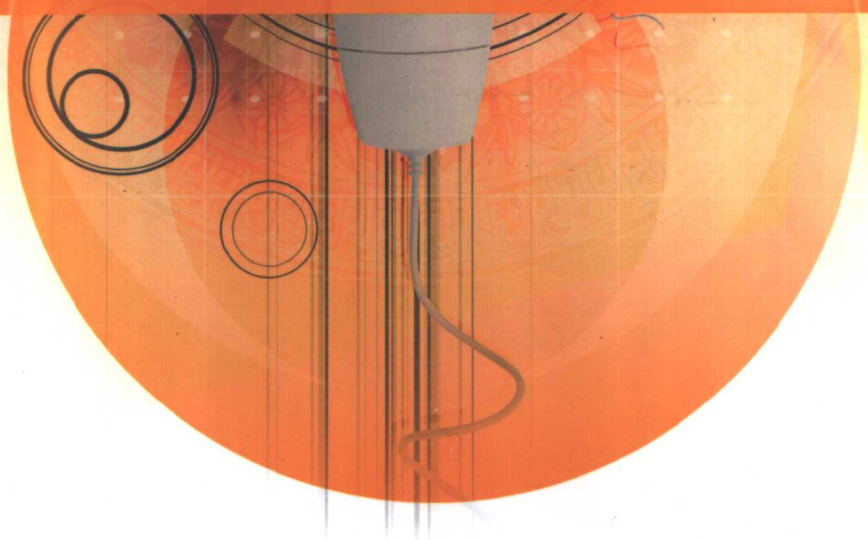
傅德胜 编著

80X86
汇编语言程序
设计及应用



「修订版」

东南大学出版社

80X86
汇编语言程序设计及应用

（原书名：宏汇编语言程序设计及应用）

傅德胜　编著

东南大学出版社

内容提要

本书以80X86微型计算机为背景，全面介绍汇编语言程序设计的原理、方法和技巧，其中包括CPU结构、指令、伪指令、宏指令、汇编语言源程序结构、基本程序(顺序，分支，循环)设计、子程序设计、输入/输出程序设计和模块化程序设计等，并配有例题200余道。本书还以专门的篇幅，展示了汇编语言在数据处理、图像处理、接口与通信等方面的应用风采，因而具有明显的特色和较强的实用性。

本书取材力求跟踪计算机新技术的发展，具有重点突出、承上启下、由浅入深、简明易懂、示例丰富、方便自学等特点。

本书可作为各种不同层次和类型高等院校计算机及相关专业的教材，亦可作为从事计算机研究、生产和开发等有关人员的培训教材和参考书。

图书在版编目(CIP)数据

80X86汇编语言程序设计及应用/傅德胜编著.—2版.—南京：东南大学出版社，2002.6

ISBN 7-81089-017-4

Ⅰ.8… Ⅱ.傅… Ⅲ.汇编语言-程序设计 Ⅳ.TP313

中国版本图书馆CIP数据核字(2002)第013434号

东南大学出版社出版发行

(南京四牌楼2号　邮编210096)

出版人：宋增民

江苏省新华书店经销　　江苏省地质测绘院印刷厂印刷

开本：787mm×1092mm 1/16　　印张：19.75　　字数：490千字

2002年8月第2版　　2002年8月第2次印刷

印数：1—4000　　定价：26.00元

前　言

汇编语言是一种面向机器的程序设计语言。汇编语言直接映射系统硬件，程序设计人员可以像使用自己的工具那样，随心地控制和使用计算机的基本资源，设计各种期望的软件。同时，汇编语言占用的内存小，执行速度快，效率高。因而汇编语言是一种强有力的计算机语言。

在撰写本书的过程中，笔者仔细研究了国家教育部颁布的计算机科学与技术专业的教学大纲要求，深入分析了当前计算机技术的发展，结合自己 20 年来讲授该课程的体会，以及笔者 1999 年 3 月出版的《宏汇编语言程序设计及应用》一书使用中读者的中肯建议，本着由浅入深、循序渐进、考虑专业、兼顾普及、注重实用的精神，反复酝酿和推敲。全书由四大部分组成。第一部分为预备篇(1,2 章)，介绍计算机中用到的数制、码制和 CPU 的结构。第二部分为基础篇(3,4,5 章)，主要讨论汇编语言的寻址方式、指令系统、伪指令、汇编语言源程序结构和汇编语言程序调试过程，为后面汇编语言程序设计的学习作准备。第三部分为程序设计篇(6,7,8,9,10 章)，阐述汇编语言的顺序结构程序、分支结构程序、循环结构程序及子程序与宏指令的设计方法，让读者通过这方面的学习顺利掌握基本的汇编语言程序设计。在此基础上，研究 CPU 与外设之间数据传送程序设计和模块化程序设计，为读者用汇编语言开发大型软件作铺垫。第四部分为应用篇，展示了汇编语言在数据处理、图像处理和接口与通信中的近 30 个方面的应用示例，供读者借鉴和移植。

本书以 80X86 计算机系统为平台。但知识介绍中考虑到汇编语言的发展过程、向下兼容性和某些读者可能面对的是 808X 系统，因而沿用了 808X、80386、80486 和 80586 的叙述轨迹。笔者认为这有利于书中内容学习的逐步深入，也有利于读者根据实际情况有选择的去掌握，可谓两全其美。

本书得到了东南大学出版社的大力支持与帮助。南京气象学院计算机科学与技术系的同仁们提出了不少宝贵的建议。研究生孙文静、刘珍丽、李慧颖、范春年、谢永华等参与了部分工作。在此，一并致以深深的谢意。

本书可作为各种不同层次和类型高等院校计算机及其相关专业的教材，亦可作为计算机研究、生产和应用开发等人员的参考书。

因水平有限，书中的错误在所难免，敬请读者指正。

编　者

2002 年元月于南京气象学院

目　录

1 计算机中数的表示

相对于其他高级语言来说，汇编语言程序设计对数制、码制和数据在计算机中的存储方式的了解显得尤为重要。作为开篇，本章将介绍数制（二进制、十进制和十六进制数之间的相互转换）、码制（原码、反码、补码、ASCII 码、BCD 码、扩充键扫描代码）和存储缓冲区中整型数存取方式。

1.1 计算机数据表示

1.1.1 数制、基数与“位权”

数制表示数的进位方式。如十进制数，逢十进位；八进制数，逢八进位等。

什么是基数呢？基数就是组成某种进位制数的数码个数。如十进制数，基数是 10，有 0，1，2，…，9 共 10 个不同的符号。“位权”（简称权）是指数码在不同数位上代表的数值。以十进制数为例

$$N_D = \begin{array}{ccccc} D_4 & D_3 & D_2 & D_1 & D_0 \\ 1 & 1 & 1 & 1 & 1 \\ \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\ 10^4 & 10^3 & 10^2 & 10^1 & 10^0 \end{array}$$

D_0 的“位权”为 1，D_3 的“位权”为 1000。

1.1.2 二进制数

二进制数 N_B 的数码序列可描述为

$$N_B = b_{n-1}b_{n-2}\cdots b_0 . b_{-1}b_{-2}\cdots b_{-m}B \qquad (1-1)$$

n 为二进制数整数位的位数，m 是二进制数小数部分的位数，B 表示二进制。

二进制的基数为 2，只使用符号 0 和 1。

二进制数的“位权”依次为

$$2^{n-1}, 2^{n-2}, \cdots, 2^0, 2^{-1}, 2^{-2}, \cdots, 2^{-m}$$

二进制数的加、减运算比较简单。自右向左，逐位进行；逢二进位，借一当二。

【例 1-1】 二进制数相加。

10011101.1B + 00101110.1B = 11001100B

【例 1-2】 二进制数相减。

10011001.0B - 01101110.1B = 00101010.1B

```
  10011101.1        10011001.0
+ 00101110.1      - 01101110.1
-------------     -------------
  11001100.0        00101010.1
```

二进制数乘法、二进制数逻辑“与”、“或”、“异或”规则如表 1.1 所示。

表 1.1　二进制数部分运算规则

A	B	A * B	A∧B	A∨B	A⊕B
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	0	1	1
1	1	1	1	1	0

注：符号∧、∨、⊕分别表示“与”、“或”、“异或”运算。

【例 1-3】　二进制数相乘。

10110B × 1011B = 11110010B

【例 1-4】　二进制数相“与”。

10111011B∧01111111B = 00111011B

```
      10110
  ×    1011
-----------
      10110
     10110
    00000
   10110
-----------
  11110010
```

```
  10111011
∧ 01111111
-----------
  00111011
```

1.1.3　十六进制数

十六进制数有 16 个数码符号(表 1.2)，一个十六进制数码序列表示为

$$N_H = h_{j-1}h_{j-2}\cdots h_0 . h_{-1}h_{-2}\cdots h_{-k}H \qquad (1-2)$$

j 为十六进制数整数位数，k 为小数部分的位数，H 表示十六进制。十六进制数的权依次为

$$16^{j-1}, 16^{j-2}, \cdots, 16^0, 16^{-1}, 16^{-2}, \cdots, 16^{-k}$$

表 1.2　十六进制数数码符号

十进制	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
十六进制	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

十六进制数加减运算时，逢十六进位，借一当十六。

【例 1-5】 十六进制数相加、减。

```
  4ab2H          8f42H
+ 3bcfH        - 2abaH
--------       --------
  8681H          6488H
```

值得注意的是，汇编语言程序设计中书写一个十六进制数时，若其最高位的数码符号为 A~F(或 a~f)，则前面必须加数字 0，以便与符号变量相区别。如指令 MOV AX，A42BH 是不合法的，须写成 MOV AX，0A42BH。

1.1.4 不同数制之间的转换

这里主要讨论十进制数与二进制数、十六进制数之间的相互转换以及二进制数、十六进制数之间的转换。

1) 十进制数转换为二进制数

十进制数转换二进制数时，可采用基数乘/除法和减权定位法。

(1) 基数乘/除法

对十进制整数，重复除以 2 取余数，直到商为 0。最后一次除 2 时的余数为二进制整数的最高位。

【例 1-6】 将$(121)_D$转换为二进制整数。

十进制整数	除数	商	余数	二进制数位
121	2	60	1	$b_0=1$
60	2	30	0	$b_1=0$
30	2	15	0	$b_2=0$
15	2	7	1	$b_3=1$
7	2	3	1	$b_4=1$
3	2	1	1	$b_5=1$
1	2	0	1	$b_6=1$

$(121)_D=1111001B$ 或 $(121)_{10}=(1111001)_2$

对十进制小数，不断乘以 2，每次乘后取积的整数部分，直到满足精度要求的位数为止。

【例 1-7】 将$(0.412)_D$转换成二进制小数(精确到二进制小数的第 4 位)。

十进制小数	乘数	积	整数积	二进制数位
0.412	2	0.824	0	$b_{-1}=0$
0.824	2	1.648	1	$b_{-2}=1$
0.648	2	1.296	1	$b_{-3}=1$
0.296	2	0.592	0	$b_{-4}=0$

$(0.412)_D=0.0110B$

(2) 减权定位法

已知二进制数的权为

$$2^{n-1},\ \cdots,\ 2^7,\ 2^6,\ 2^5,\ 2^4,\ 2^3,\ 2^2,\ 2^1,\ 2^0,\ 2^{-1},\ 2^{-2},\ 2^{-3},\ \cdots,\ 2^{-m}$$

$$\downarrow\quad\downarrow\quad\downarrow\quad\downarrow\quad\downarrow\quad\downarrow\quad\downarrow\quad\downarrow\quad\downarrow\quad\downarrow\quad\downarrow$$

128　64　32　16　8　4　2　1　0.5　0.25　0.125

将十进制整数 N_D 转换二进制整数时，运用减权定位法，首先找出与之匹配的最高二进制位 b_i，令 b_i 为 1，将 N_D 减去 b_i 位的权，差为 N_D^1。接着将 N_D^1 与二进制位 b_{i-1} 的权比较，若 $N_D^1 \geqslant$ 权，则 b_{i-1} 为 1，用 N_D^1 减去 b_{i-1} 的权，差为 N_D^2；反之，b_{i-1} 为 0。余此类推，直到比较完 b_0 位。

【例 1-8】 将 $(115)_D$ 转换为二进制数。

权	比较	$N_D = 115$	二进制数位
64	>	$N_D^1 = 51$	$b_6 = 1$
32	>	$N_D^2 = 19$	$b_5 = 1$
16	>	$N_D^3 = 3$	$b_4 = 1$
8	<	$N_D^4 = 3$	$b_3 = 0$
4	<	$N_D^5 = 3$	$b_2 = 0$
2	>	$N_D^6 = 1$	$b_1 = 1$
1	=	$N_D^7 = 0$	$b_0 = 1$

$(115)_D = 1110011B$ 或 $(115)_{10} = (1110011)_2$

十进制小数转换二进制小数，减权定位法的操作过程类似于整数。转换时若二进制小数是“不尽”小数，则根据精度要求或字长范围确定位数。

【例 1-9】 将 $(0.815)_D$ 转换为二进制小数（精确到二进制小数第 4 位）。

权	与权比较	$N_D = 0.815$	b_{-k}
0.50	>	$N_D^1 = 0.315$	$b_{-1} = 1$
0.25	>	$N_D^2 = 0.065$	$b_{-2} = 1$
0.125	<	$N_D^3 = 0.065$	$b_{-3} = 0$
0.0625	>	$N_D^4 = 0.0025$	$b_{-4} = 1$

$(0.815)_D = 0.1101B$

2）*十进制数转换为十六进制数*

转换方法与“十进制数转换二进制数”类似，只是基数改为 16，权改为

$$16^{j-1},\ \cdots,\ 16^3,\ 16^2,\ 16^1,\ 16^0,\ 16^{-1},\ 16^{-2},\ \cdots,\ 16^{-k}$$

$$\downarrow\quad\downarrow\quad\downarrow\quad\downarrow\quad\downarrow\quad\downarrow$$

4096　256　16　1　0.0625　0.0039625

【例 1-10】 将 $(4100)_D$ 转换为十六进制数。

整数转换，可用连续除以 16 取余数的方法，每次所得余数即是十六进制数位值，最后得到的余数是十

六进制数的最高位。

十进制整数	除数	商	余数	十六进制数位
4100	16	256	4	$h_0=4$
256	16	16	0	$h_1=0$
16	16	1	0	$h_2=0$
1	16	0	1	$h_3=1$

$(4100)_D=1004H$

【例 1-11】 将$(0.09)_D$转换为十六进制小数(取 3 位十六进制小数)。

用十进制小数连续乘以 16,每次乘后取积的整数部分,直到满足精度要求为止。

十进制小数	乘数	积	整数积	十六进制数位
0.09	16	1.44	1	$h_{-1}=1$
0.44	16	7.04	7	$h_{-2}=7$
0.04	16	0.64	0	$h_{-3}=0$

$(0.09)_D=0.170H$

3) 二进制数转换为十进制数

由式(1-1)可知,二进制数 N_B 对应的十进制数

$$N_D=(b_{n-1}\times 2^{n-1}+b_{n-2}\times 2^{n-2}+\cdots+b_0\times 2^0)+(b_{-1}\times 2^{-1}+b_{-2}\times b^{-2}+\cdots+b_{-m}\times 2^{-m}) \tag{1-3}$$

【例 1-12】 将$(1110.111)_B$转换为十进制数。

$$N_D=(1\times 2^3+1+2^2+1\times 2^1+0\times 2^0)+(1\times 2^{-1}+1\times 2^{-2}+1\times 2^{-3})$$
$$=(8+4+2)+(0.5+0.25+0.125)=14.875$$

4) 十六进制数转换为十进制数

转换关系如下式示:

$$N_D=(h_{j-1}\times 16^{j-1}+h_{j-2}\times 16^{j-2}+\cdots+h_0\times 16^0)+(h_{-1}\times 16^{-1}+h_{-2}\times 16^{-2}+\cdots+h_{-k}\times 16^{-k}) \tag{1-4}$$

【例 1-13】 令 $N_H=0a4.1bH$,求与其对应的十进制数 N_D。

$$N_D=(a\times 16^1+4\times 16^0)+(1\times 16^{-1}+b\times 16^{-2})$$
$$=(10\times 16+4\times 1)+(1\times 0.0625+11\times 0.00390625)$$
$$=(60+4)+(0.0625+0.04296875)=164.10546875$$

5) 二进制转换为十六进制数

从二进制数小数点开始,向左每四位一组,不满四位时在高位前面补 0;向右每四位一组,不满四位,在末位后面补 0;每四位二进制数用一位十六进制数码表示。

【例 1-14】 二进制数转换为十六进制数。

$(0111,1111,1010,0001.1001,1010)_B=7FA1.9AH$

6) 十六制数转为二进制数

方法很简单,每一位十六进制数码用四位二进制形式表示。

【例 1-15】 十六进制数转换为二进制数。

$4B2D.4BH=0100101100101101.01001011B$

1.2 原码、反码与补码

前面已经讨论了二进制、十进制、十六进制数之间的相互表示。本小节将讨论原码、反码和补码。

1.2.1 无符号数与有符号数

令字长为 n 位(bit),若 n 位均表示数值,这样的数称为无符号数,如 10111011B = 187。

约定字长的最高位表示数的符号,且规定 0 表示正号,1 表示负号,其余位为数值位,这样的数叫做有符号数。如 00001011B = +11, 10001011B = -11。

在同样字长下,有符号数和无符号数所表示的数的范围是不同的:

字长	有符号数表示范围	无符号数表示范围
8	-128 ~ +127	0 ~ 255
16	-32768 ~ +32767	0 ~ 65535
32	-2147483648 ~ +2147483647	0 ~ 4294967295

有符号数在计算机中可以用不同的编码来表示,常见的有原码、反码、补码三种表示方法。

1.2.2 原码

原码表示是一种直观的机器数表示方式,定点整数 y 的原码形式为 $y_N\, y_{N-1}\, y_{N-2} \cdots\, y_0$($y_N$ 为符号位)。原码定义式如下:

$$[y]_{原} = y_N\, y_{N-1}\, y_{N-2} \cdots\, y_0 = \begin{cases} y, & 0 \leqslant y < 2^N \\ 2^N + |y|, & -2^N < y \leqslant 0 \end{cases} \tag{1-5}$$

即 y 为正,$[y]_{原} = y$;y 为负,$[y]_{原} = 2^N + |y|$。N = 机器字长 n - 1。

【例 1-16】 设机器字长为 8 位,写出下列有符号数的原码。

十进制数	机器真值	原码
-125	-1111101	11111101
+32	+0100000	00100000

1.2.3 反码

反码是符号数在计算机中的又一种表示。在机器字长 n = N + 1 位的情况下,定点整数的反码形式为 $y_N\, y_{N-1}\, y_{N-2} \cdots\, y_0$。其定义为

$$[y]_{反} = \begin{cases} y, & 0 \leqslant y < 2^N \\ (2^{N+1} - 1) + y, & -2^N < y \leqslant 0 \end{cases} \tag{1-6}$$

反码的最高位也是符号位,0 表示正,1 表示负。正数的反码和正数的原码相同。负数的反码是负数原码的尾数变反,符号位不变。

【例 1-17】 设机器字长为 8 位,写出下列十进制数的反码。

十进制数	机器真值	反码
+0	+0000000	00000000
-0	-0000000	11111111
-125	-1111101	10000010
+32	+0100000	00100000

1.2.4 补码

补码是计算机中使用频率最高的有符号数的一种编码表示，因为在机器内部，任何一个有符号数都是以补码形式存储和处理的。

1) 补码的定义

若数 Z,V,K 满足关系 Z = nK + V，n 为整数，K 为模，则称 Z 与 V 对 K 是同余的，记为 $Z \equiv V \pmod{K}$。

从上述"同余"概念出发，数 y 的补码

$$[y]_{补} = M + y \pmod{M} \tag{1-7}$$

式中，M 为模。对定点小数，M = 10B，对定点整数，$M = 2^n$，n 为机器字长。假定 N = n - 1，由式(1-7)，可导出补码的又一种定义形式为

$$[y]_{补} = \begin{cases} y, & 0 \leqslant y < 1 \\ 2 - |y|, & -1 \leqslant y < 0 \end{cases} \pmod{2} \tag{1-8}$$

$$[y]_{补} = \begin{cases} y, & 0 \leqslant y < 2^N \\ 2^{N+1} - |y|, & -2^N \leqslant y < 0 \end{cases} \pmod{2^{n-1}} \tag{1-9}$$

式(1-8)、式(1-9)分别对应定点小数 $y_0y_1y_2y_3\cdots y_N$ 和定点整数 $y_Ny_{N-1}y_{N-2}\cdots y_0$。

【例 1-18】 令机器字长 n=8，写出下列机器数的补码。

机器数	过程	补码	模
+0.1001	0.1001	0.1001	mod 2
-0.1001	10b - 0.1001	1.0111	mod 2
-1100100	10000000 - 1100100	10011100	mod 2^8

2) 由原码求补码

除根据定义式(1-7)～式(1-9)求取补码外，还可通过数的原码，经适当步骤，获得补码。方法是：

(1) 正整数

【例 1-19】 设机器字长 n=8，求 +22 的补码。

$[y]_{原} = 00010110$

$[y]_{补} = 00010110$

正数的补码与原码同。

(2) 负整数

【例 1-20】 已知机器字长 n=8，求 -8 的补码。

$[y]_{原} = 10001000$