高等院校计算机基础教育规划教材

汇编语言

主编　冯康　　副主编　范理　王利丽

WUHAN UNIVERSITY PRESS
武汉大学出版社

内容简介

本书旨在较全面地介绍汇编语言的基本功能、相关命令和汇编语言程序开发的基本方法。通过相关理论知识的学习和实践操作训练，使读者能够熟练掌握 MASM 的开发平台，正确地、合理地进行汇编语言程序的开发，全面提高读者的程序设计水平。

本书针对职业技术教育的特点，突出计算机科学与技术学科的特点，强调理论与实践相结合的教学方式，用示例讲解的方式，引导读者学习、掌握汇编语言程序设计技术。

本书适合作为高等专科学校、高等职业学校、成人高等学校以及高等院校主办的二级职业技术学院计算机及相关专业学生使用的教材。目的在于培养学生的实际动手的能力，使得学生更加适合用人单位的技能要求。

图书在版编目(CIP)数据

汇编语言/冯康主编．—武汉：武汉大学出版社，2007.5
高等院校计算机基础教育规划教材
ISBN 978-7-307-05563-6

Ⅰ．汇… Ⅱ．冯… Ⅲ．汇编语言—高等学校—教材 Ⅳ．TP313

中国版本图书馆 CIP 数据核字(2007)第 057666 号

出版发行：武汉大学出版社 (430072 武昌 珞珈山)
(电子邮件：wdp4@whu.edu.cn 网址：www.wdp.com.cn)
印刷：北京市昌平百善印刷厂
开本：787×1092 1/16 印张：16
字数：399 千字 印数：1～3000 册
版次：2007 年 5 月第 1 版 2007 年 5 月第 1 次印刷
ISBN 978-7-307-05563-6/TP・245 定价：26.00 元

高等院校计算机基础教育规划教材

编委会

前言

PREFACE

近年来，由于微型计算机的迅速发展和广泛应用，越来越多的技术人员在系统设计或技术开发中需要学习汇编语言程序或编写汇编语言程序。在国内外的中、高等院校中，汇编语言程序设计也是计算机专业学生必修的专业基础课程之一。

本教材针对职业技术教育的特点，突出计算机科学与技术学科的特点，强调理论与实践相结合的教学方法，用示例讲解的方式，引导读者学习、掌握汇编语言程序设计技术。

本教材旨在较全面地介绍汇编语言的基本功能、相关命令和汇编语言程序开发的基本方法。通过相关理论知识的学习和实践操作训练，使学生能够熟练掌握 MASM 的开发平台，正确地、合理地进行汇编语言程序的开发，全面提高学生的程序设计水平。

全书共分 12 章，主要内容包括：基础知识、Intel 80x86 汇编语言编程结构、寻址方式、DEBUG 调试、8086/8088 CPU 指令系统、汇编语言程序格式和上机调试、汇编语言基本结构与程序设计、子程序及模块化程序设计、系统中断调用和 I/O 程序设计、高级汇编语言技术、应用系统开发、实训等。

本书适合作为高等专科学校、高等职业学校、成人高等学校以及高等院校主办的二级职业技术学院计算机及相关专业学生使用的教材。

由于编者水平有限，加之编写时间仓促，在教材中难免会有疏漏和不妥之处，欢迎读者批评指正。

编　者

2007 年 3 月

目 录

CONTENTS

第 1 章 基础知识 …… 1
1.1 汇编语言程序设计中的进位记数制 …… 1
1.1.1 二进制数 …… 1
1.1.2 十六进制数 …… 2
1.1.3 十进制数 …… 2
1.1.4 数制间的转换 …… 2
1.1.5 二进制和十六进制的运算 …… 4
1.2 汇编语言程序设计中数据的表示 …… 5
1.2.1 整数数值数据的表示 …… 5
1.2.2 字符的编码 …… 7
1.3 机器语言、汇编语言、汇编编译器 …… 8
1.3.1 机器语言 …… 9
1.3.2 汇编语言 …… 9
1.3.3 汇编编译器 …… 9
1.3.4 汇编语言的特点及用途 …… 10
本章小结 …… 10
习题 1 …… 10

第 2 章 Intel 80x86 汇编语言编程结构 …… 12
2.1 Intel 8086/8088 CPU 的汇编语言编程结构 …… 12
2.1.1 8086/8088 CPU 结构概述 …… 12
2.1.2 8086/8088 CPU 寄存器结构 …… 13
2.1.3 8086/8088 CPU 内存组织 …… 16
2.1.4 8086/8088 CPU 的 I/O 组织 …… 21
2.2 Intel 的 32 位 CPU 汇编语言编程结构 …… 22
2.2.1 工作模式概述 …… 22
2.2.2 Intel 32 位 CPU 的编程结构介绍 …… 23
本章小结 …… 25

习题 2 …… 25

第 3 章 寻址方式 …… 26
3.1 Intel CPU 的指令 …… 26
3.1.1 指令与指令系统 …… 26
3.1.2 指令的格式 …… 26
3.2 Intel 8086/8088 CPU 的寻址方式 …… 27
3.2.1 立即寻址 …… 28
3.2.2 寄存器寻址 …… 29
3.2.3 直接寻址 …… 29
3.2.4 寄存器间接寻址 …… 31
3.2.5 寄存器相对寻址 …… 32
3.2.6 基址变址寻址 …… 33
3.2.7 相对基址变址寻址 …… 34
3.2.8 端口寻址方式 …… 34
3.3 实例解析 …… 35
本章小结 …… 35
习题 3 …… 36

第 4 章 DEBUG 调试 …… 38
4.1 DEBUG 命令格式 …… 38
4.1.1 DEBUG 调试程序的启动 …… 38
4.1.2 DEBUG 命令的格式 …… 40
4.2 DEBUG 命令 …… 41
4.2.1 显示内存单元内容命令 D(DUMP) …… 41
4.2.2 修改命令 E(Enter) …… 42
4.2.3 填充命令 F(Fill) …… 43
4.2.4 寄存器命令 R(Register) …… 43
4.2.5 汇编命令 A(Assemble) …… 45
4.2.6 反汇编命令 U(Unassemble) …… 46
4.2.7 运行命令 G(Go) …… 47
4.2.8 跟踪命令 T(Trace) …… 47
4.2.9 继续命令 P(Proceed) …… 49
4.2.10 退出命令 Q(Quit) …… 50
4.3 实例解析 …… 51
本章小结 …… 51
习题 4 …… 52

第 5 章 8086/8088 CPU 指令系统 …… 53
5.1 数据传送指令 …… 53

5.1.1 通用传送指令 …… 53
5.1.2 累加器专用传送指令 …… 55
5.1.3 地址传送指令 …… 55
5.1.4 标志位传送指令 …… 56
5.2 算术运算指令 …… 57
5.2.1 加法运算指令 …… 57
5.2.2 减法运算指令 …… 58
5.2.3 乘法运算指令 …… 59
5.2.4 除法运算指令 …… 60
5.2.5 十进制调整指令 …… 62
5.3 逻辑类指令 …… 63
5.3.1 逻辑运算指令 …… 64
5.3.2 移位指令 …… 65
5.4 串操作指令 …… 67
5.4.1 串操作指令概述 …… 67
5.4.2 5种串操作指令 …… 68
5.5 控制转移指令 …… 71
5.5.1 与目标指令地址有关的寻址方式 …… 71
5.5.2 JMP无条件转移指令 …… 73
5.5.3 条件转移指令 …… 73
5.5.4 循环指令 …… 77
5.5.5 子程序调用与返回指令 …… 78
5.6 处理器控制指令 …… 81
5.7 32位CPU扩展的指令 …… 82
5.7.1 Intel 32位CPU扩充的寻址方式 …… 82
5.7.2 80386增强和扩展指令 …… 83
5.7.3 80486新增指令 …… 85
5.7.4 Pentium新增指令 …… 86
5.8 实例解析 …… 87
本章小结 …… 88
习题5 …… 88

第6章 汇编语言程序格式和上机调试 …… 90
6.1 汇编语言源程序与汇编程序 …… 90
6.2 汇编语言语句种类及其格式 …… 90
6.2.1 指令语句 …… 90
6.2.2 伪指令语句 …… 91
6.2.3 标识符 …… 92
6.3 汇编语言中的运算符与表达式 …… 92
6.3.1 常量 …… 92

6.3.2 变量 …… 93
6.3.3 表达式和运算符 …… 94
6.4 伪指令 …… 96
6.4.1 变量定义伪指令 …… 96
6.4.2 符号定义伪指令 …… 96
6.4.3 段结构伪指令 …… 97
6.4.4 源程序开始和结束伪指令 …… 98
6.4.5 定位伪指令 ORG 与汇编地址计数器 …… 99
6.4.6 过程定义伪指令 PROC 和 ENDP …… 99
6.5 汇编语言源程序的结构 …… 99
6.6 汇编语言程序的上机过程 …… 100
6.7 实例解析 …… 101
本章小结 …… 107
习题 6 …… 107

第 7 章 汇编语言基本结构与程序设计 …… 110
7.1 汇编语言程序设计的一般方法 …… 110
7.1.1 汇编语言程序设计的一般步骤 …… 110
7.1.2 结构化程序设计 …… 110
7.1.3 流程图画法 …… 112
7.2 顺序程序设计 …… 113
7.3 分支程序的设计 …… 118
7.3.1 两分支程序设计 …… 118
7.3.2 多分支程序设计 …… 121
7.4 循环程序设计 …… 136
7.4.1 单重循环程序设计 …… 136
7.4.2 多重循环程序设计 …… 141
7.5 实例解析 …… 149
本章小结 …… 151
习题 7 …… 151

第 8 章 子程序及模块化程序设计 …… 154
8.1 子程序设计方法 …… 154
8.1.1 子程序定义 …… 154
8.1.2 子程序的调用与返回 …… 155
8.1.3 现场保护与恢复 …… 157
8.1.4 主程序与子程序的参数传递 …… 158
8.2 子程序设计举例 …… 165
8.3 子程序的嵌套与递归 …… 177
8.3.1 子程序的嵌套 …… 177

8.3.2 子程序的递归调用 …… 179
8.4 模块化程序设计 …… 180
8.4.1 基本思想和主要步骤 …… 180
8.4.2 模块间通信的基本方法 …… 181
8.4.3 子程序库建立和使用的方法 …… 182
8.4.4 汇编语言与高级语言的接口 …… 183
8.5 实例解析 …… 184
本章小结 …… 185
习题 8 …… 185

第 9 章 系统中断调用和 I/O 程序设计 …… 187
9.1 中断概述 …… 187
9.2 BIOS 中断调用 …… 187
9.2.1 BIOS 中断调用方法 …… 188
9.2.2 常用 BIOS 中断 …… 188
9.3 DOS 功能调用 …… 189
9.3.1 DOS 中断调用方法 …… 189
9.3.2 常用 DOS 中断 …… 189
9.4 I/O 程序设计 …… 190
9.4.1 键盘输入中断 …… 190
9.4.2 键盘中断处理过程 …… 190
9.4.3 键盘输入举例 …… 191
9.4.4 屏幕中断功能 …… 192
9.4.5 屏幕输出的举例 …… 193
9.5 实例解析 …… 194
本章小结 …… 196
习题 9 …… 196

第 10 章 高级汇编语言技术 …… 198
10.1 宏汇编 …… 198
10.1.1 宏定义 …… 198
10.1.2 宏调用和宏展开 …… 199
10.1.3 宏调用中的参数使用 …… 200
10.1.4 宏嵌套 …… 201
10.1.5 宏汇编中的伪指令 …… 202
10.1.6 宏在编程中的应用 …… 202
10.2 重复汇编 …… 204
10.2.1 使用 REPT 伪指令的重复汇编结构 …… 204
10.2.2 使用 IRP 伪指令的重复汇编结构 …… 205
10.2.3 使用 IRPC 伪指令的重复汇编结构 …… 205

10.3 条件汇编伪指令 …… 206
10.4 宏指令与子程序的区别 …… 206
10.5 实例解析 …… 207
本章小结 …… 208
习题 10 …… 208

第 11 章 应用系统开发 …… 210
11.1 图形系统开发 …… 210
11.1.1 字符图形显示 …… 210
11.1.2 动画程序的演示 …… 219
11.2 声音系统开发 …… 221
11.2.1 音乐程序 …… 221
11.2.2 定时报警程序 …… 226
11.3 实例解析 …… 230
本章小结 …… 235
习题 11 …… 235

第 12 章 实训 …… 237
实训 1 随机数的生成 …… 237
实训 2 电话号码查询 …… 237
实训 3 学生成绩管理系统 …… 237
附录 …… 238
参考文献 …… 242

第1章　基础知识

本章提要

本章是汇编语言程序设计的开篇，主要介绍汇编语言程序设计所涉及的一些基础知识，并引出有关基本概念，为读者对后续内容的学习，打下必要的基础。

主要介绍以下几方面的内容：

- ◆ 汇编语言中常用的数制表示方法
- ◆ 数制的运算
- ◆ 汇编语言中数据的表示
- ◆ 机器语言、汇编语言、汇编程序等基本概念
- ◆ 汇编语言的特点及用途

1.1　汇编语言程序设计中的进位记数制

进位记数制是一种计数的方法，在汇编语言程序设计中常用二进制、十六进制和十进制三种进位记数制，本节将根据汇编语言程序设计的需要对这三种进位记数制予以介绍。八进制由于汇编语言程序设计中基本不使用，因此本书对于八进制编码的相关内容不做讲解，读者如有兴趣请参考有关资料；另外，在汇编语言程序设计中计数的对象均为整数，故本节的进位记数制不涉及小数。

1.1.1　二进制数

二进制数(Binary)的基数为2，只有0、1这两位数码，计数时遵循逢2进1的规则，每位的位权以2^k表示。因此，二进制数：

$$a_n a_{n-1} \cdots a_1 a_0 B$$

等于十进制数：

$$a_n \times 2^n + a_{n-1} \times 2^{n-1} + \cdots + a_1 \times 2^1 + a_0 \times 2^0$$

二进制数便于物理上电路量的实现，如用高、低电平分别表示数码1、0。因此，在计算机的具体存放数据的硬件电路中，如寄存器、存储单元、端口，所有数据的惟一真正的存在形式都是二进制。二进制数的另外优点是便于纠错，如已知某一位二进制数据出错，则只要对该位二进制数据取反，即可实现该位数据的纠错。二进制数在书写时通常以B或b结尾。如二进制数10011001B，11110000b等。

尽管计算机采用的二进制数的表示方法及运算规则很简单，但书写冗长、不直观且易出错，因此计算机的输入输出仍然采用人们习惯的十进制数。当然，十进制数在计算机中也需要用二进制编码表示。这种编码有多种形式，BCD(Binary-Coded Decimal)码比较常用。4位二进制有16种组合态，当用来表示十进制数时，要舍去6种组合态。常用的8421BCD

码，它的编码规则见表 1-1。

表 1-1 BCD 码表

十进制数	8421 码	十进制数	8421 码
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

通常 BCD 码有两种形式，即压缩 BCD 码和非压缩 BCD 码。

(1)压缩 BCD 码

压缩 BCD 码的每一位用 4 位二进制数表示，一个字节表示两位十进制数。例如，10010110B 表示十进制数 96D。

(2)非压缩 BCD 码

非压缩 BCD 码用一个字节表示一位十进制数，高 4 位总是 0000，低 4 位的 0000～1001 表示 0～9。例如，00001000B 表示十进制数 8D。

1.1.2 十六进制数

十六进制数(Hexadecimal)的基数为 16，有 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F(其中数码 A～F 也可小写)共 16 位数码，计数时遵循逢 16 进 1 的规则，第 k 位的权值以 16^k 表示。因此，十六进制数：

$$a_n a_{n-1} \cdots a_1 a_0 \mathrm{H}$$

等于十进制数：

$$a_n \times 16^n + a_{n-1} \times 16^{n-1} + \cdots + a_1 \times 16^1 + a_0 \times 16^0$$

在汇编语言程序的书写及显示中常常使用十六进制数，主要因为每一位十六进制数码和 4 位二进制数存在一一对应的关系，使得十六进制和二进制之间的转换非常方便且直观，以至于在汇编语言程序中，十六进制的使用频率远远高于二进制。如在 DEBUG 调试程序(汇编语言程序设计中最常用的工具软件，详情可参见本书第 2 章的相关内容)中，所有显示及输入的数据都默认为十六进制。一般我们在书写十六进制数据时，末尾跟一个 H(h)，当十六进制数以 A～F 开头时，一般在前面添加 0，如十六进制数 1A5BH，0ffffh 等都是 4 位的十六进制的数。

1.1.3 十进制数

十进制数(Decimal)的基数为 10，有 0、1、2、3、4、5、6、7、8、9 十位数码。计数时遵循逢 10 进 1 的规则，第 k 位的权值为 10^k。十进制是最常用的计数进制，在汇编语言程序设计中也常常使用十进制数，书写时可以以 D 结尾，也可以不写，如 1234，9236D 都是十进制数数据。

1.1.4 数制间的转换

在用汇编语言进行程序设计时，常常需要在三种进位记数制之间进行转换，其中二进制转换成十进制，十六进制转换为十进制，前已述及。

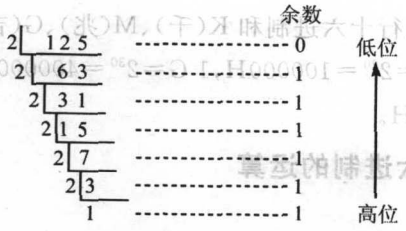
十进制转换为二进制或十六进制，整数部分采用“除基取余法”，即：将十进制整数及此期间产生的商逐次除以 2 或 16。直到商为零为止，并记下每一次相除所得到的余数。按从后往前的次序将各余数记作 $L_n L_{n-1} L_{n-2} \cdots L_0$，从而构成转换后对应的二进制或十六进制

整数。

第一次得到的余数为二进制或十六进制数的最低位，最后一次得到的余数为二进制或十六进制数的最高位。

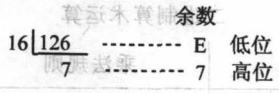
【例 1-1】将十进制整数 126 分别转换为二进制及十六进制数。

解：首先把十进制整数 126 转换为二进制数，采用“除 2 取余法。”



故 126＝1111110B。

然后把十进制整数 126 转换为十六进制数，采用“除 16 取余法”。



故 126＝7EH。

二进制和十六进制之间的相互转换，是汇编语言程序设计中经常遇到的，由于十六进制的基数 16 是二进制的基数 2 的 4 次幂，所以十六进制的每一位和二进制的 4 位形成一一对应的转换关系，具体对应情况如表 1-2。

表 1-2　常用整数各数制间的对应关系表

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

【例 1-2】将二进制数 1111010B 转换为十六进制数。

解：将二进制数 1111010B 从低位到高位 4 位一组，不足部分补 0，根据表 1-2 可知：

0111　1010 B

7　　A　H

故 1111010B＝7AH。

在汇编语言中，还常常进行十六进制和 K(千)、M(兆)、G(吉)这些内存容量单位之间的转换，1 K＝2^{10}＝400H，1 M＝2^{20}＝100000H，1 G＝2^{30}＝40000000H，所以 2 K＝800H，64 K＝10000H，128 M＝8000000H。

1.1.5　二进制和十六进制的运算

1. 二进制数的运算

在汇编语言程序设计中常常需要对二进制数进行各种算术及逻辑运算，其运算规则如下：

二进制算术运算

加法规则	减法规则	乘法规则	除法规则
0＋0＝0	0－0＝0	0×0＝0	0/0 除法溢出
0＋1＝1	0－1＝1(借位 1)	0×1＝0	0/1＝0
1＋0＝1	1－0＝1	1×0＝0	1/0 除法溢出
1＋1＝0(进位 1)	1－1＝0	1×1＝1	1/1＝1

二进制逻辑运算

逻辑与	逻辑或	逻辑非
0∧0＝0	0∨0＝0	$\overline{0}$＝1
0∧1＝0	0∨1＝1	$\overline{1}$＝0
1∧0＝0	1∨0＝1	
1∧1＝1	1∨1＝1	

其中除法溢出表示除法运算的商太大，超过了机器能够表达的范围，除法溢出的后续处理随具体的应用环境不同而不同。

2. 十六进制数的运算

十六进制数的算术运算规律和十进制类似，可以直接进行运算，只是要按照逢 16 进 1，借 1 当 16 的运算规则而已。但十六进制数的除法运算和二进制一样，有可能出现除法溢出。

【例 1-3】计算 2A6BH＋0B9D7H。

解：

　　2 A 6 B H

＋　B 9 D 7 H

0 E 4 4 2 H

【例 1-4】计算 9FA3H－73DCH。

解：

$$\begin{array}{r} 9FA3H \\ -\ 73DCH \\ \hline 2BC7H \end{array}$$

【例 1-5】计算 2A6DH×0081H。

解：

$$\begin{array}{r} 2A6DH \\ \times\ 0081H \\ \hline 2A6DH \\ +15368H \\ \hline 1560EDH \end{array}$$

【例 1-6】计算 2A8EH/1BH。

解：

$$\begin{array}{r|l} & 193H \\ \hline 1BH & 2A8EH \\ & 1BH \\ \hline & F8EH \\ & F3H \\ \hline & 5EH \\ & 51H \\ \hline & DH \end{array}$$

故 2A8EH/1BH＝193H 余 0DH。

二进制的除法可以参照乘法和减法的规则处理，这里不再赘述。

1.2　汇编语言程序设计中数据的表示

在汇编语言程序设计中使用的基本数据主要是整数数值数据和字符，下面分别介绍这两种基本数据在机器中的表示。

1.2.1　整数数值数据的表示

所谓数值数据，就是表示数值大小的数据。由于汇编语言中使用的主要是整数形式的数值数据，故这里只讨论整数数值数据在机器中的表示。

1. 无符号整数

无符号整数中，所有位都是数值位，数值本身只可能是正数，故没有必要保留符号位，如内存单元的物理地址，接口中的端口号等。n 位无符号整数 M 的表数范围为 $0 \leqslant M \leqslant 2^n-1$，如 8 位无符号整数的表数范围是 $0 \leqslant M \leqslant 255$，16 位无符号整数的表数范围是 $0 \leqslant M \leqslant 65535$。

2.有符号整数

有符号整数有正负之分，其值称为真值，如+12，-13ADH 等，当有符号整数在机器中表示时，一般用最高有效位来表示数的符号，正数用 0 表示，负数用 1 表示。有符号整数在机器中的表示形式称为机器值，具体的形式可以是原码、反码、补码、移码等。由于在计算机中，用补码表示有符号整数时，进行算术运算时在电路上更易于实现，几乎所有计算机的有符号整数表示都采用补码，本节重点介绍补码。

(1)补码表示法

正数 X 的补码采用符号绝对值法。即补码的最高位(符号位)为 0，后紧跟该正数的绝对值，总位数为机器的字长 n，即$[X]_{补}=0|X|B$。

例如：当机器的字长为 8 时，$[+1]_{补}=00000001B=01H$；

$[+127]_{补}=01111111B=7FH$。

当机器的字长为 16 时，$[+1]_{补}=0000000000000001B=0001H$；

$[+127]_{补}=0000000001111111B=007FH$；

$[+32767]_{补}=0111111111111111B=7FFFH$。

负数 X 的补码采用$2^n-|X|$表示，总位数为机器的字长 n。

【例 1-7】设机器的字长为 16，分别求-1 和-100 的补码。

解：$[-1]_{补}=2^{16}-|-1|=1111111111111111B=0FFFFH$

$[-100]_{补}=2^{16}-|-100|=1111111110011000B=0FF98H$。

二进制表示的补码，最高位(符号位)为 0 则为正数，最高位(符号位)为 1 则为负数。十六进制表示的补码，最高位以 0～7 开头则为正数，最高位以 8～F 开头则为负数。

(2)补码的符号扩展

所谓补码的符号扩展，是指补码表示的有符号数从 n_1 位变为 n_2 位时，($n_1<n_2$)为了保证数值本身不变，必须在其前面添加 n_2-n_1 位符号位。

例如：$[-127]_{补}=10000001B$(8 位补码)$=81H$(8 位补码)

$\xrightarrow{符号扩展}1111111110000001B$(16 位补码)

$=0FF81H$(16 位补码)。

其中，由 8 位补码变成等值 16 位补码的过程即为符号扩展。

两个补码表示的数不等长，当它们进行算术运算时，位数较少的数应符号扩展到和位数较多的数等长，不然，运算的结果将会出错。

例如：$[-1]_{补}=0FFH$　　$[+5]_{补}=0005H$

$[-1]_{补}+[+5]_{补}=0FFH+0005H=0104H$

$=[+260]_{补}$。

结果出错，因为$[+5]_{补}$是 16 位，而$[-1]_{补}$的补码是 8 位，没有进行符号扩展，正确的运算应该是：

$[-1]_{补}=0FFH\xrightarrow{符号扩展}0FFFFH$

$[-1]_{补}+[+5]_{补}=0FFFFH+0005H=0004H$

$=[+4]_{补}$。

(3)补码的表数范围

n 位补码的表数范围是$-2^{n-1}\leqslant X\leqslant 2^{n-1}-1$，表 1-3 是 8 位和 16 位补码的表数范围。

表 1-3　8 位和 16 位补码的表数范围

真值	8 位补码	真值	16 位补码
+127	7FH	+32767	7FFFH
+126	7EH	+327666	7FFEH
⋮	⋮	⋮	⋮
+2	02H	+2	0002H
+1	01H	+1	0001H
0	00H	0	0000H
−1	0FFH	−1	0FFFFH
−2	0FEH	−2	0FFFEH
⋮	⋮	⋮	⋮
−126	82H	−32766	8002H
−127	81H	−32767	8001H
−128	80H	−32768	8000H

(4)补码的加法和减法

在介绍补码的加法和减法时，先介绍求补运算。对一个二进制数进行按位取反，末位加 1 的运算叫求补运算。可以证明，对一个补码表示的数进行求补运算，结果是其相反数的补码，即$([X]_{补})$求补$=[-X]_{补}$。

例如：

$[-2]_{补}$为　1111 1110B

按位取反后得　0000 0001B

末位加 1 后得　0000 0010B

这正是+2 的补码。

求补运算在补码的运算中很有用。

注意：补码的加法规则：$[X+Y]_{补}=[X]_{补}+[Y]_{补}$。

【例 1-8】已知$[-125]_{补}=83H$，$[-3]_{补}=0FDH$，求$[(-125)+(-3)]_{补}$。

解：$[(-125)+(-3)]_{补}=[-125]_{补}+[-3]_{补}=83H+0FDH=80H$。

注意：补码的减法规则是：$[X-Y]_{补}=[X]_{补}+[-Y]_{补}=[X]_{补}+([Y]_{补})$求补。

【例 1-9】已知$[-125]_{补}=83H$，$[+3]_{补}=03H$，求$[(-125)-(+3)]_{补}$。

解：$([+3]_{补})$求补$=(03H)$求补$=0FDH$，

$$[(-125)-(+3)]_{补}=[-125]_{补}+[-3]_{补}$$
$$=[-125]_{补}+([+3]_{补})\text{求补}$$
$$=83H+0FDH=80H。$$

1.2.2　字符的编码

计算机处理的信息不一定都是数值数据，有时是字符或字符串。在 Intel 80x86 机器