Machine dependent page table in RAM/TLB

struct proc

struct vspace

struct vspace

vregion

pages

vregion

pages

vregion

pages

pgtbl

Machine dependent page table in RAM/TLB
vregions vs Page Tables

- Both have virtual to physical address mappings.

  - **vspace.pgtbl**
    - Used by hardware to translate virtual addresses to physical addresses
    - CR3 register holds the top level page table (i.e. `vspace.pgtbl`)
    - TLB caches virtual -> physical mappings

  - **vspace.regions**
    - Portable *architecture independent* software representation of the address space
    - Used by kernel to track/update mappings without affecting hardware page table lookups
    - May be incomplete at times (e.g. mappings in exec())

- How do we update the page table to reflect the vspace regions?
vspaceinvalidate(vs)

- "Transforms a vspace into the architecture dependent page table"
  - i.e. virtual mappings in vs.regions are reflected in vs.pgtbl
  - Git analogy: commit vspace changes to the page table
- Call when you've changed a mapping in vs.

Pop Quiz: When will you be calling vspaceinvalidate in Lab 3?
vspaceinstall\(p\)

- “Installs the page table into the page table register”
  - I.e. CR3 = vs.ptbl
  - In x86, this flushes the TLB!
  - Git analogy: pushes your committed changes to the TLB/CR3
- If there were changes in the vspace, call after invalidating.

Pop Quiz: When will you be calling vspaceinstall in Lab 3? Can you ever get away without calling vspaceinstall?
Handling Page Faults in x86_64

- CR2 register holds the faulting linear address (since virtual paging is turned on, this is the virtual address)
  - How do you read or load a control register?
- tf->err holds the exception error code
  - You can use to determine the type of fault

Great resource: https://wiki.osdev.org/Page_fault
Copy-on-write Fork FAQ

- How do we keep track of physical pages and refcounts?
  - Coremap!
- What vspace functions need to behave differently to support COW fork, and how?
  - vspacecopy()
- Synchronization in modifying the vspace in page fault in COW fork?
  - not needed -- current process has exclusive access to its vspace
  - The ref count however could be concurrently modified