

2.1.1 什么是真值与机器数?

真值:用正、负号来分别表示正数和负数,这样的数称为真值。
机器数:用1位数码0或1来表示数的正负号,这样的数称为机器数。

2.1.2 数的机器码如何表示?

1. 原码表示法

(1) 纯小数的原码定义

设 $x = x_0.x_1x_2\cdots x_{n-1}$ 共 n 位字长,则

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

(2) 纯整数的原码定义

设 $x = x_0x_1x_2\cdots x_{n-1}$ 共 n 位字长,则

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

原码表示的规则:用“0”表示正号,用“1”表示负号,有效值用二进制的绝对值表示。
真值零的原码有正零和负零两种形式:

$$[+0]_{\text{原}} = 000\cdots 00 \quad [-0]_{\text{原}} = 100\cdots 00$$

其中最高位是符号位。

2. 补码表示法

(1) 纯小数的补码定义

设 $x = x_0.x_1x_2\cdots x_{n-1}$ 共 n 位字长,则

$$[x]_{\text{补}} = \begin{cases} x & \text{当 } 0 \leq x \leq 1-2^{-(n-1)} \\ 2+x = 2-|x| & \text{当 } -1 \leq x < 0 \end{cases}$$

注意, $[-1]_{\text{补}} = 1.0\cdots 0$

纯小数的模总是为2。

(2) 纯整数的补码定义

设 $x = x_0x_1x_2\cdots x_{n-1}$ 共 n 位字长,则

$$[x]_{\text{补}} = \begin{cases} x & \text{当 } 0 \leq x \leq 2^{n-1}-1 \\ 2^n+x = 2^n-|x| & \text{当 } -2^{n-1} \leq x < 0 \end{cases}$$

注意, $[-2^{n-1}]_{\text{补}} = 10\cdots 0$ 。

n 位整数的模 M 的大小是: n 位数全为1后,在最末位加1。如果某数有 n 位整数(包括1位符号位),则它的模为 2^n 。

无论是纯小数补码还是纯整数补码,其零的补码是唯一的,即 $[+0]_{\text{补}} = [-0]_{\text{补}} = 0\cdots 0$ 。

补码表示的规则:正数的补码与原码相同,负数的补码符号为“1”,数值部分求反加1。

3. 反码表示法

(1) 纯小数的反码定义

设 $x = x_0.x_1x_2\cdots x_{n-1}$ 共 n 位字长,则

$$[x]_{\text{反}} = \begin{cases} x & \text{当 } 0 \leq x \leq 1-2^{-(n-1)} \\ 2+x-2^{-(n-1)} & \text{当 } -(1-2^{-(n-1)}) \leq x \leq 0 \end{cases}$$

(2) 纯整数的反码定义

设 $x = x_0x_1x_2\cdots x_{n-1}$ 共 n 位字长,则

$$[x]_{\text{反}} = \begin{cases} x & \text{当 } 0 \leq x \leq 2^{n-1}-1 \\ 2^n+x-1 & \text{当 } -(2^{n-1}-1) \leq x \leq 0 \end{cases}$$

正数的反码与原码相同,负数的反码符号为“1”,数值部分求反。

在反码表示中: $[+0]_{\text{反}} = 0\cdots 0$, $[-0]_{\text{反}} = 11\cdots 11$ 。

4. 负数原码、反码、补码、真值之间的相互转化

(1) 已知: $[x]_{\text{原}} = 1.x_1x_2\cdots x_n$, 则 $[x]_{\text{补}} = 1.x_1x_2\cdots x_n + 2^{-n}$, 即符号不变,数值部分求反,加1。

【证明】 当 $-1 \leq x \leq 0$ 时,原码的定义为 $[x]_{\text{原}} = 1-x$,补码的定义为 $[x]_{\text{补}} = 2+x$,所以

$$\begin{aligned} x &= [x]_{\text{补}} - 2 = 1 - [x]_{\text{原}} \\ \text{因为 } [x]_{\text{原}} &= 1.x_1x_2\cdots x_n \\ \text{所以 } [x]_{\text{补}} &= 3 - 1.x_1x_2\cdots x_n \\ &= 2 - 0.x_1x_2\cdots x_n \\ &= 1 + (1 - 0.x_1x_2\cdots x_n) \end{aligned}$$

又因为 $x_i + \overline{x_i} = 1$, 所以有

$$0.x_1x_2\cdots x_n + 0.\overline{x_1}\overline{x_2}\cdots\overline{x_n} + 2^{-n} = 0.111\cdots 1 + 2^{-n} = 1$$

$$1 - 0.x_1x_2\cdots x_n = 0.\overline{x_1}\overline{x_2}\cdots\overline{x_n} + 2^{-n}$$

所以 $[x]_{\text{补}} = 1 + 0.\overline{x_1}\overline{x_2}\cdots\overline{x_n} + 2^{-n} = 1.\overline{x_1}\overline{x_2}\cdots\overline{x_n} + 2^{-n}$

(2) 已知: $[x]_{\text{补}} = 1.x_1x_2\cdots x_n$, 则 $[x]_{\text{原}} = 1.\overline{x_1}\overline{x_2}\cdots\overline{x_n} + 2^{-n}$, 即符号不变,数值部分求反,加1。

【证明】 当 $-1 \leq x \leq 0$ 时,补码的定义为 $[x]_{\text{补}} = 1+x$,原码的定义为 $[x]_{\text{原}} = 1-x$, 所以

$$\begin{aligned} x &= [x]_{\text{补}} - 1 = -[x]_{\text{原}} \\ \text{因为 } [x]_{\text{补}} &= 1.x_1x_2\cdots x_n \\ \text{所以 } [x]_{\text{原}} &= 3 - 1.x_1x_2\cdots x_n \\ &= 2 - 0.x_1x_2\cdots x_n \\ &= 1 + (1 - 0.x_1x_2\cdots x_n) \end{aligned}$$

又因为 $x_i + \overline{x_i} = 1$, 所以有

$$0.x_1x_2\cdots x_n + 0.\overline{x_1}\overline{x_2}\cdots\overline{x_n} + 2^{-n} = 0.111\cdots 1 + 2^{-n} = 1$$

$$1 - 0.x_1x_2\cdots x_n = 0.\overline{x_1}\overline{x_2}\cdots\overline{x_n} + 2^{-n}$$

所以 $[x]_{\text{原}} = 1 + 0.\overline{x_1}\overline{x_2}\cdots\overline{x_n} + 2^{-n} = 1.\overline{x_1}\overline{x_2}\cdots\overline{x_n} + 2^{-n}$

(3) 原码与反码之间的相互转化是符号不变,数值部分求反(证明略)。

(4) 补码与反码之间的相互转化关系如下:

$$[x]_{\text{补}} = [x]_{\text{反}} + 2^{-n}, [x]_{\text{反}} = [x]_{\text{补}} - 2^{-n}$$

现证明 $[x]_{\text{补}} = [x]_{\text{反}} + 2^{-n}$ 。

【证明】 因为 $-1 < x \leq 0$ 时, $[x]_{\text{反}} = 2 - 2^{-n} + x$, $[x]_{\text{补}} = 2 + x$, 移项得

$$x = [x]_{\text{反}} - 2 + 2^{-n}$$

$$x = [x]_{\text{补}} - 2$$

所以 $[x]_{\text{补}} - 2 = [x]_{\text{反}} - 2 + 2^{-n}$

故 $[x]_{\text{补}} = [x]_{\text{反}} + 2^{-n}$

(5) 补码与真值的关系。

若 $[x]_{\text{补}} = x_0.x_1x_2\cdots x_n$, 则 $x = -x_0 + \sum_{i=1}^n x_i 2^{-i}$ 。

【证明】 当 $x \geq 0$ 时, $x_0 = 0$, $[x]_{\text{补}} = 0.x_1x_2\cdots x_n = \sum_{i=1}^n x_i 2^{-i} = x$;

当 $x < 0$ 时, $x_0 = 1$, $[x]_{\text{补}} = 1.x_1x_2\cdots x_n = 2 + x$,

$$x = [x]_{\text{补}} - 2 = 1.x_1x_2\cdots x_n - 2 = -1 + 0.x_1x_2\cdots x_n = -1 + \sum_{i=1}^n x_i 2^{-i}$$

综上所述两种情况,可得出 $x = -x_0 + \sum_{i=1}^n x_i 2^{-i}$ 。

5. 移码

若纯整数 x 为 n 位(其中包括符号位),则其移码的定义为

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

因为 $[x]_{\text{移}} = \begin{cases} x & 0 \leq x \leq 2^{n-1}-1 \\ 2^n + x & -2^{n-1} \leq x < 0 \end{cases}$

所以,当 $0 \leq x \leq 2^{n-1}-1$ 时, $[x]_{\text{移}} = [x]_{\text{补}} + 2^{n-1}$;

当 $-2^{n-1} \leq x < 0$ 时, $[x]_{\text{移}} = 2^{n-1} + [x]_{\text{补}} - 2^n = [x]_{\text{补}} - 2^{n-1}$ 。

因此,把 $[x]_{\text{补}}$ 的符号位取反即得 $[x]_{\text{移}}$ 。

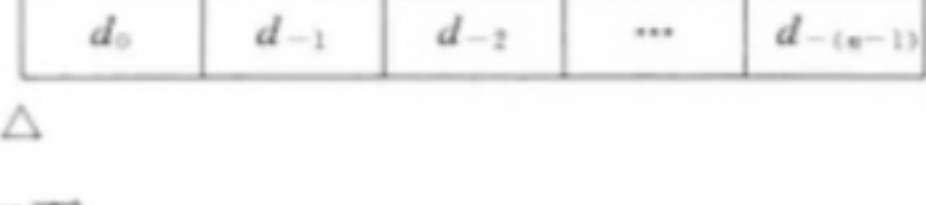
移码具有以下特点:

- 最高位为符号位,1表示正,0表示负。
 - 零的移码是唯一的,即 $[+0]_{\text{移}} = [-0]_{\text{移}} = 10\cdots 0$ 。
 - 用移码表示便于比较数的大小,移码大真值就大,移码小真值就小。
- 移码就是在真值 x 基础上加一个常数,相当于 x 在数轴上向正方向偏移了若干单位。
在多数通用计算机中,浮点数的阶码常用移码表示。

2.1.3 定点数如何表示?

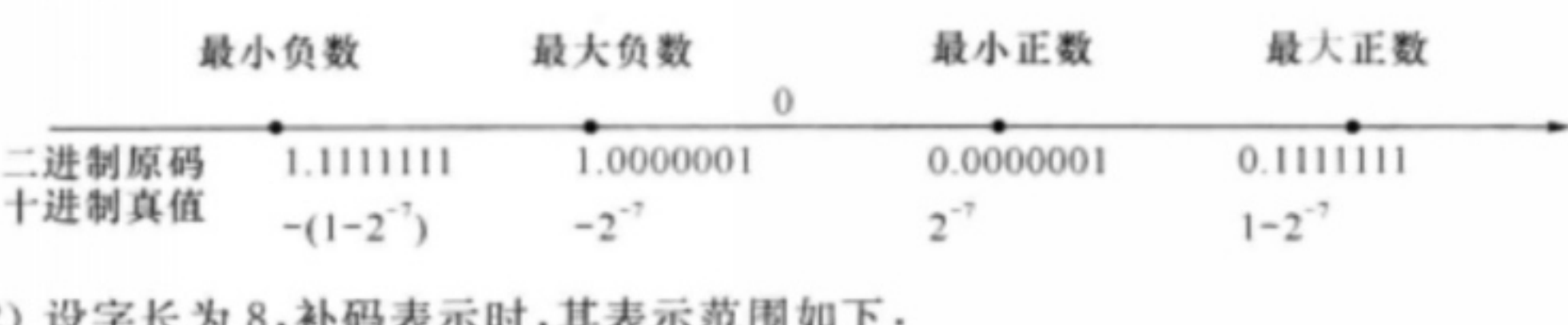
1. 定点小数

将小数点固定在符号位 d_0 之后,数值最高位 d_{-1} 之前,这就是定点小数形式。其格式如下所示:

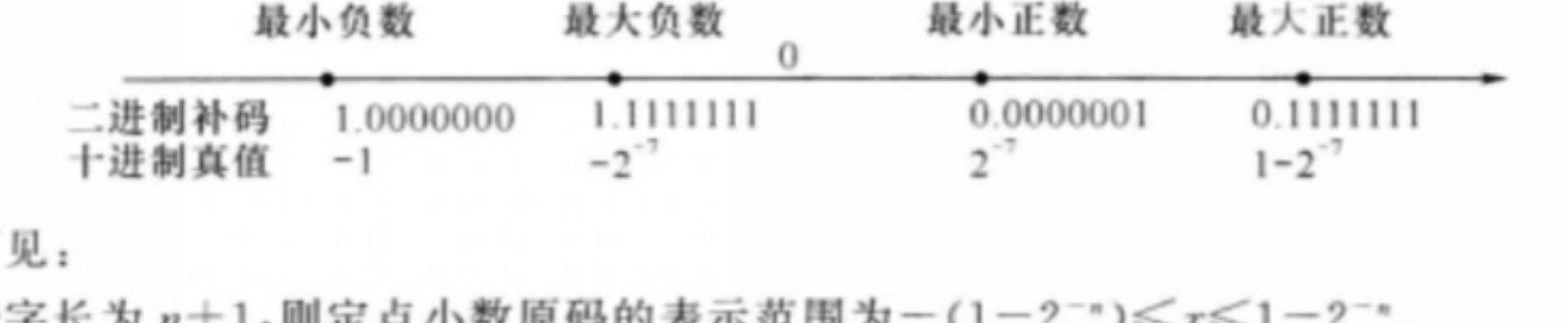


数据的表示范围分析如下。

(1) 设字长为8,原码表示时,其表示范围如下:



(2) 设字长为8,补码表示时,其表示范围如下:



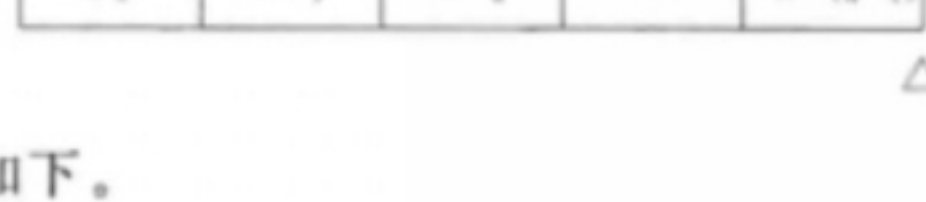
由此可见:

若字长为 $n+1$,则定点小数原码的表示范围为 $-(1-2^{-n}) \leq x \leq 1-2^{-n}$ 。

若字长为 $n+1$,则定点小数补码的表示范围为 $-1 \leq x \leq 1-2^{-n}$ 。

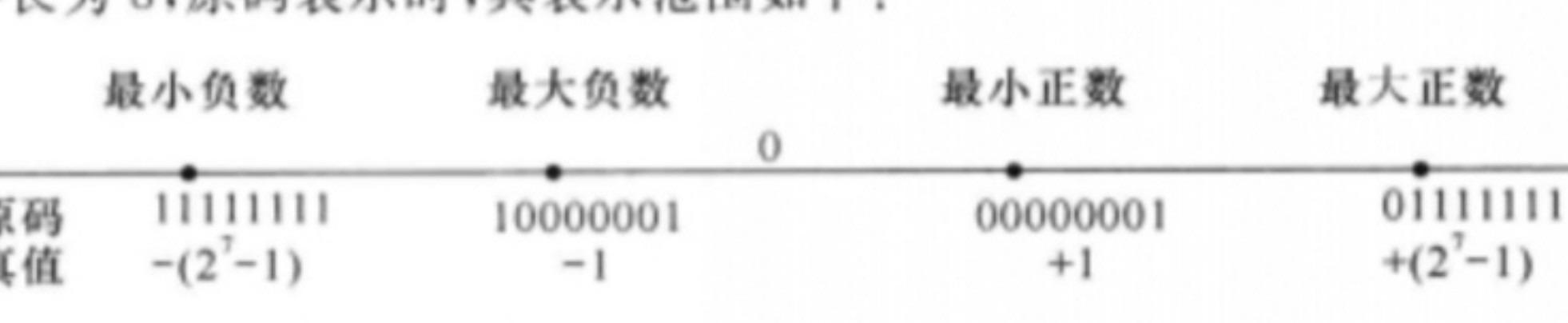
2. 定点整数

将小数点固定在数的最低位之后,这就是定点整数形式。其格式如下所示:

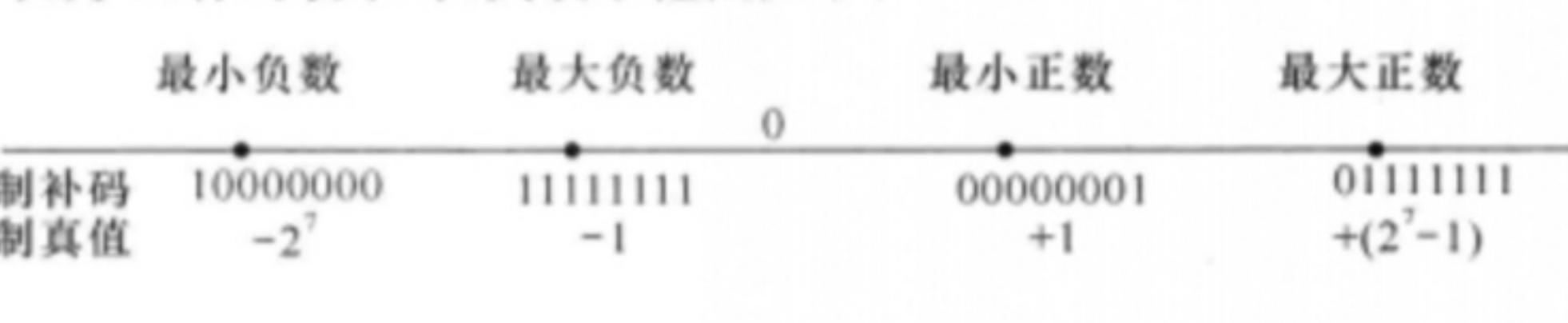


数据的表示范围分析如下。

(1) 设字长为8,原码表示时,其表示范围如下:



(2) 设字长为8,补码表示时,其表示范围如下:



由此可见:

若字长为 $n+1$,则定点整数原码的表示范围为 $-(2^n-1) \leq x \leq (2^n-1)$ 。

若字长为 $n+1$,则定点整数补码的表示范围为 $-2^n \leq x \leq (2^n-1)$ 。

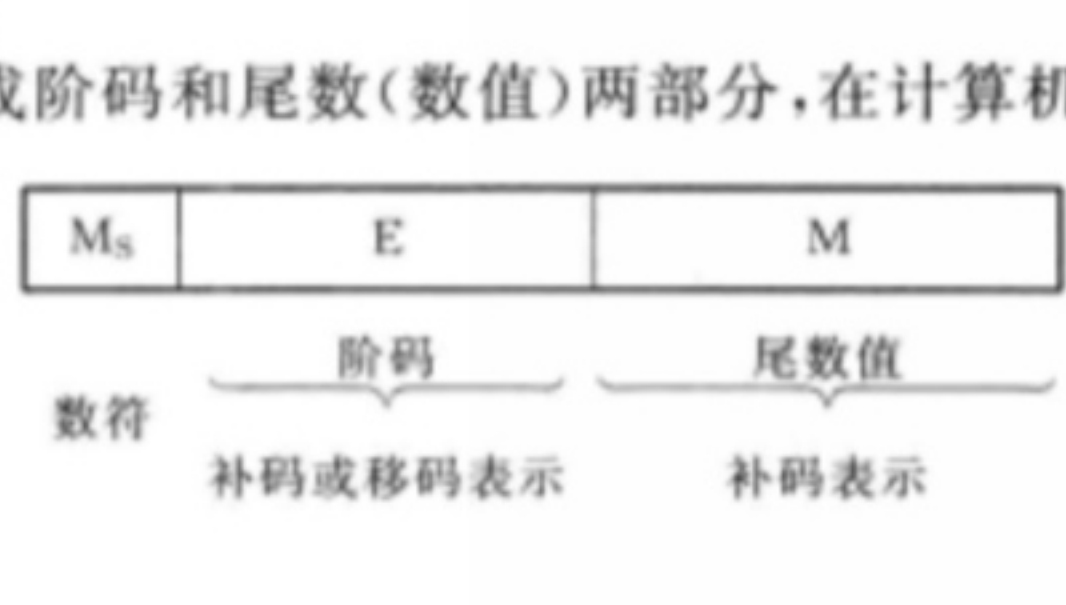
但是,如果是8位无符号定点整数,则其二进制编码范围是00000000~11111111,对应十进制真值范围是0~255。

在补码系统中,由于零有唯一的编码,因此 n 位二进制能表示 2^n 个补码。与原码表示相比,采用补码可多表示一个数。

2.1.4 浮点数如何表示?

1. 浮点数的表示格式

浮点表示是把字长分成阶码和尾数(数值)两部分,在计算机中的表示格式如下:



2. 浮点数的规格化

(1) 原码规格化后

正数的尾数形式为 $0.1 \times \cdots \times$ 的形式。

负数的尾数形式为 $1.1 \times \cdots \times$ 的形式。

(2) 补码规格化后

正数的尾数形式为 $0.1 \times \cdots \times$ 的形式。

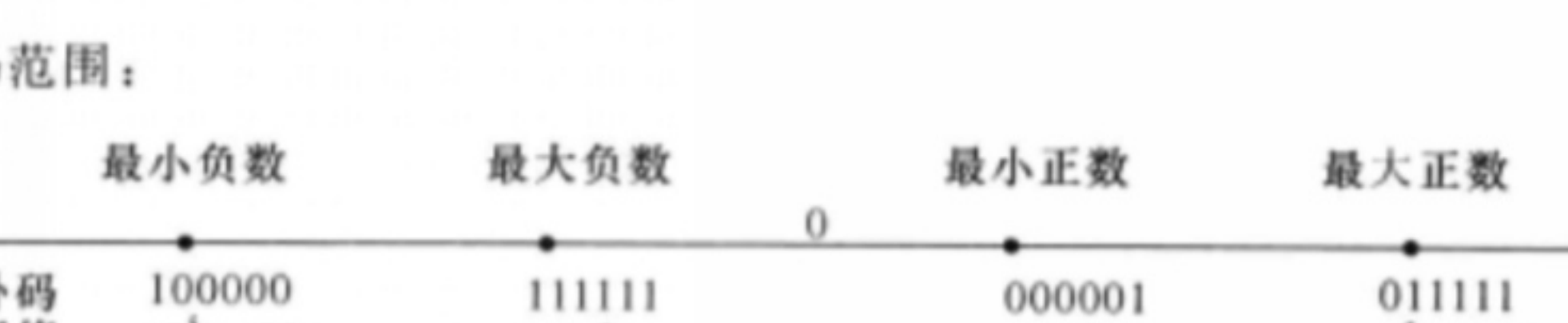
负数的尾数形式为 $1.0 \times \cdots \times$ 的形式。

3. 浮点数的表示范围

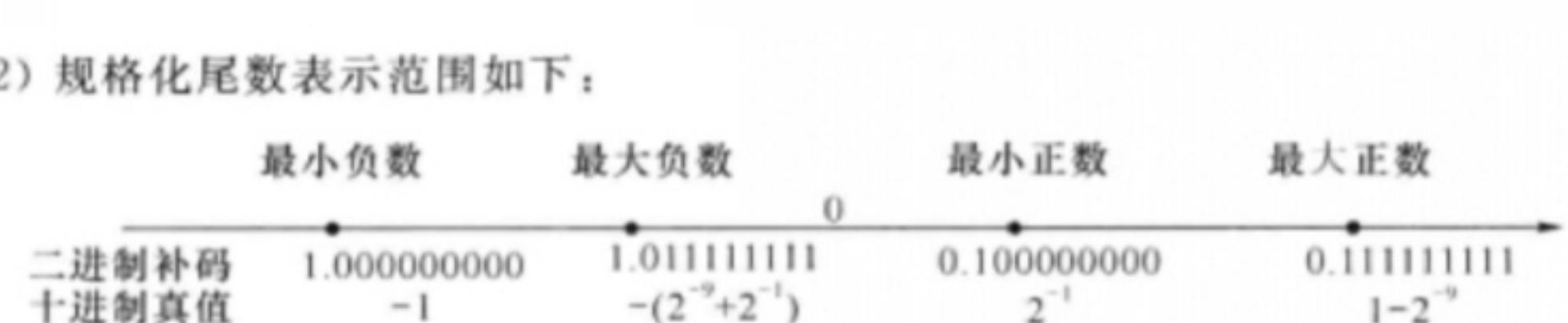
设浮点数的阶码6位(含符号位),尾数为10位(含符号位),分析其表示范围。

【分析】

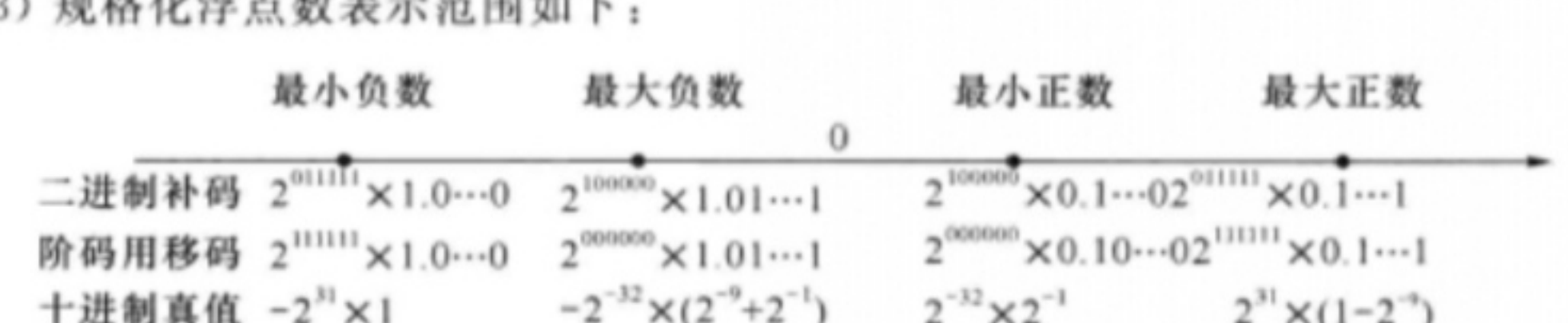
(1) 阶码范围:



(2) 规格化尾数表示范围如下:



(3) 规格化浮点数表示范围如下:



注意:这里规格化尾数的最大负数的补码是 $1.01 \cdots 1$ 的形式,而不是 $1.10 \cdots 0$ 的形式,是因为 $1.10 \cdots 0$ 不是规格化数,所以规格化尾数的最大负数应是 $-(0.10 \cdots 0 + 0.0 \cdots 01) = -0.10 \cdots 01$,而 $(-0.10 \cdots 01)_{\text{补}} = 1.01 \cdots 1$,即 $-(2^{-(n-1)}+2^{-n})$ 。

所以此时浮点数的表示范围为 $-2^{31} \times 1 \sim 2^{31} \times (1-2^{-9})$ 。

根据以上分析若某机字长为 $m+n$,其中阶码 m 位(含一位符号位),尾数 n 位(含一位符号位),设 $a=2^{m-1}-1$, $b=-2^{m-1}$,则规格化数所能表示的范围为

$$\begin{aligned} \text{最大正数} &= (1-2^{-(n-1)}) \times 2^a \\ \text{最小正数} &= 2^{-1} \times 2^b \\ \text{最大负数} &= -(2^{-(n-1)}+2^{-1}) \times 2^b \\ \text{最小负数} &= -1 \times 2^a \end{aligned}$$

例如:设字长32位,阶码8位(含一位符号位),尾数24位(含一位符号位),则 $a=2^7-1=127$, $b=-2^7=-128$,其规格化数所能表示数值的范围为

$$\begin{aligned} \text{最大正数} &= (1-2^{-23}) \times 2^{127} \\ \text{最小正数} &= 2^{-1} \times 2^{-128} \\ \text{最大负数} &= -(2^{-23}+2^{-1}) \times 2^{-128} \\ \text{最小负数} &= -1 \times 2^{127} \end{aligned}$$

浮点数的阶码决定了浮点数的表示范围;浮点数的尾数决定了浮点数的表示精度。

4. IEEE 754 浮点数标准

单精度数所表示的数值为 $(-1)^s \times 1.M \times 2^{e-127}$ 。

双精度数所表示的数值为 $(-1)^s \times 1.M \times 2^{e-1023}$ 。

其中:

$s=0$ 表示正数, $s=1$ 表示负数;

单精度数 e 的取值为 $1 \sim 254$ (8位表示), M 为23位,共32位;

双精度数 e 的取值为 $1 \sim 2046$ (11位表示), M 为52位,共64位。

5. 溢出问题

当一个浮点数阶码大于机器的最大阶码时,称为上溢;小于最小阶码时,称为下溢。

6. 浮点数的底数问题

浮点数的底一般为2,为了用相同位数的阶码表示更大范围的浮点数,在一些计算机中也有选用阶码的底为8或16的。此时浮点数 N 表示成:

$$N = 8^E \times M \quad \text{或} \quad N = 16^E \times M$$

当阶码以8为底时,只要尾数满足 $1/8 \leq M < 1$ 或 $-1 \leq M < -1/8$ 就是规格化数。执行对阶和规格化操作时,每当阶码的值增或减1,尾数要相应右移或左移3位。

当阶码以16为底时,只要尾数满足 $1/16 \leq M < 1$ 或 $-1 \leq M < -1/16$ 就是规格化数。执行对阶和规格化操作时,阶码的值增或减1,尾数必须移4位。

判为规格化数或实现规格化操作,均应使数值的最高3位(以8为底)或4位(以16为底)中至少有1位与符号位不同。

7. 隐藏位的含义

当浮点数的尾数的基值为2时,规格化的浮点数尾数的最高位一定是1(如果尾数用补码表示,规格化浮点数尾数的最高位一定与尾数符号位相反),所以浮点数在传送与存储过程中,尾数的最高位可以不表示出来,只在计算的时候才恢复这个隐藏位,或者对结果进行修正。这就是浮点数中的隐藏位的含义,用隐藏位可使浮点数的尾数多表示一位。