1.3.1 处理器

计算机的“大脑”是CPU，它从内存中取出指令并执行之。在每个CPU基本周期中，首先从内存中取出指令，解码以确定其类型和操作数，接着执行之，然后取指、解码并执行下一条指令。按照这一方式，程序被执行完成。

每个CPU都有其一套可执行的专门指令集。所以，Pentium不能执行SPARC程序，而SPARC也不能执行Pentium程序。由于用来访问内存以得到指令或数据的时间要比执行指令花费的时间长得多，因此，所有的CPU内都有一些用来保存关键变量和临时数据的寄存器。这样，通常在指令集中提供一些指令，用以将一个字从内存调入寄存器，以及将一个字从寄存器存入内存。其他的指令可以把来自寄存器、内存的操作数组合，或者用两者产生一个结果，诸如将两个字相加并把结果存在寄存器或内存中。

除了用来保存变量和临时结果的通用寄存器之外，多数计算机还有一些对程序员可见的专门寄存器。其中之一是*程序计数器*，它保存了将要取出的下一条指令的内存地址。在指令取出之后，程序计数器就被更新以便指向后继的指令。

另一个寄存器是*堆栈指针*，它指向内存中当前栈的顶端。该栈含有已经进入但是还没有退出的每个过程的一个框架。在一个过程的堆栈框架中保存了有关的输入参数、局部变量以及那些没有保存在寄存器中的临时变量。

当然还有程序状态字（Program Status Word，PSW）寄存器。这个寄存器包含了条件码位（由比较指令设置）、CPU优先级、模式（用户态或内核态），以及各种其他控制位。用户程序通常读入整个PSW，但是，只对其中的少量字段写入。在系统调用和I/O中，PSW的作用很重要。

操作系统必须知晓所有的寄存器。在时间多路复用（time multiplexing）CPU中，操作系统经常会中止正在运行的某个程序并启动（或再启动）另一个程序。每次停止一个运行着的程序时，操作系统必须保存所有的寄存器，这样在稍后该程序被再次运行时，可以把这些寄存器重新装入。

为了改善性能，CPU设计师早就放弃了同时读取、解码和执行一条指令的简单模型。许多现代CPU具有同时取出多条指令的机制。例如，一个CPU可以有分开的取指单元、解码单元和执行单元，于是当它执行指令n时，它还可以对指令$n+1$解码，并且读取指令$n+2$。这样一种机制称为*流水线*（pipeline），在图1-7a中是一个有着三个阶段的流水线示意图。更长的流水线也是常见的。在多数的流水线设计中，一旦一条指令被取进流水线中，它就必须被执行完毕，即便前一条取出的指令是条件转移，它也必须被执行完毕。流水线使得编译器和操作系统的编写者很头疼，因为它造成了在机器中实现这些软件的复杂性问题。

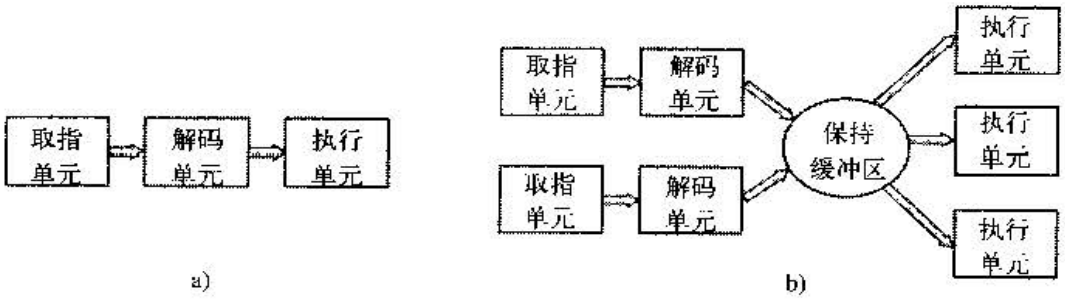

图1-7　a) 有三个阶段的流水线；b) 一个超标量CPU

比流水线更先进的设计是一种*超标量CPU*，如图1-7b所示。在这种设计中，有多个执行单元，例如，一个CPU用于整数算术运算，一个CPU用于浮点算术运算，而另一个用于布尔运算。两个或更多的指令被同时取出、解码并装入一个保持缓冲区中，直至它们执行完毕。只要有一个执行单元空闲，就检查保持缓冲区中是否还有可处理的指令，如果有，就把指令从缓冲区中移出并执行之。这种设计存在一种隐含的作用，即程序的指令经常不按顺序执行。在多数情况下，硬件负责保证这种运算的结果与顺序执行指令时的结果相同，但是，仍然有部分令人烦恼的复杂情形被强加给操作系统处理，我们在后面会讨论这种情况。

除了用在嵌入式系统中的非常简单的CPU之外，多数CPU都有两种模式，即前面已经提及的内核态和用户态。通常，在PSW中有一个二进制位控制这两种模式。当在内核态运行时，CPU可以执行指令集