Unit 5: Memory Management

Introduction :-

- In a computer system, there may be multiple processes executing at the same time & every process that needs to execute, requires a certain amount of memory.
- Memory management is one of the important function of operating system which helps in allocating the main memory space to the processes and their data at the time of their execution.
  The main task of memory management is efficient memory utilization.
- Memory management is the functionality of an operating system which handles or manages primary memory & disk during execution.
- To improve the utilization of the CPU and the speed of the computer's response to its users, we must keep several processes in memory i.e, we must share memory. Thus, multi-programming creates need of memory management.

* Basic of Memory Management :

- Memory Management in operating System is important because of it directly affects the execution time of a process. The execution time of a process depends on the availability of data in the main memory.
- Therefore, an operating System must perform the memory management in such a manner that the essential data is always present in the main memory.
- An effective memory management System ensures accuracy, availability and consistency of the data imported from the Secondary memory to the main memory.

* Functions of Memory Management :

- Memory management specifies how to utilize the memory in an efficient manner. Memory management is the functionality of an operating System which handles or manages primary memory (RAM) & moves process back & forth between main memory and disk during execution.
- If we want to archieve the efficient memory utilization we have to follow

Page No.
Date

Following Storage management function:

1) Process Isolation - process Isolation means controlling of one process interacts with the data and memory of other process.

2) Tracking of Memory location - Memory management keeps track of each & every memory location, regardless of either it is allocated to some process or it is free.

3) Automatic Allocation & Management - Memory should be allocated dynamically based on the priorities of the process.

4) Long term Storage - Long term storage of process will reduce the memory utilization.

5) Support of Modular Programming - A program is divided into number of modules, if the memory is not sufficient for the entire program, we can load at least some of the modules instead of the entire program.

6) Protection & Access Control - Do not apply the protection mechanism & access control to all the processes, better to apply to the important applications only. It will save the execution time.

7) Keeping status of Main Memory Locations - Memory management keeps track of the status of each memory location, whether it is allocated or free:

* function of memory managment

- Memory management Specifies how to utilize the memory in en efficient manner. Memory management is the functionality of an oprating System which handles or manages primary memory (RAM) and moves processes back and forth between main memory and disk during execution
- If we want to follow the following Storege management responsiblities / functions.

1] **Process Isolation :-**
process isolation means controlling of one process interacts with the data and memory of the other process.

2] **Tracking of memory Locations :-**
Memory management keeps track of each and every memory location, regradless of either it is allocated to some process or it is free. it checks how much memory is to be allocated to processes.

3] **Automatic Allocation and management :-**
Memory should be allocated dynamically based on the priorities of the process otherwise the process waiting will increase and it decrease the CPU utilization and memory Utilization.

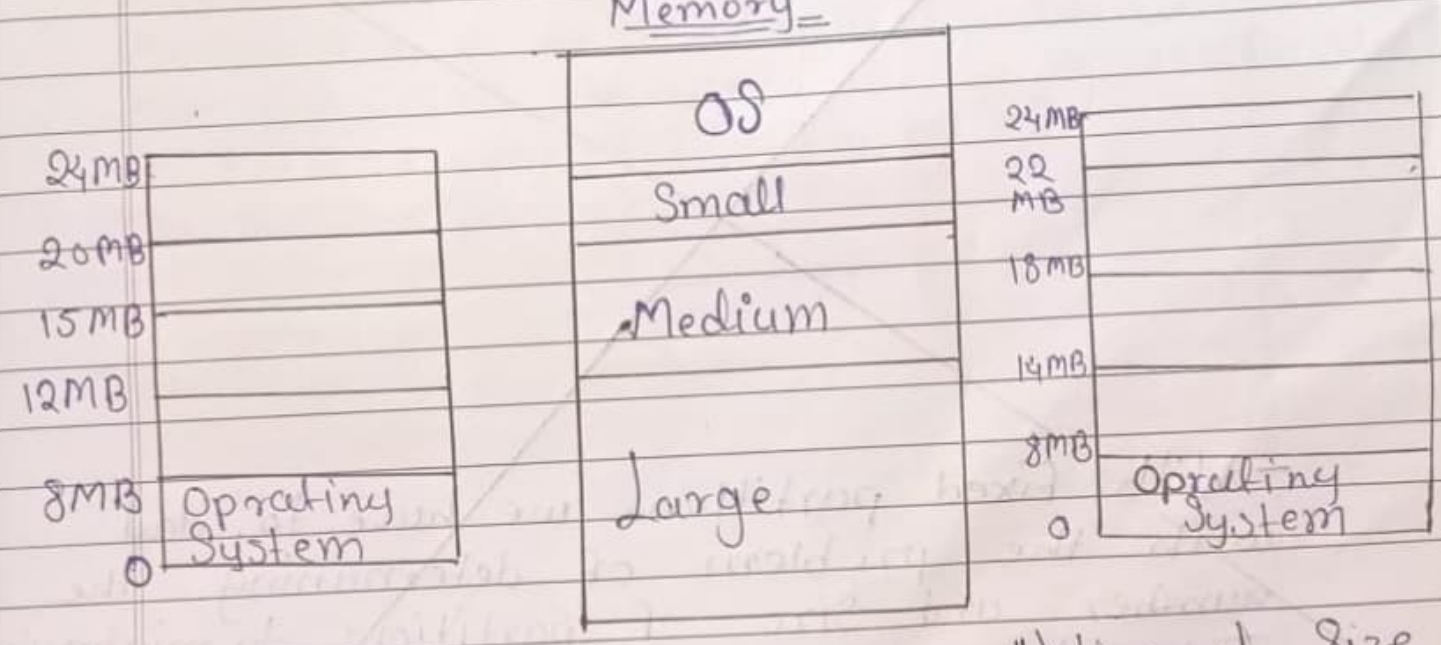
each region may have one program to be executed.
- when a region is free, a program is selected from the job queue and loaded into the free region, when it terminates the regions becomes available for another program.

* Type of Memory Partitioning :-

1) Static (fixed size) memory partitioning.

2) Dynamic (variable) memory partitioning.

1] - Static (fixed Size) memory partitionining :-



a) Equal Size partitions

b) Unequal Size partitions.

- In Static memory partitioning the memory is devided into a number of fixed sized partitions and do not chunge as the System runs.
- Each partition in static memory partitioning contains exactly one process. So the number of program to be executed. (i.e. degree of multiprogramming) depends of the number of partitions.
- there are two alternatives for fixed size memory partitioning namely, equal sized partitions and unequal size partitions.

4] Long Term Storage :-
Long term Storage of process will reduce the memory utilizations.

5] Support of modular programming :-
A program is divided into number of modules. if the memory is not Sufficient for the entire program. This will increase CPU utilization, memory utilization.

6] Protection and Access Controls :-
do not Apply the protection machanisms and access control to all the processes, better to apply to the important application only. it will Save the execution time

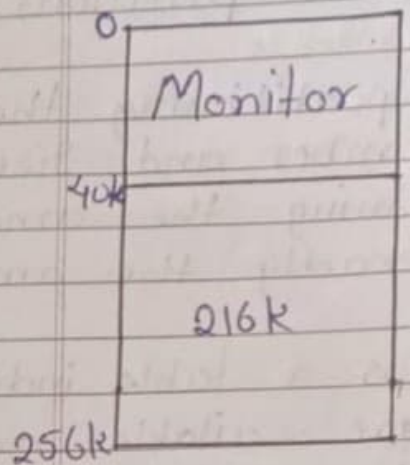
7] keeping Status of main memory location :-
memory management keep track of the Status of each memory location. whether it is allocated or free.

* Memory partitioning :-

- For multiprocessing two or more jobs Should Simaltaneously be placed in the main memory. So the memory Should partitioned into a number of Sections. This partitioning is called partitioned memory management
- In memory partitioning, memory is divided into a number of regions or partitions.

- for example :-

assume 256k memory available and a resident monitor of 40k, This situation leaves 216k for user program



Job Queue		
Job	Memory	Time
1	60k	10
2	100k	5
3	30k	20
4	70k	8
5	50k	15

2] Dynamic (Veriable memory partitioning)

- With fixed partitions we have to deal with the problem of determining the number and Size of partitions to minimaize internal and external fragmentation. If we use variable partitioning insted, then partitions Size may very dynamically.
- In ~~very~~ veriable memory partitioning the partition can vary in number and Size. In veriable memory partitioning the amount of memory allocated is exactly the amount of memory requires.
- The oprating System keeps a table indicating which part of memory are available and which are occupied, initially all memory is available for user program and is considered one large block of available memory, a hole.
- when Job arrived and need memory, we Search for a hole enough for this Job. If we fine one, we allocate only as much as is needed, ~~keeping~~ keeping the rest available to Satisfy future requests.

- Paging permits a programs memory to be non-configuous, thus allowing a program to be allocated physical memory wherever it is available.
- Every address generated by the CPU is divided into two parts namely, a page number (p) & a page offset (d).
- The page no. is used as an index into a page table. The page table Contains the base address of each page in physical memory. This base address is Combined with page offset to define the physical address that is sent to memory unit.
- Physical memory is broken into fixed size blocks called frames. Logical memory is also broken into blocks of same size called pages.

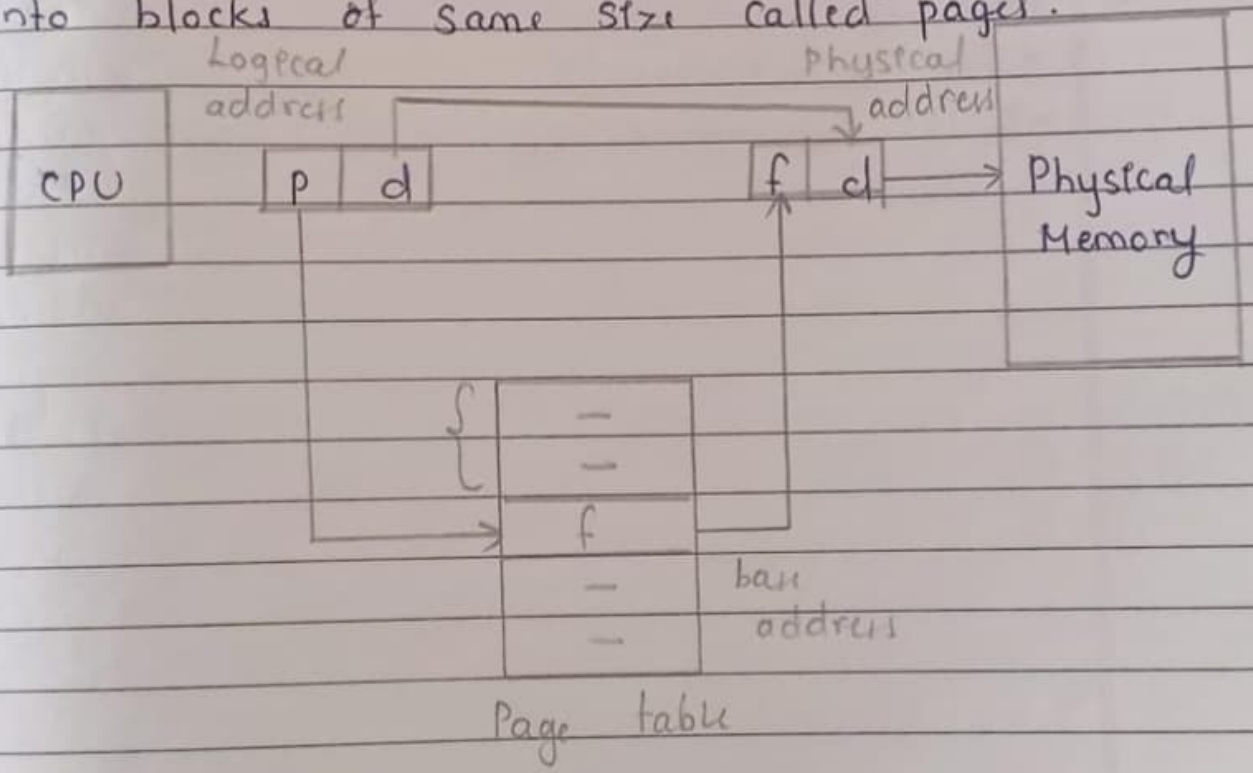


fig. Paging Hardware

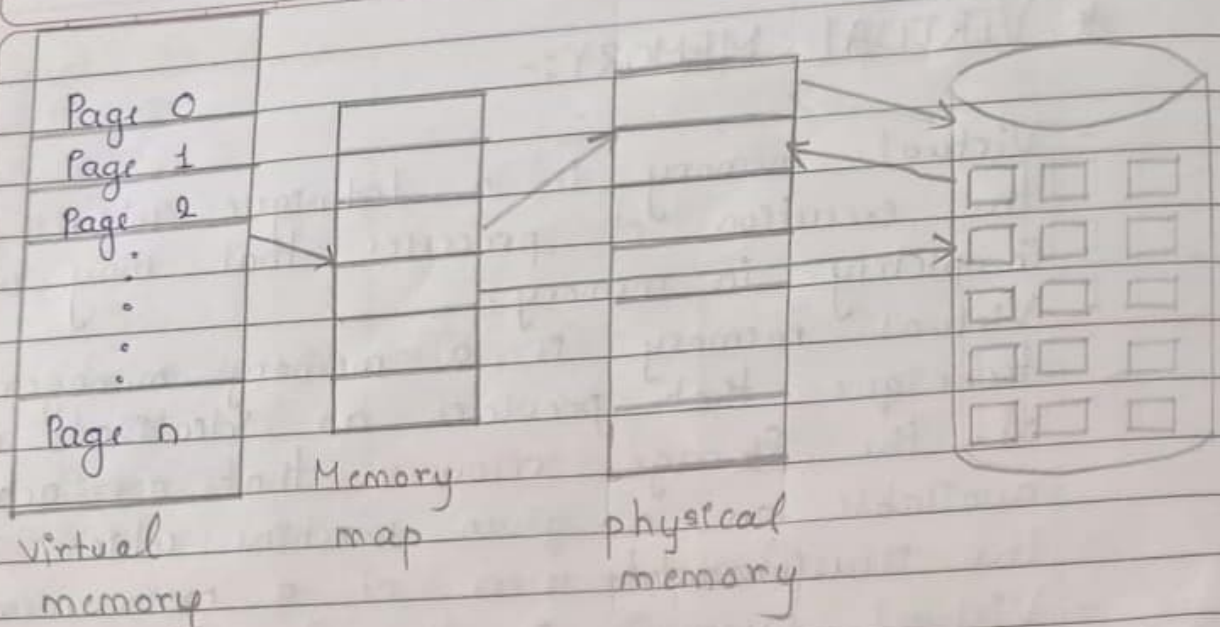


fig. Virtual Memory that is larger than physical Memory

* Introduction to Paging :-

- Paging is a memory management technique by which a Computer stores and retrieves data from secondary storage for use in main memory.
- In paging, the OS retrieves data from secondary storage in same-size blocks called pages.
- Paging is an important part of virtual memory implementations in modern OS, using secondary storage to let programs exceed the size of available physical memory.
- The basic idea behind paging is that when a process is swapped in, the pager only loads into memory those pages that it expects the process to need.

Page No.
Date

* VIRTUAL MEMORY:-

- Virtual memory is a technique which allows the execution of processes that may not be completely in memory.
- Virtual memory is a memory management technique that provides an idealized abstraction of the storage resources that are actually available on a given machine which creates the illusion to users of a main memory.
- Virtual memory is the separation of user logical memory from physical memory.
- This seperation allows an extremely large virtual memory to provided for programmers when only a smaller physical memory is available.
- Virtual memory makes the task of programming much easier because the programmer no longer needs to worry about the amount of physical memory available.
- Virtual memory is commonly implemented by demand paging.
- It can also be implemented in a segmentation system. It is not easy to implement, and may decrease performance, if it is used carelessly.

* Free Space Management Techniques:

- Files are created and deleted frequently during the operation of a computer system. Since there is only a limited amount of disk space, it is necessary to reuse the space from deleted files for new files.

- To keep track of free disk space, the file system maintains a free space list. The free space list records all disk blocks, which are free.

- To create a file, we search the free space list for the required amount of space and allocate it to the new file. This space is the removed from the free space list. When a file is deleted, its disk space is added to the free space list.

- The process of looking after and managing the free blocks of the disk is called free space management. The methods are used in free space management techniques are Bit Vector, linked list, Grouping and Counting.

i) Bit Vector

- The free space list may not be implemented as a list; it is implemented as a Bit map or Bit Vector. Bit map is series or collection of bits where each bit corresponds to a disk block

- Each block in bit map is represented by one bit. If the block is free, the bit is '0', if the block

is allocated the bit is '1'.

- For example, consider a disk where blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26 and 27 are free, the free space bit map would be, 11000011000000111001111111000 1111.....

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1	0

18	19	20	21	22	23	24	25	26	27
0	1	1	1	1	1	1	0	0	0

0 = Free block
1 = Allocated block

- The main benefit of bitmap is that it is relatively simple and efficient to find the first free blocks or n consecutive free blocks on the disk.

- Unfortunately, bit vector are inefficient unless the entire vector is kept in memory for most access. keeping it main memory is possible for smaller disks such as on microcomputers, but not for larger ones.

Advantage of Bit Map / Bit Vector :-

i) It is simple and efficient method to find the first free block or n-consecutive free blocks on the disk.

ii) A bit map requires lesser space as it uses 1 bit per block.

Disadvantage of Bit Map/ Bit Vector:

i) Extra space required to store bit map.

ii) It may require a special hardware support to do bit operation i.e. finding out which bit is 1 and which bit is 0.

iii) Bit map are inefficient unless they are kept in main memory and are written to disk occasionally for recovery needs.

ii) Linked list

- Linked list is another technique for free space management in which, the free disk blocks contains a pointer to the next free block.

- In linked list technique, link all the disk blocks together keeping a pointer to the first free block. This block contains a pointer to the next free disk block, and so on.

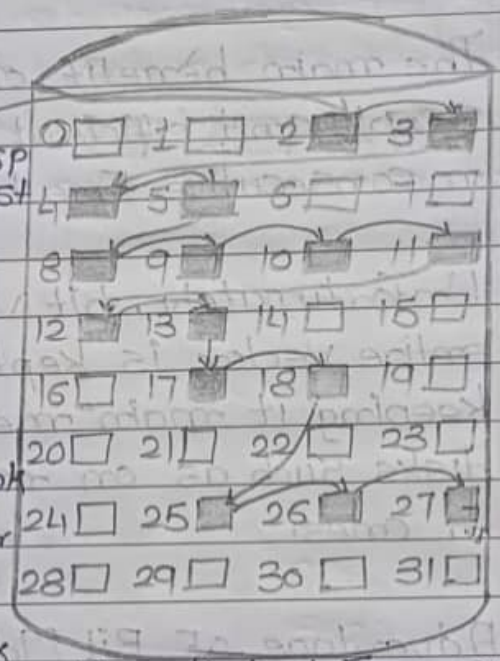


Fig:- Linked free space list on Disk

- In our example we would keep a pointer to block 2, as the first free block Block 2 would contain a pointer to block 3, which point to block 4, which would point to block 5 and so on.

- Linked list schema is not efficient; to traverse the list, each block must be read, which requires substantial I/o time.

Advantage of Linked Free Space list :-

i) If a file is to be allocated a free block, the operating system can simply allocate the first block in the free space list and move the head pointer to the next free block in the list.

ii) No disk fragmentation as every block is utilized.

Disadvantage of Linked Free space List :-

i) It is not very efficient scheme as to traverse the list; we must read each block requiring substantial time

ii) pointers use here will also consume disk space. Additional memory is therefore required.

Free blocks are:
82 127 53 251 215 23 276 361 25 26

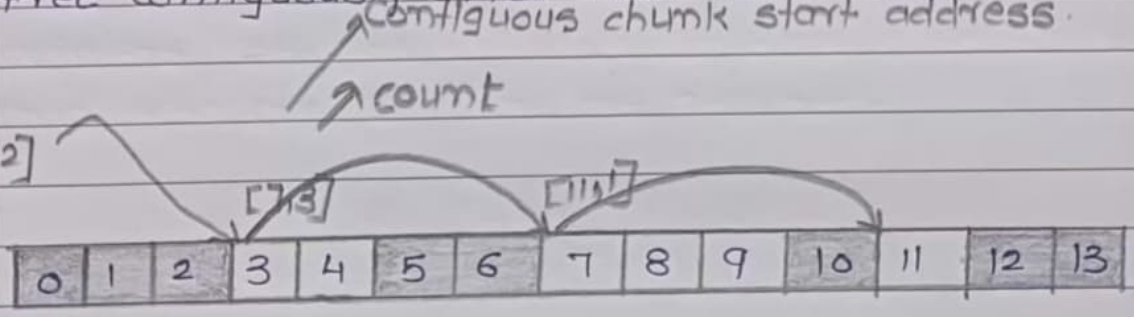
iv) Counting

- Counting is also a technique for free space management. Generally serial contiguous blocks may be allocated of free simultaneously, particularly when contiguous allocation is used.

- Thus rather than keeping a list of 'n' free disk addresses, we keep the address of first free block and number 'n' of free contiguous blocks which follows it. Each entery in the free space list then consists of a disk address and a count.

- Although each entery require more space than would a simple disk address, the overall list will be shorter, as long as the count is generally greater than 1

- Fig. 5.11 show an example of counting free space management technique. Fig 5.11 besides the free block pointer, keep a counter saying how many block are free contiguously after that free block.



Topic

* offset - no of addres locations added to base addres. in order to get specific absolute address.

* Base address - Refree point

* absolute address - Base address + offset

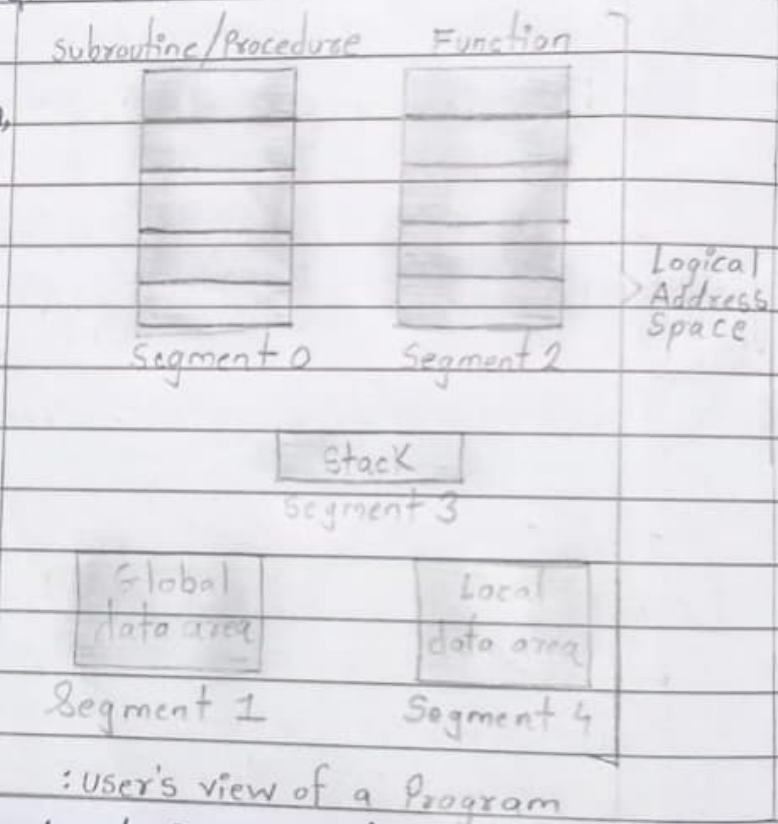
* physical address - Location in the memory.

* Logical address - genrated by cpu while program is Running.

Date / /

Segmentation

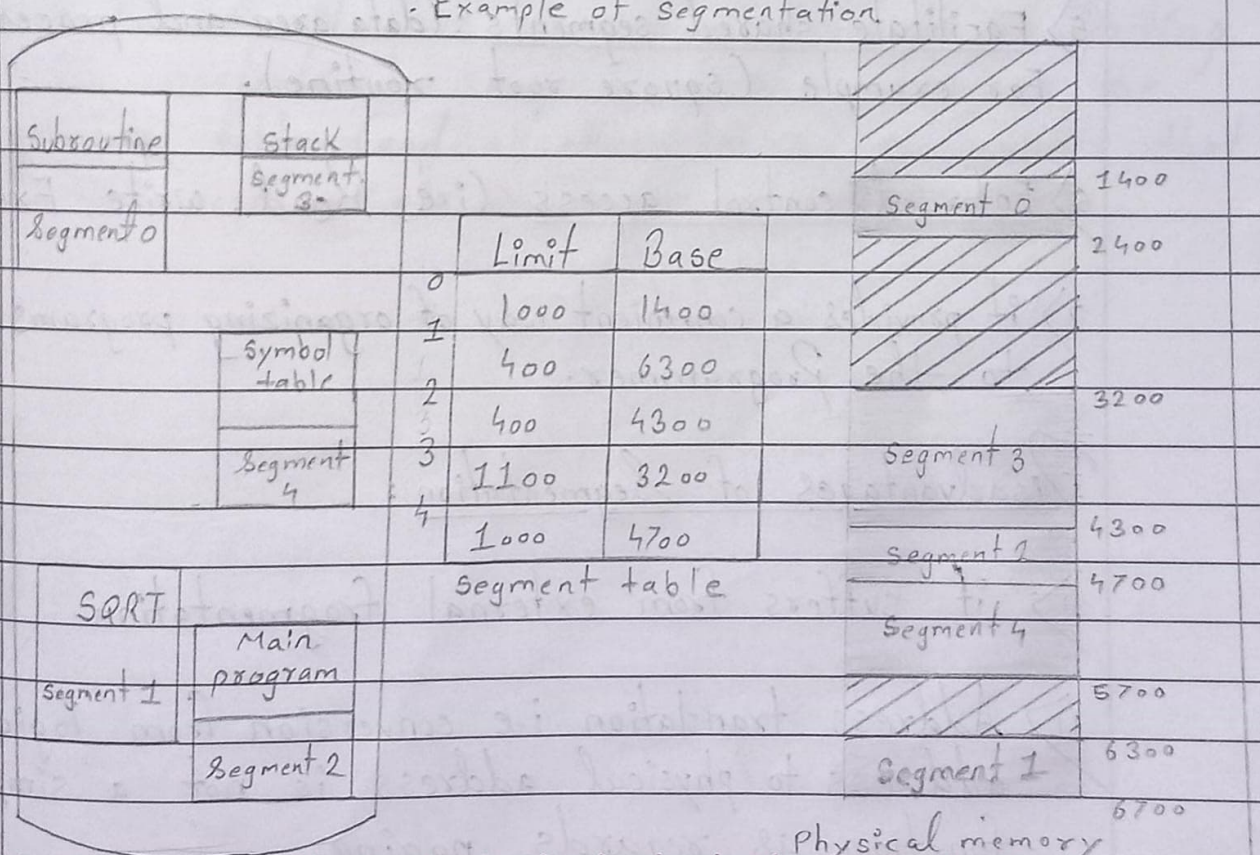
- Like paging ~~segmentation~~ Segmentation is also a memory management scheme that implements the user point of view. a program.
- In segmentation, the entire logical address space is considered as a collection of segments with each segment having a number and a length.
- The length of a segment may range from 0 to some maximum value as specified by the hardware and may also change during the execution. The user specifies each logical address consisting of a segment number (s) and an offset (d).
- A Segment is a logical unit such as main program, procedure, function, method, object, local variables, global variables, common block, stack, symbol table, arrays etc, as shown in fig.
- A segment is defined as, a logical grouping of Instructions.
- A logical address space is a collection of Segments. Every program / job is collection of segments such as subroutine array etc.



: User's view of a Program

- Each segment has a name and a length. Address specify both the segment the name and the offset within the segment. The user specifies each address by two quantities a segment name and an offset.
- A logical address consists of two parts a segment number 's' and an offset into that segment 'd'. The segment number is used as an index into segment table. Each entry of segment table has a segment base and a segment limit.
- Example, consider the situation shown in fig. we have five segments numbered from 0 through 4. The segment are actually stored in physical memory as shown in fig.

: Example of segmentation



Advantages of Segmentation:

1) Eliminate Fragmentation: By moving segments around, fragmented memory space can be combined into a single true area.

2) Provides virtual Memory: By keeping only the actively used segments in main memory. The jobs total address space size may exceed the physical memory size

3) Growing Segments: Segmentation allows dynamically growing segments.

4) Dynamic Linking and Loading: At sum time it links the subroutines and data area when explicitly referenced.

5) Facilitate shared segments: (data area and procedures). For example (square root routine).

6) Enforced control access (i.e, Read, write, Executed)

7) it provides a convenient way of organizing programs and data to the Programmer.

Disadvantages of Segmentation:

i) if suffers from external fragmentation.

ii) Address translation i.e conversion from logical address to physical address is not a simple function, as regards paging.

DOMS | Page No.
Date / /

Page Replacement Algorithms

- when the processor needs to execute a page, and if that paper page is not available in main memory then this situation is called page fault.
- for bringing in the required page into main memory, if the space is not available in memory then we need to remove the page from the main memory for allocating the space to the new page which need to be executed.
- when a page fault occurs, the operating system has to choose a page to remove from memory to make room for the page that has to be brought in. This is known as page replacement.
- The approach of page replacement, if no frame is free, we find one that is not currently being used and free it. We can free a frame by writing its contents to swap space and changing the page table (and all other tables) to indicate that the page is no longer in memory.

DOMS | Page No.
Date / /

FIFO (First in First out) Page Replacement Algorithm:

- The simplest page replacement algorithm is a FIFO A FIFO replacement algorithm associates with each page the time when that page was brought into memory.
- When a page must be replaced, the oldest page is chosen. FIFO queue is created to hold all pages in memory. We replace the page at the head of the queue.
- For example, consider the following reference string : 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1 2, 0. The three frames are initially empty. The first three refrence (7, 0, 1) cause page faults and are brought into these empty frames. The next refrences (2) replace page 7, because page 7 was brought in first

Advantages of FIFO Page Replacement Algorithm:
1) it is simple to implement.
2) it is easiest algorithm.
3) Easy to understand and execute.

Disadvantages of FIFO Page Replacement Algorithm:
1) it is not very effective.
2) system needs to track of each frame.
3) its performance is not always good.
4) it suffers from Belady's anomaly.
5) Bad replacement choice increase the page fault rate and slow process execution.

Optimal Page Replacement Algorithm

- An optimal page replacement algorithm has the lowest page fault rate of all algorithm and would never suffer from Belady's anomaly.
- Optimal replacement algorithm states replace that page which will not be used for the longest period of time.
- For example consider the following reference string 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,1,0,1,7,0,1.
- The first three reference cause faults which fills the three empty frames. The reference to page 2 replaces page 7, because 7 will be used until reference 18, whereas page 0 will be used at 5 and page 1 at 14.

example :

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
	0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
		1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
F	F	F	F		F		F			F			F				F		

Page Fault = 9
Page Hit = 11

Advantages of Optimal Page Replacement Algorithm

1) It is best possible optimal algorithm
2) It gives the smallest number of page fault
3) It never suffers from Belady's algorithm

Disadvantages of Optimal Page Replacement Algorithm:

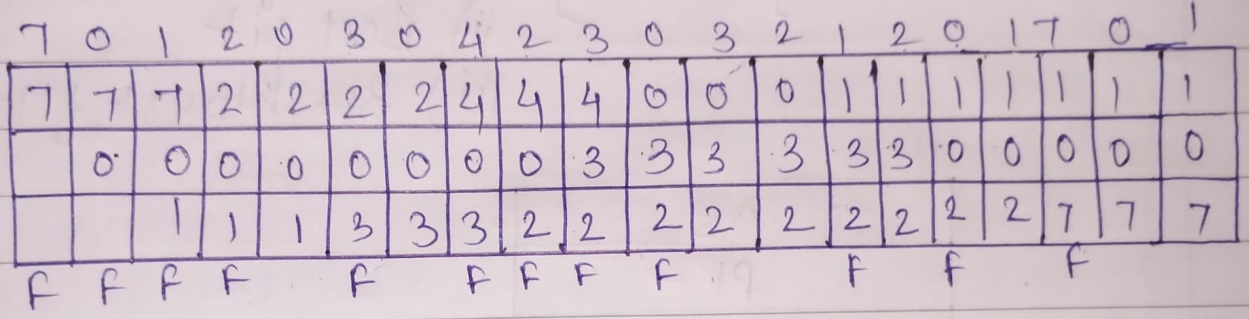
1) This algorithm is difficult to implement
2) It is only used as theoretical part of page replacement
3) It requires future knowledge of reference string.

LRU (Least Recently Used) Page Replacement Algorithm

- If we use the recent past ~~an~~ as an approximation of the near future, then we would replace that page which has not been used for the longest period of Time.
- LRU replacement associate with each page the time of its last use. When a page is to be replaced, LRU choose that page which ~~each page~~ has been not used for the longest period of time.
- We can think of this strategy as the optimal page replacement algorithm looking backward in time, rather than forward.
- For example consider the following reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 7, 2, 0, 1, 7; 0, 1.
  By applying LRU, the first five faults are same as optimal replacement. When

the reference to page 4 occur, LRU sees out of the three frames in memory, page 2 was used least recently. The most recently used page is page 0 and just before that page 3 was used.

example:



Page Fault = 12
Page Hit = 8

Advantages of LRU

1) LRU is actually quite a good algorithm
2) It never suffers from Belady's anomaly

Disadvantages of LRU

1) Its implementation is not very easy
2) LRU algorithm is that it requires additional data structure and hardware support.