CSCI 447

OPERATING SYSTEMS

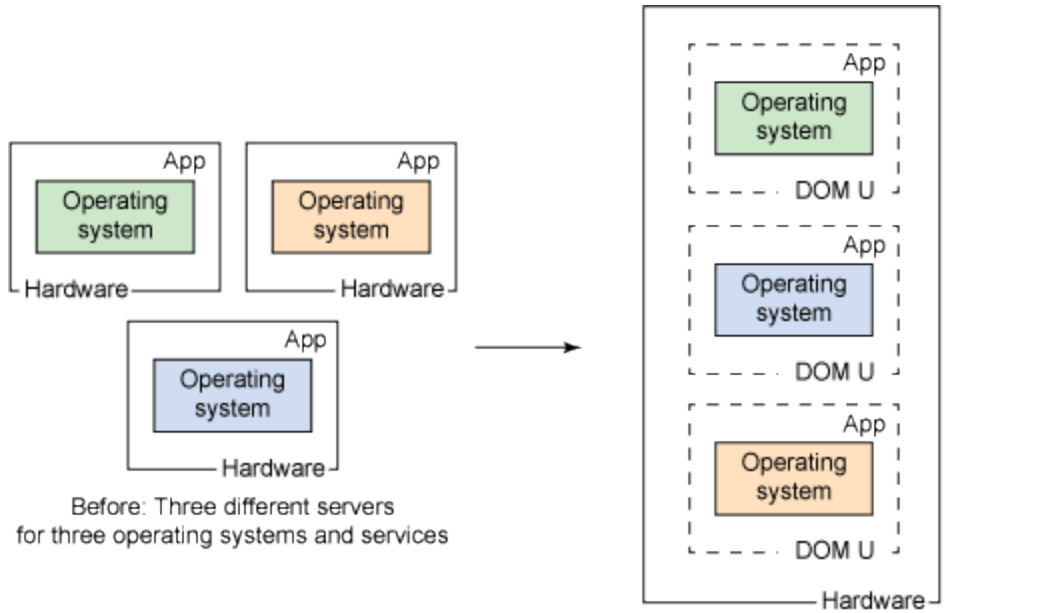
CHAPTER 18:VIRTUAL MACHINE

CHAPTER 18:VIRTUAL MACHINE

- Abstract the hardware of a single computer (the CPU, memory, disk drives, network interface cards …) into several different execution environments.
- Not very different than other abstractions/virtualization.

CHAPTER 18:VIRTUAL MACHINE

- Abstract the hardware of a single computer (the CPU, memory, disk drives, network interface cards …) into several different execution environments.
- Not very different than other abstractions/virtualization.

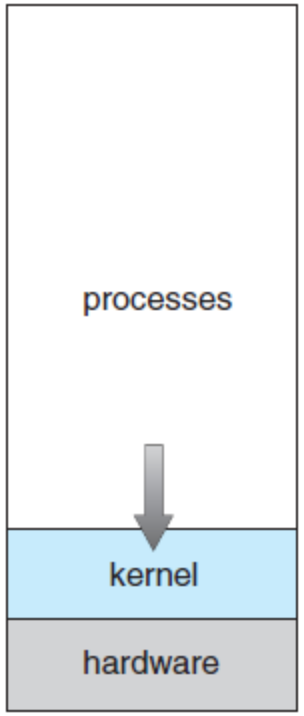


VIRTUAL MACHINE MANAGERS

- Operating Systems in Virtual Machines, run on top of Virtual Machine Mangers (VMMs) that are also called Hypervisors.

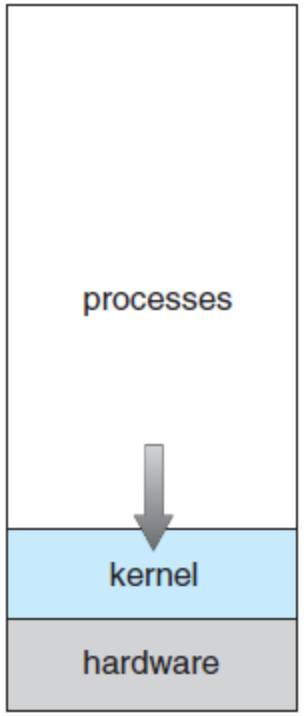
VIRTUAL MACHINE MANAGERS

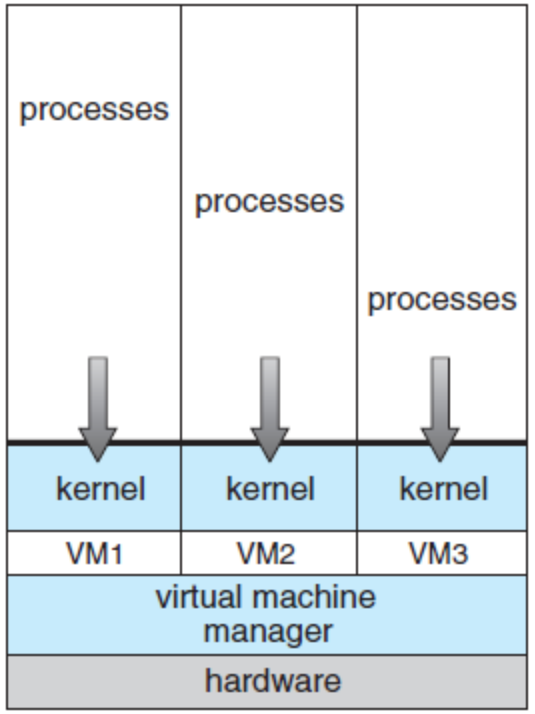
- Operating Systems in Virtual Machines, run on top of Virtual Machine Mangers (VMMs) that are also called Hypervisors.



VIRTUAL MACHINE MANAGERS

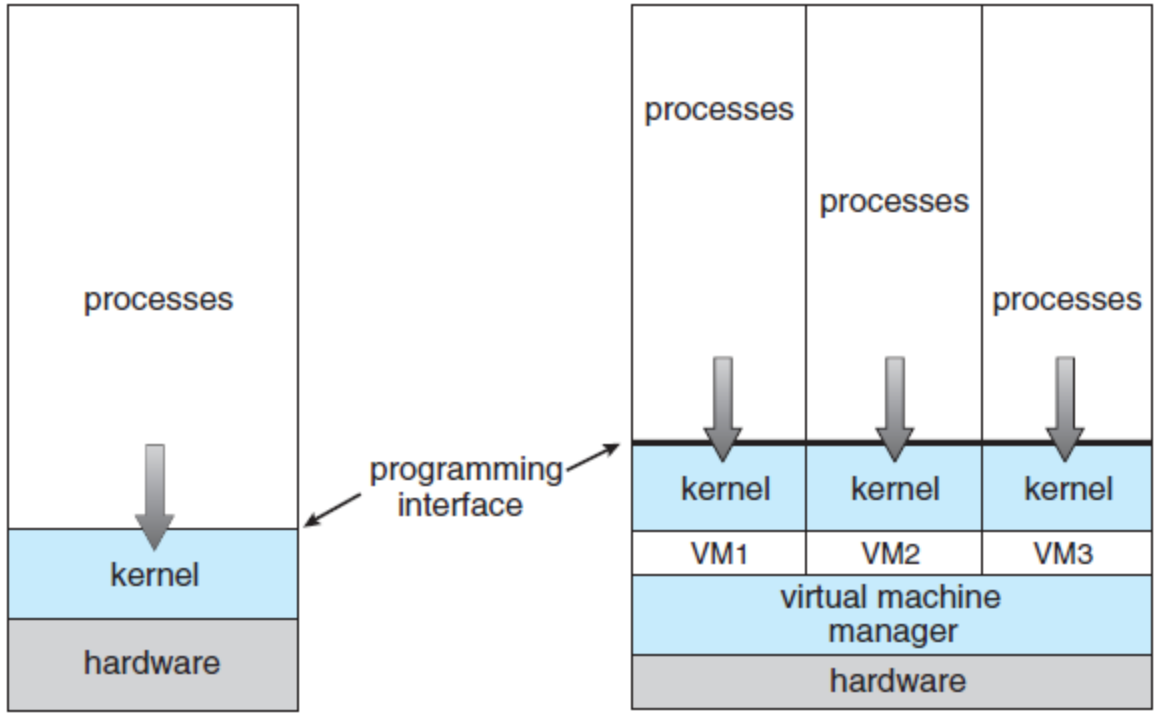
- Operating Systems in Virtual Machines, run on top of Virtual Machine Mangers (VMMs) that are also called Hypervisors.





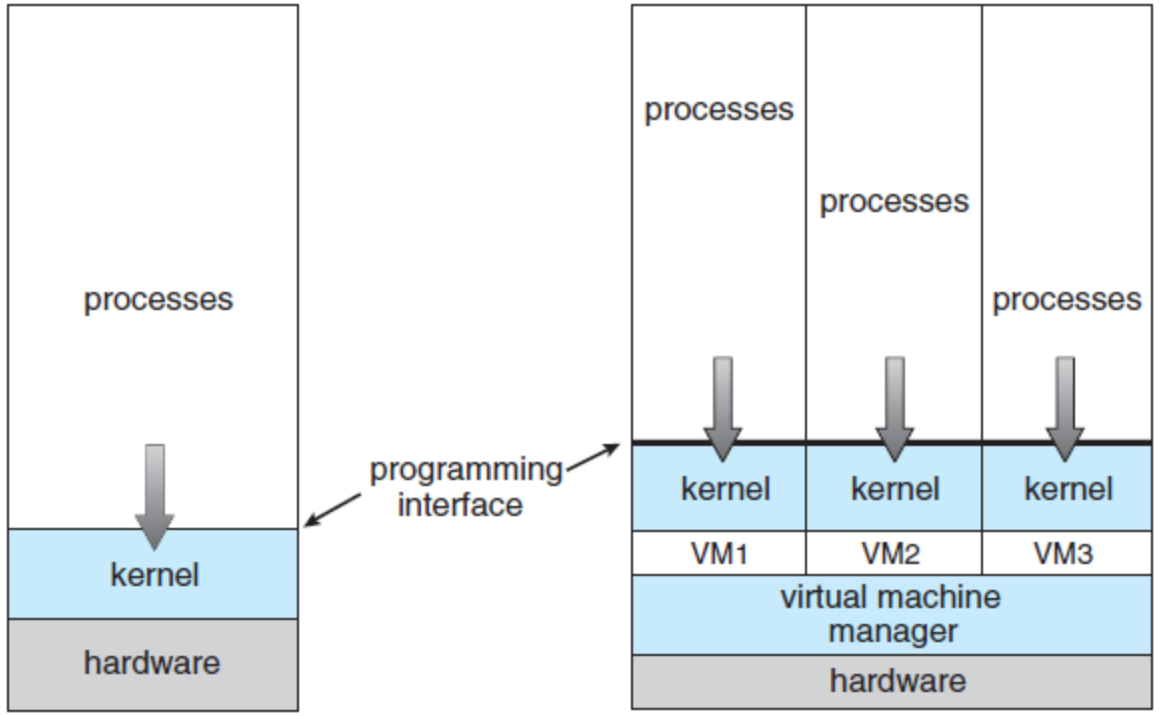
VIRTUAL MACHINE MANAGERS

- Operating Systems in Virtual Machines, run on top of Virtual Machine Mangers (VMMs) that are also called Hypervisors.



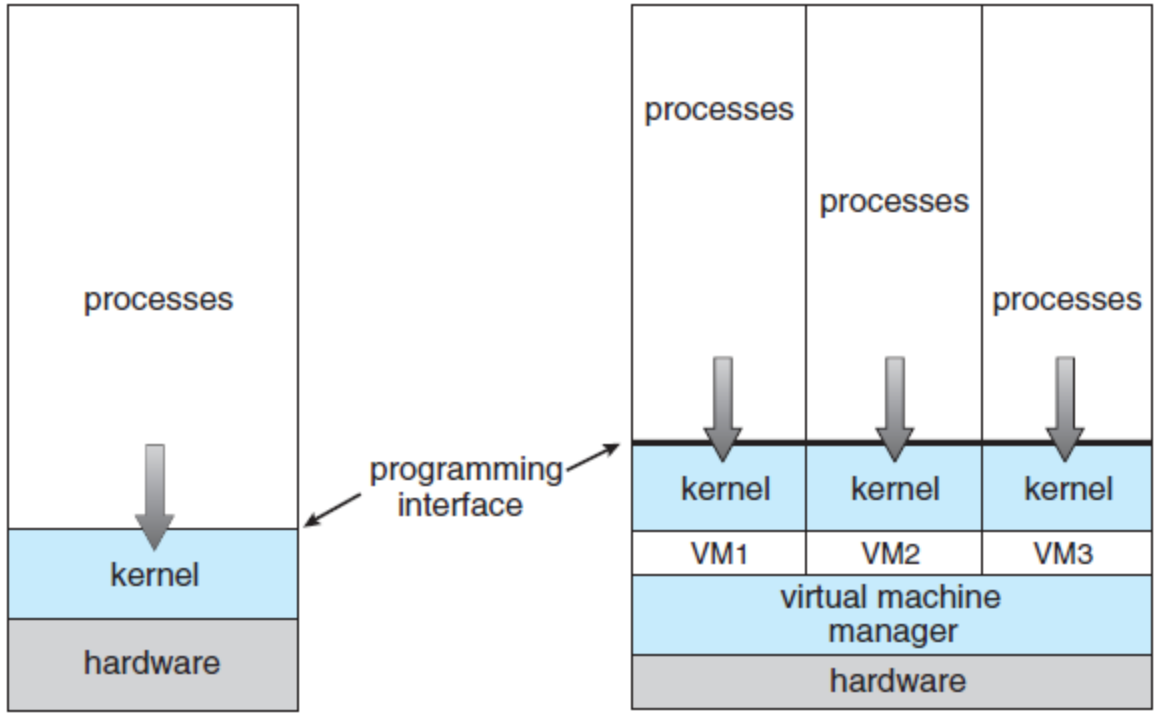
VIRTUAL MACHINE MANAGERS

- Operating Systems in Virtual Machines, run on top of Virtual Machine Mangers (VMMs) that are also called Hypervisors.
- Hypervisors provide the illusion of hardware to the OS. The interface would be identical to that of real hardware system.



VIRTUAL MACHINE MANAGERS

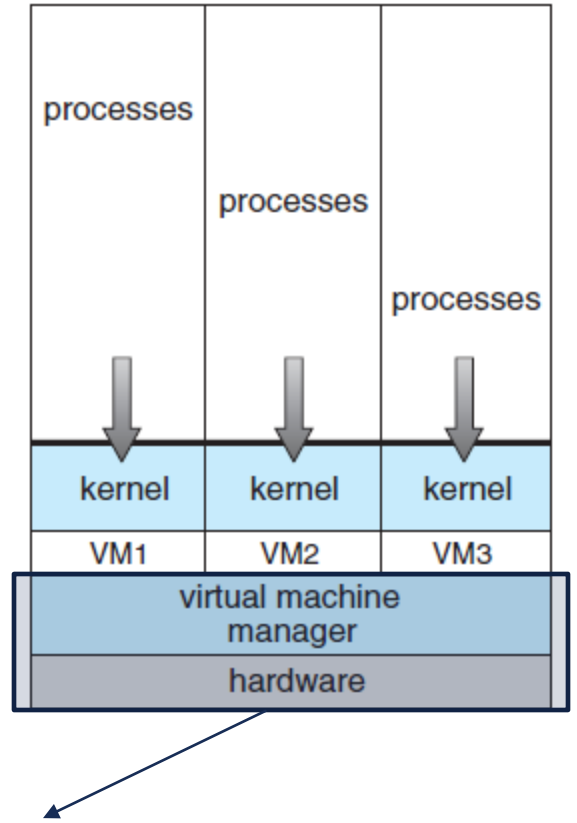
- Operating Systems in Virtual Machines, run on top of Virtual Machine Mangers (VMMs) that are also called Hypervisors.
- Hypervisors provide the illusion of hardware to the OS. The interface would be identical to that of real hardware system.
- The machine running the VMM is called the "Host" machine.
- The emulated virtual machine is called the "guest machine".



VIRTUAL MACHINE MANAGERS

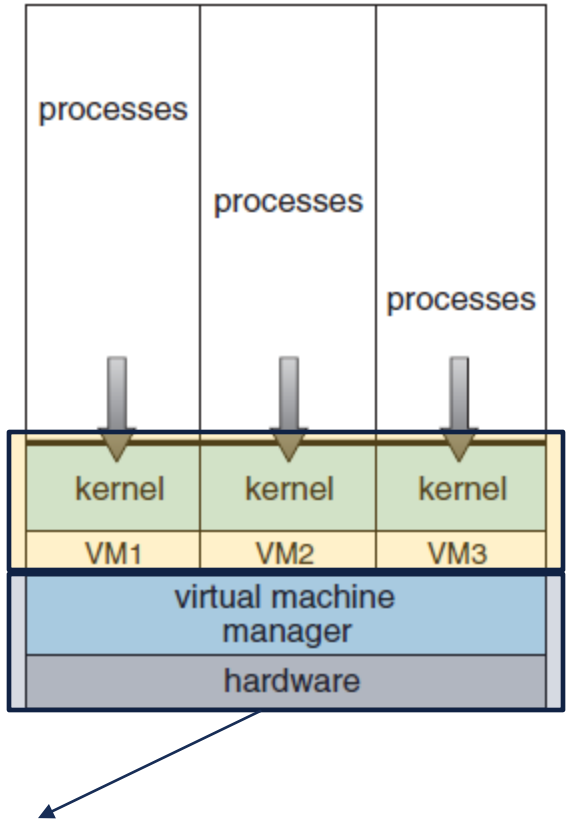
- Operating Systems in Virtual Machines, run on top of Virtual Machine Mangers (VMMs) that are also called Hypervisors.
- Hypervisors provide the illusion of hardware to the OS. The interface would be identical to that of real hardware system.
- The machine running the VMM is called the "Host" machine.
- The emulated virtual machine is called the "guest machine".

A: Guest Machine
B: Host Machine



VIRTUAL MACHINE MANAGERS

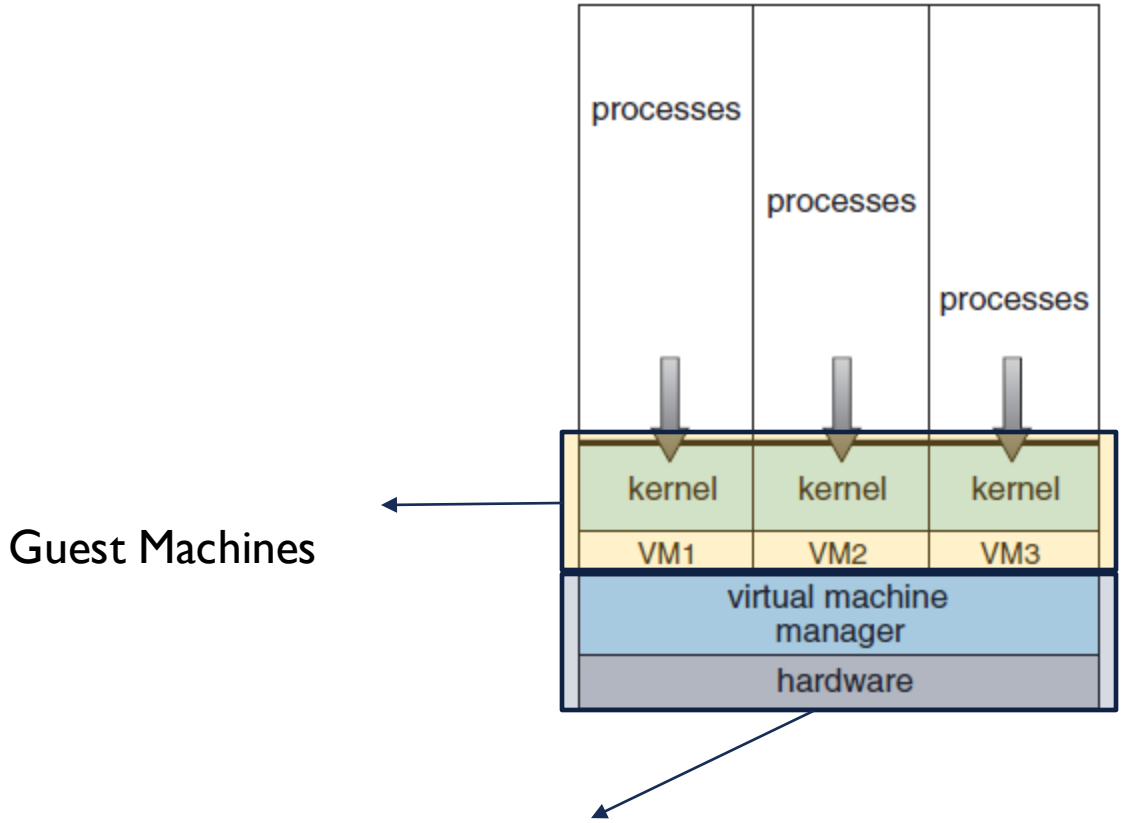
- Operating Systems in Virtual Machines, run on top of Virtual Machine Mangers (VMMs) that are also called Hypervisors.
- Hypervisors provide the illusion of hardware to the OS. The interface would be identical to that of real hardware system.
- The machine running the VMM is called the "Host" machine.
- The emulated virtual machine is called the "guest machine".



Host Machine

VIRTUAL MACHINE MANAGERS

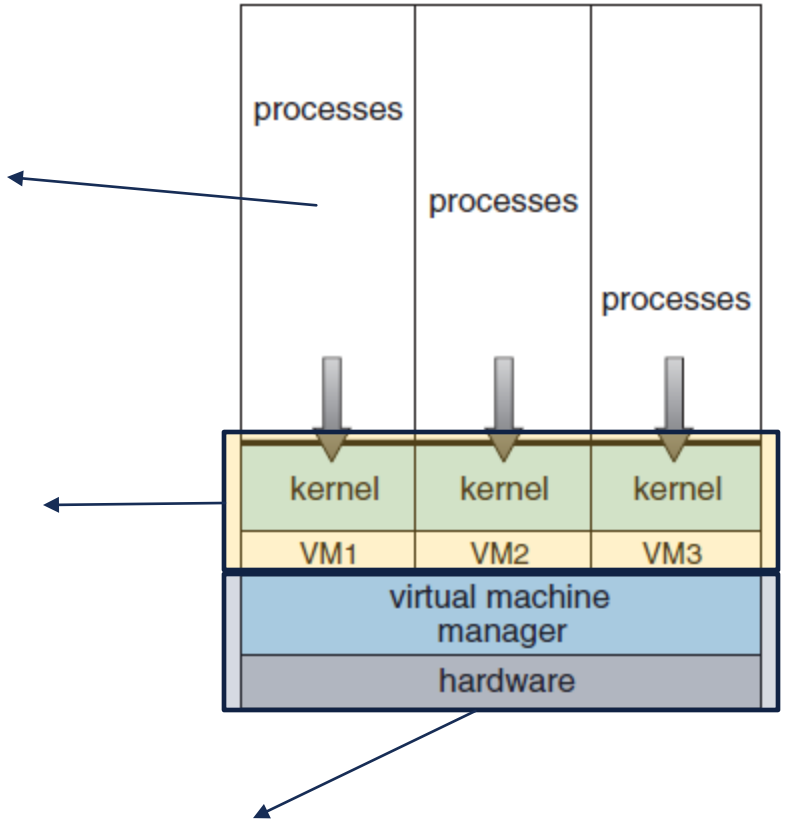
- Operating Systems in Virtual Machines, run on top of Virtual Machine Mangers (VMMs) that are also called Hypervisors.
- Hypervisors provide the illusion of hardware to the OS. The interface would be identical to that of real hardware system.
- The machine running the VMM is called the "Host" machine.
- The emulated virtual machine is called the "guest machine".



VIRTUAL MACHINE MANAGERS

- Operating Systems in Virtual Machines, run on top of Virtual Machine Mangers (VMMs) that are also called Hypervisors.
- Hypervisors provide the illusion of hardware to the OS. The interface would be identical to that of real hardware system.
- The machine running the VMM is called the "Host" machine.
- The emulated virtual machine is called the "guest machine".

A: Guest Process
B: Host Process

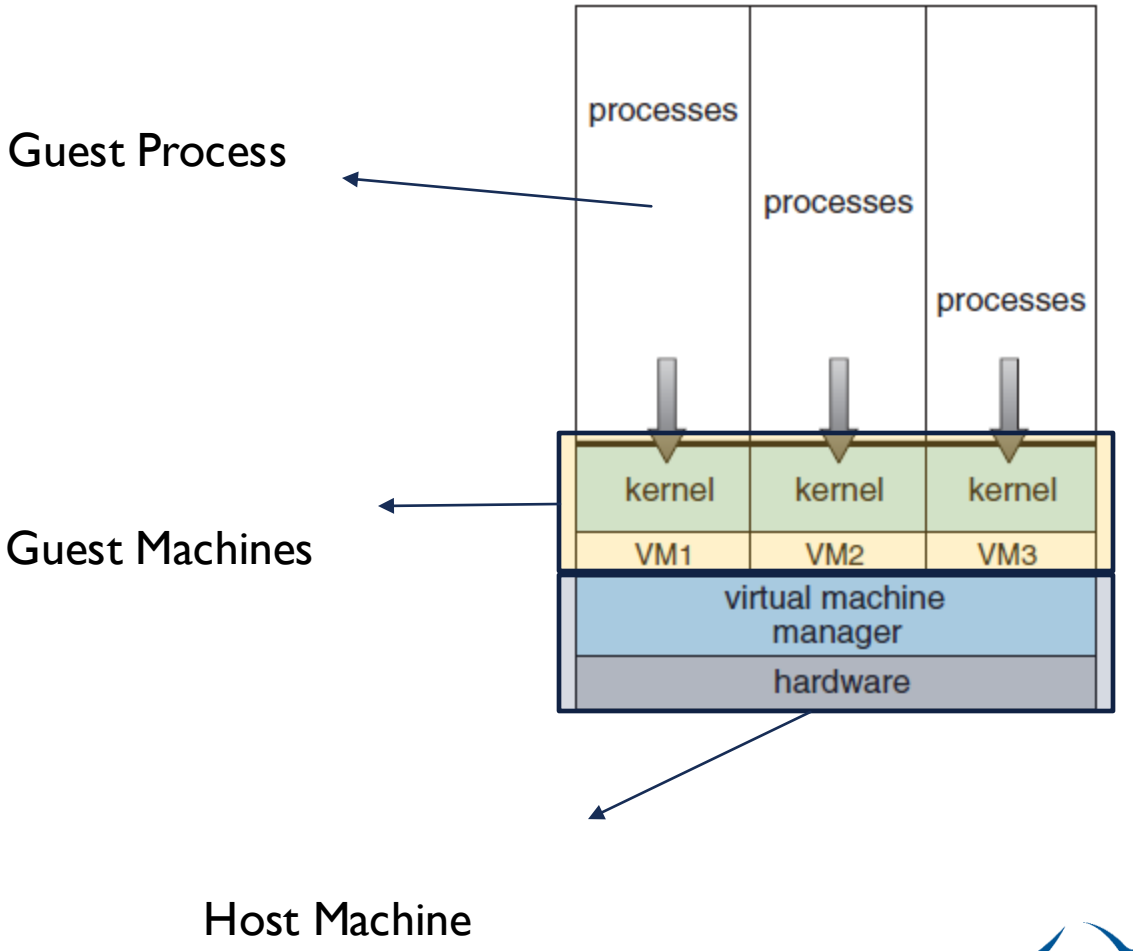


Guest Machines

Host Machine

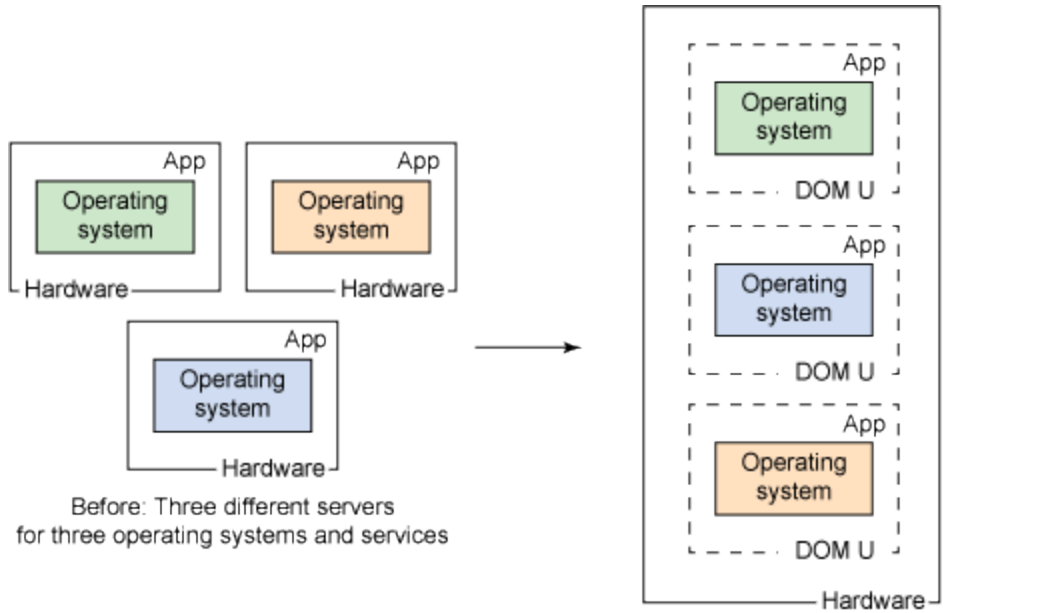
VIRTUAL MACHINE MANAGERS

- Operating Systems in Virtual Machines, run on top of Virtual Machine Mangers (VMMs) that are also called Hypervisors.
- Hypervisors provide the illusion of hardware to the OS. The interface would be identical to that of real hardware system.
- The machine running the VMM is called the "Host" machine.
- The emulated virtual machine is called the "guest machine".



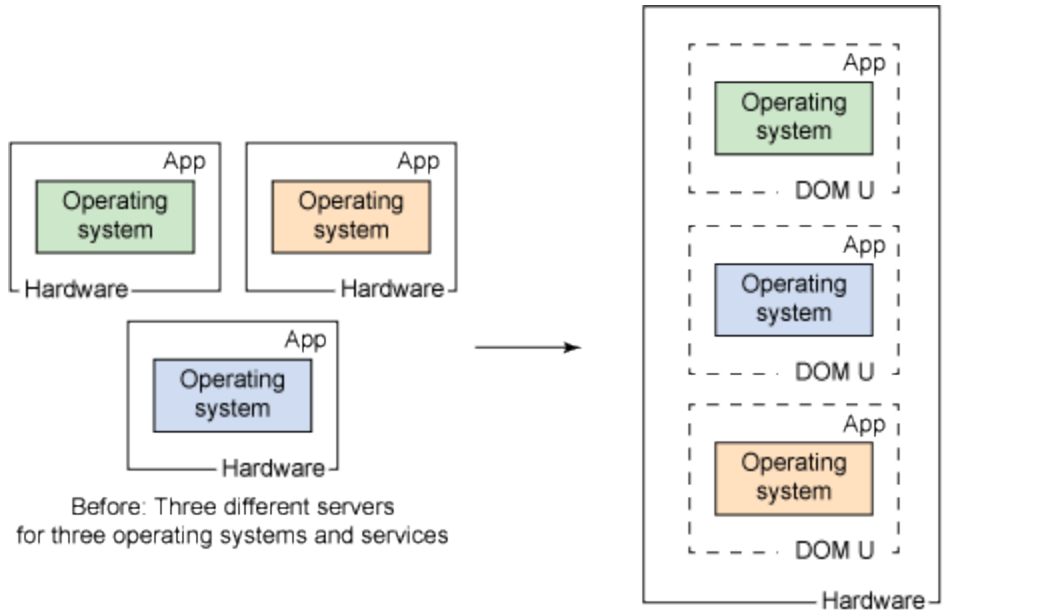
VIRTUAL MACHINE HISTORY

- Virtual machines first appeared commercially on IBM mainframes in 1972.
- What was the main challenge?



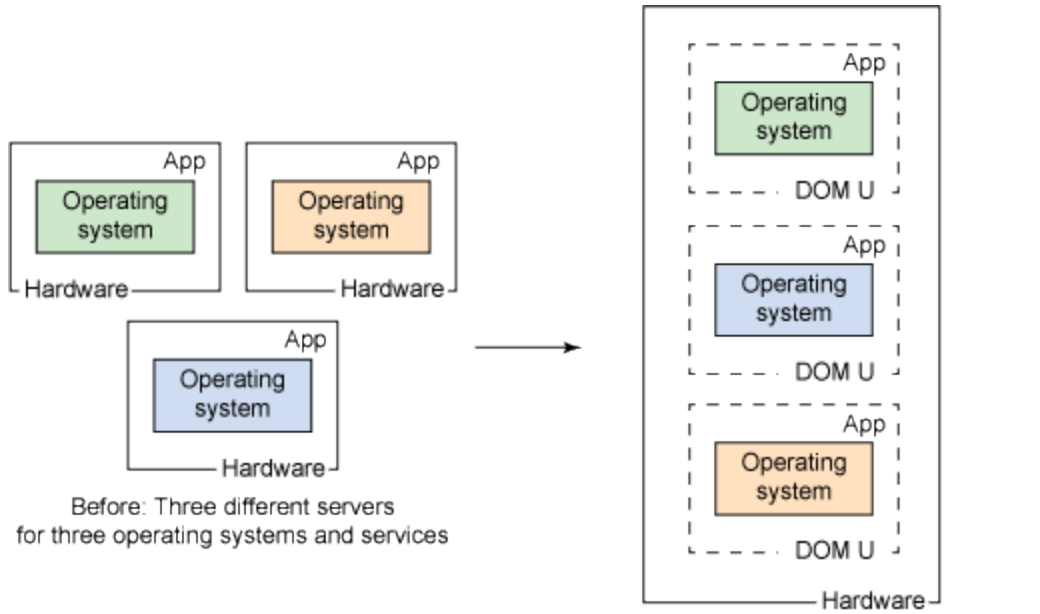
VIRTUAL MACHINE HISTORY

- Virtual machines first appeared commercially on IBM mainframes in 1972.
- What was the main challenge?
- Sharing CPU and Memory among different OS was relatively simple.
- We already do that among processes.



VIRTUAL MACHINE HISTORY

- Virtual machines first appeared commercially on IBM mainframes in 1972.
- What was the main challenge?
- Sharing CPU and Memory among different OS was relatively simple.
- We already do that among processes.
- **Sharing the disk** was a particular challenge.

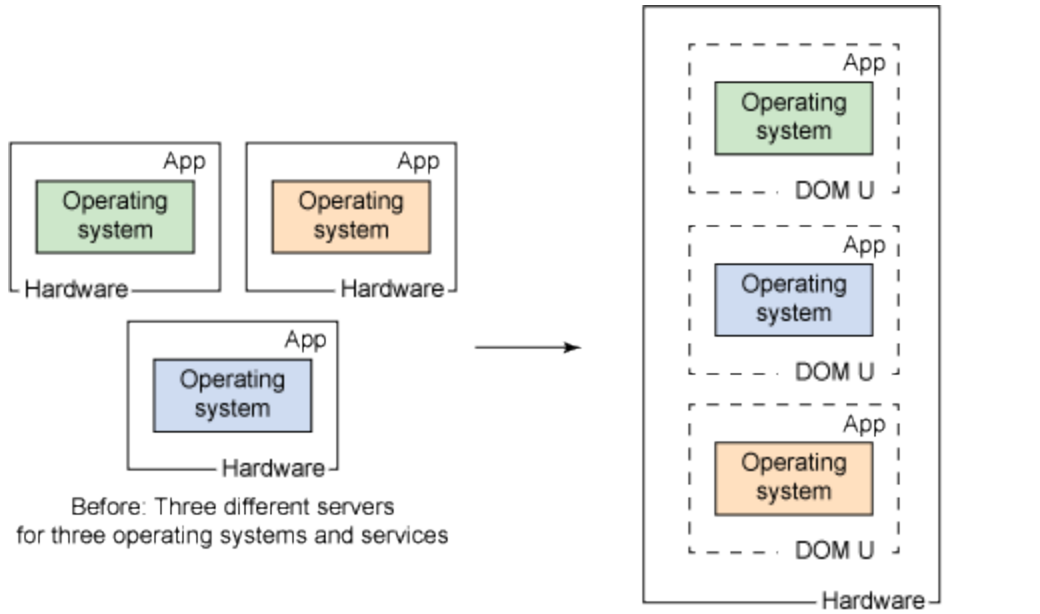


Before: Three different servers for three operating systems and services

After: Only one server required for three different servers and operating systems

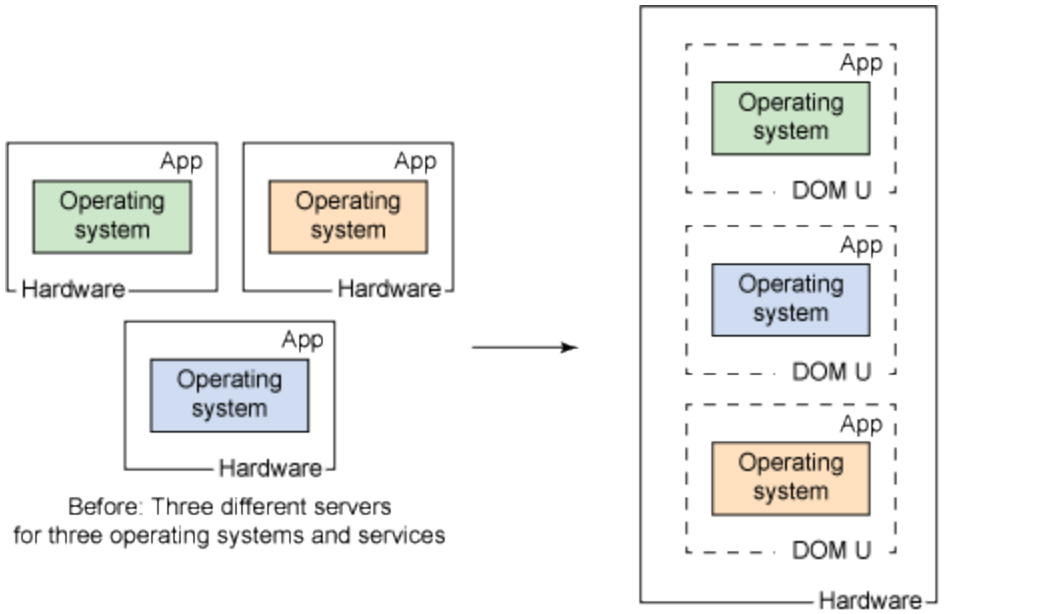
VIRTUAL MACHINE HISTORY

- Virtual machines first appeared commercially on IBM mainframes in 1972.
- What was the main challenge?
- Sharing CPU and Memory among different OS was relatively simple.
- We already do that among processes.
- **Sharing the disk** was a particular challenge.
- Disks were split into "mini disks" that used a portion of the tracks available for each system. No emulation, each logical tack corresponded to a complete physical track.



VIRTUAL MACHINE HISTORY

- Virtual machines first appeared commercially on IBM mainframes in 1972.
- What was the main challenge?
- Sharing CPU and Memory among different OS was relatively simple.
- We already do that among processes.
- **Sharing the disk** was a particular challenge.
- Disks were split into "mini disks" that used a portion of the tracks available for each system. No emulation, each logical tack corresponded to a complete physical track.
- Why tracks? Faster access, you don't want two OS sharing the same track.



Before: Three different servers for three operating systems and services

After: Only one server required for three different servers and operating systems

VIRTUAL MACHINE HISTORY

- In earlier days, VMs were restricted to servers and data centers.
  - Required high end computing.

VIRTUAL MACHINE HISTORY

- In earlier days, VMs were restricted to servers and data centers.
  - Required high end computing.
- In 1990s, general purpose processors became fast enough to support virtualization.
  - **Xen** and **VMware** created technologies, still used today
  - Virtualization has expanded to many OSes, CPUs, VMMs

VM BENEFITS

VM BENEFITS

- Security: VMs protected from each other
  - I.e. A virus less likely to spread
  - Host protected from VM
  - Sharing is provided though via shared file system volume, network communication

VM BENEFITS

- Security: VMs protected from each other
  - I.e. A virus less likely to spread
  - Host protected from VM
  - Sharing is provided though via shared file system volume, network communication
- Freeze, **suspend**, running VM
  - Then can move or copy somewhere else and **resume**
  - Snapshot of a given state, able to restore back to that state

VM BENEFITS

- Security: VMs protected from each other
  - I.e. A virus less likely to spread
  - Host protected from VM
  - Sharing is provided though via shared file system volume, network communication
- Freeze, **suspend**, running VM
  - Then can move or copy somewhere else and **resume**
  - Snapshot of a given state, able to restore back to that state
- Run multiple, different OSes on a single machine

VM BENEFITS

- Security: VMs protected from each other
  - I.e. A virus less likely to spread
  - Host protected from VM
  - Sharing is provided though via shared file system volume, network communication
- Freeze, **suspend**, running VM
  - Then can move or copy somewhere else and **resume**
  - Snapshot of a given state, able to restore back to that state
- Run multiple, different OSes on a single machine
- **Cloud computing**
  - Using APIs, programs tell cloud infrastructure (servers, networking, storage) to create new guests, VMs, virtual desktops.
  - More of a "product" of virtual machines.

VM REQUIREMENTS

What are the goals of a Virtual Machine Manager?

VM REQUIREMENTS

What are the goals of a Virtual Machine Manager?

- **Fidelity**. VMM provides an environment for programs that is essentially identical to the original machine.

VM REQUIREMENTS

What are the goals of a Virtual Machine Manager?

- **Fidelity**. VMM provides an environment for programs that is essentially identical to the original machine.
- **Performance**. Programs running within that environment show only minor performance decreases.

VM REQUIREMENTS

What are the goals of a Virtual Machine Manager?

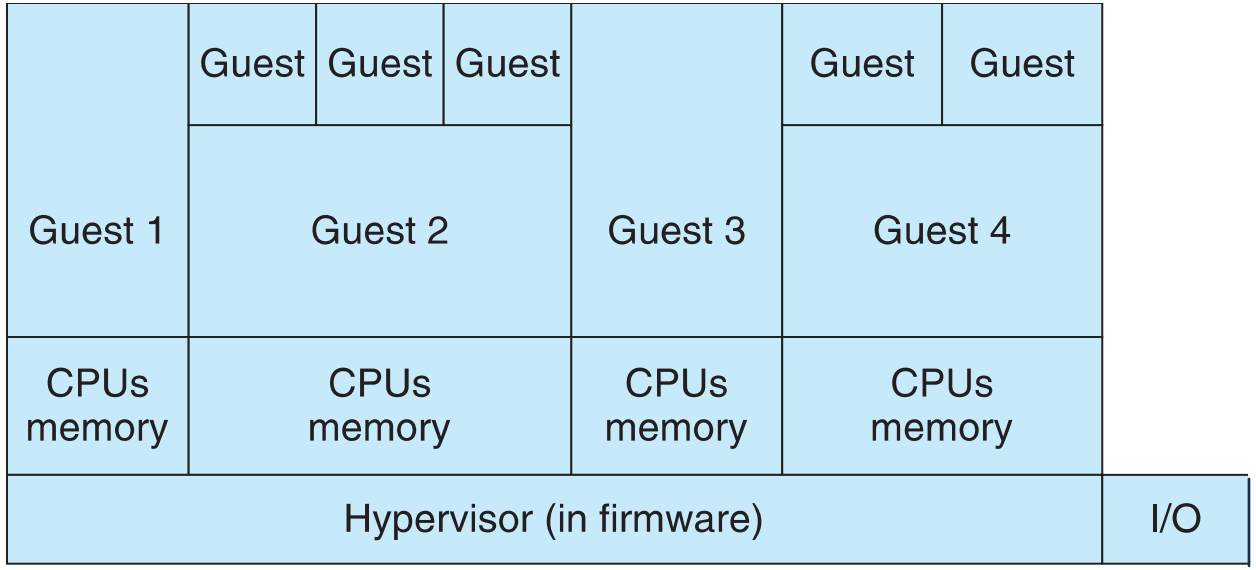
- **Fidelity**. VMM provides an environment for programs that is essentially identical to the original machine.
- **Performance**. Programs running within that environment show only minor performance decreases.
- **Safety**. The VMM is in complete control of system resources. Guest can only control resources through the VMM.

HYPERVISORS TYPES

- Type 0, I and 2

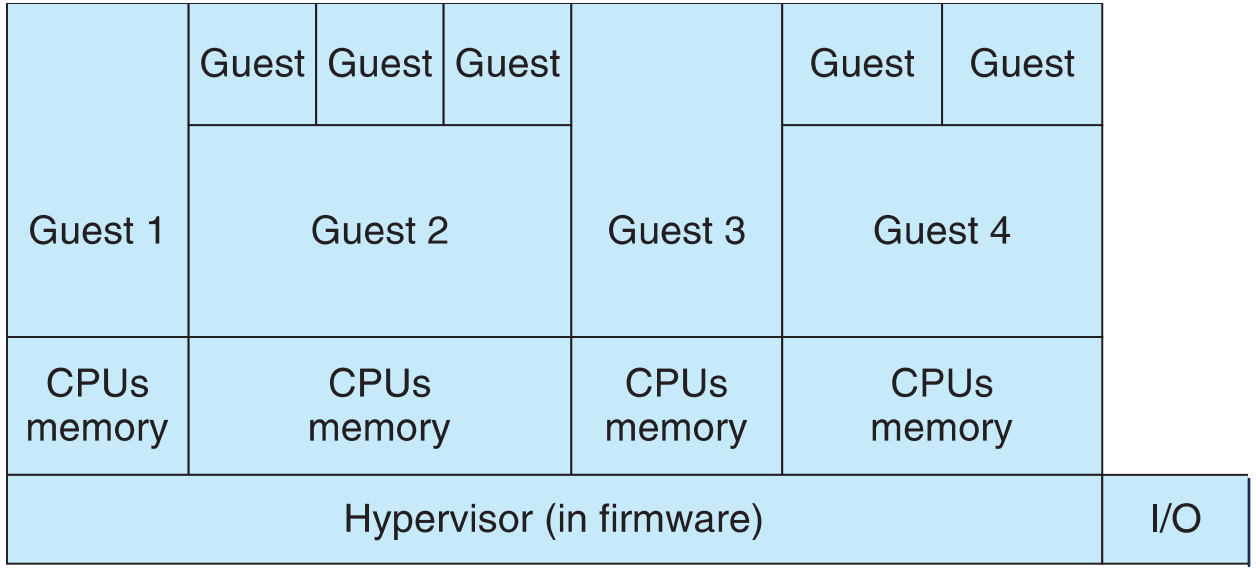
HYPERVISORS TYPES

- Type 0:
  - A HW feature implemented by firmware
  - Each guest has dedicated hardware.
  - OS needs nothing special, VMM is in firmware
  - Smaller feature set than other types



HYPERVISORS TYPES

- Type 0:
  - A HW feature implemented by firmware
  - Each guest has dedicated hardware.
  - OS needs nothing special, VMM is in firmware
  - Smaller feature set than other types

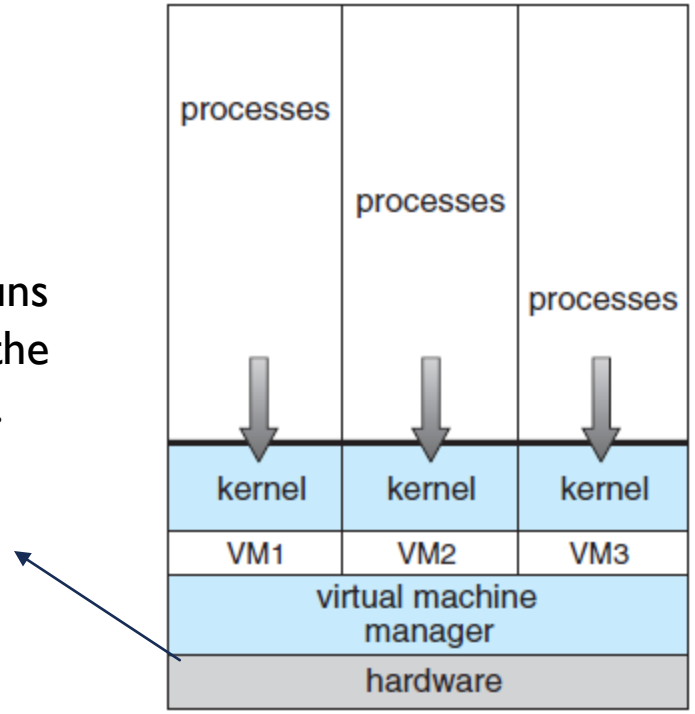


Status as a "Hypervisor" is questionable, since there is no real, virtualization. Each "guest" has its own hardware.

HYPERVISOR TYPE 1

- Type 1:
- Special purpose operating systems that run natively on HW

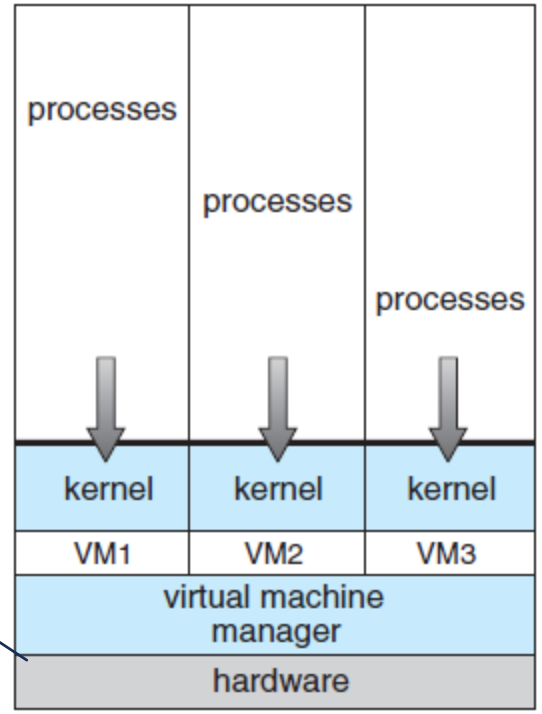
The VMM runs directly on the hardware.



HYPERVISOR TYPE 1

- Type 1:
- Special purpose operating systems that run natively on HW

The VMM runs directly on the hardware.

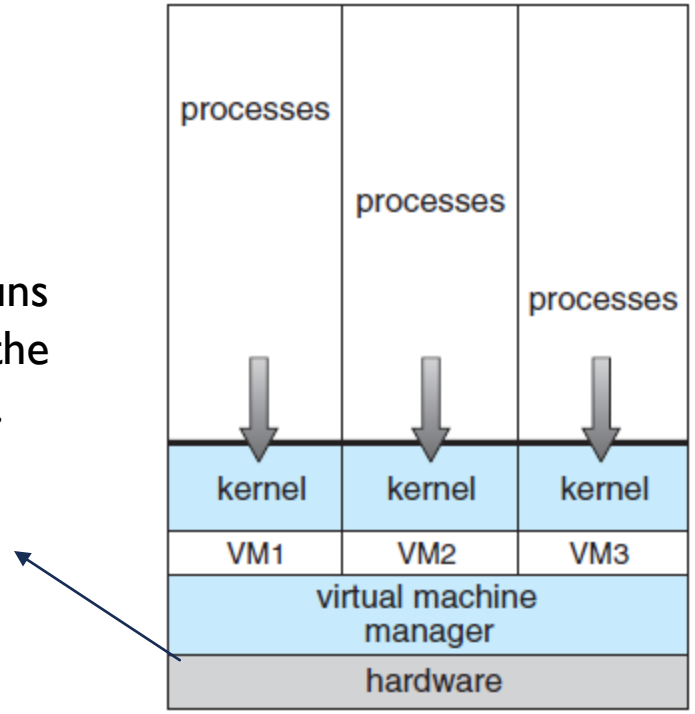


The hypervisor is the OS

HYPERVISOR TYPE 1

- Type 1:
- Special purpose operating systems that run natively on HW
  - Rather than providing system call interface, create run and manage guest OSes
  - Can run on Type 0 hypervisors but not on other Type 1s
  - Run in kernel mode
  - Guests generally don't know they are running in a VM

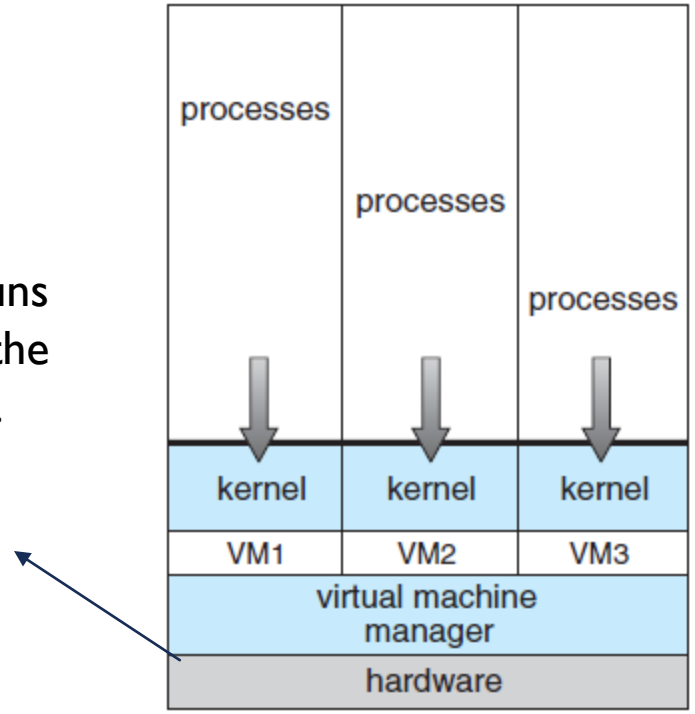
The VMM runs directly on the hardware.



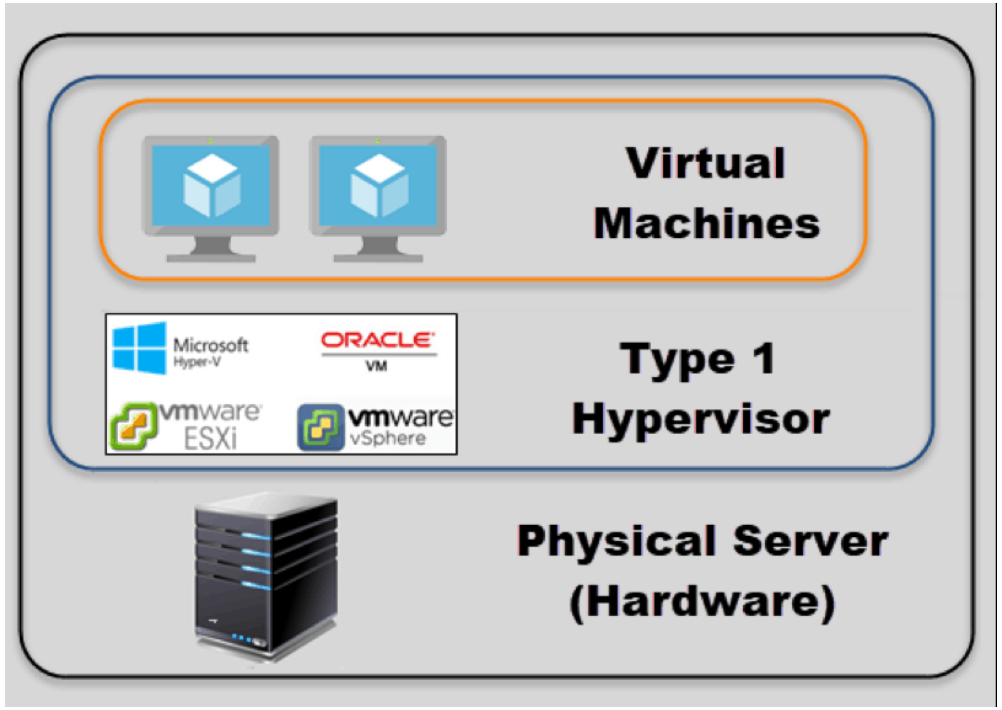
HYPERVISOR TYPE 1

- Type 1:
- Special purpose operating systems that run natively on HW
  - Rather than providing system call interface, create run and manage guest OSes
  - Can run on Type 0 hypervisors but not on other Type 1s
  - Run in kernel mode
  - Guests generally don't know they are running in a VM
- More complex to implement
  - Implement device drivers for host HW because no other component can
  - Also provide other traditional OS services like CPU and memory management

The VMM runs directly on the hardware.

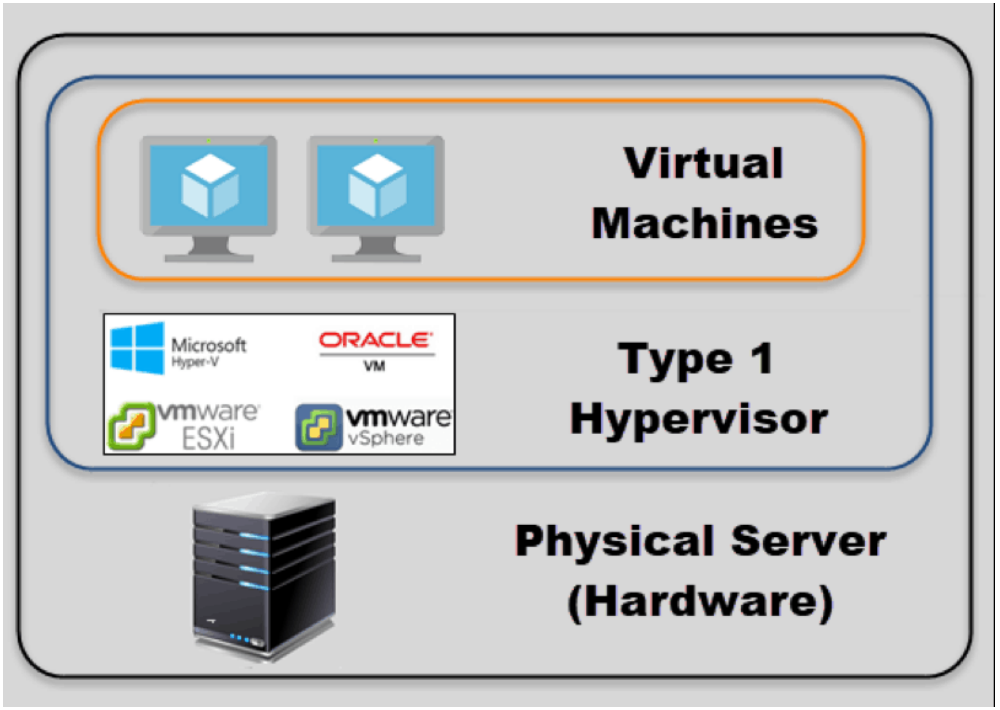


HYPERVISOR TYPE 1 EXAMPLES



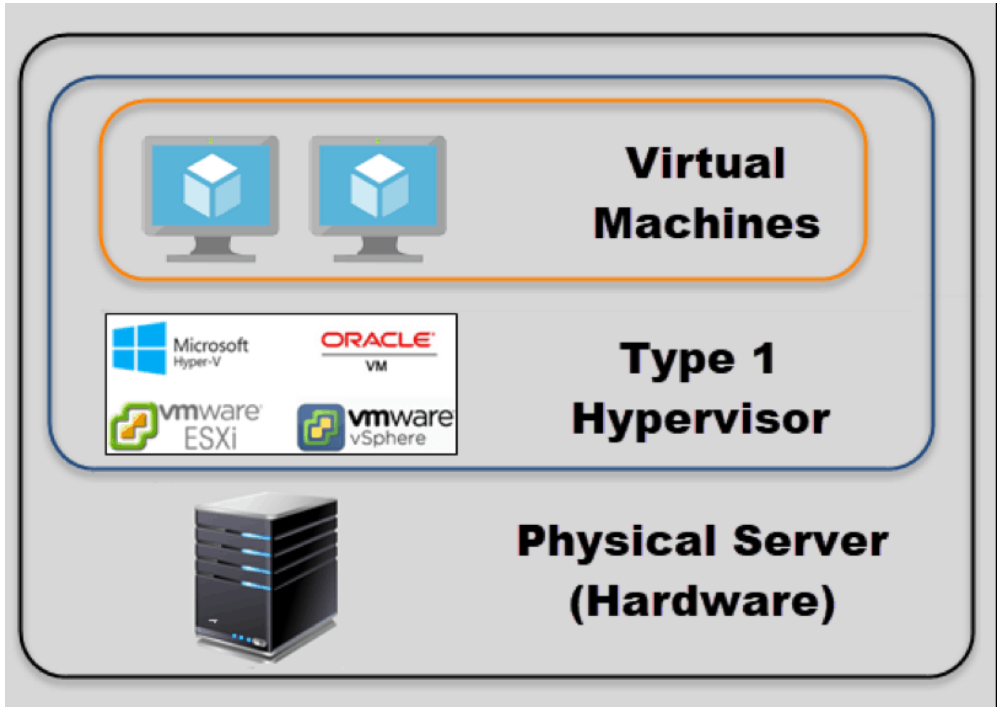
HYPERVISOR TYPE 1 EXAMPLES

- **VMware vSphere** with **ESX/ESXi** as a part of the package.
  - VMware is an industry-leading vendor of virtualization technology, and many large data centers run on their products.



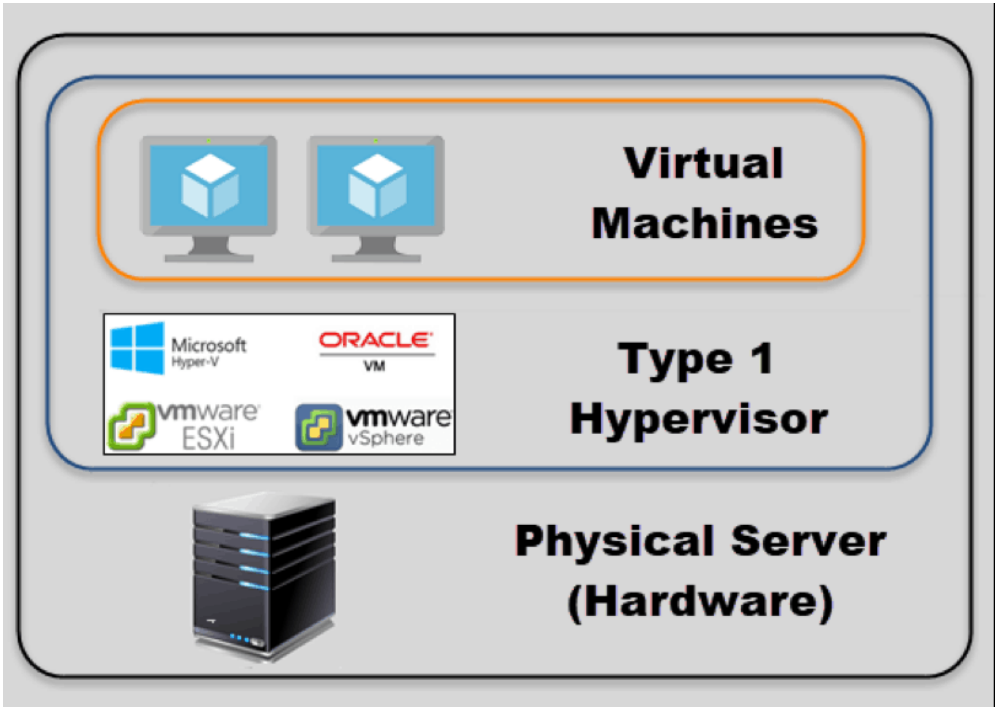
HYPERVISOR TYPE 1 EXAMPLES

- **VMware vSphere** with **ESX/ESXi** as a part of the package.
  - VMware is an industry-leading vendor of virtualization technology, and many large data centers run on their products.
- **KVM (Kernel-based Virtual Machine):**
  - KVM is built into Linux as an added functionality. It lets you convert Linux kernel into a hypervisor. It is sometimes confused to be a type 2 hypervisor. It actually has direct access to hardware along with virtual machines it hosts.



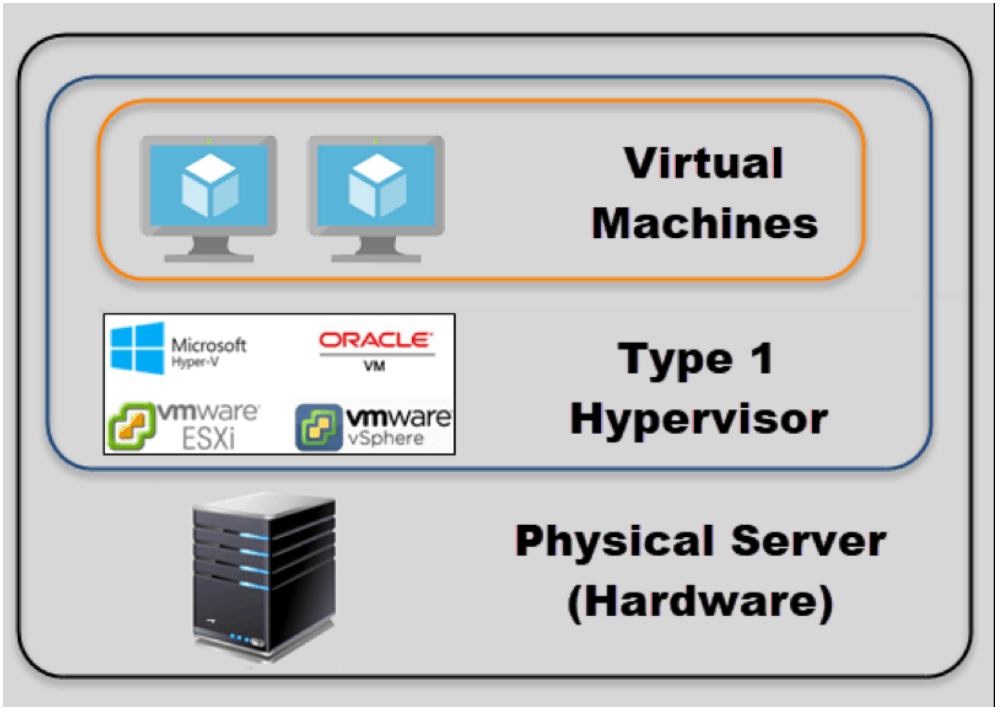
HYPERVISOR TYPE 1 EXAMPLES

- **VMware vSphere** with **ESX/ESXi** as a part of the package.
  - VMware is an industry-leading vendor of virtualization technology, and many large data centers run on their products.
- **KVM (Kernel-based Virtual Machine):**
  - KVM is built into Linux as an added functionality. It lets you convert Linux kernel into a hypervisor. It is sometimes confused to be a type 2 hypervisor. It actually has direct access to hardware along with virtual machines it hosts.
- **Microsoft Hyper-V:**
  - Microsoft also offers a free edition of their hypervisor. Hyper-V may not offer as many features as VMware vSphere.



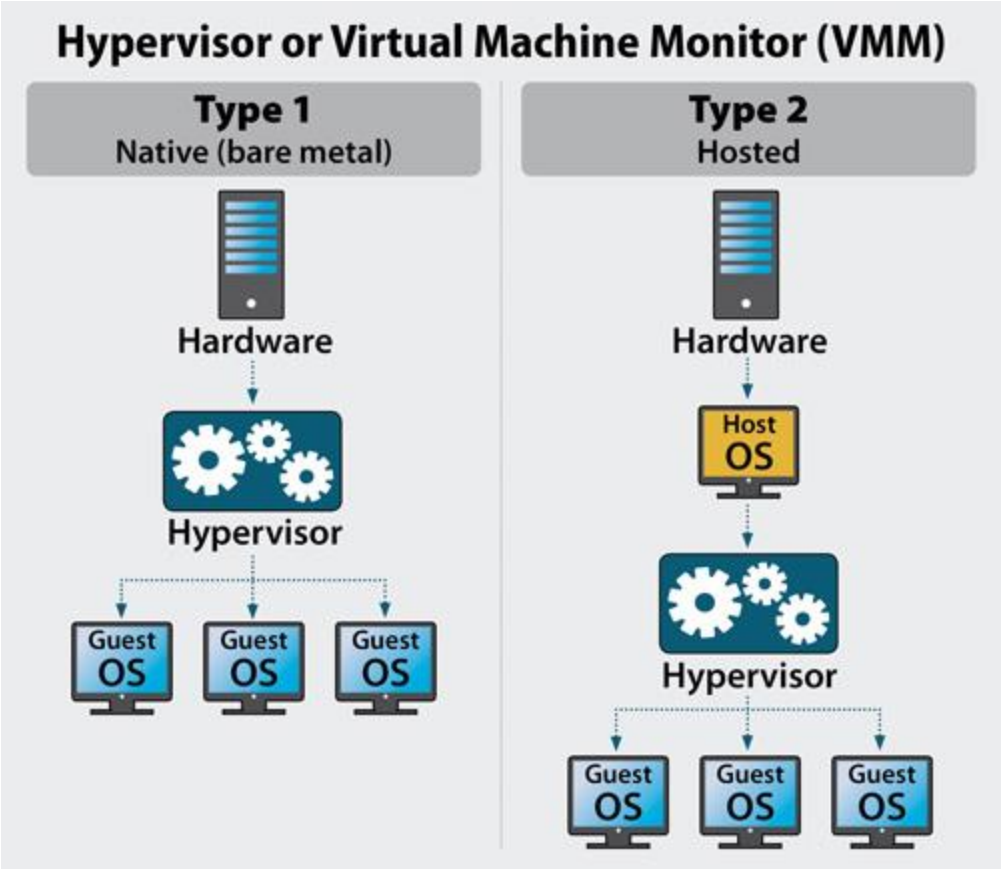
HYPERVISOR TYPE 1 EXAMPLES

- **VMware vSphere** with **ESX/ESXi** as a part of the package.
  - VMware is an industry-leading vendor of virtualization technology, and many large data centers run on their products.
- **KVM (Kernel-based Virtual Machine):**
  - KVM is built into Linux as an added functionality. It lets you convert Linux kernel into a hypervisor. It is sometimes confused to be a type 2 hypervisor. It actually has direct access to hardware along with virtual machines it hosts.
- **Microsoft Hyper-V:**
  - Microsoft also offers a free edition of their hypervisor. Hyper-V may not offer as many features as VMware vSphere.



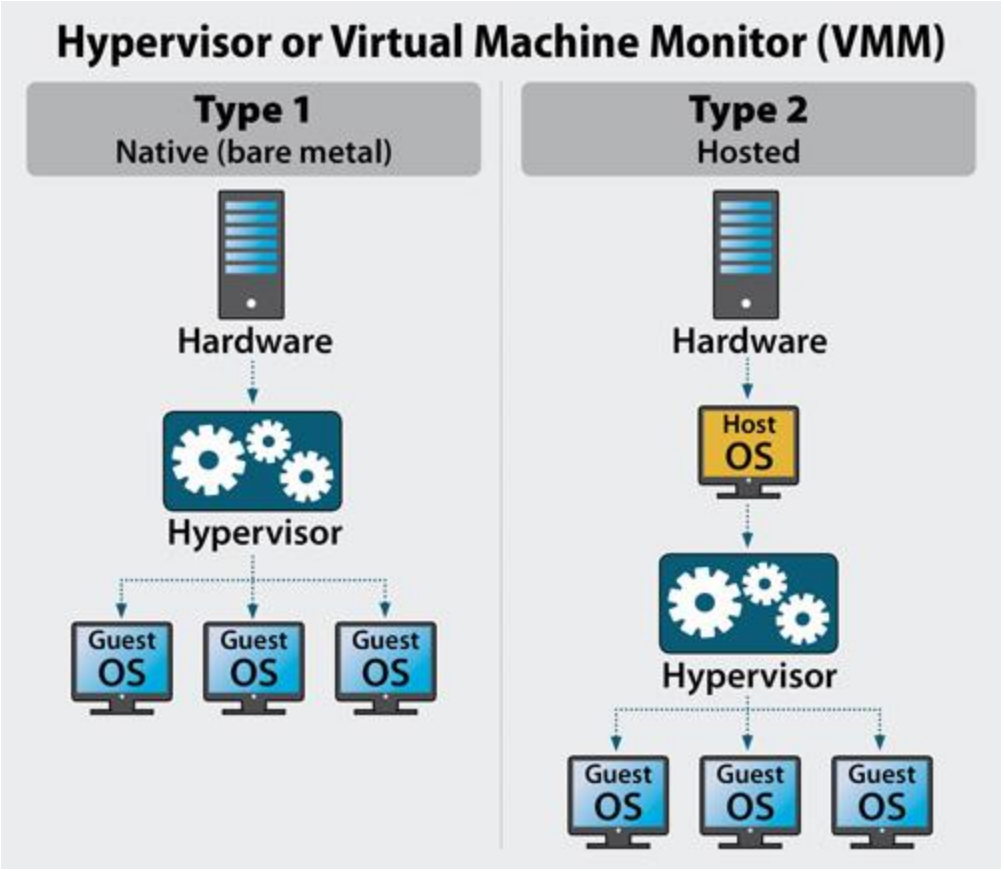
HYPERVISORS TYPES

- Type 2:
  - A type 2 hypervisor software within that operating system.



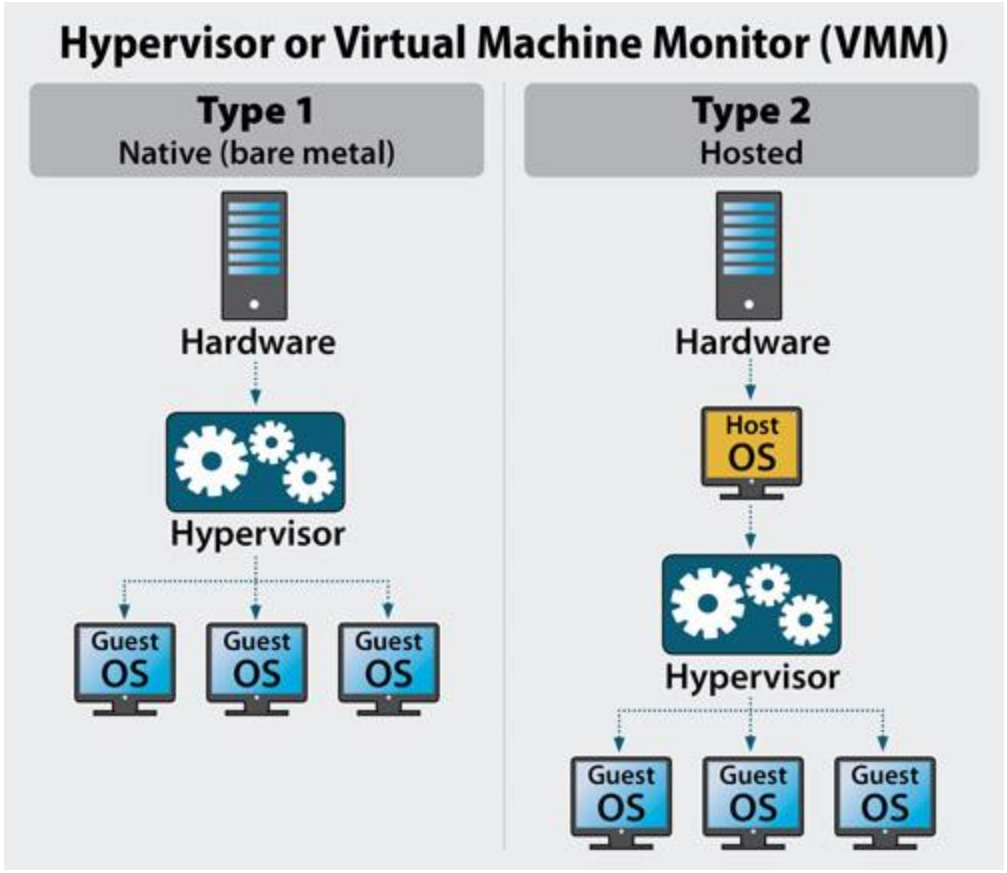
HYPERVISORS TYPES

- Type 2:
  - A type 2 hypervisor software within that operating system.
  - The actual instances of guest virtual machines are simply normal processes running on the OS.



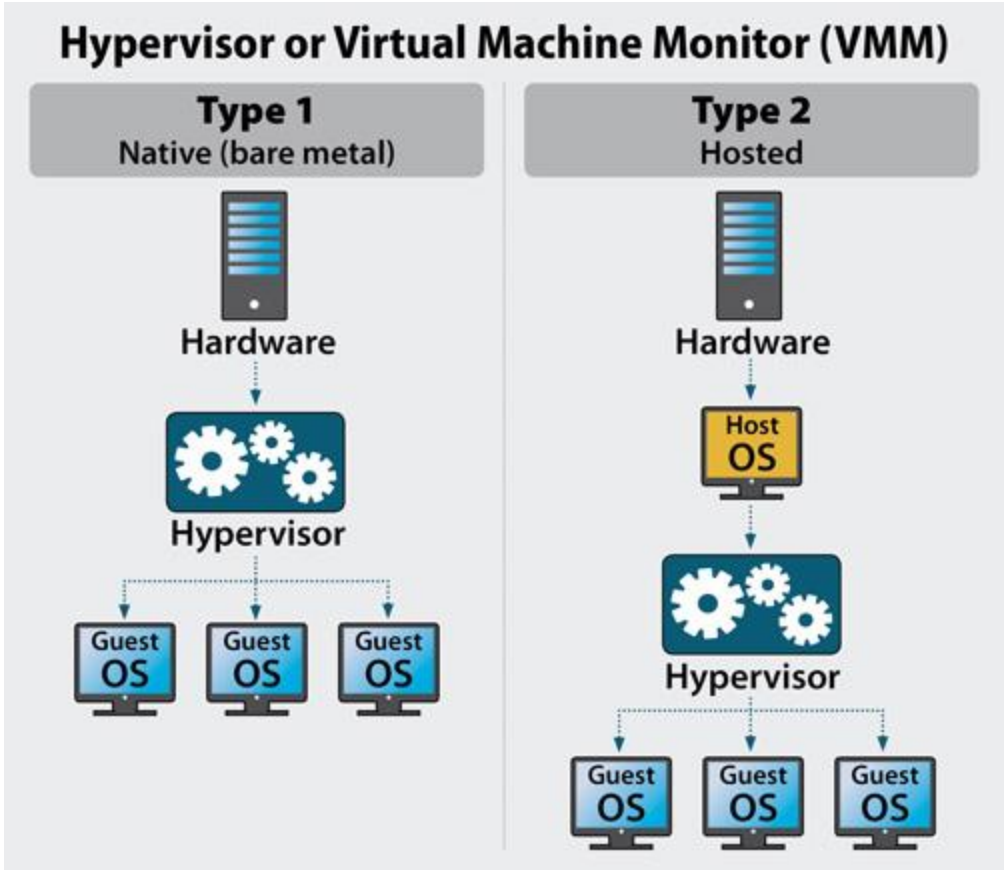
HYPERVISORS TYPES

- Type 2:
  - A type 2 hypervisor software within that operating system.
  - The actual instances of guest virtual machines are simply normal processes running on the OS.
  - Tend to have poorer overall performance because can't take advantage of some HW features



HYPERVISORS TYPES

- Type 2:
  - A type 2 hypervisor software within that operating system.
  - The actual instances of guest virtual machines are simply normal processes running on the OS.
  - Tend to have poorer overall performance because can't take advantage of some HW features
  - Require no changes to host OS, can use on most computers.



HYPERVISORS TYPE 2 EXAMPLES

- **Oracle VM VirtualBox**.
  - A free but stable product with enough features for personal use and most use cases for smaller businesses.
  - It provides support for guest multiprocessing with up to 32 vCPUs per virtual machine, PXE Network boot, snapshot trees, and many more.

HYPERVISORS TYPE 2 EXAMPLES

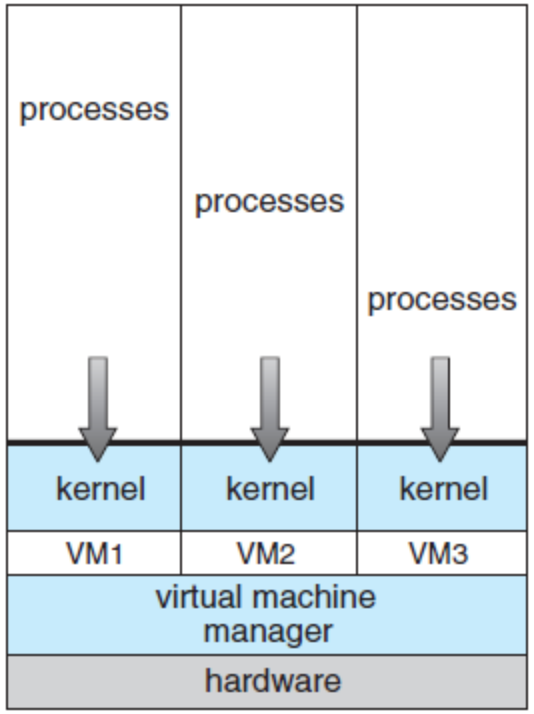
- **Oracle VM VirtualBox**.
  - A free but stable product with enough features for personal use and most use cases for smaller businesses.
  - It provides support for guest multiprocessing with up to 32 vCPUs per virtual machine, PXE Network boot, snapshot trees, and many more.
- **VMware Workstation Pro / VMware Fusion**.
  - VMware Workstation Pro is a type 2 hypervisor for Windows OS.
  - It is full of advanced features and has seamless integration with vSphere. This allows you to move your apps between desktop and cloud environments.

VM IMPLEMENTATION

Let's look at some implementation challenges.

VM IMPLEMENTATION

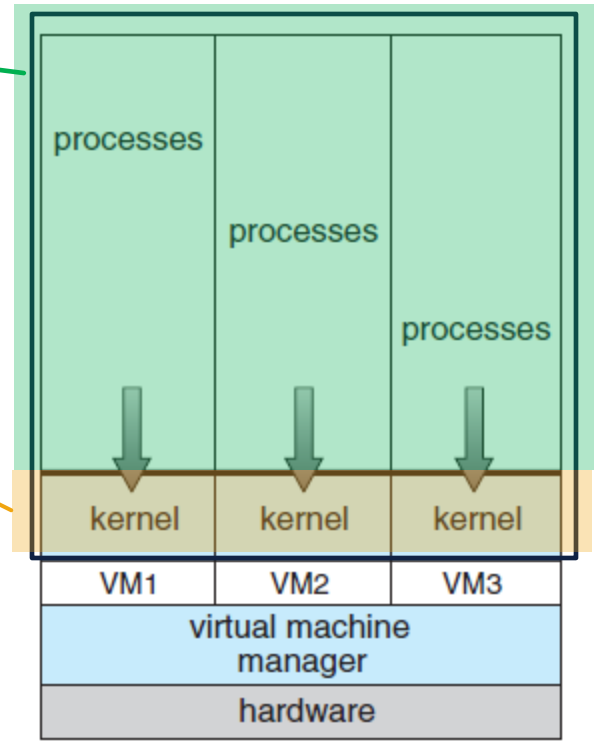
Q: How is user/kernel mode implemented in VMs?



VM IMPLEMENTATION

User Mode in eyes of Guest Machine

Kernel Mode in eyes of Guest Machine

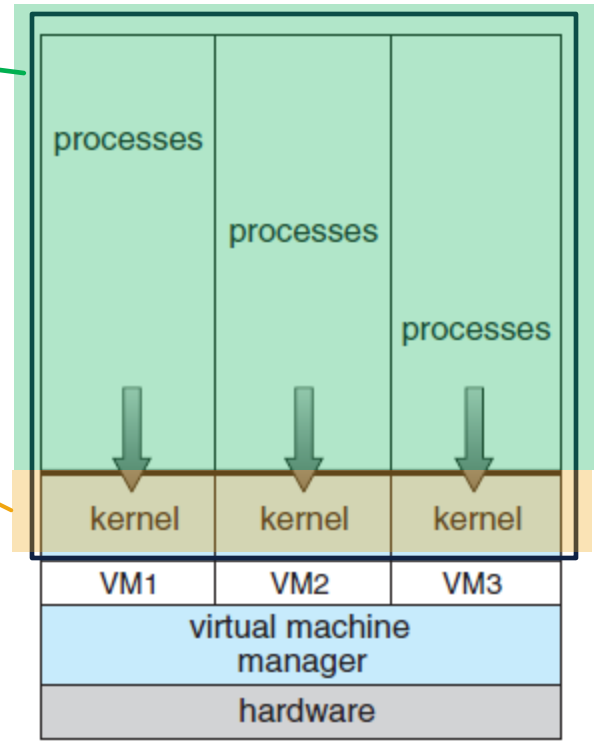


VM IMPLEMENTATION

User Mode in eyes of Guest Machine

Kernel Mode in eyes of Guest Machine

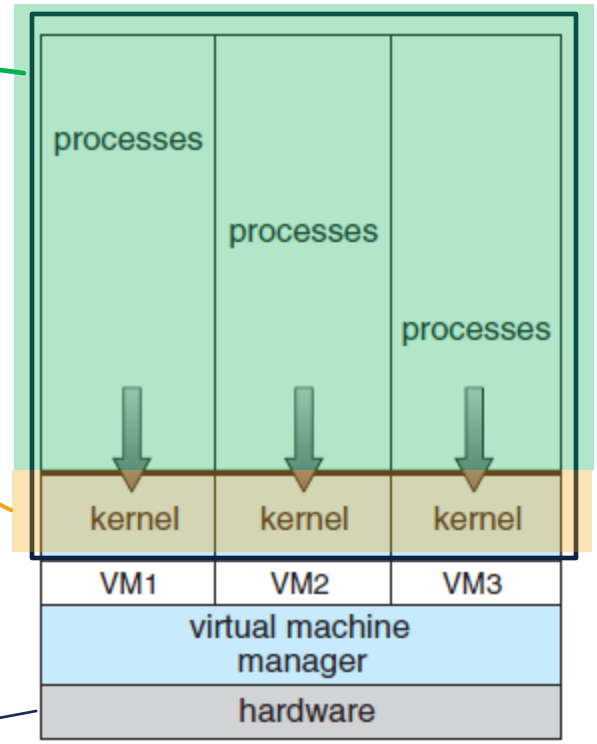
From the VM (Guest machine) perspective, the code can be in user mode or kernel mode.



VM IMPLEMENTATION

User Mode in eyes of Guest Machine

Kernel Mode in eyes of Guest Machine



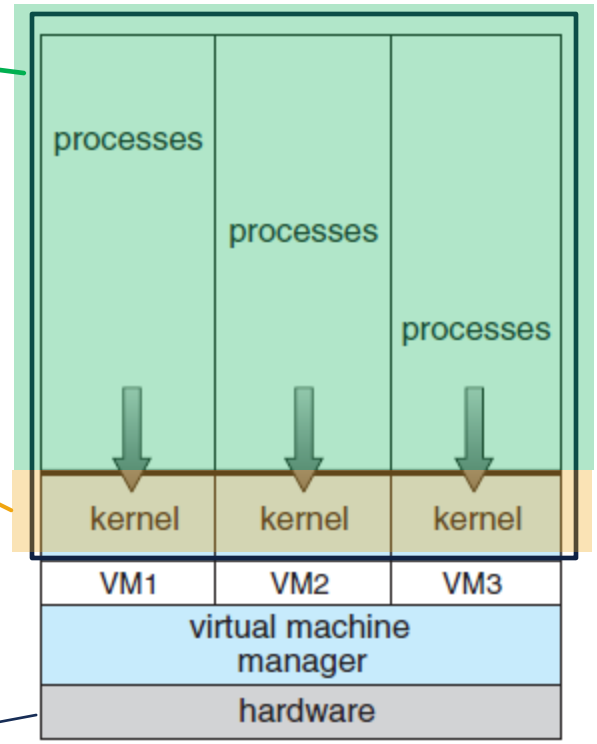
However, in the Host Machine. all VM code is running in user mode.

VM IMPLEMENTATION

User Mode in eyes of Guest Machine

This is 'virtual' kernel mode, not real kernel mode.

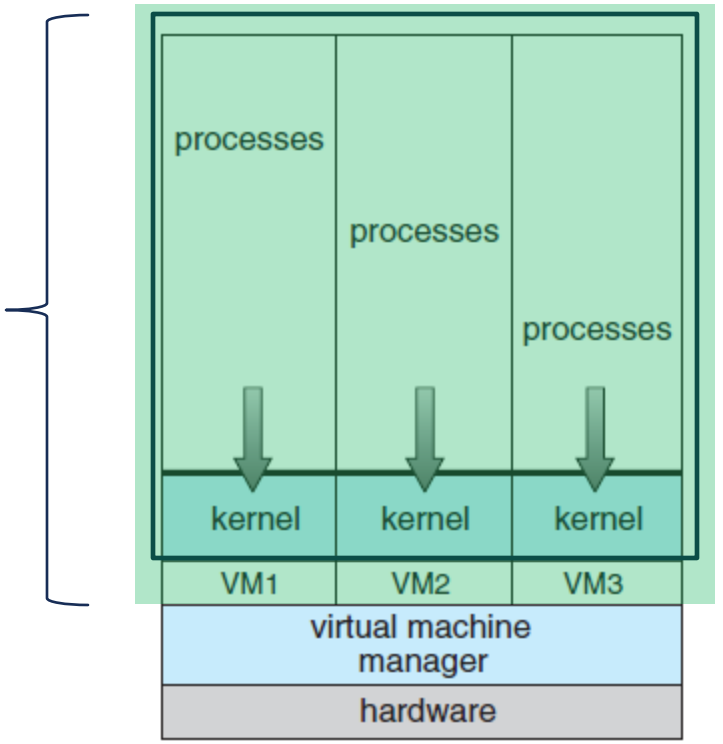
Kernel Mode in eyes of Guest Machine



However, in the Host Machine. all VM code is running in user mode.

VM IMPLEMENTATION

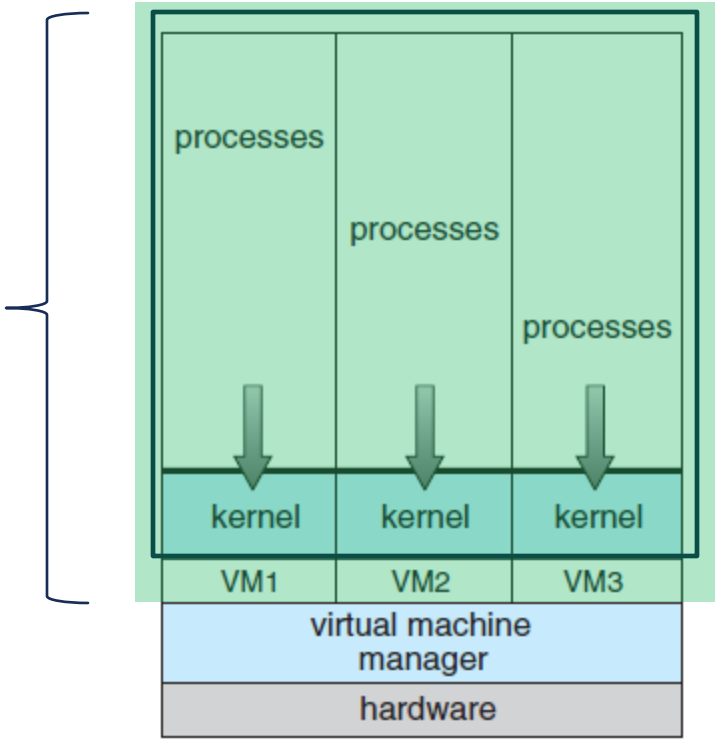
All VMs and associated processes are running in user mode.



VM IMPLEMENTATION

Q: How can we virtualize the kernel mode of the machine?

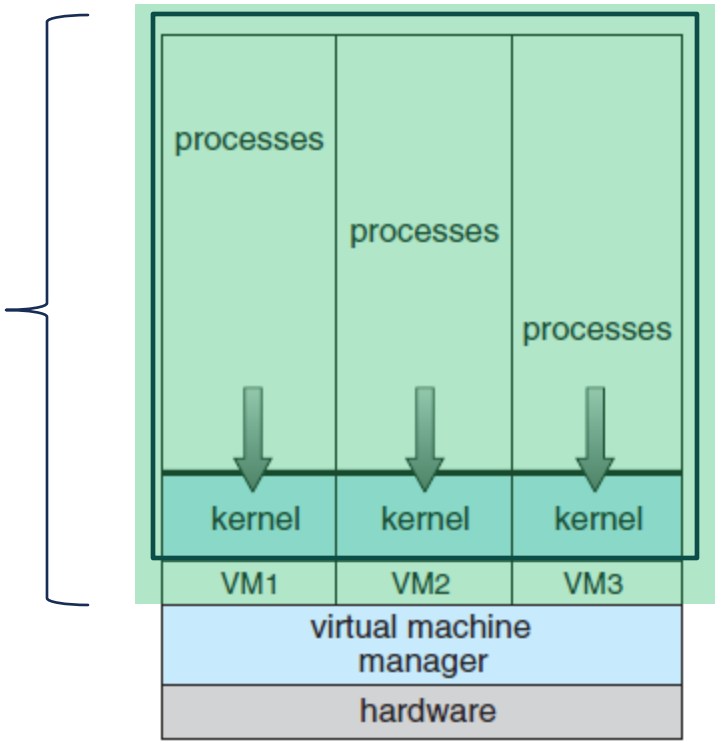
All VMs and associated processes are running in user mode.



VM IMPLEMENTATION

Q: How can we virtualize the kernel mode of the machine?

All VMs and associated processes are running in user mode.



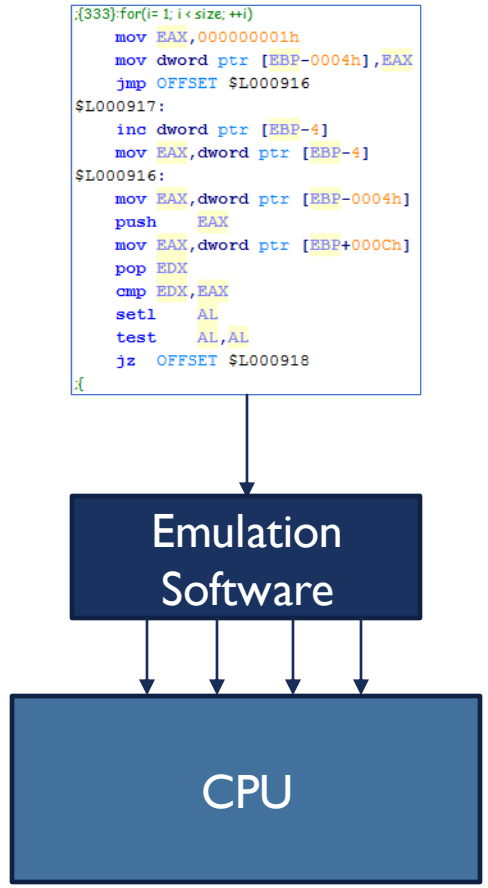
Q: Does the VM need access to kernel mode?

Emulation vs. Virtualization

- Machine Emulation: Guest machine run on a simulated processor; entirely done in software.

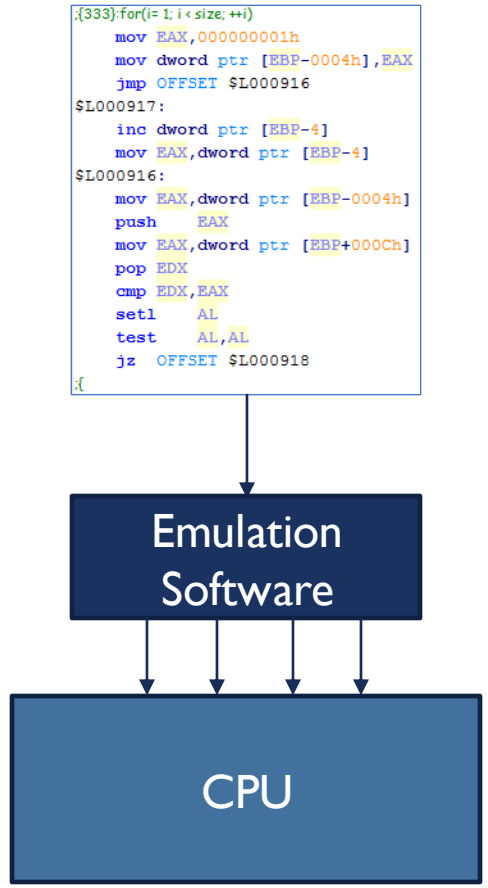
EMULATION VS VIRTUALIZATION

- Machine Emulation: Guest machine run on a simulated processor; entirely done in software.



EMULATION VS VIRTUALIZATION

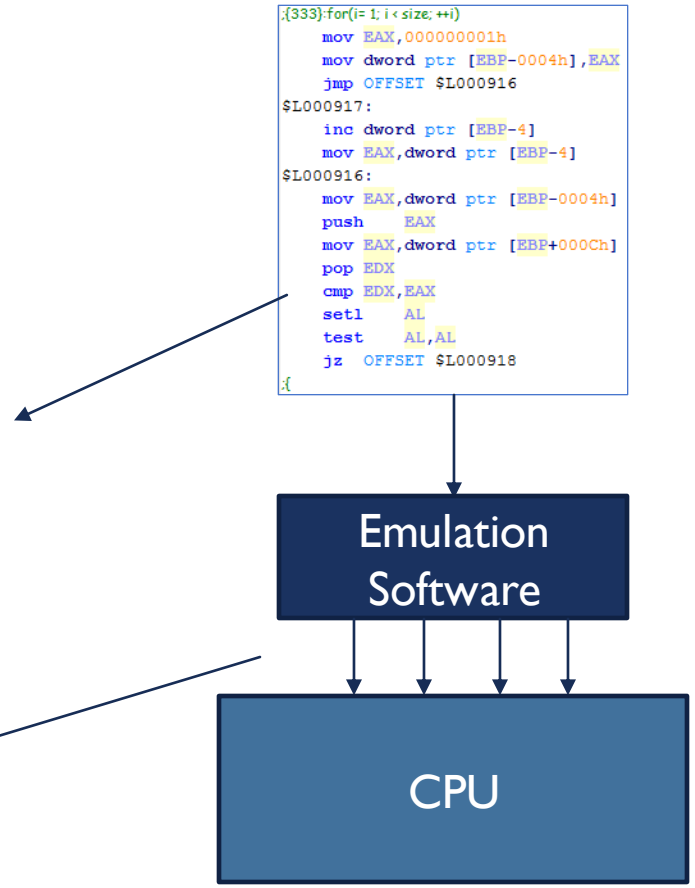
- Machine Emulation: Guest machine run on a simulated processor; entirely done in software.
- Emulation usually is done for programs rather than entire operating system.
- Carries a heavy performance penalty.



EMULATION VS VIRTUALIZATION

- Machine Emulation: Guest machine run on a simulated processor; entirely done in software.
- Emulation usually is done for programs rather than entire operating system.
- Carries a heavy performance penalty.

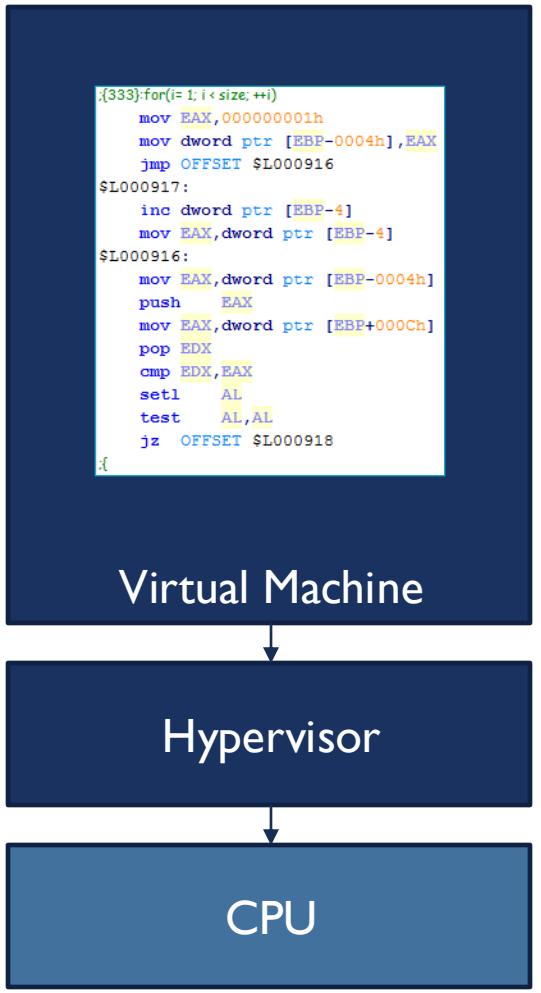
Generated code is incompatible with hardware/CPU.



What reaches the CPU is an entirely different code.

VIRTUALIZATION

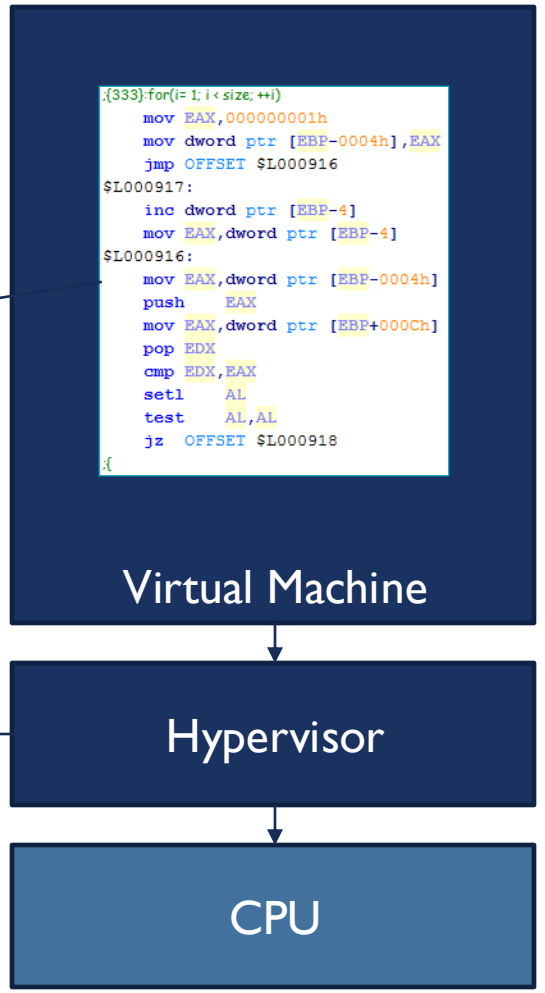
- The code of programs running in the guest machine are compatible with the CPU/hardware.
- The instructions will need to run on the actual CPU.



VIRTUALIZATION

- The code of programs running in the guest machine are compatible with the CPU/hardware.
- The instructions will need to run on the actual CPU.

Generated code is compatible with current CPU/Hardware.



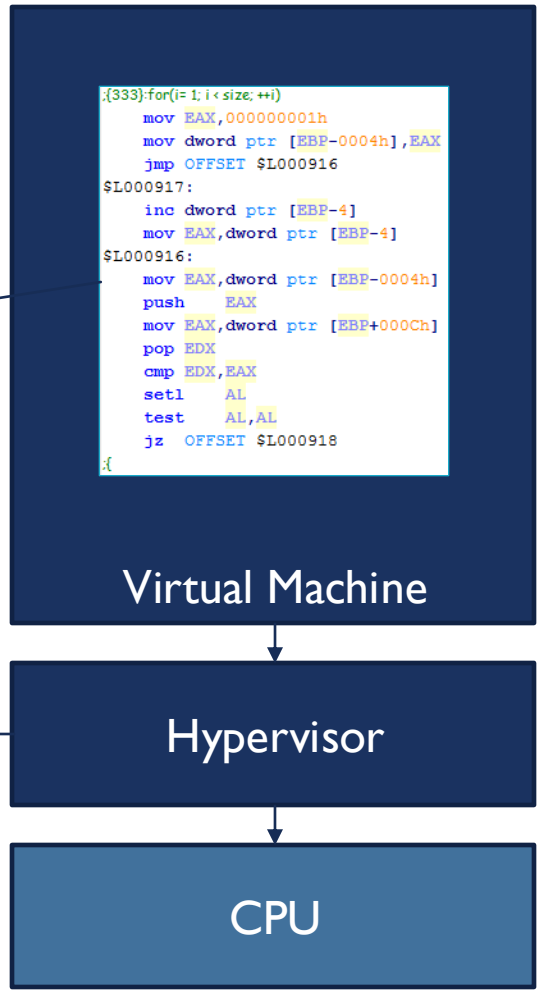
Hypervisor "manages" the code and (sometimes) does minor modifications.

VIRTUALIZATION

- The code of programs running in the guest machine are compatible with the CPU/hardware.
- The instructions will need to run on the actual CPU.

So how does relate to whether we need kernel mode in virtual machines?

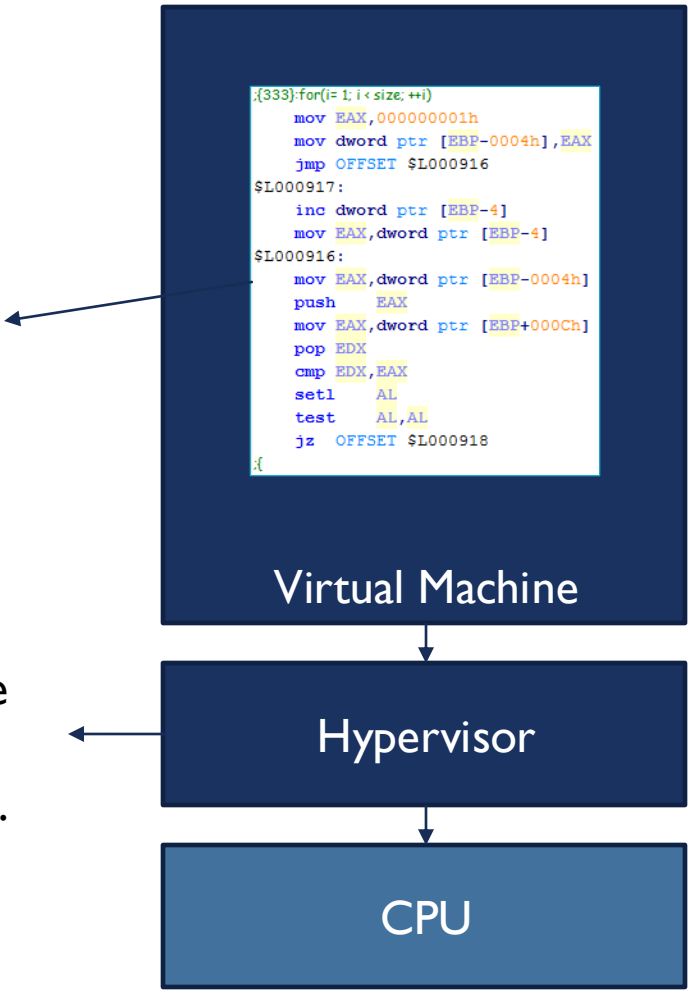
Generated code is compatible with current CPU/Hardware.



Hypervisor "manages" the code and (sometimes) does minor modifications.

VIRTUALIZATION

- The code of programs running in the guest machine are compatible with the CPU/hardware.
- The instructions will need to run on the actual CPU.
- **Some instructions are privileged and need kernel mode.**



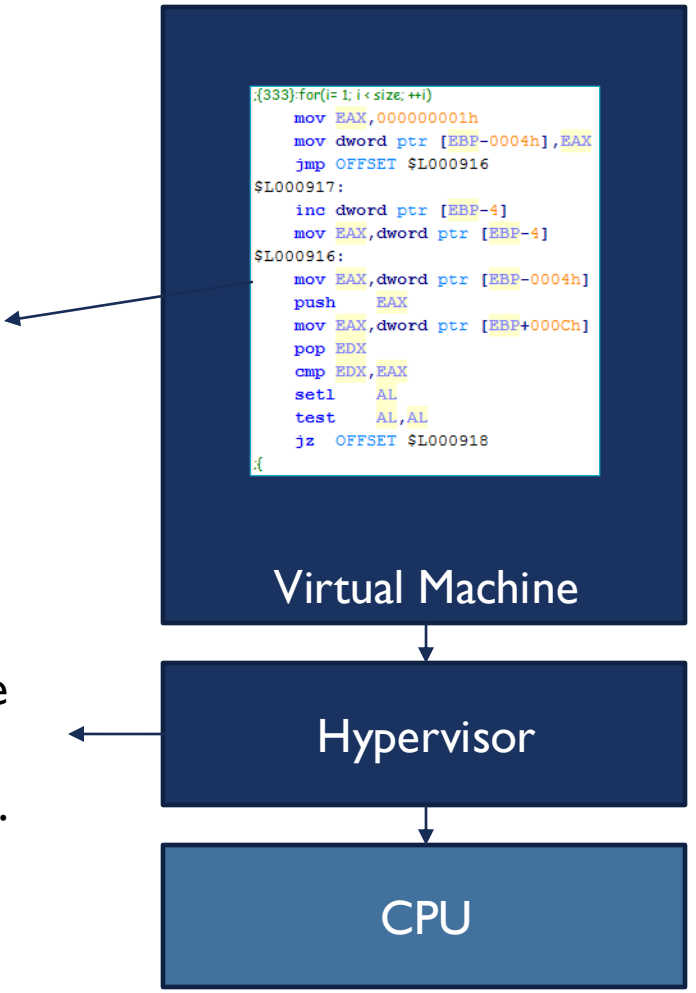
Generated code is compatible with current CPU/Hardware.

Hypervisor "manages" the code and (sometimes) does minor modifications.

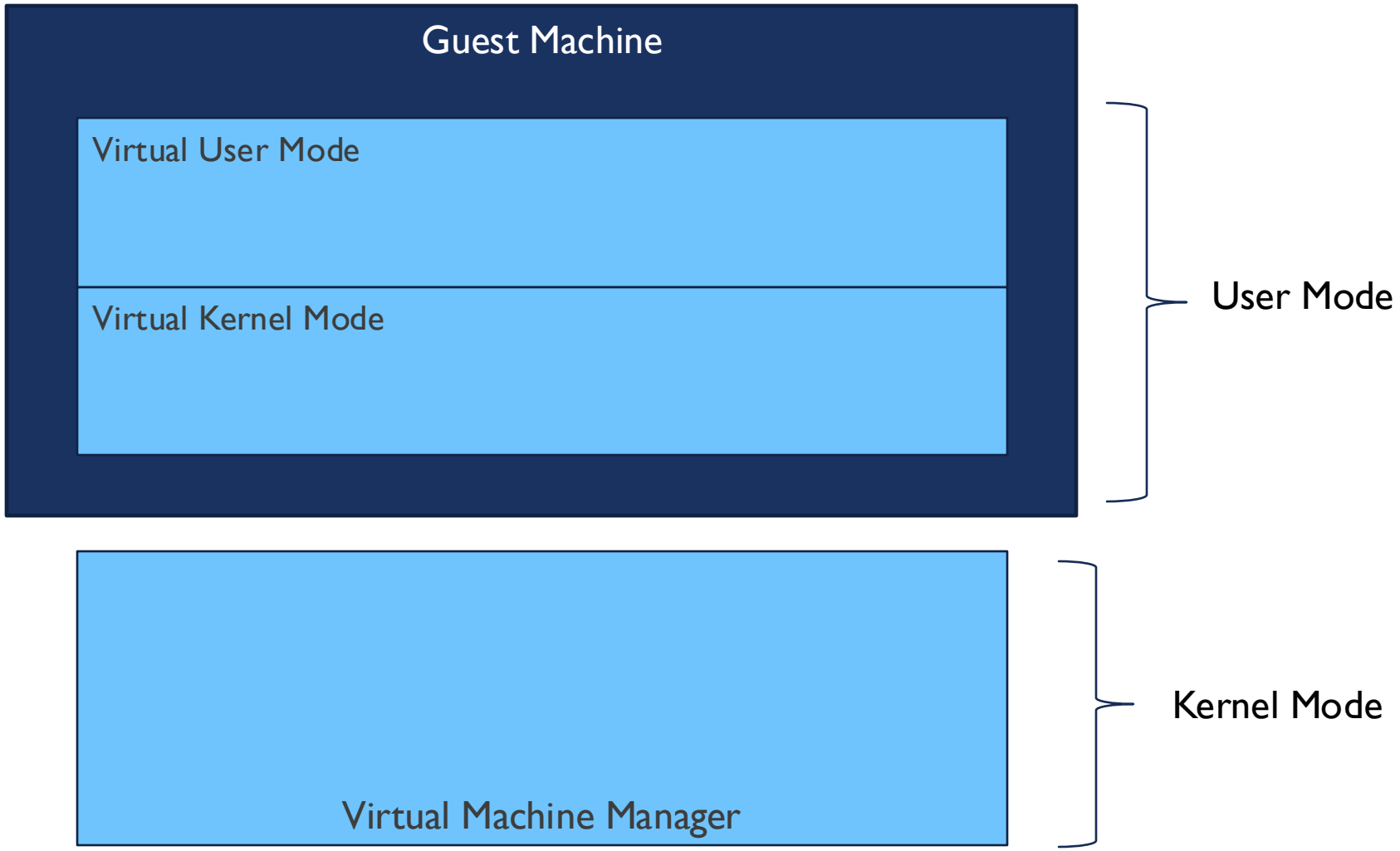
VIRTUALIZATION

- The code of programs running in the guest machine are compatible with the CPU/hardware.
- The instructions will need to run on the actual CPU.
- **Some instructions are privileged and need kernel mode.**

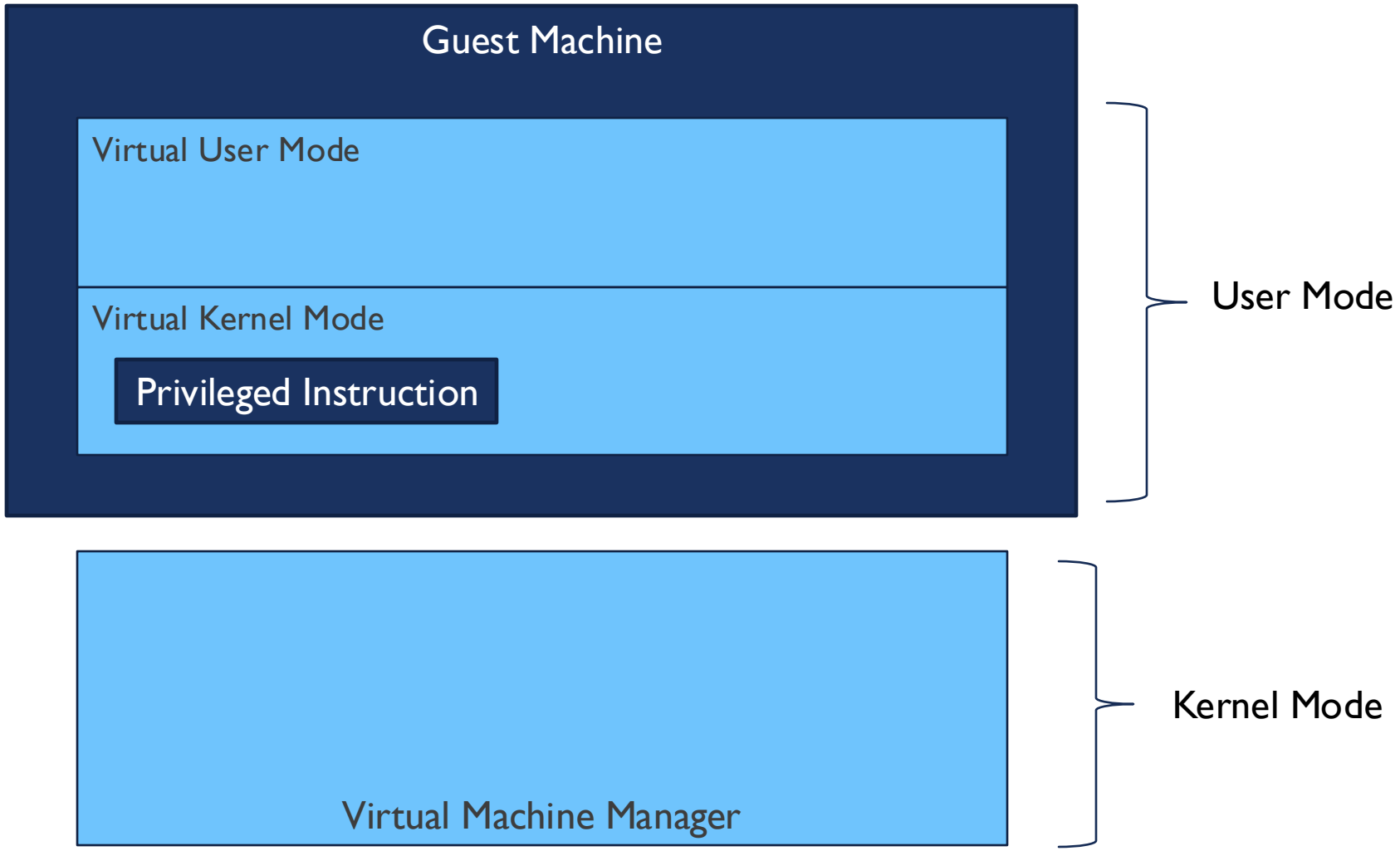
Q: How to handle kernel mode instruction for the virtual machine?



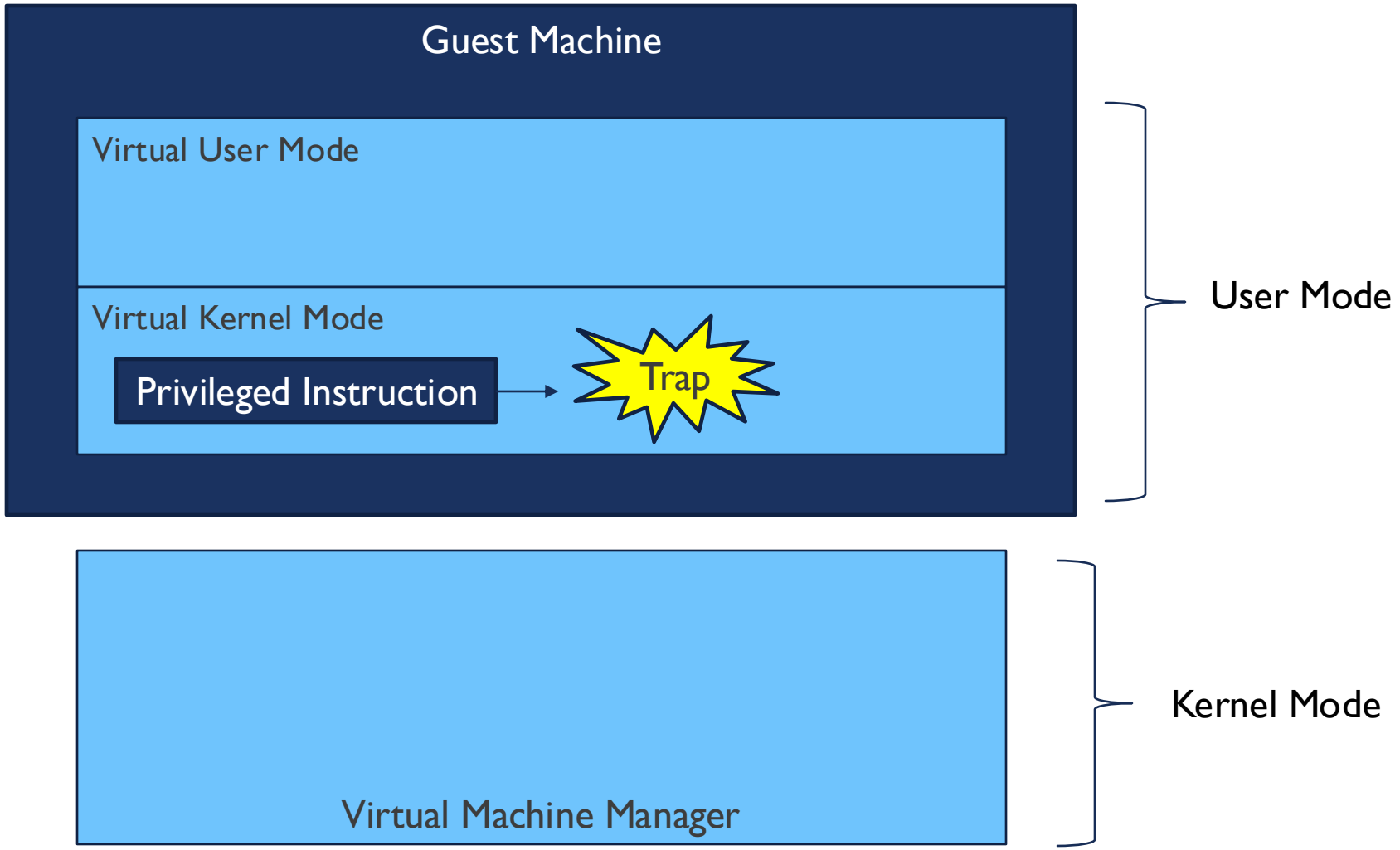
TRAP AND EMULATE



TRAP AND EMULATE

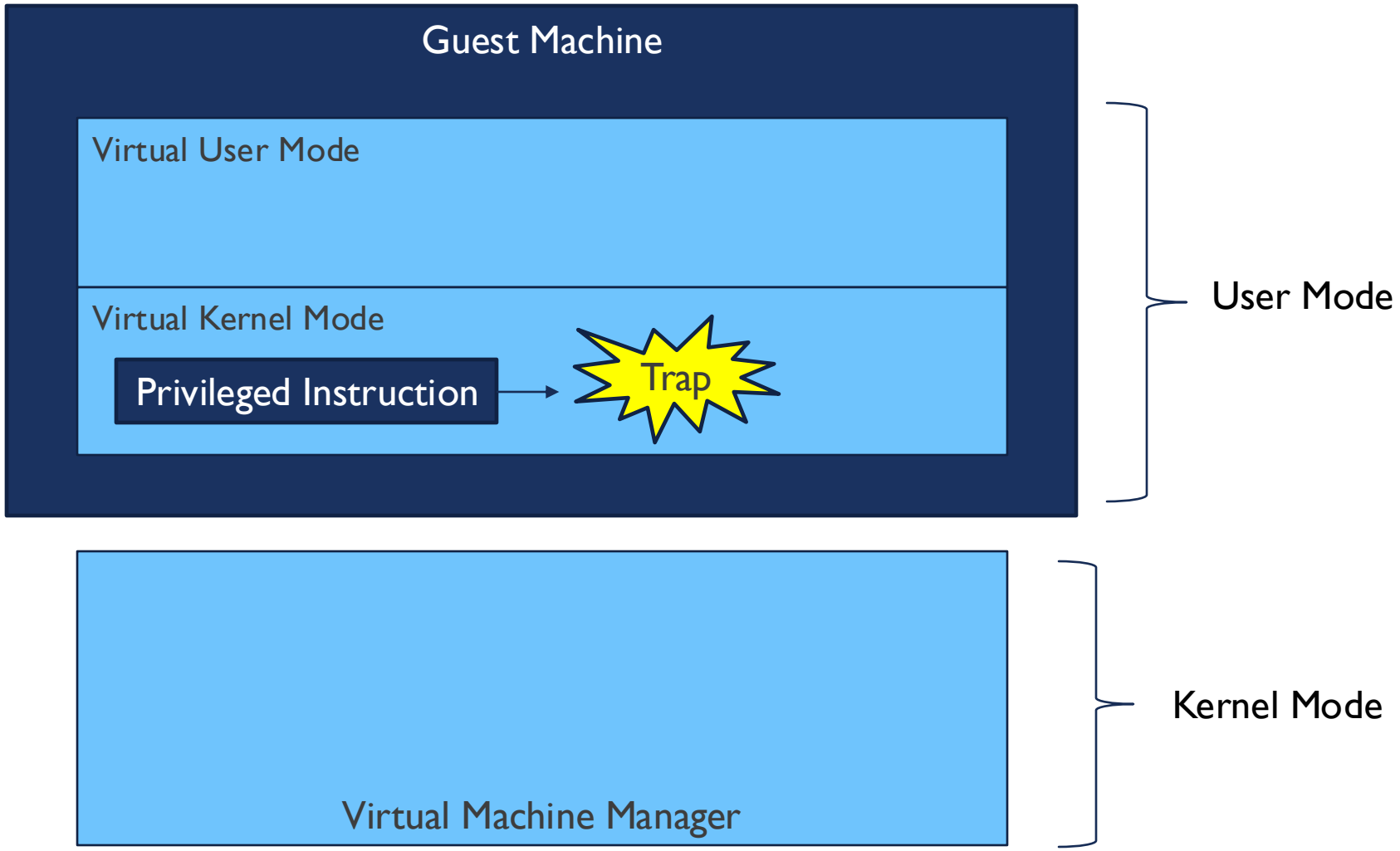


TRAP AND EMULATE



TRAP AND EMULATE

Q: Is this a virtual Trap or a real CPU Trap?

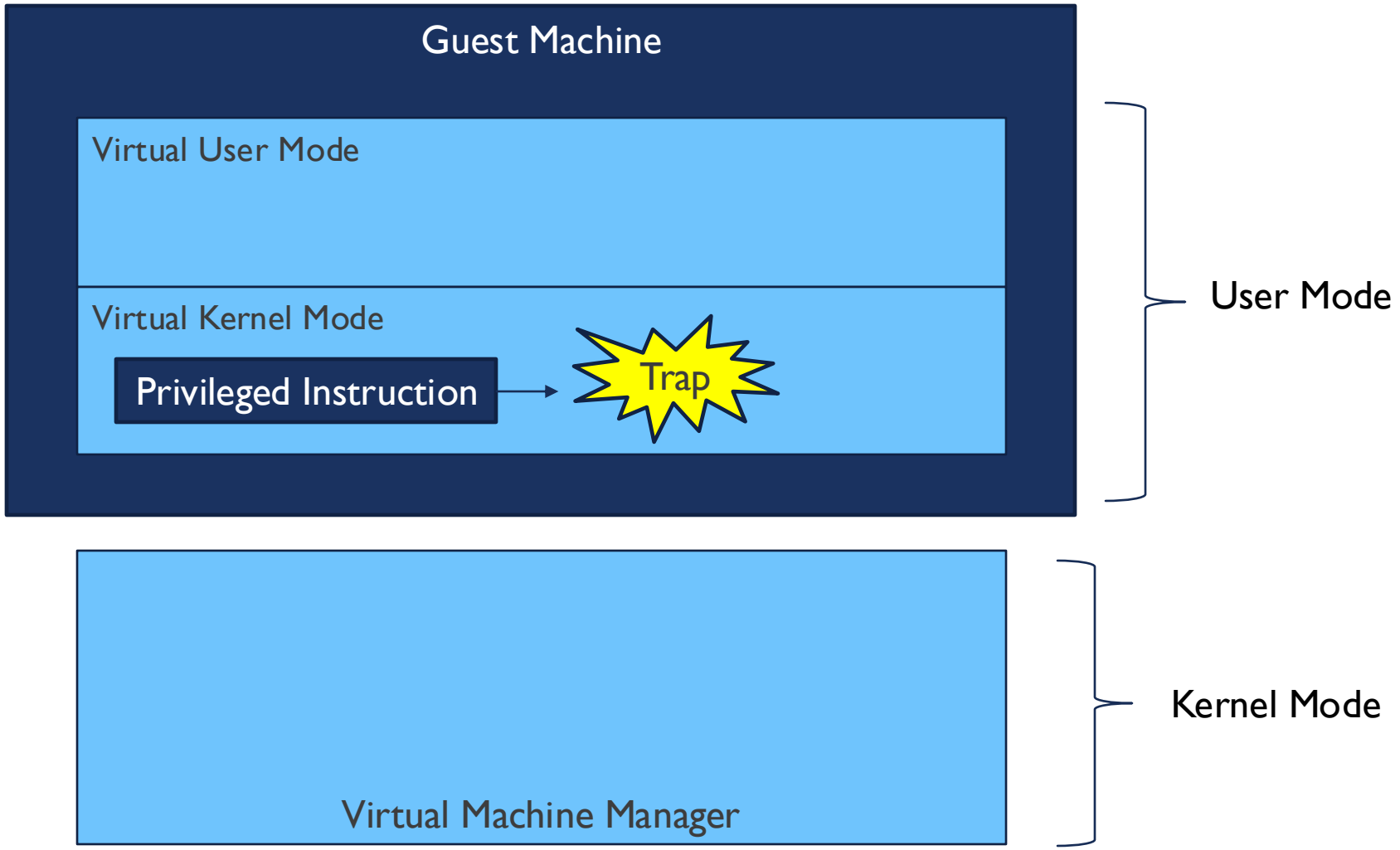


TRAP AND EMULATE

Q: Is this a virtual Trap or a real CPU Trap?

A B

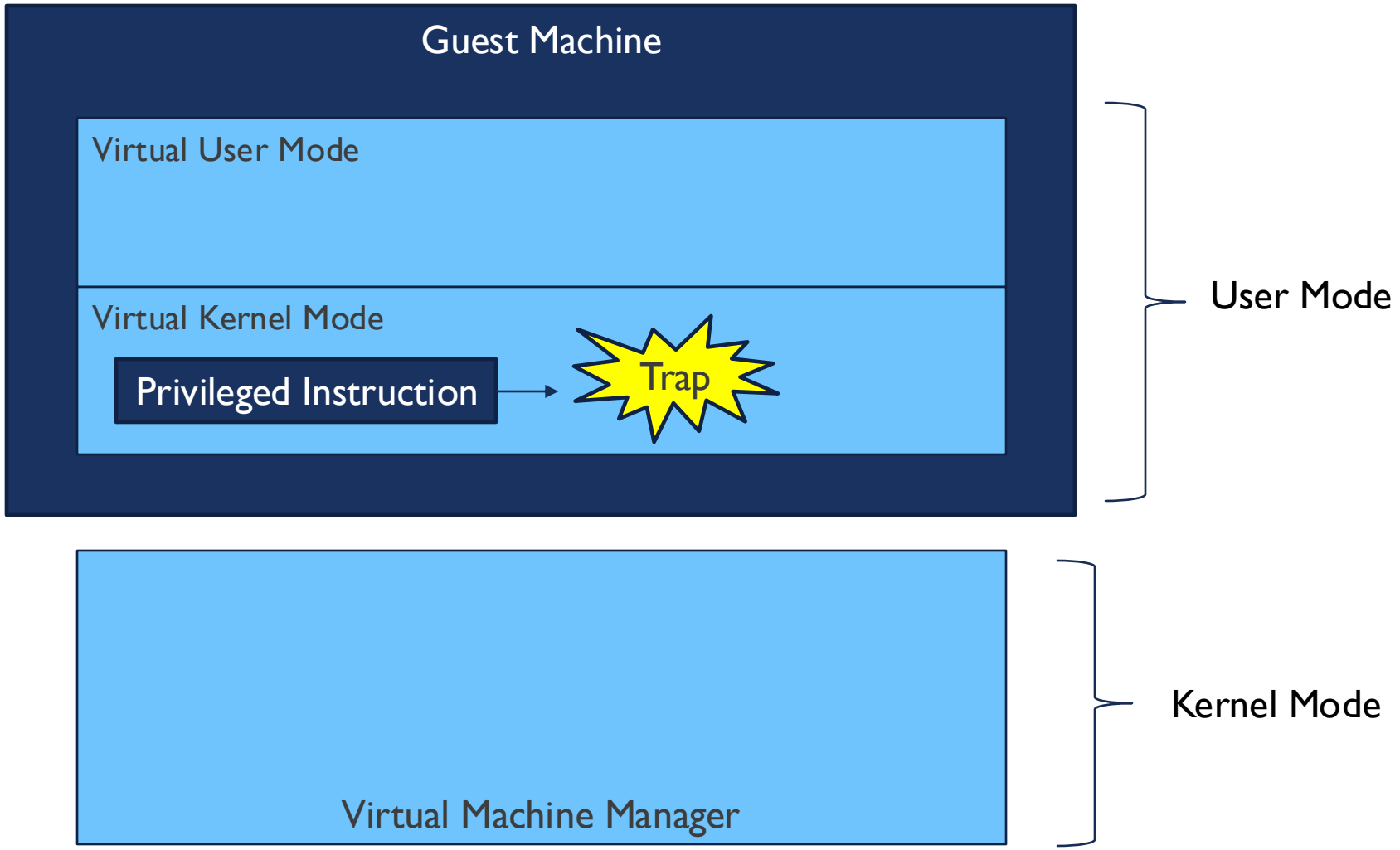
A: Real
B: Virtual



TRAP AND EMULATE

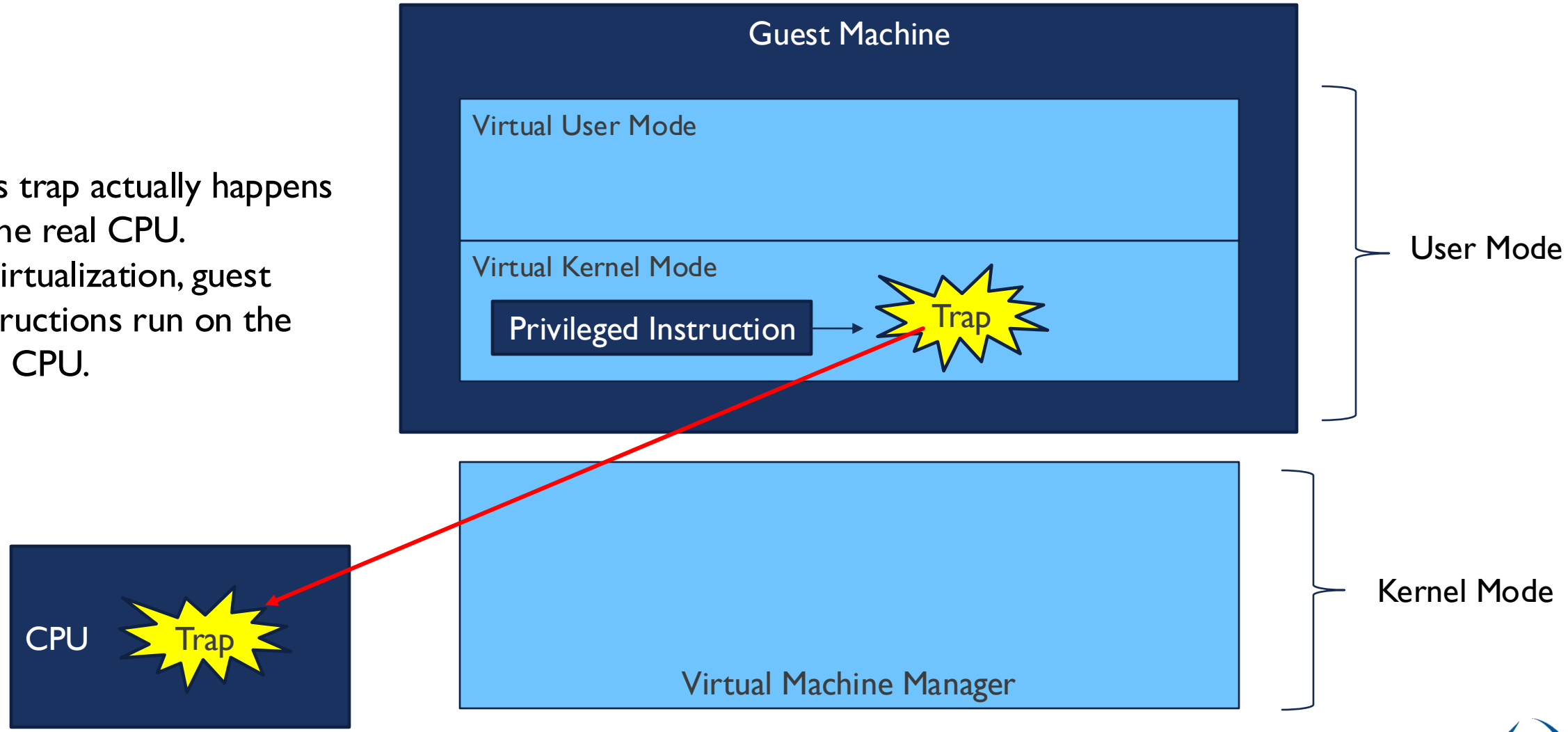
Q: Is this a virtual Trap or a real CPU Trap?

A: Real
B: Virtual

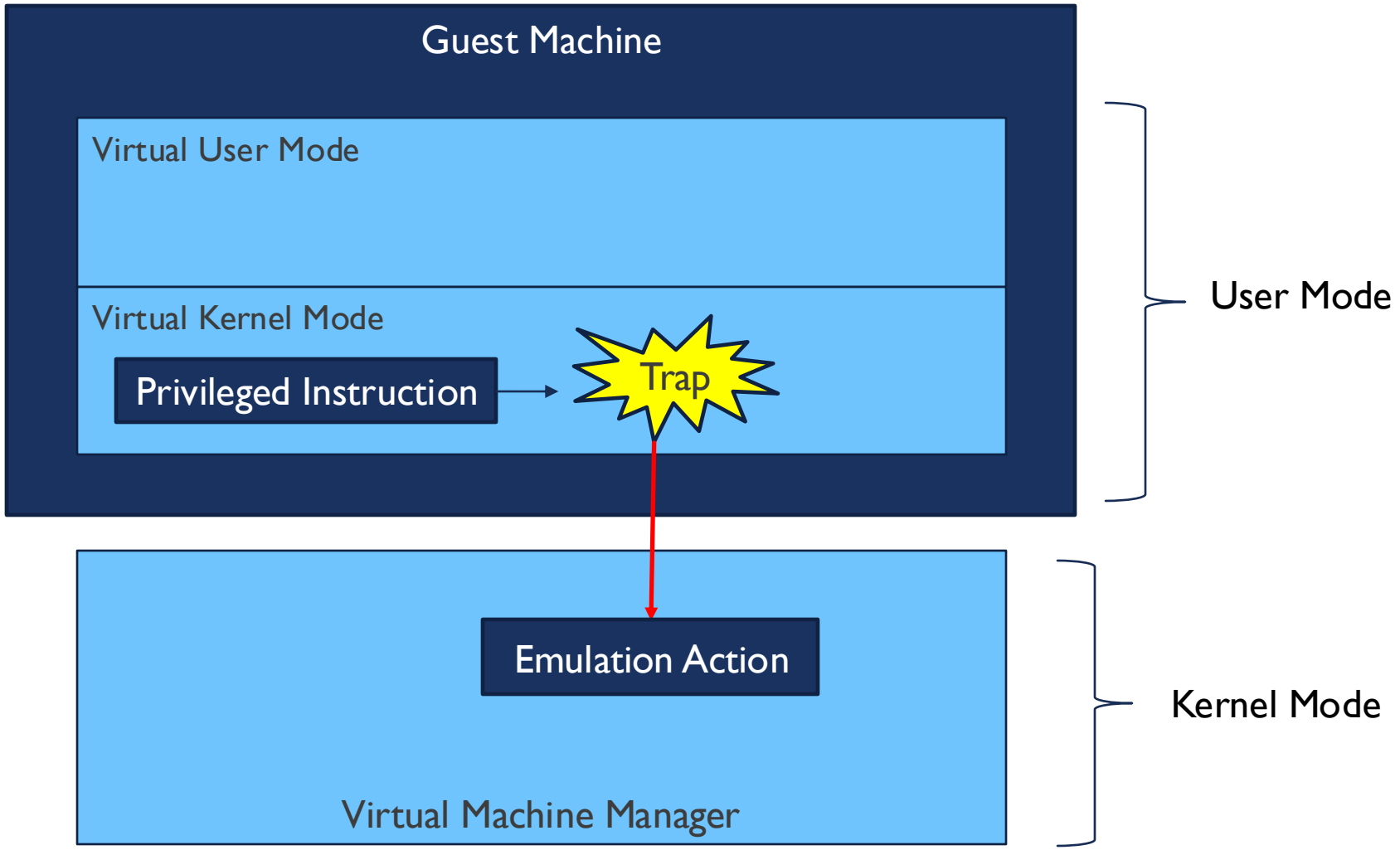


TRAP AND EMULATE

- This trap actually happens in the real CPU.
- In virtualization, guest instructions run on the real CPU.

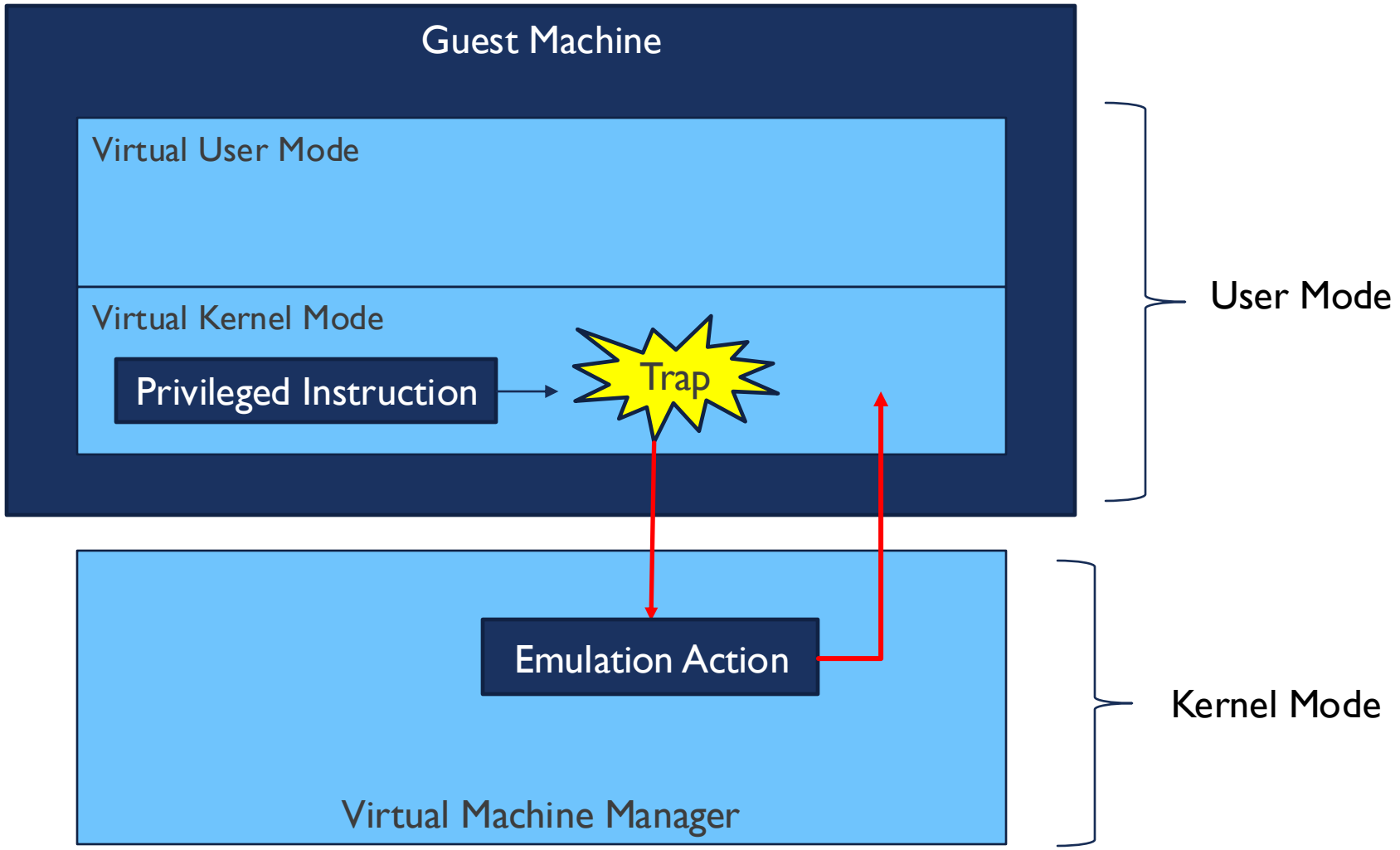


TRAP AND EMULATE



- VMM, in kernel mode, handles the trap/.
- It inspects the instruction, and "emulate it".

TRAP AND EMULATE



- VMM, in kernel mode, handles the trap/.
- It inspects the instruction, and "emulate it".
- The VMM then resumes the guest machine program.

LIMITATIONS OF TRAP AND EMULATE

- Sometimes the boundary between Kernel/User mode is not well defined.
- Processors were not designed with virtualization in mind.

LIMITATIONS OF TRAP AND EMULATE

- Sometimes the boundary between Kernel/User mode is not well defined.
- Processors were not designed with virtualization in mind.
- Some instructions are not "privileged" but rather perform differently when in kernel mode.

LIMITATIONS OF TRAP AND EMULATE

- Sometimes the boundary between Kernel/User mode is not well defined.
- Processors were not designed with virtualization in mind.
- Some instructions are not "privileged" but rather perform differently when in kernel mode.
- Example: x86 `popf`
  - Loads portion of flag register from the contents of the stack.
  - If the CPU is in privileged mode, all of the flags are replaced from the stack pop.

LIMITATIONS OF TRAP AND EMULATE

- Sometimes the boundary between Kernel/User mode is not well defined.
- Processors were not designed with virtualization in mind.
- Some instructions are not "privileged" but rather perform differently when in kernel mode.
- Example: x86 `popf`
  - Loads portion of flag register from the contents of the stack.
  - If the CPU is in privileged mode, all of the flags are replaced from the stack pop.

Q: Why is this an issue of the Trap and Emulate?

LIMITATIONS OF TRAP AND EMULATE

- Sometimes the boundary between Kernel/User mode is not well defined.
- Processors were not designed with virtualization in mind.
- Some instructions are not "privileged" but rather perform differently when in kernel mode.
- Example: x86 `popf`
  - Loads portion of flag register from the contents of the stack.
  - If the CPU is in privileged mode, all of the flags are replaced from the stack pop.

Q: Why is this an issue of the Trap and Emulate?

There is no trap!

LIMITATIONS OF TRAP AND EMULATE

- Sometimes the boundary between Kernel/User mode is not well defined.
- Processors were not designed with virtualization in mind.
- Some instructions are not "privileged" but rather perform differently when in kernel mode.
- Example: x86 `popf`
  - Loads portion of flag register from the contents of the stack.
  - If the CPU is in privileged mode, all of the flags are replaced from the stack pop.

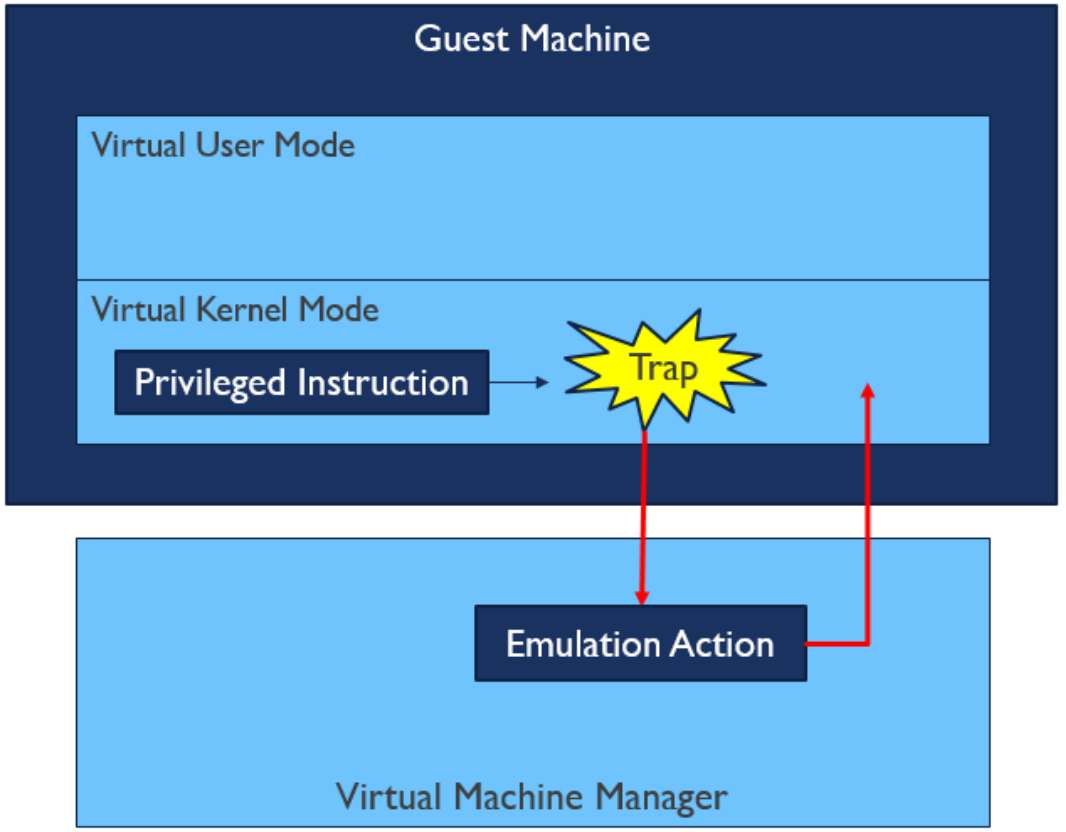
`popf` can execute both in kernel or user but behaves differently in each mode.

Q: Why is this an issue of the Trap and Emulate?

There is no trap!

LIMITATIONS OF TRAP AND EMULATE

- Sometimes the boundary between Kernel/User mode is not well defined.
- Processors were not designed with virtualization in mind.
- Some instructions are not "privileged" but rather perform differently when in kernel mode.
- Example: x86 `popf`
  - Loads portion of flag register from the contents of the stack.
  - If the CPU is in privileged mode, all of the flags are replaced from the stack pop.



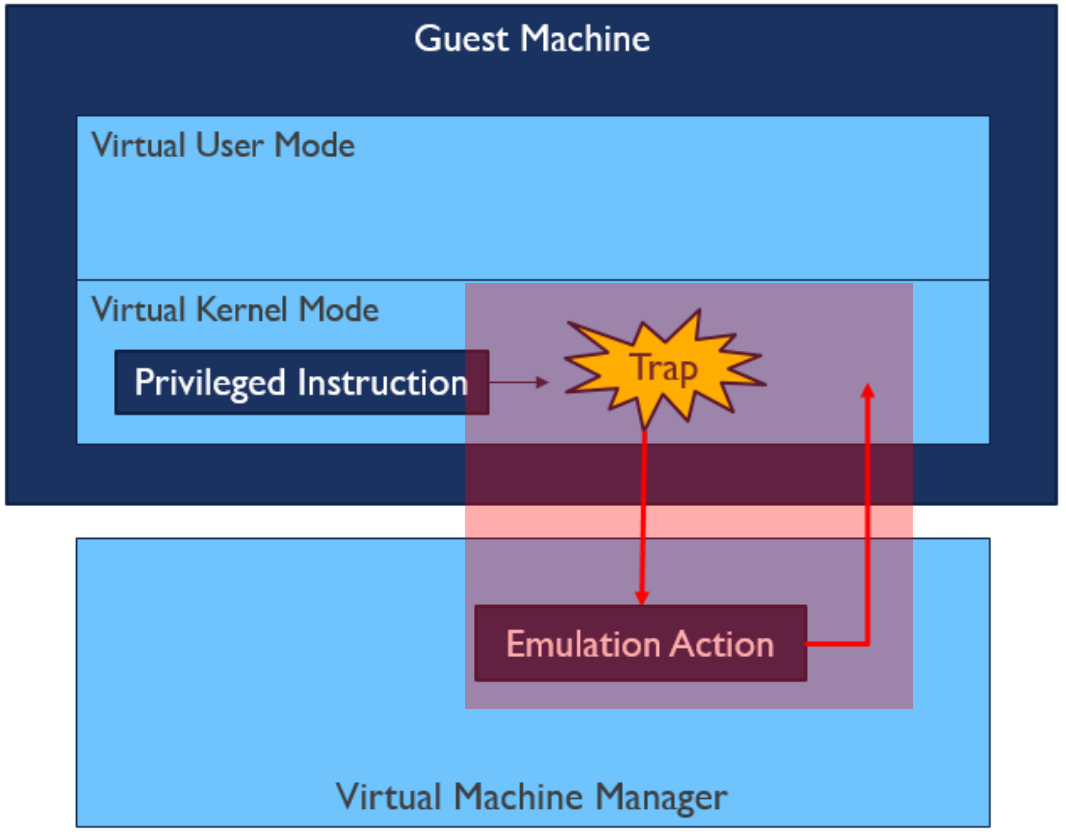
Q: Why is this an issue of the Trap and Emulate?

There is no trap!

Trap and emulate will simply not work because there is no trap in the first place.

LIMITATIONS OF TRAP AND EMULATE

- Sometimes the boundary between Kernel/User mode is not well defined.
- Processors were not designed with virtualization in mind.
- Some instructions are not "privileged" but rather perform differently when in kernel mode.
- Example: x86 `popf`
  - Loads portion of flag register from the contents of the stack.
  - If the CPU is in privileged mode, all of the flags are replaced from the stack pop.



Q: Why is this an issue of the Trap and Emulate?

There is no trap!

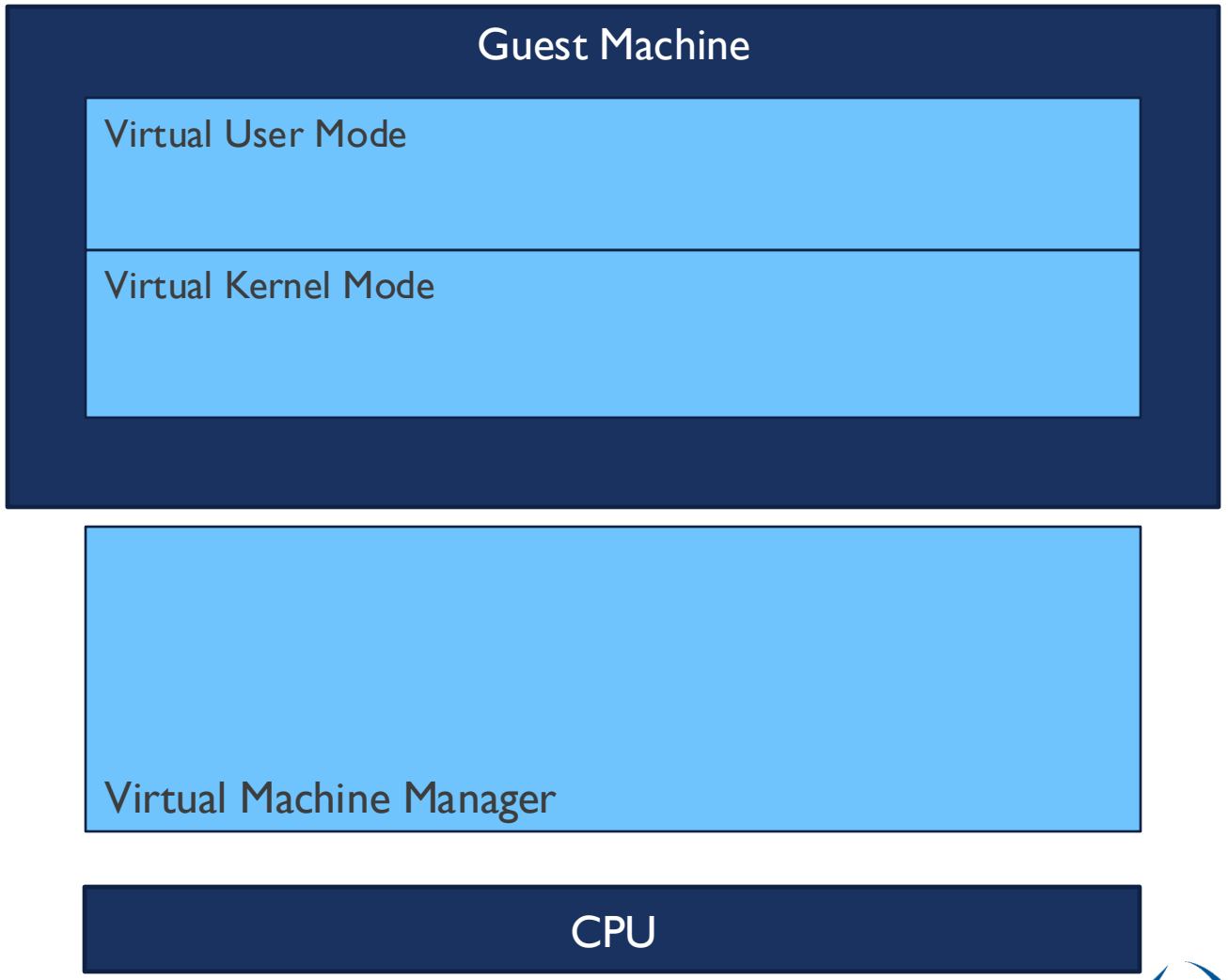
Trap and emulate will simply not work because there is no trap in the first place.

BINARY TRANSLATION

- Solution: Binary Translation

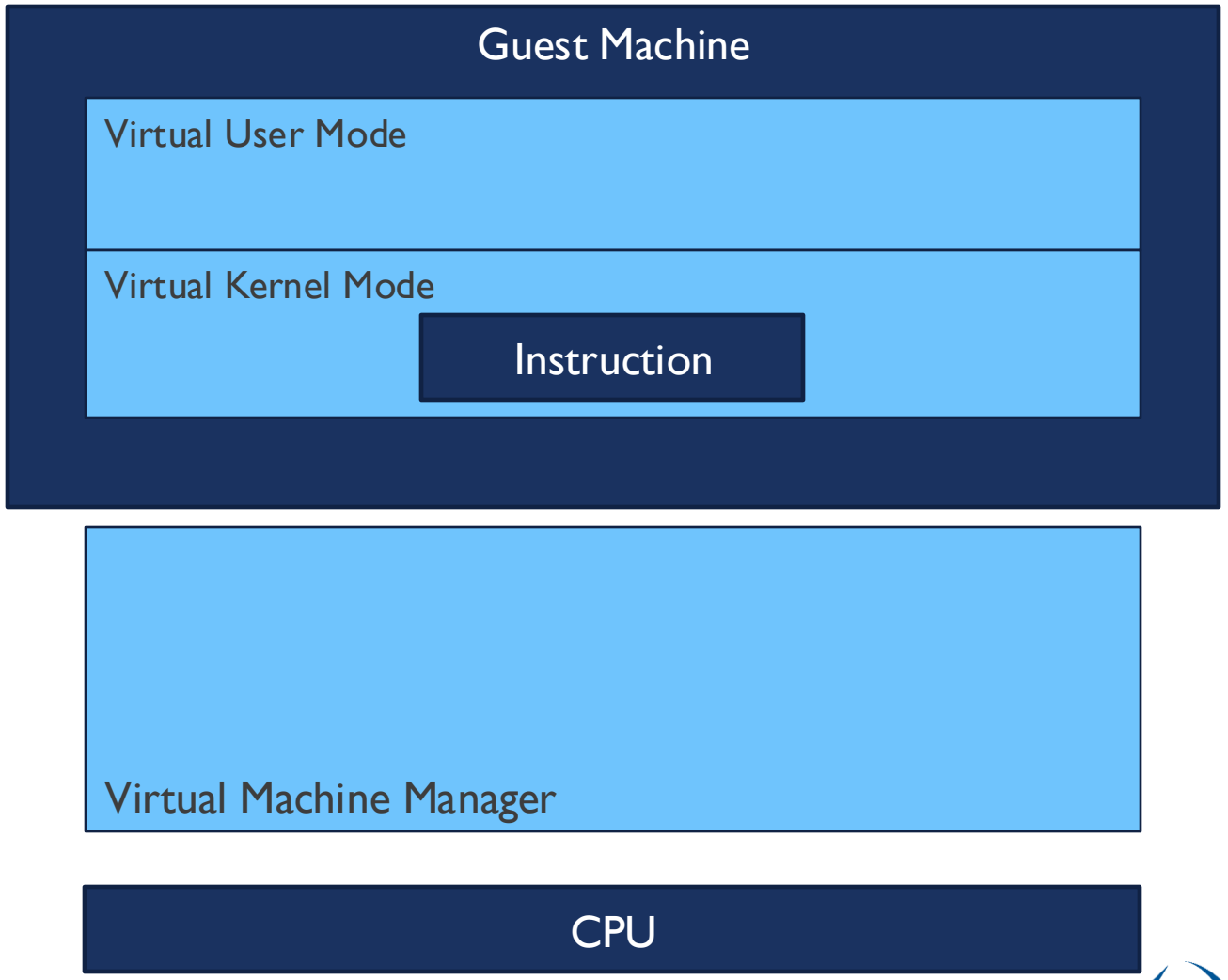
BINARY TRANSLATION

- Solution: Binary Translation



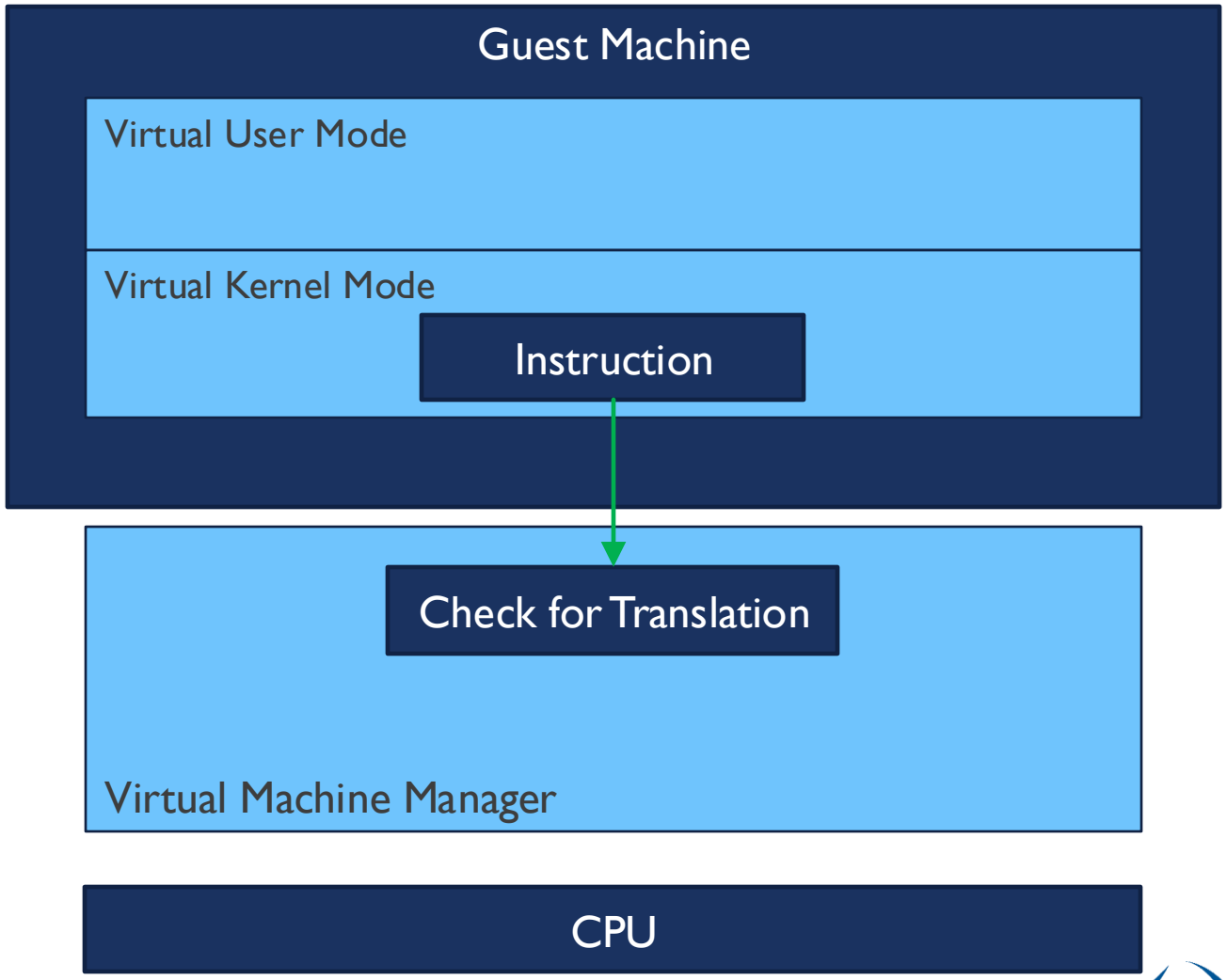
BINARY TRANSLATION

- Solution: Binary Translation



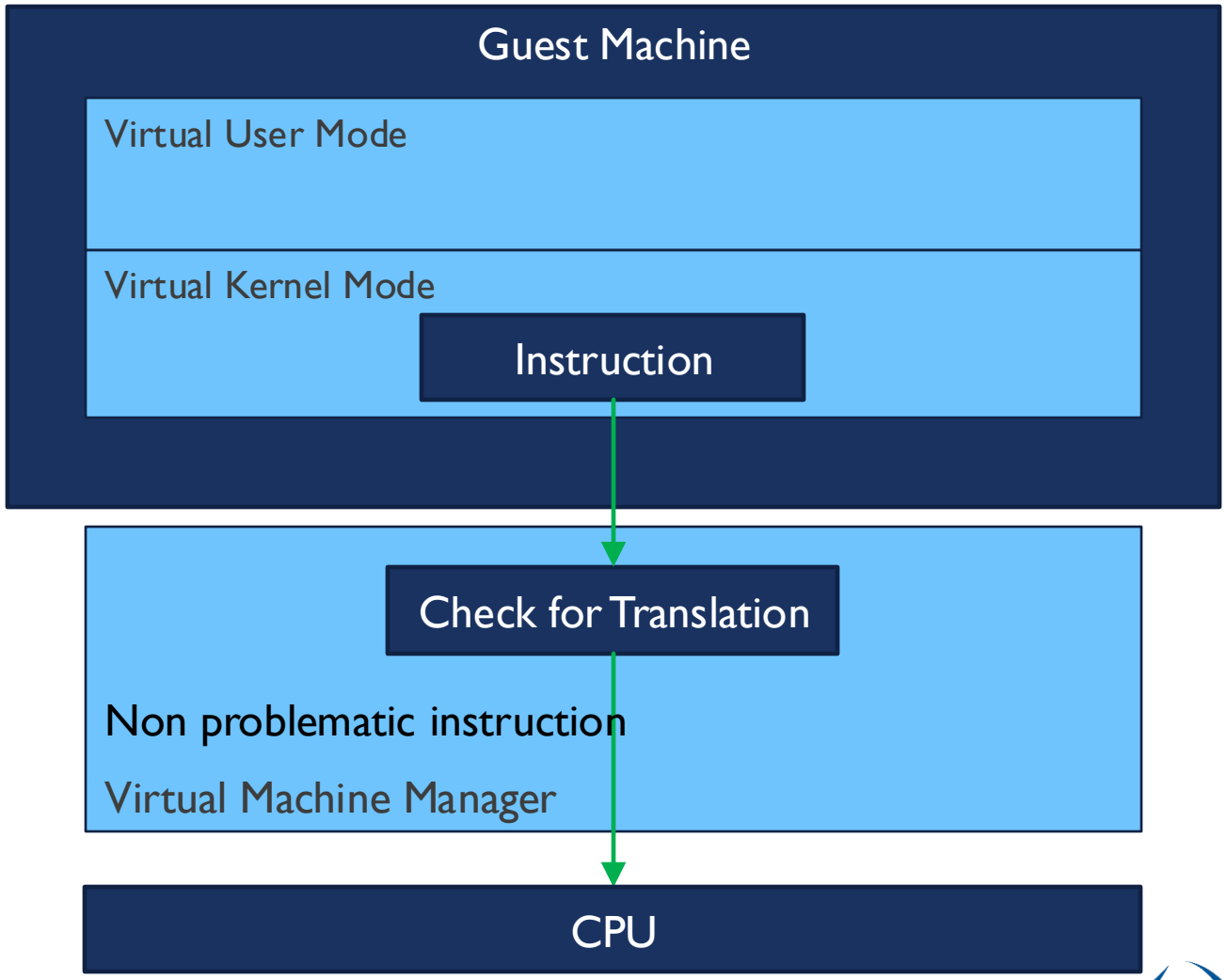
BINARY TRANSLATION

- Solution: Binary Translation



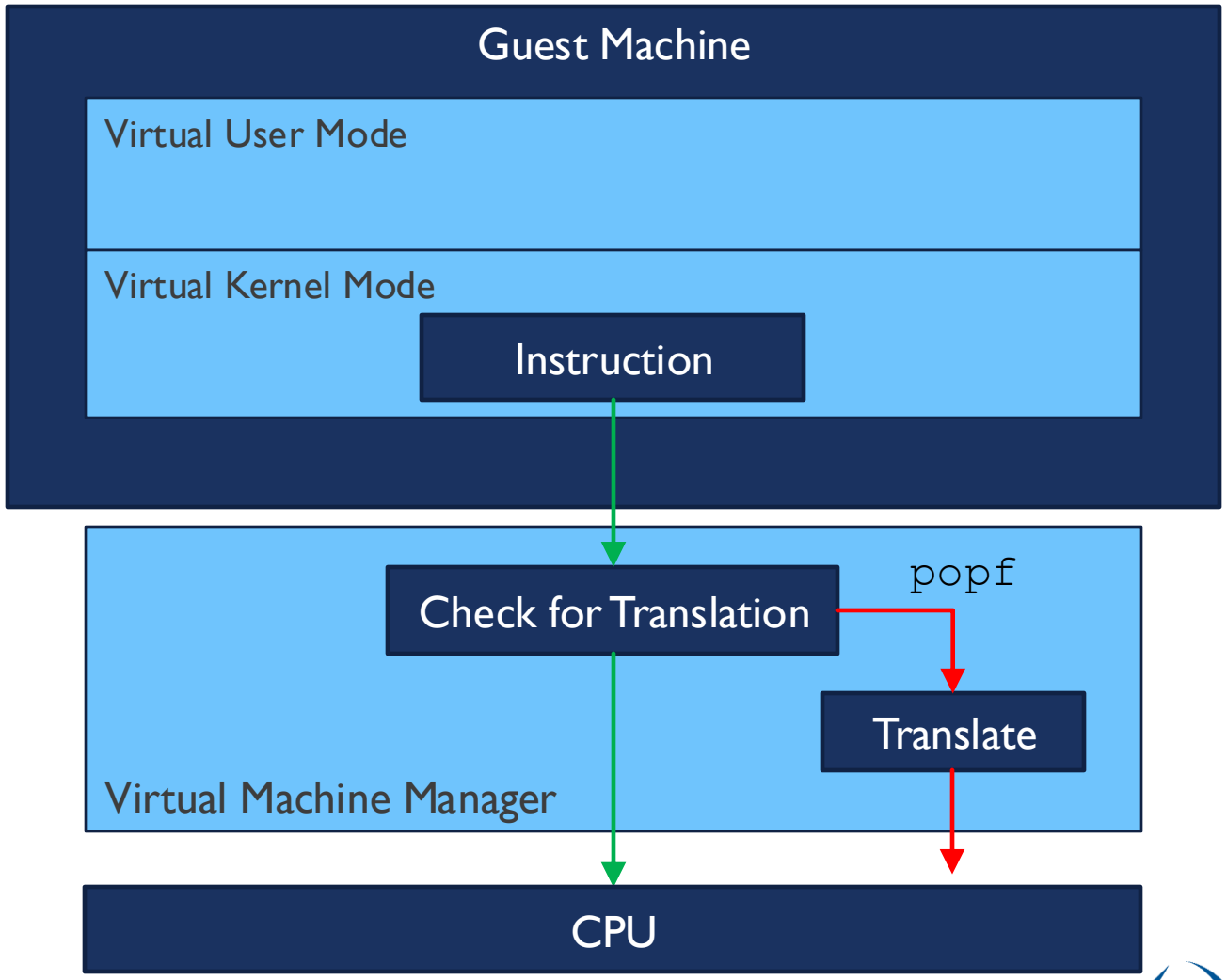
BINARY TRANSLATION

- Solution: Binary Translation



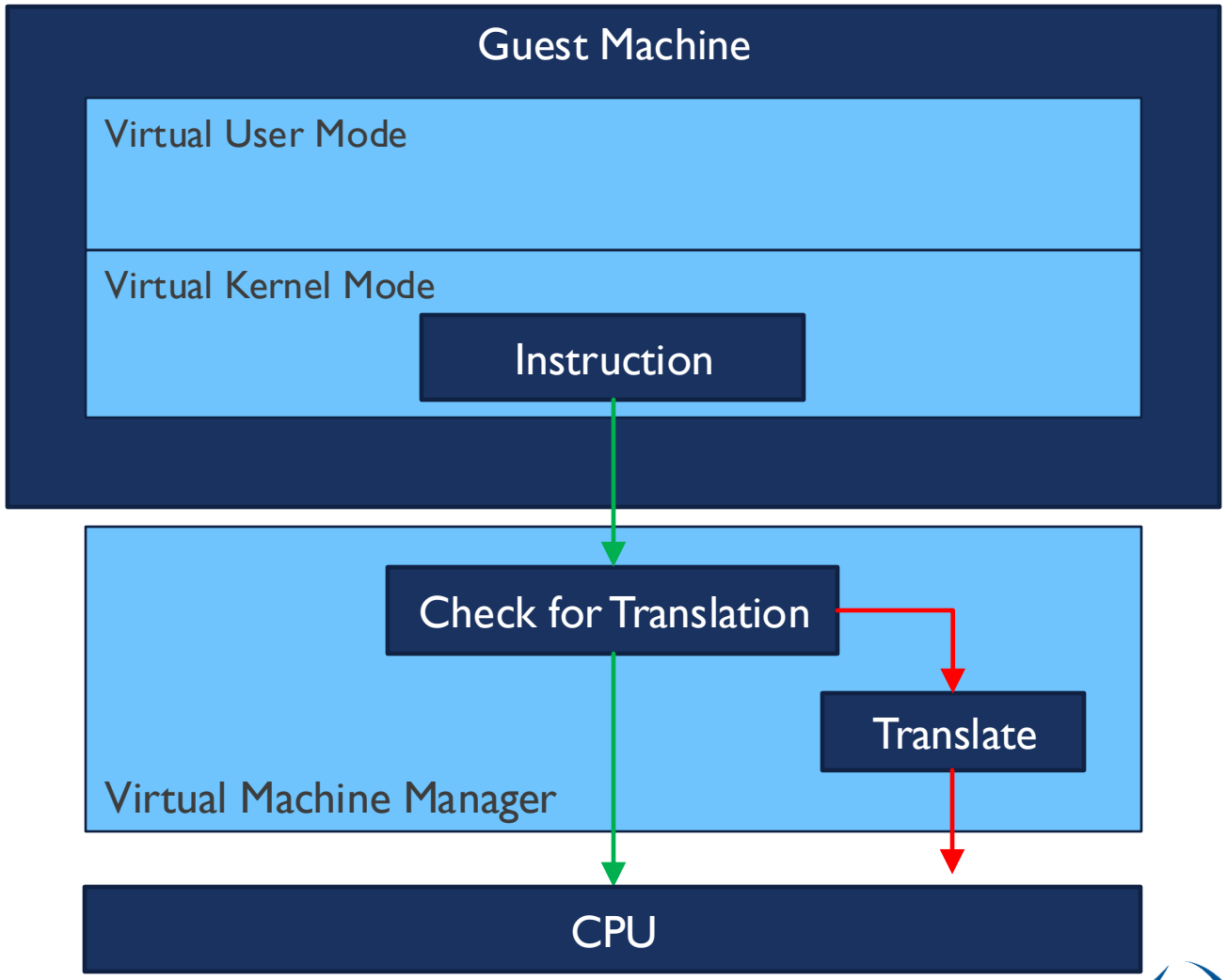
BINARY TRANSLATION

- Solution: Binary Translation



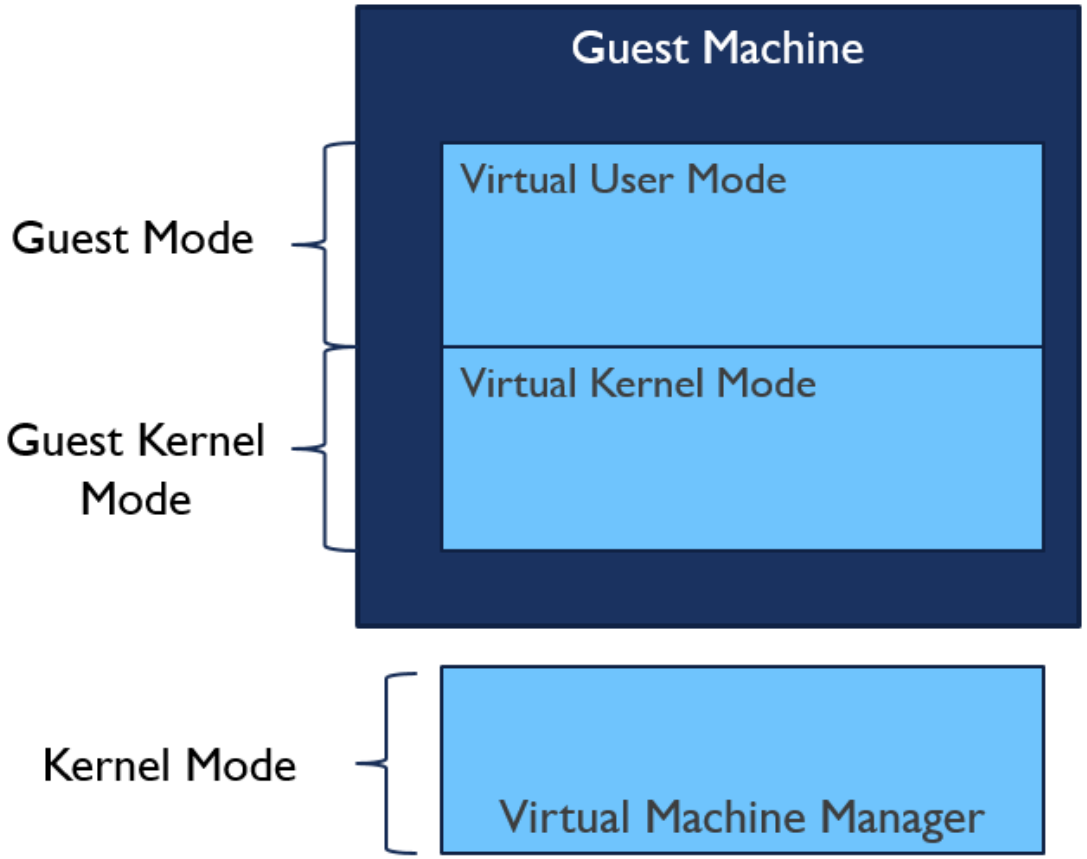
BINARY TRANSLATION

- Solution: Binary Translation



HARDWARE SUPPORT

- All virtualization needs some HW support
- More support -> more feature rich, stable, better performance of guests
- Intel added new **VT-x** instructions in 2005 and AMD the **AMD-V** instructions in 2006
  - CPUs with these instructions remove need for binary translation
  - Generally define more CPU modes – "guest" and "host"
- HW support for Nested Page Tables, DMA, interrupts as well over time



PARAVIRTUALIZATION

- We always assumed that guest systems need to be identical to real systems …
- OS running on Guest Machine are not aware they are running on a virtual machine … they are an exact replica of the OS running on real hardware.

PARAVIRTUALIZATION

- We always assumed that guest systems need to be identical to real systems …
- OS running on Guest Machine are not aware they are running on a virtual machine … they are an exact replica of the OS running on real hardware.
- This creates a lot of extra work for VMM to maintain this illusion. Maybe we should not maintain it.

PARAVIRTUALIZATION

- We always assumed that guest systems need to be identical to real systems …
- OS running on Guest Machine are not aware they are running on a virtual machine … they are an exact replica of the OS running on real hardware.
- This creates a lot of extra work for VMM to maintain this illusion. Maybe we should not maintain it.
- Solution: modify the guest OS to allow for easier interface with the VMM and hardware.
- Guest OS is no longer a replica of real OS. It has special kernel calls for memory and I/O.
- Example Xen VMM, showed increased performance.

PARAVIRTUALIZATION

- We always assumed that guest systems need to be identical to real systems …
- OS running on Guest Machine are not aware they are running on a virtual machine … they are an exact replica of the OS running on real hardware.
- This creates a lot of extra work for VMM to maintain this illusion. Maybe we should not maintain it.
- Solution: modify the guest OS to allow for easier interface with the VMM and hardware.
- Guest OS is no longer a replica of real OS. It has special kernel calls for memory and I/O.
- Example Xen VMM, showed increased performance.

Q: What is the disadvantage?

PARAVIRTUALIZATION

- We always assumed that guest systems need to be identical to real systems …
- OS running on Guest Machine are not aware they are running on a virtual machine … they are an exact replica of the OS running on real hardware.
- This creates a lot of extra work for VMM to maintain this illusion. Maybe we should not maintain it.
- Solution: modify the guest OS to allow for easier interface with the VMM and hardware.
- Guest OS is no longer a replica of real OS. It has special kernel calls for memory and I/O.
- Example Xen VMM, showed increased performance.

Q: What is the disadvantage?

- Can't use unmodified OS.
- Every OS must be modified before running on the virtual machine.

- With the advanced hardware support for virtualization … Para virtualization is no longer needed.
- Hardware-accelerated virtualization now shows similar performance: no need to modify OS anymore.
- Still used in some specialized system.

PAGING VIRTUALIZATION

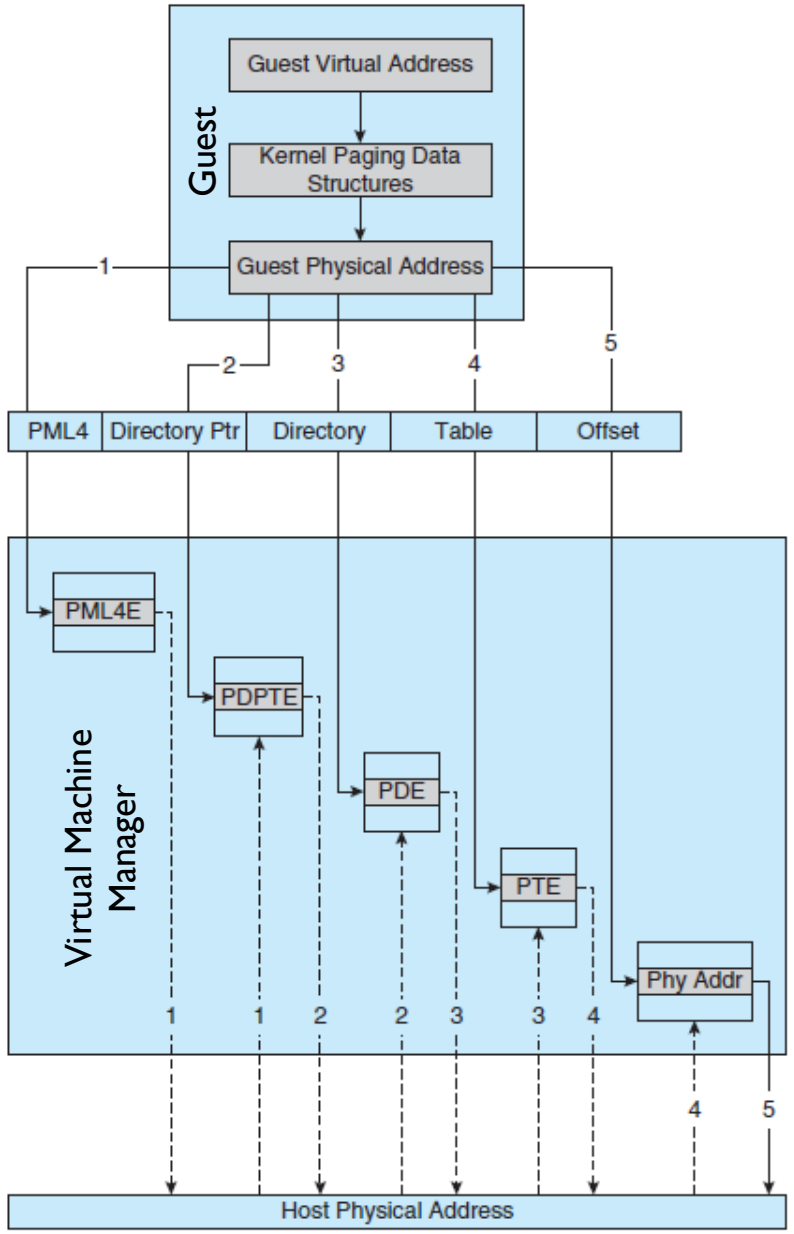
- How can VMM keep page-table state for both guests believing they control the page tables and VMM that does control the tables?

PAGING VIRTUALIZATION

- How can VMM keep page-table state for both guests believing they control the page tables and VMM that does control the tables?
- Common method (for trap-and-emulate and binary translation) is **nested page tables** (**NPTs**)

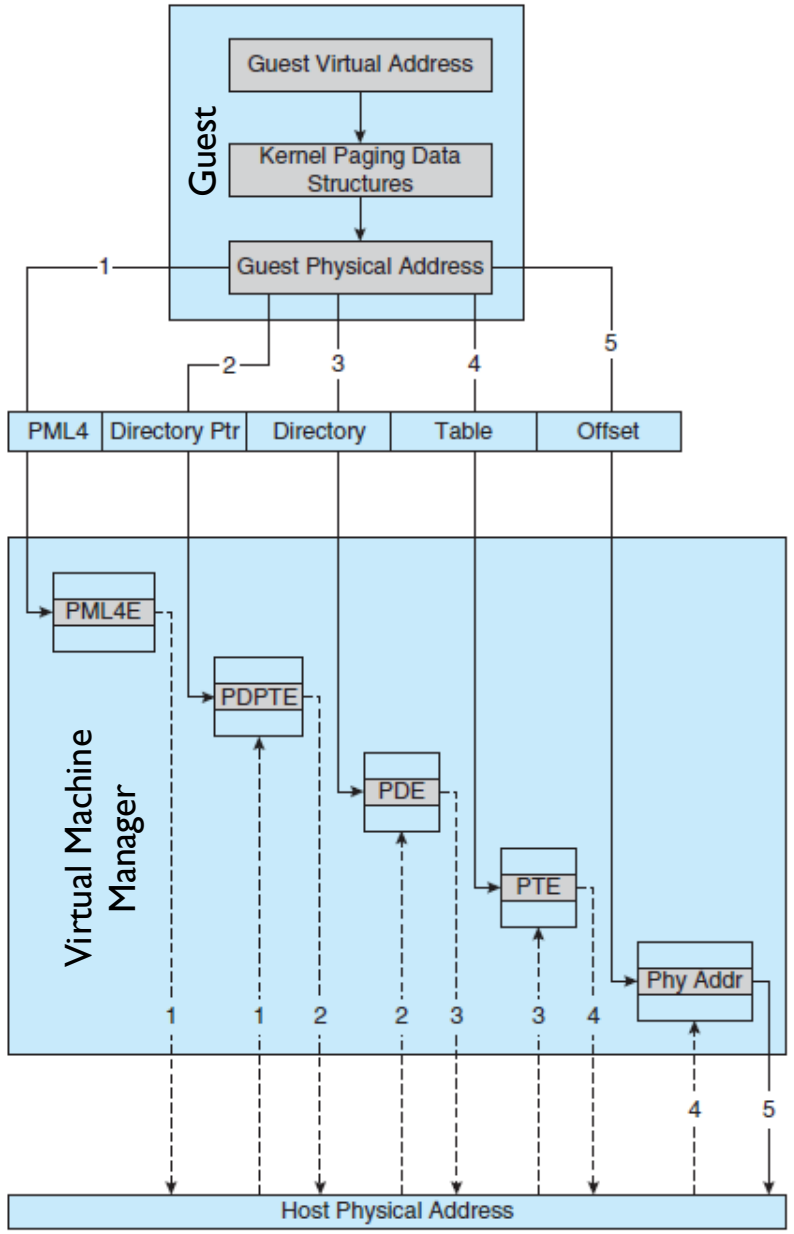
PAGING VIRTUALIZATION

- How can VMM keep page-table state for both guests believing they control the page tables and VMM that does control the tables?
- Common method (for trap-and-emulate and binary translation) is **nested page tables** (**NPTs**)



PAGING VIRTUALIZATION

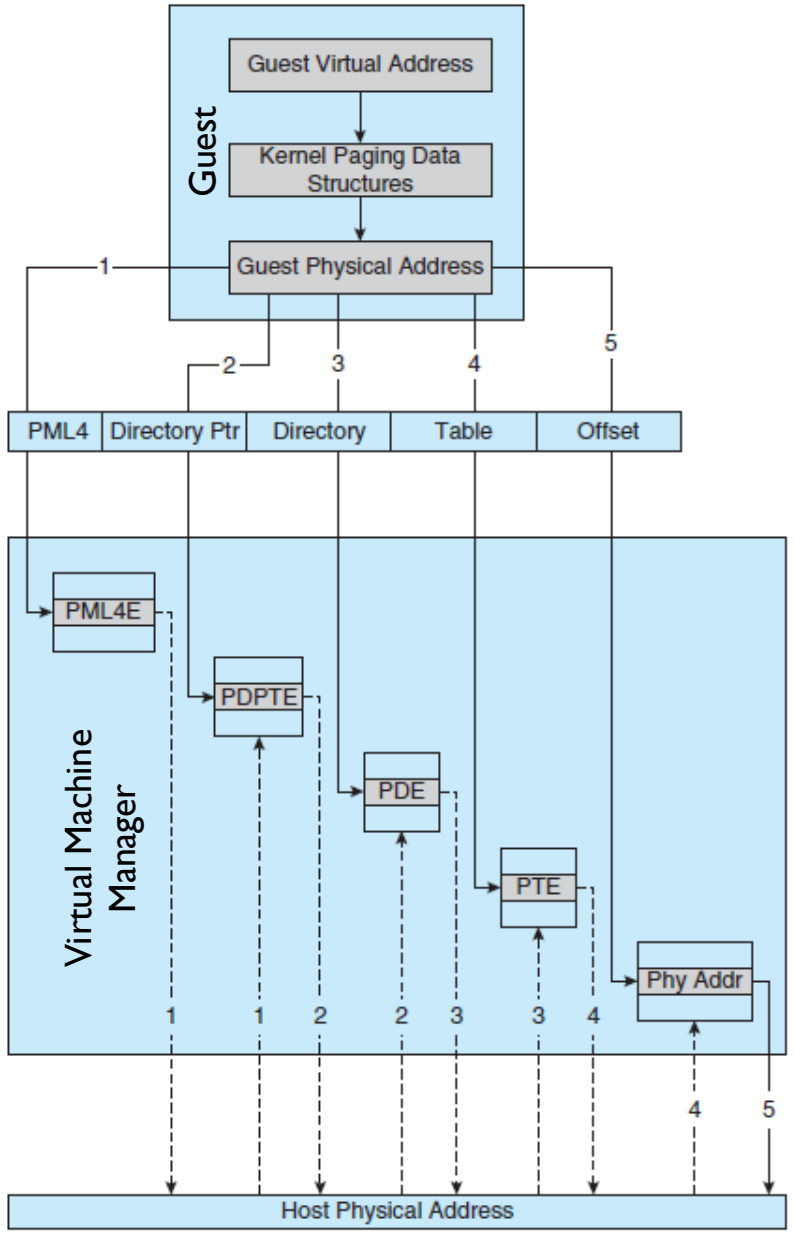
- How can VMM keep page-table state for both guests believing they control the page tables and VMM that does control the tables?
- Common method (for trap-and-emulate and binary translation) is **nested page tables** (**NPTs**)
  - Each guest maintains page tables to translate virtual to physical addresses
  - VMM maintains per guest NPTs to represent guest's page-table state
    - Just as VCPU stores guest CPU state
  - When guest on CPU
    - VMM makes that guest's NPTs the active system page tables



PAGING VIRTUALIZATION

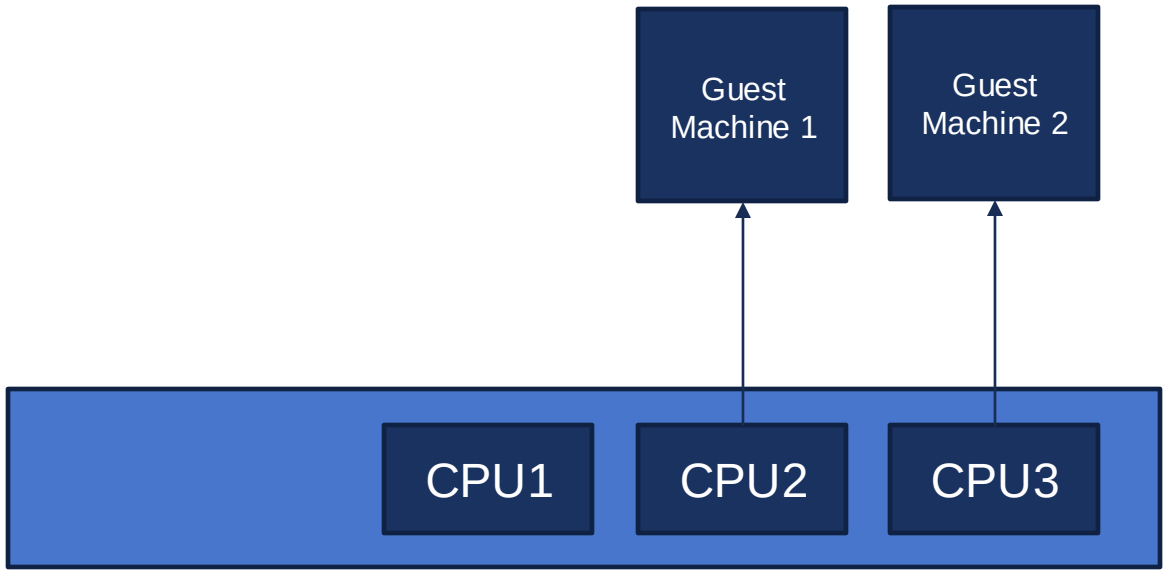
- How can VMM keep page-table state for both guests believing they control the page tables and VMM that does control the tables?
- Common method (for trap-and-emulate and binary translation) is **nested page tables** (**NPTs**)
  - Each guest maintains page tables to translate virtual to physical addresses
  - VMM maintains per guest NPTs to represent guest's page-table state
    - Just as VCPU stores guest CPU state
  - When guest on CPU -> VMM makes that guest's NPTs the active system page tables

Why is it needed?
The guest doesn't have access to physical memory.



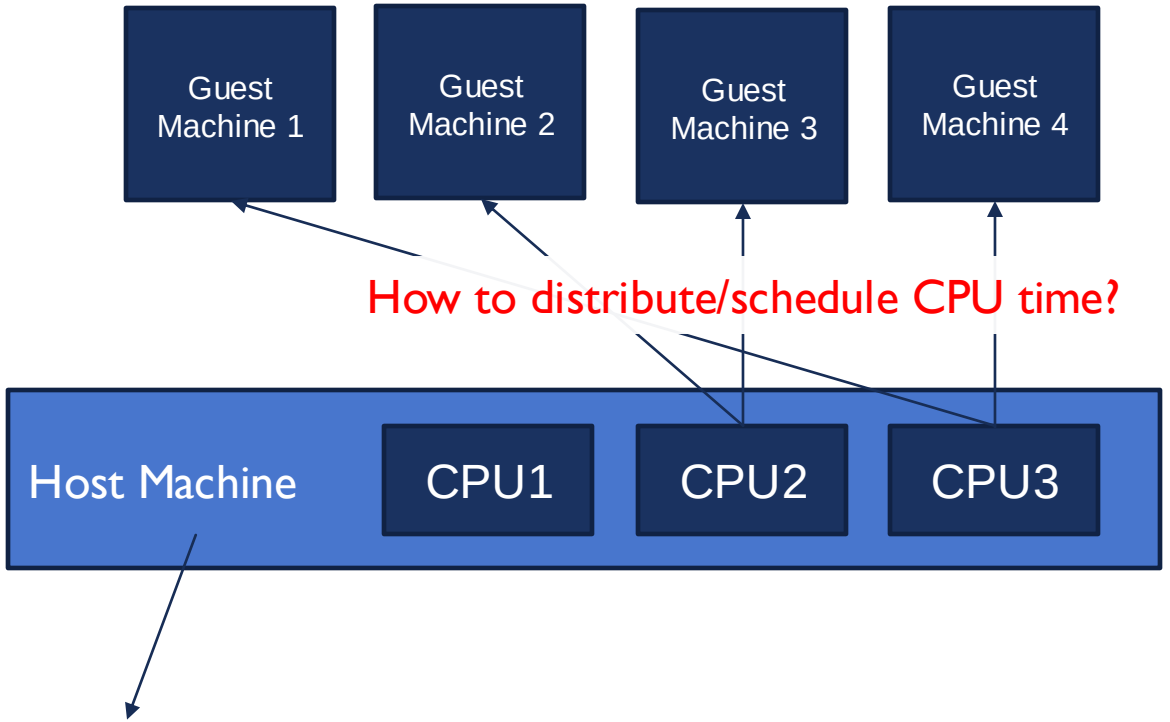
CPU MAPPING

- **How Are CPUs Mapped to virtual Machines?**



CPU MAPPING

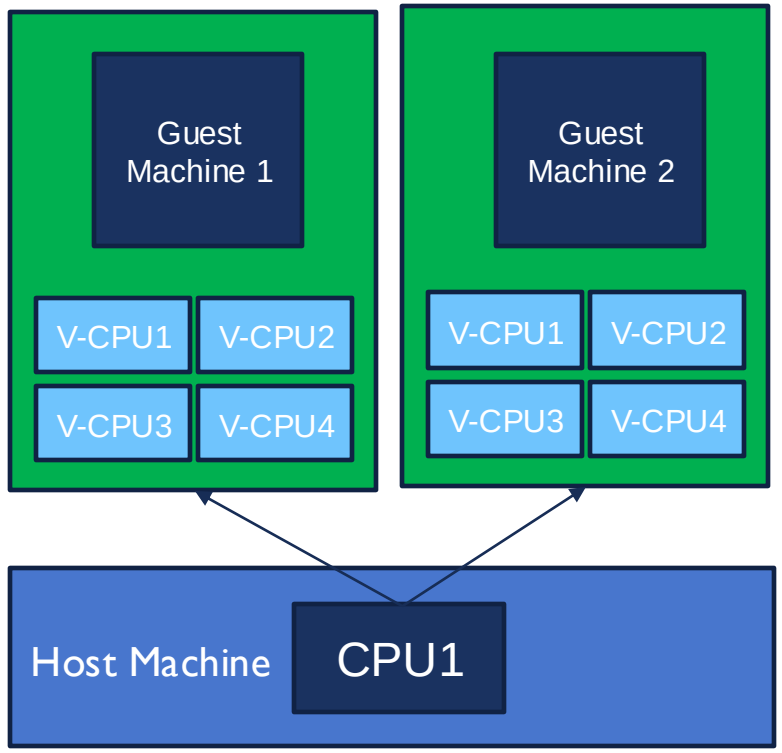
- What if we have too many virtual machines?



Host machine also needs access to CPU for VMM and maybe other applications …

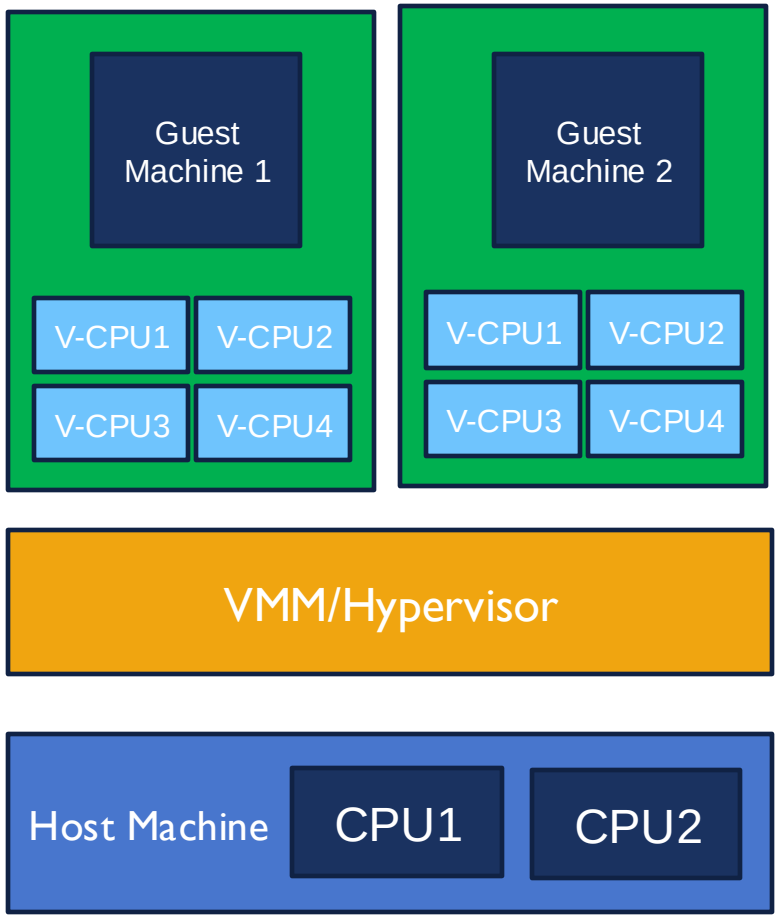
CPU MAPPING

- Sometimes VMs have multicore while the actual system has a single CPU (single core).



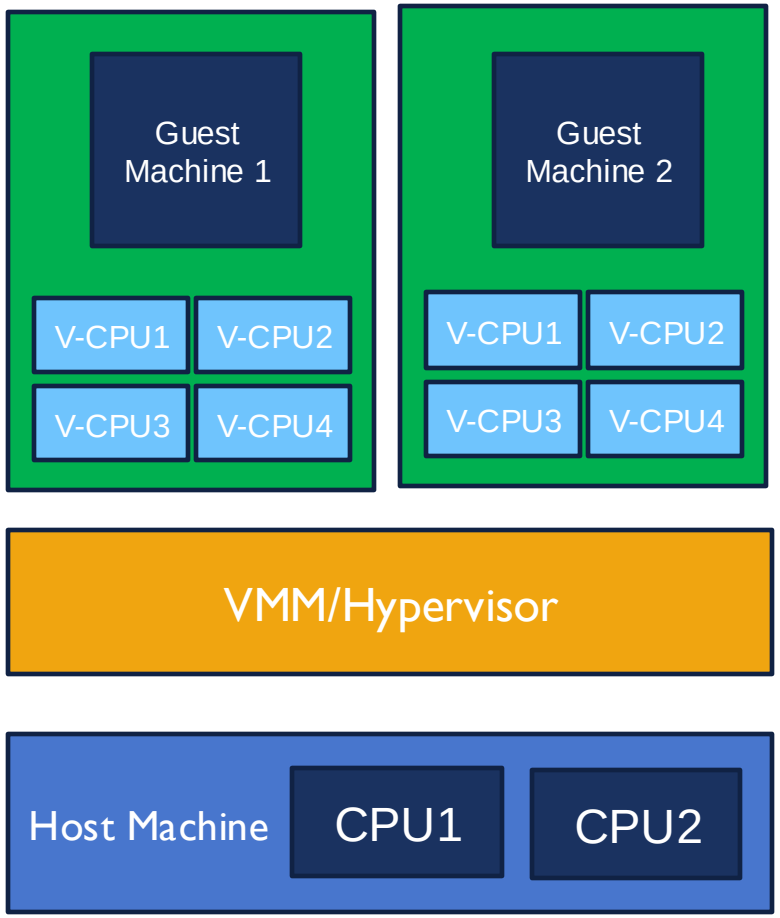
CPU MAPPING

- If there are enough CPUs … just assign one to one.
- Assignment does not mean dedicated/exclusive access. It just indicates that Guest Machine 1 code can run only on CPU1.
- CPU1 may run other code as well, like VMM code or some other non-virtualized code.
- Hypervisor does all Guest Machine scheduling.



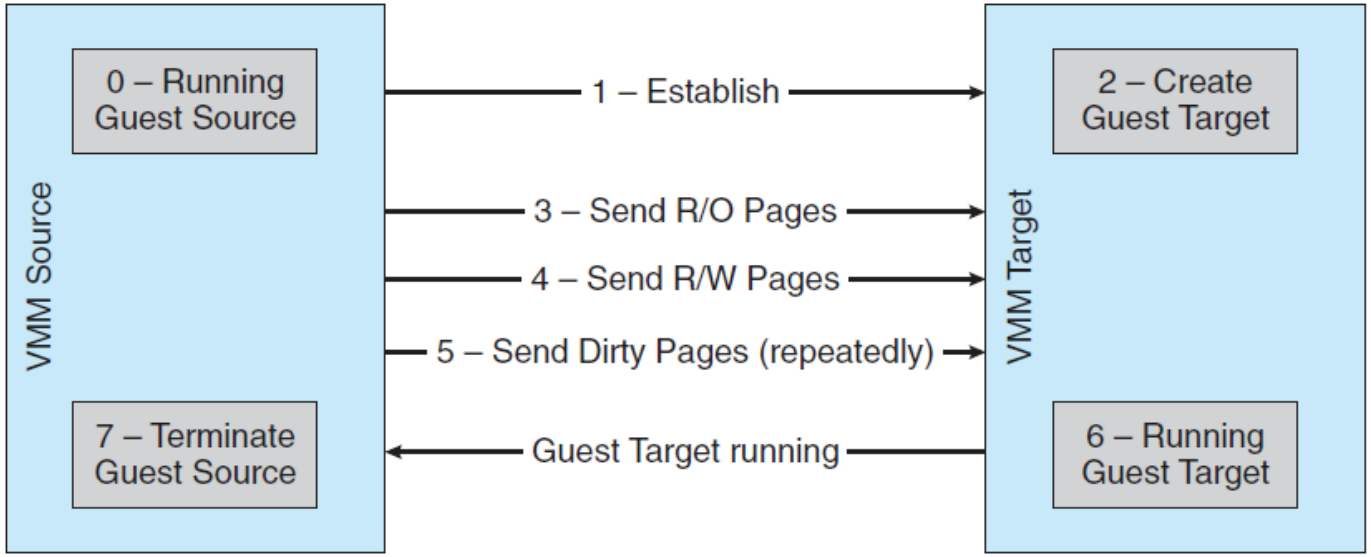
CPU MAPPING

- If there are enough CPUs … just assign one to one.
- Assignment does not mean dedicated/exclusive access. It just indicates that Guest Machine 1 code can run only on CPU1.
- CPU1 may run other code as well, like VMM code or some other non-virtualized code.
- Hypervisor does all Guest Machine scheduling.
- Overcommitment: Virtual CPU count exceeds real CPU by a large margin. Causes performance degradation.



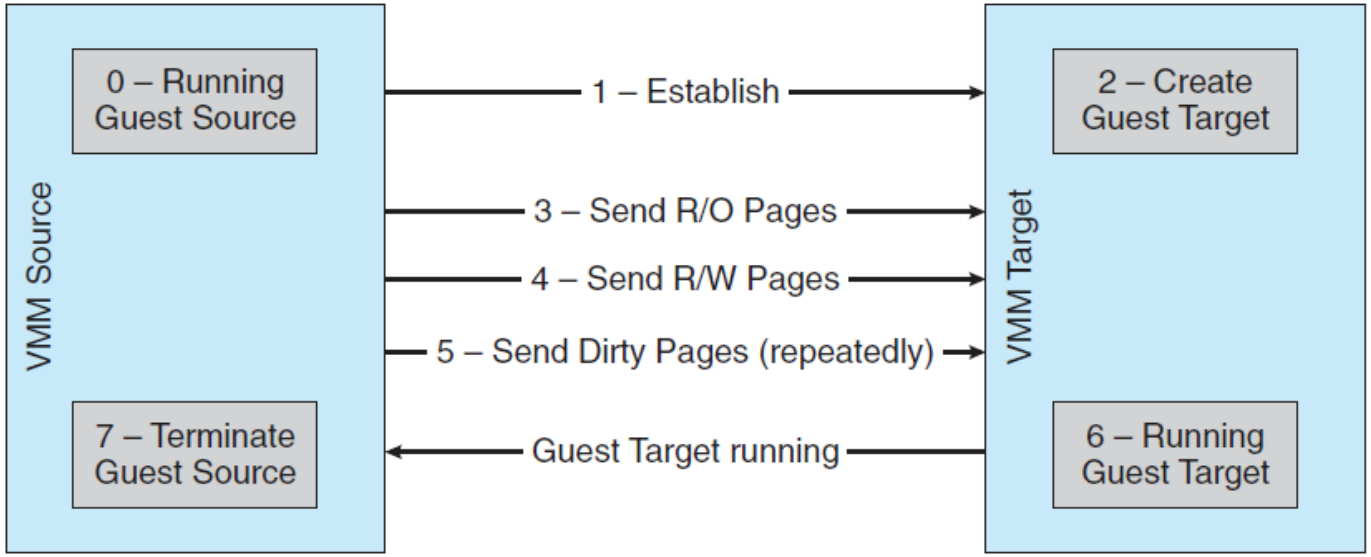
LIVE MIGRATION

- Running guest can be moved between systems, without interrupting user access to the guest or its apps.
- Very useful for resource management, maintenance downtime windows, etc



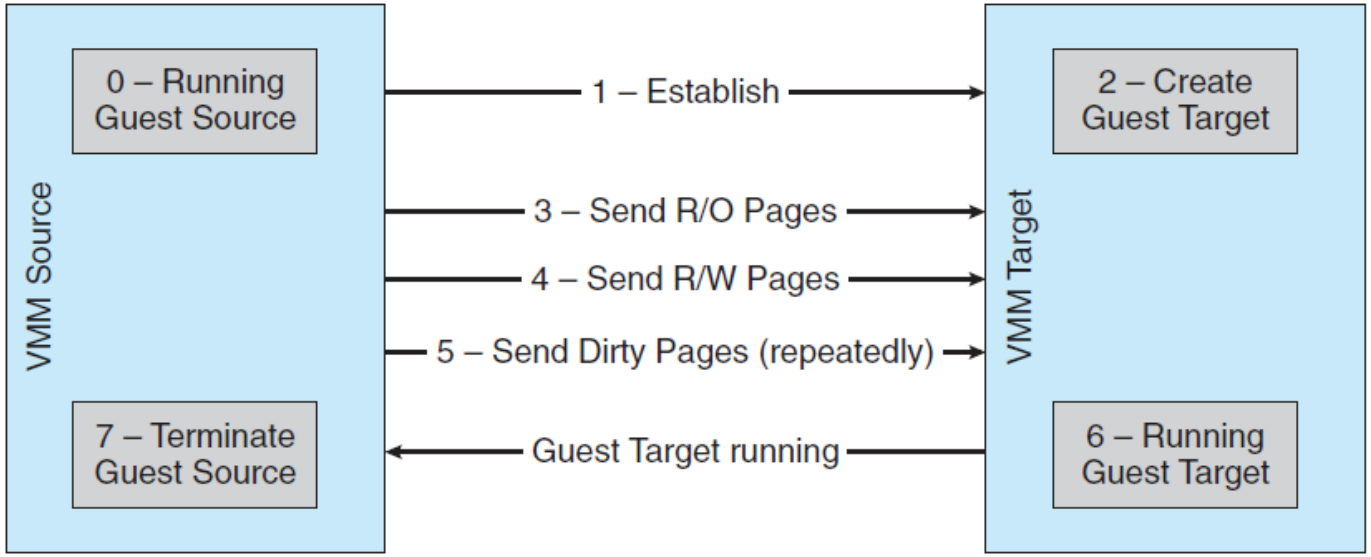
LIVE MIGRATION

1. The source VMM establishes a connection with the target VMM
2. The target creates a new guest by creating a new VCPU, etc
3. The source sends all read-only guest memory pages to the target



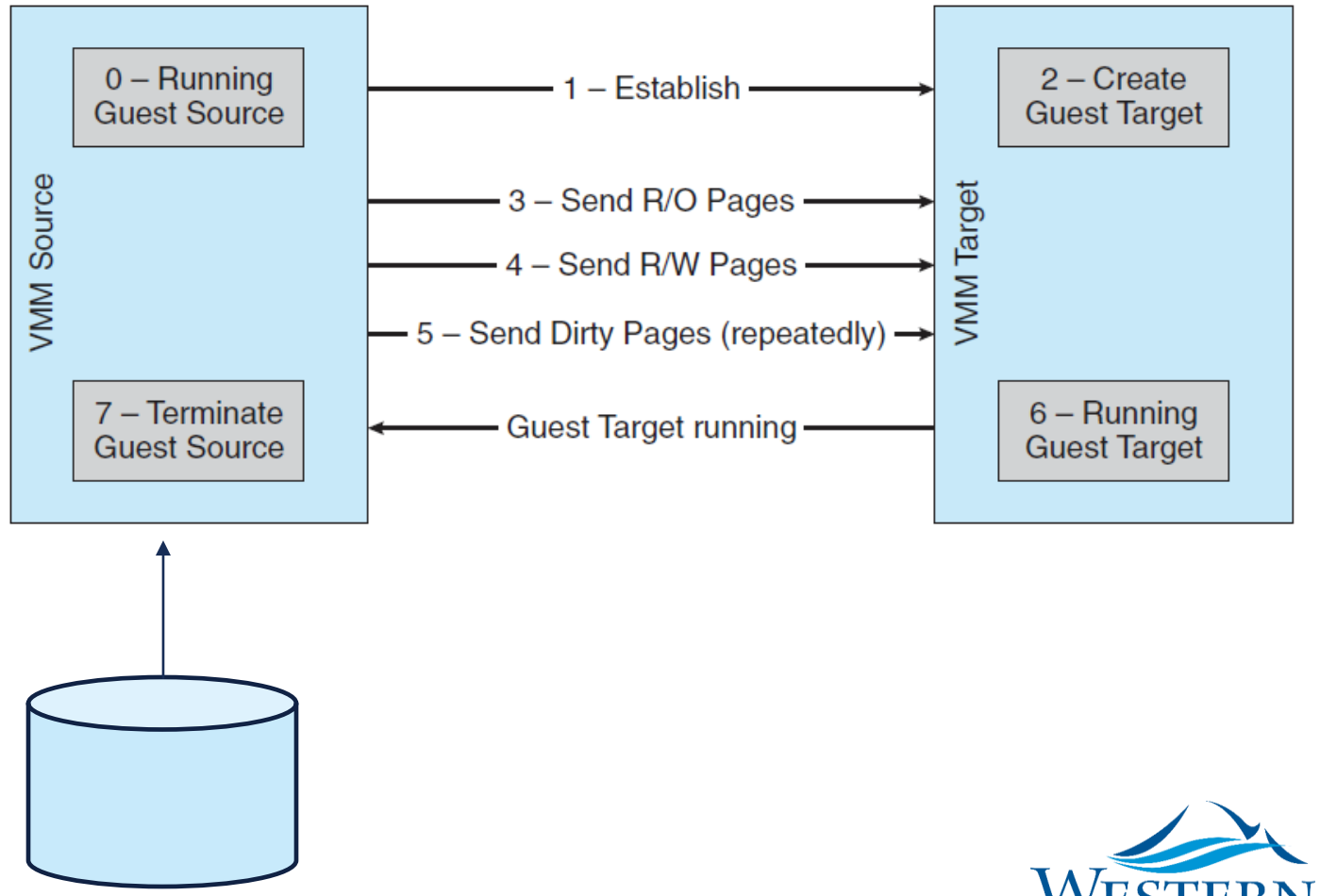
LIVE MIGRATION

4. The source sends all read-write pages to the target, marking them as clean
5. The source repeats step 4, as during that step some pages were probably modified by the guest and are now dirty
6. When cycle of steps 4 and 5 becomes very short, source VMM freezes guest, sends VCPU's final state, sends other state details, sends final dirty pages, and tells target to start running the guest



LIVE MIGRATION

Q: What about the system disk/mass storage?



LIVE MIGRATION

Q: What about the system disk/mass storage?

Disk is simply accessed remotely. Moving cost is too time consuming.