我们将在第5章讨论有关内容。

在有些计算机中，设备寄存器被映射到操作系统的地址空间（操作系统可使用的地址），这样，它们就可以像普通存储字一样读出和写入。在这种计算机中，不需要专门的I/O指令，用户程序可以被硬件阻挡在外，防止其接触这些存储器地址（例如，采用基址和界限寄存器）。在另外一些计算机中，设备寄存器被放入一个专门的I/O端口空间中，每个寄存器都有一个端口地址。在这些机器中，提供在内核态中可使用的专门IN和OUT指令，供设备驱动程序读写这些寄存器用。前一种方式不需要专门的I/O指令，但是占用了一些地址空间。后者不占用地址空间，但是需要专门的指令。这两种方式的应用都很广泛。

实现输入和输出的方式有三种。在最简单的方式中，用户程序发出一个系统调用，内核将其翻译成一个对应设备驱动程序的过程调用。然后设备驱动程序启动I/O并在一个连续不断的循环中检查该设备，看该设备是否完成了工作（一般有一些二进制位用来指示设备仍在忙碌中）。当I/O结束后，设备驱动程序把数据送到指定的地方（若有此需要），并返回。然后操作系统将控制返回给调用者。这种方式称为忙等待（busy waiting），其缺点是要占据CPU，CPU一直轮询设备直到对应的I/O操作完成。

第二种方式是设备驱动程序启动设备并且让该设备在操作完成时发出一个中断。设备驱动程序在这个时刻返回。操作系统接着在需要时阻塞调用者并安排其他工作进行。当设备驱动程序检测到该设备的操作完毕时，它发出一个中断通知操作完成。

在操作系统中，中断是非常重要的，所以需要更具体地讨论。在图1-11a中，有一个I/O的三步过程。在第1步，设备驱动程序通过写设备寄存器通知设备控制器做什么。然后，设备控制器启动该设备。当设备控制器传送完毕被告知的要进行读写的字节数量后，它在第2步中使用特定的总线发信号给中断控制器芯片。如果中断控制器已经准备接收中断（如果正忙于一个更高级的中断，也可能不接收），它会在CPU芯片的一个管脚上声明，这就是第3步。在第4步中，中断控制器把该设备的编号放到总线上，这样CPU可以读总线，并且知道哪个设备刚刚完成了操作（可能同时有许多设备在运行）。

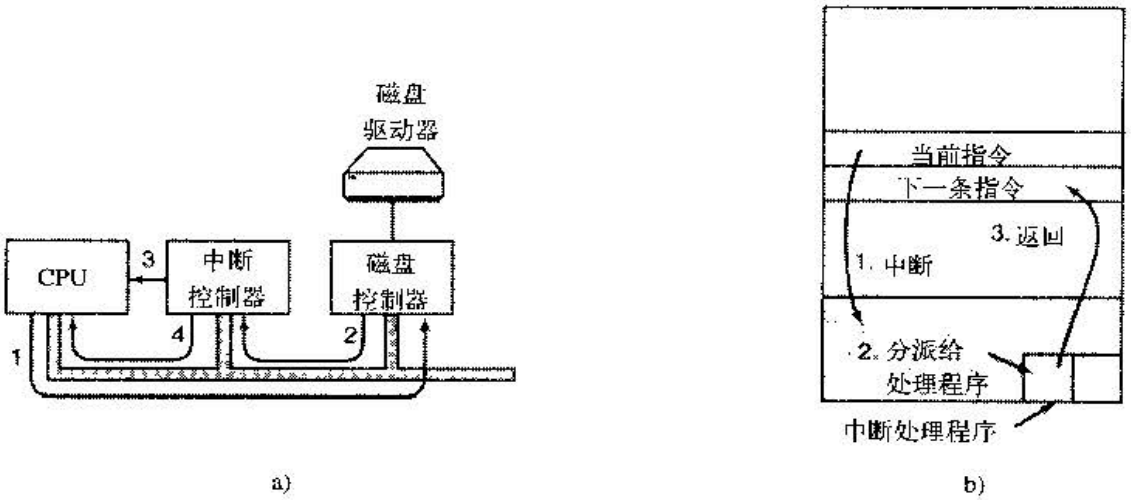

图1-11 a) 启动一个I/O设备并发出中断的过程；b) 中断处理过程包括取中断、运行中断处理程序和返回到用户程序

一旦CPU决定取中断，通常程序计数器和PSW就被压入当前堆栈中，并且CPU被切换到用户态。设备编号可以成为部分内存的一个引用，用于寻找该设备中断处理程序的地址。这部分内存称为中断向量（interrupt vector）。当中断处理程序（中断设备的设备驱动程序的一部分）开始后，它取走已入栈的程序计数器和PSW，并保存之，然后查询设备的状态。在中断处理程序全部完成之后，它返回到先前运行的用户程序中尚未执行的头一条指令。这些步骤如图1-11b所示。

第三种方式是，为I/O使用一种特殊的直接存储器访问（Direct Memory Access，DMA）芯片，它可以控制在内存和某些控制器之间的位流，而无须持续的CPU干预。CPU对DMA芯片进行设置，说明需要传送的字节数、有关的设备和内存地址以及操作方向，接着启动DMA。当DMA芯片完成时，它引发一个中断，其处理方式如前所述。有关DMA和I/O硬件会在第5章中具体讨论。