Software Emulation

An emulation duplicates the functions of one system using a different system, so that the second system behaves as the first. CPU emulation is made possible by assigning a variable to each register and flag of the simulated CPU. So variables, register (PC, SP, etc) are all involved in software emulation.

Example:

```plaintext
loop: (insn) = *PC++
switch (insn) {
  case mov: Move one reg to another:
    jmp
    PC = val
}
```

Device Virtualization:

A method for assigning a device to a host virtual machine includes connecting
the device directly or indirectly to a computer through an interconnect.

For this we use virtual memory hardware and as we have page table and page directory.

So it has to save registers, current instruction address, PC, before it goes to the page fault handler address.

So it has to save registers, current instruction address, PC, before it goes to the page fault handler address.

register PC → Page handler

register PC → Page handler

sw emulation

reg, pc'

reg, pc'

3) virtual CPU:
Virtualization: [Hypervisor mode]

Parent Partition | Child Partition

Apps: User mode

Kernel: Supervisor mode

But when we virtualize, we put the kernel in user mode and the virtual machine in supervisor mode.

In computing, a hypervisor, also called virtual machine monitor, is a virtualization platform that allows multiple operating systems to run on a host computer at the same time.

\[
\begin{align*}
\text{Ins} & \xrightarrow{pc} \text{Emulation} \\
\text{Insn} & \xrightarrow{} \\
\end{align*}
\]

- User app \( s = 0 \)
- Kernel \( s = 1 \)

- App \( s = 0 \)
- Kernel \( s = 0 \)
- Hypervisor \( s = 1 \)
Trapping the Instructions:

\[ \text{Inst} \rightarrow \text{Traps} \rightarrow \text{Save all the registers, gives the control to the emulators (which execute the instructions) and then go back.} \]

This entire process is called an trapping.

Hypervisor knows the difference between apps and kernel. Receives the trap between application and the kernel, emulates its own code and returns. Now we have to list faults on inst that needs to be virtualized.

Hypervisor is an application?

- VMware - installing handler underneath Windows.

So on x86 we have VMX running.

What do you need to be fully virtualized?

A processor to be fully virtualizable all sensitive instructions must trap.

'Sensitive Instructions' - behave different in user mode [super mode]

load cr3 → usermode will be a fault
- kernel mode it will load.
- A processor to be fully virtualize all sensitive instruction must trap.

- Every time application generates system call hypervisor emulates all the kernel code, slowing down the kernel but app's run in user mode that means OS is slow.

- How to fix e.g. x86.
  
  Think of hypervisor as an operating system:

  - Emulate all OS code.
  
  - jvm, x86 emulator
    - doing the same thing
    - virtual register
    - same techniques
      identifies the sequence of code that safely translates
      and generates safe code

  - Recover most of the speed by
    translating the kernel code (hidden cache of execution code)