1. Difference between Paging and Segmentation.

S.NO	Paging	Segmentation
1.	In paging, the program is divided into fixed or mounted size pages.	In segmentation, the program is divided into variable size sections.
2.	Page size is determined by hardware.	Here, the section size is given by the user.
3.	It is faster in comparison to segmentation.	Segmentation is slow.
4.	Paging could result in internal fragmentation.	Segmentation could result in external fragmentation.
5.	In paging, the logical address is split into a page number and page offset.	Here, the logical address is split into section number and section offset.
6.	Paging comprises a page table that encloses the base address of every page.	While segmentation also comprises the segment table which encloses segment number and segment offset.

7.	In paging, the operating system must maintain a free frame list.	In segmentation, the operating system maintains a list of holes in the main memory.
8.	Paging results in a less efficient system.	Segmentation results in a more efficient system.

1. Difference between Demand Paging and Segmentation

S.No.	Demand Paging	Segmentation
1.	In demand paging, the pages are of equal size.	While in segmentation, segments can be of different size.
2.	Page size is fixed in the demand paging.	Segment size may vary in segmentation as it grants dynamic increase of segments.
3.	It does not allows sharing of the pages.	While segments can be shared in segmentation.

4.	In demand paging, on demand pages are loaded in the memory.	In segmentation, during compilation segments are allocated to the program.
5.	Page map table in demand paging manages record of pages in memory.	Segment map table in segmentation demonstrates every segment address in the memory.
6.	It provides large virtual memory and have more efficient use of memory.	It provides virtual memory and maximum size of segment is defined by the size of memory.

Virtual Memory: Background, Demand Paging, Page Replacement Algorithm

Virtual Memory
A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard disk that's set up to emulate the computer's RAM.

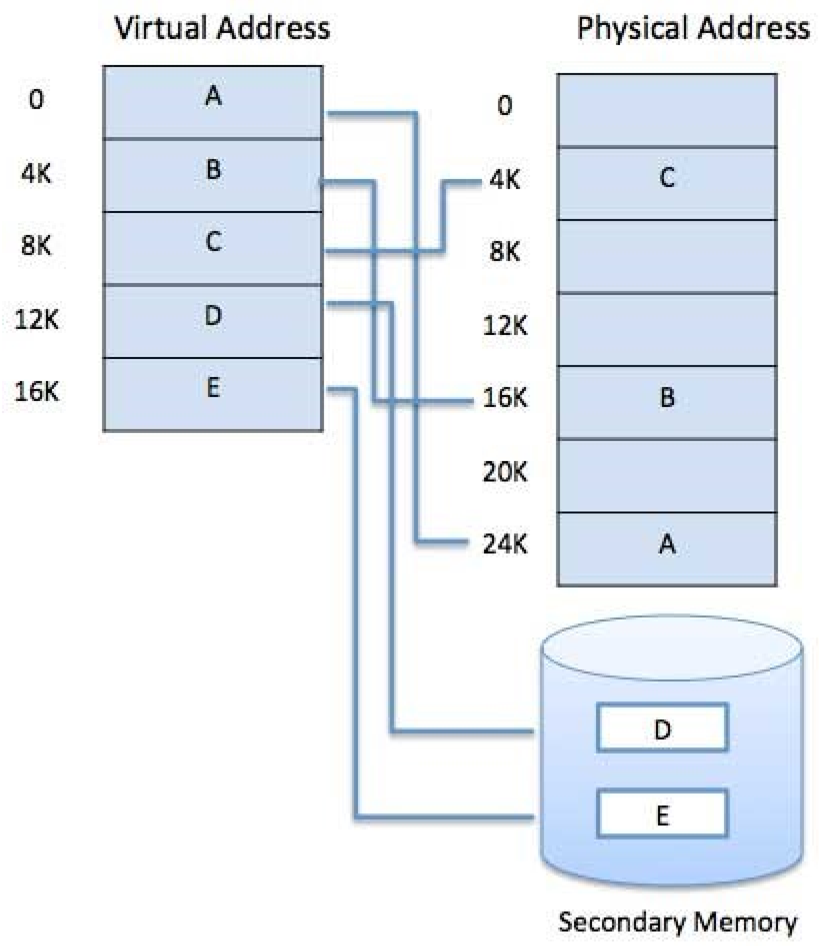
The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address.

Following are the situations, when entire program is not required to be loaded fully in main memory.

- User written error handling routines are used only when an error occurred in the data or computation.
- Certain options and features of a program may be used rarely.
- Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.

- The ability to execute a program that is only partially in memory would counter many benefits.
- Less number of I/O would be needed to load or swap each user program into memory.
- A program would no longer be constrained by the amount of physical memory that is available.
- Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.

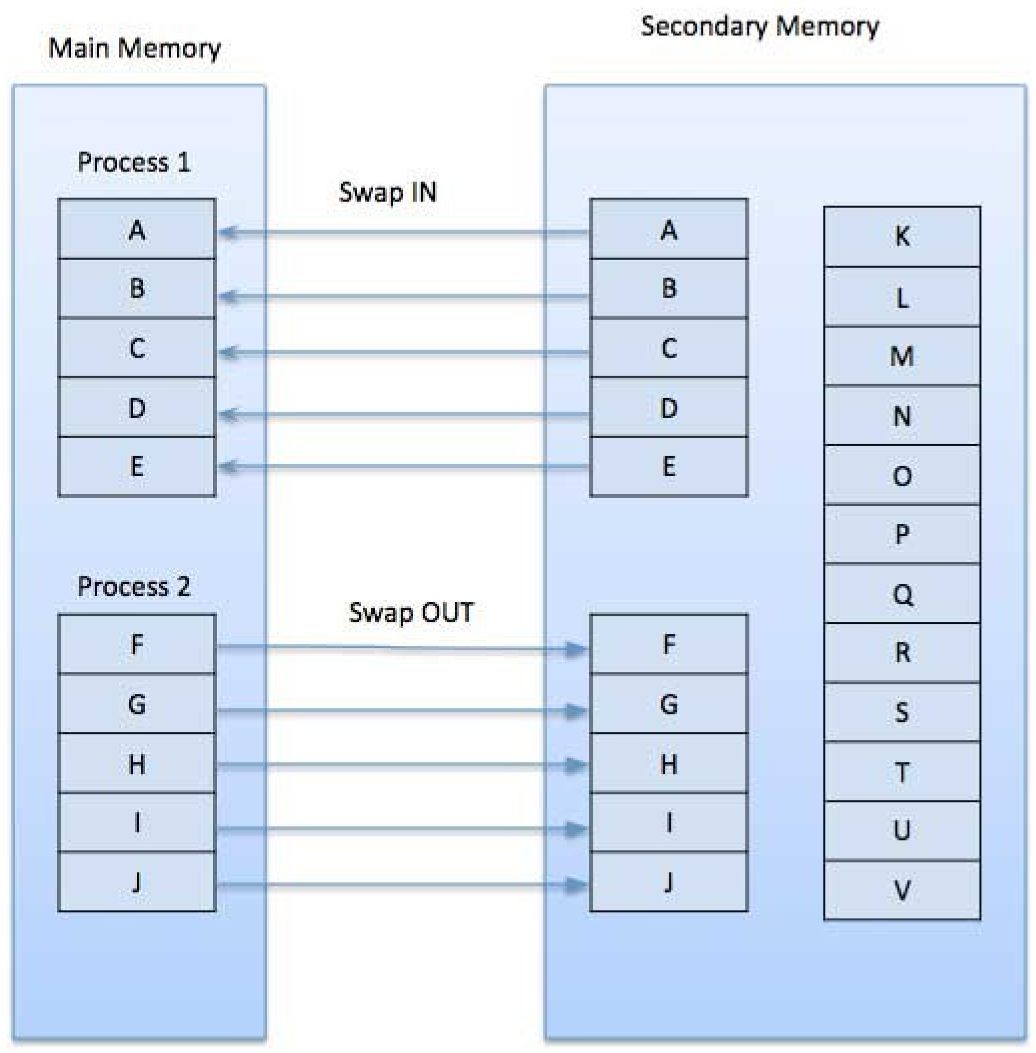
Modern microprocessors intended for general-purpose use, a memory management unit, or MMU, is built into the hardware. The MMU's job is to translate virtual addresses into physical addresses. A basic example is given below –



Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.

Demand Paging

A demand paging system is quite similar to a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance. When a context switch occurs, the operating system does not copy any of the old program’s pages out to the disk or any of the new program's pages into the main memory Instead, it just begins executing the new program after loading the first page and fetches that program's pages as they are referenced.



While executing a program, if the program references a page which is not available in the main memory because it was swapped out a little ago, the processor treats this invalid memory reference as a **page fault** and transfers control from the program to the operating system to demand the page back into the memory.

Advantages

Following are the advantages of Demand Paging −

- Large virtual memory.
- More efficient use of memory.
- There is no limit on degree of multiprogramming.

Disadvantages

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

Page Replacement Algorithm
Page replacement algorithms are the techniques using which an Operating System decides which memory pages to swap out, write to disk when a page of memory needs to be allocated. Paging happens whenever a page fault occurs and a free page cannot be used for allocation purpose accounting to reason that pages are not available or the number of free pages is lower than required pages.

When the page that was selected for replacement and was paged out, is referenced again, it has to read in from disk, and this requires for I/O completion. This process determines the quality of the page replacement algorithm: the lesser the time waiting for page-ins, the better is the algorithm.

Some Page Replacement Algorithms :
- First In First Out (FIFO)
- Least Recently Used (LRU)
- Optimal Page Replacement(OPR)

First In First Out (FIFO)
This is the simplest page replacement algorithm. In this algorithm, the OS maintains a queue that keeps track of all the pages in memory, with the oldest page at the front and the most recent page at the back.
When there is a need for page replacement, the FIFO algorithm, swaps out the page at the front of the queue, that is the page which has been in the memory for the longest time.

This is the first basic algorithm of Page Replacement Algorithms. This algorithm is basically dependent on the number of frames used. Then each frame takes up the certain page and tries to access it. When the frames are filled then the actual problem starts. The fixed number of frames is filled up with the help of first frames present. This concept is fulfilled with the help of Demand Paging

After filling up of the frames, the next page in the waiting queue tries to enter the frame. If the frame is present then, no problem is occurred. Because of the page which is to be searched is already present in the allocated frames.

If the page to be searched is found among the frames then, this process is known as Page Hit.

If the page to be searched is not found among the frames then, this process is known as Page Fault.

When Page Fault occurs this problem arises, then the First In First Out Page Replacement Algorithm comes into picture.

The First In First Out (FIFO) Page Replacement Algorithm removes the Page in the frame which is allotted long back. This means the useless page which is in the frame for a longer time is removed and the new page which is in the ready queue and is ready to occupy the frame is allowed by the First In First Out Page Replacement.

For Example:
Consider the page reference string of size 12: 1, 2, 3, 4, 5, 1, 3, 1, 6, 3, 2, 3 with frame size 4(i.e. maximum 4 pages in a frame). Use FIFO

Total

1	2	3	4	5	1	3	1	6	3	2	3

1	1	1	1	5	5	5	5	5	5	2	2
	2	2	2	2	1	1	1	1	1	1	1
		3	3	3	3	3	3	6	6	6	6
			4	4	4	4	4	4	3	3	3

M	M	M	M	M	M	H	H	M	M	M	H

M = Miss
H = Hit

Page Fault = 9
Page Fault ratio = 9/12 i.e. total miss/total possible cases

Example 2: Consider page reference string 1, 3, 0, 3, 5, 6, 3 with 3 page frames. Find the number of page faults using FIFO and LRU.

1	3	0	3	5	6	3

1	1	1	1	5	5	5
	3	3	3	3	6	6
		0	0	0	0	3
M	M	M	H	M	M	M

Total miss=Page fault=6

Page fault ratio= 6/7

Consider the page reference string of size 12: 1, 2, 3, 4, 5, 1, 3, 1, 6, 3, 2, 3 with frame size 4(i.e. maximum 4 pages in a frame). .Find the number of page faults using FIFO and LRU.

Consider the page reference string of size 12: 2, 1, 3, 2, 5, 6, 3, 1, 5, 1, 5, 3 with frame size 4(i.e. maximum 4 pages in a frame). Find out the number of page faults and its ratio respective to LRU Page Replacement Algorithm

2	1	3	2	5	6	3	1	5	1	5	3

FIFO

2	2	2	2	2	6	6	6	6	6	6	6
	1	1	1	1	1	1	1	1	1	1	1
		3	3	3	3	3	3	3	3	3	3
				5	5	5	5	5	5	5	5
M	M	M	H	M	M	H	H	H	H	H	H

LRU

2	1	3	2	5	6	3	1	5	1	5	3

2	2	2	2	2	2	2	1	1	1	1	1
	1	1	1	1	6	6	6	6	6	6	6
		3	3	3	3	3	3	3	3	3	3
				5	5	5	5	5	5	5	5
M	M	M	H	M	M	H	M	H	H	H	H

OPR

2	1	3	2	5	6	3	1	5	1	5	3

2	2	2	2	2	6	6	6	6	6	6	6
	1	1	1	1	1	1	1	1	1	1	1
		3	3	3	3	3	3	3	3	3	3
				5	5	5	5	5	5	5	5
M	M	M	H	M	M	H	H	H	H	H	H

Advantages

- Simple and easy to implement.
- Low overhead.

Disadvantages

- Poor performance.
- Doesn’t consider the frequency of use or last used time, simply replaces the oldest page.
- Suffers from Belady’s Anomaly(i.e. more page faults when we increase the number of page frames).

Least Recently Used (LRU)

Least Recently Used page replacement algorithm keeps track of page usage over a short period of time. It works on the idea that the pages that have been most heavily used in the past are most likely to be used heavily in the future too.

In LRU, whenever page replacement happens, the page which has not been used for the longest amount of time is replaced.

This is the last basic algorithm of Page Replacement Algorithms. This algorithm is basically dependent on the number of frames used. Then each frame takes up the certain page and tries to access it. When the frames are filled then the actual problem starts. The fixed number of frames is filled up with the help of first frames present. This concept is fulfilled with the help of Demand Paging

After filling up of the frames, the next page in the waiting queue tries to enter the frame. If the frame is present then, no problem is occurred. Because of the page which is to be searched is already present in the allocated frames.

If the page to be searched is found among the frames then, this process is known as Page Hit.

If the page to be searched is not found among the frames then, this process is known as Page Fault.

When Page Fault occurs this problem arises, then the Least Recently Used (LRU) Page Replacement Algorithm comes into picture.

The Least Recently Used (LRU) Page Replacement Algorithms works on a certain principle. The principle is:

Replace the page with the page which is less dimension of time recently used page in the past.

Example

Consider the page reference string of size 12: 1, 2, 3, 4, 5, 1, 3, 1, 6, 3, 2, 3 with frame size 4(i.e. maximum 4 pages in a frame). Use LRU

1	2	3	4	5	1	3	1	6	3	2	3

1	1	1	1	5	5	5	5	5	5	2	2
	2	2	2	2	1	1	1	1	1	1	1
		3	3	3	3	3	3	3	3	3	3
			4	4	4	4	4	6	6	6	6

M	M	M	M	M	M	H	H	M	H	M	H

M = Miss
H = Hit

Total Page Fault = 8
Page Fault ratio = 8/12

Advantages

- Efficient.
- Doesn't suffer from Belady’s Anomaly.

Disadvantages

- Complex Implementation.
- Expensive.
- Requires hardware support.

Optimal Page Replacement(OP)
Optimal Page Replacement algorithm is the best page replacement algorithm as it gives the least number of page faults. It is also known as OPT, clairvoyant replacement algorithm, or Belady’s optimal page replacement policy.
In this algorithm, pages are replaced which would not be used for the longest duration of time in the future, i.e., the pages in the memory which are going to be referred farthest in the future are replaced.
This algorithm was introduced long back and is difficult to implement because it requires future knowledge of the program behaviour. However, it is possible to implement optimal page replacement on the second run by using the page reference information collected on the first run.

This is the second basic algorithm of Page Replacement Algorithms. This algorithm is basically dependent on the number of frames used. Then each frame takes up the certain page and tries to access it. When the frames are filled then the actual problem starts. The fixed number of frames is filled up with the help of first frames present. This concept is fulfilled with the help of Demand Paging

After filling up of the frames, the next page in the waiting queue tries to enter the frame. If the frame is present then, no problem is occurred. Because of the page which is to be searched is already present in the allocated frames.

If the page to be searched is found among the frames then, this process is known as Page Hit.

If the page to be searched is not found among the frames then, this process is known as Page Fault.

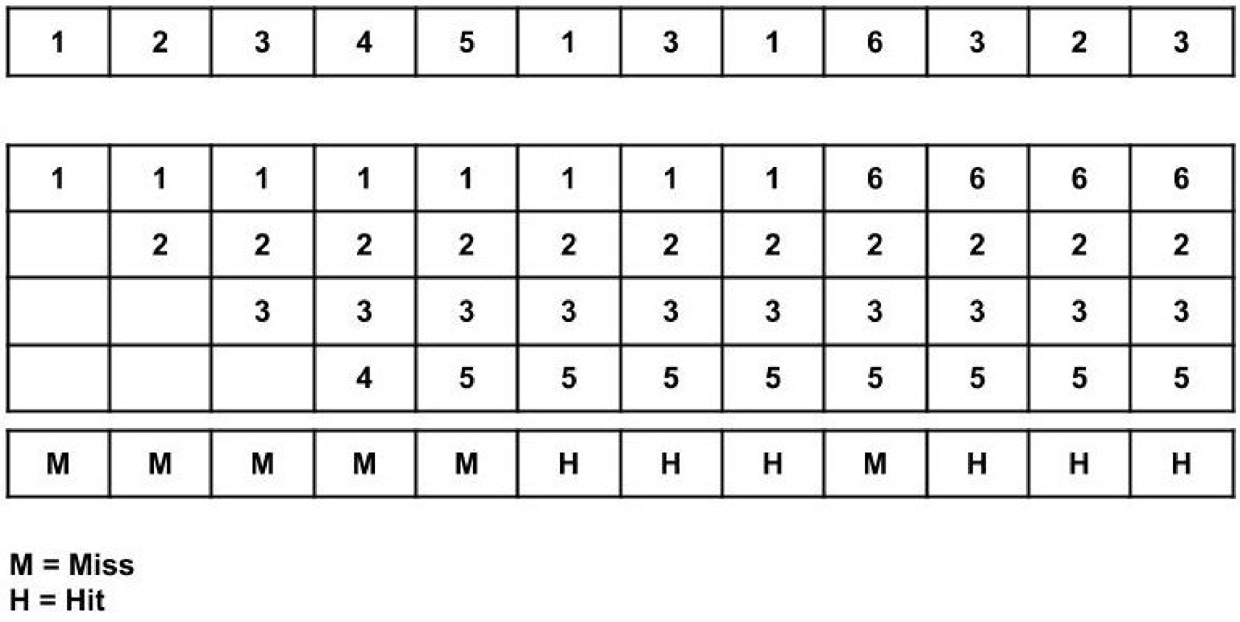
When Page Fault occurs this problem arises, then the OPTIMAL Page Replacement Algorithm comes into picture.

The OPTIMAL Page Replacement Algorithms works on a certain principle. The principle is:

Replace the Page which is not used in the Longest Dimension of time in future

This principle means that after all the frames are filled then, see the future pages which are to occupy the frames. Go on checking for the pages which are already available in the frames. Choose the page which is at last.

For Example
Consider the page reference string of size 12: 1, 2, 3, 4, 5, 1, 3, 1, 6, 3, 2, 3 with frame size 4(i.e. maximum 4 pages in a frame). Use OP



1	2	3	4	5	1	3	1	6	3	2	3

1	1	1	1	1	1	1	1	6	6	6	6
	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3	3
			4	5	5	5	5	5	5	5	5

M	M	M	M	M	H	H	H	M	H	H	H

M = Miss
H = Hit

Total Page Fault = 6
Page Fault ratio = 6/12
Advantages

- Easy to Implement.
- Simple data structures are used.
- Highly efficient.

Disadvantages

- Requires future knowledge of the program.
- Time-consuming.