21世纪高等院校计算机教材系列

汇编语言程序设计

王成耀　姚 琳　编著

机械工业出版社
CHINA MACHINE PRESS

21世纪高等院校计算机教材系列

汇编语言程序设计

王成耀　姚　琳　编著

机械工业出版社

本书以 Microsoft 宏汇编 MASM 6.1x 为背景，系统讲述了 8086 指令系统及汇编语言程序设计的基本方法和技术，介绍了 32 位 x86 指令及其程序设计，并以 Microsoft Visual C++ 6.0 为背景，介绍了 Windows 9x/2000 等 32 位环境下 C/C++ 语言与汇编语言的混合编程。全书共分 9 章，主要内容包括：汇编语言程序设计的基础知识、8086 指令系统、源程序的基本框架、程序设计的基本技术、宏指令、多模块程序设计、输入输出和中断程序设计、32 位 x86 指令及其程序设计等。其中，第 1 章至第 8 章可供学习 8086 汇编语言的读者使用。对于熟悉 8086 汇编语言的读者，也可从第 9 章得到 32 位 x86 指令及其程序设计的有关知识。书中提供了大量的程序实例，所有实例都经过上机验证。每章后均附有习题。

本书不仅可作为高等院校计算机及相关专业的学生学习 8086 汇编语言的教材或参考书，也可供学习 32 位 x86 汇编语言的读者使用。

图书在版编目（CIP）数据

汇编语言程序设计/王成耀，姚琳编著. —北京：机械工业出版社，2003.3
（21 世纪高等院校计算机教材系列）
ISBN 7-111-11650-X

Ⅰ. 汇... Ⅱ. ①王... ②姚... 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（2003）第 008209 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策　　划：胡毓坚
责任编辑：陈振虹
责任印制：付方敏

北京市密云县印刷厂印刷 ·新华书店北京发行所发行

2003 年 3 月第 1 版·第 1 次印刷
787mm×1092mm 1/16·18.25 印张·448 千字
0 001—5 000 册
定价：26.00 元

凡购本图书，如有缺页、倒页、脱页，由本社发行部调换
本社购书热线电话：（010）68993821、88379646
封面无防伪标均为盗版

出版说明

随着计算机技术的飞速发展，计算机在经济与社会发展中的地位日益重要。在高等院校的培养目标中，都将计算机知识与应用能力作为其重要的组成部分。根据计算机科学发展迅速的学科特点，计算机教育应面向社会，面向潮流，与社会接轨，与时代同行。随着计算机软硬件的不断更新换代，计算机教学内容也必须随之不断更新。

为满足高等院校计算机教材的需求，机械工业出版社聘请了清华大学、北方交通大学、北京邮电大学等院校的老师，经过反复研讨，结合当前计算机发展需要和编者长期从事计算机教学的经验精心编写出“21世纪高等院校计算机教材系列”。

本套教材理论教学和实践教学相结合，图文并茂，内容实用，层次分明，讲解清晰，系统全面，其中溶入了老师大量的教学和科研经验，是各类高等院校、高等职业学校及相关院校的最佳教材，也可作为培训班和自学使用。

前　言

随着计算机技术的发展和各种软件开发平台的不断涌现，目前完全用汇编语言实现的软件系统已极为罕见，汇编语言也难以胜任大型软件系统的开发。然而，作为一种最能充分发挥计算机硬件特性的程序设计语言，汇编语言可以实现高级语言难以胜任甚至无法完成的任务，尤其适合对时空效率要求较高、与机器硬件密切相关的软件，例如，操作系统的部分核心代码，要求能快速响应的实时系统等。

“汇编语言程序设计”作为高等院校计算机及相关专业本科生的技术基础课，是学习操作系统与编译原理等课程的基础。同时，掌握汇编语言知识，对于提高程序设计水平、加深对计算机系统的理解也是非常重要的。

考虑到读者学习 8086 汇编语言的需要，而 8086 指令系统又是整个 x86 指令系统的基础，为此，本书首先以 8086 指令系统为背景，基于 Microsoft 宏汇编 MASM 6.1x，系统讲述了汇编语言程序设计的基本方法和技术，最后介绍了 32 位 x86 指令及其程序设计。

全书共分 9 章。第 1 章介绍了学习汇编语言所需的基础知识；第 2 章介绍了 8086 计算机的基本结构与寻址方式；第 3 章详细介绍了 8086 指令系统，并给出了大量的指令使用实例；第 4 章介绍了 MASM 汇编语言源程序的基本框架以及程序开发过程；第 5 章结合具体实例，讲述了用三种基本控制结构（顺序结构、分支结构与循环结构）设计汇编语言程序的基本方法；第 6 章讨论了过程及其参数传递方法；第 7 章简要介绍了宏与多模块程序设计；第 8 章讲述了输入/输出与中断程序设计的基本方法；第 9 章介绍了 32 位 x86 指令系统及其程序设计，并以 Microsoft Visual C++ 6.0 为背景，讲述了 Windows 9x/2000 等 32 位环境下 C/C++ 语言与汇编语言混合编程的基本方法。

本书注重实用性，力求做到通俗易懂。书中含有大量的程序实例，所有实例都经过上机验证，许多实例都给出了多种解决方法。每章后均有习题，以便读者复习和检查学习效果。

通过本书的第 1～8 章，读者可以系统学习 8086 汇编语言程序设计的基本技术。这一部分内容自成体系，无需其他先修知识。如果读者具备高级语言（如 Pascal、C 等）程序设计的基础，将有助于对部分内容的深入理解。对 32 位 x86 指令系统及其程序设计感兴趣的读者，可从第 9 章学到有关知识。因此，本书不仅可作为学习 8086 汇编语言的的教材或参考书，也可供学习 32 位 x86 汇编语言的读者使用。

欢迎读者对书中的错误与不妥之处提出批评并给予指正。

编　者

目　录

第1章 基础知识

汇编语言是机器语言的符号表示。用汇编语言设计程序，可以充分利用和发挥计算机硬件的特性与优势。学习汇编语言，不仅可以加深对计算机系统尤其是程序工作原理的理解，而且有助于提高程序设计水平。本章介绍学习汇编语言程序设计所必须具备的基础知识，主要包括数制、汇编语言的基本概念以及计算机中数据的表示方法。

1.1 数制及其转换

1.1.1 数制

在日常生活中，人们最常用和最熟悉的数制是十进制(Decimal)。然而，现代计算机系统采用二进制(Binary)来表示数据。由于二进制数书写太冗长，而且二进制数与十进制数之间的转换又较为繁琐，因此，在计算机应用特别是汇编语言程序设计中，常采用十六进制(Hexadecimal)。十六进制数简短、易读，与二进制数之间的转换也非常容易。此外，有时也使用八进制(Octal)数。

十进制数用数字 0 ～ 9 描述，基数是 10，运算规则是“逢 10 进 1”。

二进制数用数字 0 和 1 描述，基数是 2，运算规则是“逢 2 进 1”。

十六进制数用数字 0 ～ 9 和 A ～ F(或 a ～ f)描述，其中 A ～ F 表示 10 ～ 15，基数是 16，运算规则是“逢 16 进 1”。

八进制数用数字 0 ～ 7 描述，基数是 8，运算规则是“逢 8 进 1”。

概括起来，对于 X 进制数来说，采用数字 0 ～ X－1 描述，基数是 X，运算规则是“逢 X 进 1”，实际值为各位数字与相应权值乘积之和。X 进制数 $a_n a_{n-1} \cdots a_0 . a_{-1} a_{-2} \cdots a_{-m}$ 对应的十进制数值为：$a_n * X^n + a_{n-1} * X^{n-1} + \cdots + a_0 * X^0 + a_{-1} * X^{-1} + a_{-2} * X^{-2} \cdots + a_{-m} * X^{-m}$

其中，数据的前导 0 可以忽略。

在汇编语言程序中，为了能表示各种不同进制数，在数据的结尾用一个字母来区分。其中，十进制数用 D 或 d，二进制数用 B 或 b，十六进制数用 H 或 h，八进制数用 O 或 o，默认为十进制数。例如，6A32H 是十六进制数，101101B 是二进制数，236 与 238D 是十进制数，等等。此外，对于十六进制数，若以字母开头，则前面需要补 0，以避免与标识符相混淆。例如，十六进制数 A9F6H 应书写为 0A9F6H。

1.1.2 数制之间的转换

计算机内部处理的数据都是二进制数。然而，在程序设计时，为了方便，常常使用一些其他进制的数据。因此，为了能用不同进制表示数据，需要掌握各进制数之间的转换方法。下面介绍整数在二进制、八进制、十六进制与十进制之间的转换方法。

1. 二进制、八进制或十六进制整数转换为十进制整数

二进制、八进制或十六进制整数转换为十进制整数比较容易，各位数字与其对应权值乘积之和即为等值的十进制整数。例如：

$10110010B = 2^7 + 2^5 + 2^4 + 2^1 = 178$

$0BF3CH = 11 * 16^3 + 15 * 16^2 + 3 * 16^1 + 12 * 16^0 = 48956$

$255O = 2 * 8^2 + 5 * 8^1 + 5 * 8^0 = 173$

2. 十进制整数转换为二进制、八进制或十六进制整数

设给定的十进制整数为 N。通常，将十进制整数转换为 X 进制整数，可采用下列两种方法。

(1)降幂法

降幂法按照从高位到低位的次序生成等值的 X 进制整数的各位数字。具体方法如下：

写出所有小于 N 的各位 X 进制权值，将这些权值以从大到小的次序排列，然后不断地用余数(开始时取 N)除以各权值，所得的商即为对应 X 进制整数各位的值。

(2)除法

除法按照从低位到高位的次序生成等值的 X 进制整数的各位数字。具体方法如下：

用商(开始时取 N)不断除以 X，记下余数，直到商为 0 为止。然后，将所得的余数逆序排列，即可得到等值的 X 进制整数。这种方法可简单地记为“除以 X 取余数，逆序排列得整数”。

【例 1-1】 设 N = 85，求 N 对应的二进制数。

方法 1：采用降幂法。

小于 N 的各位二进制权值为：

64,32,16,8,4,2,1

计算过程如下：

85/64 = 1，余 21

21/32 = 0，余 21

21/16 = 1，余 5

5/8 = 0，余 5

5/4 = 1，余 1

1/2 = 0，余 1

1/1 = 1

故 N = 1010101B。

方法 2：采用除法。

计算过程如下：

85/2 = 42，余 1

42/2 = 21，余 0

21/2 = 10，余 1

10/2 = 5，余 0

5/2 = 2，余 1

2/2 = 1，余 0

1/2 = 0，余 1

将所得的余数逆序排列,可得

N = 1010101B。

【例 1-2】 设 N = 48958,求 N 对应的十六进制数。

方法 1:采用降幂法。

小于 N 的各位十六进制权值为:

4096,256,16,1

计算过程如下:

48958/4096 = 11,余 3902

3902/256 = 15,余 62

62/16 = 3, 余 14

14/1 = 14

故 N = (11)(15)(3)(14)= 0BF3EH。

方法 2:采用除法。

计算过程如下:

48958/16 = 3059, 余 14

3059/16 = 191, 余 3

191/16 = 11, 余 15

11/16 = 0, 余 11

将所得的余数逆序排列,可得

N = 0BF3EH。

3. 二进制整数与八进制或十六进制整数之间的转换

由于 $2^4 = 16$,所以任何一个 4 位二进制数均可由一个 1 位的十六进制数来表示。因此,二进制数与十六进制数之间的转换非常方便。将二进制整数转换为十六进制整数时,只要将二进制整数从低位到高位每 4 位用一个十六进制位表示即可;反之,将十六进制整数转换为二进制整数时,只要将每个十六进制位用 4 个二进制位表示即可。例如:

1110101101B = 11 1010 1101B = 3ADH

表 1-1 给出了十进制、二进制与十六进制数字之间的对应关系。

表 1-1 十进制、二进制与十六进制数字对应表

十 进 制	二 进 制	十六进制	十 进 制	二 进 制	十六进制
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

同样,由于 $2^3 = 8$,因此,将二进制整数转换为八进制整数时,只要将二进制整数从低位到高位每 3 位用一个八进制位表示即可;反之,将八进制整数转换为二进制整数时,只要将每个八进制位用 3 个二进制位表示即可。例如:

1110101101B = 1 110 101 101B = 1655O

1.1.3 二进制与十六进制的运算规则

在进行二进制数或十六进制数的运算时，虽然可以先转换为十进制数进行运算，然后将结果再转换为二进制数或十六进制数，但这样做比较繁琐。其实，二进制、十六进制运算与十进制类似，只不过是“逢 2 进 1”或“逢 16 进 1”。

基本的二进制算术运算规则如下。

加法规则：

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0(进位 1)

减法规则：

0 - 0 = 0

0 - 1 = 1(借位 1)

1 - 0 = 1

1 - 1 = 0

乘法规则：

0 × 0 = 0

0 × 1 = 0

1 × 0 = 0

1 × 1 = 1

【例 1-3】 二进制数的算术运算。

1)
```
  1101B
+ 0011B
-------
 10000B
```

2)
```
  1101B
- 0011B
-------
  1010B
```

3)
```
   1101B
 × 0011B
--------
   1101
+ 1101
--------
 100111B
```

【例 1-4】 十六进制数的算术运算。

1)
```
  26A3H
+ 0C16H
-------
  32B9H
```

2)
```
  26A3H
- 0C16H
-------
  1A8DH
```

3)
$$\begin{array}{r} 4BH \\ \times 13H \\ \hline E1 \\ +4B \\ \hline 591H \end{array}$$

二进制数以及十六进制数的除法可根据其乘法和减法规则处理，这里不再赘述。

1.2 程序设计语言

自然语言是人类用来表达思想和实现相互交流的工具。若两个人使用同一种语言，则可以直接进行交流，否则就需要借助于翻译。要使计算机为人类服务，人们就必须通过某种工具，告诉计算机“做什么”甚至“怎么做”，这种工具就是程序设计语言。

程序设计语言通常分为三类：机器语言(Machine Language)、汇编语言(Assembly Language)和高级语言(High Level Language)。其中，机器语言和汇编语言是与机器密切相关的，统称为低级语言。

1.2.1 机器语言

计算机能直接识别并进行处理的是由 0 和 1 组成的二进制代码，计算机的工作就是对二进制信息进行处理。

1. 机器指令

机器指令是指用二进制编码的指令，指示计算机所要进行的操作以及操作的数据对象。机器指令由指令译码器所识别，并经过一定的时钟周期付诸实现，从而完成指令所规定的操作。

机器指令由操作码(Opcode)和操作数(Operand)构成。操作码指出指令要执行的操作，如加、减、乘、除或移位等。操作数指出所要操作的数据或数据地址。

2. 指令系统与机器语言

指令系统(Instruction Set)是指机器指令的集合。指令系统及其机器指令的使用规则便构成了机器语言。

机器语言是计算机惟一能够识别的语言。例如，下面是用 Intel 8086 CPU 机器语言编写的一段代码(16 进制)：

```
B019
B227
F6E2
83C03E
83E838
```

虽然只有几行代码，但很少有人能直接看出它的功能就是计算表达式 25 * 39 + 62 - 56 的值。

3. 机器语言的主要特点

机器语言主要具有下列特点：

(1) 机器语言与机器硬件密切相关

机器指令与计算机的 CPU、内存管理机制以及 I/O 机制有着十分密切的关系。通常,不同类型 CPU 的指令系统往往有较大差异,用一种机器指令设计的程序在其他类型的计算机上就不能使用。然而,同一系列 CPU 的指令系统往往具有兼容性,即较高级别的 CPU 指令系统是较低级别 CPU 指令系统的超集。例如,Intel 80486 指令系统包含 Intel 8086 指令系统。

(2) 用机器语言设计程序非常困难

用机器语言设计出的程序不仅难以阅读和理解,不易调试、维护和移植,而且通用性差,其开发效率也很低,难以满足大型软件系统的开发需要。

(3) 用机器语言设计程序容易得到时间和空间上的最优代码

由于机器语言与机器硬件的密切相关性,使得用机器语言设计程序能最大限度地发挥计算机的硬件特性和优势,因此,容易得到运行速度快、执行代码短、占用内存空间少的高时空效率的程序。

1.2.2 汇编语言

为了克服机器语言难以记忆、阅读和理解的缺点,人们将机器指令符号化,以直观、便于记忆的符号来表示机器指令,这些符号被称为指令助记符(Mnemonic)。例如,用 ADD 表示加法指令,SUB 表示减法指令,MOV 表示传送指令,JMP 表示转移指令等。

以助记符描述的指令称为汇编格式指令或符号指令,通常简称指令。指令和伪指令的集合及其程序设计规则便构成了汇编语言。伪指令的概念将在第 4 章介绍。

若使用汇编语言来计算表达式 25 * 39 + 62 - 56 的值,则可设计如下的程序:

```
mov al, 25
mov dl, 39
mul dl
add ax, 62
sub ax, 56
```

可以看出,用汇编语言编写的程序要比机器代码容易理解得多。

然而,汇编语言与机器语言并无本质区别,汇编语言仅仅是机器语言的符号表示,每条汇编语言指令均对应惟一的机器指令。因此,汇编语言同样具有机器语言"与机器硬件的密切相关性"等特点,只是在直观和记忆方面有了较大改进。

由于计算机只能识别机器语言,因此,用汇编语言编写的源程序不能在计算机上直接执行,必须将其翻译成机器语言。把汇编语言源程序翻译成由机器语言描述的目标程序的过程称为汇编。实现汇编任务的程序称为汇编器(Assembler)或汇编程序。

汇编器的主要功能是对汇编语言源程序进行语法检查,若源程序没有语法错误,则将符号指令翻译为机器指令,生成相应的目标文件(.OBJ 文件)。汇编器类似于高级语言的编译器(Compiler)。

尽管目标文件已经是机器语言程序,但还不能直接执行,需要通过连接器(Linker,或称连接程序)将其与其他目标文件或库文件连接在一起,生成可执行文件(.EXE 文件)后,才能在计算机上运行。

连接器的主要功能是实现多个目标文件及库文件的连接,定位目标文件中可能存在的外

部符号，完成浮动地址的重定位，最后生成可执行文件。

1.2.3 高级语言

尽管与机器语言相比，汇编语言在记忆和直观等方面有了很大改进，但并无本质上的提高。人们希望有一种接近自然语言或数学表达形式的程序设计语言，使程序设计工作能避开与机器硬件相关的细节，而着重于解决问题的算法本身，因此便产生了高级语言。目前，常用的高级语言有数十种，如 Pascal、C/C++、Java 等。例如，在高级语言程序中就可以直接使用表达式 $25 \times 39 + 62 - 56$。

与汇编语言相比，高级语言在程序设计的简易性以及代码的可读性和可移植性等方面有了质的飞跃。当然，用高级语言编写的源程序必须经过编译和连接，将其转换为可执行程序或借助于解释程序才能运行。

1.2.4 学习汇编语言的意义

高级语言简单、易学、程序开发效率高，而汇编语言复杂、难懂、程序开发效率低。那么，是否就意味着没有必要学习和使用汇编语言了呢？

由于汇编语言本质上就是机器语言，可直接、有效地控制计算机硬件，因而与高级语言相比，容易得到时间和空间上最优的目标程序。然而，不可否认，汇编语言具有如下缺点：

- 汇编语言难学、难理解。
- 汇编语言程序的可移植性差。
- 用汇编语言设计程序复杂、低效、易出错，难以胜任大型软件系统的开发。

同时，随着计算机速度的不断提高和内存容量的大幅增加，用高级语言就可以满足大多数问题对时间和空间方面的要求，因而汇编语言的优势已不再突出。

然而，汇编语言在某些方面确实具有高级语言无法相比的独特优势。因此，学习汇编语言的意义可归纳为如下四个方面：

1）速度：对于同一个问题，用汇编语言设计出的程序能达到“运行速度最快”。

2）空间：对于同一个问题，用汇编语言设计出的程序能达到“占用空间最少”。

3）功能：正是由于与机器的密切相关性，使得汇编语言能充分利用计算机的硬件特性，编写出与硬件紧密相关、而高级语言难以胜任甚至无法实现的程序来。

4）知识：掌握汇编语言知识，有助于加深对计算机系统特别是程序执行原理的理解，有助于写出更好的高级语言程序来。

事实上，随着计算机技术的发展和各种软件开发平台的不断涌现，目前完全用汇编语言实现的软件系统已极为罕见。因此，我们不能期望完全用汇编语言开发大型的软件系统，而应尽可能扬长避短。汇编语言正是由于其“面向机器”的特点，广泛用于高性能软件的开发中，例如，操作系统的部分核心代码、要求能快速响应的实时系统等。

需要特别指出的是，虽然汇编语言不具有通用性，不同 CPU 的指令系统可能有较大的差异，但其原理和方法是具有普遍性的。只要熟练掌握一种汇编语言，再学习其他汇编语言就相当容易了。