VTx Hypervisor

31 July 2020 12:06

In order to enter a VM, we need to do several things:

1. Turn on VMM with `VMXON` instruction.
2. Enter guest with VM Entry
    a. Exit guest with VM Exit
3. Turn off VMM with VMXOFF instruction

Non-root and root transitions are controlled by a data structure called the VMCS and is accessible by a point which is present 1 in each core.

- Can read the VMCS pointer with VMPTRST
- Can write to the VMCS write VMPTRST

The VMM configures a VMCS using VMREAD, VMWRITE, and VMCLEAR

Checking for VMX availability

Can check with the following conditions:

- Check if vendor is Intel
- CPUID.1:ECX.VMX[bit 5] = 1

Entering VMXON Mode

In order to enter VMXON mode, the driver has to do the following things:

1. Set the CR4.VMXE[bit 13] = 1.
2. Use the MSR IA32_FEATURE_CONTROL (msr at 0x3A) in the following way:
    a. Clear the lock bit (first bit), and set it to 0
    b. Clear the SMX operation bit (second bit) and set it to 0
    c. Set the SMX Out of operation bit (third bit) and set it to 1
3. Allocate 4KB Aligned region of memory for the VMXON to use. (the address to that region is provided as an operand of VMXON)
4. Set the following values to 1:
    a. CR0.PE, CR0.NE, CR0.PG, CR4.VMXE

Leaving VMXON Mode

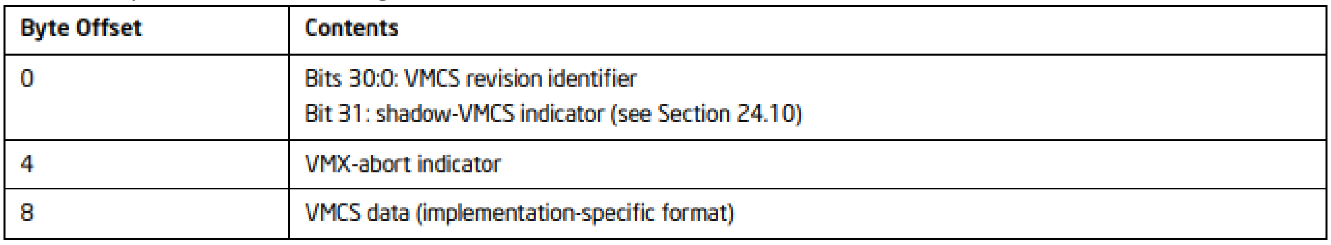
In order to leave VMXON mode, you cannot change CR4.VMXE[bit 13]=0, instead you have to use the VMXOFF instruction (since the VMM cannot modify CR4.VMXE directly)

Virtual Machine Control Data Structures

- Each logical processor has a separate VMCS structure, which stores and manages information about transitions with VM exits, and VM entries.
- The structure can only be manipulated by the following instructions:
    - VMCLEAR - The operand to this instruction is an address of a VMCS, after executing this instrution the VMCS is neither active nor current on the logical processor. If the VMCS was current, the processor no longer has a current VMCS.
    - VMPTRLD - Marks the current VMCS pointer valid and loads it with the physical address in the instruction operand.
        - Fails if:
            - Operand is not properly aligned
            - Sets unsupported physical bits
            - Is equal to the VMXON pointer
            - Does not match the VMCS revision identifier supported by the processor
    - VMREAD -
    - VMWRITE -
- A logical processor associates a region in memory with each VMCS, and that region is called the **VMCS REGION**. VMCS pointers must be aligned on a 4KB boundary.
- The following instructions operate on the "current" VMCS:

    - VMLAUNCH -
    - VMREAD -
    - VMRESUME -
    - VMWRITE -
- VMPTRST - Getting the current VMCS (remember it differs on each processor), is done with the VMPTRST instruction. It stores the address of the current VMCS into a specified location, and if there is no VMCS it stores the value 0xFFFFFFFF_FFFFFFFF.
- States of the VMCS:
    - VMLAUNCH - state = clear, execute and switch state to launched
    - VMRESUME - state = launched,
    - VMCLEAR - Modifies state = clear
    - VMWRITE Cannot modify the state of the VMCS, and there is no direct way to discover it (cannot be read using VMREAD)

The VMCS is comprised of the following structure:

- 

Byte Offset	Contents
0	Bits 30:0: VMCS revision identifier Bit 31: shadow-VMCS indicator (see Section 24.10)
4	VMX-abort indicator
8	VMCS data (implementation-specific format)

  Explanation:
    - The revision identifier is an identifier which allows processors that maintain VMCS data in different formats to avoid using a VMCS region formatted for one processor on a processor that uses a different format.
        - Bit 31 indicates whether the VMCS is a shadow VMCS.
    - The software writes the VMCS revision identifier
        - VMPTRLD fails if its operand references a VMCS region whose region identifier differs from the one used by the current processor.

The VMCS structure is organized and divided to the following groups:
- **Guest-state area**. Processor state is saved into the guest-state area on VM exits and loaded on VM entries.
- **Host-state area.** Processor state is saved into the host-state area on VM entries, and VM exits.
- **VM-execution control fields.** These fields control processor behavior in VMX non-root operation and determine the causes of the VM exits.
- **VM-exit control fields.**
- **VM-entry control fields**
- (read-only) **VM-exit information fields.** These fields receive information on VM exits and describe the cause of VM exits.

Guest-state area

- Most important registers saved inside the guest-state area:
    - CR0, CR3, CR4
    - Debug register DR7
    - RSP, RIP, RFLAGS
    - CS, SS, DS, ES, FS, GS, LDTR, TR
    - GDTR, IDTR
- In addition to registers, the activity state is saved which is:
    - 0 - Active: the logical processor is executing instructions normally
    - 1 - HLT: the logical processor is inactive because it executed the hlt instruction
    - 2 - Shutdown: The logicaly processor is inactive because it incurred a triple fault or a serious error
    - 3 - Wait for SIPI
- The VMCS link pointer: If VMCS Shadowing is enabled, VMREAD and VMWRITE instructions access the VMCS referenced by this pointer, otherwise this pointer should be 0xFFFFFFFF_FFFFFFFF
- PDPTE entries.

Host-state area

The host state area only includes the above registers.

VM Execution control fields

Primary parts:

- Pin-Based VM-Exection controls - 32 bit vector constitutes the following:
  - 0 - External interrupt exiting (exits if external interrupts occur)
  - 2:1 - Reserved
  - 3 - NMI Exiting (exits if NMIs occur)
  - 4 - Reserved
  - 5 - Virtual NMIs (NMIs never blocked)
  - 6 - Activate VMX - preemption timer (preemption timer counts down, and an exit occurs when timer counts down to 0)
  - 7 - Process posted interrupts (processor treats interrupts with the posted-interrupt notification vector)
  - 31:7
- Processor Based VM Execution Controls
  - 1:0 - Reserved
  - 2 - VM exit occurs at any instruction if interrupts are enabled
  - 3 - Use TSC offsetting (allows to modify time)
  - 6:4 - Reserved
  - 7 - HLT Exiting (HLT causes VMExit)
  - 8 - Reserved
  - 9 - INVLPG Exiting
  - 10 - MWAIT Exiting
  - 11 - RDPMC Exiting
  - 12 RDSTC Exiting (also includes RDTSCP)
  - 14:13 - Reserved
  - 15 - CR3 Load Exiting (decides whether MOVs to CR3 cause exit)
  - 16 - CR3 Store Exiting (decides whether MOVs from CR3 cause exit)
  - 18:17 - Reserved
  - 19 - CR8 Load Exiting (same as CR3)
  - 20 - CR8 Store Exiting (same as CR3)
  - 21 - Use TPR Shadow (enables TPR virtualization, and APIC virtualization)
  - 22 - NMI Window Exiting (exits if no virtual NMI blocking)
  - 23 - MOV-DR Exiting (MOV DR causes exit)
  - 24 - Unconditional IO Exiting (exits if IN, INS/INSB/INSW/INSD, OUT, OUTS/OUTSB/OUTSW/OUTSD executed)
  - 25 - Use I/O Bitmaps (0 => don't use bitmaps, 1 => use bitmaps)
  - 26 - Reserved
  - 27 - Monitor Trap Flag (monitor trap flag debugging features enabled)
  - 28 - Use MSR Bitmaps (virtualize MSR bitmaps - RDMSR / WRMSR)
  - 29 - MONITOR Exiting
  - 30 - PAUSE Exiting
  - 31 - Activate secondary controls (whether to use a second control structure)
- Secondary Processor Based VM Execution Controls - 32 bits
  - 0 - Virtualize APIC
  - 1 - Enable EPT
  - 2 - Descriptor Table Exiting (LGDT, LIDT, LLDT, LTR, SGDT, SIDT, SLDT, STR cause exits)
  - 3 - Enable RDTSCP
  - 4 - Virtualize x2 APIC (treats APIC MSRs specially, range: 0x800 - 0x8FF)
  - 5 - Enable VPID
  - 6 - WBINVD Exiting
  - 7 - Unrestricted Guest (guest software may run in unpaged protected mode or in real-address mode)
  - 8 - APIC Register Virtualization
  - 9 - Virtual Interrupt Delivery (enables evaluation and delivery of pending virtual interrupts)
  - 10 - PAUSE-loop exiting
  - 11 - RDRAND Exiting
  - 12 - Enable INVPCID
  - 13 - Enable VM Functions
  - 14 - VMCS Shadowing (allows nested virtualization)
  - 15 - Enable ENCLS Exiting
  - 16 - RDSEED Exiting

    - 17 - Enable PML
    - 18 - EPT Violation Exiting
    - 19 - Conceal VMX non-root operation from intel PT
    - 20 - Enable XSAVES / XRSTORS
    - 21 - Reserved
    - 22 - Mode-based execute control for EPT - Linear address supervisor mode or user mode
    - 24:23 - Reserved
    - 25 - Use TSC Scaling - Same as TSC offsetting, but this time is multiplying
    - 31:26 - Reserved

- EPT (only available on processors that support the 1-setting of the enable EPT:
    - Contains the address of the EPT PML4 table, as well as other EPT configuration
    - 2:0 Paging structure memory type -
        - 0 = Uncacheable (UC)
        - 6 = Write-back (WB)
        - Other values are reserved
    - 5:3 - 1 less than EPT page-walk length
    - 6 - Enable accessed and dirty flags for the EPT
    - 11:7 - Reserved
    - N-1:12 - Physical address of the EPT PML4 table (can determine N with CPUID 0x80000008, the physical address width is returned in bits 7:0 of eax (al))
    - 63:N - Reserved
- VPID (virtual processor identifier) - 16 bit field [if enable PID is on]
- PAUSE-Loop exiting - 32 bit field [if PAUSE-loop exiting is on]
- VM Function controls - 64 bit field [if activate secondary-controls and enable VM function is on]
- VMCS Shadowing Bitmap Address - 64 bit field [if VMCS shadowing is enabled]
- ENCLS-Exiting Bitmap - 64 bit field [if enable ENCLS exiting is on]
- PML Address - 64 bit field [if enable PML is 1]

VM Exit Control Fields

VM Exit controls:

- 1:0 - Reserved
- 2 - Save debug controls
- 8:3 - Reserved
- 9 - Host address space size (must be 1 on x64 intel)
- 11:10 - Reserved.
- 12 - Load IA32_PERF_GLOBAL_CTRL on VMExit
- 14:13 - Reserved
- 15 - Acknowledge interrupt on exit (Affects VM exits due to external interrupts)
- 17:16 - Reserved
- 18 - Save IA32_PAT MSR on VM Exit
- 19 - Load IA32_PAT MSR on VM Exit
- 20 - Save IA32_EFER MSR on VM Exit
- 21 - Load IA32_EFER MSR on VM Exit
- 22 - Save VMX-preemption timer on VM Exit
- 23 - Clear IA32_BNDCFGS on exit
- 24 - Conceal vm exits from intel PT
- 31:25 - Reserved

VM Entry Control Fields

- 1:0 - Reserved
- 2 - Load Debug Controls
- 8:3 - Reserved
- 9 - IA-32e mode guest (allows x64 bit processors to enter x32 bit mode after entry)
- 10 - Entry to SMM (system management mode)
- 11 - Deactivate dual monitor treatment
- 12 - Reserved
- 13 - Load IA32_PERF_GLOBAL_CTRL MSR on VM Entry

- 14 - Load IA32_PAT on VM Entry
- 15 - Load IA32_EFER on VM Entry
- 16 - Load IA32_BNDCFGS on VM Entry
- 17 - Conceal VM entries from intel PT
- 31:18 - Reserved

VM Exit Information

- 15:0 - Basic exit reason
- 26:16 - Reserved
- 27 - VM exit saves this bit to 1 to indicate a vm exit was incident to enclave mode
- 28 - Pending MTF VM Exit (set by SMMs)
- 29 - VM exit from VMX root operation
- 30 - Reserved
- 31 - VM entry failure

VMCS Migrations

In order to migrate a VMCS from one processor to another, we need to execute the following:

1. VMCLEAR on the 1st processor
2. Wait for VMCLEAR to end and then use VMPTRLD on the 2nd processor

Encodings of the VMCS fields

In order to use VMREAD, VMWRITE we need to specify an **encoding** as an argument, which is basically a field unique identifier. The encoding is determined by the width of the fields and their function in the vmcs:

- 0 - Access type:
    - 0 = 0 full
    - 1 = 1 high (must be full for 16-bit, 32-bit and natural-width fields)
- 9:1 - Index of the field
- 11:10 - Type:
    - 0 - Control
    - 1 - VM Exit Information
    - 2 - Guest State
    - 3 - Host State
- 12 - Reserved
- 14:13 - Width:
    - 0 - 16 bit
    - 1 - 64 bit
    - 2 - 32 bit
    - 3 - natural width
- 31:15 - Reserved

Sources:
https://wiki.osdev.org/CPUID
https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-vol-3c-part-3-manual.pdf
http://tce.webee.eedev.technion.ac.il/wp-content/uploads/sites/8/2015/09/BC_Micro-VMMs-and-Nested-Virtualization.pdf
https://www.digitalwhisper.co.il/files/Zines/0x61/DW97-3-PageTableExp.pdf