

Lecture 22: Virtual Memory

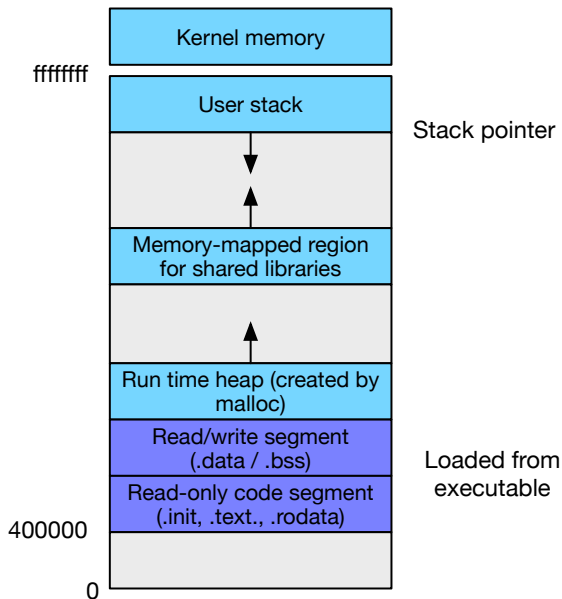
Philipp Koehn

March 28, 2022

601.229 Computer Systems Fundamentals



Recall: Process Address Space



Virtual Memory

- ▶ Abstraction of physical memory
- ▶ Purpose
 - ▶ appearance of more available memory than physically exists (DRAM)
 - ▶ handles disk caching / loading
 - ▶ insulates memory of each process

Virtual Memory

- ▶ Abstraction of physical memory
- ▶ Purpose
 - ▶ appearance of more available memory than physically exists (DRAM)
 - ▶ handles disk caching / loading
 - ▶ insulates memory of each process
- ▶ Page table: maps from virtual address to physical addresses

Virtual Memory

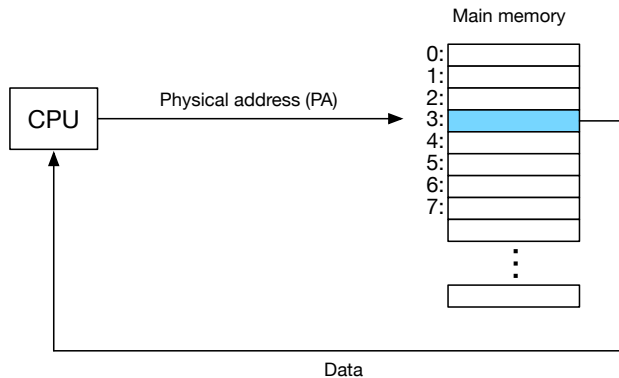
- ▶ Abstraction of physical memory
- ▶ Purpose
 - ▶ appearance of more available memory than physically exists (DRAM)
 - ▶ handles disk caching / loading
 - ▶ insulates memory of each process
- ▶ Page table: maps from virtual address to physical addresses
- ▶ Memory management unit (MMU):
hardware implementation of address translation

Warning

- ▶ This is going to get very complex
- ▶ Closely tied with multi-tasking (multiple processes)
- ▶ Partly managed by hardware, partly managed by software

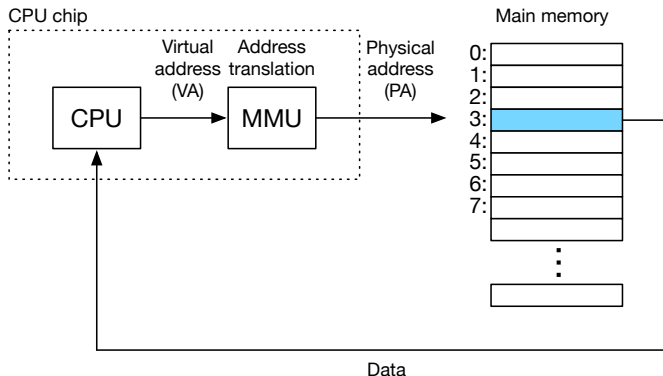
Virtual addressing

Physical Addressing



- So far, assumed CPU addresses physical memory

Virtual Addressing



- ▶ Memory management unit (MMU): maps virtual to physical addresses

Address Space

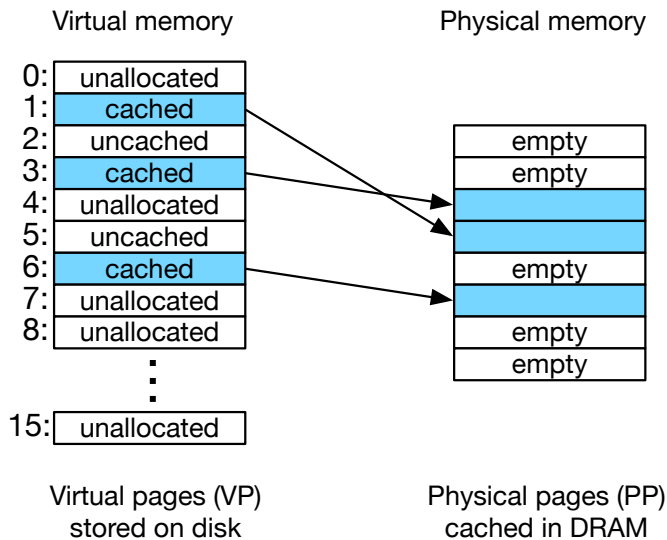
- ▶ Virtual memory size: $N = 2^n$ bytes, e.g., 256TB
- ▶ Physical memory size: $M = 2^m$ bytes, e.g., 16GB
- ▶ Page (block of memory): $P = 2^p$ bytes, e.g., 4KB
- ▶ A virtual address can be encoded in n bits

Caching

Caching... Again?

- ▶ Yes, we already discussed caching, but for on-chip cache of DRAM memory
- ▶ Now
 - ▶ caching between RAM and disk
 - ▶ driven by a large virtual memory address space
 - ▶ to avoid unnecessary and duplicate loading
- ▶ Jargon
 - ▶ previously “block”, now “page”
 - ▶ now: “swapping” or “paging”

Mapping



State of Virtual Memory Page

- ▶ Cached
 - ▶ allocated page
 - ▶ stored in physical memory

State of Virtual Memory Page

- ▶ Cached
 - ▶ allocated page
 - ▶ stored in physical memory
- ▶ Uncached
 - ▶ allocated page
 - ▶ not in physical memory

State of Virtual Memory Page

- ▶ Cached
 - ▶ allocated page
 - ▶ stored in physical memory
- ▶ Uncached
 - ▶ allocated page
 - ▶ not in physical memory
- ▶ Unallocated
 - ▶ not used by virtual memory system so far

Page Table

- ▶ Array of page table entries (PTE)

Page Table

- ▶ Array of page table entries (PTE)
(actually, a tree where the leaves store the page table entries)

Page Table

- ▶ Array of page table entries (PTE)
(actually, a tree where the leaves store the page table entries)
- ▶ Each PTE maps a virtual page to a physical page

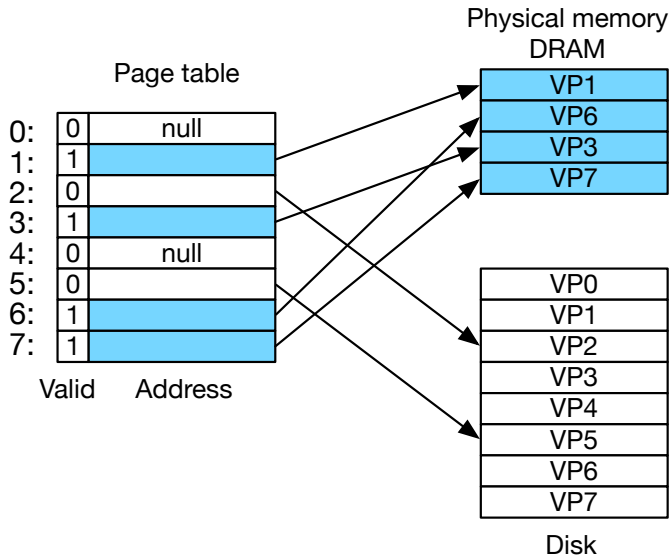
Page Table

- ▶ Array of page table entries (PTE)
(actually, a tree where the leaves store the page table entries)
- ▶ Each PTE maps a virtual page to a physical page
- ▶ Valid bit
 - ▶ set if PTE currently maps to physical address (cached)
 - ▶ not set otherwise (uncached or unallocated)

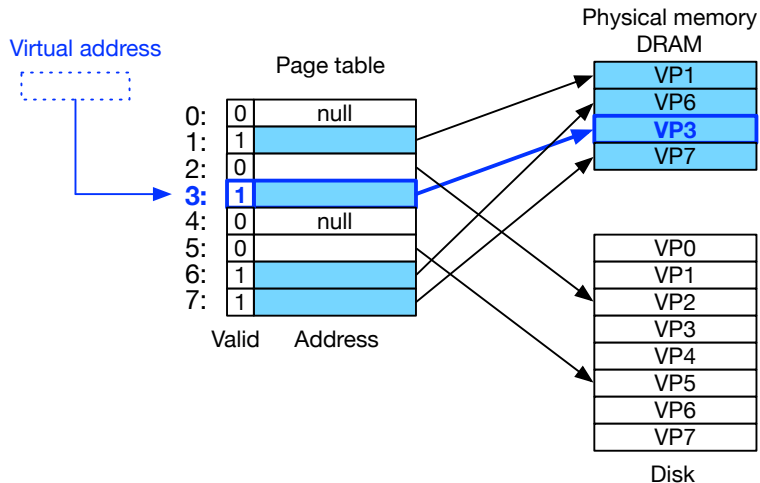
Page Table

- ▶ Array of page table entries (PTE)
(actually, a tree where the leaves store the page table entries)
- ▶ Each PTE maps a virtual page to a physical page
- ▶ Valid bit
 - ▶ set if PTE currently maps to physical address (cached)
 - ▶ not set otherwise (uncached or unallocated)
- ▶ Mapped address
 - ▶ if cached: physical address in DRAM
 - ▶ if not cached: physical address on disk

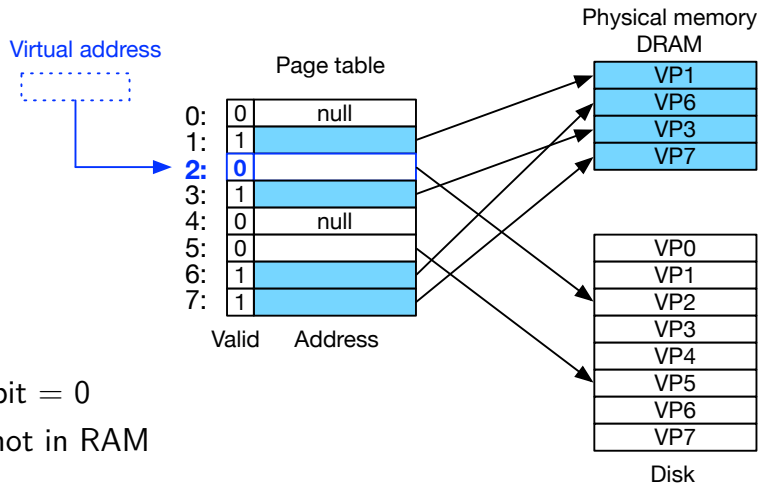
Page Table



Page Hit

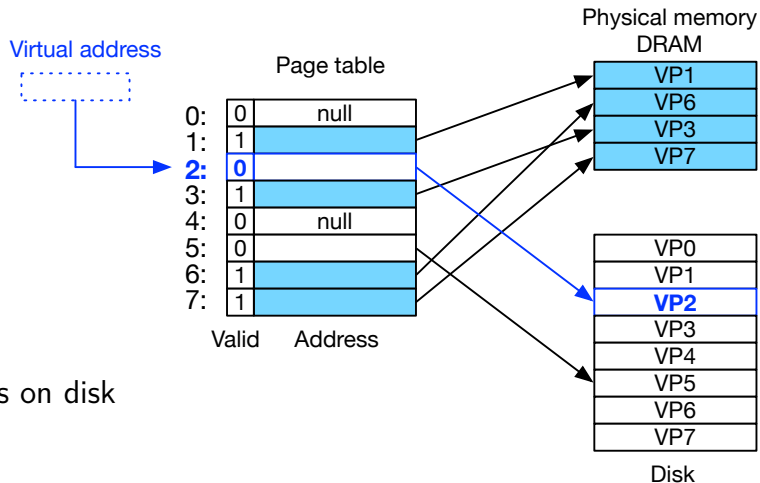


Page Fault

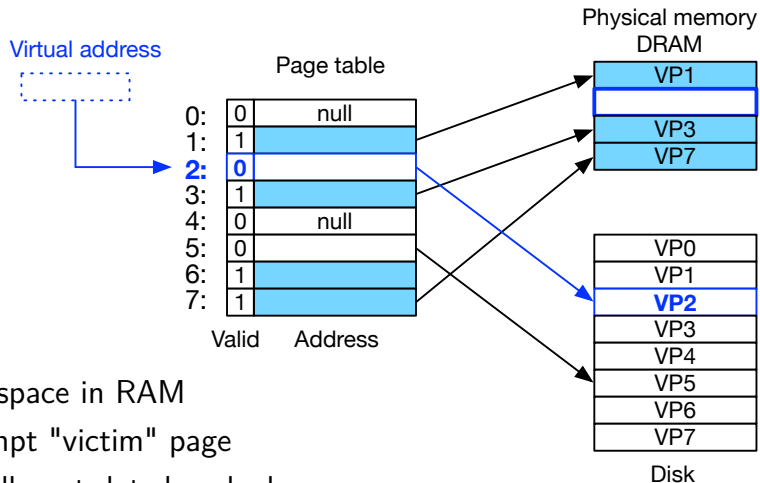


- ▶ Valid bit = 0
- ▶ Page not in RAM

Page Fault

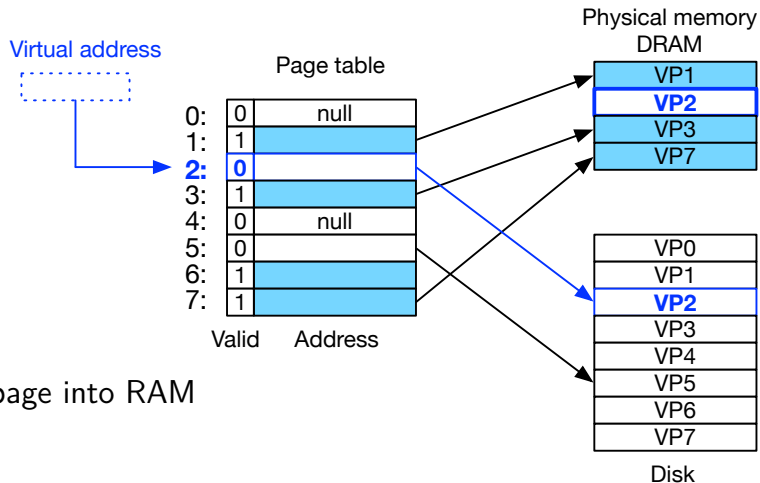


Page Fault

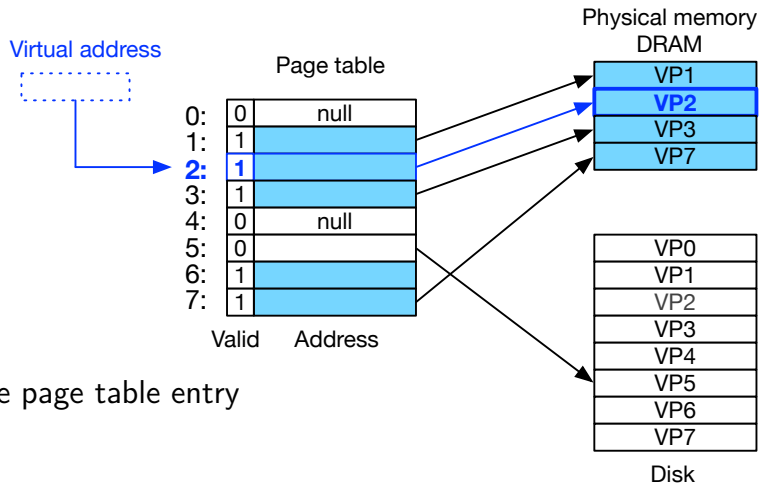


- ▶ Make space in RAM
- ▶ Pre-empt "victim" page
- ▶ Typically out-dated cached page

Page Fault



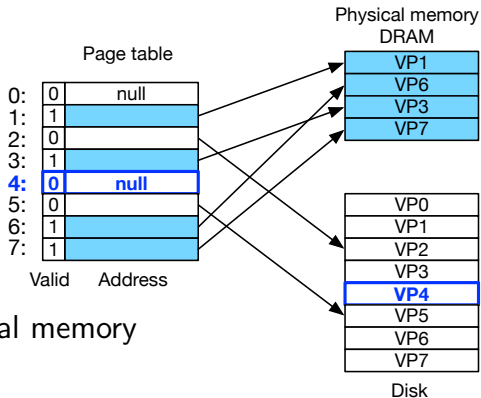
Page Fault



Allocating Pages

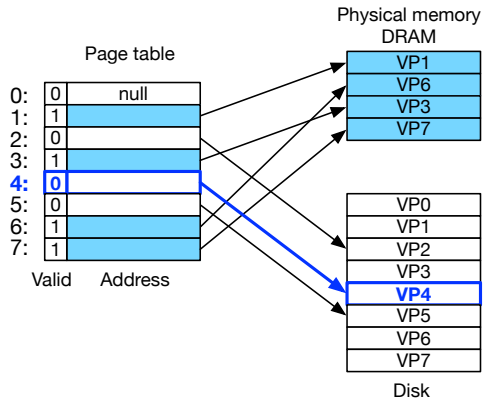
- ▶ What happens when we load a program?
- ▶ We need to load its executable into memory
- ▶ Similar: create data objects when program is running (“allocating” memory)

Allocating Page



- Identify space in virtual memory

Allocating Page



- ▶ Map to data on disk
 - ▶ do not actual load
 - ▶ just create page table entries
 - ▶ let virtual memory system handle loading

⇒ On-demand loading

Process Memory

- ▶ Nothing loaded at startup

Process Memory

- ▶ Nothing loaded at startup
- ▶ Working set (or resident set)
 - ▶ pages of a process that are currently in DRAM
 - ▶ loaded by virtual memory system on demand

Process Memory

- ▶ Nothing loaded at startup
- ▶ Working set (or resident set)
 - ▶ pages of a process that are currently in DRAM
 - ▶ loaded by virtual memory system on demand
- ▶ Thrashing
 - ▶ memory actively required by all processes
larger than physically available
 - ▶ frequent swapping of memory to/from disk
 - ▶ very bad: slows down machine dramatically