0301304 FUNDAMENTAL OF OPERATIONG SYSTEM

UNIT	MODULES	WEIGHTAGE
1	INTRODUCATION TO OPERATING SYSTEM	20 %
2	PROCESS MANAGEMENT	20 %
3	PROCESS COMMUNICATION AND SYNCHRONIZATION	20 %
4	MEMORY MANAGEMENT	20 %
5	FILE MANAGEMENT , DISK MANAGEMENT , SECURITY AND PROTECTION	20 %

UNIT – 4 Memory Management

- Basic Memory Management
  - Introduction
  - Basic Concepts
    - Static and Dynamic Allocation
    - Logical and Physical Addresses
    - Fixed and Variable Memory Partitioning
    - Fragmentation
    - Swapping
  - Contiguous Memory Allocation
    - Compaction
    - Memory Allocation Techniques

UNIT – 4 Memory Management

- Paging Concept
- Segmentation
- Virtual Memory
  - Introduction
  - Need for virtual Memory
  - Demand Paging
  - Page Replacement Algorithm
    - FIFO
    - LRU
  - Thrashing

UNIT – 4 Basic Mamory Management

- The multi programming concept of an OS gives rise to another issue known as **memory management.**
- Memory, as a resource, **needs to be partitioned and allocated to the ready processes**, such that both **processor and memory** can be utilized efficiently.
- The **division of memory for processes needs proper management,** including its efficient allocation and protection.
- **There are two types of memory management :**
  - **Real memory (Main Memory)**
  - **Secondary memory**

UNIT – 4 Basic Mamory Management

- Memory allocation is generally performed through two methods:
  - **Static Allocation**
  - **Dynamic Allocation**
- **Static Allocation**
  - The allocation is done **before the execution** of a process.
- **Dynamic Allocation**
  - If memory allocation is **deferred (at later time) till the process starts executing**, it is known as Dynamic Allocation.

UNIT – 4 Basic Mamory Management

- **Static Allocation**
  - There are two instances when this type of allocation is performed:
    - When the **location of the process in the memory is known at compile time,** the compiler generates an **absolute code for the process.**
    - When the **location of the process in the memory is NOT known at compile time,** the compiler does **not produce an actual memory address but generate a relocatable code** (Relocatable code is software whose execution address can be changed)**,** that is, the addresses that are relative to some known point.

UNIT – 4 Basic Mamory Management

- **Dynamic Memory Allocation**
  - In Multi-Programming, **Modern OS adopt dynamic memory allocation method.**
  - In this method, two types of addresses are generated.
    - **Logical Addresses**
    - **Physical Addresses**

- **Logical Addresses**
  - In dynamic allocation, the **place of allocation of the process is not known at the compile time and load time**.
  - The processor, at compile time, generate some address, known as ***logical addresses.***
  - The **set of all logical addresses** generated by the compilation of the process is **known as logical address space.**

UNIT – 4 Basic Mamory Management

- **Physical Addresses**
  - Logical addresses **need to be converted into absolute addresses** at the time of execution of the process.
  - The absolute addresses are known as **physical addresses.**
  - The set of physical addresses generated, corresponding to the logical addresses during process execution, is known as **physical address space.**
  - *When a process is compiled, the CPU generates a logical address, which is then converted into a physical address by the memory management component to map it to the physical memory.*

UNIT – 4 Basic Mamory Management

- **Swapping**
  - There are some instance in multi programming **when there is no memory for executing a new process.**
  - In this case, **if a process is taken out of memoy, there will be space for a new process.**

UNIT – 4 Basic Mamory Management

- **Swapping**
  - It raise some question :
    - Where will this process reside?
    - Which process will be taken out?
    - Where in the memory will process be brought back?

UNIT – 4 Basic Mamory Management

- **Swapping**
  - It raise some question :
    - **Where will this process reside?**
      - Secondary storage (generally Hard disk) known as backing store.
      - The action of taking out a process from memory is called **swap-out.** The process is known as a **swapped-out process.**
      - The action of bringing back the swapped-out process into memory is called **swap-in.**

UNIT – 4 Basic Mamory Management

- **Swapping**
  - A separate space in the hard disk as **swap space**, is reserved for swapped out processes.

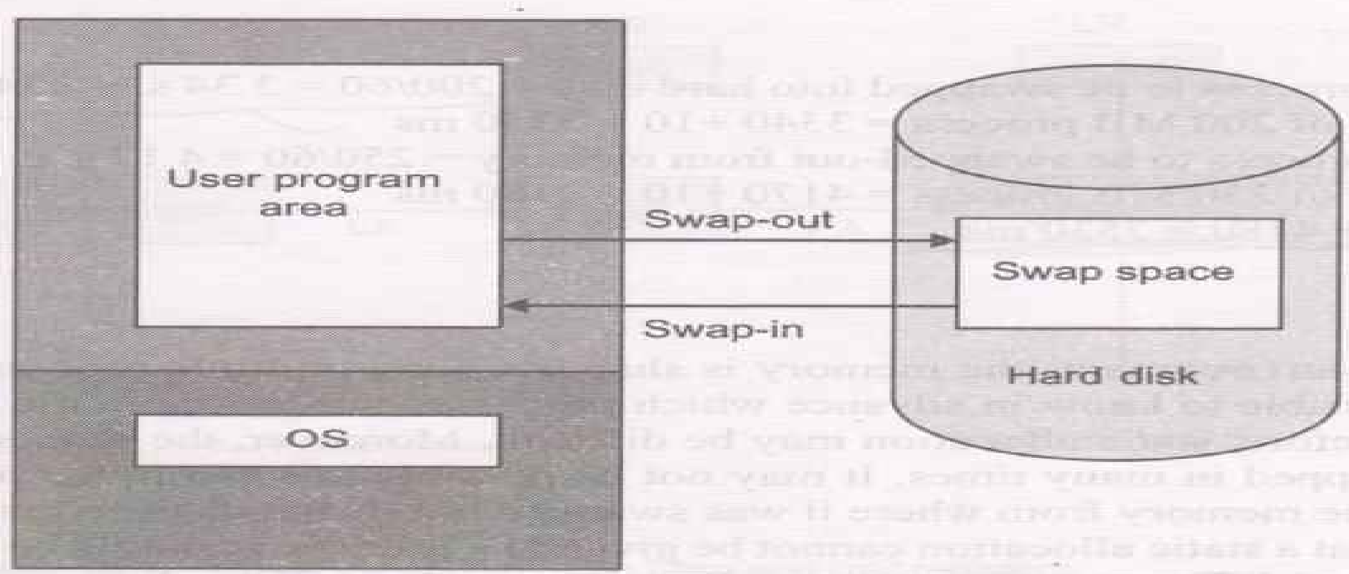


Fig. 10.1 Swapping

UNIT – 4 Basic Mamory Management

- **Swapping**
  - It raise some question :
    - **Which process will be taken out?**
      - In **round robin process-scheduling**, the processes are executed, according to the their time quantum. **If the time quantum expires and a process has not finished its execution, it can be swapped – out.**
      - In priority – driven scheduling, if a higer – priority process wishes to execute, **lower – priority process in memory will be swapped out.**
      - The **blocked processes,** which are waiting for an I/O, **can be Swapped out.**

UNIT – 4 Basic Mamory Management

- **Swapping**
  - It raise some question :
    - **Where in the memory will process be brought back?**
      - There are two options to swap
        - The **first option** is to swap – in the process at the **same location**, if there is compile time or load time binding.
        - **Other option** is to place the swapped -in process **any where** there is space. Need to relocation.

UNIT – 4 Basic Mamory Management

- **Swapping Time**
  - A time take to acces the hard disk.
- **Example :**
  - A process of size 200 MB needs to be swapped into the hard disk. But there is no space in memory. A process of size 250 MB is lying idle in memory and therefore, it can be swapped out.

    How much swap time is required to swap-in and swap-out the processes if:

    - Average latency time of hard disk = 10 ms
    - Transfer rate of hard disk = 60MB / s

UNIT – 4 Basic Mamory Management

- **Solution :**
  - The transfer time of the process to be **swapped-in** to hard disk = 200 / 60 = 3.34 s = 3340 ms
  - The swap time of 200 MB process = 3340 + 10 = 3350 ms
  - The transfer time of the process to be **swapped-out** form memory = 250 / 60 = 4.17 s = 4170 ms
  - The swap time of 250 MB process = 4170 + 10 = 4180 ms
  - **Total swap time** = 3350 + 4180 = 7530 ms

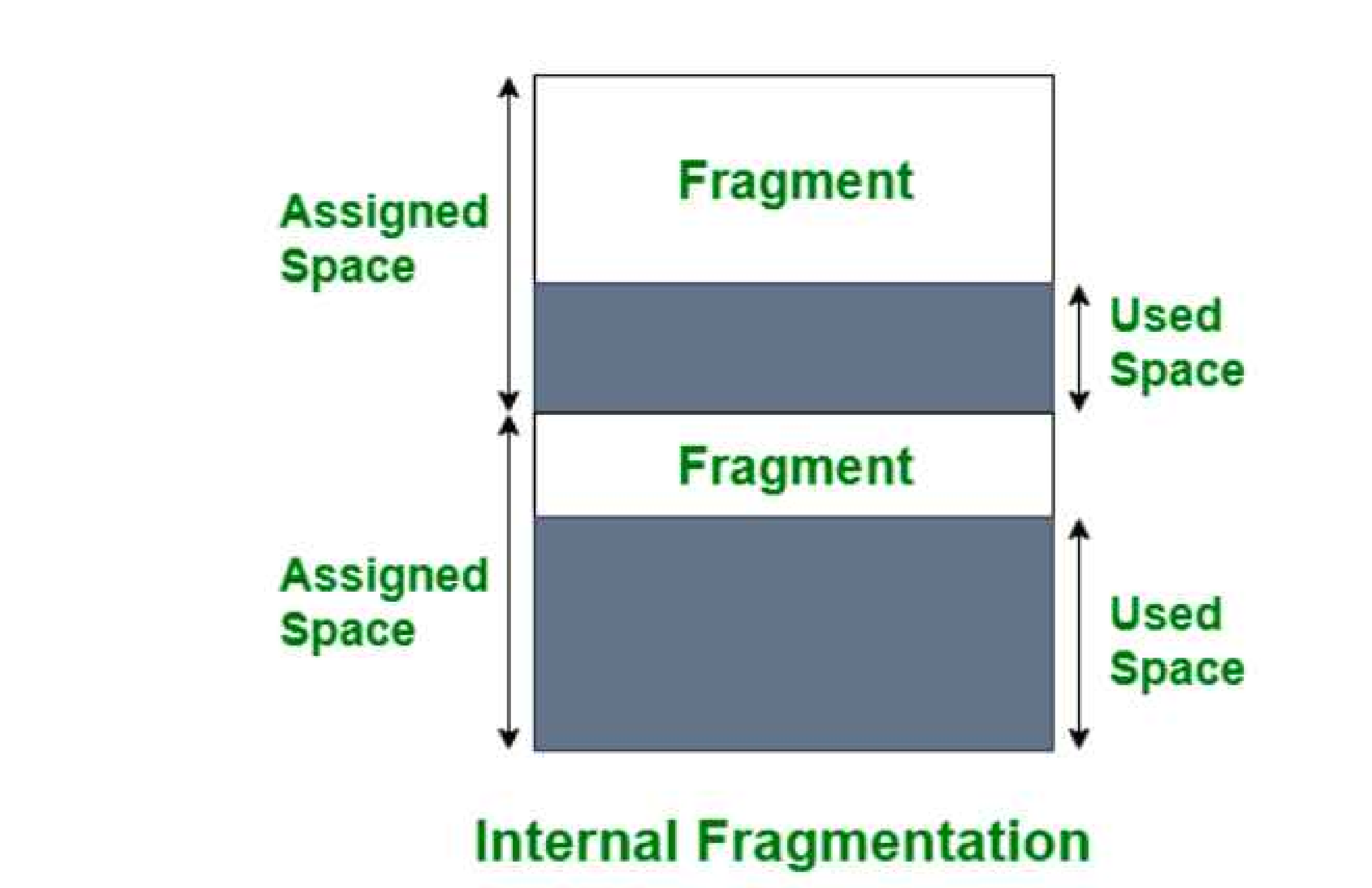
UNIT – 4 Basic Mamory Management

- **Fixed and Variable Memory Partitioning**
  - **Fixed Partitioning**
    - In this method of **partitioning, the memory is partitioned at the time of system generation.**
  - **Variable Partitioning**
    - In this method, partitioning is not performed at the system generation time.
    - The partition **are created at runtime,** by the OS

UNIT – 4 Basic Mamory Management

- **Fragmentation**
  - **Internal Fragmentation**
    - When a process is allocated to partition, it may be possible that its size is less than the size of partition.
    - It leave a space after allocation, which is unusable by any other process, this wastage of memory, internal to a partition is known as **internal Fragmentation.**

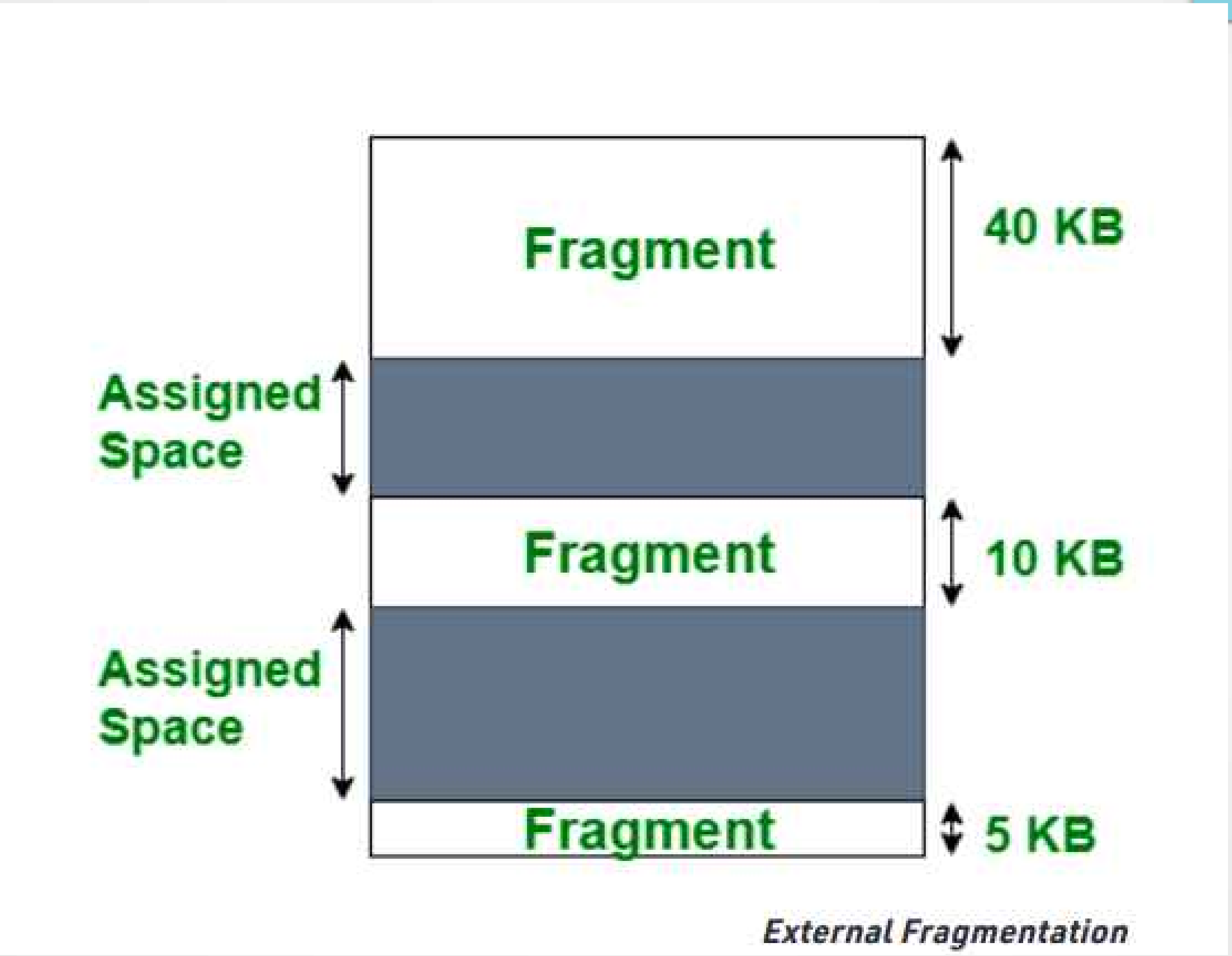
UNIT – 4 Basic Mamory Management



UNIT – 4 Basic Mamory Management

- **Fragmentation**
  - **External Fragmentation**
    - When allocating and de-allocating memory to the processes in partitions through various method.
    - It may possible that there are small spaces left in various partitions throughout the memory.
    - This memory space is known as **External Fragmentation.**

External Fragmentation



INTERNAL FRAGMENTATION VERSUS EXTERNAL FRAGMENTATION

INTERNAL FRAGMENTATION	EXTERNAL FRAGMENTATION
A form of fragmentation that arises when there are sections of memory remaining because of allocating large blocks of memory for a process than required	A form of fragmentation that arises when there is enough memory available to allocate for the process but that available memory is not contiguous
Memory block assigned to a process is large - the remaining portion is left unused as it cannot be assigned to another process	Memory space is enough to reside a process, but it is not contiguous. Therefore, that space cannot be used for allocation
Solution is to assign partitions which are large enough for the processes	Compaction or shuffle memory content is the solution to overcome this

Visit www.PEDIAA.com

UNIT – 4 Continuous Memory Allocation

- In older systems, **memory allocation is done by allocating a single contiguos area** in memory to the processes.
- But in multi -programming system, memory was divided **into two partitions.**
  - **One for the Os**
  - **Other for the User process**

UNIT – 4 Continuous Memory Allocation

- In Multi-user systems, more processes are accommodated by having multiple partitions in the memory.

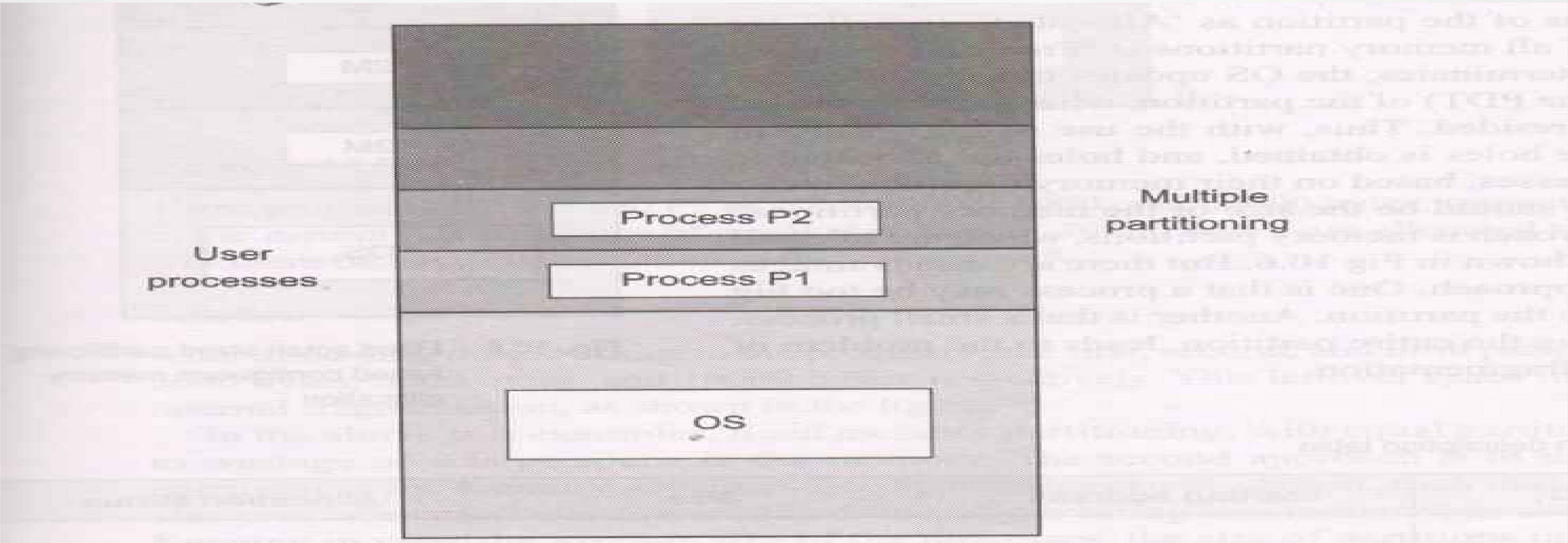


Fig. 10.5 Multiple partitioning of memory

UNIT – 4 Contiguous Memory Allocation

- Here process is allocated a contiuous memory in a single partition.
- Thus the memory partition, which fits the process, is searched and allocated.
- **The memory partition which is free to allocate, is known as a *hole.***
- When the process terminates, the occupied memory becomes free and the hole is available again.
- As soon as a process terminates, a hole becomes free, and is allocated to a waiting process.

UNIT – 4 Compaction

- **Compaction help to control memory wastage, occurring in dynamic partitioning.**
- The OS observes the number of holes in the memory and compacts them after a period, so that a contiguous memory can be allocated for a new process.
- The **compaction is done by shuffling the memory** contents, such that all occuupied memory region is moved in one direction, and all unoccupied memory region in the other direction.
- This results in contiguous free holes, as a single large hole.

UNIT – 4 Compaction

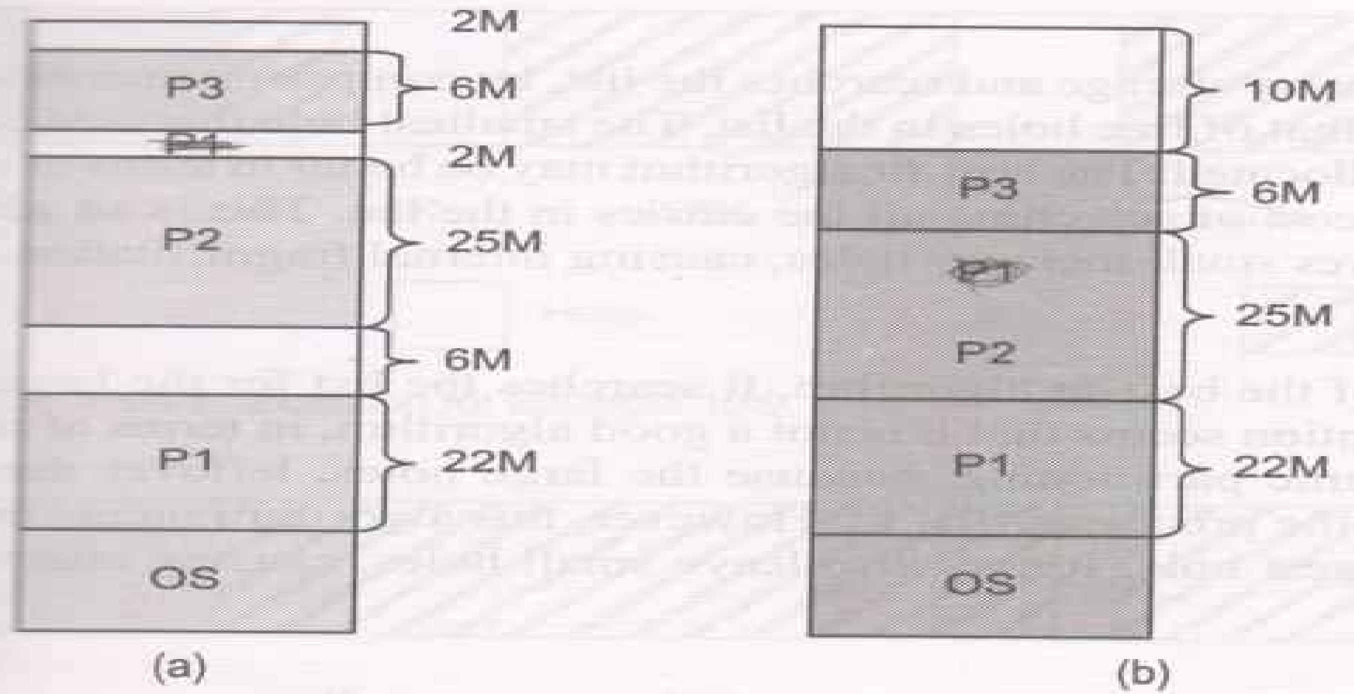


Fig. 10.9 Compaction

UNIT – 4 Memory Allocation Techniques

- Memory allocation techniques are algorithms that satisfy the memory needs of a process:
- They decide which hole from the list of free holes must be allocated to the process.
- Thus it is also known as partition selection algorithms.
- There are primarily three techniques for memory allocation
  - First-fit Allocation
  - Best-fit Allocation
  - Worst-fit allocation

UNIT – 4 Memory Allocation Techniques

- **First-Fit Allocation**
  - This algorithm searches the list of free holes and allocates the first hole in the list that is big enough to accommodate the desired process.
  - Searching is stopped when it finds the first fit hole.
- **Next -fit Allocation**
  - Searching is resumed from that location. The first hole is counted from this last location. In this case, it become the next-fit allocation.
  - First - Fit allocation does not take care of the memory wastage.

UNIT – 4 Memory Allocation Techniques

- **Best – Fit Allocation**
  - This algorithm takes care of memory storage and searches the list, by comparing memory size of the process to be allocated with that of free holes in the list.
  - **The smalled hole that is big enoughto accommodate the process is allocated.**
  - It is better interm of memory of wastege but it incure cost of searching.

UNIT – 4 Memory Allocation Techniques

- **Worst– Fit Allocation**
  - This algorithm is just reverse of the best-fit algorithm.
  - It search the list for the largest hole.
  - It is not good algorithm, in terms of memory, but it may be help ful in dynamic partitioning.

UNIT – 4 Memory Allocation Techniques

- Example :
- Consider the memory allocation scenario as next slide. Allocate memory for additional requests of 4k and 10k (in this order).
- Compare the memory allocation, using
  - First – fit Allocation
  - Best – fit Allocation
  - Worst – fit Allocations

UNIT – 4 Memory Allocation Techniques

- Problem :

Worst-fit allocation: It allocates the largest hole in the list. For the process of 4K, the larg
hole of 22K is allocated. This leaves a hole of 18K in the memory. The next request is of

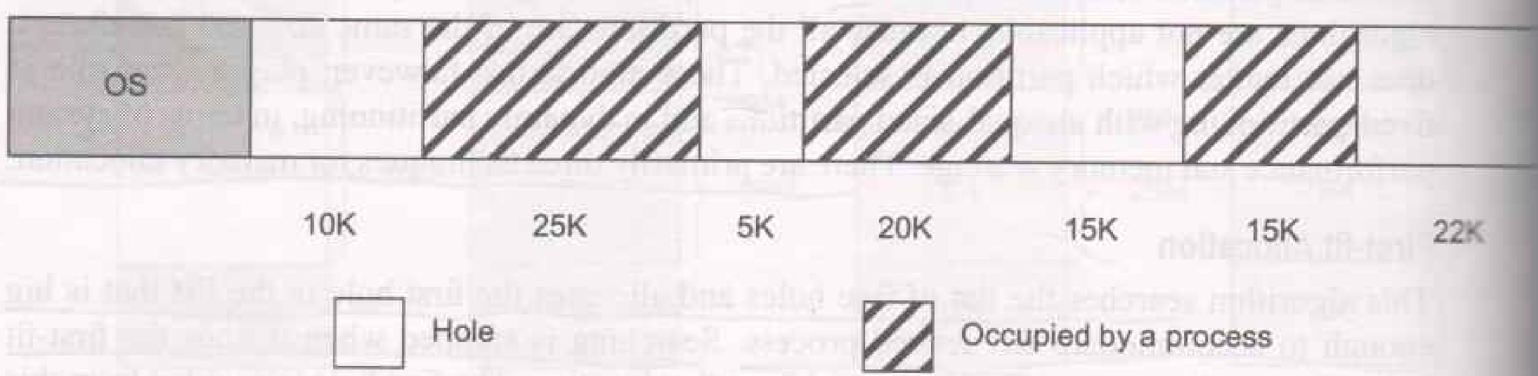


Fig. 10.10 Example memory allocation scenario