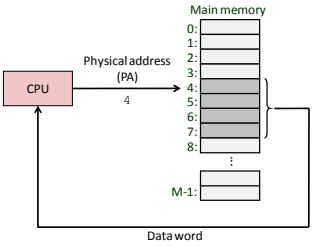
Lecture 24

# Virtual Memory

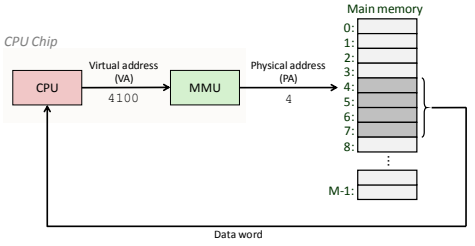
CPSC 275
Introduction to Computer Systems

## 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 computer systems.

## Address Spaces

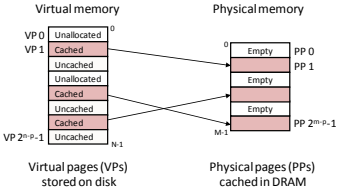
- Virtual address space: Set of $N = 2^n$ virtual addresses
  {0, 1, 2, 3, …, $N$-1}
- Physical address space: Set of $M = 2^m$ physical addresses
  {0, 1, 2, 3, …, $M$-1}
- Every byte in main memory:
  - one physical address
  - one (or more) virtual addresses

## Why Virtual Memory (VM)?

- Uses main memory efficiently
  - Use DRAM as a cache for the 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

## VM as a Tool for Caching

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

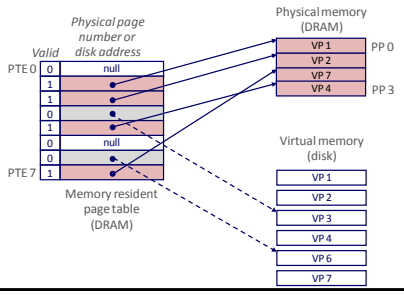


## DRAM Cache Organization

- 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 4-8 KB
  - Fully associative:
    - Any VP can be placed in any PP
    - Requires a "large" mapping function
  - Highly sophisticated, expensive replacement algorithms
  - *Write-back* rather than *write-through*

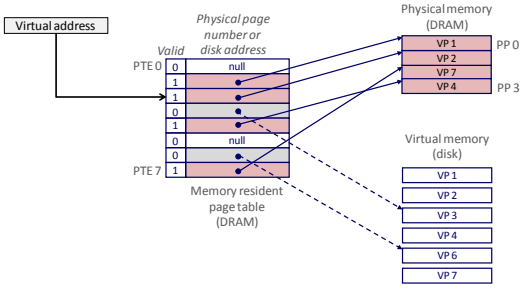
## Page Tables

- 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



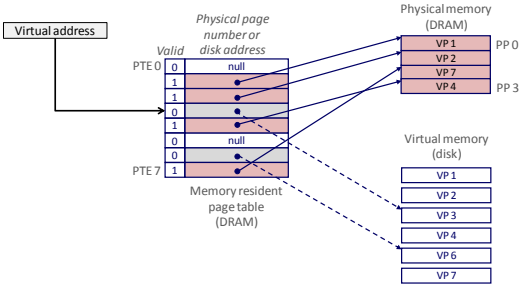

## Page Hit

- *Page hit:* reference to VM word that is in physical memory



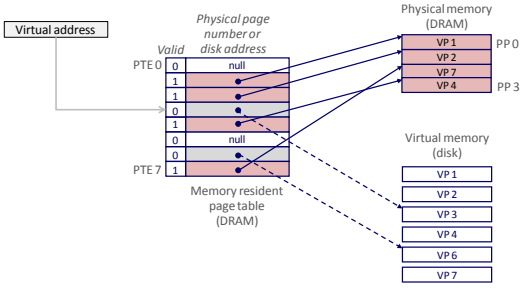

## Page Fault

- *Page fault:* reference to VM word that is not in physical memory



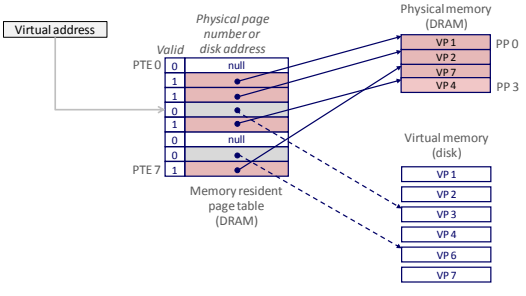

## Handling Page Fault

- Page miss causes page fault



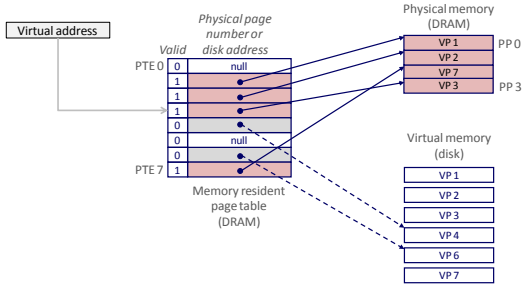

## Handling Page Fault

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



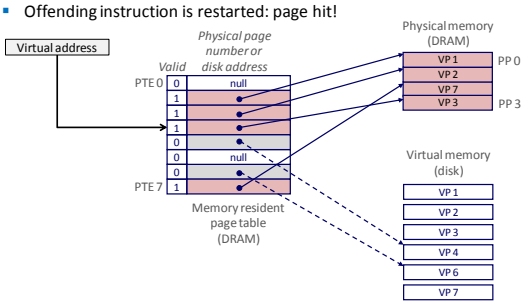
## Handling Page Fault

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




## Handling Page Fault

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




## Locality to the Rescue Again!

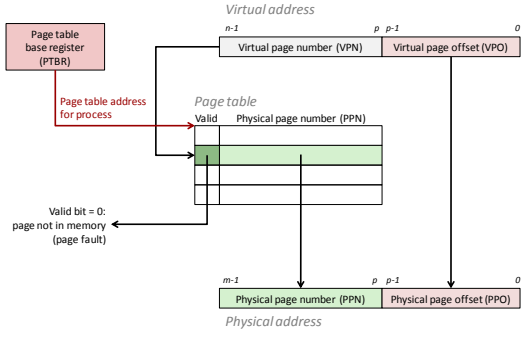
- Virtual memory works because of locality
- At any point in time, programs tend to access a set of active virtual pages called the *working set*
  - Programs with better temporal locality will have smaller working sets
- If (working set size < main memory size)
  - Good performance for one process after compulsory misses
- If ( SUM(working set sizes) > main memory size )
  - *Thrashing:* Performance meltdownwhere pages are swapped (copied) in and out continuously

## VM Address Translation

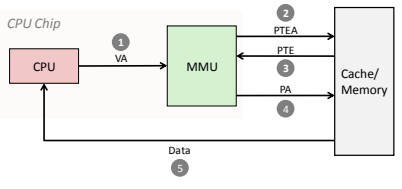
- Virtual Address Space
  - $V = \{0, 1, \ldots, N-1\}$
- Physical Address Space
  - $P = \{0, 1, \ldots, M-1\}$
- Address Translation
  - $MAP: V \rightarrow P \cup \{\varnothing\}$

  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

## Address Translation With a Page Table




## Address Translation: Page Hit



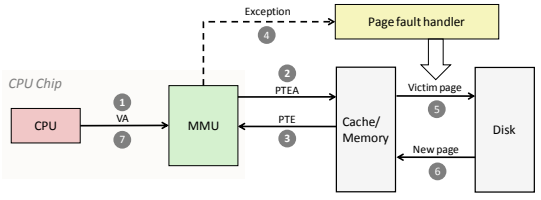

1) Processor sends virtual address to MMU

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

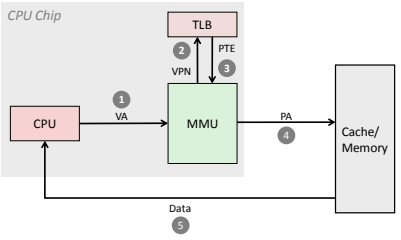



1) Processor sends virtual address to MMU
2-3) MMU fetches PTE from page table in memory
4) Valid bit is zero, so MMU triggers page fault exception
5) Handler identifies victim (and, if *dirty*, pages it out to disk)
6) Handler pages in new page and updates PTE in memory
7) Handler returns to original process, restarting faulting instruction

# Speeding up Translation with a TLB

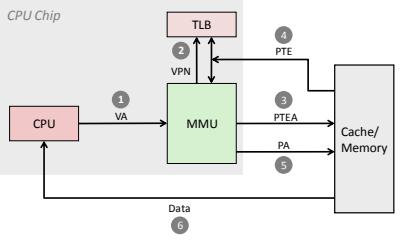
- *Translation Lookaside Buffer* (TLB)
  - Small hardware cache in MMU
  - Maps virtual page numbers to physical page numbers
  - Contains complete page table entries for small number of pages

# TLB Hit




A TLB hit eliminates a memory access

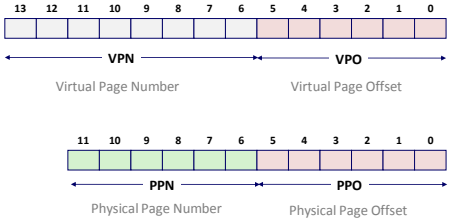
# TLB Miss




A TLB miss incurs an additional memory access

# Simple Memory System Example

- Addressing
  - 14-bit virtual addresses
  - 12-bit physical address
  - Page size = 64 bytes




# Simple Memory System Page Table

How many entries are in the page table?

Only show first 16 entries (out of 256)

| VPN | PPN | Valid |
|---|---|---|
| 00 | 28 | 1 |
| 01 | – | 0 |
| 02 | 33 | 1 |
| 03 | 02 | 1 |
| 04 | – | 0 |
| 05 | 16 | 1 |
| 06 | – | 0 |
| 07 | – | 0 |

| VPN | PPN | Valid |
|---|---|---|
| 08 | 13 | 1 |
| 09 | 17 | 1 |
| 0A | 09 | 1 |
| 0B | – | 0 |
| 0C | – | 0 |
| 0D | 2D | 1 |
| 0E | 11 | 1 |
| 0F | 0D | 1 |

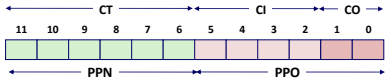
## Simple Memory System TLB

- 16 entries
- 4-way associative


| Set | Tag | PPN | Valid | Tag | PPN | Valid | Tag | PPN | Valid | Tag | PPN | Valid |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 03 | – | 0 | 09 | 0D | 1 | 00 | – | 0 | 07 | 02 | 1 |
| 1 | 03 | 2D | 1 | 02 | – | 0 | 04 | – | 0 | 0A | – | 0 |
| 2 | 02 | – | 0 | 08 | – | 0 | 06 | – | 0 | 03 | – | 0 |
| 3 | 07 | – | 0 | 03 | 0D | 1 | 0A | 34 | 1 | 02 | – | 0 |

## Simple Memory System Cache

- 16 lines, 4-byte block size
- Physically addressed
- Direct mapped

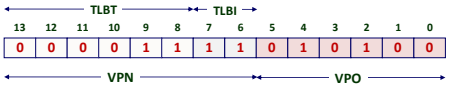



| Idx | Tag | Valid | B0 | B1 | B2 | B3 |
|---|---|---|---|---|---|---|
| 0 | 19 | 1 | 99 | 11 | 23 | 11 |
| 1 | 15 | 0 | – | – | – | – |
| 2 | 1B | 1 | 00 | 02 | 04 | 08 |
| 3 | 36 | 0 | – | – | – | – |
| 4 | 32 | 1 | 43 | 6D | 8F | 09 |
| 5 | 0D | 1 | 36 | 72 | F0 | 1D |
| 6 | 31 | 0 | – | – | – | – |
| 7 | 16 | 1 | 11 | C2 | DF | 03 |

| Idx | Tag | Valid | B0 | B1 | B2 | B3 |
|---|---|---|---|---|---|---|
| 8 | 24 | 1 | 3A | 00 | 51 | 89 |
| 9 | 2D | 0 | – | – | – | – |
| A | 2D | 1 | 93 | 15 | DA | 3B |
| B | 0B | 0 | – | – | – | – |
| C | 12 | 0 | – | – | – | – |
| D | 16 | 1 | 04 | 96 | 34 | 15 |
| E | 13 | 1 | 83 | 77 | 1B | D3 |
| F | 14 | 0 | – | – | – | – |

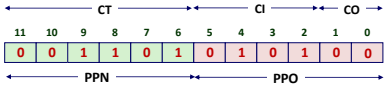
## Address Translation Example

Virtual Address: `0x03D4`




VPN 0x0F TLBI 0x3 TLBT 0x03 TLB Hit? Y Page Fault? N PPN: 0x0D

Physical Address




CO 0 CI 0x5 CT 0x0D Hit? Y Byte: 0x36

## Practice Problems

- Read CSaPP Sec. 9.8-9.9 and try Practice Problems 9.1, 9.2, 9.3, and 9.4.