作系统就把需要的程序从磁盘复制到内存中并执行；当进程运行结束后，操作系统在用户终端显示提示符并等待新的命令。收到新的命令后，它把新的程序装入内存，覆盖前一个程序。

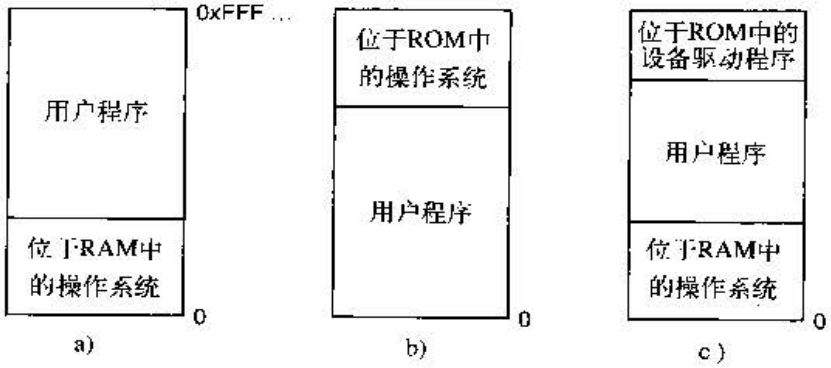

图3-1 在只有操作系统和一个用户进程的情形下，组织内存的三种简单方法（当然也存在其他方案）

在没有内存抽象的系统中实现并行的一种方法是使用多线程来编程。由于在引入线程时就假设一个进程中的所有线程对同一内存映像都可见，那么实现并行也就不是问题了。虽然这个想法行得通，但却没有被广泛使用，因为人们通常希望能够在同一时间运行没有关联的程序，而这正是线程抽象所不能提供的。更进一步地，一个没有内存抽象的系统也不大可能具有线程抽象的功能。

在不使用内存抽象的情况下运行多道程序

但是，即使没有内存抽象，同时运行多个程序也是可能的。操作系统只需要把当前内存中所有内容保存到磁盘文件中，然后把下一个程序读入到内存中再运行即可。只要在某一个时间内存中只有一个程序，那么就不会发生冲突。这样的交换概念会在下面讨论。

在特殊硬件的帮助下，即使没有交换功能，并发地运行多个程序也是可能的。IBM 360的早期模型是这样解决的：内存被划分为2KB的块，每个块被分配一个4位的保护键，保护键存储在CPU的特殊寄存器中。一个内存为1MB的机器只需要512个这样的4位寄存器，容量总共为256字节。PSW（Program Status Word，程序状态字）中存有一个4位码。一个运行中的进程如果访问保护键与其PSW码不同的内存，360的硬件会捕获到这一事件。因为只有操作系统可以修改保护键，这样就可以防止用户进程之间、用户进程和操作系统之间的互相干扰。

然而，这种解决方法有一个重要的缺陷。如图3-2所示，假设我们有两个程序，每个大小各为16KB，如图3-2a和图3-2b所示。前者加了阴影表示它和后者使用不同的内存键。第一个程序一开始就跳转到地

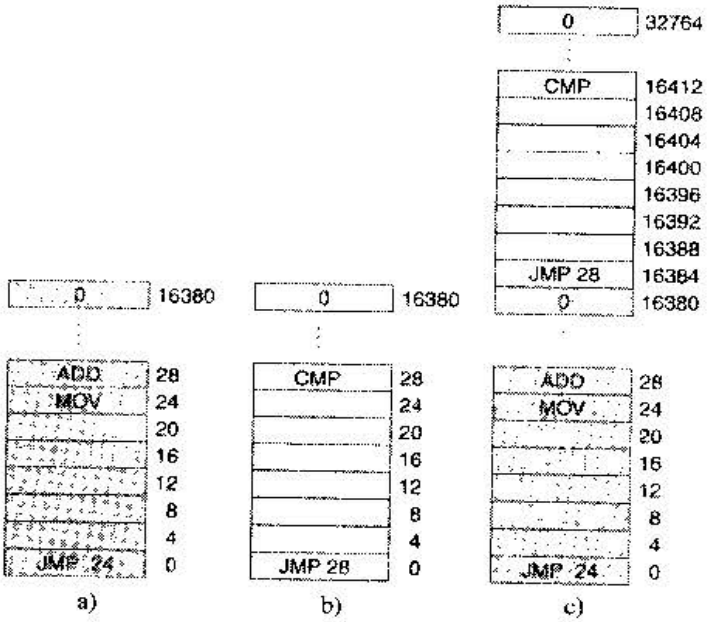

图3-2 重定位问题的说明：a) 一个16KB程序；b) 另一个16KB程序；c) 两个程序连续地装载到内存中