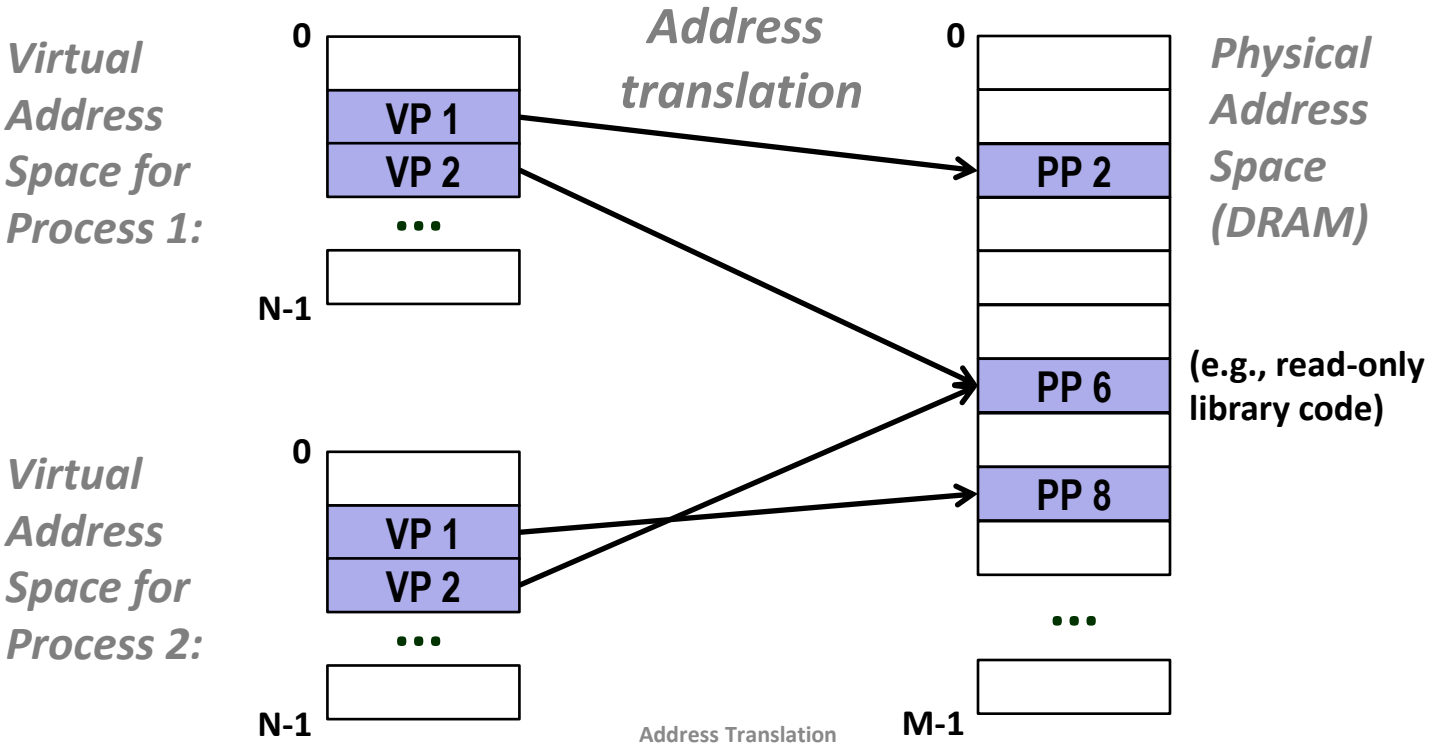
Section 9: Virtual Memory (VM)

- Overview and motivation
- Indirection
- VM as a tool for caching
- Memory management/protection and address translation
- Virtual memory example

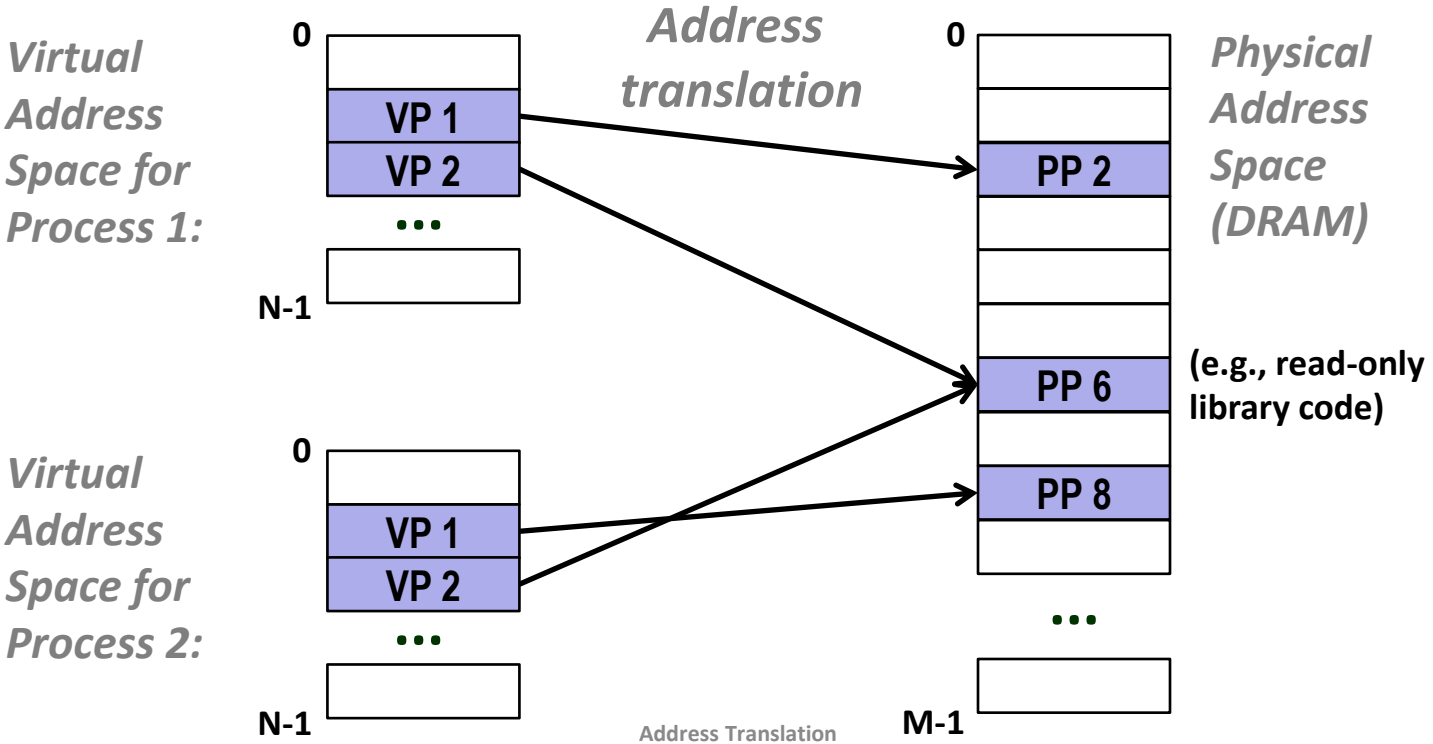
VM for Managing Multiple Processes

- **Key abstraction: each process has its own virtual address space**
  - It can view memory as *a simple linear array*
- **With virtual memory, this simple linear virtual address space need not be contiguous in physical memory**
  - Process needs to store data in another VP? Just map it to *any* PP!



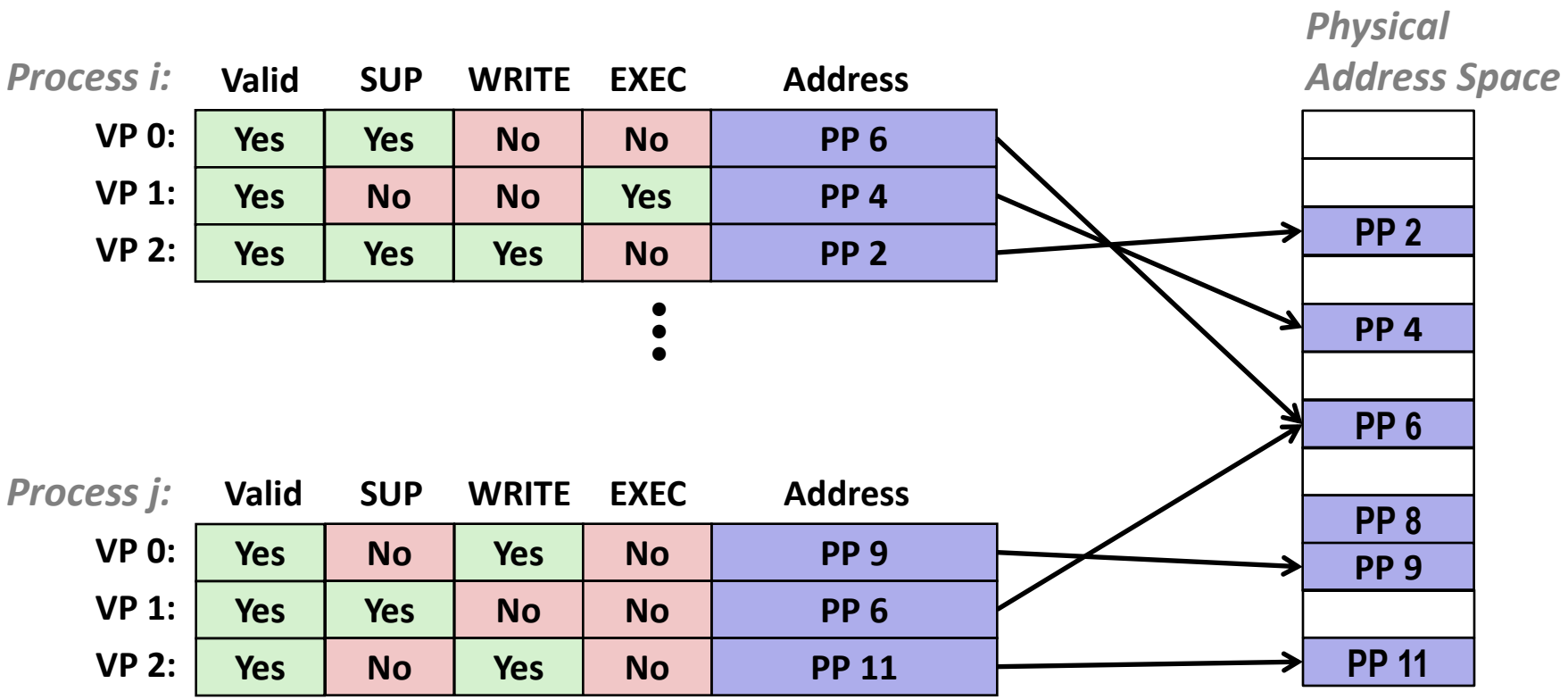
VM for Protection and Sharing

- **The mapping of VPs to PPs provides a simple mechanism for *protecting* memory and for *sharing* memory btw. processes**
  - Sharing: just map virtual pages in separate address spaces to the same physical page (here: PP 6)
  - Protection: process simply can't access physical pages it doesn't have a mapping for (here: Process 2 can't access PP 2)

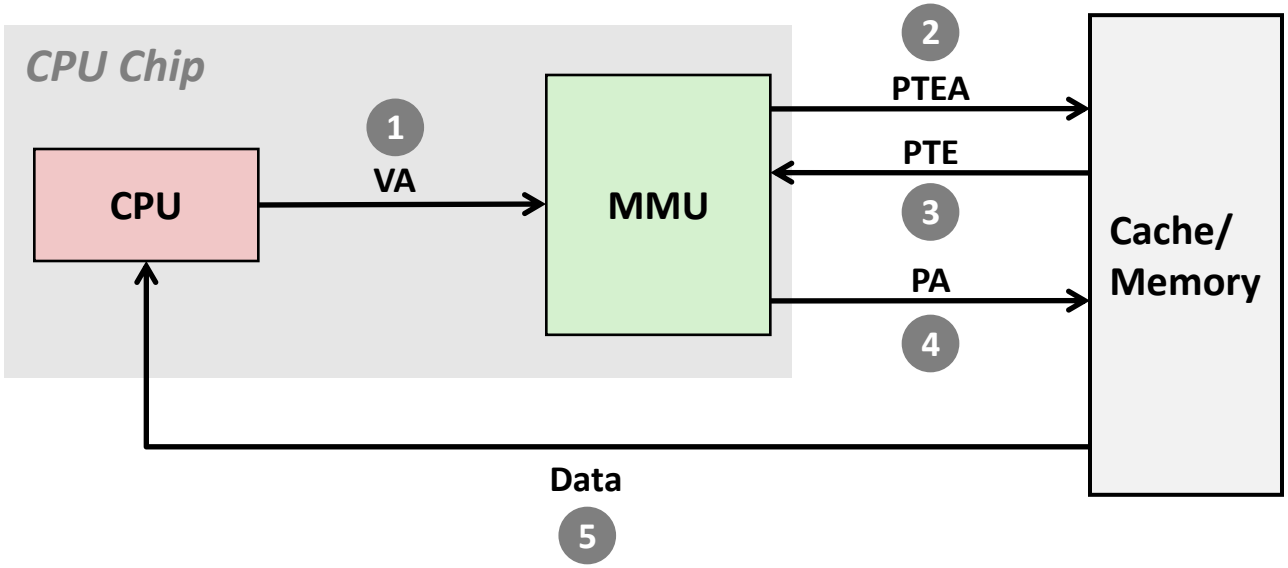


Memory Protection Within a Single Process

- Extend PTEs with permission bits
- MMU checks these permission bits on every memory access
  - If violated, raises exception and OS sends SIGSEGV signal to process



Address Translation: Page Hit



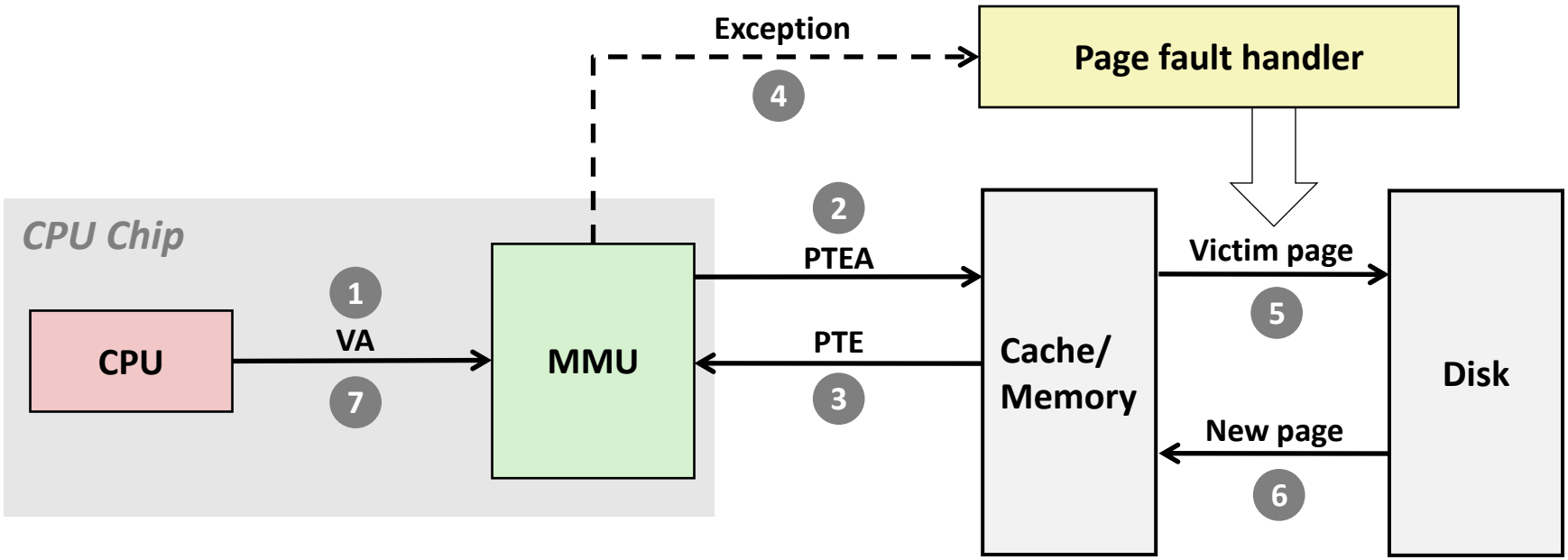
1) Processor sends virtual address to MMU (*memory management unit*)

2-3) MMU fetches PTE from page table in cache/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 cache/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

Hmm… Translation Sounds Slow!

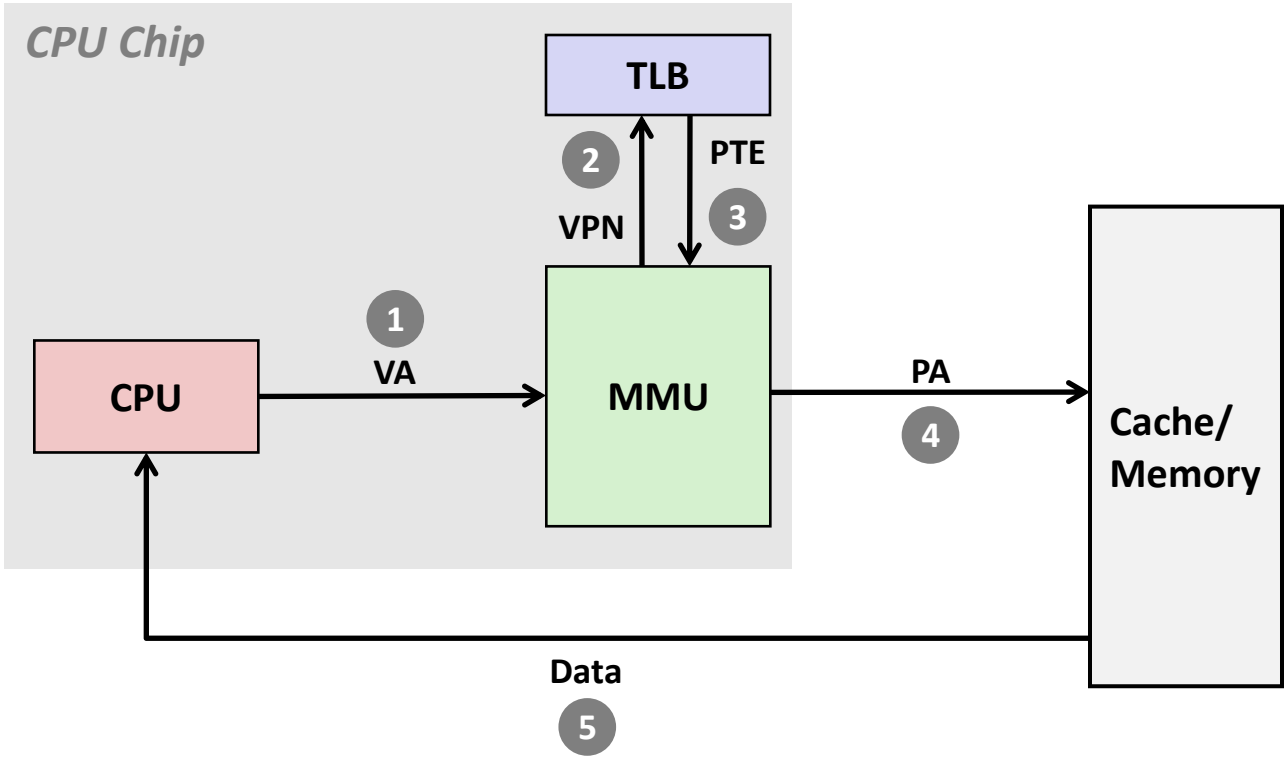
- **The MMU accesses memory *twice*: once to first get the PTE for translation, and then again for the actual memory request from the CPU**
  - The PTEs *may* be cached in L1 like any other memory word
    - But they may be evicted by other data references
    - And a hit in the L1 cache still requires 1-3 cycles

- *What can we do to make this faster?*

Speeding up Translation with a TLB

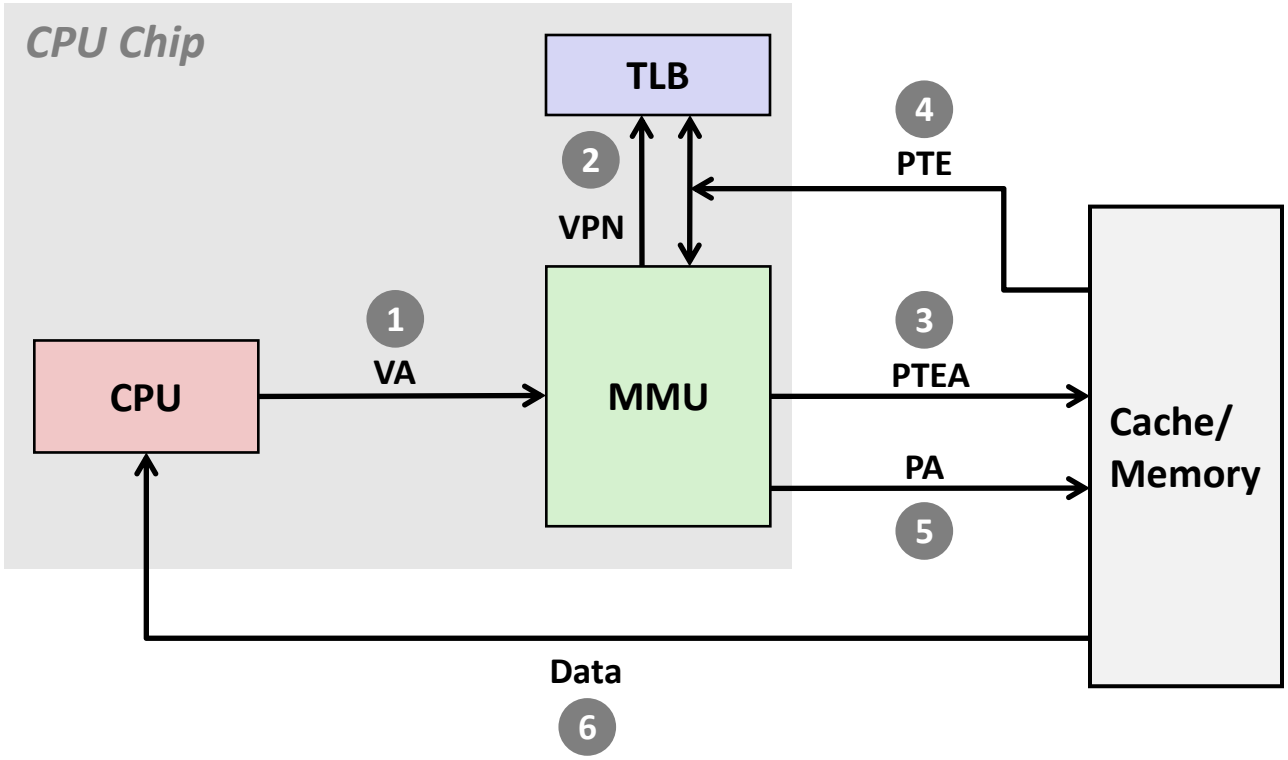
- **Solution: add another cache!**
- ***Translation Lookaside Buffer*** **(TLB):**
  - Small hardware cache in MMU
  - Maps virtual page numbers to physical page numbers
  - Contains complete *page table entries* for small number of pages
    - Modern Intel processors: 128 or 256 entries in TLB

TLB Hit



A TLB hit eliminates a memory access

TLB Miss



A TLB miss incurs an additional memory access (the PTE)

Fortunately, TLB misses are rare