高职高专软件技术专业系列教材

汇编语言程序设计

[illegible]

[illegible]
http://www.[illegible].com.cn

面向 21 世纪高等职业技术教育计算机类系列教材

汇编语言程序设计

主　编　李革新

副主编　杨金花　张先军

参　编　宋健民　宋继红　王　姝

丁春莉　吴红红

主　审　张晓云

西安电子科技大学出版社

2003

内 容 简 介

本书采用全新的编撰方法，以 Intel 8086/8088 微处理器为基础，介绍了汇编语言和机器语言的基本概念、基本原理以及它们在计算机中的基本工作过程；还介绍了汇编语言程序设计的基本方法和技巧。

本书内容的编排由浅入深、由简到繁、由易到难、循序渐进；指令和程序设计的学习均融于某一实际问题之中，针对性、趣味性较强；突出汇编语言程序设计的一般方法，读者可以边学习、边上机操作，在实践中体会知识的趣味性和可操作性。

本书为高职高专计算机专业的“汇编语言程序设计”课程的教材，也可以作为电子、自动控制等专业的相关教材，适用于高等职业技术学院、高等专科学校、成人教育学院及本科院校举办的二级职业技术学院和民办高校，更适合于电脑爱好者自学、提高之用。

★本书配有电子教案与多媒体课件， 需要者可与出版社联系，免费索取。

图书在版编目(CIP)数据

汇编语言程序设计／李革新主编. —西安：西安电子科技大学出版社，2003.7

(面向 21 世纪高等职业技术教育计算机类系列教材)

ISBN 7–5606–1258–X

Ⅰ. 汇… Ⅱ. 李… Ⅲ. 汇编语言—程序设计—高等学校：技术学校—教材 Ⅳ. TP313

中国版本图书馆 CIP 数据核字(2003)第 046796 号

策　　划　马乐惠

责任编辑　龙晖

出版发行　西安电子科技大学出版社(西安市太白南路 2 号)

电　　话　(029)8242885　8201467　邮　编　710071

http: //www.xduph.com　　E-mail: xdupfxb@pub.xaonline.com

经　　销　新华书店

印刷单位　西安文化彩印厂

版　　次　2003 年 7 月第 1 版　2003 年 7 月第 1 次印刷

开　　本　787 毫米×1092 毫米　1/16　印张 17.25

字　　数　397 千字

印　　数　1～4 000 册

定　　价　19.00 元

ISBN 7 – 5606 – 1258 – X／TP・0661(课)

XDUP 1529001-1

＊＊＊ 如有印装问题可调换 ＊＊＊

前　言

微型计算机的普及，使得人们对汇编语言产生了浓厚的兴趣。其主要原因有二：第一，用汇编语言编写的程序，占用内存空间小而且执行速度快；第二，懂得汇编语言与它所产生的机器指令码，可以使你更多地了解计算机的硬件结构，这是高级语言无法做到的。尽管现在已有许多新的易写的高级语言出现，但是目前功能最强、效率最高的软件很多都是用汇编语言编写的。总之，8086/8088 汇编语言是你进入计算机内部世界的方便之门。

全书共分 18 章。第 1、2、3 章介绍基础知识、机器语言程序的建立与调试以及编写汇编语言的基础知识；第 4 章介绍 EXE 文件与 COM 文件程序的格式；第 5、6、7 章介绍数据定义、程序的三种结构与逻辑运算；第 8 章介绍屏幕和键盘的基本管理方法；第 9、10、11、12 章介绍字符串处理、算术运算、代码转换与表处理；第 13、14 章介绍宏指令与模块化程序设计；第 15、16 章介绍屏幕处理的高级特性与屏幕彩色图形；第 17、18 章是伪指令与指令的详细参考资料。附录 A 是 ASCII 码表，附录 B 介绍最新的软件开发工具，附录 C 是课程设计。书中每章后均附有练习题，其中带星号(*)的为上机题。书中第 2 章与第 8 章配有多媒体交互式学习课件，以帮助初学者顺利地进入汇编语言的学习。

本书由李革新任主编，她负责全书的总体规划和统稿；杨金花、张先军任副主编；其中，杨金花编写第 1、2、3 章，王姝编写第 4、5 章，李革新编写第 6、7、8 章和附录 C，宋继红编写第 9、10、11 章，宋健民编写第 12、13、14 章，张先军编写第 15、16 章，丁春莉编写第 17、18 章和附录 A，吴红红编写 2.4、4.3、10.6 节和附录 B。

西安航空技术专科学院计算机系张晓云主任担任本书主审并提出了多条宝贵的意见，在此表示感谢。

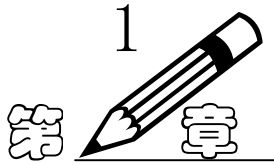
当学完此书，你将具有下面的能力：

- 了解微型计算机的硬件结构；了解机器语言与十六进制格式。
- 了解汇编、链接与执行的步骤。
- 用汇编语言编写控制键盘、屏幕、算术运算、ASCII 码与二进制数的转换以及表的搜寻与排序的程序。
- 编写自己设计的宏指令。
- 把多个汇编程序链接成一个可执行文件。

编　者

2003 年 4 月

目　　录



第 1 章　基 础 知 识

汇编语言主要应用于对设备或对象的控制。例如，机床工业中的数控车床，商业广告中的霓虹灯，军事设备中各种弹道轨迹的控制等。由此可见，汇编语言在我们的日常生活和工作中起着相当大的作用，它与实现现代化有着密切的关系。

编写汇编语言程序必须了解计算机系统的组成。计算机中信息表示的最小单位是位(bit)，最基本的单位是字节(byte)。计算机利用它们来表示存储器中的信息。用机器指令码形式表示的程序，称为机器语言程序。机器语言程序一般含有数据段、机器指令代码段、以及堆栈段等不同的段(Segment)。

为了处理算术运算、数据传送、以及对存储器单元的访问等操作，微型计算机利用了一些寄存器(Register)。

本章包含了所有这些内容，以便你能很快地进入第 2 章的第一个机器语言程序。

1.1　数 与 数 制

微型计算机亦称为微电脑，它仅能识别由 0 和 1 组成的机器语言和二进制数。由于十六进制表示的数更简短直观，与二进制数转换也很方便，所以，十六进制在计算机中也被广泛使用。

1.1.1　位与字节

微型计算机中使用二进制数表示和存储信息，位(bit)是微型计算机存储信息的最小单位。一个位有两种状态，分别为 0 状态与 1 状态，可以表示二进制数的 0 和 1。

8 位二进制数称为一个字节(byte，缩写为 B)，微型计算机中表示字母、符号的二进制编码的长度就是一个字节。例如，字母“A”与星号“*”的二进制编码，分别为 01000001 与 00101010。

计算机怎么知道 01000001 代表字母“A”呢？它是以这样的原理设计和工作的：当你从键盘上键入“A”时，系统会从键盘上收到一个二进制数，将这一二进制数设定成 01000001，存放于存储器的一个存储单元内。通过程序中的指令你可以在存储器中随意地移动此二进制数，而当程序中的指令把它送往屏幕或从打印机输出时，应为字母“A”。计算机就是以这样的约定来工作并以此来编制程序的。

为了便于说明和使用，字节中的各位自右向左从 0 开始编号到 7，比如字母“A”的各位分别如下：

字节的位编号：7 6 5 4 3 2 1 0

字节的位内容：0 1 0 0 0 0 0 1

或

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
0	1	0	0	0	0	0	1

2^{10} 等于 1024，表示 1K。例如，含有 512 KB 的存储器，是指计算机的存储容量为 512×1024×8 位或 512×1024 个字节。

由于 PC 机以及其兼容机型所用的微处理器是 16 位的 CPU，因此它能在存储器单元和寄存器中存取一个 16 位的二进制数。16 个位(2 个字节)称为一个字(word)。字中的各位自右向左从 0 开始编号至 15，如下面是表示字母“ PC ”的字：

字的位编号：15 14 13 12 11 10 9 8 | 7 6 5 4 3 2 1 0

字的位内容：0 1 0 1 0 0 0 0 | 0 1 0 0 0 0 1 1

或

b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
0	1	0	1	0	0	0	0	0	1	0	0	0	0	1	1

1.1.2 ASCII 码

为了标准化，微型计算机采用 ASCII (American Standard Code for Information Interchange)码。使用标准码可以简化不同微型计算机之间的数据传输问题。IBM PC 所使用的 8 位延伸 ASCII 码可以提供 256 个字符。ASCII 码的特点如下：

(1) 数字的 ASCII 码按照 0～9 顺序逐渐增大；

(2) 数字的 ASCII 码小于字母的 ASCII 码；

(3) 字母的 ASCII 码按 26 个字母顺序逐渐增大；

(4) 大写字母的 ASCII 码小于小写字母的 ASCII 码。

第 8 章会告诉你如何在屏幕上显示所有的 256 个 ASCII 码字符。附录 A 是 ASCII 码表。

1.1.3 二进制数及运算

由于微型计算机只能区分 0 与 1，所以微型计算机使用二进制数。位(bit)就是二进制数(Binary Digit)的缩写。

一个十进制数 $241=2\times10^2+4\times10^1+1\times10^0$，10 称为基数，$10^2$、$10^1$、$10^0$ 称为各位的“位权”。同理，一个二进制数 $1101=1\times2^3+1\times2^2+0\times2^1+1\times2^0$，2 称为基数，$2^3$、$2^2$、$2^1$、$2^0$ 亦称为各位的“位权”。以 8 位二进制数为例，当每一位均为 1 时，各位的“位权”为：

字节位的位置值(位权)：128 64 32 16 8 4 2 1

字节位的值： 1 1 1 1 1 1 1 1

8 位二进制数的“位权”从右至左依次为 1，2，4，…，128，所有“位权”的和为 $1+2+4+\cdots+128=255$ 或(2^8-1)。

前面的 0100001 可以表示字母“ A ”，但是，0100001 也可以表示数值 65(十进制)，所以在选用数据时必须确认 0100001 究竟是表示数值 65 还是字母“ A ”，区分方法如下：

(1) 如果程序中所定义的数据项是作为算术运算用的，则 01000001 表示的是二进制数值，等于十进制的 65。

(2) 如果程序中所定义的数据项是作为说明字符使用的，则 01000001 表示的是字母“A”。

当你开始写程序时，由于你必须定义每一个数据项的用途，所以将会发现此种区别是十分明显和重要的。

二进制数并不只限用于 8 位。由于个人微型计算机的微处理器 8086 是 16 位的 CPU，所以它可以处理 16 位的二进制数，16 位二进制数的范围是 0～65 535(即 $2^{16}-1$)。而 32 位的 CPU 允许使用 32 位，其范围是 0～4 294 967 295(即 $2^{32}-1$)。

1. 二进制的算术运算

微型计算机只能执行二进制数的算术运算。因此，一个汇编语言程序设计员必须熟悉二进制数的格式与二进制数的运算规则。

二进制加法：

0+0=0

1+0=1

1+1=10　　(有进位)

1+1+1=11　　(有进位)

例如，练习做 01000001 与 00101010 的加法，此处它们分别代表十进制数 65 与 42。

二进制	十进制
01000001	65
+00101010	+ 42
01101011	107

将二进制和为 1 的权相加，核对此二进制数的和确实是等于 107。下面再练习另外一个例子。

二进制	十进制
00111100	60
+00110101	+ 53
01110001	113

2. 负数

在微型计算机中数的正负号是用 0 和 1 表示的，0 表示正号，1 表示负号，并用最左边(最高)的位表示数的正负号。前面所提到的二进制数，其最左边(最高)的位都是零，所以均为正值。而负的二进制数，其最左边(最高)的位必须为 1，且负数是以 2 的补码来表示的。二进制负数补码的求法是：将二进制负数的绝对值，各位取反(0 变 1，1 变 0)再加 1。例如，下面以−65 为例。

−65 的绝对值+65：　01000001

各位取反：　10111110

加 1：　10111111(等于−65 的补码)

若直接将 10111111 的各位上 1 的“位权”值加起来，将发现其结果并不是−65。事实上，在计算机内如果一个二进制数其最左边的位是 1，则我们可以判定它为负数，并且是用补码表示。若要得到一个负二进制数的绝对值(称为真值)，只要重复上述方法，即各位取反，再加 1 就可得到真值。例如：

二进制值： 10111111 (−65 的补码)
各位取反： 01000000
加 1： 01000001 (+65 的补码)

微型计算机中的数都是用补码表示和运算的。采用补码的好处是加法和减法都可以统一为加运算。例如，将+65 的补码与−65 的补码相加，其结果将等于零。

```
  01000001   (+65 的补码)
+ 10111111   (−65 的补码)
(1) 00000000
```

8 个位均为 0，而最左边的进位 1 将自动丢弃。若有进位进入正负号位(最左面)，且此正负号位又有进位产生的话，即双高位都有进位，则结果是正确的。

微型计算机中二进制数的减法，是用加法实现的。方法是：将减数(补码)再求 2 的补码，即各位取反再加上 1，然后将此结果与被减数相加。例如：65−42=65+(−42)，−42 的绝对值二进制数表示为 00101010，所以，(−42)2 的补码表示为 11010110。

```
   65                    01000001
+(−42)                 + 11010110
   23                  (1) 00010111
```

其结果等于 23 是正确的。此次仍然有进位进入正负号位，而且也有最高位的进位产生。

若你对 2 的补码表示还不十分清楚，试考虑下面的问题。对于二进制数 00000001，试加上何值才能使它等于 00000000。就十进制而言，答案是−1。在二进制中，我们可以试用 11111111：

```
  00000001
+ 11111111
(1) 00000000
```

丢失进位 1，结果为 0 正确，即可得知二进制数 11111111 是−1 的 2 的补码，另有：

```
   0          00000000          00000000
− (+1)      − 00000001        + 11111111
  −1          11111111          11111111
```

下面是二进制数补码按递减顺序排列的情况。

+3	00000011
+2	00000010
+1	00000001
0	00000000
−1	11111111
−2	11111110
−3	11111101

1.1.4　十六进制数及运算

假设现在你想要观察存储器中某些存储单元的内容，想知道由相邻四个字节(也即两个字)所组成的二进制数值。由于四个字节是 32 个位的二进制数，如果用二进制数表示，显然位数太多，出现了难读、难写、难记的问题，现在分析如下：

二进制：　0101　1001　0011　0101　1011　1001　1100　1110

十进制：　5　9　3　5　11　9　12　14

由于有些数需要用到两个数字，如 11，12，14，所以我们扩展数字系统，使得 10=A，11=B，12=C，13=D，14=E，15=F。利用此种缩写方法，上述四个字节的内容就可以简洁地表示为：

59 35 B9 CE

这种数字系统使用了“数字”0 到 9、A 到 F，由于共有 16 个数字，所以我们称它为十六进制表示法。表 1-1 列出了从 0 到 15 的二进制、十进制以及十六进制的表示法。

表 1-1　二进制、十进制和十六进制表示法

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

汇编语言使用了十六进制的格式。经过汇编的目标程序也是以十六进制表示所有的地址、机器指令码以及常数。在使用 DOS DEBUG 帮助调试程序时，同样也是使用十六进制格式表示所有地址和存储单元内容的。

下面是一些十六进制算术运算例子。在十六进制中，是逢十六进位的。

```
   6        5        F        F       10       FF
 + 4      + 8      +  1     + F     + 10     +  1
   A        D       10       1E      20      100
```

(1) 十六进制的 20 等于十进制的 32，十六进制的 100 等于十进制的 256，而十六进制的 1000 则等于十进制的 4096。

(2) 本书有关数的表示，通常是采用十六进制，例如，4BH，其中，H 表示十六进制，或加前缀说明写成 Hex 4B；在二进制中写成 Binary 01001011 或 01001011B；十进制中则写成 75。以十进制表示时，通常前面不加前缀说明，后面也不加后缀说明。注意：在 DEBUG 状态下数、地址都默认为是十六进制，而不允许加后缀说明。

1.2 内存储器的管理方式

8086/8088 汇编语言对内存储器的管理是按段进行的，段(Segment)是存储器中的一块区域，它的最大长度为 64 KB(1 KB 等于 1024×8)。段的起始位置可以出现在存储器中的任一节边界位置(Paragraph Boundary)上，节边界位置是实际地址能被 16 整除的位置，例如，20000H，445F0H 等，最后一位总是 0H。

一般的程序均含有下列三个主要段。

1. 指令段(Code Segment)

指令段内含有将要被执行的机器指令。一般来说，第一条可执行的指令通常位于指令段的起始位置。指令段寄存器(CS)则指示出指令段的起始位置。

2. 数据段(Data Segment)

数据段内含有已定义的数据栏、常数、以及程序中所需要的工作区(即存放中间结果和结果的区域)。数据段寄存器(DS)指示出数据段的起始位置。

3. 堆栈段(Stack Segment)

在堆栈段内含有两种返回位置(即返回地址)，一种是程序返回系统的位置(地址)，另一种则为被调用的子程序返回其主程序所需要的位置(地址)。堆栈段寄存器(SS)指示出堆栈段的起始位置。

另外，还有一个段寄存器，称为附加段寄存器(ES)，它具有特殊的用途。

图 1-1 是 SS、DS 与 CS 寄存器的示意图，段与寄存器一般是以此顺序，但不一定非要按此顺序。

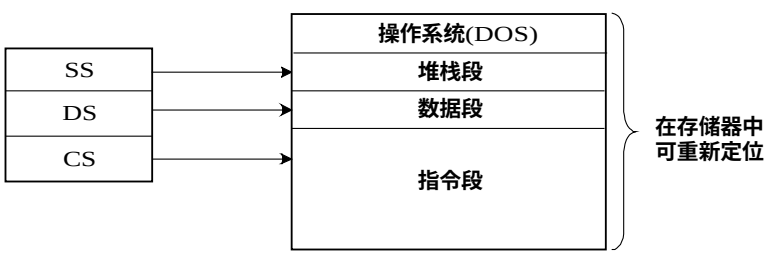


图 1-1 段与寄存器

图 1-1 中的三个寄存器分别含有每一个段的起始位置(地址)，且各段都以节边界位置开始，即实际地址的最低位是 Hex 0。在一个程序中，存储器的所有存储单元的位置(地址)都是以相对于段的起始位置(地址)的差距来确定的。此种相对于段起始位置的差距称为偏移量或偏移地址，本教材以后均采用偏移地址来描述这个差距值。差距的长度为两个字节(16 个位)，偏移地址所表示的范围可从 Hex 0000 到 Hex FFFF，亦即十进制的 0 到 65 535。所以在程序中出现的任何存储单元的位置(地址)，都是以段寄存器的值(段起始地址)与偏移地址二者共同确定的。例如，位于指令段的第一个字的偏移地址为 0000H，第二个字的偏移地址为 0002H，依次类推，最大的偏移地址为 FFFFH(即 65 535)。