Y86-64 指令集既有 CISC 指令集的属性，也有 RISC 指令集的属性。和 CISC 一样，它有条件码、长度可变的指令，并用栈来保存返回地址。和 RISC 一样的是，它采用 load/store 体系结构和规则编码，通过寄存器来传递过程参数。Y86-64 指令集可以看成是采用 CISC 指令集(x86)，但又根据某些 RISC 的原理进行了简化。

旁注　RISC 与 CISC 之争

20 世纪 80 年代，计算机体系结构领域里关于 RISC 指令集和 CISC 指令集优缺点的争论十分激烈。RISC 的支持者声称在给定硬件数量的情况下，通过结合简约式指令集设计、高级编译器技术和流水线化的处理器实现，他们能够得到更强的计算能力。而 CISC 的拥趸反驳说要完成一个给定的任务只需要用较少的 CISC 指令，所以他们的机器能够获得更高的总体性能。

大多数公司都推出了 RISC 处理器系列产品，包括 Sun Microsystems(SPARC)、IBM 和 Motorola(PowerPC)，以及 Digital Equipment Corporation(Alpha)。一家英国公司 Acorn Computers Ltd. 提出了自己的体系结构——ARM(最开始是“Acorn RISC Machine”的首字母缩写)，广泛应用在嵌入式系统中(比如手机)。

20 世纪 90 年代早期，争论逐渐平息，因为事实已经很清楚了，无论是单纯的 RISC 还是单纯的 CISC 都不如结合两者思想精华的设计。RISC 机器发展进化的过程中，引入了更多的指令，而许多这样的指令都需要执行多个周期。今天的 RISC 机器的指令表中有几百条指令，几乎与“精简指令集机器”的名称不相符了。那种将实现细节暴露给机器级程序的思想已经被证明是目光短浅的。随着使用更加高级硬件结构的新处理器模型的开发，许多实现细节已经变得很落后了，但它们仍然是指令集的一部分。不过，作为 RISC 设计的核心的指令集仍然是非常适合在流水线化的机器上执行的。

比较新的 CISC 机器也利用了高性能流水线结构。就像我们将在 5.7 节中讨论的那样，它们读取 CISC 指令，并动态地翻译成比较简单的、像 RISC 那样的操作的序列。例如，一条将寄存器和内存相加的指令被翻译成三个操作：一个是读原始的内存值，一个是执行加法运算，第三就是将和写回内存。由于动态翻译通常可以在实际指令执行前进行，处理器仍然可以保持很高的执行速率。

除了技术因素以外，市场因素也在决定不同指令集是否成功中起了很重要的作用。通过保持与现有处理器的兼容性，Intel 以及 x86 使得从一代处理器迁移到下一代变得很容易。由于集成电路技术的进步，Intel 和其他 x86 处理器制造商能够克服原来 8086 指令集设计造成的低效率，使用 RISC 技术产生出与最好的 RISC 机器相当的性能。正如我们在第 3.1 节中看到的那样，IA32 发展演变到 x86-64 提供了一个机会，使得能够将 RISC 的一些特性结合到 x86 中。在桌面、便携计算机和基于服务器的计算领域里，x86 已经占据了完全的统治地位。

RISC 处理器在嵌入式处理器市场上表现得非常出色，嵌入式处理器负责控制移动电话、汽车刹车以及因特网电器等系统。在这些应用中，降低成本和功耗比保持后向兼容性更重要。就出售的处理器数量来说，这是个非常广阔而迅速成长着的市场。

4.1.4　Y86-64 异常

对 Y86-64 来说，程序员可见的状态(图 4-1)包括状态码 Stat，它描述程序执行的总体状态。这个代码可能的值如图 4-5 所示。代码值 1，命名为 AOK，表示程序执行正常，