中山大學 软件工程学院

SUN YAT-SEN UNIVERSITY SCHOOL OF SOFTWARE ENGINEERING

计算机组成原理

授课老师：吴炜滨

大纲

- 进制转换
- 数据表示
- 无符号数
- 有符号数
- 数的定点表示和浮点表示

大纲

- 进制转换

进制转换

- 书写
  - 十进制：$(123)_{十}$
  - 二进制：1001B或$(1001)_{二}$
  - 十六进制：17DBH或$(17DB)_{十六}$

十进制数	二进制数	十六进制数	十进制数	二进制数	十六进制数
0	0 0 0 0 0	0	16	1 0 0 0 0	10
1	0 0 0 0 1	1	17	1 0 0 0 1	11
2	0 0 0 1 0	2	18	1 0 0 1 0	12
3	0 0 0 1 1	3	19	1 0 0 1 1	13
4	0 0 1 0 0	4	20	1 0 1 0 0	14
5	0 0 1 0 1	5	21	1 0 1 0 1	15
6	0 0 1 1 0	6	22	1 0 1 1 0	16
7	0 0 1 1 1	7	23	1 0 1 1 1	17
8	0 1 0 0 0	8	24	1 1 0 0 0	18
9	0 1 0 0 1	9	25	1 1 0 0 1	19
10	0 1 0 1 0	A	26	1 1 0 1 0	1A
11	0 1 0 1 1	B	27	1 1 0 1 1	1B
12	0 1 1 0 0	C	28	1 1 1 0 0	1C
13	0 1 1 0 1	D	29	1 1 1 0 1	1D
14	0 1 1 1 0	E	30	1 1 1 1 0	1E
15	0 1 1 1 1	F	31	1 1 1 1 1	1F

进制转换

- 任意一个数N可用下式表示
  - r: 基值；n、m：正整数，分别代表整数位和小数位的位数
  - d_i为系数，代表第i位的一个数码，可以是0~$(r-1)$数码中的任意一个
  - r^i为第i位的权数

$$N=(\mathrm{d}_{n-1}\mathrm{d}_{n-2}\cdots\mathrm{d}_1\mathrm{d}_0.\mathrm{d}_{-1}\mathrm{d}_{-2}\cdots\mathrm{d}_{-m})_r$$

$$=\mathrm{d}_{n-1}r^{n-1}+\mathrm{d}_{n-2}r^{n-2}+\cdots+\mathrm{d}_1r^1+\mathrm{d}_0r^0+\mathrm{d}_{-1}r^{-1}+\cdots+\mathrm{d}_{-m}r^{-m}$$

$$=\sum_{i=-m}^{n-1}d_ir^i$$

进制转换

- 二进制数转为十进制数
  - 按“权”展开法
    - $(11011.1)_{\text{二}} = 1\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1} = (27.5)_{\text{十}}$
- 十进制数转为二进制数
  - 重复相除（乘）法

进制转换

- 11.6 转二进制

$$(11.6)_{十} \approx (1011.1001)_{二}$$

整数部分除2取余数

11 ÷ 2 = 5 1	
5 ÷ 2 = 2 1	
2 ÷ 2 = 1 0	
1 ÷ 2 = 0 1	高位

直到商为0止

1011

小数部分乘2取整数

0.6 * 2 = 1.2	高位
0.2 * 2 = 0.4	
0.4 * 2 = 0.8	
0.8 * 2 = 1.6	

直到小数部分为0或满足精度要求的位数止

0.1001 1001...

大纲

- 数据表示

数据表示

数据表示

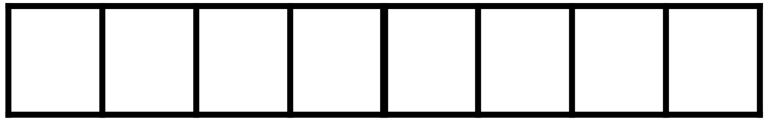
- 定义：能由计算机硬件直接识别和引用的数据类型，如定点数、浮点数等
- 在计算机系统中，有对这些数据类型进行操作的机器指令和功能部件
  - 这些数据类型及其运算需由硬件实现
- 所有数据类型中最常用、相对比较简单、用硬件实现比较容易的几种
  - 数据结构要通过软件映象变换成机器所具有的各种数据表示来实现
  - 数据表示和数据结构是计算机系统软、硬件的交界面

大纲

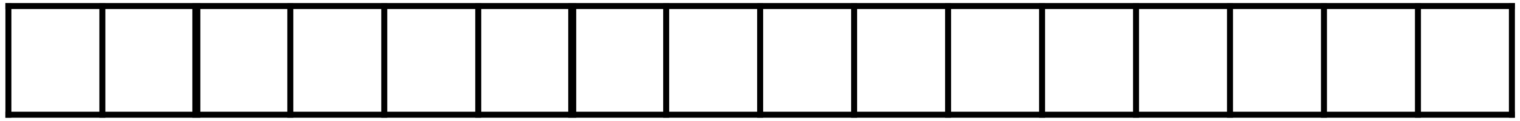
- 无符号数

无符号数

- 无符号数：没有正负号的数
- 在计算机中如何表示和存储?
  - 只有数值部分，将其转变成二进制
  - 如放在寄存器中，寄存器的位数（机器字长）反映了无符号数的表示范围



8 位 **0 （8个0）~ 255（8个1）**



16 位 **0（16个0） ~ 65535（16个1）**

大纲

- 有符号数
  - 机器数与真值
  - 原码表示法
  - 补码表示法
  - 反码表示法
  - 移码表示法

大纲

- 有符号数
  - 机器数与真值

机器数与真值

- 有符号数：有正负号的数
- 在计算机中如何表示和存储?
  - 数值部分、符号部分都保存到计算机中
- 真值：平时用的数据的真实的值，带有正负号
- 机器数：保存在计算机当中的数
- 如何将真值转换为机器数?
  - 符号数字化

机器数与真值

真值	机器数
带符号的数	符号数字化的数
+ 0.1011	0 | 1011
– 0.1011	1 | 1011
+ 1100	0 | 1100
– 1100	1 | 1100

小数点的位置：以约定的方式给出，不用硬件表示或存储

小数点的位置 → 小数定点机

小数点的位置

小数点的位置 → 整数定点机

大纲

- 有符号数
  - 原码表示法

原码定义

- 整数

$$[x]_{原} = \begin{cases} 0, \ x & 2^n > x \geq 0 \\ 2^n - x & 0 \geq x > -2^n \end{cases}$$

x 为真值　　n 为整数数值位的位数

如　$x = +1110$　　$[x]_{原} = 0,1110$

用逗号将符号位和数值部分隔开，方便阅读，不用存储

$x = -1110$　　$[x]_{原} = 2^4 + 1110 = 1,1110$

原码：带符号的绝对值表示

原码定义

- 小数

$$[x]_{原} = \begin{cases} x & 1 > x \geq 0 \\ 1 - x & 0 \geq x > -1 \end{cases}$$

x 为真值

如 $x = +0.1101$ $\quad [x]_{原} = 0.1101$

用小数点将符号位和数值部分隔开，方便阅读，不需存储

$x = -0.1101$ $\quad [x]_{原} = 1 - (-0.1101) = 1.1101$

$x = +0.1000000$ $\quad [x]_{原} = 0.1000000$

用小数点将符号位和数值部分隔开

$x = -0.1000000$ $\quad [x]_{原} = 1 - (-0.1000000) = 1.1000000$

原码表示法

- 真值变成原码
  - 整数：加一个符号位，正数加0负数加1，后面加一个逗号，数值部分照抄
  - 小数：小数点前面的那一位就用于表示数据的符号，正数用0来表示，负数用1来表示，后面加一个小数点，然后把小数点后面数值部分照写
- 原码是保存在计算机的数据
  - 位数有限，受限于计算机能保存的机器数的长度

举例

- 已知$[x]_{原} = 1.0011$　　求 x　　-0.0011

  解:　由定义得

  $$x = 1 - [x]_{原} = 1 - 1.0011 = -0.0011$$

- 已知$[x]_{原} = 1{,}1100$　　求 x　　$-\ 1100$

  解:　由定义得

  $$x = 2^4 - [x]_{原} = 10000 - 1{,}1100 = -1100$$

- 已知 $[x]_{原} = 0.1101$ 求 x

解： 根据定义 $\because [x]_{原} = 0.1101$

$\therefore \quad x = +0.1101$

- 求 $x = 0$ 的原码

解: 设 $x = +0.0000$ $\quad [+0.0000]_{原} = 0.0000$

$x = -0.0000$ $\quad [-0.0000]_{原} = 1.0000$

同理，对于整数 $\quad [+0]_{原} = 0,0000$

$[-0]_{原} = 1,0000$

$\therefore \quad [+0]_{原} \neq [-0]_{原}$

原码的特点

- 简单、直观
  - 但是用原码作加法时，会出现如下问题：

要求	数1	数2	实际操作	结果符号
加法	正	正	加	正
加法	正	负	减	可正可负
加法	负	正	减	可正可负
加法	负	负	加	负

- 能否只作加法？
  - 找到一个与负数等价的正数，来代替这个负数，就可使减→加

大纲

- 有符号数
  - 补码表示法

补数

- 时钟：六点调整到三点

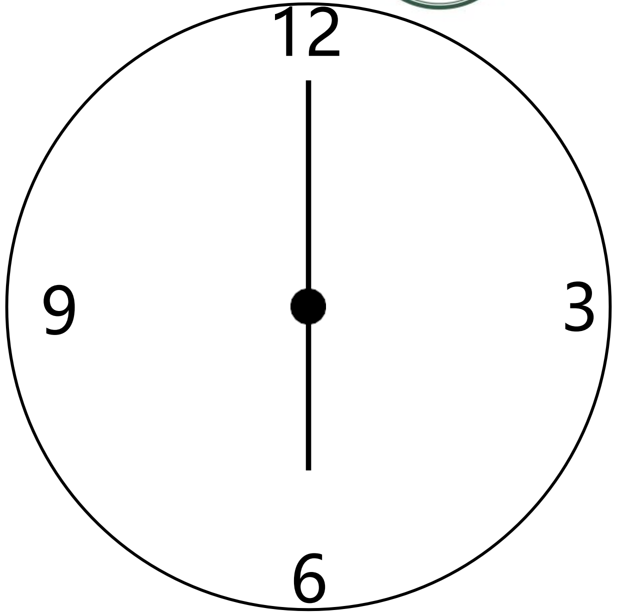
逆时针

$$\begin{array}{r} 6 \\ -3 \\ \hline 3 \end{array}$$

顺时针

$$\begin{array}{r} 6 \\ +9 \\ \hline 15 \\ -12 \\ \hline 3 \end{array}$$

时钟以12为模



可见-3可用$+9$代替　　减法 ⟶ 加法

称$+9$是-3以12为模的补数

记作 $-3 \equiv +9 \pmod{12}$

补数

- $-3 \equiv +9 \pmod{12}$
  - 同理
    - $-4 \equiv +8 \pmod{12}$
    - $-5 \equiv +7 \pmod{12}$
    - $3 \equiv 15 \equiv 27 \pmod{12}$
      - 即：$3 \equiv 3+12 \equiv 3+24 \equiv 3 \pmod{12}$
- 结论
  - 一个负数加上 **模** 即得该负数的补数
  - 一个正数和一个负数互为补数时它们绝对值之和即为 **模** 数
  - 正数的补数即为其本身

机器中的补数

- 寄存器：位数是四位（模 16 ）

$$1011 \longrightarrow 0000\ ?$$

$$\begin{array}{r} 1011 \\ -\,1011 \\ \hline 0000 \end{array} \qquad \begin{array}{r} 1011 \\ +\,0101 \\ \hline \boxed{1}0000 \end{array}$$

自然去掉

可见 -1011 可用$+\,0101$ 代替

记作 $-1011 \equiv +0101 \pmod{2^4}$

同理 $-011 \equiv +101 \pmod{2^3}$

$-0.1001 \equiv +1.0111 \pmod{2}$

机器数的补码表示

两个互为补数的数 $\boxed{-1011 \equiv +0101}$ （mod 2^4）

分别加上模 $+10000$ $+10000$

结果仍互为补数 $+0101 \equiv +\boxed{1}0101$

丢掉

$\therefore +0101 \equiv +0101$ （mod 2^4）

可见 $+0101 \longrightarrow +0101$

?

$\longrightarrow -1011$

? $\boxed{0},0101 \longrightarrow +\ 0101$

? $\boxed{1},0101 \longrightarrow -\ 1011$

（mod 2^{4+1}）

$2^{4+1} - 1011 = 100000$

$$\begin{array}{r} 100000 \\ -1011 \\ \hline 1,0101 \end{array}$$

用 逗号 将符号位和数值部分隔开

补码定义

- 整数

$$[x]_{补} = \begin{cases} 0,\ x & 2^n > x \geq 0 \\ 2^{n+1} + x & 0 > x \geq -2^n \quad (\bmod 2^{n+1}) \end{cases}$$

x 为真值　　　　n 为整数数值位的位数

如　　$x = +1010$　　　　　　$x = -1011000$

$[x]_{补} = 0,1010$　　　　$[x]_{补} = 2^{7+1} + (-1011000)$

$$\begin{array}{r} = 100000000 \\ - \quad 1011000 \\ \hline 1,0101000 \end{array}$$

用逗号将符号位和数值部分隔开

补码定义

- 小数

$$[x]_{补} = \begin{cases} x & 1 > x \geq 0 \\ 2 + x & 0 > x \geq -1 \pmod 2 \end{cases}$$

x 为真值

如　$x = +0.1110$　　$x = -0.1100000$

$[x]_{补} = 0.1110$

$$\begin{aligned} [x]_{补} &= 2 + (-0.1100000) \\ &= 10.0000000 \\ &- \ \ 0.1100000 \\ &\overline{\ \ 1.0100000\ \ } \end{aligned}$$

用小数点将符号位

和数值部分隔开

求补码的快捷方式

- 设 $x = -1010$ 时

$$[x]_{补} = 2^{4+1} - 1010 = 11111 + 1 - 1010$$

$$\begin{array}{r} = 100000 \\ -1010 \\ \hline = 1,0110 \end{array} \qquad \begin{array}{r} = 11111 + 1 \\ -1010 \\ \hline \boxed{10101} + 1 \\ = 1,0110 \end{array}$$

$$又[x]_{原} = \boxed{1,1010}$$

- 当真值为**负**时
  - 补码可用原码除符号位外，每位取反，末位加 1 求得（**求反加1**）

求补码的快捷方式

- 当真值为**负**时
  - 补码可用原码除符号位外，每位取反，末位加 1 求得（**求反加1**）
- 当真值为**正**时
  - 原码 = 补码

- 已知 $[x]_{补} = 1,1110$ 求 x

解： 由定义得

$$x = [x]_{补} - 2^{4+1}$$
$$= 1,1110 - 100000$$
$$= - 0010$$

$$[x]_{补} \xrightarrow{?} [x]_{原}$$

$$[x]_{原} = 1,0010$$

$$\therefore x = -0010$$

- 当真值为**负**时
  - 原码可用补码除符号位外，每位取反，末位加 1 求得（**求反加1**）

- 已知$[x]_{补} = 0.0001$　求x

解：由定义得　$x = +0.0001$

- 已知$[x]_{补} = 1.0001$　求x

解：由定义得

$$x = [x]_{补} - 2$$
$$= 1.0001 - 10.0000$$
$$= -0.1111$$

$[x]_{补} \rightarrow [x]_{原}$

$[x]_{原} = 1.1111$

$\therefore x = -0.1111$

求下列真值的补码

真值		$[x]_补$	$[x]_原$
$x = +70 = 1000110$		0,1000110	0,1000110
$x = -70 = -1000110$		1, 0111010	1,1000110
$x = 0.1110$		0.1110	0.1110
$x = -0.1110$		1.0010	1.1110
$x = \boxed{0.0000}$	$[+0]_补 = [-0]_补$	$\boxed{0.0000}$	0.0000
$x = \boxed{-0.0000}$		$\boxed{0.0000}$	1.0000
$x = -1.0000$		1.0000	不能表示

由小数补码定义 $[x]_补 = \begin{cases} x & 1 > x \geq 0 \\ 2+x & 0 > x \geq -1 \pmod 2 \end{cases}$

$$[-1]_补 = 2 + x = 10.0000 - 1.0000 = 1.0000$$

大纲

- 有符号数
  - 反码表示法

反码定义

■ 整数

$$[x]_{反} = \begin{cases} 0, \quad x & 2^n > x \geq 0 \\ (2^{n+1} - 1) + x & 0 \geq x > -2^n \ (\mathrm{mod}\ 2^{n+1} - 1) \end{cases}$$

x 为真值 n 为整数数值位的位数

如 $x = +1101$ $x = -1101$

$[x]_{反} = 0,1101$

$$[x]_{反} = (2^{4+1} - 1) - 1101 = 11111 - 1101 = 1,0010$$

用逗号将符号位和数值部分隔开

反码定义

- 小数

$$[x]_{反} = \begin{cases} x & 1 > x \geq 0 \\ (2 - 2^{-n}) + x & 0 \geq x > -1 \pmod{2 - 2^{-n}} \end{cases}$$

x 为真值　　　　n 为小数的位数

如　$x = +0.1101$　　　　$x = -0.1010$

$[x]_{反} = 0.1101$

$$\begin{aligned}[x]_{反} &= (2 - 2^{-4}) - 0.1010 \\ &= 1.1111 - 0.1010 \\ &= 1.0101\end{aligned}$$

用小数点将符号位和数值部分隔开

反码定义

- 已知$[x]_{反} = 0,1110$　　求 x

  解：由定义得　　$x = +1110$

- 已知$[x]_{反} = 1,1110$　　求x

  解：由定义得　　$x = [x]_{反} - (2^{4+1} - 1)$

  $= 1,1110 - 11111$

  $= -0001$

反码定义

- 求0的反码

解： 设 $x = +0.0000$　　$[+0.0000]_{反} = 0.0000$

$x = -0.0000$　　$[-0.0000]_{反} = 1.1111$

同理，对于整数

$[+0]_{反} = 0,0000$　　$[-0]_{反} = 1,1111$

$\therefore\ [+0]_{反} \neq [-0]_{反}$

三种机器数的小结

- 最高位为符号位，书写上用“，”（整数）或“.”（小数）将数值部分和符号位隔开
- 对于正数，原码= 补码 = 反码（考虑机器字长的限制）
- 对于负数，符号位为 1，其数值部分
  - 原码：数值部分和真值相同
  - 补码：原码除符号位外每位取反末位加 1
  - 反码：原码除符号位外每位取反

■ 设机器数字长为 8 位（其中 1 位为符号位），对于整数，当其分别代表无符号数、原码、补码和反码时，对应的真值范围各为多少?

二进制代码	无符号数对应的真值	原码对应的真值	补码对应的真值	反码对应的真值
00000000	0	+0	±0	+0
00000001	1	+1	+1	+1
00000010	2	+2	+2	+2
⋮	⋮	⋮	⋮	⋮
01111111	127	+127	+127	+127
10000000	128	-0	-128	-127
10000001	129	-1	-127	-126
⋮	⋮	⋮	⋮	⋮
11111101	253	-125	-3	-2
11111110	254	-126	-2	-1
11111111	255	-127	-1	-0

■ 已知 $[y]_{补}$，求$[-y]_{补}$

解：设 $[y]_{补} = y_0 . y_1 y_2 \dots y_n$

(1) $[y]_{补} = 0. y_1 y_2 \dots y_n$

$$y = 0. y_1 y_2 \dots y_n$$

$$-y = -0. y_1 y_2 \dots y_n$$

$$[-y]_{补} = 1. \overline{y_1} \overline{y_2} \dots \overline{y_n} + 2^{-n}$$

$[y]_{补}$连同符号位在内，每位取反，末位加1

即得$[-y]_{补}$

(2) $[y]_{补} = 1. y_1 y_2 \dots y_n$

$$[y]_{原} = 1. \overline{y_1} \overline{y_2} \dots \overline{y_n} + 2^{-n}$$

$$y = -(0. \overline{y_1} \overline{y_2} \dots \overline{y_n} + 2^{-n})$$

$$-y = 0. \overline{y_1} \overline{y_2} \dots \overline{y_n} + 2^{-n}$$

$$[-y]_{补} = 0. \overline{y_1} \overline{y_2} \dots \overline{y_n} + 2^{-n}$$

$[y]_{补}$连同符号位在内，每位取反，末位加 1

即得$[-y]_{补}$

大纲

- 有符号数
  - 移码表示法

移码表示法

- 补码表示很难直接判断其真值大小

如 十进制	二进制	补码	
$x = +21$	$+10101$	$0,10101$	错
$x = -21$	-10101	$1,01011$	大
$x = +31$	$+11111$	$0,11111$	错
$x = -31$	-11111	$1,00001$	大

$x + 2^5$

$+10101 + 100000 \quad = 1,10101$ 大

$-10101 + 100000 \quad = 0,01011$ 正确

$+11111 + 100000 \quad = 1,11111$ 大

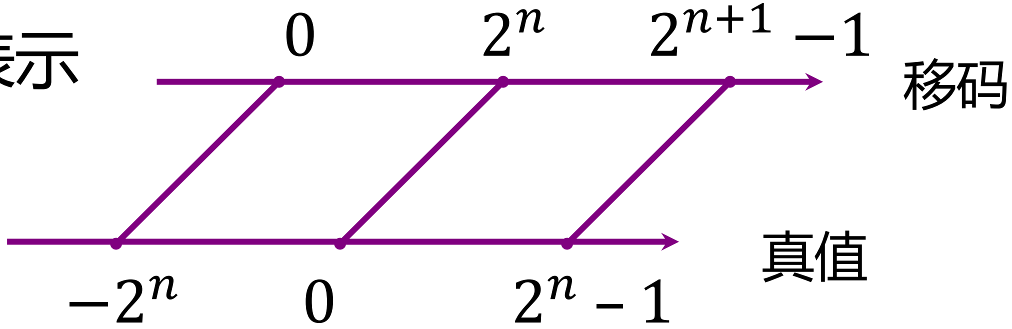
$-11111 + 100000 \quad = 0,00001$ 正确

移码

- 定义 $[x]_{移} = 2^n + x \quad (2^n > x \geq -2^n)$

x 为真值，n 为整数数值位的位数

移码在数轴上的表示



如 $\quad x = 10100$

$[x]_{移} = 2^5 + 10100 \quad = 1,10100$

$x = -10100$

$[x]_{移} = 2^5 - 10100 \quad = 0,01100$

用 逗号 将符号位和数值部分隔开

移码

- 注意
  - 真值转换成移码时，不区分对待正数和负数
  - 只有整数形式的定义，没有小数形式的定义
    - 移码在计算机的数据表示中的作用有关
    - 移码的大小很好判断，通常用来表示浮点数据表示的阶码部分，阶码都是整数

移码和补码的比较

- 设 $x = +1100100$

  $[x]_{移} = 2^7 + 1100100 \quad = 1,1100100$

  $[x]_{补} = 0,1100100$

- 设 $x = -1100100$

  $[x]_{移} = 2^7 - 1100100 \quad = 0,0011100$

  $[x]_{补} = 1,0011100$

- 同一真值的补码与移码只差一个符号位，数值部分完全相同

真值、补码和移码的对照表

真值 x (n=5)	$[x]_{补}$	$[x]_{移}$	$[x]_{移}$对应的十进制整数
- 1 0 0 0 0 0	1 0 0 0 0 0	0 0 0 0 0 0	0
- 1 1 1 1 1	1 0 0 0 0 1	0 0 0 0 0 1	1
- 1 1 1 1 0	1 0 0 0 1 0	0 0 0 0 1 0	2
⋮	⋮	⋮	⋮
- 0 0 0 0 1	1 1 1 1 1 1	0 1 1 1 1 1	31
± 0 0 0 0 0	0 0 0 0 0 0	1 0 0 0 0 0	32
+ 0 0 0 0 1	0 0 0 0 0 1	1 0 0 0 0 1	33
+ 0 0 0 1 0	0 0 0 0 1 0	1 0 0 0 1 0	34
⋮	⋮	⋮	⋮
+ 1 1 1 1 0	0 1 1 1 1 0	1 1 1 1 1 0	62
+ 1 1 1 1 1	0 1 1 1 1 1	1 1 1 1 1 1	63

移码的特点

- 当 $x = 0$， $n = 5$ 时

$$[+0]_{移} = 2^5 + 0 \quad = 1,00000$$

$$[-0]_{移} = 2^5 - 0 \quad = 1,00000$$

$$\therefore [+0]_{移} = [-0]_{移}$$

- 当 $n = 5$ 时

$$\text{最小的真值为 } -2^5 \quad = \quad -100000$$

$$[-100000]_{移} = 2^5 - 100000 \quad = \quad 000000$$

- 可见：最小真值的移码为全 0
  - 用移码表示浮点数的阶码，能方便地判断浮点数的阶码大小

大纲

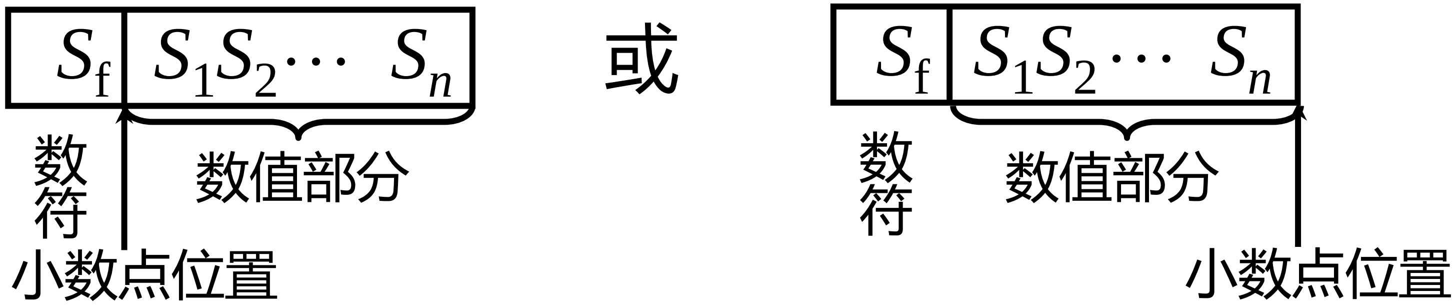
- 数的定点表示和浮点表示
  - 定点表示
  - 浮点表示

大纲

- 数的定点表示和浮点表示
  - 定点表示

定点表示

- 小数点按约定方式标出，计算机中不表示或存储小数点



	定点机	小数定点机	整数定点机
表示范围	原码	$-(1-2^{-n}) \sim +(1-2^{-n})$	$-(2^n-1) \sim +(2^n-1)$
	补码	$-1 \sim +(1-2^{-n})$	$-2^n \sim +(2^n-1)$
	反码	$-(1-2^{-n}) \sim +(1-2^{-n})$	$-(2^n-1) \sim +(2^n-1)$

有限个（2^{n+1}）离散的数表示无穷个实数

大纲

- 数的定点表示和浮点表示
  - 浮点表示

浮点表示

- 为什么在计算机中要引入浮点数表示?
  - 早期计算机因为硬件技术的限制，只有定点表示方式
  - 科学计算过程当中经常会用到浮点数，在定点机中，需要程序员通过设定比例因子来调节小数点的位置，使其变为纯小数或纯整数后进行运算
    - 增加编程难度

$$101.01 + 0.101$$

$$= (0.101010 + 0.000101) \times 2^3 \quad \text{(缩小)}$$

$$= (0.101111) \times 2^3 \quad \text{(运算)}$$

$$= 101.111 \quad \text{(还原)}$$

浮点表示

- 为什么在计算机中要引入浮点数表示?
  - 数的表示范围小，为了能表示两个大小相差很大的数据，需要很长的机器字长
    - 例如：太阳的质量是0.2×10^{34}克，一个电子的质量大约为0.9×10^{-27}克，两者相差约10^{61}倍，若用定点数据表示：$2^x>10^{61}$，得到$x\geq203$位
  - 数据存储单元的利用率往往很低

浮点表示

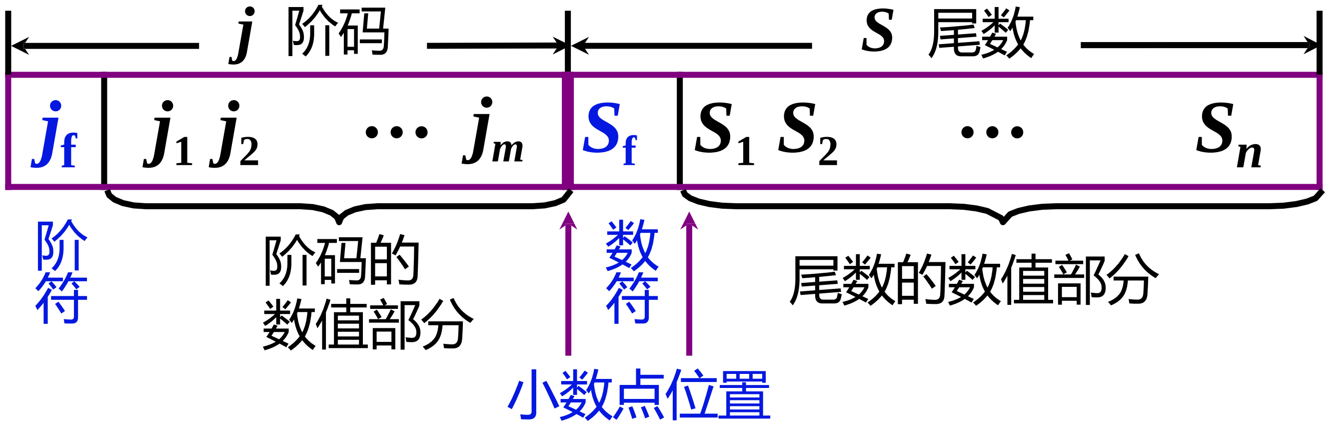
- 浮点数的一般形式： $N = S \times r^j$
  - S：尾数，j：阶码，r：尾数的基值
  - 计算机中：r 取 2、4、8、16 等；S：小数定点形式表示，绝对值小于等于1，可正可负；j： 整数，可正可负
- 当 r = 2

$$
\begin{aligned}
N &= 11.0101 \\
&= 0.110101 \times 2^{\boxed{10}} \quad ✓ \\
&= 1.10101 \times 2^{1} \\
&= 1101.01 \times 2^{-10} \\
&= 0.00110101 \times 2^{100} \quad ✓
\end{aligned}
$$

二进制表示

规格化数

浮点数的存储和表示



S_f	代表浮点数的符号
n	其位数反映浮点数的精度
m	其位数反映浮点数的表示范围
阶码	决定小数点的实际位置

浮点数的表示范围

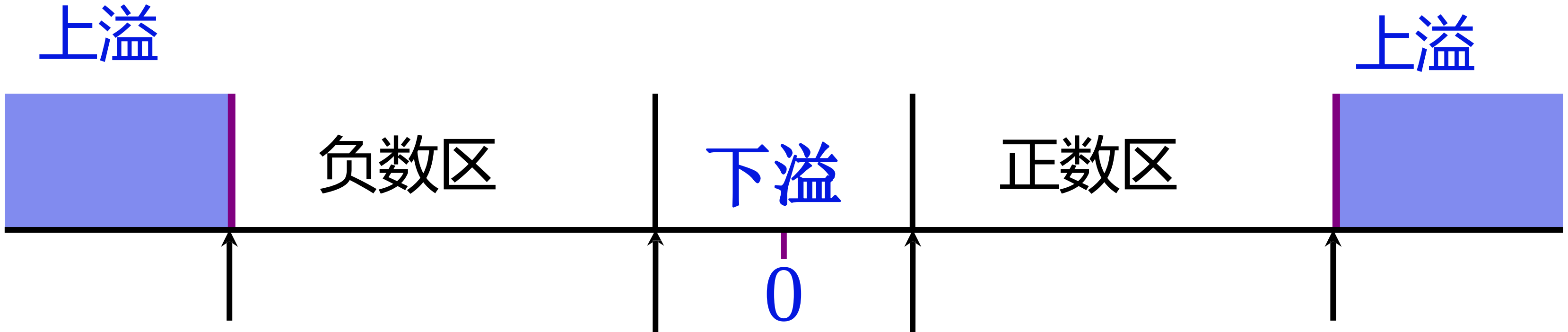
- 不考虑数据的规格化
- 无论是尾数还是阶码，都用原码形式进行表示
- 阶码的数值位取m位，尾数的数值位取n位

j 阶码		S 尾数	
j_f	$j_1\ j_2\ \cdots\ j_m$	S_f	$S_1\ S_2\ \cdots\ S_n$

j 阶码			S 尾数		
j_f	$j_1\ j_2$	$\cdots\ j_m$	S_f	$S_1\ S_2$	$\cdots\ S_n$

上溢：浮点数阶码大于最大阶码时，进行中断溢出处理

下溢：浮点数阶码小于最小阶码时，按机器零处理



最小负数

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

$$-2^{15} \times (1-2^{-10})$$

最大负数

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

$$-2^{-15} \times 2^{-10}$$

最小正数

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

$$2^{-15} \times 2^{-10}$$

最大正数

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

$$2^{15} \times (1-2^{-10})$$

设 $m = 4$

$n = 10$

可表示范围：负数区+正数区+0

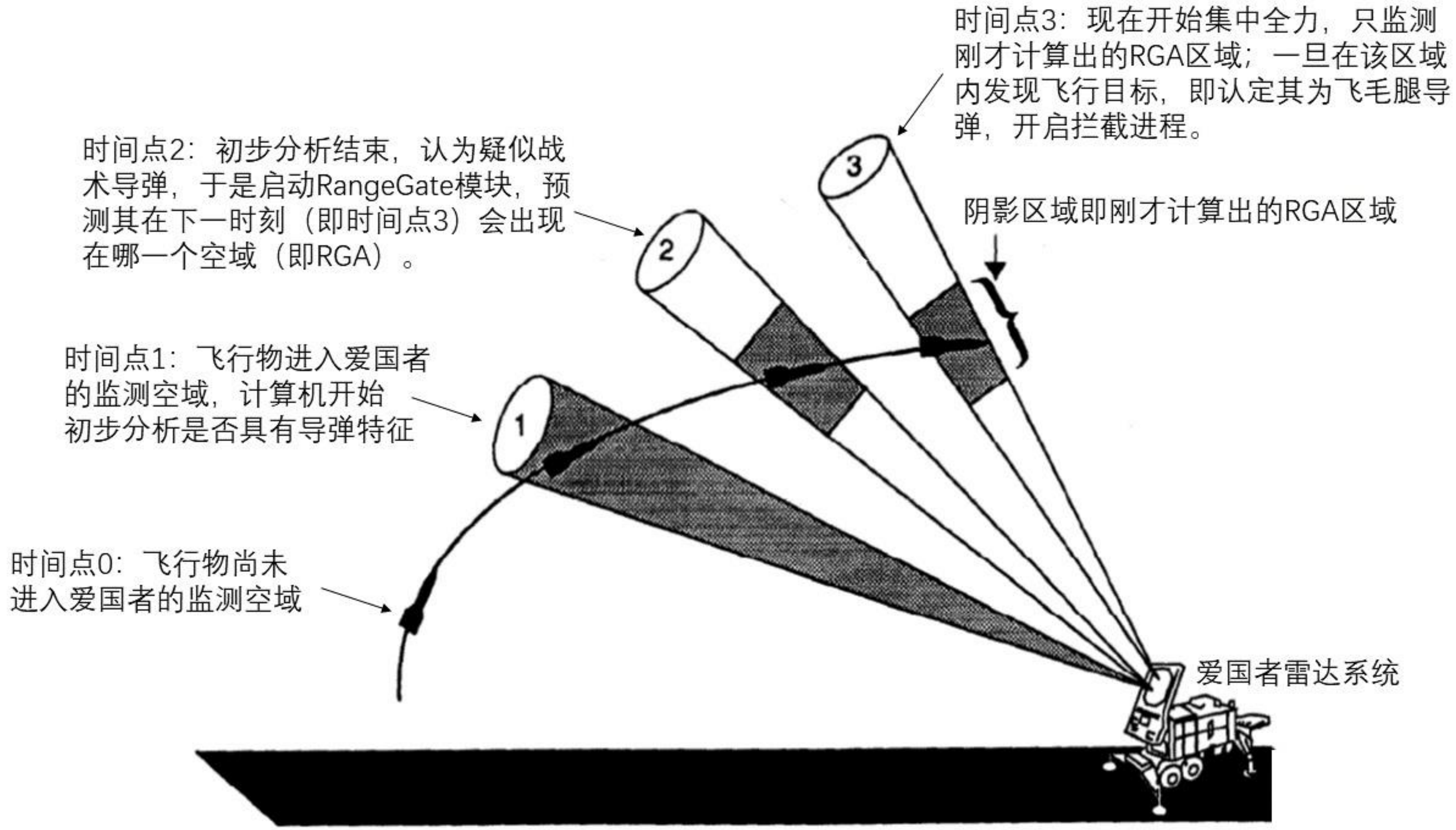
（2^{m+n+2}个离散的数表示无穷个实数）

机器零

- 当浮点数尾数为0 时，不论其阶码为何值按机器零处理
- 当浮点数阶码小于它所表示的最小数时，不论尾数为何值，按机器零处理

浮点数的精度问题

- 海湾战争中，爱国者拦截飞毛腿失败
  - 爱国者系统会计算出下一时刻该导弹最有可能出现在哪一个空域



浮点数的精度问题

- 海湾战争中，爱国者拦截飞毛腿失败
  - 爱国者系统中内置了一个时间计数器，每隔 0.1 秒就将计数器增加1
  - 0.1 这个十进制数字如果用计算机的二进制来表示，将是一个无限循环小数
  - 而爱国者使用的寄存器只有24位，所以只能截取这个无限循环小数的前23位，也就是 0.00011001100110011001100（大概是十进制的 0.0999999046）
  - 由于累积的时间误差，使得预测出错，导致了拦截失败
  - 失之毫厘，谬以千里：**注意浮点数的精度问题**

练习

- 设机器数字长为 24 位，欲表示±3万的十进制数，试问在保证数的最大精度的前提下，除阶符、数符各取1 位外，阶码、尾数各取几位？

解： ∵ $2^{14} = 16384$ $2^{15} = 32768$

∴ 如果是定点数，**15** 位二进制数可反映 ±3 万之间的十进制数

$$2^{15} \times 0.\underbrace{\times\times\times \cdots \times\times\times}_{n\text{位}}$$

$$m = 4, 5, 6, \cdots$$

为满足最大精度，可取 $m = 4$，$n = 18$

浮点数的规格化形式

- 浮点数的一般形式： $N = S \times r^j$
  - S：尾数，j：阶码，r：尾数的基值
  - 计算机中：r 取 2、4、8、16 等；S：小数定点形式表示，绝对值小于等于1，可正可负；j： 整数，可正可负
- 为何要进行规格化?
  - 尽可能保证数据的精度：机器字长有限，避免存储不必要的0，使有效的位数尽可能多

浮点数的规格化形式

- 规格化形式是什么?
  - r = 2，尾数最高位真值为 1
  - r = 4，尾数最高 2 位真值不全为 0
  - r = 8，尾数最高 3 位真值不全为 0
- 基值的影响
  - 基值不同，浮点数的规格化形式不同
  - 基值越大，可表示的浮点数的范围越大，浮点数的精度降低

浮点数的规格化

- 如何进行规格化?
  - 通过对尾数进行移动，使其变为规格化形式
  - $r = 2$
    - 左规：尾数左移 1 位，阶码减 1
    - 右规：尾数右移 1 位，阶码加 1
  - $r = 4$
    - 左规：尾数左移 2 位，阶码减 1
    - 右规：尾数右移 2 位，阶码加 1
  - $r = 8$
    - 左规：尾数左移 3 位，阶码减 1
    - 右规：尾数右移 3 位，阶码加 1

浮点数的表示范围

- 无论是尾数还是阶码，都用原码形式进行表示
- 阶码的数值位取m位，尾数的数值位取n位
  - 设 $m = 4$，$n = 10$，$r = 2$，尾数规格化后的浮点数表示范围

j 阶码		S 尾数	
j_f	$j_1\ j_2\ \cdots\ j_m$	S_f	$S_1\ S_2\ \cdots\ S_n$

■ 设 $m = 4$，$n = 10$，$r = 2$，尾数规格化后的浮点数表示范围

最大正数 $2^{+1111} \times 0.\underbrace{1111111111}_{10\text{个}1} = 2^{15} \times (1-2^{-10})$

最小正数 $2^{-1111} \times 0.1\underbrace{000000000}_{9\text{个}0} = 2^{-15} \times 2^{-1} = 2^{-16}$

最大负数 $2^{-1111} \times (-0.1\underbrace{000000000}_{9\text{个}0}) = -2^{-15} \times 2^{-1} = -2^{-16}$

最小负数 $2^{+1111} \times (-0.\underbrace{1111111111}_{10\text{个}1}) = -2^{15} \times (1-2^{-10})$

■ 将 $+\frac{19}{128}$写成在定点机和浮点机中的三种机器数形式。其中数值部分均取10位，数符取1位，浮点数阶码取5位（含1位阶符），尾数规格化。

解：设 $x=\frac{19}{128}$

二进制形式	$x=10011/10000000=0.0010011$
定点表示	$x=0.0010011$
浮点规格化形式	$x=0.10011\times 2^{-10}$

定点机中 $[x]_{原}=[x]_{补}=[x]_{反}=0.0010011\mathbf{000}$

浮点机中

$[x]_{原}=1,0010;\quad 0.1001100000$

$[x]_{补}=1,1110;\quad 0.1001100000$

$[x]_{反}=1,1101;\quad 0.1001100000$

■ 将 –58 写成在定点机和浮点机中的三种机器数及阶码为移码、尾数为补码的形式。其中数值部分均取10位，数符取1位，浮点数阶码取5位（含1位阶符），尾数规格化。

解： 设 $x = -58$

二进制形式 $x = -111010$

定点表示 $x = -111010$

浮点规格化形式 $x = -0.11101 \times 2^{110}$

定点机中

$[x]_{原} = 1,0000111010$

$[x]_{补} = 1,1111000110$

$[x]_{反} = 1,1111000101$

浮点机中

$[x]_{原} = 0,0110;1.1110100000$

$[x]_{补} = 0,0110;1.0001100000$

$[x]_{反} = 0,0110;1.0001011111$

$[x]_{阶移、尾补} = 1,0110;1.0001100000$

谢谢！