Memory Management

Memory Management

- Background
- Swapping
- Contiguous Memory Allocation
- Paging
- Structure of the Page Table
- Segmentation

Objectives

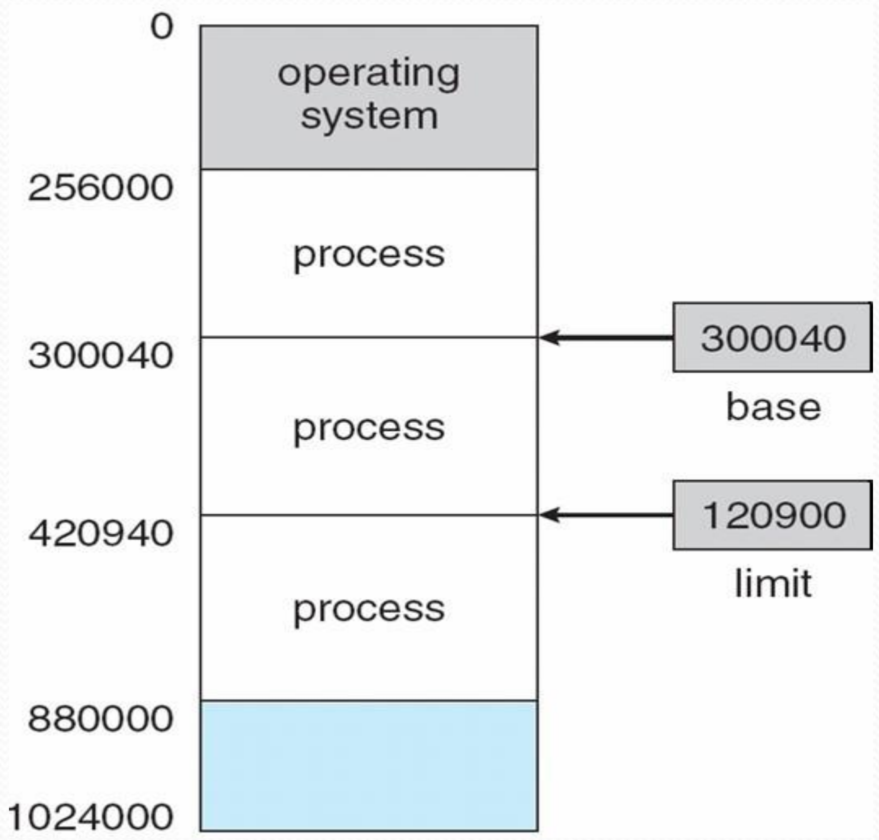
- To provide a detailed description of various ways of organizing memory hardware
- To discuss various memory-management techniques, including paging and segmentation

Background

- Program must be brought (from disk) into memory and placed within a process for it to be run
- Main memory and registers are only storage CPU can access directly
- Register access in one CPU clock (or less)
- Main memory can take many cycles
- **Cache** sits between main memory and CPU registers
- Protection of memory required to ensure correct operation

Base and Limit Registers

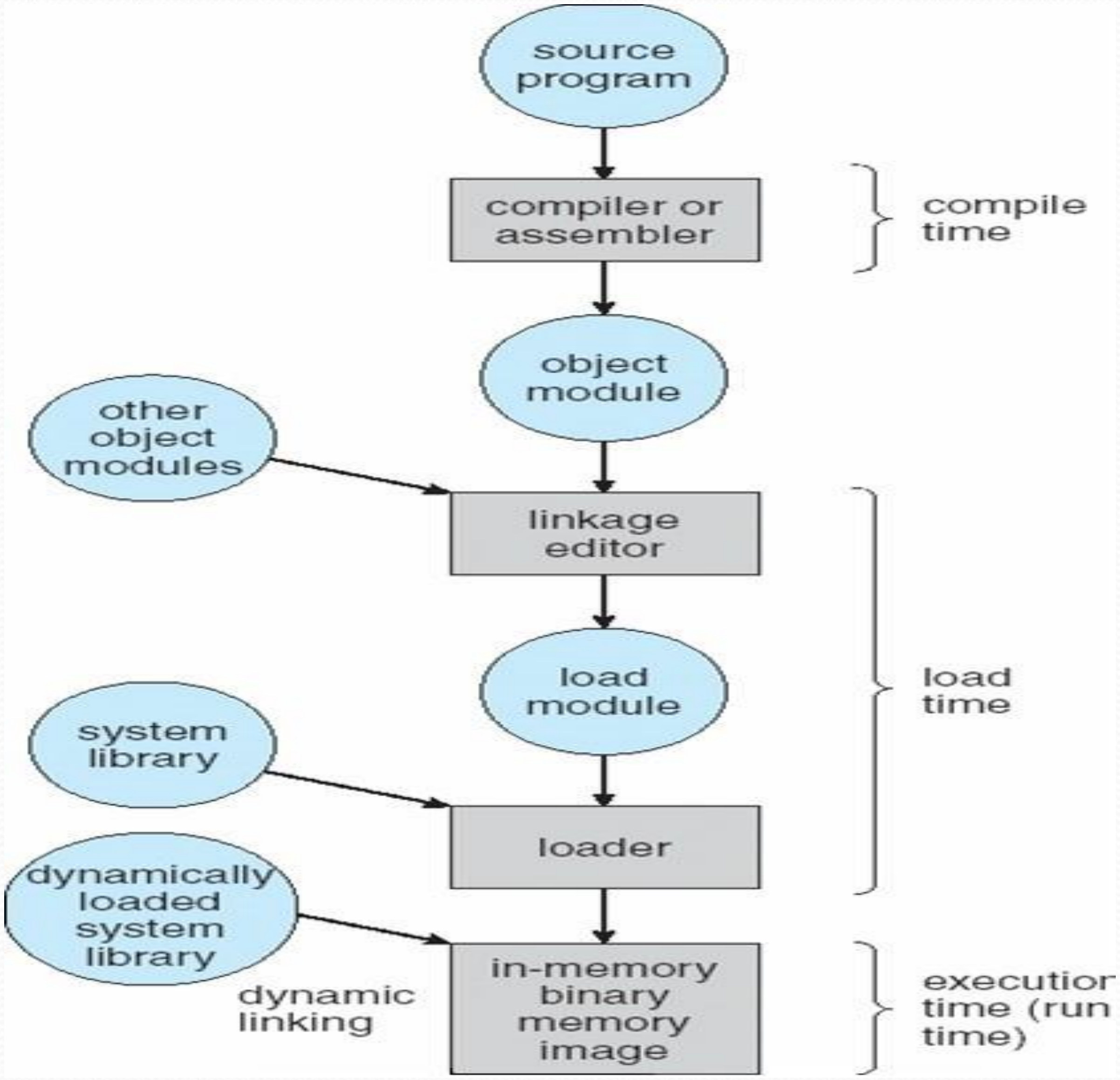
- A pair of **base** and **limit** registers define the logical address space



Binding of Instructions and Data to Memory

- Address binding of instructions and data to memory addresses can happen at three different stages
  - **Compile time**: If memory location known a priori, **absolute code** can be generated; must recompile code if starting location changes
  - **Load time**: Must generate **relocatable code** if memory location is not known at compile time
  - **Execution time**: Binding delayed until run time if the process can be moved during its execution from one memory segment to another. Need hardware support for address maps (e.g., base and limit registers)

Multistep Processing of a User Program



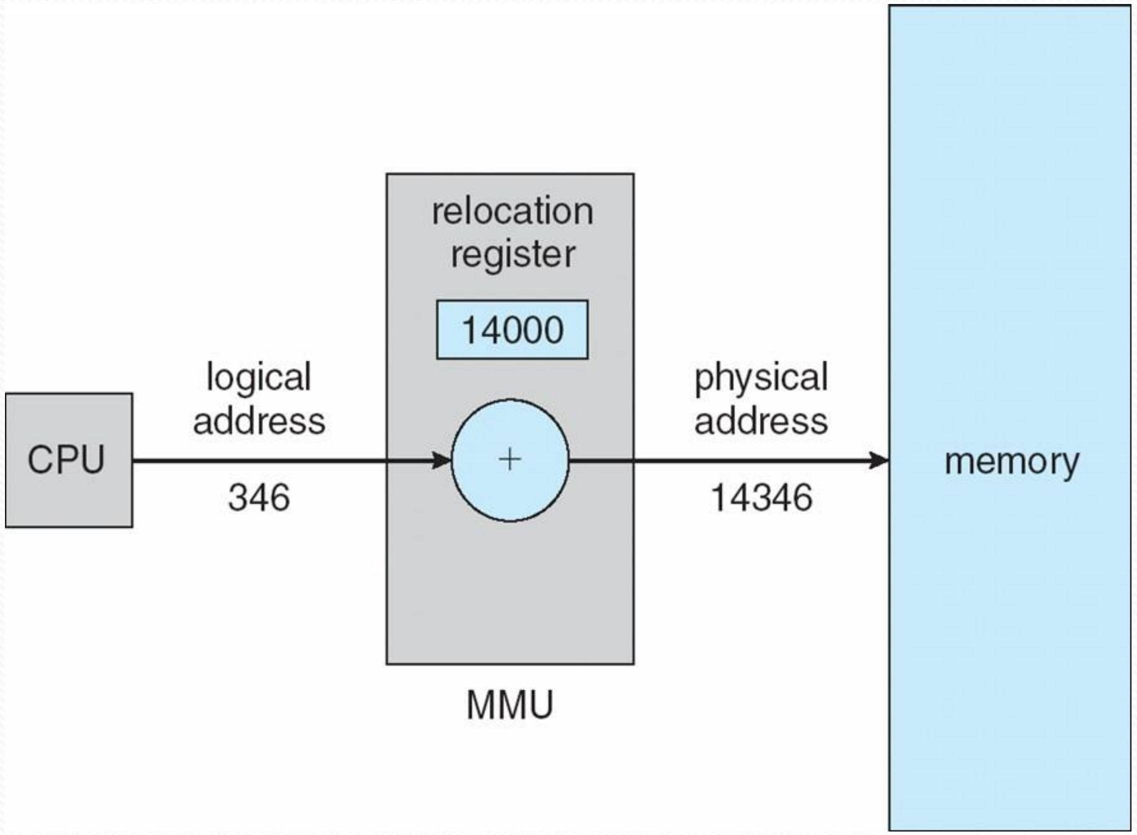
Logical vs. Physical Address Space

- The concept of a logical address space that is bound to a separate **physical address space** is central to proper memory management
  - **Logical address** – generated by the CPU; also referred to as **virtual address**
  - **Physical address** – address seen by the memory unit
- Logical and physical addresses are the same in compile- time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme

Memory-Management Unit (MMU)

- Hardware device that maps virtual to physical address
- In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory
- The user program deals with *logical* addresses; it never sees the *real* physical addresses

Dynamic relocation using a relocation register



Static vs Dynamic Loading

- At the time of loading, with **static loading**, the absolute program (and data) is loaded into memory in order for execution to start.

- If you are using **dynamic loading**, dynamic routines of the library are stored on a disk in relocatable form and are loaded into memory only when they are needed by the program.

Static vs Dynamic Linking

•when static linking is used, the linker combines all other modules needed by a program into a single executable program to avoid any runtime dependency.

•When dynamic linking is used, it is not required to link the actual module or library with the program, rather a reference to the dynamic module is provided at the time of compilation and linking.

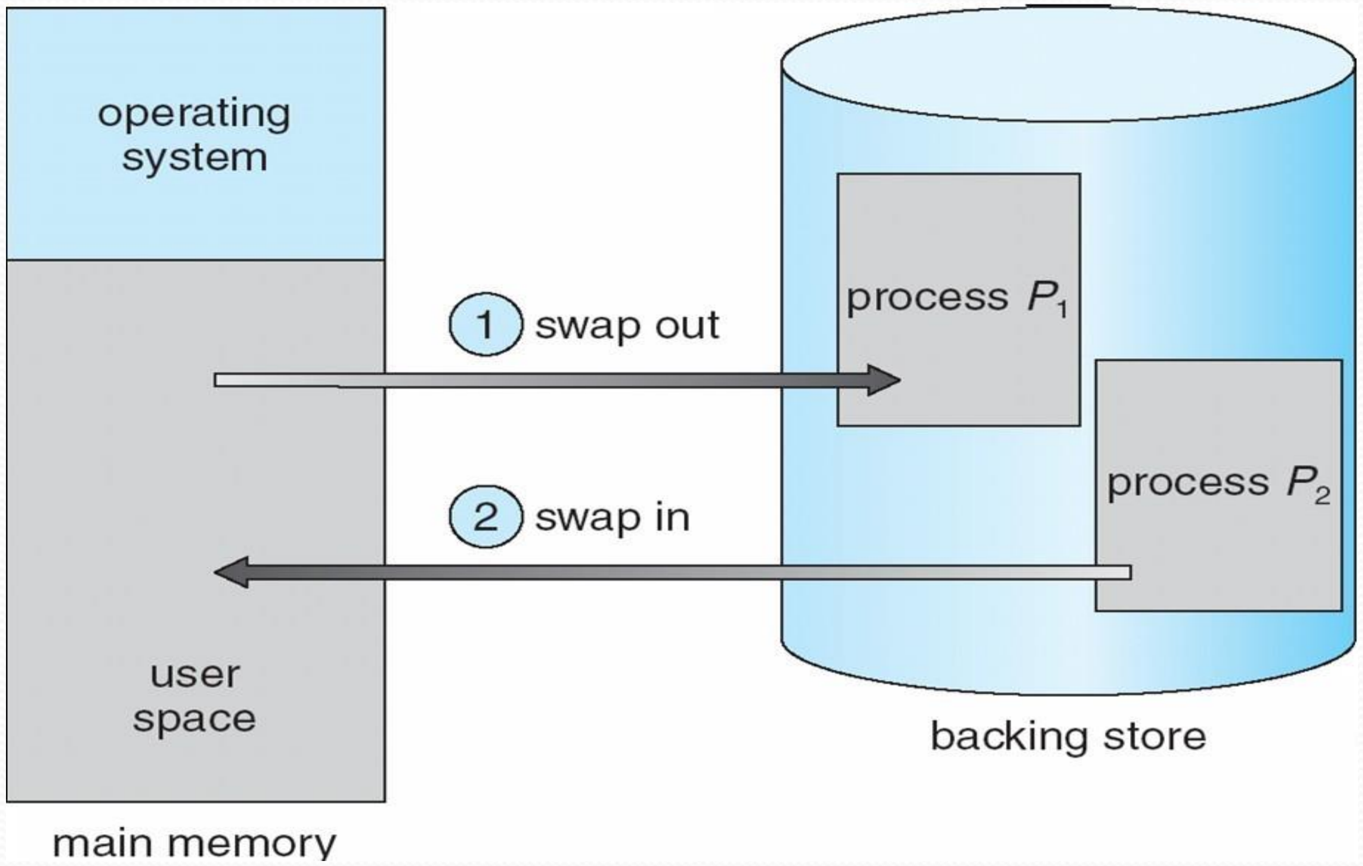
•Dynamic Link Libraries (DLL) in Windows and Shared Objects in Unix are good examples of dynamic libraries.

Swapping

- A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution

- **Backing store** – fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images

Schematic View of Swapping



Memory management Schemes

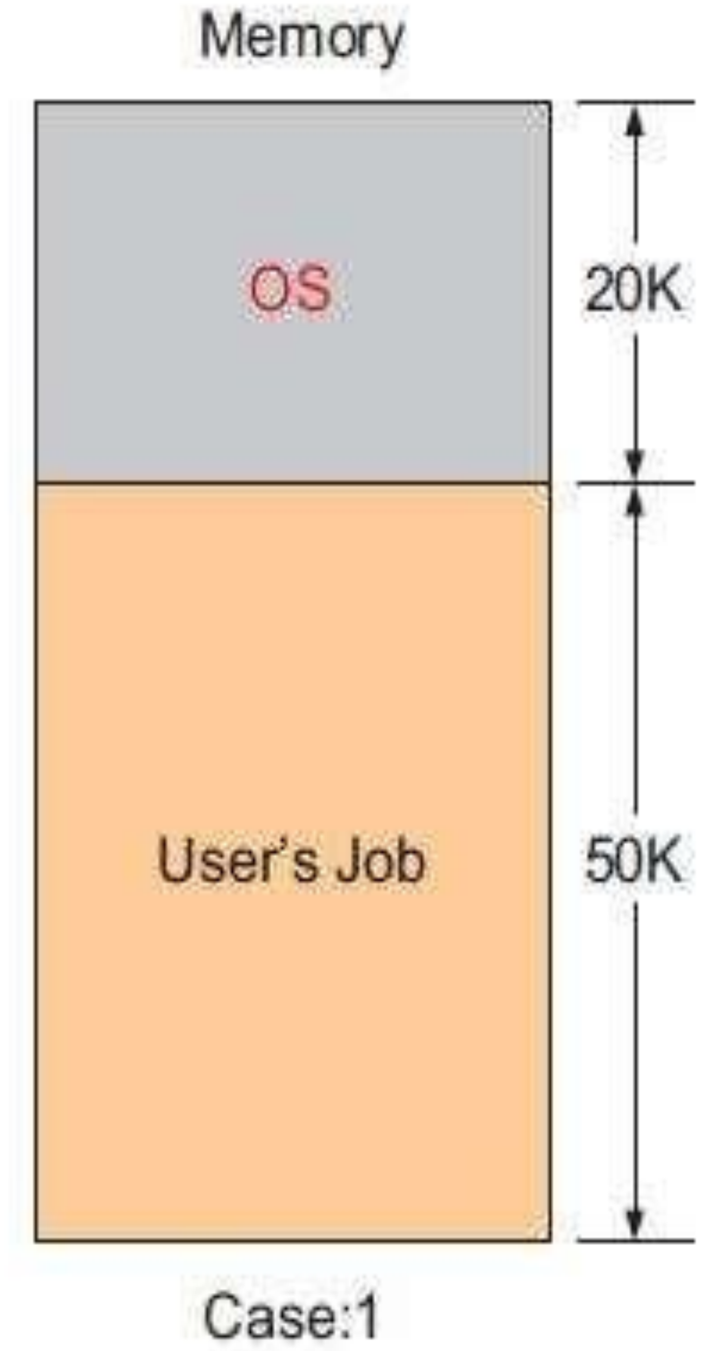
- Single contiguous memory allocation
- Fixed partition memory allocation
- Variable partition memory allocation

Single contiguous memory allocation

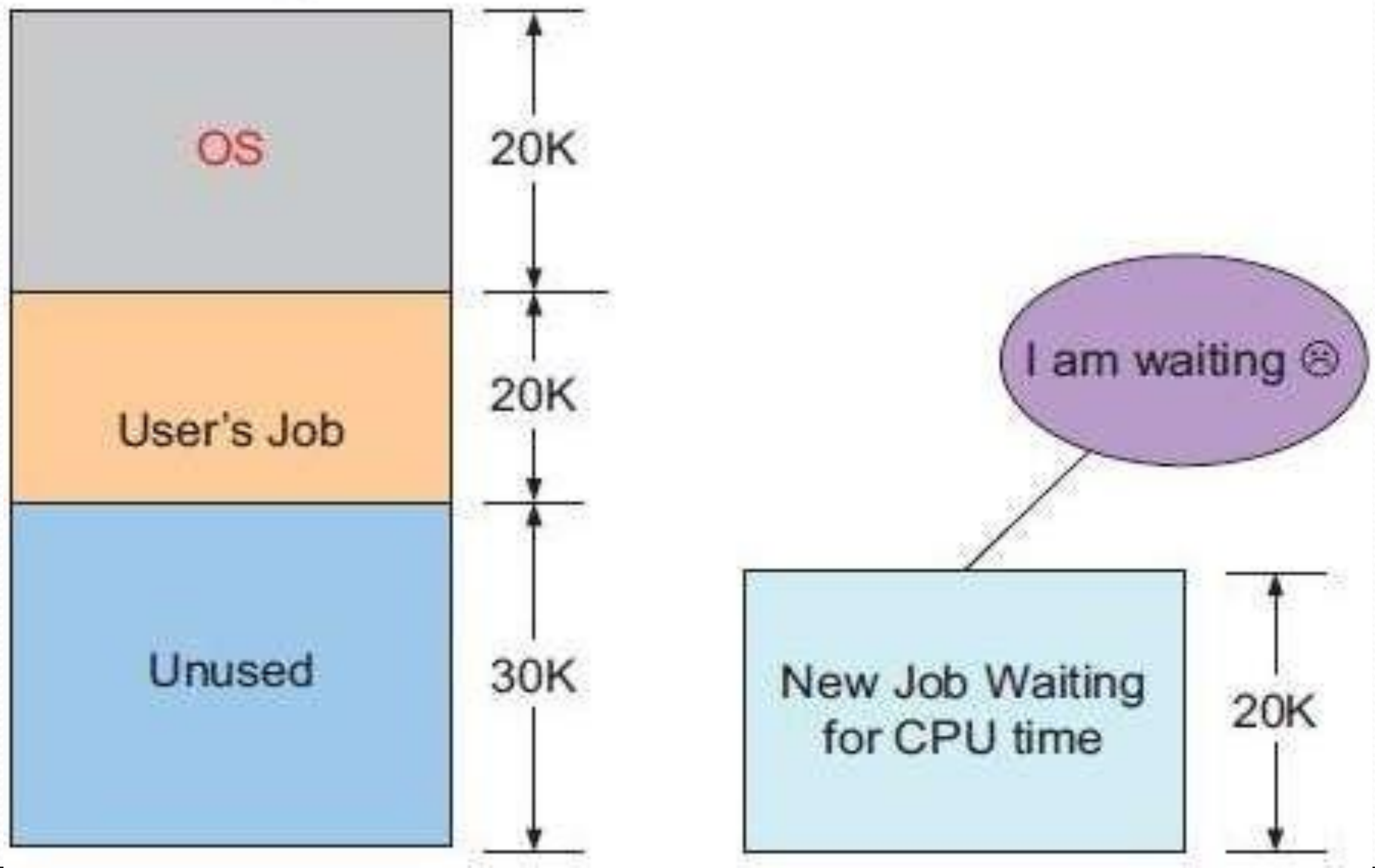
- The user’s job is assigned the complete control of the CPU until the job completes or an error occurs

- During this time, the user’s job is the only program which would reside in memory apart from the operating system

- Case1: Denotes a scenario where in the user's job occupies the complete available memory (the upper part of the memory is however, reserved for the OS program)



- Case2 : 30K of memory which is free but the new job cannot be put in memory because of the single contiguous allocation technique. From the above figure and the related discussion, the following advantages and disadvantages of single contiguous allocation technique can be seen.



Advantages:

- It is simple to implement

Disadvantages:

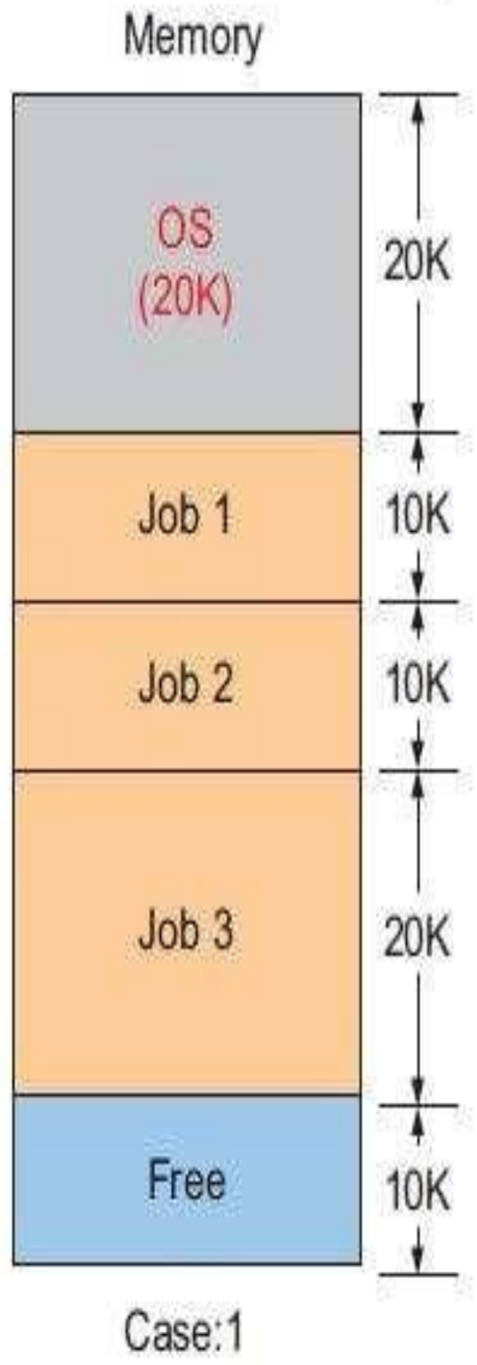
- It leads to wastage of memory which is called **fragmentation**

•This memory management technique would lead to *uniprogramming*. Hence it cannot be used for *multiprogramming*.

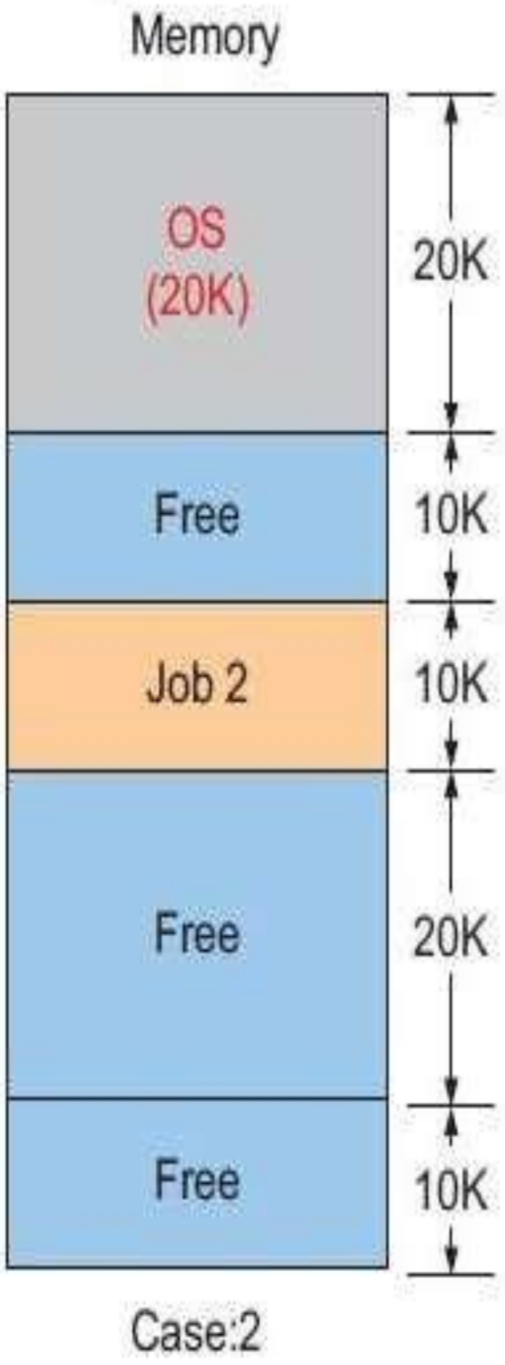
•It leads to wastage of CPU time (wastage of time). When the current job in memory is waiting for an input or output operation the CPU is left idle

Fixed partition memory allocation

- The memory is divided into various partitions each of fixed size.
- This would allow several user jobs to reside in the memory

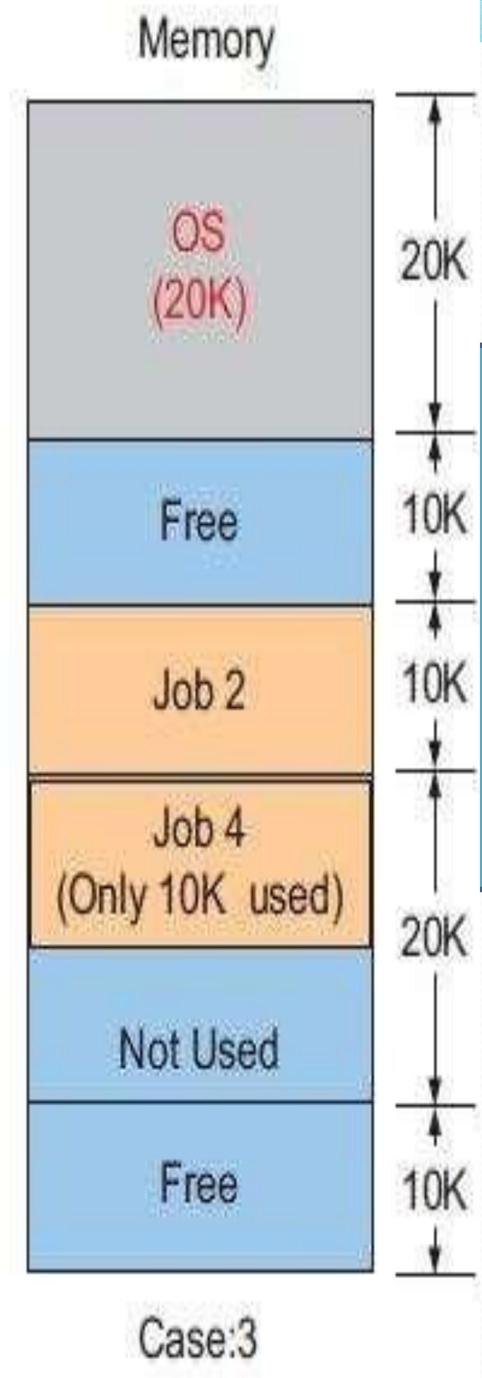


Case1: There are three jobs residing in memory each of which fits exactly into the respective partitions. One more partitioned memory is available for a user job. This type of fixed partition allocation supports multiprogramming.



Case2: Suppose that a new job of size 40K arrives for execution. It can be seen that the total amount of free memory is 40K but the new job cannot fit in to memory for execution because of lack of contiguous free space.

Case2 leads to **external fragmentation** wherein there is enough free memory for a new job but they are not contiguous



Case3: This depicts a scenario where job 4 is allocated a memory partition of 20K but it has occupied only 10K of this memory partition and the remaining 10K is unused.

Case3 leads to **internal fragmentation** wherein there is an unused part of memory internal to a memory partition.

Advantages:

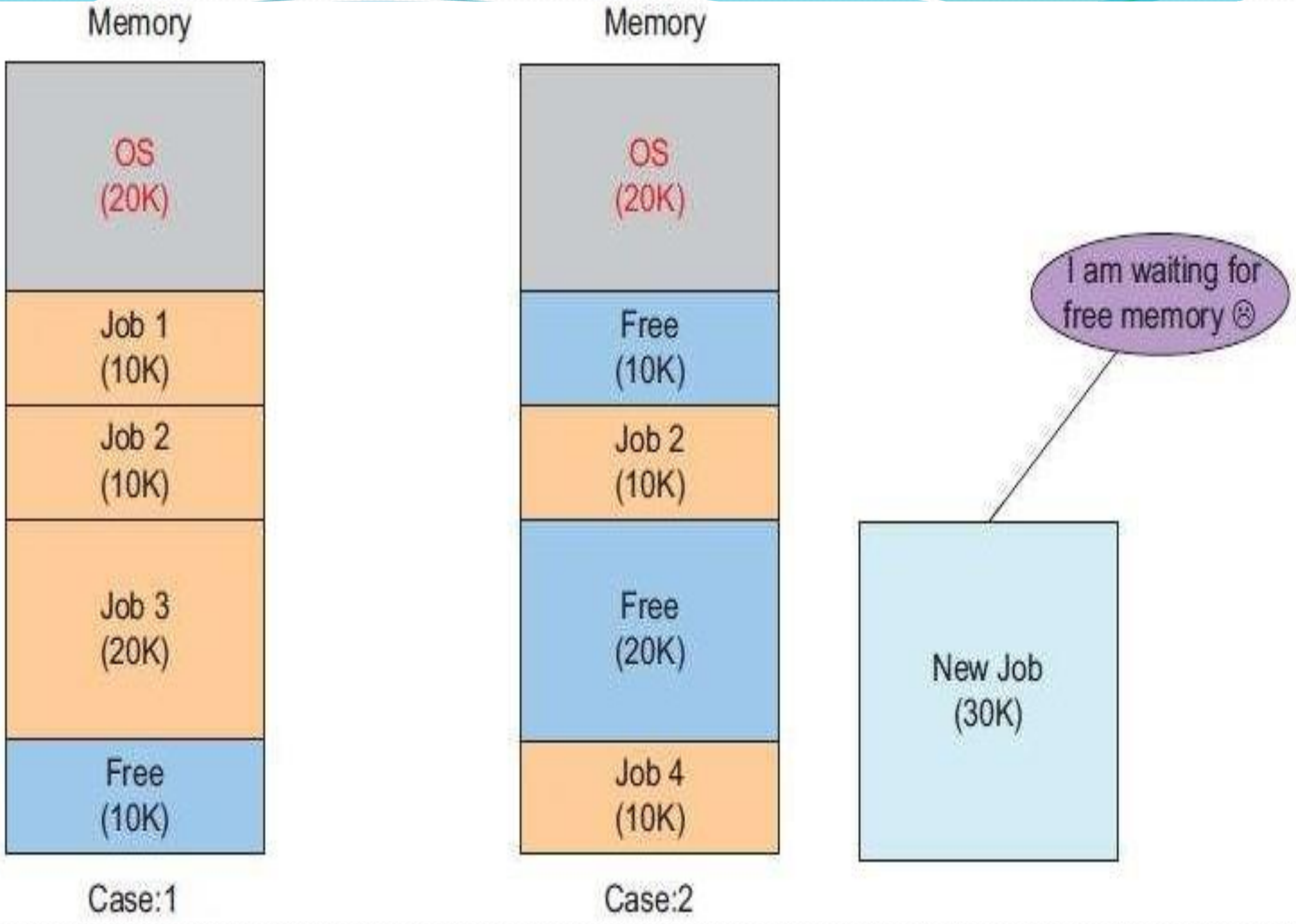
- provide multiprogramming

Disadvantages:

•internal and external fragmentation of memory

Variable partition memory allocation

- There is no pre-determined (fixed) partitioning of memory.
- This technique allocates the exact amount of memory required for a job


Memory
OS (20K)
Job 1 (10K)
Job 2 (10K)
Job 3 (20K)
Free (10K)
Case:1
Memory
OS (20K)
Free (10K)
Job 2 (10K)
Free (20K)
Job 4 (10K)
Case:2
I am waiting for free memory ☹
New Job (30K)

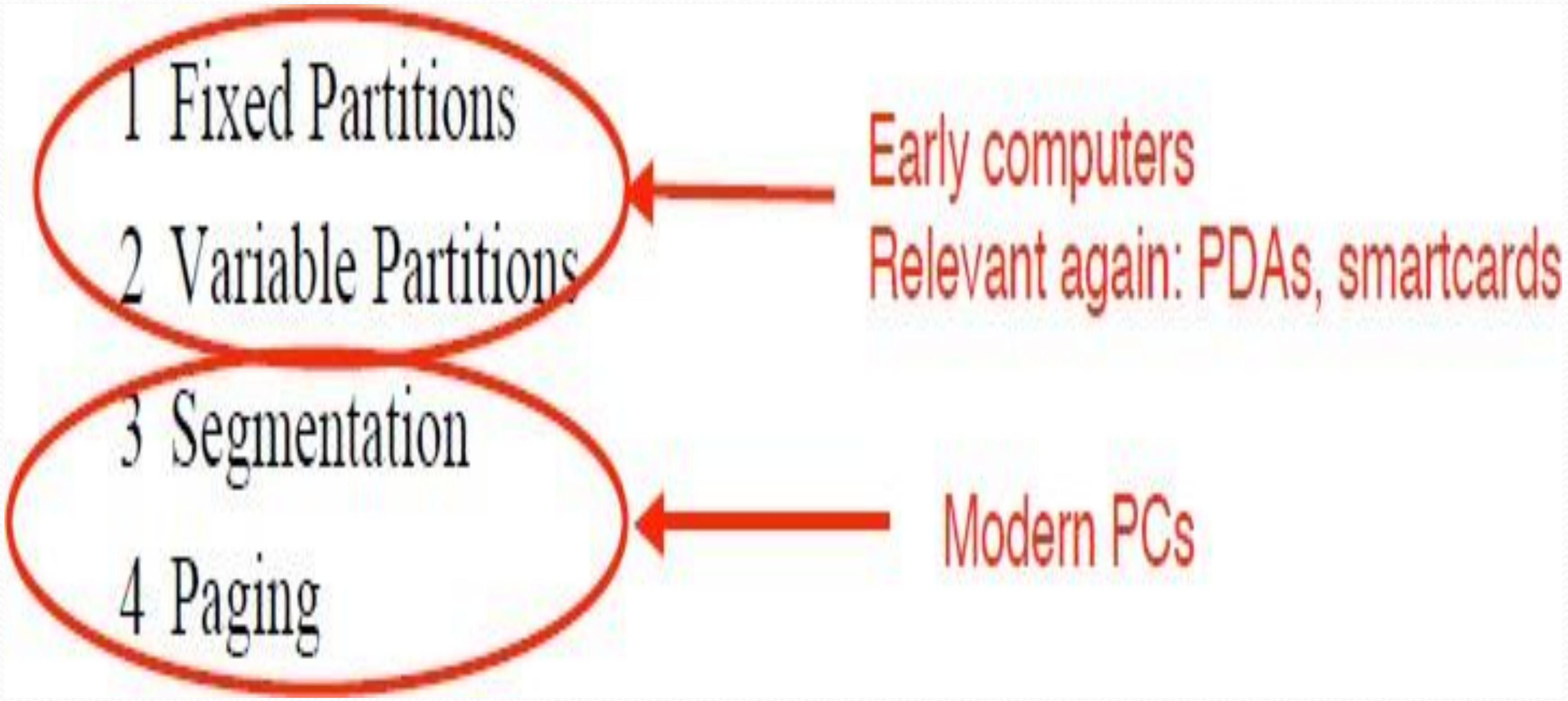
Advantages:

- prevents internal fragmentation

Disadvantages:

•cause external fragmentation of memory.

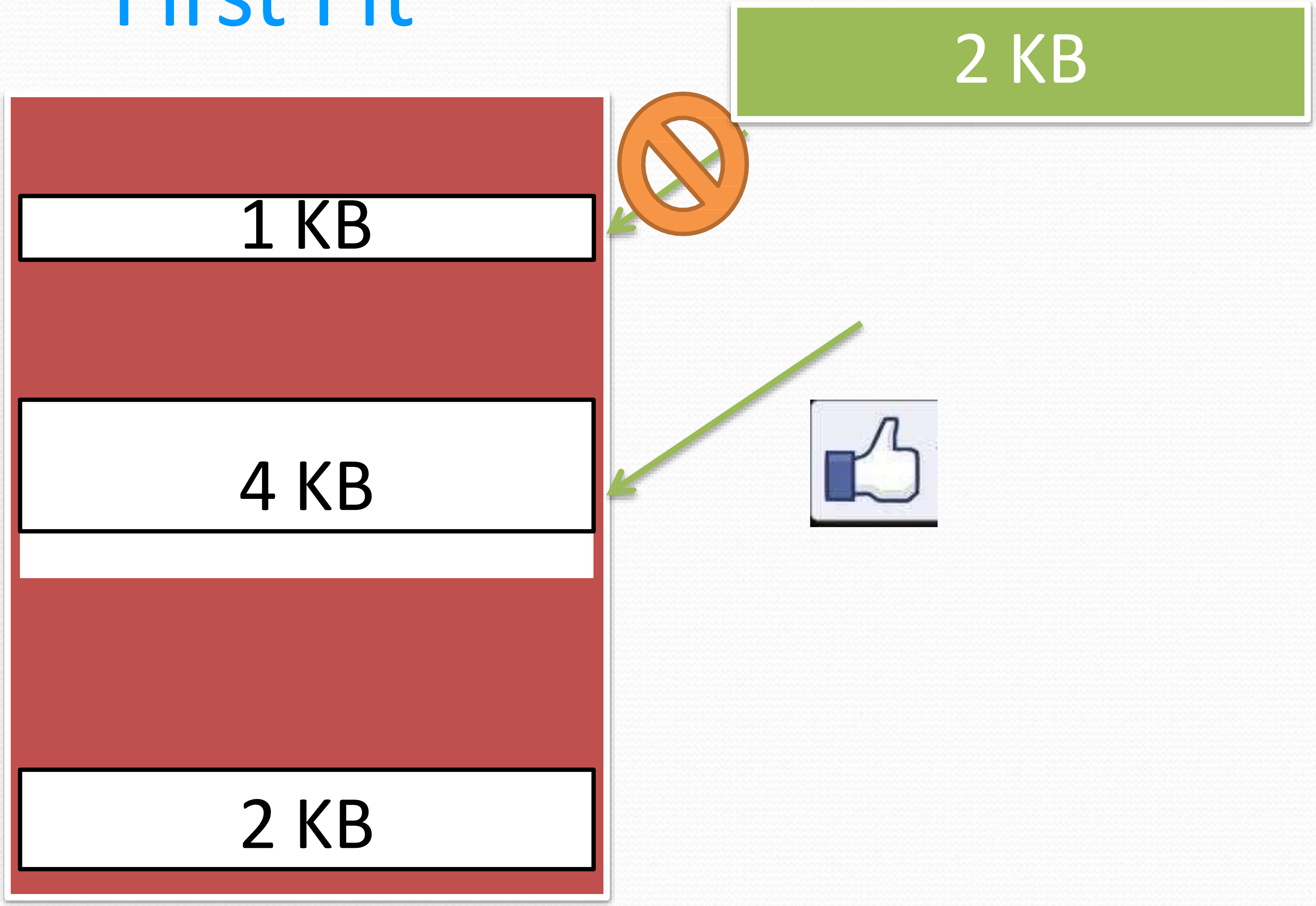
Remember!


1 Fixed Partitions
2 Variable Partitions
Early computers
Relevant again: PDAs, smartcards
3 Segmentation
4 Paging
Modern PCs

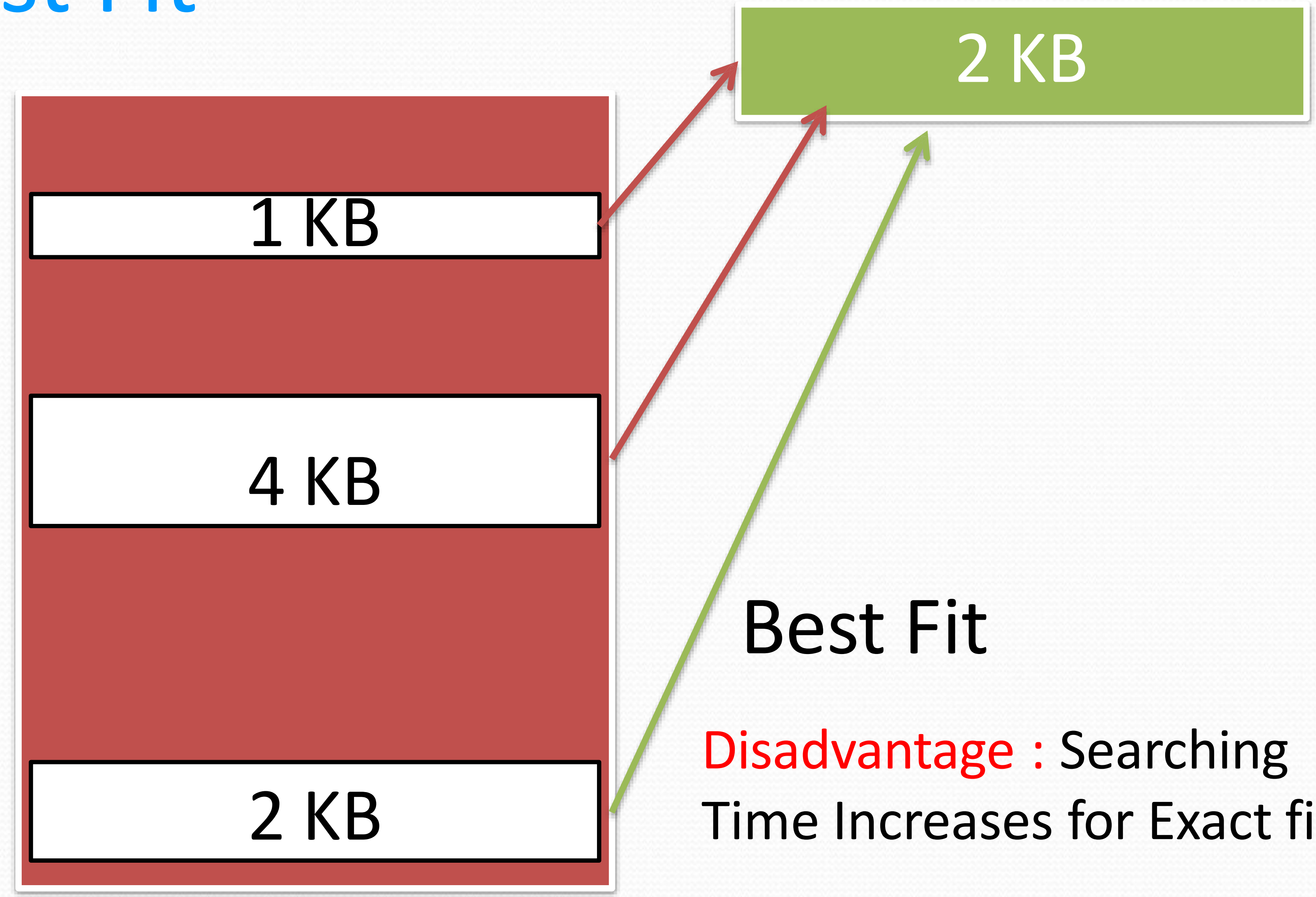
Memory Allocation Techniques

- First Fit
- Best Fit
- Worst Fit
- Next Fit
- Buddy’s System

First Fit



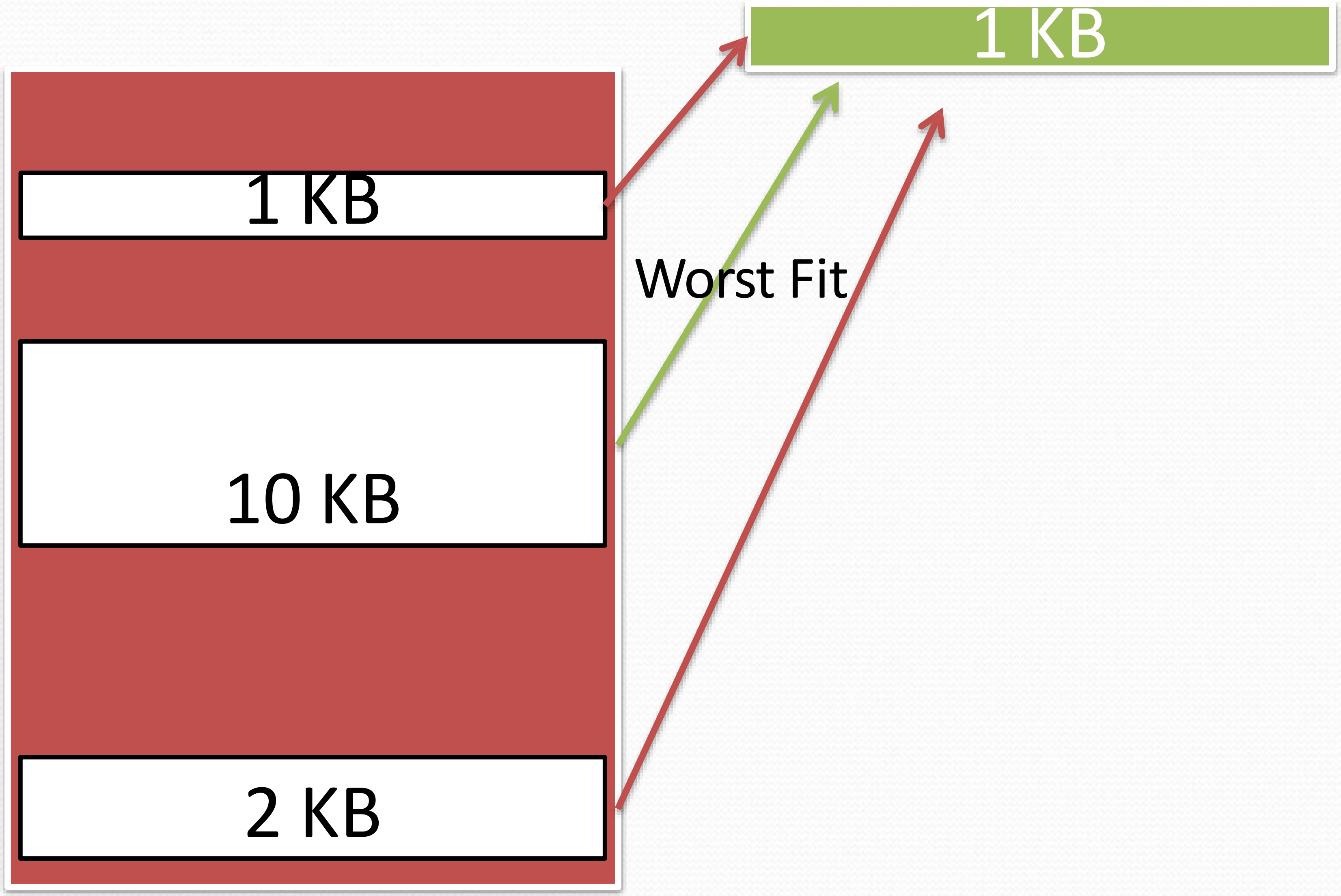
Best Fit



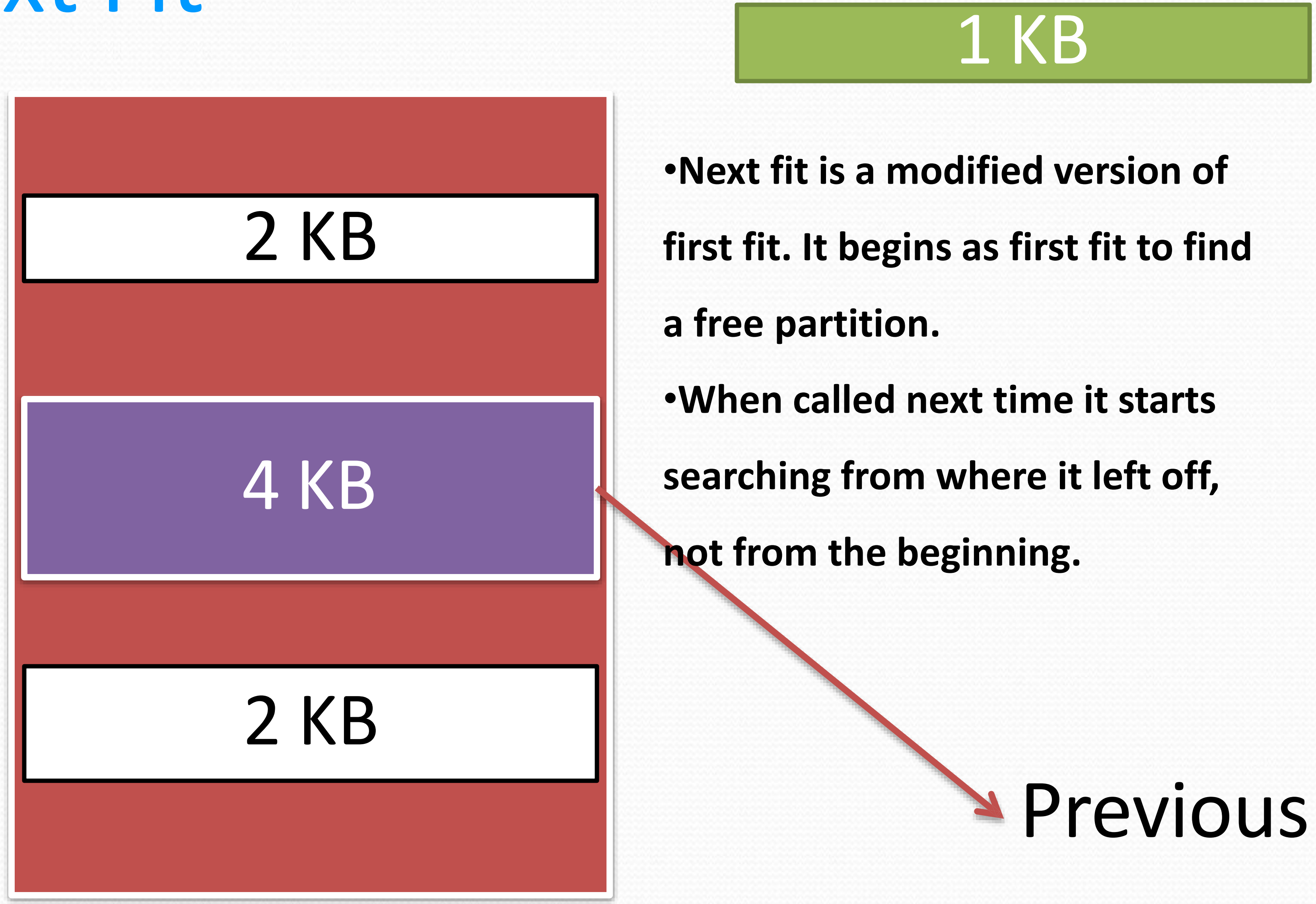
Best Fit

Disadvantage : Searching Time Increases for Exact fit

Worst Fit



Next Fit



•Next fit is a modified version of first fit. It begins as first fit to find a free partition.

•When called next time it starts searching from where it left off, not from the beginning.

Paging

•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 that's set up to emulate the computer's RAM.

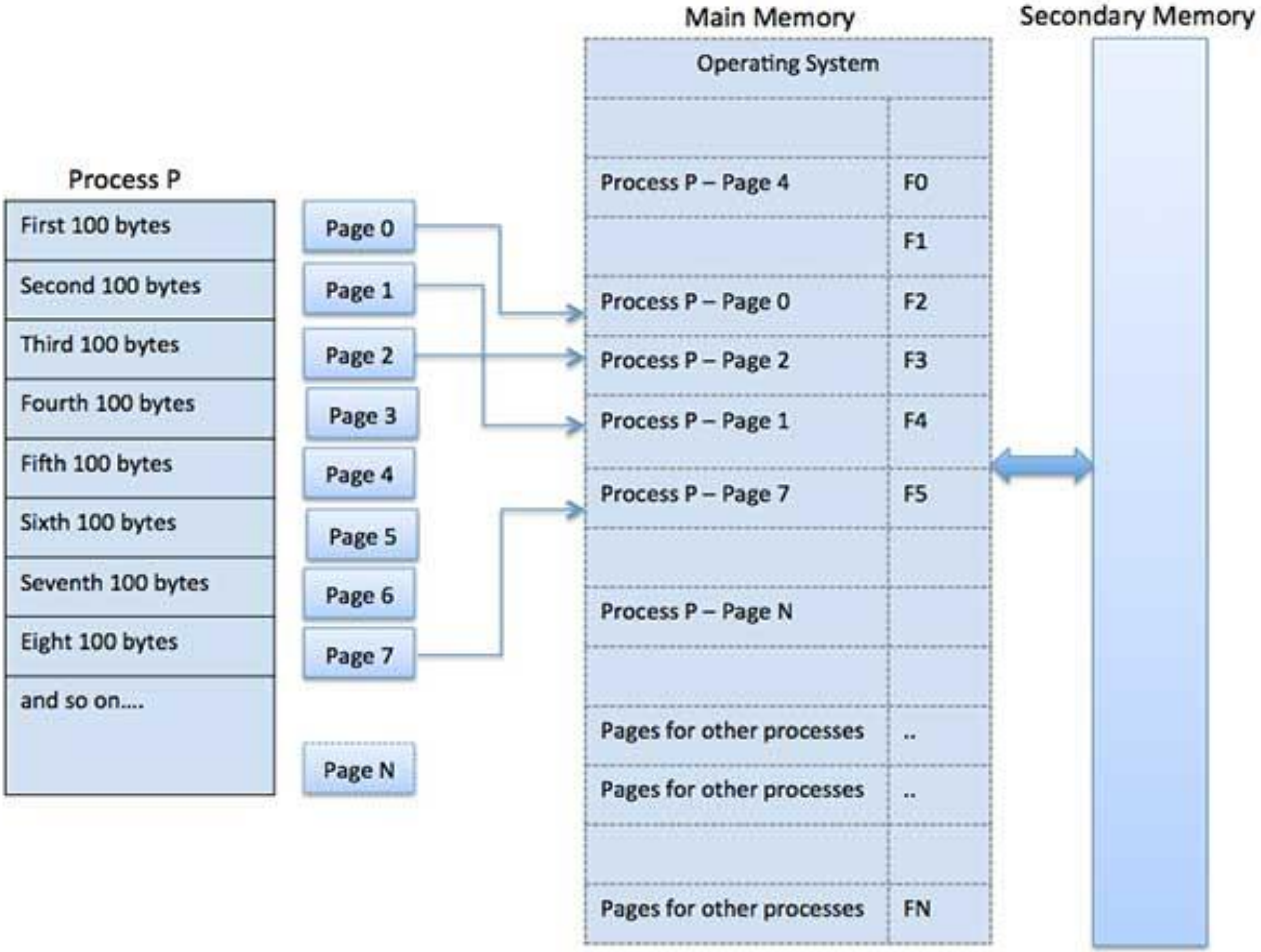
•Paging technique plays an important role in implementing virtual memory.

•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).

Paging

The size of the process is measured in the number of pages.

Similarly, 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.


Main Memory
Secondary Memory
Process P
First 100 bytes
Second 100 bytes
Third 100 bytes
Fourth 100 bytes
Fifth 100 bytes
Sixth 100 bytes
Seventh 100 bytes
Eight 100 bytes
and so on....
Page 0
Page 1
Page 2
Page 3
Page 4
Page 5
Page 6
Page 7
Page N
Operating System
Process P – Page 4
F0
F1
Process P – Page 0
F2
Process P – Page 2
F3
Process P – Page 1
F4
Process P – Page 7
F5
Process P – Page N
Pages for other processes
..
Pages for other processes
..
Pages for other processes
FN

Address Translation

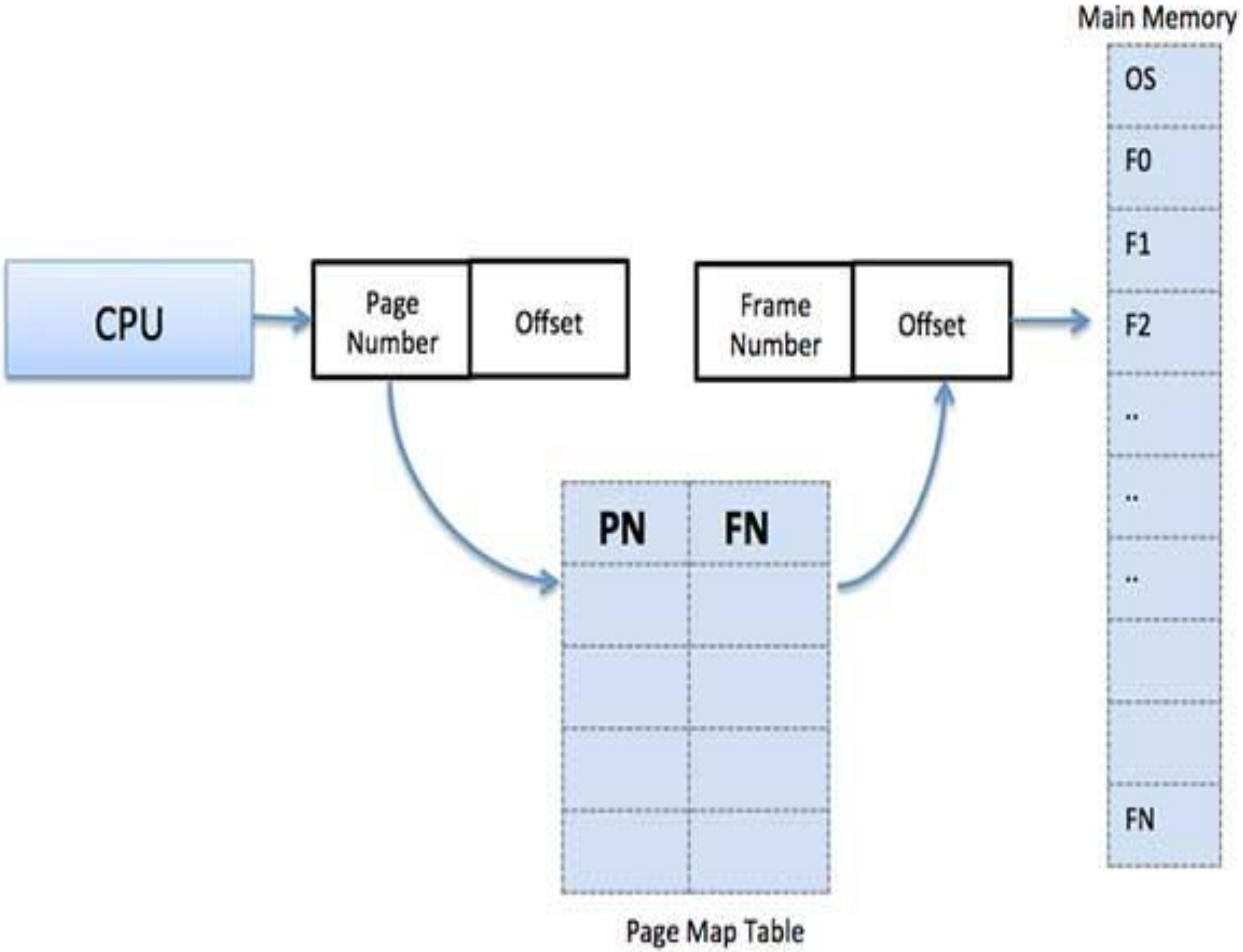
Page address is called **logical address** and represented by **page number** and the **offset**.

Logical Address = Page number + page offset

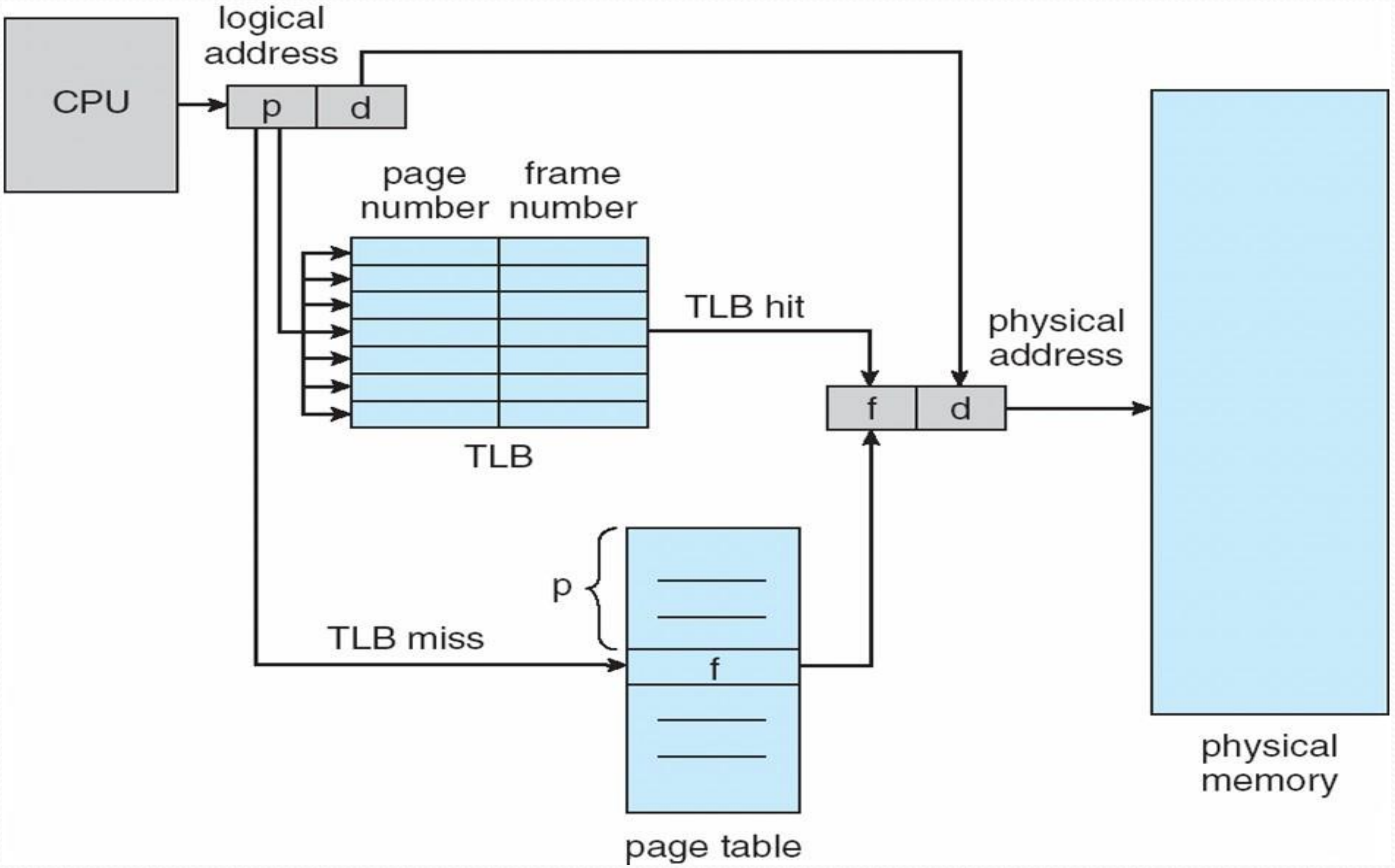
Frame address is called **physical address** and represented by a **frame number** and the **offset**.

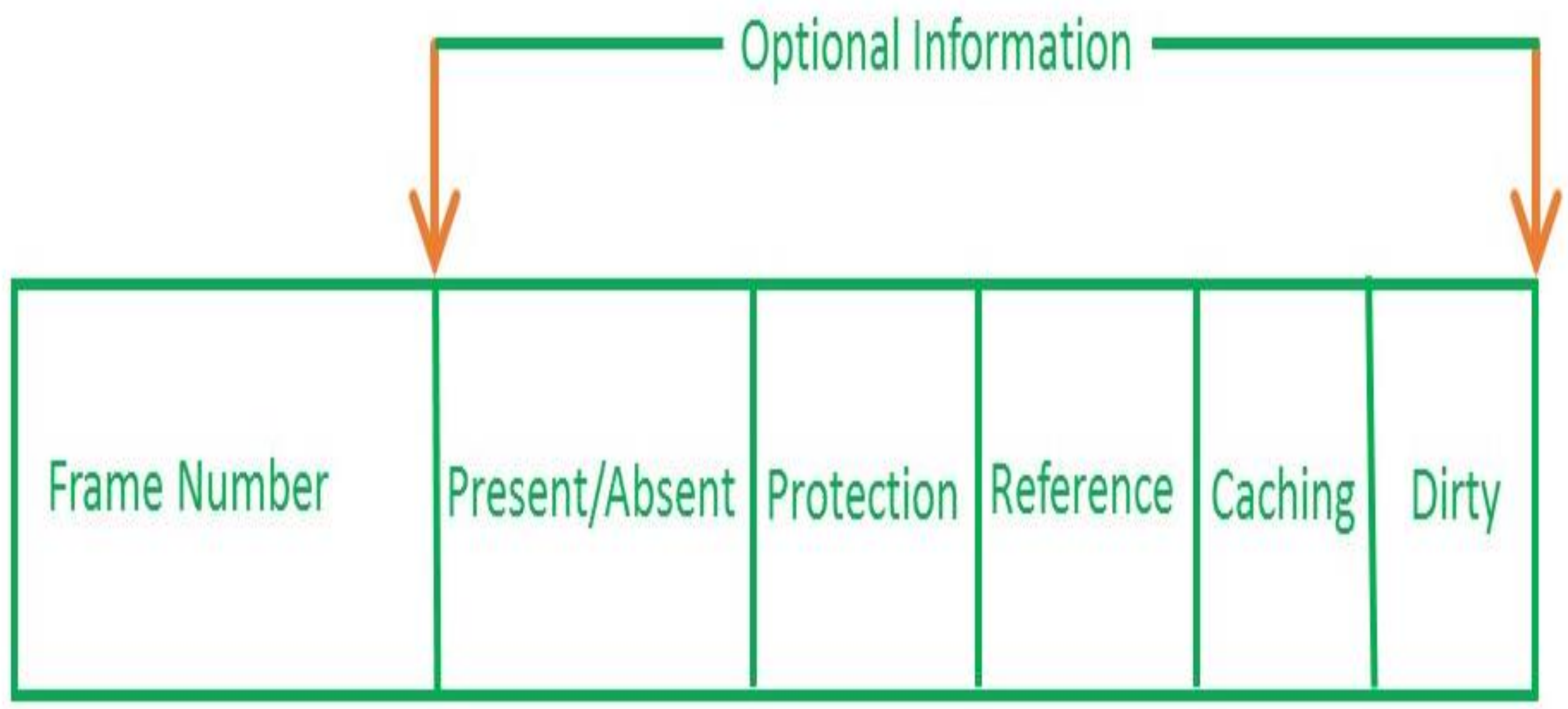
Physical Address = Frame number + page offset

A data structure called **page map table** is used to keep track of the relation between a page of a process to a frame in physical memory.


Main Memory
OS
F0
F1
F2
..
..
..
FN
CPU
Page Number
Offset
Frame Number
Offset
PN
FN
Page Map Table

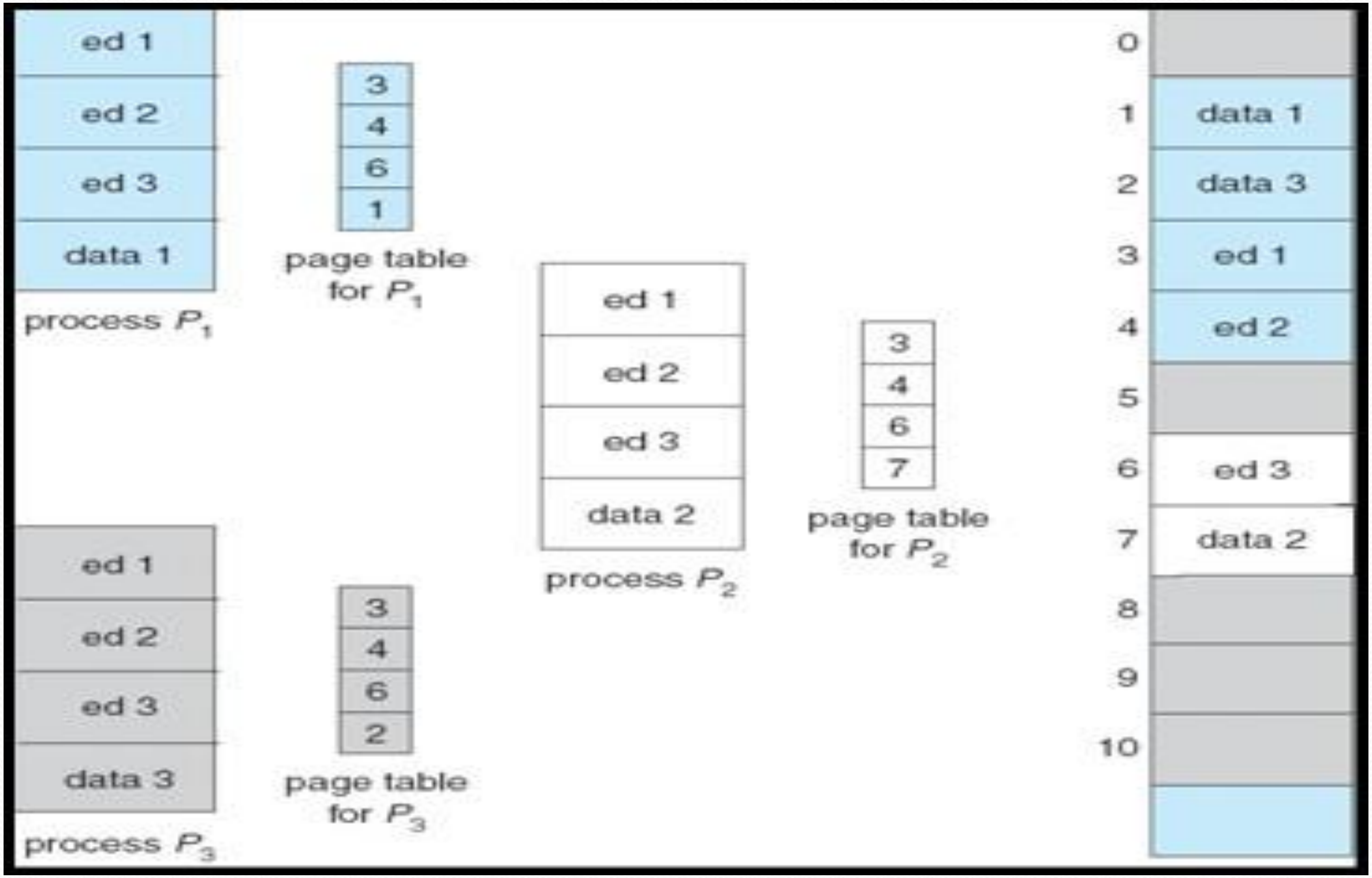
Paging Hardware With TLB





PAGE TABLE ENTRY

Shared Pages



- Reentrant code is non-self modifying code; it never changes during execution.
- Thus, two or more processes can execute the same code at the same time.
- Each process has its own copy of registers and data storage to hold the data for the process’s execution.
- The data for different processes will be different.

Throttling

- A technique used to more efficiently handle memory processing.
- During low-memory conditions, a system will slow down the processing of I/O memory requests, typically processing one sequence at a time in the order the request was received.
- I/O throttling slows down a system but typically will prevent the system from crashing.

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.
- When a process is to be executed, its corresponding segmentation are loaded into non-contiguous memory though every segment is loaded into a contiguous block of available memory.
- Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.

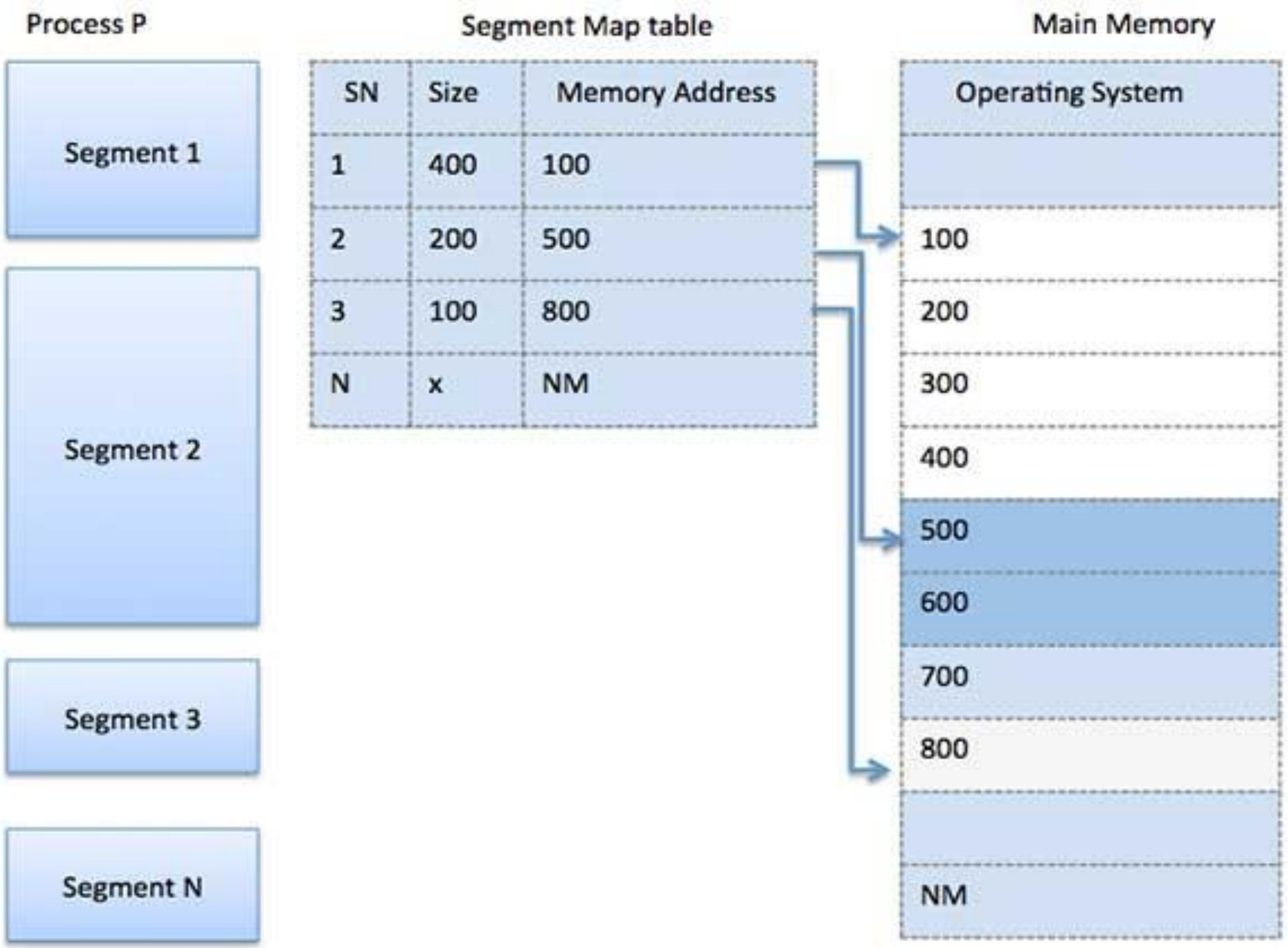
Segmentation

•A program segment contains the program's main function, utility functions, data structures, and so on.

•The operating system maintains a **segment map table** for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory.

•For each segment, the table stores the starting address of the segment and the length of the segment.

•A reference to a memory location includes a value that identifies a segment and an offset.



SN	Size	Memory Address
1	400	100
2	200	500
3	100	800
N	x	NM

Demand Paging

- According to the concept of Virtual Memory, in order to execute some process, only a part of the process needs to be present in the main memory which means that only a few pages will only be present in the main memory at any time.

- Therefore, to overcome this problem, there is a concept called Demand Paging is introduced.
- It suggests keeping all pages of the frames in the secondary memory until they are required.
- In other words, it says that do not load any page in the main memory until it is required.
- Whenever any page is referred for the first time in the main memory, then that page will be found in the secondary memory.

page fault

•When a page referenced by the CPU is not found in the main memory, it is called as a **page fault**.

•When a page fault occurs, the required page has to be fetched from the secondary memory into the main memory.

Page Replacement Algorithms

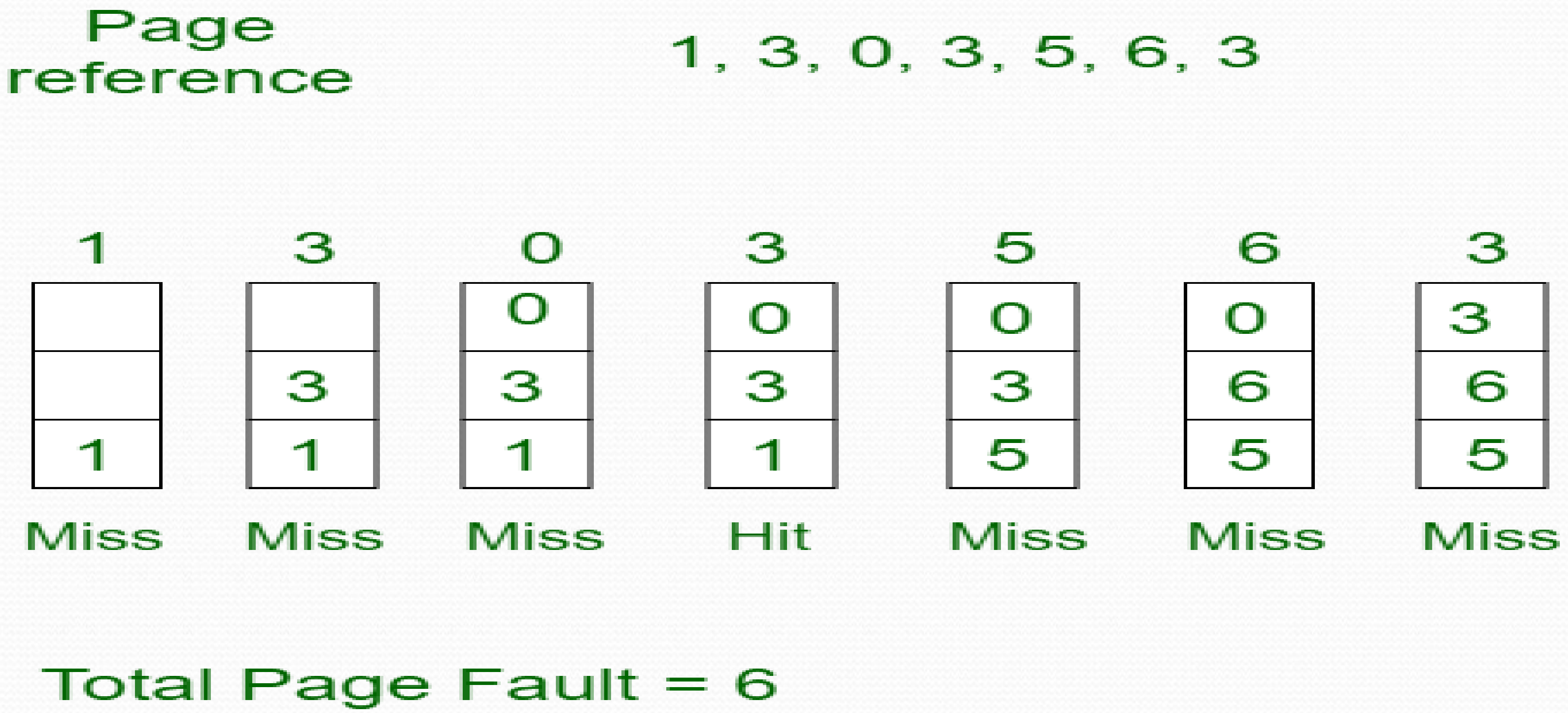
•**Optimal Page Replacement algorithm →** this algorithms replaces the page which will not be referred for so long in future. Although it can not be practically implementable but it can be used as a benchmark. Other algorithms are compared to this in terms of optimality.
•**Least recent used (LRU) page replacement algorithm →** this algorithm replaces the page which has not been referred for a long time. This algorithm is just opposite to the optimal page replacement algorithm. In this, we look at the past instead of staring at future.
•**FIFO →** in this algorithm, a queue is maintained. The page which is assigned the frame first will be replaced first. In other words, the page which resides at the rare end of the queue will be replaced on the every page fault.

Page Replacement Algorithms

- Optimal Page Replacement algorithm
- Least recent used (LRU) page replacement
- FIFO

First In First Out (FIFO)

This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

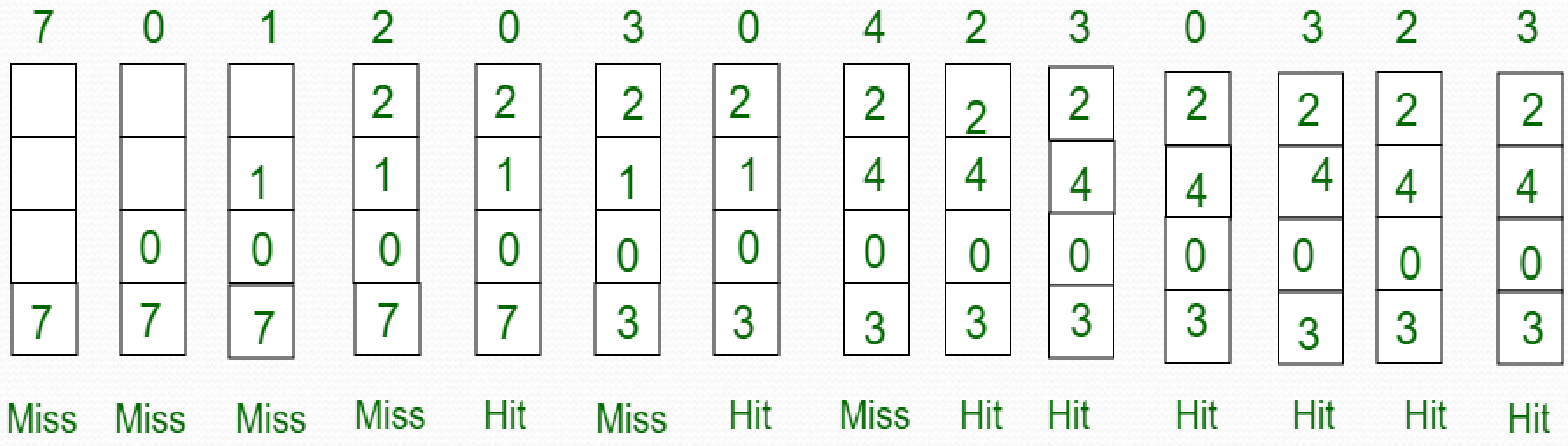


Optimal Page replacement

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

Page reference 7,0,1,2,0,3,0,4,2,3,0,3,2,3

No. of Page frame - 4



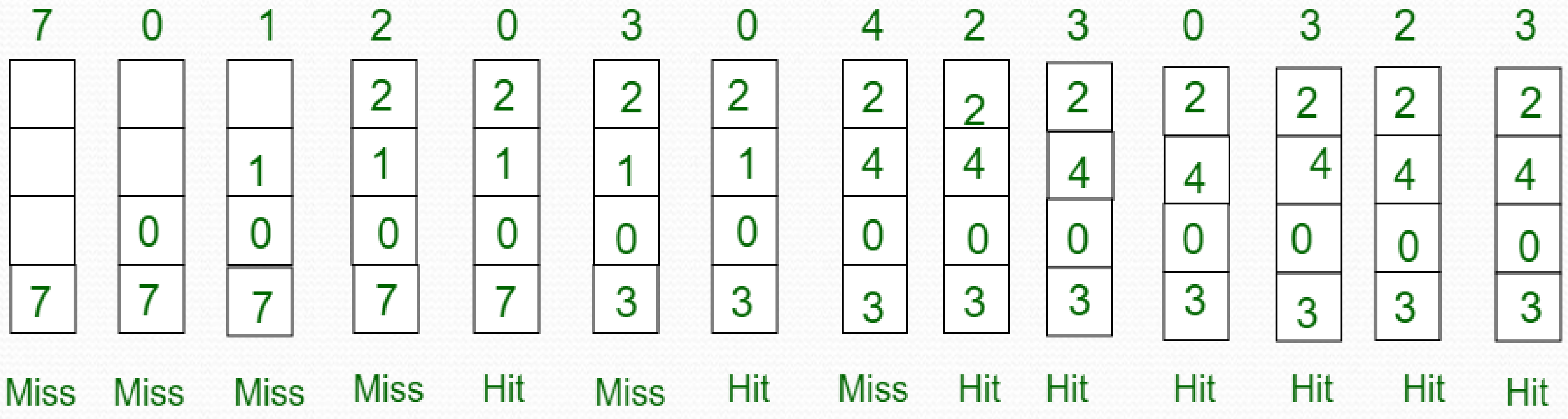
Total Page Fault = 6

Least Recently Used

In this algorithm page will be replaced which is least recently used.

Page reference 7,0,1,2,0,3,0,4,2,3,0,3,2,3

No. of Page frame - 4



7	0	1	2	0	3	0	4	2	3	0	3	2	3
			2	2	2	2	2	2	2	2	2	2	2
		1	1	1	1	1	4	4	4	4	4	4	4
	0	0	0	0	0	0	0	0	0	0	0	0	0
7	7	7	7	7	3	3	3	3	3	3	3	3	3
Miss	Miss	Miss	Miss	Hit	Miss	Hit	Miss	Hit	Hit	Hit	Hit	Hit	Hit

Total Page Fault = 6

Here LRU has same number of page fault as optimal but it may differ according to question.

Thank You