Unit-4

Memory Management

Memory Management is the process of controlling and coordinating computer memory, assigning portions known as blocks to various running programs to optimize the overall performance of the system.
It is the most important function of an operating system that manages primary memory. It helps processes to move back and forward between the main memory and execution disk. It helps OS to keep track of every memory location, irrespective of whether it is allocated to some process or it remains free.

Why Use Memory Management?

Here, are reasons for using memory management:

- It allows you to check how much memory needs to be allocated to processes that decide which processor should get memory at what time.
- Tracks whenever inventory gets freed or unallocated. According to it will update the status.
- It allocates the space to application routines.
- It also make sure that these applications do not interfere with each other.
- Helps protect different processes from each other
- It places the programs in memory so that memory is utilized to its full extent.

Memory Management Techniques

Here, are some most crucial memory management techniques:

Single Contiguous Allocation

It is the easiest memory management technique. In this method, all types of computer's memory except a small portion which is reserved for the OS is available for one application. For example, MS-DOS operating system allocates memory in this way. An embedded system also runs on a single application.

Partitioned Allocation

It divides primary memory into various memory partitions, which is mostly contiguous areas of memory. Every partition stores all the information for a specific task or job. This method consists of allotting a partition to a job when it starts & unallocated when it ends.

Paged Memory Management

This method divides the computer's main memory into fixed-size units known as page frames. This hardware memory management unit maps pages into frames which should be allocated on a page basis.

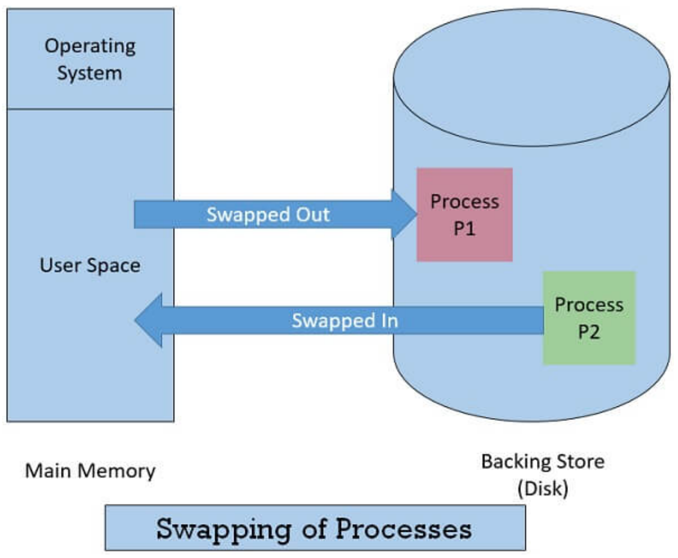
Segmented Memory Management

Segmented memory is the only memory management method that does not provide the user's program with a linear and contiguous address space.

Segments need hardware support in the form of a segment table. It contains the physical address of the section in memory, size, and other data like access protection bits and status.

What is Swapping?

Swapping is a method in which the process should be swapped temporarily from the main memory to the backing store. It will be later brought back into the memory for continue execution.
Backing store is a hard disk or some other secondary storage device that should be big enough in order to accommodate copies of all memory images for all users. It is also capable of offering direct access to these memory images.



Benefits of Swapping

Here, are major benefits/pros of swapping:

- It offers a higher degree of multiprogramming.
- Allows dynamic relocation. For example, if address binding at execution time is being used, then processes can be swap in different locations. Else in case of compile and load time bindings, processes should be moved to the same location.
- It helps to get better utilization of memory.
- Minimum wastage of CPU time on completion so it can easily be applied to a priority-based scheduling method to improve its performance.

Contiguous memory allocation in os

Contiguous Memory Allocation is a type of memory allocation technique where processes are allotted a continuous block of space in memory. This block can be of fixed size for all the processes in a fixed size partition scheme or can be of variable size depending on the requirements of the process in a variable size partition scheme.

Contiguous Memory Allocation Techniques

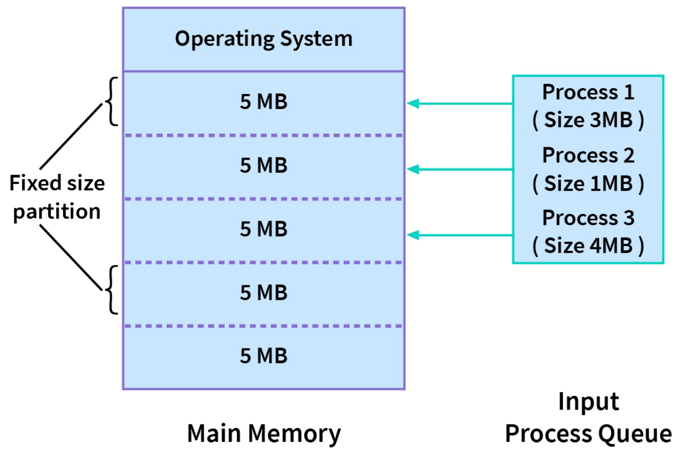
Whenever a process has to be allocated space in the memory, following the contiguous memory allocation technique, we have to allot the process a continuous empty block of space to reside. This allocation can be done in two ways:

1. Fixed-size Partition Scheme
2. Variable-size Partition Scheme

Let us look at both of these schemes in detail, along with their advantages and disadvantages.

Fixed-size Partition Scheme

In this type of contiguous memory allocation technique, each process is allotted a fixed size continuous block in the main memory. That means there will be continuous blocks of fixed size into which the complete memory will be divided, and each time a process comes in, it will be allotted one of the free blocks. Because irrespective of the size of the process, each is allotted a block of the same size memory space. This technique is also called static partitioning.



In the diagram above, we have 3 processes in the input queue that have to be allotted space in the memory. As we are following the fixed size partition technique, the memory has fixed-sized blocks. The first process, which is of size 3MB is also allotted a 5MB block, and the second process, which is of size 1MB, is also allotted a 5MB block, and the 4MB process is also allotted a 5MB block. So, the process size doesn't matter. Each is allotted the same fixed-size memory block.

It is clear that in this scheme, the number of continuous blocks into which the memory will be divided will be decided by the amount of space each block covers, and this, in turn, will dictate how many processes can stay in the main memory at once.

Advantages

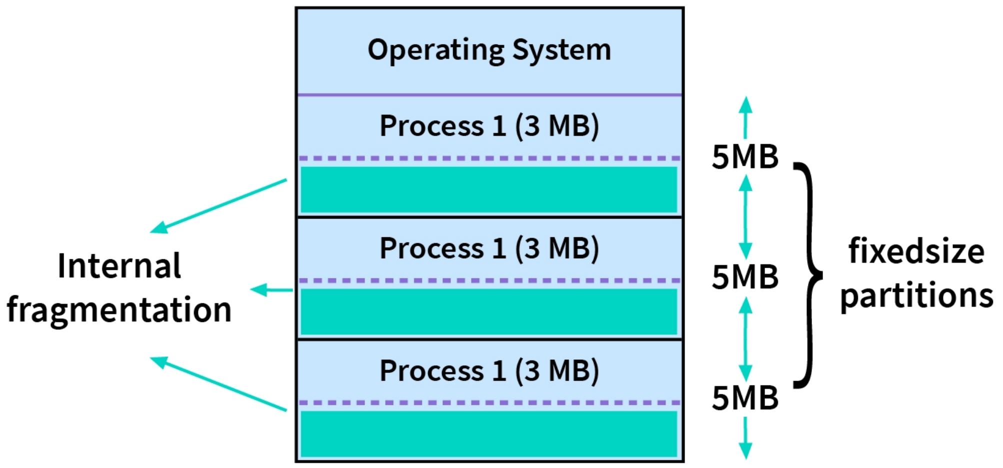
The advantages of a fixed-size partition scheme are:

1. Because all of the blocks are the same size, this scheme is simple to implement. All we have to do now is divide the memory into fixed blocks and assign processes to them.
2. It is easy to keep track of how many blocks of memory are left, which in turn decides how many more processes can be given space in the memory.
3. As at a time multiple processes can be kept in the memory, this scheme can be implemented in a system that needs multiprogramming.

Disdvantages

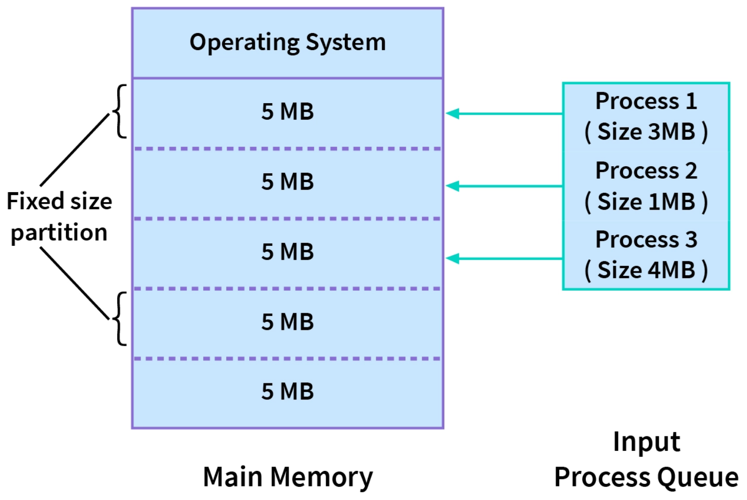
Though the fixed-size partition scheme has many advantages, it also has some disadvantages:

1. As the size of the blocks is fixed, we will not be able to allot space to a process that has a greater size than the block.
2. The size of the blocks decides the degree of multiprogramming, and only that many processes can remain in the memory at once as the number of blocks.
3. If the size of the block is greater than the size of the process, we have no other choice but to assign the process to this block, but this will lead to much empty space left behind in the block. This empty space could've been used to accommodate a different process. This is called internal fragmentation. Hence, this technique may lead to space wastage.



Variable-size Partition Scheme

In this type of contiguous memory allocation technique, no fixed blocks or partitions are made in the memory. Instead, each process is allotted a variable-sized block depending upon its requirements. That means, whenever a new process wants some space in the memory, if available, this amount of space is allotted to it. Hence, the size of each block depends on the size and requirements of the process which occupies it.



In the diagram above, there are no fixed-size partitions. Instead, the first process needs 3MB memory space and hence is allotted that much only. Similarly, the other 3 processes are allotted only that much space that is required by them.

As the blocks are variable-sized, which is decided as processes arrive, this scheme is also called Dynamic Partitioning.

Advantages

The advantages of a variable-size partition scheme are:

1. As the processes have blocks of space allotted to them as per their requirements, there is no internal fragmentation. Hence, there is no memory wastage in this scheme.
2. The number of processes that can be in the memory at once will depend upon how many processes are in the memory and how much space they occupy. Hence, it will be different for different cases and will be dynamic.
3. As there are no blocks that are of fixed size, even a process of big size can be allotted space.

Disdvantages

Though the variable-size partition scheme has many advantages, it also has some disadvantages:

1. Because this approach is dynamic, a variable-size partition scheme is difficult to implement.
2. It is difficult to keep track of processes and the remaining space in the memory.

Paging

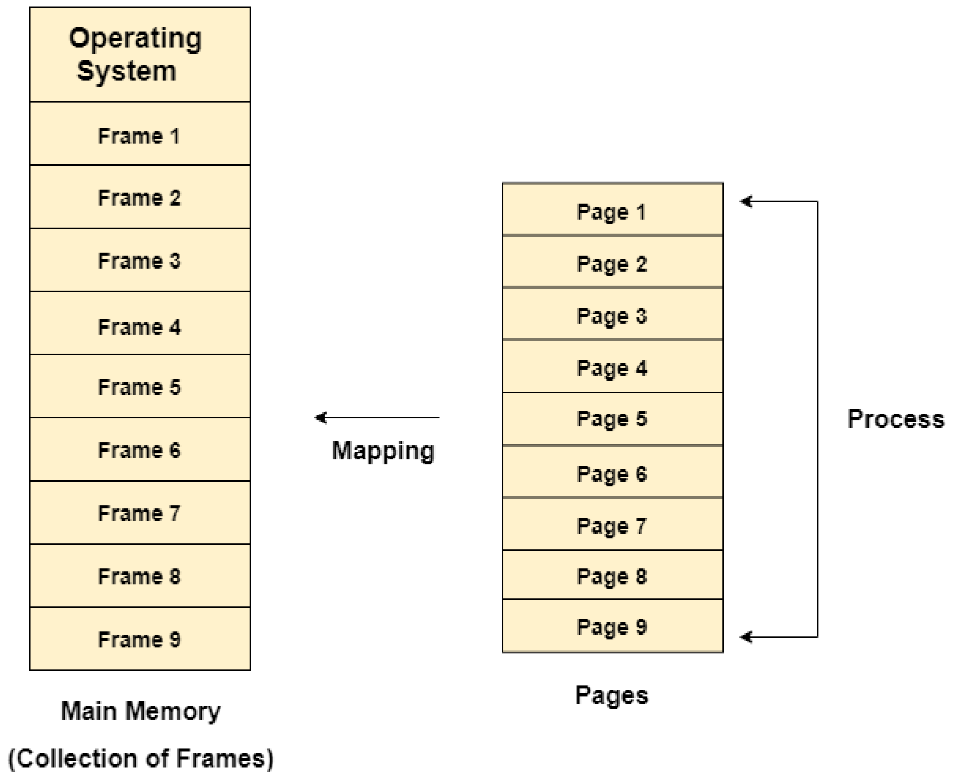
In Operating Systems, Paging is a storage mechanism used to retrieve processes from the secondary storage into the main memory in the form of pages.

The main idea behind the paging is to divide each process in the form of pages. The main memory will also be divided in the form of frames.

One page of the process is to be stored in one of the frames of the memory. The pages can be stored at the different locations of the memory but the priority is always to find the contiguous frames or holes.

Pages of the process are brought into the main memory only when they are required otherwise they reside in the secondary storage.

Different operating system defines different frame sizes. The sizes of each frame must be equal. Considering the fact that the pages are mapped to the frames in Paging, page size needs to be as same as frame size.



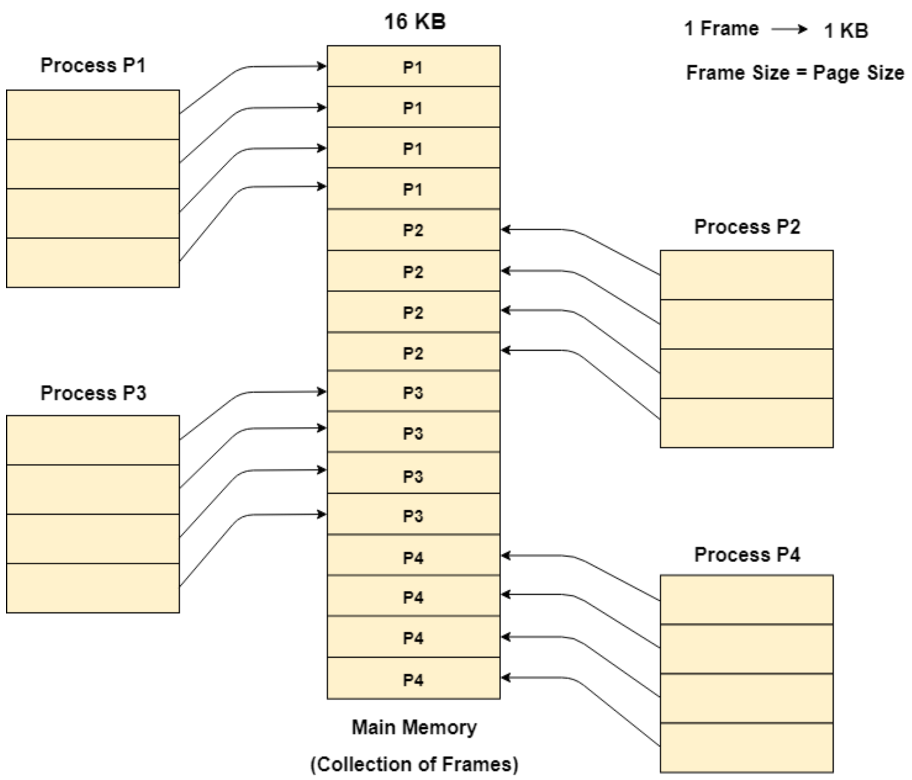
Example

Let us consider the main memory size 16 Kb and Frame size is 1 KB therefore the main memory will be divided into the collection of 16 frames of 1 KB each.

There are 4 processes in the system that is P1, P2, P3 and P4 of 4 KB each. Each process is divided into pages of 1 KB each so that one page can be stored in one frame.

Initially, all the frames are empty therefore pages of the processes will get stored in the contiguous way.

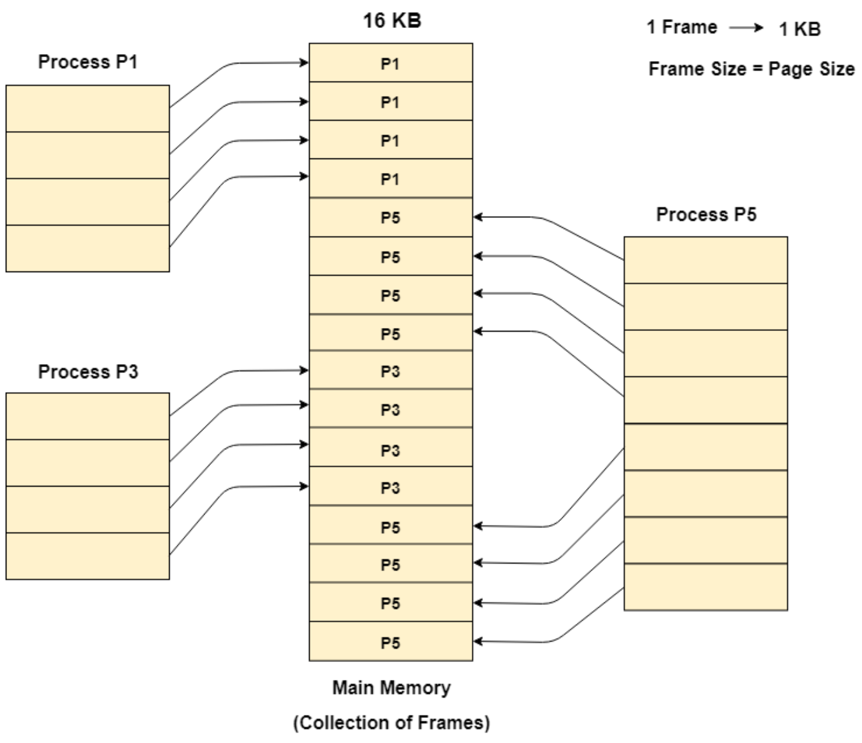
Frames, pages and the mapping between the two is shown in the image below.



Paging

Let us consider that, P2 and P4 are moved to waiting state after some time. Now, 8 frames become empty and therefore other pages can be loaded in that empty place. The process P5 of size 8 KB (8 pages) is waiting inside the ready queue.

Given the fact that, we have 8 non contiguous frames available in the memory and paging provides the flexibility of storing the process at the different places. Therefore, we can load the pages of process P5 in the place of P2 and P4.



Paging

Segmentation

In Operating Systems, Segmentation is a memory management technique in which the memory is divided into the variable size parts. Each part is known as a segment which can be allocated to a process.

The details about each segment are stored in a table called a segment table. Segment table is stored in one (or many) of the segments.

Segment table contains mainly two information about segment:

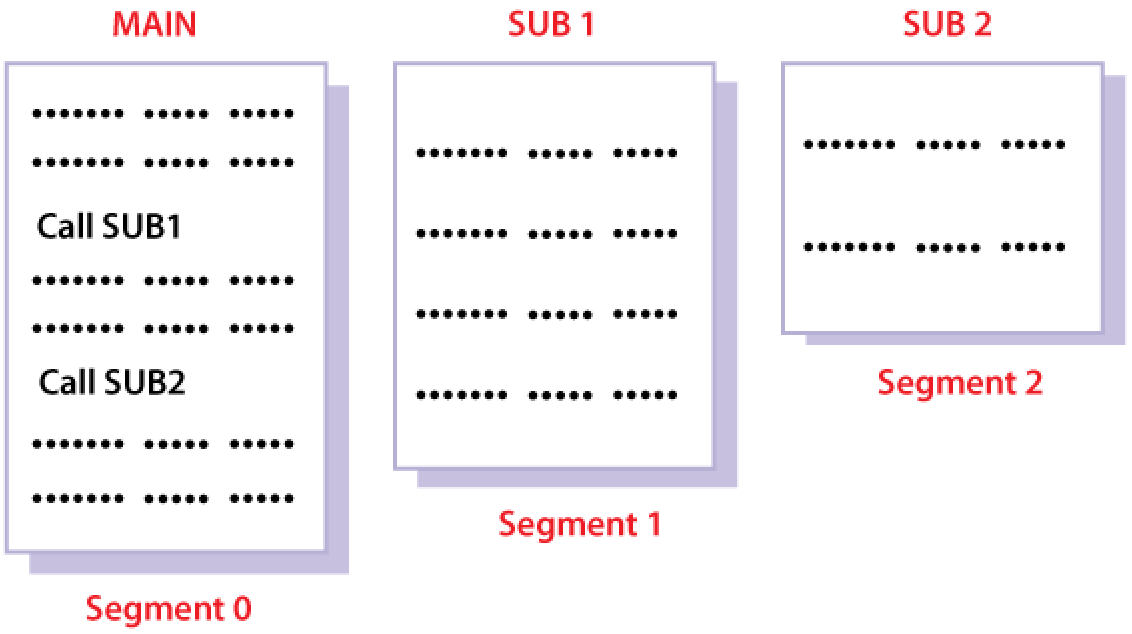
1. Base: It is the base address of the segment
2. Limit: It is the length of the segment.

Why Segmentation is required?

Till now, we were using Paging as our main memory management technique. Paging is more close to the Operating system rather than the User. It divides all the processes into the form of pages regardless of the fact that a process can have some relative parts of functions which need to be loaded in the same page.

Operating system doesn't care about the User's view of the process. It may divide the same function into different pages and those pages may or may not be loaded at the same time into the memory. It decreases the efficiency of the system.

It is better to have segmentation which divides the process into the segments. Each segment contains the same type of functions such as the main function can be included in one segment and the library functions can be included in the other segment.



Translation of Logical address into physical address by segment table

CPU generates a logical address which contains two parts:

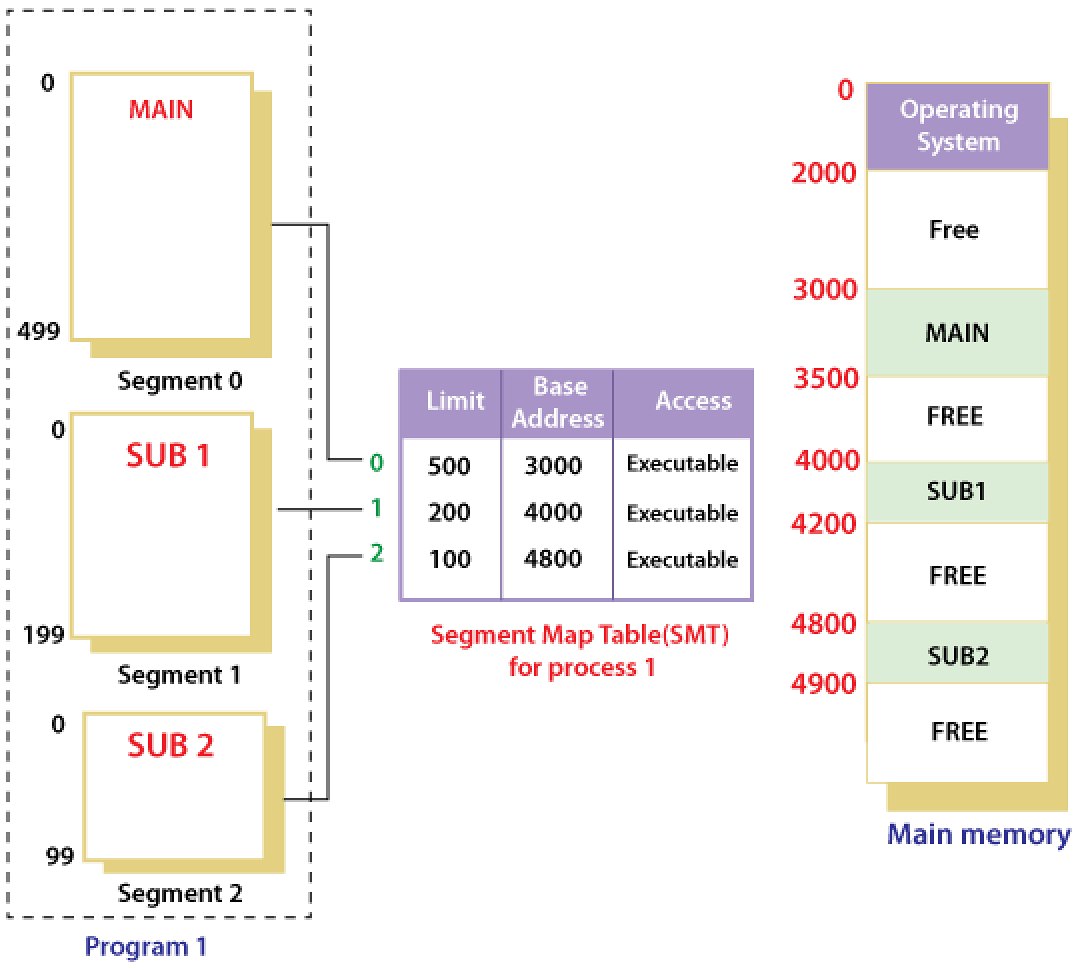
1. Segment Number
2. Offset

For Example:

Suppose a 16 bit address is used with 4 bits for the segment number and 12 bits for the segment offset so the maximum segment size is 4096 and the maximum number of segments that can be refereed is 16.

When a program is loaded into memory, the segmentation system tries to locate space that is large enough to hold the first segment of the process, space information is obtained from the free list maintained by memory manager. Then it tries to locate space for other segments. Once adequate space is located for all the segments, it loads them into their respective areas.

The operating system also generates a segment map table for each program.



Limit	Base Address	Access
500	3000	Executable
200	4000	Executable
100	4800	Executable

With the help of segment map tables and hardware assistance, the operating system can easily translate a logical address into physical address on execution of a program.

The **Segment number** is mapped to the segment table. The limit of the respective segment is compared with the offset. If the offset is less than the limit then the address is valid otherwise it throws an error as the address is invalid.

In the case of valid addresses, the base address of the segment is added to the offset to get the physical address of the actual word in the main memory.

The above figure shows how address translation is done in case of segmentation.

Advantages of Segmentation

1. No internal fragmentation
2. Average Segment Size is larger than the actual page size.
3. Less overhead

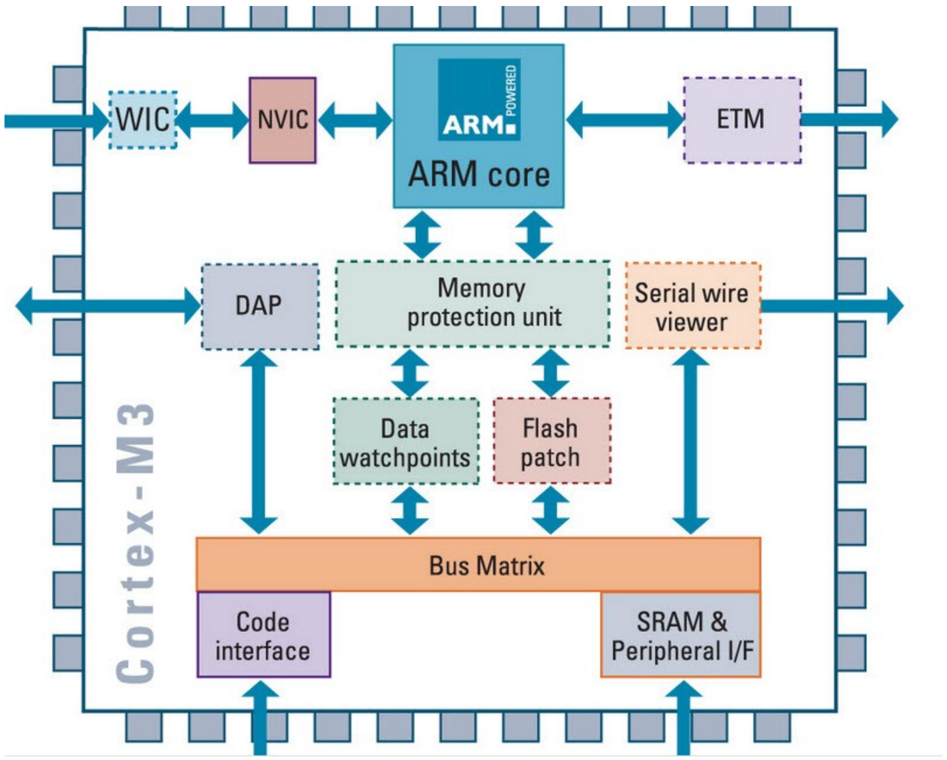
4. It is easier to relocate segments than entire address space.
5. The segment table is of lesser size as compared to the page table in paging.

Disadvantages

1. It can have external fragmentation.
2. it is difficult to allocate contiguous memory to variable sized partition.
3. Costly memory management algorithms.

ARM Architecture

The ARM architecture processor is an advanced reduced instruction set computing [RISC] machine and it's a 32bit reduced instruction set computer (RISC) microcontroller. It was introduced by the Acron computer organization in 1987. This ARM is a family of microcontroller developed by makers like ST Microelectronics,Motorola, and so on. The ARM architecture comes with totally different versions like ARMv1, ARMv2, etc., and, each one has its own advantage and disadvantages.



The ARM cortex is a complicated microcontroller within the ARM family that has ARMv7 design. There are 3 subfamilies within the ARM cortex family :

- ARM Cortex Ax-series
- ARM-Cortex Rx-series
- ARM-Cortex Mx-series

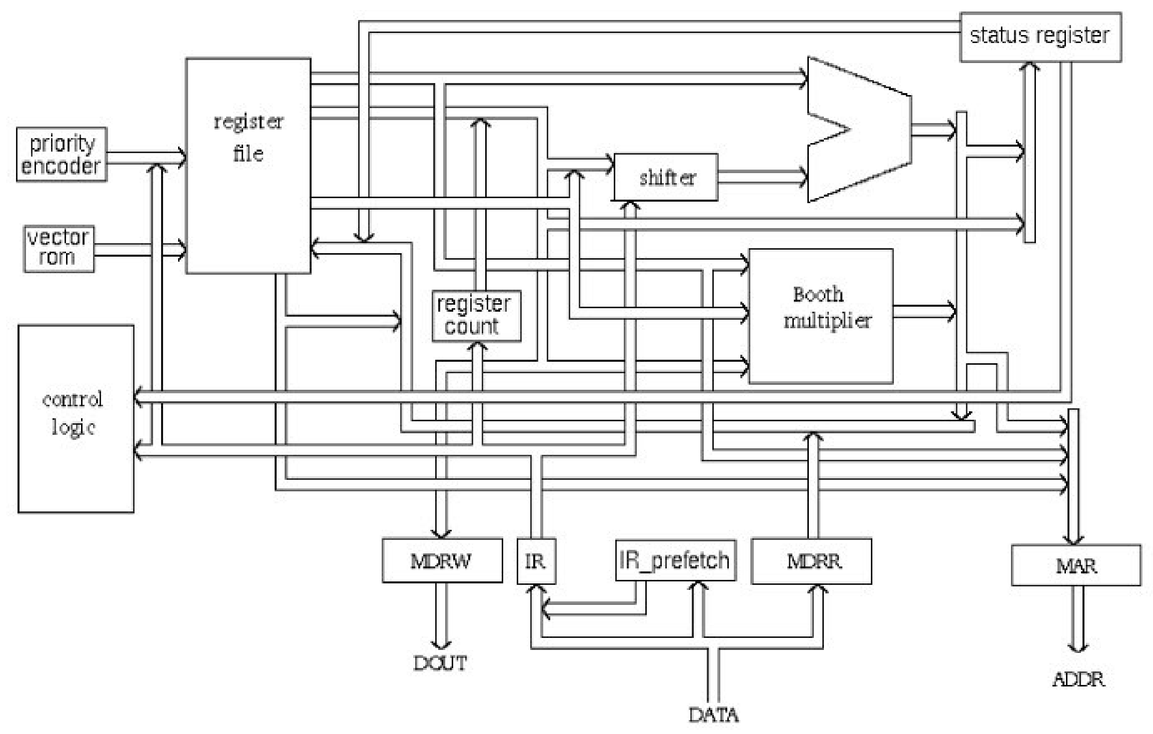
The ARM Architecture

- Arithmetic Logic Unit
- Booth multiplier
- Barrel shifter
- Control unit
- Register file

The ARM processor conjointly has other components like the Program status register, which contains the processor flags (Z, S, V and C). The modes bits conjointly exist within the program standing register, in addition to the interrupt and quick interrupt disable bits; Some special registers: Some registers are used like the instruction, memory data read and write registers and memory address register.

Priority encoder: The encoder is used in the multiple load and store instruction to point which register within the register file to be loaded or kept .

Multiplexers: several multiplexers are accustomed to the management operation of the processor buses. Because of the restricted project time, we tend to implement these components in a very behavioral model. Each component is described with an entity. Every entity has its own architecture, which can be optimized for certain necessities depending on its application. This creates the design easier to construct and maintain.



Operating System - 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.

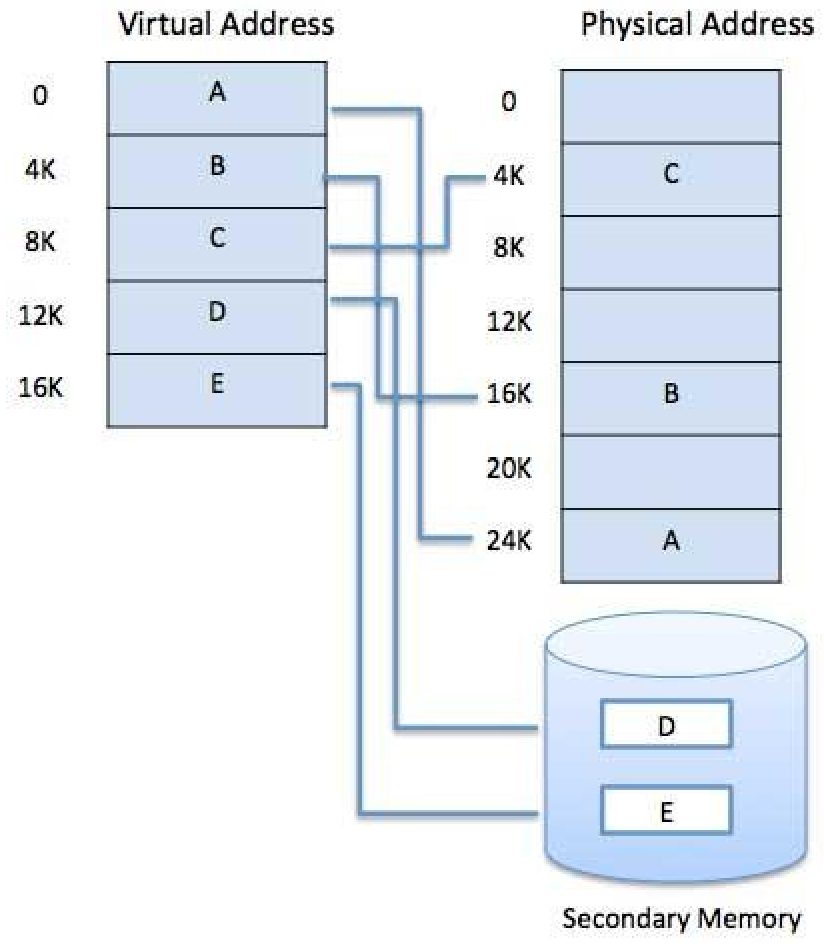
The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical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

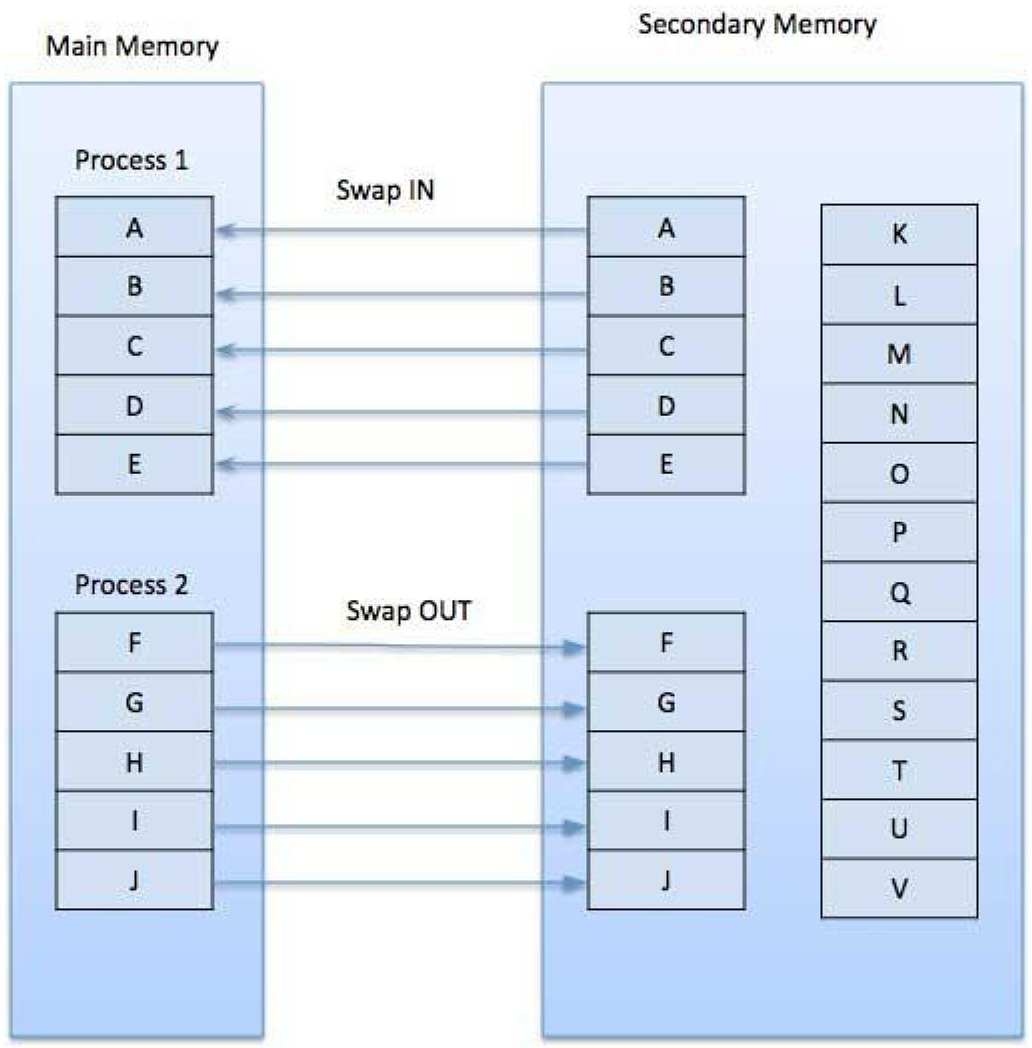
Modern microprocessors intended for general-purpose use, a memory management unit, or MMU, is built into the hardware. The MMU's job is to translate virtual addresses into physical addresses. A basic example is given below −



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.

A page replacement algorithm looks at the limited information about accessing the pages provided by hardware, and tries to select which pages should be replaced to minimize the total number of page misses, while balancing it with the costs of primary storage and processor time of the algorithm itself. There are many different page replacement algorithms. We evaluate an algorithm by running it on a particular string of memory reference and computing the number of page faults,

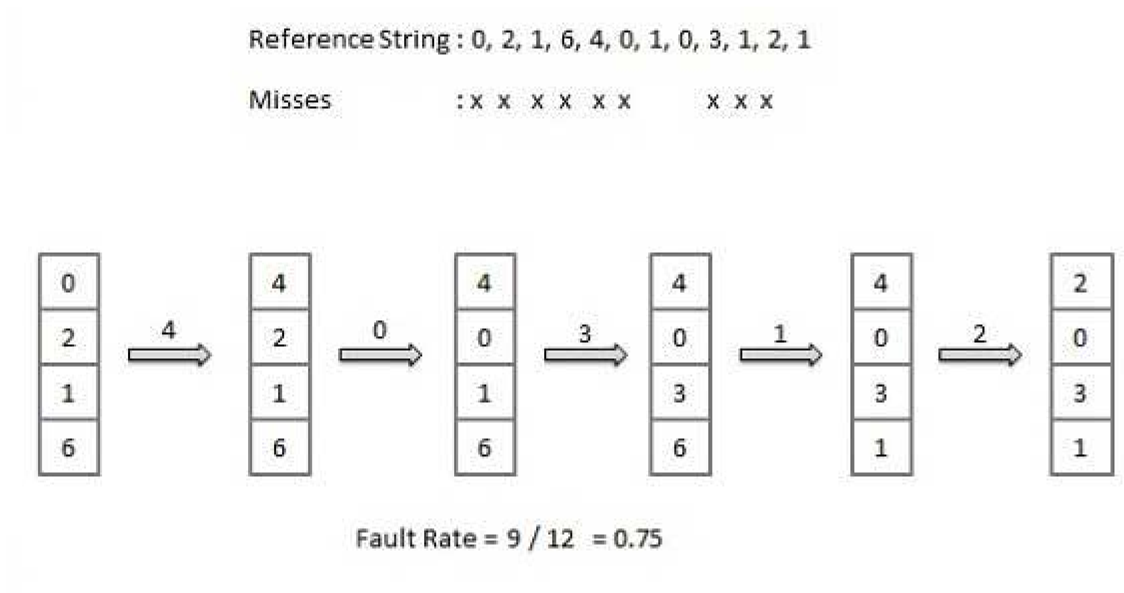
Reference String

The string of memory references is called reference string. Reference strings are generated artificially or by tracing a given system and recording the address of each memory reference. The latter choice produces a large number of data, where we note two things.

- For a given page size, we need to consider only the page number, not the entire address.
- If we have a reference to a page p, then any immediately following references to page p will never cause a page fault. Page p will be in memory after the first reference; the immediately following references will not fault.
- For example, consider the following sequence of addresses − 123,215,600,1234,76,96
- If page size is 100, then the reference string is 1,2,6,12,0,0

First In First Out (FIFO) algorithm

- Oldest page in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages from the tail and add new pages at the head.

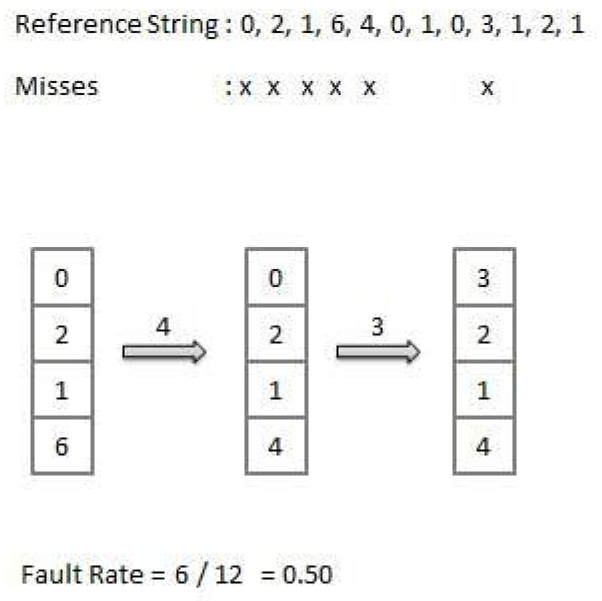

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x x x x

Fault Rate = 9 / 12 = 0.75

Optimal Page algorithm

- An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. An optimal page-replacement algorithm exists, and has been called OPT or MIN.
- Replace the page that will not be used for the longest period of time. Use the time when a page is to be used.

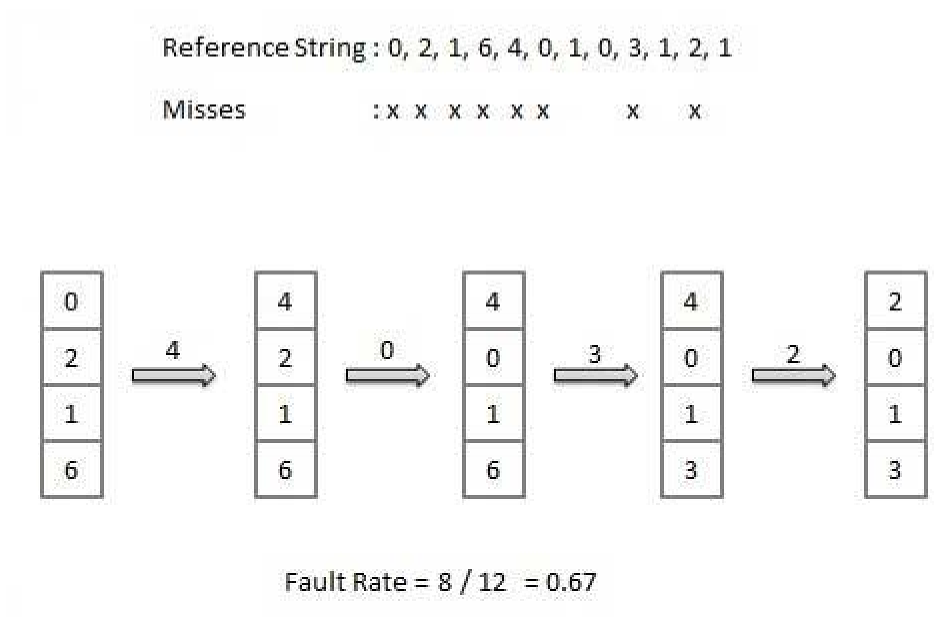

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x

Fault Rate = 6 / 12 = 0.50

Least Recently Used (LRU) algorithm

- Page which has not been used for the longest time in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages by looking back into time.


Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1
Misses : x x x x x x x x
0
2
1
6
4
4
2
1
6
0
4
0
1
6
3
4
0
1
3
2
2
0
1
3
Fault Rate = 8 / 12 = 0.67

Page Buffering algorithm

- To get a process start quickly, keep a pool of free frames.
- On page fault, select a page to be replaced.
- Write the new page in the frame of free pool, mark the page table and restart the process.
- Now write the dirty page out of disk and place the frame holding replaced page in free pool.

Least frequently Used(LFU) algorithm

- The page with the smallest count is the one which will be selected for replacement.
- This algorithm suffers from the situation in which a page is used heavily during the initial phase of a process, but then is never used again.

Most frequently Used(MFU) algorithm

- This algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.