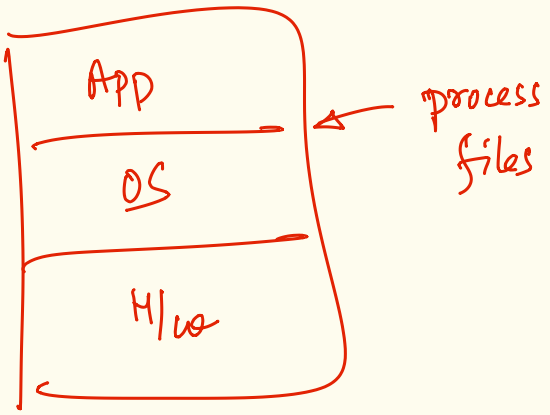
Lecture 2 — CS695

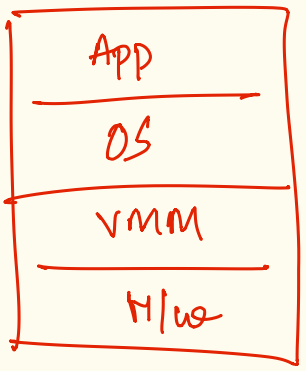
- Assignment 1 available.
- (some changes / updates to statement pending)
- due: 22nd Jan.; Wed.

(i) LKM
(ii) ioctl
(iii) procfs / sysfs

(*) process view vs OS-view



(*) virtual machine is an abstraction of the OS-view.



Q why VMs?

- unified multi-OS environments.
- ~ OS development
- ~ isolation at the m/c level
- ~ primitive for resource sharing (IaaS)
- ~ mechanisms & policies
  - elastic / flexible / on-demand resource provisioning
  - migration, checkpointing, record & replay.

(#) OS redux

OS provides abstractions & virtualizes resources. (sharing/multiplexing)

invariant of all OS design — OS is the sole-owner / controller of all hardware resources.

OS - building blocks

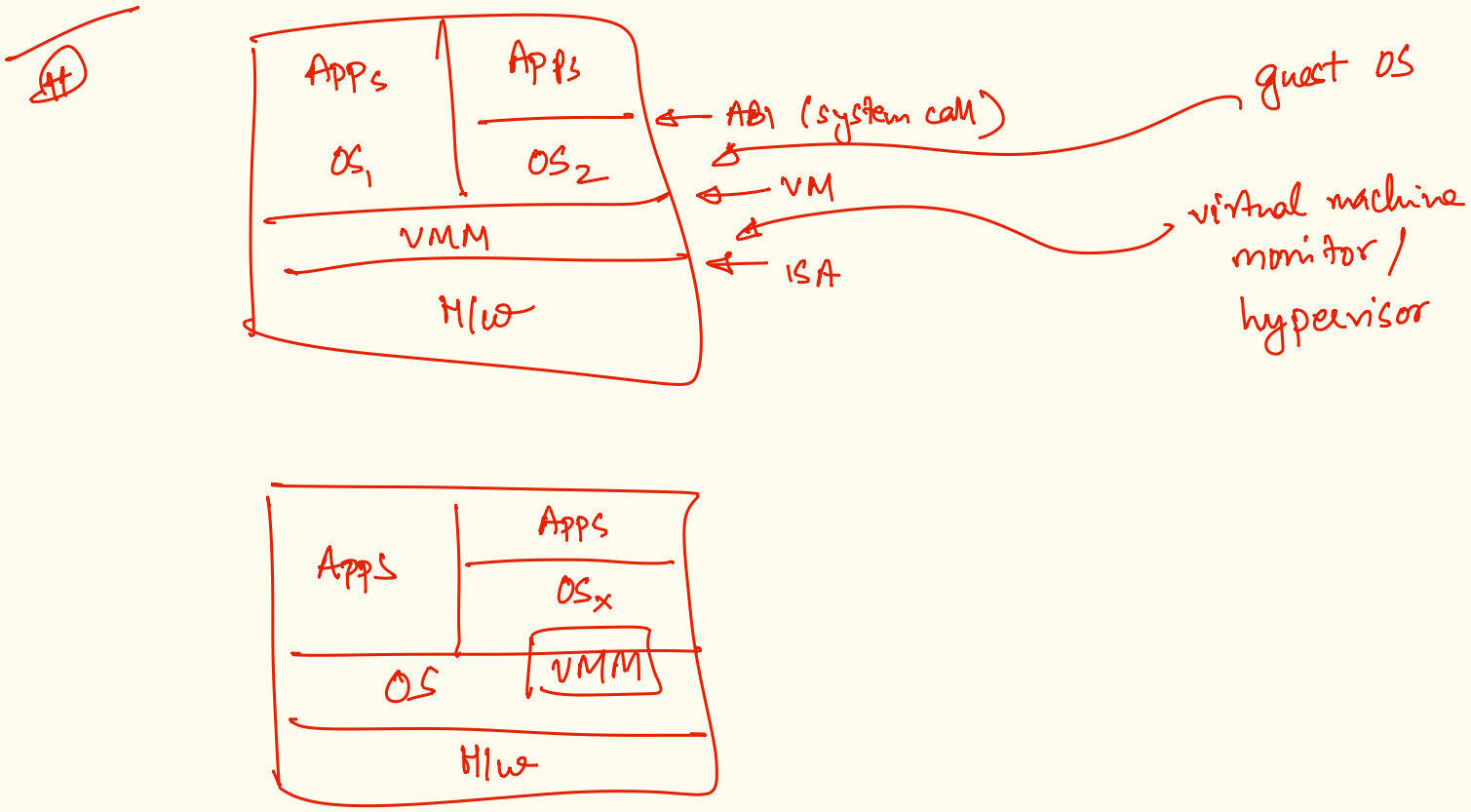
(i) LDE — limited directed execution

every instr. has a min PL to execute.
privilege level

CPU can be set to different PLs while executing.

(ii) interrupts / interrupt driven IO

(iii) system call interface.



Design challenges for VMMs?

(i) who is the owner of resources?

↳ CPL = 0 ~ current privilege level

↳ couple of bits of a control register on the CPU.

(ii) system call handling.

explicit interrupt → switching modes → interrupt handler ~> system call fn.

trap

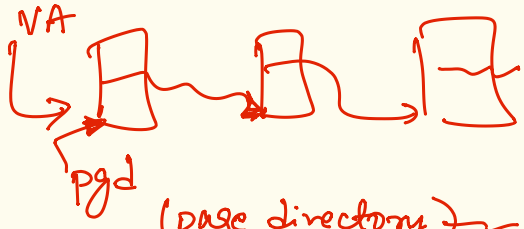
- what is the OS-specific state / view?

~ IDTR — interrupt descriptor table register (stores address of the IDT)

(iii) resource handling

- memory

VA → PA



pgd (page directory) — s/w state / PCB

CR3 ⇒ pgd used by the MMU for page table walks (in h/w).

(#) Popek & Goldberg (1974)

design principles for VMMs / hypervisors.

(i) efficiency ~ performance

(ii) resource control ~ safety / isolation / correctness

(iii) equivalence ~ (fidelity) appl^n's behave identically (semantics) on bare-metal & VMs.

(*) Design #1 — VMs & the CPU resource

OS-inspired (trap-and-emulate)

Bare-metal	VMM	Rings / PL
Apps	Apps	3
	guest-OS	1
Kernel	VMM	0