第二章 数据信息表示

- 计算机内部的信息分为：控制信息和数据信息。
- 控制信息：指令
- 数据信息：
  - 数值数据：有确定的值，可表示大小（进位计数制、小数点、符号表示）。
  - 非数值数据：无确定的值，分为逻辑数据、字符数据。

第一节　数值数据的表示

一、进位计数制及相互转换

（一）进位计数制

任意一个数 $N = N_{n-1} N_{n-2} \cdots N_0 . N_{-1} N_{-2} \cdots N_{-m}$，它的值

$$(N)_R = \sum_{i=0}^{n-1} N_i R^i + \sum_{i=-1}^{-m} N_i R^i = \sum_{i=n-1}^{-m} N_i R^i$$

R为进位计数制的基数，R^i 是第i位的权；N_i 代表第i位上的一个数字符，可以是０～（Ｒ－１）符号中的任何一个。

(二)进位数制之间的转换

1.将R进制的数转换为十进制数

- 按权相加法：

$\sum_{i=n-1}^{-m} N_i R^i$ ：将各位数字与它的权相乘，其积相加，和数就是十进制数。

例：

$(101.101)_2 = 1\times2^2 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-3} = (5.625)_{10}$

$(25.7)_8 = 2\times8^1 + 5\times8^0 + 7\times8^{-1} = (21\frac{7}{8})_{10}$

$(5A.C)_{16} = 5\times16^1 + 10\times16^0 + 12\times16^{-1} = (90\frac{3}{4})_{10}$

(二)进位数制之间的转换

2.将十进制数转换为R进制的数

（1）整数部分的转换（除基取余法）

将被转换的十进制数连续除以R取其余数，直到商等于0为止。每次所得余数即为R进制的数（第一次余数为低位）。

例1：将$N = (357)_{10}$转换成二进制数。

解：

2 | 357 …… 1 （低位）

2 | 178 …… 0

2 | 89 …… 1

2 | 44 …… 0

2 | 22 …… 0

2 | 11 …… 1

2 | 5 …… 1

2 | 2 …… 0

2 | 1 …… 1

0

↓

（高位）

则：$(357)_{10}=(101100101)_2$

例2：将 $N=(355)_{10}$ 转换成八进制数。

解：

$8 \mid 355$ …… 3 （低位）

$8 \mid 44$ …… 4

$8 \mid 5$ …… 5

0 （高位）

则 $(355)_{10}=(543)_8$

例3：将 $N=(357)_{10}$ 转换成十六进制数。

解：

$16 \mid 357$ …… 5（低位）

$16 \mid 22$ …… 6

$16 \mid 1$ …… 1

0 （高位）

则 $(357)_{10}=(165)_{16}$

(二)进位数制之间的转换

（2）小数部分的转换（乘基取整法）

将被转换的十进制数连续乘以R，取其整数，直到小数部分为0或达到要求的精度为止。（第一次整数为高位）。

例1　将 $N = (0.385)_{10}$ 转换成二进制小数

解：

	0.	385×2
（高位）	0.	77×2
	1.	54×2
	1.	08×2
	0.	16×2
	0.	32×2
（低位）	0.	64×2
	1.	28

则：$(0.385)_{10} = (0.0110001)_2$

例2　将$N = (0.385)_{10}$转换成八进制小数。

解：

	0 .	385×8
（高位）	3 .	08×8
	0 .	64×8
	5 .	12
（低位）		

则：$(0.385)_{10} = (0.305)_8$

(二)进位数制之间的转换

3.二进制与八、十六进制数之间的转换

（1）二进制数转换为八、十六进制数

- 以小数点为中心，向左右两边延伸。八进制按三位一组划分，十六进制按四位一组划分。

例如：$(100101.101)_2 = (45.5)_8 = (25.A)_{16}$

$(0.011000101)_2 = (0.305)_8 = (0.628)_{16}$

(二)进位数制之间的转换

（2）八、十六进制数转换为二进制数

将每一位八（或十六）进制数用三位（或四位）二进制数代替即可。

例如：$(45.5)_8=(100101.101)_2$

$(25.A)_{16}=(00100101.1010)_2$

二、数的小数点表示

小数点数 { 定点表示法 { 定点小数 / 定点整数 } / 浮点表示法 }

（一）定点表示法：小数点的位置固定不变。

1. 定点小数：小数点固定在有效数字的最左边，该数为一纯小数。

2. 定点整数：小数点固定在有效数字的最右边，该数为整数。

定点小数的表数范围

- 若二进制位数为n（不包括符号位），则定点小数的
- 表数范围是：

$$0 \leq |N| \leq 1-2^{-n}$$

或

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

- **定点整数的表数范围**是：(n:不包括符号位)

$$0 \leq |N| \leq 2^{n}-1$$

或

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

- **定点整数也可视为无符号整数**。n+1位无符号整数的表数范围是：

$$0 \leq N \leq 2^{n+1}-1$$

（二）浮点表示法

例：　$258.69 = 10^{1} \times 25.869$

$= 10^{2} \times 2.5869$

$= 10^{-1} \times 2586.9$

$= 10^{-2} \times 25869$

……

对于任意数N，

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

E($E_{xponent}$)被称为浮点数的阶码，M($M_{antissa}$)被称为浮点数的尾数，R(R_{adix})被称为阶的基数。

（二）浮点表示法

- 浮点数只需用一对定点数（阶码和尾数）来表示

1. 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），基数为2，

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

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

2.规格化浮点数

正数，规格化表示的尾数形式为

0.1xx…x

（二）浮点表示法

- 补码表示的负数，规格化表示的尾数形式为

  1.0xx…x

根据规格化尾数形式，当运算结果尾数出现 **00.0xx…x** 或 **11.1xx…x** 时，需将尾数左移以实现规格化；尾数每左移一位（小数点位置不动）阶码减1，直至尾数的符号和最高位具有不同的代码达到规格化为止。

三、数的符号表示

把符号位和数值位一起编码来表示相应数的各种编码方法——原码、补码、反码和移码。

（一）三种编码方法的比较

1.真值与机器数

真值:用正负号加绝对值表示的数值。

机器数:用约定数的某一位表示符号,连同数符一起数码化的数。

例：+1010 — 01010

-1010 — 11010

2.原码表示法

① 原码表示形式:最高位表示符号；

符号位为0,该数为正；

符号位为1,该数为负。

例:设机器字长共8位（含一位符号位）。

真值 x:1011,-1011,0.1011,-0.1011

原码$[x]_{原}$: 0,0001011;1,0001011;

0.1011000;1.1011000

2.原码表示法

② 原码表数范围

(1) n位整数N的表数范围（n为不包括符号位在内的整数）：

$$-(2^n-1)\leq N\leq 2^n-1$$

(2) n位小数N的表数范围（n为小数的位数）：

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

3.补码表示法

① 补码的概念:2位十进制运算器

56-24=32; 56+76=132

- 100:是两位十进制运算器的溢出量,在数学上称之为模,用M或 mod表示。计算器中数受字长的限制,运算均是有模运算。
- 所以56-24=56+76(mod 100)
- 即-24（相对模100）的补码是76。
- 补码定义：$[x]_{补}$= M+X（mod M)

② 补码的表示形式:

- 定点整数:$[x]_{补}=X_n\ X_{n-1}\ X_{n-2}\ \cdots\cdots X_1\ X_0$
- 定点小数:$[x]_{补}=X_0.X_1\ X_2\ \cdots\cdots X_{n-1}\ X_n$

(1) 从真值转换成补码表示:

正数:补码表示同原码。

例: 真值:+1011 原码：01011 补码：01011

负数:符号位为1,数值部分为真值的各位求反,末位加1。

例:真值:-1010 ;　-0.1010

　补码：10110;　1.0110

(2) 从补码求原码及真值

- 正数:原码与补码相同,真值为略去正号后的数值。
- 例:$[X]_{补}$=0010 $[X]_{原}$=0010 真值 X=010=10
- 负数:
- 原码:符号位仍为1,数值部分为:把尾数各位求反,末位加1。
- 真值：将负数原码符号变为“-”,即得到真值。
- 例:$[X]_{补}$=10110;

  $[X]_{原}$=11010; 真值 X= -1010

③ 补码的表数范围:

(1) n位整数N的表数范围:（n为不包括符号位在内的整数）

$$-2^n \leq N \leq 2^n - 1$$

(2) n位小数N的表数范围:（n为小数的位数）

$$-1 \leq N \leq 1-2^{-n}$$

4.反码的表示法

正数:与原码相同。

例:$[X]_{原}=01010$； $[X]_{反}=01010$

负数:符号位同原码,尾数部分为原码的反码。

例: $[X]_{原}=11010$；$[X]_{反}=10101$

反码表数范围同原码。

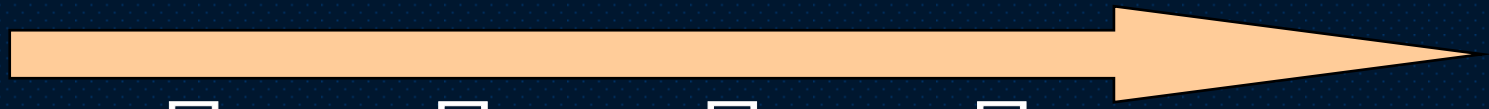
5.补码表示的浮点数

(1) 浮点数的表数范围：

阶码l位，尾数n位（均不含符号位、补码表示）

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

例：阶码E=8位,M=24位 （均含一位符号位、阶的基为2）。



最小负数	最大负数	0	最小正数	最大正数

- 最大正数: $2^{2^e-1}*(1-2^{-m})= 2^{127}*(1-2^{-23})$
- 最小正数: $2^{-2^e}*2^{-m} = 2^{-128}*2^{-23}$
- 最大负数: $-2^{-2^e}*2^{-m} = -2^{-128}*2^{-23}$
- 最小负数:$(-1)* 2^{2^e-1} =-2^{127}$

整个浮点数的表数范围:

$-2^{127} \leq N \leq 2^{127}* (1-2^{-23})$

$-2^{2^e-1} \leq N \leq 2^{2^e-1}* (1-2^{-m})$

(2) 浮点数的规格化表示(尾数的最高有效位与符号位不相同)

例:某浮点数长12位，阶码4位,尾数8位(均包括一位符号位,用补码表示)，写出真值为$(-101.011)_2$的规格化浮点代码(规格化机器数)。

$(-101.011)_2 = -0.101011*2^{+3}$ 尾数补码表示 1.0101010

阶码补码表示 0,011

补码表示

采用格式：

Ms	E	m

规格化浮点数代码:

1	0011	0101010

即：1, 0011, 0101010

例：（同上题表示法）

写出浮点代码为0，0100，0101010的规格化浮点数。

解：小数点右移一位，阶码减1（末尾补0）

0，0011，1010100

若:阶码E=4 (e=3)位，尾数M=8 (m=7)位

规格化浮点数的表数范围

上溢区 | 负浮点数 | 下溢区 | 正浮点数 | 上溢区

0

$-1*2^{2^e-1}$ $-1/2*2^{-2^e}$ $1/2*2^{-2^e}$ $1*2^{2^e-1}$

- 正数：（$1/2*2^{-2^3} \le N < 1*2^{2^3-1}$）

$$2^{1,000}*0.10\ldots0 \le N \le 2^{0,111}*0.11\ldots1$$

- 负数：（$-1*2^{2^3-1} \le N \le -1/2*2^{-2^3}$）

$$1.0\ldots0*2^{0,111} \le N \le 2^{1,000}*1.10\ldots0$$

（二）移码

例如 E = 7（包括一位符号位）

$e_{未偏置}$	-64	0	63
$e_{偏置}$	0	64	127

1. 移码定义

如果阶码有n+1位（包括一位符号位），其阶码的表数范围为-2^n~$+(2^n-1)$，则阶码x的移码定义为：

$$[x]_{移} = 2^n + x，\ -2^n \leq x \leq 2^n-1$$

2. 移码的性质

（1）移码为全0时，表示真值最小；移码为全1时，表示真值最大。

（2）当x<0时，$[x]_{移}$的符号位（最高位）为0，当x≥0时，$[x]_{移}$的符号位为1，移码符号与原、补码符号相反。

（3）$[x]_{移}$与$[x]_{补}$除符号位相反外，其他各位相同。因此由$[x]_{补}$得到$[x]_{移}$的方法是变 $[x]_{补}$的符号为其反码。

（4）在移码表示中，0有唯一的编码100…0。

（5）$[x]_{移}$等于全0时，表明阶码最小。

- 一个浮点数$N = M \cdot R^E$，当尾数M=0时，不论其阶码为何值都有N=0。
- 当$E<-2^n$时，(M=0 or M≠0)，我们称发生下溢，即数N小于机器所能表示的最小数，一般以N=0处理。
- 为了保证唯一性，规定一个标准的浮点数零的表示形式，称为“机器0”，它具有0的尾数和最小阶码。阶码采用移码表示后，浮点数的“机器0”就是尾数和阶码全为0。

（三）实用浮点格式举例

IEEE754标准浮点格式

IEEE754标准在标识浮点数时，每个浮点数均由三个部分组成：符号位S，指数部分E和尾数部分M。

浮点数可采用以下四种基本格式：

（1）单精度格式（32位）：E＝8位，M＝23位。

（2）扩展单精度格式：E≥11位，M≥31位。

（3）双精度格式（64位）：E＝11位，M＝52位。

（4）扩展双精度格式：E≥15位，M≥63位。

32位浮点单精度数据形式

当$E=0$且$M=0$，则数值N为0

当$E=0$且$M\neq 0$，则数值N为 $(-1)^S 2^{E-126}(0.M)$，非规格化数

当$0<E<255$，数值N为 $(-1)^S 2^{E-127}(1.M)$，规格化数

当$E=255$，且$M\neq 0$，则N为非数值

$E=255$且$M=0$，则N为 $(-1)^S\infty$，无穷大

32位浮点单精度数据形式特点

IEEE754标准约定32位单精度形式在小数点左部有一位隐含位，从而使其有效位实际有24位，这样便使尾数的有效值变为1.M 。阶码部分采用移码表示，移码值为127，从而使阶码值的范围由原来的-126到+127，经移码后变为1到254。

IEEE754标准使0有了精确表示，同时也明确地表示了无穷大，所以，当a/0(a≠0)时得到结果值为±∞；当0/0时得到结果值较小的数，为了避免下溢而损失精度，允许采用比最小规格化数还要小的数来表示，这些数称为非规格化数。应注意的是，非规格化数和正、负零的隐含位值不是1而是0。

例1：若采用IEEE754浮点单精度格式，试求出32位浮点代码（CC968000）$_{16}$ 的真值。

解：1，10011001，00101101000000000000000

- 阶码真值 = 阶码-（127）$_{10}$=（10011001）$_{2}$-（127）$_{10}$

  =(153) $_{10}$ -(127) $_{10}$ =(26） $_{10}$

- 尾数真值= 1+0.00101101=（1.00101101）$_{2}$

  = (1.17578125）$_{10}$

- 浮点数真值= -2^{26} *(1.17578125）$_{10}$

例2：将（-0.11）$_2$用IEEE754浮点单精度格式表示出来。

解: $(-0.11)_2 = -0.11*2^0 = -1.1*2^{-1} = -(1+0.1)*2^{-1}$

- 阶码＝阶码真值+127＝-1+127＝126 ＝$(01111110)_2$
- 浮点代码为:1,01111110,100...0

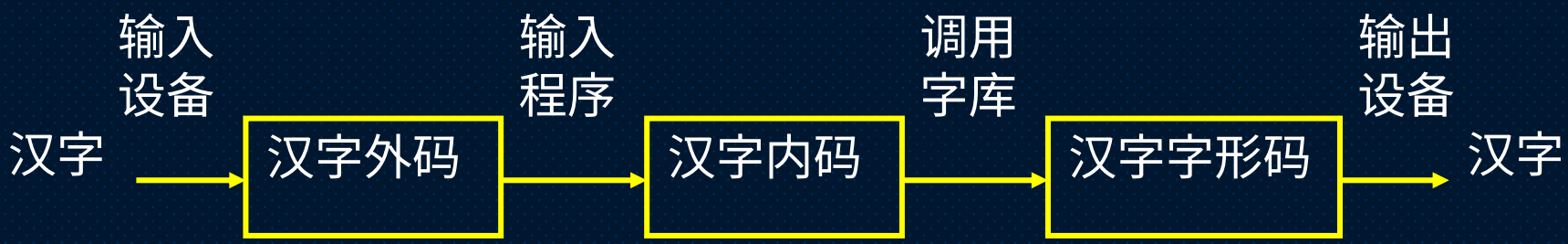
第二节　非数值数据的表示

一、字符表示

- 字符数据：ASCII码
- 一个字符：占一个字节单元。
- 多个字符：通常占用主存多个连续字节单元。
- ASCII码主要用于主机与I/O设备之间交换信息。

第二节　非数值数据的表示

二、汉字表示



三、校验码（奇偶校验码）

数据	偶校验编码C	奇校验编码C
10101010	10101010 **0**	10101010 **1**
01010100	01010100 **1**	01010100 **0**
01111111	01111111 **1**	01111111 **0**