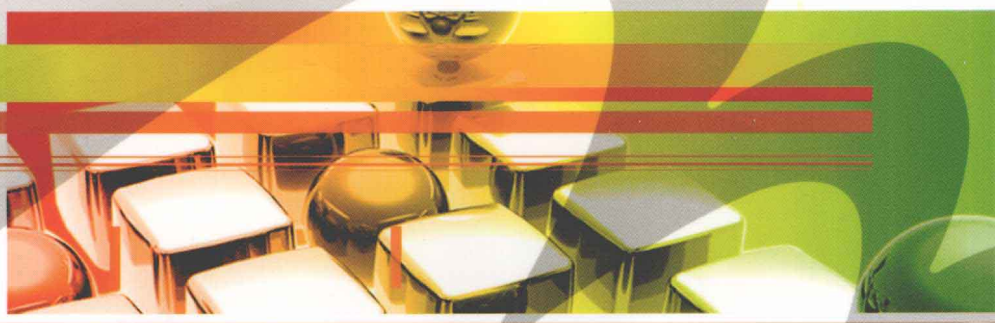
普通高等教育“十一五”国家级规划教材

教育部－微软精品课程教学成果

汇编语言程序设计

（第4版）

◆ 钱晓捷 主编



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http: //www.phei.com.cn

普通高等教育“十一五”国家级规划教材
教育部－微软精品课程教学成果

汇编语言程序设计

（第4版）

钱晓捷　主编

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书是普通高等教育“十一五”国家级规划教材，是教育部－微软精品课程教学成果。本书以 Intel 80x86 指令系统和 MASM 6.x 为主体，共 10 章，分为基础和提高两部分。前 5 章作为基础部分，以当前“汇编语言程序设计”课程的教学为目标，讲解 16 位基本整数指令及其汇编语言程序设计的知识，包括：汇编语言程序设计基础知识，8086 指令详解，MASM 伪指令和操作符，程序格式，程序结构及其设计方法。后 5 章为提高部分，介绍汇编语言程序设计的深入内容和实际应用知识，包括：32 位 80x86 CPU 的整数指令系统及其编程，汇编语言与 C/C ++ 混合编程，80x87 FPU 浮点指令系统及其编程，多媒体扩展指令系统及其编程，64 位指令简介。

本书可作为高等院校“汇编语言程序设计”课程的教材或参考书。本书内容广博、语言浅显、结构清晰、实例丰富，也适合电子信息、自动控制等专业的高校学生和成教学生、计算机应用开发人员、深入学习微机应用技术的普通读者阅读。

图书在版编目(CIP)数据

汇编语言程序设计/钱晓捷主编. —4 版. —北京：电子工业出版社，2012.6
ISBN 978-7-121-17013-3

Ⅰ.①汇… Ⅱ.①钱… Ⅲ.①汇编语言－程序设计－高等学校－教材 Ⅳ.①TP313

中国版本图书馆 CIP 数据核字(2012)第 093818 号

策划编辑：章海涛
责任编辑：章海涛　王　钰　　特约编辑：曹剑锋
印　　刷：
装　　订：三河市双峰印刷装订有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1 092　1/16　印张：20.75　字数：560 千字
印　　次：2012 年 6 月第 1 次印刷
定　　价：39.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010)88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：(010)88258888。

第 4 版前言

本书以 Intel 80x86 指令系统和 MASM 6. x 为主体，在个人微机的 MS-DOS 和 Windows 操作系统平台上，依据循序渐进的原则，以浅显的语言、清晰的结构、丰富的实例，全面而系统地介绍整数指令、浮点指令、多媒体指令的汇编语言程序设计方法，以及汇编语言的模块化开发、32 位 Windows 应用程序的编写、与 C/C ++ 混合编程等高级技术。全书共 10 章，分为前 5 章的基础部分和后 5 章的提高部分。

第 1 章总结性地给出进行汇编语言程序设计所需要学习的基本知识，包括微型计算机系统的软件和硬件组成、数据表示，并重点展开 8086 通用寄存器和寻址方式。

第 2 章详尽地讲述 8086 微处理器整数指令的功能和使用，并引导读者正确书写每条指令、理解程序段的功能，以及编写常见问题的汇编语言程序。

第 3 章结合汇编语言源程序格式，引出程序开发、语句格式、常用伪指令和操作符、段定义等内容。

第 4 章以程序设计技术为主线，结合大量的程序实例详述顺序、分支、循环、子程序结构的汇编语言程序设计方法。

第 5 章介绍汇编语言的高级程序设计技术，包括 MASM 的高级语言特性、宏结构程序、模块化程序设计和 I/O 程序设计。

第 6 章首先将 16 位指令及编程扩展到 32 位环境，其次介绍新增整数指令及其应用，最后重点展开利用汇编语言编写 32 位 Windows 控制台和窗口应用程序的开发环境和基本方法。

第 7 章讨论汇编语言与高级语言 Turbo C 和 Visual C ++ 的混合编程，并说明如何利用汇编语言优化 C ++ 代码，以及利用 Visual C ++ 集成环境开发汇编语言程序的方法。

第 8 章介绍 Intel 80x87 浮点数据格式、浮点指令及其程序设计方法。

第 9 章介绍 MMX、SSE、SSE2 和 SSE3 多媒体指令及其编程方法。

第 10 章简介 Intel 64 结构的 64 位指令。

每章最后配有相当数量的习题，既可以作为课后作业，也可以作为上机练习。

附录内容包括：详解 DEBUG 调试程序，可用于配合前 5 章，尤其是第 1 章和第 2 章寻址方式、指令功能、程序片段的学习；介绍 CodeView 调试程序的使用方法，用于第 3 章以后进行源程序级的程序调试；提供汇编程序 MASM 6. 11 的伪指令、Intel 80x86 整数指令和常见汇编错误列表；与本教材配套的输入/输出子程序库的使用说明。

本书特点

本书自 2000 年出版以来获得广大师生读者好评，第 3 版（书名为《新版汇编语言程序设计》）被评为普通高等教育“十一五”国家级规划教材。主编以此书为教材主持的“汇编语言程序设计”课程为教育部 - 微软精品课程和河南省精品课程（网址：http://jpkc. zzu. edu. cn/hbyycai/）。总结教学经验和改革思路，结合师生反馈，我们在前 3 版基础上，编写了本书，并保持了原来的诸多特点。

1. 知识全面

本书的编写参照国内高校和自学考试“汇编语言程序设计”课程的本、专科教学大纲，兼顾相关专业教学要求，既满足当前教学需求，又面向今后改革方向。

本书全面讲解 80x86 指令系统及编程，除完整的 8086 指令外，还包括 32 位指令、浮点指令、多

媒体指令。本书不仅介绍基本的汇编语言程序设计知识,还介绍高级汇编语言程序设计技术,如开发大型程序需要的模块化方法、实际应用当中的混合编程实用技术、32 位 Windows 应用程序编写等。

本书采用微软的 MASM 6.15 汇编程序,以简化段定义源程序格式为主,兼顾完整段定义源程序格式,涉及 DOS 和 Windows 操作系统平台的汇编语言程序设计。

2. 教材实用

本书示例中的指令、程序段和完整的源程序都经过验证,能够运行通过。本书经过 3 个版本、11 年的使用时间,已经发现并纠正了绝大多数错误。

本书采用浅显、明晰、循序渐进的描述方法,具有前后对照、贯穿始终的风格,加上清晰的结构、丰富的示例,使得本书既适合课堂教学,又适合读者自学。

配合本书,作者制作有精美的多媒体电子教案(PowerPoint 演示文档),为教师利用现代化教学手段提供方便。同时,作者还维护"大学微机技术系列课程教学辅助网站"(钱晓捷之微辅网 http://www2.zzu.edu.cn/qwfw)。网站主要就"汇编语言程序设计"和"微机原理及接口技术"课程提供有关教学课件、教材勘误、疑难解答、补充资料等辅助资源,是本书的动态延伸。微辅网还设置有教学论坛,让学生在课外学习中获得帮助。

3. 突出实践

本书特别强调上机实践,不仅在正文中引导读者通过调试程序或者带输出结果的源程序理解指令、程序功能,各章还配有丰富的习题和上机练习题;附录介绍调试程序的使用方法、调试指令和程序的步骤。

本书的结构安排适合尽早上机实践,并将实践过程贯穿始终。第 1 章引出 MASM 开发软件包,可用于熟悉命令行 MS-DOS 基本操作(基于 Windows),第 2 章充分利用调试程序学习指令功能和调试程序段,第 3 章引出完整的源程序格式,并给出程序开发方法。后续章节通过大量程序强化编程开发,还介绍了开发 Windows 应用程序的 MASM32 环境、利用 Visual C ++ 开发调试汇编语言程序的方法。

写给教师

"汇编语言程序设计"的教学内容在高校相关学科的教学中有两种处理方式。一种作为独立的课程,这主要是用在计算机专业的本科、专科教学中;而在电子、通信、自动控制等专业则将汇编语言作为主体融入微机原理课程。本书内容自成一个相对完整的知识体系,适合作为独立课程的教材,也可以作为微机原理课程的参考书和补充教材。

传统上,"汇编语言程序设计"课程在 MS-DOS 操作系统平台上,采用微软的 MASM 汇编程序,介绍 16 位 8086 微处理器指令系统的程序设计。当前,有些院校已将汇编语言的教学重点转向 32 位 IA-32 微处理器指令系统,操作系统平台也过渡为 Windows。同时,也出现在 Linux 环境下的汇编语言课程。针对不同层次、不同专业的学习,"汇编语言程序设计"课程教学进行了多方面的改革,形成了多元化的教学方案。

本书从简单的 16 位汇编语言入手,使其满足当前教学要求(包括与"微型机原理"或"微机接口技术"等课程的配合);在 32 位汇编语言展开许多深入的内容,使其面向以后的应用需求。本书在组织教学内容上,体现了许多新的理念。例如,没有从纯软件角度介绍汇编语言,教学的重点是硬指令而不是伪指令,强调程序设计不是程序格式,引出实用技术但淡化具体应用,通过程序实例和上机实践掌握程序设计方法,而不是通过大量细节的描述讲解程序设计。

汇编语言的基本语句是处理器指令。对于 Intel 80x86 系列微处理器,由于其属于复杂指令集计算机 CISC,再加上几十年发展的历史沉淀,因此它的指令系统非常庞大和复杂。虽然作为本书的一个特色,介绍了 Pentium 4 及之前处理器的所有指令,但教材的重点教学内容却是常用的简单指令,全书的实例程序也主要采用各种处理器指令系统所共有的基本指令编写。作为教师,尤其应该注意这个问题,否则许多学生会面对繁杂的指令望而却步、失去进一步学习的兴趣。

各种高级语言程序设计的教学中，调试程序及程序的调试方法往往被忽略或回避，但作为低级程序语言的汇编语言不应避而不谈。在汇编语言的教学过程中，可以利用调试程序的单步执行和断点执行能力，直观地理解指令和程序的功能，进而掌握程序的动态调试和排错。对于专科层次或程度较低的学生，掌握调试程序本身就是一个似乎不可逾越的难关。一方面，教师可以通过多媒体教学手段，演示调试程序的使用；本书也配套制作了相关演示动画、记录了指令调试和程序开发过程（可以通过本书的教学网站下载）；学生通过上机实践学习调试程序。另一方面，本书自编了一个显示输出和键盘输入的 I/O 子程序库，教师和学生可以直接使用其中的子程序来编写具有显示结果的源程序，同时还可以配合列表文件，暂时避开调试程序这个难点。这个 I/O 子程序库还可以作为一个教学案例，用于组织学习中有余力的学生围绕输入输出子程序库进行项目开发。

对于使用过本书前续版本的教师，新版教材在主体教学内容上保持了兼容，仍然遵循由浅入深、循序渐进的原则：先 16 位 8086 指令系统、基本汇编语言编程技术，然后介绍 32 位指令编程，并将汇编语言知识加深，从混合编程、浮点编程、多媒体编程各个角度展开。

主要的修编内容如下：

（1）基于教学实际，调整了部分内容的位置，并进行了修编（第 1 章标志的详解调整到第 2 章加减运算指令前，第 1 章 MASM 编程环境使用合并到第 3 章汇编语言程序的开发过程中，第 2 章输入/输出指令调整到第 5 章输入/输出程序设计前，第 2 章 DOS 功能调用调整到第 3 章汇编语言程序的开发过程后，第 2 章串操作类指令调整到第 4 章循环程序设计后）。

（2）基于教学需要，扩充了部分内容（第 3 章汇编语言程序的开发过程，附录 A 调试程序 DEBUG，附录 B 调试程序 CodeView 的调试示例，附录 E 常见汇编错误信息）。

（3）基于技术发展，改写了部分内容（第 1 章 Intel 80x86 系列微处理器，第 7 章使用 Visual C ++ 开发汇编语言程序）。

（4）基于内容修编，调整了部分习题。另外，还对部分知识增加图表、文字等进行更详细的讲解，改正前版中的个别错误等。

有鉴于此，教师在教学过程中，应该选择基本的、常用的指令，减少 16 位汇编语言部分的教学内容和教学时间。这样，就有时间展开对 32 位汇编语言的深入内容和实用技术的学习。

写给学生

学习汇编语言到底有什么用途？这是许多学生首先要提问的。

在计算机科学与技术的知识体系当中，“汇编语言程序设计”课程的教学内容属于计算机系统结构的一个方面。汇编语言配合“计算机组成原理”和“微机原理及接口技术”等相关课程，帮助学生从软件角度理解计算机工作原理；同时，为“操作系统”、“编译原理”等课程提供必须的基础知识，也是自动控制等与硬件相关应用领域的程序设计基础。“汇编语言程序设计”课程是继“高级语言程序设计”之后的又一门计算机语言程序设计课程，但它是一种低级语言程序设计。通过汇编语言的学习，学生能够比较全面地了解程序设计语言，利于更深入地学习和应用高级语言。

随着高级语言的发展、可视化开发工具的应用，汇编语言往往被应用程序开发人员所忽略，其应用领域也逐渐萎缩。但是，作为一个面向机器的程序设计语言，汇编语言具有直接有效控制硬件的能力，能够编写出运行速度快、代码量小的高效程序，在许多场合具有不可替代的作用，例如，操作系统的核心程序段、实时控制系统的软件、智能仪器仪表的控制程序、频繁调用的子程序或动态连接库、加密解密软件、分析和防治计算机病毒，等等。

学习什么样的汇编语言呢？这是许多学生感到困惑的。

汇编语言与处理器指令系统相关，不同的处理器指令系统具有不同的汇编语言。但是，作为一个底层开发语言，它还是有许多共性的。从指令系统的典型性、实用性、编程环境以及教学内容连续性等多方面考虑，Intel 80x86 指令系统作为“汇编语言程序设计”课程的主要教学内容具有显而易见的优势，理所当然地成为计算机及相关学科的首选。

日常工作和学习中广泛使用的个人微机(PC)采用 Intel 80x86 或与之兼容的微处理器。个人微机过去使用 DOS 操作系统,现在主要使用 Windows 或 Linux 操作系统,但后者具有 DOS 模拟环境。由于 DOS 操作系统平台比较简单,对程序员限制少,是一个相对理想的教学环境,所以本书也就利用这个平台展开汇编语言,选用 DOS 和 Windows 平台下最常用的微软 MASM 汇编程序。

许多学生总感觉 16 位指令、8086 微处理器、DOS 操作系统都是很"古老"和"陈旧"的内容,但它们实际上却是 32 位指令、Pentium 4 微处理器、Windows 或 Linux 操作系统的基础,都是最基本的知识。它们已经足够复杂、完全能够满足教学要求。当然,本书在提高部分也对使用 32 位指令编写 Windows 应用程序进行了详细讲解。因为有了 16 位汇编语言基础,这部分提高内容也就比较容易掌握了。如果直接从 32 位指令和 Windows 平台入门,往往需要大家学习很多其他内容才能够真正进入汇编语言的教学。国内外以所谓 32 位保护方式展开 Windows 汇编语言教学的教材,实际上是利用了 Windows 控制台环境。而 Windows 控制台环境的操作和外观与 DOS 操作系统一样,其基本教学内容几乎相同,所不同的仅仅是调用操作系统功能的方法。介绍用汇编语言编写 Windows 窗口应用程序的教材,都要求读者初步掌握汇编语言程序设计,并具有调用 Windows 应用程序接口 API 的编程经验(往往需要学习 Visual C ++ 之后才能够达到这个要求)。本书从基础知识入手,囊括上述所有教学内容。相信读者阅读本书后,会对此有深刻的体会。

怎样才能学好汇编语言呢?这又是许多学生深感无助的。

其实,学习汇编语言程序设计对先修知识要求不高。学生如果具备微机软件的操作能力,尤其是 DOS 环境和常用命令的使用,可以更好地完成上机实践(本书补充有这方面的知识,并构造一个 MASM 开发环境)。学生的高级语言(如 C/C ++ 或 Visual Basic)程序设计知识或经验,将有助于更好地理解程序结构和混合编程。程序设计属于软件方面的内容,但由于汇编语言与硬件相关的特点,学生在计算机或微型机原理方面的知识对于深刻体会指令功能大有益处。"汇编语言程序设计"课程的相关课程是"微机原理与接口技术",后者是微机的硬件知识,进一步加强了汇编语言在输入/输出和中断等方面的应用。两者共同为学生建立微型计算机的完整知识体系。

循序渐进的学习是对任何课程都有效的。不以循序渐进方式进行学习,往往会浪费宝贵的时间去盲目探索,最终学习到的内容可能是相对零散的知识,不能建立完整的、系统的知识结构。所以,建议学生遵循循序渐进的方法进行学习。首先,学生应理解每条常用指令的功能,能够正确书写每条指令;其次,学生通过阅读示例程序,掌握常见功能程序段的编写;再次,学生利用伪指令将程序段扩展成完整的源程序文件;随后,学生就各种程序结构编写常见问题的程序;最后,学生才编写较大型程序和有一定难度的程序。

对程序设计类课程,没有上机编程的实践是无法真正掌握的。所以,希望学生加强实践环节。学生应完成基本的上机指导编程要求,同时争取多进行编程实践,因为只有通过实际编程才能发现程序设计中的许多问题。不要轻视调试程序的作用,它是深入理解指令功能和程序执行过程的关键。请不要直接拷贝源程序代码,在一条一条语句的录入编辑过程中,就是书写正确语句、加深语句理解的绝好机会。

本书由钱晓捷主编,眷红英、穆玲玲、邱保志参与了前两版的编写和修订,关国利、张青、张行进也给予了帮助。作者对十多年来合作的同事深表谢意,大家共同的努力成就了本书、创建了精品课程。作者还要特别感谢使用本书的教师、学生和读者,是你们宝贵的意见建议和一贯支持催生了本版教材。作者非常欢迎大家通过本书的教学辅助网站(http://www2.zzu.edu.cn/qwfw)、华信教育资源网(http://www.hxedu.com.cn)或主编的电子邮件(qianxiaojie@zzu.edu.cn)进行教学交流,共同提高"汇编语言程序设计"课程的教学质量。

作 者

反侵权盗版声明

目　录

第 1 章　汇编语言基础知识

程序设计语言是开发软件的工具，它的发展经历了由低级语言到高级语言的过程。汇编语言是一种面向机器的低级程序设计语言。

汇编语言以助记符形式表示每条计算机指令，每条指令对应着计算机硬件的一个具体操作。利用汇编语言编写的程序与计算机硬件密切相关，程序员可直接对处理器内的寄存器、主存储器的存储单元及外设的端口等进行操作，从而能够有效地控制硬件。使用汇编语言编写的程序具有执行速度快、占用存储空间小的特点，这是高级语言无法替代的优势。所以，作为一个计算机专业人员，应该熟悉汇编语言，并掌握其程序设计方法。

本章介绍使用汇编语言进行程序设计所需要了解的基本知识，并引出有关基本概念。首先，介绍计算机系统的一般知识，包括计算机系统的硬件、软件及程序设计语言的发展，特别说明了汇编语言的特点和应用场合。其次，简述计算机中数据的表示，如二进制数和十六进制数、BCD 码和 ASCII 码、补码和反码、二进制数运算等。然后，认识 Intel 80x86 系列微处理器和以其为核心的微机系统。最后，展开 8086 微处理器的结构和数据寻址，作为第 2 章指令系统学习的重要基础。

1.1　计算机系统概述

计算机系统分为硬件和软件两大部分。硬件(Hardware)是计算机系统的机器部分，它是计算机工作的物质基础；软件(Software)则是为了运行、管理和维护计算机而编制的各种程序的总和，广义的软件还应该包括与程序有关的文档。利用汇编语言所编写的程序是软件，但是，每条汇编语言指令都使计算机某个具体硬件部件产生相应的动作，因此，利用汇编语言进行程序设计体现了计算机硬件和软件的结合。

1.1.1　计算机的硬件

源于冯·诺依曼设计思想的计算机由五大部件组成：控制器、运算器、存储器、输入设备和输出设备。控制器是整个计算机的控制核心；运算器是对信息进行运算处理的部件；存储器是用来存放数据和程序的部件；输入设备将数据和程序变换成计算机内部所能识别和接受的信息方式，并把它们送入存储器中；输出设备将计算机处理的结果以人们能接受的或其他机器能接受的形式送出。

现代计算机在很多方面都对冯·诺依曼计算机结构进行了改进，如五大部件演变为三个硬件子系统：处理器、存储系统和输入/输出系统。处理器(Processor)也称为中央处理单元(Central Processing Unit，CPU)，包括运算器和控制器，是计算机的核心部件。微型计算机中的处理器常采用一块大规模集成电路芯片构成，称为微处理器(Microprocessor)，它代表着整个微型计算机系统的性能。存储系统由处理器内的寄存器(Register)、高速缓冲存储器、主存储器和辅助存储器构成层次结构。处理器和存储系统在信息处理中起主要作用，是计算机硬件的主体部分，通常被称为“主机”。输入(Input)设备和输出(Output)设备统称为外部设备(Peripheral)，简称为外设或 I/O 设备；输入/输出系统的主体是外设，但还包括外设与主机之间相互连接的 I/O 接口电路。

为简化各部件的相互连接，现代计算机广泛应用总线结构，如图 1.1 所示。采用系统总线连接各功能部件使得计算机系统具有了组合灵活、扩展方便的特点。

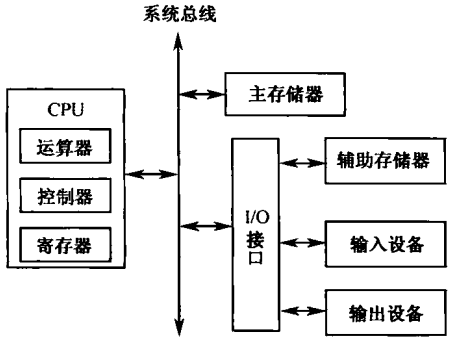


图 1.1　微型计算机的系统组成

1. 处理器

处理器除包括运算器和控制器外，还有一些高速存储单元，即寄存器，它们提供各种操作需要的数据。高性能处理器内部非常复杂，如为了提高存取主存储器的速度而增加了高速缓冲存储器（Cache），运算器中不仅有基本的整数运算器，还有浮点处理单元甚至多媒体数据运算单元，控制器还包括存储管理单元、代码保护机制等。

2. 存储器

存储器（Memory）是计算机的记忆部件，用来存放程序及所涉及的数据。存储器的内容并不因为被读出而消失，可以重复取出；只有存入新的信息后，原来的信息才会被更改。

按所起作用，存储器可以分为主存储器和辅助存储器，即主存（内存）和辅存（外存）。主存储器存放当前正在执行的程序和使用的数据，可以直接被 CPU 存取，由半导体存储器芯片构成，其成本高、容量小，但速度快。辅助存储器可用于长期保存大量程序和数据，CPU 需要通过 I/O 接口访问，由磁盘或光盘构成，成本低、容量大，但速度较慢。在计算机系统中，主存的容量是有限的，需要辅存来补充；辅存的容量比主存大得多，但存取速率比主存慢得多。一般来说，程序和数据以文件的形式保存在辅存中，只有在用到它们时，才从辅存读到主存的某个区域中，由中央处理单元控制执行。

按读/写能力，存储器可以分为随机存取存储器 RAM 和只读存储器 ROM。CPU 可以从 RAM 读出信息，也可以向 RAM 写入信息，所以 RAM 也被称为读/写存储器；而 ROM 中的信息只能被读出，不能被修改。RAM 型半导体存储器可以按地址随机读/写，但断电后不能保存信息；ROM 型半导体存储器通常只能被读出，在断电后仍能保存信息。磁盘存储器是可读可写的，而 CD-ROM 光盘是只读的，这两种存储器都是辅存，都可以长期保存数据，但它们的存取速率都慢于半导体存储器。

主存储器由大量存储单元组成。为了区别每个单元，我们将它们编号，于是每个存储单元就有了一个唯一的存储器地址（Address）。每个存储单元存放 1 字节的数据。1 字节（byte，B）包含 8 个二进制位（bit，b）。

存储容量是指存储器所具有的存储单元个数，其基本单位是字节（B）。为了表达更大的容量，常用的单位有 KB（千字节）、MB（兆字节）、GB（吉字节）甚至 TB（太字节）。$1KB = 2^{10}B = 1024B$，$1MB = 2^{20}B$，$1GB = 2^{30}B$，$1TB = 2^{40}B$。

3. 外部设备

外部设备是实现人机交互和机间通信的一些机电设备。在微机系统中，常用的输入设备有键盘、鼠标器等，输出设备有显示器、打印机等；起辅存作用的磁盘和光盘也是以外设的形式连接到系统中的，可以认为它们既是输入设备也是输出设备。

由于外设的种类繁多、工作原理各异，所以每个外设必须通过输入/输出接口（I/O 接口）电路与系统连接。程序员所见的 I/O 接口由一组寄存器组成。为了区别它们，对每个寄存器进行了编号，形成 I/O 地址，通常被称为 I/O 端口（Port）。系统实际上就是通过这些端口与外设进行通信的。

4. 系统总线

总线（Bus）是用于多个部件相互连接、传递信息的公共通道，物理上就是一组公用导线。例如，处理器芯片的对外引脚（Pin）常被称为处理器总线。这里的系统总线（System Bus）是指计算机系统中主要的总线，如处理器与存储器和 I/O 设备进行信息交换的公共通道。

对汇编语言程序员来说，处理器、存储器和外部设备依次被抽象为寄存器、存储器地址和输

入/输出地址，因为编程过程中将只能通过寄存器和地址实现处理器控制、存储器和外设的数据存取与处理等操作。

1.1.2 计算机的软件

软件是计算机系统的重要组成部分，可以使机器更好地发挥其功能。软件可分为系统软件和应用软件。

1. 系统软件

系统软件是指为了方便使用、维护和管理计算机系统而编制的一类软件及其文档，包括操作系统、语言翻译程序等。系统软件是面向计算机系统的、通常由计算机厂家提供的程序及其文档，是用户使用计算机时为产生、准备和执行用户程序所需的程序。

在系统软件中，最重要的软件是操作系统。操作系统负责管理整个系统的软件、硬件资源，向用户提供交互的界面，为所有其他程序的运行打下基础。用户借助操作系统使用计算机系统，程序员也要采用操作系统提供的驱动程序编写用户程序。

程序员采用某种程序设计语言编写源程序，利用语言翻译程序将源程序转变成可运行的程序。例如，本书介绍用汇编语言设计源程序的方法，但必须利用“汇编程序”完成源程序的翻译工作。高级语言则采用编译类或解释类程序来完成这个工作。

2. 应用软件

应用软件是解决某一问题的程序及其文档，覆盖了计算机应用的所有方面，每个应用都有相应的应用程序。

微机系统具有多种多样的应用软件。例如，进行程序设计时要采用文本编辑软件编写源程序，带有丰富格式的字处理软件帮助用户书写文章，排版软件则用于书刊出版。

大型的程序设计项目往往要借助软件开发工具（包）。这个开发工具是进行程序设计所用到的各种软件的有机集合，所以也被称为集成开发环境，包括文本编辑器、语言翻译程序、用于形成可执行文件的连接程序，以及进行程序排错的调试程序等。

1.1.3 计算机的程序设计语言

程序设计语言有很多，可以分为低级语言和高级语言。低级语言有机器语言和汇编语言，高级语言有 C/C ++ 、Java、BASIC 等。

1. 机器语言

计算机能够直接识别的是二进制数 0 和 1 组成的代码。机器指令（Instruction）就是用二进制编码的命令，一条机器指令控制计算机完成一个操作。每种处理器都有各自的机器指令集，某处理器支持的所有指令的集合就是该处理器的指令集（Instruction Set）。指令集及使用它们编写程序的规则被称为机器语言（Machine Language）。

用机器语言形成的程序是计算机唯一能够直接识别并执行的程序，用其他语言编写的程序必须经过翻译，转换成机器语言程序，所以机器语言程序常称为目标程序（或目的程序）。

机器指令一般由操作码（Opcode）和操作数（Operand）构成。操作码表明处理器要进行的操作，操作数表明参加操作的数据对象。一条机器指令是一组二进制代码，一个机器语言程序就是一段二进制代码序列。因为二进制数表达比较烦琐，常用对应的十六进制数形式表达。例如，完成两个数据 100 和 256 相加的功能，在 8086 CPU 上用十六进制数表达的代码序列如下：

```
B8 64 00
05 00 01
```

几乎没有人能够直接读懂该程序段的功能，因为机器语言看起来就是毫无意义的一串代码。用机器语言编写程序的最大缺点是，难以理解，极易出错，也难以发现错误。所以，只是在计算机发展的早期或不得已的情况下，才用机器语言编写程序。现在，除了有时在程序某处需要直接采用机器指令填充外，几乎没有人采用机器语言编写程序了。

2. 汇编语言

为了克服机器语言的缺点，人们采用便于记忆并能描述指令功能的符号来表示机器指令。表示指令操作码的符号称为指令助记符，简称助记符(Memonic)，一般是表明指令功能的英语单词或其缩写。指令操作数同样可以用易于记忆的符号表示。

用助记符表示的指令就是汇编格式指令。汇编格式指令及使用它们编写程序的规则形成汇编语言(Assembly Language)。用汇编语言书写的程序就是汇编语言程序，或称为汇编语言源程序。例如，实现 100 与 256 相加的 MASM 汇编语言程序段表达如下：

```
mov ax,100          ;取得一个数据 100(MOV 是传送指令)
add ax,256          ;实现 100 +256(ADD 是加法指令)
```

这时候，如果熟悉了有关助记符及对应指令的功能，就可以读懂上述程序段了。

汇编语言是一种符号语言，用助记符表示操作码，比机器语言容易理解和掌握，也容易调试和维护。但是，汇编语言源程序要翻译成机器语言程序才可以由处理器执行。这个翻译的过程称为“汇编”，完成汇编工作的程序就是汇编程序(Assembler)。

3. 高级语言

汇编语言虽然较机器语言直观一些，但仍然烦琐难记。于是在 20 世纪 50 年代，人们研制出了高级程序设计语言(High-level Programming Language)，简称高级语言。高级语言比较接近于人类自然语言的语法习惯及数学表达形式，它与具体的计算机硬件无关，更容易被广大计算机工作者掌握和使用。利用高级语言，即使一般的计算机用户也可以编写软件，而不必懂得计算机的结构和工作原理。当然，用高级语言编写的源程序不会被机器直接执行，而需经过编译或解释程序的翻译才可变为机器语言程序。

广泛应用的高级语言有十多种，如简单易用的 BASIC 语言、算法语言 FORTRAN、结构化语言 Pascal、系统程序语言 C/C ++ 等。用高级语言表达 100 与 256 相加，就是通常的数学表达形式“100 +256”。

4. 汇编语言程序设计的意义

高级语言简单、易学，而汇编语言复杂、难懂，是否就没有必要采用汇编语言了呢？让我们首先比较一下汇编语言和高级语言的特点。

- 汇编语言与处理器密切相关。每种处理器都有自己的指令系统，相应的汇编语言各不相同，所以汇编语言程序的通用性、可移植性较差。相对来说，高级语言与具体计算机无关，高级语言程序可以在多种计算机上编译后执行。
- 汇编语言功能有限，又涉及寄存器、主存单元等硬件细节，所以编写程序比较烦琐，调试起来也比较困难。高级语言提供了强大的功能，采用类似自然语言的语法，所以容易被掌握和应用，也不必关心诸如标志、堆栈等琐碎问题。
- 汇编语言本质上就是机器语言，可以直接、有效地控制计算机硬件，因而容易产生运行速度快、指令序列短小的高效率目标程序。高级语言不易直接控制计算机的各种操作，编译程序产生的目标程序往往比较庞大、程序难以优化，所以运行速度较慢。

可见，汇编语言的主要优点就是可以直接控制计算机硬件部件，可以编写在“时间”和“空间”两方面最有效的程序。这些优点使得汇编语言在程序设计中占有重要的位置，是不可被取代的。

汇编语言的缺点也是明显的。它与处理器密切相关，要求程序员比较熟悉计算机硬件系统、考虑许多细节问题，导致编写程序烦琐，调试、维护、交流和移植困难。因此，有时可以采用高级语言和汇编语言混合编程的方法，取长补短，更好地解决实际问题。

汇编语言主要应用场合有以下几方面。

- 程序要具有较快的执行时间，或者只能占用较小的存储空间，如操作系统的核心程序段、实时控制系统的软件、智能仪器仪表的控制程序等。
- 程序与计算机硬件密切相关，程序要直接、有效地控制硬件，如 I/O 接口电路的初始化程序段、外部设备的低层驱动程序等。
- 大型软件需要提高性能、优化处理的部分，如计算机系统频繁调用的子程序、动态链接库等。
- 没有合适的高级语言或只能采用汇编语言的时候，如开发最新的处理器程序时、暂时没有支持新指令的编译程序。
- 汇编语言还有许多实际应用，如分析具体系统尤其是该系统的低层软件、加密解密软件、分析和防治计算机病毒等。

事实上，汇编语言被称为低层程序设计语言（Low-level Programming Language）更合适。因为程序设计语言是按照计算机系统的层次结构区分的，本没有“高低贵贱”之分，只是某种语言更适合某种应用层面（或说场合）而已。

1.2 数据表示

计算机只能识别 0/1 编码，进入计算机的任何信息都要转换成 0/1 编码。本节说明计算机如何利用 0/1 表示现实当中的各种数值、字符等内容。

1.2.1 数制

人们已经习惯了用十进制数计数，但计算机则以二进制数的形式表达数值，为了便于表示二进制数，人们又常用到十六进制数。

1. 二进制数

计算机中为便于存储及计算的物理实现，采用二进制数。二进制数的特点为逢二进一，由 0、1 两个数码组成，基数为 2，各位的位权以 2^k 表示。

二进制数 $a_n a_{n-1}\cdots a_1 a_0 . b_1 b_2 \cdots b_m$ 可表示为

$$a_n \times 2^n + a_{n-1} \times 2^{n-1} + \cdots + a_1 \times 2^1 + a_0 \times 2^0 + b_1 \times 2^{-1} + b_2 \times 2^{-2} + \cdots + b_m \times 2^{-m}$$

其中，a_i、b_j 非 0 即 1。

2. 十六进制数

由于二进制数书写较长、难以记忆，因此常用易于与二进制数转换的十六进制数来描述二进制数。十六进制数的基数是 16，共 16 个数码：0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F（也可以使用小写字母 a ~ f），逢十六进位，各位的位权为 16^k。

十六进制数 $a_n a_{n-1}\cdots a_1 a_0 . b_1 b_2 \cdots b_m$ 可表示为

$$a_n \times 16^n + a_{n-1} \times 16^{n-1} + \cdots + a_1 \times 16^1 + a_0 \times 16^0 + b_1 \times 16^{-1} + b_2 \times 16^{-2} + \cdots + b_m \times 16^{-m}$$

其中，a_i、b_j 为 0 ~ F 中的一个数码。

汇编语言中通常用字母 B 或 b 结尾表示一个数据采用二进制（Binary），用字母 H 或 h 结尾表示采用十六进制（Hexadecimal）。十进制（Decimal）数据可以用字母 D 或 d 结尾，以示强调或区别，也可以不加结尾字母。在高级语言 C/C ++ 中，表示十六进制数要添加前缀 0x。

3. 数制之间的转换

十进制数的整数部分转换为二进制数和十六进制数可用除法，把要转换的十进制数的整数部分不断除以二进制数和十六进制数的基数2或16，并记下余数，直到商为0为止。由最后一个余数开始逆向取各余数，则为该十进制数整数部分转换成的二进制数和十六进制数。

例1.1 十进制整数转换为二进制数和十六进制数。

126D = 01111110B = 7EH

十进制数的小数部分转换为二进制数和十六进制数则分别乘以各自的基数，记录整数部分，直到小数部分为0为止或者选取一定的位数。

例1.2 十进制小数转换为二进制数和十六进制数。

0.8125D = 0.1101B = 0.DH

二进制数、十六进制数转换为十进制数，可分别套用各自的公式。

例1.3 二进制数和十六进制数转换为十进制数。

$0011.1010B = 1\times2^{1}+1\times2^{0}+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}=3.625D$

$1.2H = 1\times16^{0}+2\times16^{-1}=1.125D$

二进制数和十六进制数之间具有对应关系：每4个二进制位对应1个十六进制位，如表1.1所示，所以相互转换非常简单。

表1.1 不同进制间与BCD码的对应关系

十进制数	二进制数	十六进制数	BCD码
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	8	8
9	1001	9	9
10	1010	A	
11	1011	B	
12	1100	C	
13	1101	D	
14	1110	E	
15	1111	F	

例1.4 二进制数和十六进制数相互转换。

00111010B = 3AH

F2H = 11110010B

1.2.2 编码

在计算机中，各种字符只能用若干位的二进制码的组合表示，这就称为二进制编码。由于字节为计算机的基本存储单位，所以常用8个二进制位表达1个字符。

1. BCD码

1位十进制数可以用4位二进制编码来表示，这就是所谓的“二进制编码的十进制数(Binary Coded Decimal，BCD)”。常用的BCD码是8421 BCD码，用4位二进制编码的低10个数码表示0～9这10个数字，见表1.1。

BCD码是比较直观的，只要熟悉了BCD的10个编码，可以很容易地实现十进制数与BCD码