Lab 3 More

Memory Management

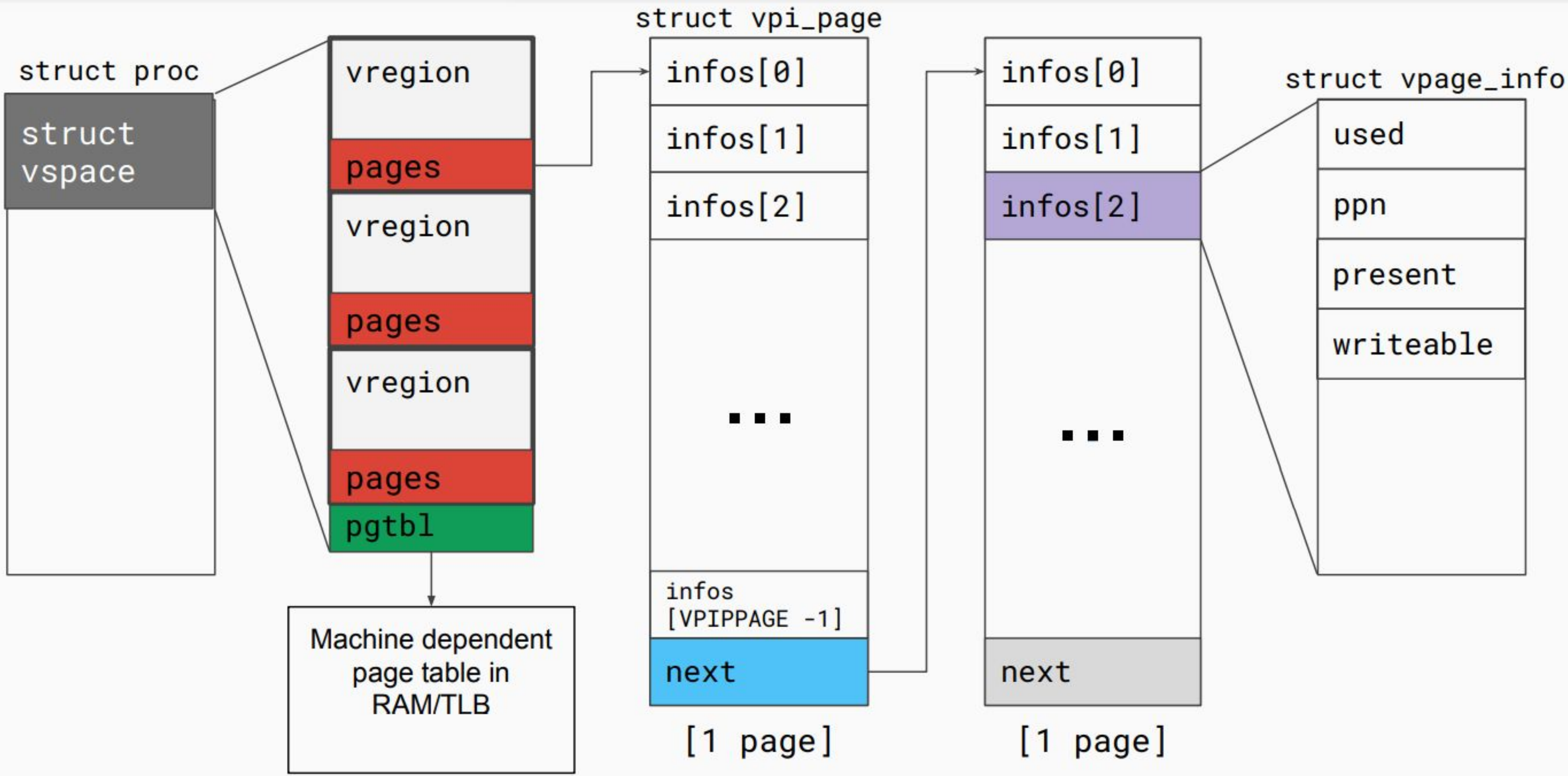
Reminder

- Lab 3 design doc is due tonight

Today’s Agenda

- More detail on vspace and vspace functions
- Some discussion questions on lab 3
- Q&A time

vspace Visual Diagram



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)

- “Build the architecture dependent page table based on vspace information”
    - I.e. virtual mappings in **`vs.regions`** are reflected in **`vs.pgtbl`**
- Call when you’ve changed a mapping in vspace

When should you call **`vspaceinvalidate`** in Lab 3?

vspaceinstall(p)

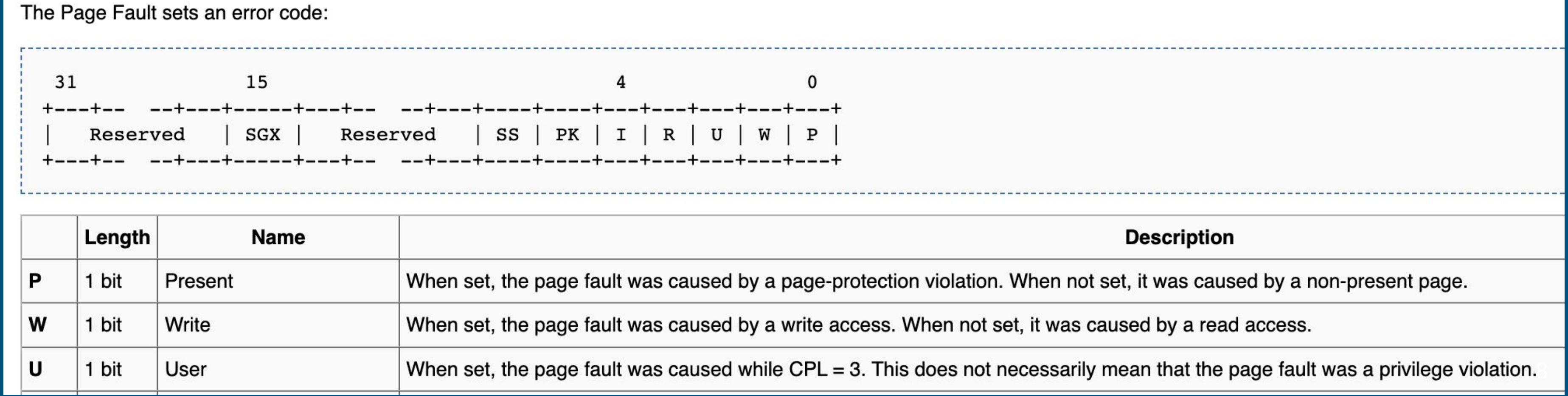
- "Installs the page table into the page table register"
  - I.e. CR3 = `vs.pgtbl`
  - In x86-64, this flushes the TLB!
- If there were changes in the vspace, call after invalidating

When should you call `vspaceinstall` in Lab3?

Can you ever get away without calling `vspaceinstall`?

Handling Page Faults in x86-64

- CR2 register holds the faulting virtual address
  - How do you read or load a control register?
  - (look in trap.c in the default case)
- tf->err holds the exception error code
  - You can use this to determine the type of fault



The Page Fault sets an error code:

```
 31              15                              4               0
+---+--  --+---+-----+---+--  --+---+----+----+---+---+---+---+---+
|   Reserved   | SGX |   Reserved   | SS | PK | I | R | U | W | P |
+---+--  --+---+-----+---+--  --+---+----+----+---+---+---+---+---+
```

	Length	Name	Description
P	1 bit	Present	When set, the page fault was caused by a page-protection violation. When not set, it was caused by a non-present page.
W	1 bit	Write	When set, the page fault was caused by a write access. When not set, it was caused by a read access.
U	1 bit	User	When set, the page fault was caused while CPL = 3. This does not necessarily mean that the page fault was a privilege violation.

More on Error codes

- Last 3 bits of tf->err
  - B2 is set if fault occurred in user mode
  - B1 is set if fault occurred on a write
  - B0 is set if the faulting page is mapped to a physical frame
    - if we page fault on a page that's mapped, then it's caused by permission issues
- What will the error code be if the page fault was from touching the stack region of memory?
- What about writing to a copy-on-write page?

Copy-on-write Fork FAQ

- How do we keep track of physical pages and refcounts?
  - Coremap! (kalloc.c)
- What vspace function to write to support COW fork?
  - vspacecowcopy (basing off of existing vspacecopy)
- What do the fields of a page (struct `vpage_info`) need to be after a copy-on-write fork?
    - fields to consider: used, ppn, present, writeable
    - feel free to add your own fields
- What happens to a page that is already read-only before COW fork?

More COW

- What needs to be changed in the `core_map_entry` to support COW fork?
  - ref count
  - access to core_map_entry should be protected
    - (hint: kalloc already has a lock for all core_map structures)
- Can the kernel cause a copy-on-write page fault?
  - Sure! E.g. accessing the user buffer during a read() system call
- Synchronization in modifying the **vspace** in page fault in COW fork?
    - Not needed -- current process has exclusive access to its own vspace (no multithreading)
    - **However, the ref count on the physical page could be concurrently modified**
- What can happen if a copy-on-write fork is not synchronized?

Helper Macros and Functions

P2V: physical addr to virtual addr

V2P: virtual addr to physical addr

PGNUM: physical addr to page number

va2vpage_info: virtual addr to vpi_info

Any questions?