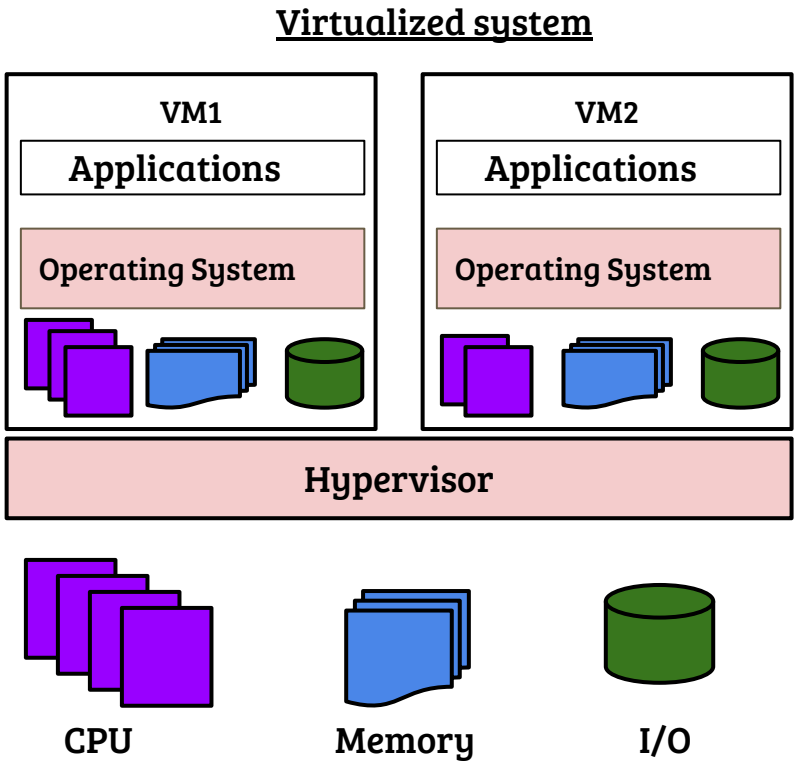
Topics in Operating Systems

Advanced isolation: Virtualization

Debadatta Mishra, CSE, IITK

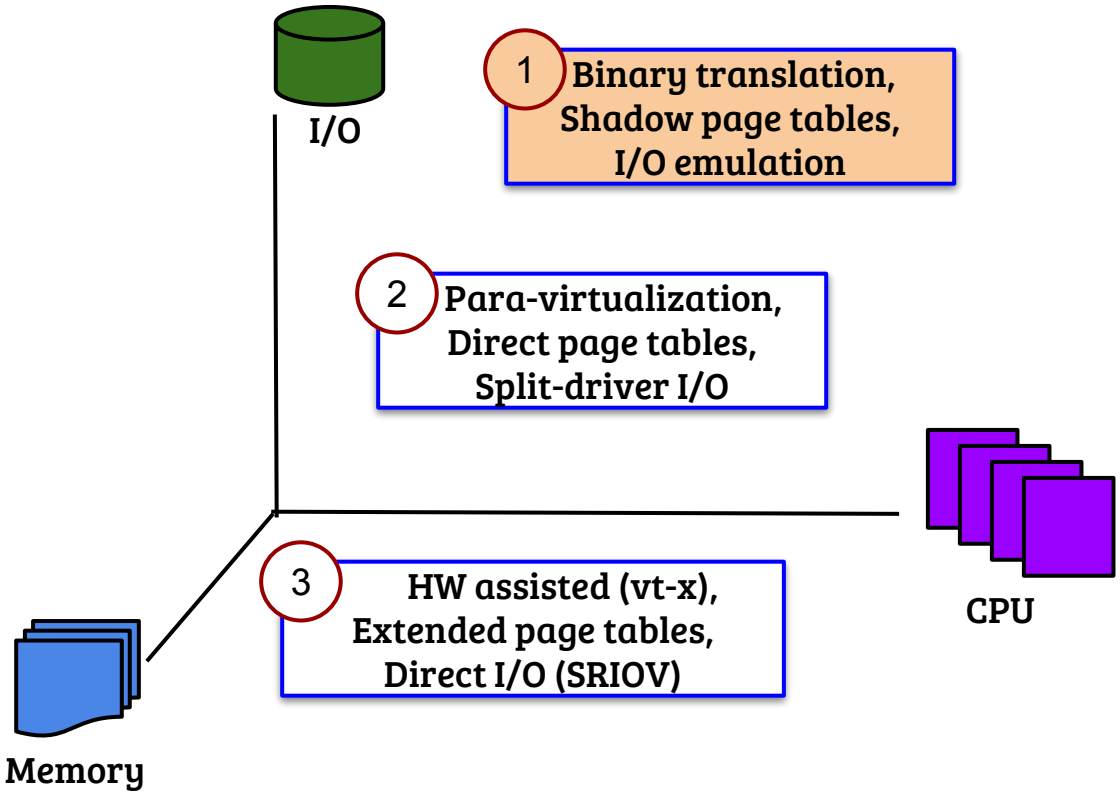
Virtualization: Resource multiplexing with isolation



- Definition [1] "Not physically existing as such but made by software to appear to do so."
- Objectives
  - Equivalence
  - Isolation
  - Resource control
  - Efficiency

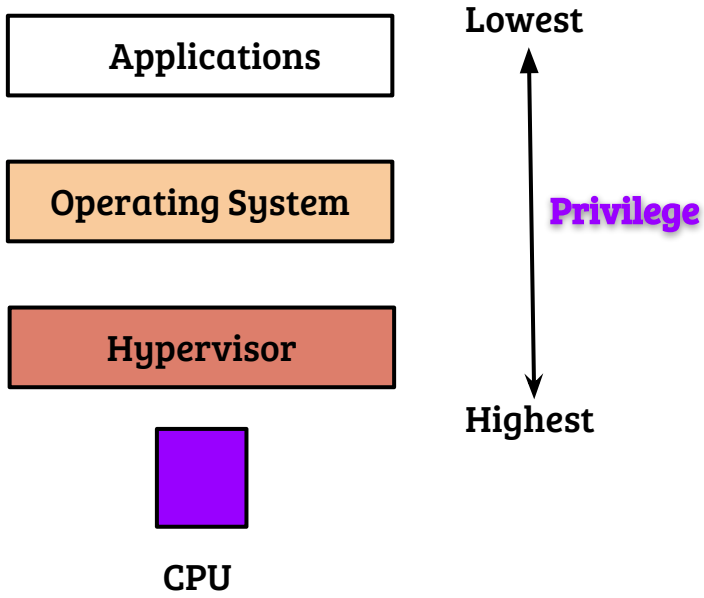
1. Oxford dictionary : https://en.oxforddictionaries.com/definition/virtual

Overview of virtualization approaches



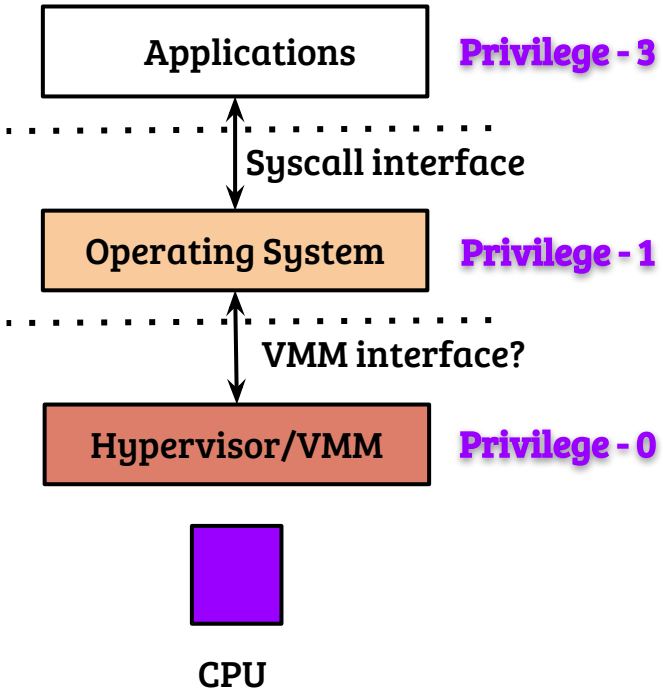
- (1) and (2) does not require any hardware assistance
- Different techniques try to strike a balance between the virtualization objectives
- Agenda for today's lecture: CPU virtualization

Software approaches of CPU virtualization



- Recap: Architectural support for Limited direct execution
  - Example: ISA support for privileges
- Can it be extended to achieve virtualization? How?

Software approaches of CPU virtualization



Trap and emulate (classic virtualization)

- when the guest OS accesses a privileged resource,
  - Trap (protection fault)
  - Hypervisor handles the Trap
  - Equivalent software state must be maintained (e.g., VCPU)
  - Example: guest OS executes HLT

Trap and emulate

- ISA assumptions
  - All sensitive instructions must trap
  - All non-trapping instructions should behave identically as in a native system
  - Does not hold true for x86 (e.g., popf)
- Design considerations
  - Maintaining a shadow state (e.g., VCPU, shadow page tables)
  - Efficiency issues: every sensitive instruction is emulated
  - Avoiding trap-and-emulate on user to kernel switch and vice-a-versa (e.g., syscall and return)

Shadow state of CPU

- Maintain a software state of CPU

```
struct VCPU {
    u64 GPRs[16];
    u64 CR[4];
    u64 cs, ds, ss, fs, gs;
    u64 flags, gdtr, idtr;
    ..................
};
```

- When the VM executes, load all non-critical registers with saved VCPU value
- Why not load all values? What happens if CR0-CR4 are loaded from the saved state and allowed to be accessed directly?

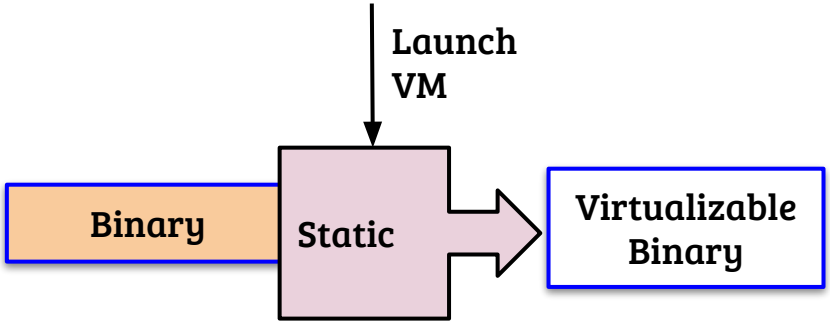
Binary translation with trap-and-emulate

- Translate between hardware and software state

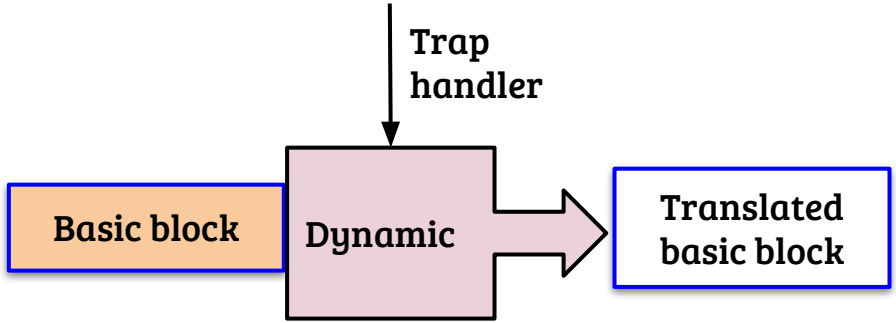
mov %cs, %eax ⇒ mov VCPU.cs, %eax
mov %eax, %cr0 ⇒ mov %eax, VCPU.CR[0]
cli ⇒ and $0xFFFFFDFF, VCPU.flags

- No traps after translation
- Direct translation may not always work, fall back to trap-and-emulate
- Example: “mov %rax, %cr3”
  - Require changing several software structures in hypervisor

Static and Dynamic BT [1]



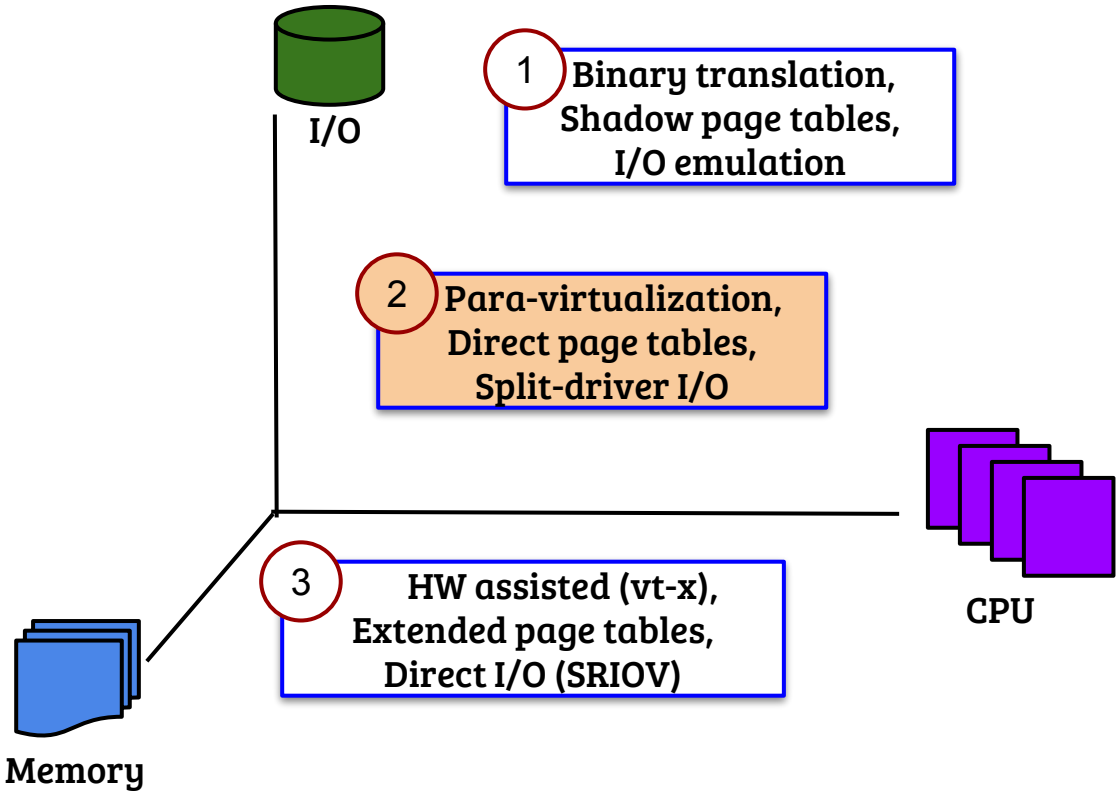
- Required to mitigate issues due to non-trapping sensitive instructions
- Mostly performed on guest OS code



- Translates sensitive code in a on-demand manner
  - Translate when required
  - Once translated, no more traps!
  - Can use software hash tables for efficient implementation

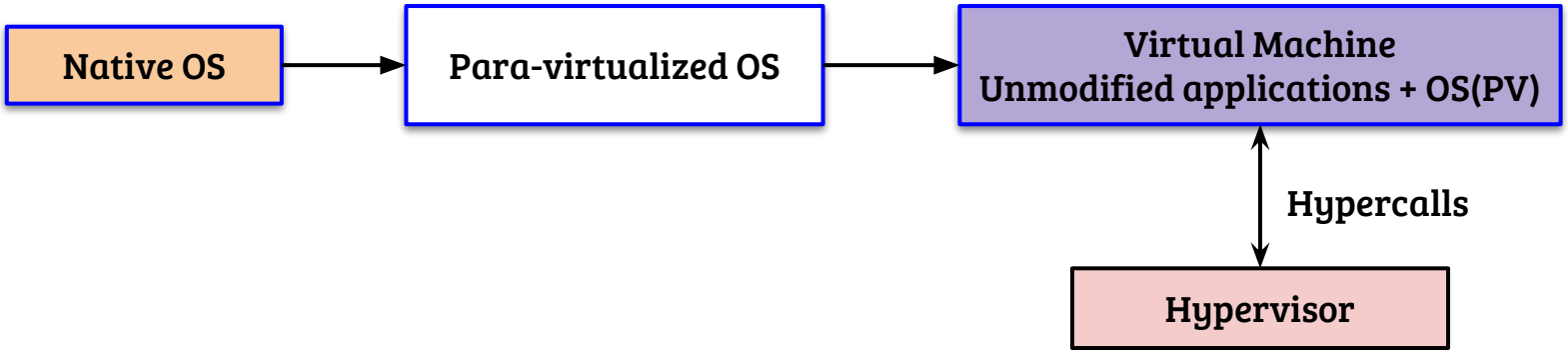
1. https://www.vmware.com/pdf/asplos235_adams.pdf

Overview of virtualization approaches



- Next: Para-virtualization

Paravirtualization [1]



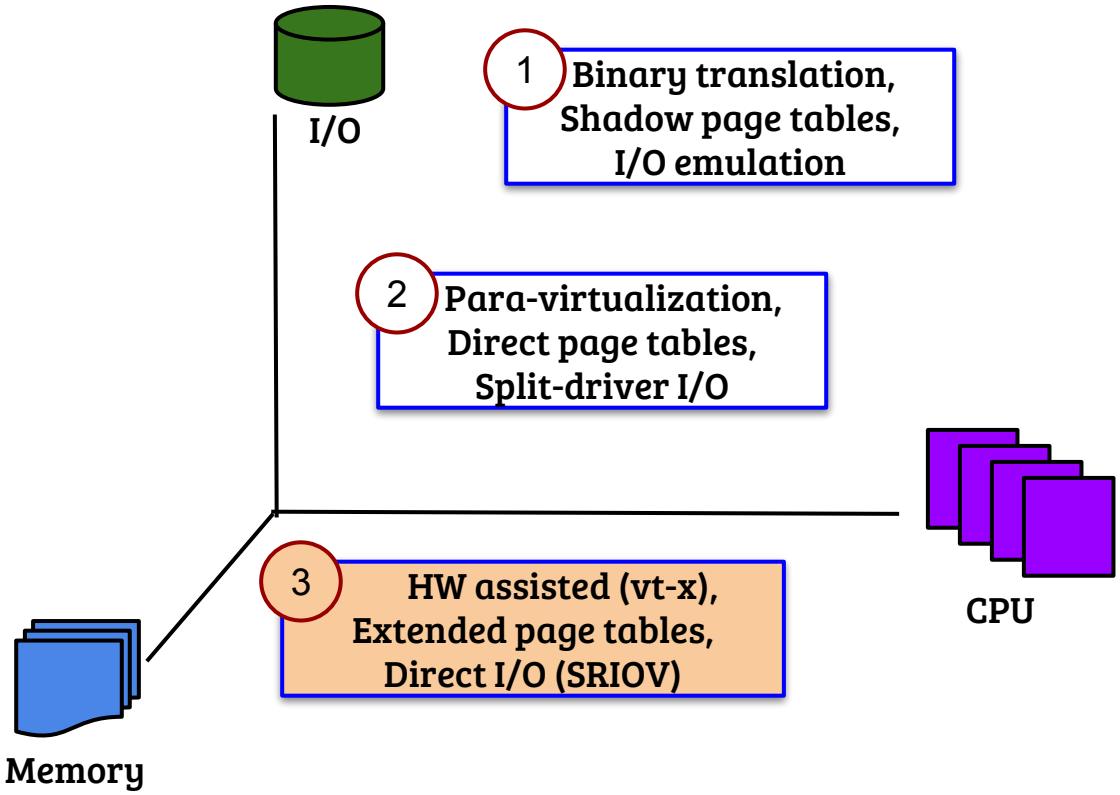
- Guest OS is made aware that it is executing in a virtualized system
- In the guest OS, carry out non-trapping sensitive operations and other sensitive operations through hypercalls (for efficiency)
- Example: load_idt( ) ⟶ xen_load_idt( ) ⟶ hypercall_set_trap_table(traps)

1. Xen and the art of virtualization, https://dl.acm.org/citation.cfm?id=945462

Xen on X86

- Protection: Guest OS executes in ring-1, Xen hypervisor in ring-0
- Exception handling
  - Guest OS registers a descriptor table through hypercall
  - Apart from page fault, other exception handling does not require hypervisor intervention
- System call handling
  - Guest OS registers fast system call handler
  - Verified by the hypervisor at the time of handler registration

Overview of virtualization approaches



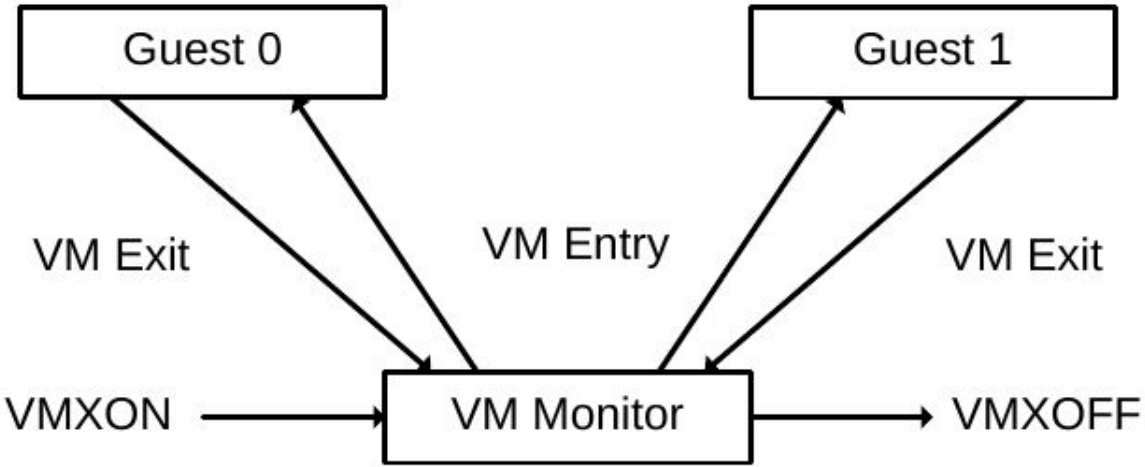
- Next: H/W assisted virtualization

Hardware assisted virtualization [1]

- Architecture designers: "Our design has holes w.r.t. Virtualization. Should we just fix it or try to enhance support for virtualization"
- Idea
  - Replicate the state of a CPU in hardware ⟶ native mode and VM mode
  - Expanded privilege levels ⟶ native mode (root mode) and Guest mode
- Example (Intel VMX)
  - Hypervisor executes in VMX-root mode
  - Guest OS executes in VMX non-root mode

1. Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3C

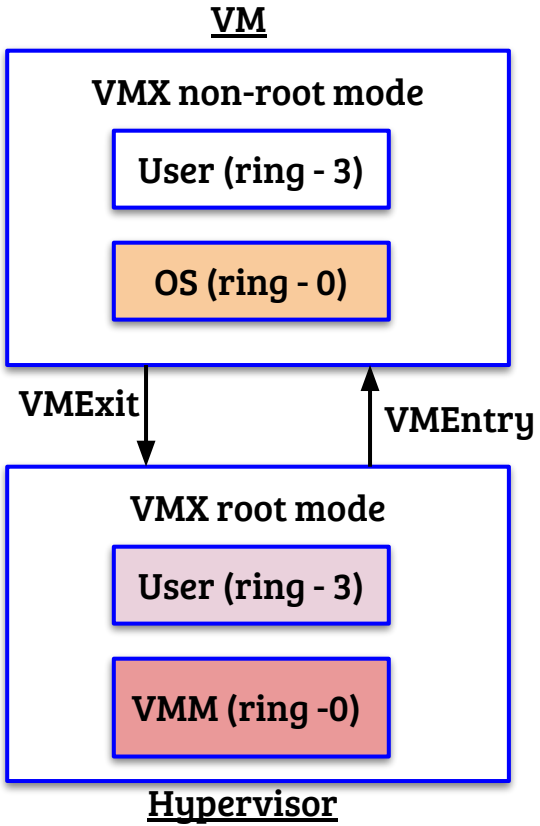
VMX modes and transition [1]

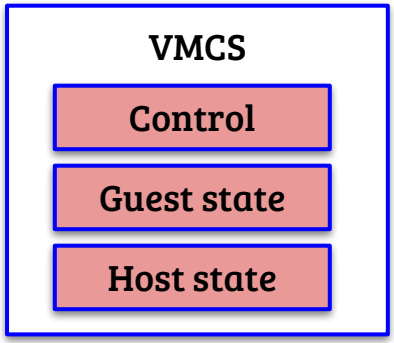


- Intel VT-X allows transition between
  - Native mode operation and virtualized mode operation
  - Transition between root mode and non-root mode in virtualized mode operation

1. Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3C

VMX structure and transition





- Hardware enabled virtual machine control structures (VMCS) provides mechanisms to define non-root mode operations and transition behaviors
- Control: Define Which events cause trap, store trap reason
- Guest state: Loaded on VMEntry
- Host state: Loaded on VMExit