entering hardware-virtualized world

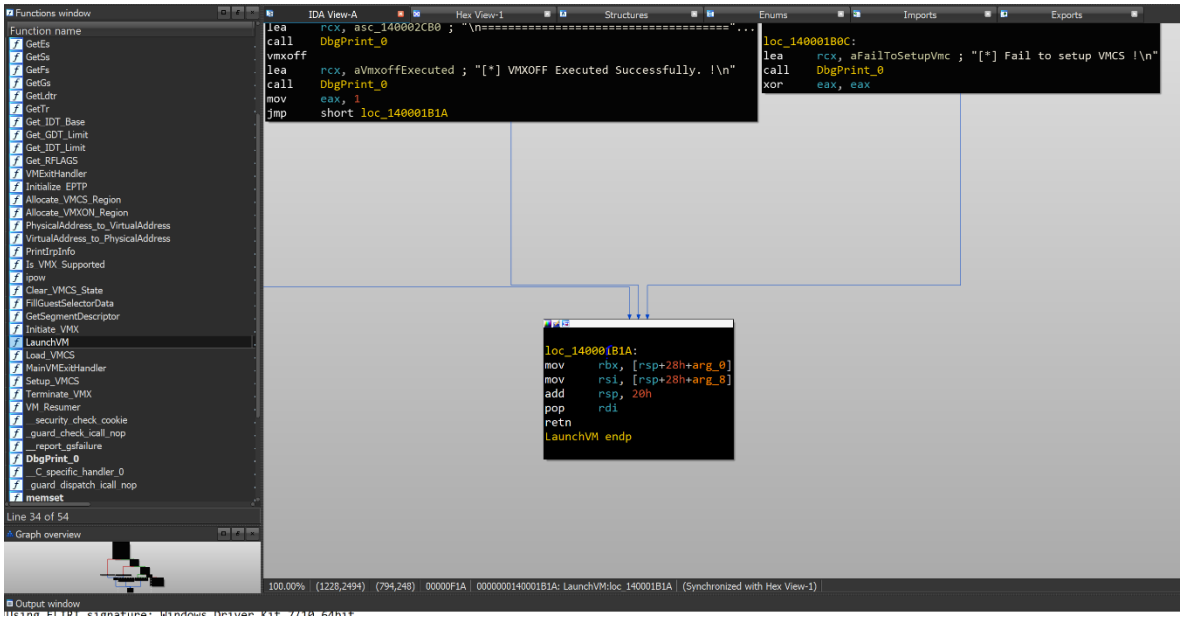
בתרגיל הקודם הכנו תמיכה ב EPT להרצה של ה VM שלנו.
כעת, אנחנו נוסיף ל Hypervisor אתחול של ה VMCS ונריץ קוד של ה VM לראשונה ! (תתרגשו)

DRIVER

כפי שניתן לראות הזזנו את כל הלוגיקה העיקרית ל DriverEntry כאשר הפונקציות initialize_EPTP, initialize_VMX כבר מוכרות לנו. שימו לב שהמשתנה הגלובלי VirtualGuestMemoryAddress מאותחל כאשר מאתחלים את ה EPTP (זה הזיכרון שהקצנו בגודל 10 דפים בתרגיל הקודם, כעת הרחבנו את זה ל 100 ושמרנו אותו במשתנה גלובלי).
אתחלנו את כל הזיכרון של ה VM בפקודה HLT שתגרור VM-Exit כאשר ה guest ינסה להריץ אותה.
לאחר מכן קראנו לפונקציה LaunchVM עם המעבד שעליו נריץ את ה VM (נניח המעבד הראשון) ועם ה EPTP שבנינו.

LaunchVM

נזכור להריץ את כל הקוד שלנו על המעבד עליו אנו מעוניינים להריץ את ה VM שקיבלנו כפרמטר.
נקצה stack עבור ה VMM שלנו כאשר יקפוץ VM-Exit (ניתן להשתמש גם ב stack הנוכחי), בנוסף נקצה מקום עבור ה MSRBitMap עליו נפרט בתרגיל הבא.
כעת נשתמש ב VMCLEAR ו VMPTRLD על מנת לעדכן את המצביע ל VMCS הנוכחי שלנו, חשוב לזכור שהוא מאופס (חוץ מה revision ID) ולכן נקרא לפונקציה Setup_VMCS על מנת לאתחל את כל השדות שלו.
לבסוף, שנייה לפני שנקרא ל VMLAUNCH נזכור לשמור את המצב הנוכחי (RSP+RBP) על מנת שנוכל לשחזר אותו כאשר נרצה לצאת מ VM-Exit ולקרוא ל VMXOFF על מנת לסיים את הוירטואליזציה (כאשר אנחנו ב VM-Exit ה RSP לא שלנו כי החלטנו להקצות אחד בשביל ה VMM, וכמובן שגם ה RBP נדרס בתחילת הפונקציה ולכן יש לשחזר אותם).
שימו לב שהשחזור די רגיש כיוון שהוא ממומש על בסיס הסתכלות ב IDA על Launch_VM:



Setup_VMCS

בעיקר אתחול של כל השדות שיש ב VMCS שאין צורך להתעקב עליהם - מוזמנים לקרוא את הקוד \ intel manual.
מאתחלים את השדות CPU_BASED_VM_EXEC_CONTROL, SECONDARY_VM_EXEC_CONTROL:

Table 24-6. Definitions of Primary Processor-Based VM-Execution Controls

Bit Position(s)	Name	Description
2	Interrupt-window exiting	If this control is 1, a VM exit occurs at the beginning of any instruction if RFLAGS.IF = 1 and there are no other blocking of interrupts (see Section 24.4.2).
3	Use TSC offsetting	This control determines whether executions of RDTSC, executions of RDTSCP, and executions of RDMSR that read from the IA32_TIME_STAMP_COUNTER MSR return a value modified by the TSC offset field (see Section 24.6.5 and Section 25.3).
7	HLT exiting	This control determines whether executions of HLT cause VM exits.
9	INVLPG exiting	This determines whether executions of INVLPG cause VM exits.
10	MWAIT exiting	This control determines whether executions of MWAIT cause VM exits.
11	RDPMC exiting	This control determines whether executions of RDPMC cause VM exits.
12	RDTSC exiting	This control determines whether executions of RDTSC and RDTSCP cause VM exits.
15	CR3-load exiting	In conjunction with the CR3-target controls (see Section 24.6.7), this control determines whether executions of MOV to CR3 cause VM exits. See Section 25.1.3. The first processors to support the virtual-machine extensions supported only the 1-setting of this control.
16	CR3-store exiting	This control determines whether executions of MOV from CR3 cause VM exits. The first processors to support the virtual-machine extensions supported only the 1-setting of this control.
19	CR8-load exiting	This control determines whether executions of MOV to CR8 cause VM exits.
20	CR8-store exiting	This control determines whether executions of MOV from CR8 cause VM exits.
21	Use TPR shadow	Setting this control to 1 enables TPR virtualization and other APIC-virtualization features. See Chapter 29.
22	NMI-window exiting	If this control is 1, a VM exit occurs at the beginning of any instruction if there is no virtual-NMI blocking (see Section 24.4.2).
23	MOV-DR exiting	This control determines whether executions of MOV DR cause VM exits.
24	Unconditional I/O exiting	This control determines whether executions of I/O instructions (IN, INS/INSB/INSW/INSD, OUT, and OUTS/OUTSB/OUTSW/OUTSD) cause VM exits.
25	Use I/O bitmaps	This control determines whether I/O bitmaps are used to restrict executions of I/O instructions (see Section 24.6.4 and Section 25.1.3). For this control, "0" means "do not use I/O bitmaps" and "1" means "use I/O bitmaps." If the I/O bitmaps are used, the setting of the "unconditional I/O exiting" control is ignored.
27	Monitor trap flag	If this control is 1, the monitor trap flag debugging feature is enabled. See Section 25.5.2.
28	Use MSR bitmaps	This control determines whether MSR bitmaps are used to control execution of the RDMSR and WRMSR instructions (see Section 24.6.9 and Section 25.1.3). For this control, "0" means "do not use MSR bitmaps" and "1" means "use MSR bitmaps." If the MSR bitmaps are not used, all executions of the RDMSR and WRMSR instructions cause VM exits.
29	MONITOR exiting	This control determines whether executions of MONITOR cause VM exits.
30	PAUSE exiting	This control determines whether executions of PAUSE cause VM exits.
31	Activate secondary controls	This control determines whether the secondary processor-based VM-execution controls are used. If this control is 0, the logical processor operates as if all the secondary processor-based VM-execution controls were also 0.

Table 24-7. Definitions of Secondary Processor-Based VM-Execution Controls

Bit Position(s)	Name	Description
0	Virtualize APIC accesses	If this control is 1, the logical processor treats specially accesses to the page with the APIC-access address. See Section 29.4.
1	Enable EPT	If this control is 1, extended page tables (EPT) are enabled. See Section 28.2.
2	Descriptor-table exiting	This control determines whether executions of LGDT, LIDT, LLDT, LTR, SGDT, SIDT, SLDT, and STR cause VM exits.
3	Enable RDTSCP	If this control is 0, any execution of RDTSCP causes an invalid-opcode exception (#UD).
4	Virtualize x2APIC mode	If this control is 1, the logical processor treats specially RDMSR and WRMSR to APIC MSRs (in the range 800H–8FFH). See Section 29.5.
5	Enable VPID	If this control is 1, cached translations of linear addresses are associated with a virtual-processor identifier (VPID). See Section 28.1.
6	WBINVD exiting	This control determines whether executions of WBINVD cause VM exits.
7	Unrestricted guest	This control determines whether guest software may run in unpaged protected mode or in real-address mode.
8	APIC-register virtualization	If this control is 1, the logical processor virtualizes certain APIC accesses. See Section 29.4 and Section 29.5.
9	Virtual-interrupt delivery	This controls enables the evaluation and delivery of pending virtual interrupts as well as the emulation of writes to the APIC registers that control interrupt prioritization.
10	PAUSE-loop exiting	This control determines whether a series of executions of PAUSE can cause a VM exit (see Section 24.6.13 and Section 25.1.3).
11	RDRAND exiting	This control determines whether executions of RDRAND cause VM exits.
12	Enable INVPCID	If this control is 0, any execution of INVPCID causes a #UD.
13	Enable VM functions	Setting this control to 1 enables use of the VMFUNC instruction in VMX non-root operation. See Section 25.5.5.
14	VMCS shadowing	If this control is 1, executions of VMREAD and VMWRITE in VMX non-root operation may access a shadow VMCS (instead of causing VM exits). See Section 24.10 and Section 30.3.
15	Enable ENCLS exiting	If this control is 1, executions of ENCLS consult the ENCLS-exiting bitmap to determine whether the instruction causes a VM exit. See Section 24.6.16 and Section 25.1.3.
16	RDSEED exiting	This control determines whether executions of RDSEED cause VM exits.
17	Enable PML	If this control is 1, an access to a guest-physical address that sets an EPT dirty bit first adds an entry to the page-modification log. See Section 28.2.5.
18	EPT-violation #VE	If this control is 1, EPT violations may cause virtualization exceptions (#VE) instead of VM exits. See Section 25.5.6.
19	Conceal VMX from PT	If this control is 1, Intel Processor Trace suppresses from PIPs an indication that the processor was in VMX non-root operation and omits a VMCS packet from any PSB+ produced in VMX non-root operation (see Chapter 35).
20	Enable XSAVES/XRSTORS	If this control is 0, any execution of XSAVES or XRSTORS causes a #UD.
22	Mode-based execute control for EPT	If this control is 1, EPT execute permissions are based on whether the linear address being accessed is supervisor mode or user mode. See Chapter 28.
25	Use TSC scaling	This control determines whether executions of RDTSC, executions of RDTSCP, and executions of RDMSR that read from the IA32_TIME_STAMP_COUNTER MSR return a value modified by the TSC multiplier field (see Section 24.6.5 and Section 25.3).

לבסוף, 3 אתחולים חשובים נוספים ששוה להזכיר הם של ה RIP וה RSP של ה guest וה host.
עבור ה guest, לעת עתה נאתחל את RSP + RIP לזכרון שהקצנו ואתחלנו עם HLT למען הבדיקה.
עבור ה host, נאתחל את ה RSP עם ה RSP החדש שהקצנו (שוב, לא היינו חייבם אבל זה נקי יותר).
את ה RIP נאתחל בפונקציה שמטפלת לנו ב VM-Exit - VMExitHandler !

VMExitHandler

שומרים את כל האוגרים הרלוונטים שאנו עלולים להשתמש בהם כדי לא להפריע להמשך הריצה של ה VM.
יוזמים קריאה ל MainVMExitHandler, ולאחר מכן משחזרים את מצב האוגרים וקוראים ל VMRESUME.

MainVMExitHandler

יכולים לחלץ מה VMCS את סיבת היציאה ולעשות עליה את הבדיקות שאנחנו מעוניינים (למשל EXIT_REASON_HLT).
פה אפשר לעשות עוד המון דברים מגניבים (כמובן שיש לשנות גם את הסביות ל VM-Exit מה VMCS בהתאמה)

ResumeToNextInstruction

בדרך כלל שמריצים VM באופן נורמטיבי והיה VM-Exit רוצים לדלג לפקודה הבאה ולא להמשיך להריץ שוב פעם את הפקודה הנוכחית (לא תמיד page-fault). על מנת לעשות זאת יש לשנות את GUEST_RIP ב VMCS לפקודה הבאה.

CHECKING VMCS LAYOUT

לפני שמריצים את ה VMLAUNCH מומלץ לבדוק שלא עשינו שגיאות בסיסיות באתחול ה VMCS, שכן זה לא כזה קל.
הנקודה הבעייתית היא שבמידה ויש כישלון כשהמעבד מקבל משהו לא צפוי מה VMCS לא מתקבלת שגיאה מפורטת של המקום המדויק ב VMCS אותו יש לתקן ולכן זה מקשה מאוד על פתרון הבעיה.
צירפתי פרויקט בשם VmcsAuditor שאמור לעזור לפתור את הבעיה - אין צורך להתעמק בו (סתם שתכירו את הבעיה).

כעת, הריצו ובדקו שלא מקבלים BSOD, ושמקבלים את ההודעות דיבוג המתאימות...
יש לנו VM שמריץ קוד שלא עושה כלום, זה כבר משהו ! (useless)