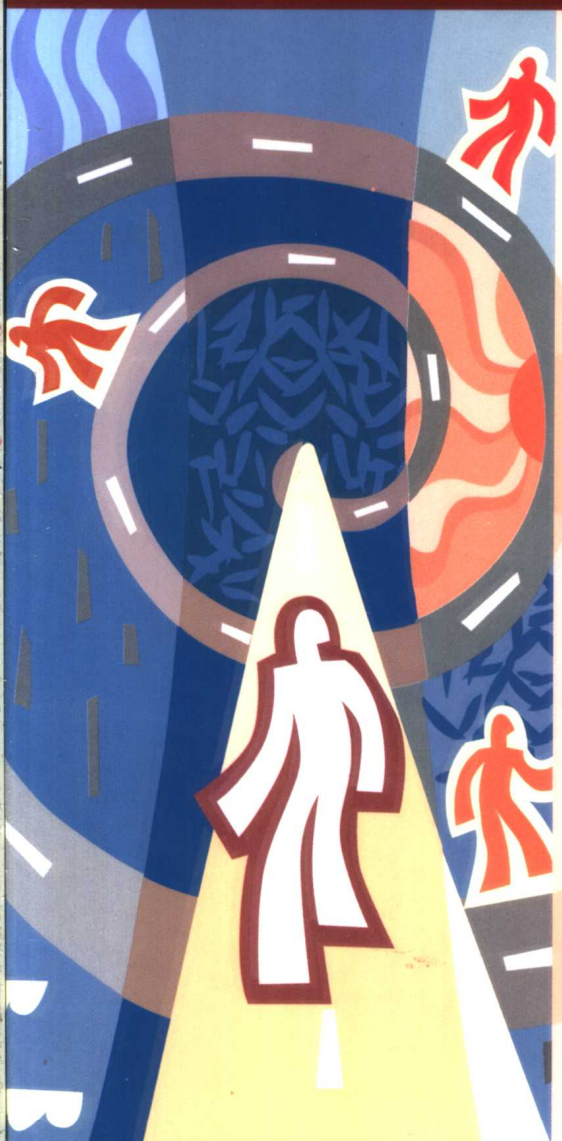
面向21世纪 高职高专计算机专业教材

汇编语言



王富荣 主编

人民交通出版社

面向21世纪 高职高专计算机专业教材

Huibian Yuyan

汇编语言

王富荣 主编

人民交通出版社

内 容 提 要

本书以8086/8088CPU为背景，系统地介绍了汇编语言程序设计的基础知识、程序设计方法和一些应用技术。

全书内容共分8章。第1、2章介绍了微机的基础知识；第3章介绍了8086CPU的寻址方式及指令系统；第4章介绍了汇编语言源程序的结构、用于编制源程序的各种伪指令；第5、6章介绍了汇编语言程序顺序结构、分支结构、循环结构、子程序结构的设计原理、方法；第7、8章讲述中断和系统功能调用。

本书可以作为高等职业学校、高等专科学校、成人高校等计算机专业及相近专业的教材使用，也可以作为从事相关技术工作人员的参考书。

图书在版编目(CIP)数据

汇编语言/王富荣主编.—北京：人民交通出版社，2003.12

ISBN 7-114-04902-1

Ⅰ.汇… Ⅱ.王… Ⅲ.汇编语言 Ⅳ.TP313

中国版本图书馆CIP数据核字(2003)第114331号

面向21世纪高职高专计算机专业教材

汇 编 语 言

王富荣 主编

正文设计：姚亚妮 责任校对：尹 静 责任印制：杨柏力

人民交通出版社出版发行

(100013 北京和平里东街10号 010 64216602)

各地新华书店经销

三河市宝日文龙印务有限公司印刷

开本：787 ×1092 1/16 印张：13.75 字数：334千

2004年1月 第1版

2004年1月 第1版 第1次印刷

印数：0001 — 3000册 定价：23.00元

ISBN 7-114-04902-1

编写人员名单

主　　编：王富荣（南通航运职业技术学院）

副 主 编：刘造新（江西交通职业技术学院）

娄　智（安徽交通职业技术学院）

本书策划组成员名单

白　崤　翁志新　张　景　黄景宇

前言 FOREWORD

根据21世纪高等职业教育的新趋势和计算机专业学科建设的要求，结合目前众多高职高专院校中教学计划，人民交通出版社组织全国十几所高职高专院校中多年从事一线教学、实践能力强且具有丰富教材编写经验的教师，编写了这套“面向21世纪高职高专计算机专业教材”，共21本（书目附后），涵盖了高职高专计算机及相关专业的主要课程。在编写过程中认真贯彻了教育部《关于加强高职高专教育人才培养工作的意见》的精神。内容以必需、够用为度，既注重基础知识的讲解，又注意从实际应用出发，满足社会对计算机类专业人才的需求，突出以能力为本位的高等职业教育的特色。

应当说明的是，凡是高等职业教育、高等专科教育和成人高等教育院校的计算机及其相关专业的师生均可使用本套教材。各学校可以根据实际需要，在教学中适当增删一些内容，从而更有针对性地帮助学生掌握计算机专业知识，并形成相关应用能力。

本套教材的出版，将促进高等职业教育的教材建设，对我国高等职业教育的发展产生积极的影响。同时，我们也希望在今后的使用中不断改进、完善此套教材，更好地为高等职业教育服务。

编　者

目 录 CONTENTS

第 1 章 基础知识

[主要内容] 本章主要介绍汇编语言的基础知识。包括:常用的进位计数制及各种数制之间的转换;二进制数的算术、逻辑运算;机器数三种不同的编码格式(原码、反码和补码);二进制数补码的符号扩展及表示范围;字符数据的表示法,如:ASCII 码和 BCD 码等。

1.1 数制及数制间的转换

1.1.1 数制、基数及位权

进位计数制是一种计数的方法,人们习惯上使用十进制数作为计数的方法。在日常生活中,也形成了多种的进位计数制,如计时使用的时、分、秒就是按六十进制计数的。而在计算机中为便于存储及计算的物理实现,采用了二进制数。对于各种进位计数制可以统一成一个 r 进制数,表示如下:

$a_n a_{n-1} \cdots a_1 a_0 . b_1 b_2 \cdots b_m$

其含义是:

$a_n \times r^n + a_{n-1} \times r^{n-1} + \cdots + a_1 \times r^1 + a_0 \times r^0 + b_1 \times r^{-1} + b_2 \times r^{-2} + \cdots + b_m \times r^{-m}$

r 进制数所用数码为 0,1,…,r-1,共 r 个,则称该进制数的基数为 r,即基数可定义为进位计数制中所用数码的个数。r^k 则称为以 r 为基数的第 k 位的权。上述数中 a_i、b_j 可以是 0,1,…,r-1 中的任意一个数码,且遵循“逢 r 进 1”的原则。

1.1.2 进位计数制

1.十进制数(Decimal)

十进制数在计算机书写时一般用数字后跟一个英文字母 D(Decimal)来表示,它的基数为 10,即数码个数为 10,分别为 0,1,2,3,4,5,6,7,8,9,且遵循“逢 10 进 1”的原则。一个十进制数如 123.45 可以表示为:

$123.45D = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$

上式中 10^k 为各位的权。

2.二进制数(Binary)

二进制数在计算机里书写时一般用数字后跟一个英文字母 B(Binary)来表示,它的基数为 2,即数码个数为 2,分别为 0、1,遵循“逢 2 进 1”的原则。各位数的权值为 2^k。一个二进制数如 10110 可表示为:

$10110B = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 22D$

其中数的后面带一个字母表示该数所用的进位计制数，二进制数 10110 按权展开求和等于十进制数 22。

n 位二进制数可以表示 2^n 个数。例如 3 位二进制数可以表示 8 个数，如表 1-1 所示：

3 位二进制数表示 表 1-1

二进制数	000	001	010	011	100	101	110	111
对应的十进制数	0	1	2	3	4	5	6	7

对应的 4 位二进制数可以表示十进制数的 0 ~ 15，共 16 个数。

3. 八进制数(Octal)

八进制数在计算机里书写时一般用数字后跟一个英文字母 O(Octal)来表示，有时为与 0 相区别，也可用 Q 来表示。它的基数为 8，即数码的个数为 8，分别为 0、1、2、3、4、5、6、7，遵循“逢 8 进 1”的原则。可以用 3 位二进制数来表示一位八进制数。

4. 十六进制数(Hexadecimal)

十六进制数在计算机里书写时一般用数字后跟一个英文字母 H(Hexadecimal)来表示，它的基数为 16，即数码的个数为 16，分别用 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F 来表示，其中 A、B、C、D、E、F 表示对应十进制数的 10、11、12、13、14、15。遵循“逢 16 进 1”的原则。可以用 4 位二进制数来表示一位十六进制数。各种进制数之间的对应关系如表 1-2 所示：

各进制数之间的对应关系 表 1-2

十进制数	二进制数	八进制数	十六进制数	十进制数	二进制数	八进制数	十六进制数
0	0	0	0	9	1001	11	9
1	1	1	1	10	1010	12	A
2	10	2	2	11	1011	13	B
3	11	3	3	12	1100	14	C
4	100	4	4	13	1101	15	D
5	101	5	5	14	1110	16	E
6	110	6	6	15	1111	17	F
7	111	7	7	16	10000	20	10
8	1000	10	8				

1.1.3 各种数制之间的转换

由于二进制数与八进制数、十六进制数之间有固定的对应关系，可按 3 位或 4 位二进制数

为一组,分别完成二进制数与八进制数、十六进制数之间的转换。因此,各种数制之间的相互转换主要是十进制数与二进制数之间的转换问题。

1.二进制数转换为十进制数

二进制数转换为十进制数的方法为:各位二进制数的数码乘以与其对应的权值之和。

如:101101.1011B = $1\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}+1\times2^{-4}$ = 45.6875D。

对于八进制数、十六进制数转换为十进制数可采用同样的办法,即加权系数之和。

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,仅介绍以下方法。

1)整数部分的转换

采用"除2取余法",即把要转换的十进制数的整数部分不断除以2,并记下余数,直到商为0为止。

例1.1 N = 115D

115/2 = 57 ($a_0=1$)
57/2 = 28 ($a_1=1$)
28/2 = 14 ($a_2=0$)
14/2 = 7 ($a_3=0$)
7/2 = 3 ($a_4=1$)
3/2 = 1 ($a_5=1$)
1/2 = 0 ($a_6=1$)

将余数从后往前排列,得到1110011B。

所以 N = 115D = 1110011B。

对于十进制整数转换为八进制数或十六进制数可以采用相应的"除以r(基数)取余法"。当然也可以采用先转换成二进制数之后,再转换成对应的八进制数或十六进制数的方法。

2)小数部分的转换

采用"乘2取整法",即将十进制数的小数部分不断乘以2,并记下其整数部分,直到结果的小数部分为0或满足所需精度要求的位数为止。

例1.2 N = 0.8125D

$0.8125\times2=1.625$ ($b_1=1$)
$0.625\times2=1.25$ ($b_2=1$)
$0.25\times2=0.5$ ($b_3=0$)
$0.5\times2=1.0$ ($b_4=1$)

将整数部分从前往后排列,得到1101B。

所以 N = 0.8125D = 0.1101B

3.二进制数与八进制数、十六进制数间的相互转换

由表1-2可见,二进制数转换八进制数,只需将二进数以小数点为中心,向前或向后分别

按每3位一组,若不足3位,那么整数部分前面用0补齐,小数部分后面用0补齐3位,用表1-1中对应的八进制数写出,即为其对应的八进制数。反之,将八进制数转换为二进制数,只要把每位八进制数用对应的3位二进制数表示即可。

二进制数与十六进制数的转换同二进制数与八进制数转换相仿,只是按4位进行分组。

例1.3 将745.362O转换为对应的二进制数。

7	4	5.	3	6	2
111	100	101.	011	110	010

即 745.362O = 111100101.011110010B

例1.4 将1011010.10111B转换为十六进制数。

101	1010.	1011	1000
5	A.	B	8

即 1011010.10111B = 5A.B8H

1.2 二进制数的运算

1.2.1 二进制数的算术运算

二进制的算术运算中,加法运算和乘法运算是基本运算,它们的运算规则是:

1.加法

1+1=0(进位1)　1+0=1　0+1=1　0+0=0

两个二进制数相加,是按照加法规则,按位相加,逢二进一进行运算。

例1.5 1101B+0111B=10100B

$$\begin{array}{r} 1101 \\ \times\ 0111 \\ \hline 10100 \end{array}$$

2.乘法

0×0=0　0×1=0　1×0=0　1×1=1

两个二进制数相乘的方法与十进制数相乘类似,先用乘数的第1位乘被乘数的各位,再用第2位、第3位……与被乘数各位相乘,依次向左移位求和,即得到相乘的结果。

例1.6 1101B×1001B=1110101B

$$\begin{array}{r} 1101 \\ \times\ 1001 \\ \hline 1101 \\ 0000 \\ 0000 \\ 1101 \\ \hline 1110101 \end{array}$$

在计算机中,二进制数的减法运算和除法运算可以转换成加法和乘法运算,故不再加以讨论。

1.2.2 二进制数的逻辑运算

逻辑运算是以二进制数为基础的,逻辑变量只有2个,一般用"1"和"0"分别表示逻辑"真"和"假"。对逻辑变量的运算常用的有:逻辑加、逻辑乘、逻辑非和逻辑异或等。

1.逻辑加法(或运算)

运算符号:"+"或"∨"。运算规则如表1-3所示:即A、B两个变量中只要有一个变量取值为"1",则它们"或"运算的结果就是"1"。

二进制逻辑加法运算规则　　表1-3

A	B	A∨B	A	B	A∨B
0	0	0	1	0	1
0	1	1	1	1	1

2.逻辑乘法(与运算)

运算符号:"×"或"∧"。运算规则如表1-4所示:即只要两个变量中只要有一个取值为"0",则它们与运算的结果为"0",两个变量的取值均为1时,它们与运算的结果才是"1"。

二进制逻辑乘法运算规则　　表1-4

A	B	A∧B	A	B	A∧B
0	0	1	1	0	0
0	1	0	1	1	1

3.逻辑非运算

运算符为在二进制变量上方加一横线,如变量为A,则它的非运算的结果用运算符 $\bar{A}$ 来表示。运算规则如表1-5所示:

逻辑非运算规则　　表1-5

A	$\bar{A}$	A	$\bar{A}$
0	1	1	0

4.逻辑异或运算

逻辑异或又称按位加,其运算符可用来"⊕"或"∀"表示。逻辑异或运算规则如表1-6所示。

由表可知两个变量A和B进行逻辑异或运算时,当两个逻辑变量的取值相异时,它们的异或结果就为1,即"同值为0,异值为1"。

逻辑异或运算规则　　表1-6

A	B	A∀B	A	B	A∀B
0	0	0	1	0	1
0	1	1	1	1	0

所有的逻辑运算都是按位操作的。

例 1.7 已知 X = 01101111B, Y = 10101010B, 求 X∨Y, X∧Y, $\overline{X}$, X∀Y

解: X∨Y = 01101111B∨10101010B = 11101111B

X∧Y = 01101111B∧10101010B = 00101010B

$\overline{X}$ = 10010000B

X∀Y = 01101111B∀10101010B = 11000101B

1.3 计算机中数和字符的表示

1.3.1 数值数据

计算机系统内部数值数据采用二进制数表示,非数值数据和计算机指令也采用二进制编码表示。计算机中存储信息的最小单位称为位,可以存储 0 和 1 两种状态。

数值数据分为有符号数和无符号数。有符号数分为正数和负数,为了表示正数和负数,专门选择一个二进制位来表示数的符号位。通常最高位作为符号位,用"0"表示正数,用"1"表示负数。无符号数则无符号位,最高位表示数值。一个数连同符号在内均用二进制表示,这样的数称为机器数。而它对应的实际数值称为真值。通常机器数有三种不同的编码形式,即原码、反码和补码。

1.原码

最高位表示符号(正数用 0,负数用 1),其它位表示数的绝对值,称为有符号数的原码表示法。若真值为 X,则其原码记作$[X]_{原}$

例 1.8 设字长为 8 位,分别求 45 和 -45 的原码

$$X = 45 = +0101101B, [X]_{原} = 00101101B$$

$$Y = -45 = -0101101B, [Y]_{原} = 10101101B$$

注意:0 的原码有两个,即$[0]_{原}$ = 00000000B,或$[0]_{原}$ = 10000000B。

原码表示法的优点是简单易懂,缺点是进行加减运算复杂。当两个数求和时,首先要判断它们的符号。如果符号相同,则绝对值部分相加;否则,绝对值部分相减。若相减,还必须先比较两个数的绝对值的大小,然后做减法,最后给结果确定符号。这使得求和过程变得复杂。为了简化运算过程,把减法运算转换为加法运算,就引进了反码和补码表示法。

2.反码

正数的反码与原码相同,符号位用 0 表示,数值位保持不变。负数的反码,符号位用 1 表示,数值位按原码数值位按位取反形成,即 0 变 1、1 变 0。若真值为 X,则其反码记作$[X]_{反}$。

例 1.9 设字长为 8 位,分别求 45 和 -45 的反码。

$$X = 45 = +0101101B, [X]_{反} = 00101101B$$

$$Y = -45 = -0101101B, [Y]_{反} = 11010010B$$

注意:0 的反码同样有两个,即$[0]_{反}$ = 00000000B,或$[0]_{反}$ = 11111111B

3.补码

正数的补码与原码相同,即符号位用0表示,数值位保持不变。负数的补码是最高位符号位为"1"不变,将该数的真值部分按位取反且在最低位加1。因此可从反码的表示法中获得求负数补码的简便方法。若真值为X,则其补码记作$[X]_{补}$。

例1.10 设字长为8位,分别求45和-45的补码。

$$X = +45 = +0101101B, [X]_{补} = 00101101B$$

$$Y = -45 = -0101101B, [Y]_{补} = 11010011B$$

注意:0的补码只有一个,即$[0]_{补} = 00000000B$,而$[-128]_{补} = 10000000B$。

在计算机中,有符号数一般采用补码形式表示,这主要的原因是补码的加减运算比原码简单,能将减法运算转换为加法运算,从而使减法运算更为简便。进行补码加减运算的规则如下:

(1)符号位要与数值位一起参加运算。

(2)符号参加运算后如有进位出现,则把这个进位舍去不要。

(3)补码的运算具有如下运算性质:

加法 $[X+Y]_{补} = [X]_{补} + [Y]_{补}$

减法 $[X-Y]_{补} = [X]_{补} + [-Y]_{补}$

此外,由$[X]_{补}$求$[-X]_{补}$可采用的方法是将$[X]_{补}$的各位(连同符号位)求反,并在末位加1。这称为求补运算。

即:$[X]_{补}$经求补运算后为$[-X]_{补}$。

例1.11 已知$[a]_{补} = 10110101B$,求$[-a]_{补}$。

由上介绍的方法得$[-a]_{补} = 01001011B$

例1.12 已知X=+0101101B,Y=-1000011B,求X+Y,X-Y

解:$[X]_{补} = 00101101B$ $[Y]_{补} = 10111101B$ $[-Y]_{补} = 01000011B$

$[X+Y]_{补} = [X]_{补} + [Y]_{补} = 00101101B + 10111101B = 11101010B$

$[X-Y]_{补} = [X]_{补} + [-Y]_{补} = 00101101B + 01000011B = 01110000B$

所以X+Y=-00010110B,X-Y=+1110000B

4.符号扩展

在数据处理时,有时需要把8位二进制数扩展成16位二进制数,当要扩展的数值是无符号数时,可在最高位前扩展8个0。如果要扩展的数是补码形式的有符号数,那么就要进行带符号位的扩展。8位有符号二进制数扩展成16位二进制数的规则是将16位二进制数的高8位全部用符号位填充。带符号位扩展后,其结果仍是该数的补码。

例1.13 将35的8位二进制补码00100011B和-35的8位二进制补码为11011101B分别符号扩展成16位二进制数形式。

解:$[35]_{补}$为00100011B,符号位为0,符号扩展,高8位全用0填充,得35的16位二进制补码0000000000100011B。

00100011B和0000000000100011B都是35的二进制补码。

$[-35]_{补}$ 11011101B,符号位为1,符号扩展,高8位全用1填充,得-35的16位二进制补码为:1111111111011101B。

11011101B 和 1111111111011101B 都是 - 35 的二进制补码。

5. 数据的表示范围和大小

n 位二进制数能够表示的无符号整数的范围是：

$$0 \leqslant N \leqslant 2^n - 1$$

n 位二进制数能够表示的有符号整数的范围是：

$$-2^{n-1} \leqslant N \leqslant +(2^{n-1} - 1)$$

所以，如果 n = 8，那么能够表示的无符号整数范围是 0 ~ 255，共 256 个数，能够表示的有符号整数范围是 - 128 ~ + 127；如果 n = 16，那么能够表示的无符号整数范围是 0 ~ 65535，能够表示有符号整数的范围是 - 32768 ~ + 32767。

对于 16 位二进制数，用补码表示的有符号数的数值大小如表 1-7 所示。

16 位二进制补码数的表示范围 表 1-7

十进制数	+ 32767	+ 32766	…	+ 2	+ 1	0	- 1	- 2	…	- 32768
十六进制数	7FFF	7FFE	…	0002	0001	0000	FFFF	FFFE	…	8000

1.3.2 字符数据

1. ASCII 码

在计算机内，任何信息（包括字母、数字、符号和汉字）都必须按特定的规则编成二进制代码才能在计算机中表示。编码方式可以有多种，现在最普遍采用的一种编码是 ASCII 码（American Standard Code for Information Interchange，美国标准信息交换码）。

标准 ASCII 码用 7 位二进制数对字符编码，共有 128 个符号，参见附录 A。计算机存储器的基本单元为 8 位二进制数，即一个字节。每个 ASCII 码字符占用一个字节来存储，通常最高位为“0”，低 7 位为字符的二进制编码；通信时，最高位常用作奇偶校验位。

ASCII 码表中的前 32 个和最后 1 个编码是不能显示的控制字符，用于表示某种操作。

例如：0DH 表示回车 CR；0AH 表示换行 LF；07H 表示响铃 BEL。

ASCII 码表中 20H 后的 95 个编码是可显示和打印的字符，其中包括数码 0 ~ 9，英文字母，标点符号等。从附录 A 可看到，数码 0 ~ 9 的 ASCII 码为 30H ~ 39H；大写字母 A ~ Z 的 ASCII 码为 41H ~ 5AH；小写字母 a ~ z 则是 61H ~ 7AH。

2. BCD 码

1）BCD 码

虽然二进制数实现容易，但不符合人们的使用习惯，且书写阅读不方便，所以在计算机输入输出时通常还是采用十进制来表示数，这就需要实现十进制与二进制之间的转换。为了转换方便，常采用二进制数编码来表示十进制数，简称为 BCD（Binary-Coded Decimal）码。常用 4 位二进制数表示一个十进制数。

BCD 码有多种，在计算机中常用两种表示法：一种是压缩 BCD 码表示法，就是用 4 位二进制数表示 1 位十进制数字，整个十进制数用一个顺序 4 位一组的二进制数位组来表示，1 个字节（8 位二进制数）可以存放 2 位压缩 BCD 码。另一种是非压缩 BCD 码表示法，它用 1 个字节表示 1 位十进制数字，其中高四位没有意义，通常取 0。我们通常所说的 BCD 码指的是压缩