Virtual Memory: Systems

CSCI 237: Computer Organization
25th Lecture, Apr 21, 2025

Jeannie Albrecht

Administrative Details

- Lab 4 – Cache simulator
  - You should be moving on to the “tricky” parts this week
  - Maintaining hit and miss rates
  - Keeping track of the LRU line for eviction
  - Make sure you free anything that you malloc!
- Next Glow HW will be posted soon and probably due Fri?
  - It’s about virtual memory…we’ll see how far we get

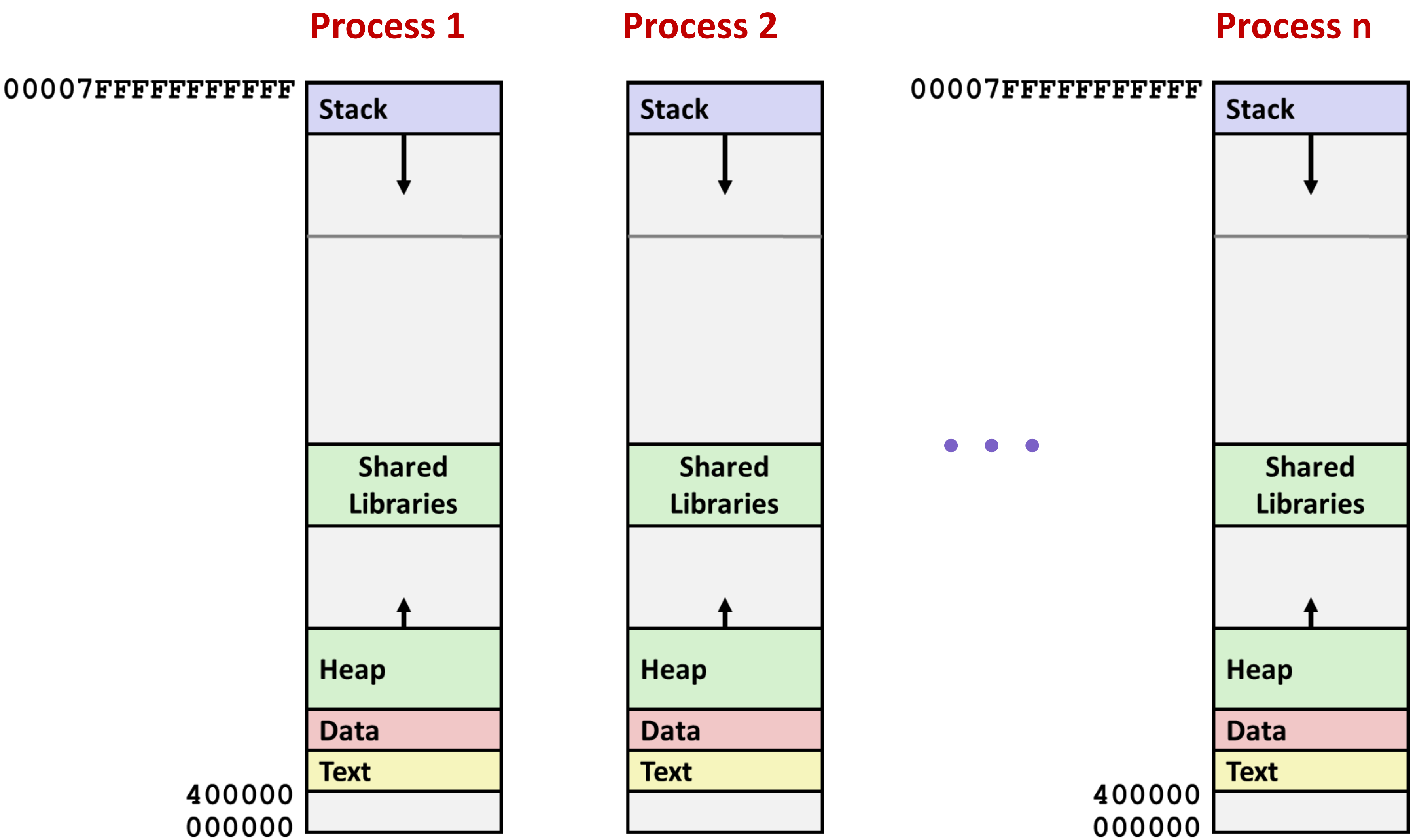
Last time

- Wrapped up caching
- Began (briefly) discussing address spaces

Today

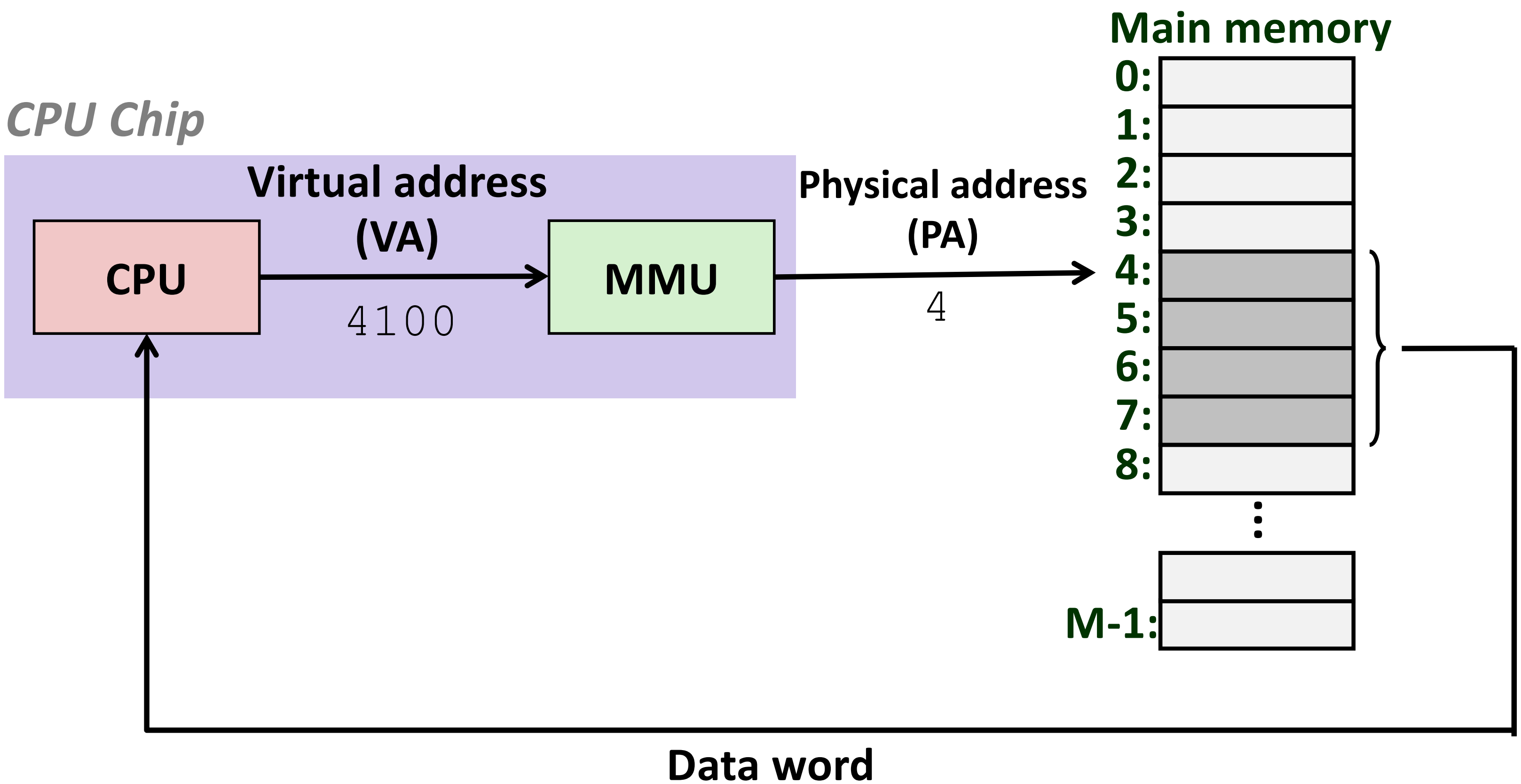
- Address spaces
- VM as a tool for caching
- VM as a tool for memory management
- VM as a tool for memory protection

Recap: How Does This Work?!



Solution: Virtual Memory

Recap: A System Using Virtual Addressing



- Used in all modern servers, laptops, PCs, and smart phones
- One of the great ideas in computer science->use indirection!
- MMU = memory management unit (part of CPU chip)

Address Spaces

- Linear address space: Ordered set of contiguous non-negative integer addresses (we always assume this):

  {0, 1, 2, 3 … }

- Virtual address space: Set of $N = 2^n$ virtual addresses

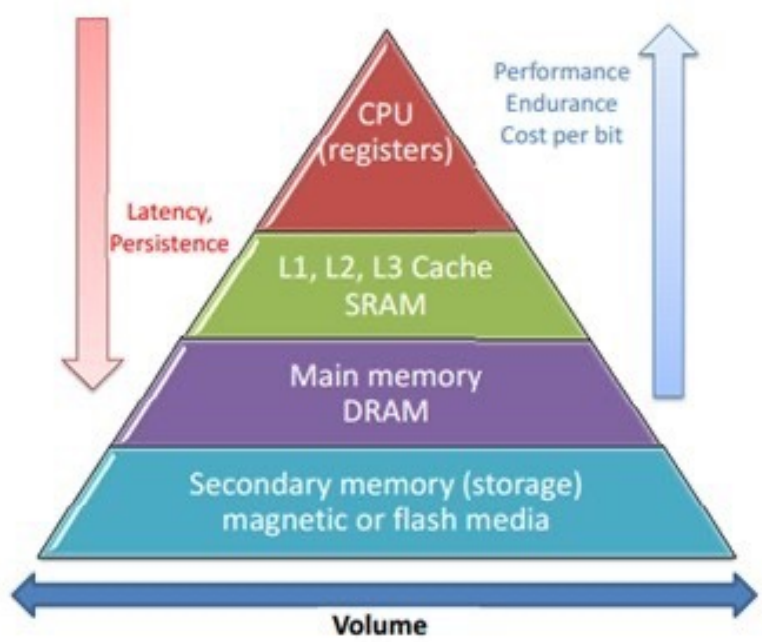
  {0, 1, 2, 3, …, N-1}

- Physical address space: Set of $M = 2^m$ physical addresses

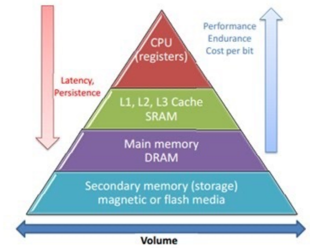
  {0, 1, 2, 3, …, M-1}

Today

- Address spaces
- VM as a tool for caching
- VM as a tool for memory management
- VM as a tool for memory protection

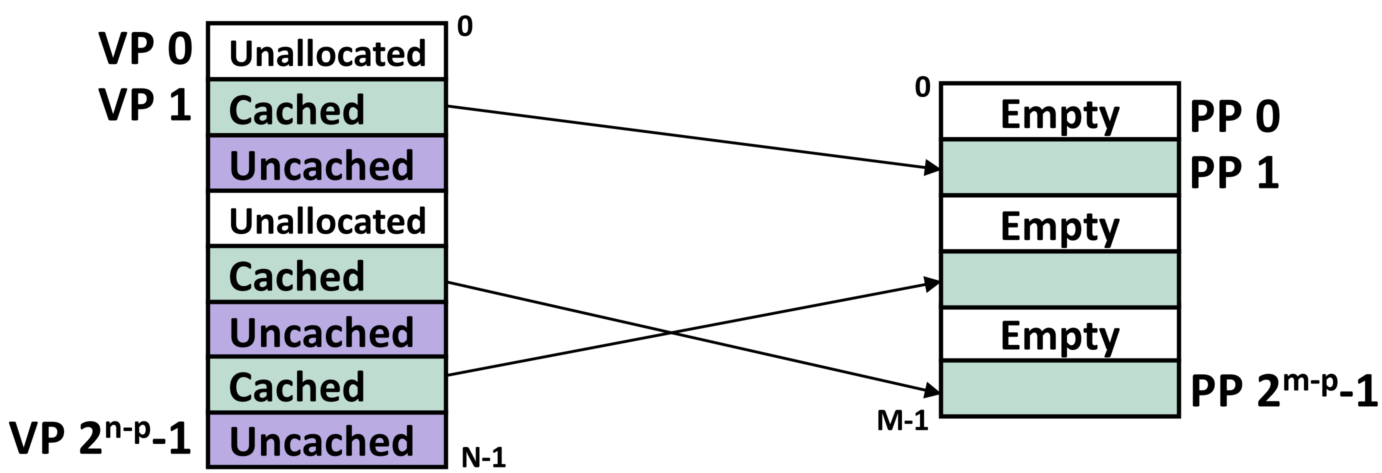


VM as a Tool for Caching



- Conceptually, *virtual memory* is an array of N contiguous bytes stored **on disk**. (we've moved down in the hierarchy!)
- The contents of the array on disk are cached in *physical memory* (*DRAM cache, one level up in hierarchy*)
  - These cache blocks are called memory *pages* (size is $P = 2^p$ bytes)

Virtual memory **Physical memory**



Virtual pages (VPs) stored on disk

Physical pages (PPs) cached in DRAM

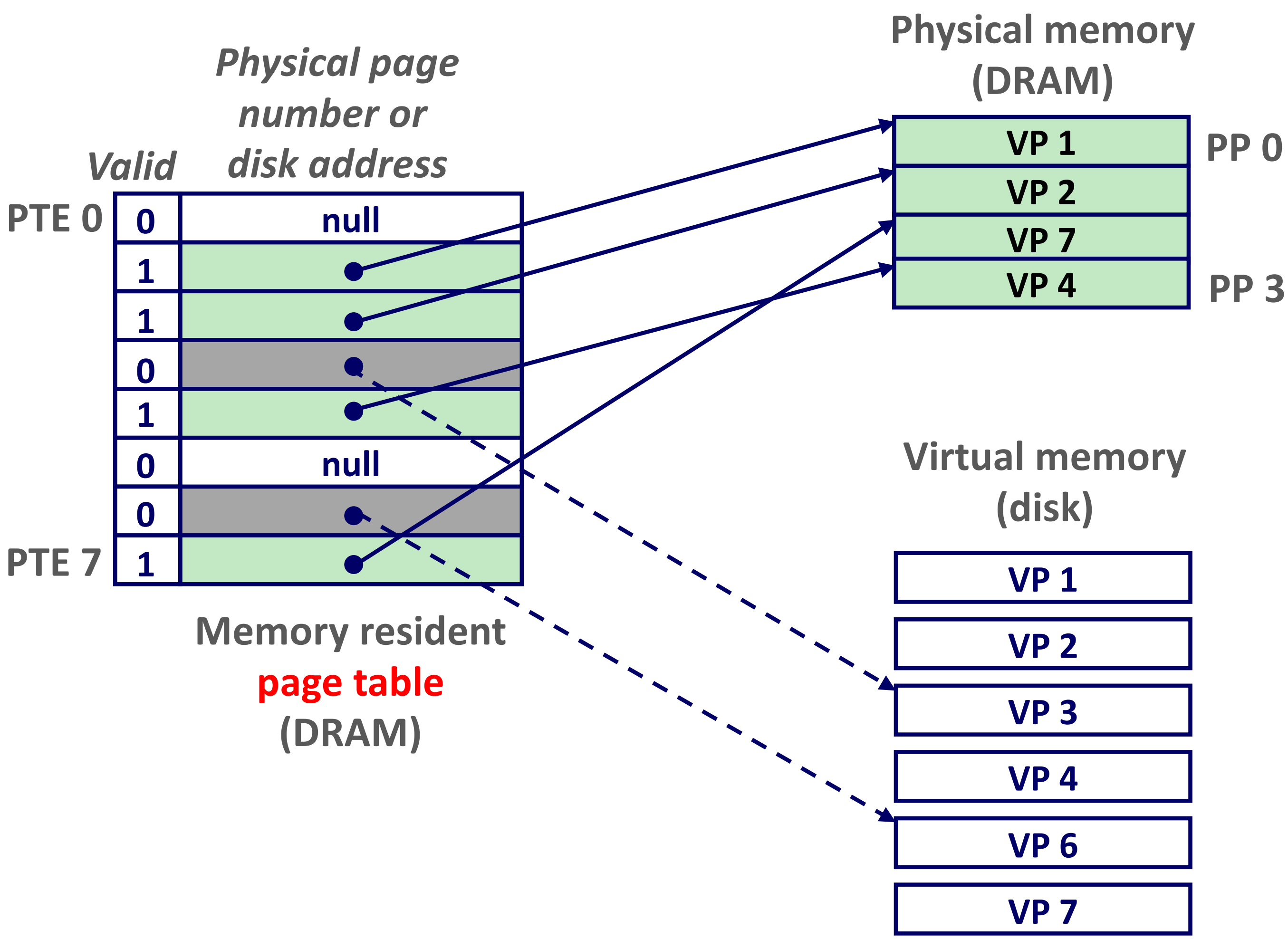
DRAM Cache Organization

- DRAM cache organization driven by the enormous miss penalty
  - DRAM is about ***10x*** slower than SRAM
  - Disk is about ***10,000x*** slower than DRAM

- Consequences
  - Large page (block) size (relatively): typically 4 KB, sometimes 4 MB
  - **Fully associative**
    - **Any VP (virtual page) can be placed in any PP (physical page)!**
    - Requires a “large” mapping function – very different from cache memories where we just use the bits for indexing
  - Highly sophisticated, expensive replacement algorithms
    - Too complicated and open-ended to be implemented in hardware
  - Write-back rather than write-through

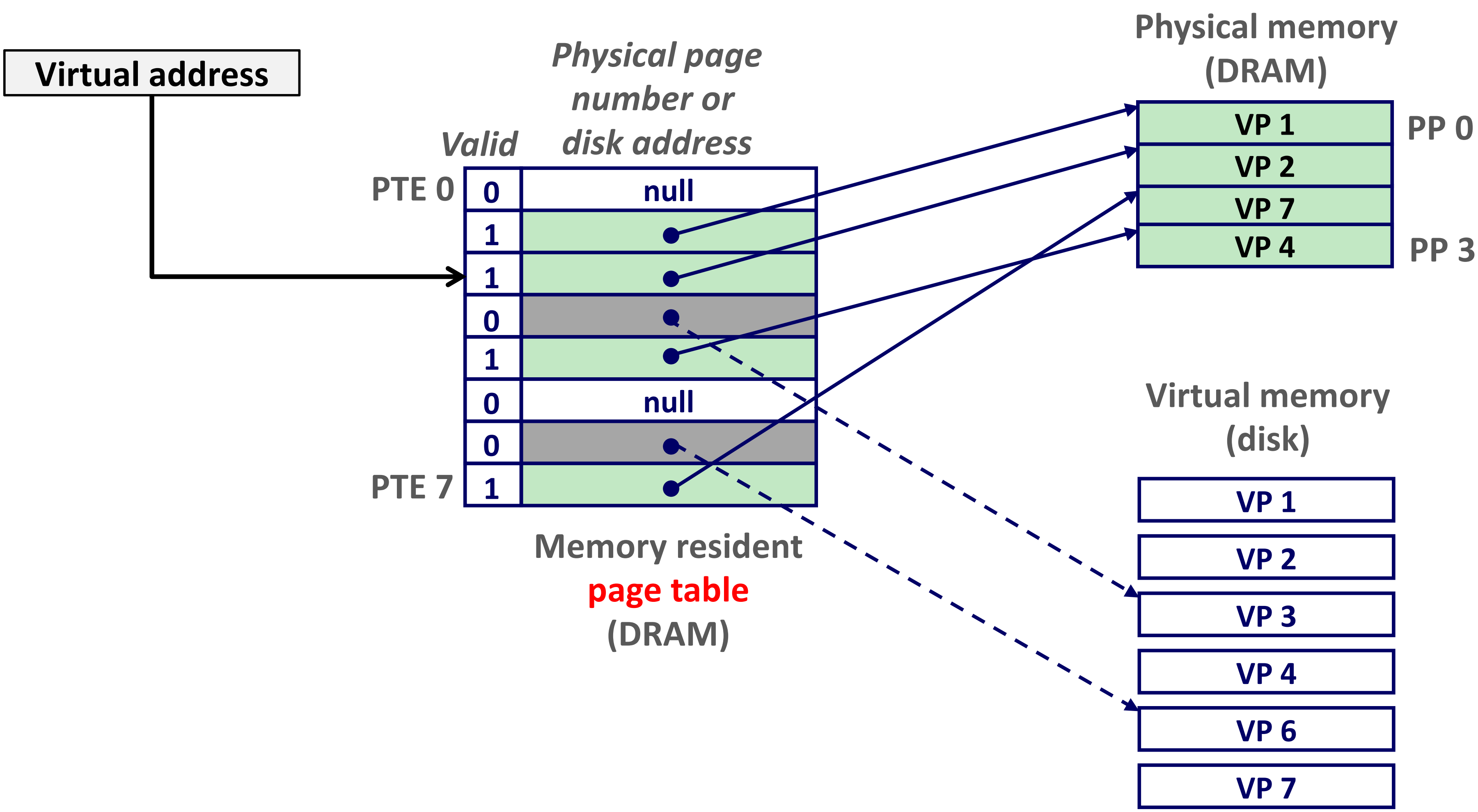
Enabling Data Structure: Page Table

- A *page table* is an array of page table entries (PTEs) that maps virtual pages to physical pages.
  - Per-process kernel (i.e., OS) data structure in DRAM



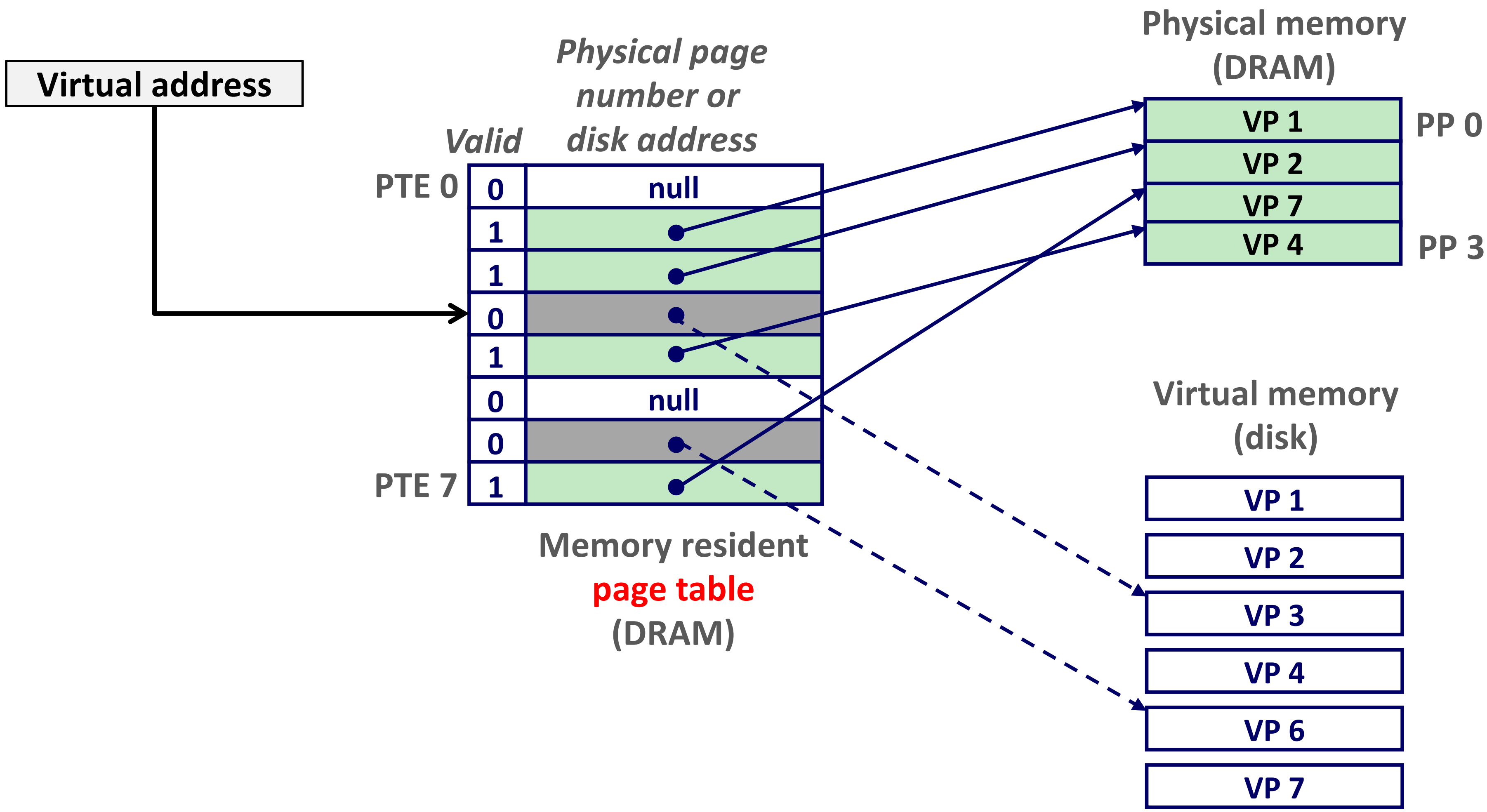
Page Hit

- *Page hit:* reference to VM word that is in physical memory (DRAM cache hit)



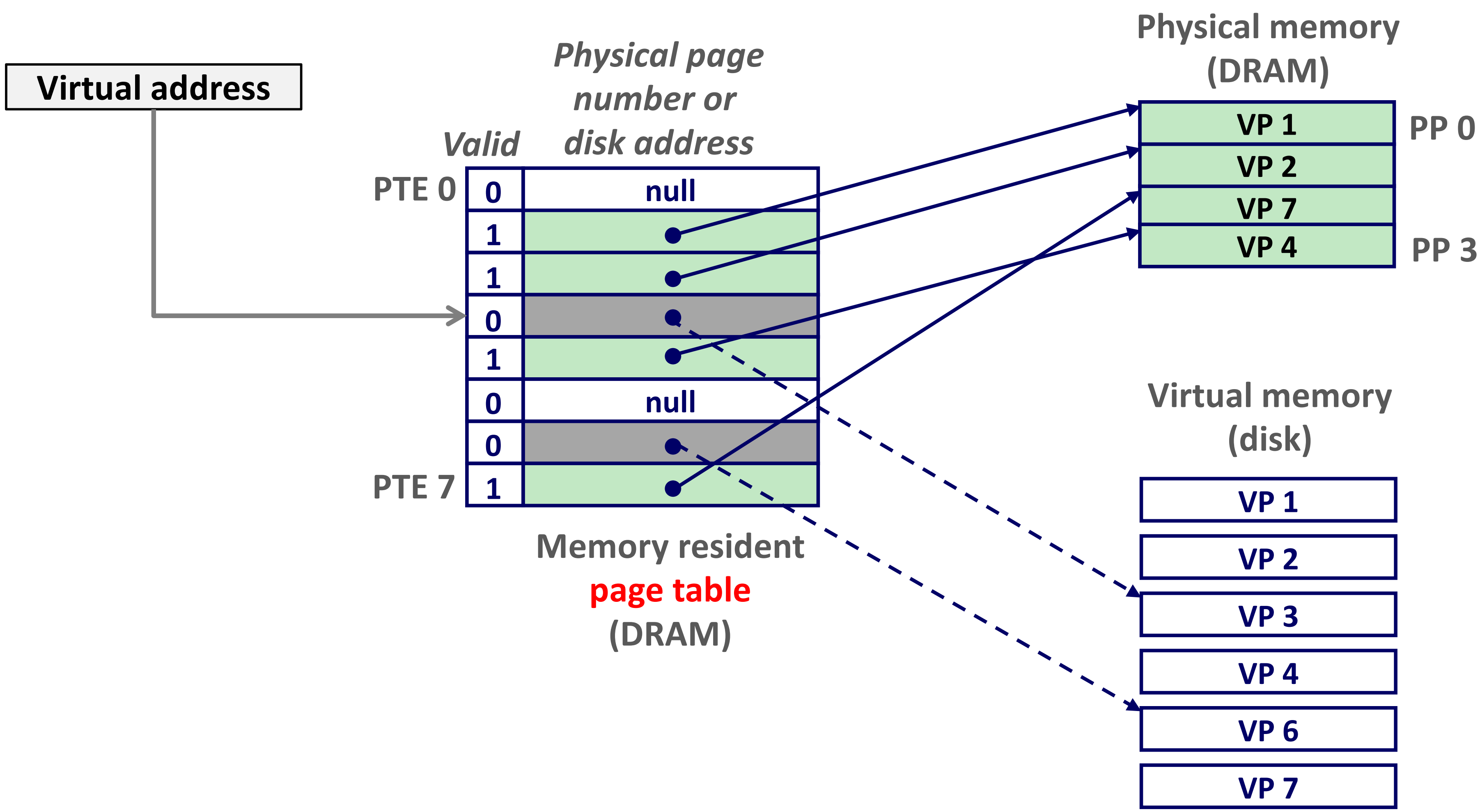
Page Fault

- *Page fault:* reference to VM word that is not in physical memory (DRAM cache miss)



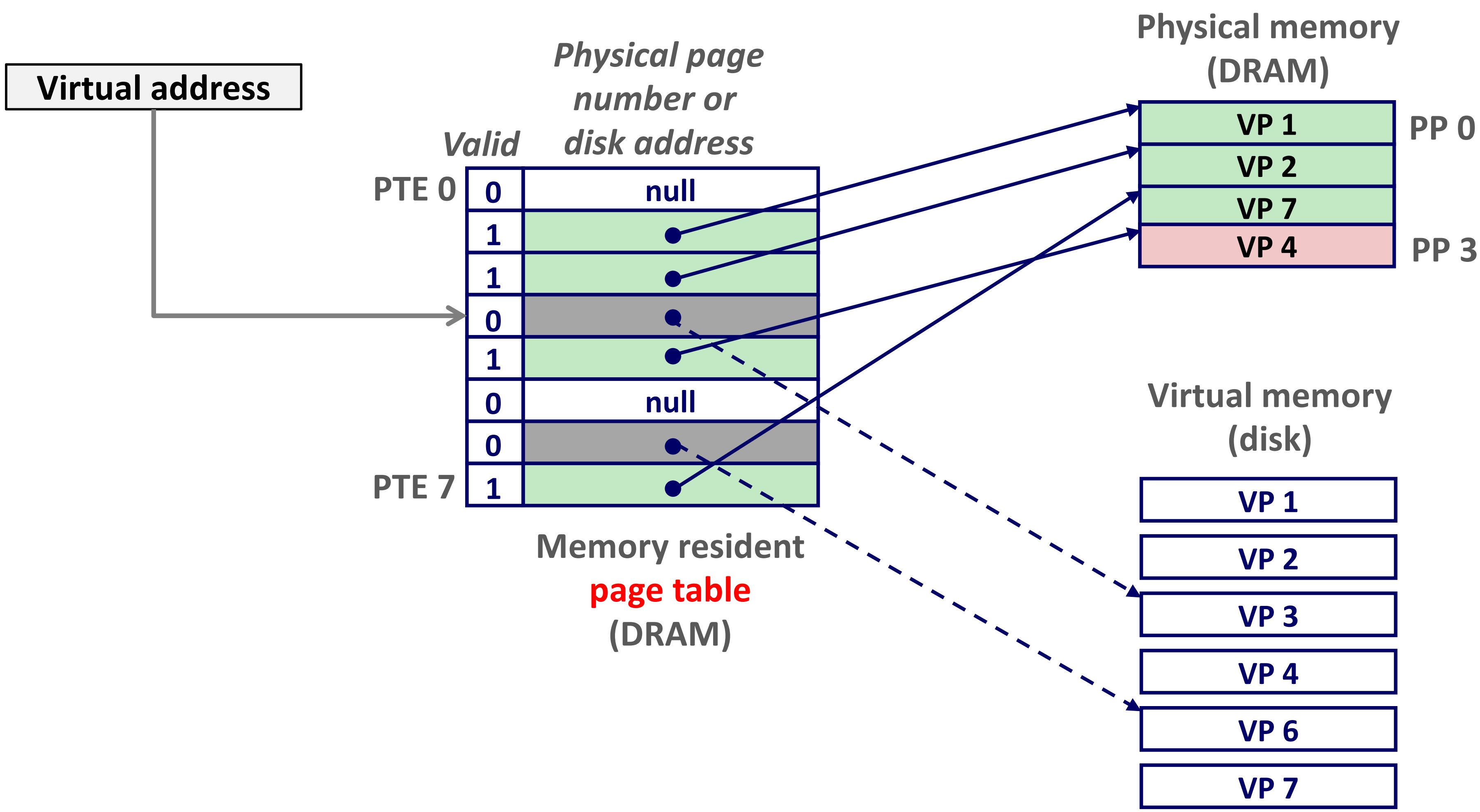
Handling Page Fault

- DRAM cache miss causes page fault (an exception)



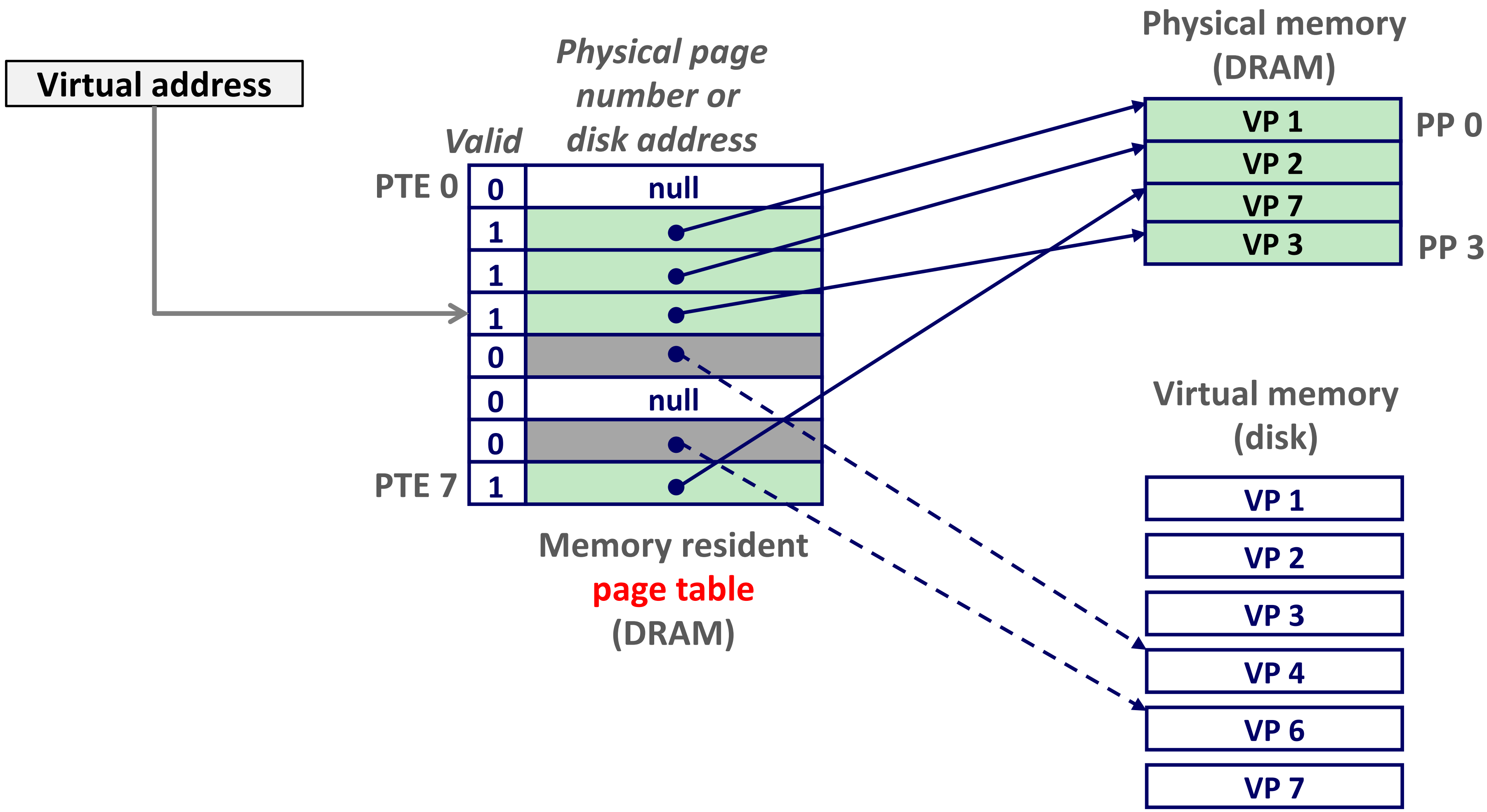
Handling Page Fault

- DRAM cache miss causes page fault (an exception)
- Page fault handler selects a victim to be evicted (here VP 4)



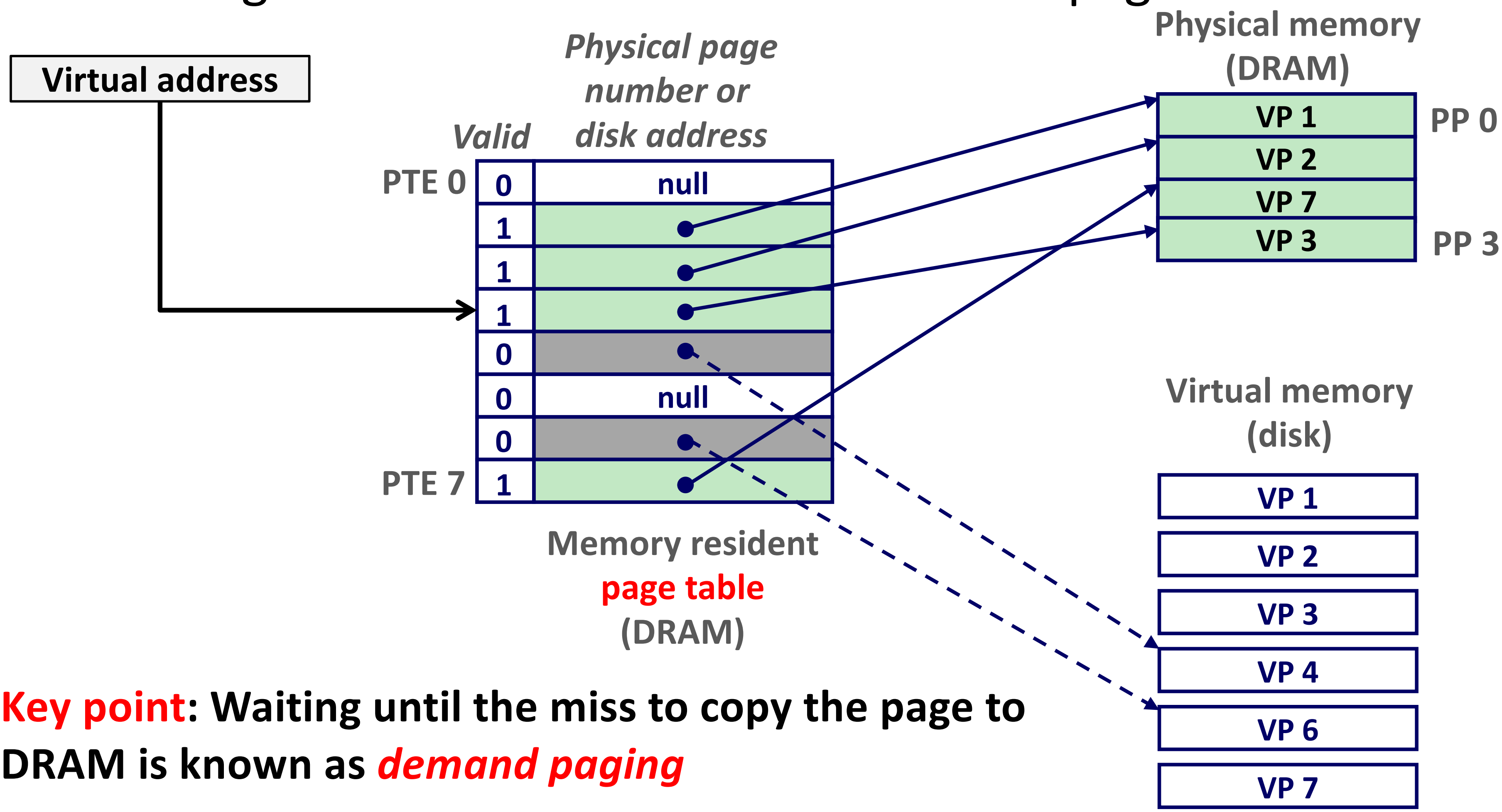
Handling Page Fault

- DRAM cache miss causes page fault (an exception)
- Page fault handler selects a victim to be evicted (here VP 4)



Handling Page Fault

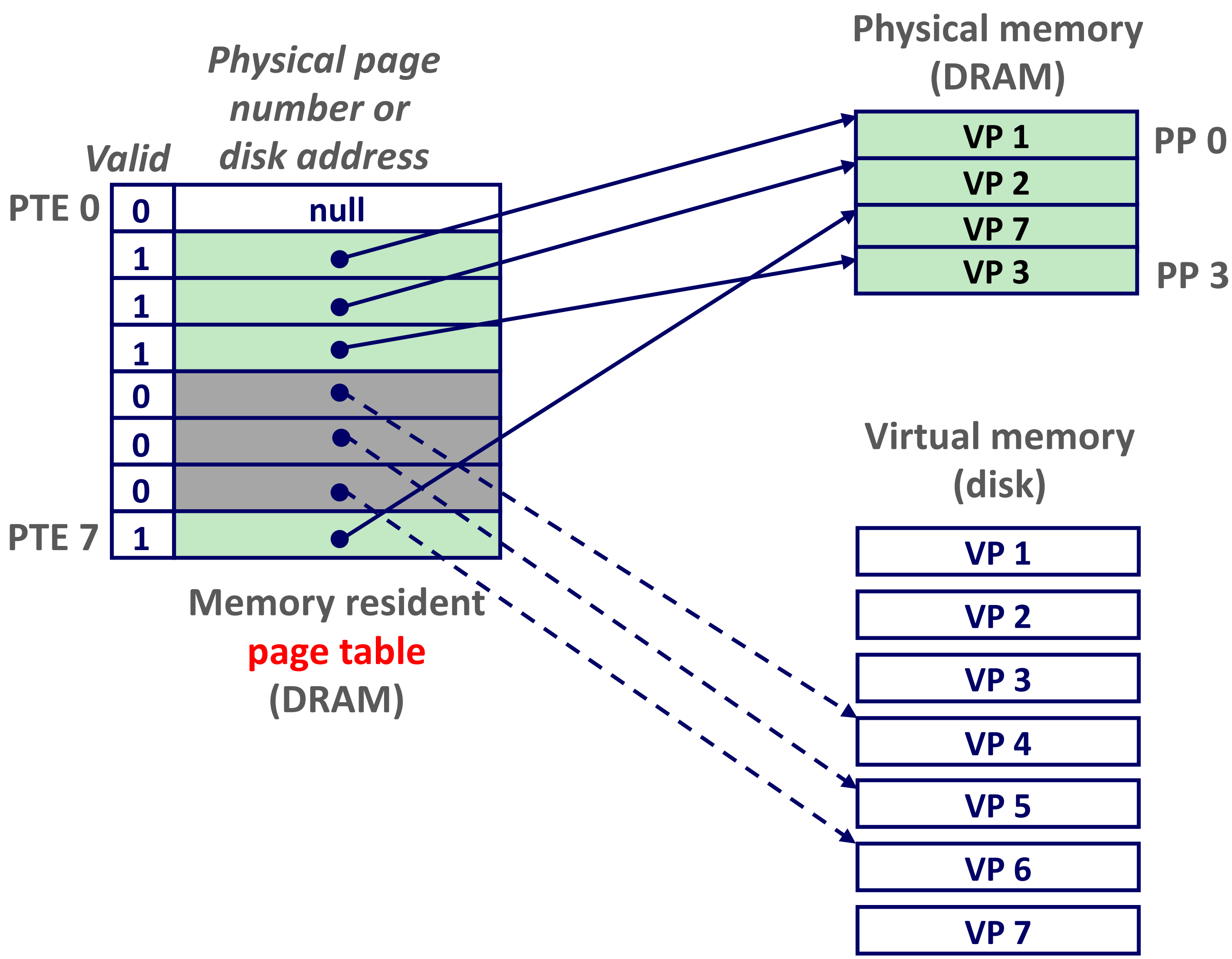
- DRAM cache miss causes page fault (an exception)
- Page fault handler selects a victim to be evicted (here VP 4)
- Offending instruction is restarted: now it's a page hit!



Key point: Waiting until the miss to copy the page to DRAM is known as *demand paging*

Allocating Pages

- Allocating a new page (VP 5) of virtual memory. Allocate on disk.



Locality to the Rescue Again!

- Virtual memory seems terribly inefficient, but it works well because of locality.

- At any point in time, programs tend to access a set of active virtual pages called the *working set*
  - Programs with better temporal locality will have smaller working sets

- If (working set size < main memory size)
  - Good performance for one process after compulsory misses

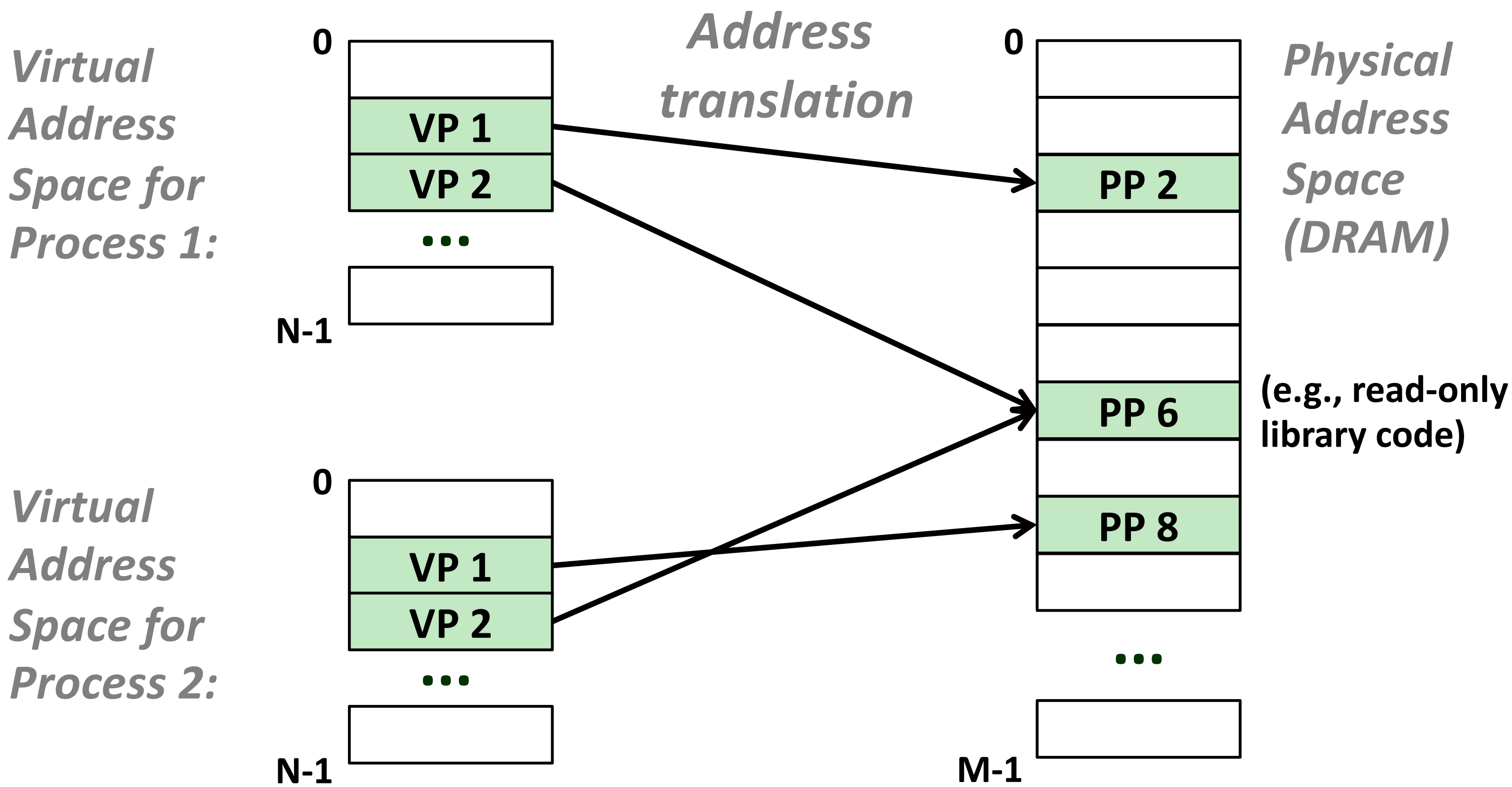
- If (SUM(working set sizes) > main memory size)
  - *Thrashing:* Performance meltdown where pages are swapped (copied) in and out continuously

Today

- Address spaces
- VM as a tool for caching
- VM as a tool for memory management
- VM as a tool for memory protection

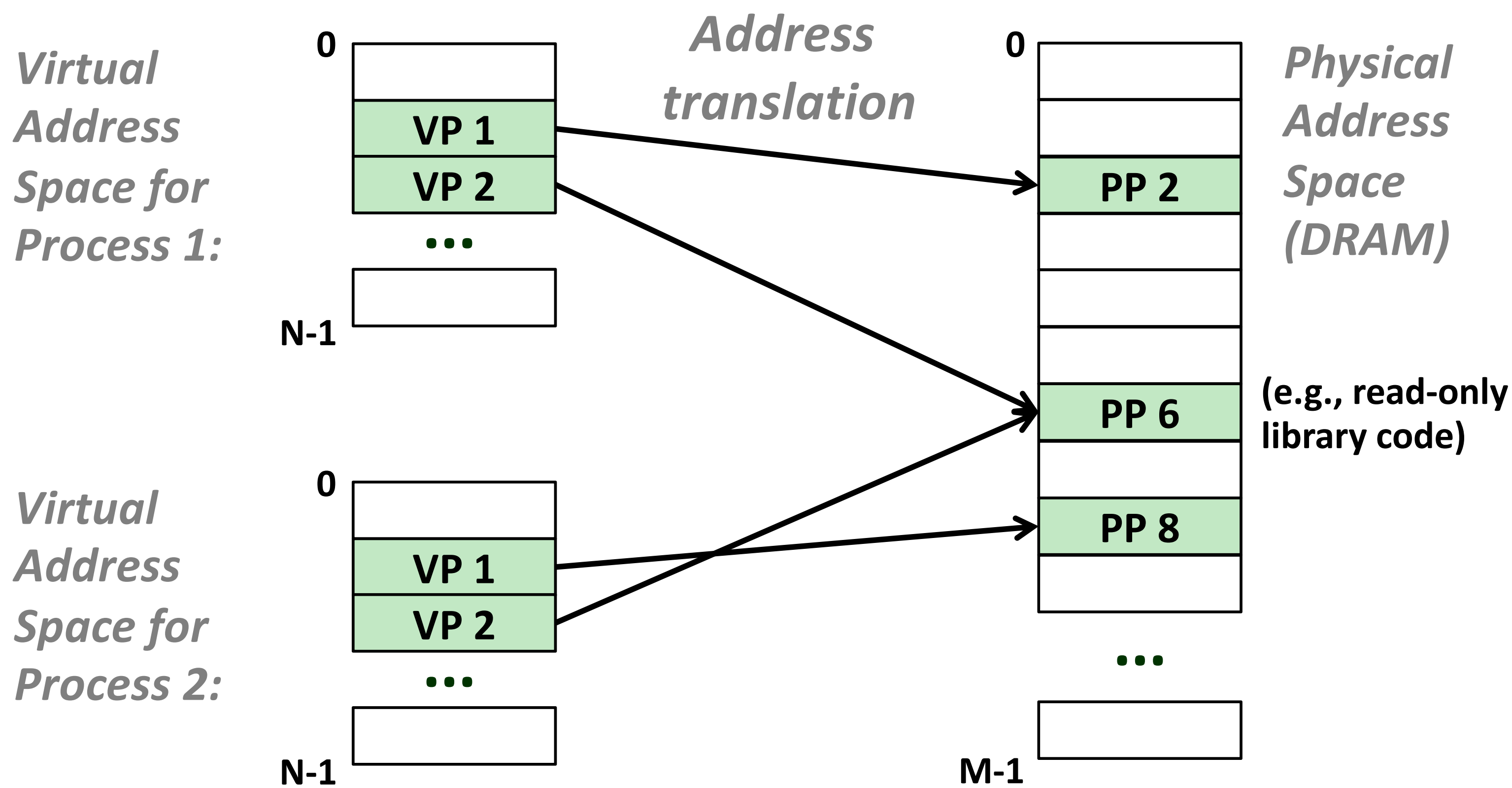
VM as a Tool for Memory Management

- Key idea: each process has its own virtual address space
  - It can view memory as a simple linear array
  - Mapping function scatters addresses through physical memory
    - Well-chosen mappings can improve locality



VM as a Tool for Memory Management

- Simplifying memory allocation
  - Each virtual page can be mapped to any physical page
  - A virtual page can be stored in different physical pages at different times
- Sharing code and data among processes
  - Map virtual pages to the same physical page (here: PP 6)



Today

- Address spaces
- VM as a tool for caching
- VM as a tool for memory management
- VM as a tool for memory protection

VM as a Tool for Memory Protection

- Extend PTEs (page table entries) with permission bits
- MMU (mem management unit, responsible for translation) checks these bits on each access

Process i:

	SUP	READ	WRITE	EXEC	Address
VP 0:	No	Yes	No	Yes	PP 6
VP 1:	No	Yes	Yes	Yes	PP 4
VP 2:	Yes	Yes	Yes	No	PP 2

⋮

Process j:

	SUP	READ	WRITE	EXEC	Address
VP 0:	No	Yes	No	Yes	PP 9
VP 1:	Yes	Yes	Yes	Yes	PP 6
VP 2:	No	Yes	Yes	Yes	PP 11