CS354 F24 W13
p. 5,6,7

11/27/24

User/Kernel Modes

What? Processor *modes* are different privilege levels a process can run at

mode bit indicates current mode 1 = Kernel 0 = user

kernel mode can execute any inst, access any mem, access devices

user mode can execute some inst, some memory, ask O.S. to access devices

flipping modes

- Start in user mode
- Only exception switches from user to Kernel mode
- Kernel exception handler(s) can switch back to user mode

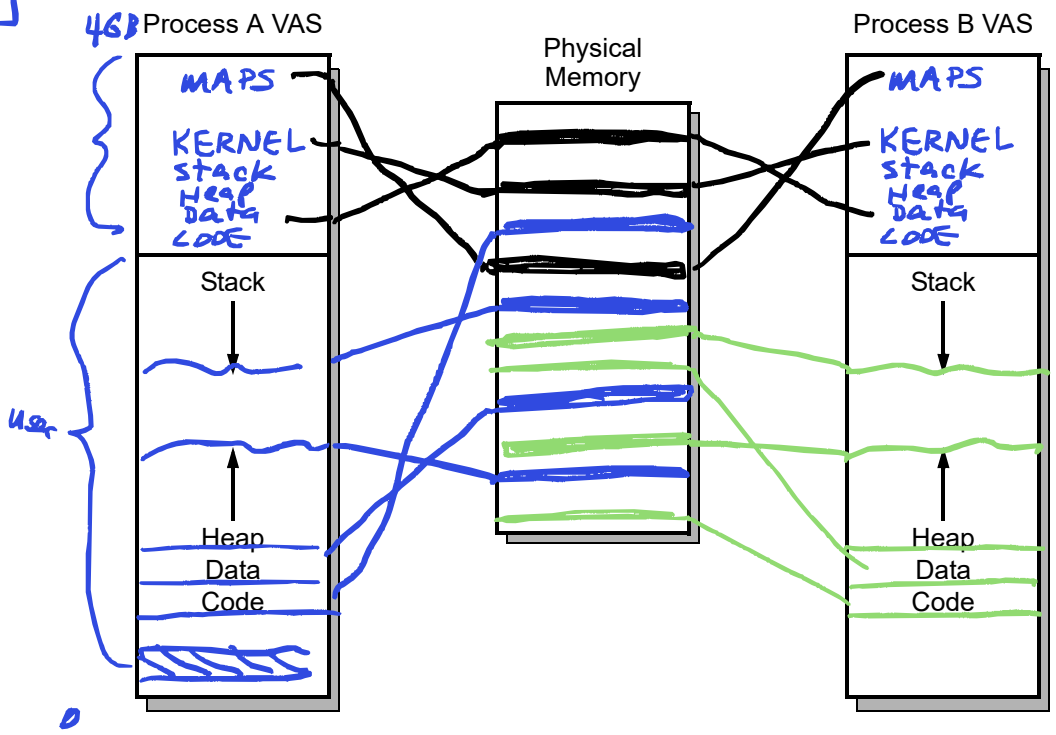
Sharing the Kernel

Key part of OS

Kernel shared by user proc

mem. resident

virtual/physical page 4K



Context Switch

What? A *context switch*

- when OS switches running one process for another
- requires preserving proc. context to restart
  1. CPU state
  2. Use stack
  3. Kernel stack
  4. Kernel D.S.
     a. Page Table
     b. Process Table
     c. File Table

When? happens as result of an exception when kernel executes another process

Why? it enables exceptions to be process

How?

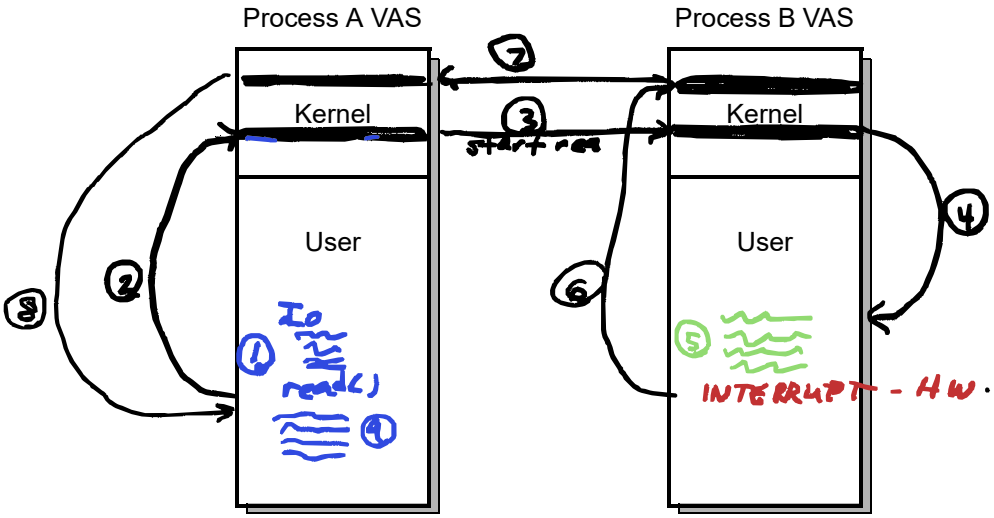
1. SAVE current process context
2. RESTORE other process context
3. TRANSFER control to restored process

❋ *Context switches* are very expensive!

→ What is the impact of a context switch on the cache? Negative! "cache pollution"

Context Switch Example

Stepping through a `read()` System Call



1. In user mode, running Proc A ... get read() - syscall svc #3 int 0x80
2. Switch to Kernel mode, run E.H. for sys call svc #3
3. In Kernel mode, DO CONTEXT SWITCH
   SAVE "A"
   RESTORE "B"
   TRANSFER to "B"
4. Switch to User mode
5. In user mode, running Proc "B" until HW Interrupt
6. Switch to Kernel mode -- run Interrupt E.H.
7. In Kernel mode -- DO CONTEXT SWITCH
   SAVE B, RESTORE A, TRANSFER CONTROL
8. Switch to User mode
9. cont in Proc A after read()