表 6.2 真值、补码和移码对照表

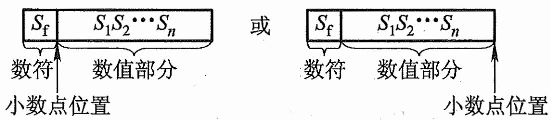
真值 x	$[x]_补$	$[x]_移$	$[x]_移$对应的十进制整数
-100000	100000	000000	0
-11111	100001	000001	1
-11110	100010	000010	2
⋮	⋮	⋮	⋮
-00001	111111	011111	31
±00000	000000	100000	32
+00001	000001	100001	33
+00010	000010	100010	34
⋮	⋮	⋮	⋮
+11110	011110	111110	62
+11111	011111	111111	63

6.2 数的定点表示和浮点表示

在计算机中，小数点不用专门的器件表示，而是按约定的方式标出，共有两种方法表示小数点的存在，即定点表示和浮点表示。定点表示的数称为定点数，浮点表示的数称为浮点数。

6.2.1 定点表示

小数点固定在某一位置的数为定点数，有以下两种格式。



当小数点位于数符和第一数值位之间时，机器内的数为纯小数；当小数点位于数值位之后时，机器内的数为纯整数。采用定点数的机器称为定点机。数值部分的位数 n 决定了定点机中数的表示范围。若机器数采用原码，小数定点机中数的表示范围是 $-(1-2^{-n}) \sim (1-2^{-n})$，整数定点机中数的表示范围是 $-(2^n-1) \sim (2^n-1)$。

在定点机中，由于小数点的位置固定不变，故当机器处理的数不是纯小数或纯整数时，必须乘上一个比例因子，否则会产生“溢出”。

6.2.2 浮点表示

实际上计算机中处理的数不一定是纯小数或纯整数(如圆周率 3.141 6),而且有些数据的数值范围相差很大(如电子的质量 9×10^{-28}g,太阳的质量 2×10^{33}g),它们都不能直接用定点小数或定点整数表示,但均可用浮点数表示。浮点数即小数点的位置可以浮动的数,如

$$\begin{aligned}352.47 &= 3.5247\times10^{2}\\ &= 3524.7\times10^{-1}\\ &= 0.35247\times10^{3}\end{aligned}$$

显然,这里小数点的位置是变化的,但因为分别乘上了不同的 10 的方幂,故值不变。

通常,浮点数被表示成

$$N=S\times r^{j}$$

式中,S 为尾数(可正可负),j 为阶码(可正可负),r 是基数(或基值)。在计算机中,基数可取 2,4、8 或 16 等。

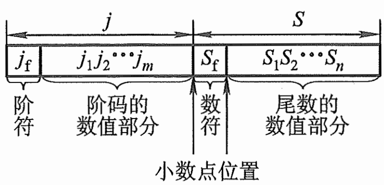
以基数 $r=2$ 为例,数 N 可写成下列不同的形式:

$$\begin{aligned}N &= 11.0101\\ &= 0.110101\times2^{10}\\ &= 1.10101\times2^{1}\\ &= 1101.01\times2^{-10}\\ &= 0.00110101\times2^{100}\\ &\ \ \vdots\end{aligned}$$

为了提高数据精度以及便于浮点数的比较,在计算机中规定浮点数的尾数用纯小数形式,故上例中 0.110101×2^{10} 和 0.00110101×2^{100} 形式是可以采用的。此外,将尾数最高位为 1 的浮点数称为规格化数,即 $N=0.110101\times2^{10}$ 为浮点数的规格化形式。浮点数表示成规格化形式后,其精度最高。

1. 浮点数的表示形式

浮点数在机器中的形式如下所示。采用这种数据格式的机器称为浮点机。



浮点数由阶码 j 和尾数 S 两部分组成。阶码是整数,阶符和阶码的位数 m 合起来反映浮点数的表示范围及小数点的实际位置;尾数是小数,其位数 n 反映了浮点数的精度;尾数的符号 S_f

代表浮点数的正负。

2. 浮点数的表示范围

以通式 $N=S\times r^j$ 为例，设浮点数阶码的数值位取 m 位，尾数的数值位取 n 位，当浮点数为非规格化数时，它在数轴上的表示范围如图 6.2 所示。

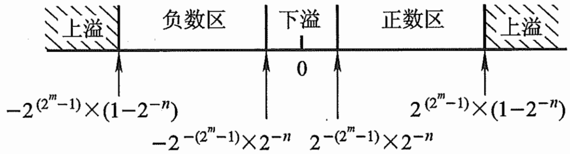


图 6.2 浮点数在数轴上的表示范围

由图中可见，其最大正数为 $2^{(2^m-1)}\times(1-2^{-n})$；最小正数为 $2^{-(2^m-1)}\times 2^{-n}$；最大负数为 $-2^{-(2^m-1)}\times 2^{-n}$；最小负数为 $-2^{(2^m-1)}\times(1-2^{-n})$。当浮点数阶码大于最大阶码时，称为上溢，此时机器停止运算，进行中断溢出处理；当浮点数阶码小于最小阶码时，称为下溢，此时溢出的数绝对值很小，通常将尾数各位强置为零，按机器零处理，此时机器可以继续运行。

一旦浮点数的位数确定后，合理分配阶码和尾数的位数，直接影响浮点数的表示范围和精度。通常对于短实数（总位数为 32 位），阶码取 8 位（含阶符 1 位），尾数取 24 位（含数符 1 位）；对于长实数（总位数为 64 位），阶码取 11 位（含阶符 1 位），尾数取 53 位（含数符 1 位）；对于临时实数（总位数为 80 位），阶码取 15 位（含阶符 1 位），尾数取 65 位（含数符 1 位）。

3. 浮点数的规格化

为了提高浮点数的精度，其尾数必须为规格化数。如果不是规格化数，就要通过修改阶码并同时左右移尾数的办法，使其变成规格化数。将非规格化数转换成规格化数的过程称为规格化。对于基数不同的浮点数，因其规格化数的形式不同，规格化过程也不同。

当基数为 2 时，尾数最高位为 1 的数为规格化数。规格化时，尾数左移一位，阶码减 1（这种规格化称为向左规格化，简称左规）；尾数右移一位，阶码加 1（这种规格化称为向右规格化，简称右规）。图 6.2 所示的浮点数规格化后，其最大正数为 $2^{(2^m-1)}\times(1-2^{-n})$，最小正数为 $2^{-(2^m-1)}\times 2^{-1}$；最大负数为 $-2^{-(2^m-1)}\times 2^{-1}$，最小负数为 $-2^{(2^m-1)}\times(1-2^{-n})$。

当基数为 4 时，尾数的最高两位不全为零的数为规格化数。规格化时，尾数左移两位，阶码减 1；尾数右移两位，阶码加 1。

当基数为 8 时，尾数的最高三位不全为零的数为规格化数。规格化时，尾数左移三位，阶码减 1；尾数右移三位，阶码加 1。

同理类推，不难得到基数为 16 或 2^n 时的规格化过程。

浮点机中一旦基数确定后就不再变了，而且基数是隐含的，故不同基数的浮点数表示形式完全相同。但基数不同，对数的表示范围和精度等都有影响。一般来说，基数 r 越大，可表示的浮点数范围越大，而且所表示的数的个数越多。但 r 越大，浮点数的精度反而下降。如 $r=16$ 的浮

点数，因其规格化数的尾数最高三位可能出现零，故与其尾数位数相同的 $r=2$ 的浮点数相比，后者可能比前者多三位精度。

6.2.3　定点数和浮点数的比较

定点数和浮点数可从如下几个方面进行比较。

① 当浮点机和定点机中数的位数相同时，浮点数的表示范围比定点数的大得多。

② 当浮点数为规格化数时，其相对精度远比定点数高。

③ 浮点数运算要分阶码部分和尾数部分，而且运算结果都要求规格化，故浮点运算步骤比定点运算步骤多，运算速度比定点运算的低，运算线路比定点运算的复杂。

④ 在溢出的判断方法上，浮点数是对规格化数的阶码进行判断，而定点数是对数值本身进行判断。例如，小数定点机中的数，其绝对值必须小于 1，否则“溢出”，此时要求机器停止运算，进行处理。为了防止溢出，上机前必须选择比例因子，这个工作比较麻烦，给编程带来不便。而浮点数的表示范围远比定点数大，仅当“上溢”时机器才停止运算，故一般不必考虑比例因子的选择。

总之，浮点数在数的表示范围、数的精度、溢出处理和程序编程方面（不取比例因子）均优于定点数。但在运算规则、运算速度及硬件成本方面又不如定点数。因此，究竟选用定点数还是浮点数，应根据具体应用综合考虑。一般来说，通用的大型计算机大多采用浮点数，或同时采用定、浮点数；小型、微型及某些专用机、控制机则大多采用定点数。当需要做浮点运算时，可通过软件实现，也可外加浮点扩展硬件（如协处理器）来实现。

6.2.4　举例

例 6.3　设浮点数字长 16 位，其中阶码 5 位（含 1 位阶符），尾数 11 位（含 1 位数符），将十进制数 $+\frac{13}{128}$ 写成二进制定点数和浮点数，并分别写出它在定点机和浮点机中的机器数形式。

解：令 $x=+\frac{13}{128}$

其二进制形式　　$x=0.0001101000$

定点数表示　　$x=0.0001101000$

浮点数规格化表示　　$x=0.1101000000\times2^{-11}$

定点机中　　$[x]_{原}=[x]_{补}=[x]_{反}=0.0001101000$

浮点机中

$[x]_{原}$：	1	0011	0	1101000000	或写成1，0011；0. 1101000000
$[x]_{补}$：	1	1101	0	1101000000	或写成1，1101；0. 1101000000
$[x]_{反}$：	1	1100	0	1101000000	或写成1，1100；0. 1101000000

例 6.4　将十进制数-54 表示成二进制定点数和浮点数,并写出它在定点机和浮点机中的机器数形式(其他要求同上例)。

解:令 $x=-54$

其二进制形式　　$x=-110110$

定点数表示　　$x=-0000110110$

浮点数规格化表示　　$x=-(0.1101100000)\times 2^{110}$

定点机中

$$[x]_{原}=1,0000110110$$
$$[x]_{补}=1,1111001010$$
$$[x]_{反}=1,1111001001$$

浮点机中

$$[x]_{原}=0,0110;\ 1.1101100000$$
$$[x]_{补}=0,0110;\ 1.0010100000$$
$$[x]_{反}=0,0110;\ 1.0010011111$$

例 6.5　写出对应图 6.2 所示的浮点数的补码形式。设图中 $n=10,m=4$,阶符、数符各取 1 位。

解:	真值	补码
最大正数	$2^{15}\times(1-2^{-10})$	0,1111; 0.1111111111
最小正数	$2^{-15}\times 2^{-10}$	1,0001; 0.0000000001
最大负数	$-2^{-15}\times 2^{-10}$	1,0001; 1.1111111111
最小负数	$-2^{15}\times(1-2^{-10})$	0,1111; 1.0000000001

计算机中浮点数的阶码和尾数可以采用同一种机器数表示,也可采用不同的机器数表示。

例 6.6　设浮点数字长为 16 位,其中阶码为 5 位(含 1 位阶符),尾数为 11 位(含 1 位数符),写出$-\frac{53}{512}$对应的浮点规格化数的原码、补码、反码和阶码用移码,尾数用补码的形式。

解:设 $x=-\dfrac{53}{512}=-\ 0.000110101=2^{-11}\times(-0.1101010000)$

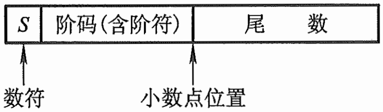
$$[x]_{原}=1,0011;\ 1.1101010000$$
$$[x]_{补}=1,1101;\ 1.0010110000$$
$$[x]_{反}=1,1100;\ 1.0010101111$$
$$[x]_{阶移,尾补}=0,1101;\ 1.0010110000$$

值得注意的是,当一个浮点数尾数为 0 时,不论其阶码为何值;或阶码等于或小于它所能表示的最小数时,不管其尾数为何值,机器都把该浮点数作为零看待,并称之为“机器零”。如果浮点数的阶码用移码表示,尾数用补码表示,则当阶码为它所能表示的最小数 2^{-m}(式中 m 为阶码的位数)且尾数为 0 时,其阶码(移码)全为 0,尾数(补码)也全为 0,这样的机器零为 000…0000,

全零表示有利于简化机器中判“0”电路。

6.2.5　IEEE 754 标准

现代计算机中,浮点数一般采用 IEEE 制定的国际标准,这种标准形式如下:



按 IEEE 标准,常用的浮点数有三种:

	符号位 S	阶码	尾数	总位数
短实数	1	8	23	32
长实数	1	11	52	64
临时实数	1	15	64	80

其中,S 为数符,它表示浮点数的正负,但与其有效位(尾数)是分开的。阶码用移码表示,阶码的真值都被加上一个常数(偏移量),如短实数、长实数和临时实数的偏移量用十六进制数表示分别为 7FH、3FFH 和 3FFFH(见附录 6A.1)。尾数部分通常都是规格化表示,即非“0”的有效位最高位总是“1”,但在 IEEE 标准中,有效位呈如下形式。

$$1_{\blacktriangle}\text{ffff}\cdots\cdots\text{fff}$$

其中▲表示假想的二进制小数点。在实际表示中,对短实数和长实数,这个整数位的 1 省略,称隐藏位;对于临时实数不采用隐藏位方案。表 6.3 列出了十进制数 178.125 的实数表示。

表 6.3　实数 178.125 的几种不同表示

实数表示	数　值		
原始十进制数	178.125		
二进制数	10110010.001		
二进制浮点表示	$1.0110010001\times2^{111}$		
短实数表示	符号	偏移的阶码	有效值
	0	00000111+01111111 =10000110	01100100010000000000000 ↑ $1_{\blacktriangle}$(隐含的)