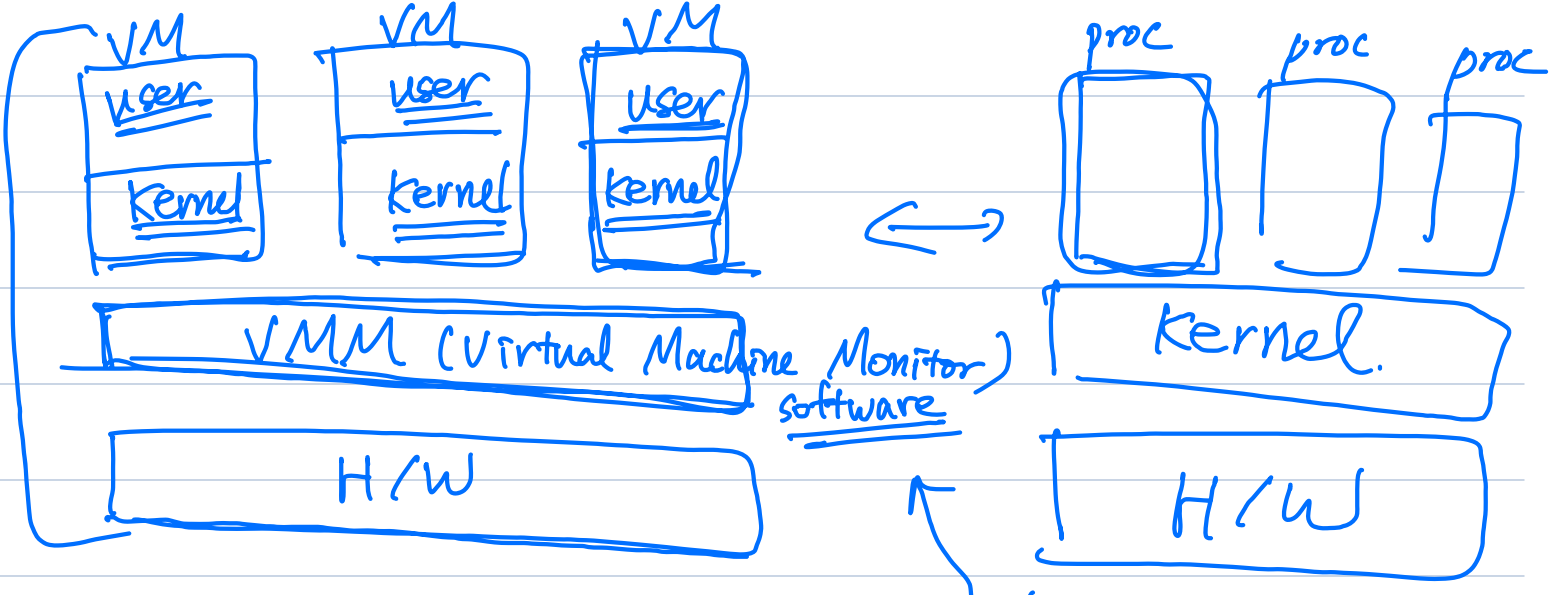
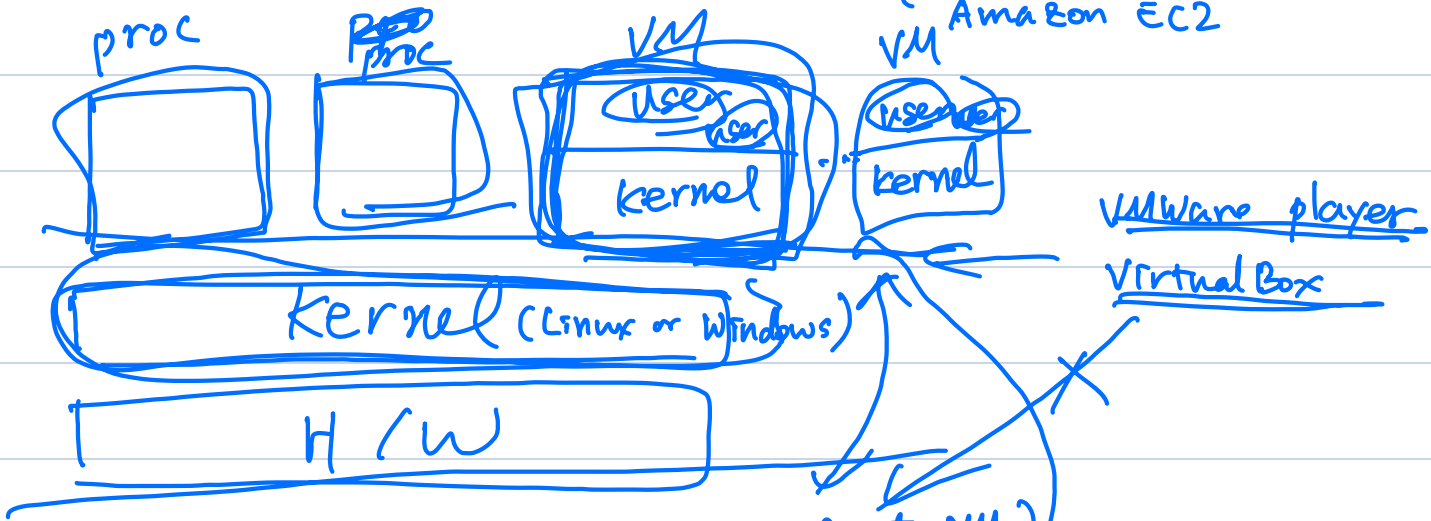
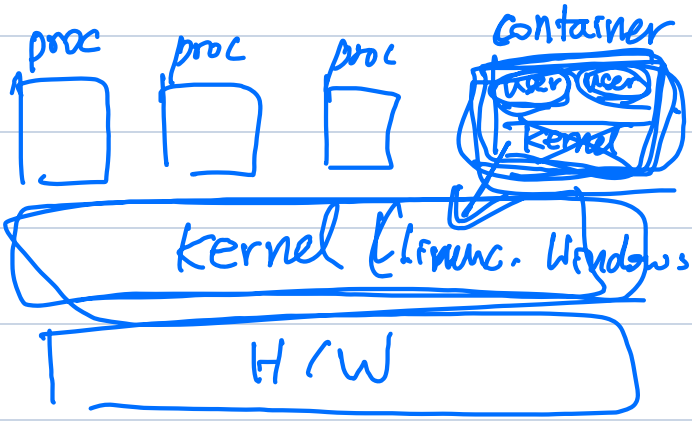


Amazon EC2



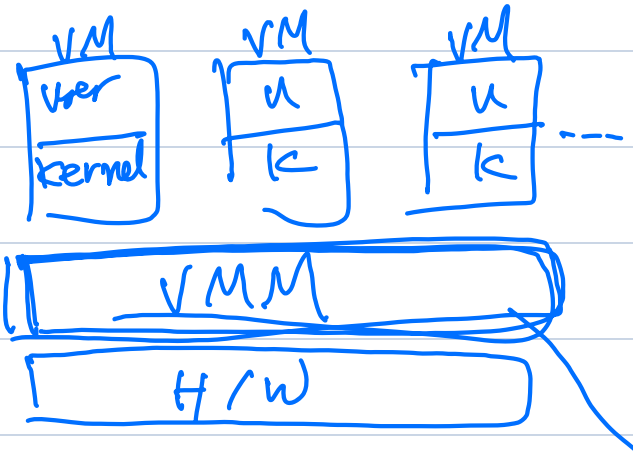
VMWare player
VirtualBox



(not VM) but works as an independent machine

containerization

Docker

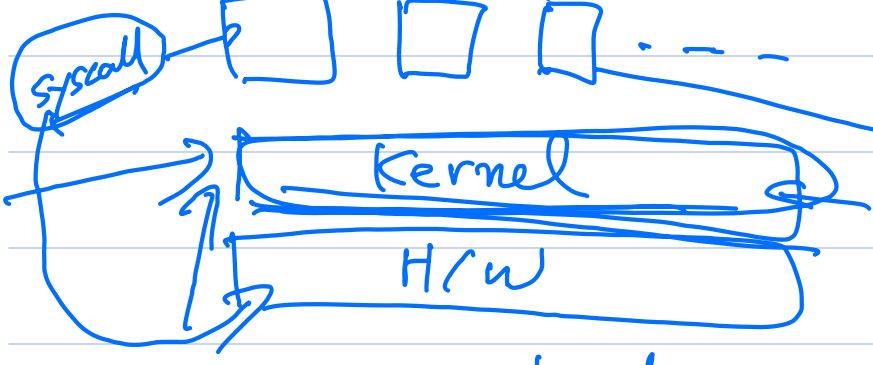


⇒ run multiple virtual machines on the same physical machine.

⇒ Each VM should get its own CPU. memory. I/O devices.

VMM provides an illusion

to each VM that they have their own H/W.



whole machine illusion

"virtualizes" a whole machine.

CPU: privileged instructions that user-level programs can't run directly.

Memory: "virtualization": Virtual memory (page table, per-process)
(kernel reserved memory)

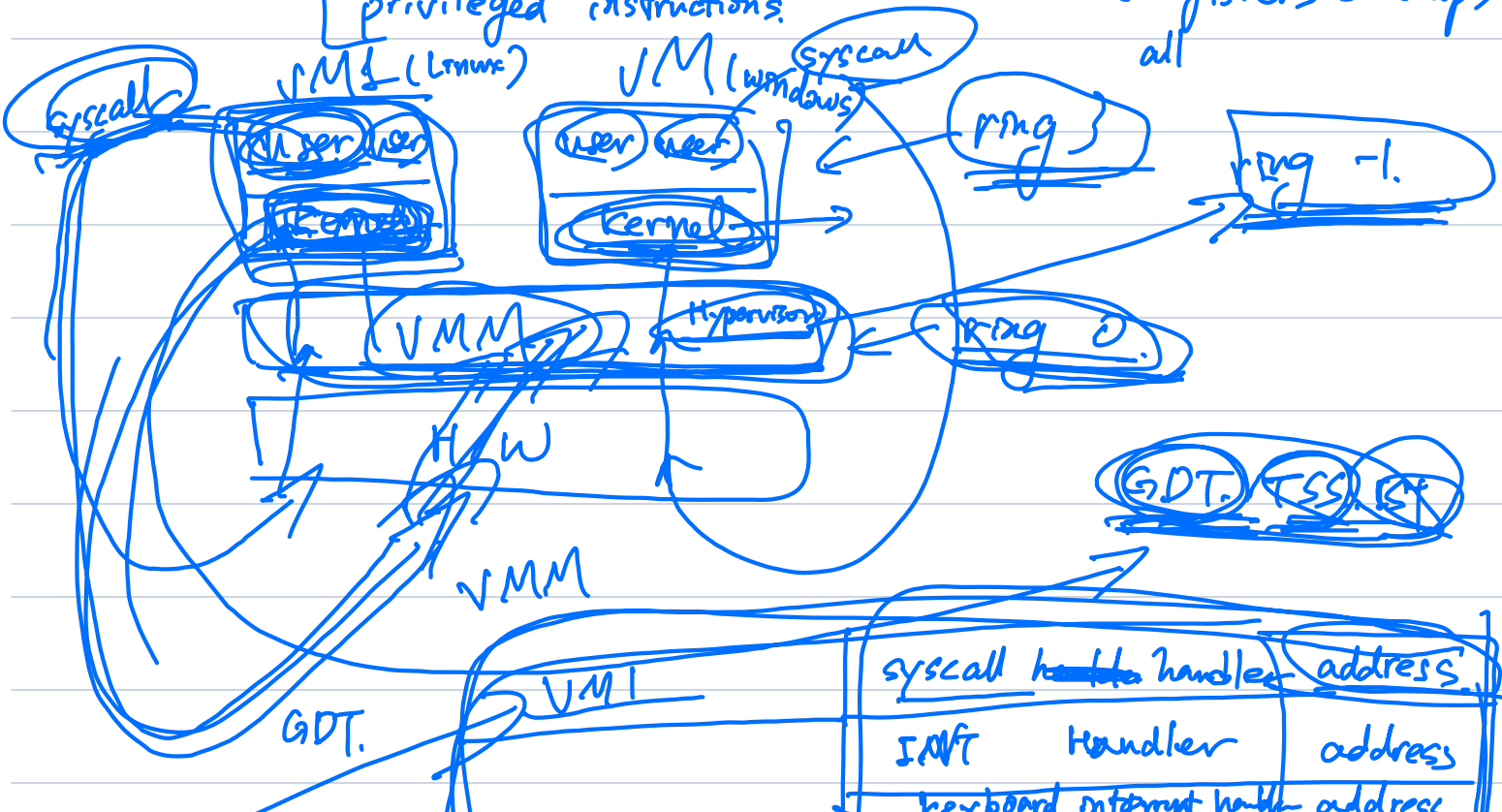
I/O: "virtualization": interrupts. privileged instructions.

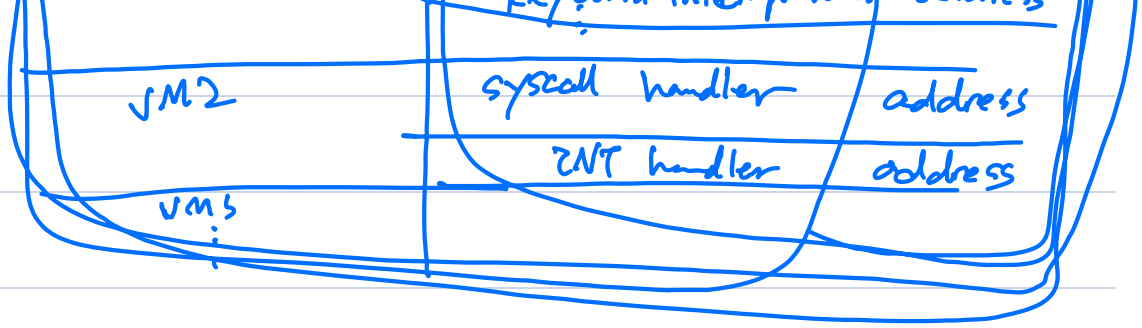
memory protection.

CPU: "virtualization": machine switch

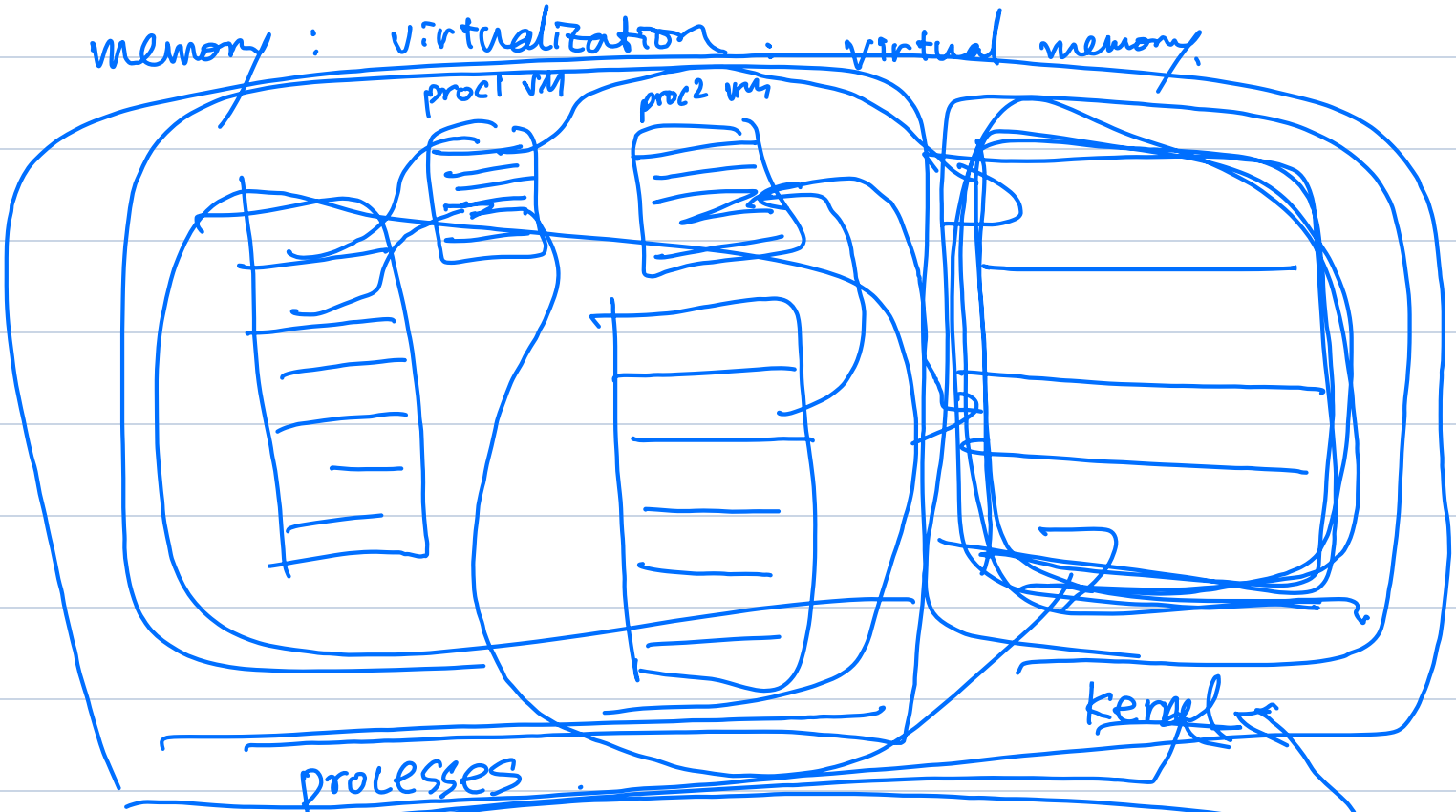
privileged instructions

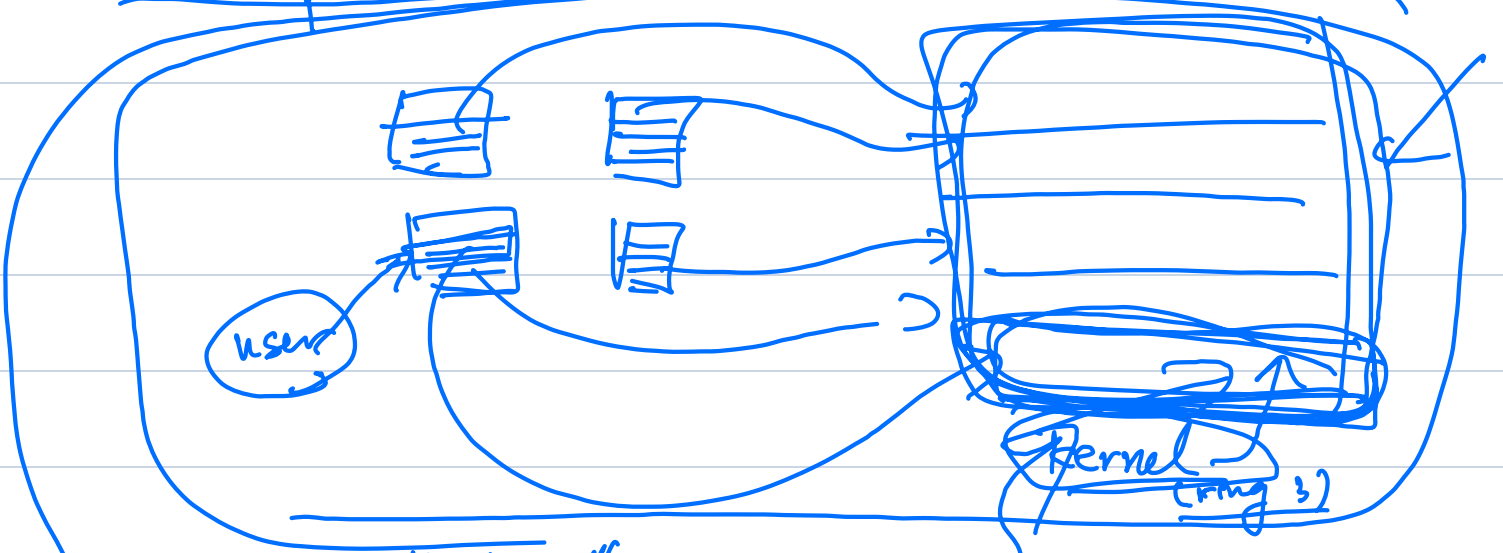
(registers backup)
all





memory : virtualization : virtual memory.



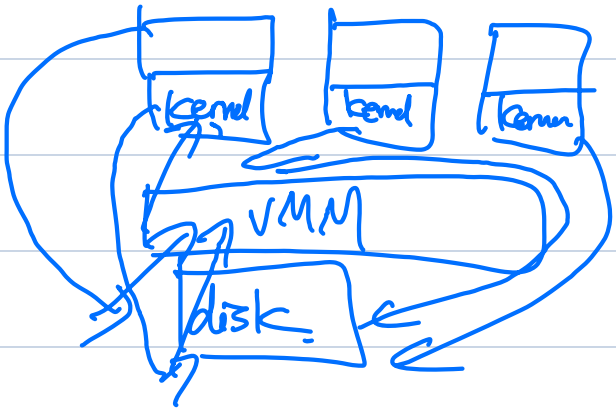


virtual "physical" memory

physical memory

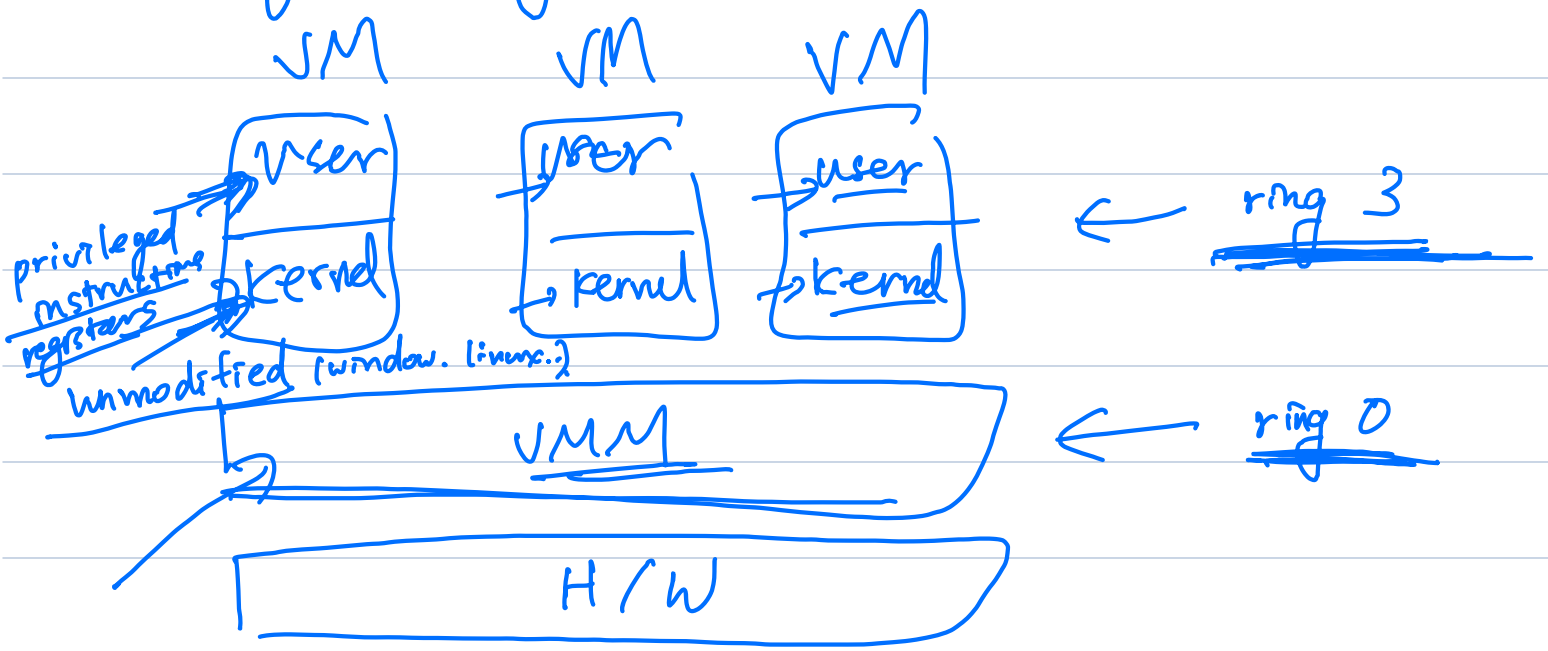
I/O * - I/O instruction (user program)

- trap.
- VMM finds a handler within the corresponding kernel.
- kernel handles it.



IN port # data

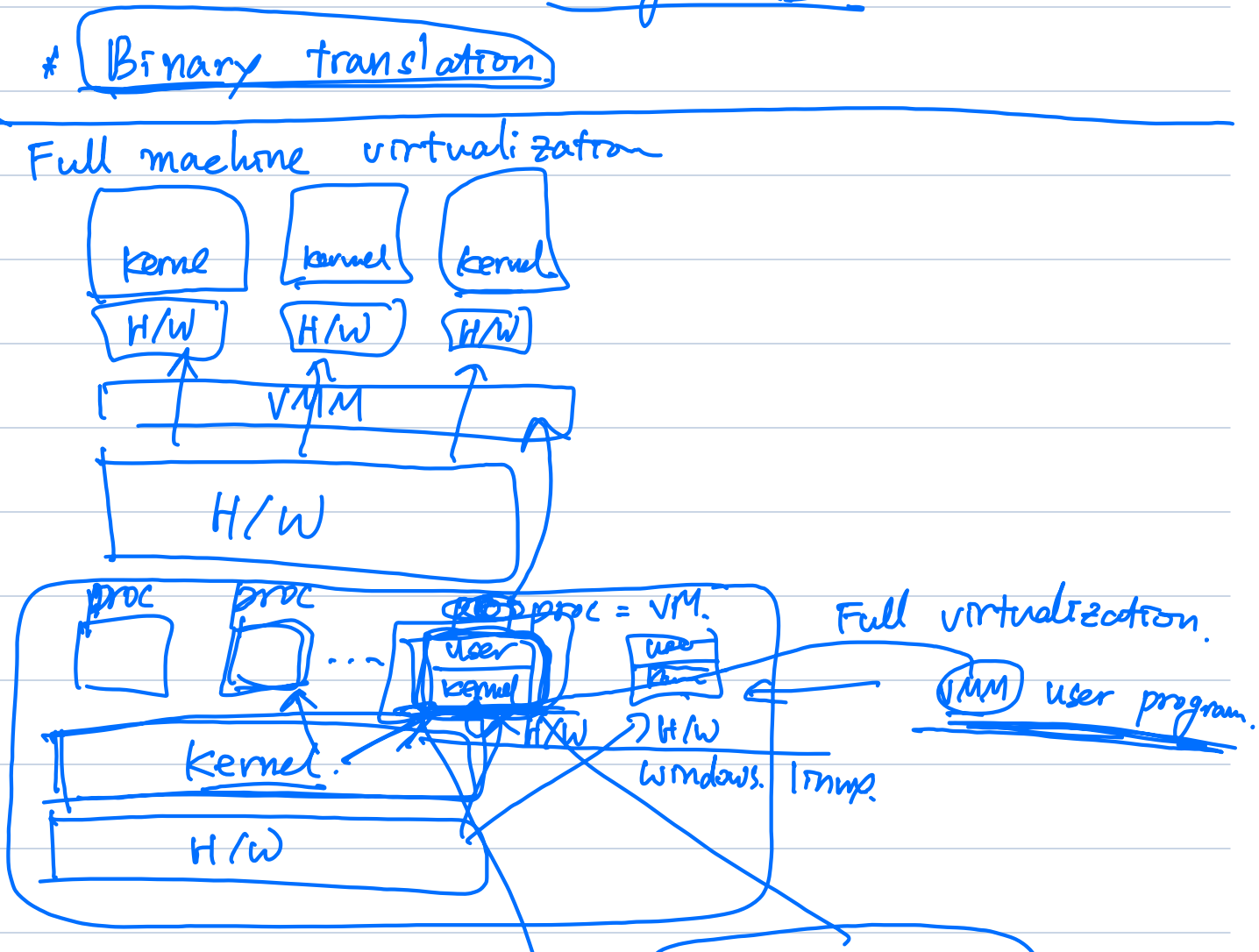
* ring 0 vs. ring 3.

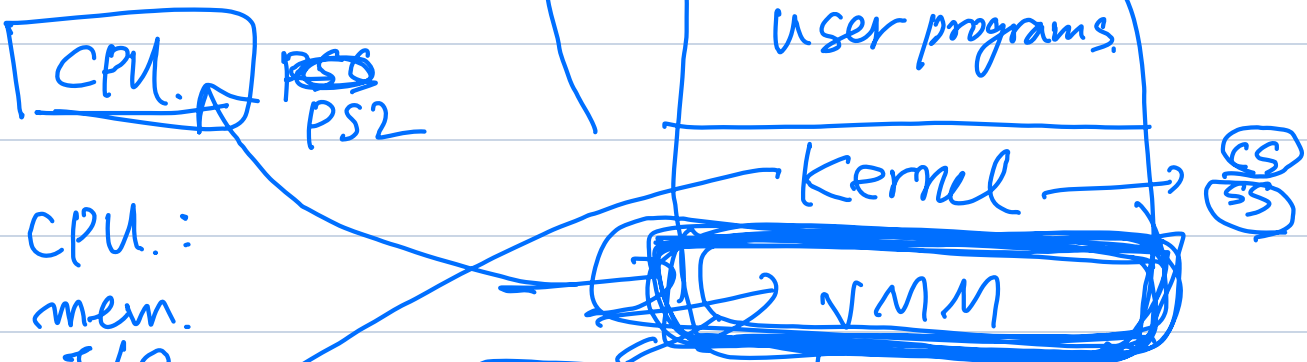


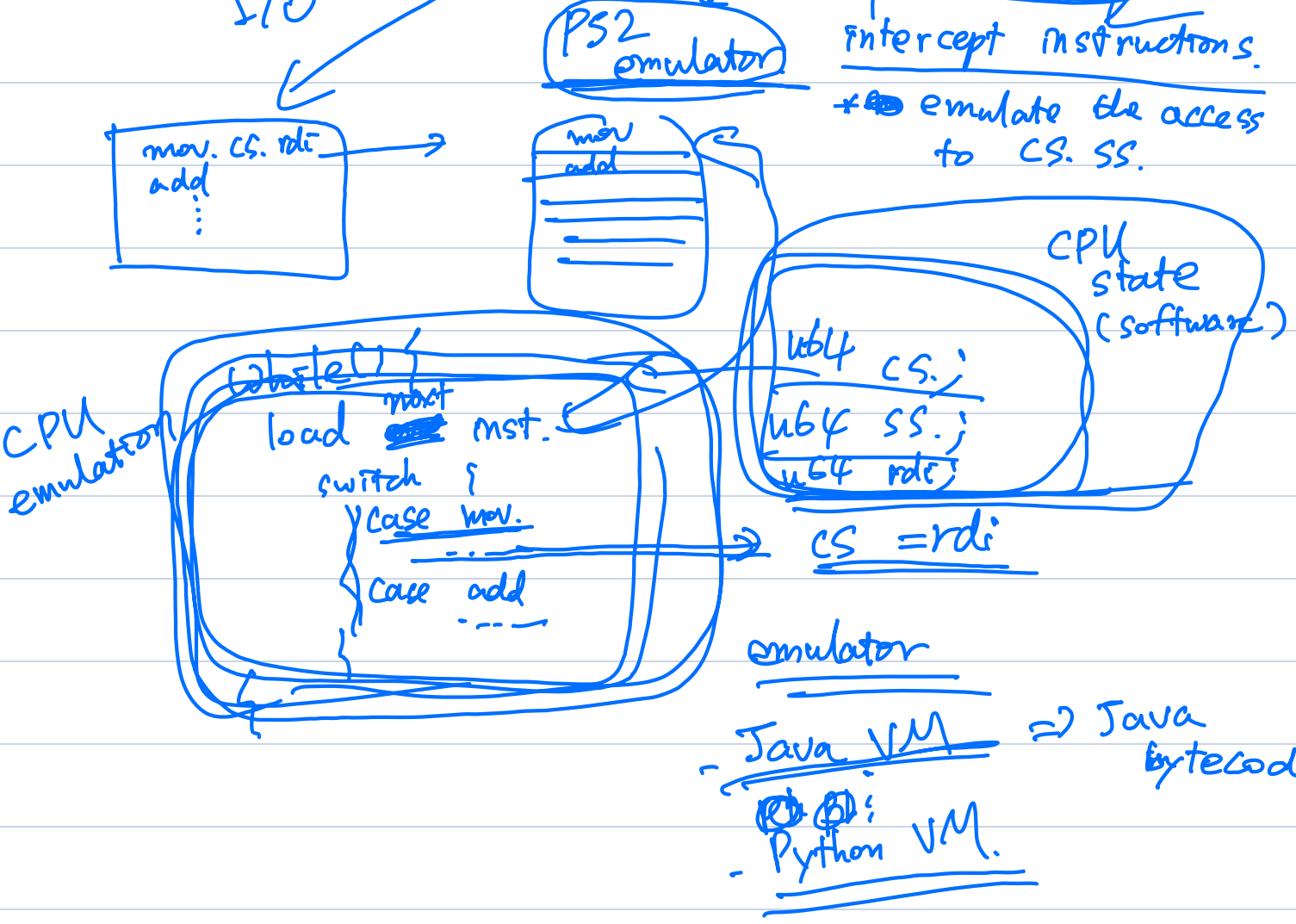
* problem with ring 3
  * kernel memory (pages) now user accessible.
  * performance: kernel can't have privileged access ⇒ VMM needs to handle traps.
  * Instructions that behave differently based on the privilege level.
    popf ⇒ silently fails in ring 3.
    In ring 0. ok

* Binary translation

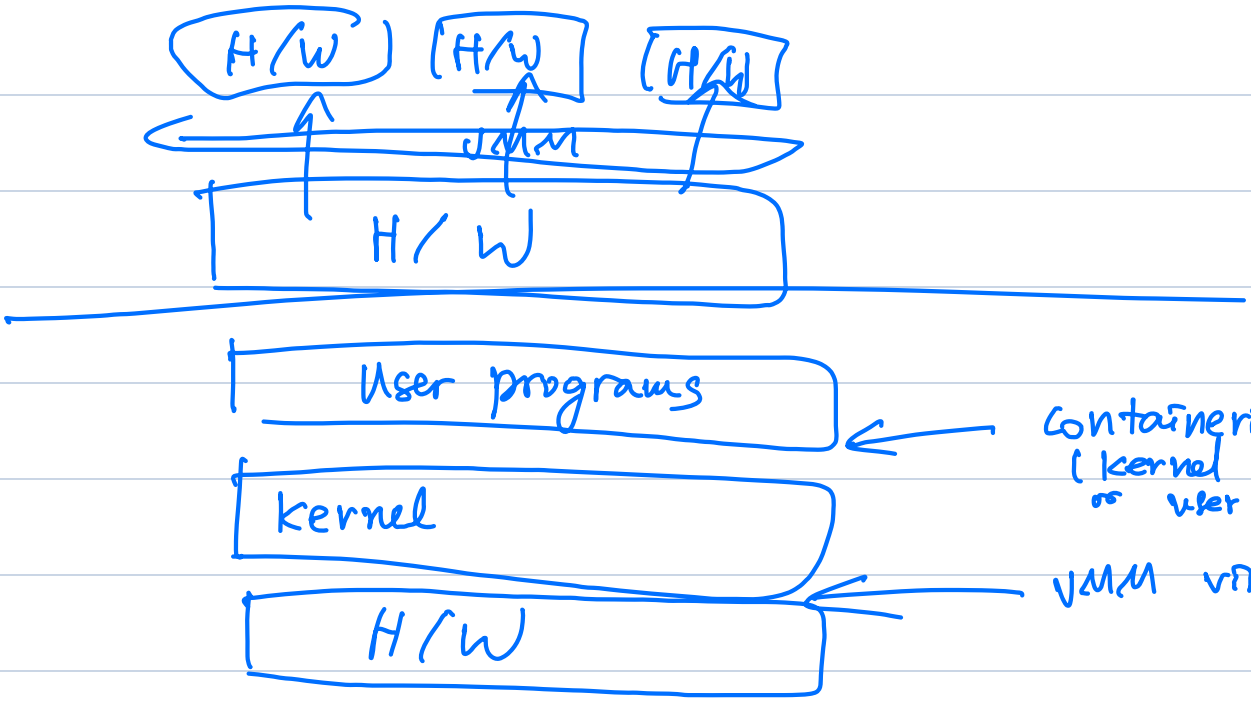
Full machine virtualization

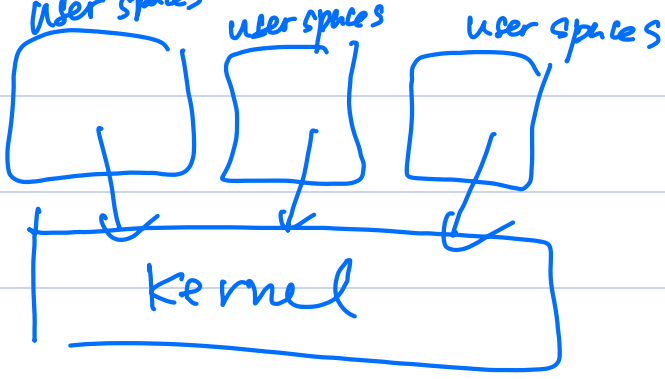






* Containerization

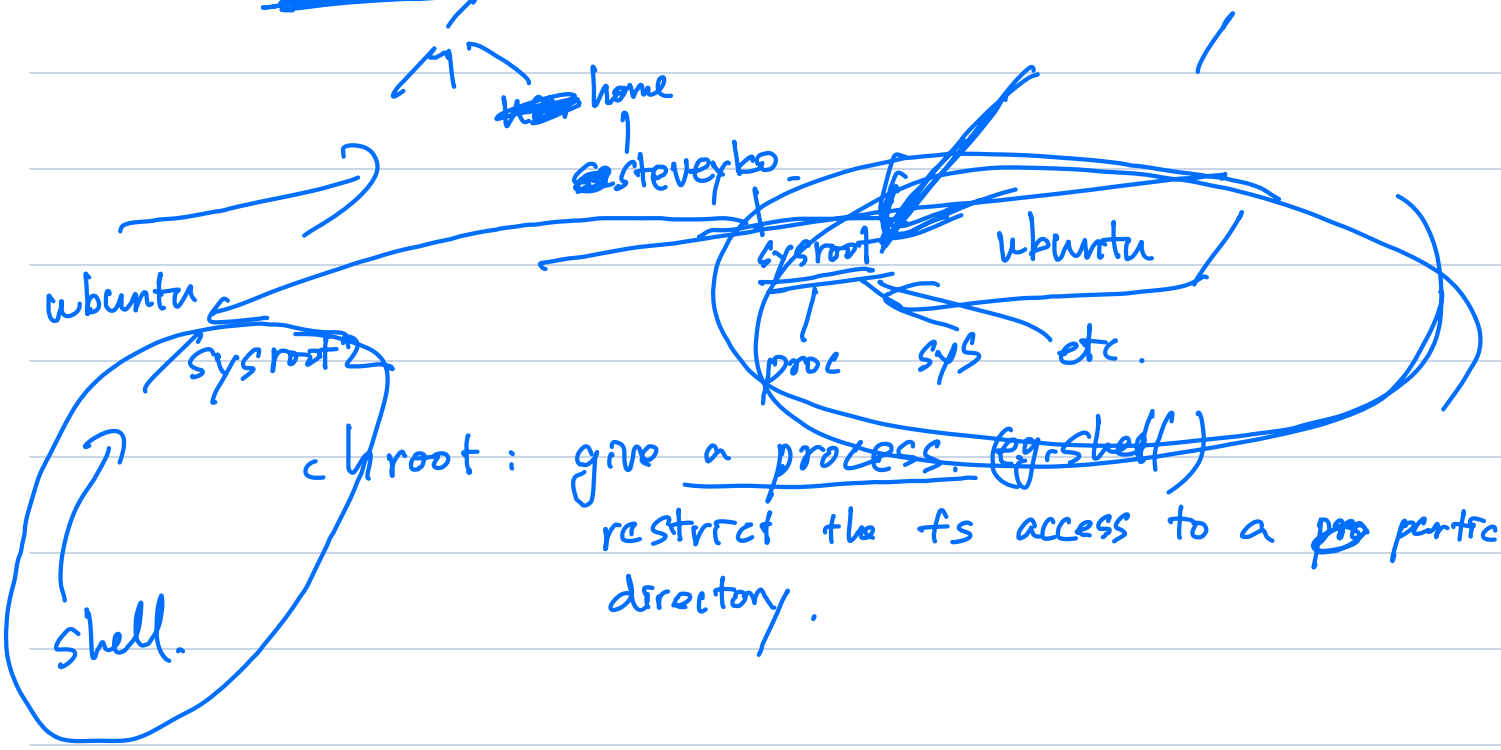




* User space -
  * processes. (UI. shell. othe user programs. ...)
  * File system.
  * I/O.

* chroot. namespace. cgroups.
  ↓ file system ↓ processes ↓ I/O

* chroot.



chroot: give a process. (eg. shell) restrict the fs access to a particular directory.

* namespace -
  * restrict a process's view of other processes.
  * Eop. ps.

* namespace. ← shell
chrome.