高等学校教学用书

微型计算机技术基础

GAODENG XUEXIAO JIAOXUE YONGSHU

冶金工业出版社

高 等 学 校 教 学 用 书

微型计算机技术基础

武汉钢铁学院　陆忠华　编

冶 金 工 业 出 版 社

高等学校教学用书
微型计算机技术基础
武汉钢铁学院　陆忠华　编
*
冶金工业出版社出版
（北京北河沿大街嵩祝院北巷39号）
新华书店北京发行所发行
冶金工业出版社印刷厂印刷
*
787×1092 1/16　印张 19 1/2　字数 464 千字
1987年 6 月第一版　　1987年 6 月第一次印刷
印数00,001～11,100册
统一书号：15062·4581　　定价3.20元

前　言

二十世纪七十年代初期，由于半导体工艺的发展，大规模集成技术有了新的突破。1971年美国Intel公司首先做成了第一片微处理器Intel4004。从此以后，以各式各样的微处理器为核心的微型计算机系统不断涌现出来。微型计算机具有价格低、体积小、结构简单、使用灵活、易于操作与维修等特点，在普及应用方面有着十分广阔的前景，在实现我国现代化的过程中将得到广泛的应用。

本书主要介绍在我国使用最广泛的Z80微处理器的原理及应用，而以TP801A单板微型计算机作为背景。内容力求简明扼要，先易后难，由浅入深。每章均附有较多数量的例题与习题。根据编者教学实践经验，在叙述方法上采用分层次讲述，使硬件与软件有机结合。

本书共分七章。第一章讲授数制码制及数字逻辑电路方面的有关预备知识。第二章从基本组成原理的角度，概述了电子计算机系统。第三章对Z80微处理器及其指令系统作了较详细的介绍。在此基础上，第四章介绍了Z80汇编语言的基本概念，给出了26个程序设计编程实例，并介绍了浮点运算的若干程序。第五章为微型计算机接口技术基础。在概述输入输出系统的控制方式、联络方式与信息传输方式的基础上，着重介绍了Z80中断系统的基本结构及常用Z80接口芯片，然后通过举例对几种常用外设的接口电路及程序进行了分析。第六章给出了10个由实际系统简化得出的微型计算机应用实例。第七章对三种典型16位微处理器与8048系列单片机作了简单介绍。

本书从软硬结合的角度出发，将“微型计算机组成原理”及“汇编语言程序设计”二门课程的内容有机地结合起来。本书可作高等院校工业电气自动化专业、计算机及应用专业或其它工科专业教材使用。可根据不同要求供60～100学时使用。某些难度较大的章节及习题（在书中用*注明），可酌情选讲及选做。

本书初稿在1980年完成，经多次教学实践后又在1983年进行了较大的修改，并在武汉钢铁学院铅印出版，曾被数十所高等院校采用作教材。1985年4月在武汉召开了教材编审会议，由曾宪昌教授、何文蛟教授、汤之璋教授、康华光教授等10位同志组成评审小组，审定了教材编写大纲。本书系根据本大纲编写的。

在本书编写过程中，曾得到武汉钢铁学院计算机教研室同志的大力支持。陈和平、夏玉勤、乐英杰、尚宏猷等同志仔细审阅了全部书稿。在此，对他们谨致谢意。

由于编者水平有限，书中难免还有些缺点和错误，殷切希望读者批评指正。

陆忠华　于武汉钢铁学院

一九八六年三月

目　　录

1 预 备 知 识

在本章中，我们将对数制与码制及数字逻辑部件的有关预备知识进行简略的复习，主要通过举例来说明问题。

1.1 数制及不同数制间的转换

1.1.1 十进制数

十进制是日常生活中最常用的记数制。它采用0～9等十个数码和一个小数点符号来表示任意十进制数。例如，235.7代表个位为5，十位为3，百位为2，十分位为7的十进制数。这种记数方法称为位置记数法。采用位置记数法时，将各数码按其在数中不同的位置而乘以不同的因子，再相加即得该十进制数的值。这个因子通常被称为“权”。显然，个位、十位及百位的“权”分别为1、10及10^2，而十分位的“权”为10^{-1}。因此，任意十进制数可按“权”展开为10的幂多项式。235.7的多项式表示形式为：

$$2\times10^2+3\times10^1+5\times10^0+7\times10^{-1}$$

式中“10”称为十进制的“基数”。

1.1.2 二进制数及其算术逻辑运算方法

在电子数字计算机中采用双稳态电子元件作为保存信息的基本元件，故要求参加运算的数一律为二进制数。在二进制中，只有0和1两个数码。二进制数也采用位置记数法，其标准表示形式为：

$$D_{n-1}\cdots\cdots D_1D_0\cdot D_{-1}D_{-2}\cdots\cdots D_{-m}$$

按权展开的多项式形式为：

$$D_{n-1}\times2^{n-1}+\cdots\cdots+D_1\times2^1+D_0\times2^0+D_{-1}\times2^{-1}+D_{-2}\times2^{-2}+\cdots\cdots+D_{-m}\times2^{-m}$$

例如，$1011.11=1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}$

二进制数的加法运算按“逢二进一”原则进行，减法运算按“借一作二”原则进行。由此可得二进制加法的公式为：0+0=0；0+1=1；1+1=0（进位1）。二进制减法的规则为：0−0=1−1=0；1−0=1；0−1=1（借位1）。二进制乘法公式为：0×0=0×1=1×0=0；1×1=1。与十进制一样，二进制除法通过试减法求商而没有专门的除法公式。下面举例说明之。

例 1.1 求以下二进制算术运算的结果：

（1）1110+1100；（2）10110−1100；（3）1101×1011；（4）10110001÷111。

解：（1）

$$\begin{array}{r} 1110 \\ +)\ 1100 \\ \hline 11010 \end{array}$$

（2）

$$\begin{array}{r} 10110 \\ -)\ 1100 \\ \hline 1010 \end{array}$$

（3）
```
    1101
  × 1011
  ------
    1101
   1101
  0000
 1101
 ------
 10001111
```

（4）
```
      11001……商
  111 ) 10110001
        111
        ---
        1000
         111
         ---
         1001
          111
          ---
           10……余数
```

二进制数的逻辑运算常用的有逻辑“与”、逻辑“或”及逻辑“异或”等三种，可分别用符号“AND”、“OR”及“XOR”来表示。在很多情况下，为书写方便起见常分别用“·”、“+”及“⊕”来代替，但应避免在逻辑表示式中出现算术加号“+”及乘号“·”，否则将引起混淆。二进制数的逻辑运算过程可通过按位运算来进行。两个二进制位的“与”运算公式为：$1\cdot0=0$，$0\cdot0=0$，$1\cdot1=1$，即两个位均等于1时逻辑“与”结果为1，否则为0。逻辑“或”运算的公式是：$1+0=1$，$1+1=1$，$0+0=0$，即两个位均为0时相“或”结果为0，否则为1，“异或”运算的公式是：$1\oplus1=0$，$0\oplus0=0$，$1\oplus0=1$，即两个位不相等时“异或”结果为1，否则为0。

由于是逻辑运算，而不是算术运算，因此各位间没有横向的联系，即没有类似算术运算的进位或借位关系，这一点应当注意。

例 1.2 求二进制数（10010111）及（00111000）的运算结果：（1）逻辑“与”；（2）逻辑“或”；（3）逻辑“异或”。

解：（1）
```
    10010111
与)  00111000
 ------------
    00010000
```

（2）
```
    10010111
或)  00111000
 ------------
    10111111
```

（3）
```
    10010111
异或)00111000
 ------------
    10101111
```

1.1.3 十六进制数及其算术逻辑运算方法

电子数字计算机要求原始数据为二进制数。二进制有其优点，但它的缺点是书写起来冗长，不直观。因此在计算机有关文本中常用十六进制代替二进制来书写原始数据。十六进制和十进制同样紧凑，同时又具有与二进制间转换关系特别简单的优点。

在十六进制中有0～F等16个数码，其中A、B、C、D、E、F等数码分别与十进制中的10、11、12、13、14、15这6个数相对应。

十六进制数同样用位置记数法，但为了避免与十进制数相混淆，通常在十六进制数后面增加一后缀H。

十六进制数87.9H的按权展开多项式表示形式为：

$$87.9H=8\times(16)^1+7\times(16)^0+9\times(16)^{-1}$$

十六进制数的加法运算可按“逢十六进一”原则，减法运算可按“借一作十六”原则进行。这里没有必要列出各种情况下的加法、减法与乘法公式。在数据位数不太多的情况

下，可根据以上进位及借位原则，通过心算得出运算结果。

例 1.3　求以下十六进制算术运算的结果：

（1）95H＋36H；（2）95H－36H；（3）95H×36H；（4）95H÷36H。

解：（1）　95H　　（2）　95H　　（3）　95H

＋）36H　　－）36H　　×）36H

CBH　　5FH　　37E

1BF

1F6EH

（4）

$$\begin{array}{r} 2\text{H}\cdots\cdots\text{商} \\ 36\overline{)95\text{H}} \\ 6\text{C} \\ \hline 29\text{H}\ \cdots\cdots\text{余数} \end{array}$$

补充讨论：第（3）小题运算的中间过程如下：

5H×6H＝30＝1EH，6H×9H＝54＝36H，3H×9H＝27＝1BH

这一过程实际上是将十六进制数转换为十进制数相乘以后，再转换回来的过程。这一点读者也可以暂时放一下，等下面讨论完不同数制间的转换问题后再来领会。

由于逻辑运算是在二进制条件下按位进行的，因此十六进制数的逻辑运算只能按以下步骤进行：先将其转换成二进制，进行逻辑运算后，再转换回来。

以上所讨论的位置记数法及按权展开为多项式的方法可以推广到任意进制。例如，五进制数243.4的展开式为：

$$2\times5^2+4\times5^1+3\times5^0+4\times5^{-1}$$

1.1.4 不同数制间的转换

先看非十进制数到十进制数的转换。非十进制数到十进制数的转换比较简单，可先将其由位置记数形式按权展开为基数的幂多项式，再将系数及权均用十进制表示后按十进制进行乘法与加法运算，所求得的多项式的值即为该数对应的十进制数。

例 1.4　试将以下各非十进制数转换为十进制数：

（1）二进制数 $(1011.11)_2$；（2）十六进制数87.9H；（3）五进制数 $(243.4)_5$；（4）三进制数 $(201.21)_3$。

解：（1）$(1011.11)_2=1\times2^3+1\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}$

$=8+2+1+0.5+0.25=11.75$

（2）$87.9\text{H}=8\times16+7+9/16=135.5625$

（3）$(243.4)_5=2\times5^2+4\times5+3+4/5=73.8$

（4）$(201.21)_3=2\times3^2+1+2/3+1/9=19.7$

下面讨论十进制到非十进制的转换。这个问题同样可从幂多项式的表示式出发来进行讨论。设给定十进制数为N，对应的K进制数为N_K。若能将N_K展开成基数为K的幂多项式，则等式$D_{n-1}K^{n-1}+\cdots\cdots+D_1K+D_0+D_{-1}K^{-1}+\cdots\cdots+D_{-m}K^{-m}=N$成立。

由于等式右边N为给定常数，故通过待定系数法可求出等式左边多项式的各项系数。设N的整数部分为N_1，小数部分为N_2，则有

$$\begin{cases} D_{n-1}K^{n-1}+\cdots\cdots+D_1K+D_0=N_1 & (1)\\ D_{-1}K^{-1}+\cdots\cdots+D_{-m}K^{-m}=N_2 & (2)\end{cases}$$

由（1）式可看出，当将N_1除以K时其余数即等于等式左边的系数D_0。若接着将商再除以K，其余数即等于系数D_1。按此进行下去，直到商小于K为止，不难求出N_1展开式的全部系数。这种方法称为“连除取余法”。

由（2）式可看出，当用K去乘N_2其所得结果的整数部分即为系数D_{-1}。若接着将结果的小数部分再乘以K，并取整即可得系数D_{-2}。按此进行下去，不难求出N_2展开式的全部系数。这种方法称为“连乘取整法”。

例 1.5　试将十进制数871.625转换成为

（1）十六进制数；（2）四进制数；（3）二进制数。

解：（1）先将整除部分871用16除，得商数 54 余数 7，故$D_0=7$。再将54用16除，得商数 3 余数 6，故 $D_1=6$。因为商数 3 小于16，故 $D_2=3$，其它系数均为零。因此整数部分的展开式为：

$$871=3\times16^2+6\times16+7$$

再对小数部分进行处理。将0.625乘以16其所得结果为 10，小数部分为零。故 $D_{-1}=10$。

合并整数部分与小数部分可得871.625的展开式为：

$$871.625=3\times16^2+6\times16+7+10\times10^{-1}$$

改用位置记数法，将系数10用十六进制数码A表示，可得与871.625 相对应的十六进制数367.AH。

以上连除及连乘过程可分别用算式列出如下：

```
16 |871
 16 |54 ……余 7
  16 | 3 ……余 6
     0 ……余 3
```

```
        0.625
  ×)      16
-------------
整A……0.000
```

（2）将K改为 4，可直接列出连除及连乘过程如下：

```
4 |871
4 |217……余 3
 4 |54……余 1
 4 |13……余 2
  4 |3……余 1
    0……余 3
```

```
        0.625
  ×)       4
--------------
整 2 ……0.500
  ×)       4
--------------
整 2 ……0.000
```

由此可求得$D_0=3$，$D_1=1$，$D_2=2$，$D_3=1$，$D_4=3$，$D_{-1}=2$，$D_{-2}=2$

$\therefore$ $871.625=(31213.22)_4$

（3）将K改为 2，连除及连乘过程如下：

```
2 |871                    0.625
2 |435·····1          ×)      2
2 |217·····1          1·····0.250
2 |108·····1              ×)      2
 2 |54·····0          0·····0.500
 2 |27·····0              ×)      2
 2 |13·····1          1·····0.000
 2 | 6·····1
 2 | 3·····0
 2 | 1·····1
   0·····1
```

∴ $871.625=(1101100111.101)_2$

表1.1～1.3给出常用的十进制数与二进制数及十六进制数之间的转换关系可供参考。

表 1.1 0～16的十进制数与十六进制、二进制数对照

十进制数	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
十六进制数	0H	1H	2H	3H	4H	5H	6H	7H	8H	9H	AH	BH	CH	DH	EH	FH	10H
二进制数	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111	10000

表 1.2 20～1000的十进制数与十六进制数对照

十进制数	20	30	40	50	60	70	80	90	100	200	500	1000
十六进制数	14H	1EH	28H	32H	3CH	46H	50H	5AH	64H	C8H	1F4H	3E8H

表 1.3 10H～100H的十六进制数与十进制数对照

十六进制数	10H	20H	30H	40H	50H	60H	70H	80H	90H	A0H	B0H	C0H	D0H	E0H	F0H	100H
十进制数	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256

二个不同的非十进制之间的转换一般以通过十进制作为过渡较为方便。例如，五进制数243.4转换为二进制数，可先将其转换为十进制数73.8，再将73.8转换为二进制数。但K进制与K^m进制（m为整数）之间的转换则可采用比较直接的方法。以二进制数到十六进制数的转换为例，设二进制展开式的系数为D_i，十六进制展开式系数为C_i，则通过比较系数法可求得等式

$$\begin{aligned}&\cdots\cdots+(D_7\times2^7+D_6\times2^6+D_5\times2^5+D_4\times2^4)+(D_3\times2^3+D_2\times2^2\\&\qquad+D_1\times2^1+D_0\times2^0)+D_{-1}\times2^{-1}+D_{-2}\times2^{-2}+\cdots\cdots\\&=\cdots\cdots+C_2\times16^2+C_1\times16^1+C_0\times16^0+C_{-1}\times16^{-1}+\cdots\cdots\\&=\cdots\cdots+C_2\times2^8+C_1\times2^4+C_0\times2^0+C_{-1}\times2^{-4}+\cdots\cdots\end{aligned}$$

成立时，系数C_i与D_i的关系如下：

$$C_0=D_3\times2^3+D_2\times2^2+D_1\times2^1+D_0$$
$$C_1=D_7\times2^3+D_6\times2^2+D_5\times2^1+D_4$$

$$C_2 = D_{11} \times 2^3 + D_{10} \times 2^2 + D_9 \times 2^1 + D_8$$

……

$$C_{-1} = D_{-1} \times 2^{-3} + D_{-2} \times 2^{-2} + D_{-3} \times 2^{-1} + D_{-4}$$

……

由此可归纳出以下转换步骤：将二进制数以小数点为界向左每4位一组，找出对应的十六进制数（参见表1.1），向右也是如此（二端不足4位的补零）。

由二进制转换为四进制及八进制的原理及步骤与此相仿，只要将4位一组分别改为2位及3位一组即可。

例 1.6　试将二进制数1101100111.101转换成：（1）十六进制数；（2）八进制数；（3）四进制数。

解：（1）转换过程如下

$$\underbrace{0011}_{3}\ \underbrace{0110}_{6}\ \underbrace{0111}_{7}\cdot\underbrace{1010}_{A}$$ 二进制 / 十六进制

∴ $(1101100111.101)_2 = 367.AH$

（2）

$$\underbrace{001}_{1}\ \underbrace{101}_{5}\ \underbrace{100}_{4}\ \underbrace{111}_{7}\cdot\underbrace{101}_{5}$$ 二进制 / 八进制

∴ $(1101100111.101)_2 = (1547.5)_8$

（3）

$$\underbrace{11}_{3}\ \underbrace{01}_{1}\ \underbrace{10}_{2}\ \underbrace{01}_{1}\ \underbrace{11}_{3}\cdot\underbrace{10}_{2}\ \underbrace{10}_{2}$$ 二进制 / 四进制

∴ $(1101100111.101)_2 = (31213.22)_4$

反过来，由四进制、八进制及十六进制到二进制的转换只须将原数按位用2位、3位及4位二进制数码替换即可。有时十进制与二进制之间的转换也可通过十六进制间接地来实现，反而比直接转换方便。

例 1.7　试进行以下数制转换：

（1）将十六进制数367.AH转换为二进制数；

（2）将十六进制数367.AH转换为八进制数；

（3）将十进制数150.25转换成二进制数；

（4）将二进制数 $(111011.11)_2$ 转换成十进制数。

解：（1）

3　6　7 . A　十六进制

0011　0110　0111.1010　二进制

∴ $367.AH = (1101100111.101)_2$

（2）由（1）得 $367.AH = (1101100111.101)_2$

$$\underbrace{001}_{1}\ \underbrace{101}_{5}\ \underbrace{100}_{4}\ \underbrace{111.101}_{7.5}$$ 二进制 / 八进制

∴ $367.AH = (1547.5)_8$

（3）用间接法，先将150.25转换为十六进制数，再转换为二进制数。

16 | 150

16 | 9 …… 6

0 …… 9

0.25

×) 16

4 ……0.00

∴ 150.25＝96.4H＝10010110.01

（4）$(111011.11)_2$＝3B.CH＝3×16＋11＋12/16＝59.75

对（3）、（4）二小题读者可试与直接法相比较，看哪种方法简便。

1.2 BCD码及ASCII字符代码

1.2.1 BCD码及其表示与运算方法

前面讲过，在日常生活中人们最熟悉的数制是十进制，而计算机却只接受二进制数。为了解决这一矛盾，工程师们提出了一种折衷的方案，即将十进制的0～9这10个数码分别用4位二进制数码（0或1）的组合来代表，在此基础上可按位对任意十进制数进行编码，所得结果称为二进制编码的十进制，简称BCD码（全名为Binary-coded Decimal）。本来4位二进制数码可有16种不同的组合，原则上可任选其中的10种作为代码，但为了便于记忆和比较直观，通常采用表1.1所给出的0～9各十进制数所对应的二进制数（不够4位的前面补0）来构成BCD码。例如，十进制数32的BCD码为0011 0010。由此可知十进制数按BCD码制编码的过程要比十进制到二进制的转换简单得多。

BCD码在形式上与二进制数没有差别，但若将其看成真正的二进制数而通过展开式计算其十进制数值，所得结果将是错误的。这一点并不奇怪，因为它本身并不是根据十进制到二进制的转换关系通过连除取余法形成的，而是通过按位用二进制代码表示的方法形成的。因此按数制转换关系进行逆转换当然就得不到正确的结果。

同样，为了书写方便起见，在形式上我们还可以将BCD码用十六进制形式表示，只要将二进制代码4位一组组合成十六进制代码即可。为简单起见，我们仍把它称为BCD码。

例1.8 先将十进制数32转换为二进制数及十六进制数，再求与32对应的BCD码的二进制形式和十六进制形式，并与前面数制转换结果相比较。

解：32＝20H＝$(100000)_2$

32的BCD码的二进制形式为$(00110010)_2$，十六进制形式为32H。由此可知它与数制转换所得结果完全不同。

讨论：十进制数32的BCD码的十六进制形式与32的唯一差别是增加了一个后缀H。某些读者可能对这一点不放心，担心在原来的十进制数32后面加上后缀H就等于无故地把它变为十六进制数，这样可能要出错。殊不知二进制及十六进制码仅仅是一种形式，它可以代表真正的二进制数或十六进制数，也可以是某种字符或某种信息的代码。这里32H在形式上虽然与十六进制数一样，但它是十六进制代码而不是十六进制数。它与BCD码的二进制形式$(00110010)_2$一样均为BCD码，代表十进制数32的一种编码结果。对于$(00110010)_2$，由于在形式上与32有较大区别，故一般不会引起误解，但实质上它与32H是等效的。

下面我们来讨论BCD码的加法与减法运算。可能出现的问题是：由于交给计算机运算的BCD码在形式上与二进制完全相同，因此计算机将把它当作二进制数来运算，结果就可能出错。例如十进制数29与48相加的正确结果应当是77。交给计算机运算时29及48的BCD码分别为00101001及01001000，计算机按二进制相加结果为：

$$\begin{array}{r} 0010\ 1001 \\ +)0100\ 1000 \\ \hline 0111\ 0001 \end{array}$$

对应十进制数为71。与正确结果77相比较，显然它是错误的。

将原始数据及运算结果改用十六进制形式表示，可得

$$\begin{array}{r} 29\text{H} \\ +)48\text{H} \\ \hline 71\text{H} \end{array}$$

对应十进制数也是71。显然它也是错误的。

解决的办法是在错误的基础上进行修正。分析产生错误的原因是，十进制数相加的原则应当是“逢十进一”，而现在按二进制数运算，每4位一组，低4位向高4位进位的情况与十六进制数低位向高位进位的情况相当，其原则是“逢十六进一”，故当按BCD码位相加结果超过9时将比正确结果少6。因此，对加法运算可采用以下的“加6修正”法则：

（1）两BCD位相加结果超过9时对该位进行加6修正。

（2）两BCD位相加结果向高位有进位时对该位进行加6修正。

（3）低位修正结果使本位大于9时对本位进行加6修正。

例 1.9　试将以下算式中的十进制数按BCD码编码后，用加6修正法对加法运算结果进行修正，并检验其正确性。

（1）29+41；（2）29+48；（3）54+85；（4）99+90；（5）27+75。

解：为了书写方便起见，参加运算的原始数据及运算结果均采用BCD码的十六进制形式。

（1）修正前：

$$\begin{array}{r} 29\text{H} \\ +)41\text{H} \\ \hline 6\text{AH} \end{array}$$

根据法则（1）对低位进行加6修正后：

$$\begin{array}{r} 6\text{AH} \\ +)06\text{H} \\ \hline 70\text{H} \end{array}$$

BCD码对应十进制数为70，结果显然正确。

（2）修正前：

$$\begin{array}{r} 29\text{H} \\ +)48\text{H} \\ \hline 71\text{H} \end{array}$$

根据法则（2）对低位进行加6修正后：

$$\begin{array}{r} 71\text{H} \\ +)06\text{H} \\ \hline 77\text{H} \end{array}$$

BCD码对应十进制数为77，结果显然正确。

（3）修正前：

$$\begin{array}{r} 54\text{H} \\ +)85\text{H} \\ \hline \text{D9H} \end{array}$$

根据法则（1）对高位进行加6修正后：

$$\begin{array}{r} \text{D9H} \\ +)\ 60\text{H} \\ \hline \boxed{1}\ 39\text{H} \end{array}$$

考虑进位后，对应十进制数为139，结果显然正确。

（4）修正前：

$$\begin{array}{r} 99\text{H} \\ +)90\text{H} \\ \hline \boxed{1}\,29\text{H} \end{array}$$

根据法则（2）对高位进行加6修正后：

$$\begin{array}{r} 29\text{H} \\ +)60\text{H} \\ \hline \boxed{1}\,89\text{H} \end{array}$$

考虑进位后，对应十进制数为189，结果显然正确。

（5）修正前：　　根据法则（1）及（3）对低位及高位均进行修正后：

$$\begin{array}{r} 27H \\ +)75H \\ \hline 9CH \end{array} \qquad \begin{array}{r} 9CH \\ +)06H \\ \hline A2H \\ +)60H \\ \hline \boxed{1}02H \end{array}$$

考虑进位后，对应十进制数为102，结果显然正确。

两BCD码进行减法运算时，当低位向高位有借位时，由于“借一作十六”与“借一作十”的差别，将比正确结果多6，故有借位时可采用“减6修正法”来修正。

例 1.10　试将以下十进制数按BCD码编码后，用减6修正法对减法运算结果进行修正。

（1）38－12；（2）75－37。

解：（1）

$$\begin{array}{r} 38H \\ -)12H \\ \hline 26H \end{array}$$

低位向高位无借位，故不需修正。

（2）修正前：

$$\begin{array}{r} 75H \\ -)37H \\ \hline 3EH \end{array}$$

减6修正后：

$$\begin{array}{r} 3EH \\ -)06H \\ \hline 38H \end{array}$$

1.2.2 ASCII字符代码

在传送信息时，有时不仅要传送数字，还要传送英文字母、标点符号及一些常用运算符号。由于这些字符的总数超过16个，因此用4位二进制数码作为代码不能满足要求。通常需要采用7位或6位二进制代码，其对应所能表示的字符数分别为128及64。

在微型计算机系统中最常用的字符代码是采用7位二进制代码的ASCII代码，其次是6位ASCII代码。ASCII是美国信息交换标准代码的缩写，全名是American Standard Code for Information Interchange。

7位及6位ASCII码的编码见表1.4及表1.5。

7位ASCII代码经常在最高位前添加一个“0”组成8位代码。在传送ASCII码时，其最高位又常常被用作奇偶校验位，用来检验代码在传送过程中是否发生错误。偶校验时每个代码的二进制形式中应有偶数个“1”。例如传送字母“G”，其ASCII码的二进制形式为1000111，因其中有4个1，故奇偶校验位为0，8位代码将是01000111。奇校验时每个代码中应有奇数个“1”。若用奇校验传送字符“G”，则其二进制表示为11000111。

1.3 有符号二进制数的补码表示法与溢出问题

1.3.1 有符号二进制数的原码及补码表示法

到目前为止，我们一直假定参加二进制运算的原始数据为无符号数，即所有二进制数位均为数字位。然而在实际工作中原始数据可能是正数，也可能是负数。因此就产生了有符号二进制数的表示方法问题。一种表示方法是否合理，主要看采用这种方法时，有符号二进制数的算术运算（主要考虑加减运算）是否方便。

首先，为了完整地表示一个有符号数，除了数据位以外还应该指定适当的位作为符号位。习惯上以这个数的最高位为符号位。为简单起见，我们假定所讨论的数为整数。对于

8位有符号二进制整数，D_7位为符号位。而且一般规定$D_7=0$时代表正数，$D_7=1$时代表负数。这一点是无论哪一种表示法都应遵守的约定。故8位有符号数的基本格式如下：

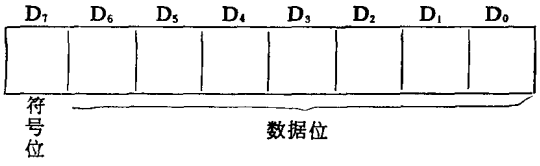

表 1.4 7位ASCII码编码

低 位	高 位							
	0	1	2	3	4	5	6	7
0			SP	0	@	P		p
1			!	1	A	Q	a	q
2			"	2	B	R	b	r
3			#	3	C	S	c	s
4			$	4	D	T	d	t
5			%	5	E	U	e	u
6			&	6	F	V	f	v
7	BEL		,	7	G	W	g	w
8			(	8	H	X	h	x
9			)	9	I	Y	i	y
A	LF		*	:	J	Z	j	z
B			+	;	K	[	k	{
C			,	<	L	/	l	|
D	CR		—	=	M	]	m	}
E			.	>	N	↑	n	~
F			/	?	O	←	o	DEL

表 1.5 6位ASCII码编码

低 位	高 位				低 位	高 位			
	0	1	2	3		0	1	2	3
0	SP	0	@	P	8	(	8	H	X
1	!	1	A	Q	9	)	9	I	Y
2	"	2	B	R	A	*	:	J	Z
3	#	3	C	S	B	+	;	K	[
4	$	4	D	T	C	,	<	L	/
5	%	5	E	U	D	—	=	M	]
6	&	6	F	V	E	.	>	N	↑
7	,	7	G	W	F	/	?	O	←

注 BEL—告警，LF—换行，CR—回车，SP—空格，DEL—删除。

在历史上曾采用过一种所谓“原码表示法”，即不管是正数还是负数，除符号位不同外，其数据位均用其绝对值来表示。按照“原码表示法”，与十进制数+3对应的有符号二进制数为$(00000011)_2$，而与十进制数−3对应的二进制数则为$(10000011)_2$。

这种有符号数表示方法看起来似乎也很合理，但用它来进行有符号数算术运算时就遇到了很大麻烦。假定我们要做的算术运算既有加法又有减法，参加运算的数既可能为正又

可能为负，这样我们将遇到正数加正数、负数加负数、正数加负数、两正数相减、两负数相减、正数减负数及负数减正数等不同情况。如果用人工来做，则可以借助人们的判断力来决定运算步骤。例如，正数加负数应先比较两数绝对值的大小，用较大的绝对值减去较小的绝对值，作为和的绝对值，然后取绝对值较大的数的符号作为和的符号。又如两正数相减，若减数大于被减数则应反过来减，然后取负号。若这一工作要计算机来做则比较麻烦。或要求计算机的结构比较复杂，或要求计算机花费很多时间来进行各种判断。这样做是很不经济的。因此目前普遍采用所谓“补码表示法”。采用“补码表示法”时，有符号数的算术运算就比较方便。

设X为一正数，我们将其用有符号8位二进制数的规定格式来表示，即令$D_7=0$，$D_6\sim D_0$为此正数的值，这一点与上面的“原码表示法”完全一致。现在再看若用零减去X，所得结果（取8位有效位）代表什么。仍设$X=(00000011)_2$则有

$$\begin{array}{r} 00000000 \\ -)\,00000011 \\ \hline 11111101 \end{array}$$

所得结果在形式上与采用原码表示法的-3有很大区别，但在本质上它却能更好地代表负数-3，因为$0-X=-X$，而$-X$即等于X的相反数，即对应的负数。另外，由于$D_7=1$，因此它也符合有符号数关于负数的符号位的基本约定。

由于数$(11111101)_2$补上$(00000011)_2$后得零，即对零来说$(11111101)_2$缺$(00000011)_2$，因此我们称$(11111101)_2$为$(00000011)_2$的补码，反过来$(00000011)_2$也是$(11111101)_2$的补码，即二者是互补的。

“补码表示法”，严格地说，应称为“负数的补码表示法”。用它来表示原始数据的法则可归纳如下：

（1）正数仍用与原码表示法相同的格式表示。

（2）负数用其对应正数的补码表示。

二进制补码也可用求反加1的方法来得到。所谓“求反”即按位将1变为0，将0变为1。这相当于用$(11111111)_2$来减原数。

由于$(11111111)_2-X+1=(11111111)_2+(00000001)_2-X=0-X$（取8位有效位）

故用求反加1法所得的结果与从基本定义出发所求得的结果是一致的。

例 1.11　设$X=(00110111)_2$。试先从基本定义出发，再用求反加1法分别求其补码。

解：（1）从基本定义出发

$$\begin{array}{r} 00000000 \\ -)\,00110111 \\ \hline \boxed{1}\,11001001 \end{array}$$

取8位有效位，得$X_{补}=(11001001)_2$。

（2）用求反加1法

$$X_{反}=11001000,\quad X_{补}=X_{反}+1=(11001001)_2。$$

例 1.12　求十进制负数-121的有符号二进制数补码表示法。