计算机语言技术系列丛书(二)

Brady PROGRAMMING LIBRARY

PC机汇编语言程序设计

ASSEMBLY LANGUAGE FOR THE PC

John Socha
Peter Norton

Disk Inside!

Disk includes all the code examples in the book plus more than 30 valuable, time-saving assembly routines for your C and C++ programs

希望

学苑出版社

计算机语言技术系列丛书(二)

Assembly Language For The PC

PC 机汇编语言程序设计

〔美〕John Socha　Peter Norton　著
章　含　朱建兴　闻　钟　译
吴　晓　刘　军　王　祥　校

学苑出版社

(京)新登字 151 号

内 容 简 介

本书是一本关于汇编语言程序设计的应用书籍。本书共分为四大部分和四个附录。第一部分讲解了机器语言以及相关基础知识；第二部分讲解了汇编语言基础以及如何用汇编语言编写汇编程序；第三部分讲解了 IBM PC 的 ROM BIOS 及其大量实例；第四部分在前面三部分的基础上用实例探讨了更深层次的问题，如在 C 和 C＋＋程序中使用汇编语言、保护模式和 Windows 程序设计等。

本书适用于从事计算机开发和程序设计的编程人员以及大专院校师生。

需要本书的用户，请直接与北京海淀 8721 信箱书刊部联系，邮政编码 100080，**电话** 2562329。

版 权 声 明

计算机语言技术系列丛书(二)

PC 机汇编语言程序设计

著　者：〔美〕John Socha　Peter Norton

译　者：章　含　朱建兴　闻　钟

校　对：吴　晓　刘　军　王　祥

责任编辑：甄国宪

出版发行：学苑出版社　**邮政编码**：100036

社　址：北京市海淀区万寿路西街 11 号

印　刷：施园印刷厂

开　本：787×1092　1/16

印　张：25.875　**字数**：593 千字

印　数：1～5000

版　次：1994 年 8 月第 1 版第 1 次

ISBN 7－5077－0905－1/TP.29

本册定价：45.00 元

学苑版图书印、装错误可随时退换

作者简介

John Socha 以他的著作和软件产品而闻名于 PC 工业界。在 IBM PC 的早期，他为现在已停刊的杂志 softalk 编写一个专栏，在此他发表了像 Scrn Save(第一个屏幕保存器) 和 Whereis (第一个搜索硬盘文件的程序) 这样的程序。当 softalk 停刊以后，John 致力于完成他的物理学博士学位并编写了 The Norton Commander，这本书成了很畅销的软件产品。他也是畅销书 PC—World DOS 5 Complete Handbook 的合作者以及 Learn Programming and Visual Basic 2 的作者。John 在 Wisconsin 长大，并在 Wisconsin 大学获得了电子工程的学士学位，在 Cornel 大学获得应用物理的硕士和博士学位。他现在在华盛顿州的 Korkland 经营一家称为 Socha Computing 的软件公司，该公司开发 Windows 和 Macintosh 实用程序软件。

软件开发的发起人 Peter Norton 是超过一打的技术书籍的作者。他的第一本书，就是很畅销、很流行的 Inside the IBM PC，于 1983 年出版，同年接着又编写了另一本书 Brady，这是一本非常受欢迎的 DOS 指南。

Mr. Norton 生于 1943 年，在华盛顿州的 Seattle 长大。他进入了 Oregon(俄勒冈)的 Portland 的 Reed 大学，主修物理和数学，他在 Berkeley 的 Califania 大学完成了他的学业，获得了数学学士学位。在越南战争期间，他服务于美国军队，给战地医生讲授急救药知识。在 Zen Buddhist 修道院呆了五年以后，他获得了神学教学证书。在他于 1982 年创立 Norton Utilities 软件程序之前，Mr. Norton 已在计算机工业领域上拥有了各种地位。接着他就建立了自己的公司：Peter Norton Computing，Inc. ，开发和推销他的软件。在过去的十年中，他已经为计算机杂志编写了许多很受欢迎的栏目。1990 年，他把自己的公司卖给了 Symantec 公司，从而致力于 Norton Family Foundation(Norton 家族基金会)，这是一个为洛杉机地区的艺术组织提供财政援助的基金会。目前他与他的妻子 Eileen 和他们的两个孩子 Diana 和 Michael 住在 Santa Monica，California。

目　录

第一部分　机器语言

第二部分　汇编程序

第三部分 IBM PC 的 ROM BIOS

第四部分　高级课题

第零章　简介

0.1　为什么学习汇编语言

学习汇编语言有许多理由。既便是仅用C/C++或Pascal这种高级语言的程序员也能从学习汇编语言中得到益处。

当我们在几年前着手编写本书的第一版本时，程序员和公司实际上都在用汇编语言编写整个程序。例如，2.0版本以前的Lotus 1－2－3完全是用汇编语言编写的。既便是现在，1－2－3也是用C语言重新编写的，而许多程序员把目光移向了C++。C++语言是一种功能强的语言，是从C语言派生而来的。那么，如果不用汇编语言编写大型程序，为什么要学习并使用汇编语言呢？

了解汇编语言会使读者成为更好的程序员，即使读者从未编写过任何汇编语言子例程，或程序代码。在C/C++或Pascal程序中的许多特性在微处理器及其设计中有其根。在学习汇编语言时，读者实际上在学习计算机的体系结构。这一观察会使读者更清楚地了解计算机语言的许多人工因素。例如，读者会看到为什么整数范围在－32,768～32,767之间，并了解有关字节、字和段的所有内容。

汇编语言程序是任意PC兼容机的核心。与所有其他程序设计语言相比，汇编语言是最小的公分母。它要比高级语言更能使读者了解机器。因此，学习汇编语言就意味着了解计算机内的80X86微处理器。

了解微处理器在Microsoft Windows中极为有用，在第三十三章中读者会看到这一点（第三十三章讨论了保护方式和Windows程序设计）。读者将掌握Windows程序设计的某些特性，而在其他书中是不会谈及这些特性的。这是因为通过汇编语言所学到的程序设计技术一般来说通过高级语言是学不到的。

有时，绝大多数程序员编写或修改某些汇编语言代码。对Microsoft Windows程序来说更是如此。用汇编语言编写程序的最一般的理由是因为速度。计算机，现代C/C++和Pascal编译器是非常快的；所编译的代码运行非常快。但是，有时，即便是已编译的C/C++或Pascal代码也不够快。对此，读者可经常编写少量的汇编语言代码以执行内部循环操作。

自从我们编写了本书的第一版本，编译器有了长足的进展。几年前，读者必须使用像Microsoft的Macro Assembler这样的汇编语言将汇编语言代码添加到程序中。但是，如今几乎所有汇编语言都支持直接插入式汇编语言。这种汇编语言用于将若干汇编语言代码行直接添加到程序中，在C/C++或Pascal代码之间。这更适用于在程序中使用少量的汇编语言代码。在第三十一章中，读者还会了解如何使用汇编。

要用汇编语言编写较大型代码还需程序员掌握一些内容。如果要编写DOS或Windows

或任意操作系统(除 Windows NT)的设备驱动器,则必须用汇编语言编写这种设备驱动器。例如,Windows 拥有一个功能极强的机制——虚拟设备驱动器(缩写为 VxD)。对于在 Windows 下运行的所有 Windows 和 DOS 程序,该驱动器用于更改设备在 Windows 内的工作方式。这些 VxD 必须用汇编语言编写。

0.2 我们使用的方法

像介绍汇编语言程序设计的多数书籍一样,本书向读者展示了如何使用 80X86 微处理器的指令。但是,我们要更进一步并涉及一些高级技术。当读者开始编写自己的程序时,读者会觉得这些高级技术没有什么价值。

到读者读完本书时,会知道如何编写大型程序以及如何在 C/C++和 Windows 程序中使用汇编语言。此外,读者还会掌握专业程序员用以使其工作更简单的许多技术。这些技术包括模块化设计和逐步求精法,会使程序设计速度快两倍或三倍,并有助于读者编写更具可读性和可靠性的程序。

特别是逐步求精法为编写复杂程序减少了大量工作。如果读者有从何开始的感觉,就会发现,逐步求精法是编写程序的一种简单而自然的方法。这有多么惬意!

我们还试图向读者说明注释如何有助于编写更佳的程序。注释写得好可以清楚地解释读者为什么这样做而不那样做。如果代码编写得好,代码要做些什么显而易见,但读者要做什么也许并不清楚。

本书并不是纯理论性的。我们还将建立 Dskpatch(表示 Disk Patch)程序,且有许多理由说明它是有用的。首先,读者会看到逐步求精法和模块化设计在实际程序中的应用,并有机会看到为什么这些技术这么有用。此外,Dskpatch 是一个通用的全屏幕磁盘扇区编辑器。在读完这本书后,读者还可继续全部或部分使用之。

0.3 本书编排

我们已选择一种方法教读者如何使用汇编语言。我们认为这会尽可能快地加快速度,而不必使读者面面俱到。用这种方法,读者就可以在必须学会使用一种汇编语言之前学到许多有关 80X86 指令的知识。我们还有选择地编辑许多有关 DOS 的举例程序,因为编写 DOS 程序要比编写相同的 Windows 程序容易得多。因此,如果读者要编写有关 Windows 的程序,可阅读第一和第二部分,之后跳到第三十一和第三十三章。

本书分为四部分。每部分都有不同的重点。有关微处理器或汇编语言的内容,读者可找到自己感兴趣的章节。

第一部分共七章,注重于 80X86 微处理器。读者会掌握有关位,字节或机器语言的知识。每章都有大量实例使用 Debug 程序。Debug 程序是 DOS 所拥有的,它用于在其运行 DOS 时查看 PC 中 80X86 微处理器。第一部分仅假设读者拥有程序设计语言的基本知识,且知道如何操作计算机。

第二部分从第八章到第十六章,讲解汇编语言以及如何用汇编语言编写程序。方法并不复杂。我们没有涉及汇编语言本身的方方面面,但我们注重于编写实用程序所需的一组汇编语言

命令。

我们将用汇编语言重写第一部分的某些程序，之后开始建立 Dskpatch，我们将循序渐进地建立该程序，以便读者学会如何用逐步求精法建立大型程序。我们还要涉及到像模块化设计这样的技术。这种设计方法有助于读者编写出清晰的程序。如上所述，这些技术去掉了一般与编写汇编语言有关的某些复杂因素，简化了程序设计过程。

第三部分包括第十七章到第二十八章。我们将注重于如何使用 PC 中更为先进的特性。这些特性包括移动光标和清屏。我们还将讨论调试大型汇编语言程序的技术。不用做很多的工作，汇编语言程序快速增长，很快就有两三页长(Dskpatch 会更长)。既便我们对长达几页的程序使用调试技术，读者也会发现它们对小程序也有用。

第四部分涉及许多高级论题。当读者开始编写实际程序时，会对这些论题很感兴趣。前两章详细讲解 COM 程序，内存和段。之后，有一章将涉及如何将非常快速的屏幕显示直接写入屏幕内存。接下来，有一章讲解如何编写在 C/C++程序中可以使用的汇编语言过程。读者将学会如何完全用汇编语言并以任意内存模式编写通用例程，还将学会如何使用直接插入式汇编。再下一章介绍 RAM 驻存程序，其中有一个 DISKLITE 程序。最后，还有一章谈及保护方式和 Windows 程序设计，向读者极富有兴趣地详细介绍了 Windows 在 80386 以上微处理器上面运行的方式。

0.4 要使用书中举例读者需要什么

要使用书中举例读者需要一些工具。对于第一部分中的所有举例，仅需要 DOS 的 Debug 程序。在第二部分中，需要一个简单的文本编辑器，如 DOS 5 以上版本中的 Edit 程序，以及一个汇编程序，如微软的 Macro Assembler (MASM)或 Borland 的 Turbo Assembler (TASM)。书中的所有这些举例在 MASM 和 TASM 的各种版本中测试过。不过，在第三十一和三十三章，就需要 MASM6(或以上)或 TASM 汇编举例。

最后，要编译第三十三章中的某些 C 举例，需要 Quick C for Windows，Borland C++或 Microsoft C 6.0 以上版本。所有这些 C 编译器支持直接插入式汇编并用于编译 Windows 程序。

0.5 Dskpatch

在我们使用汇编语言时，我们将直接查看磁盘扇区并显示由 DOS 以十六进制方式存储的字符和数字。Dskpatch 是一个针对磁盘的全屏幕编辑器，并用于更改磁盘扇区中的这些字符和数字。例如，通过用 Dskpatch，读者可以查看 DOS 将磁盘目录存储在何处，并能更改文件名或其他信息。这样做了解 DOS 如何将信息存储在磁盘上的一种好方法。

Dskpatch 只不过是一道程序。它含有大约 50 个子例程。其中许多子例程是通用子例程序。当读者编写自己的程序时，就会发现这些子例程是些非常有用的子例程。因此，本书不仅是一本介绍 80X86 和汇编语言程序设计的书，且包含了大量的子程序。

第一部分

机器语言

第一章　学习调试和计算机算术操作

在本章中，读者将学会计算机如何处理算术操作。这是汇编语言的基本概念。它会直接影响到你用 C、汇编语言或任意其他计算机语言编写的所有程序。

主题

- Intel 微处理机介绍
- 计算机计数方式
- 用十六进制数计数
- 使用 Debug 程序
- 进行十六进制算术操作
- 位、字节、字和二进制表示
- 二进制补码——负数的奇数排序
- 小结

在这一章，读者将开始接触计算机内的微处理器，并将用 Debug 程序开始进行计算机算术操作。这是一个好的开端，因为 Debug 是几乎所有汇编语言程序的核心。在本书的所有章节中，我们将继续使用 Debug 程序，用以编写和运行非常简单的程序。

1.1　Intel 微处理机简介

在我们讲解汇编语言之前，先介绍几个有关微处理机的词汇。在 1992 年，有 4 个主处理机用在 PC 兼容机中：8088、80286、80386 和 80486 微处理机（而 Intel 着眼于下一代——Pentium）。在原 IBM PC 中第一次使用了 8088 微处理机，而 8088 微处理机的功能是最低的。计算机很少用 8088 微处理机。

在 IBM NT 中使用 80286。它要比 8088 快 4 倍，且是第一个可以运行 IBM OS/2 的计算机。80286 也是运行 Windows 3.1 所需的最小微处理机。绝大多数计算机目前都装配运行更快的 80386 或 80486 微处理器。它们还拥有 Windows 可以使用的附加功能。

80286、80386 和 80486 是 8088 微处理机的超集。也就是说，为 8088 微处理机编写的任何程序都可在任意其他微处理机上运行。几乎所有为 MS－DOS 或 PC－DOS（除 Windows 外）编写的程序都是使用 8088 特性编写的，所以它们可以在所有 MS－DOS 计算机上运行。在本书中，我们将涉及几乎所有 8088 指令，这样人们所编写的程序都可在所有 MS－DOS 计算机上运行。不过，在本书后面，我们将谈及如何用 80286 以上的微处理机的保护方式进行 Windows