第6章---计算机的运算方法

- §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

第6章 计算机的运算方法---6.1无符号数和有符号数

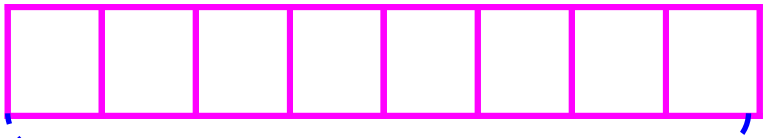
一、无符号数

含义：没有符号的数，所有数位均用以表示数值。

基本常识：机器字长＝＝＝CPU中寄存器的位数

举例：

（1）

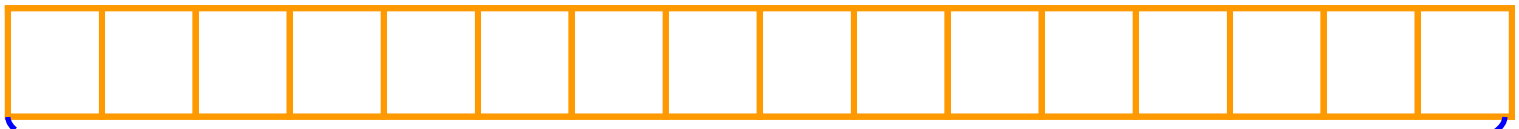


8 位

Min：00000000 ＝ 0

Max：11111111 ＝ 2^8-1 ＝ 255

（2）



16 位

Min：00…0000 ＝ 0

Max：11…1111 ＝ $2^{16}-1$ ＝ 65535

二、有符号数

含义：**用一位作为符号位，其余位为数值。通常** { **0** 表示正数 / **1** 表示负数

（一）机器数与真值

1. 真值：真正的数值，即带“＋”、“－”号的数。

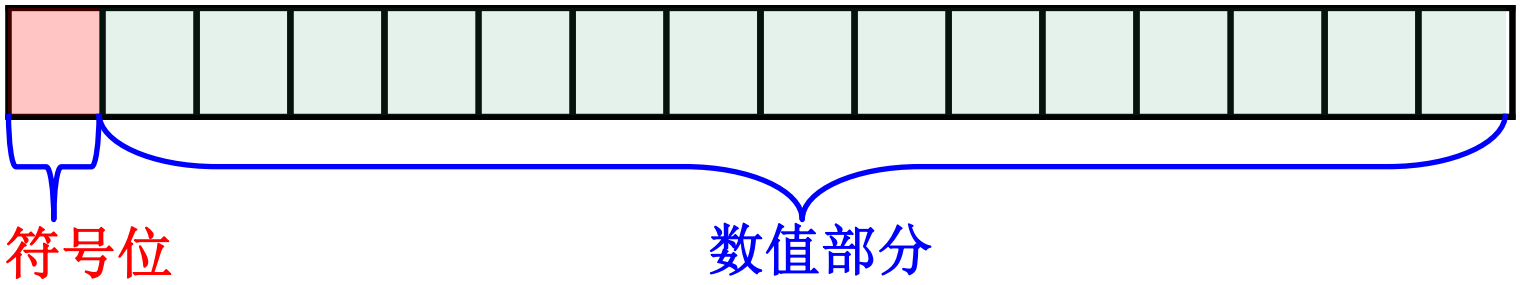
＋0.1001	＋1001
－0.0101	－1100

精髓：（即日常数学计算中所用到的数字表示形式）

思考：如何将这些真值连同符号位在计算机中表示呢？

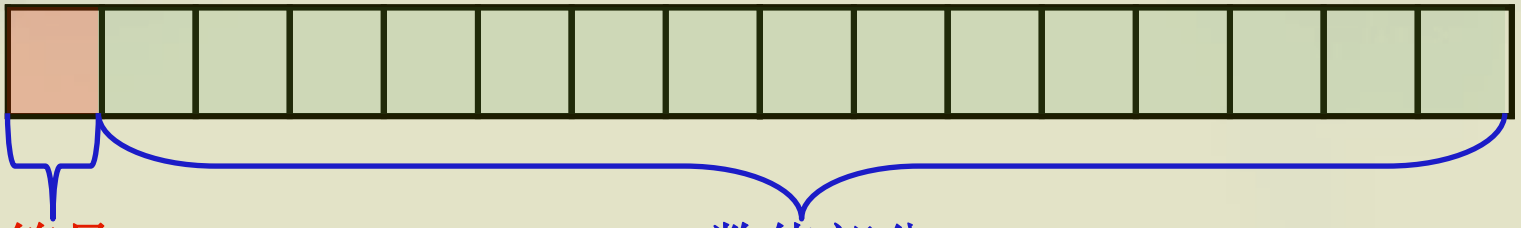
2．机器数： 将符号数字化之后的数。（用于存储在计算机中）

模型：



二、有符号数

模型：



一种新的编码

（保证编码后的数可以进行高效的各种运算）

依据编码方案不同，机器数分为：

- 原码
- 补码
- 反码
- 移码

重难点

（二）数的机器码表示

1. 原码表示法

(1) 定点小数

①公式

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

其中：X：真值

$[x]_{原}$：机器数的原码表示形式

②示例

X = +0.1001，则$[x]_{原}$ = 0.1001

X = −0.1001，则$[x]_{原}$ = 1 + 0.1001 = 1.1001

③规律

若X = +0.x_1x_2……x_n → 则$[x]_{原}$ = 0.x_1x_2……x_n

若X = - 0.x_1x_2……x_n → 则$[x]_{原}$ = 1.x_1x_2……x_n

若x = 0 → 则$[x]_{原}$ = ？？？？？？

（二）数的机器码表示

1. 原码表示法

(2) 定点整数

①公式

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

其中：
- x 为真值
- $[x]_{原}$为机器数的原码表示形式
- n 为整数的位数

②示例

X = + 1001，则$[x]_{原}$ = 0，1001

X = −1001，则$[x]_{原}$ = 10000 −（ − 1001） = 1，1001

③规律

若X = $+x_1x_2\cdots\cdots x_n$ → 则$[x]_{原}$ = 0，$x_1x_2\cdots\cdots x_n$

若X = $-x_1x_2\cdots\cdots x_n$ → 则$[x]_{原}$ = 1，$x_1x_2\cdots\cdots x_n$

若x = 0 → 则$[x]_{原}$ = ？？？？？？

（二）数的机器码表示

1. 原码表示法

(3) 综合举例

原码 ⟺ 真值

例 1　已知 $[x]_{原} = 1.0011$　　求 x

解：　由定义得

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

例 2　已知 $[x]_{原} = 1.0101$　　求 x

解：　由定义得

$$x = 1 - [x]_{原} = 1 - 1.0101 = -0.0101$$

例 3　已知 $[x]_{原} = 1，1010$　　求 x

解：　由定义得

$$x = 2^4 - [x]_{原} = 10000 - 1,1010 = -1010$$

（二）数的机器码表示

1. 原码表示法

(3) 综合举例

原码 ⟺ 真值

例 4　已知 $[x]_{原}=1,1100$求 x

解：　由定义得

$$x=2^4-[x]_{原}=10000-1,1100=-1100$$

例 5　求 $x=0$ 的原码

解：

设 $x=+0.0000$　　$[+0.0000]_{原}=0.0000$

$x=-0.0000$　　$[-0.0000]_{原}=1.0000$

同理，对于整数

$[+0]_{原}=0,0000$

$[-0]_{原}=1,0000$

结论：$[+0]_{原}\neq[-0]_{原}$

（二）数的机器码表示

1. 原码表示法

(4) 原码特点

优点：简单、直观、易懂

缺点：加法运算复杂
- 同号则直接相加
- 异号，则要进行减法运算。比较绝对值的大小然后大的减去小的，最后还要给结果加上恰当的符号。

能否找到一个与负数相等价的正数呢？？

减法运算转换成加法运算

（二）数的机器码表示

2. 补码表示法

(1) 补数的概念

时钟

欲表示3点钟

方法1：逆时针旋转 3 格

方法2：顺时针旋转 9 格

可见：−3 相互等价 +9

我们称：+9 是 –3 以 12 为模的补数

记作：$-3 \equiv +9 \pmod{12}$

同理：$-4 \equiv +8 \pmod{12}$

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

结论：确定了“模”，一个负数有其相等价的正数减法运算 转化为 加法运算

（二）数的机器码表示

2. 补码表示法

(1) 补数的概念

例1：X = 8 ， Y = 3，
求X － Y （mod 10）

解：
8 －3 = 8 + 7
= 15 （mod 10）
= （ + 5） （mod 10）
= 5

丢掉

例2：X = 1101 ， Y= 1001 ，
求X － Y（mod 2^4）

解：
1101 － 1001 = 1101 + 0111
= 10100 （mod 2^4）
= + 0100 （mod 2^4）
= 0100

丢掉

结论：
①一个负数加上 “模” 即得该负数的补数。
②两个互为补数的数 它们绝对值之和即为 模数。
③正数的补数即为其本身。

（二）数的机器码表示

2. 补码表示法

(1) 定点小数

①定义

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

其中 x 为真值；$[x]_{补}$ 为机器数的补码表示形式

② 例1：已知x = 0.1100，则 $[x]_{补}$ = 0.1100

例2：已知x = −0.1100，则 $[x]_{补}$ = 2 − 0.1100 = 1.0100

例3：已知 $[x]_{补}$ = 0.1010 则 x = 0.1010

例4：已知 $[x]_{补}$ = 1.1010 则 x = −0.0110

例5：$[+0.0000]_{补}$ = 0.0000

例6：$[-0.0000]_{补}$ = 10.0000−0.0000 = 10.0000（mod2）=0.0000

（二）数的机器码表示

2. 补码表示法

(2) 定点整数

①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 为整数的位数
- $[x]_{补}$为机器数的补码形式

②示例

例1：已知 x ＝ ＋1001 ，则$[x]_{补}$ ＝ 0，1001

例2：已知 x ＝ －1001 ，则$[x]_{补}$ ＝

$2^{4+1}-1001$
＝ 100000
－1001
1，0111

（二）数的机器码表示　　(3)求补码的快捷方式

① 若$x = +0.x_1x_2\ldots x_n$，则 $[x]_{原} = 0.x_1x_2\ldots x_n$

$[x]_{补} = 0.x_1x_2\ldots x_n$

若$x = -0.x_1x_2\ldots x_n$，则 $[x]_{原} = 1.x_1x_2\ldots x_n$

$[x]_{补} = 1.\overline{x_1}\,\overline{x_2}\ldots\overline{x_n} + 0.00\ldots1$

② 若$x = +x_1x_2\ldots x_n$，则 $[x]_{原} = 0，x_1x_2\ldots x_n$

$[x]_{补} = 0，x_1x_2\ldots x_n$

若$x = -x_1x_2\ldots x_n$，则 $[x]_{原} = 1，x_1x_2\ldots x_n$

$[x]_{补} = 1,\overline{x_1}\,\overline{x_2}\ldots\overline{x_n} + 1$

③ 原码 负数 符号位除外，各位取反，末尾加1 → 负数 补码

负数 符号位除外，各位取反，末尾加1 ←

$[x]_{补}$ 符号位在内，各位取反，末尾加1 → $[-x]_{补}$

$[x]_{补}$ ← 符号位在内，各位取反，末尾加1 $[-x]_{补}$

（二）数的机器码表示

2. 补码表示法

(4) 综合举例

例1

设 $x = -1010$ 时

则 $[x]_{补} = 2^{4+1} - 1010$

$= 100000 - 1010$

$= 1,0110$

又 $[x]_{原} = 1,1010$

$= 11111 + 1 - 1010$

$= 11111 + 1 - 1010$

$= 10101 + 1$

$= 1,0110$

相反

结论： 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

（二）数的机器码表示

(4) 综合举例

例3

已知 $[x]_{补} = 1.1011$,则 $[-x]_{补} =$ _____

已知 $[x]_{补} = 0.1011$,则 $[-x]_{补} =$ _____

已知 $[-x]_{补} = 1，1011$,则 $[x]_{补} =$ _____

已知 $[-x]_{补} = 0，1011$,则 $[x]_{补} =$ _____

已知 $[x]_{原} = 0.1011$,则 $[-x]_{补} =$ _____

已知 $[x]_{原} = 1.1011$,则 $[-x]_{补} =$ _____

已知 $[x]_{补} = 1，1011$,则 $[-x]_{原} =$ _____

已知 $[x]_{补} = 0，1011$,则 $[-x]_{原} =$ _____

(4) 综合举例

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
$x = +0.0000$ $[+0]_{补} = [-0]_{补}$	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 $[x]_{补} = \begin{cases} x & 1 > x \geq 0 \\ 2 + x & 0 > x \geq -1 \ (\text{mod } 2) \end{cases}$

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

（二）数的机器码表示

3.反码表示法

（1）定点小数

①定义

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

其中
- x 为真值
- n为整数的位数
- $[x]_{反}$为机器码的反码表示

②示例

例1：已知 x ＝＋ 0.1101，则 【x】$_{反}$ ＝ 0.1101

例2：已知 x ＝ －0.1001，则 【x】$_{反}$ ＝

$$2-2^{-4}-0.1001$$

$$=\begin{array}{r} 10.0000 \\ -\ 0.0001 \\ \hline 1.1111 \end{array} \qquad \begin{array}{r} 1.1111 \\ -\ 0.1001 \\ \hline 1.0110 \end{array}$$

（二）数的机器码表示

3.反码表示法

（2）定点整数

①定义

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

其中
- x 为真值
- n 为整数的位数
- $[x]_{反}$为机器数的反码表示

②示例

例1：已知 x ＝＋ 1001，则 【x】$_{反}$ ＝ 0，1001

例2：已知 x ＝－ 1001，则 【x】$_{反}$ ＝

$2^{4+1}-1-1001$

100000 − 1001 = 010111

010111 − 1 = 1，0110

（二）数的机器码表示

3.反码表示法

（3）规律

定点小数

若 $x = +0.x_1x_2\cdots\cdots x_n$ 则 $[x]_{反} = 0.x_1x_2\cdots\cdots x_n$

若 $x = -0.x_1x_2\cdots\cdots x_n$ 则 $[x]_{反} = 1.\overline{x_1}\,\overline{x_2}\cdots\cdots\overline{x_n}$

定点整数

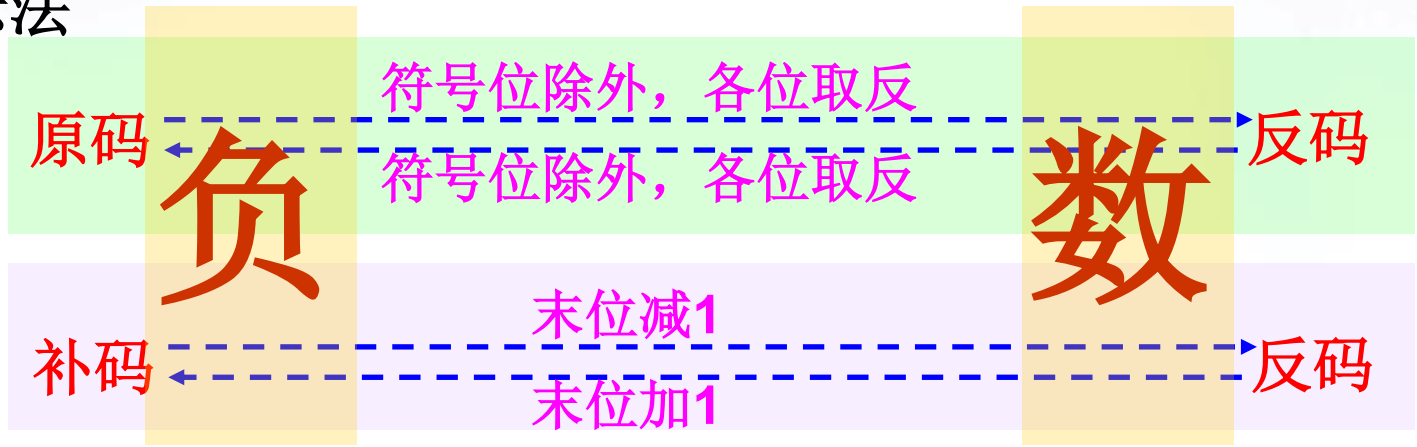
若 $x = +x_1x_2\cdots\cdots x_n$ 则 $[x]_{反} = 0，x_1x_2\cdots\cdots x_n$

若 $x = -x_1x_2\cdots\cdots x_n$ 则 $[x]_{反} = 1，\overline{x_1}\,\overline{x_2}\cdots\cdots\overline{x_n}$

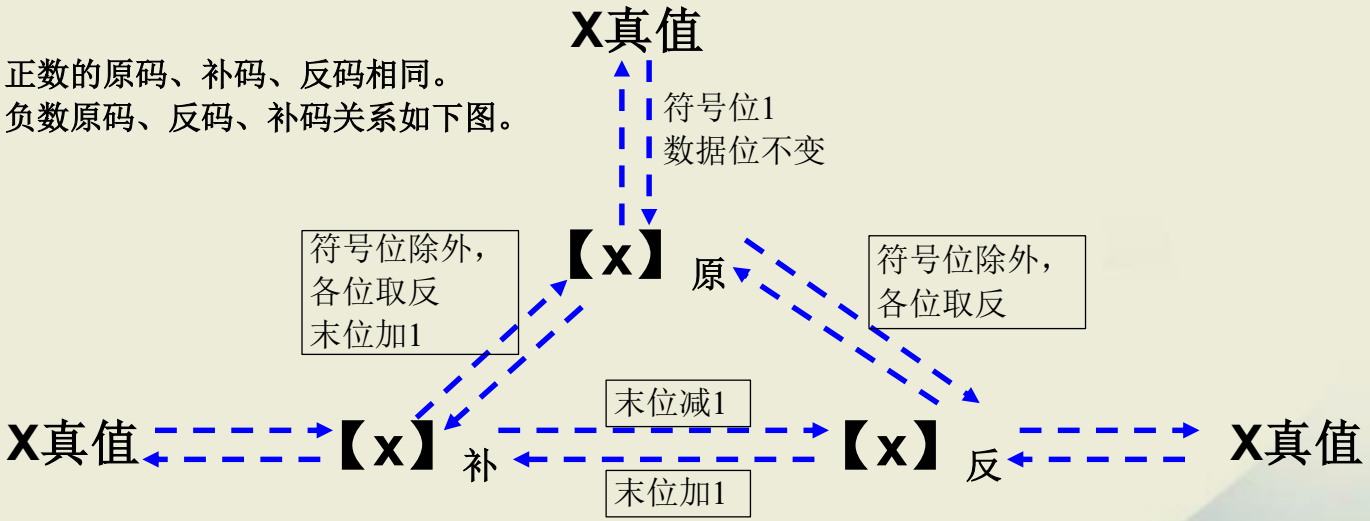
（二）数的机器码表示

3.反码表示法

（3）规律



正数的原码、补码、反码相同。
负数原码、反码、补码关系如下图。



（二）数的机器码表示

3.反码表示法

（4）综合举例

例1　已知 $[x]_{反} = 0,1110$　求 x

解：　由定义得　$x = +1110$

例2　已知 $[x]_{反} = 1,1110$　求 x

解：　由定义得　$x = [x]_{反} - (2^{4+1} - 1)$

$= 1,1110 - 11111$

$= -0001$

例 3　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　$[+0]_{反} = 0,0000$　　$[-0]_{反} = 1,1111$

$\therefore\ [+0]_{反} \neq [-0]_{反}$

（二）数的机器码表示

例4： 设机器数字长为 8 位（其中一位为符号位）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

相同

不相同

主观题 10分

设置

P290-6.4 设机器数字长为8位（含1位符号位在内），写出对应下列各真值的原码、补码和反码。

（1）-13/64

（2）100

正常使用主观题需2.0以上版本雨课堂

作答

主观题 10分

设置

P290-6.5 已知$[x]_{补}$，求$[x]_{原}$和x。

$[x]_{补}$=1.1100;

$[x]_{补}$=1,0101;

正常使用主观题需2.0以上版本雨课堂

作答

（二）数的机器码表示

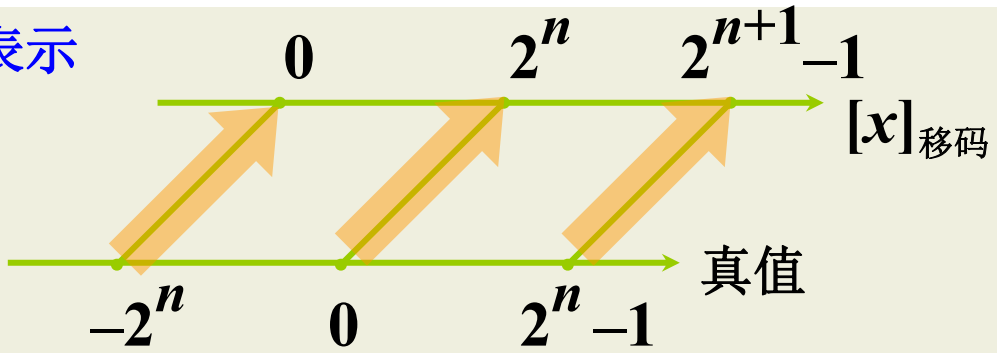
4.移码表示法

（1）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
- $[x]_{移}$为机器数的移码表示

（2）移码在数轴上的表示



（3）示例

如 $x = 10100$

$[x]_{移} = 2^5 + 10100 = 1,10100$

用 逗号 将符号位和数值位隔开

$x = -10100$

$[x]_{移} = 2^5 - 10100 = 0,01100$

（二）数的机器码表示

4.移码表示法

(5)移码的特点

已知 X = 0，则【x】移 =

$[+0]_{移} = 2^5 + 0 = 1,000000$

$[-0]_{移} = 2^5 - 0 = 1,000000$

【+0】移 = 【−0】移

当 $n = 5$ 时　　最小的真值为 -2^5　　$= -100000$

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

可见，最小真值的移码为全 0

移码的符号位：“0” 表示负数；“1”表示正数

(6)移码的应用

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

（二）数的机器码表示

真值、补码和移码的对照表

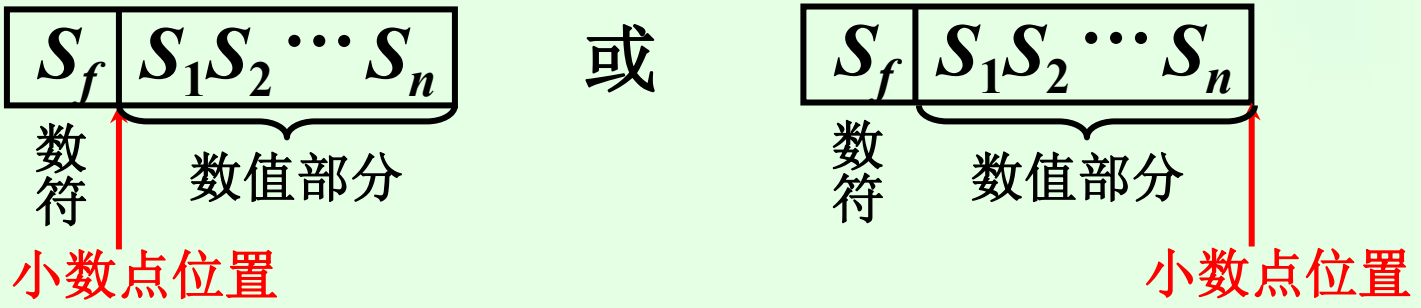
真值 x（n=5）	$[x]_{补}$	$[x]_{移}$	$[x]_{移}$对应的十进制整数
-10000	10000	000000	0
- 11111	10001	000001	1
- 11110	10010	000010	2
⋮	⋮	⋮	⋮
- 00001	11111	011111	31
± 00000	00000	100000	32
+ 00001	00001	100001	33
+ 00010	00010	100010	34
⋮	⋮	⋮	⋮
+ 11110	01110	111110	62
+ 11111	01111	111111	63

对应

第6章 计算机的运算方法---6.2数的定点表示和浮点表示

常识 在计算机中，小数点不用专门的器件表示，而是按约定方式标出。

一、定点表示（小数点固定在某一个位置）



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

1.含义 小数点的位置是浮动的，不是固定的。

2.示例

$N = 111.010101011$

$N = 11.1010101011 \times 2^1$

$N = 1.11010101011 \times 2^{10}$

$N = 456789.1$

$N = 45.67891 \times 10^4$

$N = 4.567891 \times 10^5$

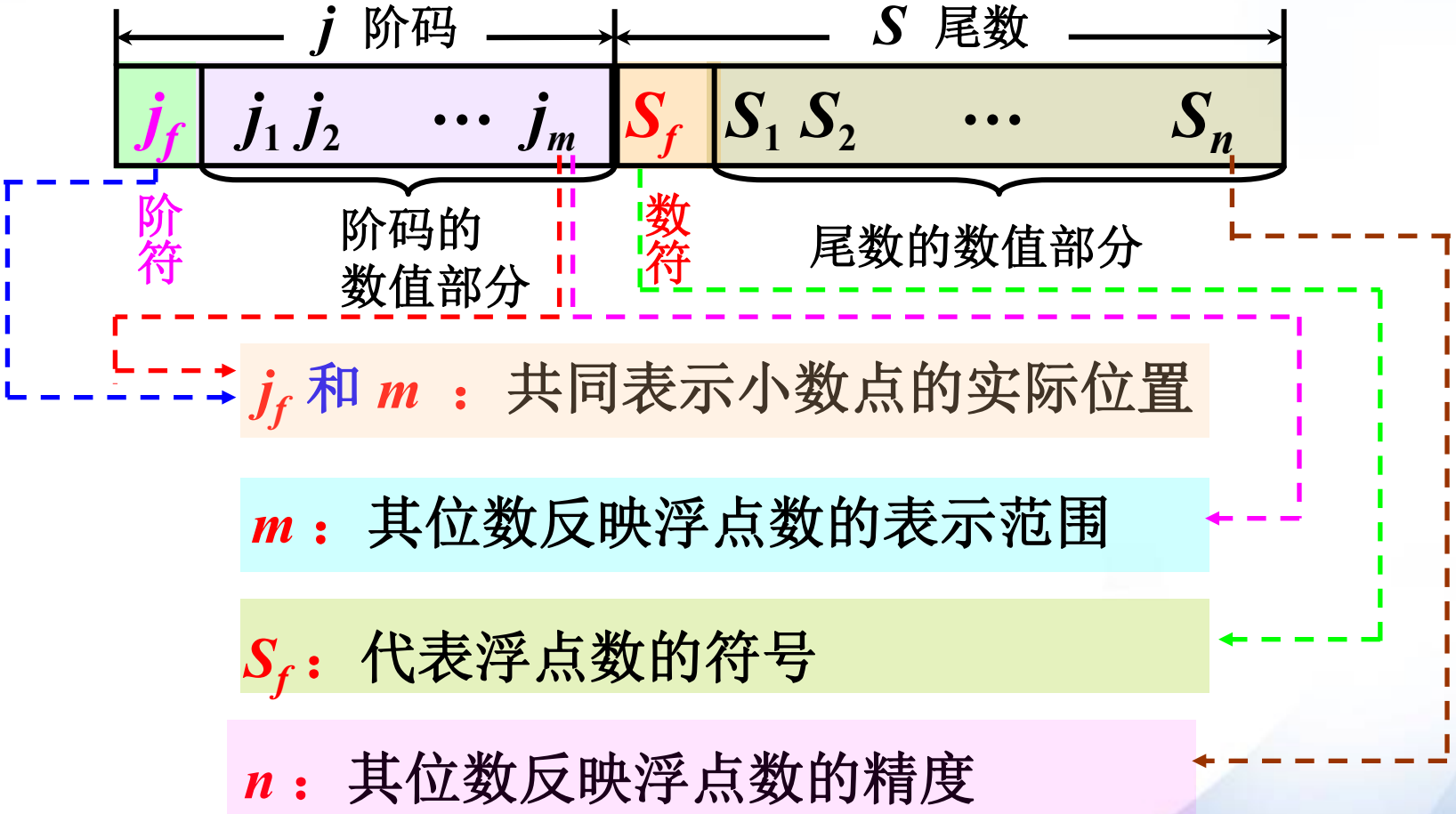
浮点数的一般形式：$N = S \times r^j$

S 尾数　r 基数（基值）　j 阶码

注：
- ①计算机中 r 取 2、4、8、16 等
- ②计算机中　S　小数、可正可负
- ③计算机中 j　整数、可正可负

二、浮点表示

3. 浮点数在计算机中的表示形式（$N = S \times r^j$）



4.浮点数的表示形式

上溢　阶码 > 最大阶玛
下溢　阶码 < 最小阶码　按 机器零 处理

最小负数

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

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

最小正数

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

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

最大负数

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

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

最大正数

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

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

设 $m=4$

$n=10$

5.浮点数的规格化形式

目的： 为了提高精度，统一将浮点数表示成为 $N = 0.11011 \times 2^{10101}$ 这种形式。此时精度最高。

办法：

比如，$N = 0.001011 \times 2^{1101}$

左规：尾数左移 1 位，阶码减 1

规格化形式：0.1011×2^{1011}

比如，$N = 1011.11 \times 2^{1001}$

右规：尾数右移 1 位，阶码加 1

规格化形式：0.101111×2^{1101}

规格化形式

(1) 规格化数的定义

$$r = 2 \qquad 1/2 \le |S| < 1$$

(2) 规格化数的判断

$S>0$	规格化形式	$S<0$	规格化形式
真值	0.1×× ··×	真值	–0.1×× ··×
原码	0.1×× ··×	原码	1.1×× ··×
补码	0.1×× ··×	补码	1.0×× ··×
反码	0.1×× ··×	反码	1.0×× ··×

原码　不论正数、负数，第一数位为1

补码　符号位和第 一数位不同

特例

$$S = -\frac{1}{2} = -0.100\cdots0$$

$$[S]_{原} = 1.100\cdots0$$

$$[S]_{补} = 1.100\cdots0$$

$\therefore$ $[-\frac{1}{2}]_{补}$ 不是规格化的数

$$S = -1$$

$$[S]_{补} = 1.000\cdots0$$

$\therefore$ $[-1]_{补}$ 是规格化的数

第6章 计算机的运算方法---6.2数的定点表示和浮点表示

6.浮点数的规格化形式和非规格化形式比较

(1)当浮点机和定点机中的数其位数相同时：
- 浮点数的表示范围 大
- 定点数表示范围 小

(2)当浮点数为规格化数时：
- 浮点数精度 高
- 定点数精度 低

(3) 从运算来看
- 浮点数运算过程 复杂；定点数运算过程 简单
- 浮点数运算速度 慢；定点数运算速度 快
- 浮点数运算线路 复杂；定点数运算线路 简单

浮点数
- 表示范围
- 表示精度
- 溢出处理
- 程序编写

优越于

优越于

运算规则
运算速度
硬件成本
定点数

7.举例

设浮点数总长度为16位，阶码占5位(含1位阶符)，尾数占11位(含1位数符)

写出$-\frac{87}{512}$对应的浮点数规格化形式的{原码 / 补码 / 反码}及阶码用移码表示，尾数用补码表示的形式。

（1）将$\frac{87}{512}$写成二进制格式

$\frac{87}{512}$ ……扩大512倍……▶ 87

$$87 = 64 + 23$$
$$= 64 + 16 + 7$$
$$= 64 + 16 + 4 + 2 + 1$$

87_{10}——————》1010111_2　　512_{10}——————》1000000000_2

所以$-\frac{87}{512} = -0.001010111$

（2）写成规格化形式

$$-\frac{87}{512} = -0.001010111$$

规格化成

$(-0.1010111)\times 2^{-10}$

阶码	尾数
$[x]_{原}$：1，0010	1.1010111000
$[x]_{补}$：1，1110	1.0101001000
$[x]_{反}$：1，1101	1.0101000111

移码表示方式：0，1110　　1.0101001000

7.举例

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： $x = +\frac{19}{128}$

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	$[x]_{原} = 1, 0010;\ 0.1001100000$
	$[x]_{补} = 1, 1110;\ 0.1001100000$
	$[x]_{反} = 1, 1101;\ 0.1001100000$

7.举例

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）。

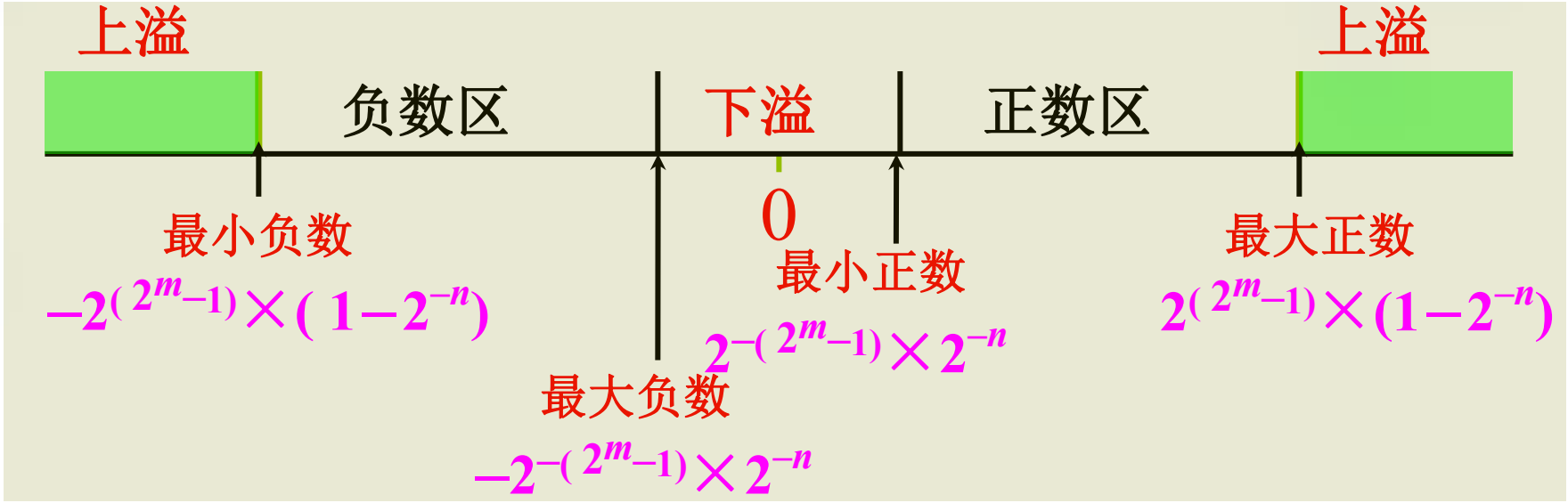
解： $x = -58$

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	浮点机中
$[x]_{原} = 1, 0000111010$	$[x]_{原} = 0, 0110; 1.1110100000$
$[x]_{补} = 1, 1111000110$	$[x]_{补} = 0, 0110; 1.0001100000$
$[x]_{反} = 1, 1111000101$	$[x]_{反} = 0, 0110; 1.0001011111$
	$[x]_{阶移、尾补} = 1, 0110; 1.0001100000$

7.举例

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：

类型	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

课堂练习

课后习题P290-6.12

1-2班：$\frac{51}{128}$；7.375

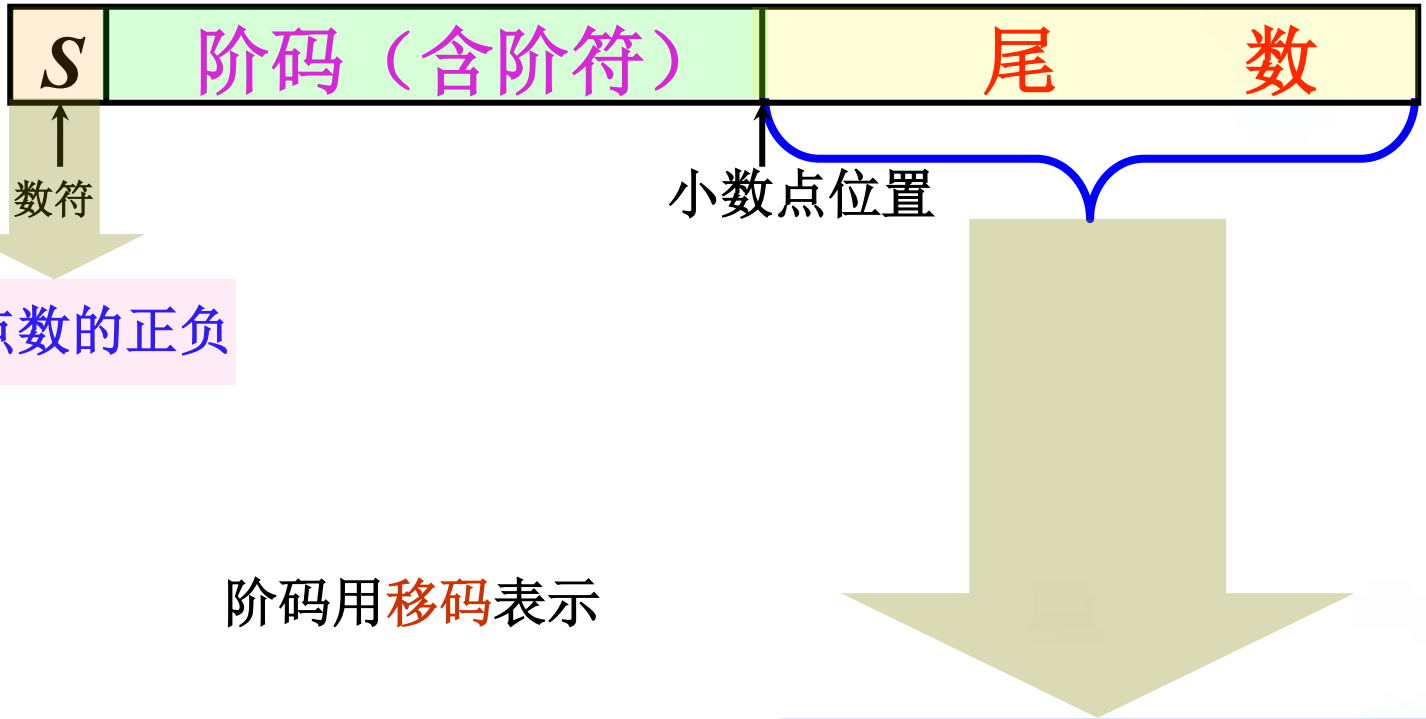
3-4班：$-\frac{27}{1024}$；-86.5

了解

三、IEEE 754 标准

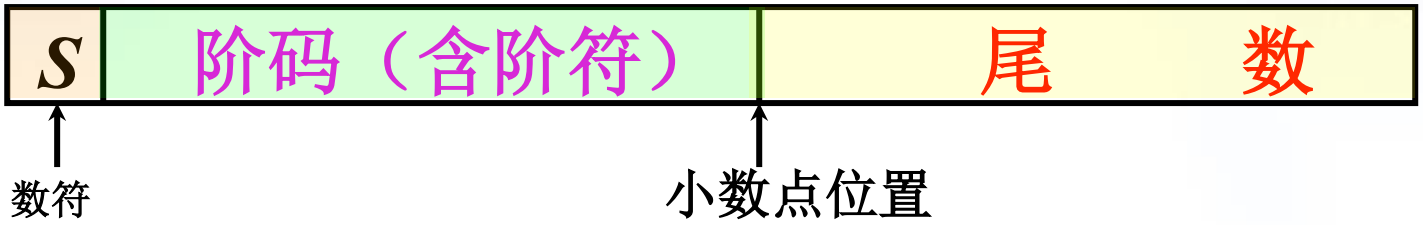
常识：采用浮点数表示形式的计算机中，浮点数一般有规定的格式标准。

IEEE 754 标准



三、IEEE 754 标准

IEEE 754
标准



依据IEEE标准，浮点数有三种类型，如下：

	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

第6章 计算机的运算方法---6.3定 点 运 算

一、移位运算

定点运算（定点数的运算） ⟷ 浮点运算（浮点数的运算）

1. 移位的意义

888_{10}

88800 ← 左移两位

右移两位 → 8.88

101101_2

10110100 ← 左移两位

右移两位 → 1011.01

当某个十进制数相对于小数点左移n位时， 相当于该数乘以10^n；右移n位时，相当于该数除以10^n。

当某个二进制数相对于小数点左移n位时， 相当于该数乘以2^n；右移n位时，相当于该数除以2^n。

意义：在计算机中，移位与加减配合，能够实现乘除运算。

第6章 计算机的运算方法---6.3定 点 运 算

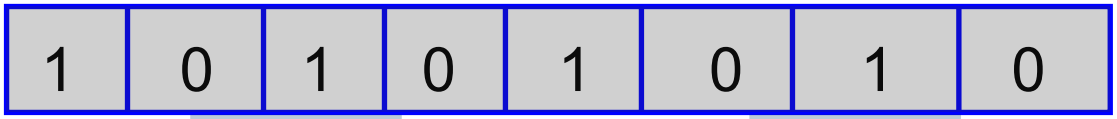
一、移位运算

2. 算术移位规则

（1）含义：有符号数的移位。有符号数 ⟺ 机器码/机器数 { 原码 / 补码 / 反码 }

机器数模型：

给定的机器数



左移怎么移呢？

右移怎么移呢？

？ ？

？ ？

一、移位运算

2. 算术移位规则

总原则：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

第6章 计算机的运算方法---6.3定 点 运 算

一、移位运算

2. 算术移位规则

机器数为正
- $[x]_{原}$ = 真值
- $[x]_{反}$ = 真值　移位后出现的空位填补“0”，符号位不变。
- $[x]_{补}$ = 真值

机器数为负
- 原码：符号位除外，其数值位与真值相同，移位后出现的空位添“0”。
- 反码：符号位除外，其数值位与真值相反，移位后出现的空位添“1”。
- 补码：
  - 左移时添“0”。
  - 右移时添“1”。

一、移位运算

3.例 设机器数字长为 8 位（含一位符号位），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： $A=+26 \quad =+11010$

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
←1	0,0110100	+52
←2	0,1101000	+104
→1	0,0001101	+13
→2	0,0000110	+6

正确

影响精度

第6章 计算机的运算方法---6.3定 点 运 算

一、移位运算

3.例 设机器数字长为 8 位（含一位符号位），写出$A=-43$时的原码、补码、反码及三种机器数左、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真值	−101011
【A】原	1，0101011
【A】补	1，1010101
【A】反	1，1010100

原码

移位操作	机器数	对应真值	
移位前	1，0101011	−43	正确
左移1位	1，1010110	−86	
左移2位	1，0101100	−44	错误
右移1位	1，0010101	−21	影响精度
右移2位	1，0001010	−10	

第6章 计算机的运算方法---6.3定 点 运 算

一、移位运算

补码

移位操作	机器数	对应真值	
移位前	1，1010101	−43	正确
左移1位	1，0101010	−86	
左移2位	1，1010100	−44	错误
右移1位	1，1101010	−22	影响精度
右移2位	1，1110101	−11	

反码

移位操作	机器数	对应真值	
移位前	1，1010100	−43	正确
左移1位	1，0101001	−86	
左移2位	1，1010011	−44	错误
右移1位	1，1101010	−21	影响精度
右移2位	1，1110101	−10	

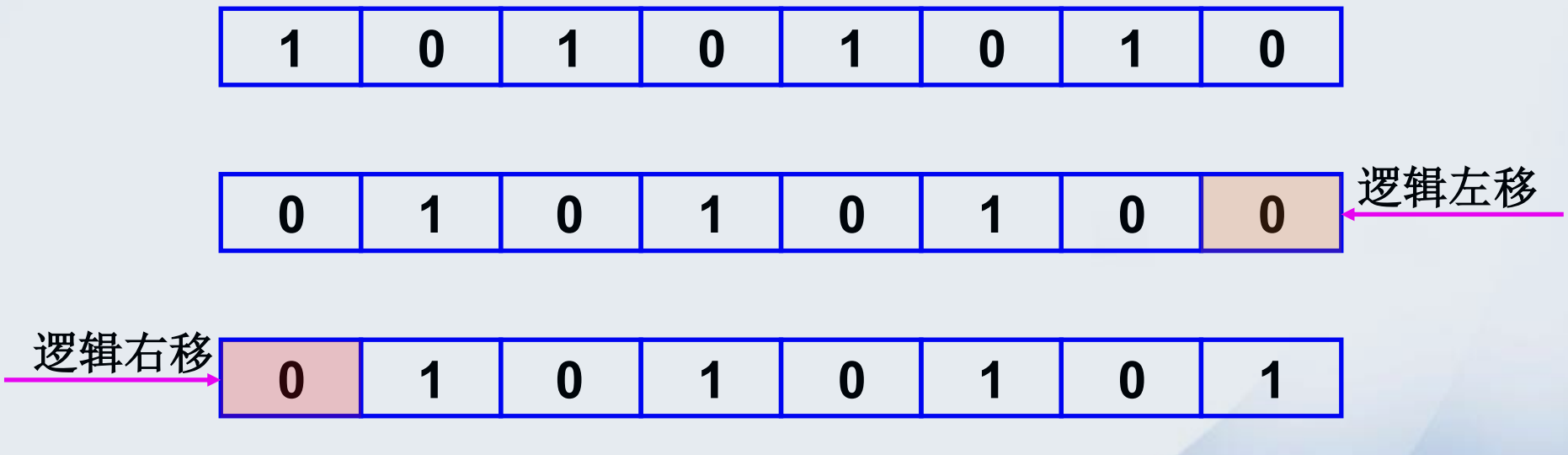
第6章 计算机的运算方法---6.3定 点 运 算

一、移位运算

3. 逻辑移位规则

（1）含义：无符号数的移位

（2）规则：逻辑左移时，高位移出，低位添0；逻辑右移时，低位移出，高位添0



二、加减法运算

1. 补码加减运算公式

（1）含义：

整数 $[A]_{补} + [B]_{补} = [A+B]_{补}\ (\mathrm{mod}\ 2^{n+1})$

小数 $[A]_{补} + [B]_{补} = [A+B]_{补}\ (\mathrm{mod}\ 2)$

精髓：和的补码等于补码的和

两个用补码表示的数在进行加法运算时，可以把符号位与数值位同等处理，运算后的结果按2或2^{n+1}取模，即得本次运算结果。

即在模2(或模2^{n+1})意义下，任意两数的补码之和等于两数之和的补码。

二、加减法运算

推论：

$$【A】_{补} + 【-B】_{补} = 【A-B】_{补} \quad (\mathrm{mod}\ 2^{n+1})$$

$$【A】_{补} + 【-B】_{补} = 【A-B】_{补} \quad (\mathrm{mod}\ 2)$$

二、加减法运算

1. 补码加减运算公式

（3）举例

例1：已知 $x = 0.1001$，$y = 0.0101$

求 $x+y$

解 $[x]_{补} = 0.1001$

$[y]_{补} = 0.0101$

$[x+y]_{补} = [x]_{补} + [y]_{补}$

$$\begin{array}{r} = \quad 0.1001 \\ + \quad 0.0101 \\ \hline 0.1110 \end{array}$$

所以 $[x+y]_{补} = 0.1110$

所以 $x+y = 0.1110$

例2：已知 $x_1 = -0.1110$，$x_2 = +0.1101$

求 $[x_1]_{补}$ $[-x_1]_{补}$ $[x_2]_{补}$ $[-x_2]_{补}$ $x_1 + x_2$

解 $[x_1]_{补} = [-0.1001]_{补} = 1.0010$

$[-x_1]_{补} = [0.1110]_{补} = 0.1110$

$[x_2]_{补} = [0.1101]_{补} = 0.1101$

$[-x_2]_{补} = [-0.1101]_{补} = 1.0011$

$[x_1+x_2]_{补} = [x_1]_{补} + [x_2]_{补}$

$$\begin{array}{r} = \quad 1.0010 \\ + \quad 0.1101 \\ \hline 1.1111 \end{array}$$

所以，$x_1+x_2 < 0$，

$|x_1+x_2| = 2 - [x_1+x_2]_{补} =$

$$\begin{array}{r} = \quad 10.0000 \\ - \quad 1.1111 \\ \hline 0.0001 \end{array}$$

所以，$x1+x2 = -0.0001$

二、加减法运算

1. 补码加减运算公式

例3：已知 x = 0.1011，y = － 0.0101，求x+y

解 $[x]_{补} = 0.1011$

$[y]_{补} = 1.1011$

$[x+y]_{补} = [x]_{补} + [y]_{补}$

$$\begin{array}{r} = \quad 0.1011 \\ + \quad 1.1011 \\ \hline 10.0110 \end{array}$$

10.0110 ＝ 10.0000＋0.0110

所以 $[x+y]_{补} = 0.0110$

所以x＋y ＝ 0.0110

例4：已知 x = 0.1101，y = 0.0110，求x －y

解 $[x]_{补} = 0.1101$

$[-y]_{补} = 1.1010$

$[x-y]_{补} = [x]_{补} + [-y]_{补}$

$$\begin{array}{r} = \quad 0.1101 \\ + \quad 1.1010 \\ \hline 10.0111 \end{array}$$

10.0111 ＝ 10.0000＋0.0111

所以 $[x-y]_{补} = 0.0111$

所以x－y ＝ 0.0111

二、加减法运算

1. 补码加减运算公式

例5：已知 A = －1001 ， B = －0101，求【 A+B 】$_{补}$ 和 A+B。

解

【A】$_{补}$ = 1,0111　　【B】$_{补}$ = 1,1011

【A + B】$_{补}$ = 【A】$_{补}$ + 【B】$_{补}$

= 1,0111

＋ 1,1011

= 11,0010

= 100000 + 1,0010

= 2^5 + 1,0010

所以【A + B】$_{补}$ = 1，0010

所以 A + B = _____

验证：

A = －1001 = －9

B = －0101 = －5

A + B = －9 －5 = －14

【A + B】$_{补}$ = 1，0010

所以，A + B < 0

【A + B】$_{补}$ —求→ A + B

最终求得 A + B = －14

okay

二、加减法运算

1. 补码加减运算公式

例6：设机器数字长为8位(含一位符号位在内)，若A=+15，B=+24，求A-B之值。

解

A = 15 = 8 + 4 + 2 + 1 = 1111　　根据要求，所以【A】$_{补}$ = 0，0001111

B = 24 = 16 + 8 = 11000　　根据要求，所以【B】$_{补}$ = 0，0011000

则【 － B】$_{补}$ = 1,1101000

【A － B】$_{补}$ = 【A】$_{补}$ + 【－B】$_{补}$

$$\begin{array}{rr} & 0，0001111 \\ + & 1，1101000 \\ \hline & 1，1110111 \end{array}$$

最终求得 A － B = －9

二、加减法运算

1. 补码加减运算公式

例7：设机器数字长为8位(含一位符号位在内)，若A=－93，B=+45，求A-B之值。

解

－A = 93 = 64 +16 + 8 + 4＋1 = 1011101　根据要求，所以【A】$_{补}$ ＝ 1，0100011

B = 45 = 32 + 8 ＋ 4＋1 = 101101　根据要求，所以【B】$_{补}$ ＝ 0，0101101

则【 － B】$_{补}$ ＝ 1,1010011

【A － B】$_{补}$ ＝【A】$_{补}$ ＋【－B】$_{补}$

＝

1，0100011

＋ 1，1010011

丢掉 ■ 0，1110110

【A － B】$_{补}$ ＝ 0，1110110

验证：

A － B = －93 －45 = －138

如果【A － B】$_{补}$ ＝ 0，1110110

则 A － B = 118

出错啦！

二、加减法运算

例8：已知 x = +0.1011，y = +0.1001，求x+y。

解

$[x]_{补} = 0.1011 \quad [y]_{补} = 0.1001$

$[x+y]_{补} = [x]_{补} + [y]_{补}$

$$\begin{array}{r} = \quad 0.1011 \\ + \quad 0.1001 \\ \hline = \quad 1.0100 \end{array}$$

$x + y < 0$

出错啦！

例9：已知 x = −0.1101，y = −0.1011，求x+y。

解

$[x]_{补} = 1.0010 \quad [y]_{补} = 1.0101$

$[x+y]_{补} = [x]_{补} + [y]_{补}$

$$\begin{array}{r} = \quad 1.0011 \\ + \quad 1.0101 \\ \hline = \quad 0.1000 \end{array}$$

丢掉

$x + y > 0$

出错啦！

二、加减法运算

2. 溢出判断

➢ 用一位符号位判断溢出

（1）原理

① 对于加法：

- 正数 + 正数
- 负数 + 负数

→ 可能会溢出

- 符号不同的两个数相加

→ 决不会溢出

②对于减法：

- 正数 – 负数
- 负数 – 正数

→ 可能会溢出

- 符号相同的两个数相减

→ 决不会溢出

（2）现象

溢出时

- 两个正数的符号 → 负
- 两个负数的符号 → 正

（3）判断原则

参加操作的 两个数（减法时即为被减数和“求补”以后的减数）符号相同，其结果的符号又与原操作数的符号不同，即为溢出。

2.溢出判断

- 用一位符号位判断溢出

例1： $A=-\frac{11}{16}$，$B=-\frac{7}{16}$，求【A+B】补

解

【A】补 = 1.0101 【B】补 = 1.1001

【A + B】补 = 【A】补 + 【B】补

丢掉 = 1.0101 + 1.1001 = 0.1110

两个操作数符号均为1，结果的符号为0，二者不同故溢出。

例2：A = −0.1001, B = − 0.1011 求A+B.

解

【A】补 = 1.0111 【B】补 = 1.0100

【A + B】补 = 【A】补 + 【B】补

丢掉 = 1.0111 + 1.0101 = 0.1100

两个操作数符号均为1，结果的符号为0，二者不同故溢出。

2. 溢出判断

➢ 1.用一位符号位判断溢出

例3：设A = −0.1000 ,B = − 0.1000　求【A+B】补

解

【A】补 = 1.1000　　【B】补 = 1.1000

【A + B】补 = 【A】补 + 【B】补

丢掉 = 1.1000
+ 1.1000
1.0000

两个操作数符号均为1，结果的符号为1，二者相同故未溢出。

硬件实现 最高有效位的进位 ⊕ 符号位的进位 = 1　溢出

如

$1 \oplus 0 = 1$
$0 \oplus 1 = 1$
} 有 溢出

$0 \oplus 0 = 0$
$1 \oplus 1 = 0$
} 无 溢出

➢ 两位符号位判溢出

例1：x = +0.1100，y = +0.1000，
求 x +y

解

$[x]_{补}$ =00.1100 $[y]_{补}$ = 00.1000

$[x+y]_{补}$ = $[x]_{补}$ + $[y]_{补}$

```
   00.1100
 + 00.1000
 ----------
   01.0100
```

①两个符号位出现“01”，表示已溢出，即结果大于1；

②但最高符号位 “0”，是正确的。

例2：x = −0.1100，y = −0.1000，
求 x +y

解

$[x]_{补}$ =11.0100 $[y]_{补}$ = 11.1000

$[x+y]_{补}$ = $[x]_{补}$ + $[y]_{补}$

```
   11.0100
 + 11.1000
 ----------
  (1)10.1100
```

丢掉

①两个符号位出现“10”，表示已溢出，即结果小于1；

②但最高符号位 “1”，是正确的。

➢ 两位符号位判溢出

例3：$x = +\frac{11}{16}$，$y = +\frac{3}{16}$，

求 x+y

解

$[x]_{补} = 00.1011$ $[y]_{补} = 00.0011$

$[x+y]_{补} = [x]_{补} + [y]_{补}$

$$\begin{array}{r} = \quad 00.1011 \\ + \quad 00.0011 \\ \hline 00.1110 \end{array}$$

① 两个符号位“00”，表示未溢出；

② 结果正确，结果为 0.1110

例4：$x = -\frac{11}{16}$，$y = -\frac{7}{16}$，

求 x+y

解

$[x]_{补} = 11.0101$ $[y]_{补} = 11.1001$

$[x+y]_{补} = [x]_{补} + [y]_{补}$

$$\begin{array}{r} = \quad 11.0101 \\ + \quad 11.1001 \\ \hline 10.1110 \end{array}$$

丢掉

①两个符号位出现“10”，表示已溢出，即结果小于1；

②但最高符号位 “1”，是正确的。

①上述方法及结论对于整数同样试用之。

②机器通过逻辑电路可自动检查出这种溢出，并进行中断处理。

实验安排

（2）算术逻辑运算单元实验

时间：周五上午

1-2节：3-4班

3-4节：1-2班

地点：计算机学科楼403

课堂练习

课后习题P290-6.19

填空题 6分

设置

1、正数补码算术移位时，[填空1]位不变，空位补[填空2]。负数补码算术左移时，[填空3]位不变，低位补[填空4]。负数补码算术右移时，[填空5]位不变，高位补[填空6]。

作答

填空题 4分

设置

2、设寄存器内容为FFH，若其表示127，则为 [填空1] 码，若其表示-127，则为 [填空2] 码，若其表示-1，则为 [填空3] 码，若其表示-0，则为 [填空4] 码。

作答

主观题 10分

设置

3、设机器数字长为8位（含1位符号位），设A=-87，B=53，计算[A±B]补，并还原成真值。

作答

三、乘法运算

1. 分析笔算乘法

$A = -0.1101 \qquad B = 0.1011$

$A \times B = -0.10001111$ 乘积的符号心算求得

```
  0.1101
×0.1011
--------
    1101
   1101
  0000
 1101
--------
0.10001111
```

符号位单独处理

乘数的某一位决定是否加被乘数

? 4个位积一起相加

乘积的位数扩大一倍

2. 笔算乘法改进

$$A \cdot B = A \cdot 0.1011$$

$$= 0.1A + 0.00A + 0.001A + 0.0001A$$

$$= 0.1A + 0.00A + 0.001(A + 0.1A)$$

$$= 0.1A + 0.01[0 \cdot A + 0.1(A + 0.1A)]$$

$$= 0.1\{A + 0.1[0 \cdot A + 0.1(A + 0.1A)]\}$$

右移一位

$$= 2^{-1}\{A + 2^{-1}[0 \cdot A + 2^{-1}(A + 2^{-1}(A + 0))]\}$$

第一步　被乘数A + 0　　①

第二步　右移 一 位，得新的部分积　　②

第三步　部分积 + 被乘数　　③

⋮

第八步　右移 一 位，得结果　　⑧

3. 改进后的笔算乘法过程（竖式）

部 分 积	乘 数	说 明
0.0000 +0.1101	1011 =	初态，部分积 = 0 乘数为 1，加被乘数
0.1101 0.0110 +0.1101	1101 =	→1，形成新的部分积 乘数为 1，加被乘数
1.0011 0.1001 +0.0000	1 1110 =	→ 1，形成新的部分积 乘数为 0，加 0
0.1001 0.0100 +0.1101	11 1111 =	→ 1，形成新的部分积 乘数为 1，加 被乘数
1.0001 0.1000	111 1111	→1，得结果

•小结

➢ 乘法 运算可用 加和移位实现

n = 4，加 4 次，移 4 次

➢由乘数的末位决定被乘数是否与原部分积相加，然后⟶1 位形成新的部分积，同时 乘数 ⟶1 位（末位移丢），空出高位存放部分积的低位。

➢ 被乘数只与部分积的高位相加

•硬件 •3 个寄存器，具有移位功能

•1 个全加器

5. 补码乘法

(1) 补码一位乘运算规则

以小数为例　设 被乘数　$[x]_{补} = x_0.\, x_1x_2 \cdots \quad x_n$

乘数　　$[y]_{补} = y_0.\, y_1y_2 \cdots \quad y_n$

① 被乘数任意，乘数为正

同原码乘　　但 加 和 移位 按 补码规则 运算

乘积的符号自然形成

② 被乘数任意，乘数为负

乘数$[y]_{补}$，去掉符号位，操作同 ①

最后 加$[-x]_{补}$，校正

③ Booth 算法 （被乘数、乘数符号任意）

设$[x]_{补}=x_0.x_1x_2\cdots x_n$　　$[y]_{补}=y_0.y_1y_2\cdots y_n$

$[x\cdot y]_{补}$

$-[x]_{补}=+[-x]_{补}$

$=[x]_{补}(0.y_1\cdots y_n)-[x]_{补}\cdot y_0$

$=[x]_{补}(y_1\,2^{-1}+y_2 2^{-2}+\cdots+y_n 2^{-n})-[x]_{补}\cdot y_0$

$2^{-1}=2^0-2^{-1}$

$=[x]_{补}(-y_0+y_1\,2^{-1}+y_2 2^{-2}+\cdots+y_n 2^{-n})$

$2^{-2}=2^{-1}-2^{-2}$

$=[x]_{补}[-y_0+(y_1-y_1 2^{-1})+(y_2 2^{-1}-y_2 2^{-2})+\cdots+(y_n 2^{-(n-1)}-y_n 2^{-n})]$

$=[x]_{补}[(y_1-y_0)+(y_2-y_1)2^{-1}+\cdots+(y_n-y_{n-1})2^{-(n-1)}+(0-y_n)2^{-n})]$

$=[x]_{补}[(y_1-y_0)+(y_2-y_1)2^{-1}+\cdots+(y_{n+1}-y_n)2^{-n}]$

附加位 y_{n+1}

$y_1\,2^{-1}+\cdots+y_n\,2^{-n}$

④ Booth 算法递推公式

$$[z_0]_{补} = 0$$

$$[z_1]_{补} = 2^{-1}\{(y_{n+1} - y_n)[x]_{补} + [z_0]_{补}\} \qquad y_{n+1} = 0$$

$$\vdots$$

$$[z_n]_{补} = 2^{-1}\{(y_2 - y_1)[x]_{补} + [z_{n-1}]_{补}\}$$

$$[x \cdot y]_{补} = [z_n]_{补} + (y_1 - y_0)[x]_{补}$$

最后一步不移位

如何实现 $y_{i+1} - y_i$?

y_i y_{i+1}	$y_{i+1} - y_i$	操作
0 0	0	→1
0 1	1	$+[x]_{补}$ →1
1 0	-1	$+[-x]_{补}$ →1
1 1	0	→1

例6.23　已知 $x = +0.0011$　$y = -0.1011$　求$[x \cdot y]_{补}$

解：

	00.0000	1.0101	0	
	+ 11.1101			$+[-x]_{补}$
补码右移	11.1101			
	11.1110	1 1010	1	→1
	+ 00.0011			$+[x]_{补}$
补码右移	00.0001	1		
	00.0000	11 101	0	→1
	+ 11.1101			$+[-x]_{补}$
补码右移	11.1101	11		
	11.1110	111 10	1	→1
	+ 00.0011			$+[x]_{补}$
补码右移	00.0001	111		
	00.0000	1111 1	0	→1
	+ 11.1101			$+[-x]_{补}$
	11.1101	1111		最后一步不移位

$[x]_{补} = 0.0011$

$[y]_{补} = 1.0101$

$[-x]_{补} = 1.1101$

$\therefore [x \cdot y]_{补}$

$= 1.11011111$

主观题 10分

设置

P291-6.20 用补码一位乘（Booth 算法）计算 x*y。

（3）x= 19，y= 35

正常使用主观题需2.0以上版本雨课堂

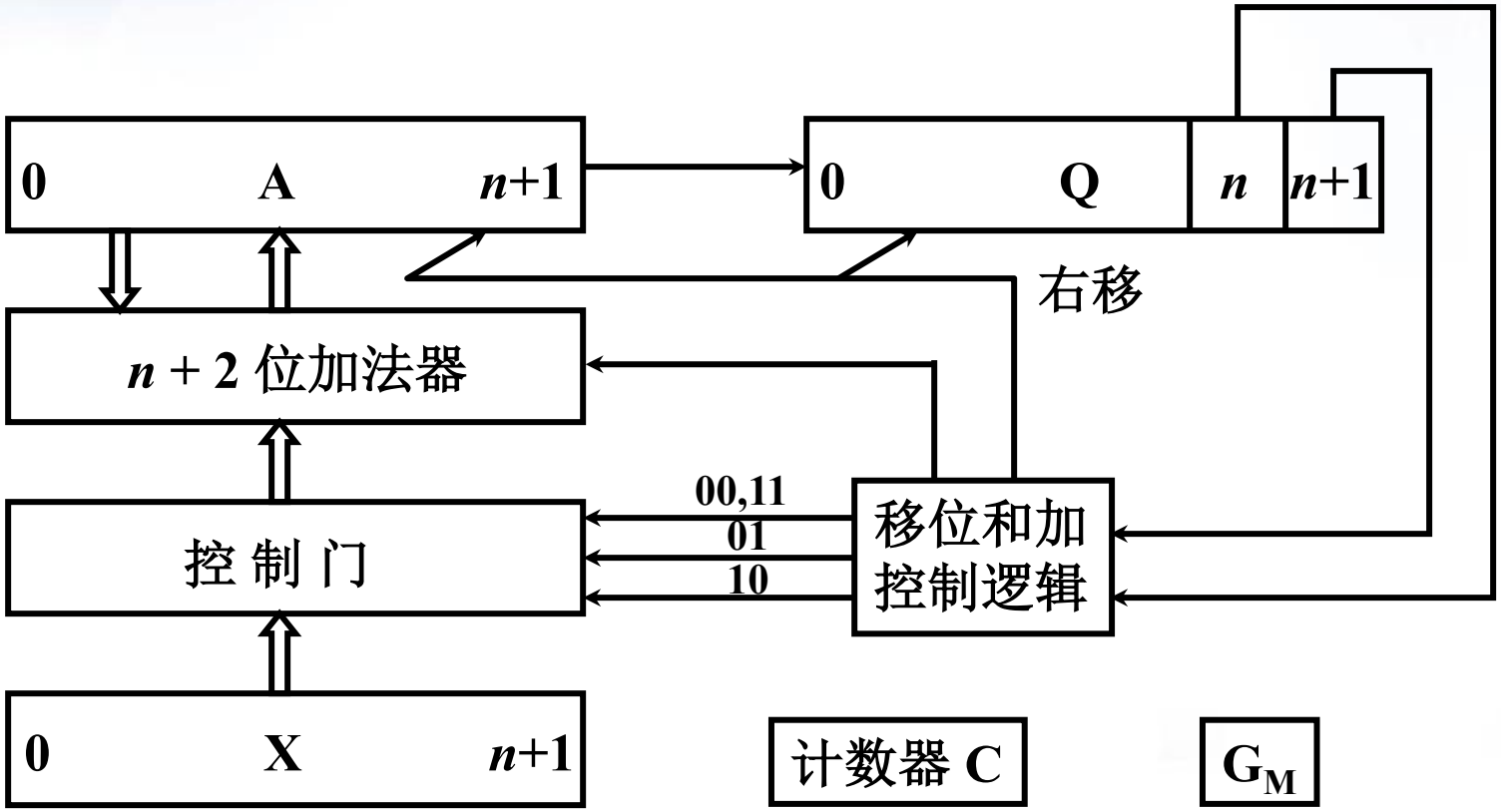
作答

$[x]_{补}$=0,10011，$[-x]_{补}$=1,01101，$[y]_{补}$=0,100011

部分积	乘数	Y_{n+1}	说明
00 , 000 00 +11, 011 01	0,100011	0	$Y_n y_{n+1}$=10，部分积加$[-x]_{补}$
11 , 011 01			右移1位
11 , 101 10 11 , 110 11 + 00, 100 11	1 010001 01 01000	1 1	$Y_n y_{n+1}$=11，部分积右移1位 $Y_n y_{n+1}$=01，部分积加$[x]_{补}$
00 , 011 10	01		右移1位
00 , 001 11 00 , 000 11 00 , 000 01 +11, 011 01	001 0100 1001 010 11001 01	0 0 0	$Y_n y_{n+1}$=00，部分积右移1位 $Y_n y_{n+1}$=00，部分积右移1位 $Y_n y_{n+1}$=10，部分积加$[-x]_{补}$
11,011 10	11001		右移1位
11,10111 + 00,100 11	011001 0	1	$Y_n y_{n+1}$=01，部分积加$[x]_{补}$
00,010 10	011001		

即 $[x \times y]_{补}$=0,01010011001，x·y= 0, 01010011001

(2) Booth 算法的硬件配置



A、X、Q 均 $n+2$ 位

移位和加受末两位乘数控制

乘法小结

- 整数乘法与小数乘法完全相同
  可用 逗号 代替小数点

- 原码乘　符号位 单独处理
  补码乘　符号位 自然形成

原码乘去掉符号位运算　即为无符号数乘法

- 不同的乘法运算需有不同的硬件支持

一、浮点加减运算

$$x = S_x \cdot 2^{j_x} \qquad y = S_y \cdot 2^{j_y}$$

1. 对阶

(1) 求阶差

$$\Delta j = j_x - j_y = \begin{cases} = 0 & j_x = j_y \quad \text{已对齐} \\ > 0 & j_x > j_y \begin{cases} x \text{ 向 } y \text{ 看齐} & S_x \leftarrow 1,\ j_x - 1 \\ y \text{ 向 } x \text{ 看齐} & S_y \rightarrow 1,\ j_y + 1 \end{cases} \\ < 0 & j_x < j_y \begin{cases} x \text{ 向 } y \text{ 看齐} & S_x \rightarrow 1,\ j_x + 1 \\ y \text{ 向 } x \text{ 看齐} & S_y \leftarrow 1,\ j_y - 1 \end{cases} \end{cases}$$

(2) 对阶原则

小阶向大阶看齐

例如 $x = 0.1101 \times 2^{01}$ $y = (-0.1010) \times 2^{11}$ 求 $x+y$

解： $[x]_{补} = 00, 01; 00.1101$ $[y]_{补} = 00, 11; 11.0110$

1. 对阶

① 求阶差 $[\Delta j]_{补} = [j_x]_{补} - [j_y]_{补}$

$$\begin{array}{rr} & 00, 01 \\ + & 11, 01 \\ \hline & 11, 10 \end{array}$$

阶差为负（-2） $\therefore S_x \longrightarrow 2$ $j_x + 2$

② 对阶 $[x]_{补'} = 00, 11; 00.0011$

2. 尾数求和

$$\begin{array}{rll} & [S_x]_{补'} & = 00.0011 \\ + & [S_y]_{补} & = 11.0110 \\ \hline & & 11.1001 \end{array}$$

对阶后的$[S_x]_{补'}$

$\therefore [x+y]_{补} = 00, 11; 11.1001$

3. 规格化

(1) 规格化数的定义

$$r = 2 \qquad 1/2 \le |S| < 1$$

(2) 规格化数的判断

$S>0$	规格化形式	$S<0$	规格化形式
真值	0.1×× ··×	真值	–0.1×× ··×
原码	0.1×× ··×	原码	1.1×× ··×
补码	0.1×× ··×	补码	1.0×× ··×
反码	0.1×× ··×	反码	1.0×× ··×

原码　不论正数、负数，第一数位为1

补码　符号位和第 一数位不同

特例

$$S = -\frac{1}{2} = -0.100\cdots0$$

$$[S]_{原} = 1.100\cdots0$$

$$[S]_{补} = 1.100\cdots0$$

$\therefore$ $[-\frac{1}{2}]_{补}$ 不是规格化的数

$$S = -1$$

$$[S]_{补} = 1.000\cdots0$$

$\therefore$ $[-1]_{补}$ 是规格化的数

(3) 左规

尾数左移一位，阶码减 1，直到数符和第一数位不同为止

上例　$[x+y]_{补} = 00,\ 11;\ 11.\ 1001$

左规后　$[x+y]_{补} = 00,\ 10;\ 11.\ 0010$

$$\therefore\ x + y = (-0.1110) \times 2^{10}$$

(4) 右规

当 尾数溢出（ >1）时， 需 右规

即尾数出现 01. ××　…×或 10. ××　…×时

尾数右移一位，阶码加 1

例6.27 $x = 0.1101 \times 2^{10}$ $y = 0.1011 \times 2^{01}$

求 $x+y$（除阶符、数符外，阶码取 3 位，尾数取 6 位）

解：$[x]_{补} = 00, 010; 00.110100$

$[y]_{补} = 00, 001; 00.101100$

① 对阶

$$[\Delta j]_{补} = [j_x]_{补} - [j_y]_{补} = 00, 010 + 11, 111 = 100, 001$$

阶差为 +1　　$\therefore S_y \longrightarrow 1,\ j_y+1$

$\therefore [y]_{补'} = 00, 010; 00.010110$

② 尾数求和

$$[S_x]_{补} = 00.110100$$
$$+\ [S_y]_{补'} = 00.010110$$
$$01.001010$$

对阶后的$[S_y]_{补'}$

尾数溢出需右规

③ 右规

$[x+y]_{补} = 00,\ 010;\ 01.\ 001010$

右规后

$[x+y]_{补} = 00,\ 011;\ 00.\ 100101$

$\therefore\ x+y = 0.\ 100101 \times 2^{11}$

4. 舍入

在 对阶 和 右规 过程中，可能出现 尾数末位丢失 引起误差，需考虑舍入

(1) 0 舍 1 入法

(2) 恒置 “1” 法

例 6.28 $x=(-\frac{5}{8})\times 2^{-5}$ $y=(\frac{7}{8})\times 2^{-4}$

求 $x-y$（除阶符、数符外，阶码取 3 位，尾数取 6 位）

解： $x=(-0.101000)\times 2^{-101}$ $y=(0.111000)\times 2^{-100}$

$[x]_{补}=11,011;\ 11.011000$ $[y]_{补}=11,100;\ 00.111000$

① 对阶

$$[\Delta j]_{补}=[j_x]_{补}-[j_y]_{补}$$

$$\begin{array}{r} 11,011 \\ +\ 00,100 \\ \hline 11,111 \end{array}$$

阶差为 –1 ∴ $S_x \longrightarrow 1$, j_x+1

∴ $[x]_{补'}=11,100;\ 11.101100$

② 尾数求和

$$[S_x]_{补'} = 11.101100$$
$$+\ [-S_y]_{补} = 11.001000$$
$$\overline{\qquad\qquad\qquad}$$
$$110.110100$$

③ 右规

$$[x-y]_{补} = 11,100;\ 10.110100$$

右规后

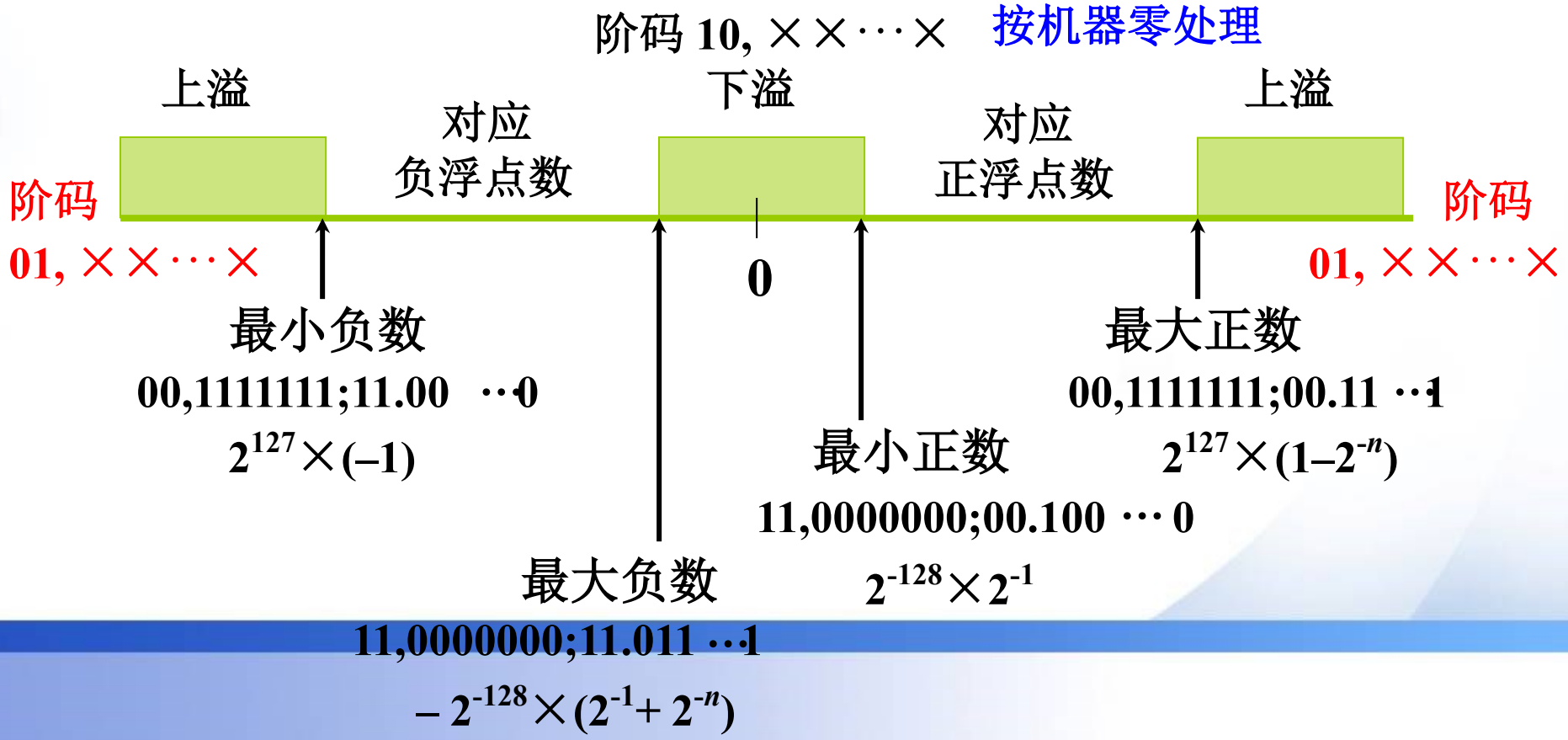
$$[x-y]_{补} = 11,101;\ 11.011010$$

$$\therefore\ x-y = (-0.100110)\times 2^{-11}$$

$$= (-\frac{19}{32})\times 2^{-3}$$

5. 溢出判断

设机器数为补码，尾数为 规格化形式，并假设阶符取 2 位，阶码的数值部分取 7 位，数符取 2 位，尾数取 n 位，则该 补码在数轴上的表示为



主观题 10分

设置

P292-6.26 按机器补码浮点运算步骤，计算[x±y]补.

（1）$x=2^{-011}\times 0.101\ 100$，

$y=2^{-010}\times(-0.011\ 100)$；

正常使用主观题需2.0以上版本雨课堂

作答

课后作业

雨课堂发布：第6章练习

提交截止时间：4月26日（周五）早8：00，

纸质作业周五上课提交