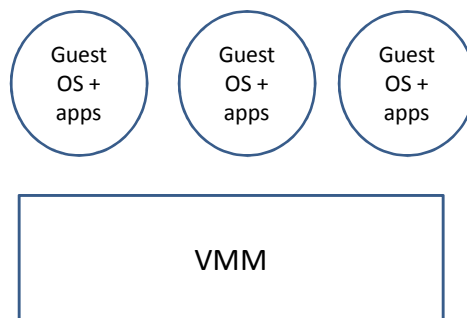


# System Virtual Machines

## Introduction

## Purpose

- Present the abstraction of a different machine to *an operating system*.



## Sensitive Instructions

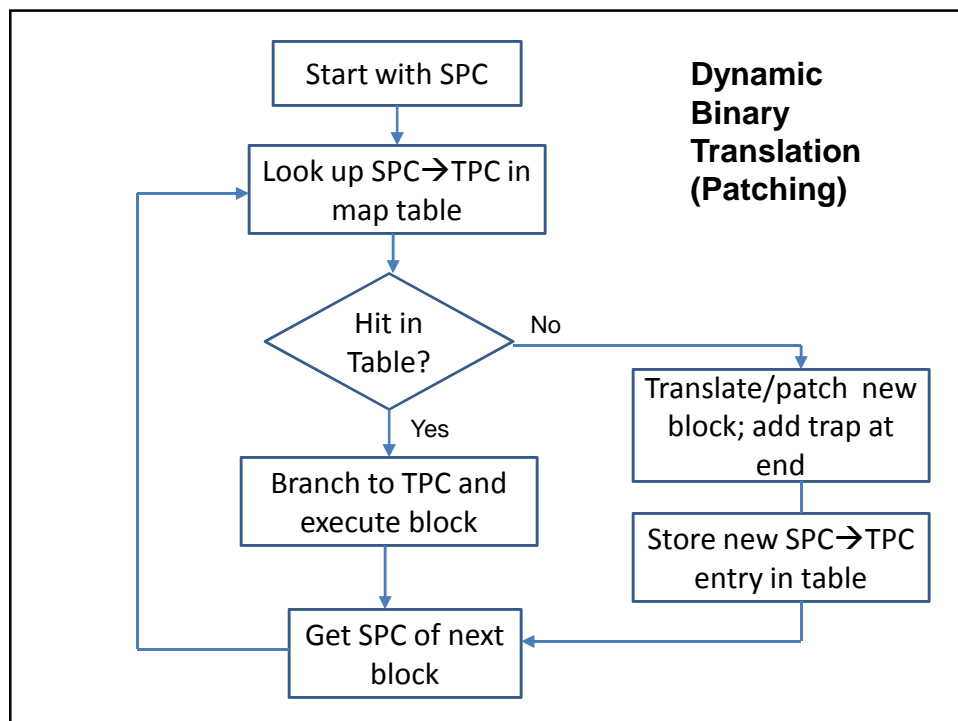
- Any instructions that directly affect resource allocation or access privileges are *sensitive*
- Guest OSes should not be allowed to perform sensitive instructions directly because they may gain unfair advantage over other VMs
  - Note: Guest OSes usually run in privileged mode.
  - When emulated, they run in user mode.
  - Privileged instructions result in traps
  - If all sensitive instructions are privileged, they will trap to VMM, which can decide how to execute them

## Problem Instructions

- What if a sensitive instruction is not privileged? (i.e., is allowed to be executed in user mode on the given architecture)

## Problem Instructions

- What if a sensitive instruction is not privileged? (i.e., is allowed to be executed in user mode on the given architecture)
  - Must replace all such instructions by traps manually, called *patching* (similar to dynamic binary translation)

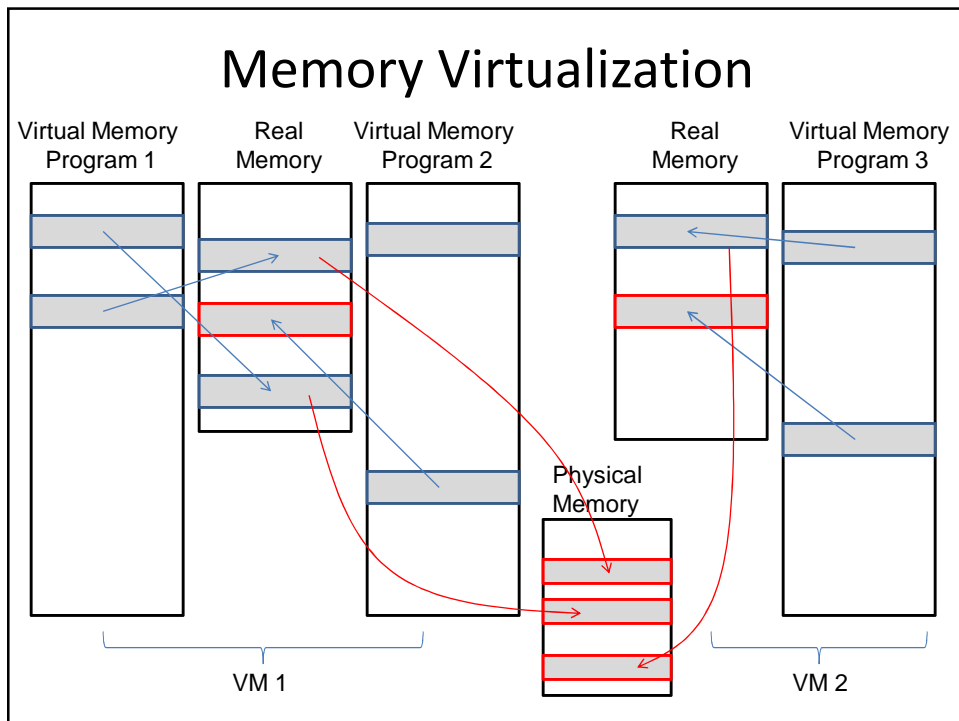
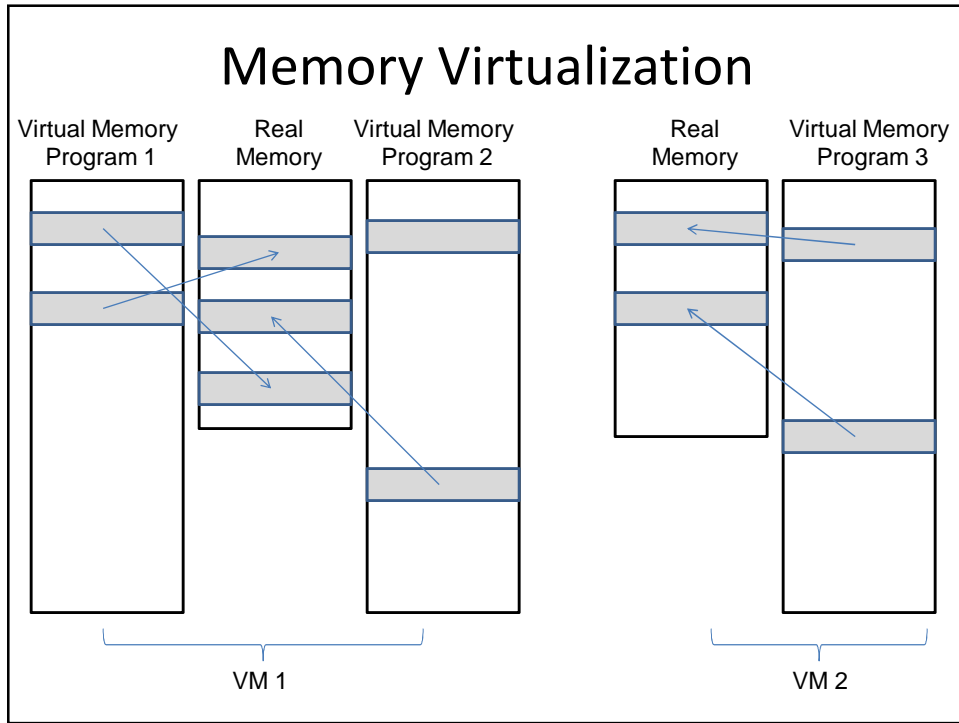


## Memory Virtualization

- In a typical machine, application virtual address space is mapped to real memory via page table + TLB
- What if there are multiple VMs sharing a single physical machine (physical memory)?
  - Distinguish between real memory and physical memory
  - Each VM maps its applications' virtual memory to "real memory" of VM
  - VMM maps "real memory" of VMs to physical memory of host

## Memory Virtualization

- In a typical machine, application virtual address space is mapped to real memory via page table + TLB
- What if there are multiple VMs sharing a single physical machine (physical memory)?
  - Distinguish between real memory and physical memory
  - Each VM maps its applications' virtual memory to "real memory" of VM
  - VMM maps "real memory" of VMs to physical memory of host



## Memory Virtualization

- The OS in each VM maintains own page table (virtual-to-real mapping)
- Virtual-to-physical mapping is maintained by VMM in shadow page tables, one per VM/program.
- Hardware uses shadow page tables in virtual-to-physical translation
- Page table pointer register must be virtualized

## Page Table Pointer Register

- Virtualized: all accesses to register trap to VMM
  - Read access: VMM returns the virtual page table pointer value (the guest PTP register)
  - Write access: VMM updates the virtual page table pointer value and sets the real page table pointer register to point to the corresponding shadow table.

## Page Faults

- If guest page table does not have entry to map given virtual page (i.e., page is on disk), VMM shadow page table does not map page
- If guest page table has entry mapping given virtual page to real memory, VMM may or may not have mapping of virtual page to physical memory
  - Spurious page faults possible.

## Page Faults

- If page fault (on shadow page table):
  - If virtual page is mapped to real memory:
    - Page fault is handled by VMM.
    - Page is fetched into physical memory
    - Update real map table and shadow page table
    - Return (note: page fault is transparent to guest OS)
  - If virtual page is not mapped to real memory:
    - Transfer control to guest page fault trap handler; write-protect guest page table
    - Guest loads-in page and updates page table
    - Page table update traps to VMM.
    - VMM updates both guest table (where the page would have been in real memory) and shadow page table (where the page actually is in physical memory)