Cloud Operating Systems

Virtual Memory and Structural Setup

Fabian Rauscher, Jonas Juffinger, Daniel Gruss

2024-03-07

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: />
```

Setup

- Upstream: https://extgit.iaik.tugraz.at/cloudos/upstream
- 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 handeling)
- uSTL (STL, provides vector, string, map,...)
- Loader (user space binary loader)

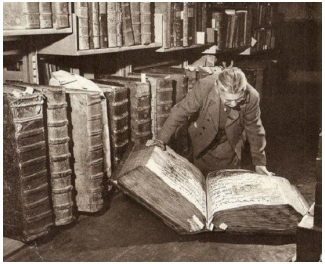
→ For more info:
https://www.iaik.tugraz.at/teaching/materials/os/tutorials/

VMX

What is VMX?

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

Intel SDM



- 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 recources
- Manages the ability of the guest to access
  - Processor Recources
  - Physical Memory
  - Interrupts
  - I/O
  - ...

Getting Started

Enabling VMX

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

[1] Intel 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 CR0[1]
  - Set bits specified by IA32_VMX_CR0_FIXED0 (0x486)
  - Clear bits specified by IA32_VMX_CR0_FIXED1 (0x487)
- Setup CR4[1]
  - 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 set[2]

[1] Intel SDM Volume 3, 24.8
[2] Intel SDM Volume 3, 24.7

Entering VMX Operation (Part 2)

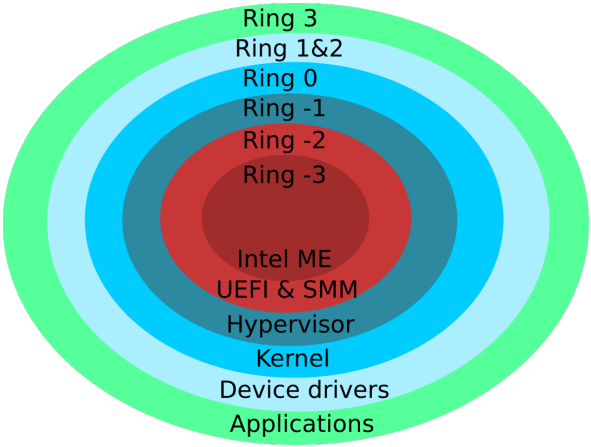
VMXON region [2]

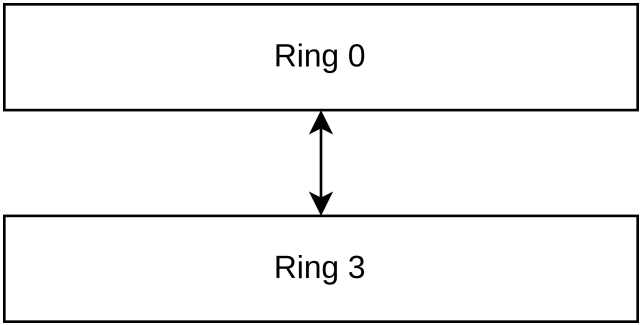
- Used by the processor to support VMX operation
- Up to 4KB in size
- VMXON pointer needs to be a 4KB aligned valid physical address
- Bits 30:0 must contain the VMCS revision identifier (IA32_VMX_BASIC, 0x480)
- Can be loaded using the VMXON instruction to enter VMX operation

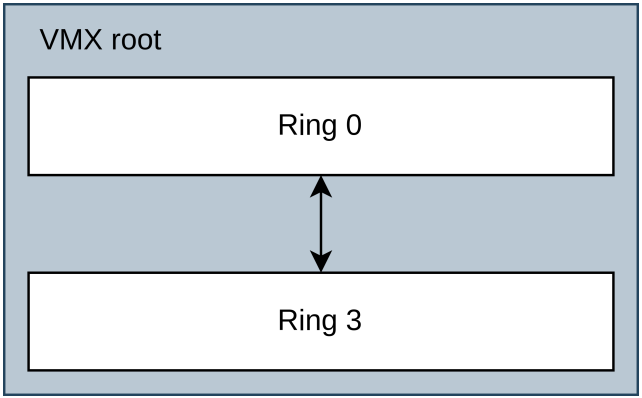
[1] Intel SDM Volume 3, 31

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

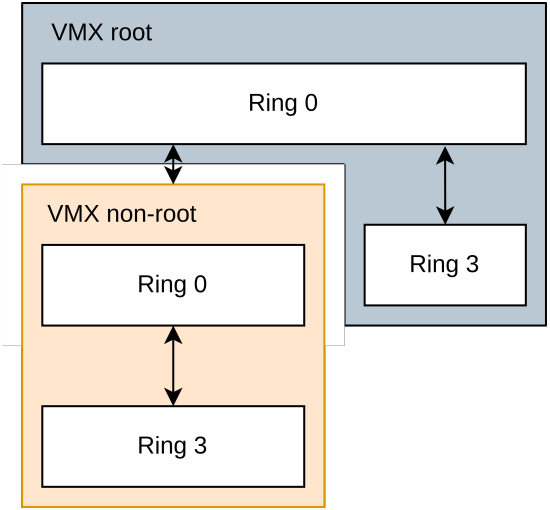
Rings


Ring 3
Ring 1&2
Ring 0
Ring -1
Ring -2
Ring -3
Intel ME
UEFI & SMM
Hypervisor
Kernel
Device drivers
Applications

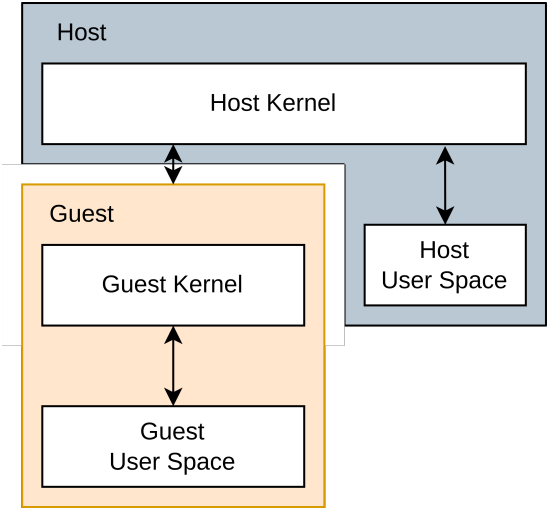

Ring 0
Ring 3

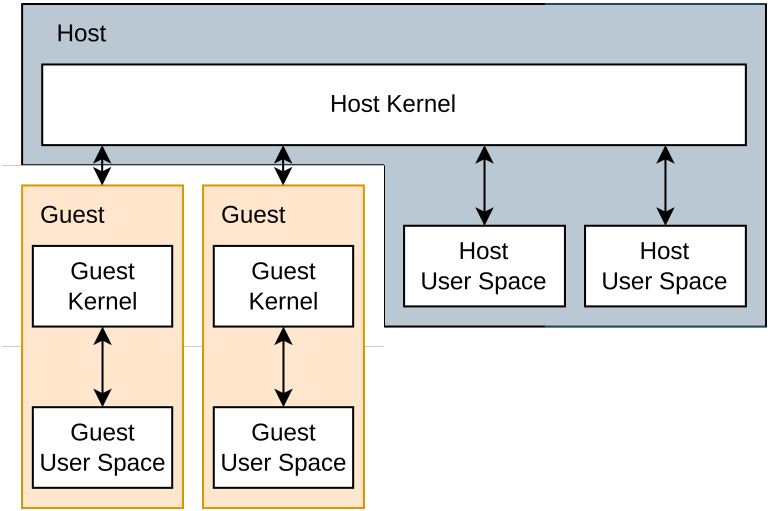

VMX root
Ring 0
Ring 3

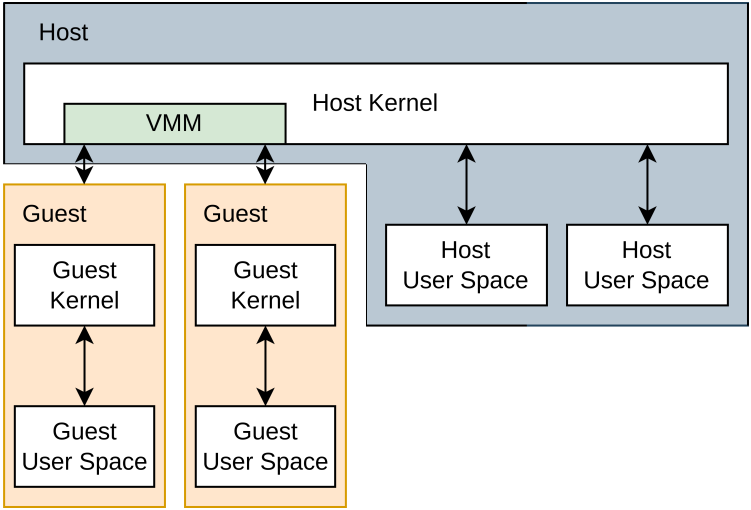
VMM Overview



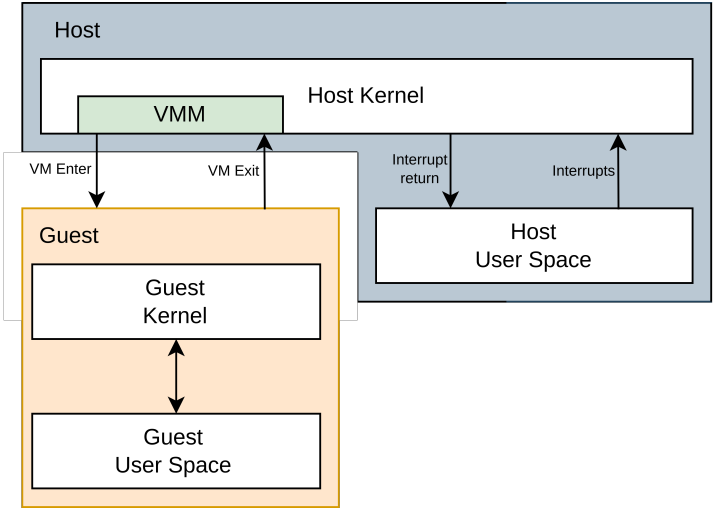
VMM Overview




Host
Host Kernel
Guest
Guest Kernel
Guest User Space
Guest
Guest Kernel
Guest User Space
Host User Space
Host User Space

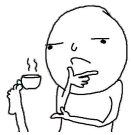

Host
Host Kernel
VMM
Guest
Guest Kernel
Guest User Space
Guest
Guest Kernel
Guest User Space
Host User Space
Host User Space

VMM Overview



VMCS

What is a 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

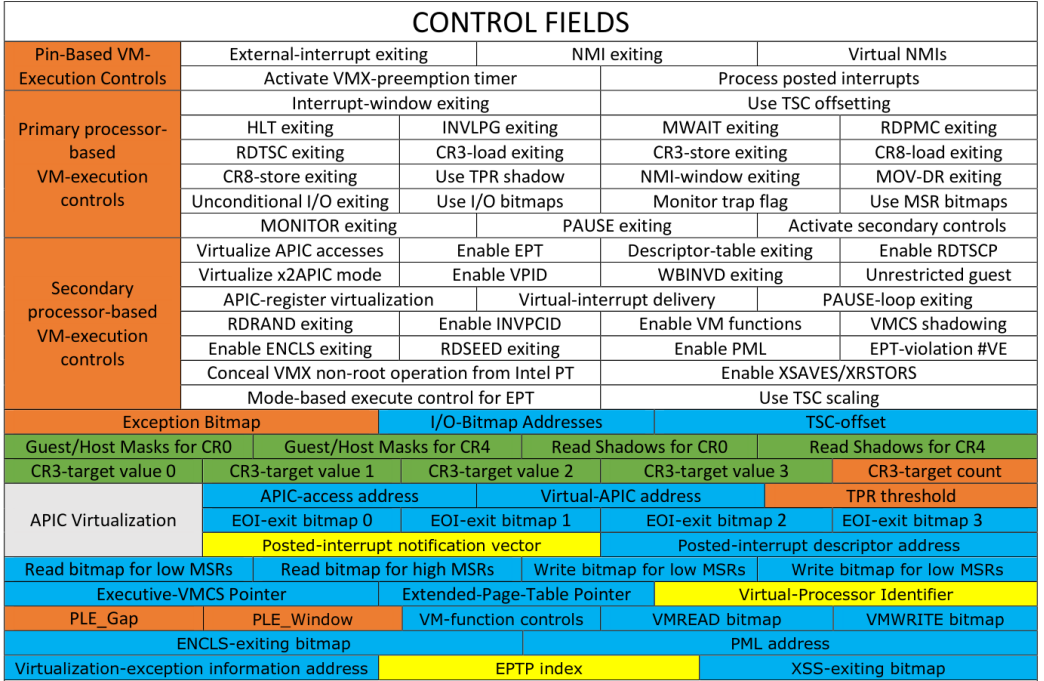
[2] Intel SDM Volume 3, 25.2

- 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]

[1] Intel SDM Volume 3, 25.2

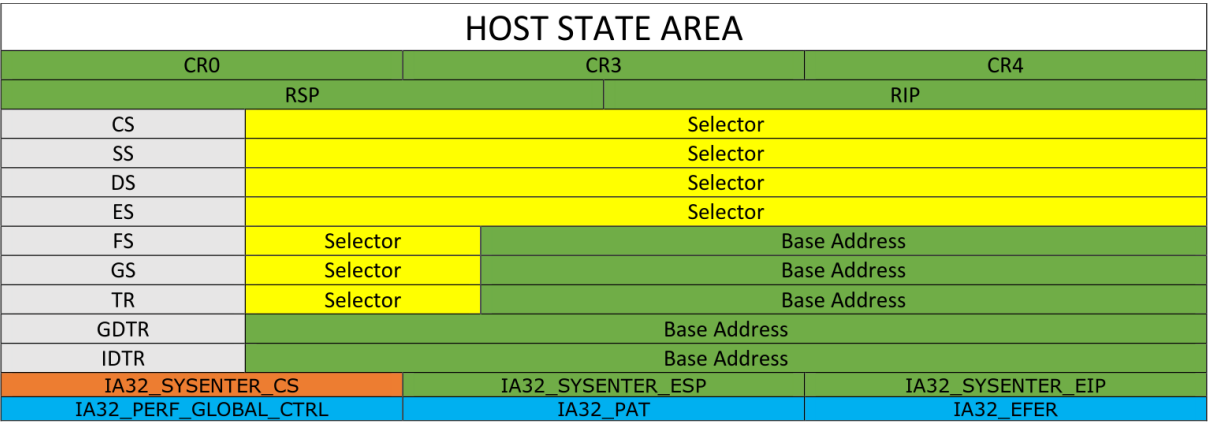
- Guest configuration
- Some registers are not easy to restore
  - CR4/CR0
  - Segment bases and access rights
  - CR3
  - ...

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

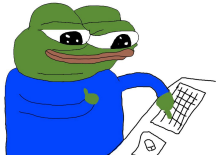

CONTROL FIELDS
Pin-Based VM-Execution Controls
External-interrupt exiting
NMI exiting
Virtual NMIs
Activate VMX-preemption timer
Process posted interrupts
Primary processor-based VM-execution controls
Interrupt-window exiting
Use TSC offsetting
HLT exiting
INVLPG exiting
MWAIT exiting
RDPMC exiting
RDTSC 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	[redacted]	Base Address	Limit	[redacted]
IDTR	[redacted]	Base Address	Limit	[redacted]
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
CR0
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

Guest State



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

$\rightarrow$ 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)

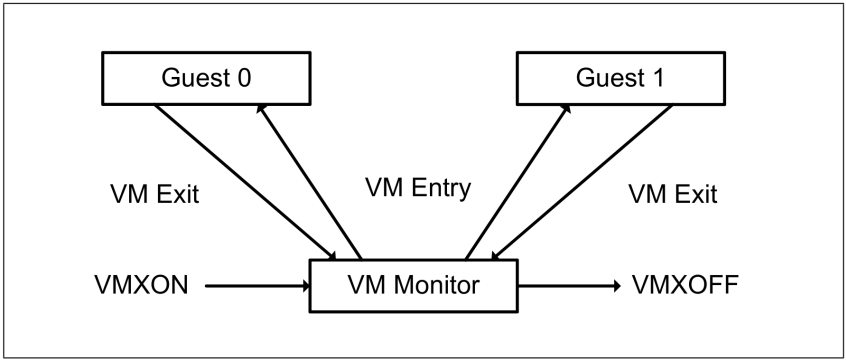
VMCS Some Important Control Fields

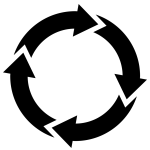
- Pin-Based VM-Execution Controls
- Primary Processor-Based VM-Execution Controls
- Secondary Processor-Based VM-Execution Controls
- VM entry controls
- VM exit controls
- EPTP

Running VMs

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

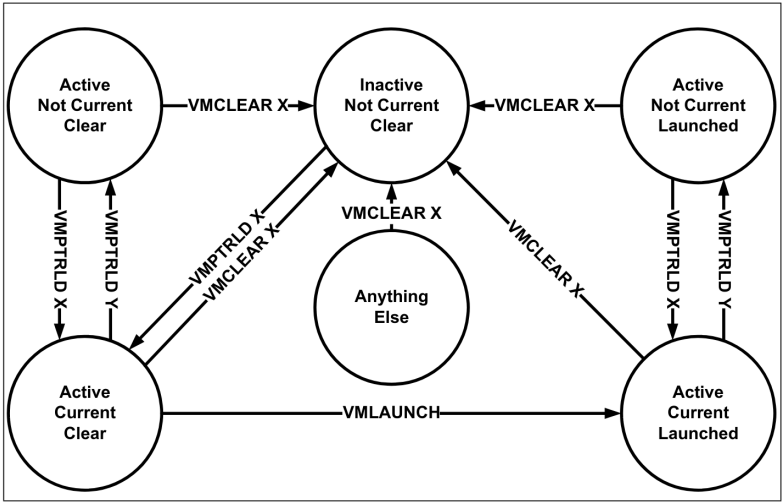
[1] Intel SDM Volume 3, 25.1


Guest 0
Guest 1
VM Exit
VM Entry
VM Exit
VMXON
VM Monitor
VMXOFF



- Current
  - only one VMCS can is 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

[1] Intel SDM Volume 3, 25.1


Active
Not Current
Clear
VMCLEAR X
Inactive
Not Current
Clear
VMCLEAR X
Active
Not Current
Launched
VMPTRLD X
VMPTRLD Y
VMPTRLD X
VMCLEAR X
VMCLEAR X
Anything
Else
VMCLEAR X
VMPTRLD X
VMPTRLD Y
Active
Current
Clear
VMLAUNCH
Active
Current
Launched

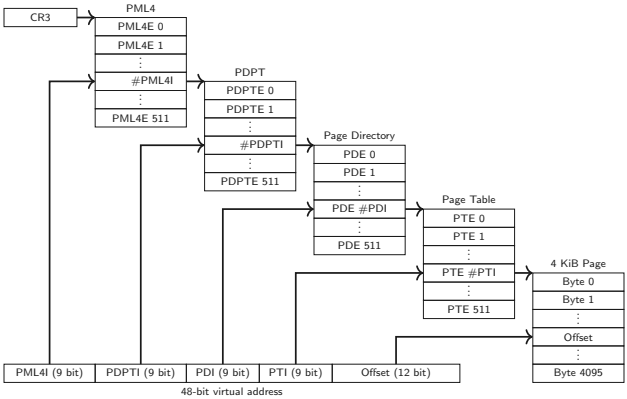
- VMCLEAR
  - initializes a new VMCS
  - sets active VMCS to inactive, not current, and not launched
- VMPTRLD
  - sets VMCS to active and current
  - sets previous current VMCS to not current
- VMLAUNCH
  - sets VMCS to launched
- VMRESUME
  - can only be used on launched VMCS

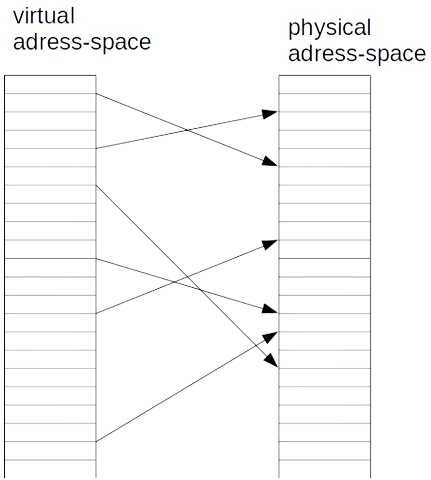
[1] Intel SDM Volume 3, 25.1

EPT

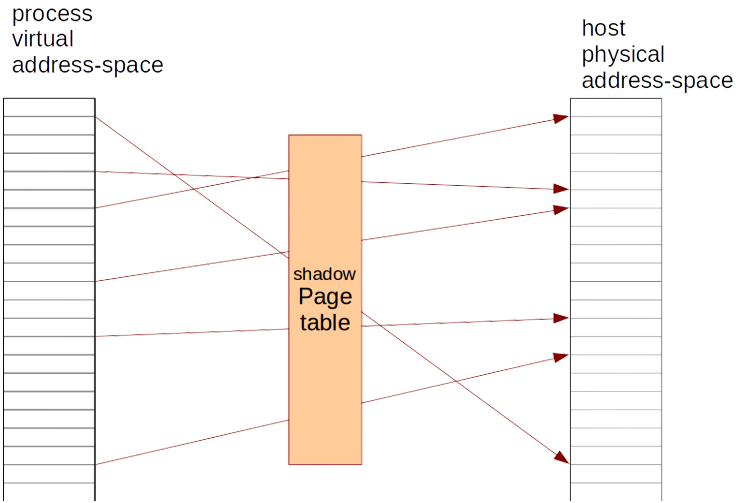
How can we give a guest access to a physical address space without giving it access to the hosts physical address space?

Virtual Address Translation




virtual
adress-space
physical
adress-space

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

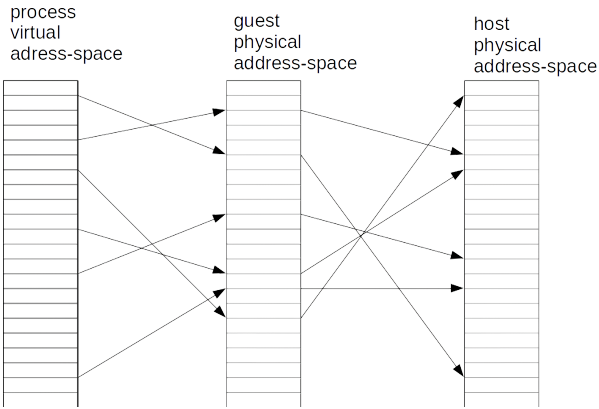
Advantages:

+ address translations are fast
+ requires only minimal hardware support

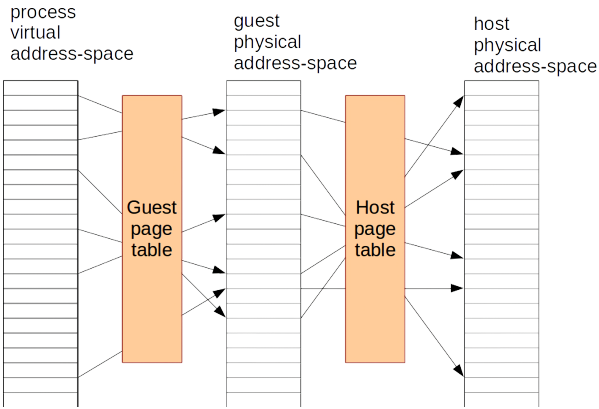
Disadvantages:

- page table modifications are slow
- only works with paging enabled
- x2 memory overhead for page tables in the guest
- complicated to implement

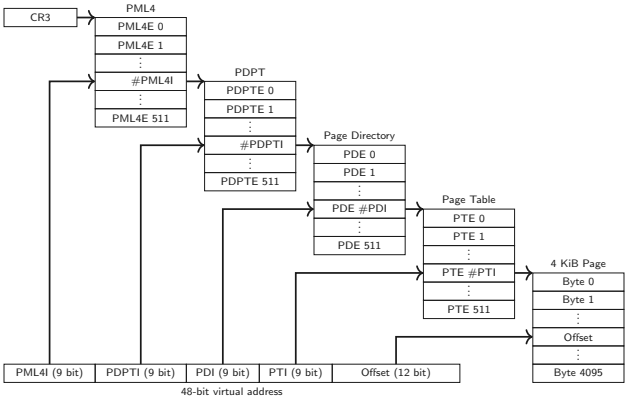

WE NEED TO GO DEEPER


process
virtual
adress-space
guest
physical
address-space
host
physical
address-space

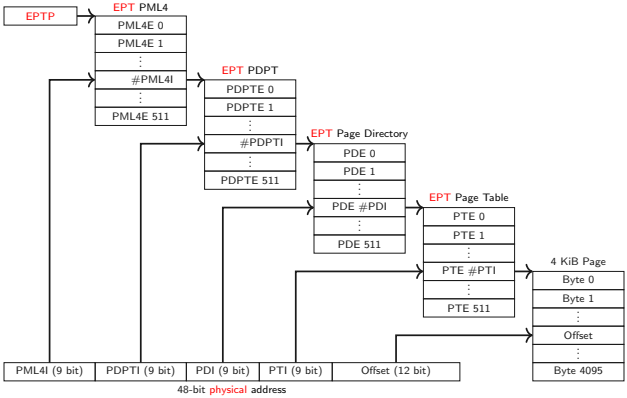
Page Tables


process
virtual
address-space
guest
physical
address-space
host
physical
address-space
Guest
page
table
Host
page
table

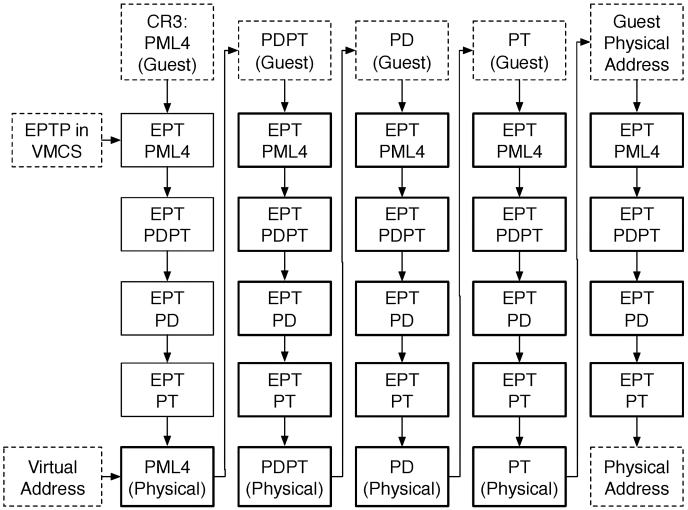
Virtual Address Translation



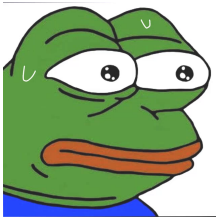
Physical Address Translation



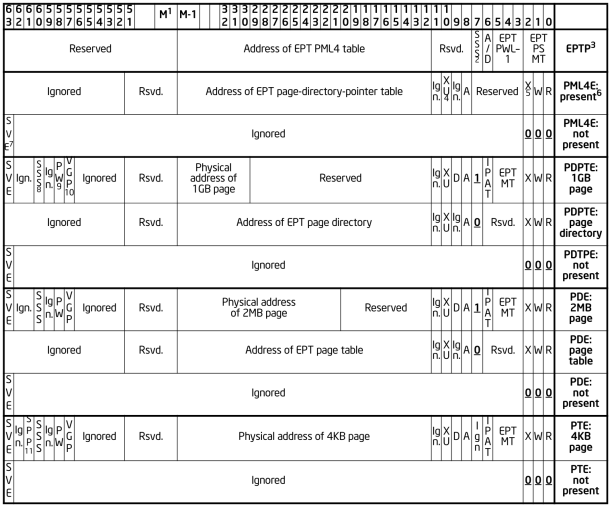
Isn't this slow?



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


63 62 61 60 59 58 57 56 55 54 53 52 51 M[1] M-1 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Reserved
Address of EPT PML4 table
Rsvd.
SSS2
A/D
EPT PWL−1
EPT PS MT
EPTP[3]
Ignored
Rsvd.
Address of EPT page-directory-pointer table
Ign.
XU[4]
Ign.
A
Reserved
XS
W
R
PML4E: present[6]
SVE[5]
Ignored
0 0 0
PML4E: not present
SVE
Ign.
SSS[8]
Ign.
PW[9]
VGP[10]
Ignored
Rsvd.
Physical address of 1GB page
Reserved
Ign.
XU
D
A
1
IPAT
EPT MT
X
W
R
PDPTE: 1GB page
Ignored
Rsvd.
Address of EPT page directory
Ign.
XU
Ign.
A
0
Rsvd.
X
W
R
PDPTE: page directory
SVE
Ignored
0 0 0
PDTPE: not present
SVE
Ign.
SSS
Ign.
PW
VGP
Ignored
Rsvd.
Physical address of 2MB page
Reserved
Ign.
XU
D
A
1
PAT
EPT MT
X
W
R
PDE: 2MB page
Ignored
Rsvd.
Address of EPT page table
Ign.
XU
Ign.
A
0
Rsvd.
X
W
R
PDE: page table
SVE
Ignored
0 0 0
PDE: not present
SVE
Ign.
SPP[11]
Ign.
SSS
Ign.
PW
VGP
Ignored
Rsvd.
Physical address of 4KB page
Ign.
XU
D
A
Ign
IPAT
EPT MT
X
W
R
PTE: 4KB page
SVE
Ignored
0 0 0
PTE: not present