

# Virtual Memory: Concepts

CSE4100: System Programming

Youngjae Kim (PhD)

<https://sites.google.com/site/youkim/home>

Distributed Computing and Operating Systems Laboratory (DISCOS)

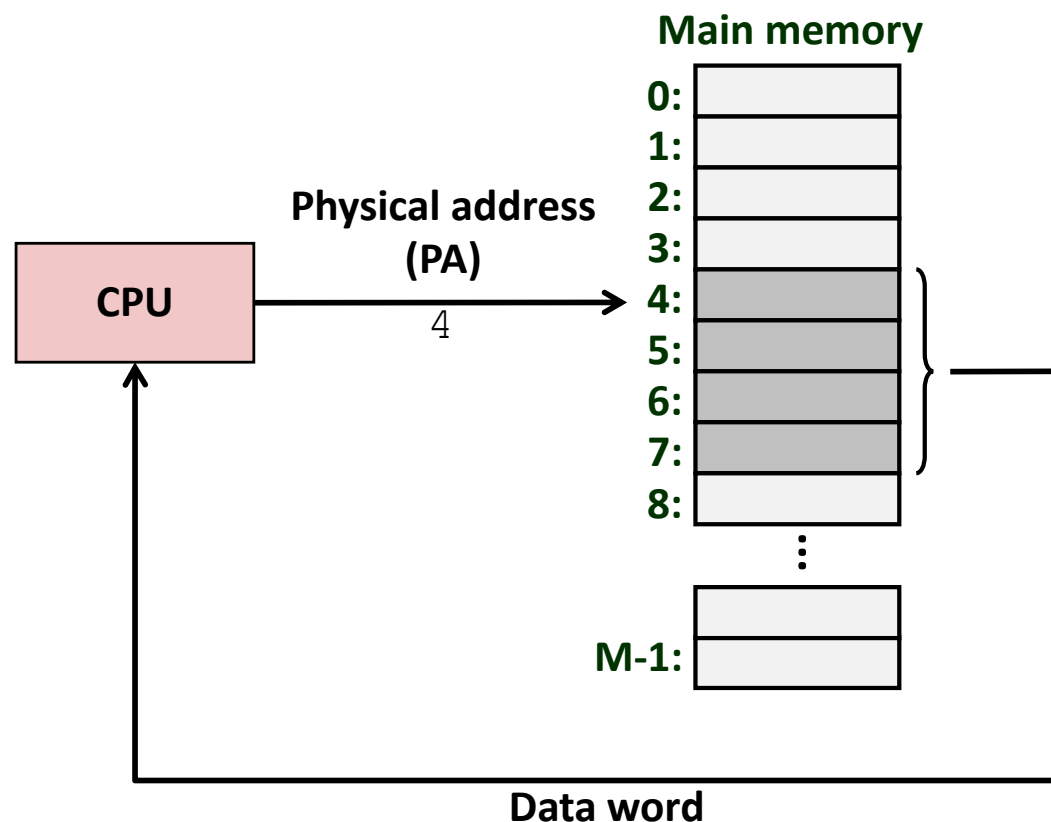
<https://discos.sogang.ac.kr>

Office: R911, E-mail: [youkim@sogang.ac.kr](mailto:youkim@sogang.ac.kr)

# Today

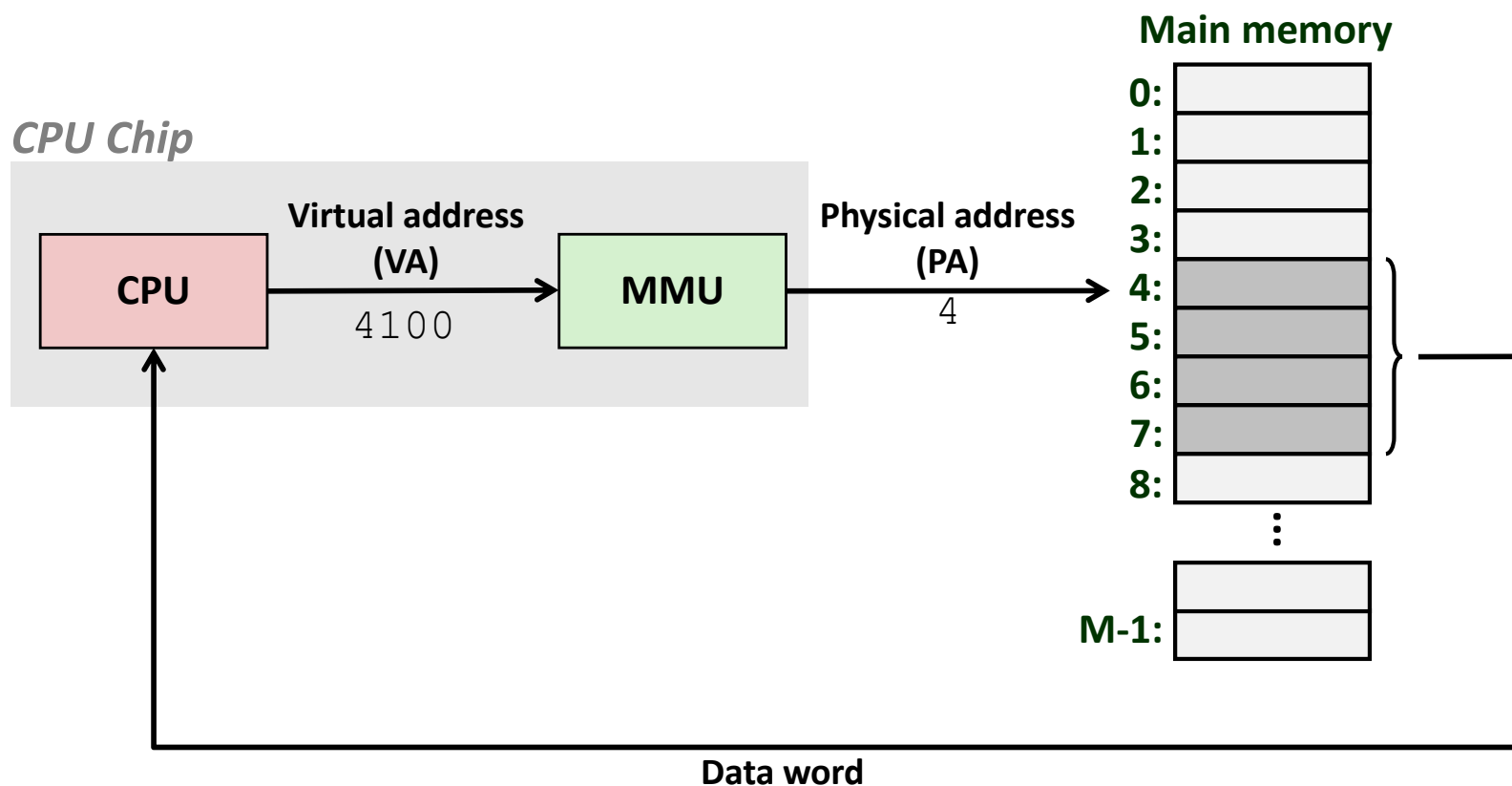
- **Address spaces**
- VM as a tool for caching
- VM as a tool for memory management
- VM as a tool for memory protection

# A System Using Physical Addressing



- Used in “simple” systems like embedded microcontrollers in devices like 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:

$\{0, 1, 2, 3 \dots \}$

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

$\{0, 1, 2, 3, \dots, N-1\}$

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

$\{0, 1, 2, 3, \dots, M-1\}$

# Why Virtual Memory (VM)?

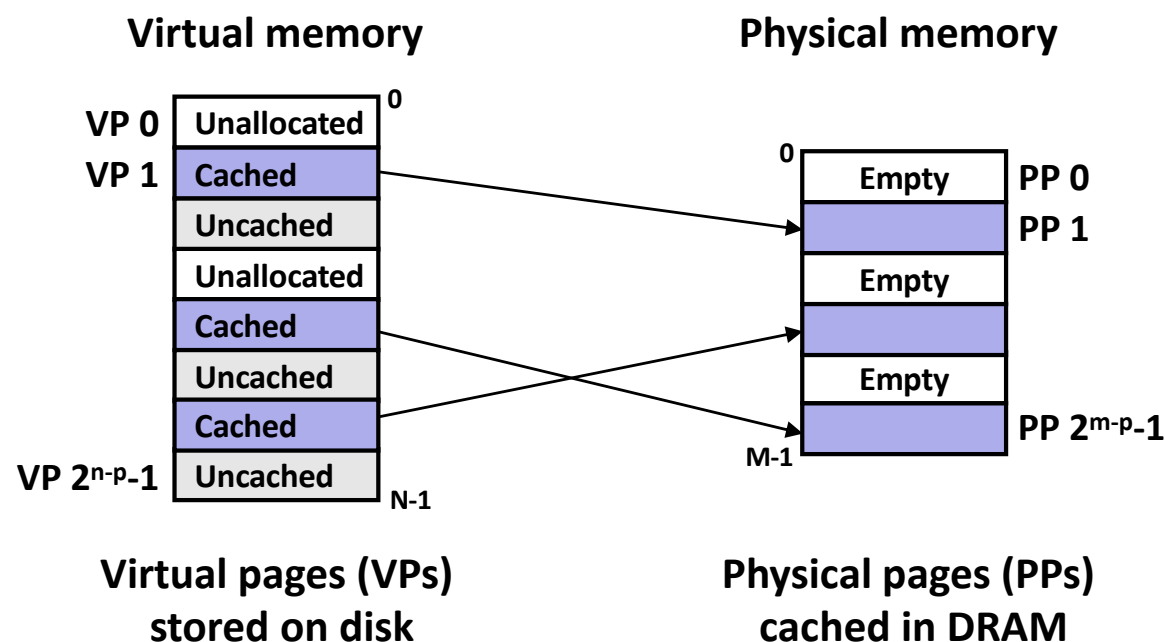
- **Uses main memory efficiently**
  - Use DRAM as a cache for parts of a virtual address space
- **Simplifies memory management**
  - Each process gets the same uniform linear address space
- **Isolates address spaces**
  - One process can't interfere with another's memory
  - User program cannot access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

# VM as a Tool for Caching

- Conceptually, *virtual memory* is an array of  $N$  contiguous bytes stored on disk.
- The contents of the array on disk are cached in *physical memory (DRAM cache)*
  - These cache blocks are called *pages* (size is  $P = 2^p$  bytes)





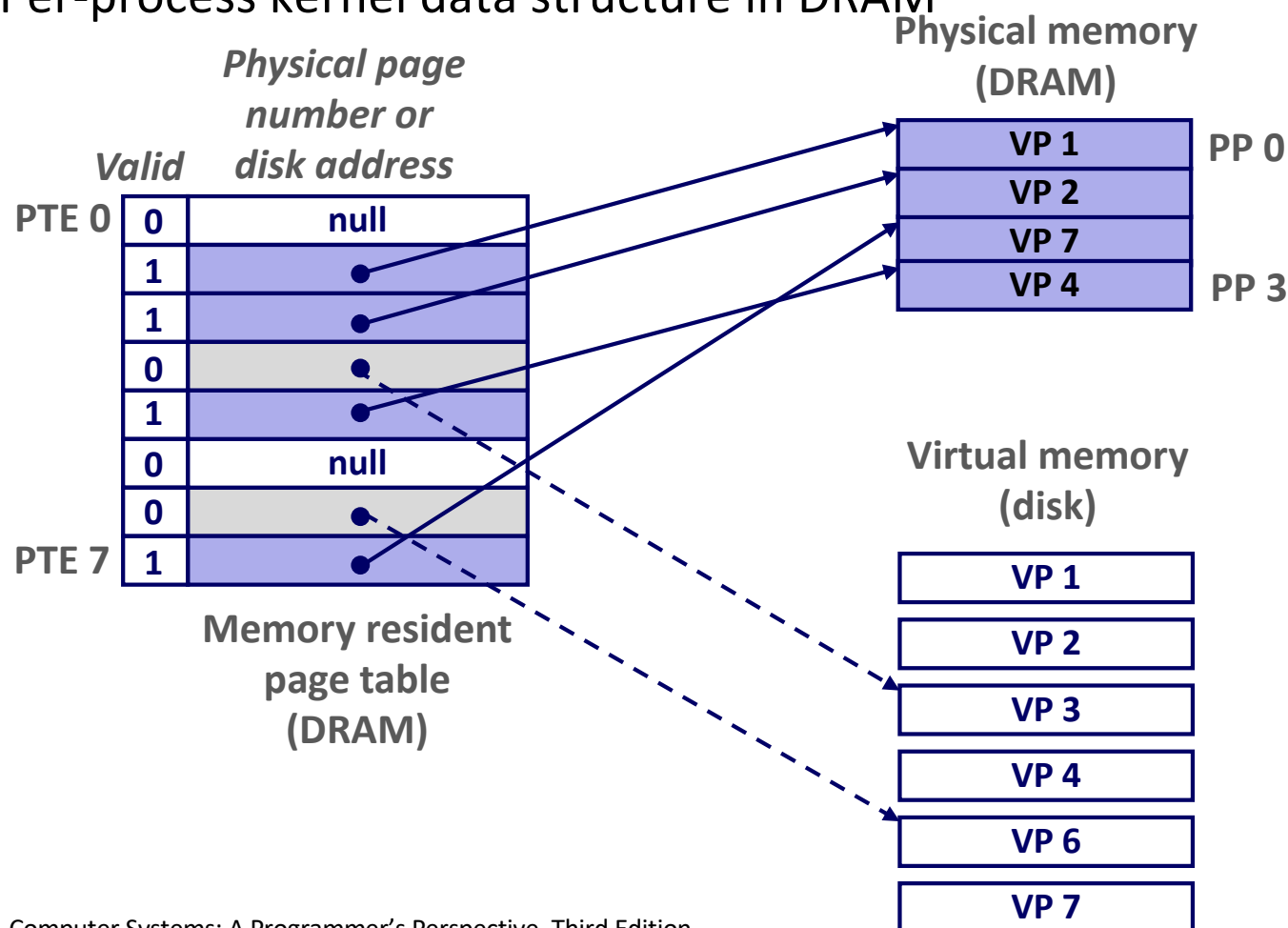
# DRAM Cache Organization

- **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: typically 4 KB, sometimes 4 MB
  - Fully associative
    - Any VP can be placed in any PP
    - Requires a “large” mapping function – different from cache memories
  - 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

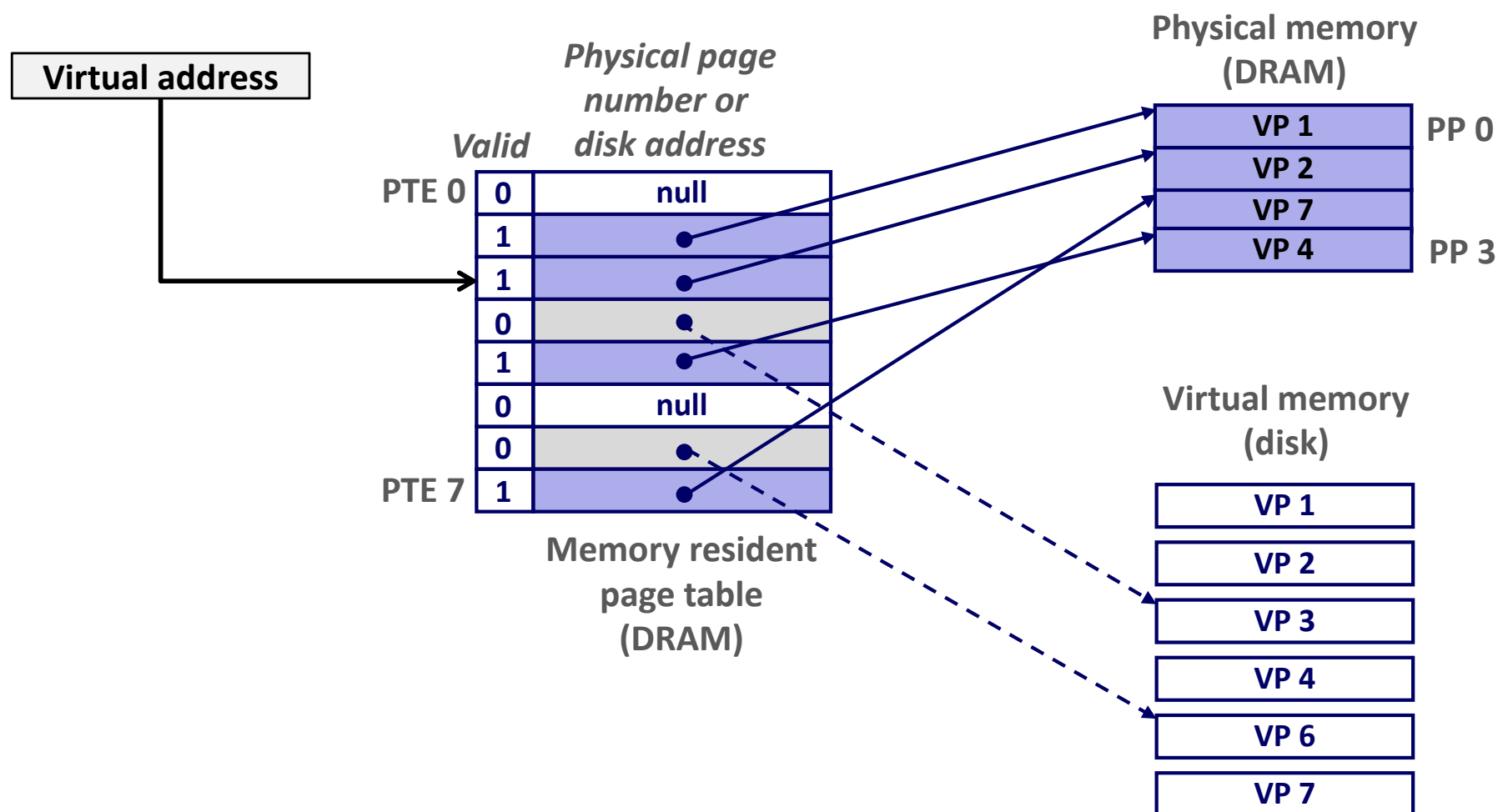
- A **page table** is an array of page table entries (PTEs) that maps virtual pages to physical pages.

- Per-process kernel data structure in DRAM



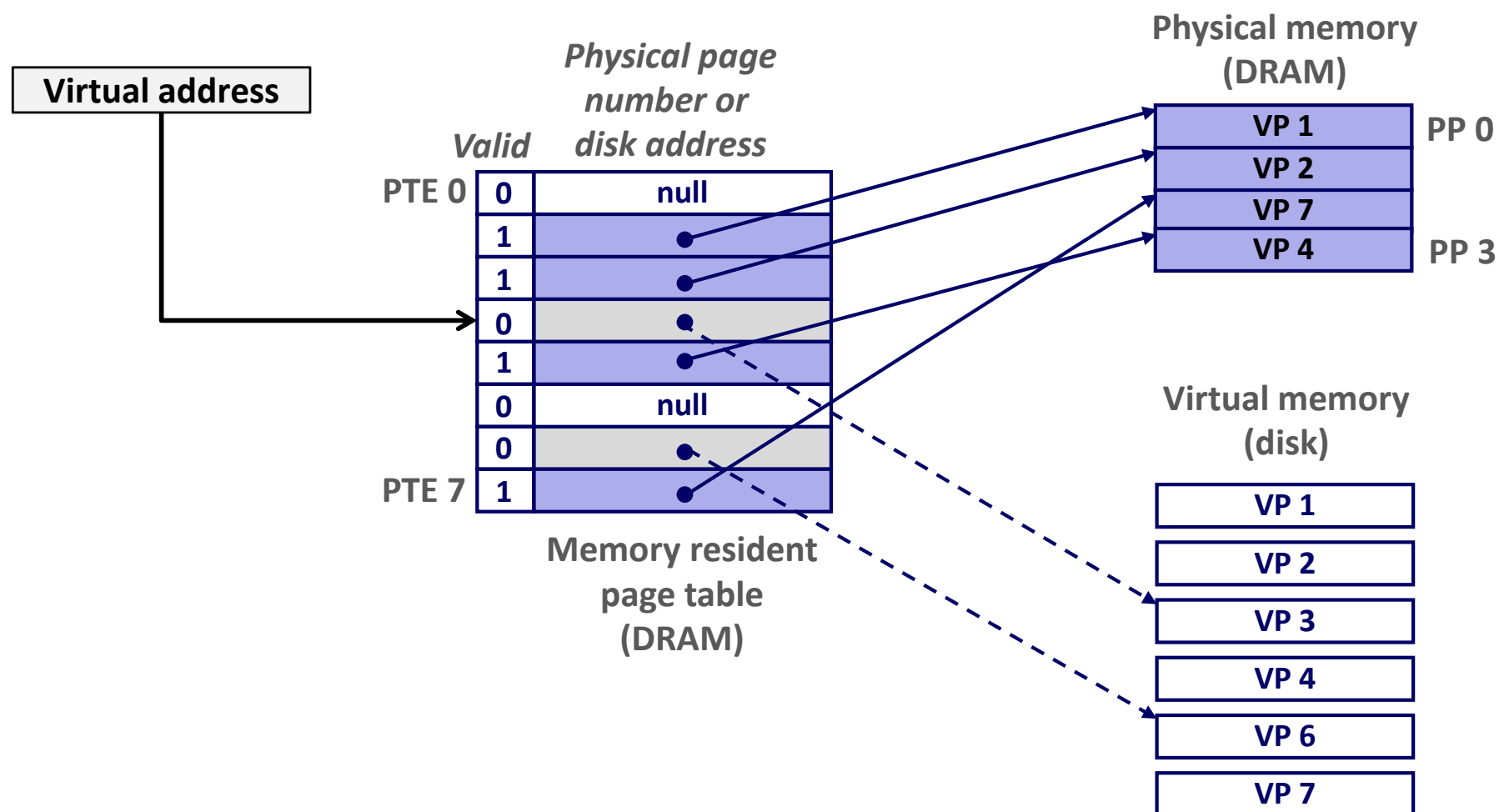
# Page Hit

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



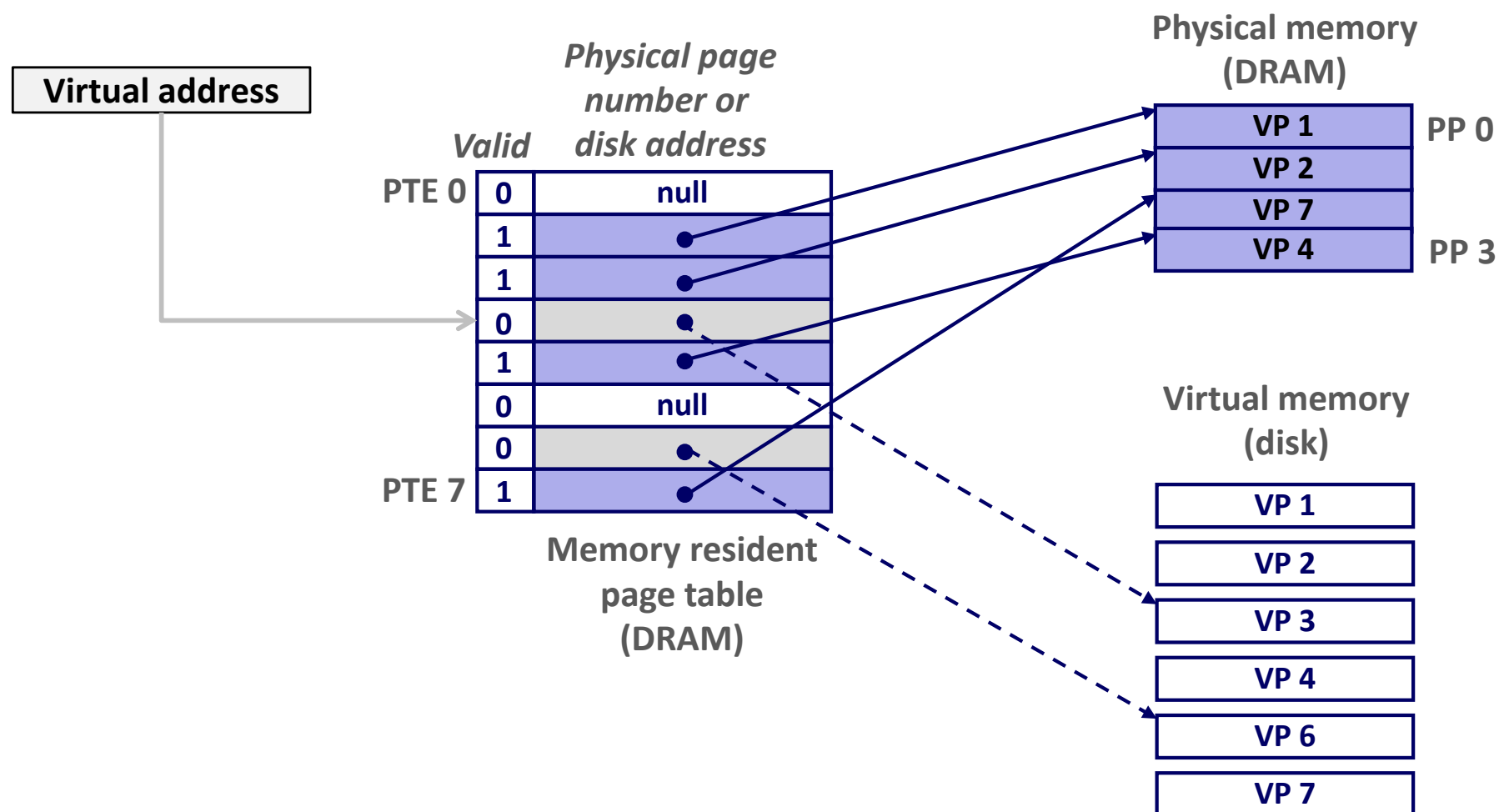
# Page Fault

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



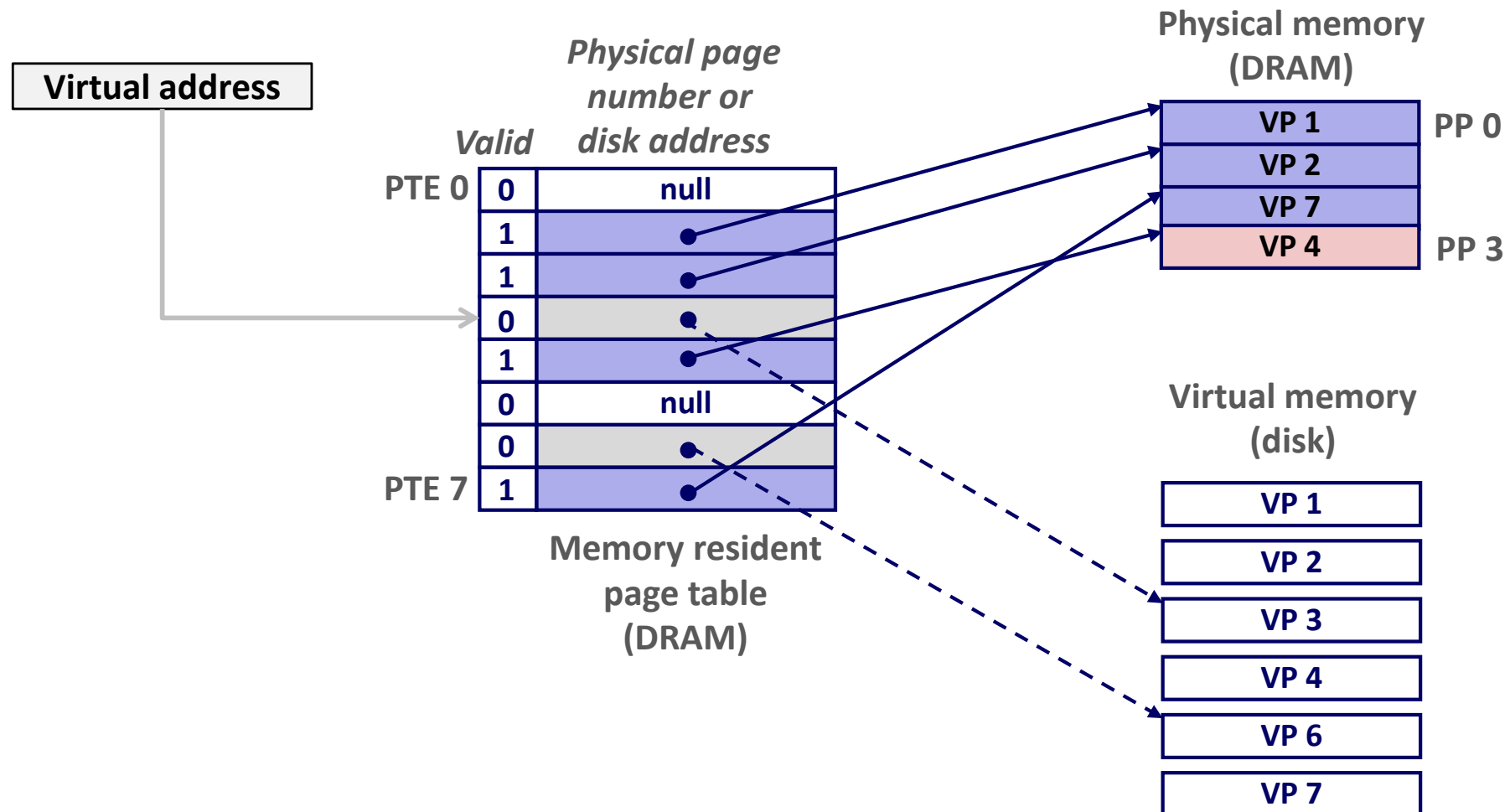
# Handling Page Fault

- Page miss causes page fault (an exception)



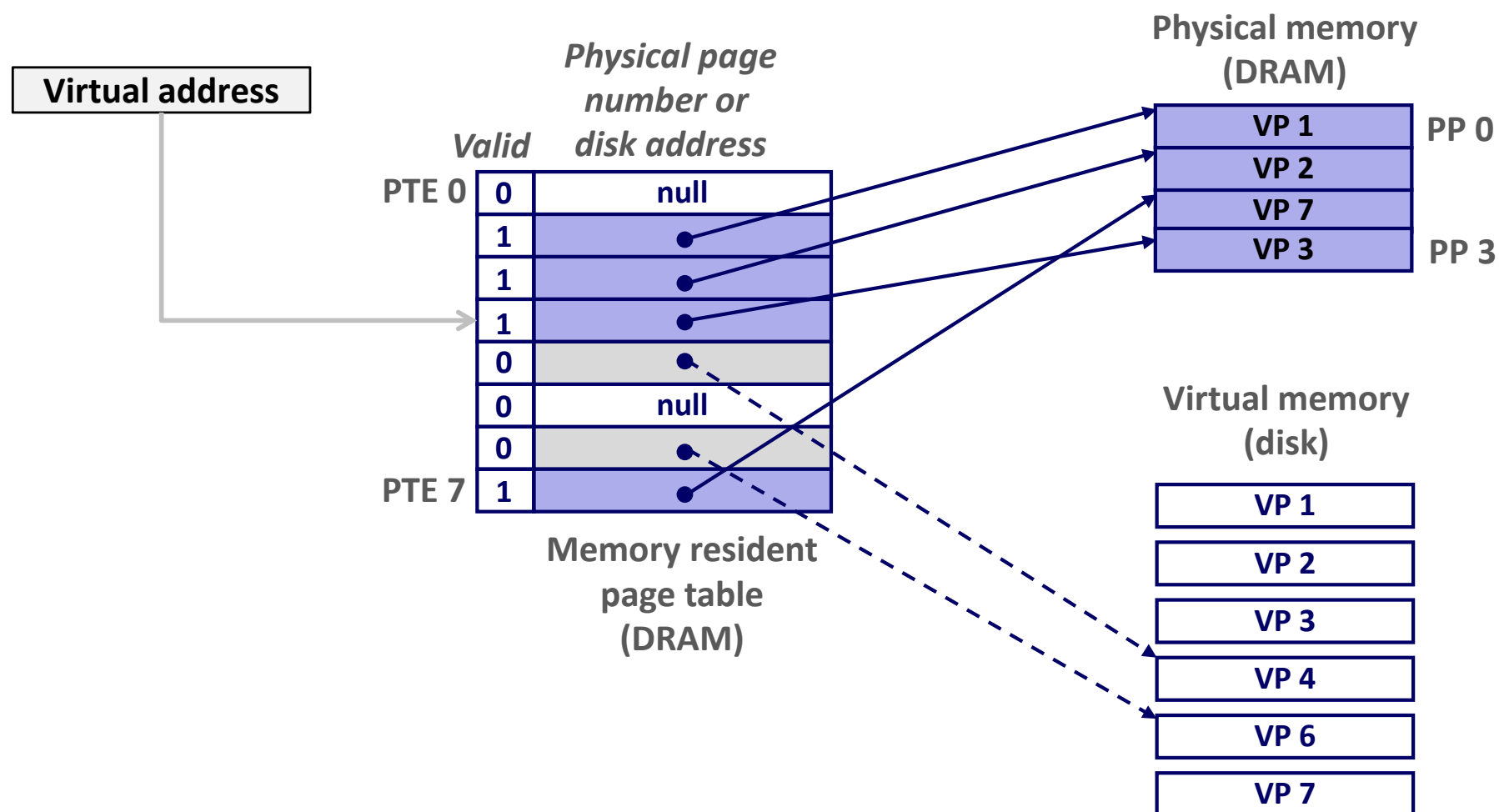
# Handling Page Fault

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



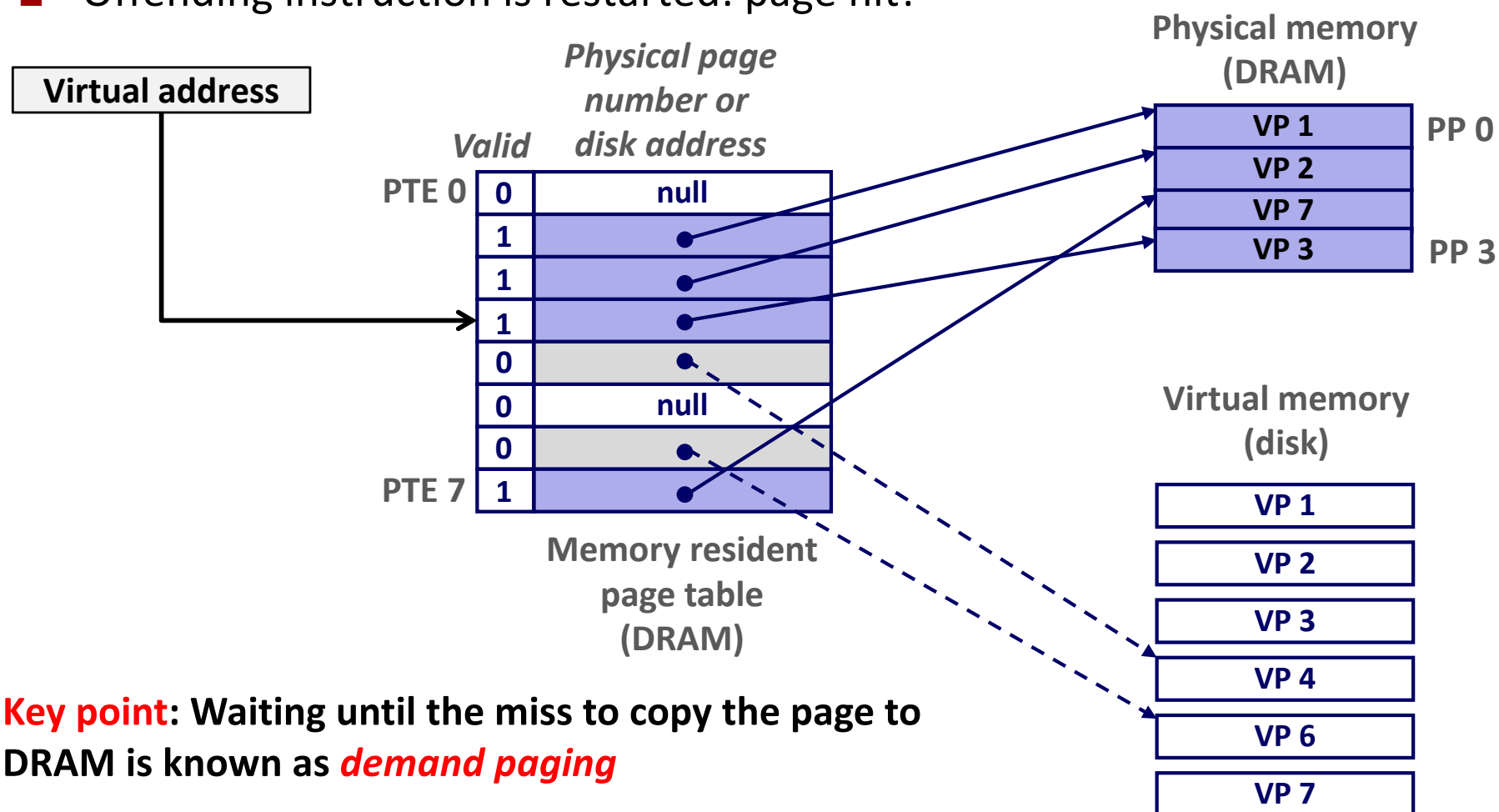
# Handling Page Fault

- 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!

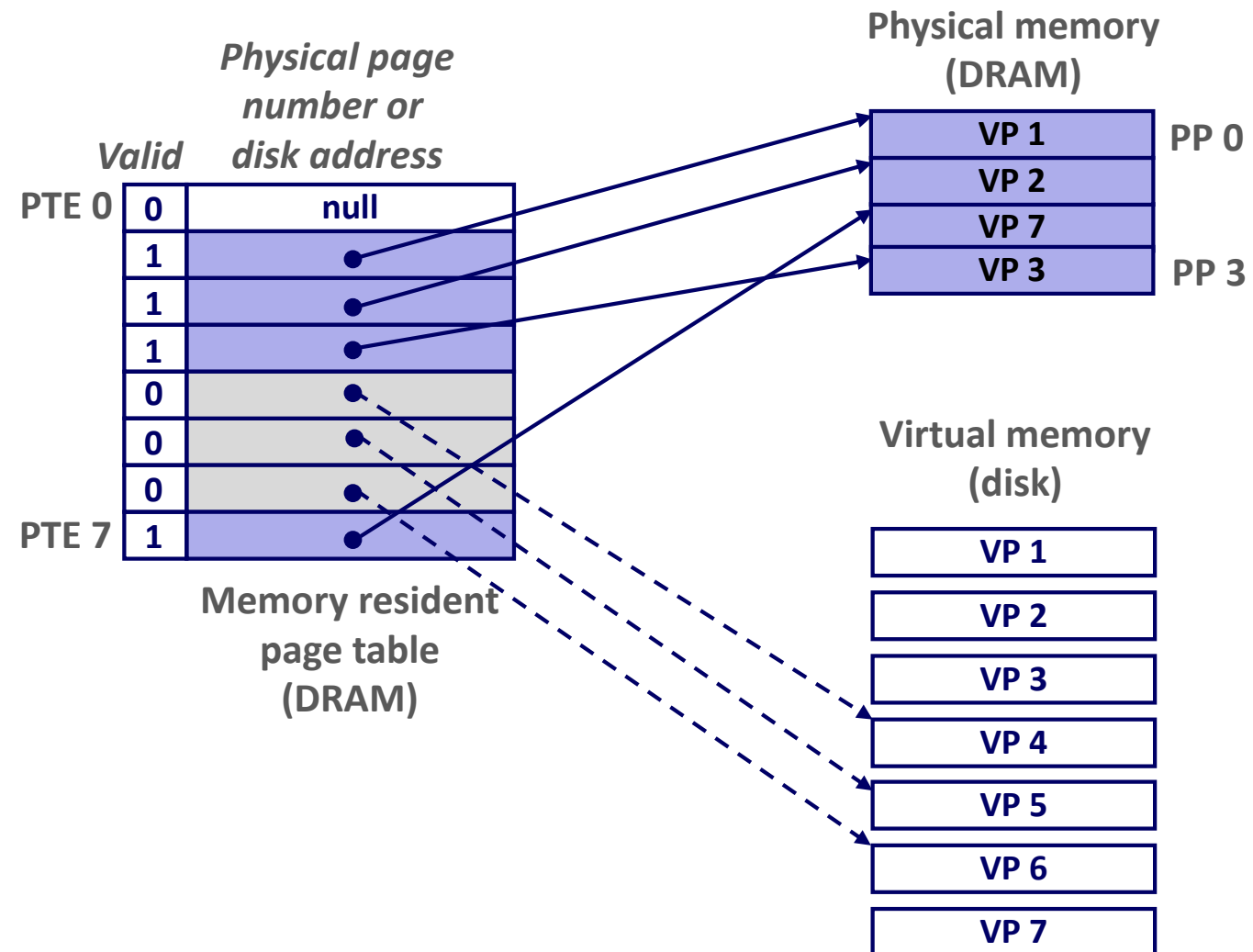


**Key point:** Waiting until the miss to copy the page to DRAM is known as **demand paging**



# Allocating Pages

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



# Locality to the Rescue Again!

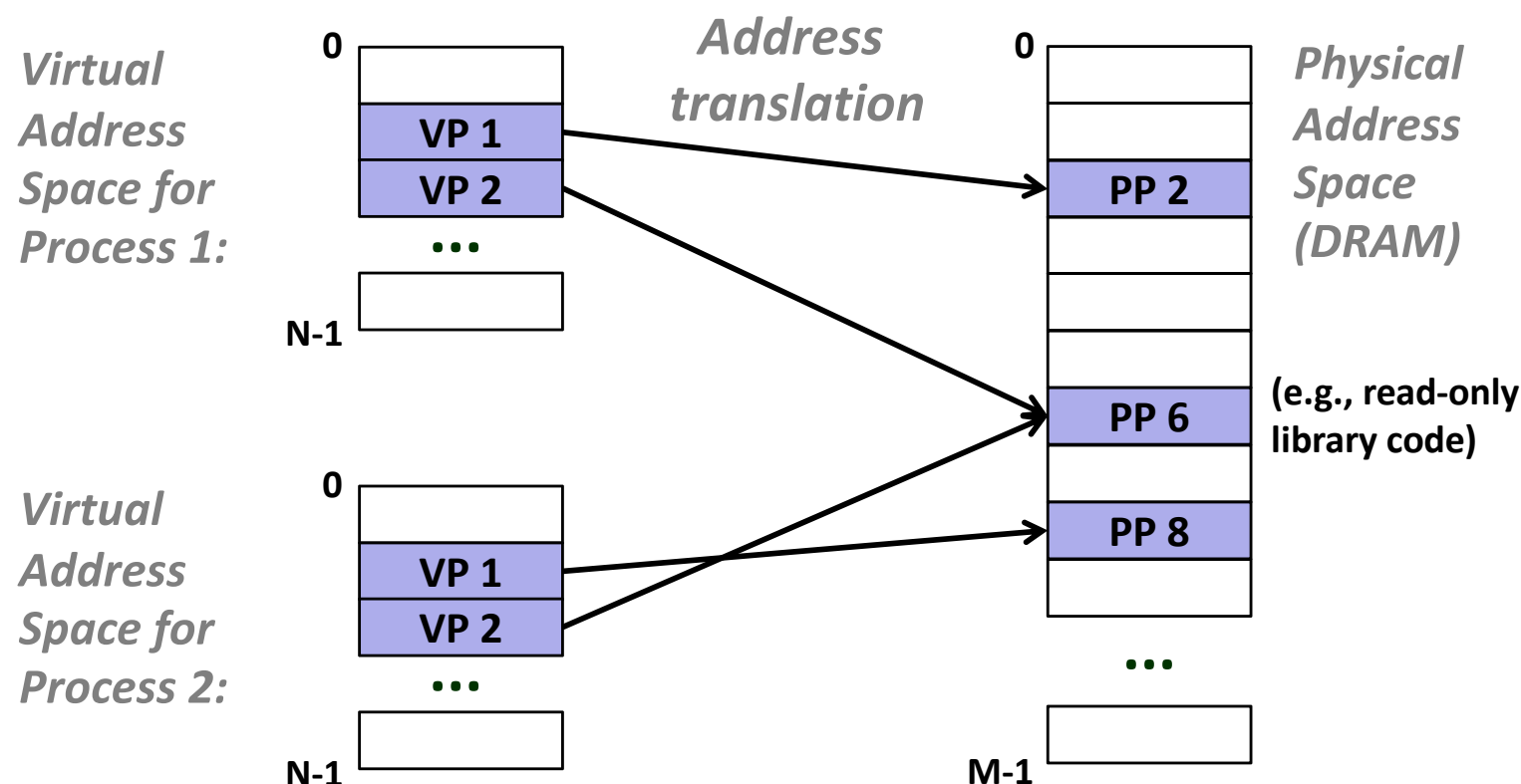
- Virtual memory seems terribly inefficient, but it works because of locality.
- At any point in time, programs tend to access a set of active virtual pages called the *working set*
  - Programs with better temporal locality will have smaller working sets
- If (working set size < main memory size)
 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



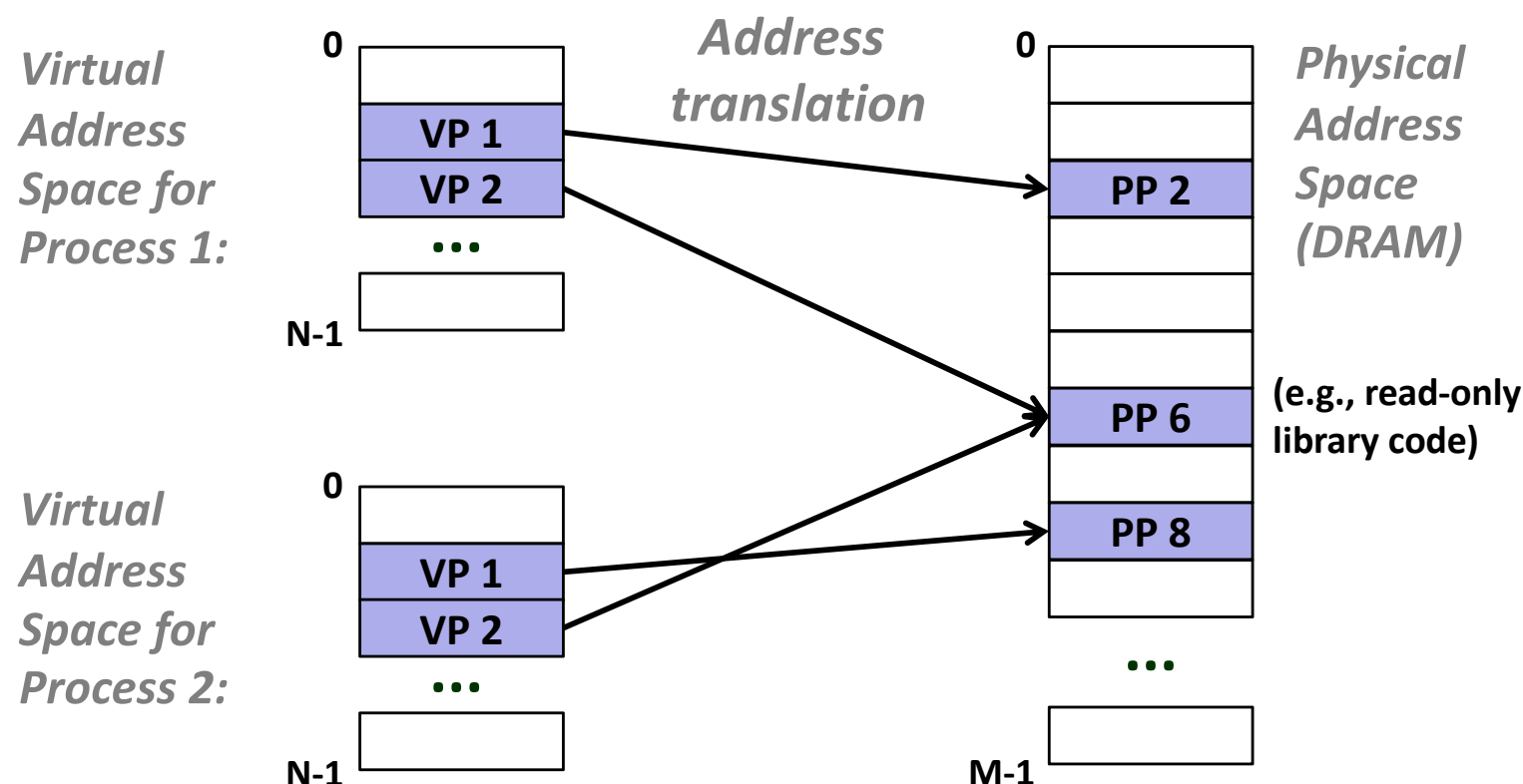
# VM as a Tool for Memory Management

## ■ 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)



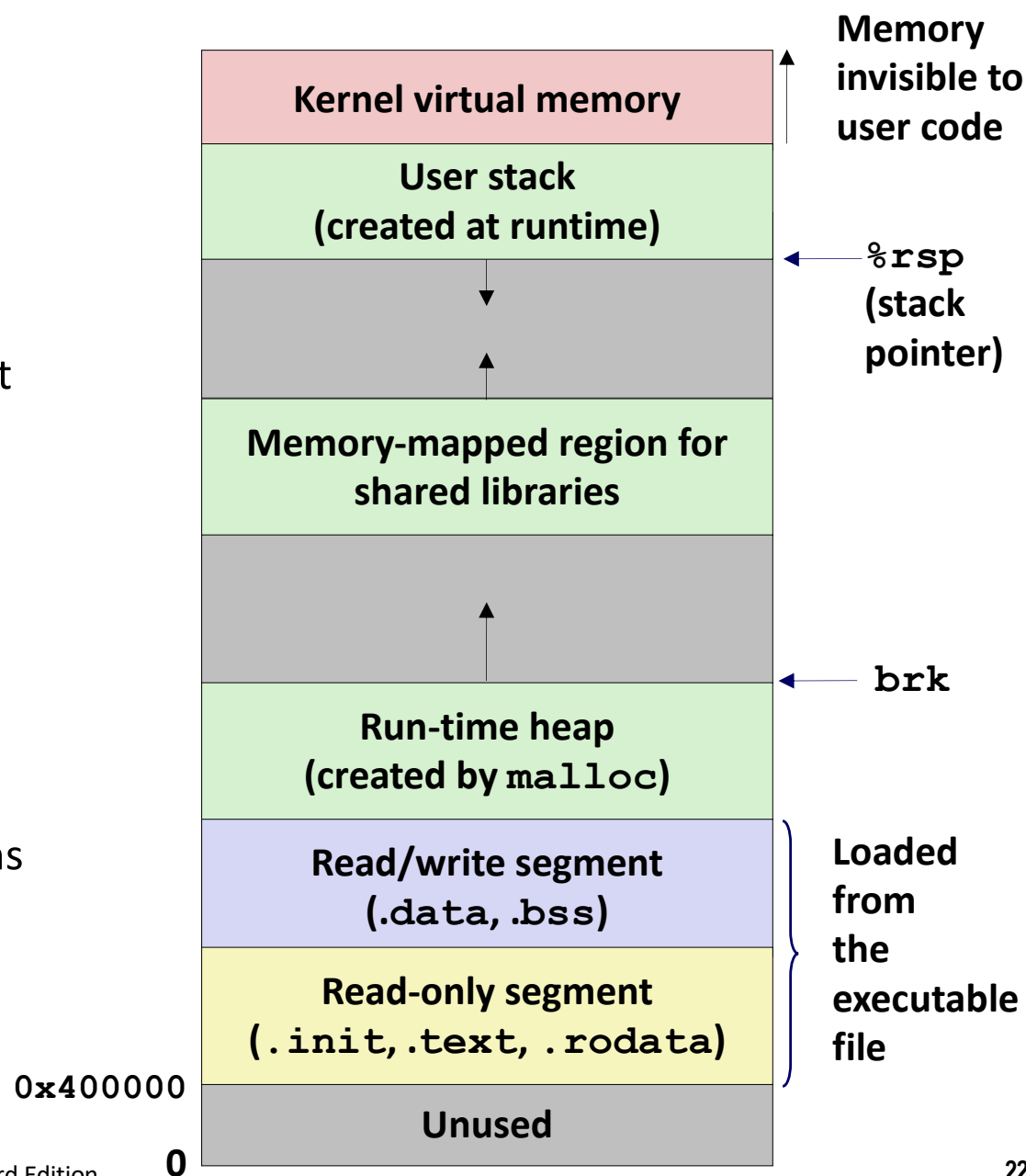
# Simplifying Linking and Loading

## ■ Linking

- Each program has similar virtual address space
- Code, data, and heap always start at the same addresses.

## ■ Loading

- **execve** allocates virtual pages for `.text` and `.data` sections & creates PTEs marked as invalid
- The `.text` and `.data` sections are copied, page by page, on demand by the virtual memory system

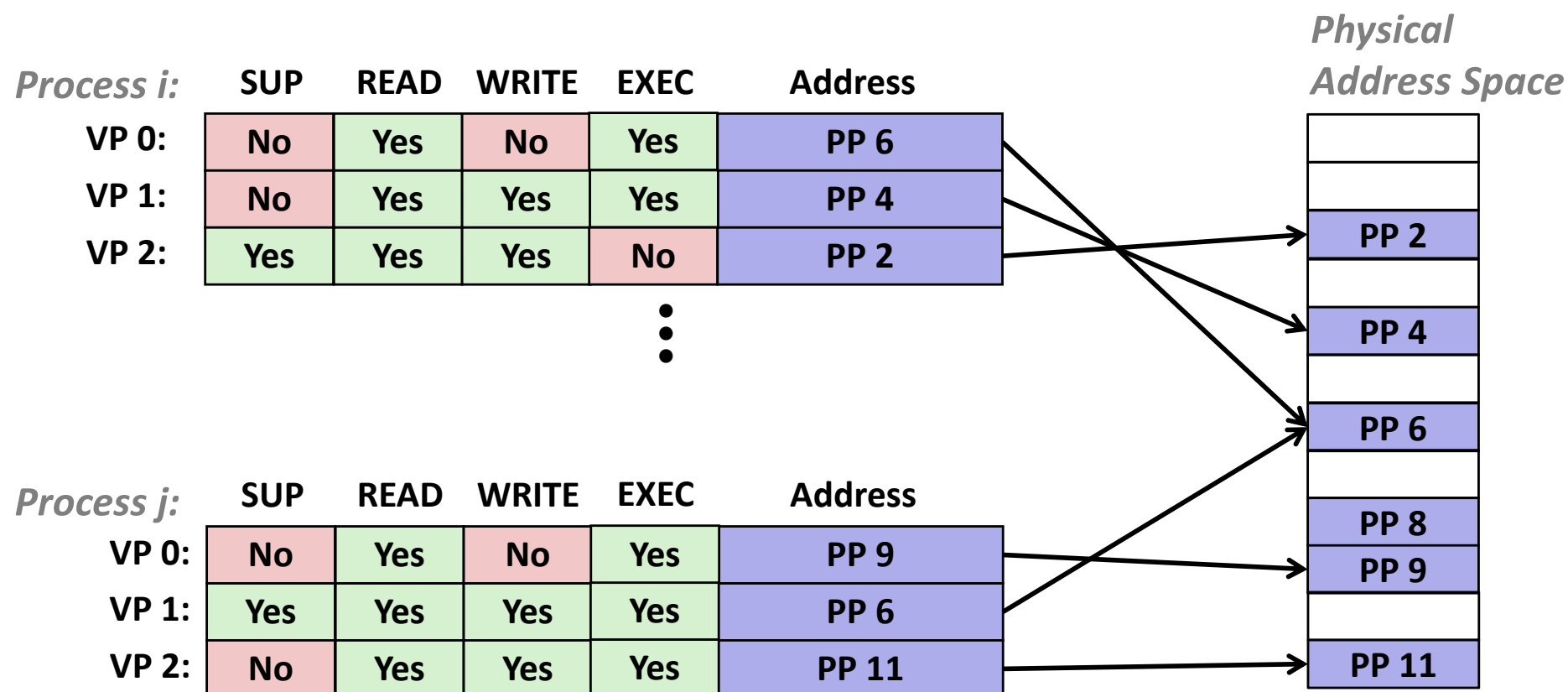


# Today

- 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 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