Memory Management

- Memory access and memory management are a very important part of modern computer operation.
- Every instruction has to be fetched from memory before it can be executed, and most instructions involve retrieving data from memory or storing data.
- The advent of multi-tasking, operating system increases the complexity of memory management.
- More complicated in case of shared memory, virtual memory etc.,
- CPU utilisation and performance can be greatly improved by sharing memory among several processes.

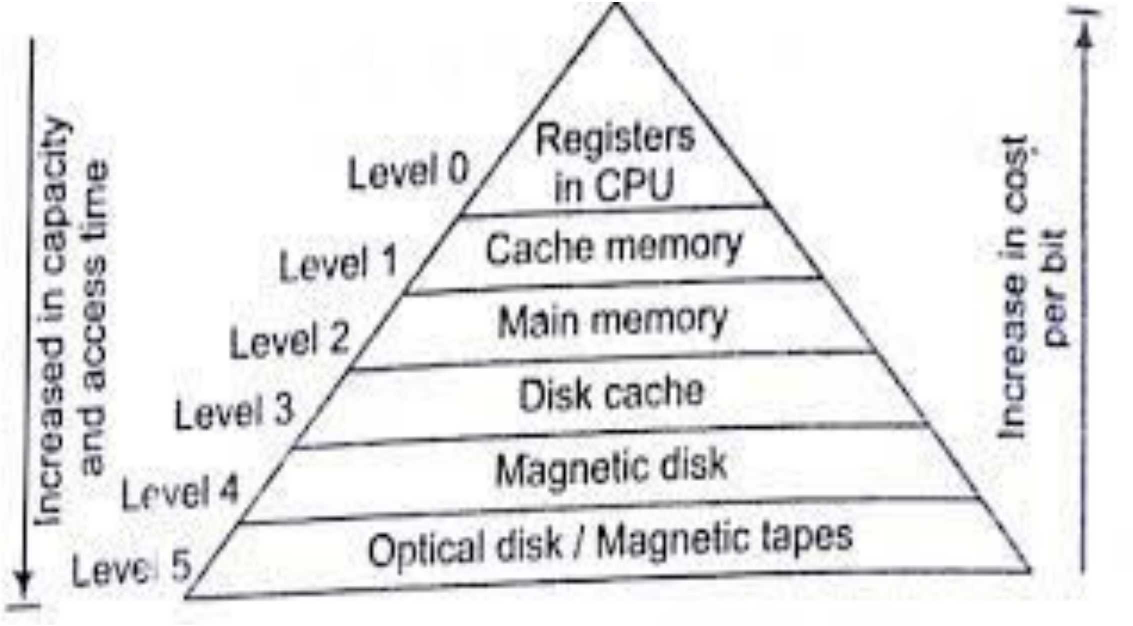
Main Memory

- Main memory refers to physical memory that is the internal memory of the computer
- Main memory is also known as RAM.
- Main memory is the place where programs and information are kept when the processor is in execution.
- Main memory is associated with the processor, so moving the instructions and information into and out of the processor is extremely fast.

Functions

- It keeps track of every memory location.
- Tracks to check whether memory is allocated or not.
- Track of how many memory is allocated
- It takes the decision of which process will get memory and when.
- Updates the status of memory location when it is allocated or freed.
- Protection - It is implemented with the help of fence address.
- It any process is entered it needs to access the memory location then it will check with the fence address.
- Process can’t access OS and other process location.

Memory Hierarchy

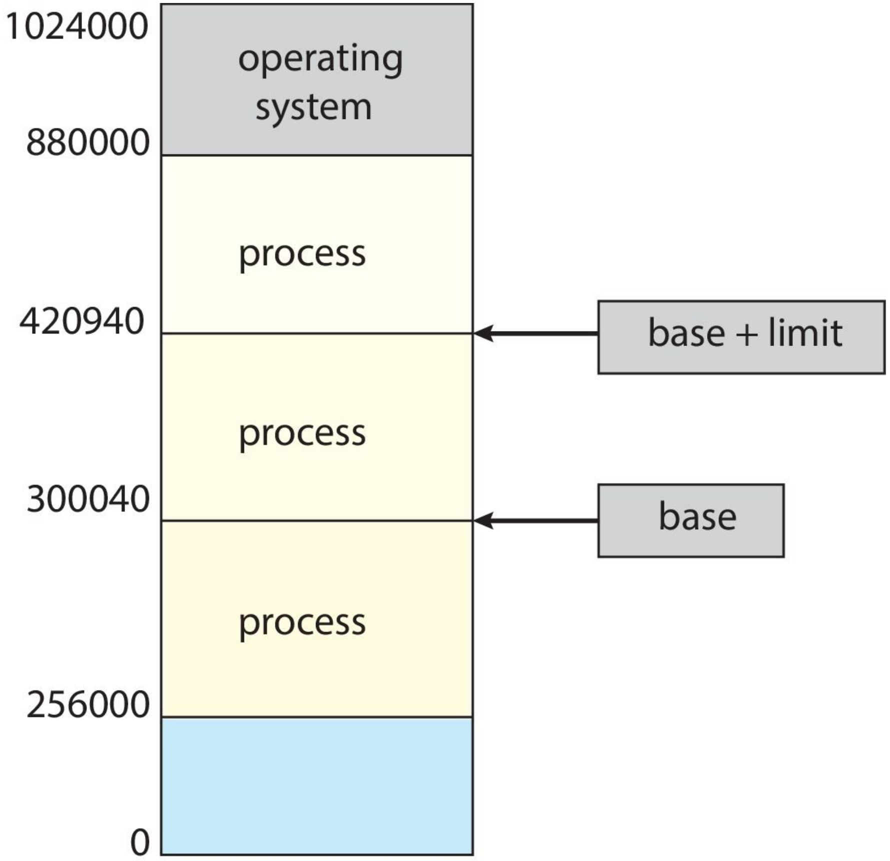


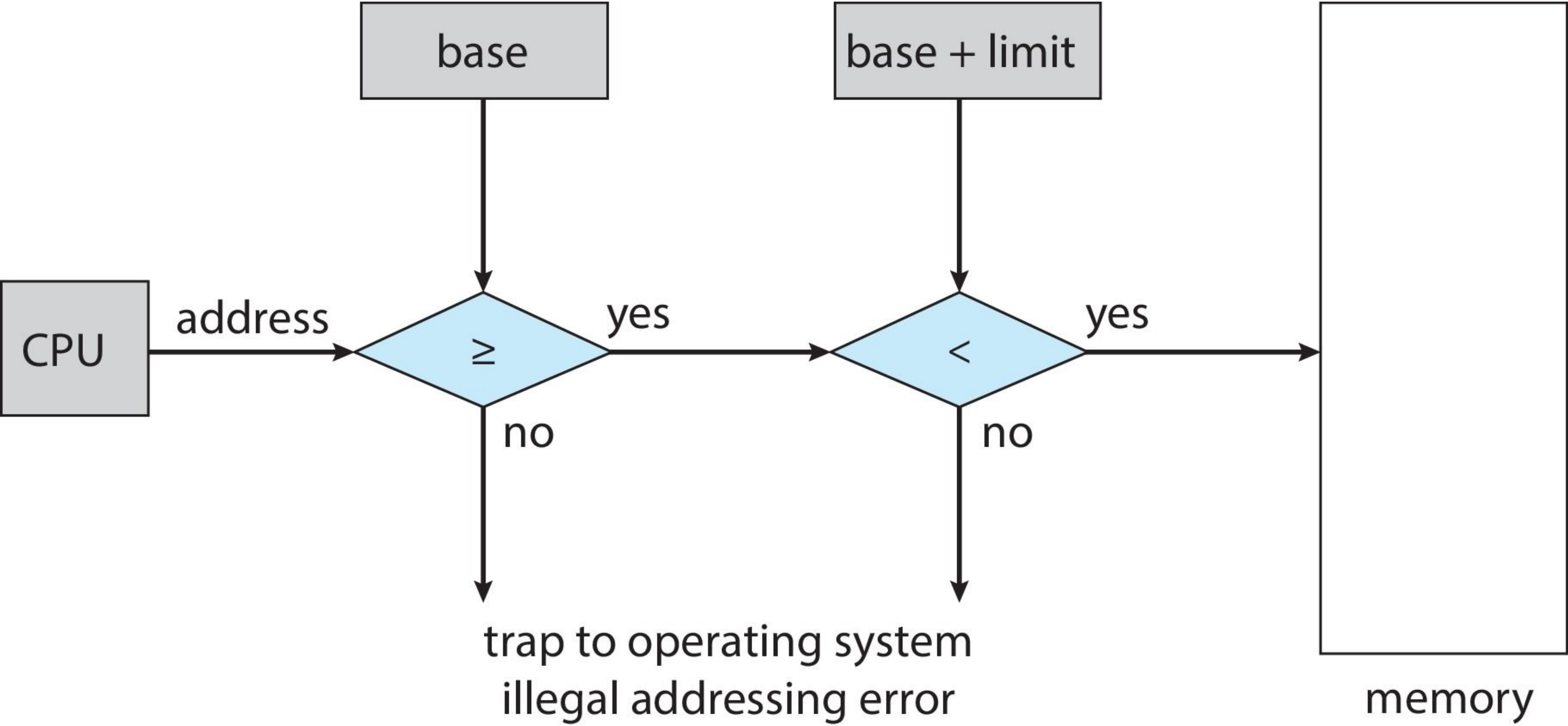
Memory Management Requirements

1. Basic hardware
2. Address Binding
3. Dynamic Loading
4. Dynamic Linking
5. Logical and Physical address space
6. Overlays.

Basic Hardware

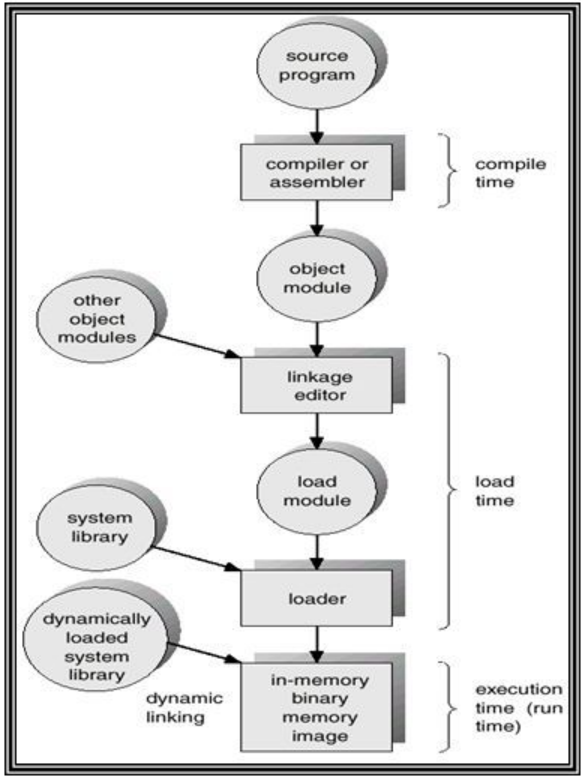
- The CPU can only access its registers and main memory.
- Memory accesses to registers are very fast, generally one clock tick, and a CPU may be able to execute more than one machine instruction per clock tick.
- The operating system must be protected from access by user processes and user processes must be restricted so that they can access only memory locations belonging to that particular process.
- This is implemented by using a base register and a limit register for each process.

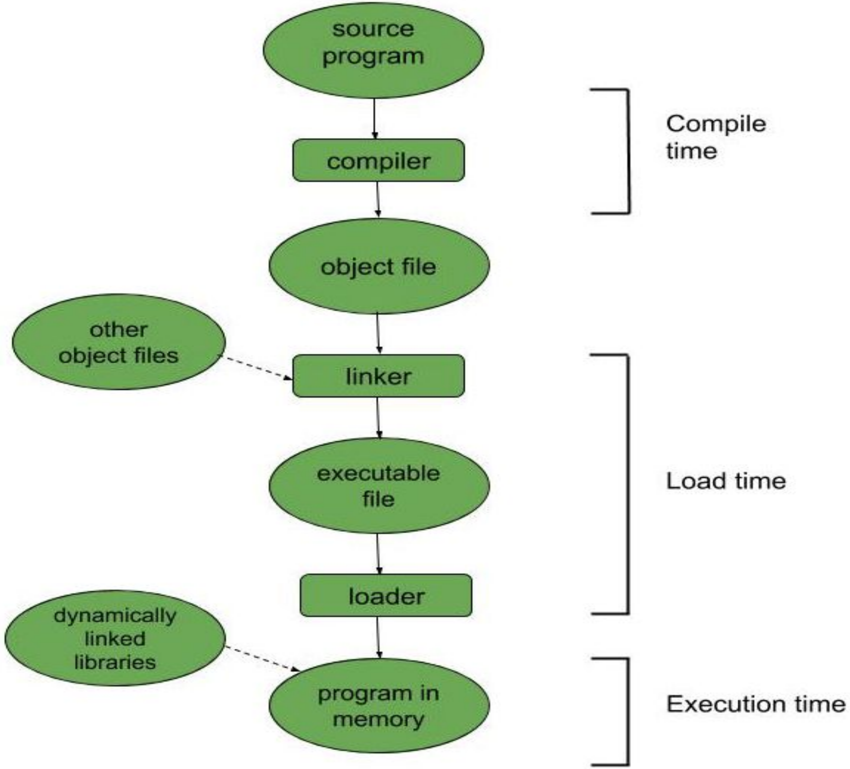

1024000
operating system
880000
process
420940
base + limit
process
300040
base
process
256000
0


base
base + limit
CPU
address
≥
yes
<
yes
no
no
trap to operating system
illegal addressing error
memory

Multistep Processing of a User Program

- ***Compiler*** **and** ***Assembler*** generate an *object file* (containing code and data segments) from each *source file*
- ***Linker*** combines all the object files for a program into a single executable object file, which is complete and self-sufficient
- ***Loader*** (part of OS) loads an executable object file into memory at locations determined by the operating system




source program
compiler
object file
other object files
linker
executable file
loader
dynamically linked libraries
program in memory
Compile time
Load time
Execution time

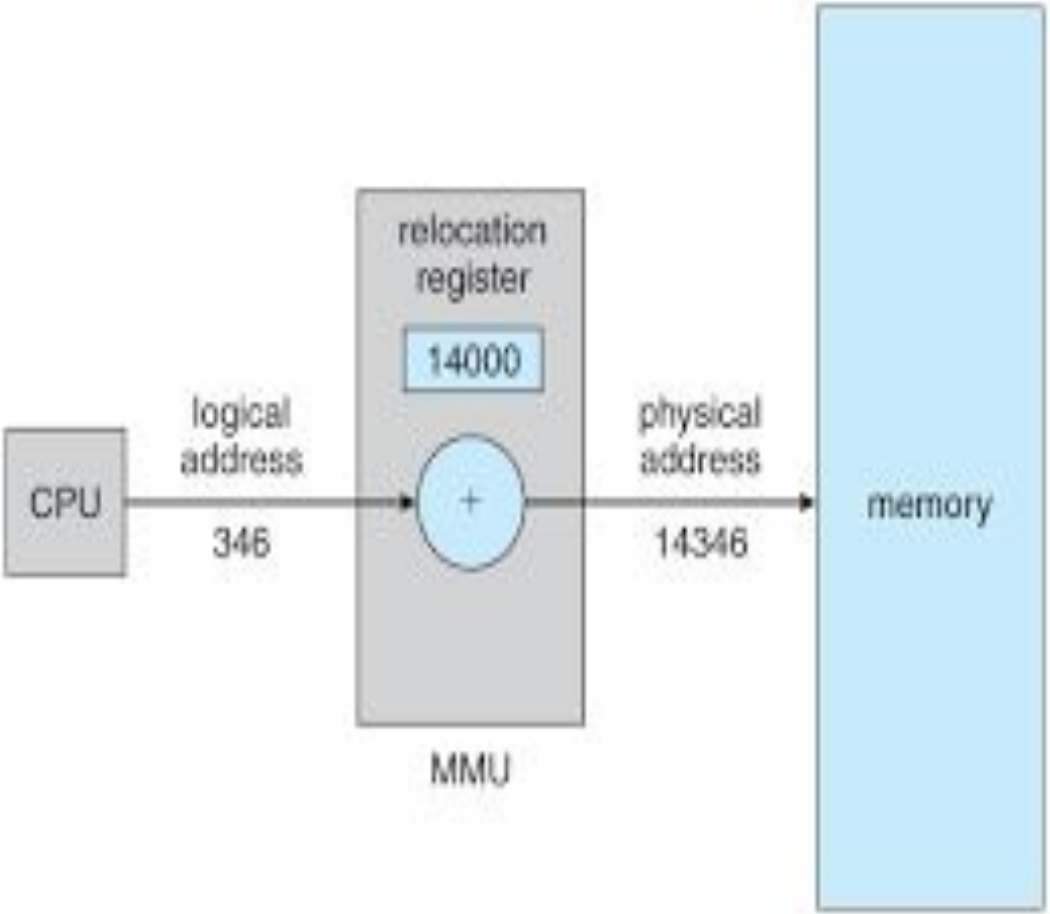
Logical and Physical Addresss

Logical Address:

It is the address generated by the CPU while program is running.

Physical Address:

It is the location in a memory unit.


relocation register
14000
CPU
logical address
346
+
physical address
14346
memory
MMU

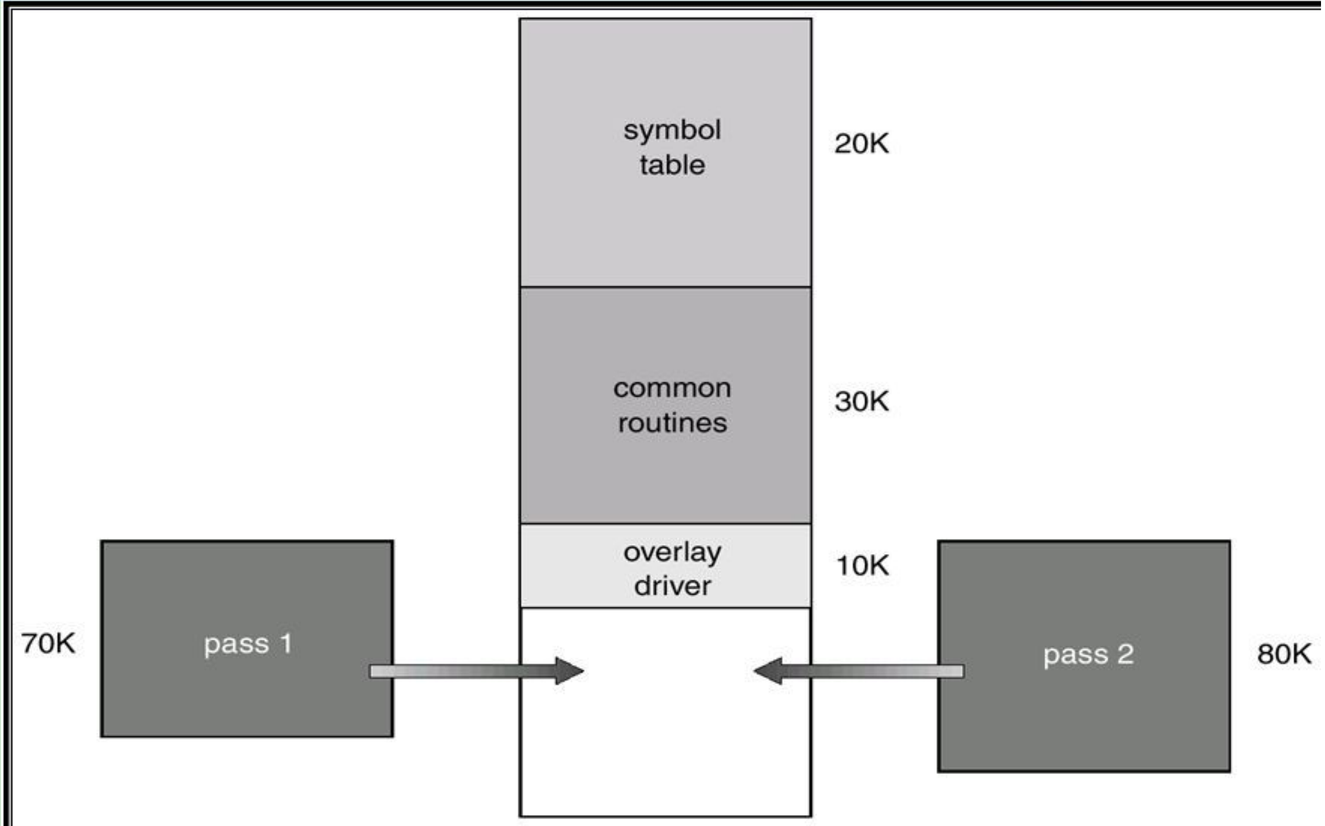
Logical Address	**Physical Address**
User can view the Logical address of Program	Can’t be viewed by the user
Used as a reference to access physical memory location	User can’t access it directly
Generated by CPU	Memory Management Unit

Overlays

- Overlays are used to enable a process to be larger than the amount of memory allocated to it.

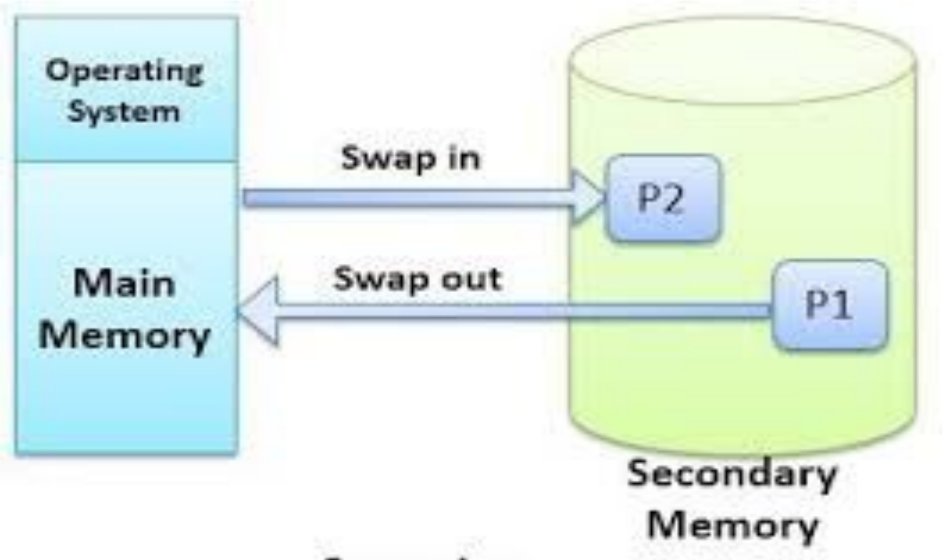
- The main objective of this scheme is to keep only those instructions and data in memory, which are required at that time.

Overlays for a Two-Pass Assembler

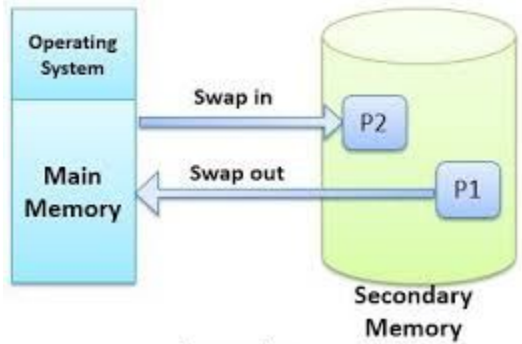


SWAPPING

Swapping of process between main memory and secondary memory.

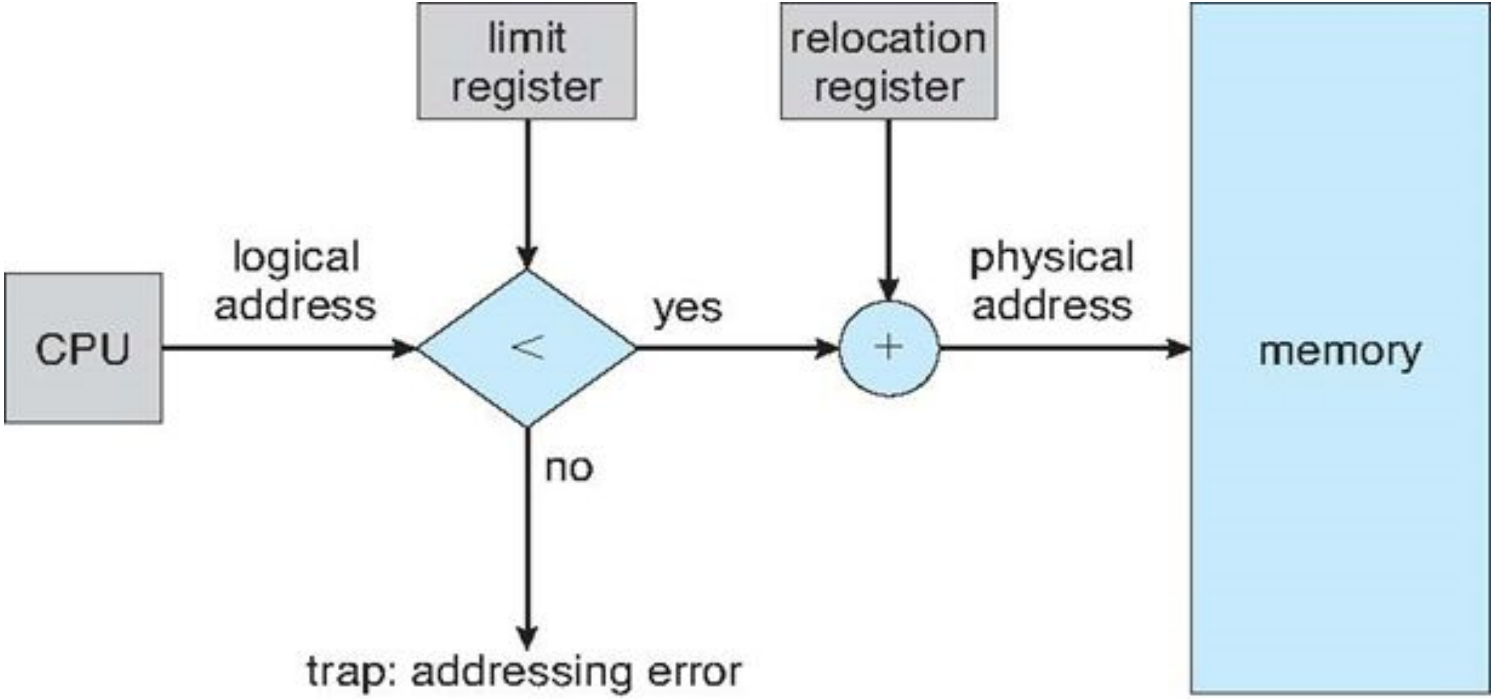


Swapping


Operating System
Swap in
P2
Main Memory
Swap out
P1
Secondary Memory

Swapping

MEMORY PROTECTION



CONTIGUOUS MEMORY ALLOCATION

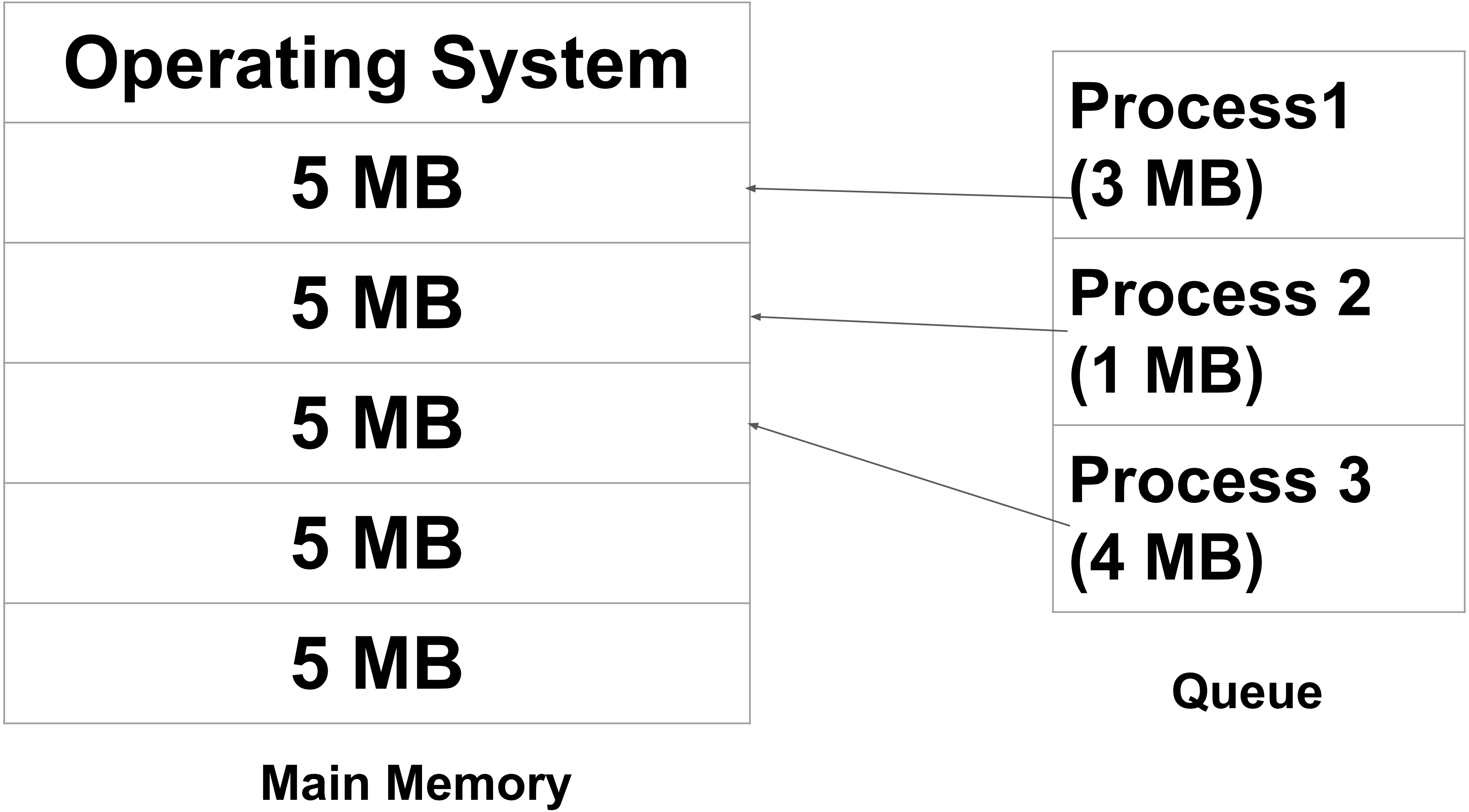
- Each process resided in a single contiguous section of memory.
- One of the ways of memory allocation is to divide the memory into several partitions of fixed size.
- The size do the partitions depends on the OS.
- Each partitions may accommodate only one process for execution.
- When a program terminates, that partition becomes free and another program from the queue is loaded into that partition.

CONTIGUOUS MEMORY ALLOCATION

- Multi-Programming with fixed Partitions.
- Multi-Programming with Dynamic Partitions

Fixed -size Partition Scheme

- Each process is allotted a fixed size continuous block in the main memory.



Operating System

5 MB

5 MB

5 MB

5 MB

5 MB

Main Memory

Process1 (3 MB)

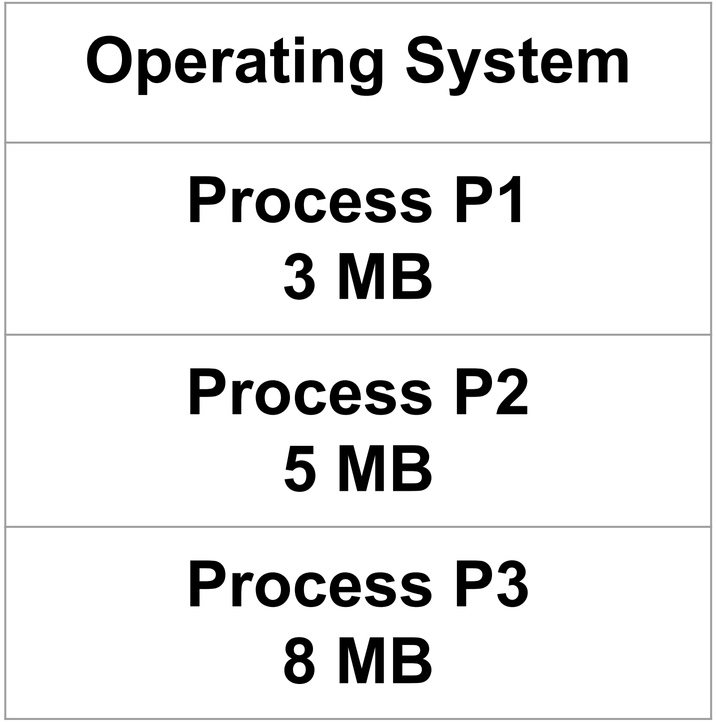
Process 2 (1 MB)

Process 3 (4 MB)

Queue

Variable-size Partition Scheme

- **No fixed blocks or partitions are made in the memory.**
- **Each process is allotted a variable sized block depending upon its requirements.**
- **Whenever a new process wants some space in the memory, if available, this amount of space is allotted to it.**
- **As the blocks are variable-sized, which is decided as processes arrive, this scheme is also called Dynamic Partitioning.**



Operating System

Process P1
3 MB

Process P2
5 MB

Process P3
8 MB

Strategies used for Continguous memory allocation.

1. First fit

   First fit allocated the first free partition which is large enough to accommodate the process. Searching may start from low memory or high memory. However searching is stopped as soon as a free partition that is large enough is found.

2. Best fit

It allocated the smallest free partition that is large enough for the process. In this the entire list must be searched to get the nearest exact size.

3. Worst fit

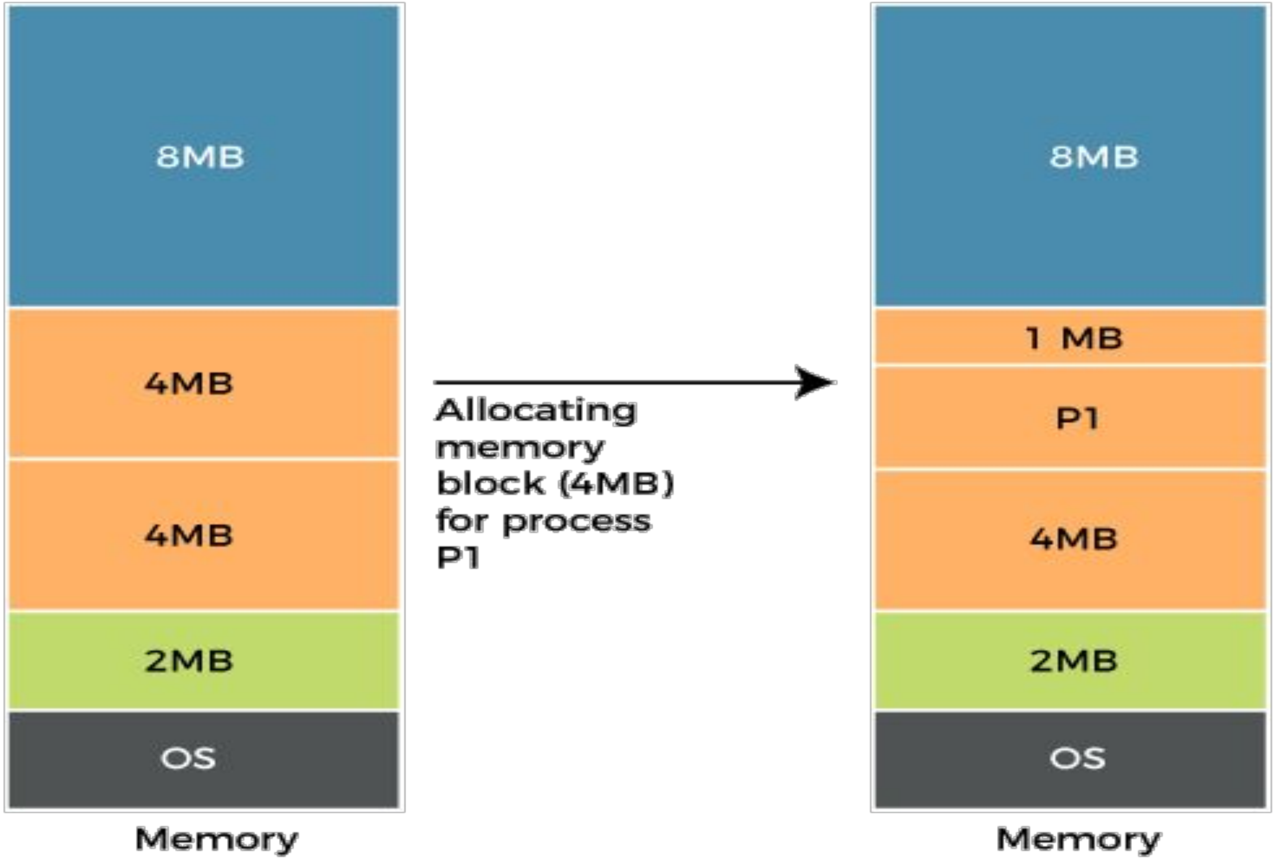
It allocates the largest free partition. The entire list must be searched for the largest sized partition.

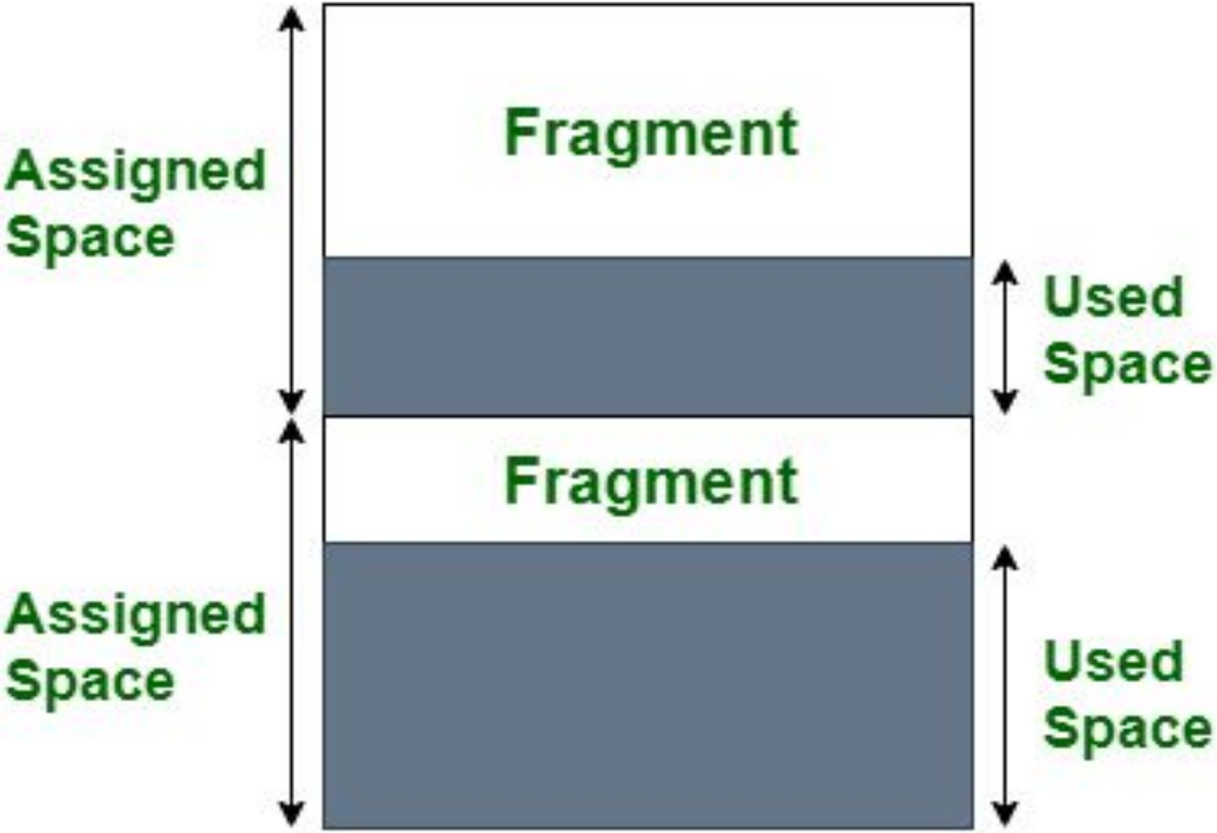
First fit	Best fit	Worst fit
Allocates the first free partition which is big enough	Allocates the smallest free partition	Allocates the largest partition
Terminates searching as soon as it finds the required partition	Searches the entire partition description table	Searches the entire partition description table
Execution Fastest	Comparatively slower execution	Slower compared to first fit & best fit
Lesser efficient utilization of memory	Most efficient utilization of memory	Lesser efficient utilization compared to both the other methods

Fragmentation

Internal Fragmentation

- It occurs when fixed size memory blocks are allocated to the processes.
- A free space is created when memory assigned to the process is slightly larger than the memory requested by process.

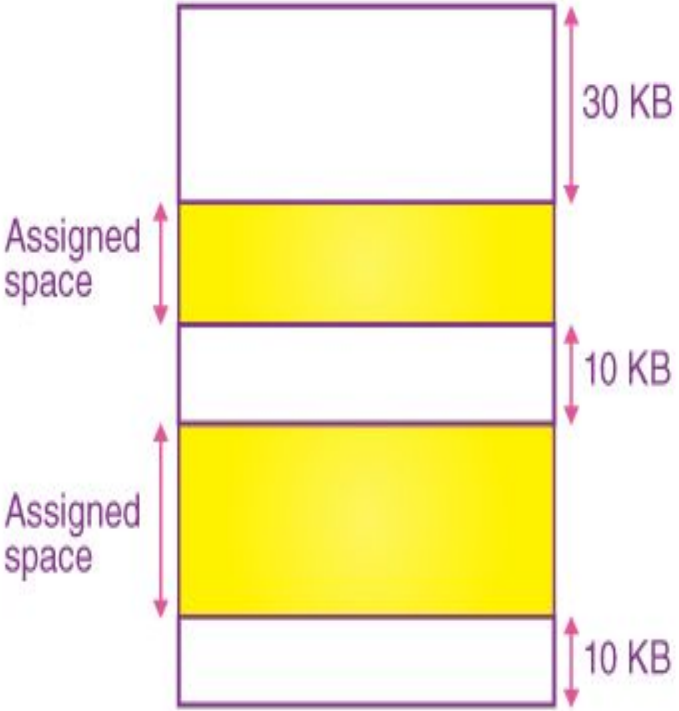

8MB
4MB
4MB
2MB
OS
Memory
Allocating memory block (4MB) for process P1
8MB
1 MB
P1
4MB
2MB
OS
Memory



Internal Fragmentation

External Fragmentation

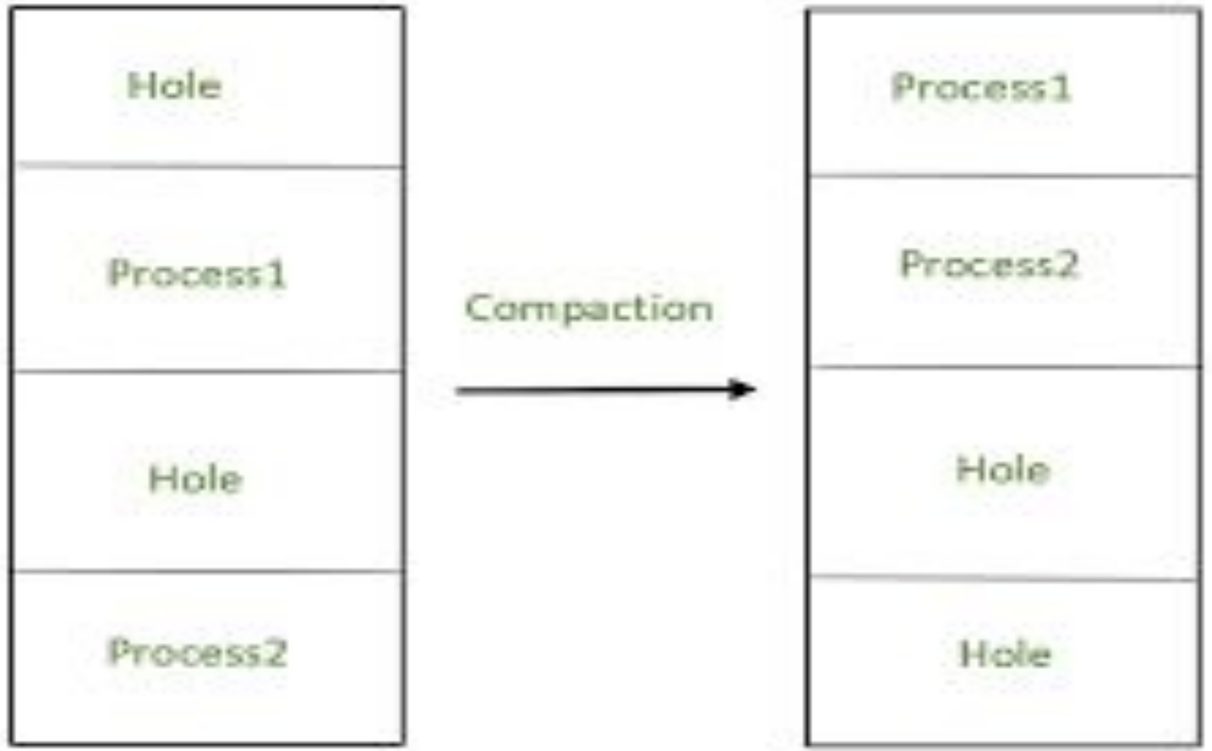
- It occurs when variable size memory space are allocated to process dynamically.
- It happens when there's sufficient memory to satisfy request but can't be fulfilled as it is non contiguous manner.



Process 05 needs 45 KB space

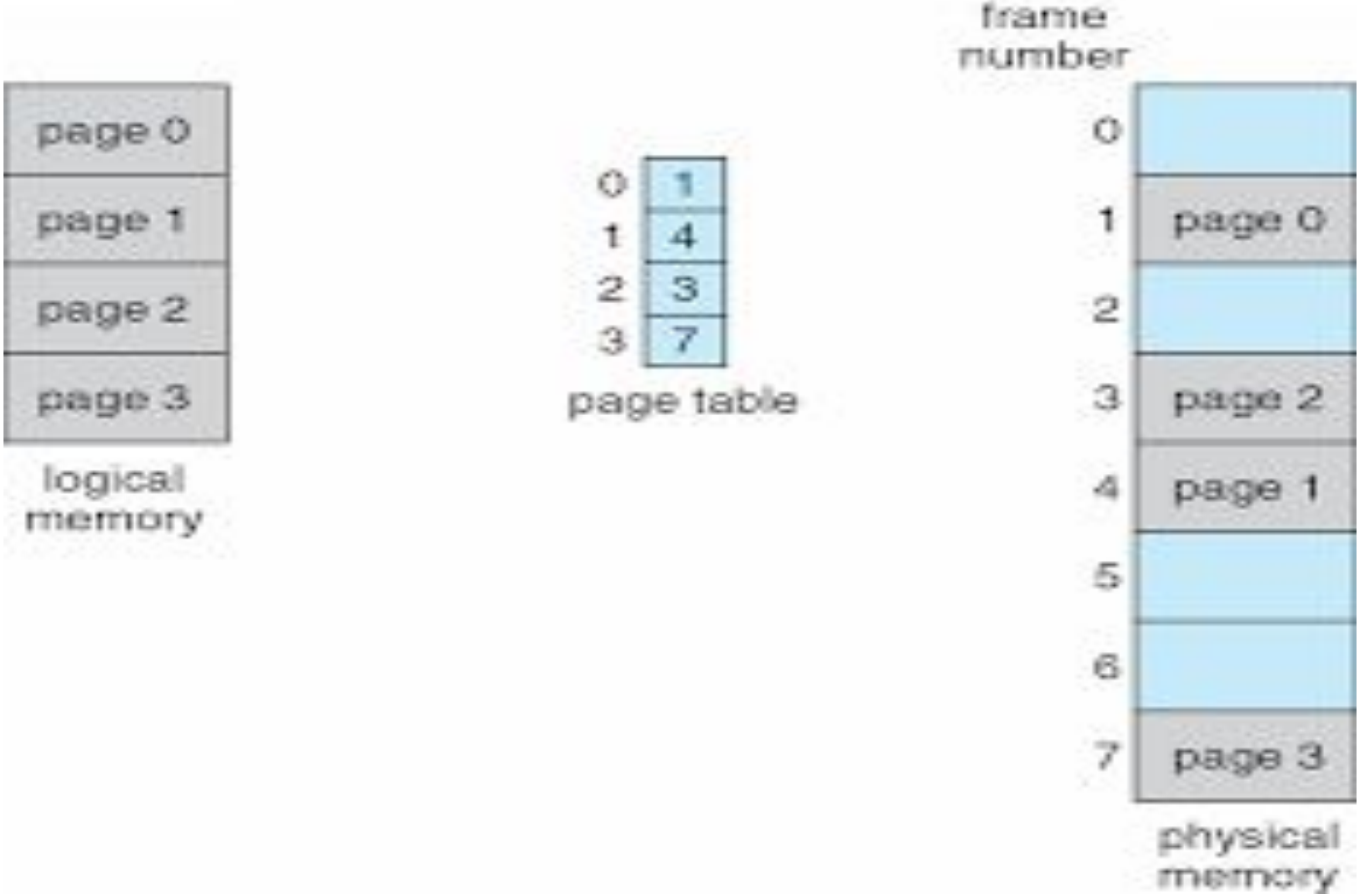
Compaction

Compaction is **a technique to collect all the free memory present in form of fragments into one large chunk of free memory**, which can be used to run other processes.


Hole
Process1
Hole
Process2
Compaction
Process1
Process2
Hole
Hole

Paging

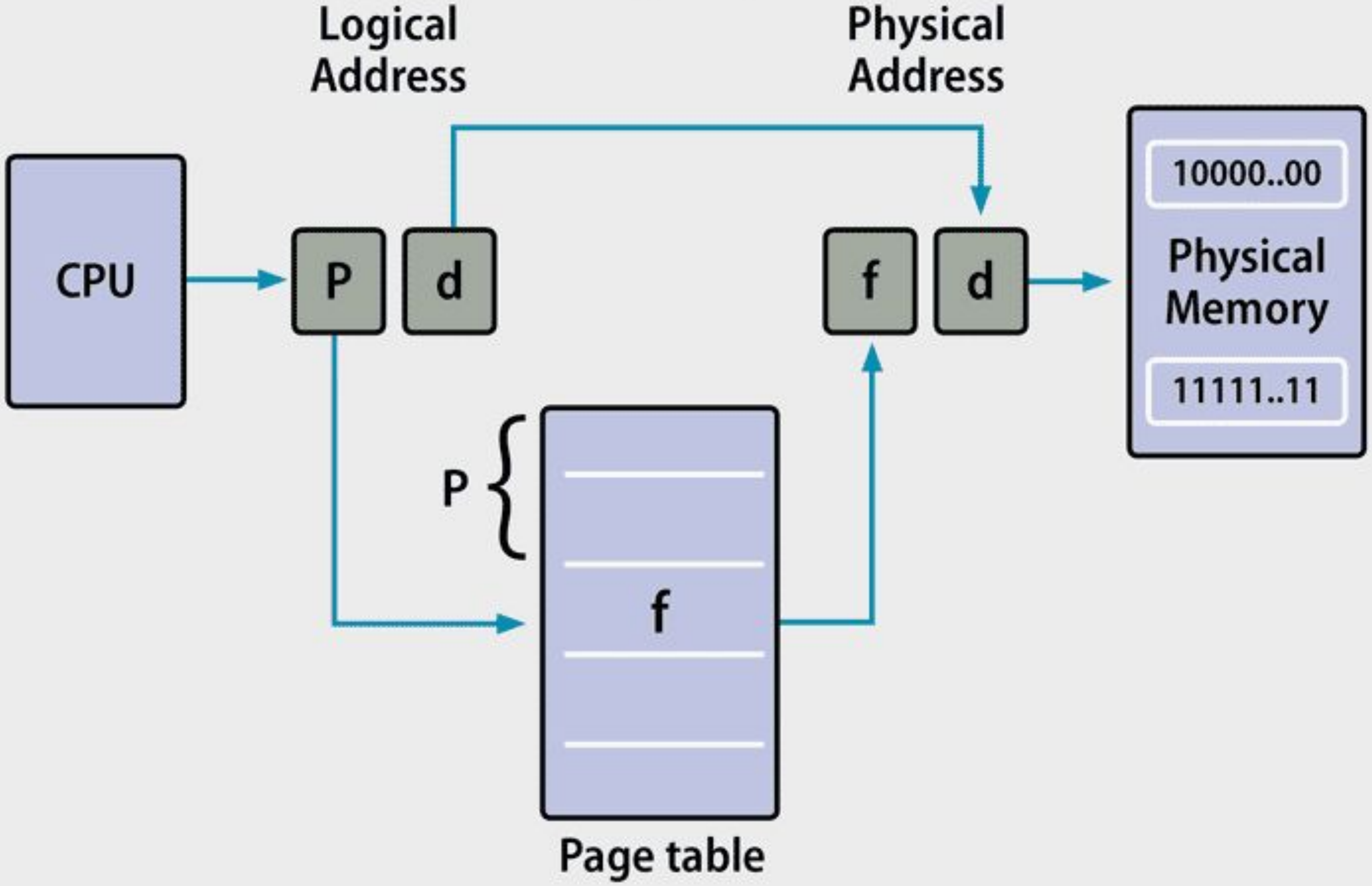
Paging is **a storage mechanism used in OS to retrieve processes from secondary storage to the main memory as pages**. The primary concept behind paging is to break each process into individual pages. Thus the primary memory would also be separated into frames.



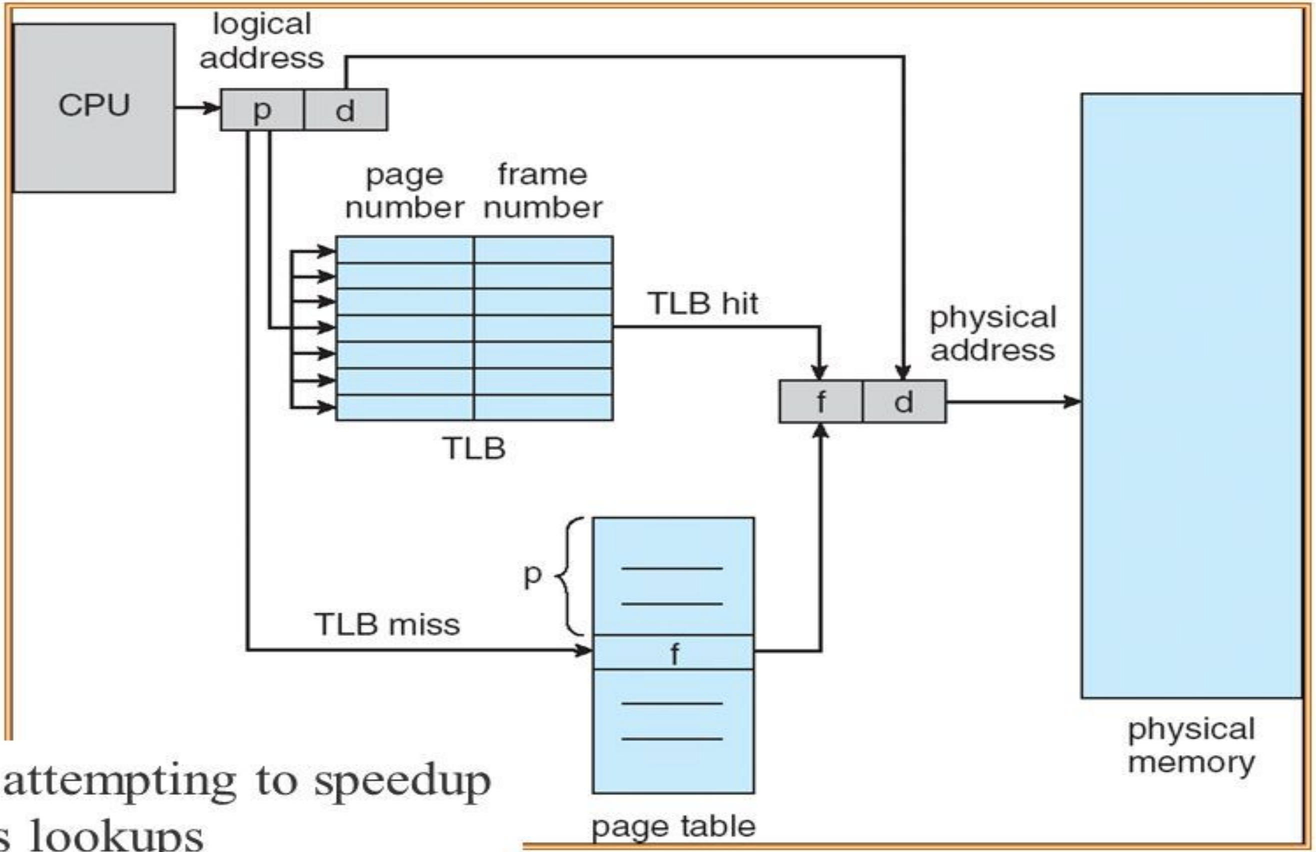
page 0
page 1
page 2
page 3
logical memory
page table
frame number
0
1
2
3
4
5
6
7
page 0
page 2
page 1
page 3
physical memory

0	1
1	4
2	3
3	7

Paging

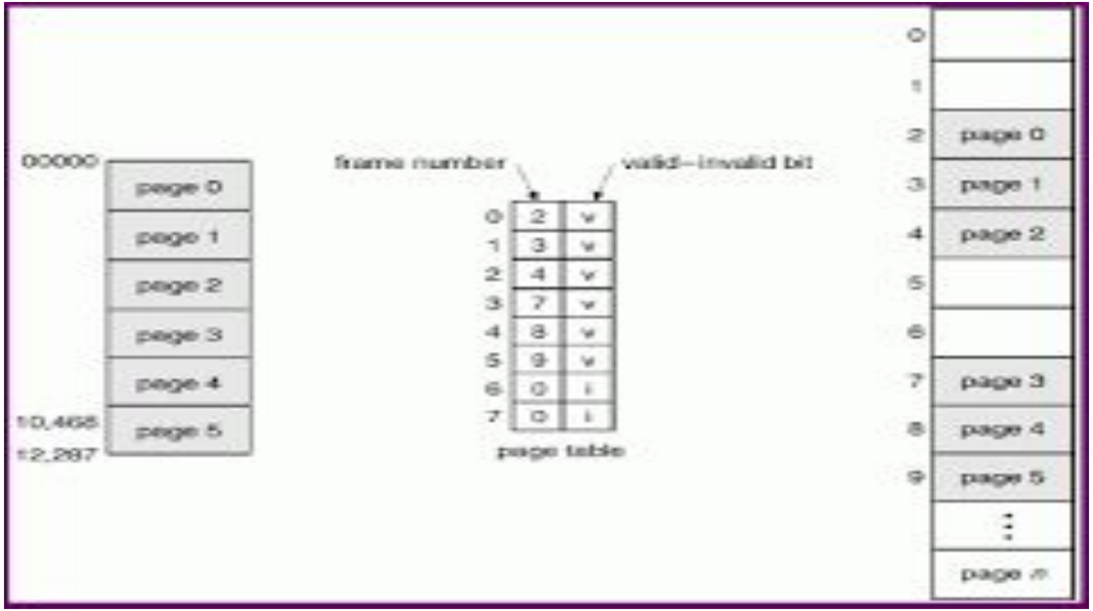

Logical Address
Physical Address
CPU
P
d
f
d
10000..00
Physical Memory
11111..11
P
f
Page table

Paging Hardware With TLB



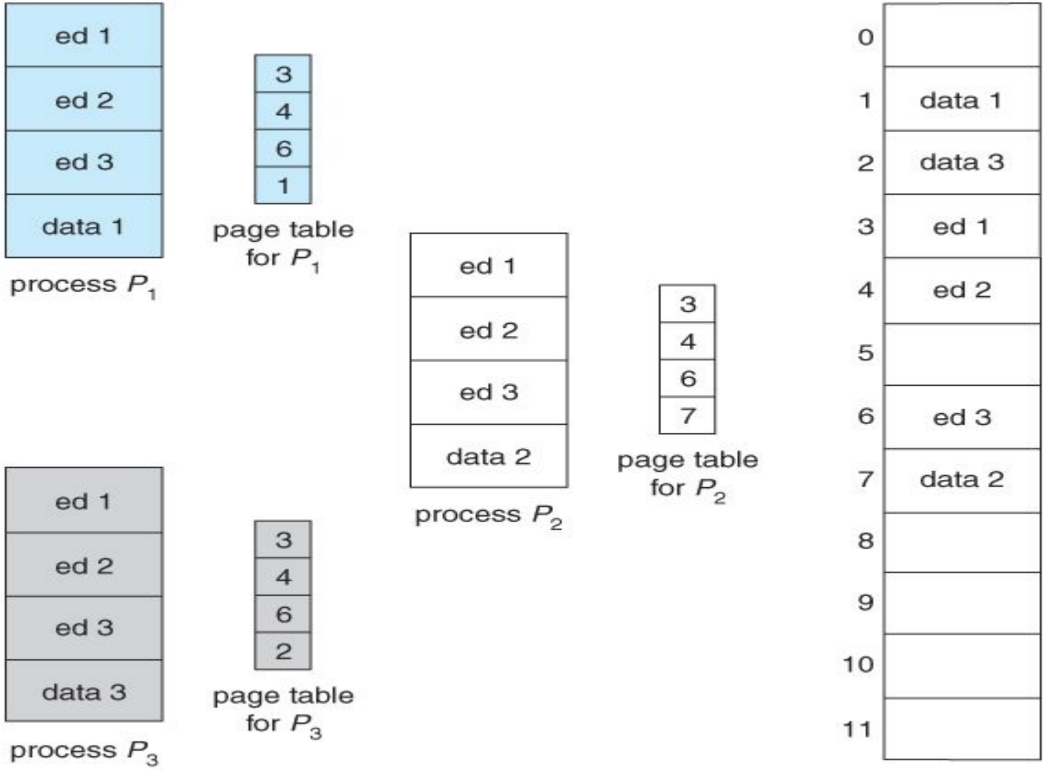
We are attempting to speedup address lookups

Protection



0	2	v
1	3	v
2	4	v
3	7	v
4	8	v
5	9	v
6	0	i
7	0	i

Shared Pages



segmentation

Problem with paging - Paging don’t bother where our programs start and end, it simply divides our program into pages

To achieve user view of memory allocation, segmentation is used.

ALLOCATION OF FRAMES

1. Equal frame allocation
2. Proportional frame allocation
3. Priority Frame allocation
4. Global Replacement Allocation
5. Local Replacement Allocation

1. Equal frame allocation

The frames will be allocated equally among available processes.

P1 —-> 10 FRAMES

P2 —---> 30 FRAMES

MAIN MEMORY - 40 FRAMES

P1 —-->20 FRAMES

P2—--->20 FRAMES

2. Proportional frame allocation

Frames will be allocated based upon size of the process.

P1 —--> 10 FRAMES

P2 —----> 30 FRAMES

MAIN MEMORY - 40 FRAMES

P1 = 10/(10+30)* 40 = 10

P2 = 30/(10+30)*40 = 30

3. Priority Frame allocation

Based on the priority the frames will be allocated to the process. The process which is having higher priority will be allocated more number of frames. The process which is having low priority will be allocated less number of frames.

4. Global Replacement Allocation

Allows a process to select a frame from the list of all the frames, eventhough the frame is allocated to some other process.

One process may take a frame from another.

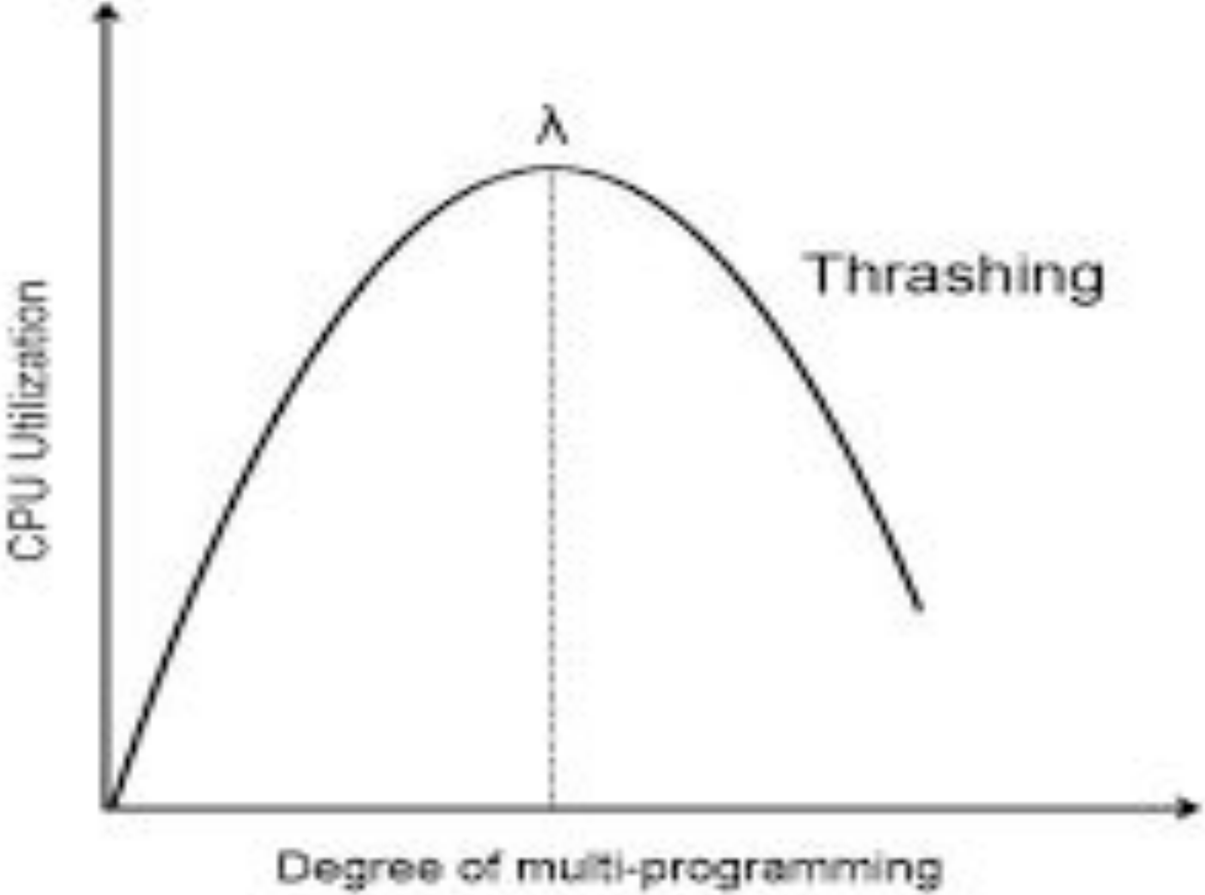
5. Local Replacement Allocation

Allows a process to select a frame from its own allocated frames.

Thrashing

- ın the initial stage when we increase the degree of multiprogramming, CPU utilization is very high upto lamda.
- Further, if we increase the degree of multiprogramming , CPU utilization is drastically falling down.
- This situation in the system is Thrashing
- Thrashing degrades the performance of system.

- **There can be a situation when main memory is full of pages that are accessed frequently. A page fault will occur if the required page is not present.**
- **Inorder to make space for swapping in the required page, one of the frequently accessed page is swapped out.**
- **Soon the swapped out page is required for execution and this again results in page fault**
- **Thus a series of page fault occur and swapping becomes a large overhead.**


λ
Thrashing
CPU Utilization
Degree of multi-programming

File Management

- **A file is a collection of related information stored on a secondary storage device.**
- **A file is a collection of records, where each record contains some fields, where each field represents some data.**
- **A file is a collection of bits, bytes, characters and words.**

File Attributes

- Name
- Identifier
- Type
- Location
- Protection
- Time, date and user identification

File Operation

Create

Write

Read

Delete

Close

Append

Rename

File Type

ordinary files

These are files that contain user information. These may have text, databases or executable program.

Directory files:

These files contain list of file names and other information related to these files.

Special files

These files are also known as device files, represent physical devices like disk, terminals, printers, networks…etc.

executable	exe, com, bin
object	obj, o
Source code	Java, c, perl
batch	Bat, sh
multimedia	Mov, mp3, mp4, avi

File Access Methods

1. Sequential Access Methods
2. Direct Access Methods
3. Indexed Access Method.

1. Sequential Access Methods

Eg: Magnetic tapes

One record is processed after other.

Supports following operation

Read next

Write next

rewind

2. Direct Access Methods

It is based on disk model. It allows random access. User can jump to any record and access that record. We can access any record directly

Read n

Write n

Jump to n

Indexed Access Methods

Index is created which contains a key field and pointers to various blocks

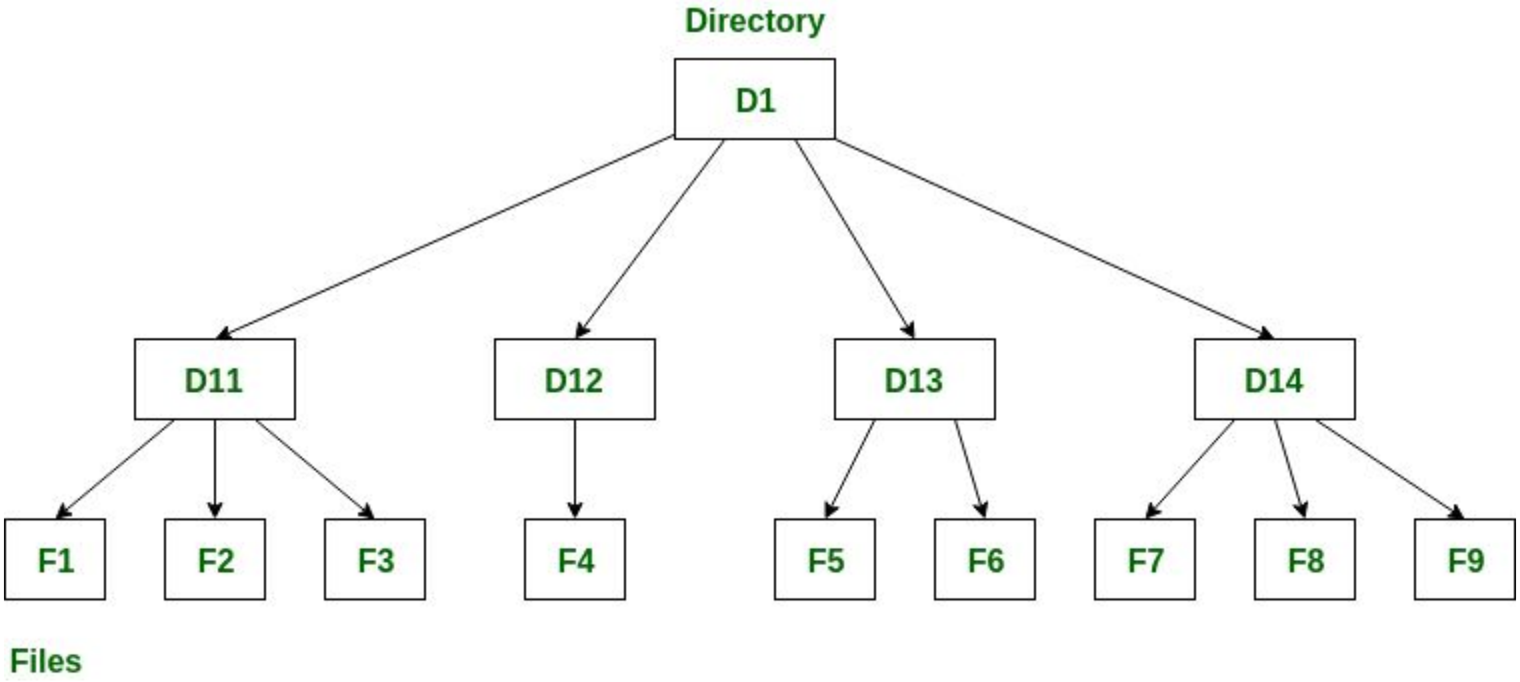
Directory Structure

Directory is mainly used inorder to organize files. Its a collection of files.

Directory may contain sub directory

Using Directory we can store the files and also attributes(size , access permission, when the file created) of the files are stored under the directory

A **directory** is a container that is used to contain folders and files. It organizes files and folders in a hierarchical manner.


Directory
D1
D11
D12
D13
D14
F1
F2
F3
F4
F5
F6
F7
F8
F9
Files

Four levels of Directory Structure

Single-Level Directory

Two-Level Directory

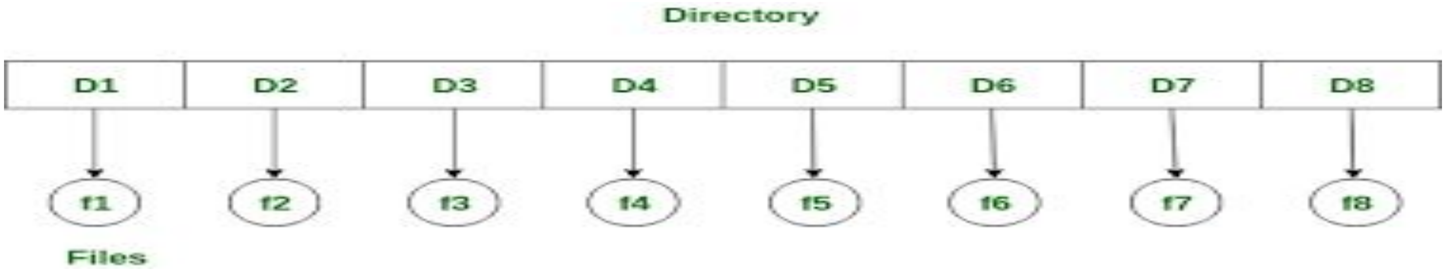
Tree-Structured Directory

Acyclic Graph Directory

General Graph Directory Structure.

Single level Directory

A single directory for all the users



Advantages

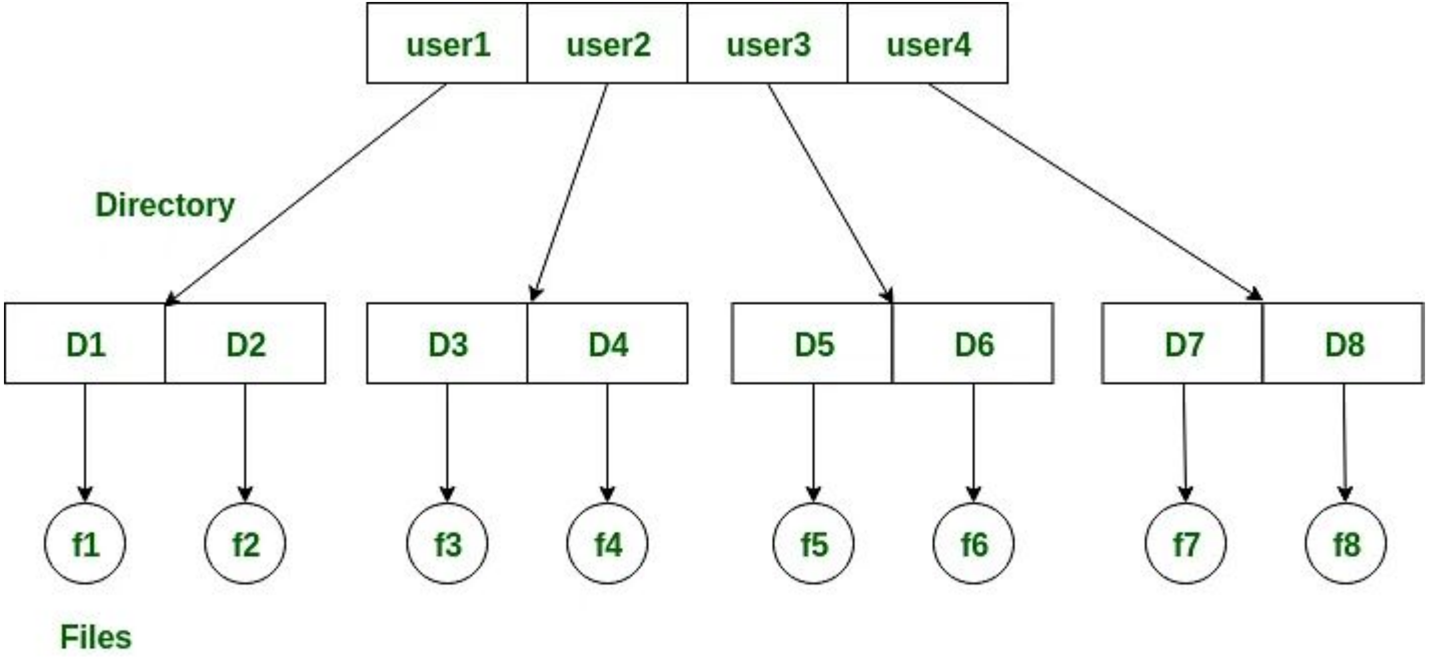
- It is simple to implement
- If files are smaller in size, searching will be faster.
- File operations can be implemented very easily

Disadvantages

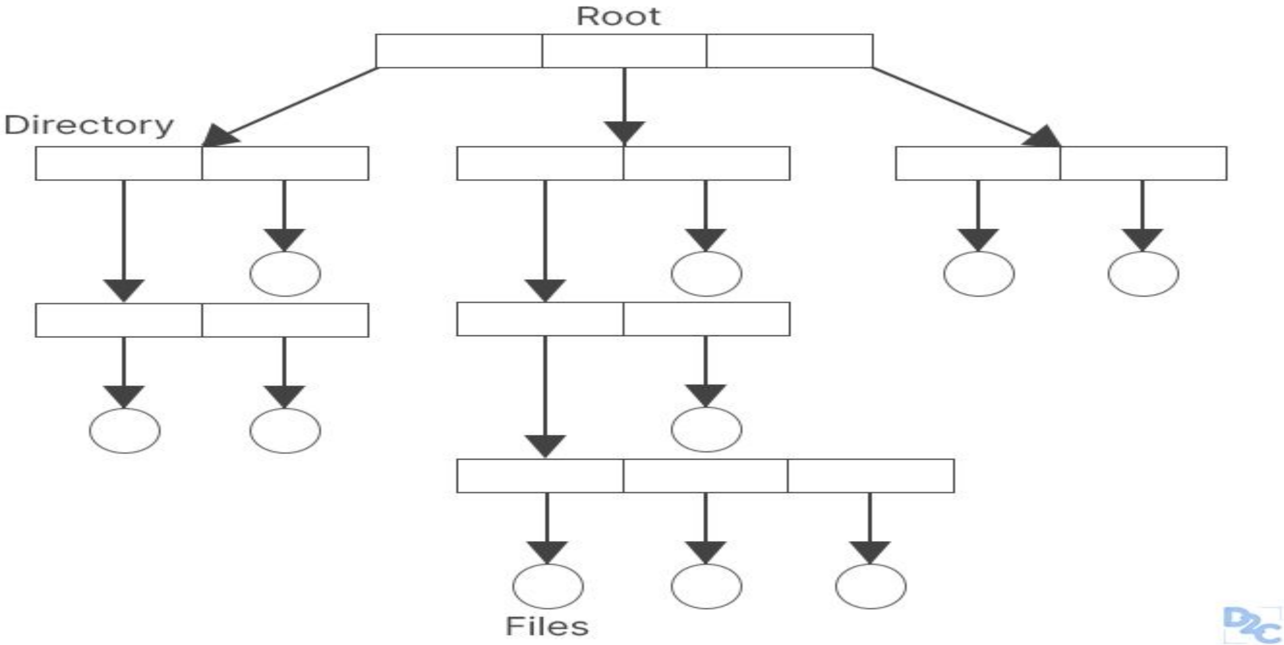
Chance of name collision because two files cannot have the same name.

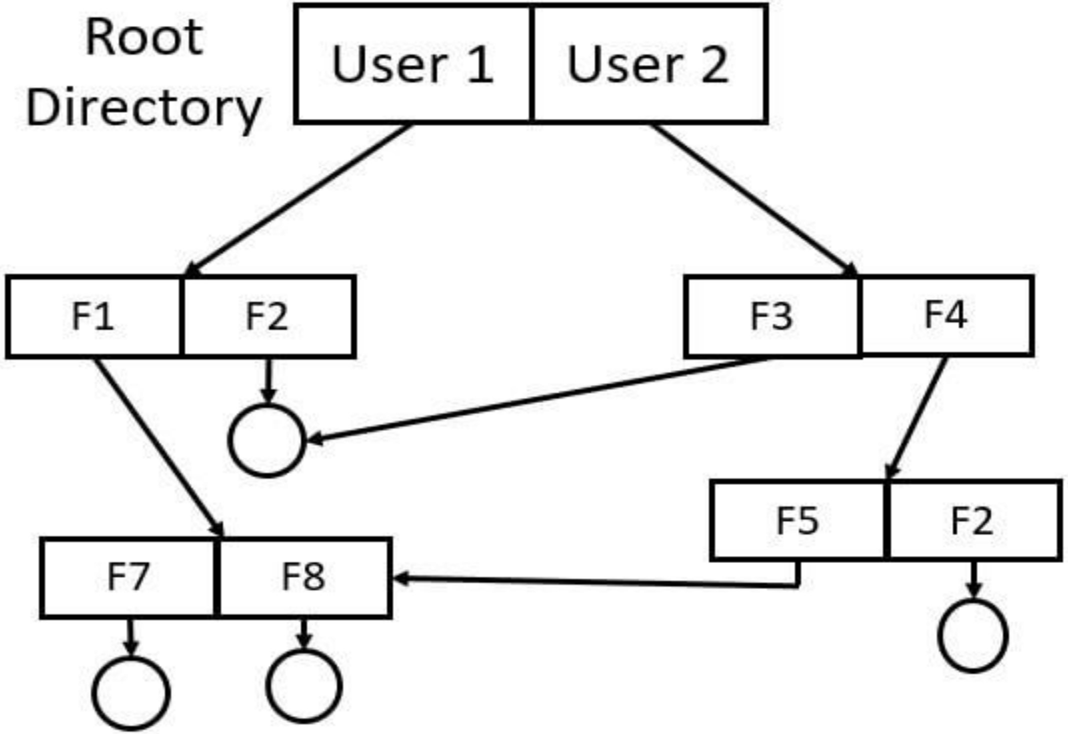
Two level Directory

Seperate Directory for each user



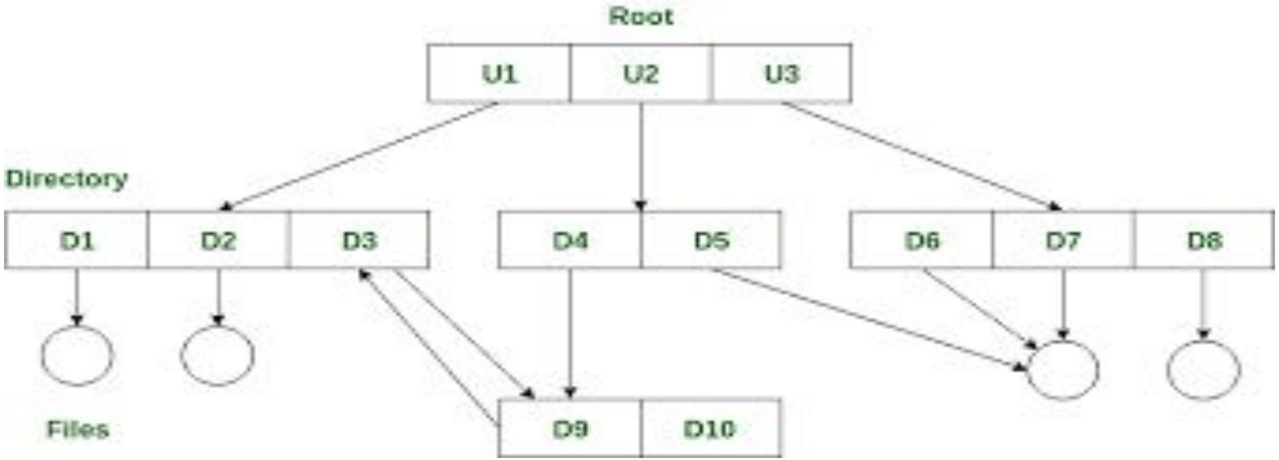
Tree-Structured Directory





Acyclic Graph Directory Structure

General Graph Directory Structure


Root
U1
U2
U3
Directory
D1
D2
D3
D4
D5
D6
D7
D8
Files
D9
D10