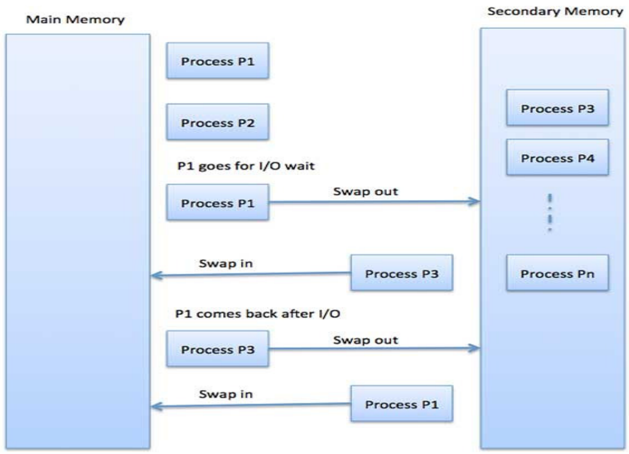
UNIT 4

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.

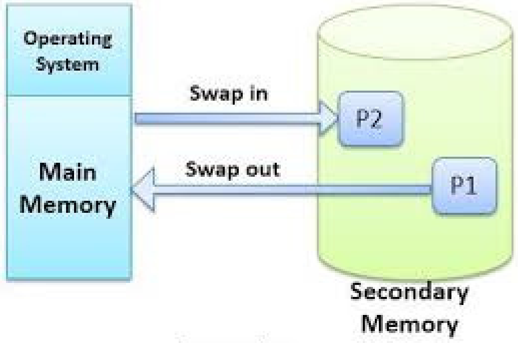
Let us assume that the user process is of size 2048KB and on a standard hard disk where swapping will take place has a data transfer rate around 1 MB per second. The actual transfer of the 1000K process to or from memory will take2048KB / 1024KB per second

```
2048KB / 1024KB per second
= 2 seconds
= 2000 milliseconds
```

Now considering in and out time, it will take complete 4000 milliseconds plus other overhead where the process competes to regain main memory.

Memory Swapping

- Memory swapping is a computer technology that enables an operating system to provide more memory to a running application or process than is available in physical random access memory.
- When the physical system memory is exhausted, the operating system can opt to make use of memory swapping techniques to get additional memory.
- Memory swapping is among the multiple techniques for memory management in modern systems.
- Physical memory alone is sometimes not sufficient, which is why there are different ways of augmenting RAM in a system with these additional options.



Swapping

How Memory Swapping Improves Performance

- Memory swapping works by making use of virtual memory and storage space in an approach that provides additional resources when required.
- In short, this additional memory enables the computer to run faster and crunch data better.
- With memory swapping, the operating system makes use of storage disk space to provide the functional equivalent of memory storage execution space.
- The space on the storage device is referred to as "swap space" and is used to run processes that have been swapped out of main physical memory.
- The process of memory swapping is managed by an operating system or by a virtual machine hypervisor. Swapping is often enabled by default, though users can choose to disable the capability.
- The actual memory swapping process and the creation of a swap file is automatically managed by the operating system. It is initiated when needed as physical RAM is used and additional capacity is required by processes and applications. As additional RAM is required, the state of the physical memory page is mapped to the swap space, enabling a form of virtual (non-physical RAM) memory capacity.
- In other words, the main purpose of swapping in memory management is to enable more usable memory than held by the computer hardware.
- There are times when physical memory will be allocated and a process needs additional memory. Rather than limiting a system to only having memory that is based on physical RAM, memory swapping enables operating systems and their users to extend memory to disk.

- Memory swapping is an essential component of modern memory management helping to ensure availability and overall system stability.

What is Swap Space or Swap File?

Swap space is storage space that is used as temporary memory capacity, when physical memory space is already exhausted. The swap file is the physical disk storage file for swap space that is used by an operating system to extend usable memory.

Understanding swap file and swap space is all about understanding memory management. Physical memory in a modern operating system is segmented in different ways, using virtual memory as an abstraction to combine both physical RAM and often swap space, as usable RAM for application processes. In memory management, operating systems make use of a page table to segment and define different memory locations. With memory swapping, the contents of memory stored in a physical element of the page table are copied to disk to maintain the same state for processes.

A swap file and its associated page of memory can be restored to different areas of a system's virtual memory as physical memory is reclaimed over time by the operating system.

The process of how to check swap memory can vary based on operating system. In Microsoft Windows operating systems, information about swap memory is listed under task manager as virtual memory. In Linux, swap space can be checked from the command line with by typing 'swapon-s', which will show allocated swap space usage.

Advantages of Memory Swapping

- **More Memory.** Memory swapping is a critical component of memory management, enabling an operating system to handle requests that would otherwise overwhelm a system.

- **Continuous Operations.** Swap file memory can be written to disk in a continuous manner, enabling faster lookup times for operations.

- **System Optimization.** Application processes of lesser importance and demand can be relegated to swap space, saving the higher performance physical memory for higher value operations.

Limitations of Memory Swapping

- **Performance.** Disk storage space, when called up by memory swapping, does not offer the same performance as physical RAM for process execution.

- **Disk Limitations**. Swap files are reliant on the stability and availability of storage media, which might not be as stable as system memory.

- **Capacity.** Memory swapping is limited by the available swap space that has been allocated by an operating system or hypervisor.

Bit Map for Dynamic Partitioning

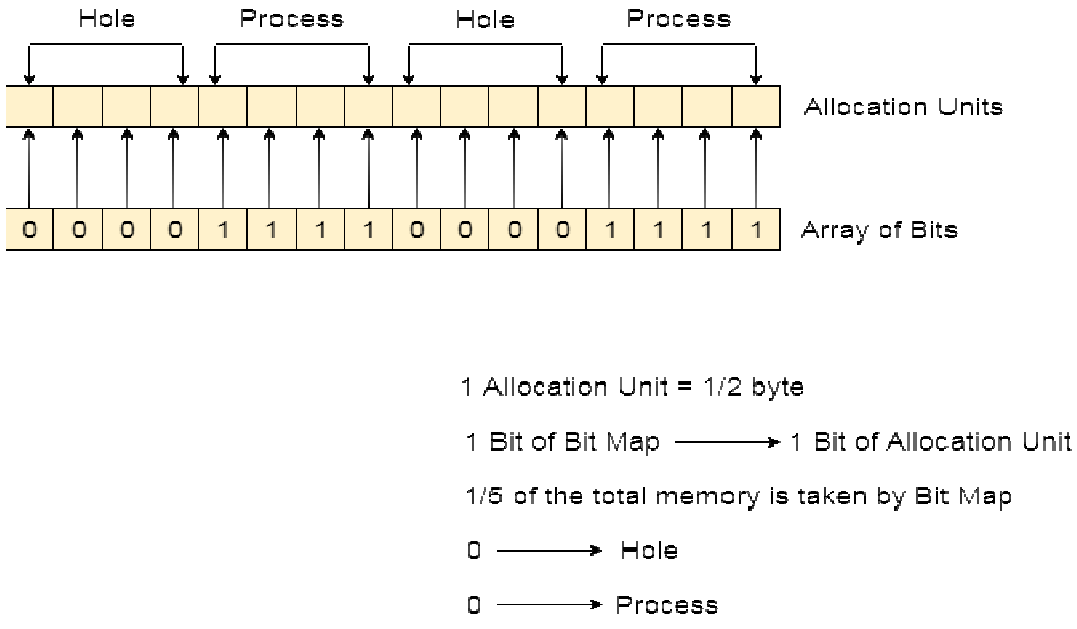
- The Main concern for dynamic partitioning is keeping track of all the free and allocated partitions. However, the Operating system uses following data structures for this task.

1. Bit Map
2. Linked List

- Bit Map is the least famous data structure to store the details. In this scheme, the main memory is divided into the collection of allocation units. One or more allocation units may be allocated to a process according to the need of that process. However, the size of the allocation unit is fixed that is defined by the Operating

System and never changed. Although the partition size may vary but the allocation size is fixed.

- The main task of the operating system is to keep track of whether the partition is free or filled. For this purpose, the operating system also manages another data structure that is called bitmap.
- The process or the hole in Allocation units is represented by a flag bit of bitmap. In the image shown below, a flag bit is defined for every bit of allocation units. However, it is not the general case, it depends on the OS that, for how many bits of the allocation units, it wants to store the flag bit.
- The flag bit is set to 1 if there is a contiguously present process at the adjacent bit in allocation unit otherwise it is set to 0.
- A string of 0s in the bitmap shows that there is a hole in the relative Allocation unit while the string of 1s represents the process in the relative allocation unit.


Hole
Process
Hole
Process
Allocation Units
0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
Array of Bits
1 Allocation Unit = 1/2 byte
1 Bit of Bit Map → 1 Bit of Allocation Unit
1/5 of the total memory is taken by Bit Map
0 → Hole
0 → Process

Bit Map for Dynamic Partitioning

Disadvantages of using Bitmap

1. The OS has to assign some memory for bitmap as well since it stores the details about allocation units. That much amount of memory cannot be used to load any process therefore that decreases the degree of multiprogramming as well as throughput.

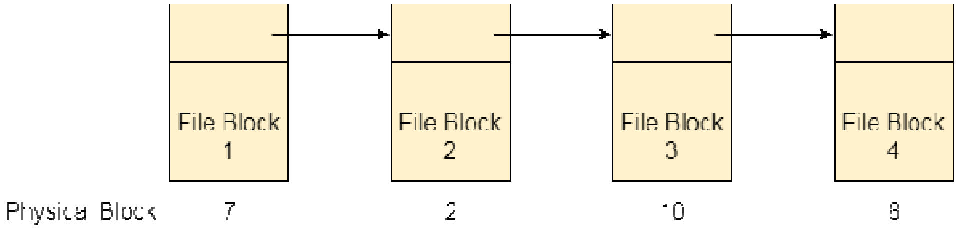
In the above image,The allocation unit is of 4 bits that is 0.5 bits. Here, 1 bit of the bitmap is representing 1 bit of allocation unit.

Therefore, in this bitmap configuration, 1/5 of total main memory is wasted.

2. To identify any hole in the memory, the OS need to search the string of 0s in the bitmap. This searching takes a huge amount of time which makes the system inefficient to some extent

Linked List Allocation

Linked List allocation solves all problems of contiguous allocation. In linked list allocation, each file is considered as the linked list of disk blocks. However, the disks blocks allocated to a particular file need not to be contiguous on the disk. Each disk block allocated to a file contains a pointer which points to the next disk block allocated to the same file.



Linked List Allocation

Advantages

1. There is no external fragmentation with linked allocation.
2. Any free block can be utilized in order to satisfy the file block requests.
3. File can continue to grow as long as the free blocks are available.
4. Directory entry will only contain the starting block address.

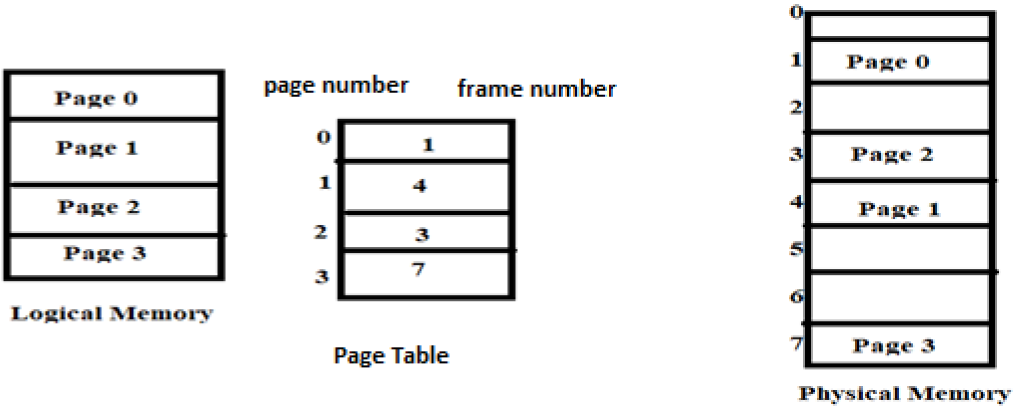
Disadvantages

1. Random Access is not provided.
2. Pointers require some space in the disk blocks.
3. Any of the pointers in the linked list must not be broken otherwise the file will get corrupted.
4. Need to traverse each block.

****

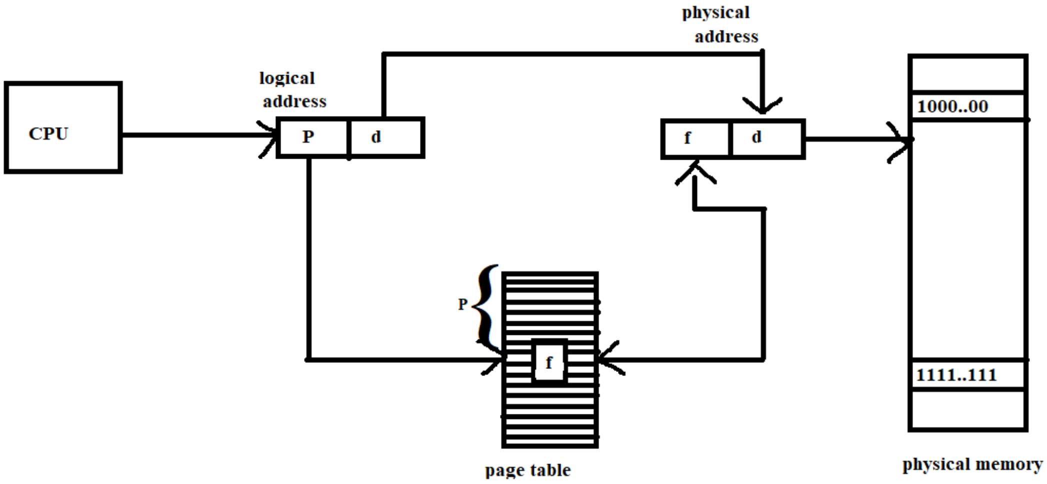
PAGING:

- Paging is a non contiguous allocation method.
- Logical memory is broken into blocks of same size called pages
- Physical memory is broken into fixed size partition called frames.
- The mapping of pages with frames is done using page table. The page table consists of entry for each page along with the frame number where the pages are stored.



CONVERSION OF LOGICAL ADDRESS TO PHYSICAL ADDRESS:

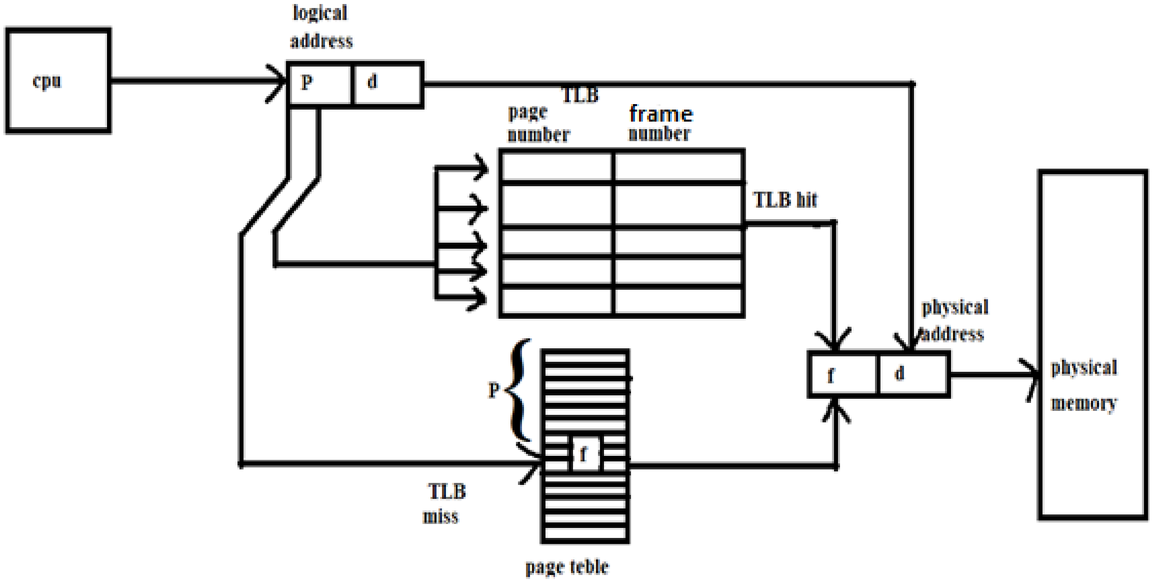
- An address generated by the CPU is refereed as logical address and an address seen by memory is referred as physical address.
- In paging method logical address is divided into two parts; page number (p)and page offset (d)where P is an index into the page table and d is the displacement within the page.
- Physical address can be generated by concatenating the offset to the frame number.



HARDWARE SUPPORT: (Translation look aside buffer (TLB)):

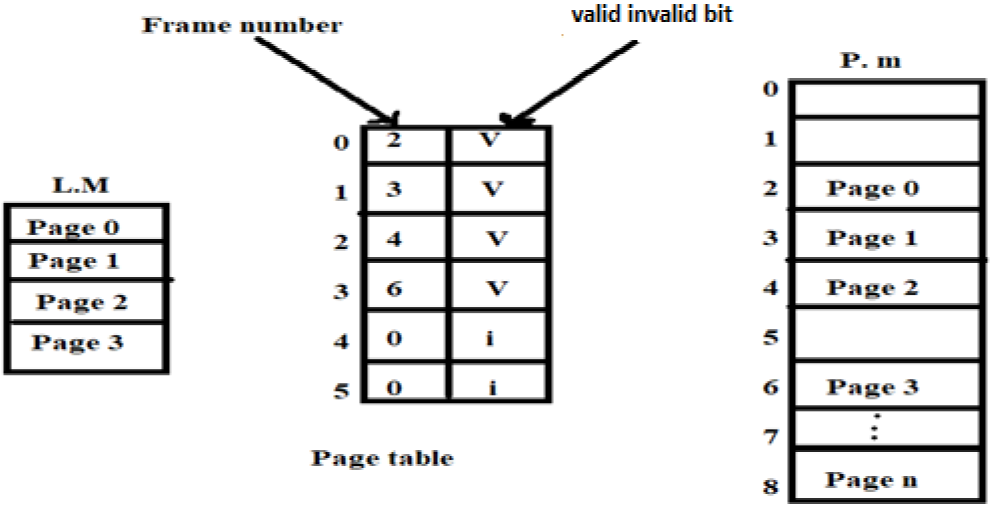
- Hardware implementation of page table can be done in many ways.
- One way is to keep the page table in memory and a page table base register (PTBR) points to the page table.To find particular page we need two memory access.
- Another way to implement page table is to use associative register or **translation look aside buffer** (TLB).
- The TLB contain only a few of the page table entries.
- To find a particular page first the page number is searched in associative registers and if it is not found there then it is searched in page table that is in memory.

Paging Hardware with TLB



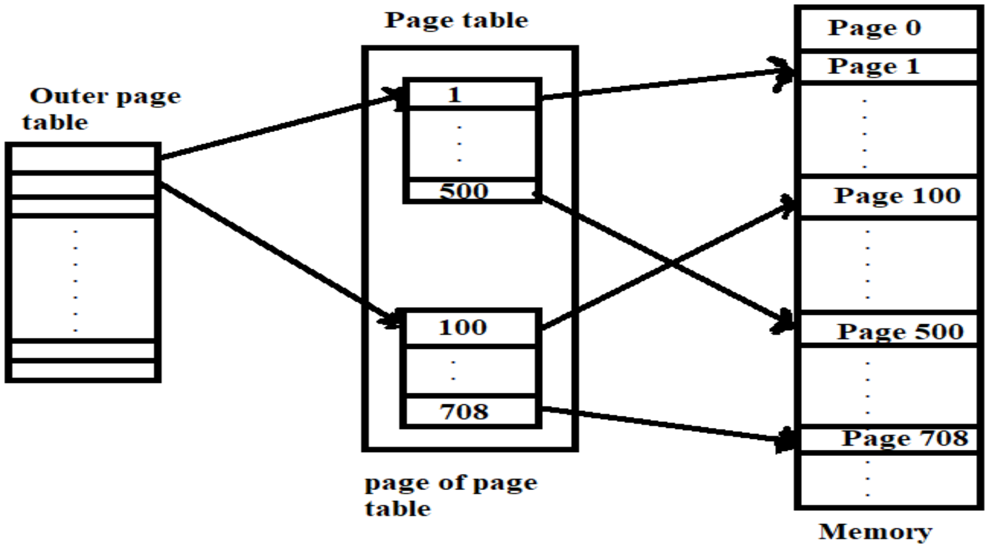
PROTECTION:

- In paging method protection can be provided by protection bits that are associated with each frame. These bits are kept in the page table and defines whether a page to be read and write or read only.
- One more bit is attached to each entry in the page table called valid- invalid bit.valid indicates that the page is a legal page and invalid indicates illegal page.
- To avoid wastage of space protection can also be provided by using page table length register (PTLR) to indicate the size of the page table.



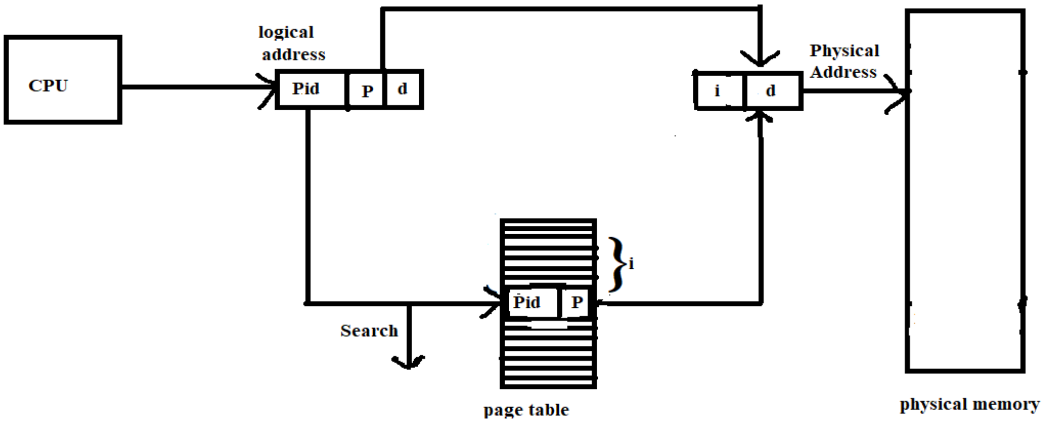
MULTILEVEL PAGING:

When page table is too large then it can be divided into smaller pieces.One way to do this is to use two level paging schemes,in which the page table itself is also paged.



INVERTED PAGE TABLE:

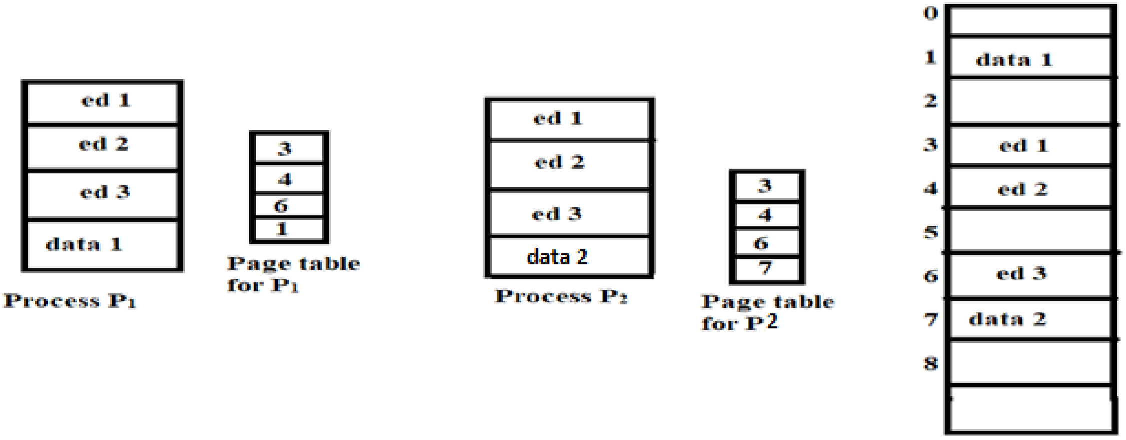
- When the page table is large it occupies more space . To solve this problem inverted page table is used.
- An inverted page table has one entry for each real page (frame) of memory.
- Each logical address in the system consists of < process id,page number ,offset>.
- Each inverted page table entry is a pair of < process id,page number >.



ADVANTAGES;

❖ SHARED PAGES:

- one advantages of paging is sharing common code.



- External fragmentation problem is solved.
- Multiprogramming is allowed.

❖ DISADVANTAGE:

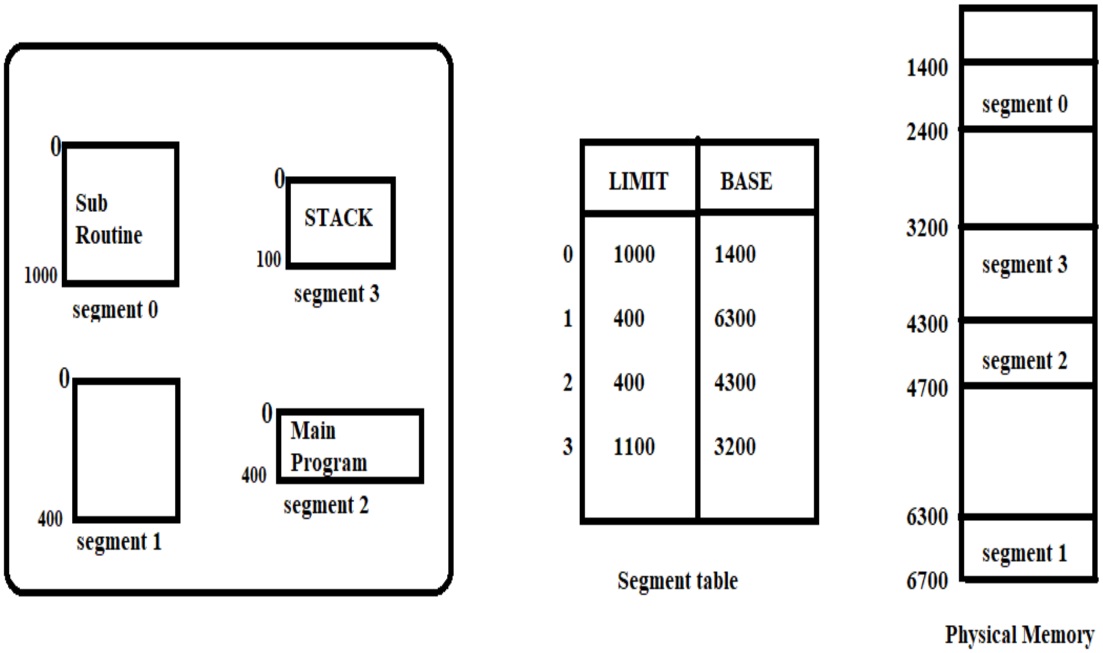
- If the number of free frames available is less than the number of page required , then memory cannot be allocated.

- Some frames may be allocated but unused which leads to internal fragmentation.
- More number of tables and register has to be maintained.

****

SEGMENTATION:

- Segmentation is a non contiguous memory management scheme.
- In this method the logical address space is divided into subdivisions called as segments.A segment is a logical group of information such as subroutine ,array or any other data structure.
- The segments need not be in equal size.
- The physical address space is divided into blocks according o the size of the segment.
- The mapping of segment and blocks are done with the help of the segment table.
- Each entry of the segment table has a segment base and a segment limit.The segment base contain the starting physical address where the segment resides in memory and segment limit specifies the length of the segment.

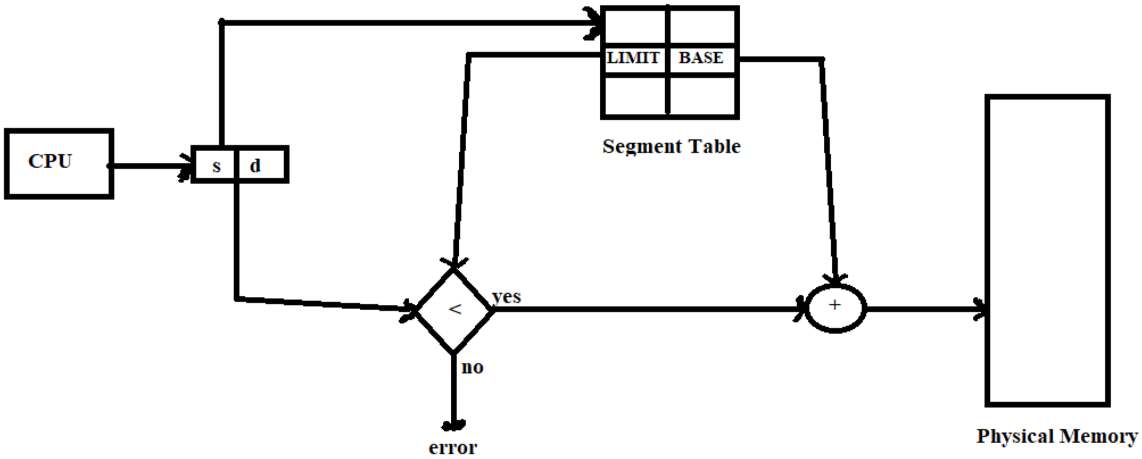


	LIMIT	BASE
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200

Segment table

❖ CONVERSION OF LOGICAL ADDRESS TO PHYSICAL ADDRESS:

- In segmentation method logical address is divided into two: segment number (s) and segment offset (d)where s is an index into the segment table and d is the displacement within the segment.
- Physical address can be generate by concatenation the offset to the segment base.



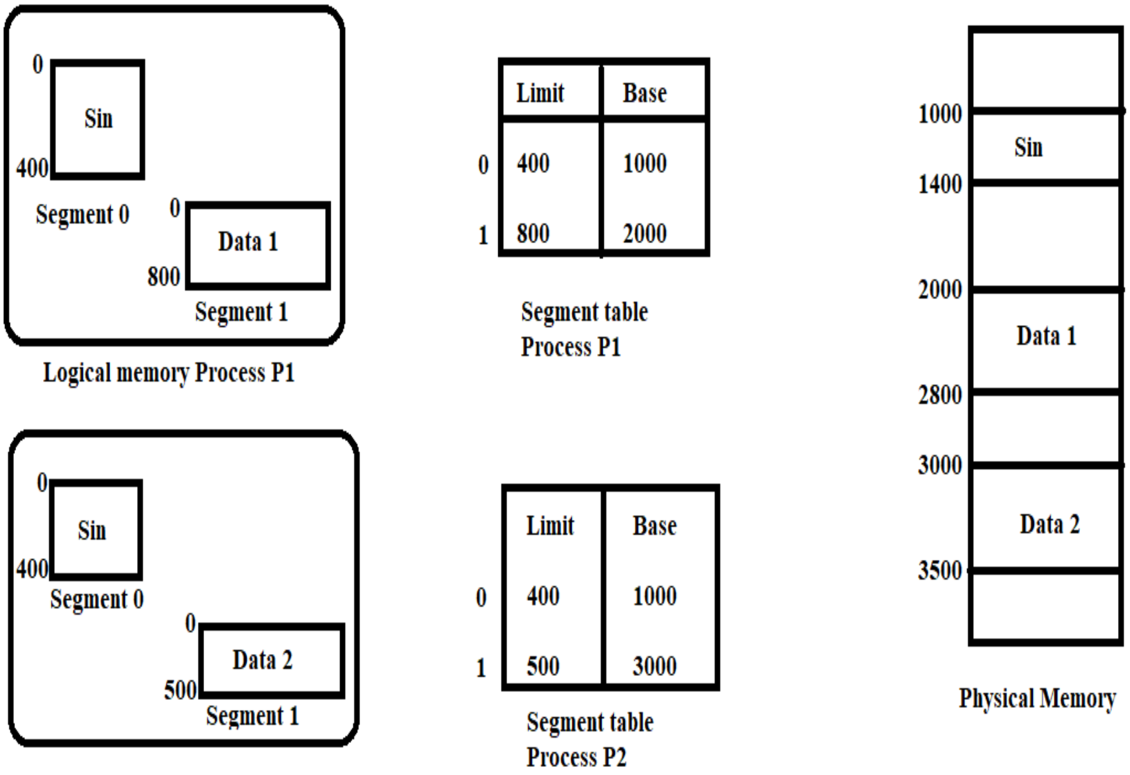
SEGMENTION HARDWARE

❖ HARDWARE SUPPORT:

- Hardware implementationof segment table can be done in many ways.
- One ways is to keep the segment table in registers and refer quickly.
- Another way is to keep the segment table in memory and a segment table base register (STBR)points to the segment table.To find a particular segment we need two memory accesses.
- Another way is to use associative register that hold the most recently used segment table entries.

❖ PROTECTION AND SHARING:

- Protection can be provided using protection bits associated with each segment table entry and prevent illegal access to memory.
- Another way to provided protection is to use segment table length register (STLR)to indicate the size.
- In segmentation method sharing of code or data is possible.



❖ DISADVANTAGES:

Segmentation may cause external fragmentation when all blocks of free memory are too small to accommodate a segment.

****

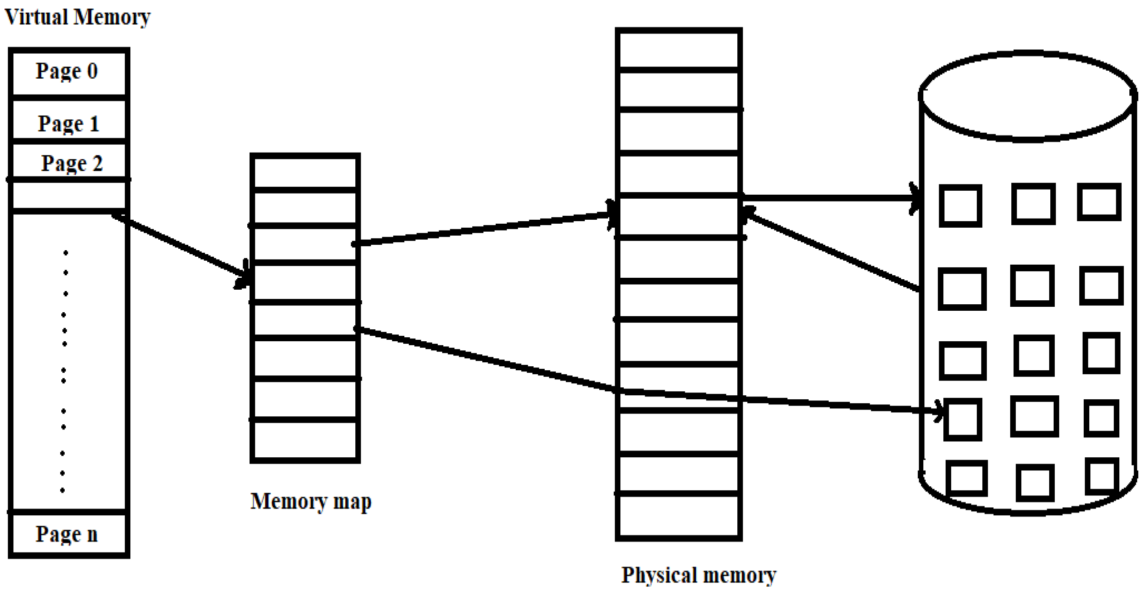
VIRTUAL MEMORY:

Virtual memory is a technique that allows the execution of process that may not be completely in memory.

- **CASES WHERE ENTIRE PROGRAM IS NOT NEEDED**
  - ✓ Programs have code that are never executed.(eg.error conditions)
  - ✓ allocate more memory for array(eg.int a [10]).

- ✓ In program certain options may be rarely used.

- **BENEFITS OF VIRTUAL MEMORY**
  - ✓ user program > physical memory.
  - ✓ more programs can run at same time and increase CPU utilization and throughput.
  - ✓ user program run faster .



DEMAND PAGING:

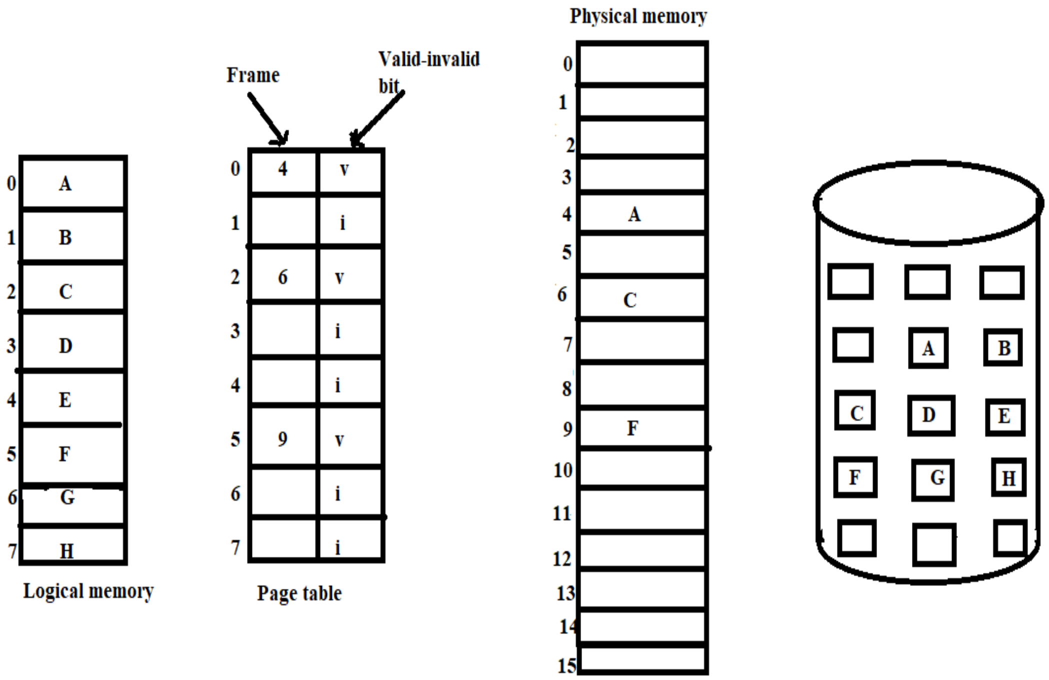
- ✓ It is similar to paging with swapping .In this method process will reside on secondary memory.
- ✓ The pages of the process are swapped into the memory only when required. Swapping is performed using lazy swapper.
- ✓ Instead of swapper we use the term pager because swapper manipulates ,entire process where as pager is concerned with

individual pages.

- ✓ In this method valid - invalid bit is used to distinguish between those pages that are in memory and those pages that are in disk.

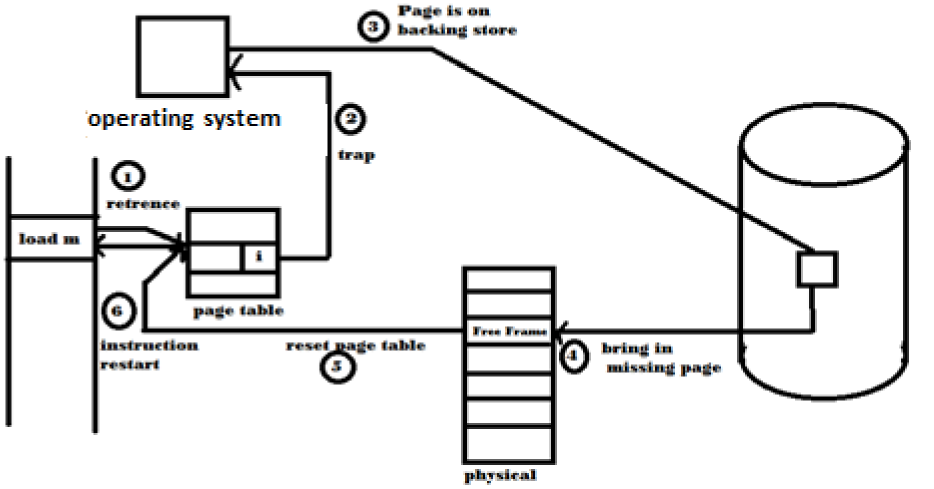
valid - page in the memory.

Invalid - not legal or page is currently on the disk.



➢ PAGE FAULT:

- Access to a page that has invalid bit causes page fault.



➢ STEPS IN HANDLING PAGE FAULT:

1. We 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 ,we terminate the process .If it was valid but,we have not yet brought in the page,we now bring it.
3. we find a free frame.
4. we schedule a disk operation to read the desired page into the newly allocated frame.
5. when the disk read is complete,we modify the internal table kept with the process and the page table to indicate that the page is now in memory .
6. we restart the instruction that was interrupt by the illegal address trap.The process can now access that page as though it had always been in memory.

- It is important to save the state when page fault occurs so that we can restart the process from the same place.
- **PURE DEMAND PAGING :**
  - A pure can also be executed with no pages in memory.This scheme is called pure demand paging.

- **HARDWARE SUPPORT:**
  - The hardware to support demand paging is given below:
    1. PAGE TABLE;
       - this table has valid - invalid bit.
    2. SECONDARY MEMORY:
       - Holds pages that are not in main memory.
       - It is known as swap device.
       - The section of disk used for demand paging is called as swap space or baking store.

- **PERFORMANCE OF DEMAND PAGING:**

- Performance of a system depends on the demand paging.
- Effective access time ==(1-p)* ma + P X page fault time .

p-probaility of page fault.

ma-memory access time.

- It is important to keep page fault rate low otherwise.
- Effective access time increase and slows process execution.

PAGE REPLACEMENT:

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 is a page fault and not an illegal memory access .The operating system determines where the desired page is residing on the disk ,but then finds there are no free frames on the free - frames list ;all memory is in use.At this case page replacement is done.

PAGE REPLACEMENT INCLUDING THE FOLLOWING STEPS:

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

2.Find a free frame.

- If there is a free frame ,use it.
- otherwise ,use a page -replacement algorithm to select a victim frame.
- write the victim page to the disk,change the page and frame tables accordingly.

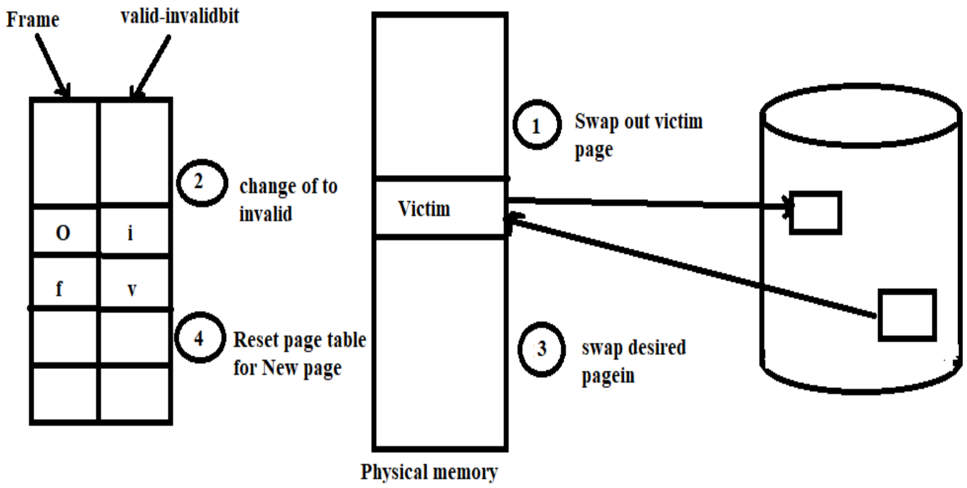
3.Read the desired page into the (newly)free frame; change the page and frame tables.

4.Restart the user process.Here two page transfer are needed which increase the effective access time , to overcome this modify bit is used.

- Each page or frame may have a modify bit associated with it in the hardware.The modify bit for a page is set when the pages modified.
- When we select a page for replacement ,we examine it's modify bit.If the bit is set, we know that the page has been modified and so we must write that page to disk .
- If the modify bit is not set , however,the page has not been

modified and so we can avoid writing page to the disk because it is already there.

- Page replacement is basic to demand paging.
- We must solve two major problem to implement demand paging we must develop a frame allocation algorithm and a page replacement algorithm.



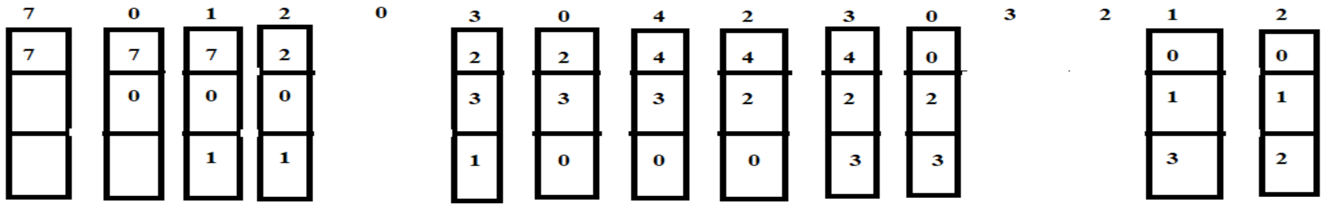
PAGE REPLACEMENT

PAGE REPLACEMENT ALGORITHMS

There are many page replacement algorithms.Generally the page replacement algorithm with lowest fault rate is selected.

FIFO ALGORITHM

- In FIFO replacement algorithm the oldest page is chosen and replaced.
- Let us apply FIFO algorithm to the following reference string with frame 3,7,0,1,2,0,3,0,4,2,3,0,3,2,1,2.



➢ ADVANTAGE:

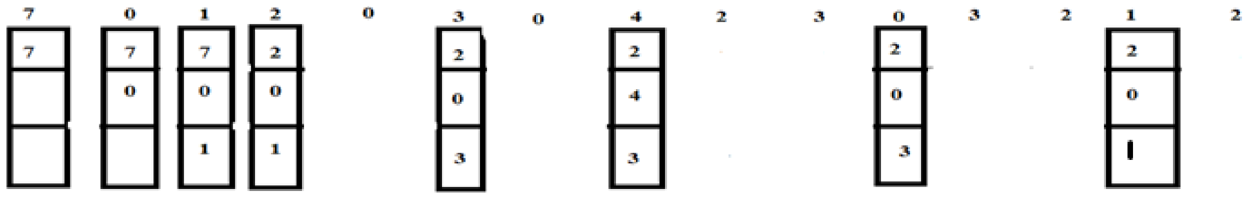
FIFO is one of the simplest method .It is easy to underst**and .**

➢ Drawback:

- The drawback of FIFO algorithm is belady'sanomaly .
- For some page replacement algorithm ,the page fault rate may increase as the number of allocated frames increases.This is called as belady's anomaly.(1,2,3,4,1,2,5,1,2,3,4,5)9 page fault -- 3 frames ,10 page fault --4 frames)

OPTIMAL ALGORITHM

- An optimal algorithm replace the page that will not be used for longest period of time.
- For example the optimal algorithm is applied to the following reference string with frame 3 as follows;



- **ADVANTAGE:**
  - An optimal algorithm has the lowest page fault rate of all algorithms.
  - This algorithm never suffer from belady's anomaly.

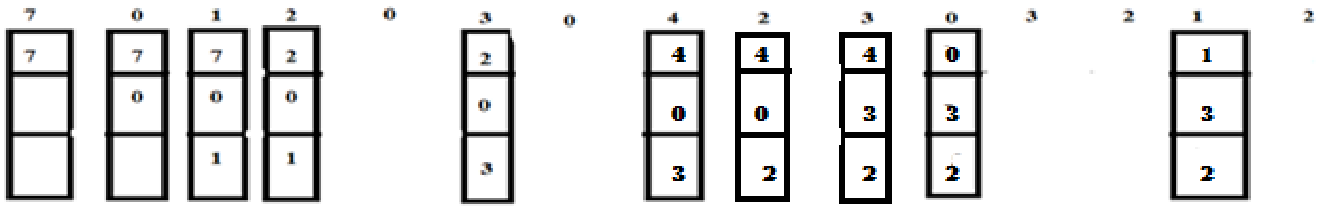
DRAWBACK ;

The drawback of optimal algorithm is it is difficult to implement because it requires future knowledge of the reference string.

LRU ALGORITHM

LRU algorithm replace the page that has not been used for longest period of time .It is called as least recently used (LRU)Algorithm.

For example the LRU algorithm is applied to the following reference string with frame 3 as follows:



Advantage:

LRU algorithm is to be quite good.

DRAWBACK:

The major problem with LRU method is implementation .LRU method can be implemented in two ways;

- **COUNTERS:**

  (a)One way to implementation LRU is to be use counter. In this a time of use field is attached with each page table entry and a clock is added with CPU.The clock is incremented for every memory reference and value is copied to the time of use field .

  b. The page with less time value is replaced

- **STACK :**

Another way to implement LRU is to keep a stack of page numbers.whenever a page is referenced ,it is removed from the stack and put on the top .In this way the top of the stack is always the most recently used page and bottom is the LRU page.

LRU Approximation Algorithms

Some algorithm may be implemented using reference bit.

1.ADDITIONAL REFERENCE BIT ALGORITHM.

In this method each page entry contains the shift register ,which is shifted to the right in equal period of time.

Each time when the pages is reference ,high bit of register is set to 1

The page withe smallest number in the register is the victim.

2.SECOND CHANCE ALGORITHM:

This method select page using FIFO and check the reference bit.

If 0-choose page as the victim and if 1-choose from the reset of pages .Reference bit should be reset periodically.

3.ENHANCED SECOND CHANCE ALGORITHM:

In this both reference bit and modified bit is considered to select a victim page.

(0,0)- victim page

(0,1)-not recently used but modified,do not replace.

(1,0)-recently used but clean,probably will be used again.

(1,1)-recently used and modified,do not replace.

COUNTING ALGORITHM

LFU algorithm and MFU algorithm comes under counting algorithm and is implemented using counters.

1) LFU ALGORITHM

This method select the least frequently used page as a victim.

2) MFU ALGORITHM

This method select the most frequently used page as a victim.

PAGE BUFFERING ALGORITHM:

Keep pool of free frames.when victim frame is chosen , place victim frame to the pool without writing it to the disk.pool a free frame out of the pool and read demanded page to this frame .Keep the list of modified frames.write the modified frame to the disk sometime when device is idle,update the list and modified bit.