Unit - 4

Memory Management

Introduction

- Memory is a resource , needs to be pertitioned and allocated to the ready processes, such that both processor and memory can be utilized efficiently.
- It is partitioned into two parts : One for the OS and the other for the user area
- Memory management needs hardware support, means management techniques depends on the hardware available
- There are two types of memory menagement : real memory (Main Memory) and Virtual Memory

Static and Dynamic Allocation

- Memory allocation is generally performed through two methods : static allocation and dynamic allocation
- Static allocation is done before the execution of a process.
- There are two instances when this type of allocation is performed :

  1. when the location of the process in the memory is known at compile time, the compiler generates an absolute code for the process.
- If the location of the process needs to be changed on the memory , the code must be recompiled.

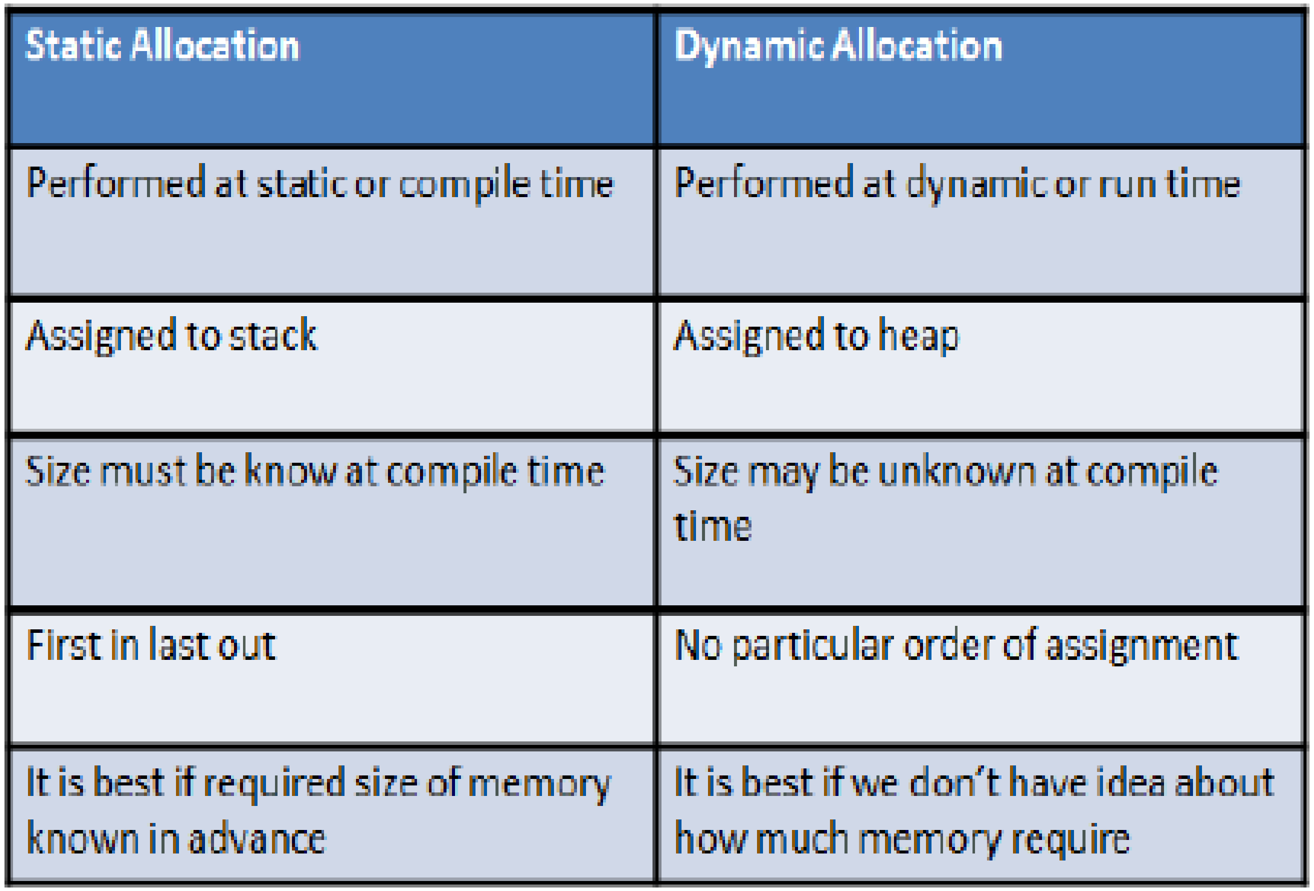
Static Allocation

- 2. when the location of the process in the memory is not known at compile time, the compiler does not produce an actual memory address bt generates a relocatable code that is, the addresses that are relative to some known point.
- With this relocatable code, it is easy to load the process to a changed location and there is no need to recompile the code.
- In both cases of static allocation, size should be known before start of the execution of the process.

Dynamic Allocation

- If memory allocation is deferred till the process starts executing, it is known as dynamic allocation.
- It means the process is loaded in memory initially with all the memory references in relative form.
- The absolute addresses in memory are calculated as an instruction in the process executed.
- In this way, memory allocation is done during execution of a program.
- Dynamic allocation also has the flexibility to allocate memory in any region.

Difference Between Static and Dyanamic Allocation



Static Allocation	Dynamic Allocation
Performed at static or compile time	Performed at dynamic or run time
Assigned to stack	Assigned to heap
Size must be know at compile time	Size may be unknown at compile time
First in last out	No particular order of assignment
It is best if required size of memory known in advance	It is best if we don't have idea about how much memory require

Logical and Physical Addresses

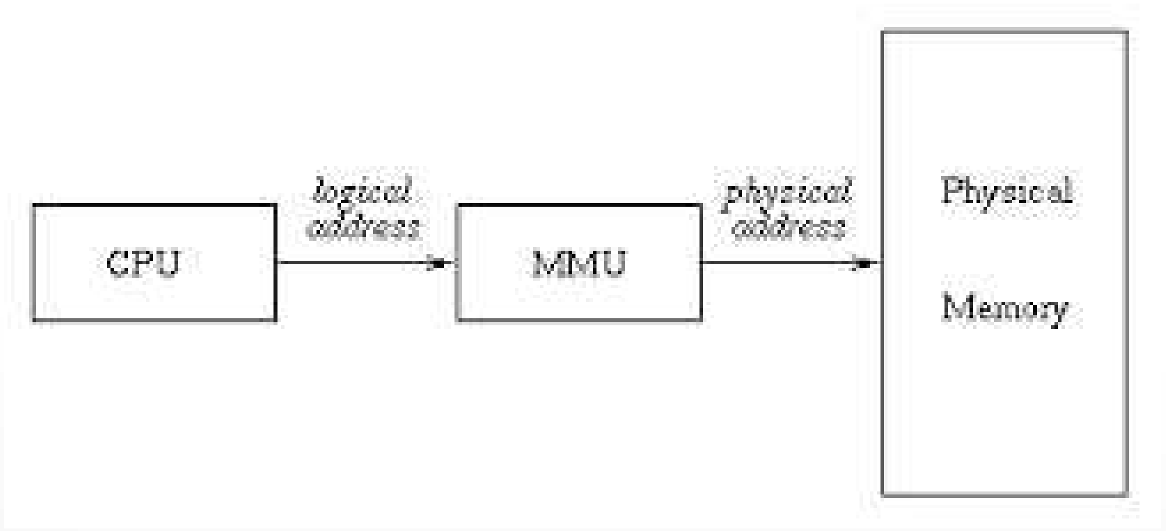
- In dynamic memory allocation, the place of allocation of the process is not known at the compile time or load time.
- In dynamic memory allocation two types of addresses are generated : Logical Address and Physical Address
- Logical Address : processor , at compile time , generates some addresses, known as logical address.
- The set of all logical addresses generated by the compilation of the process is known as logical address space.

Logical and Physical Addresses

- These 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 ofphysical addresses generated, corresponding to the logical addresses during process execution, is known as physical address space.
- There are now two types of memory: logical memory and physical memory. The logical memory views the memory from 0 to its maximum limit, say m.
- The user process generates only logical addresses in the range 0 to m, and a user thinks that the process runs in this logical address space.
- But the user process, in the form of logical address space, is converted into physical address space with a reference or base in memory.

Logical and Physical Addresses

- The memory management component of the OS performs this conversion of logical addresses into physical addresses.
- Thus, 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.

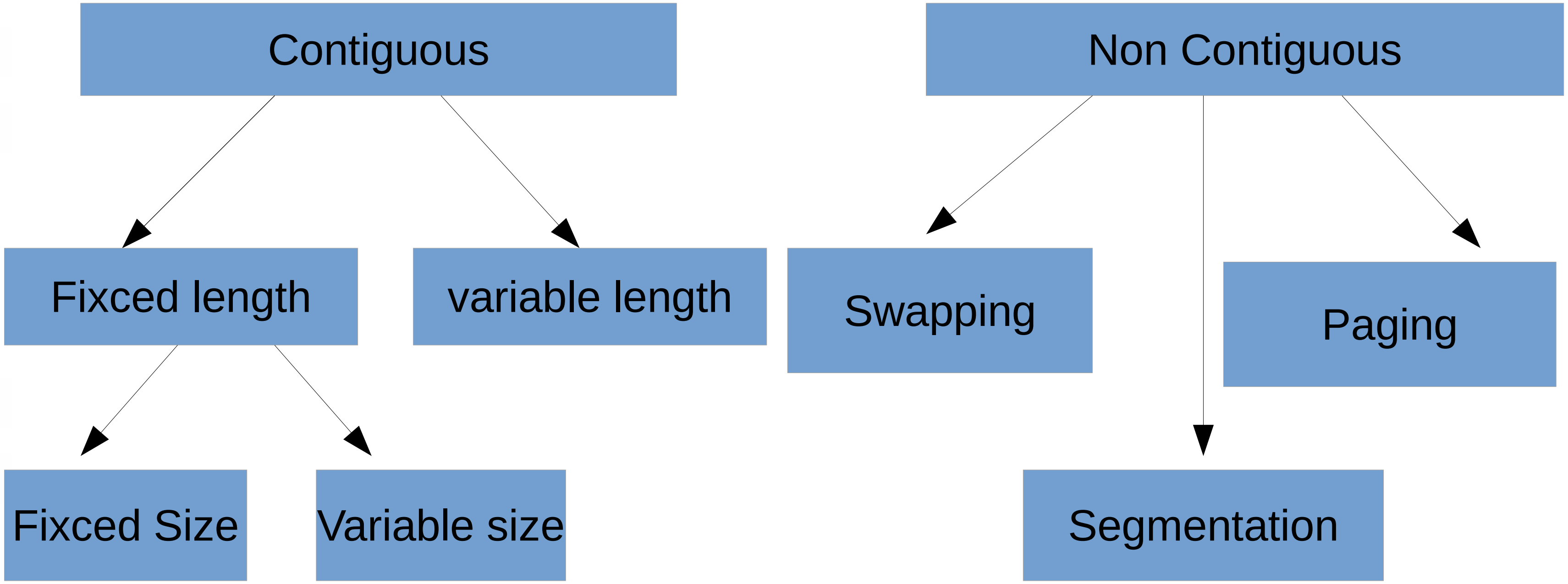


Memory Partition

- 2 types – Contiguous memory partition
    - Non Contiguous Memory partition

  Contiguous : Must exist as a sigle block of continuous address sometimes it is impossible to find large enough block.

- Low overhead on os
- Non Contiguous : Programs divided into chunks called segment
- Each segment can be place in a different parts of memeory as dynamic storage.
- Easier to find holes in which segment will fit.


Contiguous
Non Contiguous
Fixced length
variable length
Swapping
Paging
Fixced Size
Variable size
Segmentation

Fixed and Variable Memory Partitioning

- The memory space is divided into fixed or variable partitions.
- Fixed partitioning is a method of partitioning the memory at the time of system generation.
- In fixed partitioning, the partition size can be of fixed size as well as variable, but once fixed, it cannot be changed.
- Variable partitioning is not performed at the system generation time.
- In this partitioning, the number and size of the memory partition vary and are created at run time, by the OS.

Fixed and Variable Memory Partitioning

Figure 4: Fixed Partitioning

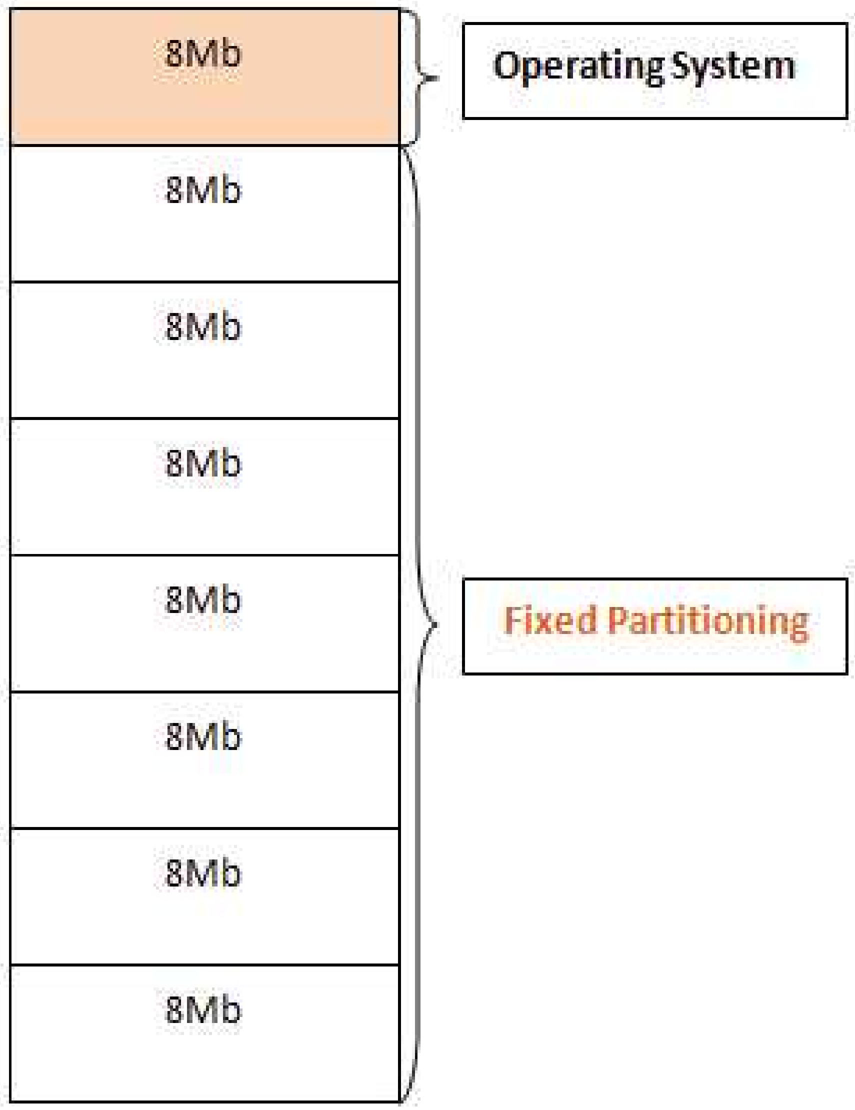
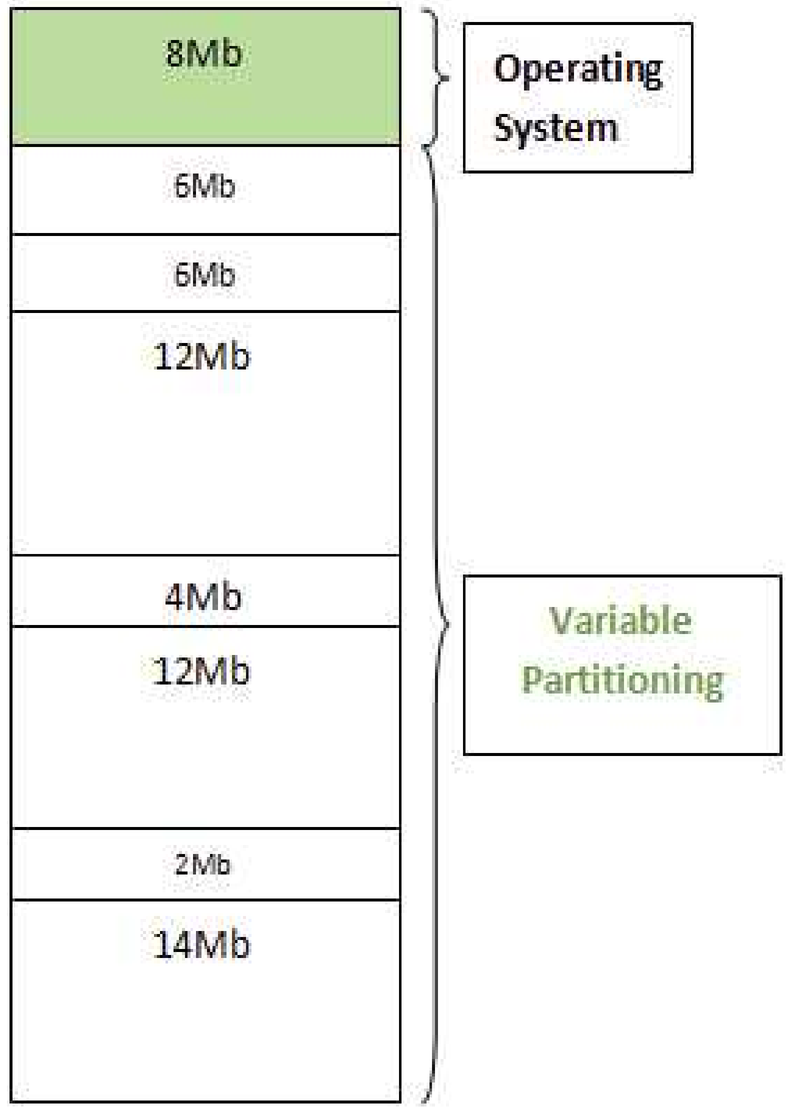
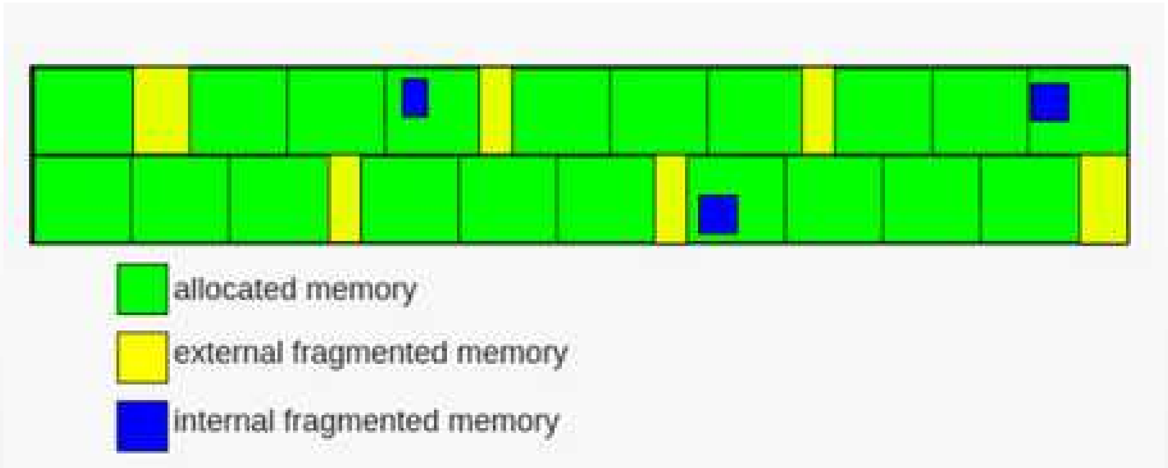


Figure 5: Variable Partitioning

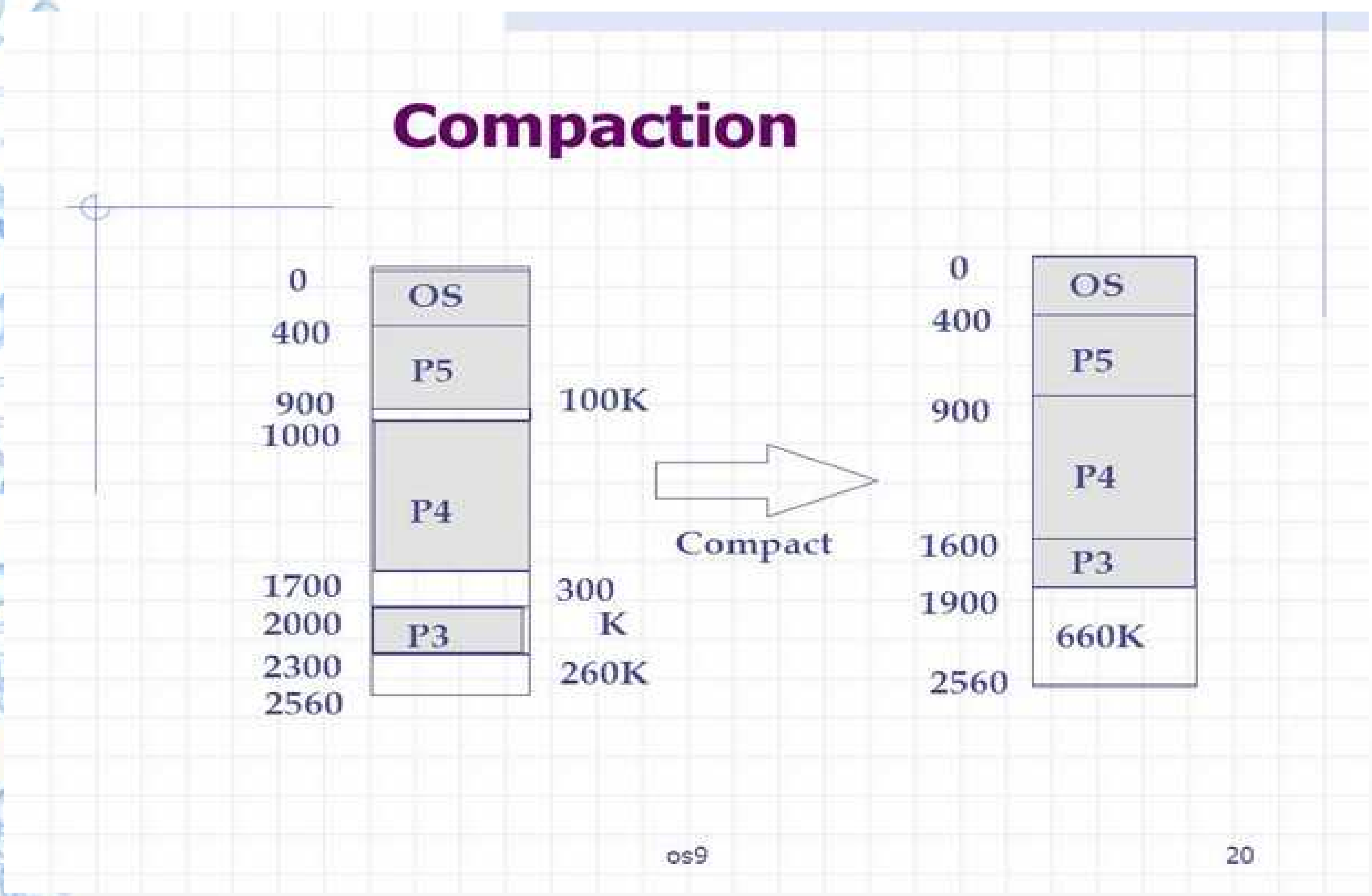


Fragmentation

- When a process is allocated to a partition, it may be possible that its size is less than the size of partition, leaving a space after allocation, which is unusable by any other process.
- This wastage of memory, internal to a partition, is known as internal fragmentation.
- While allocating and de-allocating memory to the processes in partitions through various methods, it may be possible that there are small spaces left in various partitions throughout the memory, such that if these spaces are combined, they may satisfy some other process' request.
- But these spaces cannot be combined. This total memory space fragmented, external to all the partitions, is known as external fragmentation.



Compaction



Compaction

Compaction

- External fragmentation in dynamic partitioning can be reduced, if all the small holes formed during partitioning and allocation, are compacted together.
- Compaction helps 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 occupied memory region is moved in one direction, and all unoccupied memory region in the other direction.
- This results in contiguous free holes, that is, a single large hole, which is then allocated to a desired process.

Compaction

- Compaction, however, is not always possible. One limitation is that it can be applied, only if the memory is relocated dynamically at the execution time, so that it is easy to move one
- process from one region to another. Another limitation is that compaction incurs cost, because it is time consuming and wastes CPU time.

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.
- In fixed partitioning with equal-sized partitions, these algorithms are not applicable, because all the partitions are of the same size and therefore, it does not matter which partition is selected.
- These algorithms, however, play a great role in fixed-partitioning with unequal-sized partitions and in dynamic partitioning, in terms of system performance and memory wastage.
- There are primarily three techniques for memory allocation.

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. The next time, searching is resumed from that location.
- The first hole is counted from this last location. In this case, it becomes the next-fit allocation.
- The first-fit algorithm does not take care of the memory wastage. It may be possible that the first-fit hole is very large, compared to the memory required by the process, resulting in wastage of memory.

Best-fit Allocation

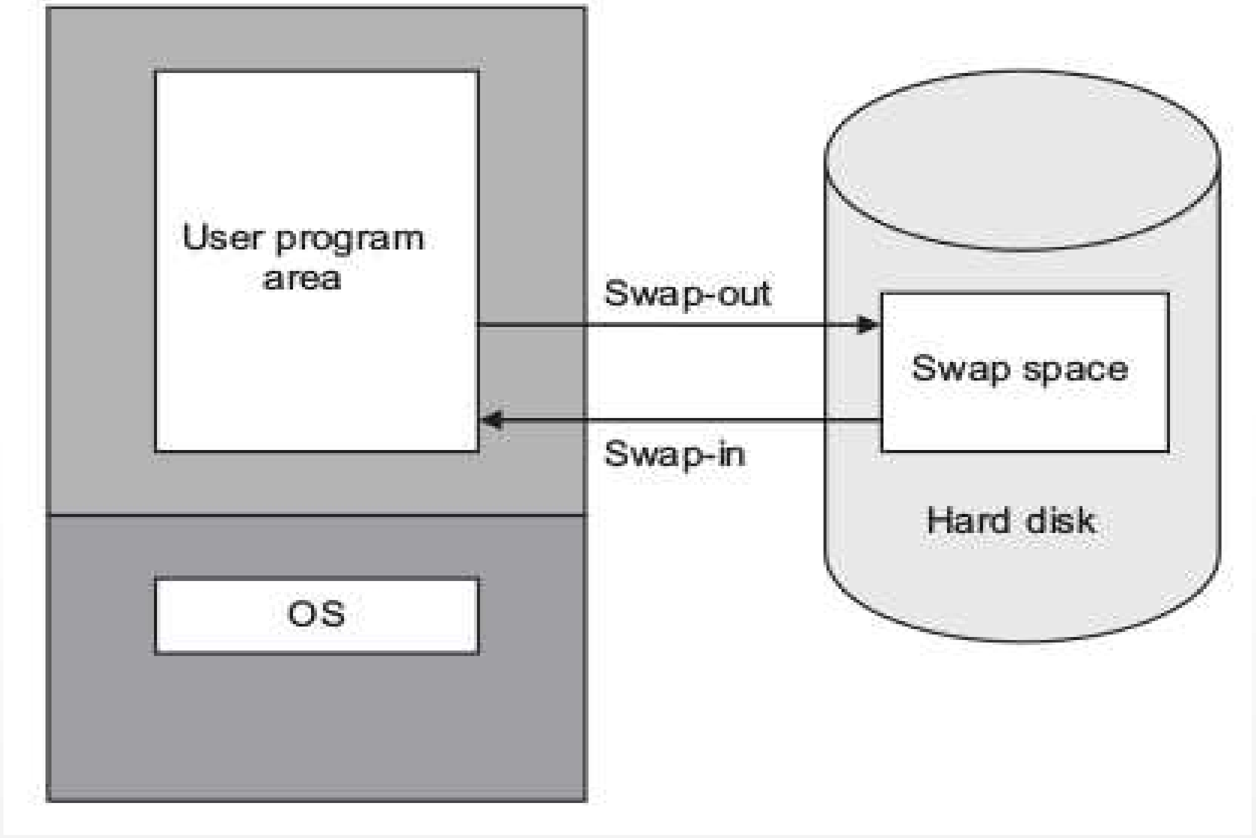
- This algorithm takes care of memory storage and searches the list, by comparing memory size of the process to be allocated with that of free holes in the list.
- The smallest hole that is big enough to accommodate the process is allocated.
- The best-fit algorithm may be better in terms of wastage of memory space, but it incurs cost of searching all the entries in the list.
- This is an additional overhead. Moreover, it also leaves small memory holes, causing internal fragmentation.

Worst-fit Allocation

- This algorithm is just reverse of the best-fit algorithm. It searches the list for the largest hole.
- This algorithm in its first perception seems that it is not a good algorithm, in terms of memory.
- But it may be helpful in dynamic partitioning, because the large holes, leftover due to this algorithm, may be allocated to the processes that fit.
- However, this algorithm incurs overhead of searching the list for the largest hole. It may also leave small holes, causing internal fragmentation.

Swapping

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



Swapping

- Where will this process reside?
- the help of any secondary storage (generally, hard disk) known as backing store, is taken, and the process is stored there.
- **The action of taking out a process from memory is called swap-out, and the process is known as a swapped-out process.**
- The action of bringing back the swapped-out processes into memory is called swap-in. **A separate space in the hard disk, known as swap space, is reserved for swapped-out processes.**
- The swap space should be large enough such that a swapped out process can be accommodated. The swap space stores the images of all swapped out processes.
- Thus, whenever a process is selected by the scheduler to execute, the dispatcher checks whether or not the desired process is in the ready queue. If not, a process is swapped out of memory and the desired process is swapped in.

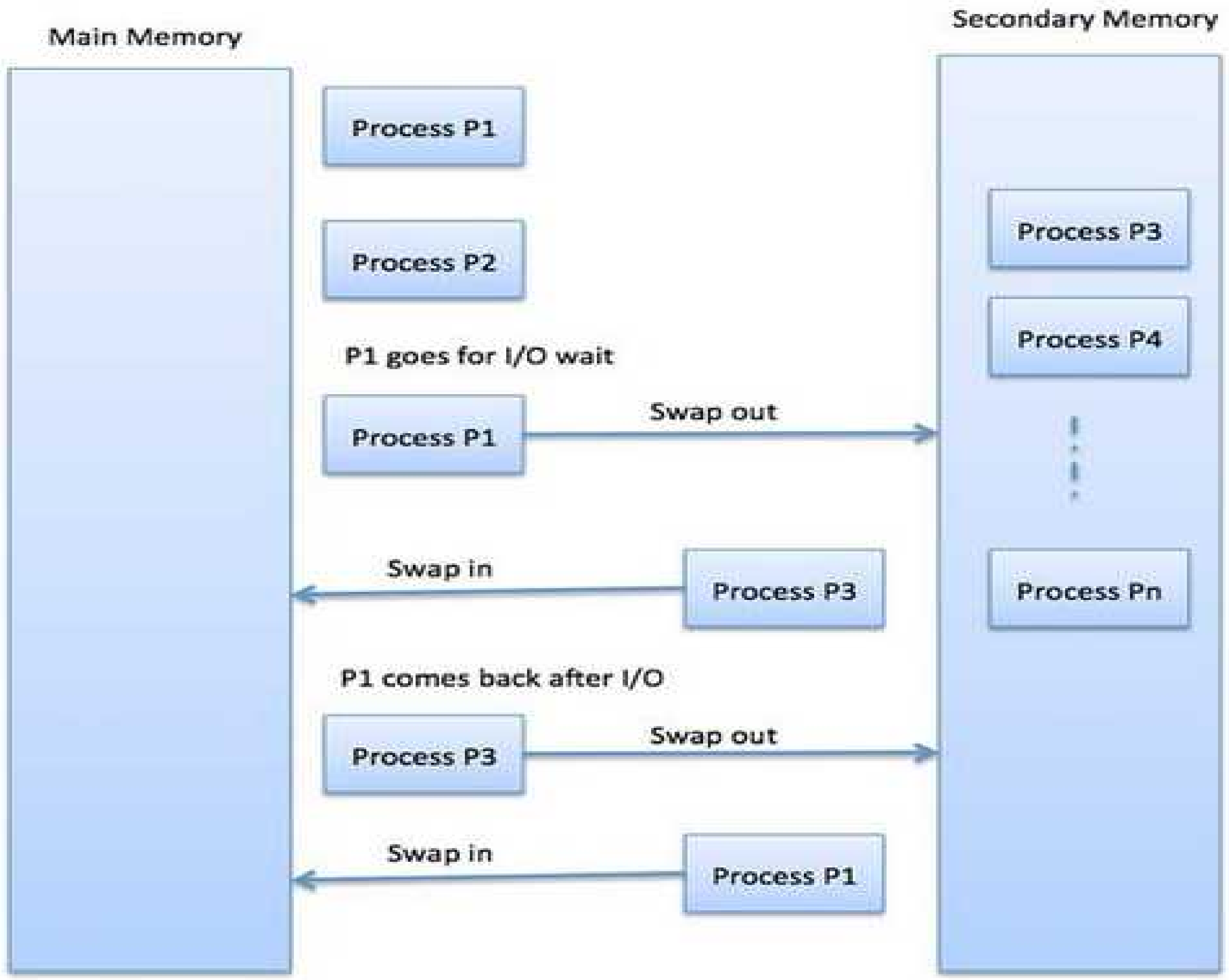
Swapping

- Which process will be taken out?
- In round robin process-scheduling, the processes are executed, according to 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 higher-priority process wishes to execute, a lower-priority process in memory will be swapped out.
- The blocked processes, which are waiting for an I/O, can be swapped out.

Swapping

- Where in the memory will the process be brought back?
- there are two options to swap in a process.
- The first method is to swap-in the process at the same location, if there is compile time or load time binding.
- However, this may not be possible every time and it is inconvenient.
- Therefore, another method is to place the swapped-in process anywhere in the memory where there is space. But this requires the relocation
- Swapping incurs the cost of implementation.

  1. Swap space
- 2. Swap Time


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

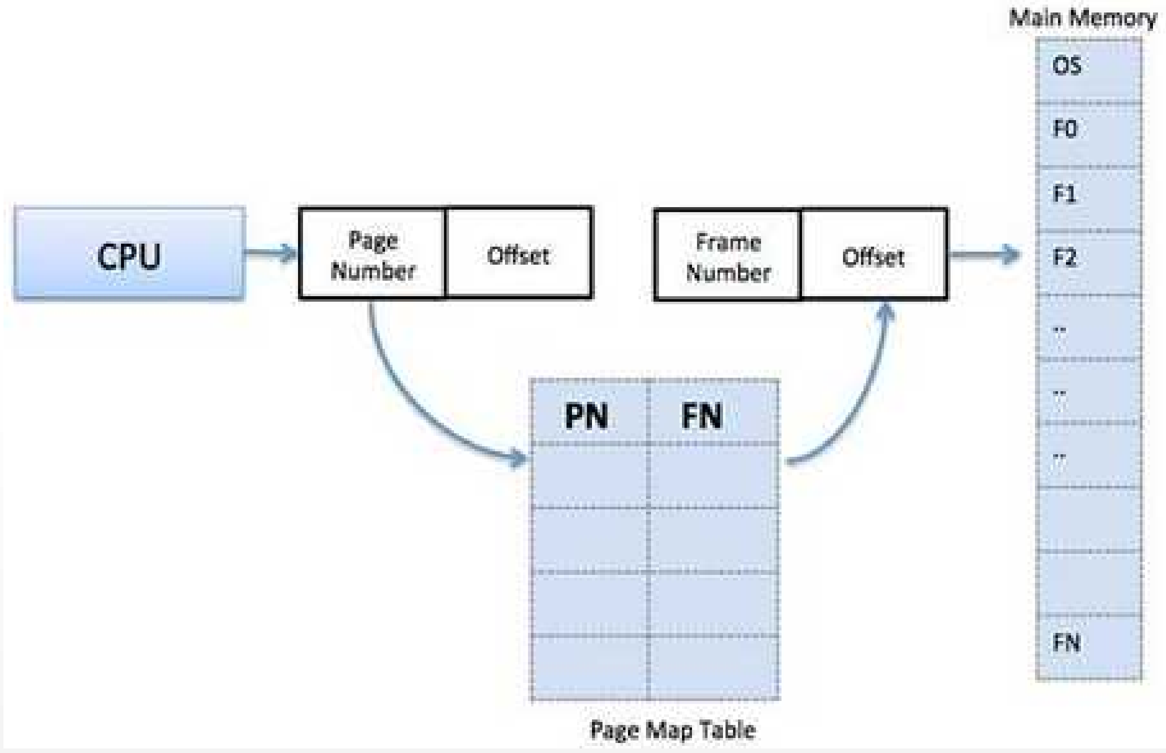
Paging Concept

- The first non-contiguous allocation method is paging, where memory is divided into equal-size partitions. The partitions are relatively smaller, compared to the contiguous method.
- They are known as frames. Since the idea is to reduce external fragmentation, the logical memory of a process is also divided into small chunks or blocks of the same size as frames.
- These chunks are called pages of a process.
- In this way, whenever a frame in memory is allocated to a page, it fits well in the memory, thereby eliminating the problem of external fragmentation.
- Moreover, the hard disk is also divided into blocks, which are of same size as frames.

Paging

- Thus, paging is a logical concept that divides the secondary storage into fixed-size pages, and is implemented in main memory through frames.
- All the pages of the process to be executed are loaded into any available frame in the memory.
- A address in the paging concept has two parts: **a page number and its displacement or offset in the page**. Consequently, this logical address is converted into a physical address.
- **Disk address = Page number + page offset**
- For that, **the start address of the page in the memory, that is, address in the base register**, must be known. In this way, every page will have one start address in the memory.
- unlike contiguous allocation. Instead of a base register for every page, the start addresses of pages are stored in the form of a table, known as a page table.

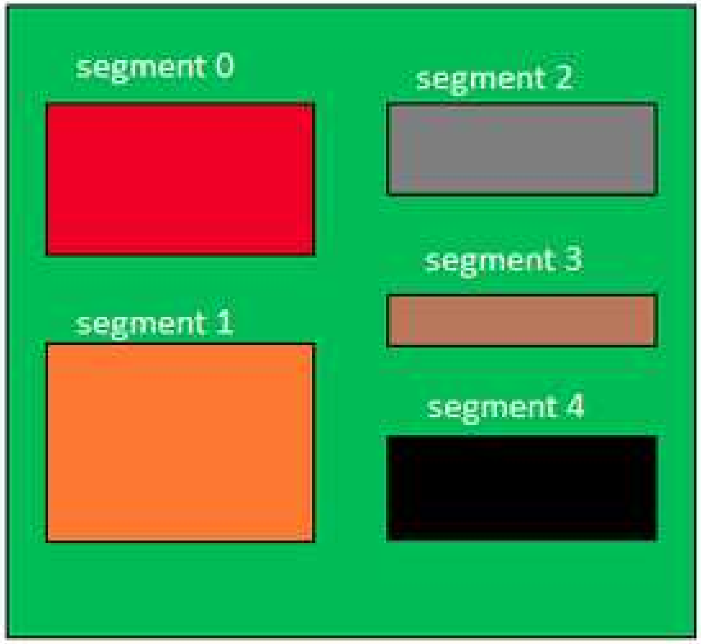
- **A page table is a data structure used to store the base addresses of each page in the process, that is, the entry in the page table indicates the frame location of pages in the memory.**
- Thus, the paging concept involves the logical memory division into pages, page table to keep the base addresses of pages, and physical memory divided into frames



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. A table stores the information about all such segments and is called Segment Table.
- 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.
- Translation of Two dimensional Logical Address to one dimensional Physical Address.
-

Logical View of Segmentation

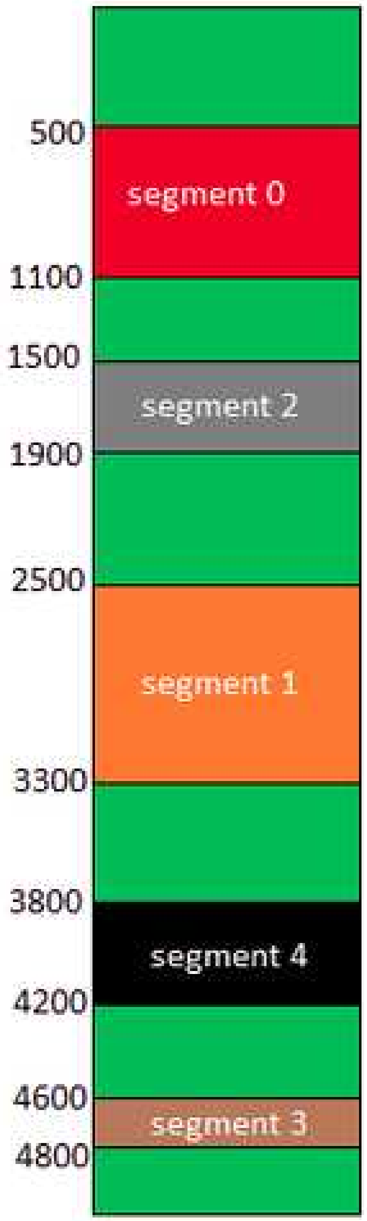


Logical Address Space

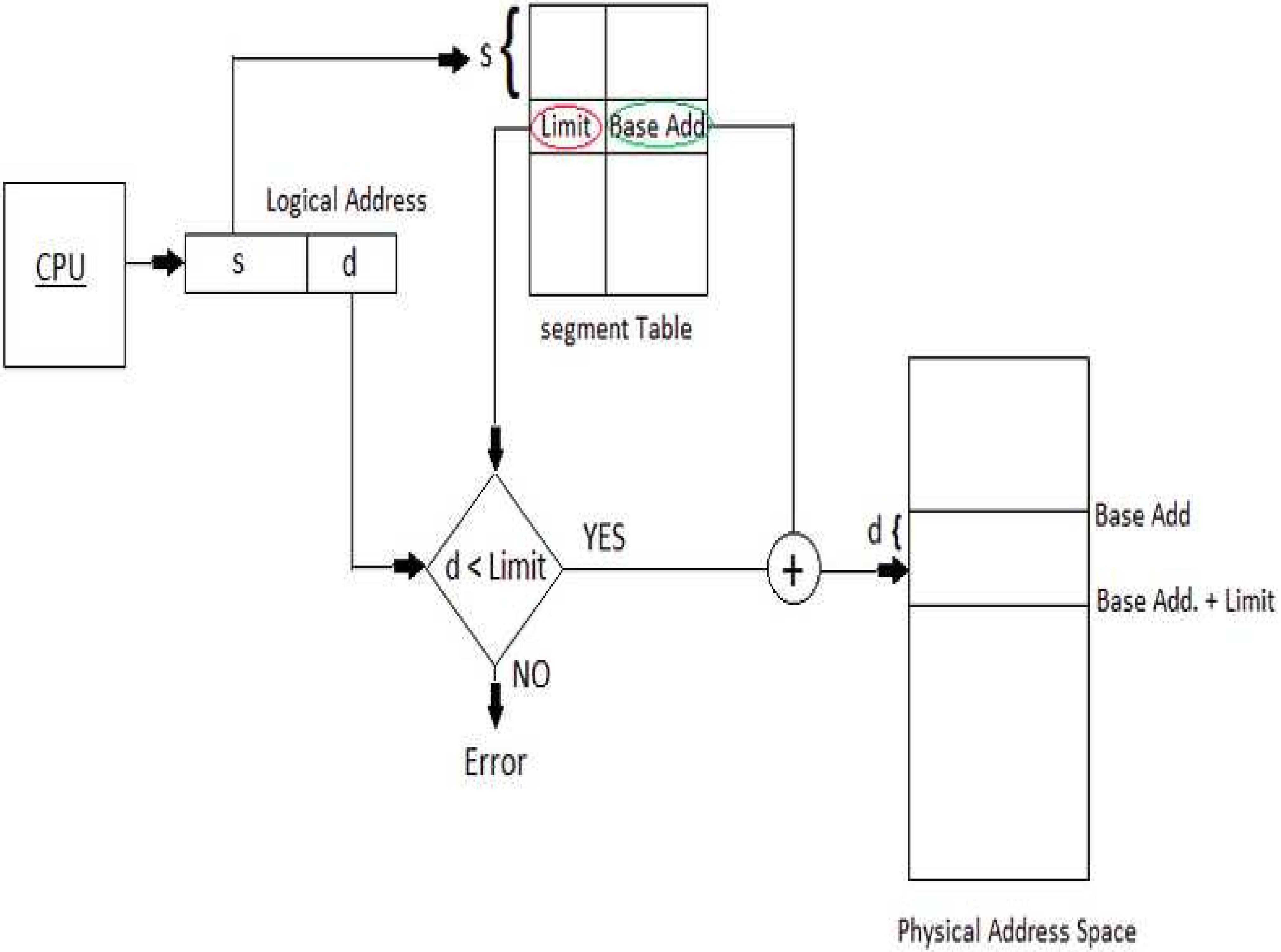
Segment Number

	base address	Limit
0	500	600
1	2500	800
2	1500	400
3	4600	200
4	3800	400

Segment Table



Physical Address Space


CPU
Logical Address
s
d
s
Limit
Base Add
segment Table
d < Limit
YES
NO
Error
+
d
Base Add
Base Add. + Limit
Physical Address Space

Segmentation

- Address generated by the CPU is divided into:
- Segment number (s): Number of bits required to represent the segment.
- Segment offset (d): Number of bits required to represent the size of the segment.
- **Advantages of Segmentation –**

  1. No Internal fragmentation.

  2. Segment Table consumes less space in comparison to Page table in paging.
- **Disadvantage of Segmentation –**

  1. As processes are loaded and removed from the memory, the free memory space is broken into little pieces, causing External fragmentation.

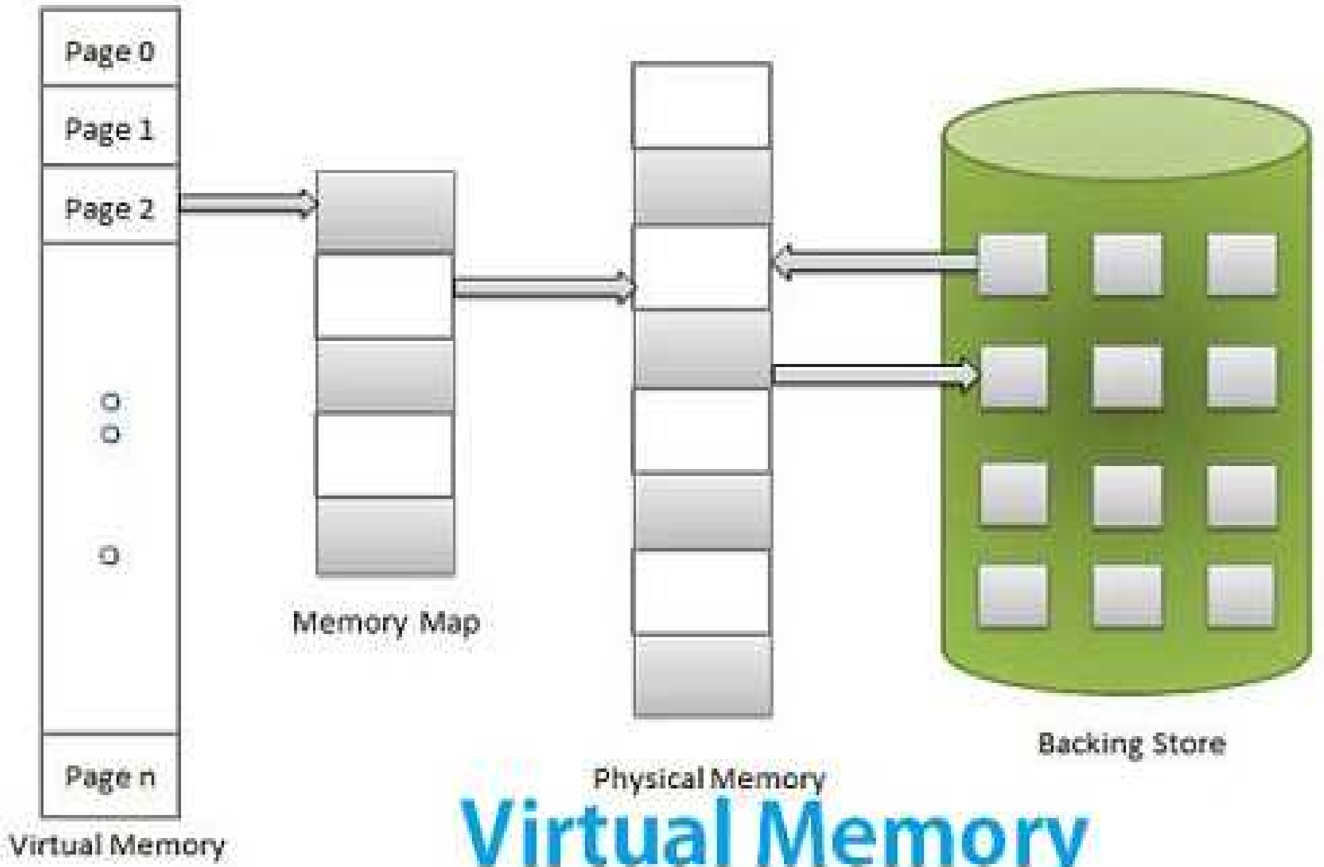
Virtual Memory

Introduction

- Virtual memory is a memory management technique which is used when the process size is too large to fit in the real memory.
- Therefore, a virtual memory is created that does not limit the size of the real memory and a programmer is free to write a large-size program.
- In virtual memory, combined approach of paging and segmentation is used.

Need for Virtual Memory

- The need of virtual memory arises from the fact that physical memory is “limited”.
- System may be running on low physical memory that is not allowing the loader to load all the pages of executable into memory. Even if the processes currently sitting in memory are swapped out, physical memory requirements of the new process are not met.



Virtual Memory

Demand Paging

- A demand paging system is quite similar to a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance.
- When a context switch occurs, the operating system does not copy any of the old program’s pages out to the disk or any of the new program’s pages into the main memory Instead, it just begins executing the new program after loading the first page and fetches that program’s pages as they are referenced.
- While executing a program, if the program references a page which is not available in the main memory because it was swapped out a little ago, the processor treats this invalid memory reference as a page fault and transfers control from the program to the operating system to demand the page back into the memory.

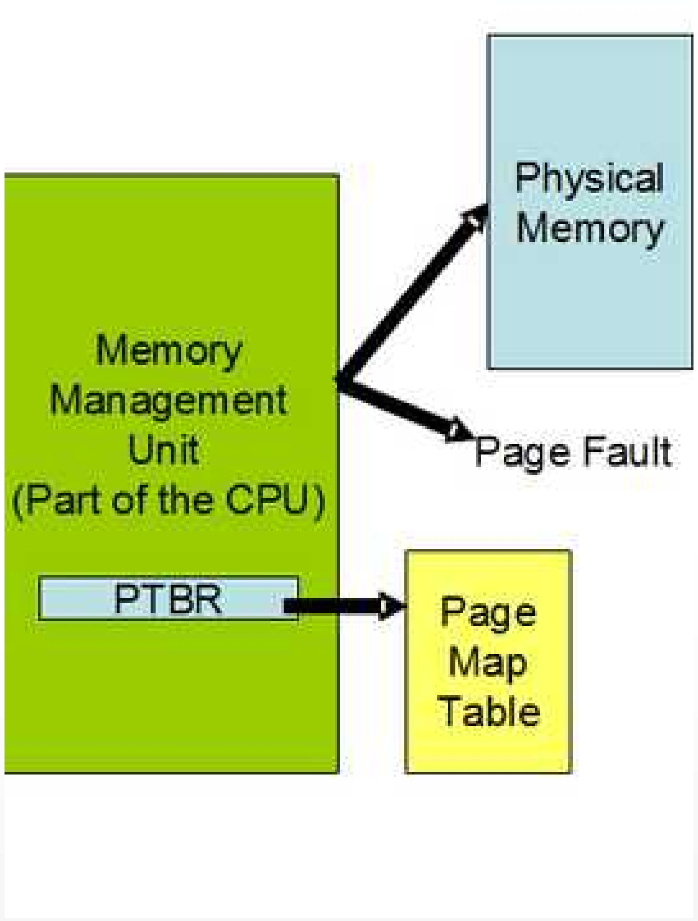
- Advantages
- Following are the advantages of Demand Paging −

  Large virtual memory.

  More efficient use of memory.

  There is no limit on degree of multiprogramming.
- Disadvantages

  Number of tables and the amount of processor overhead for handling page interrupts are greater than in the case of the simple paged management techniques.



Page Replacement Algorithm

- Page replacement algorithms are the techniques using which an Operating System decides which memory pages to swap out, write to disk when a page of memory needs to be allocated.
- There are two algorithm for page replacement

  1. FIFO – First In First Out

  2. LRU – Least Recently Used

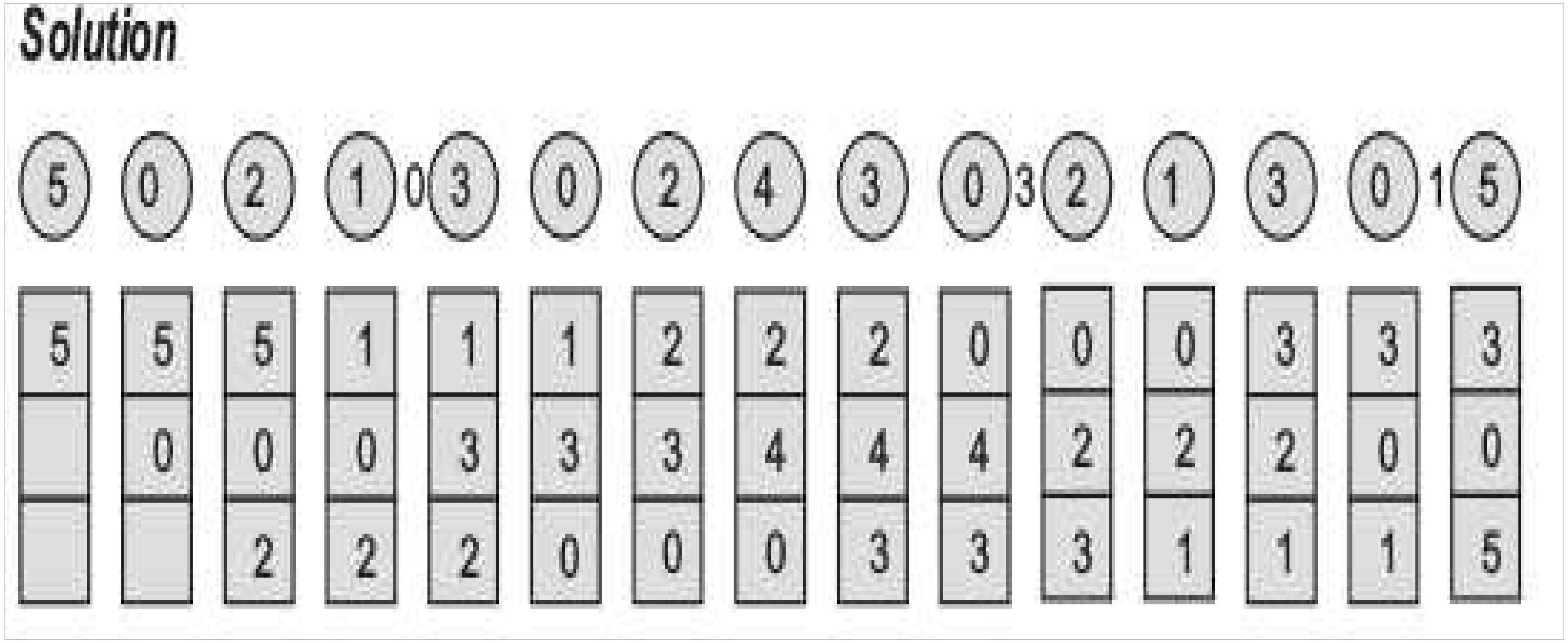
FIFO

- According to FIFO, the oldest page among all the pages in the memory is chosen as the victim.
- The question is how to know the oldest page in the memory.
- One approach may be to attach the time while storing a page in the memory. However, an easier approach is to store all the pages in the memory in a FIFO queue.
- The page at the head of the queue will be paged-out first and a new page will be inserted at the tail of the queue

FIFO

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

FIFO



Solution

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

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

FIFO

- Initially, all the three frames are empty. Page number 5 is first referenced, and it is a page fault.
- After handling the page fault, the page is brought into the memory in one of the frames. Similarly, Page numbers 0 and 2 occupy the other two frames. Next, Page number 1 is referenced but there is no free frame.
- Here, the FIFO algorithm comes into the picture and replaces the page that was brought first, that is, Page number 5. The next referenced Page number 0 is already in the memory; therefore, it will not fault.
- After this, the next referenced page number is 3, which is a page fault. Therefore, Page number 0 is replaced. This process goes on resulting in 15 page faults. All the page references causing page faults have been circled to show the page faults.

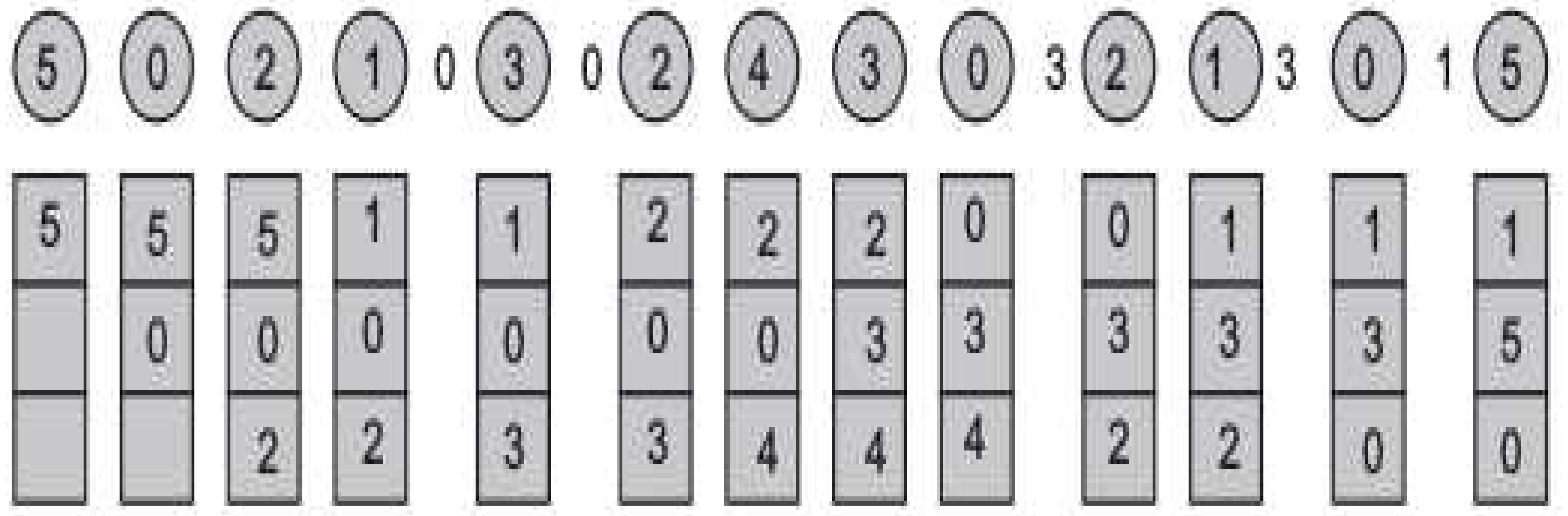
LRU – Least Recently Used

- The idea is to predict future references based on the past data, that is, a page that has not been referenced for a long time in the past may not be referenced for a long time in the future either.
- In this way, LRU page-replacement algorithm replaces a page that has not been used for the longest period of time in the past.
- Calculate the number of page faults for the following reference string using LRU
- page-replacement algorithm with frame size as 3.
- 5 0 2 1 0 3 0 2 4 3 0 3 2 1 3 0 1 5

LRU

- The page references 5, 0, and 2 will result in page faults. The next page reference 1 needs
- page replacement. Out of 5, 0, and 2, Page number 5 has not been used in the past. Therefore,
- it will be replaced. This process goes on resulting in total 13 page faults,

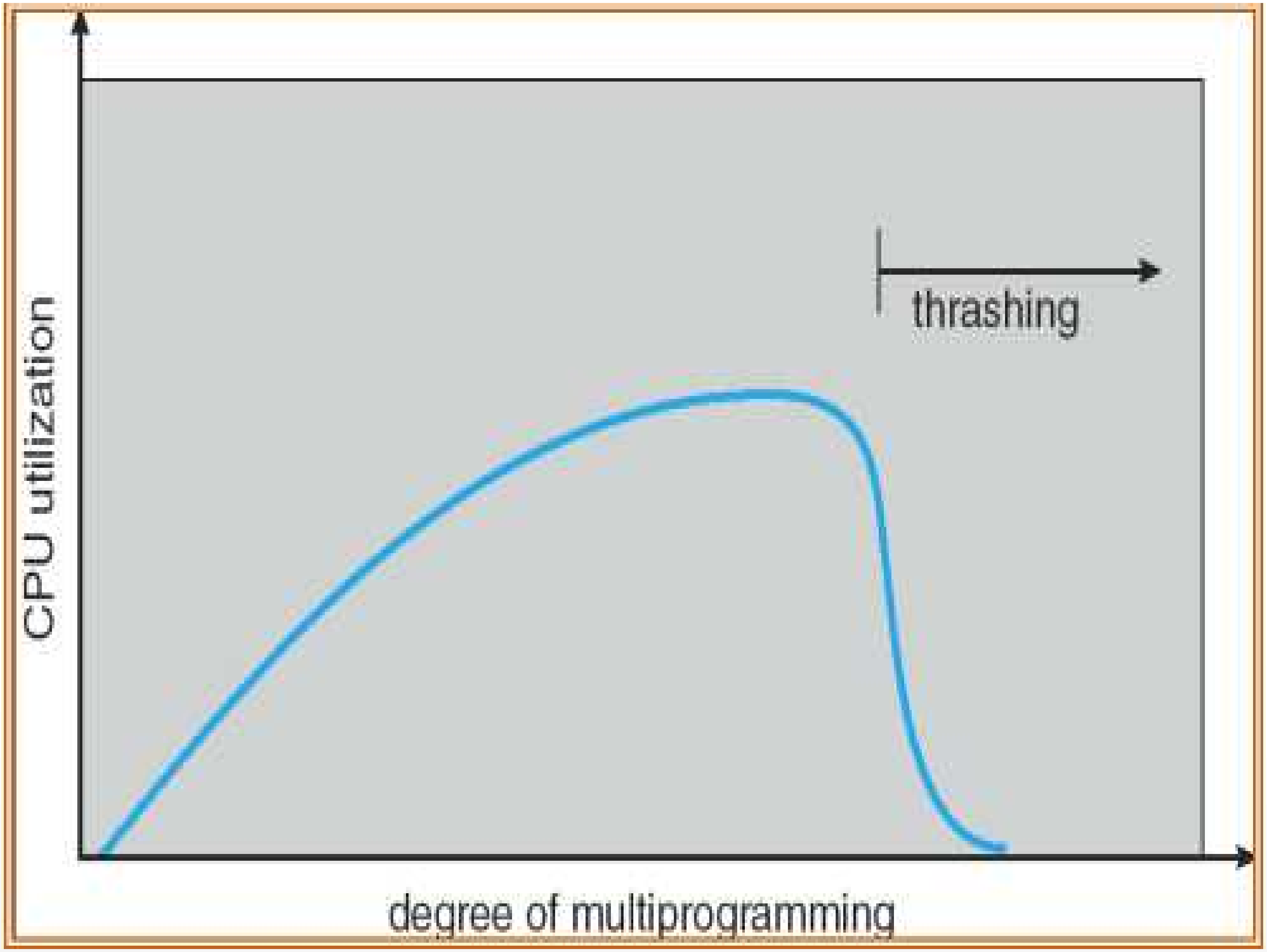
Solution



Thrashing

- If it happens that your system has to swap pages at such a higher rate that major chunk of CPU time is spent in swapping then this state is known as thrashing.
- So effectively during thrashing, the CPU spends less time in some actual productive work and more time in swapping.
- Memory thrashing is a problem which arises when the memory is located more than the physical memory and it is not available.
- Now the effects of thrashing and also the extent to which thrashing occurs will be decided by the type of page replacement policy.

Thrashing


thrashing
CPU utilization
degree of multiprogramming