CS 1217

Swapping Wrap; Page Replacement Policies; Free Space Management

Logistics

- Lab 3 is out today; might have one more assignment on Synchronization

- 1 make-up lecture during reading week, followed by final exam in same week

Recap: Swapping

- Swap Space: The on-disk space that operating systems typically place data stored in memory in order to *borrow* memory.

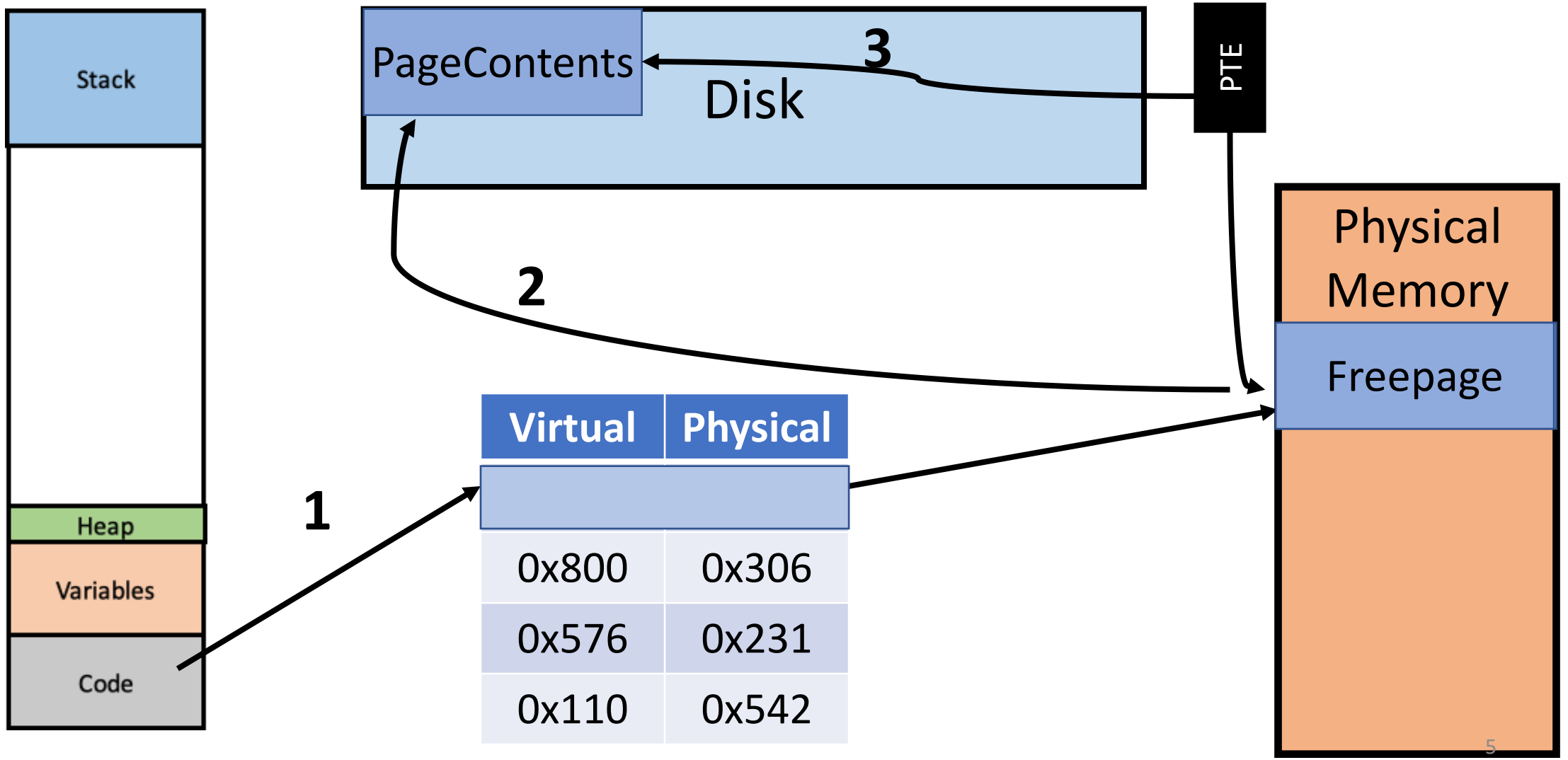
- Swapping: Process of moving (swapping) data back and forth between memory and disk

- Goal: make it **feel** like the system has memory that is as *large* as the size of the disk and as *fast* as actual RAM

Swap Out Process

- **Remove** the translation from the TLB, if it exists
- **Copy** the contents of the page to disk.
- **Update** the page table entry to indicate that the page is on disk

Swap Out Process



Swap Out Speed

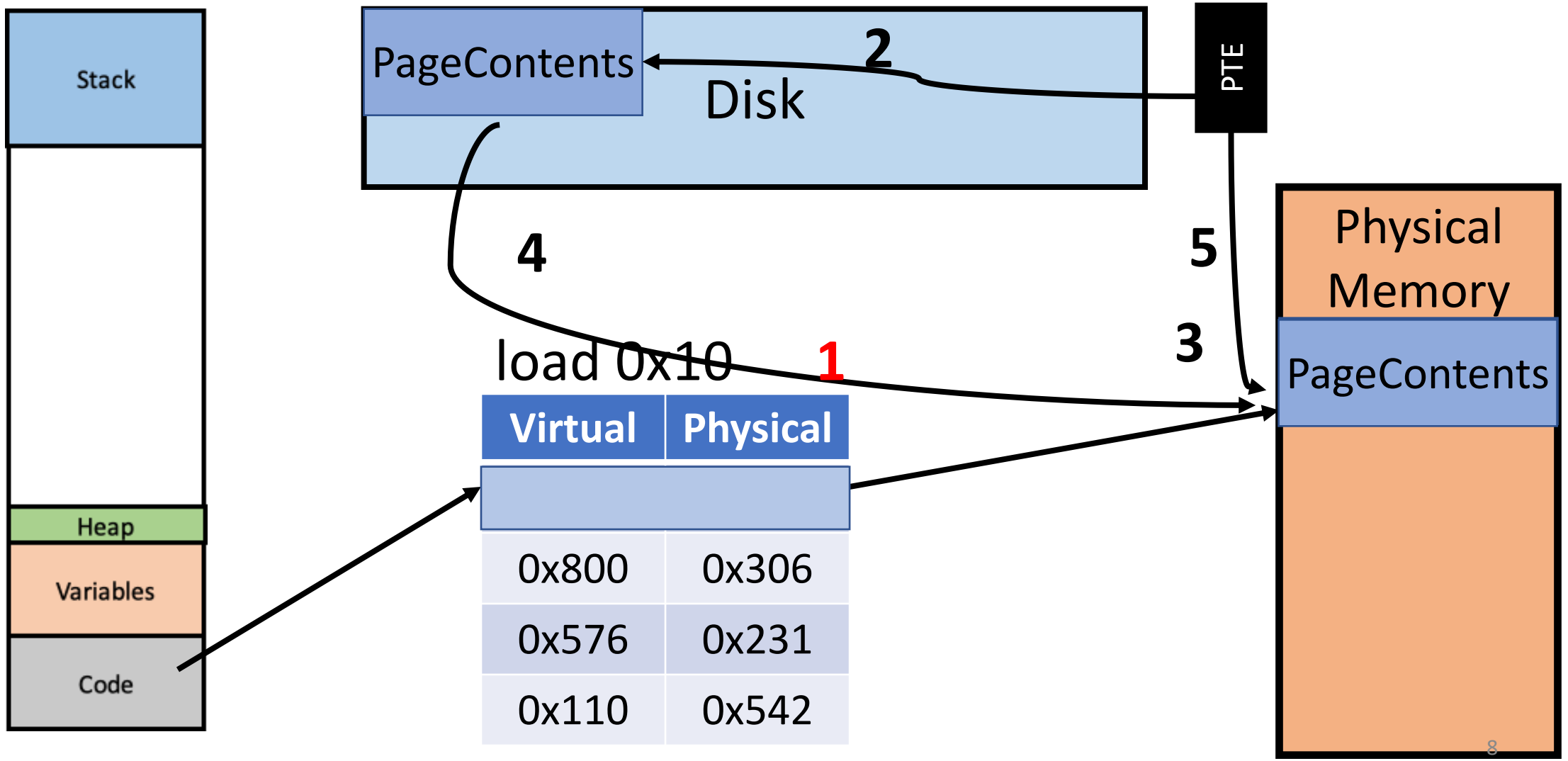
- **Remove** the translation from the TLB
  - Fast or Slow?
- **Copy** the contents of the page to disk
  - Fast or Slow?
- **Update** the page table entry to indicate that the page is on disk
  - Fast or Slow?

Swap In

- When should swap in be done?
  - When the virtual address is used by the process!
- To translate an address for a process that has been swapped out:

  - **Stall** the instruction till contents of page are retrieved
  - **Allocate** a page in memory to hold the new page contents.
  - **Locate** the page on disk using the page table entry.
  - **Copy** the contents of the page from disk.
  - **Update** the page table entry to indicate that the page is in memory
  - **Load** the TLB
  - **Restart** the instruction using the retrieved virtual address.

Swap In Process



On Demand Paging

- Until an instruction on a code page is **executed**, or
- a **read** or **write** occurs to a data or heap page
- the kernel does **not** load the contents of that page into memory!

- WHY?
- A **lot** of code is never executed and some global variables are never used. **Why waste memory?**
- Map virtual addresses to physical addresses “on demand”

On Demand Paging

- What happens the first time a process executes an instruction from a **new code page**?
  - That page contents are loaded from disk and the instruction is restarted

- What happens the first time a does a load or store to an uninitialized heap, stack or data page?
  - The kernel allocates **a new page** filled with appropriate information and the instruction is restarted.

Page Eviction: What can go wrong?

Thrashing *is a colloquialism normally used to describe a computer whose virtual memory subsystem is in a constant state of paging, rapidly exchanging data in memory for data on disk, to the exclusion of most application-level processing. This causes the performance of the computer to degrade or collapse.*

- Wikipedia