旁注　RISC 和 CISC 指令集

x86-64 有时称为“复杂指令集计算机”(CISC，读作“sisk”)，与“精简指令集计算机”(RISC，读作“risk”)相对。从历史上看，先出现了 CISC 机器，它从最早的计算机演化而来。到 20 世纪 80 年代早期，随着机器设计者加入了很多新指令来支持高级任务(例如处理循环缓冲区，执行十进制数计算，以及求多项式的值)，大型机和小型机的指令集已经变得非常庞大了。最早的微处理器出现在 20 世纪 70 年代早期，因为当时的集成电路技术极大地制约了一块芯片上能实现些什么，所以它们的指令集非常有限。微处理器发展得很快，到 20 世纪 80 年代早期，大型机和小型机的指令集复杂度一直都在增加。x86 家族沿着这条道路发展到 IA32，最近是 x86-64。即使是 x86 系列也仍然在不断地变化，基于新出现的应用的需要，增加新的指令类。

20 世纪 80 年代早期，RISC 的设计理念是作为上述发展趋势的一种替代而发展起来的。IBM 的一组硬件和编译器专家受到 IBM 研究员 John Cocke 的很大影响，认为他们可以为更简单的指令集形式产生高效的代码。实际上，许多加到指令集中的高级指令很难被编译器产生，所以也很少被用到。一个较为简单的指令集可以用很少的硬件实现，能以高效的流水线结构组织起来，类似于本章后面描述的情况。直到多年以后 IBM 才将这个理念商品化，开发出了 Power 和 PowerPC ISA。

加州大学伯克利分校的 David Patterson 和斯坦福大学的 John Hennessy 进一步发展了 RISC 的概念。Patterson 将这种新的机器类型命名为 RISC，而将以前的那种称为 CISC，因为以前没有必要给一种几乎是通用的指令集格式起名字。

比较 CISC 和最初的 RISC 指令集，我们发现下面这些一般特性。

CISC	早期的 RISC
指令数量很多。Intel 描述全套指令的文档[51]有 1200 多页。	指令数量少得多。通常少于 100 个。
有些指令的延迟很长。包括将一个整块从内存的一个部分复制到另一部分的指令，以及其他一些将多个寄存器的值复制到内存或从内存复制到多个寄存器的指令。	没有较长延迟的指令。有些早期的 RISC 机器甚至没有整数乘法指令，要求编译器通过一系列加法来实现乘法。
编码是可变长度的。x86-64 的指令长度可以是 1～15 个字节。	编码是固定长度的。通常所有的指令都编码为 4 个字节。
指定操作数的方式很多样。在 x86-64 中，内存操作数指示符可以有许多不同的组合，这些组合由偏移量、基址和变址寄存器以及伸缩因子组成。	简单寻址方式。通常只有基址和偏移量寻址。
可以对内存和寄存器操作数进行算术和逻辑运算。	只能对寄存器操作数进行算术和逻辑运算。允许使用内存引用的只有 load 和 store 指令，load 是从内存读到寄存器，store 是从寄存器写到内存。这种方法被称为 load/store 体系结构。
对机器级程序来说实现细节是不可见的。ISA 提供了程序和如何执行程序之间的清晰的抽象。	对机器级程序来说实现细节是可见的。有些 RISC 机器禁止某些特殊的指令序列，而有些跳转要到下一条指令执行完了以后才会生效。编译器必须在这些约束条件下进行性能优化。
有条件码。作为指令执行的副产品，设置了一些特殊的标志位，可以用于条件分支检测。	没有条件码。相反，对条件检测来说，要用明确的测试指令，这些指令会将测试结果放在一个普通的寄存器中。
栈密集的过程链接。栈被用来存取过程参数和返回地址。	寄存器密集的过程链接。寄存器被用来存取过程参数和返回地址。因此有些过程能完全避免内存引用。通常处理器有更多的(最多的有 32 个)寄存器。