计算机组成原理

第三讲

张展

哈尔滨工业大学计算学部

容错与移动计算研究中心

第 6 章 计算机的运算方法

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6.11　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6.12　已知 $[y]_{补}$　　求$[-y]_{补}$　　

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

<I>　$\boxed{[y]_{补} = 0. y_1 y_2 \cdots y_n}$

$[y]_{补}$连同符号位在内，　每位取反，　末位加 1

即得$[-y]_{补}$

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

<II>　$\boxed{[y]_{补} = 1. y_1 y_2 \cdots y_n}$

$[y]_{补}$连同符号位在内，　每位取反，　末位加 1

即得$[-y]_{补}$

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

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

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

$x + 2^5$

$+10101 + 100000 = 110101$ 大

$-10101 + 100000 = 001011$ 正确

$+11111 + 100000 = 111111$ 大

$-11111 + 100000 = 000001$ 正确

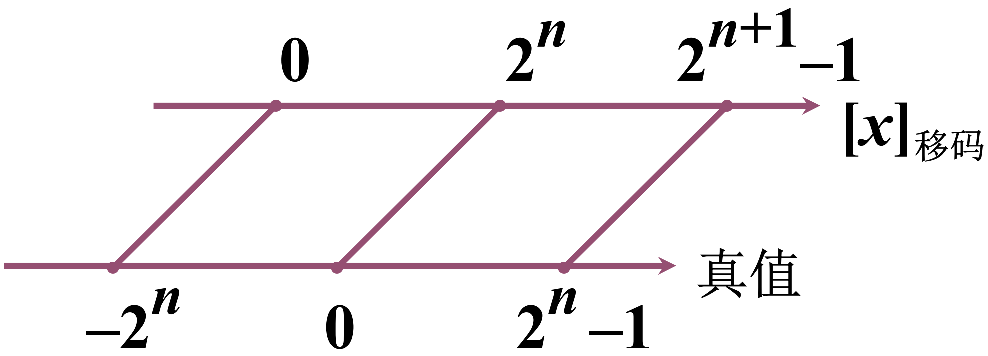
(1) 移码定义

6.1

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

x 为真值，n 为 整数的位数

移码在数轴上的表示



如　　$x = 10100$

$$[x]_{移} = 2^5 + 10100 = 1,10100$$

$$x = -10100$$

$$[x]_{移} = 2^5 - 10100 = 0,01100$$

用 逗号 将符号位和数值部分隔开

(2) 移码和补码的比较

设 $x = +1100100$

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

$$[x]_{补} = 0,1100100$$

设 $x = -1100100$

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

$$[x]_{补} = 1,0011100$$

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)	$[x]_{补}$	$[x]_{移}$	$[x]_{移}$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

(4) 移码的特点

- 当 $x = 0$ 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 最小的真值为 $-2^5 = -100000$

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

  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

6.2 数的定点表示和浮点表示

小数点按约定方式标出

一、定点表示

$$\boxed{S_f}\boxed{\underbrace{S_1S_2\cdots S_n}_{\text{数值部分}}}$$

数符　　小数点位置

或

$$\boxed{S_f}\boxed{\underbrace{S_1S_2\cdots S_n}_{\text{数值部分}}}$$

数符　　小数点位置

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

$N = S \times r^j$ 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 $r = 2$

$N = 11.0101$

二进制表示

✓ $= 0.110101 \times 2^{\boxed{10}}$ 规格化数

$= 1.10101 \times 2^{1}$

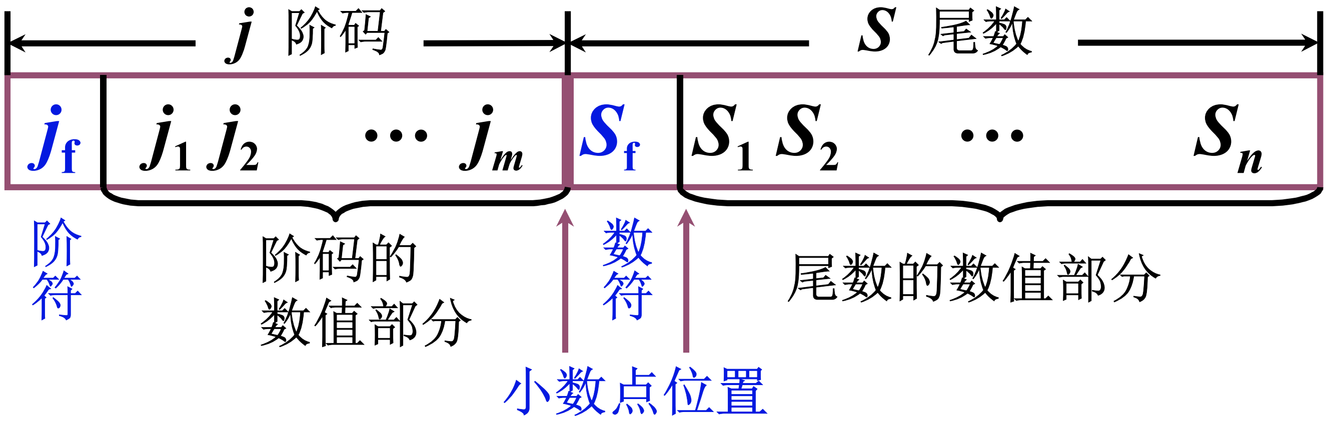
$= 1101.01 \times 2^{-10}$

✓ $= 0.00110101 \times 2^{100}$

计算机中 S 小数、可正可负

j 整数、可正可负

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

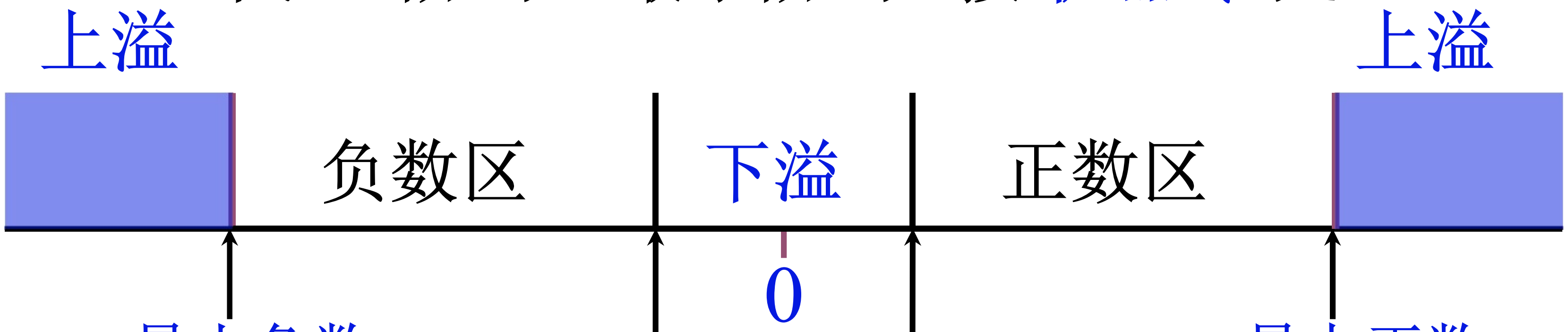
m　　其位数反映浮点数的表示范围

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



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 $m=4$

$n=10$

练习 6.2

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： $\because \quad 2^{14} = 16384 \qquad 2^{15} = 32768$

$\therefore$ 如果是定点数15 位二进制数可反映 ±3 万之间的十进制数

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

$m = 4，5，6，\cdots$

满足 最大精度 可取 $m = 4，n = 18$

3. 浮点数的规格化形式

6.2

$r = 2$ 尾数最高位为 1

$r = 4$ 尾数最高 2 位不全为 0

$r = 8$ 尾数最高 3 位不全为 0

基数不同，浮点数的规格化形式不同

4. 浮点数的规格化

$r = 2$ 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

基数 r 越大，可表示的浮点数的范围越大

基数 r 越大，浮点数的精度降低

例如：设 $m = 4$，$n = 10$，$r = 2$ **6.2**

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

三、举例

6.2

例 6.13　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 $x = +\frac{19}{128}$

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　$[x]_{原} = [x]_{补} = [x]_{反} = 0.0010011000$

浮点机中　$[x]_{原} = 1, 0010;\ 0.1001100000$

$[x]_{补} = 1, 1110;\ 0.1001100000$

$[x]_{反} = 1, 1101;\ 0.1001100000$

例 6.14 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 $x = -0000\,111010$

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

定点机中

$[x]_{原} = 1, 0000111010$

$[x]_{补} = 1, 1111000110$

$[x]_{反} = 1, 1111000101$

浮点机中

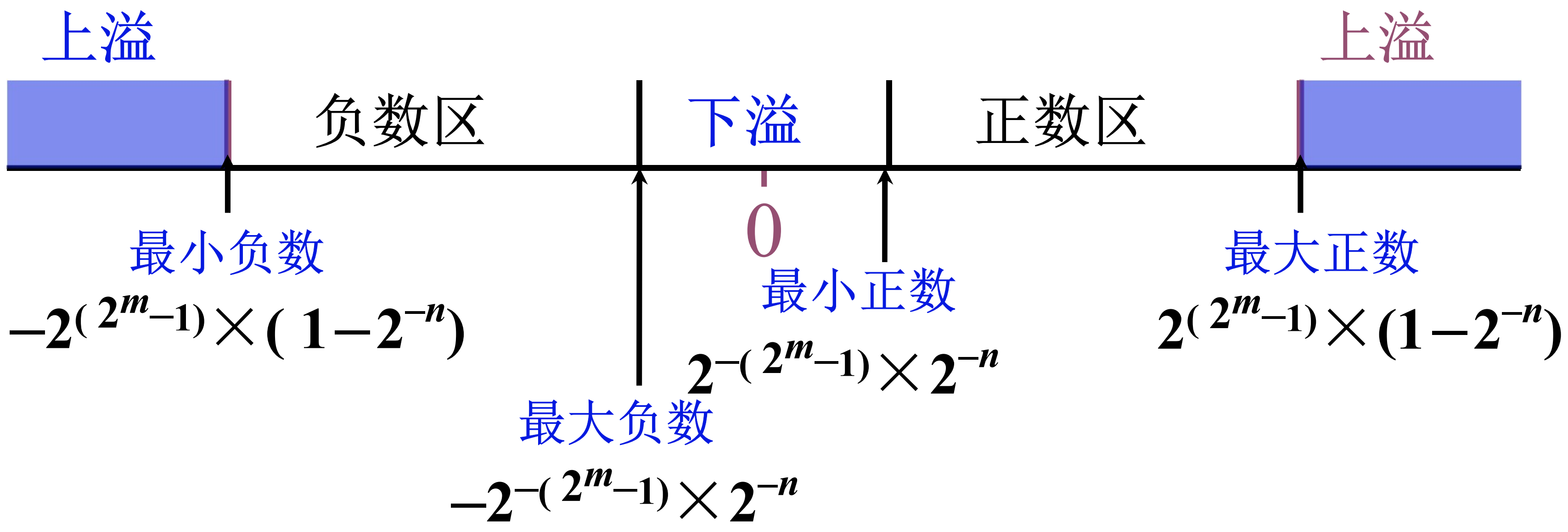
$[x]_{原} = 0, 0110; 1.1110100000$

$[x]_{补} = 0, 0110; 1.0001100000$

$[x]_{反} = 0, 0110; 1.0001011111$

$[x]_{阶移、尾补} = 1, 0110; 1.0001100000$

例6.15 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	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

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

×, × × × ×;　0. 0 0 ⋯ 0

（阶码 = −16）　1, 0 0 0 0 ;　×.×× ⋯ ×

当阶码用移码，尾数用补码表示时，机器零为

0, 0 0 0 0;　0. 0 0 ⋯ 0

有利于机器中 “ 判 0 ” 电路的实现

四、IEEE 754 标准

S	阶码（含阶符）	尾　　数

数符　　　　　　　　小数点位置

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
短实数	**1**	**8**	**23**	**32**
长实数	**1**	**11**	**52**	**64**
临时实数	**1**	**15**	**64**	**80**

“Father” of the IEEE 754 standard

- 直到80年代初，各个机器内部的浮点数表示格式还没有统一，因而相互不兼容，机器之间传送数据时，带来麻烦
- 1970年代后期，IEEE成立委员会着手制定浮点数标准
- 1985年完成浮点数标准IEEE 754的制定
- 现在所有计算机都采用IEEE 754来表示浮点数

This standard was primarily the work of one person, UC Berkeley math professor William Kahan.

www.cs.berkeley.edu/~wkahan/ieee754status/754story.html

Prof. William Kahan

摘自“深入理解计算机系统”

IEEE 754 Floating Point Standard

规格化数： $+/-1.xxxxxxxxxx_{two} \times 2^{Exponent}$

Single Precision ： (Double Precision is similar)

S	Exponent	Significand
1 bit	8 bits	23 bits

- Sign bit: 1 表示negative ; 0表示 positive
- Exponent（阶码 / 指数）： 全0和全1用来表示特殊值！
  - •SP规格化数阶码范围为0000 0001 (-126) ~ 1111 1110 (127)
  - •bias为127 (single), 1023 (double)

为什么用127？若用128，则阶码范围为多少？

- Significand（尾数）：
  - 规格化尾数最高位总是1，所以隐含表示，省1位
  - 1 + 23 bits （ single），1 + 52 bits （double）

SP: $(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-127)}$

0000 0001 (-127) ～ 1111 1110 (126)

DP: $(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-1023)}$

摘自“深入理解计算机系统”

Ex: Converting Binary FP to Decimal

BEE00000H is the hex. Rep. Of an IEEE 754 SP FP number

1	0111 1101	110 0000 0000 0000 0000 0000

$$(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-127)}$$

- **Sign: 1 => negative**
- **Exponent:**
  - $0111\ 1101_{two} = 125_{ten}$
  - **Bias adjustment: 125 - 127 = -2**
- **Significand:**

  $1 + 1\text{x}2^{-1} + 1\text{x}2^{-2} + 0\text{x}2^{-3} + 0\text{x}2^{-4} + 0\text{x}2^{-5} + \ldots$

  $= 1+2^{-1} + 2^{-2} = 1+0.5 + 0.25 = 1.75$
- **Represents:** $-1.75_{ten}\text{x}2^{-2} = -0.4375$

摘自“深入理解计算机系统”

Ex: Converting Decimal to FP

-12.75

1. Denormalize: -12.75

2. Convert integer part:

$12 = 8 + 4 = 1100_2$

3. Convert fractional part:

$.75 = .5 + .25 = .11_2$

4. Put parts together and normalize:

$1100.11 = 1.10011 \times 2^3$

5. Convert exponent: $127 + 3 = 128 + 2 = 1000\ 0010_2$

1	1000 0010	100 1100 0000 0000 0000 0000

The Hex rep. is C14C0000H

摘自“深入理解计算机系统”

6.3 定 点 运 算

一、移位运算

1. 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

2. 算术移位规则

符号位不变

	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

例6.16

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 $[A]_{原}=[A]_{补}=[A]_{反}$	对应的真值
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6.17

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： $A = -26 = -11010$

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

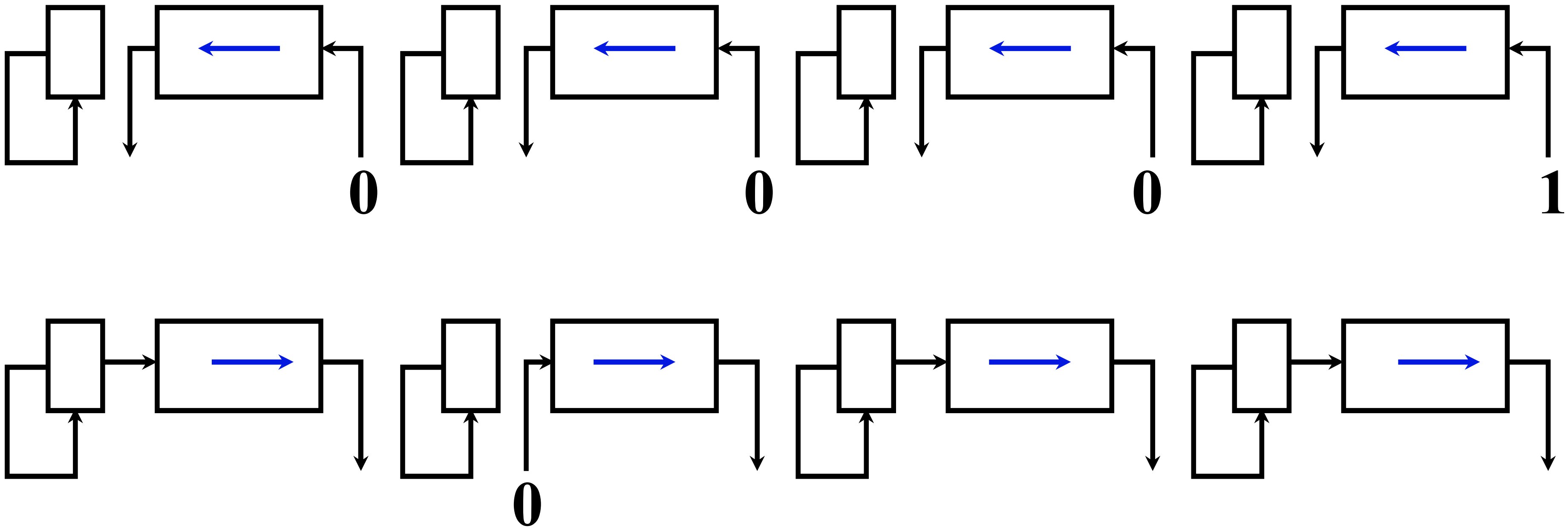
补码

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



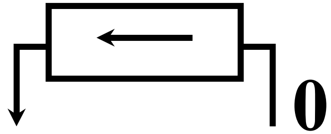
	（a）真值为正	（b）负数的原码	（c）负数的补码	（d）负数的反码
←丢 1	出错	出错	正确	正确
→丢 1	影响精度	影响精度	影响精度	正确

4. 算术移位和逻辑移位的区别 　　6.3

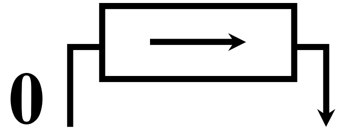
算术移位　　有符号数的移位

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