Memory Management

Main Memory

- Main memory is the most critical resource as the speed of the programs depend on memory.
- Consists of large array of bytes or words each having their own address
- Limited size
- Stores programs and data required by CPU and I/O devices.

- Program must be brought into memory and placed within a process for it to be run.
- *Input queue* – collection of processes on the disk that are waiting to be brought into memory to run the program.
- User programs go through several steps before being run.

Memory Management

- Ideally programmers want memory that is
  - large
  - fast
  - non volatile

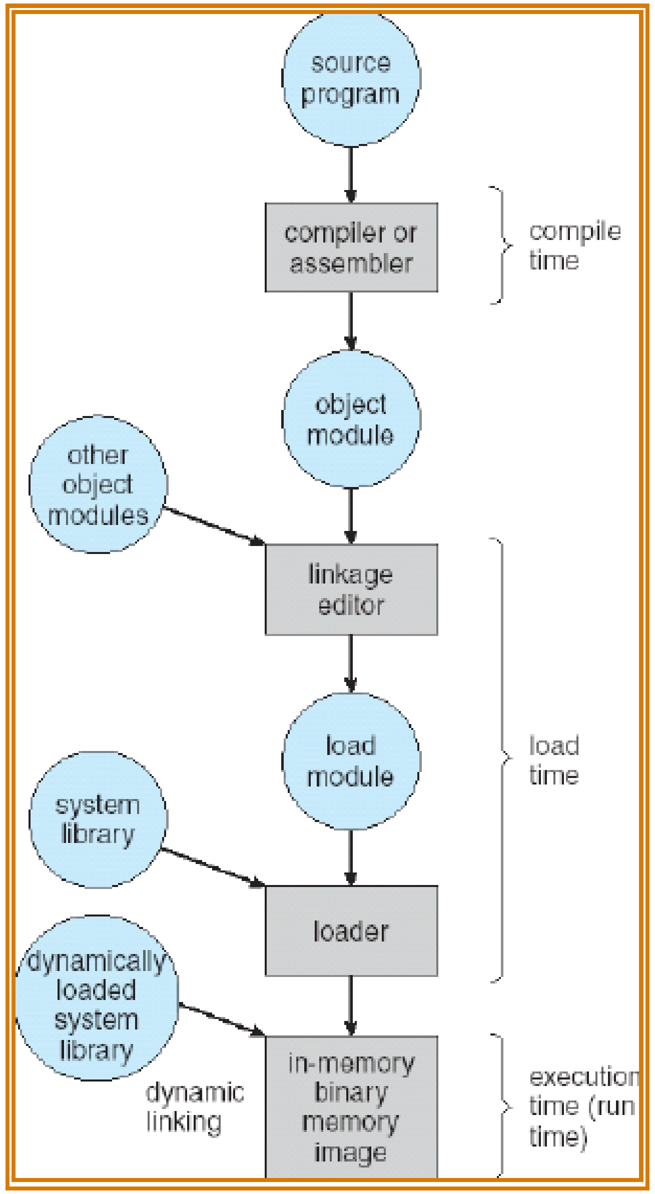
- Memory hierarchy
  - small amount of fast, expensive memory – cache
  - some medium-speed, medium price main memory
  - gigabytes of slow, cheap disk storage

- Memory manager handles the memory hierarchy

Binding of Instructions and Data to Memory

Address binding of instructions and data to memory addresses can happen at three different stages.

- **Compile time**: If memory location known a priori, absolute code can be generated; must recompile code if starting location changes.
- **Load time**: Must generate *relocatable* code if memory location is not known at compile time.
- **Execution time**: Binding delayed until run time if the process can be moved during its execution from one memory segment to another. Need hardware support for address maps (e.g., *base* and *limit registers*).


source program
compiler or assembler
compile time
object module
other object modules
linkage editor
load module
load time
system library
loader
dynamically loaded system library
dynamic linking
in-memory binary memory image
execution time (run time)

Logical vs. Physical Address Space

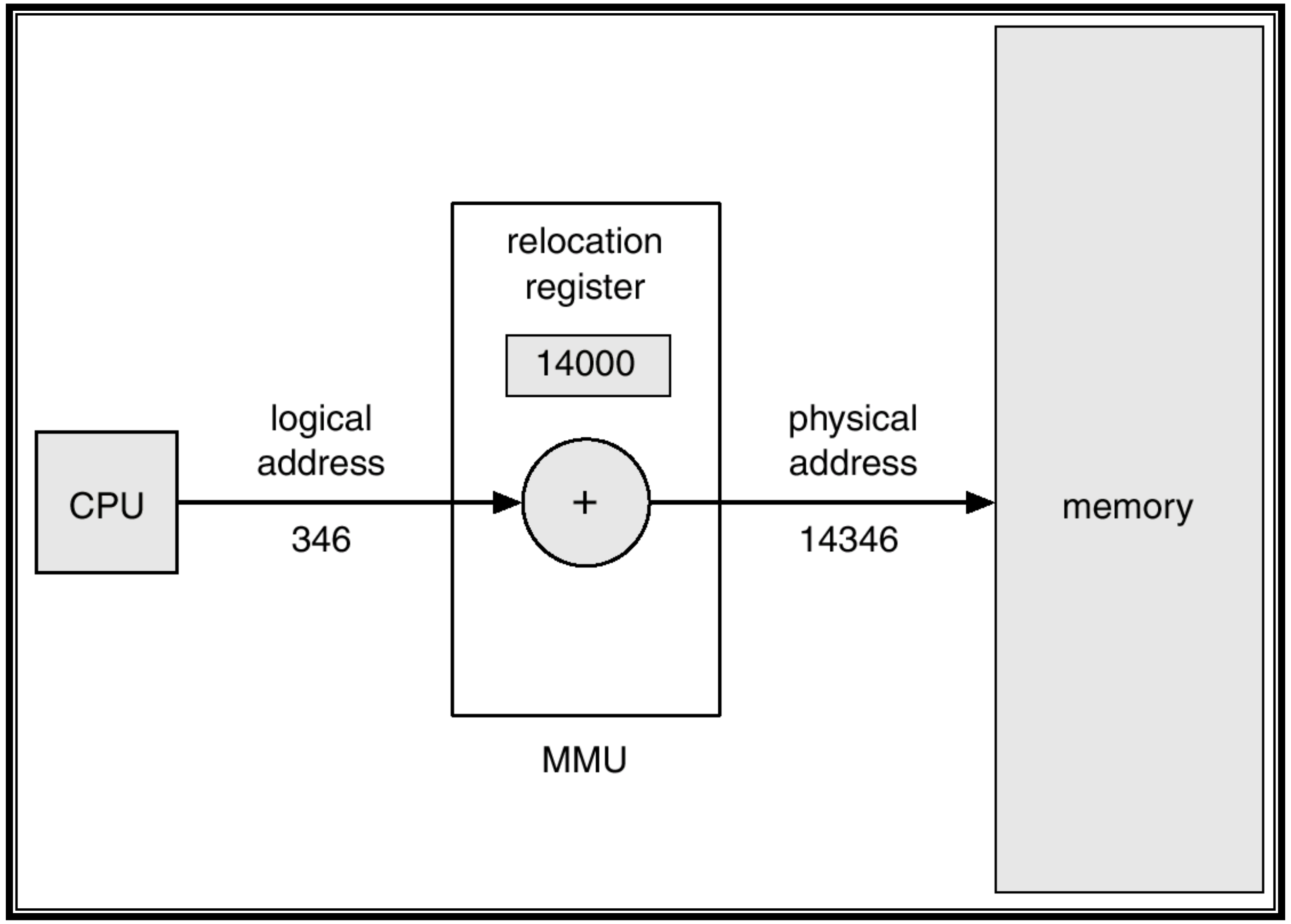
- The concept of a logical *address space* that is bound to a separate *physical address space* is central to proper memory management.
  - *Logical address* – generated by the CPU; also referred to as *virtual address*.
  - *Physical address* – address seen by the memory unit.

- Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme.

Memory-Management Unit (MMU)

- Hardware device that maps virtual to physical address.
- In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory.
- The user program deals with *logical* addresses; it never sees the *real* physical addresses.

Dynamic relocation using a relocation register



- The OS itself permanently occupies a portion of the memory for its programs and their static data.
- The remaining portion of the memory stores application programs and their static data and the dynamic data of processes and of OS.
- Memory space is recycled to store applications programs, process data and dynamic kernel programs and data. The subsystem that manages the allocation and de-allocation of memory space is called the *memory manager*.

Memory Manager

- It is an OS component concerned with the system's memory organization scheme and memory management strategies.
- It determines how available memory space is allocated to processes and how to respond to changes in a process's memory usage.
- It also interacts with special purpose memory management hardware (if any is available) to improve performance.
- An ideal memory manager should thus minimize wasted memory and have minimal time complexity and memory access overhead, while providing good protection and flexible sharing.

Allocation of memory can be classified as contiguous allocation and noncontiguous allocation.

- Contiguous allocation scheme allocates a single block of memory for the requesting process
- Noncontiguous scheme allocates chunks or pieces of memory to a process.

Contiguous Memory Management

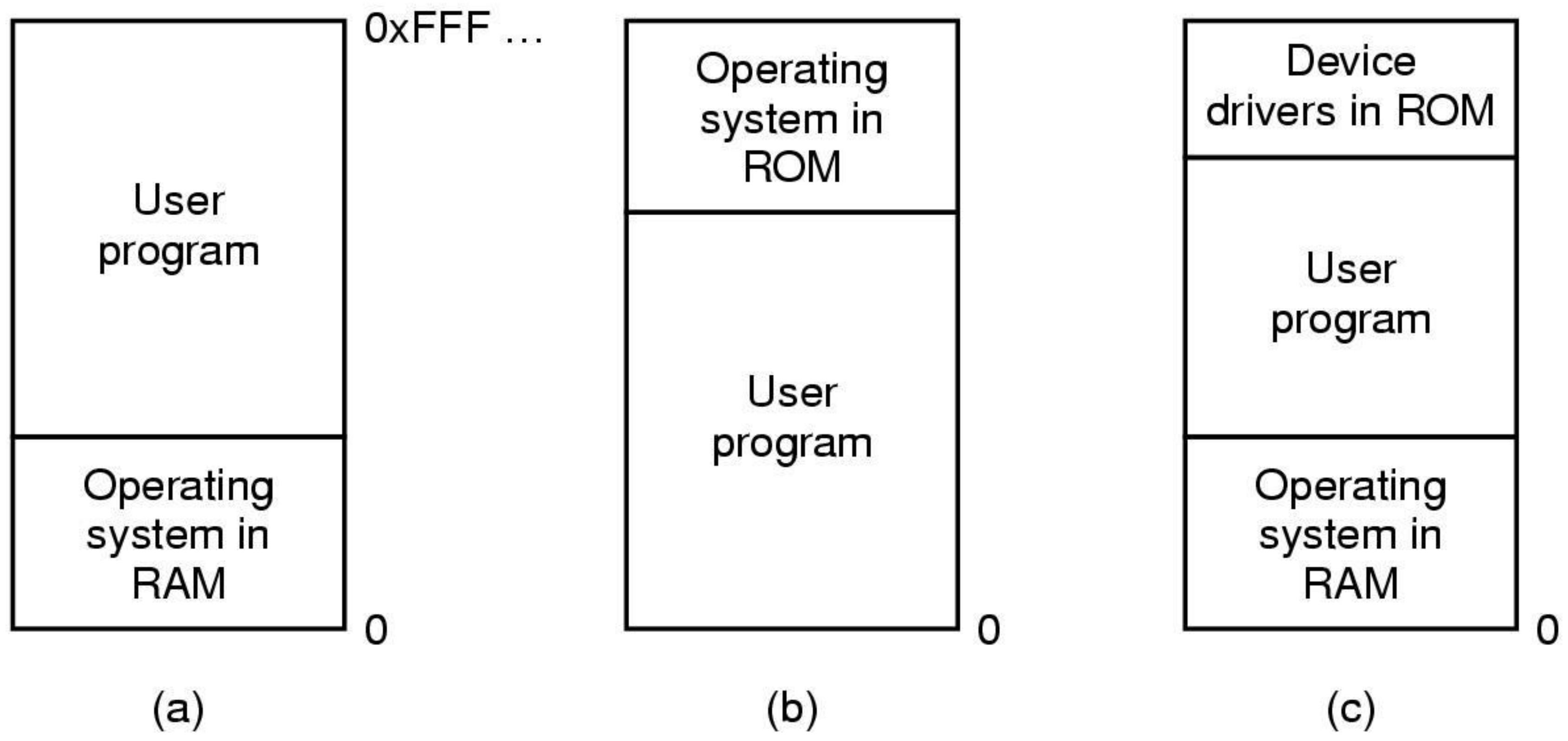
Single Process Monitor/ Single Partition Monoprogramming :

- This approach is one of the simplest ways of managing memory used in single process microcomputer.
- Memory is divided into two contiguous areas and one portion is permanently allocated to the resident portion of OS (monitor) and the remaining portion is allocated to the transient processes which are loaded and executed one at a time in response to user commands.

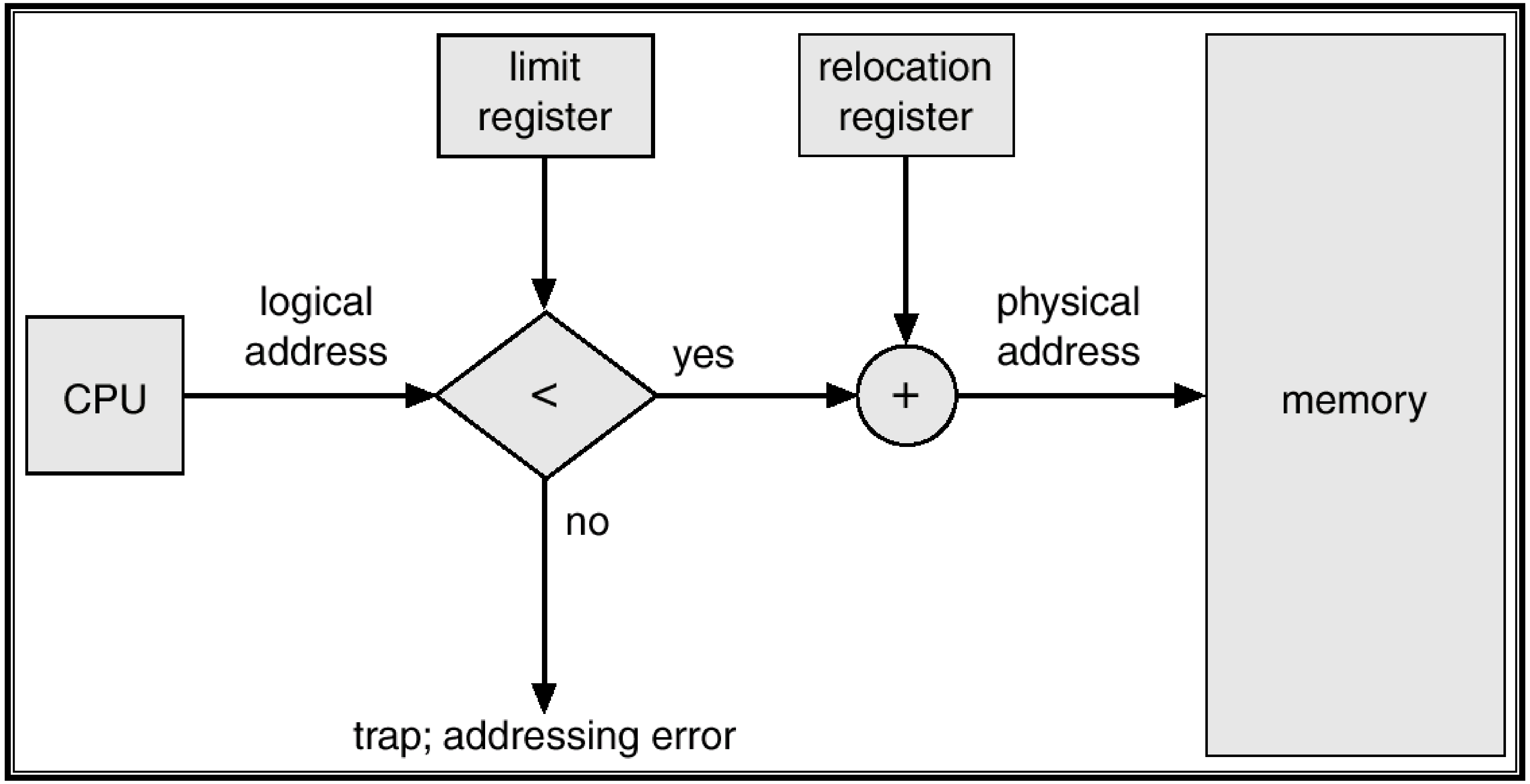
Basic Memory Management
Monoprogramming without Swapping or Paging

Three simple ways of organizing memory

- an operating system with one user process



Hardware Support for Relocation and Limit Registers



Advantages:

- Straightforward memory allocation; Absence of multiprogramming complexities; Relatively simple to design and to comprehend; Used in systems with little hardware support for more advance forms of memory management.

Disadvantages:

- Lack of support for multiprogramming reduces utilization of both processor and memory.

Partitioned memory allocation

- One way to support multiprogramming is to divide the available physical memory into several partitions each of which may be allocated to a different process.
- Depending on when and how partitions are created and modified, memory partitioning may be static or dynamic.

Static Partitioned memory allocation

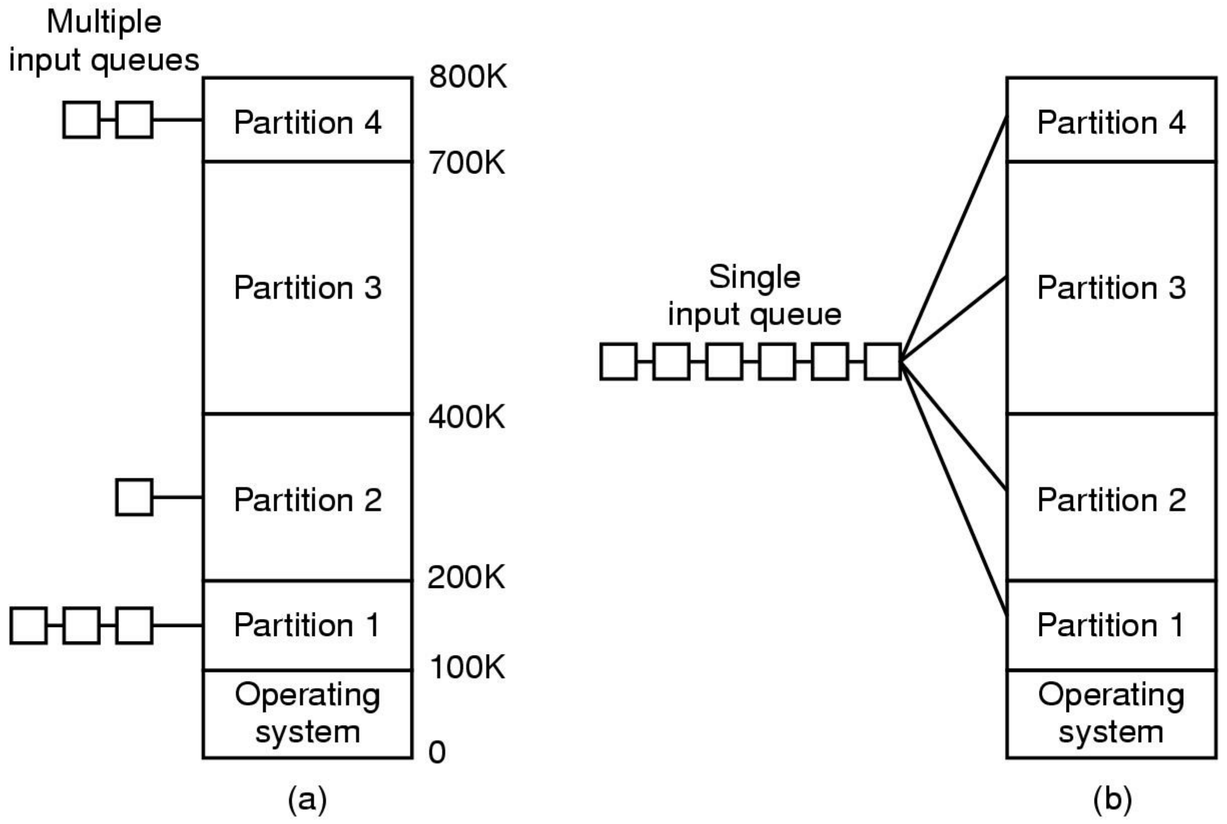
- Static partitioning implies that the division of memory is made at some time prior to the execution of user programs and that partitions remain fixed thereafter.

Allocation of partition:

- The OS must first allocate a memory region large enough to hold the process image which is a file that contains a program in executable form and the related data and may also contain process attributes such as priority and memory requirements.

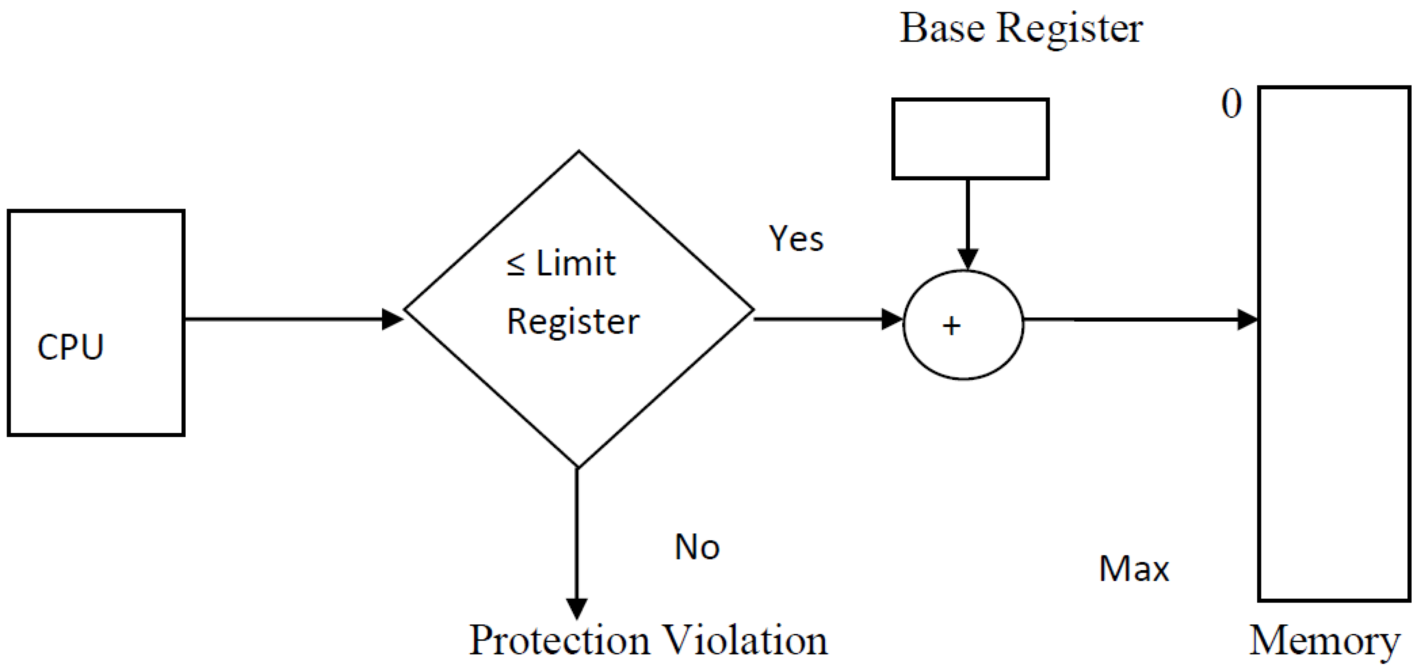
Multiprogramming with Fixed Partitions

- Fixed memory partitions
  - separate input queues for each partition
  - single input queue



- Partition Descriptor Table (PDT) is a data structure used by OS to collect current Partition Status and attributes such as Partition Number, Partition Base and Partition Size.
- In static partitioning, partition number, base and size are fixed, only the partition status varies depending on whether the partition is allocated or not.
- When a non-resident process is to be created or activated, OS attempts to allocate a free memory partition of sufficient size by searching the PDT.

Partition Number	Partition Base	Partition Size	Partition Status
0	0K	100K	Allocated
1	100K	200K	Free
2	300K	100K	Allocated
3	400K	200K	Allocated
4	600K	100K	Free
5	700K	150K	Allocated
6	850K	150K	Free



Base Register
0
CPU
≤ Limit Register
Yes
+
No
Max
Protection Violation
Memory

The partition allocation strategy can be

- First Fit: Allocating the first free partition large enough to accommodate the process being created.
- Best Fit: OS allocates the smallest free partition that meets the requirements of the process under consideration.

Problems associated with allocation

1. No partition is large enough to accommodate the incoming process.

    Solution is to redefine the partitions accordingly or reduce the memory requirements by recoding or by using overlays.

2. All partitions are allocated.

    Solution is by deferring the loading of the incoming process until a suitable partition can be allocated to it or forcing a memory resident process to vacate a sufficiently large partition.

    Additional overhead of selecting a suitable victim and rolling it out to disk will be incurred. This operation is called swapping.

3. Some partitions are free, but none of them is large enough to accommodate the incoming process.

    Solution to this problem is by both deferring and swapping.

Swapping

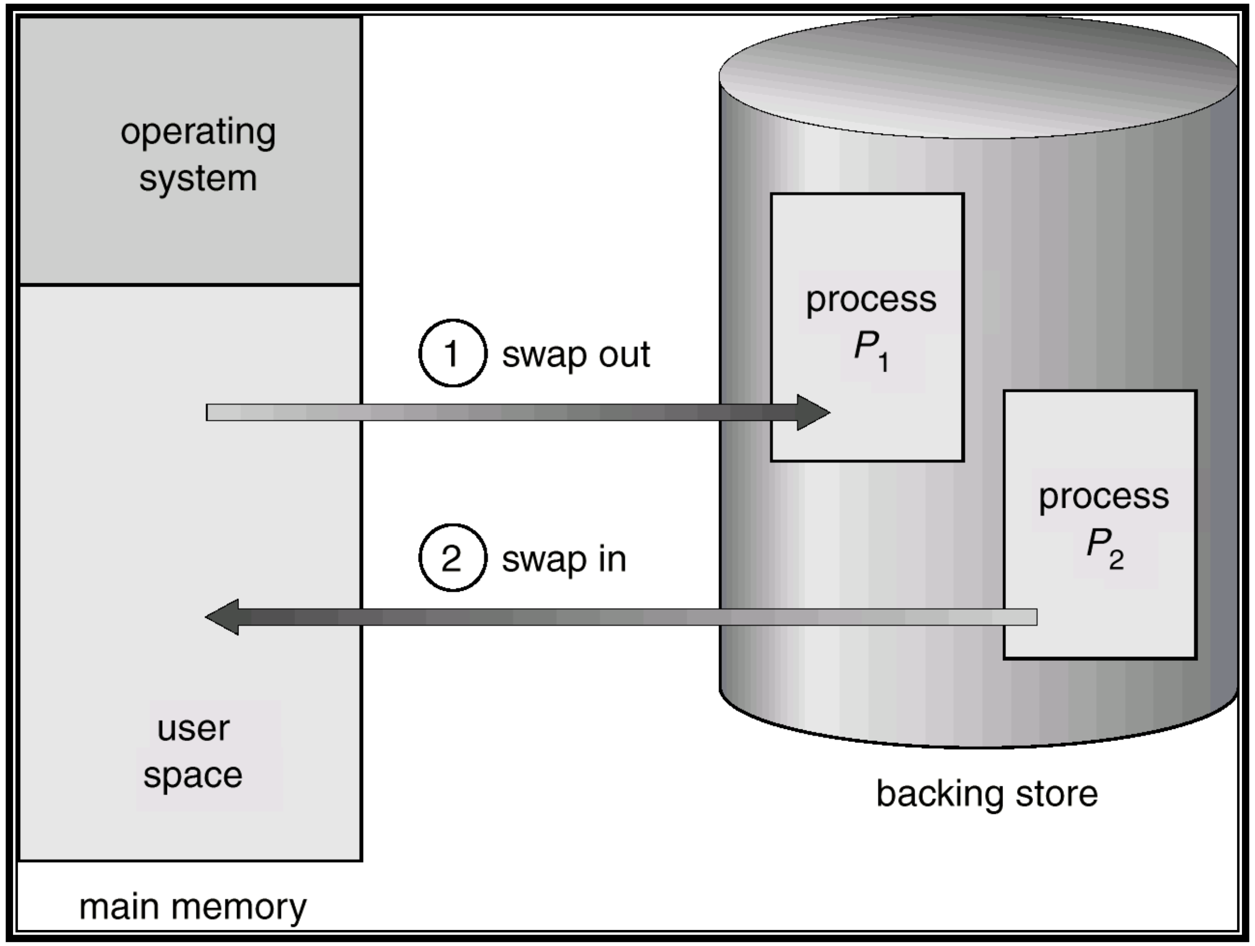
Removing suspended or pre-empted processes from memory and their subsequent bringing back is called swapping.

The major responsibilities of swapper are

- Selection of processes to swap out,
- Selection of processes to swap in and
- Allocation and management of swap space.

- **Backing store** – fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images
- **Roll out, roll in** – swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed
- Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped

Schematic View of Swapping

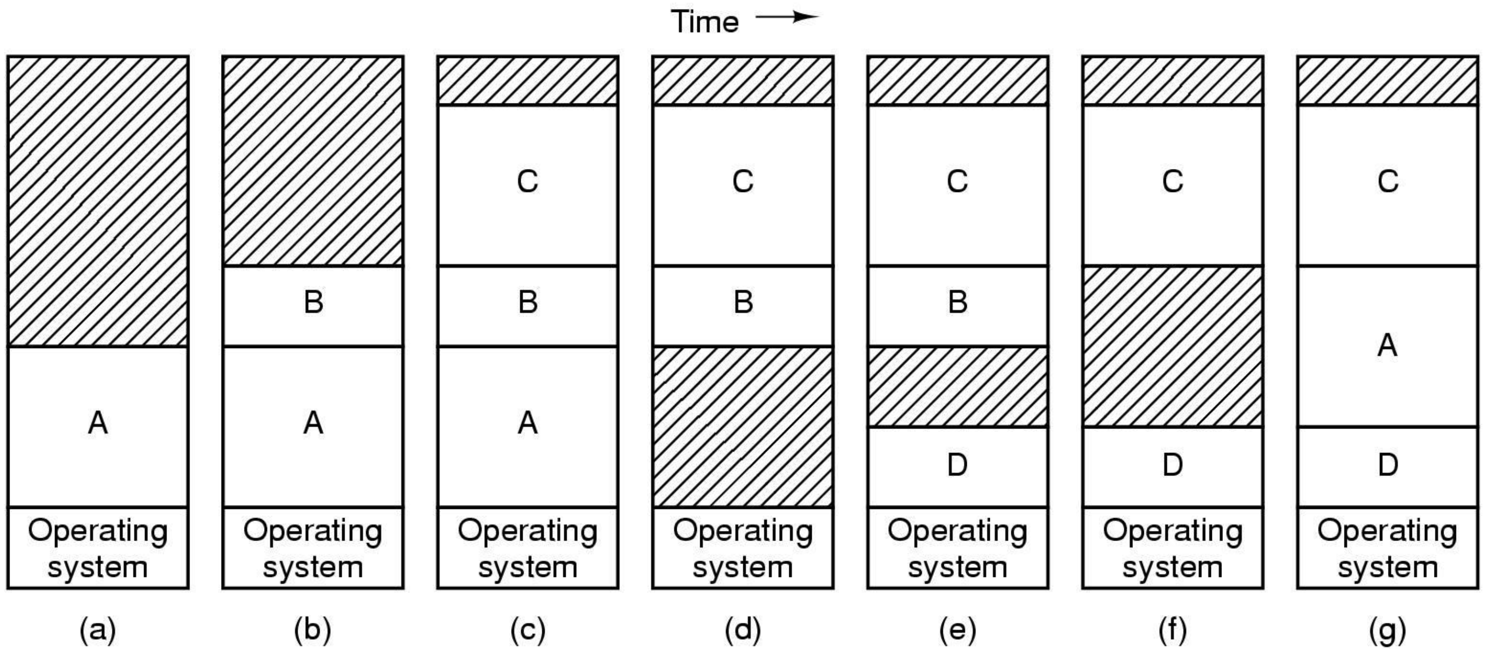


Swapping

Memory allocation changes as

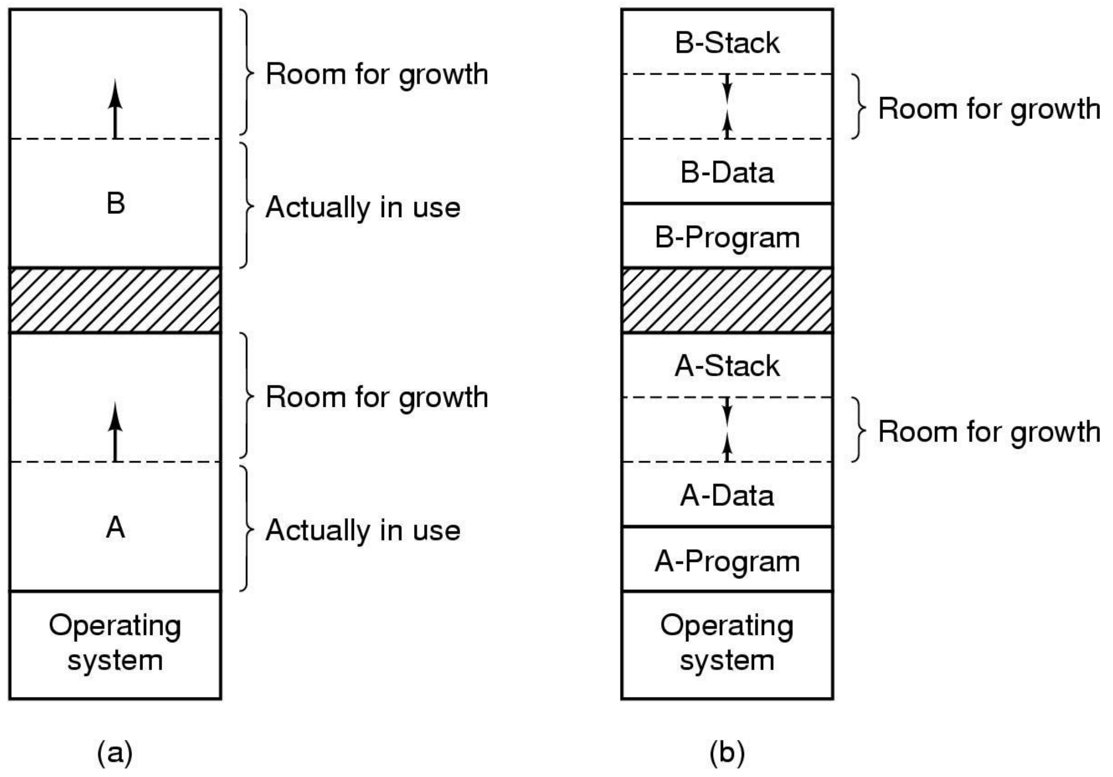
- processes come into memory
- leave memory

Shaded regions are unused memory



Swapping

- Allocating space for growing data segment
- Allocating space for growing stack & data segment



A separate swap file must be available for storing the dynamic image of a rolled out process.

Two options for placing a swap file

i) System wide swap file: Stored in a fast secondary storage device so as to reduce the latency of swapping. The disadvantage is the size of the file. If small it may lead to run time errors and system stoppage due to inability to swap a designated process.

ii) Dedicated per process swap file. File overflow errors are avoided but it consumes more disk space.

Advantage:

- Suitable for static environments where the workload is predictable and its characteristics are known.

Disadvantages:

- Inflexible and inability to adapt to changing system needs.
- Internal fragmentation of memory resulting from the difference between the size of a partition and the actual requirements of the process that resides in it.
- Fixed size of the partitions does not support dynamically growing data structures such as heaps or stacks.

Dynamic/ Variable Partitioned memory allocation

- Partition size may vary dynamically to overcome the problem of determining the number and sizes of partitions and to minimize internal fragmentation.

Allocation of Partition:

- In this method, in addition to PDT, a free list of partitions is maintained. Initially, the whole memory is free and it is considered as one large block.
- When a new process arrives, the OS searches for a block of free memory large enough for that process. The rest of the available free space are kept for the future usage.

- If a block/partition becomes free, then the OS tries to merge it with its neighbours if they are also free otherwise append it to the free list. This appending may lead to a free list of holes of very small size.
- External fragmentation is possible i.e. a suitable single partition meeting the process requirement may not be available, but if the sizes of the holes are added, it may be greater than or equal to the requesting process's size.

The procedure for creating a partition of size P for a requesting process T is as follows:

1. Search for a partition of size F ≥ P. If no match is found then error.

2. Calculate D= F- P. If D ≤ c where c is a small constant value, allocate the entire free area by setting P = F and base address of P as F's base address and adjust the links in free list.

If D > c, then allocate space by setting base address of P as F's base address. Modify F's base address as P's base address + P’s size and F's size as F's original size - P's size.

3. Find an unused entry in the PDT and record the base address, size of the partition and set the status as allocated.

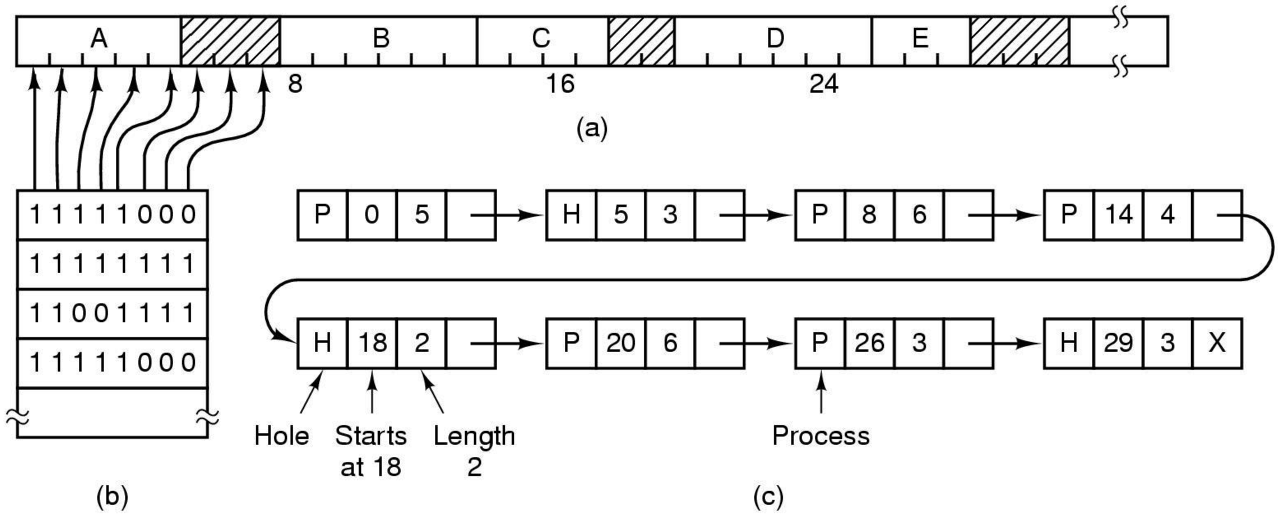
4. Record the PDT's entry number in the process control block of the process T.

Memory Allocation – Mechanism

- MM system maintains data about free and allocated memory alternatives
  - Bit maps – 1 bit per “allocation unit”
  - Linked Lists – free list updated and coalesced when not allocated to a process
- At swap-in or process create
  - Find free memory that is large enough to hold the process
  - Allocate part (or all) of memory to process and mark remainder as free
- Compaction
  - Moving things around so that holes can be consolidated
  - Expensive in OS time

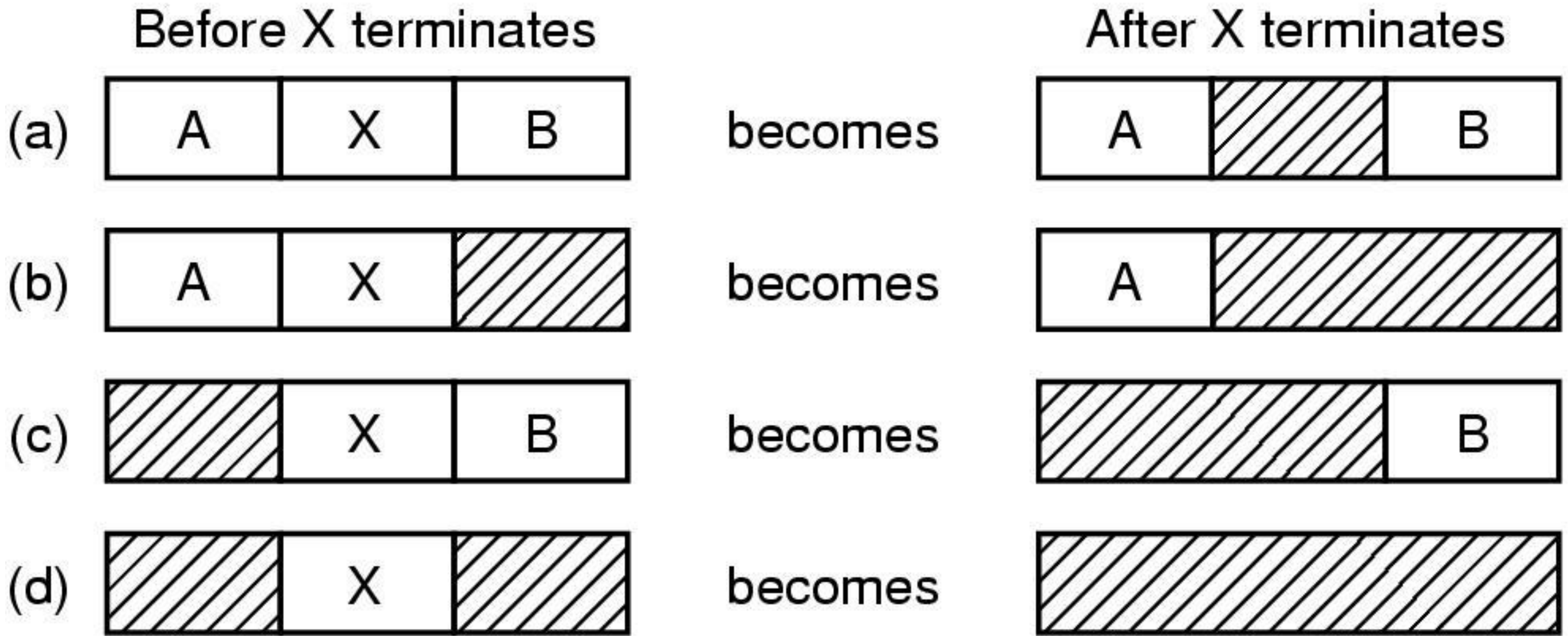
Memory Management with Bit Maps

- Part of memory with 5 processes, 3 holes
  - tick marks show allocation units
  - shaded regions are free
- Corresponding bit map
- Same information as a list



Memory Management with Linked Lists

Four neighbor combinations for the terminating process X



Selection of partitions

- First Fit : Allocate the first free block that is large enough for the new process. This is a fast algorithm.
- Next Fit : A variant of first fit in which the pointer to the free list is saved following an allocation and used to begin the search for the subsequent allocation.
- Best Fit : Allocate the smallest block among those that are large enough for the new process.
- Worst Fit : Allocate the largest block among those that are large enough for the new process. Again a search of the entire list or sorting is needed.

Buddy System

- An allocation-deallocation strategy, called Buddy system, facilitates merging of free space by allocating free areas with an affinity to recombine.
- The size of the partitions are powers of base 2 and the requests for free areas are rounded to the next power of base 2.
- If there is no partition of size of requesting process then a partition of next larger size is split into two buddies to satisfy the request.
- When the block is freed then the block may be recombined with its buddy if it is free.

Compaction

- It is a method to overcome the external fragmentation problem.
- All small free blocks are brought together as one large block of free space. Compaction requires dynamic relocation.
- Selection of an optimal compaction strategy is difficult.
- One method for compaction is swapping out those processes that are to be moved within the memory, and swapping them into different memory locations.

Protection and Sharing

- The mechanisms used are same as that of static partitioning except that dynamic partitioning allows adjacent partitions in physical memory to overlap. Sharing of code must be reentrant or executed in a mutually exclusive fashion with no preemptions.

Advantages:

- All available memory except for the resident portion of the OS may be allocated to a single program.
- It can accommodate processes whose memory requirements are increasing during execution.

Disadvantages:

- It needs complex bookkeeping and memory management algorithms.
- External fragmentation may lead to time penalty for compaction.