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**

UNIT – 4 Basic Mamory Management

- **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.

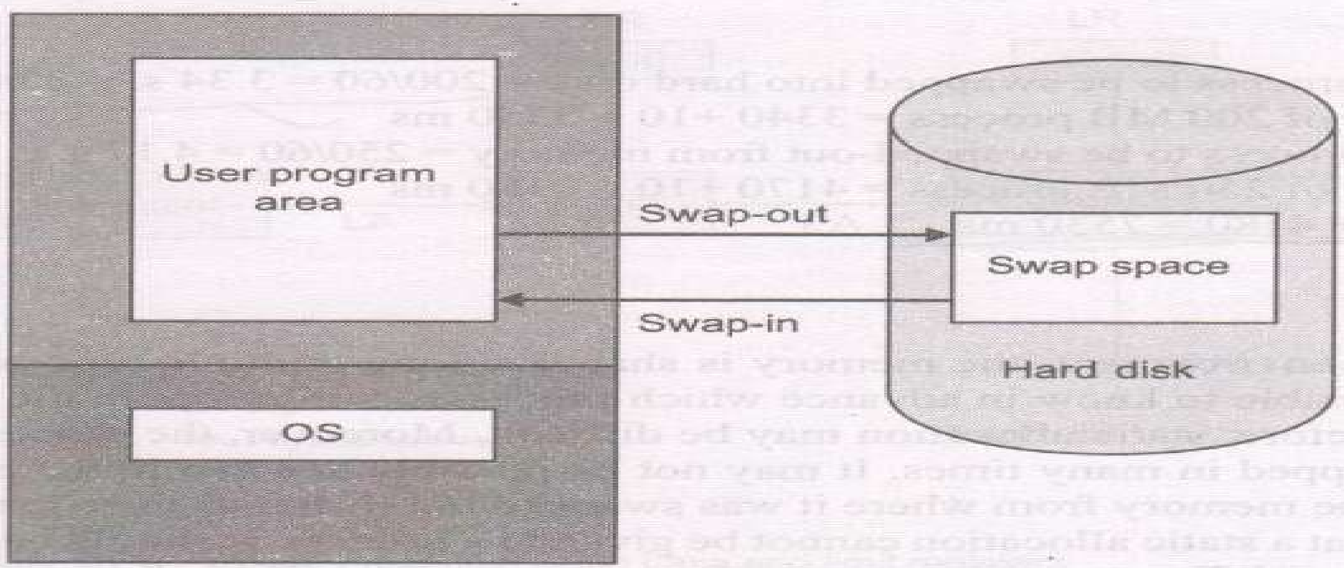


Fig. 10.1 Swapping

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

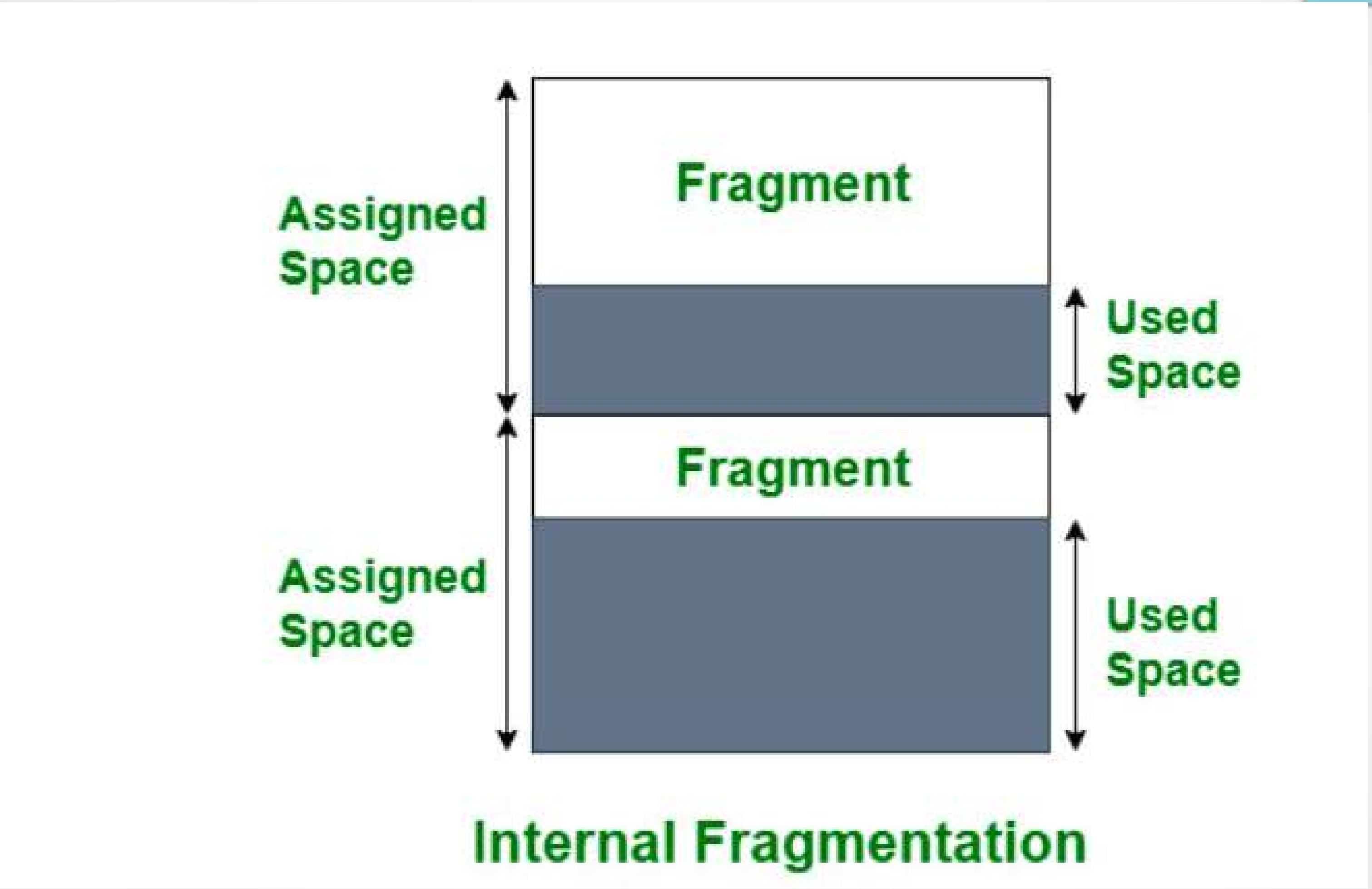
UNIT – 4 Basic Mamory Management

- **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.**

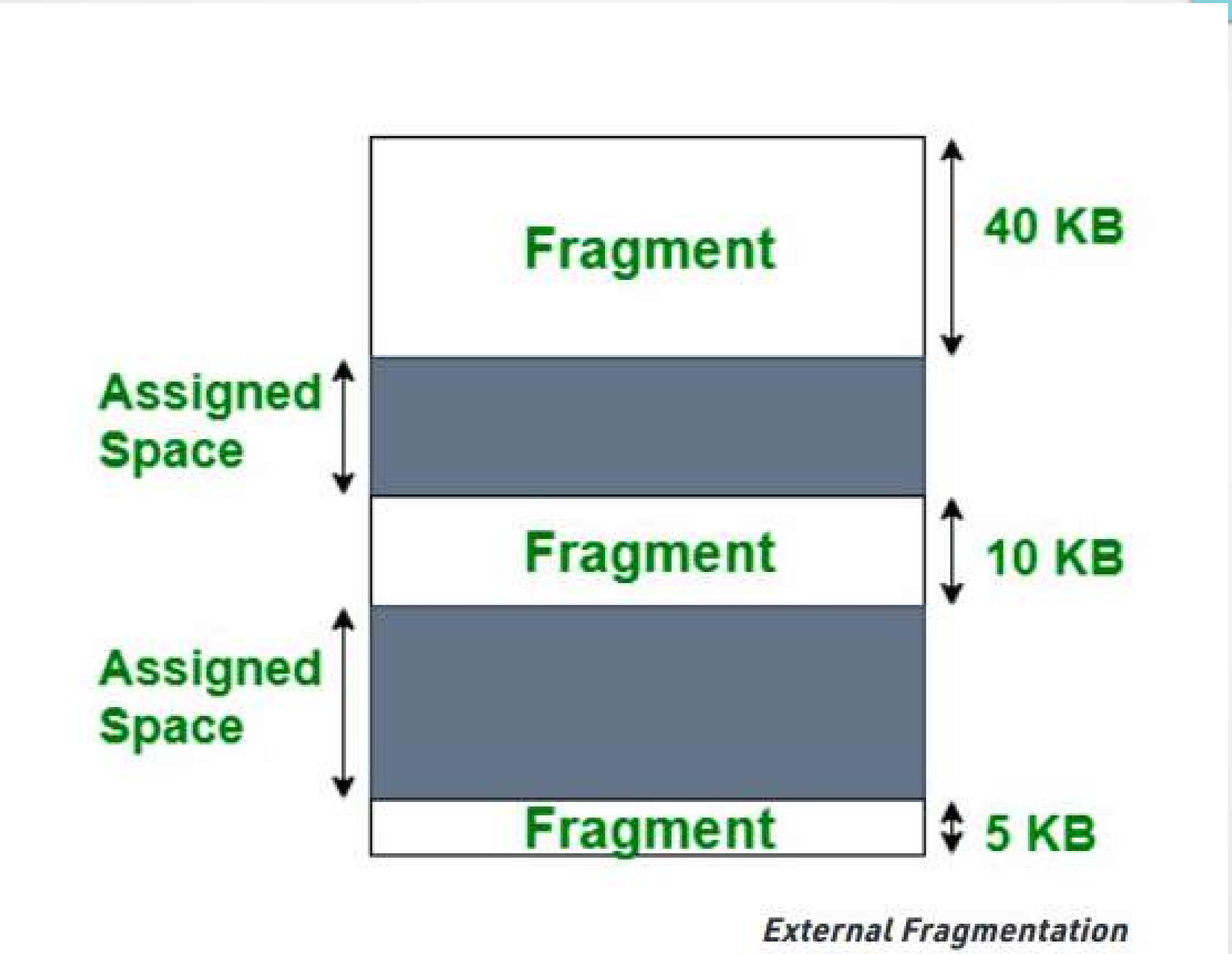
UNIT – 4 Basic Mamory Management



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 i**nto 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.

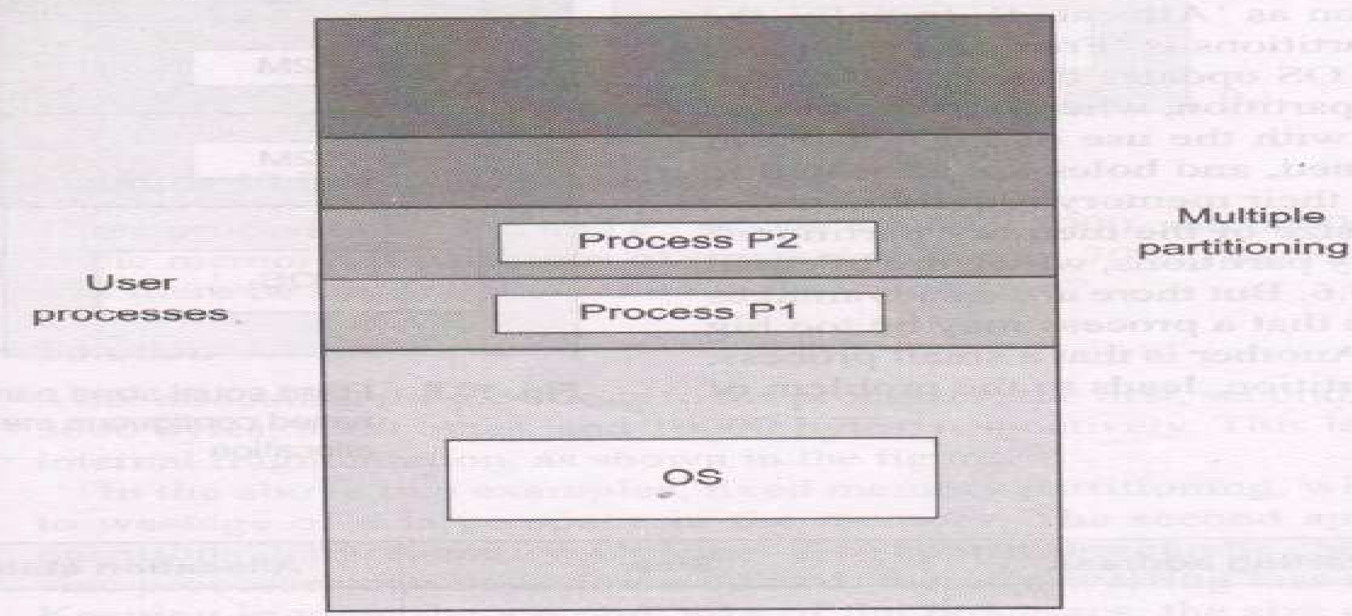


Fig. 10.5 Multiple partitioning of memory

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

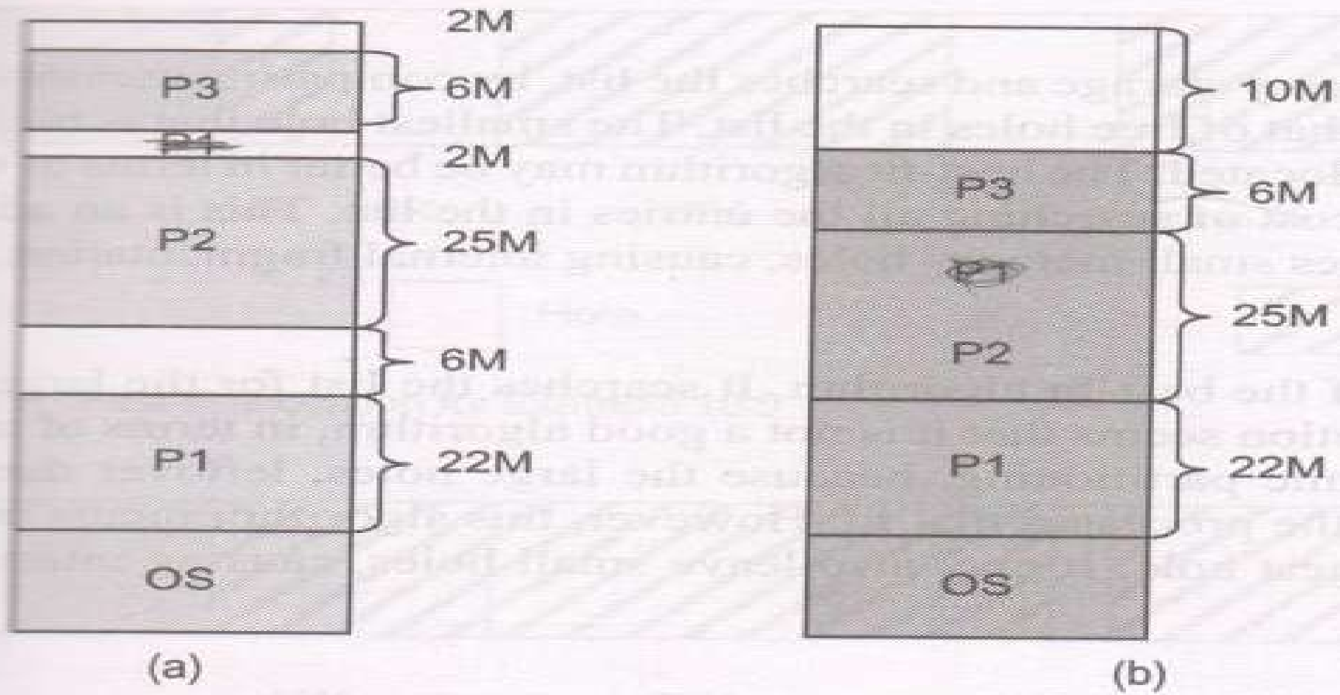


Fig. 10.9 Compaction

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 1

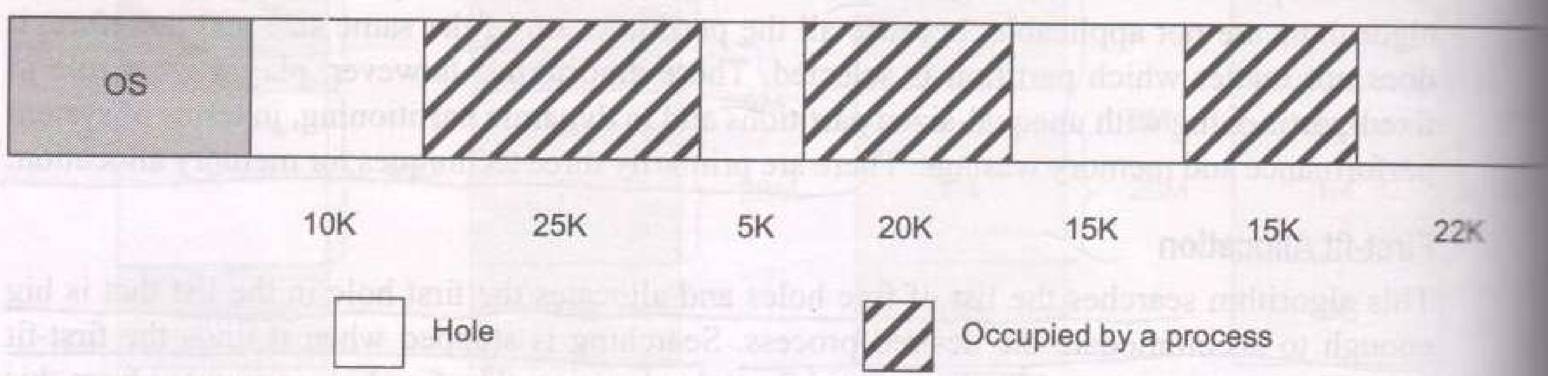


Fig. 10.10 Example memory allocation scenario

UNIT – 4 Paging Concept

- The first **non-contiguous** memory allocation method is paging.
- In this memory is divided into equal size partitions.
- The partitions are relatively smaller, compared to the contiguous method. They are known as **frames.**
- The logical memory of a process is also divided into small chunks or blocks of the same size as frame. These chunks are **called pages** of a process.
- Pagining is a logical concept that divides the **logical address space** of a process into fixed size **pages**, and is implemented in **physical memory** through **frame.**

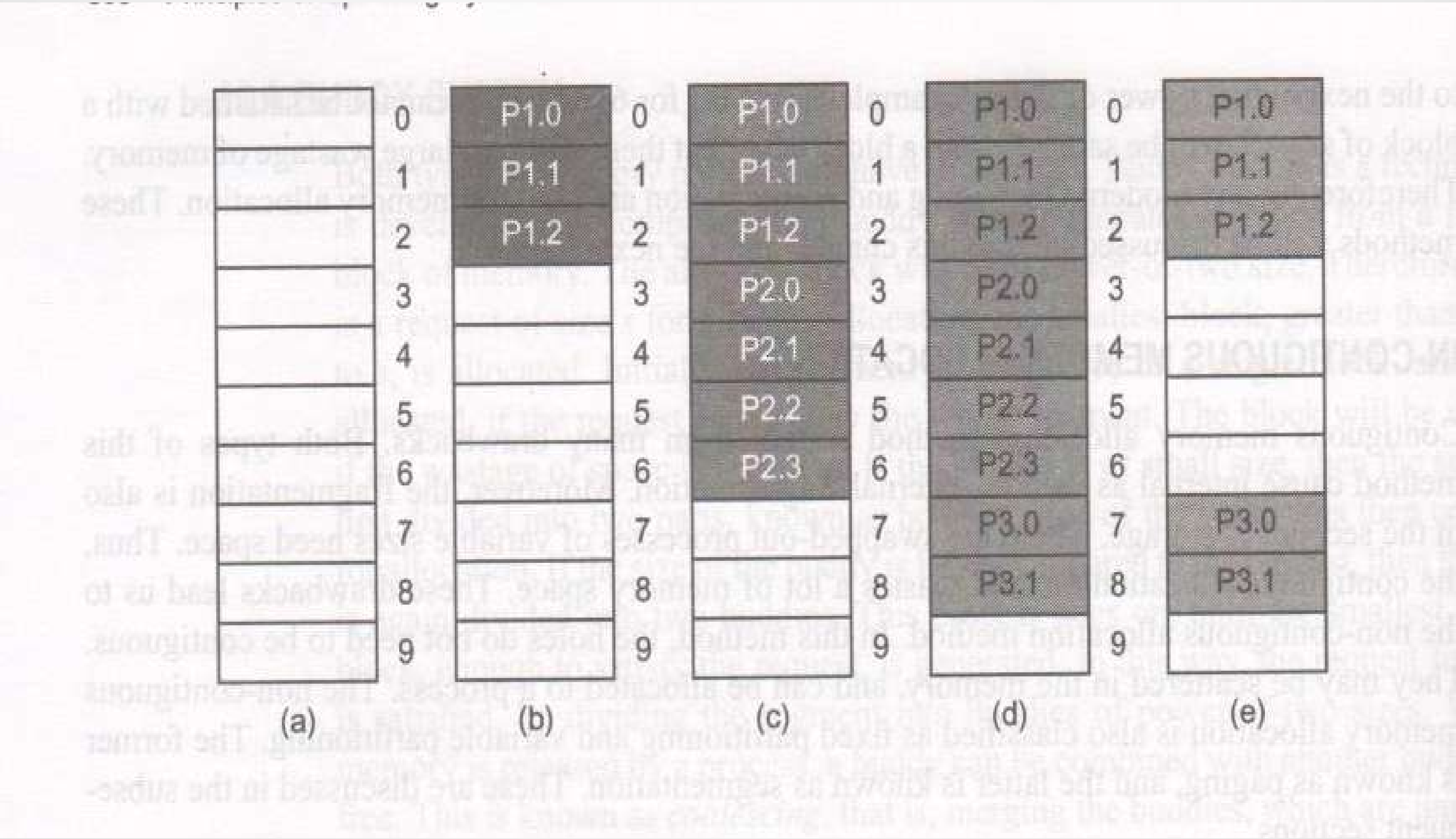
UNIT – 4 Paging Concept example

- There are 10 free frames in the memory.
- There are 4 processes P1, P2 ,P3, P4 consisting of 3, 4, 2, 5 pages respectively.
- For P1 (fig 10.15(b))
- For P2 (fig 10.15(c))
- For P3 (fig 10.15(d))
- Now only one frame is free in the memory, where P4 required 5 frames.
- After some time P2 finishes its exectuion and therefore, release memory. These five frames, through non contiguous are alocated to P4 (fig 10.15(e ,f)).

UNIT – 4 Paging Concept example

- Suppose after some time P1 release page 1, P4 release page 2 and P3 releases page 1 (fig 10.15(g))
- Now P5 is introduced in the system with 5 pages, but only 3 pages to be accommodated in the memory (fig 10.15(h))

UNIT – 4 Paging Concept example



UNIT – 4 Paging Concept example

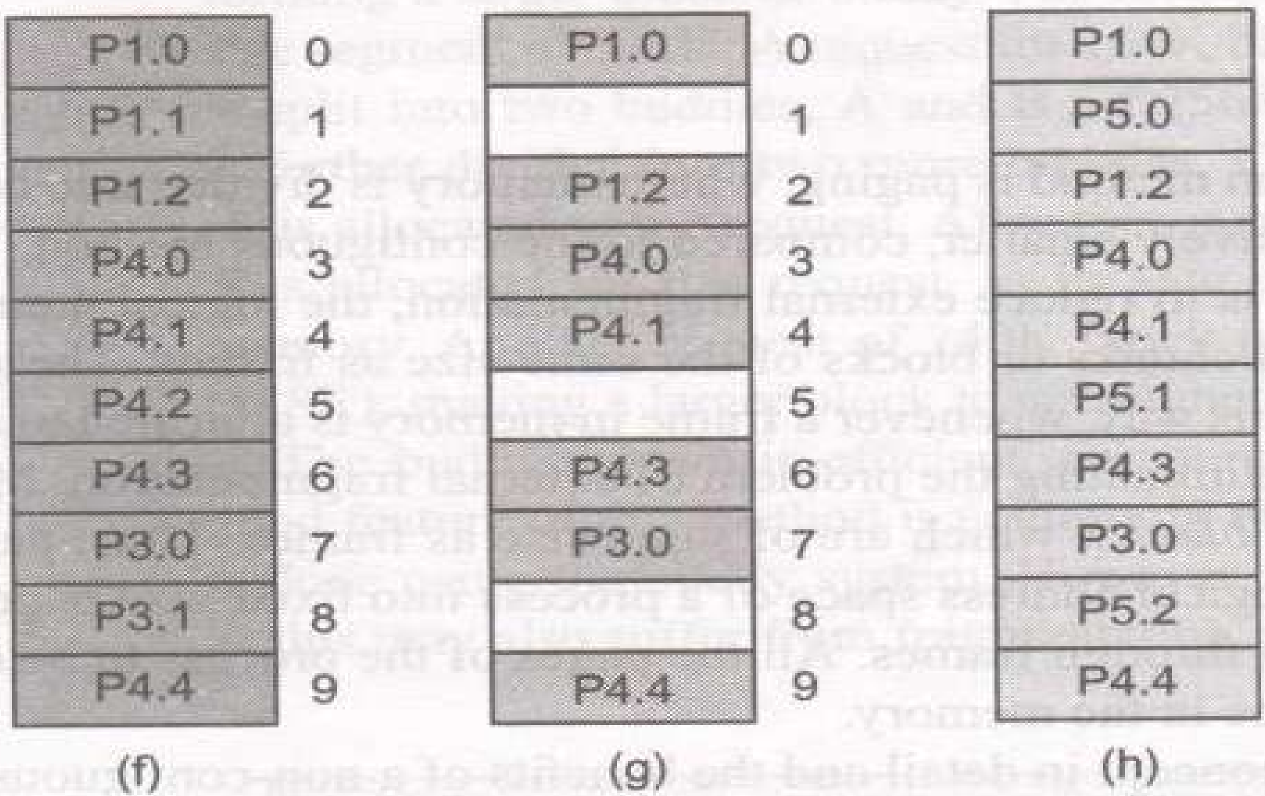
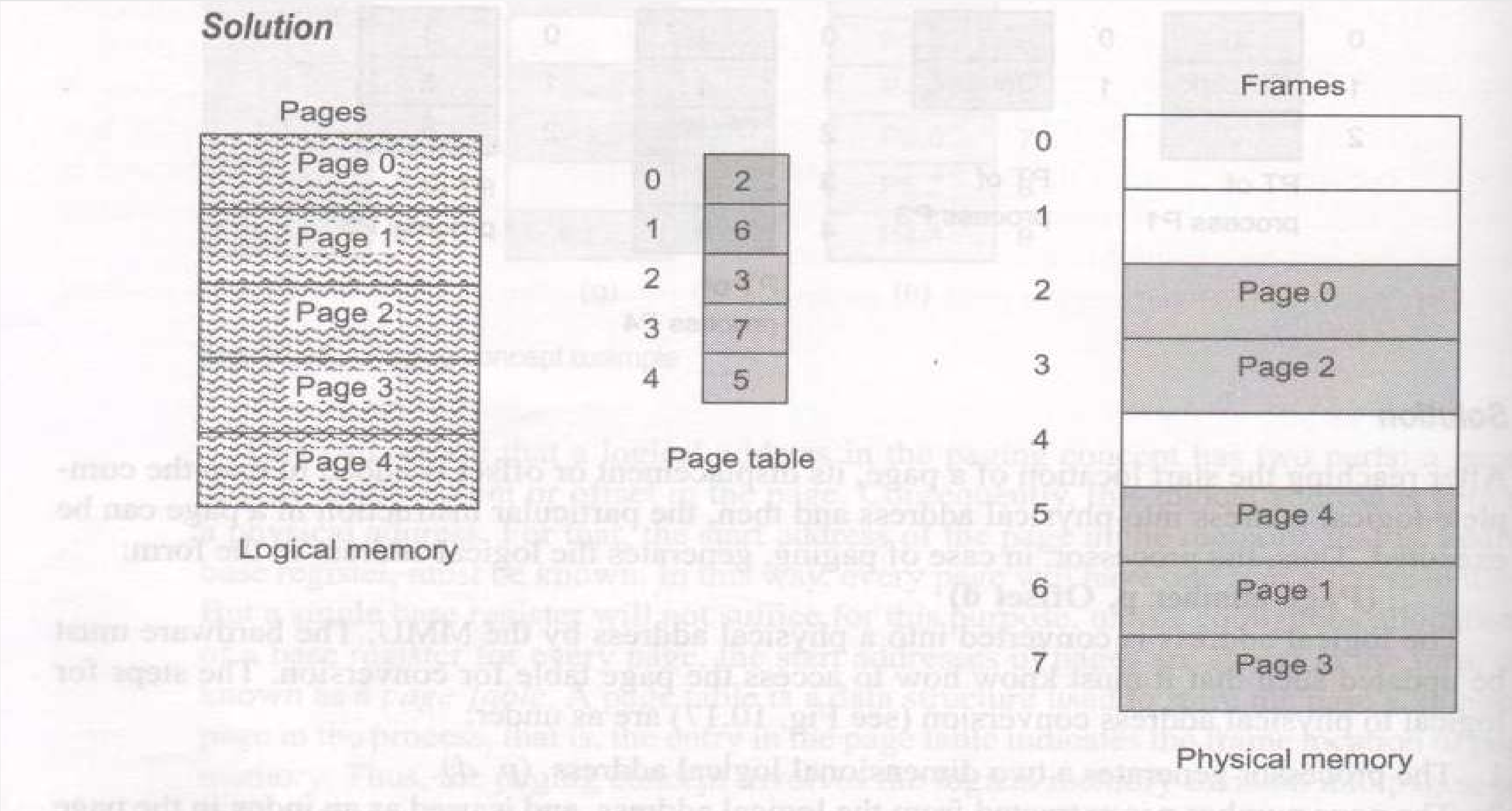


Fig. 10.15 Paging concept example

UNIT – 4 Paging Concept example

- A program's logical memory has been divided into 5 pages and these pages are allocated frames 2, 6,3, 7 and 5.
- Show the mapping of logical memory to physical memory.

UNIT – 4 Paging Concept example



UNIT – 4 Segmentation

- A programmer writes programs not in terms of pages, but modules, to reduce the problem complexity.
- There may be modules : main program, procedures, stacks, data etc.
- **It would be better if memory management is also implemented in terms of these modules.**
- **Segmentation is a memory management technique that supports the concept of modules. The modules in this technique are called segments.**
- The segements are logical divisions of a program, and they may be of different sizes, where as pages in the paging concept are physical divisions of program and are of equal size.

UNIT – 4 Segmentation

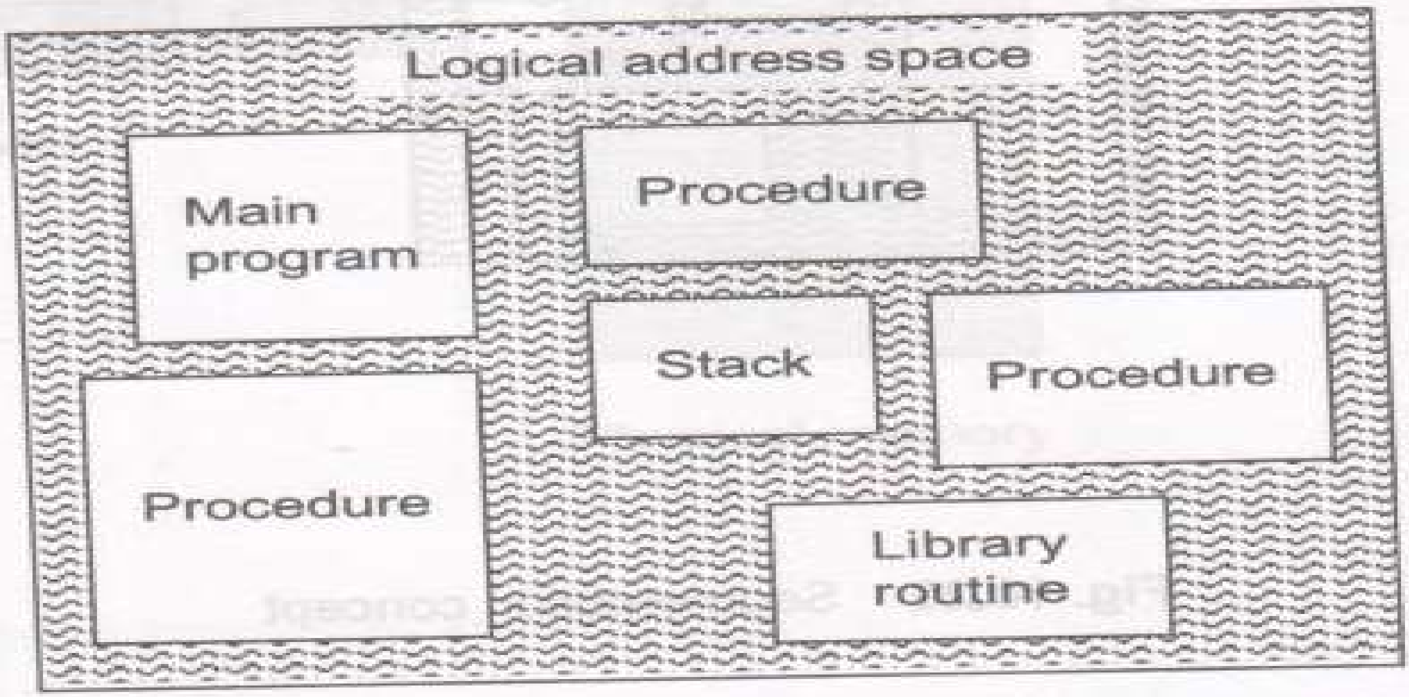


Fig. 10.28 Logical address space divided into segments

UNIT – 4 Segmentation

- Segementation has two advantages:
  - Segementation has logical memory which is **closer to a programmers way of thinking**
  - The segmentation need **not be of the same size** as compared to pages.

UNIT – 4 Segmentation

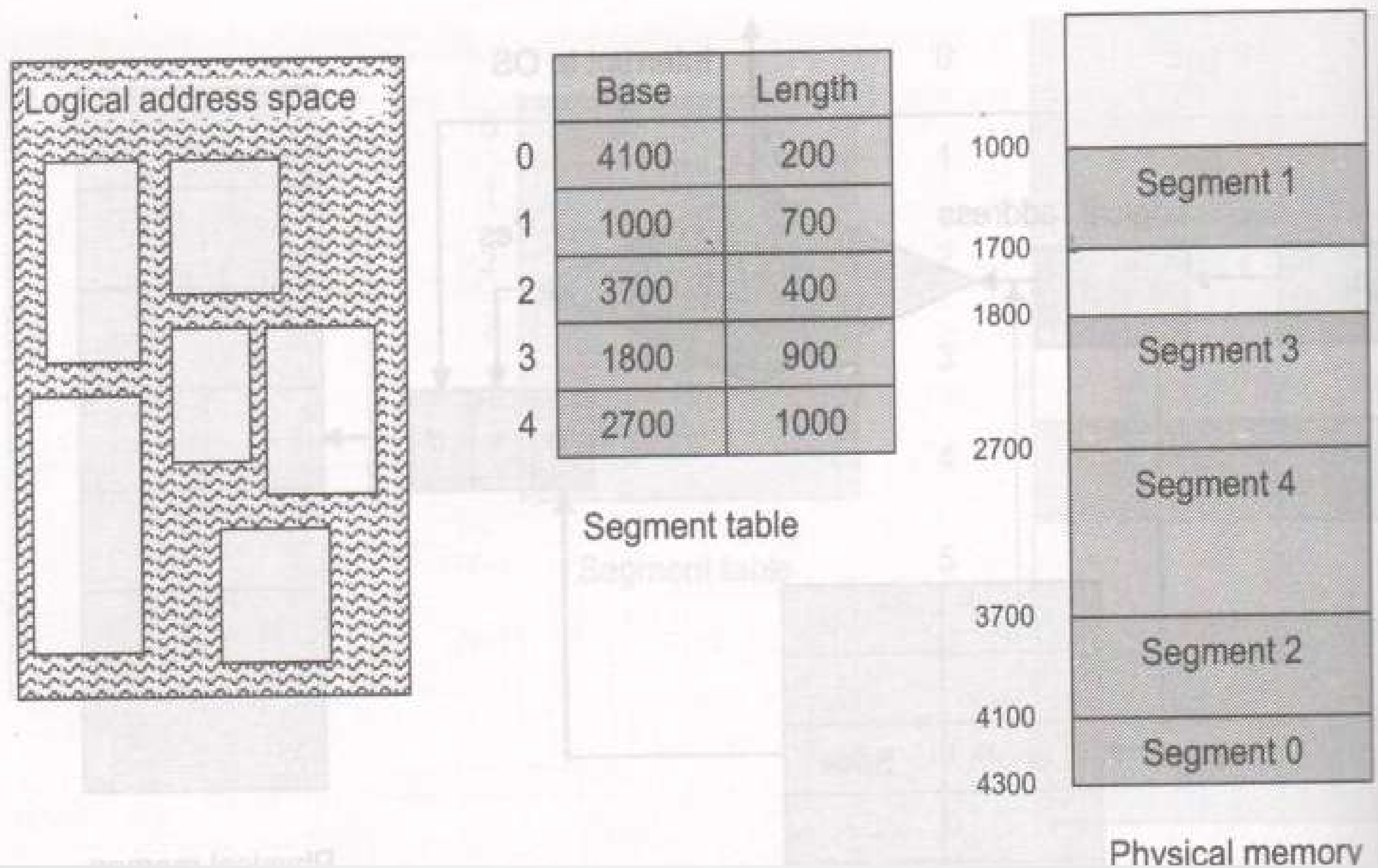
- Segementation logical addreses:
  - Logical address of segement has two part
    - **The segment name (or Segment Number)**
    - **Its offset**
  - It has three major segment
    - Code segment
    - Data segment
    - Stack segment

UNIT – 4 Segmentation

- To convernt logical address of segment in to physical address use **Segment tabel.**
- **Example :**

Segment Number	Length/ limit	Base Address
0	200	4100
1	700	1000
2	400	3700
3	900	1800
4	1000	2700

UNIT – 4 Segmentation



UNIT – 4 Virtual Memory

- Virtual memory is used **when process size is too large to fit in the real memory**, therefore virtual memory is created.
- In virtual memory, **combined approach of paging and segmentation is used.**
- Virtual memory implementaion **is complex as compared with real memory.**
- It needs the assistance of hardware support known as **paging hardware.**
- Also OS have a module known as **virtual memory handler (VM Handler).**

UNIT – 4 Need of Virtual Memory

- **Paging and segmentation are two basic memory management techniques** that require an entire process to reside in the main memory before its exectution.
- The increase in the degree of multi-programming means that **more number of processes should be accommodated in the memory**.
- But the degree of **multi programming is limited with the size of the memory**. This limitaiton may lead to several problems.

UNIT – 4 Overlay

- The first solution was in the form of **Overlay, years ago.**
- **An overlay is a portion of a process.**
- A **program is first divided into many overlays** and store in the disk.
- A program containing overlaysis **called an overlay structure program.**
- This program consists of a set of overlay and a permanently **resident portion known a root.**
- As the root executes, the overlays are loaded as and whenever required.

UNIT – 4 Overlay

- The required **overlays are swapped in the memory** and later on **swapped out** when the memory is full.
- Moreover, today, overlay is an obsolete technique.

UNIT – 4 Virtual Memory

- Due to Overlay is obsolete, it **gives rise to the concept of virtual memory in modern system.**
- **Virtual memory is a method that manage the exceded size of larger processes as compared to the available space in the memory.**
- It means the degree of multi-programming can be increased without worrying about the size of the memory.

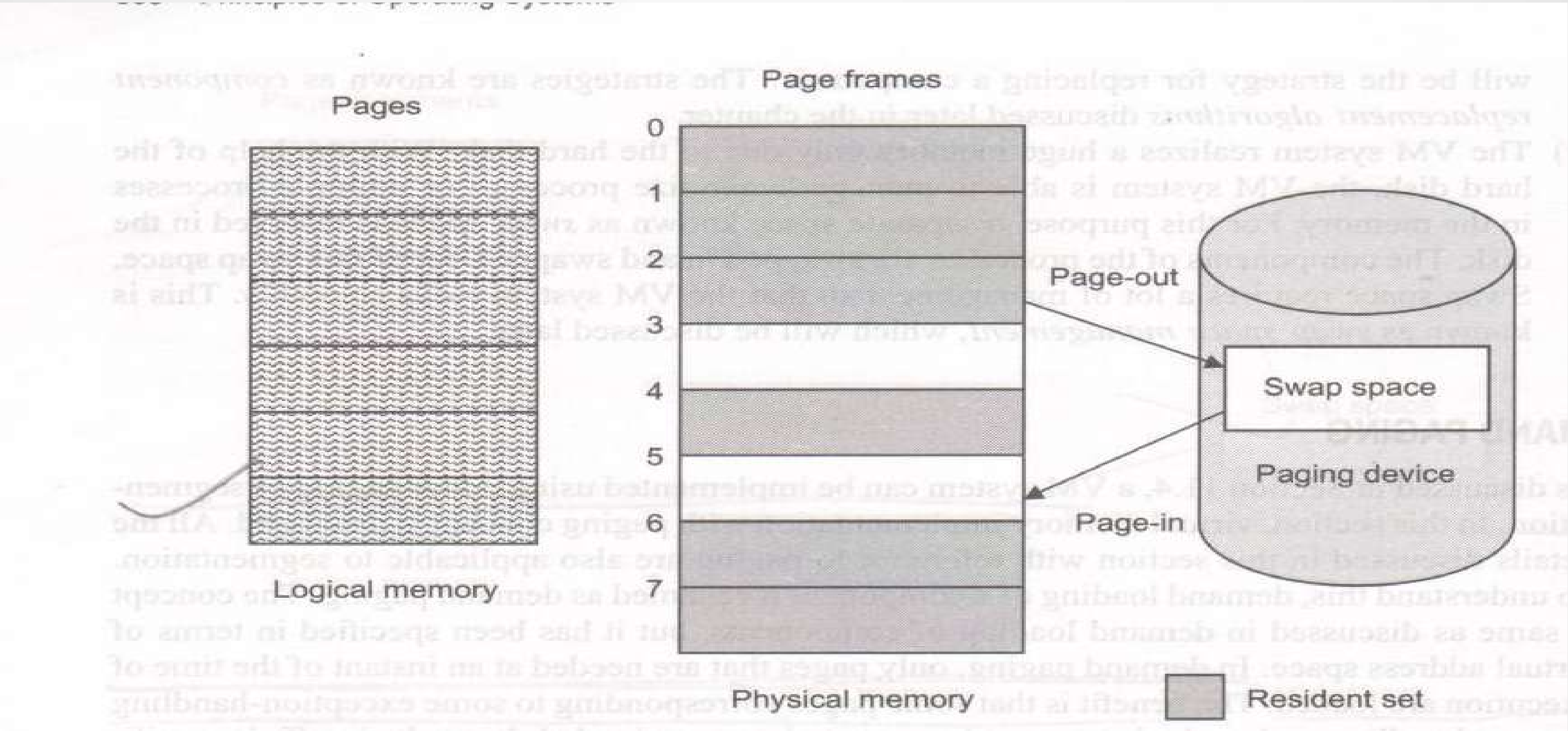
UNIT – 4 Virtual Memory – Demand Paging

- VM system can be implemented **using either Paging or segmentation.**
- In **Demand Paging, only pages that are needed at an instant of the time of execution are loaded.**
- The benefits is that some pages, corresponding to some exception – handling or error- handling code, which may not be executed, are not loaded.
- It results in efficient utilization of memory and efficient execution.

UNIT – 4 Virtual Memory – Demand Paging

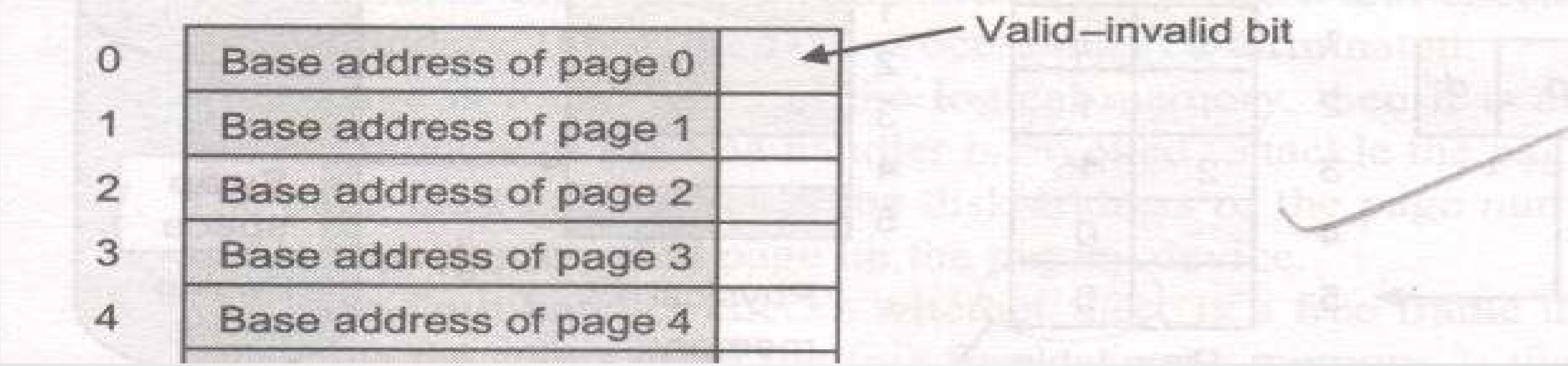
- **Demand Paging is same as Swapping.**
- Except that an entire process is not swapped in or swapped out.
- Here, a **Lazy swapper** is used that loads only those pages that are needed.
- So here insted of “swap -in” is called “page-in”
- And “swap – out” is called “page-out”

UNIT – 4 Virtual Memory – Demand Paging



UNIT – 4 Virtual Memory – Demand Paging

- Demand Paging has some issues
  - **(1)** how to recognize whether a page is present in the memory.
    - The page table with valid-invalid bit can be used for this purpose.
    - Valid bit -> “1” page is memory at the time
    - Invalid bit -> “0” page is either not valid or not present in the memory.



UNIT – 4 Virtual Memory – Demand Paging

- Demand Paging has some issues
  - **(2)** it is a situation when a process execution does not get a page in the memory.
    - A situation will occur in demand paging when the page referenced is not present in the memory.
    - This is known as a **Page fault.**

UNIT – 4 Virtual Memory – Page Replacement Algorithms

- When a page fault occurs during the execution of a process, a page needs to be paged into the memory from the disk.
- However, it may be the case that **there is no free frame in the memory. In such case, an already existing page should be replaced so that there is room** for a page that needs to be paged. **This is known as a page replacement.**
- If the page replaced by a random approach, then it may affect the performance.
- Thus, instead of replacing any page, the use of pages in the memory is to be observed and a page should be replaced such that effect on performance is the least.

UNIT – 4 Virtual Memory – Page Replacement Algorithms

- The page replacement increases the overhead because there are two page transfer
  - **Page – in & Page – out**
- This over heads can reduced if it is known whether a page has been modified.
- **Any instance of time it is not necessary every pages is modified and also some page are read only.**
- In this **case simply be over written by** another page becase its copy is already on the disk. So one page – transfer time can be reduced.
- This is implemented by including **M – bit or Dirty bit** with each page.

UNIT – 4 Virtual Memory – Page Replacement Algorithms

of replacing any page, the use of pages in
replaced such that the effect on performan
the best page to be replaced in the memory

	Valid–Invalid Bit	M-bit
Base address of page 0		
Base address of page 1		
Base address of page 2		
Base address of page 3		
Base address of page 4		
Base address of page 5		

Fig. 11.6 Page table with valid–invalid and M-bits

T
there
fore,
servi
head
been
mem
Som
be. N
has r
then
writt

UNIT – 4 Virtual Memory – Page Replacement Algorithms

- If the page is modified, then need to implement the page replacement algorithms.
- A page replacement algorithms must satisfy the following requirements:
  - The algorithms must not replace a page that many be referenced in the near future. **It is known as non-interference with the program's locality of reference.**
  - The PFR should not increase with an increase in the size of the memory.

UNIT – 4 Virtual Memory – Page Replacement Algorithms

- Types of Page Replacement Algorithms
  - **FIFO** (First in First out Page Replacement Algo)
  - **LRU** (Least Recently Used Page – Replacement Algo)

UNIT – 4 FIFO Page Replacement Algorithm

- According to FIFO, the oldest page among all the pages in the memory is chosen as the victim.
- **All the page in the memory in a FIFO queue. The page at the head of the queue will be page – out first and a new page will be inserted at the tail of the queue.**

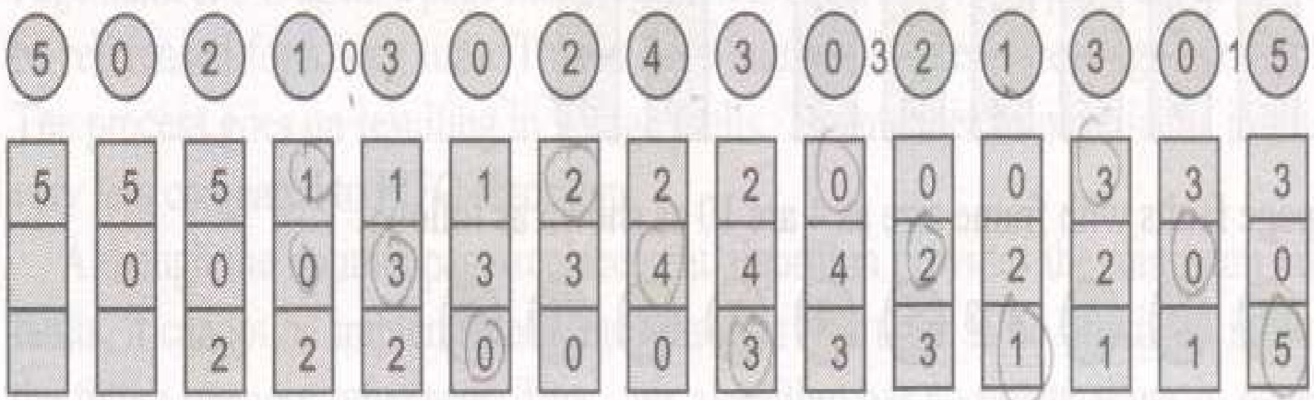
UNIT – 4 FIFO Page Replacement Algorithm

Example 11.6

Calculate the number of page faults for the following reference string using FIFO algorithm with frame size as 3.

5 0 2 1 0 3 0 2 4 3 0 3 2 1 3 0 1 5

Solution

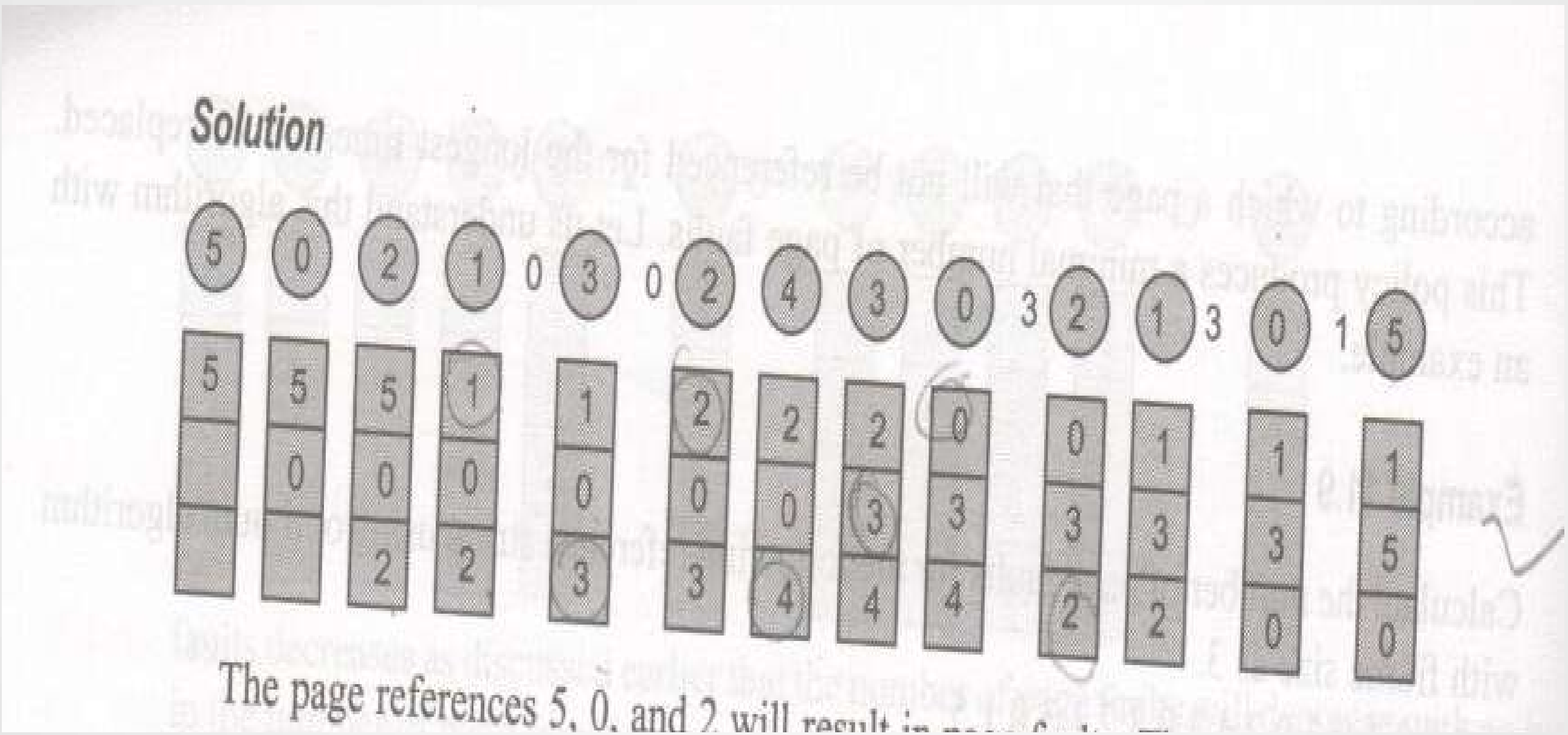


Initially, all the three frames are empty. Page number 5 is first referenced, and it is a page fault.

UNIT – 4 LRU Page Replacement Algorithm

- In LRU, a page the that has not been referenced for a long time in the past may not referenced for a long time in the future either.
- LRU page – replacement algorithm replaces a page that has not been used for the longest period of time in the past.

UNIT – 4 LRU Page Replacement Algorithm



UNIT – 4 LRU Page Replacement Algorithm

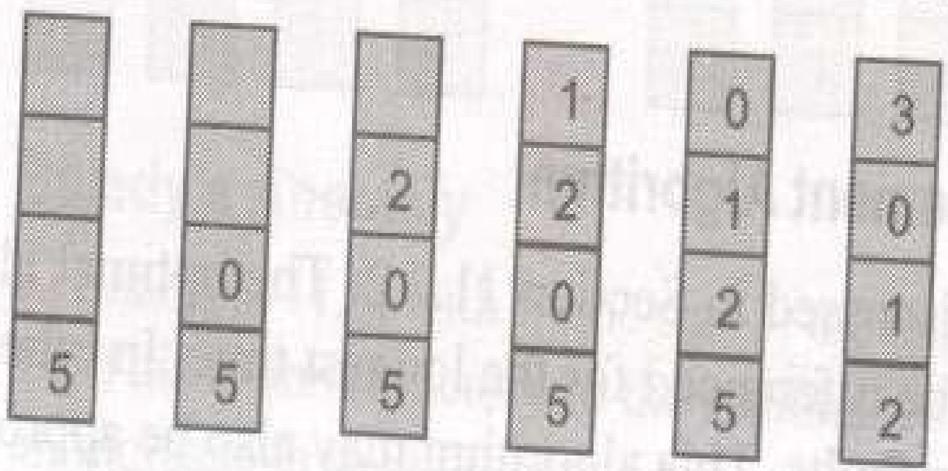
- **Stack Implementation**
  - To implement LRU, a linked list of all the pages in the memory can be maintained.
  - The list can be structured as a stack such that **whenever a page is referenced, it is placed at the top of the stack.**
  - This way, the **most recently used page will always be at the top** and consequently, the **least recently used page will be at the bottom** of the stack.
  - This implementation requires removing one entry from the middle and placing it at the top of the stack.
  - The stack needs to be updated with every memory reference, which incurs a cost.

UNIT – 4 LRU Page Replacement Algorithm

A page reference string is given by

5 0 2 1 0 3 0 2 4 3 0 3 2 1 3 0 1 5

The stack implementation that records the most recent refere
page reference at the bottom of the stack. The following figure s
the first reference of Page number 3 in the reference string.



Counter Implementation

UNIT – 4 Thrashing

- In the VM, if there is **no free frame in the memory** and all the **pages currently in the memory are referenced** frequently, then **an active page will be replaced** to bring in the desired pages.
- When an **active page is replaced, it will be needed again, right away resulting in a page fault.**
- After some time, the processes will try to replace the active pages of another process to get a free frame in the memory causing a large number of page fault.
- This is known as **high paging activity**, and high paging activity is known as **thrashing.**

UNIT 4 Completed