Project 3 Design Considerations

SECTION… 6? 5? 7?

Preamble

- I know you all are waist-deep in ASST2 right now
- At the very least pay attention enough that you can think to yourself, “Oh yeah, I think he mentioned that in the slides” later when working on your ASST3 design doc

Address Translation

- On memory access: check TLB
- TLB miss? Trap to kernel!
- Kernel looks through page table(s)
- Page table hit?
  - In physical memory? Load the TLB!
  - Not in physical memory? Swap in the proper page!
- Bad address entirely? Kill the program!
- Example on board

TLB

- Entries managed by kernel
- Relevant functions:
  - `tlb_read/write/random/probe`
  - arch/mips/include/tlb.h
- TLB miss → trap to kernel
  - `vm_fault(int faulttype, vaddr_t faultaddress)`
- TLB entry bits
  - `TLBHI_VPAGE`: virtual page index (mask)
  - `TLBLO_PPAGE`: physical page index (mask)
  - `TLBLO_DIRTY`: whether page is writable (flag)
  - `TLBLO_VALID`: whether page is valid (flag)
- What should happen on a context switch?
- Eviction scheme?

TLB (cont.)

- TLB shootdown!!
  - pew pew pew
  - OS/161 terminology is slightly different from ours
    - `ipi_tlbshootdown`: shoot down specified entries on specified CPU
    - `vm_tlbshootdown`: shoot down specified entries on this CPU
    - `vm_tlbshootdown_all`: shoot down all entries on this CPU
    - You need to implement `vm_tlbshootdown/_all`
      - Note: Shooting down all entries technically shoots down any specified entries

Page Tables

- Segments
  - Matches addrspace API and ELF layout
  - What's in a segment?
    - Page range
    - Permissions
- Multiple levels
  - Don't want to keep entire address space for each process
  - Level of splitting is up to you
- What will your page table look like?
- What will your page table entries look like?

Swapping

- We can't fit every user page in memory
- Swap pages out to disk
  - Eviction scheme?
- Protip: use LHD0 raw
  - `"lhd0raw:"`
  - Remember that `vfs_open` mangles the path string
- Need to manage disk locations
  - How will you represent this information?
- Need to map pages to disk locations
  - Where will you keep this information?

Core Map

- Mapping from physical pages to virtual pages
- Remember: core map must also be in physical memory!
  - Core map must be in core map
- How big should the core map be?
  - How many entries does the core map have?
- How do you reserve space for it?
  - `ram_stealmem()`
- When should you reserve space for it?

Kernel / User Memory

- arch/mips/include/vm.h:42
- Kernel memory is linearly mapped
  - arch/mips/include/vm.h:68
  - Might be useful to define a corresponding macro
- What happens when you ask for address…
  - `0x00406a9b`?
  - `0x8160a4df`?
- How do you access physical address `0x07f29c20`?
- Kernel and user pages must coexist
  - Which has priority?