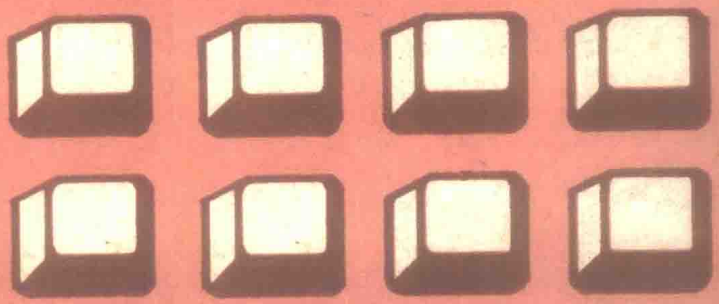
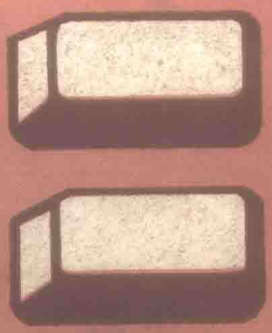
微型计算机技术

（高校工科非电专业适用）

黄义源　任仕纯　编





高等教育出版社

微型计算机技术

（高校工科非电类专业适用）

黄义源　任仕纯　编

高等教育出版社

这是一本国家教委高校工科电工学课程教学指导小组1986～1990年教材规划中计划编写的、以Z80汇编语言程序设计和接口技术为主要内容的简明教材，主要适用于高校工科机械类或其他各非电类专业，也可供各专业工程技术人员参考。

全书共分9章：数字电路基础、数字逻辑部件、微机的硬件结构、CPU的指令系统、汇编语言程序设计、接口技术、单板机及其应用、单片计算机、16位微处理器8086等。

本书有明显的特色：易学。讲授全书大约只需50学时，其中微机部分约32学时。书中文字流畅，重点突出，难点分散，思路清晰；从实用角度出发，删繁就简，联系实际；并将数字电路与微机相结合。本书数字电路内容符合国家教委批准的《电子技术(电工学II)课程教学基本要求》。

本书责任编辑　王辑忠

微型计算机技术

(高校工科非电类专业适用)

黄义源　任仕纯　编

*

高等教育出版社出版

新华书店北京发行所发行

商务印书馆上海印刷厂印装

开本 787×[illegible]92[illegible]1/16　印张 19[illegible]　字数 435[illegible]000

1988年1[illegible]1版　1988年10月第1次印刷

印数 0001—10,620

ISBN 7-04-001094-1/TP·31

定价 3.90元

前　言

随着微型计算机技术的飞速发展，微型计算机的应用已深入到各个领域，促进众多专业的发展。微型计算机现已成为工科各专业共同的基础知识。为了适应这种形势的需要，我们为本校非电专业学生编写了《微型计算机技术》教材，并从1984年开始试用，本书就是在这本教材多次试用的基础上修改而成的。

本书可作为高等学校工科非电专业的教材，也可作为广大工程技术人员学习和应用微型计算机的参考书。

考虑到非电专业的实际情况，全书力求突出两个特点：

1．便于学习。在内容的处理上，一是将数字电路与微型计算机有机的结合起来。数字电路是微型计算机的硬件基础；微型计算机是数字电路的典型应用。将两者紧密结合起来，不仅可以减少重复，使得内容精简、结构紧凑，而且由数字电路过渡到微型计算机，思路清晰，循序渐进，对某些内容的学习容易达到事半功倍的效果。二是在讲述CPU、PIO、CTC等典型芯片时，从应用的观点出发，内部结构仅作最简要的说明，力戒"技术说明书"式地介绍，尽量分散难点，便于非电类专业的学生理解，便于非电类专业的技术人员自学。

2．面向应用。非电专业学生学习微型计算机，主要是用它来研究和解决本专业的问题，推动本专业科学技术的发展。微型计算机的应用，主要是两个方面的问题，一是程序编制，二是接口技术。本书围绕这两方面组织微型计算机教学内容，并编入了大量的例题和习题。

本书数字电路部分是依据国家教委批准、1987年秋季始在全国试行的《高等工业学校电子技术(电工学II)课程教学基本要求》编写的，又按照微型计算机的需要，作了适当的扩充。

全书按70学时编写(含实验15～20学时)，其中数字电路部分约占25学时，微机部分占45学时。前面六章为基础部分，是本书的重点。七、八、九章可根据学时多少和需要选学。

本书一、二、三章由任仕纯执笔，其余各章由黄义源编写。清华大学梁毓厚副教授审阅了书稿，提出了许多宝贵的意见。在本书编写过程中，湖南大学杨贻馨教授给予了大量的帮助和指导，在此一并表示深切的感谢。

由于我们的学识水平有限，书中难免有错误和不妥之处，殷切期望读者批评指正。

编　者

1986年12月于湖南大学机械电子学教研室

目　录

第一章　数字逻辑基础

本章主要介绍数字逻辑电路的一些基本知识，如各种数制之间的相互转换，码制，基本逻辑关系以及相应的数学基础——逻辑代数，基本的逻辑器件——门电路和触发器等，为学习微型计算机打下基础。

1-1　计算机中数的表示方法

计算机的主要功能是对数进行处理，数的表示方法是首先要解决的问题。

一、计数制

10 进制是日常生活中最常用的计数制，它用 0～9 共 10 个数字符号来表示数的大小。这些数字符号也叫数码。数制所用数码的个数称为基数。例如 10 进制有 10 个数码，基数为 10，计数时逢 10 进 1。除 10 进制外，常用到的数制还有 2 进制、8 进制、12 进制、16 进制等。基数小于 10 的计数制，用 10 进制的相应字符作为它的字符，如 2 进制的字符为 0～1，8 进制的字符为 0～7。基数大于 10 的计数制，采用 10 进制字符加上大写的英文字母(从首字符开始)，作为它的字符。如 16 进制的字符，除用 0～9 的十个数字外，还用 A～F 的六个字母。表 1-1 列出了这几种数制的基数和字符。

表 1-1　计数制

数　制	基　数	字　符
2 进制	2	0　1
8 进制	8	0　1　2　3　4　5　6　7
10 进制	10	0　1　2　3　4　5　6　7　8　9
12 进制	12	0　1　2　3　4　5　6　7　8　9　A　B
16 进制	16	0　1　2　3　4　5　6　7　8　9　A　B　C　D　E　F

一个数常常是由多个数码组合成的。数码在数中的位置不同，其值也不同。例如 10 进制数 123.4，其中数码 1 位于百位，其值为一百；2 位于十位，其值为二十；3 位于个位，其值为三；4 在小数点后一位，即十分之一位，其值为十分之四。123.4 这个数可以展开成下面的多项式：

$$123.4=1\times10^{2}+2\times10^{1}+3\times10^{0}+4\times10^{-1}$$

式中：10^{2}、10^{1}、10^{0}、10^{-1} 等称为该位的“权”，每一位上的数码与该位“权”的乘积，就是该位数值的大小。上式称为按权展开式。

其它进制的数，也可按权展开，其一般形式为：

$$N=d_{n-1}b^{n-1}+d_{n-2}b^{n-2}+\cdots+d_{-m}b^{-m}=\sum_{i=n-1}^{-m}d_{i}b^{i} \tag{1-1}$$

式中：d_i——第 i 位的数码；

b——基数；

b^i——权；

n——整数的总位数；

m——小数的总位数。

例如：2 进制数 $11011=1\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0$

8 进制数 $256=2\times8^2+5\times8^1+6\times8^0$

16 进制数 $3AF4=3\times16^3+A\times16^2+F\times16^1+4\times16^0$

在计算机中，主要采用 2 进制。因此，下面将重点介绍 2 进制。

二、2 进制

1．计算机中为什么采用 2 进制

(1) 2 进制只有 **0, 1** 两种状态，很容易用电子元件来实现。例如可设开关的闭合为 **1**，断开为 **0**；电压的高为 **1**，低为 **0**；脉冲的有为 **1**，无为 **0** 等等。而且这种简单的状态工作可靠，抗干扰能力强。

(2) 2 进制数运算简单

2 进制数只有两个数码，运算规则较 10 进制简单得多。

加法：进位规则是逢 2 进 1，即 **1+1=10**

```
  1101……被加数
+1011……加数
------
 10110……和
```

减法：借位规则是“借 1 当 2”。

```
  1.0.01……被减数
-0 1 10……减数
---------
  0 0 11……差
```

乘法：规则是：**0×0=0**, **1×0=0**, **1×1=1**

```
   1011…被乘数
×  101…乘数
------
   1011
  0000
 1011
------
 110111……积
```

除法：除法是乘法的逆运算。

```
          000111……商
除数…101 ) 100011……被除数
           101
           ---
            111
            101
            ---
              101
              101
              ---
                0
```

由上可见，在加、减、乘、除四则运算中，乘法实质上是做移位加法，除法则是做移位减法（本书第五章中将详细讨论）。后面将要讲到，减法也可改为做加法。因此，在计算机中只要用一种加法器，就可完成四则运算，从而使运算电路大为简化。

2．2 进制到 10 进制的转换

如前所述，计算机中采用 2 进制，而人们习惯的是 10 进制，因此常需进行 2 进制和 10 进制的相互转换。

2 进制转换成 10 进制的基本方法，是对 2 进制的按权展开式求和，得到等值的 10 进制数。

[例 1-1] 将 2 进制数 1101、1011011、0.111、11.101 转换成等值的 10 进制数。

解：(1) $1101_2=1\times2^3+1\times2^2+0\times2^1+1\times2^0$

$=8+4+1$

$=13_{10}$

(2) $1011011_2=1\times2^6+1\times2^4+1\times2^3+1\times2^1+1$

$=64+16+8+2+1$

$=91_{10}$

(3) $0.111_2=2^{-1}+2^{-2}+2^{-3}$

$=\frac{1}{2}+\frac{1}{4}+\frac{1}{8}$

$=0.875_{10}$

(4) $11.101_2=2^1+2^0+2^{-1}+2^{-3}$

$=2+1+0.5+0.125$

$=3.625_{10}$

上述方法，对 N 进制都是适用的。

[例 1-2] 将 8 进制数 156 和 16 进制数 64 转换成等值的 10 进制数。

解：(1) $156_8=1\times8^2+5\times8^1+6\times8^0$

$=64+40+6$

$=110_{10}$

(2) $64_{16}=6\times16^1+4\times16^0$

$=96+4$

$=100_{10}$

3．10 进制到 2 进制的转换

基本方法是除 2 取余法：将 10 进制数不断用 2 除，所得余数从后往前读，得到等值的 2 进制数。

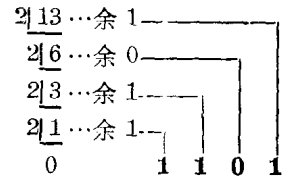
[例 1-3] 将 10 进制数 5、13、16 转换成等值的 2 进制数。

解：(1)

2|5 …余 1
2|2 …余 0
2|1 …余 1
0

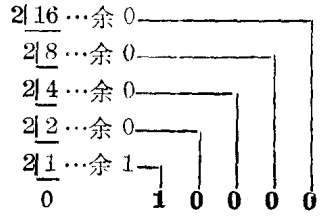
即 $5_{10}=101_2$

(2)



即 $13_{10}=\mathbf{1101}_2$

(3)



即 $16_{10}=\mathbf{10000}_2$

上述方法，如将基数 2 改为 N，则适用于将 10 进制数转换成等值的 N 进制数。

将 10 进制小数化为等值的 2 进制小数，稍微麻烦一点。通常采用乘 2 取整法，即将 10 进制小数，不断用 2 乘，每次将整数取出，即得相应的 2 进制小数。

[例 1-4] 将 10 进制小数 0.875、0.325 转换成等值的 2 进制小数。

解：(1)

0.875	0.750	0.5
× 2	× 2	× 2
1.750	1.500	1.0
↓	↓	↓
1	1	1

即 $0.875_{10}=\mathbf{0.111}_2$

(2)

0.325	0.65	0.3	0.6
× 2	× 2	× 2	× 2
0.650	1.30	0.6	1.2
↓	↓	↓	↓
0	1	0	1

即 $0.325_{10}\approx\mathbf{0.0101}_2$

三、16 进制

2 进制的一个缺点是位数太多，不易识别，写起来太麻烦。为了识别方便，在书写计算机

表 1-2 各种进制的对照表

10 进制数	2 进制数	16 进制数
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	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
16	10000	10
17	10001	11
18	10010	12
19	10011	13
20	10100	14

的程序时，常将它们写成16进制数。表1-2是2进制、16进制的对照表。

1．16进制与2进制的转换

由表1-2可见，每位16进制数可由4位2进制数表示。因此将16进制数的各位用相应的4位2进制表示，即可转换成等值的2进制数。

[例1-5] 将16进制数13AF、5C06转换成等值的2进制数。

解：(1) $13AF_{16}=\underline{0001}\ \underline{0011}\ \underline{1010}\ \underline{1111}$
$=1001110101111_2$

(2) $5C06_{16}=\underline{0101}\ \underline{1100}\ \underline{0000}\ \underline{0110}$
$=101110000000110_2$

(有效数字前的0舍掉)

2进制转换为16进制时，将2进制数从右至左每4位一组，每组用等值的16进制表示。

[例1-6] 将2进制数10110101、1001101转换成等值的16进制数。

解：(1) $10110101_2=\underline{1011}\ \underline{0101}$
$=B5_{16}$

(2) $1001101_2=\underline{0100}\ \underline{1101}$
$=4D_{16}$

2．书写时的规定

在书写计算机程序时，一般不是用基数作下标来区别各种进制，而是用相应的英文字母作后缀来标识各种进制的数。

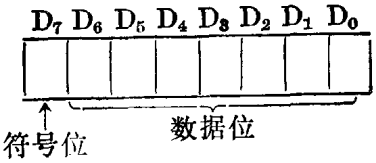
例如：B(Binary)——表示2进制数，可将100_2写成100B；

D(Decimal)——表示10进制数，一般约定D可省略，即无后缀的数字为10进制数字100D=100；

H(Hexadecimal)——表示16进制数，如100_{16}=100H。

四、有符号数的表示方法

前面提到的2进制数，没有涉及数的正负问题。不考虑正负的数称为无符号数。工业过程控制中，表示各种开关状态的数据(如用**1**表示闭合，用**0**表示断开)，属无符号数。算术运算中的数，自然会有正有负，这类数称为有符号数。为了在计算机中正确地表示有符号数，通常规定，数字的最高位为符号位，并用**0**表示正，用**1**表示负。在8位微型计算机中，一个数用8位2进制数表示，其标准格式如下：

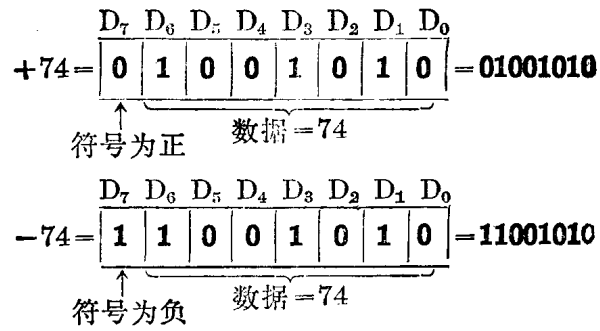


最高位D_7为符号位，D_6～D_0为2进制数据位。

例如：+74=+**1001010**

−74=−**1001010**

在机器中表示为：



这种连正负号也数字化的数，称为机器数，是计算机所能识别的数。

机器数有原码、反码、补码三种表示方法，兹分述如下：

1．原码

在符号位中用 **0** 表示正、用 **1** 表示负的 2 进制数，称为原码。

[例 1-7] 设 $x_1=+0010101$，$x_2=-0010101$，试写出 x_1 和 x_2 的原码。

解：(1) $[x_1]_{原}=0\ 0010101$

0 ——用 0 表示正；0010101 ——数值不变

(2) $[x_2]_{原}=1\ 0010101$

1 ——用 1 表示负；0010101 ——数值不变

[例 1-8] 写出 +13、−13、+127、−127、+0、−0 的原码(用 8 位 2 进制数表示，下同)。

解：$[+13]_{原}=0\ 0001101$

$[-13]_{原}=1\ 0001101$

$[+127]_{原}=0\ 1111111$

$[-127]_{原}=1\ 1111111$

$[+0]_{原}=0\ 0000000$

$[-0]_{原}=1\ 0000000$

由上面两个例子可以看出，用原码表示带符号数，其数值部分不变，易于识别，这是方便的一面；但在进行加减法运算时，比较复杂。例如，在进行加法运算时，机器首先要判断两数的符号是否相同；若相同，则做加法；若不同，则做减法。做减法时还要判断两数绝对值的大小，以便于大者减去小者。然后再根据绝对值大的数决定差值的符号。

引入反码和补码的概念，就可解决这个困难，使运算简化。

2．反码

反码的定义是：

正数：反码＝原码

负数：反码＝原码的符号位不变而数值按位取反

所谓按位取反，就是将各位的 **1** 变成 **0**，**0** 变成 **1**。

[例 1-9] 求 +13、−13 的反码。

解：$[+13]_{反}=[+13]_{原}$

$=0\ 0001101B$

$[-13]_{原}=1\ 0001101B$

$[-13]_{反}=1\ 1110010B$

数值按位取反

符号位不变

即：$[-13]_{反}=11110010B$

[例 1-10] 求$[+0]_{反}$、$[-0]_{反}$、$[[-0]_{反}]_{反}$。

解：$[+0]_{反}=[+0]_{原}$

$=0\ \ 0000000B$

$[-0]_{原}=1\ \ 0000000$

$[-0]_{反}=1\ \ 1111111$ ← 符号位不变，数值按位取反

$[[-0]_{反}]_{反}=1\ \ 0000000$ ← 符号位不变，数值按位取反

可见，$[[-0]_{反}]_{反}=[-0]_{原}$，也就是说，0 的反码的反码等于原码，这个结论可以推广为：

$$[[x]_{反}]_{反}=[x]_{原} \tag{1-2}$$

其正确性，读者可自行验证。

3. 补码

补码的定义是：

正数：补码＝原码

负数：补码＝反码＋1

[例 1-11] 求 +13、−13、+127、−127 的补码。

解：一般采用先由原码求反码，再求补码的方法。

(1) $[+13]_{补}=[+13]_{原}$

$=0\ 0001101$

(2) $[-13]_{原}=1\ 0001101$

$[-13]_{反}=1\ 1110010$ ← 符号位不变，数值按位取反

$[-13]_{补}=1\ 1110011$ ← 加 1

(3) $[+127]_{补}=[+127]_{原}$

$=0\ 1111111$

(4) $[-127]_{原}=1\ 1111111$

$[-127]_{反}=1\ 0000000$ ← 符号位不变，数值按位取反

$[-127]_{补}=1\ 0000001$ ← 加 1

[例 1-12] 求$[-3]_{补}$、$[[-3]_{补}]_{补}$。

解：(1) $[-3]_{原}=10000011$ ①

$[-3]_{反}=11111100$

$[-3]_{补}=11111101$

(2) 设 $x=[-3]_{补}=11111101$，则

$[x]_{原}=11111101$

$[x]_{反}=10000010$

$[x]_{补}=10000011$

即 $[x]_{补}=[[-3]_{补}]_{补}=10000011$ ②

比较 ①② 两式可知，$[[-3]_{补}]_{补}=[-3]_{原}$。即其补码的补码等于原码，推广之，有

$$[[x]_{补}]_{补}=[x]_{原} \tag{1-3}$$

结论的正确性，读者可自行验证。

上面简要地介绍了原码、反码和补码。目的在于解决带符号数的表示方法和计算问题。它们的特点是：

(1) 正负符号数字化。规定数字的最高位为符号位，用 2 进制数 **0** 表示正、**1** 表示负。

(2) 数值为正时，三种机器数相同，即

$$[+x]_{原}=[+x]_{反}=[+x]_{补}=x$$

(3) 数值为负时，三种机器数的符号位均为 **1**，但后面的数值部分各不相同：

原码的数值部分与原数值相同；

反码的数值部分，为原数值按位取反；

补码的数值部分，为原数值按位取反后加 **1**。

表 1-3 列出了 8 位机器数与无符号数、有符号数的原码、反码、补码的对照表。

表 1-3　8 位机器数对照表

8 位机器数		无符号数	有　符　号　数		
2 进制数	16 进制数		原　码	反　码	补　码
00000000	00	0	+0	+0	+0
00000001	01	1	+1	+1	+1
00000010	02	2	+2	+2	+2
⋮	⋮	⋮	⋮	⋮	⋮
01111100	7C	124	+124	+124	+124
01111101	7D	125	+125	+125	+125
01111110	7E	126	+126	+126	+126
01111111	7F	127	+127	+127	+127
10000000	80	128	−0	−127	−128
10000001	81	129	−1	−126	−127
10000010	82	130	−2	−125	−126
⋮	⋮	⋮	⋮	⋮	⋮
11111100	FC	252	−124	−3	−4
11111101	FD	253	−125	−2	−3
11111110	FE	254	−126	−1	−2
11111111	FF	255	−127	−0	−1

在计算机中，负数一律用补码表示。

利用补码，就可将减法运算变为加法运算，兹举例说明如下：

[例 1-13]　35−24=11

解：用减法：

$$\begin{array}{r} 00100011 \leftarrow 35 \\ -\ 00011000 \leftarrow 24 \\ \hline 00001011 \leftarrow 11 \end{array}$$

用加法：

$$\begin{array}{r} 00100011 \leftarrow 35 \\ +\ 11101000 \leftarrow [-24]_{补} \\ \hline 100001011 \leftarrow 11 \end{array}$$

↑ 自然丢失

在8位机中，最高位的进位，已超出机器字长的范围，所以自然丢失。减法运算与补码的加法运算，结果相同。

[例1-14] 18−14=4

解：

$$\begin{array}{l} 00010010 \leftarrow 18 \\ +\,11110010 \leftarrow [-14]_{补} \\ \hline 100000100 \leftarrow 4 \end{array}$$

↑ 自然丢失

[例1-15] (−5)+(−7)=−12

解：

$$\begin{array}{l} 11111011 \leftarrow [-5]_{补} \\ +\,11111001 \leftarrow [-7]_{补} \\ \hline 111110100 \leftarrow [-12]_{补} \end{array}$$

↑ 丢失

在和中，符号位为1，故结果是负数，将结果的数值部分按位取反加1，求得原码为10001100，即为−12。

[例1-16] 13−14=−1

解：

$$\begin{array}{l} 00001101 \leftarrow 13 \\ +\,11110010 \leftarrow [-14]_{补} \\ \hline 11111111 \leftarrow [-1]_{补} \end{array}$$

可见，引入补码的概念以后，加减法运算都可用加法来实现；数值的符号位也当作数值处理，一道参加运算，非常方便。因此，在计算机中，加减法一般都采用补码运算。

五、2-10进制编码

如前所述，计算机中采用2进制，而人们却习惯用10进制。为了便于人机联系，通常还采用一种2-10进制的编码方式。

2-10进制的本质是10进制，具有10进制的特点；但每位10进制数用相应的4位2进制码表示，又具有2进制的形式。实际上，2-10进制是一种用2进制编码的10进制数(Binary Coded Decimal)，简称BCD码。

2-10进制编码的方法很多，最常用的是8421码。

4位2进制数能表示0000～1111共16种状态。按自然顺序舍去了后面六种状态(1010～1111)，只取前面10种状态(0000～1001)，用来表示0～9的10进制数，自左至右每位2进制数字的权分别为8、4、2、1，这种码制称为8421码，如表1-4所示。8421码中，被舍去的6种状态称为非法码。

表1-4 2-10进制编码

10进制数	8421码(BCD码)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

由表 1-4 可以看出，对于一位 10 进制数，其结果与 2 进制完全相同。但如果是两位以上的 10 进制数，就不相同了。例如 10 进制数的 12，其 BCD 码为 8 位：**00010010**，而不是 2 进制的 **1100**。

BCD 码计数法，比较直观，易于识别。例如 5678 的 BCD 码为：

0101	**0110**	**0111**	**1000**
5	6	7	8

BCD 码既满足了计算机必须使用 2 进制的要求，又保持了 10 进制的方便形式，因此得到了广泛的应用。

在用 BCD 码进行计算时，进位规则与 4 位 2 进制的进位规则不同。4 位 2 进制是逢 16 进 1，8421 码是 10 进制，逢 10 进 1。两数相加结果大于 9 时，就出现非法码。例如：

34+9=43

十进制	8421 码
34	**0011 0100**
+ 9	+ **1001**
43	**0011 1101**
	十位 个位

可见，个位数相加结果大于 9，出现了非法码，整个结果都错了，因此要进行修正。4+9=13，用 8421 码表示为 00010011，原计算结果中的"1101"比"00010011"要小 6。因此为了得到正确的结果，应进行加 6 修正。即

0011 1101←原结果
\+ **0110**←加 6 修正
0100 0011=43←最终结果

[例 1-17] 用 8421 码求

48+69=117

解：

0100 1000
+**0110 1001**
1011 0001
+**0110 0110**←加 66 修正
10001 0111=117

此题中，个位数为 **10001**，有进位，须加 **6** 修正，而十位数出现了非法码 **1011**，也要作加 6 修正。

对减法运算则用减 6 修正，规律与加法相同。

[例 1-18] 用 8421 码求 34-9=25

0011 0100←34
−**0000 1001**←9
0010 1011≠25←个位是非法码
− **0110** ←减 6 修正
0010 0101=25

在计算机中，一般都有专门的指令来完成 BCD 码运算中的加 6、减 6 修正功能（称为 10 进制调整）。

六、字符代码

在计算机中，除数字外，还需处理各种字符，如字母、运算符号、标点符号等等。这些字符