Broadview
WWW.BROADVIEW.COM.CN

立足实用·面向教学

# Windows
# 汇编语言程序设计教程

谭毓安 张雪兰 编著

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

# Windows汇编语言程序设计教程

谭毓安　张雪兰　编著

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

## 内 容 简 介

本书以 Windows 操作系统和 Intel 80x86/Pentium 系列 CPU 为背景，全面系统地介绍了 32 位保护模式下的汇编语言程序设计，包括 Windows 编程环境、Intel 80x86/Pentium 指令系统、MASM 的伪指令与操作符等知识，以及分支、循环、子程序、C/C++混合编程、上机调试过程等程序设计方法。本书的核心是 Windows 下的 32 位保护模式编程，摒弃了过时的 16 位 DOS 实模式编程环境，与当前的软件开发和微机应用环境结合紧密。

本书不但能帮助读者顺利地掌握汇编语言程序设计方法，而且能够提高读者的 C/C++编程水平。书中的程序具有很强的实用性，强调 C/C++和汇编语言的联合编程能力，通过这些实例与其他课程相呼应，有利于各门课程之间的融会贯通。

本书适合作为高等院校汇编语言程序设计相关课程的教材或教学辅导书，也可作为希望掌握 Windows 汇编程序设计的中高级程序开发人员的自学参考书。



**图书在版编目（CIP）数据**

Windows 汇编语言程序设计教程 / 谭毓安，张雪兰编著. —北京：电子工业出版社，2005.4
ISBN 7-121-00986-2

I. W… II. ①谭… ②张… III. 汇编语言—程序设计—教材 IV. TP313

中国版本图书馆 CIP 数据核字（2005）第 014695 号

责任编辑：张 毅 zhangyi @ phei.com.cn
印 刷：北京东光印刷厂
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
经 销：各地新华书店
开 本：787×1092 1/16 印张：27 字数：670 千字
印 次：2005 年 4 月第 1 次印刷
印 数：6000 册 定价：32.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：（010）68279077。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。

# 前　言

汇编语言是执行效率最高、功能最强的一种程序设计语言，它能够直接控制计算机硬件，并最大限度地发挥硬件的能力。在对程序的执行时间和占用空间要求很高的场合，必须使用汇编语言才能满足要求。汇编语言还可以与高级语言进行混合编程，以发挥各自的优势。通过学习汇编语言，有助于理解操作系统和应用程序的运行原理，准确地分析程序错误。它也是掌握加密/解密技术、病毒蠕虫机理、剖析关键代码等高级技术的必备基础。

《汇编语言程序设计》是一门程序设计课程，然而目前许多教材仍然基于 DOS 系统和 8086/8088 之上，与十几年前的教材相比，仅仅增加了 32 位指令和保护模式的介绍，而核心内容仍然是 DOS 下的 16 位实模式编程。其中的 BIOS/DOS 功能调用、64KB 内存分段结构、上机环境、16 位 C/汇编混合编程等已经严重落后于当前的应用环境，与国内外主流的软件开发需求严重脱节。我们认为，随着 DOS 逐步退出操作系统的历史舞台，这门课程的基础知识和实践环节必须与时俱进，需要从 DOS 下的 16 位实模式编程过渡到 Windows 下的 32 位保护模式编程。

本书的编写就是在这样的背景下完成的。本书选择了 Windows 操作系统和 32 位 Pentium 微机作为平台，介绍汇编语言的基础知识和应用，达到与当前软件开发和应用环境紧密结合的目的。本书有以下几个特点。

（1）从 DOS 过渡到 Windows。不再介绍过时的 BIOS/DOS 功能调用和 DOS 执行环境，取而代之的是 Windows 的 API 和 C 库函数，书中所有的例子都是在 Windows 保护模式环境下运行的 32 位程序。

（2）实践性强。将实践环节提前，将 100 多条指令贯穿于全书，全部程序都能上机调试运行，使读者能在实践中掌握汇编语言程序设计并熟悉 Pentium 的指令系统。尤其是将 Visual C/C++作为汇编的开发环境，极大地方便了汇编程序的编写、调试和运行。将各种指令和伪操作等内容分解到全书中，逐步接触并使用这些指令，会达到循序渐进的效果。

（3）与 C/C++紧密结合。C/C++语言是目前的主流开发语言，往往也是汇编语言的先修课程。书中的部分 C/C++程序与汇编程序相对照，在汇编程序中也调用了 C 库函数。这不但有利于对汇编语言的掌握，而且能够提高 C/C++程序的编程水平，理解和解决在 C/C++程序开发过程中遇到的结构对齐、字节序、程序运行效率、堆栈溢出、内存覆盖、系统崩溃、访问违例等问题。C/C++和汇编语言联合编程能够应用于当前实际的软件开发过程，具有很强的实践指导性。

（4）介绍了保护模式的运行机制。在介绍保护模式时，使用了大量的实例以帮助读者更好地理解描述符、页表、门、任务等复杂概念，其中还介绍了一个 Windows 内核驱动程序。通过对 Pentium 保护模式的学习，能够更透彻地理解虚拟内存管理、进程环境等其他相关知识。

（5）采用典型编程实例。在示例程序中采用了指针、数组、结构、链表等数据结构，以及冒泡排序、快速排序、折半查找、加密/解密等算法，既体现了汇编语言的编程特点，又具有很强的实用性。

（6）知识全面。增加了 MASM 6.1 版本的汇编高级语法等，并介绍了 DLL、C 库函数、Windows API 等内容。

（7）适合自学。对部分不能在课堂上讲授的内容，可安排自学。在编写过程中，采用循序渐进的叙述方法，配以大量的实例来帮助读者对这些内容进行理解。书中的许多习题都是以例子程序为基础的，可达到举一反三的效果。

Pentium 指令系统中还包括浮点运算、MMX 多媒体扩展、数据流 SIMD 扩展（SSE、SSE2）等部分，考虑到它们的使用范围较窄，由于篇幅所限，本书没有介绍这方面的内容，感兴趣的读者可以参考其他书籍。

全书共分 10 章。第 1，2，3 章由张雪兰教授编写，其余各章由谭毓安副教授编写。7.4 节、7.5 节、8.2 节、8.5 节及第 10 章属于较高要求的内容，在教学时可以根据情况进行删减。

在本书的编写过程中，不仅融合了作者多年来讲授汇编语言程序设计课程的经验，还请多位老师和同学阅读了书稿，提出了改进意见，在此深表谢意。同时，本书还参考了国内外的众多教材和资料，在此谨对这些教材和资料的原创者致谢。

对书中的错误和不当之处，敬请广大读者批评指正。作者的电子邮箱是：victortan@yeah.net，欢迎通过电子邮件方式向本人索取本书的 PowerPoint 讲稿和其他教学资料。

编　者

# 目　录

# 第1章 基础知识

汇编语言是符号化的、面向机器的语言，在学习汇编语言编程之前，需要了解计算机是如何表示、存储信息的。这些知识对于汇编语言的学习十分重要，是学习后续章节的重要基础。

## 1.1 常用数制及其相互转换

人们习惯使用十进制来表示数字，但在计算机中使用的是二进制，这是因为数字电路能够高效地对二进制数字进行计算、存储、传输等操作。然而，为了阅读和书写方便，又经常使用十六进制，例如，二进制数字 $0011011011001101_2$ 有 16 个数位，如果用十六进制，只需要 4 个数位，即 $36CD_{16}$。

**1. 十进制**

在十进制（Decimal）表示中，每个数位的范围是 0～9 十个数字。从低到高，数位又有个位、十位、百位、千位、万位等，例如：

$$466_{10} = 4\times10^2+6\times10^1+6\times10^0$$

**2. 二进制**

在二进制（Binary）表示中，每个数位的范围是 0、1 两个数字，例如：

$$\begin{aligned}111010010_2 &= 1\times2^8+1\times2^7+1\times2^6+0\times2^5+1\times2^4+0\times2^3+0\times2^2+1\times2^1+0\times2^0\\ &= 256_{10}+128_{10}+64_{10}+16_{10}+2_{10}\\ &= 466_{10}\end{aligned}$$

这就是从二进制数到十进制数的转换过程。而从十进制数转换到二进制数，方法是将这个数除以 2，每次得到的商继续进行除法，一直到商为 0 时结束。每次除法得到的余数按照顺序即构成二进制数，第 1 次得到的余数为最低位，依次类推。过程如下：

$$\begin{aligned}466_{10}\div2_{10} &= 233_{10}\ 余\ 0_{10}\\ 233_{10}\div2_{10} &= 116_{10}\ 余\ 1_{10}\\ 116_{10}\div2_{10} &= 58_{10}\ 余\ 0_{10}\\ 58_{10}\div2_{10} &= 29_{10}\ 余\ 0_{10}\\ 29_{10}\div2_{10} &= 14_{10}\ 余\ 1_{10}\\ 14_{10}\div2_{10} &= 7_{10}\ 余\ 0_{10}\\ 7_{10}\div2_{10} &= 3_{10}\ 余\ 1_{10}\\ 3_{10}\div2_{10} &= 1_{10}\ 余\ 1_{10}\end{aligned}$$

$$\begin{aligned}111010010_2\div10_2 &= 11001001_2\ 余\ 0_2\\ 11101001_2\div10_2 &= 1100100_2\ 余\ 1_2\\ 1110100_2\div10_2 &= 110010_2\ 余\ 0_2\\ 111010_2\div10_2 &= 11001_2\ 余\ 0_2\\ 11101_2\div10_2 &= 1100_2\ 余\ 1_2\\ 1110_2\div10_2 &= 110_2\ 余\ 0_2\\ 111_2\div10_2 &= 11_2\ 余\ 1_2\\ 11_2\div10_2 &= 1_2\ 余\ 1_2\end{aligned}$$

$1_{10}+2_{10}$ = $0_{10}$ 余 $1_{10}$　　$1_2+10_2$ = $0_2$ 余 $1_2$

结果为 $111010010_2$。

右边进行的二进制除法与左边的十进制除法是等价的。

### 3. 十六进制

在十六进制（Hexadecimal）表示中，每个数位的范围是 0～15 十六个数字，使用 0～9、A～F 来表示这十六个数字，其中 A 代表 10，B 代表 11，C 代表 12，D 代表 13，E 代表 14，F 代表 15，如表 1-1 所示。

表 1-1　十六进制的数位

| 十六进制 | 十进制 | 二进制 | 十六进制 | 十进制 | 二进制 |
|---|---|---|---|---|---|
| $0_{16}$ | $0_{10}$ | $0000_2$ | $8_{16}$ | $8_{10}$ | $1000_2$ |
| $1_{16}$ | $1_{10}$ | $0001_2$ | $9_{16}$ | $9_{10}$ | $1001_2$ |
| $2_{16}$ | $2_{10}$ | $0010_2$ | $A_{16}$ | $10_{10}$ | $1010_2$ |
| $3_{16}$ | $3_{10}$ | $0011_2$ | $B_{16}$ | $11_{10}$ | $1011_2$ |
| $4_{16}$ | $4_{10}$ | $0100_2$ | $C_{16}$ | $12_{10}$ | $1100_2$ |
| $5_{16}$ | $5_{10}$ | $0101_2$ | $D_{16}$ | $13_{10}$ | $1101_2$ |
| $6_{16}$ | $6_{10}$ | $0110_2$ | $E_{16}$ | $14_{10}$ | $1110_2$ |
| $7_{16}$ | $7_{10}$ | $0111_2$ | $F_{16}$ | $15_{10}$ | $1111_2$ |

从十六进制数到十进制数的转换过程为：

$$1D2_{16} = 1\times16^2+13\times16^1+2\times16^0$$
$$= 256_{10}+208_{10}+2_{10}$$
$$= 466_{10}$$

从十进制数转换到十六进制数，方法是将这个数除以 16，每次得到的商继续进行除法，一直到商为 0 时结束。每次除法得到的余数按照顺序即构成十六进制数，第 1 次得到的余数为最低位，依次类推。过程如下：

$402_{10}+16_{10}$ = $29_{10}$ 余 $2_{10}$
$29_{10}+16_{10}$ = $1_{10}$ 余 $13_{10}$
$1_{10}+16_{10}$ = $0_{10}$ 余 $1_{10}$

$1D2_{16}+10_{16}$ = $1D_{16}$ 余 $2_{16}$
$1D_{16}+10_{16}$ = $1_{16}$ 余 $D_{16}$
$1_{16}+10_{16}$ = $0_{16}$ 余 $1_{16}$

结果为 $1D2_{16}$。

### 4. 十六进制与二进制的相互转换

从表 1-1 可知，1 个十六进制数位对应于 4 个二进制数位，因此十六进制数和二进制的转换可以参照表 1-1 直接进行，而不必使用乘除法。从十六进制数到二进制数转换时，将每 1 位十六进制数按照表 1-1 中的对应关系转换到对应的 4 个二进制数位即可，例如：

$1D2_{16}$= 0001 1101 $0010_2$　($1_{16}$= $0001_2$, $D_{16}$= $1101_2$, $2_{16}$= $0010_2$)

从二进制数到十六进制数转换时，将每 4 个二进制数位按照表 1-1 中的对应关系转换到

对应的1个十六进制数即可，注意要从最低位开始，按4位一组进行转换，高位不足的部分补0。例如：

$111010010_2 = 0001\ 1101\ 0010_2 = 1D2_{16}$ ($0001_2 = 1_{16}$, $1101_2 = D_{16}$, $0010_2 = 2_{16}$)

## 1.2 存储器内的数字表示

信息在计算机中是按字节为单位存储的，而内存则按字节给每一个单元赋予一个地址。除字节之外，还有字、双字等单位。下面就来具体介绍一下。

### 1.2.1 存储器

存储器是计算机的记忆部件，用来存放程序和数据。按所在的位置，存储器可以分成内存（主存储器）和外存（辅助存储器）。

**1. 内存和外存**

内存存放当前正在执行的程序和使用的数据，CPU可以直接存取。它由半导体存储器芯片构成，其存取速度快，但成本高，容量比外存小，断电后内存中的内容会丢失。某些计算机带有后备电池，在断电后为内存提供电源，因此可在断电后维持内存的内容。

外存可用来长期保存大量的程序和数据，CPU需要通过I/O接口访问，例如软盘、硬盘或CD-ROM、DVD-ROM等。与内存相比，其成本低、容量大，但存取速度较慢。

**2. 存储器容量**

内存和外存的容量都是以字节（Byte）为单位来描述的，字节也是内存最基本的存储单元。1字节包括8个二进制位。

在描述比较大的存储容量时，又经常使用KB(KiloByte)、MB(MegaByte)、GB(GigaByte)、TB、PB、EB等单位。它们之间的关系为：

$1KB = 2^{10}B = 1\ 024B$

$1MB = 2^{20}B = 1\ 024KB = 1\ 048\ 576B$

$1GB = 2^{30}B = 1\ 024MB = 1\ 073\ 741\ 824B$

$1TB = 2^{40}B = 1\ 024GB = 1\ 099\ 511\ 627\ 776B$

$1PB = 2^{50}B = 1\ 024TB = 1\ 125\ 899\ 906\ 842\ 624B$

$1EB = 2^{60}B = 1\ 024PB = 1\ 152\ 921\ 504\ 606\ 846\ 976B$

目前，主流微机的内存容量已超过64MB，硬盘容量也达到了200GB左右。磁盘阵列中可包括上百个硬盘，达到了TB数量级的容量，甚至可达到PB级。

这里有一个误导，有的存储器生产厂家采用这样的公式：

$1KB = 10^3B = 1\ 000B$

$1MB = 10^6B = 1\ 000KB = 1\ 000\ 000B$

$1GB = 10^9B = 1\ 000MB = 1\ 000\ 000\ 000B$

这就导致这些产品的实际存储容量达不到这些厂家标称的数值。例如：某硬盘的标称容

量为 36GB，其容量为 36 420 074 496B（字节）。按照 1GB = 1 000 000 000B 计算：

```
36 420 074 496/1 000 000 000 = 36.420074496
```

容量约为 36.42GB。但是，如果按照 1GB = 1 073 741 824B 计算：

```
36 420 074 496/1 073 741 824 = 33.91883754730224609375
```

实际容量应约为 33.92GB。

### 3. 内存地址

对于计算机系统中的内存，可以由 CPU 直接读取和写入数据。内存中的每一个字节都有一个惟一的地址。CPU 在读写这个字节时，首先给出该字节的内存地址，然后内存将存储单元中的数据读出送给 CPU，或者将 CPU 给出的数据写入这个地址对应的存储单元。地址的作用就是惟一地标识每一个存储单元。这里的地址指的是物理地址（Physical Address），在本书后面的内容中还会介绍逻辑地址（Logical Address）。

图 1-1 表示出存储器中部分存储单元的地址及内容。这里的地址和内容都用十六进制表示。

| 地址 | 内容 |
|---|---|
| 0E4E1000H | 25H |
| 0E4E1001H | E0H |
| 0E4E1002H | D6H |
| 0E4E1003H | 0EH |
| 0E4E1004H | 25H |
| 0E4E1005H | 40H |
| 0E4E1006H | FCH |
| 0E4E1007H | 00H |
| 0E4E1008H | 00H |
| 0E4E1009H | 00H |
| 0E4E100AH | 00H |
| 0E4E100BH | 00H |
| 0E4E100CH | 25H |
| 0E4E100DH | 60H |
| 0E4E100EH | FCH |
| 0E4E100FH | 00H |

图 1-1　内存的地址及内容

在一些工具软件中，常以下面这种方式紧凑地显示存储器的内容：

```
0E4E1000 25 E0 D6 0E 25 40 FC 00-00 00 00 00 25 60 FC 00  %...%@......% ..
0E4E1010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0E4E1020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0E4E1030 00 00 00 00 00 00 00 00-00 00 00 00 25 20 EA 0E  ............% ..
0E4E1040 00 00 00 00 67 40 27 0F-67 E0 7A 00 00 00 00 00  ....g@'.g.z.....
0E4E1050 67 30 E8 0E 25 00 DB 0E-00 00 00 00 00 00 00 00  g0..%...........
0E4E1060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0E4E1070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
```

最左边的8位十六进制数表示存储器的地址。每一行可显示16字节的内容，前8字节与后8字节的中间用分隔线隔开。这里全部使用十六进制格式，为了节省显示位置，这些数字和地址后面都没有加表示十六进制数的字母H。

从中可知，地址0E4E1000H中的内容为25H，0E4E1001H中的内容为E0H，0E4E1002H中的内容为D6H，依次类推。

### 4. 字节、字、双字

1个字节中有8个二进制位，字节是内存的最基本存储单元。这8个二进制位按照从高到低的顺序称为第7, 6, …, 1, 0位。第7位是最高位，简写为MSB（Most Significant Bit）。第0位是最低位，简写为LSB（Least Significant Bit）。如图1-2所示，25H是1个字节，它的8个二进制位从高到低依次是0, 0, 1, 0, 0, 1, 0, 1，最高位为0，最低位为1。

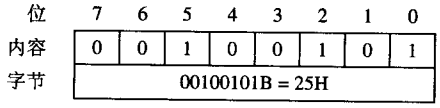


图1-2 字节和二进制位

16位及16位以上的CPU可以在一次操作过程中处理更多的二进制位。例如，80386/80486/Pentium系列处理器可以进行32位的计算，8086/8088/80286处理器可以进行16位的计算。因此，就需要使用字和双字的概念。

1个字包括2个字节，也就是16个二进制位。这两个字节分别称为高字节（High Order Byte）和低字节（Low Order Byte）。如图1-3所示，6025H是1个字，它的高字节是60H，低字节是25H。这16个二进制位按照从高到低的顺序称为第15, 14, …, 1, 0位。第15～8位是高字节，第7～0位是低字节。第15位是最高位（MSB），第0位是最低位（LSB）。6025H的最高位为0，最低位为1。

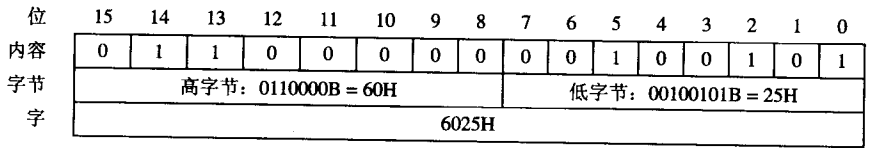


图1-3 字的高字节和低字节

1个双字包括两个字，或4字节，也就是32个二进制位。这两个字分别称为高字（High Order Word）和低字节（Low Order Word）。如图1-4所示，双字0ED66025H的高字为0ED6H，低字为6025H。第31位是最高位（MSB），第0位是最低位（LSB）。

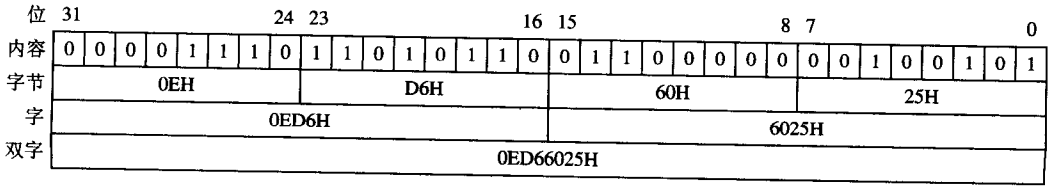


图1-4 双字的高字和低字