Memory

Memory
Learning Objectives

- Overlays & Fixed Partitions
  - Internal Fragmentation
  - Why separate queues are inefficient
- Virtual Addresses
  - Relocation using Base Register
- Dynamic contiguous allocation
  - Bitmaps versus linked lists
  - Allocation Schemes (Best,First, Worst, Next)

Memory Management

- The ideal world has memory that is
  - Very large
  - Very fast
  - Very cheap
  - Non-volatile (doesn't go away when power is turned off)
- The real world has memory that is
  - Very large
  - Very fast
  - Affordable!
  - ⇒ Pick any two…
- Memory management goal
  - Make the real world look as much like the ideal world as possible

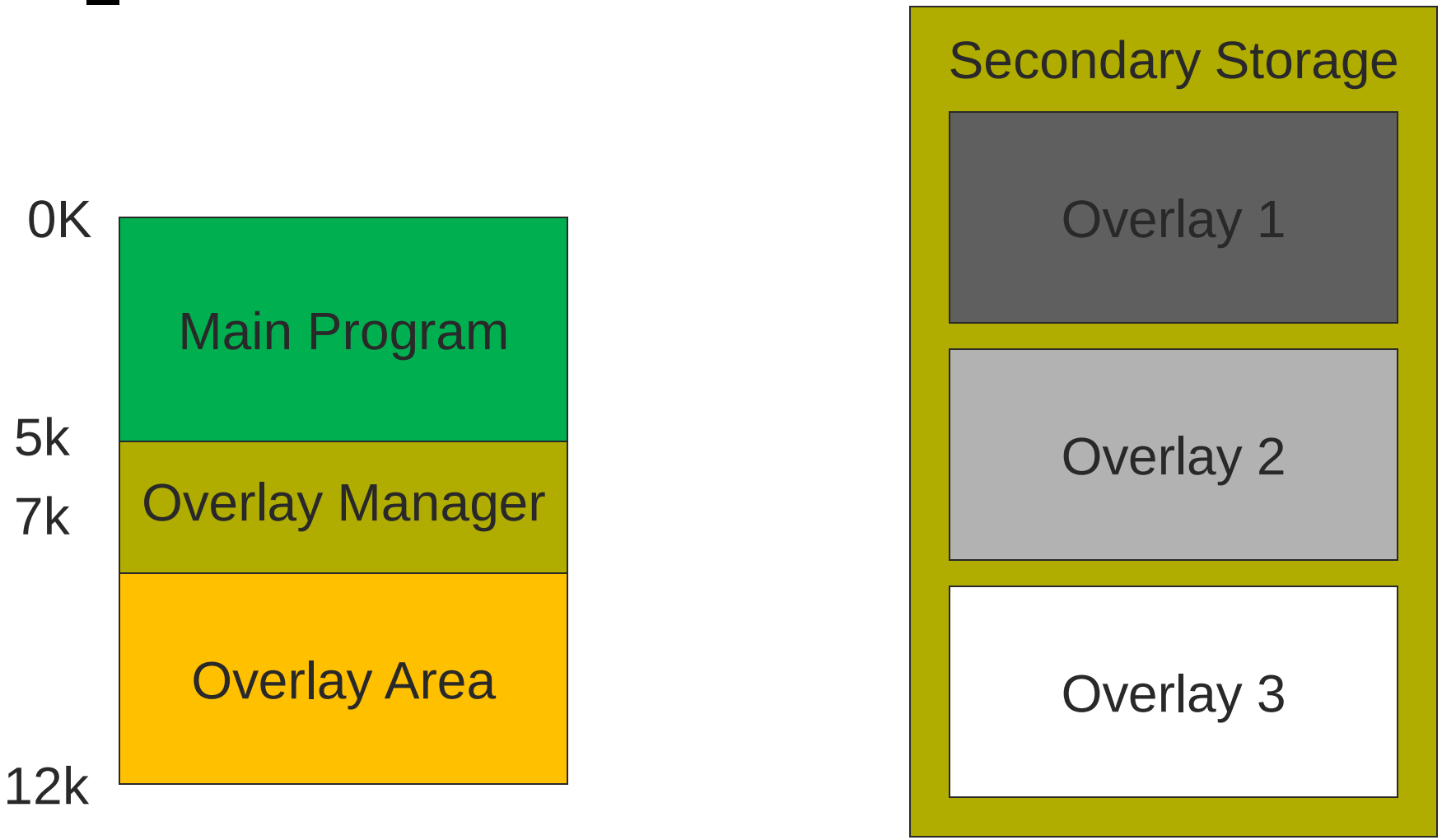
Memory Management

- Goal
  - Layout the programs in memory as needed
- Potential issues
  - Utilization
  - Protection
  - Flexibility

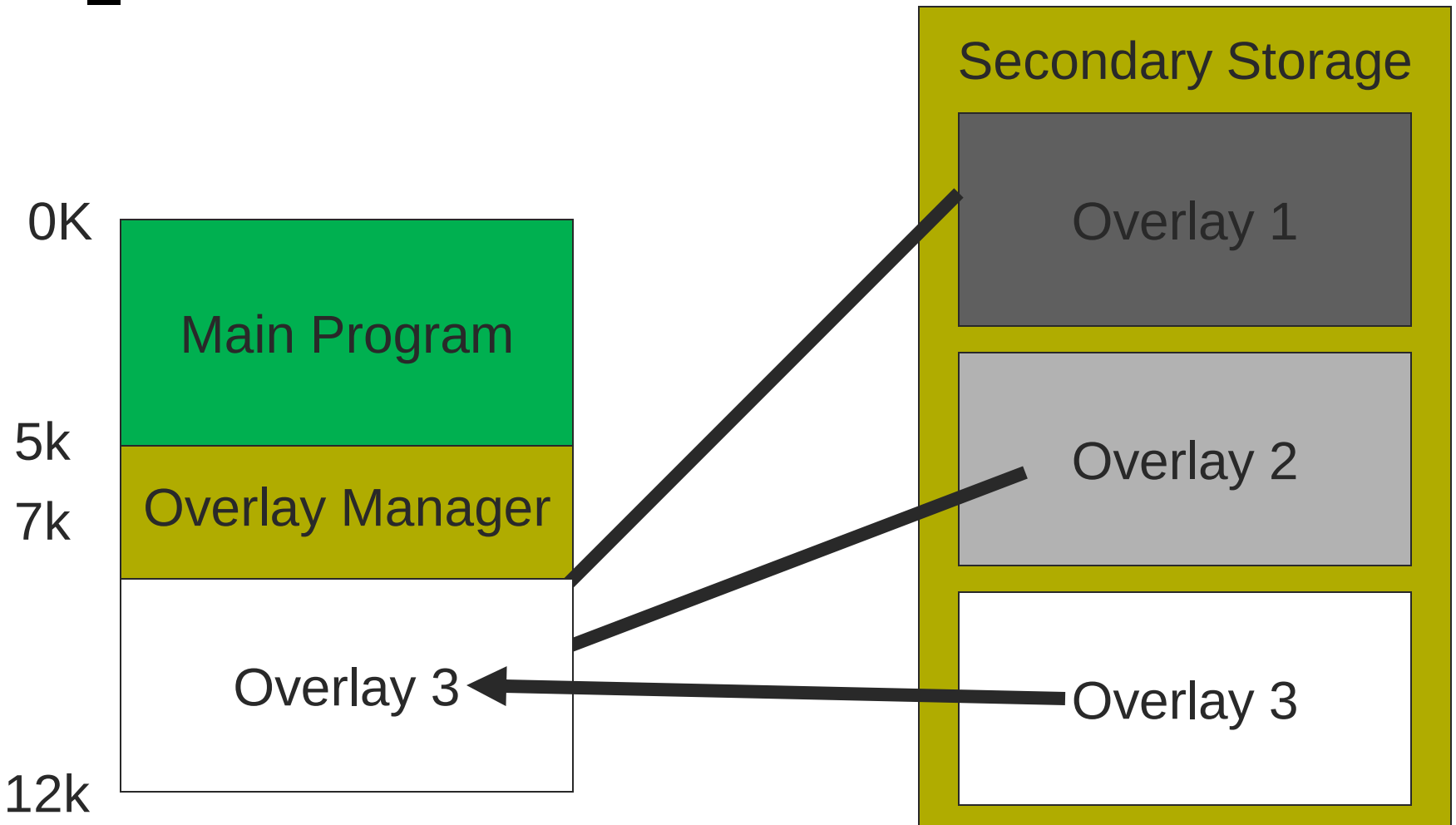
Space Constraints

- Problem
  - Not enough memory to keep all application data
- One solution
  - Overlays: keep only important instructions and data in memory

Overlays

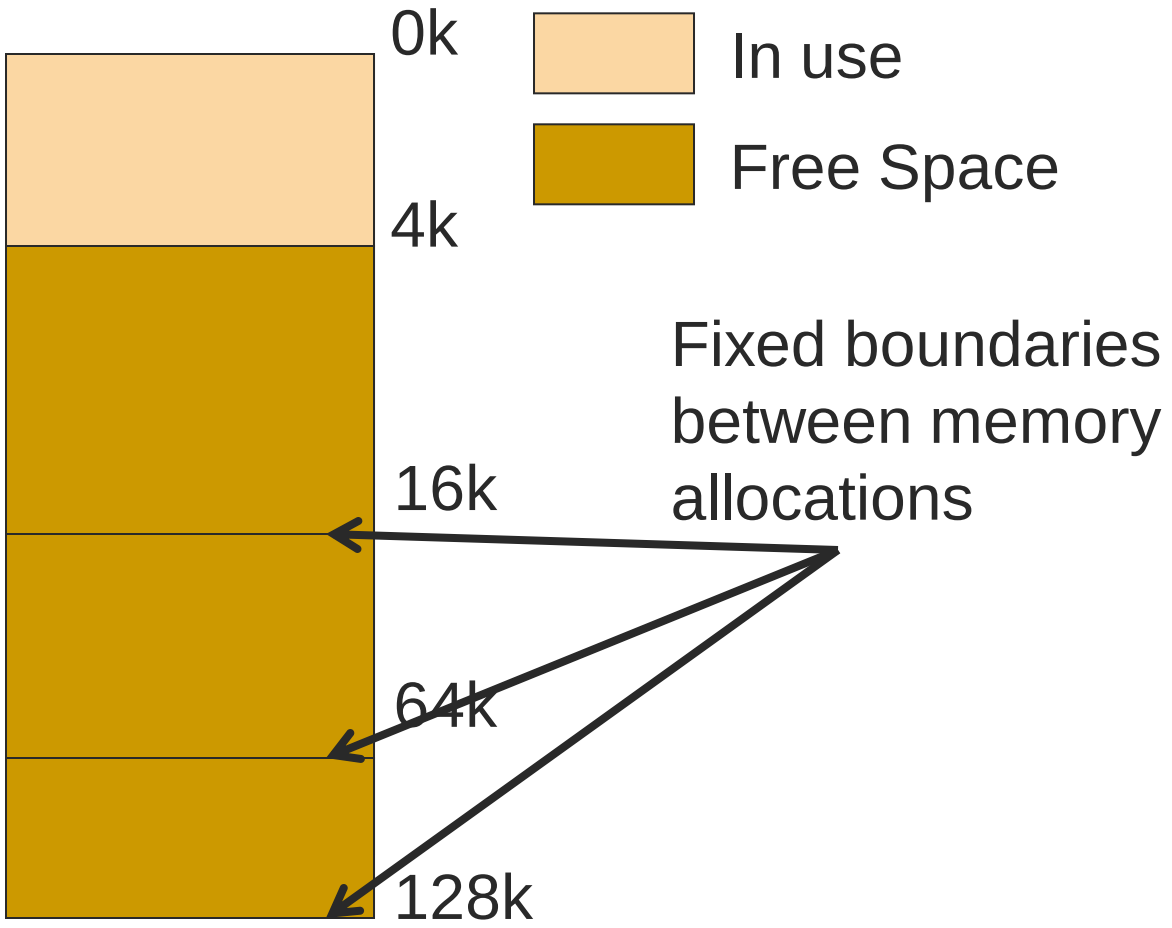


Overlays

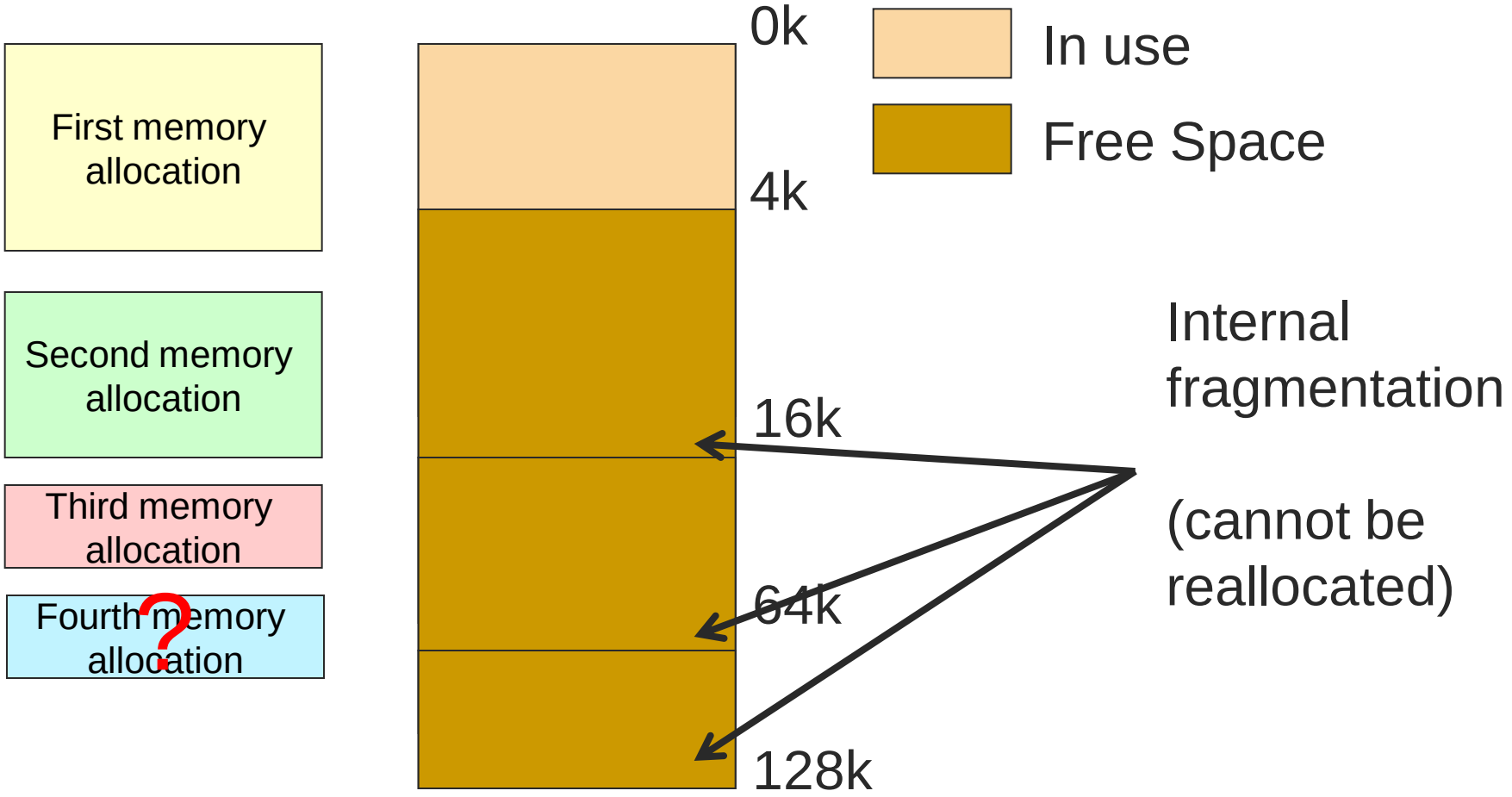


Mulitple Fixed Partitions

Divide memory into *n* (possible unequal) partitions.

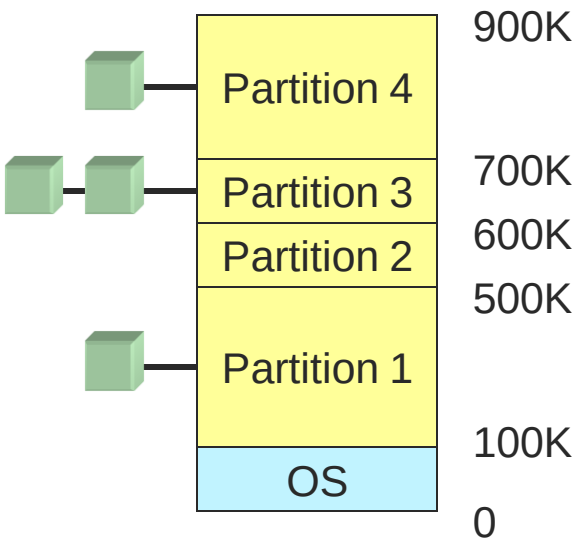


Multiple Fixed Partitions



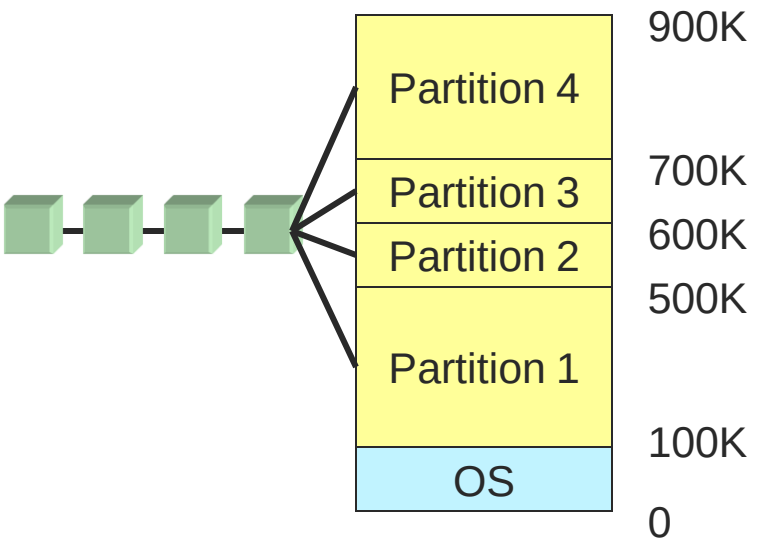
Fixed Partition Implementation

- Separate input queue for each partition
  - Put incoming jobs into separate partition queues
    - Requires sorting the incoming jobs and putting them into separate queues
  - Inefficient utilization of memory
    - Small jobs still wait



Fixed Partition Implementation

- Solution: One single input queue for all partitions.
  - Allocate a partition where the job fits and use…
    - Best Fit
    - Worst Fit
    - First Fit



Virtual addresses

- "Any programming problem can be solved by adding a level of indirection.“
- Logical address
  - Address generated by the CPU
  - Virtual address
- Physical address
  - Address seen by the memory unit

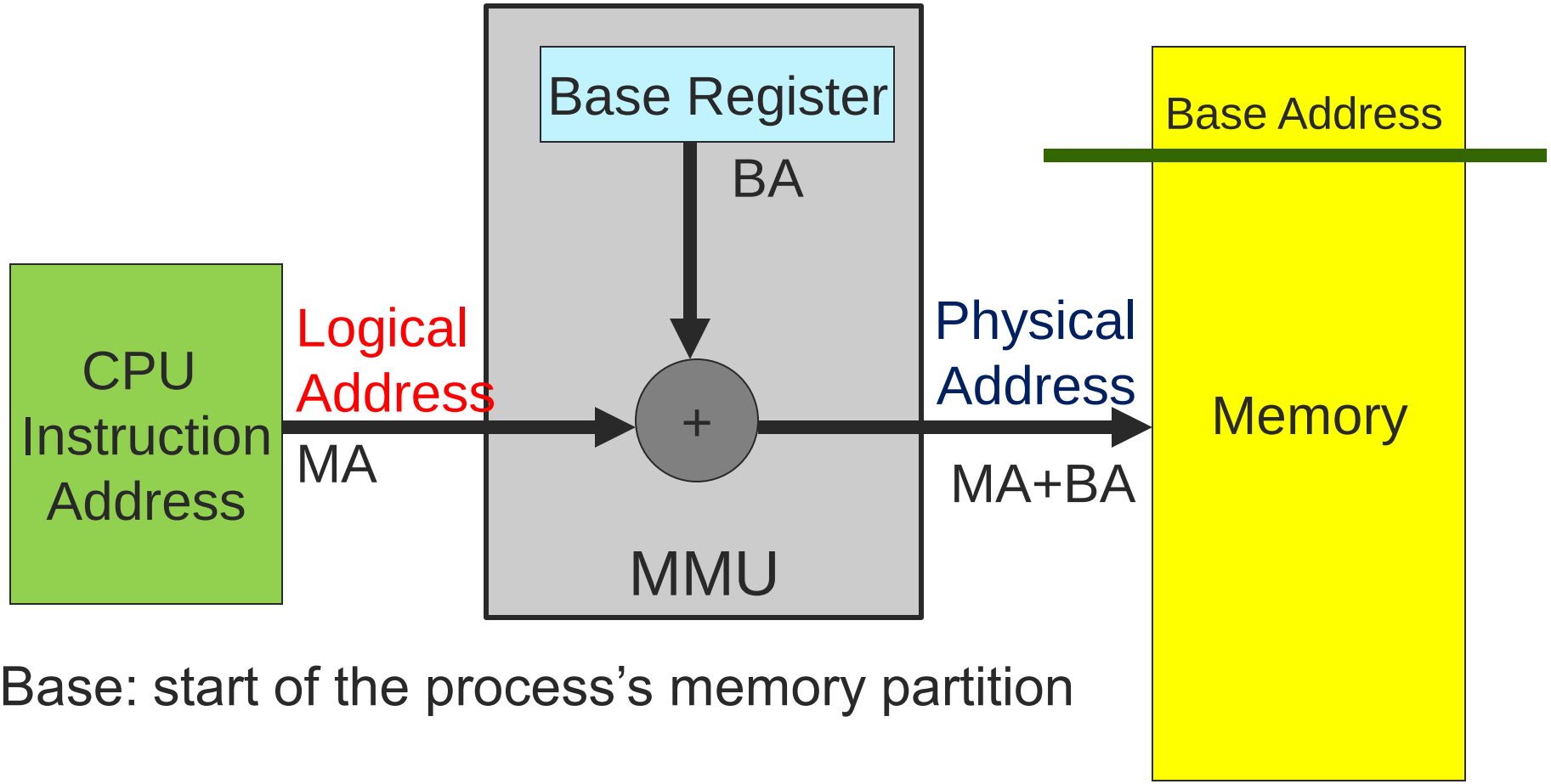
Virtual addresses

- Different jobs will run at different addresses
  - When a program is linked, the linker must know at what address the program will begin in memory
  - Program never sees physical address
  - Correct starting address when a program starts in memory

Base Register

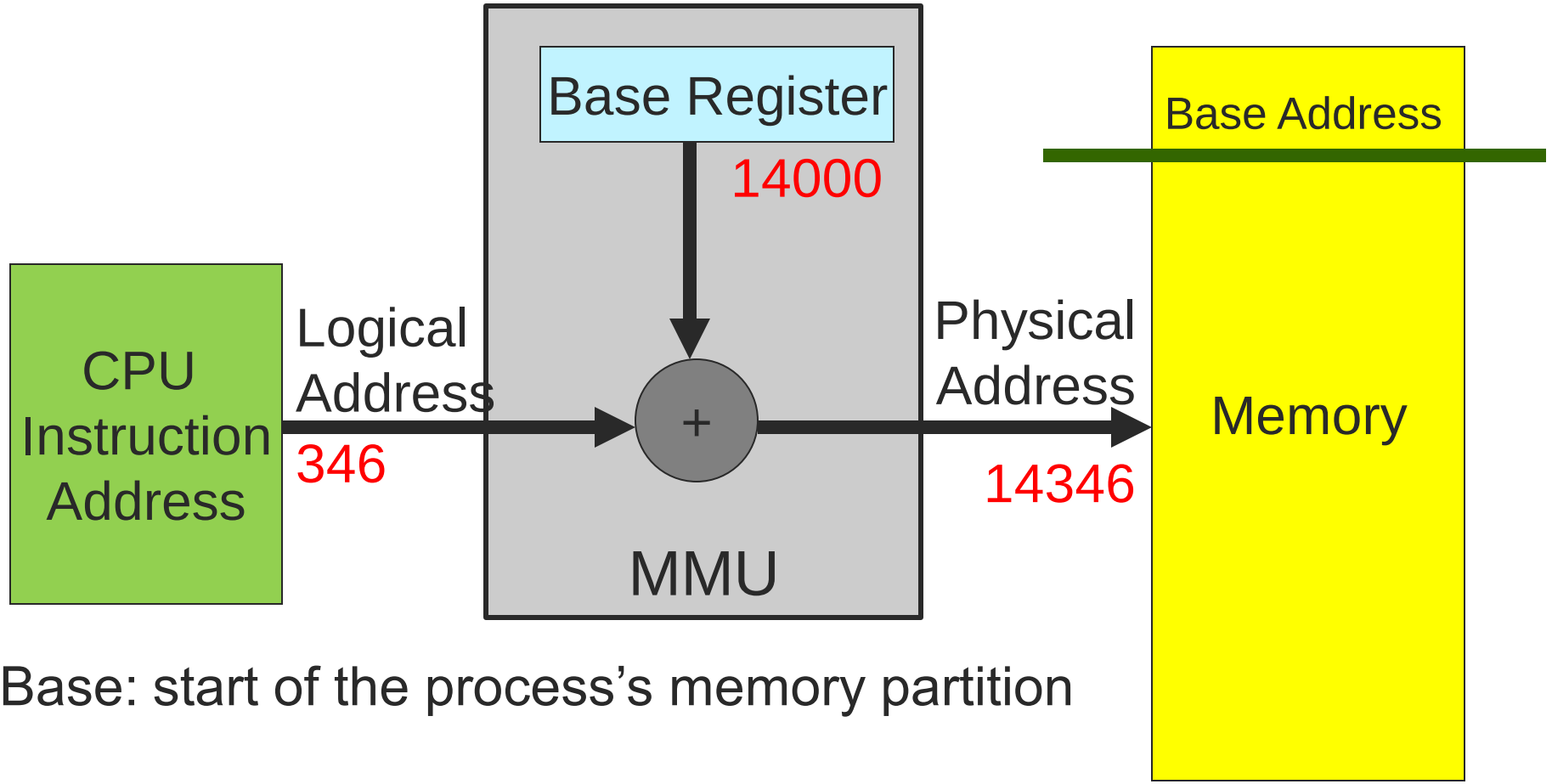
- Logical or "Virtual" addresses
  - Logical address space
  - Range: 0 to max
- Physical addresses
  - Physical address space
  - Range: R+0 to R+max for base value R
- How
  - Memory-management unit (MMU)
    - Map virtual to physical addresses
  - Relocation register
    - Mapping requires hardware (MMU) with the base register

Relocation Register



Base: start of the process's memory partition

Relocation Register

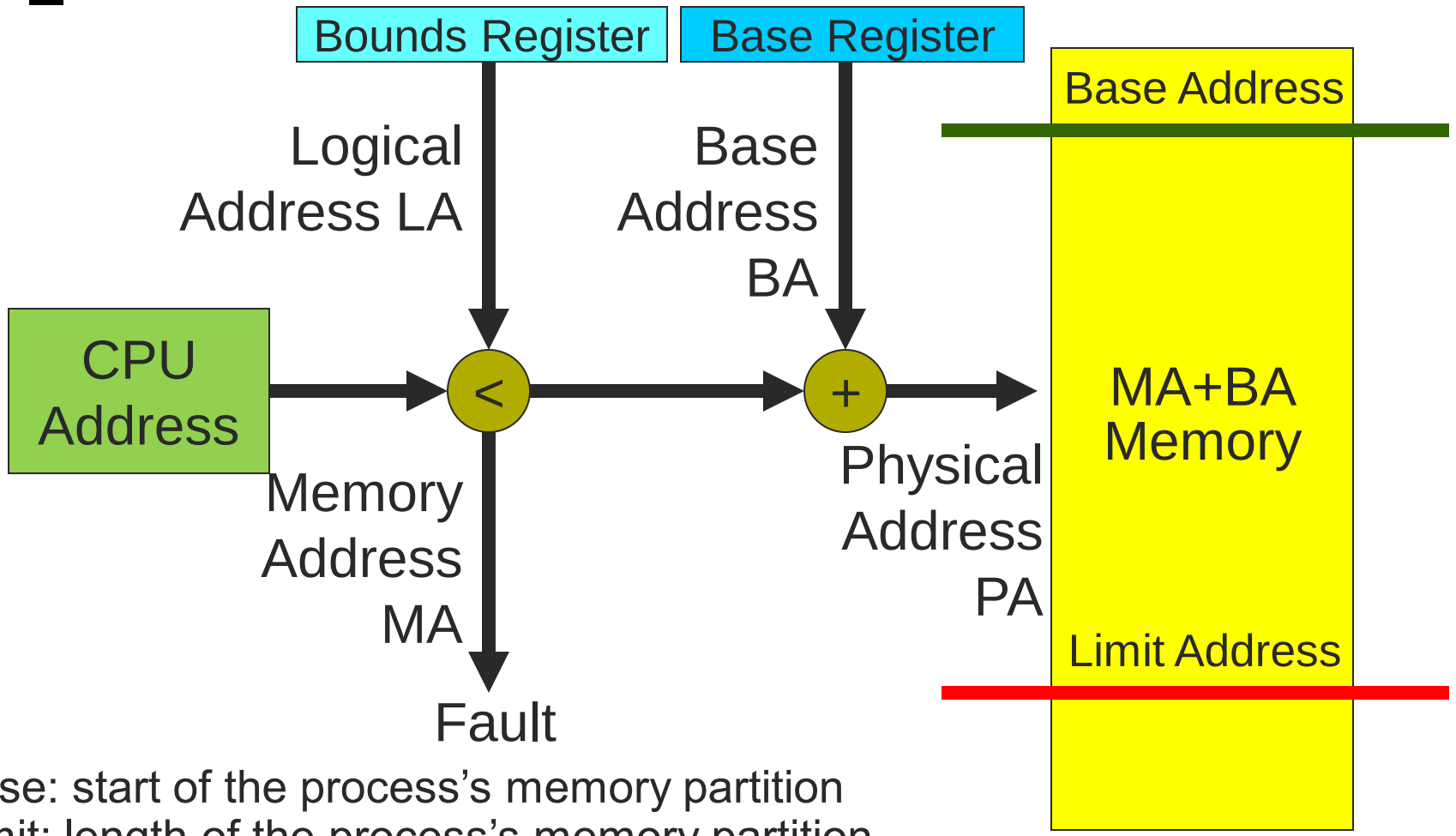


Base: start of the process's memory partition

Protection

- Problem
  - How to prevent a malicious process from writing or jumping into other user's or OS partitions
- Solution
  - Base bounds registers

Base Bounds Registers



Base: start of the process's memory partition
Limit: length of the process's memory partition

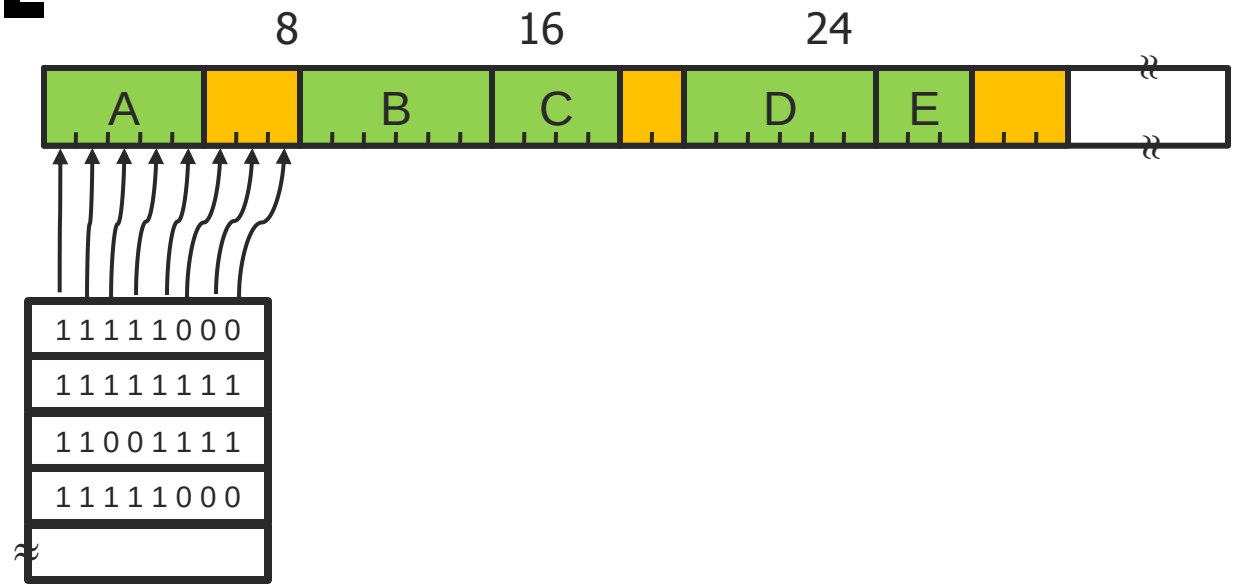
Memory Management

- Goal
  - Keep track of free / allocated memory regions
- Mechanisms
  - Bitmaps
    - One bit in map corresponds to a fixed-size region of memory
  - Linked lists
    - Each entry in the list corresponds to a contiguous region of memory

Bit Maps and Linked Lists

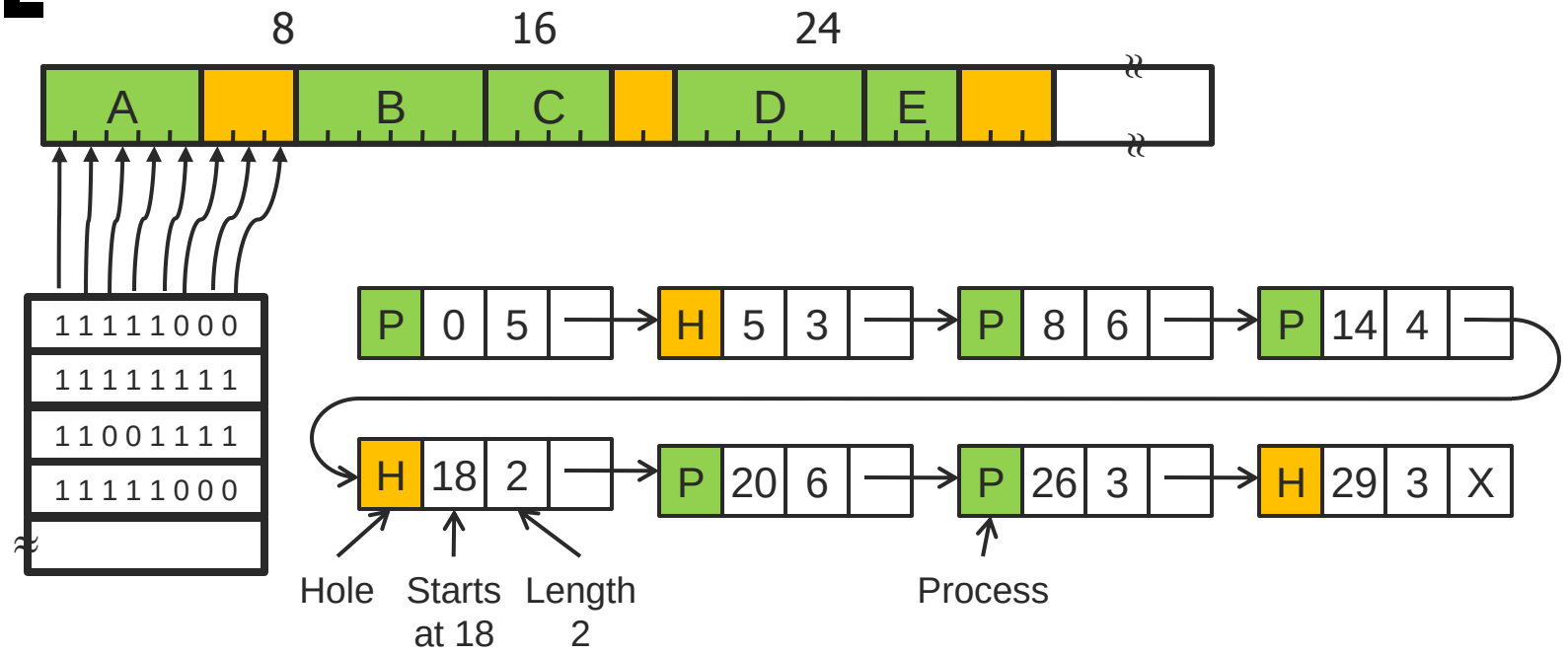
- Part of memory with 5 processes, 3 holes
  - Tick marks show allocation units
  - Green regions are free

Bit Maps and Linked Lists



- Part of memory with 5 processes, 3 holes
  - Tick marks show allocation units
  - Green regions are free

Bit Maps and Linked Lists



- Part of memory with 5 processes, 3 holes
  - Tick marks show allocation units
  - Green regions are free

Partition Allocation schemes

- Bitmap vs. link list
  - Which one occupies more space?
    - Depends on the individual memory allocation scenario
    - In most cases, bitmap usually occupies more space
  - Which one is faster to reclaim freed space?
    - On average, bitmap is faster because it just needs to set the corresponding bits
  - Which one is faster to find a free hole?
    - On average, a link list is faster because we can link all free holes together

Storage Placement Strategies

- First fit
  - Use the first available hole whose size is sufficient to meet the need
  - Rationale?
- Best fit
  - Use the hole whose size is equal to the need, or if none is equal, the hole that is larger but closest in size
  - Rationale?
- Worst fit
  - Use the largest available hole
  - Rationale?

Example

- Consider a swapping system in which memory consists of the following hole sizes in memory order:
  - 10K, 4K, 20K, 18K, 7K, 9K, 12K, and 15K.
  - Which hole is taken for successive segment requests of:
    - 12K
    - 10K
    - 9K

Example

- Consider a swapping system in which memory consists of the following hole sizes in memory order:
  - 10K, 4K, 20K, 18K, 7K, 9K, 12K, and 15K.
  - Which hole is taken for successive segment requests of:
    - 12K
    - 10K
    - 9K

First fit: 20K, 10K, 18K.	Best fit: 12K, 10K, 9K.	Worst fit: 20K, 18K, and 15K.

Storage Placement Strategies

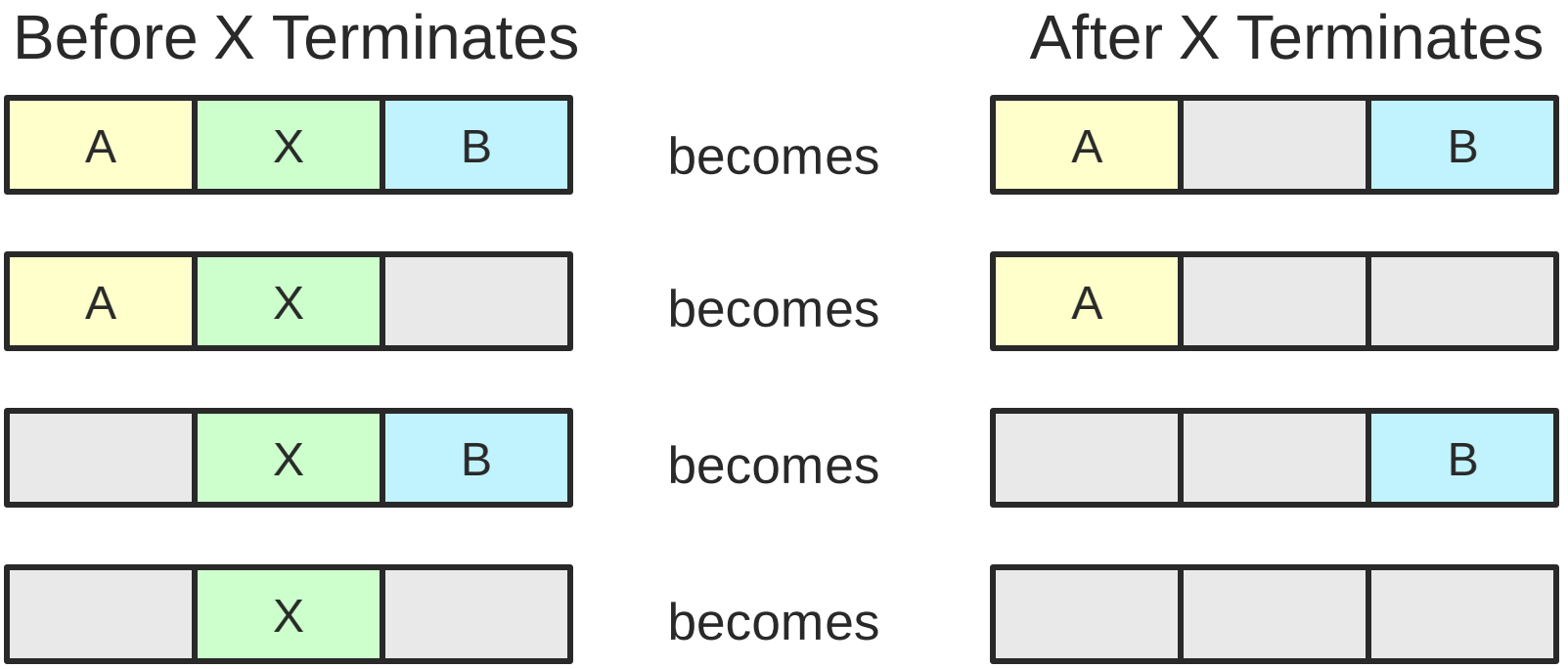
- Best fit
  - Produces the smallest leftover hole
  - Creates small holes that cannot be used
- Worst Fit
  - Produces the largest leftover hole
  - Difficult to run large programs
- First Fit
  - Creates average size holes

- First-fit and best-fit better than worst-fit in terms of speed and storage utilization

Fragmentation

- External Fragmentation
  - Memory space exists to satisfy a request, but it is not contiguous
- Internal Fragmentation
  - Allocated memory may be slightly larger than requested memory
  - The size difference is memory internal to a partition, but not being used

Memory Management: Process Termination



- Four neighbor combinations for the termination of process X

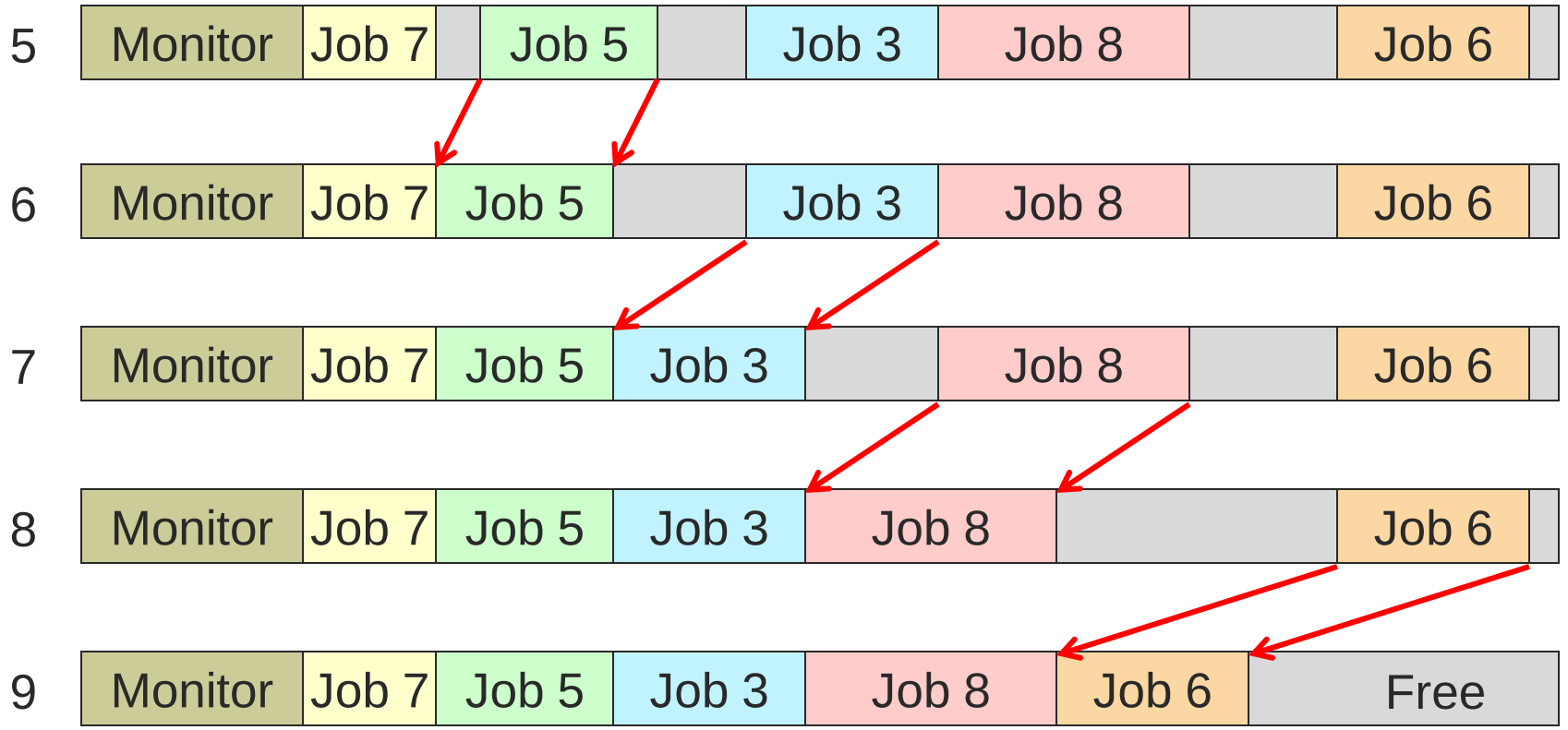
How Bad Is Fragmentation?

- Statistical arguments - Random sizes
- First-fit
  - Given N allocated blocks
  - 0.5∗N blocks will be lost because of fragmentation
- Known as 50% RULE

Compaction

- Reduce external fragmentation by compaction
  - Shuffle memory contents to place all free memory together in one large block
  - Compaction is possible only if relocation is dynamic, and is done at execution time

Solve Fragmentation w. Compaction



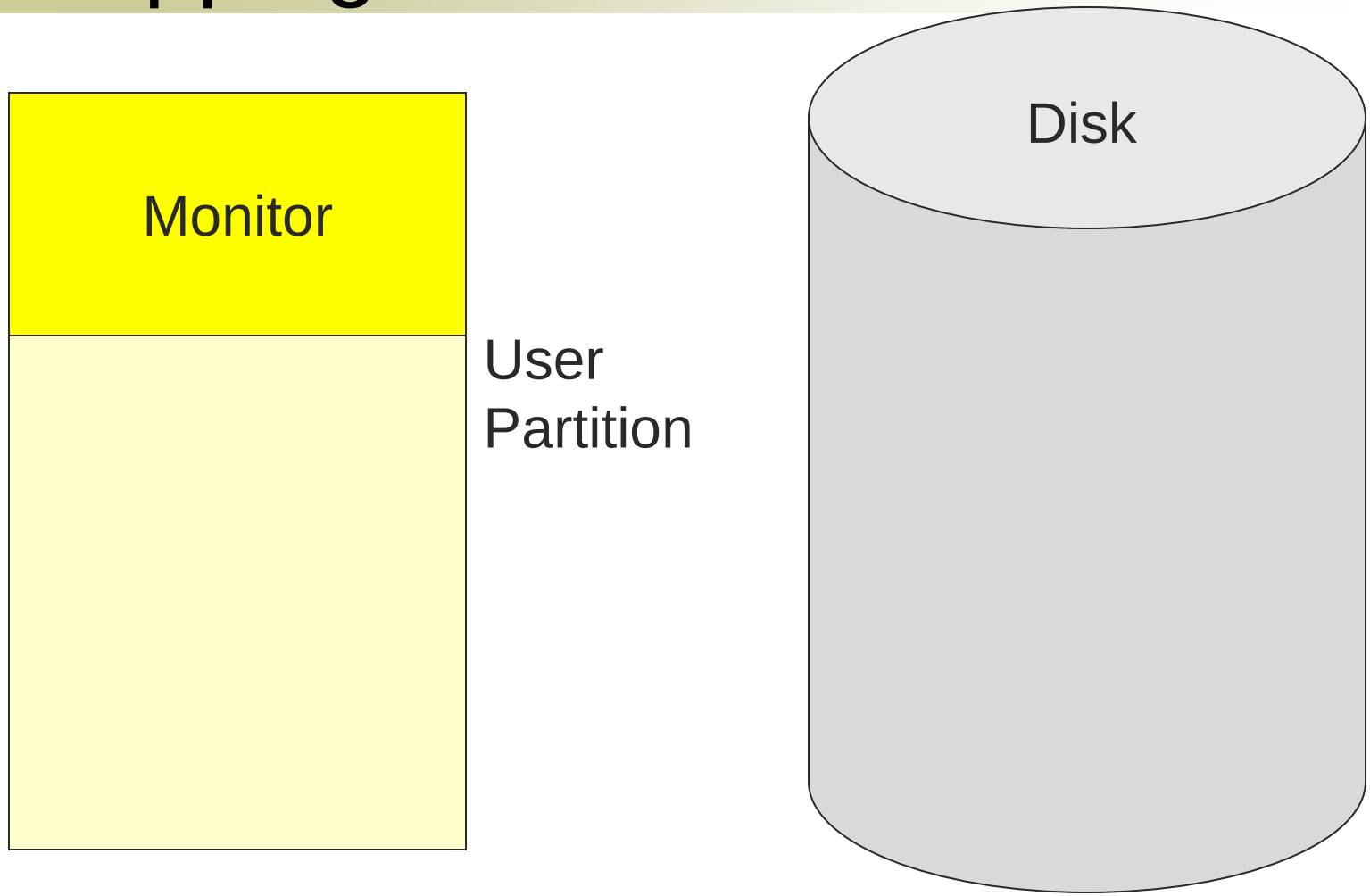
Storage Management Problems

- Fixed partitions suffer from
  - Internal fragmentation
- Variable partitions suffer from
  - External fragmentation
- Compaction suffers from
  - Overhead

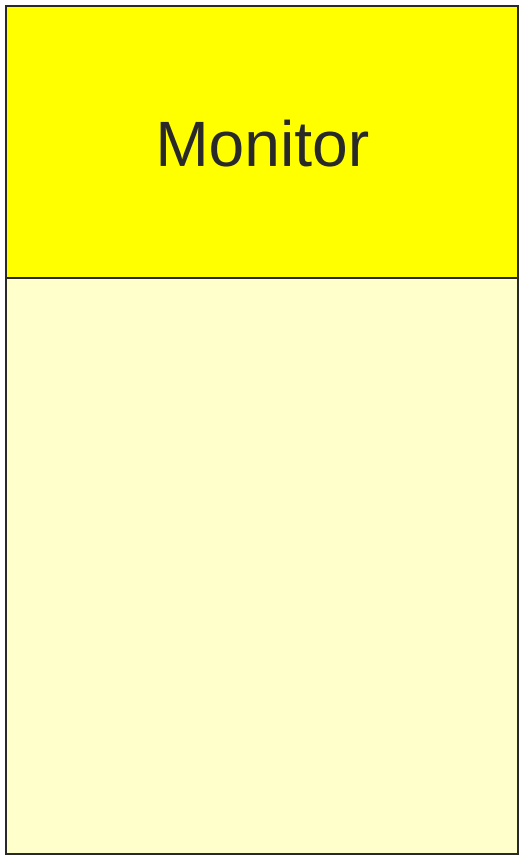
Question

- What if there are more processes than what could fit into the memory?
- Swapping
- Memory allocation changes as
  - Processes come into memory
  - Processes leave memory
    - Swapped to disk
    - Complete execution

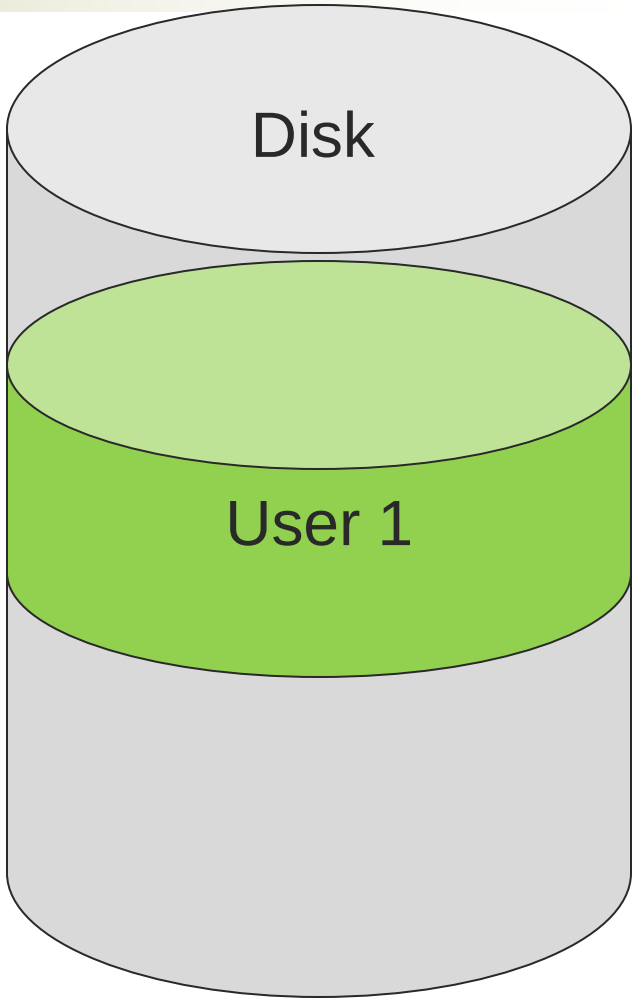
Swapping



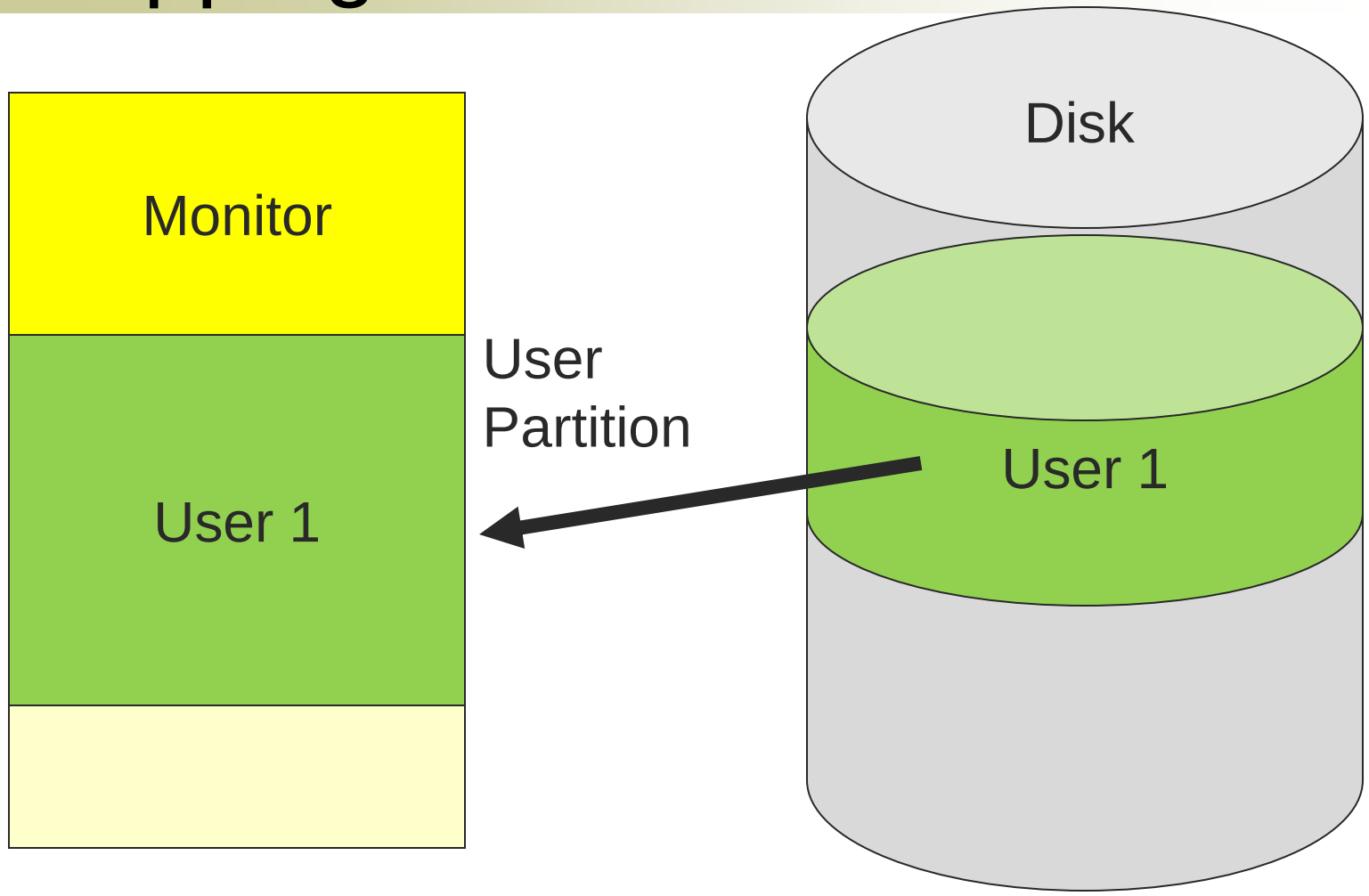
Swapping



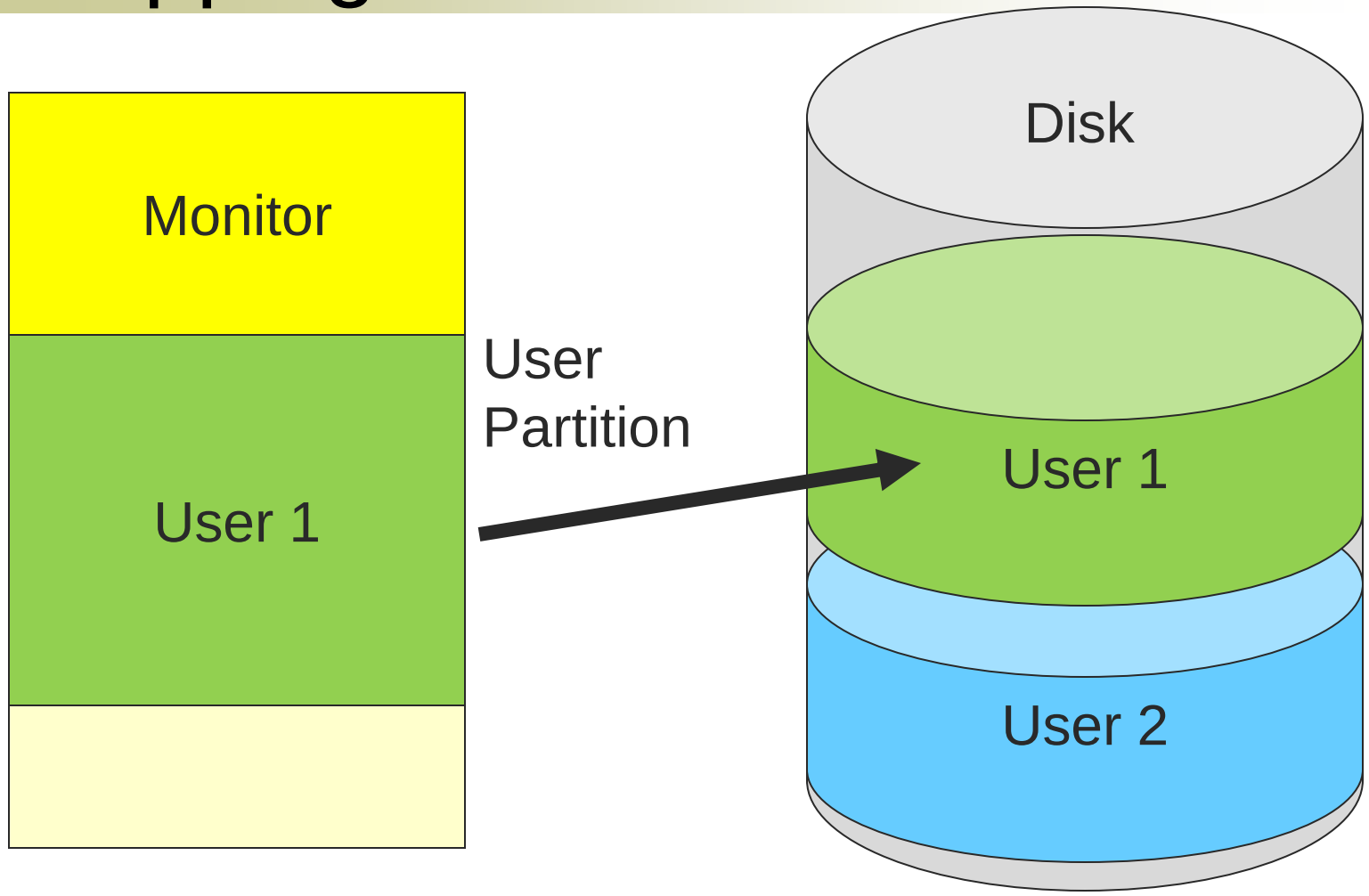
User Partition



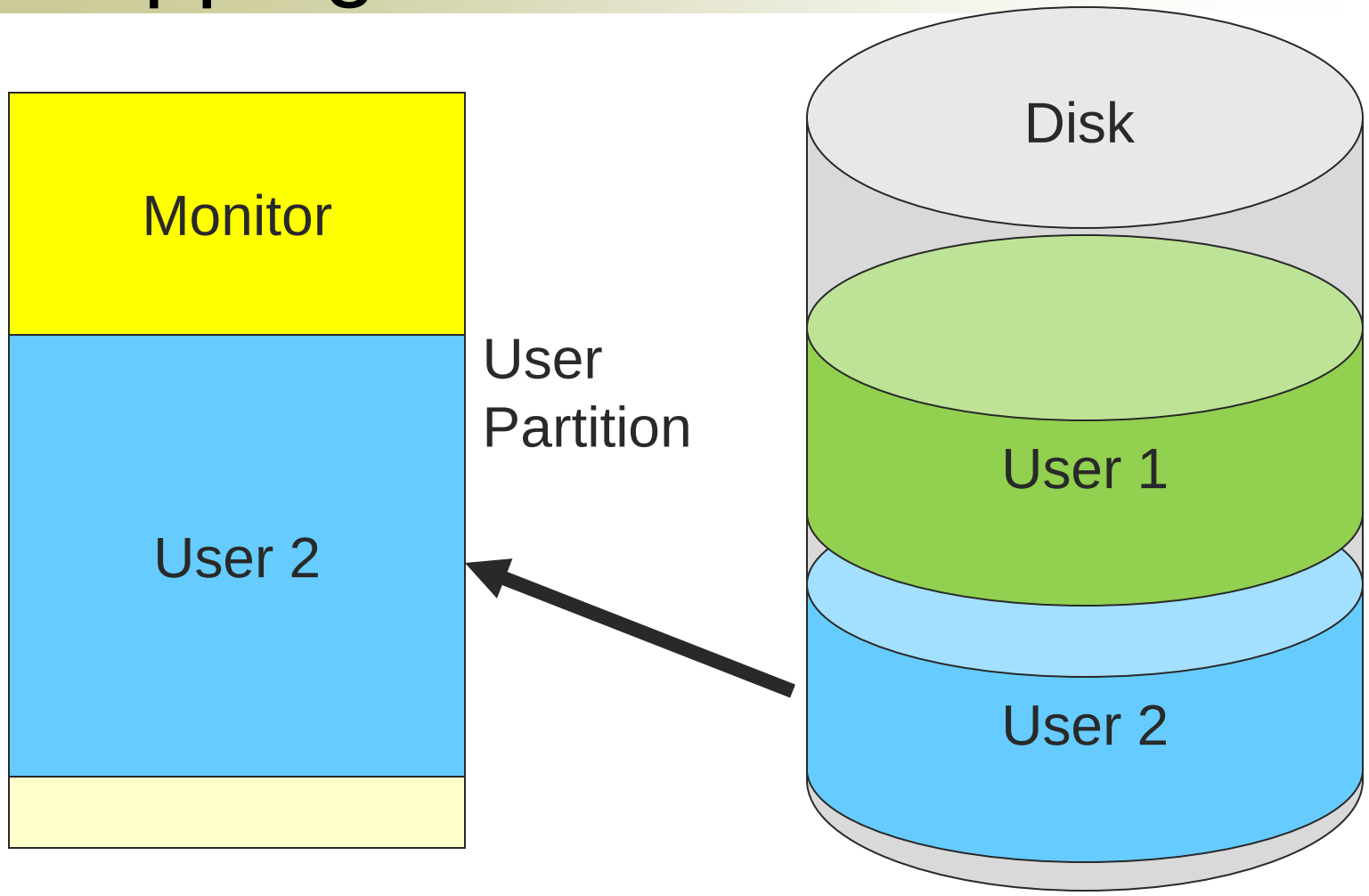
Swapping



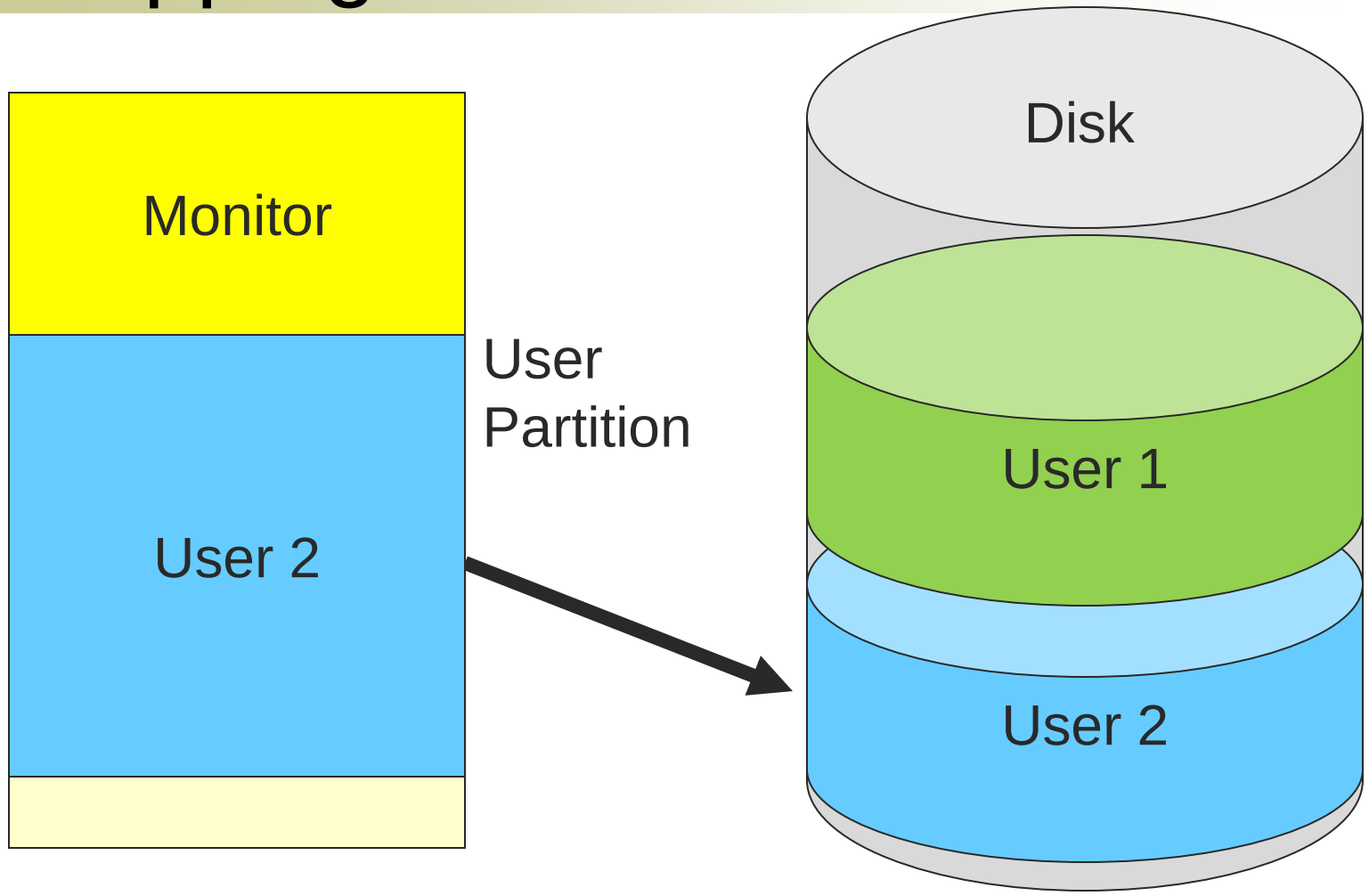
Swapping



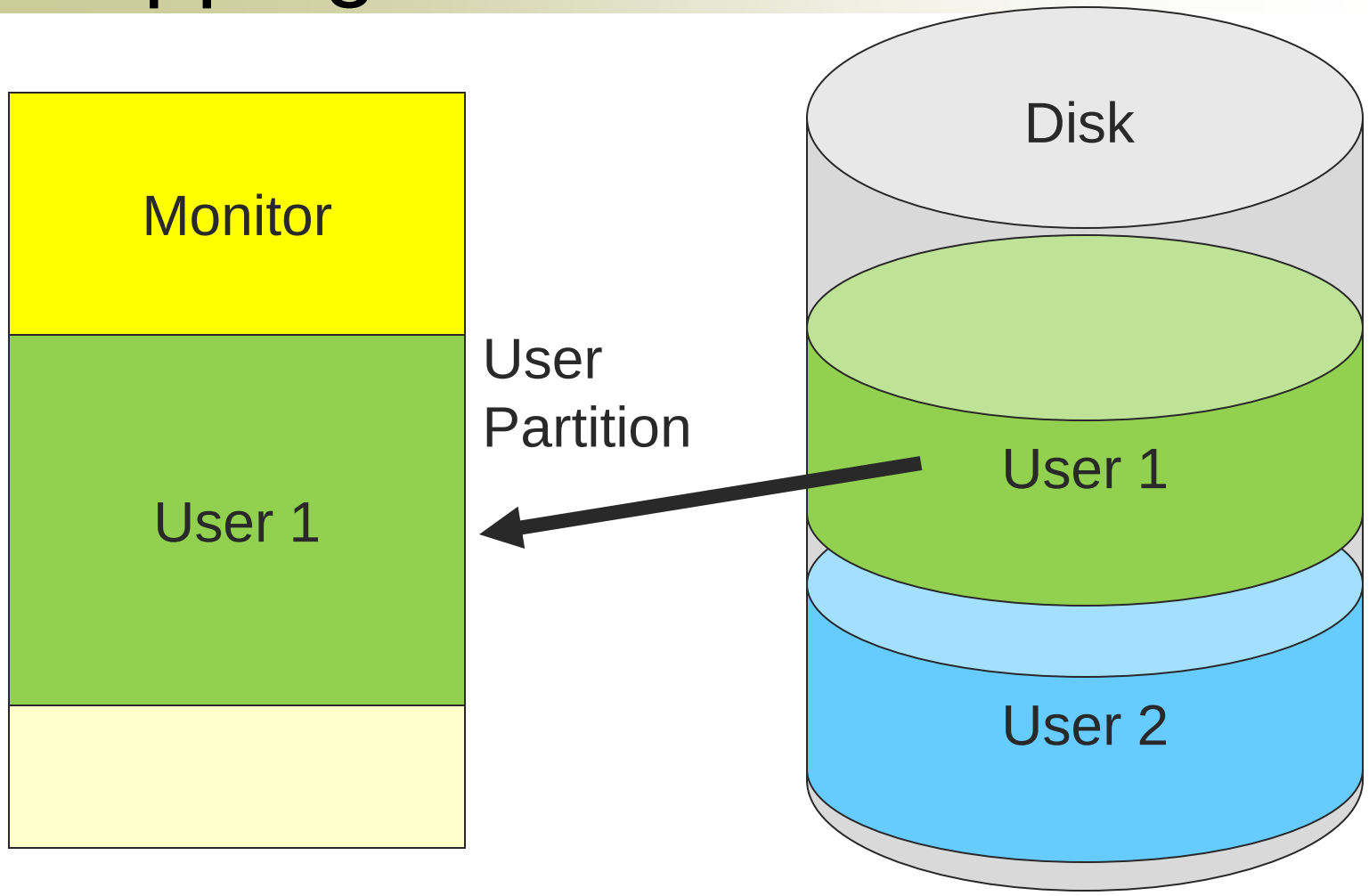
Swapping



Swapping



Swapping



Limitations of Swapping

- Problems with swapping
  - Process must fit into physical memory (impossible to run larger processes)
  - Memory becomes fragmented
    - External fragmentation
      - Lots of small free areas
    - Compaction
      - Reassemble larger free areas
  - Processes are either in memory or on disk: half and half doesn't do any good