处于不同的完成状态，它们与程序计数器之间没有什么关系。

将机器留在一个明确状态的中断称为*精确中断*（precise interrupt）（Walker和Cragon，1995）。精确中断具有4个特性：

1) PC（程序计数器）保存在一个已知的地方。

2) PC所指向的指令之前的所有指令已经完全执行。

3) PC所指向的指令之后的所有指令都没有执行。

4) PC所指向的指令的执行状态是已知的。

注意，对于PC所指向的指令之后的那些指令来说，此处并没有禁止它们开始执行，而只是要求在中断发生之前必须撤销它们对寄存器或内存所做的任何修改。PC所指向的指令有可能已经执行了，也有可能还没有执行，然而，必须清楚适用的是哪种情况。通常，如果中断是一个I/O中断，那么指令就会还没有开始执行。然而，如果中断实际上是一个陷阱或者页面故障，那么PC一般指向导致错误的指令，所以它以后可以重新开始执行。图5-6a所示的情形描述了精确中断。程序计数器（316）之前的所有指令都已经完成了，而它之后的指令都还没有启动（或者已经回退以撤销它们的作用）。

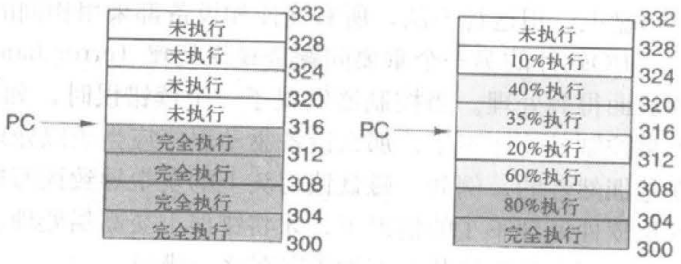

图5-6 a) 精确中断；b) 不精确中断

不满足这些要求的中断称为*不精确中断*（imprecise interrupt），不精确中断使操作系统编写者过得极为不愉快，现在操作系统编写者必须断定已经发生了什么以及还要发生什么。图5-6b描述了不精确中断，其中邻近程序计数器的不同指令处于不同的完成状态，老的指令不一定比新的指令完成得更多。具有不精确中断的机器通常将大量的内部状态“吐出”到堆栈中，从而使操作系统有可能判断出正在发生什么事情。重新启动机器所必需的代码通常极其复杂。此外，在每次中断发生时将大量的信息保存在内存中使得中断响应十分缓慢，而恢复则更加糟糕。这就导致具有讽刺意味的情形：由于缓慢的中断使得非常快速的超标量CPU有时并不适合实时工作。

有些计算机设计成某些种类的中断和陷阱是精确的，而其他的不是。例如，可以让I/O中断是精确的，而归因于致命编程错误的陷阱是不精确的，由于在被0除之后不需要尝试重新开始运行的进程，所以这样做也不错。有些计算机具有一个位，可以设置它强迫所有的中断都是精确的。设置这一位的不利之处是，它强迫CPU仔细地将正在做的一切事情记入日志并且维护寄存器的影子副本，这样才能够在任意时刻生成精确中断。所有这些开销都对性能具有较大的影响。

某些超标量计算机（例如Pentium系列）具有精确中断，从而使老的软件正确工作。为精确中断付出的代价是CPU内部极其复杂的中断逻辑，以便确保当中断控制器发出信号想要导致一个中断时，允许直到某一点之前的所有指令完成而不允许这一点之后的指令对机器状态产生任何重要的影响。此处付出的代价不是在时间上，而是在芯片面积和设计复杂性上。如果不是因为向后兼容的目的而要求精确中断的话，这一芯片面积就可以用于更大的片上高速缓存，从而使CPU的速度更快。另一方面，不精确中断使得操作系统更为复杂而且运行得更加缓慢，所以断定哪一种方法更好是十分困难的。

5.2 I/O软件原理

在讨论了I/O硬件之后，下面我们来看一看I/O软件。首先我们将看一看I/O软件的目标，然后从操作系统的观点来看一看I/O实现的不同方法。

5.2.1 I/O软件的目标

在设计I/O软件时一个关键的概念是*设备独立性*（device independence）。它的意思是应该能够编写出这样的程序：它可以访问任意I/O设备而无需事先指定设备。例如，读取一个文件作为输入的程序应该能够在硬盘、CD-ROM、DVD或者USB盘上读取文件，无需为每一种不同的设备修改程序。类似地，用户应该能够键入这样一条命令