持一台虚拟机的I型管理程序。像所有的I型管理程序一样，它在裸机上运行。虚拟机在用户态以用户进程的身份运行，因此，它不允许执行敏感指令。虚拟机内运行着一个客户操作系统，该客户操作系统认为自己是运行在内核态的，但是实际上它是运行在用户态的。我们把这种状态称为*虚拟内核态*（virtual kernel mode）。虚拟机内还运行着用户进程，这些进程认为自己是运行在用户态的（事实上也正是如此）。

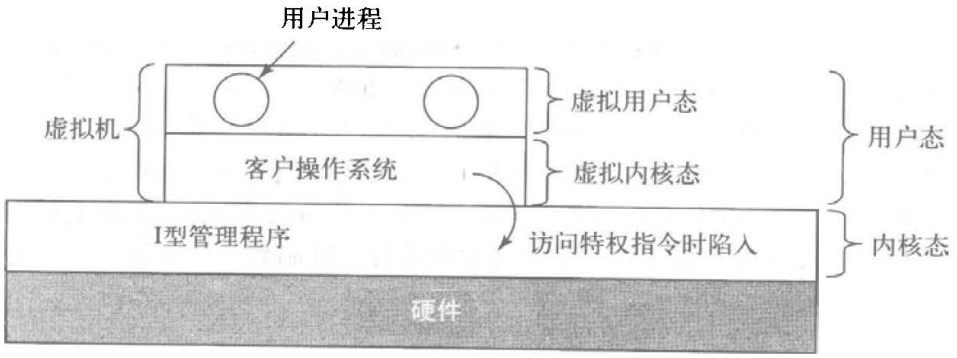

图8-26 当虚拟机当中的操作系统执行了一个内核指令时，如果支持虚拟化技术，那么它会陷入到管理程序

当操作系统（认为自己运行在内核态）执行一条敏感指令（只在内核态下可以执行）的时候会发生什么事情呢？在不支持VT技术的处理器上，指令失效并且操作系统通常情况下会崩溃。这意味着虚拟化是不可行的。有人争辩说所有在用户态执行的敏感指令都应该陷入，但那不是386和它的non-VT后代们的工作模式。

在支持VT技术的处理器上，当客户操作系统运行一条敏感指令时，发生到内核的陷入，如图8-26所示。管理程序分析指令，查看它是来自于虚拟机中的客户操作系统还是来自于虚拟机中的用户程序。如果是前一种情况，管理程序调度将要执行的指令；如果是后一种情况，它仿真面对运行在用户态的敏感指令时真实硬件的行为。如果虚拟机不支持VT技术，指令通常会被忽略；如果虚拟机支持VT技术，它陷入到虚拟机的客户操作系统中。

8.3.3 II型管理程序

当采用VT技术的时候，建立一个虚拟机系统相对比较直接，但是在VT技术出现之前，人们是怎么做的呢？很明显，在一台虚拟机上运行完整的操作系统是不可行的，因为（一些）敏感指令会被忽略掉，从而导致系统崩溃。于是人们发明了称为*II型管理程序*的替代品，如图1-29b所示。最早的一代产品是VMware（Adams 和 Agesen，2006；以及 Waldspurger，2002），它是斯坦福大学（Bugnion 等人，1997）DISCO研究项目的发展成果。VMware在Windows或Linux的宿主操作系统上作为普通用户程序运行。当它第一次运行的时候，它就像是一个新启动的计算机，试图在光驱中寻找含有操作系统的光盘。然后通过运行光盘上的安装程序，在它的*虚拟磁盘*（实际上就是Windows或Linux文件）上安装操作系统。一旦在虚拟磁盘上安装好了客户操作系统，虚拟机就可以运行了。

现在让我们来仔细研究VMware是如何工作的。当运行一个Pentium二进制文件的时候，这个二进制文件可能来自于安装光盘或虚拟磁盘，VMware首先浏览代码段以寻找*基本块*（basic block）。所谓基本块，是指以jump指令、call指令、trap指令或其他改变控制流的指令结束的可顺序运行的指令序列。根据定义，除了基本块的最后一条指令，基本块内不会含有其他改变程序计数器的指令。检查基本块是为了找出该基本块中是否含有敏感指令（见Popek和Goldberg的论述）。如果基本块中含有敏感指令，每条敏感指令被替换成处理相应情况的VMware过程调用。基本块的最后一条指令也被VMware的过程调用所替代。

上述操作完成之后，基本块在VMware中缓存并执行。在VMware中，不含任何敏感指令基本块的运行与它在裸机上的运行完全相同——因为它就是在裸机上运行的。通过这种方式找出、仿真敏感指令。这种技术称为*二进制翻译*（binary translation）。

基本块执行结束之后，控制返回到VMware，它会定位下一个基本块的位置。如果下一个基本块已经翻译完毕，它就可以被立刻执行。如果还没有翻译完毕，那么依次进行翻译、缓存、执行。最后，大多数程序被缓存并且接近全速的执行。很多优化方法得到了运用，例如，如果一个基本块跳转或调用另