汇编语言教程之七 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/137/2021_2022__E6_B1_87_E7_BC_96_E8_AF_AD_E8_c98_137775.htm 5.0 编译优化概述 优化是一件非常重要的事情。作为一个程序设计者，你肯定希望自己的程序既小又快。DOS时代的许多书中都提到，“某某编译器能够生成非常紧凑的代码”，换言之，编译器会为你把代码尽可能地缩减，如果你能够正确地使用它提供的功能的话。目前，Intel x86体系上流行的C/C 编译器，包括Intel C/C Compiler, GNU C/C Compiler，以及最新的Microsoft和Borland编译器，都能够提供非常紧凑的代码。正确地使用这些编译器，则可以得到性能足够好的代码。 但是，机器目前还不能像人那样做富于创造性的事情。因而，有些时候我们可能会不得不手工来做一些事情。 使用汇编语言优化代码是一件困难，而且技巧性很强的工作。很多编译器能够生成为处理器进行过特殊优化处理的代码，一旦进行修改，这些特殊优化可能就会被破坏而失效。因此，在你决定使用自己的汇编代码之前，一定要测试一下，到底是编译器生成的那段代码更好，还是你的更好。 本章中将讨论一些编译器在某些时候会做的事情(从某种意义上说，本章内容更像是计算机专业的基础课中《编译程序设计原理》、《计算机组成原理》、《计算机体系结构》课程中的相关内容)。本章的许多内容和汇编语言程序设计本身关系并不是很紧密，它们多数是在为使用汇编语言进行优化做准备。编译器确实做这些优化，但它并不总是这么做；此外，就编译器的设计本质来说，它确实没有义务这么做编译器做的是等义变换，而不是等效

变换。考虑下面的代码： // 程序段1 int gaussianSum(){ int i, j=0. for(i=0. i return j. } 好的，首先，绝大多数编译器恐怕不会自作主张地把它“ 篡改 ”为 // 程序段1(改进1) int gaussianSum(){ int i, j=0. for(i=1. i return j. } 多数（但确实不是全部）编译器也不会把它改为 // 程序段1(改进2) inline int gaussianSum(){ return 5050. } 这两个修改版本都不同于原先程序的语义。首先我们看到，让i从0开始是没有必要的，因为j=i时，i=0不会做任何有用的事情；然后是，实际上没有必要每一次都计算1 ... 100的和它可以被预先计算，并在需要的时候返回。 这个例子也许并不恰当(估计没人会写出最初版本那样的代码)，但这种实践在程序设计中确实可能出现。我们把改进2称为编译时表达式预先计算，而把改进1成为循环强度削减。 然而，一些新的编译器的确会进行这两种优化。不过别慌，看看下面的代码： // 程序段2 int GetFactorial(int k){ int i, j=1. if((k=10)) return -1. 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com