有危害的代码，改变代码的顺序，交换寄存器，把某条指令用它的等价指令替换。变异引擎本身与病毒体一起也可以通过加密的方法隐藏起来。

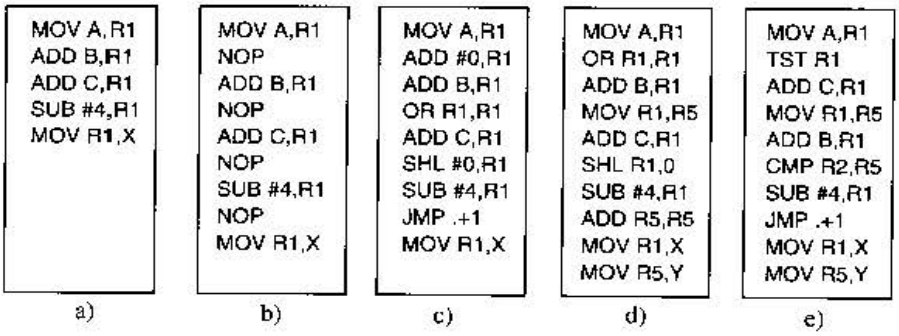

图9-33 多形态病毒的实例

要求较差的反病毒软件意识到图9-33a至图9-33e具有相同的代码功能是相当困难的，特别是当变异引擎有能力“狡兔三窟”时。反病毒软件可以分析病毒代码，了解病毒原理，甚至可以试图模拟代码操作，但我们必须记住有成千上万的病毒和成千上万的文件需要分析，所以每次测试不能花费太多的时间，否则运行起来会惊人地慢。

另外，储存在变量Y里的值是为了让人们难以发现与R5有关的代码是死码的事实，死码不会做任何事情。如果其他代码段对Y进行了读写，代码就会看上去十分合法。一个写得十分好的变异引擎代码会产生极强的变种，会给反病毒软件的作者带来噩梦般的麻烦。惟一让人安慰的是这样的引擎很难编写，所以Virgil的朋友都使用他的代码，结果在病毒界里并没有种类繁多的变异引擎。

到目前为止，我们讨论的是如何识别被感染的可执行文件里的病毒。而且，反病毒扫描器必须检查MBR、引导扇区、坏扇区列表、闪速ROM、CMOS等区域。但是如果有内存驻留病毒在运行会怎样呢？该内存驻留病毒不会被发现。更糟的是假设运行的病毒正在控制所有的系统调用，它就能轻易地探测到反病毒程序正在读引导扇区（用以查找病毒）。为了阻止反病毒程序，病毒进行系统调用，相反它把真正的引导区从坏扇区列表的藏身之地返回。它也可以作记录，在被扫描器检查以后会再次感染所有的文件。

为了防止被病毒欺骗，反病毒程序也可以会跳过操作系统直接去读物理磁盘。不过这样做需要具有用于IDE、SCSI和其他种类硬盘的内置设备驱动程序，这样会降低反病毒程序的可移植性，遇到不通用的硬盘就会一筹莫展。而且，跳过操作系统来读取引导扇区是可以的，但是跳过操作系统来读取所有的可执行文件却是不可能的，所以仍然存在病毒产生出与可执行文件相关的欺骗性数据的危险。

2. 完整性检查程序

另一种完全不同的病毒检测方法是实施完整性检查（integrity checking）。采用这种方法的反病毒程序首先扫描硬盘上的病毒，一旦确信硬盘是干净的，它就开始为每个可执行文件计算一个校验和。计算校验和的算法应该是很简单的，就像把程序段中的所有字作为32位或者64位整数加起来求和一样简单，但是这种算法也要像加密的散列算法一样，是不可能逆向求解的。然后，要把一个目录中的所有相关文件的校验和写到一个文件中去。下一次运行的时候，程序重新计算校验值，看是否与校验和文件里的值相匹配。这样被感染的文件会立刻被查出。

问题在于Virgil并不愿意让病毒被查出，他可以写一段病毒代码把校验和文件移走。更糟的是，他可以计算已感染病毒的文件校验值，用这一值替代校验和文件里的正常值。为了保护校验值不被更改，反病毒程序可以尝试把该文件藏起来，但对长时间研究反病毒程序的Virgil来说，这种方法也难以奏效。比较好的方法是对文件加密以便使得其上的破坏容易被发现。理想状态是加密采用了智能卡技术，加密密钥被放在芯片里使得程序无法读到。

3. 行为检查程序

第三种反病毒程序使用的方法是实施行为检查（behavioral checking）。通过这种方法，反病毒程序在系统运行时驻留在内存里，并自己捕捉所有的系统调用。这一方法能够监视所有的系统活动，并试图捕捉任何可能被怀疑的行为。例如，通常没有程序会覆盖引导扇区，所以有这种企图的程序几乎可以肯