CS3281 / CS5281

Virtual Memory

CS3281 / CS5281

Spring 2026

**Some lecture slides borrowed and adapted from CMU's "Computer Systems: A Programmer's Perspective"*

Tel (615) 343-7472 | Fax (615) 343-7440
1025 16th Avenue South Nashville, TN 37212
www.isis.vanderbilt.edu

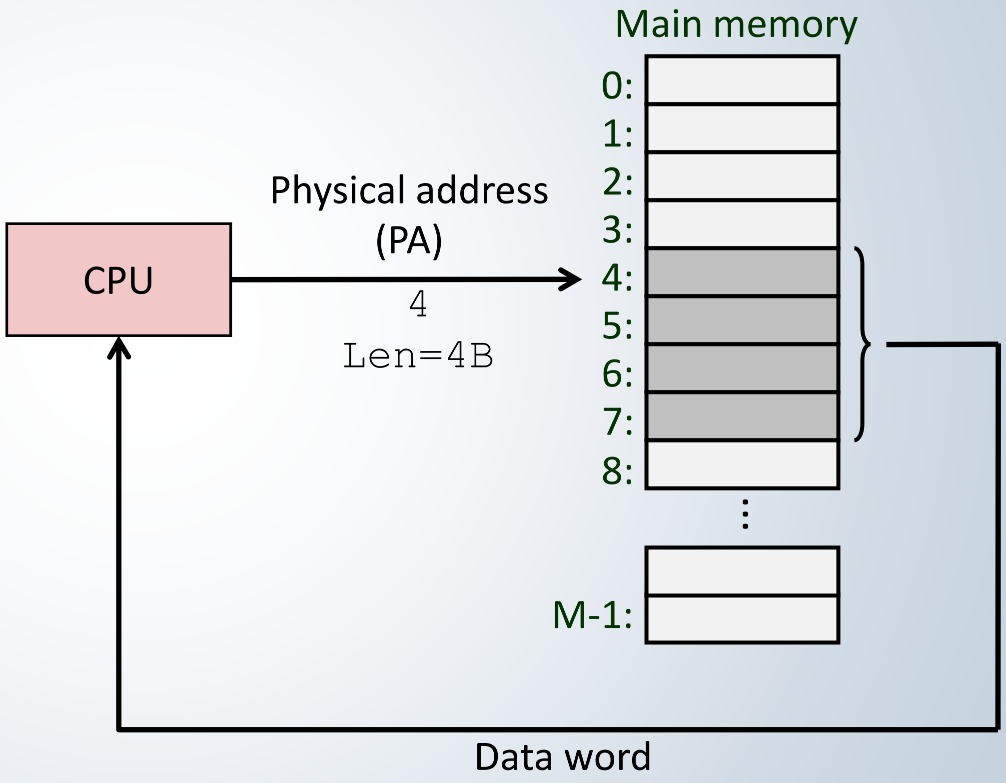
VANDERBILT UNIVERSITY

Today

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

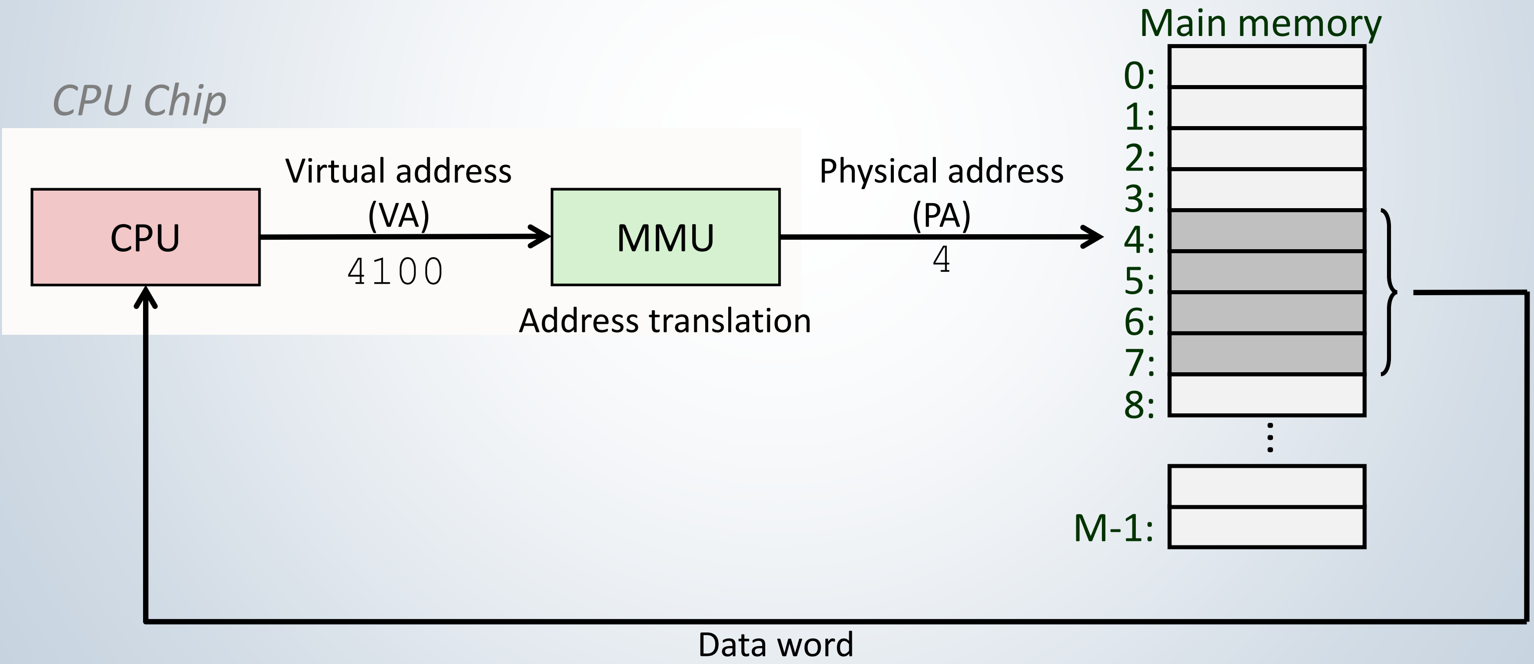
A System Using Physical Addressing

- 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 … }
- Physical address space: Set of M = 2^m physical addresses
  {0, 1, 2, 3, …, M-1}
  – If you have 16 GB of RAM, you need at least 34 bits of byte addressing (2^{34} = 16 GiB) to address all bytes.
- Virtual address space: Set of N = 2^n virtual addresses
  {0, 1, 2, 3, …, N-1}
  – If a machine uses **48-bit virtual addresses**, then the virtual address space size is 2^{48} bytes.

Tel (615) 343-7472 | Fax (615) 343-7440
1025 16th Avenue South Nashville, TN 37212
www.isis.vanderbilt.edu

Why Virtual Memory (VM)?

- Simplifies memory management
  - Each process gets the same linear address space
- Isolates address spaces
  - One process can't interfere with another's memory
- Uses main memory (RAM) efficiently
  - Use DRAM as a cache for parts of a virtual address space. A very large virtual memory is perceived by programmers when only a small physical memory is available
- Many other benefits (some discussed later)
  - Shared memory, memory deduplication, lazy allocation, etc.

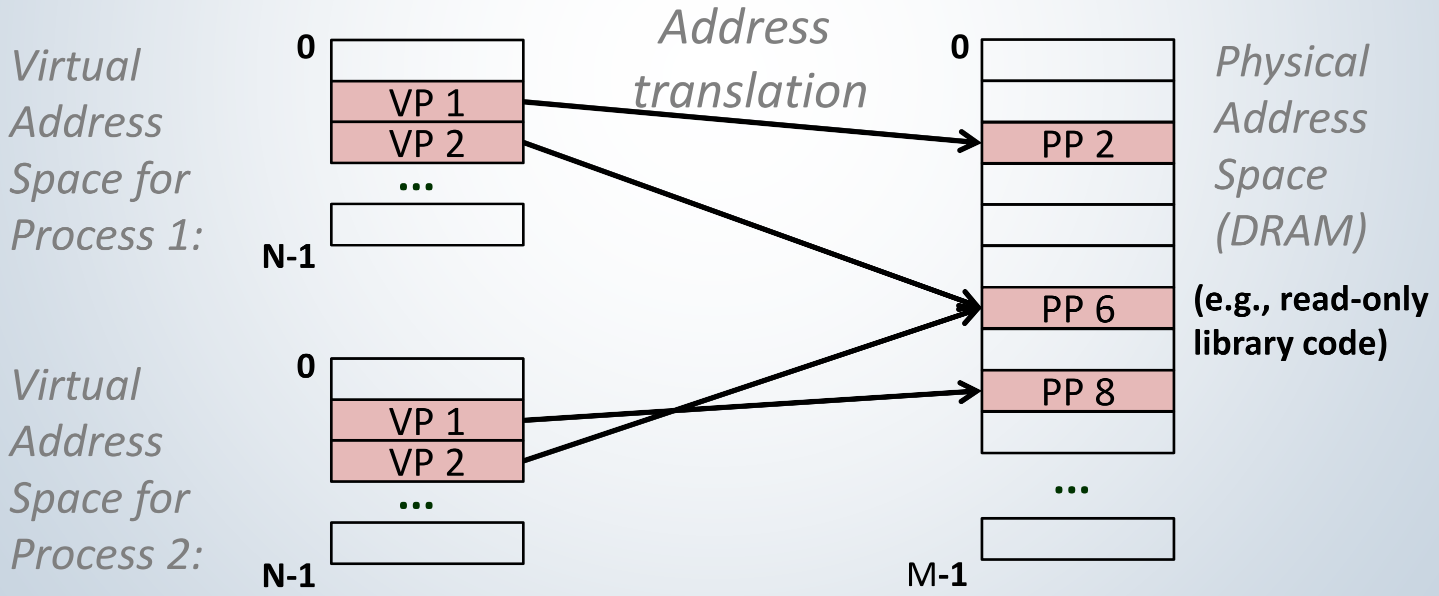
Today

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

ISIS
Tel (615) 343-7472 | Fax (615) 343-7440
1025 16th Avenue South Nashville, TN 37212
www.isis.vanderbilt.edu
VANDERBILT UNIVERSITY

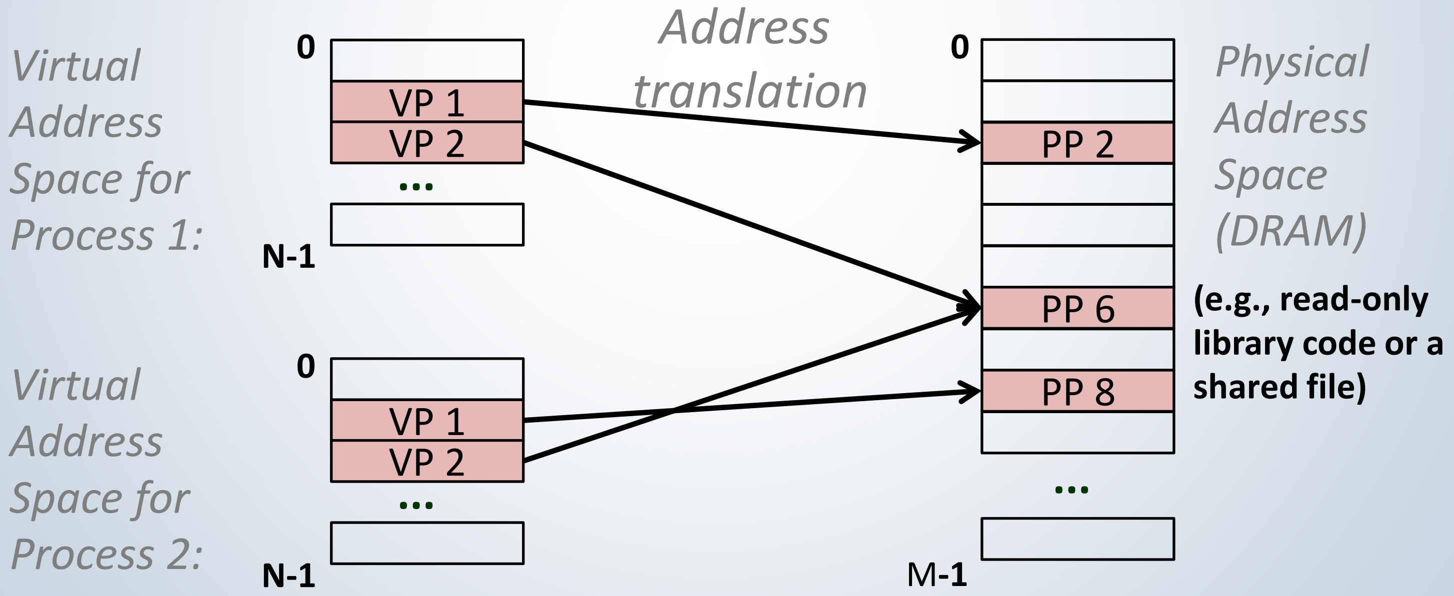
VM as a Tool for Memory Management

- **Key idea: each process has its own <u>virtual address space</u>**
  - It can view memory as a simple linear array
  - Mapping function scatters addresses through physical memory
  - Page: a fixed-size block of memory (e.g., 4KB)



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)

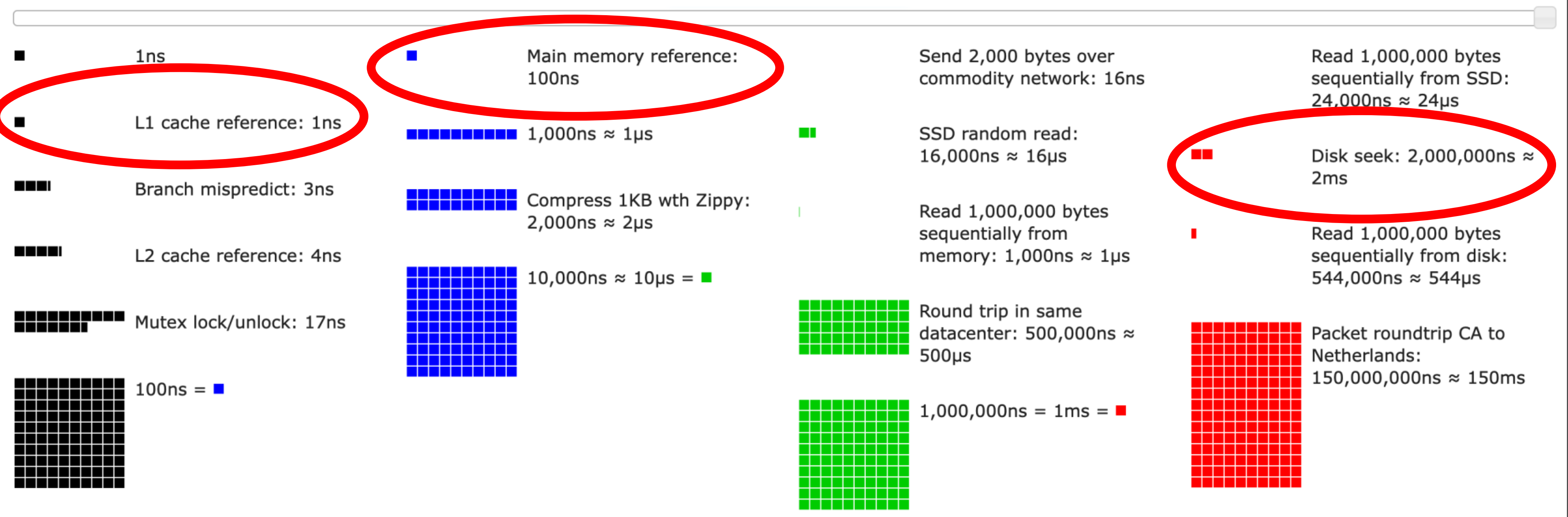


Today

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

Latencies in Different Levels of Memories

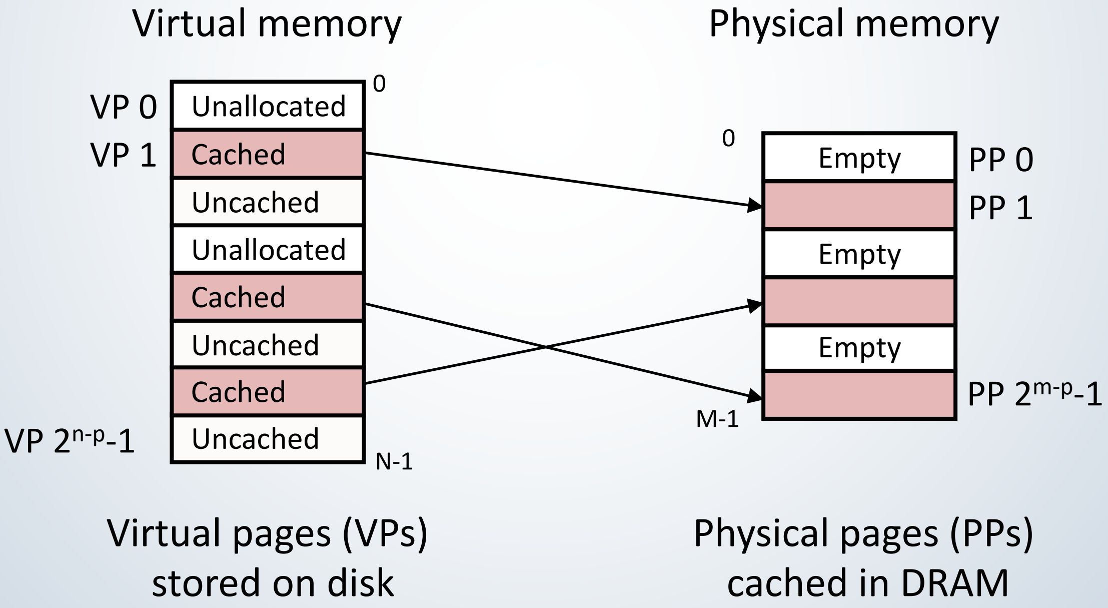
Latency Numbers Every Programmer Should Know



Source: https://colin-scott.github.io/personal_website/research/interactive_latency.html

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). Pages may (not) be contiguous in the physical memory



DRAM Cache Organization

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

- Consequences (page fault)
  - Large page (block) size: typically 4 KB (Huge pages are 2MB – 1GB.)
  - Only a subset of virtual pages are stored in the main memory (working set)
  - Highly sophisticated, expensive replacement algorithms
    - Too complicated and open-ended to be implemented in hardware

Tel (615) 343-7472 | Fax (615) 343-7440
1025 16th Avenue South Nashville, TN 37212
www.isis.vanderbilt.edu

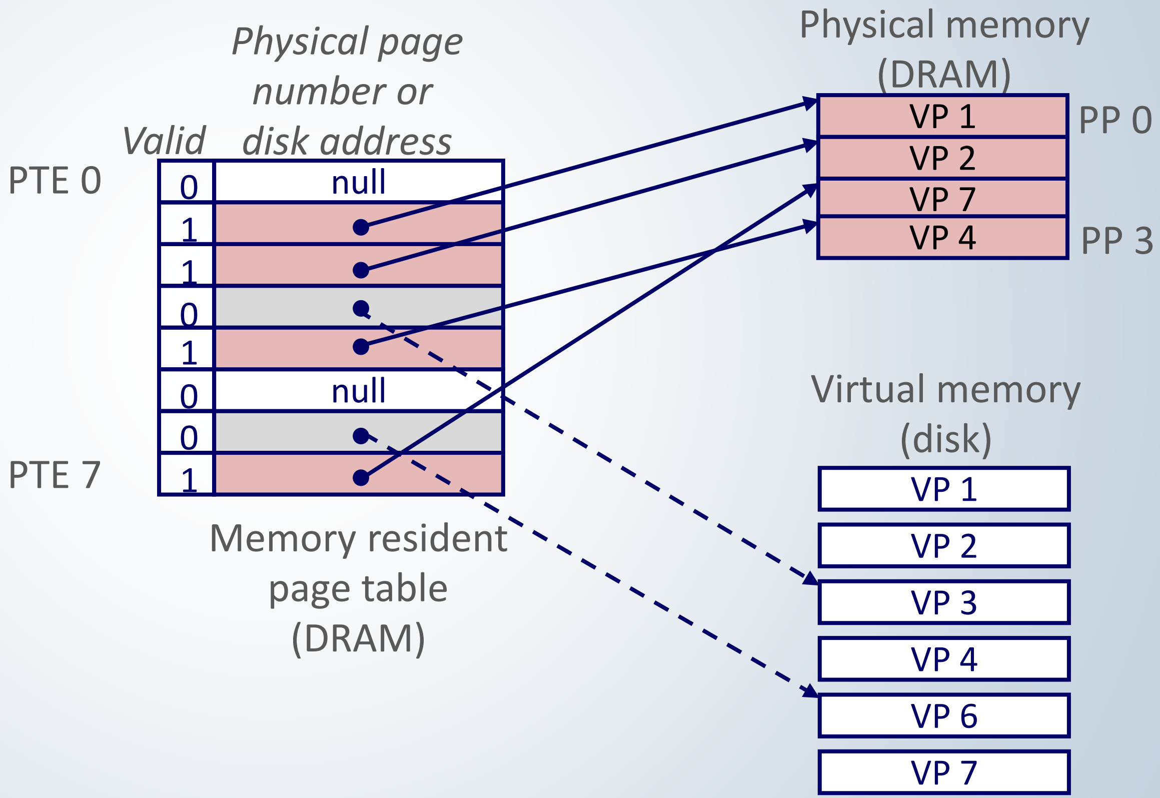
Internal Fragmentation

- Depending on a page size, a program size may not be a multiple of the number of pages. Thus, the last page is partially filled. This loss of usable memory is known as internal fragmentation

- The OS prevents out-of-range accesses by leaving pages outside the allocated address range unmapped; accesses to unmapped pages fault (segfault). Internal fragmentation remains because the last page may be only partially used.

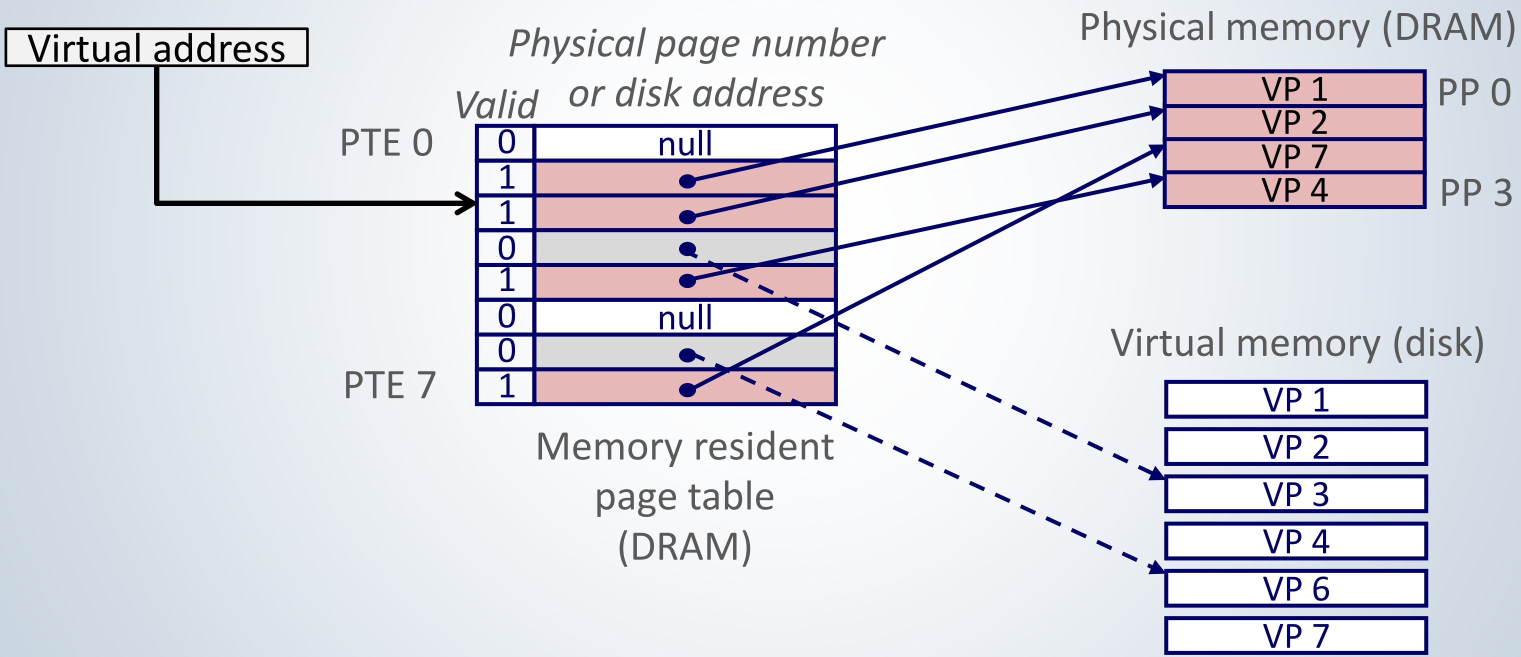
Enabling Data Structure: Page Table

- 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

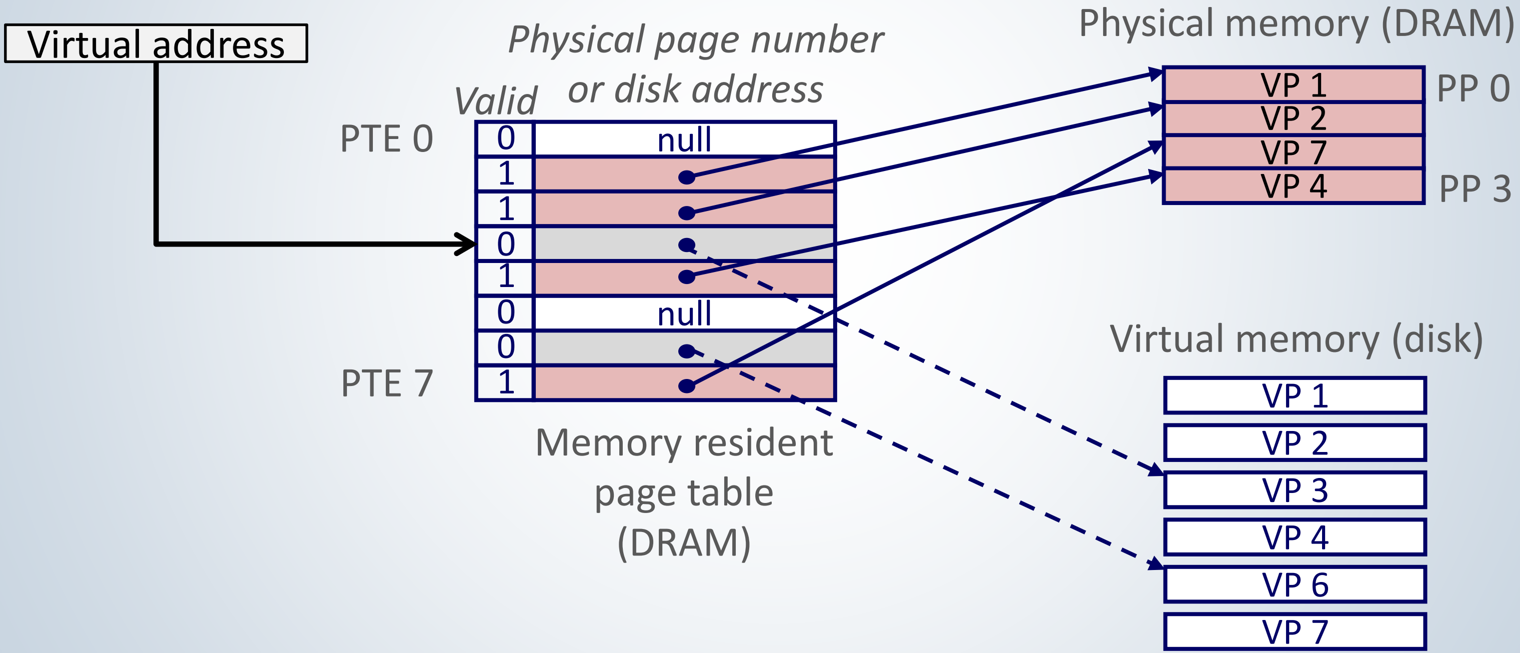
- *Page hit:* reference to VM word that is in physical memory (DRAM cache hit)



Tel (615) 343-7472 | Fax (615) 343-7440
1025 16th Avenue South Nashville, TN 37212
www.isis.vanderbilt.edu

Page Fault

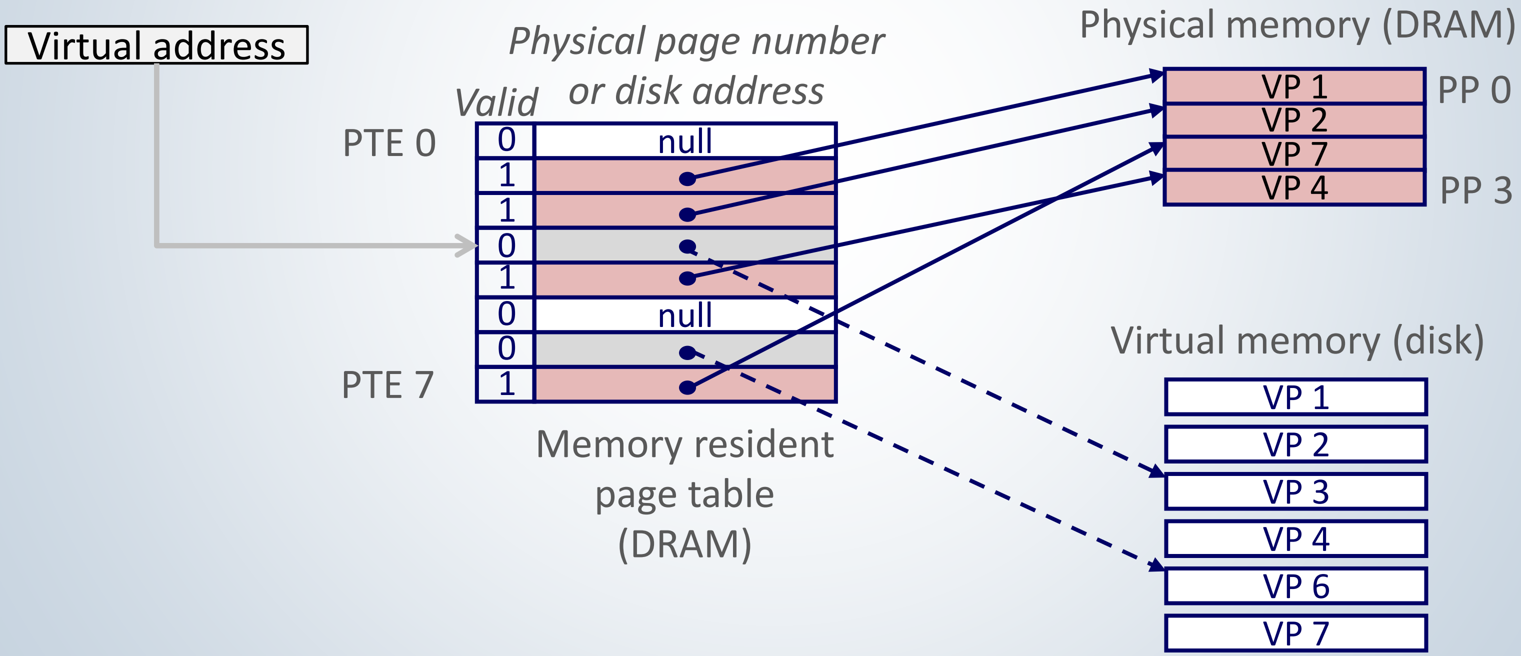
- *Page fault:* reference to VM word not in physical memory (DRAM cache miss). The VM word can be instruction or data



Tel (615) 343-7472 | Fax (615) 343-7440
1025 16th Avenue South Nashville, TN 37212
www.isis.vanderbilt.edu

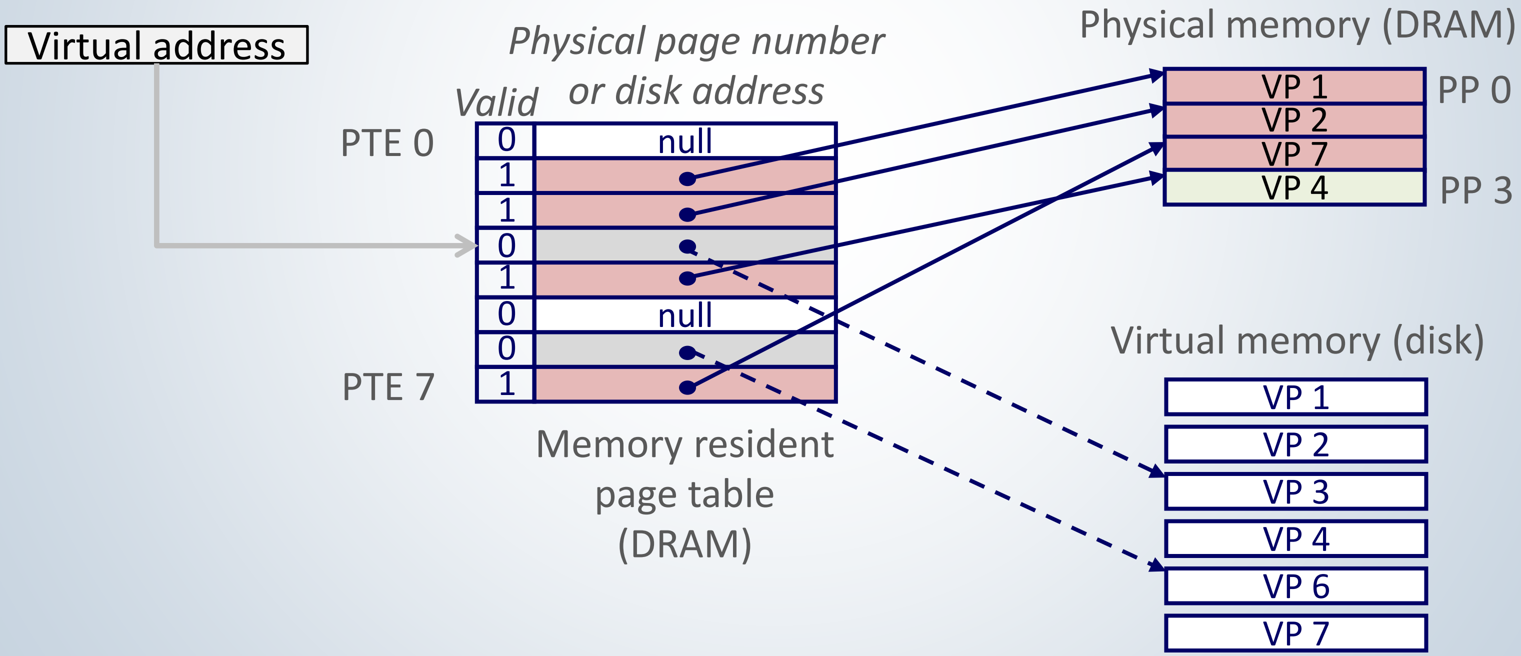
Handling Page Fault

- Page miss causes page fault (an exception)



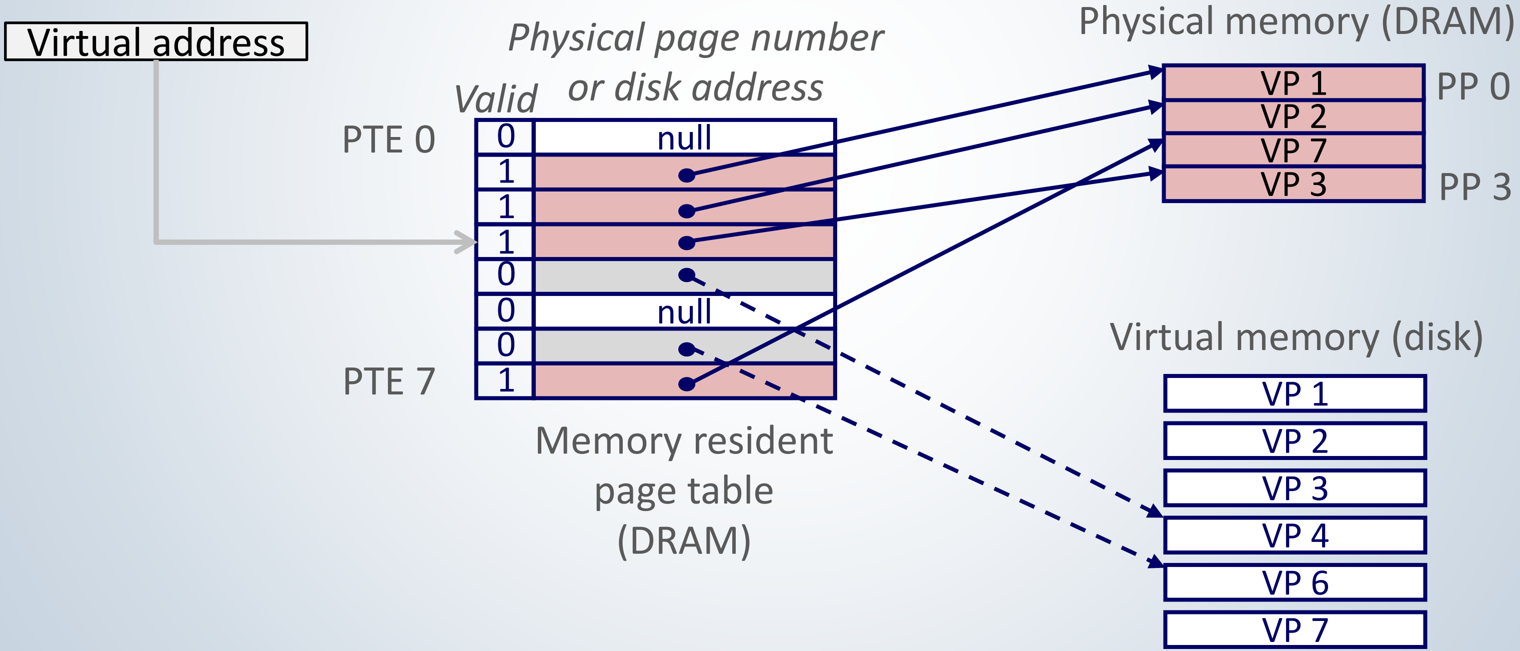
Handling Page Fault

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



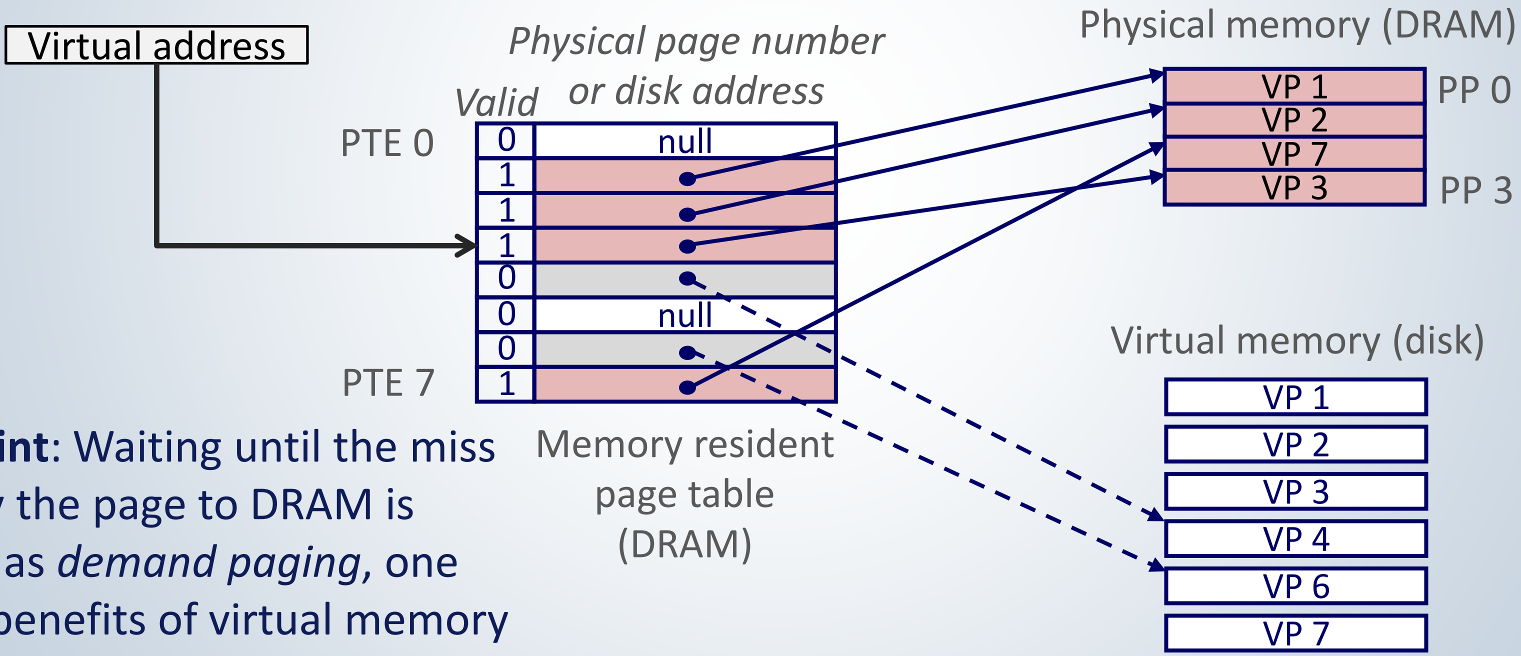
Handling Page Fault

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



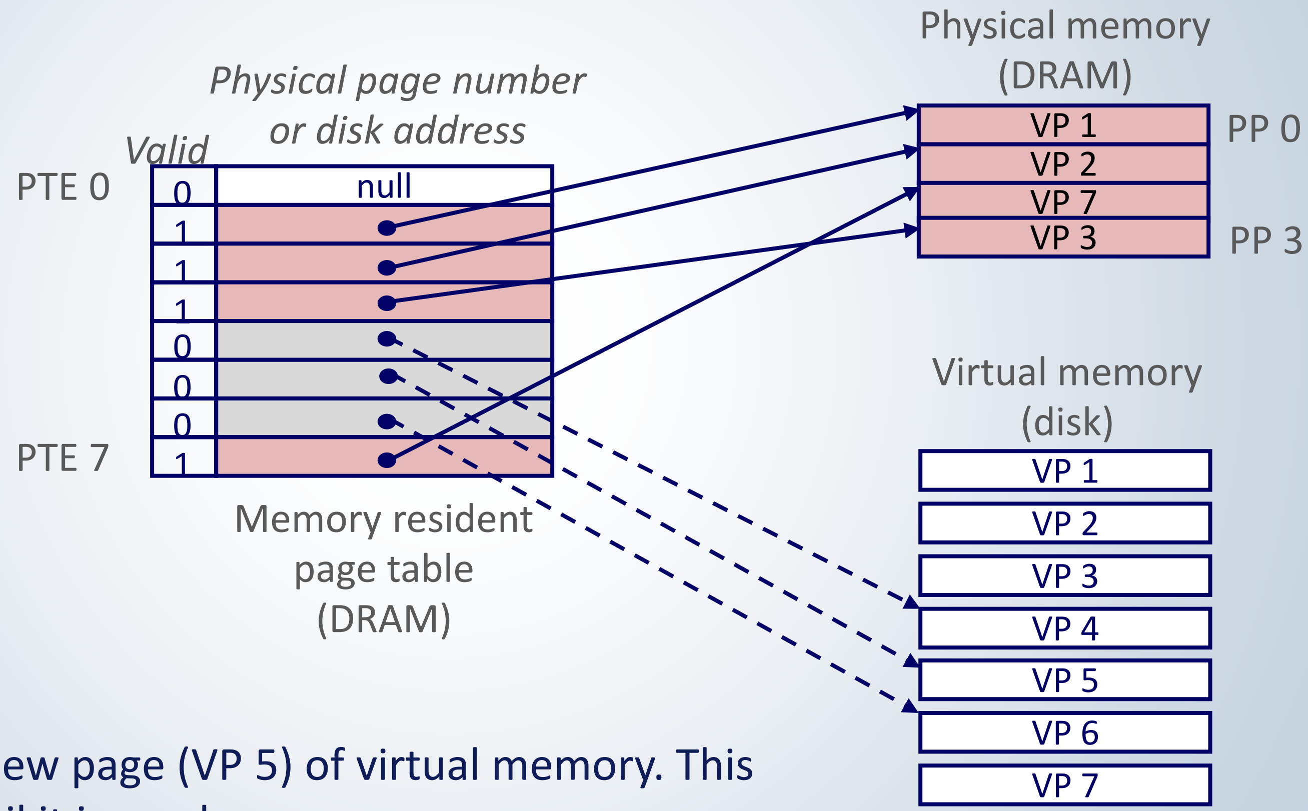
Handling Page Fault

- Page miss causes page fault (an exception)
- Page fault handler selects a victim to be evicted (here VP 4)
- Offending (faulting) instruction is restarted: page hit! If the page fault is caused by memory miss, the data operand is fetched from memory



Key point: Waiting until the miss to copy the page to DRAM is known as *demand paging*, one of the benefits of virtual memory

Allocating Pages



malloc() allocates a new page (VP 5) of virtual memory. This page stays in disk until it is used

Today

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

Process i:

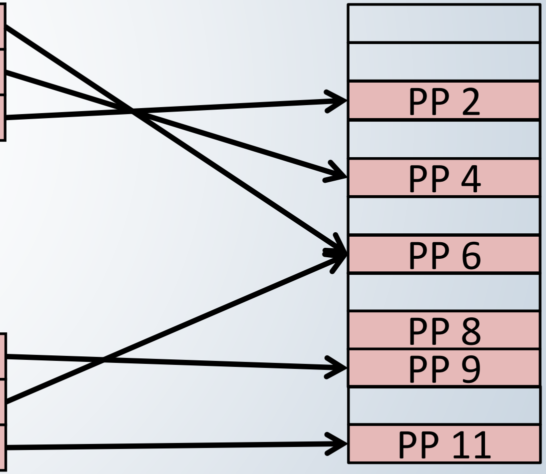
	SUP	READ	WRITE	EXEC	Address
VP 0:	No	Yes	No	Yes	PP 6
VP 1:	No	Yes	Yes	Yes	PP 4
VP 2:	Yes	Yes	Yes	No	PP 2

•
•
•

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

Physical Address Space



Tel (615) 343-7472 | Fax (615) 343-7440
1025 16th Avenue South Nashville, TN 37212
www.isis.vanderbilt.edu

Today

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

VM Address Translation

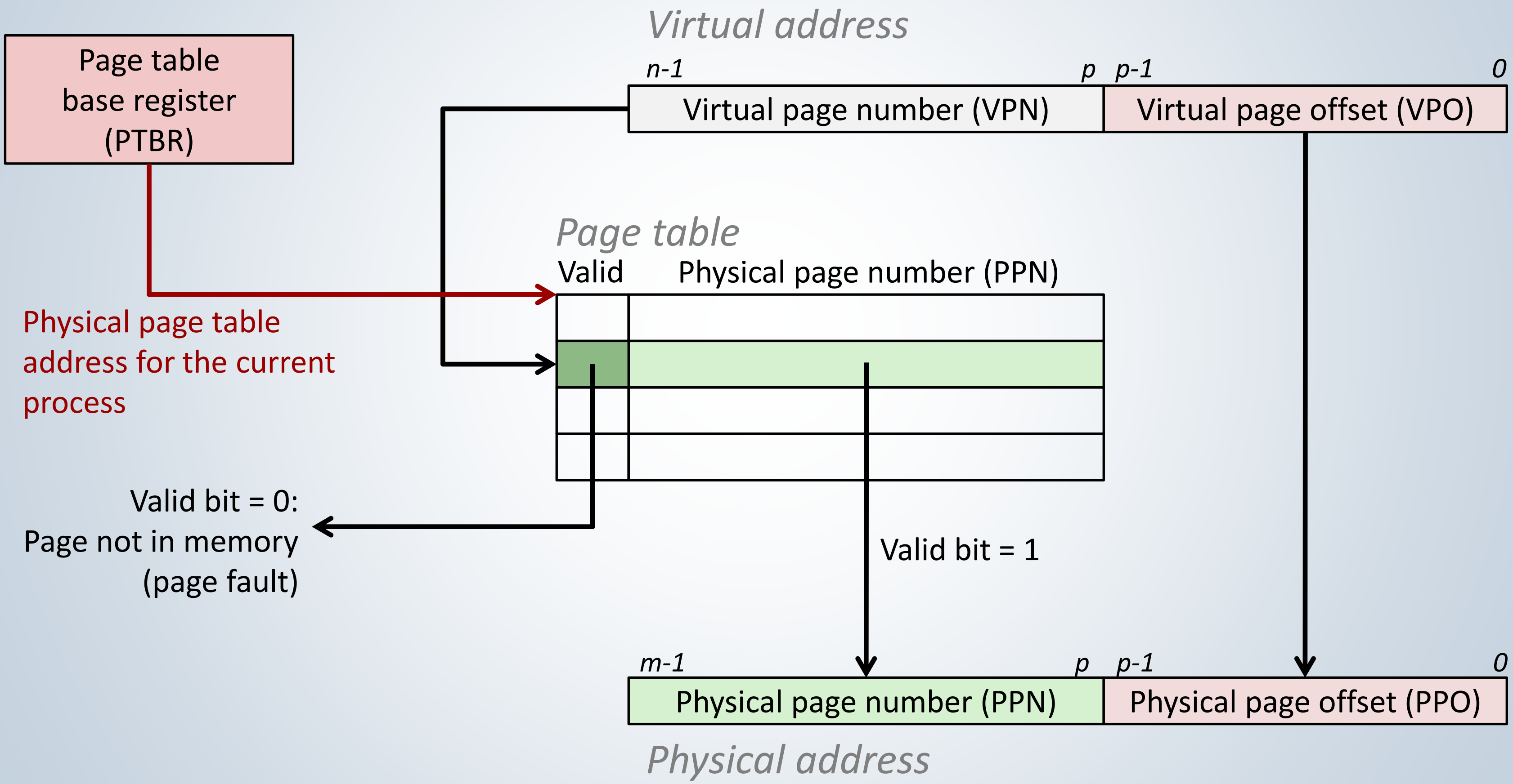
- Virtual Address Space
  - $V = \{0, 1, \ldots, N-1\}$
- Physical Address Space
  - $P = \{0, 1, \ldots, M-1\}$
- Address Translation
  - $\boldsymbol{MAP: V \rightarrow P \cup \{\varnothing\}}$
  - For virtual address ***a***:
    - ***MAP(a)* = *a'*** if data at virtual address ***a*** is at physical address ***a'*** in ***P***
    - ***MAP(a)* = $\varnothing$** if data at virtual address ***a*** is not in physical memory (page fault)

Tel (615) 343-7472 | Fax (615) 343-7440
1025 16th Avenue South Nashville, TN 37212
www.isis.vanderbilt.edu

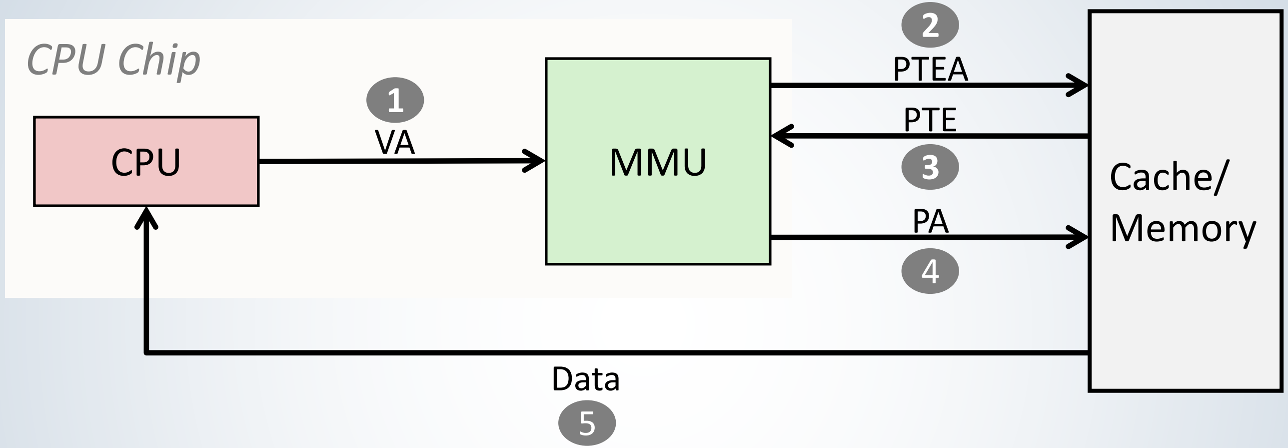
Summary of Address Translation Symbols

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

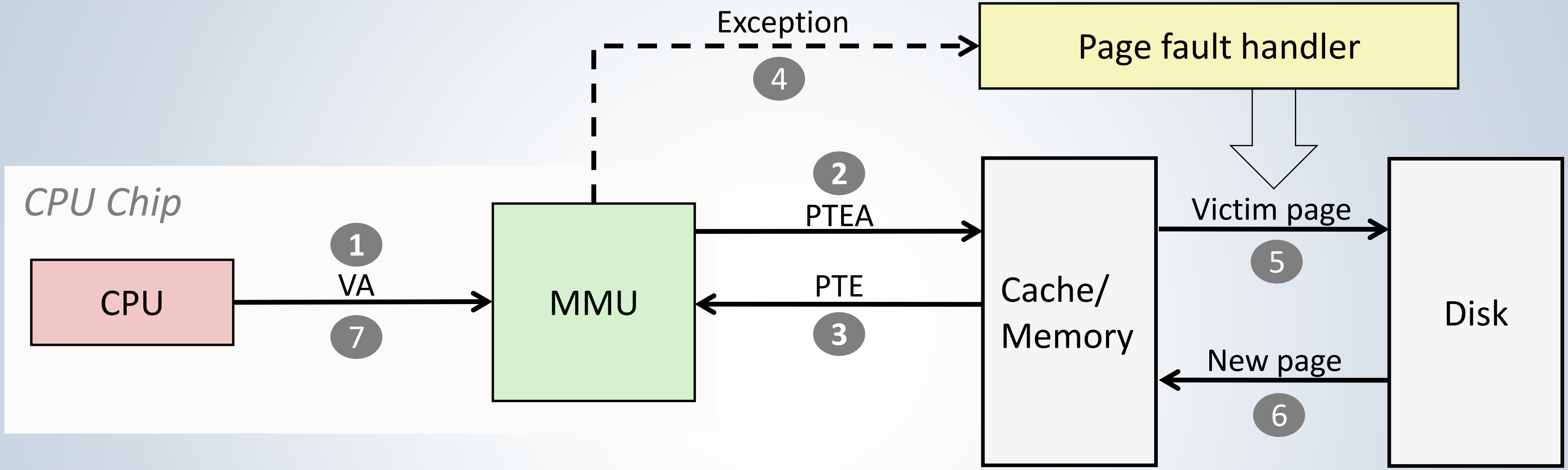


Address Translation: Page Hit



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

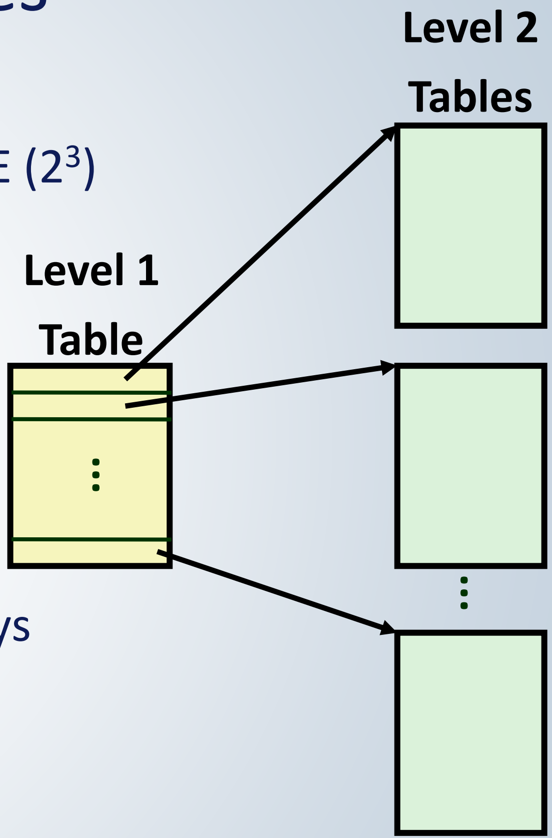
Address Translation: Page Fault



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

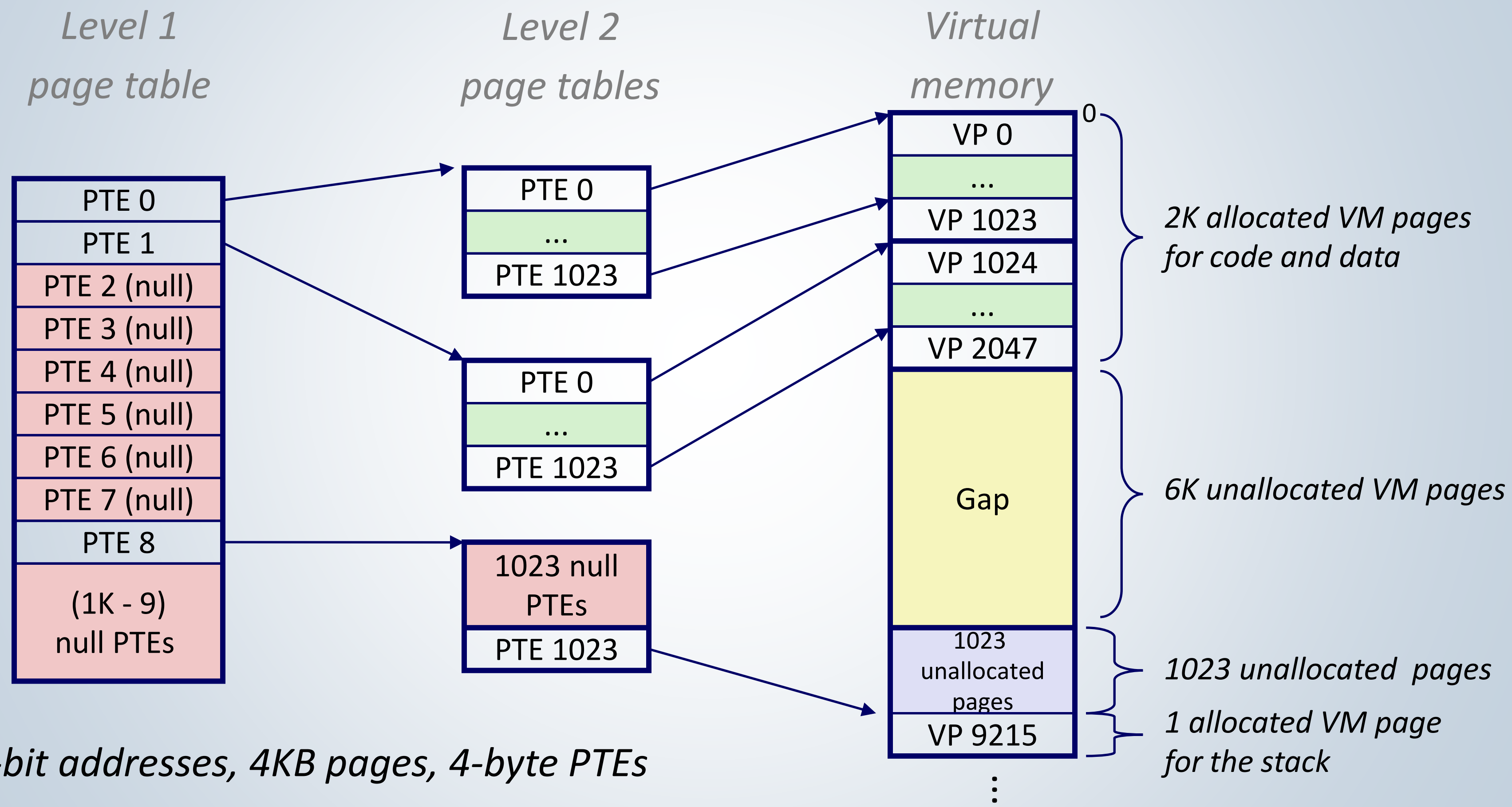
Multi-Level Page Tables

- Suppose:
  - 4KB (2^{12}) page size, 48-bit address space, 8-byte PTE (2^3)
- Problem:
  - Would need a 512 GB page table!
    - $2^{48} * 2^{-12} * 2^3 = 2^{39}$ bytes
- Common solution: Multi-level page table
- Example: 2-level page table
  - Level 1 table: each PTE points to a page table (always memory resident)
  - Level 2 table: each PTE points to a data page (paged in and out like any other data)



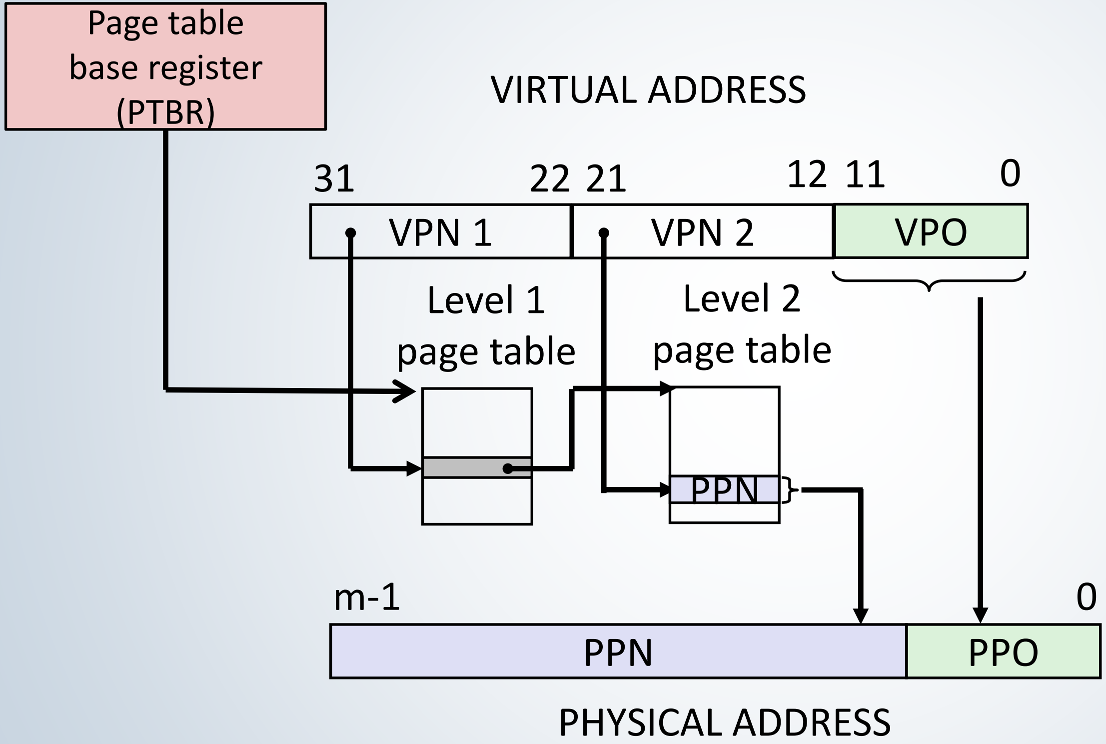
Tel (615) 343-7472 | Fax (615) 343-7440
1025 16th Avenue South Nashville, TN 37212
www.isis.vanderbilt.edu
VANDERBILT UNIVERSITY

Two-Level Page-Table Hierarchy



32-bit addresses, 4KB pages, 4-byte PTEs

Translating with a 2-Level Page Table



Tel (615) 343-7472 | Fax (615) 343-7440
1025 16th Avenue South Nashville, TN 37212
www.isis.vanderbilt.edu

Multi-level page table (example)

- Consider a 3-level page table. A page table has 64 entries. If the page size is 4KB, how much is the size of the virtual address space?

Tel (615) 343-7472 | Fax (615) 343-7440
1025 16th Avenue South Nashville, TN 37212
www.isis.vanderbilt.edu

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 point to check permissions