Section 7: Lab 3 contd.

CSE 451 20wi

section 7: 2/20/2020
Announcements

- Lab 3 due tomorrow (no late days for lab4)
Copy-on-write Fork Tips

● How is a page different from a memory region?
  ○ A memory region is a contiguous region of memory, can contain multiple pages
  ○ A page is a 4096 byte virtual address range inside a memory region

● Anything else?
Virtual Memory System Visual Diagram

- `struct proc` and `struct addrspace` as part of the virtual memory system.
- `vpmap` as the machine-dependent page table in memory.
- `Memregion [0x0- 0x2000)`
- `Memregion [0x2000- 0x4000)`
- `Memregion [... - ...)`
- Permission in `memregion` is the ground truth, mapped permission might differ if kernel does COW.
TLB Flush

- vpmap_flush_tlb
- when you map a virtual to a different physical page
  - flush to get rid of cached old translation
- when you change permission of a mapped page
  - flush to get rid of cached permission
- memregion_invalidate flushes TLB only if modified region belongs to current process (tries to reduce unnecessary flush)

When should you flush TLB in Lab 3?
Copy-on-write Fork Tips

- when copying over parent's page table entry, make sure to only copy mapped parent's pages
  - if (*parent_pte & PTE_P) { ... }
  - if parent's page is not mapped to a physical page, there is nothing to share

- Do we need to lock around pmem_*?
  - No :) pmem is synchronized internally

- How is a physical page freed?
  - when pmem_dec_ref is decrementing the last reference, pmem_free is invoked
  - no need to explicitly free it, just make sure reference count is updated correctly
Copy-on-write Fork Tips

- What are present, user, write bit used for in page fault handler?
  - `present` indicates if the page exists
  - `user` indicates if the fault occurred in user or kernel mode
    - Doesn't really matter for stack/heap growth since kernel can trigger stack or heap growth
  - `write` indicates if the memory access is a write

- Do I need to call `handle_page_fault` myself?
  - Nope. Page fault handler is a trap handler that is invoked on exception, you should never call this yourself
Office Hour