Unit 5: Memory Management

Marwadi University

Computer Engineering
Diploma

Unit 5
Memory Management
Operating System
09CE2405

Prof. Foram Chovatiya

Basic Memory Management

- Most computers have a memory hierarchy, with a small amount of very fast, expensive, volatile cache memory(temporary memory), volatile main memory (RAM) and non volatile disk storage.
- It is the job of operating system to coordinate how these memories are used.
- “The part of the operating system that manages memory hierarchy is called the **memory manager”.**

Memory Management

- Memory management is the functionality of an operating system which handles or manages primary memory and moves processes from one place to another and back again, between main memory and disk during execution.
- Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free.
- It checks how much memory is to be allocated to processes.
- It decides which process will get memory at what time.

Memory Management Terms

Term	Description
Frame	***Fixed***-length block of main memory.
Page	***Fixed***-length block of data in secondary memory (e.g. on disk).
Segment	***Variable-length*** block of data that resides in secondary memory.

Types of Address

- **Logical Address** is generated by CPU while a program is running.
- The logical address is virtual address as it does not exist physically therefore it is also known as Virtual Address.
- This address is used as a reference to access the physical memory location by CPU.

Types of Address

- **Physical Address** identifies a physical location of required data in a memory.
- The user never directly deals with the physical address but can access by its corresponding logical address.

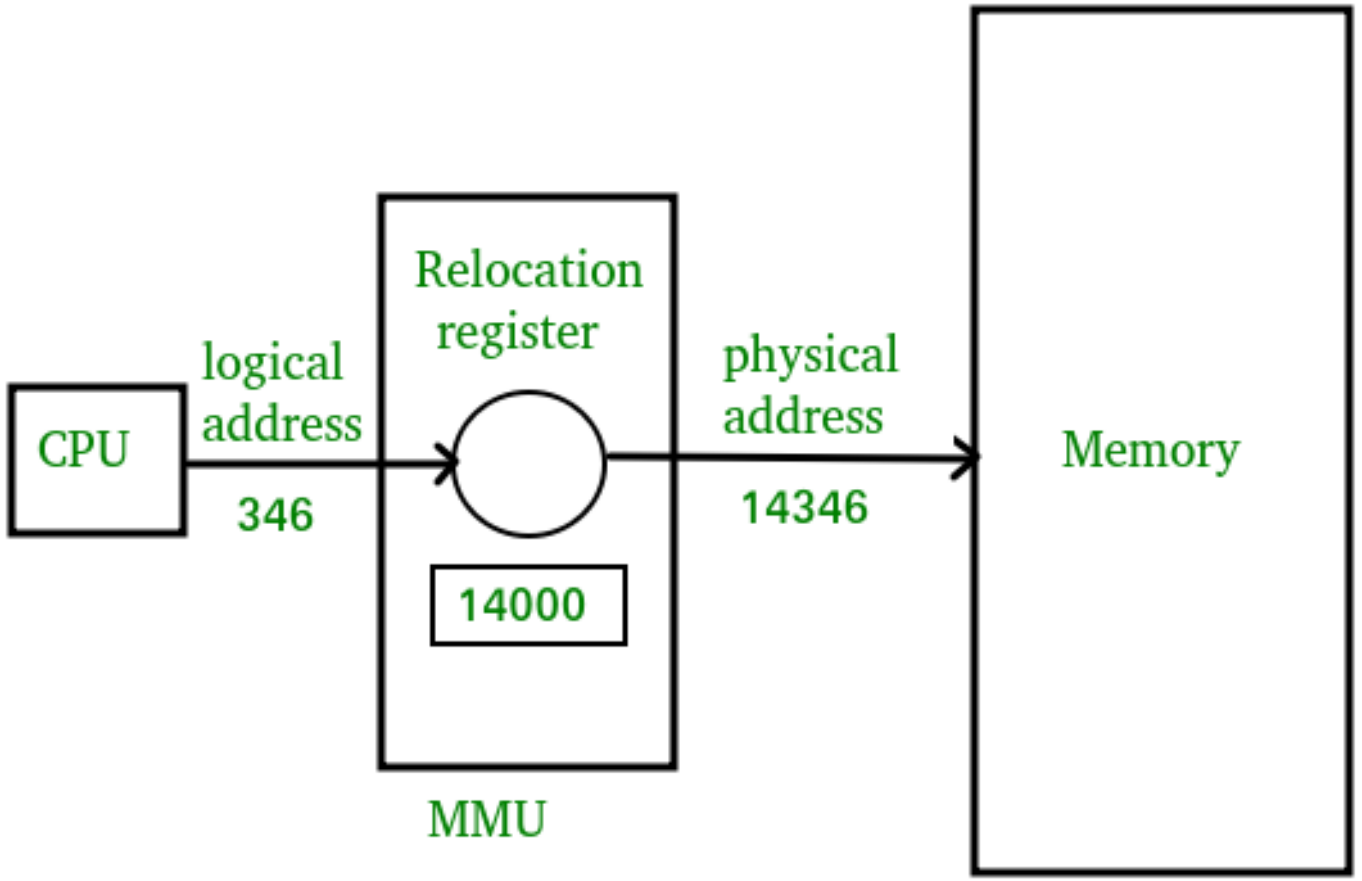
Process Address Space

- The process address space is the set of logical addresses that a process references in its code.
- Memory-Management Unit takes care of mapping the logical addresses to physical addresses at the time of memory allocation to the program.
- The set of all logical addresses generated by a program is referred to as a **logical address space.**
- The set of all physical addresses corresponding to these logical addresses is referred to as a **physical address space.**

Memory Management Unit

- The runtime mapping from logical (virtual) to physical address is done by the memory management unit (MMU) which is a hardware device.
- MMU uses following mechanism to convert logical (virtual) address to physical address.
- The value in the relocation register is added to every address generated by a user process, which is treated as offset at the time it is sent to memory.

Memory Management Unit



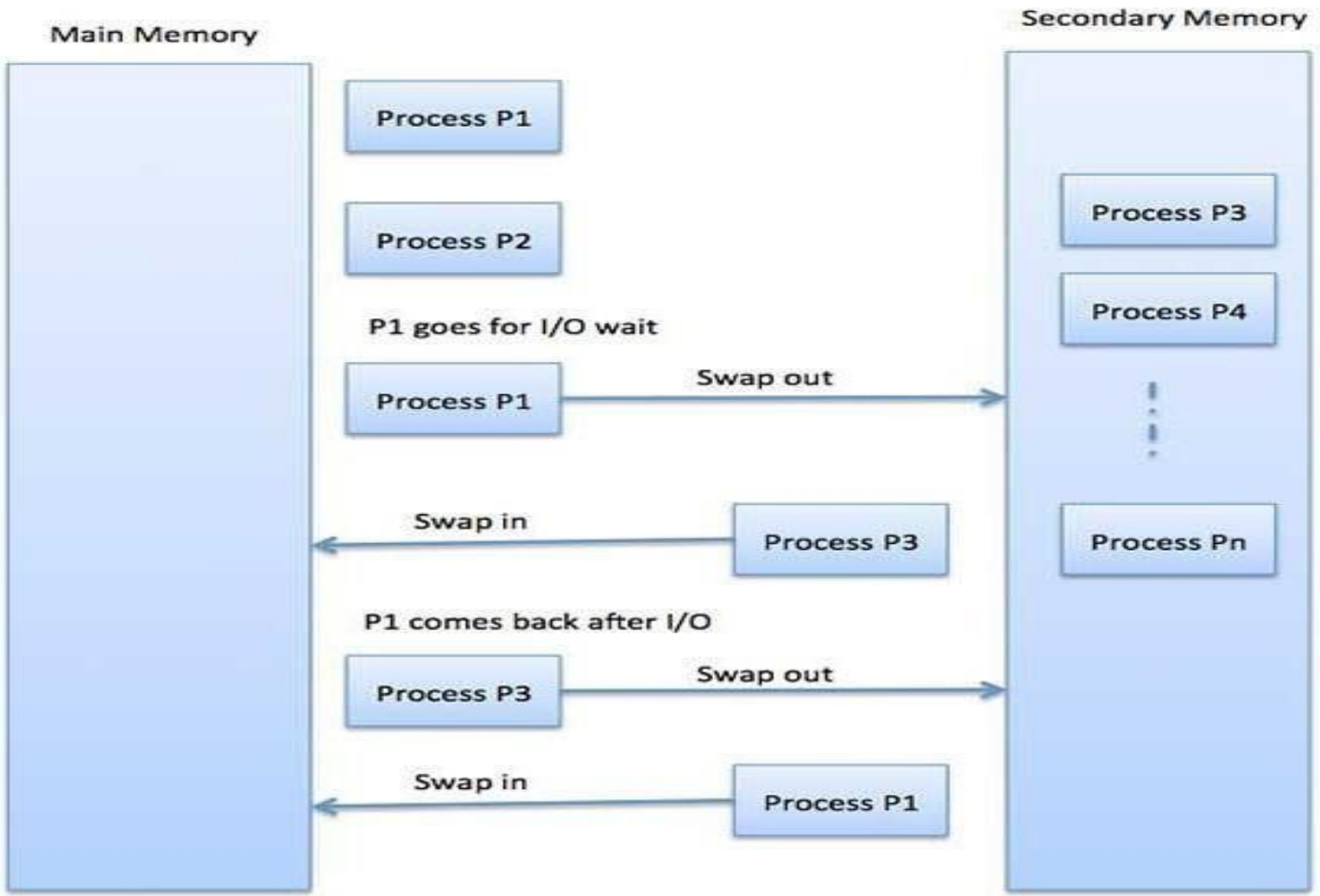
Memory Management Unit

- For example, if the base register value is 14000, then an attempt by the user to use address location 346 will be dynamically reallocated to location 14346.
- The user program deals with logical addresses; it never sees the real physical addresses.

Swapping

- Swapping is a mechanism in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes.
- At some later time, the system swaps back the process from the secondary storage to main memory.

Swapping


Main Memory
Secondary Memory
Process P1
Process P2
Process P3
Process P4
P1 goes for I/O wait
Process P1
Swap out
Swap in
Process P3
Process Pn
P1 comes back after I/O
Process P3
Swap out
Swap in
Process P1

Contiguous Memory Allocation

- Contiguous memory allocation is a classical memory allocation model that assigns a process consecutive(Continue) memory blocks (that is, memory blocks having consecutive addresses).
- Contiguous memory allocation is one of the oldest memory allocation schemes.
- When a process needs to execute, memory is requested by the process.

Contiguous Memory Allocation

- The size of the process is compared with the amount of contiguous main memory available to execute the process.
- If sufficient contiguous memory is found, the process is allocated memory to start its execution.
- Otherwise, it is added to a queue of waiting processes until sufficient free contiguous memory is available.

Memory Partition

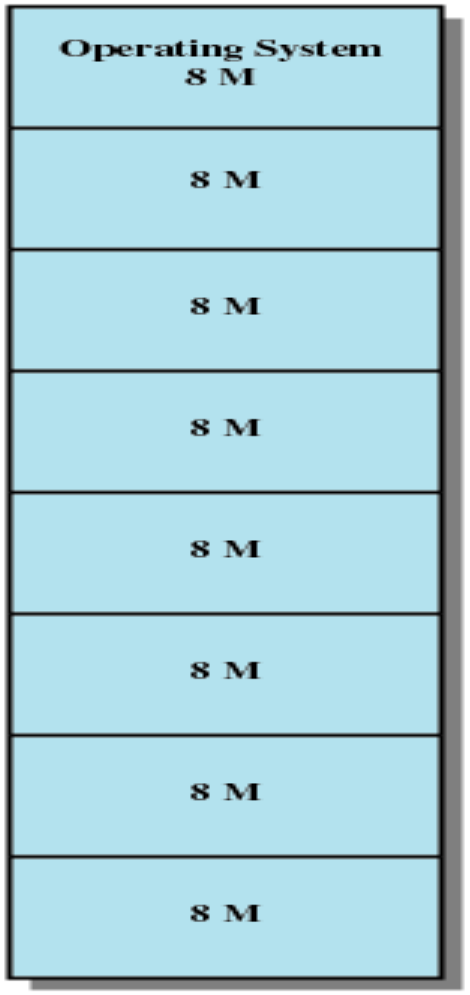
(1) Equal-size(Fix or Static) partitions:-

- Because all partitions are of equal size, it does not matter which partition is used

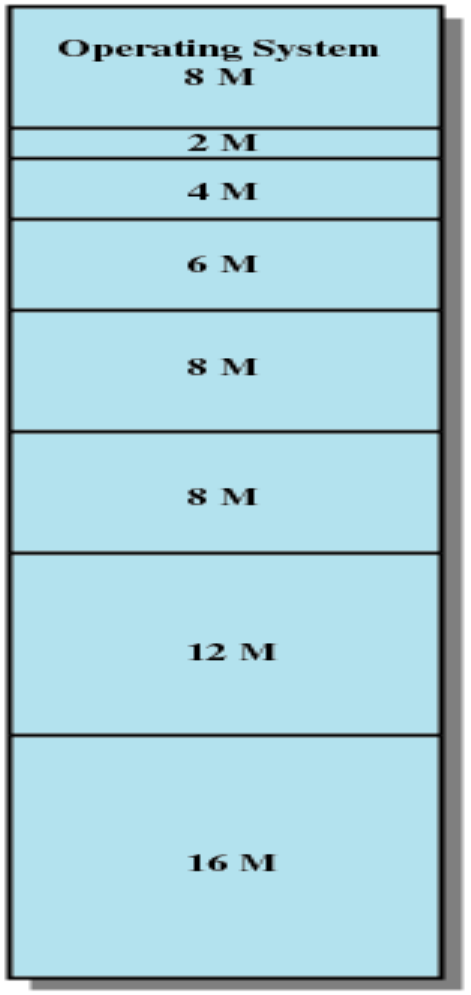
(2) Unequal-size(Variable or Dynamic) partitions:-

- Can assign each process to the smallest partition within which it will fit
- Queue for each partition
- Processes are assigned in such a way as to minimize wasted memory within a partition

Memory Partition



(a) Equal-size partitions

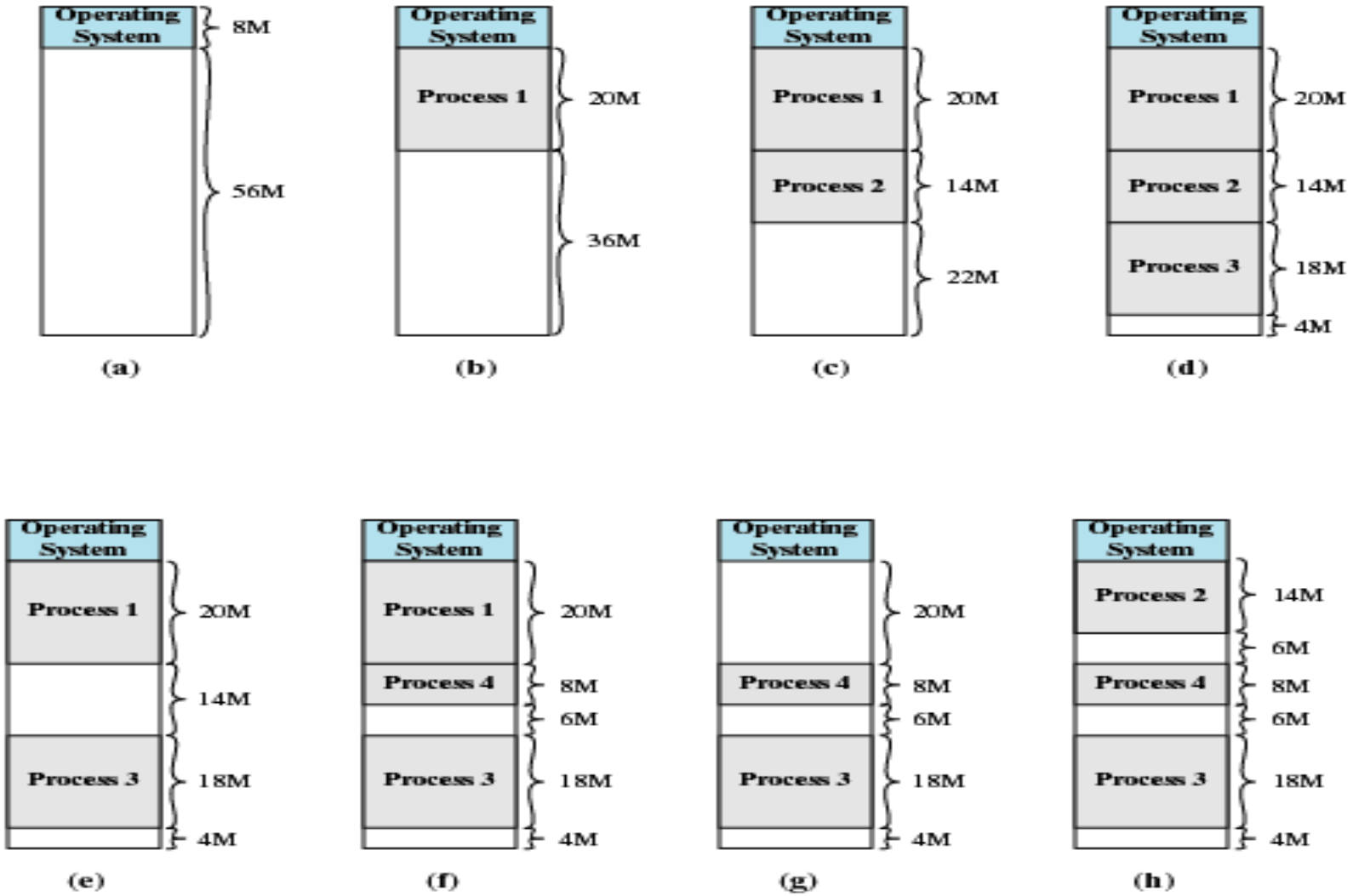


(b) Unequal-size partitions

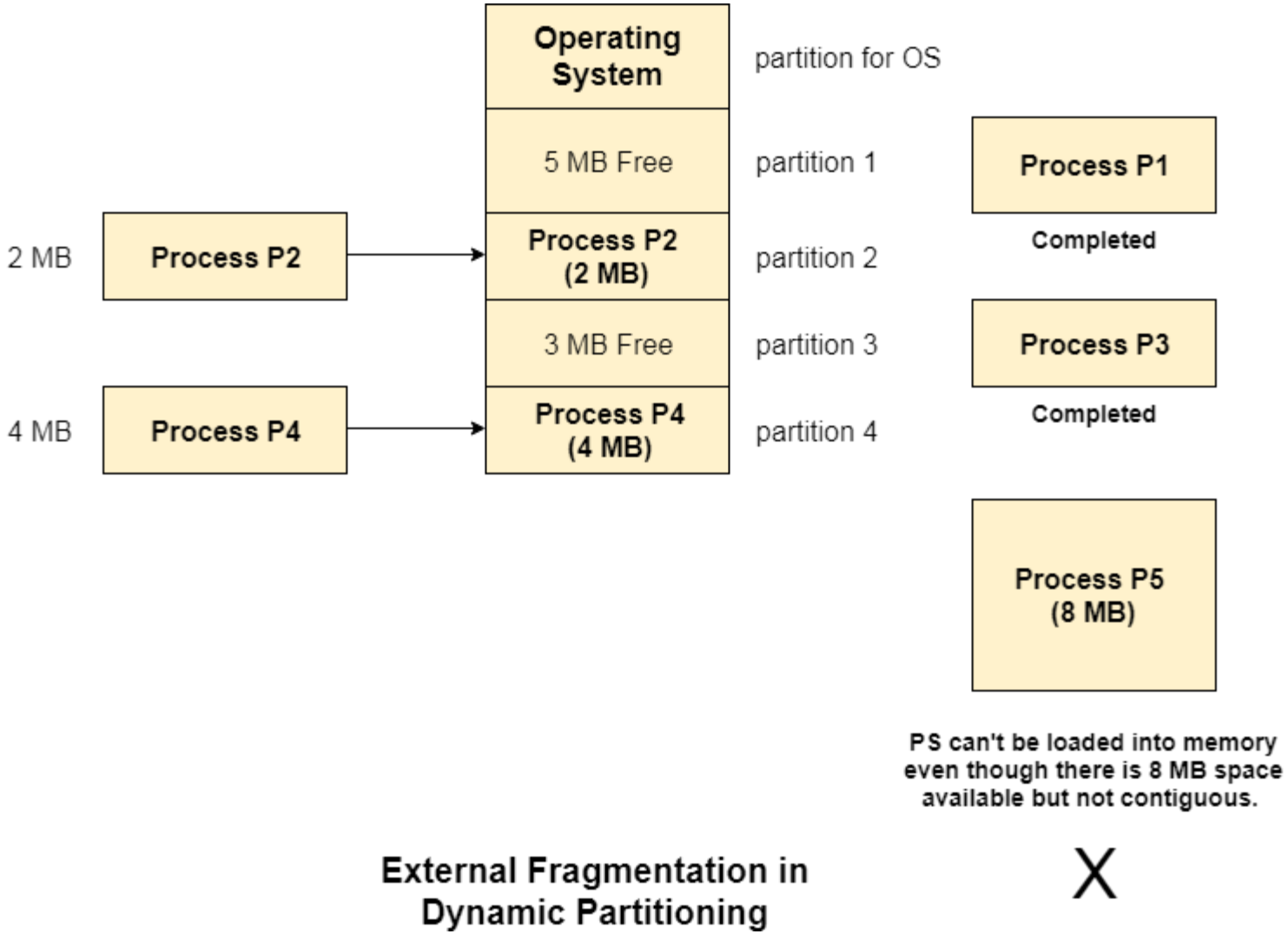
Dynamic Partition

- Partitions are of variable length and number
- Process is allocated exactly as much memory as required
- Eventually free different slots (hole) are not enough size to satisfy the request in the memory.
- This is called external fragmentation
- Must use compaction to shift processes so they are contiguous and all free memory is in one block

Dynamic Partition



Dynamic Partition



External Fragmentation in Dynamic Partitioning

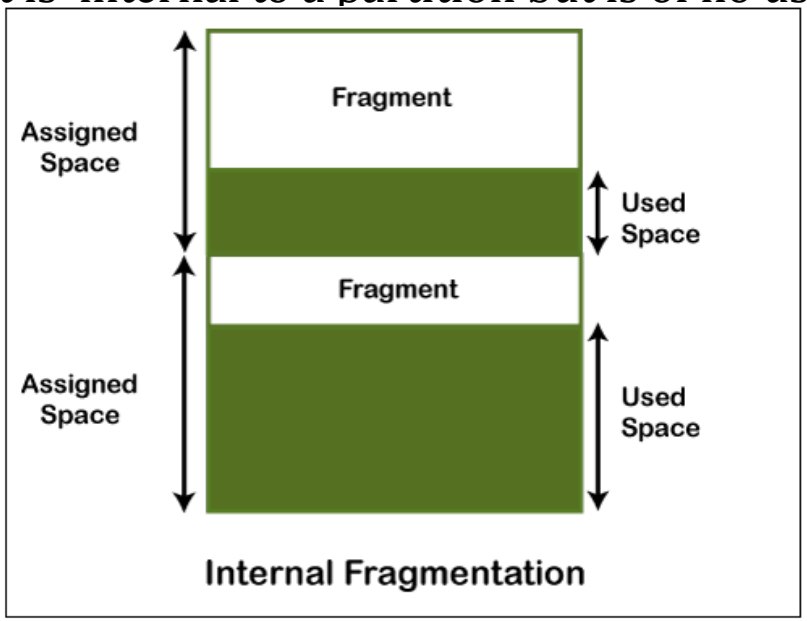
Fragmentation

- Fragmentation occurs in a dynamic memory allocation system when most of the free blocks are too small to satisfy any request.
- It is generally termed as inability to use the available memory.
- It is divided in to following categories:-
  1) Internal Fragmentation
  2) External Fragmentation

Fragmentation

(1) Internal Fragmentation

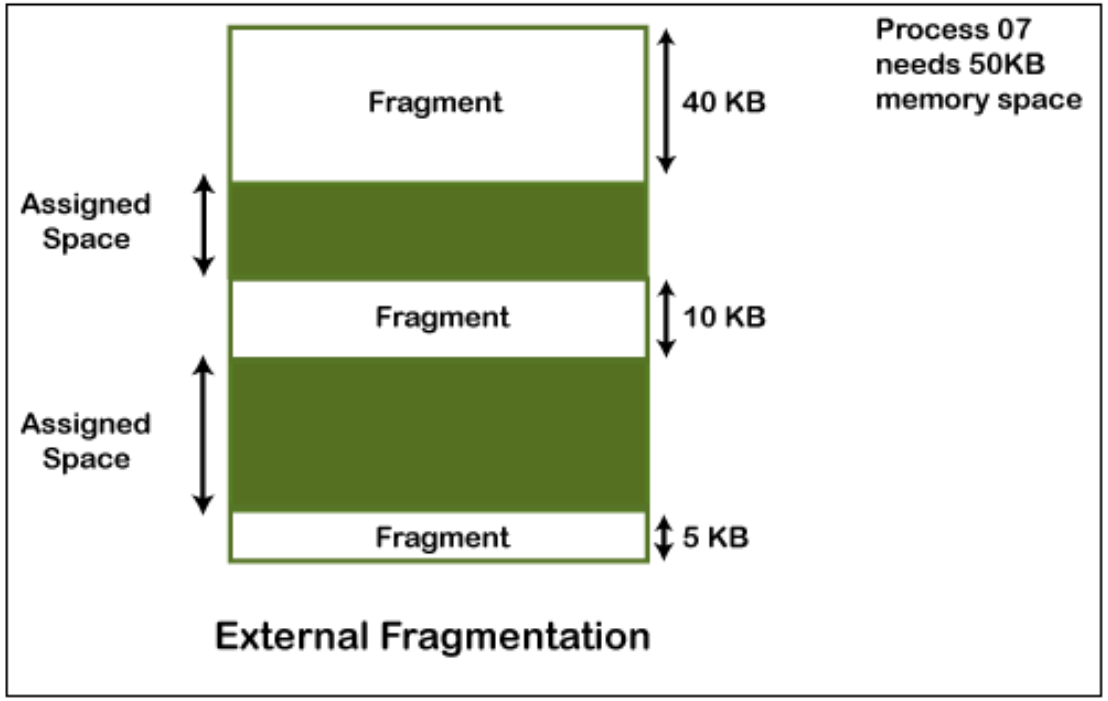
- At times the physical memory is broken into fixed size blocks and memory is allocated in unit of block sizes.
- The memory allocated to a space may be slightly larger than the requested memory.
- The difference between allocated and required memory is known as Internal fragmentation i.e. the memory that is internal to a partition but is of no use.



Fragmentation

(2) External Fragmentation

- In such situation processes are loaded and removed from the Memory.
- As a result of this, free holes exists to satisfy a request but is non contiguous i.e. the memory is fragmented into large no. of small holes.
- This phenomenon is known as External Fragmentation.



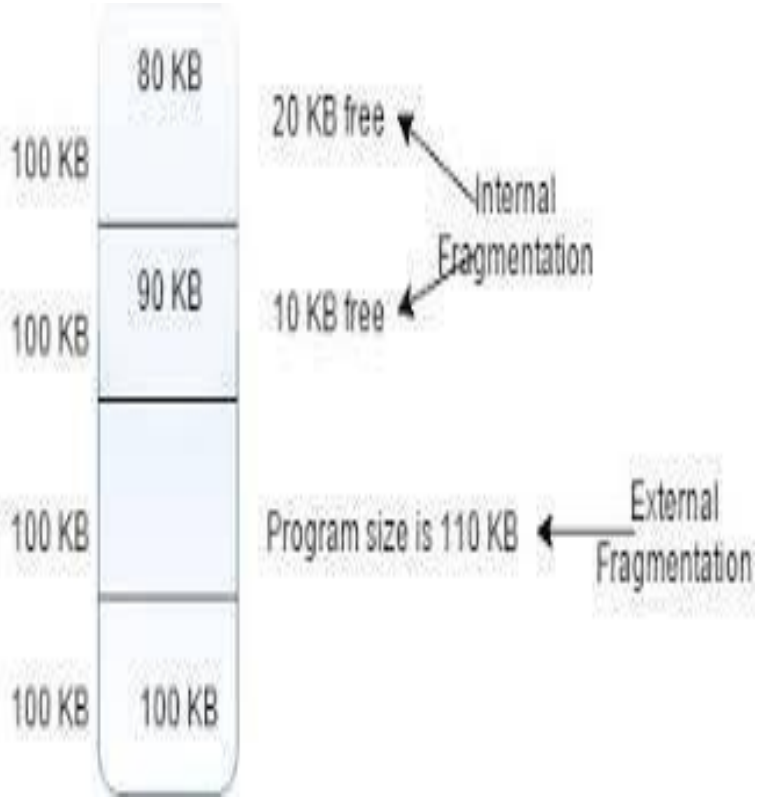
Fragmentation

1. Internal Fragmentation:-

Internal fragmentation is the wasted space within each allocated block.

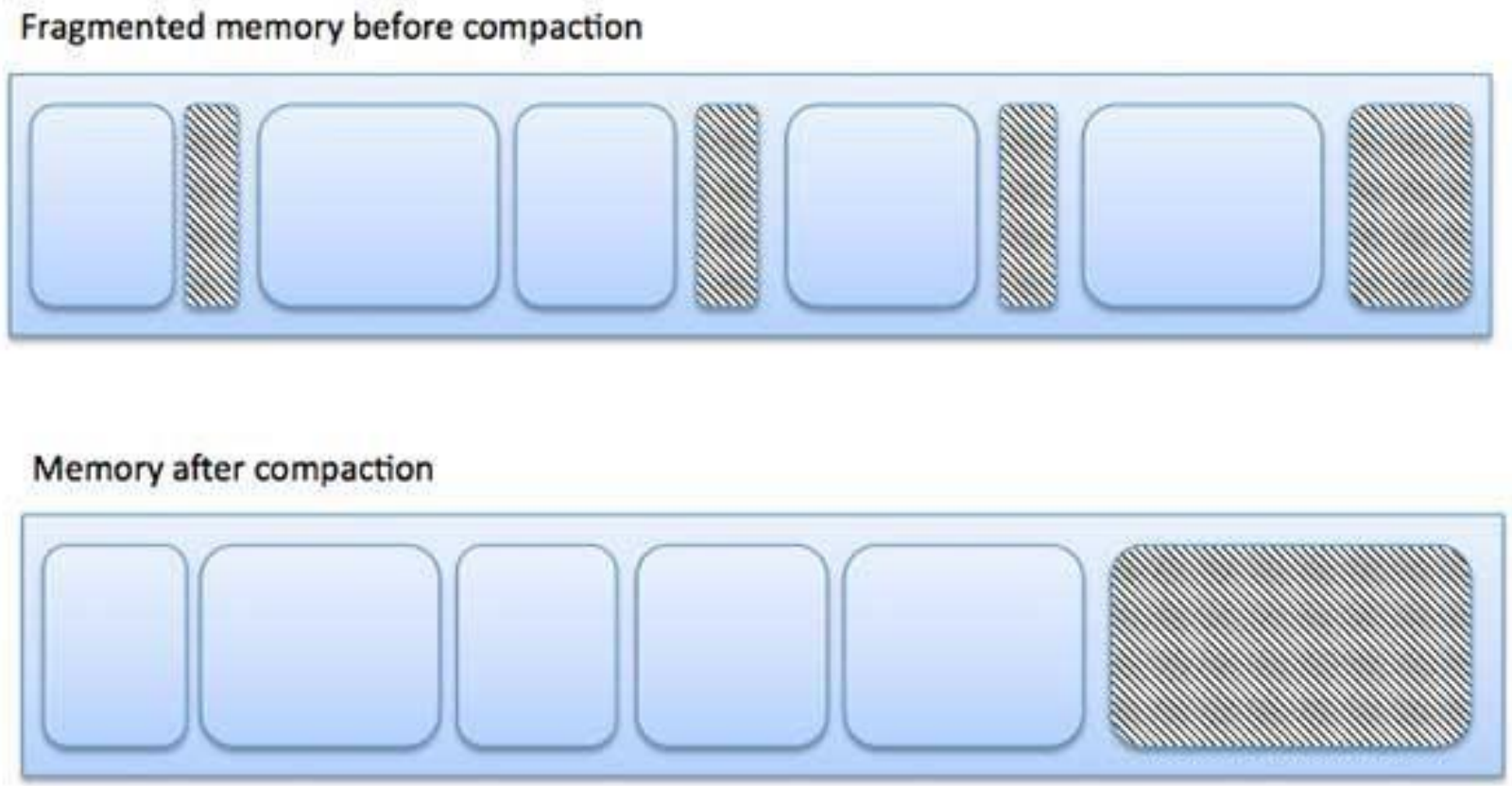
2. External Fragmentation:-

External fragmentation is the various free spaced holes generated in memory that are not able to satisfy the request



Compaction

- Compaction is a process in which the free space is collected in a large memory chunk to make some space available for processes.



Dynamic Storage Allocation Strategies

Memory Management Algorithms

OR

Contiguous Memory Allocation Technique

OR

Partitioning Algorithms

Dynamic Storage Allocation Strategies

1) First fit
2) Best fit
3) Worst fit

First Fit

- This method keeps the free/busy list of jobs organized by memory location, low-ordered to high-ordered memory.
- In this method, first job claims the first available memory with space more than or equal to it’s size.
- The operating system doesn’t search for appropriate partition but just allocate the job to the nearest memory partition available with sufficient size.

First Fit

Job Number	Memory Requested
J1	20 K
J2	200 K
J3	500 K
J4	50 K

Memory location	Memory block size	Job number	Job size	Status	Internal fragmentation
10567	200 K	J1	20 K	Busy	180 K
30457	30 K			Free	30
300875	700 K	J2	200 K	Busy	500 K
809567	50 K	J4	50 K	Busy	None
Total available :	980 K	Total used :	270 K		710 K

First Fit

Advantages of First-Fit Memory Allocation:

- It is fast in processing.
- As the processor allocates the nearest available memory partition to the job, it is very fast in execution.

Disadvantages of First-Fit Memory Allocation :

- It wastes a lot of memory.
- The processor ignores if the size of partition allocated to the job is very large as compared to the size of job or not.
- It just allocates the memory. As a result, a lot of memory is wasted and many jobs may not get space in the memory, and would have to wait for another job to complete.

Best Fit

- This method keeps the free/busy list in order by size – smallest to largest.
- In this method, the operating system first searches the whole of the memory according to the size of the given job and allocates it to the closest-fitting free partition in the memory, making it able to use memory efficiently.
- Here the jobs are in the order from smallest job to largest job.

Best Fit

Job Number	Memory Requested
J1	20 K
J2	200 K
J3	500 K
J4	50 K

Memory location	Memory block size	Job number	Job size	Status	Internal fragmentation
10567	30 K	J1	20 K	Busy	10 K
30457	50 K	J4	50 K	Busy	None
300875	200 K	J2	200 K	Busy	None
809567	700 K	J3	500 K	Busy	200 K
Total available :	980 K	Total used :	770 K		210 K

Best Fit

Advantages of Best-Fit Allocation :

- Memory Efficient.
- The operating system allocates the job minimum possible space in the memory, making memory management very efficient.
- To save memory from getting wasted, it is the best method.

Disadvantages of Best-Fit Allocation :

- It is a Slow Process.
- Checking the whole memory for each job makes the working of the operating system very slow.
- It takes a lot of time to complete the work

Worst Fit

- In this allocation technique, the process traverses the whole memory and always search for the largest hole/partition, and then the process is placed in that hole/partition.
- It is a slow process because it has to traverse the entire memory to search the largest hole.

Worst Fit

Advantages of Worst-Fit Allocation :

- Since this process chooses the largest hole/partition, therefore there will be large internal fragmentation.
- Now, this internal fragmentation will be quite big so that other small processes can also be placed in that leftover partition.

Disadvantages of Worst-Fit Allocation :

It is a slow process because it traverses all the partitions in the memory and then selects the largest partition among all the partitions, which is a time-consuming process.

Worst Fit

Process Number	Process Size
P1	30K
P2	100K
P3	45K

MEMORY LOCATION	MEMORY BLOCK SIZE	PROCESS NUMBER	PROCESS SIZE	STATUS	INTERNAL FRAGMENTATION
12345	50K	P3	45K	Busy	5K
45871	100K	P2	100K	Busy	None
1245	400K	P1	30K	Busy	370K
TOTAL AVAILABLE:	550K	TOTAL USED:	175K		375K

Virtual Memory

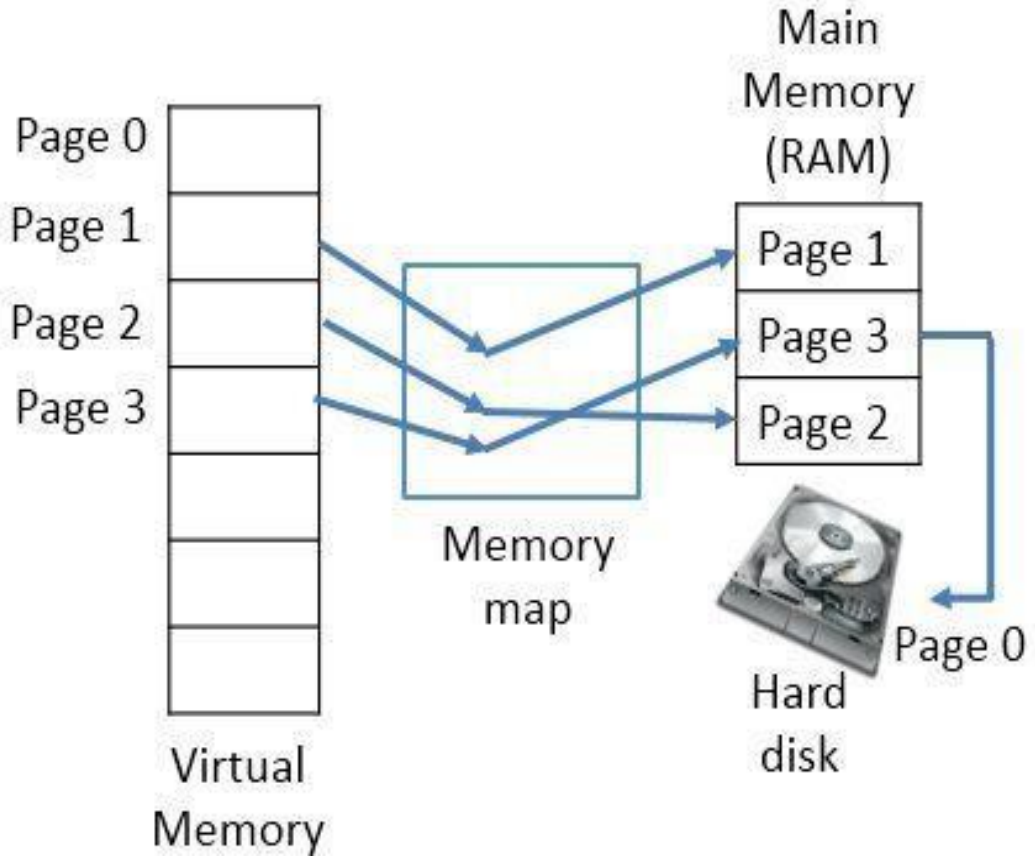
- Virtual Memory is a storage scheme that provides user an illusion of having a very big main memory. This is done by treating a part of secondary memory as the main memory.
- In this scheme, User can load the bigger size processes than the available main memory by having the illusion that the memory is available to load the process.
- Instead of loading one big process in the main memory, the Operating System loads the different parts of more than one process in the main memory.
- By doing this, the degree of multiprogramming will be increased and therefore, the CPU utilization will also be increased.

Virtual Memory

- During their execution, only the required pages or portions of processes are loaded into the main memory.
- This technique is useful as large virtual memory is provided for user programs when a very small physical memory is there.
- **The operating system keeps those parts of the program currently in use in main memory, and the rest on the disk.**
- In an Operating system, the memory is usually stored in the form of units that are known as **pages**.
- Basically, these are atomic units used to store large programs.

Virtual Memory

- These pages are mapped onto the physical memory but, to run the program, all pages are not required to be present in the physical memory.



Paging

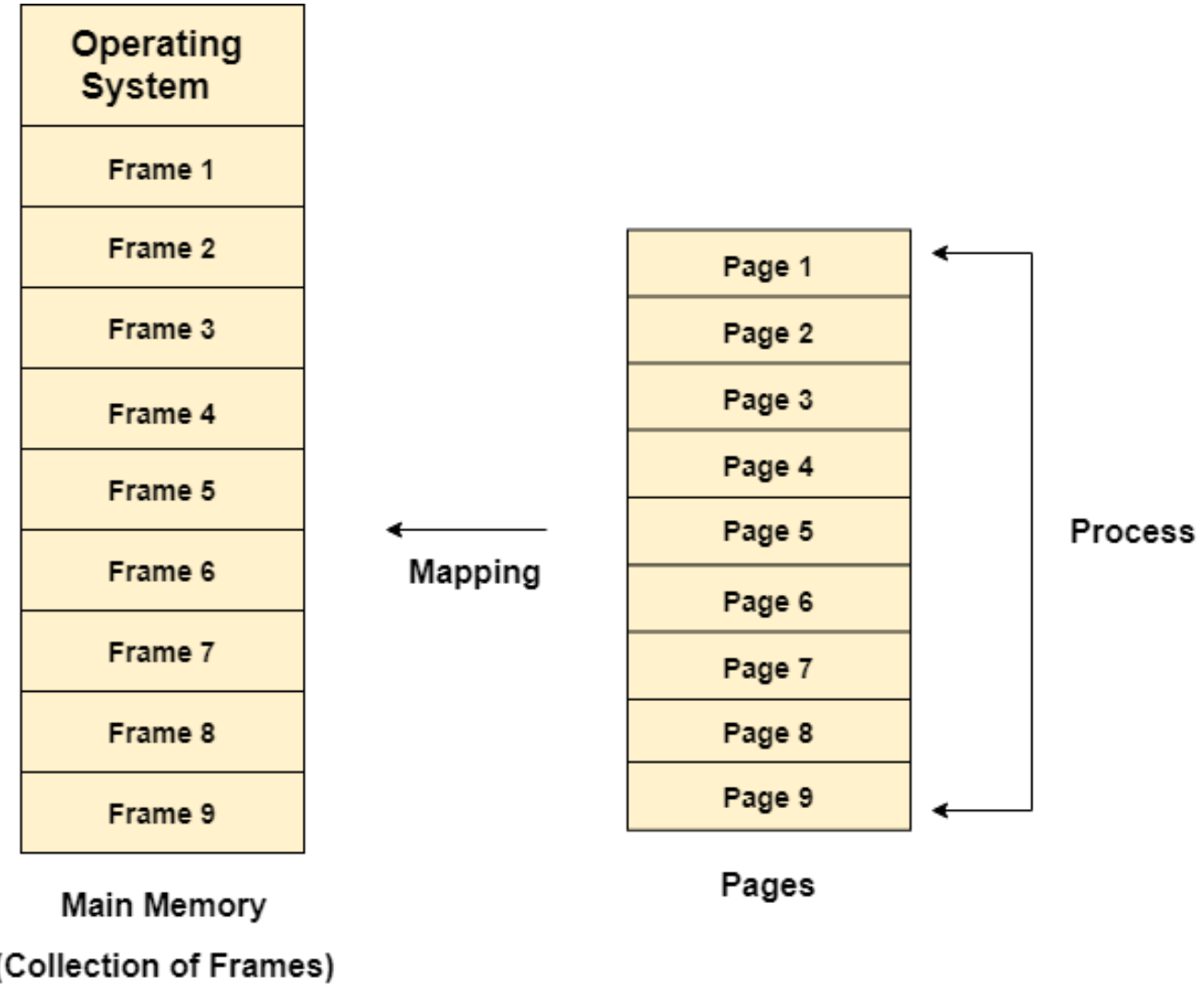
- Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory.
- This scheme permits the physical address space of a process to be non – contiguous.
- Divide logical memory into blocks of same size called **pages.**
- Divide physical memory into fixed-sized blocks called **frames**.
- Keep track of all free frames.
- **To run a program of size *n* pages, need to find *n* free frames.**

Paging

- The main idea behind the paging is to divide each process in the form of pages.
- The main memory will also be divided in the form of frames.
- One page of the process is to be stored in one of the frames of the memory.
- The pages can be stored at the different locations of the memory but the priority is always to find the contiguous frames or holes.
- Pages of the process are brought into the main memory only when they are required otherwise they reside in the secondary storage.

Paging

- Different operating system defines different frame sizes. The sizes of each frame must be equal. Considering the fact that the pages are mapped to the frames in Paging, page size needs to be as same as frame

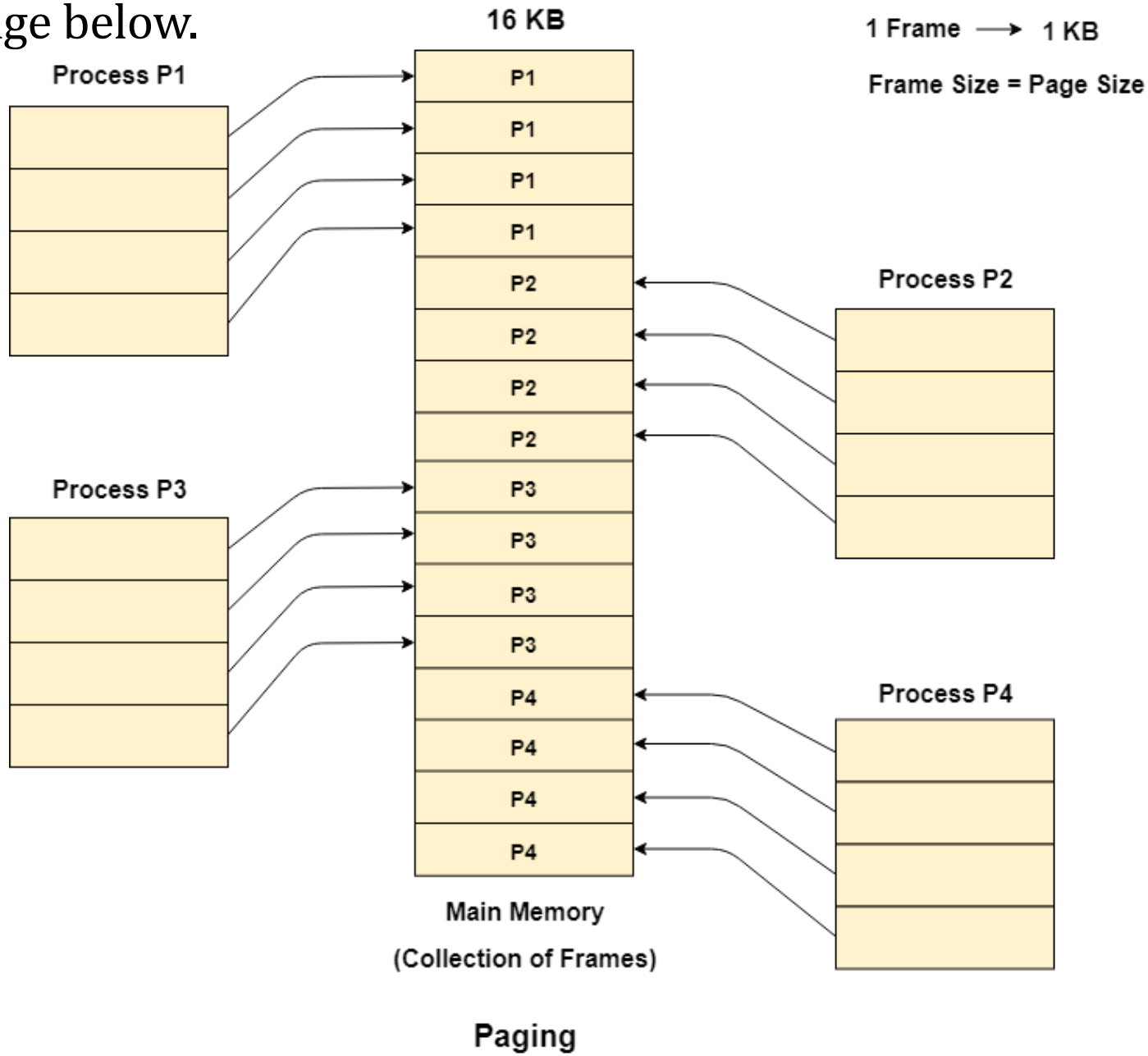


Paging - Example

- Let us consider the main memory size 16 Kb and Frame size is 1 KB therefore the main memory will be divided into the collection of 16 frames of 1 KB each.
- There are 4 processes in the system that is P1, P2, P3 and P4 of 4 KB each.
- Each process is divided into pages of 1 KB each so that one page can be stored in one frame.
- Initially, all the frames are empty therefore pages of the processes will get stored in the contiguous way.

Paging - Example

- Frames, pages and the mapping between the two is shown in the image below.

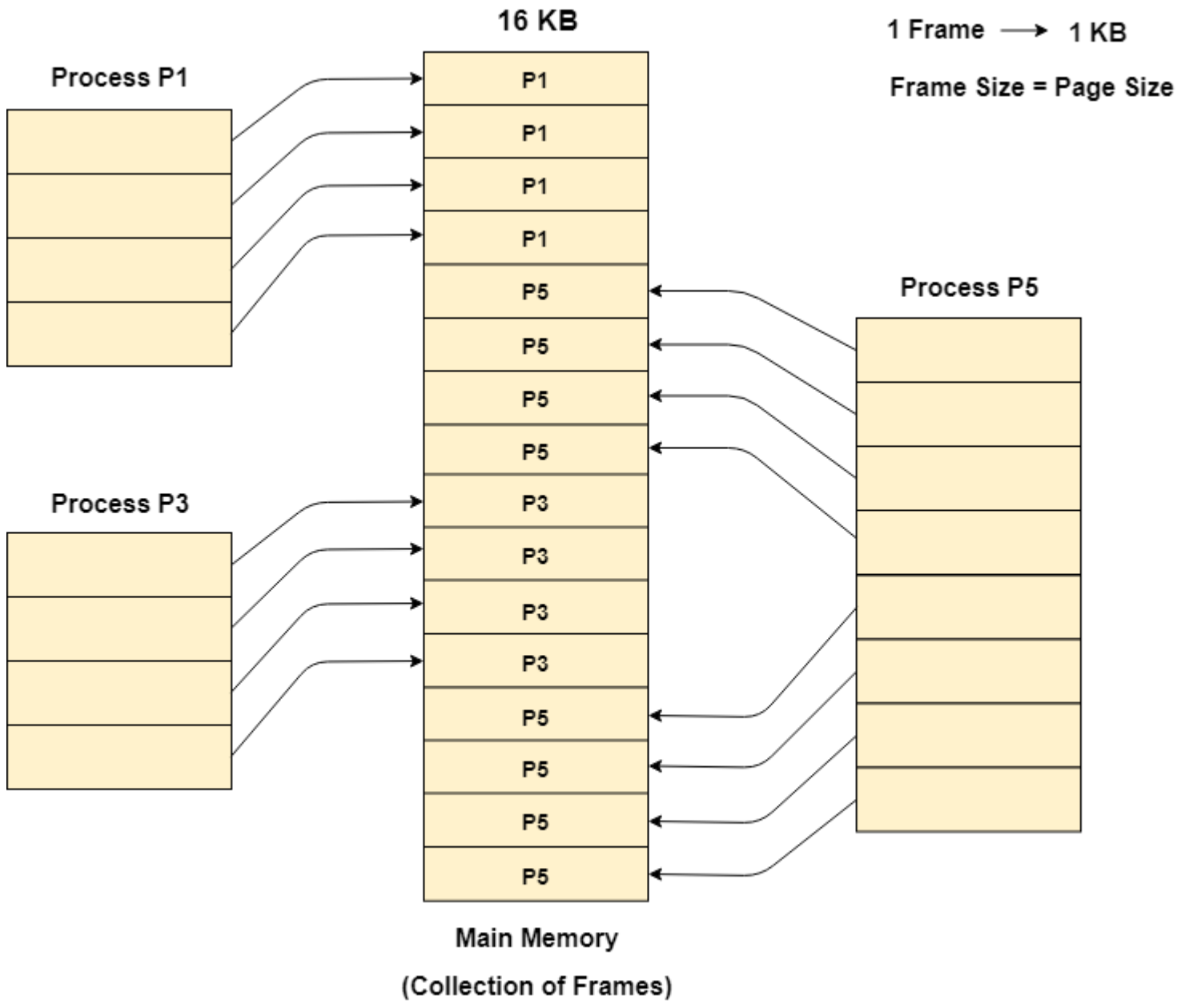


Paging

Paging - Example

- Let us consider that, P2 and P4 are moved to waiting state after some time.
- Now, 8 frames become empty and therefore other pages can be loaded in that empty place.
- The process P5 of size 8 KB (8 pages) is waiting inside the ready queue.
- Given the fact that, we have 8 non contiguous frames available in the memory and paging provides the flexibility of storing the process at the different places.
- Therefore, we can load the pages of process P5 in the place of P2 and P4.

Paging - Example

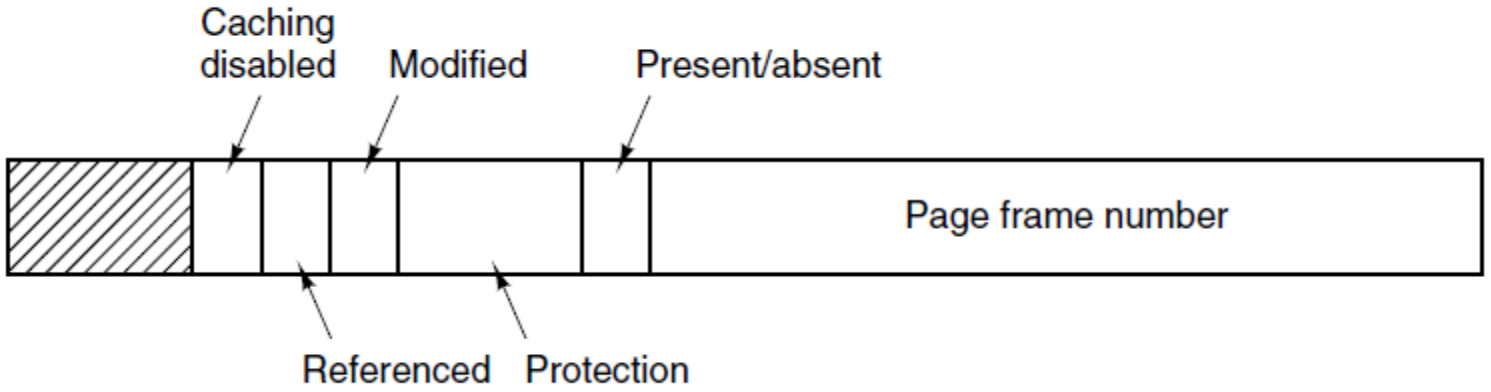


Paging

Page Table

- **“Page table** is a data structure which translates virtual address into equivalent physical address”.
- The **virtual page number** is used as an **index** into the Page table to find the entry for that virtual page and from the Page table physical page frame number is found.
- Thus the purpose of page table is to map virtual pages onto page frames.

Page Table



- Typical fields of page table are:-
  - Page frame number,
  - Present / absent bit,
  - Protection bits,
  - Modified field(Dirty bit),
  - Referenced field,
  - Cashing disabled field

Page Table

1) **Page frame Number:**

This field shows the corresponding physical page frame number for a particular virtual page.

1) **Present/Absent bit:**

If this bit is **1**, the entry is **valid** and can be used.

if this bit is **0**, the virtual page to which the entry belongs is **not currently in memory**. Accessing page table entry with this bit set to 0 causes a page fault.

3) **The Protection bits:**

This tells what kind of access is permitted.

If this bit is **1**, page is read only.

If this bit is **0**, page has read/write permission.

Page Table

4) Modified bit:

- If this bit is **1**, If the page in it has been **modified** (i.e., is "dirty"),

  it must be written back to the disk.

- If this bit is **0**, If it has **not** been **modified** (i.e., is "clean"),

5) Referenced bit:

A references bit is set whenever a page is referenced, either for reading or writing.

Its value helps operating system in page replacement algorithm.

Page Table

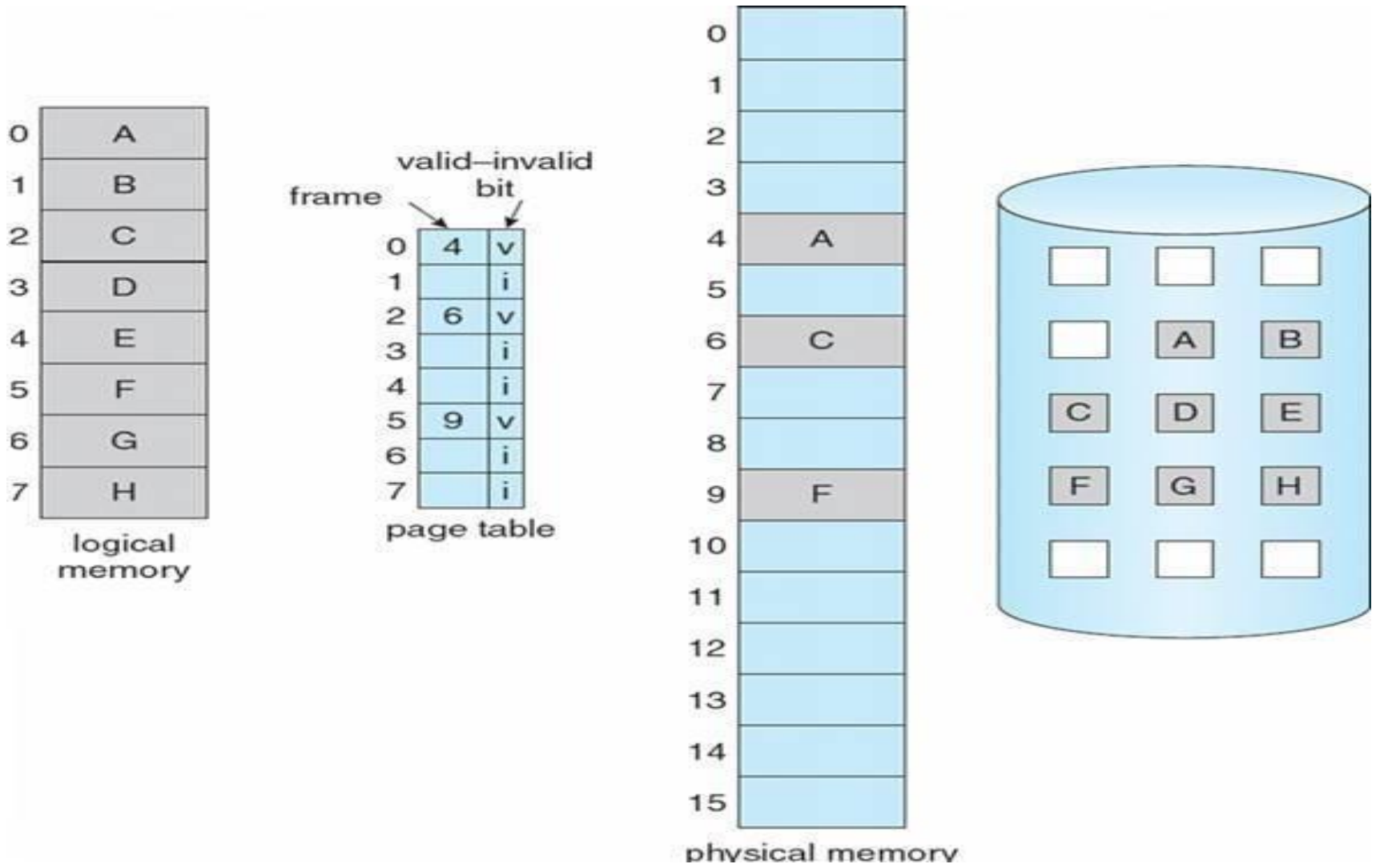
6) Cashing Disabled bit:

- With this bit cashing can be turned off.
- If this bit is **1**, then caching is turned off.
- If this bit is **0**, then caching is turned on.

Page Fault

- A page fault occurs when a program attempts to access data or code that is in its address space, but is not currently located in the system RAM.
- A **page fault** is a type of exception raised by computer hardware when a running program accesses a memory page that is not currently in main memory (mapped by the memory management unit(MMU)).
- The interrupt triggers the operating system to fetch the data from a virtual memory and load it into RAM.

Page Fault


0 A
1 B
2 C
3 D
4 E
5 F
6 G
7 H
logical memory
valid–invalid bit
frame
0 4 v
1 i
2 6 v
3 i
4 i
5 9 v
6 i
7 i
page table
0
1
2
3
4 A
5
6 C
7
8
9 F
10
11
12
13
14
15
physical memory
A B
C D E
F G H

Page Fault Steps

1. If CPU try to refer a page that is currently not available in the main memory, it generates an interrupt indicating memory access fault.
2. The OS puts the interrupted process in a blocking state.
3. For the execution to proceed the OS must bring the required page into the memory.
4. The OS will search for the required page in the logical address space.
5. The required page will be brought from logical address space to physical address space.
6. The page replacement algorithms are used for the decision making of replacing the page in physical address space.

Page Fault Steps

7. The page table will updated accordingly.

8. The signal will be sent to the CPU to continue the program execution and it will place the process back into ready state.

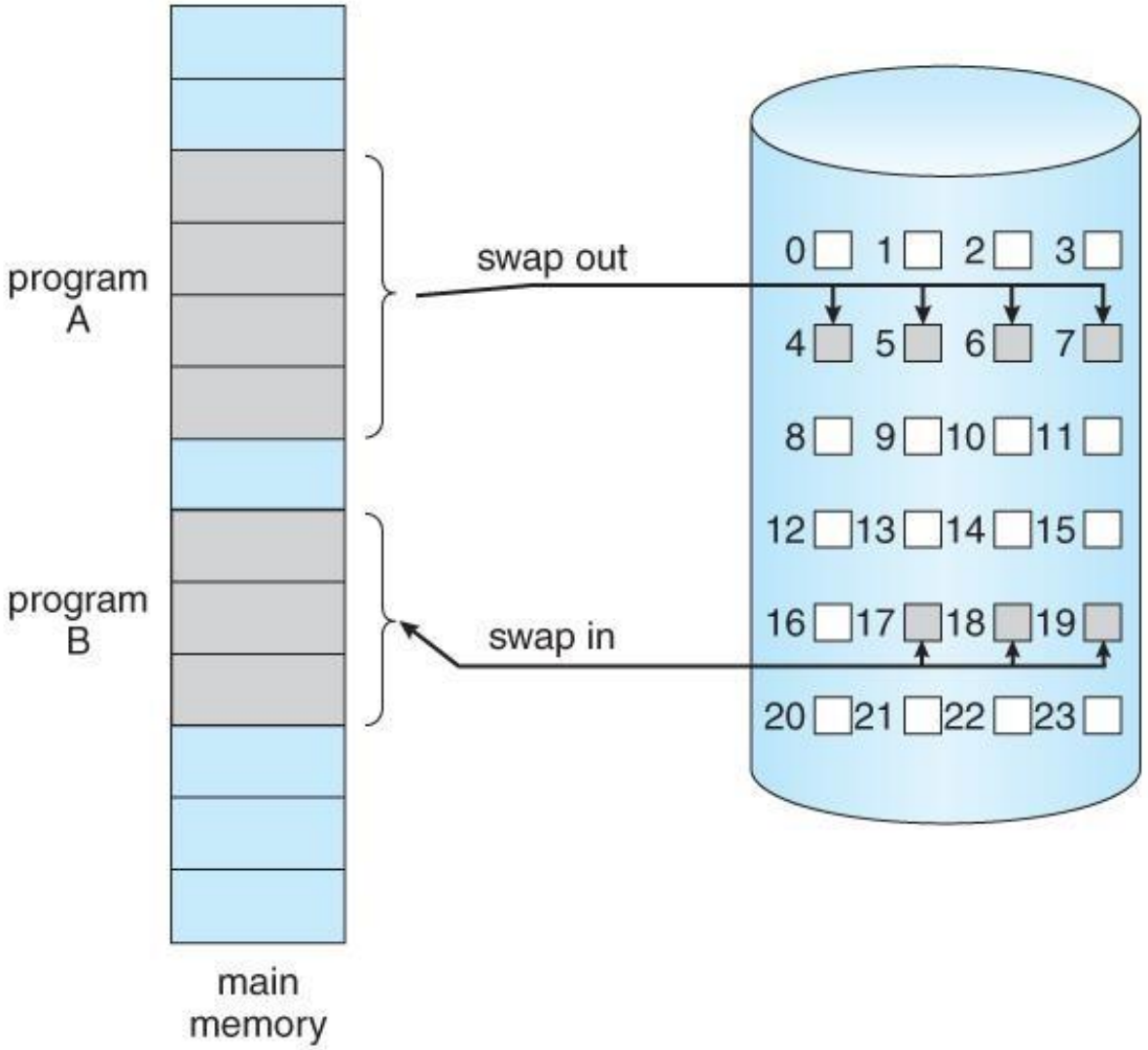
Example: Page Fault

- Total Frame Slot = 3
- Consider 1, 3, 0, 3, 5 page slots.
- Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots
- when 3 comes, it is already in memory so ➔ **0 Page Faults.**
- Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1. ➔ **1 Page Fault.**

Demand Paging

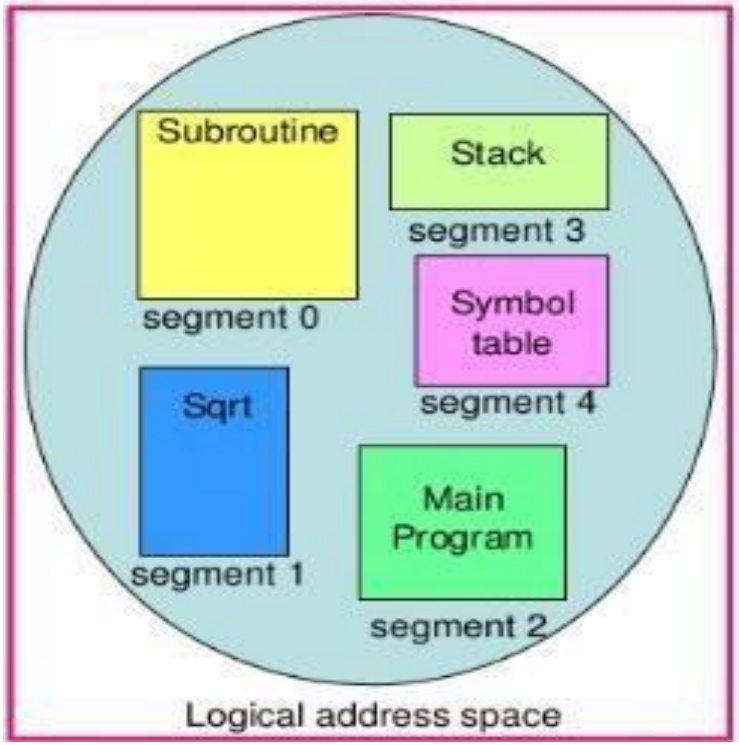
- The basic idea behind demand paging is that when a process is loaded in main memory, its all pages are not loaded in main memory.
- They are swapped in only when the process needs them (On demand).
- Initially only those pages are loaded which will be required by process immediately.
- “The pages are loaded only on demand, not in advance”.

Demand Paging



Segmentation

- A Memory Management technique in which memory is divided into variable sized chunks which can be allocated to processes.
- Each chunk is called a **Segment**.



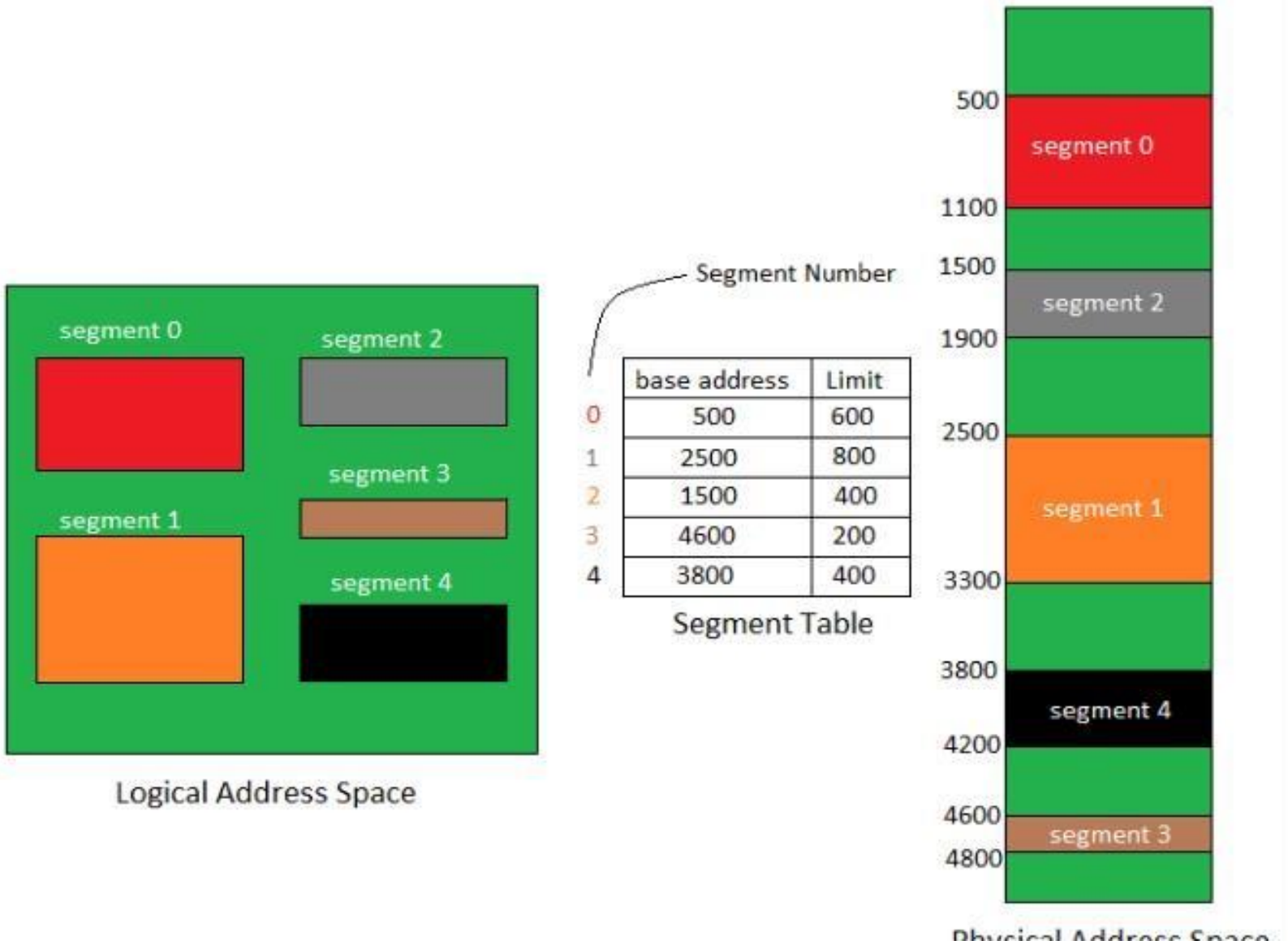
Segmentation

- 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.
- Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.
- A table stores the information about all such segments and is called **Segment Table.**

Segmentation

- **Segment Table:** It maps two dimensional Logical address into one dimensional Physical address.
- It's each table entry has
  - **Base Address:** It contains the starting physical address where the segments reside in memory.
  - **Limit:** It specifies the length of the segment.

Segmentation



Page Replacement Policy

PAGE REPLACEMENT

- When a page fault occurs, the operating system must choose a page to remove from memory to make space for the page that has to be brought in.

❑ Following are different types of **page replacement algorithms**

- FIFO Page Replacement Algorithm
- Optimal Page Replacement Algorithm
- LRU (Least Recently Used) Page Replacement Algorithm

FIFO

- It is a very simple way of Page replacement and is referred to as First in First Out.
- This algorithm mainly replaces the oldest page that has been present in the main memory for the longest time.
- This algorithm is implemented by keeping the track of all the pages in the queue.
- As new pages are requested and are swapped in, they are added to the tail of a queue and the page which is at the head becomes the victim.
- This is not an effective way of page replacement but it can be used for small systems.

FIFO

Advantages:-

- This algorithm is simple and easy to use.
- FIFO does not cause more overhead.

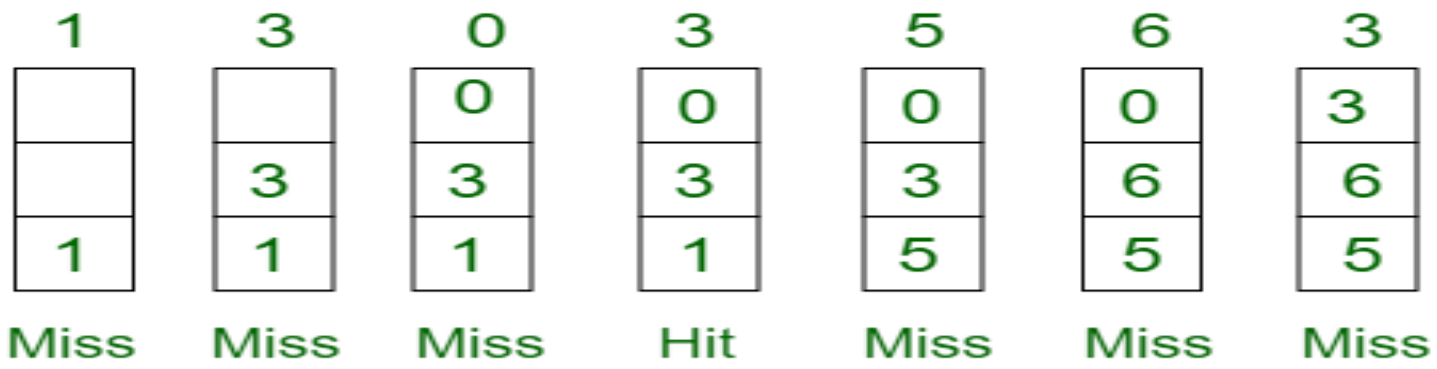
Disadvantages:-

- There is an increase in **page faults** as page frames increases.
- The performance of this algorithm is the worst.

FIFO Example

- Consider page reference string 1, 3, 0, 3, 5, 6 with 3 page frames.
- Find number of page faults.

Page reference 1, 3, 0, 3, 5, 6, 3



Total Page Fault = 6

FIFO Example

- Consider a reference string: 4, 7, 6, 1, 7, 6, 1, 2, 7, 2.
- the number of frames in the memory is 3. Find out the number of page faults

Request	4	7	6	1	7	6	1	2	7	2
Frame 3			6	6	6	6	6	6	7	7
Frame 2		7	7	7	7	7	7	2	2	2
Frame 1	4	4	4	1	1	1	1	1	1	1
Miss/Hit	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Miss	Miss	Hit

Number of Page Faults in FIFO = 6

Optimal Page Replacement

- This algorithm mainly replaces the page that will not be used for the longest time in the future.
- The practical implementation of this algorithm is not possible.
- Practical implementation is not possible because we cannot predict in advance those pages that will not be used for the longest time in the future.
- This algorithm leads to less number of page faults and thus is the best-known algorithm.
- Also, this algorithm can be used to measure the performance of other algorithms.

Optimal Page Replacement

Advantages of OPR

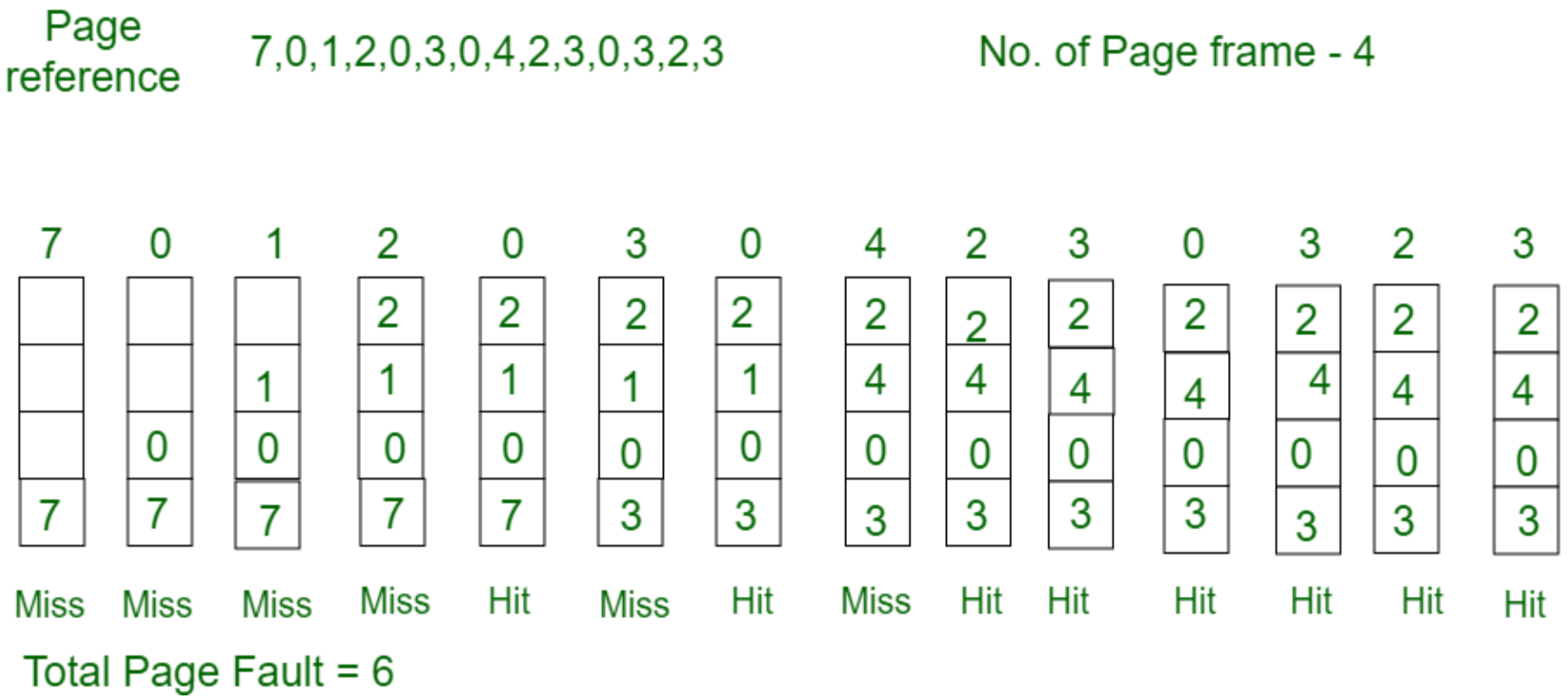
- This algorithm is easy to use.
- This algorithm provides excellent efficiency and is less complex.
- For the best result, the implementation of data structures is very easy.

Disadvantages of OPR

- In this algorithm future awareness of the program is needed.
- Practical Implementation is not possible because the operating system is unable to track the future request

Optimal Page Replacement Example:

- Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, with 4 page frame.
- Find number of page fault.



Optimal Page Replacement Example:

- Consider a reference string: 4, 7, 6, 1, 7, 6, 1, 2, 7, 2.
- the number of frames in the memory is 3.
- Find out the number of page faults

Request	4	7	6	1	7	6	1	2	7	2
Frame 3			6	6	6	6	6	2	2	2
Frame 2		7	7	7	7	7	7	7	7	7
Frame 1	4	4	4	1	1	1	1	1	1	1
Miss/Hit	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Miss	Hit	Hit

Number of Page Faults in Optimal Page Replacement Algorithm = 5

LRU – Least Recently Used

- This algorithm stands for "Least recent used" and this algorithm helps the Operating system to search those pages that are used over a short duration of time frame.
- In this algorithm page will be replaced which is least recently used.
- The page that has not been used for the longest time in the main memory will be selected for replacement.
- This algorithm is easy to implement.

LRU – Least Recently Used

Advantages of LRU:-

- It is an efficient technique.
- With this algorithm, it becomes easy to identify the faulty pages that are not needed for a long time.
- It helps in Full analysis.

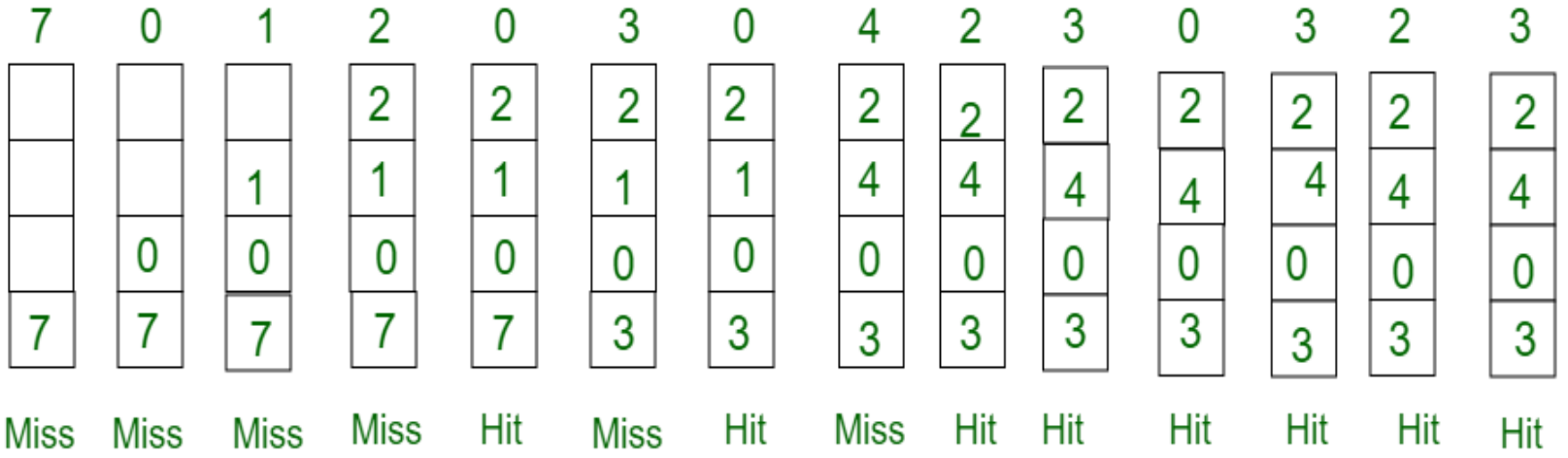
Disadvantages of LRU:-

- It is expensive and has more complexity.

LRU – Least Recently Used Example

- Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 with 4 page frames.
- Find number of page faults.

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

LRU – Least Recently Used Example

Consider a reference string: 4, 7, 6, 1, 7, 6, 1, 2, 7, 2. the number of frames in the memory is 3. Find out the number of page faults.

Request	4	7	6	1	7	6	1	2	7	2
Frame 3			6	6	6	6	6	6	7	7
Frame 2		7	7	7	7	7	7	2	2	2
Frame 1	4	4	4	1	1	1	1	1	1	1
Miss/Hit	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Miss	Miss	Hit

Number of Page Faults in LRU = 6

Thank You….