普通高等教育“十一五”国家级规划教材

普通高等学校计算机专业特色教材

汇编语言程序设计

（第2版）

王成端 主编

高等教育出版社
Higher Education Press

普通高等教育“十一五”国家级规划教材

普通高等学校计算机专业特色教材

汇编语言程序设计

（第2版）

主　编　王成端
副主编　王宇晓　王　丰　李晓波

高等教育出版社

内容提要

本书为《汇编语言程序设计》第2版，为适应微机技术发展和高校教学改革的需要，本书将部分章节内容重新合理组合，内容的选取、概念的引入、文字的叙述、例题和习题的选取等方面力求做到精益求精、循序渐进、结构清晰。

本书以 Intel 8086/8088 系列微机作为基础机型全面、系统地介绍了汇编语言程序的设计方法。内容主要包括：基础知识、寻址方式和指令系统、汇编语言、基本程序设计、子程序设计、数值与非数值程序设计、输入/输出程序设计、中断与系统功能调用、模块化程序设计以及汇编语言程序设计上机指导。本书内容充实、重点突出，内容编排突出了汇编语言程序设计的一般方法。从第4章开始每章后面附有一定数量的实训项目，实训项目具有很强的典型性、实用性。学生可一边学习，一边上机操作，便于在实践中巩固所学的理论知识。

本书可作为应用性本科院校计算机科学与技术专业、自动化、机械设计制造及其自动化等专业的教材，也可作为工程技术人员的参考用书。

图书在版编目(CIP)数据

汇编语言程序设计/王成端主编. —2版. —北京：
高等教育出版社，2008.6
ISBN 978-7-04-023965-2

Ⅰ.汇… Ⅱ.王… Ⅲ.汇编语言-程序设计-
高等学校-教材 Ⅳ.TP313

中国版本图书馆 CIP 数据核字(2008)第056678号

策划编辑	刘 艳	**责任编辑**	彭立辉	**封面设计**	张志奇	**责任绘图**	黄建英
版式设计	张 岚	**责任校对**	王 超	**责任印制**	朱学忠		

出版发行	高等教育出版社	**购书热线**	010-58581118
社　　址	北京市西城区德外大街4号	**免费咨询**	800-810-0598
邮政编码	100120	**网　　址**	http://www.hep.edu.cn
总　　机	010-58581000		http://www.hep.com.cn
		网上订购	http://www.landraco.com
			http://www.landraco.com.cn
经　　销	蓝色畅想图书发行有限公司	**畅想教育**	http://www.widedu.com
印　　刷	北京新丰印刷厂		
		版　　次	2003年8月第1版
开　　本	787×1092　1/16		2008年6月第2版
印　　张	19.5	**印　　次**	2008年6月第1次印刷
字　　数	440 000	**定　　价**	24.50元

物料号　23965-00

第 2 版前言

汇编语言程序设计是计算机及相关专业的一门重要的专业基础课，为将来学习操作系统、微机原理与接口技术、计算机组成原理、计算机控制、计算机体系结构等课程打好基础。在众多的程序设计语言中，汇编语言属于低级语言。这里所谓的低级，是指汇编语言相对于高级语言而言，高级语言面向用户，而汇编语言则面向机器。汇编语言可以直接利用硬件资源进行编程，用于编制对时间和空间要求很高的程序。它适用于编制主机和外部设备之间的接口程序，适用于编制计算机控制系统、仪器仪表和家用电器等的应用程序。另外，在为各类实际应用设计专用系统中，出于种种需要须改写操作系统中的部分操作，使之成为专用操作系统或嵌入式操作系统，此时也需要对汇编语言与高级语言进行混合编程。

本书第 1 版于 2003 年出版以来，得到了普遍应用。但是，随着计算机技术的迅猛发展，为了更好地为读者服务，在第 1 版的基础上进行了较大幅度的修订，出版第 2 版《汇编语言程序设计》教材。

在《汇编语言程序设计》第 2 版教材中，对第 1 章结构做了适当调整，并根据教学经验增加了对溢出和进位的比较讲解，概括性地介绍 80x86/Pentium 微处理器以及安腾处理器，其中重点讲解 80386 寄存器组，增加 80386 结构图和 Pentium 微处理器结构图。对每一种微处理器的介绍都在原有基础上做了适当的调整和扩充。第 2 章主要是根据教学过程中学生的理解方式，对寻址方式和 8086/8088 指令系统的示意图进行修改，这些图比上一版更清晰地表示了各种寻址方式和指令执行过程，便于学生接受和理解。此外，还扩充了 80x86/Pentium 指令和说明。子程序设计单独作为第 5 章，增加了嵌套与递归子程序设计内容，且实训项目做了相应地调整。第 1 版第 6 章中的“6.2.3 中断 I/O 方式”调整为第 2 版第 8 章中的“8.1 8086/8088 中断系统”，同时补充了部分内容；第 6 章中的实训三“PC 机间的相互通信：中断方式”调整作为第 8 章实训一。将“汇编语言程序设计上机指导”单独作为第 10 章。另外，还修正了原书中的错误与疏漏之处。

本书以 8086/8088 系列微机为背景，以 8086/8088CPU 为基础，由浅入深地介绍了汇编语言程序设计的相关知识。同时，考虑到计算机本身的发展，书中还专门介绍了 80x86 和 Pentium CPU 的新增指令。全书内容充分考虑了应用性本科学生的培养目标和教学特点。在内容的组织上，本着由浅入深、循序渐进的原则，注重基本知识和基本概念的介绍，结合实例重点介绍实用性较强的内容，对应用较少、难度过大的内容则少量介绍或不予介绍。全书注重层次，每一章的开头首先介绍本章主要内容，每一章的最后给出本章小结，使学生有的放矢，掌握所学内容。本书突出应用性，精选了一些实际应用例题，并介绍了汇编语言与高级语言混合编程的方法，目的是使学生学完本课程后能够用汇编语言解决一些实际问题。本书的另一个特色是理论教学与实践教学紧密结合，从第 4 章开始，每章都安排实训内容。理论部分主要介绍有关概念和程序设计

方法，实训部分则给出这些基本知识的应用范例。由此做到理论教学与实践教学同步融合，达到学以致用的目的。

本书编者多年从事汇编语言教学与科研工作，对汇编语言的教学与应用有深刻的理解和丰富的经验。书中的许多例题都来自具体的科研项目，而且所有例题都已调试通过。通过学习，学生能水到渠成地掌握汇编语言程序设计。

全书共分10章，第1章介绍汇编语言的基础知识，包括数据表示与运算、8086/8088CPU的系统结构、80x86/Pentium微处理器简介、计算机语言基本概念。第2章介绍8086/8088 CPU的寻址方式和指令系统以及80x86/Pentium新增指令。第3章介绍汇编语言的伪指令和汇编语言程序格式。第4章介绍基本程序设计，包括顺序程序设计、分支程序设计、循环程序设计。第5章介绍子程序设计。第6章介绍数值与非数值程序设计，包括数值运算程序设计和非数值处理程序设计。第7章介绍输入/输出程序设计。第8章介绍中断与系统功能调用。第9章介绍模块化程序设计与混合编程，包括汇编语言与高级语言的接口设计。第10章介绍汇编语言程序设计上机过程。

本书由王成端教授担任主编，王宇晓、王丰、李晓波任副主编，刘磊参编。其中，王成端编写第1、9章，并负责全书的统稿。王宇晓编写第4、5、6章，王丰编写第7、8章，李晓波编写第2、3章，刘磊编写第10章。魏先民参与程序上机调试工作，周建梁、张风云参与部分文字的录入、排版和绘图工作，在此表示感谢。本书编写前，对编写提纲进行了多次讨论，参与讨论提纲的学校有西安联合大学、上海第二工业大学、华北航天工业学院、承德石油高等专科学校等，并提出了许多好的建议。

山东大学的石冰教授对全稿进行了仔细认真的审阅，并提出了许多宝贵意见，在此表示衷心感谢。

由于作者水平有限，书中错误和不足之处在所难免，敬请广大读者批评指正。

编　者

2007年9月

目　录

第1章 基础知识

汇编语言是能够利用硬件资源直接进行编程的语言，因此具有占用空间小、执行时间效率高等特点。若用户需要编制直接控制硬件的程序，可以使用汇编语言进行程序设计。由于汇编语言具有面向机器的特性，因此随着 CPU 的发展，也增加了许多新的汇编语言指令。本章将介绍计算机中的数据表示及运算、8086/8088 系统结构，同时简要介绍了 80x86/Pentium 微处理器的性能及结构。

1.1 数据表示与运算

▶▶ 1.1.1 进位计数制与不同进制数之间的转换

1. 二进制数、八进制数和十六进制数

(1) 二进制数

日常生活中，人们一般采用十进制进行计数，但在计算机内部，数的运算和存储都是采用二进制，即计算机采用二进制进行计数。二进制数只有 0、1 两个数码，其基数为 2，遵循逢二进一的原则，它的第 k 位权以 2^k 表示。

二进制数 $a_n a_{n-1} \cdots a_0 \, a_{-1} a_{-2} \cdots \, a_{-m}$ 的值是：

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

其中 a_i 为 0、1 两个数码中的一个。二进制的描述是在其尾部加注字母 B，例如：

$$10100101.11B = 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = 128 + 32 + 4 + 1 + 0.5 + 0.25 = 165.75$$

n 位二进制数可以表示 2^n 个数，例如 3 位二进数可以表示 8 个数，如表 1-1 所示。

表 1-1 3 位二进制与十进制数对应表

二进制数	000	001	010	011	100	101	110	111
相应的十进制数	0	1	2	3	4	5	6	7

4 位二进制数则表示十进制的 0～15 共 16 个数，如表 1-2 所示。

表 1-2 4位二进制与十进制数的对应表

二进制数	0000	0001	0010	0011	0100	0101	0110	0111
相应的十进制数	0	1	2	3	4	5	6	7
二进制数	1000	1001	1010	1011	1100	1101	1110	1111
相应的十进制数	8	9	10	11	12	13	14	15

从上表可以看出,二进制数位数越多,所表示的十进制数越大。二进制数不便于人们阅读、书写和记忆,为此经常使用八进制数或十六进制数来表示二进制数。它们的基数和数码表示如表 1-3 所示。

表 1-3 几种常用的进位制的基数和数码

进位计数制	基 数	数 码
十六进制数	16	0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F
十进制数	10	0、1、2、3、4、5、6、7、8、9
八进制数	8	0、1、2、3、4、5、6、7
二进制数	2	0、1

(2) 八进制数

八进制数有 0、1、2、3、4、5、6、7 八个数码,其基数为 8,遵循逢八进一的原则,它的第 k 位权用 8^k 表示。八进制的描述是在其尾部加注字母 O 或 Q。

八进制数 $a_n a_{n-1} \cdots a_0 a_{-1} a_{-2} \cdots a_{-m}$ 的值是:

$$a_n \times 8^n + a_{n-1} \times 8^{n-1} + \cdots + a_0 \times 8^0 + a_{-1} \times 8^{-1} + a_{-2} \times 8^{-2} + \cdots + a_{-m} \times 8^{-m}$$

例如,$534.5Q = 5 \times 8^2 + 3 \times 8^1 + 4 \times 8^0 + 5 \times 8^{-1} = 5 \times 64 + 3 \times 8 + 4 \times 1 + 0.625 = 348.625$

(3) 十六进制数

十六进制数有 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F 十六个数码,其中 A、B、C、D、E、F 表示 10~15 六个数码,其基数为 16,遵循逢十六进一的原则,它的第 k 位权用 16^k 表示。十六进制的描述是在其尾部加注字母 H。

十六进制数 $a_n a_{n-1} \cdots a_0 a_{-1} a_{-2} \cdots a_{-m}$ 的值是:

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

例如,$2AC.CH = 2 \times 16^2 + 10 \times 16^1 + 12 \times 16^0 + 12 \times 16^{-1} = 2 \times 256 + 10 \times 16 + 12 \times 1 + 12 \times 16^{-1} = 684.75$

2. 不同数制之间的转换

(1) 非十进制数转换为十进制数

各位非十进制数码乘以与其对应的权之和即为该数对应的十进制数。例如:

$$1011100.1011B = 1 \times 2^6 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^{-1} + 1 \times 2^{-3} + 1 \times 2^{-4} = 92.6875$$

$A031H = 10 \times 16^3 + 3 \times 16^1 + 1 \times 16^0 = 41009$

$1001Q = 1 \times 8^3 + 1 \times 8^0 = 513$

（2）十进制数转换为非十进制数

十进制数转换为非十进制数一般整数部分采用除基数取余法，小数部分采用乘基数取整法。除基数取余法的具体方法是把待转换的十进制数的整数部分不断除以要转换为的非十进制数的基数，并记下余数，直到商为 0 时为止；乘基数取整法的具体方法是把待转换的十进制数的小数部分不断乘以要转换为的非十进制数的基数，逐次记下乘积整数部分的值，直到小数部分为 0 为止。

现以十进制数转换为二进制数为例进行说明，十进制数转换为二进制数的方法是将除基数取余法或乘基数取整法中的“基数”换成 2 就成了除 2 取余法和乘 2 取整法，即对整数部分的处理是把待转换的十进制数的整数部分不断除以 2，并记下余数，直到商为 0 时为止；对小数部分的转换是把待转换的十进制数的小数部分不断乘以 2，逐次记下乘积整数部分的值，直到小数部分为 0 为止。

【例 1.1】 将十进制数 $N = 137.8125D$ 转换成二进制数。

整数部分 137D，按除 2 取余法有：

$137/2 = 68$ $(a_0 = 1)$

$68/2 = 34$ $(a_1 = 0)$

$34/2 = 17$ $(a_2 = 0)$

$17/2 = 8$ $(a_3 = 1)$

$8/2 = 4$ $(a_4 = 0)$

$4/2 = 2$ $(a_5 = 0)$

$2/2 = 1$ $(a_6 = 0)$

$1/2 = 0$ $(a_7 = 1)$

故 137D = 10001001B。

小数部分为 0.8125D，按乘 2 取整法有：

$0.8125 \times 2 = 1.625$ $(a_{-1} = 1)$

$0.625 \times 2 = 1.25$ $(a_{-2} = 1)$

$0.25 \times 2 = 0.5$ $(a_{-3} = 0)$

$0.5 \times 2 = 1.0$ $(a_{-4} = 1)$

故 0.8125D = 0.1101B。

因此，$N = 137.8125D = 10001001.1101B$。

十进制数转换为十六进制数和八进制数的方法与十进制数转换为二进制数的方法类似。

（3）十六进制数与二进制数之间的转换

因为 $16 = 2^4$，所以一个十六进制数中的每一位可以用 4 位二进制数表示，便可形成相应的二进制数。

【例 1.2】 将十六进制数 CB9A 转换成二进制数。

C	B	9	A
1100	1011	1001	1010

即 CB9AH = 1100101110011010B。

反之，只要把二进制数从低位到高位每4位组成一组，再用十六进制数来表示即可。

【例1.3】 将二进制数0111010110111111转换成十六进制数。

0111	0101	1011	1111
7	5	B	F

即 0111010110111111B = 75BFH

▶▶ 1.1.2 二进制数和十六进制数运算

1. 二进制数的运算

加法规则：

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 10(1为进位)

乘法规则：

0 × 0 = 0

0 × 1 = 0

1 × 0 = 0

1 × 1 = 1

2. 十六进制数的运算

(1) 十六进制数的加法

十六进制数的运算按照逢十六进一的规则进行，即当两个一位数之和 S 小于16时，与十进制数同样处理，若两个一位数之和 S 大于等于16时，则应该用 S 减16的值替代 S，并向高位进1。

【例1.4】 十六进制加法示例。

```
  15C3H
+ 3D45H
--------
  5308H
```

(2) 十六进制数的减法

与十进制数类似，够减时可以直接相减，不够减时，按照向高位借1为16的规则。

【例1.5】 十六进制减法示例。

```
  3DA6H
- 0FC3H
--------
  2DE3H
```

▶▶ 1.1.3 数据表示

计算机内的数据有多种形式，其中最主要的是数值数据和字符数据。

1. 数值数据的表示

数值数据可以用不同的码制来表示，常用的有原码、补码、反码和移码表示法。由于作加减运算时补码表示法的符号位可以参加运算，而且不影响运算结果的正确性，因此多数机器的有符号整数都采用补码表示法。这里只介绍补码表示法。

(1) 数的补码表示

① 补码表示法中正数的表示。

正数采用符号—绝对值表示，即数的最高有效位为0表示符号为正，数的其余部分则表示数的绝对值。

例如，假设机器字长为8位，则

$[+0]_{补}=00000000B$

$[+1]_{补}=00000001B$

$[+100]_{补}=01100100B$

② 补码表示法中负数的表示。

用补码表示法来表示负数时可以采用“求反加1”的方法来完成：先写出与该负数相对应的正数的补码表示(用符号－绝对值法)，然后将其按位求反(即0变为1，1变为0)，最后在末位(最低位)加1，就可以得到该负数的补码。

【例1.6】 机器字长为8位，写出 $N=-27$ 的补码表示。

+27D可表示为	0001	1011
按位求反为	1110	0100
末位加1后为	1110	0101
用十六进制数表示为E5H		

即 $[-27]_{补}=E5H$。

【例1.7】 机器字长为16位，写出 $N=-32768$ 的补码表示。

32768D可表示为	1000	0000	0000	0000
按位求反为	0111	1111	1111	1111
末位加1后为	1000	0000	0000	0000
用十六进制数表示为	8	0	0	0

即 $[-32768]_{补}=8000H$。

③ 数的表示范围。

• 有符号数的表示范围。

一般来说，n 位二进制补码表示的数的表示范围是：

$$-2^{n-1}\leqslant N\leqslant 2^{n-1}-1$$

8位二进制数可以表示 $2^8=256$ 个数。因为在补码表示法中0只有一种表示，即00000000；对于10000000这个数，在补码表示法中被定义为－128。这样，8位补码能表示的范围为－128～＋127。

$n=16$ 时的数的表示范围是：

$$-32768 \leqslant N \leqslant +32767$$

• 无符号整数的表示范围。

在做无符号数处理时，把最高有效位作为数值处理。因此，16 位无符号数的表示范围是 $0 \leqslant N \leqslant 65535$，8 位无符号数的表示范围是 $0 \leqslant N \leqslant 255$。

（2）补码的运算

① 求补运算。

已知$[X]_{补}$，求$[-X]_{补}$的运算规则是：$[-X]_{补}=\overline{[X]_{补}}+1$，即将$[X]_{补}$按位求反，末位加 1。

【例 1.8】 已知$[+117]_{补}=0075H$，求$[-117]_{补}$。

运算如下：

$[+117]_{补}$为	0000	0000	0111	0101
按位求反后得	1111	1111	1000	1010
末位加 1 后得	1111	1111	1000	1011

即$[-117]_{补}=FF8BH$。

② 补码的加法、减法运算。

补码的加法规则是：

$$[X+Y]_{补}=[X]_{补}+[Y]_{补}$$

补码的减法规则是：

$$[X-Y]_{补}=[X]_{补}+[-Y]_{补}$$

【例 1.9】 机器字长假定为 8 位，完成下列补码加法运算。

	十进制	二进制
①	23	00010111
	+36	+00100100
	59	00111011
②	36	00100100
	+(-23)	+11101001
	13	(进位 1)00001101
③	23	00010111
	+(-36)	+11011100
	-13	11110011

④	−23	11101001
	+(−36)	+11011100
	−59	(进位1)11000101

可以看出，上述4个例子的计算结果都是正确的，在②和④中，从最高有效位向高位的进位由于机器字长的限制而自动丢失，但这并不会影响运算结果的正确性。

【例1.10】 机器字长假定为8位，完成下列补码减法运算。

	十进制	补码	二进制
①	23	00010111	00010111
	−36	00100100	+11011100
	−13	11110011	11110011
②	36	00100100	00100100
	−(−23)	11101001	+00010111
	59	00111011	00111011
③	−23	11101001	11101001
	−(+36)	00100100	+11011100
	−59	(进位为1)11000101	(进位为1)11000101
④	−23	11101001	11101001
	−(−36)	11011100	+00100100
	13	(进位为1)00001101	(进位为1)00001101

可以看出，补码减法是用对减数求补后把减法转换为加法进行的，它能自动地得到正确的结果。

2. 字符数据的表示

计算机内的常见字符包括：

- 字母：A、B、…、Z，a、b、…、z。
- 数字：0～9。
- 标点符号：!、,、?、;、:和、等。
- 算术运算符号：+、−、×、/。
- 关系运算符号：>、<、=。
- 专用字符：*、$、@、%、#和空格等。
- 非打印字符：BEL(响铃)、LF(换行)、CR(回车)等。

这些字符一般采用目前最常用的美国信息交换标准代码ASCII码来表示。这种代码用一个字节(8位二进制码)来表示一个字符，其中低7位为字符的ASCII值，最高位一般用作校验位。

常见字符的 ASCII 码表见附录Ⅰ。

1.1.4 定点数与浮点数

计算机中根据小数点的位置是否固定,将数的表示分为定点数表示和浮点数表示。

1. 定点数

定点数是指小数点位置固定不变的数。小数点的位置通常只有两种约定,小数点约定在最低数位右面的称为定点整数,可用来表示一个纯整数。小数点约定在符号位右面,最高数位左面的数称为定点小数,可用来表示一个纯小数。

无符号定点整数,即正整数,不需要设符号位,所有各数位都用来表示数值大小,并约定小数点在最低位数的右面。

在定点整数或定点小数的表示法中,参加运算的数以及运算的结果必须在该定点数所能表示的数值范围之内,否则"溢出"。当发生溢出时,CPU 中的状态标志寄存器中的溢出标志 OF 置 1。

2. 浮点数

定点数的表示比较单一,要么为纯整数,要么为纯小数,表示数的范围比较小,运算过程中也很容易发生溢出。计算机中也引入了类似于十进制的科学标识法(如 1.23456×10^4)来表示二进制实数,这种方法用来表示值很大或很小的数,也可以用来表示既有整数又有小数的数。这种方法称为浮点表示法,其小数点的实际位置随指数(阶)的大小而浮动。

浮点数由两部分组成:阶码 E 和尾数 M。浮点数表示的数值为 $M \times R^E$。若尾数 M 为 m 位,阶码 E 为 e 位(含 1 位阶符 E_S)。典型的浮点数格式如图 1-1 所示。

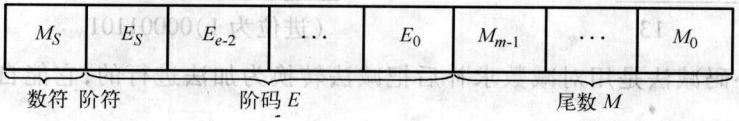

图 1-1 浮点数格式

图 1-1 中,E 是阶码,即指数,为带符号定点整数,可用补码表示。E_S 是阶符,表示阶的正负。若阶为正,小数点实际位置向右浮动;若阶为负,小数点实际位置则向左浮动。

M 是尾数,是带符号的定点小数,常用补码表示。M_S 是尾数的符号位,安排在最高位,表示该浮点数的正负。

小数点的位置约定在阶码最低位的右面,尾数最高数值位的左面。

R 是阶码的底,也就是尾数 M 的基(Radix)。一般基为 2,它是隐含约定的。

浮点数的表示范围由阶码的位数决定,浮点数的精度则取决于尾数的位数。