务，因为产生的代码遵循比较规则的模式，而且我们可以做试验，让编译器产生许多不同程序的代码。本章提供了许多示例和大量的练习，来说明汇编语言和编译器的各个不同方面。精通细节是理解更深和更基本概念的先决条件。有人说："我理解了一般规则，不愿意劳神去学习细节！"他们实际上是在自欺欺人。花时间研究这些示例、完成练习并对照提供的答案来检查你的答案，是非常关键的。

我们的表述基于 x86-64，它是现在笔记本电脑和台式机中最常见处理器的机器语言，也是驱动大型数据中心和超级计算机的最常见处理器的机器语言。这种语言的历史悠久，开始于 Intel 公司 1978 年的第一个 16 位处理器，然后扩展为 32 位，最近又扩展到 64 位。一路以来，逐渐增加了很多特性，以更好地利用已有的半导体技术，以及满足市场需求。这些进步中很多是 Intel 自己驱动的，但它的对手 AMD(Advanced Micro Devices)也作出了重要的贡献。演化的结果是得到一个相当奇特的设计，有些特性只有从历史的观点来看才有意义，它还具有提供后向兼容性的特性，而现代编译器和操作系统早已不再使用这些特性。我们将关注 GCC 和 Linux 使用的那些特性，这样可以避免 x86-64 的大量复杂性和许多隐秘特性。

我们在技术讲解之前，先快速浏览 C 语言、汇编代码以及机器代码之间的关系。然后介绍 x86-64 的细节，从数据的表示和处理以及控制的实现开始。了解如何实现 C 语言中的控制结构，如 `if`、`while` 和 `switch` 语句。之后，我们会讲到过程的实现，包括程序如何维护一个运行栈来支持过程间数据和控制的传递，以及局部变量的存储。接着，我们会考虑在机器级如何实现像数组、结构和联合这样的数据结构。有了这些机器级编程的背景知识，我们会讨论内存访问越界的问题，以及系统容易遭受缓冲区溢出攻击的问题。在这一部分的结尾，我们会给出一些用 GDB 调试器检查机器级程序运行时行为的技巧。本章的最后展示了包含浮点数据和操作的代码的机器程序表示。

网络旁注 ASM:IA32　IA32 编程

IA32，x86-64 的 32 位前身，是 Intel 在 1985 年提出的。几十年来一直是 Intel 的机器语言之选。今天出售的大多数 x86 微处理器，以及这些机器上安装的大多数操作系统，都是为运行 x86-64 设计的。不过，它们也可以向后兼容执行 IA32 程序。所以，很多应用程序还是基于 IA32 的。除此之外，由于硬件或系统软件的限制，许多已有的系统不能够执行 x86-64。IA32 仍然是一种重要的机器语言。学习过 x86-64 会使你很容易地学会 IA32 机器语言。

计算机工业已经完成从 32 位到 64 位机器的过渡。32 位机器只能使用大概 4GB(2^{32}字节)的随机访问存储器。存储器价格急剧下降，而我们对计算的需求和数据的大小持续增加，超越这个限制既经济上可行又有技术上的需要。当前的 64 位机器能够使用多达 256TB(2^{48}字节)的内存空间，而且很容易就能扩展至 16EB(2^{64}字节)。虽然很难想象一台机器需要这么大的内存，但是回想 20 世纪 70 和 80 年代，当 32 位机器开始普及的时候，4GB 的内存看上去也是超级大的。

我们的表述集中于以现代操作系统为目标，编译 C 或类似编程语言时，生成的机器级程序类型。x86-64 有一些特性是为了支持遗留下来的微处理器早期编程风格，在此，我们不试图去描述这些特性，那时候大部分代码都是手工编写的，而程序员还在努力与 16 位机器允许的有限地址空间奋战。

3.1　历史观点

Intel 处理器系列俗称 x86，经历了一个长期的、不断进化的发展过程。开始时，它是第