



Chapter -6


Memory management



Introduction



- Main Memory refers to a physical memory that is the internal memory to the computer.
- The word main is used to distinguish it from external mass storage devices such as disk drives.
- Main memory is also known as RAM.
- The computer is able to change only data that is in main memory.
- Therefore, every program we execute and every file we access must be copied from a storage device into main memory.




Requirements of Memory Management System

- Memory management keeps track of the status of each memory location, whether it is allocated or free. It allocates the memory dynamically to the programs at their request and frees it for reuse when it is no longer needed.

- These Requirements of memory management are:

1. **Relocation :**


- Programmer does not know where the program will be placed in memory when it is executed.
- While the program is executing, it may be swapped to disk and returned to main memory at a different location (relocated) .
- Memory references must be translated in the code to actual physical memory address



Requirements of Memory Management System

2. Protection


- Processes should not be able to reference memory locations in another process without permission.
- Memory protection requirement must be satisfied by the processor (hardware) rather than the operating system (software).
- Operating system can hardly control a process when it occupies the processor. all of the memory references a program will make



Requirements of Memory Management System

3. Sharing


- ▶ Allow several processes to access the same portion of memory.
- ▶ Better to allow each process access to the same copy of the program rather than have their own separate copy
- ▶ For example , multiple processes may use the same system file and it is natural to load one copy of file into main memory and let it shared by those processes.



Requirements of Memory Management System

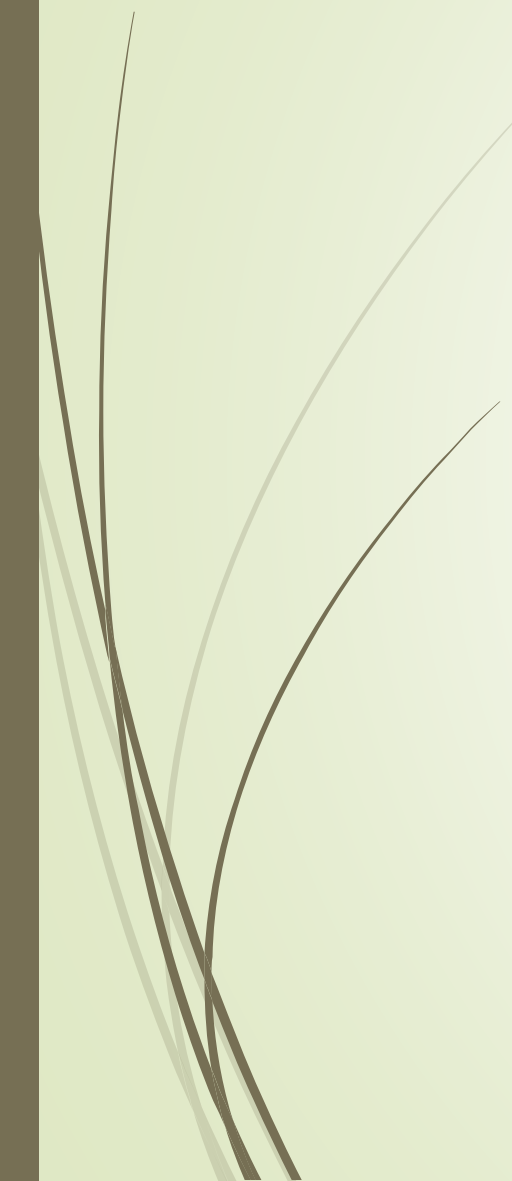
4. Logical Organization

- To effectively deal with user program the os and computer hardware must support basic module to provide the required protection and sharing.
- Programs are written in modules.
- Modules can be written and compiled independently.
- Different degrees of protection given to modules (read-only, execute-only)
- Share modules among processes



Requirements of Memory Management System

5. Physical Organization

- Memory available for a program plus its data may be insufficient
 - Overlaying(programming method that allows programs to be larger than the computer's main memory) allows various modules to be assigned the same region of memory.
 - Programmer does not know how much space will be available.
- 



Memory Address

Logical Address :

- It 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.
- The hardware device called Memory-Management Unit (MMU) is used for mapping logical address to its corresponding physical address.



Memory Address

➤ **Physical Address**

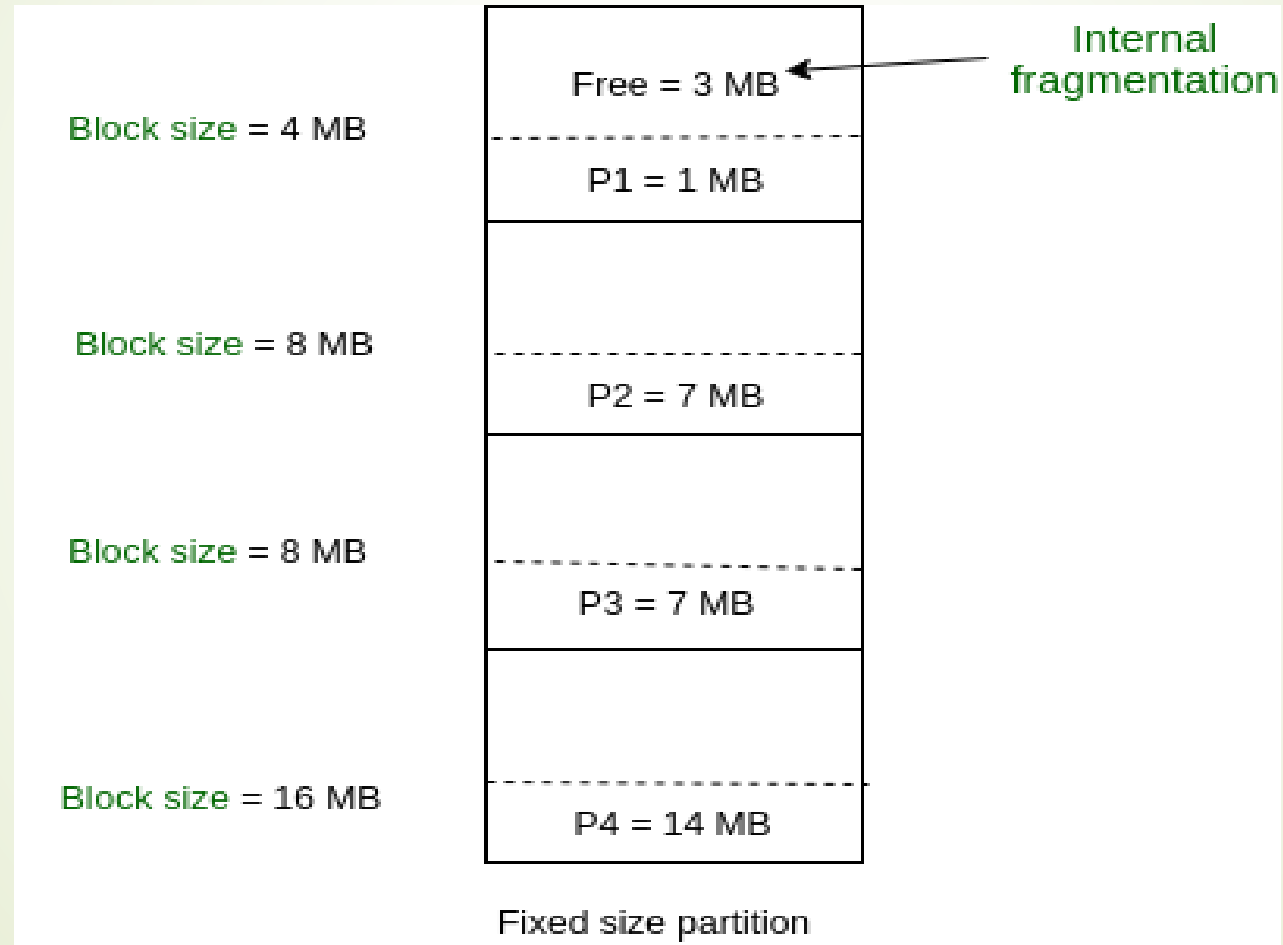
- It identifies a physical location of required data in a memory.
- The user never directly deals with the physical address but can access by its corresponding logical address.
- The user program generates the logical address and thinks that the program is running in this logical address but the program needs physical memory for its execution, therefore, the logical address must be mapped to the physical address by MMU before they are used.



Memory partitions : Fixed Partitions

- Here memory is divided into fixed sized partitions.
- Size can be equal or unequal for different partitions.
- Generally unequal partitions are used for better utilizations.
- Each partition can accommodate exactly one process, means only single process can be placed in one partition.
- The partition boundaries are not movable.
- Whenever any program needs to be loaded in memory, a free partition big enough to hold the program is found. This partition will be allocated to that program or process.
- If there is no free partition available of required size, then the process needs to wait. Such process will be put in a queue.

Memory Partitioning : Fixed partitioning





Advantages of Fixed Partitioning

- **Easy to implement:**

Algorithms needed to implement Fixed Partitioning are easy to implement. It simply requires putting a process into certain partition without focussing on the emergence of Internal and External Fragmentation.

- **Little OS overhead:**

Processing of Fixed Partitioning require lesser excess and indirect computational power.



Disadvantages of Fixed Partitioning

- **Internal Fragmentation:**

Main memory use is inefficient. Any program, no matter how small, occupies an entire partition. This can cause internal fragmentation.

- **External Fragmentation:**

The total unused space (as stated above) of various partitions cannot be used to load the processes even though there is space available but not in the contiguous form (as spanning is not allowed).

- **Limit process size:**

Process of size greater than size of partition in Main Memory cannot be accommodated. Partition size cannot be varied according to the size of incoming process's size. Hence, process size of 32MB in above stated example is invalid.



Disadvantages of Fixed Partitioning

- **Limitation on Degree of Multiprogramming:**

Partition in Main Memory are made before execution or during system configure. Main Memory is divided into fixed number of partition.



Memory partitions : Variable Partitioning

Variable Partitioning

- It is a part of Contiguous allocation technique. It is used to alleviate the problem faced by Fixed Partitioning.
- In contrast with fixed partitioning, partitions are not made before the execution or during system configure. Various **features** associated with variable Partitioning
 - Initially RAM is empty and partitions are made during the run-time according to process's need instead of partitioning during system configure.
 - The size of partition will be equal to incoming process.
 - The partition size varies according to the need of the process so that the internal fragmentation can be avoided to ensure efficient utilisation of RAM.
 - Number of partitions in RAM is not fixed and depends on the number of incoming process and Main Memory's size.

Memory partitions : Variable Partitioning

Dynamic partitioning

Operating system	
P1 = 2 MB	Block size = 2 MB
P2 = 7 MB	Block size = 7 MB
P3 = 1 MB	Block size = 1 MB
P4 = 5 MB	Block size = 5 MB
Empty space of RAM	

Partition size = process size
So, no internal Fragmentation



Advantages of Variable Partitioning

- **No Internal Fragmentation:**

In variable Partitioning, space in main memory is allocated strictly according to the need of process, hence there is no case of internal fragmentation. There will be no unused space left in the partition.

- **No restriction on Degree of Multiprogramming:**

More number of processes can be accommodated due to absence of internal fragmentation. A process can be loaded until the memory is empty.

- **No Limitation on the size of the process:**

In Fixed partitioning, the process with the size greater than the size of the largest partition could not be loaded and process can not be divided as it is invalid in contiguous allocation technique. Here, In variable partitioning, the process size can't be restricted since the partition size is decided according to the process size.

Disadvantages of Variable Partitioning

- **Difficult Implementation:**

Implementing variable Partitioning is difficult as compared to Fixed Partitioning as it involves allocation of memory during run-time rather than during system configure.

- **External Fragmentation:**

There will be external fragmentation inspite of absence of internal fragmentation.

- For example, suppose in above example- process P1 (2MB) and process P3 (1MB) completed their execution. Hence two spaces are left i.e. 2MB and 1MB. Let's suppose process P5 of size 3MB comes. The empty space in memory cannot be allocated as no spanning is allowed in contiguous allocation. The rule says that process must be contiguously present in main memory to get executed. Hence it results in External Fragmentation.

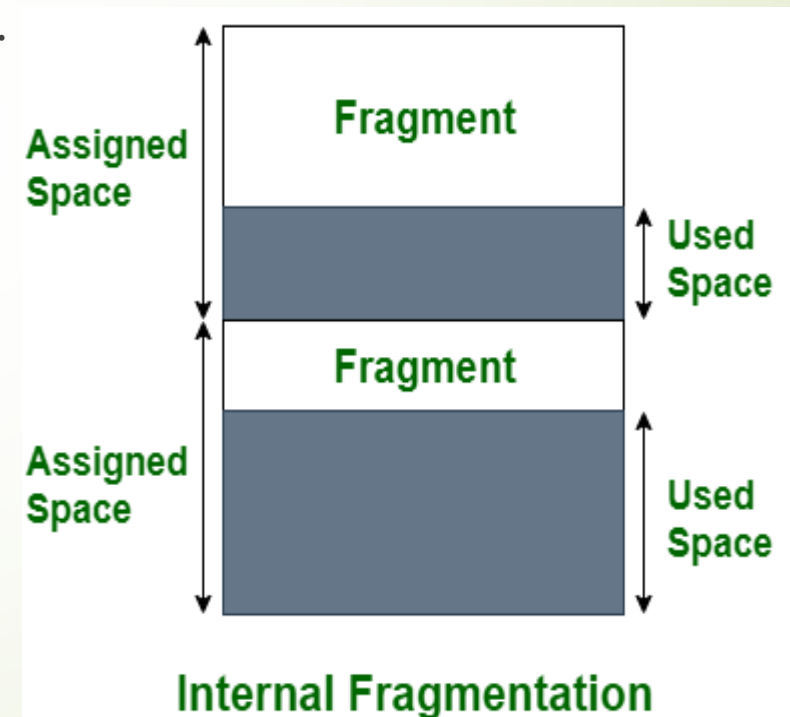


Fragmentation in OS

- There are two types of fragmentation in OS which are given as: Internal fragmentation, and External fragmentation.
- **Internal Fragmentation:**
Internal fragmentation happens when the memory is split into mounted sized blocks.
- Whenever a method request for the memory, the mounted sized block is allotted to the method.
- just in case the memory allotted to the method is somewhat larger than the memory requested, then the distinction between allotted and requested memory is that the Internal fragmentation.

Fragmentation in OS

The above diagram clearly shows the internal fragmentation because the difference between memory allocated and required space or memory is more so it is called Internal fragmentation.





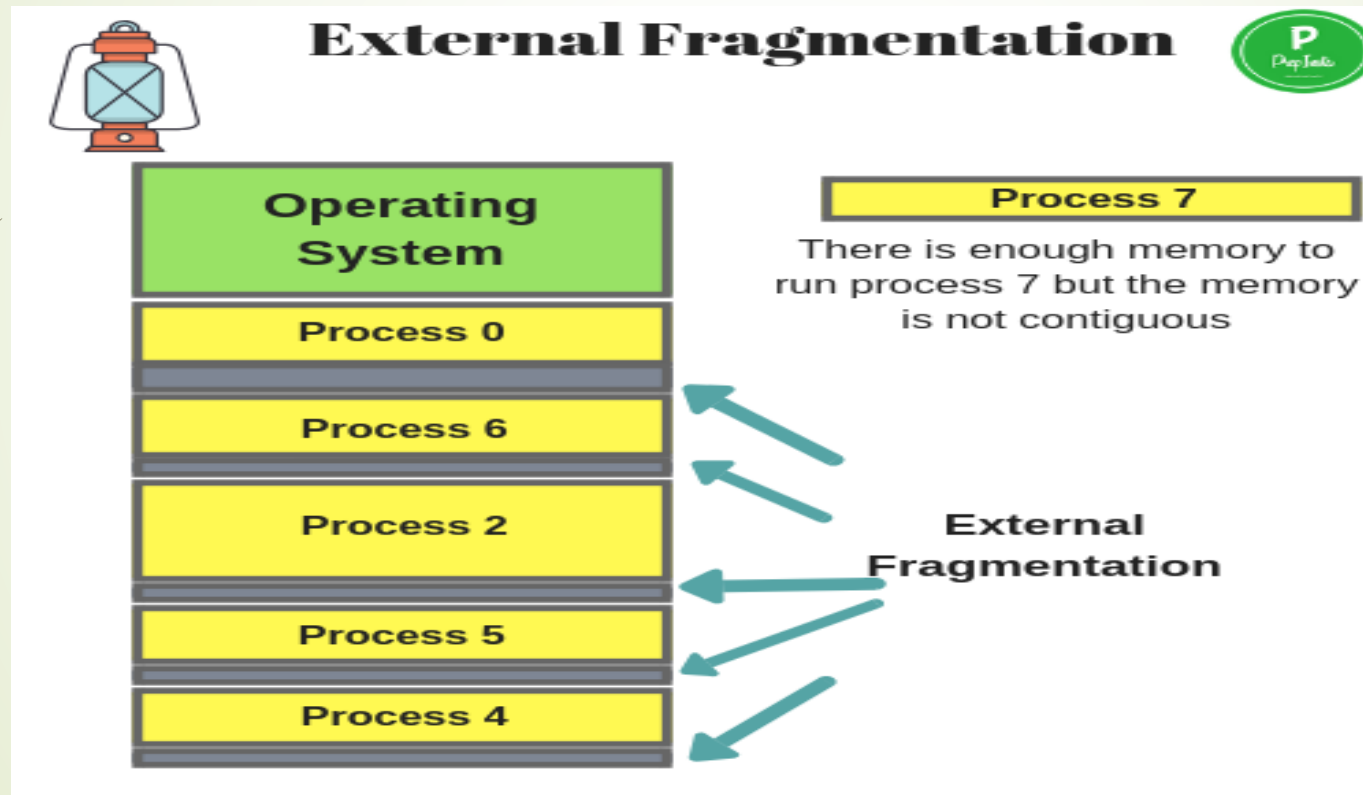
Fragmentation in OS

- **External Fragmentation:**

External fragmentation happens when there's a sufficient quantity of area within the memory to satisfy the memory request of a method.

- however the process's memory request cannot be fulfilled because the memory offered is during a non-contiguous manner.
- Either you apply first-fit or best-fit memory allocation strategy it'll cause external fragmentation.

Fragmentation in OS





Contiguous & Non-contiguous memory allocation

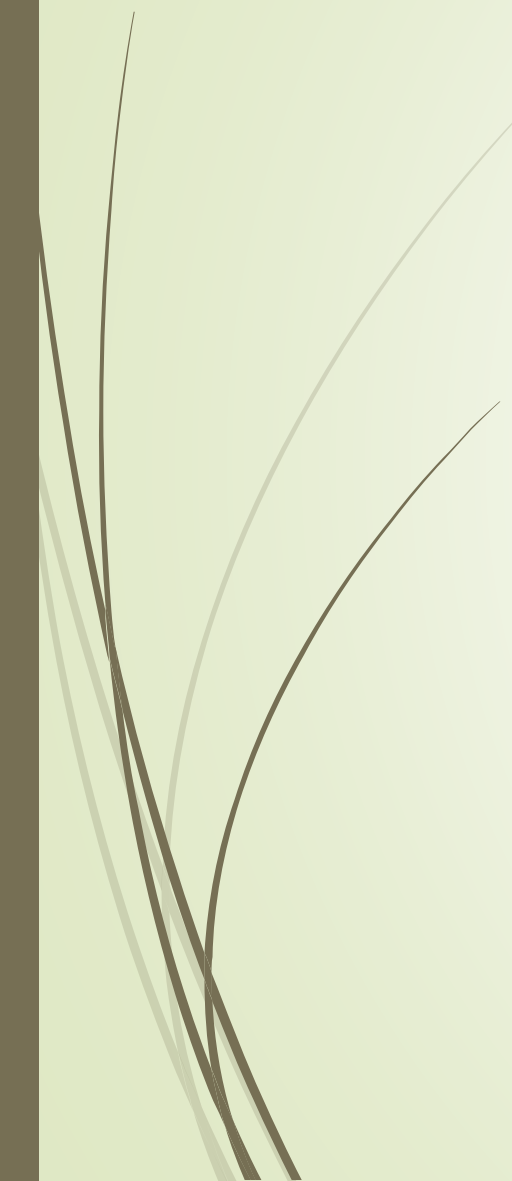
Contiguous memory allocation

- assigns the consecutive blocks of memory to a process requesting for memory.
- Contiguous memory allocation does not have the overhead of address translation while execution of a process.
- A process executes faster in contiguous memory allocation.



Contiguous & Non-contiguous memory allocation

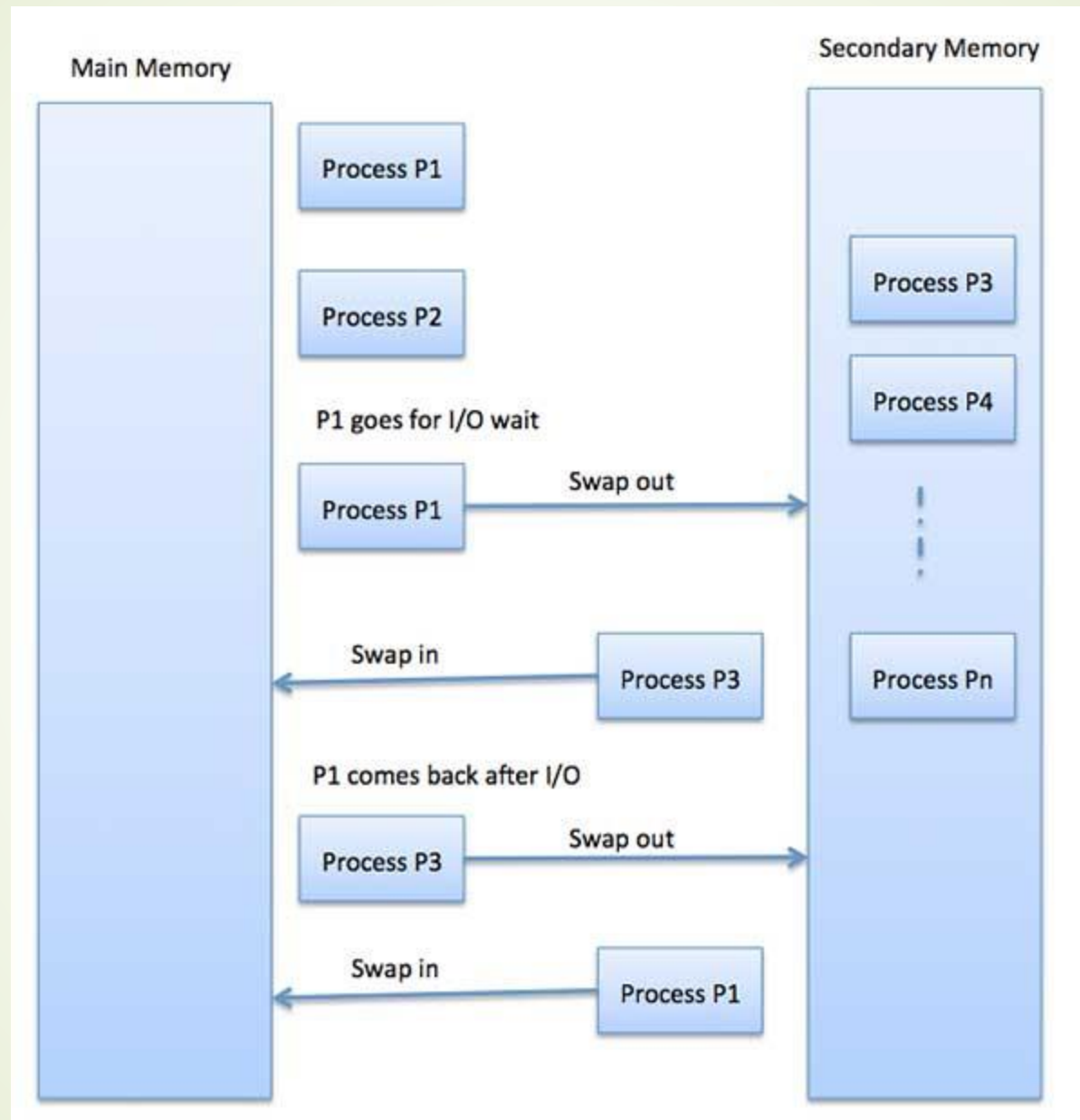
Non-contiguous memory allocation assigns the separate memory blocks at a different location in memory space in a non consecutive manner to a process requesting for memory.

- Non-contiguous memory allocation has overhead of address translation while execution of a process.
 - A process executes quite slower comparatively in non -contiguous memory allocation.
- 



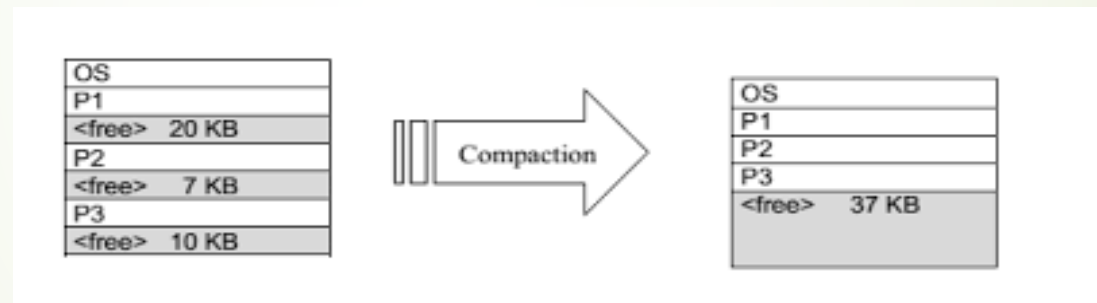
Swapping

- 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.
- Though performance is usually affected by swapping process but it helps in running multiple and big processes in parallel and that's the reason **Swapping is also known as a technique for memory compaction.**
- The total time taken by swapping process includes the time it takes to move the entire process to a secondary disk and then to copy the process back to memory, as well as the time the process takes to regain main memory.



Memory compaction

- Compaction solves the problem of external fragmentation.
- When swapping creates multiple holes in memory, it is possible to combine them all in one big hole by moving all the processes downward as far as possible.



- All free blocks are brought together as one large block of free space.
- This technique is known as memory compaction.
- It requires a lot of CPU time and wastes CPU time.



Base and Limit register

- An address space is set of addresses that a process can use to address memory.
- An address space is a range of valid addresses in memory that are available for a program or process.
- Two registers: Base and Limit
 1. Base register: Start address of a program in physical memory.
 2. Limit register: Length of the program.
- For every memory access
 - Base is added to the address
 - Result compared to Limit
- Only OS can modify Base and Limit register.



Memory Allocation Algorithms

➤ Four memory allocation algorithms are as follow

1. First fit
2. Next fit
3. Best fit
4. Worst fit



First Fit Algorithm



- Search starts from the starting location of the memory.
- First available hole that is large enough to hold the process is selected for allocation.
- The hole is then broken up into two pieces, one for process and another for unused memory.
- For example : a process request 12KB of memory and memory manager currently has a list of un allocated blocks of 6KB,14 KB,19KB,11 KB,13KB blocks. Then first fit will allocate the14KB block to the process.
- Fastest algorithm because it searches as little as possible.
- Memory loss is higher, as very large hole may be selected for small process.



Next Fit Algorithm

- It works in the same way as first fit, except that it keeps the track of where it is whenever it finds a suitable hole.
- The next time when it is called to find a hole, it starts searching the list from the place where it left off last time.
- Search time is smaller.
- Memory manager must have to keep track of last allotted hole to process.
- It gives slightly worse performance than first fit.



Best Fit Algorithm

- Entire memory is searched here.
- The smallest hole, which is large enough to hold the process, is selected for allocation.
- For example : a process request 12KB of memory and memory manager currently has a list of un allocated blocks of 6KB,14 KB,19KB,11 KB,13KB blocks. Then Next fit will allocate the13KB block to the process.
- Search time is high, as it searches entire memory every time.
- Memory loss is less.



Worst Fit Algorithm

- ▶ Entire memory is searched here also.
- ▶ The largest hole, which is largest enough to hold the process, is selected for allocation.
- ▶ For example : a process request 12KB of memory and memory manager currently has a list of un allocated blocks of 6KB,14 KB,19KB,11 KB,13KB blocks. Then Next fit will allocate the19KB block to the process.
- ▶ Search time is high, as it searches entire memory every time.
- ▶ This algorithm can be used only with dynamic partitioning.



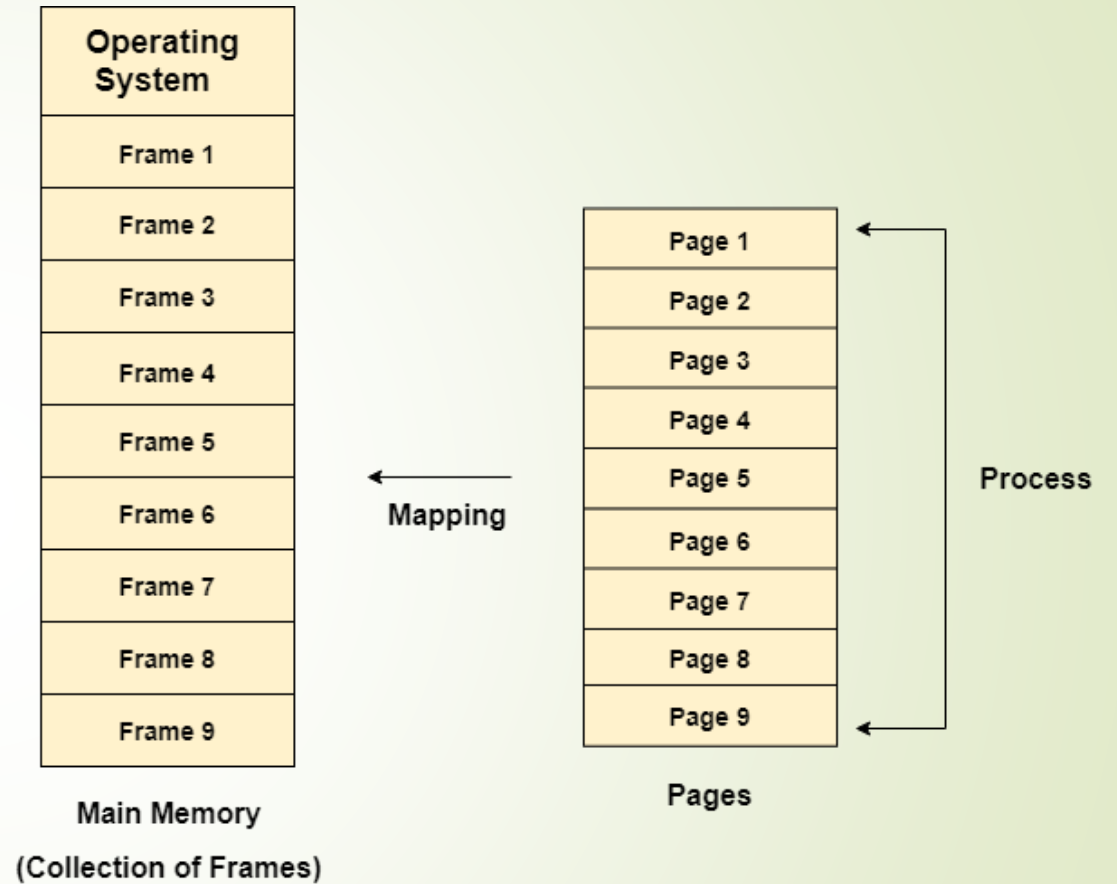
Paging

- Paging overcomes the problem of external fragmentation in os by allowing non-contiguous memory allocation.
- There are two popular techniques used for non-contiguous memory allocation : **Paging & Segmentation.**
- Paging is a storage mechanism used to retrieve processes from the secondary storage into the main memory in the form of pages.
- 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.

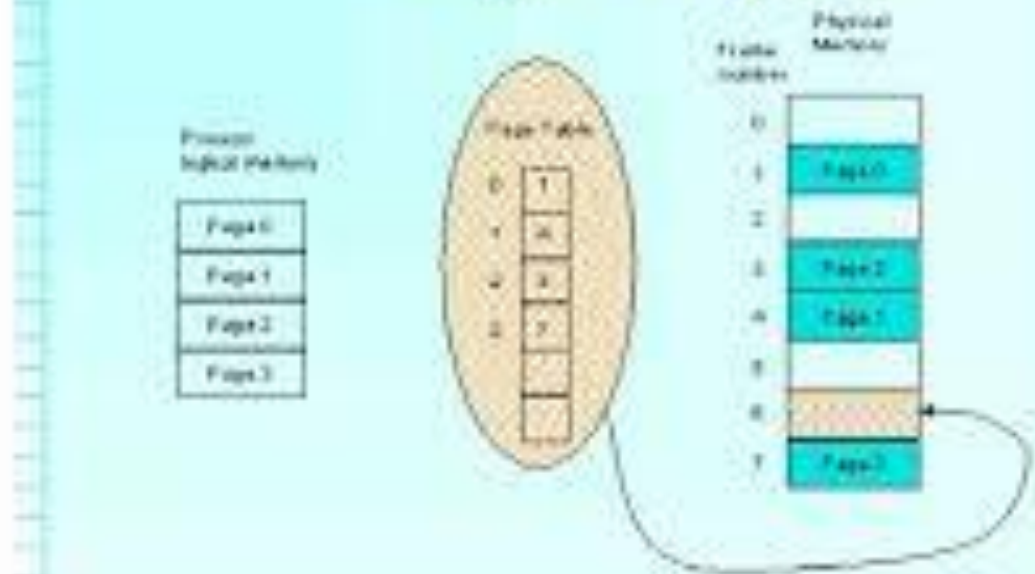
Paging

Example :

- 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.



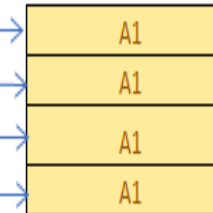
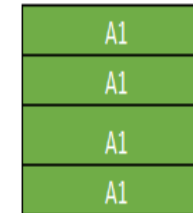
Example of Paging



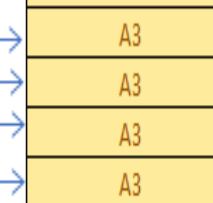
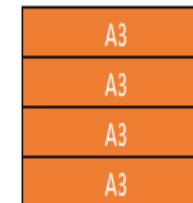
Process A1

16 KB

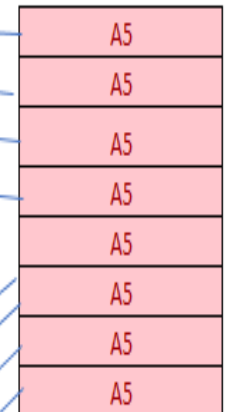
1 Frame = 1 KB
Frame Size = Page Size



Process A3



Process A5



Main Memory
(Collection of Frames)
Paging

Guru99.com

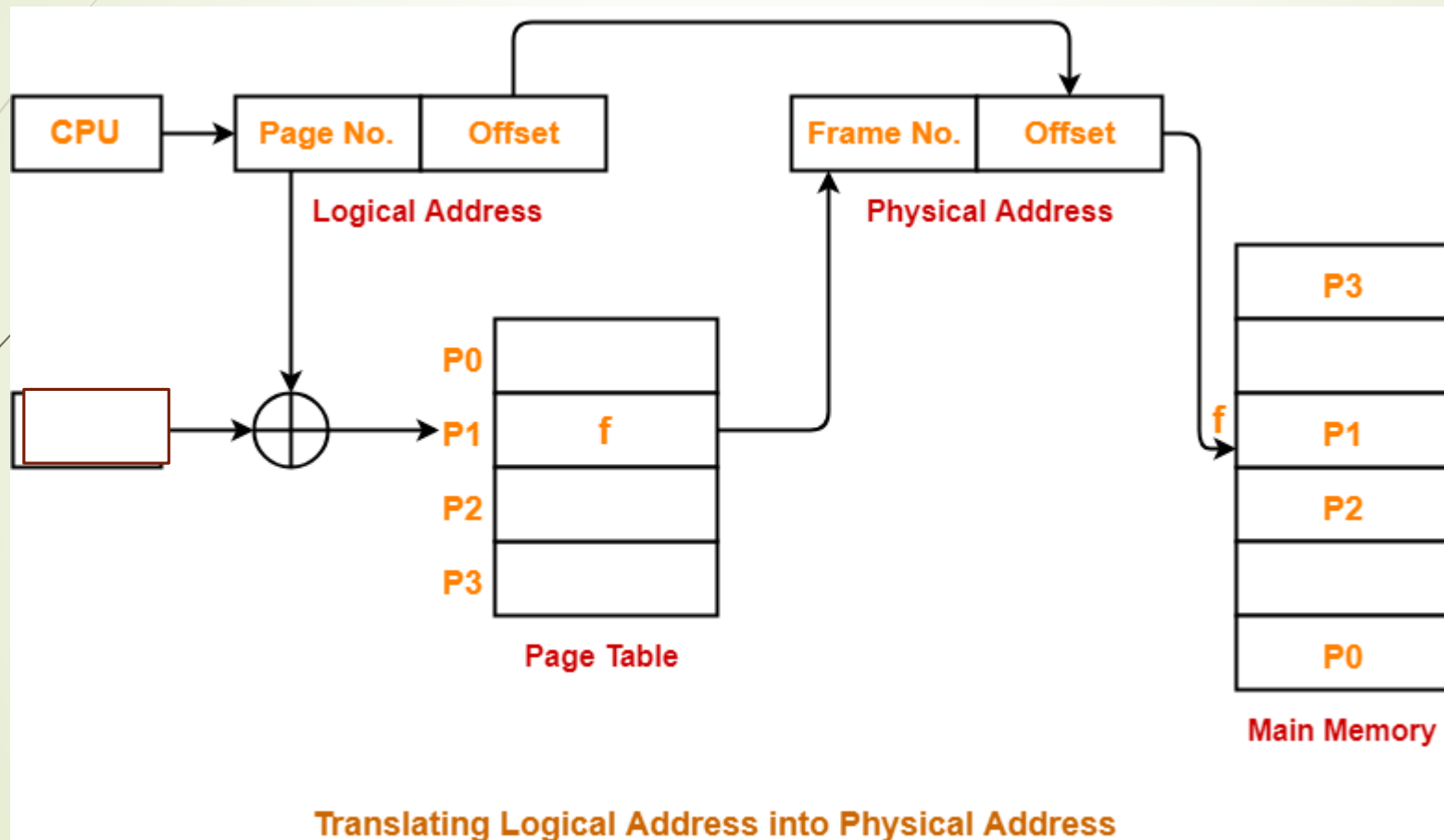
Translating Logical Address into Physical Address

- CPU always generates a logical address.
- A physical address is needed to access the main memory.
- CPU generates a logical address consisting of two parts : **Page Number & Page Offset**
- Page Number specifies the specific page of the process from which CPU wants to read the data.
- Page Offset specifies the specific word on the page that CPU wants to read.
- For the page number generated by the CPU, Page Table provides the corresponding frame number (base address of the frame) where that page is stored in the main memory.
- The frame number combined with the page offset forms the required physical address.



Continue....

- Frame number specifies the specific frame where the required page is stored.
- Page Offset specifies the specific word that has to be read from that page.



Conversion of Virtual Address to Physical Address

- The virtual address is split into a virtual page number (high order bits) and an offset (low-order bits).
- With a 16-bit address and a 4KB page size, the upper 4 bits could specify one of the 11 virtual pages and the lower 12 bits would then specify the byte offset (0 to 4095) within the selected page.
- The virtual page number is used as an index into the Page table.
- If the present/absent bit is 0, it is page-fault; a trap to the operating system is caused to bring required page into main memory.
- If the present/absent bit is 1, required page is there with main memory and page frame number found in the page table is copied to the higher order bit of the output register along with the offset.
- Together Page frame number and offset creates physical address.
- $\text{Physical Address} = \text{Page frame Number} + \text{offset of virtual address}.$



Page Table

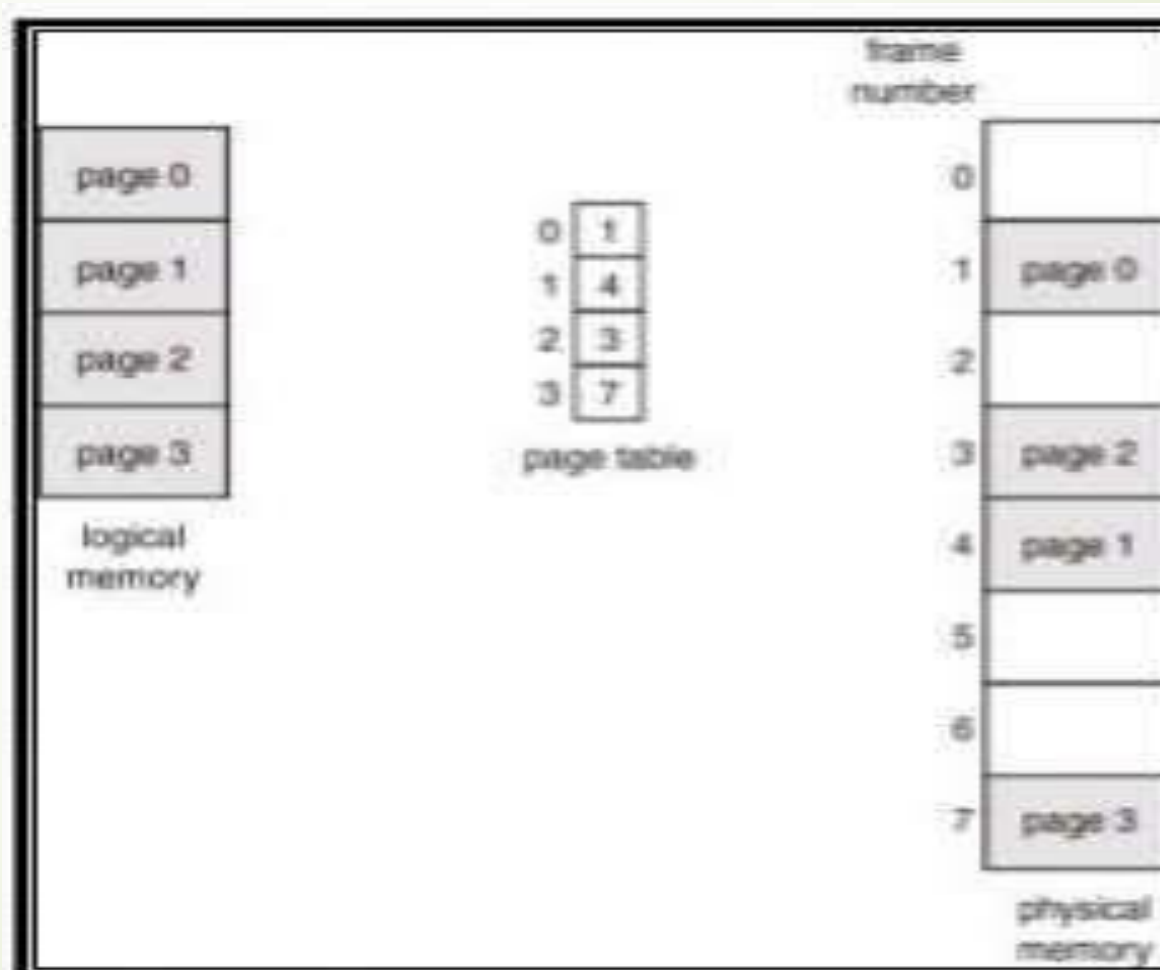
- Page table is a data structure which translates virtual address into equivalent physical address.
- The virtual page number is used as an index into the Page table to find the entry for that virtual page and from the Page table physical page frame number is found.
- Thus the purpose of page table is to map virtual pages onto page frames.
- Page table consist of following fields :
- Page frame Number: It gives the frame number in which the current page you are looking for is present.
- Present/Absent bit: Present or absent bit says whether a particular page you are looking for is present or absent. If it is not present, that is called Page Fault. It is set to 0 if the corresponding page is not in memory. Sometimes this bit is also known as valid/invalid bits.



Continue...

- The Protection bits: Protection bit says that what kind of protection you want on that page. In the simplest form, 0 for read/write and 1 for read only.
- Modified bit: Modified bit says whether the page has been modified or not. If the page in memory has been modified, it must be written back to disk. This bit is also called as dirty bit as it reflects the page's state.
- 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.
- Cashing Disabled bit: This feature is important for pages that maps onto device registers rather than memory. With this bit caching can be turned off.

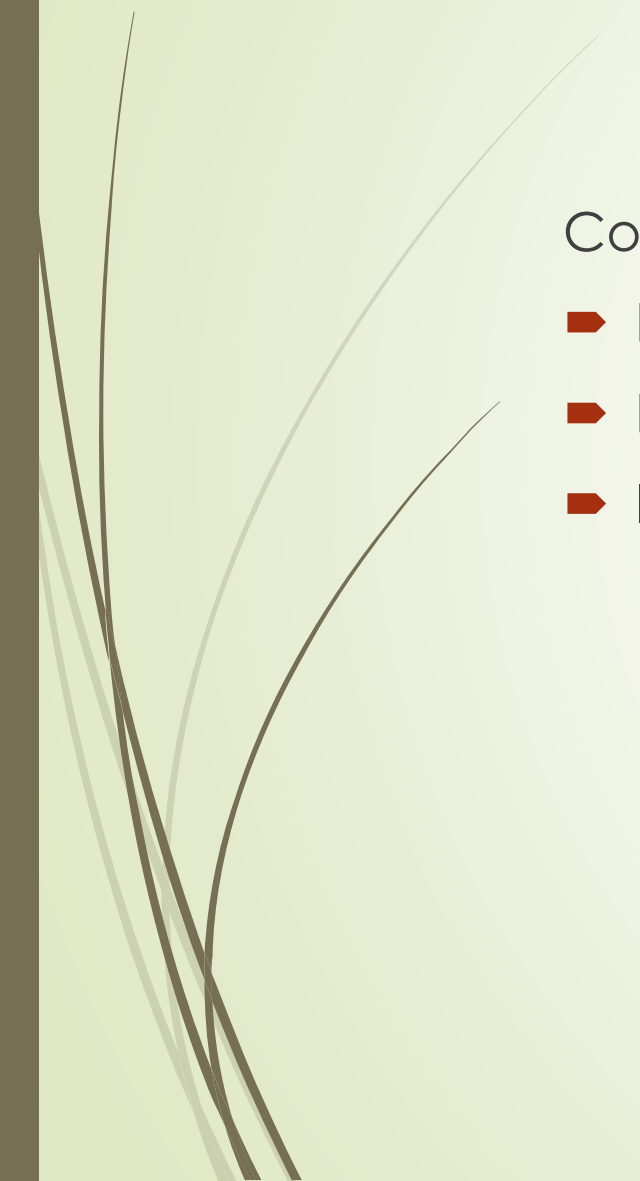
Continue..





Page Table Structure

Common Techniques used for structuring the page table are :

- Hierarchical paging
 - Hashed page tables
 - Inverted page tables
- 



HIERARCHICAL PAGING

Multilevel or Hierarchical Page Table :

- It is also known as multilevel paging.
- The page table might be too big to fit in a contiguous space , so we may have a hierarchy with several levels.
- So , we break up the logical address space into multiple page tables.
- For this a simple techniques we can use are:
- Two level page table
- Three level page table

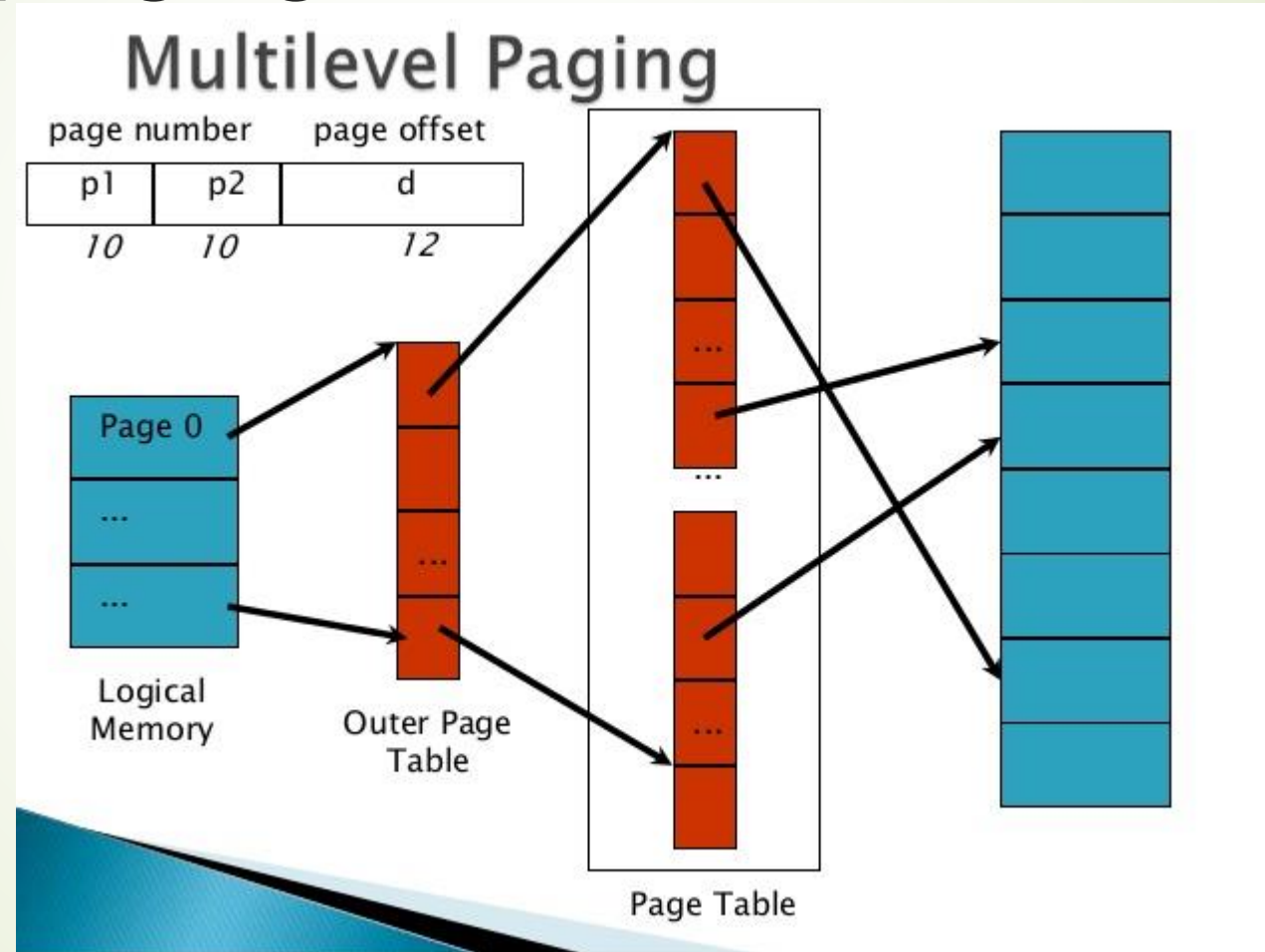
Continue..

TWO LEVEL PAGING:

- A logical address (on 32-bit machine with 4k page size) is divided into:
- A page number consisting of 20 bits.
- A page offset consisting of 12 bits.
- Since the page table is paged , the page number is further divided into:
- A 10-bit page number.
- A 10-bit page offset.
- Thus a logical address is as follows: page number | page offset



Address translation scheme for the two level paging



Continue..

THREE LEVEL PAGING:

- A logical address(on 64-bit machine with 4k page size) is divided into:

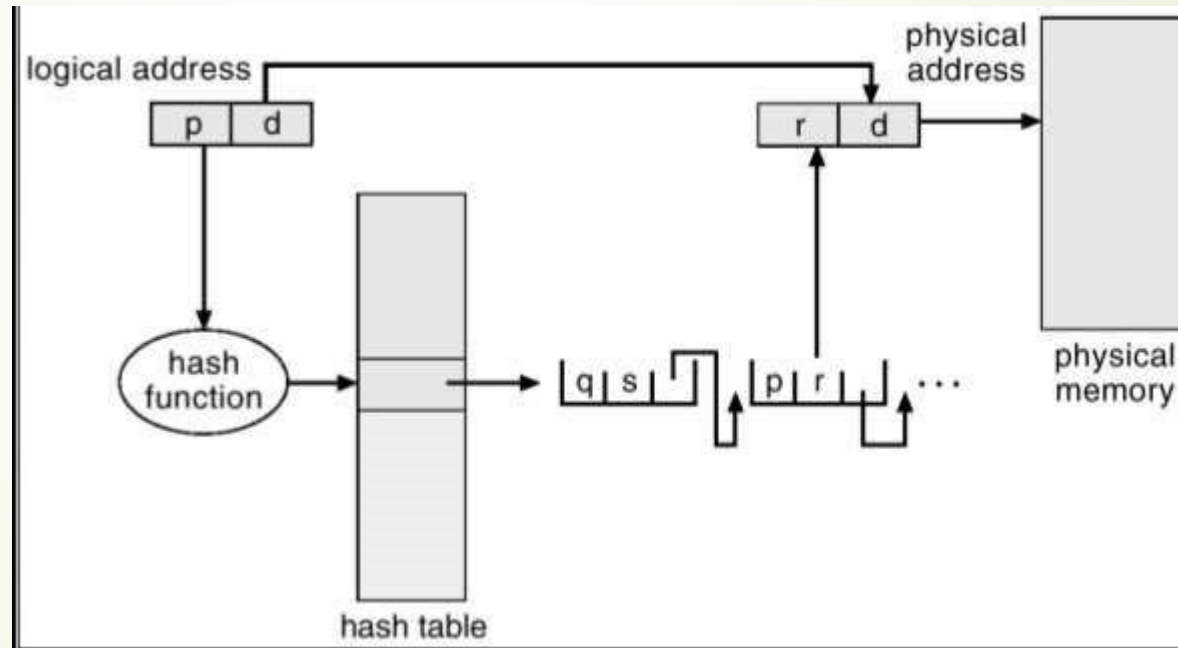
- outer page inner page offset



- 2nd outer page outer page inner page offset



Address translation scheme for the hashed page table

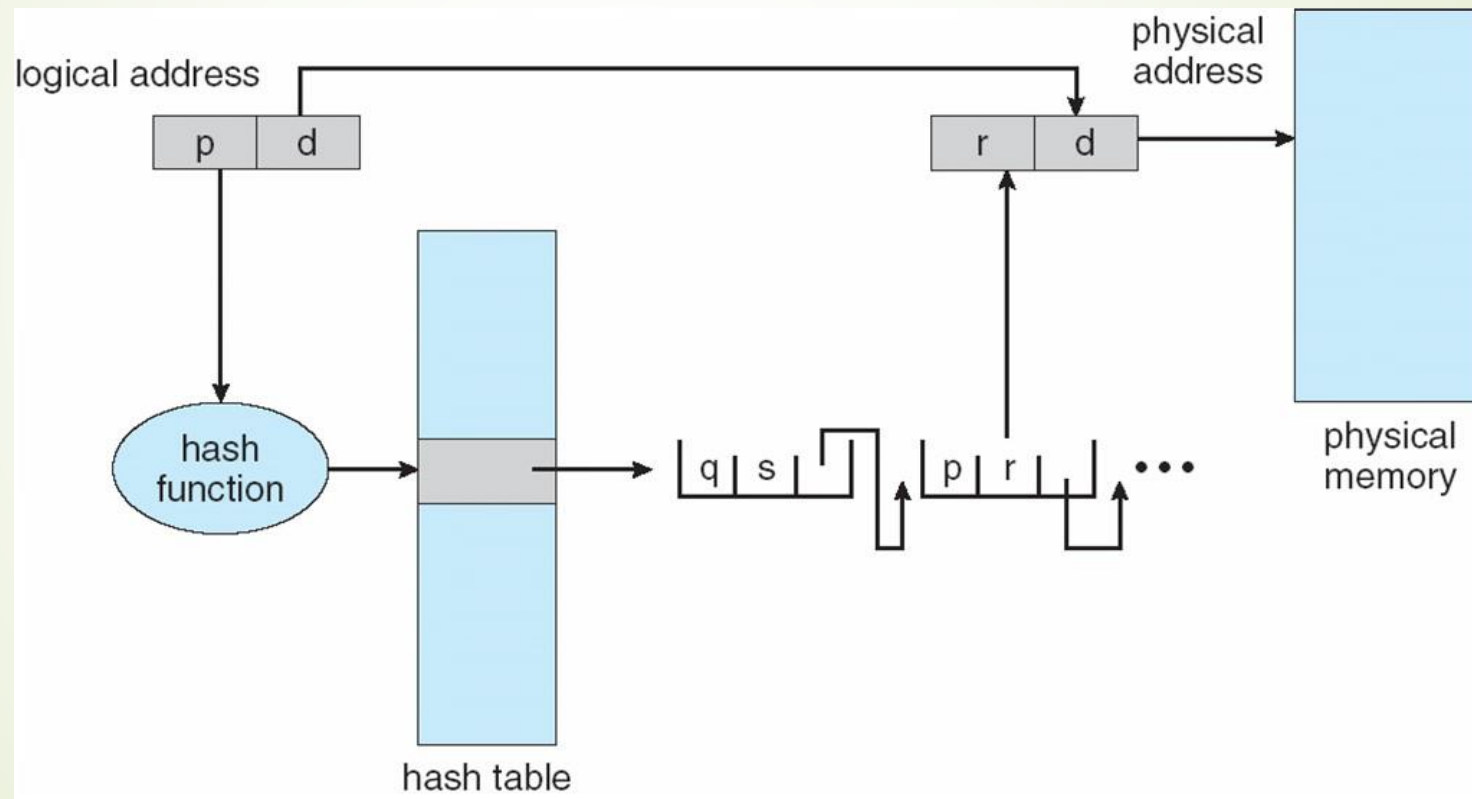




HASHED PAGE TABLE

- Common in address spaces > 32 bits
- The virtual page number is hashed into a page table
 - This page table contains a chain of elements hashing to the same location
- Each element contains
 - (1) the virtual page number
 - (2) the value of the mapped page frame
 - (3) a pointer to the next element
- Virtual page numbers are compared in this chain searching for a match
 - If a match is found, the corresponding physical frame is extracted
- Variation for 64-bit addresses is **clustered page tables**
 - Similar to hashed but each entry refers to several pages (such as 16) rather than 1
 - Especially useful for **sparse** address spaces (where memory references are non-contiguous and scattered)

HASHED PAGE TABLE

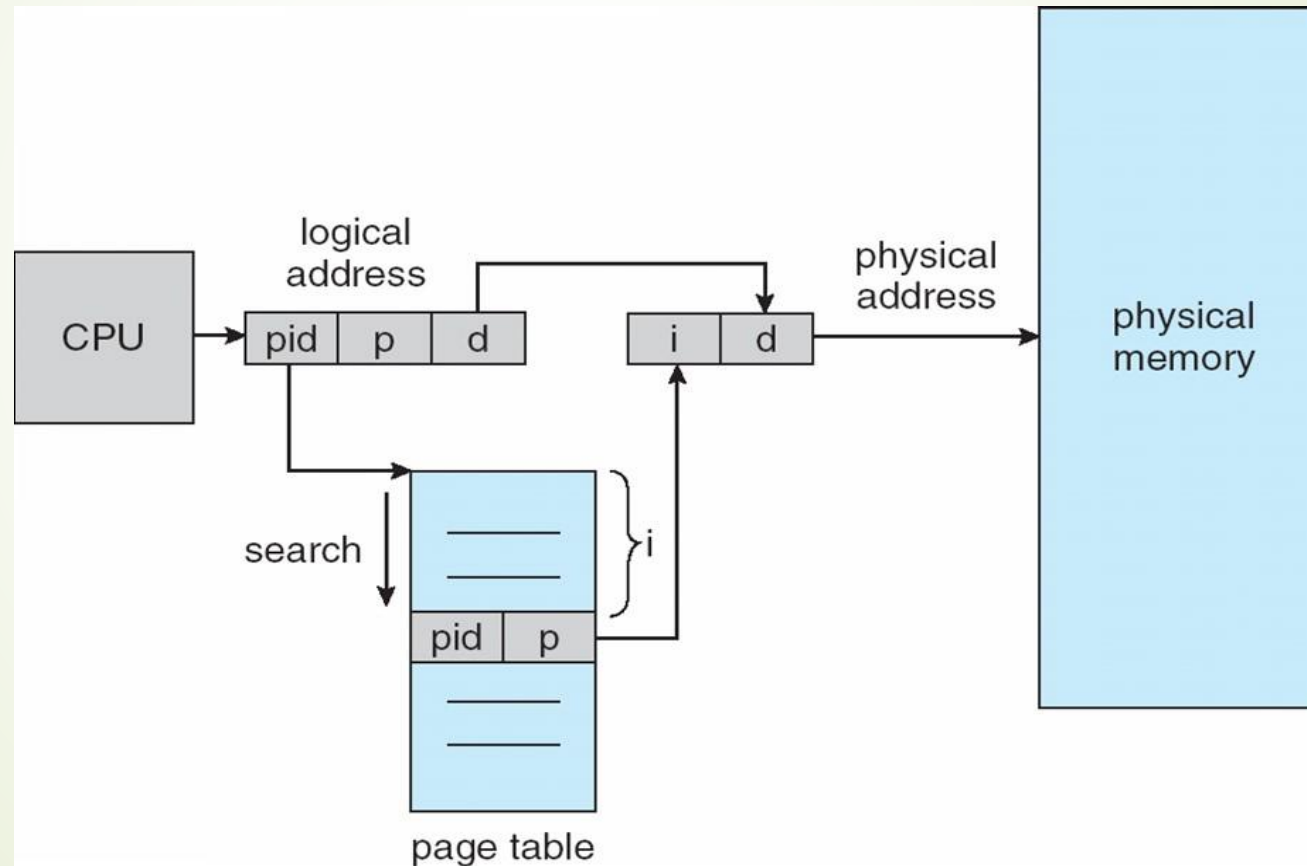





Inverted Page Table

- Rather than each process having a page table and keeping track of all possible logical pages,
 - track all physical pages
- One entry for each real page of memory
- Entry consists of
 - the virtual address of the page stored in that real memory location,
 - information about the process that owns that page
- Decreases memory needed to store each page table
 - but increases time needed to search the table when a page reference occurs
- Use hash table to limit the search to one/few page-table entries – TLB can accelerate access

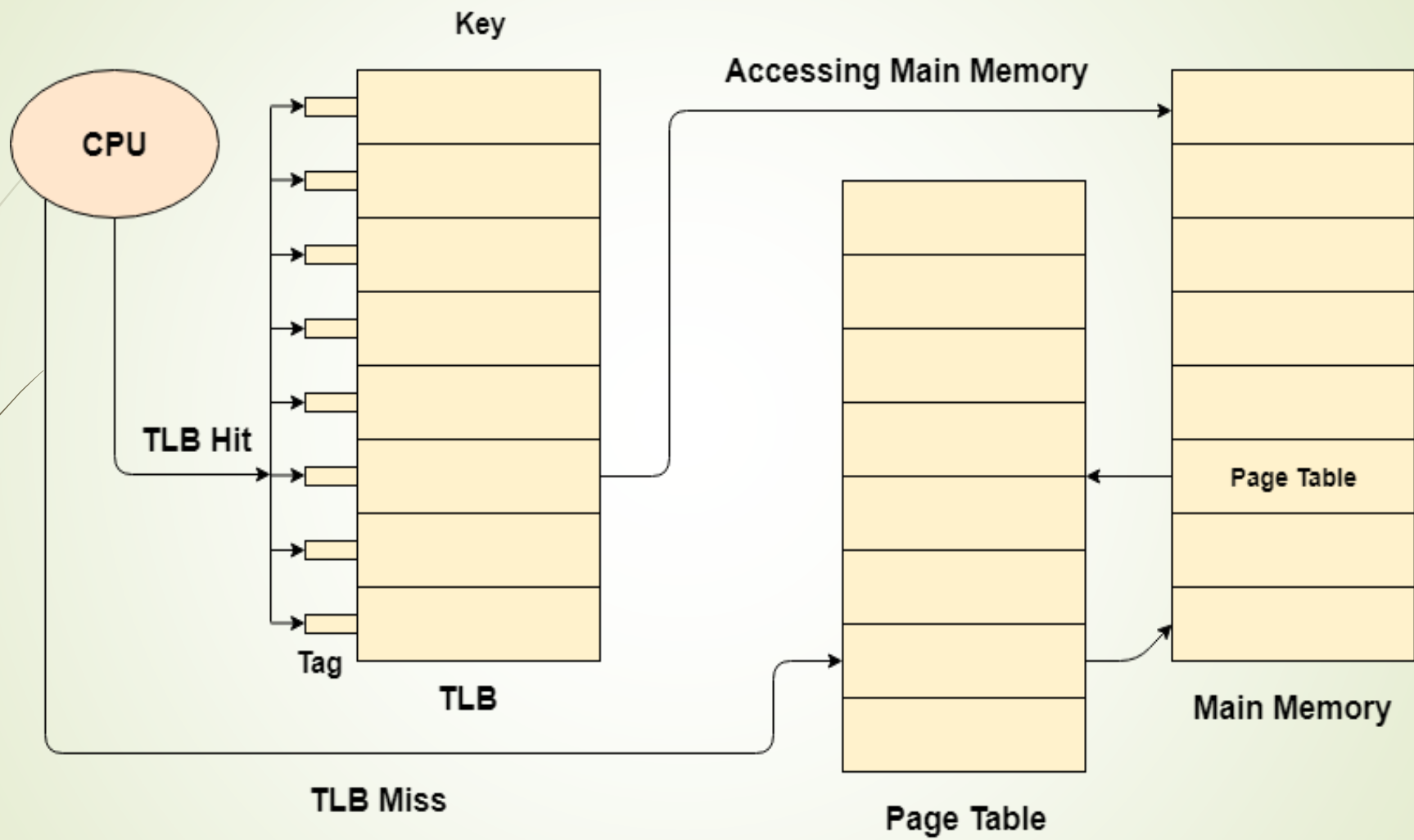
Inverted Page Table Architecture





Paging with Translation look aside buffer (TLB)

- ▶ A Translation look aside buffer can be defined as a memory cache which can be used to reduce the time taken to access the page table again and again.
- ▶ It is a memory cache which is closer to the CPU and the time taken by CPU to access TLB is lesser than that taken to access main memory.
- ▶ In other words, we can say that TLB is faster and smaller than the main memory but cheaper and bigger than the register.
- ▶ TLB follows the concept of locality of reference which means that it contains only the entries of those many pages that are frequently accessed by the CPU.





Continue..

- In translation look aside buffers, there are tags and keys with the help of which, the mapping is done.
- TLB hit is a condition where the desired entry is found in translation look aside buffer. If this happens then the CPU simply access the actual location in the main memory.
- However, if the entry is not found in TLB (TLB miss) then CPU has to access page table in the main memory and then access the actual frame in the main memory.
- Therefore, in the case of TLB hit, the effective access time will be lesser as compare to the case of TLB miss.



Advantages & Disadvantages of Paging

Advantages :

- It allows to store parts of a single process in a non-contiguous fashion.
- It solves the problem of external fragmentation.
- Supports multiprogramming.

Disadvantages :

- It suffers from internal fragmentation.
- There is an overhead of maintaining a page table for each process.
- Some memory will still be unused if the number of available blocks is not sufficient for the address spaces of the jobs to be run.



Segmentation

- Segmentation is a memory management technique in which, the memory is divided into the variable size parts. Each part is known as segment which can be allocated to a process.
- The details about each segment are stored in a table called as segment table. Segment table is stored in one (or many) of the segments.
- 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.

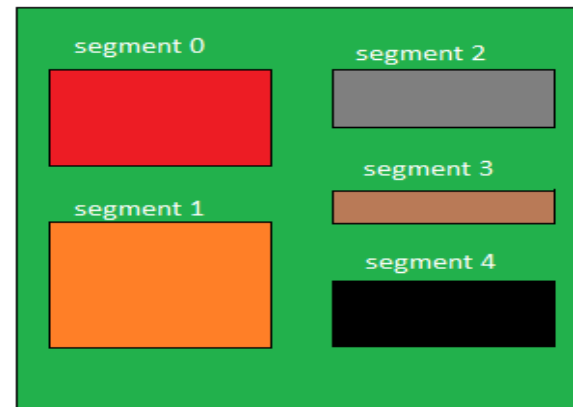


Continue..

- 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.
- 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.
- Segment map table contains 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.

Continue..

Logical View of Segmentation

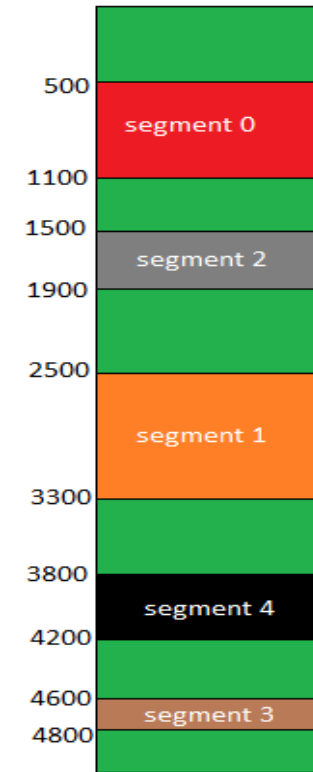


Logical Address Space

Segment Number

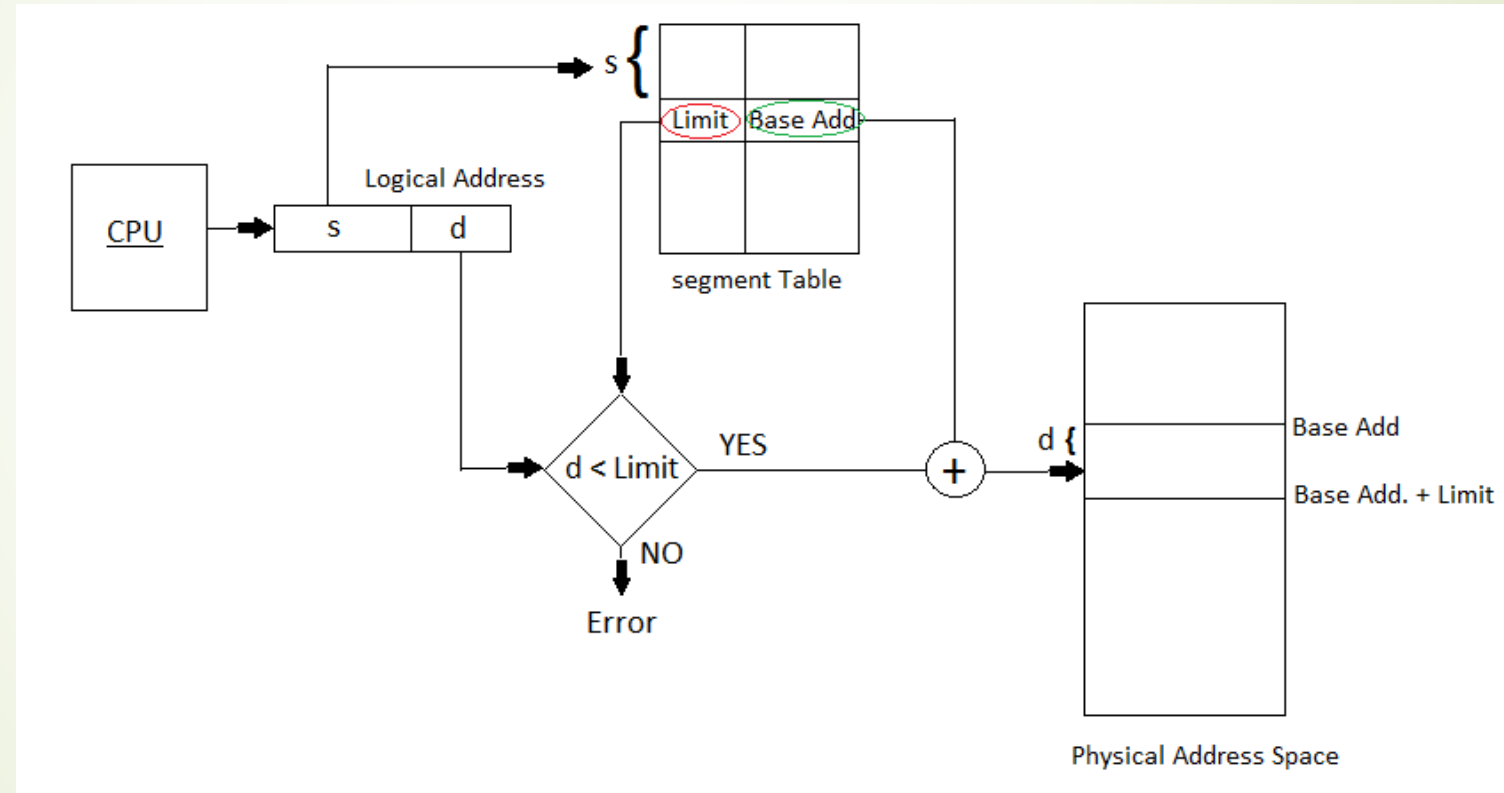
	base address	Limit
0	500	600
1	2500	800
2	1500	400
3	4600	200
4	3800	400

Segment Table



Physical Address Space

Translation of Two dimensional Logical Address to one dimensional Physical Address.





Advantages & Disadvantages

- **Advantages of Segmentation –**

- No Internal fragmentation.
- Segment Table consumes less space in comparison to Page table in paging.

- **Disadvantage of Segmentation –**

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



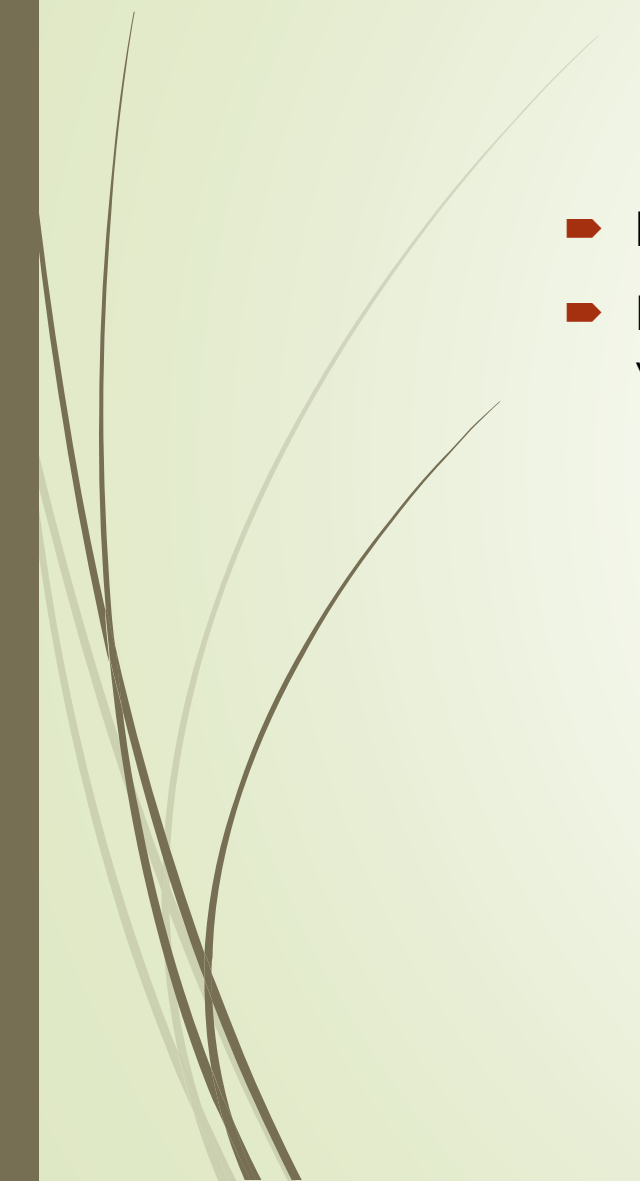
Virtual Memory



- Memory is hardware that your computer uses to load the operating system and run programs.
- Computer consists of one or more RAM chips that each have several memory modules.
- The amount of real memory in a computer is limited to the amount of RAM installed. Common memory sizes are 1GB, 2GB, and 4GB.
- Because your computer has a finite amount of RAM, it is possible to run out of memory when too many programs are running at one time.
- This is where virtual memory comes in.
- Virtual memory increases the available memory of your computer by enlarging the "address space," or places in memory where data can be stored.

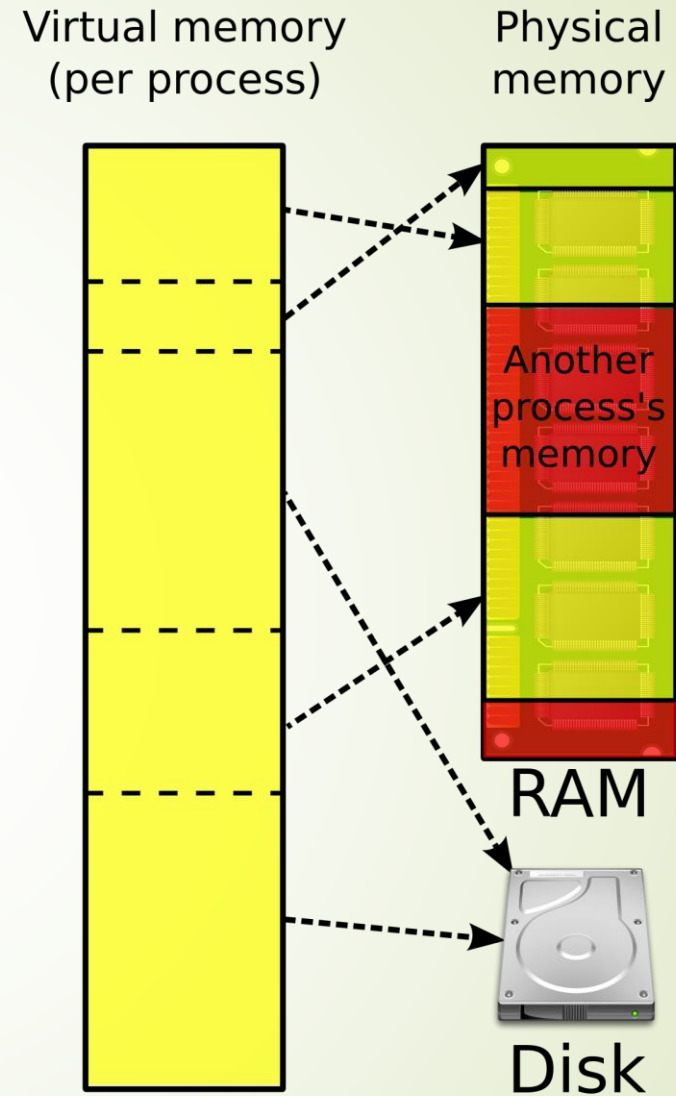


Virtual Memory

- It does this by using hard disk space for additional memory allocation.
 - However, since the hard drive is much slower than the RAM, data stored in virtual memory must be mapped back to real memory in order to be used.
- 

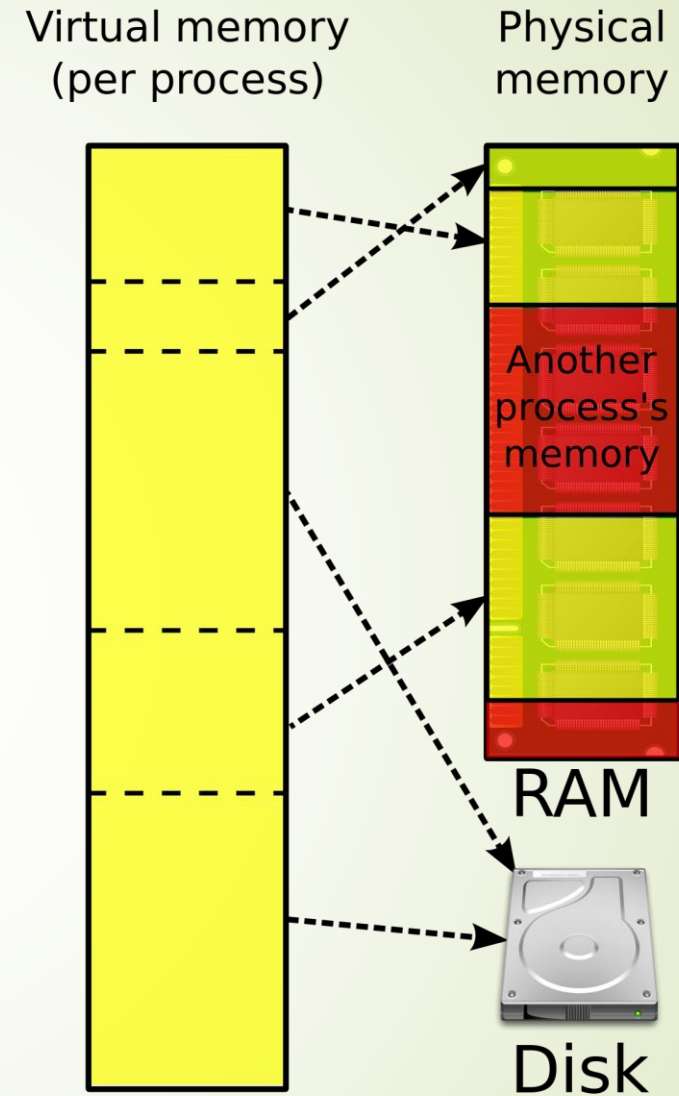
Virtual Memory

- Each program has its own address space, which is broken up into pages.
- Each page is a contiguous range of addresses.
- 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.
- The operating system keeps those parts of the program currently in use in main memory, and the rest on the disk.



Virtual Memory

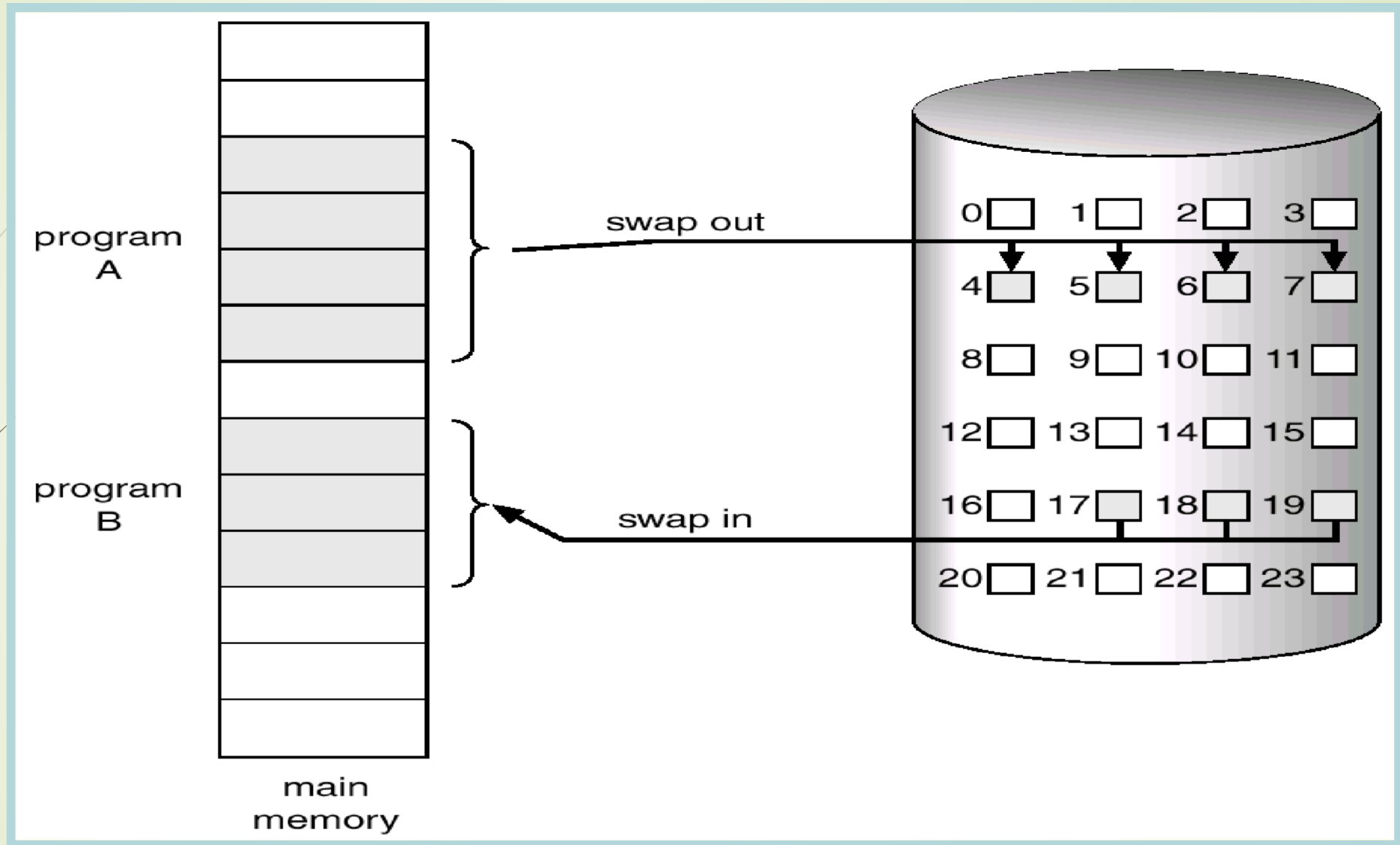
- In a system using virtual memory, the physical memory is divided into page frames and the virtual address space is divided in into equally-sized partitions called pages.
- Virtual memory works fine in a multiprogramming system, with bits and pieces of many programs in memory at once.





Demand Paging

- Demand paging system is similar to a paging system with swapping
- A lazy swapper swaps a page into memory only when it is needed
- A process is a sequence of pages rather than one large contiguous address space
- In demand paging instead of swapper the term pager is used
- Bring a page into memory only when it is needed
- When a page is referenced, and that page isn't in memory, then get the page from disk and re-execute the statement.



Valid-Invalid Bit

- With each page table entry a valid-invalid bit is associated
- (1 -> in-memory, 0 -> not-in-memory)
- Initially valid-invalid bit is set to 0 on all entries
- Example of a page table snapshot
- During address translation, if valid-invalid bit in page table entry is
- 0 → page fault

Frame #	Valid-invalid bit
	1
	1
	1
	1
	0
	0
	0

Page Table When Some Pages Are Not in Main Memory

0	A
1	B
2	C
3	D
4	E
5	F
6	G
7	H

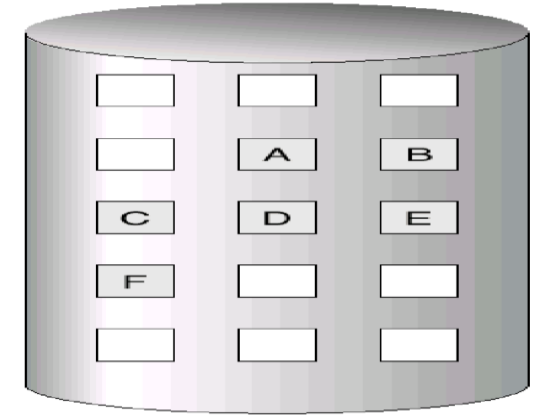
logical memory

	frame	valid-invalid bit
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





Page Fault

Steps in handling a page fault

When a page fault occurs a system performs the following steps:

- Check the process control block table for reference which bit is set i.e. valid bit or invalid bit for memory access.
- If invalid bit is set user will terminate the process. If valid bit is loaded then page is loaded into main memory.
- Check for free frame for new page loading.
- Start reading secondary storage device for desired page for allocated frame.
- Required page loaded into memory after completion of read operation.
- System restart the instruction that was interrupted by error. page is loaded into memory so process can use it.



Page Replacement Policies

- Please refer below link for the algorithms
- <https://www.geeksforgeeks.org/page-replacement-algorithms-in-operating-systems/>

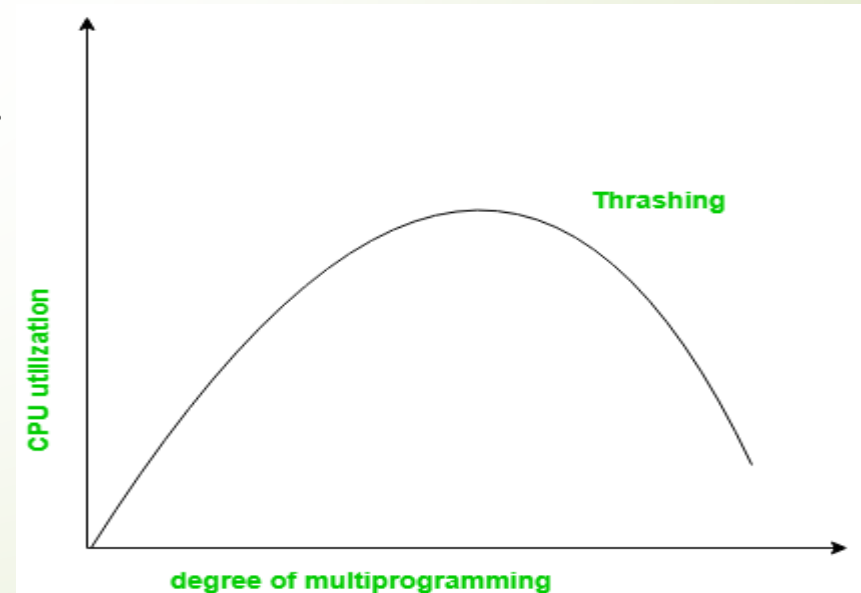


Thrashing

- When a program need space larger than RAM or it need space when RAM is full, Operating System will try to allocate space from secondary memory and behaves like it has that much amount of memory by serving to that program.
- This concept is called virtual memory. To know about thrashing we first need to know what is page fault and swapping.
- **Page fault and swapping:** We know every program divided into some pages.
- When a program need a page which is not in RAM that is called page fault.
- Whenever a page fault happens, operating system will try to fetch that page from secondary memory and try to swap it with one of the page in RAM. This is called swapping.

Continue..

- If this page fault and then swapping happening very frequently at higher rate, then operating system has to spend more time to swap these pages. This state is called **thrashing**. Because of this, CPU utilization is going to be reduced.
- Whenever thrashing starts, operating system tries to apply either **Global page replacement** Algorithm or **Local page replacement** algorithm.





Global & Local Page Replacement

Global Page Replacement

- Since global page replacement can access to bring any page, it tries to bring more pages whenever thrashing found. But what actually will happen is, due to this, no process gets enough frames and by result thrashing will be increase more and more. So global page replacement algorithm is not suitable when thrashing happens.

Local Page Replacement

- Unlike global page replacement algorithm, local page replacement will select pages which only belongs to that process. So there is a chance to reduce the thrashing. But it is proven that there are many disadvantages if we use local page replacement. So local page replacement is just alternative than global page replacement in thrashing scenario.



Techniques to Handle Thrashing

➤ **Working Set Model**

- This model is based on locality.
- What locality is saying, the page used recently can be used again and also the pages which are nearby this page will also be used.
- Working set means set of pages in the most recent D time. The page which completed its D amount of time in working set automatically dropped from it.
- So accuracy of working set depends on D we have chosen.
- This working set model avoid thrashing while keeping the degree of multiprogramming as high as possible.



Techniques to Handle Thrashing

Page Fault Frequency

- It is some direct approach than working set model. When thrashing occurring we know that it has few number of frames.
- if it is not thrashing that means it has too many frames. Based on this property we assign an upper and lower bound for the desired page fault rate.
- According to page fault rate we allocate or remove pages. If the page fault rate become less than the lower limit, frames can be removed from the process.
- Similarly, if the page fault rate become more than the upper limit, more number of frames can be allocated to the process.
- And if no frames available due to high page fault rate, we will just suspend the processes and will restart them again when frames available.

Continue..

