CS 423
Operating System Design: Virtual Memory Mgmt

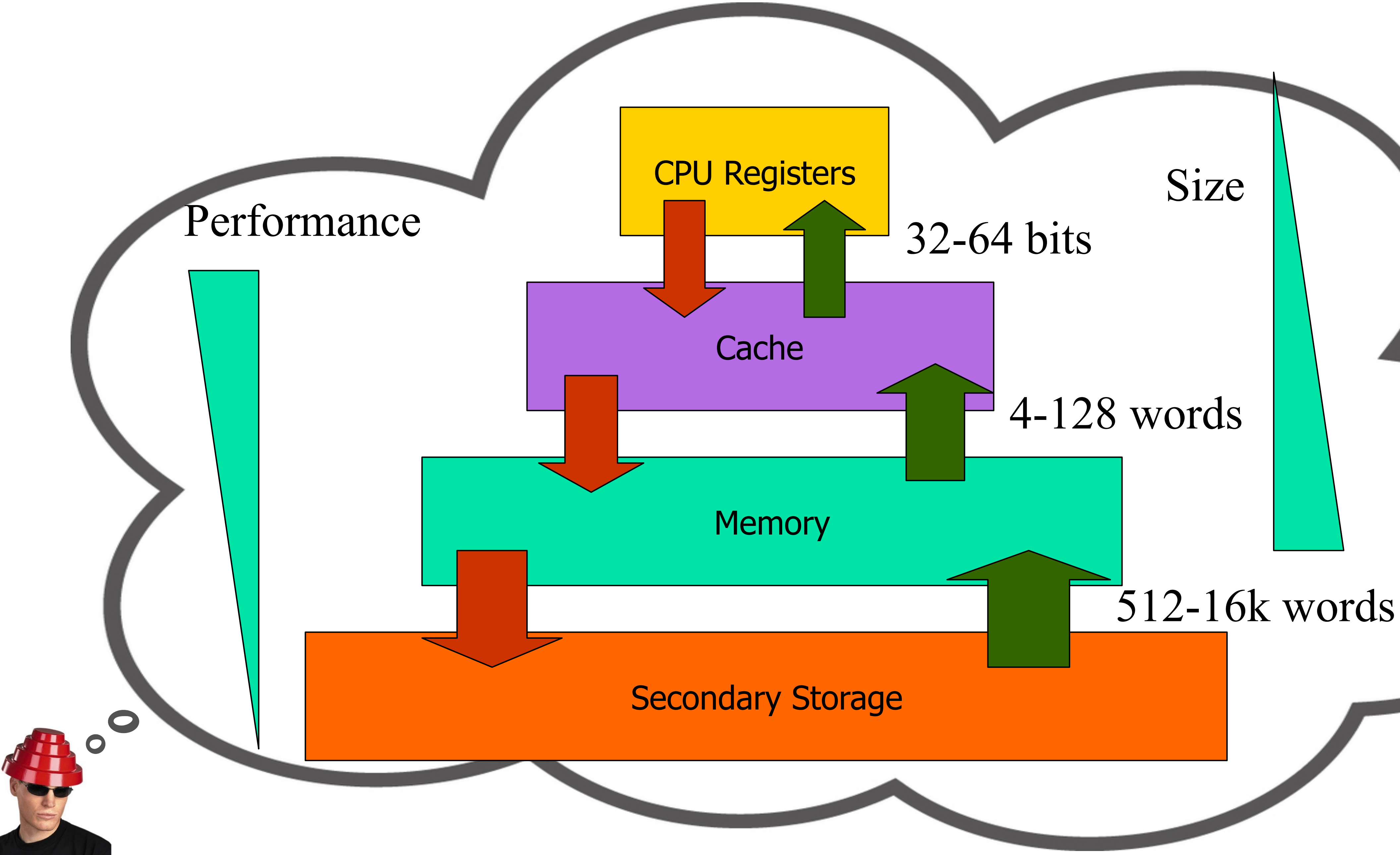
Professor Adam Bates
Spring 2018

Goals for Today

- Learning Objective:
  - Navigate the history of memory systems in OS
- Announcements, etc:
  - MP1 Due Tonight!
  - MP2 Out later This Week

Reminder: Please put away devices at the start of class

Storage Hierarchy

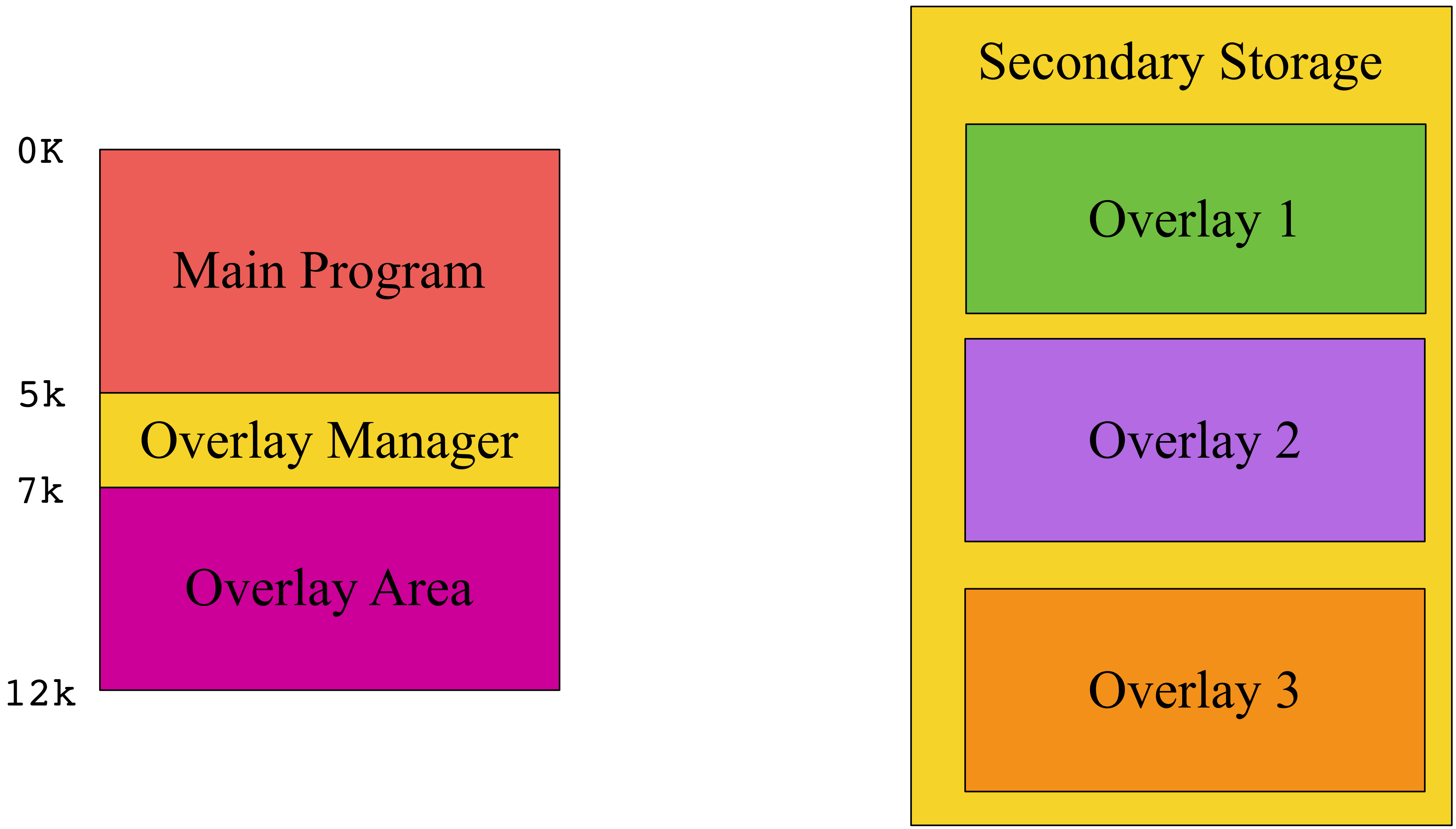


Problem Statement

We have limited amounts of fast resources,
and large amounts of slower resources…

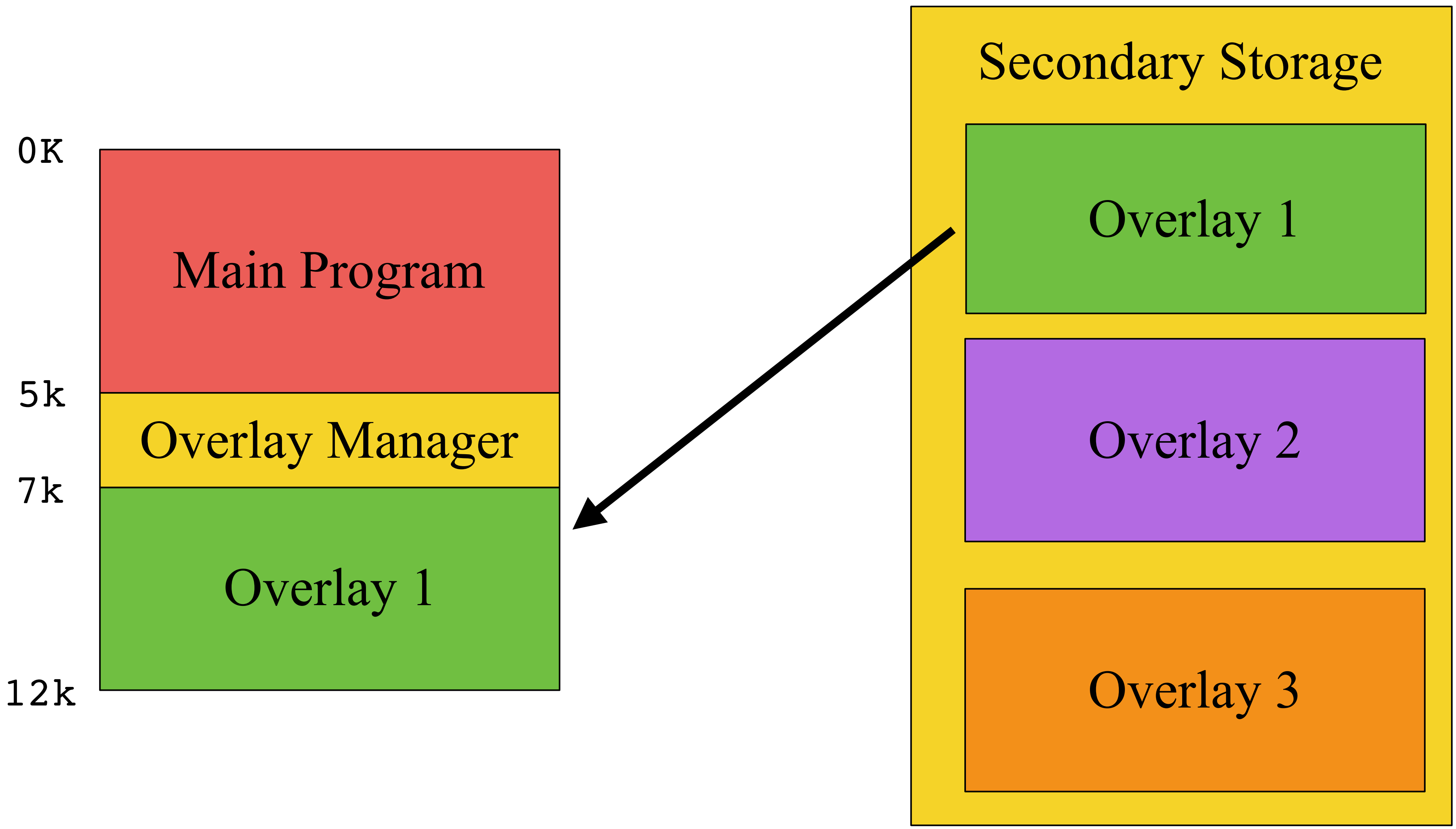
How to create the illusion of an abundant fast resource?

History: Mem Overlays



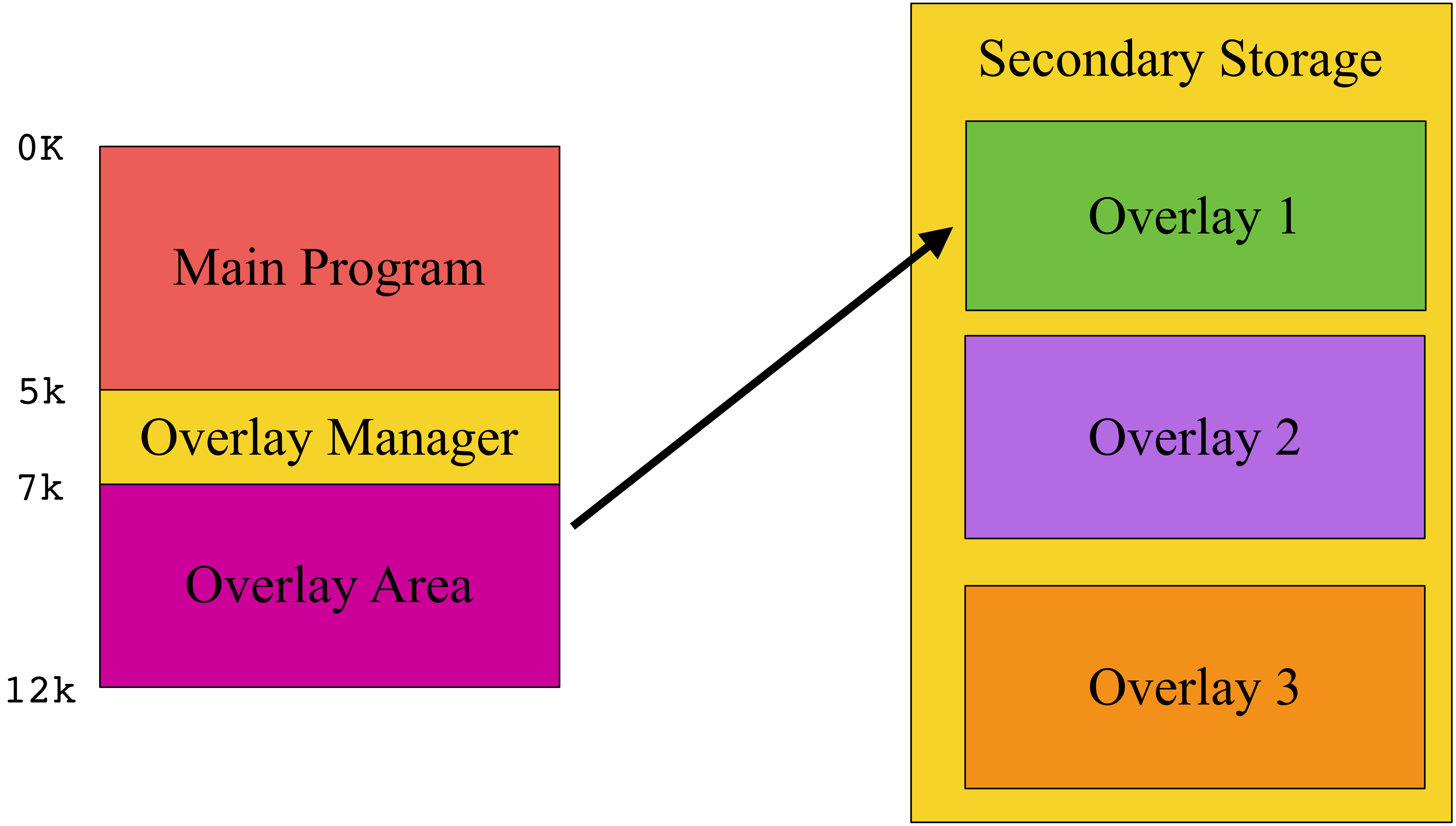
Used when process memory requirement exceeded the physical memory space

History: Mem Overlays



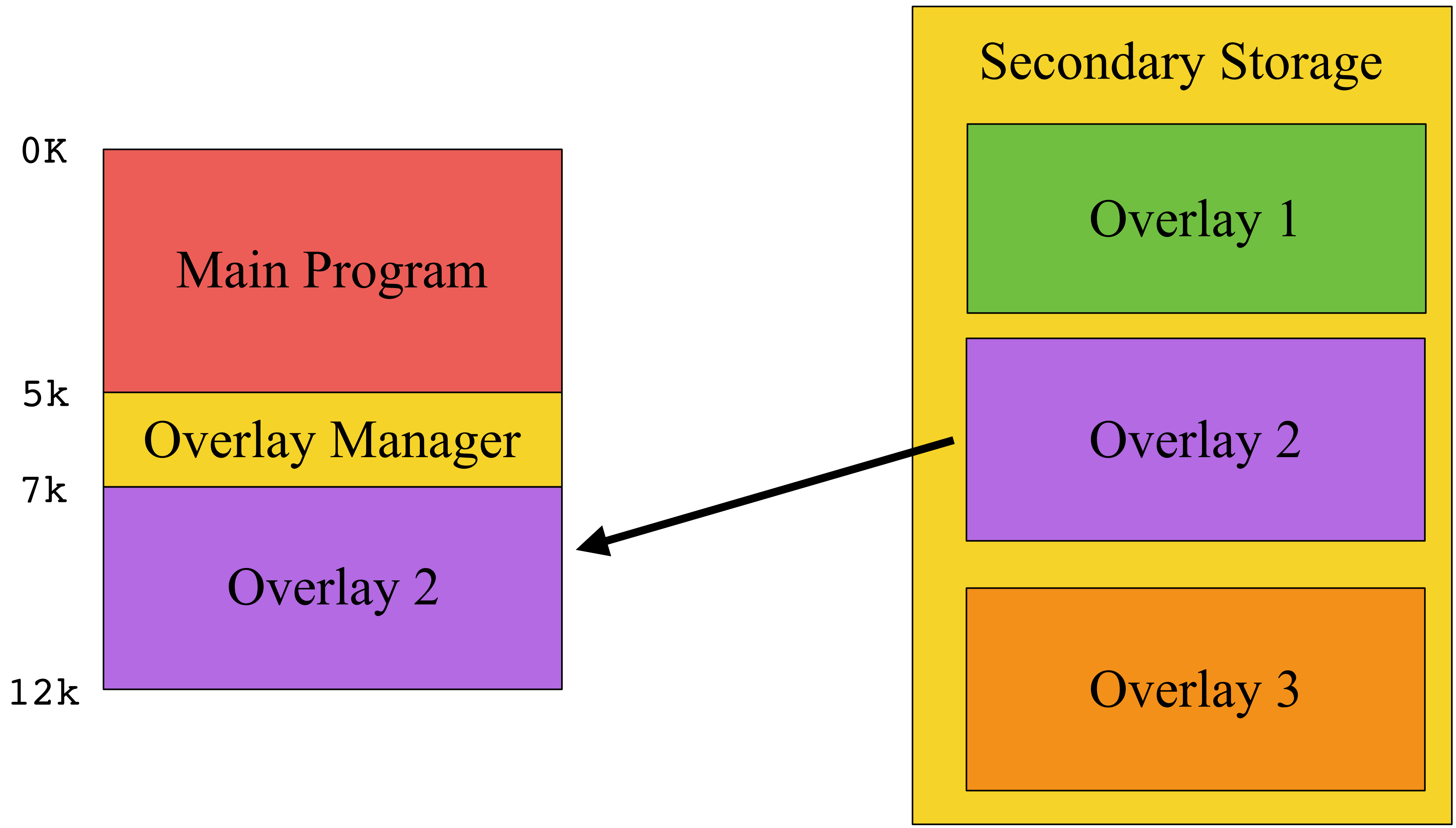
Used when process memory requirement exceeded the physical memory space

History: Mem Overlays



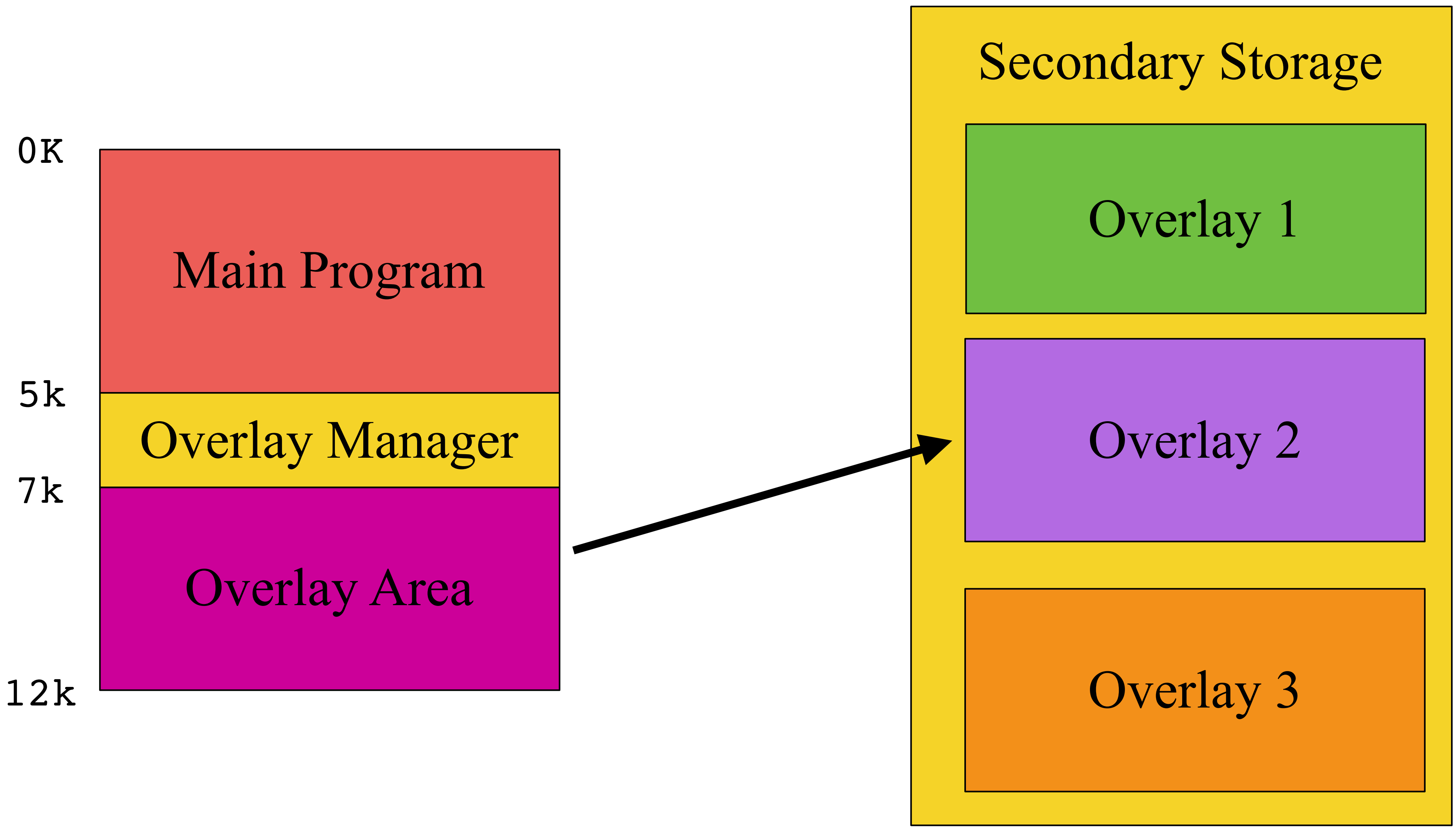
Used when process memory requirement exceeded the physical memory space

History: Mem Overlays



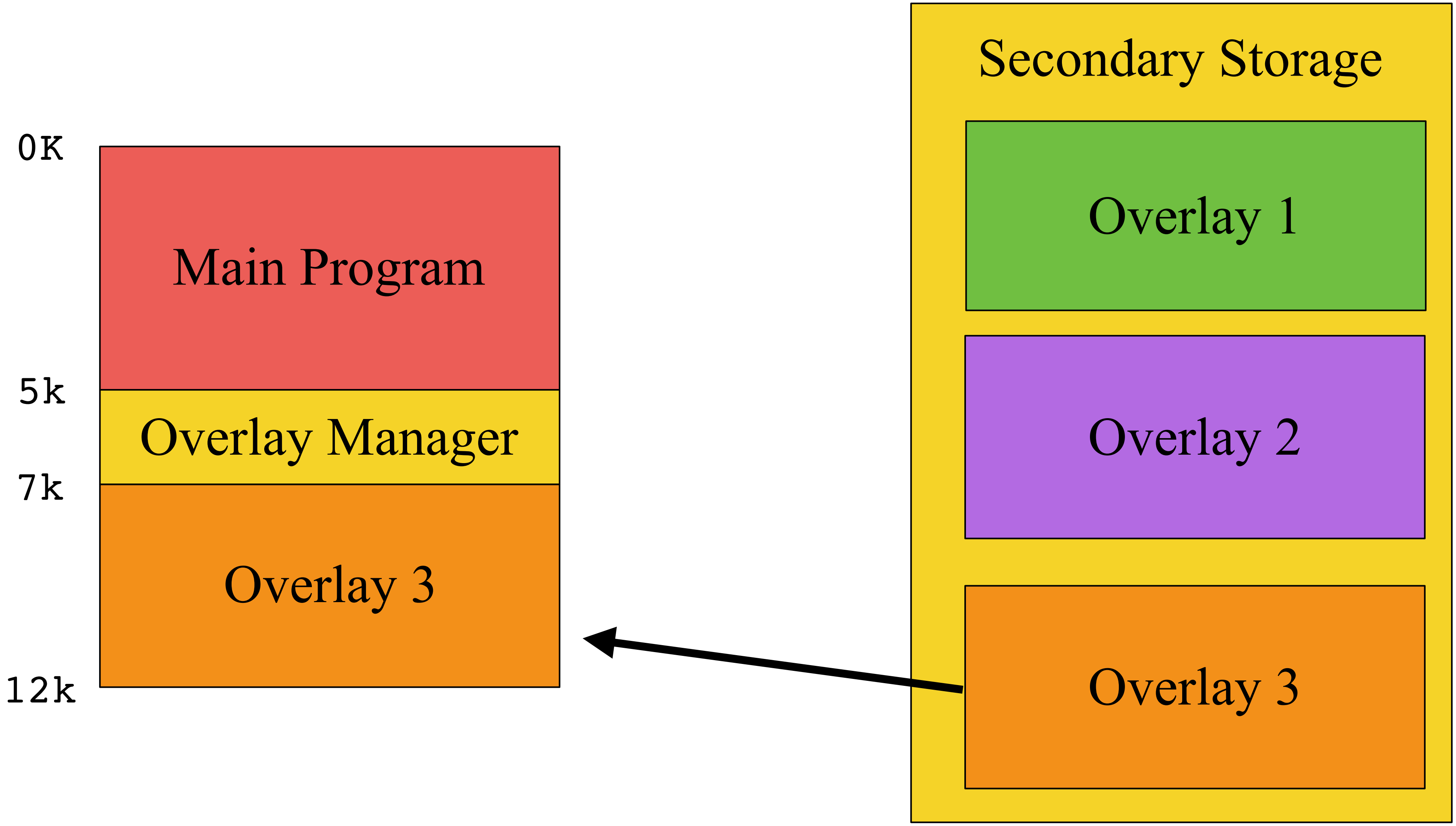
Used when process memory requirement exceeded the physical memory space

History: Mem Overlays



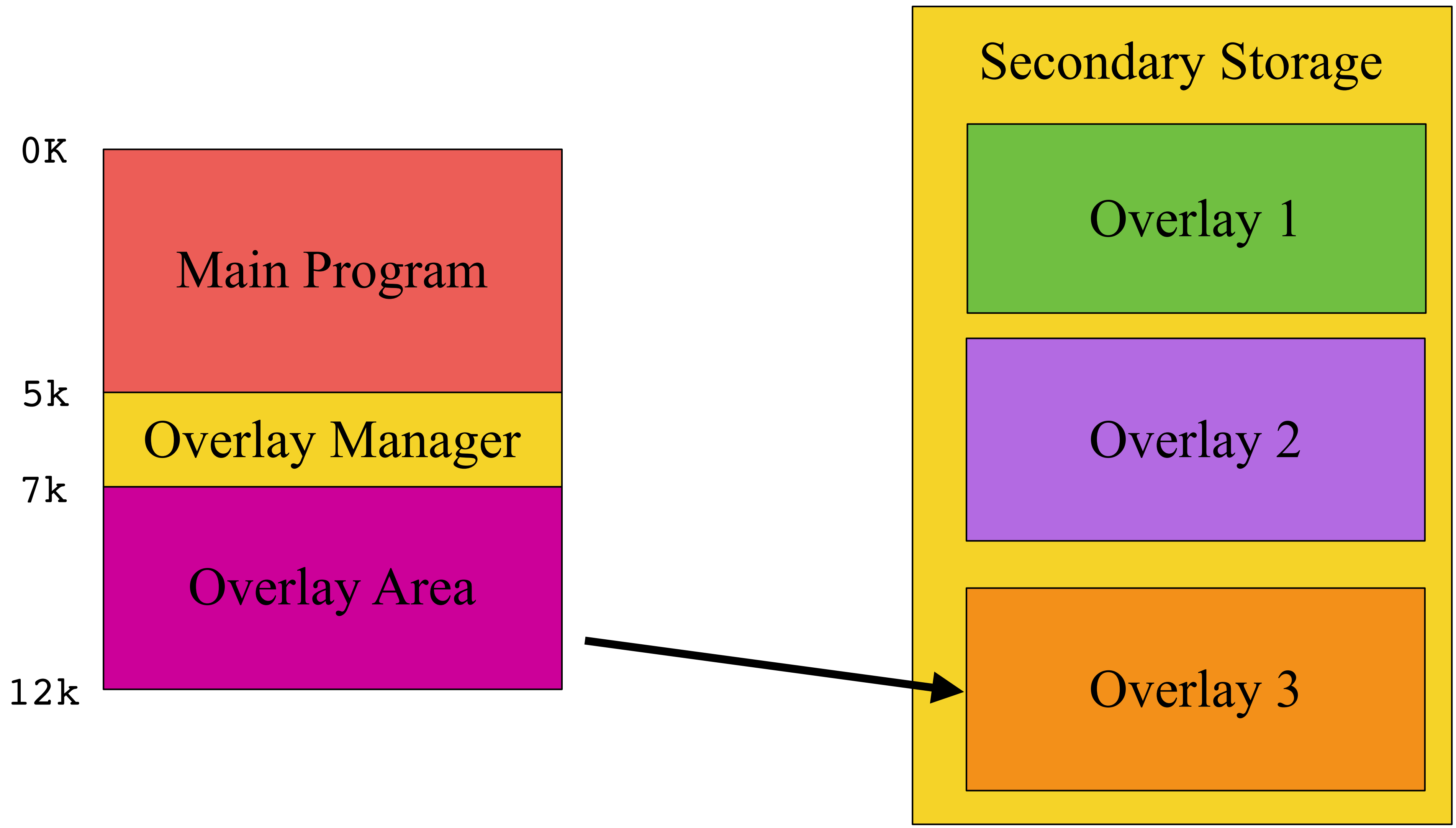
Used when process memory requirement exceeded the physical memory space

History: Mem Overlays



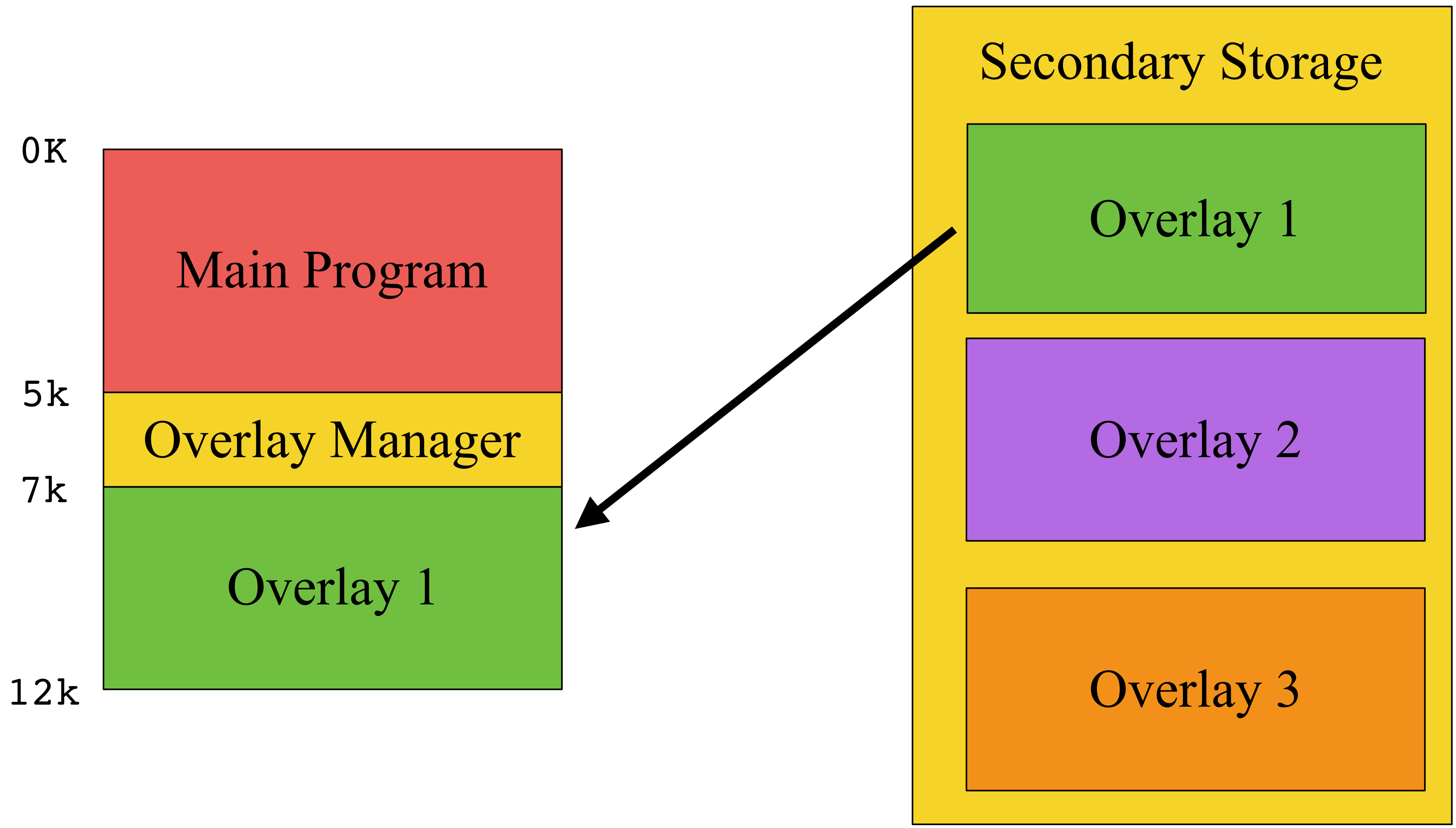
Used when process memory requirement exceeded the physical memory space

History: Mem Overlays



Used when process memory requirement exceeded the physical memory space

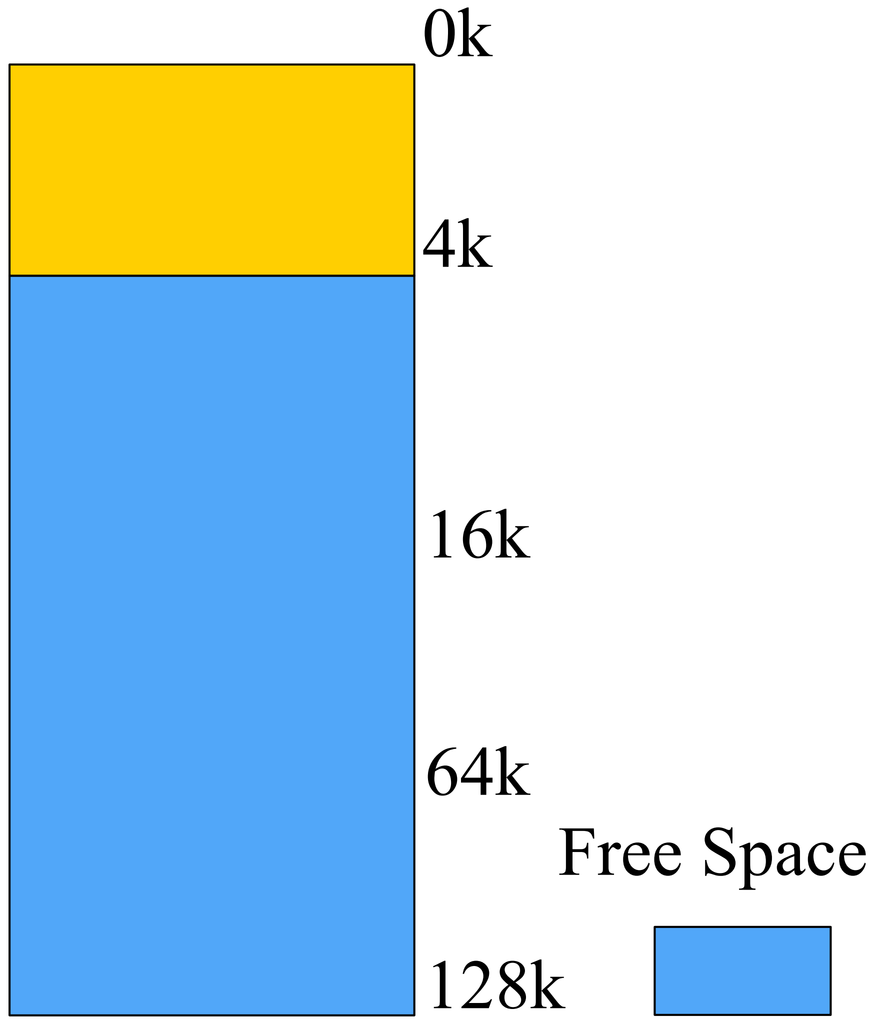
History: Mem Overlays



Used when process memory requirement exceeded the physical memory space

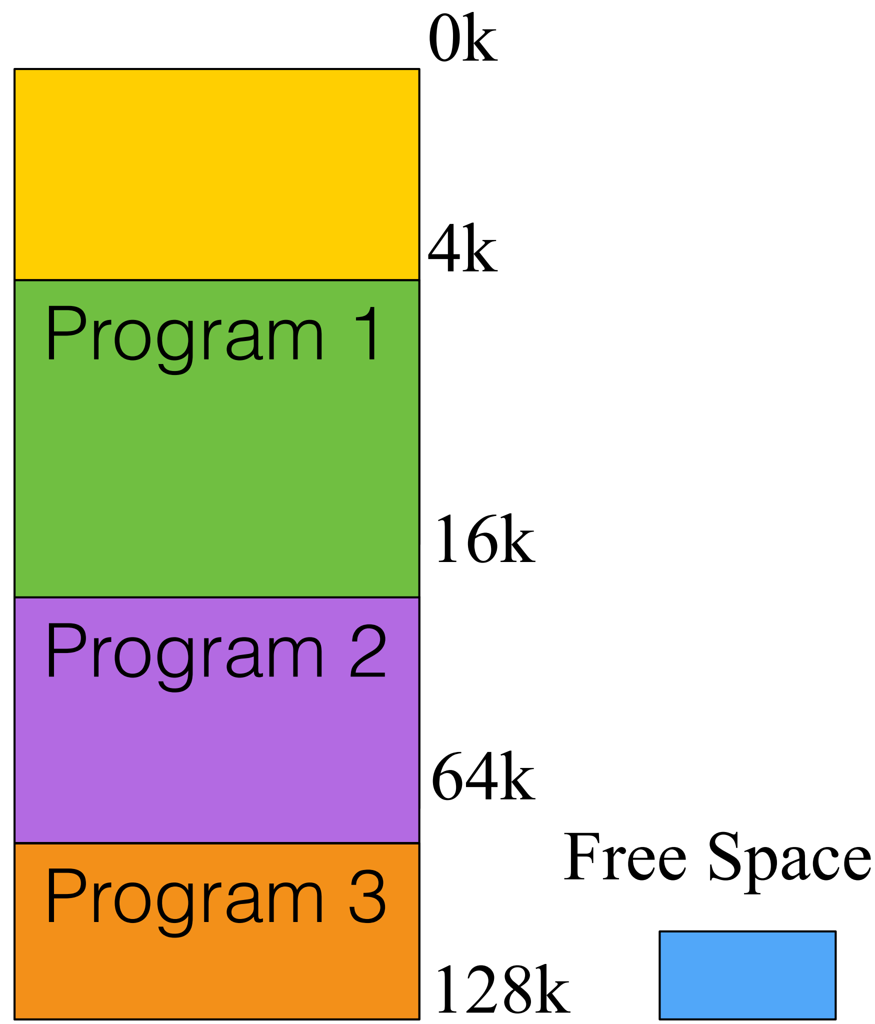
History: Fixed Partitions

- Approach: Multiprogramming with fixed memory partitions
- Divides memory into *n* fixed partitions (possible unequal)
- Problem?



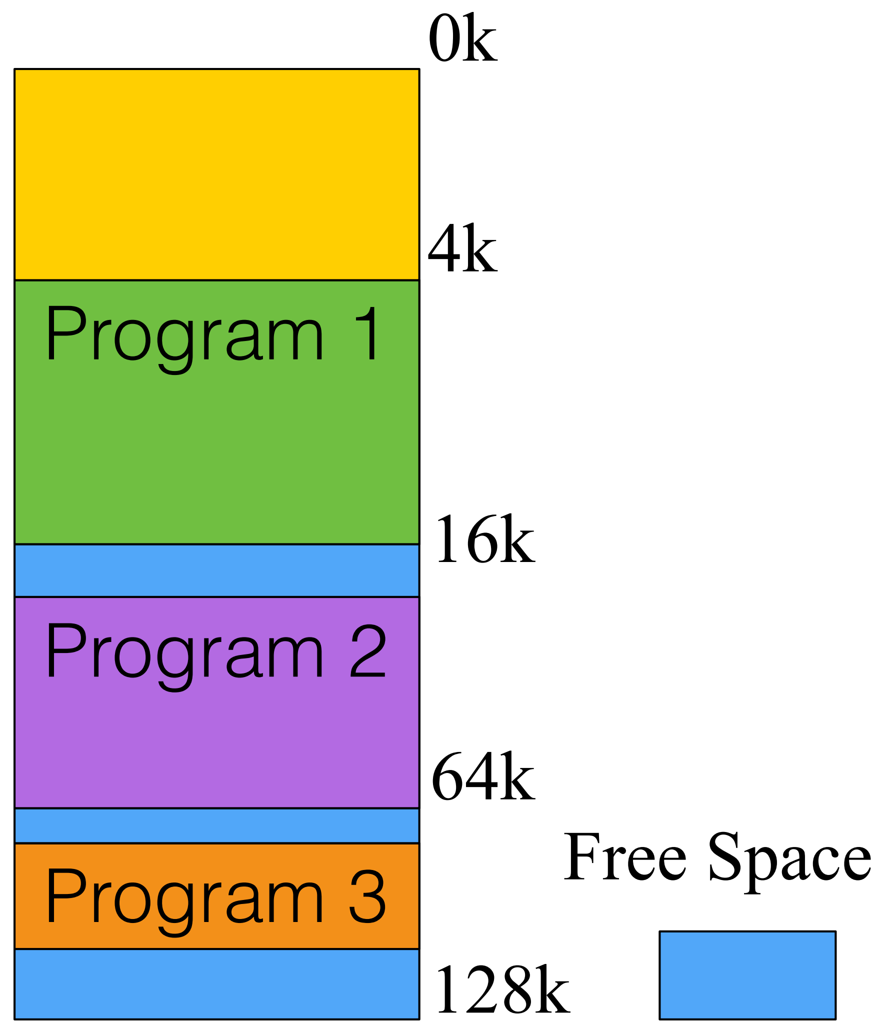
History: Fixed Partitions

- Approach: Multiprogramming with fixed memory partitions
- Divides memory into *n* fixed partitions (possible unequal)
- Problems?



History: Fixed Partitions

- Approach: Multiprogramming with fixed memory partitions
- Divides memory into *n* fixed partitions (possible unequal)
- Problems?
  - Internal Fragmentation! Also,
  - Level of Multiprogramming



History: Fixed Partition Allocation

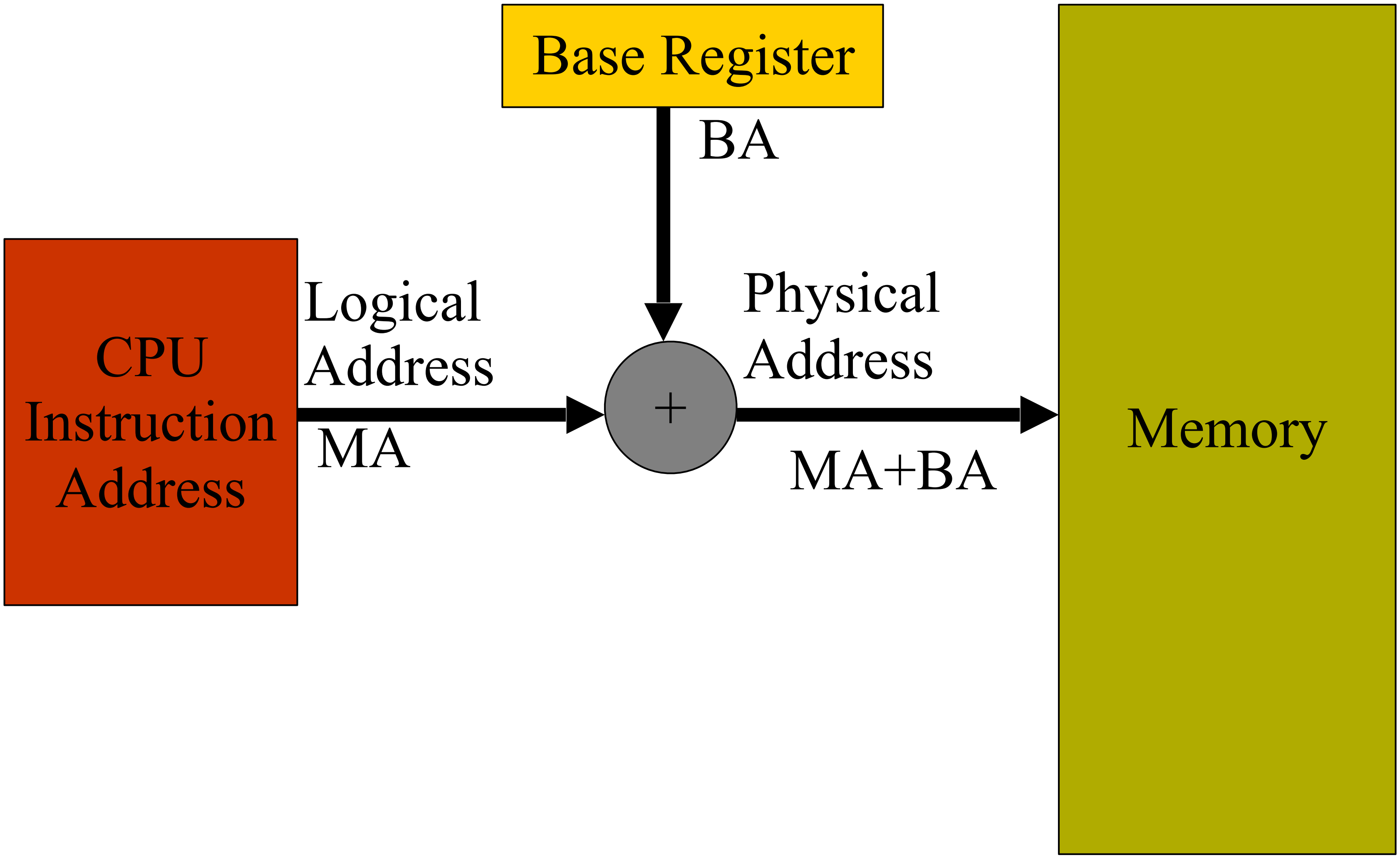
Placement Algorithms for Fixed Partitions

- Trivial for equal size partitions. For unequal…
- Multiple Queues:
  - Assign (i.e., enqueue) each incoming job to the smallest partition within which it fits
  - Decreases fragmentation
  - when the queue for a large partition is empty but the queue for a small partition is full. Small jobs have to wait to get into memory even though plenty of memory is free.
- Single Queue:
  - Assign each process to the smallest **available** partition within which it fits
  - Increases amount of multiprogramming on the expense of fragmentation

History: Relocation

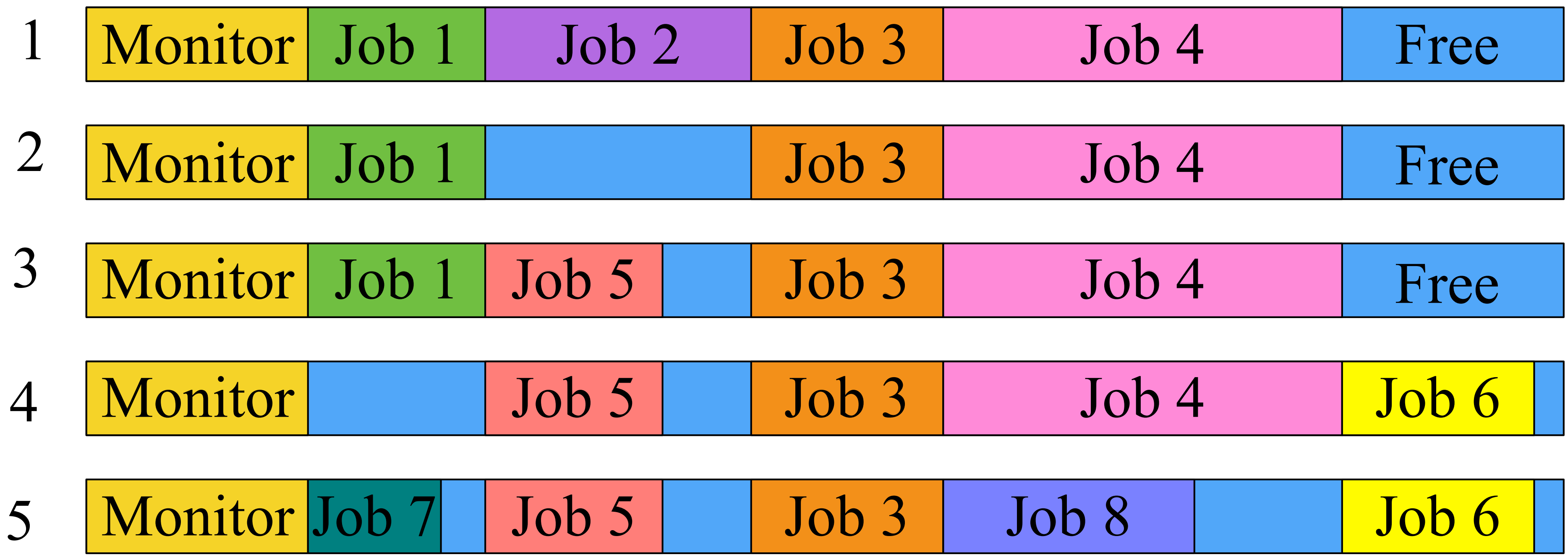
- Correct starting address when a program should start in the memory
- Different jobs will run at different addresses
  - When a program is linked, the linker must know at what address the program will begin in memory.
- Logical addresses
  - Logical address space , range (0 to max)
  - Physical addresses, Physical address space range (R+0 to R+max) for base value R.
  - User program never sees the real physical addresses
- Relocation register
  - Mapping requires hardware with the base register

History: Relocation Register



Relocation => "Variable Partition Allocation"

History: Variable Partition Allocation



Memory wasted by External Fragmentation

Bad Parking Analogy

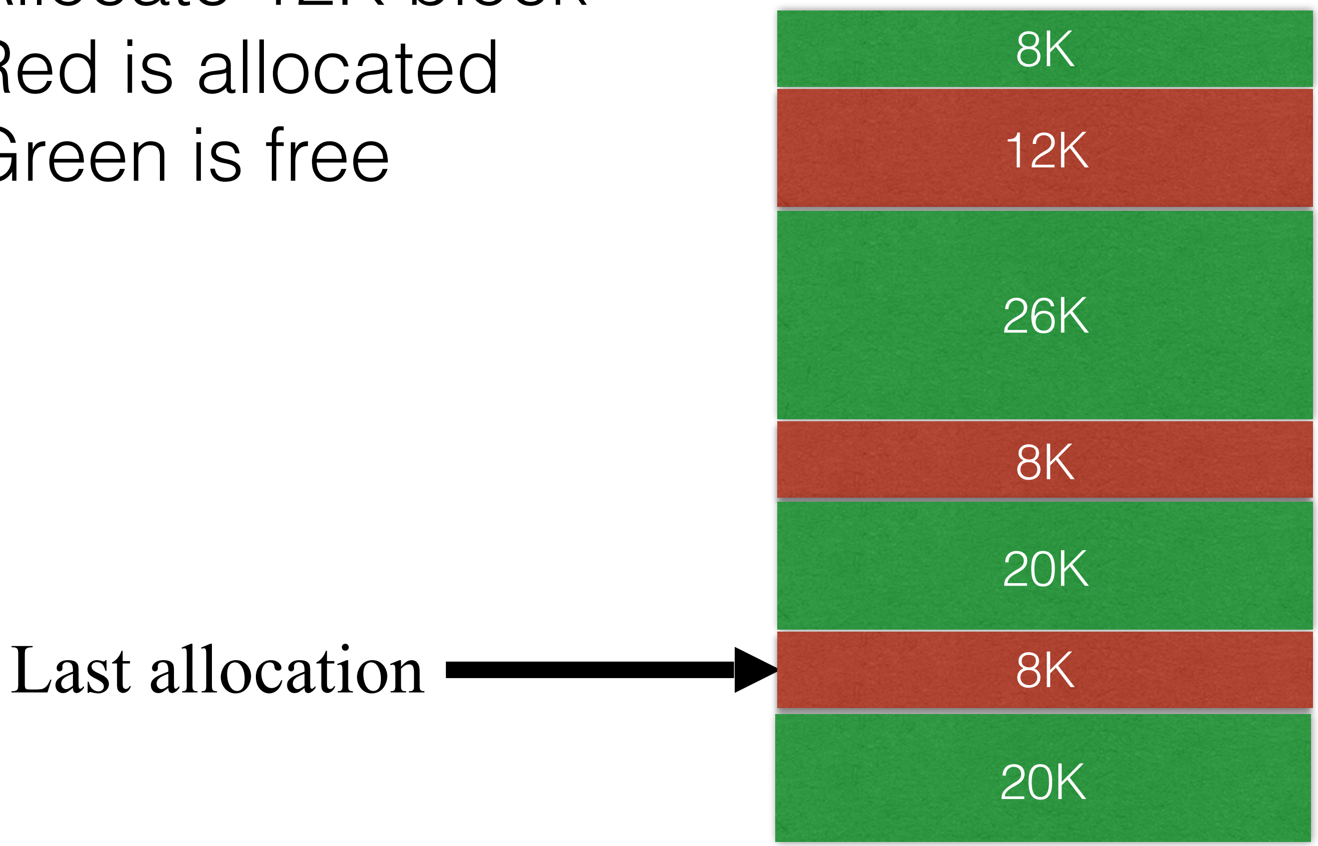
Source: https://xkcd.com/562/

History: Storage Placement Strategy

- **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.
  - Problem: Creates small holes that can't be used.
- **Worst Fit?**
  - Use the largest available hole.
  - Problem: Gets rid of large holes making it difficult to run large programs.
- **First Fit**
  - Use the first available hole whose size is sufficient to meet the need.
  - Problem: Creates average size holes.
- **Next Fit.**
  - Minor variation of first fit: search from the last hole used.
  - Problem: slightly worse performance than first fit.

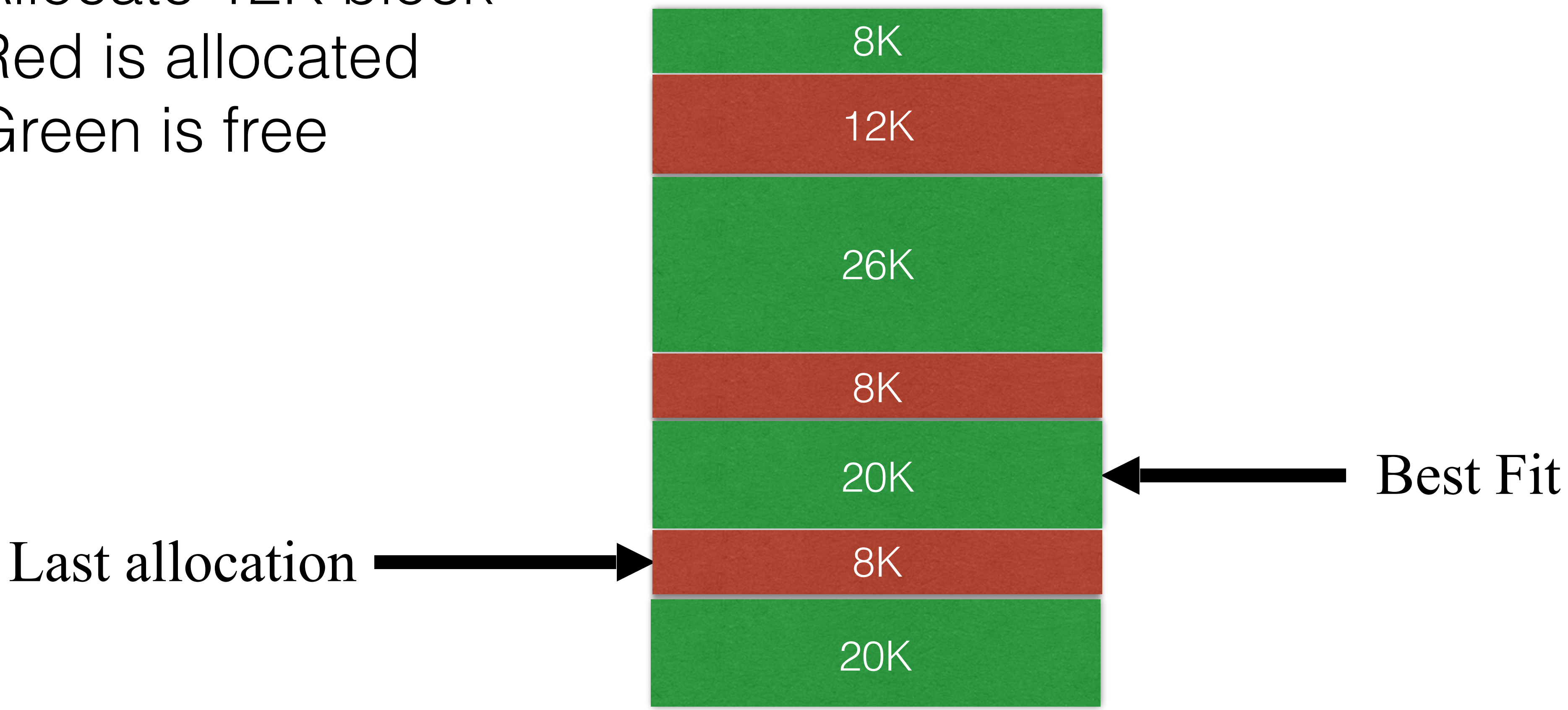
History: Example Policies

- Allocate 12K block
- Red is allocated
- Green is free



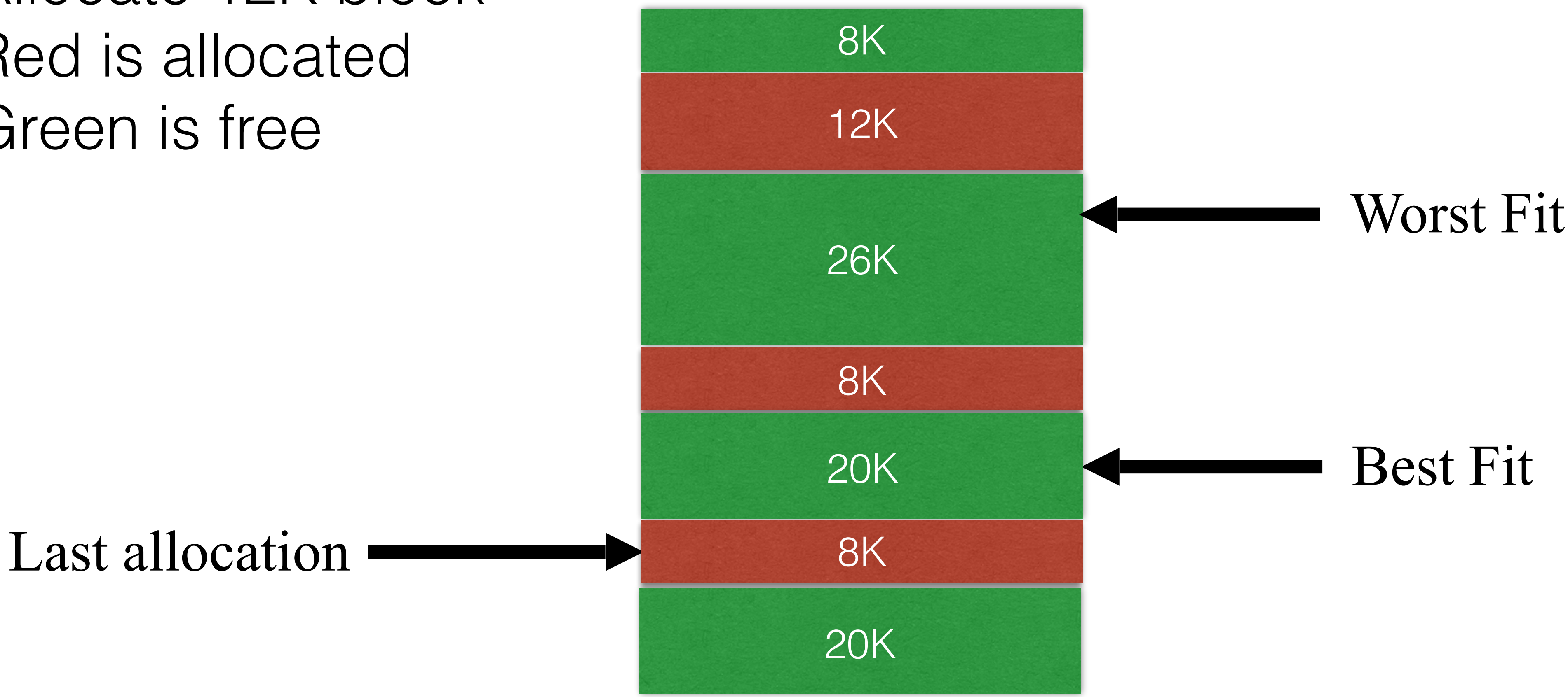
History: Example Policies

- Allocate 12K block
- Red is allocated
- Green is free



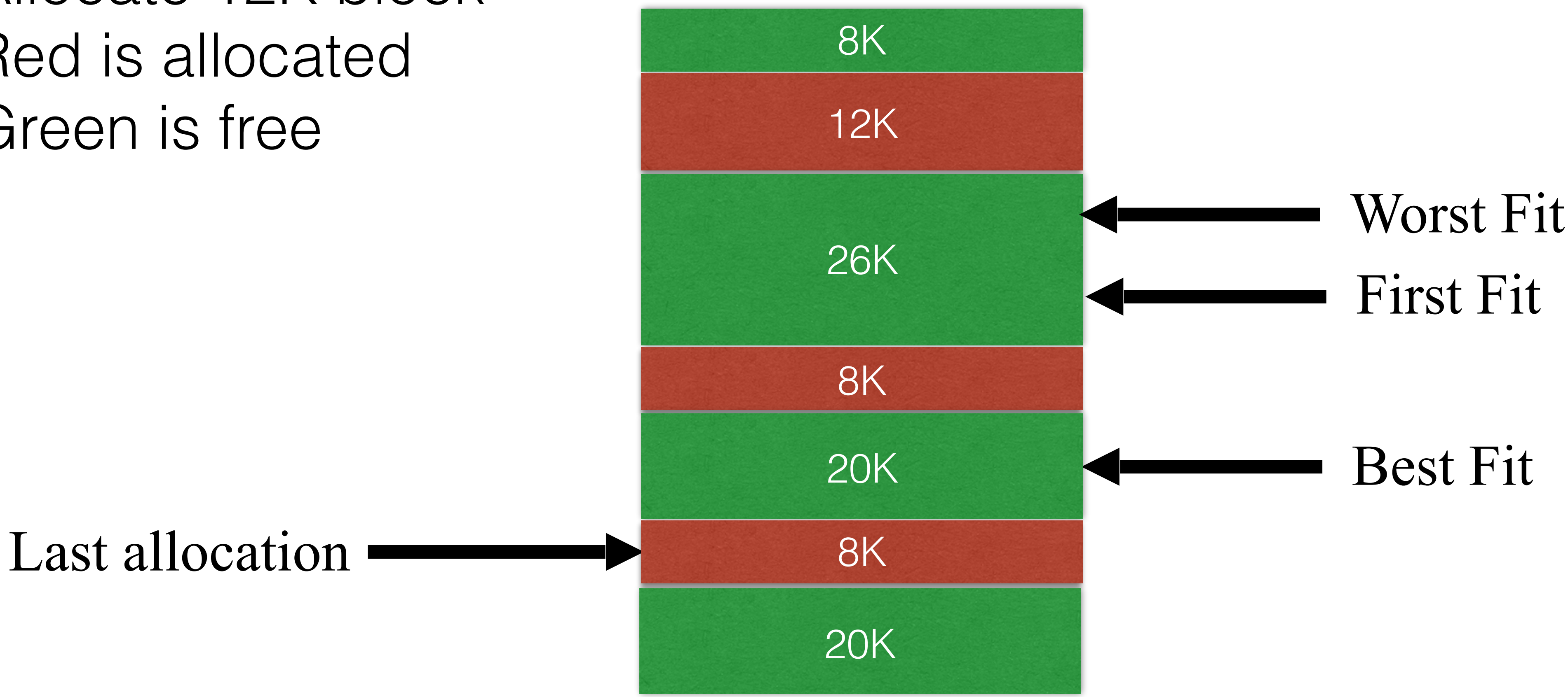
History: Example Policies

- Allocate 12K block
- Red is allocated
- Green is free



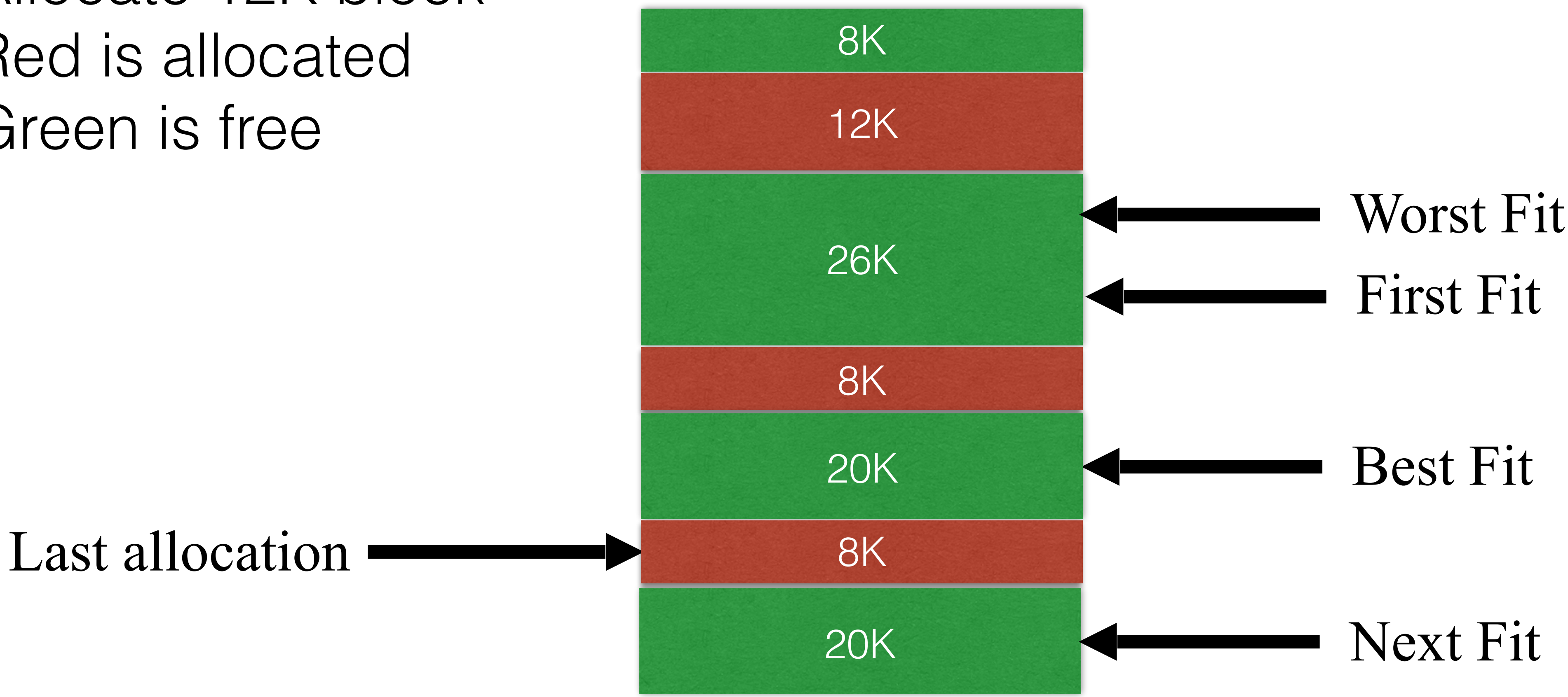
History: Example Policies

- Allocate 12K block
- Red is allocated
- Green is free



History: Example Policies

- Allocate 12K block
- Red is allocated
- Green is free



History: Summary

Overlay → Fixed Partitions → Relocation

Overlay
- No multi-programming support

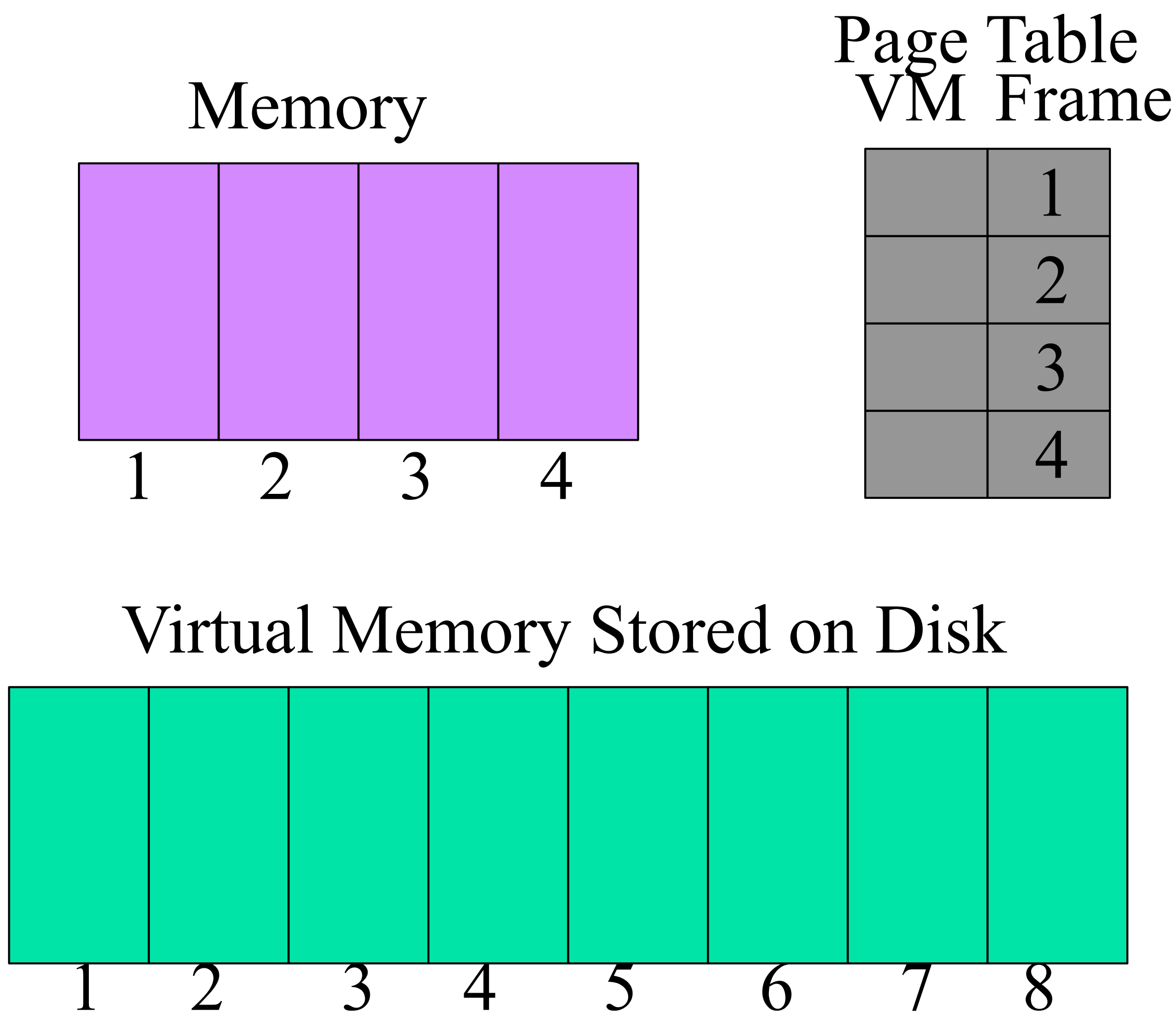
Fixed Partitions
- Supports multi-programming
- Internal fragmentation

Relocation
- No internal fragmentation
- Introduces external fragmentation

Virtual Memory

- Provide user with virtual memory that is as big as user needs
- Store virtual memory on disk
- Cache parts of virtual memory being used in real memory
- Load and store cached virtual memory without user program intervention

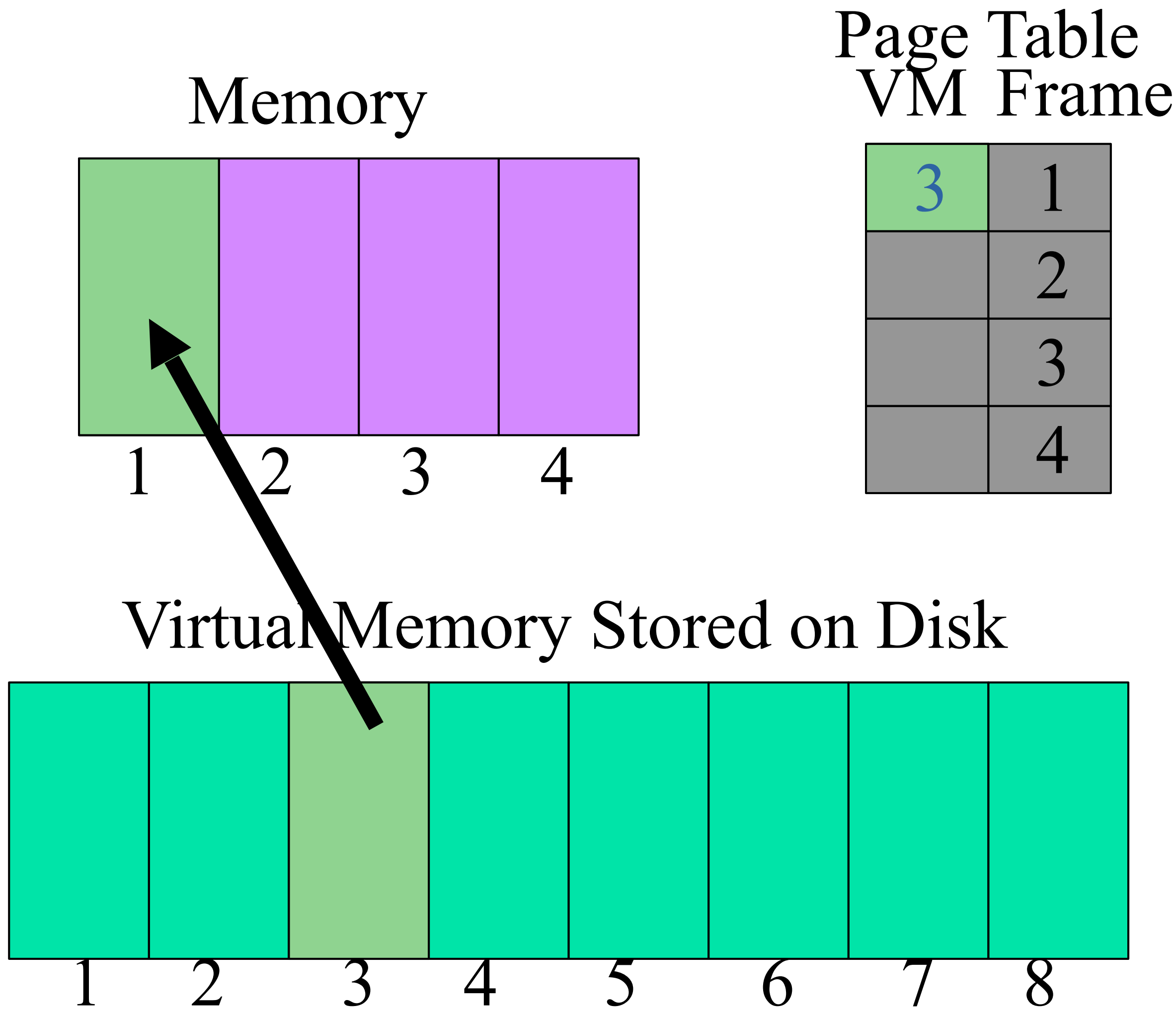
Paging



Memory
1
2
3
4
Page Table
VM
Frame
1
2
3
4
Virtual Memory Stored on Disk
1
2
3
4
5
6
7
8

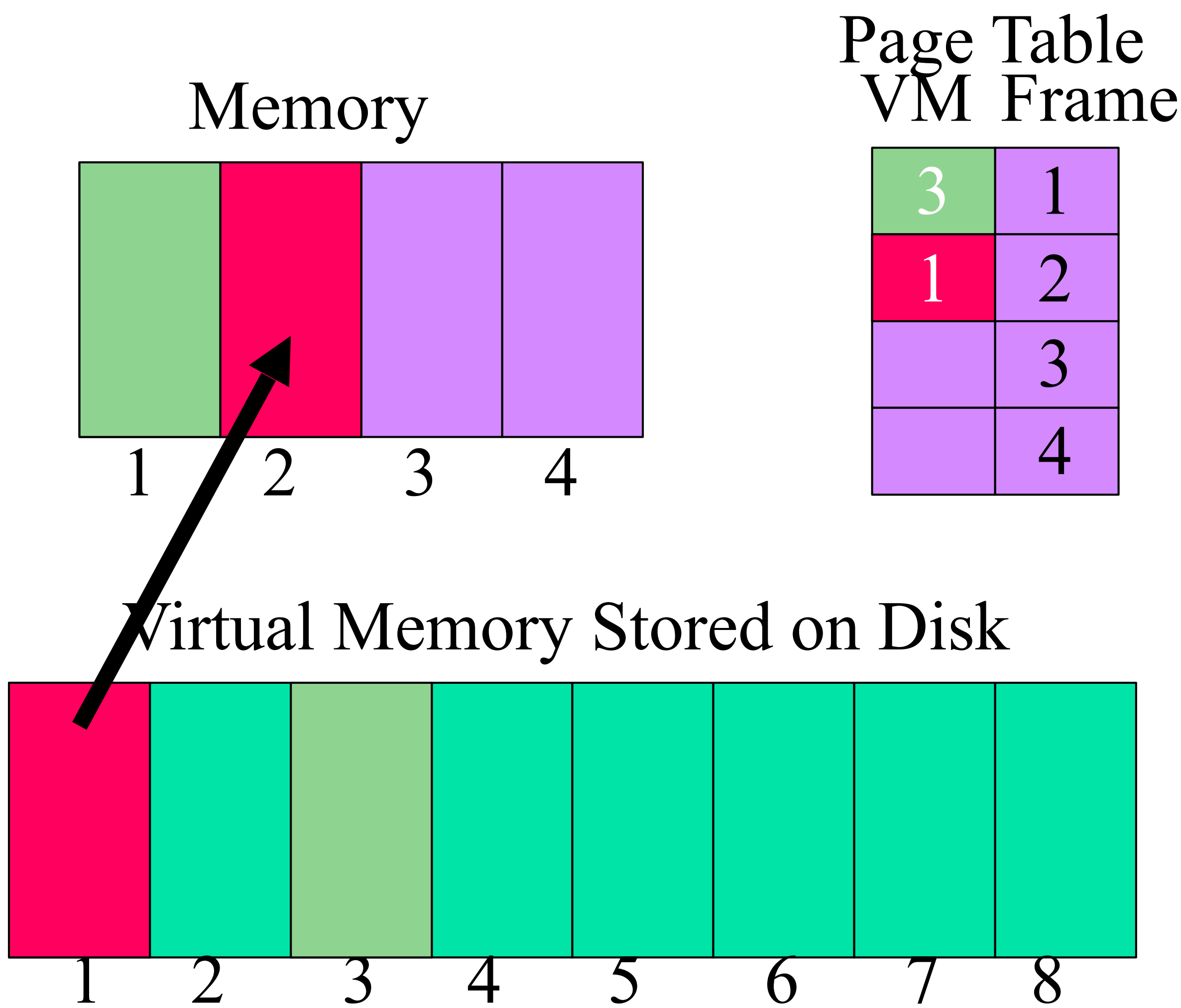
Paging

Request Page 3…



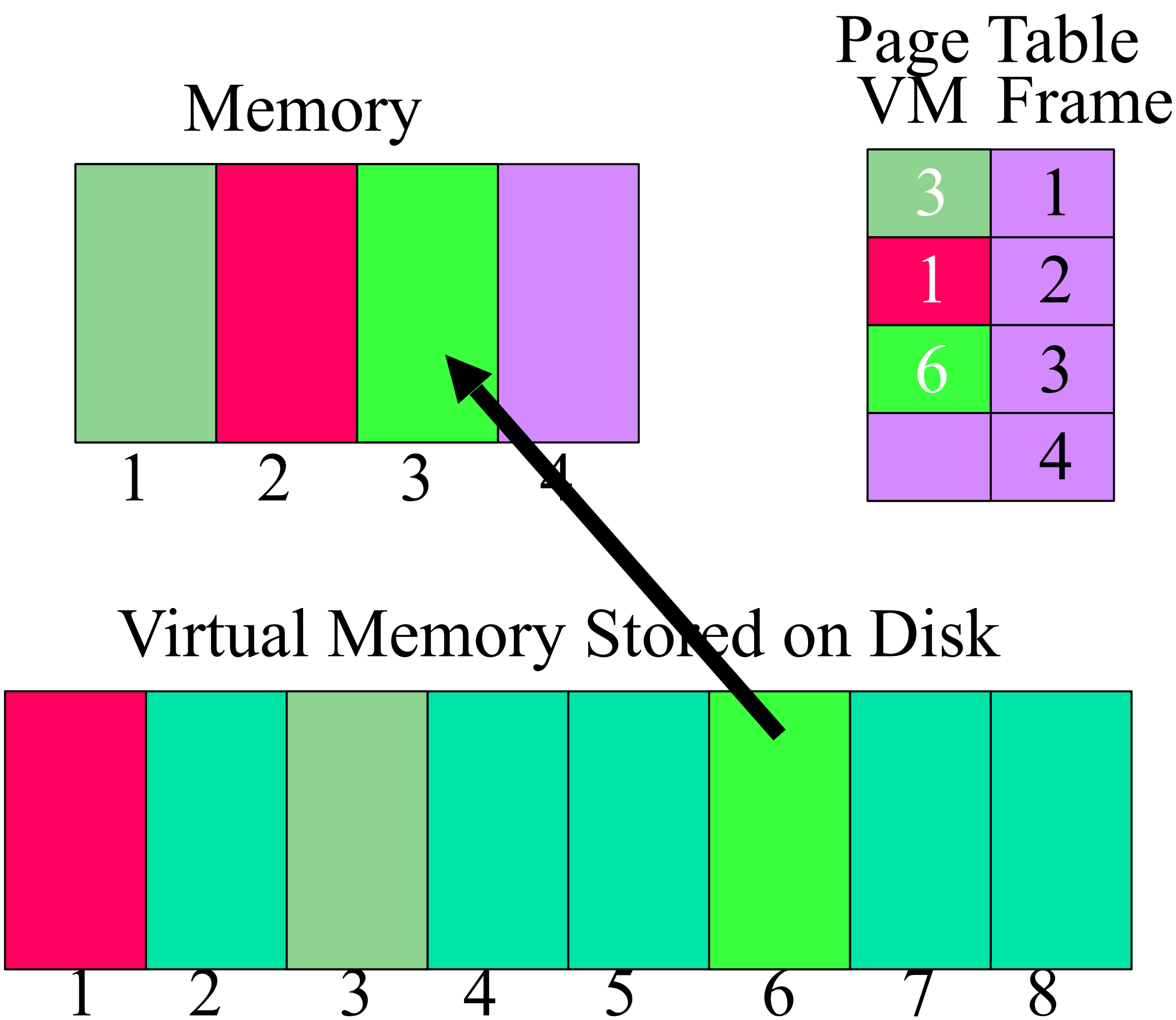
Paging

Request Page 1…



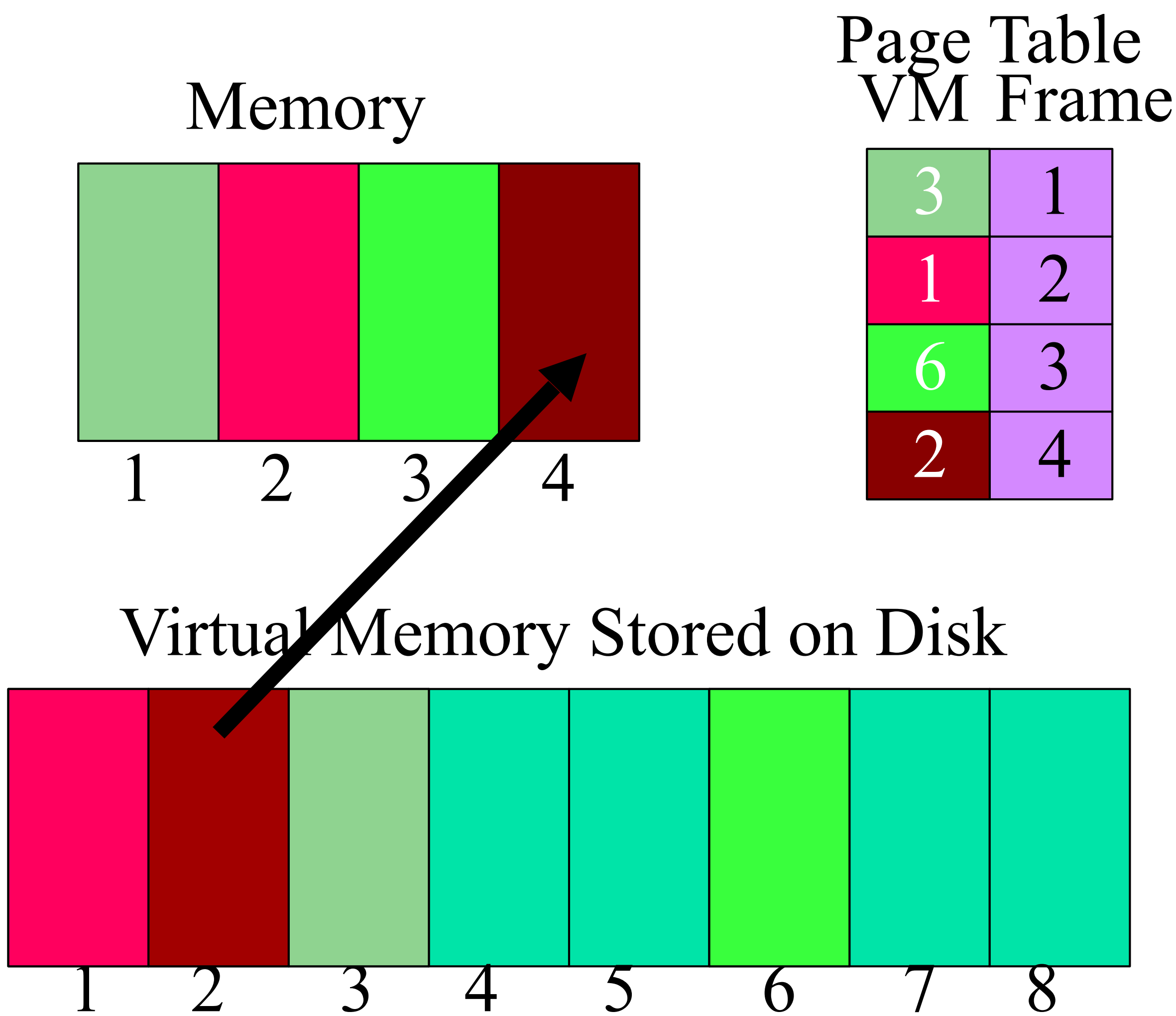
Paging

Request Page 6…



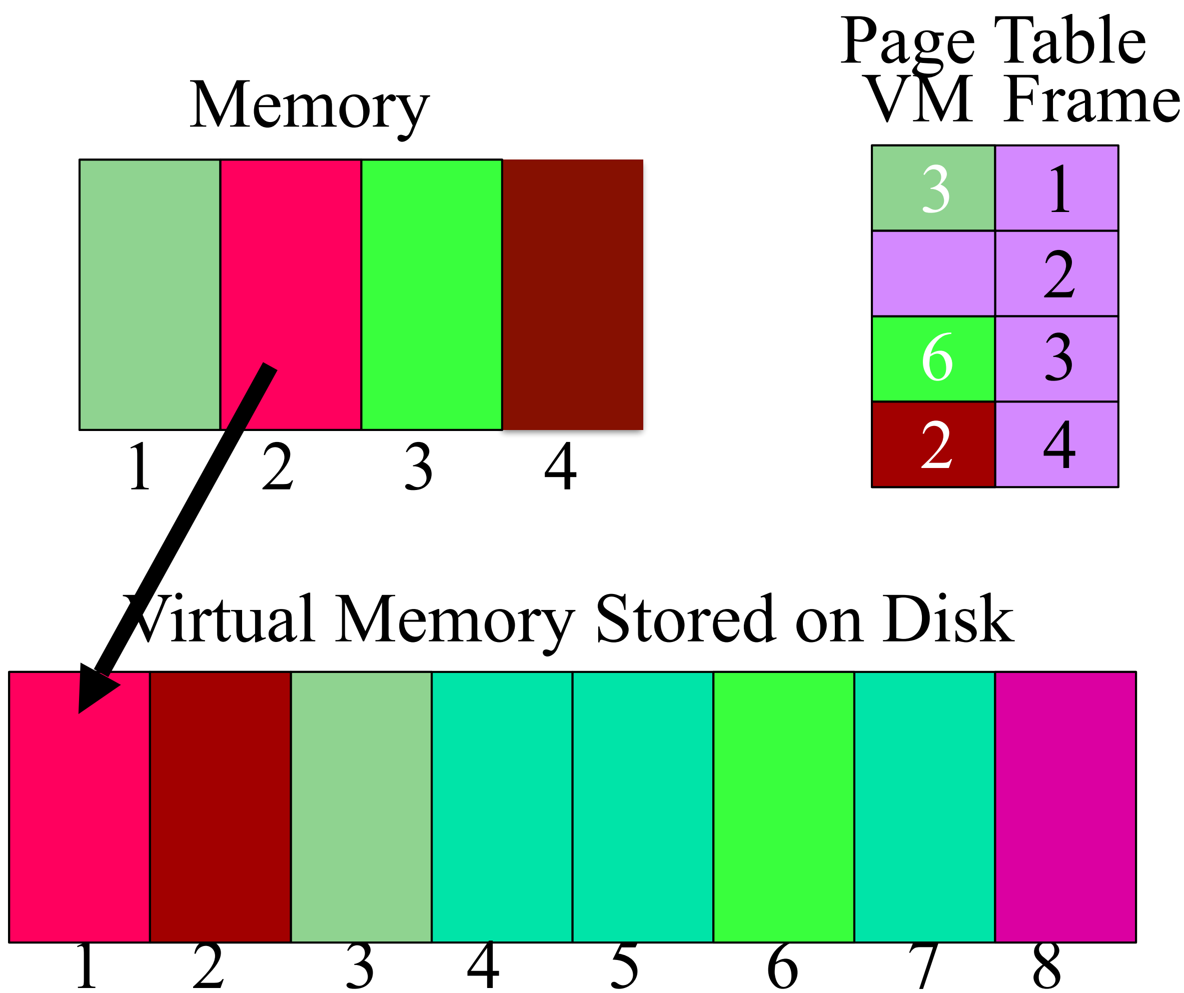
Paging

Request Page 2…



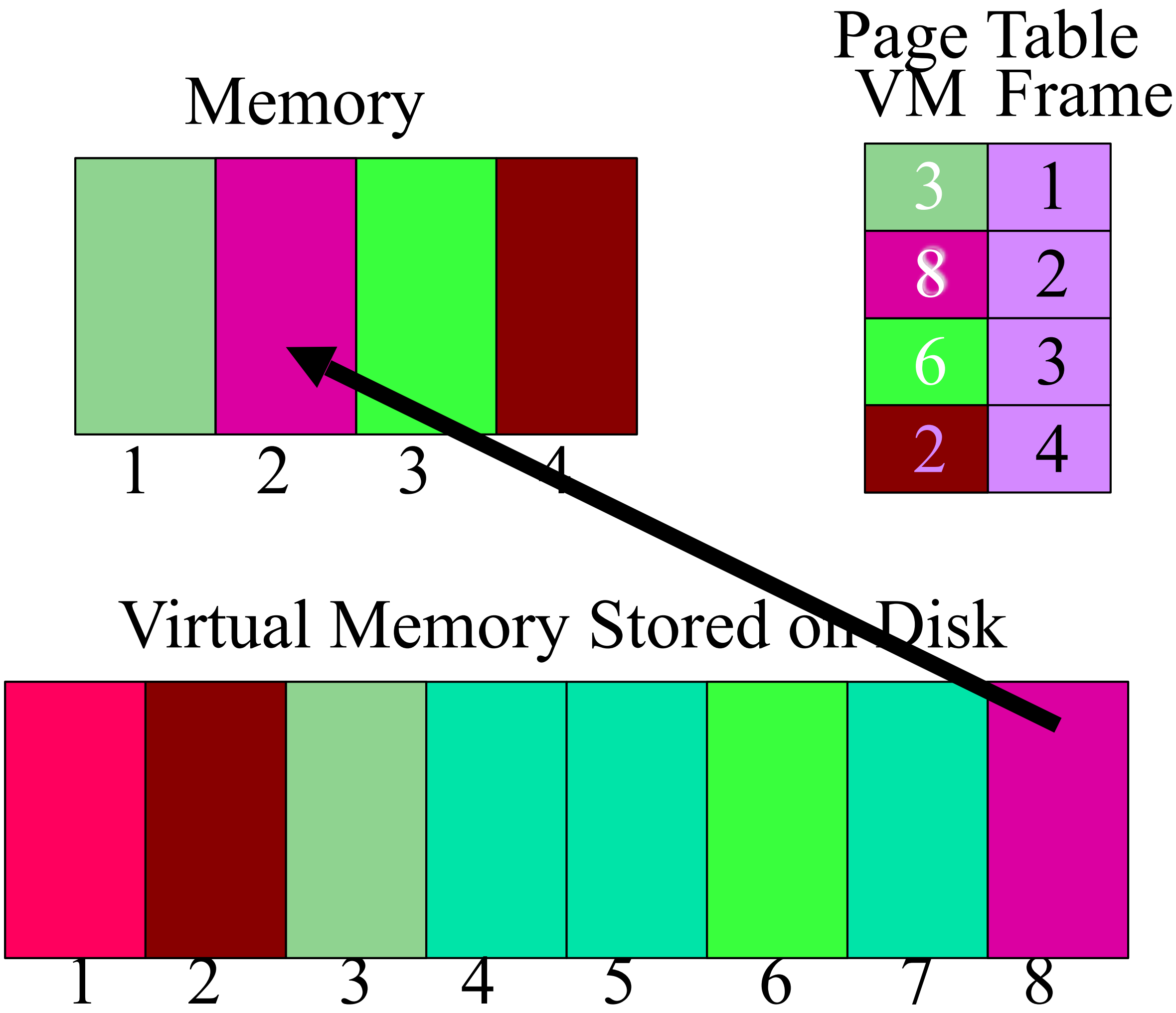
Paging

Request Page 8. Swap Page 1 to Disk First…

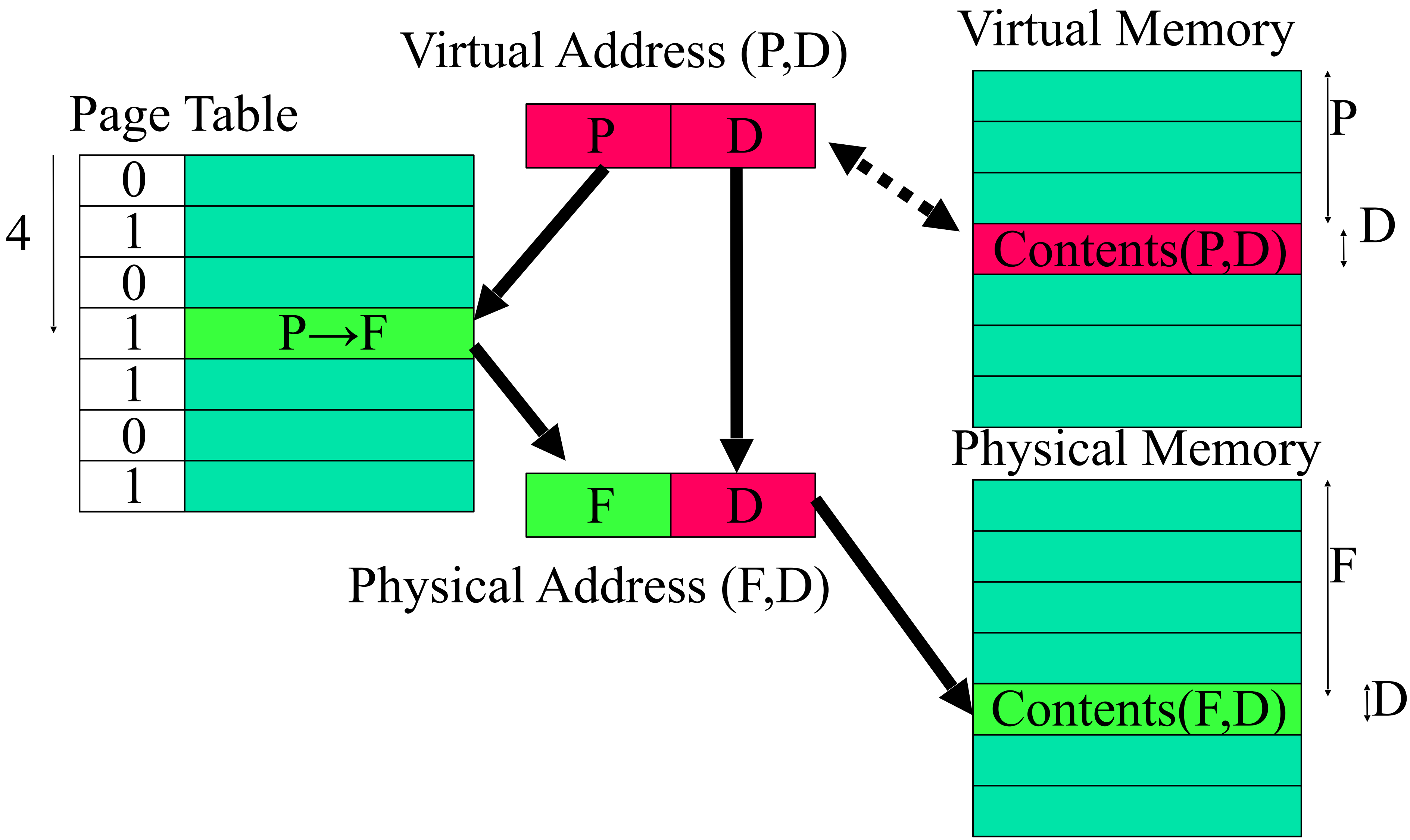


Paging

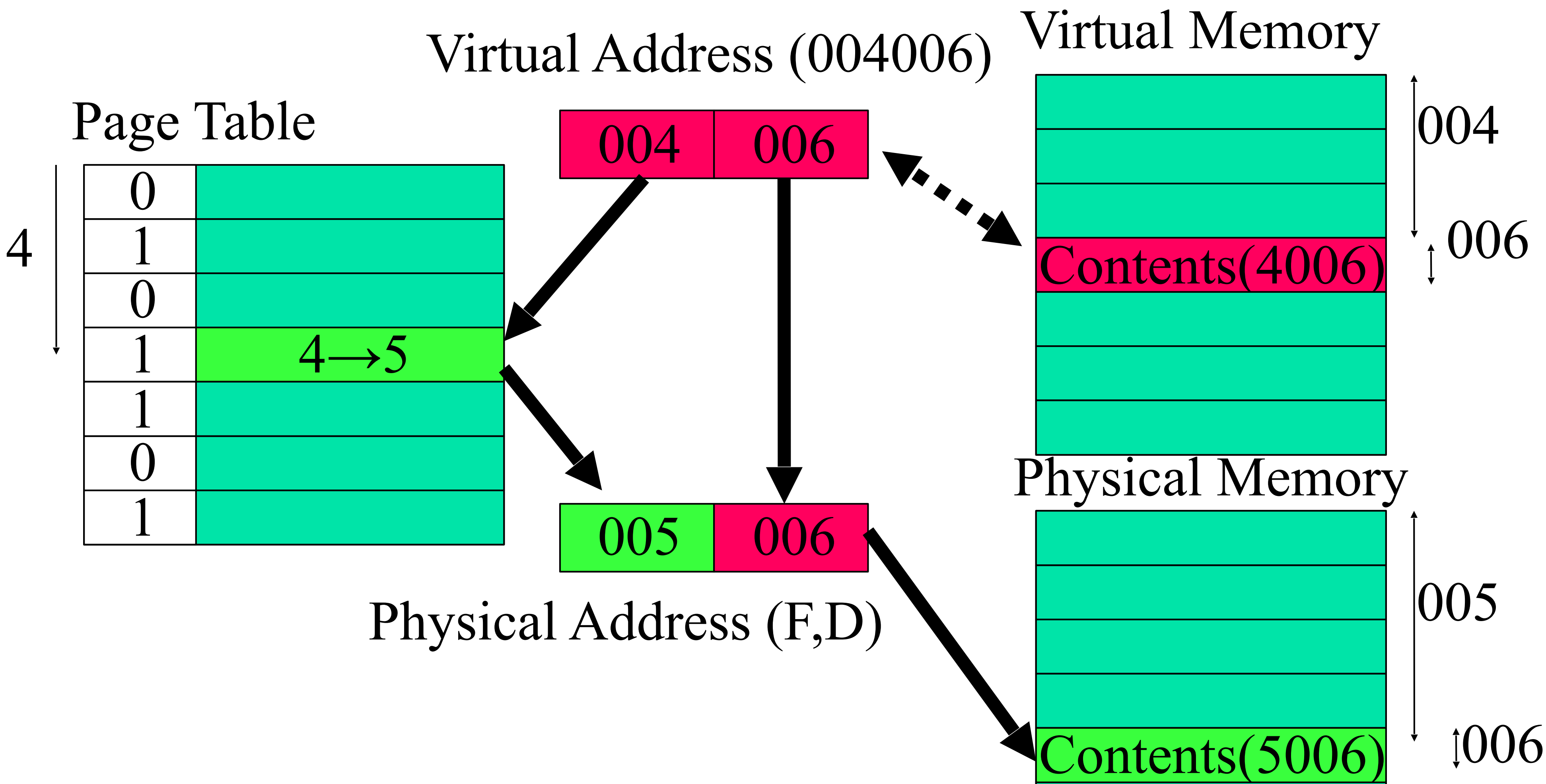
Request Page 8. … now load Page 8 into Memory.



Page Mapping Hardware



Page Mapping Hardware



Page size 1000
Number of Possible Virtual Pages 1000
Number of Page Frames 8

Page Faults

- Access a virtual page that is not mapped into any physical page
  - A fault is triggered by hardware
- Page fault handler (in OS's VM subsystem)
  - Find if there is any free physical page available
    - If no, evict some resident page to disk (swapping space)
  - Allocate a free physical page
  - Load the faulted virtual page to the prepared physical page
  - Modify the page table

Paging Issues

- Page size is 2^n
  - usually 512 bytes, 1 KB, 2 KB, 4 KB, or 8 KB
  - E.g. 32 bit VM address may have 2^{20} (1 MB) pages with 4k (2^{12}) bytes per page
- Page table:
  - 2^{20} page entries take 2^{22} bytes (4 MB)
  - Must map into real memory
  - Page Table base register must be changed for context switch
- No external fragmentation; internal fragmentation on last page only