计算机组成原理

群名称:2022春-计组4班
群　号:139369553

文杰

计算机科学与技术学院

wenjie@hit.edu.cn

个人主页：http://faculty.hitsz.edu.cn/wenjie

第二章 RISC-V汇编及其指令系统

第二章 RISC-V汇编及其指令系统

- **计算机中数的表示**
- RISC-V概述
- RISC-V汇编语言
- RISC-V指令表示
- 案例分析

计算机中数的表示

- 计算机中数的表示
  - **无符号数和有符号数**

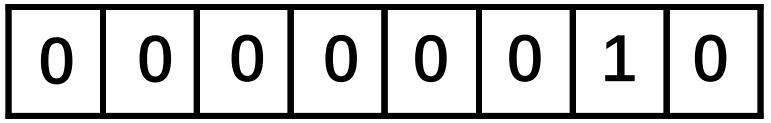
    机器数与真值；原码/补码/反码/移码表示法
  - 定点表示和浮点表示
  - IEEE754标准
  - 算数移位与逻辑移位

无符号数

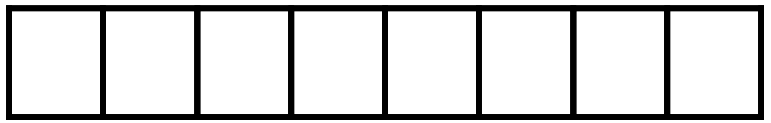
- 以8位寄存器为例

十进制　　二进制

无符号数2

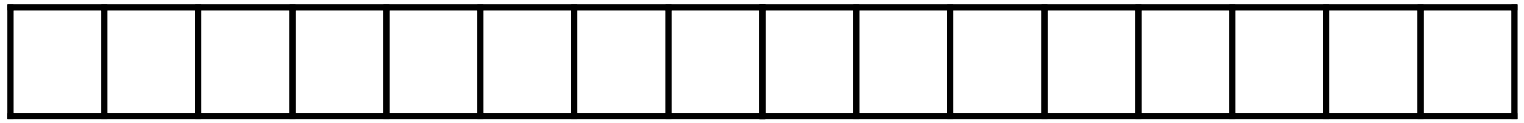


8 位



0 ~ 255

16 位



0 ~ 65535

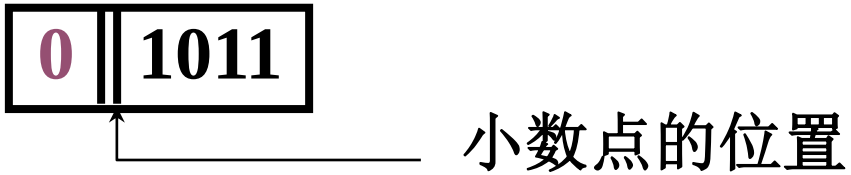
寄存器的位数反映无符号数的表示范围

有符号数：真值与机器数

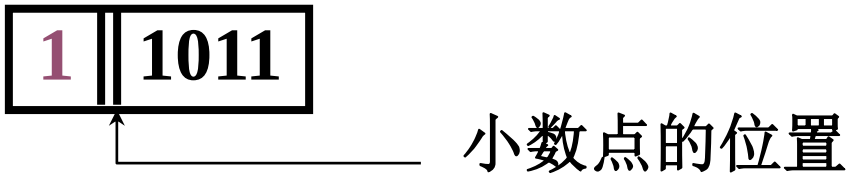
真值：带符号的数

机器数：符号数字化的数

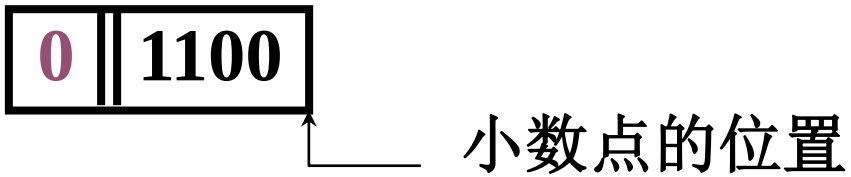
+ **0.1011或0.1011**



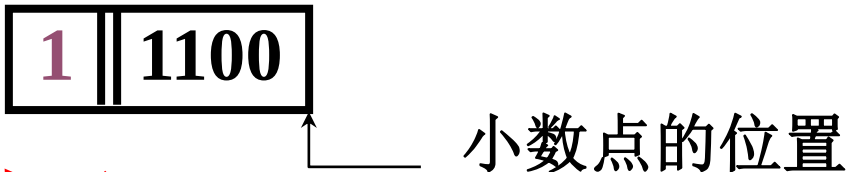
– **0.1011**



+ **1100或1100**



– **1100**



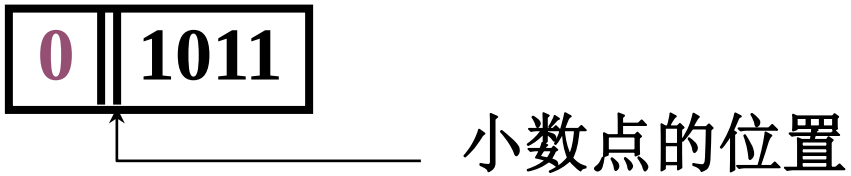
注：以后非特殊说明，默认二进制数表示；

二进制数位数不是8的倍数，只是为了讲解方便。

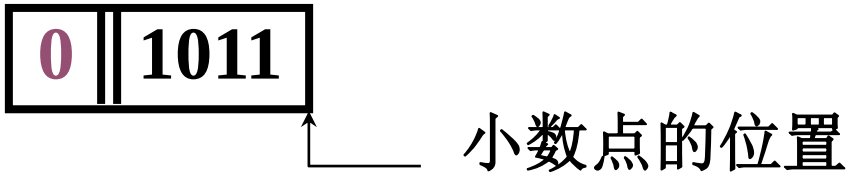
有符号数：真值与机器数

真值：带符号的数　　**机器数：符号数字化的数**

+ 0.1011



+ 1011



- 小数和整数在计算机中的表示一模一样?
- 如何表示既有整数又有小数的数据，如+11.01?

原码表示法：整数

带符号的绝对值表示

$x = +1110 \qquad [x]_{原} = 0\,,\,1110$

$x = -1110 \qquad [x]_{原} = 1\,,\,1110$

用 逗号 将符号位和数值部分隔开

$$[x]_{原} = 2^4 + 1110 = 1\,,\,1110$$

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

x是真值，n 是数值位数

原码表示法：小数

$x = +0.1101$ $[x]_{原} = 0.1101$

用 小数点 将符号位和数值部分隔开

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

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

用 小数点 将符号位和数值部分隔开

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

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

x是真值

举例：根据原码求真值

- 例1. 已知 $[x]_{原} = 1.0011$，求 x

  解：由小数原码定义得

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

- 例2. 已知 $[x]_{原} = 1,1100$，求 x

  解：由整数原码定义得

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

举例

- 例3. 已知 $[x]_{原} = 0.1101$, 求 x

  解：根据定义 $\because [x]_{原} = 0.1101$

  $\therefore \quad x = +0.1101$

- 例4. 求 $x = 0$ 的原码

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

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

  同理，对于整数 $[+0]_{原} = 0,0000$ 假设整数的数值位是4位

  $[-0]_{原} = 1,0000$

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

  注意：$x = 0$的原码要分成小数和整数分别讨论

原码的优缺点

- 优点：简单、直观
- 缺点：1）+0和-0原码不一样

2）作加法运算时，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	加法	正
加法	正	负	减法	可正可负
加法	负	正	减法	可正可负
加法	负	负	加法	负

实际运算时能否只作加法运算?

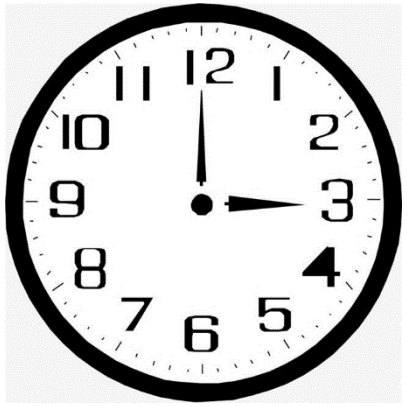
原码的缺点

十进制		原码	
	3		0 0 0 0 0 0 1 1
+	-3	+	1 0 0 0 0 0 1 1
	0		
	-6	转十进制	1 0 0 0 0 1 1 0

原码运算 不等于 十进制运算?

补数表示法

- 例：将时钟从5点调到3点
- 补的概念：时钟以12为模
  - 逆时针：5 - 2 = 3
  - 顺时针：5+10 = 3 + 12



- 可见 -2 可用 +10 代替
  - 称 +10 是 -2 （以 12 为模）的**补数**
  - 记作 $-2 \equiv +10 \pmod{12}$

    同理 $-4 \equiv +8 \pmod{12}$

    $-5 \equiv +7 \pmod{12}$

减法 ⟶ 加法

补数——续

计数器（模 16）1011 ⟶ 0000 ？

$$\begin{array}{r} 1011 \\ -\ 1011 \\ \hline 0000 \end{array} \qquad \begin{array}{r} 1011 \\ +\ 0101 \\ \hline 10000 \end{array}$$

自然去掉

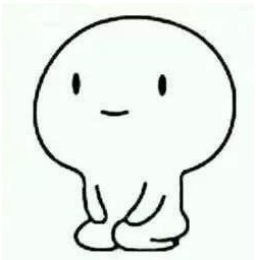
可见 −1011 与 +0101 作用等价

记作 $-1011 \equiv +0101 \pmod{2^4}$

同理 $-011 \equiv +101 \pmod{2^3}$

$-0.1001 \equiv +1.0111 \pmod{2^1}$

正数的补数？



- 结论(真值的绝对值小于模)
  - 一个负数加上 “模” 即得该负数的补数
  - 一个负数和一个正数互为补数时，绝对值之和即为模数

补数——续

正数的补数即为其本身

对于时钟：3点、15点、27点都是3点 ⟹ $3 \equiv 15 \equiv 27 \pmod{12}$

$3 \equiv 3+12 \equiv 3+24 \equiv 3 \pmod{12}$

同理：$+0101 \equiv +0101 + 2^4 \equiv +0101 \pmod{2^4}$

前面已证明：$-1011 \equiv -1011 + 2^4 \equiv +0101 \pmod{2^4}$

$+\ 0101 \longrightarrow 0,\ 0101$

$-\ 1011 \longrightarrow 1,\ 0101$

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

-1011

$1,0101$

$\pmod{2^{4+1}}$

问题：+ 0101 究竟是-1011的补数还是+0101的补数呢?

补码表示法：二进制整数

整数补码定义：

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

其中：x 为真值，n 为二进制整数的位数

$x = +0101000$

$[x]_{补} = 0,0101000$

$x = -1011000$

$[x]_{补} = 2^{7+1} + (-1011000)$

$= 100000000$

$- \quad 1011000$

$1,0101000$

用 逗号 将符号位和数值部分隔开

补码表示法：二进制（纯）小数

小数补码定义：

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

其中：x 为真值

$x = +0.1110$

$[x]_{补} = 0.1110$

用 小数点 将符号位和数值部分隔开

$x = -0.1100000$

$[x]_{补} = -0.1100000 + 2$

$= 10.0000000$

$- 0.1100000$

$= 1.0100000$

求补码的快捷方式

设 $x = -1010$ 时

则 $[x]_{补} = 2^{4+1} - 1010 = 100000 - 1010 = 1,0110$

$= 11111 + 1 - 1010 = 11111 + 1 - 1010 = 10101 + 1 = 1,0110$

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

举例：已知小数补码求真值

已知 $[x]_{补} = 1.0001$，求 x和$[x]_{原}$ 。

解：由定义得 $x = [x]_{补} - 2$

$= 1.0001 - 10.0000$

$= -0.1111$

根据真值x

$[x]_{补}$除符号位外，按位取反

$\Rightarrow [x]_{原} = 1.1111 = 1.1110 + 0.0001$

结论：当真值为负时，已知补码求原码的快捷方法：

- 补码除符号位外，每位取反，末位加 1
- 补码除符号位外，末位减 1，再每位取反

练习：已知负数补码求真值

已知 $[x]_{补} = 1,1110$，求 x

解：由定义得 $x = [x]_{补} - 2^{4+1}$

$= 1,1110 - 100000$

$= -0010$

或：$[x]_{补}$ ➡ $[x]_{原}$ ➡ x

根据快捷求法：$[x]_{原} = 1,0001+1 = 1,0010$

$\therefore x = -0010$

练习：求下列真值的补码

真值	$[x]_{补}$	$[x]_{原}$
$x = -70 = -1000110$	1, 0111010	1,1000110
$x = -0.1110$	1.0010	1.1110
$x = +0.0000$ $[+0]_{补} = [-0]_{补}$	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 $[x]_{补} = \begin{cases} x & 1 > x \geq 0 \\ 2 + x & 0 > x \geq -1 \ (\mathrm{mod}\ 2) \end{cases}$

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

反码表示法：二进制整数

整数反码定义：

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

其中：x 为真值，n为数值位数

$x = +1101$

$[x]_{反} = 0,1101$

用 逗号 将符号位和数值部分隔开

$x = -1101$

$[x]_{反} = (2^{4+1}-1)-1101$

$= 11111-1101$

$= 1,0010$

$[x]_{原} = 1,1101$

数值位按位取反

反码表示法：二进制小数

小数反码定义：

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

其中：x 为真值，n为数值位数

$$x = -0.1010$$

$$[x]_{反} = (2-2^{-4}) - 0.1010$$

$$= 1.1111 - 0.1010$$

$$= 1.0101$$

用 小数点 将符号位和数值部分隔开

$$[x]_{原} = 1.1010$$

数值位按位取反

例子：已知反码求真值，0的反码

- 已知 $[x]_{反} = 1,1110$，求 x

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

  $= 1,1110 - 11111$

  $= -0001$

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

  $[+0]_{反} \neq [-0]_{反}$

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，符号位为0，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（**反码**） 末位加 1 -> **补码**
- 当真值为 负 时，**已知补码求原码**的方法：
  - 补码除符号位外，每位取反，末位加 1
  - 补码除符号位外，末位减 1，再每位取反

例子：机器数的真值

- 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

例：已知$[y]_{补}$，求$[-y]_{补}$

例：设 $[y]_{补} = y_0.y_1y_2\cdots y_n$，求$[-y]_{补}$。

解：<I> $[y]_{补} = 0.y_1y_2\cdots y_n$

$y = 0.y_1y_2\cdots y_n$

$-y = -0.y_1y_2\cdots y_n$

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

<II> $[y]_{补} = 1.y_1y_2\cdots y_n$

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

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

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

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

$[y]_{补}$连同符号位在内，每位取反，末位加 1，即得$[-y]_{补}$

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
以上 $x + 2^5$		+10101 + 100000 = 110101		大 正确
		−10101 + 100000 = 001011		
		+11111 + 100000 = 111111		大 正确
		−11111 + 100000 = 000001		

移码表示法：二进制整数

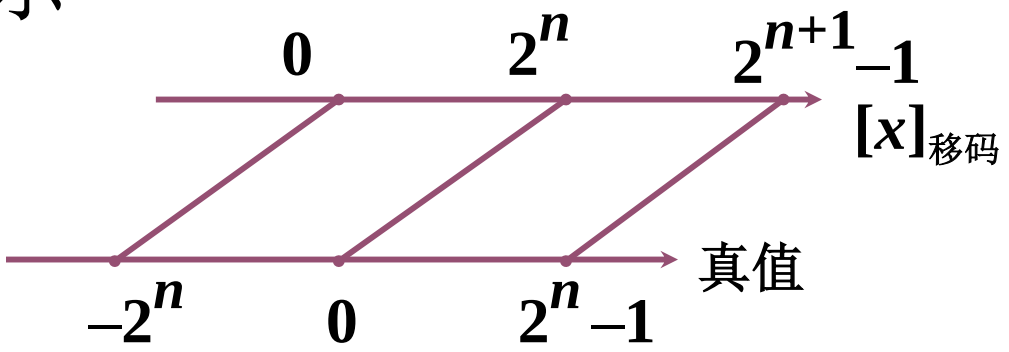
- 移码定义

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

其中：x 为真值，n 为 整数的位数

小数的移码定义呢？

- 移码在数轴上的表示



- 例：

$x = 10100$

$[x]_{移} = 2^5 + 10100 = 1,10100$

用 逗号 将符号位和数值位隔开

$x = -10100$

$[x]_{移} = 2^5 - 10100 = 0,01100$

移码和补码的比较

设 $x = +1100100$

$[x]_{移} = 2^7 + 1100100 = 1,1100100$

$[x]_{补} = 0,1100100$

设 $x = -1100100$

$[x]_{移} = 2^7 + (-1100100) = 0,0011100$

$[x]_{补} = 2^{7+1} - 1100100 = 1,0011100$

补码与移码只差一个符号位

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

续前表，$n=5$

$$[+0]_{移} = 2^5 + 0 = 1,00000$$

$$[-0]_{移} = 2^5 - 0 = 1,00000$$

$$[+0]_{移} = [-0]_{移}$$

最小真值 $-2^5 = -100000$对应的移码为 $2^5 - 100000 = 000000$

最小真值的移码为全 0

可用移码思想表示浮点数的阶码，便于判断浮点数阶码大小

第二章 计算机中数的表示

- **计算机中数的表示**
  - 无符号数和有符号数
  - 定点表示和浮点表示
  - IEEE754标准
  - 算数移位与逻辑移位

IEEE 754浮点数：单精度为例

指数	尾数	表示对象	换算方法
0	0	0	**规定** （符号位不同，存在+0.0和-0.0）
0	非0	正负非规格化数	正负非规格化数 = $(-1)^S * (尾数_2) * 2^{(0-126)}$ （S代表符号位，1为负数，0为正数）
[1: 254]	任意	正负浮点数	正负浮点数 = $(-1)^S * (1 + 尾数_2) * 2^{(指数-127)}$
255	0	正负无穷 (inf)	**规定**
255	非零	NaN	**规定**

IEEE 754浮点数：真值转二进制

- 例题
  - 将十进制 -0.75 转为单精度 IEEE 754格式二进制
- **解：** 根据十进制小数转二进制小数算法：$-0.75_{10} = -0.11_2$

  规格化：$-0.11 = -1.1 * 2^{-1}$，能够规格化，说明是正负浮点数表示

  $-1.1 * 2^{-1} = (-1)^S \times (1 + 尾数_2) \times 2^{(指数 - 127)}$

  $= (-1)^1 \times (1 + 0.1_2) \times 2^{(126 - 127)}$

  符号位：1；指数部分：126；尾数部分：0.1_2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	0	1	1	1	1	1	1	0	1	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IEEE754相关网址：https://www.h-schmidt.net/FloatConverter/IEEE754.html

IEEE 754浮点数：真值转二进制

IEEE754相关网址：https://www.h-schmidt.net/FloatConverter/IEEE754.html

Tools & Thoughts

IEEE-754 Floating Point Converter

Translations: de

This page allows you to convert between the decimal representation of numbers (like "1.02") and the binary format used by all modern CPUs (IEEE 754 floating point).

IEEE 754 Converter (JavaScript), V0.22

	Sign	Exponent	Mantissa
Value:	-1	2^{-1}	1.5
Encoded as:	1	126	4194304
Binary:	☑	☐ ☑ ☑ ☑ ☑ ☑ ☑ ☐	☑ ☐

You entered	-0.75
Value actually stored in float:	-0.75
Error due to conversion:	0.00
Binary Representation	10111111010000000000000000000000
Hexadecimal Representation	0xbf400000

+1 -1

Update

There has been an update in the way the number is displayed. Previous version would give you the represented value as a possibly rounded decimal number and the same number with the increased pre float. Now the original number is shown (either as the number that was entered, or as a possibly rounded decimal string) as well as the actual full precision decimal number that the float value is represe a nice example to see this behaviour. The difference between both values is shown as well, so you can easier tell the difference between what you entered and what you get in IEEE-754

IEEE 754浮点数：二进制转真值

- 例题
  - 将二进制IEEE754浮点数表示转换为十进制浮点数（空白为0）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- 解

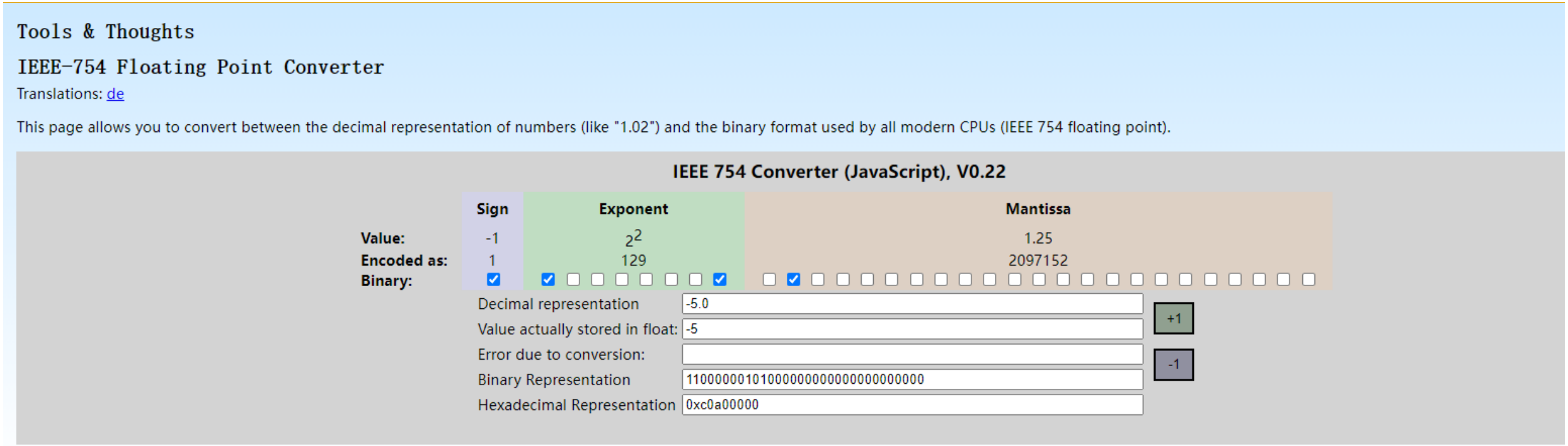
符号位为1，指数字段为129，尾数字段为$2^{-2} = 0.25$。是浮点数

$$(-1)^S \times (1 + \text{尾数}_2) \times 2^{(\text{指数} - 127)} = (-1)^1 \times (1 + 0.25) \times 2^{(129 - 127)}$$

$$= -1 \times 1.25 * 2^2$$

$$= -5.0$$

IEEE 754浮点数：二进制转真值



补码加减


计算器
程序员
记忆
127 + 1 =
-128
存储器中未保存数据
HEX 80
DEC -128
OCT 200
BIN 1000 0000
BYTE
MS
按位
位移位
A
<<
>>
CE
B
(
)
%
÷
C
7
8
9
×
D
4
5
6
−
E
1
2
3
+
F
+/-
0
.
=

第二章 计算机中数的表示

- **计算机中数的表示**
  - 无符号数和有符号数
  - 定点表示和浮点表示
  - IEEE754标准
  - **算数移位与逻辑移位**

移位运算

- 移位的意义

15.m = 1500.cm

小数点右移 2 位

机器用语　　15 相对于小数点 左移 2 位

（ 小数点不动 ）

左移　　绝对值扩大

右移　　绝对值缩小

- 在计算机中，移位与加减配合，能够实现乘除运算

算术移位规则

- 符号位不变
- 机器数的移位 与 真值移位 一致

真值	码　制	添补代码
正数	原码、补码、反码	0
负数	原　码	0
	补　码	左移 添 0
		右移 添 1
	反　码	1

$$y = 0.y_1y_2\ldots y_n100\ldots00$$

$$y = -0.y_1y_2\ldots y_n100\ldots00$$

$$[y]_{原} = 1.y_1y_2\ldots y_n100\ldots00$$

$$[y]_{反} = 1.\overline{y_1}\,\overline{y_2}\ldots\overline{y_n}\,011\ldots11$$

$$[y]_{补} = 1.\overline{y_1}\,\overline{y_2}\ldots\overline{y_n}\,011\ldots11\text{的末位}+1$$

$$= 1.\overline{y_1}\,\overline{y_2}\ldots\overline{y_n}\,100\ldots00$$

- 例17. 设机器数字长为 8 位（含 1 位符号位），写出 $A = +26$ 时，三种机器数左、右移一位和两位后的表示形式及对应的真值，并分析结果的正确性。

解： $A = +26 = +11010$

则 $[A]_{原} = [A]_{补} = [A]_{反} = 0,0011010$

移位操作	机　器　数	对应的真值
	$[A]_{原}=[A]_{补}=[A]_{反}$	
移位前	0,0011010	+26
左移一位	0,0110100	+52
左移两位	0,1101000	+104
右移一位	0,0001101	+13
右移两位	0,0000110	+6

- 例18. 设机器数字长为 8 位（含 1 位符号位），写出 $A = -26$时，三种机器数左、右移一位和两位后的表示形式及对应的真值，并分析结果的正确性。

解： $A = -26 = -11010$

原码

移位操作	机　器　数	对应的真值
移位前	1,0011010	– 26
左移一位	1,0110100	– 52
左移两位	1,1101000	– 104
右移一位	1,0001101	– 13
右移两位	1,0000110	– 6

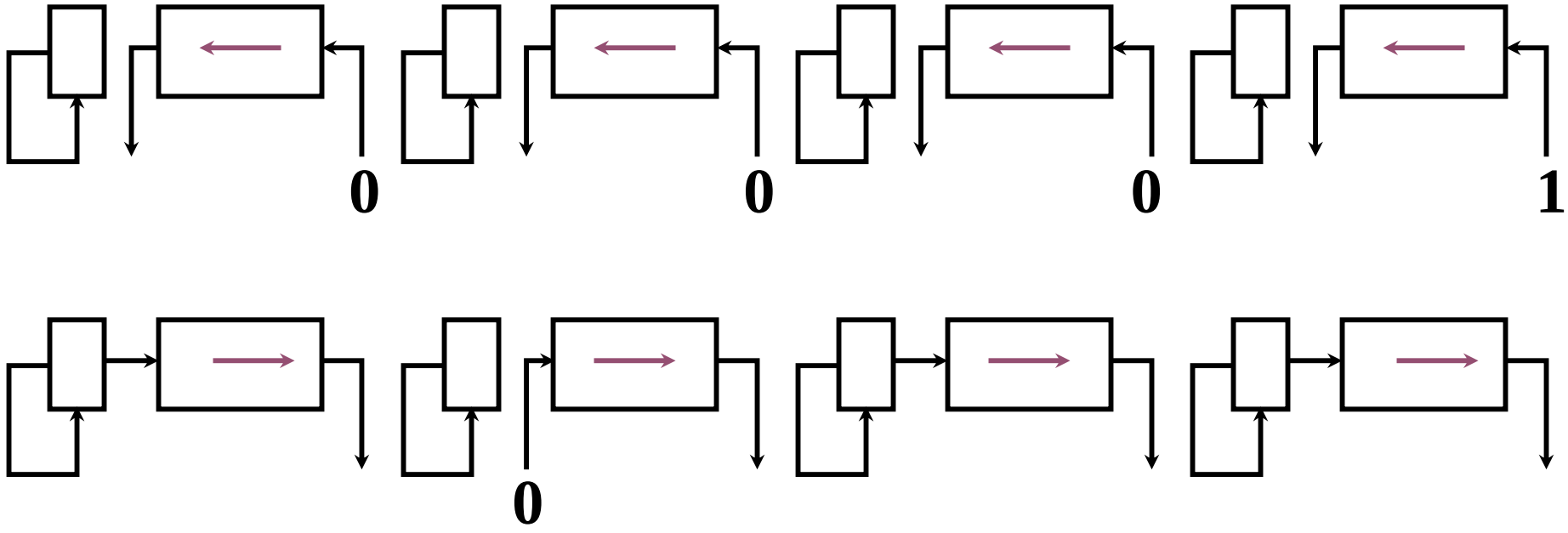
补码

移位操作	机　器　数	对应的真值
移位前	1,1100110	– 26
左移一位	1,1001100	– 52
左移两位	1,0011000	– 104
右移一位	1,1110011	– 13
右移两位	1,1111001	– 7

反码

移位操作	机　器　数	对应的真值
移位前	1,1100101	– 26
左移一位	1,1001011	– 52
左移两位	1,0010111	– 104
右移一位	1,1110010	– 13
右移两位	1,1111001	– 6

3. 算术移位的硬件实现



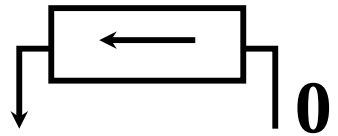
	（a）真值为正	（b）负数的原码	（c）负数的补码	（d）负数的反码
←丢 1	出错	出错	正确	正确
→丢 1	影响精度	影响精度	影响精度	正确

算术移位和逻辑移位的区别

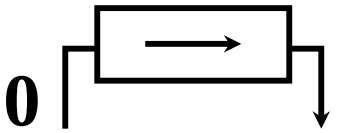
算术移位　　有符号数的移位

逻辑移位　　无符号数的移位

逻辑左移　　低位添 0，高位移丢



逻辑右移　　高位添 0，低位移丢



例如　　01010011　　10110010

逻辑左移　　10100110　　逻辑右移　　01011001

算术左移　　00100110　　算术右移　　11011001（补码）

高位 1 移丢