Basic of. Memory Management

- Management of main memory is an important issue in design of operating system. Memory management is primarily concerned with allocation of memory to requesting processes. A process can not run before a certain amount of memory is allocated to it.

- In a single process system, main memory is divided into two parts:
  1) One part for operating system
  2) Second part for the program currently being exceccuted.

- Memory management in a single process system is trival. In multiprogramming environment multiple programs reside in the memory at any instance of time. Operating system has to accommodate multiple programs in the limited memory. Memory needs to be allocated efficiently to pack as many processes into memory as possible.

- Memory management can be divided into two classes.
  - Memory management without swapping.
  - memory management with swapping and paging.

- Swapping involves moving of processes between main memory and disk during execution

swapping and paging are necessary as the size of the main memory is never sufficient to fully accommodate all running processes.

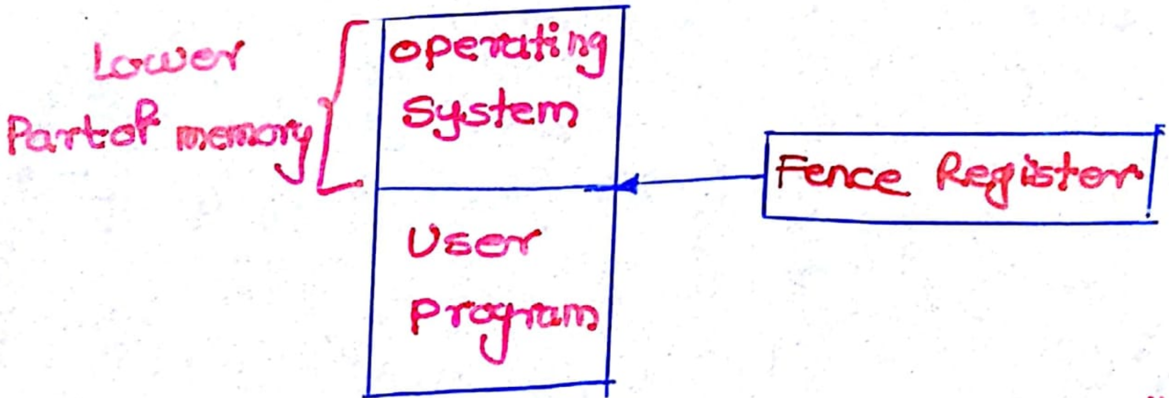
- memory management schemes are required to satisfy the following requirements:

① Relocation ② protection ③ Sharing
④ Logical organization ⑤ physical organization.

Monoprogramming without Swapping

- This is the simplest memory management approach. The memory is divided into two sections

① One part for operating system
② second part for user program.



Memory layout for monoprogramming operating system.

- In this approach, operating system keeps the track of the first and the last location available for allocation of user program.
- operating system is loaded either at the bottom or at the top.

Interrupt vectors are often loaded in low momo
therefore it makes sense to load operating system in low momory.

- sharing of data and code does not make much sense in a single process environment
- operating system can be protected from the user program with the help of the fence register. During the execetion of a program, if the address generated is below the fence. it can be trapped.

(fence- barrier between two areas)

Advantage :

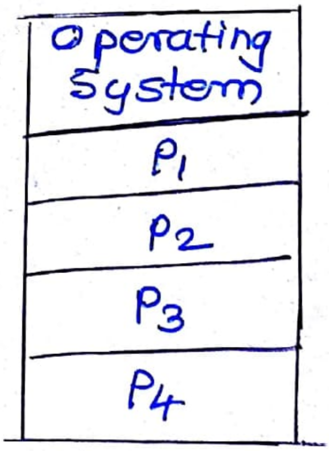
It is a simple management approach.

Disadvantages:

- It does not support multiprogramming.
- There is lower utilization of CPU and memory. CPU sits idle when a running program requires some I/O
- Memory is wasted.

Multiprogramming with Fixed Partitions (without Swapping).

- Memory partitioning scheme with fixed number of partitions was introduced to support multiprogramming. This scheme is based on contiguous allocation (neset to each other)



Fixed -size partitioning

- Memory is partitioned into a fixed number of partitions
- Each partition is of fixed size.
- Each partition can accommodate one program for execution.
- The number of programs (i.e degree of multiprogramming) is bound by the number of partitions.
- As shown in fig memory partitioned into 5 regions. The first region is reserved for operating system the remaining four partitions are for user programs.

Partition Table:

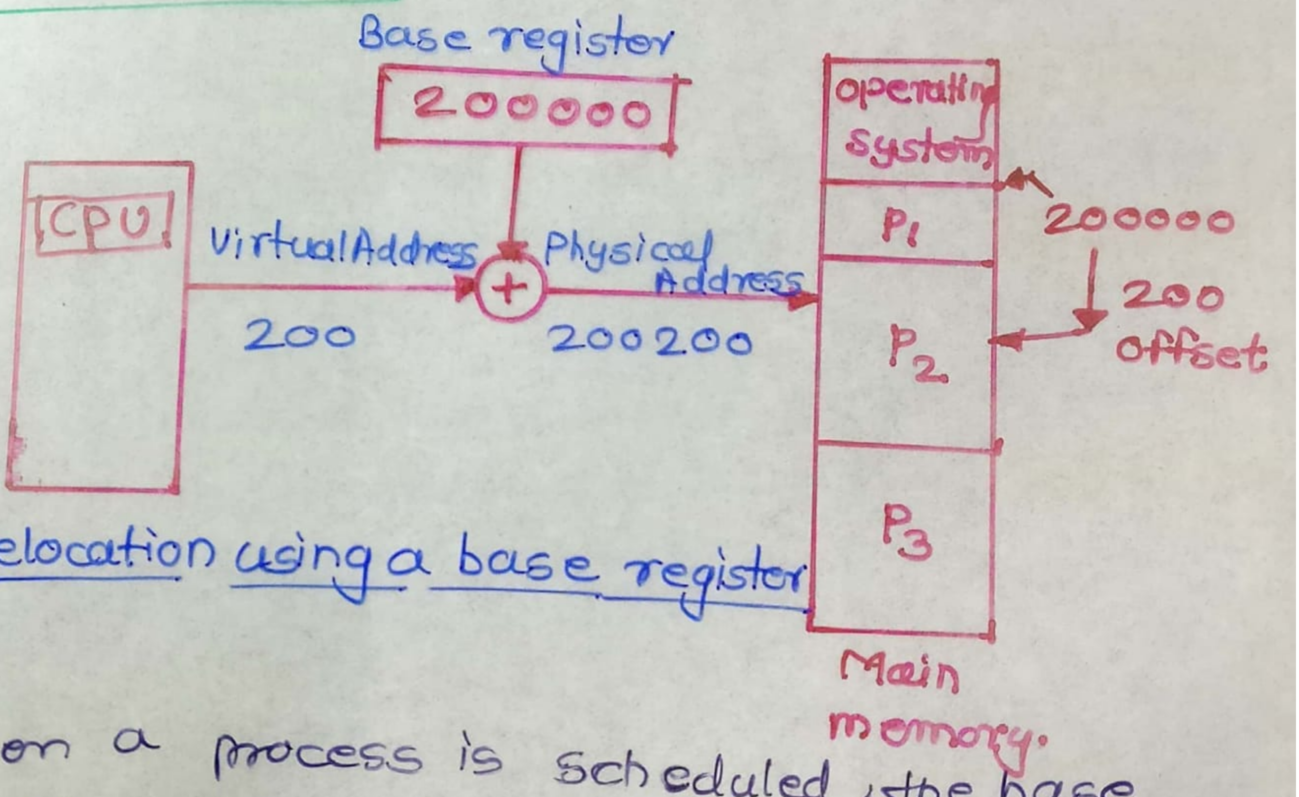
Once partitions are defined, operating system keeps track of status (whether allocated or free) of memory partitions. It is done through data structure called partition table.

	Starting address of partition	Size of partition	status.
1	0K	200K	allocated
2	200K	100K	free
3	300K	150K	free
4	450K	250K	allocated
5	700K	300K	free

A sample partition Table.

Logical versus physical Address

- An address generated by the CPU is commonly referred to as a logical (virtual) address. The address seen by the memory unit (address loaded into MAR) is known as physical Address.
- A logical address can be mapped to physical address by the hardware with the help of base register. This is also known as dynamic relocation of memory reference. This is as shown.



Relocation using a base register

When a process is scheduled, the base register is loaded with the starting address of the partition.

- The physical address of an instruction stored at an offset is given by

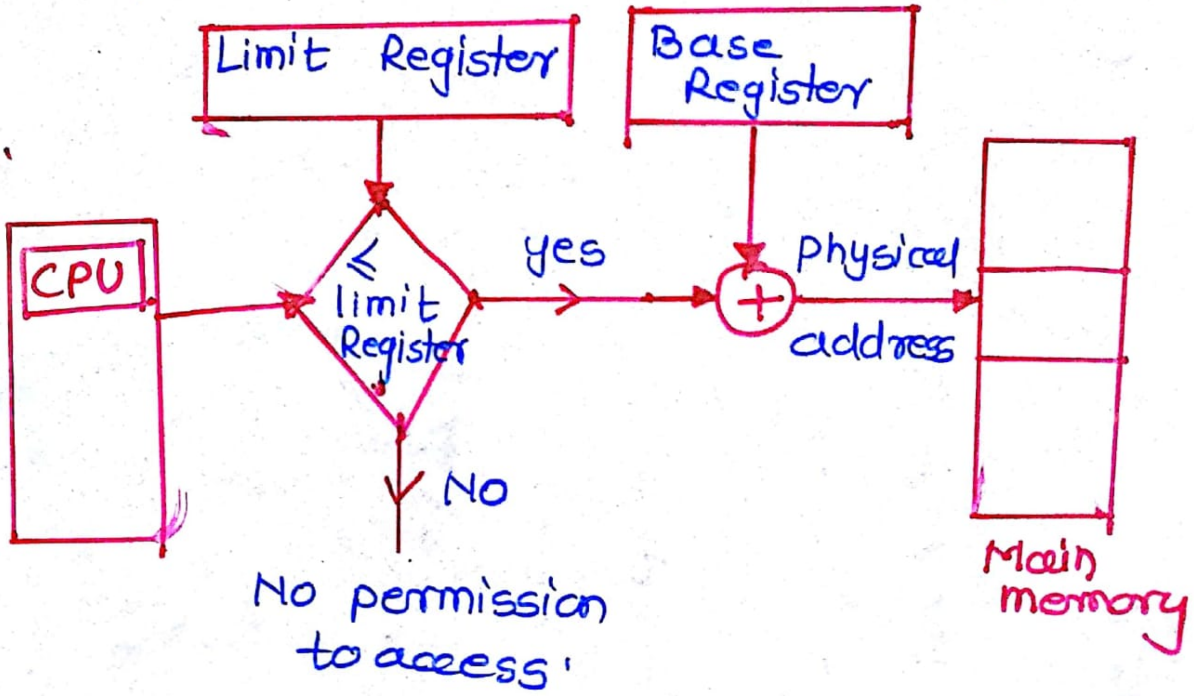
200000 + 200 = 200200

200000 → base register contents.

200 → Virtual (logical or offset) address of the Instruction to be executed.

Protection and sharing

- Protection is necessary in multiprogramming.
- Operating system should be protected from user processes.
- A user process should be protected from other user processes.
- A limit register is used for protection. The primary function of a limit register is to detect attempts to access memory locations beyond the boundaries of the partition. When a process the is scheduled, the limit resistor is loaded with the size of the partition. Each memory access is compared with limit register. If it exceeds the limit register, no permission is given to the user process to access the memory.



Protection through limit register

Placement algorithms

- For execution of a program, it is assigned to a partition. Partition should be large enough to accommodate the program to be executed. After allocation, some memory (at the end) in the partition may remain unused. It is also known as fragmentation.
- The most common strategies to allocate free partition to a program are
  ① First fit ② Best-fit ③ Worst-fit.

First Fit: The first fit strategy allocates the first free partition large enough to the process

Best fit: This startegy allocates the smallest free partition that can accommodate the process

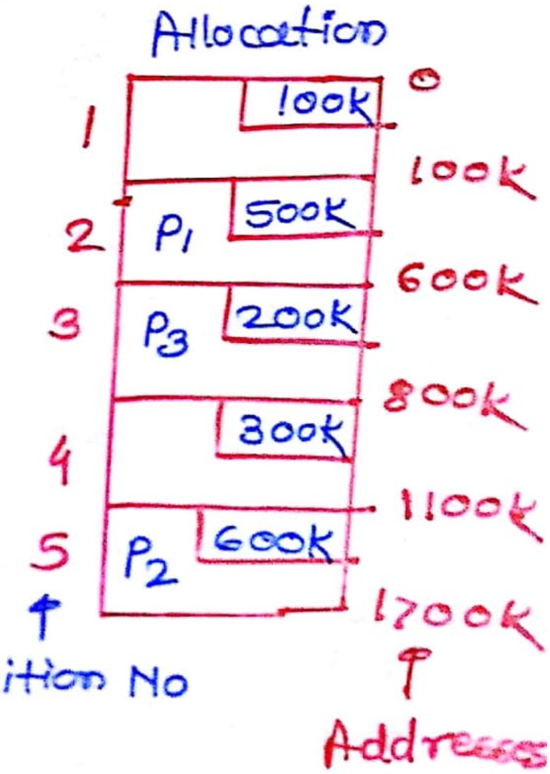
Worst fit: The worst fit strategy allocates the largest free portion that can accomodate the processes.

- All the three strategies have to scan the partition table to find out free partitions.
- First fit terminates after finding the first suitable partition, whereas best-fit and worst fit continue searching the entire partition table.

Example : Given the memory partitions of size 100k, 500k, 200k, 300k and 600k (in order). How would each of the first-fit, best-fit, worst-fit algorithms place the processes of 212k, 417k, 112k and 426k (in order) ? which algorithm makes the most efficient use of memory ?

Soln :

Processes	size
P1	212k
P2	417k
P3	112k
P4	426k



First-Fit

- Partition no 2 of size 500k is assigned to P1 (size = 212k) It is the first position that can accommodate P1
- Partition no 5 of 600k is assigned to P2 (size = 417k) It is the first empty partition that can accommodate P2
- P3 is assigned to Partition 3
- P4 can not be executed.

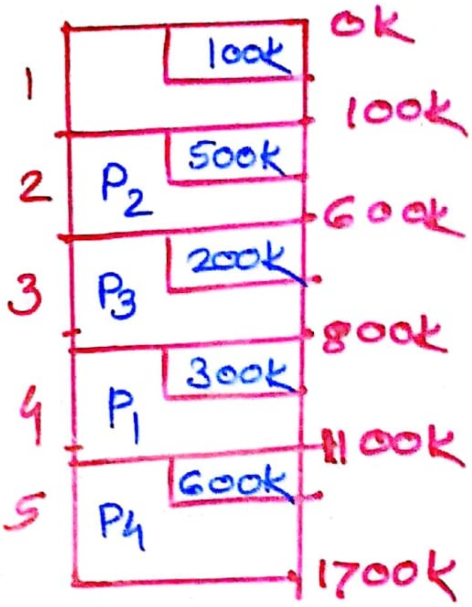
$$\text{memory utilization} = \frac{\text{memory utilized}}{\text{Total memory}}$$

$= \text{memory utilized by } P1, P2 \ \& \ P3 / \text{Total memory}$

$= 212k + 417k + 112k / 1700k = 741/1700$

$= \boxed{0.436}$

Best fit :



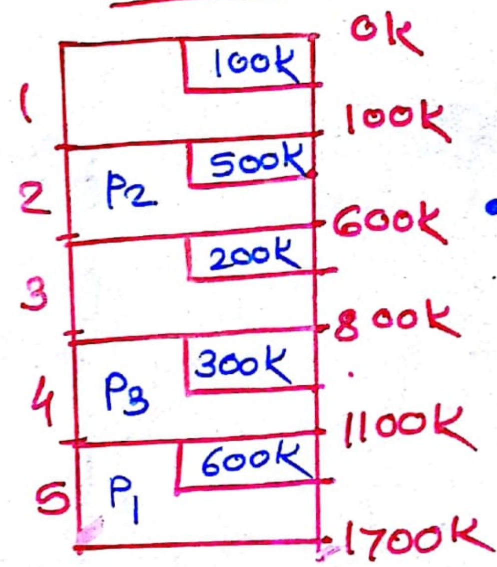
- Partition no 4 of 300k is allocated to P1 (212k) It is the smallest free portion that can accommodate P1
- partition no 2 of 500k is allocated to P2 (417k) It is the smallest free portion that can accommodate P2
- similarly partition no. 3 is allocated to P3 and partition no 5 is allocated to P4

Memory utilization = $\frac{\text{memory utilized by } P_1, P_2, P_3 \& P_4}{\text{Total memory}}$

$= \frac{(212+417+112+426)k}{1700k} = \frac{1167}{1700}$

$= 0.686$

Worst-Fit :



- The largest free partition no 5 of size 600k is allocated to P1 (212k)
- P2 (417k) is assigned to the largest free portion ie Partition 2
- P3 (112k) is assigned to partition no 4 that is largest free partition.
- P4 can not be executed as there is no free partition that can accommodate P4

Memory utilization $= \frac{(212+417+112)k}{1700k}$

$= 0.436$

Disadvantages of fixed Partitioning

- No single program/process may exceed the size of the largest partition in a given system
- It does not support dynamic data structure
- It limits the degree of multiprogramming which in turn may reduce the effectiveness of short term scheduling.
- There is a problem of internal fragmentation.