

# 计算机组成原理

## (第六讲-1)



---

厦门大学信息学院软件工程系 曾文华  
2021年4月28日



# 第3篇 中央处理器

---

## 第 6 章 计算机的运算方法

## 第 7 章 指令系统

## 第 8 章 CPU 的结构和功能



# 第 6 章 计算机的运算方法

共81页

6.1 无符号数和有符号数

6.2 数的定点表示和浮点表示

6.3 定点运算

6.4 浮点四则运算

6.5 算术逻辑单元 (ALU)



## 6.1 无符号数和有符号数

一、无符号数

二、有符号数

1、原码

2、补码

3、反码

4、移码

# 一、无符号数

寄存器的位数，反映无符号数的表示范围

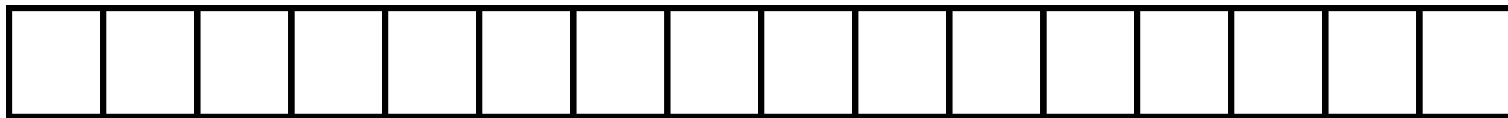


8 位      表示范围：  $0 \sim 255(2^8-1)$

二进制	十六进制	十进制
00000000	00H	0
00000001	01H	1

.....

11111110	FEH	254
11111111	FFH	255



16 位      表示范围：  $0 \sim 65535(2^{16}-1)$

二进制	十六进制	十进制
0000000000000000	0000H	0
0000000000000001	0001H	1

.....

1111111111111110	FFFEH	65534
1111111111111111	FFFFH	65535

## 二、有符号数

$\frac{1}{2}$     $\frac{1}{4}$     $\frac{1}{8}$     $\frac{1}{16}$

$0.5+0+0.125+0.0625$

### 1. 机器数与真值

真值

带符号的数

+ 0.1011

- 0.1011

+ 1100

- 1100

机器数

符号数字化的数

0 | 1011

+0.6875

小数点的位置

1 | 1011

-0.6875

小数点的位置

0 | 1100

+12

小数点的位置

1 | 1100

-12

小数点的位置

## 2. 原码表示法

### (1) 定义

整数

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

真值

如  $x = +1110$

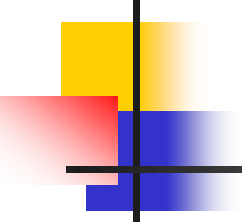
$$[x]_{\text{原}} = 0, 1110$$

$x = -1110$

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

“,” 表示整数

## 小数


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

## 真值

如  $x = +0.1101$

$$[x]_{\text{原}} = 0.1101$$

“.” 表示小数

$$x = -0.1101$$

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

$$x = +0.1000000$$

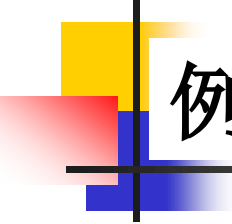
$$[x]_{\text{原}} = 0.1000000$$

$$x = -0.1000000$$

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



## (2) 举例



例 已知  $[x]_{\text{原}} = 1.0011$  求  $x$  ?

---

解：由定义得

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

例 已知  $[x]_{\text{原}} = 1,1100$  求  $x$  ?

解：由定义得

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

例 已知  $[x]_{\text{原}} = \mathbf{0.1101}$  求  $x$  ?

解: 根据定义  $\because [x]_{\text{原}} = \mathbf{0.1101}$

$$\therefore x = + \mathbf{0.1101}$$

例 求  $x = 0$  的原码

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

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

“0”的  
原码有  
二个

同理, 对于整数

$$[+0]_{\text{原}} = \mathbf{0,0000}$$

$$[-0]_{\text{原}} = \mathbf{1,0000}$$

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

- **8位二进制数整数**

- 原码的表示范围是： **-127 ~ +127**

真值	原码
-127	1,1111111
-126	1,1111110
.....	
-1	1,0000001
-0	1,0000000
+0	0,0000000
+1	0,0000001
.....	
+126	0,1111110
+127	0,1111111

- **8位二进制数小数**

- 原码的表示范围是： **-127/128 ~ +127/128**

真值	原码
-127/128	1.1111111
-126/128	1.1111110
.....	
-1/128	1.0000001
-0	1.0000000
+0	0.0000000
+1/128	0.0000001
.....	
+126/128	0.1111110
+127/128	0.1111111

### 3. 补码表示法



#### (1) 补数的概念

将时钟从6点拨回到3点有两种方法

时钟

逆时针

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

顺时针

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

可见  $-3$  可用  $+9$  代替


称  $+9$  是  $-3$  以 12 为模的补数

记作  $-3 \equiv +9 \quad (-3 + 12) \quad (\text{mod } 12)$

一个负数加上“模”即得该负数的补数

#### (2) 正数的补数即为其本身

### (3) 补码定义



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

正数

负数

正数

如

$$x = +1010$$

$$[x]_{\text{补}} = 0,1010$$

负数

$$x = -1011000 \quad -88$$

$$\begin{aligned} [x]_{\text{补}} &= 2^{7+1} + (-1011000) \\ &= 100000000 \\ &\quad - 1011000 \end{aligned}$$

$$[x]_{\text{补}} = 1,0101000 \quad 40$$

$$-88 + 128 = 40$$

# 小数

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

正数

如

$$x = + 0.1110$$

$$[x]_{\text{补}} = 0.1110$$

负数

$$x = - 0.1100000 \quad -0.75$$

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

$$[x]_{\text{补}} = 1.0100000 \quad 0.25$$

$$0.25 = -0.75 + 1$$

#### (4) 求补码的快捷方式

设  $x = -1010$

$$\begin{aligned} \text{则 } [x]_{\text{补}} &= 2^{4+1} - 1010 = 11111 + 1 - 1010 \\ &= 100000 = 11111 + 1 \\ &\quad - 1010 \\ &\quad \hline &= 1,0110 \end{aligned}$$
$$\begin{aligned} &= 11111 + 1 \\ &\quad - 1010 \\ &\quad \hline &= 10101 + 1 \\ &= 1,0110 \end{aligned}$$

$$[x]_{\text{原}} = 1,1010 \quad 1010 \text{取反为 } 0101, \quad 0101+1=0110$$

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

负数的补码为“取反加1”

## (5) 举例

正数的补码即为其本身

负数的补码为“取反加1”

例 已知  $[x]_{\text{补}} = 0.0001$  求  $x$ ?

解：由定义得  $x = + 0.0001$

例 已知  $[x]_{\text{补}} = 1.0001$  求  $x$ ?

解：由定义得

$$\begin{aligned} x &= [x]_{\text{补}} - 2 \\ &= 1.0001 - 10.0000 \\ &= -0.1111 \end{aligned}$$

取反加1

$$\begin{aligned} [x]_{\text{补}} &= 1.0001 \\ \text{取反: } &1110 \\ \text{加1: } &1110 + 1 = 1111 \end{aligned}$$



例 6.7 已知  $[x]_{\text{补}} = 1,1110$  求  $x$ ?

解：由定义得

$$\begin{aligned} x &= [x]_{\text{补}} - 2^{4+1} \\ &= 1,1110 - 100000 \\ &= -0010 \end{aligned}$$

取反加1

$$[x]_{\text{补}} = 1,1110$$

取反：0001

$$\text{加1： } 0001 + 1 = 0010$$

# 练习 求下列真值的补码

真值	$[x]_{\text{补}}$	$[x]_{\text{原}}$
$x = +70 = 1000110$	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
$x = 0.0000$ $[+0]_{\text{补}} = [-0]_{\text{补}}$	0.0000	0.0000
$x = -0.0000$	0.0000	1.0000
$x = -1.0000$	1.0000	不能表示

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

$$[-1]_{\text{补}} = 2 + x = 10.0000 - 1.0000 = 1.0000$$

## ■ 8位二进制数整数

- 原码的表示范围是：-127 ~ +127
- 补码的表示范围是：-128 ~ +127

真值	原码
-127	1,1111111
-126	1,1111110
.....	
-1	1,0000001
-0	1,0000000
+0	0,0000000
+1	0,0000001
.....	
+126	0,1111110
+127	0,1111111

真值	补码
-128	1,0000000
-127	1,0000001
-126	1,0000010
.....	
-1	1,1111111
-0	0,0000000
+0	0,0000000
+1	0,0000001
.....	
+126	0,1111110
+127	0,1111111

## ■ 8位二进制数小数

- 原码的表示范围是：-127/128 ~ +127/128
- 补码的表示范围是：-1 ~ +127/128

真值	原码
-127/128	1.1111111
-126/128	1.1111110
.....	
-1/128	1.0000001
-0	1.0000000
+0	0.0000000
+1/128	0.0000001
.....	
+126/128	0.1111110
+127/128	0.1111111

真值	补码
-1	1.0000000
-127/128	1.0000001
-126/128	1.0000010
.....	
-1/128	1.1111111
-0	0.0000000
+0	0.0000000
+1/128	0.0000001
.....	
+126/128	0.1111110
+127/128	0.1111111

## 4. 反码表示法

### (1) 定义

整数

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

$x$  为真值

$n$  为整数的位数

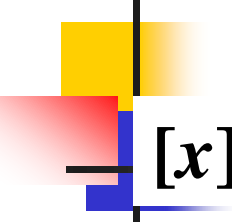
如  $x = +1101$

$[x]_{\text{反}} = 0, 1101$

$x = -1101$

$$\begin{aligned} [x]_{\text{反}} &= (2^{4+1} - 1) - 1101 \\ &= 11111 - 1101 \\ &= 1,0010 \end{aligned}$$

## 小数


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

如

$$x = +0.1101$$

$$[x]_{\text{反}} = \mathbf{0.1101}$$

$$x = -0.1010$$

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

## (2) 举例

正数的反码即为其本身

负数的反码为“按位取反”

例 已知  $[x]_{\text{反}} = 0,1110$  求  $x$

解: 由定义得  $x = +1110$

例 已知  $[x]_{\text{反}} = 1,1110$  求  $x$

解: 由定义得 
$$\begin{aligned} x &= [x]_{\text{反}} - (2^{4+1} - 1) \\ &= 1,1110 - 11111 \\ &= -0001 \end{aligned}$$

例 求 0 的反码

解: 设  $x = +0.0000$

$$[+0.0000]_{\text{反}} = 0.0000$$

$x = -0.0000$

$$[-0.0000]_{\text{反}} = 1.1111$$

同理, 对于整数

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

$$\therefore [+0]_{\text{反}} \neq [-0]_{\text{反}}$$

- **8位二进制数整数**
  - 原码的表示范围是: **-127 ~ +127**
  - 补码的表示范围是: **-128 ~ +127**
  - 反码的表示范围是: **-127 ~ +127**

真值	原码
-127	1,1111111
-126	1,1111110
.....	
-1	1,0000001
-0	1,0000000
+0	0,0000000
+1	0,0000001
.....	
+126	0,1111110
+127	0,1111111

真值	补码
-128	1,0000000
-127	1,0000001
-126	1,0000010
.....	
-1	1,1111111
-0	0,0000000
+0	0,0000000
+1	0,0000001
.....	
+126	0,1111110
+127	0,1111111

真值	反码
-127	1,0000000
-126	1,0000001
.....	
-1	1,1111110
-0	1,1111111
+0	0,0000000
+1	0,0000001
.....	
+126	0,1111110
+127	0,1111111


- **8位二进制数小数**
  - 原码的表示范围是: **-127/128 ~ +127/128**
  - 补码的表示范围是: **-1 ~ +127/128**
  - 反码的表示范围是: **-127/128 ~ +127/128**

真值	原码
-127/128	1.1111111
-126/128	1.1111110
.....	
-1/128	1.0000001
-0	1.0000000
+0	0.0000000
+1/128	0.0000001
.....	
+126/128	0.1111110
+127/128	0.1111111

真值	补码
-1	1.0000000
-127/128	1.0000001
-126/128	1.0000010
.....	
-1/128	1.1111111
-0	0.0000000
+0	0.0000000
+1/128	0.0000001
.....	
+126/128	0.1111110
+127/128	0.1111111

真值	反码
-127/128	1.0000000
-126/128	1.0000001
.....	
-1/128	1.1111110
-0	1.1111111
+0	0.0000000
+1/128	0.0000001
.....	
+126/128	0.1111110
+127/128	0.1111111

# 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
  - 对于负数，符号位为 1，其数值部分：

原码除符号位外每位取反末位加 1 → 补码

原码除符号位外每位取反 → 反码



**例6.1** 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	$\pm 0$	+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

机器数

真值（无符号数）

真值（原码）

真值（补码）

真值（反码）

**例6.1** 设机器数字长为 8 位（其中 1 位为符号位）  
 对于**小数**，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
0.0000000	0	+0	$\pm 0$	+0
0.0000001	1/128	+1/128	+1/128	+1/128
0.0000010	2/128	+2/128	+2/128	+2/128
⋮	⋮	⋮	⋮	⋮
0.1111111	127/128	+127/128	+127/128	+127/128
1.0000000	1	-0	-1	-127/128
1.0000001	129/128	-1/128	-127/128	-126/128
⋮	⋮	⋮	⋮	⋮
1.1111101	253/128	-125/128	-3/128	-2/128
1.1111110	254/128	-126/128	-2/128	-1/128
1.1111111	255/128	-127/128	-1/128	-0

机器数

真值（无符号数）

真值（原码）

真值（补码）

真值（反码）

## 例6.2 已知 $[y]_{\text{补}}$ 求 $[-y]_{\text{补}}$

解： 设  $[y]_{\text{补}} = y_0 \cdot y_1 y_2 \cdots y_n$

<I>

$$[y]_{\text{补}} = 0 \cdot y_1 y_2 \cdots y_n$$

正数

$$y = 0 \cdot y_1 y_2 \cdots y_n$$

$$-y = -0 \cdot y_1 y_2 \cdots y_n$$

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

<II>

$$[y]_{\text{补}} = 1 \cdot y_1 y_2 \cdots y_n$$

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

负数

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

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

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

结论：由  $[y]_{\text{补}}$  求  $[-y]_{\text{补}}$  为  
“连同符号位在内，每位求反，末位加1”



---

例如:  $[y]_{\text{补}} = 0,1111010$ , 求  $[-y]_{\text{补}}$

解:  $y = 1111010$ ,  $-y = -1111010$ ,  $[-y]_{\text{补}} = 1,0000110$

$[-y]_{\text{补}} = [y]_{\text{补}}$  的每位求反 + 1 =  $1,0000101 + 1 = 1,0000110$

例如:  $[y]_{\text{补}} = 1,0000110$ , 求  $[-y]_{\text{补}}$

解:  $y = -1111010$ ,  $-y = 1111010$ ,  $[-y]_{\text{补}} = 0,1111010$

$[-y]_{\text{补}} = [y]_{\text{补}}$  的每位求反 + 1 =  $0,1111001 + 1 = 0,1111010$

## 5. 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 大
$x = -21$	$-10101$	$1,01011$	

$x = +31$	$+11111$	$0,11111$	错 大
$x = -31$	$-11111$	$1,00001$	

如果：  
 $x + 2^5$

$+10101 + 100000 = 110101$	大 正确
$-10101 + 100000 = 001011$	

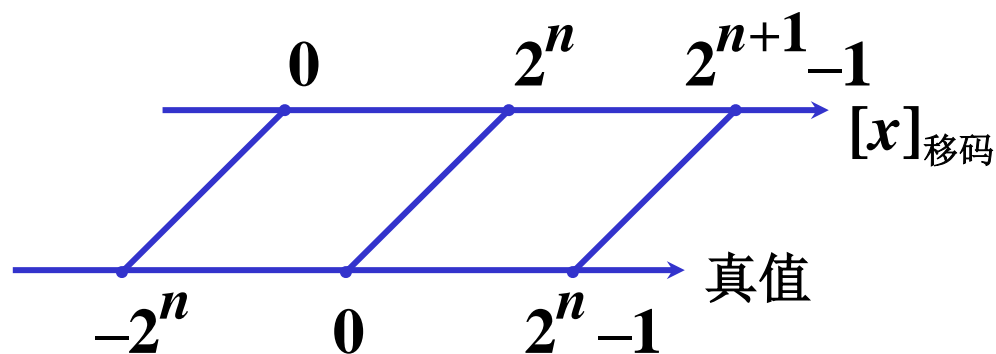
$+11111 + 100000 = 111111$	大 正确
$-11111 + 100000 = 000001$	

## (1) 移码定义

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

$x$  为真值,  $n$  为 整数的位数

移码在数轴上的表示



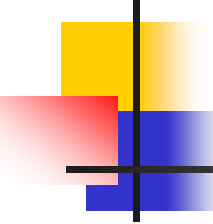
如  $x = 10100$

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

如  $x = -10100$

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

## (2) 移码和补码的比较



设  $x = +1100100$

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

$$[x]_{\text{补}} = \mathbf{0},1100100$$

设  $x = -1100100$

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

$$[x]_{\text{补}} = \mathbf{1},0011100$$

补码与移码只差一个符号位

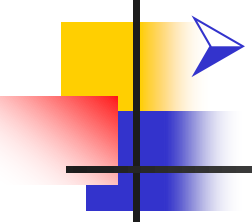
### (3) 真值、补码和移码的对照表

	真值 $x$ ( $n=5$ )	$[x]_{\text{补}}$	$[x]_{\text{移}}$	$[x]_{\text{移}}$ 对应的 十进制整数
-32	- 1 0 0 0 0 0	1 0 0 0 0 0	0 0 0 0 0 0	0
-31	- 1 1 1 1 1	1 0 0 0 0 1	0 0 0 0 0 1	1
-30	- 1 1 1 1 0	1 0 0 0 1 0	0 0 0 0 1 0	2
	⋮	⋮	⋮	⋮
-1	- 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 0	1 0 0 0 0 0	32
1	+ 0 0 0 0 1	0 0 0 0 0 1	1 0 0 0 0 1	33
2	+ 0 0 0 1 0	0 0 0 0 1 0	1 0 0 0 1 0	34
	⋮	⋮	⋮	⋮
+30	+ 1 1 1 1 0	0 1 1 1 1 0	1 1 1 1 1 0	62
+31	+ 1 1 1 1 1	0 1 1 1 1 1	1 1 1 1 1 1	63

真值+32



#### (4) 移码的特点



➤ 当  $x = 0$  时  $[+0]_{\text{移}} = 2^5 + 0 = 1,00000$

$[-0]_{\text{移}} = 2^5 - 0 = 1,00000$

$$\therefore [+0]_{\text{移}} = [-0]_{\text{移}}$$

➤ 当  $n = 5$  时 最小的真值为  $-2^5 = -100000$

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

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

- **8位二进制数整数**
  - 原码的表示范围是：-127 ~ +127
  - 补码的表示范围是：-128 ~ +127
  - 反码的表示范围是：-127 ~ +127
  - 移码的表示范围是：-128 ~ +127

真值	原码
-127	1,1111111
-126	1,1111110
.....	
-1	1,0000001
-0	1,0000000
+0	0,0000000
+1	0,0000001
.....	
+126	0,1111110
+127	0,1111111

真值	补码
-128	1,0000000
-127	1,0000001
-126	1,0000010
.....	
-1	1,1111111
-0	0,0000000
+0	0,0000000
+1	0,0000001
.....	
+126	0,1111110
+127	0,1111111

真值	反码
-127	1,0000000
-126	1,0000001
.....	
-1	1,1111110
-0	1,1111111
+0	0,0000000
+1	0,0000001
.....	
+126	0,1111110
+127	0,1111111

真值	移码
-128	0,0000000
-127	0,0000001
-126	0,0000010
.....	
-1	0,1111111
-0	0,0000000
+0	0,0000000
+1	1,0000001
.....	
+126	1,1111110
+127	1,1111111

- **8位二进制数小数**
  - 原码的表示范围是：-127/128 ~ +127/128
  - 补码的表示范围是：-1 ~ +127/128
  - 反码的表示范围是：-127/128 ~ +127/128

小数没有移码

真值	原码
-127/128	1.1111111
-126/128	1.1111110
.....	
-1/128	1.0000001
-0	1.0000000
+0	0.0000000
+1/128	0.0000001
.....	
+126/128	0.1111110
+127/128	0.1111111

真值	补码
-1	1.0000000
-127/128	1.0000001
-126/128	1.0000010
.....	
-1/128	1.1111111
-0	0.0000000
+0	0.0000000
+1/128	0.0000001
.....	
+126/128	0.1111110
+127/128	0.1111111

真值	反码
-127/128	1.0000000
-126/128	1.0000001
.....	
-1/128	1.1111110
-0	1.1111111
+0	0.0000000
+1/128	0.0000001
.....	
+126/128	0.1111110
+127/128	0.1111111

# 有符号数

- 真值（十进制、二进制、十六进制）

- $+10 = +1010$

$$-10 = -1010$$

- $+0.625 = 0.1010$

$$-0.625 = -0.1010$$

- 机器数（二进制、十六进制）

- 原码  $+1010 = 0,1010$

$$-1010 = 1,1010$$

$$+0.1010 = 0.1010$$

$$-0.1010 = 1.1010$$

- 补码  $+1010 = 0,1010$

$$-1010 = 1,0110$$

$$+0.1010 = 0.1010$$

$$-0.1010 = 1.0110$$

- 反码  $+1010 = 0,1010$

$$-1010 = 1,0101$$

$$+0.1010 = 0.1010$$

$$-0.1010 = 1.0101$$

- 移码  $+1010 = 1,1010$

$$-1010 = 0,0110$$

小数没有移码



## 6.2 数的定点表示和浮点表示

一、定点表示（定点数）

二、浮点表示（浮点数）

三、定点数和浮点数的比较

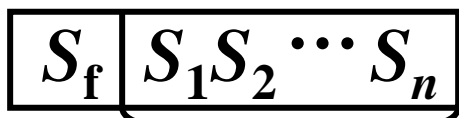
四、举例

五、**IEEE 754** 标准

# 一、定点表示

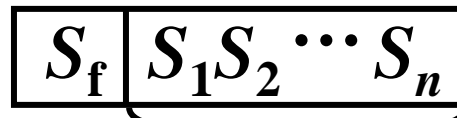
## 小数点按约定方式标出

纯小数



数符  
数值部分  
小数点位置

或



数符  
数值部分  
小数点位置

纯整数

定点机

小数定点机

整数定点机

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

$$-(1 - 2^{-n}) \sim +(1 - 2^{-n})$$

$$-(2^n - 1) \sim +(2^n - 1)$$

## 二、浮点表示

电子的质量= $9.1 \times 10^{-28} \text{g} = 0.91 \times 10^{-27} \text{g}$

太阳的质量= $1.989 \times 10^{33} \text{g} = 0.1989 \times 10^{34} \text{g}$


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

浮点数的一般形式

$S$  尾数     $j$  阶码     $r$  基数（基值）

计算机中  $r$  取 2、4、8、16 等

当  $r = 2$

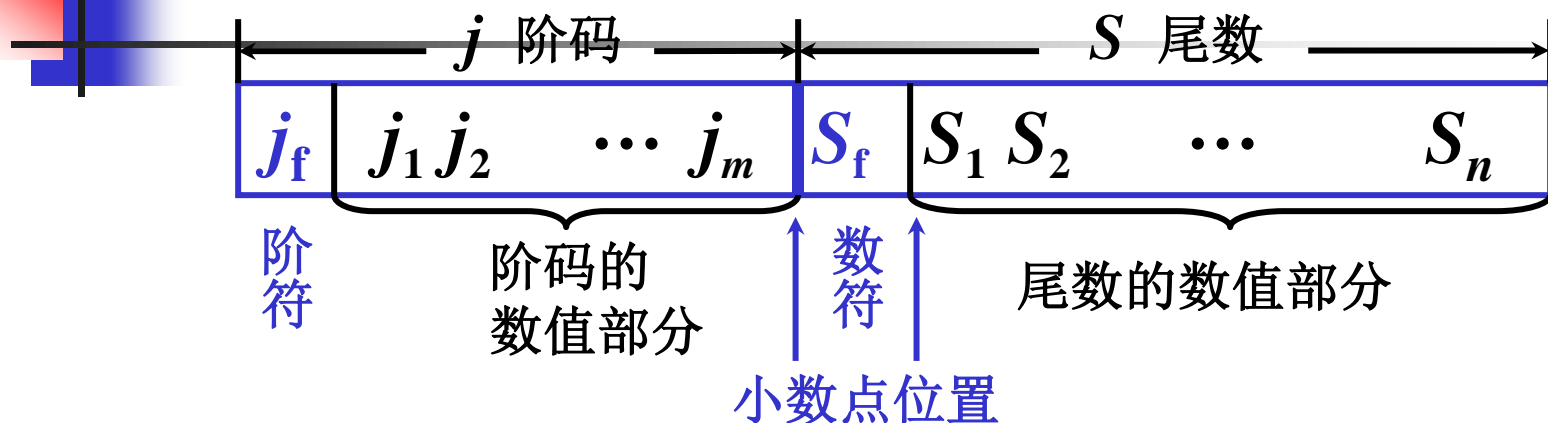
$$\begin{aligned} N &= 11.0101 \\ &= 0.110101 \times 2^{10} \quad \text{规格化数} \\ &= 1.10101 \times 2^1 \\ &= 1101.01 \times 2^{-10} \\ &= 0.00110101 \times 2^{100} \end{aligned}$$

计算机中     $S$  小数、可正可负  
                   $j$  整数、可正可负

$$0.110101 \times 2^{10}$$

0 10 0 110101

# 1. 浮点数的表示形式



$S_f$  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

$j_f$  和  $m$  共同表示小数点的实际位置

## 2. 浮点数的表示范围

上溢 阶码 > 最大阶码

下溢 阶码 < 最小阶码 按 **机器零** 处理

上溢

上溢

负数区

下溢

正数区

0

最小负数(绝对值最大负数)

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

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

最大正数

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

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

最小正数

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

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

最大负数(绝对值最小负数)

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

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

设  $m = 4$

$n = 10$



## 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：  $2^{14} = 16384$        $2^{15} = 32768$

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

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

$$2^m - 1 = 15 \quad m=4$$

$$n = 24 - 4 - 1 - 1 = 18$$

阶码、尾数分别取4位和18位

-15 ~ +15

$-(1-2^{-18}) \sim -(2^{-18})$     0     $+(2^{-18}) \sim +(1-2^{-18})$

X , XXXX

X . XXXXXXXXXXXXXXXXXXXXXXXX

1位    4位

1位    18位

1,1111

1.00000000000000000001

1,1111

0.00000000000000000001

0,1111    1.11111111111111111111

0,1111    0.11111111111111111111

0

$$2^{-15} \times (-2^{-18}) = -2^{-33}$$

最大负数

$$2^{-15} \times 2^{-18} = 2^{-33}$$

最小正数

最小负数

$$2^{15} \times (-(1-2^{-18})) = -32768 + 2^{-3}$$

最大正数

$$2^{15} \times (1-2^{-18}) = 32768 - 2^{-3}$$

X , XXXX

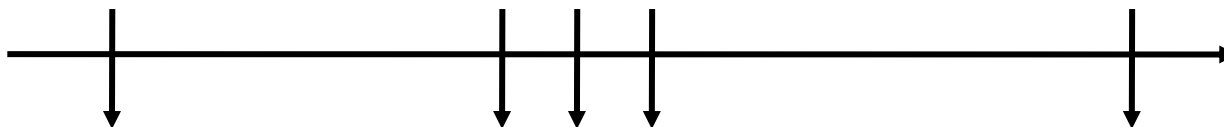
1位 3位

X . XXXXXXXXXXXXXXXXXXXXXXXX

1位 19位

表示范围不够

0



最大负数

$$2^{-7} \times (-2^{-19}) = -2^{-26}$$

$$2^{-7} \times 2^{-19} = 2^{-26}$$

最小正数

最小负数

$$2^7 \times (-(1 - 2^{-19})) = -128 + 2^{-12}$$

最大正数

$$2^7 \times (1 - 2^{-19}) = 128 - 2^{-12}$$

X, XXXX

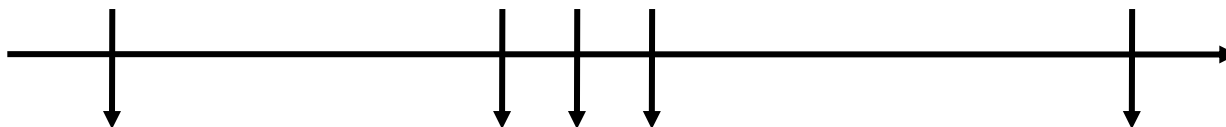
1位 5位

X. XXXXXXXXXXXXXXXXXXXXX

1位 17位

精度减少

0



最大负数

$$2^{-31} \times (-2^{-17}) = -2^{-48}$$

$$2^{-31} \times 2^{-17} = 2^{-48}$$

最小正数

最小负数

$$2^{31} \times (-(1 - 2^{-17}))$$

最大正数

$$2^{31} \times (1 - 2^{-17})$$

### 3. 浮点数的规格化形式

本章的重点 (2)

电子的质量= $9.1 \times 10^{-28}g$

非规格化表示

电子的质量= $0.91 \times 10^{-27}g$

规格化表示

太阳的质量= $0.01989 \times 10^{35}g$

非规格化表示

太阳的质量= $0.1989 \times 10^{34}g$

规格化表示

十进制数

尾数为[0.1 1)

$N = 11.0101$

非规格化表示

$N = 0.110101 \times 2^{10}$

规格化表示

$M = 0.0110101$

非规格化表示

$M = 0.110101 \times 2^{-1}$

规格化表示

二进制数

尾数为[0.5 1)

## 4. 浮点数的规格化

右规 尾数右移 1 位，阶码加 1

左规 尾数左移 1 位，阶码减 1

例：

右规 尾数右移 2 位，阶码加 2

$N = 11.0101$

非规格化表示

$N = 0.110101 \times 2^{10}$

规格化表示

例：

左规 尾数左移 1 位，阶码减 1

$M = 0.0110101$

非规格化表示

$M = 0.110101 \times 2^{-1}$

规格化表示

例如： 设  $m = 4$ ,  $n = 10$ ,  $r = 2$

尾数规格化后的浮点数表示范围


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

$$\text{最小正数} \quad 2^{-1111} \times 0.\overset{0.5}{1000000000} = 2^{-15} \times 2^{-1} = 2^{-16}$$

$$\text{最大负数} \quad 2^{-1111} \times (-0.\overset{-0.5}{1000000000}) = -2^{-15} \times 2^{-1} = -2^{-16}$$

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

X, XXXX

1位 4位

X. XXXXXXXXXXXX

1位 10位

尾数规格化后的浮点数表示范围

0

最大负数

$$2^{-15} \times (-2^{-1}) = -2^{-16}$$

$$2^{-15} \times 2^{-1} = 2^{-16}$$

最小正数

最小负数

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

最大正数

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



X, XXXX

1位 4位

X. XXXXXXXXXXXX

1位 10位

非规格化的浮点数表示范围

0

最大负数

$$2^{-15} \times (-2^{-10}) = -2^{-25}$$

$$2^{-15} \times 2^{-10} = 2^{-25}$$

最小正数

最小负数

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

最大正数

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

# 浮点数规格化

## ■尾数:

■十进制数: 绝对值  $\geq 0.1$

[0.1 1)

■二进制数: 绝对值  $\geq 0.5$

[0.5 1)

## ■尾数:

### ■正数:

■原码、反码、补码: 0 (符号位) 1 (最高有效位)

相反

$X=0.1100$

$X_{\text{原}}=0.1100$

$X_{\text{补}}=0.1100$

$X_{\text{反}}=0.1100$

### ■负数:

■原码: 1 (符号位) 1 (最高有效位)

相同

■反码、补码: 1 (符号位) 0 (最高有效位)

相反

$X=-0.1100$

$X_{\text{原}}=1.1100$

$X_{\text{补}}=1.0100$

$X_{\text{反}}=1.0011$

### 三、定点数和浮点数的比较

- 浮点数的表示范围比定点数大
- 浮点数的精度比定点数高
- 浮点运算步骤比定点运算步骤多、运算速度比定点运算低、运算线路比定点运算的复杂
- 溢出判断：浮点数是对规格化数的阶码进行判断，定点数是对数值本身进行判断

## 四、举例

例 6.3 将  $+\frac{19}{128}$  写成二进制定点数、浮点数及在定点机和浮点机中的机器数形式。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）。

解： 设  $x = +\frac{19}{128}$

x . xxxx

x . xxxxxxxxxxxx

二进制形式  $x = 0.0010011$

定点表示  $x = 0.0010011\ 000$

浮点规格化形式  $x = 0.1001100000 \times 2^{-10}$

定点机中 **11位**  $[x]_{\text{原}} = [x]_{\text{补}} = [x]_{\text{反}} = 0.0010011000$

浮点机中 **共16位**  $[x]_{\text{原}} = 1, 0010; 0.1001100000$

$[x]_{\text{补}} = 1, 1110; 0.1001100000$

$[x]_{\text{反}} = 1, 1101; 0.1001100000$

**例6.4** 将  $-58$  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三种机器数及阶码为移码、尾数为补码的形式（其他要求同上例）。

**解：** 设  $x = -58$

二进制形式  $x = -111010$

定点表示  $x = -0000111010$

浮点规格化形式  $x = -(0.1110100000) \times 2^{110}$

**定点机中** 11位

$[x]_{\text{原}} = 1, 0000111010$

$[x]_{\text{补}} = 1, 1111000110$

$[x]_{\text{反}} = 1, 1111000101$

移码为补码的符号位取反

**浮点机中** 共16位

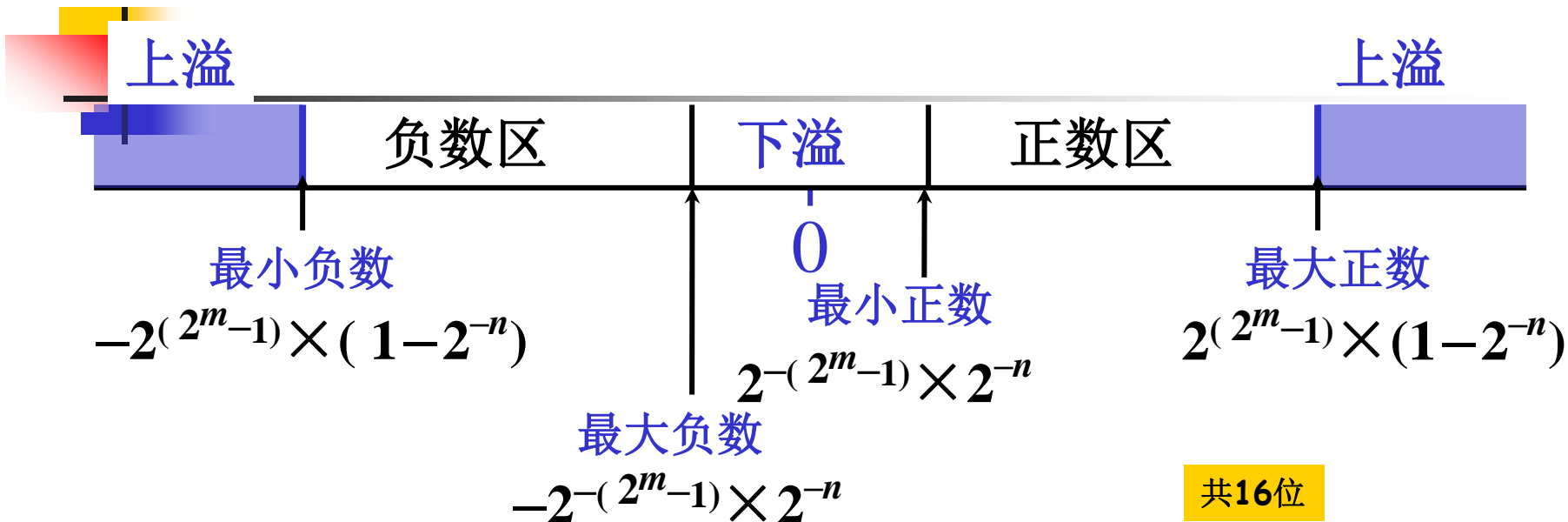
$[x]_{\text{原}} = 0, 0110; 1. 1110100000$

$[x]_{\text{补}} = 0, 0110; 1. 0001100000$

$[x]_{\text{反}} = 0, 0110; 1. 0001011111$

$[x]_{\text{阶移、尾补}} = 1, 0110; 1. 0001100000$

**例6.5** 写出对应下图所示的浮点数的补码形式。 设  $n = 10$ ,  $m = 4$ , 阶符、数符各取 1 位。



解:

真值

阶码和尾数均为补码

最大正数  $2^{15} \times (1-2^{-10})$

0,1111; 0.1111111111

最小正数  $2^{-15} \times 2^{-10}$

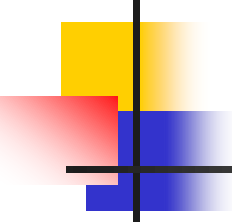
1,0001; 0.0000000001

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

- 
- **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**-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：**  $x = -53/512 = -0.000110101$

$$= 2^{-11} \times (-0.1101010000)$$

左移 3 位，阶码减 3

$$[x]_{\text{原}} = 1,0011; 1.1101010000$$

$$[x]_{\text{补}} = 1,1101; 1.0010110000$$

$$[x]_{\text{反}} = 1,1100; 1.0010101111$$

共16位

$$[x]_{\text{移码,尾补}} = 0,1101; 1.0010110000$$

# 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 **阶码等于或小于它所表示的最小 数** 时，不论尾数为何值，按机器零处理

如  $m = 4$        $n = 10$

当阶码和尾数都用补码表示时，机器零为

$\times, \times \times \times \times; \quad 0.00 \dots 0$   
(阶码 = -16)  $1, 0000; \quad \times.\times\times \dots \times$

当阶码用**移码**，尾数用补码表示时，机器零为

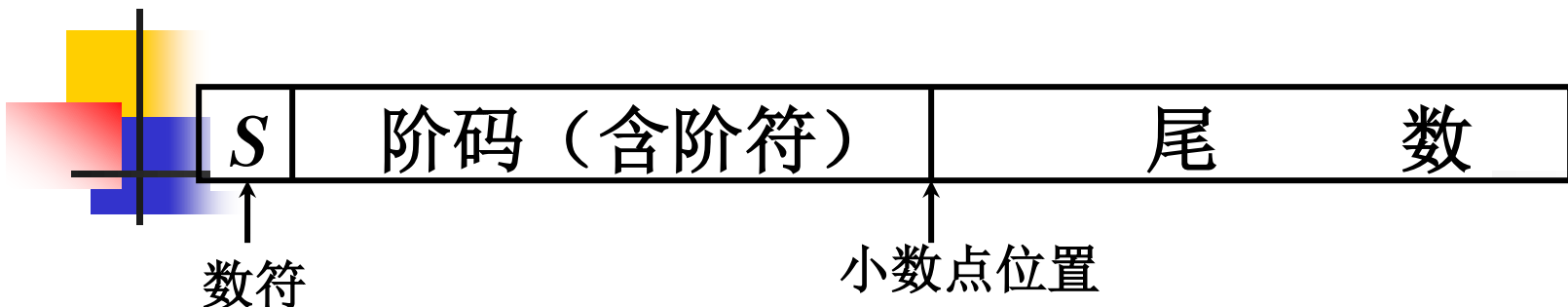
$0, 0000; \text{ 或 } 0.00 \dots 0$

有利于机器中 “判 0” 电路的实现

这就是为什么阶码要用移码表示的原因



## 五、IEEE 754 标准



尾数为规格化表示

非“0”的有效位最高位为“1”（隐含）

		符号位 $S$	阶码	尾数	总位数
单精度	短实数	1	8	23	32
双精度	长实数	1	11	52	64
扩展精度	临时实数	1	15	64	80

双精度（长实数）：偏移的阶码是“+1023”（011 1111 1111）

扩展精度（临时实数）：偏移的阶码是“+16383”（011 1111 1111 1111）

## ■ 实数178.125的表示：

单精度（短实数） 32位

■ 二进制数：10110010.001

■ 二进制浮点表示：1.0110010001X2<sup>111</sup>

右移7位，阶码加7

■ 短实数表示：

■ 符号：0

■ 偏移的阶码：10000110（=00000111+01111111）

7FH=127

■ 尾数的有效位：011001000100000000000000

实际上是 1 011001000100000000000000

隐含

178.125 = 0 10000110 011001000100000000000000

1位 8位

23位

**IEEE 754 标准与前面的浮点数表示方式不一样！**

- 例子：-9.625用单精度（短实数）IEEE 754怎么表示（十六进制）？

- 解：-9.625表示为二进制数：

$$-1001.101 = -1.001101 * 2^{11}$$

短实数表示：

符号位：1

偏移的阶码：11+0111 1111=1000 0010

尾数：1 001101000000000000000000

隐含

7FH=127



1      1000 0010      001101000000000000000000

十六进制：1100 0001 0001 1010 0000 0000 0000 0000

C      1      1      A      0      0      0      0

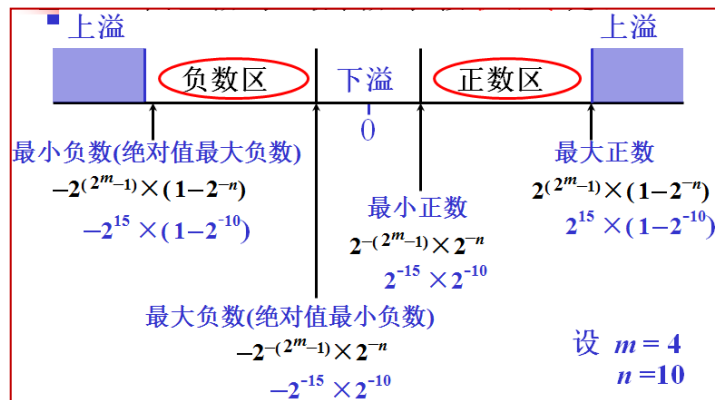
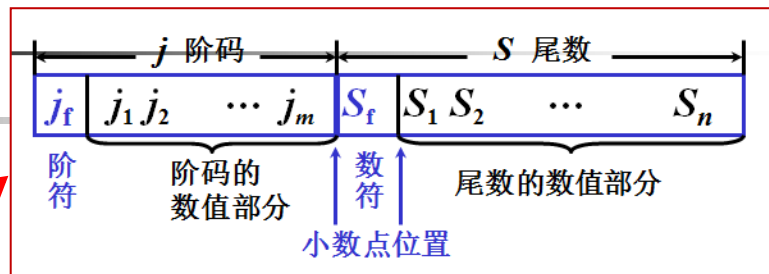
-9.625 = 0xC11A0000H

## ■ 定点数（定点机）

- 定点整数（纯整数）
- 定点小数（纯小数）

## ■ 浮点数（浮点机）

- 浮点数的表示形式



本章的重点

- 浮点数的表示范围

本章的重点

- 浮点数的规格化

### ■ 尾数:

- 十进制数: 绝对值  $\geq 0.1$  [0.1 1)
- 二进制数: 绝对值  $\geq 0.5$  [0.5 1)

### ■ 尾数:

- 正数:
  - 原码、反码、补码: 0 (符号位) 1 (最高有效位)

$X=0.1100$   $X_{原}=0.1100$   $X_{补}=0.1100$   $X_{反}=0.1100$

### ■ 负数:

- 原码: 1 (符号位) 1 (最高有效位)
- 反码、补码: 1 (符号位) 0 (最高有效位)

$X=-0.1100$   $X_{原}=1.1100$   $X_{补}=1.0100$   $X_{反}=1.0011$



# 第10次作业——习题(P289-293)

---

■ 6.4

■ 6.5

■ 6.9

■ 6.11

■ 6.12

■ 6.14

■ 6.15

■ 6.16



# 关于作业的提交

- **1周内**必须提交（上传到学院的**FTP**服务器上），否则认为是迟交作业；如果期末仍然没有提交，则认为是未提交作业
  - 作业完成情况成绩=第**1**次作业提交情况\*第**1**次作业评分+第**2**次作业提交情况\*第**2**次作业评分+.....+第**N**次作业提交情况\*第**N**次作业评分
  - 作业评分：**A**（好）、**B**（中）、**C**（差）三挡
  - 作业提交情况：按时提交（**1.0**）、迟交（**0.5**）、未提交（**0.0**）
- 请采用电子版的格式（**Word**文档）上传到**FTP**服务器上，文件名取“学号+姓名+第**X**次作业.doc”
  - 例如：**11920192203642+袁佳哲+第8次作业.doc**
- 第**10**次作业提交的截止日期为：**2021年5月7日晚上24点**



**The End**

---

**Thanks**