Memory Management

Before starting this first read about what is memory and types of memory:

1. https://www.geeksforgeeks.org/types-computer-memory-ram-rom/
2. https://www.geeksforgeeks.org/introduction-to-memory-and-memory-units/
3. https://www.geeksforgeeks.org/different-types-ram-random-access-memory/

Memory:

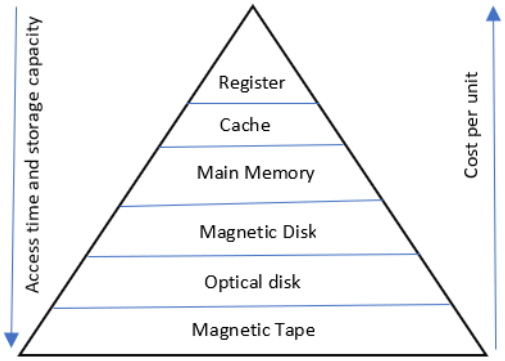
Memory consists of large array of words or bytes, each with its own addresses.
Main memory and registers built into the processor itself are the only repository that the CPU can access directly.

Since each process has a separate memory space. So, we must determine the range of addresses that the process can access. There are two registers that represents the range of legal addresses are base register and limit register.

Base Register: holds the smallest legal physical addresses i.e. starting address.
Limit register: specifies the size of range.
For **example**: base register = 1400
limit register = 1000
then program can legally access all addresses from 1400 through 2400 (inclusive).

Memory hierarchy:

It organizes the computer repository or storage into hierarchy based on response time. This hierarchy shows the all storage devices contained in a computer system.



Storage capacity and access time increases as we go from top to bottom. Cost per unit decreases from top to bottom.

Learn about memory hierarchy design and its characteristic: https://www.geeksforgeeks.org/memory-hierarchy-design-and-its-characteristics/

Need for multiprogramming:

CPU can access Registers, Cache and Main memory directly but using these in large amount can be costly.

CPU utilization:

The Amount of time that we are going to use CPU to run the program.

CPU utilization = 1 - p^n

Where, p = fraction of time that every process waiting for I/O
n = total number of process in the main memory.

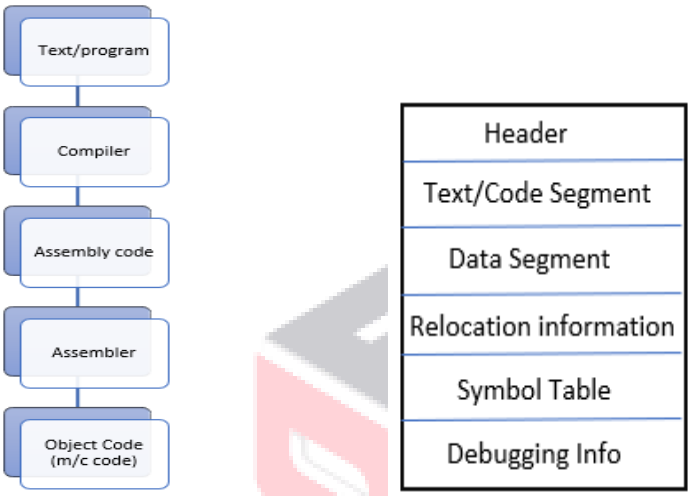
Therefore, CPU utilization depends on the degree of multiprogramming.

Example: Memory = 32mb, process = 4mb, p = 80%
Therefore, 8 processes in memory and each is going to spend 'p' fraction of time.
CPU utilization = 1 - p^n
=> 1 - p^8 = 83%

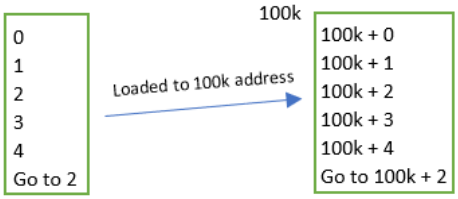
Requirement of multiprogramming: https://www.geeksforgeeks.org/operating-system-requirements-of-memory-management-system/

Object Code, Relocation and linker:

Object code: it is the output of the compiler after it processes source code.



Header: it contains information regarding what are presents in object code. It acts like an index.
Text Segment: it is also called Code segment. It contains set of instructions.
Data Segment: it contains all data and constants.
Relocation: it is a way to map virtual address to physical address.
Definition: https://www.computerhope.com/jargon/d/dynamicr.htm



Original address = 2 (relocatable address)
Final address (after loading into memory/after relocation) = 100k + 2 (absolute address)

If we don't have relocation information, then we are not able to find out where all we have to change the address. The output produced by compiler and assembler contain relocatable address which we have to change before we load into the memory .If we don't change it then process trying to access the code from wrong address(from where it doesn't supposed to access) that will lead to a trap or error.

Symbol table: it is a data structure. Symbol table created and maintained by compiler which contain information regarding various entities like classes, variables, functions, procedure of a program.

main ()	Address 100
printf ()	---------
scanf ()	---------

printf() and scanf() are unresolved functions as address is not known.
Therefore, unresolved symbols should be resolved before loading them in memory i.e. address of these unresolved symbols should be found out. This is mainly done by linker.

Stub:

It is a small piece of code. When program is running then this stub will execute and then, it will requests the operating system regarding the unresolved symbols' code (like printf()). If it is available in memory then operating system give the reference of that or operating system may replace the this stub with the address where unresolved symbol's code (like printf()) is present.

Debugging Info: in this debugging is done to check how the variable is changing.

Linker:

A linker links and combine objects generated by a compiler into a single executable. It generates executable module of a source program.
Responsibilities of linker:

1. Relocation
2. Symbol resolution: means that all external references that must be resolved. Symbol is the only way by which we can refer to another program from our program. So, these symbols must be resolved.

Loader:

Object code is given to the loader and loader knows where this program is going to be loaded. Loader is responsible for loading a program in memory. It is the only program that always reside in main memory.
Functionality of loader:

1. **Program loading**: loader get the program from the hard disk and then load it in main memory so that it will run.
2. **Relocation**: when the program is loaded into the main memory then relocatable address must be converted into absolute address.

Note:

1. Compile time: symbolic names → relocatable address
2. Link time: relocate address → relocatable address
3. Load time: relocatable address → absolute address

Dynamic linking and Shared libraries:
https://www.geeksforgeeks.org/static-vs-dynamic-libraries/

***Logical and physical address space*:**

Before starting, please read this link: https://www.geeksforgeeks.org/memory-management-mapping-virtual-address-physical-addresses/

Difference between logical and physical address: https://www.geeksforgeeks.org/logical-vs-physical-address-in-operating-system/

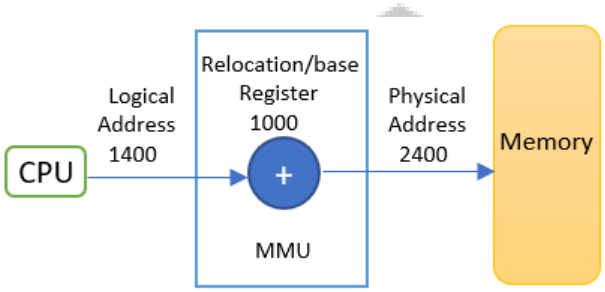
Key points:

1. Logical address:
    - An address generated by the CPU is known as logical address.
    - The set of all addresses generated by a program is known as logical address space.
    - Logical address space = size of process
    - The user program deals with the logical address.

2. Physical address:
    - An address seen by the memory unit (i.e. loaded into the memory address register of the memory) is known as physical address space.
    - The set of all physical addresses corresponding to the logical addresses is physical address space.
    - Physical address space = size of main memory
    - The user program never knows the real physical address.

Memory management Unit:

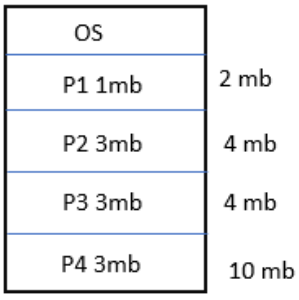
It is a hardware device which is responsible for the run time mapping from logical address to physical address.



Fixed Partitioning:

Before starting, first read this link: https://www.geeksforgeeks.org/fixed-or-static-partitioning-in-operating-system/

Summary:
Fixed partitioning is the simplest technique in which memory is divided into fixed partitions. Partition is fixed but not all partitions are of equal size.



Disadvantages:

1. **Internal Fragmentation**: in fixed partition, we are trying to put a process and if the size of process is smaller than the size of a partition then that partition is going to have some extra free space so we cannot fill that space with any other process. Example: size of process p1 is 1 mb which is smaller than the size of partition (2 mb) so 1 mb space is wasted in that partition.

2. **External fragmentation**: we have total 10 mb total free space (total free space after allocating processes) available but these are not contiguously, space is fragmented into large number of small free space. So, if any other process requires 10mb space we cannot fulfill its requirement.

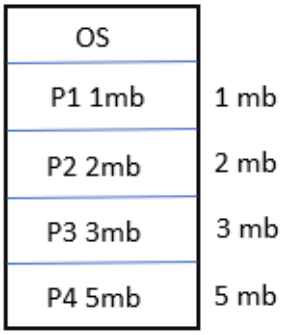
3. **Limitation of process size**: if the process size is greater than 10mb then there is no such partition available that fulfill that process. So, for this type of problem we can use virtual memory.

4. Degree of multiprogramming is fixed.

Dynamic partitioning:

Before starting, first read this link: https://www.geeksforgeeks.org/variable-or-dynamic-partitioning-in-operating-system/

Partition size depends on the process's size. It is also called variable portioning.



Advantages:

1. No Internal fragmentation as partition size is equal to the process size.
2. No memory wastage.
3. Degree of multiprogramming is not fixed.
4. Size of process is not limited by size of partition.

Note:

If there is an internal fragmentation, then external fragmentation is guaranteed but converse is not true (i.e. If there is no internal fragmentation, then we cannot say that there is no external fragmentation)
Suppose, processes p1 and p3 are free then these 1mb and 3mb spaces are free but not contiguous so solution for that is process can be present in parts in partition so for this we can use the concept of paging or compaction.

Compaction:

It is also called defragmentation. It is the solution to the problem of external fragmentation. It is a technique in which free spaces are put together and filled spaces together. The problem with compaction is processes must stop for some time until the compaction process is completed.

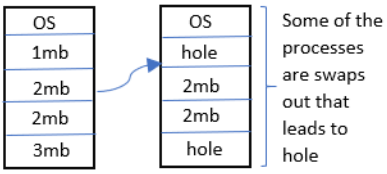
Disadvantage of dynamic partitioning:

Allocation and deallocation of memory is complex.

Hole:

If process occupies some memory and pulled out from memory so it leads to 'hole'. Whenever a process occupied an amount of memory. Whenever the process swaps out from memory and that slot is giving a hole or vacant slot because of that hole management is going to be very difficult. So, managing holes becomes overhead. So, for this

various data structure are used that keep record of what are the holes and what are the free spaces and various algorithms is used to allocate and deallocate the memory.



Bitmap for dynamic partitioning:

Data structures used to keep track of holes and occupied memory are Bitmap and linked list.

1. **Bitmap**:
   Keep track of free spaces and occupied spaces in memory. The main advantage of this approach is simplicity. This approach is also known as bit vector. Since memory is divided into small parts known as allocation units.

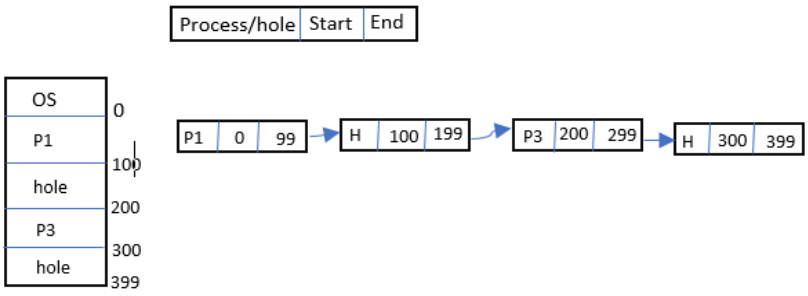
   Example: consider a disk where blocks 2, 3, 4, 5, 8 , 9 are free and the rest of the blocks are allocated and if the block is free, the bit is set to 1 ; if the block is occupied, the bit is 0.
   So, the free space bitmap would be: 001111001100…

   Note:
   Bit map size is depending on the size of allocation unit.
   If the memory is M and then half of the memory is wasted for bitmap.

   Advantage:
   If some processes are swapped out, then only the bits are changed to show hole and it will automatically be concatenated with other hole.

   Disadvantage:
   In order to find out run of holes to allocate to process can take more time.

2. **Linked List**:



   a. Maintaining in increasing order of start address, as if process p3 is free i.e. the space from 200 to 299 is free then we can easily merge holes from hole 100 to hole 300.
   b. Double link list is also beneficial as we can have knowledge of previous node whether it is hole or not.

   Advantage:
   1. Searching is fast.
   2. If any process is created, we can easily add that at end or beginning of link list.

Partition allocation method:

Learn about memory management techniques: https://www.geeksforgeeks.org/operating-system-memory-management-partition-allocation-method/

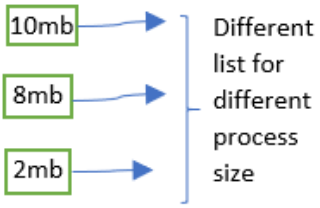
Partition allocation methods are:

1. **First fit:** It start from the beginning, scans the list and it will be found out what are the holes and stop whenever it comes across the first hole which is big enough to hold the process.

2. **Next Fit**: It start from where the first fit algorithm stops, and rest will be same as first fit algorithm. Next fit in detail: https://www.geeksforgeeks.org/program-next-fit-algorithm-memory-management/
3. **Best fit**: it finds the smallest hole which is big enough to accommodate the process. It will not waste lot of space. The problem with best fit approach is, even though if we found out the smallest hole early that is big enough to accommodate the process, we are supposed to search the entire list that is why it is time consuming.

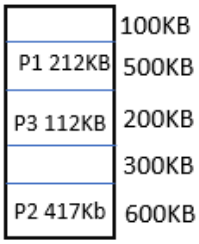
4. **Worst fit:** Allocate the largest hole. We must search the entire list in order to find out the largest hole.

5. **Quick fit**: Maintain a link list which will contain most frequently used process size. In this allocation is quick but deallocation can take lots of time.



Example: Given 5 memory partitions of 100KB, 500KB, 200KB, 300KB, 600KB (in order) how would each of first fit/ best fit/ worst fit algorithms place process of 212KB, 417KB, 112KB, 426KB (in order)
Solution:

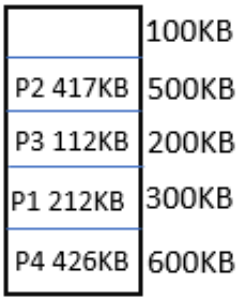
a. First fit:



P1: 212KB
P2: 417KB
P3: 112KB
P4: 426KB
P4 can't fit anywhere.

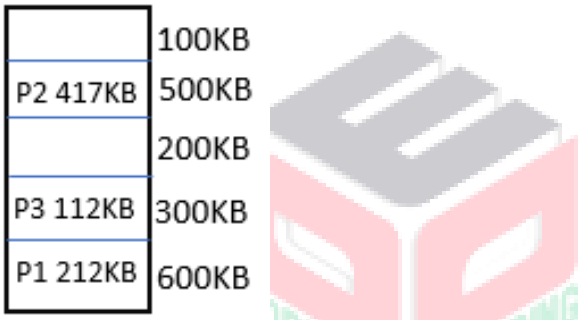
b. Best fit:

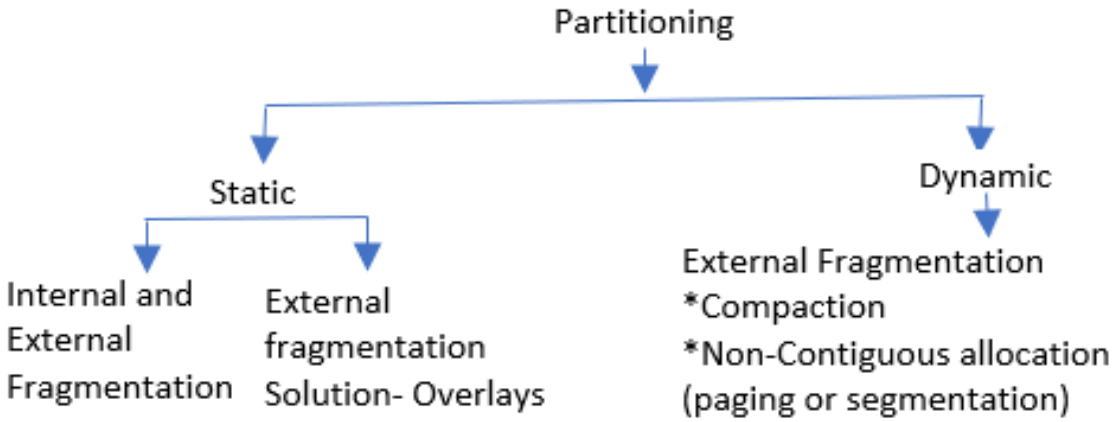
P1: 212KB
P2: 417KB
P3: 112KB
P4: 426KB



Choose smallest partition i.e. large enough to fit the process.

c. **Worst fit**:
P1: 212KB
P2: 417KB
P3: 112KB
P4: 426KB
Scan from start to end and allocate the largest partition i.e. very bigger.
P4 can't fit in any partition.





Overlays:
Overlays in memory management: https://www.geeksforgeeks.org/operating-system-overlays-memory-management/

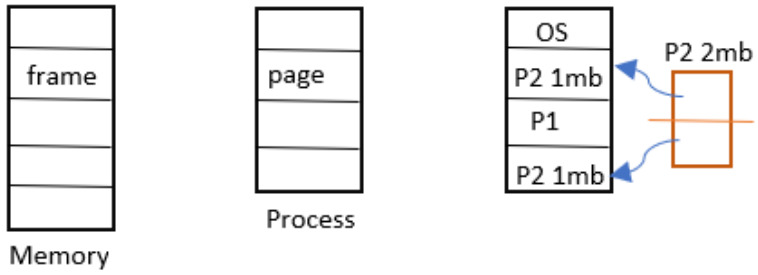
Even though a process is running it will not use the complete program at the same time, it will only be using part of it then overlays concept is used. In overlays, whatever part is required we just load it, once the part is done, we just unload it (pull it back) and get the new part which is required and run it.

Learn about noncontiguous allocation: https://www.geeksforgeeks.org/non-contiguous-allocation-in-operating-system/

Paging:

First learn the basics of paging with the help of diagram: https://www.geeksforgeeks.org/operating-system-paging/

Since internal fragmentation is removed by dynamic allocation but external fragmentation is still available. So paging is an approach through which external fragmentation can be removed.



In paging, process breaks into small parts.

Page: small parts in which process is divided.
Frame: small parts in which memory is divided.

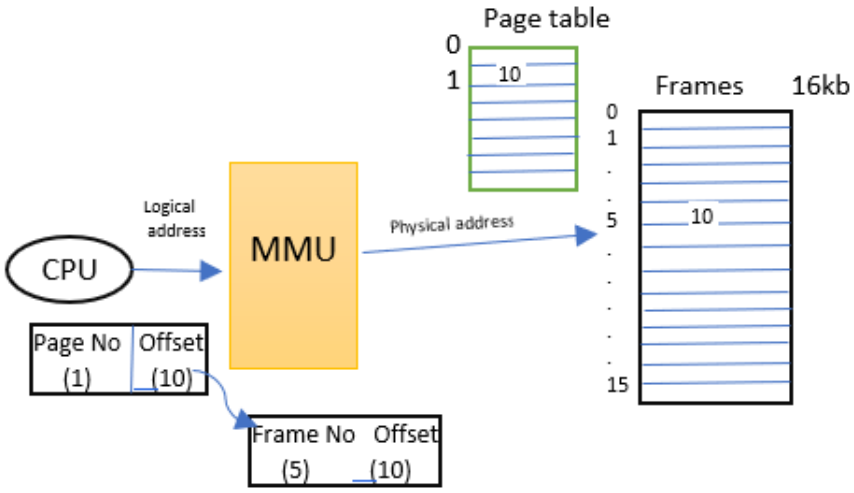
Page is going to fit in frame. Page size and frame size both are same. Whenever a process is executed its pages are loaded into any available memory frames.

Challenge for paging is - how to map address.

Each process has its own page table which contains information regarding where the page present in the main memory frame.

Number of entries present in the page table = Number of pages in the process.

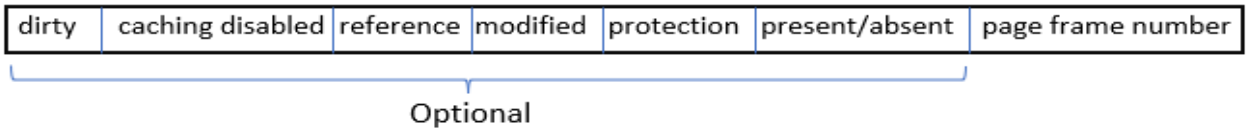
Whenever the CPU generates the logical address that sent to the page table and from the page table, we get the frame number and finally we get physical address and sent to main memory.



Page table entry:

First read this link: https://www.geeksforgeeks.org/operating-system-page-table-entries/

Key points:
Page table contains page table entries where each page table entry shows a frame number and some optional bits. CPU should also know where the page table should place so it uses registers for that.



Caching disabled: Sometimes we need the fresh data so for this caching should be disabled.

Referenced: This bit shows whether the page has been referred. This bit has important role in page replacement algorithm.

Modified: whether the page has been modified or not.
Protection: this is a field which contains may bits. This field shows the what kind of protection we want on page (read mode, write mode, executable mode etc.).

Present/Absent: this bit is important for virtual memory concept/demand paging. This bit shows the page which we are looking for is present or absent or the information about page fault.

Virtual memory: whenever the size of process is larger than the size the main memory then we are going to put only a part of process in main memory and we able to run it. The operating system only gives the illusion of larger main memory.

Page Fault: page fault occurs when the page which we are looking for is not present in main memory**.**
Page frame Number: it gives the frame number in which the current page we are looking for is present in**.**

Multilevel paging:

First explore this link it will give you information about multilevel paging , what is the role of multilevel paging, benefits of multilevel paging - https://www.geeksforgeeks.org/multilevel-paging/

Multilevel paging is a scheme where there exists a hierarchy of page tables.

Need of multilevel paging:
*The size of page table is greater than the size of frame, as a result the page table cannot be stored in a single frame in main memory.

Working:
1. The page table having size greater than the frame size is divided into several parts.
2. The size of each part is same as frame size except possibility of last part.
3. The pages of page table are stored in different frames of the main memory.
4. To keep track of the frames storing the pages of the divided page table, another page table is maintained.
5. As a result, the hierarchy of the page tables gets generated.
6. It is done till level is reached where the entire page table can be stored in a single frame.