普通高等教育“十一五”国家级规划教材
普通高等教育电子通信类国家级特色专业系列规划教材

微机原理与接口技术

（第二版）

楼顺天　周佳社　张伟涛　编著

科学出版社

普通高等教育“十一五”国家级规划教材
普通高等教育电子通信类国家级特色专业系列规划教材

微机原理与接口技术

（第二版）

楼顺天　周佳社　张伟涛　编著

科学出版社

北京

内 容 简 介

本书以 Intel 公司生产的 8086/8088 CPU 为核心，详细介绍汇编语言程序设计技术、系统总线形成、存储器设计、常用和专用芯片的接口技术及其应用编程方法。

在汇编语言程序设计中，分别介绍计算机中的数制和码制、补码的运算规则、数据和转移地址的寻址方式、8086/8088 的指令系统，着重介绍汇编语言的编程技术，并结合示例介绍许多实际应用编程技巧，强调汇编语言中指针的使用。在接口技术中，介绍 8086/8088 系统总线的形成、常用芯片与系统总线的接口、专用芯片的接口与工作方式控制、中断技术及其应用，重点介绍存储器的设计和专用芯片的应用设计，结合示例介绍一些实际应用系统的设计方法。

本书第二版经过较大篇幅的修改充实了内容，补充了大量的习题。本书可作为高等院校各专业本科生的教材，也可供相关工程技术人员、管理人员和自学者参考。

图书在版编目（CIP）数据

微机原理与接口技术/楼顺天，周佳社，张伟涛编著. —2 版 .—北京：科学出版社，2015.5

普通高等教育“十一五”国家级规划教材

ISBN 978-7-03-044464-6

Ⅰ. ①微… Ⅱ. ①楼… ②周…③张… Ⅲ. ①微型计算机-理论-高等学校-教材 ②微型计算机-接口技术-高等学校-教材 Ⅳ. ①TP36

中国版本图书馆 CIP 数据核字（2015）第 114357 号

责任编辑：匡 敏 潘斯斯 张丽花 / 责任校对：桂伟利

责任印制：霍 兵 / 封面设计：迷底书装

科学出版社出版

北京东黄城根北街16号

邮政编码：100717

http://www.sciencep.com

文林印务有限公司印刷

科学出版社发行 各地新华书店经销

*

2006 年 8 月第 一 版 开本：787×1092 1/16

2015 年 6 月第 二 版 印张：23 1/2

2015 年 6 月第十五次印刷 字数：557 000

定价：49.00 元

（如有印装质量问题，我社负责调换）

前　言

在大多数高等院校的本科生教学中，计算机课程的设置和教学实施得到了广泛重视，一般都开设了“计算机文化基础”、“C 语言程序设计”、“软件技术基础”、“微机原理与接口技术”（或“微机原理与系统设计”）等课程，本书就是专门为“微机原理与接口技术”（或“微机原理与系统设计”）课程所编写的本科生教材，适合各个专业使用。

微机原理与接口技术课程是重要的专业基础课，是学生参加全国、省三级计算机等级考试（PC 技术）的重要课程，也是电路、信号与系统、信号处理、自动控制等硕士学科的考研科目之一。这门课程是“计算机文化基础”、“C 语言程序设计”的继续，又是“数据结构”、“软件技术基础”、“计算机网络”等课程的基础，在计算机教学中占据重要地位。

本书作者均有十几年“微机原理与系统设计”课程的教学经验，多年来一直承担着研究生试卷的命题任务，完成了十多项相关的科研项目，自主开发了“8086 单板型微型计算机实验教学系统”和“基于 EISA/PCI 系统总线的微机原理及接口技术实验教学系统”，并有出版 10 多本教材和专著的写作经验。本书具有下列特点：

（1）从学生学习、教师讲授的角度出发，内容由浅入深，循序渐进，前后连贯并呼应，使结构系统而完整。

（2）内容层次分明，既有基础知识的系统介绍，又有拓宽知识的基本介绍，给学生留下广阔的思维空间。

（3）理论联系实际，在知识介绍的同时，结合实际示例，采用面向应用的启发式教学方法。

（4）以编程思路为主线，介绍汇编语言的程序设计方法，让学生切实掌握编程知识。

（5）以实际应用设计为主线，介绍微机系统的接口设计技术。

（6）将工程设计中常用的实例、常见问题解决方法等作较全面的总结，提高了本书的实用价值。

（7）写作语言流畅，内容选择合理，结构编排适当。

本书共 11 章，由楼顺天、周佳社、张伟涛共同编写，楼顺天负责统稿。其中第1～4章由楼顺天编写，第 5～8 章由周佳社编写，第 9～11 章由张伟涛编写。

第 1～4 章构成 8086 汇编语言程序设计技术，重点介绍 8086 的指令系统及其编程方法。第 5～11 章构成了 8086 的系统总线及其接口设计技术，重点介绍 8086/8088 系统的总线形成、存储器设计、常用芯片接口设计、专用芯片（8259A、8253、8255A）的接口设计、实际应用接口（D/A、A/D、非编码矩阵键盘、LED 数码显示、光电隔离、步进

电机等）的设计及其应用编程。

为方便教师授课和学生学习，附录 A 列出 8086/8088 的指令系统，附录 B 给出 DOS 中断 INT 21H 的部分功能列表，附录 C 是本书的例题索引。

感谢西安电子科技大学的有关老师给予本书写作的支持，感谢参与本书绘图和校稿的研究生，更要感谢科学出版社的编辑给予细致的润色加工及发行同志为本书的推广所作的辛苦努力。

由于作者水平和知识面的限制，书中难免会有错误和欠妥之处，敬请读者批评指正，以便及时更正，使本书更有益于广大的读者。

编　者

2015 年 5 月

目　录

第 1 章　数制与码制

1.1　数制表示及其转换

1.1.1　数制的表示

因为人有 10 根手指，所以自古以来就习惯使用 10 根手指来计数，因此逢十进一的十进制系统很自然就成为人类常用的计数方法。

数制是以表示数值所用的数字位数来命名，例如，十进制用 10 位数字（0～9）表示，二进制用 2 位（数字 0、1）表示，十六进制用 10 位数字和 6 位符号（A、B、C、D、E、F）表示。各种数制中数字或符号的个数称为数制的基数。

任意进制数可以通过多项式形式表示，设数制的基数为 b，则数 x 可以表示成

$$x=\sum_{i=-m}^{n}k_ib^i=k_{-m}b^{-m}+\cdots+k_{-1}b^{-1}+k_0+k_1b+\cdots+k_nb^n \tag{1.1}$$

其中 k_{-m}，…，$k_n\in[0,1,\cdots,b-1]$，m、n 为非负整数。上式表示数 x 可以表示成 b 进制数，整数 $n+1$ 位，小数 m 位。这一式子也称为数值的按权值表示。

（1）十进制。

常用的十进制数可以直接用 0～9 数字表示，也可以在数字后加 D（decimal）表示，例如，257 和 369.2D 可以按式（1.1）分别表示成

$$257=2\times10^2+5\times10+7\times10^0$$

$$369.2\mathrm{D}=3\times10^2+6\times10+9\times10^0+2\times10^{-1}$$

（2）二进制。

在数字计算机中，经常用二进制数来表示数值，这是因为在数字电路中，只能用高电平和低电平表示不同的事件。二进制数可以用 0～1 数字后加 B（binary）表示，例如，10101B 可以按权值展开成

$$10101\mathrm{B}=1\times2^4+0\times2^3+1\times2^2+0\times2^1+1\times2^0$$

这里介绍几个常用的术语：

位（bit）：二进制位，只有两种状态：0 和 1，它是计算机中存储信息的最小单位。

字节（byte）：8 个二进制位，可以存储 8 位二进制数，如果是无符号数，则其范围为 0～255。通常计算机中的存储单元按字节设置，也就是说，8086 微机系统中可以访问的最小存储单元为一个字节，1 byte=8 bit。

字（word）：16 个二进制位，2 个字节，可以存储 16 位二进制数，如果是无符号数，则其范围为 0～65535。

双字（double word）：32 个二进制位，2 个字，4 个字节，可以存储 32 位二进制数，如果是无符号数，则其范围为 0～4294967295。

字长：基本数据单元所包含的二进制位数，8086 微处理器中经常采用的字长为 8 和 16。

（3）十六进制。

为书写表示方便，通常将 4 位二进制数看作为 1 位十六进制数，这时用数字 0～9 和符号 A～F 表示，并在十六进制数后加 H（heximal）表示。在书写十六进制数时，如果最高位是字符，则在其前面要加上 0，以便与标识符区别开来。这样我们有

0AH＝1010B　　0BH＝1011B　　0CH＝1100B

0DH＝1101B　　0EH＝1110B　　0FH＝1111B

十六进制数也可以表示成权值展开形式，如

$$325\mathrm{H}=3\times16^2+2\times16+5\times16^0$$

$$0\mathrm{B6H}=11\times16+6\times16^0$$

1.1.2　数制的转换

同一个数值可以用各种数制来表示，各种数制表示的数值之间可以进行转换。

1. 十进制数转换成其他进制数

将十进制数 N 分成两部分：整数部分 z 和纯小数部分 f。设要将十进制数 $z.f$ 转换成 b 进制数，则整数部分 z 采用除 b 取余的方法，即有

$$z_1=\left[\frac{z}{b}\right]\cdots\cdots y_1$$

$$z_2=\left[\frac{z_1}{b}\right]\cdots\cdots y_2$$

$$\vdots$$

$$z_n=\left[\frac{z_{n-1}}{b}\right]\cdots\cdots y_n$$

其中 z_1，…，z_n 为商，y_1，…，y_n 为余数，［·］表示取整运算。当 $z_n=0$ 时迭代过程终止，这样得到的 y_1，…，y_n 就是 b 进制数的各位数字，y_1 为最低位，y_n 为最高位。

例 1.1　将十进制数 125 转换成二进制数。

解：转换过程为

2 | 125

2 | 62 ……1

2 | 31 ……0

2 | 15 ……1

2 | 7 ……1

2 | 3 ……1

2 | 1 ……1

0 ……1

因此，125＝1111101B。

纯小数部分 f 采用乘以 b 取整的方法，即有

$$
\begin{aligned}
r_1 &= [f \times b] & f_1 &= f \times b - r_1 \\
r_2 &= [f_1 \times b] & f_2 &= f_1 \times b - r_2 \\
&\vdots \\
r_m &= [f_{m-1} \times b] & f_m &= f_{m-1} \times b - r_m
\end{aligned}
$$

其中 r_1，…，r_m 为取整结果，f_1，…，f_m 为小数部分。当 $f_m=0$ 时迭代过程终止，这样得到的 r_1，…，r_m 就是 b 进制数的各位数字，r_1 为最高位，r_m 为最低位。

例 1.2 将十进制数 0.6875 转换成二进制数。

解：转换过程为

$$
\begin{array}{rl}
0.6875 & \\
\times \quad 2 & \\
\hline
1.3750 & \cdots\cdots\cdots\cdots\cdots\cdots \ 1 \\
0.375 & \\
\times \quad 2 & \\
\hline
0.750 & \cdots\cdots\cdots\cdots\cdots\cdots \ 0 \\
\times \quad 2 & \\
\hline
1.50 & \cdots\cdots\cdots\cdots\cdots\cdots \ 1 \\
0.5 & \\
\times \quad 2 & \\
\hline
1.0 & \cdots\cdots\cdots\cdots\cdots\cdots \ 1
\end{array}
$$

因此，0.8125＝0.1011B。

2. 其他进制数转换成十进制数

将任意进制数变换成十进制数，可以按照权值表示进行展开，例如

$$
\begin{aligned}
10110110\text{B} &= 1\times 2^7 + 1\times 2^5 + 1\times 2^4 + 1\times 2^2 + 1\times 2^1 \\
&= 128 + 32 + 16 + 4 + 2 = 182 \\
1011.011\text{B} &= 1\times 2^3 + 1\times 2^1 + 1\times 2^0 + 1\times 2^{-2} + 1\times 2^{-3} \\
&= 8 + 2 + 1 + 0.25 + 0.125 = 11.375 \\
78.\text{AH} &= 7\times 16^1 + 8\times 16^0 + 10\times 16^{-1} \\
&= 120.625
\end{aligned}
$$

1.2 二进制数的运算规则

1.2.1 二进制数的算术运算

十进制数的运算规则是我们所熟悉的。计算机中是以二进制数来表示的，为书写方

便，经常写成十六进制数。因此这里主要讨论二进制数和十六进制数的算术运算规则。其他进制数的运算规则与十进制数类似，二进制数加法运算采用逢二进一，减法运算采用借一作二；十六进制数加法运算采用逢十六进一，减法运算采用借一作十六，在乘除法运算时，也采用类似的规则。例如

1011011B＋10011B＝1101110B
1011B×10011B＝11010001B
65H＋7AH＝0DFH
65H×7AH＝3022H

1.2.2 二进制数的逻辑运算

二进制数的逻辑运算是位对位的运算，即本位运算结果不会对其他位产生任何影响，这一点与算术运算是截然不同的。二进制数的逻辑运算有四种：与（AND）、或（OR）、异或（XOR）、非（NOT），其规则如表 1.1 所示。

表 1.1 二进制数位的逻辑运算规则

输入 2 个位值	(0，0)	(0，1)	(1，0)	(1，1)
AND 运算	0	0	0	1
OR 运算	0	1	1	1
XOR 运算	0	1	1	0
NOT 运算	NOT 0＝1		NOT 1＝0	

例如

10010111B AND 00111000B＝00010000B
10010111B OR 00111000B＝10111111B
10010111B XOR 00111000B＝10101111B

利用逻辑运算可以完成特定的操作，AND 运算可以对指定位进行清 0，OR 运算可以对指定位进行置 1，XOR 运算可以对指定位进行取反。例如，对 x 的第 0、3 位清零操作：x AND 11110110B，对 x 的第 1、2 位置 1 操作：x OR 00000110B，对 x 的第 3、7 位取反操作：x XOR 10001000B。

1.3 有符号数的表示

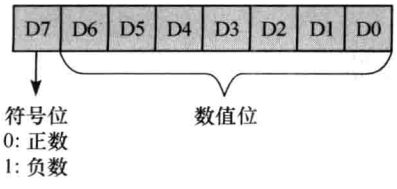

图 1.1 有符号数的表示

利用二进制数来表示有符号数时，必须有一位用来表示符号位，一般采用最高位表示，如图 1.1 所示。这样表示的数称为机器数，其实际值称为机器数的真值。

1.3.1 原码表示法

除符号位外，剩余 7 位就是真值的绝对位，这种

表示方法称为原码表示法。例如，+1011001B表示成 01011001B，－1011001B 表示成 11011001B，这种表示方法的优点是直观，但加减运算时比较麻烦。例如，在对两个数进行加法运算时，应该先对其符号进行判断，如果同号，则进行相加运算，如果异号，则实际上应该进行相减运算。

另外，对于特殊值 0 有两种表示：+0 表示成 00000000B，－0 表示成10000000B，但实际上，00000000B 和 10000000B 表示同一个值 0。

1.3.2 补码表示法

计算机中有符号二进制数采用补码表示，x 的补码表示 $[x]_{补}$定义为

$$[x]_{补}=\begin{cases} x, & \text{当 } 0\leqslant x<2^{n-1} \text{ 时,} \\ x+2^{n}, & \text{当 } -2^{n-1}\leqslant x<0 \text{ 时,} \end{cases} \quad (\mathrm{mod}2^{n})$$

求一个数 x 的补码，可以表示成 $[x]_{补}$，这种过程称为求补运算。从定义可以看出，当 x 为正数时，其原码与补码一致，只有当 x 为负数时，才有求补码的问题。

当 $n=8$ 时，有符号二进制数的表示范围为－128～127，当 $n=16$ 时，有符号二进制数的表示范围为－32768～32767。例如，+12 的原码和补码表示均为 00001100B，而－12 的补码表示为 $[-12]_{补}=-12+2^{8}=244=11110100$B，实际上，它就是原码 10001100B 按位取反（符号位除外）再加 1 的结果。

因此负数的补码为原码取反加 1（符号位除外），即当 x 为负数时，有

$$[x]_{补}=\overline{[x]_{原}}+1 \quad \text{（除符号位）}$$

如果对已经表示成补码的数 $[x]_{补}$ 再求补码，则可以得到其原码表示，即

$$[[x]_{补}]_{补}=[x]_{原}$$

实际上，对负数 x 求补码时，只需要先求出（$-x$）的原码，然后按位取反再加 1，即

$$[x]_{补}=\overline{[-x]_{原}}+1 \quad \text{（含符号位）}$$

这为求负数补码运算提供了简捷的运算方法。

例 1.3 求－15 的补码表示。

解：分两步进行。

（1）$[15]_{补}=00001111$B。

（2）$[-15]_{补}=\overline{0000\ 1111}$B+1=11110001B。

1.4 有符号数的运算及其溢出规则

1.4.1 补码运算规则

有符号二进制数以补码形式表示以后，可以直接进行加减法运算，并满足下列规则：

（1）加法 $[x+y]_{补}=[x]_{补}+[y]_{补}$ $(\mathrm{mod}2^{n})$

该式表明：当带符号数的两个数采用补码形式表示时，进行加法运算可以把符号位和

数值位一起进行运算（若符号位有进位，则丢掉），其结果为两数之和的补码形式。例如

(+57)+(+45)= 00111001B+00101101B= 01100110B （+102）

(+57)+(−45)= 00111001B+11010011B= [1]00001100B （进位舍弃，+12）

(−57)+(−45)= 11000111B+11010011B= [1]10011010B （进位舍弃，−102）

（2）减法 $[x-y]_{补}=[x]_{补}-[y]_{补}\ (\mathrm{mod}2^n)$

$$
\begin{aligned}
(+57)-(+45) &= 00111001\mathrm{B}-00101101\mathrm{B}\\
&= 00001100\mathrm{B}\quad(+12)\\
(+57)-(-45) &= 00111001\mathrm{B}-11010011\mathrm{B}\\
&= [1]01100110\mathrm{B}\quad(\text{借位舍弃，}+102)\\
(-57)-(-45) &= 11000111\mathrm{B}-11010011\mathrm{B}\\
&= [1]11110100\mathrm{B}\quad(\text{借位舍弃，}-12)
\end{aligned}
$$

（3）用加法完成相减运算 $[x-y]_{补}=[x]_{补}+[-y]_{补}\ (\mathrm{mod}2^n)$

已知 $[y]_{补}$ 而求 $[-y]_{补}$ 的过程称为变补或求负。其规则为：对 $[y]_{补}$ 的每一位（包括符号位）进行按位取反，然后再加 1，其结果即为 $[-y]_{补}$。例如，+87 的补码表示为 01010111B，而−87 的补码就可以这样计算：$[-87]_{补}=\overline{01010111\mathrm{B}}+1=10101001\mathrm{B}$。这样就又提供了一种求负数补码的方法。

（4）加法与减法互换 $[x]_{补}-[y]_{补}=[x]_{补}+[-y]_{补}\ (\mathrm{mod}2^n)$

应注意，一旦采用补码进行加减运算，所有参加运算的数及结果都是用补码表示的。计算机里的实际情况就是这样。

总结：求负数补码的三种方法：

（1）按照定义。

（2）$[x]_{补}=\overline{[x]_{原}}+1$（符号位除外）。

（3）先求−x 的补码，然后$[x]_{补}=\overline{[-x]_{补}}+1$（含符号位）。

1.4.2 有符号数运算时的溢出问题

设计算机字长为 n 位，则有符号数的范围为

$$-2^{n-1}\leqslant x\leqslant+2^{n-1}-1$$

当 $n=8$ 时，其有符号数的范围为$-128\leqslant x\leqslant+127$；当 $n=16$ 时，有符号数的范围为$-32768\leqslant x\leqslant+32767$。

当两个有符号数进行加减运算时，如果运算结果超出可表示的有符号数的范围时，就会发生溢出，这时结果就会出错。溢出发生在两种情况下：两个同符号数相加，两个异符号数相减。

有符号数的溢出规则为：

（1）在加法运算时，如果次高位（数值最高位）相加形成进位，而最高位（符号位）相加（包括次高位的进位）却没有进位时，则结果溢出；或者相反，如果次高位无进位，而最高位有进位，则结果溢出。

（2）在减法运算时，如果次高位不需借位，而最高位需借位，则结果溢出；或者相

反，如果次高位需借位，而最高位不需借位，则结果溢出。

在微机系统中，当加减运算溢出时，溢出标志位 OF 会自动置 1。

例 1.4 计算（+121）+（+75）和（−121）−（+75），并判断有无溢出。

解：（+121）+（+75）=01111001B+01001011B

=11000100B （−60 有溢出）

（−121）−（+75）=10000111B−01001011B=00111100B （60 有溢出）

1.5 BCD 编码方法及其运算

用 4 位二进制数来表示一位十进制数，这种方法称为 BCD 编码方法。4 位二进制数有 16 种组合，如果取前 10 种组合分别表示 0～9，这称为 8421BCD 码。

一般来说，8 位二进制数（一个字节）可以表示两位十进制数，这种表示方法称为组合 BCD 数，如果 8 位二进制数只表示一位十进制数，则称为分离BCD 数。

当计算机中两个 BCD 数进行运算时，由于 CPU 只能实现二进制数的运算，因此 BCD 数的运算结果可能出错。这是因为在 BCD 码加法运算时，应该是逢十进一，而 CPU 则逢十六进一（4 位二进制数），因此，在某些位应作“加 6 修正”，BCD 码加法运算的修正规则为：

（1）两个 BCD 码位相加无进位，并且结果少于或等于 9，则该位不需要修正。

（2）两个 BCD 码位相加有进位，或者结果大于 9，则该位应作加 6 修正。

（3）低 BCD 码位修正结果使高 BCD 码位大于 9，则高位进行加 6 修正。

同样，BCD 码的减法运算时，需要对 BCD 码位进行“减 6 修正”，其修正规则为：

（1）两个 BCD 码位相减无借位，则该位不需要修正。

（2）两个 BCD 码位相减有借位，则该位应作减 6 修正。

例 1.5 已知 $(56)_{BCD}$=01010110B，$(38)_{BCD}$=00111000B，计算这两个 BCD 数的加、减运算，并进行适当的修正。

解： $(38)_{BCD}+(56)_{BCD}$=00111000B+01010110B

=10001110B(高、低 BCD 码位都没有进位，

但低 BCD 码超出 9)+0110B

=10010100B （结果为 94，正确）

$(56)_{BCD}-(38)_{BCD}$=01010110B−00111000B

=00011110B(低 BCD 码位有借位)−0110B

=00011000B （结果为 18，正确）

1.6 ASCII 编码方法

由于数字计算机只能识别和处理由 1 和 0 组成的二进制代码（高低电平分别表示 1 和 0），因此计算机中用来表示信息的字母、数字以及一些常用的符号都应该用二进制代码来

表示，这就涉及编码方法的问题。

信息编码方法并不是唯一的，为了实现正常的信息交换，通信双方应该使用统一的编码规则，最常用的编码方法是 ASCII 码（American Standard Code for Information Interchange），即美国国家信息交换标准码。它是美国国家标准协会提出的一种单字节字符编码方案，主要用于西文字符的编码。它不仅是美国国家标准码，而且已被国际标准化组织定为国际标准。

ASCII 码如表 1.2 所示。需要指出的是，表 1.2 中前 128 个字符对应的编码（00H～7FH）是标准 ASCII 码，而后 128 个字符对应的编码（80H～FFH）是扩展 ASCII 码。其中标准 ASCII 码是国际标准，而扩展 ASCII 码已不再是国际标准了。

表 1.2　ASCII 码表

十进制值	➡	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
⬇	十六进制值	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	空位（零位）	▶	空位（空间）	0	@	P	·	p	C	É	á	▒	└	╨	∝	≡
1	1	☺	◀	!	1	A	Q	a	q	ü	æ	í	▓	┴	╤	β	±
2	2	☻	↕	"	2	B	R	b	r	é	Æ	ó	▓	┬	╥	Γ	≥
3	3	♥	!!	#	3	C	S	c	s	â	ô	ú	┃	├	╙	Π	≤
4	4	◆	¶	$	4	D	T	d	t	ä	ö	ñ	┤	─	╘	∑	⌠
5	5	♣	§	%	5	E	U	e	u	à	ò	Ñ	╡	┼	╒	O	⌡
6	6	♠	▬	&	6	F	V	f	v	å	ō	a	╢	╞	╓	⌠⌡	÷
7	7	○	↨	'	7	G	W	g	w	ç	ù	Ō	╖	╟	╫	τ	≈
8	8	◘	↑	(	8	H	X	h	x	ê	ÿ	G	╕	╚	╪	Φ	°
9	9	○	↓	)	9	I	Y	i	y	ë	Ö	┌	╣	╔	┘	Θ	•
10	A	▩	→	*	:	J	Z	j	z	è	Ü	¬	‖	╩	┌	Ω	•
11	B	♂	←	+	;	K	[	k	{	ï	ç	1/2	╗	╦	■	δ	√
12	C	♀	└	,	<	L	\	l	¦	î	£	1/4	╝	╠	▬	∞	n
13	D	♪	↔	—	=	M	]	m	}	ì	¥	i	╗	═	▌	∅	2
14	E	♫	▲	.	>	N	^	n	~	Ä	₧	«	╝	╬	▄	∈	■
15	F	☼	▼	/	?	O	_	o	△	Å	ƒ	»	¬	╧	▀	∩	空 FFH

标准 ASCII 码用 7 位二进制数来表示 128 个字符，这 128 个字符包括非显示字符和显示字符两部分，其中非显示字符有 33 个（00H～1FH 和 7FH），主要控制字符和通信专用字符，它们并没有对应的可显示图形，但在应用程序中对文本显示有特定的控制作用，如回车（CR）、换行（LF）、退格（BS）等。可显示字符有 95 个（20H～7EH），主要是数字、大小写字母、标点符号、运算符号以及一些特殊符号。

在汇编语言程序设计中，经常要用到字符的 ASCII 码，常用的符号有：数字 0～9、英文字母 A～Z 和 a～z、空格、回车、换行、Esc 等，这些符号的 ASCII 码必须牢牢记住。

1.7 小　结

在这一章中，从微处理器系统中经常采用的二进制数出发，介绍了十进制数转换为二进制数的方法及二进制数的运算规则，重点介绍了有符号数的两种表示方法：原码表示和补码表示，并讨论了补码运算规则、溢出规则。给出了 BCD 编码方法和 BCD 码运算的修正规则，最后给出了 ASCII 码表。

通过本章的学习，要求学生掌握数制之间的转换、有符号数的补码表示、有符号数运算时溢出判断、BCD 码运算的修正规则及常用符号的 ASCII 码。

习　题

1.1　将下列十进制数转换成二进制数：

(1) 58；　　(2) 67.625；　　(3) 5721；

(4) 31.75；　　(5) 11.3；　　(6) 511。

1.2　将二进制数变换成十六进制数：

(1) 10010101B；　　(2) 1101001011B；　　(3) 1111111111111101B；

(4) 0100000010101B；　　(5) 01111111.11B；　　(6) 010000000001B。

1.3　将十六进制数变换成二进制数和十进制数：

(1) 78H；　　(2) 0A6H；　　(3) 1000H；　　(4) 0FFFFH；

(5) 0F1.2H；　　(6) 3D5H；　　(7) 1A.34H；　　(8) 80H。

1.4　将下列十进制数转换成十六进制数：

(1) 39；　　(2) 299.34375；　　(3) 54.5625；

(4) 85.1875；　　(5) 2048；　　(6) 65535。

1.5　将下列二进制数转换成十进制数：

(1) 10110.101B；　　(2) 10010010.001B；　　(3) 11010.1101B；

(4) 1100.0101B；　　(5) 100011011.011B；　　(6) 11111111B。

1.6　计算（按原进制运算）：

(1) 10001101B+11010B；　　(2) 10111B+11100101B；　　(3) 1011110B−1110B；

(4) 124AH+78FH；　　(5) 5673H+123H；　　(6) 1000H−F5CH。

1.7　已知 a=1011B，b=11001B，c=100110B，按二进制完成下列运算，并用十进制运算检查计算结果：

(1) a+b；　　(2) c−a−b；　　(3) a×b；　　(4) c÷b。

1.8　已知 a=00111000B，b=11000111B，计算下列逻辑运算：

(1) a AND b；　　(2) a OR b；　　(3) a XOR b；　　(4) NOT a。

1.9　设机器字长为 8 位，写出下列各数的原码和补码：

(1) +1010101B；　　(2) −1010101B；　　(3) +1111111B；

(4) −1111111B；　　(5) +1000000B；　　(6) −1000000B。

1.10　写出下列十进制数的二进制补码表示（设机器字长为 8 位）：

(1) 15；　　(2) −1；　　(3) 117；　　(4) 0；

(5) −15；　　(6) 127；　　(7) −128；　　(8) 80。

1.11　设机器字长为 8 位，已知下列两组为 a 和 b 的原码，求 $[a+b]_{补}$ 和 $[a-b]_{补}$，并判断结果是否溢出：