Hardware-assisted virtualization

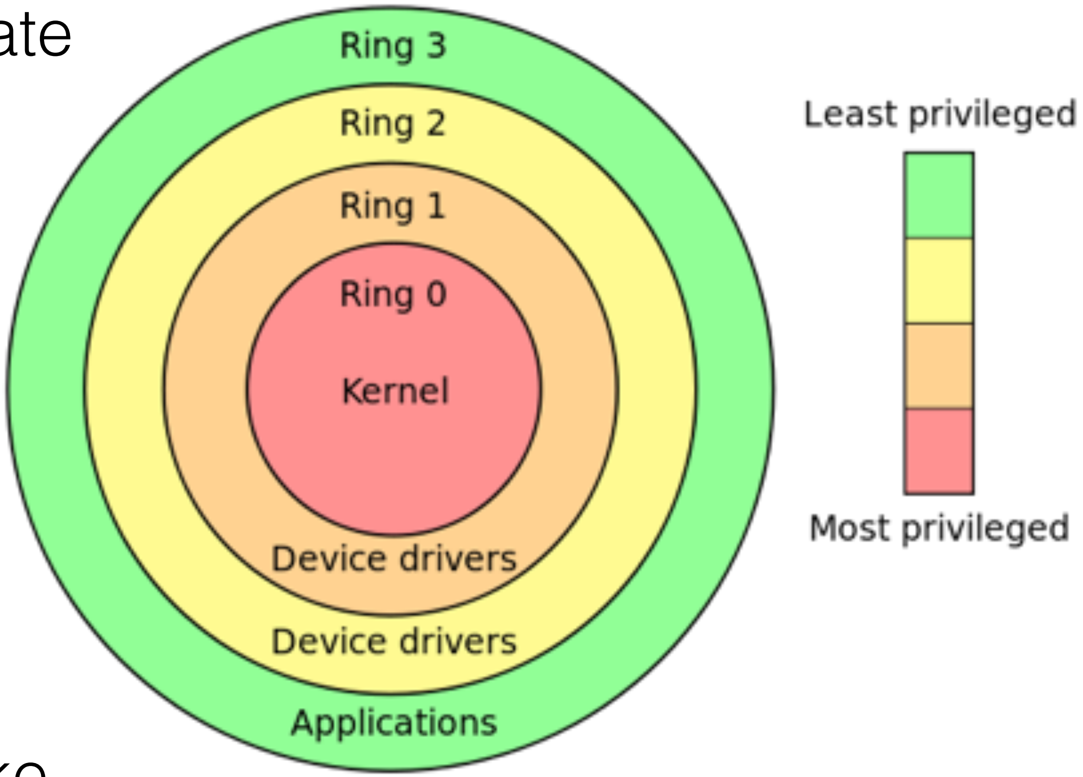
- Why hardware-assisted virtualisation?
  - Higher demand for virtualization
  - Increase performance, lower cost of virtualization
  - Lower Virtual Machine Monitor(VMM) complexity
- Mostly used hardware for virtualization is x86 and maybe soon also ARM

Timeline x86

- Before 2005
    - Binary translation
- After 2005, CPU virtualization
    - Trap and emulate, Intel VT-x, AMD-V
- After 2010, Memory virtualization
    - Second Level Address Translation, Intel Extended Page Table(EPT), AMD Rapid Virtualization Indexing(RVI)
    - Device virtualisation, Intel VT-d, AMD-Vi
- After 2013, CPU virtualization
    - Nested virtual machines, Intel Virtual machine Control Structure(VMCS) shadowing

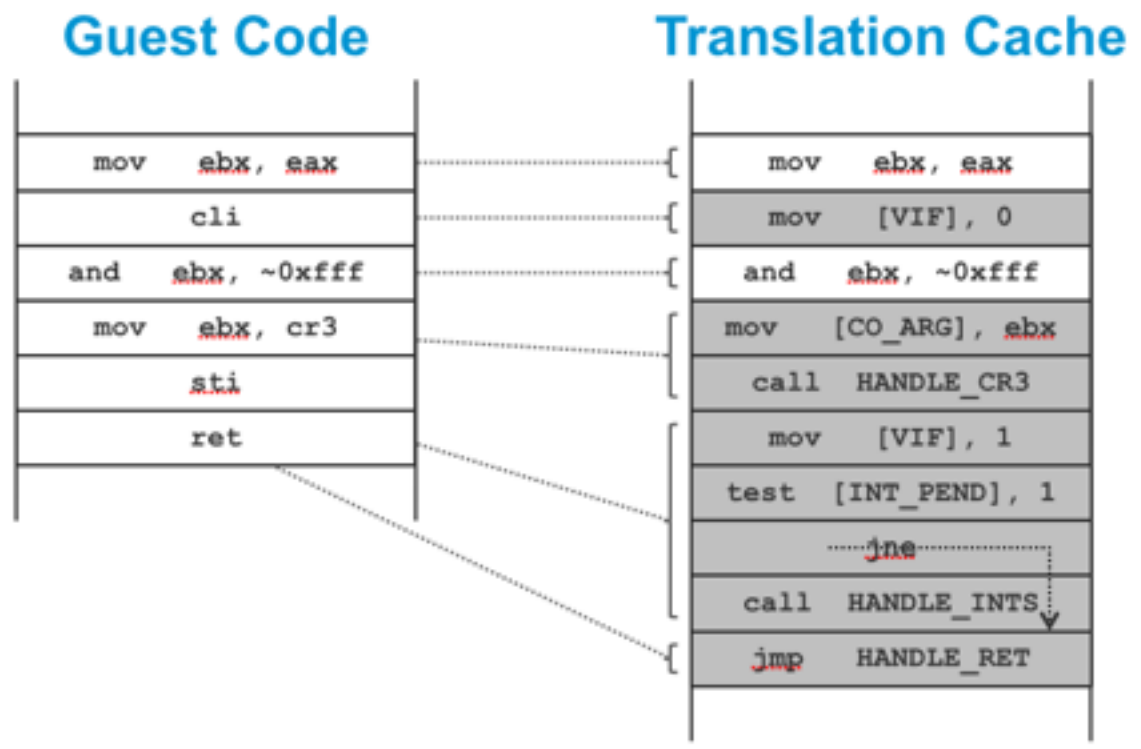
Why can't x86 be classically virtualized?

- Classically means virtualized with trap and emulate
- Visibility of privilege state(Ring, %cs)
- Lack of trap on privileged instructions running at user-level(Ring 3)
  - Example: popf instruction
    - Same instruction behaves differently depending on privileged state
    - User Mode(Ring 3): changes ALU flags like the ZeroFlag(ZF)
    - Kernel Mode(Ring 0): changes ALU and system flags like Interrupt Flag(IF)
    - Does not generate a trap in user mode(Ring 3)



Binary Translation

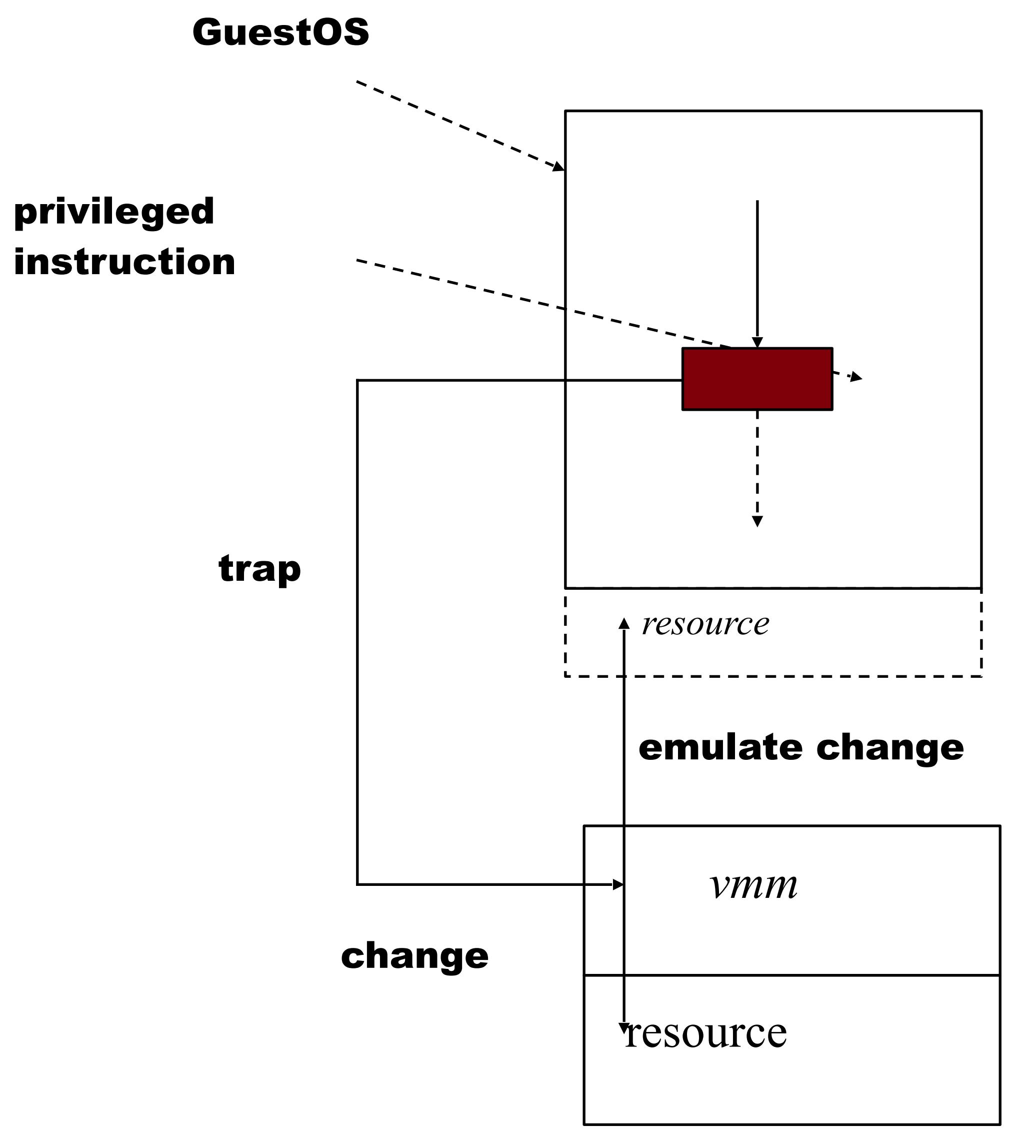
- Interpret the binary code
  - x86 → x86 assembly



```
Guest Code            Translation Cache
mov  ebx, eax         mov   ebx, eax
cli                   mov   [VIF], 0
and  ebx, ~0xfff      and   ebx, ~0xfff
mov  ebx, cr3         mov   [CO_ARG], ebx
sti                   call  HANDLE_CR3
ret                   mov   [VIF], 1
                      test  [INT_PEND], 1
                      jne
                      call  HANDLE_INTS
                      jmp   HANDLE_RET
```

  - Most instructions remain identical, except control flow (calls, jumps, branches, ret, etc.), and privileged instructions
- Avoids traps, which can be expensive
- Translation cache is used to speed up

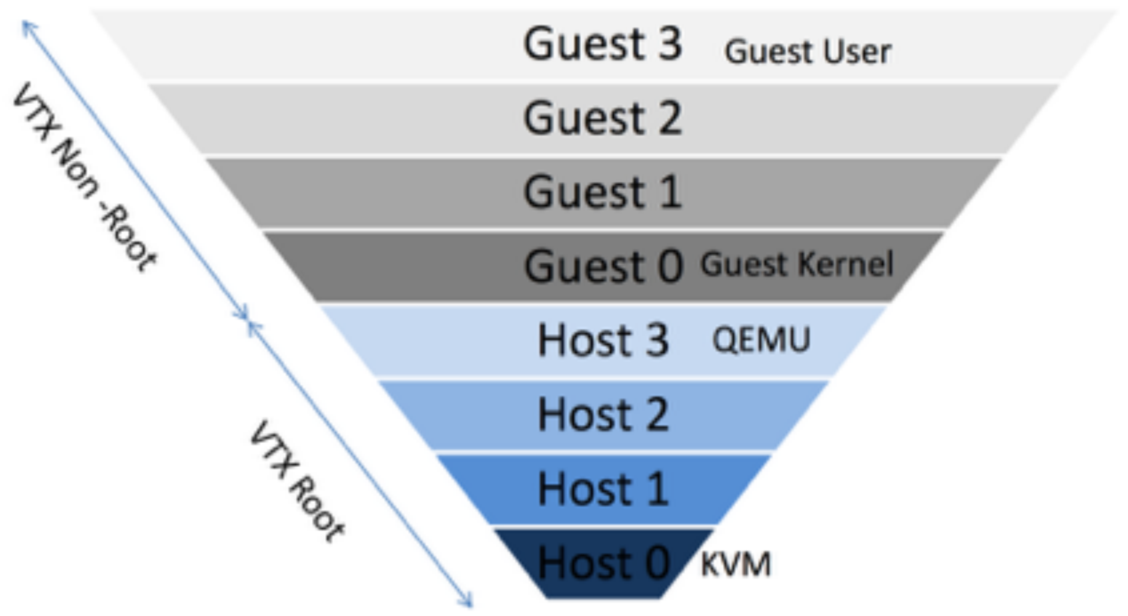
Trap & Emulate



- Run guest VM in unprivileged mode
- Execute guest instructions on real CPU when possible
    - E.g., addl %eax, %ex
- Privileged instructions trap, and VMM emulates
    - E.g., movl %eax, %cr3
    - Traps into VMM so the effect can be emulated

Enable trap and emulate

- A new set of CPU protection rings for guest(non-root) mode in addition to the old host(root) mode



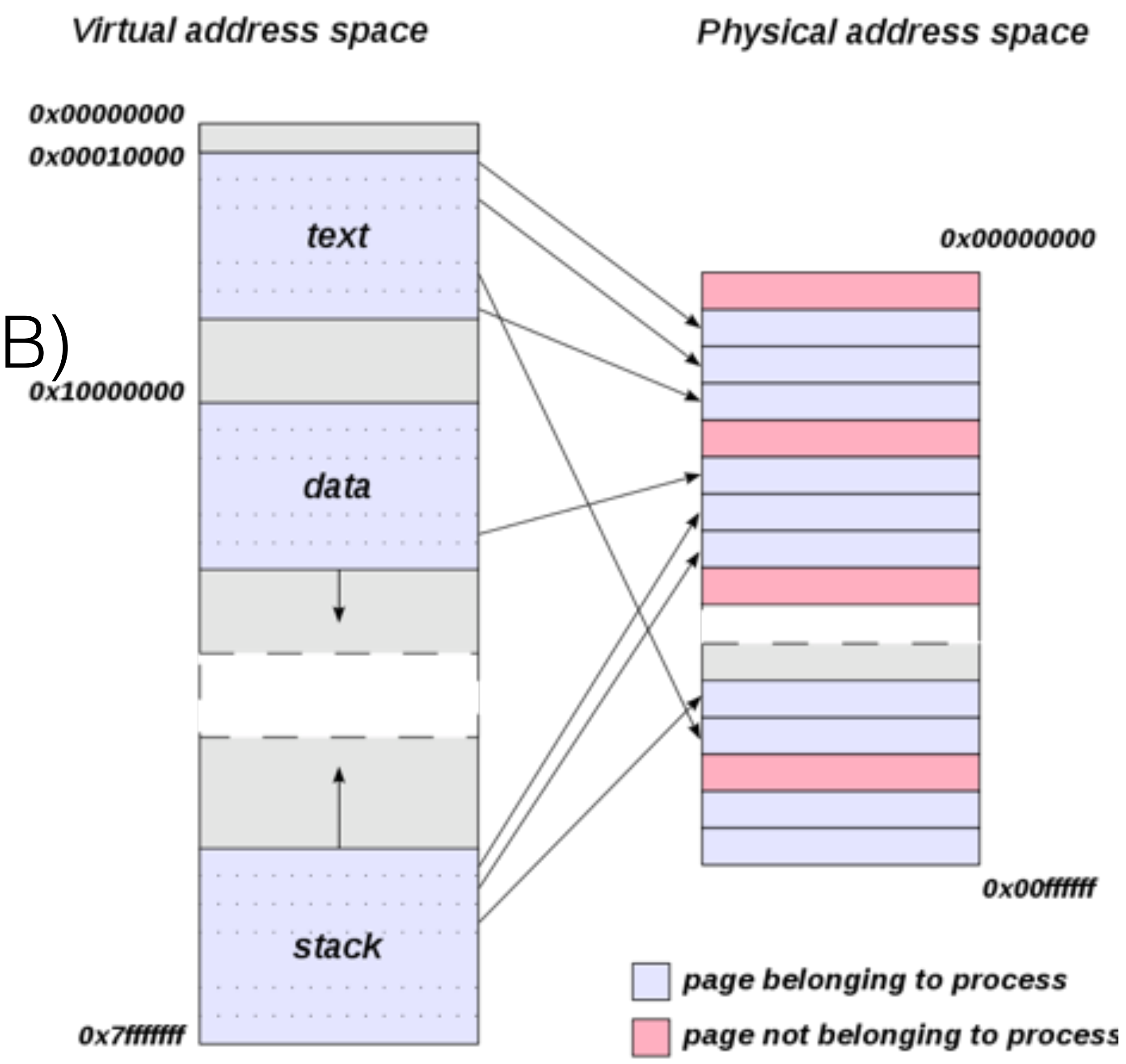
    - New instructions for moving between host and guest mode called “VMRUN” and also instructions for setting the new Virtual Machine Control Structure(VMCS) pointer.
    - VMM fills the VMCS and execute “VMRUN”
- VMM software emulation still needed.

Memory Virtualization

- Traditionally, Host OS fully controls all physical memory space and provides a continuous addressing space(virtual addresses) to each process
- Guest OS is just one of many user space processes, but under VMM control
- In system virtualization, VMM should make all virtual machines share the same physical memory space
  - Before HW support, Shadow Page Tables
  - Second Level Address Translation(SLAT), Intel EPT, AMD RVI
- Virtual memory and MMU

Virtual Memory

- Each process has its own space (usually starting at 0x0)
- A memory page is a fixed length contiguous block (4KB, 2 MB) of data used for memory allocation
- A page table keeps all mapping between the virtual blocks and physical blocks where data is stored. It also contains read, write and execute flags on the blocks.
- Virtual memory enables memory isolation between user processes



Memory Management Unit

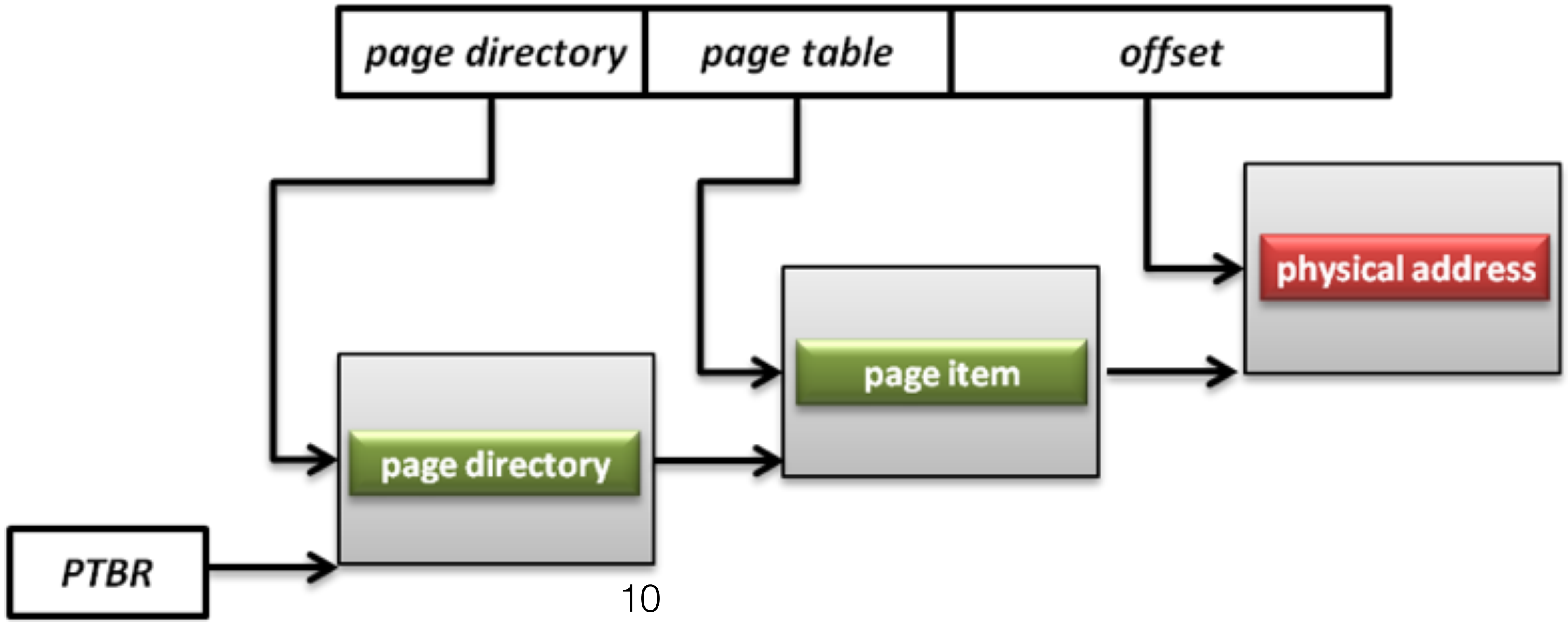
- A hardware component responsible for handling accesses to memory requested by the CPU
  - Address translation: virtual address to physical address (VA to PA)
  - Memory protection(read/write/execute)
  - Cache control
  - Bus arbitration

- The MMU keeps a in-memory(RAM) table called page table that maps logical pages to physical pages.

Page Tables

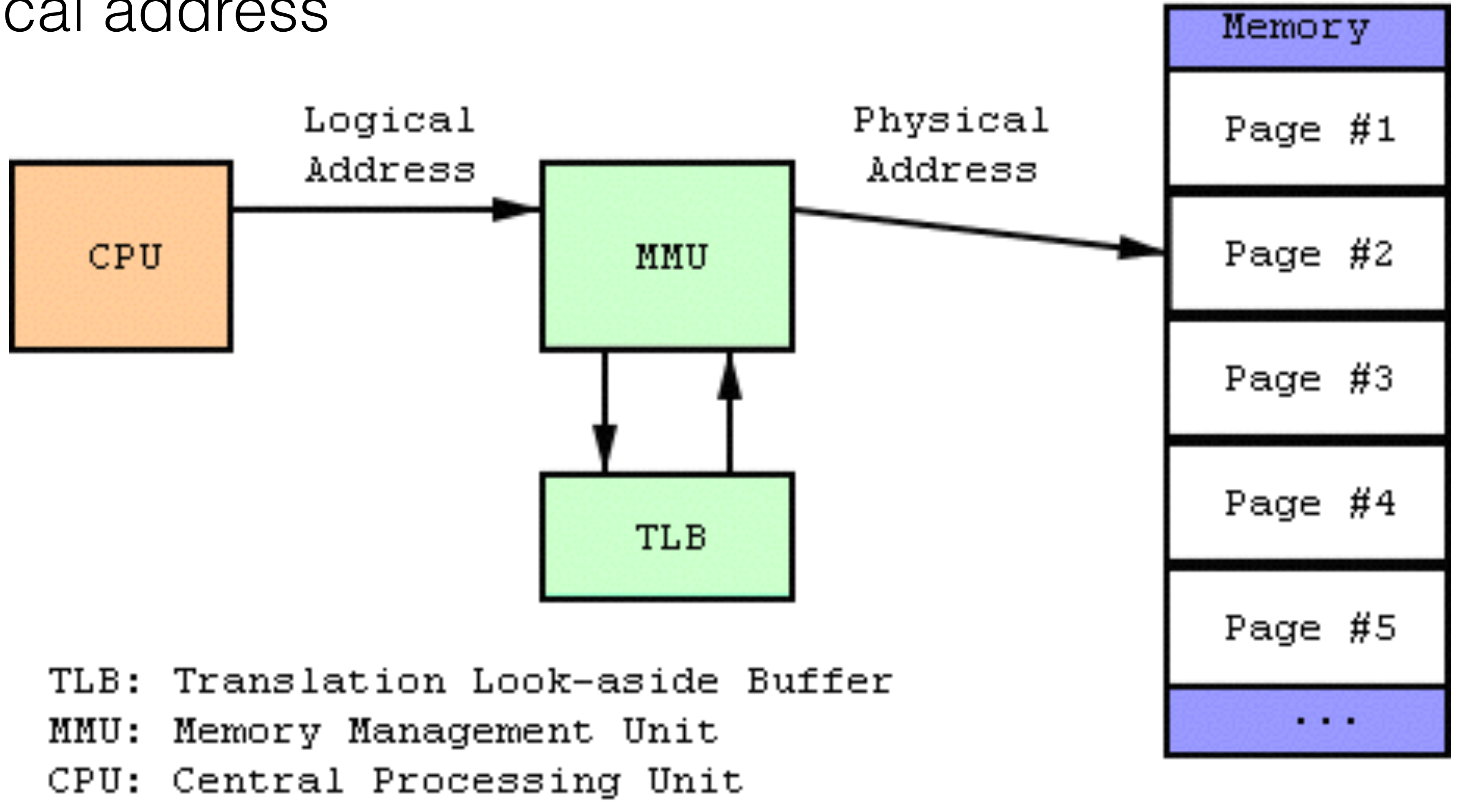
- A page table is the data structure used by a virtual memory system to store the mapping between virtual addresses and physical addresses

- Page table base register(PTBR, %cr3 on x86)
  - Stores the address of the base page table for MMU

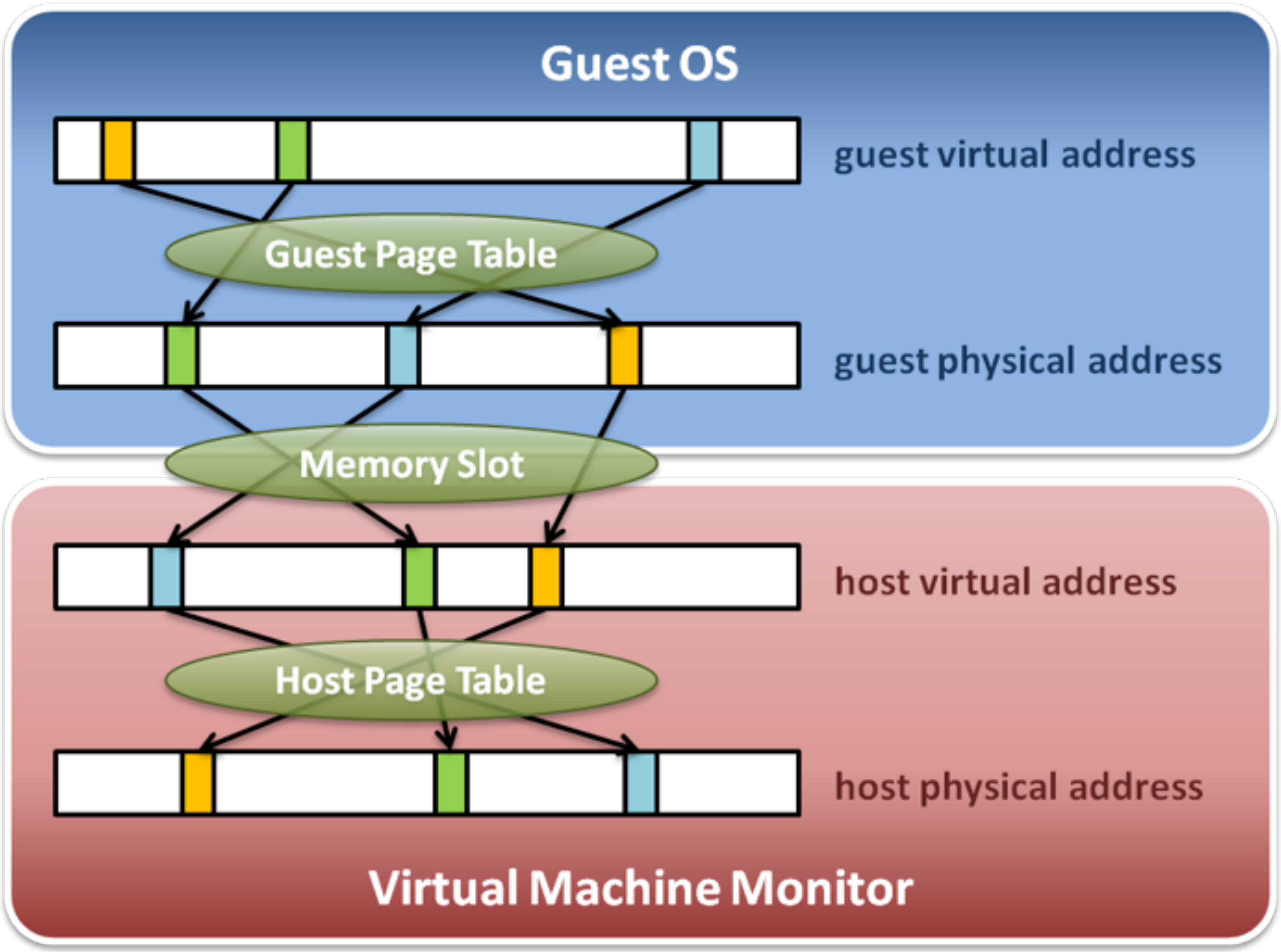


Translation Look-aside Buffer(TLB)

- Translation look-aside buffer
  - A CPU cache that MMU hardware uses to improve virtual address translation speed
  - Avoid accessing and walking the page table in main memory
  - The search key is the virtual address and the search result is a physical address


CPU
Logical Address
MMU
Physical Address
TLB
Memory
Page #1
Page #2
Page #3
Page #4
Page #5
...
TLB: Translation Look-aside Buffer
MMU: Memory Management Unit
CPU: Central Processing Unit

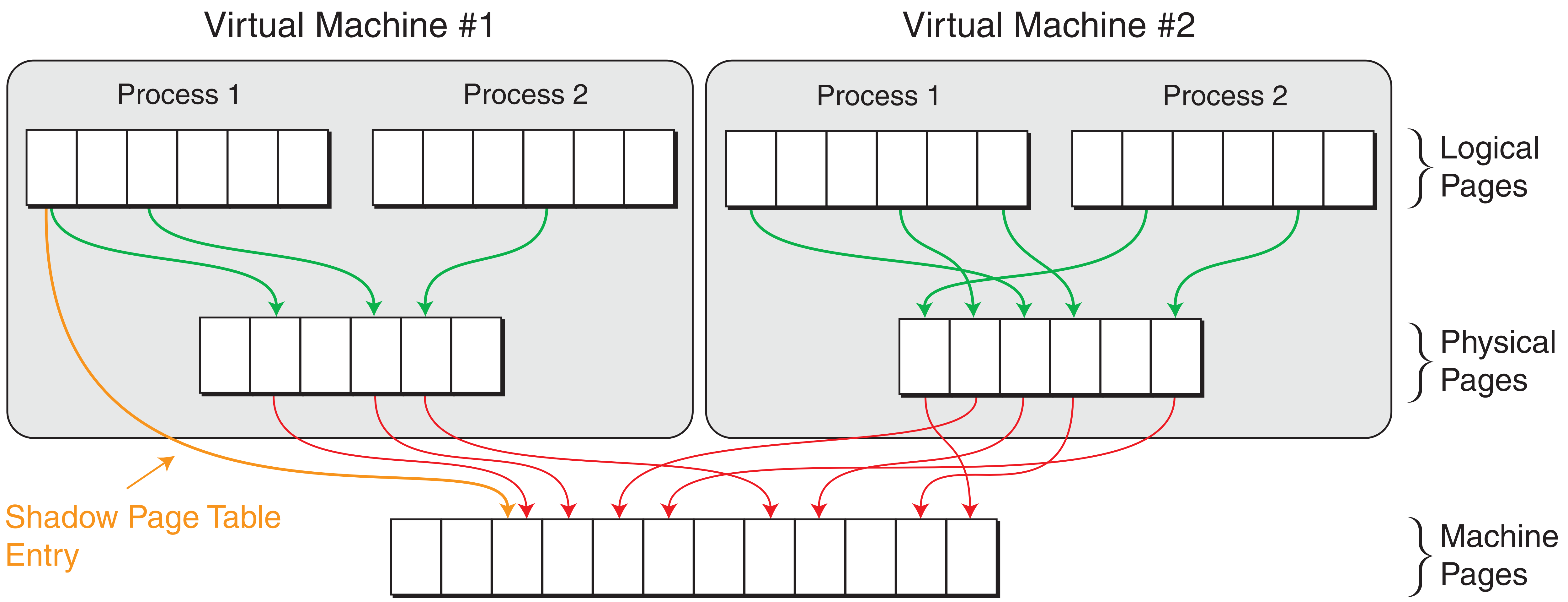
Memory Virtualization Architecture



Software memory virtualization

- VMM creates and maintains page tables that map guest virtual pages directly to machine pages, called the shadow page table
  - Shadow page table is the one used by the MMU
- In each VM, OS creates and manages its own page table
  - Not used by MMU Hardware
- Guest page table is protected from writing with MMU by VMM
  - Manipulation of the guest page table is tracked, and the VMM updates the shadow page table and the guest page table accordingly

Shadow page table

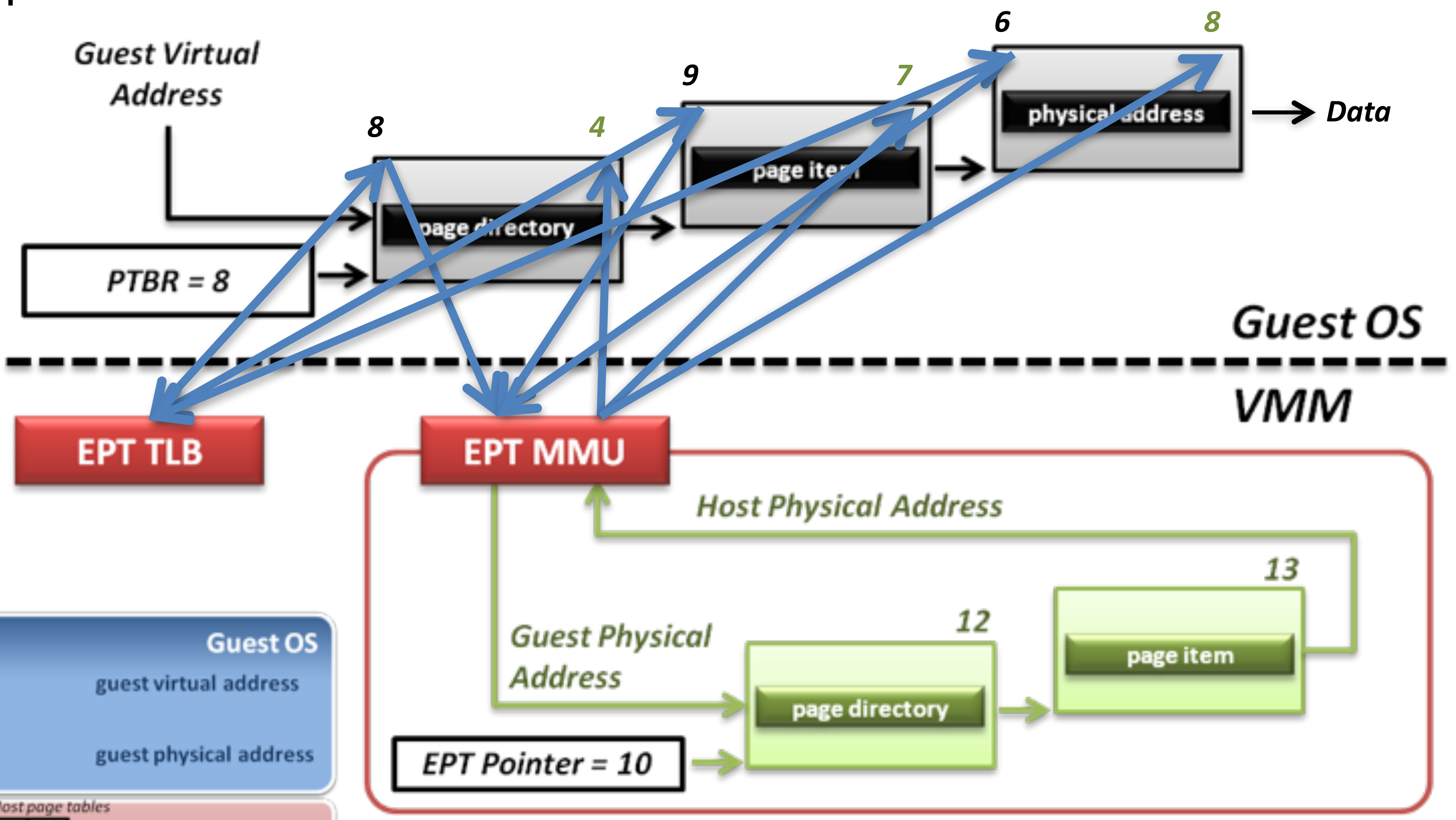


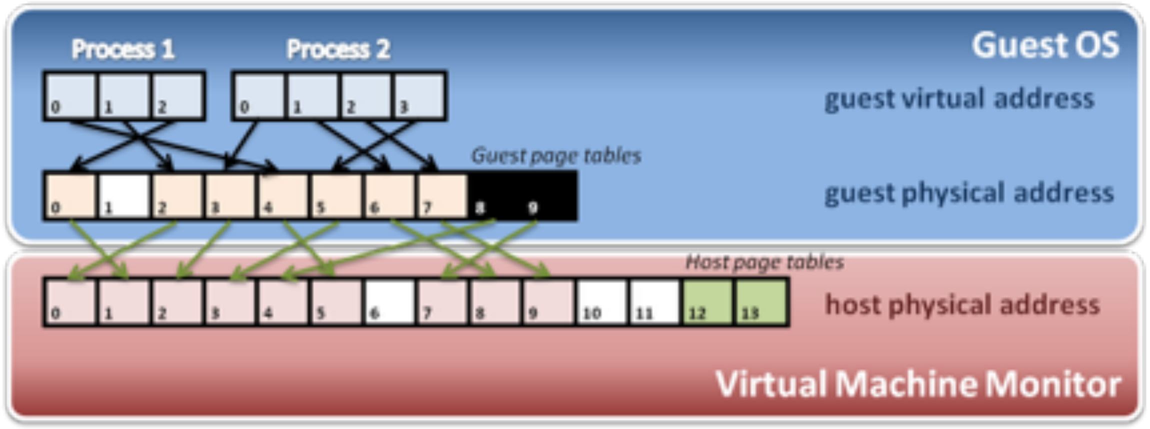
Hardware memory virtualization

- Second Level Address Translation(SLAT), Intel EPT, AMD RVI
  - Shadow page tables now handled by hardware.
  - Two page tables are exposed to hardware
  - The EPT its set with an entry in the VMCS
    - One walker does Guest VA - PA on page table managed by VM
    - One walker does Guest PA - MA on page table managed by VMM
  - TLB miss create extra penalty due to the extra walk in nested page table

Extended Page Table

- Memory operation :





Cost

- Binary translation vs VT-x(2005), VMWare

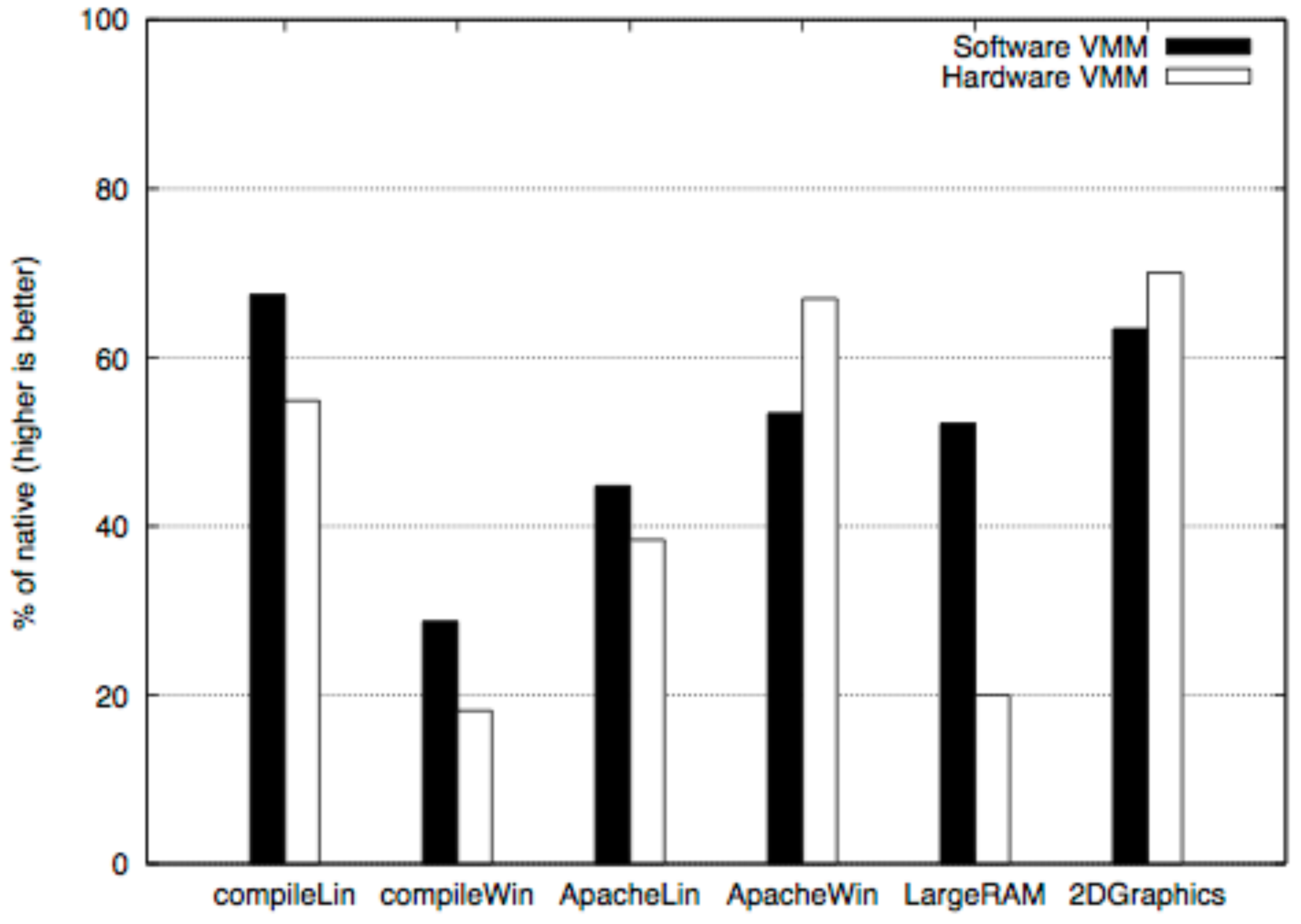
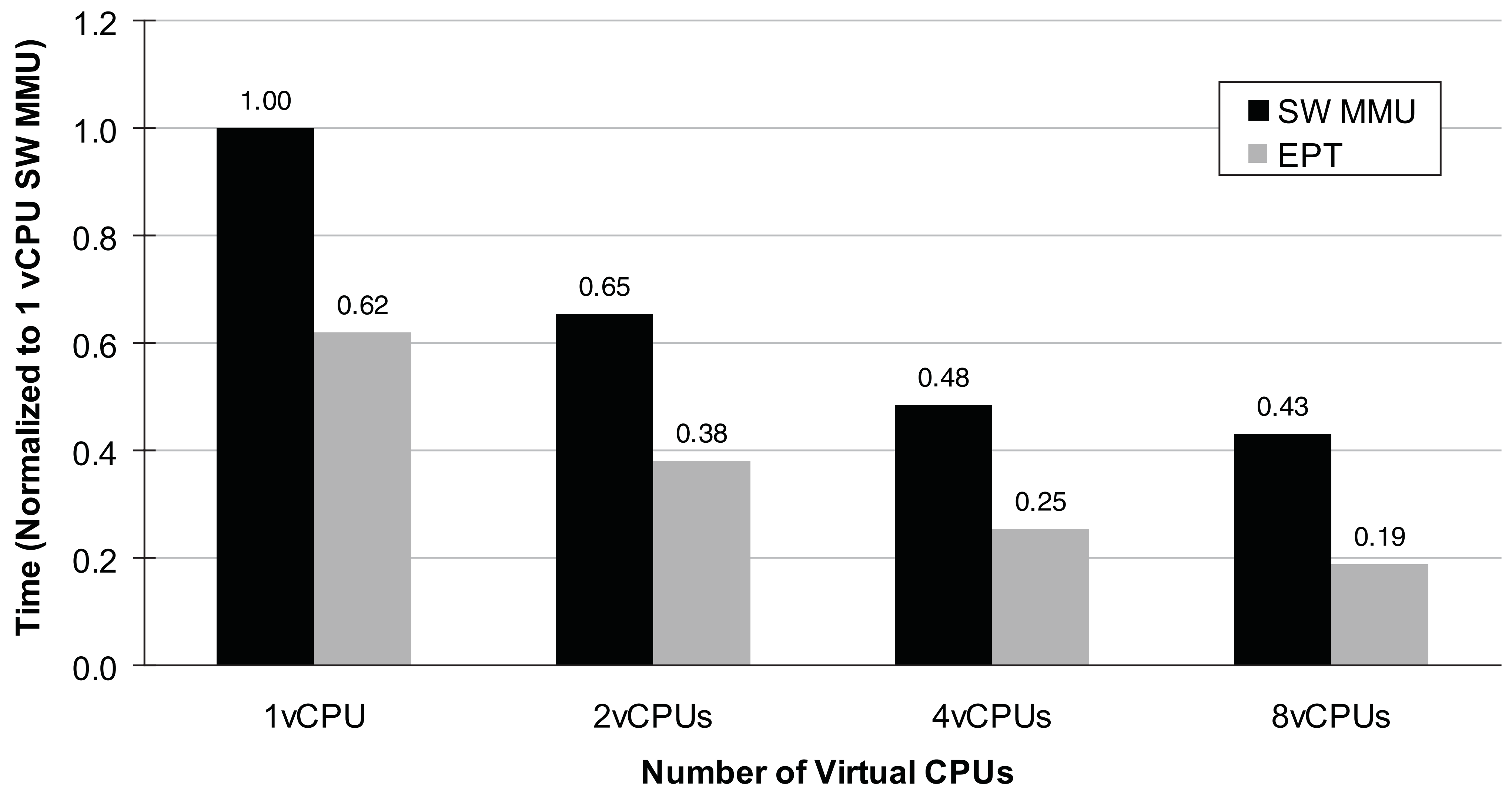


Figure 3. Macrobenchmarks.

Gain

- Second level address translation(EPT) gain

Figure 7. 64-bit Apache Compile Time (Lower is Better)



Cost

- Bare metal comparison 2012, CPU, IPC, filesystem

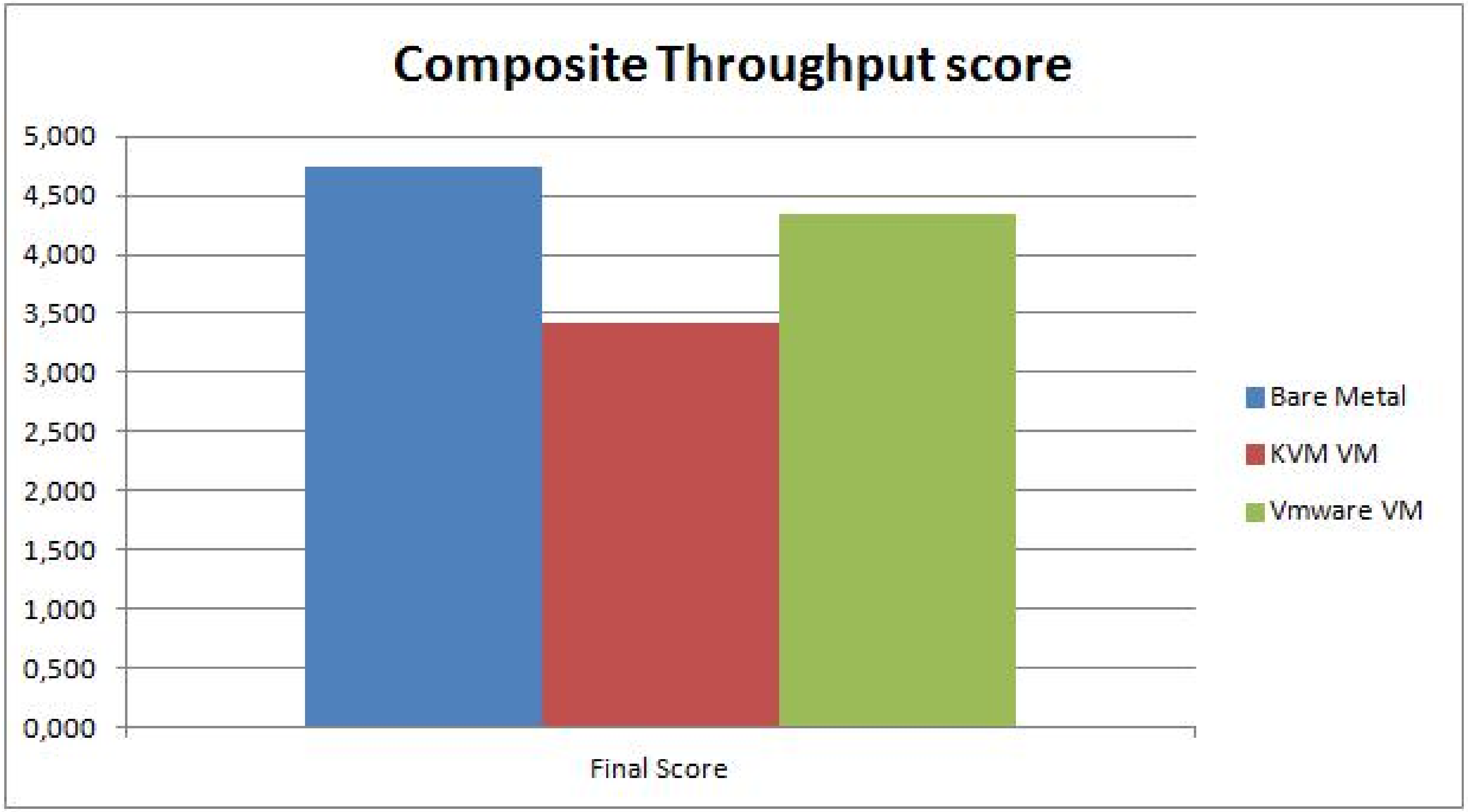
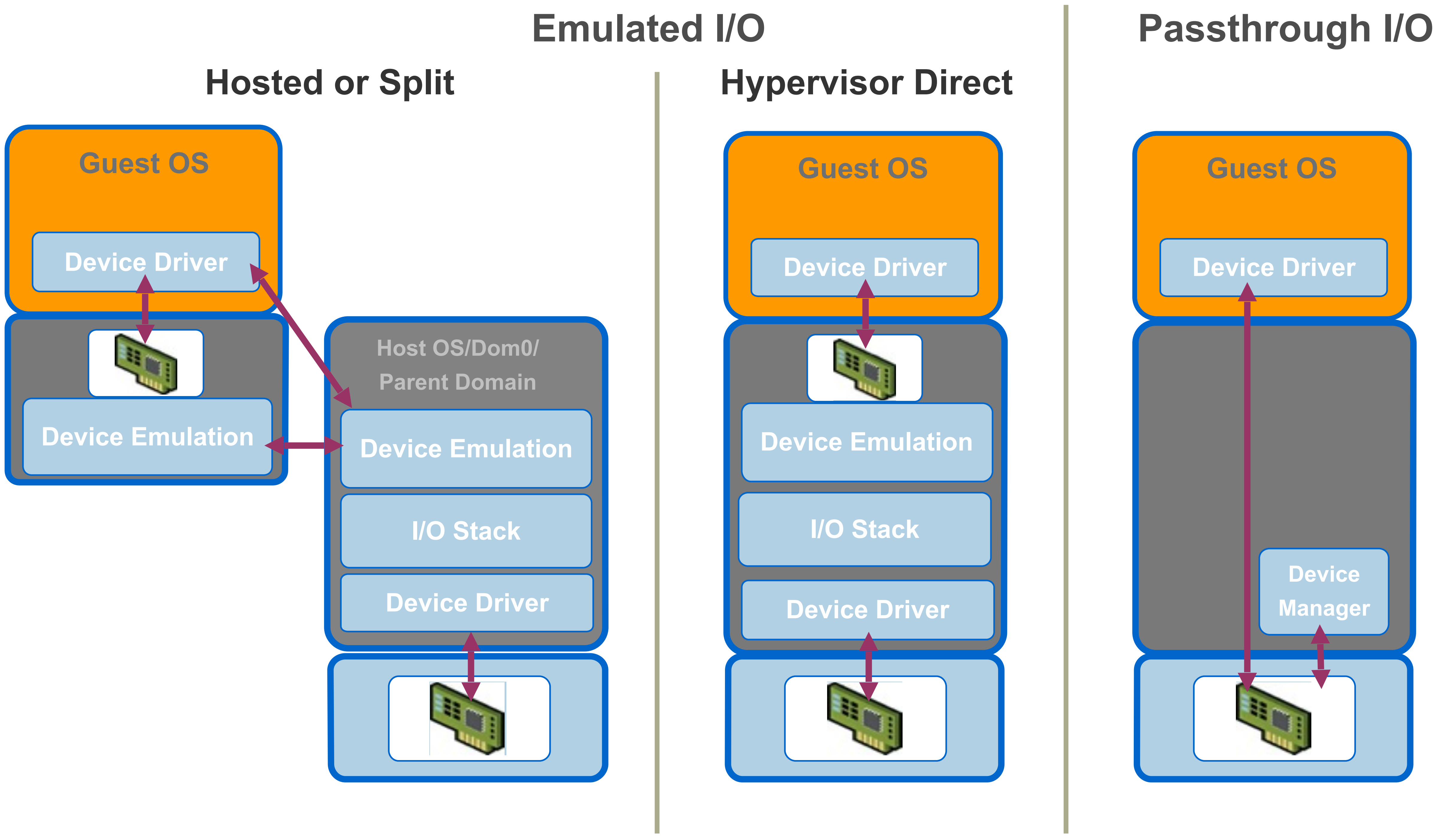


Figure 5.33: UnixBench composite throughput score

Device virtualization

- Needs CPU, chipset and system firmware support
- I/O MMU virtualization(Intel VT-d, AMD-Vi)
  - For full control over devices with DMA and interrupt remapping.
  - Devices on PCI bus must support Function Level Reset(FLR)
- Network virtualization(Intel VT-c)
  - Intel I/O accelerated Technologies for reduction of CPU loads
  - Virtual machine device queues(VMDq)
  - Single root I/O virtualization(SR-IOV)
    - Allows PCIe devices to appear to be multiple separate physical devices, good for NIC.
    - Network interface with support can get up to 95% performance of bare metal.

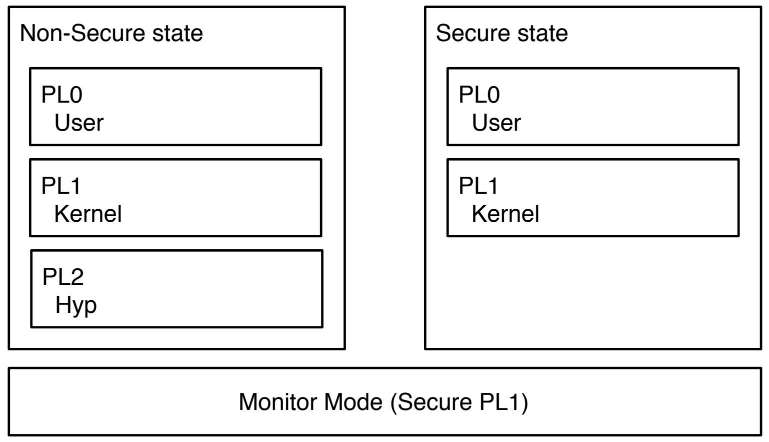
Device Virtualization



Timeline ARM

- Before 2013
    - Binary translation, if any :)
- After 2013
    - Trap and emulate, ARMv7 with extensions and ARMv8

ARM vs x86

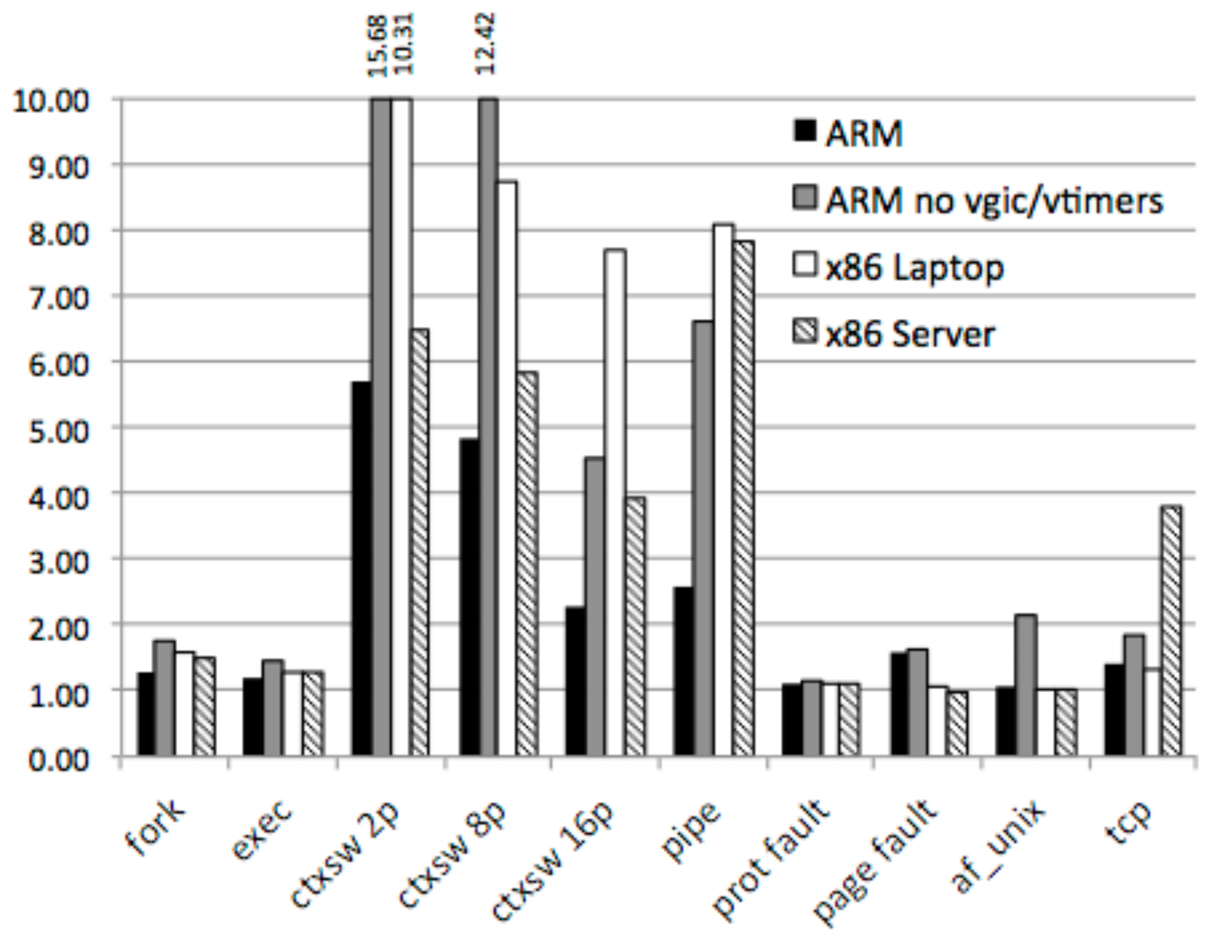


- CPU virtualization
  - Introduces hyp mode below kernel mode.
  - No hardware support for saving and restoring guest states.
- Memory virtualisation
  - More or less the same function as EPT
- I/O virtualization
  - Uses MMU to trap access to non RAM memory
  - x86 uses special instructions(inl, outl) for accessing MMIO

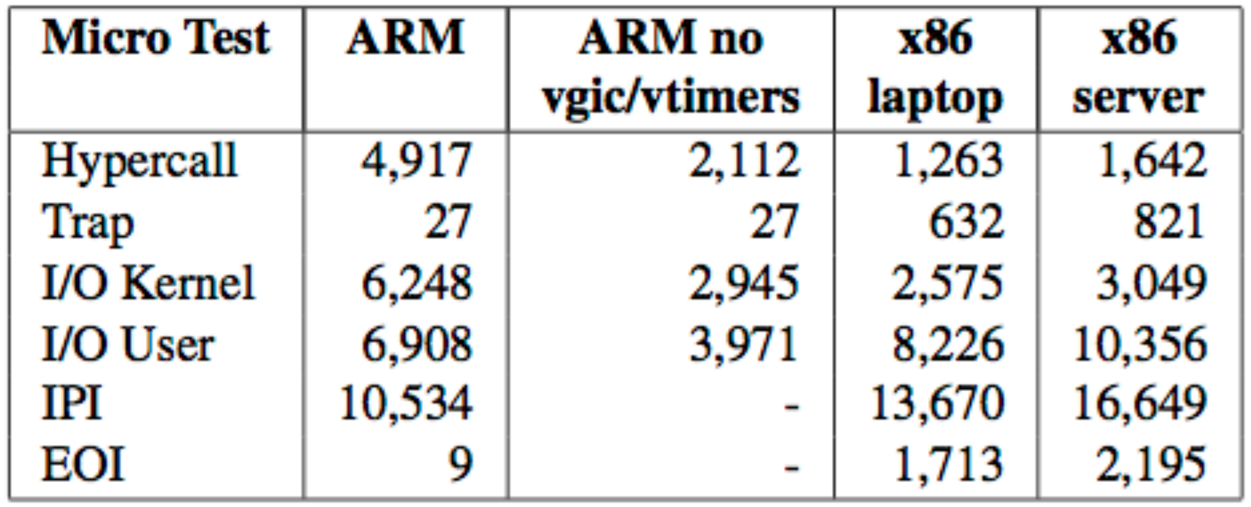
ARM vs x86

- Interrupt virtualization
    - ARM extends the Global interrupt Controller(GIC) with virtualization support(VGIC)
    - VMM can program GIC to trap directly to guest kernel mode for virtual and physical interrupts.
    - Shared device access must trap to hyp mode.
- Timer virtualization
    - Virtual timers and counters.
    - Controlled from guest without trap to hyp mode.

ARM vs x86 cost



(b) SMP VM normalized lmbench performance

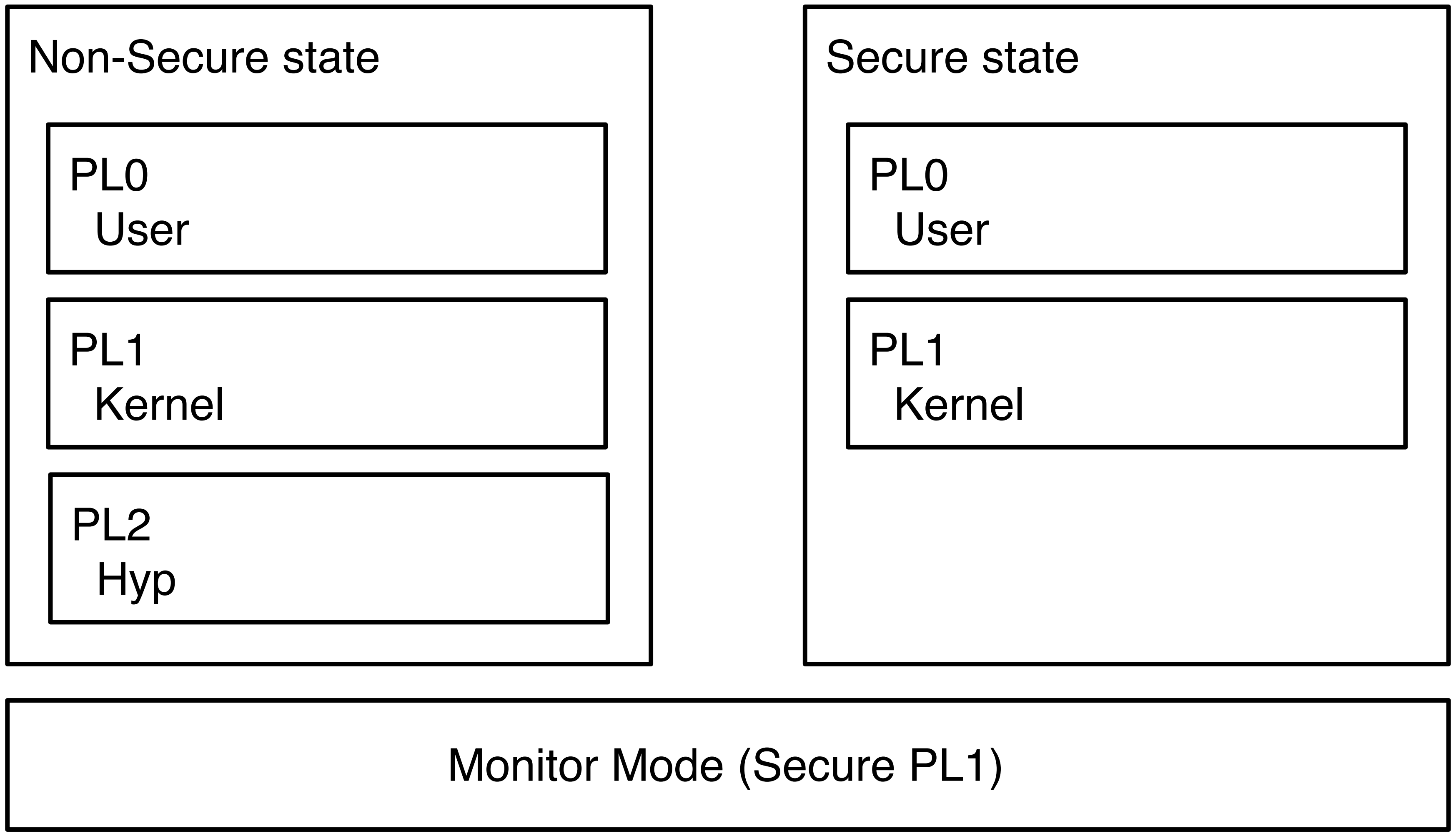


Micro Test	ARM	ARM no vgic/vtimers	x86 laptop	x86 server
Hypercall	4,917	2,112	1,263	1,642
Trap	27	27	632	821
I/O Kernel	6,248	2,945	2,575	3,049
I/O User	6,908	3,971	8,226	10,356
IPI	10,534	-	13,670	16,649
EOI	9	-	1,713	2,195

Table 3: Micro-architectural cycle counts.

Virtual end

ARM vs x86


Non-Secure state
PL0
User
PL1
Kernel
PL2
Hyp
Secure state
PL0
User
PL1
Kernel
Monitor Mode (Secure PL1)