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.**

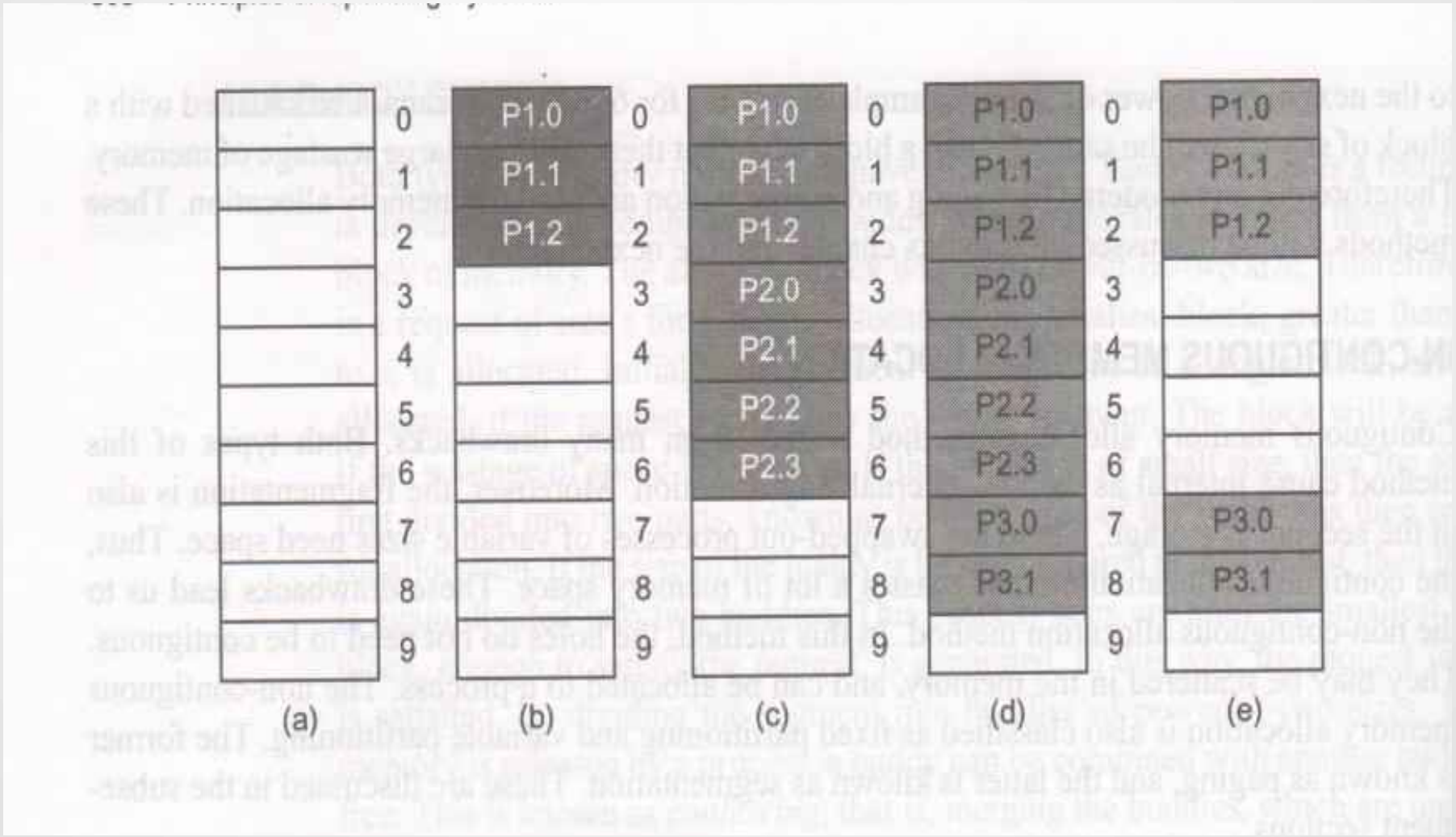
UNIT – 4 Paging Concept example

- 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

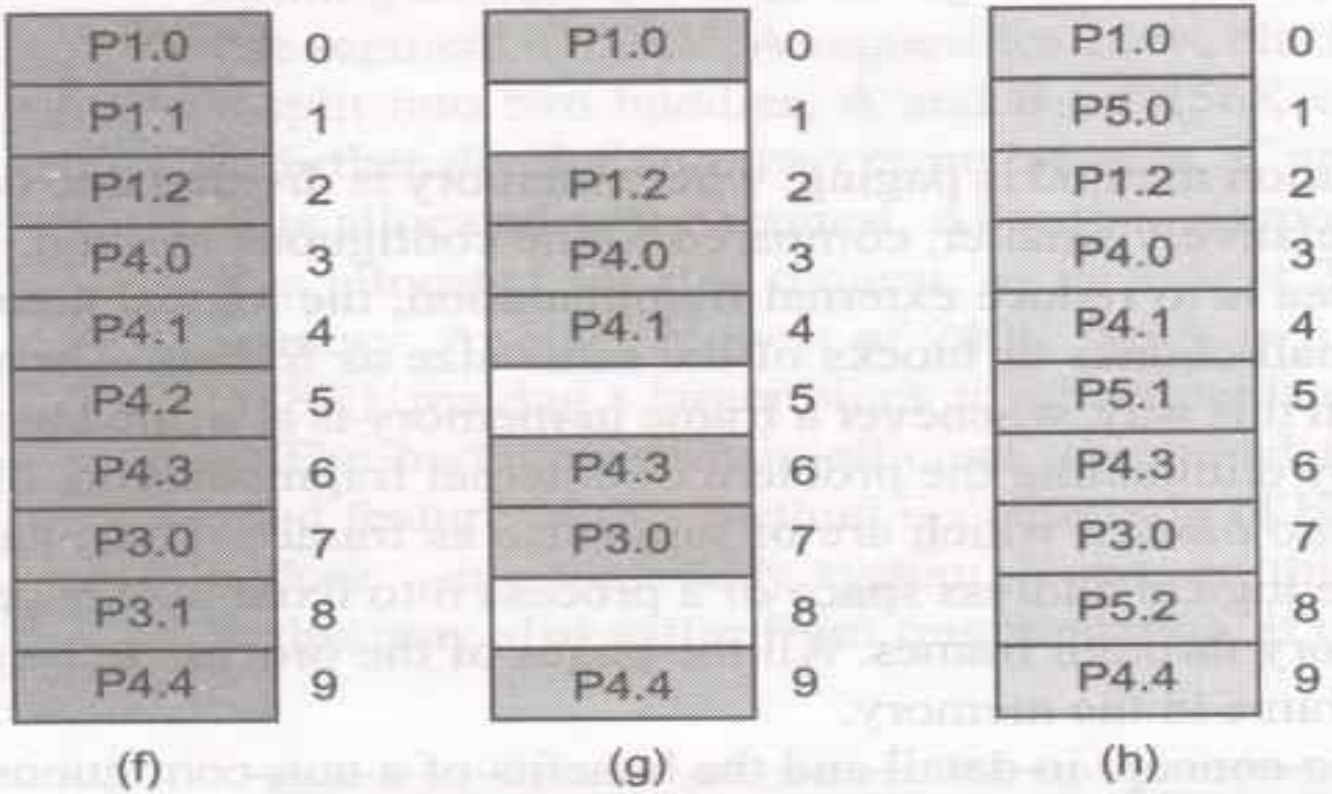
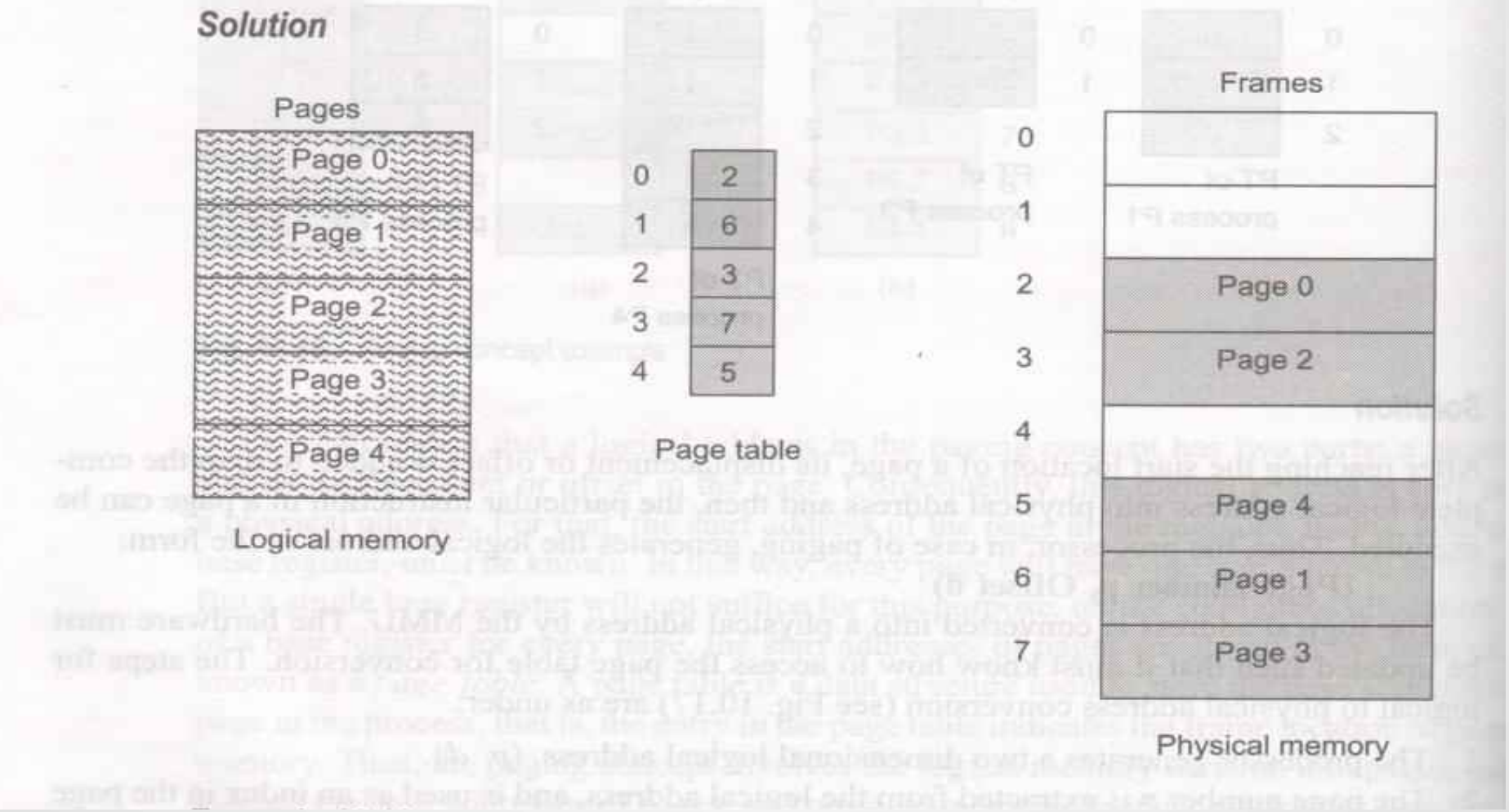


Fig. 10.15 Paging concept example

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

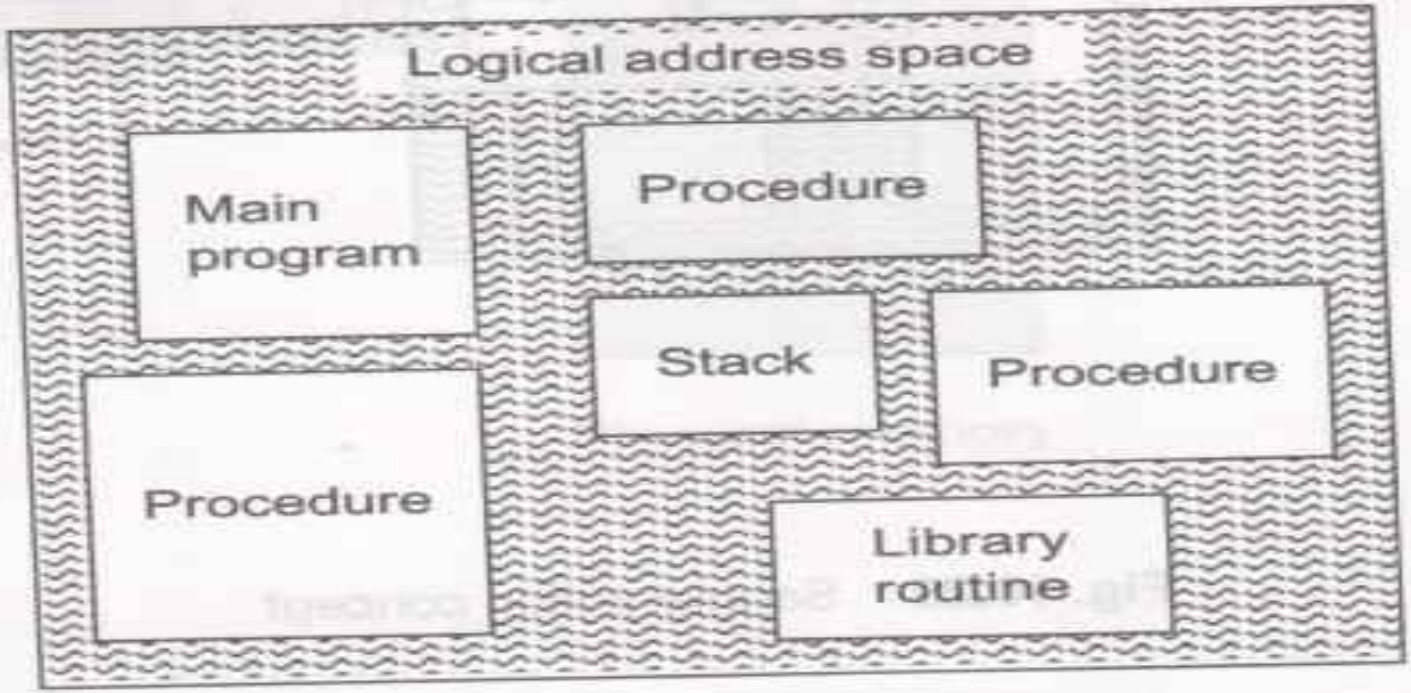


Fig. 10.28 Logical address space divided into segments

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

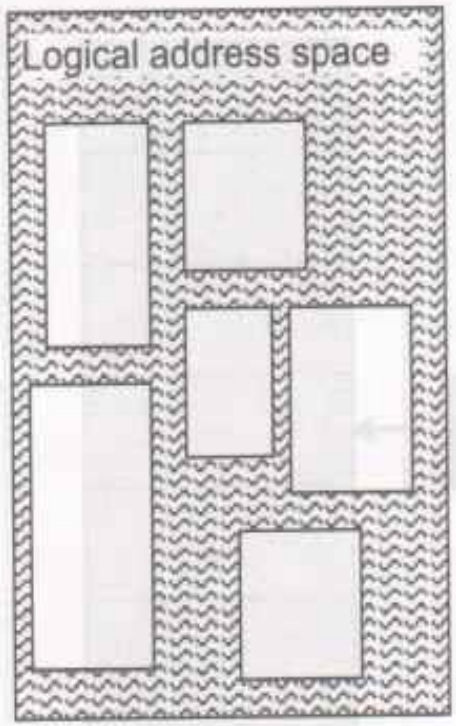
- Segementation logical addreses:
  - Logical address of segement has two part
    - **The segment name (or Segment Number)**
    - **Its offset**
  - It has three major segment
    - Code segment
    - Data segment
    - Stack segment

UNIT – 4 Segmentation

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

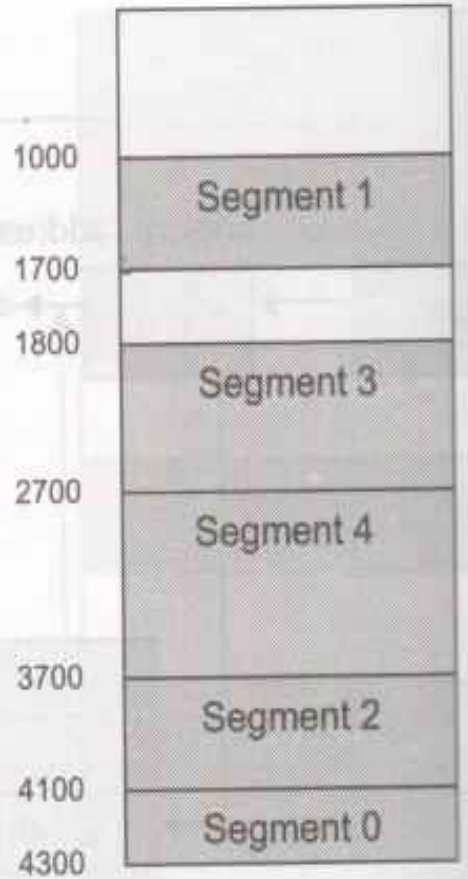
Segment Number	Length/ limit	Base Address
0	200	4100
1	700	1000
2	400	3700
3	900	1800
4	1000	2700

UNIT – 4 Segmentation



	Base	Length
0	4100	200
1	1000	700
2	3700	400
3	1800	900
4	2700	1000

Segment table



Physical memory

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.

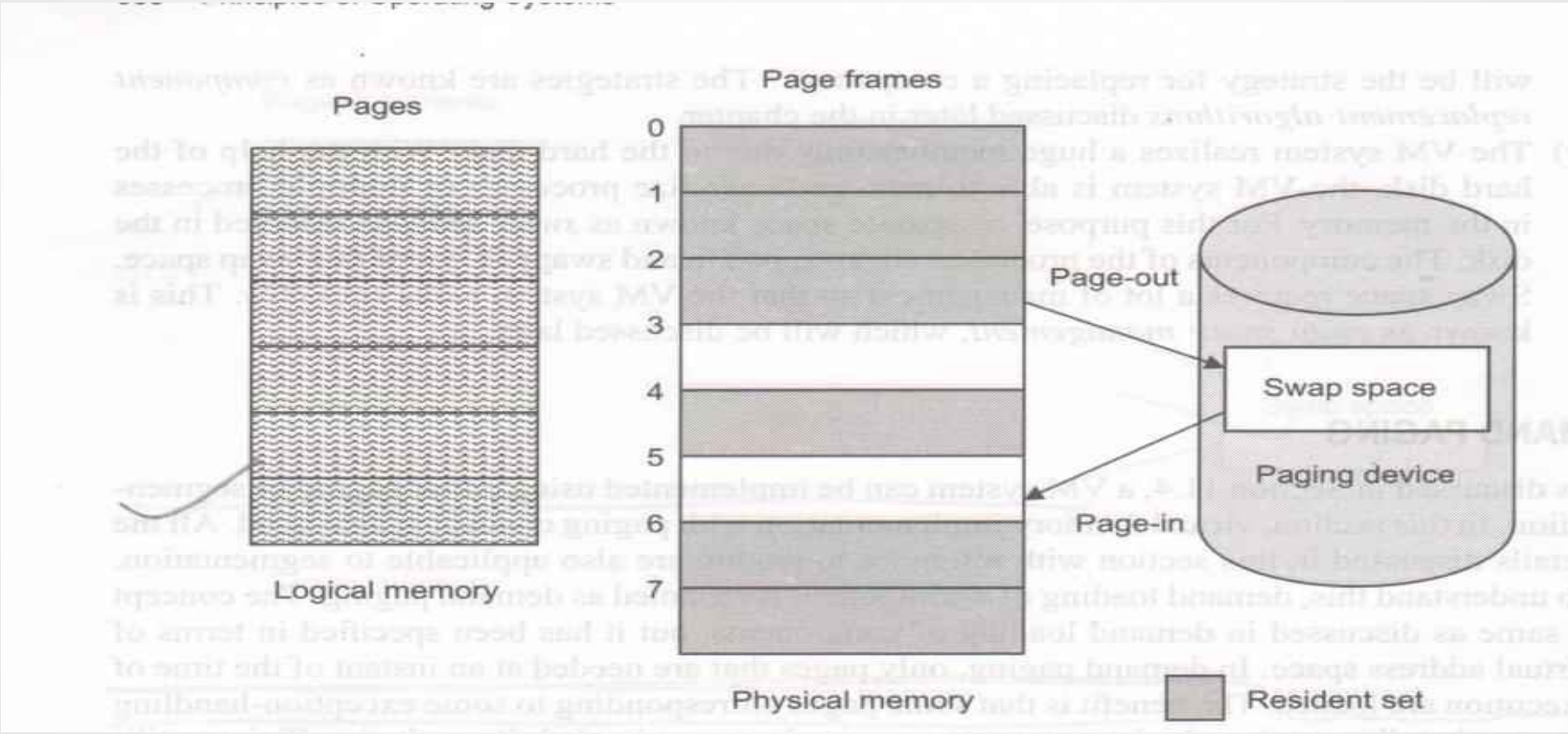
UNIT – 4 Virtual Memory – Demand Paging

- 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

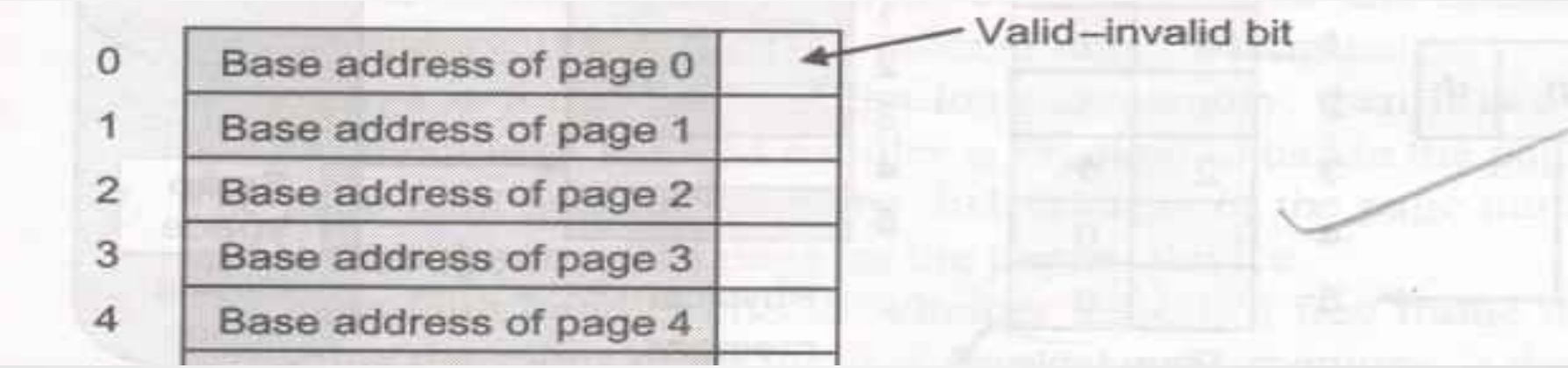
- **Demand Paging is same as Swapping.**
- Except that an entire process is not swapped in or swapped out.
- Here, a **Lazy swapper** is used that loads only those pages that are needed.
- So here insted of “swap -in” is called “page-in”
- And “swap – out” is called “page-out”

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

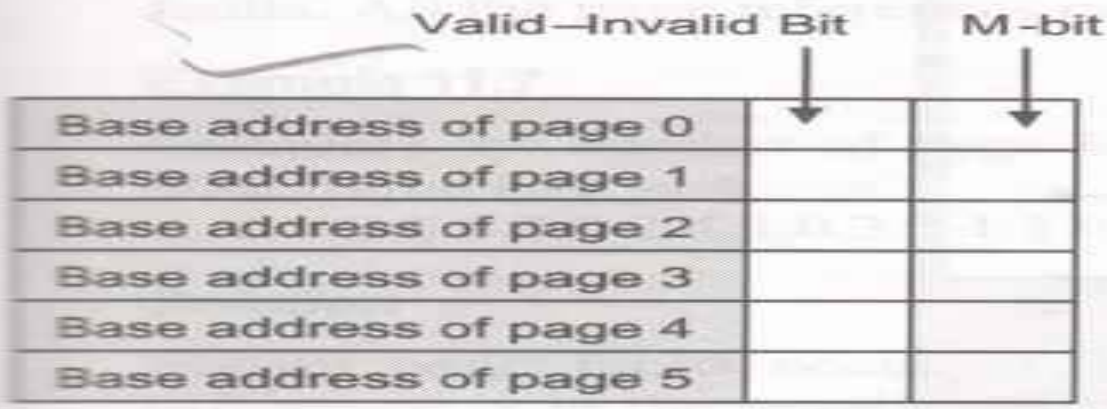
- When a page fault occurs during the execution of a process, a page needs to be paged into the memory from the disk.
- However, it may be the case that **there is no free frame in the memory. In such case, an already existing page should be replaced so that there is room** for a page that needs to be 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



11.6 Page table with valid–invalid and M-bits

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.**

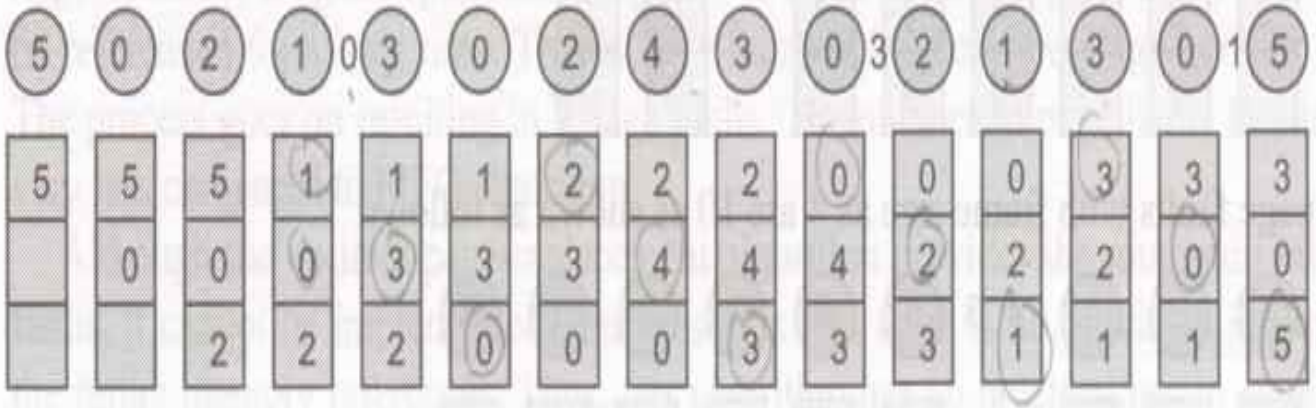
UNIT – 4 FIFO Page Replacement Algorithm

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

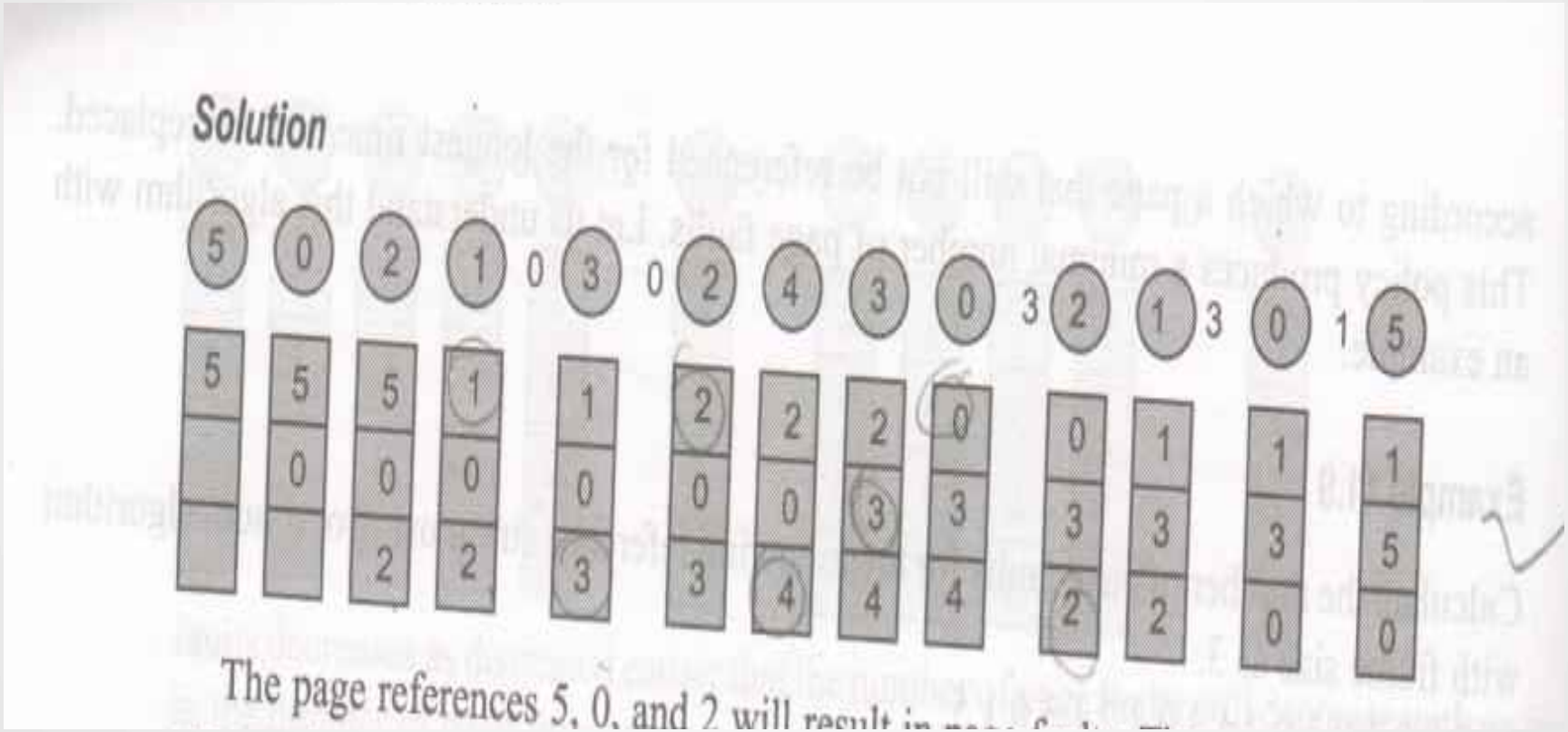


Initially, all the three frames are empty. Page number 5 is first referenced, and it is a page fault.

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

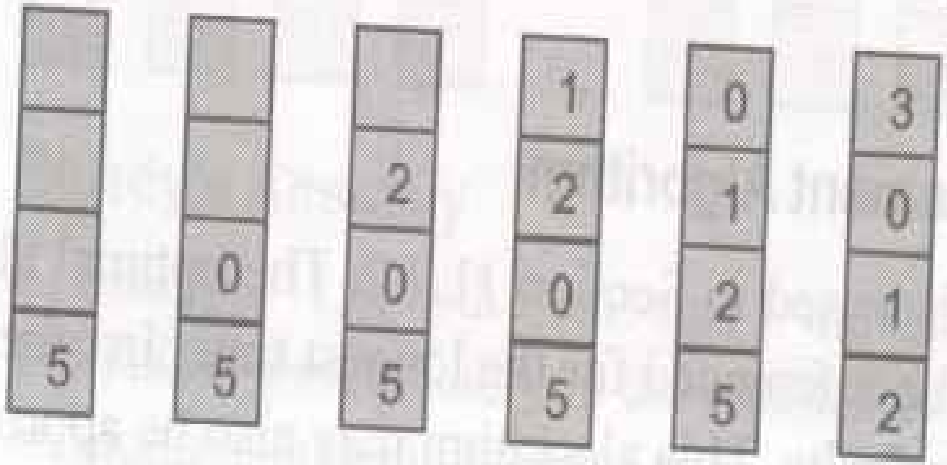
- **Stack Implementation**
  - To implement LRU, a linked list of all the pages in the memory can be maintained.
  - The list can be structured as a stack such that **whenever a page is referenced, it is placed at the top of the stack.**
  - This way, the **most recently used page will always be at the 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