EDU ENGINEERING

PIONEER OF ENGINEERING NOTES

TAMIL NADU'S BEST EDTECH PLATFORM FOR ENGINEERING

CONNECT WITH US

WEBSITE: www.eduengineering.net

TELEGRAM: @eduengineering

INSTAGRAM: @eduengineering

- **Regular Updates for all Semesters**
- **All Department** Notes AVAILABLE
- **Handwritten** Notes AVAILABLE
- **Past Year** Question Papers AVAILABLE
- **Subject wise** Question Banks AVAILABLE
- **Important Questions** for Semesters AVAILABLE
- Various **Author Books** AVAILABLE

UNIT - III

STORAGE MANAGEMENT

1) Main Memory.
2) Background
3) Swapping
4) Contiguous Memory Allocation.
5) Paging
6) Segmentation
7) Segmentation with paging.
8) 32 and 64 bit architecture
9) Examples
10) Virtual memory
11) Background
12) Demand Paging
13) Page Replacement
14) Allocation
15) Thrashing
16) Allocating Kernel Memory
17) OS Examples.

(1) Main Memory :

Memory management is the functionality of an operating system which handles (or) manages primary memory and moves processes back and forth between main memory and disk during execution.

It keeps track of each and every memory location, regardless of either it is allocated to Processes. It decides which Process will get memory at what time. It tracks whenever some memory gets freed (or) Unallocated and correspondingly it updates the status.

(2) Background :

* Basic Hardware
* Address Binding
* Logical Vs physical Address space
* Dynamic Loading
* Dynamic Linking & Shared Libraries

In general, to run a Program, it must be brought into memory.

Input Queue - Collection of Processes on the disk that are waiting to be brought into memory to run the program.

User Programs go through Several steps before being run.

Address binding : Mapping of instructions and data from one address to another address in memory.

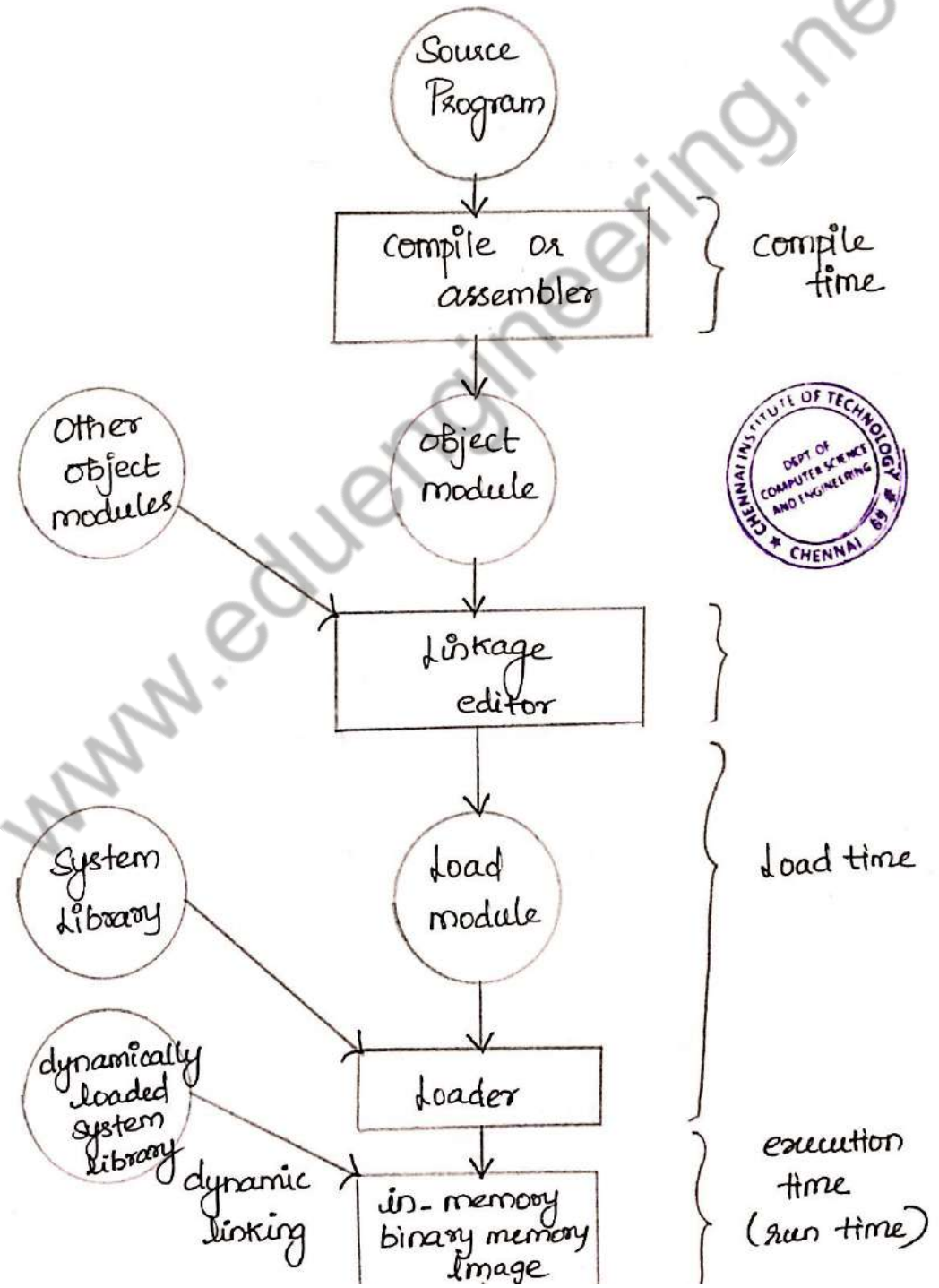
Basic Hardware and address Binding:

1) compile time : Must generate absolute code if memory location is known in prior.

2) Load time : Must generate relocatable code if memory location is not known at compile time.

3) Execution time : Need hardware support for address maps (eg., base and limit registers).

Multistep Processing of a User Program.



* Logical Vs Physical address space [N/D - 2019]

An address generated by the CPU is commonly referred to as a logical address, whereas an address seen by the memory unit - that is, the one loaded into the memory-address register of the memory - is commonly referred to as a physical address.

Logical and physical addresses are the same in compile-time and load-time address-binding schemes. Logical (virtual) and physical addresses differ in execution-time address-binding scheme.

The set of all logical addresses generated by a program is a logical address space; the set of all physical addresses corresponding to these logical addresses is a physical address space.

Memory-Management Unit (MMU):

✓ The run time mapping from virtual to physical addresses is done by a hardware device called the memory-management unit (MMU).

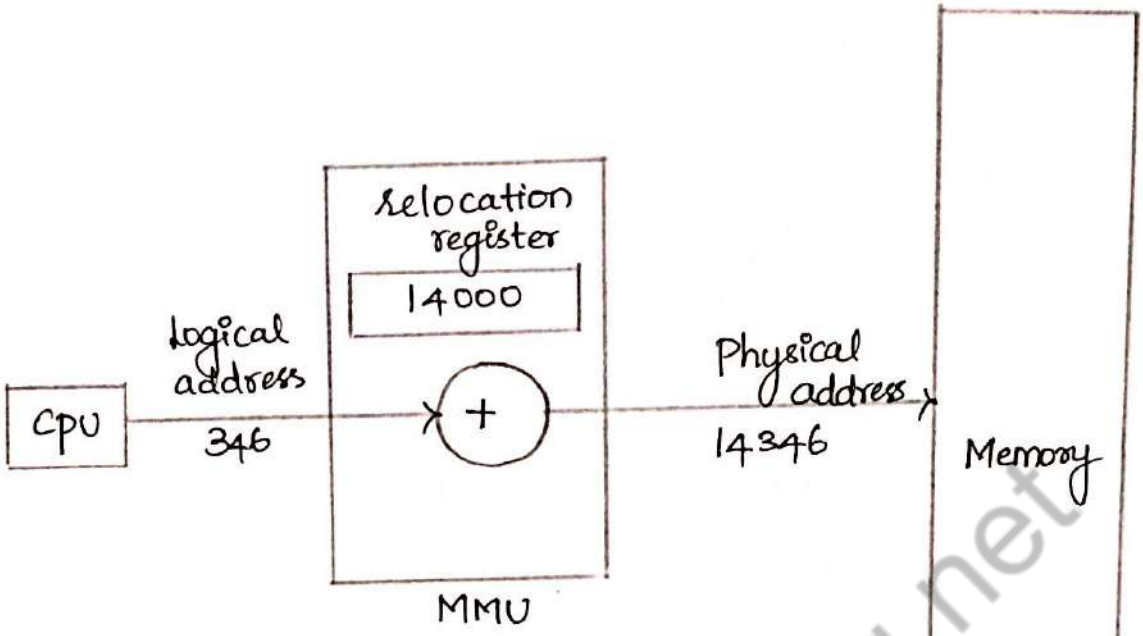
✓ It is a hardware device that maps virtual/logical address to physical address.

✓ In this scheme, the relocation register's value is added to logical address generated by a user process.

✓ The user program deals with logical addresses; it never sees the real physical addresses.

✓ Logical address range : 0 to max.

✓ Physical address range : R+0 to R+ max,
where R - value in relocation register.



for eg., if the base is at 14000, then an attempt by the user to address location 346 is mapped to location 14346. The User Program deals with logical addresses. The memory-mapping hardware converts logical addresses into physical addresses.

Dynamic Loading :

Though this, the routine is not loaded until it is called.

* Better memory-space utilization ; Unused routine is never loaded.
* Useful when large amounts of code are needed to handle infrequently occuring cases.
* No special support from the operating system is required implemented through program design.

Dynamic Linking and Shared Libraries :

✓ Linking postponed until execution time and is particularly useful for libraries.

✓ Small piece of code called Stub used to locate the appropriate memory-resident library routine (or) function.

✓ Stub replaces itself with the address of the routine, and executes the routine.

✓ operating system needed to check if routine is in processes' memory address.

Shared libraries : Programs linked before the new library was installed will continue using the older library.

(3) Swapping : [NID-19]

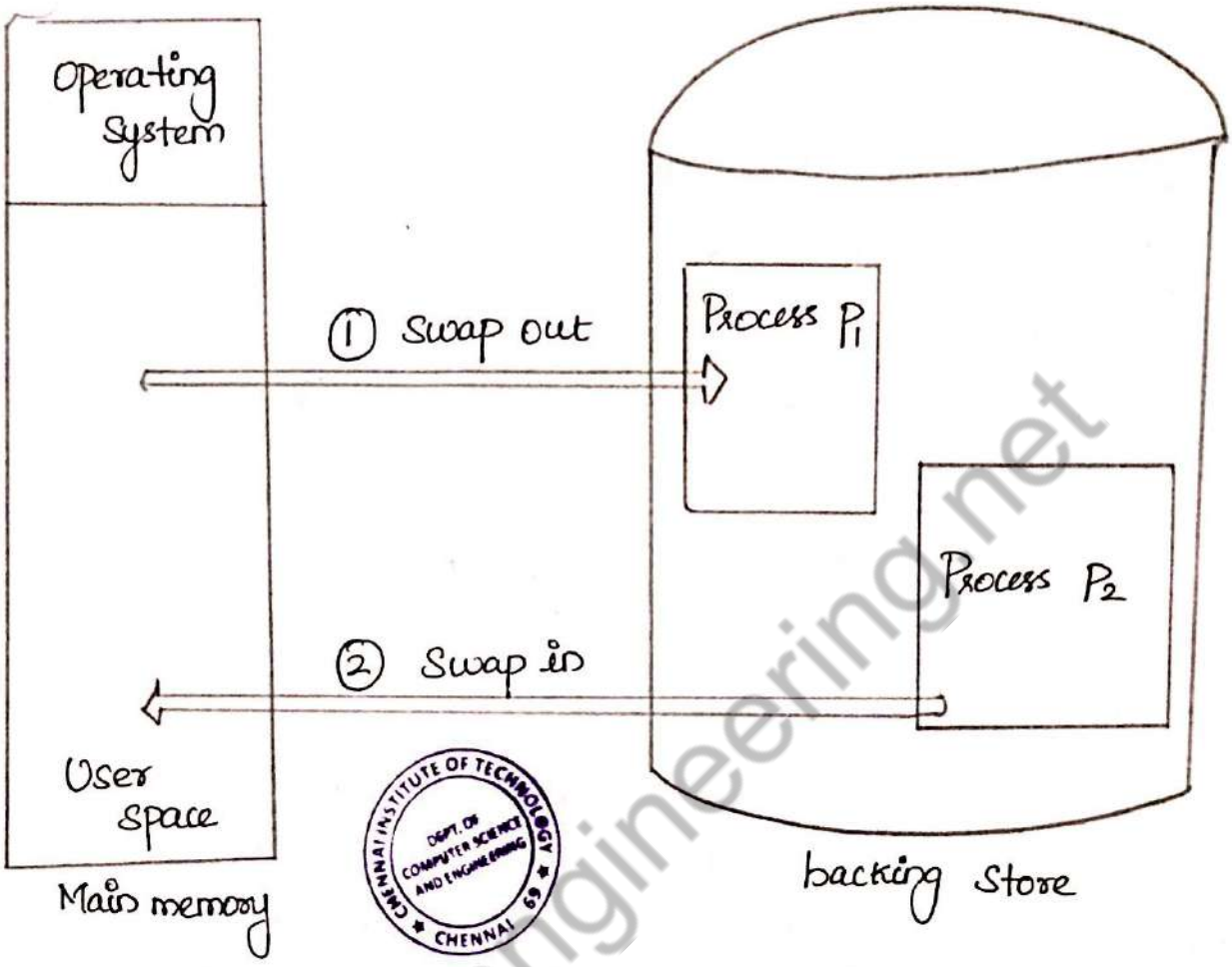
A process must be in memory to be executed. A process can be swapped temporarily out of memory to a backing store (SWAP OUT) and then brought back into memory for continued execution (SWAP IN).

✓ Backing Store - fast disk large enough to accommodate copies of all memory images for all users and it must provide direct access to these memory images.

✓ Roll out, Roll in - Swapping variant used for Priority-based scheduling algorithms; lower-priority Process is swapped out so higher-priority process can be loaded and executed.

✓ Transfer time - Major part of swap time is transfer time. Total transfer time is directly Proportional to the amount of memory swapped.

Swapping of two processes using a disk as a backing store.



* Swapping requires a backing store (normally a fast disk)

* The backing store must be big enough to accommodate all copies of memory images for all users, and must provide direct access.

* The system has a ready queue with all processes, whose memory images are on the backing store or in memory and ready to run.

→ The cpu scheduler calls the dispatcher before running a process.

→ The dispatcher checks if the next process in queue is in memory.

* If not and there is no free memory region, the dispatcher swaps out a process currently in memory and swaps in the desired one.
* It then reloads registers and transfers control to the process.
* The context-switch time in such a swapping system is fairly high.

Eg; Let us assume the user process is of size 1 MB and the backing store is a standard hard disk with a transfer rate of 5 MBPS.

Transfer time = 1000 KB / 5000 KB per second.
= 1/5 sec = 200 ms.

(ii) Contiguous Memory Allocation :

It is a classical memory allocation model that assigns a process consecutive 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. Each process is contained in a single contiguous section of memory.

There are two methods namely;

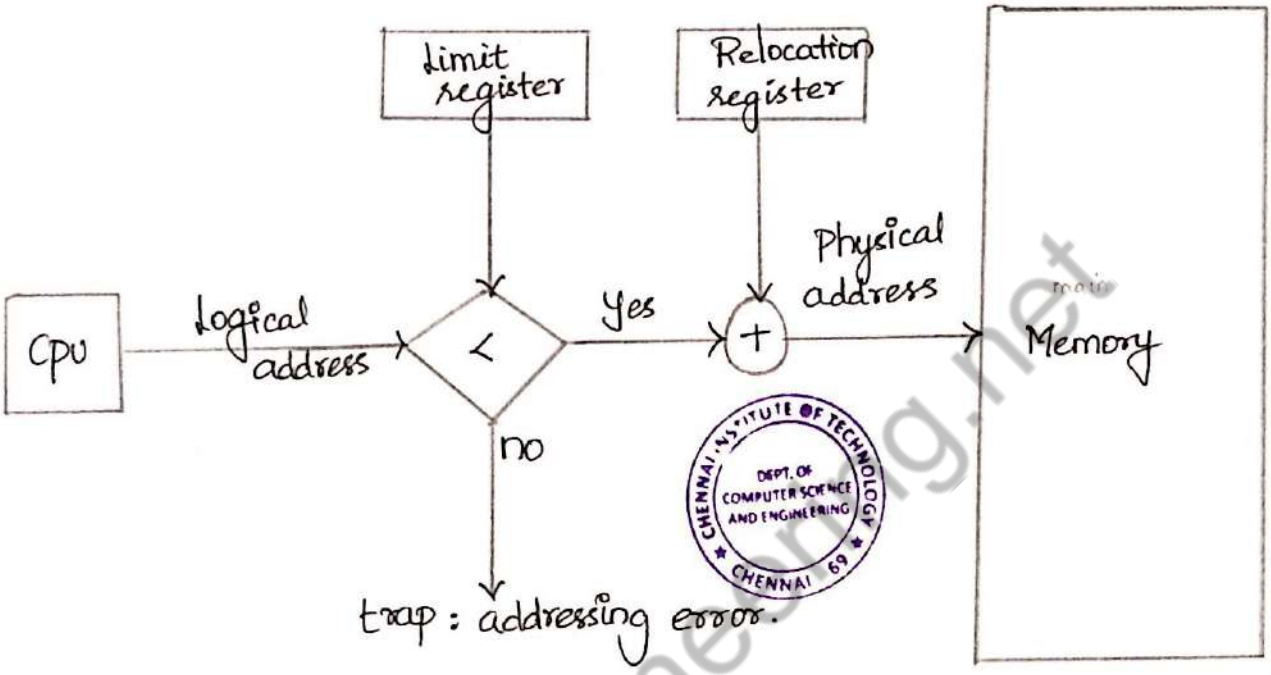
* fixed-partition method.
* Variable-partition method.

fixed-partition Method :

⇒ Divide memory into fixed size partitions, where each partition has exactly one process.

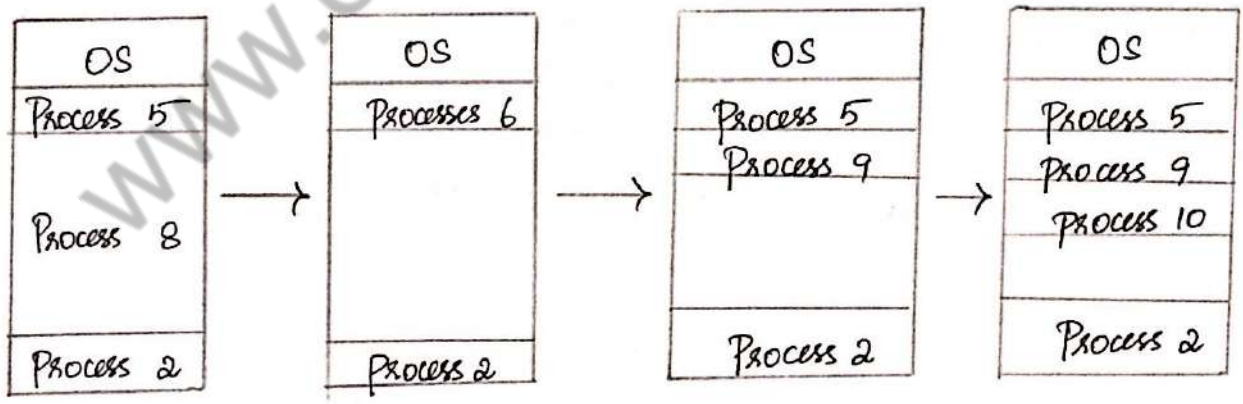
Internal fragmentation - memory space allocated within a partition is wasted. Eg., When Process size < Partition size.

Hardware support to relocation and limit registers.



Variable - Partition Method :

* Divide memory into variable size partitions, depending upon the size of the incoming Process. When a Process terminates, the partition becomes available for another process. As processes complete and leave they create holes in the main memory.



Hole – Block of available memory ; holes of various size are scattered throughout memory.

Dynamic storage - Allocation problem:

How to satisfy a request of size n from a list of free holes?

Solution:

- first-fit: Allocate the first hole that is big enough.
- Best-fit: Allocate the smallest hole that is big enough; must search entire list; Unless Ordered by size. Produces the smallest leftover hole.
- Worst-fit: Allocate the largest hole; must also search entire list. Produces the largest leftover hole.

Note: first-fit and best-fit are better than worst-fit in terms of speed and storage utilization.

Fragmentation:

* External fragmentation
* Internal fragmentation.

External fragmentation:

This takes place when enough total memory space exists to satisfy a request, but it is not contiguous ie., Storage is fragmented into a large number of small holes scattered throughout the main memory.

Internal fragmentation:

Allocated memory may be slightly larger than requested memory. Eg: hole = 184 bytes.
Process size = 182 bytes.
We are left with a hole of 2 bytes.

Solutions:

1) Coalescing: Merge the adjacent holes together.
2) Compaction: Move all processes towards one end of memory, hole towards other end of memory, producing one large hole of available memory. This scheme is expensive as it can be done if relocation is dynamic and done at execution time.

3) Permits the logical address space of a process to be non-contiguous. This is achieved through two memory management schemes namely Paging and segmentation.

(5) Paging : [N|D-19]

- ✓ Basic Method
- ✓ Hardware Support
- ✓ Protection
- ✓ Shared pages
- ✓ Structure of Page table

Paging : It is a memory management scheme that Permits the physical address space of a process to be noncontiguous. It avoids the considerable problem of fitting the varying size memory chunks on to the backing store.

Basic Method :-

→ Divide logical memory into blocks of same size called "pages"

⇒ Divide physical memory into fixed-sized blocks called "frames"

⇒ Page size is a power of 2, between 512 bytes and 16 MB.

Address Translation scheme :

Address generated by cpu (logical address) is divided into ;

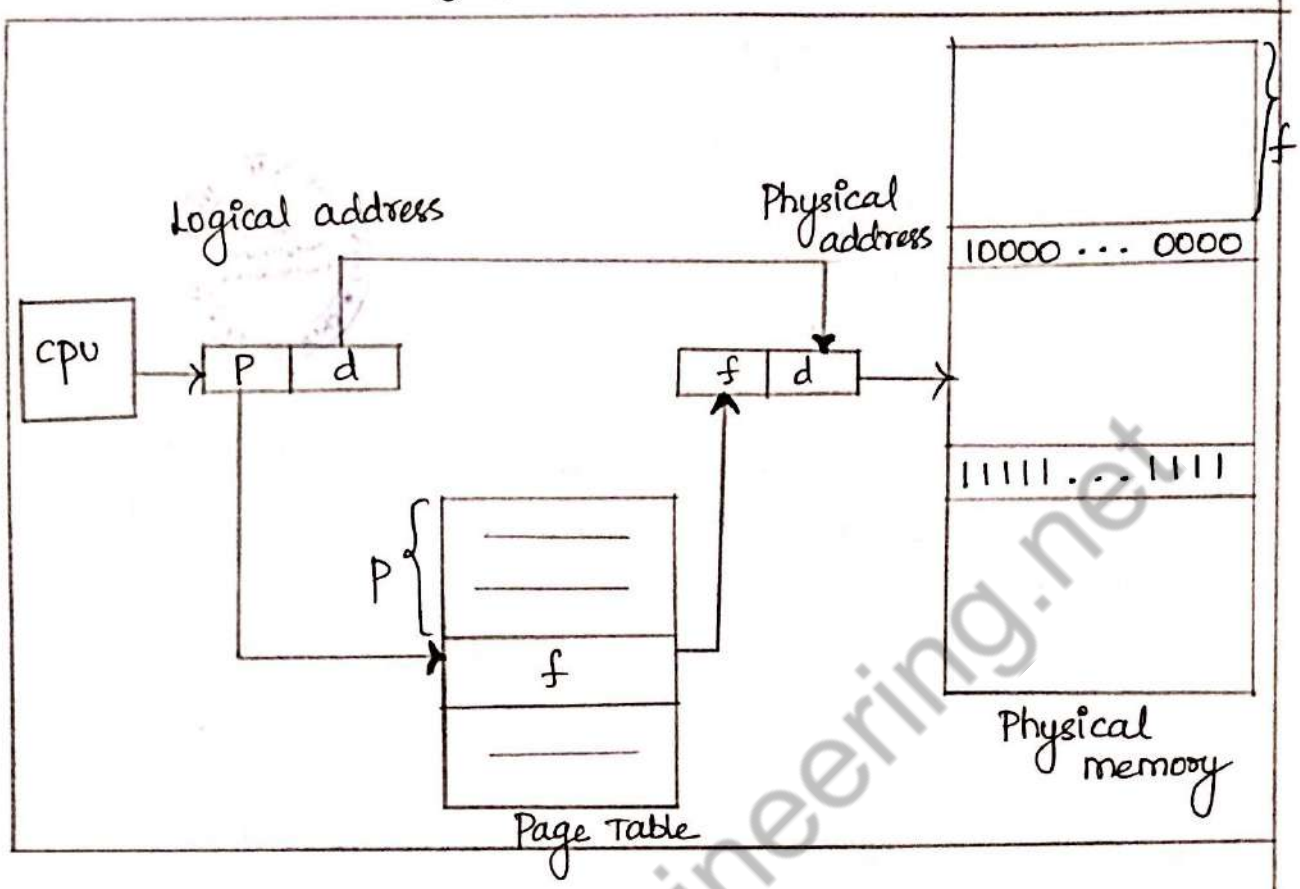
✓ Page number (P) – Used as an index into a page table which contains base address of each page in Physical memory.

✓ Page offset(d) - Combined with base address to define the physical address ie.,

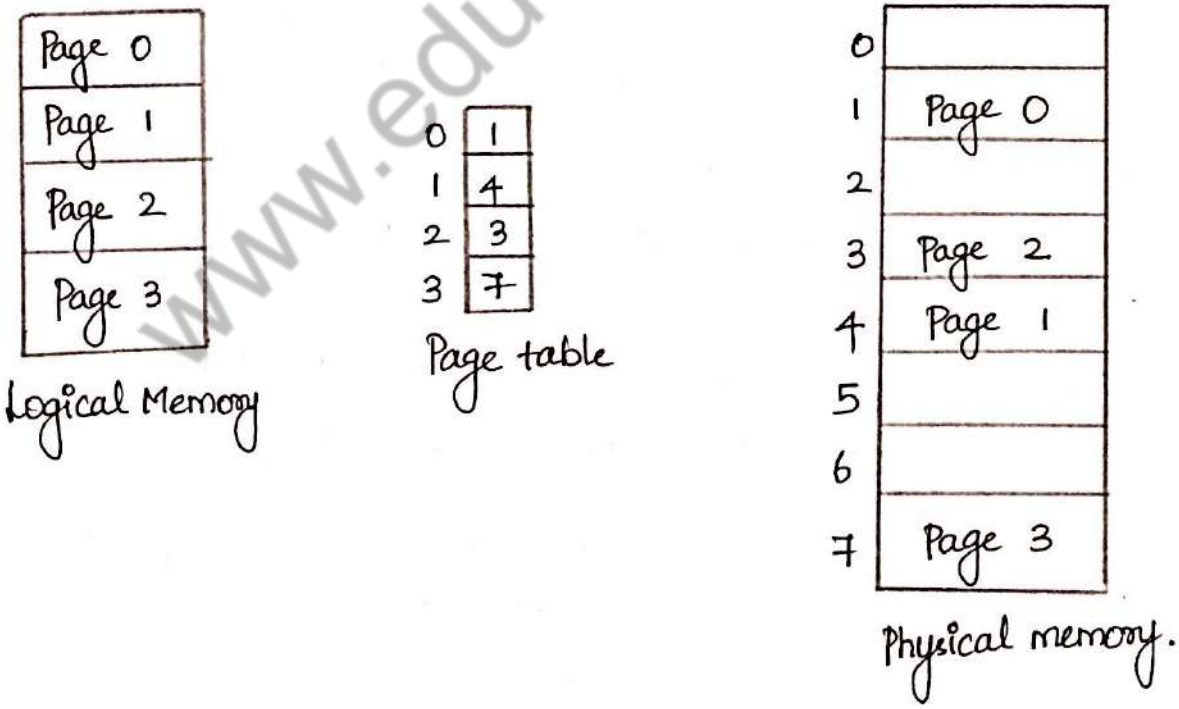
Physical address = base address + offset.

* Hardware Support:

Paging Hardware:



Paging model of logical and physical memory.



Paging example for a 32-byte memory with 4-byte pages:-

Page size = 4 bytes.

Physical memory size = 32 bytes ie., (4 × 8 = 32 so, 8 pages)

logical address = 0 (maps to physical address 20 ie., (5×4)+0) when frame no = 5, Page size = 4, offset = 0.

Paging example for a 32-byte memory with 4-byte pages.

0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k
11	l
12	m
13	n
14	o
15	p

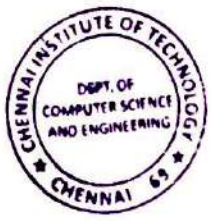
Logical Memory

0	5
1	6
2	1
3	2

Page table

0	
4	i j k l
8	m n o p
12	
16	
20	a b c d
24	e f g h
28	

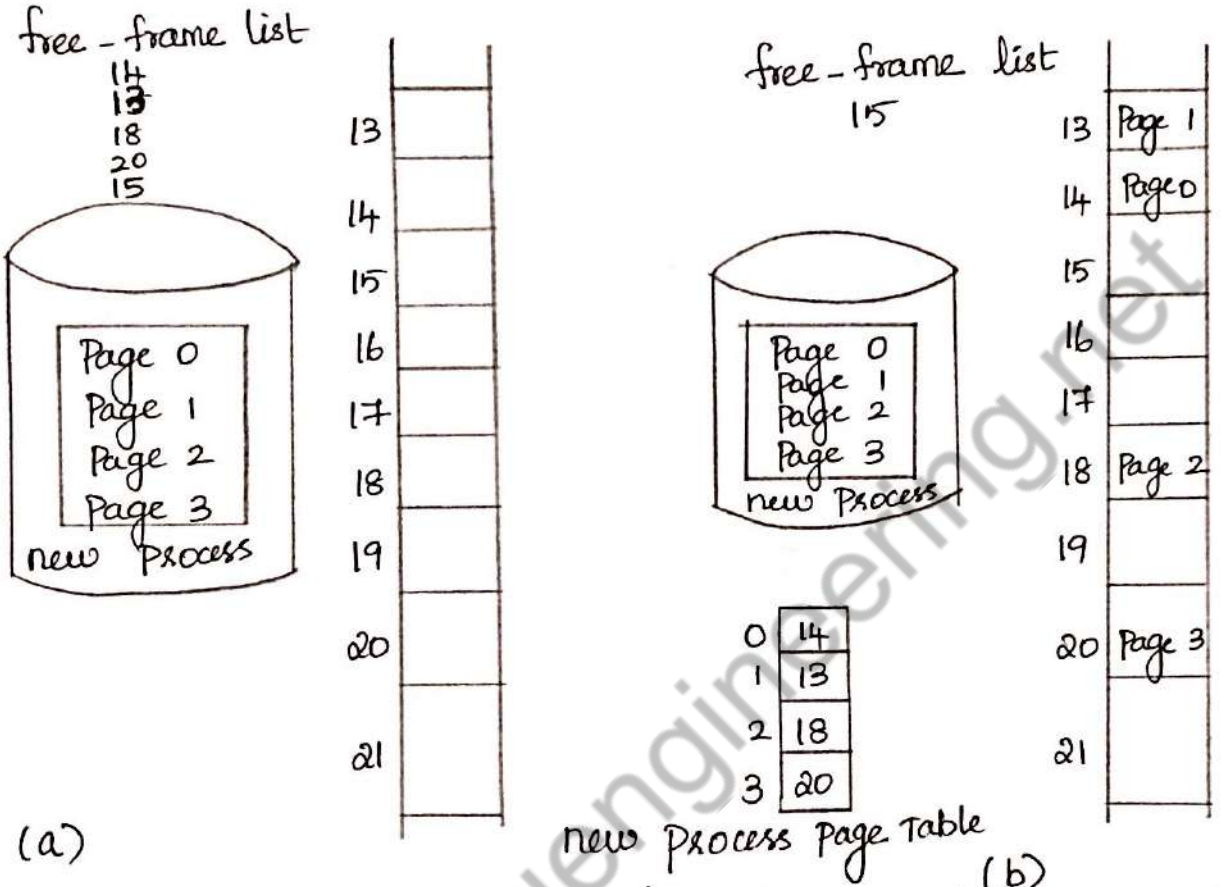
Physical Memory



Allocation:

- ✓ When a process arrives into the system, its size (expressed in pages) is examined.
- ✓ Each page of process needs one frame. Thus if the Process requires n pages, at least n' frames must be available in memory.
- ✓ If n' frames are available, they are allocated to this arriving process.

* The 1st page of the process is loaded into one of the allocated frames and the frame number is put in the page table.
* Repeat the above step for the next pages and so on.



Frame table: It is used to determine which frames are allocated, which frames are available, how many total frames are there, and so on. (ie) It contains all the information about the frames in the physical memory.

Hardware implementation of Page Table:

This can be done in several ways:

1) Using PTBR
2) TLB.

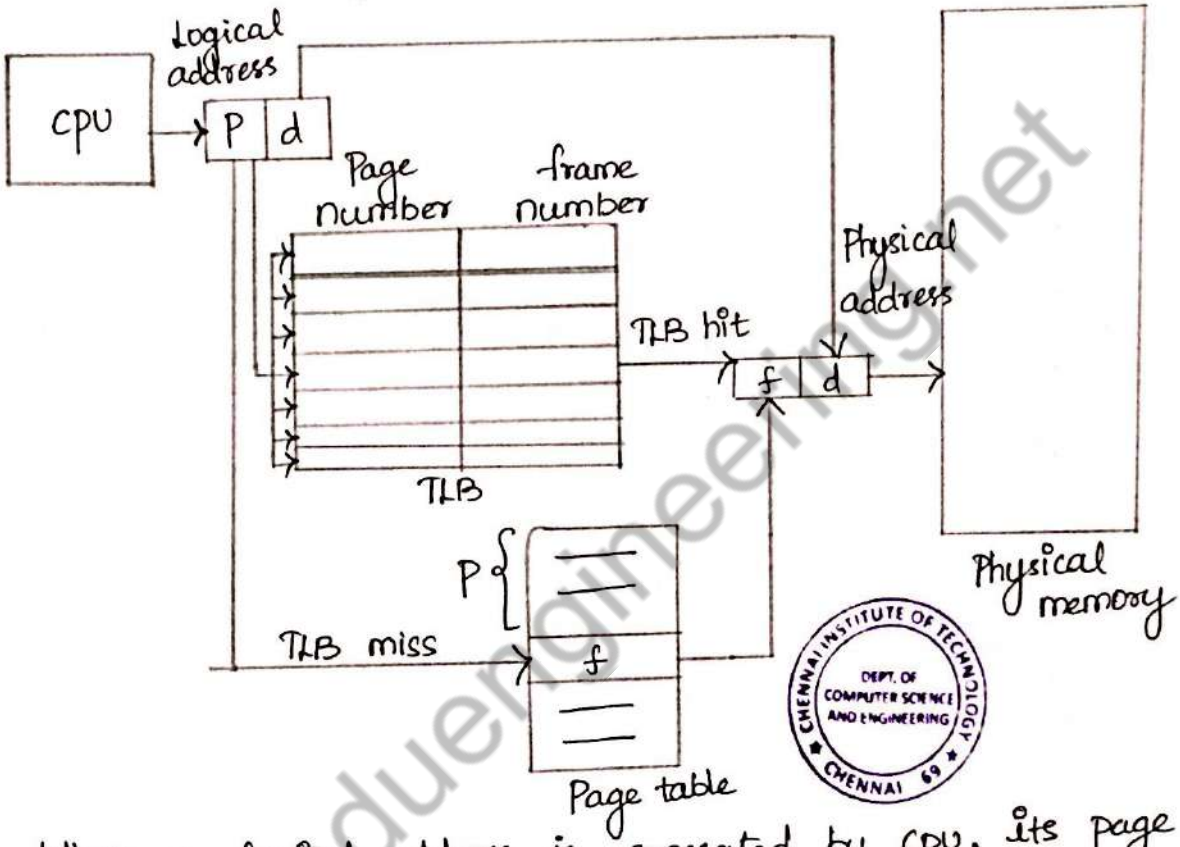
The simplest case is Page-table base register (PTBR), is an index to point the page table.

TLB (Translation look aside Buffer)

- It is a fast lookup hardware cache.
- It contains the recently or frequently used page table entries.
- It has two parts : key (tag) & value.

More expensive.

Paging Hardware with TLB.



- When a logical address is generated by CPU, its page number is presented to TLB.

TLB hit : If the page number is found, its frame number is immediately available and is used to access memory.

TLB miss : If the page number is not in the TLB, a memory reference to the page table must be made.

Hit ratio : Percentage of times that a particular page is found in the TLB.

⇒ for Eg., hit ratio is 80% means that the desired page number in the TLB is 80% of the time.

Effective Access time:

- Assume hit ratio is 80%.
- If it takes 20 ns to search TLB and 100 ns to access memory, then the memory access takes 120 ns (TLB hit)
- If we fail to find page no. in TLB (20 ns), then we must 1st access memory for page table (100 ns) and then access the desired byte in memory (100 ns).

$\therefore$ Total = 20 + 100 + 100
= 220 ns (TLB miss)

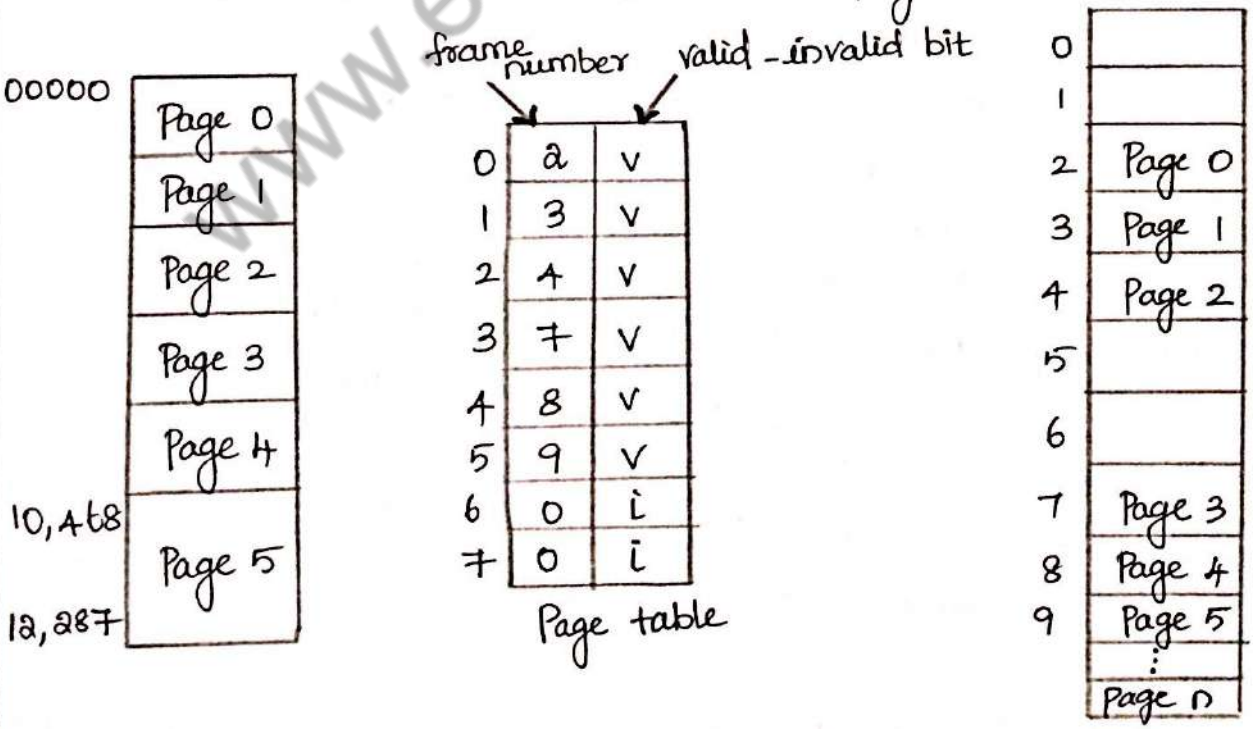
Then Effective Access Time (EAT) = $0.80 \times (120 + 0.20) \times 220$
= 140 ns.

* Protection :

Memory protection implemented by associating protection bit with each frame. Valid - invalid bit attached to each entry in the page table:

- ✓ Valid (v)" indicates that the associated page is in the process' logical address space, and is thus a legal page.
- ✓ Invalid (i)" indicates that the page is not in the Process' logical address spaces.

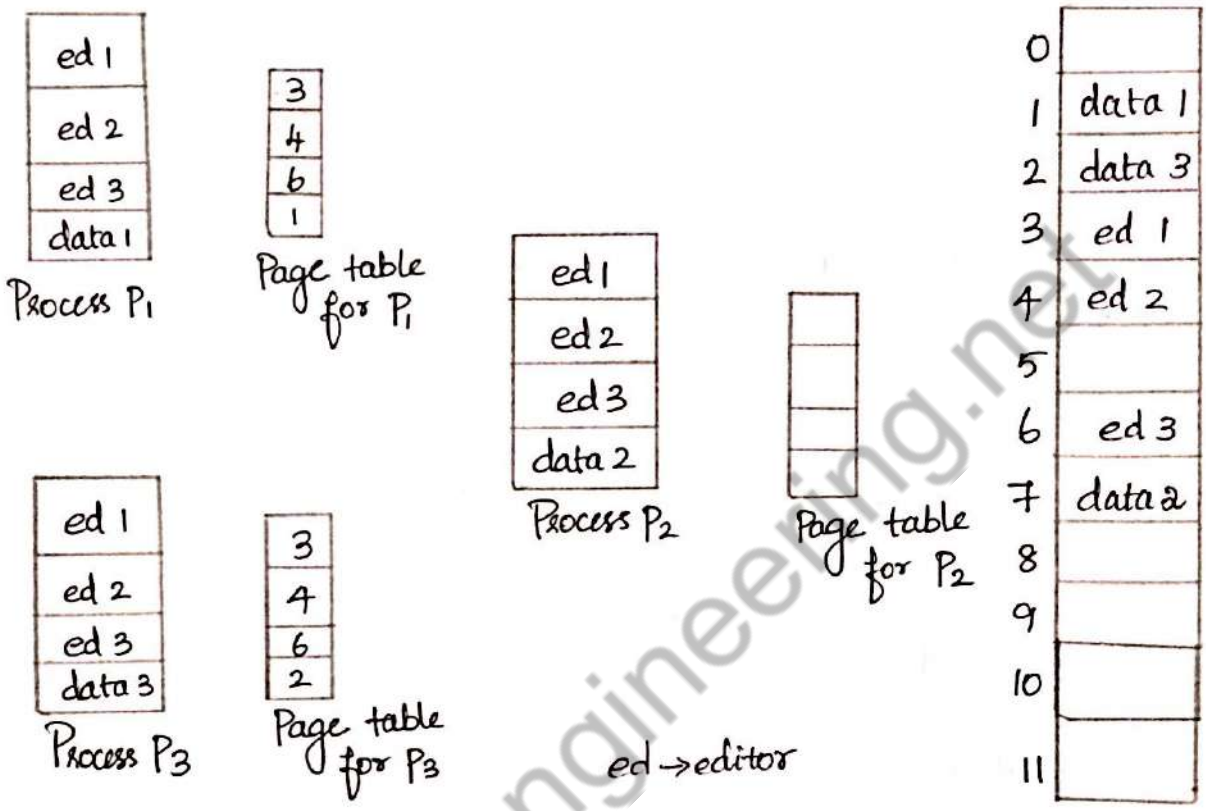
Valid (v) or invalid (i) bit in a page table.



* Shared pages:

An advantage of paging is the possibility of sharing common code. This consideration is particularly important in a time-sharing environment.

Sharing of code in a paging environment.

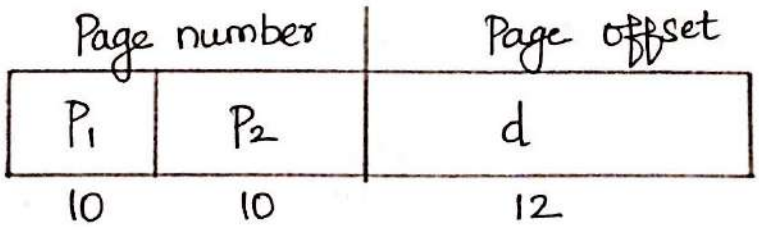


Structures of the page Table:

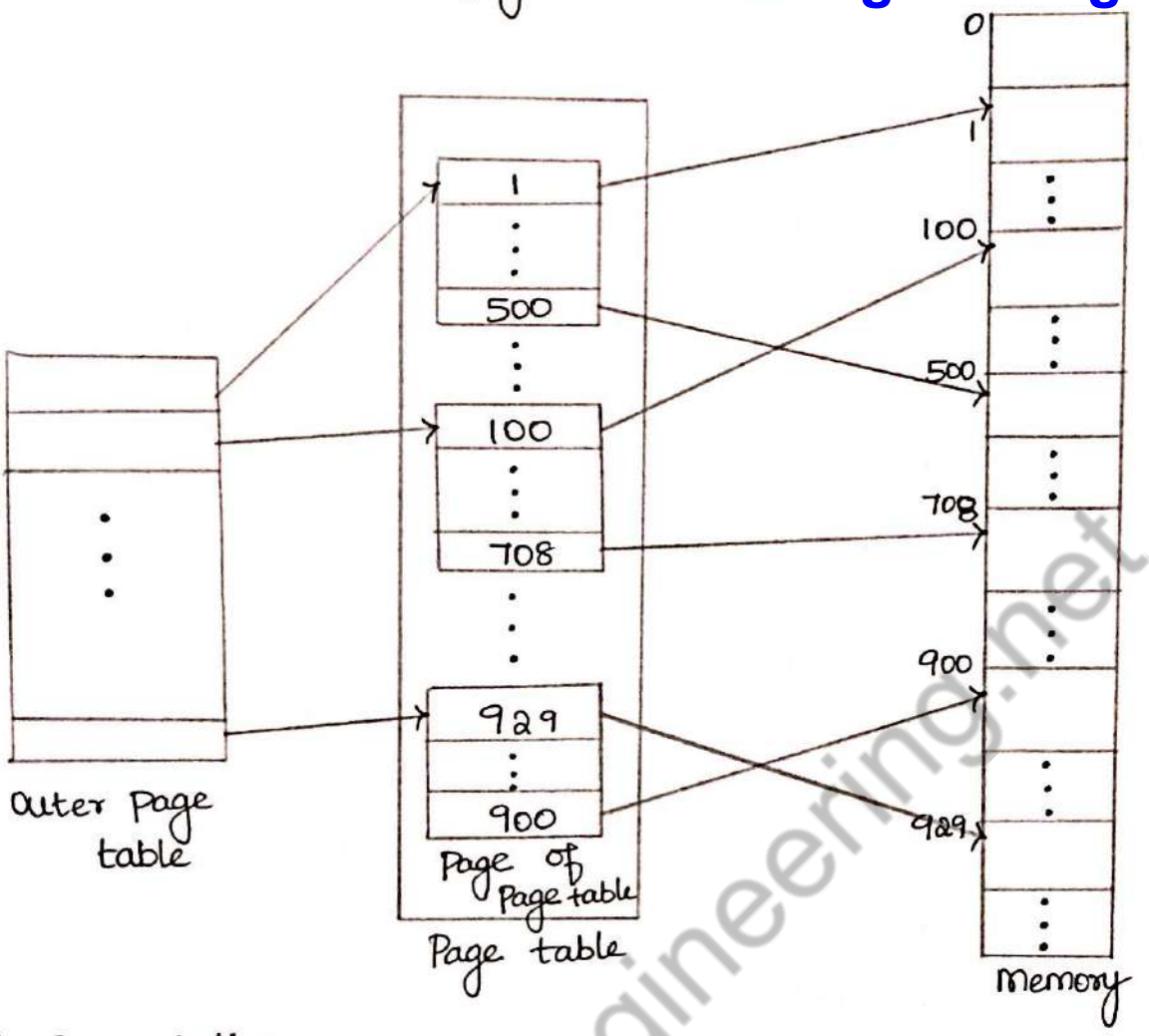
Hierarchical paging:

Break up the page table into smaller pieces. Because if the page table is too large then it is Quite difficult to search the page number.

Eg: "Two-level paging".



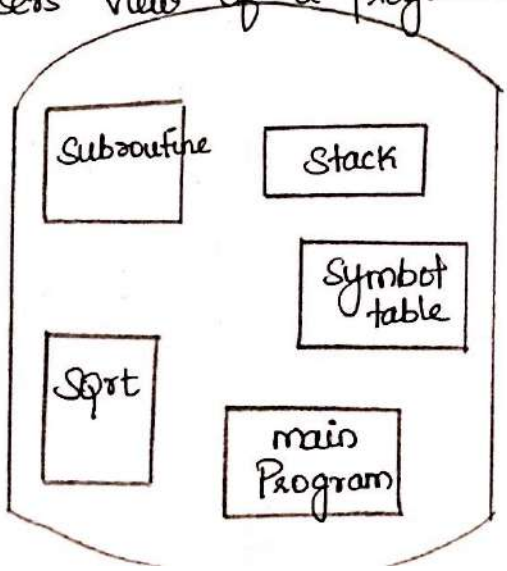
A two-level Page table scheme:



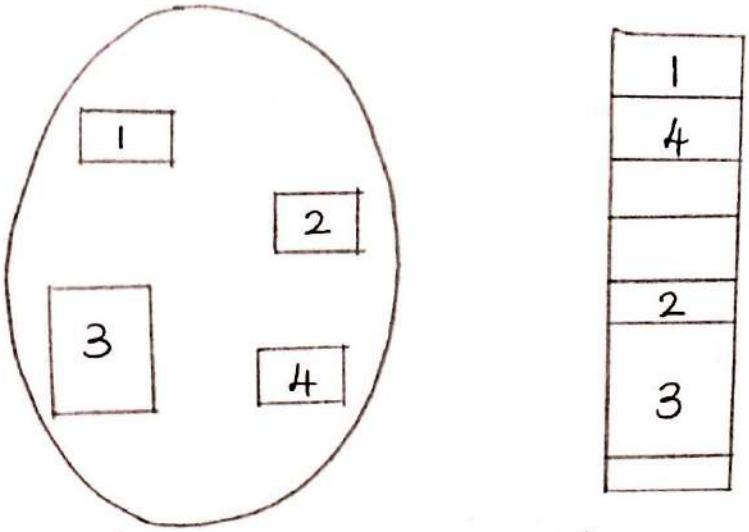
(6) Segmentation :

- Memory management scheme that supports user view of memory.
- A program is a collection of segments. A segment is a logical unit such as: Main program, Procedure, function, Method, object, Local variables, global variables, common block, Stack, symbol table, arrays.

User's View of a Program.



Logical View of Segmentation

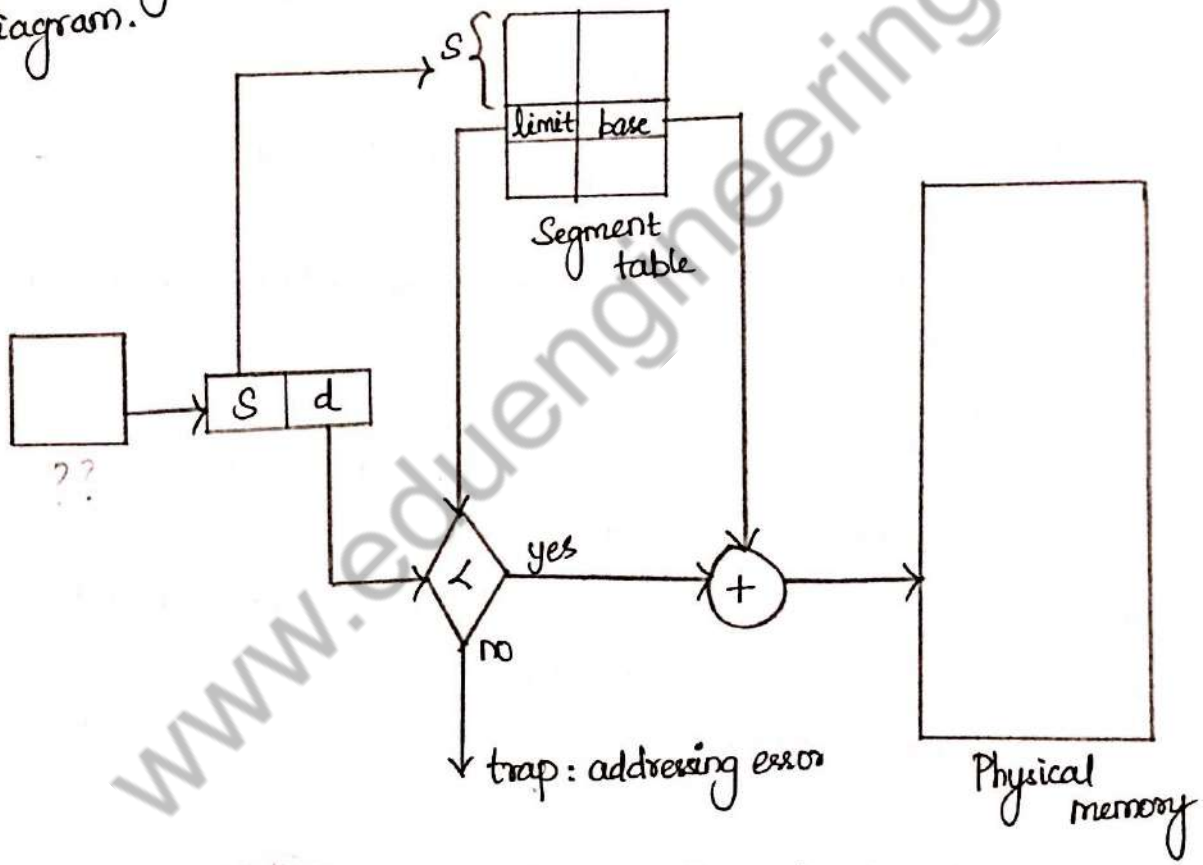


User space Physical memory space

Segmentation Hardware:

- Logical address consists of a two tuple:
  <Segment-number, offset>
- Segment table – maps two dimensional physical addresses; each table entry has:
- Base: contains the starting physical address where the segments reside in memory.
- Limit: specifies the length of the segment.
- Segment-table base register (STBR): points to the segment table's location in memory.
- Segment-table length register (STLR): Indicates number of segments used by a program;
  Segment number = s' is legal, if s < STLR.
- Relocation
  - dynamic
  - by segment table.
- Sharing.
  - shared segments
  - same segment number.

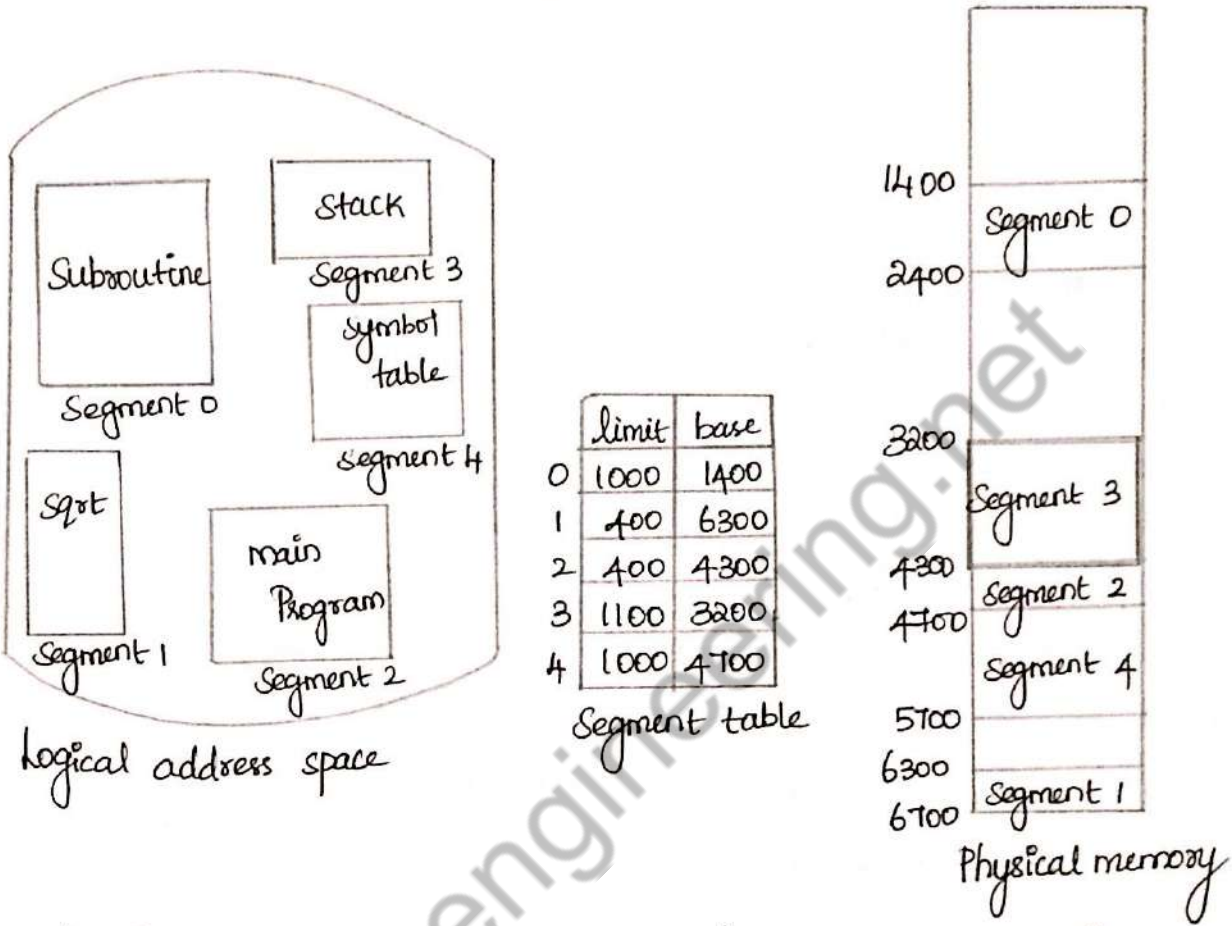
- Allocation
  - first fit / best fit
  - external fragmentation.
- Protection : With each entry in segment table associate :
  - Validation bit = 0
  - illegal segment
  - read/write/execute Privileges.
- Protection bits associated with segments; code sharing occurs at segment level.
- Since segments vary in length, memory allocation is a dynamic storage-allocation problem.
- A segmentation example is shown in the following diagram.



- Another advantage of segmentation involves the sharing of code (or) data.
- Each process has a segment table associated with it, which the dispatcher uses to define the hardware segment table when this process is given the CPU.

Segments are shared when entries in the segment tables of two different processes point to the same physical location.

Example of Segmentation:-



We have 5 segments numbered from 0 through 4. The segments are stored in physical memory. for eg., Segment 2 is 400 bytes long and begins at location 4300. Thus, a reference to byte 53 of segment 2 is mapped onto location 4300 + 53 = 4353. A reference to segment 3, byte 852, is mapped to 3200 (the base of segment 3) + 852 = 4052.

A reference to byte 1222 of segment 0 would result in a trap to the operating system, as this segment is only 1000 bytes long.

(7) Segmentation with Paging :

✓ The IBM OS/2.32 bit Version is an operating System running on top of the Intel 386 architecture. The 386 uses segmentation with paging for memory management. The maximum number of segments per process is 16 KB, and each segment can be as large as 4 gigabytes.

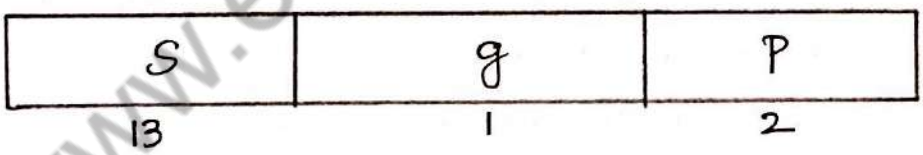
✓ The local address space of a process is divided into two partitions.

⇒ The first partition consists of up to 8 KB segments that are private to that process.

⇒ The second partition consists of up to 8 KB segments that are shared among all the processes.

Information about the first partition is kept in the local descriptor table. (LDT), information about the second Partition is kept in the global descriptor table (GDT).

Each entry in the LDT and GDT consists of 8 bytes, with detailed information about a particular segment including the base location and length of the segment. The logical address is a pair (selector, offset) where the selector is a 16-bit number:

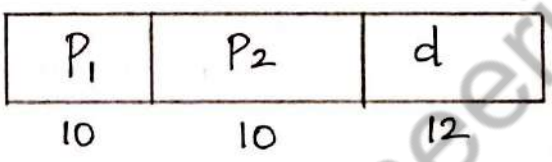


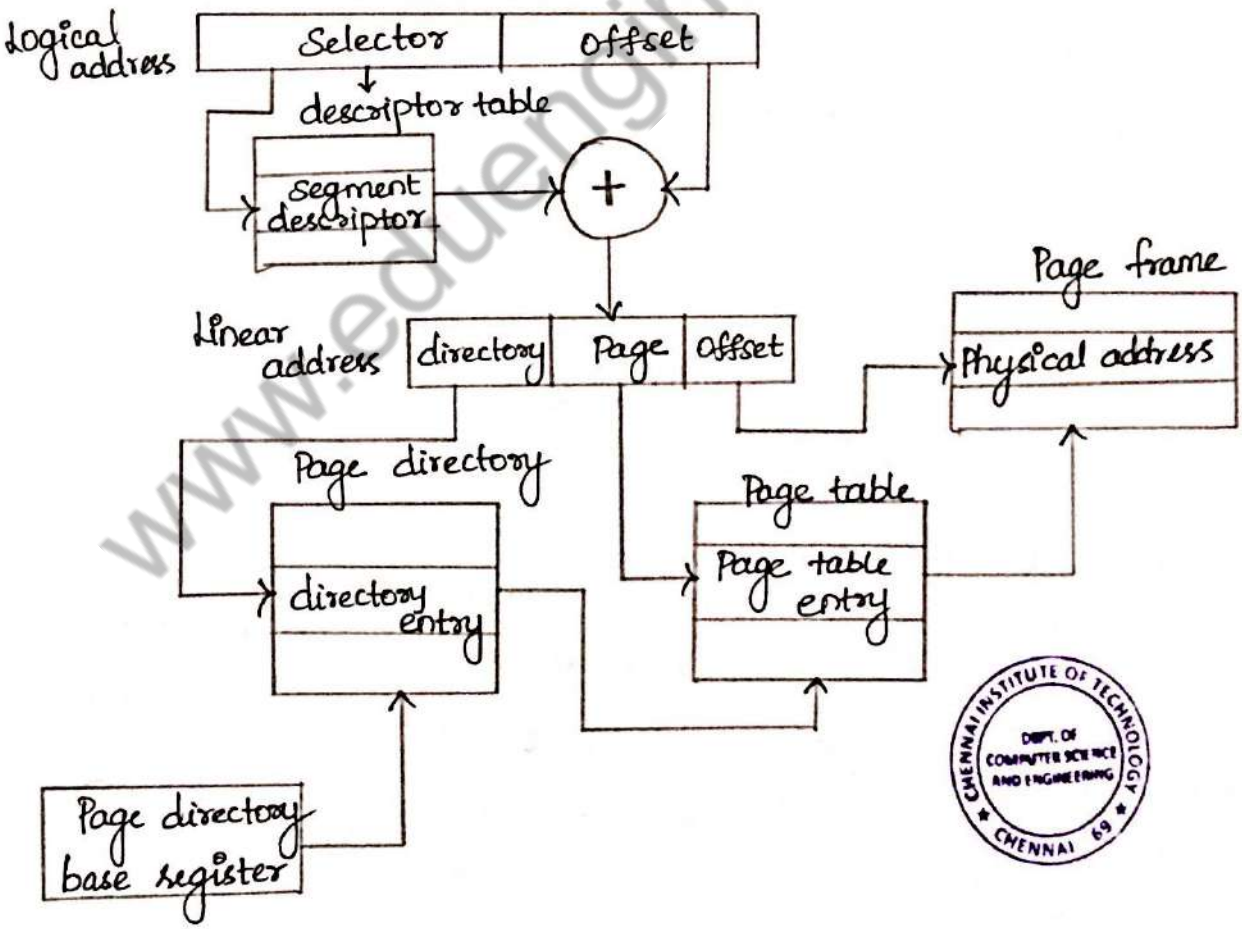
Where s designates the segment number, g indicates whether the segment is in the GDT (or) LDT, and p deals with protection. The offset is a 32-bit number specifying the location of the byte within the segment in Question.

⇒ The base and limit information about the segment in question are used to generate a linear-address.

⇒ first, the limit is used to check for address validity. If the address is not valid, a memory fault is generated, resulting in a trap to the operating system. If it is valid, then the value of the offset is added to the value of the base, resulting in a 32-bit linear address. This address is then translated into a physical address.

⇒ The linear address is divided into a page number consisting of 20 bits, and a page offset consisting of 12 bits. Since we page the page table, the page number is further divided into a page table pointer. The logical address is as follows.





* To improve the efficiency of Physical memory use.

Intel 386 page tables can be swapped to disk. In this case, an invalid bit is used in the page directory entry to indicate whether the table to which the entry is pointing is in memory or on disk.

* If the table is on disk, the operating system can use the other 31 bits to specify the disk location of the table; the table then can be brought into memory on demand.

(No) Virtual Memory :

Segmentation Advantages :-

1) Segmentation eliminates fragmentation
2) It provides virtual memory.
3) Allows dynamic segment growth.
4) Segmentation assists dynamic linking.
5) Segmentation is visible.

Disadvantages :-

1) Maximum size of a segment is limited by the size of main memory.
2) Difficulty to manage variable size segments on Secondary storage.

Advantages of paging :

1) Paging eliminates fragmentation.
2) Support higher degree of multiprogramming.
3) Paging increases memory and processor utilization.
4) compaction overhead required for the relocatable Partition scheme is also eliminated.

Disadvantages of paging:

1) Page address mapping hardware usually increases the cost of the computer.
2) Memory must be used to store the various tables like page table, memory map table etc.,

Advantages of Segmentation with paging:

Combines all advantages of paging & segmentation.

Disadvantages:

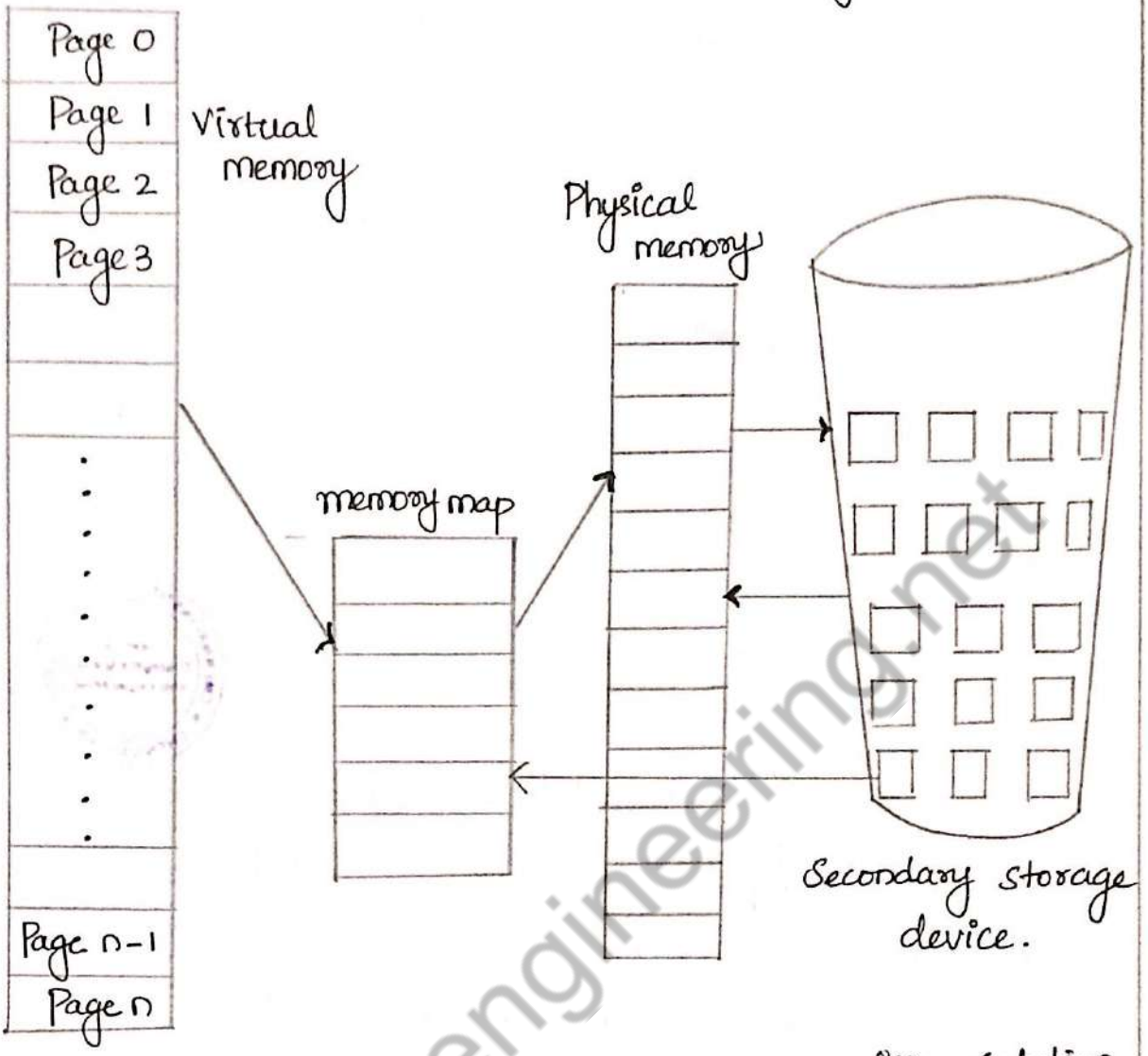
1) It increases hardware cost.
2) It increases processor overheads.
3) Dangers of thrashing.

(10) Virtual Memory :- (Background)

In many computers, Programmers often realize that some of their large Programs cannot fit in main memory for execution. Even if there is enough main memory for one Program, the main memory may be shared with other users, causing any one Program to execute. The usual solution is to introduce management schemes that intelligently allocate portions of memory to users as necessary for the efficient running of their programs. The use of virtual memory is to achieve this goal.

* Virtual memory allows execution of partially loaded Process. The ability to execute a partially loaded Process is also advantageous from the operating system point of view.

Virtual memory with other memory:



The fig., shows the virtual memory with relation to physical memory.

In fact, an examination of real programs shows us that, in many cases, the entire program is not needed.

Following are the situations, when entire program is not required to load fully.

1) User written error handling routines are used only when an error occurs in the data (or) computation.

2) Certain options and features of a program may be used rarely.

3) Many data structures are allocated memory space address space even though only a small amount of the table is actually used.

4) Many routines are commonly used at mutually exclusive times during a run.

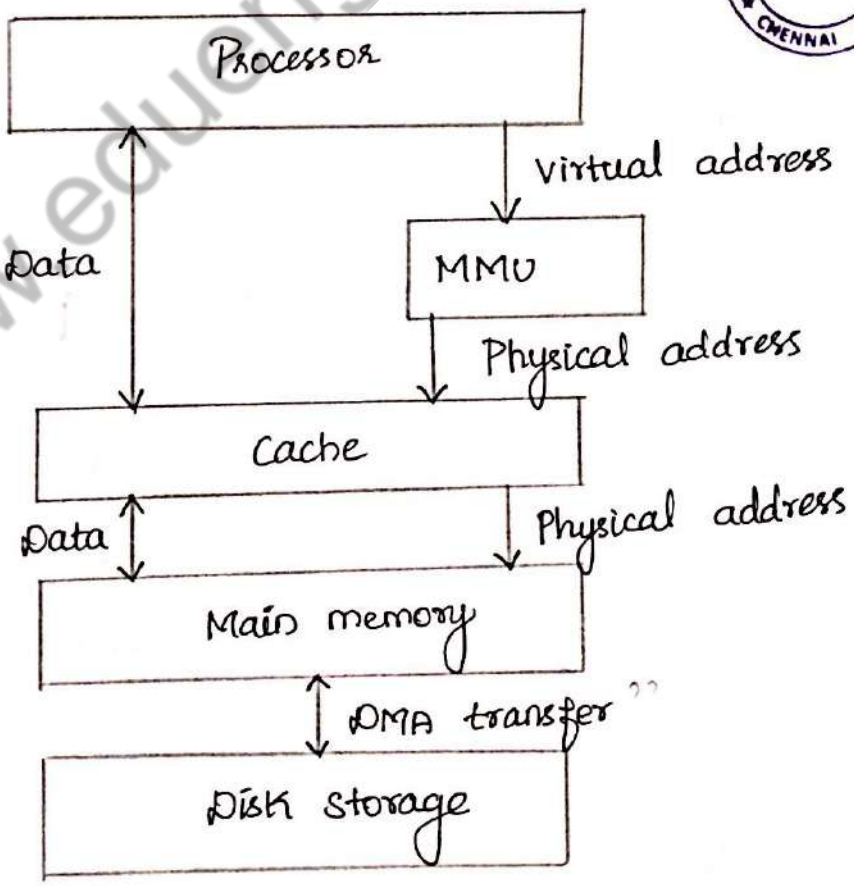
The ability to execute a program that is only Partially in memory would confer many benefits.

1) Less number of I/o would be needed to load (or) swap each user program into memory.

2) A program would no longer be constrained by the amount of physical memory that is available.

3) Each user program could take less physical memory, more programs could be run at the same time, with a corresponding increase in cpu Utilization and throughput.

Virtual memory organization



Virtual memory makes the task of Programming much easier. Virtual memory is commonly implemented by demand paging.

Virtual memory mechanism bridges the size and speed gaps between the main memory and secondary storage and is usually implemented in part by software technique.

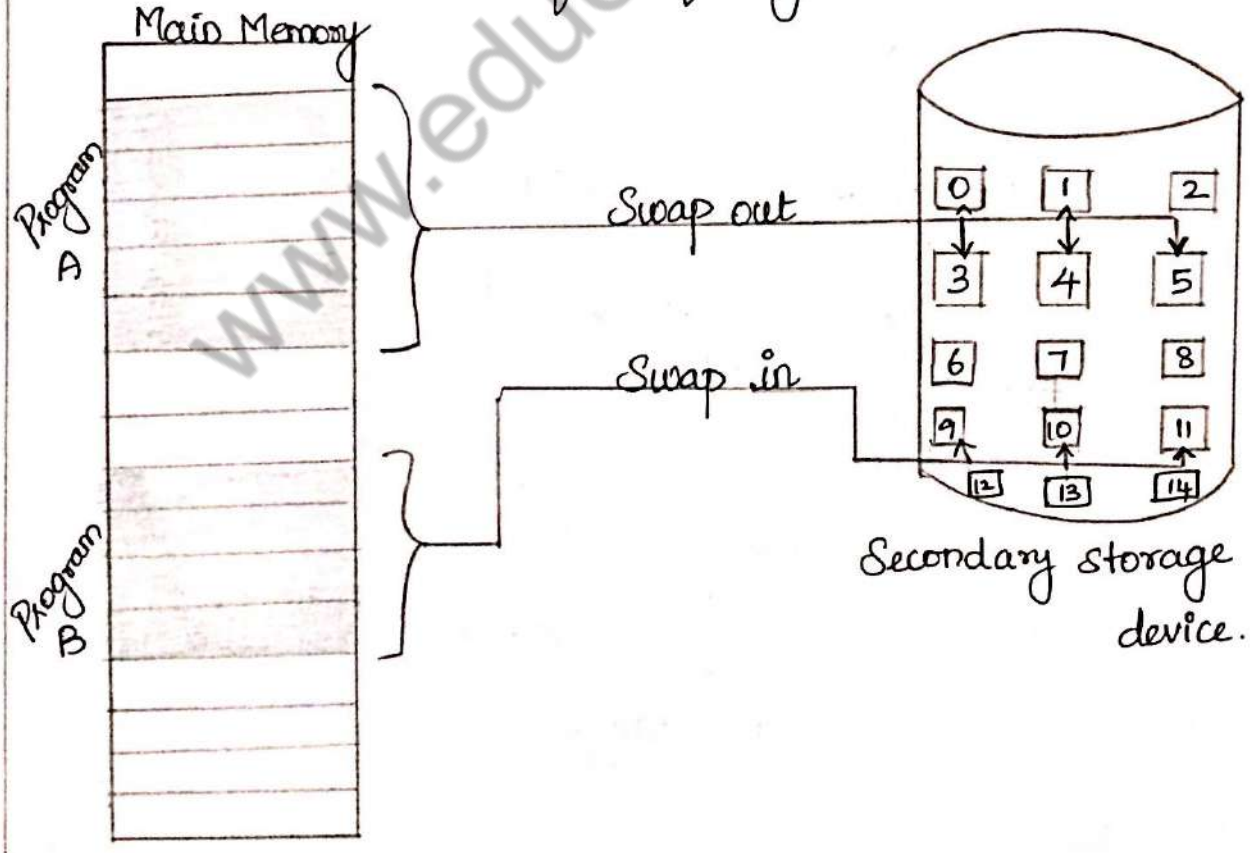
(12) Demand Paging :

Demand paging = Paging system + swapping.

* With demand paging, a page is brought into main memory only when a reference is made to a location.

* Lazy swapper concept is used in used in demand paging. A lazy swapper never swaps a page into a memory unless that page will be needed.

Transfer of page.



* Demand paging combines the features of simple paging and overlaying to implement virtual memory.

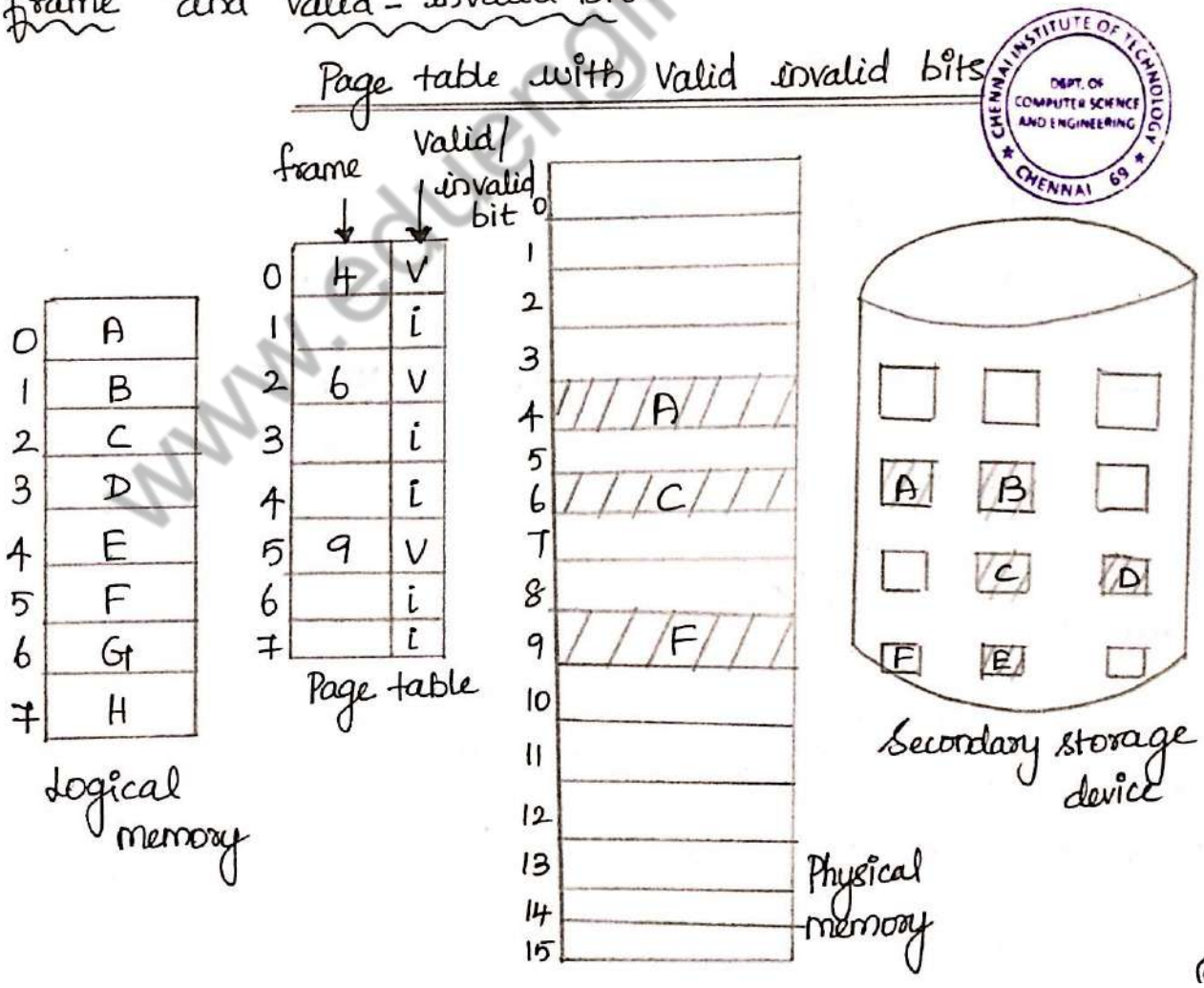
* In this, each page of a program is stored contiguously in the paging swap space on a secondary storage.

* As locations in pages are referenced, the pages are copied into memory page frames.

* Once the page is in the memory, it is accessed as in simple paging. The previous diagram shows the transfer of page from secondary storage to main memory.

* Some form of hardware support is required to distinguish between those pages that are in memory and those pages that are on the disk. The valid/invalid bit scheme can be used for this purpose. Each entry in the page table has at a minimum two fields. Page frame and Valid - invalid bit.

Page table with Valid invalid bits



* When a valid bit is set, then the associated page is legal and present in the memory. During translation address is generated, the memory management hardware extracts the page number from the address, and the appropriate entry in the page table is accessed.

The valid-invalid bit is checked. If the page is not in memory, a page fault occurs, transferring control to the page fault routine in the Operating System.

* When the running process experiences a page fault, it must be suspended until the missing page is brought into main memory.

* The disk address of the faulted page is usually provided in the File Map Table (FMT). This table is parallel to the page map table. Thus, when Processing a page number provided by the mapping hardware of index the FMT and to obtain the related disk address.

* Steps in Handling a page fault
* Hardware support
* Software Algorithm
* Performance of demand paging.

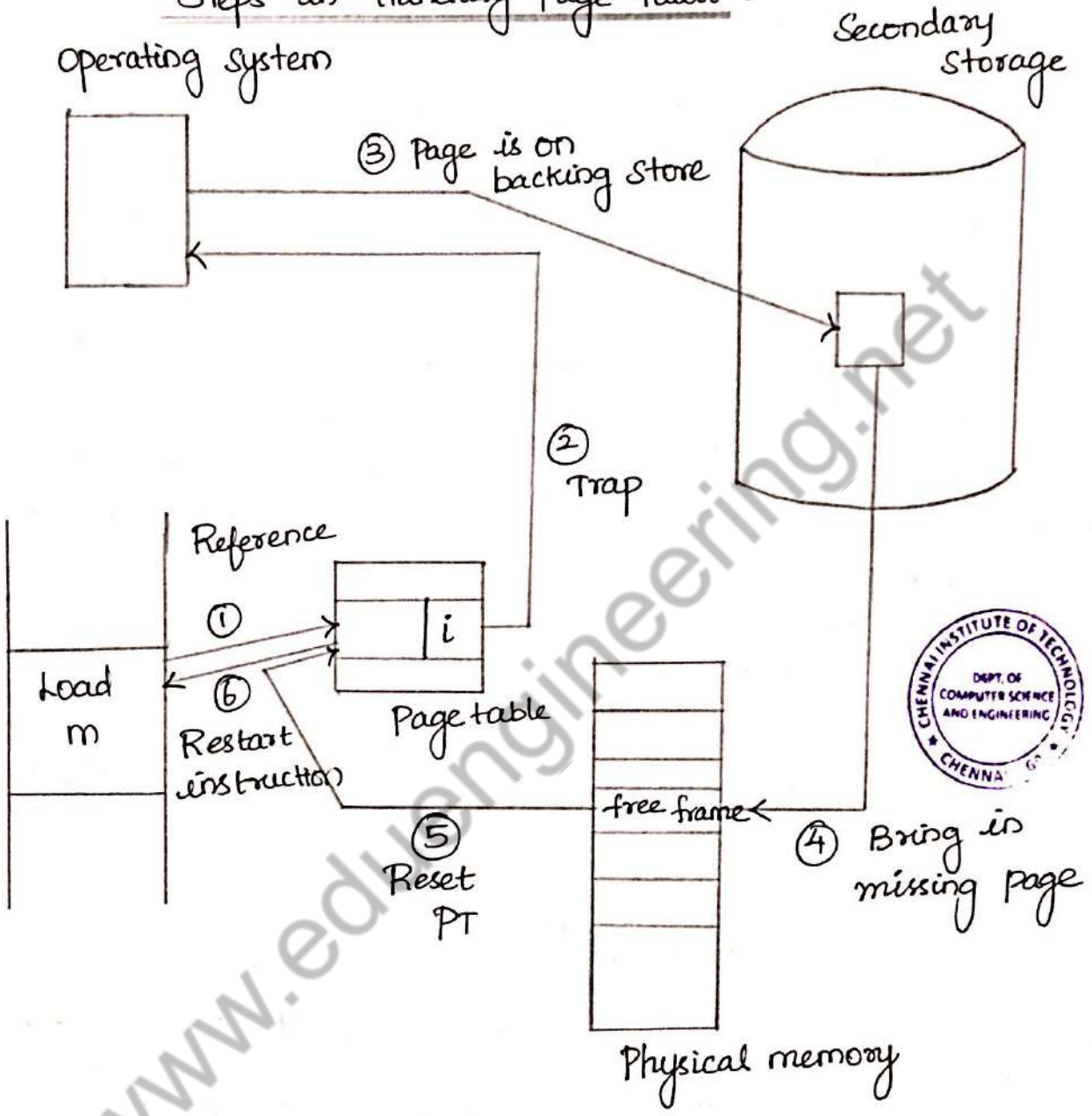
* Steps in Handling a page fault : [N|D-19]

1) Check an internal table for this process, to determine whether the reference was a valid or invalid memory access.

2) If the reference was invalid, terminate the process. If it was valid, but not yet brought in that page, now page it in.

* find a free frame.
* Schedule a disk operation to read the desired page into the newly allocated frame.

Steps in Handling Page fault :



* Schedule a disk operation to read the desired Page into the newly allocated frame.
* When disk read is complete, modify the internal table kept with the Process and the page table to indicate that the page is now in memory. Restart the instruction that was interrupted by the illegal address trap. The Process can now access the page as though it had always been in memory.

Hardware support :-

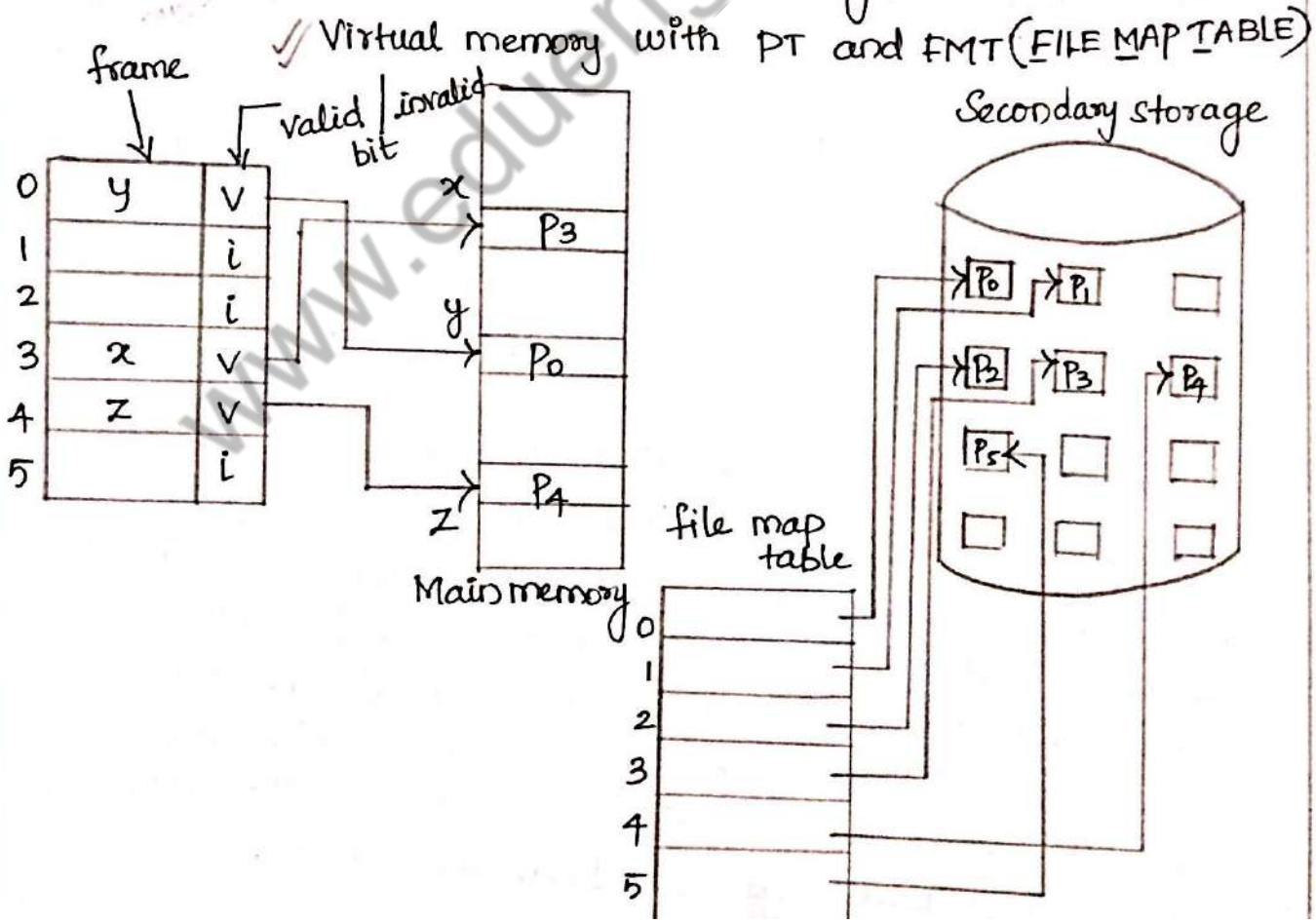
The hardware to support demand paging is the same as the hardware for paging and swapping.

Page table : It has the ability to mark an entry invalid through a valid - invalid bit.

Secondary memory : This memory holds those pages that are not present in main memory. It is usually high speed disk.

Software Algorithm :

Demand Page memory management provides tremendous flexibility for the operating system. It must interact with information management to access and store copies of the jobs address space on secondary storage. file map table is used to store the information regarding a file. FMT is not used by the hardware. It is usually accessed by software. A possible format and use of the file map table is shown in fig.,



Performance of demand Paging :-

Demand paging can have a significant effect on the performance of a computer system. Let us calculate effective access time for a demand paged memory. Let p be the probability of a page fault ($0 \leq p \leq 1$).

Effective access time = $(1-p) \times ma + p \times$ Page fault time

where., $p \rightarrow$ Page fault ; ma = Memory access time.

Effective access time is directly proportional to the Page fault rate.

It is important to keep the page-fault rate low in a demand paging system. Otherwise, the effective access time increases, slowing process execution dramatically.

Advantages of demand Paging :

1) Large virtual memory.
2) More efficient use of memory.
3) Unconstrained multiprogramming. There is no limit on degree of multiprogramming.

Disadvantages of demand paging :

1) Number of tables and amount of Processor overhead for handling page interrupts are greater than in the case of the simple paged management techniques.
2) Due to the lack of an explicit constraints on a job, address space size.

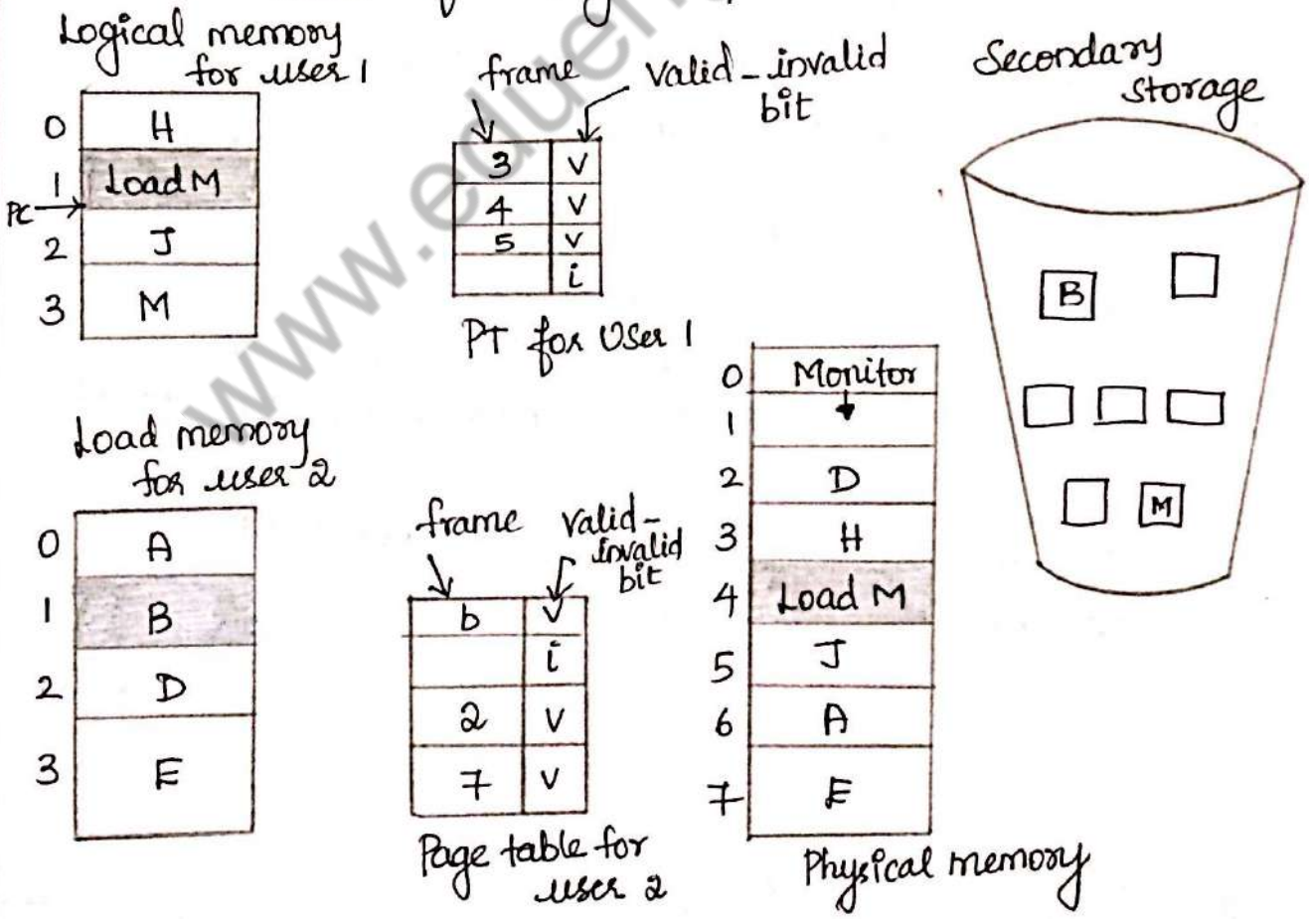
(15) Page Replacement Algorithms : [Unit-V]

* Page replacement policy deals with the selection of a page in memory to be replaced when a new page must be brought in. While a user process is executing, a page fault occurs.

* The hardware traps to the operating system, which checks its internal tables to see that this page fault is a genuine one rather than an illegal memory access.

* The operating system determines where the desired Page is residing on the disk, but then finds that there are no free frames, on the free frame list. All memory is in use. When all the frames in main memory are occupied and it is necessary to bring in a new page to satisfy a page fault, replacement Policy is concerned with selecting a page currently in memory to be replaced.

Need for Page replacement.



* All the Policies have as their objective that the page that is removed should be the page least likely to be referenced in the near future.

* Working of Page replacement Algorithm :-

1) Find the location of the desired page on the disk.

2) Find a free frame

   a) If there is a free frame, use it.

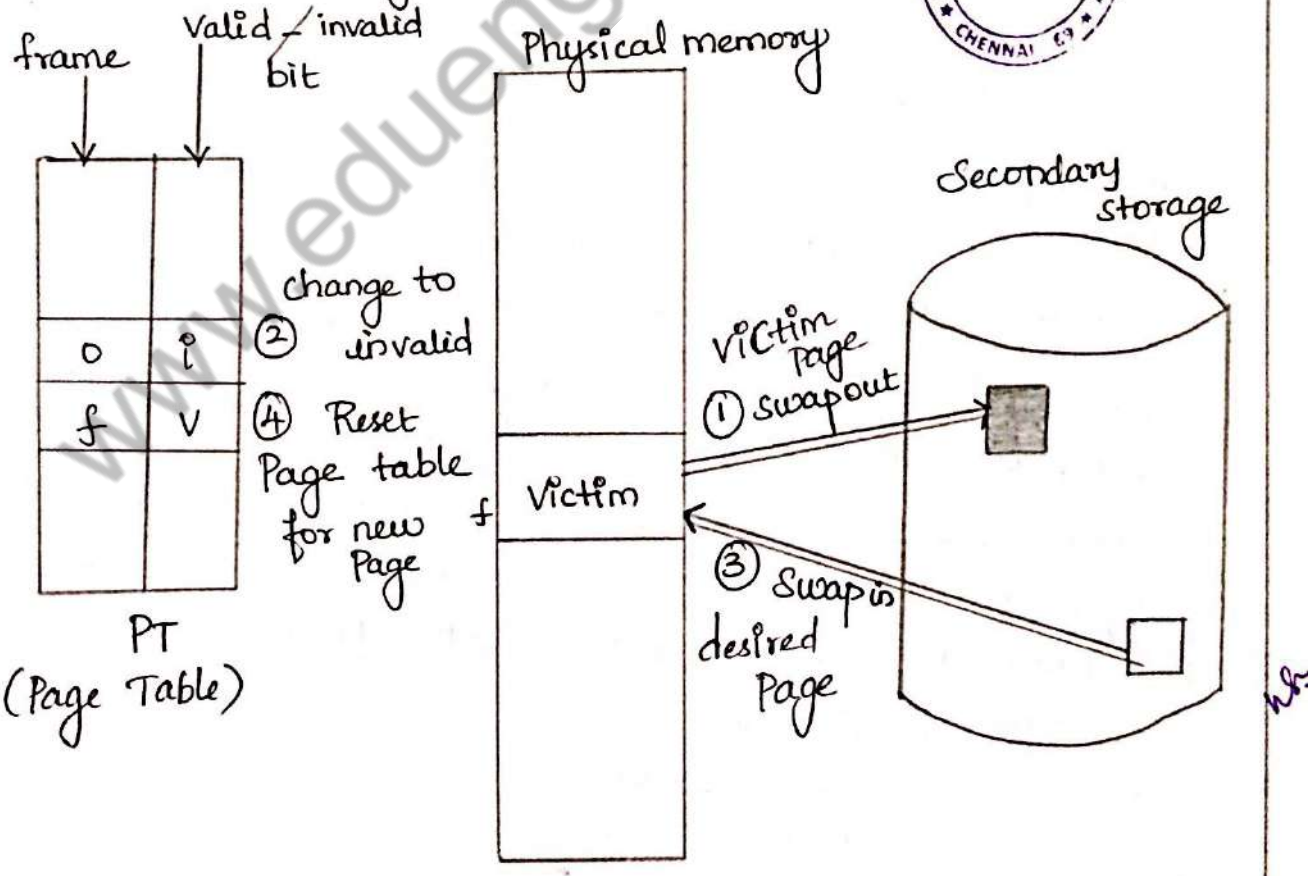
   b) If there is no free frame, use a page replacement algorithm to select a victim frame.

   c) Write a Victim page to the disk, change the Page and frame tables accordingly.

3) Read the desired page into the free frame, Change the page and frame tables.

4) Restart the user process.

Page replacement.



Memory Reference String :-

The string of memory references is called as reference string. A succession of memory references made by a program executing on a computer with 1 MB of memory is given below in hex notation.

....., 14489, 1448B, 14494, 14496, A1F8, 14497, 14499, 2638E, 1449A, ...

When analyzing page replacement algorithms, we are interested only in the pages being referenced.

Replacement Algorithms.

1) First In First Out (FIFO)
2) Least Recently Used (LRU)
3) Optimal

1) First In First Out (FIFO) Page Replacement :

⇒ It is one of the simplest method. It selects the page that has been in memory the longest. When a page must be replaced, the oldest page is chosen.

⇒ To implement, the memory manager must keep track of the relative order of the loading of pages into the memory.

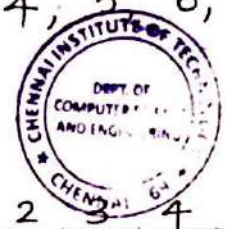
⇒ When a page is brought into memory, it is inserted at the tail of the Queue.

⇒ It is easy to understand and program. FIFO is not the 1st choice of the operating system designess for page replacement algorithm, since its performance is not always good.

Eg: Reference string:

0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, 4, 5, 6, 7

Page frame : 3.



Solution :

frame	0	1	2	3	0	1	2	3	0	1	2	3	4	5	6	7
0	0	0	0	3	3	3	2	2	2	1	1	1	4	4	4	7
1		1	1	1	0	0	0	3	3	3	2	2	2	5	5	5
2			2	2	2	1	1	1	0	0	0	3	3	3	6	6
Page fault	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

∴ No. of Page fault is : 16.

If Page frame : 4

Frame	0	1	2	3	0	1	2	3	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
1		1	1	1	1	1	1	1	1	1	1	1	1	5	5	5
2			2	2	2	2	2	2	2	2	2	2	2	2	6	6
3				3	3	3	3	3	3	3	3	3	3	3	3	7
Page fault	*	*	*	*									*	*	*	*

∴ No. of Page fault is : 8

Belady's Anomaly :- As the page fault rate may increase as the number of allocated frames increases. FIFO page replacement algorithm may space this problem. (for some page replacement algorithm)

2) LRU Pagereplacement :-

⇒ It replaces the page in memory that has not been referenced for the longest time.

⇒ It performs better than FIFO.

⇒ It do not suffer from Belady's Anomaly.

Let us apply LRU algorithm to the reference string with frame 3.

0 1 2 3 0 1 2 3 0 1 2 3 4 5 6 7

Solution :

Frame	0	1	2	3	0	1	2	3	0	1	2	3	4	5	6	7
0	0	0	0	3	3	3	2	2	2	1	1	1	4	4	4	7
1		1	1	1	0	0	0	3	3	3	2	2	2	5	5	5
2			2	2	2	1	1	1	0	0	0	3	3	3	6	6
Page fault	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

∴ No. of page fault : 16

If Page frame = 4, then

frame	0	1	2	3	0	1	2	3	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
1		1	1	1	1	1	1	1	1	1	1	1	1	5	5	5
2			2	2	2	2	2	2	2	2	2	2	2	2	6	6
3				3	3	3	3	3	3	3	3	3	3	3	3	7
Page fault	*	*	*	*		Hit							*	*	*	*

∴ No. of page fault : 8

Implement using { counters / stack.

Counters :

1) Each time a page is referenced, copy the counter into the time-of-use field.

2) When a page needs to be replaced, replace the Page with the smallest counter value.

Stack : 1) Whenever a page is referenced, remove the Page from the stack and put it on top of the stack.

2) When a page need to be replaced, replace the Page that is at the bottom of the stack (LRU page)

LRU Approximation :

* A reference bit is associated with each memory block and this bit is automatically set to 1 by the hardware whenever that page is referenced. The single reference bit per clock can be used to approximate LRU removal.

* The Page removal software Periodically resets the reference bit to 0, while the execution of the users job causes some reference bits to be set to 1. If the reference bit is 0, then that page has not been referenced since the last time the reference bit was reset to 0.

3) Optimal Page Replacement :

The optimal policy selects for replacement that page for which the time to the next reference is the longest. An optimal page replacement algorithm has the lowest page fault rate of all algorithms and will never suffer from Belady's anomaly. This algorithm is impossible to implement because it would require the operating system to have perfect knowledge of future events.

Eg., Ref. frame : 3

Ref string : 0 1 2 3 0 1 2 3 0 1 2 3 4 5 6 7

→

Frame	0	1	2	3	0	1	2	3	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0	0	1	1	1	4	4	4	7
1		1	1	1	1	1	2	2	2	2	2	2	2	5	5	5
2			②	3	3	3	3	3	3	3	3	3	3	3	6	6
Page fault	*	*	*	*			*			*			*	*	*	*

∴ Number of Page fault : 10.

With Page Frame = 4 for same reference string.

frame	0	1	2	3	0	1	2	3	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
1		1	1	1	1	1	1	1	1	1	1	1	1	5	5	5
2			2	2	2	2	2	2	2	2	2	2	2	2	6	6
3				3	3	3	3	3	3	3	3	3	3	3	3	7
Page fault	*	*	*	*									*	*	*	*

No. of Page fault : 8

4) Counting based Page replacement :-

⇒ It depends on the counter of the no. of references.

(LFU) Least Frequently Used.

⇒ It selects a page for replacement if the Page has not been used often in the past. LFU tends to react slowly to change in locality. LFU uses frequency counts from the beginning of the Page reference stream.

Eg; Ref. String :- 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, 4, 5, 6, 7.
Ref. frame :- 3.

Frame	0	1	2	3	0	1	2	3	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0	0	0	0	3	3	3	3	3
1		1	1	1	1	①	1	3	③	1	1	1	1	1	1	1
2			②	③	3	3	2	2	2	2	2	2	4	5	6	7
Page fault	*	*	*	*			*	*		*		*	*	*	*	*

∴ No. of Page fault = 12.

12) Allocation of Frames :

The allocation policy in a virtual memory system governs the operating system decisions regarding the amount of real memory to be allocated to each active Process.

In Paging System,

⇒ If more real pages are allocated to a process, it reduces page fault frequency and improved turnaround time.

⇒ If too few pages are allocated to a process, its page fault frequency and turnaround time may decreases to unacceptable levels.

The minimum no. of frames per process is defined by the architecture, the maximum number is defined by the amount of available physical memory.

Allocation Schemes

- Equal Allocation
- Proportional Allocation

Equal Allocation :

If there are 'n' processes and 'm' frames, then allocate m/n frames to each process.

Eg ; If there are '5' processes and '100' frames, give each process '20' frames.

Proportional Allocation : Allocate according to size of process.

Let S_i be the size of Process i.

Let m be the total no. of frames.

then., $S = \sum S_i$

$$a_i = S_i / S * m$$

where a_i is the no. of frames allocated to process i

With multiple Processes competing for frames

Page replacement

Local Replacement Global Replacement.

Global Replacement : Each process selects a replacement frame from the set of all frames; One Process can take a frame from another.

Local Replacement : Each process selects from only its own set of allocated frames.

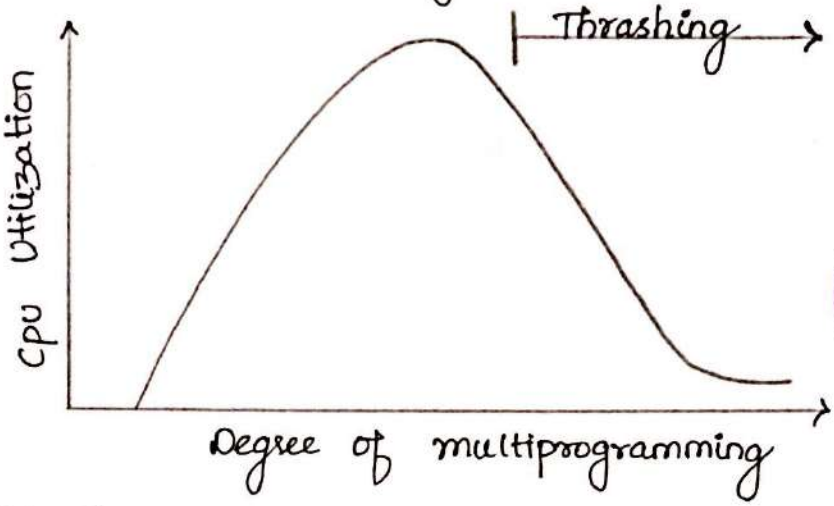
(15) Thrashing :

High paging activity is called thrashing. If the number of frames allocated to a low priority process falls below the minimum number required by the computer architecture, Process must be suspend for execution.

The phenomenon of excessively moving pages back and forth between memory and secondary storage is called thrashing.

It consumes lot of computer energy but accomplishes very little useful results. A process is thrashing if it is spending more time paging than executing. Thrashing results in several performance problems.

The operating system monitors cpu utilization. If cpu utilization is too low, we increase the degree of multiprogramming by introducing new process to the system. As processes wait for the paging device, cpu utilization decreases. The cpu scheduler sees the decreasing cpu utilization and increases the degree of multiprogramming as a result.



At this point, to increase CPU utilization and stop thrashing, we must decrease the degree of multiprogramming.

We can limit the effect of thrashing by using a local replacement algorithm. To prevent thrashing, we must provide a process as many frames as it needs. But how to calculate the required frame. Solution to this is by using working set theory.

Locality of Reference :

The locality model states that, as a process executes, it moves from locality to locality. A locality is a set of pages that are actively used together.

Locality

Spatial locality Temporal locality.

Spatial locality : It refers to the fact that if a memory location is accessed, it is likely that a location near it will be accessed in the next instruction.

Temporal locality : If a memory location has been referenced, there is good chance it will be referenced again in a short period of time.

To limit thrashing, we must use a local replacement algorithm. To prevent thrashing, there are two methods namely.,

* Working set strategy
* Page fault frequency.

* Working set Strategy :

* It is based on the assumption of the model of locality.
* Locality is defined as the set of pages actively used together.
* Working set is the set of pages in the most recent, Δ page references.
* Δ is the working set window.
  - Δ if Δ too small, it will not encompass entire locality.
  - Δ if Δ too large, it will encompass several localities.
  - Δ if Δ = ΔΔ it will encompass entire program.

$D = \Delta WSS_i$.

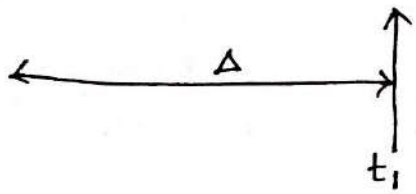
Δ where WSS_i is the working set size for Process i.

Δ D is the total demand of frames.

* if D > m then, Thrashing will occur.

Page reference table :- [Working set model]

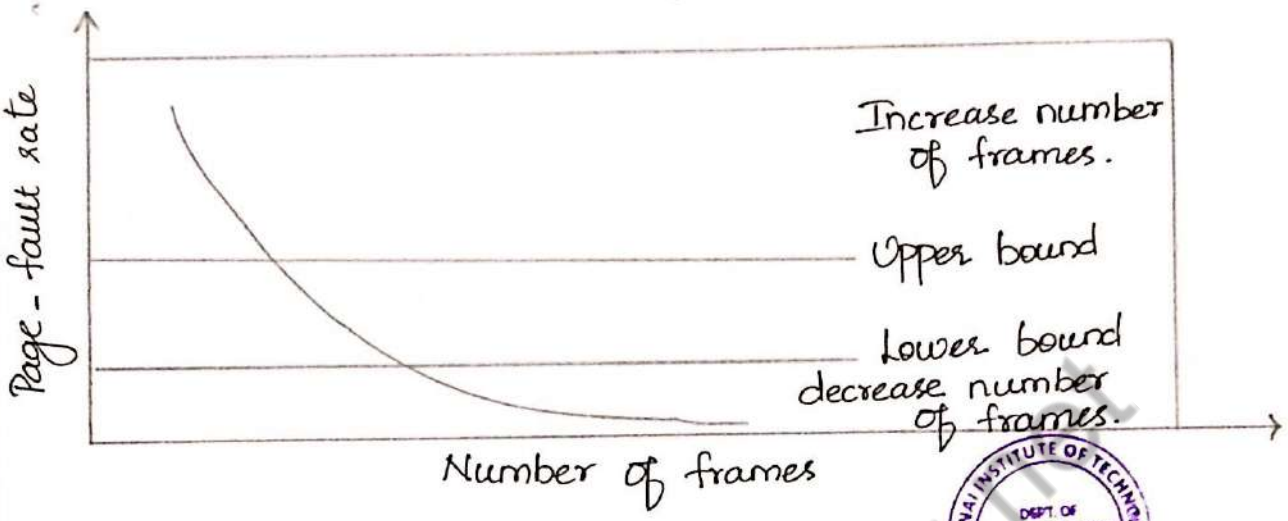
. . . 2 6 1 5 7 7 7 7 5 1 6 2 3 4 1 2 3 4 4 4 3 4 4 1 3 2 3 4 4 4 3 4 4 ...



$WS(t_1) = \{1, 2, 5, 6, 7\}$

$WS(t_2) = \{3, 4\}$

* Page fault frequency scheme:
  → If actual rate too low, process loses frame.
  → If actual rate too high, Process gains frame.



(16) Allocating Kernel Memory:

When a process running in user mode requests additional memory, pages are allocated from the list of free Page frames maintained by the kernel.

This list is typically populated using a page replacement algorithm such as those most likely contains free pages scattered throughout physical memory. ~~Internal fragmentation with~~ If a user process requests a single byte of memory, internal fragmentation will result, as the process will be granted an entire Page frame.

Kernel memory is often allocated from a free-memory pool different from the list used to satisfy Ordinary user-mode processes. There are two primary reasons for this:

(1) The kernel requests memory for data structures of varying sizes, some of which are less than a page in size.

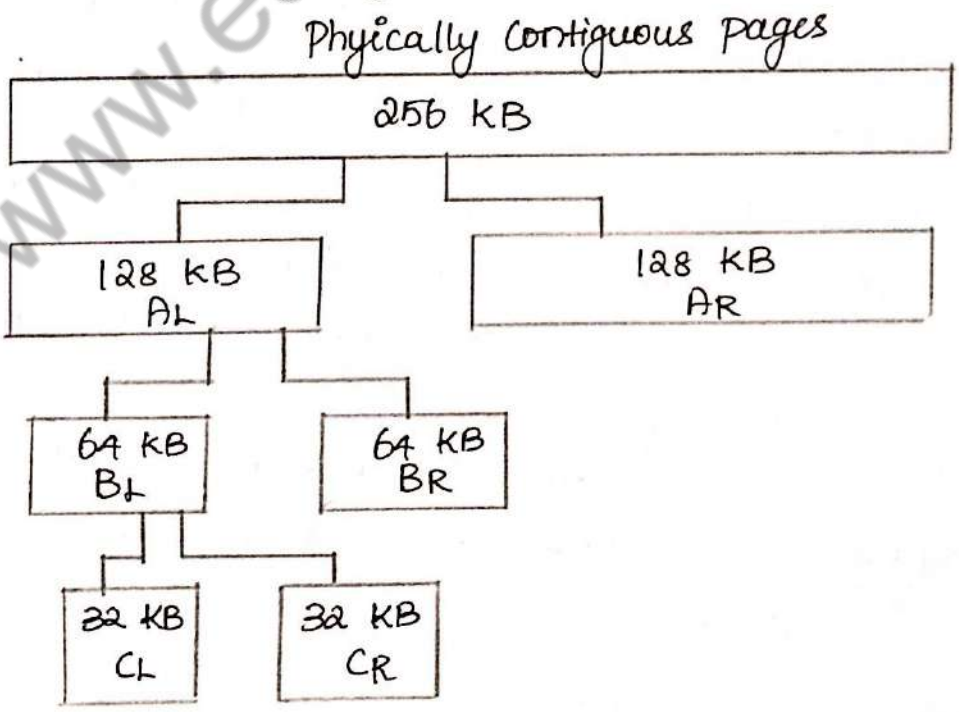
As a result, the Kernel must use memory conservatively and attempt to minimize waste due to fragmentation. This is especially important because many operating systems do not subject Kernel code or data to the paging system.

(2) page allocated to User-mode processes do not necessarily have to be in contiguous physical memory. However certain hardware devices interact directly with physical memory - without the benefit of a virtual memory interface - and consequently may require memory residing in physically contiguous pages.

Buddy system:

The buddy system allocates memory from a fixed-size consisting of physically contiguous pages. Memory is allocated from this segment using a power-of-2 allocator, which satisfies requests in units sized as a power of 2 (4 KB, 8KB, 16KB, and so forth). A request in units not appropriately sized is rounded up to the next highest power of 2. for eg-, a request for 11 KB is satisfied with a 16k segment.

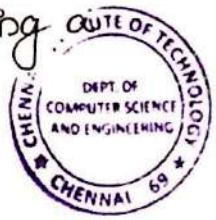
Buddy system allocation.



Windows : Windows implements virtual memory using demand paging with clustering. clustering handles Page faults by bringing in not only the faulting Page but also several pages following the faulting Page. When a process is first created, it is assigned a working set minimum and maximum. The working-set minimum is the minimum number of Pages the process is guaranteed to have in memory.

If sufficient memory is available, process may be assigned as many pages as its working-set maximum. The virtual memory manager maintains a list of free page frames. Associated with this list is a threshold value that is used to indicate whether sufficient free memory is available.

If a page fault occurs for a process that is below its working-set maximum, the virtual memory manager allocates a page from this list of free pages. If a process that is at its working set maximum incurs a page fault, it must select a page for replacement using local LRU page-replacement policy.



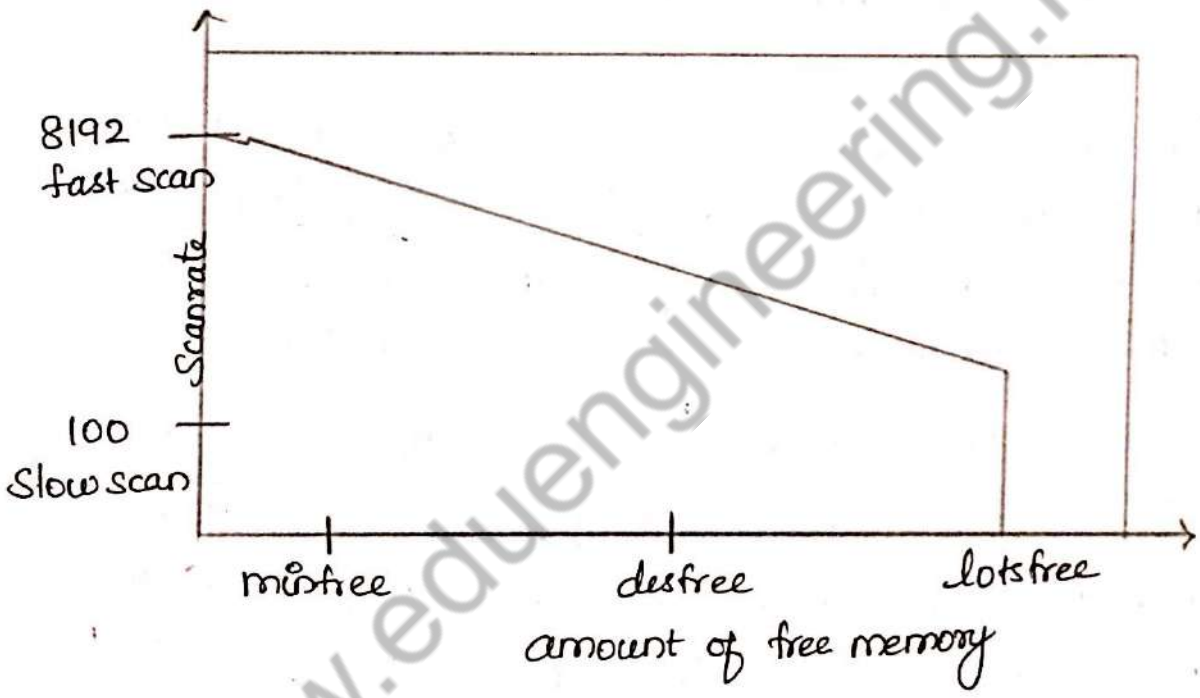
Solaris :-

In solaris, when a thread incurs a page fault, the kernel assigns a page to the faulting thread from the list of free pages it maintains.

Therefore, it is imperative that the kernel keep a sufficient amount of free memory available.

Associated with this list of free pages is a Parameter - lotsfree - that represents a threshold to begin Paging. The lotsfree parameter is typically set to 1/64 the size of the physical memory. Four times per second, the kernel checks whether the amount of free memory is less than lotsfree. If the number of free pages falls below lotsfree, a process known as a pageout starts up.

Solaris page scanner :-



K. 29/12/18
Prepared By

10/01/19
Verified By

22/1/19
Approved By

CHENNAI INSTITUTE OF TECHNOLOGY
DEPT. OF COMPUTER SCIENCE AND ENGINEERING
CHENNAI 69

EDU ENGINEERING

PIONEER OF ENGINEERING NOTES

TAMIL NADU'S BEST EDTECH PLATFORM FOR ENGINEERING

CONNECT WITH US

WEBSITE: www.eduengineering.net

TELEGRAM: @eduengineering

INSTAGRAM: @eduengineering

- **Regular Updates for all Semesters**
- **All Department** Notes AVAILABLE
- **Handwritten** Notes AVAILABLE
- **Past Year** Question Papers AVAILABLE
- **Subject wise** Question Banks AVAILABLE
- **Important Questions** for Semesters AVAILABLE
- Various **Author Books** AVAILABLE