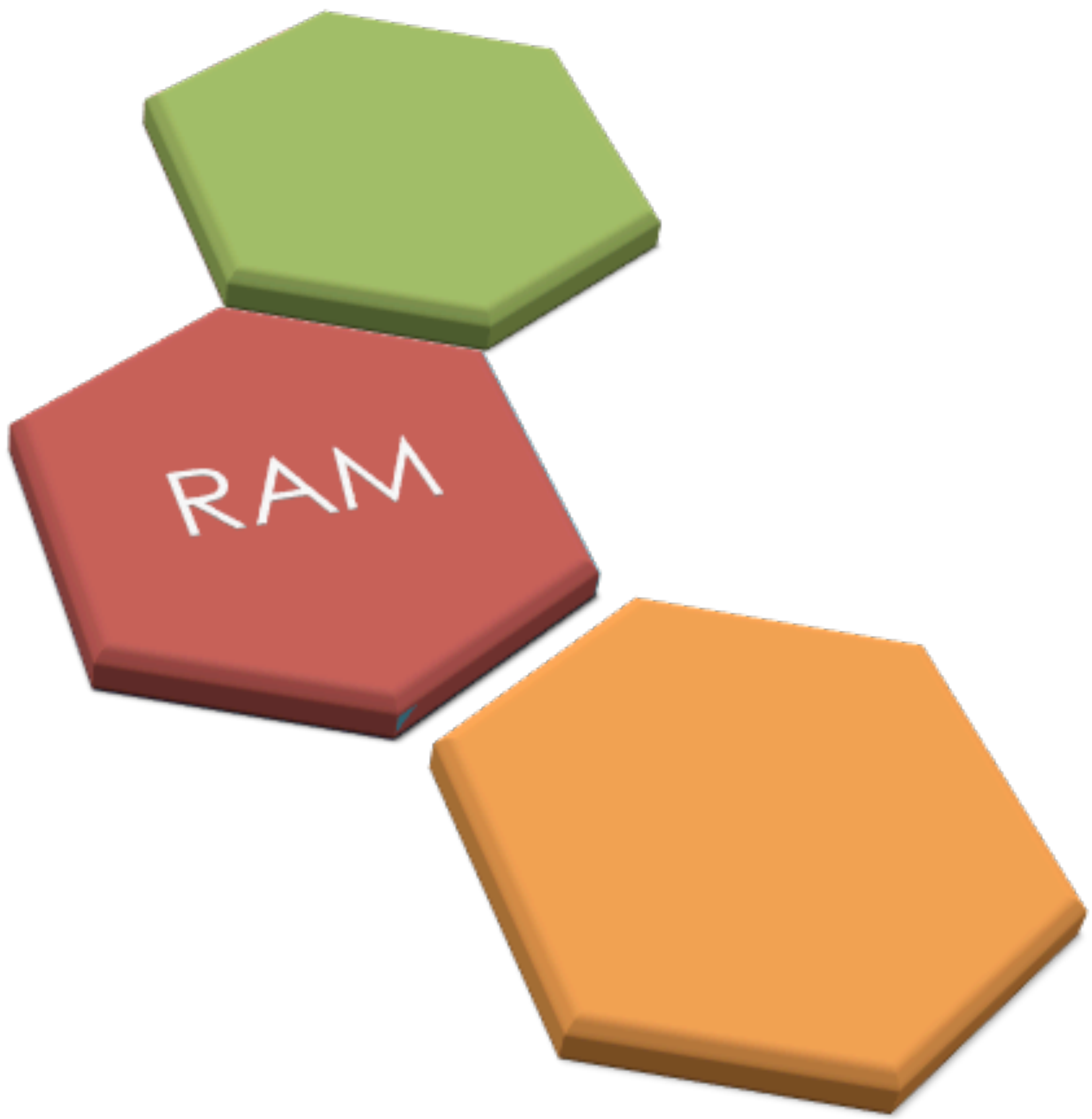
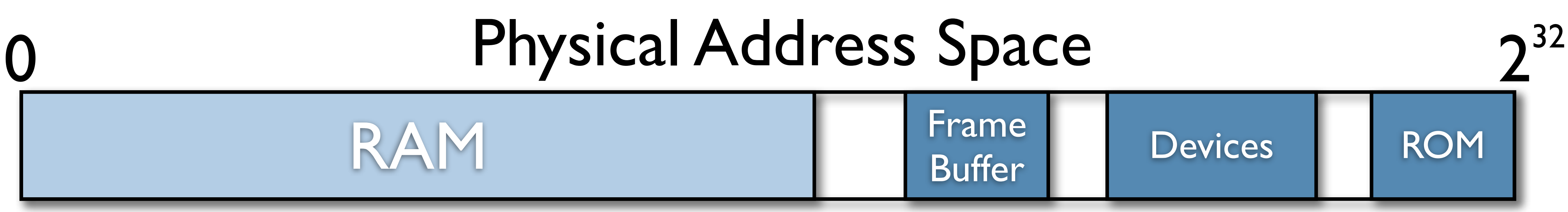
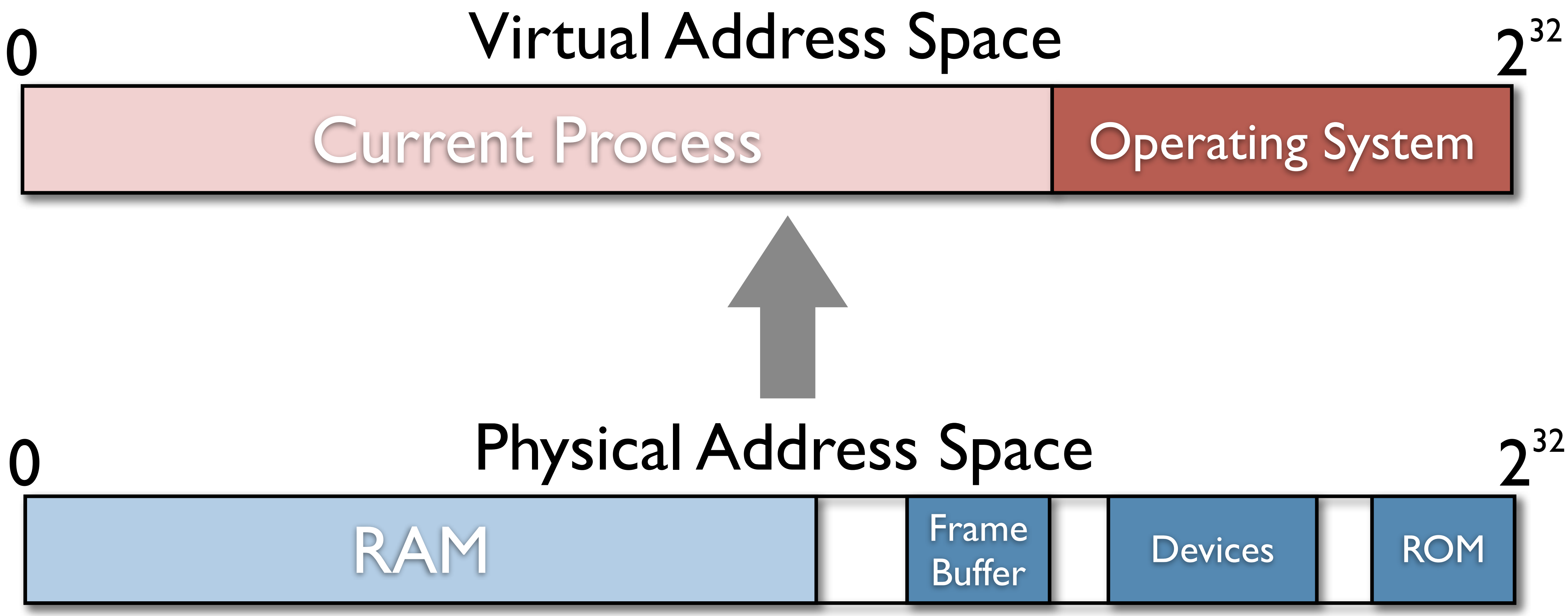
Memory Virtualization



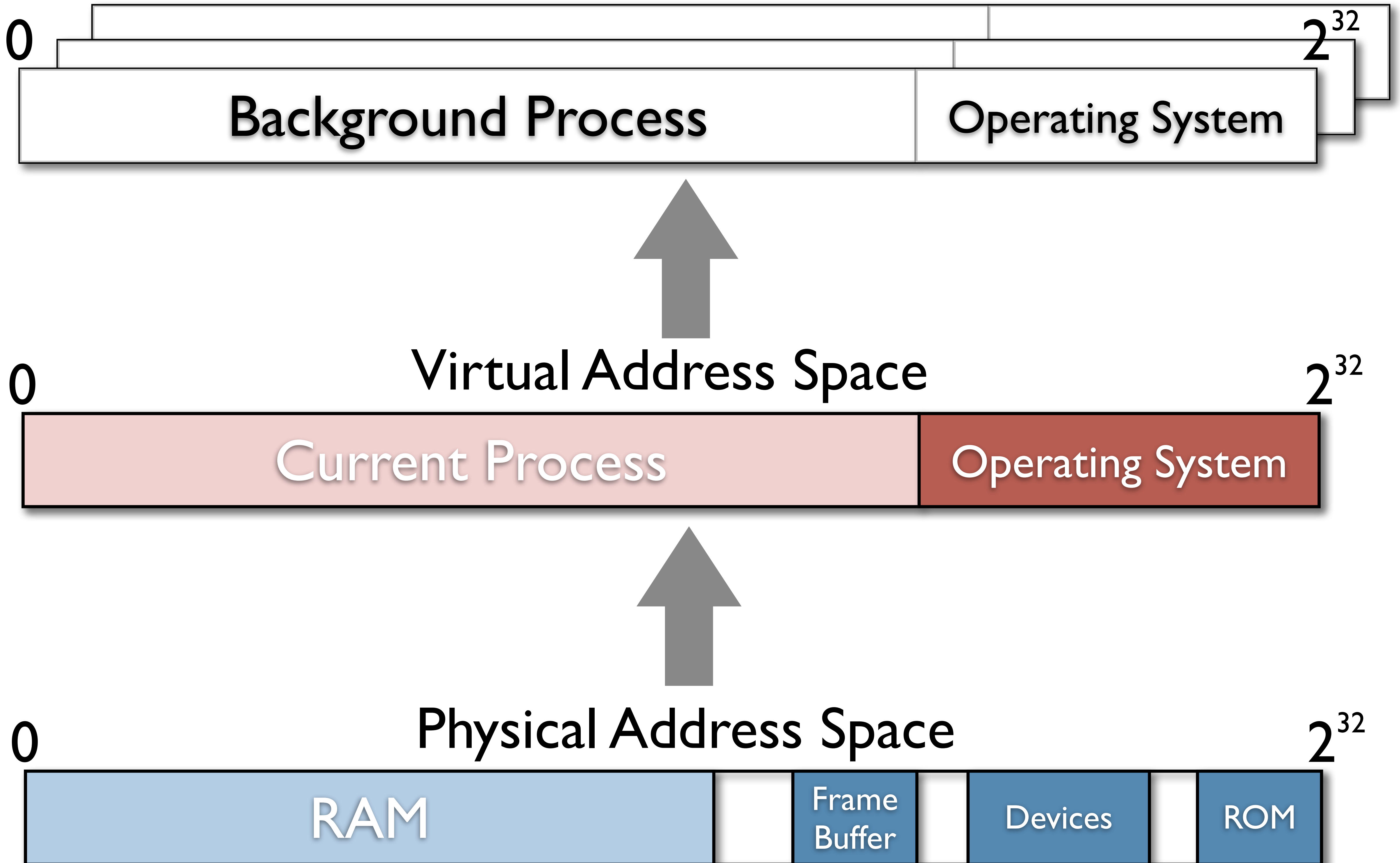
Virtual Memory



Virtual Memory

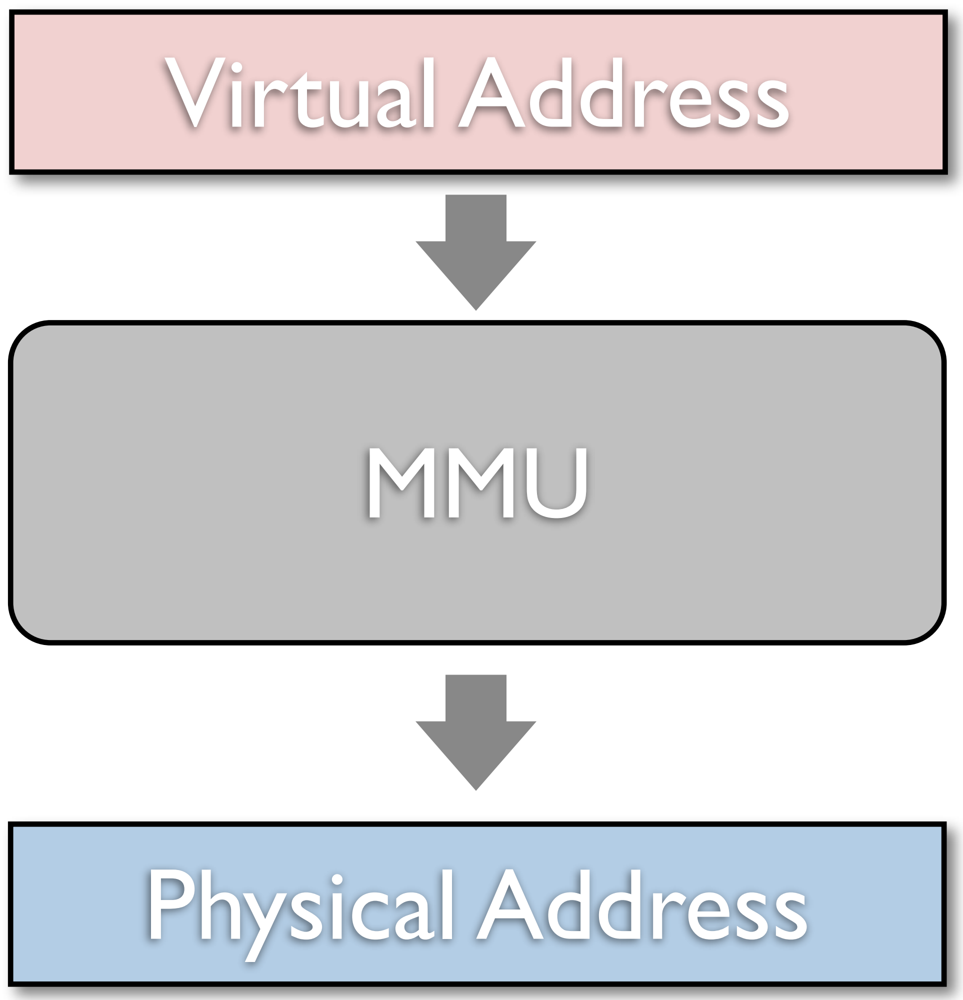


Virtual Memory



Memory Management Unit

- The MMU implements the virtual address space
  - Handles accesses to memory requested by CPU
  - It is an hardware component
  - Uses data structures in memory

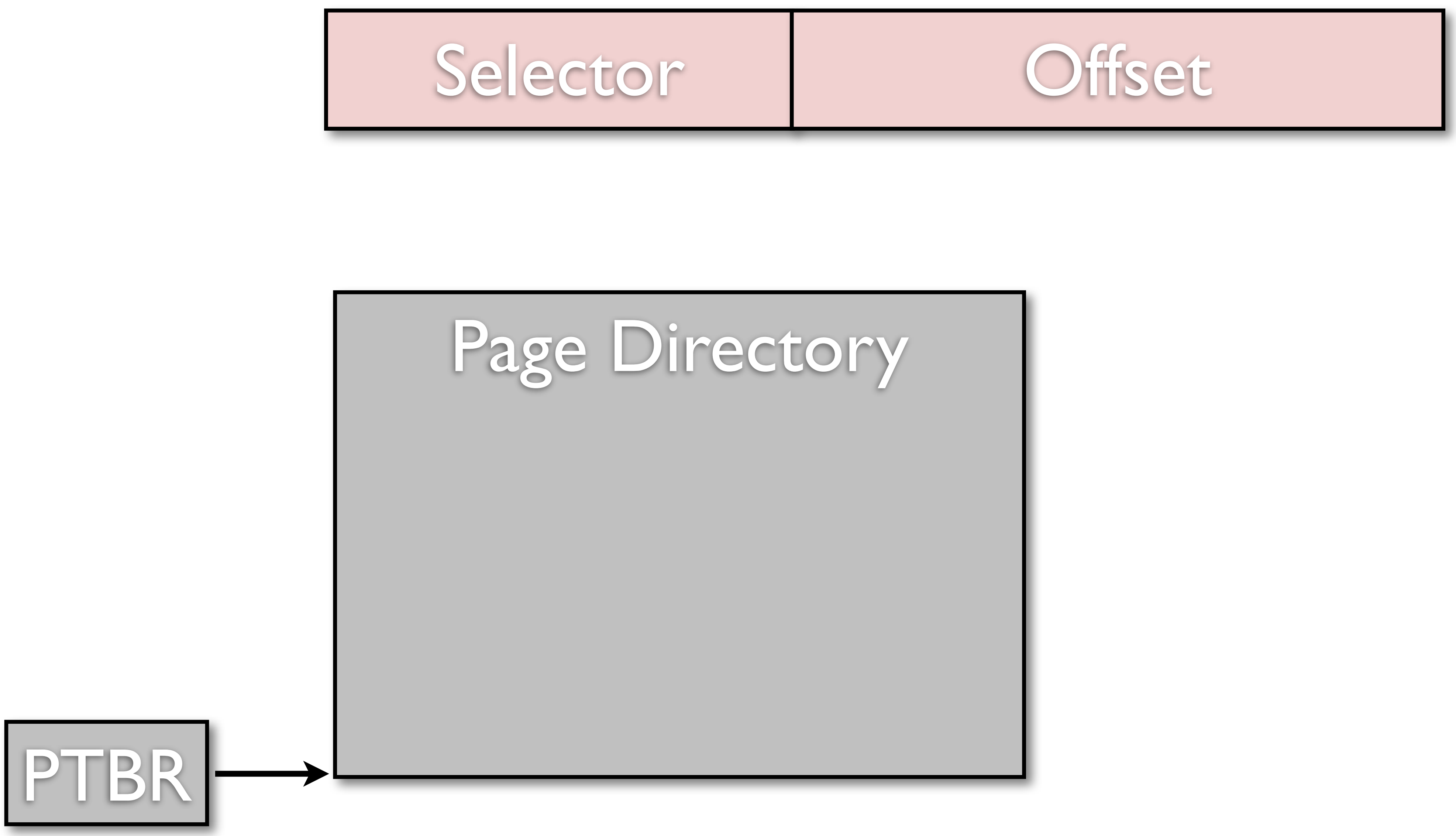


Simple Address Translation

Virtual Address

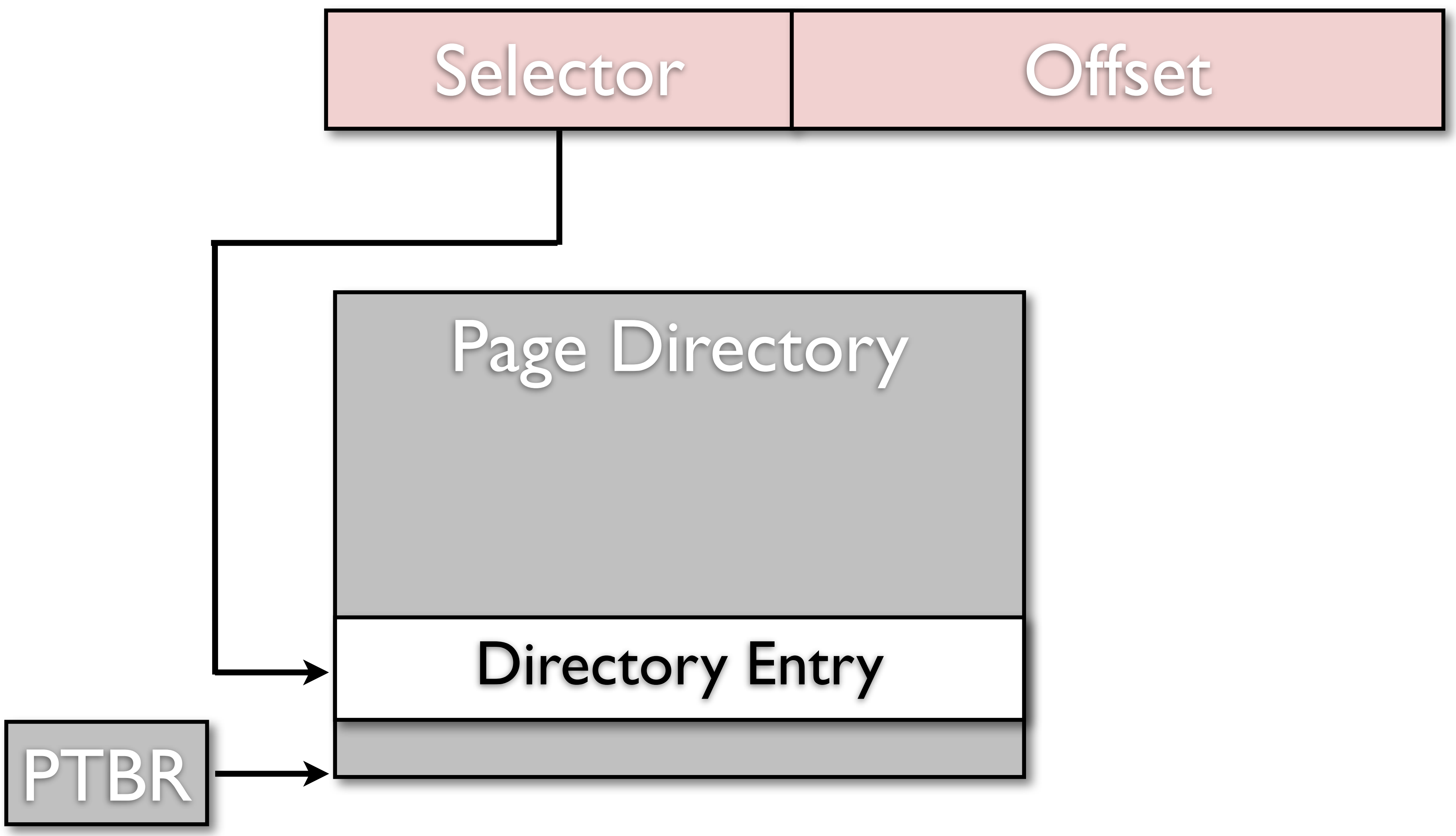
Simple Address Translation

Selector	Offset

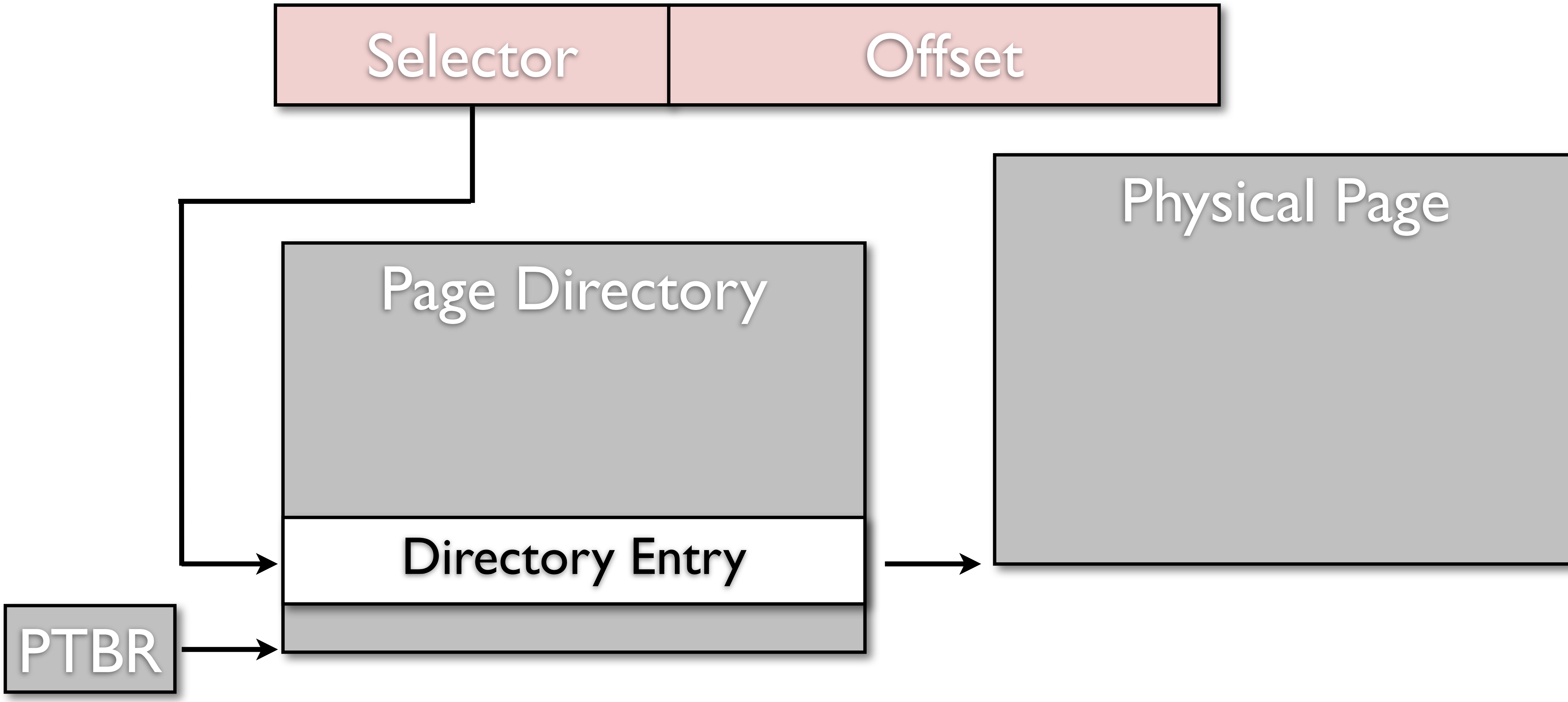


Selector
Offset
Page Directory
PTBR

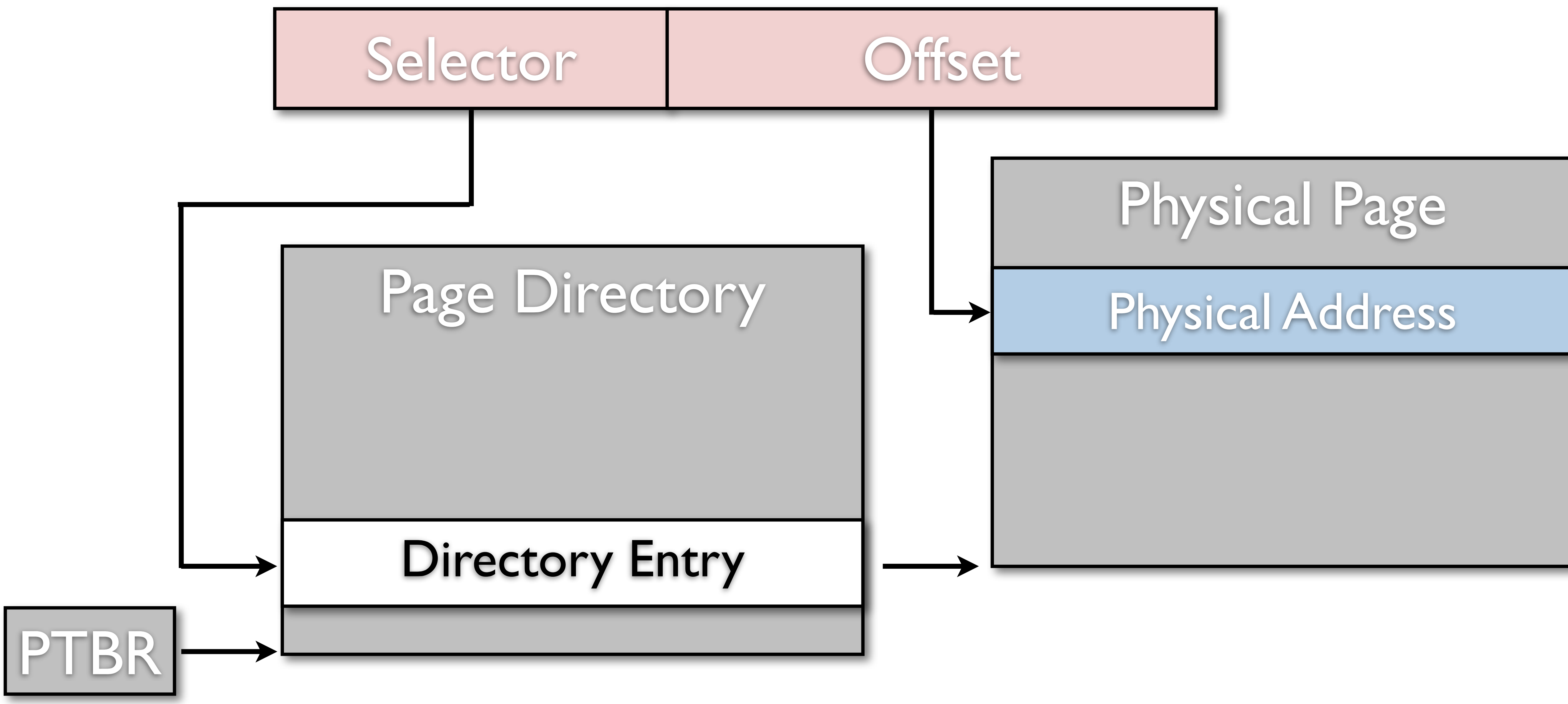
Simple Address Translation



Simple Address Translation



Simple Address Translation



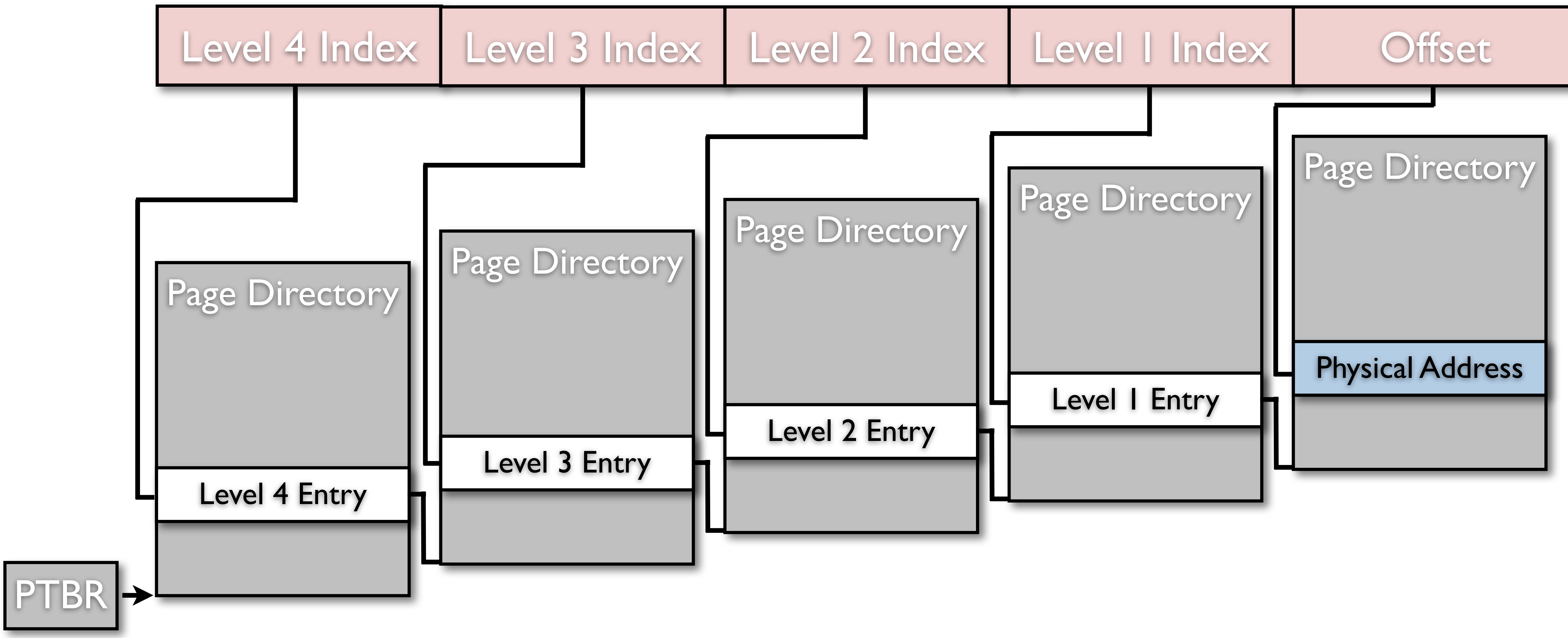
Some notes...

- The complete physical address of the memory cell is determined by combining the page address from the page directory with the lower bits from the virtual address
- More than one entry in the page directory can point to the same physical address
- The page directory entry also contains some additional info about the page
  - Access permission, etc..
- The data structure for the page directory is in main memory
  - The OS has to allocate contiguous physical memory and store the base address of this memory region in a special CPU register
  - The OS individually sets each entry in the page directory
- Layout Example (X86)
  - Address space: 32 bit
  - Page size: 4MB, i.e., 22 bits to address every byte (offset)
  - Page directory size: 1024 entries, i.e., 10 bits to address every entry (directory)

... and problems

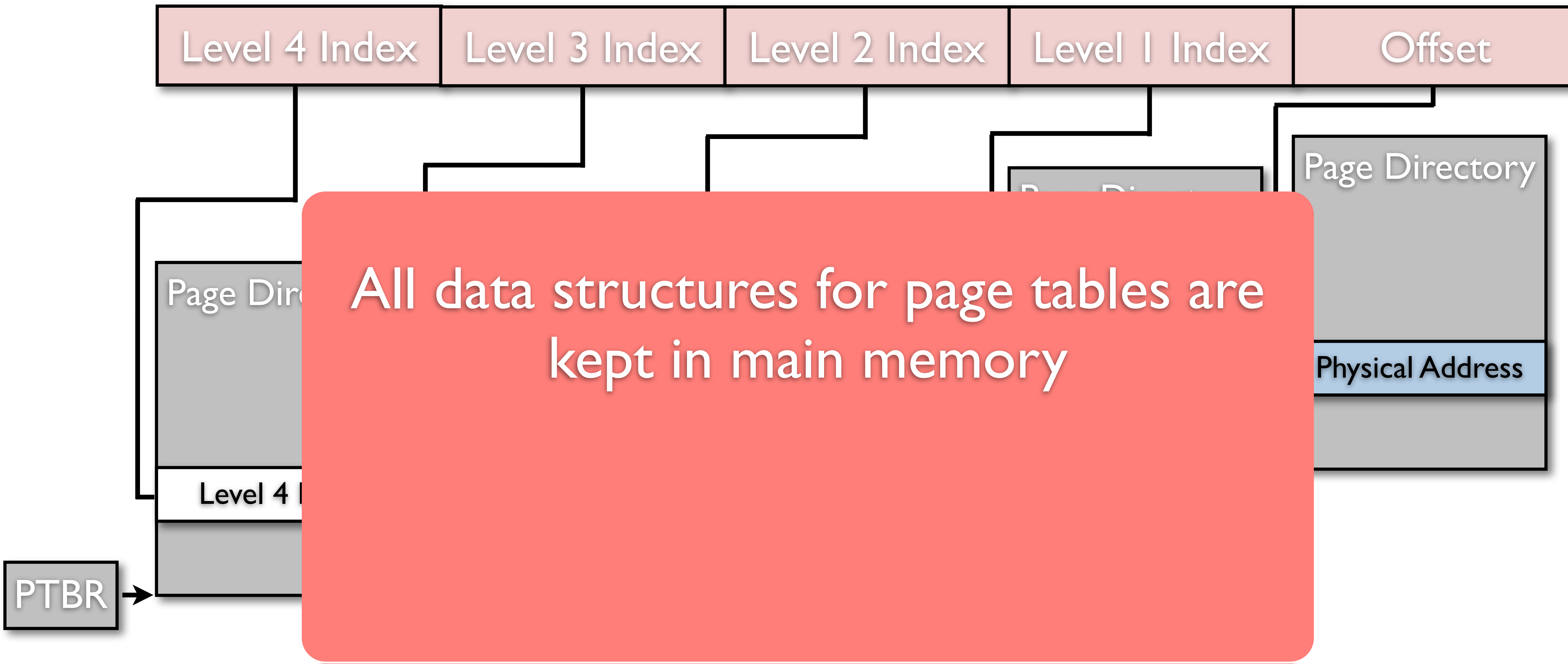
- Typical page size is 4KB, no 4MB
  - Selector 20 bits, Offset 12 bits
- Page table with 1,048,576 entries
  - If page entry is 4 bytes, page size is 4MB
- Each process needs a page table
  - 256 processes will occupy 1GB of memory just for page tables
- The solution is to use a huge, sparse page directory
  - Address space regions which are not actually used do not require allocated memory

Multi Level Page Tables



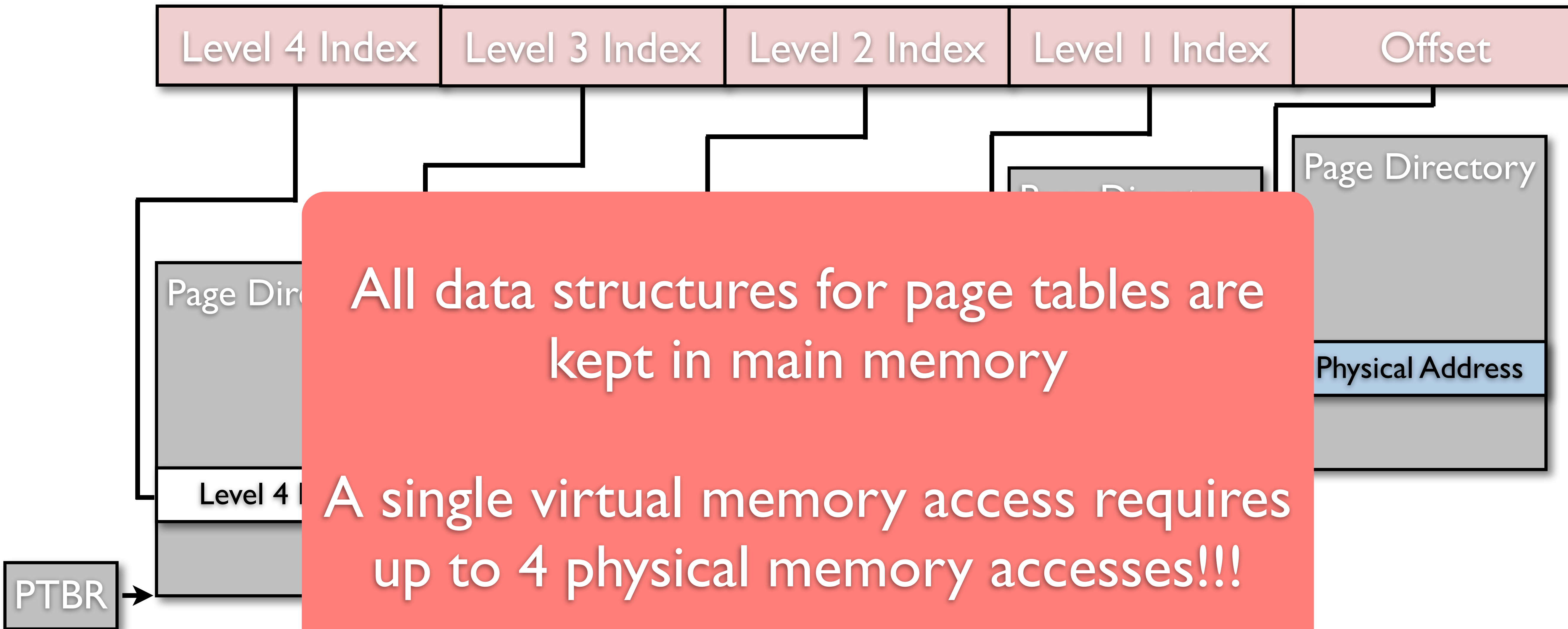
- The process of determining the physical address is called page tree walking
  - Some processors do it in hardware, others need help from the OS
- A small program might get by with using just one directory at each of levels 2, 3 and 4 and a few level 1 directories.
- 1GB of memory can be addressed with one directory for levels 2 to 4 and 512 directories for level 1

Multi Level Page Tables



- The process of determining the physical address is called page tree walking
  - Some processors do it in hardware, others need help from the OS
- A small program might get by with using just one directory at each of levels 2, 3 and 4 and a few level 1 directories.
- 1GB of memory can be addressed with one directory for levels 2 to 4 and 512 directories for level 1

Multi Level Page Tables

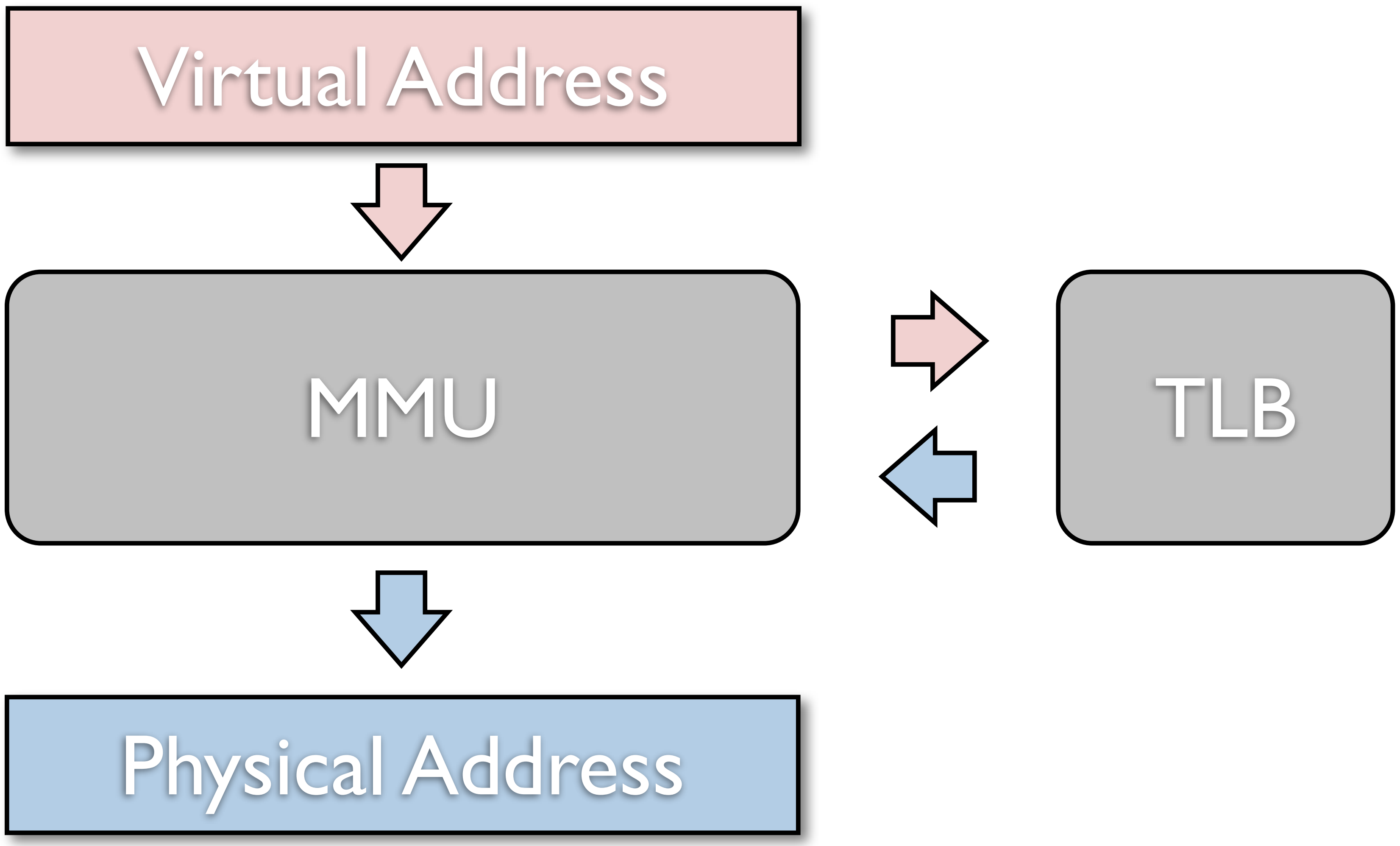


All data structures for page tables are kept in main memory

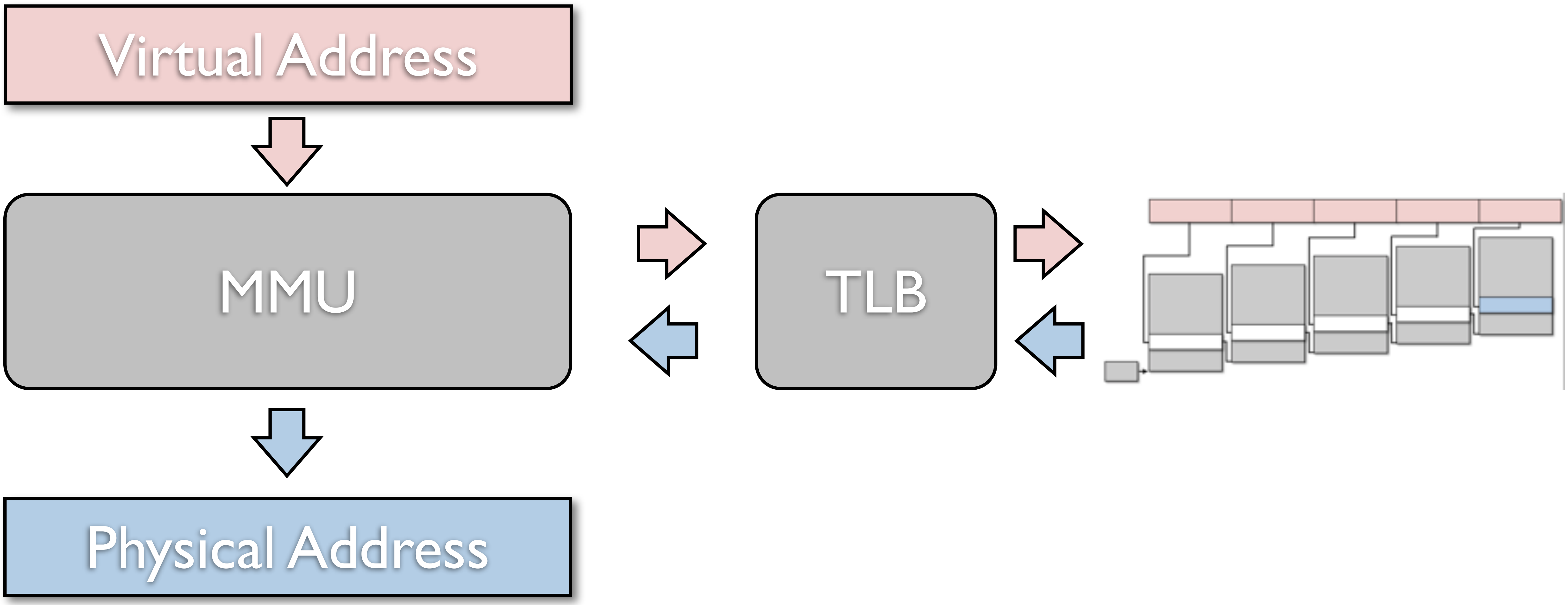
A single virtual memory access requires up to 4 physical memory accesses!!!

- The process of determining the physical address is called page tree walking
  - Some processors do it in hardware, others need help from the OS
- A small program might get by with using just one directory at each of levels 2, 3 and 4 and a few level 1 directories.
- 1GB of memory can be addressed with one directory for levels 2 to 4 and 512 directories for level 1

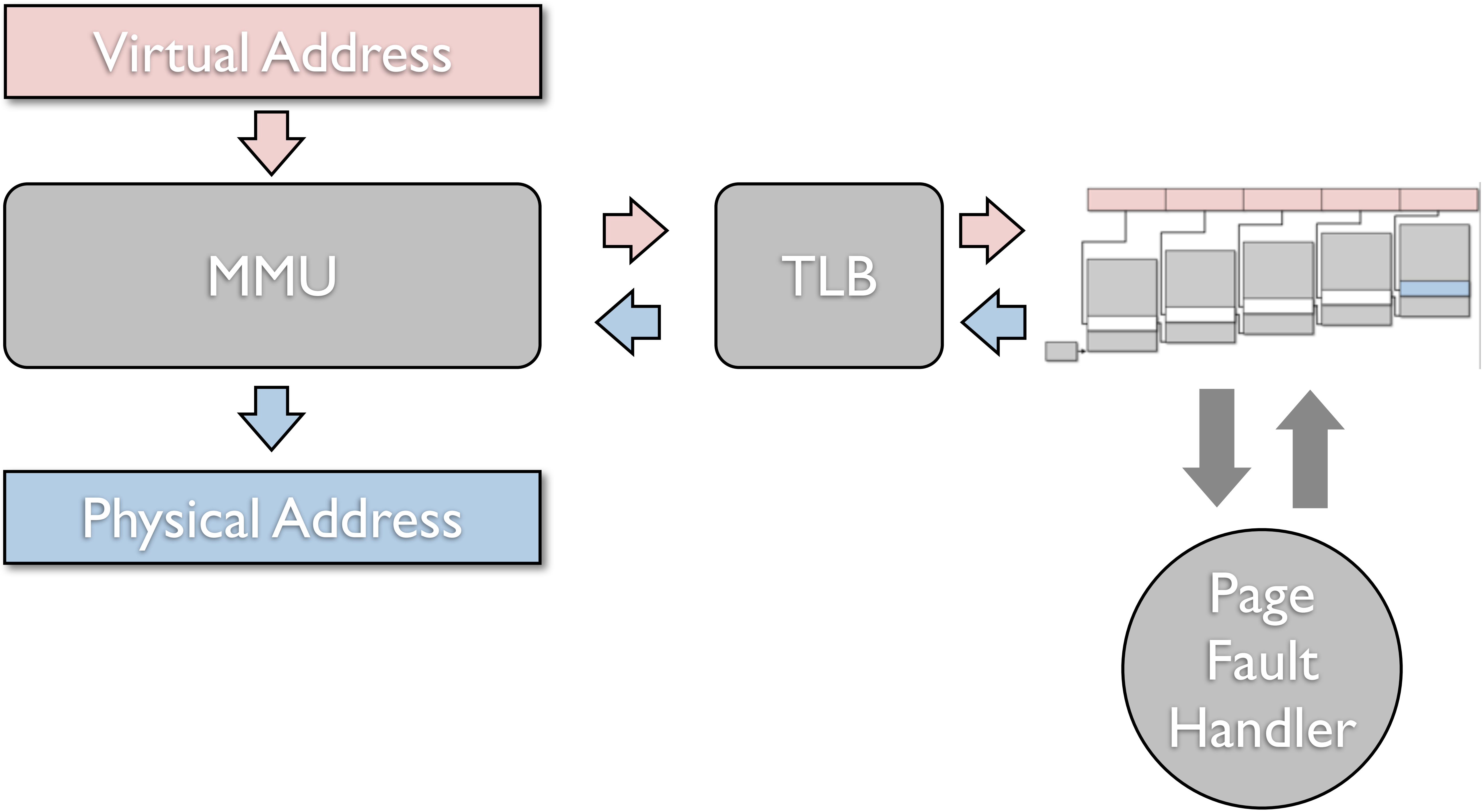
Translation Lookaside Buffer



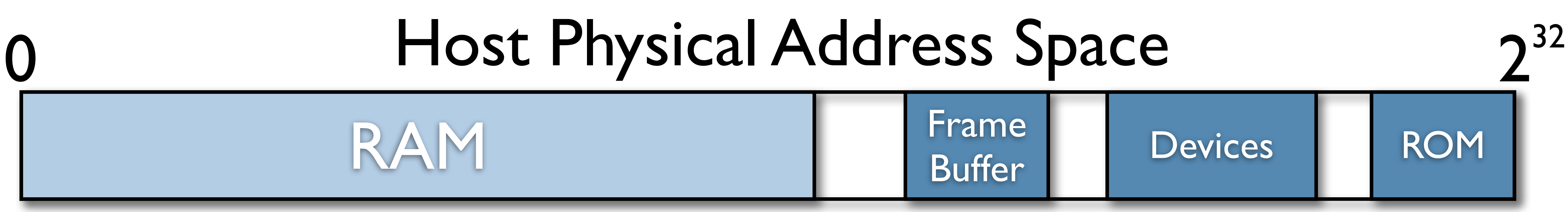
Translation Lookaside Buffer



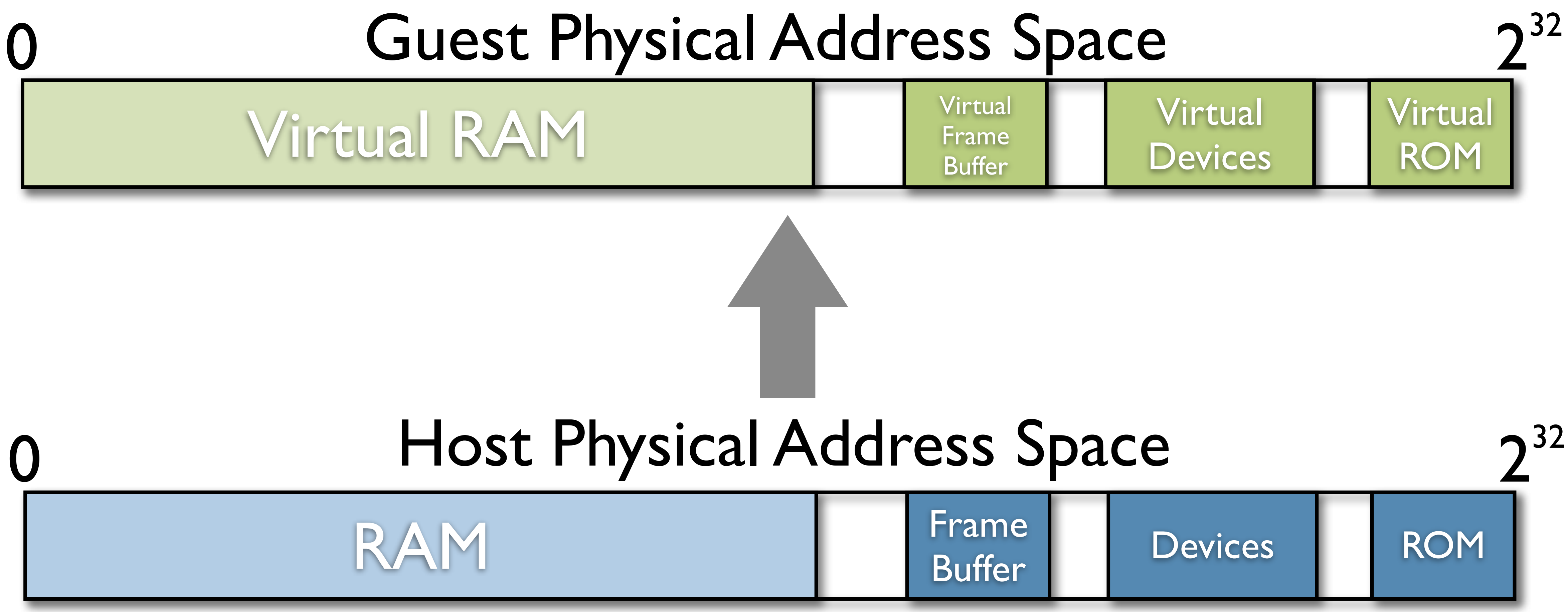
Translation Lookaside Buffer



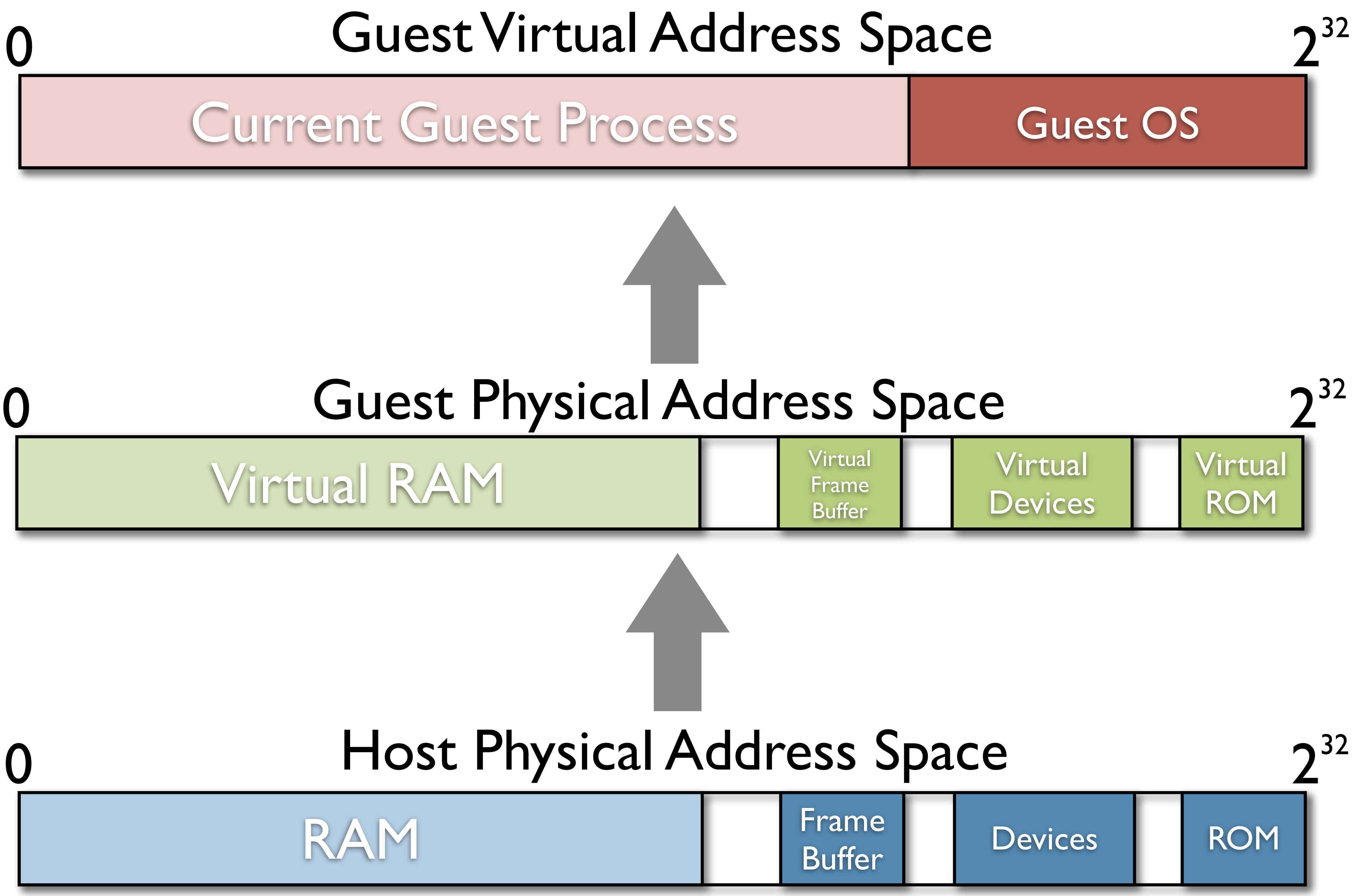
Virtualizing Virtual Memory



Virtualizing Virtual Memory



Virtualizing Virtual Memory



Shadow Page Table

Shadow Page Table

- The VMM must map guest virtual address to host physical address

Shadow Page Table

- The VMM must map guest virtual address to host physical address
- Guest OS maintains its own virtual memory page table in the guest physical memory

Shadow Page Table

- The VMM must map guest virtual address to host physical address
- Guest OS maintains its own virtual memory page table in the guest physical memory
- The VMM maintains a mapping from each guest physical memory page to the host physical memory page
  - PMAP data structure

Shadow Page Table

- The VMM must map guest virtual address to host physical address
- Guest OS maintains its own virtual memory page table in the guest physical memory
- The VMM maintains a mapping from each guest physical memory page to the host physical memory page
  - PMAP data structure
- The VMM seems able to intercept MMU hardware requests to translate GVAs
  - Monitoring PTBR
  - Two memory accesses, to guest virtual memory page table and PMAP

Shadow Page Table

- The VMM must map guest virtual address to host physical address
- Guest OS maintains its own virtual memory page table in the guest physical memory
- The VMM maintains a mapping from each guest physical memory page to the host physical memory page
  - PMAP data structure
- The VMM seems able to intercept MMU hardware requests to translate GVAs
  - Monitoring PTBR
  - Two memory accesses, to guest virtual memory page table and PMAP
- So what the hell is a shadow page table?

Shadow Page Table

- The VMM must map guest virtual address to host physical address
- Guest OS maintains its own virtual memory page table in the guest physical memory
- The VMM m … emory page to the host
  - PMAP data
- The VMM s … s to translate G
  - Monitoring
  - Two memory accesses, to guest virtual memory page table and PMAP
- So what the hell is a shadow page table?

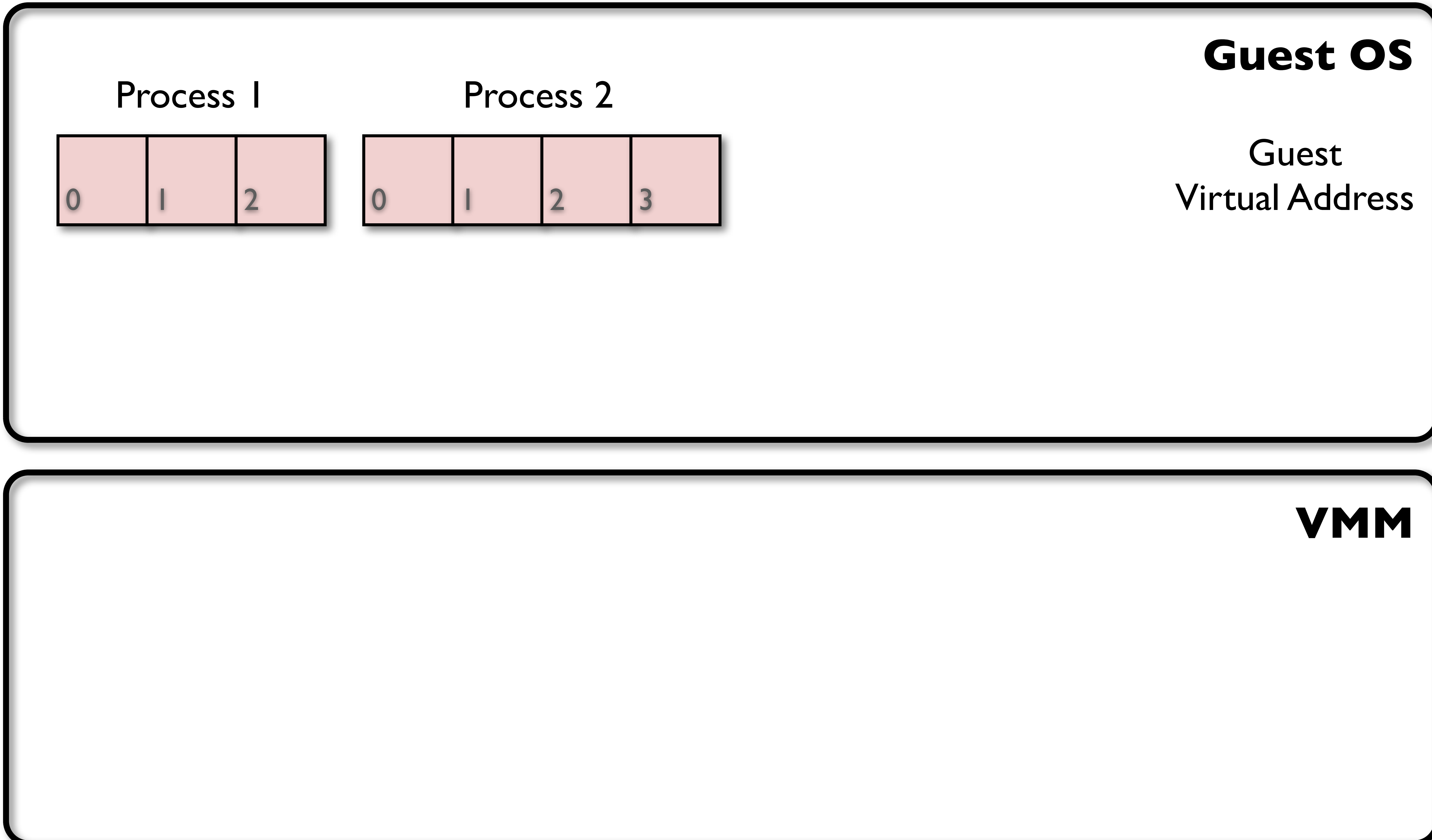
What about the TLB?

Shadow Page Table

- The VMM must intercept all VM instructions that manipulate:
  - The hardware TLB contents
  - Guest OS page table
- The actual hardware TLB is updated based on the separate shadow page tables
  - They contain the guest virtual to host physical address mapping
- The VMM must protect the host frames containing the guest page tables!

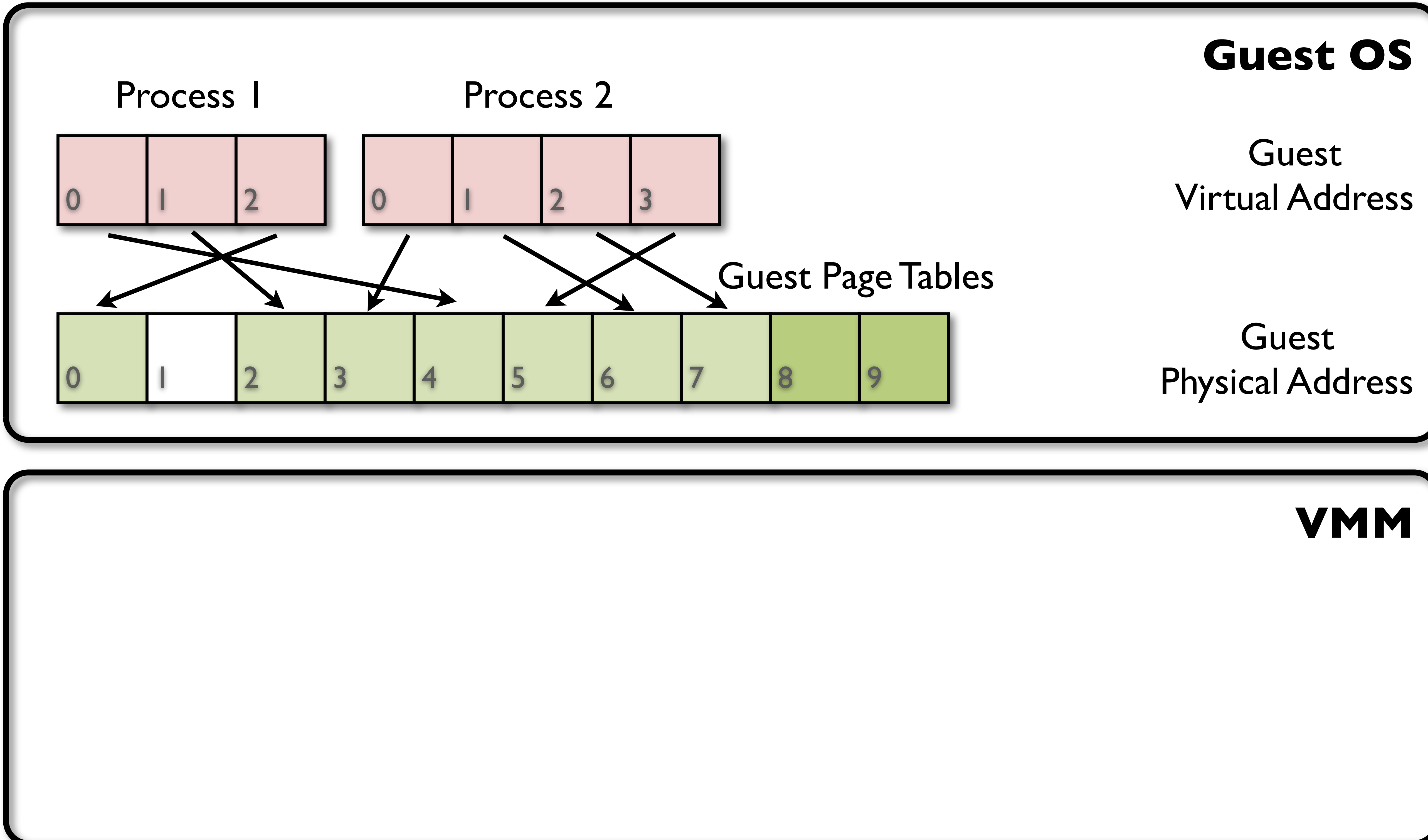
Example

Example



ISTITUTO DI SCIENZA E TECNOLOGIE
DELL'INFORMAZIONE "A. FAEDO"

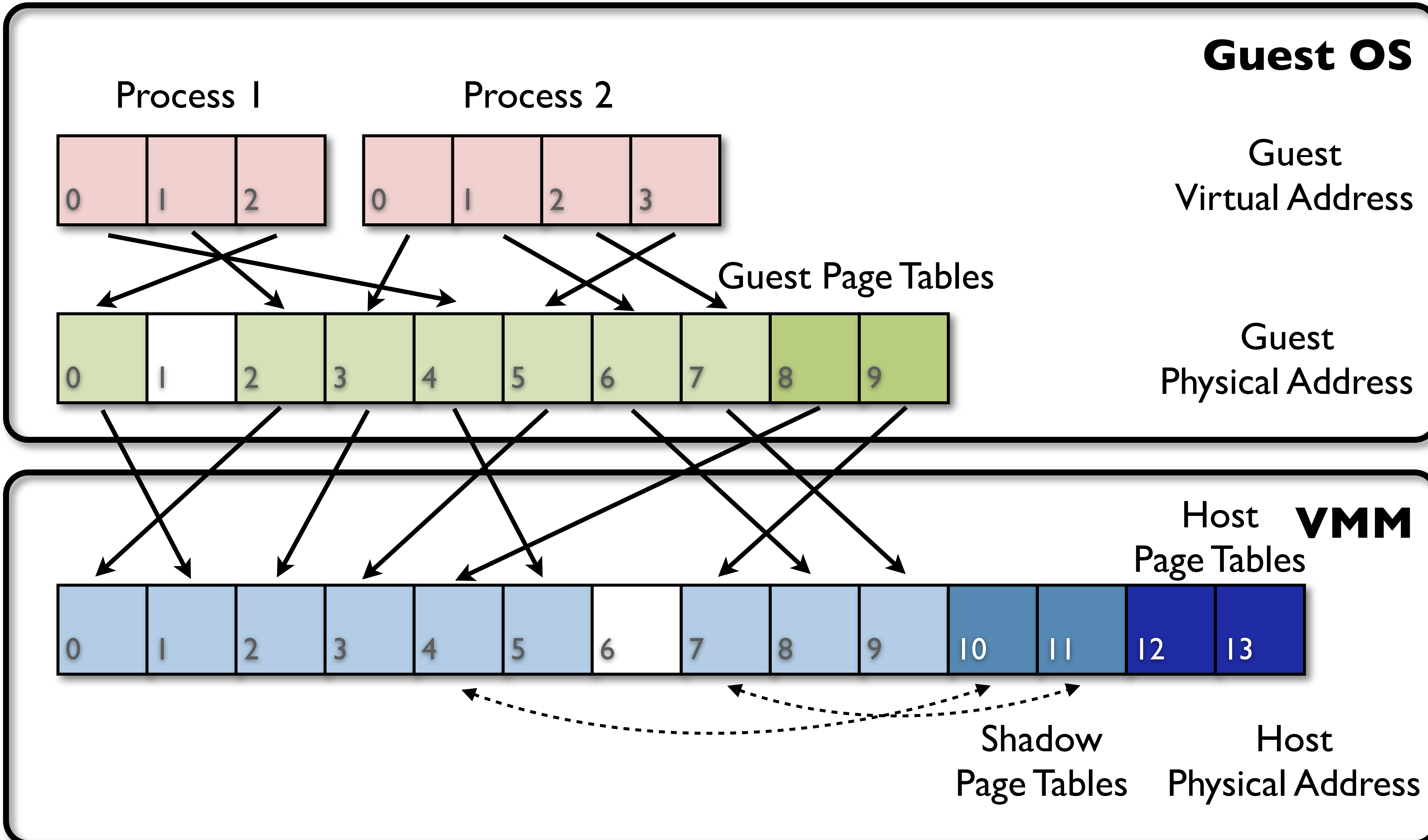
Example



ISTITUTO DI SCIENZA E TECNOLOGIE DELL'INFORMAZIONE "A. FAEDO"

lunedì 17 dicembre 12

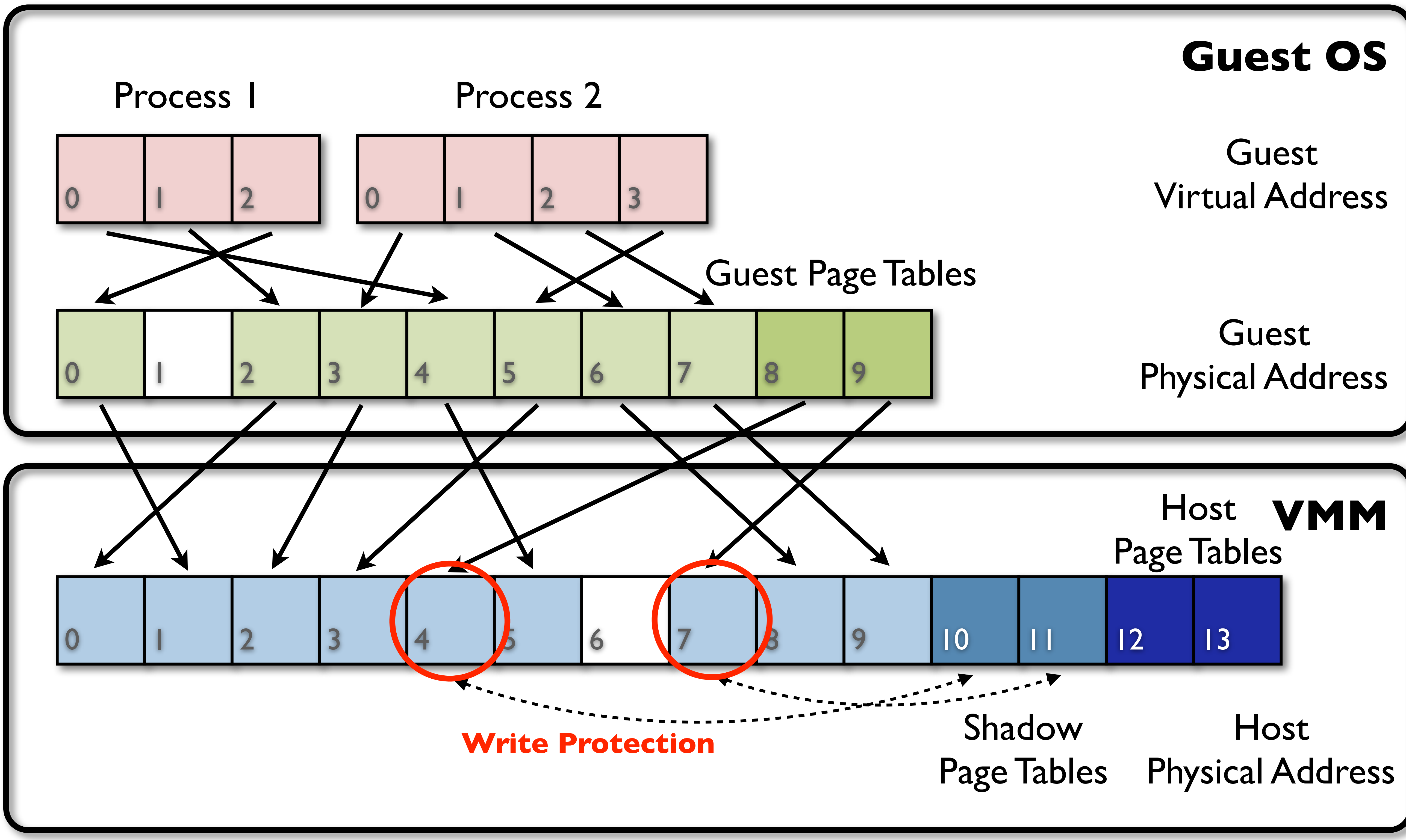
Example



ISTITUTO DI SCIENZA E TECNOLOGIE DELL'INFORMAZIONE "A. FAEDO"

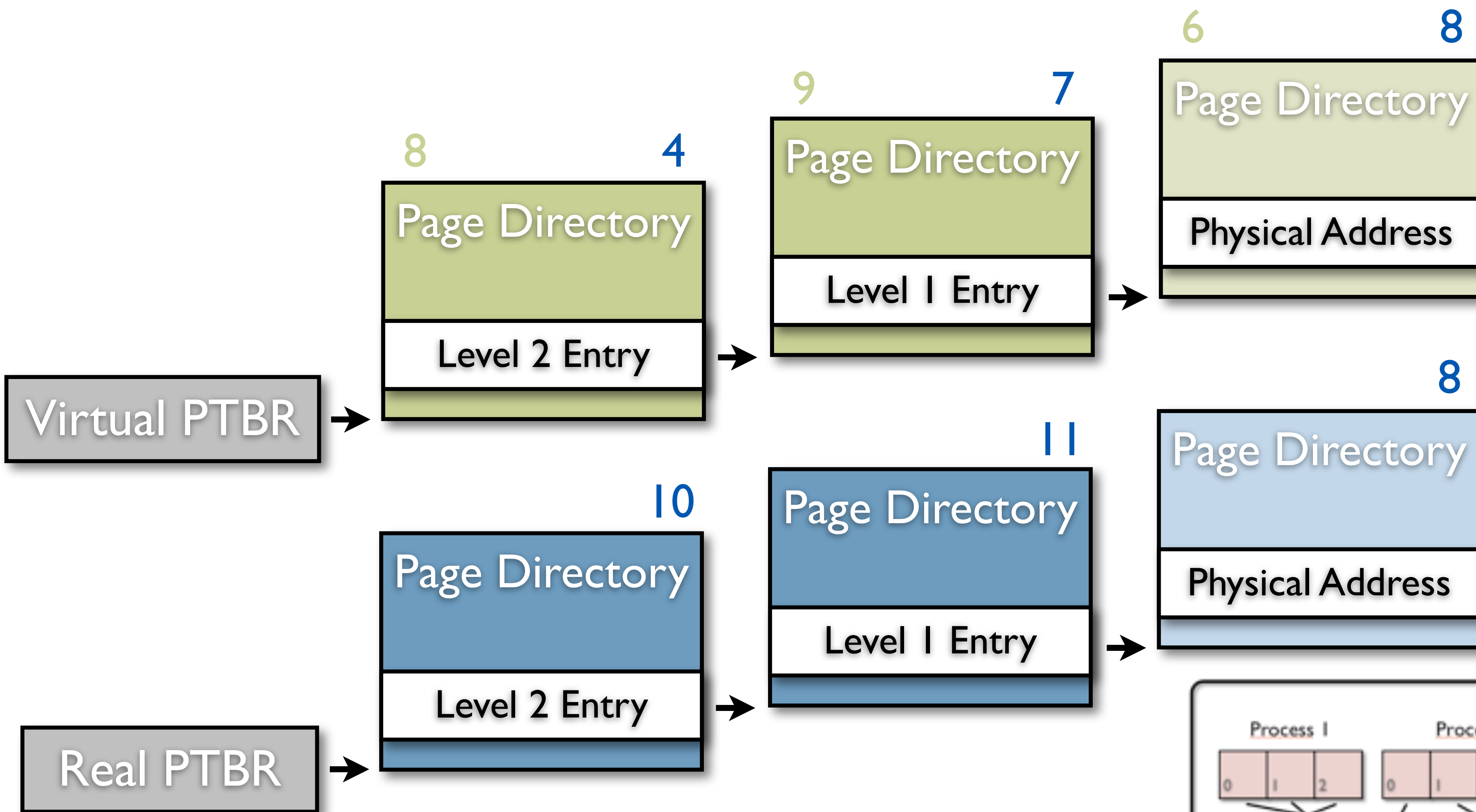
lunedì 17 dicembre 12

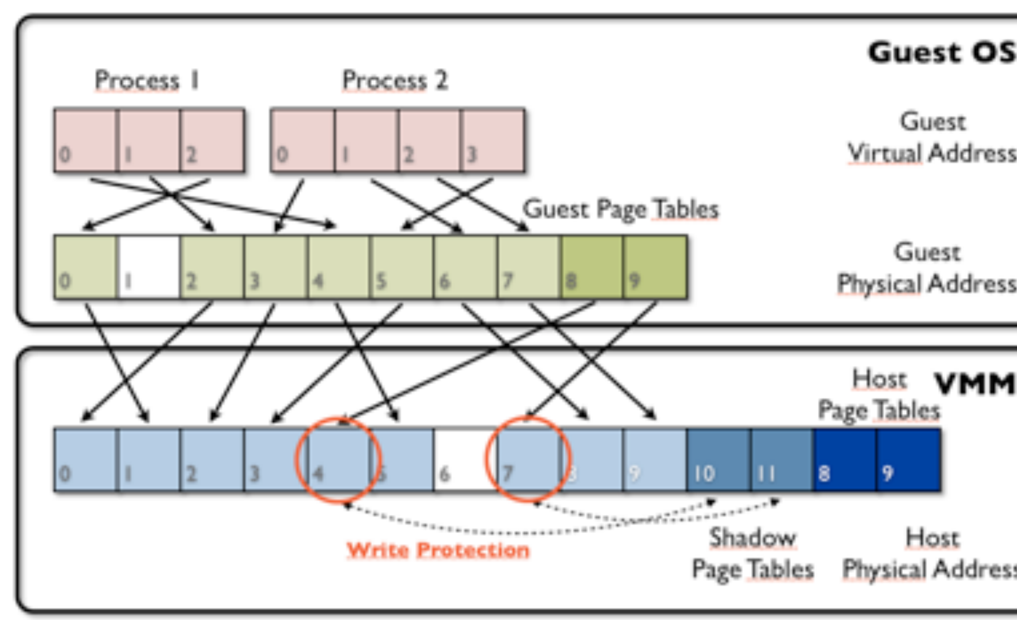
Example


Guest OS
Process 1
Process 2
0 1 2
0 1 2 3
Guest Virtual Address
Guest Page Tables
0 1 2 3 4 5 6 7 8 9
Guest Physical Address
Host Page Tables
VMM
0 1 2 3 4 5 6 7 8 9 10 11 12 13
Write Protection
Shadow Page Tables
Host Physical Address

Example

Process 2 in Guest OS want to access its memory whose page number is 1





MMU Events

The MMU is driven by events

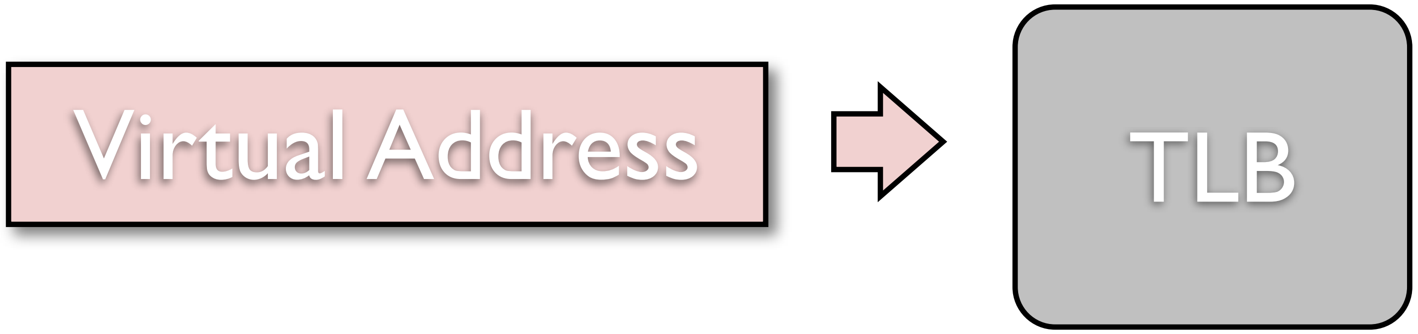
- generated from guest
  - write instructions to control registers (in particular PTBR)
  - page invalidation instructions (in case of page faults)
  - access to missing or protected entries

- generated from host
  - Changes in PMAP translation (GPA > HPA)
    - GPA > HVA changes
    - HVA > HPA changes
  - Memory pressure

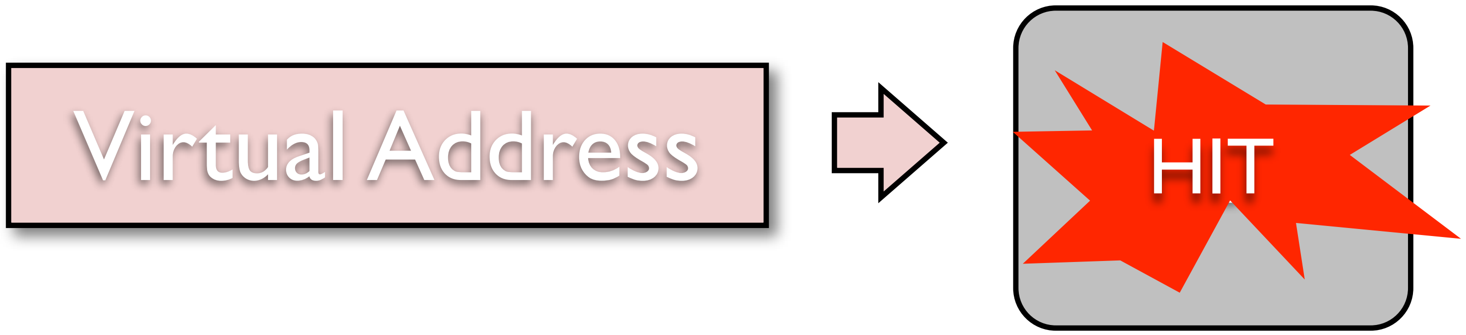
Shadow Page Tables with TLB

Virtual Address

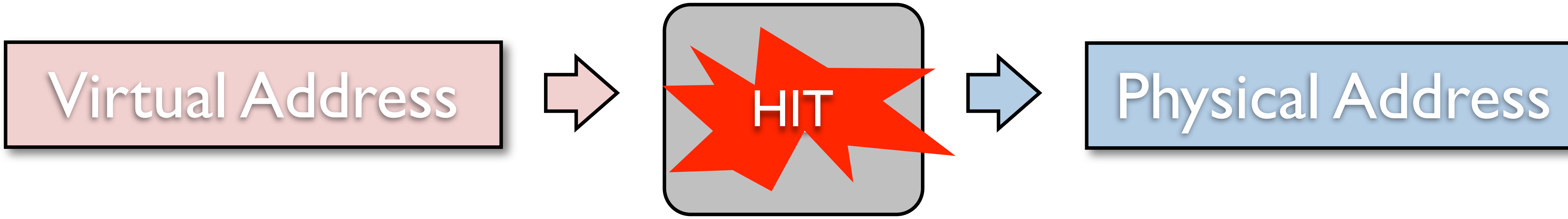
Shadow Page Tables with TLB



Shadow Page Tables with TLB



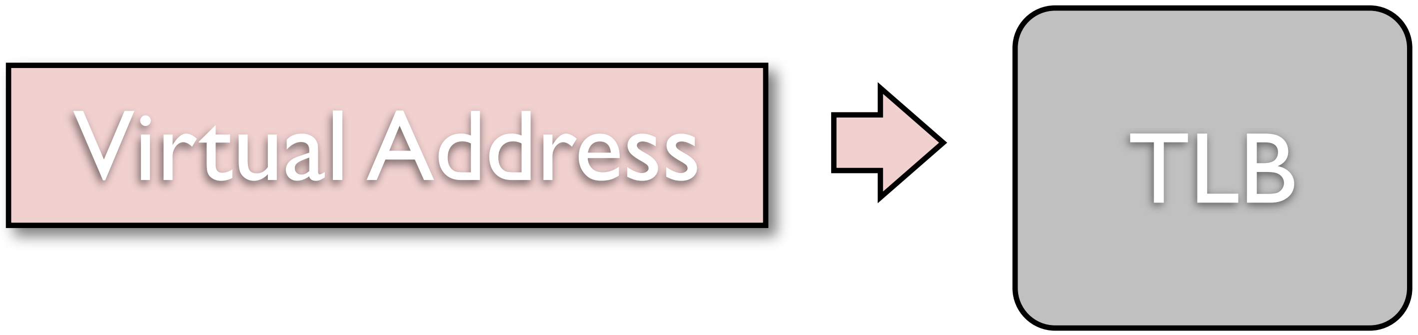
Shadow Page Tables with TLB



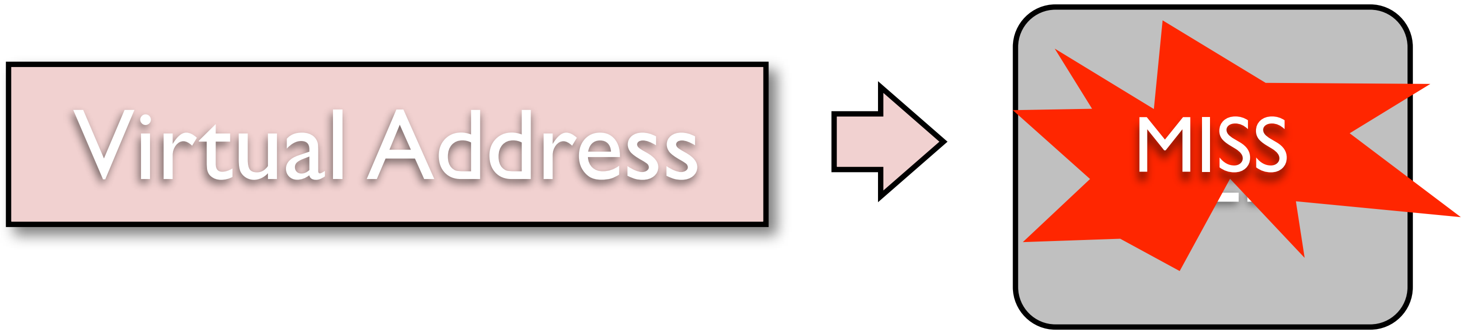
Shadow Page Tables with TLB

Virtual Address

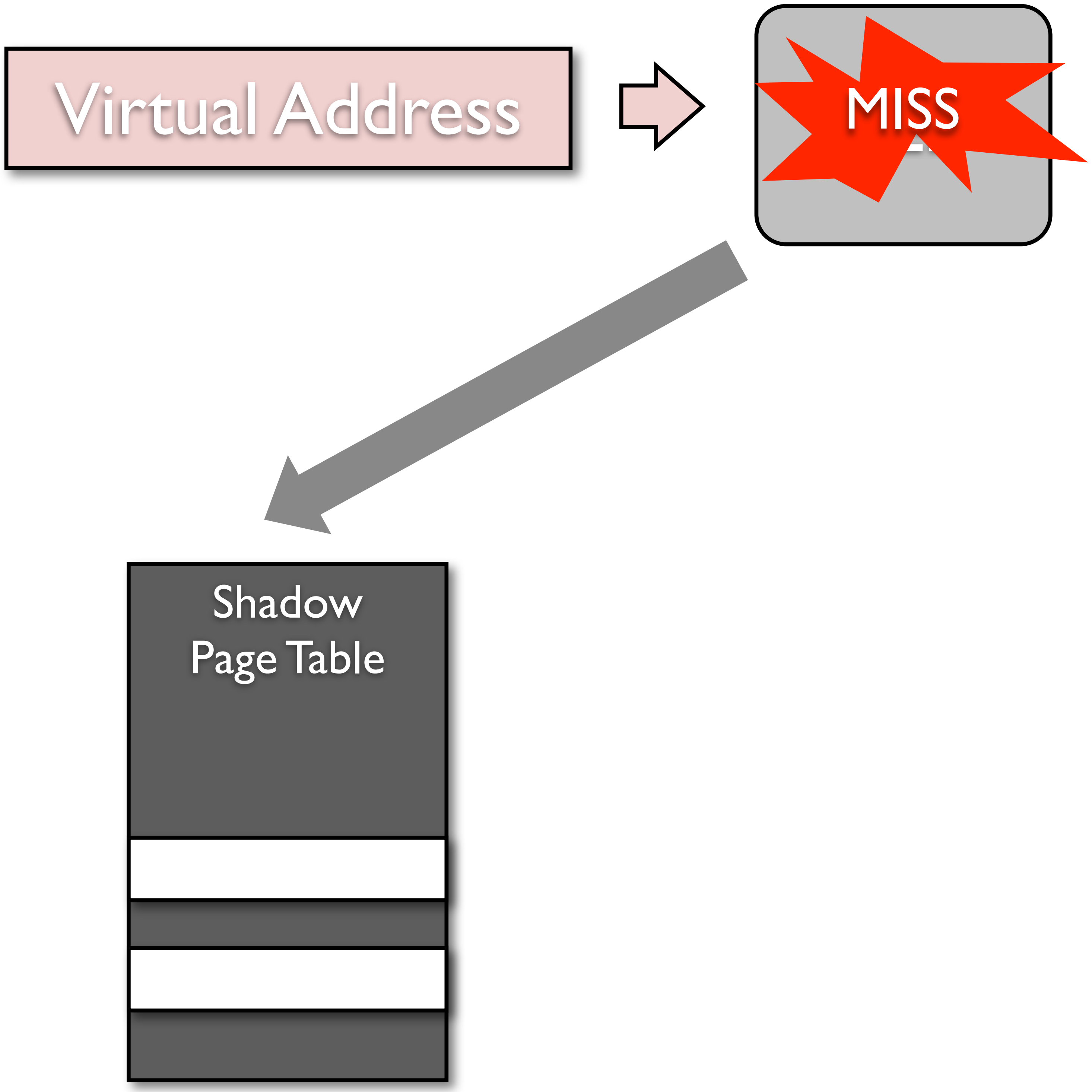
Shadow Page Tables with TLB



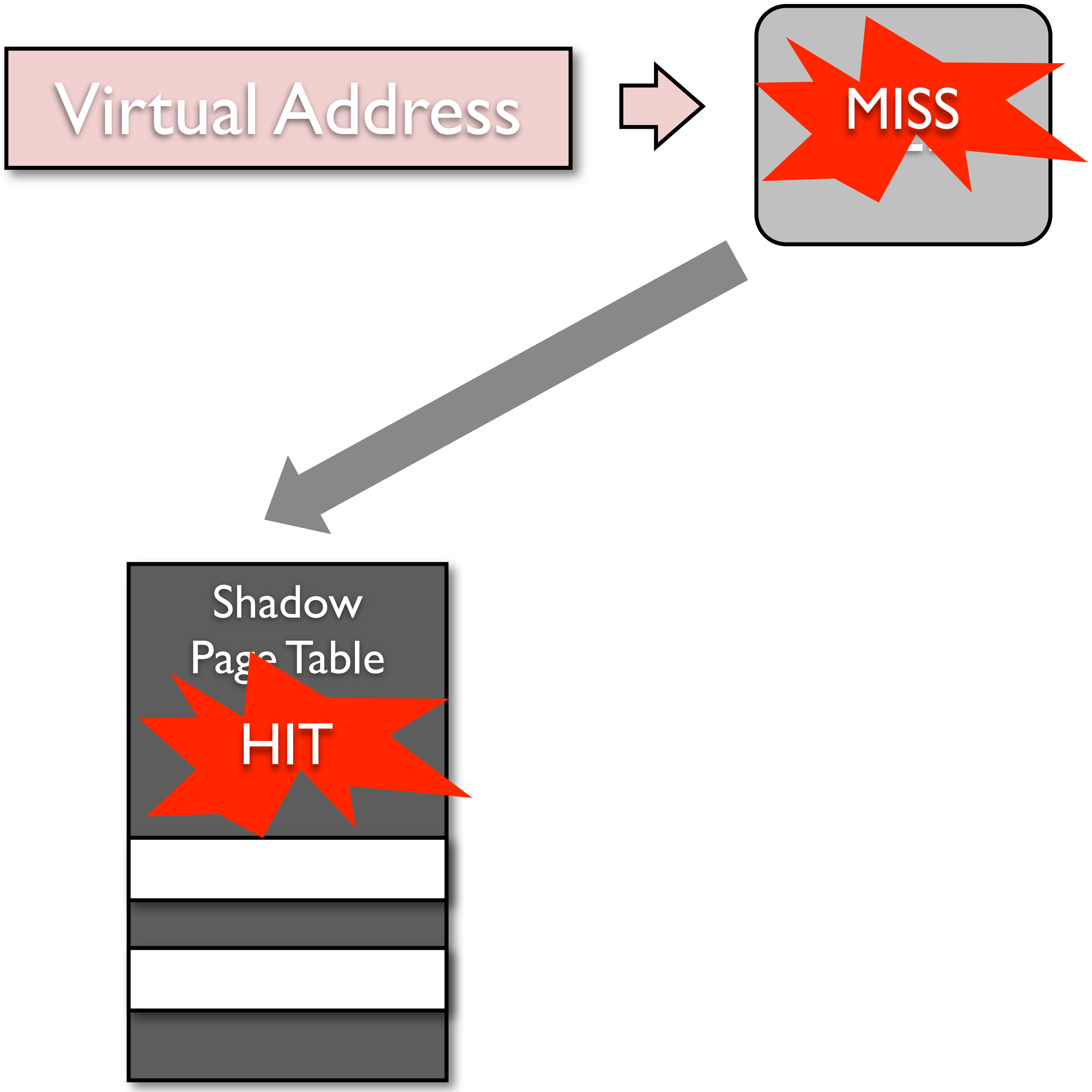
Shadow Page Tables with TLB



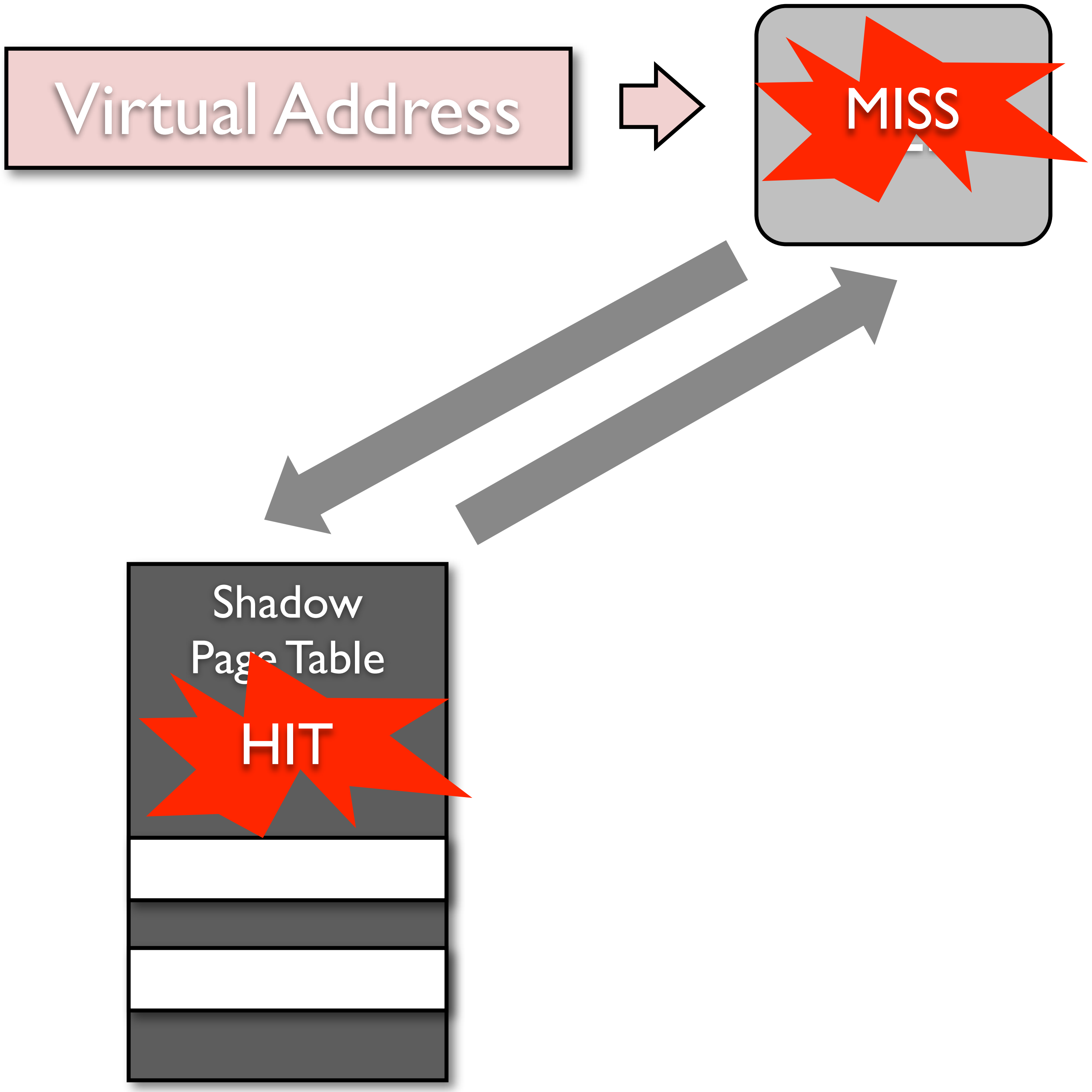
Shadow Page Tables with TLB



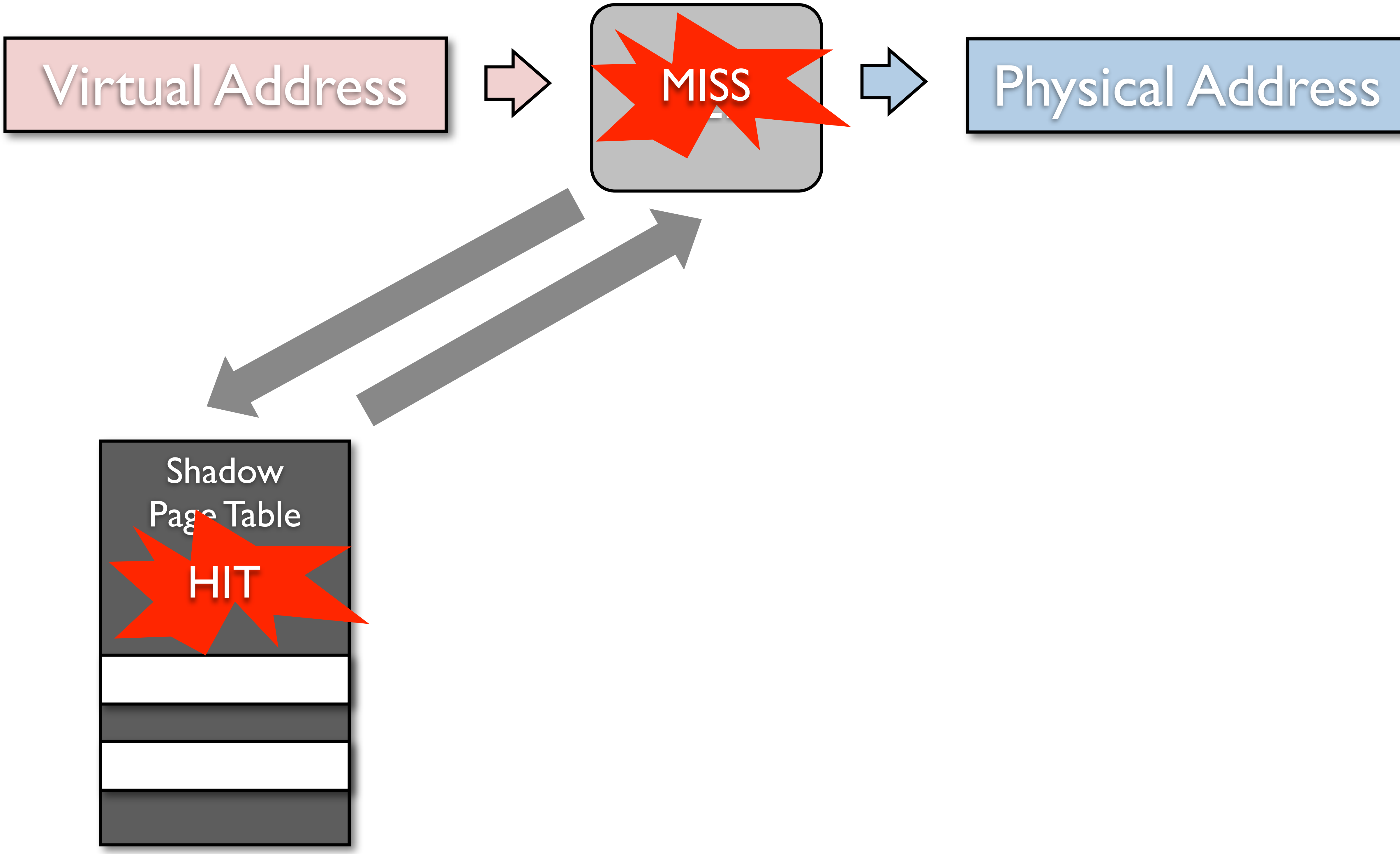
Shadow Page Tables with TLB



Shadow Page Tables with TLB



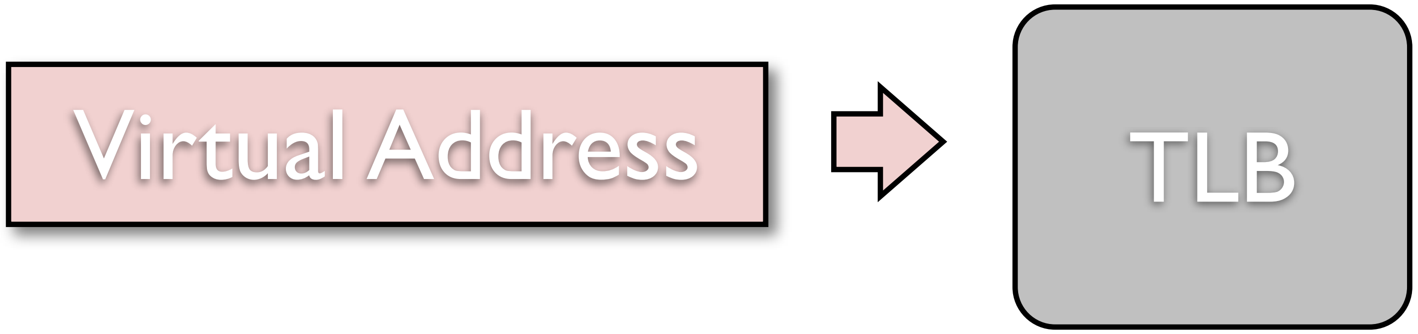
Shadow Page Tables with TLB



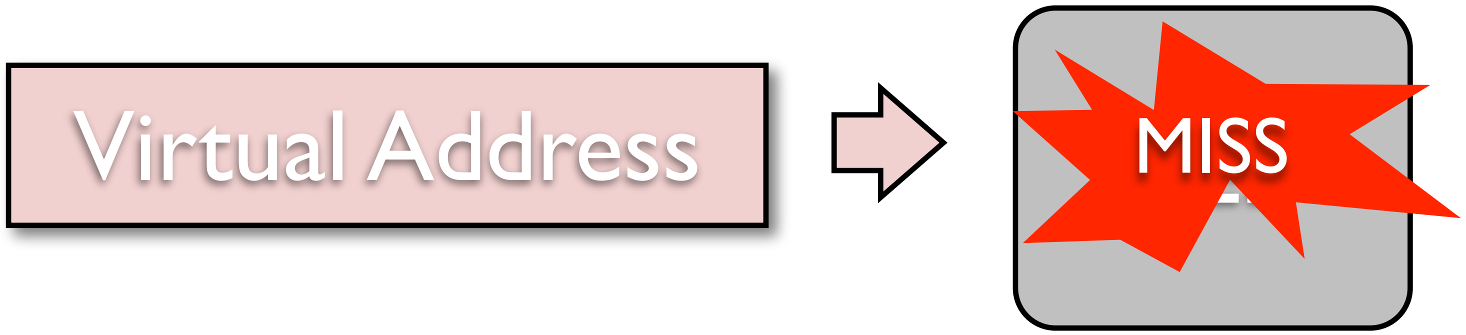
Shadow Page Tables with TLB

Virtual Address

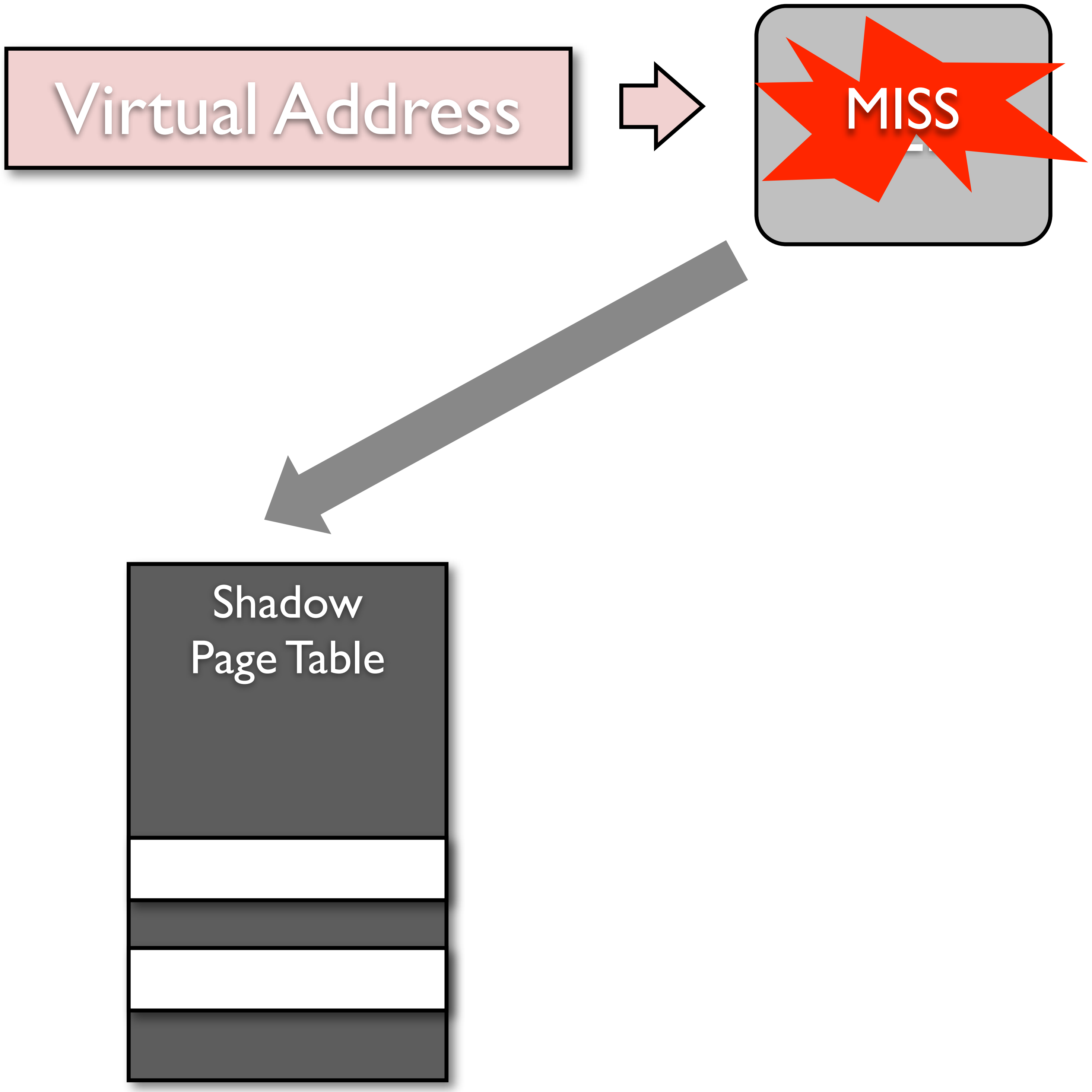
Shadow Page Tables with TLB

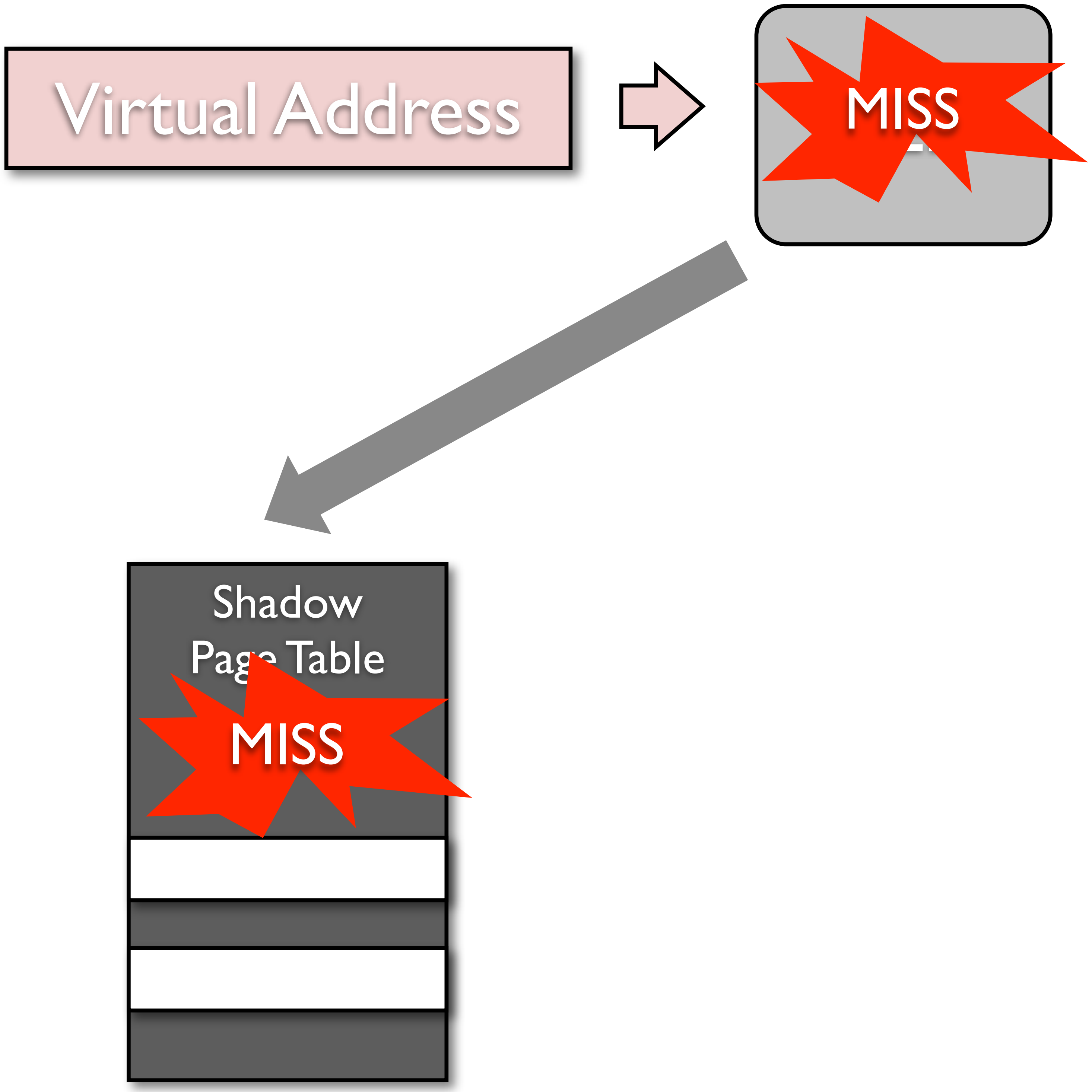


Shadow Page Tables with TLB



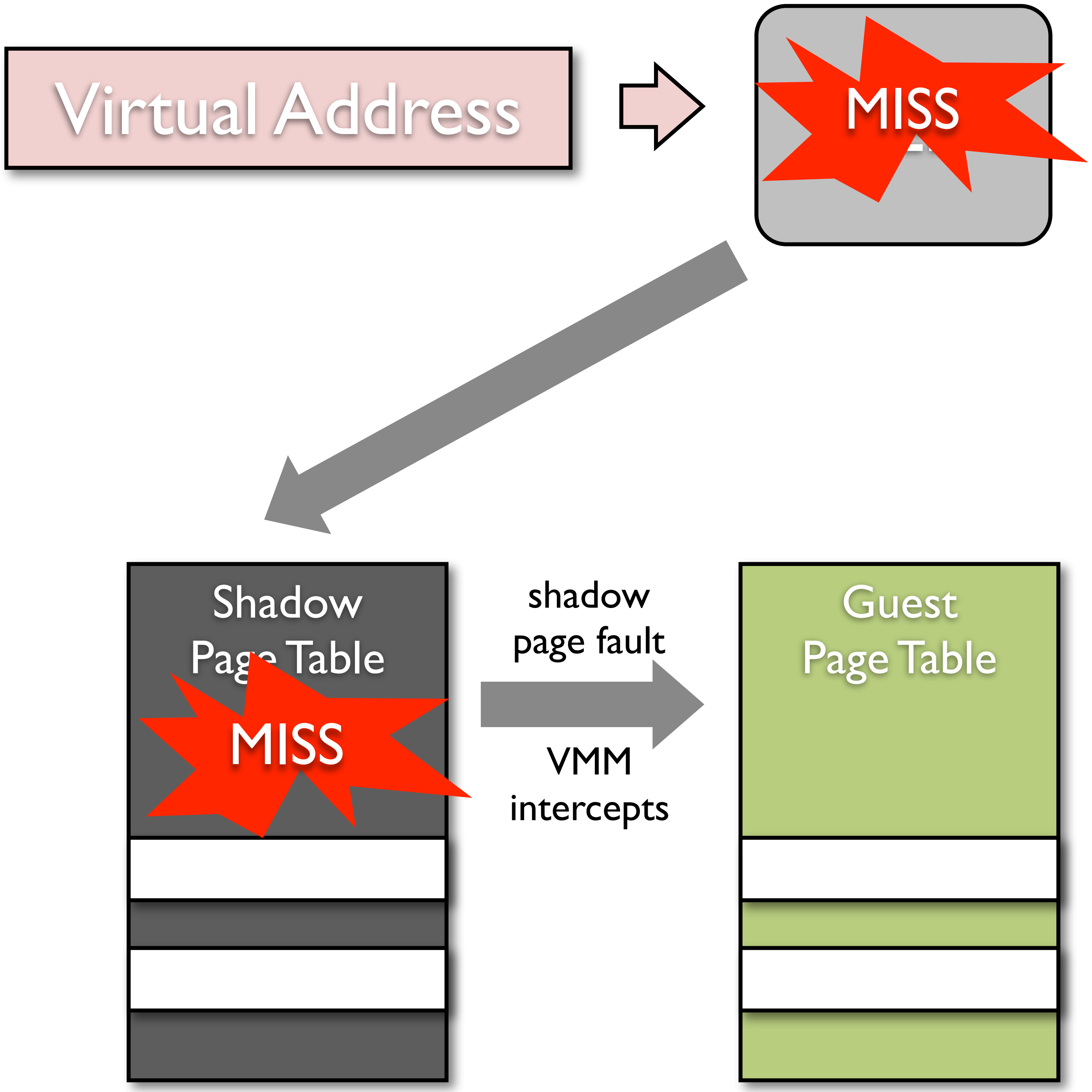
Shadow Page Tables with TLB



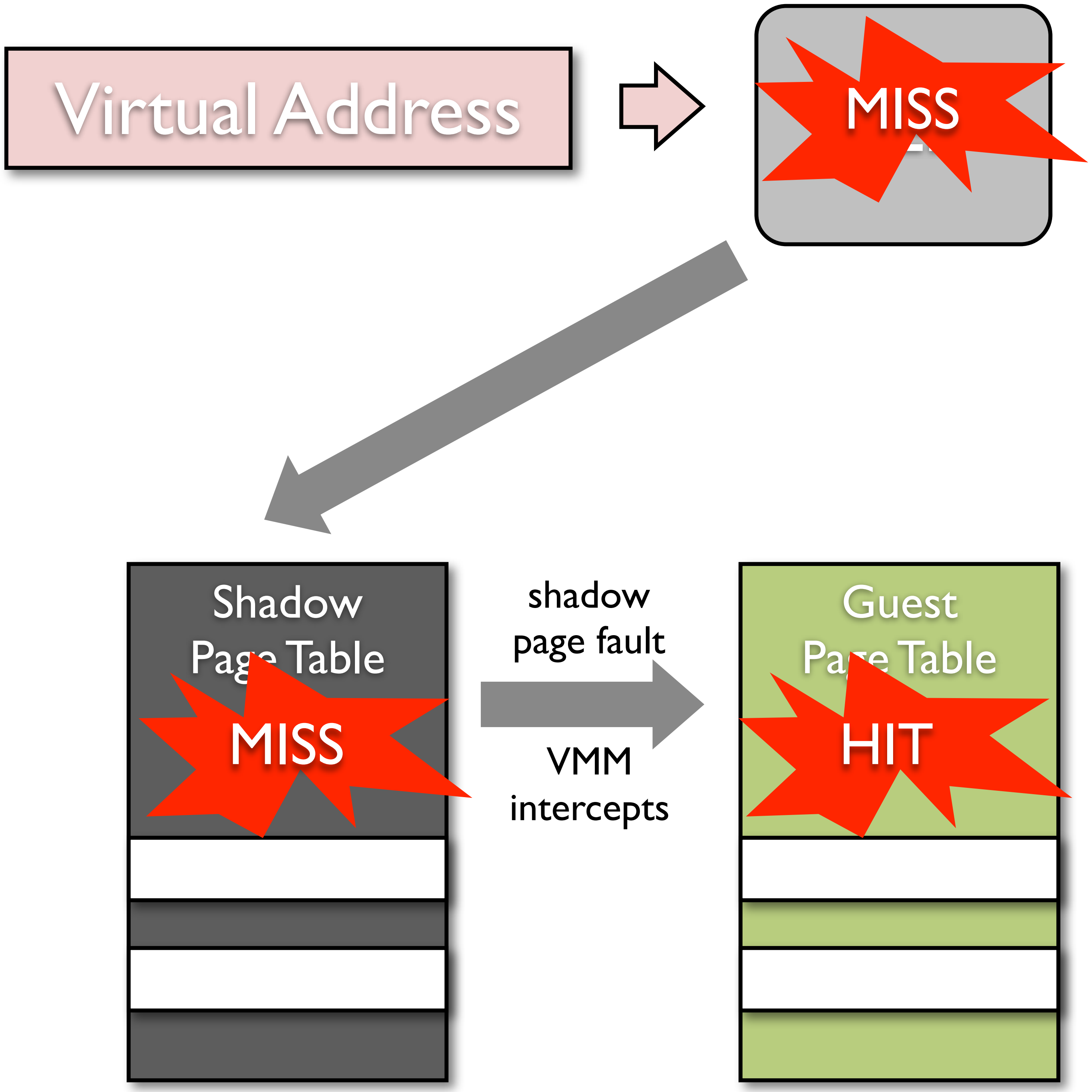


Virtual Address
MISS
Shadow
Page Table
MISS

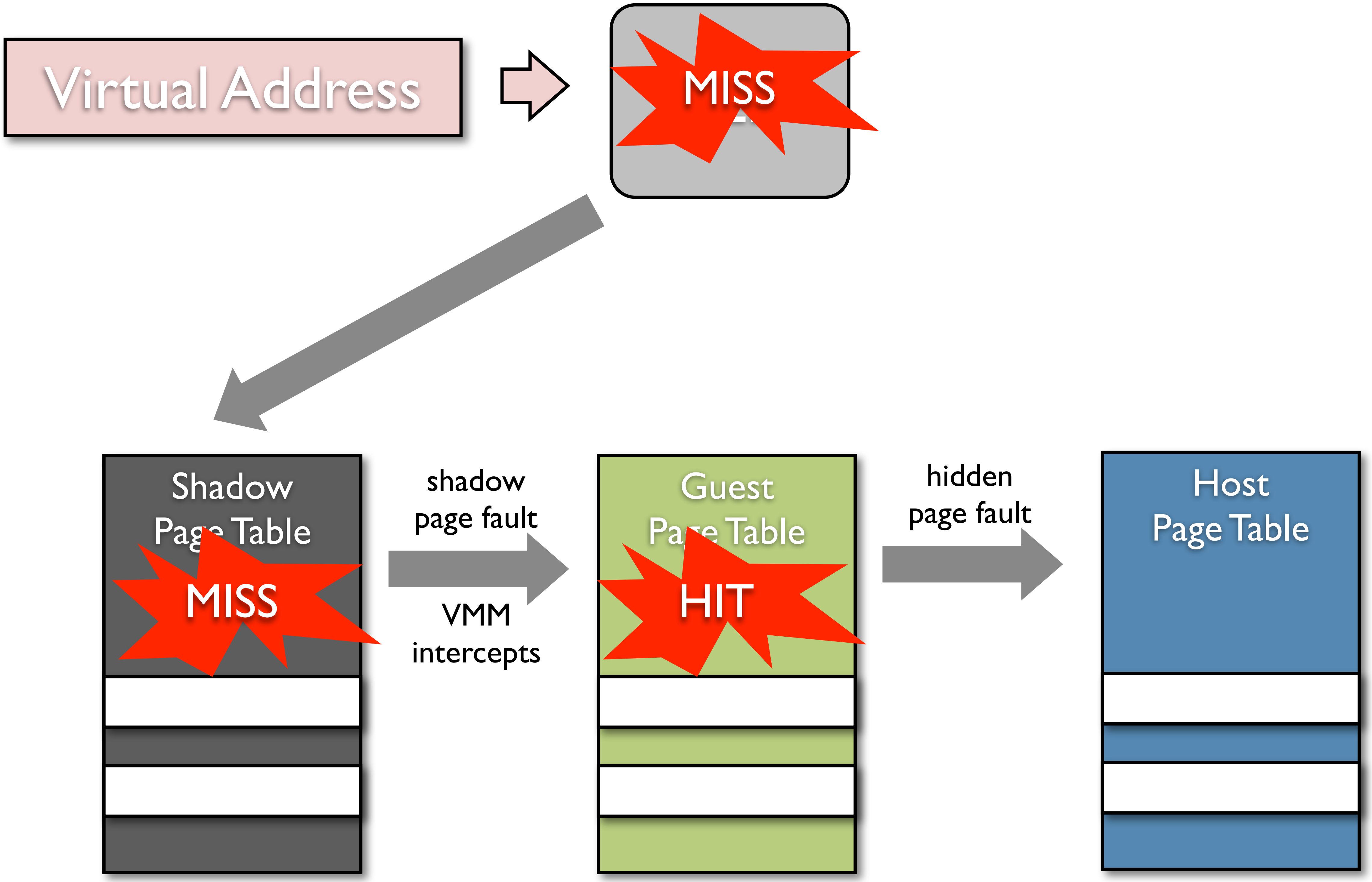
Shadow Page Tables with TLB



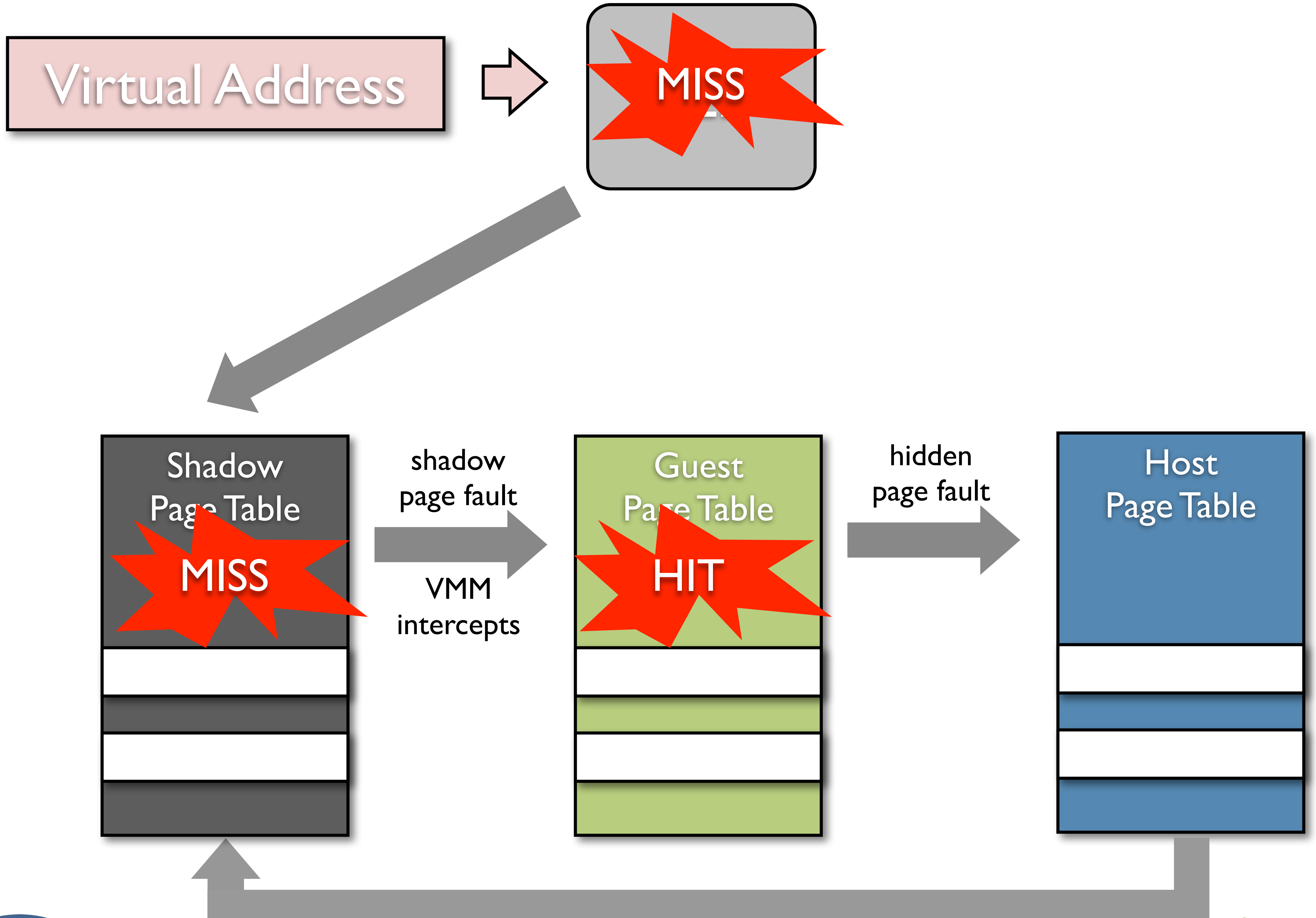
Shadow Page Tables with TLB



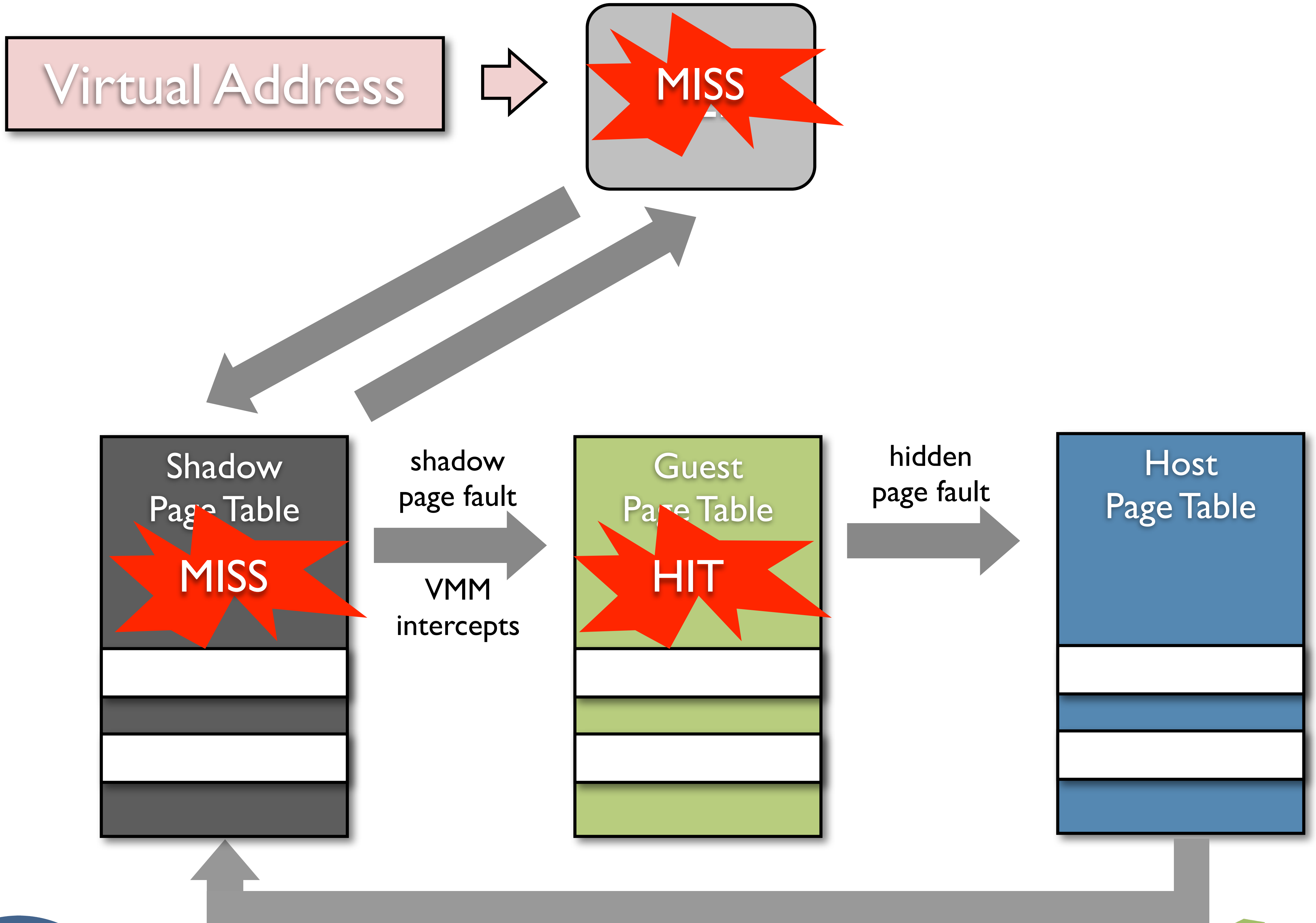
Shadow Page Tables with TLB



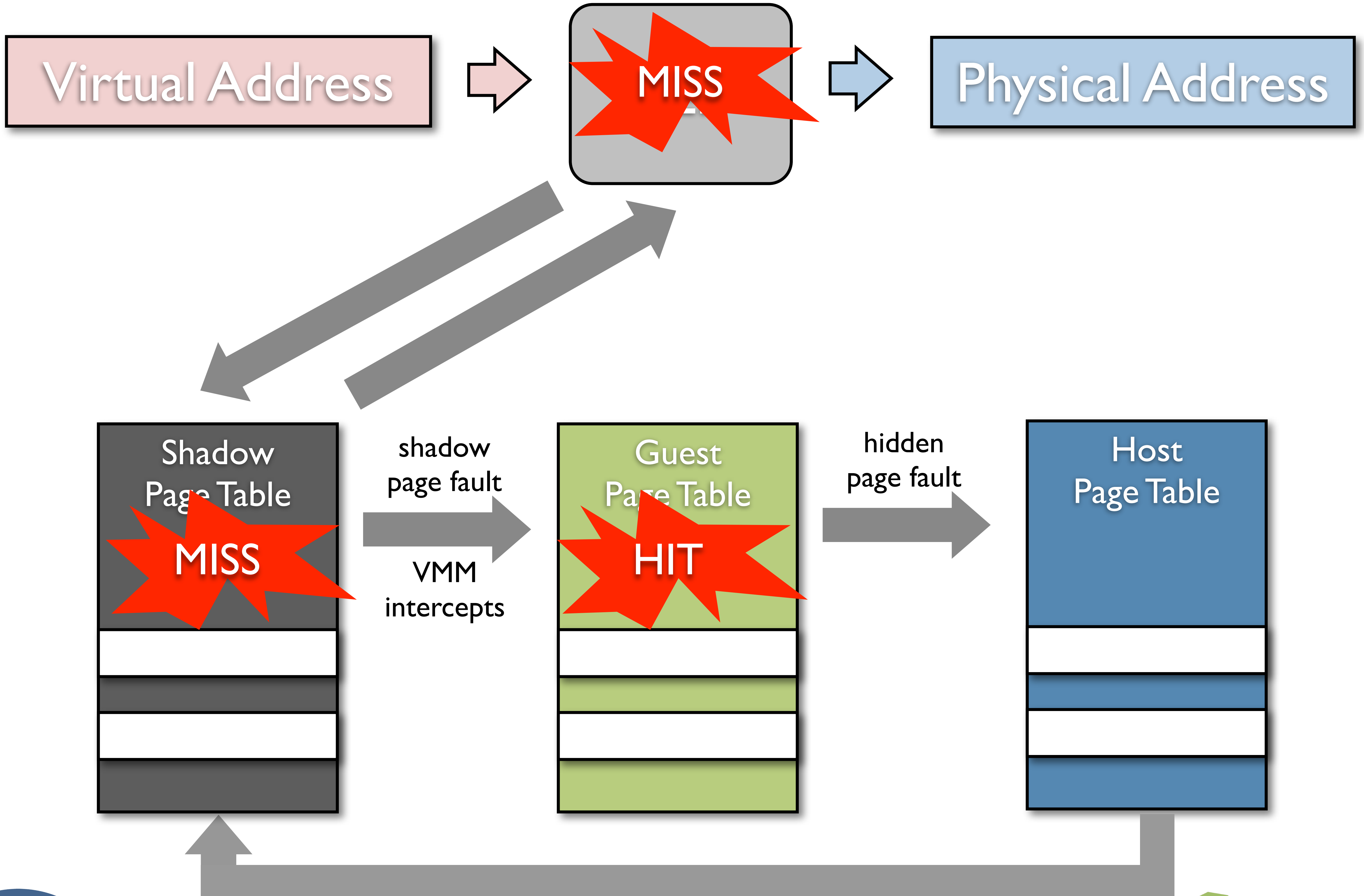
Shadow Page Tables with TLB



Shadow Page Tables with TLB



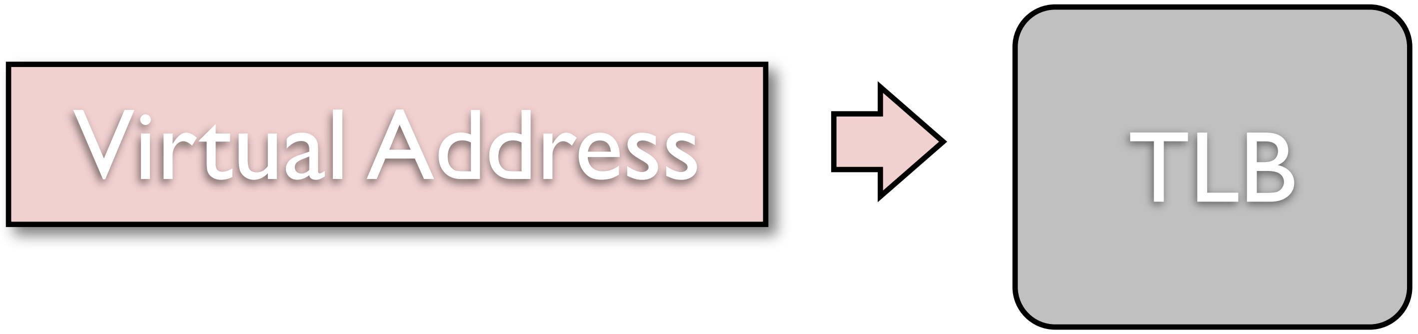
Shadow Page Tables with TLB



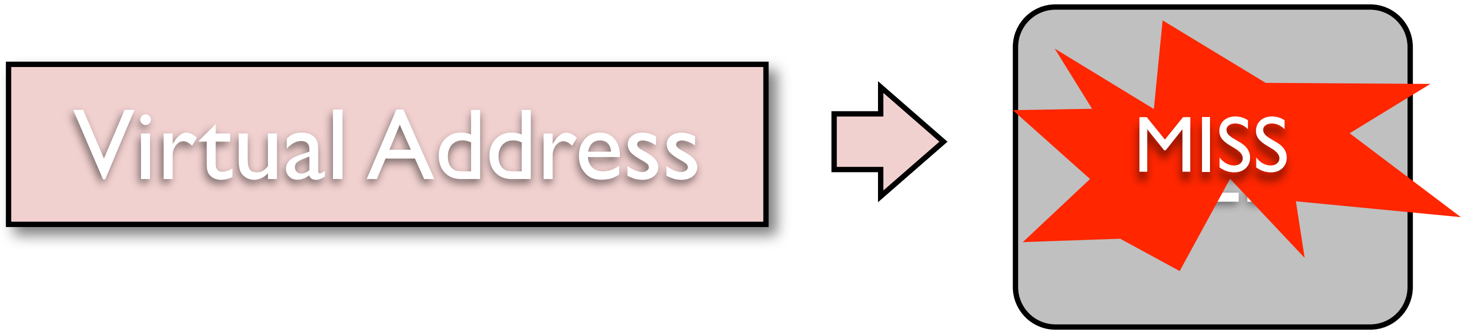
Shadow Page Tables with TLB

Virtual Address

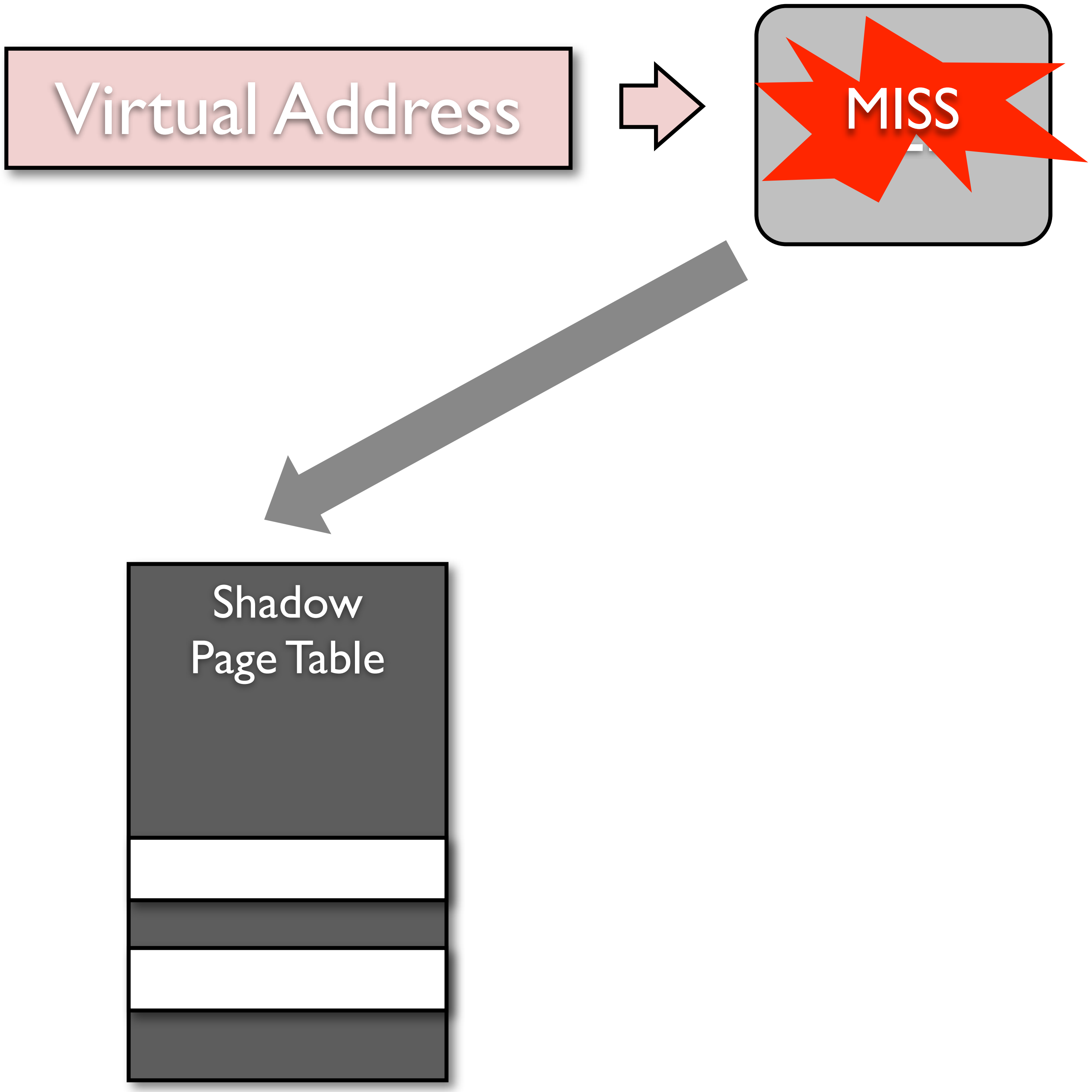
Shadow Page Tables with TLB



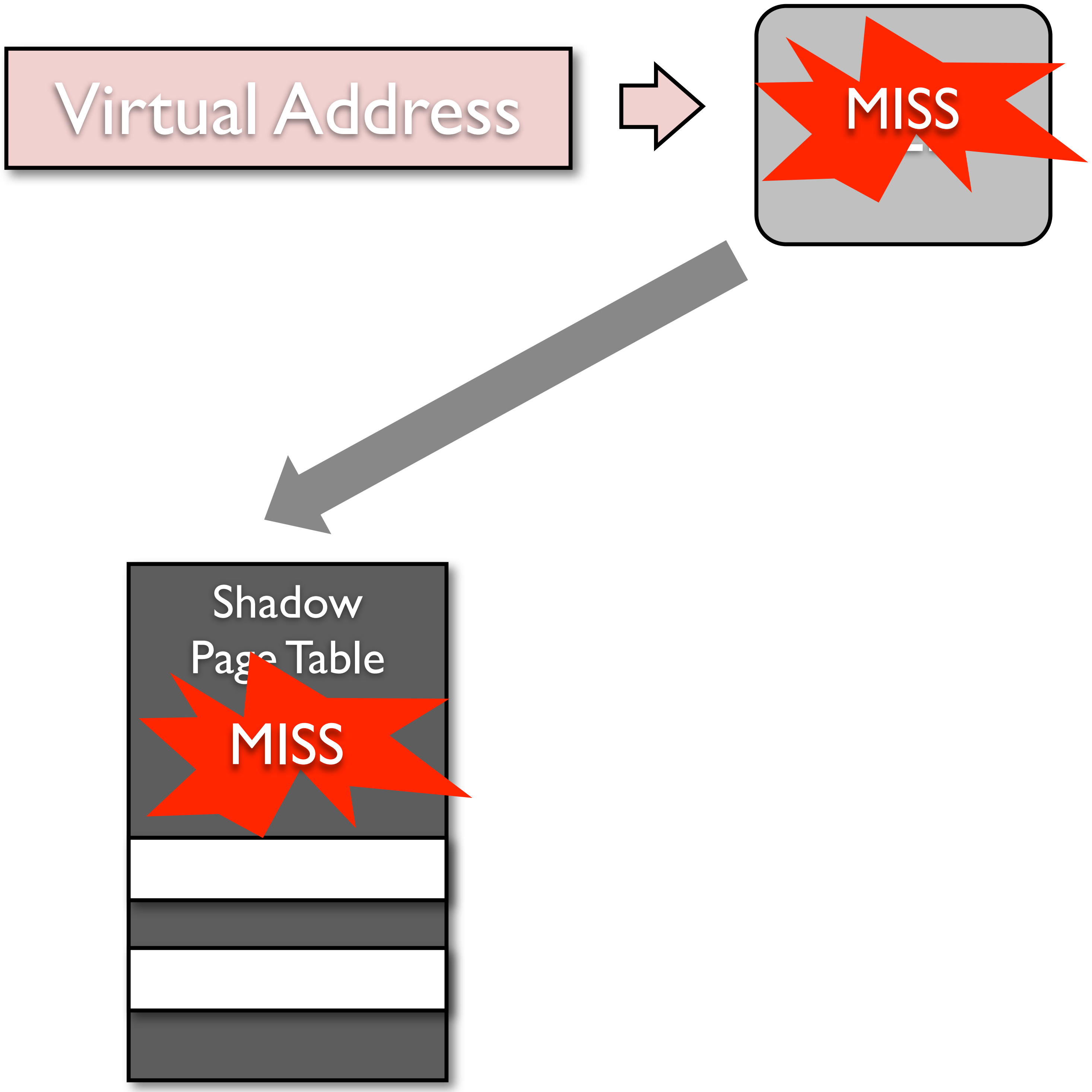
Shadow Page Tables with TLB



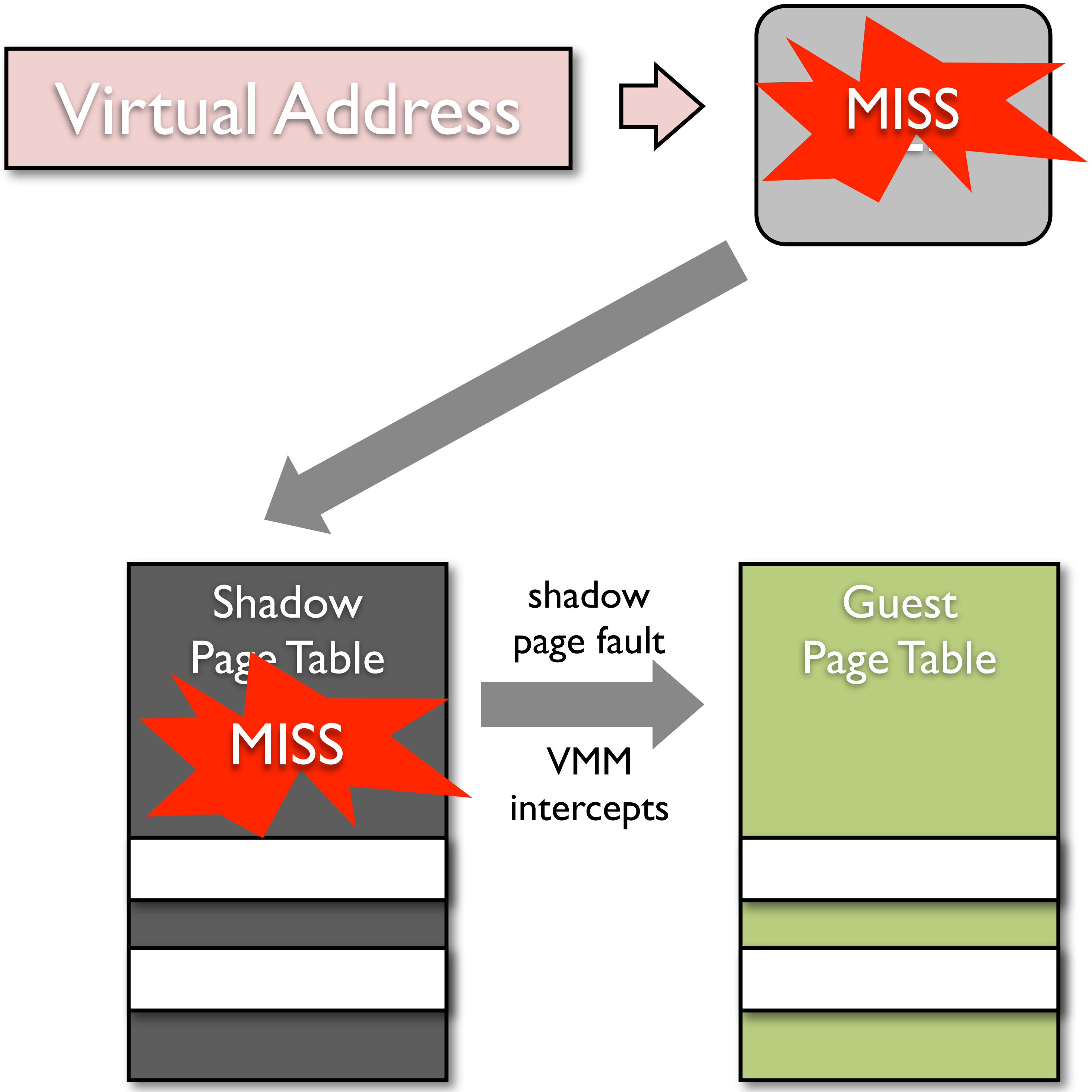
Shadow Page Tables with TLB



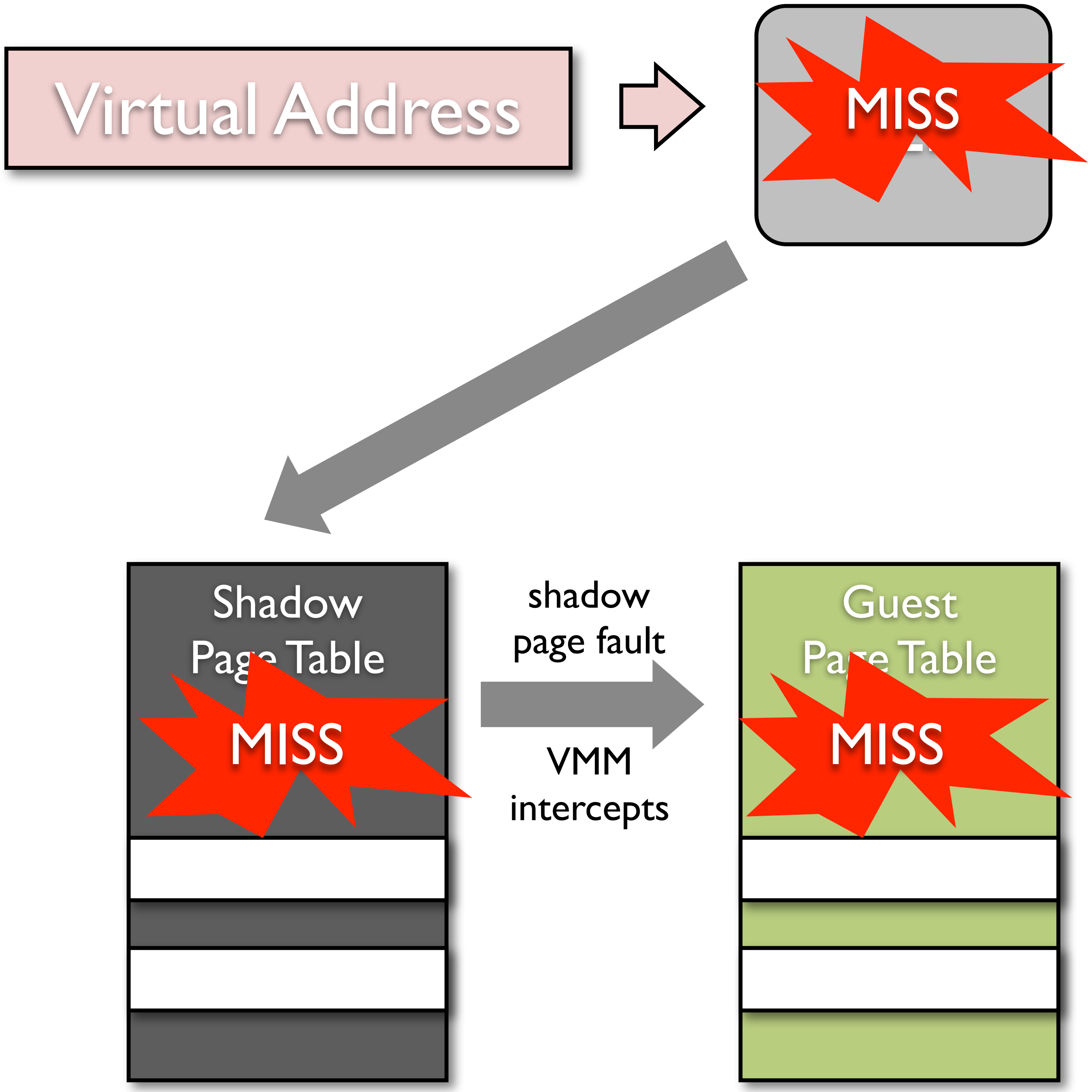
Shadow Page Tables with TLB



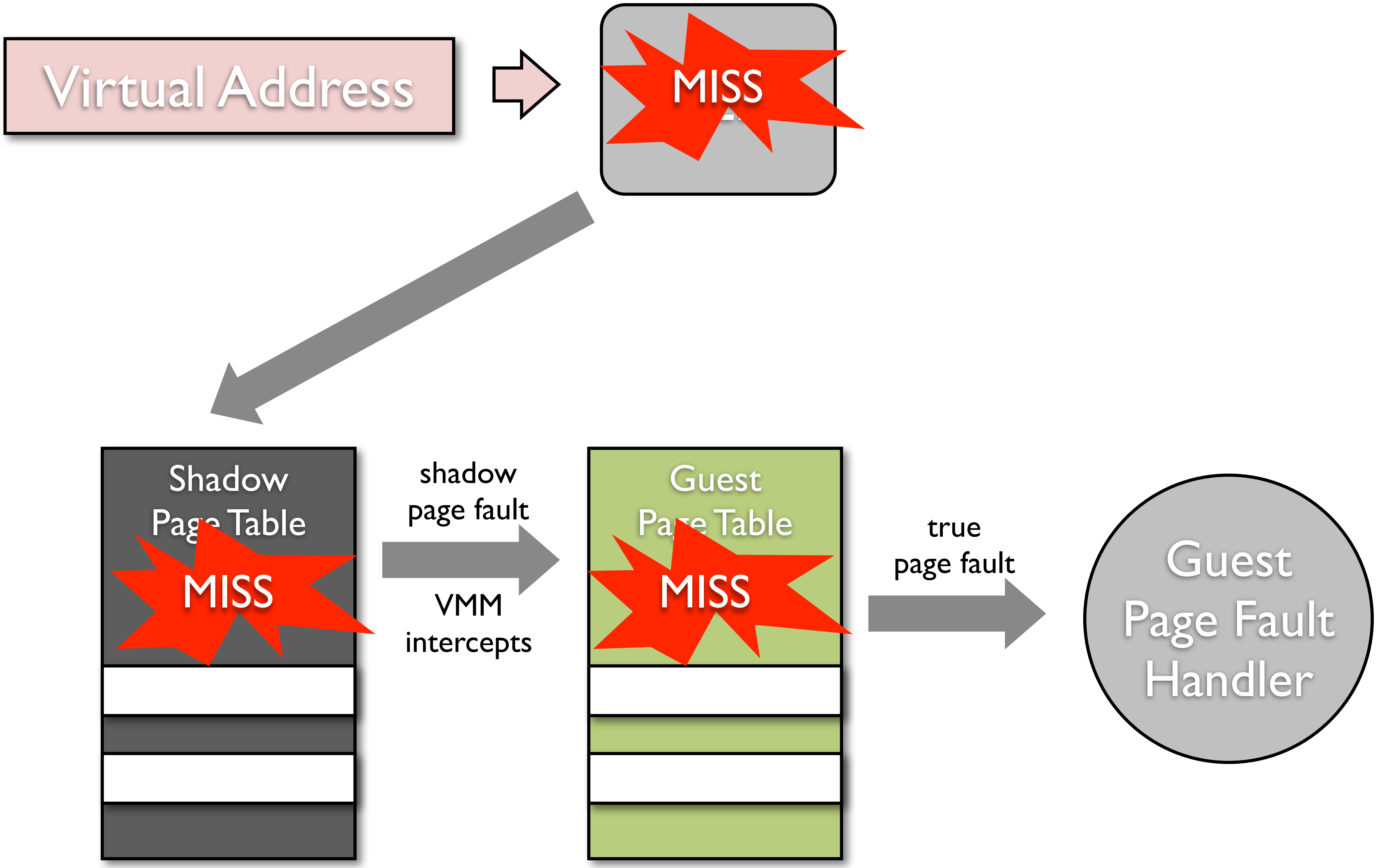
Shadow Page Tables with TLB



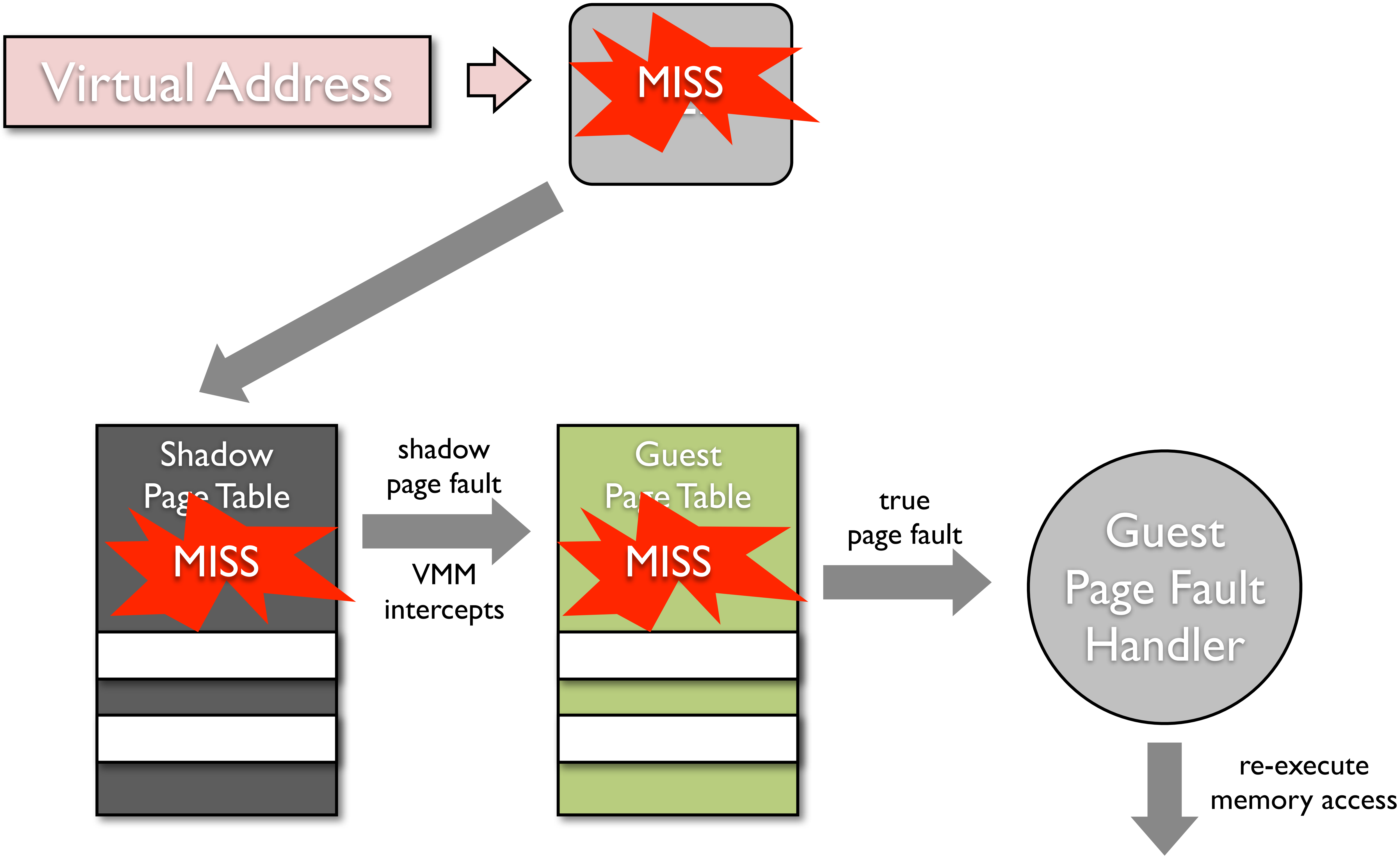
Shadow Page Tables with TLB



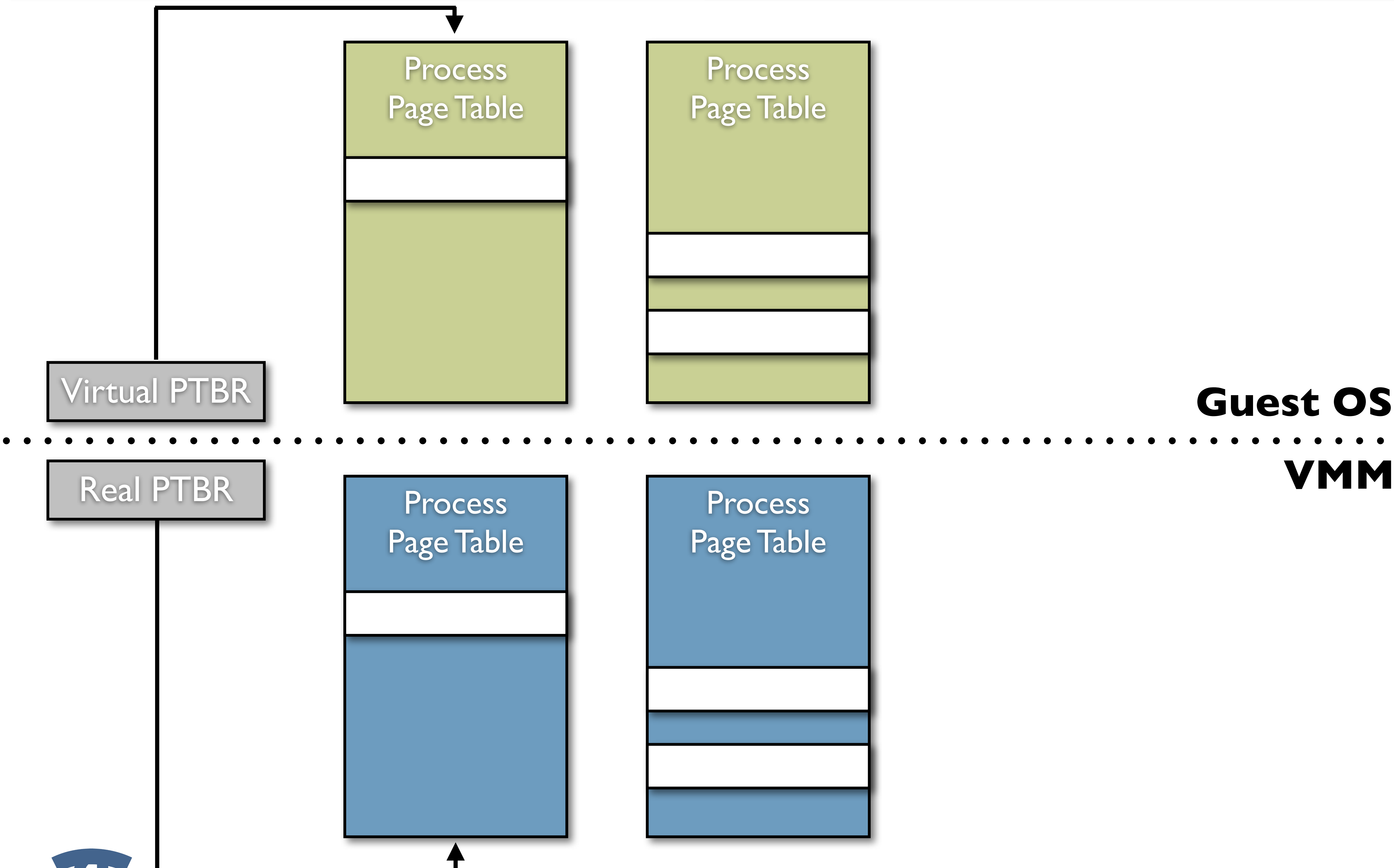
Shadow Page Tables with TLB



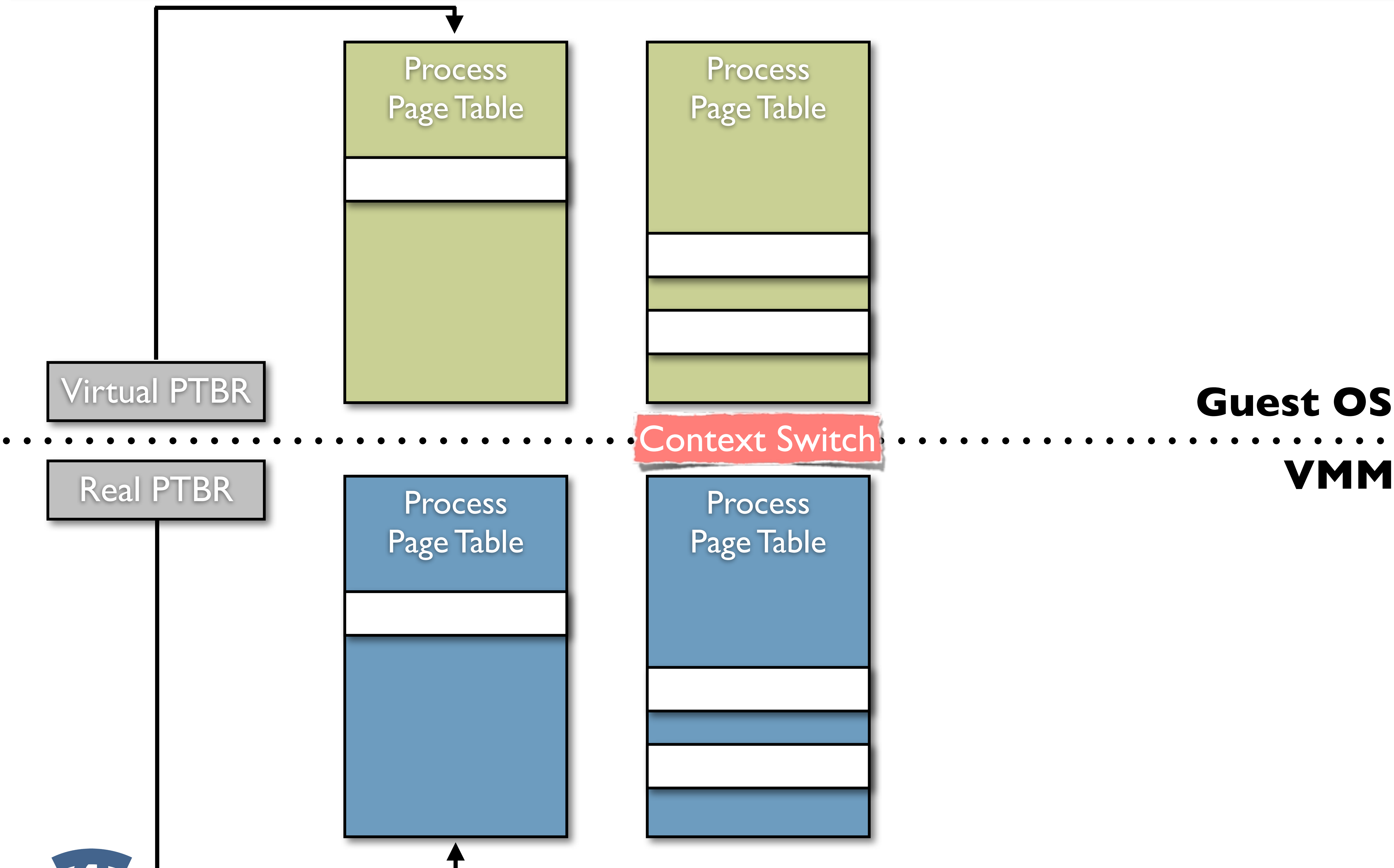
Shadow Page Tables with TLB



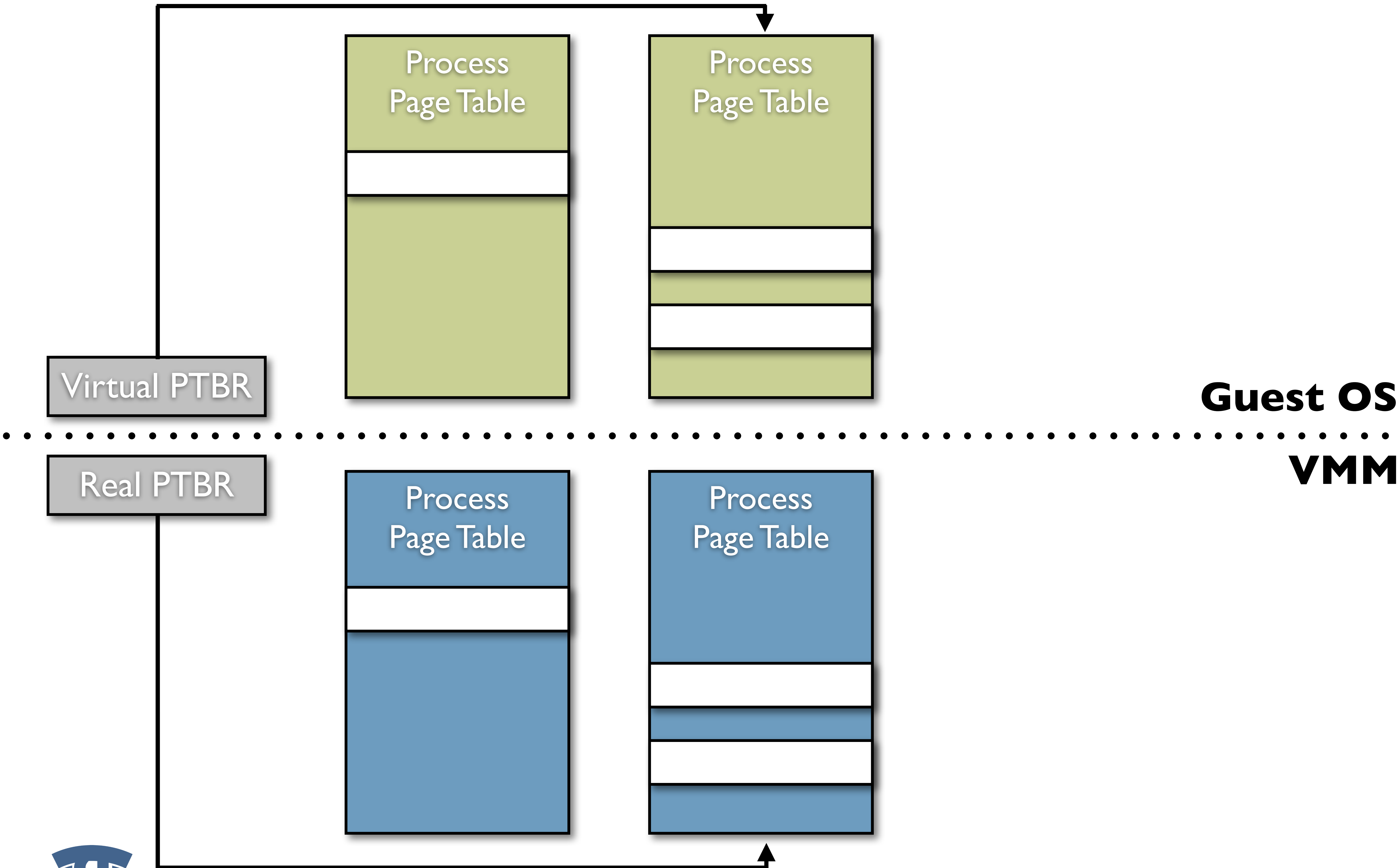
Shadow Page Tables in action...



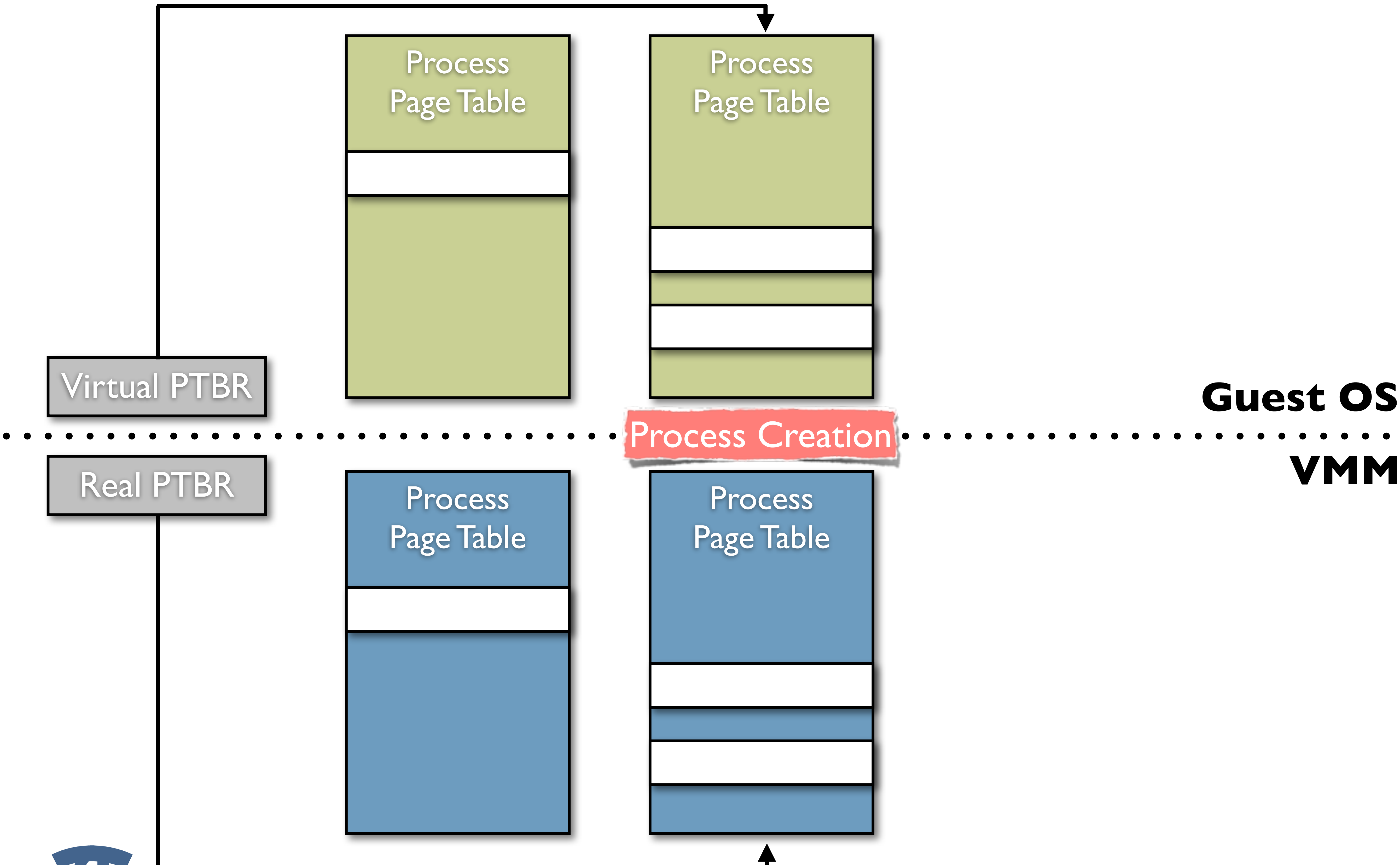
Shadow Page Tables in action...



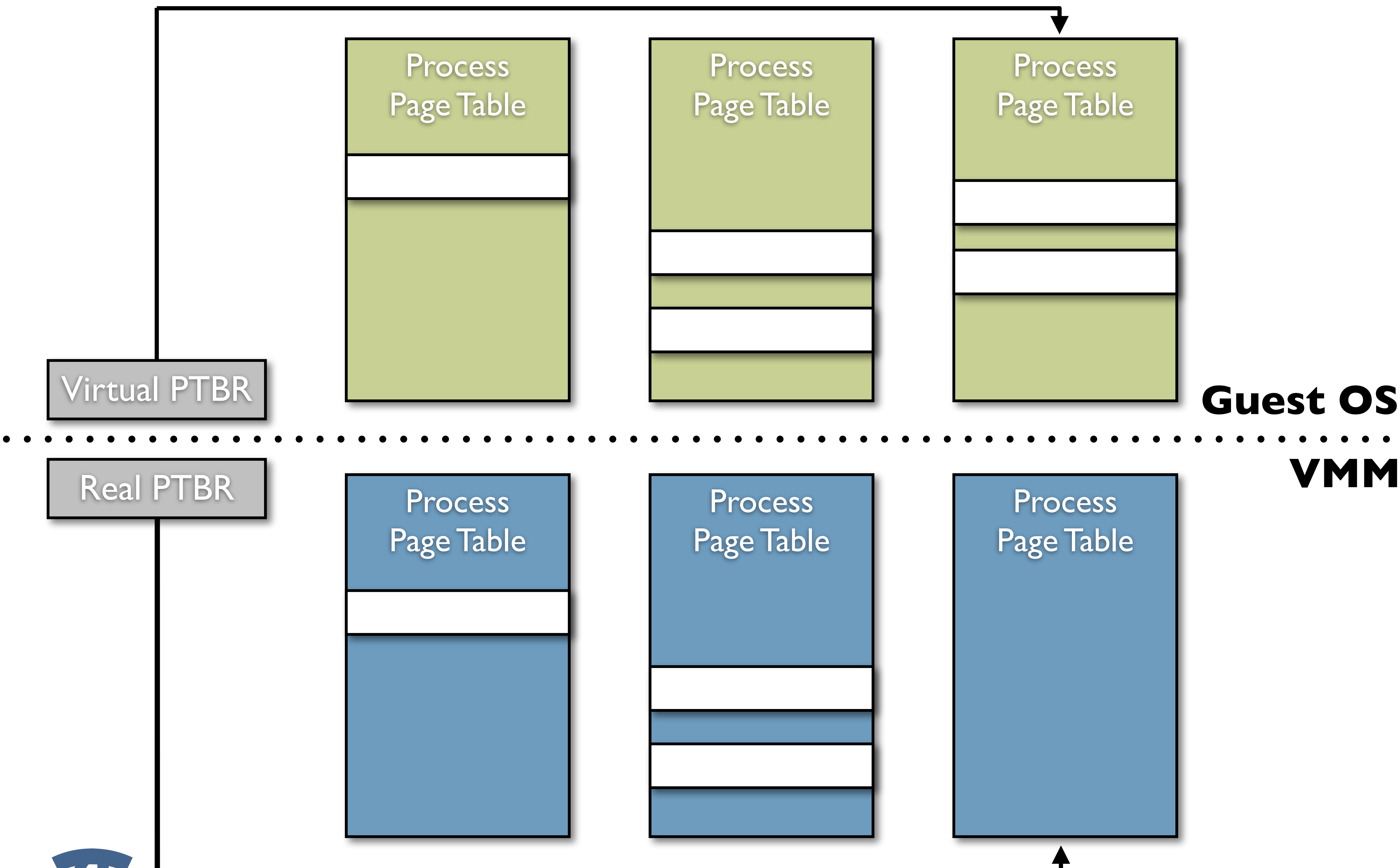
Shadow Page Tables in action...



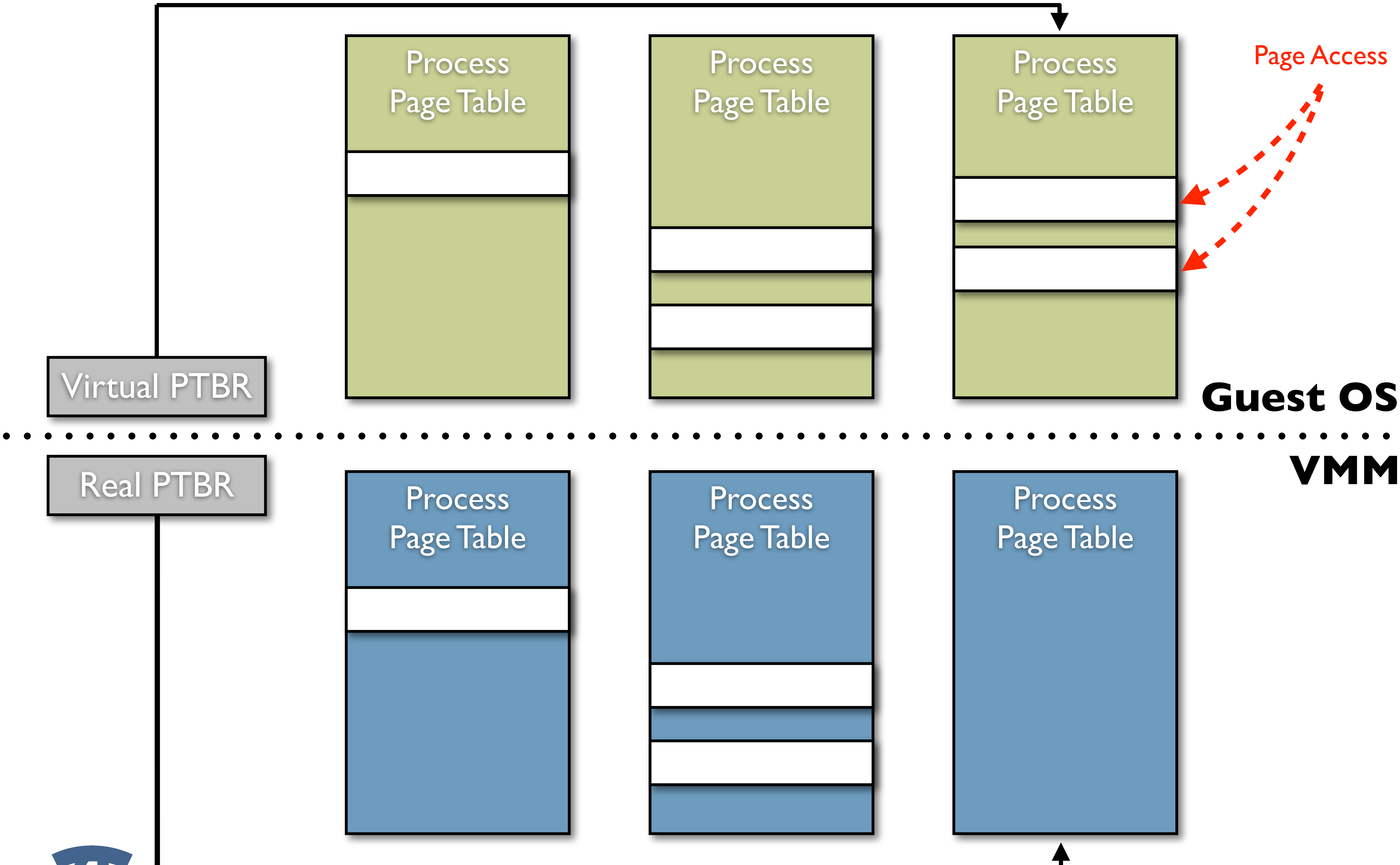
Shadow Page Tables in action...



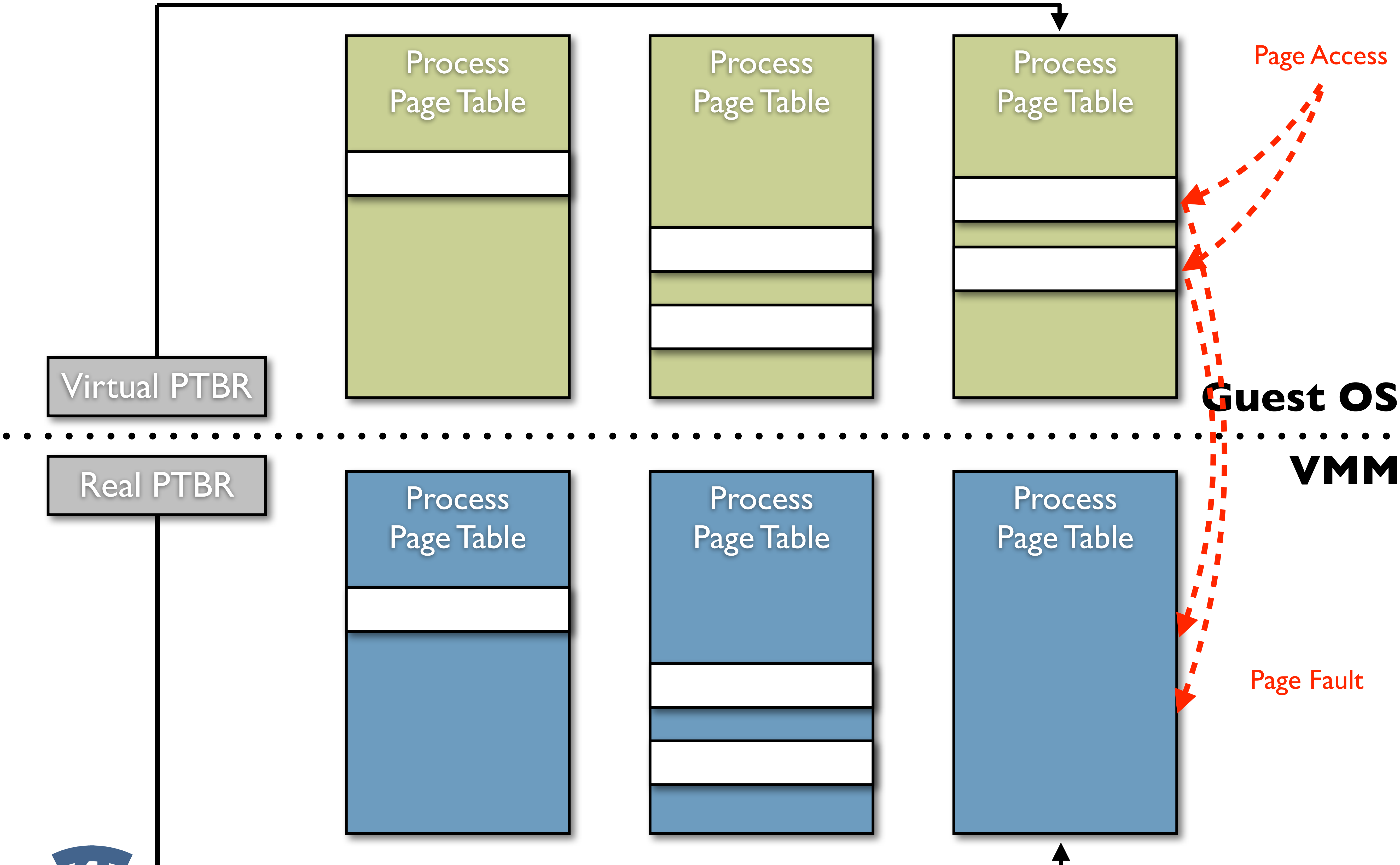
Shadow Page Tables in action...



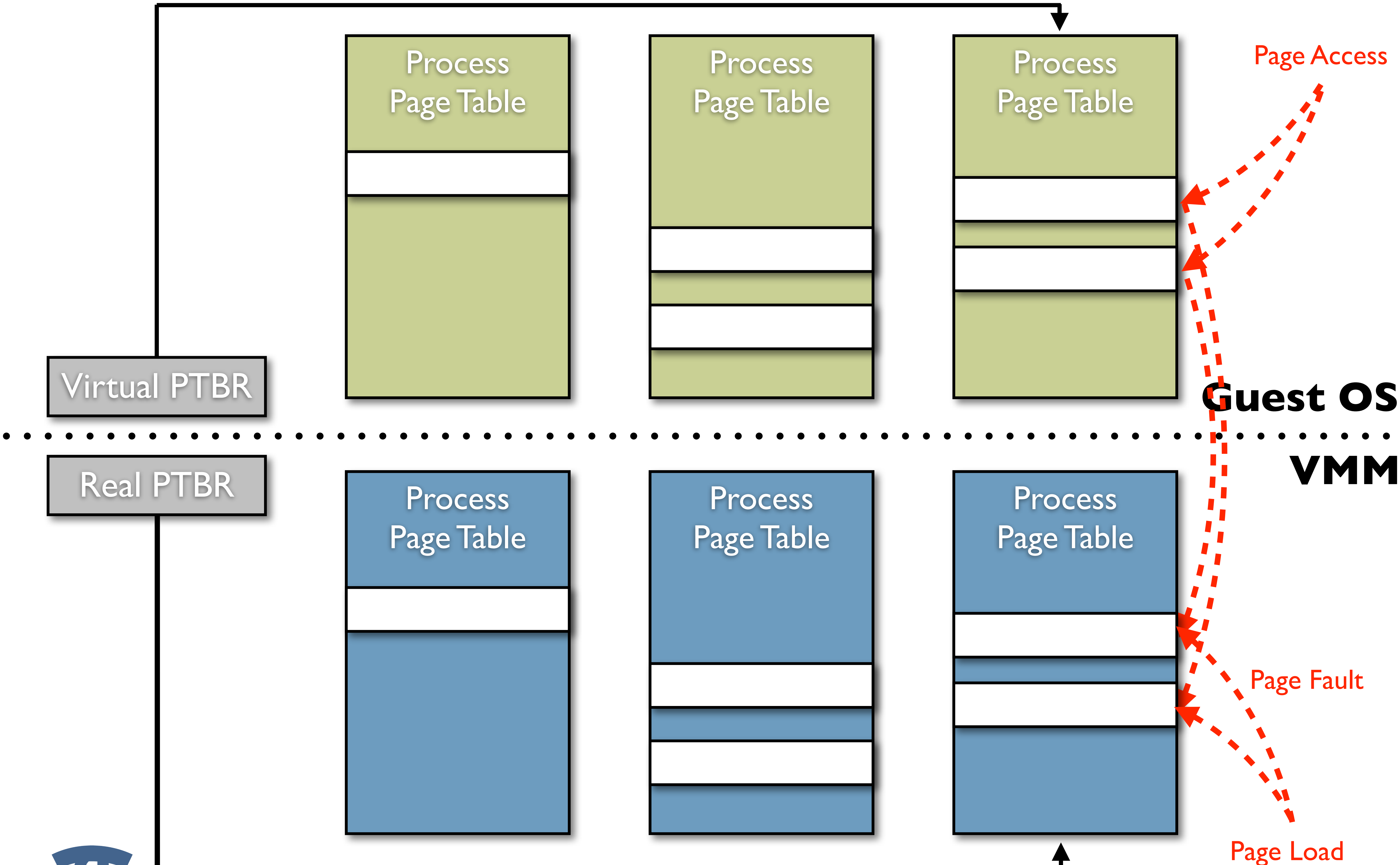
Shadow Page Tables in action...



Shadow Page Tables in action...



Shadow Page Tables in action...



Hardware Assisted Virtualization

Hardware Assisted Virtualization

- Difficulties of shadow page tables
  - Complex software-only implementation
  - Page fault and synchronization are critical mechanisms
  - Host memory overhead

Hardware Assisted Virtualization

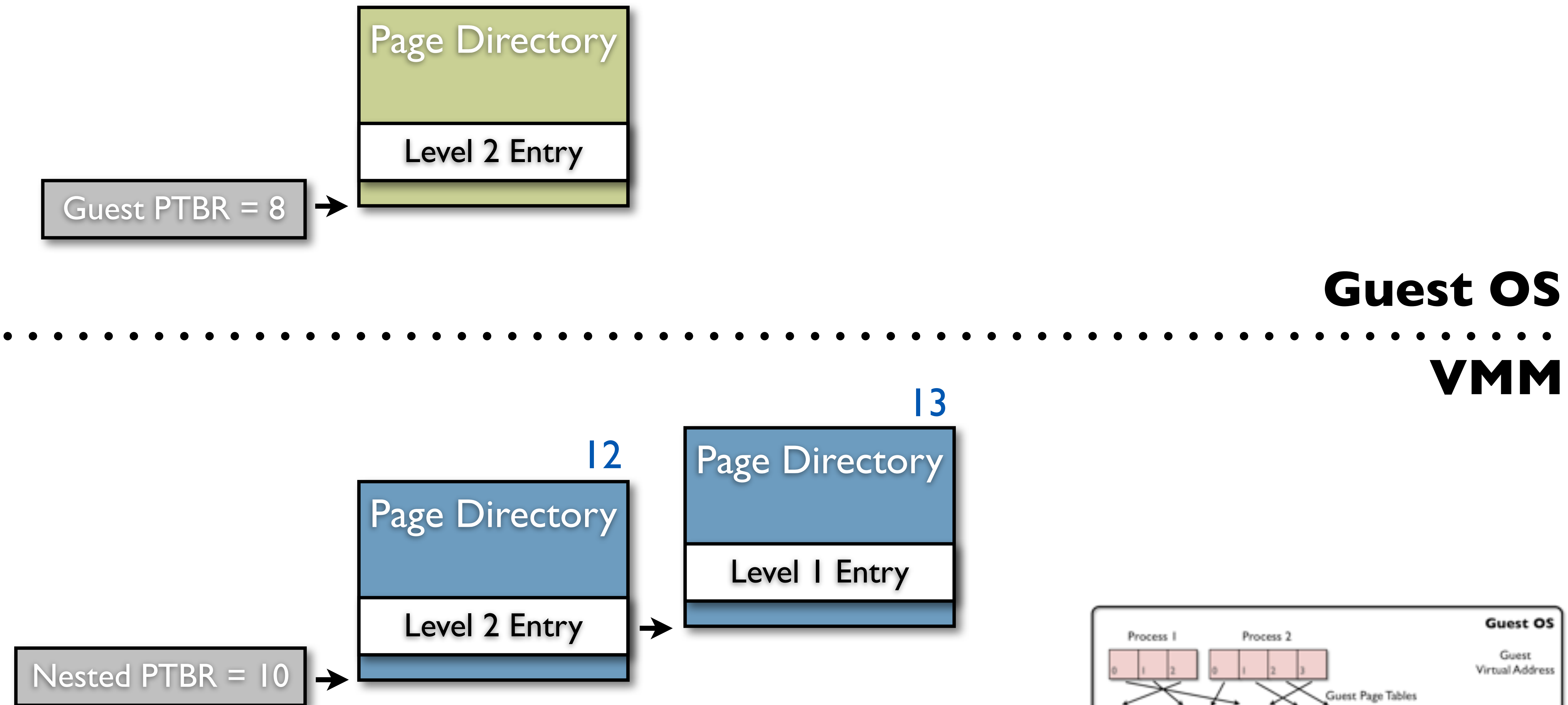
- Difficulties of shadow page tables
  - Complex software-only implementation
  - Page fault and synchronization are critical mechanisms
  - Host memory overhead

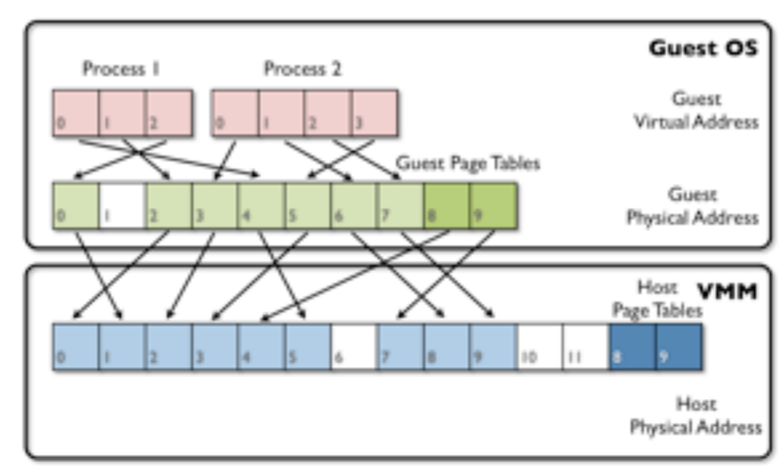
- Why do we need them?
  - MMU was not designed for virtualization
  - MMU is not aware of the two-levels address translation

Hardware Assisted Virtualization

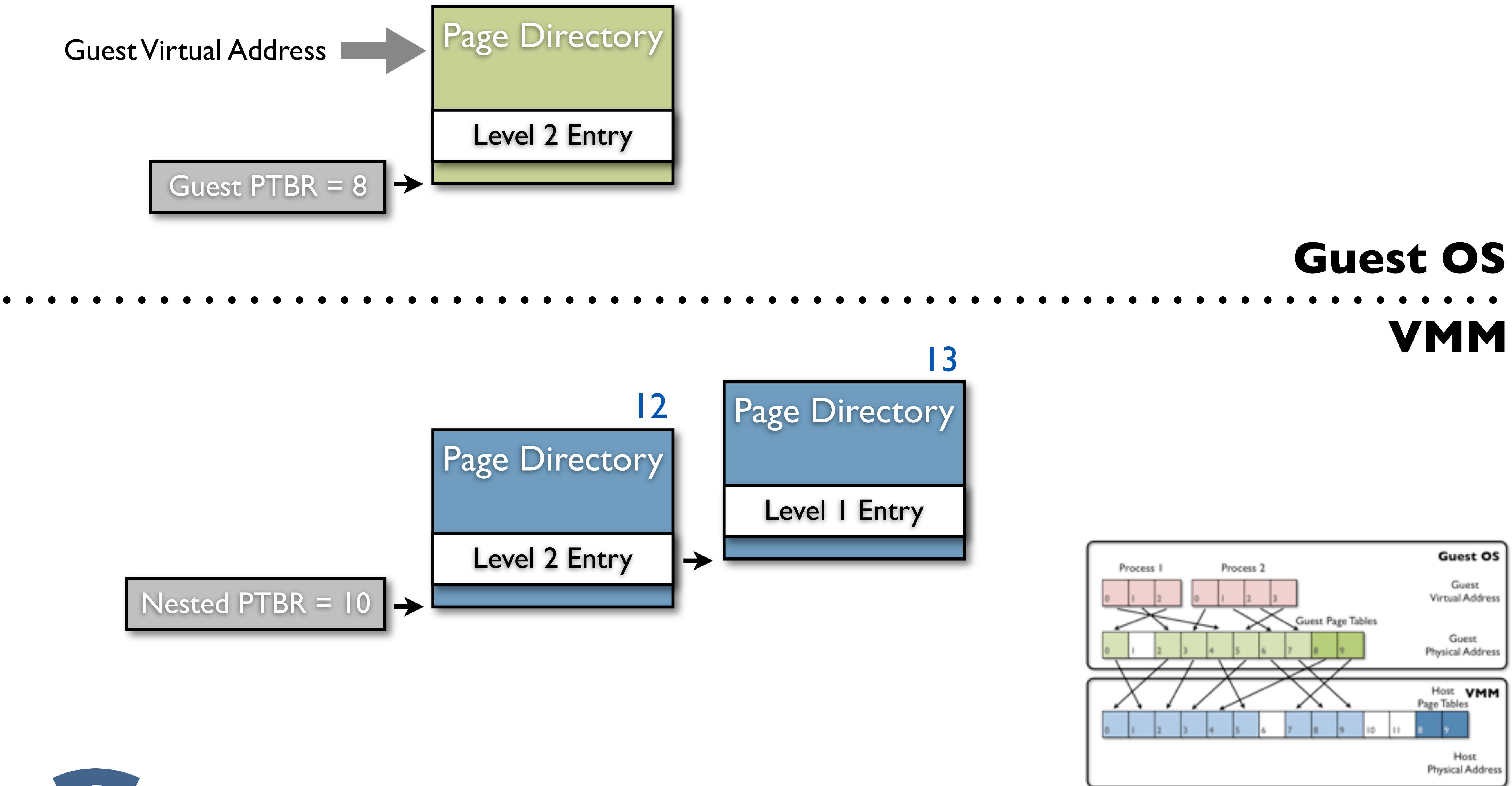
- Difficulties of shadow page tables
  - Complex software-only implementation
  - Page fault and synchronization are critical mechanisms
  - Host memory overhead
- Why do we need them?
  - MMU was not designed for virtualization
  - MMU is not aware of the two-levels address translation
- New CPUs support two-levels address translation in hardware!
  - Nested Page Tables a.k.a. Rapid Virtualization Indexing (AMD)
  - Extended Page Table (INTEL)

Nested Page Tables

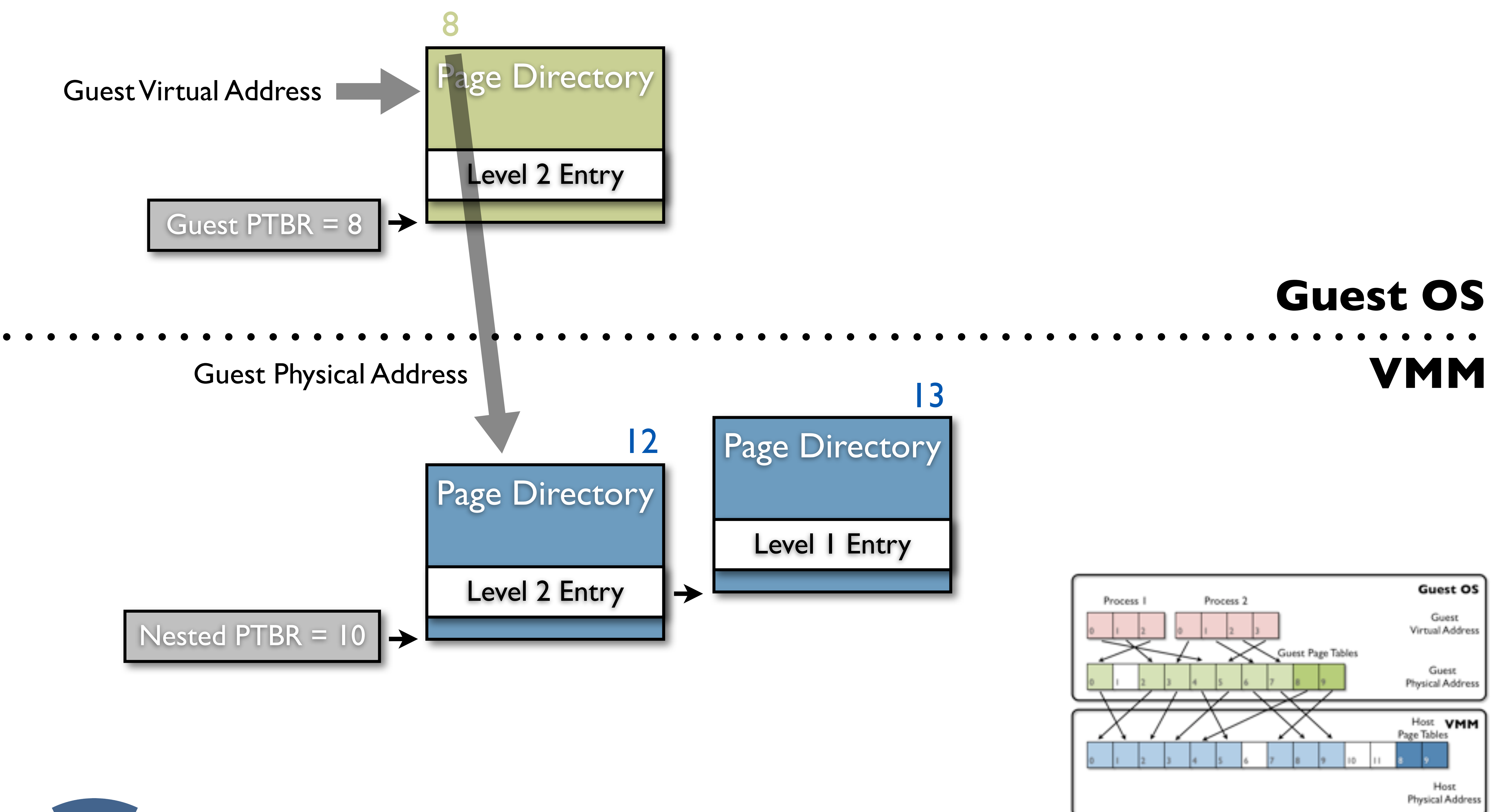




Nested Page Tables



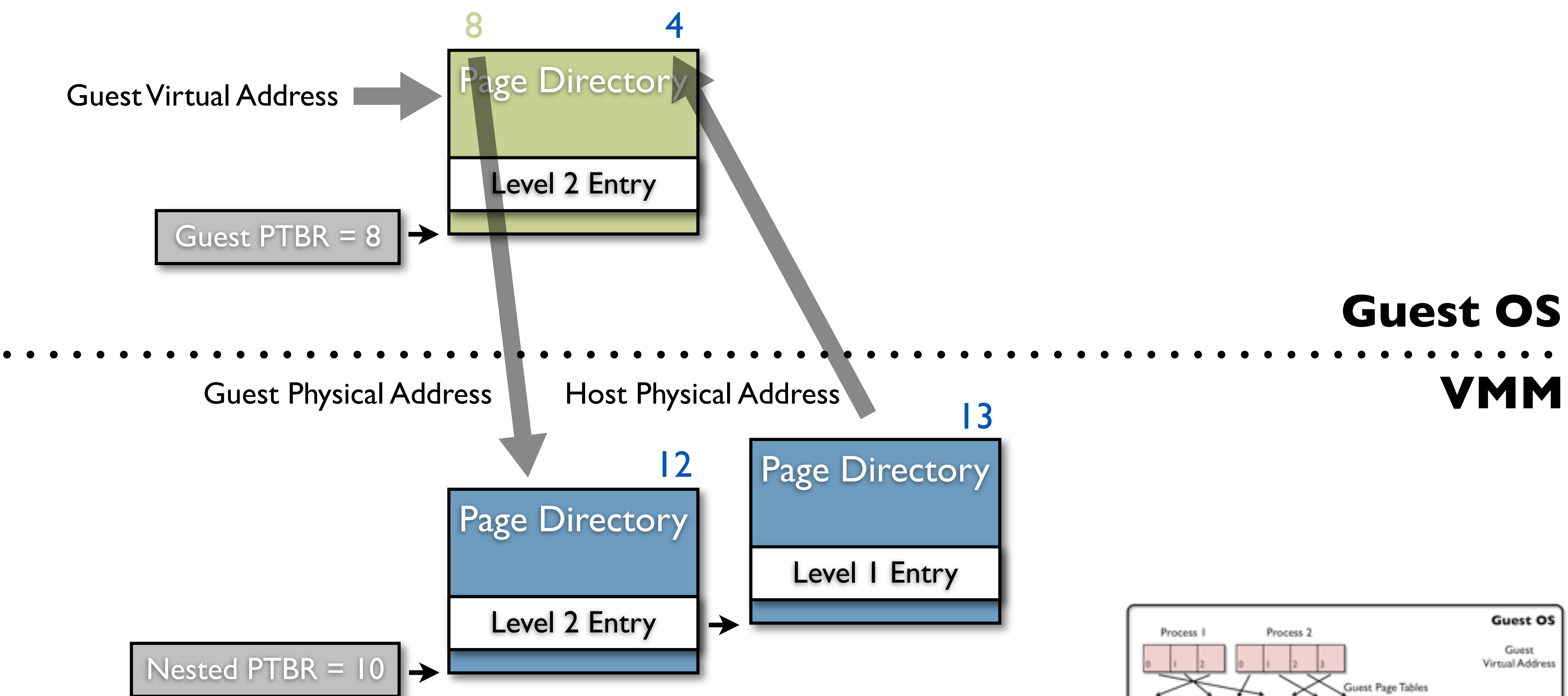
Nested Page Tables

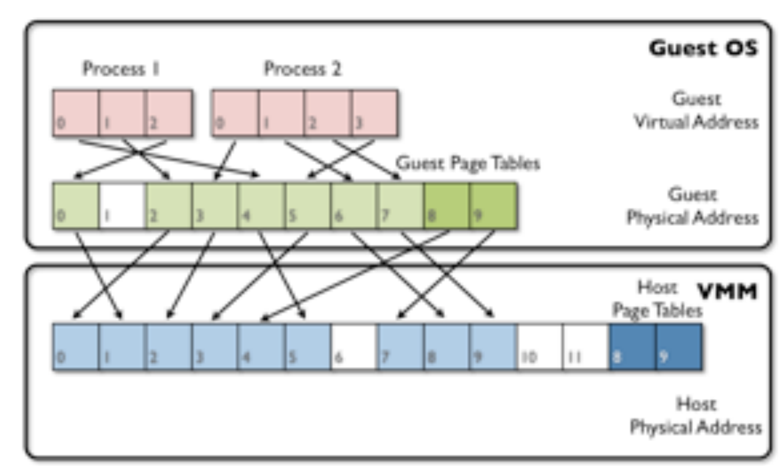


ISTITUTO DI SCIENZA E TECNOLOGIE DELL'INFORMAZIONE "A. FAEDO"

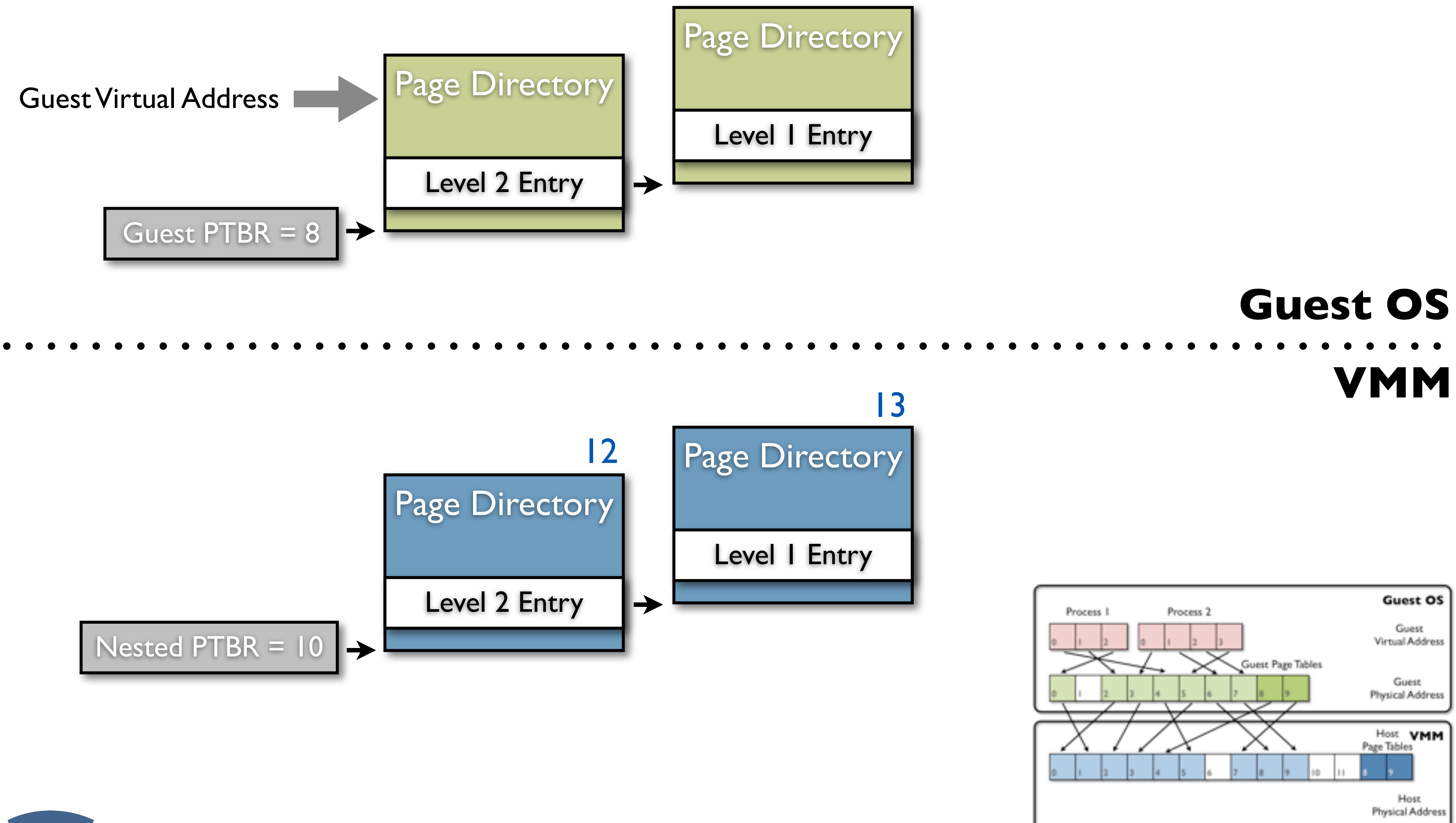
lunedì 17 dicembre 12

Nested Page Tables

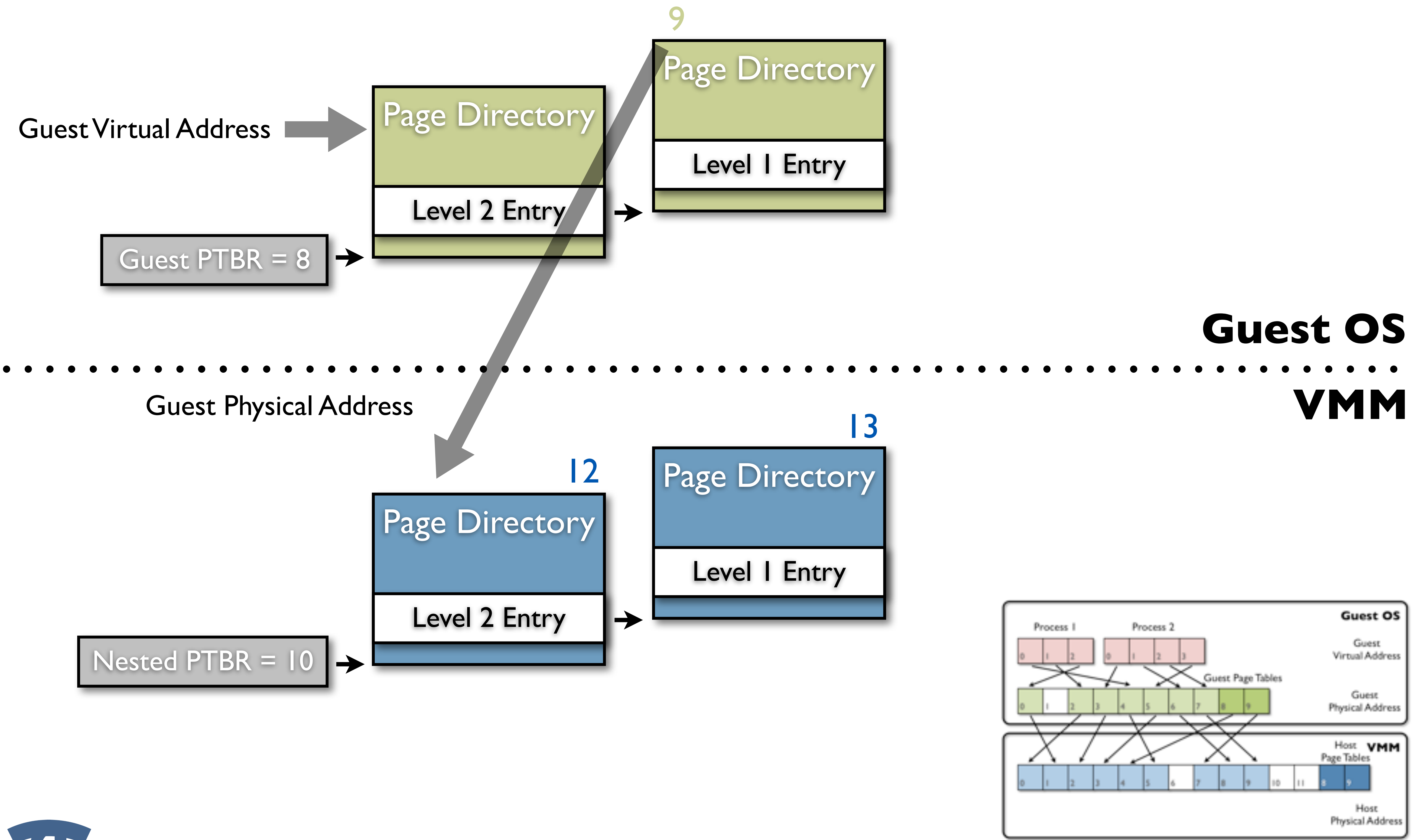




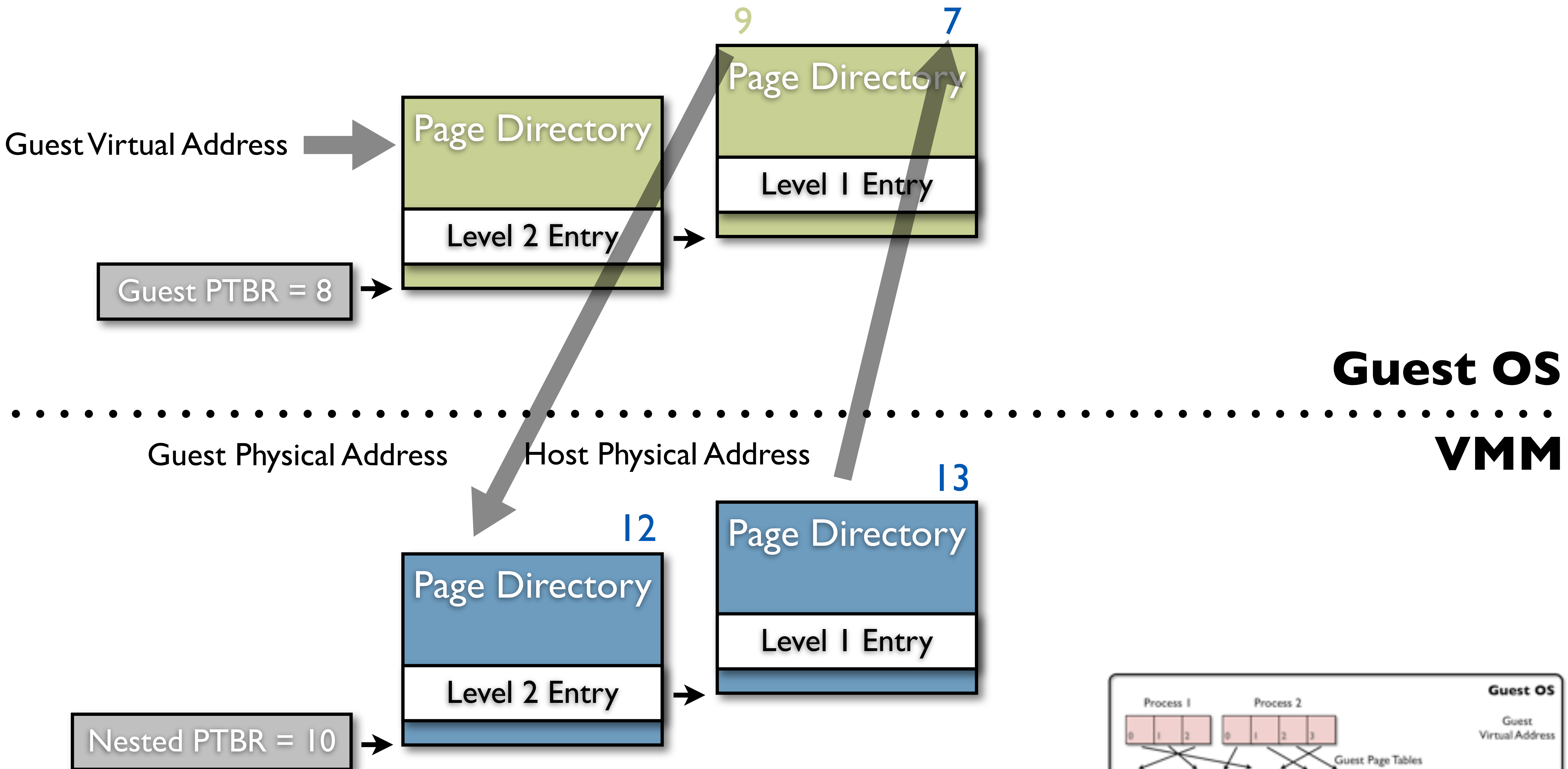
Nested Page Tables

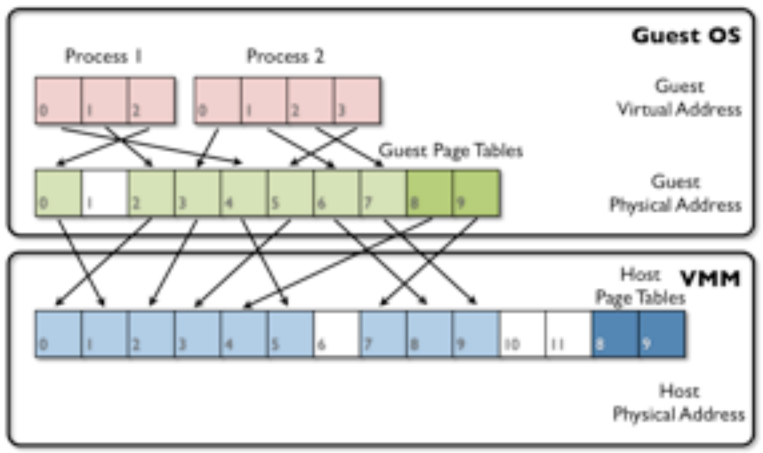


Nested Page Tables

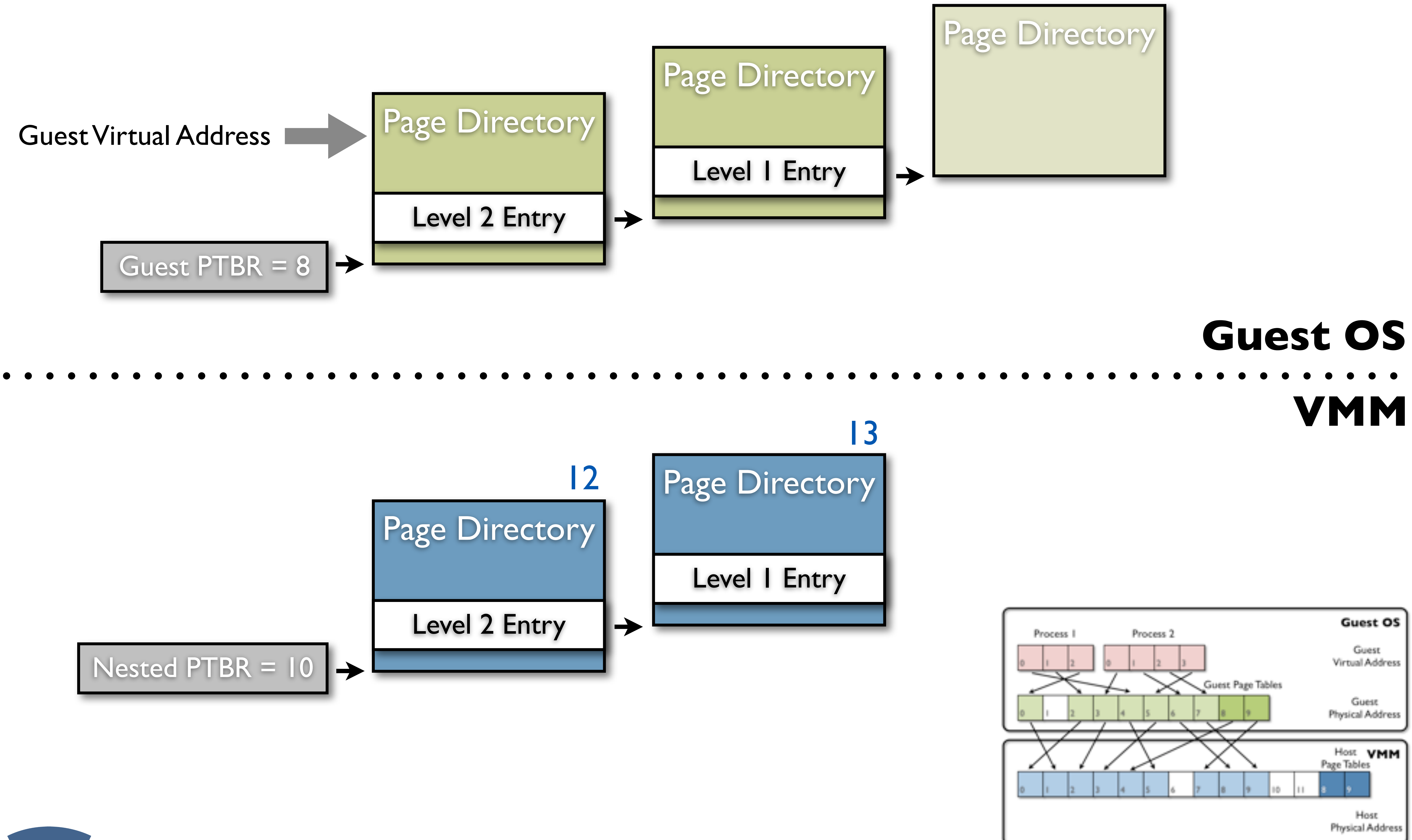


Nested Page Tables

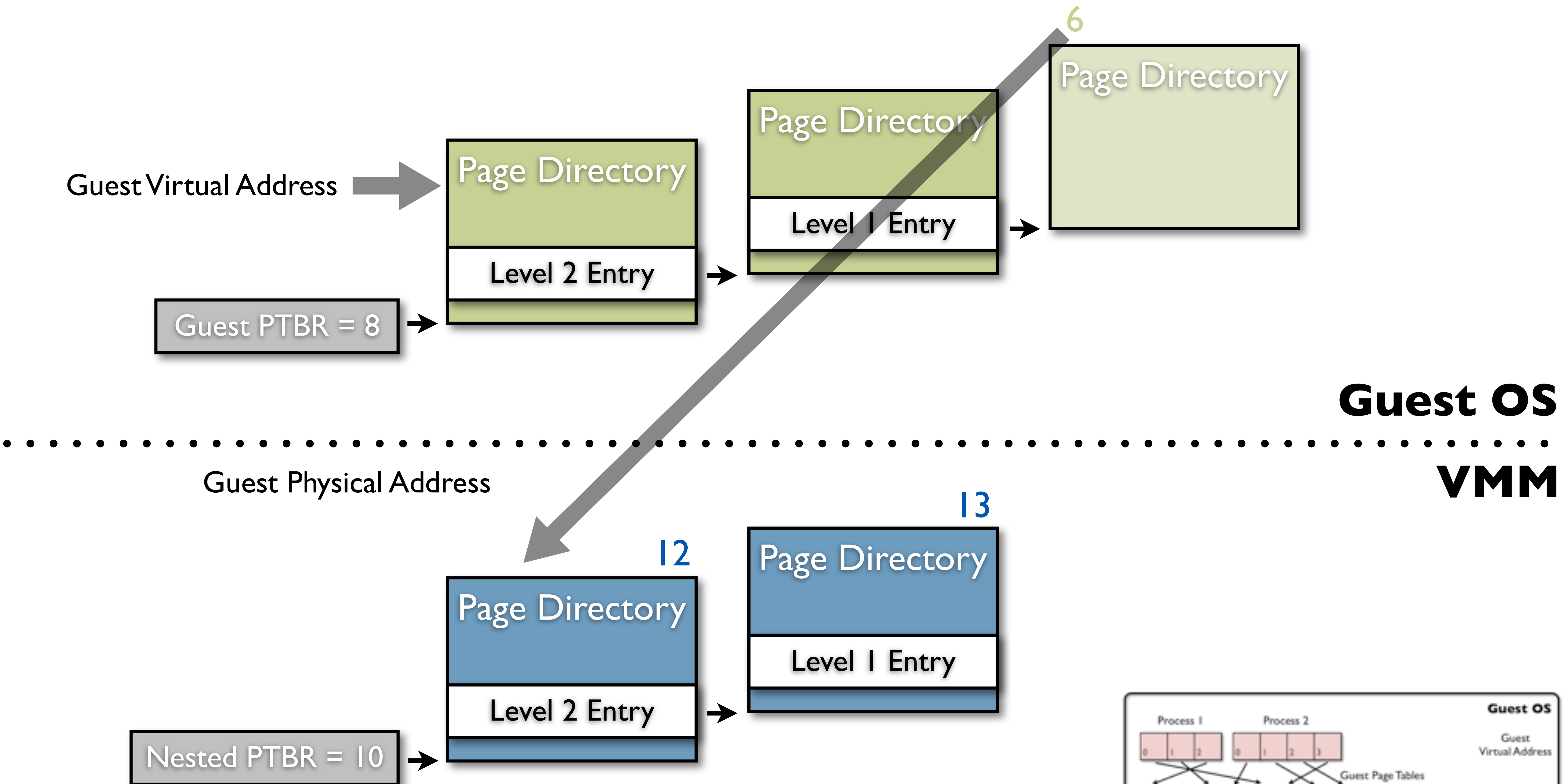


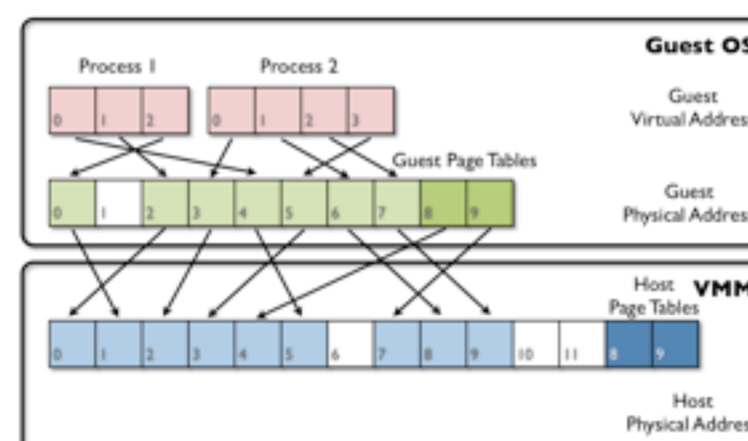


Nested Page Tables

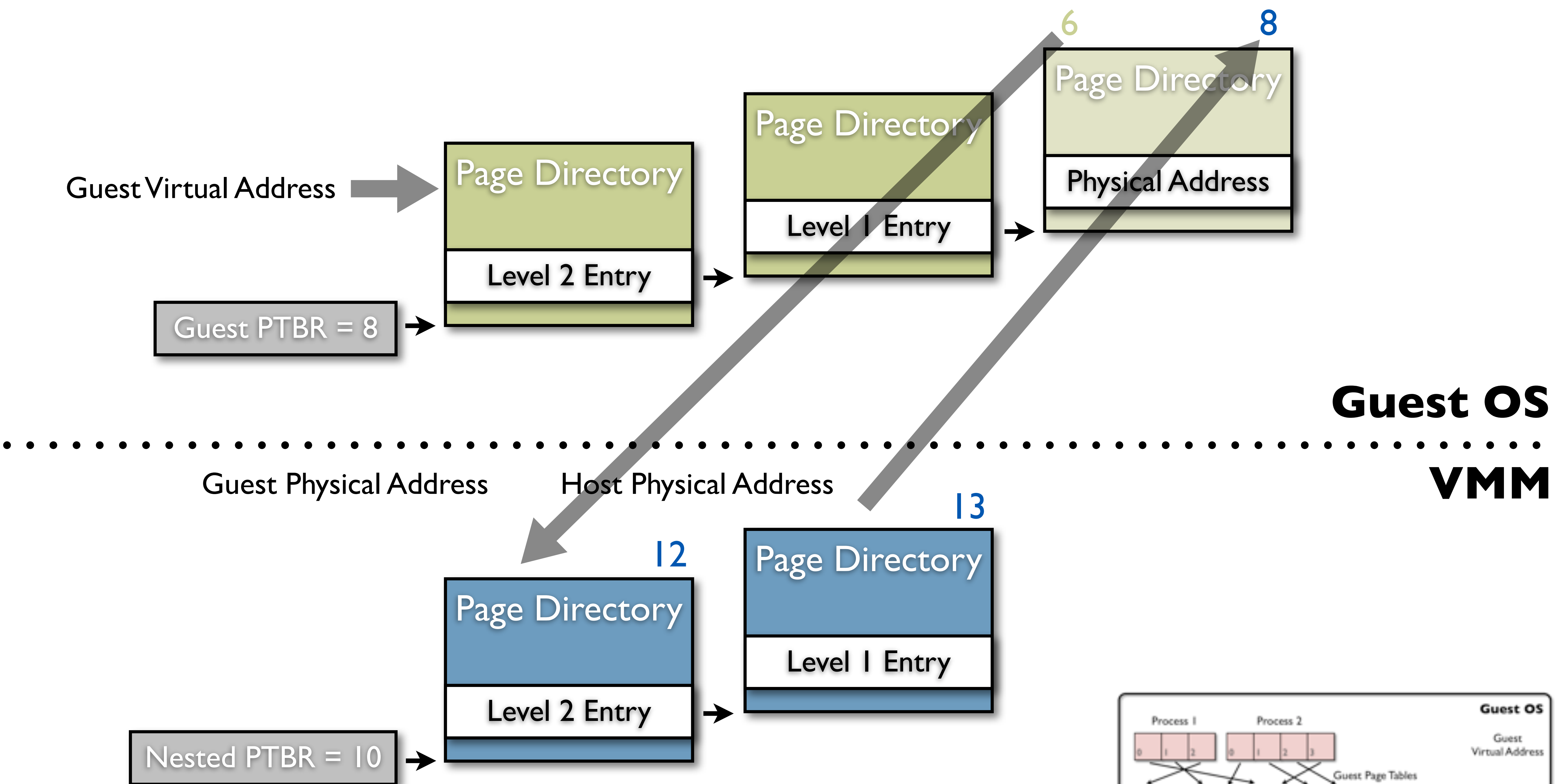


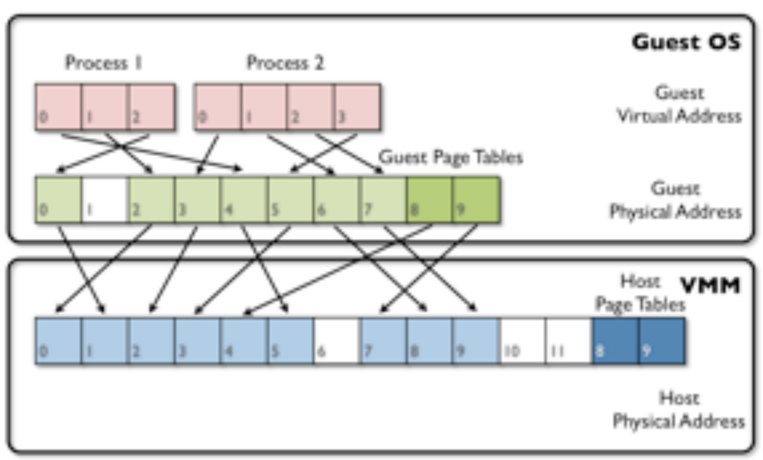
Nested Page Tables





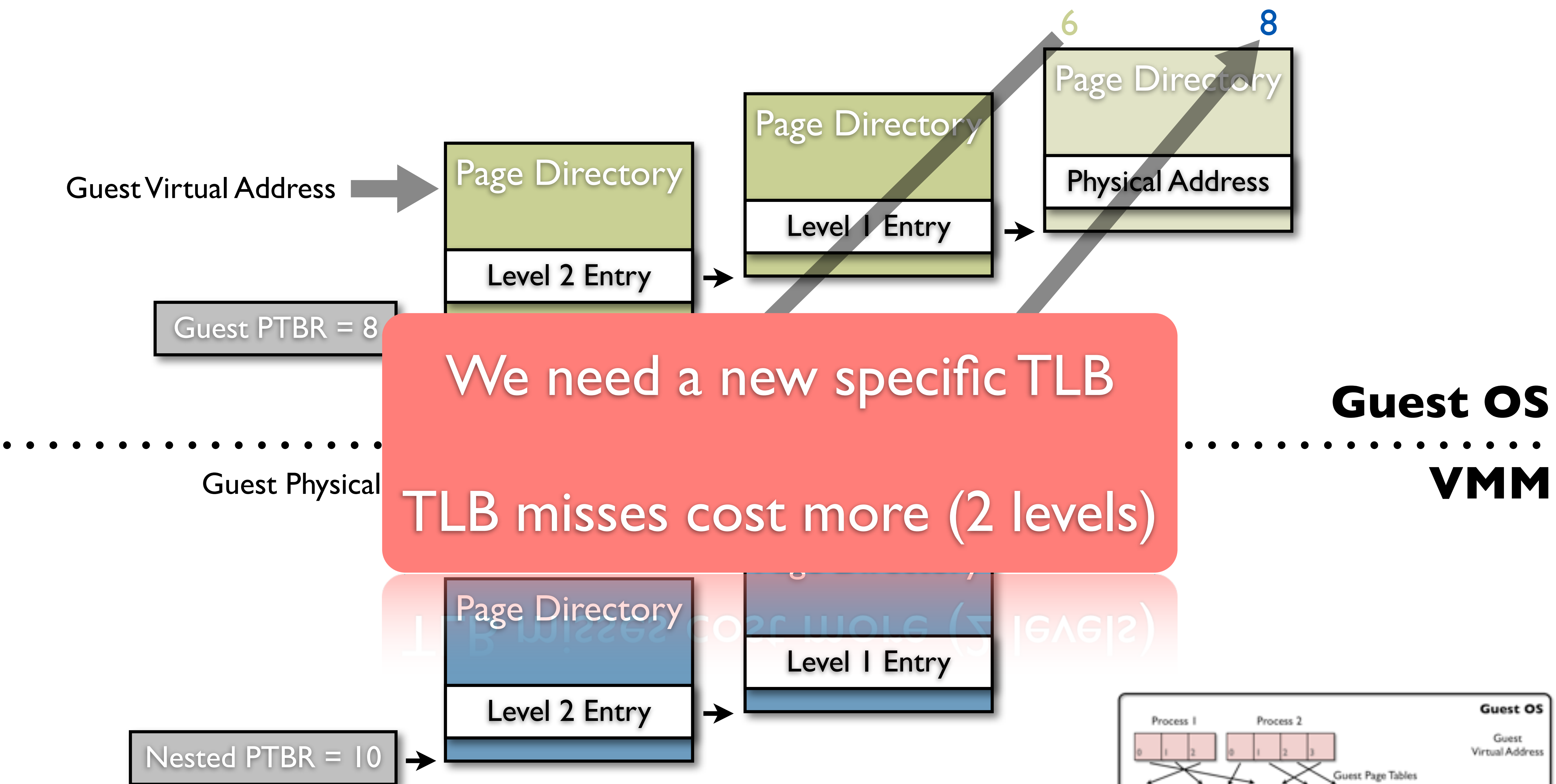
Nested Page Tables





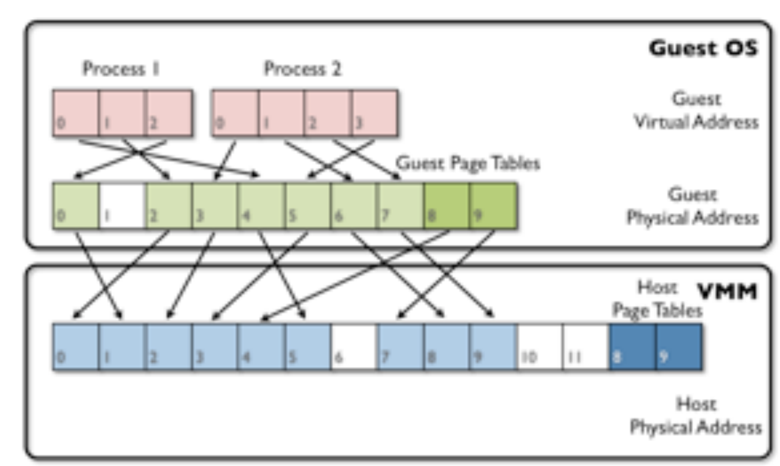
UNIVERSITÀ DI PISA

Nested Page Tables



We need a new specific TLB

TLB misses cost more (2 levels)



ISTITUTO DI SCIENZA E TECNOLOGIE DELL'INFORMAZIONE "A. FAEDO"

Memory Management

- An application uses the OS interfaces to explicitly allocate and deallocate virtual memory during execution
  - malloc and free from GNU Lib C
- In a non virtual environment, the OS assumes it owns all physical memory in the system
- Hardware does not explicitly provide interfaces to allocate and free physical memory
- OS implements its own mechanism to track memory allocations
  - allocated and free lists
- The VMM must implements analogous data structures
  - Allocation is easy via interception of memory accesses
  - Deallocation is hard: free lists can not be intercepted

Memory Reclamation

- The VMM can not reclaim host physical memory when the guest OS frees guest physical memory
- The VMM does not allocate host physical memory on every VM's memory allocation
- The VMM only allocates host physical memory when the VM touches the physical memory that is has never touched before
- The guest OS reuses the same host physical memory for the rest of allocations

VM's "host memory" usage ≤ VM's "guest memory" size + VM "overhead" memory

Memory Overcommitment

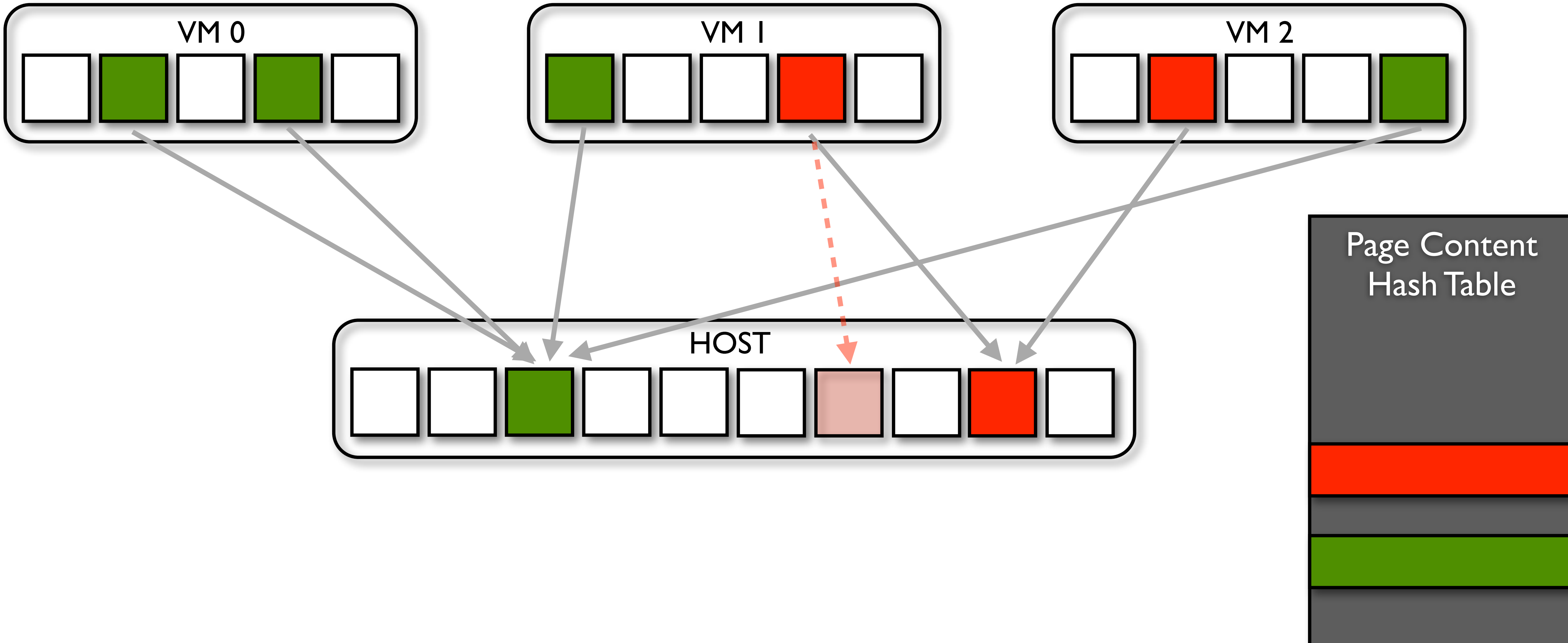
- The VMM must reserve enough host physical memory to back all VM's guest physical memory
  - Plus their overhead memory
- Overcommitment seems not supportable
  - Memory overcommitted if sum of VM memory excesses host memory
- Overcommitment benefits:
  - Higher memory utilization
    - if some VM does not use completely its committed memory, another VM can benefit
  - Higher memory consolidation
    - VMs will have small footprints, so more VMs can be hosted at the same time
- To support memory overcommitment, VMM must be able to reclaim host memory
  - Transparent page sharing
  - Ballooning
  - Host swapping

Transparent Page Sharing

- Some VMs can have identical sets of memory content
  - Several VMs running the same OS
  - Several VMs executing the same applications
  - Several VMs accessing the same user data
- Reduce memory occupation by reclaiming memory copies

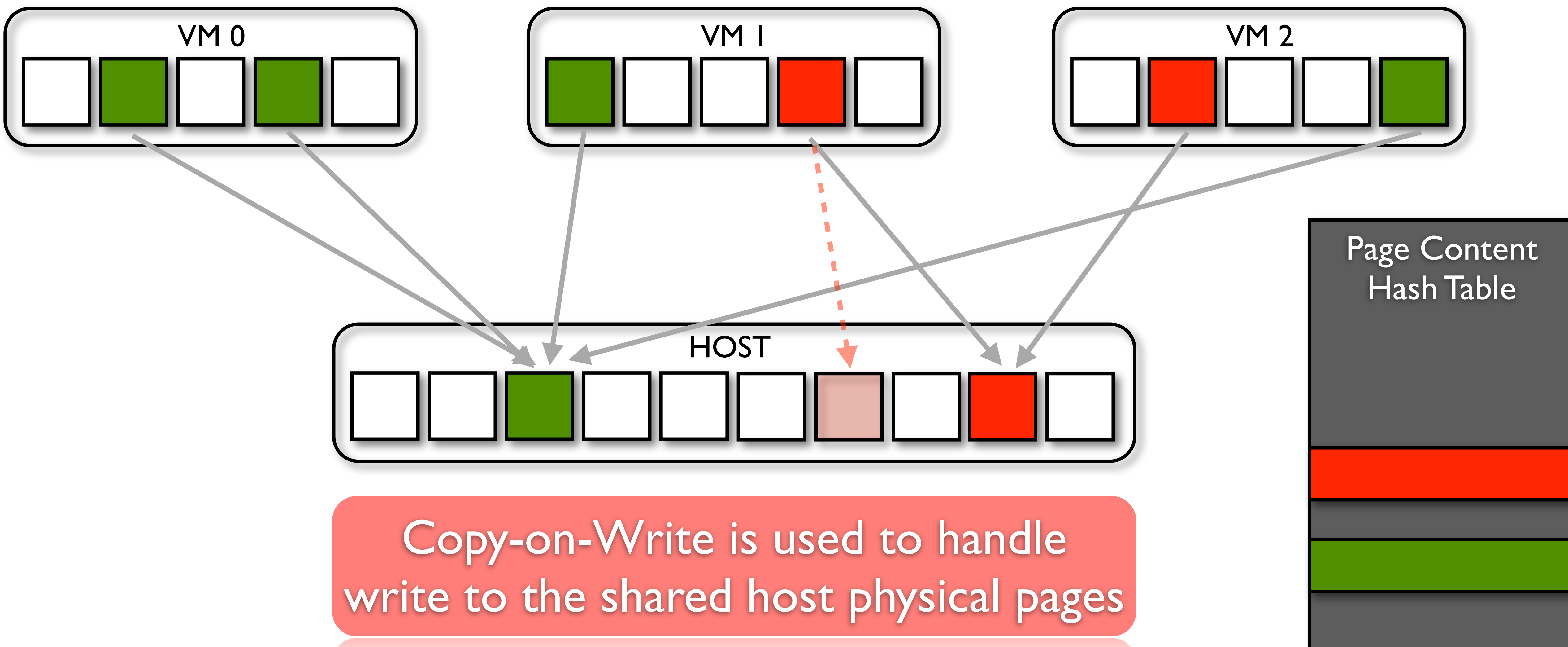
Transparent Page Sharing

- Some VMs can have identical sets of memory content
  - Several VMs running the same OS
  - Several VMs executing the same applications
  - Several VMs accessing the same user data
- Reduce memory occupation by reclaiming memory copies



Transparent Page Sharing

- Some VMs can have identical sets of memory content
  - Several VMs running the same OS
  - Several VMs executing the same applications
  - Several VMs accessing the same user data
- Reduce memory occupation by reclaiming memory copies



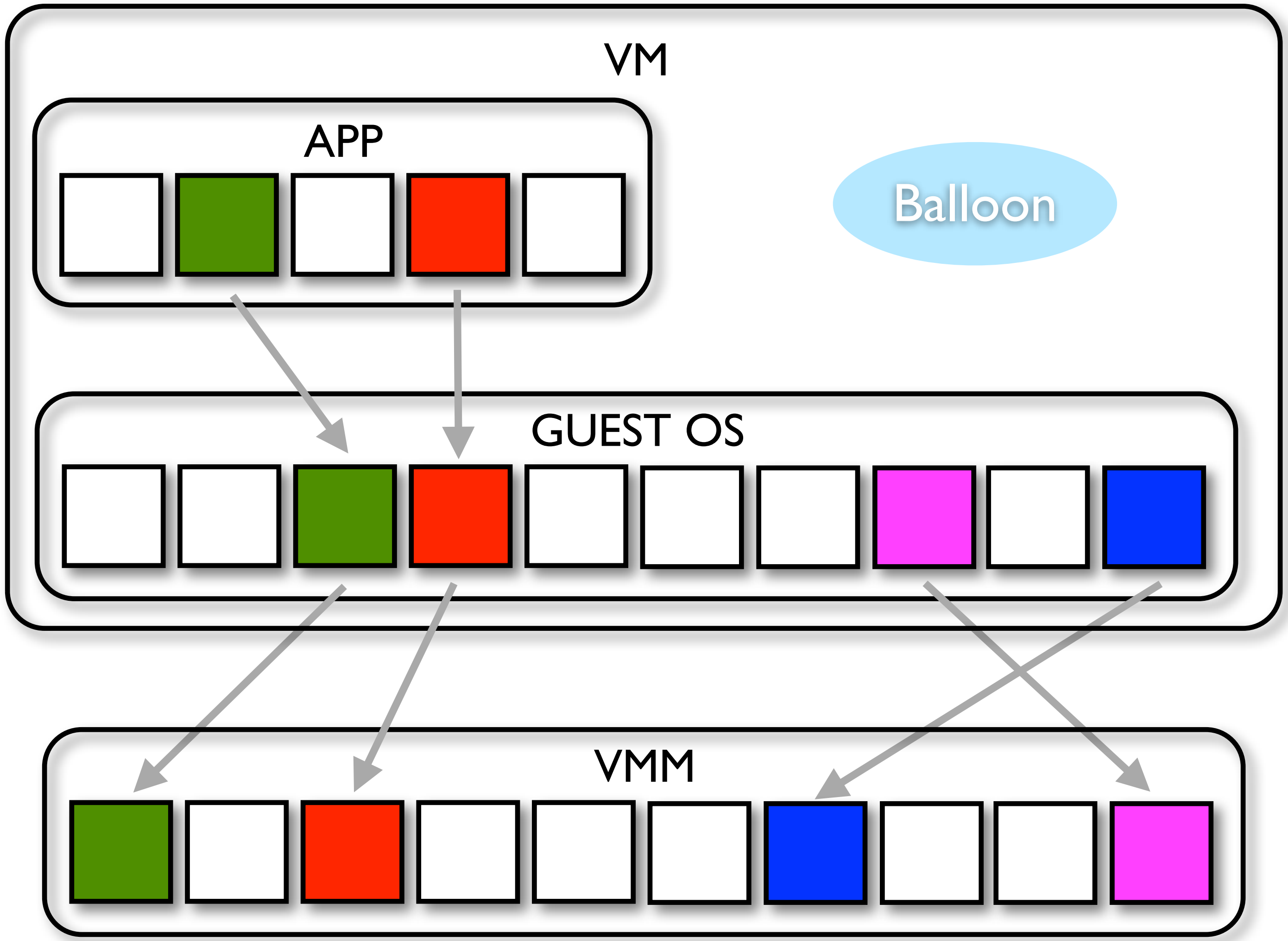
Copy-on-Write is used to handle write to the shared host physical pages

Ballooning

- Guest OS is not aware of the host memory status
  - In particular, it does not free memory if host is running out of it
- A pseudo driver is installed in each guest OS
  - Balloon driver
  - No exposed interfaces to the guest OS
  - Privately communicates with the VMM only
  - It requires memory allocation, depending on its "size"
- If the VMM requires two pages, it sets the balloon size to two pages
- After allocation, these two pages are "pinned"
  - Guest OS assures pinned page will never be flushed to disk
- After pinning, the VMM can safely reclaim the respective host physical pages
  - Nobody actually relies on the content (read or write)
- If the balloon deflates, it will release the "pins"

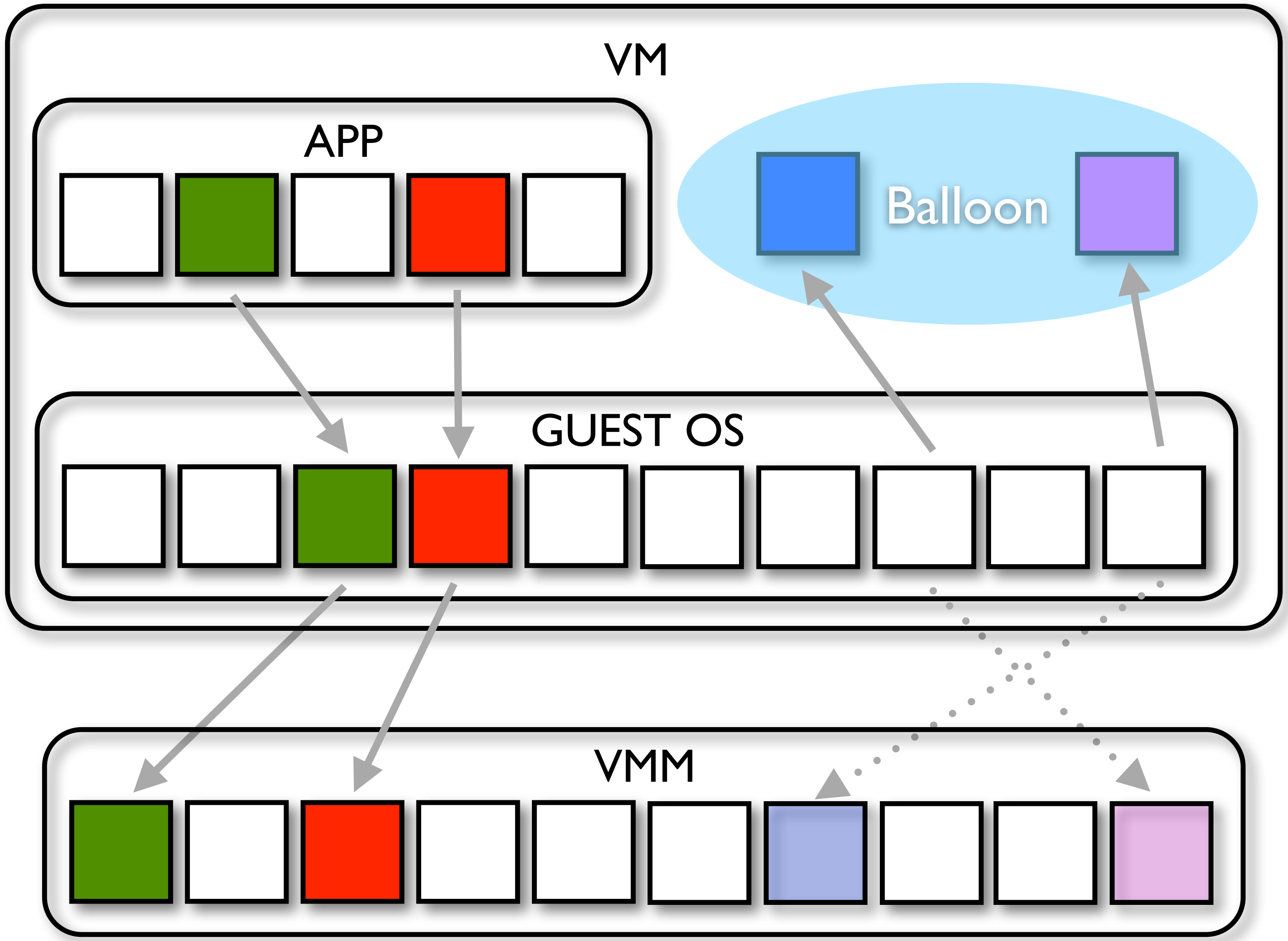
Ballooning

- Guest OS is not aware of the host memory status
  - In particular, it does not free memory if host is running out of it
- A pseudo driver is installed in each guest OS
  - Balloon driver
  - No exposed interfaces to the guest OS
  - Privately communicates with the VMM only
  - It requires memory allocation, depending on its "size"



Ballooning

- Guest OS is not aware of the host memory status
  - In particular, it does not free memory if host is running out of it
- A pseudo driver is installed in each guest OS
  - Balloon driver
  - No exposed interfaces to the guest OS
  - Privately communicates with the VMM only
  - It requires memory allocation, depending on its "size"



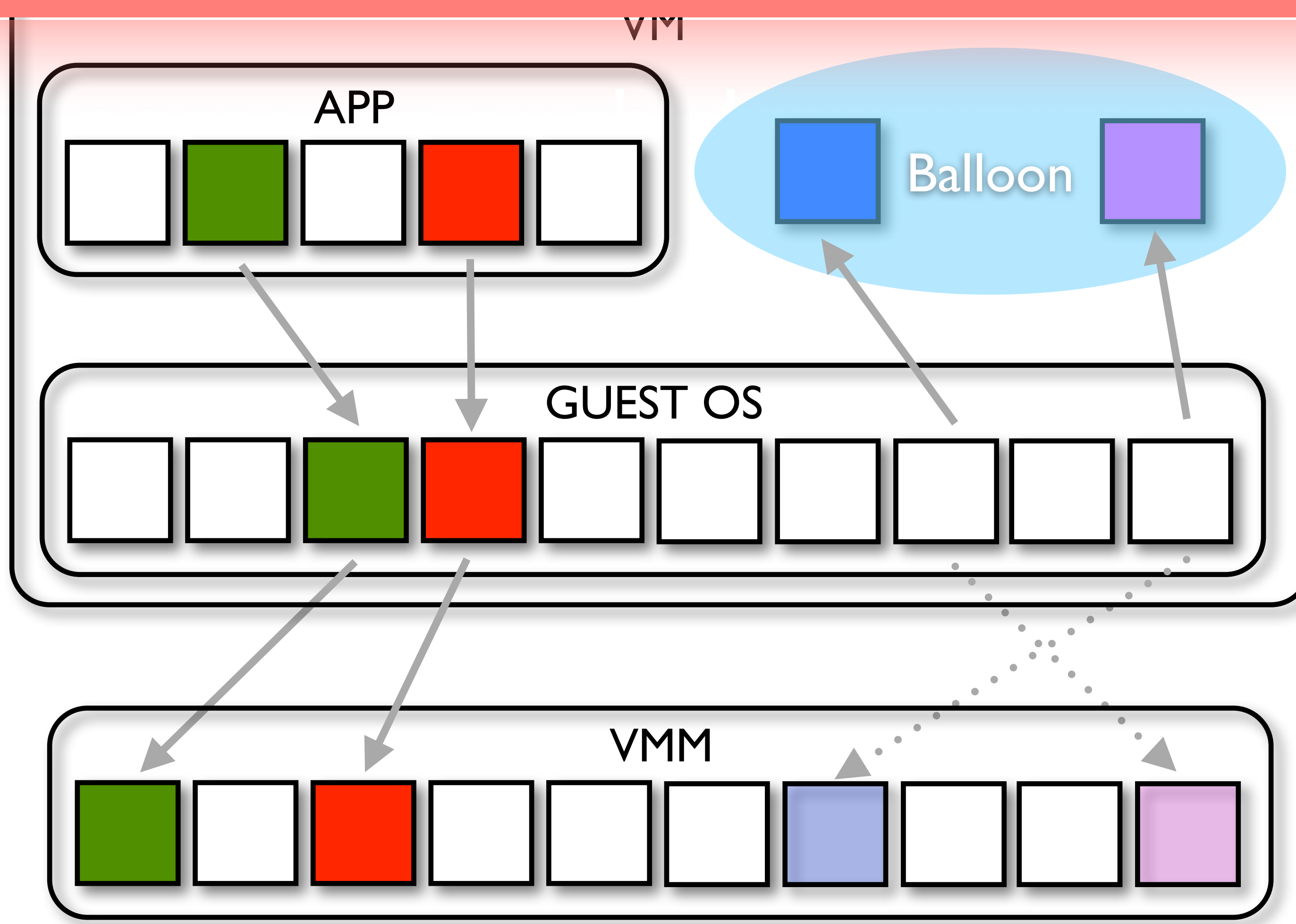
ISTITUTO DI SCIENZA E TECNOLOGIE DELL'INFORMAZIONE "A. FAEDO"

lunedì 17 dicembre 12

Ballooning

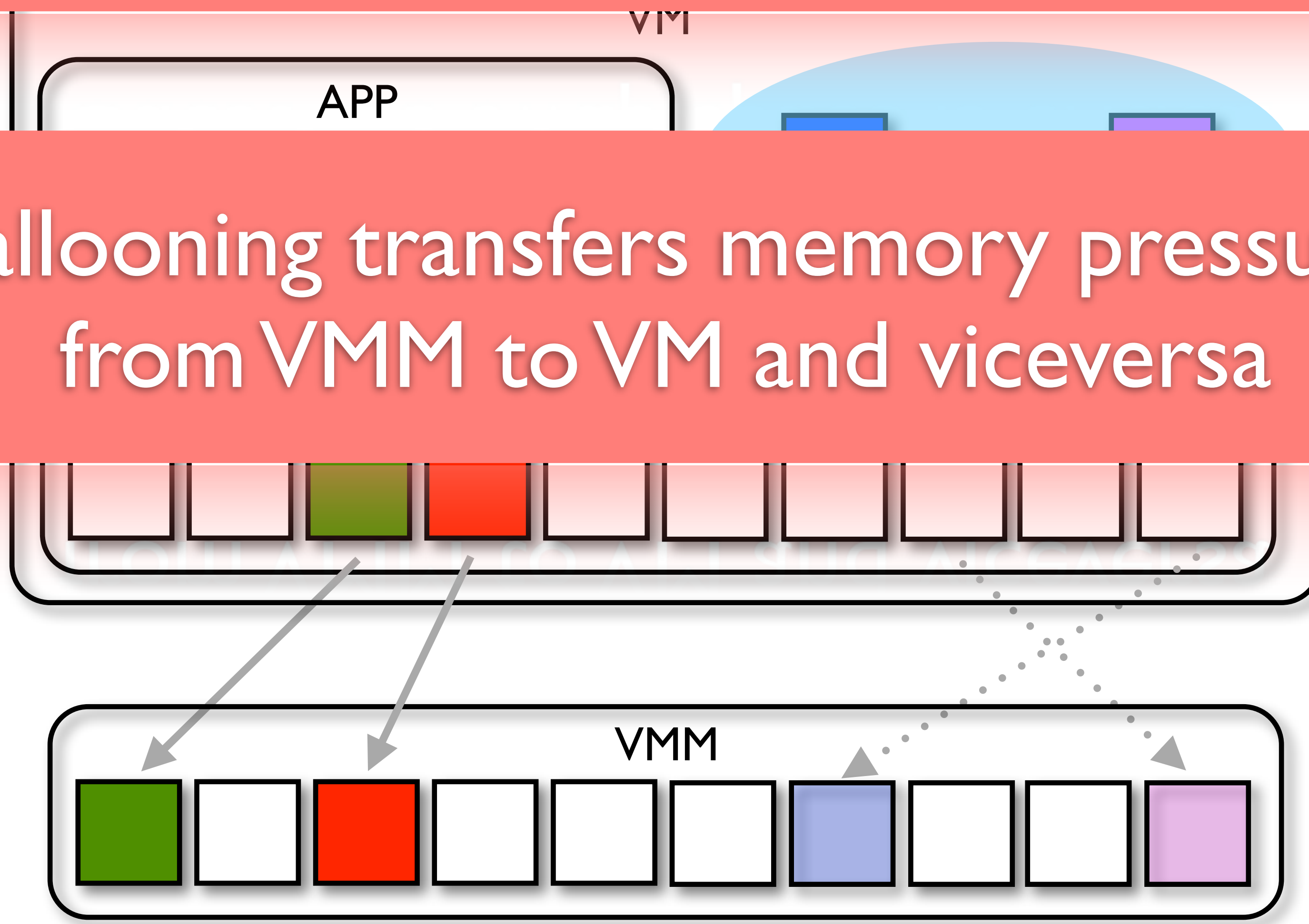
- Guest OS is not aware of the host memory status
  - In particular, it does not free memory if host is running out of it
- A pseudo driver is installed in each guest OS
  - Balloon driver
  - No exposed interfaces to the guest OS
  - Privately commu
  - It requires memo

Guest OS swap space is critical



- Guest OS is not aware of the host memory status
  - In particular, it does not free memory if host is running out of it
- A pseudo driver is installed in each guest OS
  - Balloon driver
  - No exposed interfaces to the guest OS
  - Privately commu
  - It requires memo

Guest OS swap space is critical



Ballooning transfers memory pressure from VMM to VM and viceversa

Host swapping

- Transparent page sharing and ballooning have performance impacts on the VMM
- Host swapping is used if VMM performance is critical
  - When a VM is started, the VMM creates a separate swap file for the virtual machine
  - When necessary, the VMM can swap out the guest memory to its swap file
- The VMM performance is guaranteed
- The VM performance is severely degraded
- Double paging problem
  - Assume the hypervisor swaps out a guest physical page
  - It is possible that the guest OS system pages out the same physical page
    - If the guest is also under memory pressure
  - This causes the page to be swapped in from the hypervisor swap device and immediately to be paged out to the virtual machine's virtual swap device.
  - Note that it is impossible to find an algorithm to handle all these pathological cases properly
- Due to the potential high performance penalty for VMs, host swapping is the last resort to reclaim memory from a VM

References

- Performance Evaluation of Intel EPT Hardware Assist
  - http://www.vmware.com/pdf/Perf_ESX_Intel-EPT-eval.pdf
- AMD-V™ Nested Paging
  - http://developer.amd.com/assets/NPT-WP-1%201-final-TM.pdf
- The x86 kvm shadow mmu
  - http://www.mjmwired.net/kernel/Documentation/kvm/mmu.txt
- What Every Programmer Should Know About Memory
  - http://www.akkadia.org/drepper/cpumemory.pdf
- Understanding Memory Resource Management in VMware® ESX™ Server
  - http://www.vmware.com/files/pdf/perf-vsphere-memory_management.pdf