E6998 - Virtual Machines
Lecture 3
Memory Virtualization

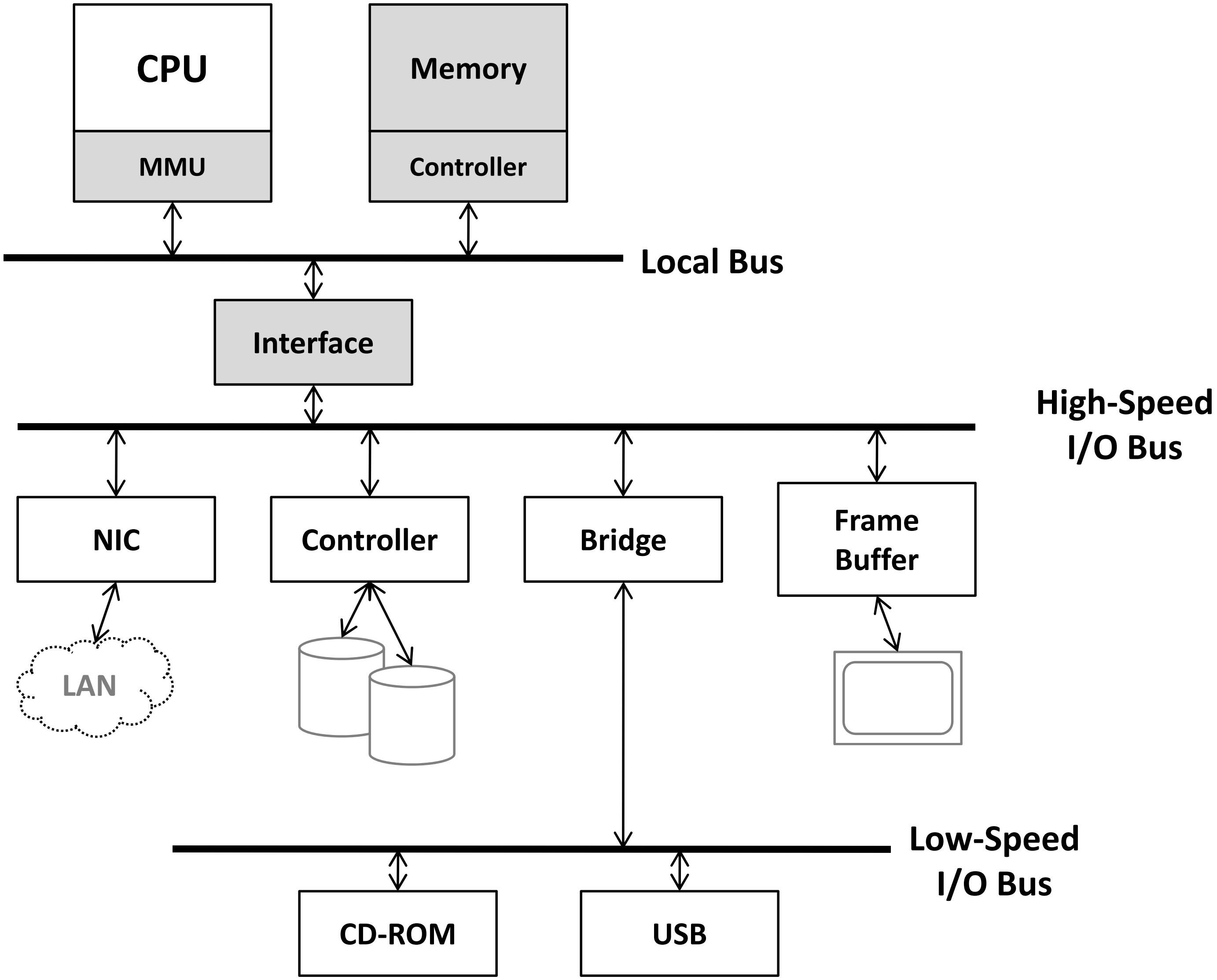
Scott Devine

VMware, Inc.

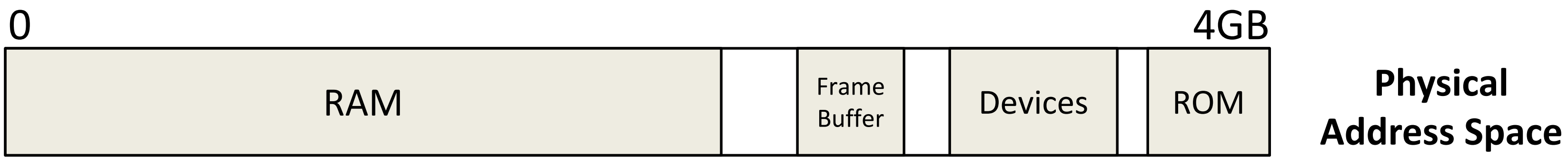
Outline

- **Background**
- **Virtualization Techniques**
  - Emulated TLB
  - Shadow Page Tables
- **Page Protection**
  - Memory Tracing
  - Hiding the Monitor
- **Hardware-supported Memory Virtualization**
  - Nested Page Tables

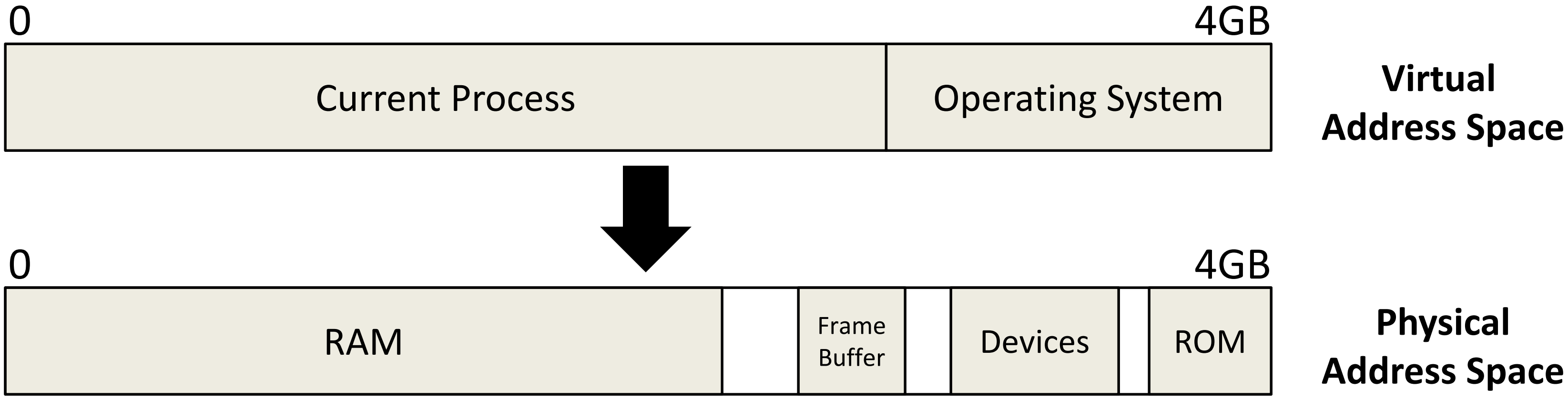
Computer System Organization



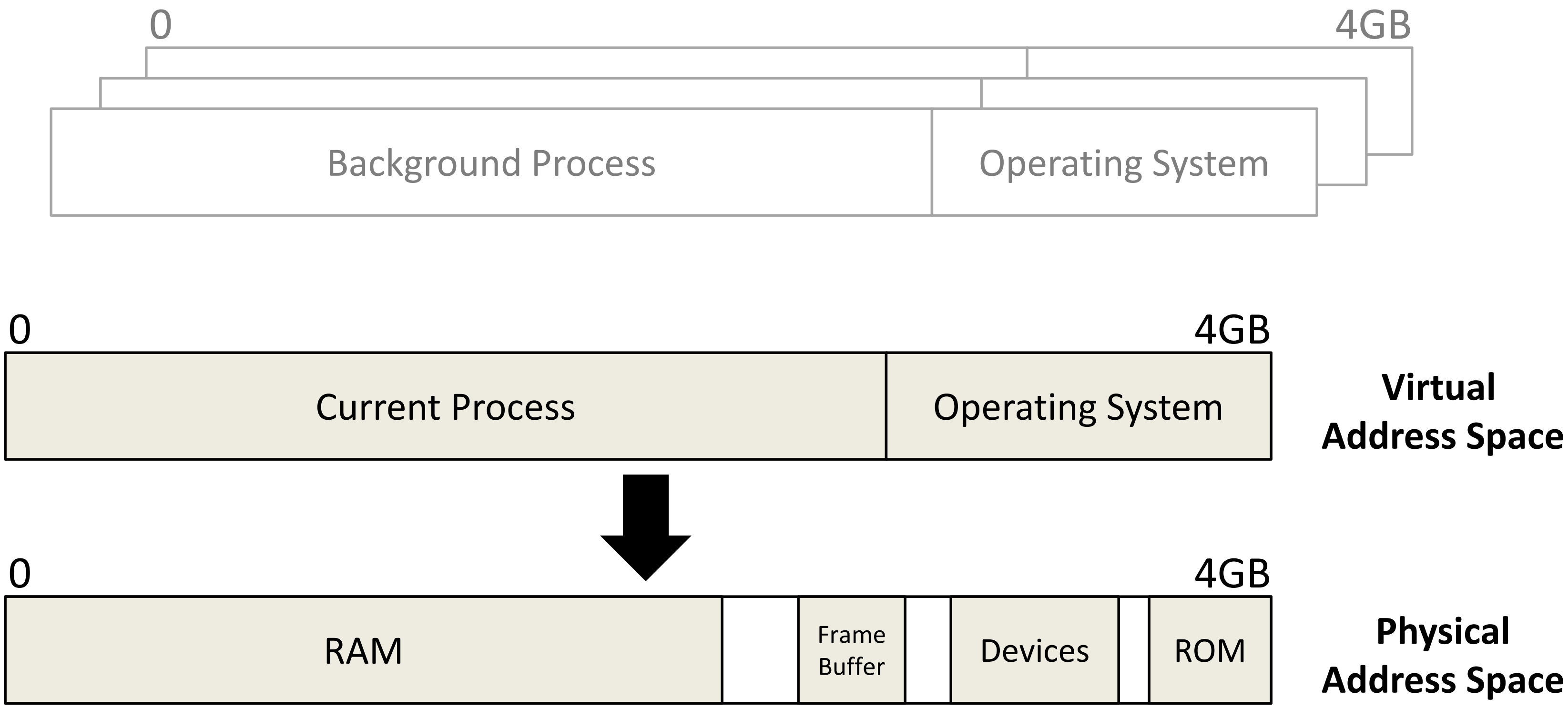
Traditional Address Spaces



Traditional Address Spaces



Traditional Address Spaces



Memory Management Unit (MMU)

- **Virtual Address to Physical Address Translation**
  - Works in fixed-sized pages
  - Page Protection
- **Translation Look-aside Buffer**
  - TLB caches recently used Virtual to Physical mappings
- **Control registers**
  - Page Table location
  - Current ASID
  - Alignment checking

Types of MMUs

- **Architected Page Tables**

  x86, x86-64, ARM, IBM System/370, PowerPC

  – Hardware defines page table layout
  – Hardware walks page table on TLB miss
- **Architected TLBs**

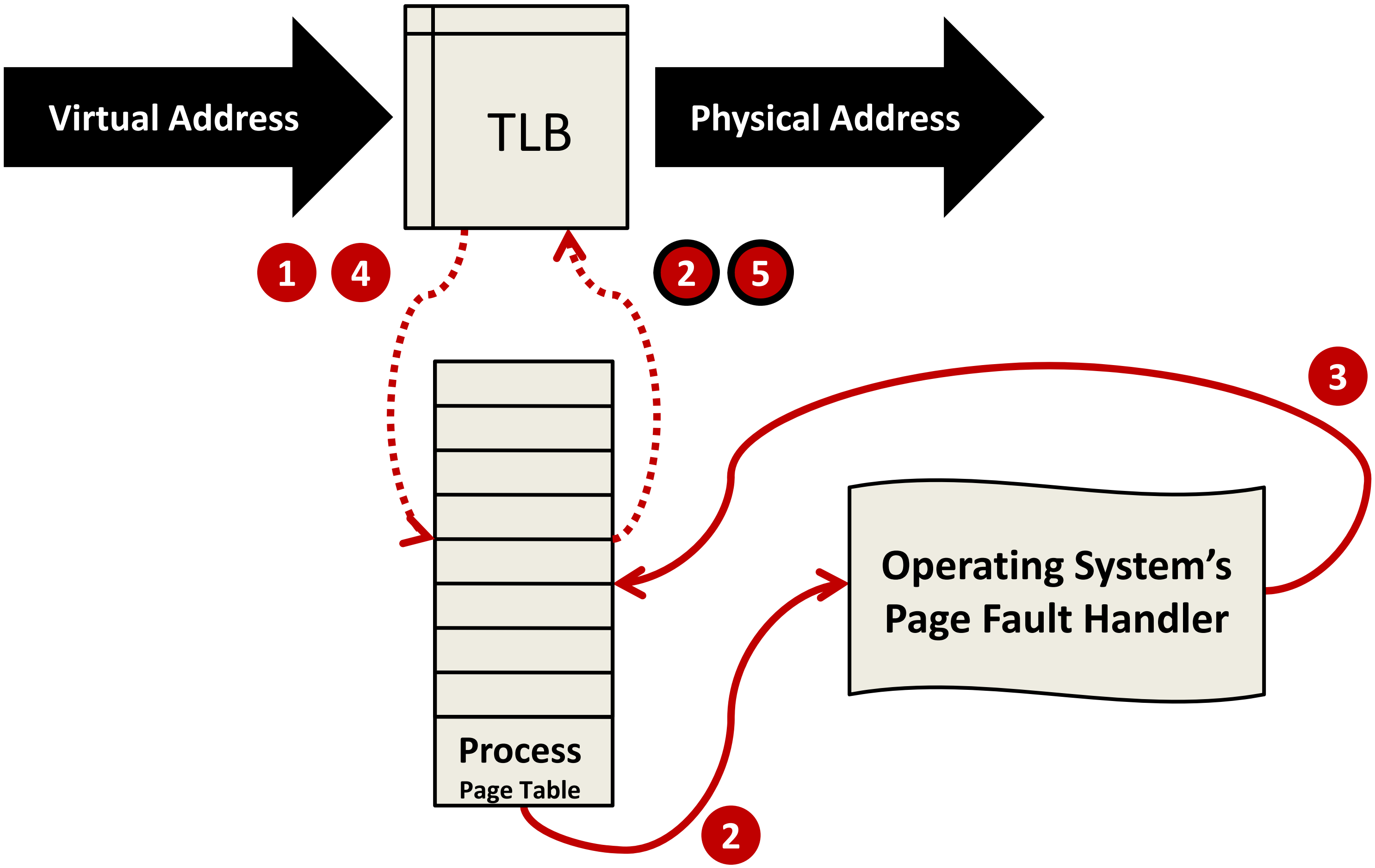
  MIPS, SPARC, Alpha

  – Hardware defines the interface to TLB
  – Software reloads TLB on misses
  – Page table layout free to software
- **Segmentation / No MMU**

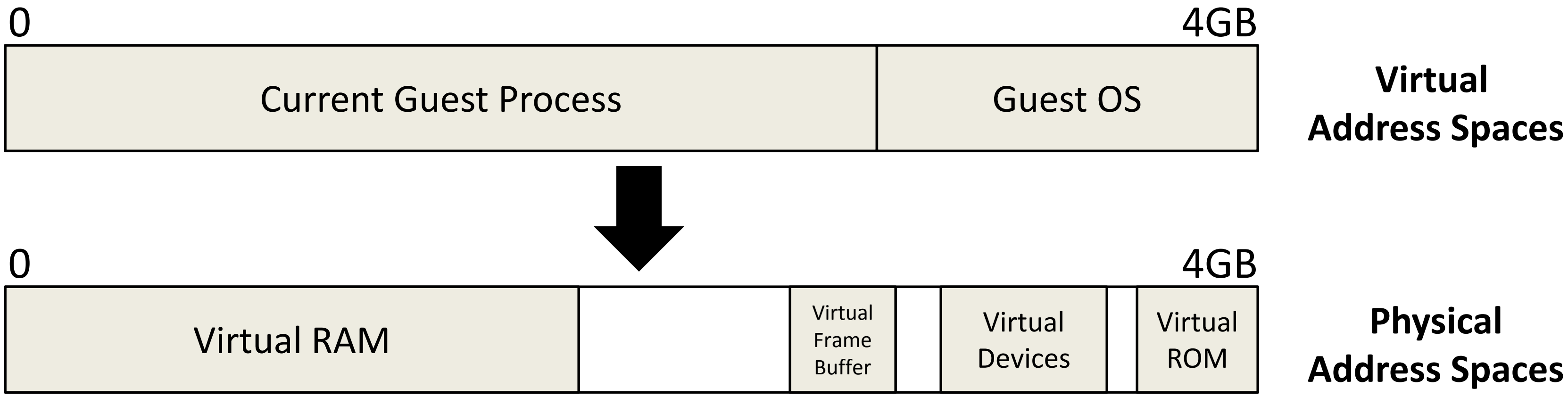
  Low-end ARMs, micro-controllers

  – Para-virtualization required

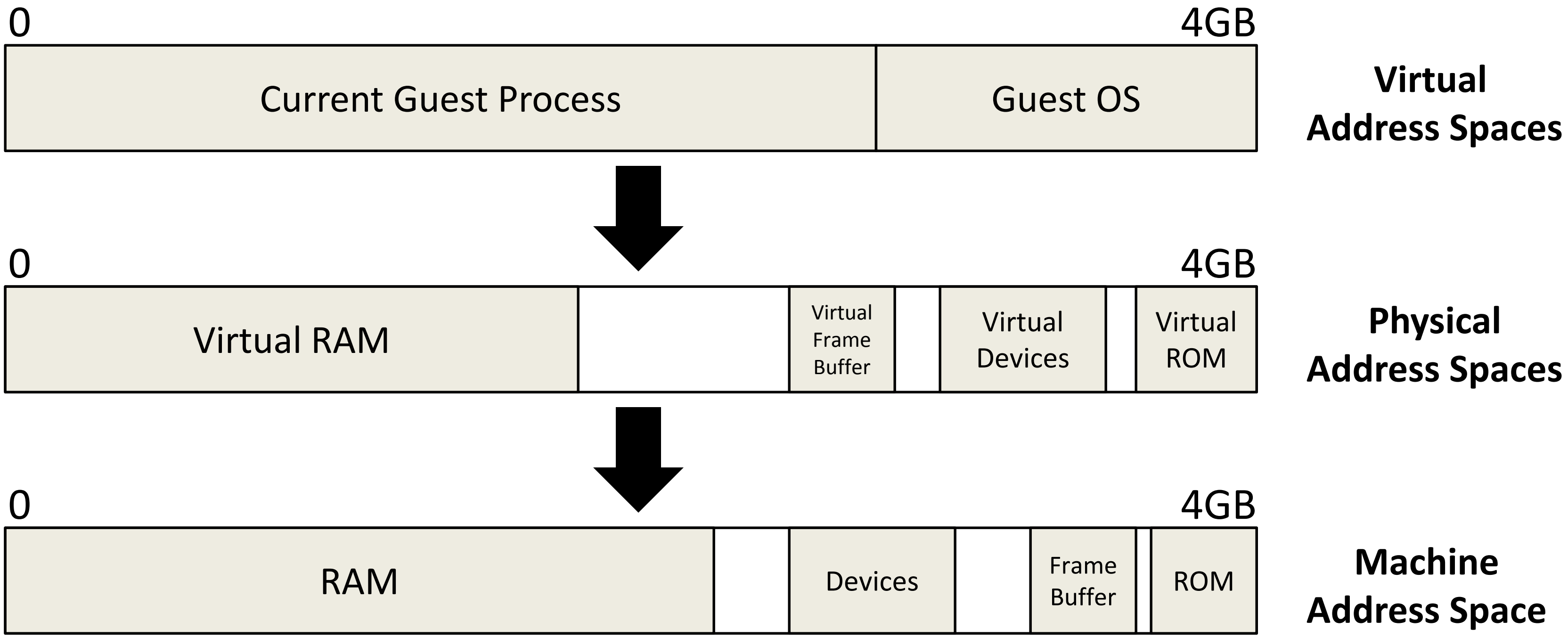
Traditional Address Translation w/ Architected Page Tables



Virtualized Address Spaces



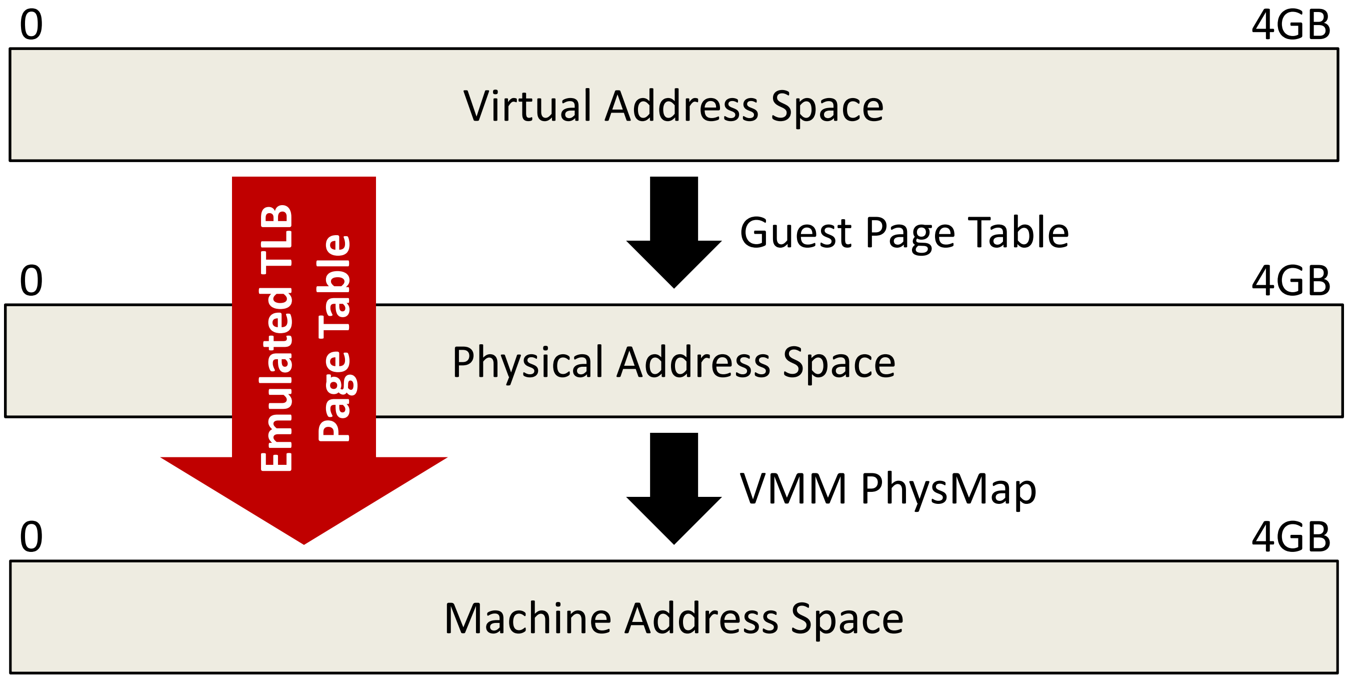
Virtualized Address Spaces



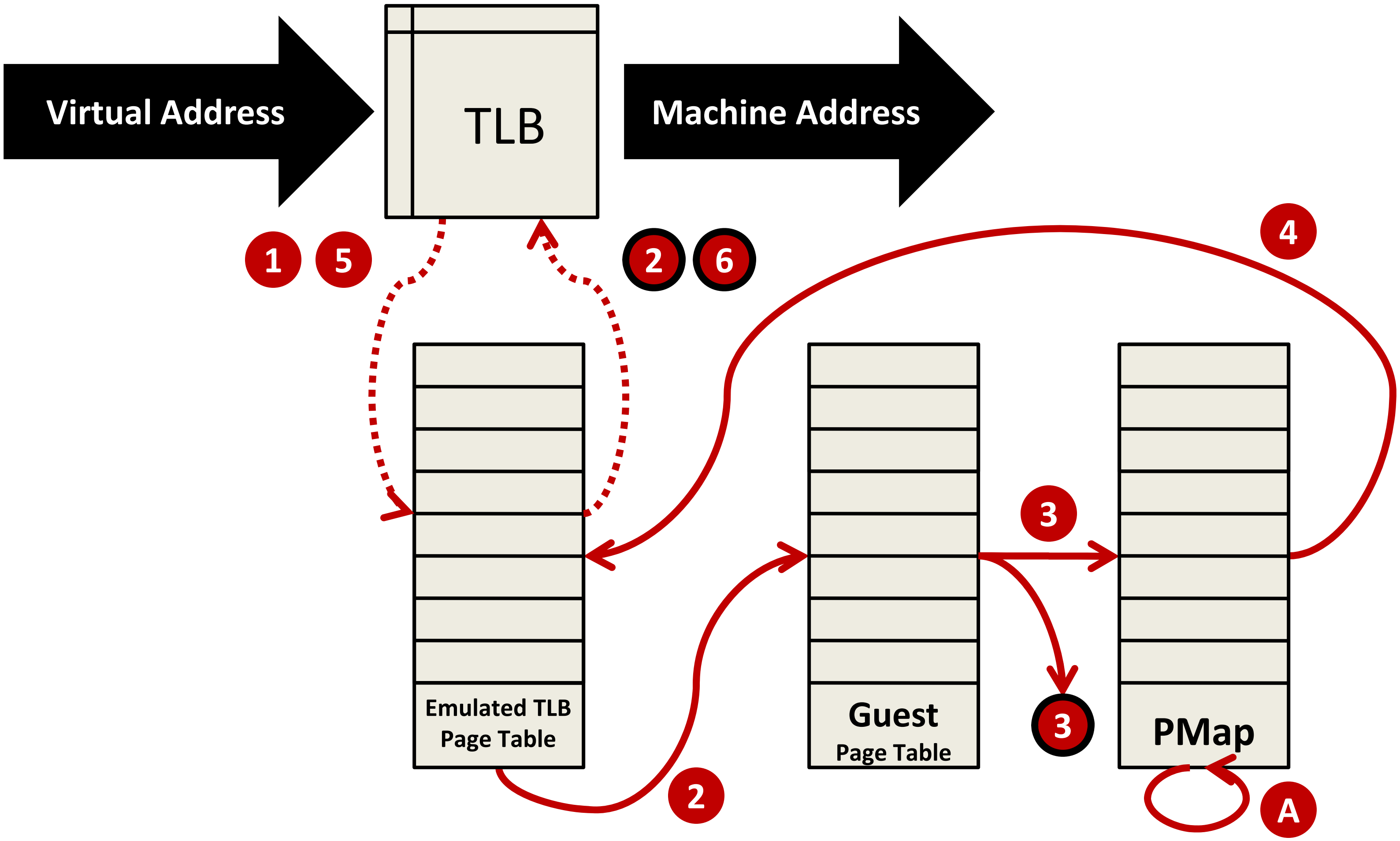
Outline

- **Background**
- **Virtualization Techniques**
  - Emulated TLB
  - Shadow Page Tables
- **Page Protection**
  - Memory Tracing
  - Hiding the Monitor
- **Hardware-supported Memory Virtualization**
  - Nested Page Tables

Virtualized Address Spaces w/ Emulated TLB



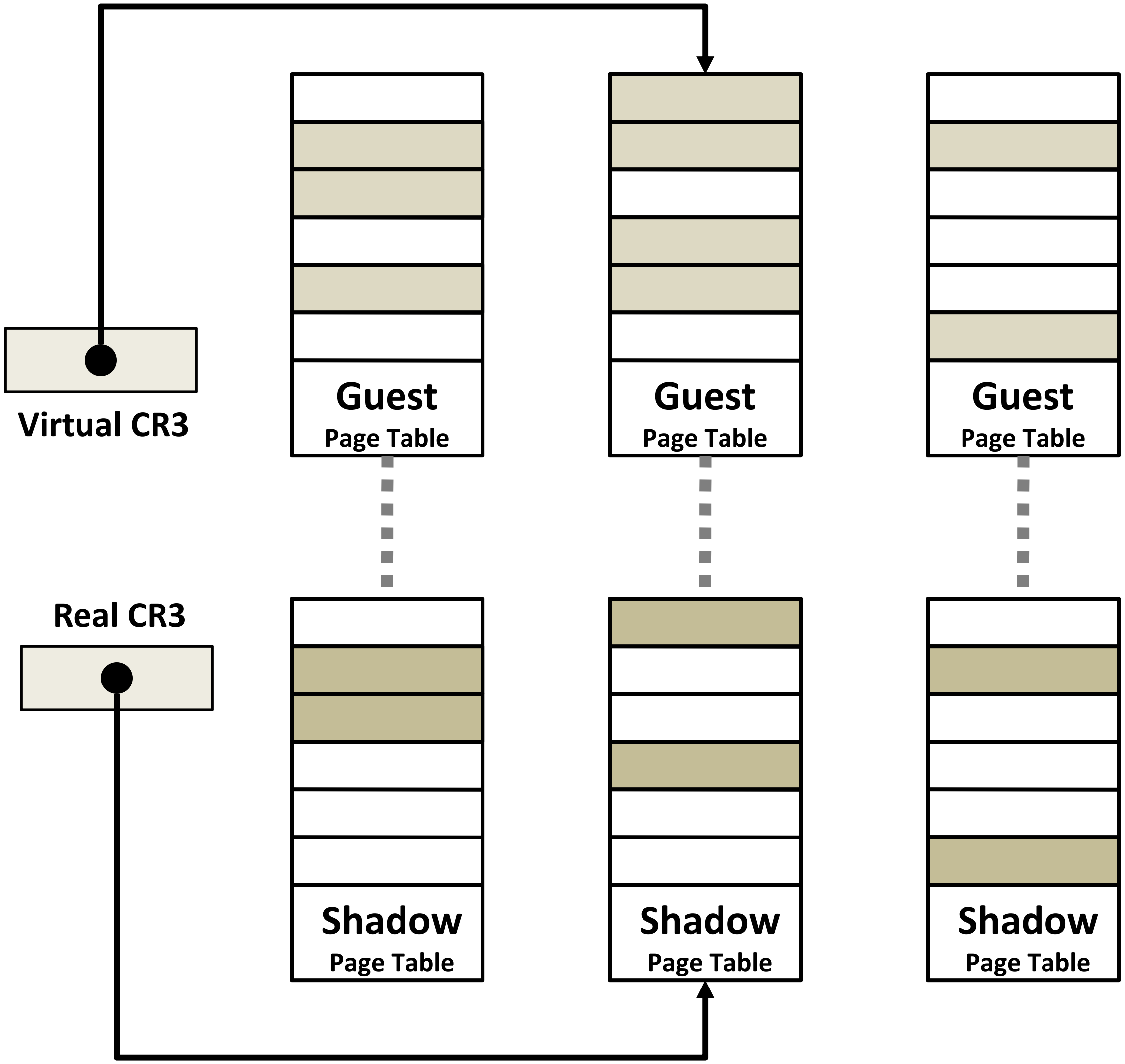
Virtualized Address Translation w/ Emulated TLB



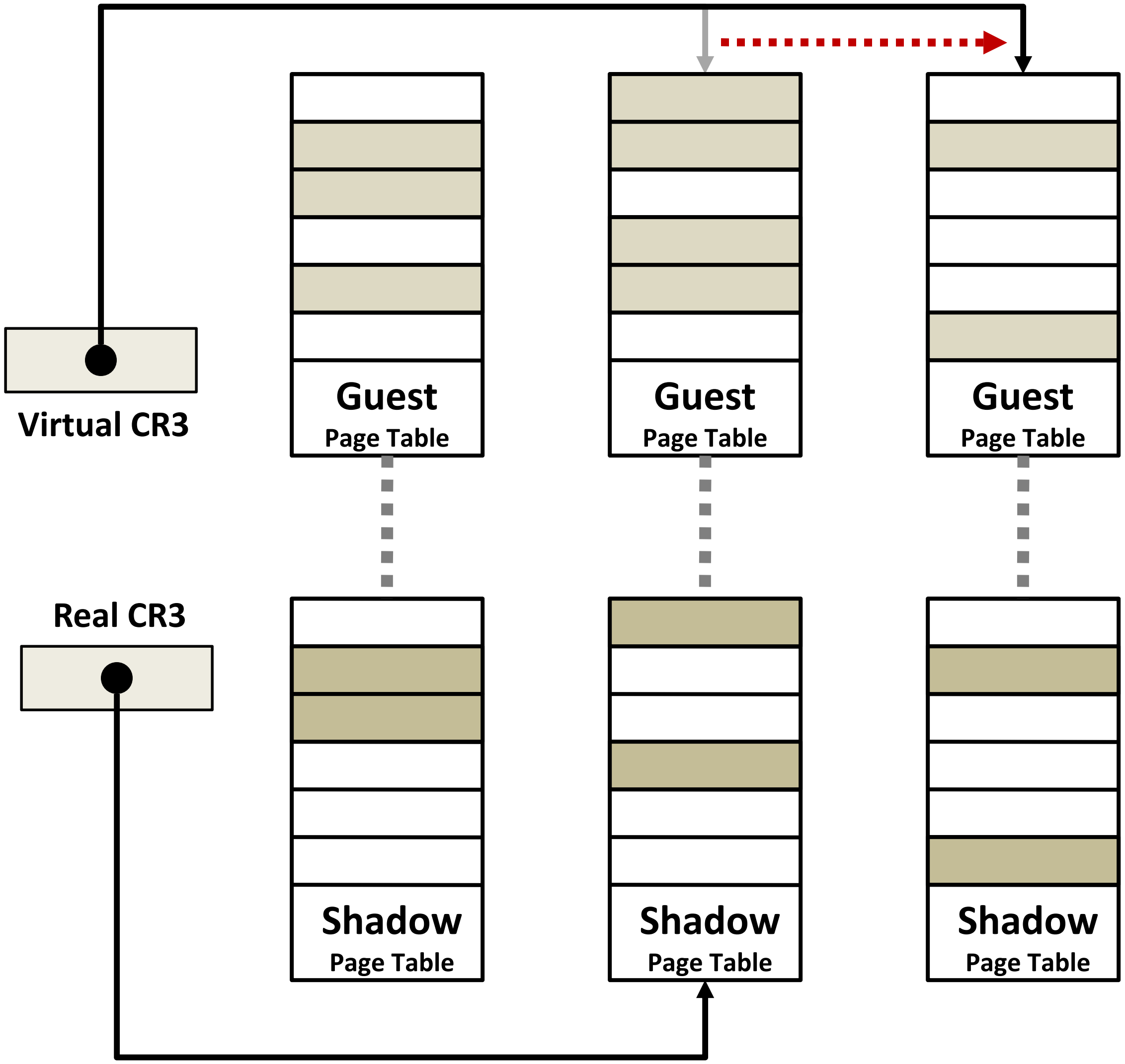
Issues with Emulated TLBs

- **Guest page table consistency**
  - Rely on Guest’s need to invalidate TLB
  - Guest TLB invalidations caught by monitor, emulated
- **Performance**
  - Guest context switches flush entire software TLB

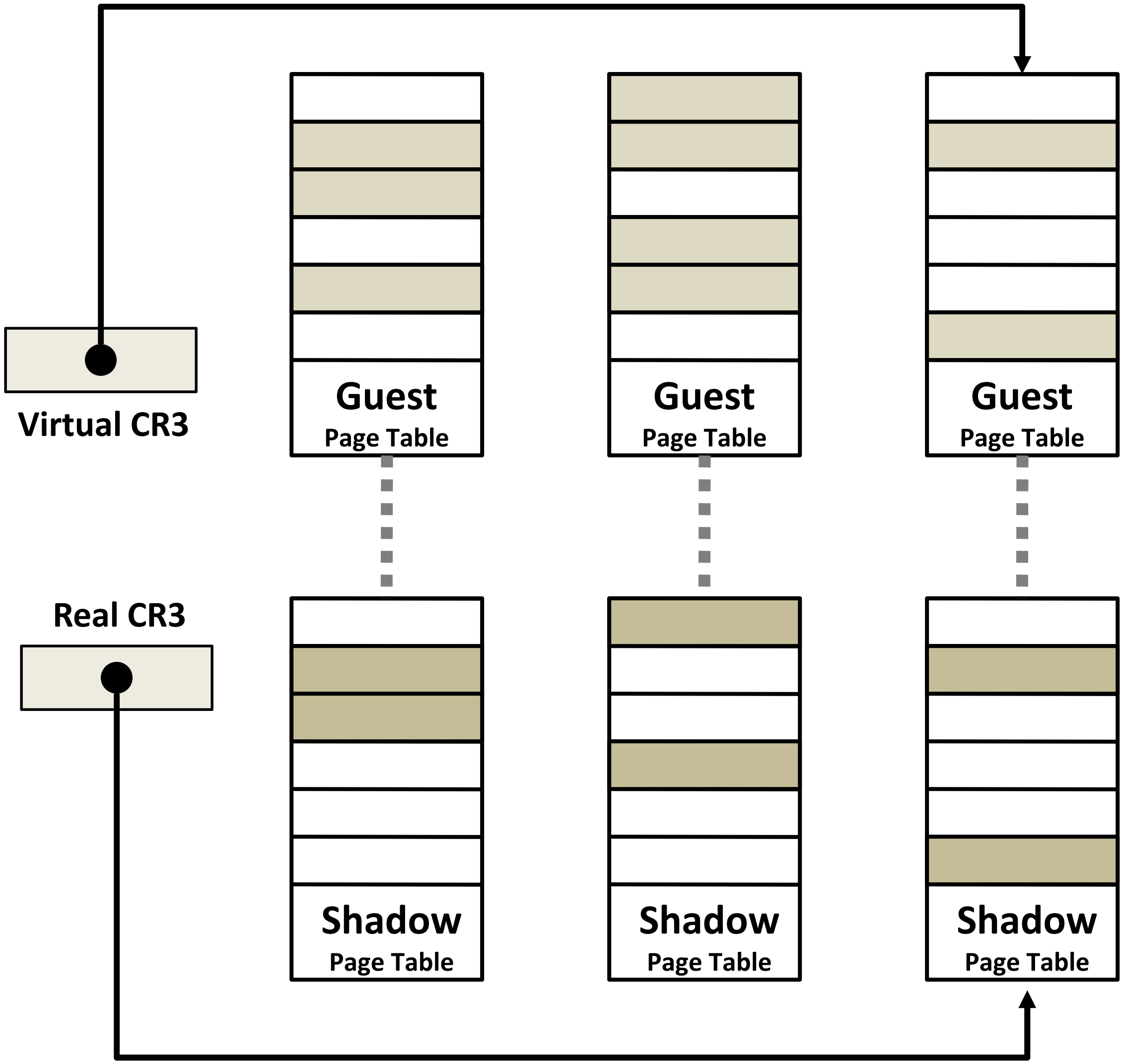
Shadow Page Tables



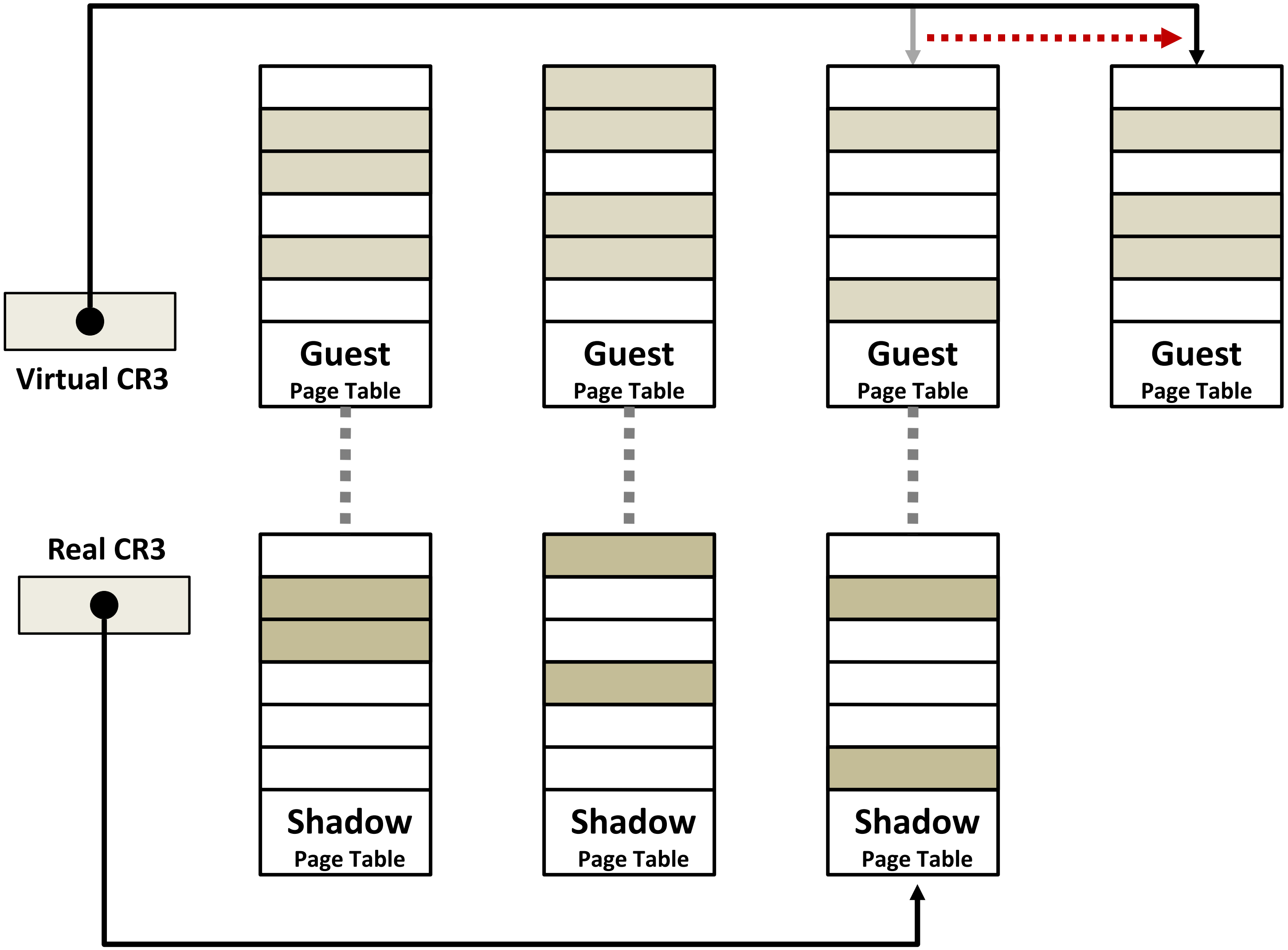
Guest Write to CR3



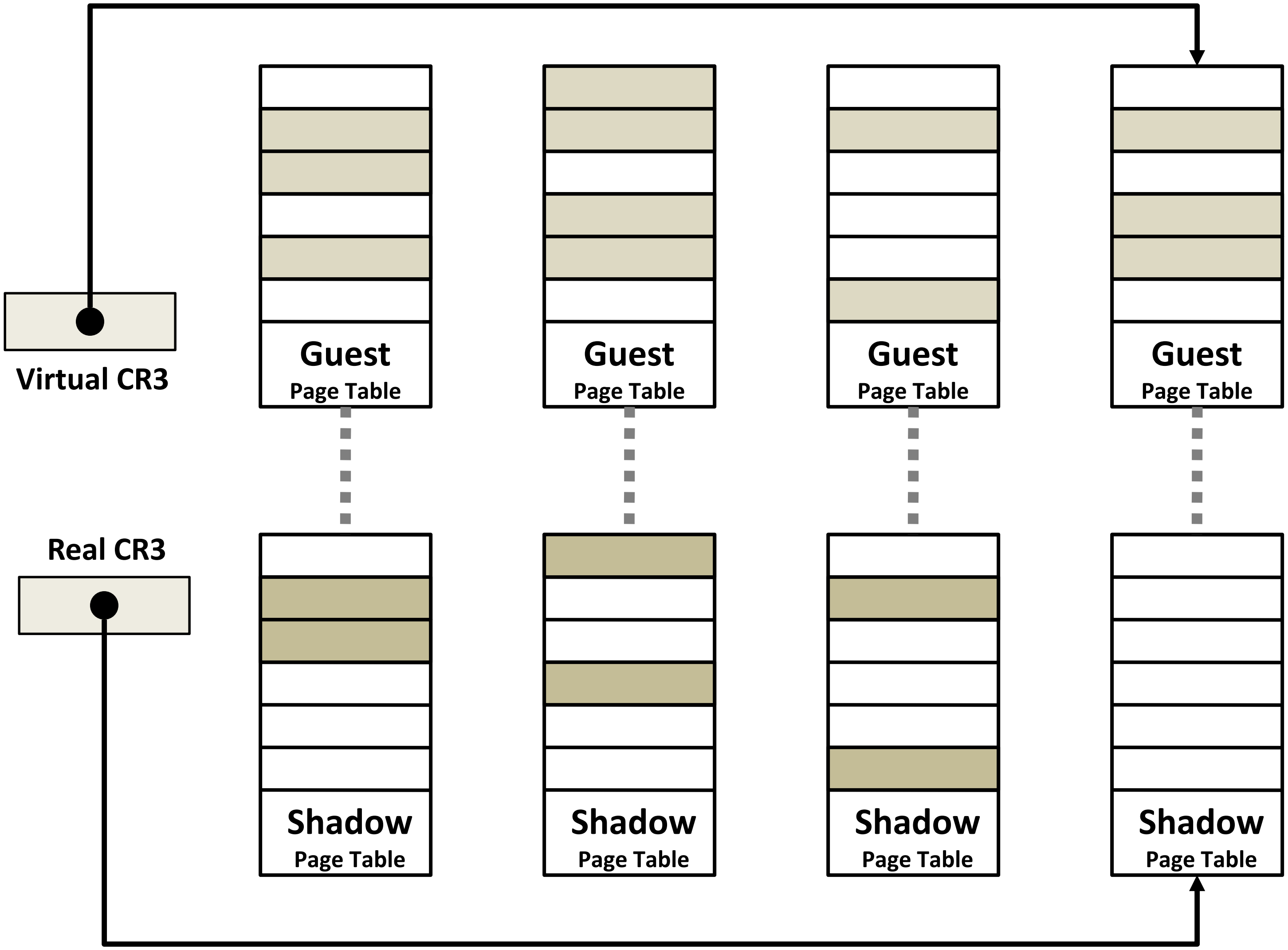
Guest Write to CR3



Undiscovered Guest Page Table



Undiscovered Guest Page Table



Issues with Shadow Page Tables

- **Positives**
  - Handle page faults in same way as Emulated TLBs
  - Fast guest context switching
- **Page Table Consistency**
  - Guest may not need invalidate TLB on writes to off-line page tables
  - Need to trace writes to shadow page tables to invalidate entries
- **Memory Bloat**
  - Caching guest page tables takes memory
  - Need to determine when guest has reused page tables

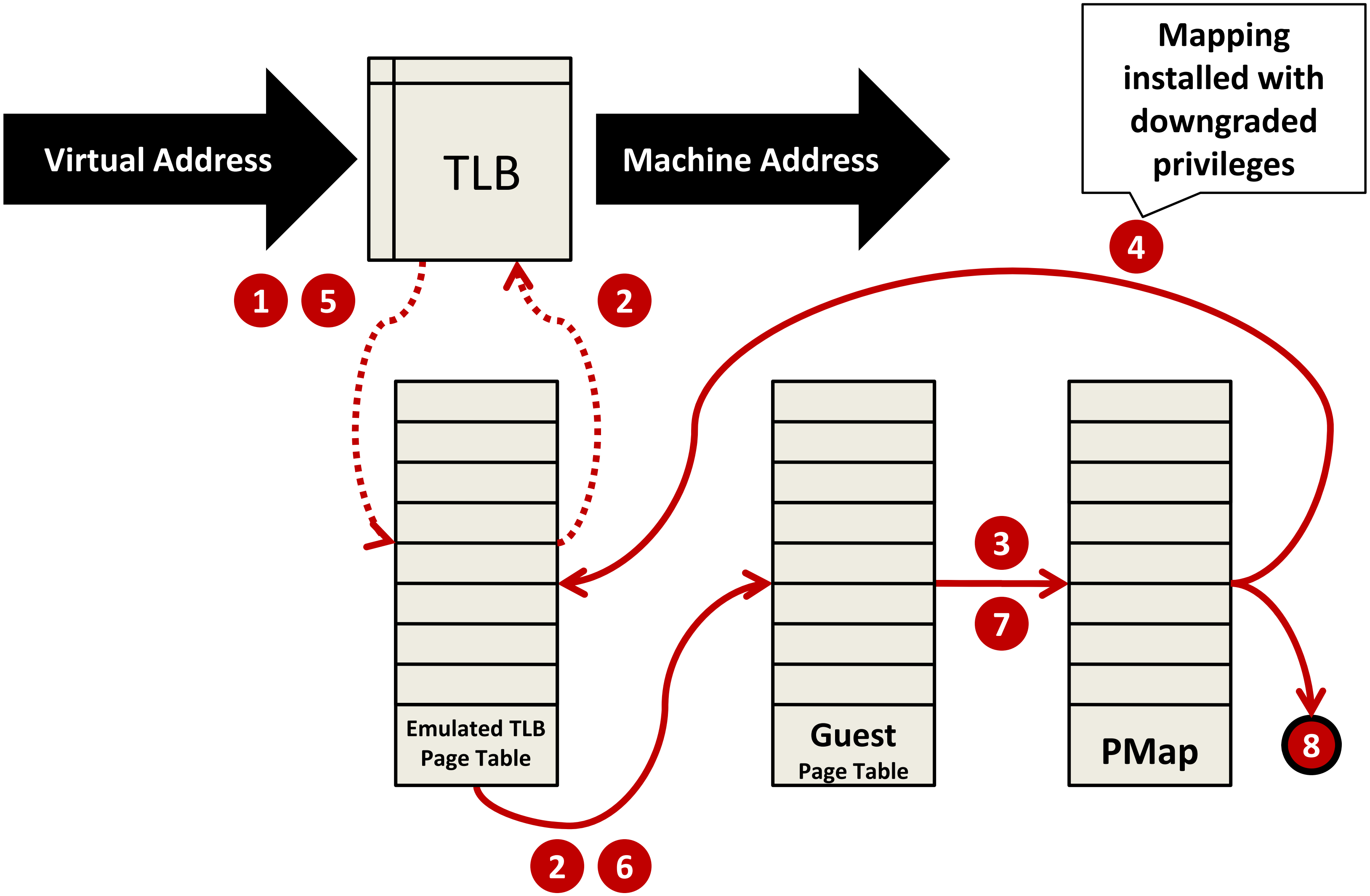
Memory Tracing

- **Call a monitor handler on access to a traced page**
  - Before guest reads
  - After guest writes
  - Before guest writes
- **Modules can install traces and register for callbacks**
  - Binary Translator for cache consistency
  - Shadow Page Tables for cache consistency
  - Devices
    - Memory-mapped I/O, Frame buffer
  - ROM
  - COW

Memory Tracing (cont.)

- **Traces installed on Physical Pages**
  - Need to know if data on page has changed regardless of what virtual address it was written through
- **Use Page Protection to cause traps on traced pages**
  - Downgrade protection
    - Write traced pages downgrade to read-only
    - Read traced pages downgrade to invalid

Trace Callout Path



Hiding the Monitor

- **Monitor must be in the Virtual Address space**
  - Exception / Interrupt handlers
  - Binary Translator
    - Translation Cache
    - Callout glue code
    - Register spill / fill locations
    - Emulated control registers

Hiding the Monitor
Options for Trap-and-Emulate

- **Address space switch on Exceptions / Interrupts**
    - Must be supported by the hardware
- **Occupy some space in guest virtual address space**
    - Need to protect monitor from guest accesses
        - Use page protection
    - Need to emulate guest accesses to monitor ranges
        - Manually translate guest virtual to machine
        - Emulate instruction
            - Must be able to handle all memory accessing instructions

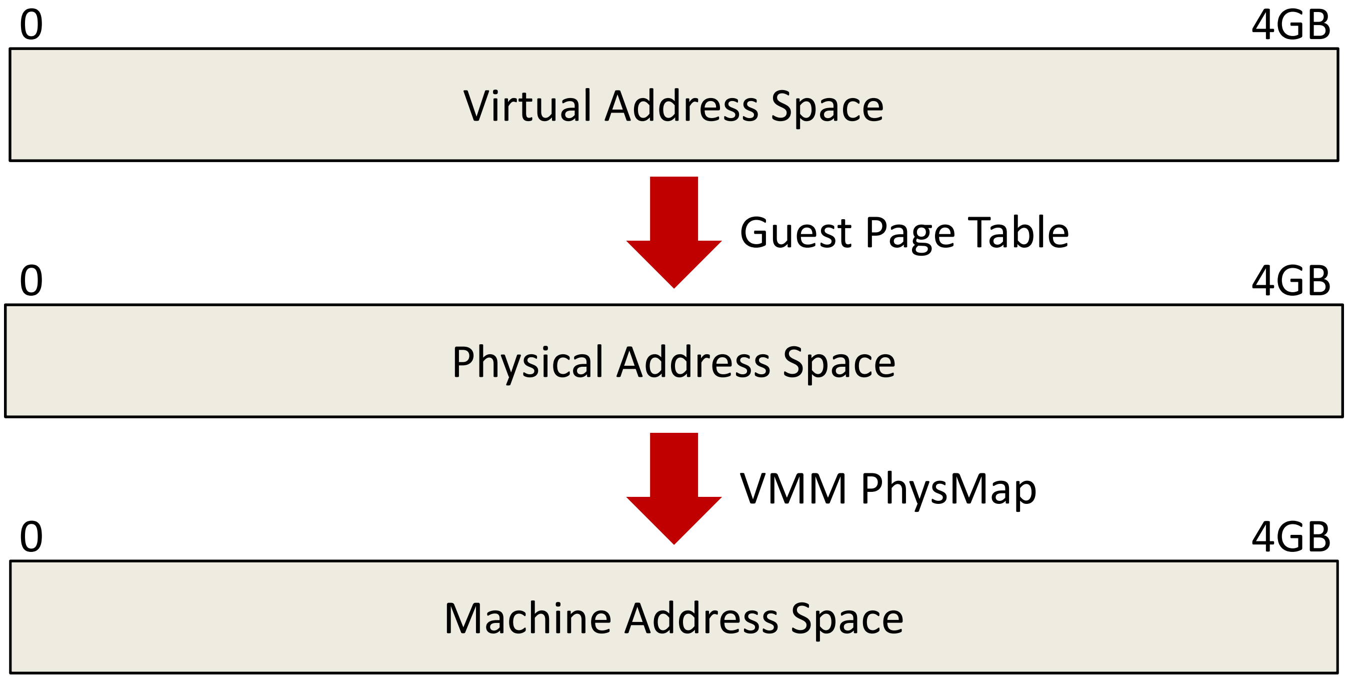
Hiding the Monitor
Options for Binary Translation

- **Translation cache intermingles guest and monitor memory accesses**
  - Need to distinguish these accesses
  - Monitor accesses have full privileges
  - Guest accesses have lesser privileges
- **On x86 can use segmentation**
  - Monitor lives in high memory
  - Guest segments truncated to allow no access to monitor
  - Binary translator uses guest segments for guest accesses and monitor segments for monitor accesses

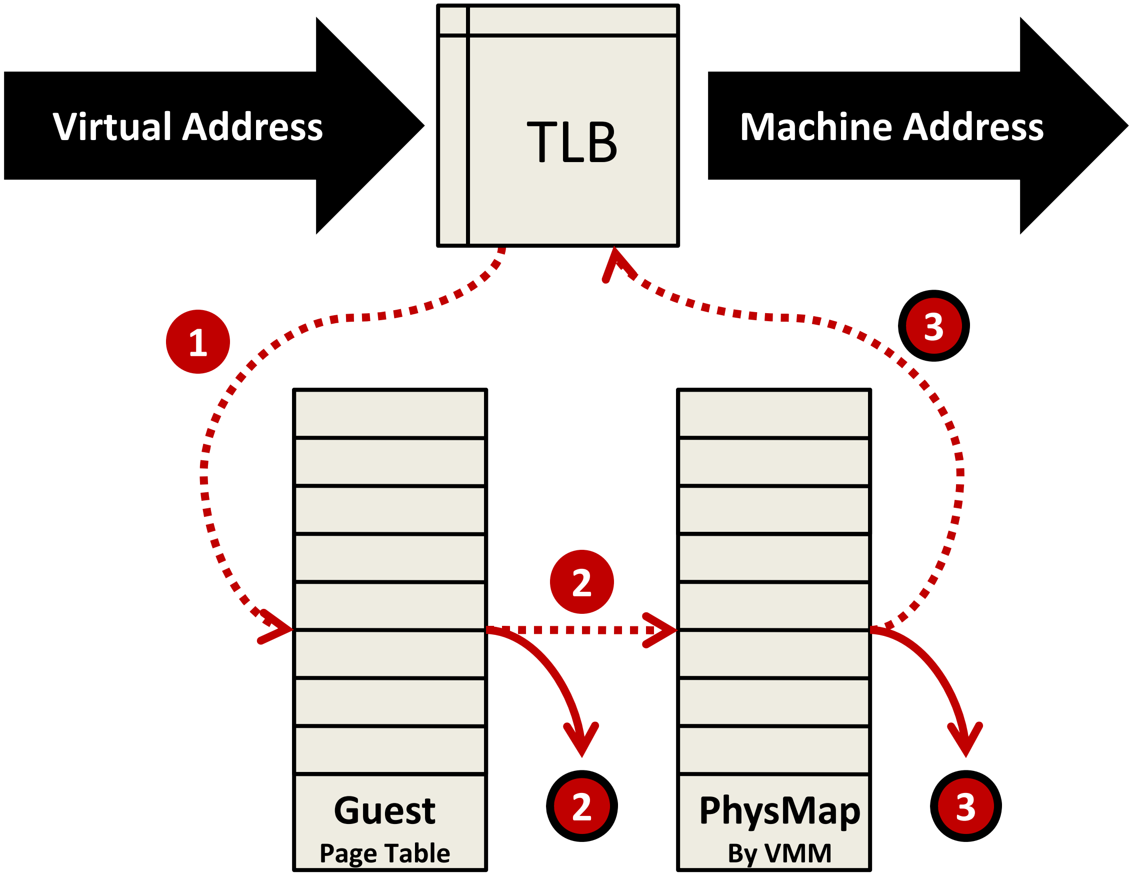
Outline

- **Background**
- **Virtualization Techniques**
  - Emulated TLB
  - Shadow Page Tables
- **Page Protection**
  - Memory Tracing
  - Hiding the Monitor
- **Hardware-supported Memory Virtualization**
  - Nested Page Tables

Virtualized Address Spaces w/ Nested Page Tables



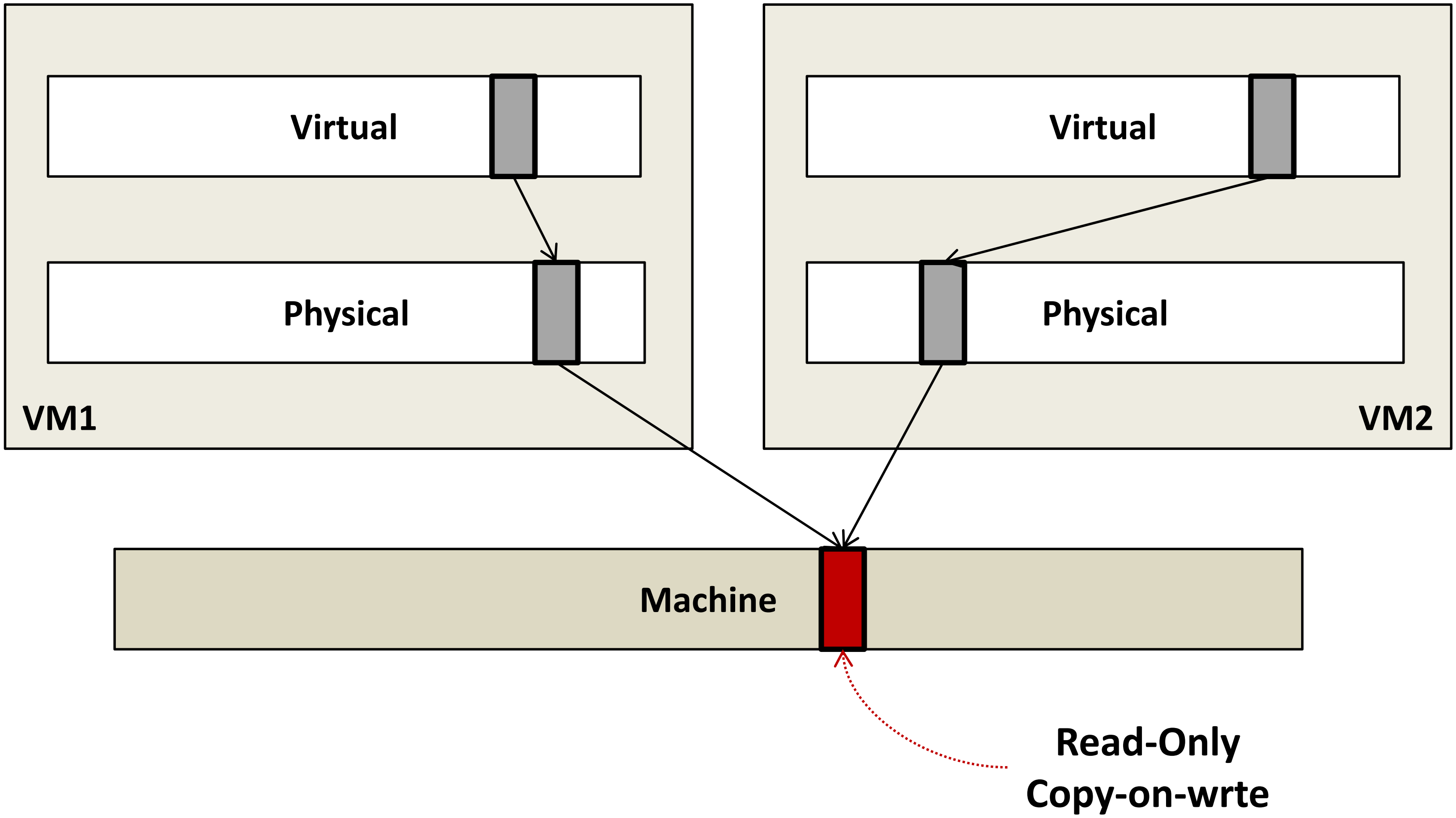
Virtualized Address Translation w/ Nested Page Tables



Issues with Nested Page Tables

- **Positives**
  - Simplifies monitor design
  - No need for page protection calculus
- **Negatives**
  - Guest page table is in physical address space
  - Need to walk PhysMap multiple times
    - Need physical to machine mapping to walk guest page table
    - Need physical to machine mapping for original virtual address
- **Other Memory Virtualization Hardware Assists**
  - Monitor Mode has its own address space
    - No need to hide the monitor

Interposition with Memory Virtualization
Page Sharing


Virtual
Physical
VM1
Virtual
Physical
VM2
Machine
Read-Only
Copy-on-wrte