第 2 章　计算机中的数据表示

本章介绍计算机中数据的表示方法，重点讲述计算机中常用的数制及其转换、带符号数的表示、字符编码和汉字编码的基本知识。通过本章的学习，读者应掌握以下内容：

- 计算机中数制的基本概念、数制之间的相互转换
- 无符号数和带符号数的表示方法
- ASCII 码和 BCD 码的相关概念和应用
- 汉字编码及其应用

2.1　计算机中的数制及其转换

在计算机内，不论是指令还是数据，都采用了二进制编码形式，包括图形和声音等信息，也必须转换成二进制数的形式，才能存入计算机中。由于计算机的处理对象是各种各样的数据，在使用上，我们把计算机中的数据分为两类：

（1）数：用来直接表示量的多少，它们有大小之分，能够进行加减等运算。

（2）码：通常指代码或编码，在计算机中用来代表某个事物或描述某种信息。

在计算机中，数和码仅仅在使用场合上有区别，用于表示不同性质的数据，而在使用形态上并没有区别。

2.1.1　数制的基本概念

1. 数的表示

人们在日常生活中最熟悉、最常用的数是十进制数，它采用 0～9 共 10 个数字符号及其进位来表示数的大小。其相关概念如下：

- 0～9 这些数字符号称为“数码”。
- 全部数码的个数称为“基数”，十进制数的基数为 10。
- 用“逢基数进位”的原则进行计数，称为进位计数制。十进制数的基数是 10，所以其计数原则是“逢十进一”。
- 进位以后的数字，按其所在位置的前后将代表不同的数值，表示各位有不同的“位权”。

例如：十进制数个位的“1”，代表 1，即个位的位权是 1。

十进制数十位的“1”，代表 10，即十位的位权是 10。

十进制数百位的“1”，代表 100，即百位的位权是 100，依此类推。

- 位权与基数的关系：位权的值等于基数的若干次幂。

例如：十进制数 2518.234，可以展开为下面的多项式：

$2518.234=2\times10^3+5\times10^2+1\times10^1+8\times10^0+2\times10^{-1}+3\times10^{-2}+4\times10^{-3}$

式中：10^3、10^2、10^1、10^0、10^{-1}、10^{-2}、10^{-3} 即为该位的位权，每一位上的数码与该位权的乘积就是该位的数值。

- 任何一种数制表示的数都可以写成按位权展开的多项式之和，其一般形式为：

$$N=d_{n-1}b^{n-1}+d_{n-2}b^{n-2}+d_{n-3}b^{n-3}+\ldots d_{-m}b^{-m}=\sum_{i=-m}^{n-1}d_i\times b^i$$

式中：

n——整数的总位数。

m——小数的总位数。

d_i——表示该位的数码。

b——表示进位制的基数。

b^i——表示该位的位权。

2. 计算机中常用的进位计数制

计算机内部的电子部件有两种工作状态，即电流的“通”与“断”（或电压的“高”与“低”），因此计算机能够直接识别的只是二进制数，这就使得它所处理的数字、字符、图像、声音等信息都是以 1 和 0 组成的二进制数的某种编码。

由于二进制在表达一个数字时，位数太长，不易识别，且容易出错，因此在书写计算机程序时，经常将它们写成对应的十六进制数或八进制数，也采用人们熟悉的十进制数表示。

在计算机内部可以根据实际情况的需要分别采用二进制数、八进制数、十进制数和十六进制数。

表 2-1 给出了计算机中常用计数制的基数、数码以及进位关系。

表 2-1　计算机中常用计数制的基数、数码以及进位关系

计数制	基数	数码	进位关系
二进制	2	0、1	逢二进一
八进制	8	0、1、2、3、4、5、6、7	逢八进一
十进制	10	0、1、2、3、4、5、6、7、8、9	逢十进一
十六进制	16	0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F	逢十六进一

表 2-2 给出了常用计数制的表示方法。

表 2-2 计算机中常用计数制的表示方法

十进制	二进制	八进制	十六进制	十进制	二进制	八进制	十六进制
0	0000	0	0	10	1010	12	A
1	0001	1	1	11	1011	13	B
2	0010	2	2	12	1100	14	C
3	0011	3	3	13	1101	15	D
4	0100	4	4	14	1110	16	E
5	0101	5	5	15	1111	17	F
6	0110	6	6	16	10000	20	10
7	0111	7	7	17	10001	21	12
8	1000	10	8	18	10010	22	13
9	1001	11	9	19	10011	23	14

3. 计数制的书写规则

为了区分各种计数制的数据，经常采用如下方法进行书写表达。

（1）在数字后面加写相应的英文字母作为标识。

B（Binary）：表示二进制数，二进制数的 100 可以写成 100B。

O（Octonary）：表示八进制数，八进制数的 100 可以写成 100O。

D（Decimal）：表示十进制数，十进制数的 100 可以写成 100D，通常其后缀可以省略。

H（Hexadecimal）：表示十六进制数，十六进制数的 100 可以写成 100H。

（2）在括号外面加数字下标。

$(1011)_2$：表示二进制数的 1011。

$(3157)_8$：表示八进制数的 3157。

$(2468)_{10}$：表示十进制数的 2468。

$(2DF2)_{16}$：表示十六进制数的 2DF2。

2.1.2 数制之间的转换

1. 十进制数转换为二进制数

一个十进制数通常由整数部分和小数部分组成，这两部分的转换规则是不同的，在实际应用中，整数部分与小数部分要分别进行转换。

（1）十进制整数转换为二进制整数。

十进制整数转换为二进制整数的方法是：采用基数 2 连续去除该十进制整数，直至商等于“0”为止，然后逆序排列余数，即可得到与该十进制整数相应的二进制整数各位的系数。

【例 2-1】将十进制整数$(105)_{10}$转换为二进制整数，采用“除 2 倒取余”的方法，过程如下：

```
2 | 105
  2 | 52        余数为1
    2 | 26      余数为0
      2 | 13    余数为0
        2 | 6   余数为1
          2 | 3 余数为0
            2 | 1 余数为1
              0 余数为1
```

所以，$(105)_{10}=(1101001)_2$

（2）十进制小数转化为二进制小数。

十进制小数转换为二进制小数的方法是：连续用基数 2 去乘以该十进制小数，直至乘积的小数部分等于“0”，然后顺序排列每次乘积的整数部分，即可得到与该十进制小数相应的二进制小数各位的系数。

【例 2-2】将十进制小数$(0.8125)_{10}$转换为二进制小数，采用“乘 2 顺取整”的方法，过程如下：

```
0.8125×2=1.625    取整数位 1
0.625×2=1.25      取整数位 1
0.25×2=0.5        取整数位 0
0.5×2=1.0         取整数位 1
```

所以，$(0.8125)_{10}=(0.1101)_2$

如果出现乘积的小数部分一直不为“0”，则根据精度的要求截取一定的位数即可。

2. 十进制数转换为八进制数和十六进制数

同理，十进制数转换为八进制数或十六进制数时，可以参照十进制数转换为二进制数的对应方法来处理。

（1）十进制整数转换为八进制整数或十六进制整数。

十进制整数转换为八进制整数或十六进制整数的方法是：采用基数 8 或基数 16 连续去除该十进制整数，直至商等于“0”为止，然后逆序排列所得到的余数，即可得到与该十进制整数相应的八进制整数或十六进制整数各位的系数。

【例 2-3】将十进制整数$(1687)_{10}$转换为八进制整数，采用“除 8 倒取余”的方法，过程如下：

```
8 | 1687
  8 | 210     余数为7
    8 | 26    余数为2
      8 | 3   余数为2
        0     余数为3
```

所以，$(1687)_{10}=(3227)_8$

将十进制整数$(2347)_{10}$转换为十六进制整数，采用“除 16 倒取余”的方法，过程如下：

16 | 2347
16 | 146　　余数为11（十六进制数为B）
16 | 9　　余数为2
0　　余数为9

所以，$(2347)_{10}=(92B)_{16}$

（2）十进制小数转换为八进制小数或十六进制小数。

十进制小数转换为八进制小数或十六进制小数的方法是：连续用基数 8 或基数 16 去乘以该十进制小数，直至乘积的小数部分等于“0”，然后顺序排列每次乘积的整数部分，即可得到与该十进制小数相应的八进制小数或十六进制小数各位的系数。

【例 2-4】将十进制小数$(0.9525)_{10}$转换为八进制小数，采用“乘 8 顺取整”的方法，过程如下：

0.9525×8=7.62　　取整数位 7
0.62×8=4.96　　取整数位 4
0.96×8=7.68　　取整数位 7
0.68×8=5.44　　取整数位 5

如果数据的计算精度取小数点后 4 位数，则其后的数可以不再计算。

所以，$(0.9525)_{10}=(0.7475)_8$

【例 2-5】将十进制小数$(0.5432)_{10}$转换为十六进制小数，采用“乘 16 顺取整”的方法，过程如下：

0.5432×16=8.6912　　取整数位 8
0.6912×16=11.0592　　取整数位 11（十六进制数为 B）
0.0592×16=0.9472　　取整数位 0
0.9472×16=15.1552　　取整数位 15（十六进制数为 F）

取数据的计算精度为小数点后 4 位数。

所以，$(0.5432)_{10}=(0.8B0F)_{16}$

3. 二进制数、八进制数、十六进制数转换为十进制数

二进制数、八进制数、十六进制数转换为十进制数时，按照“位权展开求和”的方法即可得到。

（1）二进制数转换为十进制数。

二进制数转换为十进制数时，用其各位所对应的系数 1（系数为 0 时可以不必计算）来乘以基数为 2 的相应位权，即可得到与二进制数相应的十进制数。

【例 2-6】将二进制数$(1011001.101)_2$转换为十进制数，过程如下：

$$(1011001.101)_2=1\times2^6+1\times2^4+1\times2^3+1\times2^0+1\times2^{-1}+1\times2^{-3}$$
$$=64+16+8+1+0.5+0.125$$
$$=(89.625)_{10}$$

（2）八进制数转换为十进制数。

八进制数转换为十进制数时，用其各位所对应的系数来乘以基数为 8 的相应位权，即可得到与八进制数相应的十进制数。

【例 2-7】将八进制数$(1476.52)_8$转换为十进制数，过程如下：

$(1476.52)_8 = 1\times8^3+4\times8^2+7\times8^1+6\times8^0+5\times8^{-1}+2\times8^{-2}$

$=512+256+56+6+0.625+0.03125$

$=(830.65625)_{10}$

（3）十六进制数转换为十进制数。

十六进制数转换为十进制数时，用其各位所对应的系数来乘以基数为 16 的相应位权，即可得到与十六进制数相应的十进制数。

【例 2-8】将十六进制数$(2D7.A)_{16}$转换为十进制数，过程如下：

$(2D7.A)_{16} = 2\times16^2+13\times16^1+7\times16^0+10\times16^{-1}$

$=512+208+7+0.625$

$=(727.625)_{10}$

4. 二进制数与八进制数和十六进制数之间的转换

（1）二进制数与八进制数之间的转换。

因为 $8=2^3$，所以一位八进制数相当于三位二进制数，这样八进制数与二进制数之间的转换就很方便了。由八进制数转换成二进制数时，只要将每位八进制数用三位二进制数表示即可；而由二进制数转换成八进制数时，先要从小数点开始分别向左或向右将每三位二进制数分成一组，不足三位数的要补 0，然后将每三位二进制数用一位八进制数表示即可。

- 八进制数转换为二进制数的方法是“一分为三”。

八进制数：	0	1	2	3	4	5	6	7
二进制数：	000	001	010	011	100	101	110	111

【例 2-9】将八进制数$(3257.461)_8$转换为二进制数，过程如下：

3	2	5	7.	4	6	1
↓	↓	↓	↓	↓	↓	↓
011	010	101	111.	100	110	001

所以，$(3257.461)_8=(11010101111.100110001)_2$

- 二进制数转换为八进制数的方法是“三合一”。

整数部分：自右向左三位一组，不够位时补 0，每组对应一个八进制数码。

小数部分：自左向右三位一组，不够位时补 0，每组对应一个八进制数码。

【例 2-10】将二进制数$(11010010110.10101101)_2$转换为八进制数，过程如下：

011	010	010	110.	101	011	010
↓	↓	↓	↓	↓	↓	↓
3	2	2	6.	5	3	2

所以，$(11010010110.10101101)_2=(3226.532)_8$

（2）二进制数与十六进制数之间的转换。

因为 $16=2^4$，所以一位十六进制数相当于四位二进制数。从十六进制数转换为二进制数时，只要将每位十六进制数用四位二进制数表示即可；而从二进制数转换为十六进制数时，先要从小数点开始分别向左或向右将每四位二进制数分成一组，不足四位的要补 0，然后将每四位二进制数用一位十六进制数表示即可。

- 十六进制数转换为二进制数的方法是“一分为四”。

十六进制数：　0　1　2　3　4　5　6　7

二进制数：0000　0001　0010　0011　0100　0101　0110　0111

十六进制数：　8　9　A　B　C　D　E　F

二进制数：1000　1001　1010　1011　1100　1101　1110　1111

【例 2-11】将十六进制数$(32A8.C69)_{16}$转换为二进制数，每位十六进制数用四位二进制数表示，过程如下：

3　2　A　8.　C　6　9

↓　↓　↓　↓　↓　↓　↓

0011　0010　1010　1000.　1100　0110　1001

所以，$(31A8.C69)_{16}=(11001010101000.110001101001)_2$

- 二进制数转换为十六进制数的方法是“四合一”。

整数部分：自右向左四位一组，不够位时补 0，每组对应一个十六进制数码。

小数部分：自左向右四位一组，不够位时补 0，每组对应一个十六进制数码。

【例 2-12】将二进制数$(1110110010110.010101101)_2$转换为十六进制数，从小数点开始分别向左或向右将每四位二进制数分成一组，过程如下：

0001　1101　1001　0110.　0101　0110　1000

↓　↓　↓　↓　↓　↓　↓

1　D　9　6.　5　6　8

所以，$(1110110010110.010101101)_2=(1D96.568)_8$

2.2　计算机中数值数据的表示及运算

2.2.1　基本概念

在计算机内部表示二进制数的方法通常称为数值编码，把一个数及其符号在机器中的表示加以数值化，这样的数称为机器数。机器数所代表的数称为该机器数的真值。

要全面完整地表示一个机器数，应考虑以下三个因素：

- 机器数的范围
- 机器数的符号
- 机器数中小数点的位置

1. 机器数的范围

通常机器数的范围由计算机的硬件决定。当使用 8 位寄存器时，字长为 8 位，所以一个无符号整数的最大值是：11111111B=255D，此时机器数的范围是 0～255。

当使用 16 位寄存器时，字长为 16 位，所以一个无符号整数的最大值是：1111111111111111B=FFFFH=65535D，此时机器数的范围是 0～65535。

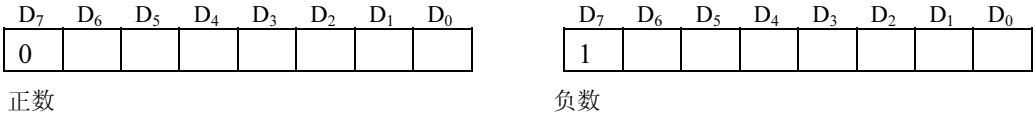
2. 机器数的符号

在算术运算中，数据是有正有负的，这类数据称为带符号数。

为了在计算机中正确地表示带符号数，通常规定每个字长的最高位为符号位，并用 0 表示正数，用 1 表示负数。

字长为 8 位二进制数时，D_7 为符号位；字长为 16 位二进制数时，D_{15} 为符号位。

例如：在一个 8 位字长的计算机中，带符号数据的格式如下：



正数　　负数

其中，最高位 D_7 是符号位，其余 D_6～D_0 为数值位，这样把符号数字化并和数值位一起编码的方法很好地解决了带符号数的表示及其计算问题。

这类编码方法常用的有原码、反码、补码三种。

3. 机器数中小数点的位置

在机器中，小数点的位置通常有两种约定：一种规定小数点的位置固定不变，这时的机器数称为“定点数”；另一种规定小数点的位置可以浮动，这时的机器数称为“浮点数”。

2.2.2　带符号数的原码、反码、补码表示

1. 原码

正数的符号位为 0，负数的符号位为 1，其他位按照一般的方法来表示数的绝对值。用这样的表示方法得到的就是数的原码。

【例 2-13】当机器字长为 8 位二进制数时：

X=+1011011　　$[X]_{原码}$=01011011

Y=+1011011　　$[Y]_{原码}$=11011011

$[+1]_{原码}$=00000001　　$[-1]_{原码}$=10000001

$[+127]_{原码}$=01111111　　$[-127]_{原码}$=11111111

在二进制数的原码表示中，0 的表示有正负之分：

$[+0]_{原码}$=00000000　　$[-0]_{原码}$=10000000

原码表示的整数范围是$-(2^{n-1}-1)$～$+(2^{n-1}-1)$，其中 n 为机器字长。

则：8 位二进制原码表示的整数范围是-127～+127。

16 位二进制原码表示的整数范围是-32767～+32767。

两个符号相异但绝对值相同的数的原码，除了符号位以外，其他位的表示都是一样的。数的原码表示简单直观，而且与其真值转换方便。

但是，如果有两个符号相异的数要进行相加或两个同符号数相减，则要做减法运算。做减法运算会产生借位的问题，很不方便。为了将加法运算和减法运算统一起来，以加快运算速度，引进了数的反码和补码表示。

2. 反码

对于一个带符号的数来说，正数的反码与其原码相同，负数的反码为其原码除符号位以外的各位按位取反。

【例 2-14】当机器字长为 8 位二进制数时：

X=+1011011　　$[X]_{原码}$=01011011　　$[X]_{反码}$=01011011

Y=-1011011　　$[Y]_{原码}$=11011011　　$[Y]_{反码}$=10100100

$[+1]_{反码}$=00000001　　$[-1]_{反码}$=11111110

$[+127]_{反码}$=01111111　　$[-127]_{反码}$=10000000

可以看出，负数的反码与负数的原码有很大的区别，反码通常用作求补码过程中的中间形式。

反码表示的整数范围与原码相同。

数据 0 在二进制数的反码表示中有以下形式：

$[+0]_{反码}$=$[+0]_{原码}$=00000000　　$[-0]_{反码}$=11111111

3. 补码

正数的补码与其原码相同，负数的补码为其反码在最低位加 1。

【例 2-15】X=+1011011　　$[X]_{原码}$=01011011　　$[X]_{补码}$=01011011

Y=-1011011　　$[Y]_{原码}$=11011011　　$[Y]_{反码}$=10100100

$[Y]_{补码}$=10100101

$[+1]_{补码}$=00000001　　$[-1]_{补码}$=11111111

$[+127]_{补码}$=01111111　　$[-127]_{补码}$=10000001

在二进制数的补码表示中，0 的表示是唯一的。

即：$[+0]_{补码}$=$[-0]_{补码}$=00000000

引入补码以后，加法和减法运算都可以统一用加法运算来实现，符号位也被当作数值参与处理。

在很多计算机系统中都采用补码来表示带符号的数。

由于计算机中存储数据的字节数是有限制的，所以能存储的带符号数也有一定的范围。

补码表示的整数范围是-2^{n-1}～$+(2^{n-1}-1)$，其中 n 为机器字长。

则：8 位二进制补码表示的整数范围是-128～+127。

16位二进制补码表示的整数范围是-32768～+32767。

当运算结果超出这个范围时，就不能正确表示数了，此时称为溢出。

对于 8 位字长的二进制数，其原码、反码、补码的对应关系如表 2-3 所示。

表 2-3　8 位二进制数的原码、反码、补码的对应关系

二进制数	无符号数	带符号数		
		原码	反码	补码
00000000	0	+0	+0	+0
00000001	1	+1	+1	+1
…	…	…	…	…
01111111	127	+127	+127	+127
10000000	128	-0	-127	-128
…	…	…	…	…
11111110	254	-126	-1	-2
11111111	255	-127	-0	-1

4. 补码与真值之间的转换

给定机器数的真值可以通过补码的定义来完成真值到补码的转换，若已知某数的补码求其真值，则可以采用以下方法来计算。

正数补码的真值等于补码的本身；负数补码转换为其真值时，将负数补码数值位按位求反，末位加 1，即可得到该负数补码对应的真值的绝对值。

【例 2-16】（1）$[X]_{补码}$=01011001B，求其真值 X。

（2）$[X]_{补码}$=11011001B，求其真值 X。

（1）由于$[X]_{补码}$代表的数是正数，则其真值：X=+1011001B

$=+(1\times2^6+1\times2^4+1\times2^3+1\times2^0)$

=+(64+16+8+1)

=+(89)D

（2）由于$[X]_{补码}$代表的数是负数，则其真值：X=-($[1011001]_{求反}$+1)B

=-(0100110+1)B

=-(0100111)B

$=-(1\times2^5+1\times2^2+1\times2^1+1\times2^0)$

=-(32+4+2+1)

=-(39)D

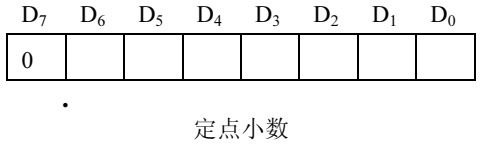
2.2.3　定点数和浮点数表示

计算机在进行算术运算时，需要指出小数点的位置。针对小数点的处理，计算机有两种表示数的方法，定点表示法和浮点表示法。

1. 定点表示法

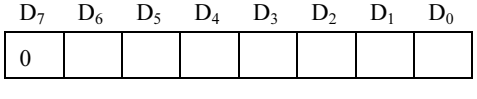
定点表示约定所有数据小数点的位置固定不变，通常把小数点固定在有效数字的前向或末尾，这就形成了两类定点数。

（1）定点小数。小数点固定在最高有效数字之前，符号位之后，则该数为一纯小数，格式如下：



定点小数

（2）定点整数。小数点固定在最低有效数字之后，则该数为整数，格式如下：



定点整数

在机器中，小数点的位置是隐含约定的，并不需要把它真正地表示出来。

（3）定点数的表数范围（字长为 n+1 位）。

	小数	整数
原码	$-(1-2^{-n})\leqslant N\leqslant 1-2^{-n}$	$-(2^n-1)\leqslant N\leqslant 2^n-1$
补码	$-1\leqslant N\leqslant 1-2^{-n}$	$-2^n\leqslant N\leqslant 2^n-1$
反码	$-(1-2^{-n})\leqslant N\leqslant 1-2^{-n}$	$-(2^n-1)\leqslant N\leqslant 2^n-1$

只有定点数据的计算机称为定点计算机。定点机只能表示纯小数或整数，而且所能表示的数值范围有限，尤其是定点小数的表数范围小于 1。所以，定点小数表示法只用在早期的计算机中，现代计算机都设计成能处理多种整数类型的计算机，如用 8 位、16 位、32 位或 64 位二进制数来表示一个整数。而定点小数讨论编码方便，也用于表示浮点数的尾数。

2. 浮点表示法

（1）浮点数。使用定点表示法能表示一定范围的整数，通过重新设定小数点的位置，这种格式也能用来表示带有分数的数。

定点表示法，因为小数点只能定在某一位置上，所以表数范围有限。为了表示更大范围的数据，数学上通常采用科学计数法，把数据表示成一个小数乘以一个以 10 为底的指数。

例如 368000000000000 可以表示成 3.68×10^{14}，而 0.0000000000000368 可以表示成 3.68×10^{-14}。显然这里小数点的位置可以动态变化，只要相应地改变 10 的指数就可以使整个数的值不变。

浮点表示法就是一个数的小数点的位置不固定，可以浮动。

对于任一数 N 可以表示成：

$$N=R^{E}\times M=\pm R^{\pm e}\times m$$

其中，E（Exponent）是指数，被称为浮点数的阶码，用定点整数表示。早期的计算机系统 E 用补码表示，此时需要设置符号位；现在计算机 E 多用移码表示。M（Mantissa）称为浮点数的尾数，用定点小数表示，尾数的符号表示数的正负，用补码或原码表示。R（Radix）是阶码的底，又称为尾数的基值。基值 R 在计算机中一般为 2、8 或 16，是个常数，在系统中是事先隐含约定的，不需要用代码表示。所以浮点数只需要用一对定点数（阶码和尾数）表示，存于如下一个二进制字的三个字段中。

阶符	阶码	尾数

其中，阶符表示数的正负，阶码表示小数点的位置，而尾数表示有效数字。

（2）表数范围。设 l 和 n 分别表示阶码和尾数的位数（均不包括符号位），基值为 2，阶码和尾数均采用原码表示，则浮点数的表数范围是：

$$0\leqslant|N|\leqslant 2^{2^{l}-1}\ (1-2^{-n})$$

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

如果用 32 位表示一个浮点数，数符占一位，阶码 8 位，尾数 23 位，则此浮点数的表数范围为：

$$-2^{2^{7}-1}(1-2^{-23})\leqslant N\leqslant 2^{2^{7}-1}(1-2^{-23})$$

N 的取值范围近似为 $\pm2^{127}$，相当于 $\pm10^{38}$。

如果采用定点小数，同样 32 位字长，则表数范围为：

$$-(2^{31}-1)\leqslant N\leqslant 2^{31}-1$$

显然浮点数的表数范围比定点数大得多。

在浮点数中，阶码指出小数点在数据中的实际位置，它决定浮点数的表数范围；而尾数可以给出有效数字的位数，它决定浮点数的表数精度。

2.2.4　定点补码加法运算溢出判断

补码的加法规则是：$[X+Y]_{补}=[X]_{补}+[Y]_{补}$。

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

掌握上述规则后，不仅可以把补码减法运算变为补码加法运算，而且可以把带符号数和无符号数统一起来。计算机内部采用统一的方法处理，即加法可直接进行，减法用减数变补与被减数相加来实现（结果为负数时，应对其再求补，以便将其转换为真值）。

【例 2-17】 设字长 n=8，X=66，Y=22，试用补码进行下列运算：

（1）X+Y　（2）X-Y　（3）-X+Y　（4）-X-Y

解：利用$[X\pm Y]_{补}=[X]_{补}+[\pm Y]_{补}$ 进行解答。

X=66=01000010B　Y=22=0001 0110B

$[X]_{补}$=01000010B　$[Y]_{补}$=0001 0110B　$[-X]_{补}$=10111110B　$[-Y]_{补}$=11101010B

① 01000010 $[X]_{补}$
+ 00010110 $[Y]_{补}$
01011000 $[X+Y]_{补}$

X+Y=0101 1000B=88

② 0100 0010 $[X]_{补}$
+ 1110 1010 $[-Y]_{补}$
1 0010 1100 $[X-Y]_{补}$
进位自动丢失

X-Y=0010 1100B=44

③ 1011 1110 $[-X]_{补}$
+ 0001 1110$[Y]_{补}$
1101 01 00$[-X+Y]_{补}$
结果为负，再求补

-X+Y=-010 1100B=-44

④ 1011 1110$[-X]_{补}$
+ 1110 1010 $[-Y]_{补}$
1 1010 1000 $[-X-Y]_{补}$
进位自动丢失，再求补

-X-Y=-101 1000B=-8

特别要注意的是溢出问题，即进行运算时由于计算机字长的限制，会产生运算结果超出数所能表示的范围的现象。可以用直接观察法来判断运算是否溢出：当正数加正数的结果为负数时或负数加负数的结果为正数时，结果都产生溢出。也可以用双高位法来判断运算是否溢出：$OV=Cs\oplus Cp$。式中 Cs 为加减运算中最高位（符号位）的进位值，Cp 为加减运算中最高数值位的进位值，当有进位时，取值为 1，无进位时，取值为 0。若 Cs、Cp 的异或运算结果为 1，即 OV=l，则表明结果产生溢出，反之则表示不溢出。

【例 2-18】 设$[X]_{补}$=0111 1111B，$[Y]_{补}$=0000 0110B，试用补码进行下列运算，并用直接观察法判断结果是否产生溢出。

（1）$[X+Y]_{补}$　（2）$[-X-Y]_{补}$

解：用直接观察法判断结果是否产生溢出。

① 0111 1111 $[X]_{补}$
+ 0000 0110 $[Y]_{补}$
1000 0101 $[X+Y]_{补}$
正加正，结果为负
$[X+Y]_{补}$ 溢出

② 1000 0001 $[-X]_{补}$
+ 1111 1010 $[-Y]_{补}$
0111 1011 $[-X-Y]_{补}$
负加负，结果为正
$[-X-Y]_{补}$ 溢出

【例 2-19】 设$[X]_{补}$=0111 1111B，$[Y]_{补}$=0000 0110B，试用补码进行下列运算，并用双高位法判断结果是否产生溢出。

（1）$[X+Y]_{补}$ （2）$[-X-Y]_{补}$

解：用双高位法判断结果是否产生溢出。

① 0111 1111 $[X]_{补}$
+ 0000 0110 $[Y]_{补}$
0 1000 0101 $[X+Y]_{补}$
Cs=0 Cp=1 OV= Cs ⊕ Cp=1
$[X+Y]_{补}$ 溢出

② 1000 0001 $[-X]_{补}$
+ 1111 1010 $[-Y]_{补}$
1 0111 1011 $[-X-Y]_{补}$
Cs=1 Cp=0 OV= Cs ⊕ Cp=1
$[-X-Y]_{补}$ 溢出

2.3 其他数据表示方法

前面我们讨论的数据类型都是数值数据，即计算机进行算术运算所使用的操作数，它有大小，可以在数轴上表示出来。但是计算机并不是一种仅仅能用来存储数字并进行高速计算的机器。在很多情况下，计算机处理的是另一种数据类型——非数值数据。例如，计算机使用者编写的程序是字符形式的，即包括字母、数字及各种特殊符号。另外，在情报检索、企业管理、办公自动化等许多应用场合，计算机处理的也是字符，很少处理数值。在中文信息处理系统中，计算机还要处理汉字。计算机能够接收这些字符和汉字，将它们存储在存储器中，对它们进行某些操作，并将结果送至输出设备。除此之外，在多媒体技术中，计算机还能处理图形、图像和语音等信息，这些信息在计算机内转换成 0、1 表示的编码。我们把字符、汉字、图形、图像、语音以及逻辑数据都称为非数值数据。随着计算机应用领域的不断扩大，计算机处理非数值数据远比处理数值数据多得多。

1. 逻辑数据

逻辑数据由若干位无符号的二进制数字串组成，每位之间没有权的内在联系，因此也就没有数值，只用逻辑值：真值或假值。

逻辑数据只能参加逻辑运算，如“逻辑加”、“逻辑乘”、“逻辑非”等以及它们的各种组合运算。其特点是参加运算的逻辑数据是在对应的两个二进制位之间进行的，与相邻的高位和低位的值均无关，不存在算术运算的进位和借位等问题。

数值数据和逻辑数据在机器内部都表示成二进制数串，如 10110110，机器本身不能识别它是哪种数据，必须根据程序中的指令来识别它。算术运算指令所指定的操作数一定是数值数据，而逻辑运算指令指定的操作数一定是逻辑数据。

2. 字符编码

字符在机器里也必须用二进制数来表示，但是这种二进制数是按照特定规则编码表示的。计算机为了识别和区分这些符号，采用了以下方法：

（1）使用由若干位组成的二进制数去代表一个符号。

（2）一个二进制数只能与一个符号唯一对应，即符号集内所有的二进制数不能相同。

这样，二进制数的位数自然取决于符号集的规模。

例如：128 个符号的符号集，需要 7 位二进制数。

256 个符号的符号集，需要 8 位二进制数。

这就是所谓的字符编码，由此可以看出：计算机解决任何问题都是建立在编码技术上的。目前最通用的西文字符编码是美国信息交换标准代码（ASCII 码）。

2.3.1　美国信息交换标准代码（ASCII 码）

ASCII（American Standard Code for Information Interchange）码是美国信息交换标准代码的简称，用于给西文字符编码，包括英文字母的大小写、数字、专用字符、控制字符等。

这种编码由 7 位二进制数组合而成，可以表示 128 种字符，目前在国际上广泛流行。ASCII 码的编码内容如表 2-4 所示。

表 2-4　7 位 ASCII 码编码表

低 4 位代码		高 3 位代码							
		0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P	、	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	“	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	’	7	G	W	g	w
8	1000	BS	CAN	（	8	H	X	h	x
9	1001	HT	EM	）	9	I	Y	i	y
A	1010	LF	SUB	*	：	J	Z	j	z
B	1011	VT	ESC	+	；	K	[	k	{
C	1100	FF	FS	，	<	L	\	l	|
D	1101	CR	GS	—	=	M	]	m	}
E	1110	SO	RS	.	>	N	↑	n	～
F	1111	SI	US	/	?	O	←	o	DEL

表中用英文字母缩写表示的“控制字符”在计算机系统中起各种控制作用，它们在表中占前两列，加上 SP 和 DEL，共 34 个；其余的是 10 个阿拉伯数字、52 个英文大小写字母、

32 个专用符号等“图形字符”可以显示或打印出来，共 94 个。

表 2-4 中“控制字符”的含义如表 2-5 所示。

表 2-5 ASCII 编码表中控制字符的含义

控制字符	含义	控制字符	含义	控制字符	含义
NUL	空	FF	走纸控制	CAN	作废
SOH	标题开始	CR	回车	EM	纸尽
STX	正文结束	SO	移位输出	SUB	减
ETX	本文结束	SI	移位输入	ESC	换码
EOT	传输结束	DLE	数据链换码	FS	文字分隔符
ENQ	询问	DC1	设备控制 1	GS	组分隔符
ACK	承认	DC2	设备控制 2	RS	记录分隔符
BEL	报警符	DC3	设备控制 3	US	单元分隔符
BS	退一格	DC4	设备控制 4	SP	空格
HT	横向列表	NAK	否定	DEL	作废
LF	换行	SYN	空转同步		
VT	垂直制表	ETB	信息组传输结束		

ASCII 码是 7 位二进制编码，而计算机的基本存储单位是字节（Byte），一个字节包含 8 个二进制位（bit）。因此，ASCII 码的机内码要在最高位补一个 0。在存储、处理和传送信息时，最高位常用作奇偶校验位，用来检验代码在存储和传送过程中是否发生错误。奇校验时，每个代码的二进制形式中应有奇数个 1；偶校验时，每个代码的二进制形式中应有偶数个 1。

后来 IBM 公司将 ASCII 码的位数增加了一位，用 8 位二进制数构成一个字符编码，共有 256 个符号。扩展后的 ASCII 码除了原来的 128 个字符外，又增加了一些常用的科学符号和表格线条。

2.3.2 二—十进制编码——BCD 码

BCD（Binary-Coded Decimal）码又称为“二—十进制编码”，专门解决用二进制数表示十进数的问题。

“二—十进制编码”的方法很多，有 8421 码、2421 码、5211 码、余 3 码等，最常用的是 8421 编码，其方法是用四位二进制数表示一位十进制数，自左至右每一位对应的位权是 8、4、2、1。

应该指出的是，4 位二进制数有 0000～1111 共 16 种状态，而十进制数 0～9 只取 0000～1001 的 10 种状态，其余 6 种不用。

8421 编码如表 2-6 所示。

表 2-6　8421 编码表

十进制数	8421 编码	十进制数	8421 编码
0	0000	8	1000
1	0001	9	1001
2	0010	10	0001　0000
3	0011	11	0001　0001
4	0100	12	0001　0010
5	0101	13	0001　0011
6	0110	14	0001　0100
7	0111	15	0001　0101

通常，BCD 码有两种形式：压缩 BCD 码和非压缩 BCD 码。

1. 压缩 BCD 码

压缩 BCD 码的每一位数采用 4 位二进制数来表示，即一个字节表示两位十进制数。

例如：二进制数 10001001B，采用压缩 BCD 码表示为十进制数 89D。

2. 非压缩 BCD 码

非压缩 BCD 码的每一位数采用 8 位二进制数来表示，即一个字节表示一位十进制数。而且用每个字节的低 4 位来表示 0～9，高 4 位为 0。

例如：十进制数 89D，采用非压缩 BCD 码表示为二进制数是 00001000　00001001B。

2.3.3　汉字编码

在我国，计算机的应用应该具有汉字信息处理能力，对于这样的计算机系统，除了配备必要的汉字设备和接口外，还应该装配有支持汉字信息输入、输出和处理的操作系统。汉字信息的输入、输出及其处理要比西文困难得多，原因是汉字的编码和处理非常复杂。经过多年的努力，我国在汉字信息处理的研制和开发方面取得了突破性进展，使我国的汉字信息处理技术处于世界领先地位。

1. 基本概念

计算机处理汉字信息的前提条件是对每个汉字进行编码，这些编码统称为汉字代码。在汉字信息处理系统中，对于不同部位，存在着多种不同的编码方式。比如，从键盘输入汉字使用的汉字代码（外码）就与计算机内部对汉字信息进行存储、传送、加工所使用的代码（内码）不同，但它们都是为系统各相关部分标识汉字使用的。

系统工作时，汉字信息在系统的各部分之间传送，它到达某个部分就要用该部分所规定的汉字代码表示汉字。因此，汉字信息在系统内传送的过程就是汉字代码转换的过程。这些代码构成该系统的代码体系，汉字代码的转换和处理是由相应的程序来完成的。

2. 汉字代码的表示方法

目前计算机中常用的汉字代码有以下几种：

（1）汉字输入码。汉字输入码是为用户由计算机外部输入汉字而编制的汉字编码，又称为汉字外部码，简称外码。汉字输入码位于人机界面上，面向用户，所以它的编码原则应该是

简单易记、操作方便、有利于提高输入速度。目前使用较多的有以下 4 类：

1）顺序码：将汉字按一定顺序排好，然后逐个赋予一个号码作为该汉字的编码。这种编码方法简单，但由于与汉字的特征没有联系，所以很难记忆。例如区位码、电报码等。

2）音码：根据汉字的读音进行编码。只要具有汉语拼音的基础就会掌握，这种编码的最大弱点是对于那些不知道读音的字无法输入。例如拼音码、自然码等。

3）形码：根据汉字的字形进行编码。一个汉字只要能写出来，即使不会读，也能得到它的编码。例如五笔字型、大众码等。

4）音形码：根据汉字的读音和字形进行编码。它的编码规则既与音素有关，又与形素有关，即取音码实施简单、易于接受的优点和形码形象、直观之所长，从而获得了较好的输入效果。例如双拼码、五十字元等。

（2）汉字机内码。汉字机内码是汉字处理系统内部存储、处理汉字而使用的编码，简称内码。在设计汉字内码时，应考虑以下基本原则：编码空间应该足够大；中西文兼容性要好；具有较好的定义完备性；编码要简单、系统应该容易实现；同时应与国家标准 GB2312-80 汉字字符集有简明的一一对应关系。

（3）汉字字形码。汉字字形码是表示汉字字形信息的编码。目前，在汉字信息处理系统中大多以点阵方式形成汉字，所以汉字字形码就是确定一个汉字字形点阵的代码。全点阵字形中的每一个点用一个二进制位来表示，随着字形点阵的不同，它们所需要的二进制位数也不同。

例如：24×24 的字形点阵，每字需要 72 字节；32×32 的字形点阵，每字需要 128 字节。与每个汉字对应的这一串字节就是汉字的字形码。

（4）汉字交换码。汉字交换码是汉字信息处理系统之间或通信系统之间传输信息时，对每个汉字所规定的统一编码。

我国已指定了汉字交换码的国家标准“信息交换用汉字编码字符集—基本集”，代号 GB2321-80，又称“国标码”。

实际上，汉字处理过程就是这些代码的转换过程。可以把汉字信息处理系统抽象为一个简单的模型，如图 2-1 所示。

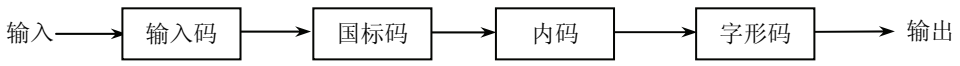


图 2-1　汉字信息处理系统

3. 几种常用的汉字编码

（1）区位码。将 GB2312-80 全部字符集组成的一个 94×94 的方阵，每一行称为一个“区”，编号从 01～94；每一列称为一个“位”，编号也是从 01～94。这样，每一个字符便具有一个区码和一个位码，将区码置前，位码置后，组合在一起就成为区位码。

因此，国标码与区位码是一一对应的。可以这样认为：区位码是十进制表示的国标码，国标码是十六进制表示的区位码。

例如：汉字“中”在第 54 行、第 48 列的位置，它的区码是 54，位码是 48，所以区位码就是 5448。在选择区位码作为汉字输入码时，只要键入 5448，便输入了“中”字。

区位码中的所有 94 个区划分为如下四个部分：

1）1～15 区：图形符号区。其中 1～9 区为标准区，10～15 为自定义符号区。

2）16～55 区：一级汉字区。该区的汉字按汉语拼音排序，同音字按笔画顺序排序，55 区的 90～94 位未定义汉字。

3）56～87 区：二级汉字和偏旁部首区，该区按笔画顺序排序。

4）88～94 区：自定义汉字区。

（2)国标码。我国制定的“中华人民共和国国家标准信息交换汉字编码”(代号 GB2312-80）就是国标码。该码规定：一个汉字用两个字节表示，每个字节只用 7 位，与 ASCII 码相似。

国标码字符集共收录汉字和图形符号 7445 个，其中一级常用汉字 3755 个，二级非常用汉字和偏旁部首 3008 个，图形符号 682 个。

图形符号中包括：一般符号、序号、数字、英文字母、日文假名、希腊字母、俄文字母、汉语拼音字母、汉语拼音符号等。

在这个字符集中，汉字的选择是按使用频度确定的，其中 6763 个一二级汉字的使用覆盖率达到 99.9%。

国标码是所有汉字编码都应该遵循的标准，自公布这一标准后，汉字机内码的编码、汉字字库的设计、汉字输入码的转换、输出设备的汉字地址码等都以此标准为基础。

我国大陆使用的汉字机内码就是将两个字节各 7 位的国标码经过如下转换形成的：

1）用两个字节各 8 位来表示一个汉字的机内码。

2）在原来国标码的基础上，将两个字节的最高位置 1（避免了与 ASCII 码的冲突)。

以汉字“大”为例，将它的国标码、机内码、对应两个字节的 ASCII 码进行比较，如表 2-7 所示。

表 2-7　汉字“大”的国标码、机内码、两字节的 ASCII 码比较

名称	编码（十进制）	编码（二进制）
国标码	3473	00110100　01110011
机内码	B4F3	10110100　11110011
ASCII 码	3473	00110100　01110011

可以看出：同一个汉字的国标码与机内码相比，只在两个字节的最高位有差别。前者为 0，后者为 1，另外的 7 位则完全相同。由此体现了机内码与国标码有着明确的一一对应关系。

当机内码与国标码完全相同时，这两个字节肯定代表两个西文字符。由此体现了汉字机内码与 ASCII 码的兼容性。

（3）机内码。为了在内部能区分汉字与 ASCII 字符，把两个字节汉字国标码中每个字节的最高位置“1”，这样就形成了汉字的另外一种编码，称为汉字机内码（内码)。如果已知国标码，则机内码就唯一确定，即机内码的每个字节为国标码相应字节加 80H。内码用于统一不同系统所使用的不同汉字输入码，使花样繁多各种不同的汉字输入法进入系统后一律转换为内码，致使不同系统的汉字信息可以相互转换。

因此，国标码、区位码与机内码的关系为：

机内码的每个字节=国标码的相应字节+80H

国标码的每个字节=区位码相应字节+20H

（4）BIG-5 码。BIG-5 码是我国台湾地区编制和使用的一套中文内码。它是为了解决各

生产厂家中文内码不统一的问题而设计出的一套编码，并采用五大套装软件的“五大”命名为“BIG-5”码，俗称“大五码”。

BIG-5 码也是采用两个字节编码，取值范围如下：

第一字节：81H～8DH、8EH～A0H、A1H～FEH。

第二字节：40H～7EH、A1H～FEH。

BIG-5 码包括常用字 5401 个、次常用字 7652 个、特殊字符和图形字符 441 个，共计 13053 个编码。

（5）GB13000 码。为了统一表示世界各国的文字，国际标准化组织（ISO）于 1993 年公布了“通用多八位编码字符集”的国际标准（ISO/IEC 10646），简称 UCS。该标准对包括汉字在内的各种文字都规定了统一的编码方案。与此相适应，以我国为主，联合日、韩等国家和地区，经过互相认同与合并后，确定了 20902 个中日韩统一汉字，并被正式批准为 ISO/IEC 10646 中的汉字字符集。我国发布了与其一致的国家标准，即 GB13000 码，并提出相应的机内码扩充规范。目前 Windows 95、Windows 98 中文版都使用这种机内码表示汉字。

2.3.4 图像（图形）信息的表示方法

一幅模拟连续图像在输入到计算机时，必须通过输入设备（扫描仪、摄像机等）将其变为数字图像才能存储和处理。图像的数字化包括采样和量化两个步骤。空间坐标的数字化称为图像采样，幅度的数字化称为图像量化。连续图像 f(x,y)可以按等间隔采样，将（x，y）平面分成网眼似的小方格，则 f(x,y)可以表示为一个 M×N 数组。

$$f(x,y)=\begin{bmatrix} f(0,0), f(0,1), \cdots, f(0,M-1) \\ f(1,0), f(1,1), \cdots, f(1,M-1) \\ \vdots \\ f(N-1,0), f(N-1,1), \cdots, f(N-1,M-1) \end{bmatrix}$$

数组中的每个元素称为一个像素，每个像素的灰度再进行数字化，通常用 K 级（比特数）表示。经过上述处理后，一幅数字化图像所需要的存储位可用下式表示：

$$b=M\times N\times K$$

存储一幅数字图像所需的比特数通常很大，例如一幅 128×128、64 个灰度级的图像需要 98304b 来存储，而一幅 512×512、256 个灰度级的图像需要 2097152b 来存储。

图像分辨率（区分细节的程度）与采样点和灰度级两个参数紧密相关。理论上讲，这两个参数越大，离散数组与原始图像就越接近，但相应地存储和处理的需求也将随 N、M 和 K 的增加而迅速增加。

2.3.5 语音信息的表示方法

语音在计算机中的表示相对比较复杂，因为语音是模拟量，但计算机能处理的只是数字量，这就需要把语音的模拟量通过 A/D 设备转换成计算机能处理的二进制组成的数字量。人的语音通过拾音设备把声音信号转换成频率、幅度连续变化的电流信号——模拟量，通过计算机多媒体声卡（或声霸卡）对电信号采样（标准采样频率可以是 11.025kHz、22.05kHz 和 44.1kHz 三种，声卡的采样点可以是 8 位或 16 位样本信息量），得到与此信号相对应的电流或电压的离

散值。采样频率越高，采样点越密，需要存储采样点数值的空间就越大，且越接近模拟量的波形。单声道需要存储字节数=(采样频率×每个采样波形点的位数×时间)/8。若采用 16 位声卡，采样频率为 44.1kHz，单声道声音每分钟需要 5.29MB 存储空间，则一小时就需要 317.52MB 存储空间。若采用双通道（立体声），则相同采样频率和时间比单声道需要增加一倍的存储空间。因此，可以看出声音文件要占据较大的存储空间。语音信息由模拟量转换成数字量的过程如图 2-2 所示。

为了得到较小声音文件的存储空间，必须对其声音文件进行压缩。对已得到的声音文件可以进行编辑操作（包括删除、增加、粘贴等），然后把该声音文件通过声卡中的相反转换（D/A）设备把数字量转换成模拟量后，可用扬声器把声音还原输出。

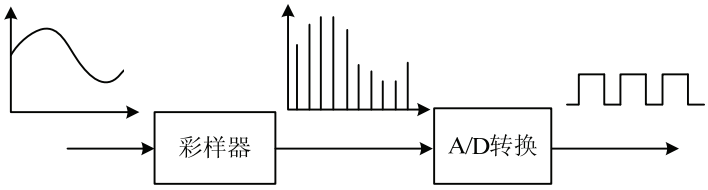


图 2-2　模拟量转换成数字量

我们这里只讨论了语音在计算机中的表示问题，至于要把语音输入变成文字文件，这是一个极其复杂的语音识别过程；而要把计算机内的文字转换成语音信息通过扬声器输出，则是语音的合成问题。这两方面的内容涉及的知识很多，此处不再讨论。

计算机语音处理系统已经在各个领域中得到了广泛的应用，一旦自然语言的语音识别和自然文字的识别真正解决后，将在计算机多媒体领域中发生一场深刻的变革。

本章小结

各种信息在计算机中均表示为二进制数据。但是二进制数据书写比较冗长、容易出错，所以人们在数据的表示中又引入了八进制、十进制和十六进制。此外，为了表示数的符号位，又引入了数的原码、反码和补码的概念。为了描述某种信息，在计算机领域还有一些编码方式，例如表示西文和 I/O 设备动作的 ASCII 码、表示十进制数据的 BCD 码、表示汉字的国标码等。

本章着重介绍了计算机中数据的表示方法，重点处理了二进制数、八进制数、十进制数、十六进制数的相关概念及各类数制之间相互转换的方法、无符号数和带符号数的机器内部表示、字符编码和汉字编码等。通过本章的学习，读者要掌握计算机内部的信息处理方法和特点，熟悉各类数制之间的相互转换，理解无符号数和带符号数的表示方法，掌握 BCD 码和字符的 ASCII 码以及汉字编码及其应用。

习题2

1．简述计算机中“数”和“码”的区别，计算机中常用的数制和码制有哪些？

2．将下列十进制数分别转换为二进制数、八进制数、十六进制数和压缩 BCD 数。

（1）125.74　　（2）513.85　　（3）742.24

（4）69.357　　（5）158.625　　（6）781.697

3．将下列二进制数分别转换为十进制数、八进制数和十六进制数。

（1）101011.101　　（2）110110.1101　　（3）1001.11001　　（4）100111.0101

4．将下列十六进制数分别转换为二进制数、八进制数、十进制数和压缩 BCD 数。

（1）5A.26　　（2）143.B5　　（3）6AB.24　　（4）E2F3.2C

5．根据 ASCII 码的表示，查表写出下列字符的 ASCII 码。

（1）0　　（2）9　　（3）K　　（4）G　　（5）t

（6）DEL　　（7）ACK　　（8）CR　　（9）$　　（10）＜

6．写出下列十进制数的原码、反码、补码表示（采用 8 位二进制数，最高位为符号位）。

（1）104　　（2）52　　（3）-26　　（4）-127

7．已知补码，求出其真值。

（1）48H　　（2）9DH　　（3）B2H　　（4）4C10H

8．已知某个 8 位的机器数 65H，在其作为无符号数、补码带符号数、BCD 码以及 ASCII 码时分别表示什么真值和含义？

9．ASCII 码是由几位二进制数组成的？它可以表示哪些信息？

10．中文信息如何在计算机内表示？