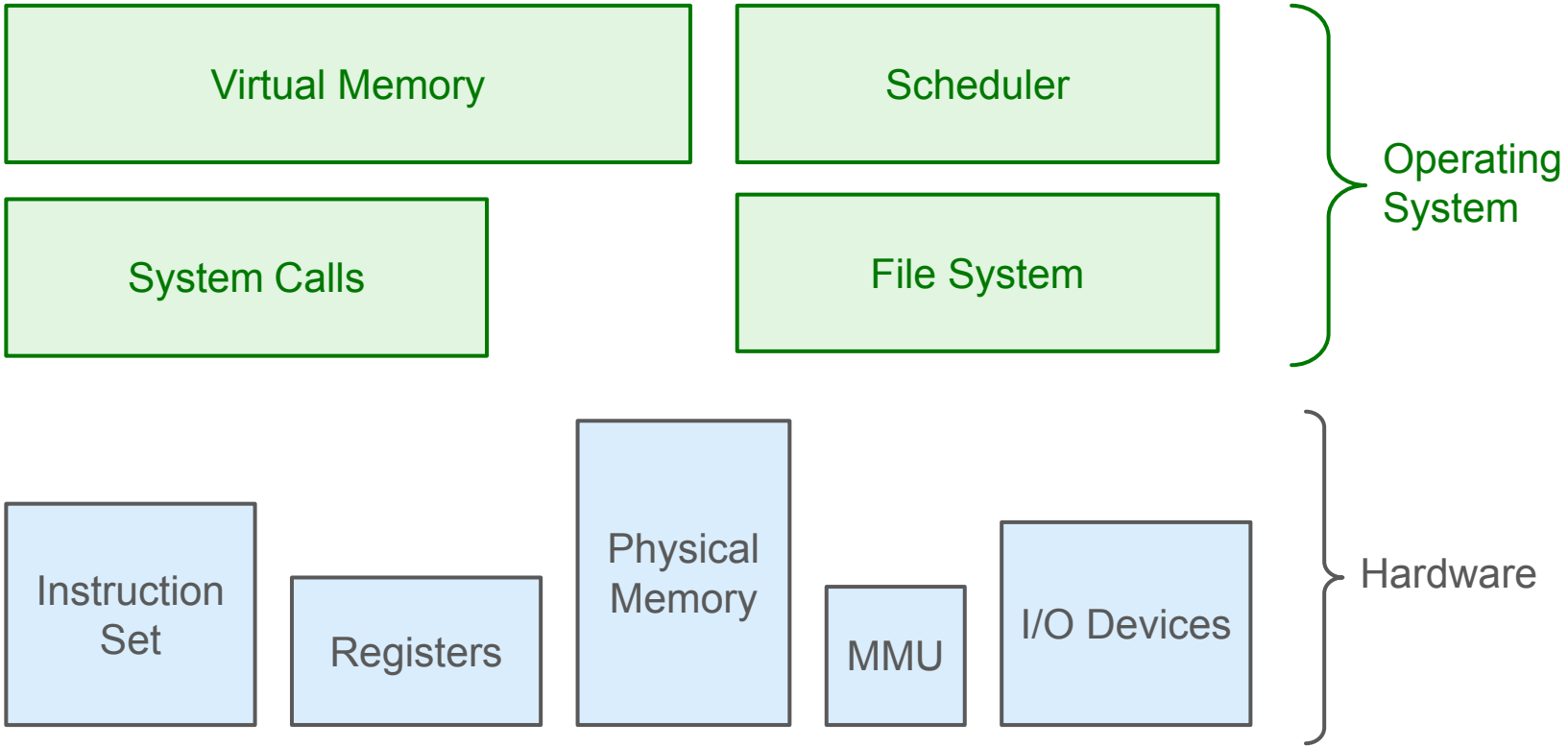
Virtual Machines

Mendel Rosenblum

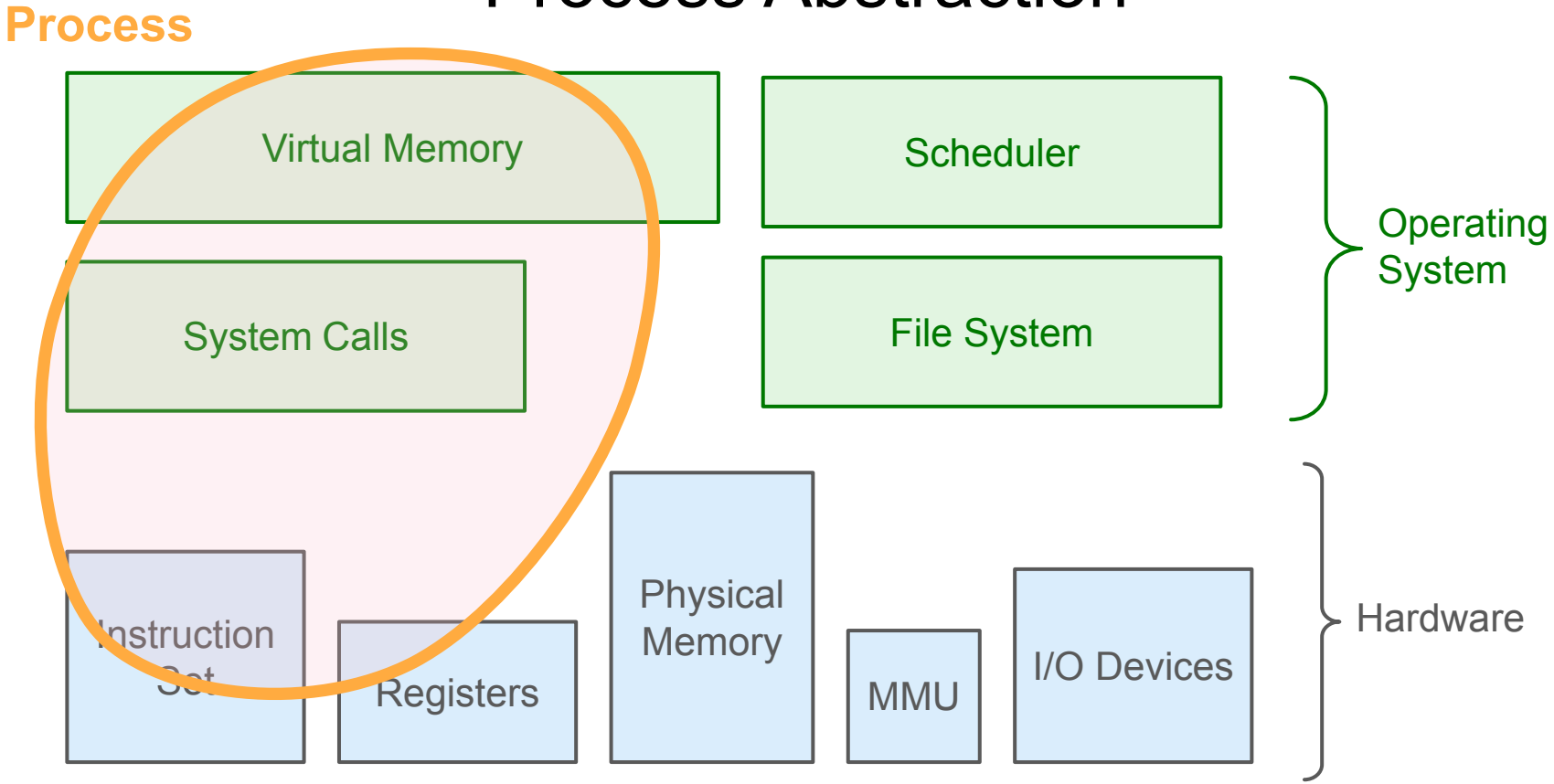
Virtual Machines

Optional readings: Operating Systems: Principles and Practice: Section 10.2

Operating System

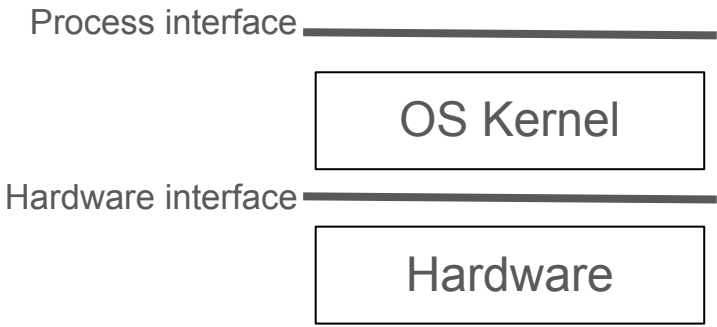


Process Abstraction



OS Process Abstraction

- Memory:
  - Linear array of virtual memory pages
- CPU:
  - All non-privileged instructions and registers
- System calls:
  - FIle I/O (e.g. open/read/write)
  - Process operations (e.g fork, thread create, wait, exit)
  - Few other calls
- Overall: a subset of the facilities of the underlying machine
  - Some close (e.g. CPU), other pretty different (e.g. Memory, Files)



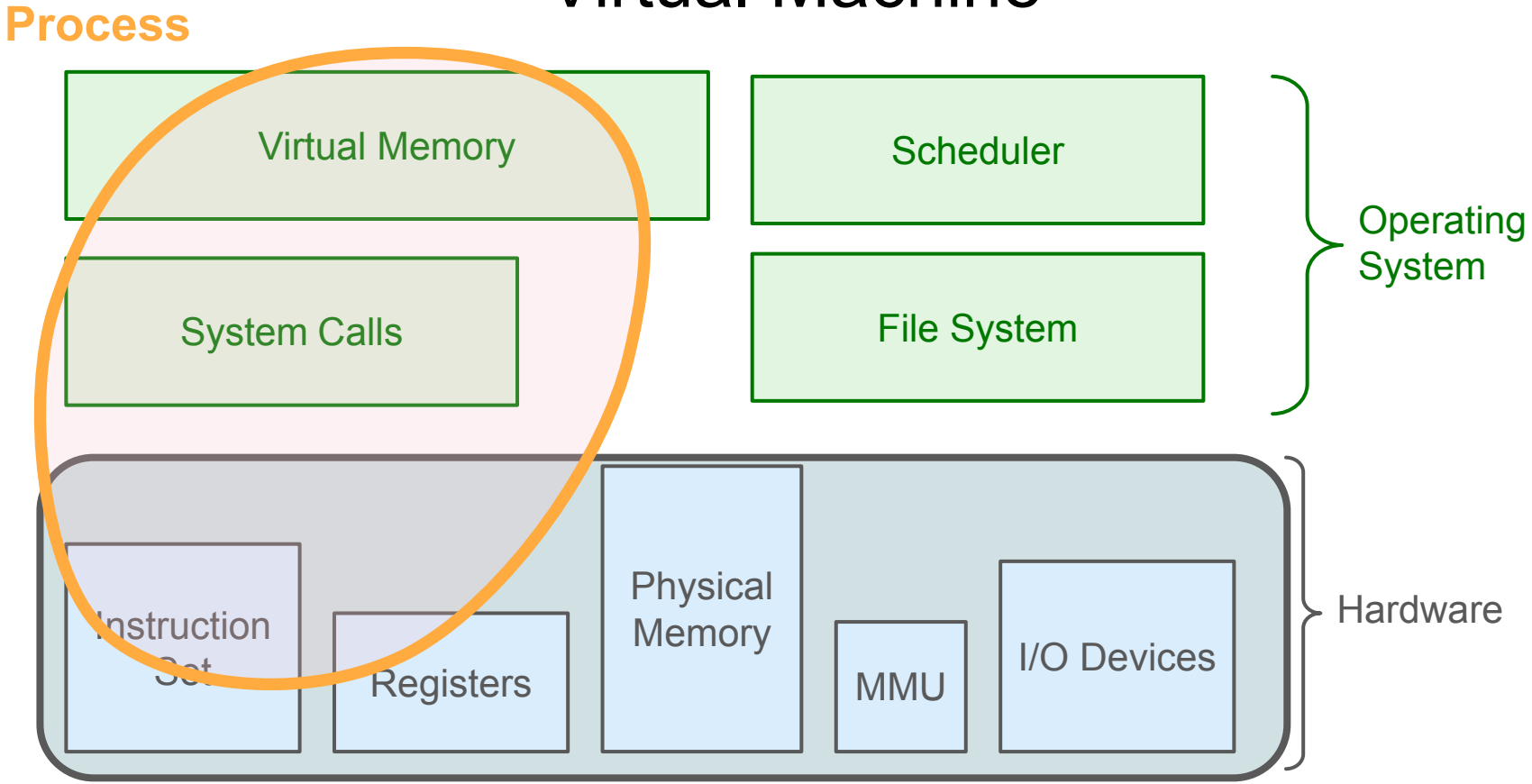
What if we made process look like hardware?

- CPU:
  - All instructions and registers, privileged and non-privileged
- Memory:
    - Physical memory pages
    - Memory management unit (page maps, etc.)
- I/O devices
  - Timer, Disks, Network interface, Display
- Traps and interrupts
  - System calls are just traps

Virtual Machines

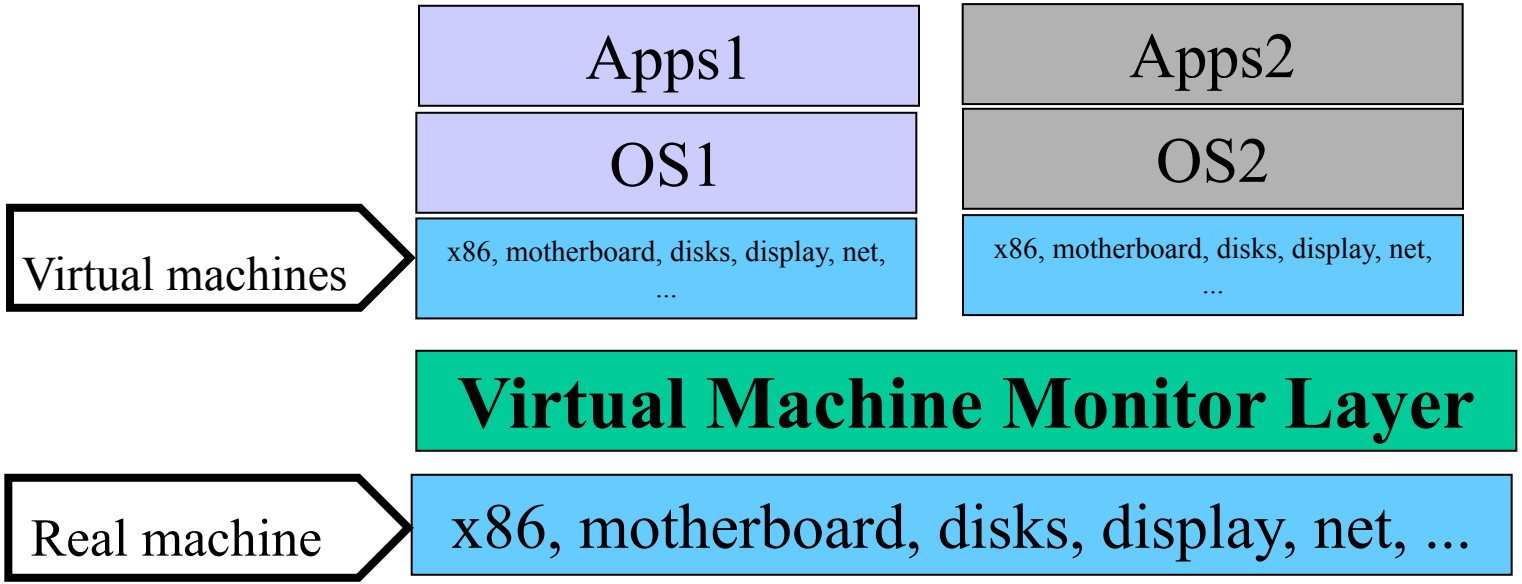
- "Process" looks like its own private machine
  - Process like this is called a **virtual machine**
  - Can run complete OS and applications in the private machine
  - Multiple virtual machines can share a single machine
- The operating system like this called a **hypervisor**
  - OS inside a virtual machine called a **guest operating system**
  - Each virtual machine could run a different guest operating systems

Virtual Machine

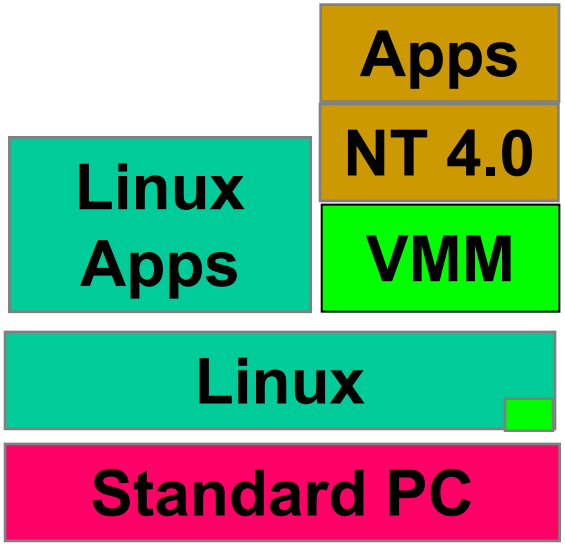


Virtual Machine

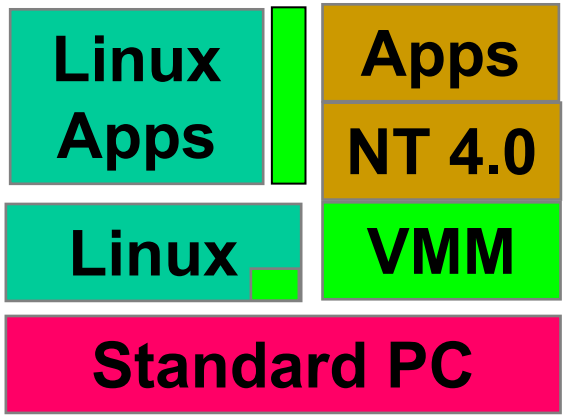
A slide from my research group 1999



Hosted Virtual Machine Monitor



Linux controls hardware



VMM controls hardware

Implementing hypervisors: One approach simulation

- Write program that simulates machine
  - Simulate CPU instruction
  - Simulate memory management unit and physical memory (e.g. large array)
  - Simulate I/O devices
- Examples:
  - Use one large file to hold contents of a "disk": a **disk image** file
  - Simulate kernel/user bit, interrupt vectors, etc.
- Problem: too slow
  - 100x slowdown for CPU/memory
  - 2x slowdown for I/O

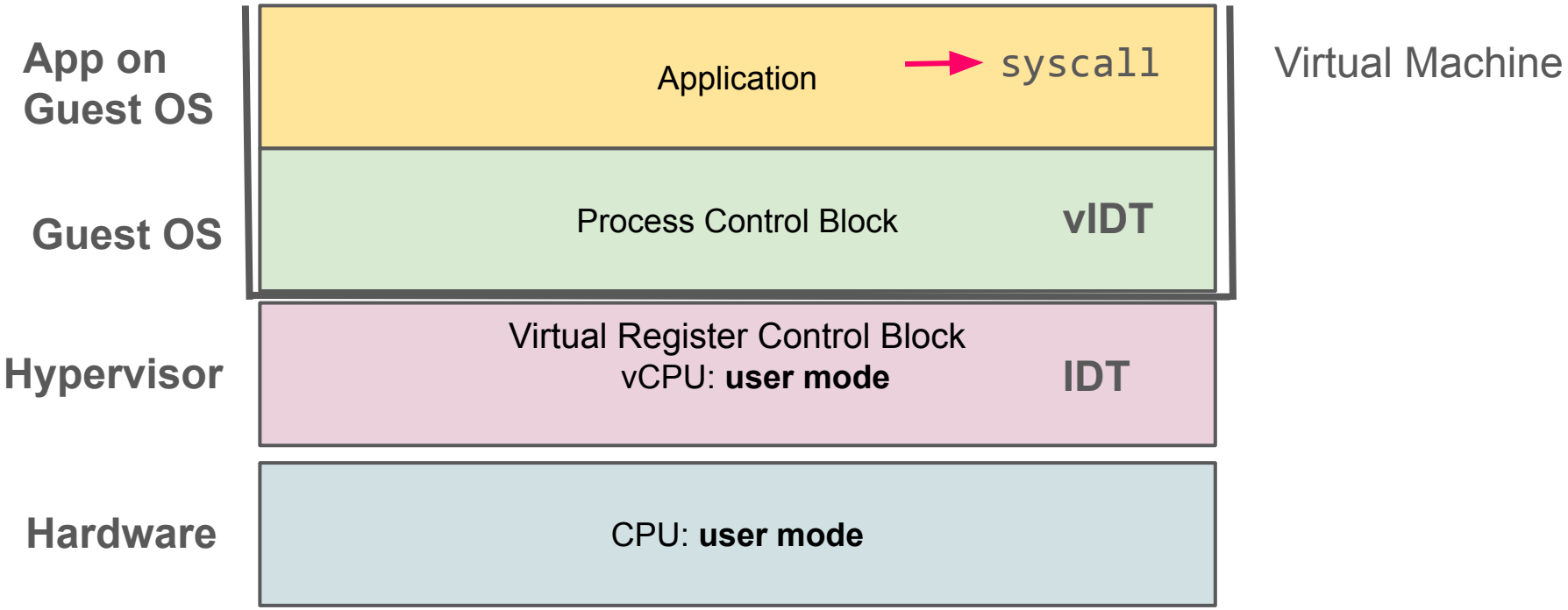
Implementing hypervisors: Better Approach

- Use CPU to simulate itself (direct execution)
  - Run guest OS in user mode
  - Most instructions execute at the full speed of the CPU
  - Privilege instructions trap into the hypervisor
    - Hypervisor has CPU simulator. Examples:, `CLI, STI, POPF, HALT`
- Privileged instructions relativity rare in kernel code
  - Simulation overheads small part of overall execution

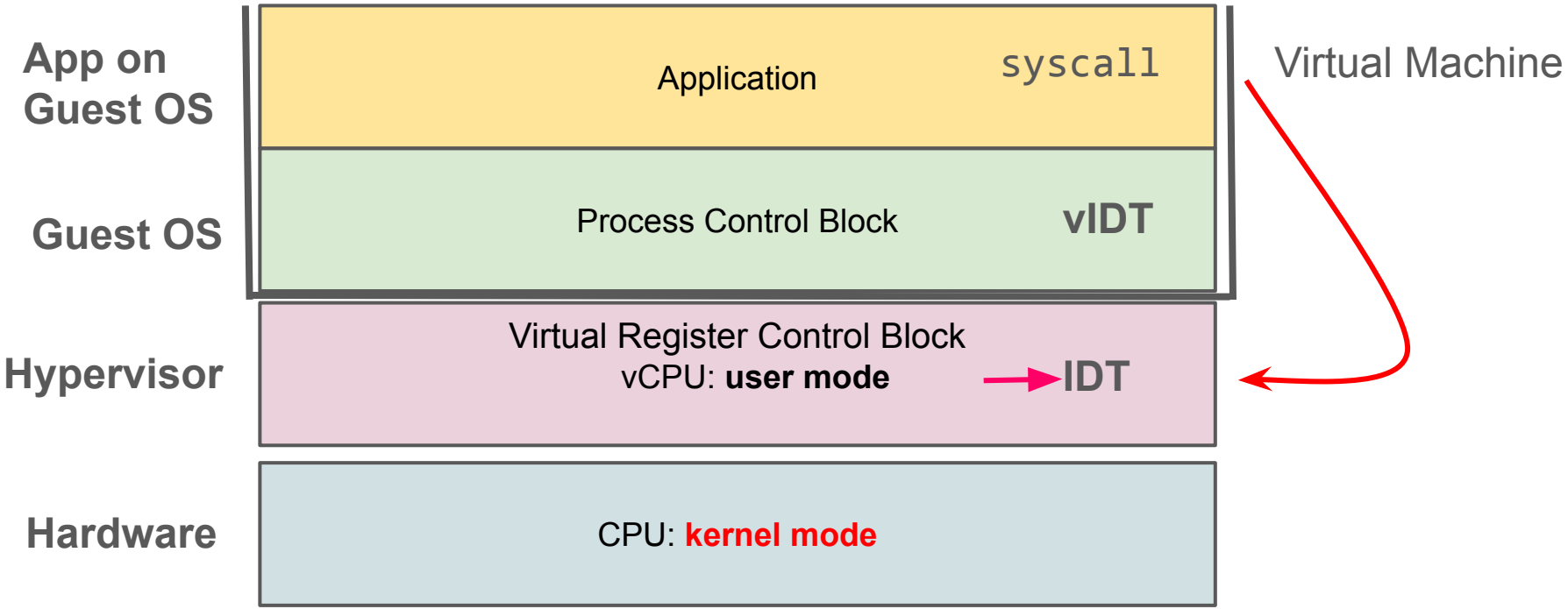
Trap and simulate example: `CLI`

- Guest OS running (CPU in user mode) encounters `CLI` instruction
  - Illegal instruction trap generated
- Hypervisor trap handler runs
  - Machine's IDT points to handler in the hypervisor
  - Hypervisor inspects trap instruction and sees the `CLI` instruction
- Hypervisor simulates `CLI`
  - Mark interrupts masked for this virtual machine
- Hypervisor returns from trap
  - CPU back to user mode at address after `CLI` instruction
  - Guest OS resumes execution now with interrupt masked

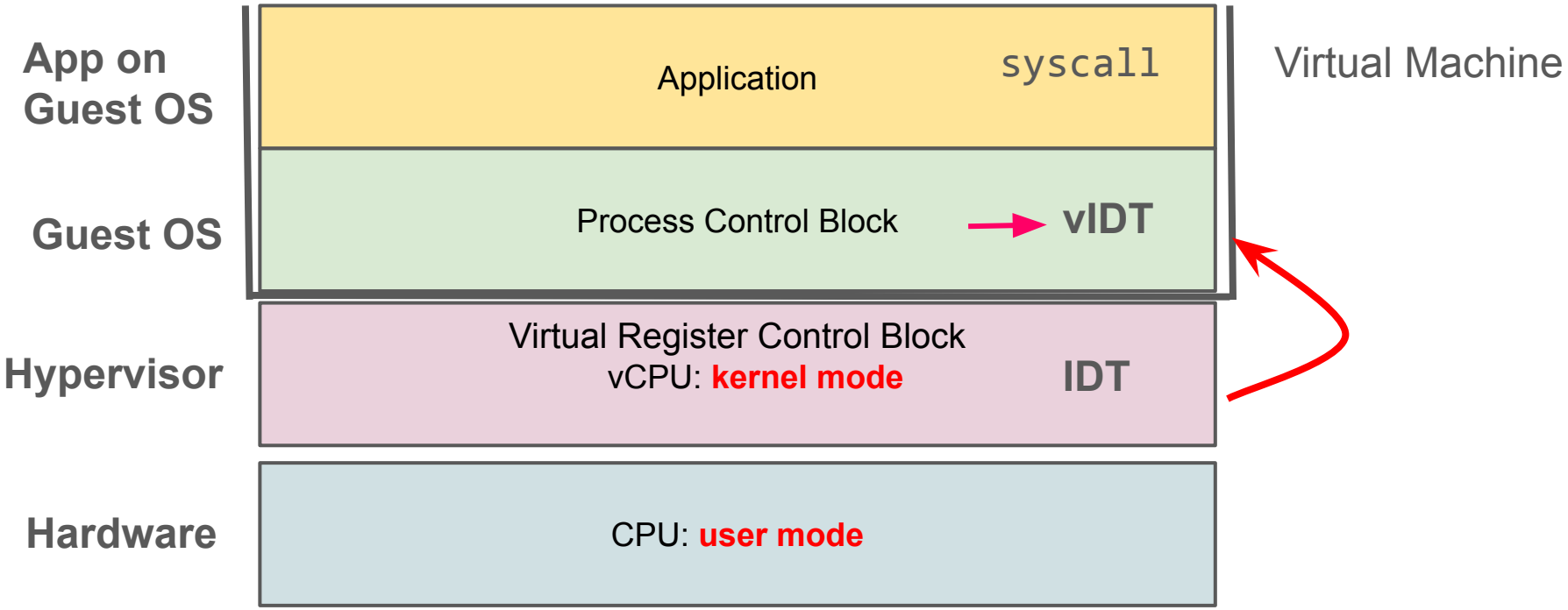
Example: Application system call in a virtual machine



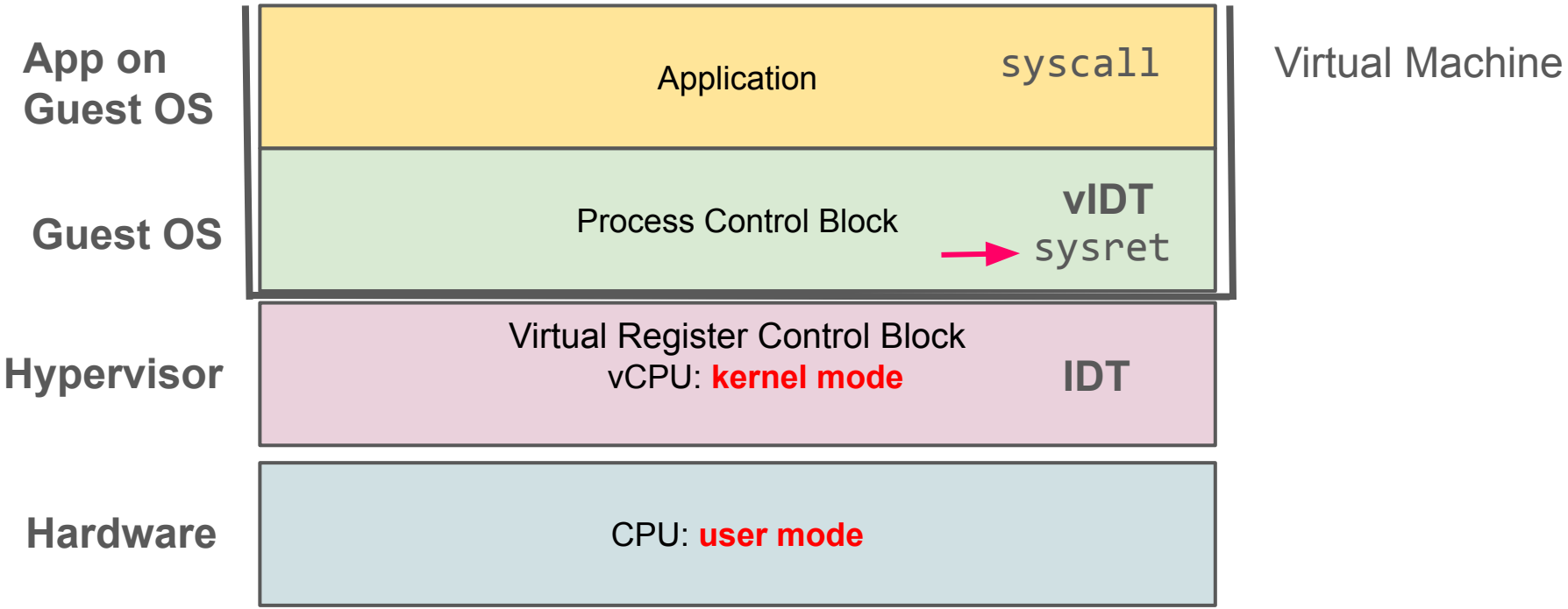
`syscall` instruction traps in to hypervisor



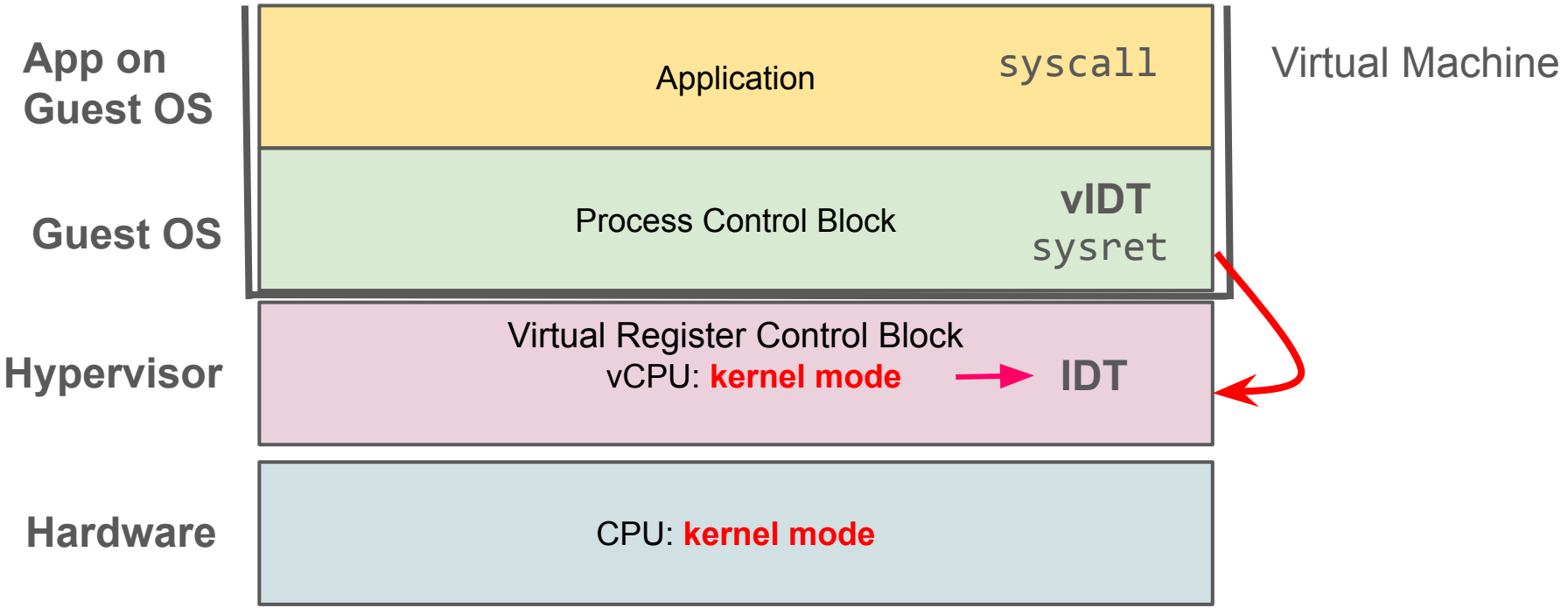
Hypervisor simulated `syscall` instruction



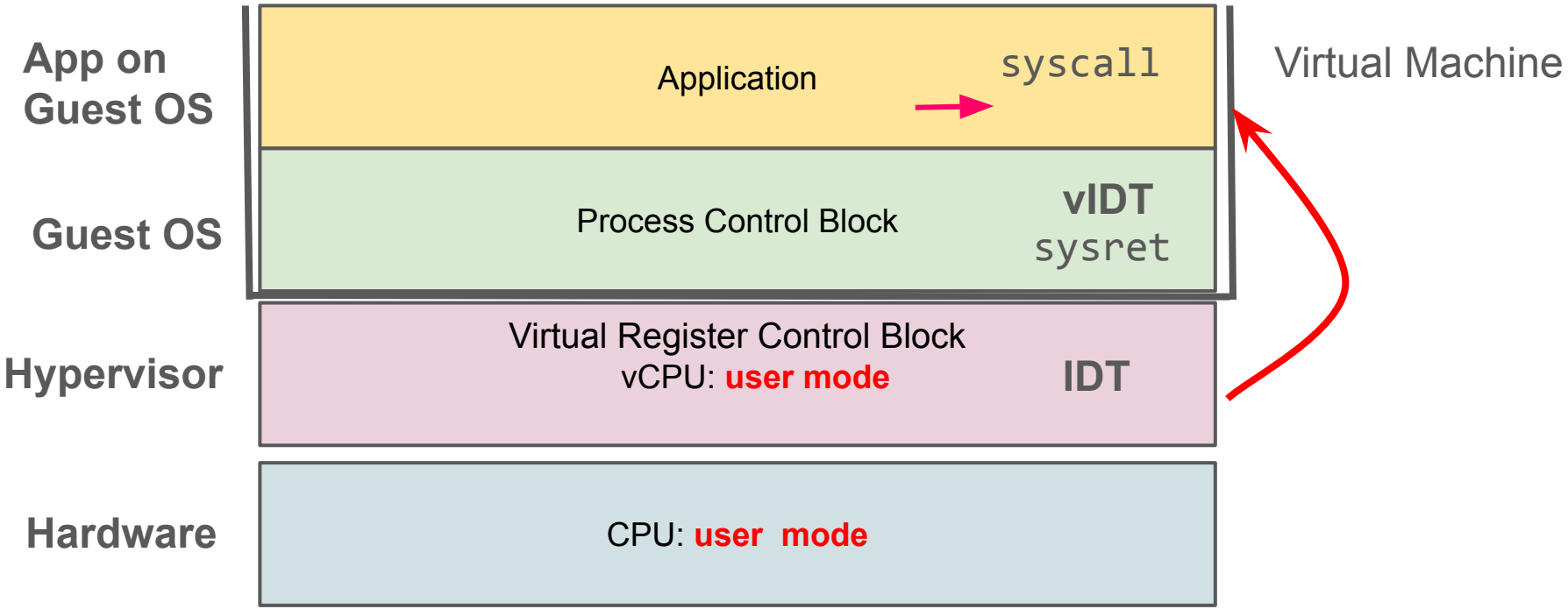
Guest OS executes syscall - ends with `sysret`



`Sysret` traps into hypervisor



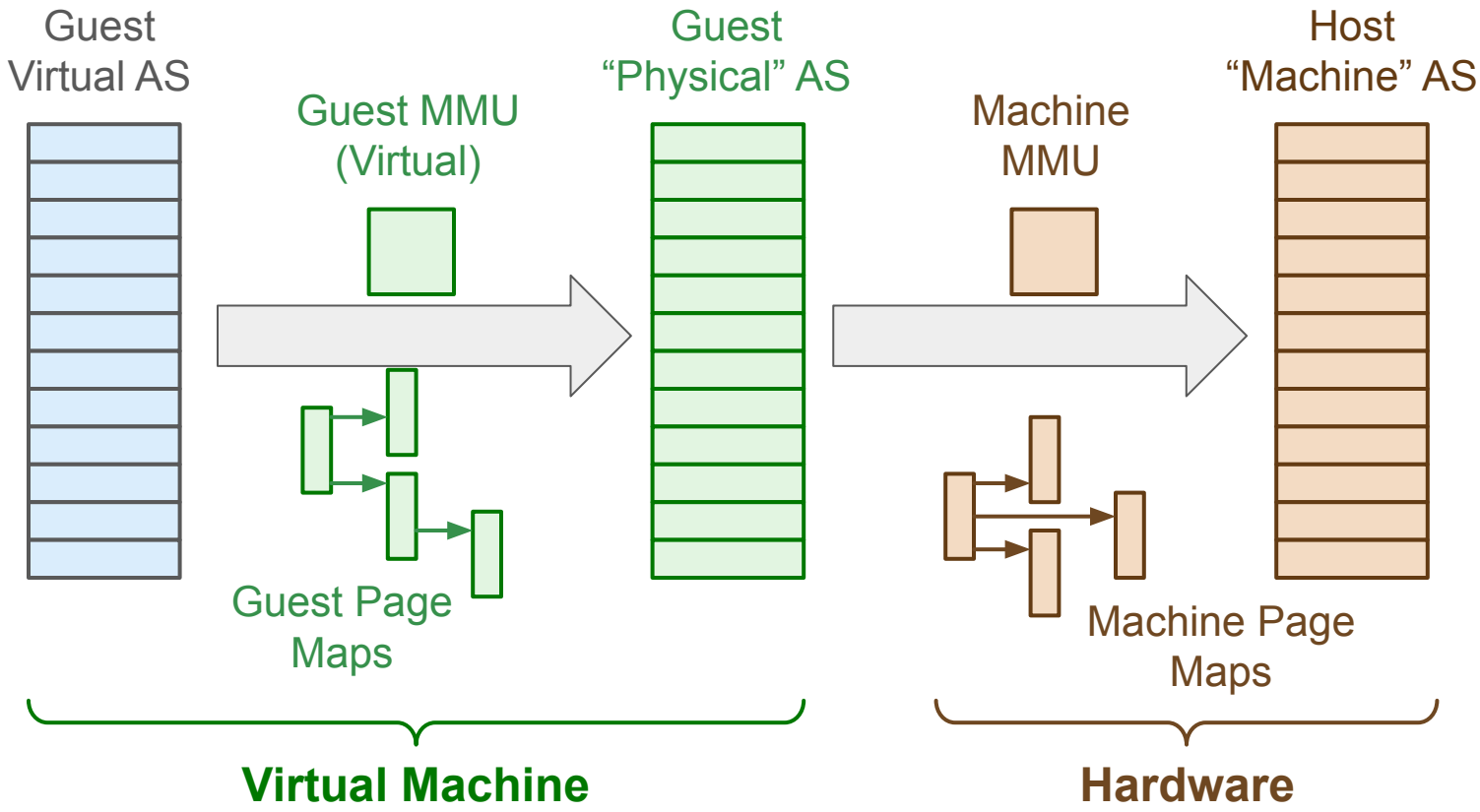
Hypervisor simulates `sysret`



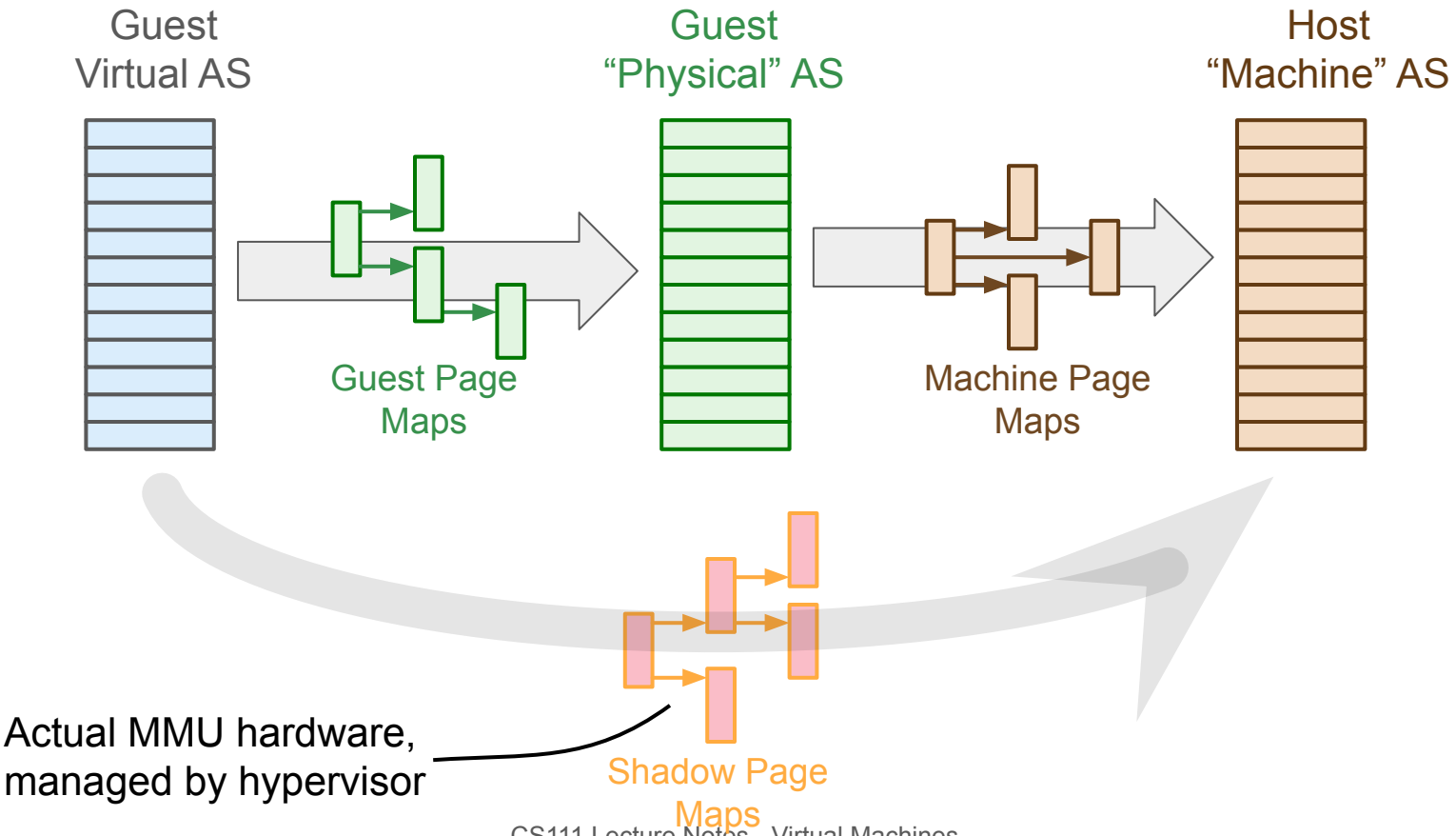
Virtual I/O Devices

- Recall OS ⇔ I/O devices: memory-mapped load/store, DMA, interrupts
- Guest OS reads/writes virtual I/O device register
  - Hypervisor arrange for traps on these device register accesses
  - Trap handlers simulate I/O device functionality
- On completion, hypervisor simulates an interrupt on the virtual CPU
- For fewer traps, write new Guest OS device drivers that use system calls
  - Called **paravirtualization**

Virtualizing Virtual Memory



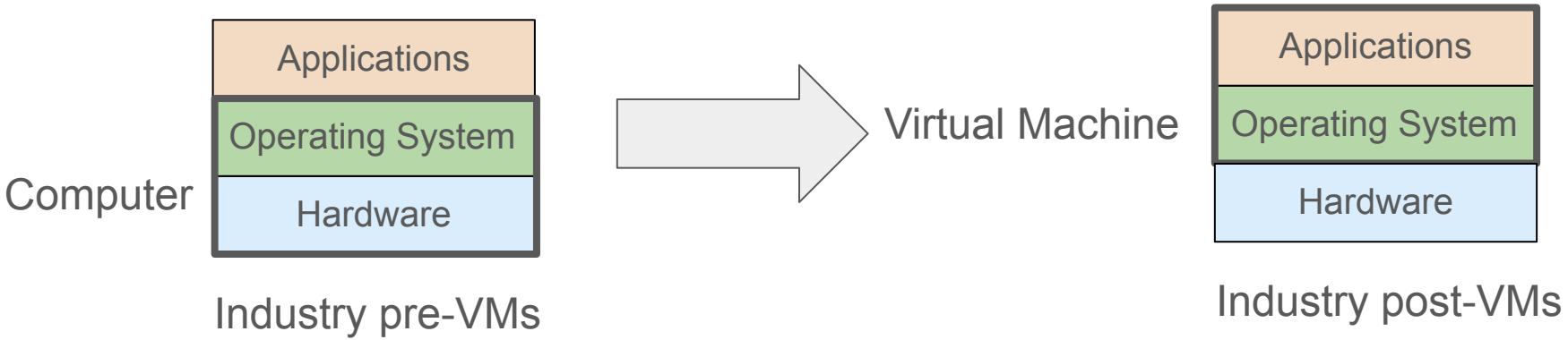
Shadow Page Maps



x86-64 memory virtualization extensions

- Maintaining shadow page tables had high overhead on some workloads
  - Too many OS updates to page tables
- Both Intel and AMD added another level of page tables for the hypervisor
  - Inside virtual machine - Page table: Virtual Page => Physical Page
  - Inside hypervisor - Page table: Physical Page => Machine Page

Virtual Machines as a resource



- Virtual machines encapsulate all execution state
  - Can duplicate, save, move around virtual machines

History of virtual machines

- Invented by IBM in late 1960's - Computers rare and expensive
  - Original usage: One VM per user - Each user ran a different single-user guest OS
- Interest waned in the 1980's and 1990's:
  - Each user had a private machine
  - Time shared operating systems better than a hypervisor with many single-user OSes
  - Disappeared from OS courses, CPU architectures quit supporting it
- Virtual machines got interesting again in mid-1990s
  - Microsoft Windows dominate position encouraged it

Virtual Machine Usage

- Software development:
  - Need to test software on different OS versions
  - Keep one VM for each OS version
  - Use a single machine to test all versions
  - Use encapsulation during testing
- Data Centers:
  - Problem: many machines, each running a single application
    - Need separate machines for isolation: application crash brings down the entire machine
    - Most applications only need a fraction of machine's resources
  - Solution: data center consolidation
    - One VM per application: Run several VM's on a single machine
    - Reduce # of machines
  - Hardware provisioned independently of software
  - Enabled Cloud Computing