Introduction to Virtual Machines

Scott Devine

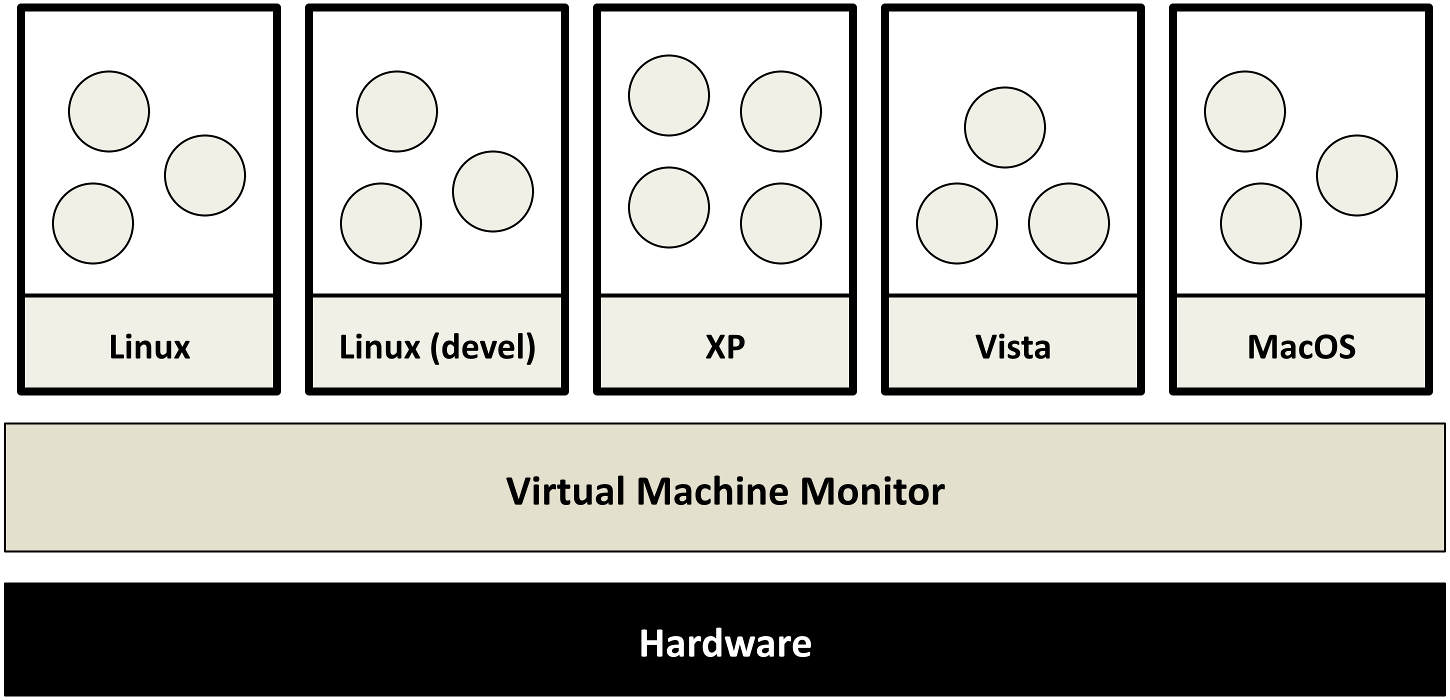
Principal Engineer, Co-Founder

VMware, Inc.

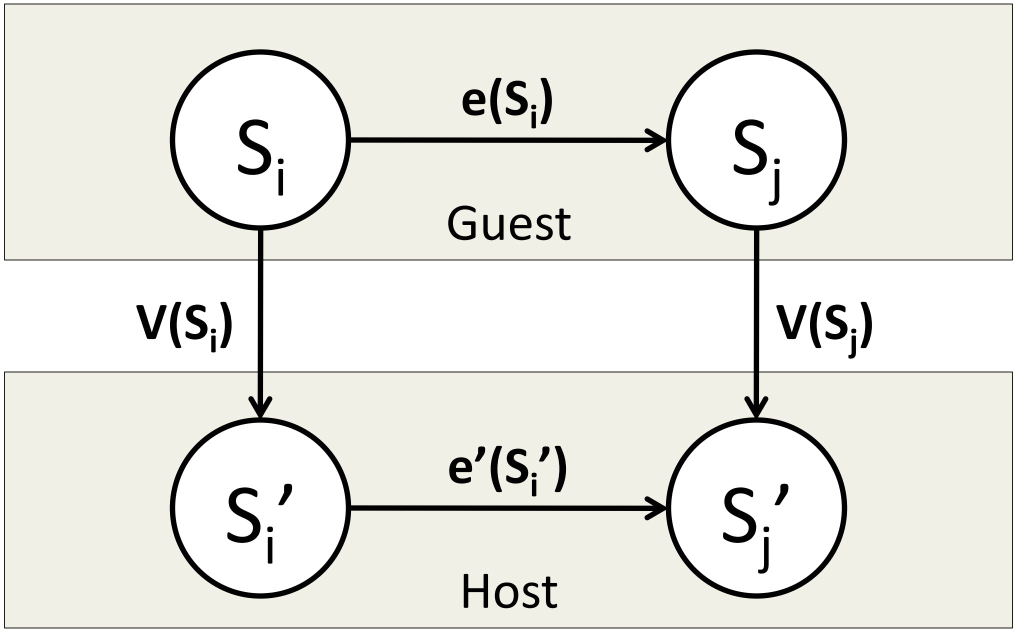
Outline

- What is virtualization?
- How-to virtualize
  - CPU
  - Memory
  - I/O

What is Virtualization



Isomorphism



Formally, virtualization involves the construction of an **isomorphism** from **guest** state to **host** state.

Virtualization Properties

- Isolation
- Encapsulation
- Interposition

Types of Virtualization

- Process Virtualization
  - Language construction
  - Cross-ISA emulation
    - Apple's 68000-PowerPC-Intel Transition
- Device Virtualization
  - RAID
- **System Virtualization**
  - VMware
  - Xen
  - Microsoft
  - KVM

System Virtualization Applications

- Server Consolidation
- Data Center Management
  - VMotion
- High Availability
  - Automatic Restart
- Disaster Recovery
- Fault Tolerance
- Test and Development
- Application Flexibility

CPU Virtualization

- Instruction Interpretation
- Trap and Emulate
- Binary Translation
- Hybrid

Instruction Interpretation

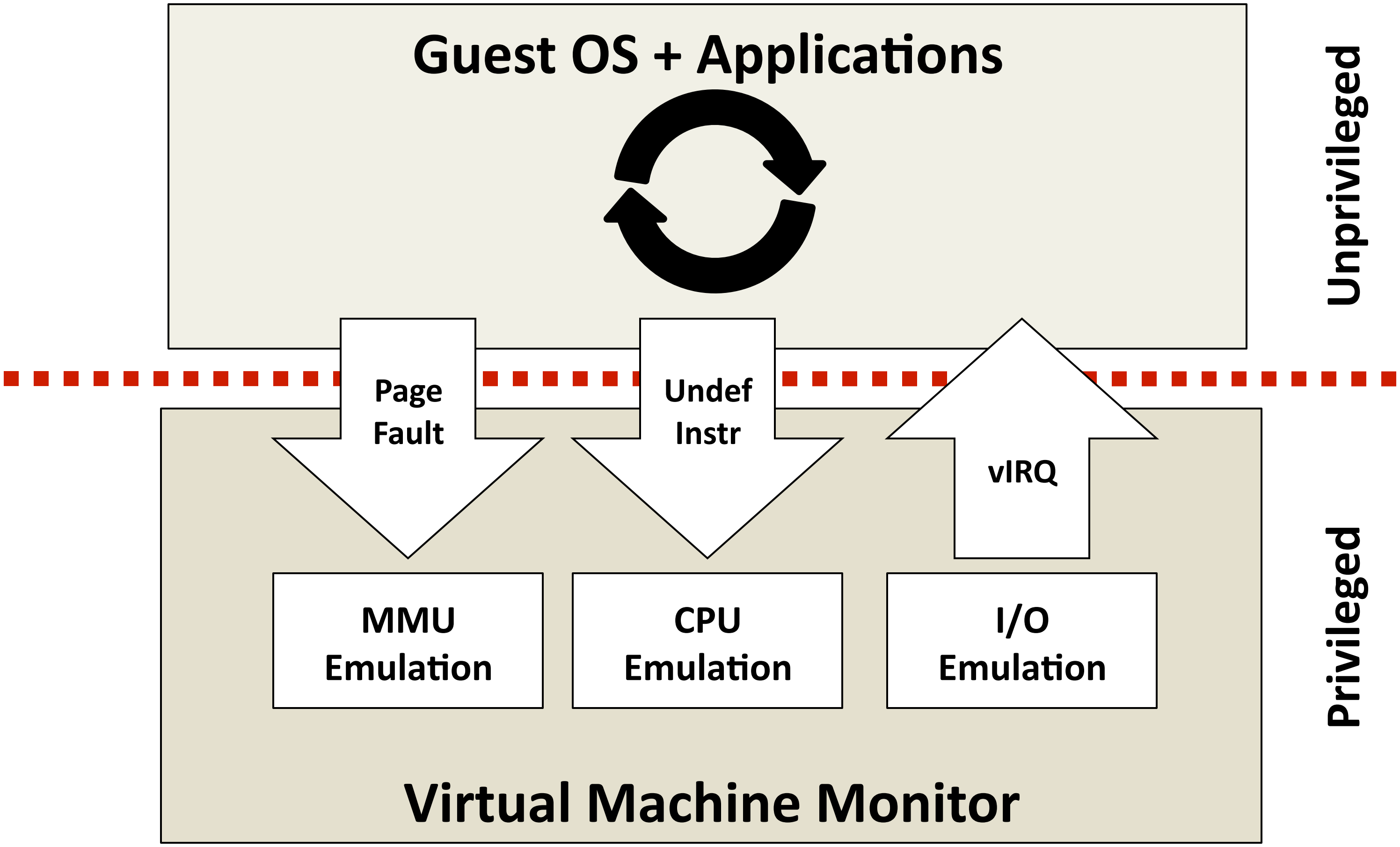
- Emulate Fetch/Decode/Execute pipeline in software
- Postives
  - Easy to implement
  - Minimal complexity
- Negatives
  - Slow!

Example: Virtualizing the Interrupt Flag w/ Instruction Interpreter

```
void CPU_Run(void)
{
   while (1) {
      inst = Fetch(CPUState.PC);

      CPUState.PC += 4;

      switch (inst) {
      case ADD:
         CPUState.GPR[rd]
            = GPR[rn] + GPR[rm];
         break;
      …
      case CLI:
         CPU_CLI();
         break;
      case STI:
         CPU_STI();
         break;
      }

      if (CPUState.IRQ
          && CPUState.IE) {
         CPUState.IE = 0;
         CPU_Vector(EXC_INT);
      }
   }
}
```

```
void CPU_CLI(void)
{
   CPUState.IE = 0;
}

void CPU_STI(void)
{
   CPUState.IE = 1;
}

void CPU_Vector(int exc)
{
   CPUState.LR = CPUState.PC;
   CPUState.PC = disTab[exc];
}
```

Trap and Emulate



“Strictly Virtualizable”

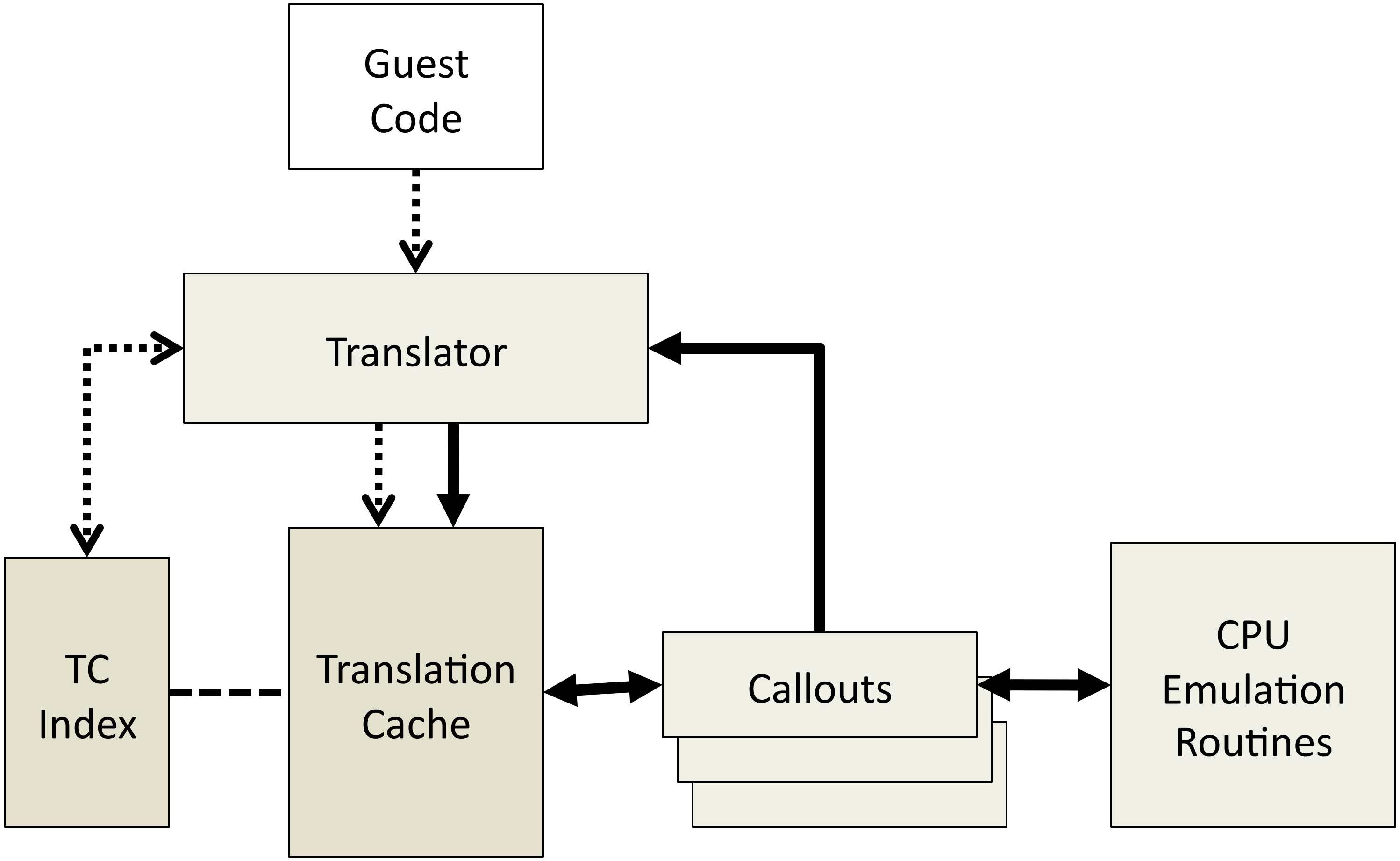
A processor or mode of a processor is strictly virtualizable if, when executed in a lesser privileged mode:

- all instructions that access privileged state trap
- all instructions either trap or execute identically
- …

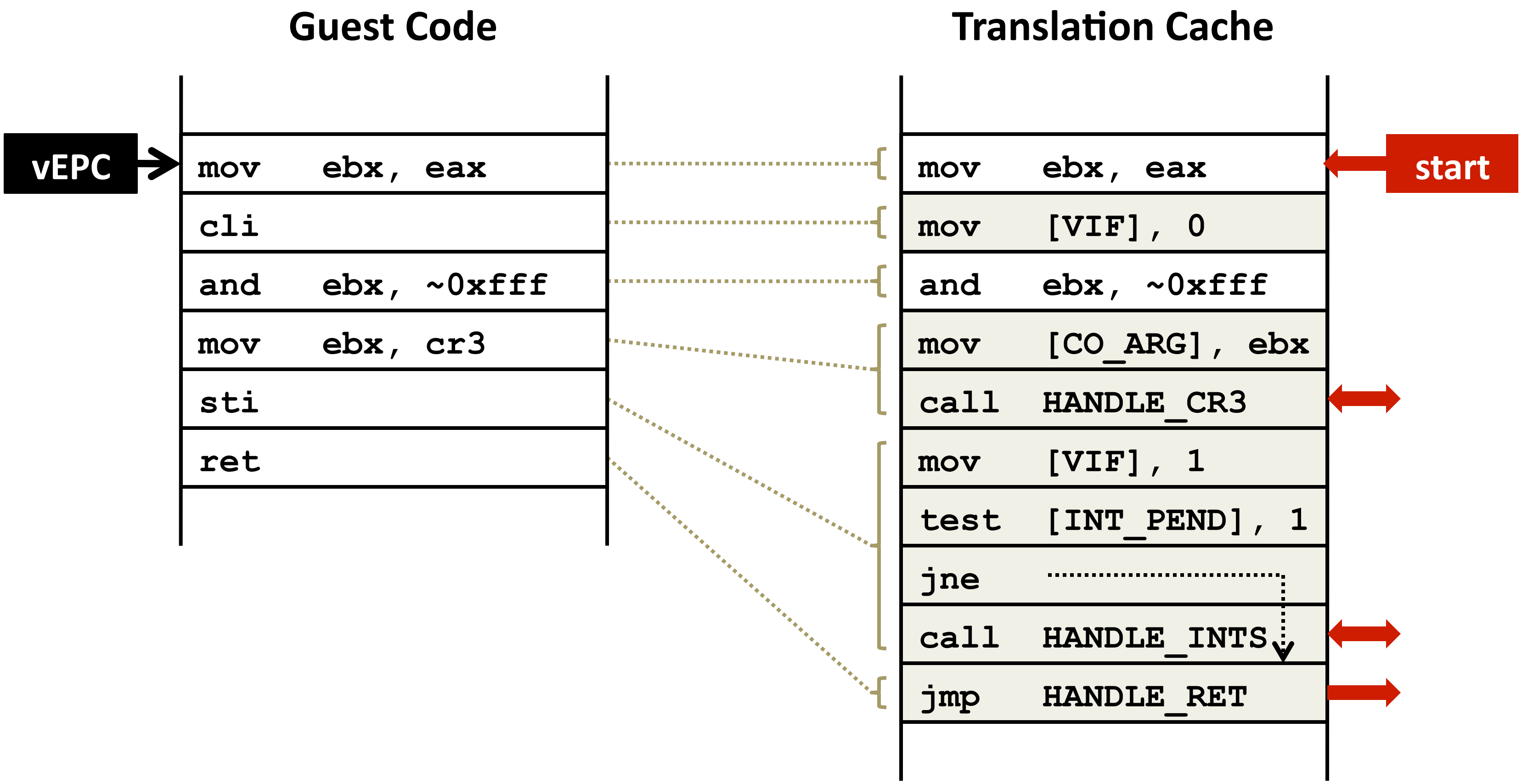
Issues with Trap and Emulate

- Not all architectures support it
- Trap costs may be high
- Monitor uses a privilege level
  - Need to virtualize the protection levels

Binary Translator



Binary Translation



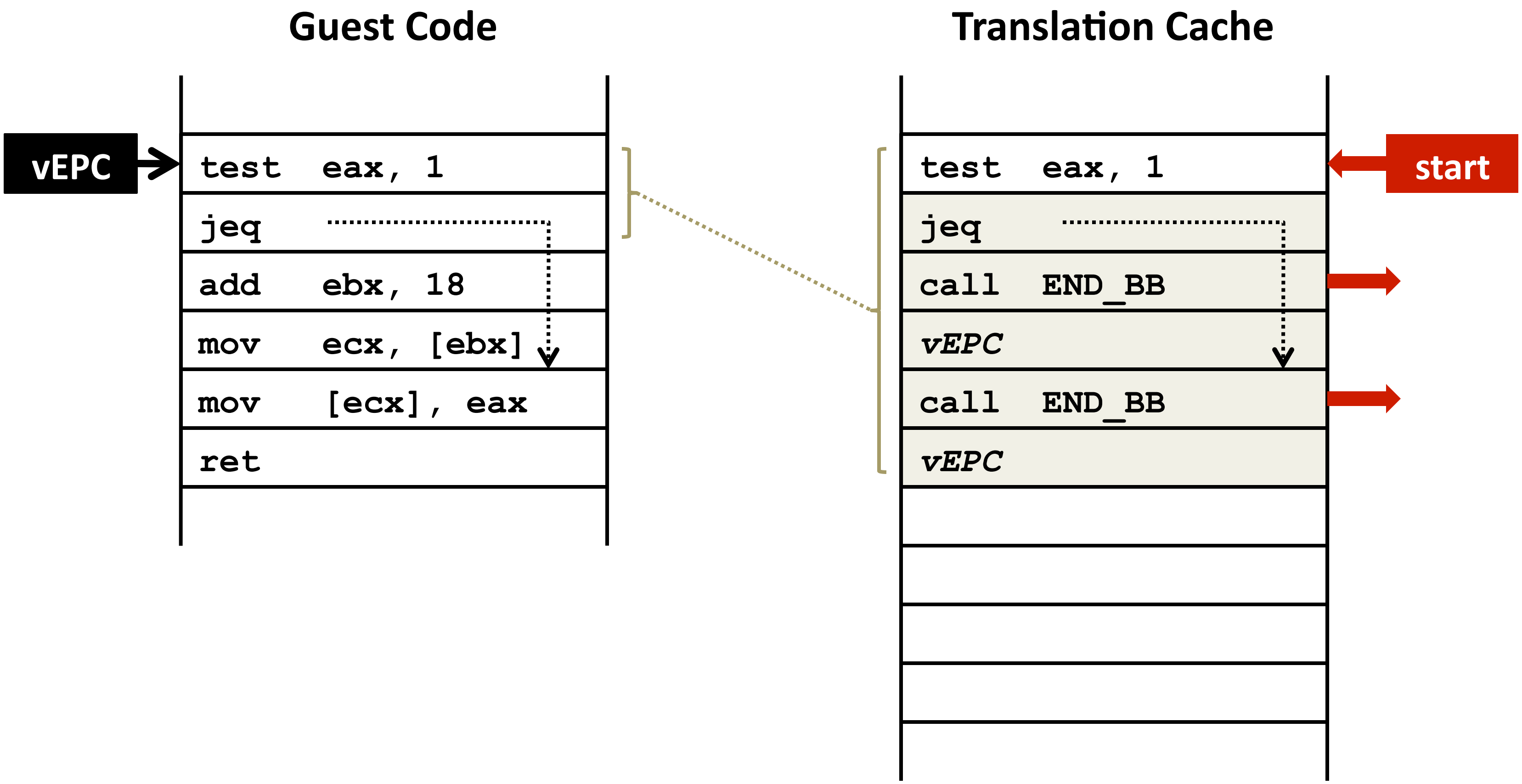
Guest Code

```
mov   ebx, eax
cli
and   ebx, ~0xfff
mov   ebx, cr3
sti
ret
```

Translation Cache

```
mov   ebx, eax
mov   [VIF], 0
and   ebx, ~0xfff
mov   [CO_ARG], ebx
call  HANDLE_CR3
mov   [VIF], 1
test  [INT_PEND], 1
jne
call  HANDLE_INTS
jmp   HANDLE_RET
```

Controlling Control Flow



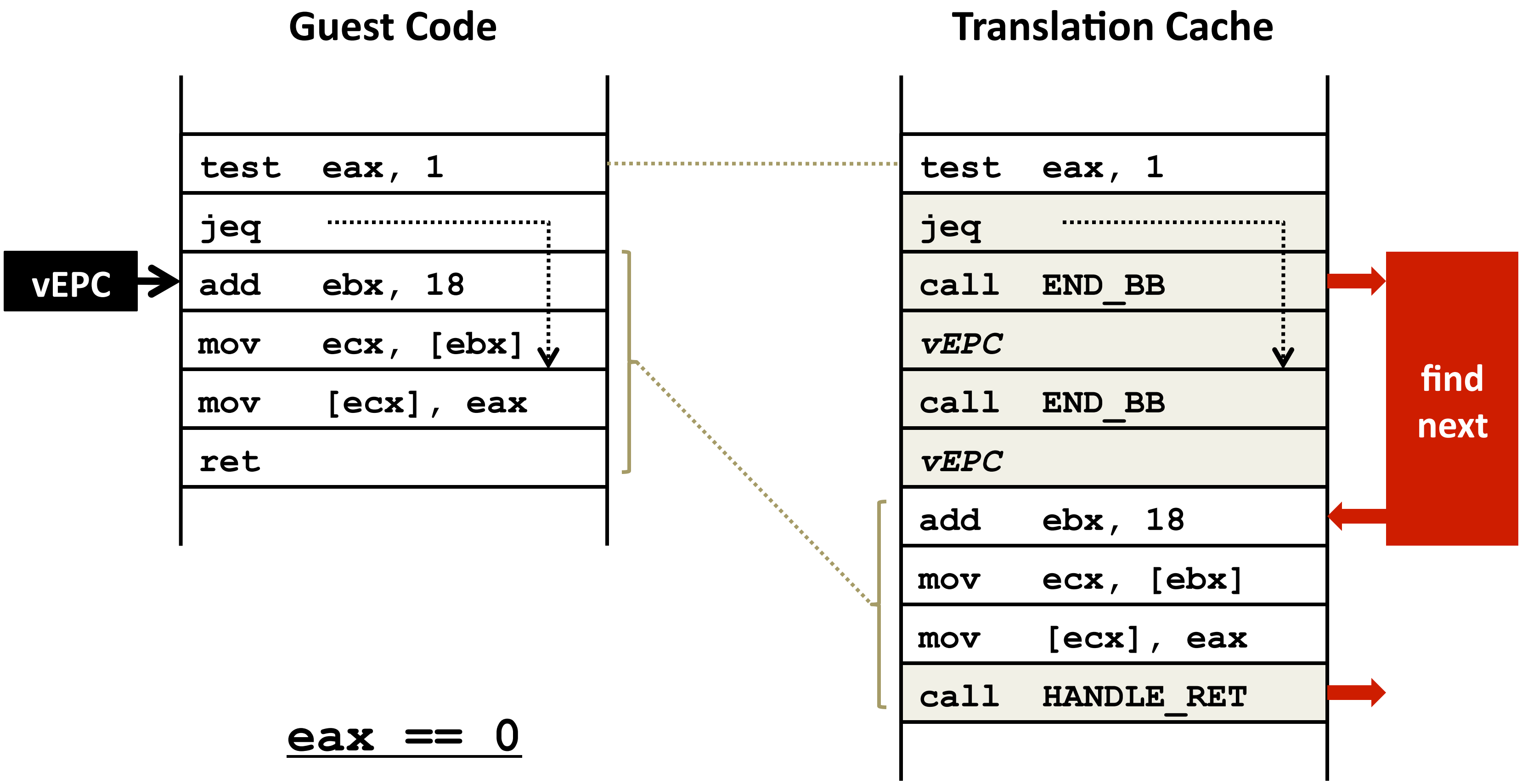
Guest Code

```
test   eax, 1
jeq
add    ebx, 18
mov    ecx, [ebx]
mov    [ecx], eax
ret
```

Translation Cache

```
test   eax, 1
jeq
call   END_BB
vEPC
call   END_BB
vEPC
```

Controlling Control Flow



Guest Code

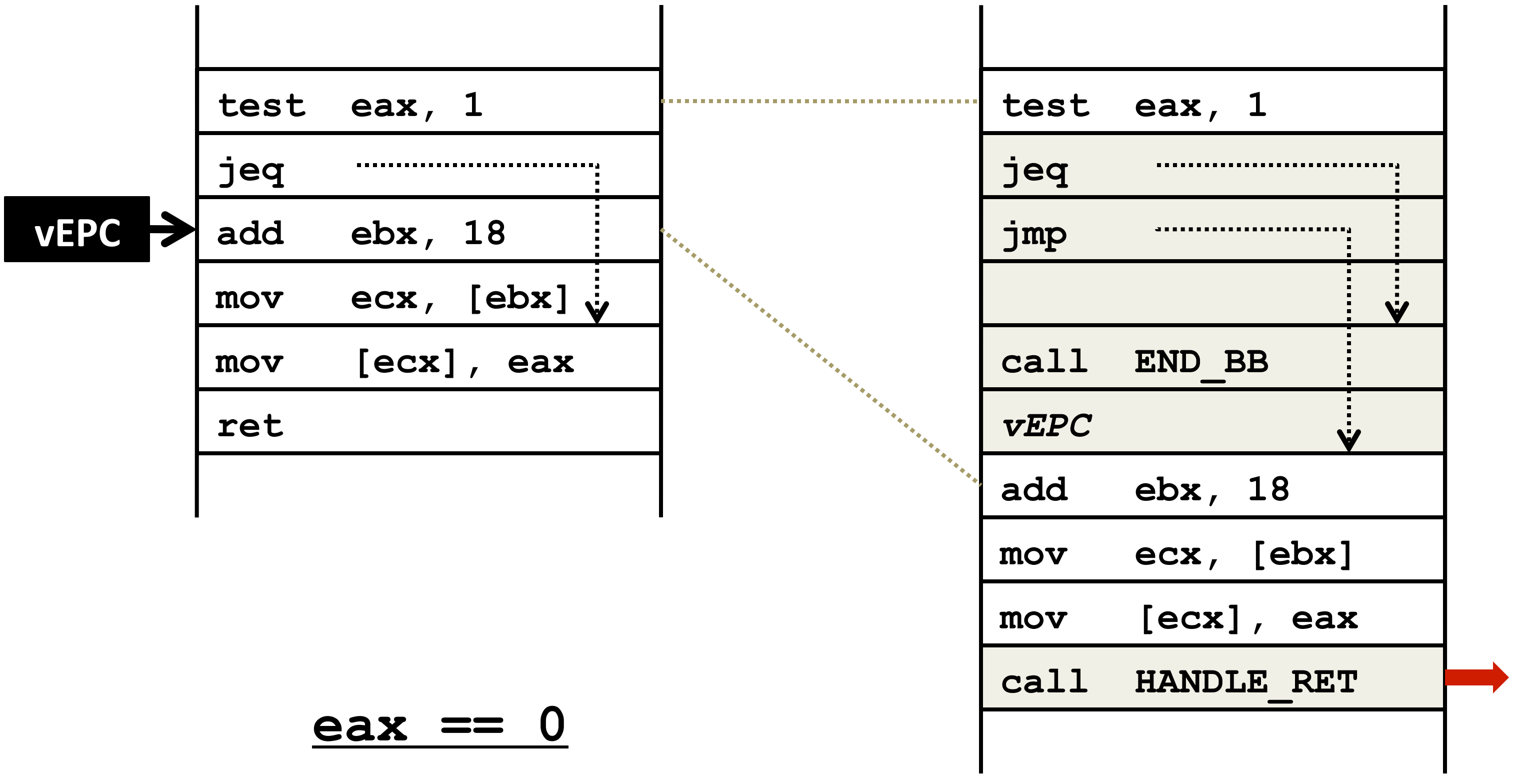
```
test  eax, 1
jeq
add   ebx, 18
mov   ecx, [ebx]
mov   [ecx], eax
ret
```

Translation Cache

```
test  eax, 1
jeq
call  END_BB
vEPC
call  END_BB
vEPC
add   ebx, 18
mov   ecx, [ebx]
mov   [ecx], eax
call  HANDLE_RET
```

`eax == 0`

Controlling Control Flow



Guest Code

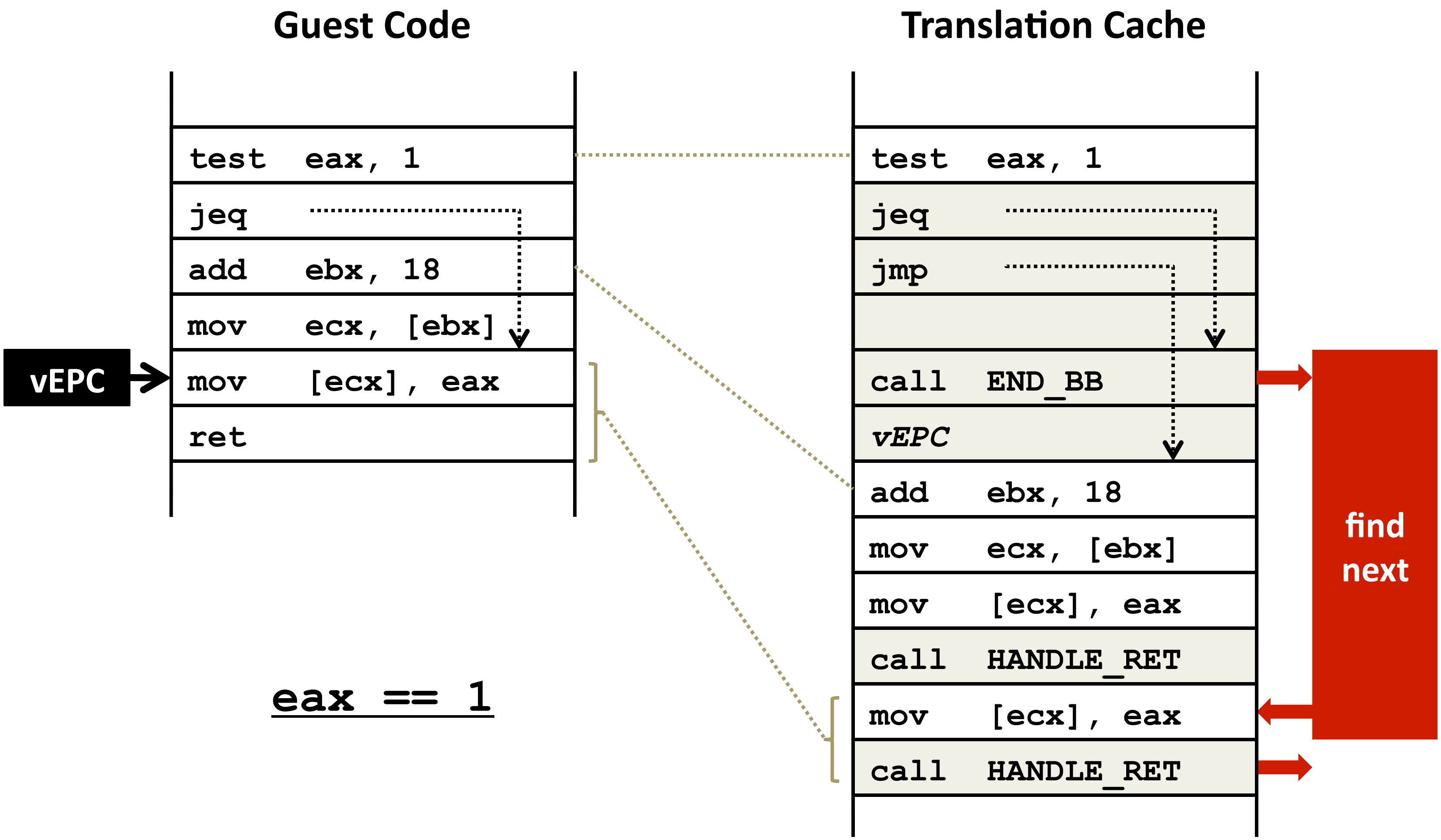
```
test  eax, 1
jeq
add   ebx, 18
mov   ecx, [ebx]
mov   [ecx], eax
ret
```

Translation Cache

```
test  eax, 1
jeq
jmp

call  END_BB
vEPC
add   ebx, 18
mov   ecx, [ebx]
mov   [ecx], eax
call  HANDLE_RET
```

<u>**`eax == 0`**</u>

Controlling Control Flow



Guest Code

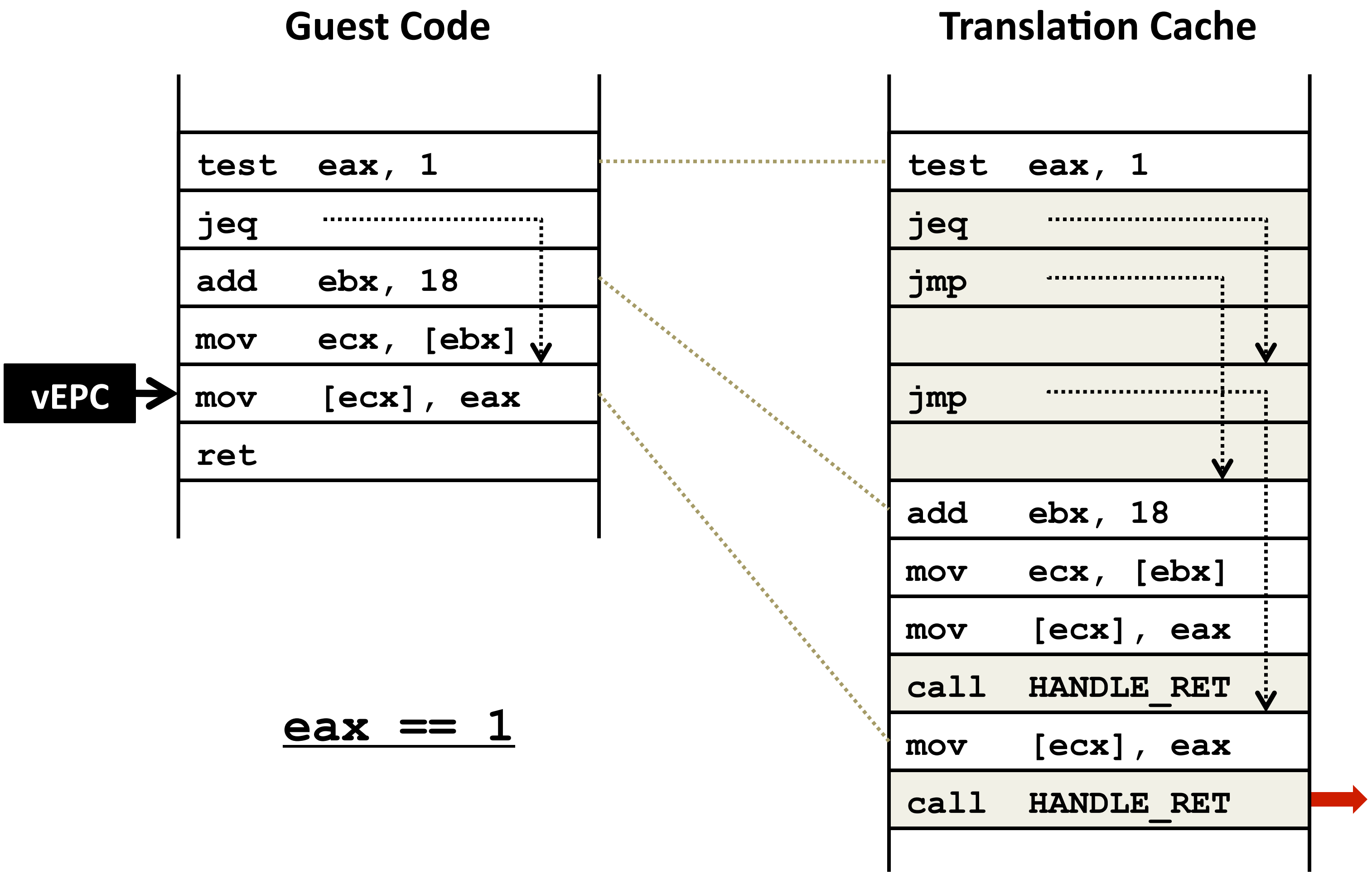
```
test  eax, 1
jeq
add   ebx, 18
mov   ecx, [ebx]
mov   [ecx], eax
ret
```

Translation Cache

```
test  eax, 1
jeq
jmp

call  END_BB
vEPC
add   ebx, 18
mov   ecx, [ebx]
mov   [ecx], eax
call  HANDLE_RET
mov   [ecx], eax
call  HANDLE_RET
```

eax == 1

Controlling Control Flow



Guest Code

```
test  eax, 1
jeq
add   ebx, 18
mov   ecx, [ebx]
mov   [ecx], eax
ret
```

Translation Cache

```
test  eax, 1
jeq
jmp

jmp

add   ebx, 18
mov   ecx, [ebx]
mov   [ecx], eax
call  HANDLE_RET
mov   [ecx], eax
call  HANDLE_RET
```

`eax == 1`

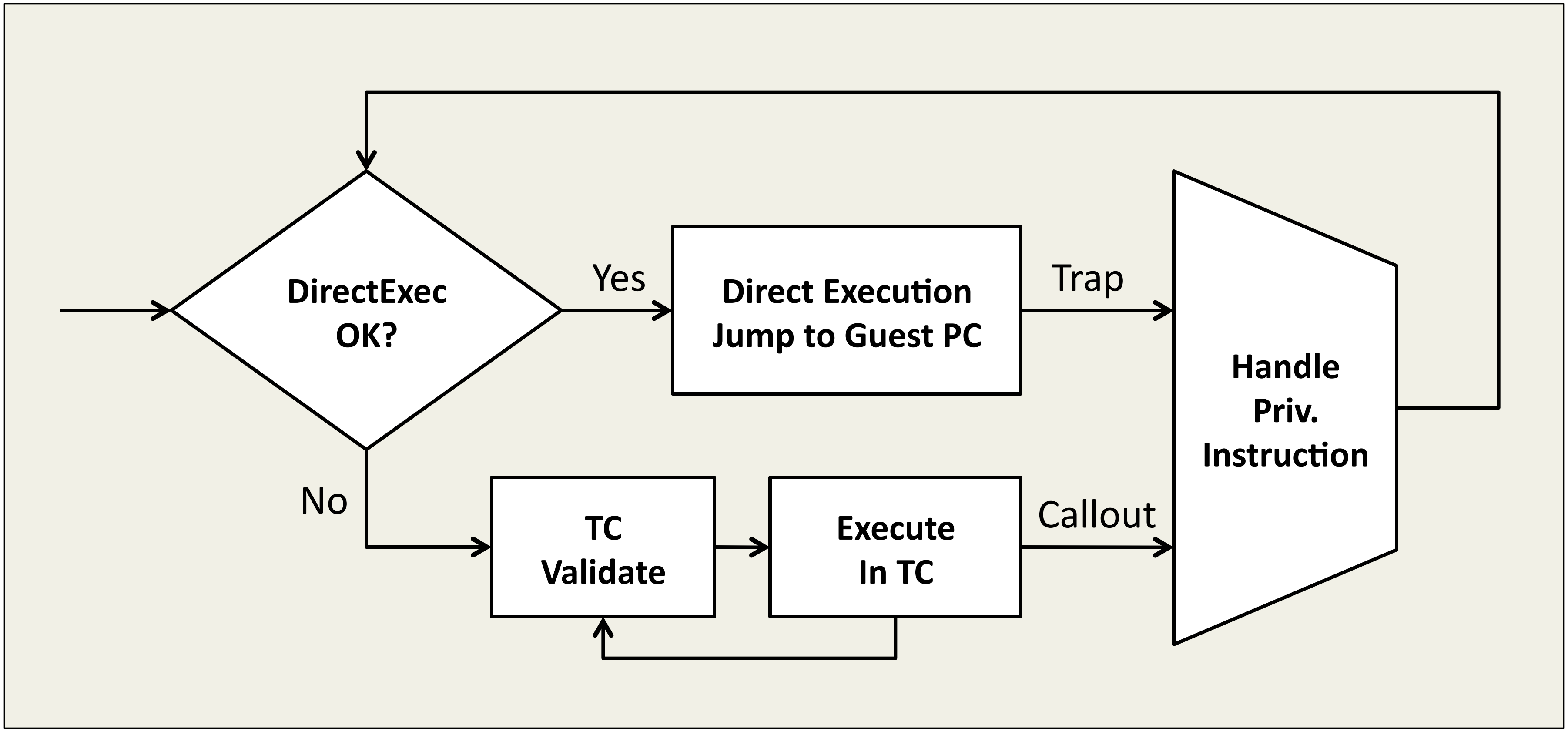
Issues with Binary Translation

- Translation cache index data structure
- PC Synchronization on interrupts
- Self-modifying code
  - Notified on writes to translated guest code

Other Uses for Binary Translation

- Cross ISA translators
- Optimizing translators
- High level languages

Hybrid Approach

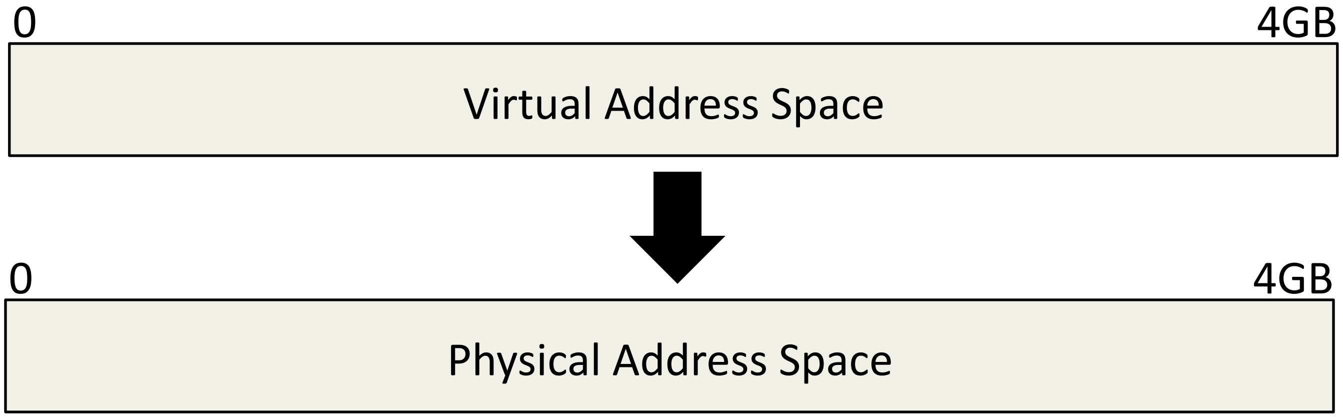


- Binary Translation for the Kernel
- Direct Execution (Trap-and-emulate) for the User
- U.S. Patent 6,397,242

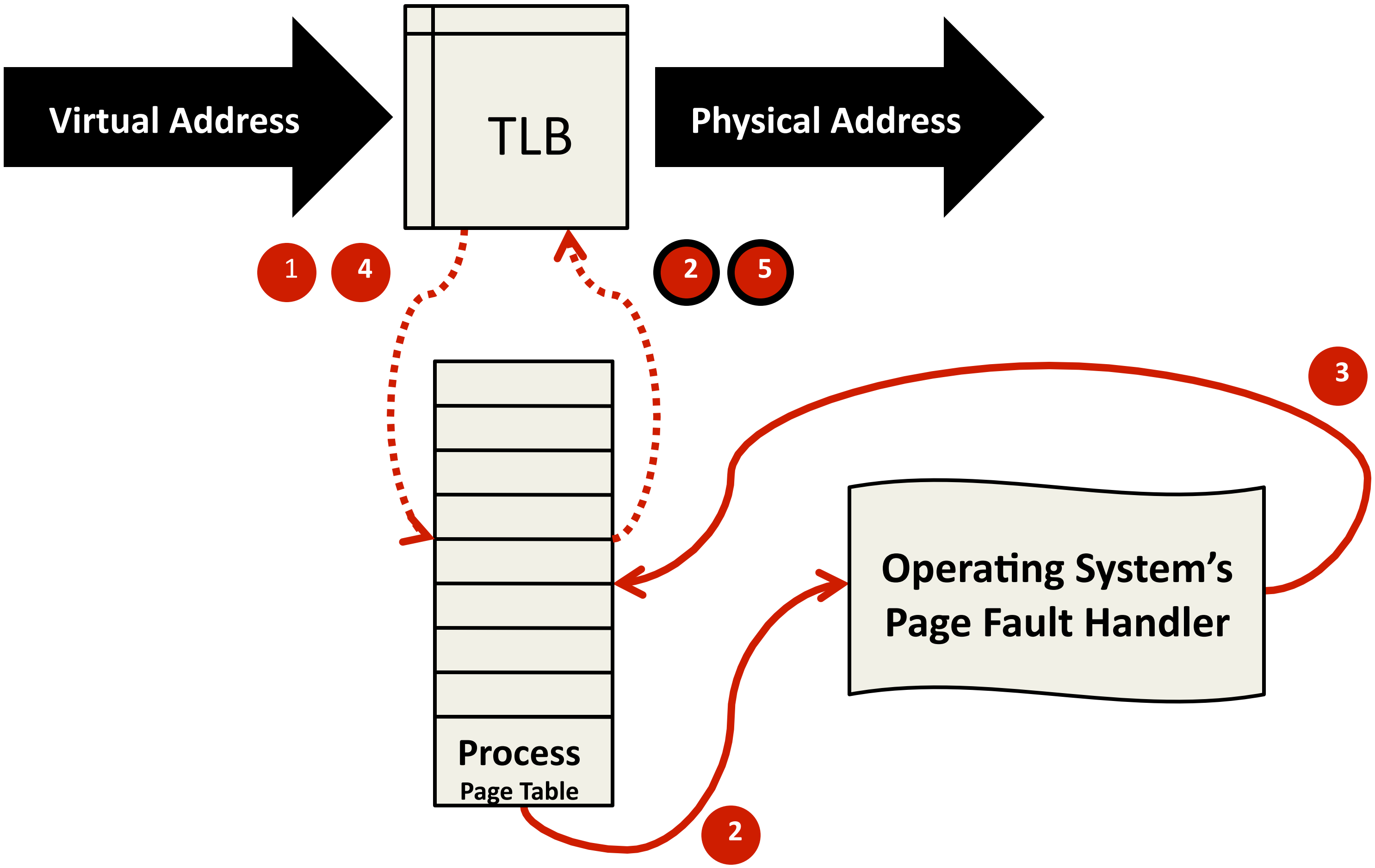
Memory Virtualization

- Shadow Page Tables
- Nested Page Tables

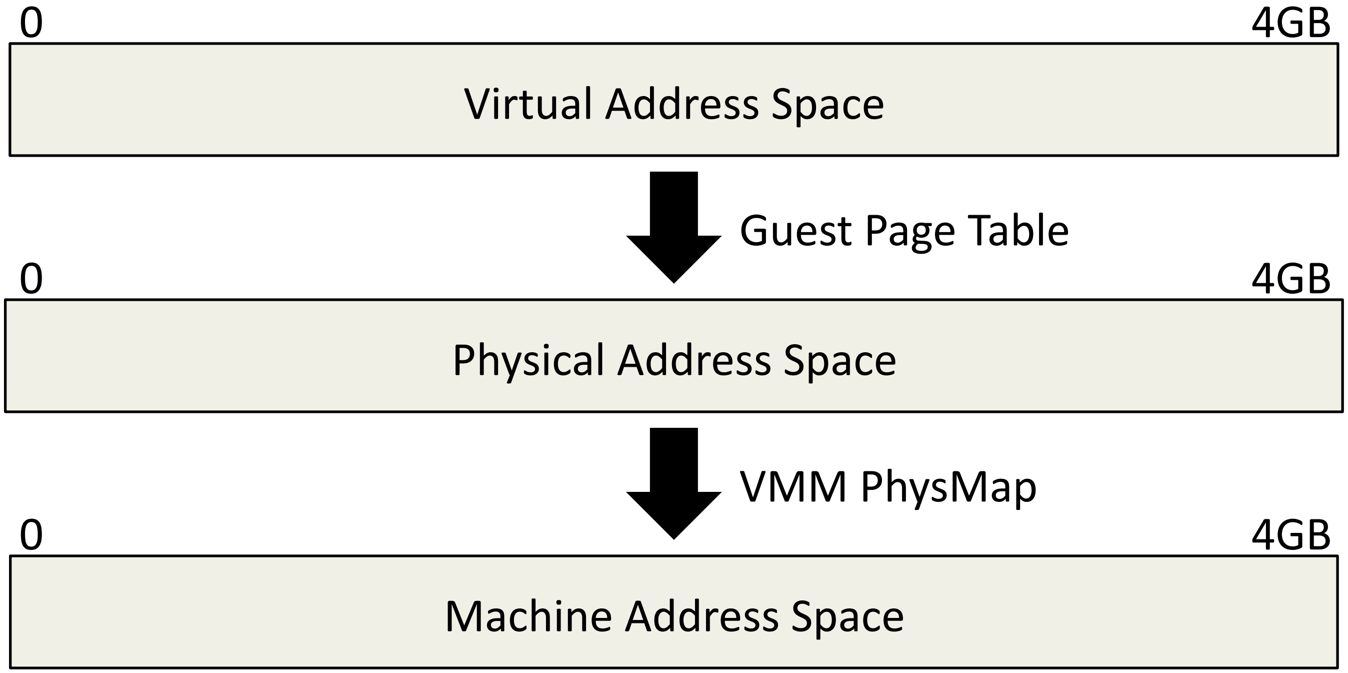
Traditional Address Spaces



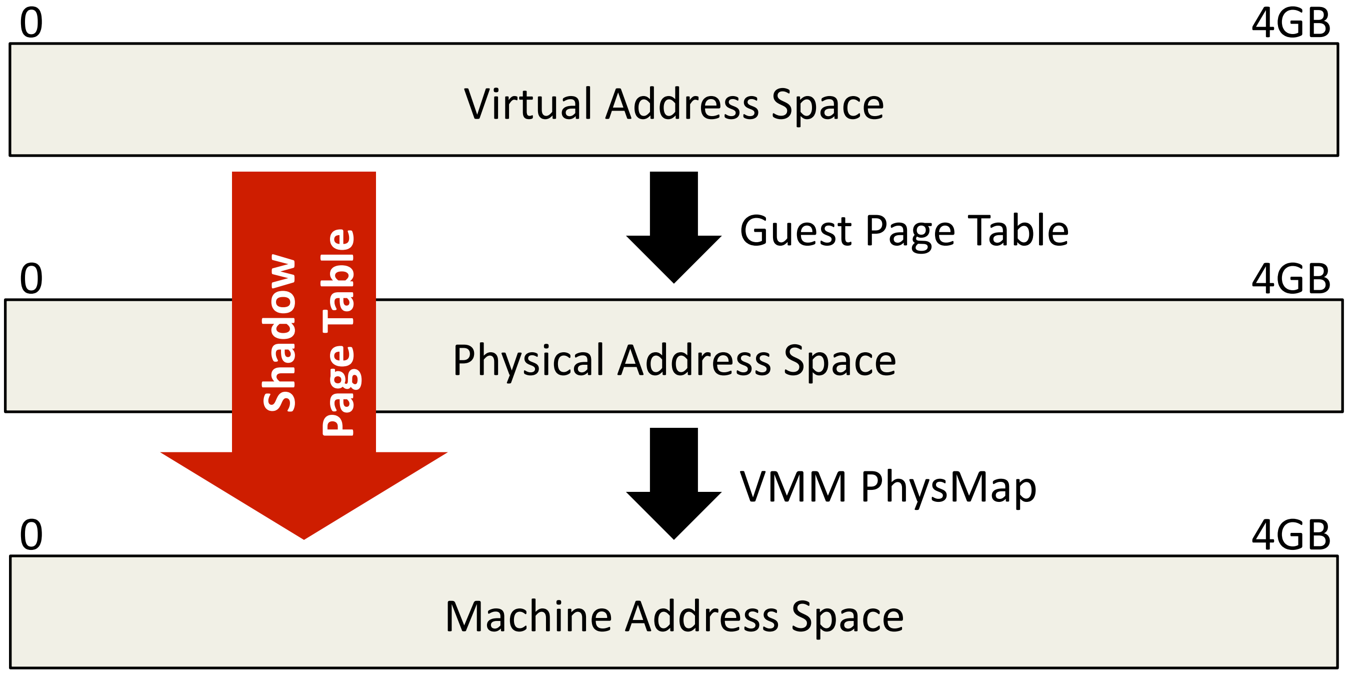
Traditional Address Translation



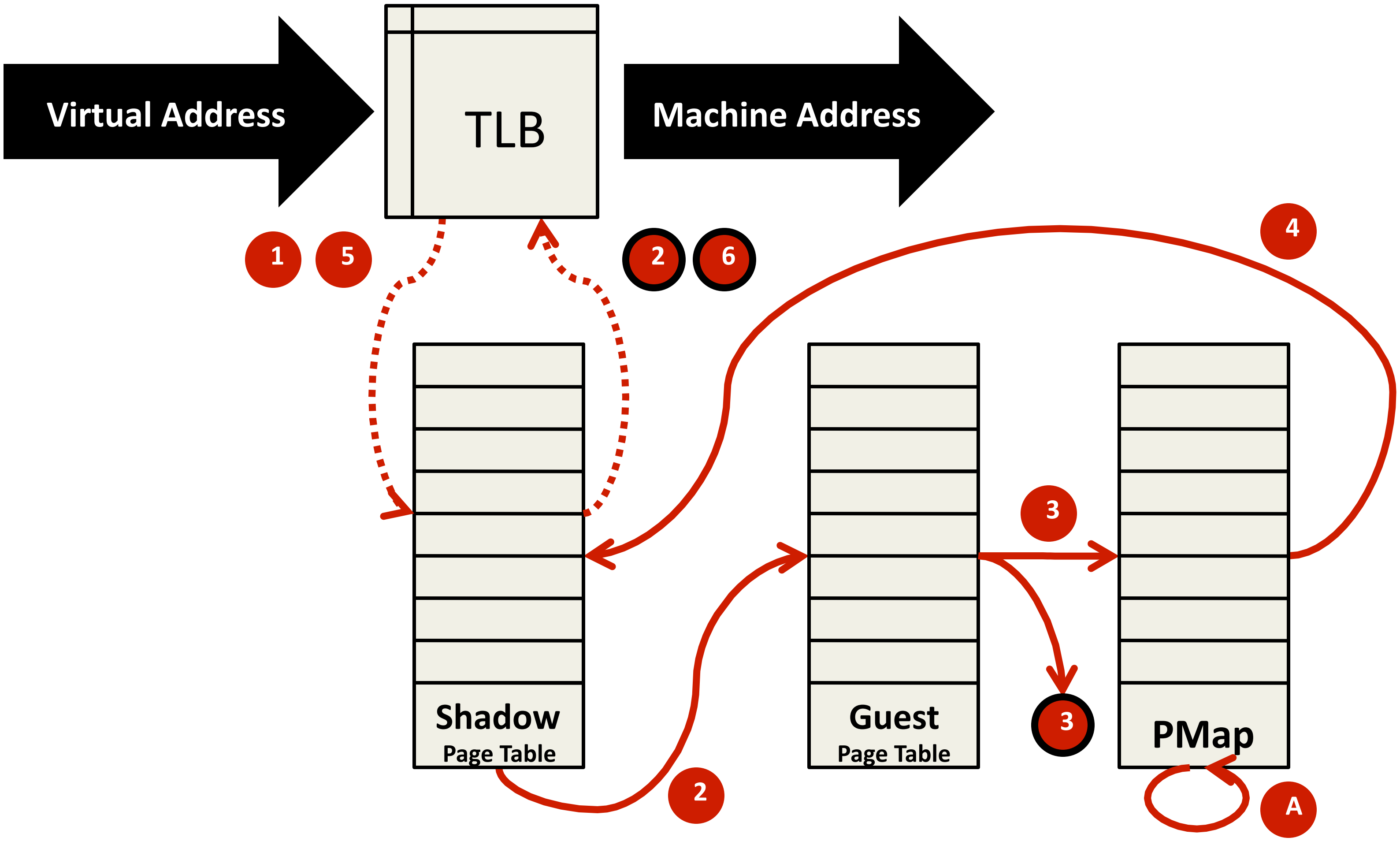
Virtualized Address Spaces



Virtualized Address Spaces w/ Shadow Page Tables



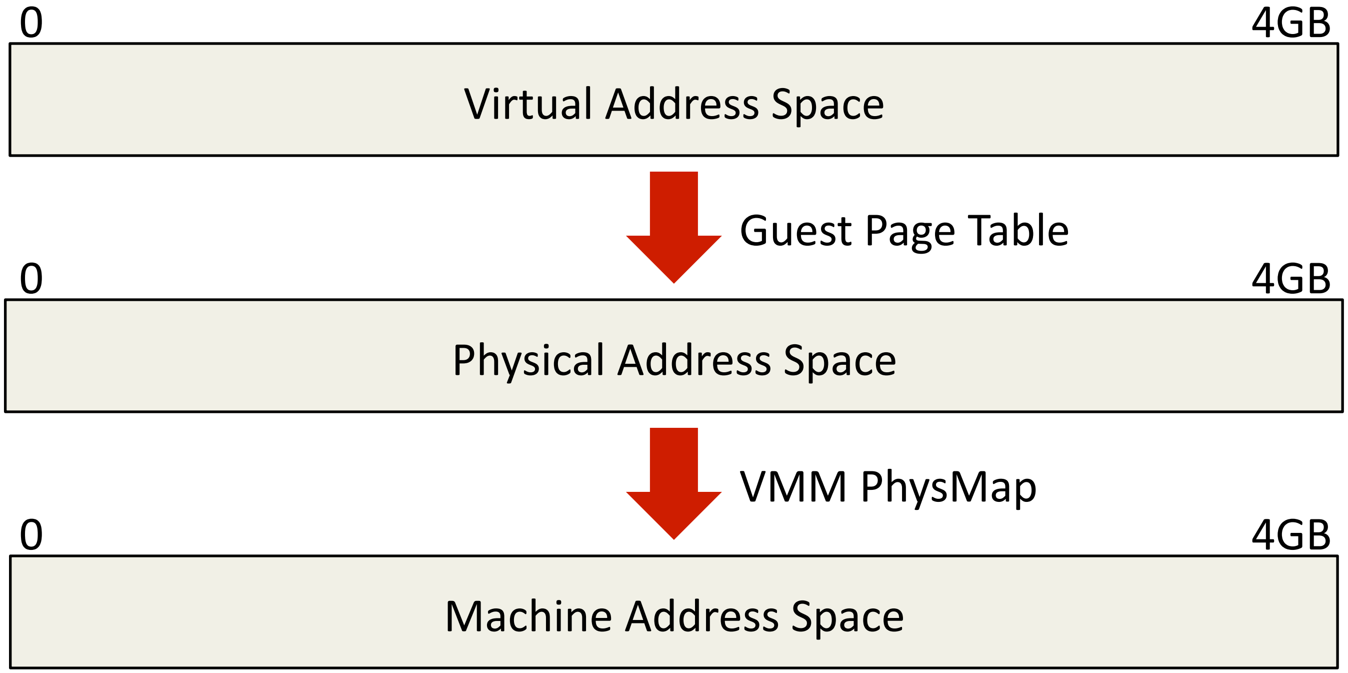
Virtualized Address Translation w/ Shadow Page Tables



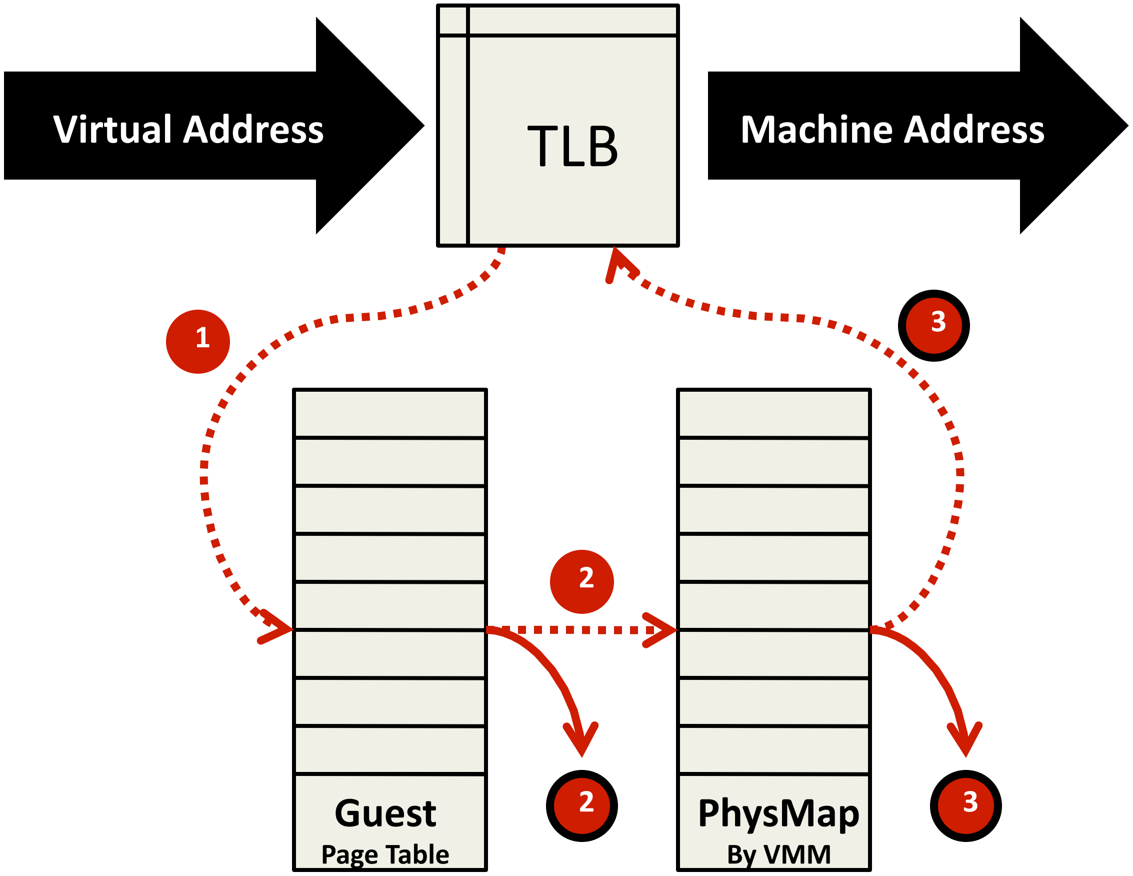
Issues with Shadow Page Tables

- Guest page table consistency
  - Rely on Guest’s need to invalidate TLB
- Performance considerations
  - Aggressive shadow page table caching necessary
  - Need to trace writes to cached page tables

Virtualized Address Spaces w/ Nested Page Tables



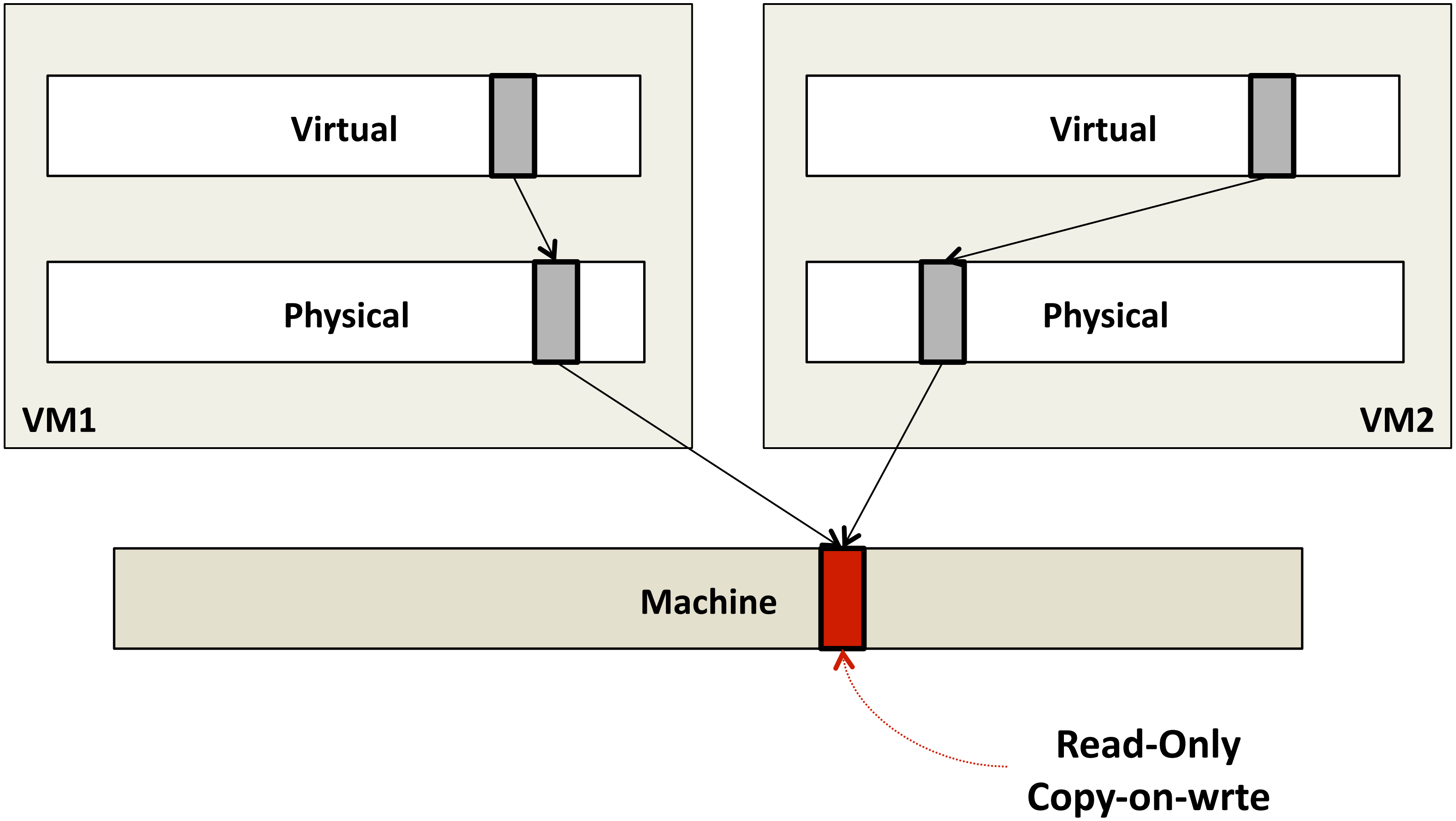
Virtualized Address Translation w/ Nested Page Tables



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