

Cloud Operating Systems

Virtual Memory and Structural Setup

Fabian Rauscher, Daniel Gruss, Andreas Kogler

2023-03-06

Setup

Free pages 994 F9 MemInfo F10 Locks F11 Stacktrace F12 Threads

This is on term 0, you should see me now
Kernel end address is 0xffffffff80165000
Now enabling Interrupts...

SWEB-Pseudo-Shell starting...

SWEB: />

- Upstream: <https://github.com/IAIK/sweb>
- SWEB Tutorials: <https://www.iaik.tugraz.at/teaching/materials/os/tutorials/>
- Useful links in the Discord `cloudos-announcements` channel
- We recommend you work on Linux



Building and running SWEB

- `mkdir -p /tmp/sweb`
- `cd /tmp/sweb`
- `cmake /path/to/sourcecode/of/sweb`
- `make`
- `make kvm`



- Condition (condition variable)
 - Mutex
 - PageManager (allocate and free pages)
 - ArchMemory (manages virtual address space)
 - main (boot code)
 - Syscall (syscall handling)
 - uSTL (STL, provides vector, string, map,...)
 - Loader (user space binary loader)
- For more info:
<https://www.iaik.tugraz.at/teaching/materials/os/tutorials/>

VMX



- Intel's Virtual-Machine Extension
- Provides hardware support for virtualization
- Two different classes of software:
 - Virtual-machine monitors (VMM)
 - Guest software



- DO NOT READ IT LIKE A BOOK
- Look things up that you need
- Important volumes for this lecture:
 - Volume 3C: virtual machine extensions (vmx)
 - Volume 3D: appendices with values for constants
 - Volume 3A: operating-system support environment



- Gives the guest the illusion of running on real hardware
- Has full control of the processors resources
- Manages the ability of the guest to access
 - Processor Resources
 - Physical Memory
 - Interrupts
 - I/O
 - ...

Getting Started



- Check for VMX support: $\text{CPUID.1:ECX.VMX}[\text{bit } 5] = 1$
- Enable VMX by setting bit 13 in CR4 ¹

¹Intel SDM Volume 3, 24.7



- VMX root operation
 - new instructions (VMX instructions)
 - restrictions on certain control register values
 - used for the VMM
- VMX non-root operation
 - more restricted than normal operation
 - certain instructions and events cause VM exits
 - used for the guest
- Transition between the two
 - VM entries: VMX root operation \rightarrow VMX non-root operation
 - VM exits: VMX non-root operation \rightarrow VMX root operation

Entering VMX Operation (Part 1)



Preparations

- Setup CR0¹
 - Set bits specified by IA32_VMX_CR0_FIXED0 (0x486)
 - Clear bits specified by IA32_VMX_CR0_FIXED1 (0x487)
- Setup CR4¹
 - Set bits specified by IA32_VMX_CR4_FIXED0 (0x488)
 - Clear bits specified by IA32_VMX_CR4_FIXED1 (0x489)
- **Ensure** that bit 2 of IA32_FEATURE_CONTROL (0x3A) is set²

¹Intel SDM Volume 3, 24.8

²Intel SDM Volume 3, 24.7



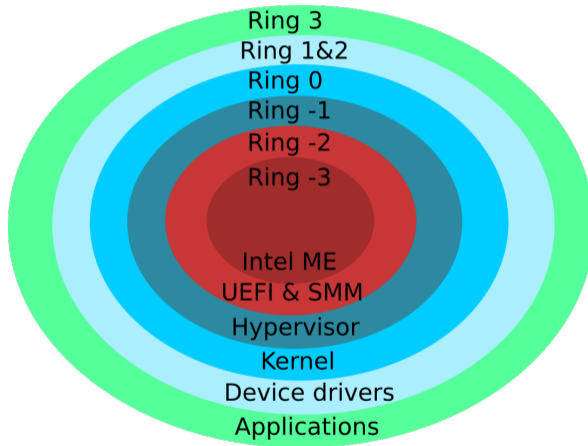
VMXON region ²

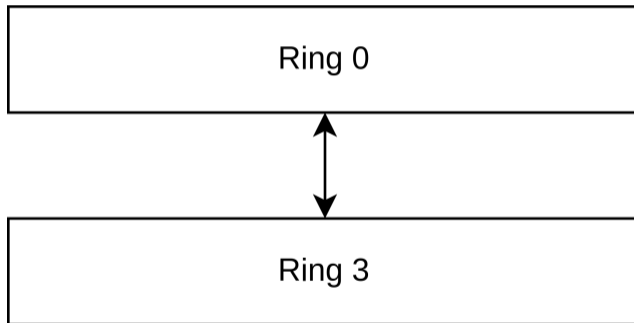
- Used by the processor to support VMX operation
- Up to 4KB in size
- VMXON pointer needs to be a 4KB aligned valid physical address
- Bits 30:0 must contain the VMCS revision identifier (IA32_VMX_BASIC, 0x480)
- Can be loaded using the VMXON instruction to enter VMX operation

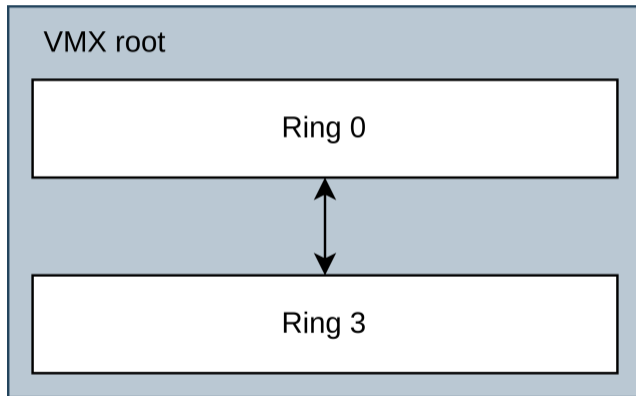
¹Intel SDM Volume 3, 31

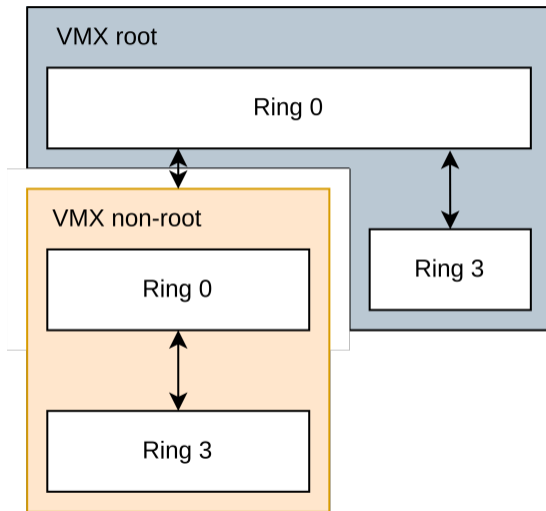


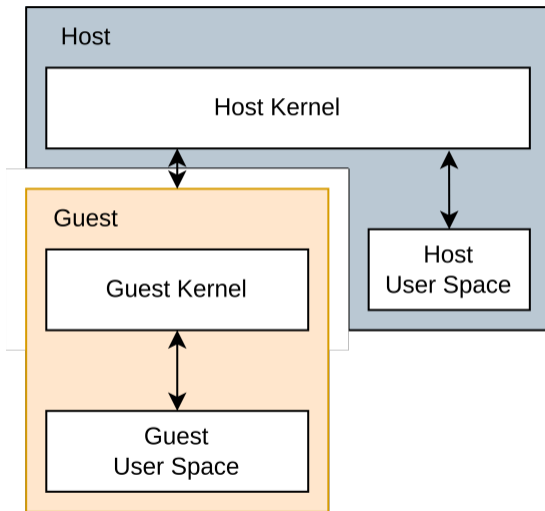
- VMXON
- VMXOFF
- INVEPT
- INVVPID
- VMCLEAR
- VMPTRLD
- VMPTRSTR
- VMLAUNCH/VMRESUME
- VMREAD
- VMWRITE
- VMCALL

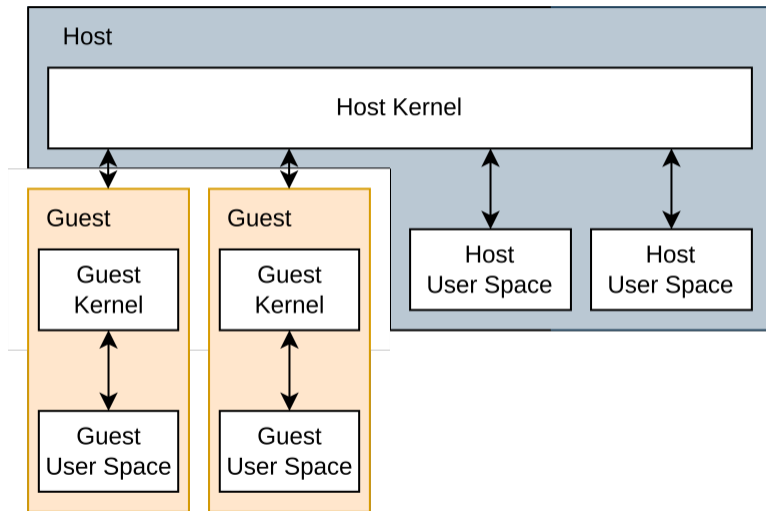


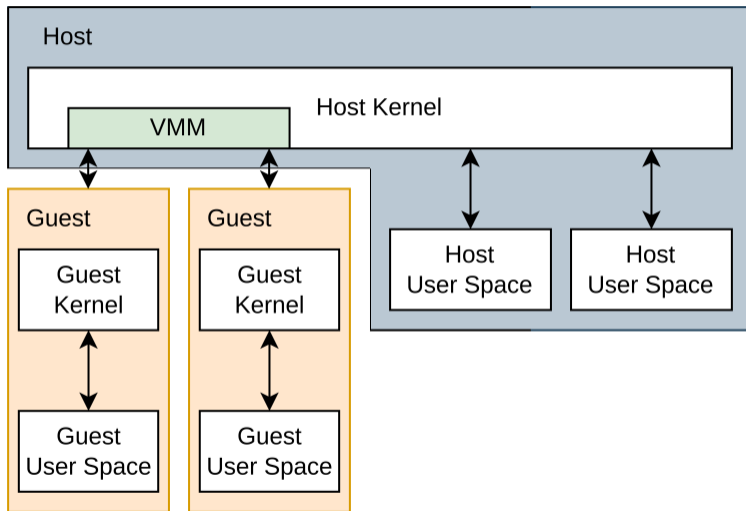


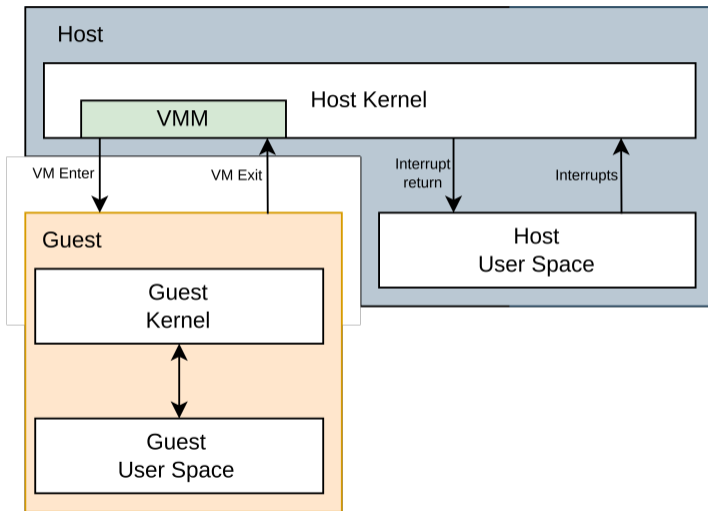




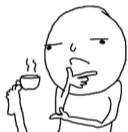








VMCS



Virtual-Machine Control Data Structure (VMCS)

- Manages VM entries and VM exits
- Used to setup processor behavior in VMX non-root operation
- Can be manipulated using VMCLEAR, VMPTRLD, VMREAD, VMWRITE
- One VMCS per virtual processor
- The current VMCS can be accessed using the VMWRITE and VMREAD instructions

²Intel SDM Volume 3, 25.2



- Up to 4KB in size
- VMCS pointer needs to be a 4KB aligned valid physical address
- Bits 30:0 must contain the VMCS revision identifier (IA32_VMX_BASIC, 0x480)¹

¹Intel SDM Volume 3, 25.2



- Guest configuration
- Some registers are not easy to restore
 - CR4/CR0
 - Segment bases and access rights
 - CR3
 - ...

- Natural-Width fields.
- 16-bits fields.
- 32-bits fields.
- 64-bits fields.

Copyright 2017, [@Noteworthy](#) (Intel Manual of July 2017)

CONTROL FIELDS

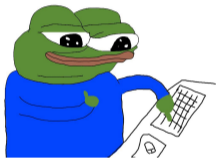
Pin-Based VM-Execution Controls	External-interrupt exiting	NMI exiting		Virtual NMIs	
	Activate VMX-preemption timer		Process posted interrupts		
Primary processor-based VM-execution controls	Interrupt-window exiting		Use TSC offsetting		
	HLT exiting	INVLPG exiting	MWAIT exiting	RDPMS exiting	
	RDTSC exiting	CR3-load exiting	CR3-store exiting	CR8-load exiting	
	CR8-store exiting	Use TPR shadow	NMI-window exiting	MOV-DR exiting	
	Unconditional I/O exiting	Use I/O bitmaps	Monitor trap flag	Use MSR bitmaps	
	MONITOR exiting		PAUSE exiting	Activate secondary controls	
Secondary processor-based VM-execution controls	Virtualize APIC accesses	Enable EPT	Descriptor-table exiting	Enable RDTSCP	
	Virtualize x2APIC mode	Enable VPID	WBINVD exiting	Unrestricted guest	
	APIC-register virtualization		Virtual-interrupt delivery	PAUSE-loop exiting	
	RDRAND exiting	Enable INVPCID	Enable VM functions	VMCS shadowing	
	Enable ENCLS exiting	RDSEED exiting	Enable PML	EPT-violation #VE	
	Conceal VMX non-root operation from Intel PT			Enable XSAVES/XRSTORS	
	Mode-based execute control for EPT			Use TSC scaling	
Exception Bitmap		I/O-Bitmap Addresses		TSC-offset	
Guest/Host Masks for CR0		Guest/Host Masks for CR4	Read Shadows for CR0	Read Shadows for CR4	
CR3-target value 0	CR3-target value 1	CR3-target value 2	CR3-target value 3	CR3-target count	
APIC Virtualization	APIC-access address		Virtual-APIC address		TPR threshold
	EOI-exit bitmap 0	EOI-exit bitmap 1	EOI-exit bitmap 2	EOI-exit bitmap 3	
	Posted-interrupt notification vector			Posted-interrupt descriptor address	
Read bitmap for low MSRs		Read bitmap for high MSRs	Write bitmap for low MSRs	Write bitmap for low MSRs	
Executive-VMCS Pointer		Extended-Page-Table Pointer		Virtual-Processor Identifier	
PLE_Gap	PLE_Window	VM-function controls	VMREAD bitmap	VMWRITE bitmap	
ENCLS-exiting bitmap			PML address		
Virtualization-exception information address		EPTP index		XSS-exiting bitmap	

GUEST STATE AREA

CR0	CR3			CR4	
DR7					
RSP	RIP			RFLAGS	
CS	Selector	Base Address	Segment Limit	Access Right	
SS	Selector	Base Address	Segment Limit	Access Right	
DS	Selector	Base Address	Segment Limit	Access Right	
ES	Selector	Base Address	Segment Limit	Access Right	
FS	Selector	Base Address	Segment Limit	Access Right	
GS	Selector	Base Address	Segment Limit	Access Right	
LDTR	Selector	Base Address	Segment Limit	Access Right	
TR	Selector	Base Address	Segment Limit	Access Right	
GDTR	Selector	Base Address	Limit	Access Right	
IDTR	Selector	Base Address	Limit	Access Right	
IA32_DEBUGCTL	IA32_SYSENTER_CS	IA32_SYSENTER_ESP		IA32_SYSENTER_EIP	
IA32_PERF_GLOBAL_CTRL	IA32_PAT	IA32_EFER		IA32_BNDCFGS	
SMBASE					
Activity state	Interruptibility state				
Pending debug exceptions					
VMCS link pointer					
VMX-preemption timer value					
Page-directory-pointer-table entries	PDPTE0	PDPTE1	PDPTE2	PDPTE3	
Guest interrupt status					
PML index					

HOST STATE AREA

CRO		CR3		CR4	
RSP			RIP		
CS	Selector				
SS	Selector				
DS	Selector				
ES	Selector				
FS	Selector	Base Address			
GS	Selector	Base Address			
TR	Selector	Base Address			
GDTR	Base Address				
IDTR	Base Address				
IA32_SYSENTER_CS		IA32_SYSENTER_ESP		IA32_SYSENTER_EIP	
IA32_PERF_GLOBAL_CTRL		IA32_PAT		IA32_EFER	



- What about things that are not in the guest state area, like general-purpose registers?
 - We have to save and restore them by hand!
- We need a context switch

VMCS Some Important Host Fields

- Host EFER
- Host PAT
- Host CRx
- Host RSP
- Host RIP
- Host segment selectors
- TR selector and base address
- GDTR and IDTR base address

VMCS Some Important Guest Fields

- Guest EFER
- Guest PAT
- Guest CRx
- Guest RSP
- Guest RIP
- Guest RFLAGS
- Guest segment selectors, base addresses, limits, access rights
- GDTR and IDTR base address, limit
- VMCS link pointer (set it to -1)

VMCS Some Important Control Fields

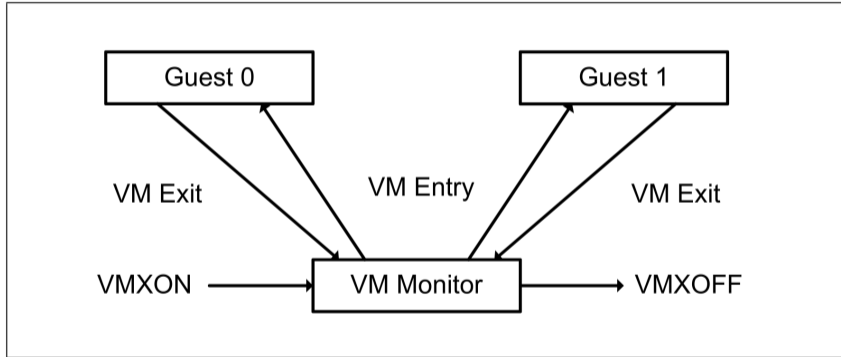
- Pin-Based VM-Execution Controls
- Primary Processor-Based VM-Execution Controls
- Secondary Processor-Based VM-Execution Controls
- VM entry controls
- VM exit controls
- EPTP

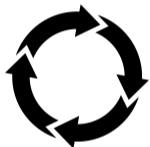
Running VMs



- VMLAUNCH
 - VMCS not launched yet
- VMRESUME
 - Continue launched VMCS
- VMCLEAR resets the launch state!

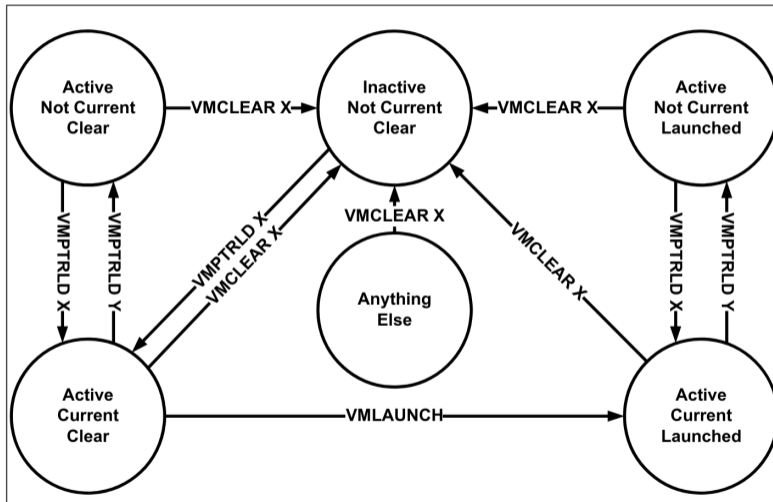
¹Intel SDM Volume 3, 25.1





- Current
 - only one VMCS can be current at a time
 - VMWRITE, VMREAD, VMLAUNCH, VMRESUME operate on the current VMCS
- Active
 - VMCS is managed by the CPU
 - multiple VMCS can be active at a time
- Launched
 - VMCS is launched and can only be continued with VMRESUME

¹Intel SDM Volume 3, 25.1





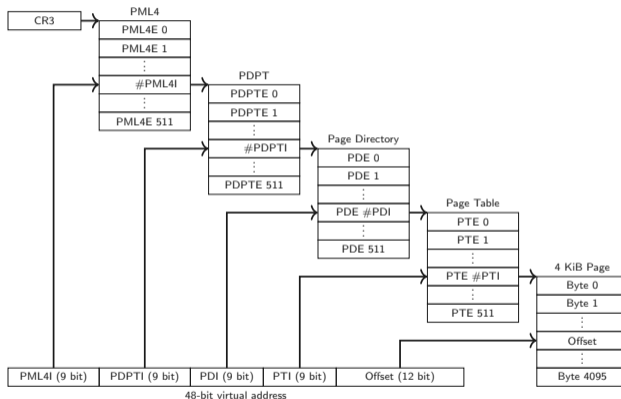
- VMCLEAR
 - initializes a new VMCS
 - sets active VMCS to inactive, not current, and not launched
- VMPTRLD
 - sets VMCS to active and current
 - sets previous current VMCS to not current
- VMLAUNCH
 - sets VMCS to launched
- VMRESMUE
 - can only be used on launched VMCS

¹Intel SDM Volume 3, 25.1

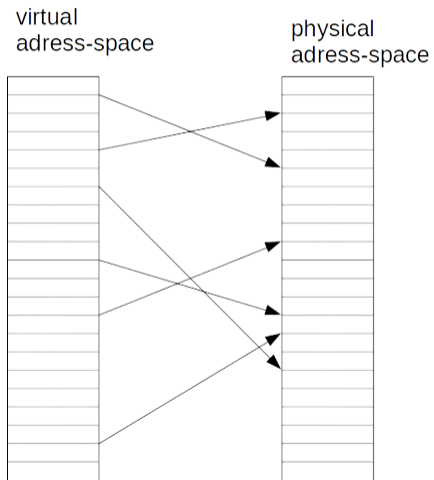
EPT

How can we give a guest access to a physical address space without giving it access to the hosts physical address space?

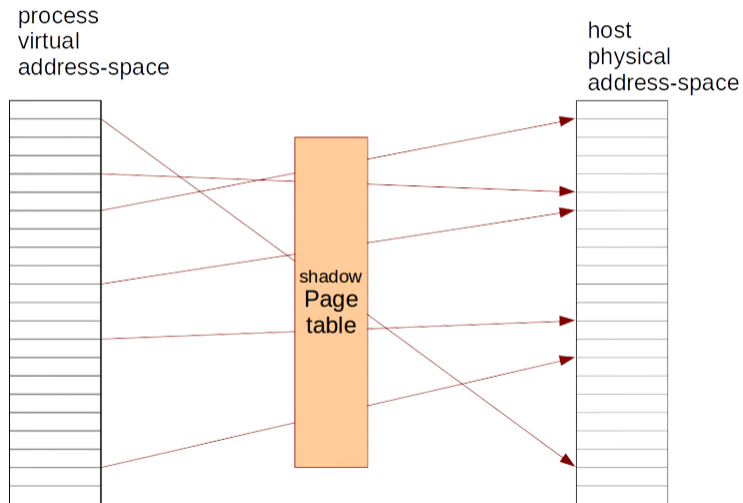
Virtual Address Translation



Paging



Shadow Page Table





- guest runs on a normal page table
 - guest can not modify their real page table
 - instead the guest has a copy of the real page table
 - when the guest changes page table copy
 - Hypervisor has to catch access
 - update shadow (real) page table
- guest can manage virtual memory without access to arbitrary physical memory or real PPNs



Advantages:

- + address translations are fast
- + requires only minimal hardware support

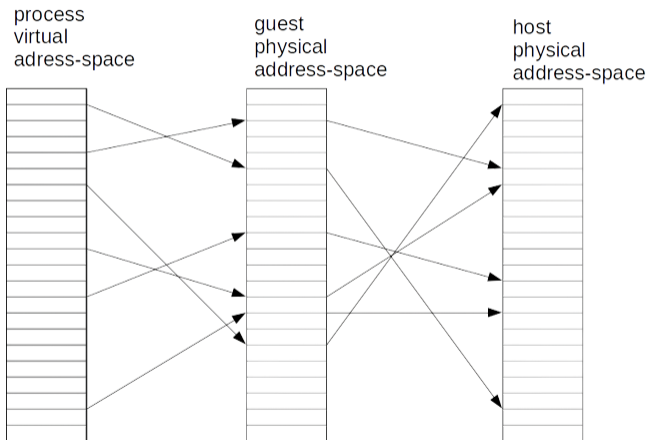
Disadvantages:

- page table modifications are slow
- the hypervisor needs a trampolin in the guest address spaces
- only works with paging enabled
- x2 memory overhead for page tables in the guest
- complicated to implement

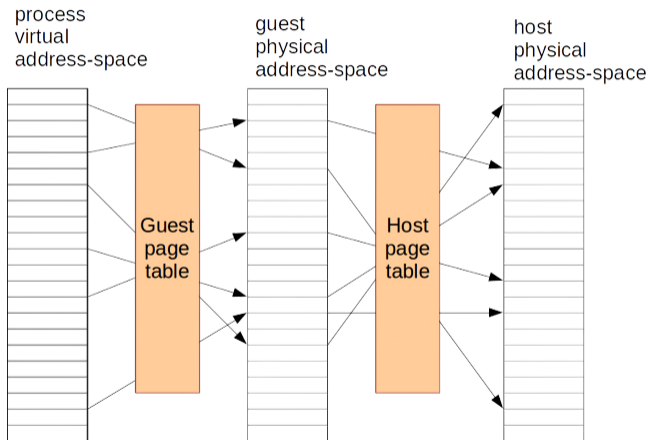


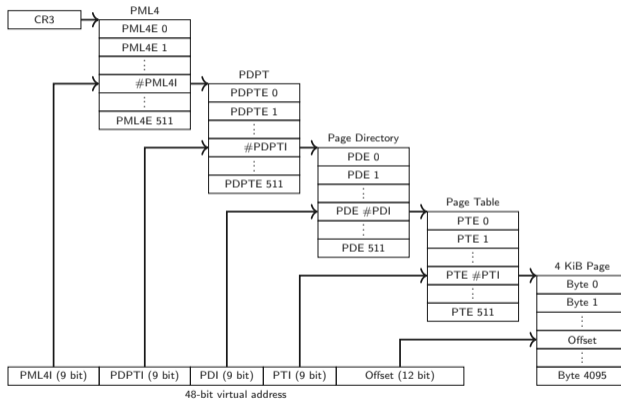
WE NEED TO GO DEEPER

Combined Paging

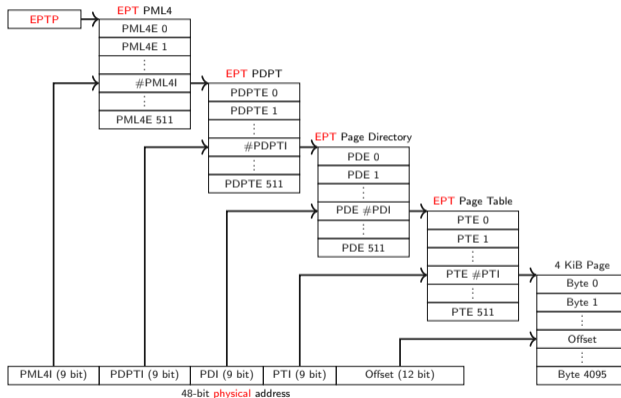


Page Tables

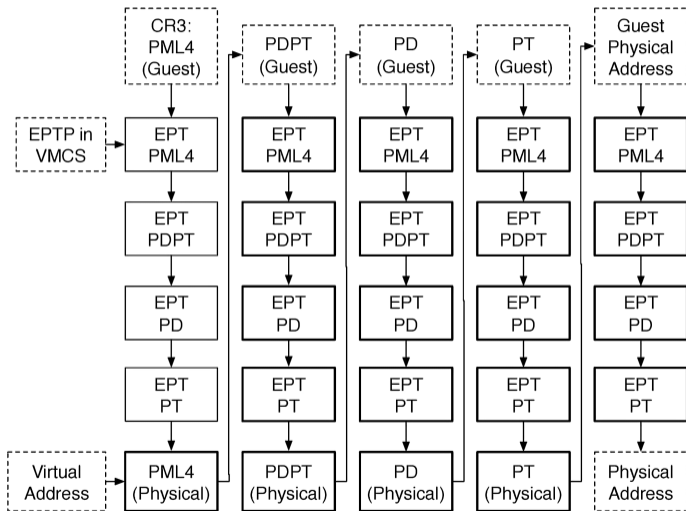




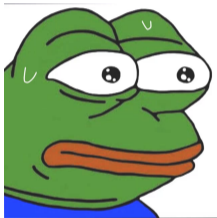
Physical Address Translation



Isn't this slow?



Caching saves us yet again



- EPT translations are cached in the TLB
- Translations are tagged with the EPTP
- We can invalidate all TLB entries for a given EPTP or all EPT TLB entries using the INVEPT instruction

EPT Bits

6	6	6	5	5	5	5	5	5	5	5	M ¹	M-1	3	3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	0	9	8	7	6	5	4	3	2	1	0		
Reserved												Address of EPT PML4 table												Rsvd.	S	A	EPT	EPT	EPT ³																				
Ignored												Rsvd.	Address of EPT page-directory-pointer table												Ig	X	Ig	Reserved	X	W	R	PML4E: present ⁶																	
S	V	E	Ignored																								Q	Q	Q	PML4E: not present																			
S	V	E	Ig	S	S	P	V	Ignored				Rsvd.	Physical address of 1GB page				Reserved				Ig	X	D	A	1	P	EPT	X	W	R	PDPTE: 1GB page																		
Ignored												Rsvd.	Address of EPT page directory												Ig	X	Ig	Reserved	X	W	R	PDPTE: page directory																	
S	V	E	Ignored																								Q	Q	Q	PDPTE: not present																			
S	V	E	Ig	S	S	P	V	Ignored				Rsvd.	Physical address of 2MB page				Reserved				Ig	X	D	A	1	P	EPT	X	W	R	PDE: 2MB page																		
Ignored												Rsvd.	Address of EPT page table												Ig	X	Ig	Reserved	X	W	R	PDE: page table																	
S	V	E	Ignored																								Q	Q	Q	PDE: not present																			
S	V	E	Ig	S	S	P	V	Ignored				Rsvd.	Physical address of 4KB page				Ig	X	D	A	Ig	P	EPT	X	W	R	PTE: 4KB page																						
S	V	E	Ignored																								Q	Q	Q	PTE: not present																			

Questions?

