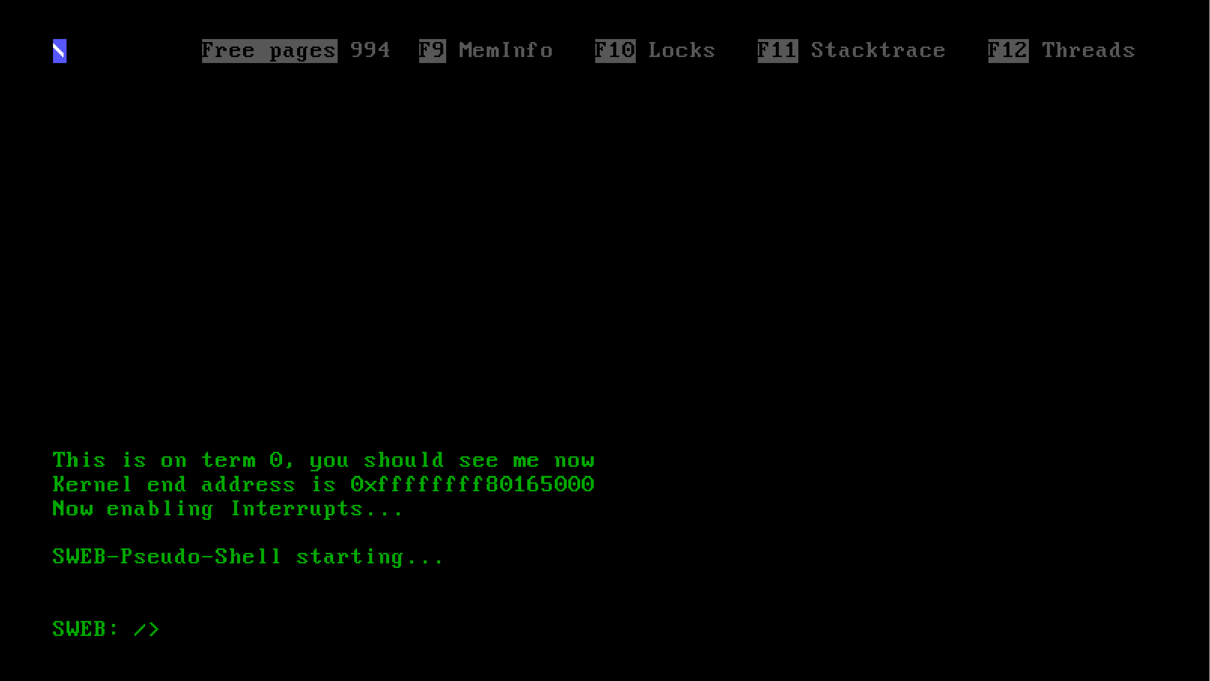
Cloud Operating Systems

Fabian Rauscher

2021-03-22

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://github.com/IAIK/sweb

Setup

- Upstream: https://github.com/IAIK/sweb
- We recommend you work on Linux

Setup

Setup

Setup

Building and running SWEB

Setup

Building and running SWEB

- mkdir -p /tmp/sweb

Setup

Building and running SWEB

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

Setup

Building and running SWEB

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

Setup

Building and running SWEB

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

Setup

Building and running SWEB

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

VMX

What is VMX?

What is VMX?

What is VMX?

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

Virtual-machine monitor

Virtual-machine monitor

Virtual-machine monitor

Virtual-machine monitor

- Gives the guest the illusion of running on real hardware

Virtual-machine monitor

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

Virtual-machine monitor

- 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

Virtual-machine monitor

- 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

Virtual-machine monitor

- 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

Virtual-machine monitor

- 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

Virtual-machine monitor

- 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/0

Virtual-machine monitor

- 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/0
  - ...

Getting Started

Enable VMX

Enable VMX

Enable VMX

Enable VMX

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

Enable VMX

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

VMX operation

VMX operation

VMX operation

VMX operation

- VMX root operation

VMX 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 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 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

VMX operation

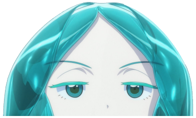
- VMX root operation
  - new instructions (VMX instructions)
  - restrictions on certain control register values
  - used for the VMM
- VMX non-root operation
  - more restricted than normal operation
  - certain instructions and events cause VM exits
  - used for the guest
- Transition between the two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

VMX operation

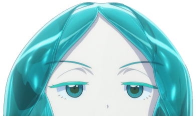
- VMX root operation
    - new instructions (VMX instructions)
    - restrictions on certain control register values
    - used for the VMM
- VMX non-root operation
    - more restricted than normal operation
    - certain instructions and events cause VM exits
    - used for the guest
- Transition between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

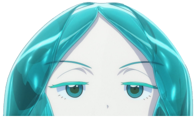


Entering VMX operation (Part 1)



Perpetrations

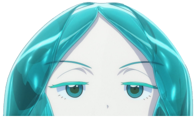
Entering VMX operation (Part 1)



Perpetrations

- Setup CR0

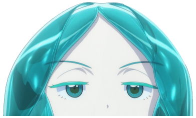
Entering VMX operation (Part 1)



Perpetrations

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

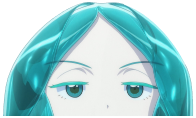
Entering VMX operation (Part 1)



Perpetrations

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

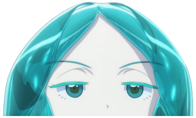
Entering VMX operation (Part 1)



Perpetrations

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

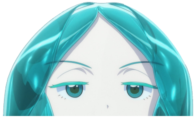
Entering VMX operation (Part 1)



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)

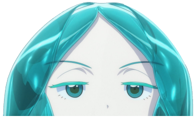
Entering VMX operation (Part 1)



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

New Instructions

New Instructions

New Instructions

New Instructions

- VMXON

New Instructions

- VMXON
- VMXOFF

New Instructions

- VMXON
- VMXOFF
- INVEPT

New Instructions

- VMXON
- VMXOFF
- INVEPT
- INVVPID

New Instructions

- VMXON
- VMXOFF
- INVEPT
- INVVPID
- VMCALL

New Instructions

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

New Instructions

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

New Instructions

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

New Instructions

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

New Instructions

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

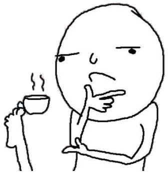
New Instructions

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

VMCS

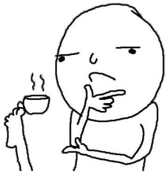
What is a VMCS?

What is a VMCS?

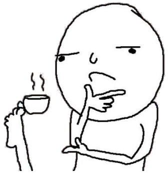


What is a VMCS?

Virtual-Machine Control Data Structure (VMCS)



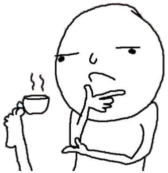
What is a VMCS?



Virtual-Machine Control Data Structure (VMCS)

- Manages VM entries and VM exits

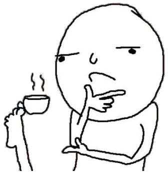
What is a VMCS?



Virtual-Machine Control Data Structure (VMCS)

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

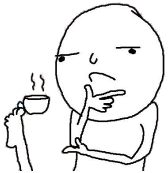
What is a VMCS?



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

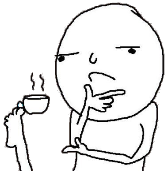
What is a VMCS?



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

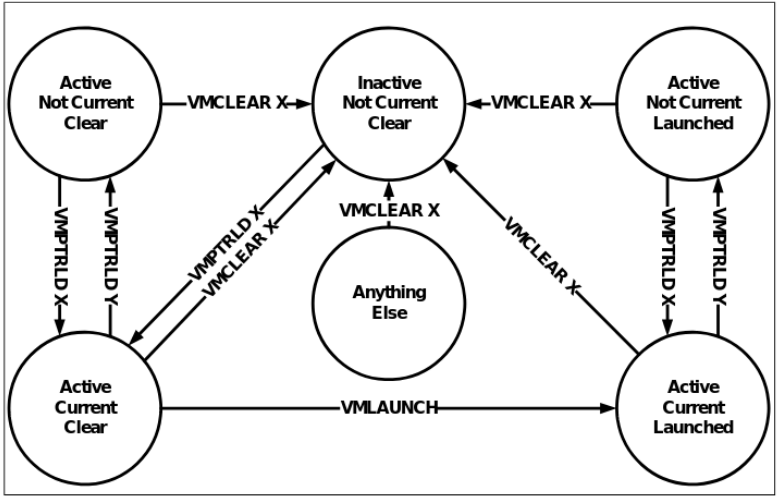
VMCS region

- 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)

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
VMCLEAR X
Anything Else
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

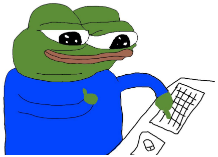
Guest interrupt status
PML index

HOST STATE AREA

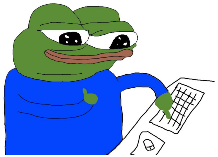
CR0	CR3	CR4
RSP	RIP	
CS	Selector	
SS	Selector	
DS	Selector	
ES	Selector	
FS	Selector	Base Address
GS	Selector	Base Address
TR	Selector	Base Address
GDTR	Base Address	
IDTR	Base Address	
IA32_SYSENTER_CS	IA32_SYSENTER_ESP	IA32_SYSENTER_EIP
IA32_PERF_GLOBAL_CTRL	IA32_PAT	IA32_EFER

Guest State

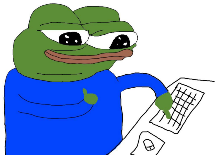
Guest State



Guest State

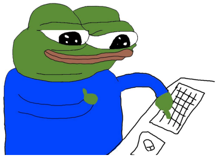


Guest State



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

Guest State



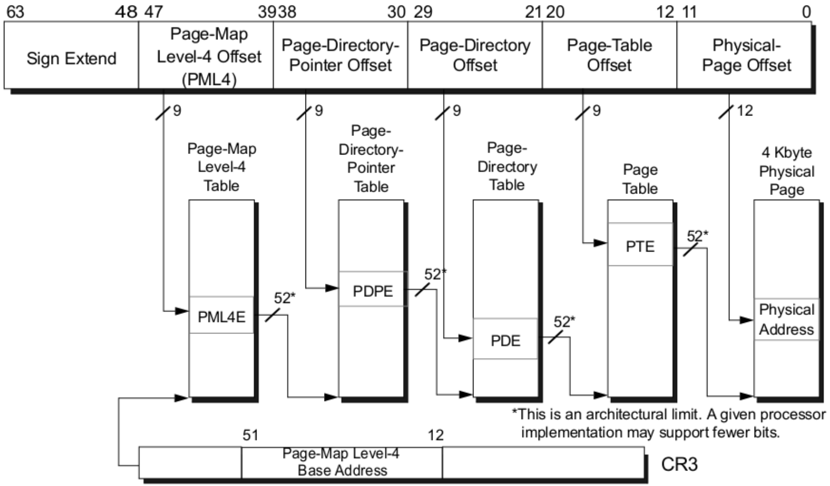
- What about things that are not in the guest state area, like general-purpose registers?
- We have to save and restore them by hand

EPT

Paging

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

Remember paging?

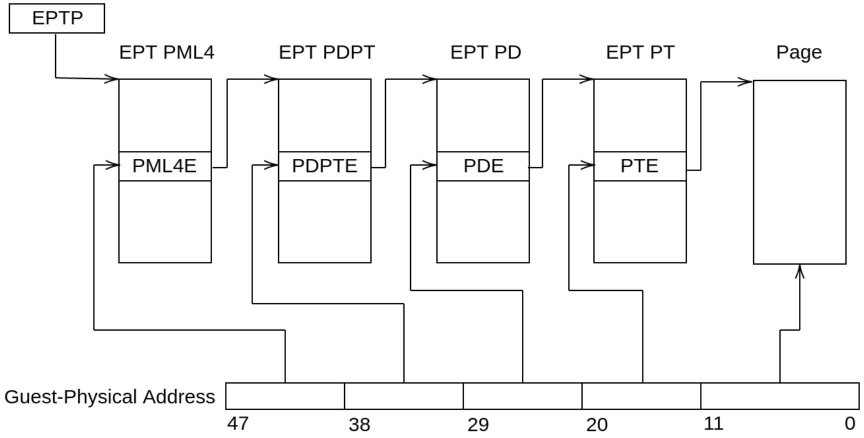


Virtualizing physical memory

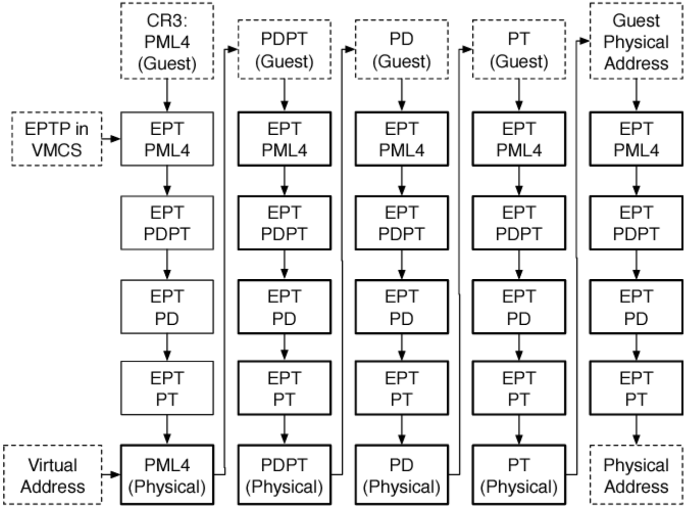
- If this works so well with linear addresses couldn't we do the same with guest-physical addresses?

WE NEED TO GO DEEPER
quickmeme.com

Introducing Extended Page Tables

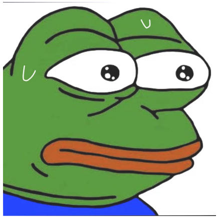


Isn't this slow?

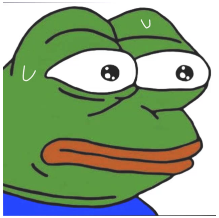


Caching saves us yet again

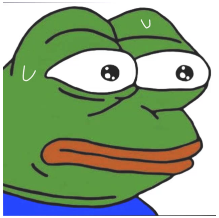
Caching saves us yet again



Caching saves us yet again

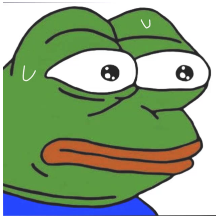


Caching saves us yet again



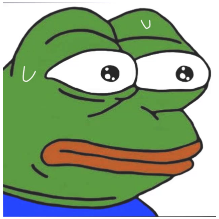
- EPT translations are cached in the TLB

Caching saves us yet again



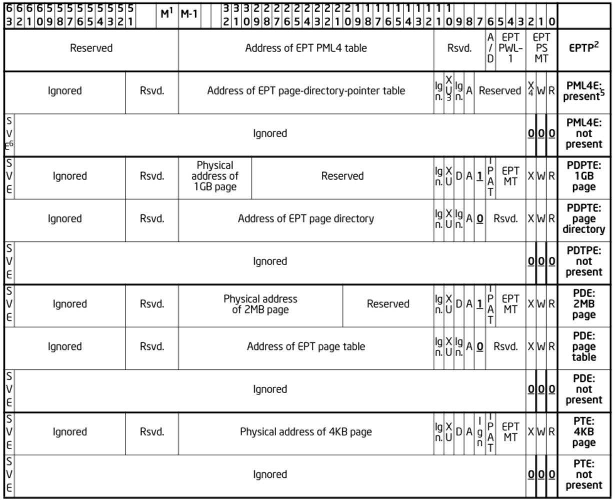
- EPT translations are cached in the TLB
- Translations are tagged with the EPTP

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 … 52	51 … M[1]	M-1 …	… 12	11	10	9	8	7	6	5 4 3	2	1	0	
Reserved			Address of EPT PML4 table		Rsvd.					A/D	EPT PWL-1	EPT PS MT			**EPTP[2]**
Ignored		Rsvd.	Address of EPT page-directory-pointer table		Ign.	XU[3]	Ign.	A	Reserved			X[4]	W	R	**PML4E: present[5]**
SVE[6]	Ignored											0	0	0	**PML4E: not present**
SVE	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	Ignored	Rsvd.	Physical address of 2MB page	Reserved	Ign.	XU	D	A	1	IPAT	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	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**

VM Exit

VM exit

VM exit

VM exit

VM exit

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

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

- 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
    - The VM-exit instruction length field contains the length of the instruction that caused the VM exit

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
  - The VM-exit instruction length field contains the length of the instruction that caused the VM exit
  - The VM-exit instruction information contains detailed information on the instruction that caused the VM exit

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
    - The VM-exit instruction length field contains the length of the instruction that caused the VM exit
    - The VM-exit instruction information contains detailed information on the instruction that caused the VM exit
- Host RFLAGS is cleared except for bit 1

VM exit reasons

- Instructions that cause VM exits unconditionaly

VM exit reasons

- Instructions that cause VM exits unconditionaly
  - CPUID

VM exit reasons

- 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

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

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
- ...

VM exit reasons - VMCALL

VM exit reasons - VMCALL

VM exit reasons - VMCALL

VM exit reasons - VMCALL

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

VM exit reasons - VMCALL

- 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

VM exit reasons - VMCALL

- 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

VM exit reasons - VMCALL

- 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

VM exit reasons - EPT violation

VM exit reasons - EPT violation

VM exit reasons - EPT violation

VM exit reasons - EPT violation

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

VM exit reasons - EPT violation

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

VM exit reasons - EPT violation

- 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:

VM exit reasons - EPT violation

- 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

VM exit reasons - EPT violation

- 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

VM exit reasons - EPT violation

- 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
    - ...

VM exit reasons - IN/OUT and RDMSR/WRMSR

- 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

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
- Giving the guest direct access to external hardware can be dangerous

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
- Giving the guest direct access to external hardware can be dangerous
    - Guests could mess up the host state

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
- 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

VM exit reasons - Interrupt-Windows

VM exit reasons - Interrupt-Windows

VM exit reasons - Interrupt-Windows

VM exit reasons - Interrupt-Windows

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

VM exit reasons - Interrupt-Windows

- 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?