1、概述

包括两个部分

VMM

Guest software

VMX

VMX是对应的虚拟化操作的名称，有两种，root和non-root，正如想的那样，VMM通常在root，而Guest在non-root，从VMM进入non-root的VMX称为VM entries，而从non-root进入root则称为VM exits。

除了VM陷入的那些指令，其他没有任何指令或者比特位可以标志着他处于一个VMM中，哪怕是CPL0级别的指令。

VM软件的生命周期

VMXON->VMXLAUNCH->VMXRESUME->VMXOFF

在resume之后，就进入了Guest环境下，并通过一些VM exit的指令来VM exit，VMXOFF来离开虚拟机。

VMCS结构体

这是用来管理虚拟机的结构体，后面会讲到，是一个64bit的并且首地址4k对齐的结构体

对于该结构体，使用一个寄存器存放该指针，使用VMPTRST和VMPTRLD来读和写。

而使用VMREAD，VMWRITE和VMCLEAR来进行结构体的configure

如何判断是否支持VT-x

使用CPUID命令，然后传入的参数中，EAX设为1，返回的值放在ECX中，其中bit5如果为1，标识了支持VT-x。

Enable VT-x

通过在CR4中的bit13使能它，然后VMXON，如果CR4没有置位，那么会报一个例外。如果已经VMXON了，那么不能够再清除掉CR4中的bit13。

同样的过程，发生在结束VM，先VMXOFF，再CR4中的bit13清零。

除此之外，想要Enable VT-x，软件还需要分配一个4k对齐的页，用于作为VMXON的参数。软件不应当在该内存块的内容中写任何东西，在VMXON和VMXOFF之间，不应当用任何方式去访问该内存块。

一些其他标志位的约束

CR0的PE，NE和PG，以及CR4的13bit都需要置位。表明VT-x指令必须开启硬件分页机制。如果想要在裸板子上面不带内存管理运行，那么VMM必须支持这件事情。

2、VMCS结构体

VMCS结构体和那个64位的4k对齐的地址相联系起来。当使用VMPTRLD之后，那么对应的VMCS结构体就active了，该操作会将对应的那个加载到当前的VMCS指针里面去。

直到使用VMCLEAR把相同的这个地址清理掉之后。才算结束。

如果VMCLEAR清理的地址就是当前VMCS结构体，那么他就失效了。

同一时刻可能存在多个VMCS是active的，而current VMCS指针则是最近的那个VMPTRLD执行的，并且仍然active的那个。

如果不是active的，那么就没有current VMCS。

VMPTRST则将current VMCS（如果没有current VMCS则是0xFFFFFFFF FFFFFFF）存到某个存储位置中去。

VMCS结构体内容

包括6个字段（以下基本就是直接抄）

Guest State Area，当VM entry的时候加载，而exit的时候则保存

Host State Area，当VM exit的时候加载

VM execution control field，这部分定义了会导致VM exit的指令和情况等等

VM exit control field，控制VM exit

VM entry control field，控制VM entry

VM exit information field，这部分是只读的，记录了一些exit的原因等等的信息

Guest State Area

包括以下内容

CR0，CR3和CR4（每个各64位）

Debug寄存器（64位）

RSP，RIP，RFLAGS（每个64位）

段寄存器，包括CS，SS，DS，ES，FS，GS，LDTR和TR，每个寄存器保存如下内容

段选择子（16位）

基地址（64位），即便是架构中只有32位的段寄存器，也保存64位。

段限长（32位）

AR bytes（32位），具体每一位各不相同，低16位有效。

这部分其实是段寄存器的隐藏部分，但是之所以也要保存，是因为寄存器有可能跟GDT或者LDT描述符中的内容不一致。

GDTR和IDTR寄存器的部分fields：

基地址（64位）

段限长（32位）

还包括一个32位的描述中断状态的内容

Host State Area

这部分只有在VM exit，也就是进入VMM的时候才加载，但是entry的时候并不保存

包括以下内容