中的每一条指令，并且使用硬件的每种功能。操作系统在内核态下运行，从而可以访问整个硬件。

相反，用户程序在用户态下运行，仅允许执行整个指令集的一个子集和访问所有功能的一个子集。一般而言，在用户态中有关I/O和内存保护的所有指令是禁止的。当然，将PSW中的模式位设置成内核态也是禁止的。

为了从操作系统中获得服务，用户程序必须使用*系统调用*（system call）系统调用陷入内核并调用操作系统。TRAP指令把用户态切换成内核态，并启用操作系统。当有关工作完成之后，在系统调用后面的指令把控制权返回给用户程序。在本章的后面我们将具体解释系统调用过程，但是在这里，请读者把它看成是一个特别的过程调用指令，该指令具有从用户态切换到内核态的特别能力。作为排印上的说明，我们在行文中使用小写的Helvetica字体，表示系统调用，比如read。

有必要指出，计算机使用陷阱而不是一条指令来执行系统调用。其他的多数陷阱是由硬件引起的，用于警告有异常情况发生，诸如试图被零除或浮点下溢等。在所有的情况下，操作系统都得到控制权并决定如何处理异常情况。有时，由于出错的原因程序不得不停止。在其他情况下可以忽略出错（如下溢数可以被置为零）。最后，若程序已经提前宣布它希望处理某类条件时，那么控制权还必须返回给该程序，让其处理相关的问题。

多线程和多核芯片

Moore 定律指出，芯片中晶体管的数量每18个月翻一番。这个“定律”并不是物理学上的某种规律，诸如动量守恒定律等，它是 Intel公司的共同创始人Gordon Moore对半导体公司如何能快速缩小晶体管能力上的一个观察结果。Moore 定律已经保持了30年，有希望至少再保持10年。

使用大量的晶体管引发了一个问题：如何处理它们呢？这里我们可以看到一种处理方式：具有多个功能部件的超标量体系结构。但是，随着晶体管数量的增加，再多晶体管也是可能的。一件由此而来的必然结果是，在CPU 芯片中加入了更大的缓存，人们肯定会这样做，然而，原先获得的有用效果将最终消失掉。

显然，下一步不仅是有多个功能部件，某些控制逻辑也会出现多个。Pentium 4和其他一些CPU芯片就是这样做的，称为*多线程*（multithreading）或*超线程*（hyperthreading，这是Intel公司给出的名称）。近似地说，多线程允许CPU保持两个不同的线程状态，然后在纳秒级的时间尺度内来回切换。（线程是一种轻量级进程，也即一个运行中的程序。我们将在第2章中具体讨论）。例如，如果某个进程需要从内存中读出一个字（需要花费多个时钟周期），多线程CPU则可以切换至另一个线程。多线程不提供真正的并行处理。在一个时刻只有一个进程在运行，但是线程的切换时间则减少到纳秒数量级。

多线程对操作系统而言是有意义的，因为每个线程在操作系统看来就像是单个的CPU。考虑一个实际有两个CPU的系统，每个CPU有两个线程。这样操作系统将把它看成是4个CPU。如果在某个时间的特定点上，只有能够维持两个CPU忙碌的工作量，那么在同一个CPU上调度两个线程，而让另一个CPU完全空转，就没有优势了。这种选择远远不如在每个CPU上运行一个线程的效率高。Pentium 4的后继者，Core（还有Core 2）的体系结构并不支持超线程，但是Intel公司已经宣布，Core的后继者会具有超线程能力。

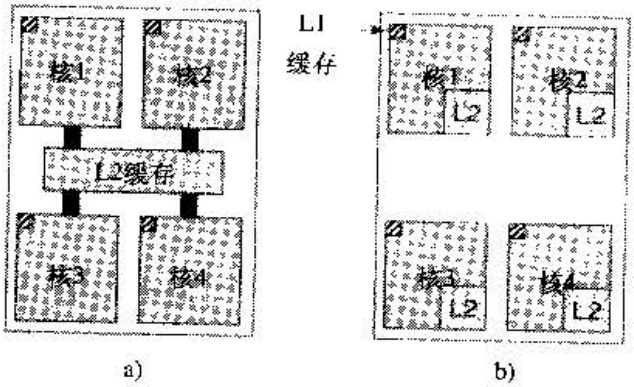

图1-8 a) 带有共享L2缓存的4核芯片；
b) 带有分离L2缓存的4核芯片

除了多线程，还出现了包含2个或4个完整处理器或内核的CPU芯片。图1-8中的多核芯片上有效地装有4个小芯片，每个小芯片都是一个独立的CPU。（后面将解释缓存。）要使用这类多核芯片肯定需要多处理器操作系统。

1.3.2 存储器

在任何一种计算机中的第二种主要部件都是存储器。在理想情形下，存储器应该极为迅速（快于执