

计算机组成原理


翁睿

哈尔滨工业大学

二、总线通信控制

1. 目的 解决通信双方 协调配合 问题

2. 总线传输周期



申请分配阶段	主模块申请，总线仲裁决定
寻址阶段	主模块向从模块 给出地址 和 命令
传数阶段	主模块和从模块 交换数据
结束阶段	主模块 撤消有关信息

以输入数据为例：

寻址	T_1	主模块发地址
	T_2	主模块发命令
传数	T_3	从模块提供数据
结束	T_4	从模块撤销数据，主模块撤销命令

以输入数据为例：



3.5

寻址 { T_1 主模块发地址
 T_2 主模块发命令

传数 { T_{3-1} 从模块提供数据
 T_{3-2} 从模块提供数据
.....
 T_{3-n} 从模块提供数据

结束 T_4 从模块撤销数据，主模块撤销命令

赠送小题两道


1. 某同步总线的时钟频率为 100MHz ，宽度为 32 位，地址/数据线复用，每传输一个地址或数据占用一个时钟周期。若该总线支持突发（猝发）传输方式，则一次“主存写”总线事务传输 128 位数据所需要的时间至少是_____。

A. 20ns B. 40ns C. 50ns D. 80n

2. 假设某系统总线在一个总线周期中并行传输 4B 信息，一个总线周期占用 2 个时钟周期，总线时钟频率为 10MHz ，则总线带宽是_____。

A. 10MB/s B. 20MB/s C. 40MB/s D. 80MB/s

3. 总线通信的四种方式

- 
- 同步通信 由 统一时标 控制数据传送
 - 异步通信 采用 应答方式，没有公共时钟标准
 - 半同步通信 同步、异步结合、加入等待信号
 - 分离式通信 充分 挖掘 系统 总线每个瞬间 的 潜力

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

6.1 无符号数和有符号数

一、无符号数

寄存器的位数 n

反映无符号数的表示范围

所能表示的无符号数范围：

000...000 ~ 111...111

n 个0

n 个1

0 ~ ($2^n - 1$)

二、有符号数

6.1

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

小数点的位置

1 . 1011

小数点的位置

0 , 1100

小数点的位置

1 , 1100

小数点的位置

2. 原码表示法 带符号的绝对值表示 6.1

(1) 定义

整数

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

x 为真值 n 为整数的位数

小数

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

x 为真值

3. 补码表示法

6.1

(1) 补的概念

• 时钟

逆时针 $\frac{-3}{3}$

顺时针 $\frac{+9}{15}$

可见 -3 可用 $+9$ 代替 减法 \rightarrow 加法 $\frac{-12}{3}$

称 $+9$ 是 -3 以 12 为模的 补数

记作 $-3 \equiv +9 \pmod{12}$

时钟以
12为模

- 一个负数加上 模 数即得该负数的补数
- 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
- 正数的补数等于它本身

(3) 补码定义

6.1

整数

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

x 为真值

n 为整数的位数

小数

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

x 为真值

(4) 求补码的快捷方式

$$[x]_{\text{原}} \xrightarrow{?} [x]_{\text{补}}$$

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

$$[x]_{\text{补}} \xrightarrow{?} [x]_{\text{原}}$$

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

$$[x]_{\text{补}} \xrightarrow{?} [-x]_{\text{补}}$$

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

4. 反码表示法

6.1

(1) 定义

整数

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

x 为真值

n 为整数的位数

小数

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

x 为真值

n 为小数的位数

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

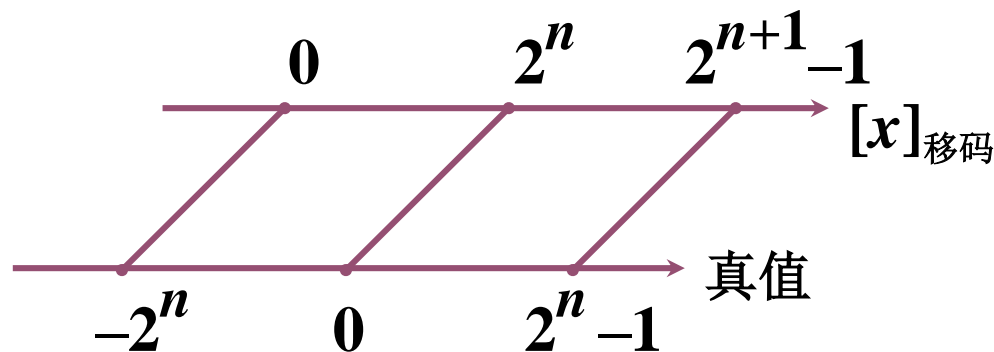
5. 移码表示法

6.1

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 $x = 10100$

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

$$x = -10100$$

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

用 逗号 将符号位
和数值部分隔开

(2) 移码和补码的比较

设 $x = +1100100$

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

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

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

6.1

真值 x ($n=5$)	$[x]_{\text{补}}$	$[x]_{\text{移}}$	$[x]_{\text{移}}$ 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

(4) 移码的特点

6.1

➤ 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

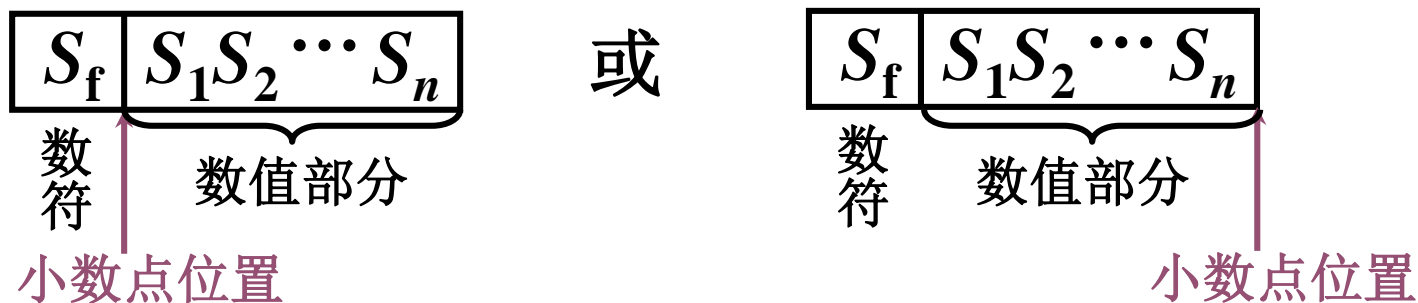
用移码表示浮点数的阶码

能方便地判断浮点数的阶码大小

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



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

6.2

$N = S \times r^j$ 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

✓ $= 0.110101 \times 2^{10}$ 规格化数

$$= 1.10101 \times 2^1$$

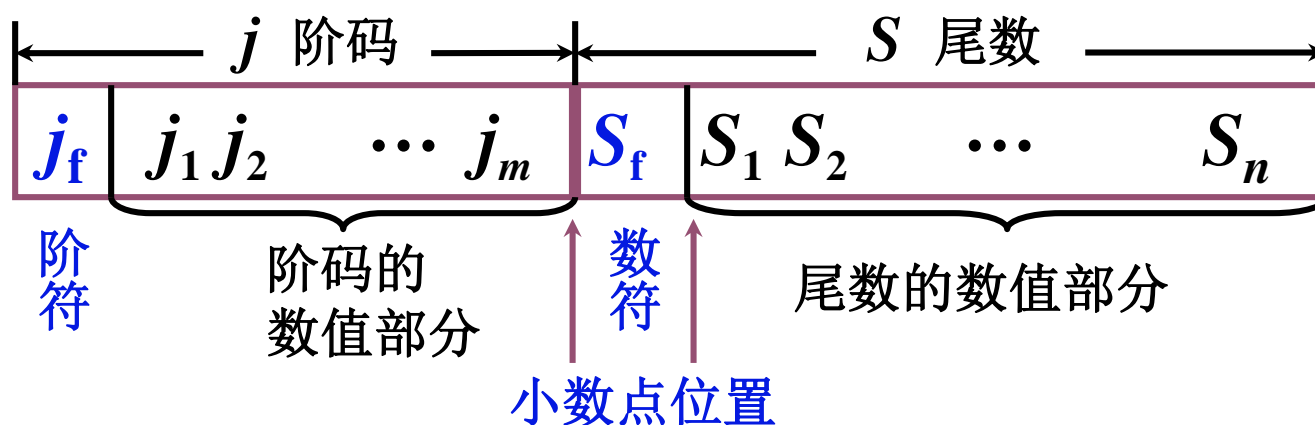
$$= 1101.01 \times 2^{-10}$$

$$✓ = 0.00110101 \times 2^{100}$$

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

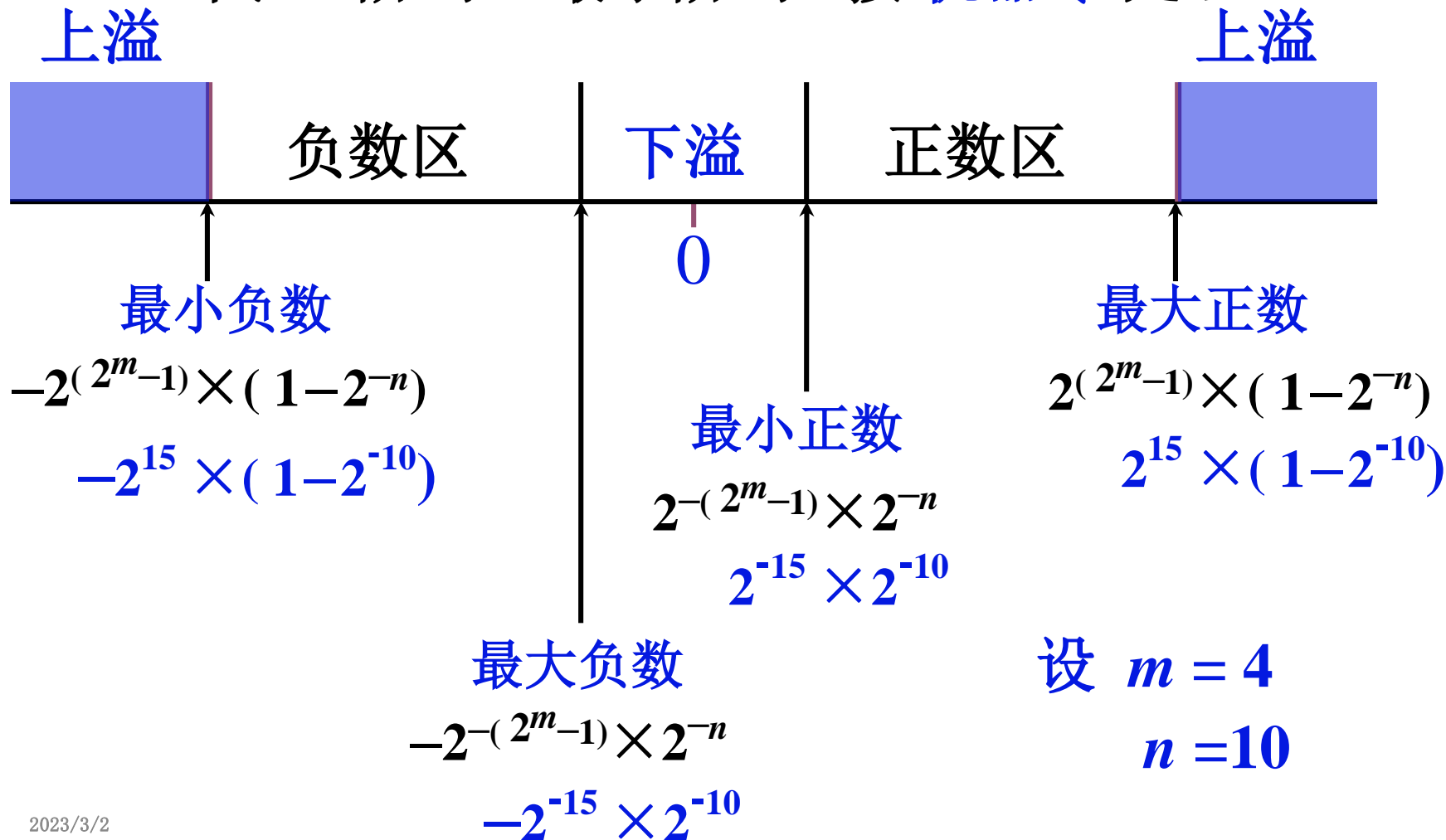
共同表示小数点的实际位置

2. 浮点数的表示范围 (以原码为例)

6.2

上溢 阶码 > 最大阶码

下溢 阶码 < 最小阶码 按 机器零 处理



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

\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

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

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

满足 最大精度 可取 $m = 4, n = 18$

3. 浮点数的规格化形式

$r = 2$ 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 $2^{+1111} \times \underbrace{0.1111111111}_{10 \text{ 个 } 1} = 2^{15} \times (1 - 2^{-10})$

最小正数 $2^{-1111} \times \underbrace{0.1000000000}_{9 \text{ 个 } 0} = 2^{-15} \times 2^{-1} = 2^{-16}$

最大负数 $2^{-1111} \times (-\underbrace{0.1000000000}_{9 \text{ 个 } 0}) = -2^{-15} \times 2^{-1} = -2^{-16}$

最小负数 $2^{+1111} \times (-\underbrace{0.1111111111}_{10 \text{ 个 } 1}) = -2^{15} \times (1 - 2^{-10})$

三、举例

6.2

例 4.13 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 $x = 0.0010011\textbf{000}$

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

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

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

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

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

例 4.14 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 $x = -0000111010$

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

定点机中

$$[x]_{\text{原}} = 1, 0000111010$$

$$[x]_{\text{补}} = 1, 1111000110$$

$$[x]_{\text{反}} = 1, 1111000101$$

浮点机中

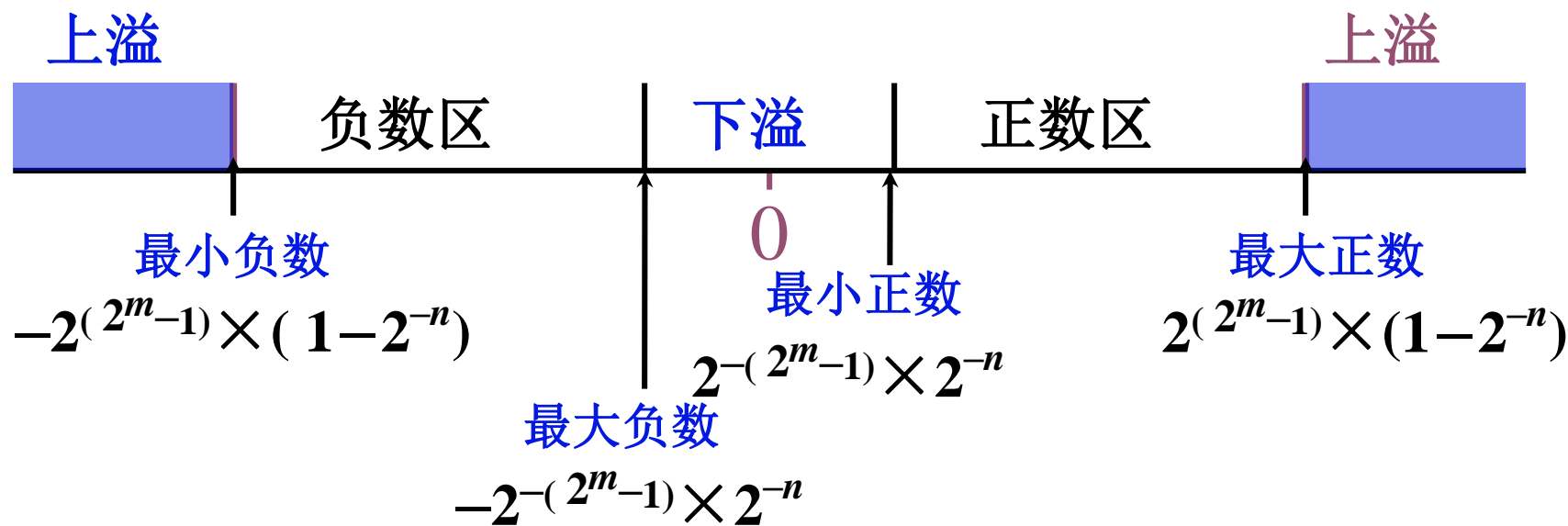
$$[x]_{\text{原}} = 0, 0110; 1. 1110100000$$

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

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

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

例4.15 写出对应下图所示的浮点数的补码 **6.2**
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

6.2

- 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

$\times, \times \times \times \times; \quad \mathbf{0.00 \dots 0}$

(阶码 = -16) $\mathbf{1, 0000; \quad \times.\times \times \dots \times}$

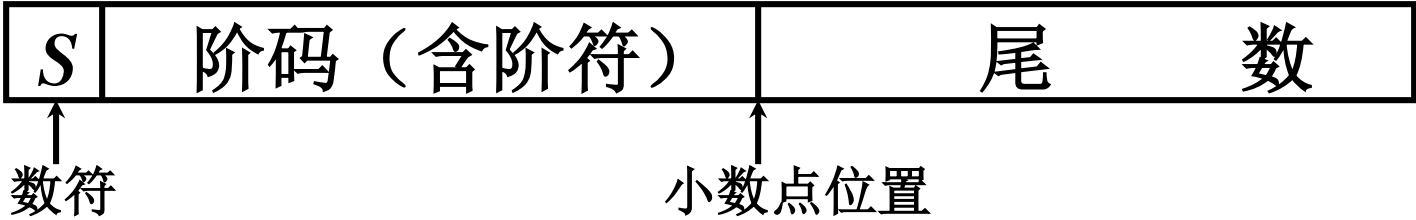
当阶码用移码，尾数用补码表示时，机器零为

$\mathbf{0, 0000; \quad 0.00 \dots 0}$

有利于机器中“判 0”电路的实现

四、IEEE 754 标准

6.2



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

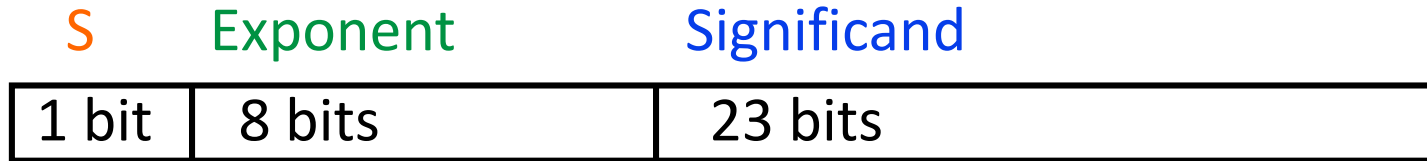


Prof. William Kahan

IEEE 754 Floating Point Standard

规格化数: $\pm 1.\text{xxxxxxxxxx}_{\text{two}} \times 2^{\text{Exponent}}$

Single Precision : (Double Precision is similar)



- Sign bit: 1 表示 negative ; 0 表示 positive
- Exponent (阶码) :
 - SP 规格化数阶码范围为 0000 0001 (-126) ~ 1111 1110 (127)
 - bias 为 127 (single), 1023 (double)
- 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)

NaN

Inf

全 0 和全 1 用来表示特殊值 !

若用 128, 则阶码的范围变为多少?

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

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

0000 0001 (-127) ~
1111 1110 (126)

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:**
 - $0111\ 1101_{\text{two}} = 125_{\text{ten}}$
 - Bias adjustment: $125 - 127 = -2$
- **Significand:**
$$1 + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 0 \times 2^{-4} + 0 \times 2^{-5} + \dots$$
$$= 1 + 2^{-1} + 2^{-2} = 1 + 0.5 + 0.25 = 1.75$$
- **Represents:** $-1.75_{\text{ten}} \times 2^{-2} = -0.4375$

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: $127 + 3 = 128 + 2 = 1000\ 0010_2$

11000 0010	100 1100 0000 0000 0000 0000
------------	------------------------------

The Hex rep. is C14C0000H

4.3 定点运算

一、移位运算

1. 移位的意义

$$15.\text{ m} = 1500.\text{ cm}$$

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4.16

4.3

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

移位操作	机 器 数	对应的真值
	$[A]_{\text{原}} = [A]_{\text{补}} = [A]_{\text{反}}$	
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4.17

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

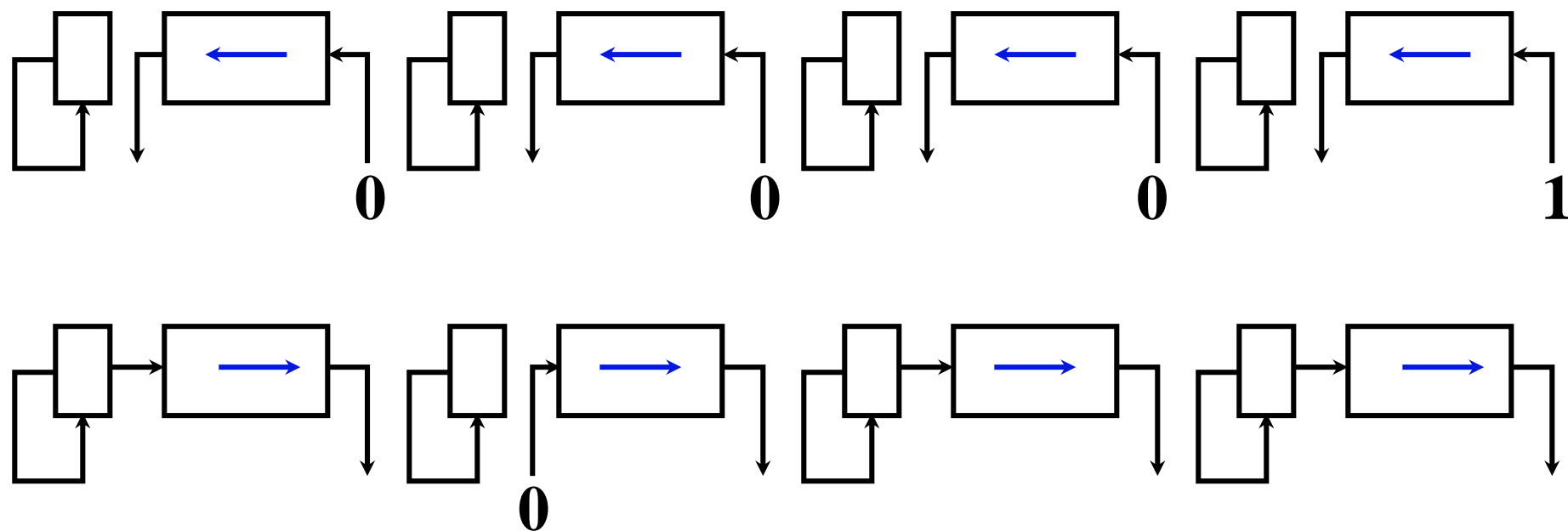
移位操作	机 器 数	对应的真值
移位前	1,1100110	– 26
左移一位	1,1001100	– 52
左移两位	1,0011000	– 104
右移一位	1,1110011	– 13
右移两位	1,1111001	– 7

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

4.3



(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

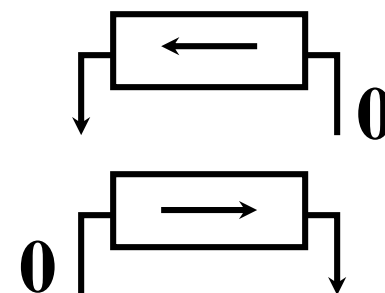
4.3

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

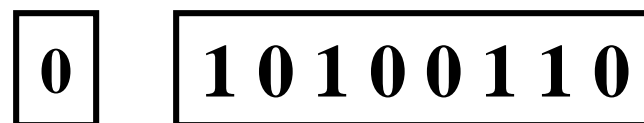
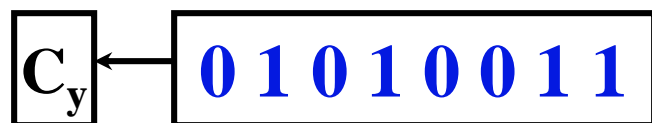
逻辑左移 **10100110**

逻辑右移 **01011001**

算术左移 **00100110**

算术右移 **11011001** (补码)

高位 1 移丢



二、加减法运算

4.3

1. 补码加减运算公式

(1) 加法

整数 $[A]_{\text{补}} + [B]_{\text{补}} = [A+B]_{\text{补}} \pmod{2^{n+1}}$

小数 $[A]_{\text{补}} + [B]_{\text{补}} = [A+B]_{\text{补}} \pmod{2}$

(2) 减法

$$A-B = A+(-B)$$

整数 $[A-B]_{\text{补}} = [A+(-B)]_{\text{补}} = [A]_{\text{补}} + [-B]_{\text{补}} \pmod{2^{n+1}}$

小数 $[A-B]_{\text{补}} = [A+(-B)]_{\text{补}} = [A]_{\text{补}} + [-B]_{\text{补}} \pmod{2}$

连同符号位一起相加，符号位产生的进位自然丢掉

2. 举例

4.3

例 6.18 设 $A = 0.1011$, $B = -0.0101$

求 $[A + B]_{\text{补}}$

验证

$$\begin{array}{rcl}
 \text{解: } [A]_{\text{补}} & = & 0.1011 \\
 + [B]_{\text{补}} & = & 1.1011 \\
 \hline
 [A]_{\text{补}} + [B]_{\text{补}} & = & 10.0110 = [A + B]_{\text{补}} \\
 \therefore A + B & = & 0.0110
 \end{array}
 \qquad
 \begin{array}{r}
 0.1011 \\
 - 0.0101 \\
 \hline
 0.0110
 \end{array}$$

例 6.19 设 $A = -9$, $B = -5$

求 $[A+B]_{\text{补}}$

验证

$$\begin{array}{rcl}
 \text{解: } [A]_{\text{补}} & = & 1, 0111 \\
 + [B]_{\text{补}} & = & 1, 1011 \\
 \hline
 [A]_{\text{补}} + [B]_{\text{补}} & = & 11, 0010 = [A + B]_{\text{补}} \\
 \therefore A + B & = & -1110
 \end{array}
 \qquad
 \begin{array}{r}
 -1001 \\
 + -0101 \\
 \hline
 -1110
 \end{array}$$

例 6.20 设机器数字长为 8 位（含 1 位符号位） **4.3**

且 $A = 15$, $B = 24$, 用补码求 $A - B$

解: $A = 15 = 0001111$

$B = 24 = 0011000$

$[A]_{\text{补}} = 0, 0001111$ $[B]_{\text{补}} = 0, 0011000$

$+ [-B]_{\text{补}} = 1, 1101000$

$[A]_{\text{补}} + [-B]_{\text{补}} = 1, 1110111 = [A - B]_{\text{补}}$

$\therefore A - B = -1001 = -9$

练习 1 设 $x = \frac{9}{16}$ $y = \frac{11}{16}$, 用补码求 $x+y$

$x + y = -0.1100 = -\frac{12}{16}$ **错**

练习 2 设机器数字长为 8 位（含 1 位符号位）

且 $A = -97$, $B = +41$, 用补码求 $A - B$

$A - B = +1110110 = +118$ **错**

3. 溢出判断

4.3

(1) 一位符号位判溢出

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，其结果的符号与原操作数的符号不同，即为溢出**

硬件实现

最高有效位的进位 \oplus 符号位的进位 = 1 溢出

如

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

4.3

$$[x]_{\text{補}} = \begin{cases} x & \mathbf{1} > x \geq \mathbf{0} \\ \mathbf{4} + x & \mathbf{0} > x \geq \mathbf{-1} \pmod{4} \end{cases}$$

$$[x]_{\text{补}} + [y]_{\text{补}} = [x + y]_{\text{补}} \pmod{4}$$

$$[x - y]_{\text{补}} = [x]_{\text{补}} + [-y]_{\text{补}} \quad (\text{mod } 4)$$

结果的符号位	双符号位	是否溢出	溢出类型
00	××××××	未溢出	
11	××××××	未溢出	

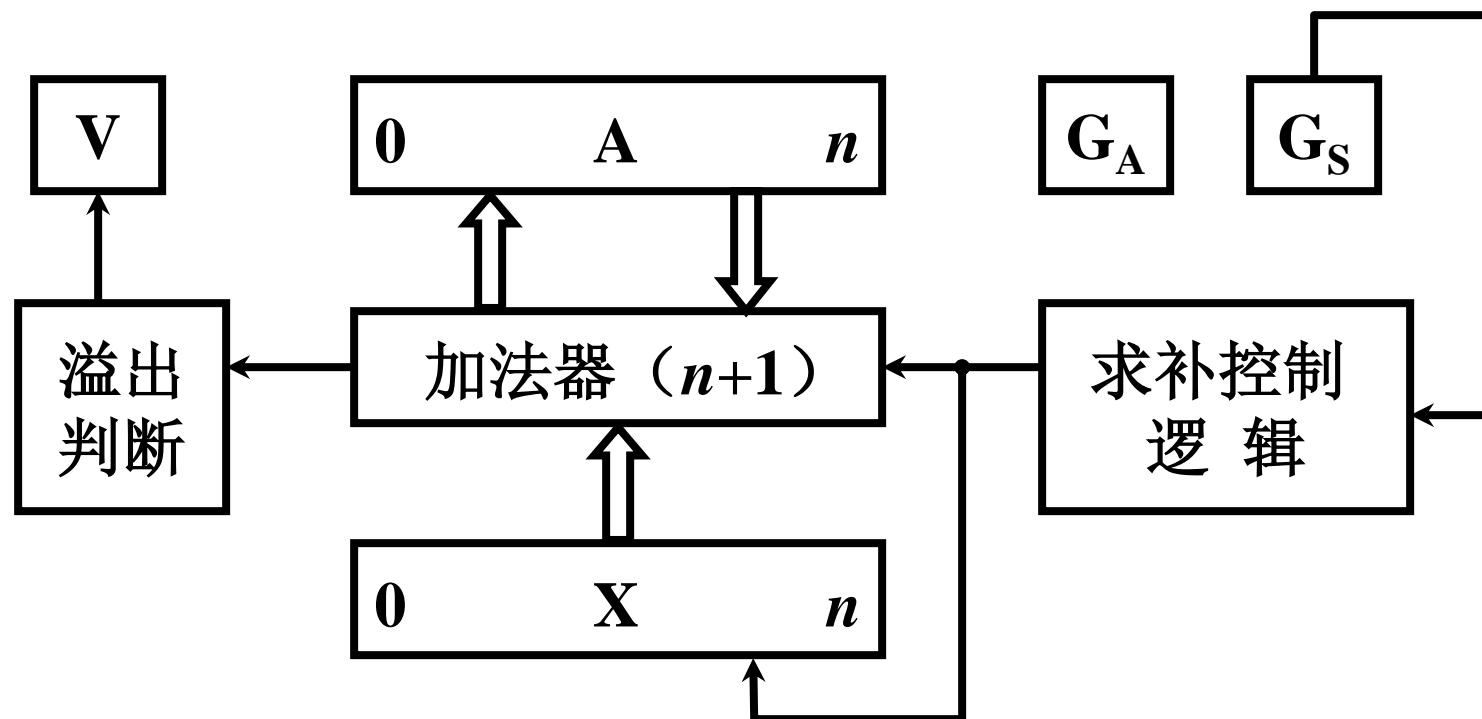
结果的双符号位 **不同** **溢出**

10, ×××××

01, ×××××

最高符号位 代表其 真正的符号

4. 补码加减法的硬件配置



A、X 均 $n+1$ 位

用减法标记 G_S 控制求补逻辑

第6章 课后作业 ★第一弹

作业：T6.4, T6.9, T6.19 (1) (3)

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。

$$-\frac{13}{64}, \frac{29}{128}, 100, -87$$

6.9 当十六进制数 9BH 和 FFH 分别表示为原码、补码、反码、移码和无符号数时,所对应的十进制数各为多少(设机器数采用 1 位符号位)?

6.19 设机器数字长为 8 位(含 1 位符号位),用补码运算规则计算下列各题。

(1) $A = \frac{9}{64}, B = -\frac{13}{32}$, 求 $A+B$ 。

(3) $A = -\frac{3}{16}, B = \frac{9}{32}$, 求 $A+B$ 。

友情提醒：下次上课时交前3次作业 ☺