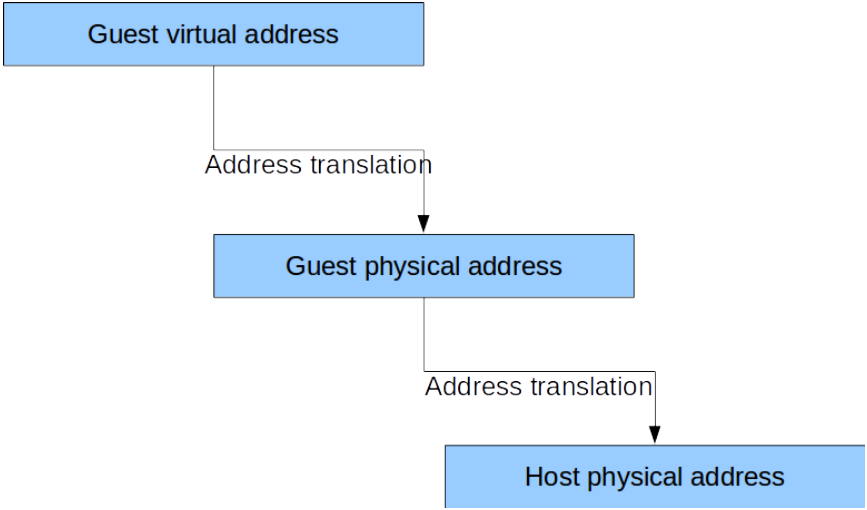
Cloud Operating Systems

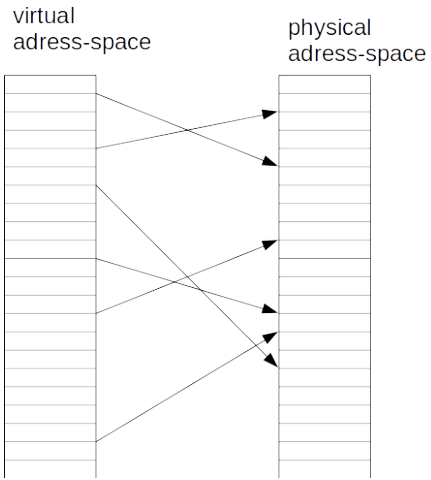
Virtual Memory and Structural Setup

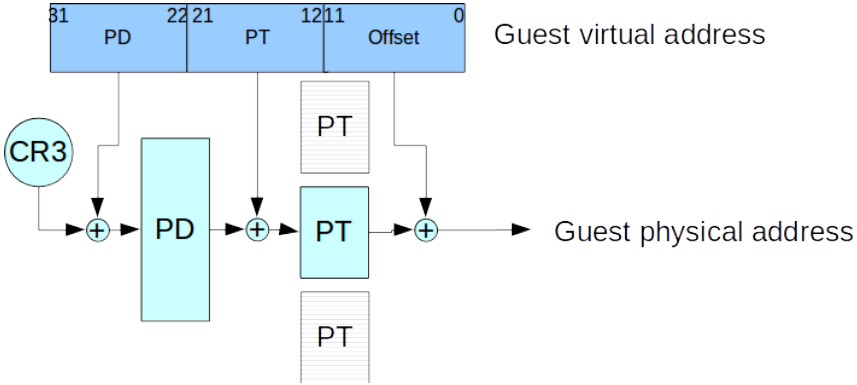
Fabian Rauscher, Daniel Gruss, Andreas Kogler

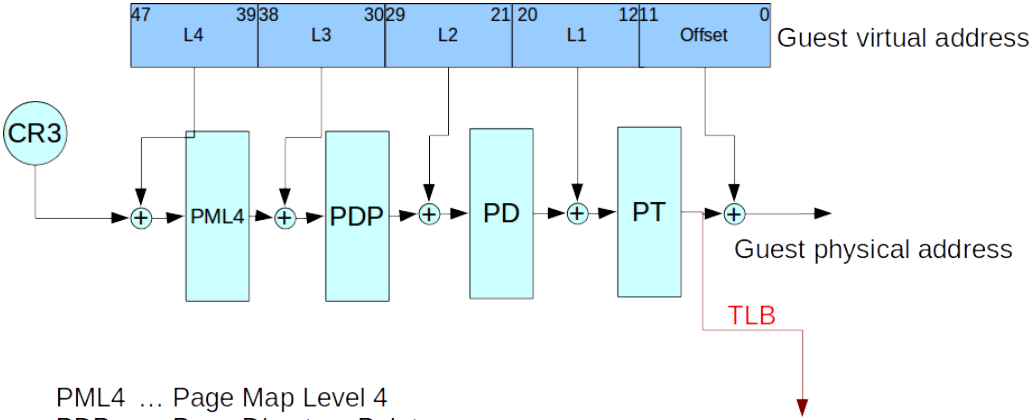
2022-03-14


Guest virtual address
Address translation
Guest physical address
Address translation
Host physical address

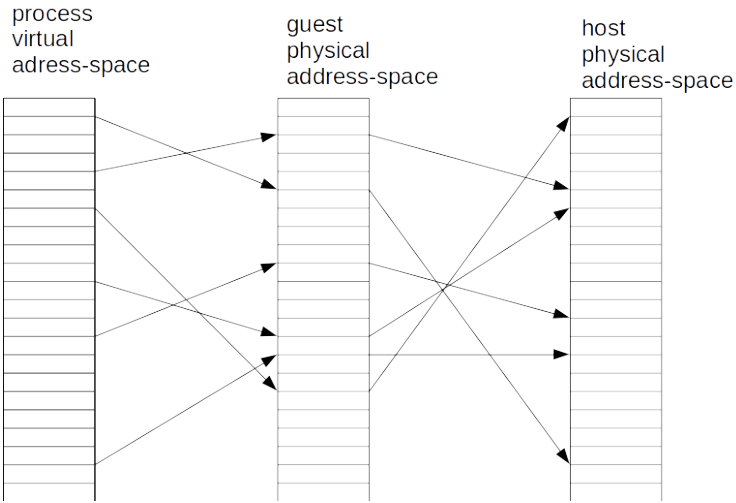
Paging



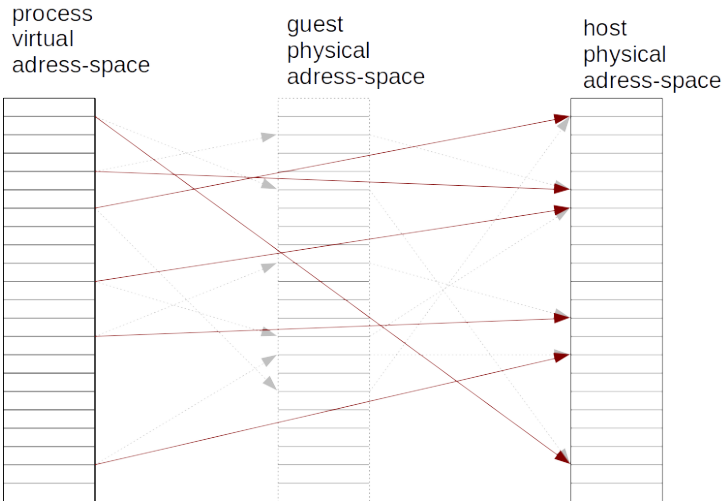

31
22
21
12
11
0
PD
PT
Offset
Guest virtual address
CR3
PD
PT
PT
PT
Guest physical address



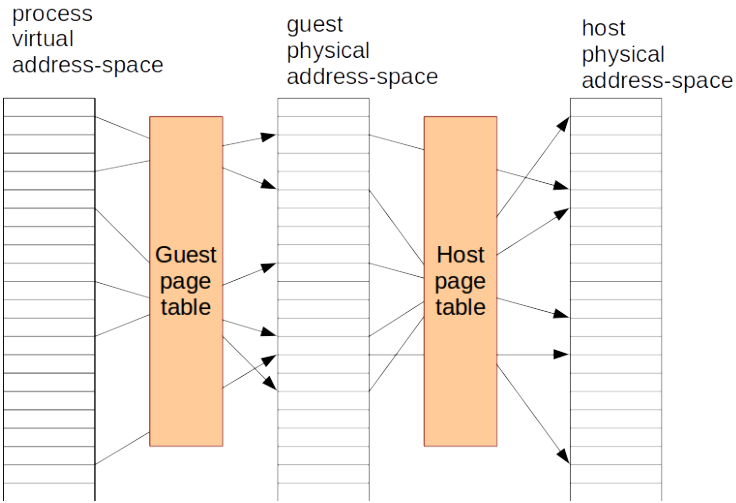
PML4 ... Page Map Level 4
PDP ... Page Directory Pointer
PD ... Page Directory
PT ... Page Table


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

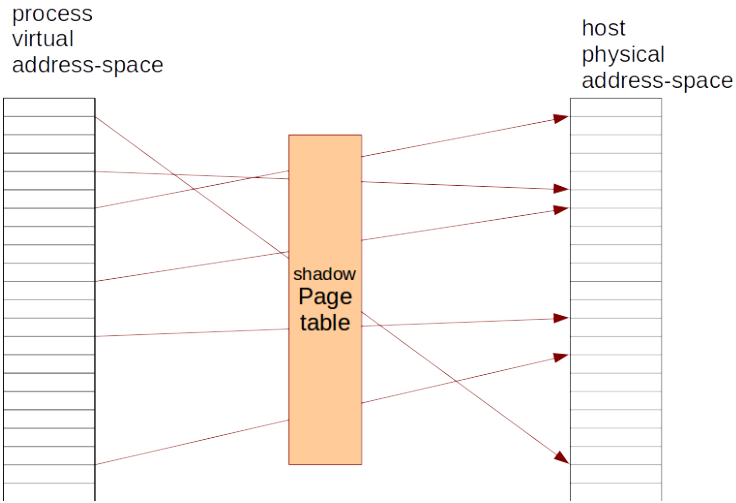
Shadow Page Table



Page Tables


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

Shadow Page Table



- **merges** both page tables into one that the HW uses

Shadow Page Table

- **merges** both page tables into one that the HW uses
- when guest changes own page table

Shadow Page Table

- **merges** both page tables into one that the HW uses
- when guest changes own page table
  - Hypervisor has to catch access

Shadow Page Table

- **merges** both page tables into one that the HW uses
- when guest changes own page table
  - Hypervisor has to catch access
  - update shadow page table

Shadow Page Table

- when HW changes shadow page table

- when HW changes shadow page table
- update guest PT

Shadow Page Table

- when HW changes shadow page table
- update guest PT
  - expensive!

Shadow Page Table

- when HW changes shadow page table
- update guest PT
  - expensive!
  - page faults caught by hypervisor

Shadow Page Table

- when HW changes shadow page table
- update guest PT
  - expensive!
  - page faults caught by hypervisor
  - must run through guest PTs

Shadow Page Table

- when HW changes shadow page table
- update guest PT
  - expensive!
  - page faults caught by hypervisor
  - must run through guest PTs
  - must emulate accessed and modified bits for guest

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

- Upstream: https://github.com/IAIK/sweb
- SWEB Tutorials: https://www.iaik.tugraz.at/teaching/materials/os/tutorials/

- 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

- 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

Setup

Building and running SWEB

Building and running SWEB

- mkdir -p /tmp/sweb

Building and running SWEB

- mkdir -p /tmp/sweb
- cd /tmp/sweb

Building and running SWEB

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

Building and running SWEB

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

Building and running SWEB

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

VMX

What is VMX?


intel
inside


intel
inside

What is VMX?

- Intels Virtual-Machine Extension

What is VMX?

- Intels Virtual-Machine Extension
- Provides hardware support for virtualization

What is VMX?

- Intels Virtual-Machine Extension
- Provides hardware support for virtualization
- Two different classes of software:

What is VMX?

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

What is VMX?

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

- Gives the guest the illusion of running on real hardware

- Gives the guest the illusion of running on real hardware
- Has full control of the processors recources

- 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

- 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

- 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

- 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

- 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

- 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


works on
my
machine


works on
my
machine

- Check for VMX support: CPUID.1:ECX.VMX[bit 5] = 1

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

- VMX root operation

- VMX root operation
  - new instructions (VMX instructions)

VMX Operation

- VMX root operation
  - new instructions (VMX instructions)
  - restrictions on certain control register values

VMX Operation

- VMX root operation
  - new instructions (VMX instructions)
  - restrictions on certain control register values
  - used for the VMM

VMX Operation

- VMX root operation
  - new instructions (VMX instructions)
  - restrictions on certain control register values
  - used for the VMM
- VMX non-root operation

- 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

VMX Operation

- 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

VMX Operation

- 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

- 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

- 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

- 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)

Entering VMX Operation (Part 1)

Perpetrations

Perpetrations

- Setup CR0

Entering VMX Operation (Part 1)

Perpetrations

- Setup CR0
  - Set bits specified by IA32_VMX_CR0_FIXED0 (0x486)

Entering VMX Operation (Part 1)

Perpetrations

- Setup CR0
  - Set bits specified by IA32_VMX_CR0_FIXED0 (0x486)
  - Clear bits specified by IA32_VMX_CR0_FIXED1 (0x487)

Entering VMX Operation (Part 1)

Perpetrations

- Setup CR0
  - Set bits specified by IA32_VMX_CR0_FIXED0 (0x486)
  - Clear bits specified by IA32_VMX_CR0_FIXED1 (0x487)
- Setup CR4

Entering VMX Operation (Part 1)

Perpetrations

- Setup CR0
  - Set bits specified by IA32_VMX_CR0_FIXED0 (0x486)
  - Clear bits specified by IA32_VMX_CR0_FIXED1 (0x487)
- Setup CR4
  - Set bits specified by IA32_VMX_CR4_FIXED0 (0x488)

Entering VMX Operation (Part 1)

Perpetrations

- Setup CR0
  - Set bits specified by IA32_VMX_CR0_FIXED0 (0x486)
  - Clear bits specified by IA32_VMX_CR0_FIXED1 (0x487)
- Setup CR4
  - Set bits specified by IA32_VMX_CR4_FIXED0 (0x488)
  - Clear bits specified by IA32_VMX_CR4_FIXED1 (0x489)

Entering VMX Operation (Part 1)

Perpetrations

- Setup CR0
  - Set bits specified by IA32_VMX_CR0_FIXED0 (0x486)
  - Clear bits specified by IA32_VMX_CR0_FIXED1 (0x487)
- Setup CR4
  - 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

Entering VMX Operation (Part 2)

VMXON region

- Used by the processor to support VMX operation

Entering VMX Operation (Part 2)

VMXON region

- Used by the processor to support VMX operation
- Up to 4KB in size

Entering VMX Operation (Part 2)

VMXON region

- Used by the processor to support VMX operation
- Up to 4KB in size
- VMXON pointer needs to be a 4KB aligned valid physical address

Entering VMX Operation (Part 2)

VMXON region

- 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)

Entering VMX Operation (Part 2)

VMXON region

- 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

- VMXON

- VMXON
- VMXOFF

- VMXON
- VMXOFF
- INVEPT

- VMXON
- VMXOFF
- INVEPT
- INVVPID

- VMXON
- VMXOFF
- INVEPT
- INVVPID
- VMCALL

New Instructions

- VMXON
- VMXOFF
- INVEPT
- INVVPID
- VMCALL
- VMCLEAR

- VMXON
- VMXOFF
- INVEPT
- INVVPID
- VMCALL
- VMCLEAR
- VMLAUNCH/VMRESUME

- VMXON
- VMXOFF
- INVEPT
- INVVPID
- VMCALL
- VMCLEAR
- VMLAUNCH/VMRESUME
- VMPTRLD

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

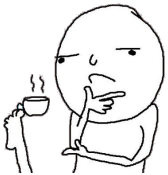
- VMXON
- VMXOFF
- INVEPT
- INVVPID
- VMCALL
- VMCLEAR
- VMLAUNCH/VMRESUME
- VMPTRLD
- VMPTRSTR
- VMREAD

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

VMCS

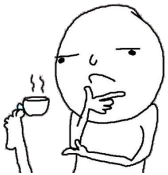
What is a VMCS?

What is a VMCS?

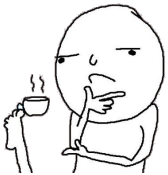


What is a VMCS?

Virtual-Machine Control Data Structure (VMCS)



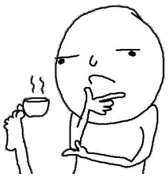
What is a VMCS?



Virtual-Machine Control Data Structure (VMCS)

- Manages VM entries and VM exits

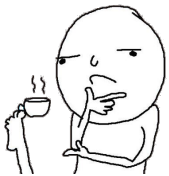
What is a VMCS?



Virtual-Machine Control Data Structure (VMCS)

- Manages VM entries and VM exits
- Used to setup processor behavior in VMX non-root operation

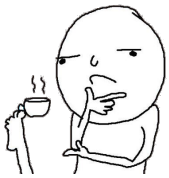
What is a VMCS?



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

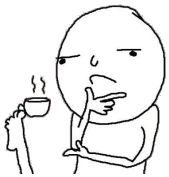
What is a VMCS?



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

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

VMCS Region

- Up to 4KB in size

- Up to 4KB in size
- VMCS pointer needs to be a 4KB aligned valid physical address

VMCS Region

- 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)

Why?

Why?

Why?

Why?

- Guest configuration

Why?

- Guest configuration
- Some registers can not easily be restored!

Why?

- Guest configuration
- Some registers can not easily be restored!
  - CR4/CR0

Why?

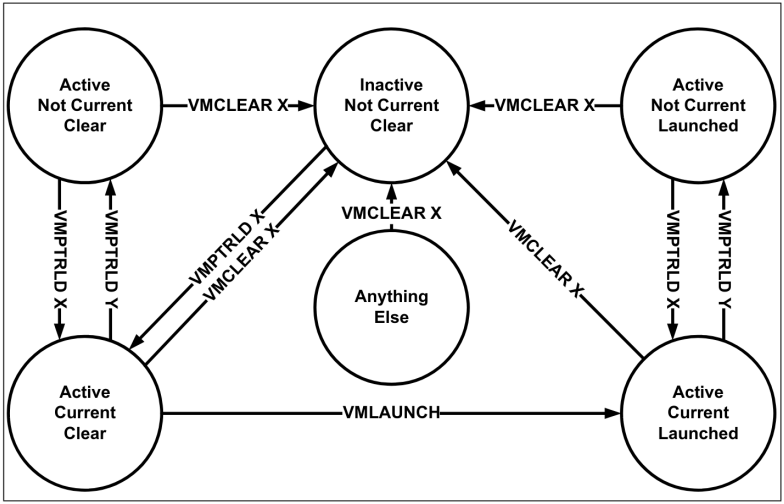
- Guest configuration
- Some registers can not easily be restored!
  - CR4/CR0
  - Segment bases and access rights

Why?

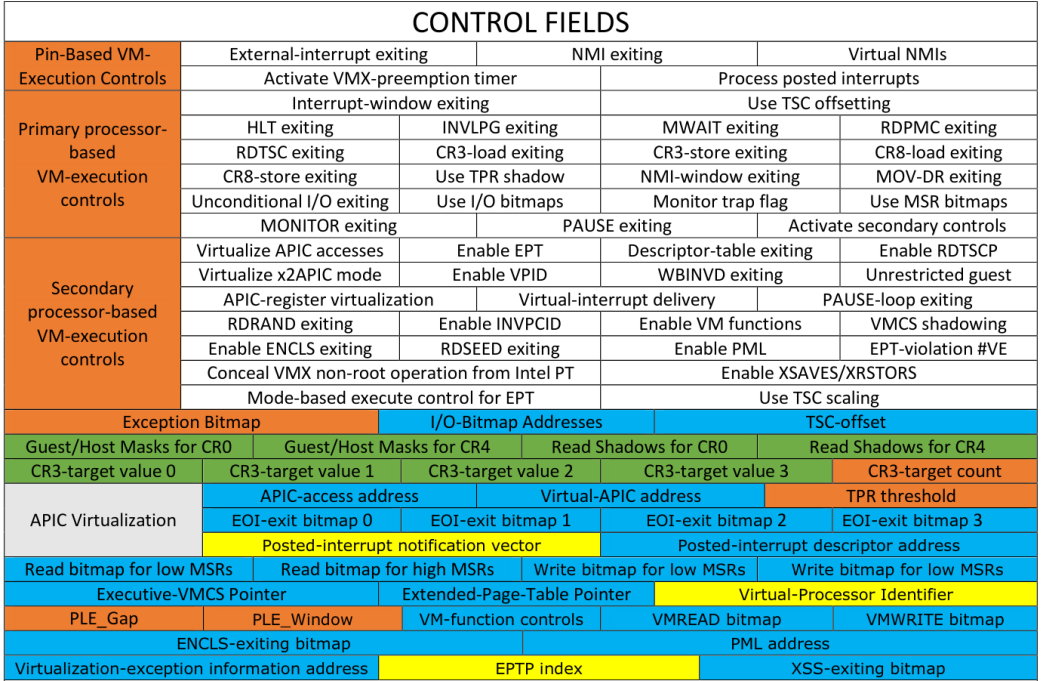
- Guest configuration
- Some registers can not easily be restored!
  - CR4/CR0
  - Segment bases and access rights
  - CR3

- Guest configuration
- Some registers can not easily be restored!
  - CR4/CR0
  - Segment bases and access rights
  - CR3
  - ...

VMCS States

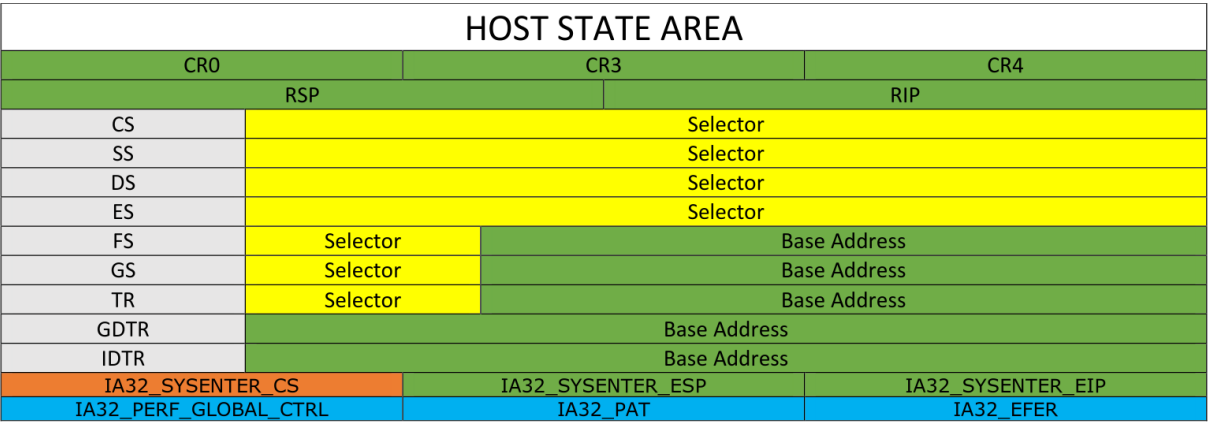

Active
Not Current
Clear
VMCLEAR X
Inactive
Not Current
Clear
VMCLEAR X
Active
Not Current
Launched
VMPTRLD X
VMPTRLD Y
VMPTRLD X
VMCLEAR X
VMCLEAR X
Anything
Else
VMCLEAR X
VMPTRLD X
VMPTRLD Y
Active
Current
Clear
VMLAUNCH
Active
Current
Launched

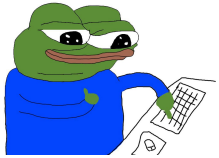
- Natural-Width fields.
- 16-bits fields.
- 32-bits fields.
- 64-bits fields.

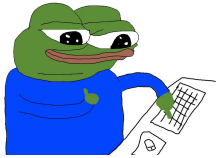

CONTROL FIELDS
Pin-Based VM-Execution Controls
External-interrupt exiting
NMI exiting
Virtual NMIs
Activate VMX-preemption timer
Process posted interrupts
Primary processor-based VM-execution controls
Interrupt-window exiting
Use TSC offsetting
HLT exiting
INVLPG exiting
MWAIT exiting
RDPMC exiting
RDTSC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

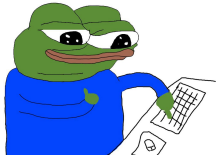
GUEST STATE AREA

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	Selector	Base Address	Segment Limit	Access Right
IDTR	Selector	Base Address	Segment 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				

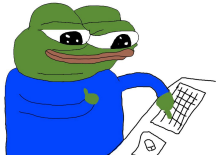

HOST STATE AREA
CR0
CR3
CR4
RSP
RIP
CS
Selector
SS
Selector
DS
Selector
ES
Selector
FS
Selector
Base Address
GS
Selector
Base Address
TR
Selector
Base Address
GDTR
Base Address
IDTR
Base Address
IA32_SYSENTER_CS
IA32_SYSENTER_ESP
IA32_SYSENTER_EIP
IA32_PERF_GLOBAL_CTRL
IA32_PAT
IA32_EFER







- What about things that are not in the guest state area, like general-purpose registers?



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

VMCS Some Important Fields

- Pin-Based VM-Execution Controls

VMCS Some Important Fields

- Pin-Based VM-Execution Controls
- Primary Processor-Based VM-Execution Controls

VMCS Some Important Fields

- Pin-Based VM-Execution Controls
- Primary Processor-Based VM-Execution Controls
- Secondary Processor-Based VM-Execution Controls

VMCS Some Important Fields

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

VMCS Some Important Fields

- Pin-Based VM-Execution Controls
- Primary Processor-Based VM-Execution Controls
- Secondary Processor-Based VM-Execution Controls
- VM entry/exit controls
- GDTR/IDTR Base and Limit

VMCS Some Important Fields

- Pin-Based VM-Execution Controls
- Primary Processor-Based VM-Execution Controls
- Secondary Processor-Based VM-Execution Controls
- VM entry/exit controls
- GDTR/IDTR Base and Limit
- Segment Bases and Access Rights

VMCS Some Important Fields

- Pin-Based VM-Execution Controls
- Primary Processor-Based VM-Execution Controls
- Secondary Processor-Based VM-Execution Controls
- VM entry/exit controls
- GDTR/IDTR Base and Limit
- Segment Bases and Access Rights
- RFLAGS

VMCS Some Important Fields

- Pin-Based VM-Execution Controls
- Primary Processor-Based VM-Execution Controls
- Secondary Processor-Based VM-Execution Controls
- VM entry/exit controls
- GDTR/IDTR Base and Limit
- Segment Bases and Access Rights
- RFLAGS
- MSR/IO Bitmap addresses

VMCS Some Important Fields

- Pin-Based VM-Execution Controls
- Primary Processor-Based VM-Execution Controls
- Secondary Processor-Based VM-Execution Controls
- VM entry/exit controls
- GDTR/IDTR Base and Limit
- Segment Bases and Access Rights
- RFLAGS
- MSR/IO Bitmap addresses
- Host/Guest EFER

VMCS Some Important Fields

- Pin-Based VM-Execution Controls
- Primary Processor-Based VM-Execution Controls
- Secondary Processor-Based VM-Execution Controls
- VM entry/exit controls
- GDTR/IDTR Base and Limit
- Segment Bases and Access Rights
- RFLAGS
- MSR/IO Bitmap addresses
- Host/Guest EFER
- Host/Guest PAT

VMCS Some Important Fields

- Pin-Based VM-Execution Controls
- Primary Processor-Based VM-Execution Controls
- Secondary Processor-Based VM-Execution Controls
- VM entry/exit controls
- GDTR/IDTR Base and Limit
- Segment Bases and Access Rights
- RFLAGS
- MSR/IO Bitmap addresses
- Host/Guest EFER
- Host/Guest PAT
- Host/Guest CRx

VMCS Some Important Fields

- Pin-Based VM-Execution Controls
- Primary Processor-Based VM-Execution Controls
- Secondary Processor-Based VM-Execution Controls
- VM entry/exit controls
- GDTR/IDTR Base and Limit
- Segment Bases and Access Rights
- RFLAGS
- MSR/IO Bitmap addresses
- Host/Guest EFER
- Host/Guest PAT
- Host/Guest CRx
- Host/Guest RSP

VMCS Some Important Fields

- Pin-Based VM-Execution Controls
- Primary Processor-Based VM-Execution Controls
- Secondary Processor-Based VM-Execution Controls
- VM entry/exit controls
- GDTR/IDTR Base and Limit
- Segment Bases and Access Rights
- RFLAGS
- MSR/IO Bitmap addresses
- Host/Guest EFER
- Host/Guest PAT
- Host/Guest CRx
- Host/Guest RSP
- Host/Guest RIP

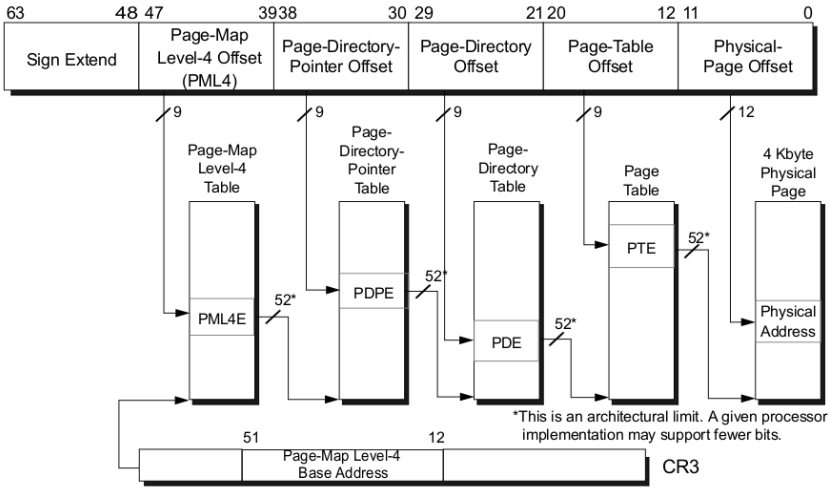
VMCS Some Important Fields

- Pin-Based VM-Execution Controls
- Primary Processor-Based VM-Execution Controls
- Secondary Processor-Based VM-Execution Controls
- VM entry/exit controls
- GDTR/IDTR Base and Limit
- Segment Bases and Access Rights
- RFLAGS
- MSR/IO Bitmap addresses
- Host/Guest EFER
- Host/Guest PAT
- Host/Guest CRx
- Host/Guest RSP
- Host/Guest RIP
- ...

EPT

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

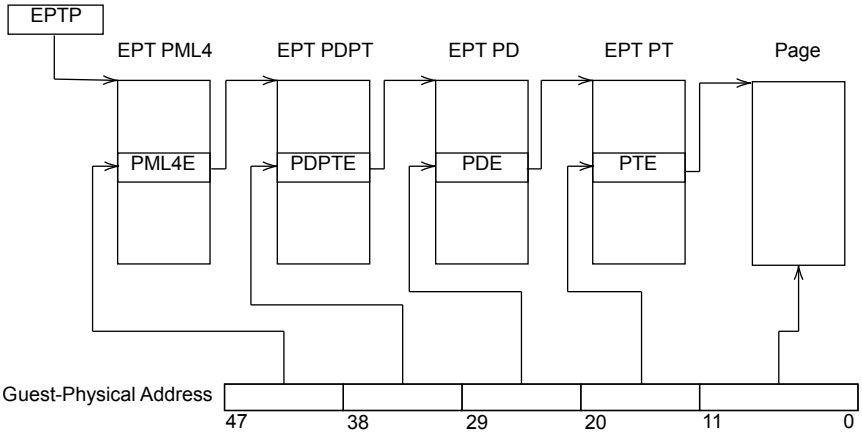
Remember paging?

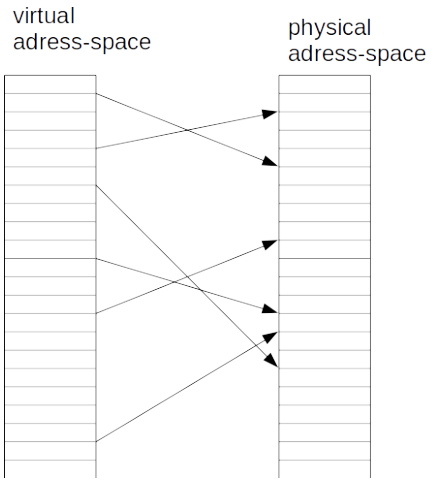


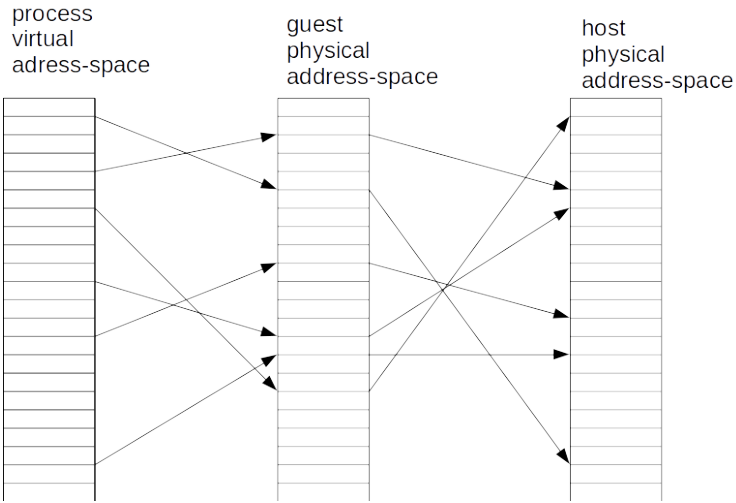
- If this works so well with linear addresses couldn't we do the same with guest-physical addresses?

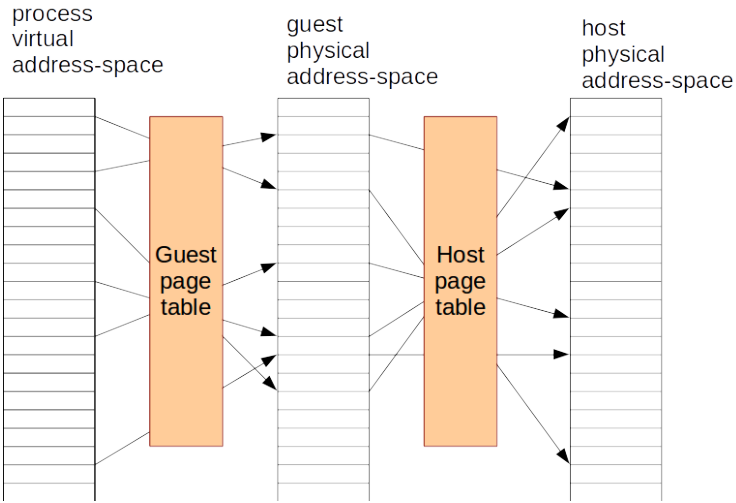

WE NEED TO GO DEEPER

Introducing Extended Page Tables

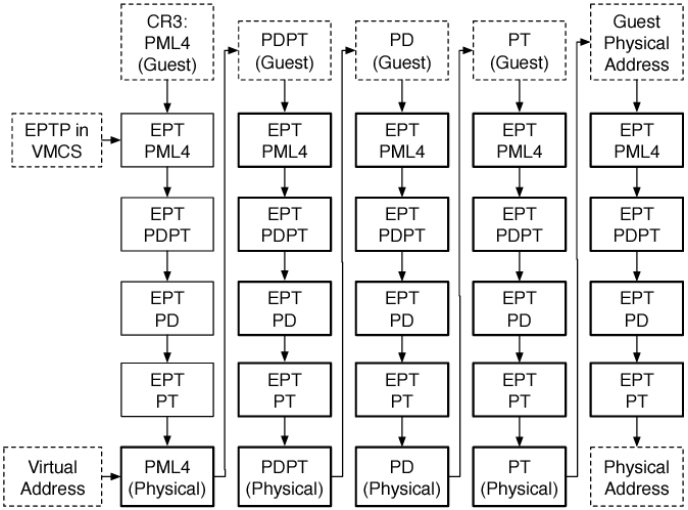



virtual
adress-space
physical
adress-space

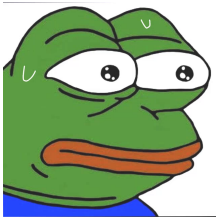

process
virtual
adress-space
guest
physical
address-space
host
physical
address-space


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

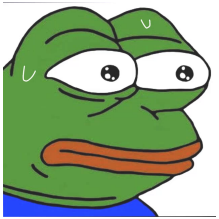
Isn't this slow?



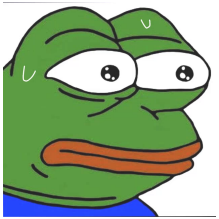
Caching saves us yet again



Caching saves us yet again

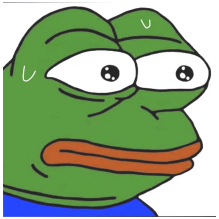


Caching saves us yet again



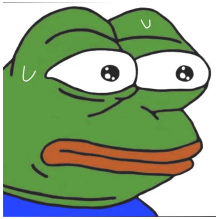
- EPT translations are cached in the TLB

Caching saves us yet again



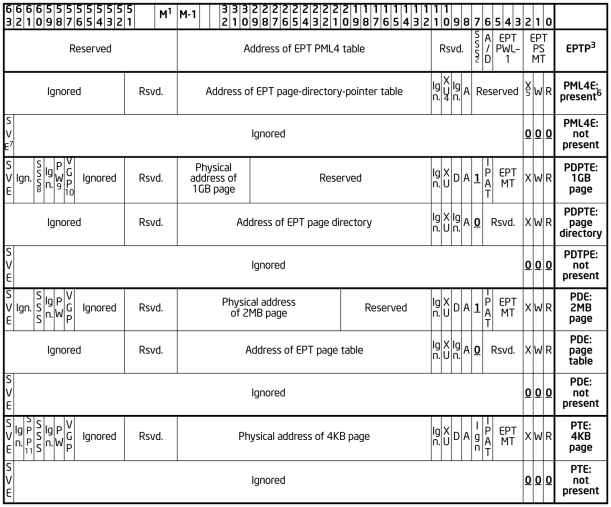
- EPT translations are cached in the TLB
- Translations are tagged with the EPTP

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


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
Ign.
A
Reserved
XS
W
R
PML4E: present[6]
SVE[7]
Ignored
0 0 0
PML4E: not present
SVE
Ign.
SSS
Ign.
PWL
VGP
Ignored
Rsvd.
Physical address of 1GB page
Reserved
Ign.
XU
D
A
1
PAT
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.
PWL
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]
SSS
Ign.
PWL
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

Running VMs

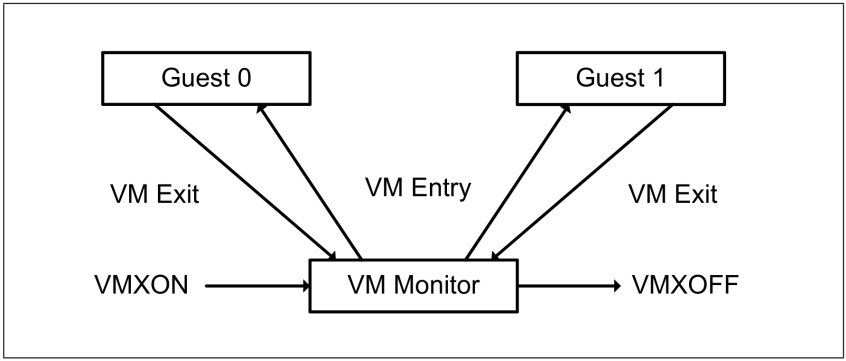
- VMLAUNCH

- VMLAUNCH
  - → VMCS not launched yet

- VMLAUNCH
  - → VMCS not launched yet
- VMRESUME

- VMLAUNCH
  - → VMCS not launched yet
- VMRESUME
  - → Continue launched VMCS

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


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

VM Exit

- An event that causes a VM exit does not update the architectural state

- An event that causes a VM exit does not update the architectural state
- If a VM exit is triggered by executing specified instructions we can gain further information

- An event that causes a VM exit does not update the architectural state
- If a VM exit is triggered by executing specified instructions we can gain further information
  - VM exit qualification $\rightarrow$ VM-exit reason specific information

- An event that causes a VM exit does not update the architectural state
- If a VM exit is triggered by executing specified instructions we can gain further information
  - VM exit qualification → VM-exit reason specific information
  - VM exit instruction length → length of the instruction that caused the VM exit

- An event that causes a VM exit does not update the architectural state
- If a VM exit is triggered by executing specified instructions we can gain further information
  - VM exit qualification → VM-exit reason specific information
  - VM exit instruction length → length of the instruction that caused the VM exit
  - VM exit instruction information → detailed information on the instruction that caused the VM exit

- An event that causes a VM exit does not update the architectural state
- If a VM exit is triggered by executing specified instructions we can gain further information
  - VM exit qualification → VM-exit reason specific information
  - VM exit instruction length → length of the instruction that caused the VM exit
  - VM exit instruction information → detailed information on the instruction that caused the VM exit
- Host RFLAGS is cleared (except bit 1)

Table 27-5. Exit Qualification for I/O Instructions

Bit Position(s)	Contents
2:0	Size of access: 0 = 1-byte 1 = 2-byte 3 = 4-byte Other values not used
3	Direction of the attempted access (0 = OUT, 1 = IN)
4	String instruction (0 = not string; 1 = string)
5	REP prefixed (0 = not REP; 1 = REP)
6	Operand encoding (0 = DX, 1 = immediate)
15:7	Not currently defined
31:16	Port number (as specified in DX or in an immediate operand)
63:32	Not currently defined. These bits exist only on processors that support Intel 64 architecture.

- Instructions that cause VM exits unconditionaly

- Instructions that cause VM exits unconditionaly
  - CPUID

- Instructions that cause VM exits unconditionaly
  - CPUID
  - GETSEC

VM Exit Reasons

- Instructions that cause VM exits unconditionaly
  - CPUID
  - GETSEC
  - INVD

VM Exit Reasons

- Instructions that cause VM exits unconditionaly
  - CPUID
  - GETSEC
  - INVD
  - XSETBV

VM Exit Reasons

- Instructions that cause VM exits unconditionaly
  - CPUID
  - GETSEC
  - INVD
  - XSETBV
  - VMX instructions (excluding VMFUNC)

VM Exit Reasons

- Instructions that cause VM exits unconditionaly
  - CPUID
  - GETSEC
  - INVD
  - XSETBV
  - VMX instructions (excluding VMFUNC)
- Instructions that cause VM exits conditionaly

- Instructions that cause VM exits unconditionaly
  - CPUID
  - GETSEC
  - INVD
  - XSETBV
  - VMX instructions (excluding VMFUNC)
- Instructions that cause VM exits conditionaly
  - RDMSR/WRMSR

VM Exit Reasons

- Instructions that cause VM exits unconditionaly
  - CPUID
  - GETSEC
  - INVD
  - XSETBV
  - VMX instructions (excluding VMFUNC)
- Instructions that cause VM exits conditionaly
  - RDMSR/WRMSR
  - IN/OUT

VM Exit Reasons

- Instructions that cause VM exits unconditionaly
  - CPUID
  - GETSEC
  - INVD
  - XSETBV
  - VMX instructions (excluding VMFUNC)
- Instructions that cause VM exits conditionaly
  - RDMSR/WRMSR
  - IN/OUT
  - MOV to/from CRx

VM Exit Reasons

- Instructions that cause VM exits unconditionaly
  - CPUID
  - GETSEC
  - INVD
  - XSETBV
  - VMX instructions (excluding VMFUNC)
- Instructions that cause VM exits conditionaly
  - RDMSR/WRMSR
  - IN/OUT
  - MOV to/from CRx
  - PAUSE

VM Exit Reasons

- Instructions that cause VM exits unconditionaly
  - CPUID
  - GETSEC
  - INVD
  - XSETBV
  - VMX instructions (excluding VMFUNC)
- Instructions that cause VM exits conditionaly
  - RDMSR/WRMSR
  - IN/OUT
  - MOV to/from CRx
  - PAUSE
  - ...

VM Exit Reasons

- Instructions that cause VM exits unconditionaly
  - CPUID
  - GETSEC
  - INVD
  - XSETBV
  - VMX instructions (excluding VMFUNC)
- Instructions that cause VM exits conditionaly
  - RDMSR/WRMSR
  - IN/OUT
  - MOV to/from CRx
  - PAUSE
  - ...
- Exceptions

VM Exit Reasons

- Instructions that cause VM exits unconditionaly
  - CPUID
  - GETSEC
  - INVD
  - XSETBV
  - VMX instructions (excluding VMFUNC)
- Instructions that cause VM exits conditionaly
  - RDMSR/WRMSR
  - IN/OUT
  - MOV to/from CRx
  - PAUSE
  - ...
- Exceptions
- VMX-preemption timer

VM Exit Reasons

- Instructions that cause VM exits unconditionaly
  - CPUID
  - GETSEC
  - INVD
  - XSETBV
  - VMX instructions (excluding VMFUNC)
- Instructions that cause VM exits conditionaly
  - RDMSR/WRMSR
  - IN/OUT
  - MOV to/from CRx
  - PAUSE
  - ...
- Exceptions
- VMX-preemption timer
- NMIs

VM Exit Reasons

- Instructions that cause VM exits unconditionaly
  - CPUID
  - GETSEC
  - INVD
  - XSETBV
  - VMX instructions (excluding VMFUNC)
- Instructions that cause VM exits conditionaly
  - RDMSR/WRMSR
  - IN/OUT
  - MOV to/from CRx
  - PAUSE
  - ...
- Exceptions
- VMX-preemption timer
- NMIs
- ...


[redacted]
VMM
GUEST
I REQUEST YOUR ASSISTANCE


[redacted]
VMM
GUEST
I REQUEST YOUR ASSISTANCE

- The only purpose of this instruction is to trigger a VM exit

- The only purpose of this instruction is to trigger a VM exit
- Can be used by the guest to specifically request something from the VMM

- The only purpose of this instruction is to trigger a VM exit
- Can be used by the guest to specifically request something from the VMM
- This instruction does nothing special

- The only purpose of this instruction is to trigger a VM exit
- Can be used by the guest to specifically request something from the VMM
- This instruction does nothing special
- It's literally just a normal VM exit


VIOLATION


VIOLATION

- Guest memory accesses that violate the permissions set in the EPT cause VM exits

- Guest memory accesses that violate the permissions set in the EPT cause VM exits
- The VMM can handle EPT violations accordingly

- Guest memory accesses that violate the permissions set in the EPT cause VM exits
- The VMM can handle EPT violations accordingly
- This gives us a lot of room to play around with:

- Guest memory accesses that violate the permissions set in the EPT cause VM exits
- The VMM can handle EPT violations accordingly
- This gives us a lot of room to play around with:
  - Copy on write

- Guest memory accesses that violate the permissions set in the EPT cause VM exits
- The VMM can handle EPT violations accordingly
- This gives us a lot of room to play around with:
  - Copy on write
  - EPT Hooking

- Guest memory accesses that violate the permissions set in the EPT cause VM exits
- The VMM can handle EPT violations accordingly
- This gives us a lot of room to play around with:
  - Copy on write
  - EPT Hooking
  - ...

- We can control I/O and MSR accesses

VM Exit Reasons - IN/OUT and RDMSR/WRMSR

- We can control I/O and MSR accesses
- Bitmaps define which I/O ports and MSRs cause VM exits

- We can control I/O and MSR accesses
- Bitmaps define which I/O ports and MSRs cause VM exits
- Giving the guest direct access to external hardware can be dangerous

- We can control I/O and MSR accesses
- Bitmaps define which I/O ports and MSRs cause VM exits
- Giving the guest direct access to external hardware can be dangerous
  - Guests could mess up the host state

- We can control I/O and MSR accesses
- Bitmaps define which I/O ports and MSRs cause VM exits
- Giving the guest direct access to external hardware can be dangerous
  - Guests could mess up the host state
  - DMAs don't care about our EPT

- We can force a VM exit as soon as the guest is able to recieve interrupts (RFLAGS.IF = 1)

- We can force a VM exit as soon as the guest is able to recieve interrupts (RFLAGS.IF = 1)
- Very useful for injecting interrupts

Questions?

Questions?

Questions?