# 第 1 章　数的表示与运算

## 1.1　数　制

### 1.1.1　数制的表示

**1．计数制**

数制也称为计数制，是指用一组固定的数字符号和统一的规则表示数的方法。对于任意 $r$ 进制数 $x$，可以用下式表示为

$$\sum_{i=-m}^{n} a_i r^i = a_{-m}r^{-m} + \cdots + a_{-2}r^{-2} + a_{-1}r^{-1} + a_0 r^0 + a_1 r^1 + \cdots + a_n r^n$$

其中：

① $a_i$ 为数码，每种进制数都由固定的数字符号来表示，这个符号称为：数码。

二进制有 2 个数码：0 和 1；

八进制有 8 个数码：0、1、2、3、4、5、6 和 7；

十进制有 10 个数码：0、1、2、3、4、5、6、7、8 和 9；

十六进制有 16 个数码：0、1、2、3、4、5、6、7、8、9、A、B、C、D、E 和 F（字母不区分大小写）。

② $i$ 为数位，数位是指数码在一个数中所处的位置。

例如，十六进制数 56.78 从左到右的数位分别为：1、0、−1 和−2。

③ $r$ 为基数，基数是指在某计数制中，每个数位上能使用的数码的个数。

二进制基数为 2；

八进制基数为 8；

十进制基数为 10；

十六进制基数为 16。

④ $r^i$ 为权，权是基数的幂，幂由数位决定。

二进制第 $i$ 位上的权为 $2^i$；

八进制第 $i$ 位上的权为 $8^i$；

十进制第 $i$ 位上的权为 $10^i$；

十六进制第 $i$ 位上的权为 $16^i$。

例如，十六进制数 56.78 从左到右每位的权分别为：$16^1$、$16^0$、$16^{-1}$ 和 $16^{-2}$。

**2．计算机中常用的计数制**

在日常生活中，人们最常用的是十进制计数制。而在计算机中，为了便于数的存储和表示，使用的是二进制计数制。由于二进制数据书写和记忆不方便，在计算机中还常使用八进制和十六进制等计数制。计算机中常用计数制见表 1-1。

❋ 说明

- 为了区别所使用的计数制，一般用以下两种书写格式表示：

① 用括号将数字括起，后面加数制区分，数制用下标的形式给出；

② 用后缀区分，二进制数、十进制数、八进制数、十六进制数的后缀分别为字母 B（或 b）、D（或 d）、O（或 o）或 Q（或 q）、H（或 h）。

表 1-1　计算机中常用计数制

| 数制 | 基数 | 数码 | 运算规则 | 书写后缀 |
|---|---|---|---|---|
| 二进制 | 2 | 0, 1 | 逢二进一，借一当二 | B |
| 八进制 | 8 | 0, 1, 2, 3, 4, 5, 6, 7 | 逢八进一，借一当八 | O 或 Q |
| 十进制 | 10 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 | 逢十进一，借一当十 | D |
| 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，借一当十六 | H |

例如：十六进制数 56.78 可以表示成$(56.78)_{16}$或 56.78H；

十进制数 56.78 可以表示成$(56.78)_{10}$或 56.78D。

- 汇编程序规定，使用首字符是字母的十六进制数时，前面需加 0 来表示。

例如：B56.A8H 在汇编程序中需采用 0B56.A8H 这种表示形式。

- 在没有混淆的情况下，十进制数可以省略下标或后缀 D（或 d）。

### 1.1.2　数制之间的转换

**1．其他数制数转换为十进制数**

二进制数、八进制数和十六进制数转换为十进制数的方法是——按权展开。

**【例 1-1】**将 1010.101B、23.4Q 和 FA3.4H 转换成十进制数。

**解**　$1010.101B = 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} = 10.625D$

$23.4Q = 2\times8^1 + 3\times8^0 + 4\times8^{-1} = 19.5D$

$FA3.4H = 15\times16^2 + 10\times16^1 + 3\times16^0 + 4\times16^{-1} = 4003.25D$

**2．十进制数转换为其他数制数**

把十进制数转换为其他数制数的方法很多，通常采用的方法有降幂法和乘除法。

（1）降幂法

假设要转换的十进制数为 $N$。

步骤 1：找出最接近 $N$ 并小于等于 $N$ 的 $r$ 进制位权值 $r^i$；

步骤 2：找到满足 $0 \leqslant C < r$ 的最大数 $C$，使得 $N - C\times r^i < r^i$，$C$ 即为转换结果（$r$ 进制数）第 $i$ 位的位码 $a_i$；

步骤 3：计算 $N - C\times r^i$，并用此值作为新的 $N$ 值，即 $N \leftarrow N - C\times r^i$；

步骤 4：$i$ 自减 1，即 $i \leftarrow i-1$，得到下一个位权值 $r^i$。

重复步骤 2～步骤 4，直至 $N$ 为 0 或转换结果达到所需精度。

（2）乘除法

采用乘除法把十进制数转换为二、八或十六进制数的过程是：待转换的数的整数部分除以基数取余，直至商为0；小数部分乘以基数取整，直至积为整数或转换结果的小数位数达到所需精度要求。

**【例 1-2】**把十进制数 117.8125 转换成二进制数。

**解**　方法一：降幂法

小于该数 117.8125 的位权值有：64、32、16、8、4、2、1、0.5、0.25、0.125 和 0.0625，按下列过程求出每位的位码。

| $N$ | $C$ | | $r^i$ | | | $a_i$ | |
|---|---|---|---|---|---|---|---|
| 117 | −1 | × | $2^6$ | = | 53 | （$a_6$=1） | 高位 |
| 53 | −1 | × | $2^5$ | = | 21 | （$a_5$=1） | |
| 21 | −1 | × | $2^4$ | = | 5 | （$a_4$=1） | |
| 5 | −0 | × | $2^3$ | = | 5 | （$a_3$=0） | |
| 5 | −1 | × | $2^2$ | = | 1 | （$a_2$=1） | |
| 1 | −0 | × | $2^1$ | = | 1 | （$a_1$=0） | |
| 1 | −1 | × | $2^0$ | = | 0 | （$a_0$=1） | |
| 0.8125 | −1 | × | $2^{-1}$ | = | 0.3125 | （$a_{-1}$=1） | |
| 0.3125 | −1 | × | $2^{-2}$ | = | 0.0625 | （$a_{-2}$=1） | |
| 0.0625 | −0 | × | $2^{-3}$ | = | 0.0625 | （$a_{-3}$=0） | |
| 0.0625 | −1 | × | $2^{-4}$ | = | 0 | （$a_{-4}$=1） | 低位 |

根据上述过程，可求得 117.8125D=1110101.1101B。

方法二：乘除法

运算过程如下：

整数部分：

| | 商 | 余数 | |
|---|---|---|---|
| 117÷2＝ | 58 | → 1 | 整数部分低位 |
| 58÷2＝ | 29 | → 0 | |
| 29÷2＝ | 14 | → 1 | |
| 14÷2＝ | 7 | → 0 | |
| 7÷2＝ | 3 | → 1 | |
| 3÷2＝ | 1 | → 1 | |
| 1÷2＝ | 0 | → 1 | 整数部分高位 |

小数部分：

| | 积 | 整数 | |
|---|---|---|---|
| 0.8125×2＝ | 1.625 | → 1 | 小数部分高位 |
| 0.625 ×2＝ | 1.25 | → 1 | |
| 0.25 ×2＝ | 0.5 | → 0 | |
| 0.5 ×2＝ | 1.0 | → 1 | 小数部分低位 |

**❋ 说明**

- 整数部分取余数时，先得到的数值是转换结果整数部分的低位，后得到的是转换结果整数部分的高位；
- 小数部分取整时，先得到的数值是转换结果小数部分的高位，后得到的是转换结果小数部分的低位。

根据上述过程，可求得：117.8125D=111 0101.1101B。

**【例 1-3】**把十进制数 48956 转换成十六进制数。

**解** 方法一：降幂法

小于该数 48956 的位权值有：4096、256、16 和 1，按下列过程求出每位的位码。

| $N$ | $C$ | | $r^i$ | | | $a_i$ | |
|---|---|---|---|---|---|---|---|
| 48956 | −11 | × | $16^3$ | = | 3900 | （$a_3$=B，11 的十六进制数码为 B） | 高位 |
| 3900 | −15 | × | $16^2$ | = | 60 | （$a_2$=F，15 的十六进制数码为 F） | |
| 60 | −3 | × | $16^1$ | = | 12 | （$a_1$=3，3 的十六进制数码为 3） | |
| 12 | −12 | × | $16^0$ | = | 0 | （$a_0$=C，12 的十六进制数码为 C） | 低位 |

根据上述过程，可求得 48956D= BF3CH。

方法二：乘除法

商　余数

48956÷16＝3059 →12（对应的十六进制数码为C） 低位

3059÷16＝ 191 → 3（对应的十六进制数码为3） ↑

191÷16＝ 11 →15（对应的十六进制数码为F） |

11÷16＝ 0 →11（对应的十六进制数码为B） 高位

根据上述过程，可求得 48956D=BF3CH。

**3．其他数制数之间的转换**

（1）二进制数与八进制数之间的转换

由于八进制数以 8 为基数，而 $8=2^3$，所以 3 位二进制数对应 1 位八进制数，对应关系见表 1-2。

**表 1-2　二进制数与八进制数对应表**

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

二进制数转换为八进制数的转换过程是：以小数点为界，整数部分向左，小数部分向右，每 3 位二进制数为一组，用 1 位八进制数表示；不足 3 位的，整数部分高位补 0，小数部分低位补 0。

八进制数转换为二进制数的过程与上述过程相反，把每位八进制数用 3 位二进制数表示即可。

**【例 1-4】**把数 11010.101B 转换为八进制数。

11010.101B = 011 010 .101B=32.5Q

**【例 1-5】**把数 34.56Q 转换为二进制数。

34.56Q =011 100.101 110B=11100.10111B

（2）二进制数与十六进制数之间的转换

由于十六进制数以 16 为基数，而 $16=2^4$，所以 4 位二进制数对应 1 位十六进制数，对应关系见表 1-3。

**表 1-3　二进制数与十六进制数对应表**

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

二进制数转换为十六进制数的过程是：以小数点为界，整数部分向左，小数部分向右，每 4 位二进制数为一组，用 1 位十六进制数表示；不足 4 位的，整数部分高位补 0，小数部分低位补 0。

十六进制数转换为二进制数时的过程与上述过程相反，把每位十六进制数用 4 位二进制数表示即可。

**【例 1-6】**把二进制数 11010.101B 转换为十六进制数。

11010.101B =0001 1010.1010B = 1A.AH

【例 1-7】把十六进制数 56.78H 转换为二进制数。

56.78H =0101 0110.0111 1000B = 1010110.01111B

# 1.2 二进制数的表示与运算

任意一个二进制数 $x$ 都可以表示为

$$x=(-1)^S\times M\times 2^E$$

式中，$S$ 表示符号位，$S$=0 时，$x$ 为正数；$S$=1 时，$x$ 为负数。$M$ 是尾数，$E$ 是阶码。当 $E$ 是固定值时，$x$ 的小数点位置固定，称为定点数；当 $E$ 的值可变时，$x$ 的小数点位置是浮动的，称为浮点数。

在计算机中，常用的定点数有：纯整数和纯小数。本书只涉及定点纯整数的表示与运算。关于浮点数等相关知识，读者可以自行参考“计算机组成原理”等课程的相关内容进行学习。

计算机处理的数包括有符号数和无符号数两种类型。无符号数不分正负，有符号数有正数和负数之分。计算机中对于无符号数和有符号数的处理方法是不一样的。读者在处理数时，要注意区分。

有符号数的二进制格式中包括符号位和数值位两部分，通常用最高位作为符号位。有符号数连同符号位在内的数值化表示形式称为机器数，而这个数本身的值称为真值。而有符号数的运算则根据编码方式的不同，有不同的运算规则。

## 1.2.1 无符号二进制数的表示

在某些情况下，计算机要处理的数全是正数，此时不需要考虑符号位的表示问题，这样的数称为无符号数。无符号数的二进制格式中的数位都是数值位。8 位无符号整数的表数范围是：0～255D，16 位无符号整数的表数范围是：0～65535D。

在计算机中，无符号整数常用来表示地址。

## 1.2.2 无符号二进制数的运算

在计算机中，无符号数的运算采用二进制数的算术和逻辑运算规则进行运算。

### 1．算术运算规则

二进制数的算术运算包括加法、减法、乘法和除法 4 种，运算规则见表 1-4。

表 1-4 二进制数的算术运算规则

| 运算名 | 运算符 | 运算规则 | 说明 |
|---|---|---|---|
| 加法 | + | 0 + 0 = 0　1 + 0 = 1　0 + 1 = 1　1 + 1 = 10 | 逢二进一 |
| 减法 | − | 0−0= 0　1−0 = 1　0−1 =1　1−1= 0 | 借一当二 |
| 乘法 | × | 0 × 0=0　0 × 1=0　1 × 0=0　1 × 1=1 | |
| 除法 | ÷ | 0÷1=0　1÷1=1 | 除数不得为 0 |

### 2．逻辑运算规则

二进制数常用的逻辑运算有与、或、非和异或 4 种，运算规则见表 1-5。

【例 1-8】无符号二进制数的算术运算举例。

1010 1010B+0101 1101B = 1 0000 0111B

1010 1010B−0101 1101B = 0100 1101B

表 1-5　二进制数常用的逻辑运算规则

| 运算名 | 运算符 | 运算规则 |
|---|---|---|
| 与（AND） | ∧ | $0\wedge0=0$　$0\wedge1=0$　$1\wedge0=0$　$1\wedge1=1$ |
| 或（OR） | ∨ | $0\vee0=0$　$0\vee1=1$　$1\vee0=1$　$1\vee1=1$ |
| 非（NOT） | – | $\overline{0}=1$　$\overline{1}=0$ |
| 异或（XOR） | ⊕ | $0\oplus0=0$　$0\oplus1=1$　$1\oplus0=1$　$1\oplus1=0$ |

【例 1-9】无符号二进制数的逻辑运算举例。

1010 1010B ∧ 0101 1101B = 0000 1000B

1010 1010B ∨ 0101 1101B = 1111 1111B

1010 1010B ⊕ 0101 1101B = 1111 0111B

### 1.2.3　有符号二进制数的表示

机器数可以有多种不同的编码表示方法，常见的编码方式有：原码、反码和补码。

**1．原码**

原码表示法规定：最高位是符号位，用 0 表示正数，用 1 表示负数。数值部分用该数的二进制绝对值表示。

数 $x$ 的原码记作$[x]_{原}$，如机器字长为 $n$，则整数原码的定义如下

$$[x]_{原}=\begin{cases}x & 0\leqslant x\leqslant 2^{n-1}-1\\ 2^{n-1}+|x| & -(2^{n-1}-1)\leqslant x\leqslant 0\end{cases}$$

当机器字长 $n$=8 时，有：

$[+0D]_{原}$= 0000 0000，　$[-0D]_{原}$= 1000 0000

$[+1D]_{原}$= 0000 0001，　$[-1D]_{原}$= 1000 0001

$[+45D]_{原}$= 0010 1101，　$[-45D]_{原}$= 1010 1101

$[+127D]_{原}$= 0111 1111，$[-127D]_{原}$= 1111 1111

当机器字长 $n$=16 时，有：

$[+0D]_{原}$=0000 0000 0000 0000，　$[-0D]_{原}$= 1000 0000 0000 0000

$[+1D]_{原}$=0000 0000 0000 0001，　$[-1D]_{原}$= 1000 0000 0000 0001

$[+45D]_{原}$=0000 0000 0010 1101，　$[-45D]_{原}$= 1000 0000 0010 1101

$[+32767D]_{原}$=0111 1111 1111 1111，$[-32767D]_{原}$= 1111 1111 1111 1111

按照定义，设 $n$ 为机器字长，则原码的表数范围是：$-(2^{n-1}-1)$～$+(2^{n-1}-1)$。

例如，8 位二进制原码的表数范围是：−127D～+127D，16 位二进制原码的表数范围是：−32767D～+32767D。

原码表示法简单直观，与真值之间的转换方便，但由于符号位不能参与运算，而且对于数 0 有+0 和−0 两种表示形式，所以用它进行运算不方便。

**2．反码**

反码表示法规定：一个正数的反码和原码相同；一个负数的反码的符号位与其原码的符号位相同，数值位通过对其原码的数值部分按位求反得到。

数 $x$ 的反码记作$[x]_{反}$，如机器字长为 $n$，则整数反码的定义如下

$$[x]_{反}=\begin{cases}x & 0\leqslant x\leqslant 2^{n-1}-1\\ (2^n-1)-|x| & -(2^{n-1}-1)\leqslant x\leqslant 0\end{cases}$$

当机器字长 $n$=8 时，有：

$[+0D]_{反}$= 0000 0000，　$[-0D]_{反}$= 1111 1111

$[+1D]_{反}$= 0000 0001，　$[-1D]_{反}$= 1111 1110

$[+45D]_{反}$= 0010 1101，　$[-45D]_{反}$= 1101 0010

$[+127D]_{反}$= 0111 1111，$[-127D]_{反}$= 1000 0000

当机器字长 $n$=16 时，有：

$[+0D]_{反}$= 0000 0000 0000 0000，　$[-0D]_{反}$= 1111 1111 1111 1111

$[+1D]_{反}$= 0000 0000 0000 0001，　$[-1D]_{反}$= 1111 1111 1111 1110

$[+45D]_{反}$= 0000 0000 0010 1101，　$[-45D]_{反}$= 1111 1111 1101 0010

$[+32767D]_{反}$= 0111 1111 1111 1111，$[-32767D]_{反}$= 1000 0000 0000 0000

按照定义，设 $n$ 为机器字长，则反码的表数范围是：$-(2^{n-1}-1)$～$+(2^{n-1}-1)$。

例如，8 位二进制反码的表数范围是−127D～+127D，16 位二进制反码的表数范围是−32767D～+32767D。

反码与原码的表数范围相同，而且数 0 有+0 和−0 两种表示形式。所以，用反码进行运算也不方便。

根据反码求真值的方法是：若反码的最高位为 0，则该数是正数，其后的数值部分就是其真值；若反码的最高位为 1，则该数是负数，将其后的数值部分按位取反后，即可得到真值。

**3．补码**

补码表示法规定：一个正数的补码和反码、原码相同；一个负数的补码的符号位与其原码的符号位相同，其余位可通过将其反码数值部分加 1 得到（但有一个负数例外，详见表 1-6 下面的说明）。

数 $x$ 的补码记作$[x]_{补}$，如机器字长为 $n$，则整数补码的定义如下

$$[x]_{补}=\begin{cases}x & 0\leqslant x\leqslant 2^{n-1}-1\\ 2^n-|x| & -(2^{n-1}-1)\leqslant x\leqslant 0\end{cases}$$

当机器字长 $n$= 8 时，有：

$[+0D]_{补}$= 0000 0000，　$[-0D]_{补}$= 0000 0000

$[+1D]_{补}$= 0000 0001，　$[-1D]_{补}$= 1111 1111

$[+45D]_{补}$= 0010 1101，　$[-45D]_{补}$= 1101 0011

$[+127D]_{补}$= 0111 1111，　$[-127D]_{补}$= 1000 0001

按照定义，设 $n$ 为机器字长，则补码的表数范围是：$-2^{n-1}$～$+(2^{n-1}-1)$。

例如，8 位二进制补码的表数范围是−128D～+127D，16 位二进制补码的表数范围是−32768D～+32767D。

补码比原码、反码所能表示的数的范围大，数 0 的补码只有一种表示形式。

根据补码求真值的方法是：若补码的最高位为 0，则该数是正数，其后的数值部分就是其真值；若反码的最高位为 1，则该数是负数，将其后的数值部分按位取反加 1 后，即可得到真值（但有一个负数例外，详见表 1-6 下面的说明）。

表 1-6 是部分 8 位二进制数的原码、反码和补码表示形式的对照表。

**表 1-6　部分 8 位二进制数的原码、反码和补码对照表**

| 真值 | | 有符号数 | | |
|---|---|---|---|---|
| 十进制格式 | 二进制数格式 | 原码 | 反码 | 补码 |
| 0 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 |
| 1 | 0000 0001 | 0000 0001 | 0000 0001 | 0000 0001 |
| … | … | … | … | … |
| +126 | 0111 1110 | 0111 1110 | 0111 1110 | 0111 1110 |
| +127 | 0111 1111 | 0111 1111 | 0111 1111 | 0111 1111 |
| −128 | −1000 0000 | 无 | 无 | 1000 0000 |
| −127 | −0111 1111 | 1111 1111 | 1000 0000 | 1000 0001 |
| … | … | … | … | … |
| −1 | −0000 0001 | 1000 0001 | 1111 1110 | 1111 1111 |
| −0 | −0000 0000 | 1000 0000 | 1111 1111 | 0000 0000 |

特别说明：−128D 没有原码和反码，其补码通过定义式求得。

### 1.2.4　有符号二进制数的运算

计算机中普遍采用补码来表示有符号数。补码的算术运算包括加减运算和乘除运算，本书仅讨论补码的加减运算。补码的逻辑运算规则与 1.2.2 节介绍的无符号数的逻辑运算规则相同。

**1．补码的加减运算规则**

采用补码表示的有符号数进行加减运算时，符号位和数值位同时参与运算，运算结果仍然是补码。任何两数相加，无论正负，只要把它们的补码相加即可。加法运算得到的结果是两数和的补码。任何两数相减，无论正负，只要把减数相反数的补码与被减数的补码相加即可。减法运算结果是两数差的补码。运算公式如下

$$[x+y]_{补}=[x]_{补}+[y]_{补}$$
$$[x-y]_{补}=[x]_{补}+[-y]_{补}$$

从上面的公式可以看出，补码的减法运算可以转换成加法来完成，因此，在计算机中利用加法器就可以实现补码的加法和减法运算。

**2．补码加减运算的溢出判断**

由于计算机的字长有限，因此，所能表示的数是有范围的。当运算结果超过这个范围时，运算结果将出错，这种情况称为溢出。产生溢出的原因是：数值的有效位占据了符号位。补码加减运算溢出的判定有以下两种方法。

（1）利用符号位判别运算结果是否溢出

- 若两个同号数相加，结果的符号位与之相反，则溢出；
- 若两个异号数相减，结果的符号位与减数相同，则溢出；
- 若两个异号数相加或两个同号数相减，则不溢出。

（2）利用运算过程中的进位产生情况判别运算结果是否溢出

- 若次高位（最高数值位）和最高位（符号位）不同时产生进位或借位，则溢出；
- 若次高位（最高数值位）和最高位（符号位）都产生进位或借位，则不溢出。

**【例 1-10】**当字长为 8 位时，计算−64D+64D。

**解**

$$\begin{array}{rr} -64D & [-64D]_{补}=1100\ 0000 \\ +\ \ 64D & +\ [64D]_{补}=0100\ 0000 \\ \hline 0 & 10000\ 0000 \end{array}$$

进位（丢失）

本例中，−64D+64D=0，运算结果在 8 位补码的表示范围（−128D～+127D）之内，不会溢出。通过分析运算过程可知，次高位（最高数值位）和最高位（符号位）都产生了进位。这种情况下，运算过程中产生的进位，自然丢失。根据前述的溢出判别规则，可知运算结果不溢出。

**【例 1-11】**当字长为 8 位时，计算 127D+1D。

**解**

$$\begin{array}{rr} 127D & [127D]_{补}=0111\ 1111 \\ +\ \ 1D & +\ [1D]_{补}=0000\ 0001 \\ \hline 128D & 1000\ 0000 \end{array}$$

溢出

本例中，127D+1D=128D，运算结果超出 8 位补码的表示范围：−128D～+127D，所以溢出。通过分析运算过程可知，次高位（最高数值位）产生了进位，而最高位（符号位）没有产生进位。这就使得和的数值部分占用了符号位，运算结果的符号与两个加数的符号相异，即两正数相加得到了一个负数和。根据前述的溢出判定方法，可知运算结果溢出。

# 1.3 BCD 码的表示与运算

## 1.3.1 BCD 码的编码方法

在计算机内部采用二进制形式表示数，但人们习惯使用十进制数。BCD 码，是二进制编码的十进制数的简称，是为了便于人机交往而设计的一种数字编码。BCD 码的编码规则是：用 4 位二进制数字表示 1 位十进制数字。在十进制数码与 4 位二进制数码表示的数之间选择不同的对应规律，就可以得到不同形式的编码。

常用的 BCD 码有：8421BCD 码、余 3 码、格雷码等。由于 80x86 微机系统支持 8421BCD 码的运算，故本书仅介绍 8421BCD 码。

**1．8421BCD 码的编码规则**

8421BCD 码的 4 位二进制数码的位权分别是：8、4、2 和 1，8421BCD 码的名称也就是由此而来的。

根据 8421BCD 码求其十进制数的过程是：将每位数码与对应的权相乘后求和，就可以得到它代表的十进制数。十进制数与 8421BCD 码的对应关系见表 1-7。

**表 1-7　十进制数与 8421BCD 码的对应关系表**

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

8421BCD 码有如下特点：

- 十进制数的每位表示法与该数的二进制形式一样，容易识别。
- 1010～1111 这 6 个编码没有用到，是无意义的编码。因此，当运算结果是这 6 个编码之一时，就需要经过调整后才能得到正确的结果。在 80x86 微机系统中，这种调整操作通过十进制调整指令实现。

**2．8421BCD 码的格式**

8421BCD 码有压缩 BCD 码和非压缩 BCD 码两种格式。

（1）压缩 8421BCD 码

压缩 8421BCD 码，也称组合 BCD 码，用 4 位二进制数表示 1 位十进制数。因此，一字节可以表示两位十进制数。例如，96D 的压缩 8421BCD 码是：1001 0110。

（2）非压缩 8421BCD 码

非压缩 8421BCD，也称非组合 BCD 码，用一字节的低 4 位表示 1 位十进制数，高 4 位任意，通常设为 0000。例如，96D 的非压缩 8421BCD 码是：0000 1001 0000 0110。

### 1.3.2 8421BCD 码的加减运算

**1．压缩 8421BCD 码的加减运算**

压缩 8421BCD 码进行加减运算时，参与运算的操作数为压缩 8421BCD 码，结果也是压缩 8421BCD 码。

下面举例说明压缩 8421BCD 码的加减运算及十进制调整方法。

**【例 1-12】**用压缩 8421BCD 码计算：16D+18D。

**解** 第一步：求 8421BCD 码，做二进制加法。

16D=00010110BCD，18D=00011000BCD。运算过程如下：

$$\begin{array}{rll} & 0001\ 0110 & \longrightarrow\ 16D \\ + & 0001\ 1000 & \longrightarrow\ 18D \\ \hline & 0010\ 1110 & \end{array}$$

第二步：分析上式运算结果，进行十进制调整。

我们知道，16D+18D=34D，用压缩 8421BCD 码进行加法运算，结果应为 00110100BCD。但上式运算的结果是 0010 1110。经分析，运算结果的低 4 位是：1110，该编码不是有效的压缩 8421BCD 码。造成这个问题的原因是，采用压缩 8421BCD 码运算时，运算器仍然进行的是二进制数的运算，采用的进位规则不是十进制运算规定的“逢十进一”规则，而是二进制运算的“逢二进一”规则。对应到本例中，就是个位上 6 和 8 相加，结果是 14，大于 9 了，应该向十位有一个进位（逢十进一），但是实际这一进位并没有产生。

解决的办法是：将出错的那一位压缩 8421BCD 码与 6 相加。相加后如果产生进位，则该进位应该加到压缩 8421BCD 码的高位。

$$\begin{array}{rll} & 0010\ 1110 & \\ + & 0000\ 0110 & \longrightarrow\ \text{个位加6调整} \\ \hline & 0011\ 0100 & \longrightarrow\ \text{和=34，正确结果} \end{array}$$

压缩 8421BCD 运算的十进制调整规则如下：

（1）加法运算后的十进制调整规则

- 若加法和的个位大于9或向十位有进位，则需要“加06H调整”，即所得和要加上00000110B；
- 若加法和的十位大于9或向百位有进位，则需要“加60H调整”，即所得和要加上01100000B。

（2）减法运算后的十进制调整规则

- 若减法差的个位大于 9 或向十位有借位，则需要“减 06H 调整”，即所得差要减去 00000110B；
- 若减法差的十位大于 9 或向百位有进位，则需要“减 60H 调整”，即所得差要减去 01100000B。

**【例 1-13】**用压缩 8421BCD 码计算：39D+98D。

**解** 39D=00111001BCD，98D=10011000BCD。

```
   0011 1001  ⟶  39D
+  1001 1000  ⟶  98D
-------------
   1101 0001  ⟶  因为个位向十位有进位，所以需加06H调整
+  0000 0110
-------------
   1101 0111  ⟶  因为十位为1101，它是无效编码，所以需加60H调整
+  0110 0000
-------------
1  0011 0111  ⟶  和=37D，进位=1，结果正确
```

**【例 1-14】**用压缩 8421BCD 码计算：35D−16D。

**解** 35D=00110101BCD，16D=00010110BCD。

```
   0011 0101  ⟶  35D
−  0001 0110  ⟶  16D
-------------
   0001 1111  ⟶  因为低位向高位有借位，所以需减06H调整
−  0000 0110
-------------
   0001 1001  ⟶  差=19D，结果正确
```

### 2．非压缩 BCD 码的加减运算

进行非压缩 BCD 码的加减运算时，参与运算的操作数为非压缩 BCD 码，结果也是非压缩 BCD 码。

如果操作数为两位十进制数，则用两字节表示该十进制数。例如，16D+17D=33D 的运算过程应该描述成

00000001 00000110BCD+00000001 00000111BCD=00000011 00000011BCD

假设任一非压缩 BCD 码表示为：$d_7d_6d_5d_4d_3d_2d_1d_0$，则进行加减运算时，若和/差大于 9 或 $d_3$ 位向 $d_4$ 位有进位/借位，就需要进行十进制调整操作。

下面举例说明非压缩 8421BCD 码的加减运算及十进制调整方法。

**【例 1-15】**将用非压缩 BCD 码表示的两个十进制数 8 和 7 相加。

**解**

```
   d7d6···d1d0
   0000 1000  ⟶  8
+  0000 0111  ⟶  7
-------------
   0000 1111  ⟶  结果错，因为1111＞9，所以需要加06H调整
+  0000 0110
-------------
   0001 0101  ⟶  因为d3向d4有进位，所以需要进行扩展调整
       ↓
   0000 0001 0001 0101  ⟶  将进位扩展到高8位
∧  1111 1111 0000 1111  ⟶  将d4～d0位清零
-------------------
   0000 0001 0000 0101  ⟶  和=0105BCD，结果正确
```

**【例 1-16】**将用非压缩 BCD 码表示的两个十进制数 9 和 8 相加。

解

$d_7d_6\cdots d_1d_0$

0000 1001 ⟶ 9

\+ 0000 1000 ⟶ 8

0001 0001 ⟶ 结果错，因为$d_3$向$d_4$有进位，所以需要加06H调整

\+ 0000 0110

0001 0111 ⟶ 因为$d_3$向$d_4$有进位，所以需要进行扩展调整

↓

0000 0001 0001 0111 ⟶ 将进位扩展到高8位

^ 1111 1111 0000 1111 ⟶ 将$d_4$～$d_0$位清零

0000 0001 0000 0101 ⟶ 和=0107，结果正确

# 1.4 字符的表示

在计算机中处理的信息并不全是数，还有字符或字符串。例如，姓名、编号等信息。因此，计算机必须能表示和处理字符。这些字符包括：大小写英文字母、数字字符、加减运算等专用符号、回车及换行等非打印字符等。这些字符必须用二进制形式表示后，才能在计算机中进行处理。

目前计算机中普遍采用美国标准信息交换代码——ASCII 码（American Standard Code for Information Interchange）表示一个字符。一个字符的 ASCII 码用一字节表示，其中低 7 位是字符的 ASCII 值，最高位常用作校验位或用于 ASCII 码的扩充。表 1-8 列出了用十六进制数表示的部分常用字符的 ASCII 值。

表 1-8 常用字符 ASCII 值（用十六进制表示）

| 字符 | ASCII 值 | 字符 | ASCII 值 | 字符 | ASCII 值 | 字符 | ASCII 值 |
|---|---|---|---|---|---|---|---|
| NUL | 00 | 4 | 34 | M | 4D | f | 66 |
| BS | 08 | 5 | 35 | N | 4E | g | 67 |
| LF | 0A | 6 | 36 | O | 4F | h | 68 |
| CR | 0D | 7 | 37 | P | 50 | i | 69 |
| ESC | 1B | 8 | 38 | Q | 51 | j | 6A |
| SP | 20 | 9 | 39 | R | 52 | k | 6B |
| ! | 21 | : | 3A | S | 53 | l | 6C |
| “ | 22 | ; | 3B | T | 54 | m | 6D |
| # | 23 | < | 3C | U | 55 | n | 6E |
| $ | 24 | = | 3D | V | 56 | o | 6F |
| % | 25 | > | 3E | W | 57 | p | 70 |
| & | 26 | ? | 3F | X | 58 | q | 71 |
| ‘ | 27 | @ | 40 | Y | 59 | r | 72 |
| ( | 28 | A | 41 | Z | 5A | s | 73 |
| ) | 29 | B | 42 | [ | 5B | t | 74 |
| * | 2A | C | 43 | \ | 5C | u | 75 |
| + | 2B | D | 44 | ] | 5D | v | 76 |
| , | 2C | E | 45 | ^ | 5E | w | 77 |
| - | 2D | F | 46 | _ | 5F | x | 78 |
| . | 2E | G | 47 | ` | 60 | y | 79 |
| / | 2F | H | 48 | a | 61 | z | 7A |
| 0 | 30 | I | 49 | b | 62 | { | 7B |
| 1 | 31 | J | 4A | c | 63 | \| | 7C |
| 2 | 32 | K | 4B | d | 64 | } | 7D |
| 3 | 33 | L | 4C | e | 65 | ～ | 7E |

说明：NUL—空，BS—退格，LF—换行，CR—回车，ESC—退出，SP—空格。

基本 ASCII 码有 128 个值，能表示 32 个控制符、10 个数字、26 个大写英文字母、26 个小写英文字母及 34 个专用符号。

扩充后的 ASCII 码有 256 个值，包括基本的 ASCII 码值、扩充的 128 个字符和图形符号的 ASCII 码值。

字符的 ASCII 值可以看作字符的码值，如字符“A”的 ASCII 码值为 41H，“Z”的 ASCII 码值为 5AH。利用这个值的大小可以将字符排序。以后遇到的字符串大小比较，实际上就是比较 ASCII 码值的大小。

# 习　题　1

1．总结计算机中十进制数、二进制数、八进制数及十六进制数的书写形式，并举例说明。

2．123D、0AFH、77Q、1001110B 分别采用的是什么计数制？

3．字长为 8 位和 16 位二进制数的原码和补码能表示的整数的最大值和最小值分别是多少？

4．把下列十进制数分别转换为二进制数和十六进制数。

（1）125　　（2）255　　（3）72　　（4）5090

5．把下列二进制数分别转换为十进制数和十六进制数。

（1）1111 0000　　（2）1000 0000　　（3）1111 1111　　（4）0101 0101

6．把下列十六进制数分别转换为十进制数和二进制数。

（1）FF　　（2）ABCD　　（3）123　　（4）FFFF

7．写出下列十进制数在字长为 8 位和 16 位两种情况下的原码和补码。

（1）16　　（2）−16　　（3）+0　　（4）−0

（5）127　　（6）−128　　（7）121　　（8）−9

8．实现下列转换。

（1）已知$[x]_{原}$=10111110，求$[x]_{补}$　　（2）已知$[x]_{补}$=11110011，求$[-x]_{补}$

（3）已知$[x]_{补}$=10111110，求$[x]_{原}$　　（4）已知$[x]_{补}$=10111110，求$[x]_{反}$

9．已知数 $A$ 和 $B$ 的二进制格式分别是 01101010 和 10001100，试根据下列不同条件，比较它们的大小。

（1）上述格式是 $A$、$B$ 两数的补码　　（2）$A$、$B$ 两数均为无符号数

10．下列各数均为十进制数，请用 8 位补码计算下列各题，并判断是否溢出；若无溢出，用十六进制形式表示运算结果。

（1）90 + 71　　（2）90−71　　（3）−90−71　　（4）−90 + 71　　（5）−90−(−71）

11．完成下列逻辑运算：

（1）11001100 ∧ 10101010　　（2）11001100 ∨ 10101010　　（3）11001100 ⊕ 10101010

（4）10101100 ∧10101100　　（5）10101100 ⊕ 10101100　　（6）10101100 ∨ 10101100

（7）$\overline{10101100}$

12．以下为十六进制数，试说明当把它们分别看作无符号数或字符的 ASCII 码值时，它们所表示的十进制数和字符是什么？

（1）30　　（2）39　　（3）42　　（4）62　　（5）20　　（6）7

13．以下为十进制数，分别写出其压缩 8421BCD 码和非压缩 8421BCD 码。

（1）49　　（2）123　　（3）7　　（4）62