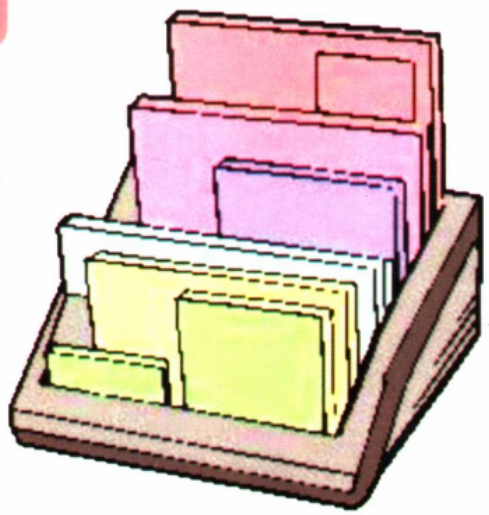
高等学校计算机基础教育课程系列教材

汇编语言程序设计

精 解 重 练　　快 捷 方 便

计算机基础知识
计算机的基本结构与组成
80x86的指令系统和寻址方式
80x86汇编语言程序
汇编语言程序设计
子程序设计
高级汇编语言技术
输入/输出程序设计
中断调用程序设计
Win 32 汇编语言编程



★ 每章后附有大量习题和上机练习

HUIBIANYUYAN CHENGXUSHEJI

王彬华　刘盛军　编著

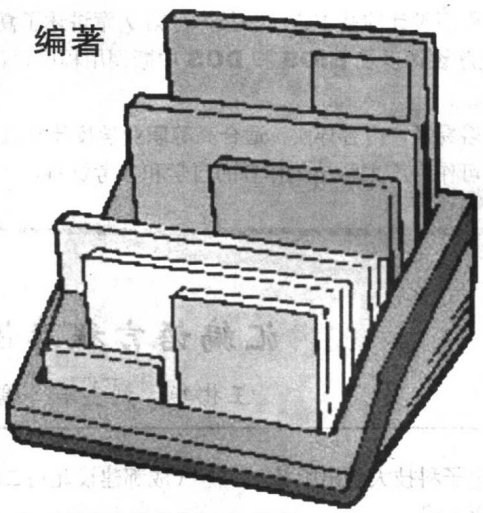
电子科技大学出版社

高等学校计算机基础教育课程系列教材

汇编语言程序设计

精 解 重 练　　快 捷 方 便

王彬华　刘盛军　编著



HUIBIANYUYAN CHENGXUSHEJI

电子科技大学出版社
DIANZIKEJIDAXUECHUBANSHE

内 容 简 介

本书在介绍 Win32 汇编语言指令和基本语法的基础上，重点介绍如何使用汇编语言和 Windows SDK API 开发 Win32 应用程序，同时还探讨了汇编语言和 Visual C++的混合编程、驱动程序的开发、COM 组件的使用和开发、数据库开发、代码优化、异常处理以及程序跟踪调试等问题。

对于每个主题，书中都提供了开发要领及应用的实例和技巧，本书主要面向具备一定汇编语言基础和初步的 Win32 编程经验的用户。

图书在版编目(CIP)数据

Win32 汇编语言实用教程/北银科文，冉林仓编著.—北京：清华大学出版社，2004
ISBN 7-302-07954-4
(高等院校计算机教育系列教材)

Ⅰ.W… Ⅱ.①北…②冉… 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 001608 号

出 版 者：清华大学出版社　　地　　址：北京清华大学学研大厦
　　http://www.tup.com.cn　　邮　　编：100084
　　社 总 机：010-62770175　　客户服务：010-62776969
组稿编辑：应　勤
文稿编辑：桑任松
封面设计：陈刘源
印 刷 者：北京市人民文学印刷厂
装 订 者：三河市化甲屯小学装订二厂
发 行 者：新华书店总店北京发行所
开　　本：185×260　印张：22.25　字数：534 千字
版　　次：2004 年 2 月第 1 版　2004 年 2 月第 1 次印刷
书　　号：ISBN 7-302-07954-4/TP·5777
印　　数：1～5000
定　　价：29.00 元

前　言

我国高等教育的一个重要组成部分——高等职业教育，在近年来得到了长足的发展。探索高等职业教育领域中计算机学科的教材体系和课程改革是计算机教育工作者的一个重要课题。

本文的作者从事计算机工作多年，有着丰富的实践经验。结合当前微型计算机软硬件新技术的发展趋势及高等职业教育的特殊要求，编写了此书。

使用者在使用本教材实施教学时，可以根据教学计划的安排适当调整教学课时。全文选用时，参考学时为 80 学时（含上机），其中课堂授课不多于 70 学时；侧重程序设计时，参考学时为 60 学时（含上机），其中第 1～2 章 10 学时，第 3～7 章 50 学时，第 8～10 章选讲 20 学时。要学习汇编语言程序设计，本书只起到抛砖引玉的作用，希望读者多上机操作、实践。

本书由创世纪工作室王彬华、刘盛军组织编写，创世纪工作室的许多人员也做了许多工作，在此一并表示诚挚的感谢！

本书是为高等职业教学专科以上层次编写的一本实用教材，亦可作为广大科技工作者的自学和参考资料。由于时间仓促，水平有限，本书中难免有错误和不妥之处，希望广大读者批评指正。

编　者

2004 年 1 月

目　录

第1章 ⟹ 计算机基础知识

第 2 章 计算机的基本结构与组成

第 3 章 80x86 的指令系统和寻址方式

第 4 章 80x86 汇编语言程序

第 5 章　汇编语言程序设计

第 6 章 子程序设计

第 7 章 高级汇编语言技术

第 8 章　输入/输出程序设计

第 9 章　中断调用程序设计

第 10 章 Win 32 汇编语言编程

第1章　计算机基础知识

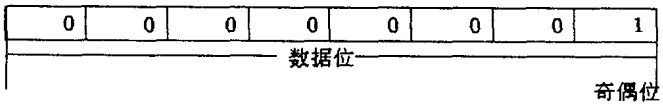
1.1　位 与 字 节

1.1.1　位

计算机存储的基本构造单元是“位”（bit）。一个位可能是“关闭”（off）状态，它的值被看作是 0；或者它是“开通”（on）状态，它的值则被看作是 1。单个的位提供不了更多的信息，但一些位要是组合在一起就会有意想不到的作为。

1.1.2　字节

一组 9 个相关的位称为字节，它代表内存储器和外部设备的一个存储单元。每个字节由 8 个数据位和 1 个奇偶位组成：



8 个数据位为二进制运算和诸如字母 A 与星号（*）这样一些字符的表示打下了基础。字节中的 8 位允许有 256（2^8）个“开”～“关”状态的不同组合，从所有都为“关”状态（00000000）到所有都为“开”状态（11111111）。例如，字母 A 的位表示为 01000001，而星号的位表示则是 00101010，至于这些位的值是多少并不需要你去记它。

根据奇偶校验规则，在每个字节中，1 的个数必须总是奇数。因为字母 A 中有两个位值是 1，所以处理器按奇偶校验规则自动地把奇偶位置为 1（01000001-1）。类似地，由于星号包含了 3 个 1，处理器按奇偶校验规则就把奇偶位置为 0（00101010-0）。当指令访问内存储器中的一个字节时，处理器在检查这个字节的奇偶性。如果奇偶性是偶数，系统便会假定“丢”了一位并显示出错信息。奇偶错可能是由硬件故障或是电气干扰造成的，总之是偶发的事件。

怎么做才能使计算机“知道”位值 01000001 是代表字母 A 呢？当你在键盘上键入 A 时，系统传送这个特定键盘信号到存储器并设置一个字节（在你的程序的一个单元内），其位值为 01000001。你可以按需要在存储器范围内传送字节的内容，还可以把字母 A 打印出来或显示在屏幕上。

处理器奇偶校验是一种自动的硬件功能，我们不必去管它。一个字节中的各位从右到左是按 0 到 7 编号的，如字母 A 表示如下：

位内容(A)：	01000001
位编号：	76543210

相关字节，程序可以把一个或多个相关字节的组看作是一个数据单元，比如时间或距

离。定义成特定值的一组字节，通常是作为数据项或字段来识别的。处理器还支持一些特定的数据大小：

字：2 字节（16 位）数据项。

双字：4 字节（32 位）数据项。

四字：8 字节（64 位）数据项。

小段：16 字节（128 位）区。

千字节（KB）：数 2^{10} 等于 1 024（正好是 K 值）。因此，640KB 就是 640×1 024=655 360 字节。

兆字节（MB）：数 2^{20} 等于 1 048 576，MB 是为 1 兆字节。

在一个字中，各位从右到左的编号是 0 到 15。对于字母 PC 则表示成最左边的字节为 P（01010000），而最右边的字节为 C（01000011）。

位内容(A):	0	1	0	1	0	0	0	0	0	1	0	0	0	0	1	1
位编号:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

在存储器中的每个字节都有一个惟一的地址。第一个字节在最低的存储器单元中，编号为 0，第二个字节编号为 1，等等。

1.2 二进制数系统

1.2.1 二进制数

由于计算机只能辨别 0 位和 1 位，所以它只能工作在以 2 为基数的称为二进制的计数系统中。实际上，二进制位“bit”（“位”）是“Binary Digit”的缩写。

位的集合可以表示任何数字的值。 二进制数的值是由为 1 的位用其相关位置所决定的。如同十进制数一样，位置代表从右到左幂次的上升（但这是 2 的幂，不是 10 的幂）。在以下的 8 位数中，所有位均置为 1：

位的值:	1	1	1	1	1	1	1	1
位置的值:	128	64	32	16	8	4	2	1
位编号:	7	6	5	4	3	2	1	0

最右边的位设其值为 1（2^0），向左的下一位设其值为 2（2^1），再下一位的值为 4（2^2），以此类推。在这种情况下，二进制数的值是 1+2+4+8+16+32+64+128=255（或 2^8-1）。

让我们考虑一个二进制数 01000001 的值：

位的值:	0	1	0	0	0	0	0	1
位置的值:	128	64	32	16	8	4	2	1

你可以计算它的值 1+64=65。但是你可能会想到位置 01000001 也是字母 A。的确，01000001 既可以代表 65 这个数，也可以代表字母 A：

• 如果程序是为算术运算目的而定义与使用数据的，那么位置 01000001 就是表示一个二进制数，它等于十进制数目 65。

• 如果程序是为描述目的而定义与使用数据的，如标题，那么 01000001 就代表一个字母字符。

当你开始编程的时候，由于要为一个特定目的而定义与使用每个数据项（即为运算目的而用的运算数据以及为显示输出而用的描述数据），所以你就会更清楚地看到这种区别。实际上，两种用法很少会发生混淆。

二进制数不受 8 位限制。使用 16 位（或 32 位）体系结构的处理机会自动处理 16 位（或 32 位）数。对于 16 位机，所能提供的值在 2^{16}-1（即 65 535）之内；而对于 32 位机，所能提供的值可以达 2^{32}-1 即（4 294 967 295）。

1.2.2　二进制算术运算

因为微型计算机只以二进制格式完成算术运算，所以一个汇编语言程序员必须熟悉二进制格式与二进制加法。以下 4 个例子可以简单说明二进制加法：

```
  0      0      1      1
+ 0    + 1    + 1    + 1
---    ---    ---    ---
  0      1     10     11
```

最后两个例子表明有一个 1 的进位到下一个(左边)位置。现在，让我们把位置 01000001 和 00101010 相加。我们是把字母 A 和一个星号相加吗？不是的，这时它们代表的是十进制值 65 和 42。

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

为了检验二进制和 01101011 实际上就是 107，可以把是 1 的位的值相加。作为另一个例子，我们把十进制值 60 与 53 以及它们的二进制等效值相加：

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

再一次检查二进制的和，实际上就是 113。

1.2.3　负二进制数

负二进制数：带符号的二进制数（即一种用于算术运算的数），如果它的最左边的位是 0，它就是正数；反之，带符号的负二进制数的最左边的位是 1。但是，表示一个二进制数是负的，不能简单地把最左边的位置成 1 就行了，如把 01000001（+65）变成 11000001，而应该把负二进制值用二进制补码表示法来表示。也就是说，把一个二进制数表示成负

数的规则是：位值求反并加 1。例如，利用这一规则求出 01000001（或 65）的二进制补码：

数+65：	01000001
按位求反：	10111110
加 1：	+ 1
数−65	10111111

当带符号的二进制数最左边的位是 1 时，它是负数，并且处理器会对它做相应的处理。但是，如果你想用加 1 的办法求得三进制数 10111111 的十进制值，你将不会得到 65。为了求得一个负二进制数的绝对值，需要采用二进制补码规则，也就是按位求反再加 1：

数 −65	10111111
按位求反：	01000000
加 1：	+ 1
数 +65	01000001

为了说明这种做法是正确的，+65 与−65 的和应该是零。让我们试一下：

+ 65	01000001
+（−65）	+ 10111111
00	（1）00000000

注意，8 位数以外还有一位溢出的 1。在这个和中，8 位的值是全 0，而在左边溢出的 1 则被丢弃了。但是，因为既有符号位的进位，又有进位输出，所以认为结果是正确的。

为了处理二进制减法，需要把进行减法的数转换成二进制补码格式，再把它们相加。例如，65 减 42。42 的二进制表示是 00101010，它的二进制补码是 11010110；把−42 加上 65 就变成这样：

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

结果 23 是正确的。注意，又有一个有效的进位输入和符号位输出。

假如对于二进制补码表示法还没有弄清楚，可以考虑以下问题：你必须把一个什么样的值加到二进制数 00000001 上，才会使它等于 000000000 呢？对于十进制数而言，答案将是−1。00000001 的二进制补码是 11111111，所以+1 与−1 相加等于零：

1	00000001
+（−1）	+ 11111111
结果	（1）00000000

忽略进位的 1，你会看到二进制数 11111111 等于十进制数−1。你会看到二进制数按值减少的一种模式：

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

−2	11111110
−3	11111101

实际上，在负二进制数中为 0 的位可以指出这个数的绝对值：把每个是 0 的位置看成是 1，把这些值求和并加 1。

1.3　十六进制数表示法

我们知道，在计算机内部，数的运算和存储都是采用二进制的。但是，二进制数对于人的阅读、书写及记忆很不方便的。十进制数虽然是人们最熟悉的一种进位计数制，但它与二进制数之间并无直接的对应关系。为了便于人们对二进制数的描述，应该选择一种易于与二进制数相互转换的数制。显然，使用 2n 作为基数的数制是能适合人们的这种要求的，常用的有八进制数和十六进制数，我们在这里主要介绍十六进制数。

1.3.1　十六进制的表示

尽管一个字节可以包含 256 种位的任意组合，但它们当中许多是无法按标准的 ASCII 字符显示或打印的。这类字符的例子包括这样一些操作的位组合，比如制表符（Tab）、回车符（Enter）、换页符（Form Feed）以及换码符（Escape）。因此，计算机设计人员开发了一种二进制数据的简化方法，把每个字节分成两半并把每半个字节的值表示出来。

假设你想看看存储器中四个邻接字节（一个双字）的二进制值的内容。考虑以下四个字节，它们分别表示为二进制与十进制格式。

二进制：	0101	1001	0011	0101	1010	1001	1100	1110
十进制：	5	9	3	9	10	9	12	14

由于十进制 10，12 和 14 每个都需要两个数字，让我们采用扩充记数系统，使 10=A，11=B，12=C，13=D，14=E，15=F。这样，记数系统就包括了“数字”0 到 F，由于有 16 个这样的数，所以这一系统便被称为十六进制（或 Hex）表示法。下面就是修正过的简化数，它表示出了上面所给字节的内容：

二进制：	0101	1001	0101	1001	0101	1001	0101	1001
十进制：	5	9	5	9	A	9	C	E

表 1.1 给出了十进制数 0～15 及与其等值的二进制值与十六进制值。

汇编语言大量使用十六进制格式。一个汇编程序的列表文件中，所有地址、机器码指令以及数据常数的内容都是以十六进制格式表示的。为了调试程序，你可以使用 DEBUG 程序，它是按十六进制格式显示地址与字节内容的。

表 1.1 二进制、十进制及十六进制表示法

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

1.3.2 十六进制数的运算

十六进制数的运算可以采用先把该十六进制数转换为十进制数，经过计算后再把结果转换为十六进制数的方法，但这样做是比较繁琐。其实，只要按照逢十六进一的规则，直接用十六进制数来计算也是很方便的。

十六进制数加法：

当两个一位数之和 S 小于 16 时，与十进制数同样处理；如两个一位数之和 S≥16 时，则应该用 S-16 及进位 1 来取代 S。例如：

```
  05C3H
+ 3D25H
-------
  42E8H
```

十六进制数的减法也与十进制数类似，够减时可直接相减，不够减时服从向高位借 1 为 16 的规则。例如：

```
  3D25H
- 05C3H
-------
  3762H
```

十六进制的乘法可以用十进制数的乘法规则来计算，但结果必须用十六进制数来表示。例如：

```
   05C3H
×  00ABH
--------
   3F61
+ 399E
--------
  3D941H
```

十六进制数的除法可以根据其乘法和减法规则处理，需要时读者可自行处理，这里不再赘述。

为了指明在汇编语言程序中的十六进制数，在这个数的后面你要写上一个“H”，如 25H（十进制数的 37）。汇编语言要求十六进制数永远是用十进制数 0～9 作为开始，所以就要把 B8H 写成 0B8H。在本书中，十六进制的值是用数的前面放一个“hex”或者数后跟一个“H”来表示的（如 hex 4C 或 4CH）；二进制的值用放一个“binary”在数的前面或

者数的后面跟一个“B”来表示（如 binary 01001100 或 01001100B）；而十进制值就简单地用一个数表示（如 76）。如果根据上下文能够明确地确定该基数值是十进制还是十六进制，也可以不必像上述那样标得那么清楚。

1.4　不同基数的数之间的转换

1.4.1　二进制数和十进制数之间的转换

1. 二进制数转换为十进制数

各位二进制数码乘以与其对应的权之和即为与该二进制数相对应的十进数。例如：

$1011100.10111B=2^6+2^4+2^3+2^2+2^{-1}+2^{-3}+2^{-4}+2^{-5}=92.71875D$

2. 十进制数转换为二进制数

十进制数转换为二进制数的方法很多，这里只说明比较简单的降幂法及除法两种。

• 降幂法

首先写出要转换的十进制数，其次写出所有小于此数的各位二进制权值，然后用要转换的十进制数减去与它最相近的二进制权值，如够减则减去并在相应位记以 1；如不够减则在相应位记以 0 并跳过此位；如此不断反复，直到该数为 0 为止。

【例 1.1】N=117D，小于 N 的二进制权为：

	64	32	16	8	4	2	1
对应的二进制数是 1	1	1	0	1	0	1	

计算过程如下：

$117-2^6=117-64=53$　　$(a_6=1)$

$53-2^5=53-32=21$　　$(a_5=1)$

$21-2^4=21-16=5$　　$(a_4=1)$

　　$(a_3=0)$

$5-2^2=5-4=1$　　$(a_2=1)$

　　$(a_1=0)$

$1-2^1=1-1=0$　　$(a_0=0)$

所以　N=117D=11110100B

【例 1.2】N=0.8125D，小于此数的二进制权为：

	0.5	0.25	0.125	0.0625
对应的二进制数是 1	1	0	1	

计算过程如下：

$0.8125-2^{-1}=0.8125-0.5=0.3125$　　$(b_1=1)$

$0.3125-2^{-2}=0.3125-0.25=0.0625$　　$(b_2=1)$

　　$(b_3=0)$

$0.0625-2^{-4}=0.0625-0.0625=0$　　$(b_4=1)$