## Memory

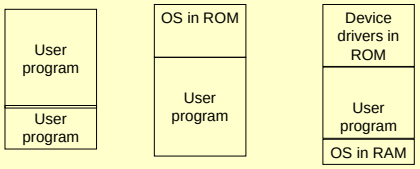
- Machines today have 10,000 times more memory that best machines of 1960s
- Cost of memory has dropped dramatically (and continues to drop – unlike s/w)
- Bill Gates: *"640K should be enough"*
- Programmes expand to fill available memory
- Each programme cannot have infinitely large and fast memory
- OS must manage memory

## Memory Management

- Memory manager:
  - The part of the OS that manages memory
- Terms:
  - Allocate: allocate a virtual memory block to use
  - Deallocate: free a virtual memory block allocated before
  - Reallocate: grow or shrink virtual memory allocated before
- Role:
  - Keep track of what parts of memory are in use or free
  - Allocate memory to processes (and deallocate when finished)
  - Manage swapping between main memory and disk memory
- Note: programme must be brought into memory and placed within a process so that it can be run

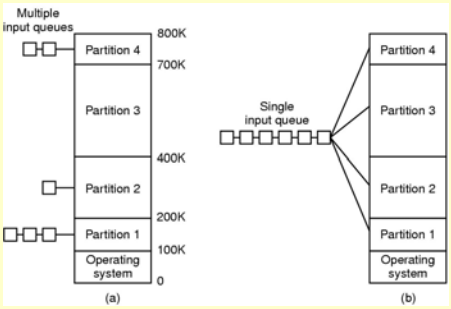
## Simple Memory Management

- Simplest possible memory management scheme:
  - 1 process in memory at one time
  - That process can use all of memory
  - (not realistic)
  - Basic Input Output System (BIOS)
  - ROM: Read Only Memory
  - RAM: Random Access Memory




## Multiprogramming and Memory Usage

- Monoprogramming rarely used
- Many processes "running" at the same time
- Not all processes can be in memory at the same time
- Process need to be swapped into (and out of) main memory
- Solutions:
  - Fixed memory partitions of (different sizes) with separate input queues
  - Fixed memory partitions (of different sizes ) with a single input queue




(a) Separate input queues for each partition
(b) Single input queue

## Single Queue vs Multiple Queues

- Single Queue:
  - Job closest to front that fits partition is loaded
  - What is the problem with taking the first job that fits into the partition?
  - What is the problem with taking the largest job that fits into the partition?
  - What do you think **internal fragmentation** is?
- Multiple Queues:
  - What is the disadvantage?

## Multiprogramming and Memory Usage

- One queue:
  - What is the problem with taking the first job that fits into the partition?
    - Mightn't make good use of partitions
  - What is the problem with taking the largest job that fits into the partition?
    - Discriminates against small jobs (usually I/O ones)
    - Solution:
      - Have at least 1 small partition
      - A job can only be skipped *k* times before being loaded
- Multiple/separate queues:
  - One queue can be full, other empty

## Memory Management: Issues

- How to allocate/manage memory?
- What if memory does not suffice? (Swapping)
- Transparency
  - No process needs to be aware of *how* memory management works
  - Process should not care about physical memory allocation
- Safety
  - Process should not be able to write to each other's space
- Efficiency
  - Space usage
  - Time

## Two Problems

- With multiprogramming, processes will be run at different addresses
- Two problems: Relocation and protection
- Relocation:
  - User writes code
  - Code is compiled
  - Code is linked with library procedures and combined into a single address space
  - Linker must know at what address the programme will begin in memory

## Relocation

- How to know where to start (or jump to) a section of code, *regardless* of where it is loaded into memory
- Static relocation:
  - Fix up address when programme is loaded
  - But … once a process is assigned a placed in memory, the OS cannot move it (not good)
- Dynamic relocation:
  - Use Dynamic Address Translation
  - Translate *programme* address into *main memory* address during execution

## Protection

- Processes should not be able to reference the memory of another process without permission
- A process needs to be prevented from accessing memory beyond that allocated to it
- Need to prevent malicious or malfunctioning code in one programme from interfering with the operation of other running programmes
- Early solution:
  - Divide memory into blocks of 2K bytes and assign a protection code to each block
  - OS kept track of protection codes
- Alternate solution for relocation and protection:
  - Use base and limit registers
  - Base register: where the code starts
  - Limit register: length of partition

## Swapping

- Swapping
  - Moving processes from disk to main memory and back
  - Could be based on fixed partitions
    - Wasted memory by small programmes
  - Variable partitions
    - Improves memory utilisation
    - Complicates allocating/de-allocating memory
    - Memory compaction: moving all processes downward (not usually done)

## Algorithms for Memory Management

- Best Fit
  - Return the smallest hole (free space) which is larger than the requested size
    - Expensive search
    - External fragmentation: generates small and unusable blocks
- First Fit
  - Return the first hole which is larger than the requested size
    - Small holes tend to accumulate near the beginning of the free list
    - Solution –next fit: start search where the last one left off

## Algorithms for Memory Management

- Buddy system
  - All blocks and holes have sizes which are powers of two
  - Every block or hole starts at an address that is an exact multiple of its size
  - Then every block has its buddy with the same size. Always coalesce a block with its buddy, generating a free block twice as large as the free block

## Algorithms for Memory Management

- Worst fit
  - Take the largest available hole, so that the hole broken off will be big enough to be useful
  - Not a good idea
- Quick fit
  - Maintains separate lists for some of the more common sizes requested
  - Finding a hole is fast
  - Finding neighbours for merge is expensive

## Memory Allocation

- Fragmentation
  - External: Where there is enough memory is free to satisfy a request, but it is split into two or more chunks, none of which is big enough to satisfy the request
  - Internal: memory is wasted within a segment
- Running out of memory
  - Return error
  - Compaction
    - move all the allocated blocks to one end of memory, thus combining all the holes
  - Garbage collection
    - also known as *automatic memory management*, is the automatic recycling of dynamically allocated memory

## Virtual Memory

- The combined size of the programme, data on the stack may exceed the amount of physical memory available for it
- OS keeps those parts of the programme currently in use in main memory (the rest on disk)
- Most virtual memory systems use **paging**
- **Virtual addresses** go to a **memory management unit** (MMU) that maps them onto the physical memory addresses
- Virtual address space is divided up into **pages**
- The corresponding units in the physical memory are called **page frames**

## Virtual Memory

- **Page tables** are used to map between the two addresses
  - 1. page table can be very large
  - 2. mapping must be fast
  - complex
- **Page fault**
  - Page not in memory
    - Interrupt OS
    - Swap out (inactive) page
    - Swap in required page

## Page Replacement Algorithms

- Not-Recently-Used
  - Keeps track of recently used/modified pages
  - Removes a page at random from the lowest numbered nonempty class
  - Easy to understand, efficient to implement, good performance
- First-In-First-Out (FIFO)
  - Easy to understand, not used
- Second Chance
  - Checks for oldest, non-referenced page
- Least Recently Used
  - Throw out a page that has been unused for the longest time
  - Costly

## Thrashing

- If a process does not have “enough” pages, the page-fault rage is very high
- **Thrashing** is when a process is busy swapping pages in and out
- Fixes
  - Working Set Model
    - Pages loaded on demand
    - Locality of reference
    - prepaging
  - Page Fault Frequency
    - Global (vs local) allocation policies
    - Try to keep page faults within a given range

## Other Issues

- Page size
  - Small
    - Only required programme in memory
    - Large page table, transfer delay
  - Large
    - Opposite of above
- Segments
  - Data, stack, code

## Summary

- Simplest system
  - No swapping
- Swapping
    - System handles more processes than it has room for in memory
- Virtual memory
  - Process’ address space divided into blocks called pages
  - Pages placed in page frames in memory
- Many page replacement algorithms
- Segmentation is an alternative to pure paging