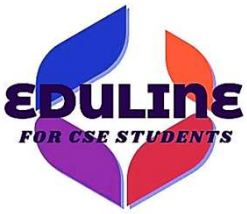
MODULE 4

CHAPTER 1 – MAIN MEMORY MANAGEMENT

CO – Students will be able to discuss main memory and virtual memory management concepts and strategies



Prepared By Mr. EBIN PM, AP

MAIN MEMORY

- Primary memory or main memory is called real storage. Memory can be partitioned. The partition can be rigid manner or dynamic partition.
- Some jobs can be placed in one contiguous location or it can be partitioned up in to a separate blocks and placed in available slots in main memory. All these processes are done by the memory management component.
- In multiprogramming system, main memory is divided in to two parts. One part for the OS and one part for the program currently being executed. The user part of the memory is further subdivided to accommodate multiple processes.

Prepared By Mr. EBIN PM, AP EDULINE 2

STORAGE MANAGEMENT STRATEGIES

❖**FETCH STRATEGY -** It is concerned with when to obtain the next piece of program or data for transfer to main storage from secondary storage. Two types are

➢**Demand Fetch:-** In demand fetch , next piece of program or data is brought in to the main memory when a demand is occurred.

➢**Anticipatory fetch:-** Here the system predicts the programs need and attempt to load the appropriate program and data pieces in to main storage before they are actually needed. When they are requested they will be available and the program may continue without delay. The advantage of demand fetch is that it does not allocate unnecessary programs.

Prepared By Mr. EBIN PM, AP EDULINE 3

❖**PLACEMENT STRATEGY –** It determines where in main memory to place the fetch program or job. Different placement strategies are

➢**First fit :-** The unused or free space in main memory is known as holes. A hole list is provided and it is in the form of linked list. The first fit places the program in the first storage hole which is large enough to hold it.

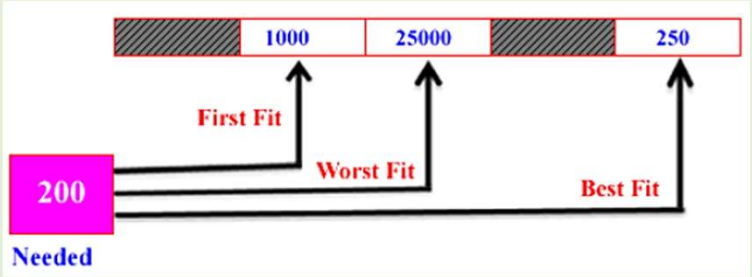
➢**Best fit:-** Best fit places the program in the tightest fitting hole. Here minimum waste of space is occurred.

➢**Worst fit:-** It places the program or data in the largest available hole that will hold it. Here more memory space wastage is occurred.

Prepared By Mr. EBIN PM, AP EDULINE 4

❖REPLACEMENT STRATEGY

- If the available memory is full and no free space for allocating a new job, then for allocating a new job, we must replace one of the processes from main memory.
- That is replacement strategies are concerned with determining which piece of program or data to displace to make room for incoming programs.



Prepared By Mr. EBIN PM, AP EDULINE 5

LOGICAL ADDRESS & PHYSICAL ADDRESS

❖LOGICAL ADDRESS

- It is the virtual address generated by the CPU that can be viewed by the user
- Logical address is generated by the CPU during a program execution
- The logical address is virtual as it does not exist physically. Hence it is also called Virtual address
- This address is used as a reference to access the physical memory location(physical address)

➢**Logical address space:-** set of all logical addresses generated by the CPU in reference to a program is referred as logical address space.

Prepared By Mr. EBIN PM, AP EDULINE 6

❖PHYSICAL ADDRESS

- Physical address is a location in a memory unit.
- The user can never view the physical address of program.
- The user cannot directly access the physical address. Instead, the physical address is accessed by its corresponding logical address by the user

➢**Physical address space:-** set of all physical addresses corresponding to the logical address is called physical address space.

❖MEMORY MANAGEMENT UNIT (MMU)

- The user program generates the logical address. But the program needs physical memory for its execution. Hence the logical address must be mapped to the physical address before they are used.
- MMU is a hardware device
- The logical address is mapped to its corresponding physical address by a hardware device called Memory Management Unit (MMU)

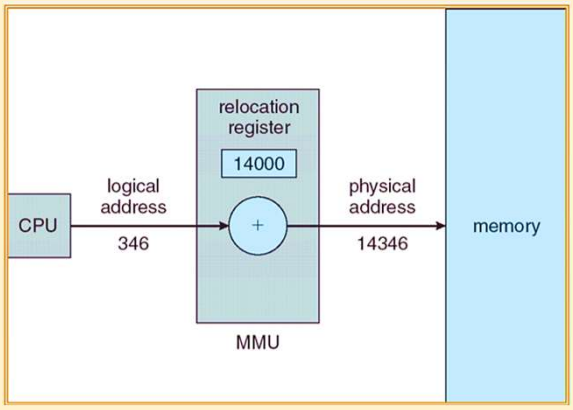
❖ADDRESS BINDING

- Mapping from one address space to another address space is called address binding. Three types of binding are

1. **Compile time binding** 2. **Load time binding** 3. **Run time binding**

- Compile time /load time binding (Logical address=Physical address)
- Runtime binding (Logical address not equal to Physical address)
- The address binding methods used by the MMU generates identical logical and physical addresses during compile time and load time.
- However while run time the address binding methods generate different logical and physical address.

Physical address = Logical address + contents of relocation register

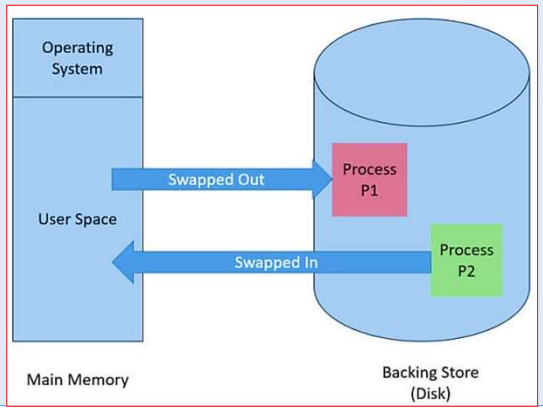


- Relocation register contains the smallest physical address
- Kernel loads relocation register when scheduling a process
- The value in the relocation register is added to every address generated by a user process at the time it is sent to memory

SWAPPING

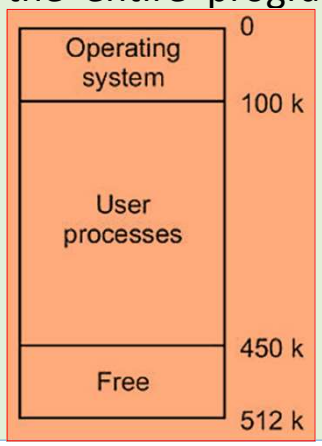
- Swapping is a mechanism in which a process can be swapped temporarily out of main memory or move to secondary storage(disk) and make that memory available to other processes.
- At some later time, the system swaps back the processes from the secondary storage to main memory.
- Though performance is usually affected by swapping process but it helps in running multiple and big processes in parallel
- Swapping may happen in the case of Round Robin scheduling. A process is swapped out when its time quantum finishes and later it is brought in to the memory for continued execution

- This may also happen when it is desired to place a high priority process in the memory. A lower priority process may be swapped out so that higher priority process may be loaded and executed.
- Swapping increases the OS overhead due to the need to perform swap-in and swap-out operation. Windows and UNIX perform swapping

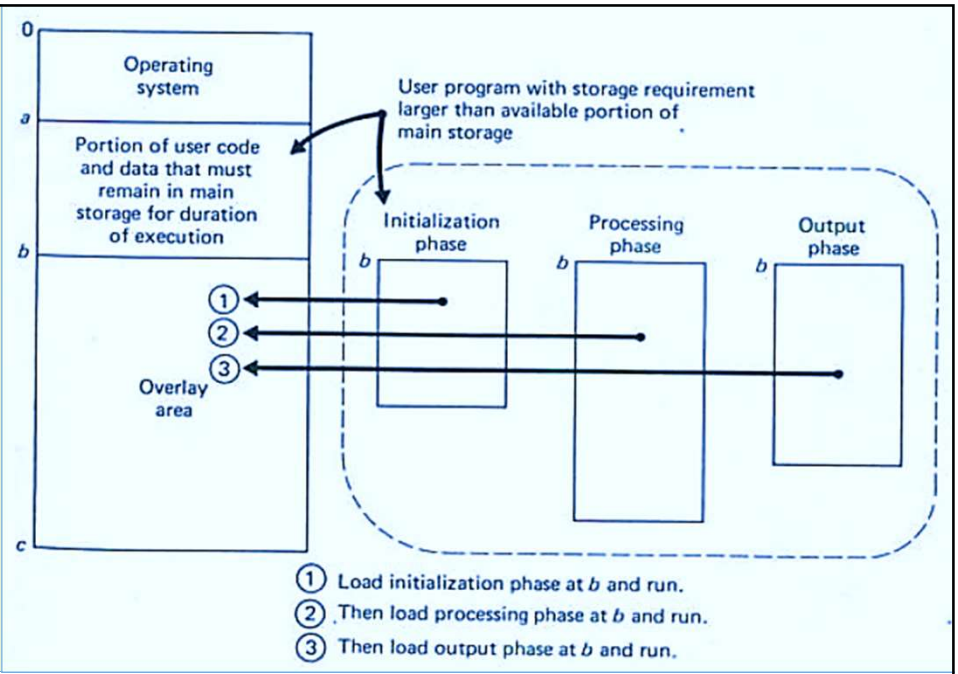


SINGLE USER CONTIGUOUS STORAGE ALLOCATION

- Here the size of the program is small
- The main memory size is limited and the entire program must enter into the main memory
- If the size of the user program is large, then it can also be execute using overlays structure.



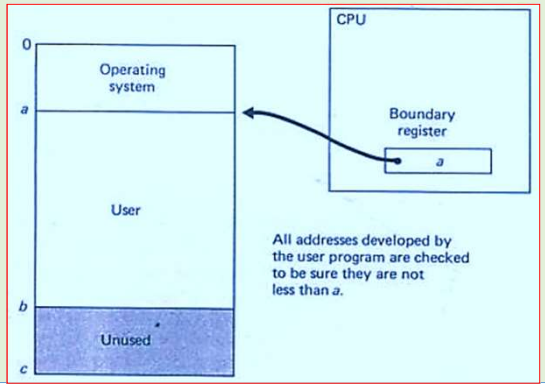
Overlay structure



- When the size of the user program is large, the available program is divided in to segments
- Some portions of user code must remain in main memory which is stored in region a to b
- It stores before the execution started.
- The overlay area is started on address b. so each layer has a switching address b.
- If the execution of one phase is completed, then the next phase is taken for execution.

❖PROTECTION

- The OS must be protected from destruction by the user's program.
- For protection of OS a single boundary register is used into the CPU
- Each time a user program refers to a storage address, the boundary register is checked to be certain that the user is not about to destroy the OS.
- Boundary register contains the highest address used by the OS.



- Here some user program may want some support from OS. In this situation, the generated address refers into the OS area.
- This problem is solved by giving the user specific instruction with which to request services from the OS. That is a supervisor call instruction.
- When a supervisor call is generated, the system is converted into monitor mode. So the user cannot directly interact and halt the system.

MULTIPROGRAMMING

❖Multiprogramming can be implemented in two

1. Multiprogramming with fixed partition
2. Multiprogramming with variable partition

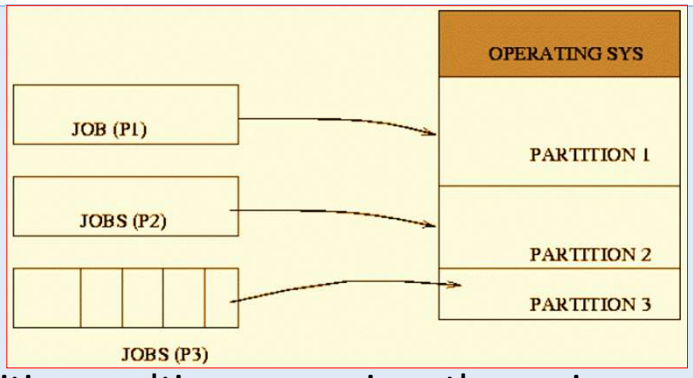
❖Multiprogramming with fixed partition can be divided in to two

1. Fixed partition with absolute addressing and loading
2. Fixed partition with relocatable addressing and loading

❖MULTIPROGRAMMING WITH FIXED PARTITION

1. Absolute addressing and loading

- Single user system wastes a considerable amount of the computing resources. When the CPU bound process come, the CPU time is more consumed.
- When an I/O transfer is needed, the CPU goes to a waiting state. So the CPU utilization is not efficient.
- When an I/O bound process come, in a single user system, CPU is reside in waiting state.

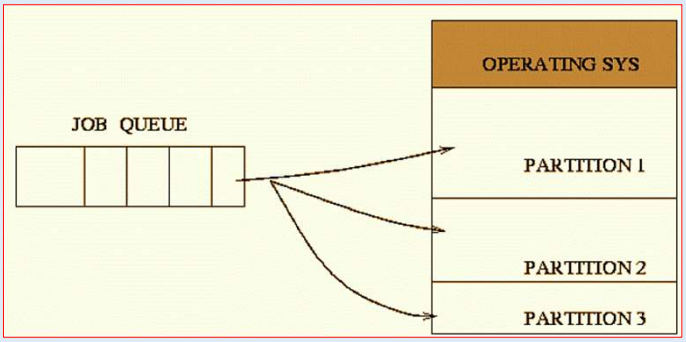


- In fixed partition multiprogramming, the main memory is divided in to fixed size partitions. Each partition has its own corresponding job queue.
- The first job queue is only for first partition. The processes in the first job queue are loaded in to the first partition.

- Here, fragmentation (memory wastage) is occurred. Consider 3 partitions P1, P2 and P3. The job queue of corresponding P1 and P2 are full and P3 has no jobs in the job queue and also P3 has large in size.
- At this time we can execute the jobs in P1 and P2 using the free space of P3.
- But this is not possible because, each job has its own corresponding memory partition and the jobs are only executed in that allocated space.

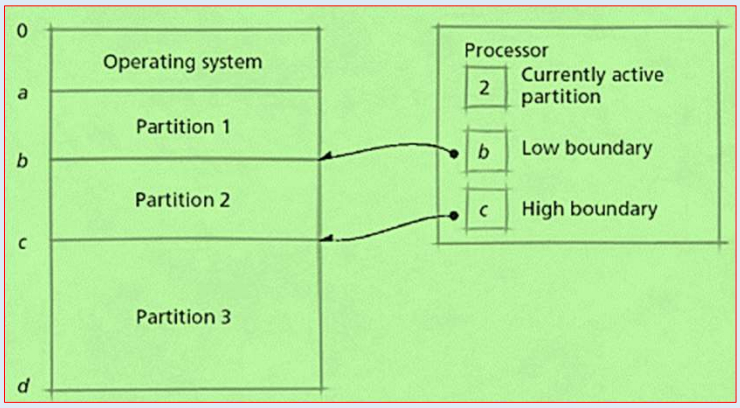
2. Relocatable addressing and loading

- Here memory is divided in to fixed partitions, but each partition does not have its own corresponding job queue.
- Only one job queue is present here. Job is executed that has enough size. The jobs are loaded in to a partition that has a capability of executing the job.
- No memory fragmentation.

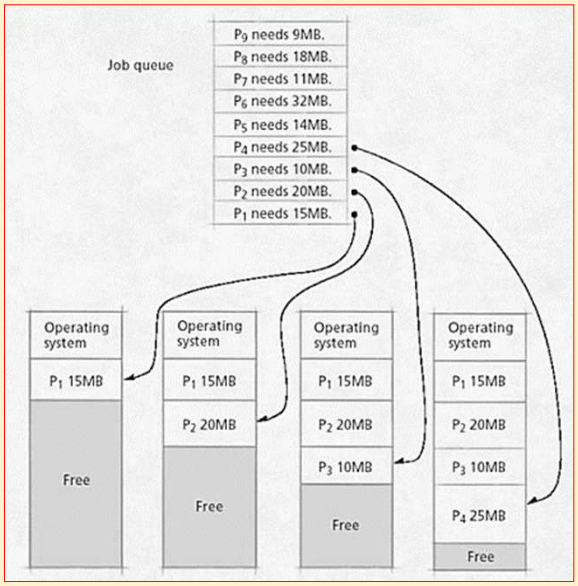


➢Memory Protection In Fixed Partition

- Here, boundary registers(base register and limit register) are used
- While the user in partition 2 is active, all storage addresses developed by the running program are checked to be sure they fall between b and c



❖MULTIPROGRAMMING WITH VARIABLE PARTITION



- Here memory is not fixed. Memory space is allocated according to the requirements, so no memory fixed partition is done.
- External fragmentation is occurred. In variable partition multiprogramming, the waste does not become obvious until jobs start to finish and leave holes in the main storage.
- These holes can be used for other jobs, but even as this happens, the remaining holes get smaller eventually becoming too small to hold new jobs.
- So in variable partition multiprogramming waste does not occur

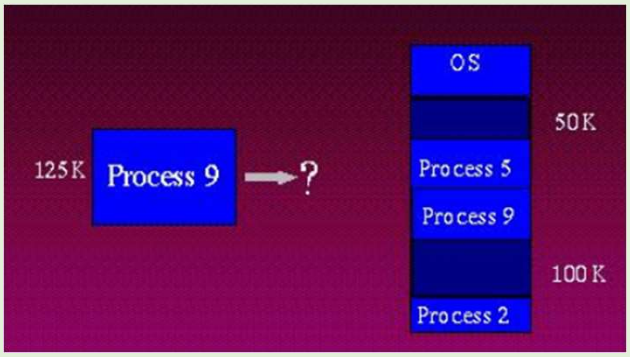
FRAGMENTATION

- As processes are loaded and removed from memory, the free memory space is broken in to little pieces.
- It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is called fragmentation.
- Two types of fragmentations are

1. **External fragmentation**
2. **Internal fragmentation**

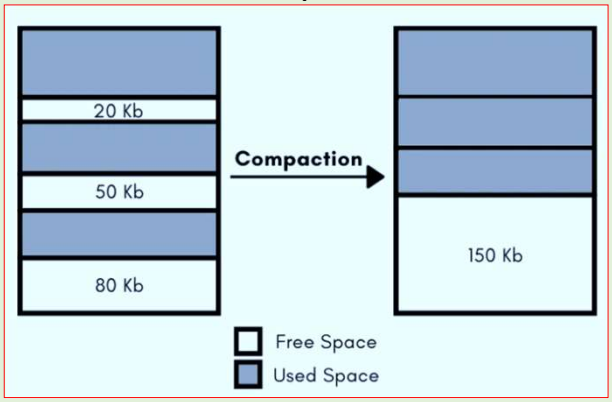
❖EXTERNAL FRAGMENTATION

- External fragmentation is the unused area between two used areas. It is a serious problem. Here memory space to satisfy a request is available, but is not contiguous.



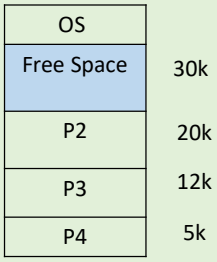
➢Memory compaction

- External fragmentation can be reduced using memory compaction. In memory compaction, we move the used space in upper memory or in lower memory. That is used space is moved in one place and unused are collected in another place.



❖INTERNAL FRAGMENTATION

- Consider the following figure. Suppose a 25K request is coming, then 30K is fully allocated because it is a fixed partition. Here 5k is wasted.
- That is the wasted space contained in a partition that is allocated for a request is called internal fragmentation.
- It is a wasted space with in a partition.
- Internal fragmentation can be reduced effectively assigning the smallest partition but large enough for the process.



Prepared By Mr. EBIN PM, AP EDULINE 29