第3章 信息编码与数据表示

计算机最基本的功能是进行数据的计算和处理，这些数据包括数值、文字、图形、图像、声音、视频等数据形式。由于计算机内部只能表示、识别、存储、处理和传送二进制数，所以各种数据信息都必须经过二进制数字化编码后，才能在计算机中进行处理。将各种不同类型的数据信息转换为二进制代码的过程称为信息编码。不同类型的信息具有不同的编码方式。

3.1 计算机中的数制

计算机中采用二进制数，为了书写和阅读的方便，引入了八进制数和十六进制数。而在日常生活及数学中，人们习惯使用十进制数。在人和计算机交换信息时，输入的数据或输出的结果是十进制数的，但在计算机存储、处理、传输信息时，二进制数工作得更好，这就需要在各种数值之间进行相互转换。虽然这种转换过程由计算机系统自动完成，但还是有必要了解各种数制的特点及其转换过程。

3.1.1 计算机中采用二进制数

现代计算机存储和处理的信息是以 0 和 1 的模式编码的，这些数字称为位(bit)。它们只是符号，其意义取决于正在处理的应用：有时用来表示数值，有时表示字母表中的字符，有时表示标点符号，有时表示图像，有时表示声音。

单个的位不是非常有用的，然而，当把位组合在一起，加上某种解释，即赋予了它们含义，就能够表示任何有限集合的元素。例如，一个二进制数字系统能够用来表示数值。通过使用标准的字符编码，能够对字母表中的字符和标点符号进行编码。

3.1.2 常用数制及其特点

按进位的原则进行计数称为进位计数制，简称“数制”。在计算机中常用的数制有二进制数、八进制数、十进制数、十六进制数。下面介绍数制的基本概念及常用数制的特点。

1. 数码

每一种数制都使用一组固定的数学符号来表示数字的大小，该数学符号称为“数码”。

不同进制数所使用的数码及数码的个数是不同的。例如，二进制的数码有 2 个：0 和 1。

2. 基数

不同进制数所使用的数码的个数，称为该数制的基数。一般来说，基数是几就是几进制。进制的规则是“逢几进一”。例如，十进制就是“逢十进一”，二进制就是“逢二进一”。表 3-1 展示了常用数制的基本要素及其表示方法。

表 3-1　常用数制的基本要素及其表示方法

数　制	基数	进制规则	位权	数　　码	表示
二进制	2	逢二进一	2^i	0、1	B
八进制	8	逢八进一	8^i	0,1,2,3,4,5,6,7	O
十进制	10	逢十进一	10^i	0,1,2,3,4,5,6,7,8,9	D
十六进制	16	逢十六进一	16^i	0,1,…,9,A,B,C,D,E,F	H

其中，十六进制数中：A 表示 10，B 表示 11，C 表示 12，D 表示 13，E 表示 14，F 表示 15。

3. 位权

在一个数中，每个数码表示的值不仅取决于数码本身，还与它所处的位置有关。例如，十进制数 123.45，可以表示成如下形式：

$$123.45=1\times10^2+2\times10^1+3\times10^0+4\times10^{-1}+5\times10^{-2}$$

式中 10^2、10^1、10^0、10^{-1}、10^{-2} 称为各位数字的权。可以看出，各位数字只有乘上它们的权值，才是它的实际值，如上例中最左边的数字 1，乘上 10^2，才是它的实际值 100。上式称为十进制数 123.45 的按权展开式。十进制数的权值都是 10 的若干次幂，二进制数的权值都是 2 的若干次幂。

一般地，可以将一个数按照“基数”和“权”展开为如下形式：

$$D=\int_{i=1}^{n} N_i K^{i-1}+\int_{j=-1}^{-m} N_j K^j$$

其中，N_i 和 N_j 表示第 i 位和小数点后第 j 位上的数码；K^{i-1} 和 K^j 表示该数码的权，K 是基数。上式中的前一项表示数的整数部分，后一项表示数的小数部分，它们分别有 n 位和 m 位。

任何一种进制中的任何一个数都可以写成按权展开的形式。例如，按照以上展开式，十进制数 87654.32 可以表示为：

$$87654.32=8\times10^4+7\times10^3+6\times10^2+5\times10^1+4\times10^0+3\times10^{-1}+2\times10^{-2}$$

二进制数 1011.1 可以表示为：

$$(1011.1)_2=1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}$$

八进制数 357.4 可以表示为：

$$(357.4)_8=3\times8^2+5\times8^1+7\times8^0+4\times8^{-1}$$

十六进制数 A8F 可以表示为：

$$(A8F)_{16}=10\times16^2+8\times16^1+15\times16^0$$

为了对不同进制的数进行区分，书写时，在数字后面加 B(Binary)表示二进制数，加 O(Octal)表示八进制数，加 H(Hexadecimal)表示十六进制数，十进制数则可用后缀 D(Decimal)表示或者不加。也可在数字后面加下标来表示相应的进制数。例如，1011B 或 $(1011)_2$ 表示 1011 是二进制数。

3.1.3 不同数制之间的转换

1. 将非十进制数转换成十进制数

将非十进制数转换成十进制数的方法是把非十进制数按位权重展开并求和。

【例 3-1】 将二进制数 10110101 转换成十进制数。

解： $(10110101)_2$

$=1\times2^7+0\times2^6+1\times2^5+1\times2^4+0\times2^3+1\times2^2+0\times2^1+1\times2^0$

$=128+32+16+4+1$

$=(181)_{10}$

即二进制数 10110101 转换成十进制数为 181。

【例 3-2】 将八进制数 123 转换成十进制数。

解： $(123)_8=1\times8^2+2\times8^1+3\times8^0=64+16+3=(83)_{10}$

即八进制数 123 转换成十进制数为 83。

【例 3-3】 将十六进制数 15C 转换成十进制数。

解： $(15C)_{16}=1\times16^2+5\times16^1+12\times16^0=256+80+12=(348)_{10}$

即十六进制数 15C 转换成十进制数为 348。

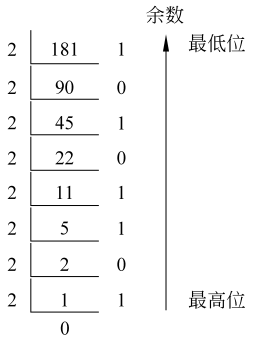
2. 将十进制数转换成二进制数

整数部分和小数部分的转换方法是不同的，下面分别介绍。

(1) 整数部分的转换。将一个十进制整数转换为二进制整数，采用的方法是“除 2 取余”法，即对一个十进制整数反复进行除以 2 和保留余数的操作，直至商为 0，然后将所得到的余数自下而上排列即可。

【例 3-4】 将十进制数 181 转换成二进制数。

解：

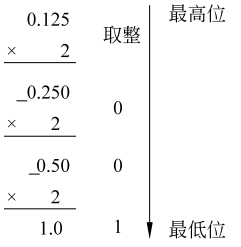


结果：$(181)_{10}=(10110101)_2$

（2）小数部分的转换。将一个十进制小数转换为二进制小数，采用的方法是“乘 2 取整”法，即将一个十进制小数不断地乘以 2，直到小数的当前值为 0 或满足所要求的精度为止。每乘一次取一次整数，最后将所得到的乘积的整数部分自上而下排列即可。

【例 3-5】 将十进制小数 0.125 转换成二进制小数。

解：

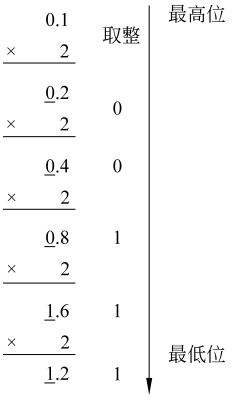


结果：$(0.125)_{10}=(0.001)_2$

通常一个非十进制小数能够完全准确地转换成十进制小数，但一个十进制小数并不一定能完全准确地转换成非十进制小数。例如，十进制小数 0.1 就不能完全准确的转换成二进制小数。在这种情况下，可以根据精度要求只转换到小数点后某一位为止，这个数就是该小数的近似值。

【例 3-6】 将十进制小数 0.1 转换成二进制小数，精确到 5 位小数。

解：



结果：$(0.1)_{10}\approx(0.00111)_2$

在进行转换时，如果一个数既有整数部分，又有小数部分，应分别进行转换，然后再组合起来。

【例 3-7】 将十进制数 181.1 转换成二进制数，精确到 5 位小数。

解： $(181)_{10}=(10110101)_2$

$(0.1)_{10}\approx(0.00111)_2$

$(181.1)_{10}\approx(10110101.00111)_2$

3. 二进制数与八进制数、十六进制数之间的转换

二进制数适合计算机内部数据的表示和运算，但书写起来位数比较长，如表示一个十进制数 1024，写成等值的二进制数就需要 11 位，很不方便。而八进制数和十六进制数比等值的二进制数的长度短得多，而且它们之间的转换也非常方便。因此，在书写程序和数据时，在用到二进制数的地方，往往采用八进制数或十六进制数的形式。

由于二进制数、八进制数和十六进制数之间存在特殊的关系，即 $8^1=2^3$，$16^1=2^4$，因此转换方法相对简单。几种常用数制之间的对应关系如表 3-2 所示。

表 3-2　几种常用数制之间的对应关系

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

（1）八进制数与二进制数之间的转换。因为 $8^1=2^3$，所以每位八进制数都可以用 3 位二进制数表示，也可以说 3 位二进制数可以表示 1 位八进制数。根据这种对应关系，将二进制数转换成八进制数时，只需以小数点为界，分别向左、向右，每 3 位二进制数分为一组，最后不足 3 位时用 0 补足 3 位（整数部分在高位补 0，小数部分在低位补 0）；然后将每组转换成相应的八进制数，即可完成转换。

【例 3-8】 将二进制数 10110101.0100101 转换成八进制数。

解：将每 3 位二进制数转换为 1 位的八进制数即可。

$$(010\quad 110\quad 101\,.\,010\quad 010\quad 100)_2$$

$$(2\quad\ 6\quad\ 5\,.\,2\quad\ 2\quad\ 4)_8$$

结果：$(10110101.0100101)_2=(265.224)_8$

【例 3-9】 将八进制数 725.16 转换成二进制数。

解：将每位八进制数转换为 3 位的二进制数即可。

$$(7\quad 2\quad 5.\quad 1\quad 6)_8$$
$$(111\quad 010\quad 101.001\quad 110)_2$$

结果：$(725.16)_8=(111010101.00111)_2$

(2) 十六进制数与二进制数之间的转换。因为 $16^1=2^4$，所以每位十六进制数都可以用 4 位二进制数表示，也可以说 4 位二进制数可以表示 1 位十六进制数。根据这种对应关系，将二进制数转换成十六进制数时，只需以小数点为界，分别向左、向右，每 4 位二进制数分为一组，最后不足 4 位时用 0 补足 4 位(整数部分在高位补 0，小数部分在低位补 0)，然后将每组转换成相应的十六进制数，即可完成转换。

【例 3-10】 将二进制数 1110110101.0100101 转换成十六进制数。

解：将每 4 位二进制数转换为 1 位的十六进制数即可。

$$(0011\quad 1011\quad 0101.0100\quad 1010)_2$$
$$(3\quad B\quad 5.\quad 4\quad A)_{16}$$

结果：$(1110110101.0100101)_2=(3B5.4A)_{16}$

【例 3-11】 将十六进制数 7C5.E6 转换成二进制数。

解：将每位十六进制数转换为 4 位的二进制数即可。

$$(7\quad C\quad 5.\quad E\quad 6)_8$$
$$(0111\quad 1100\quad 0101.1110\quad 0110)_2$$

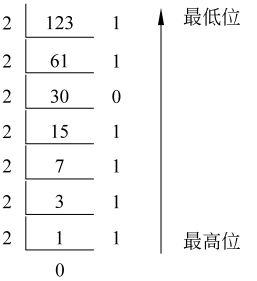
结果：$(7C5.E6)_{16}=(11111000101.1110011)_2$

(3) 将十进制数转换成八、十六进制数。若要将十进制数转换为八进制数或十六进制数，一般借助于二进制数，即先将十进制数转换成二进制数，再将此二进制数转换为八进制数或十六进制数。

【例 3-12】 将十进制数 123 转换成十六进制数。

解：具体步骤如下。

第一步：先将十进制数 123 转换为二进制数：



结果：$(123)_{10}=(1111011)_2$

第二步：再将二进制数 1111011 转换为十六进制数，即：

$$(1111011)_2=(0111\quad 1011)_2=(7B)_{16}$$

结果：$(123)_{10}=(1111011)_2=(7B)_{16}$

将十进制数转换成八进制数的情况与将十进制数转换成十六进制数相似，不再赘述。

3.1.4 二进制的常用单位

1. 位

位(bit)是度量信息的单位，也是表示信息量的最小单位，只有 0、1 两种二进制状态。

2. 字节

字节(Byte)是计算机中数据处理和存储容量的基本单位，1 个 Byte 由 8 个 bit 组成，能够容纳一个英文字符，一个汉字需要两个字节的存储空间。

计算机常用的存储单位及其关系如下：

1Byte(字节)=8bit(比特)

1KB(KiloByte 千字节)=1024B(字节)

1MB(MegaByte 兆字节)=1024KB

1GB(GigaByte 吉字节)=1024MB

1TB(TeraByte 太字节)=1024GB

1PB(PetaByte 拍字节)=1024TB

1EB(ExaByte 艾字节)=1024PB

1ZB(ZetaByte 泽字节)=1024EB

1YB(YottaByte 尧字节)=1024ZB

1BB(BrontoByte 珀字节)=1024YB

1NB(NonaByte 诺字节)=1024BB

1DB(DoggaByte 刀字节)=1024NB

3. 字

在计算机中占据一个单独的地址(内存单元的编号)并作为一个单元(由多个字节组合而成)处理的一组二进制数称为“字”(Word)，它由若干个位或字节所组成。在计算机的运算器、控制器中，数据或指令通常都是以字为单位进行传送的。字出现在不同的位置是有不同的含义的，对计算机的运算器和内存器来说，一个字或几个字是一个数据；对于控制器来说，一个字或几个字是一条指令。

4. 字长

一个字所包含的二进制位的数量称为字长，它反映了计算机处理信息的一种能力。字长是 CPU 性能的重要标志之一，字长越长，说明计算机一次能处理的数据量就越大。例如，8 位的 CPU 字长为 8 位，一个字等于一字节，一次只能处理一字节，而 32 位的 CPU 字长为 32 位，一个字等于 4 字节，一次能处理 4 字节。同理，字长为 64 位的 CPU 一次可以处理 8 字节，一个字等于 8 字节。

3.2 数值数据在计算机中的表示

数据是计算机处理的对象。从不同的处理角度来看，数据有不同的表现形态。从外部形式的角度来看，计算机可处理数值、文字、图、声音、视频及各种模拟信息，它们被称为感觉媒体。从算法描述的角度来看，有图、表、树、队列、矩阵等结构类型的数据。从高级语言程序员的角度来看，有数组、结构、指针、实数、整数、布尔型、字符和字符串等类型的数据。不管以什么形态出现，在计算机内部，数据最终都是由机器指令来处理。而从机器指令的角度来看，数据只有整数、浮点数和位串这几类简单的基本数据类型。其中，整数和浮点数统称数值数据。

数值数据可用来表示数量的多少，可比较其大小，分为整数和实数，整数又分为无符号整数和带符号整数。在计算机内部，整数用定点数表示，实数用浮点数表示。非数值数据就是一个没有大小之分的位串，不表示数量的多少，主要用来表示字符数据和逻辑数据。

由于计算机采用二进制，所以一切信息都要由 0 和 1 两个数字的组合，即二进制数字化编码来表示。

3.2.1 机器数和真值

日常生活中，常使用带正负号的十进制数表示数值数据。在计算机中，对带符号的数的正号和负号，必须用“0”和“1”进行编码。通常把一个数的最高位定义为符号位，用 0 表示正，用 1 表示负，称为数符，其余位表示数值。把在计算机内存放的正、负号数码化的数，称为机器数，而把机器外部由正、负号表示的数，称为真值。

真值一般用十进制表示。例如，真值为＋7，8 位二进制数为 00000111；真值为－7，8 位二进制数为 10000111。

对于无符号数，由于不涉及符号问题，所以在计算机中用一个数的全部有效位来表示数的大小。例如，真值为无符号整数 123，8 位二进制数为 01111011。

3.2.2 原码、反码和补码

带符号数的数值和符号都用二进制数码来表示，那么计算机对数据进行运算时，符号位应如何处理？是否也同数值位一起参加运算？为了妥善地处理这个问题，就产生了把符号位和数值位一起进行编码的各种方法，这就是原码、反码和补码。

1. 原码

正数的符号位用“0”表示，负数的符号位用“1”表示，数值部分用真值的绝对值来表示的二进制数，称为原码。用$[X]_{原}$表示，设 X 为整数。例如：

$$X_1 = +66 = +1000010 \qquad [X_1]_{原} = 0\ 1000010$$
$$X_2 = -66 = -1000010 \qquad [X_2]_{原} = 1\ 1000010$$

原码的特点如下：

① 用原码表示数简单、直观，与真值之间转换方便。

② 0 的原码表示不唯一：$[+0]_{原} = 0\ 0000000$，$[-0]_{原} = 1\ 0000000$。

③ 因为原码的最高位表示符号位，所以 8 位二进制原码表示的整数范围是：$-127 \sim +127$。

④ 加、减法运算复杂。不能用原码直接对两个同号数相减或两个异号数相加，必须先判断数的正负，再决定使用加法还是减法，才能进行具体的计算，因而使机器的结构相应地复杂化或增加机器的运算时间。例如，将十进制数“+23”的原码与十进制数“−36”的原码直接相加：$[+23]_{原} + [-36]_{原} = 0\ 0010111 + 1\ 0100100 = 1\ 0111011$。其结果是，符号位为“1”表示负数，数值部分为“0111011”，是十进制数“59”，所以计算结果为“−59”，这显然是错误的。现代计算机中不用原码来表示整数，只用定点原码小数来表示浮点数的尾数部分。

因此，为运算方便，在计算机中通常将减法运算转换为加法运算，由此引入了反码和补码。

2. 反码

反码是为了解决负数加法运算问题，将减法运算转换为加法运算，从而简化运算规则。反码表示法规定：正数的反码与其原码相同；负数的反码符号位为“1”，数值位为其原码数值位按位取反，即“1”都换成“0”，“0”都换成“1”。例如：

$$X_1 = +66 = +1000010 \qquad [X_1]_{反} = 0\ 1000010$$
$$X_2 = -66 = -1000010 \qquad [X_2]_{反} = 1\ 0111101$$

反码的特点如下：

① 0 的反码表示不唯一：$[+0]_{反} = 0\ 0000000$，$[-0]_{反} = 1\ 1111111$。

② 8 位二进制反码的取值范围是：$-127 \sim +127$。

反码在计算机中很少被使用，有时用作数码变换的中间表示形式或用于数据效验。

3. 补码

在人们的计算概念中，“0”是没有正负之分的，于是就引入了补码概念。

数的补码与“模”有关。“模”是指一个计数系统的计数量程或一个计量器的容量。任何有“模”的计量器，均可化减法运算为加法运算。例如，时钟的模为 12，若当前时钟指向 10 点，而准确时间为 6 点，这时可以使用两种方法来调整时钟时间：一是逆时针拨时针 4 小时，即 $10-4=6$；二是顺时针拨时针 8 小时，即 $10+8=12+6\equiv 6 \pmod{12}$，仍为 6 点。可见，在以 12 为模的系统中，加 8 和减 4 的效果是一样的。因此，可以说 -4 的补码为 8，或者说 -4 和 $+8$ 对模 12 来说互为补码。

计算机中的存储、运算和传送部件都只有有限位，因此计算机表示的机器数的位数也只有有限位。两个 n 位二进制数在进行运算过程中，可能会产生一个多于 n 位的结果，此时，计算机会舍弃高位而保留低 n 位，这样做可能会产生两种结果。

① 剩下的低 n 位数不能正确表示运算结果，即丢掉的高位是运算结果的一部分。例如，在两个同号数相加时，当相加得到的和超出了 n 位数可表示的范围时出现这种情况，称此时发生了溢出现象。

② 剩下的低 n 位数能正确表示运算结果，即高位的舍去并不影响其运算结果。在两个同号数相减或两个异号数相加时，运算结果就是这种情况。舍去高位的操作相当于“将一个多于 n 位的数去除以 2^n，保留其余数作为结果”的操作，也就是“模运算”操作。

补码表示法规定：对于正数，其补码与原码相同；对于负数，其补码为其反码加 1，即 $[X]_{补}=[X]_{反}+1$。例如：

$$X_1=+66=+1000010 \qquad [X_1]_{补}=0\ 1000010$$

$$X_2=-66=-1000010 \qquad [X_2]_{补}=1\ 0111110$$

补码的特点如下：

① 0 的补码表示唯一：$[+0]_{补}=0\ 0000000$，$[-0]_{补}=0\ 0000000$。

② 8 位二进制补码的取值范围是：$-128\sim+127$。

③ 加、减法运算方便。当负数用补码表示时，可以把减法运算转化为加法运算。

④ 由补码求真值：补码最高位为“1”表示真值为负数，真值的绝对值为补码数值位“按位求反加 1 的和”。

3.2.3 数的定点表示与浮点表示

日常生活中所使用的数有整数和实数之分，整数的小数点固定在数的最右边，可以省略不写，而实数的小数点则不固定。计算机中只能表示 0 和 1，无法表示小数点，因此，要使计算机能够处理日常使用的数值数据，必须解决小数点的表示问题。通常计算机中通过约定小数点的位置来实现。小数点位置约定在固定位置的数，称为定点数；小数点位置约定为可浮动的数，称为浮点数。

1. 定点表示法

定点表示法用来对定点小数和定点整数进行表示。对于定点小数，其小数点总是固定在数的左边，一般用来表示浮点数的尾数部分。对于定点整数，其小数点总是固定在数的最右边，因此可用“定点整数”来表示整数 。

2. 浮点表示法

对于任意一个实数 X，可以表示成如下形式：

$$X=(-1)^S\times M\times R^E$$

其中，S 取值为 0 或 1，用来决定数 X 的符号；M 是一个二进制定点小数，称为数 X 的尾数；E 是一个二进制定点整数，称为数 X 的阶或指数；R 是基数，可以取值为 2、4、16 等。在基数 R 一定的情况下，尾数 M 的位数反映数 X 的有效位数，它决定了数的表示精度，有效位数越多，表示精度就越高；阶 E 的位数决定数 X 的表示范围；阶 E 的值确定了小数点的位置。

假定浮点数的尾数是纯小数，那么，从浮点数的形式来看，绝对值最小的非零数形如

$0.0\cdots01\times R^{-11\cdots1}$，绝对值最大的数形如 $0.11\cdots1\times R^{11\cdots1}$。所以，假设 m 和 n 分别表示阶和尾数的位数，基数是 2，则浮点数 X 的绝对值的范围是：

$$2^{-(2^m-1)}\times 2^{-n}\leqslant|X|\leqslant(1-2^{-n})\times 2^{(2^m-1)}$$

上述公式中，紧靠 $|X|$ 左右两边的两个因子就是非零定点小数的绝对值表示范围，浮点数的最小数是定点小数的最小数 2^{-n} 除以一个很大的数 $2^{(2^m-1)}$，而浮点数的最大数则是定点小数最大数 $(1-2^{-n})$ 乘以这个大数 $2^{(2^m-1)}$，由此可见，浮点数的表示范围比定点数的表示范围要大得多。

3.3 二进制数的运算

二进制数是有自己的运算规则，以下简单介绍其中的算术运算和逻辑运算的规则。

3.3.1 二进制数的算术运算

1. 二进制数的加法

二进制数的加法运算法则是：

$0+0=0$　　$0+1=1$　　$1+0=1$　　$1+1=0$（向高位进位）

【例 3-13】 求 $(1101)_2+(0111)_2=?$

解：

被加数：　1101
加数：+　0111
结果：　10100

结果：$(1101)_2+(0111)_2=(10100)_2$

2. 二进制数的减法

二进制数的减法运算法则是：

$0-0=0$　　$1-0=1$　　$1-1=0$　　$0-1=1$（向高位借位）

【例 3-14】 求 $(1101)_2-(0111)_2=?$

解：

被减数：　1101
减数：−　0111
结果：　0110

结果：$(1101)_2-(0111)_2=(0110)_2$

3. 二进制数的乘法

二进制数的乘法运算法则是：

$0\times0=0$　　$0\times1=1\times0=0$　　$1\times1=1$

【例 3-15】 求$(1101)_2\times(101)_2=?$

解：

$$
\begin{array}{rr}
\text{被乘数：} & 1101 \\
\text{乘数：}\times & 101 \\
\hline
 & 1101 \\
 & 0000 \\
 & 1101 \\
\hline
\text{结果：} & 1000001
\end{array}
$$

结果：$(1101)_2\times(101)_2=(1000001)_2$

4. 二进制数的除法

二进制数的除法运算法则是：

$$0\div1=0 \qquad 1\div1=1$$

【例 3-16】 求$(1001110)_2\div(110)_2=?$

解：

$$
\begin{array}{r}
1101 \\
\hline
110\overline{)1001110} \\
110 \\
\hline
111 \\
110 \\
\hline
110 \\
110 \\
\hline
0
\end{array}
$$

结果：$(1001110)_2\div(110)_2=(1101)_2$

3.3.2 二进制数的逻辑运算

逻辑数据只有两个值：“真”与“假”、“是”与“否”、“条件成立”与“条件不成立”，在计算机中用二进制的“1”与“0”表示。

逻辑运算与算术运算不同，算术运算是将一个二进制数的所有位综合为一个数值整体，低位的运算结果会影响到高位（如进位等），而逻辑运算是按位进行运算，故逻辑运算没有进位或借位。逻辑运算的结果并不表示数值大小，而是表示一种逻辑概念，若成立用真或 1 表示，若不成立用假或 0 表示。常用的逻辑运算有“与”运算、“或”运算、“非”运算和“异或”运算。

1. “与”运算

做一件事情取决于多种因素，只有当所有条件都成立时才去做，否则就不做，这种因果关系称为逻辑“与”。“与”运算通常用符号“∧”“∩”“AND”表示。

“与”运算规则：

$$0\wedge0=0 \qquad 0\wedge1=0 \qquad 1\wedge0=0 \qquad 1\wedge1=1$$

即两个参与运算的数，若有一个数为0，则运算结果为0；若都为1，则运算结果为1。

【例 3-17】 求 01101011∧11001110＝？

$$\begin{array}{rr} & 01101011 \\ \wedge & 11001110 \\ \hline & 01001010 \end{array}$$

结果：01101011∧11001110＝01001010

2. “或”运算

做一件事情取决于多种因素，只要其中有一个因素满足就去做，这种因果关系称为逻辑“或”。“或”运算通常用符号“∨”“∪”OR来表示。

“或”运算规则：

$$0 \vee 0 = 0 \qquad 0 \vee 1 = 1 \qquad 1 \vee 0 = 1 \qquad 1 \vee 1 = 1$$

即两个参与运算的数，若有一个数为1，则运算结果为1；若都为0，则运算结果为0。

【例 3-18】 求 01101011∨11001110＝？

$$\begin{array}{rr} & 01101011 \\ \vee & 11001110 \\ \hline & 11101111 \end{array}$$

结果：01101011∨11001110＝11101111

3. “非”运算

逻辑“非”实现逻辑否定，即求“反”运算，“真”变“假”、“假”变“真”。表示逻辑“非”常在逻辑变量上面加一横线，如“非”A 写 $\overline{A}$。

“非”运算规则：

$$\overline{1} = 0 \qquad \overline{0} = 1$$

【例 3-19】 已知 A＝01101011，求 $\overline{A}$。

解： $\overline{A}$＝10010100

4. “异或”运算

当参与运算的两个数的值相同时，逻辑“异或”的结果是“假”；当参与运算的两个数的值不同时，逻辑“异或”的结果是“真”。“异或”运算通常用符号“⊕”表示。

“异或”运算规则：

$$0 \oplus 0 = 0 \qquad 1 \oplus 1 = 0 \qquad 0 \oplus 1 = 1 \qquad 1 \oplus 0 = 1$$

【例 3-20】 求 01101011⊕11001110＝？

$$\begin{array}{rr} & 01101011 \\ \oplus & 11001110 \\ \hline & 10100101 \end{array}$$

结果：01101011⊕11001110＝10100101

3.4 常用的信息编码

所谓信息编码，就是采用少量的基本符号（数码）和一定的组合规则来区别和表示信息。计算机是以二进制方式组织、存放信息的，所以现实世界中的各种数据信息如果要在计算机中进行运算和处理，都必须用二进制数码 0 和 1 的不同组合，即二进制编码表示。下面介绍几种常用的信息编码方式。

3.4.1 BCD 码

BCD 码是以二进制编码表示的十进制数，其编码方式是用 4 位二进制数表示 1 位十进制数。而 4 位二进制数可以组合成 16 种状态，去掉 10 种状态后还有 6 种冗余状态，所以从 16 种状态中选取 10 种状态表示十进制数的方法很多，因而存在多种 BCD 码方案。

1. 有权 BCD 码

有权 BCD 码是指表示每个十进制数位的 4 个二进制数位（称为基 2 码）都有一个确定的权。最常用的一种编码就是 8421 码，它选取 4 位二进制数按计数顺序的前 10 个代码与十进制数字相对应，每位的权从左到右分别为 8、4、2、1，因此称为 8421 码，也称自然 BCD 码，记为 NBCD 码。

2. 无权 BCD 码

无权 BCD 码是指表示每个十进制数位的 4 个基 2 码没有确定的权。在无权码方案中，用得较多的是余 3 码和格雷（gray）码。

一个十进制数通常用多个对应的 BCD 码组合表示，每个数字对应 4 位二进制数，两个数字占一个字节，数符可用 1 位二进制数表示（1 表示负数，0 表示正数），或用 4 位二进制数表示，并放在数字串最后，通常用 1100 表示正号，用 1101 表示负号。例如，奔腾处理器中的十进制数占 80 位，第一个字节中的最高位为符号位，后面的 9 个字节可表示 18 位十进制数。

3.4.2 ASCII 码

美国信息交换标准代码，简称 ASCII 码，使用指定的 7 位或 8 位二进制数组合来表示 128 或 256 种可能的字符。标准 ASCII 码也叫基础 ASCII 码，使用 7 位二进制数（剩下的 1 位二进制数为 0）来表示所有的大写和小写字母、数字 0～9、标点符号，以及在美式英语中使用的特殊控制字符。

在标准 ASCII 码中，其最高位（b7）用作奇偶校验位。所谓奇偶校验，是指在代码传送过程中用来检验是否出现错误的一种方法，一般分为奇校验和偶校验两种。奇校验规定：正确的代码一个字节中 1 的个数必须是奇数，若非奇数，则在最高位 b7 添 1；偶校验

规定：正确的代码一个字节中 1 的个数必须是偶数，若非偶数，则在最高位 b7 添 1。通过对奇偶校验位设置“1”或“0”状态，保持 8 位字节中的“1”的个数总是奇数（称为奇校验）或偶数（称为偶校验），用以检测字符在传送（写入或读出）过程中是否出错。

后 128 个称为扩展 ASCII 码。许多基于 x86 的系统都支持使用扩展 ASCII 码。扩展 ASCII 码允许将每个字符的第 8 位用于确定附加的 128 个特殊符号字符、外来语字母和图形符号。

3.4.3 汉字编码

汉字编码是为汉字设计的一种便于输入计算机的代码。汉字种类繁多，编码比英文字符复杂，从汉字的输入、处理到输出，不同的阶段要采用不同的编码，包括汉字输入码、汉字内码、汉字字形码。汉字输入码比较容易学习和记忆，汉字内码是计算机内部对汉字的表示，要在显示器上显示或在打印机上打印出用户所输入的汉字，需要汉字字形码。用户用汉字输入码输入汉字，系统由汉字输入码找到相应的汉字内码，系统由汉字内码再找到相应的字形码。

1. 汉字输入码

汉字输入码所解决的问题是如何使用英文标准键盘把汉字输入到计算机内。其编码方案主要分从音编码和从形编码两大类。其他类型是相互结合型，或与字义结合，或与检字法结合，或与词组结合。因设计的目的、思想不同，用于编码的元素、所用码元的数量、取码方法和规则，避开同码字和占用键盘键位的方法等，都因设计者而异，因此产生了数百种汉字输入编码方案。

从音编码是以《汉语拼音方案》为基本编码元素，在汉语拼音键盘或经过处理的西文键盘上，根据汉字读音直接键入拼音。只要是掌握汉语拼音的人不需训练和记忆即可使用，但汉字同音字太多，输入重码率较高，因此按拼音输入后还必须进行同音字选择，这就会影响输入速度。

从形编码是以笔画和字根为编码元素，所有的汉字都由横、竖、撇、点、折、弯有限的几种笔画构成，并且又可分为“左右”“上下”“包围”“单体”有限的几种构架，每种笔画都赋予一个编码并规定选取字形构架的顺序，不同的汉字因组成的笔画和字形构架不同，就能获得一组不同的编码来表达一个特定的汉字，广泛使用的“五笔字形”就属于这一种。

除此之外，还有数字编码，利用一串数字表示一个汉字，电报码就属于这种。数字编码输入的优点是无重码，输入码与内部编码的转换比较方便；缺点是代码难记忆。

2. 汉字内码

同一个汉字以不同输入方式进入计算机时，编码长度以及 0、1 组合顺序差别很大，使得汉字信息进一步存取、使用、交流十分不方便，必须转换成长度一致且与汉字唯一对应的，能在各种计算机系统内通用的编码，满足这种规则的编码称为汉字内码。

汉字内码是用于汉字信息的存储、交换检索等操作的机内代码，一般采用两个字节表示。英文字符的机内代码是 7 位的 ASCII 码，当用一个字节表示时，最高位为“0”。为了

与英文字符区别，一个汉字的国标码占两个字节，因为西文字符和汉字都是字符，为了在计算机内部能够区分是汉字编码还是 ASCII 码，将汉字国标码两个字节的最高位均规定为“1”，变换后的国标码称为汉字机内码。由此可知，汉字机内码的每个字节都大于 128，而每个西文字符的 ASCII 码值均小于 128。有些系统中字节的最高位用于奇偶校验位或采用扩展 ASCII 码，这种情况下用 3 个字节表示汉字内码。

3. 汉字字形码

计算机内存储的汉字需要在屏幕上显示或在打印机上输出时，需要知道汉字的字形信息，汉字内码并不能直接反映汉字的字形，而要采用专门的字形码。

目前的汉字处理系统中，汉字字形码通常有两种表示方式：点阵和矢量表示方法。

用点阵表示字形时，是将字符的字形分解成若干“点”组成的点阵，将此点阵置于网状上，每一个小方格是点阵中的一个“点”，点阵中的每一个点可以有黑白两种颜色，有字形笔画的点用黑色，反之用白色，这样就能描写出汉字字形。如果用二进制的“1”表示黑色点，用“0”表示白色点，每一行 16 个点用两字节表示，则需 32 个字节描述一个汉字的字形。

一个计算机汉字处理系统常配有宋体、黑体、楷体等多种字体，同一个汉字不同字体的字形编码也是不相同的。

根据输出汉字的要求不同，点阵的多少也不同。简易型汉字为 16 * 16 点阵，提高型汉字为 24 * 24 点阵、32 * 32 点阵、48 * 48 点阵等。点阵规模越大，字型越清晰美观，所占存储空间也越大。

矢量表示方式存储的是描述汉字字形的轮廓特征，当要输出汉字时，通过计算机的计算，由汉字字形描述生成所需大小和形状的汉字点阵。矢量化字形描述与最终文字显示的大小，分辨率无关，因此可以产生高质量的汉字输出。Windows 中使用的 TrueType 技术就是汉字的矢量表示方式。

3.4.4 Unicode 码

Unicode 码是由多语言软件制造商组成的统一码联盟制定的一种国际通用字符编码标准，是为了解决传统的字符编码方案的局限而产生的。它为每种语言中的每个字符设定了统一并且唯一的二进制编码，以满足跨语言、跨平台进行文本转换、处理的要求。

Unicode 编码共有三种具体实现，分别为 UTF-8、UTF-16、UTF-32，其中 UTF-8 占用 1～4 字节、UTF-16 占用 2～4 字节、UTF-32 占用 4 字节。Unicode 是字符集，UTF 是一种编码方式，字符集和编码是不同的概念，但有时称呼上有些模糊，经常笼统的称这些 Unicode 字符集为 Unicode 编码。Unicode 字符集只规定了有哪些字符，而最终决定采用哪些字符，每一个字符用多少个字节表示等问题，则由编码来决定。

Unicode 码在全球范围的信息交换领域均有广泛的应用。

本章小结

1. 计算机中采用二进制数存储数据，为了书写和阅读方便，引入八进制数和十六进制数；在日常生活中，更习惯使用十进制数。

2. 每种数值都有自己的数码、基数和权重。

3. 将非十进制数转换成十进制数是按位权重展开并求和。

4. 将十进制数的整数部分转为二进制数采用的是“除 2 取余”法，将十进制数的小数部分转为二进制数采用的是“乘 2 取整”法。

5. 计算机中表示信息量的最小单位是位(bit)，最基本的单位是字节(Byte)。

6. 数值数据在计算机中有原码、补码和反码三种表示方式。

7. 小数点位置约定在固定位置的数，称为定点数；小数点位置约定为可浮动的数，称为浮点数。

8. 二进制数的算术运算有“加”“减”“乘”“除”四种，逻辑运算有“与”“或”“非”“异或”四种。

9. 常用的信息编码方式有：BCD 码、ASCII 码、汉字编码、Unicode 码等。

10. 汉字编码比较复杂，从汉字的输入、处理到输出，不同的阶段要采用不同的编码，包括汉字输入码、汉字内码、汉字字形码。

习　　题

1. 将下列十进制数转换为二进制数。

100　　21.75　　−3.1　　1.35　　255　　64

2. 将下列二进制数转换为十进制数。

11011　　101　　−10.011　　0.11　　−111.111

3. 将下列二进制数转换为八进制数和十六进制数。

$(10110011)_2$　　$(1101001.11)_2$

4. 将下列八进制数转换为二进制数。

$(1257)_8$　　$(425.354)_8$

5. 将下列十六进制数转换为二进制数。

$(AB3F)_{16}$　　$(2AB.4CD)_{16}$

6. 分别求下列真值的原码、反码和补码(码的长度为 8 位二进制数)。

+1111　　−1111　　−0　　+0　　+1010　　−1010

7. 十进制小数到二进制形式的转换是不精确的，用这一点能否否定在计算机中引入二进制的合适性？为什么？

8. 计算机中处理、存储、传输信息的最小单位和最基本的单位分别是什么？

9. 常用的信息编码方式有哪些？