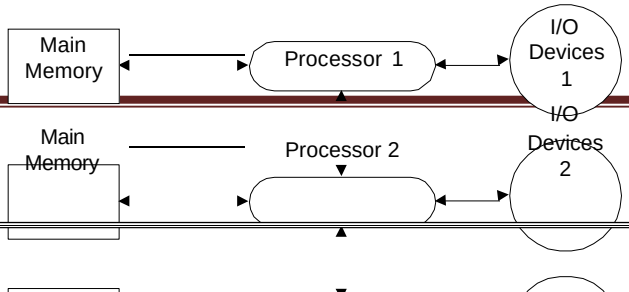


Pros & Cons of Master/Slaves

- The primary advantage is its simplicity.
- Reliability is no higher than for a single processor system because if master processor fails, entire system fails.
- Can lead to poor use of resources because if a slave processor becomes free while master is busy, slave must wait until the master can assign more work to it.
- Increases number of interrupts because all slaves must interrupt master every time they need OS intervention (e.g., I/O requests).

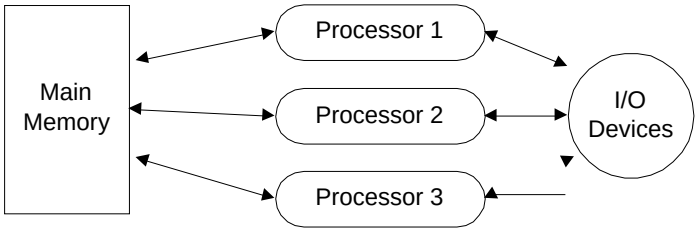
Loosely Coupled Multiprocessing Configuration

- Features several complete computer systems, each with own memory, I/O devices, CPU, & OS.
- Each processor controls own resources, maintains own commands & I/O management tables.
- Each processor can communicate and cooperate with the others.
- Each processor must have "global" tables indicating jobs each processor has been allocated.



Symmetric Multiprocessing Configuration

- Processor scheduling is decentralized.
- A single copy of OS& a global table listing each process and its status is stored in a common area of memory so every processor has access to it.
- Each processor uses same scheduling algorithm to select which process it will run next.



Advantages of Symmetric over Loosely Coupled Configurations

- More reliable.
- Uses resources effectively.
- Can balance loads well.
- Can degrade gracefully in the event of a failure.
- Most difficult to implement because processes must be well synchronized to avoid problems of races and deadlocks.

17. Explain about Direct Access Storage Devices

Direct Access Storage Devices (DASDs)

Direct access storage devices (DASDs) are *fixed* or *removable* storage devices. Typically, these devices are hard disks. A fixed storage device is any storage device defined during system configuration to be an integral part of the system DASD. The operating system detects an error if a fixed storage device is not available at some time during normal operation.

A removable storage device is any storage device defined by the person who administers your system during system configuration to be an optional part of the system DASD. The removable storage device can be removed from the system at any time during normal operation. As long as the device is logically unmounted first, the operating system does not detect an error.

The following types of devices are not considered DASD and are not supported by the logical volume manager (LVM):

- Diskettes
- CD-ROM (compact disk read-only memory)
- DVD-ROM (DVD read-only memory)
- WORM (write-once read-many)

18. Explain about Magnetic Disk Drive Access.

Magnetic storage or **magnetic recording** is the storage of data on a magnetised medium. Magnetic storage uses different patterns of magnetisation in a magnetisable material to store data and is a form of non-volatile memory. The information is accessed using one or more read/write heads.

As of 2017, magnetic storage media, primarily hard disks, are widely used to store computer data as well as audio and video signals. In the field of computing, the term *magnetic storage* is preferred and in the field of audio and video production, the term *magnetic recording* is more commonly used. The distinction is less technical and more a matter of preference. Other examples of magnetic storage media include floppy disks, magnetic recording tape, and magnetic stripes on credit cards.

19. Explain about Components of IO Subsystem.

Results in IO subsystem generally being the "messiest" part of the OS

- So much variety of devices
- So many applications
- So many dimensions of variation:
- Character-stream or block
- Sequential or random-access
- Synchronous or asynchronous
- Shareable or dedicated
- Speed of operation
- Read-write, read-only, or write-only

Thus, completely homogenising device API is not possible so OS generally splits devices into four classes

DEVICE CLASSES

Block devices (e.g. disk drives, CD)

- Commands include read, write, seek
- Can have raw access or via (e.g.) file system ("cooked") or memory-mapped

Character devices (e.g. keyboards, mice, serial):

- Commands include get, put
- Layer libraries on top for line editing, etc

Network Devices

- Vary enough from block and character devices to get their own interface
- Unix and Windows NT use the Berkeley Socket interface

Miscellaneous

- Current time, elapsed time, timers, clocks
- (Unix) ioctl covers other odd aspects of IO

20. Discuss about Interacting with the File Manager.

Interacting with the File Manager

• User Commands – OPEN, DELETE, RENAME, COPY

• Device independent – Physical location knowledge not needed

• Cylinder, surface, sector – Device medium knowledge not needed

• Tape, magnetic disk, optical disc, flash storage – Network knowledge not needed

Logical commands – Broken into lower-level signals – Example: READ

• Move read/write heads to record cylinder

• Wait for rotational delay (sector containing record passes under read/write head)

• Activate appropriate read/write head and read record

• Transfer record to main memory

• Send flag indicating free device for another request

• Performs error checking and correction – No need for error-checking code in programs

21. Discuss about File Access Mechanisms

File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files −

- Sequential access
- Direct/Random access
- Indexed sequential access

Sequential access

A sequential access is that in which the records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other. This access method is the most primitive one. Example: Compilers usually access files in this fashion.

Direct/Random access

- Random access file organization provides, accessing the records directly.
- Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.
- The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.

Indexed sequential access

- This mechanism is built up on base of sequential access.
- An index is created for each file which contains pointers to various blocks.
- Index is searched sequentially and its pointer is used to access the file directly.

TEN MARK QUESTIONS

1. Explain how operating system acts as a Resource Manager?

Modern computers consist of processors, memories, timers, disks, mice, network interfaces, printers, and a wide variety of other devices. In the alternative view, the job of the operating system is to provide for an orderly and controlled allocation of the processors, memories, and input/output devices among the various programs competing for them.

When a computer (or network) has multiple users, the need for managing and protecting the memory, input/output devices, and other resources is even greater, since the users might otherwise interface with one another. In addition, users often need to share not only hardware, but information (files, databases, etc.) as well. In short, this view of the operating system holds that its primary task is to keep track of which programs are using which resources, to grant resource requests, to account for usage, and to mediate conflicting requests from different programs and users.

Resource management includes **multiplexing** (sharing) resources in two different ways:

1. Time Multiplexing
2. Space Multiplexing

1. Time Multiplexing

When the resource is time multiplexed, different programs or users take turns using it. First one of them gets to use the resource, then another, and so on.

For example:

With only one CPU and multiple programs that want to run on it, operating system first allocates the CPU to one long enough, another one gets to use the CPU, then another and ten eventually the first one again.

Determining how the resource is time multiplexed – who goes next and for how long – is the task of the operating system.

2. Space Multiplexing

In space multiplexing, instead of the customers taking turns, each one gets part of the resource.

For example:

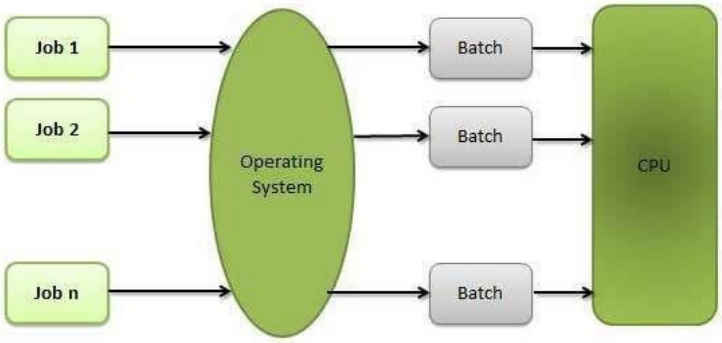
Main memory is normally divided up among several running programs, so each one can be resident at the same time (for example, in order to take turns using the CPU). Assuming there is enough memory to hold multiple programs, it is more efficient to hold several programs in memory at once rather than give one of them all of it, especially if it only needs a small fraction of the total. Of course, this raises issues of fairness, protection, and so on, and it is up to the operating system to solve them.

2. Explain about types of Operating System

Batch processing

Batch processing is a technique in which an Operating System collects the programs and data together in a batch before processing starts. An operating system does the following activities related to batch processing −

- The OS defines a job which has predefined sequence of commands, programs and data as a single unit.
- The OS keeps a number a jobs in memory and executes them without any manual information.
- Jobs are processed in the order of submission, i.e., first come first served fashion.
- When a job completes its execution, its memory is released and the output for the job gets copied into an output spool for later printing or processing.



Advantages

- Batch processing takes much of the work of the operator to the computer.
- Increased performance as a new job get started as soon as the previous job is finished, without any manual intervention.

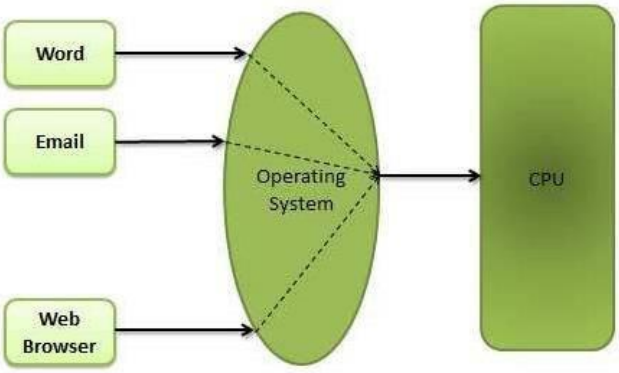
Disadvantages

- Difficult to debug program.
- A job could enter an infinite loop.
- Due to lack of protection scheme, one batch job can affect pending jobs.

Multitasking

Multitasking is when multiple jobs are executed by the CPU simultaneously by switching between them. Switches occur so frequently that the users may interact with each program while it is running. An OS does the following activities related to multitasking −

- The user gives instructions to the operating system or to a program directly, and receives an immediate response.
- The OS handles multitasking in the way that it can handle multiple operations/executes multiple programs at a time.
- Multitasking Operating Systems are also known as Time-sharing systems.
- These Operating Systems were developed to provide interactive use of a computer system at a reasonable cost.
- A time-shared operating system uses the concept of CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared CPU.
- Each user has at least one separate program in memory.

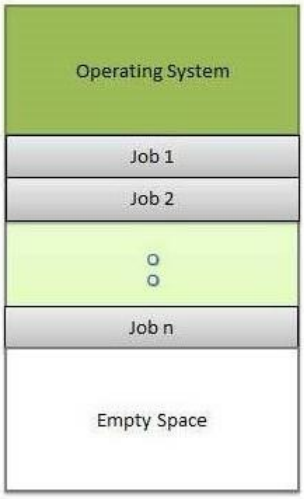


- A program that is loaded into memory and is executing is commonly referred to as a **process**.
- When a process executes, it typically executes for only a very short time before it either finishes or needs to perform I/O.
- Since interactive I/O typically runs at slower speeds, it may take a long time to complete. During this time, a CPU can be utilized by another process.
- The operating system allows the users to share the computer simultaneously. Since each action or command in a time-shared system tends to be short, only a little CPU time is needed for each user.
- As the system switches CPU rapidly from one user/program to the next, each user is given the impression that he/she has his/her own CPU, whereas actually one CPU is being shared among many users.

Multiprogramming

Sharing the processor, when two or more programs reside in memory at the same time, is referred as **multiprogramming**. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.

The following figure shows the memory layout for a multiprogramming system.



An OS does the following activities related to multiprogramming.

- The operating system keeps several jobs in memory at a time.
- This set of jobs is a subset of the jobs kept in the job pool.
- The operating system picks and begins to execute one of the jobs in the memory.
- Multiprogramming operating systems monitor the state of all active programs and system resources using memory management programs to ensures that the CPU is never idle, unless there are no jobs to process.

Advantages

- High and efficient CPU utilization.
- User feels that many programs are allotted CPU almost simultaneously.

Disadvantages

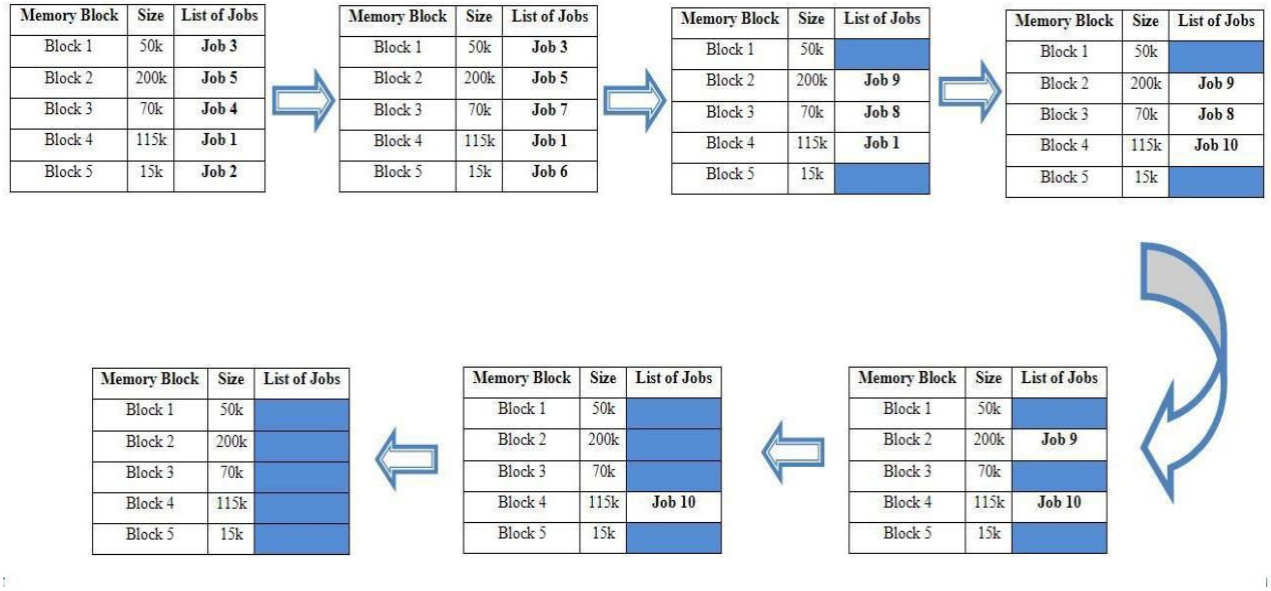
- CPU scheduling is required.
- To accommodate many jobs in memory, memory management is required.

3. Explain in detail about First fit and Best fit memory allocation

Best fit memory allocation:

Best-fit memory allocation makes the best use of memory space but slower in making allocation. In the illustration below, on the first processing cycle, jobs 1

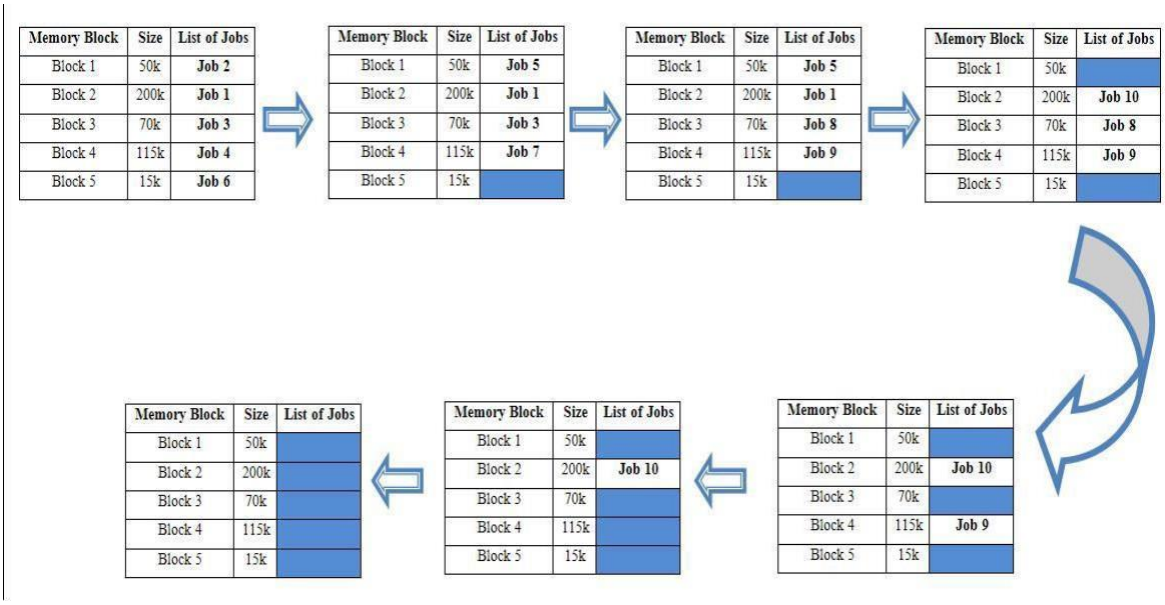
to 5 are submitted and be processed first. After the first cycle, job 2 and 4 located on block 5 and block 3 respectively and both having one turnaround are replace by job 6 and 7 while job 1, job 3 and job 5 remain on their designated block. In the third cycle, job 1 remain on block 4, while job 8 and job 9 replace job 7 and job 5 respectively (both having 2 turnaround). On the next cycle, job 9 and job 8 remain on their block while job 10 replace job 1 (having 3 turnaround). On the fifth cycle only job 9 and 10 are the remaining jobs to be process and there are 3 free memory blocks for the incoming jobs. But since there are only 10 jobs, so it will remain free. On the sixth cycle, job 10 is the only remaining job to be process and finally on the seventh cycle, all jobs are successfully process and executed and all the memory blocks are now free.



First fit memory allocation:

First-fit memory allocation is faster in making allocation but leads to memory waste. The illustration below shows that on the first cycle, job 1 to job 4 are submitted first while job 6 occupied block 5 because the remaining memory space is enough to its required memory size to be process. While job 5 is in waiting queue because the memory size in block 5 is not enough for the job 5 to be process. Then on the next cycle, job 5 replace job 2 on block 1 and job 7 replace job 4 on block 4 after both job 2 and job 4 finish their process. Job 8 is in waiting queue because the remaining block is not enough to accommodate the memory size of job 8. On the third cycle, job 8 replace job 3 and job 9 occupies block 4 after processing job 7. While Job 1 and job 5 remain on its designated block. After the third cycle block 1 and block 5 are free to serve the incoming jobs but since there

are 10 jobs so it will remain free. And job 10 occupies block 2 after job 1 finish its turns. On the other hand, job 8 and job 9 remain on their block. Then on the fifth cycle, only job 9 and job 10 are to be process while there are 3 memory blocks free. In the sixth cycle, job 10 is the only remaining job to be process and lastly in the seventh cycle, all jobs are successfully process and executed and all the memory blocks are now free.



4. Briefly explain about Relocatable Dynamic Partitions.

In relocatable dynamic partition, memory manager in operating system relocates the program, that is all empty blocks are gathered to form one single block of large memory enough to accommodate some or all of the jobs waiting in the queue.

Consider the following example:

Job	Memory utilized	Turnaround time
Job1	100K	3
Job2	10K	1
Job3	30K	2
Job4	20K	1
Job5	25K	2

Job6	5K	1
Job7	25K	1
Job8	50K	2
Job9	90K	3
Job10	100K	3

Let us assume the memory size is 220k with 10k allocated to operating system programs. The above table gives the data of number of jobs, memory utilized by individual job and their respective turnaround time. There are 10 jobs to be loaded into memory. Initially with 220k of memory, we can load job1 to job6 with free memory of 20k.

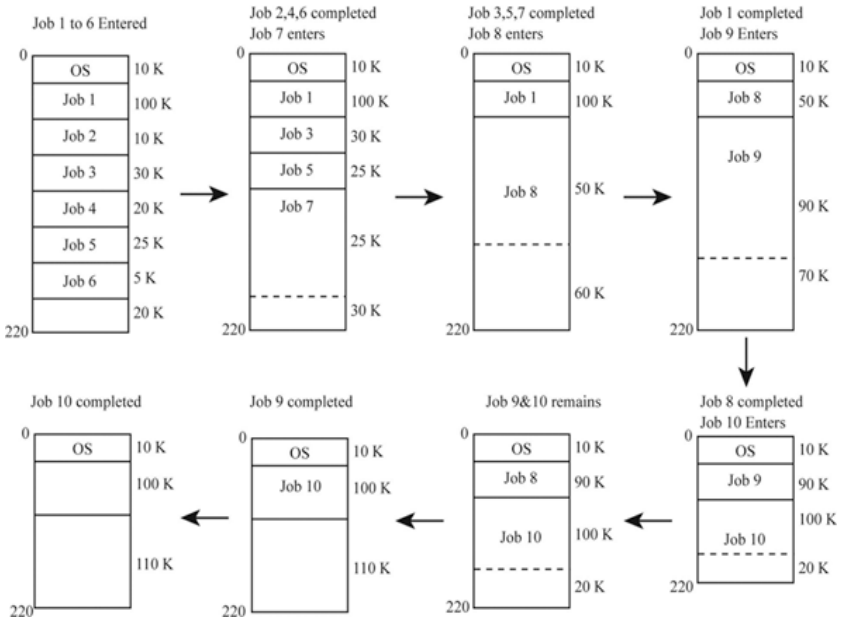
At first turnaround time, job2 of 10k, job4 of 20k, and job6 of 5k are completed leaving job 1, job 3, and job 5 as it is with free memory of 55k, in other words job1, job3, and job5 are still processing. Job 7 of 25k now enters into the memory partition leaving only 30k free memory where neither of the remaining jobs (job 8, job 9, and job 10) can accommodate.

At second turnaround time job3 of 30k, job5 of 25k, and job7 of 25k are completed leaving job 1 of 100k with free memory of 110k. Now job 8 of 50K is entered where 50k is allocated leaving 60K of free memory space where either job 9 or job 10 can accommodate.

Note: for job3 and job5 turnaround time is finished and at the same time turnaround time 1 of job 7 is also finished.

At third turnaround time, job1 of 100k is completed leaving free memory space of 160k. Job9 now can be allocated with 90k leaving remaining 70K of free memory where job10 cannot accommodate. Once job 8 finishes turnaround time of 2, it is completed leaving 120k of free memory space. Now job10 can be allocated with 100k of memory leaving 20K of free memory space. Process repeats until all the memory locations are freed leaving 10k allocated to operating system processes.

Diagrammatic representation of above example is shown below:



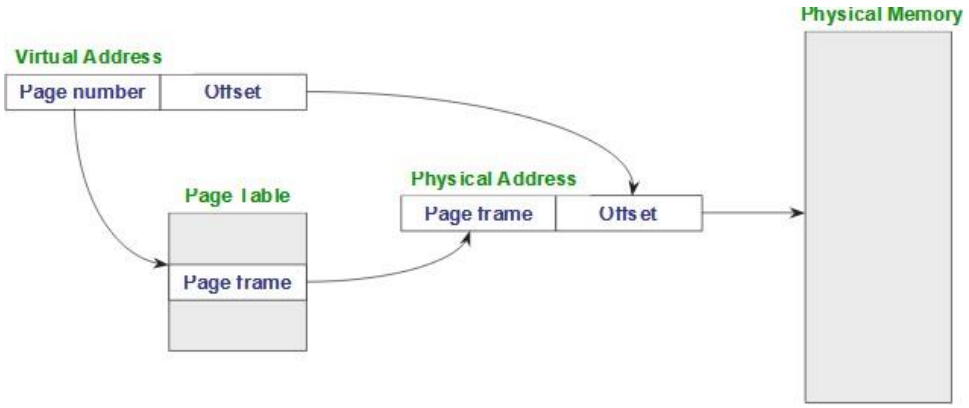
5. Explain in detail about Paged memory management.

Translating addresses

- Virtual address has two parts: virtual page number and offset
- Virtual page number (VPN) is an index into a page table
- Page table determines page frame number (PFN)
- Physical address is PFN::offset

Page tables

- Map virtual page number (VPN) to page frame number (PFN)
  - »VPN is the index into the table that determines PFN
- One page table entry (PTE) per page in virtual address space
  - »Or, one PTE per VPN



Paging Examples

Pages are 4K

- VPN is 20 bits (220 VPNs), offset is 12 bits

Virtual address is 0x7468

- Virtual page is 0x7, offset is 0x468

Page table entry 0x7 contains 0x2

- Page frame base is 0x2 * 0x1000 (4K) = 0x2000
- Seventh virtual page is at address 0x2000 (3rd physical

page) Physical address = 0x2000 + 0x468 = 0x2468

Page Table Entries (PTEs)

Page table entries control mapping

- The Modify bit says whether or not the page has been written
  - »It is set when a write to the page occurs
- The Reference bit says whether the page has been accessed
  - »It is set when a read or write to the page occurs
- The Valid bit says whether or not the PTE can be used
  - »It is checked each time the virtual address is used
- The Protection bits say what operations are allowed on page
  - »Read, write, execute

◆The page frame number (PFN) determines physical page.

Paging Advantage

Easy to allocate memory
◆Memory comes from a free list of fixed size chunks
◆Allocating a page is just removing it from the list
◆External fragmentation not a problem

Easy to swap out chunks of a program
◆All chunks are the same size
◆Use valid bit to detect references to swapped pages
◆Pages are a convenient multiple of the disk block size

Paging Limitations

Can still have internal fragmentation
◆Process may not use memory in multiples of a page

Memory reference overhead
◆2 references per address lookup (page table, then memory)
◆Solution – use a hardware cache of lookups (more later)

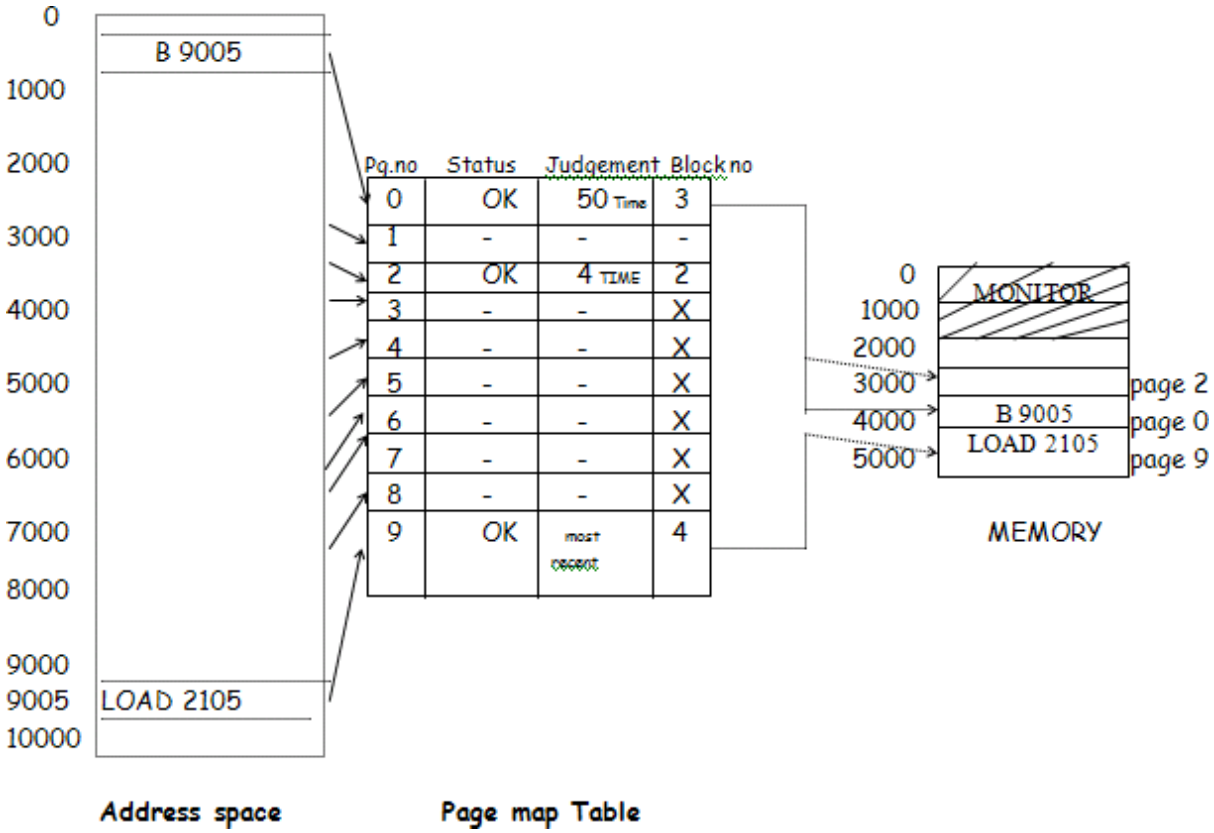
Memory required to hold page table can be significant
◆Need one PTE per page
◆32 bit address space w/ 4KB pages = 2^{20} PTEs
◆4 bytes/PTE = 4MB/page table
◆25 processes = 100MB just for page tables!
◆Solution – page the page tables.

6. Explain in detail about Demand Paged Memory Allocation

Demand Paged Memory Allocation : In the demand paged memory allocation one programmer assumes infinite memory called virtual memory. In which the Job operates under demand.

Paging virtual memory is large in size then the available memory the demand paging allocation is shown in figure.

In demand pages memory allocation a particular page is swapped from the secondary storage device if it is required for execution.



In the above fig. Job address space consists of 10,000B of programs called page and we have only 3000B of memory available. This 10,000B program run in 3000B of memory by using demand paging.

The page map table consists of a page number, page frame number, status and judgement field. Initially the allocation algorithm places the first three pages of a program in the main memory. The job starts execution in one of these pages. Suppose in page O there is a transfer instruction to the location 9055. The hard ware mechanism uses 9 as index to the page lable that is page 9 is required in the execution. then the pagemap table is checked to find whether the page in the main memory job not if the required page is not in the main memory then the page fault occurs, then the hard ware generates interrept to the operating system then the operating system searches the secondary storage device for page 9 and brought

into the memory by replacing one of the pages in the memory It may use a page remove algorithms like FIFO(First in first out) LRU(Least Recently Use) for removing a page. The decision which page has to replace it with judgement field in the page map table. It is desirable to keep highly used pages in the memory.

Advantages :

1.Fragmentation is eliminated.
2.Large Virtual memory is available.
3.More efficient use of memory.
4.Demand paging is valuable in time sharing, systems.

Dis-Advantages :

1. Page address mapping hard ware increases the cost of the computer system
2. Extra memory, extra register needed for page map table.

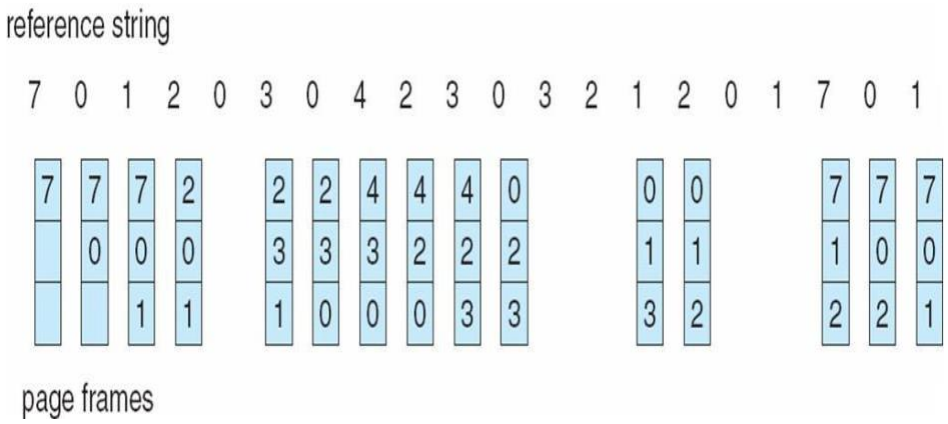
Page Replacement Algorithms

- Want lowest page-fault rate.
- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults and page replacements on that string.
- In all our examples, we use a few recurring reference strings.

The FIFO Policy

- Treats page frames allocated to a process as a circular buffer:
  - When the buffer is full, the oldest page is replaced. Hence first-in, first-out:
    - A frequently used page is often the oldest, so it will be repeatedly paged out by FIFO.
  - Simple to implement:
    - requires only a pointer that circles through the page frames of the process.

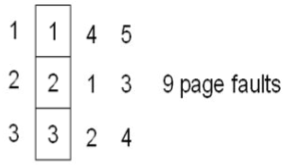
FIFO Page Replacement



First-In-First-Out (FIFO) Algorithm

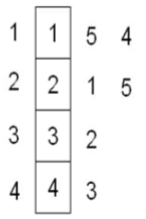
- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory at a time per process):



9 page faults

- 4 frames:



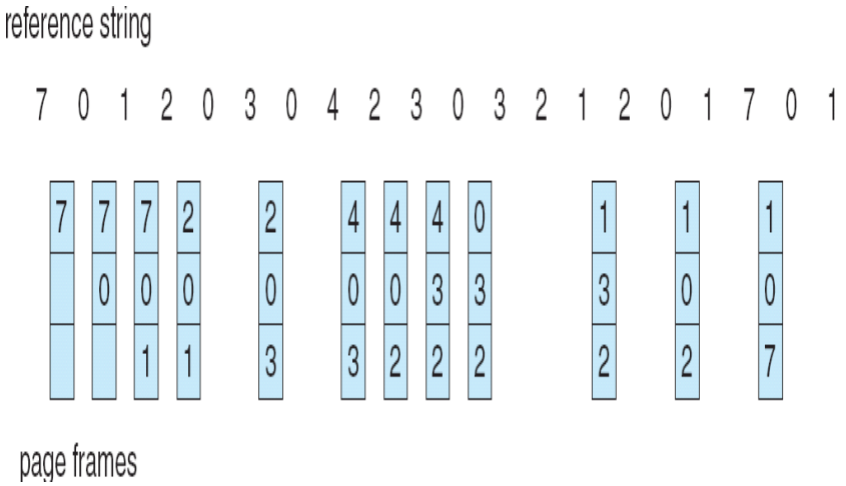
10 page faults

- FIFO Replacement manifests Belady's Anomaly:
  - more frames ⇒ more page faults

The LRU Policy

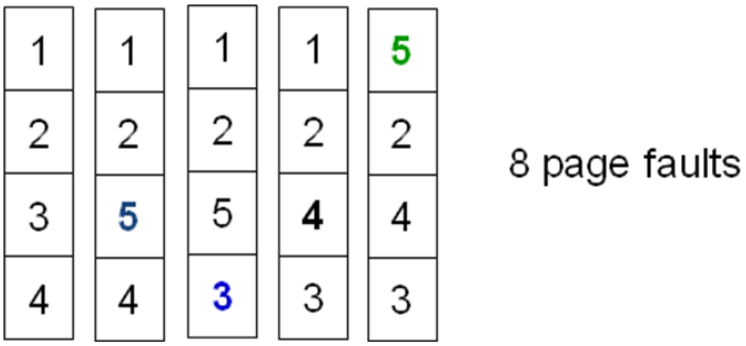
- Replaces the page that has not been referenced for the longest time:
  - By the principle of locality, this should be the page least likely to be referenced in the near future.
  - performs nearly as well as the optimal policy.

LRU Page Replacement



Least Recently Used (LRU) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, **5**, 1, 2, **3**, **4**, **5**



7. Explain about Interrupts.

Definition : an event external to the currently executing process that causes a change in the normal flow of instruction execution; usually generated by hardware devices external to the CPU

• From " Design and Implementation of the FreeBSD Operating System" , Glossary

• Key point is that interrupts are asynchronous w.r.t. current process

• Typically indicate that some device needs service

Why Interrupts?

People like connecting devices

- A computer is much more than the CPU
- Keyboard, mouse, screen, disk drives
- Scanner, printer, sound card, camera, etc.

These devices occasionally need CPU service

• But we can't predict when

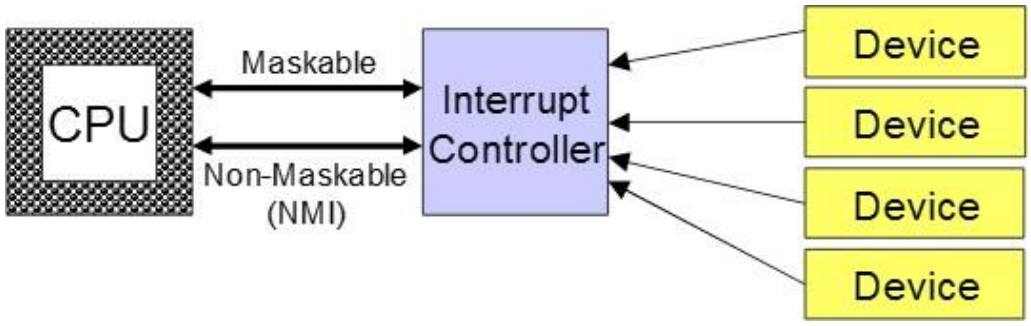
External events typically occur on a macroscopic timescale

• we want to keep the CPU busy between events

Give each device a wire (interrupt line) that it can use to signal the processor

• When interrupt signaled, processor executes a routine called an interrupt handler to deal with the interrupt

• No overhead when no requests pending



Polling vs. Interrupts

" Polling is like picking up your phone every few seconds to see if you have a call. Interrupts are like waiting for the phone to ring. "

• Interrupts win if processor has other work to do and event response time is not critical

• Polling can be better if processor has to respond to an event ASAP

• May be used in device controller that contains dedicated secondary processor

Hardware Interrupt Handling

Details are architecture dependent!

• Interrupt controller signals CPU that interrupt has occurred, passes interrupt number
• Interrupts are assigned priorities to handle simultaneous interrupts
• Lower priority interrupts may be disabled during service
• CPU senses (checks) interrupt request line after every instruction; if raised, then:
• uses interrupt number to determine which handler to start
• interrupt vector associates handlers with interrupts
• Basic program state saved (as for system call)
• CPU jumps to interrupt handler
• When interrupt done, program state reloaded and program resumes

Software Interrupt Handling

Typically two parts to interrupt handling
• The part that has to be done immediately
• So that device can continue working
• The part that should be deferred for later
• So that we can respond to the device faster
• So that we have a more convenient execution context

8. What are the common Strategies for handling deadlocks?

- Deadlock prevention. Prevents deadlocks by restraining requests made to ensure that at least one of the four deadlock conditions cannot occur.
- Deadlock avoidance. Dynamically grants a resource to a process if the resulting state is safe. A state is safe if there is at least one execution sequence that allows all processes to run to completion.
- Deadlock detection and recovery. Allows deadlocks to form; then finds and breaks them.

Two types of deadlocks

- Resource deadlock: uses AND condition.

AND condition: a process that requires resources for execution can proceed when it has acquired all those resources.

- Communication deadlock: uses OR condition.

OR condition: a process that requires resources for execution can proceed when it has acquired at least one of those resources.

- P-out-of –Q condition which means that a process simultaneously requests Q resources and remains blocked until it is granted any P of those resources.
- AND-OR model, which may specify any combination of AND and OR models.

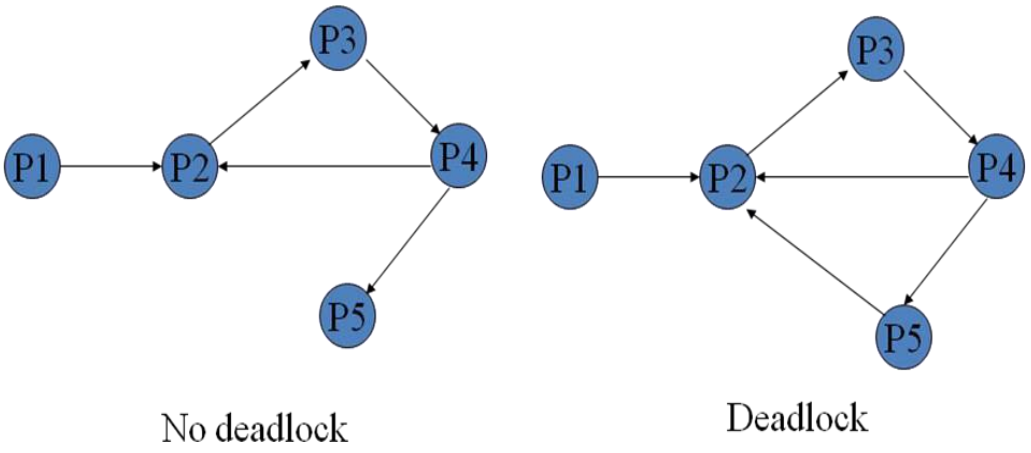
E.g. a AND (b OR c).

Deadlock conditions

- The condition for deadlock in a system using the AND condition is the existence of a *cycle*.
- The condition for deadlock in a system using the OR condition is the existence of a *knot*.

A knot (K) consists of a set of nodes such that for every node a in K, all nodes in K and only the nodes in K are reachable from node a.

Example: OR condition



No deadlock Deadlock

Deadlock Prevention

- 1. A process acquires all the needed resources simultaneously before it begins its execution, therefore breaking the hold and wait condition.
- E.g. In the dining philosophers' problem, each philosopher is required to pick up both forks at the same time. If he fails, he has to release the fork(s) (if any) he has acquired.

- Drawback: over-cautious.
- 2. All resources are assigned unique numbers. A process may request a resource with a unique number I only if it is not holding a resource with a number less than or equal to I and therefore breaking the circular wait condition.
- E.g. In the dining philosophers problem, each philosopher is required to pick a fork that has a larger id than the one he currently holds. That is, philosopher P5 needs to pick up fork F5 and then F1; the other philosopher Pi should pick up fork Fi followed by Fi-1.
- Drawback: over-cautions.
- 3. Each process is assigned a unique priority number. The priority numbers decide whether process Pi should wait for process Pj and therefore break the non-preemption condition.
- E.g. Assume that the philosophers' priorities are based on their ids, i.e., Pi has a higher priority than Pj if i <j. In this case Pi is allowed to wait for Pi+1 for I=1,2,3,4. P5 is not allowed to wait for P1. If this case happens, P5 has to abort by releasing its acquired fork(s) (if any).
- Drawback: starvation. The lower priority one may always be rolled back. Solution is to raise the priority every time it is victimized.
- 4. Practically it is impossible to provide a method to break the mutual exclusion condition since most resources are intrinsically non-sharable, e.g., two philosophers cannot use the same fork at the same time.

A Deadlock Prevention Example

Wait-die

- Wants Resource Hold Resource
- Old process -----→ Young process
- 10 20
- Waits
- Wants resource Holds resource
- Young process 20 Old process 10
- Dies
- Wait-die is a non-preemptive method.

- Wound-wait
- Wants resource Hold resource

- Old process 10 Young process 20
- Preempts

- Wants resource Hold resource
- Young process 20 Old process 10

- Waits

An example

Process id	priority	1st request time	length	Retry interval
P1	2	1	1	1
P2	1	1.5	2	1
P3	4	2.1	2	2
P4	5	3.3	1	1
P5	3	4.0	2	3

Deadlock Avoidance

Four resources ABCD. A has 6 instances, B has 3 instances, C

Has 4 instances and D has 2 instances.

Process	Allocation	Max
	ABCD	ABCD
P1	3011	4111
P2	0100	0212
P3	1110	4210
P4	1101	1101

P5 0000 2110

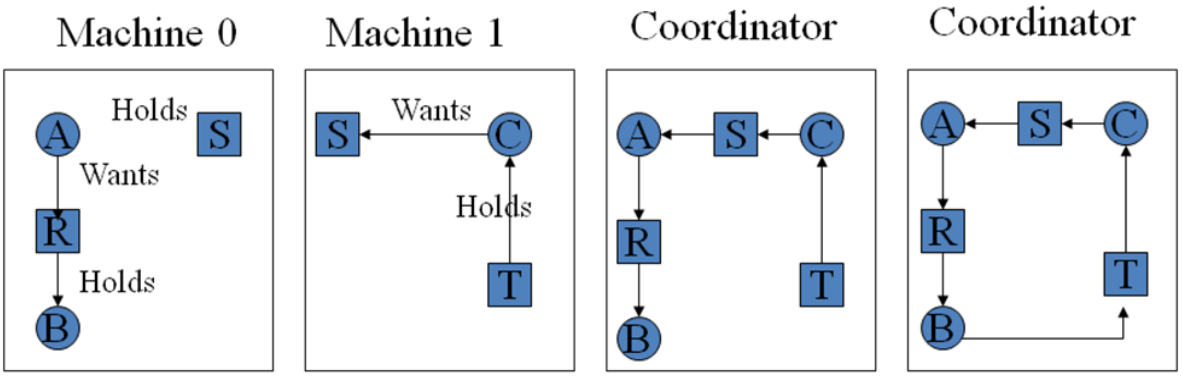
Is the current state safe?

If P5 requests for (1,0,1,0), can this be granted?

Deadlock Detection and Recovery

- Centralized approaches
- Distributed approaches
- Hierarchical approaches

Centralized approaches

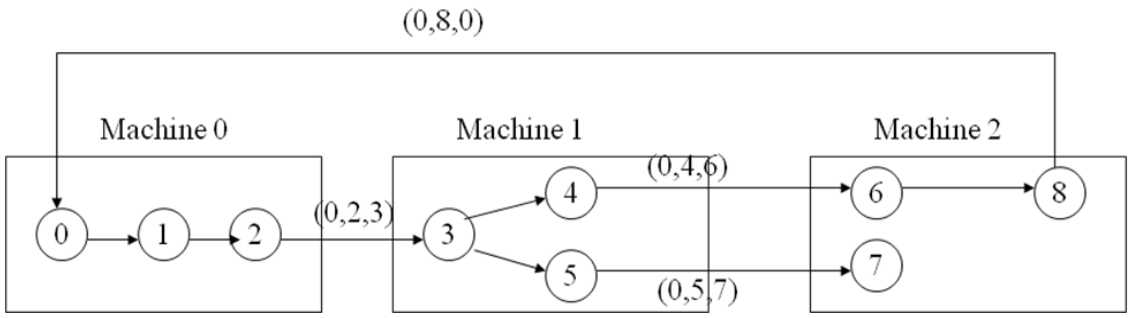


B releases R and then B wants T.
But B wants T reaches coordinator first
and results in false deadlock.

Distributed approaches

- A copy of the global wait-for graph is kept at each site with the result that each site has a global view of the system.
- The global wait-for graph is divided and distributed to different sites.

Chandy-Misra-Haas distributed deadlock detection algorithm



Hierarchical approaches

- In hierarchical deadlock detection algorithms, sites are arranged hierarchically in a tree. A site detects deadlocks involving only its descendant sites.
- For example, let A, B and C be controllers such that C is the lowest common ancestor of A and B. Suppose that node Pi appears in the local wait-for graph of controllers A and B. Then Pi must also appear in the local wait-for graph as:
-
- · Controller of C.
- · Every controller in the path from C to A.
- · Every controller in the path from C to B.
-
- In addition, if Pi and Pj appear in the wait-for graph of controller D and there exists a path from Pi to Pj in the wait-for graph of one of the children of D, then an edge (Pi, Pj) must be in the wait-for graph of D.

9. Explain about Multi-Core Processors.

History

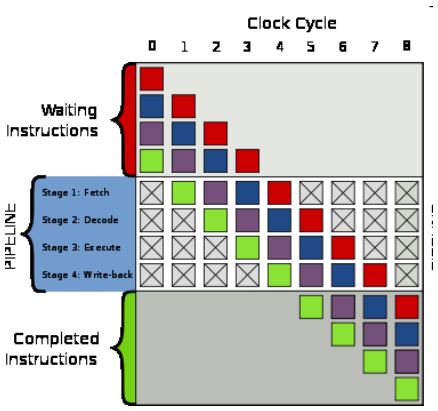
- In the early 1970's the first Microprocessor was developed by Intel.
- It was a 4 bit machine that was named the 4004
- The 4004 was followed by Intel's 8008 and 8080, as well as motorola's 6800 and 68000

With each new generation of processors there were several developments such as:

- Smaller size
- Faster
- Increased heat dissipation
- Greater Consumption of power

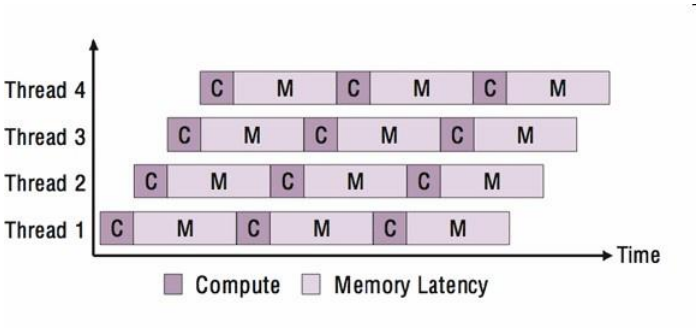
On technique used to increase single core performance was:

- Pipelining: beginning other waiting instructions before the first finishes



Another technique was multithreading

- Multithreading involves execution of two separate threads.
- Time is divided and interlaced between the two threads in order to simulate simultaneous execution



Problems with Single Core

To execute the tasks faster you must increase the clock time.

Increasing clock times too high drastically increases power consumption and heat dissipation to extremely high levels, making the processor inefficient.

Multi Core solution

Creating two cores or more on the same Die increases processing power while keeping clock speeds at an efficient level.

A processor with 2 cores running at efficient clock speeds can process instructions with similar speed to a single core processor running at twice the clock speed, yet the dual core processor would still consume less energy.

Multi-Core Advantages

- While working with many threads, a Multi Core processor with n cores can execute n threads simultaneously by assigning a core to each thread. If it must process more than n threads , say x, it can apply multithreading procedures with each core working with an average of x/n threads.
- A Single core processor must multithread with every single thread.

Improving an existing single core design or creating a better one is problematic in that it:

- Is time consuming to design
- Is risky to modify existing designs

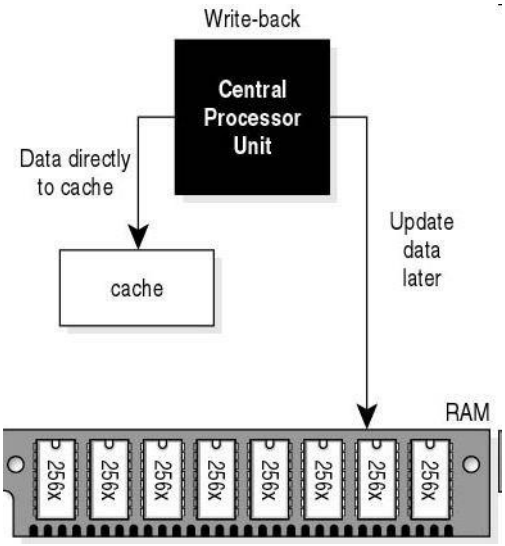
Creating multi core processors is convenient in that:

- The name “core dual” and similar names are good for marketing.
- It has lower manufacturing costs.
- Uses proven processor designs.

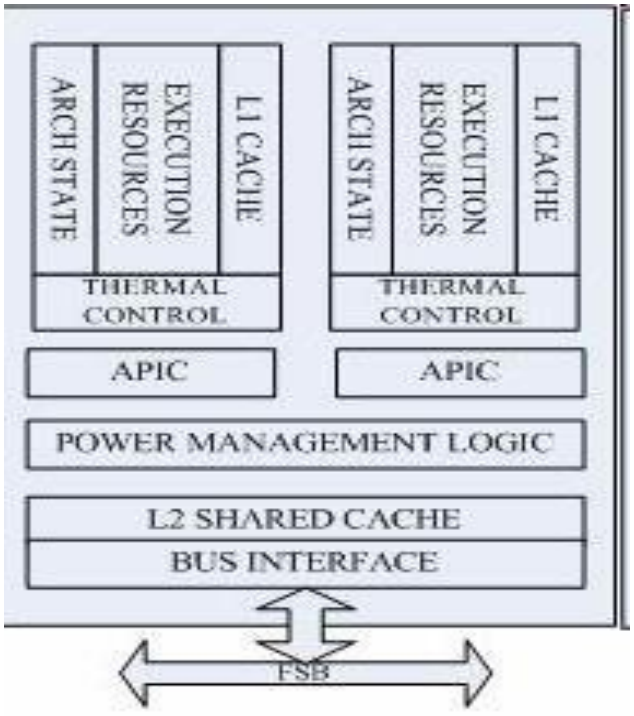
Implementations

- Two main ways to have multiple cores interact are the shared memory model, and the distributed memory model.
- In the shared memory model, all cores share the same cache memory.

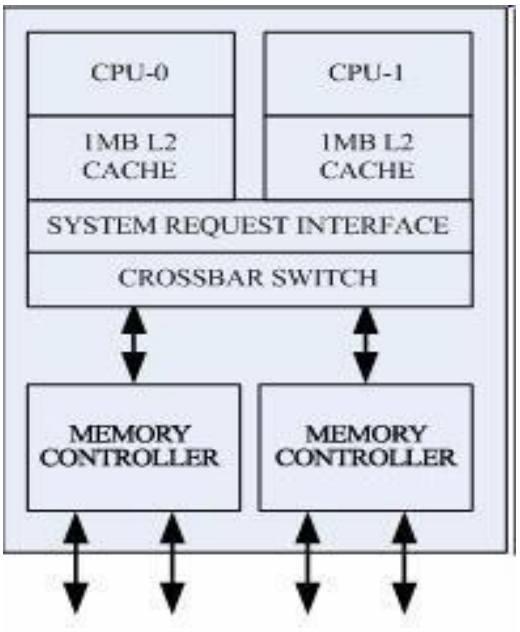
- In the distributed memory model, each core has its own cache memory.



- The Intel core duo design has a separate L1 cache memory for each core, but both cores share an L2 cache.



- The AMD Athlon 64 X2 implementation has separate L1 and L2 cache memory for each core.



Some of the current problems found with multi core processors include:

- Memory/Cache coherence. As mentioned earlier, some implementations have distributed L1 caches but must share an L2 cache. This poses the problem of making sure each core keeps the other updated with changes in the data in its own cache.
- Multi threading is also a problem when the software being run is not designed to take advantage of the multi core processor. This may mean that one core does most of the work which means that the processor is running no more efficiently than a single core.

10. Process Synchronization Software

- Success of **process synchronization** hinges on capability of OS to make a resource unavailable to other processes while it's being used by one of them.
  - E.g., I/O devices, a location in storage, or a data file.
  - In essence, used resource must be locked away from other processes until it is released.
- Only when it is released is a waiting process allowed to use resource. A mistake could leave a job waiting indefinitely.

Synchronization Mechanisms

- Common element in all synchronization schemes is to allow a process to finish work on a **critical region** of program before other processes have access to it.
  - Applicable both to multiprocessors and to 2+ processes in a single-processor (time-shared) processing system.
  - Called a critical region because its execution must be handled as a unit.

Lock-and-Key Synchronization

- Process first checks if key is available
- If it is available, process must pick it up and put it in lock to make it unavailable to all other processes.
- For this scheme to work both actions must be performed in a single machine cycle.
- Several locking mechanisms have been developed including test-and-set, WAIT and SIGNAL, and semaphores.

Test-And-Set (TS) Locking

- **Test-and-set** is a single indivisible machine instruction (TS).
- In a single machine cycle it tests to see if key is available and, if it is, sets it to "unavailable."
- Actual key is a single bit in a storage location that can contain a zero (if it's free) or a one (if busy).
- Simple procedure to implement.
- Works well for a small number of processes.

Problems with Test-And-Set

- When many processes are waiting to enter a critical region, **starvation** could occur because processes gain access in an arbitrary fashion.
  - Unless a first-come first-served policy were set up, some processes could be favored over others.
  - Waiting processes remain in unproductive, resource-consuming wait loops (**busy waiting).**
  - Consumes valuable processor time.
  - Relies on the competing processes to test key.

WAIT and SIGNAL Locking

- Modification of test-and-set.
- Adds 2 new operations, which are mutually exclusive and become part of the process scheduler's set of operations
  - WAIT
  - SIGNAL
  - Operations WAIT and SIGNAL frees processes from "busy waiting" dilemma and returns control to OS which can then run other jobs while waiting processes are idle.

WAIT

- Activated when process encounters a busy condition code.
- Sets process control block (PCB) to the blocked state
- Links it to the queue of processes waiting to enter this particular critical region.
- Process Scheduler then selects another process for execution.

SIGNAL

- Activated when a process exits the critical region and the condition code is set to "free."
- Checks queue of processes waiting to enter this critical region and selects one, setting it to the READY state.
- Process Scheduler selects this process for running.

Semaphores

- **Semaphore** -- nonnegative integer variable used as a flag.
- Signals if & when a resource is free & can be used by a process.
- Most well-known semaphores are signaling devices used by railroads to indicate if a section of track is clear.
- Dijkstra (1965) -- 2 operations to operate semaphore to overcome the process synchronization problem.
  - P stands for the Dutch word *proberen* (to test)
  - V stands for *verhogen* (to increment)

P (Test) and V (Increment)

- If we let *s* be a semaphore variable, then the V operation on *s* is simply to increment *s* by 1.

V(*s*): *s*: = *s* + 1

- Operation P on *s* is to test value of *s* and, if it's not zero, to decrement it by one.

P(*s*): If *s* > 0 then *s*: = *s* – 1

- P and V are executed by OS in response to calls issued by any one process naming a semaphore as parameter.

MUTual EXclusion (Mutex)

- P and V operations on semaphore *s* enforce concept of mutual exclusion, which is necessary to avoid having 2 operations attempt to execute at same time.
- Called **mutex** (MUTual EXclusion)

P(mutex): if mutex > 0 then mutex: = mutex – 1

V(mutex): mutex: = mutex + 1

11. Explain in detail about Communication among Devices.

The CPU must have a way to pass information to and from an I/O device. There are three approaches available to communicate with the CPU and Device.

- Special Instruction I/O
- Memory-mapped I/O
- Direct memory access (DMA)

Special Instruction I/O

This uses CPU instructions that are specifically made for controlling I/O devices. These instructions typically allow data to be sent to an I/O device or read from an I/O device.

Memory-mapped I/O

When using memory-mapped I/O, the same address space is shared by memory and I/O devices. The device is connected directly to certain main memory locations so that I/O device can transfer block of data to/from memory without going through CPU.

While using memory mapped IO, OS allocates buffer in memory and informs I/O device to use that buffer to send data to the CPU. I/O device operates asynchronously with CPU, interrupts CPU when finished.

The advantage to this method is that every instruction which can access memory can be used to manipulate an I/O device. Memory mapped IO is used for most high-speed I/O devices like disks, communication interfaces.

Direct Memory Access (DMA)

Slow devices like keyboards will generate an interrupt to the main CPU after each byte is transferred. If a fast device such as a disk generated an interrupt for each byte, the operating system would spend most of its time handling these interrupts. So a typical computer uses direct memory access (DMA) hardware to reduce this overhead.

Direct Memory Access (DMA) means CPU grants I/O module authority to read from or write to memory without involvement. DMA module itself controls exchange of data between main memory and the I/O device. CPU is only involved at the beginning and end of the transfer and interrupted only after entire block has been transferred.

Direct Memory Access needs a special hardware called DMA controller (DMAC) that manages the data transfers and arbitrates access to the system bus. The controllers are programmed with source and destination pointers (where to read/write the data), counters to track the number of transferred bytes, and settings, which includes I/O and memory types, interrupts and states for the CPU cycles.

The operating system uses the DMA hardware as follows −

Step	Description
1	Device driver is instructed to transfer disk data to a buffer address X.
2	Device driver then instruct disk controller to transfer data to buffer.

3 Disk controller starts DMA transfer.

4 Disk controller sends each byte to DMA controller.

5 DMA controller transfers bytes to buffer, increases the memory address, decreases the counter C until C becomes zero.

6 When C becomes zero, DMA interrupts CPU to signal transfer completion.

12. Explain about File Organization.

- File organization refers to the way data is stored in a file. File organization is very important because it determines the methods of access, efficiency, flexibility and storage devices to use. There are four methods of organizing files on a storage media. This include:

- sequential,
- random,
- serial and
- indexed-sequential

1. Sequential file organization

- Records are stored and accessed in a particular order sorted using a key field.
- Retrieval requires searching sequentially through the entire file record by record to the end.
- Because the record in a file are sorted in a particular order, better file searching methods like the binary search technique can be used to reduce the time used for searching a file .
- Since the records are sorted, it is possible to know in which half of the file a particular record being searched is located, Hence this method repeatedly divides the set of records in the file into two halves and searches only the half on which the records is found.
- For example, of the file has records with key fields 20, 30, 40, 50, 60 and the computer is searching for a record with key field 50, it starts at 40 upwards in its search, ignoring the first half of the set.

Advantages of sequential file organization

- The sorting makes it easy to access records.
- The binary chop technique can be used to reduce record search time by as much as half the time taken.

Disadvantages of sequential file organization

- The sorting does not remove the need to access other records as the search looks for particular records.
- Sequential records cannot support modern technologies that require fast access to stored records.
- The requirement that all records be of the same size is sometimes difficult to enforce.

2. Random or direct file organization

- Records are stored randomly but accessed directly.
- To access a file stored randomly, a record key is used to determine where a record is stored on the storage media.
- **Magnetic** and **optical** disks allow data to be stored and accessed randomly.

Advantages of random file access

- Quick retrieval of records.
- The records can be of different sizes.

3. Serial file organization

- Records in a file are stored and accessed one after another.
- The records are not stored in any way on the storage medium this type of organization is mainly used on **magnetic tapes**.

Advantages of serial file organization

- It is simple
- It is cheap

Disadvantages of serial file organization

- It is cumbersome to access because you have to access all proceeding records before retrieving the one being searched.
- Wastage of space on medium in form of inter-record gap.
- It cannot support modern high speed requirements for quick record access.

4. Indexed-sequential file organization method

- Almost similar to sequential method only that, an index is used to enable the computer to locate individual records on the storage media. For example, on a **magnetic drum**, records are stored sequential on the tracks. However, each record is assigned an index that can be used to access it directly.