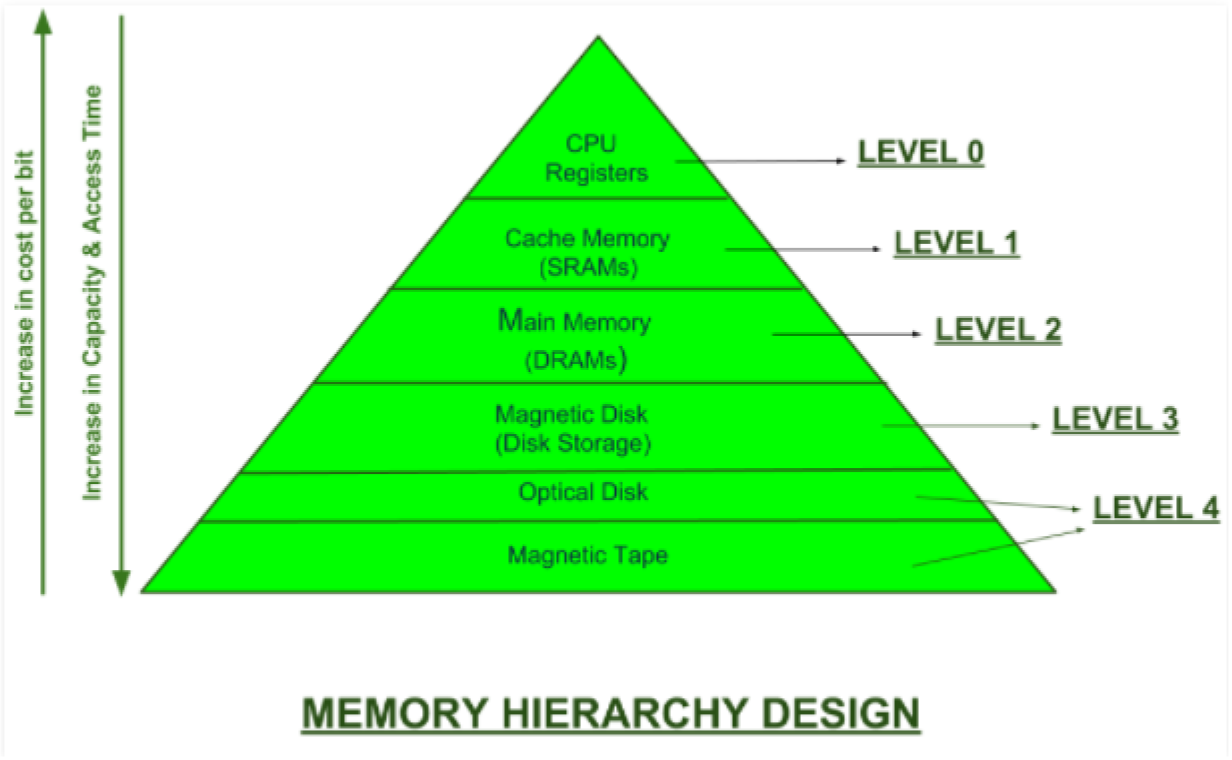
Memory Management

Memory Hierarchy is an enhancement to organize the memory such that it can minimize the access time. Different levels of memory hierarchy :



It is divided into 2 main types:

1. **External Memory or Secondary Memory –**
   Comprising Magnetic Disk, Optical Disk, Magnetic Tape i.e. peripheral storage devices which are accessible by the processor via I/O Module.
2. **Internal Memory or Primary Memory –**
   Comprising of Main Memory, Cache Memory & CPU registers which is directly accessible by the processor.

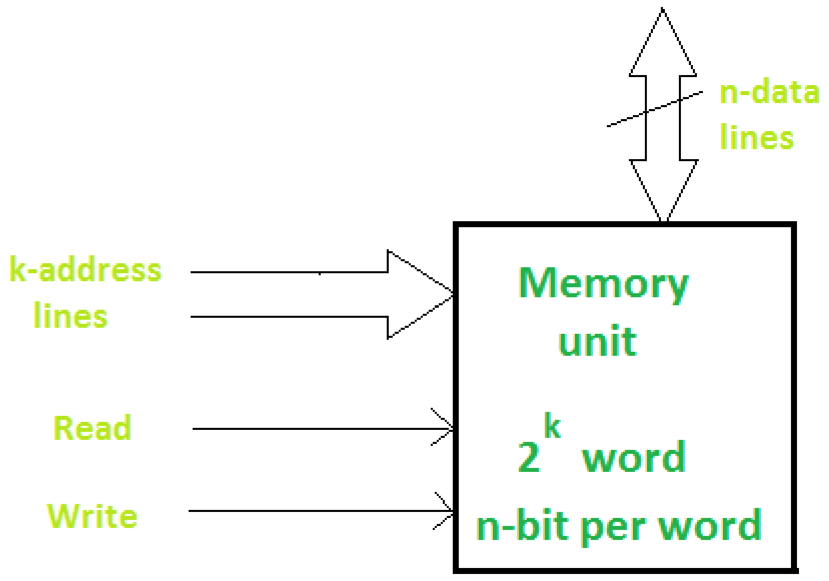
Characteristics of Memory Hierarchy Design from above figure:

1. **Capacity:**
   It is the global volume of information the memory can store. As we move from top to bottom in the Hierarchy, the capacity increases.
2. **Access Time:**
   It is the time interval between the read/write request and the availability of the data. As we move from top to bottom in the Hierarchy, the access time increases.
3. **Performance:**
   Earlier, the computer system was designed without Memory Hierarchy design, the access time was very large resulting in lower performance of the system and thus, enhancement was required which was made in the form of Memory Hierarchy Design.
4. **Cost per bit:**
   As we move from bottom to top in the Hierarchy, the cost per bit increases i.e. Internal Memory is costlier than External Memory.

Terms related to memory:

1. Memories are made up of registers. Each register in the memory is one storage location.
2. **Address**: Identification of memory address.
3. A storage element is called a **Cell**.
4. The data in a memory are stored and retrieved by the process called **writing** and **reading** respectively.
5. **Word:** It is a group of bits where a memory unit stores binary information.
6. **Byte**: It is a word with a group of 8 bits.

A memory unit consists of data lines, address selection lines, and control lines that specify the direction of transfer.Data lines provide the information to be stored in memory. The control inputs specify the direct transfer. The k-address lines specify the word chosen. When there are k address lines, 2^k memory words can be accessed. The block diagram of a memory unit is shown below:



Requirements of memory management are:-

1. Relocation – As the memory is shared among various processes, so swapping the active processes in and out of the main memory enables the operating system to have a larger pool of ready-to-execute processes in which there are chances that it may not get the previous location. Here relocation is required
Within a program, there are memory references in various instructions and these are called logical addresses. After loading of the program into main memory, the processor and the operating system must be able to translate logical addresses into physical addresses.

2. Protection – As there is multiprogramming, every process must be protected against unwanted interference when another process tries to write in a process whether accidental or incidental. There is always a trade-off between relocation and protection requirements.

3. Sharing – It is the task of memory management to allow controlled access to the shared areas of memory without compromising the protection. Mechanisms are used to support relocation and supporting sharing capabilities.

Different Types of RAM (Random Access Memory)

1. Directly accessible by CPU.
2. Used to read and write data.
3. Volatile in nature.

Different types of RAM are:-

SRAM	DRAM
Lower access time, hence faster.	Higher access time, hence slower.
More expensive.	Less expensive.
Requires constant power supply.	Requires less power as information is stored in the capacitor.

Types of DRAM

There are mainly 5 types of DRAM:

1. **Asynchronous DRAM (ADRAM):** The timing of the memory device is controlled asynchronously. A specialized memory controller circuit generates the necessary control signals to control the timing. The CPU must take into account the delay in the response of the memory.
2. **Synchronous DRAM (SDRAM):** These RAM chips' access speed is directly synchronized with the CPU's clock for which the memory chips remain ready for operation when the CPU expects them to be ready. These memories operate at the CPU-memory bus without imposing wait states.
3. **Double-Data-Rate SDRAM (DDR SDRAM):** This faster version of SDRAM performs its operations on both edges of the clock signal; whereas a standard SDRAM performs its operations on the rising edge of the clock signal. Since they transfer data on both edges of the clock, the data transfer rate is

doubled. To access the data at high rate, the memory cells are organized into two groups. Each group is accessed separately.

4. **Rambus DRAM (RDRAM):** The RDRAM provides a very high data transfer rate over a narrow CPU-memory bus. It uses various speedup mechanisms, like synchronous memory interface, caching inside the DRAM chips and very fast signal timing. The Rambus data bus width is 8 or 9 bits.
5. **Cache DRAM (CDRAM):** This memory is a special type DRAM memory with an on-chip cache memory (SRAM) that acts as a high-speed buffer for the main DRAM.

There are two Memory Management Techniques: Contiguous and Non-Contiguous. Contiguous Technique can further be divided into:

1. Fixed (or static) partitioning
2. Variable (or dynamic) partitioning

	STATIC PARTITIONING	**BUDDY SYSTEM**	**DYNAMIC PARTITIONING**
DEFINITION	In this, fixed partitions of either equal or unequal size are made in RAM before execution.	In this, memory management is done in **power of two increments**. Assume the memory size is 2_U, suppose a size of S is required. **If 2_{U-1}<S<=2_U:** Allocate the whole block **Else:** Recursively divide the block equally and test the condition at each	In this, partitions are made during runtime and its size is equal to the size of the process and hence no.of partitions depend on the number of incoming processes and Main Memory's size.

		time, when it satisfies, allocate the block and get out the loop.	
ADVANTAGES	Easy to implement. Little OS overhead.	Easy to implement a buddy system Allocates block of correct size It is easy to merge adjacent holes Fast to allocate memory and deallocating memory.	No internal fragmentation. No restriction on Degree of Multiprogramming No Limitation on the size of the process.
DISADVANTAGES	Internal Fragmentation. External Fragmentation. Limit process size. Reduces degree of multiprogramming.	It requires all allocation units to be powers of two. Internal Fragmentation.	Difficult Implementation. External Fragmentation.

In fixed partitioning, memory blocks are given to a process under some conditions for which different methods are there:

1. First Fit: It scans memory from the beginning and chooses the first available block that is large enough for the process to fit in.

2. Best Fit It searches the entire list of holes to find the smallest hole whose size is greater than or equal to the size of the process.

3. Worst Fit It searches the entire list of holes to find the largest hole and allocate it to process. It is opposite to the best fit algorithm.

4. Next Fit: Next fit is similar to the first fit but it will search for the first sufficient partition from the last allocation point.

Four types of Buddy system:-

1.Binary buddy system

It maintains a list of the free blocks of each size (called a free list), so that it is easy to find a block of the desired size, if one is available. If no block of the requested size is available, Allocate searches for the first nonempty list for blocks of at least the size requested. In either case, a block is removed from the free list.

2.Fibonacci buddy system

This is the system in which blocks are divided into sizes which are fibonacci numbers. It satisfy the following relation:

```
 Zi = Z(i-1)+Z(i-2)
```

3.Weighted buddy system

4.Tertiary buddy system

Coalescing is defined as how quickly adjacent buddies can be combined to form larger segments.

There are 2 methods to implement non-contiguous allocation:

1. Paging
2. Segmentation

In this, os needs to maintain a table known as a page table for each process which contains the base address of the each block which is acquired by the process in memory space.

	LOGICAL ADDRESS	**PHYSICAL ADDRESS**
BASIC	generated by CPU	location in a memory unit
ADDRESS SPACE	It is set of all logical addresses generated by the CPU in reference to the program.	It is a set of all physical addresses mapped to corresponding logical addresses.
VISIBILITY	Users can view the logical address of a program.	Users can never view the physical address of the program.
ACCESS	The user can use the logical address to access the physical address.	The user can indirectly access physical addresses but not directly.

Address binding is the process of mapping from one address space to another address space. It can be done as follows:

1. Compile Time – Symbolic names -> Relocatable address.
2. Link Time – Relocatable address -> Relocatable address.
3. Load Time –> Relocatable address -> Absolute address.
4. Run Time –> Dynamic Linking.

Paging

It is a memory management scheme that eliminates the need for contiguous allocation of physical memory. This scheme permits the physical address space of a process to be non – contiguous.
Example:

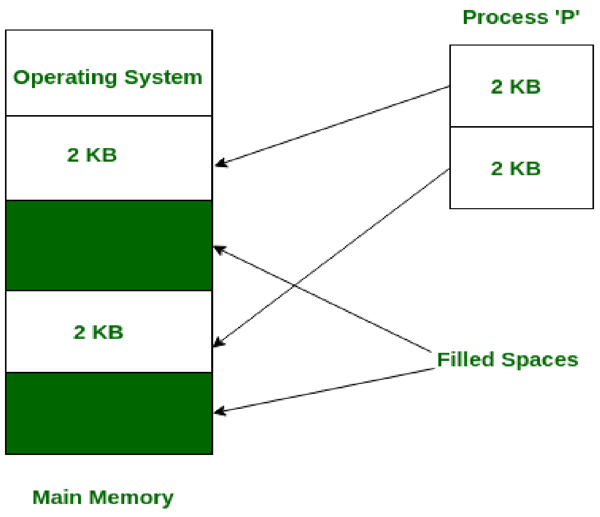
- If Logical Address = 31 bit, then Logical Address Space = 2^31 words = 2 G words (1 G = 2^30)
- If Logical Address Space = 128 M words = 2^7 * 2^20 words, then Logical Address = log2 2^27 = 27 bits
- If Physical Address = 22 bit, then Physical Address Space = 2^22 words = 4 M words (1 M = 2^20)
- If Physical Address Space = 16 M words = 2^4 * 2^20 words, then Physical Address = log2 2^24 = 24 bits

The mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device and this mapping is known as paging technique and to store this mapping virtual memory system use a data structure, known as page table

- The Physical Address Space is conceptually divided into a number of fixed-size blocks, called **frames**.
- The Logical address Space is also splitted into fixed-size blocks, called **pages**.
- Page Size = Frame Size

Let us consider an example:

- Physical Address = 12 bits, then Physical Address Space = 4 K words
- Logical Address = 13 bits, then Logical Address Space = 8 K words
- Page size = frame size = 1 K words (assumption)



In the above diagram, the size of program is 4 kb. So we divide it into 2 pages of equal sizes and the physical memory is also divided into frames of size equal to page size and hence we can allocate it non-contiguously. But this division is a very time consuming process and hence this division is done in advance in secondary memory before reaching the main memory for its execution.

Address generated by CPU is divided into:

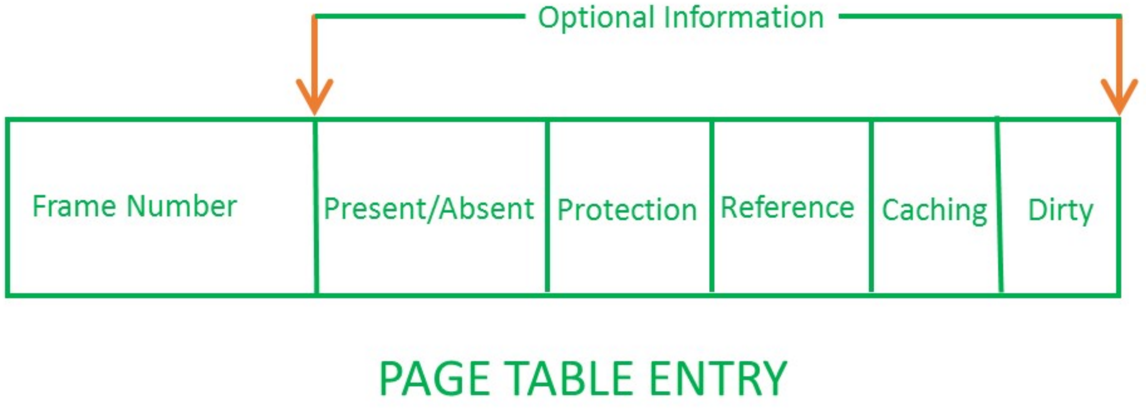
- **Page number(p):** Number of bits required to represent the pages in Logical Address Space or Page number
- **Page offset(d):** Number of bits required to represent a particular word in a page or page size of Logical Address Space or word number of a page or page offset.

Physical Address is divided into:

- **Frame number(f):** Number of bits required to represent the frame of Physical Address Space or Frame number.
- **Frame offset(d):** Number of bits required to represent a particular word in a frame or frame size of Physical Address Space or word number of a frame or frame offset.

Page table has page table entries where each PTE stores a frame number and optional status (like protection) bits. Many of the status bits used in the virtual memory system.

The most important thing in PTE is frame Number.



1. Frame Number

2. Present/Absent bit – It tells whether a particular page you are looking for is present or absent. Sometimes this bit is also known as **valid/invalid** bits.
3. Protection bit – It tells what kind of protection you want on that page. So, these bit for the protection of the page frame (read, write etc).

4. Referenced bit – It tells whether this page has been referred in the last clock cycle or not. It is set to 1 by hardware when the page is accessed.
5. **Caching enabled/disabled –** Consider a situation when user is typing some information from the keyboard which will come into the main memory and hence will be the latest information and when you try to put it in cache again it will show the old information. So whenever freshness is required, we don't want to go for caching or many levels of the memory. That is the reason we want to disable caching. So, this bit **enables or disables** caching of the page.
6. Modified bit – It tells whether the page has been modified or not. If a page is modified, then whenever you should replace that page with some other page, then the

modified information should be kept on the hard disk or it has to be written back or it has to be saved back. Sometimes this modified bit is also called the **Dirty bit**.

The hardware implementation of page tables can be done by using dedicated registers but only if the page table is small else we can use TLB(translation Look-aside buffer), a special, small, fast look up hardware cache.

- The TLB is associative, high speed memory.
- Each entry in TLB consists of two parts: a tag and a value.
- When this memory is used, then an item is compared with all tags simultaneously. If the item is found, then corresponding value is returned.

If page table are kept in main memory,

Effective memory access time = m(for page table + for particular page in page table), where m is main memory access time.

If TLB is used,

Effective memory access time = x*(c+m) + (1-x)(c+m+m)

where m: main memory access time
c : TLB access time
x : TLB hit ratio.

Inverted Page Table

Sometimes the amount of memory occupied by the page tables can turn out to be a huge overhead. An alternate solution to this is to use the concept of inverted page table.
It consists of one-page table entry for every frame of the main memory. Hence PTE reduces to the no of frames in physical memory and a single page table is used to represent the paging information of all the processes.

Through the inverted page table, the overhead of storing an individual page table for every process gets eliminated and only a fixed portion of memory is required to store the paging information of all the processes together. This technique is called inverted paging as the indexing is done with respect to the frame number instead of the logical page number. Each entry in the page table contains the following fields.

- **Page number –** It specifies the page number range of the logical address.
- **Process id –** It acts as an address space identifier and ensures that a virtual page for a particular process is mapped correctly to the corresponding physical frame.
- **Control bits –** These bits are used to store extra paging-related information. These include the valid bit, dirty bit, reference bits, protection and locking information bits.
- **Chained pointer –** When two or more processes share a part of the main memory they map to the same PTE then a chaining pointer is used to map the details of these logical pages to the root page table.

Advantage –

Reduced memory space.

Disadvantage –

Longer lookup time.

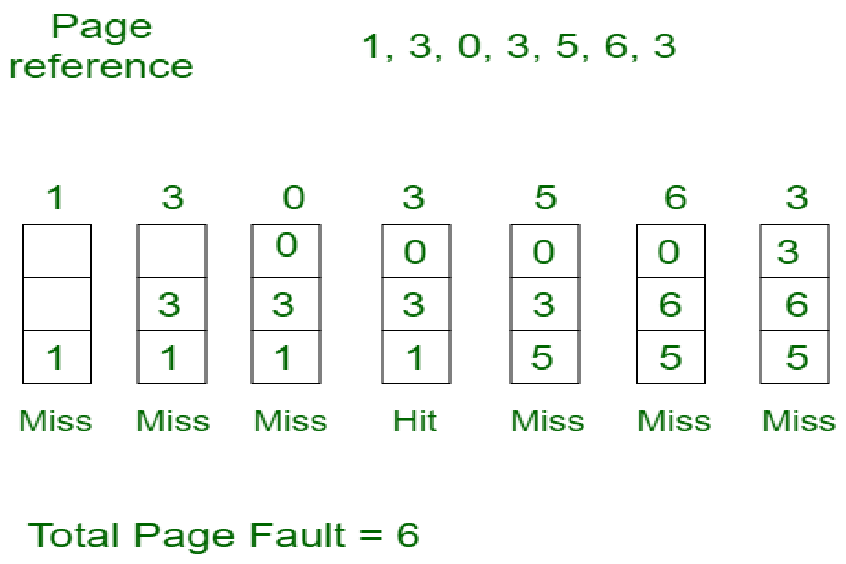
Difficult shared memory implementation.

Page Replacement Algorithms

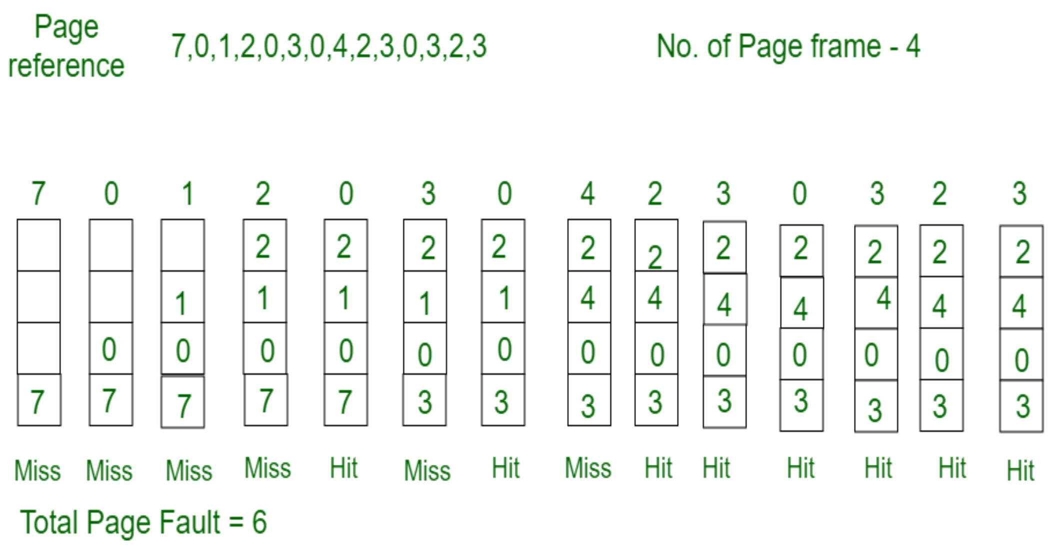
A page replacement algorithm is needed to decide which page needs to be replaced when a new page comes in during paging.
A page fault happens when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.

As the physical memory is much smaller than the virtual memory, so page fault occurs. Hence different page replacement algorithms are there with a single motive to replace a page in such a way that page fault reduces.

First In First Out (FIFO) – It is the simplest algorithm in which os keeps track of all pages in a queue, the oldest page is in front of queue. Whenever a page needs to be replaced the oldest page is selected for replacement.

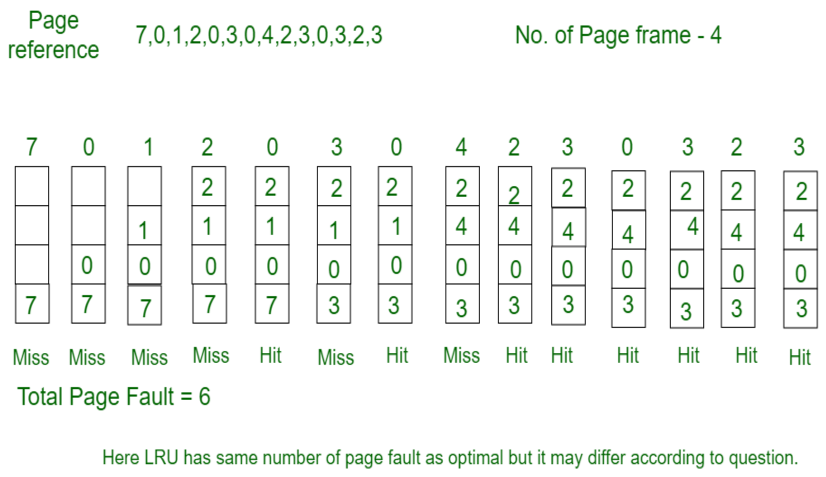


Optimal Page replacement – In this, whenever a page replacement is required the one which would not be used for the longest duration of time is replaced. For ex:



This one is perfect, but not possible in practice as the os can't know future requests. It is used to set up a benchmark so that other replacement algorithms can be analyzed against it.

3. **Least Recently Used –** In this, whenever a page replacement is required the one which is least recently used is replaced. For ex:


Page reference
7,0,1,2,0,3,0,4,2,3,0,3,2,3
No. of Page frame - 4
Miss Miss Miss Miss Hit Miss Hit Miss Hit Hit Hit Hit Hit Hit
Total Page Fault = 6
Here LRU has same number of page fault as optimal but it may differ according to question.

<u>Algorithm for LRU:</u>

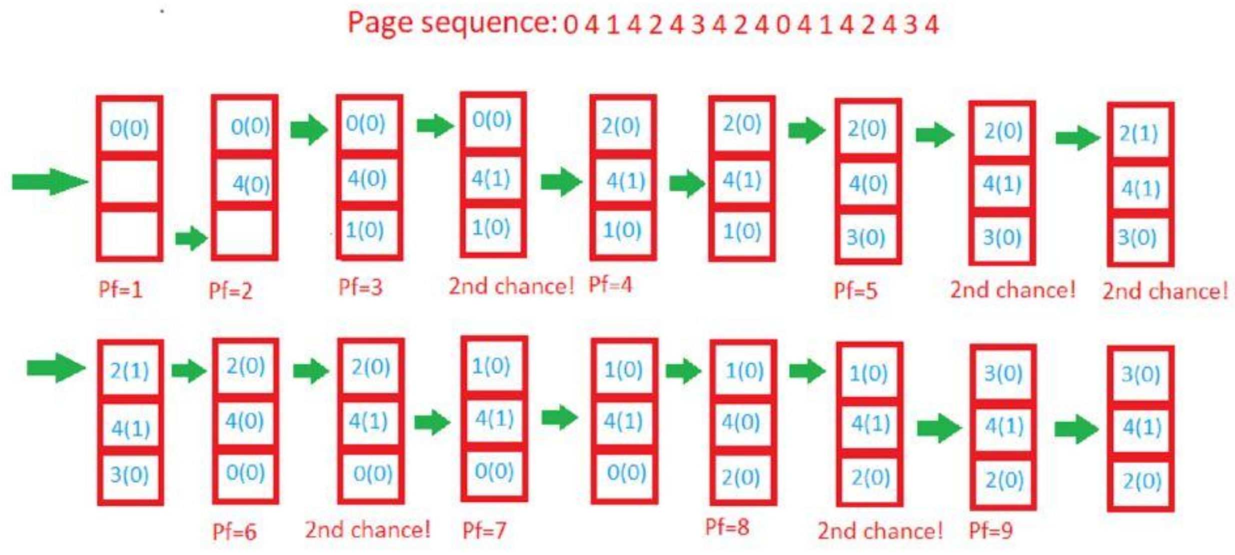
Let **capacity** be the number of pages that memory can hold. Let **set** be the current set of pages in memory.

```
1- Start traversing the pages.
 i) If a set holds less pages than capacity.
   a) Insert page into the set one by one until the size  of the set reaches
capacity or all page requests are processed.
   b) Simultaneously maintain the recent occurred index of each page in a
map called indexes.
   c) Increment page fault

 ii) Else
   If the current page is present in set,
       do nothing.
   Else
      a) Find the page in the set that was least recently used. We find it using
an index array. We basically need to replace the page with a minimum index.
      b) Replace the found page with the current page.
      c) Increment page faults.
      d) Update index of current page.
2. Return page faults.
```

4. Least Frequently Used – In this, whenever a page replacement is required the one which is least frequently used is replaced.

5. Second chance(or clock) policy – In this, a "second chance" bit is given to each memory frame-every time this bit is set to 1, which gives the page a second chance. Thus, a page with the "second chance" bit set to 1 is never replaced during the first consideration and will only be replaced if all the other pages deserve a second chance too. For ex:



Algorithm –

Create an array frames to track the pages currently in memory and another Boolean array second_chance to track whether that page has been accessed since it's last replacement (that is if it deserves a second chance or not) and a variable pointer to track the target for replacement.

1. Start traversing the array arr. If the page already exists, simply set its corresponding element in second_chance to true and return.
2. If the page doesn't exist, check whether the space pointed to by pointer is empty (indicating cache isn't full yet) – if so, we will put the element there and return, else we'll traverse the array arr one by one (cyclically using the value of pointer), marking all corresponding second_chance elements as false, till we find a one that's already false. That is the most suitable page for replacement, so we do so and return.
3. Finally, we report the page fault count.

Demand Paging

The process of loading the page into memory on demand (whenever page fault occurs) is known as demand paging.
In this, if CPU tries to refer a missing page, it generates an interrupt. OS then block the process and to further continue it, os will search the page in LAS which will be brought from LAS to PAS. Page replacement algorithms will be used for replacing the page and the page table will be updated accordingly. Finally, a signal will be sent to the CPU to continue the program execution and it will place the process back into a ready state.

<u>Effective memory access time</u> = p*(s) + (1-p)m

where m: main memory access time
s : page fault access time
p : page fault rate

Swapping

It means removing all of its pages from memory, or marking them so that they will be removed by the normal page replacement process. Suspending a process ensures that it is not runnable while it is swapped out. At some later time, the system swaps back the process from the secondary storage to main memory.

Thrashing

Thrashing is a condition or a situation when the system is spending a major portion of its time in servicing the page faults, but the actual processing done is very negligible.
<u>Causes of Thrashing :</u>

1. High degree of multiprogramming.
2. Lack of frames.

Recovery of Thrashing :

- Don't allow the system to go into thrashing by instructing the long term scheduler not to bring the processes into memory after the threshold.
- If the system is already in thrashing then instruct the mid term scheduler to suspend some of the processes so that we can recover the system from thrashing.

Segmentation

A process is divided into Segments. The chunks that a program is divided into which are not necessarily all of the same sizes are called segments. Segmentation gives the user's view of the process which paging does not give. Here the user’s view is mapped to physical memory.

There are types of segmentation:

1. **Virtual memory segmentation –**
   Each process is divided into a number of segments, not all of which are resident at any one point in time.
2. **Simple segmentation –**
   Each process is divided into a number of segments, all of which are loaded into memory at run time, though not necessarily contiguously.

There is no simple relationship between logical addresses and physical addresses in segmentation. A table stores the information about all such segments and is called Segment Table.

Segment Table – It maps a two-dimensional Logical address into one-dimensional Physical address. It’s each table entry has:

- **Base Address:** It contains the starting physical address where the segments reside in memory.
- **Limit:** It specifies the length of the segment.

Address generated by the CPU is divided into:

- **Segment number (s):** Number of bits required to represent the segment.
- **Segment offset (d):** Number of bits required to represent the size of the segment.

Advantages of Segmentation –

- No Internal fragmentation.
- Segment Table consumes less space in comparison to Page table in paging.

Disadvantage of Segmentation –

- As processes are loaded and removed from the memory, the free memory space is broken into little pieces, causing External fragmentation.

Overlays

The process of transferring a block of program code or other data into internal memory, replacing what is already stored.

So overlay is a technique to run a program that is bigger than the size of the physical memory by keeping only those instructions and data that are needed at any given time.Divide the program into modules in such a way that not all modules need to be in the memory at the same time.

Advantage –

- Reduce memory requirement
- Reduce time requirement

Disadvantage –

- Overlap map must be specified by programmer
- Programmer must know memory requirement
- Overlapped module must be completely disjoint
- Programming design of overlays structure is complex and not possible in all cases

Static Libraries

When a C program is compiled, the compiler generates object code after which the compiler also invokes linker. Linker makes code of library functions (e.g. printf(), scanf(), sqrt(), ..etc) available to your program. A linker can accomplish this task in two ways, by copying the code of the library function to your object code, or by making some arrangements so that the complete code of library functions is not copied, but made available at run-time.

Static Linking and Static Libraries is the result of the linker making a copy of all used library functions to the executable file. Static Linking creates larger binary files, and needs more space on disk and main memory.
Steps to create a static library Let us create and use a Static Library in UNIX or UNIX like OS.

1. Create a C file that contains functions in your library.
2. Create a header file for the library
3. Compile library files.
4. Create a static library.

Step to create a driver program that uses the above created static library.

1. Create a C file with main function
2. Compile the driver program.
3. Link the compiled driver program to the static library.
4. Run the driver program

Dynamic Libraries

Dynamic Linking doesn’t require the code to be copied, it is done by just placing the name of the library in the binary file. The actual linking happens when the program is run, when both the binary file and the library are in memory.