Introduction to Computer Systems
Lecture 12 – Virtual Memory: Concepts

2022 Spring, CSE3030

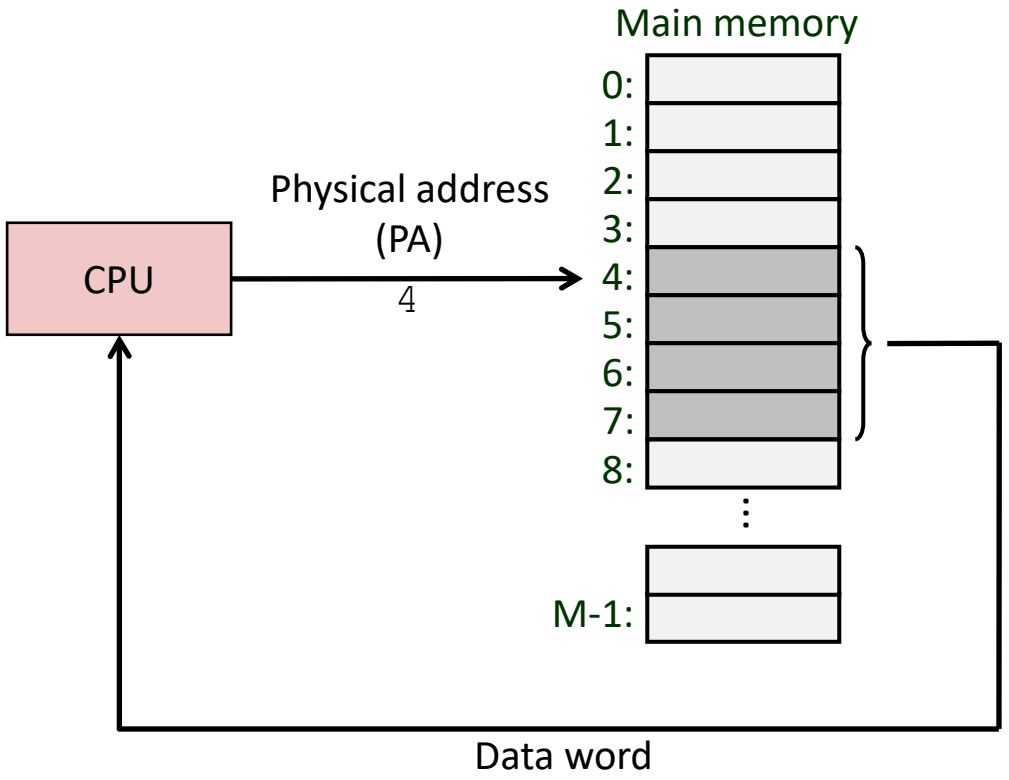
Sogang University

Today

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

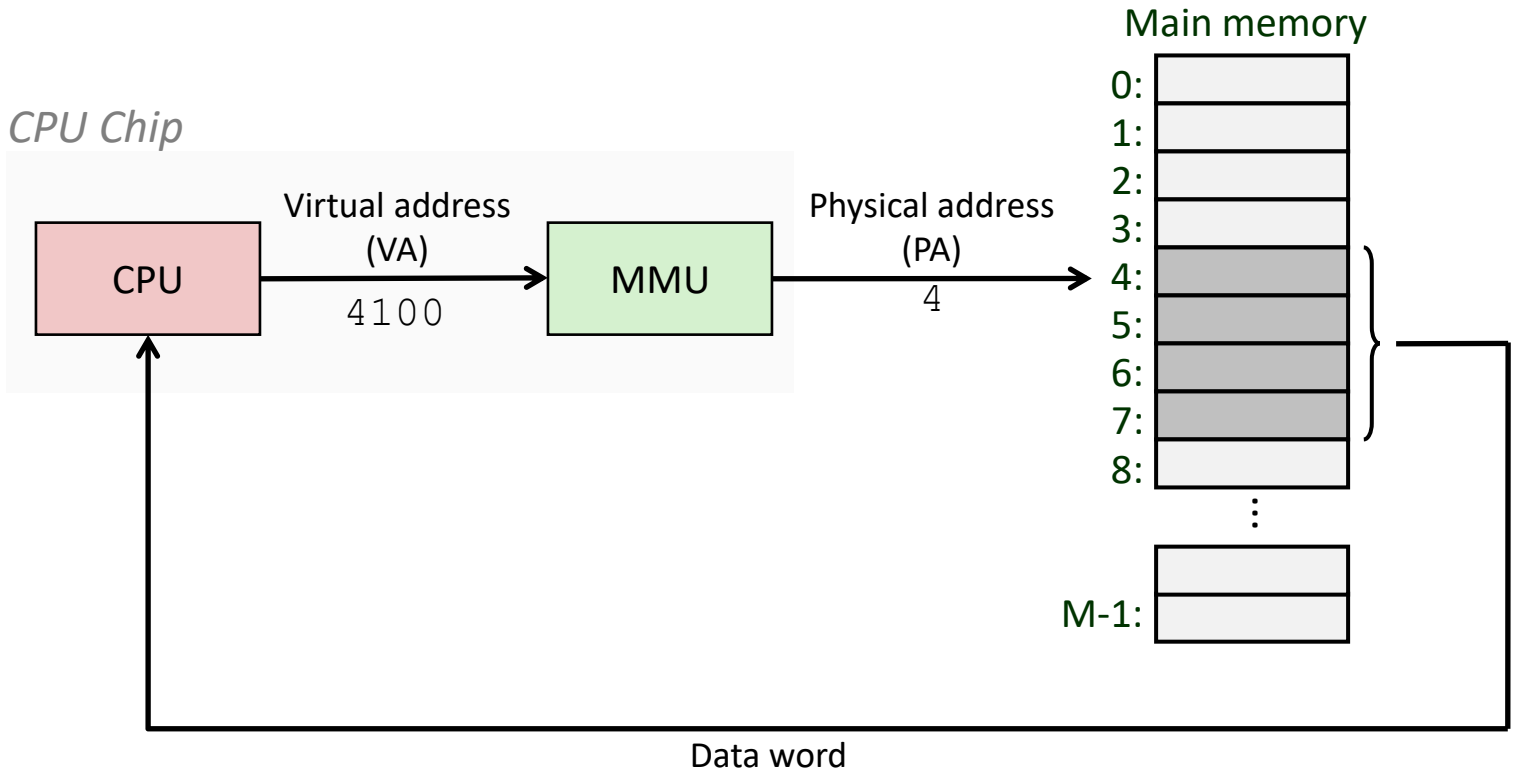
A System Using Physical Addressing

- Used in “simple” systems like embedded microcontrollers in devices li ke cars, elevators, and digital picture frames



A System Using Virtual Addressing

- Used in all modern servers, laptops, and smart phones
- One of the great ideas in computer science



Address Spaces

- Linear address space: Ordered set of contiguous non-negative integer addresses:
  {0, 1, 2, 3 … }

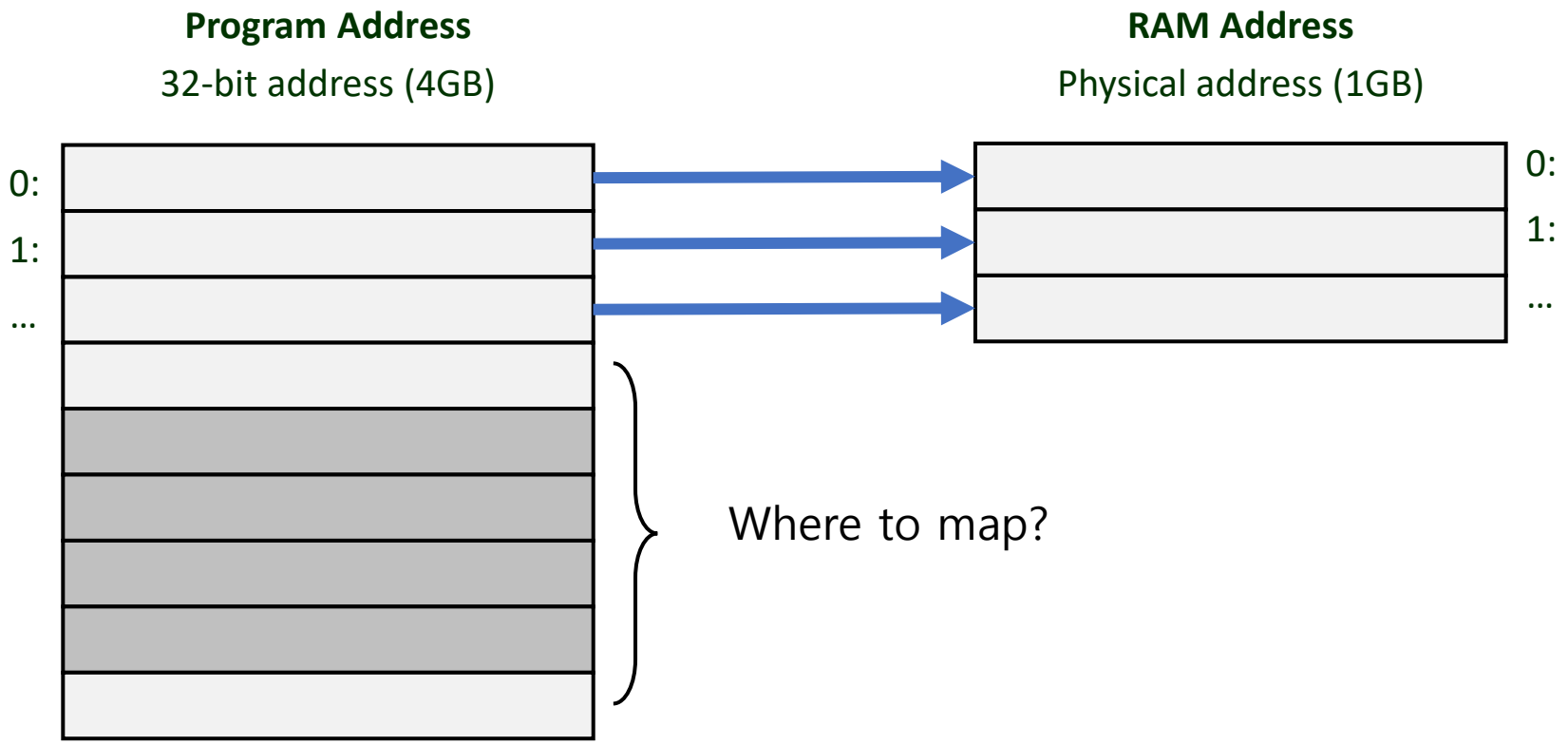
- Virtual address space: Set of $N = 2^n$ virtual addresses
  {0, 1, 2, 3, …, N-1}

- Physical address space: Set of $M = 2^m$ physical addresses
  {0, 1, 2, 3, …, M-1}

- Basic Parameters
  - **$N = 2^n$** : Number of addresses in virtual address space
  - **$M = 2^m$** : Number of addresses in physical address space
  - **$P = 2^p$** : Page size (bytes)
  - **2^{n-p}**: The number of virtual pages
  - **2^{m-p}**: The number of physical pages

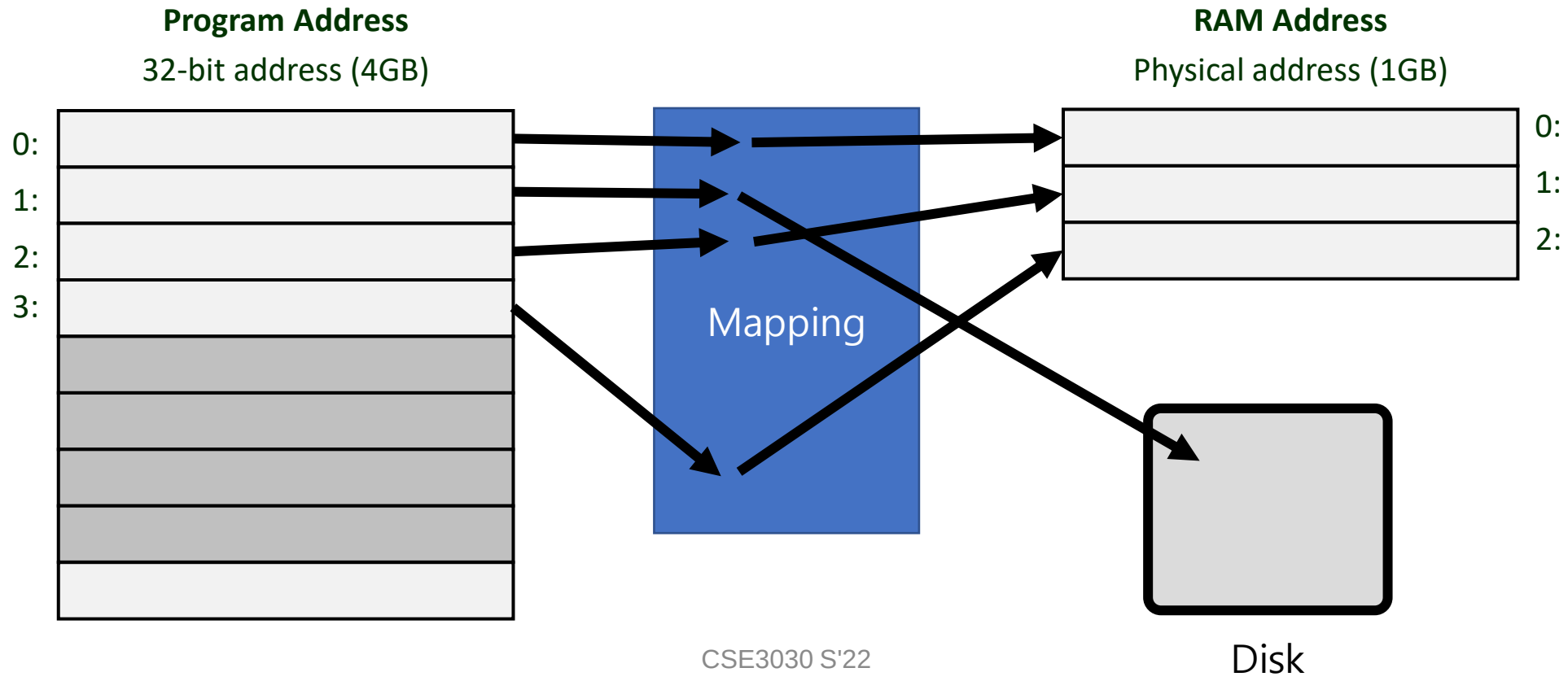
Why virtual memory?

- Direct mapping cannot use a whole of 32-bit memory space.



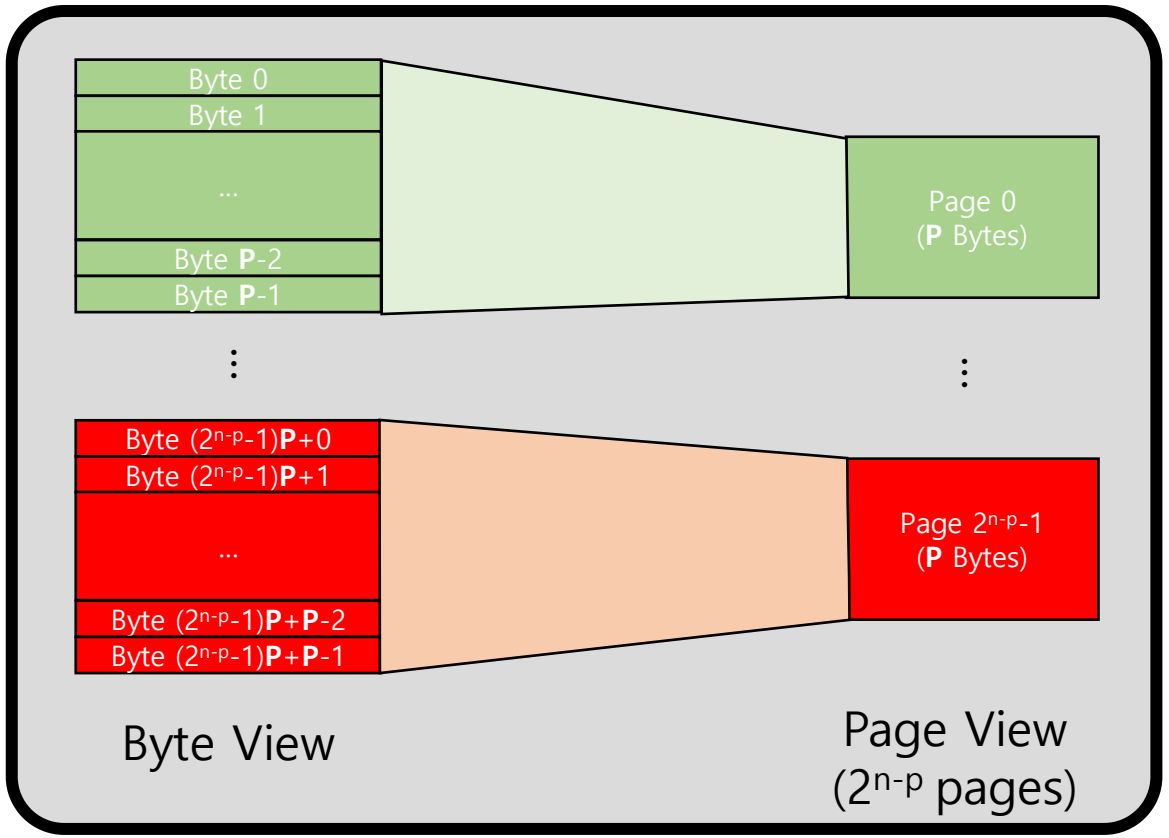
Why virtual memory?

- A mapping table maps program address to RAM address.
- Some virtual space maps to disk space.
- If we use it, bring the mapped space into RAM.



Overview of Virtual Memory: Pages

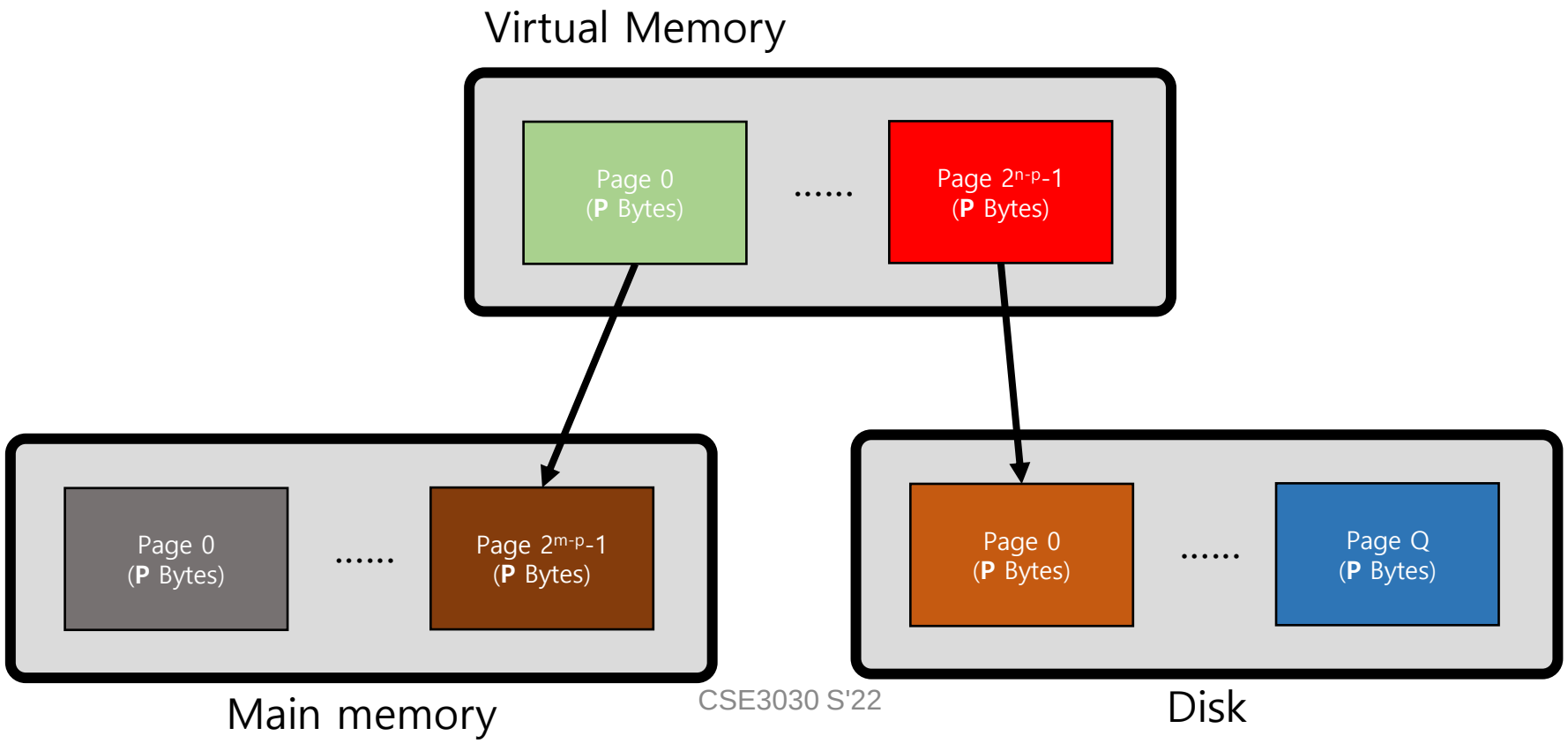
- Pages: a unit of memory space
  - Page size **P:** typically 4 KB, sometimes 4 MB



Virtual Memory

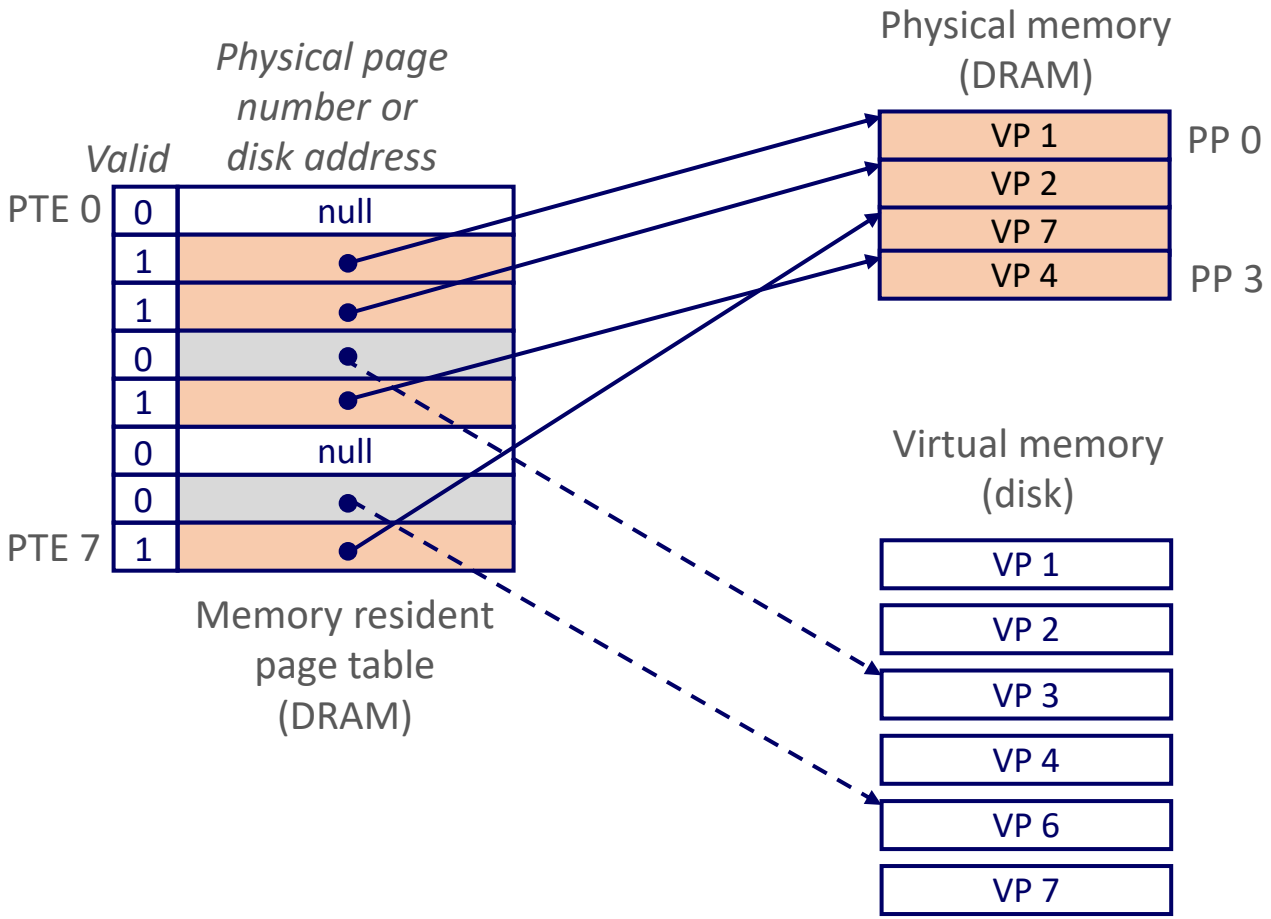
Overview of Virtual Memory: Mapping

- We assign a physical page to a virtual page when the CPU accesses the virtual page at first.
- Mapped physical pages can be in either main memory or a disk.



Enabling Data Structure: Page Table

- A *page table* is an array of page table entries (PTEs) that maps **virtua l pages** to **physical pages.**



Why Virtual Memory (VM)?

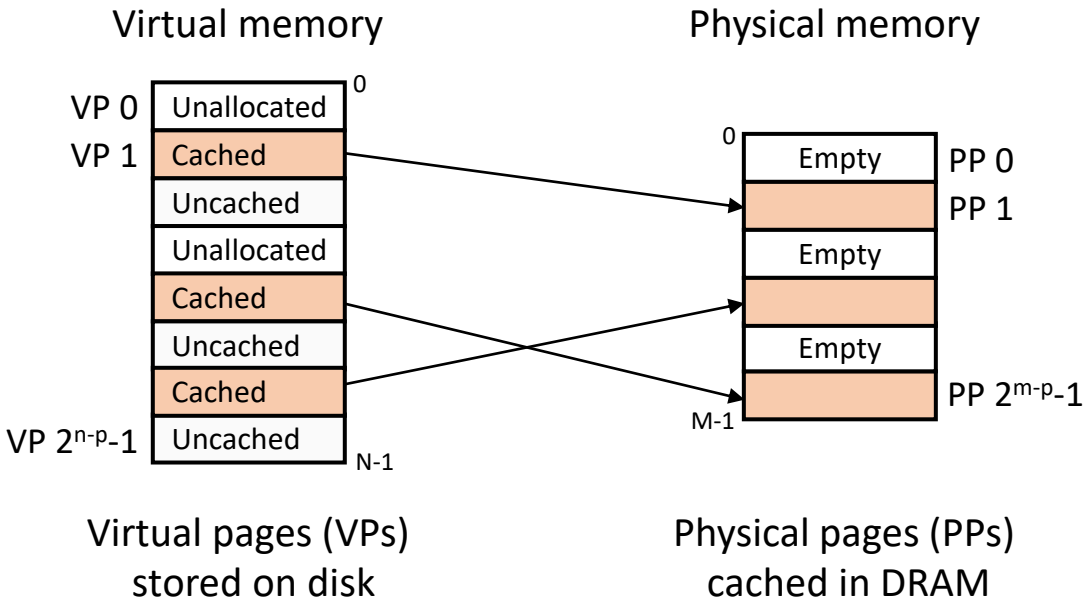
- Uses main memory efficiently
  - Use DRAM as a cache for parts of a virtual address space
- Simplifies memory management
  - Each process gets the same uniform linear address space
- Isolates address spaces
  - One process can’t interfere with another’s memory
  - User program cannot access privileged kernel information and code

Today

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

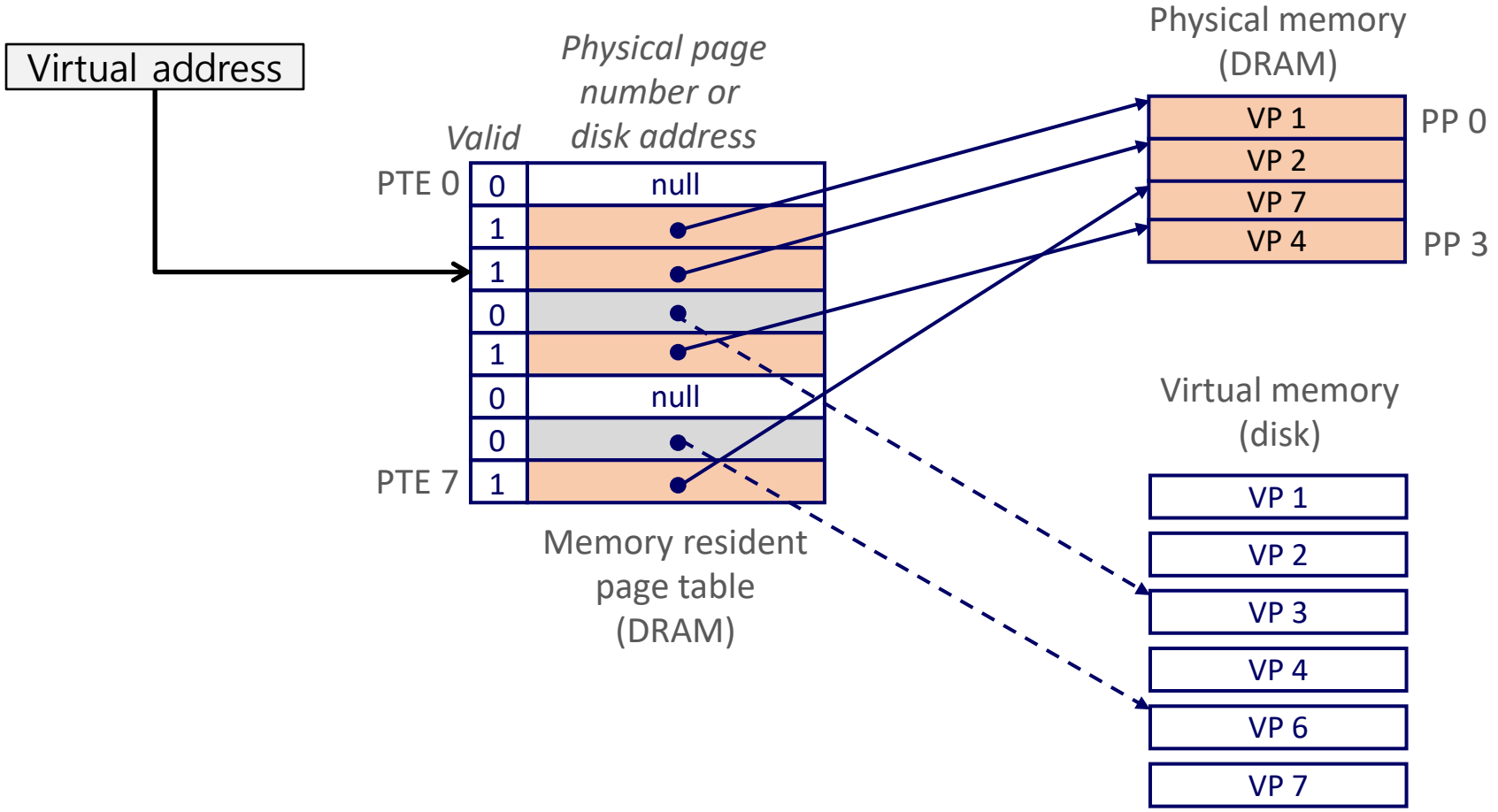
VM as a Tool for Caching

- The contents of the array on virtual disk are cached in *physical memory* (*DRAM memory*)
  - If a page is accessed at first, assign a new physical page in main memory.
  - If pages are recently used, they are in DRAM memory.
  - If pages are evicted, they are placed in the disk.



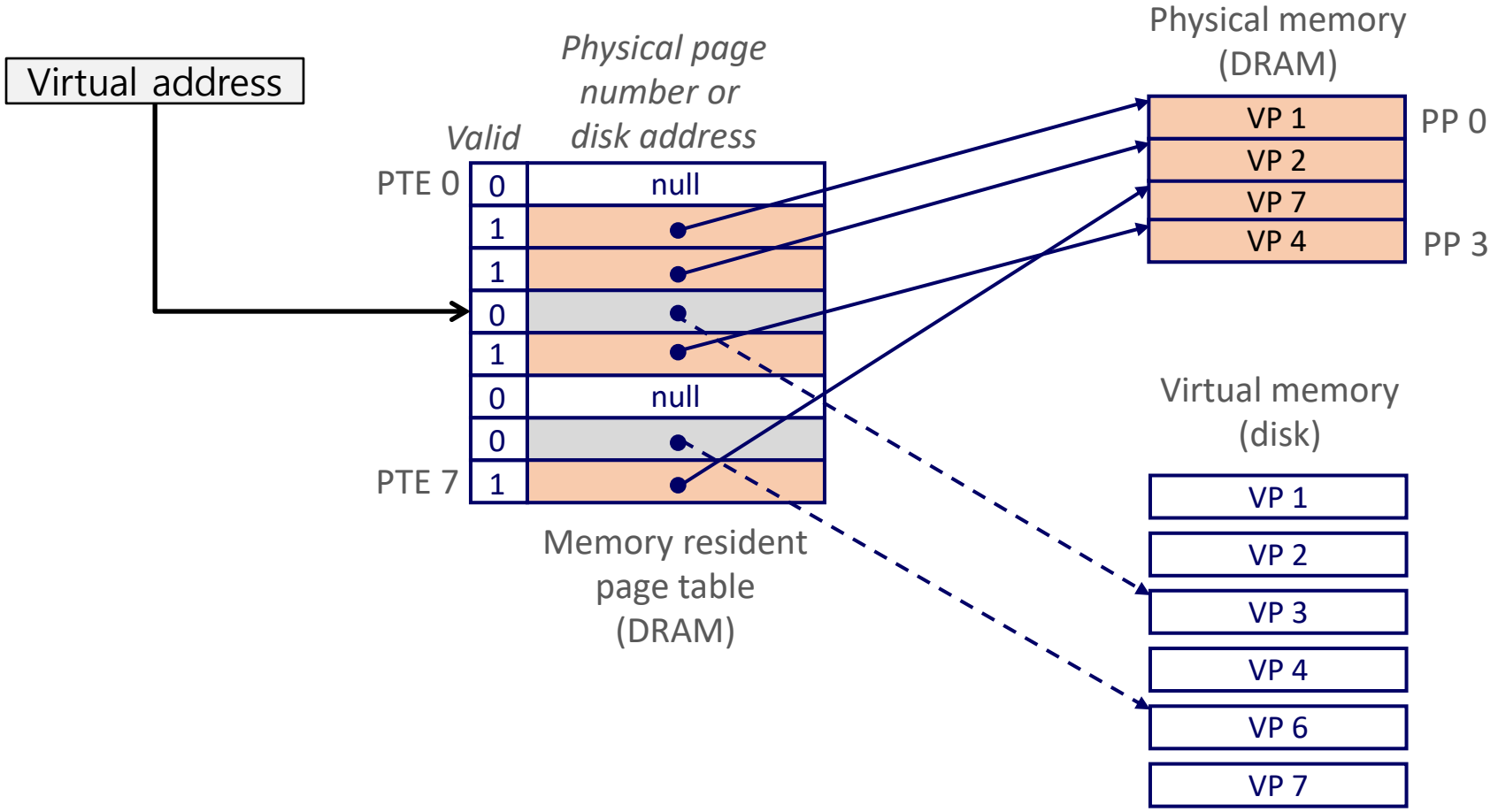
Page Hit

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



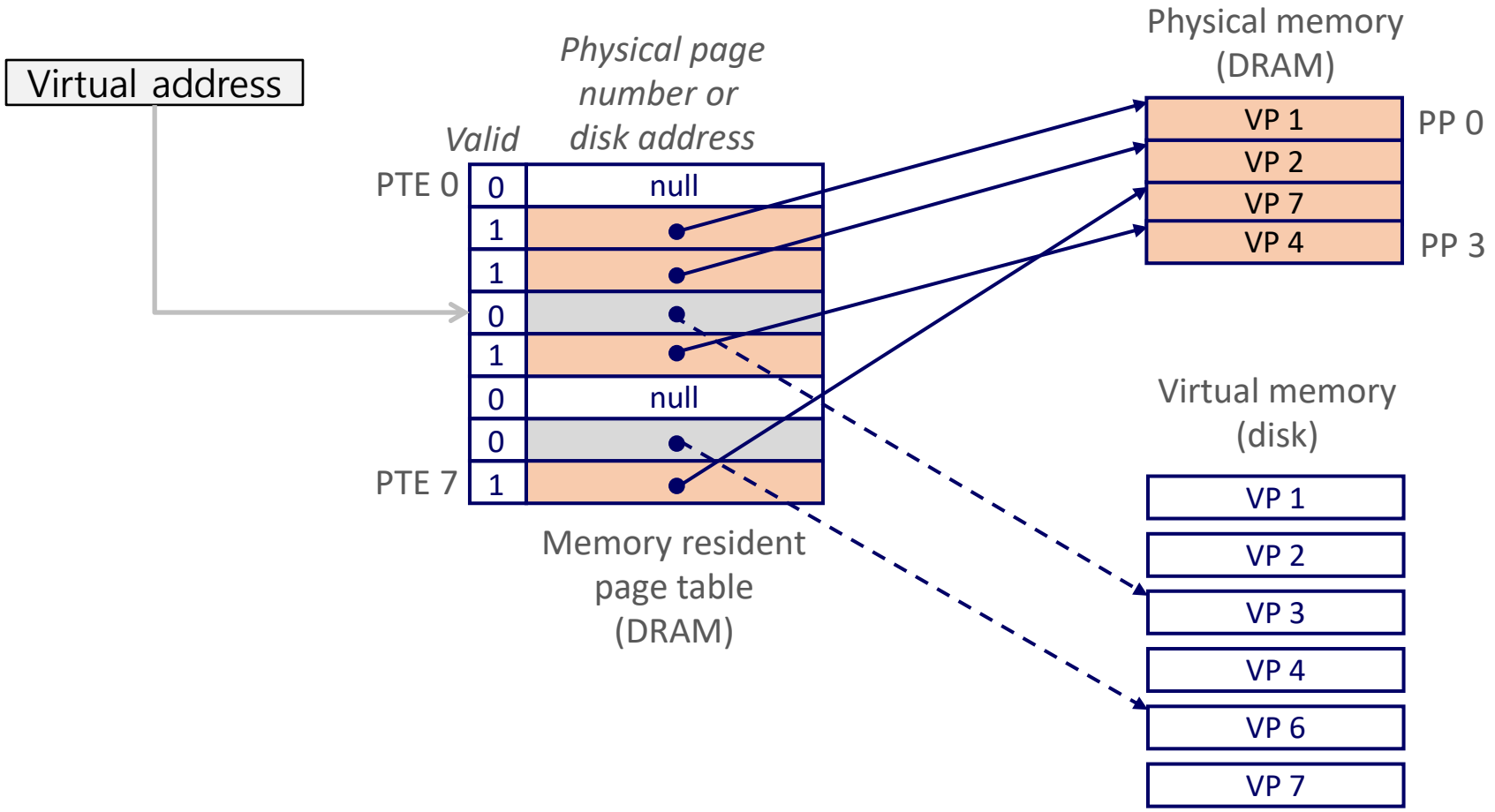
Page Fault

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



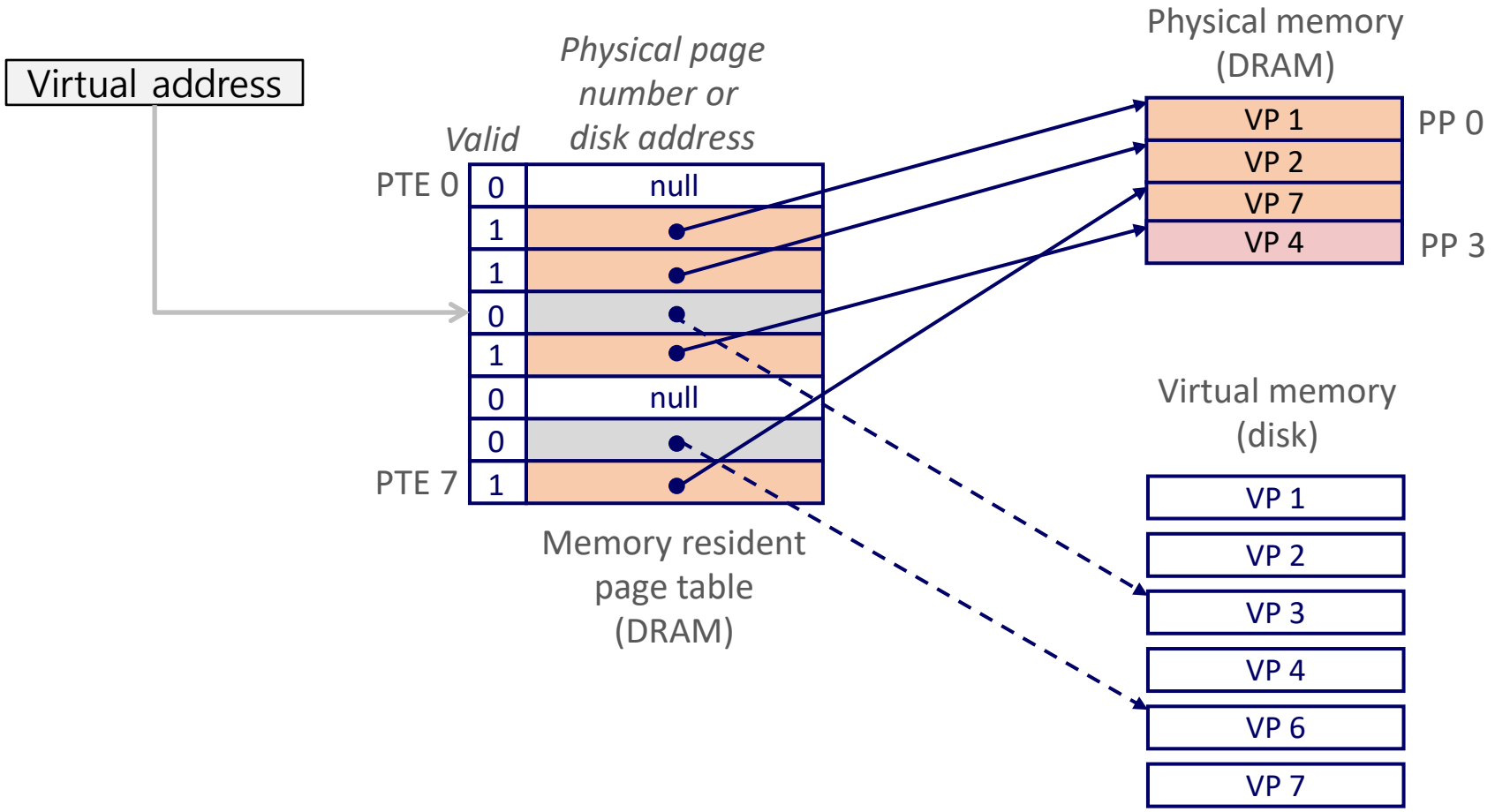
Handling Page Fault

- Page miss causes page fault (an exception)



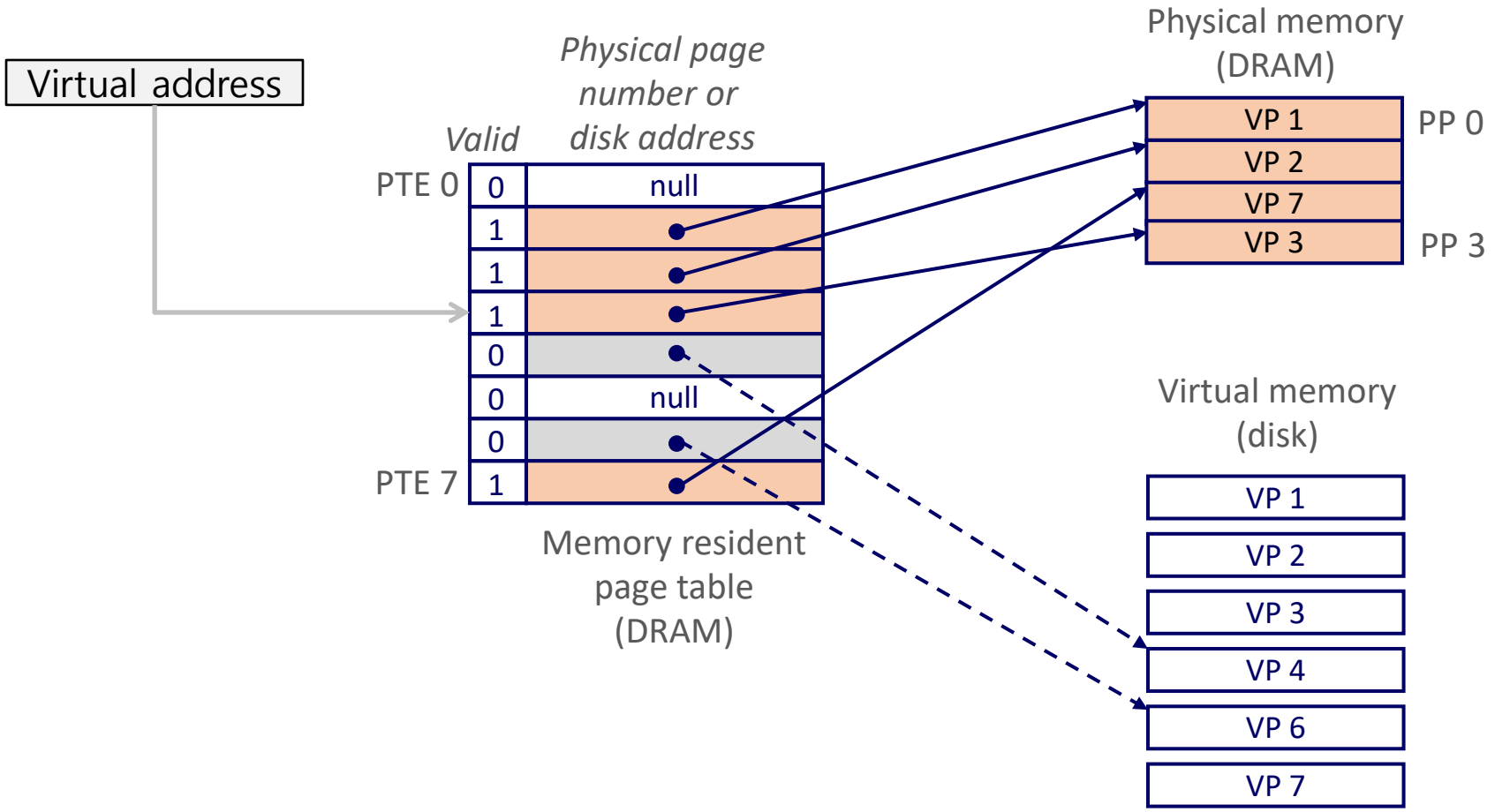
Handling Page Fault

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



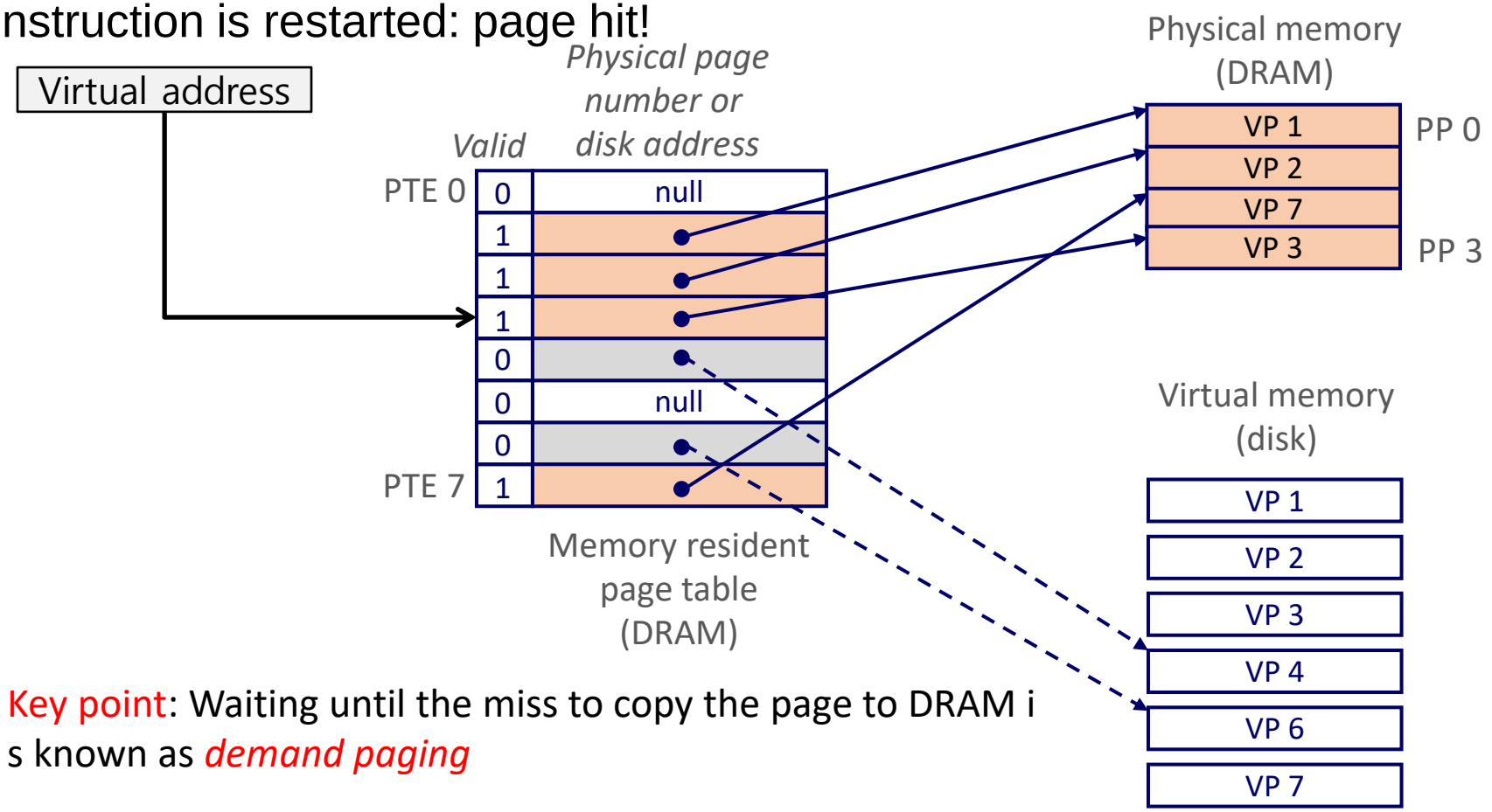
Handling Page Fault

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



Handling Page Fault

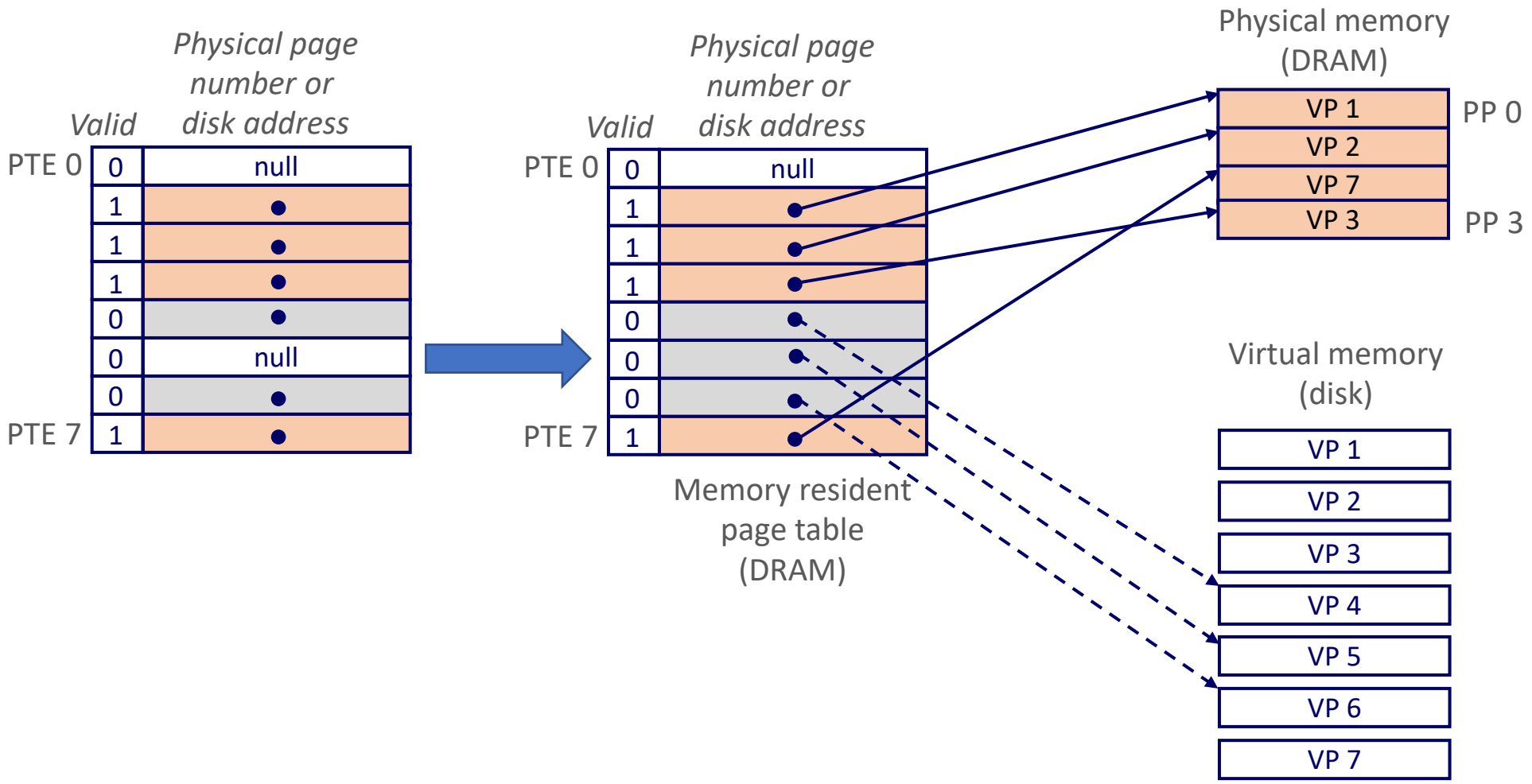
- Page miss causes page fault (an exception)
- Page fault handler selects a victim to be evicted (here VP 4)
- Offending instruction is restarted: page hit!



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

Allocating Pages

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



Locality to the Rescue Again!

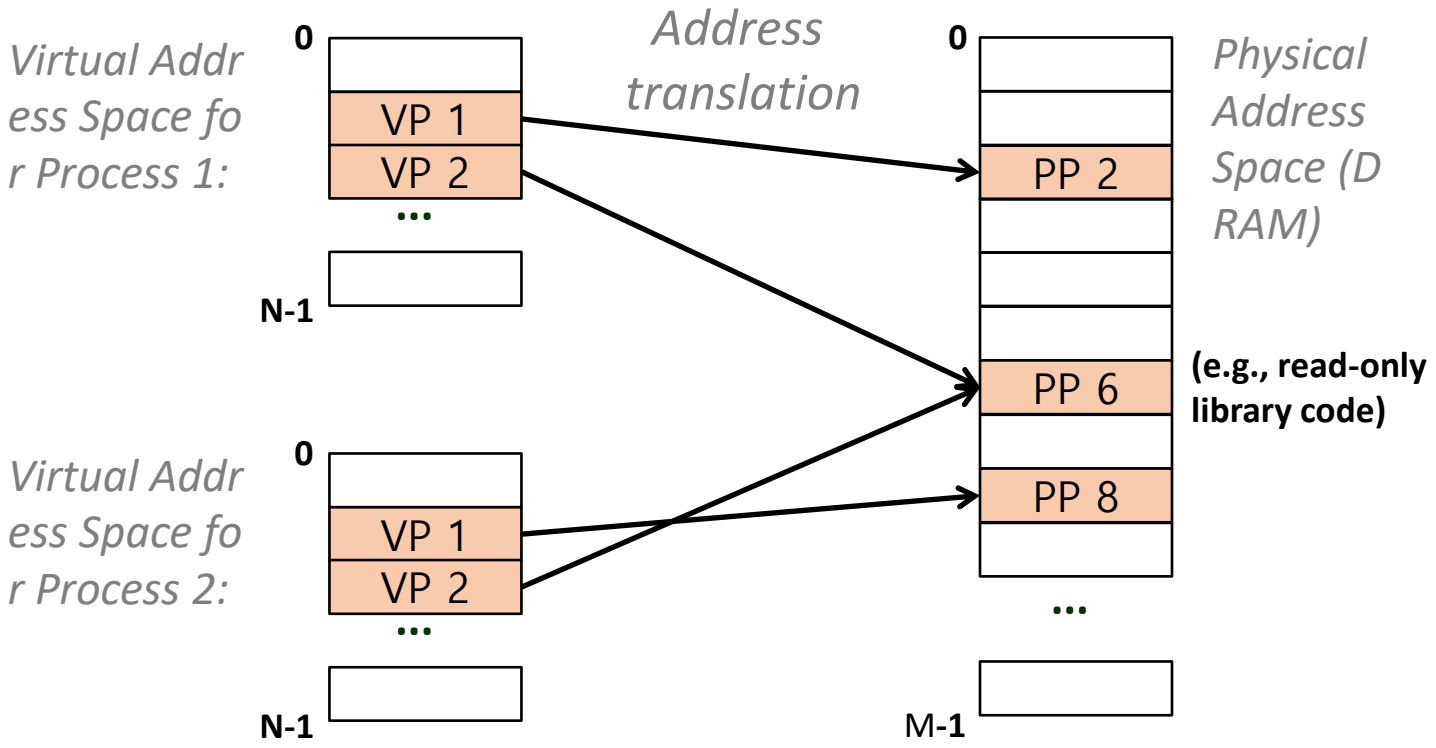
- Virtual memory seems terribly inefficient, but it works because of locality.
- At any point in time, programs tend to access a set of active virtual pages called the *working set*
  - Programs with better temporal locality will have smaller working sets
- If (working set size < main memory size)
  - Good performance for one process after compulsory misses
- If (SUM(working set sizes) > main memory size)
  - Need to move pages between disk and main memory!
  - *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
- Address translation

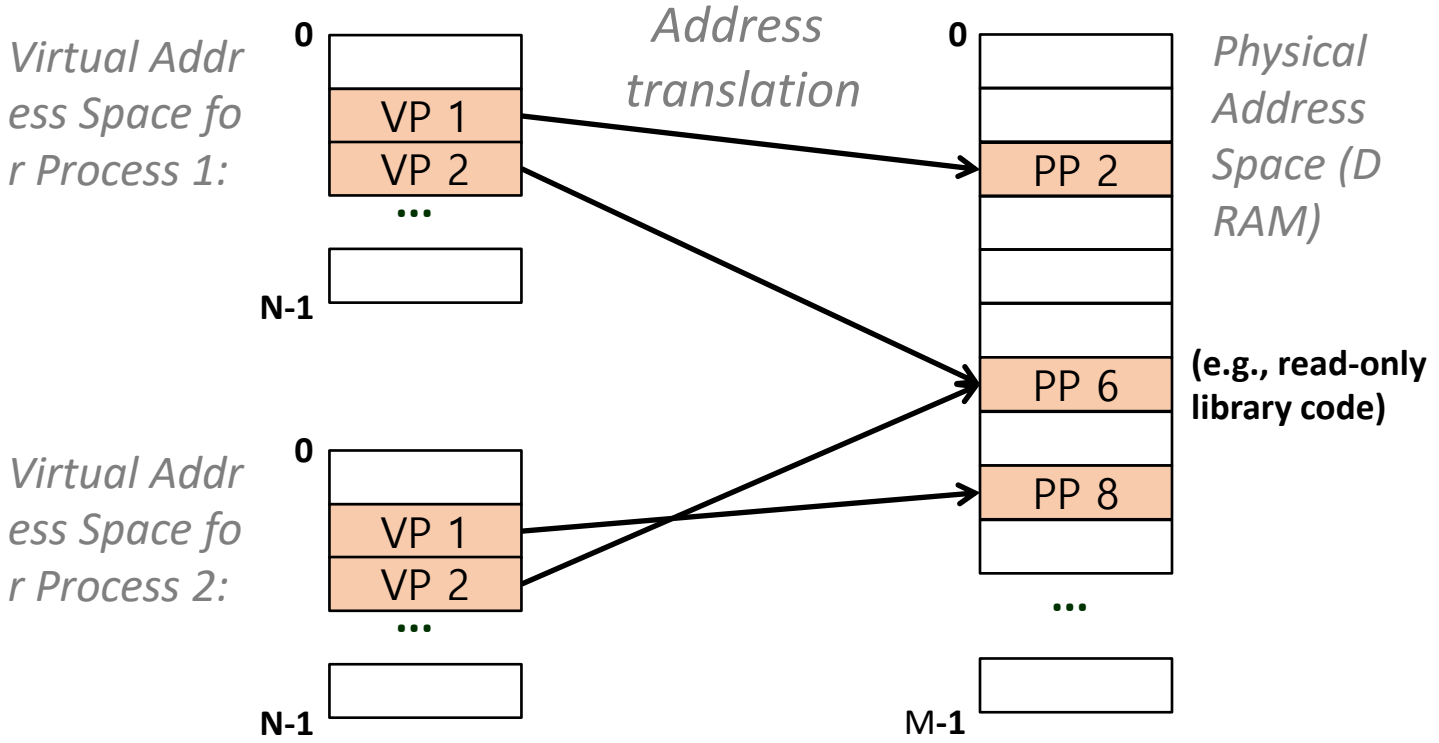
VM as a Tool for Memory Management

- 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)

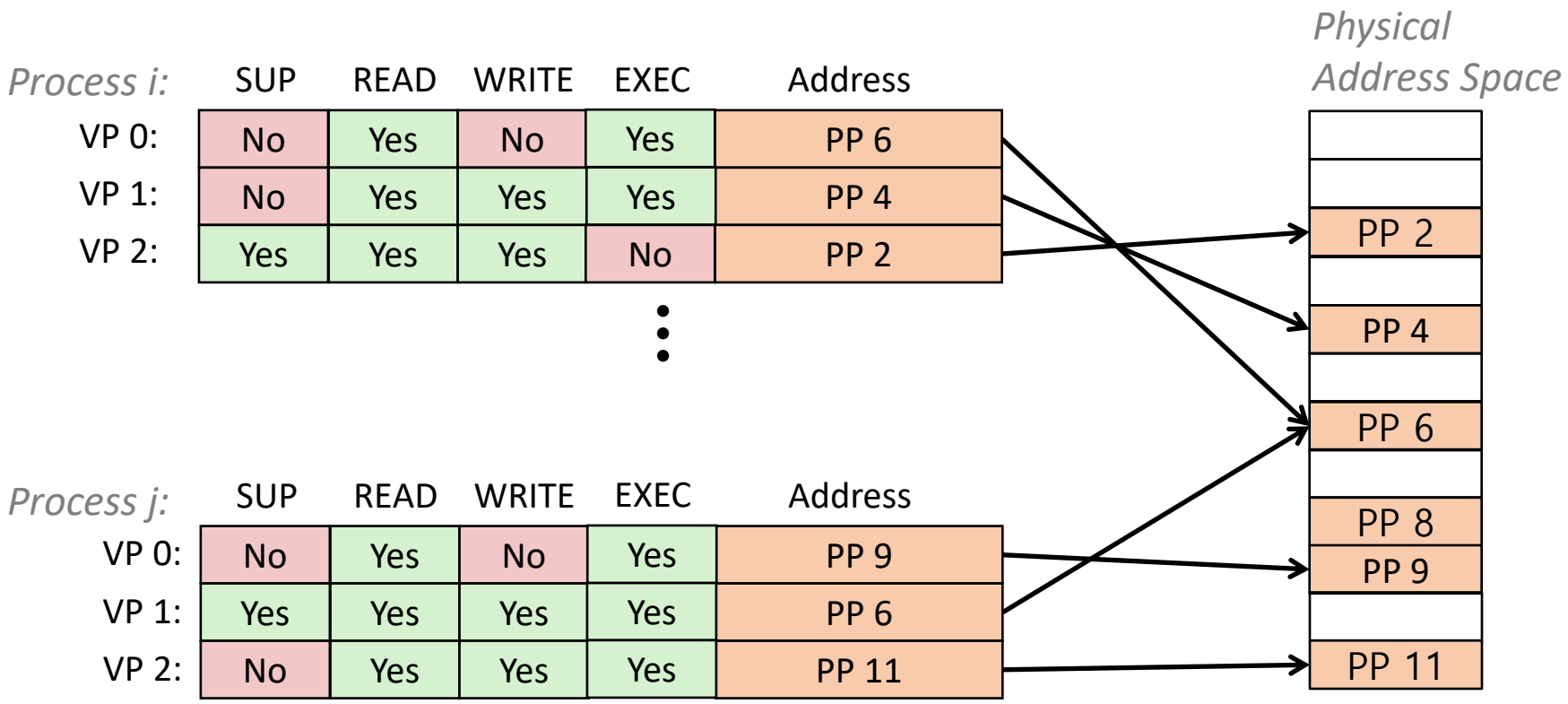


Today

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

VM as a Tool for Memory Protection

- Extend PTEs with permission bits
- MMU checks these bits on each access



Today

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

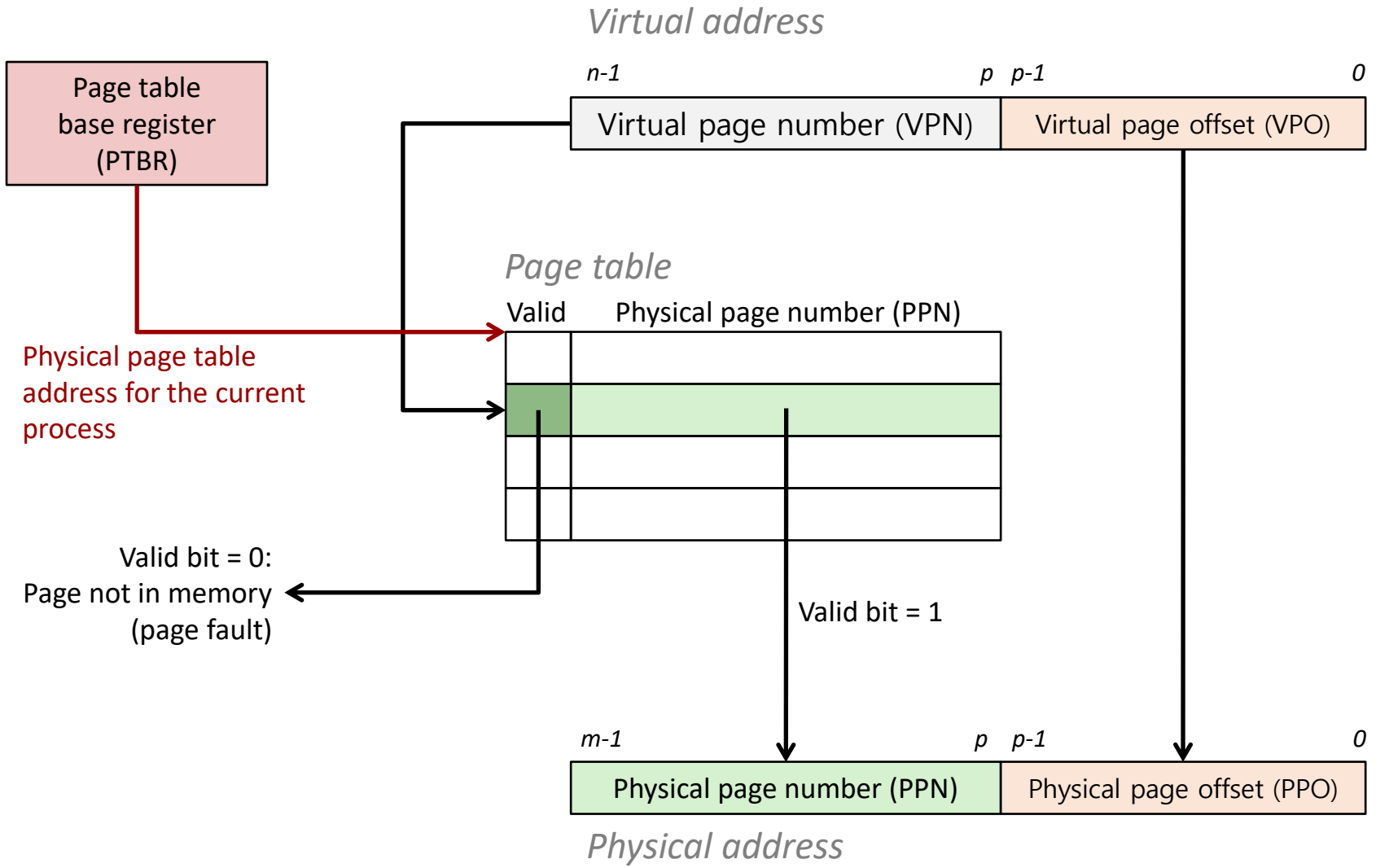
VM Address Translation

- Virtual Address Space
  - *V = {0, 1, …, N–1}*
- Physical Address Space
  - *P = {0, 1, …, M–1}*
- Address Translation
  - ***MAP: V → P U {∅}***
  - For virtual address ***a***:
    - ***MAP(a) = a'*** if data at virtual address ***a*** is at physical address ***a'*** in ***P***
    - ***MAP(a) = ∅*** if data at virtual address ***a*** is not in physical memory
      - Either invalid or stored on disk

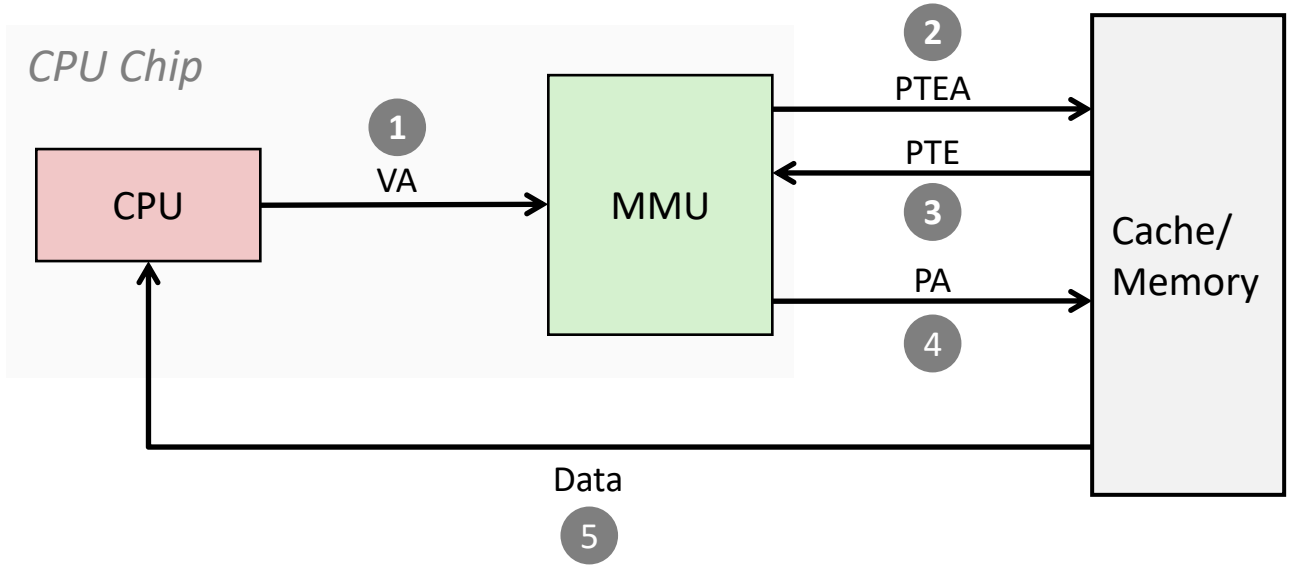
Summary of Address Translation Symbols

- Components of the virtual address (VA)
  - **VPO**: Virtual page offset
  - **VPN**: Virtual page number
- Components of the physical address (PA)
  - **PPO**: Physical page offset (same as VPO)
  - **PPN:** Physical page number

Address Translation With a Page Table



Address Translation: Page Hit



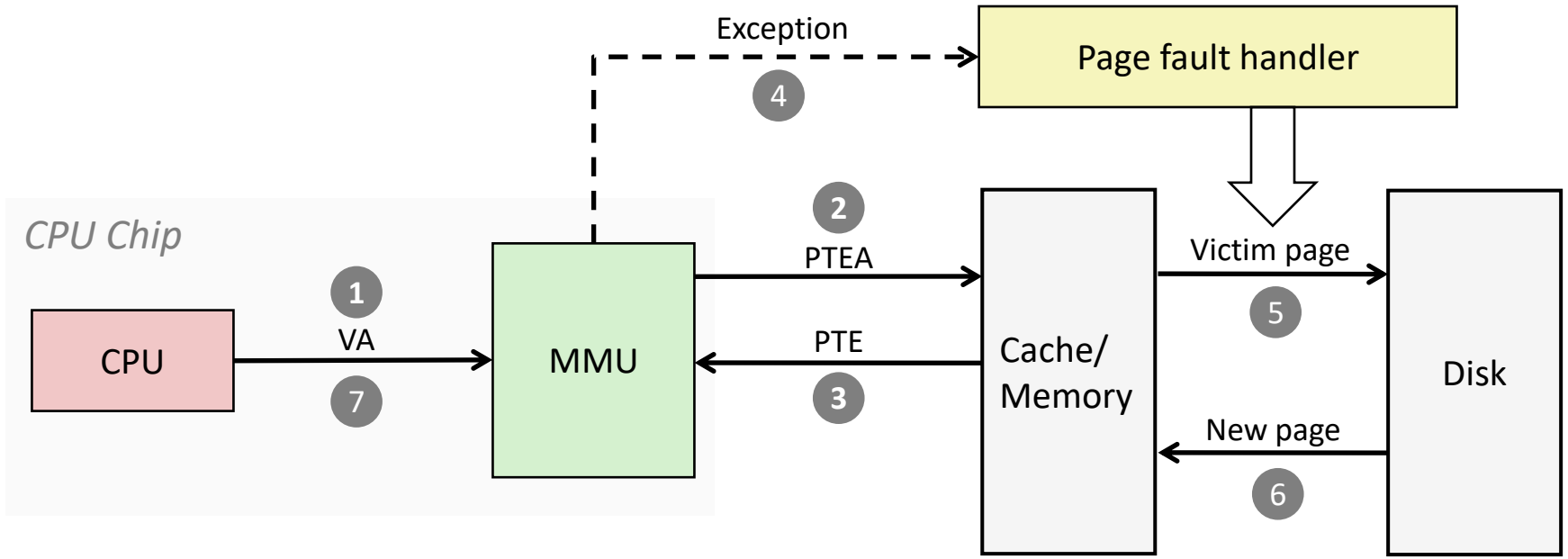
1) Processor sends virtual address to MMU

2-3) MMU fetches PTE from page table in memory

4) MMU sends physical address to cache/memory

5) Cache/memory sends data word to processor

Address Translation: Page Fault



1) Processor sends virtual address to MMU

2-3) MMU fetches PTE from page table in memory

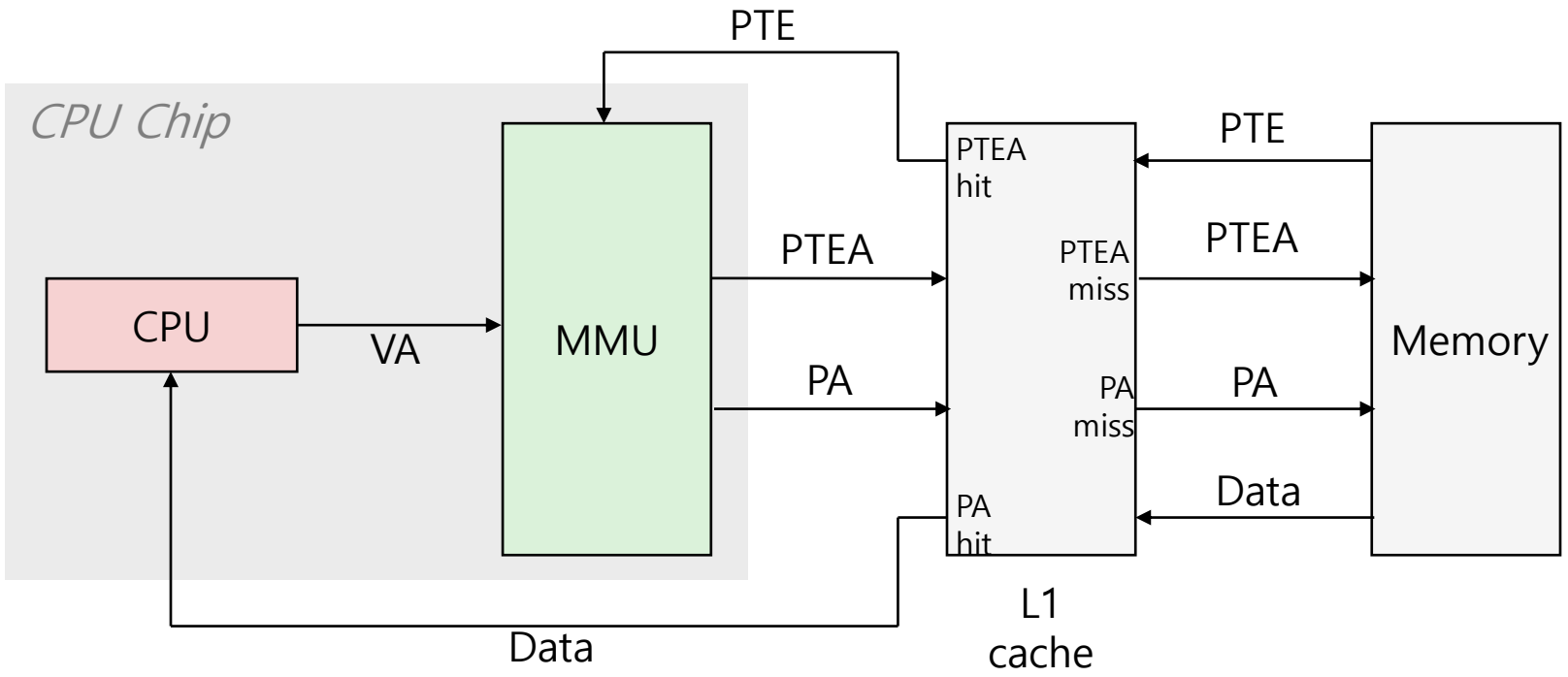
4) Valid bit is zero, so MMU triggers page fault exception

5) Handler identifies victim (and, if dirty, pages it out to disk)

6) Handler pages in new page and updates PTE in memory

7) Handler returns to original process, restarting faulting instruction

Integrating VM and Cache



VA: virtual address, PA: physical address, PTE: page table entry, PTEA = PTE address

Summary

- Programmer’s view of virtual memory
  - Each process has its own private linear address space
  - Cannot be corrupted by other processes

- System view of virtual memory
  - Uses memory efficiently by caching virtual memory pages
    - Efficient only because of locality
  - Simplifies memory management and programming
  - Simplifies protection by providing a convenient interpositioning point to check permissions