MEMORY MANAGEMENT

Chapter 4.1

MEMORY MANAGEMENT

- *The entire program and data of a process must be in main memory for the process to execute.*
- How to keep the track of processes currently being executed?
- Which processes to load when memory space is available?
- How to load the processes that are larger than main memory?
- How do processes share the main memory?

- OS component that is responsible for handling these issues is a ***memory manager***.

MEMORY MANAGER

- The part of the operating system that manages the memory hierarchy is called the **memory manager**.
- Its job is to
  - keep track of which parts of memory are in use
  - allocate memory to processes
  - deallocate memory when processes are done,
  - manage swapping between main memory and disk

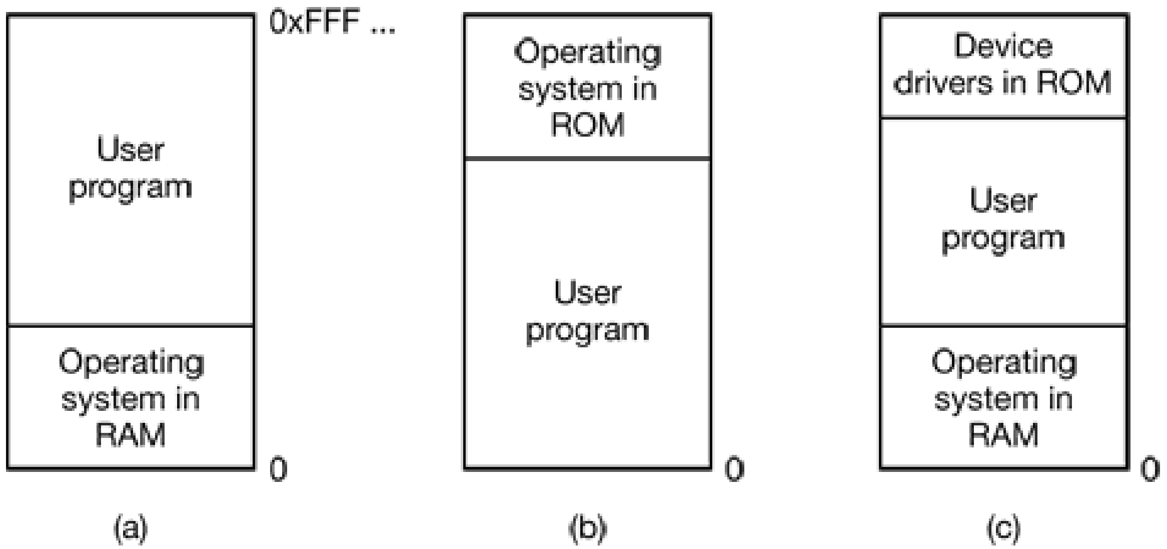
MEMORY MANAGEMENT SCHEMES

- Two classes:

- Type 1. Those that move processes back and forth between main memory and disk during execution (swapping and paging)
- Type 2. and those that do not.

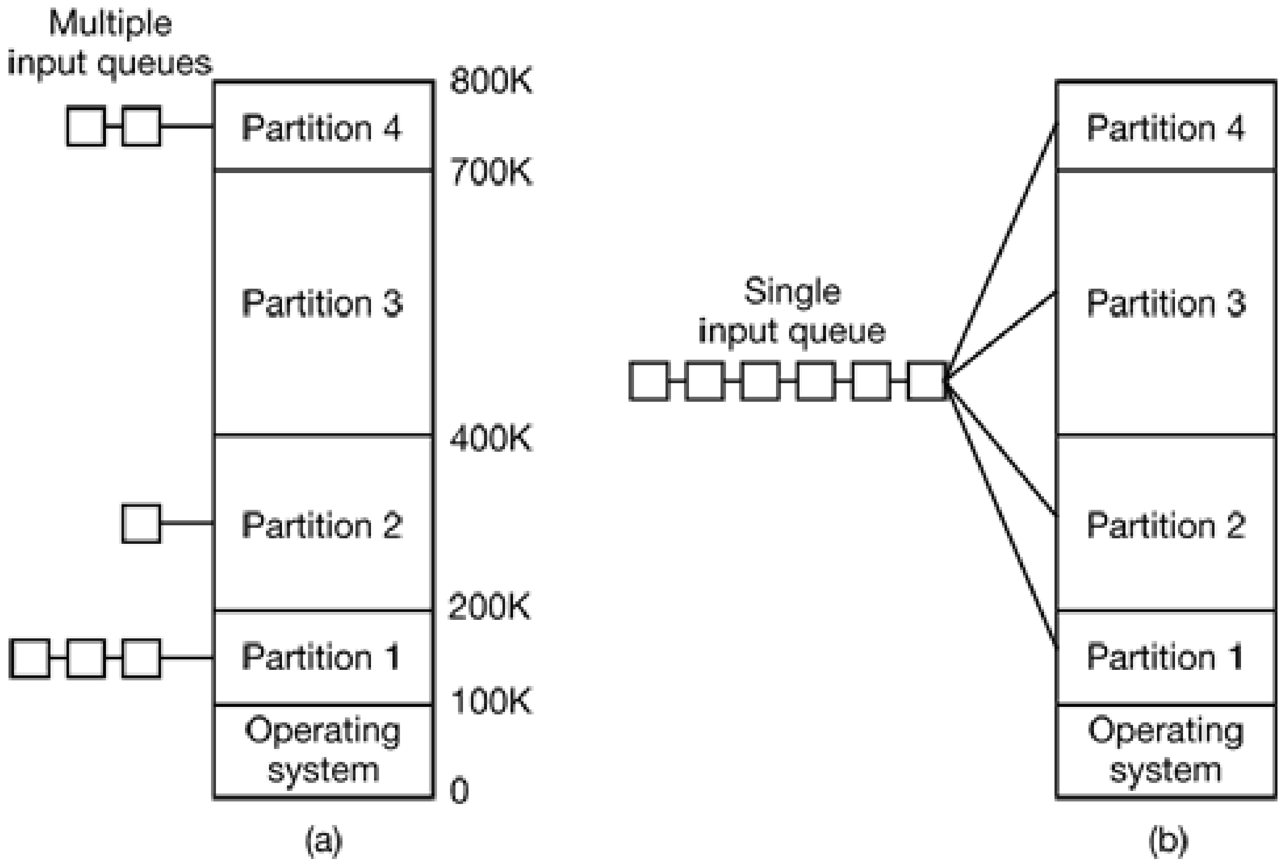
TYPE 2. MONOPROGRAMMING WITHOUT SWAPPING OR PAGING

- run just one program at a time
- share the memory between that program and the operating system.
- Three variations on this scheme are shown in Fig.
  - (a)on mainframes and minicomputer
  - (b)on some palmtop computers and embedded systems
  - (c) early personal computers (e.g., running MS-DOS),
    - where the portion of the system in the ROM is called the **BIOS** (Basic Input Output System).



TYPE 2. MULTIPROGRAMMING WITH FIXED PARTITIONS

- multiple processes run at the same time
- Memory partitioned into fixed no. of partitions



TYPE 2. MULTIPROGRAMMING WITH FIXED PARTITIONS

- With multiple input queues
  - large partition could be empty while the queue for a small partition is full.
- With Separate input queues
  - Since it is undesirable to waste a large partition on a small job,
  - **pick the largest job that fits**
    - **discrimination against small jobs**
    - usually it is desirable to give the smallest jobs (often interactive jobs) the best service, not the worst.
- One way is to have at least one small partition around
  - allow small jobs to run without having to allocate a large partition for them.
- Another approach is to have a rule stating that a job that is eligible to run may not be skipped over more than *k* times. Each time it is skipped over, it gets one point. When it has acquired *k* points, it may not be skipped again.