6.1800 Spring 2024

Lecture #6: Virtual Machines

even more virtualization, plus kernel designs

Katrina LaCurts | lacurts@mit.edu | 6.1800 2024

6.1800 in the news

The Two-Decade Fight for Two Letters on the Internet

The South Pacific island of Niue says it was cheated out of .nu, a domain that turned out to be very lucrative on the other side of the world.

Share full article

A village on Niue, a South Pacific island of about 100 square miles, roughly the same area as Lincoln, Neb. Jill Ferry/Getty Images

https://www.nytimes.com/2024/02/21/world/asia/niue-nu-domain-sweden.html

6.1800 in the news

The South Pacific island of Niue is one the most remote places in the world. Its closest neighbors, Tonga and American Samoa, are hundreds of miles away. The advent of the internet promised, in a small way, to make Niue and its 2,000 or so residents more connected to the rest of the world.

In the late 1990s, an American businessman offered to hook up the island to the internet. All he wanted in exchange was the right to control the .nu suffix that Niue was assigned for its web addresses. The domain did not seem as lucrative as .tv — which was slotted to Tuvalu, another South Pacific nation — and the leaders of Niue (pronounced New-ay) signed off on the deal. But the two sides were soon at odds.

Now, after more than two decades of back and forth, the disagreement is finally nearing a resolution in a court of law. Disputes over domain names were not uncommon during the internet's infancy but experts are hard pressed to recall one that has lasted this long.

It turned out that .nu was, in fact, very valuable. "Nu" means now in Swedish, Danish and Dutch, and thousands of Scandinavians registered websites with that suffix, creating a steady business for Niue's business partner, Bill Semich.

The Two-Decade Fight for Two Letters on the Internet

The South Pacific island of Niue says it was cheated out of .nu, a domain that turned out to be very lucrative on the other side of the world.

Share full article

A village on Niue, a South Pacific island of about 100 square miles, roughly the same area as Lincoln, Neb. Jill Ferry/Getty Images

6.1800 in the news

"We are victims of digital colonialism," Prime Minister Dalton Tagelagi of Niue said over a crackling video link from his office in the capital of Alofi. "This domain, the .nu, recognizes Niue as a sovereign country. This is how important it is to our identity."

Critics question that assessment, as there is formally no such thing as sovereignty in cyberspace, only administrative zones that divide the web into domains like .nu and, for instance, the .nz suffix assigned to New Zealand.

Winning the case could help ensure the long-term survival of Niue, Mr. Tagelagi said. The island's population is now about a third of what it was in the 1960s, and the empty homes that dot the island are a reminder of the people who left for better economic opportunities. A victory could help fund its bid to join the United Nations, similar to how Tuvalu obtained U.N. membership after monetizing .tv.

If Niue manages to get .nu back, it could bring in up to $2 million in revenue a year, according to Par Brumark, a domain name expert who is acting on Niue's behalf in the Swedish case.

https://www.nytimes.com/2024/02/21/world/asia/niue-nu-domain-sweden.html

6.1800 in the news

in the case of DNS, names have meaning outside of the system, even if they were only originally intended to denote "administrative zones that divide the web into domains"

"We are victims of digital colonialism," Prime Minister Dalton Tagelagi of Niue said over a crackling video link from his office in the capital of Alofi. "This domain, the .nu, recognizes Niue as a sovereign country. This is how important it is to our identity."

Critics question that assessment, as there is formally no such thing as sovereignty in cyberspace, only administrative zones that divide the web into domains like .nu and, for instance, the .nz suffix assigned to New Zealand.

Winning the case could help ensure the long-term survival of Niue, Mr. Tagelagi said. The island's population is now about a third of what it was in the 1960s, and the empty homes that dot the island are a reminder of the people who left for better economic opportunities. A victory could help fund its bid to join the United Nations, similar to how Tuvalu obtained U.N. membership after monetizing .tv.

If Niue manages to get .nu back, it could bring in up to $2 million in revenue a year, according to Par Brumark, a domain name expert who is acting on Niue's behalf in the Swedish case.

https://www.nytimes.com/2024/02/21/world/asia/niue-nu-domain-sweden.html

6.1800 in the news

in the case of DNS, names have meaning outside of the system, even if they were only originally intended to denote "administrative zones that divide the web into domains"

many of the decisions we make when we build our systems **impact** people who might not even be aware of the system

"We are victims of digital colonialism," Prime Minister Dalton Tagelagi of Niue said over a crackling video link from his office in the capital of Alofi. "This domain, the .nu, recognizes Niue as a sovereign country. This is how important it is to our identity."

Critics question that assessment, as there is formally no such thing as sovereignty in cyberspace, only administrative zones that divide the web into domains like .nu and, for instance, the .nz suffix assigned to New Zealand.

Winning the case could help ensure the long-term survival of Niue, Mr. Tagelagi said. The island's population is now about a third of what it was in the 1960s, and the empty homes that dot the island are a reminder of the people who left for better economic opportunities. A victory could help fund its bid to join the United Nations, similar to how Tuvalu obtained U.N. membership after monetizing .tv.

If Niue manages to get .nu back, it could bring in up to $2 million in revenue a year, according to Par Brumark, a domain name expert who is acting on Niue's behalf in the Swedish case.

https://www.nytimes.com/2024/02/21/world/asia/niue-nu-domain-sweden.html

operating systems enforce modularity on a single machine using **virtualization**

in order to enforce modularity + have an effective operating system, a few things need to happen

1. programs shouldn't be able to refer to (and corrupt) each others' **memory** → **virtual memory**
2. programs should be able to **communicate** with each other → **bounded buffers** (virtualize communication links)
3. programs should be able to **share a CPU** without one program halting the progress of the others → **threads** (virtualize processors)

operating systems enforce modularity on a single machine using **virtualization**

in order to enforce modularity + have an effective operating system, a few things need to happen

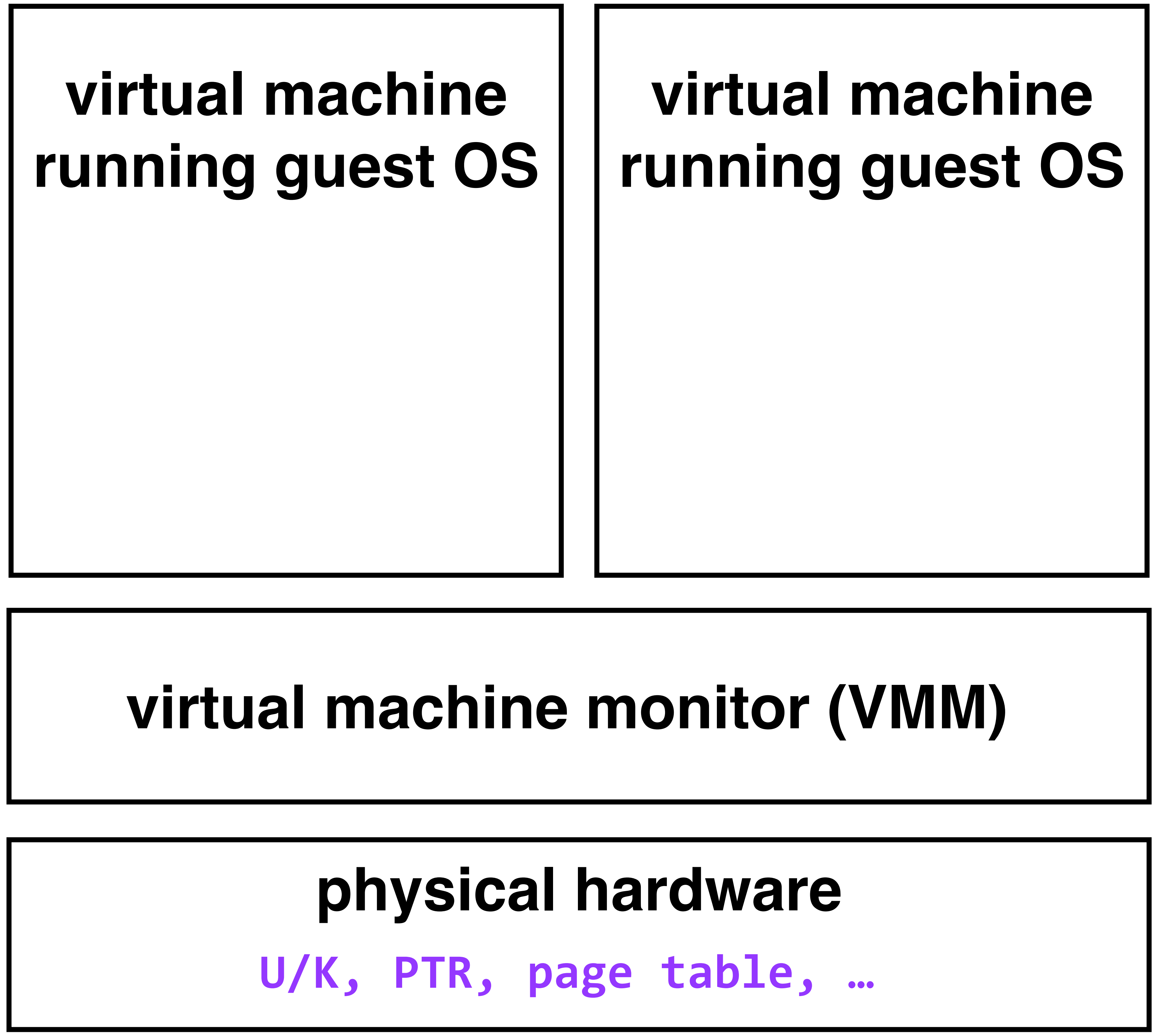
1. programs shouldn't be able to refer to (and corrupt) each others' **memory** → **virtual memory**

2. programs should be able to **communicate** with each other → **bounded buffers** (virtualize communication links)

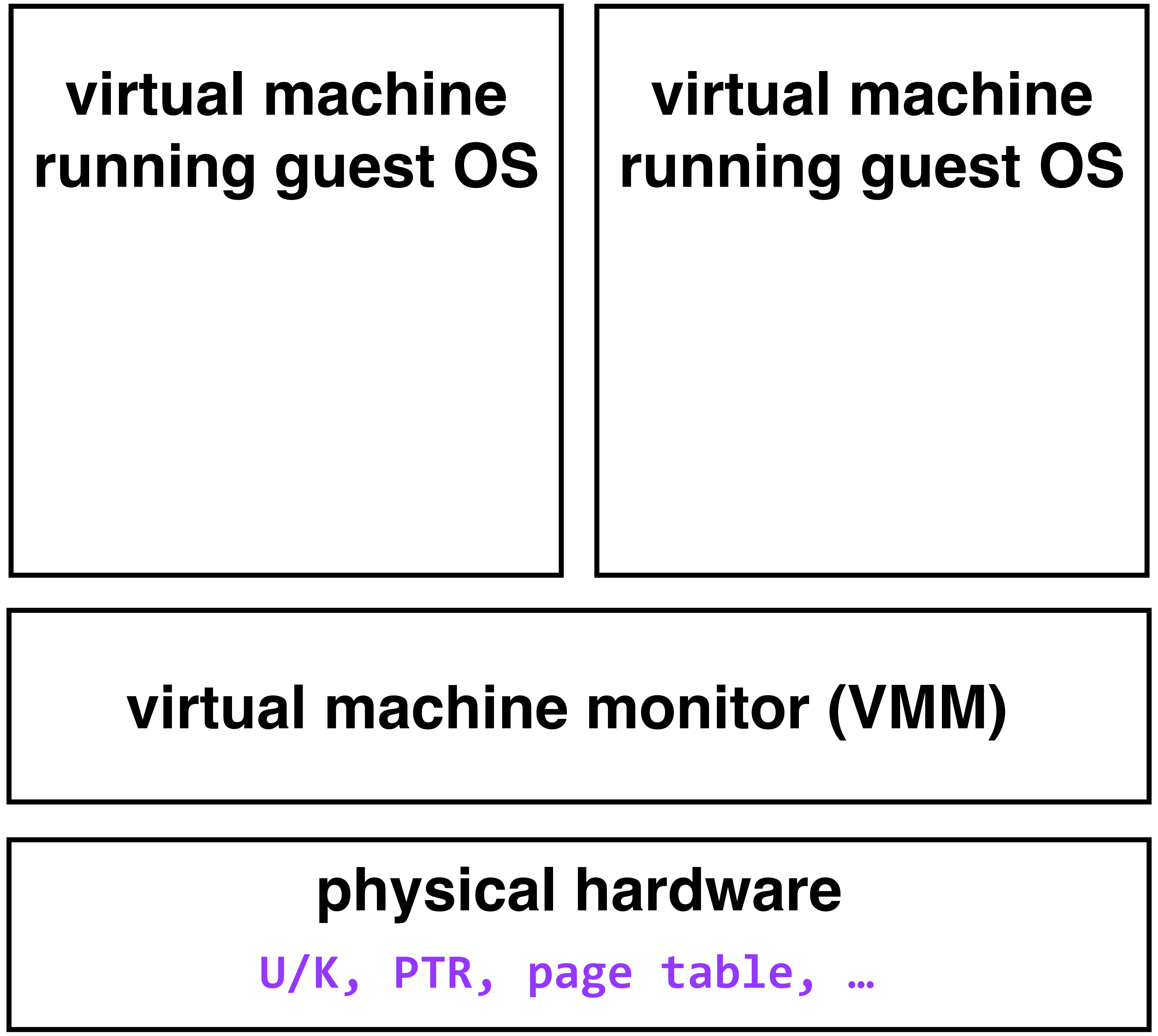
3. programs should be able to **share a CPU** without one program halting the progress of the others → **threads** (virtualize processors)

today's goal: run multiple operating systems at once

virtual machine monitor virtualizes the physical hardware for the guest OSes

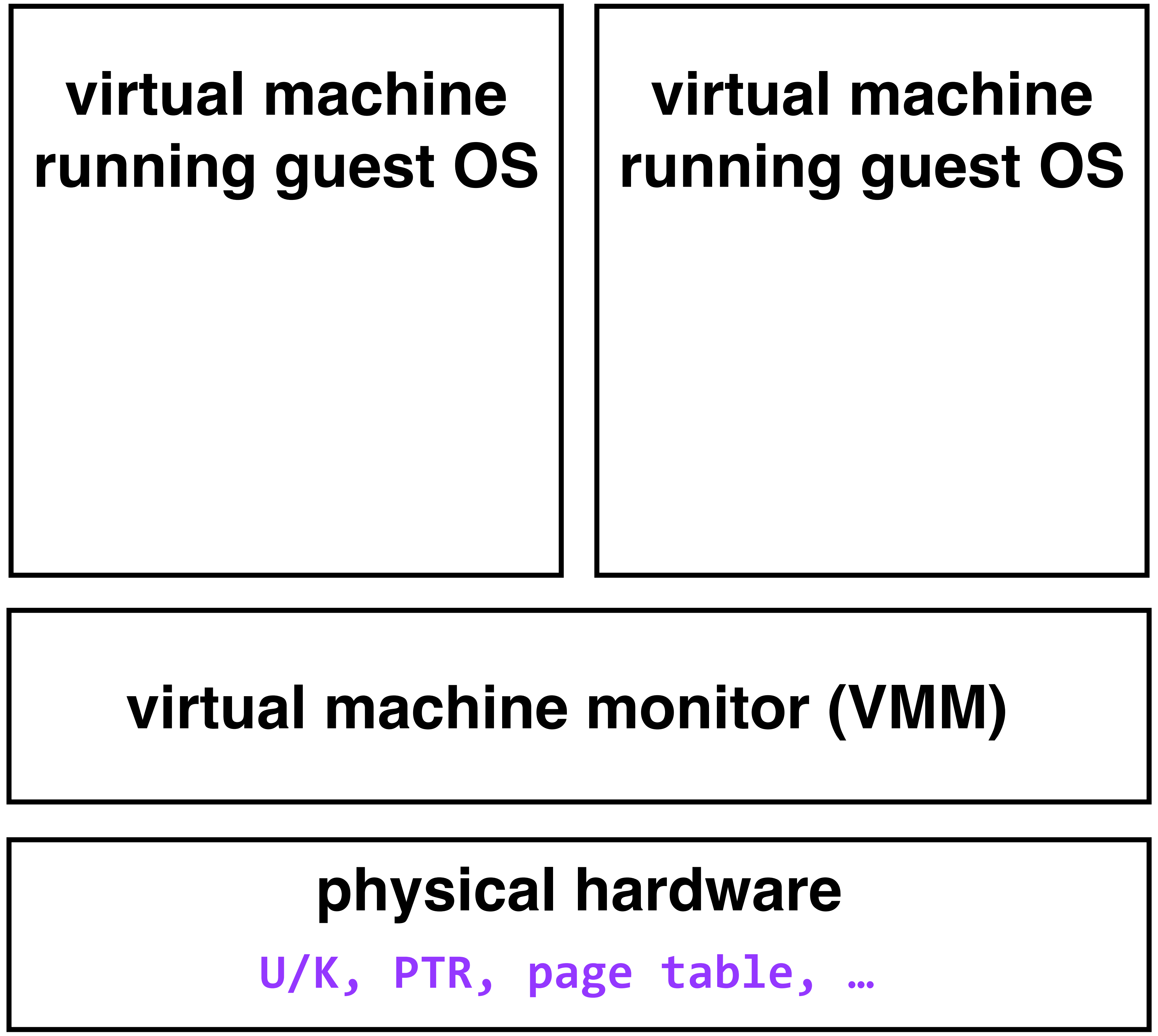


virtual machine monitor virtualizes the physical hardware for the guest OSes



guest OSes run in user mode

virtual machine monitor virtualizes the physical hardware for the guest OSes

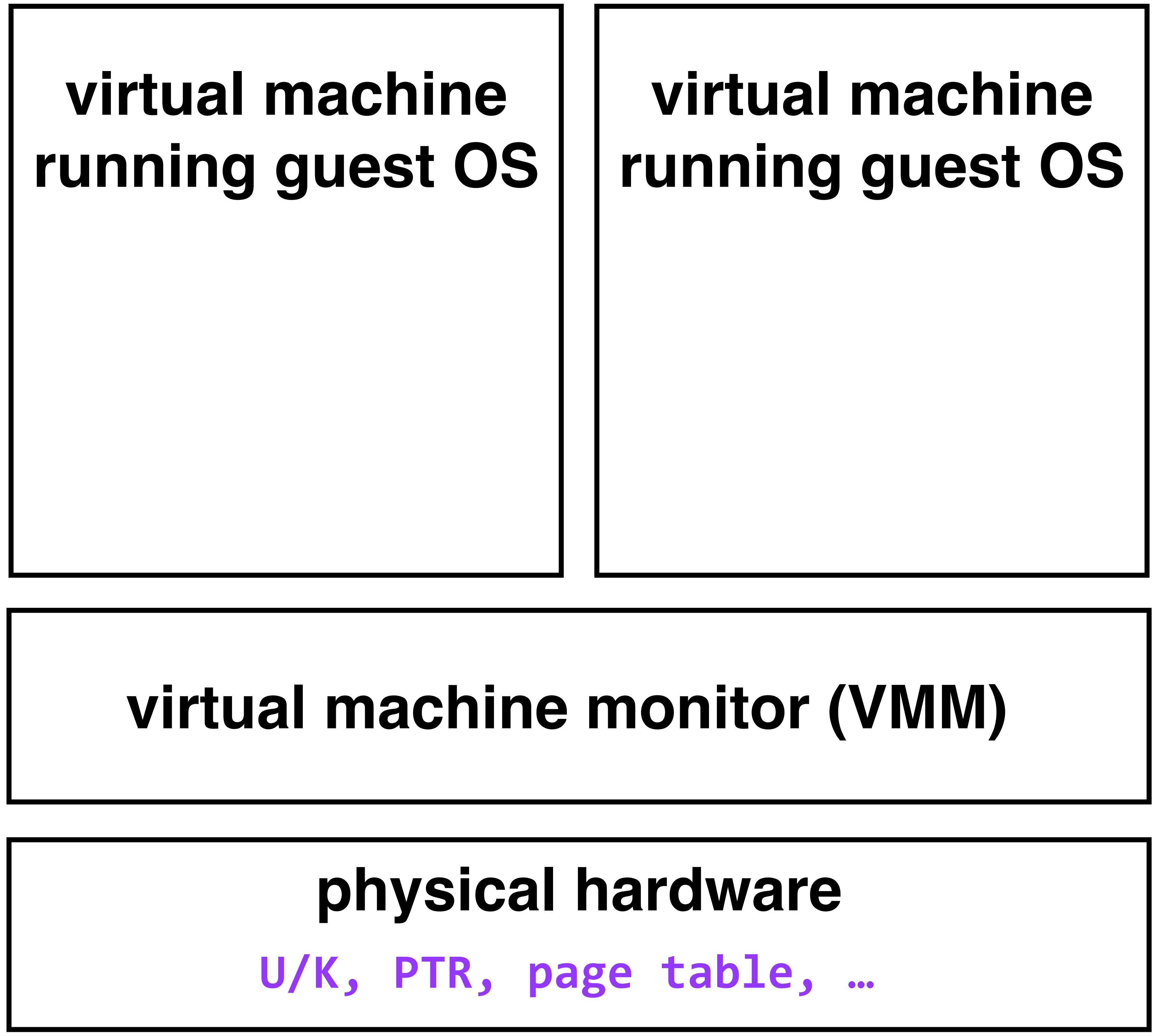


guest OSes run in user mode

privileged instructions in guest OS will cause an exception, which the VMM will intercept ("**trap**") and **emulate**

if the VMM *can't* emulate an instruction, it will send the exception back to the guest OS for handling

virtual machine monitor virtualizes the physical hardware for the guest OSes



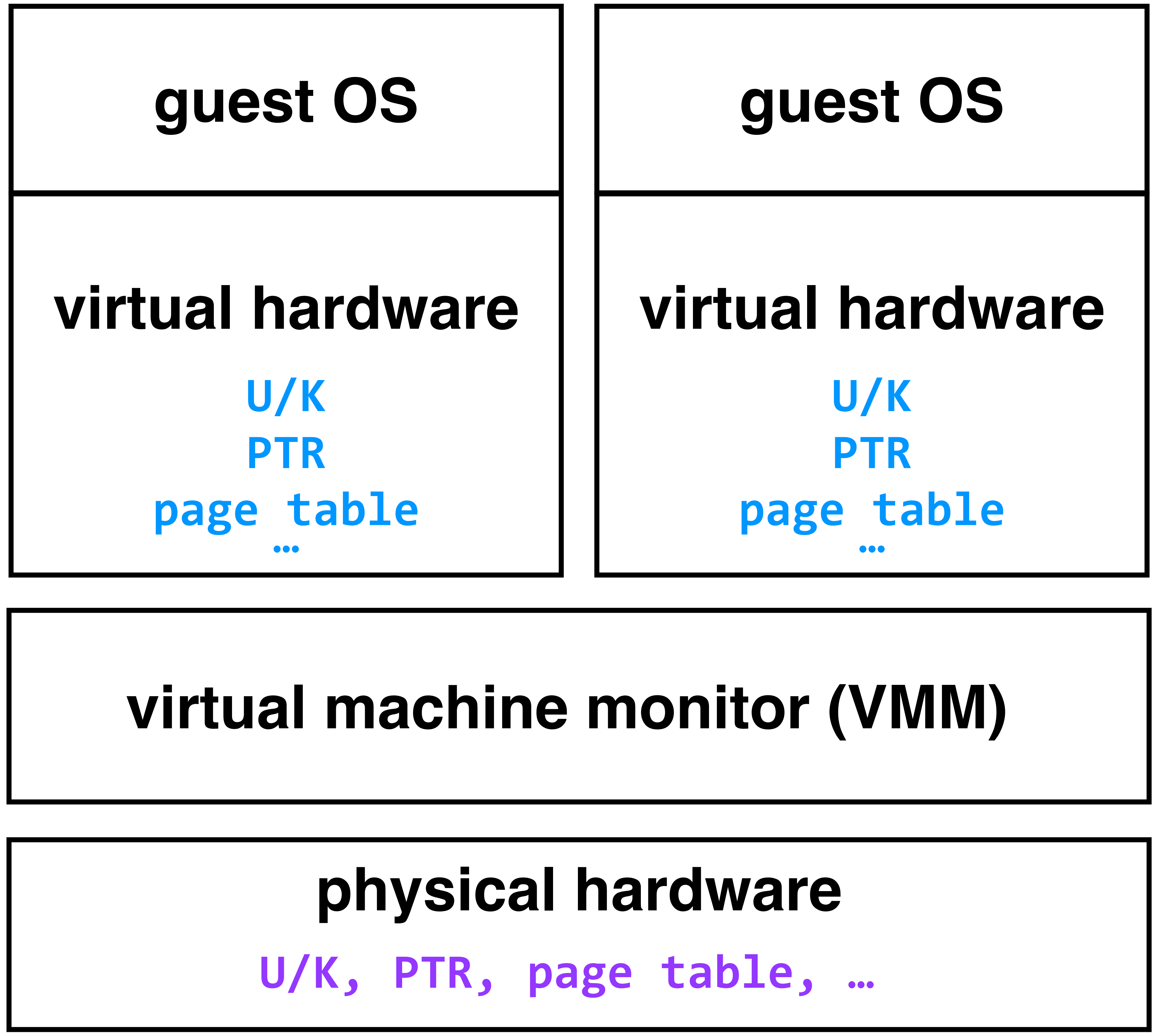
guest OSes run in user mode

privileged instructions in guest OS will cause an exception, which the VMM will intercept ("**trap**") and **emulate**

if the VMM *can't* emulate an instruction, it will send the exception back to the guest OS for handling

first question: what does it mean to emulate?

virtual machine monitor virtualizes the physical hardware for the guest OSes



guest OSes run in user mode

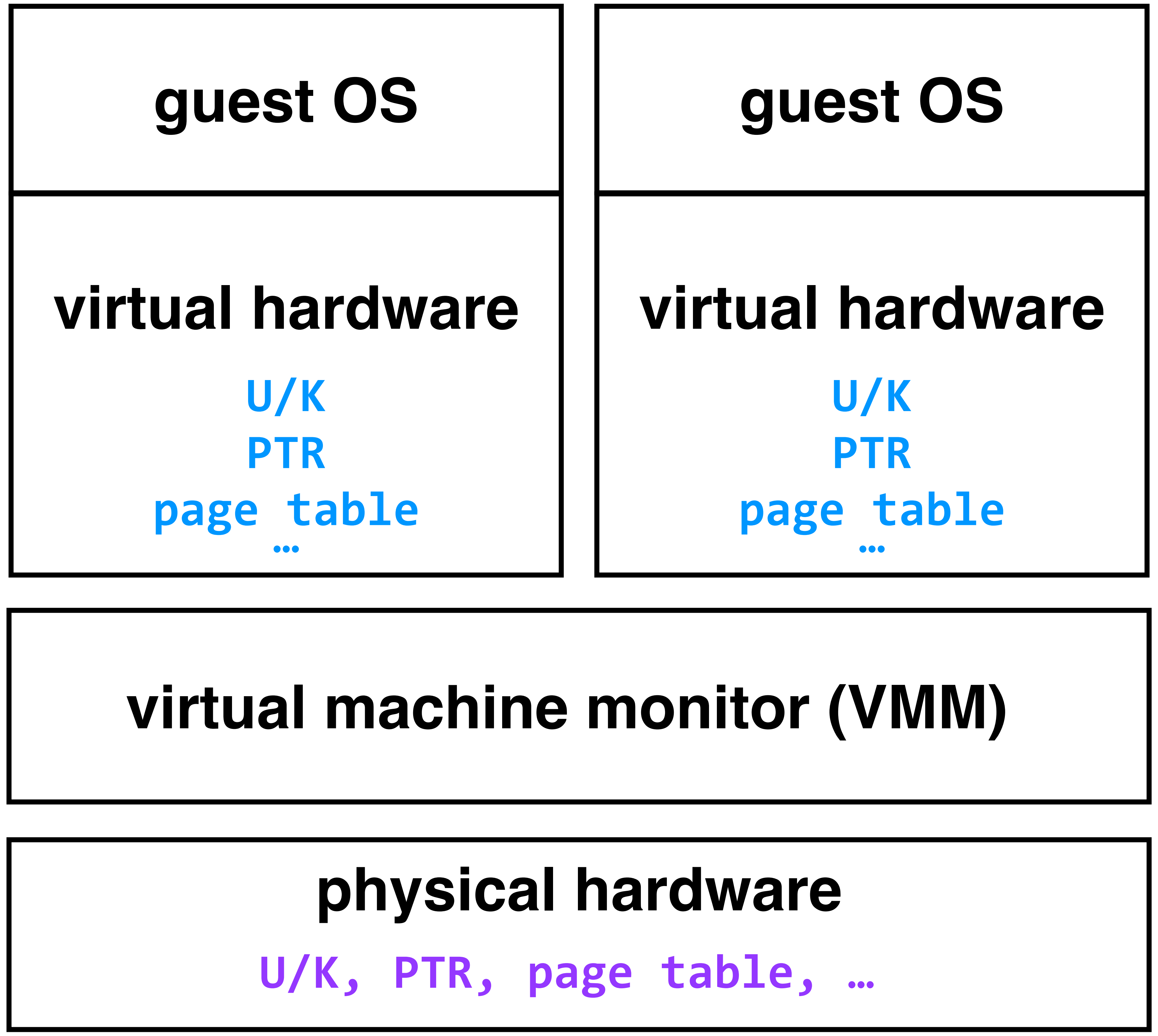
privileged instructions in guest OS will cause an exception, which the VMM will intercept ("**trap**") and **emulate**

if the VMM *can't* emulate an instruction, it will send the exception back to the guest OS for handling

first question: what does it mean to emulate?

virtual machine monitor virtualizes the physical hardware for the guest OSes

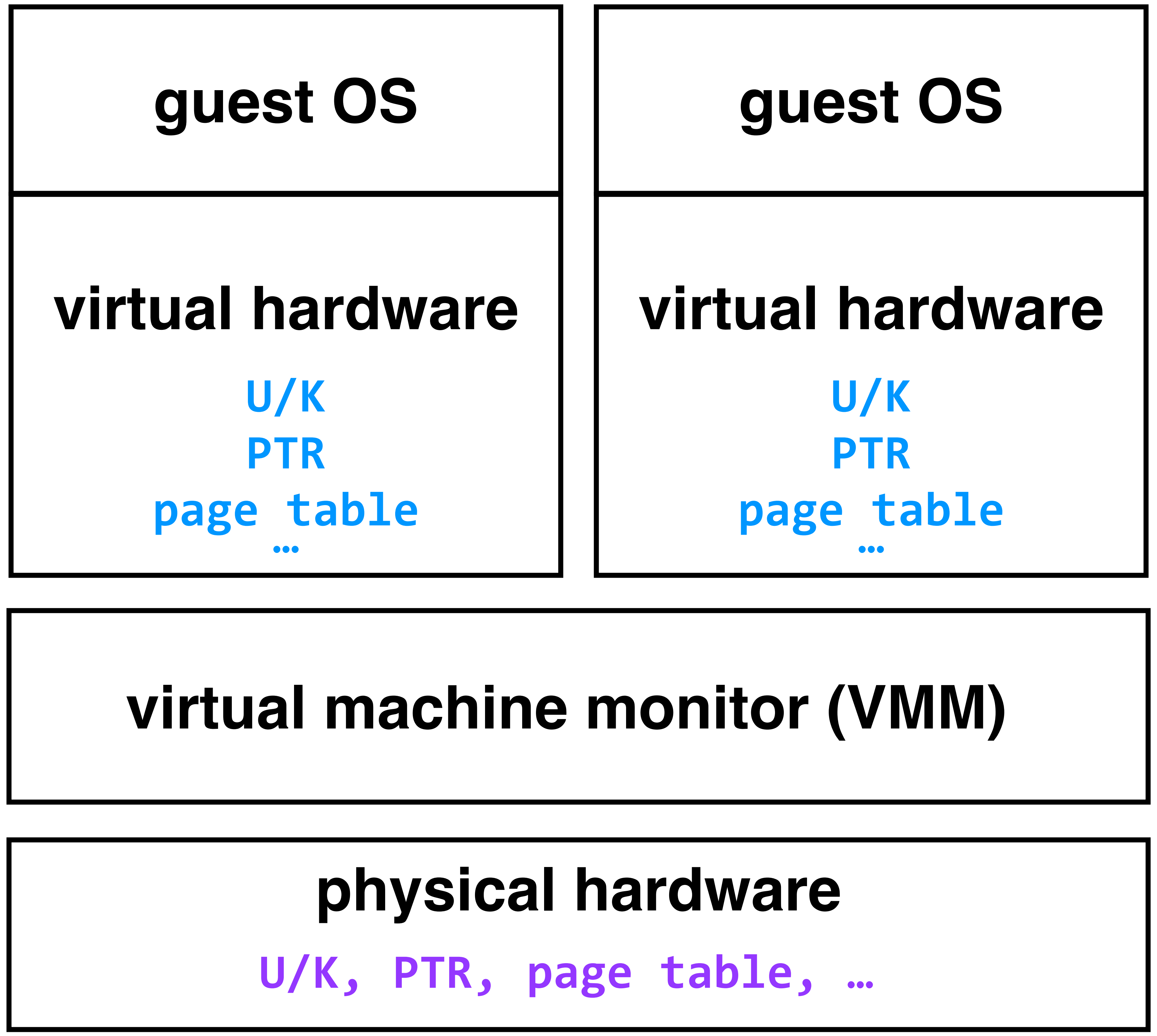
first example: virtualizing memory (again!)



first question: what does it mean to emulate?

virtual machine monitor virtualizes the physical hardware for the guest OSes

first example: virtualizing memory (again!)

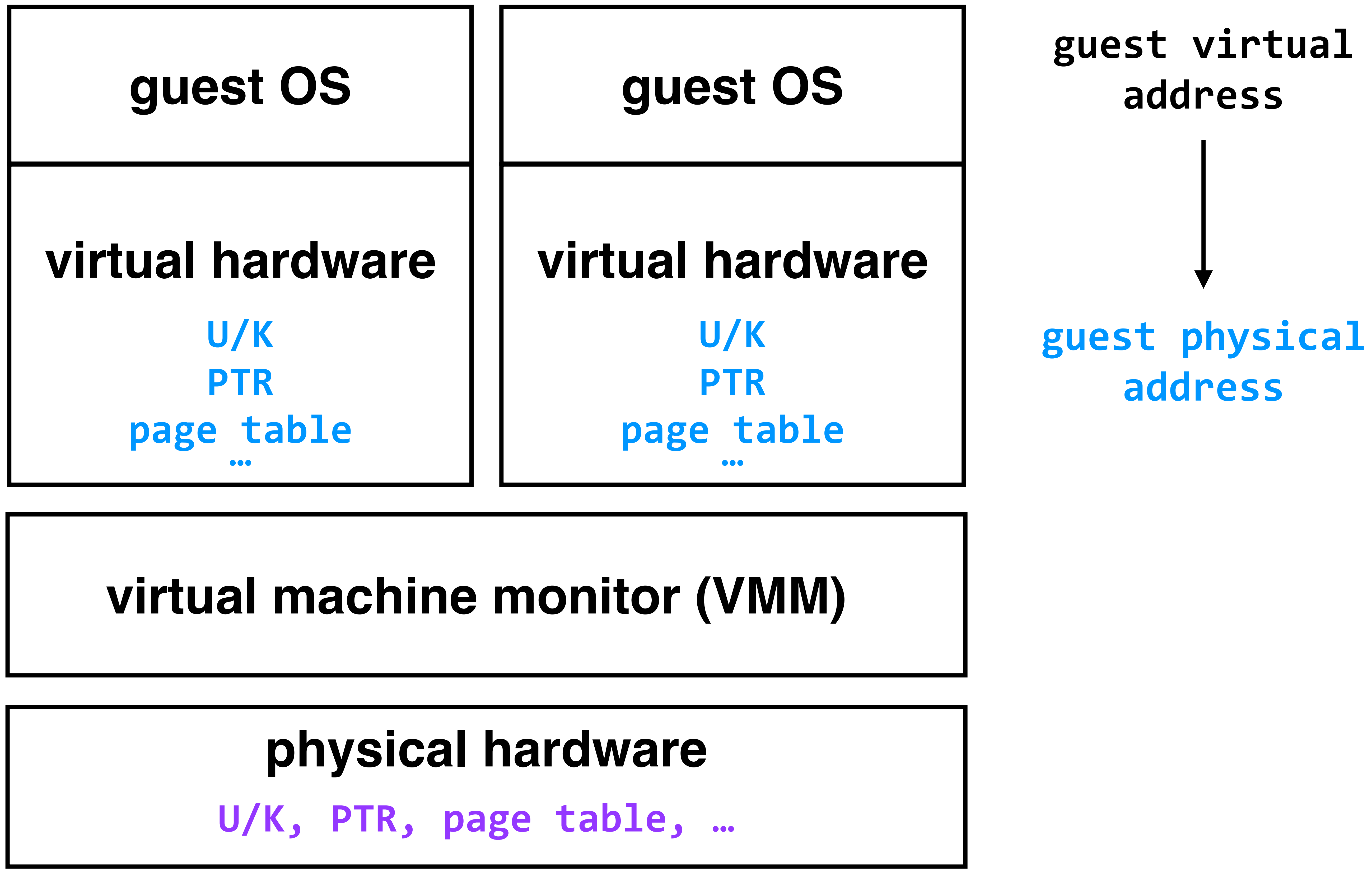


`guest virtual address`

first question: what does it mean to emulate?

virtual machine monitor virtualizes the physical hardware for the guest OSes

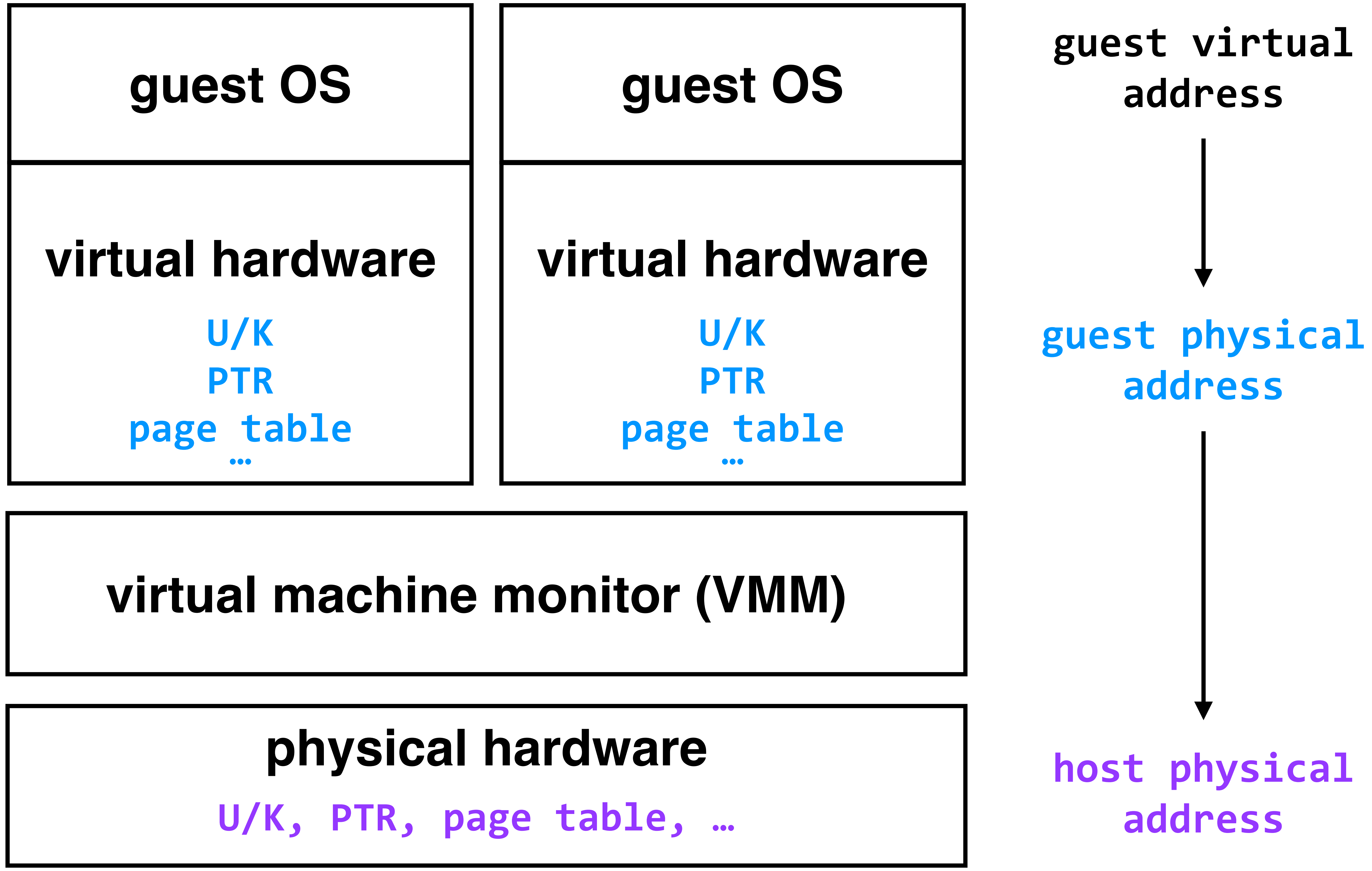
first example: virtualizing memory (again!)



first question: what does it mean to emulate?

virtual machine monitor virtualizes the physical hardware for the guest OSes

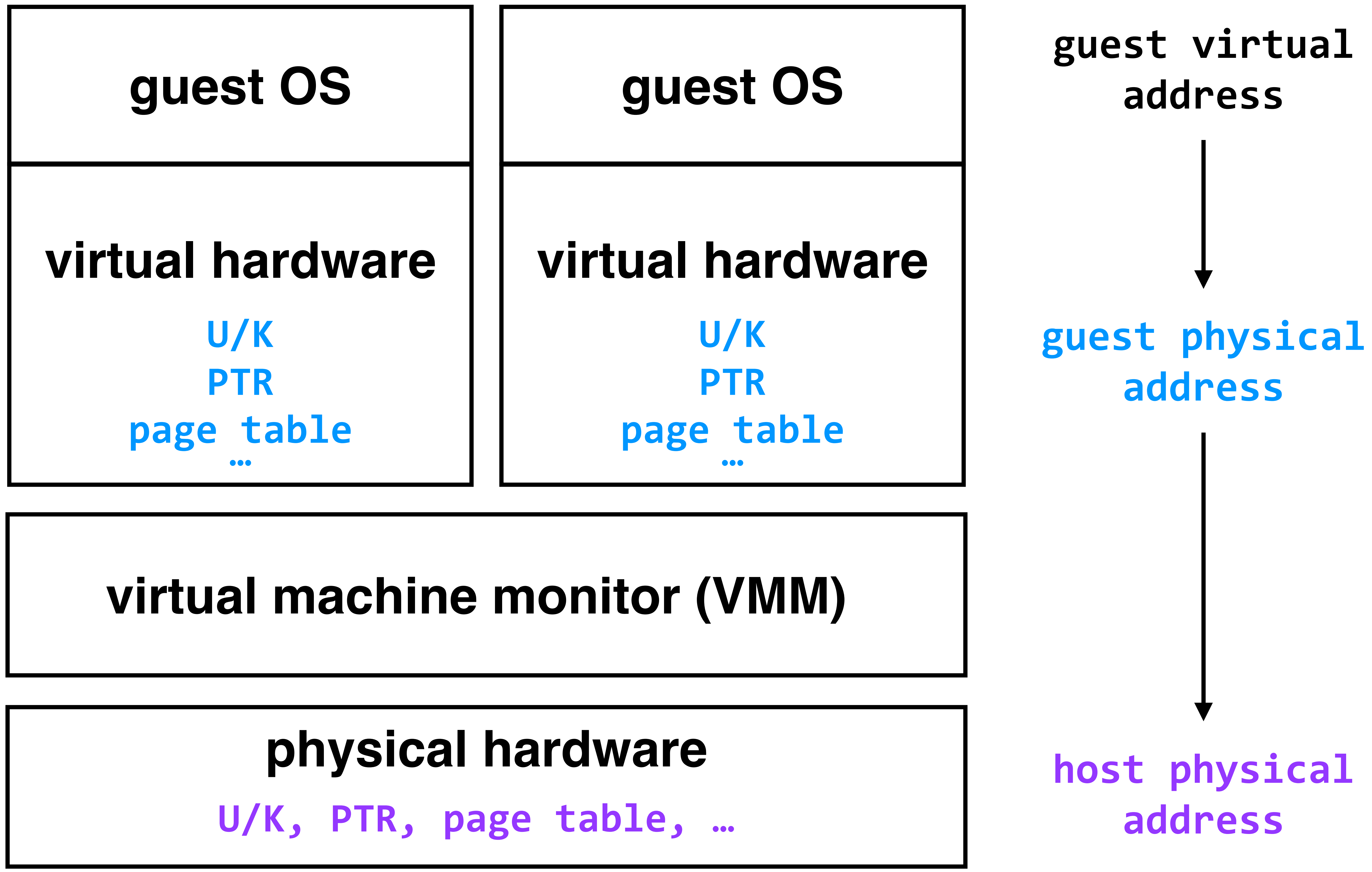
first example: virtualizing memory (again!)



first question: what does it mean to emulate?

virtual machine monitor virtualizes the physical hardware for the guest OSes

first example: virtualizing memory (again!)

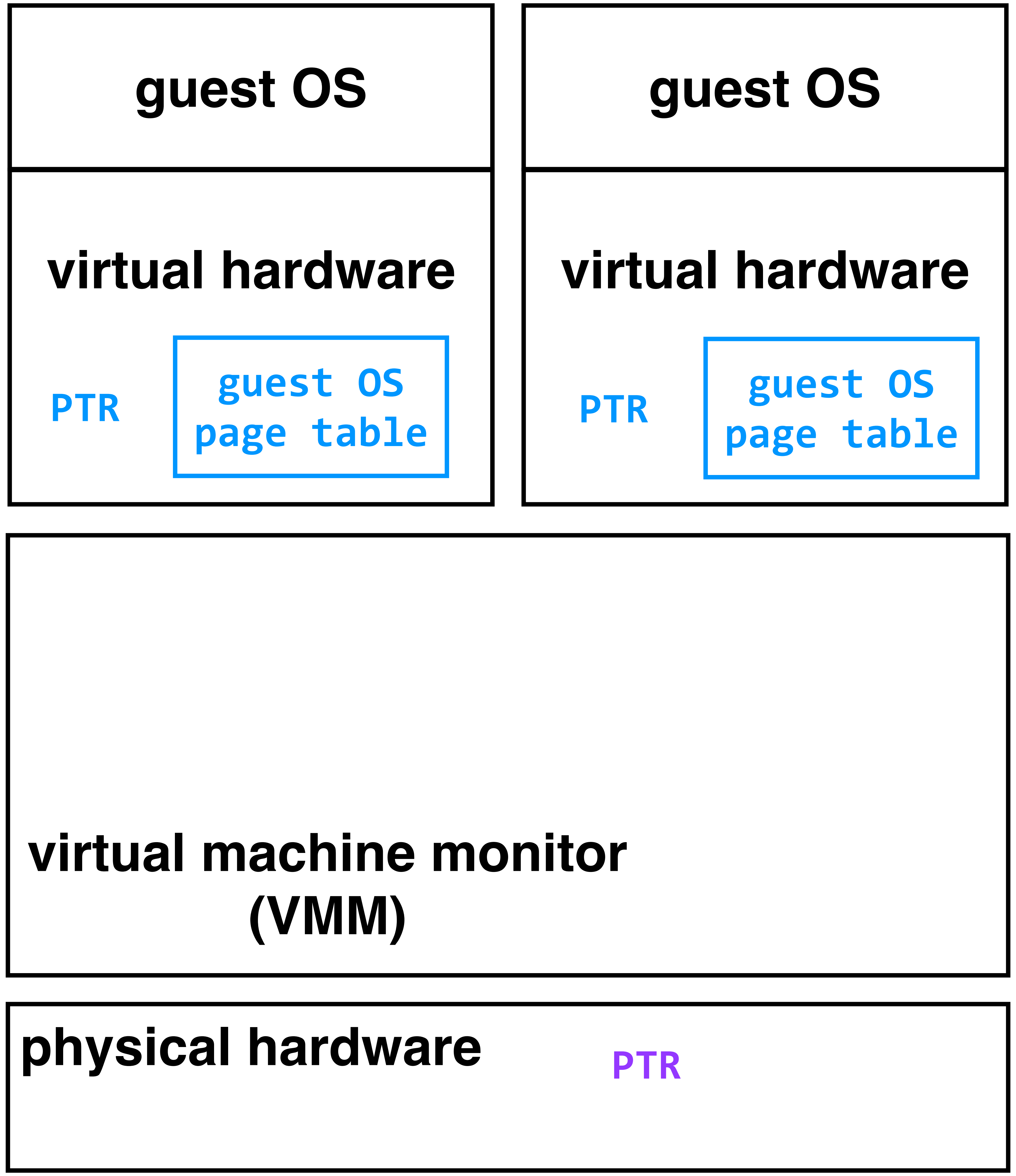


first question: what does it mean to emulate?

in this example, it means that the VMM needs to step in and translate guest physical addresses to host physical addresses

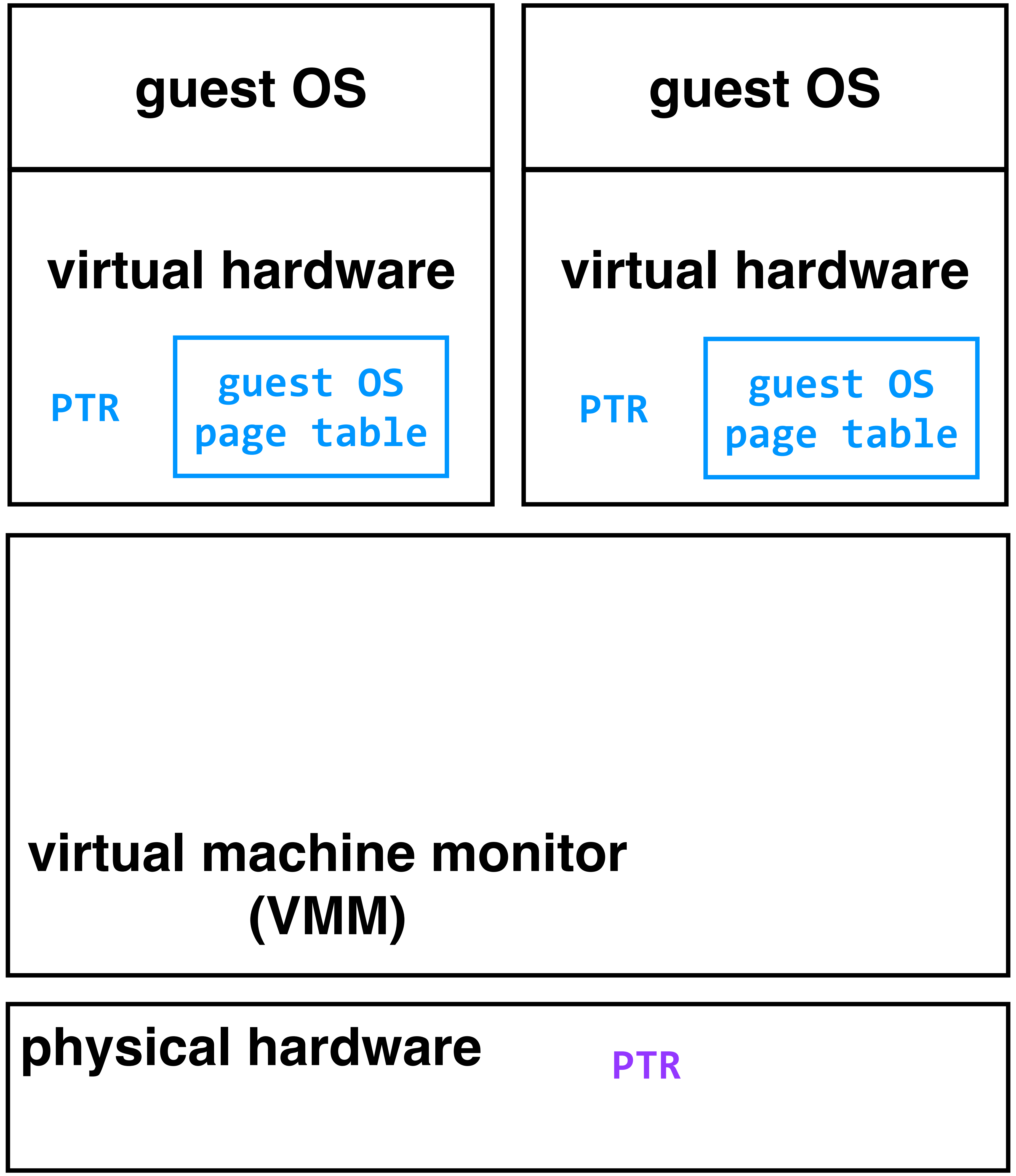
virtual machine monitor virtualizes the physical hardware for the guest OSes

first example: virtualizing memory (again!)



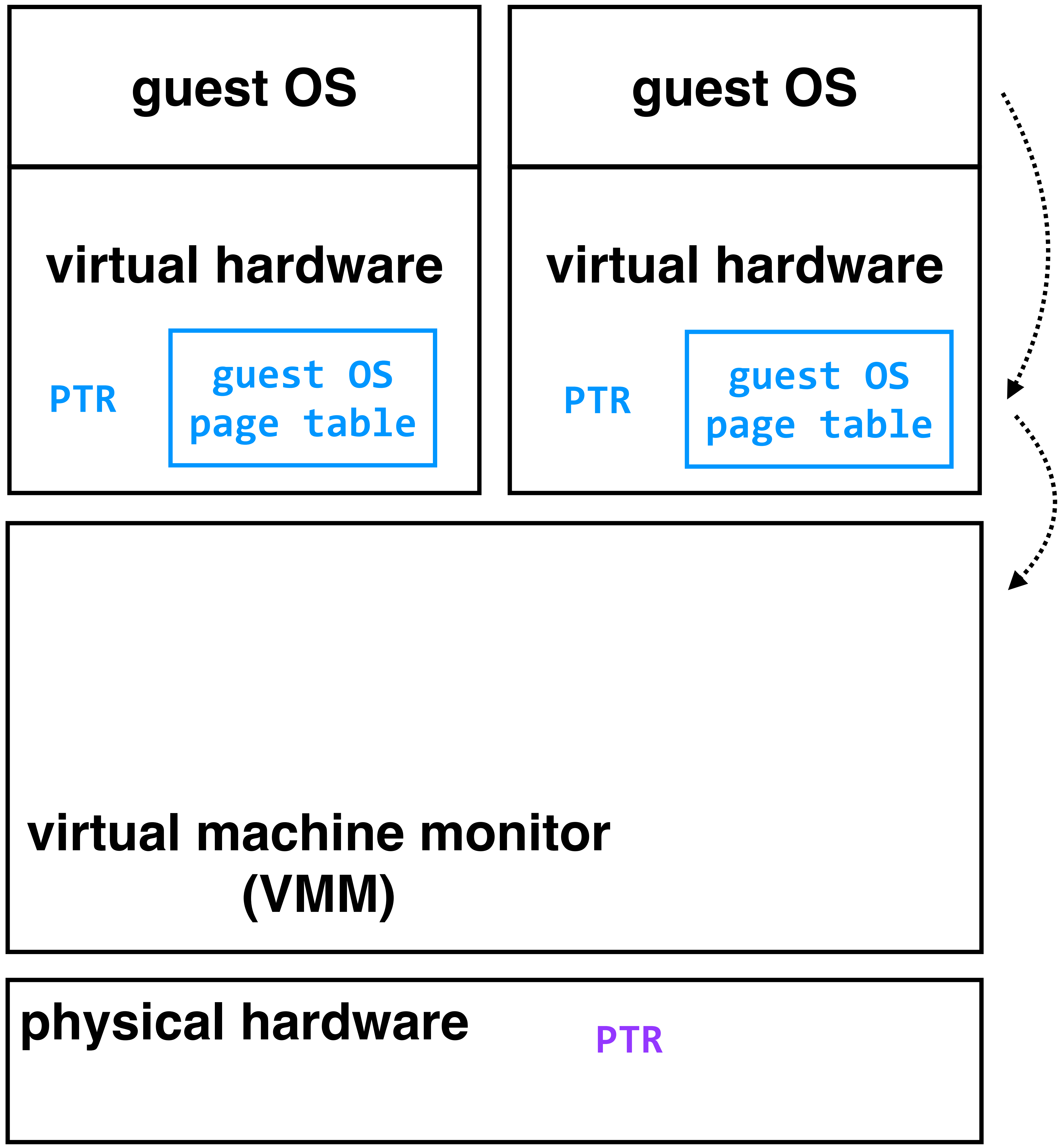
virtual machine monitor virtualizes the physical hardware for the guest OSes

first example: virtualizing memory (again!)



virtual machine monitor virtualizes the physical hardware for the guest OSes

first example: virtualizing memory (again!)

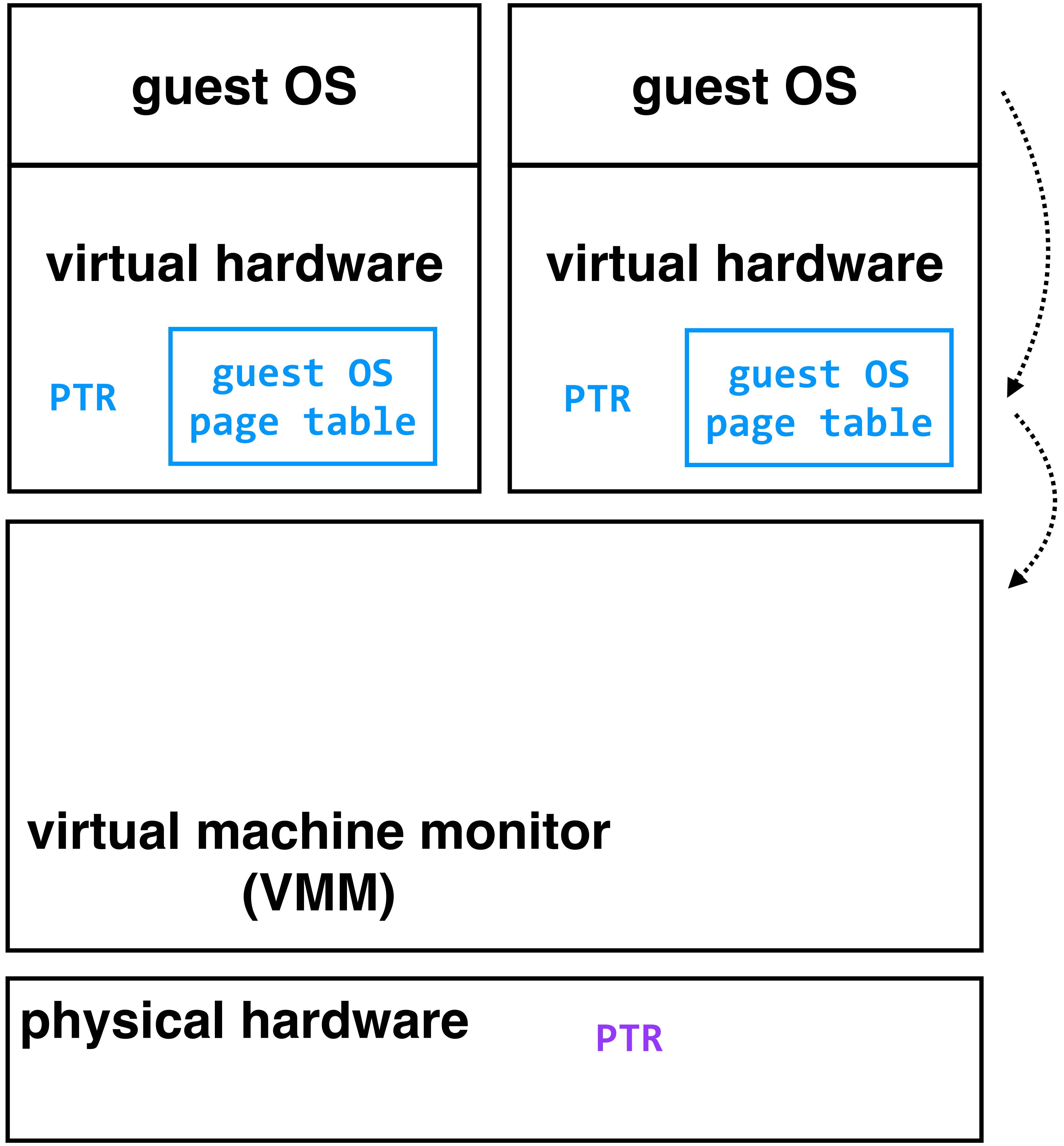


1. guest OS loads its PTR, which triggers an exception; the VMM intercepts

guest virtual → guest physical

virtual machine monitor virtualizes the physical hardware for the guest OSes

first example: virtualizing memory (again!)

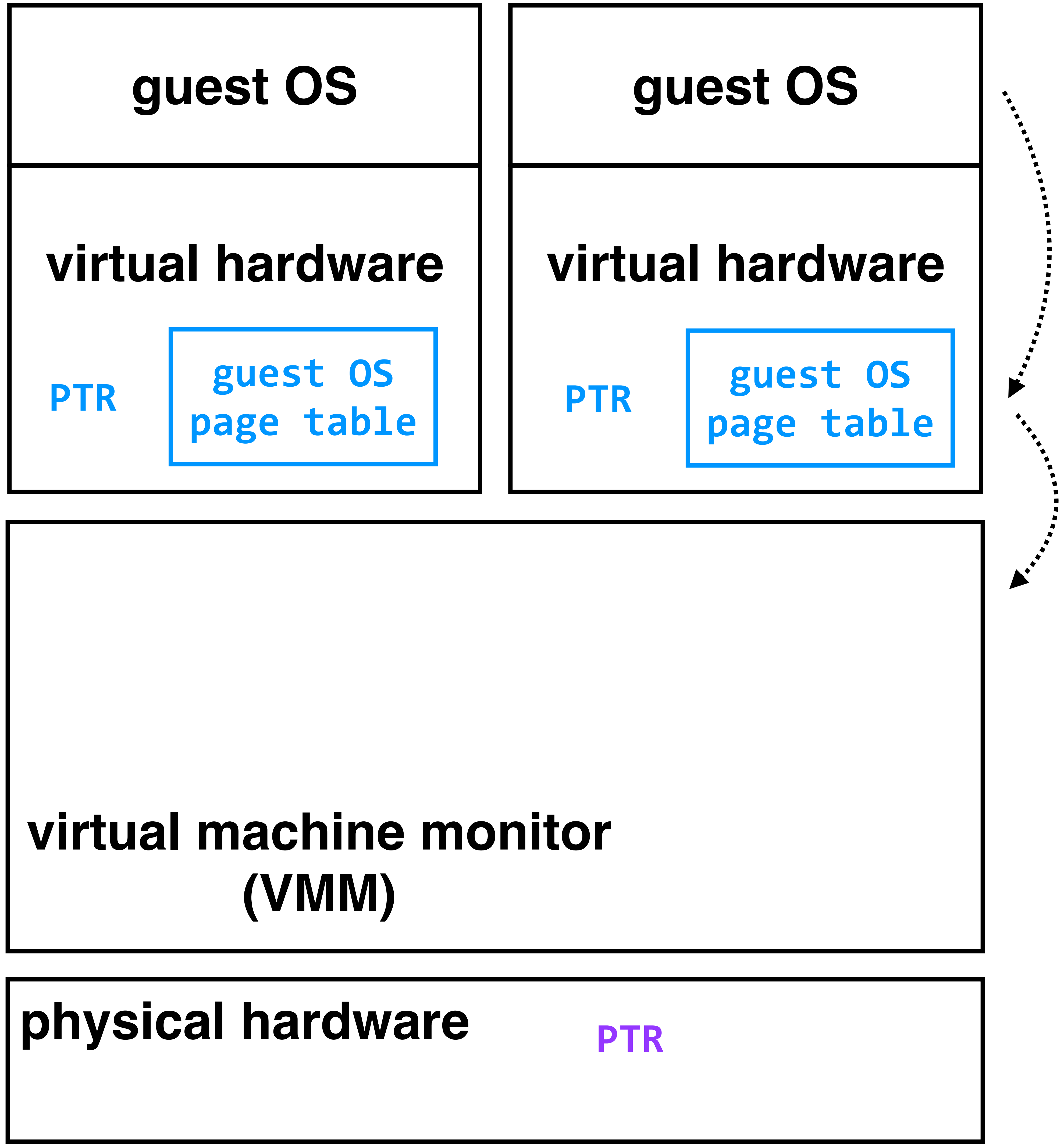


1. guest OS loads its PTR, which triggers an exception; the VMM intercepts

guest virtual → guest physical

virtual machine monitor virtualizes the physical hardware for the guest OSes

first example: virtualizing memory (again!)

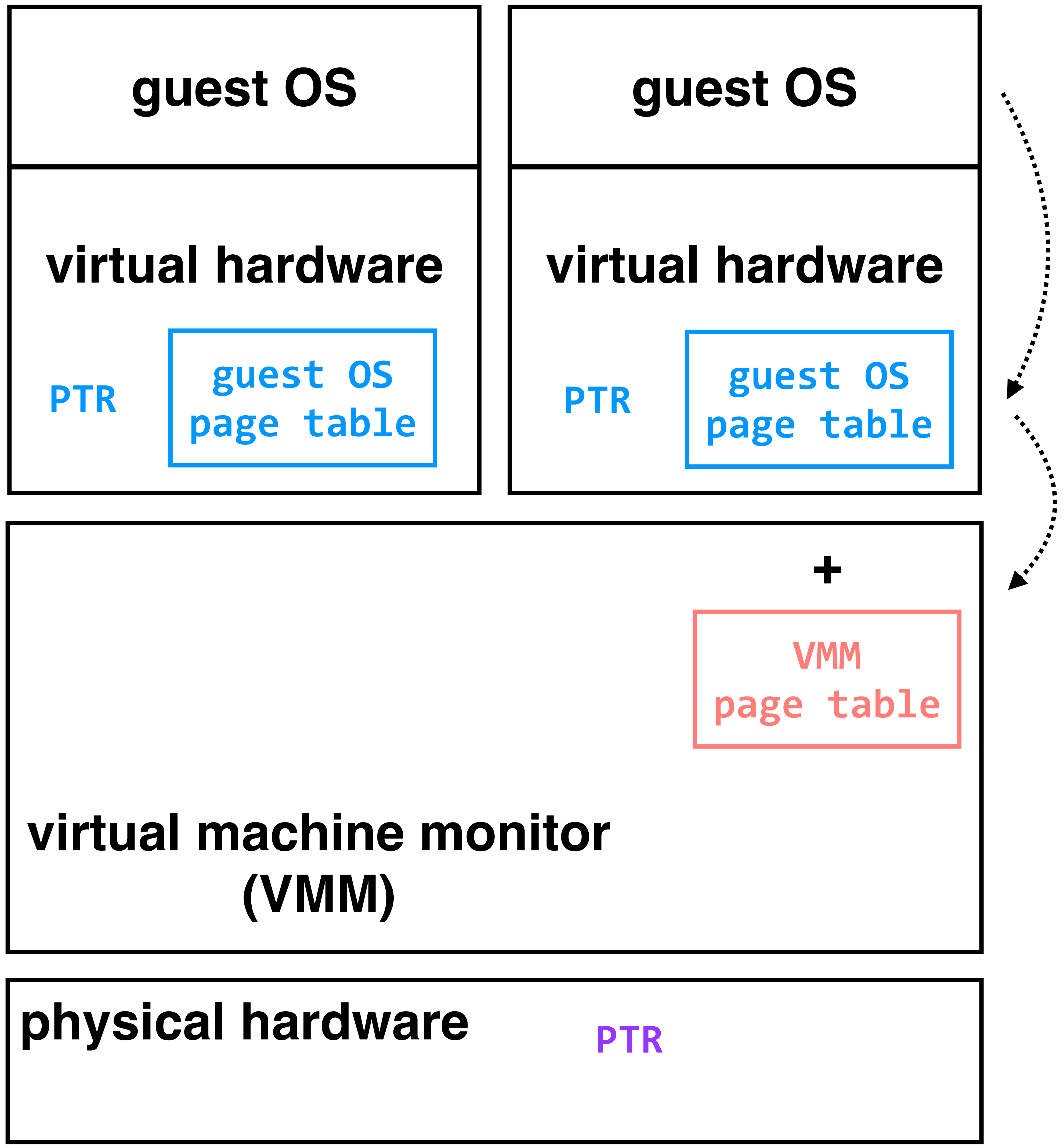


1. guest OS loads its PTR, which triggers an exception; the VMM intercepts

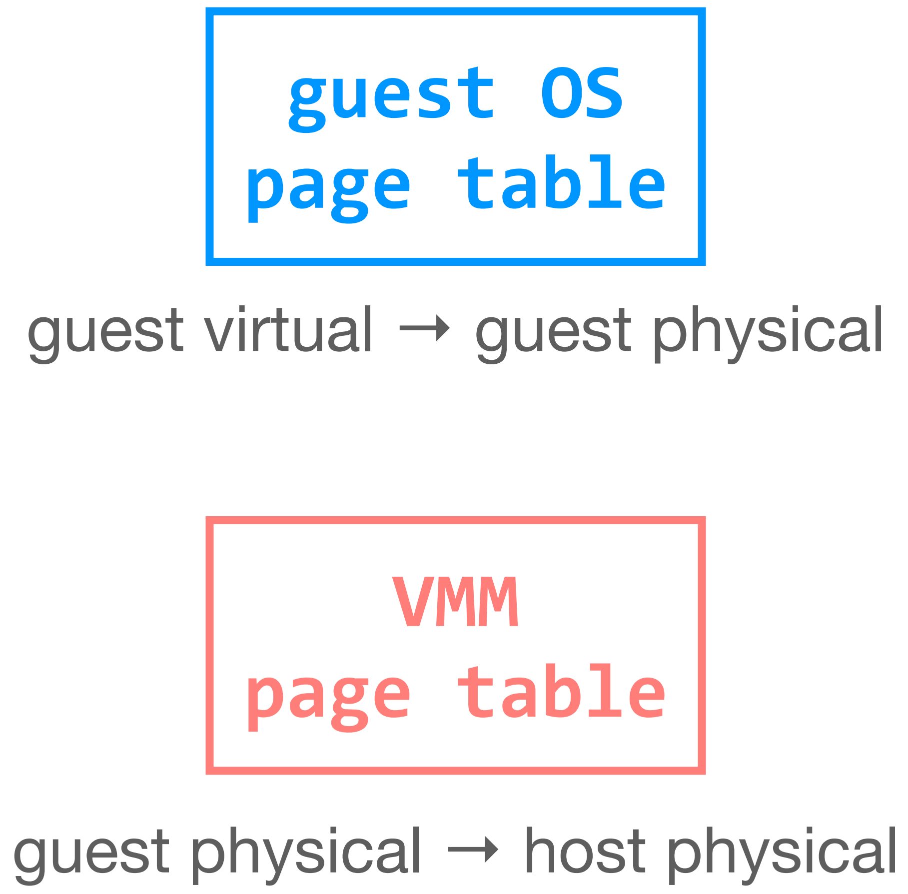
2. VMM combines the guest page table with its own page table to create a host page table

virtual machine monitor virtualizes the physical hardware for the guest OSes

first example: virtualizing memory (again!)

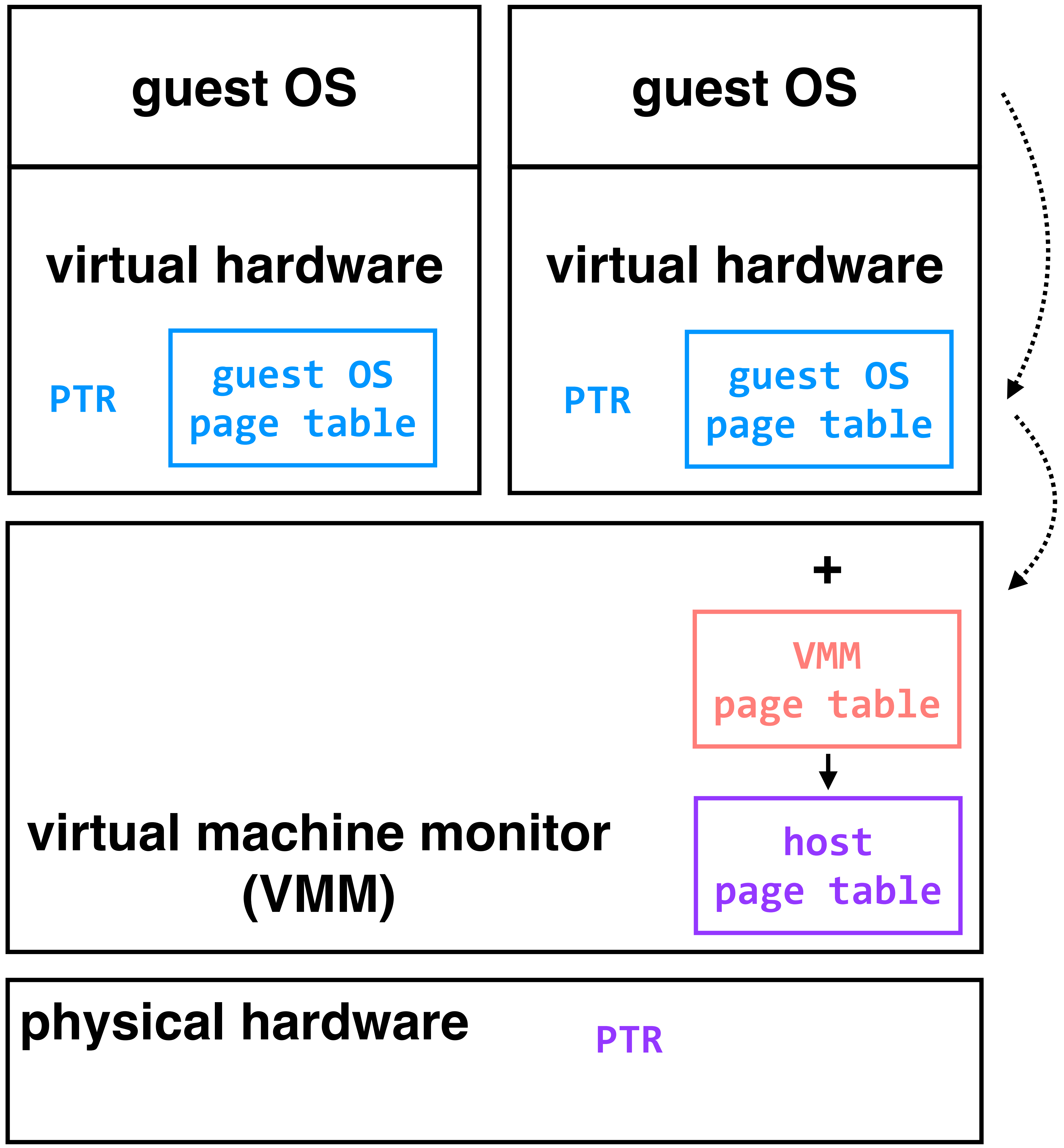


1. guest OS loads its PTR, which triggers an exception; the VMM intercepts
2. VMM combines the guest page table with its own page table to create a host page table

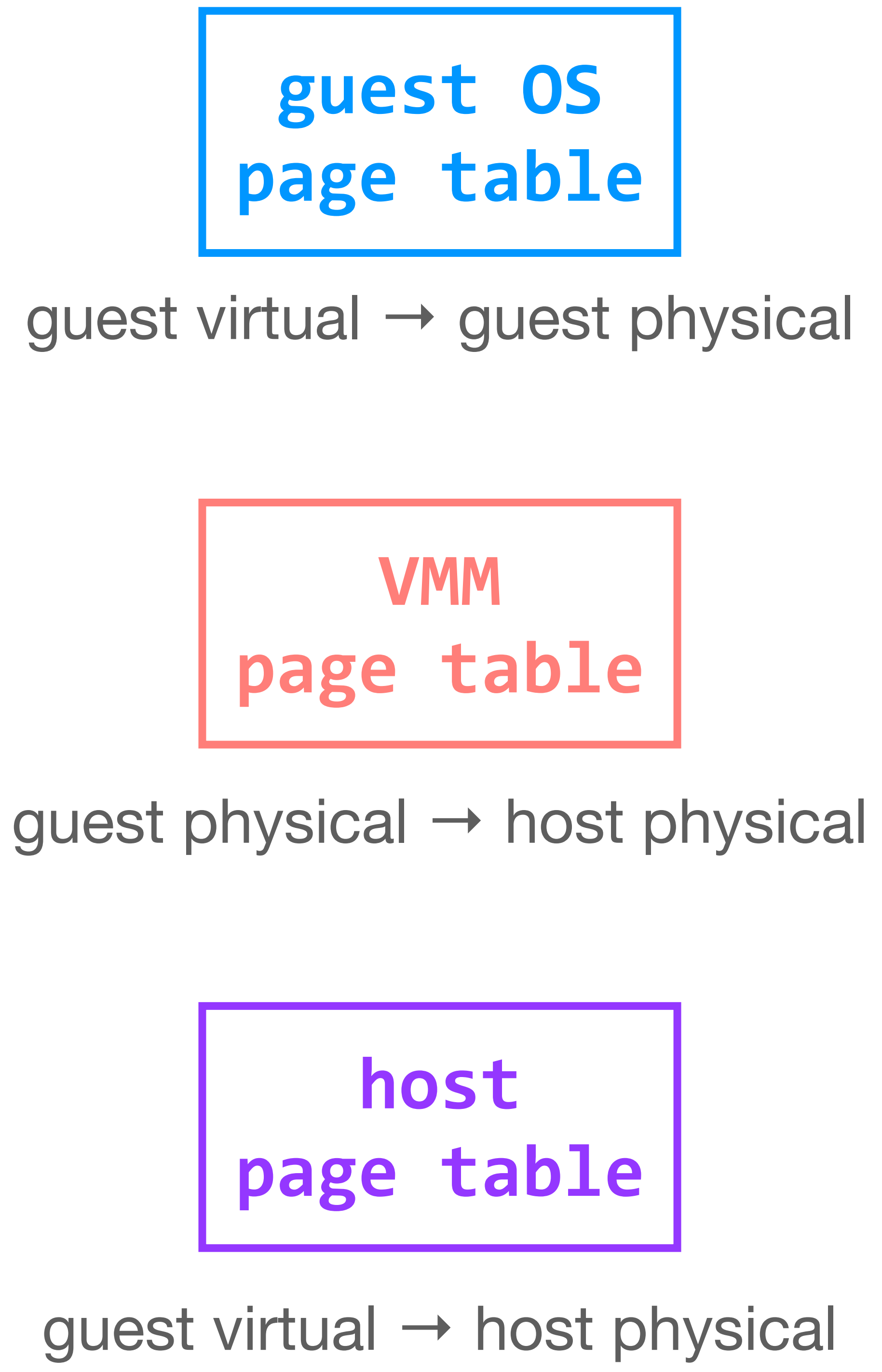


virtual machine monitor virtualizes the physical hardware for the guest OSes

first example: virtualizing memory (again!)

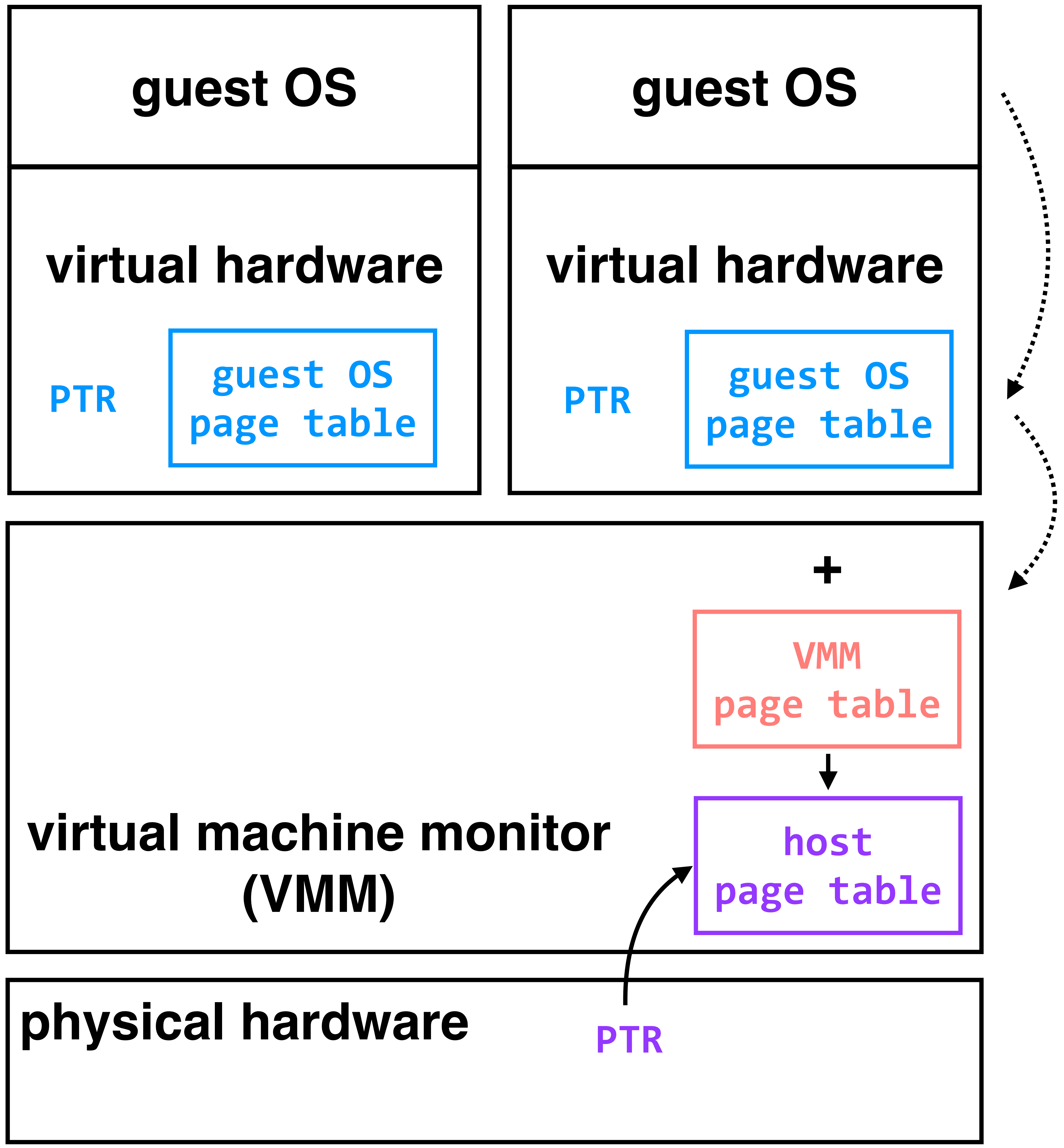


1. guest OS loads its PTR, which triggers an exception; the VMM intercepts
2. VMM combines the guest page table with its own page table to create a host page table
3. physical hardware uses the host page table

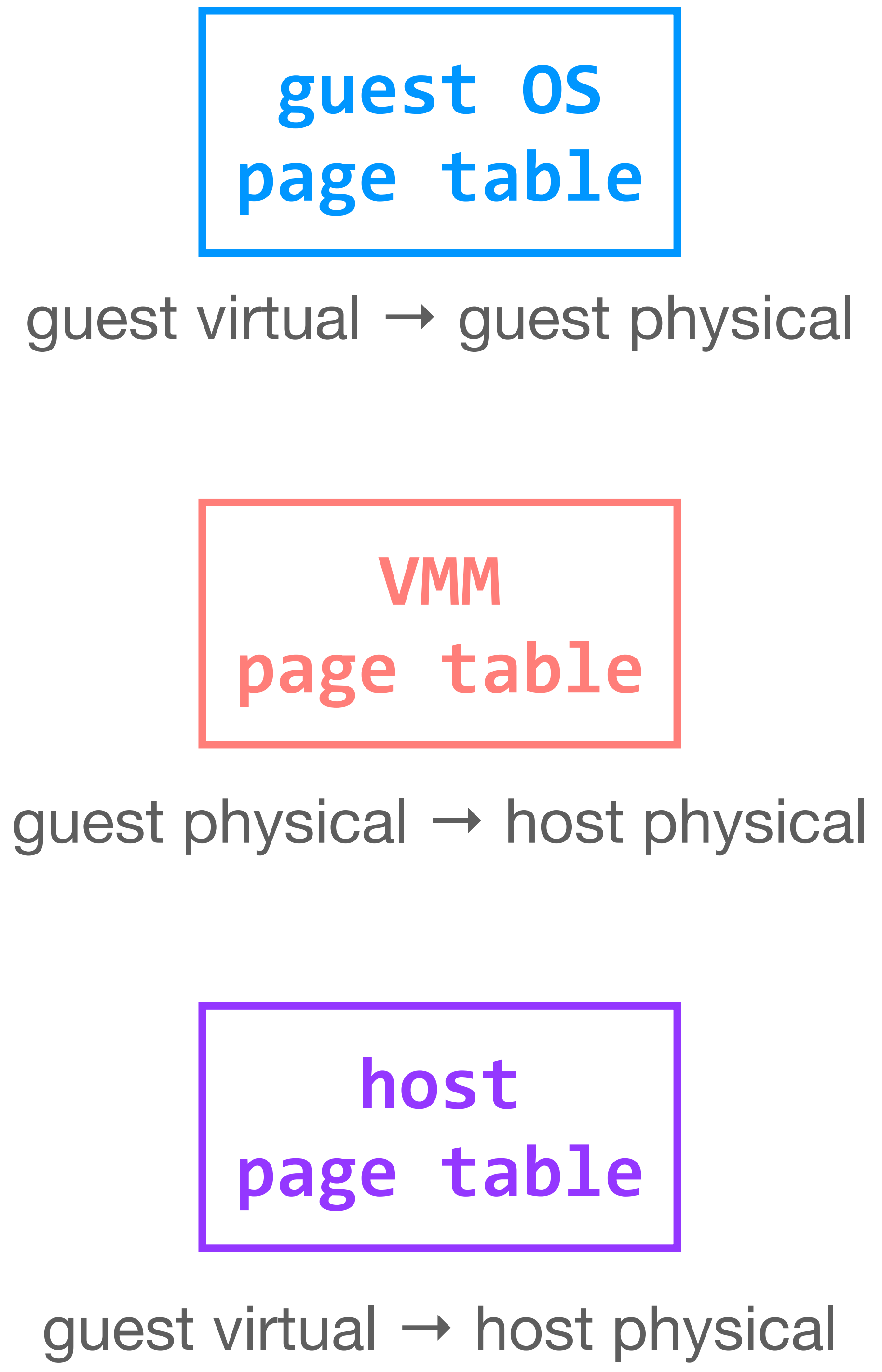


virtual machine monitor virtualizes the physical hardware for the guest OSes

first example: virtualizing memory (again!)

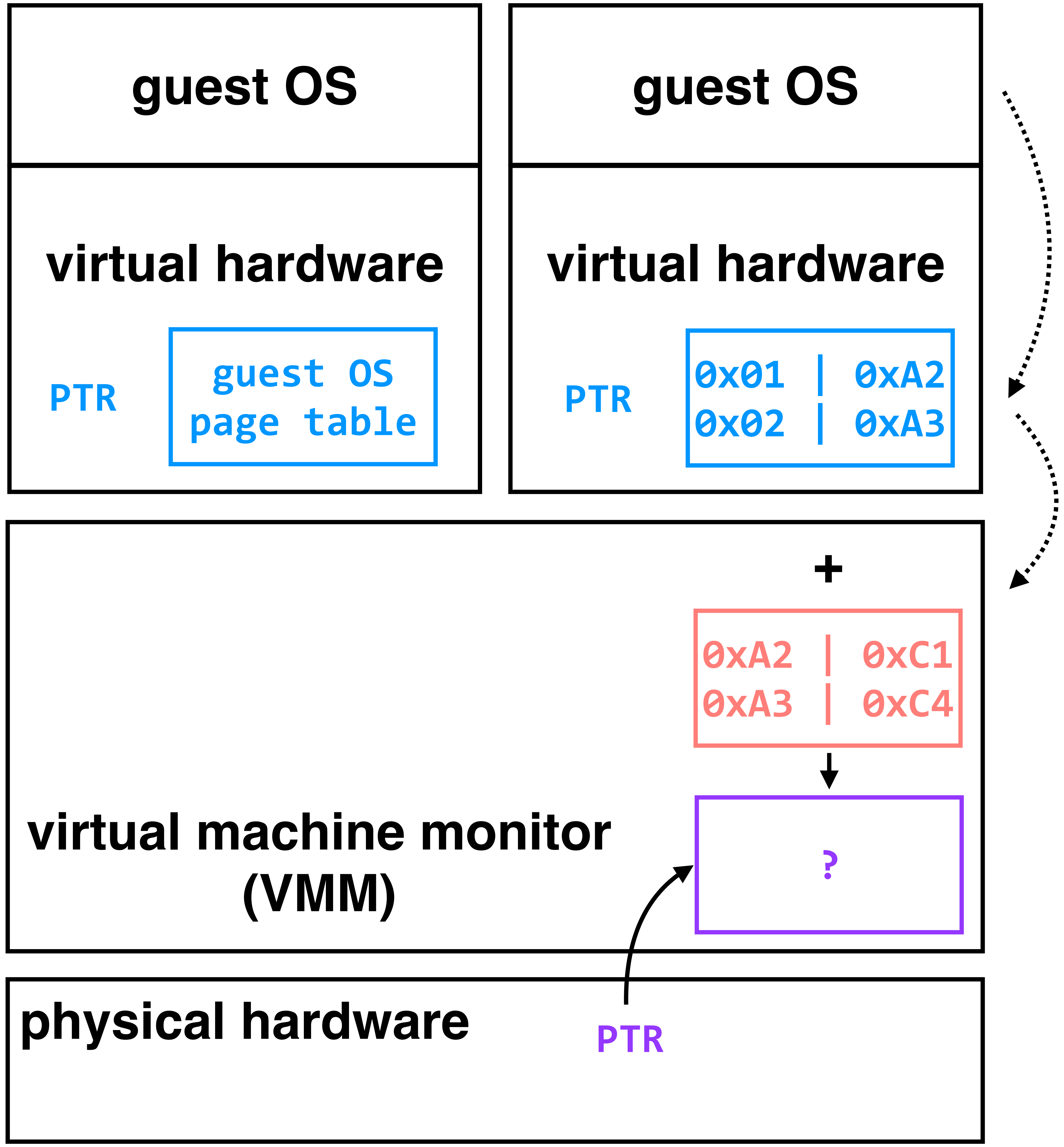


1. guest OS loads its PTR, which triggers an exception; the VMM intercepts
2. VMM combines the guest page table with its own page table to create a host page table
3. physical hardware uses the host page table

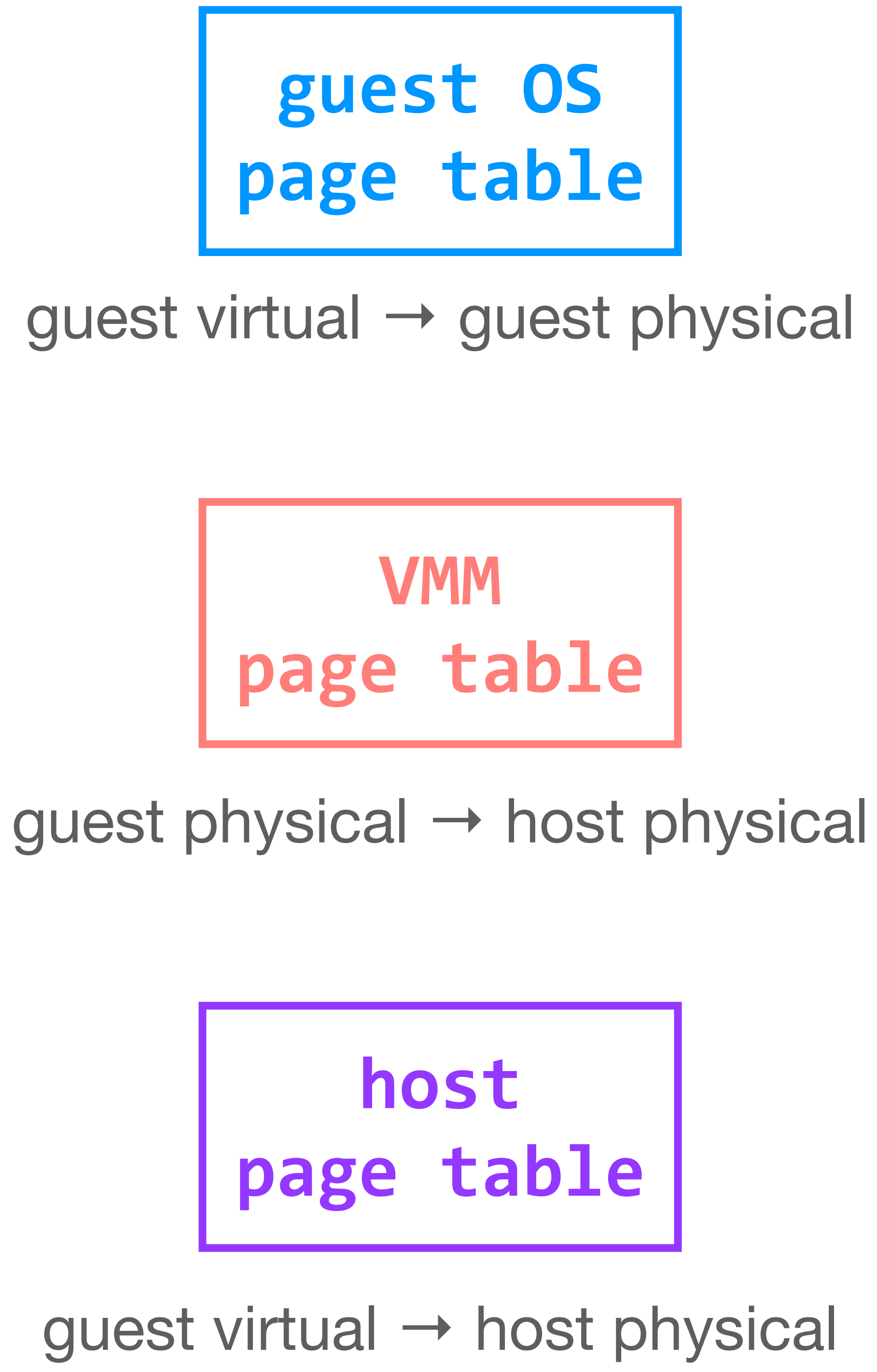


virtual machine monitor virtualizes the physical hardware for the guest OSes

first example: virtualizing memory (again!)

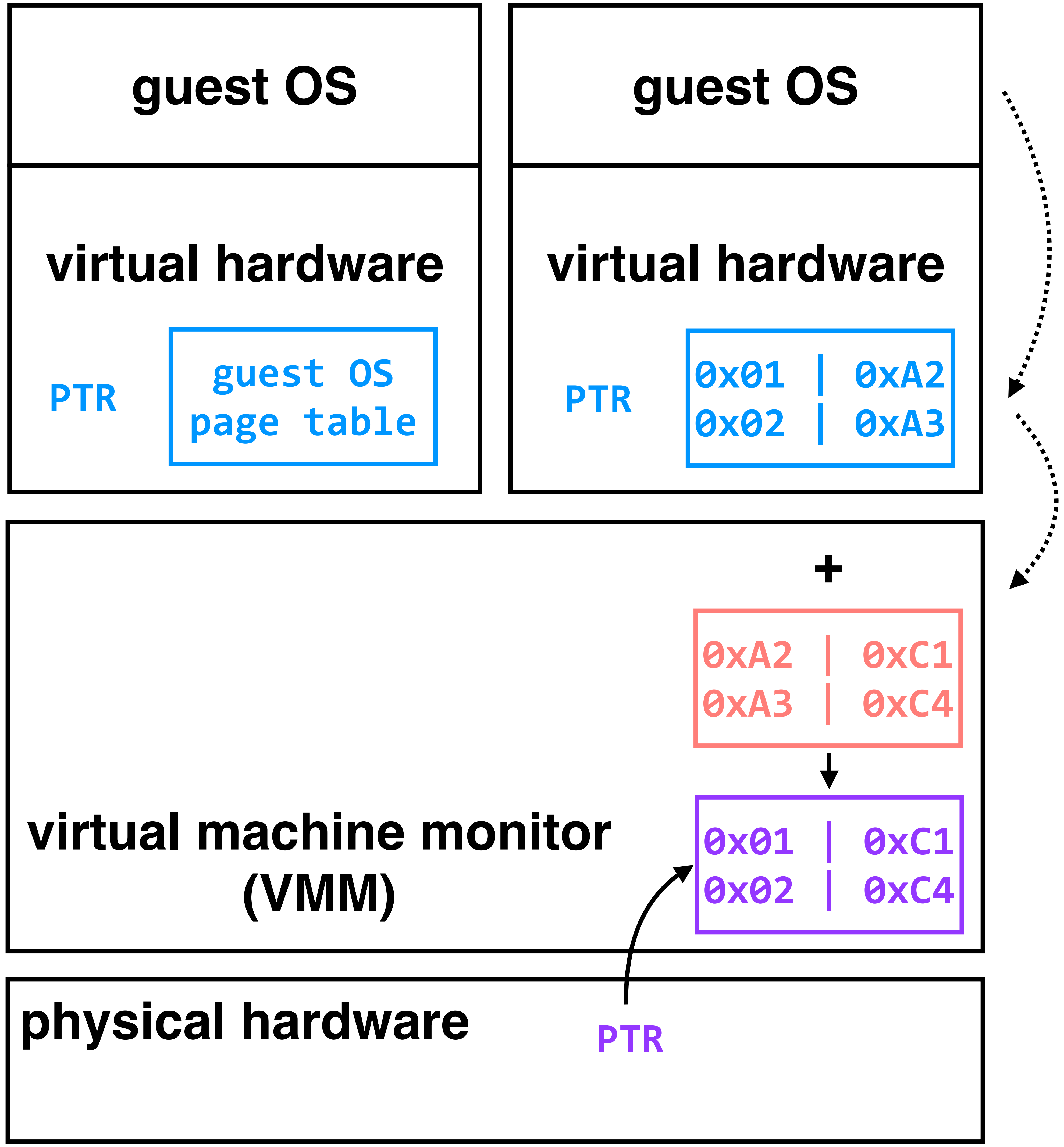


1. guest OS loads its PTR, which triggers an exception; the VMM intercepts
2. VMM combines the guest page table with its own page table to create a host page table
3. physical hardware uses the host page table

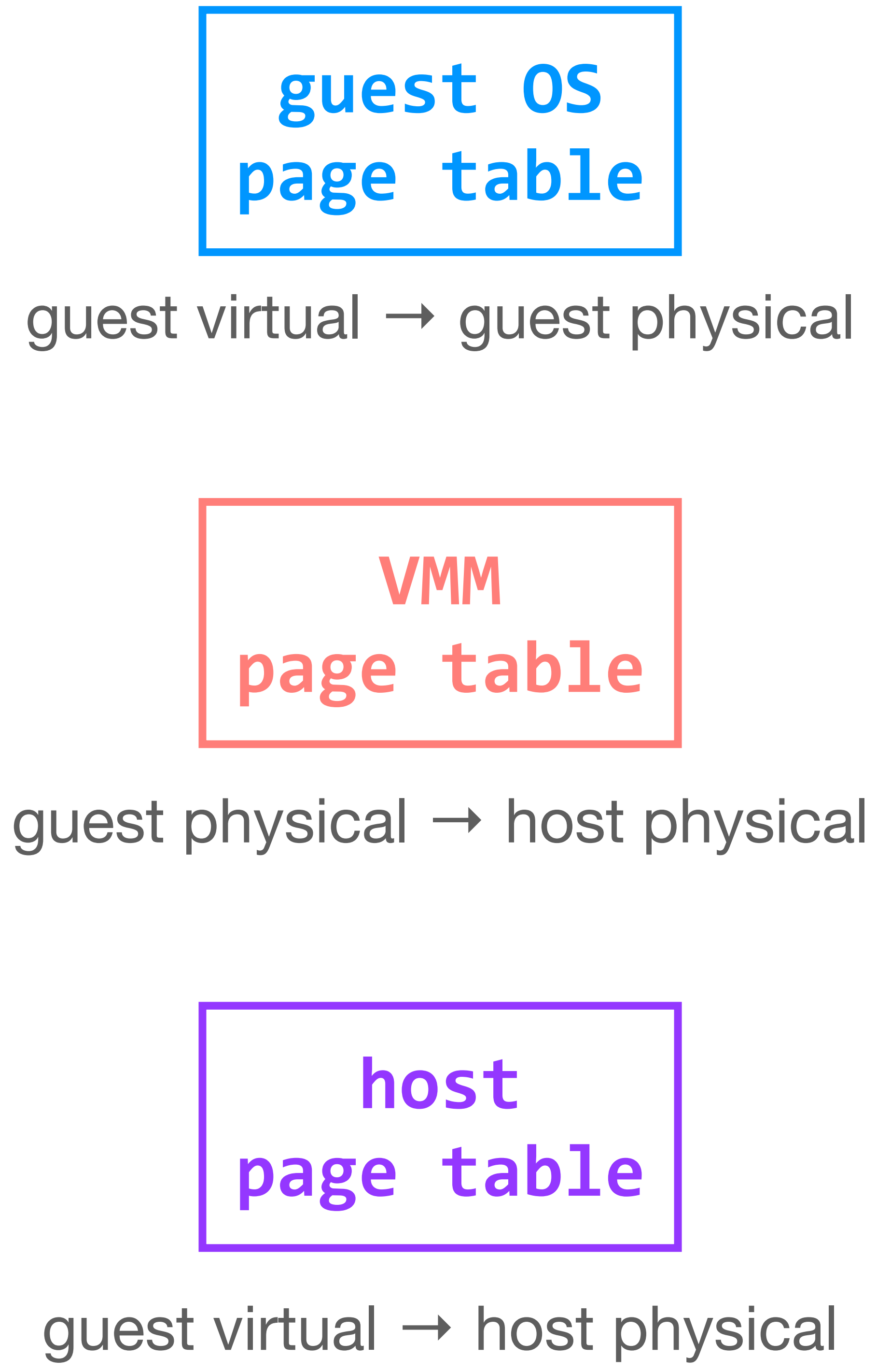


virtual machine monitor virtualizes the physical hardware for the guest OSes

first example: virtualizing memory (again!)

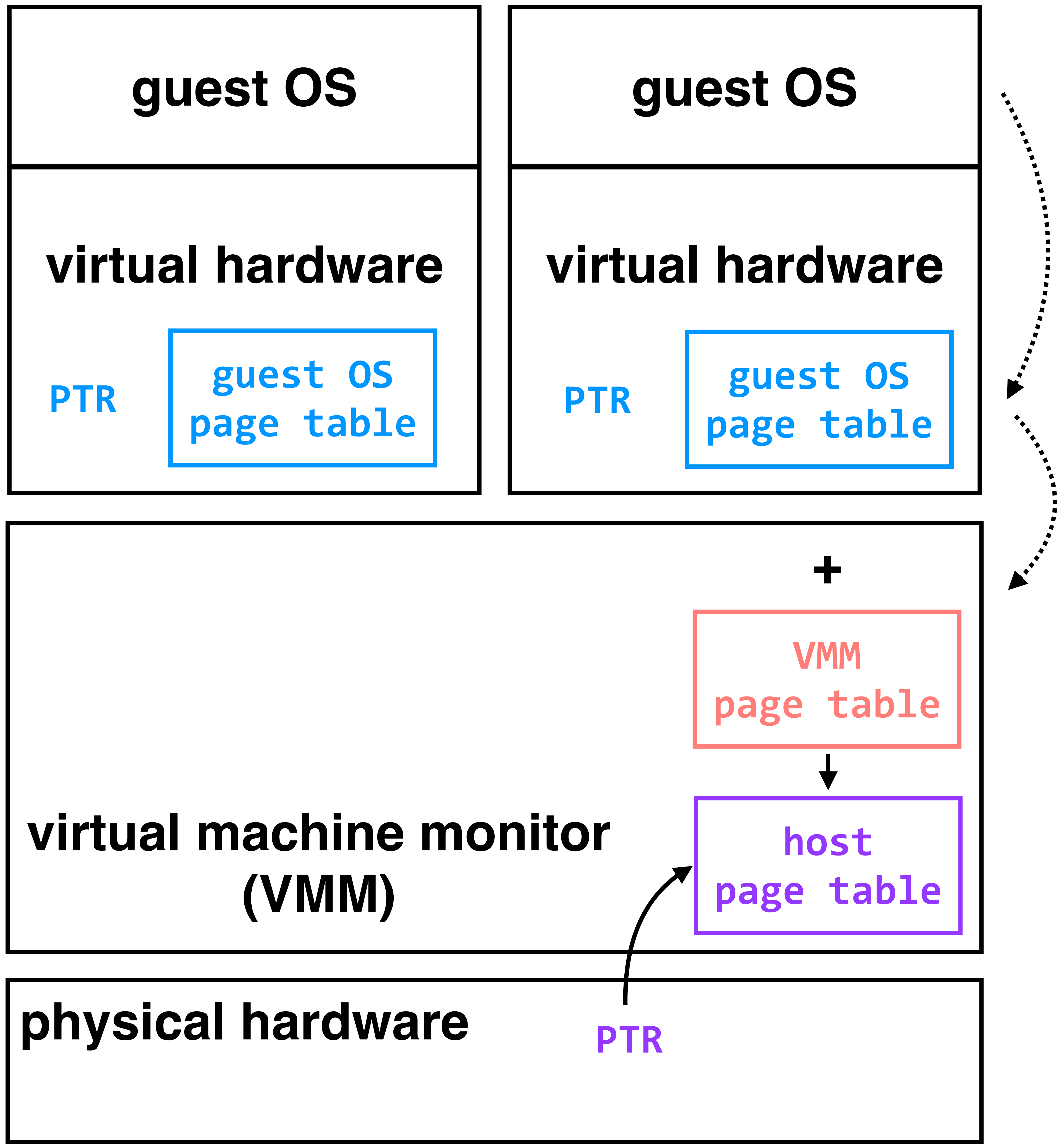


1. guest OS loads its PTR, which triggers an exception; the VMM intercepts
2. VMM combines the guest page table with its own page table to create a host page table
3. physical hardware uses the host page table

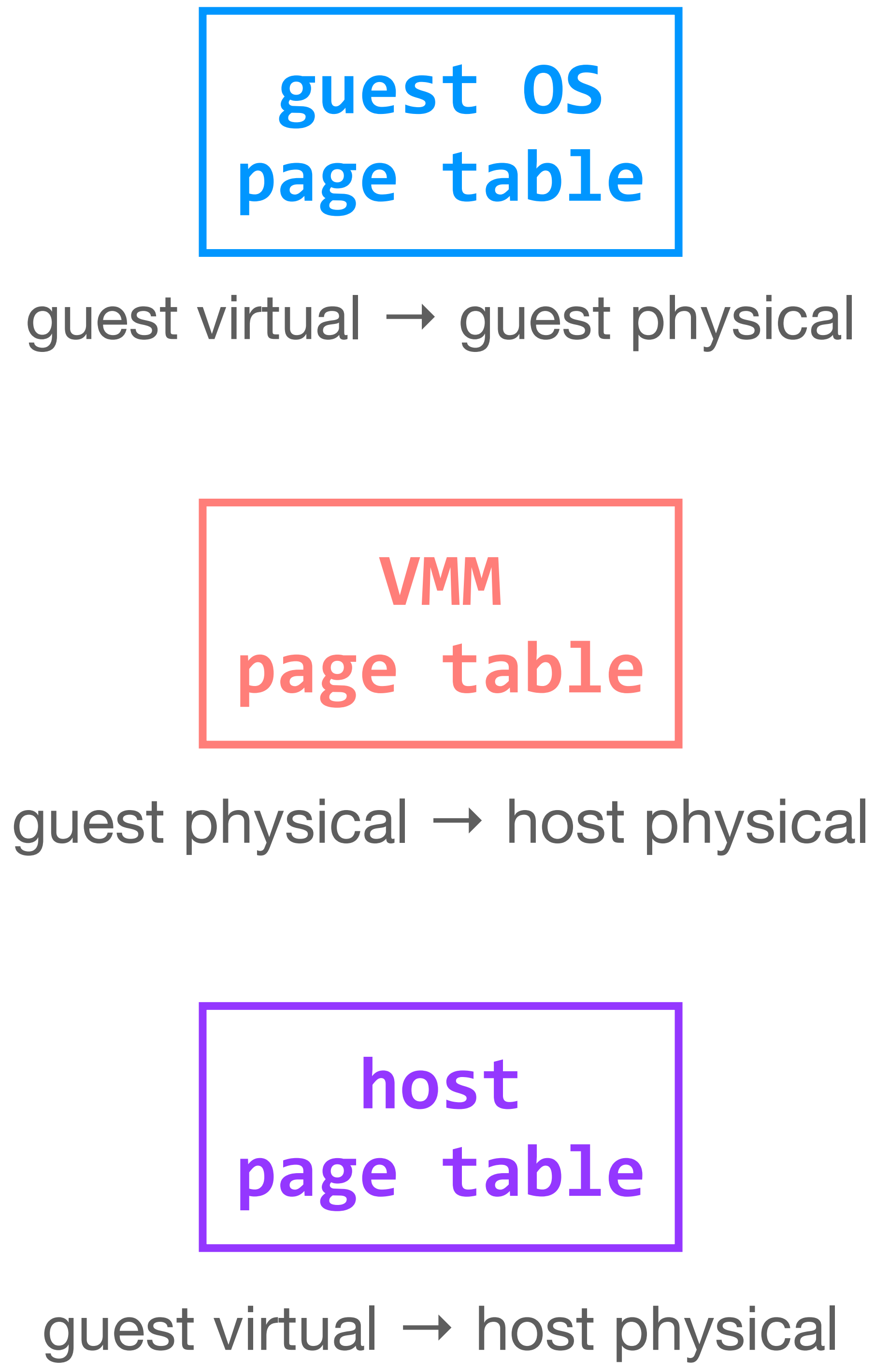


virtual machine monitor virtualizes the physical hardware for the guest OSes

first example: virtualizing memory (again!)

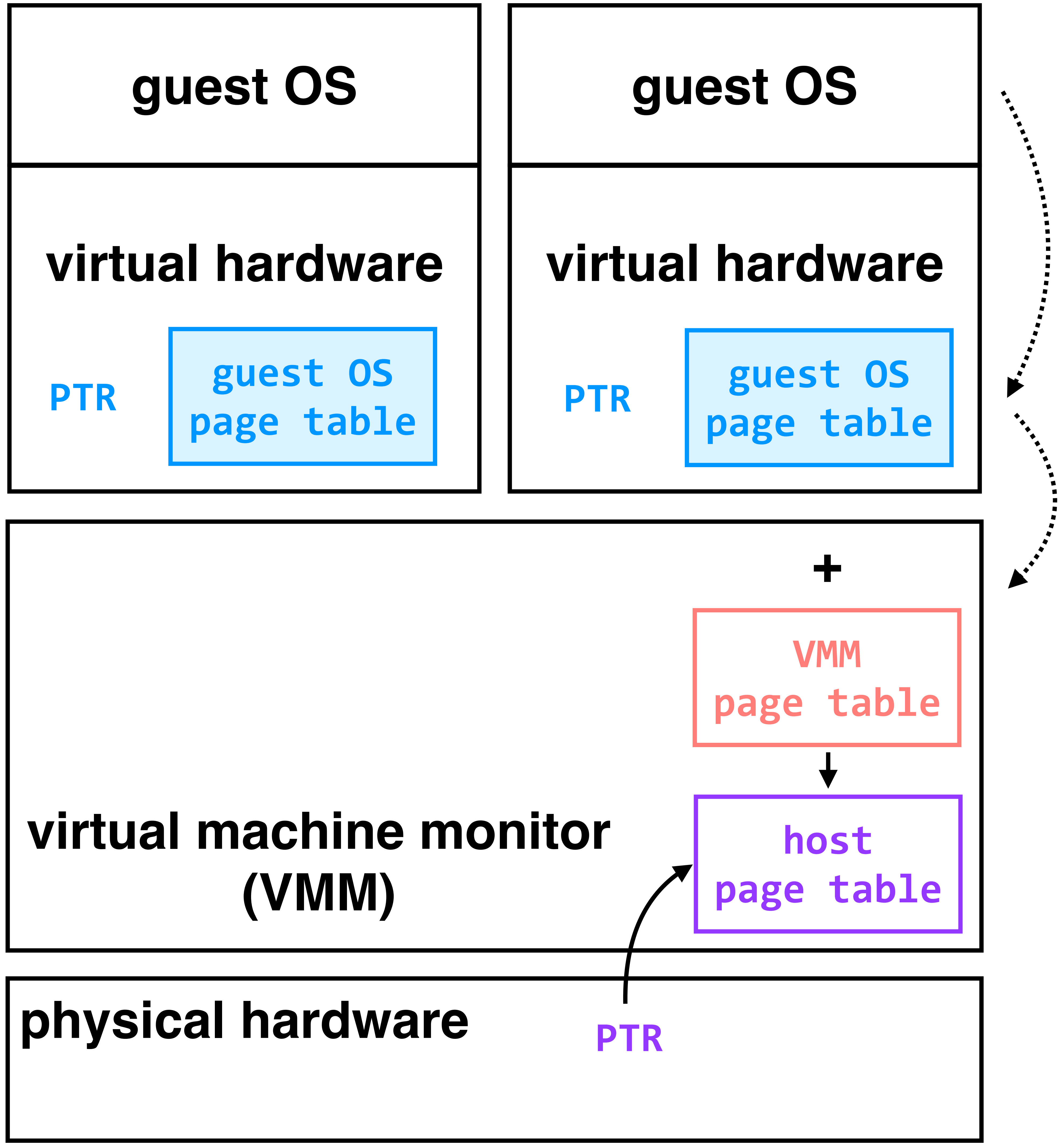


1. guest OS loads its PTR, which triggers an exception; the VMM intercepts
2. VMM combines the guest page table with its own page table to create a host page table
3. physical hardware uses the host page table



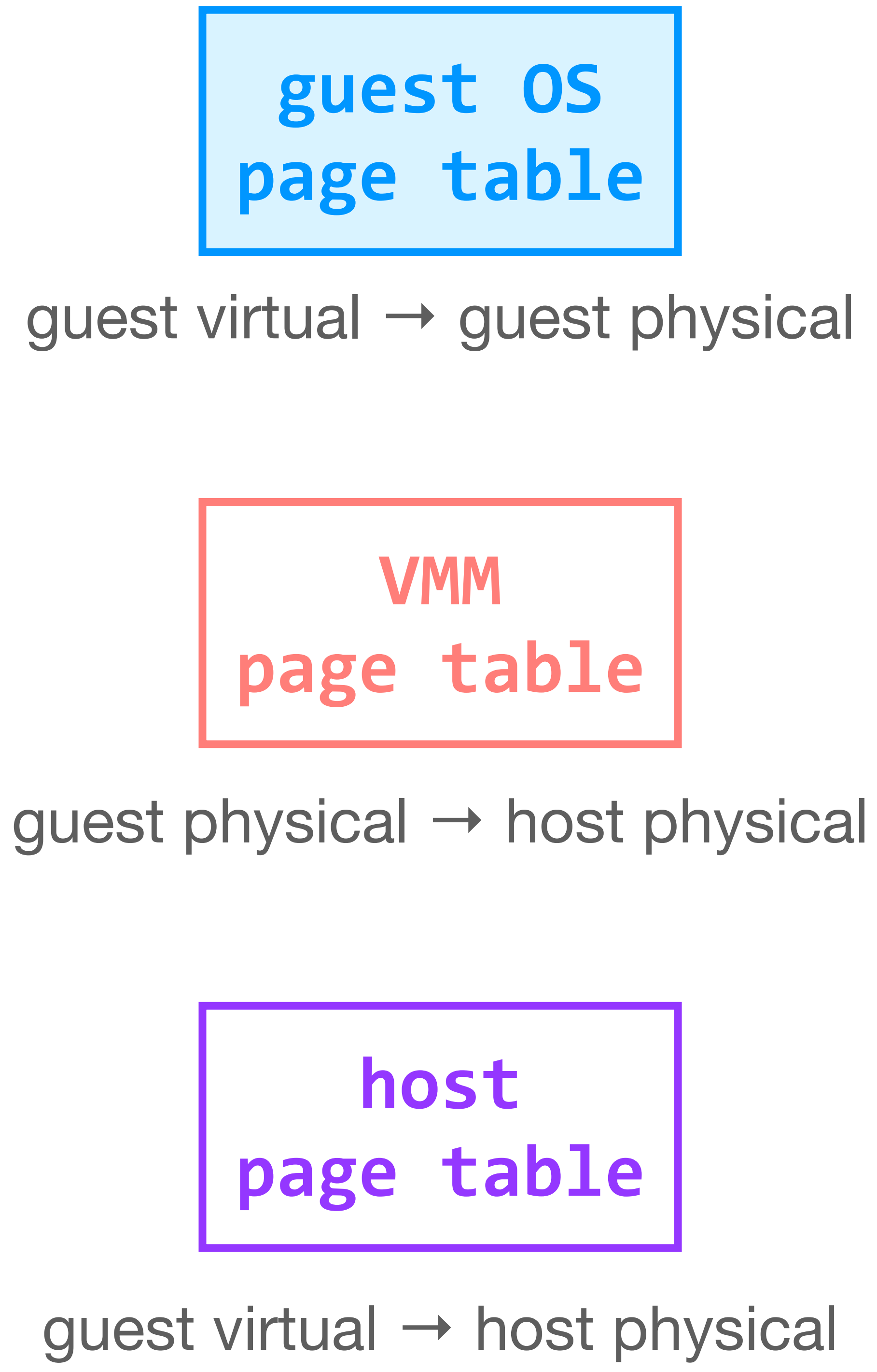
virtual machine monitor virtualizes the physical hardware for the guest OSes

first example: virtualizing memory (again!)



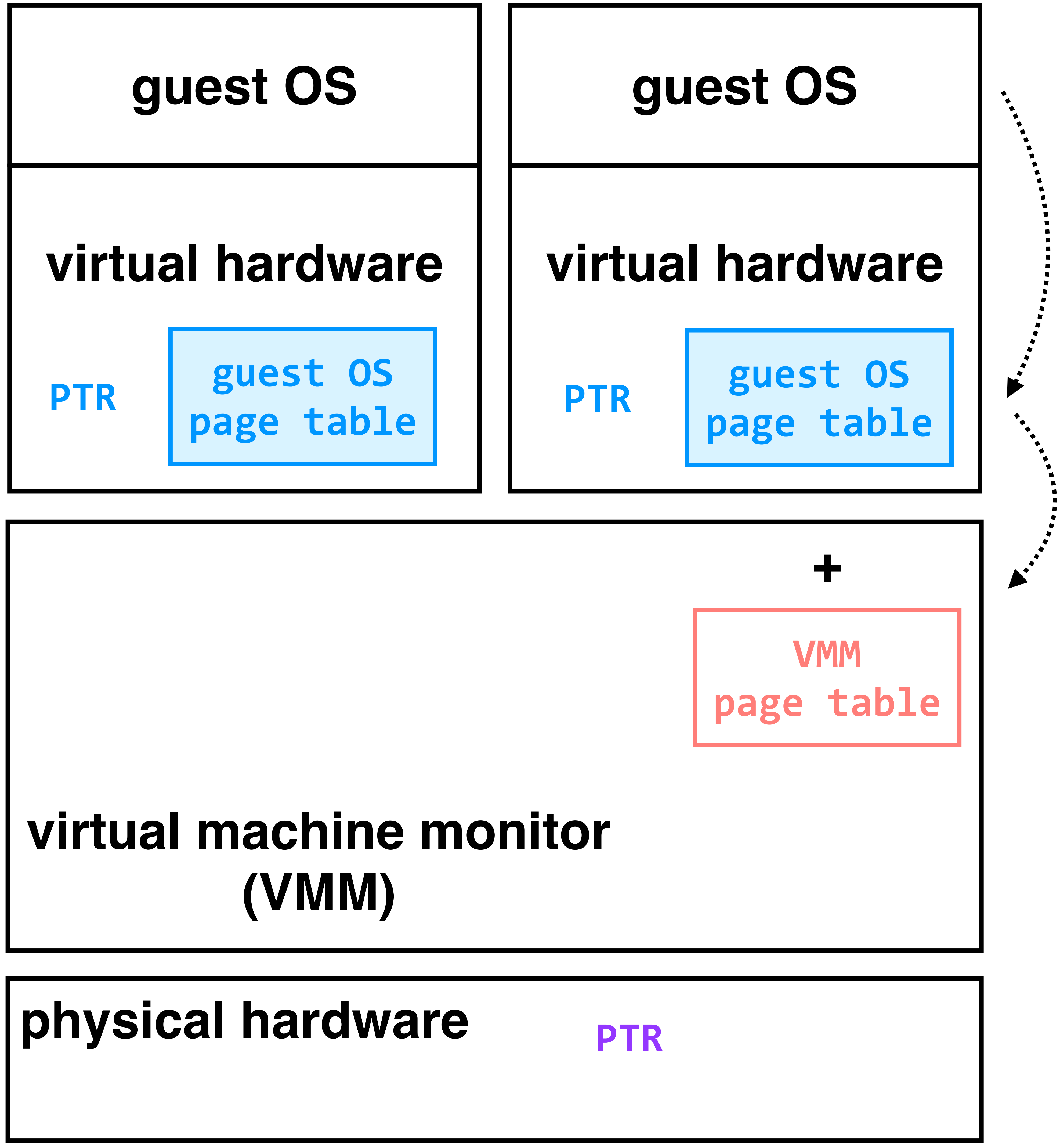
1. guest OS loads its PTR, which triggers an exception; the VMM intercepts

guest OS page tables are marked as **read-only memory** so that modifications to these page tables also trigger exceptions (and thus allow the VMM to update the other tables)

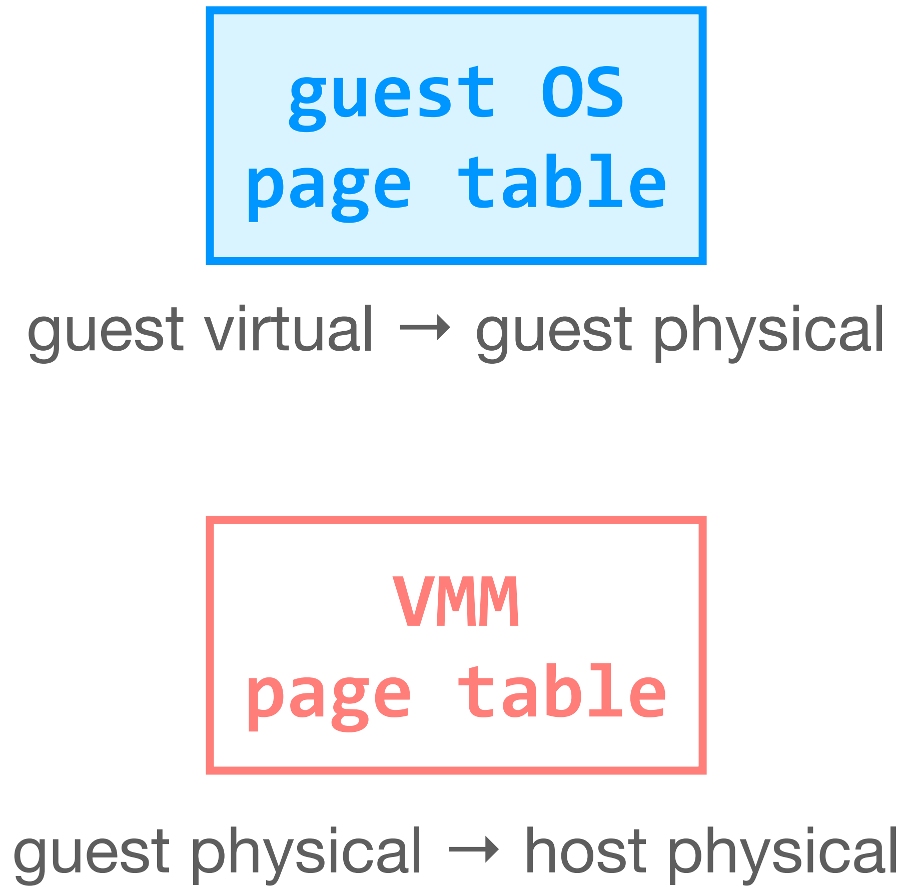


virtual machine monitor virtualizes the physical hardware for the guest OSes

first example: virtualizing memory (again!)

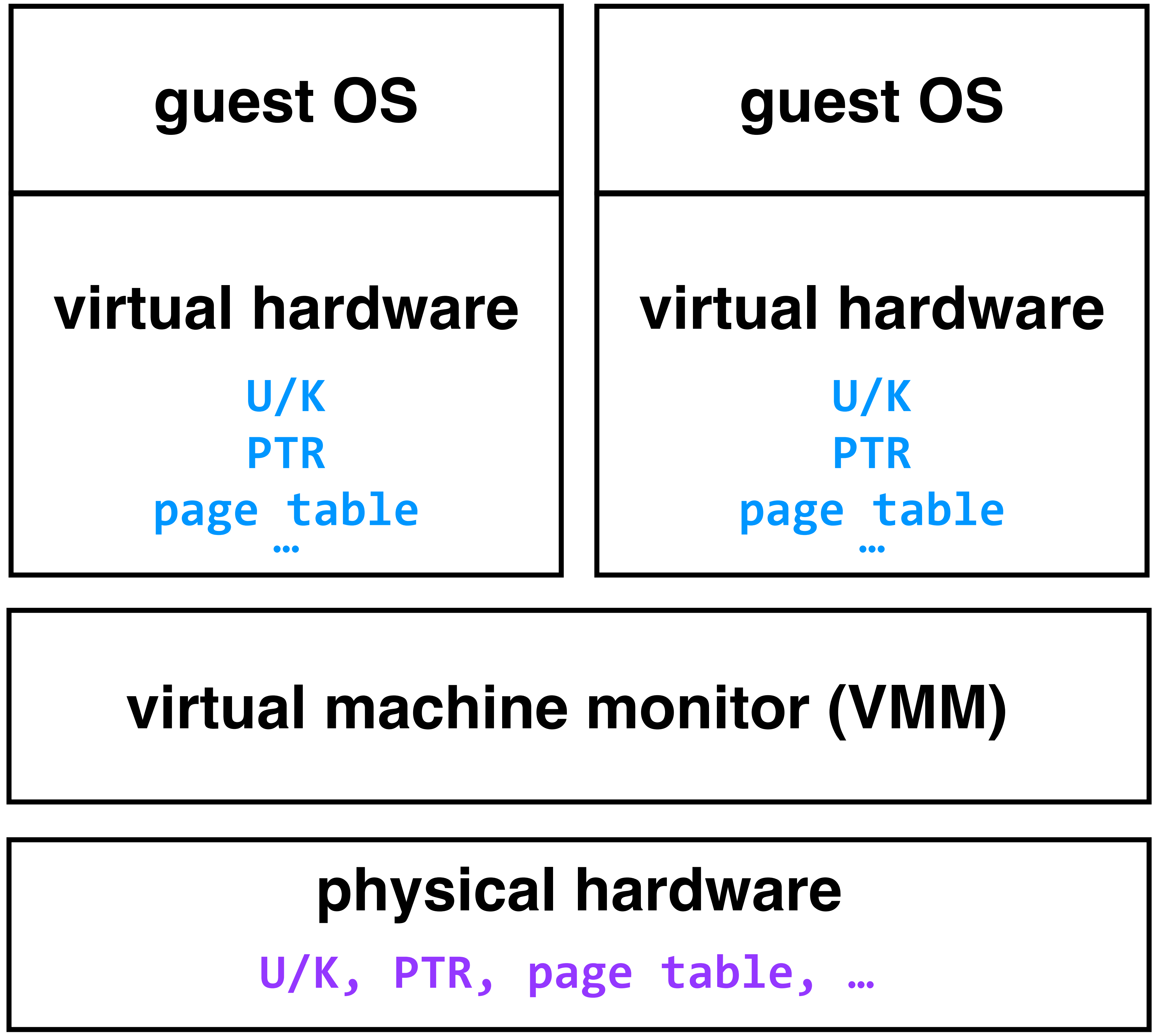


1. guest OS loads its PTR, which triggers an exception; the VMM intercepts



in modern hardware, the physical hardware is aware of both page tables, and performs the translation from guest virtual to host physical itself

virtual machine monitor virtualizes the physical hardware for the guest OSes

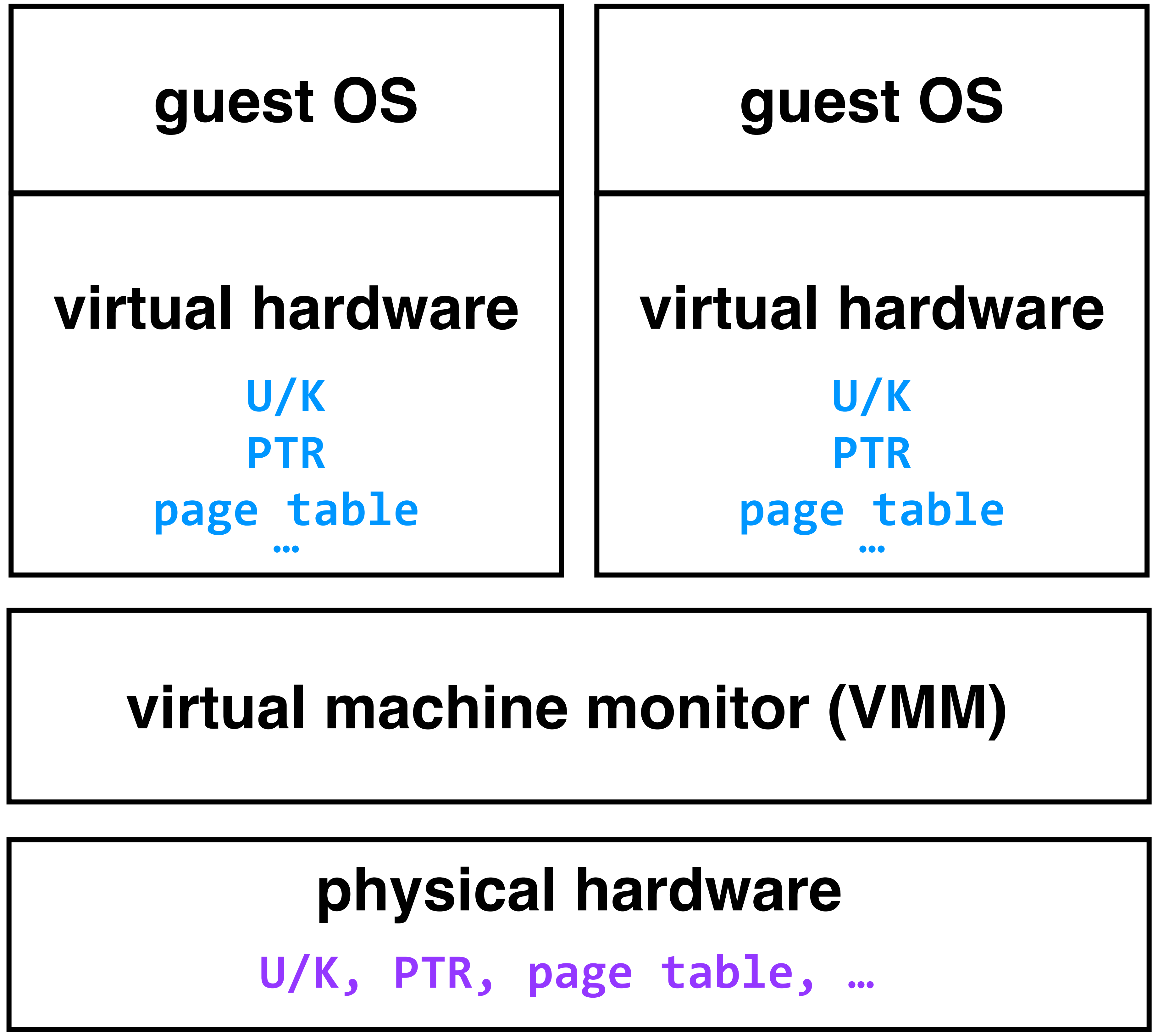


guest OSes run in user mode

privileged instructions in guest OS will cause an exception, which the VMM will intercept ("**trap**") and **emulate**

if the VMM *can't* emulate an instruction, it will send the exception back to the guest OS for handling

virtual machine monitor virtualizes the physical hardware for the guest OSes



guest OSes run in user mode

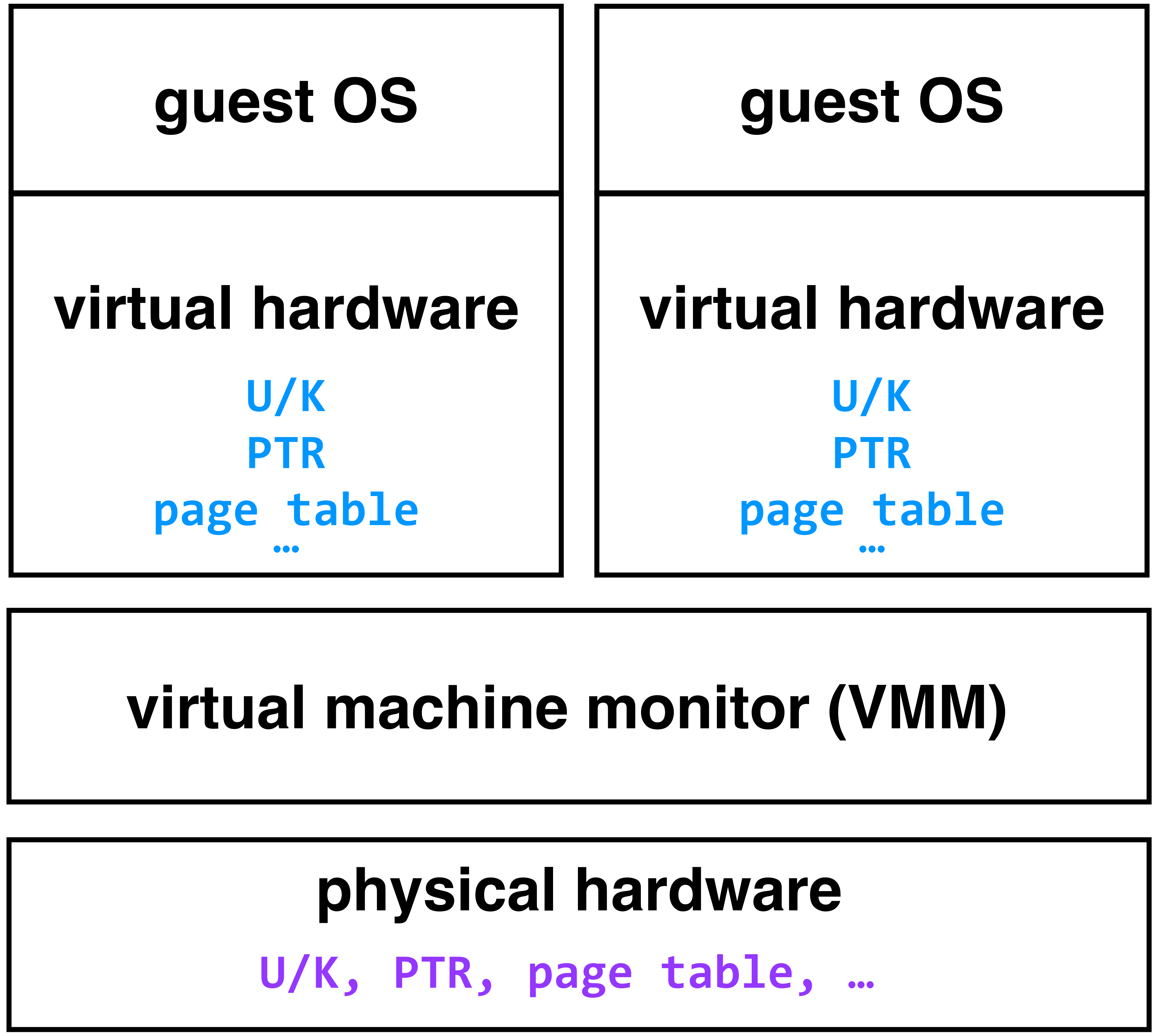
privileged instructions in guest OS will cause an exception, which the VMM will intercept ("**trap**") and **emulate**

if the VMM *can't* emulate an instruction, it will send the exception back to the guest OS for handling

figuring out how to emulate an instruction is not enough; we also need to make sure that the VMM is trapping all relevant instructions

virtual machine monitor virtualizes the physical hardware for the guest OSes

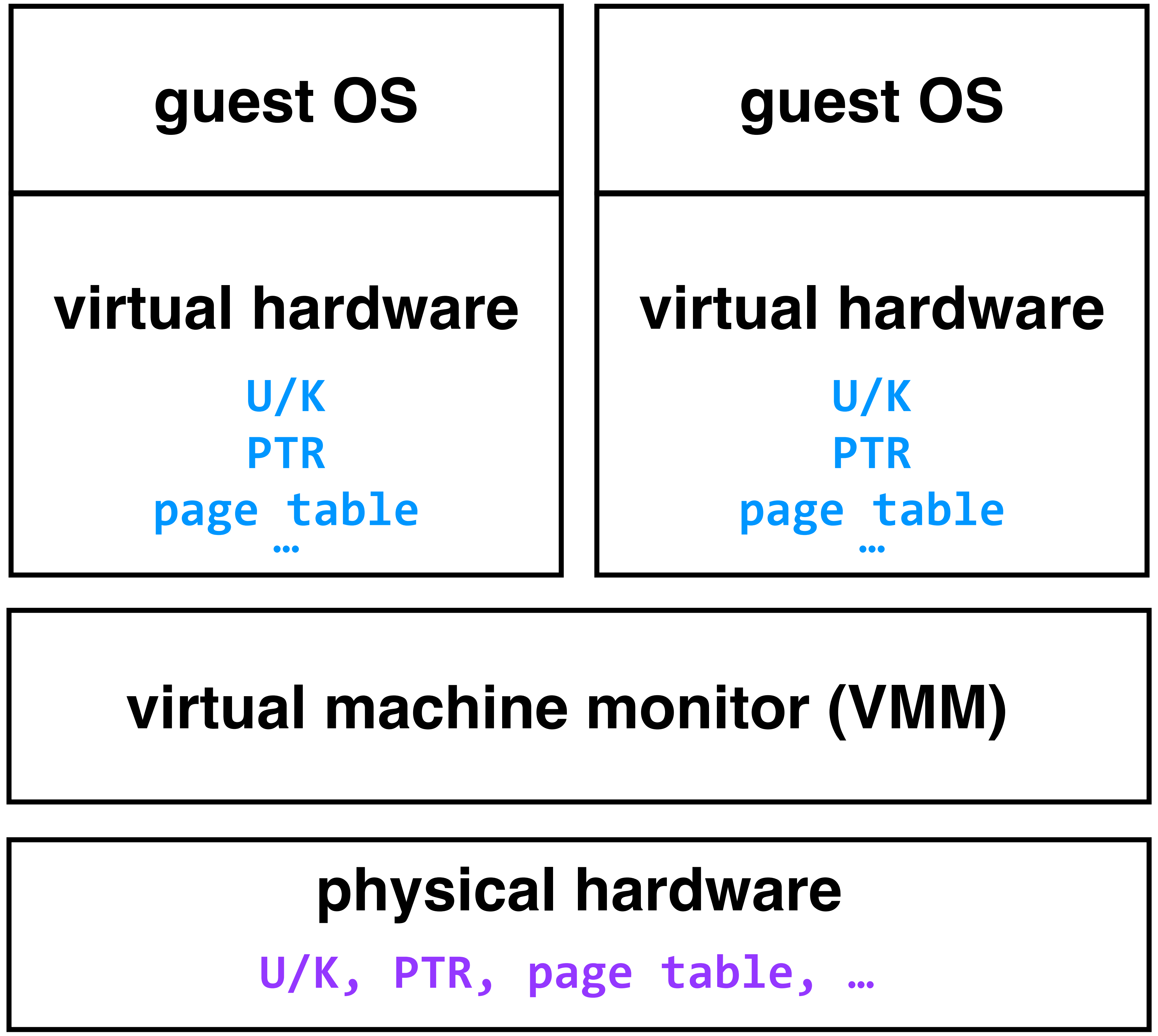
second example: virtualizing the U/K bit



figuring out how to emulate an instruction is not enough; we also need to make sure that the VMM is trapping all relevant instructions

virtual machine monitor virtualizes the physical hardware for the guest OSes

second example: virtualizing the U/K bit

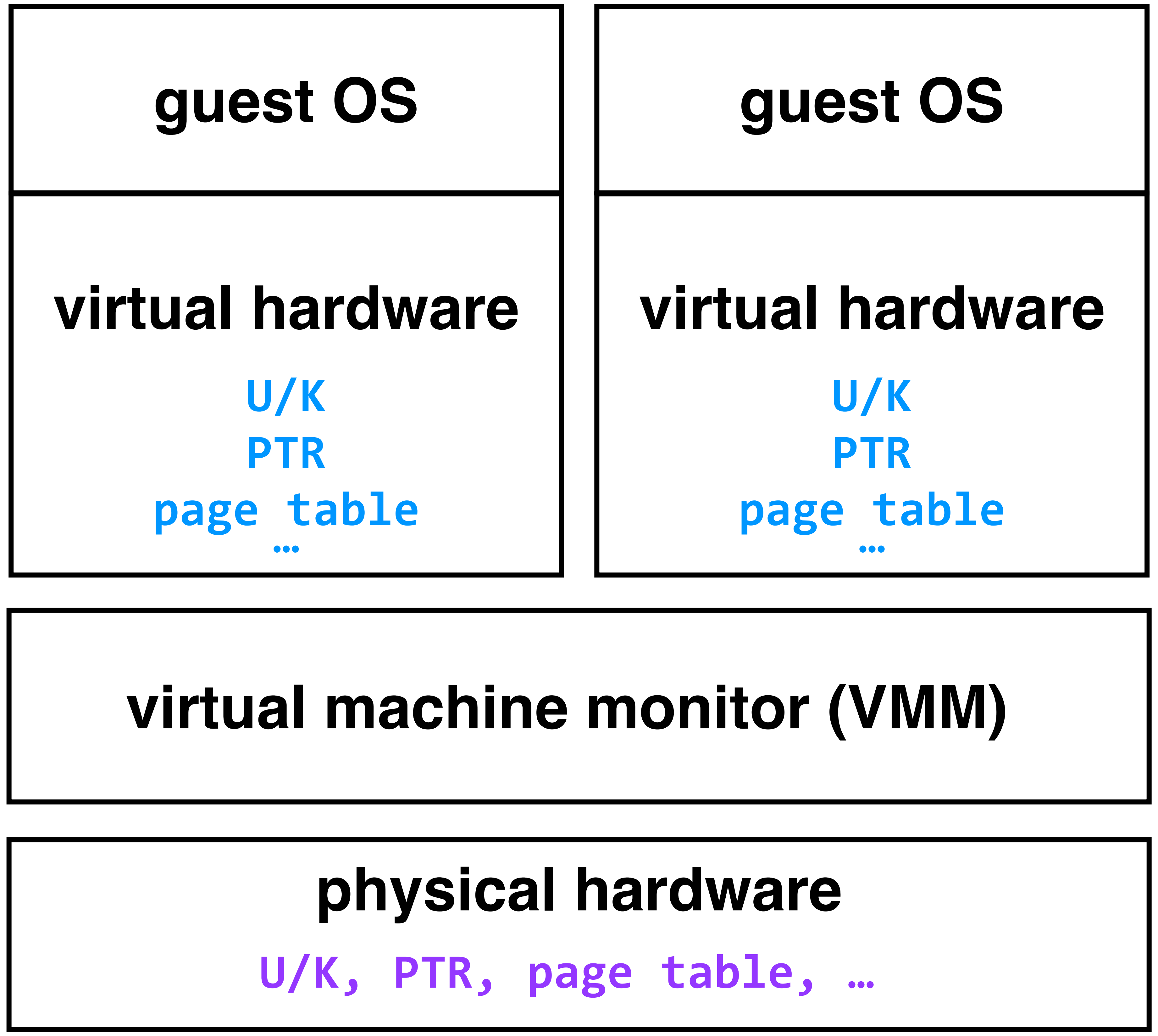


para-virtualization: modify guest OS slightly

figuring out how to emulate an instruction is not enough; we also need to make sure that the VMM is trapping all relevant instructions

virtual machine monitor virtualizes the physical hardware for the guest OSes

second example: virtualizing the U/K bit



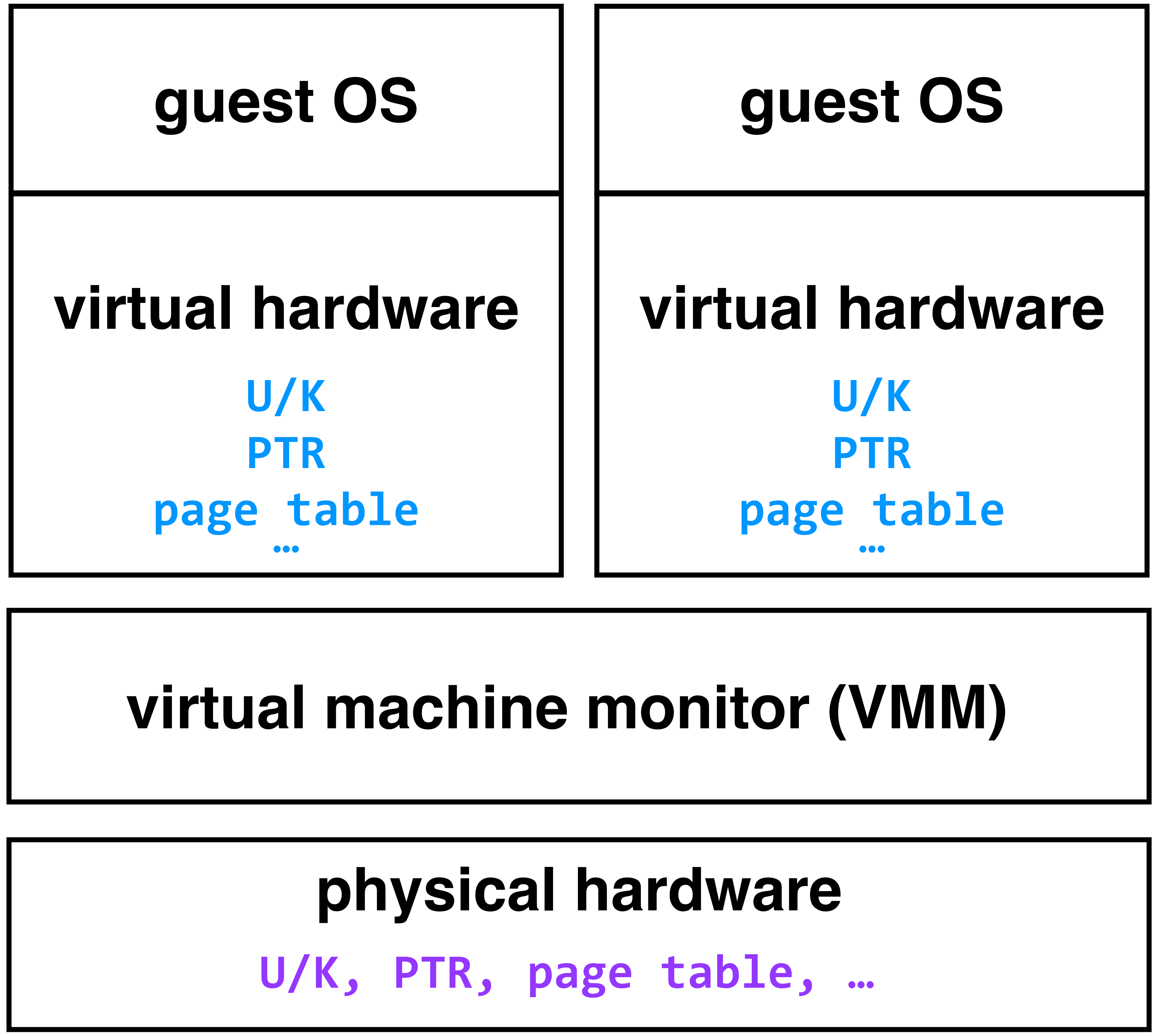
para-virtualization: modify guest OS slightly

binary translation: VMM replaces problematic instructions with ones that it can trap and emulate

figuring out how to emulate an instruction is not enough; we also need to make sure that the VMM is trapping all relevant instructions

virtual machine monitor virtualizes the physical hardware for the guest OSes

second example: virtualizing the U/K bit



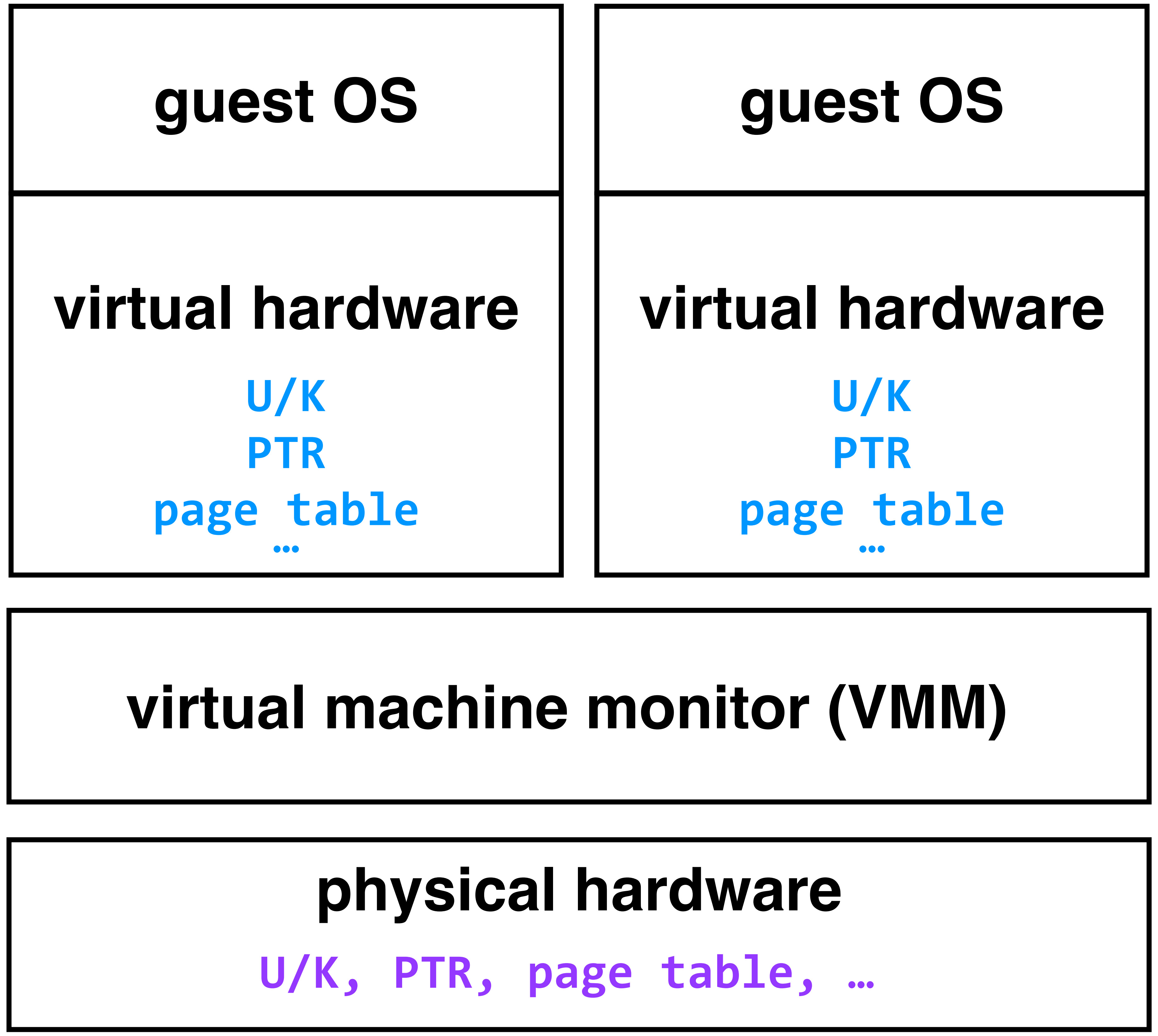
para-virtualization: modify guest OS slightly

binary translation: VMM replaces problematic instructions with ones that it can trap and emulate

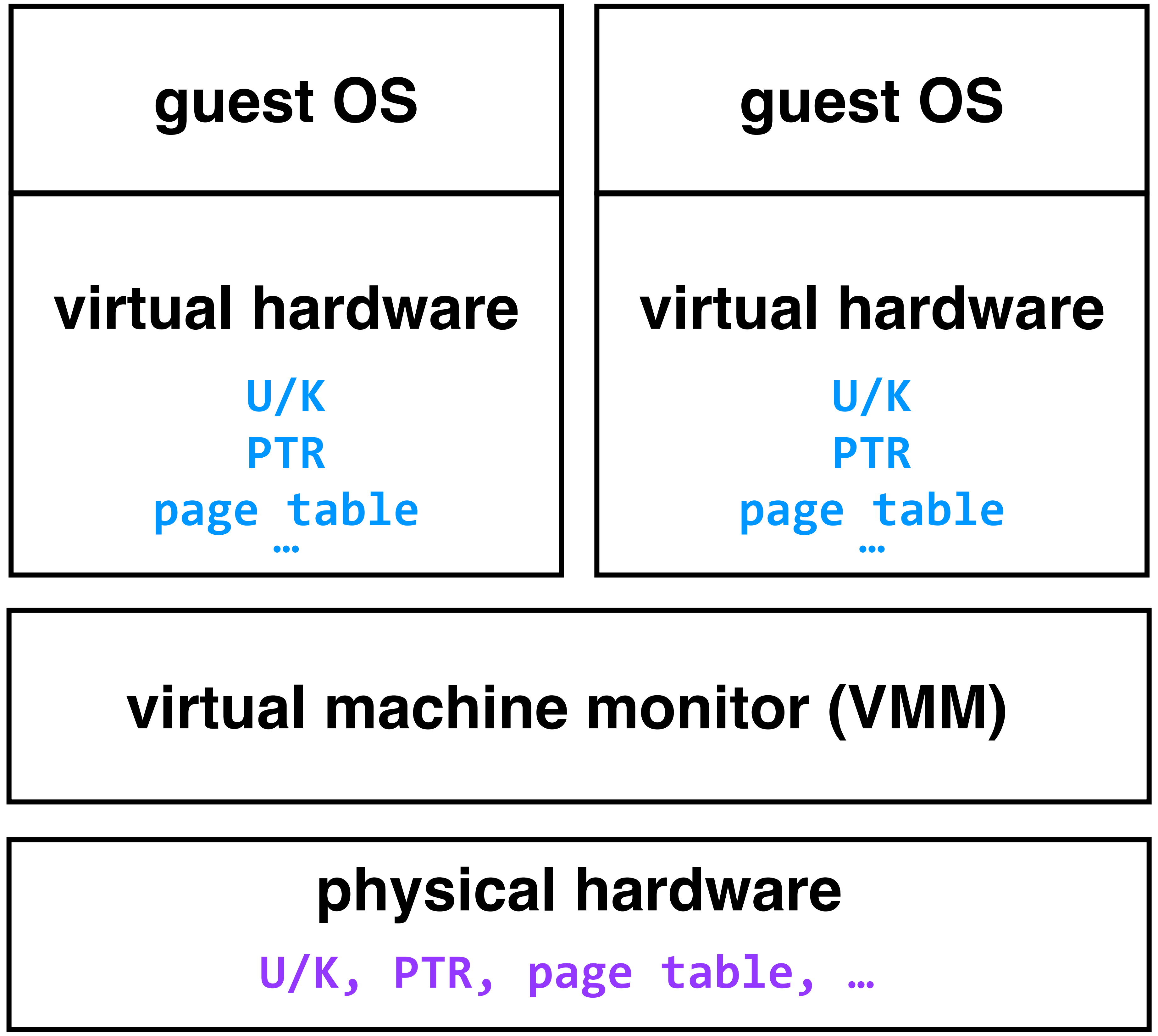
hardware support: architecture provides a special operating mode for VMMs in addition to user mode, kernel mode

figuring out how to emulate an instruction is not enough; we also need to make sure that the VMM is trapping all relevant instructions

virtual machine monitor virtualizes the physical hardware for the guest OSes

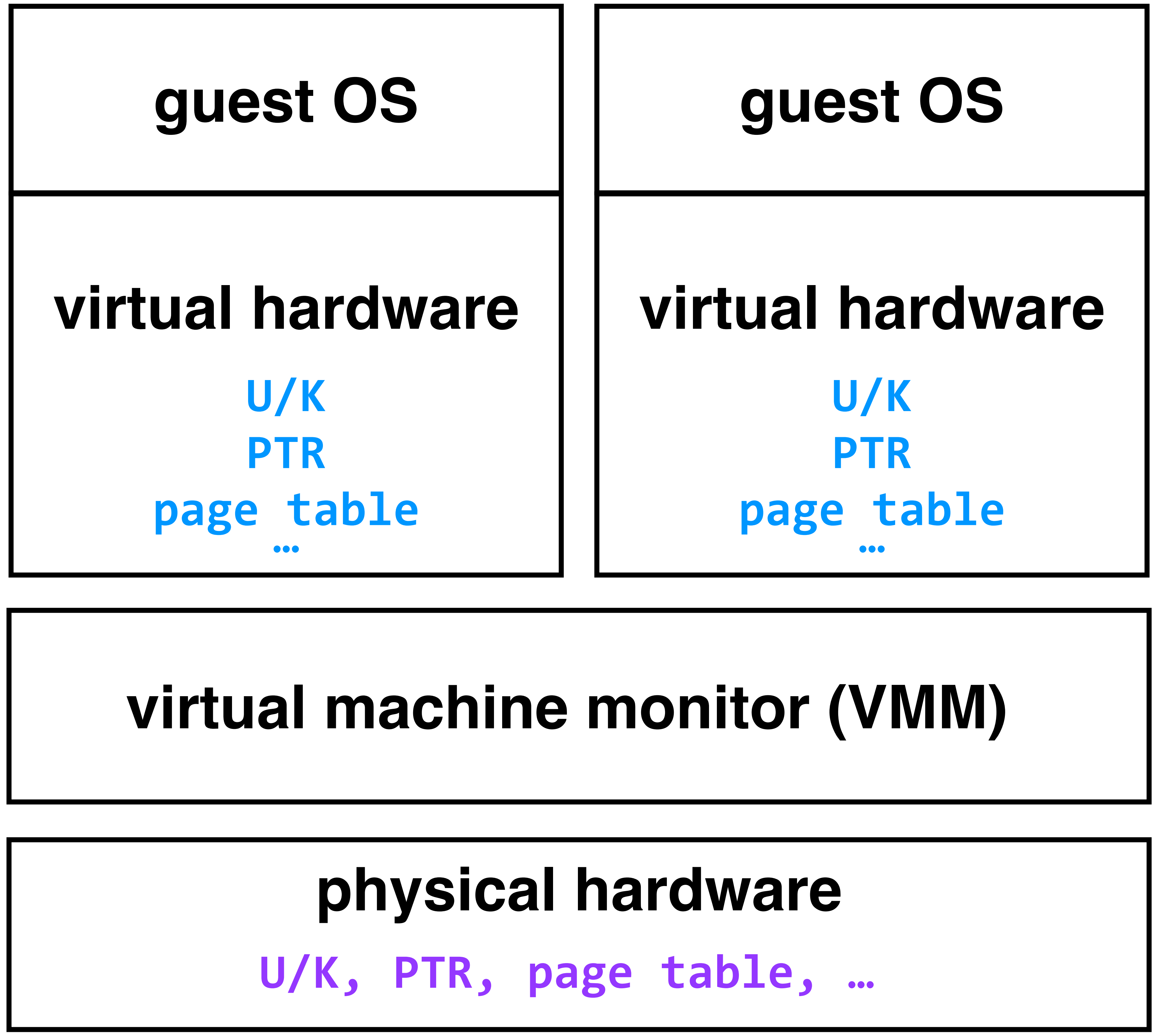


virtual machine monitor virtualizes the physical hardware for the guest OSes



VMMs work by **trapping** and **emulating** important instructions

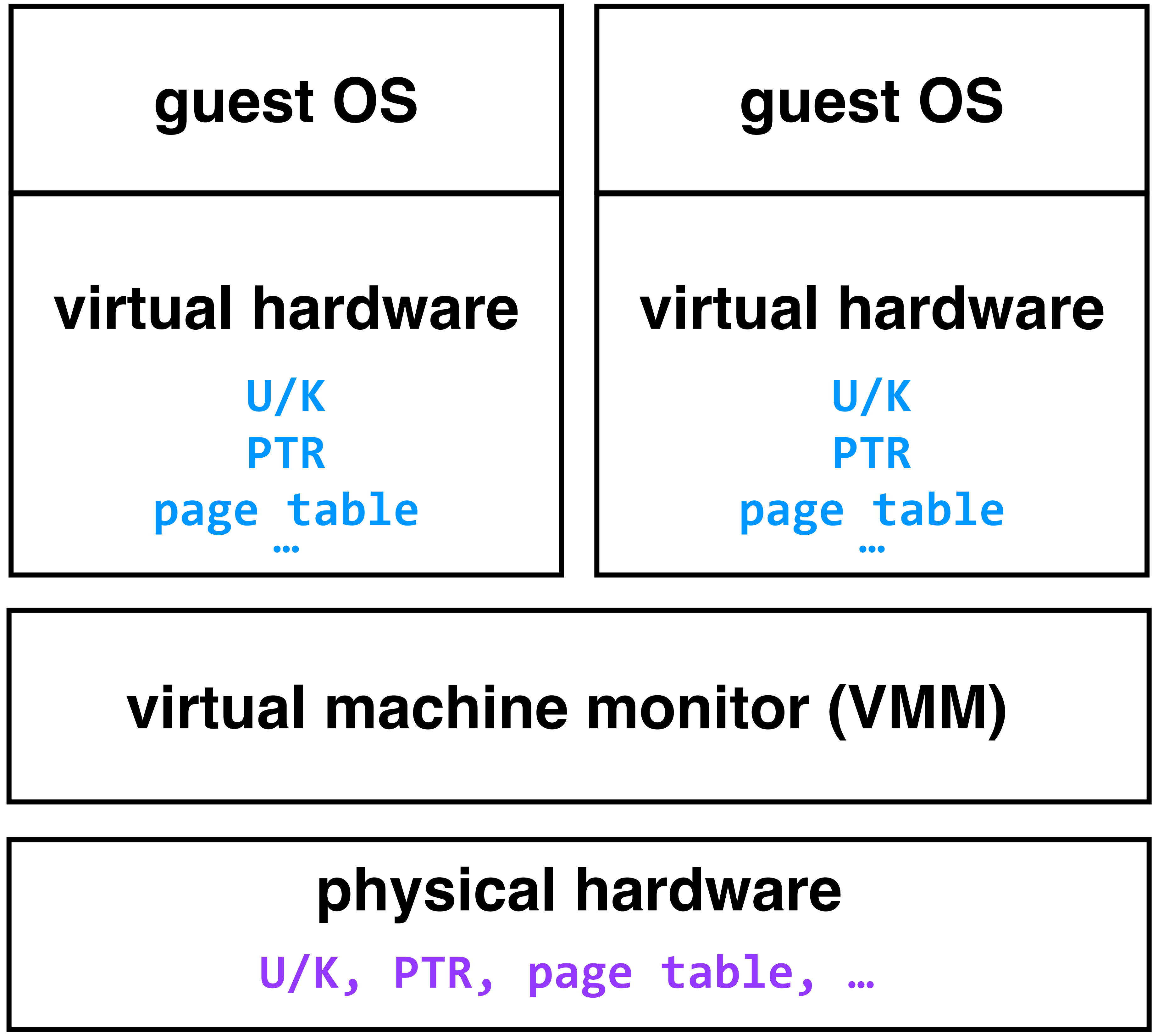
virtual machine monitor virtualizes the physical hardware for the guest OSes



VMMs work by **trapping** and **emulating** important instructions

the actual **emulation** looks different depending on what we're trying to do. at times — e.g., in the case of virtual memory — it's a fairly straightforward extension of what the OS does

virtual machine monitor virtualizes the physical hardware for the guest OSes

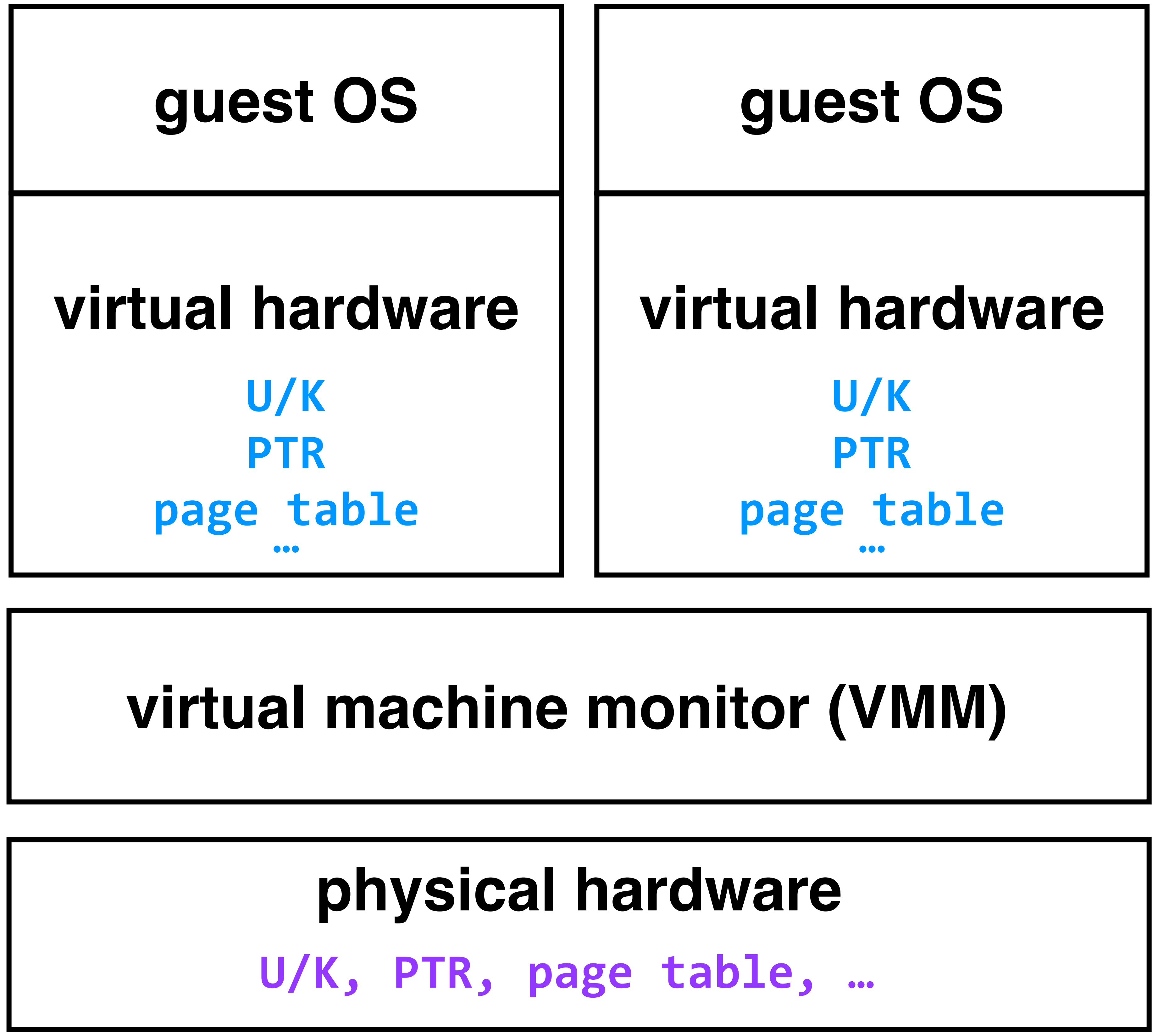


VMMs work by **trapping** and **emulating** important instructions

the actual **emulation** looks different depending on what we're trying to do. at times — e.g., in the case of virtual memory — it's a fairly straightforward extension of what the OS does

modern architectures build support for virtualization into their CPUs, which allow the VMM to operate **efficiently**

virtual machine monitor virtualizes the physical hardware for the guest OSes



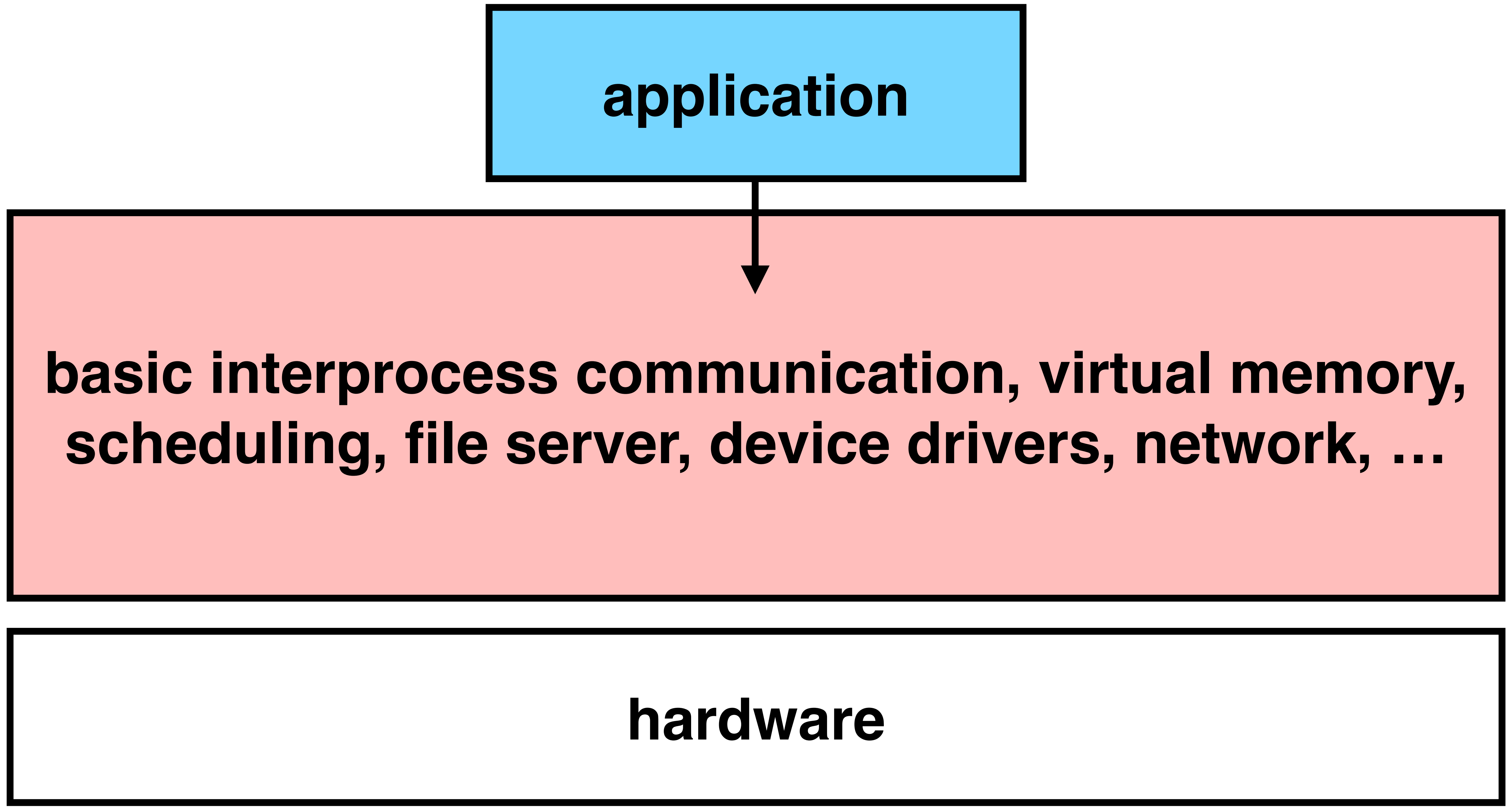
VMMs work by **trapping** and **emulating** important instructions

the actual **emulation** looks different depending on what we're trying to do. at times — e.g., in the case of virtual memory — it's a fairly straightforward extension of what the OS does

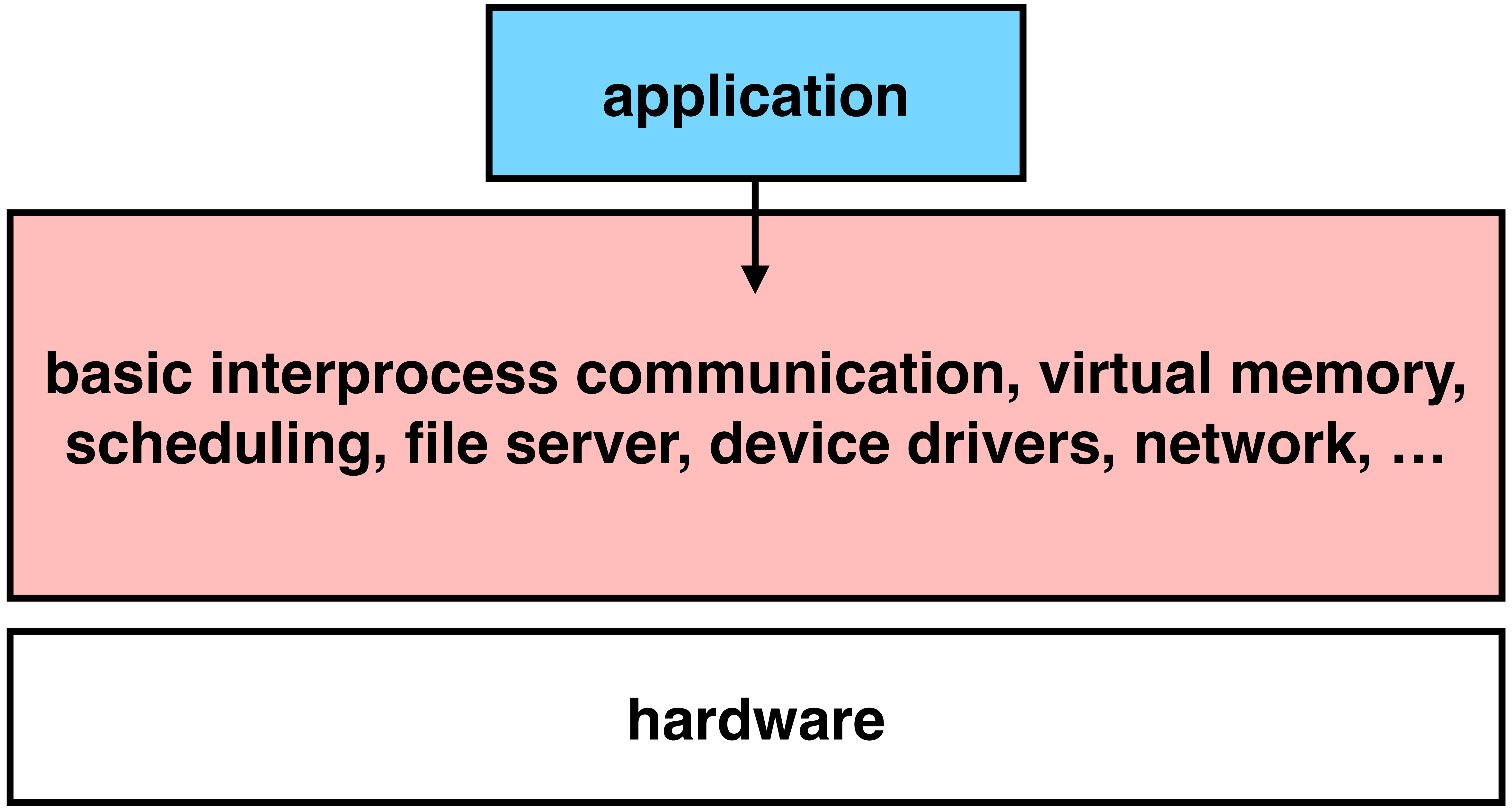
modern architectures build support for virtualization into their CPUs, which allow the VMM to operate **efficiently**

this is all yet another application of **virtualization**. the details change depending on what problem we're solving, but the goal of virtualization remains the same.

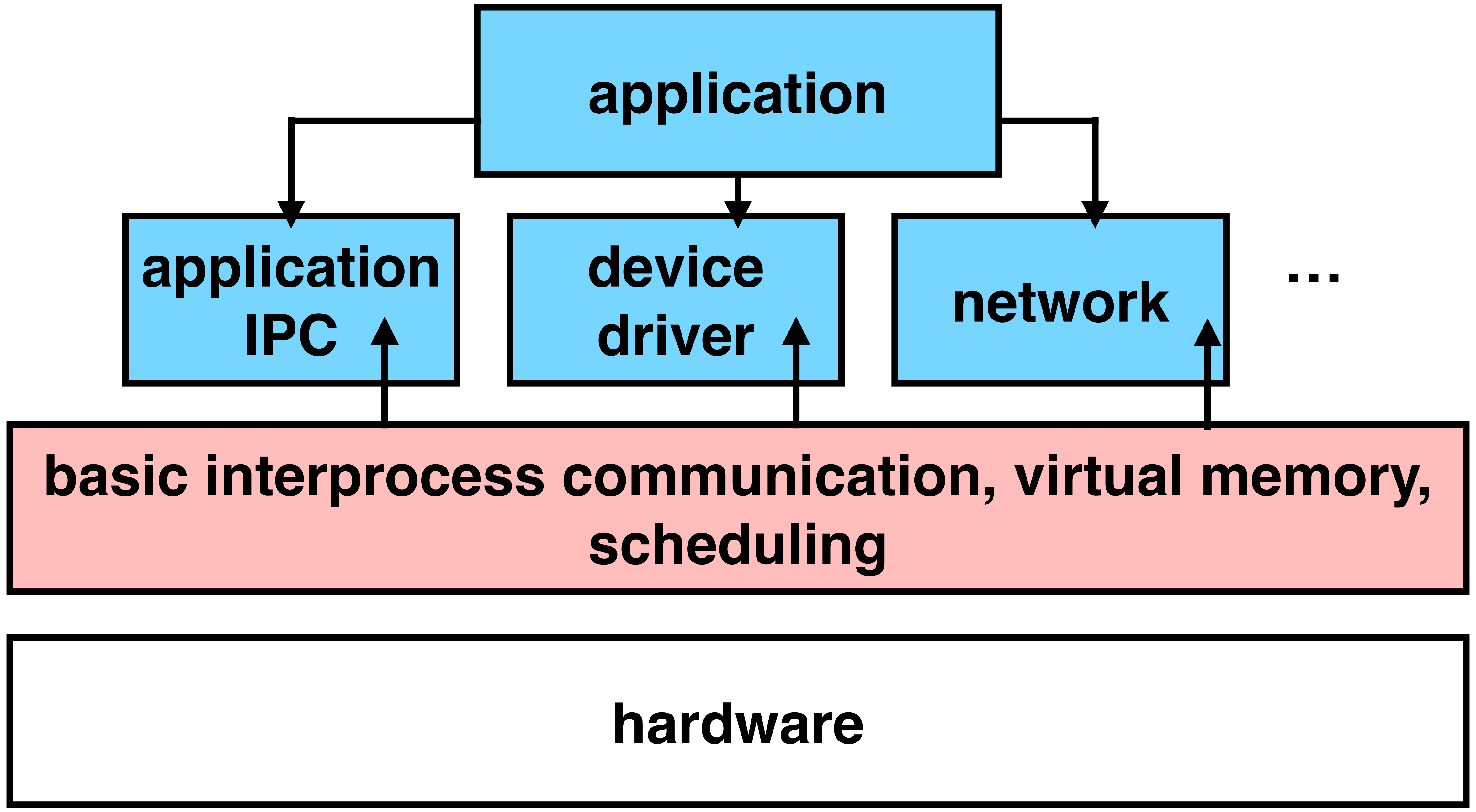
monolithic kernel: no enforced modularity within the kernel itself



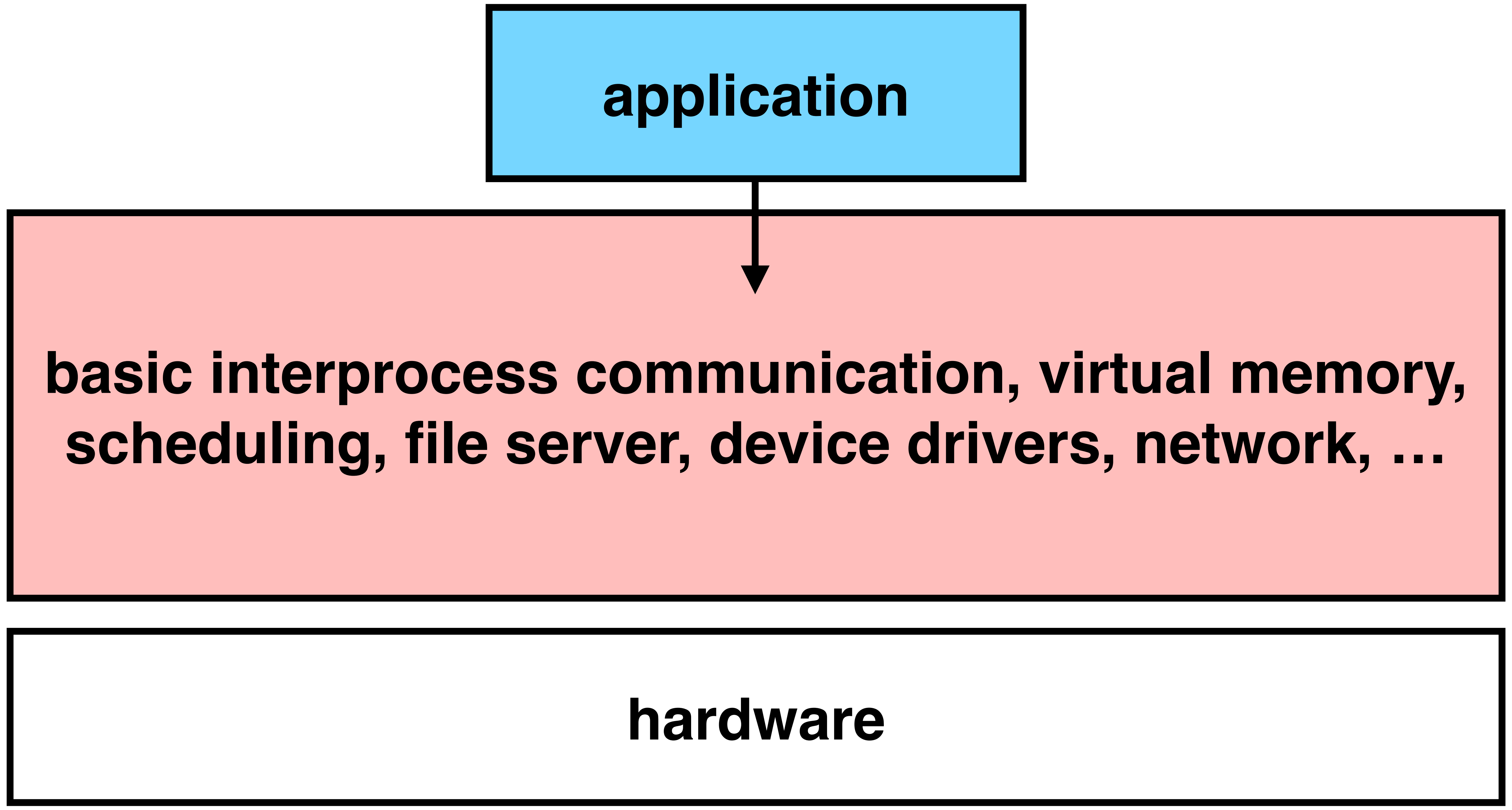
monolithic kernel: no enforced modularity within the kernel itself



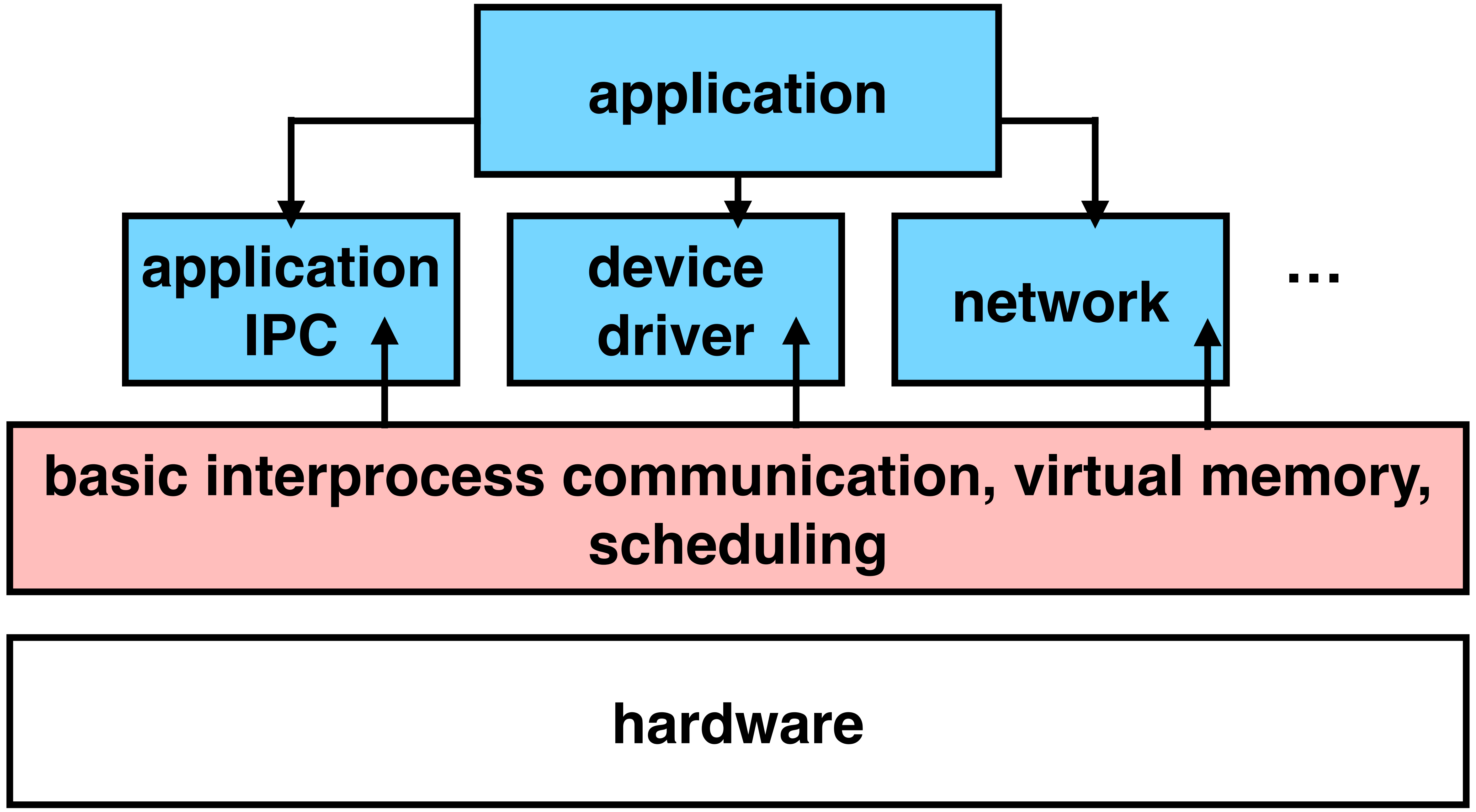
microkernels: enforce modularity by putting subsystems in user programs



monolithic kernel: no enforced modularity within the kernel itself



microkernels: enforce modularity by putting subsystems in user programs



despite the modularity, it’s not clear that redesigning an operating system from a monolithic kernel to a microkernel is a good idea, in part for reasons of **performance**

virtual machines allow us to run multiple **isolated** OSes on a single physical machine, similar to how we used an OS to run multiple programs on a single CPU

this set-up also enables many cloud compute infrastructures, which back many of the applications you use today

virtual machines allow us to run multiple **isolated** OSes on a single physical machine, similar to how we used an OS to run multiple programs on a single CPU

this set-up also enables many cloud compute infrastructures, which back many of the applications you use today

monolithic kernels provide no enforced modularity within the kernel. **microkernels** do, but redesigning monolithic kernels as microkernels is challenging

virtual machines allow us to run multiple **isolated** OSes on a single physical machine, similar to how we used an OS to run multiple programs on a single CPU

this set-up also enables many cloud compute infrastructures, which back many of the applications you use today

monolithic kernels provide no enforced modularity within the kernel. **microkernels** do, but redesigning monolithic kernels as microkernels is challenging

we have cared about **performance** in all aspects of our operating systems journey so far, and next time we'll start to think about performance more generally

virtual machines allow us to run multiple **isolated** OSes on a single physical machine, similar to how we used an OS to run multiple programs on a single CPU

this set-up also enables many cloud compute infrastructures, which back many of the applications you use today

monolithic kernels provide no enforced modularity within the kernel. **microkernels** do, but redesigning monolithic kernels as microkernels is challenging

we have cared about **performance** in all aspects of our operating systems journey so far, and next time we'll start to think about performance more generally

you have now seen **virtualization** applied as a solution to many different problems. the details change depending on what problem we're solving, but the goal of virtualization remains the same.