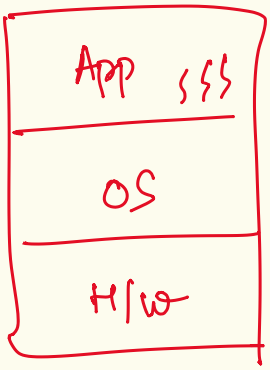
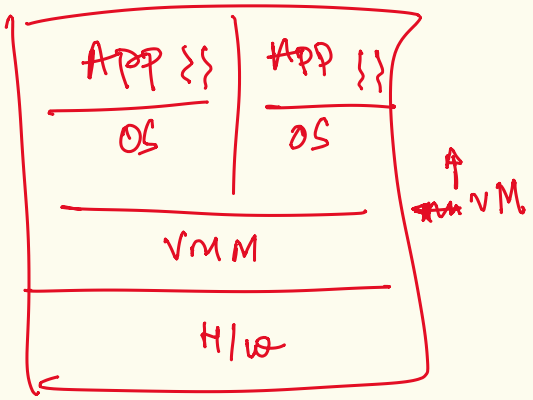
Lecture #3

CS 695





new abstraction — VM
virtual m/c

that provides a system/m/c view.

Assignment #1

due: 29th Jan 11.59pm

3 parts
- LKMs
- ioctl
- procfs

Assignment #2

— our own VM!
~ /dev/kvm
+ ioctl
+ io + mem virtualization

Why VMs?

additional

~ isolation sandboxes on a single m/c

~ server consolidation (sharing/multiplexing PM with several VMs).

~ portability — tricky! (not easy from cost).

~ elastic resources

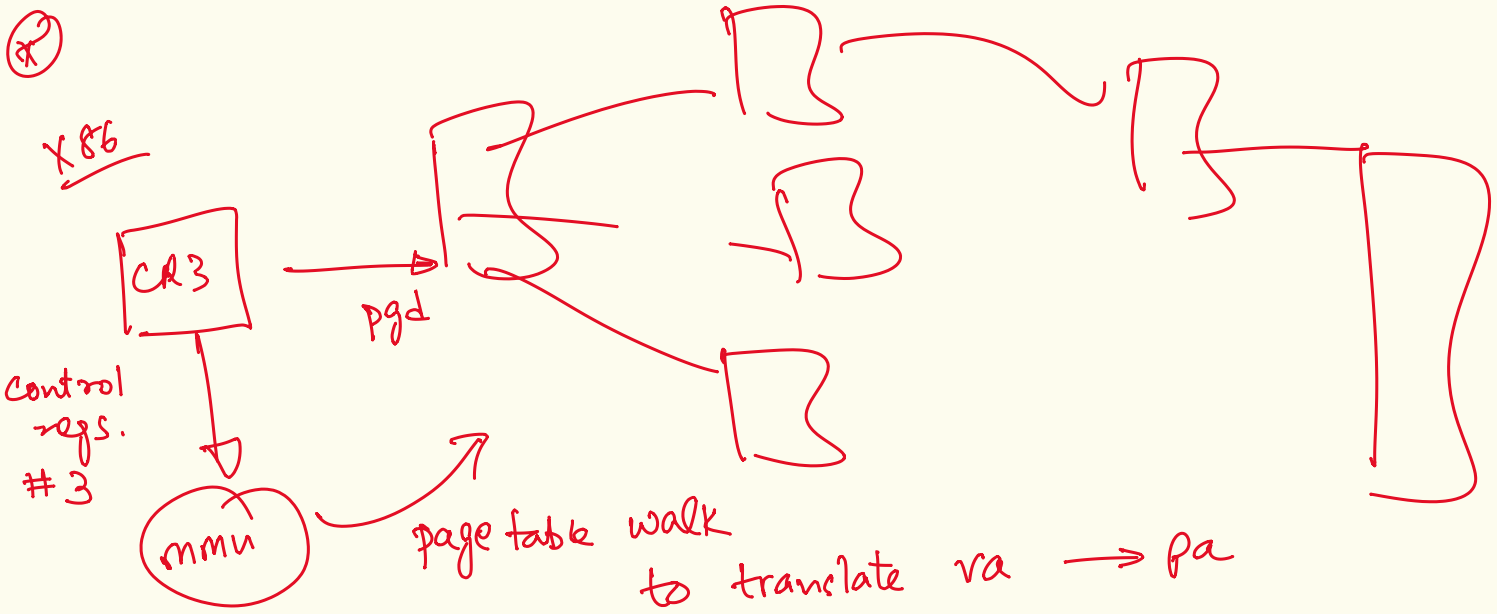
~ multi-OS systems on single m/cs

mechanism + policy

- testing & debugging

— new mechanisms: snapshot, record-replay, ballooning...

(*)



ldcr3 — load value to CR3 regs. instruction

(*) OS building blocks ⇒ ensure that the OS is the sole owner / arbitrator of all resources

(i) LDE — limited direct execution.

(ii) interrupts / interrupt-driven execution

(iii) system call interface

(*) execute all non-OS instructions directly on the CPU, interject when critical/sensitive updates are to be made.

- privilege levels of execution.

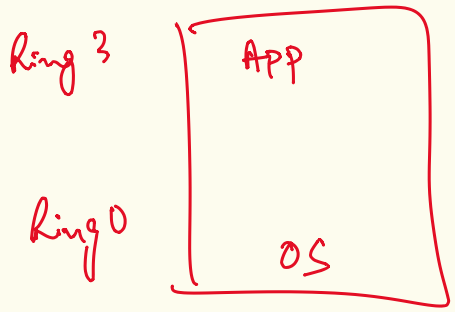
CPU executes in a PL

x86 → 3, 2, 1, 0

every instruction of the ISA has min PL for correct execution semantics.

ldcr3

(*)

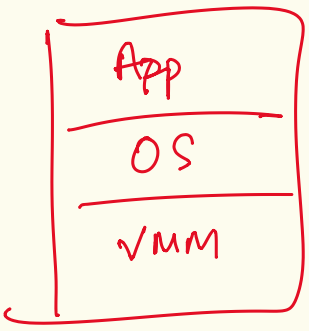


user mode — HCR3 PL=3 ✗

Kernel mode — HCR3 PL=0 ✓

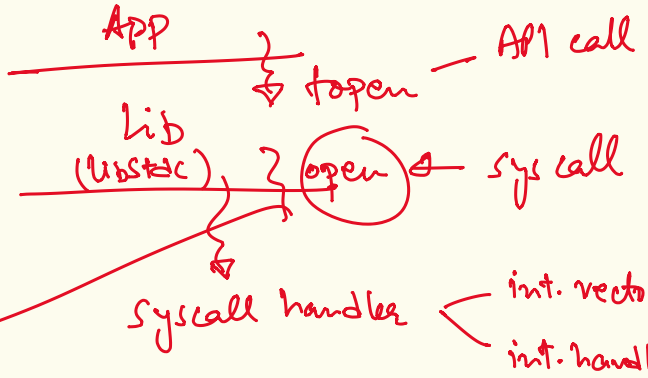
PL=0 highest.

Challenges of hypervisor design!



① who is the owner? of all resources
- Guest OS believes that it is the owner
- multiple guest OSes.
- CPL ~ bit mask of the CS / CR0

② System call handling



- s/w modes from user to kernel
- save user state
- jump to interrupt handler ← IDT ← IDTR

%eax ← 2 — open's sys call no.

int 0x80 ≡ interrupt 0x80

syscall number

③ resource virtualization
- cpu, memory, IO
- eg. CR3

④ device virtualization
- eg: NIC
network interface cards

Design : trap & emulate.