第2章 数据表示、运算与校验

主要介绍：

①数字型数据的计数制、符号数的表示、定点数和浮点数；

②基本的运算方法；

③字符的表示；

④常用的数据校验方法；

2.1 数值型数据的表示方法

2.1.1 进位计数制

※ 数制的基与权

- 在任一数制中，每一个数位上允许使用的记数符号的个数被称为该数制的**基数**。
- 每1位都对应1个表示该位在数码中的位置的值，这个值就称为数位的**权值**w。

［例］ $\underline{1}28_{10}$， $\underline{1}101_{2}$

$W=10^2$ $W=2^3$

1. 常用的几种进位制

（1）2进制：0、1

（2）8进制：0、1、2、…、7

（3）16进制：0、…、9、A、B、C、D、E、F

2. 进制之间的转换

（1）整数 10 → 2 (除2取余法)

（2）小数 10 → 2 (乘2取整法)

（3）整数 2 → 10(按权相加)

（4）小数 2 → 10(按权相加)

（5）16进制↔2进制(逐位转换/分组转换)

(1) $29_{10} \to X_2$ 除2取余

$2\,|\,29$

$2\,|\,14 \longrightarrow 1$ 低位

$2\,|\,7 \longrightarrow 0$

$2\,|\,3 \longrightarrow 1$

$2\,|\,1 \longrightarrow 1$

$0 \longrightarrow 1$ 高位

$X_2=11101_2$

(2) $0.6875_{10} \to X_2$ 乘2取整

$0.6875 \times 2 = 1.375 \longrightarrow 1$ 高位

$0.375 \times 2 = 0.75 \longrightarrow 0$

$0.75 \times 2 = 1.5 \longrightarrow 1$

$0.5 \times 2 = 1.0 \longrightarrow 1$ 低位

$X_2=0.1011_2$

(3) $1101.11_2 \to X_{10}$ 按权相加

$$X_{10}=1\times 2^3+1\times 2^2+0\times 2^1+1\times 2^0+1\times 2^{-1}+1\times 2^{-2}$$

$$=13.75_{10}$$

(4) $111101_2 \rightarrow X_{16}$　4位分组、按组转换

$X_{16}=111101=0011\ 1101=\underline{0011}\ \underline{1101}=3D_{16}$

(5) $28AF_{16} \rightarrow X_2$　逐位转换

$X_2=28AF$

$=\underline{0010}\ \underline{1000}\ \underline{1010}\ \underline{1111}$

(6) $28AF_{16} \rightarrow X_{10}$　按权相加

$X_{10}=28AF=2\times16^3+8\times16^2+10\times16^1+15\times16^0$

$=10415_{10}$

2.1.2 带符号数的表示

数的符号表示规则：

"0"表示正号"+"，"1"表示负号"–"

二进制数的码制：原码、反码、补码和移码

1、原码

一个二进制数，用0-1代码表示符号，数值位不变就得到与该二进制数真值对应的原码

真值	+1001010	-1001010
	↓↓↓↓↓↓↓↓	↓↓↓↓↓↓↓↓
原码	<u>0</u>1001010	<u>1</u>1001010

字长为8位的原码

表示范围为：-127～+127

$[+127]_{原}$ =0 1111111

$[-127]_{原}$ =1 1111111

数值“0”有两种原码形式：

$[+0]_{原}$=0 0000000

$[-0]_{原}$=1 0000000

2、反码

① 正数情况

$X_{反}=X_{原}$（$X\geq 0$）

[例] X= +1101001 (真值+105)

$X_{反}=X_{原}=\underline{0}\ 1101001$

② 负数情况

符号位保持为“1”，数值位分别“按位取反”

[例] X= -1101001 (真值-105)

$X_{原}=\underline{1}\ 1101001$

$X_{反}=\underline{1}\ 0010110$

字长8位的反码

表示范围为：-127 ～ +127

$[+127]_{反}=0\ 1111111$

$[-127]_{反}=1\ 0000000$

数值“0”也有两反码形式:

$[+0]_{反}=0\ 0000000$

$[-0]_{反}=1\ 1111111$

3、补码

编码定义：$[X]_{补}=X+2^n$ (模2^n)，n为编码位数

※补码的编码规则：

(a)对于正数（字长8位）

$[X]_{补}=[X]_{原}$ （即$X\geq 0$时）

(b)对于负数（字长8位）

符号位仍保持为“1”

其余各数值位“按位取反，末位再加1”

$[X]_{补}=[X]_{反}+\ldots\underline{1}$ （即$X<0$时）

字长8位的补码

表示范围为：-128 ～ +127

$[+127]_{补}=0\ 1111111$；

$[-128]_{补}=1\ 0000000$

※注意

补码比原码和反码多表示1个负值，即−128

数值“0”只有1种补码形式：

$[+0]_{补}=[-0]_{补}=0\ 0000000$

二进制代码	无符号数值	原码值	反码值	补码值
0000 0000	0	+0	+0	+0
0000 0001	1	+1	+1	+1
⋮	⋮	⋮	⋮	⋮
0111 1110	126	+126	+126	+126
0111 1111	127	+127	+127	+127
1000 0000	128	-0	-127	-128
1000 0001	129	-1	-126	-127
1000 0010	130	-2	-125	-126
⋮	⋮	⋮	⋮	⋮
1111 1101	253	-125	-2	-3
1111 1110	254	-126	-1	-2
1111 1111	255	-127	-0	-1

4、原码和补码之间的转换

(1) 已知$[X]_{原}$，求$[X]_{补}$

[例] 已知$[X]_{原}$= 10011010，求$[X]_{补}$

解：$[X]_{原}$= 10011010　原码为负数

↓↓↓↓↓↓↓↓　符号位不变，其余各位变反

11100101

\+　　　　　　1　末位加1

$[X]_{补}$=11100110

(2) 已知$[X]_{补}$，求$[X]_{原}$

$$[[X]_{补}]_{补}=[X]_{原}$$

[例] 已知$[X]_{补}$=11101100，求$[X]_{原}$

解：$[X]_{补}$= 11101100　　补码为负数

↓↓↓↓↓↓↓↓　　符号位不变，其余各位变反

10010011

\+ 　　　　　1　　末位加1

$[X]_{原}$ =10010100

(3) 求补 (变补)，即已知$[X]_补$，求$[-X]_补$

$[X]_补$的代码连同符号位一起变反，末位再加1，即得到$[-X]_补$

[例] 已知$[X]_补$=01010110，求$[-X]_补$

解：$[X]_补$=01010110　　不区分正负数

↓↓↓↓↓↓↓↓　　连同符号位一起变反

10101001

\+　　　　1　　末位加1

$[-X]_补$=10101010

5、移(增)码

移码通常用于表示浮点数的阶码。

阶码一般为整数，故移码通常只用于表示整数

对定点整数x，它的移码是：

$$[x]_{移} = 2^{n-1} + x，其中 -2^{n-1} < x < 2^{n-1}$$

这里的n为$X_{原}$位数

上述规则等价于将x正向平移或者增加2^{n-1}，因此称之为移码或增码。

[例] 阶码的为6位，X表示其真值

$X_{移}=2^5+X \quad (-2^5<X<2^5)$

[例] 当正数X＝+10101时, $X_{移}=2^5+X=110101$

当负数X＝-10101时, $X_{移}=2^5+X$

$=2^5-10101$

$=001011$

移码表示范围与补码一致，0也只有1个移码。

正数：将原码符号位变反，即得到移码。

负数：将原码连同符号位一起变反，末位再加1，即得到移码(与变补等效)。

补码和移码：符号相反、数值位相同

2.1.3 定点数与浮点数

1、定点数的表示

数的小数点固定在同一位置不变。

①带符号的定点小数

约定所有数的小数点的位置，固定在符号位之后。

1	.　0 1 0 1 1 1 0

符号位 小数点　数值部分

字长 n+1位，则表示范围为：

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

②带符号的定点整数

小数点的位置固定在最低数值位之后

1	0 1 0 1 1 1 0 .

符号位　　数值部分　小数点

字长n+1位，则数的表示范围为：

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

③无符号定点整数

小数点的位置固定在最低数值位之后

若代码序列为$X_nX_{n-1}\ldots X_1X_0$，共n+1位，则有：

典型值	真值	代码序列
最大正数	$2^{n+1}-1$	11…1111
最小非零正数	1	00….001

表示范围为：

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

分辨率为：1

※字长8位的定点数的表示范围

无符号数：　$00000000 \sim 11111111$

0　　255

定点整数：　$11111111_{原} \sim 01111111_{原}$

-127　　127

$10000000_{补} \sim 01111111_{补}$

-128　　127

定点小数：　$1.1111111_{原} \sim 0.1111111_{原}$

$-(1-2^{-7})$　　$(1-2^{-7})$

$1.0000000_{补} \sim 0.1111111_{补}$

-1　　$(1-2^{-7})$

2、浮点数的表示原理

①浮点表示中，小数点的位置可按需浮动

格式模型：

阶符	阶码位 ．	数符	． 尾数位

隐含小数点　　隐含小数点

阶码（带符号的定点整数）

尾数（带符号的定点小数）

②引入浮点数表示的意义

[例]某字长为8位的原码二进制数

定点数：11111111 ~ 01111111

整数： -127　　127　　精度：1

小数： $-(1-2^{-7})$　　$1-2^{-7}$　　精度：2^{-7}

浮点数：5位阶码+3位尾数

01111111 ~ 01111011

$2^{15}\times(-0.75) \sim 2^{15}\times 0.75$

精度：11111001=$2^{-15}\times 0.25$

相同字长时，浮点数的表示范围更大、精度更高！

③浮点数的机器(存储)格式

浮点数真值： $N = \pm R^{E} \times M$

阶符				数符			
E_f	E_1	…	E_m	M_f	M_1	…	M_n
阶码E				尾数M			

R： 阶码的底数，隐含约定为2。

E： 阶码，定点整数，补码或移码表示，其**位数**决定了**数值的范围**；

M： 尾数，为定点小数，原码或补码表示，其**位数**决定着**数的精度**；**数符表示数的正负。**

④尾数M的规格化表示

规格化的目的 → 使浮点数的表示代码“唯一”

128=<u>0.128</u>×10^3=<u>1.28</u>×10^2=<u>12.8</u>×10^1， …

科学计数法约定：1≤|M|<10

则规范形式：128=1.28×10^2（唯一）

3.2=1.6×2^1=0.8×2^2=0.4×2^3, …

可以表示成任意多个代码形式 → 计算不便

约定尾数M的值域，使数的表示是唯一的、确定的

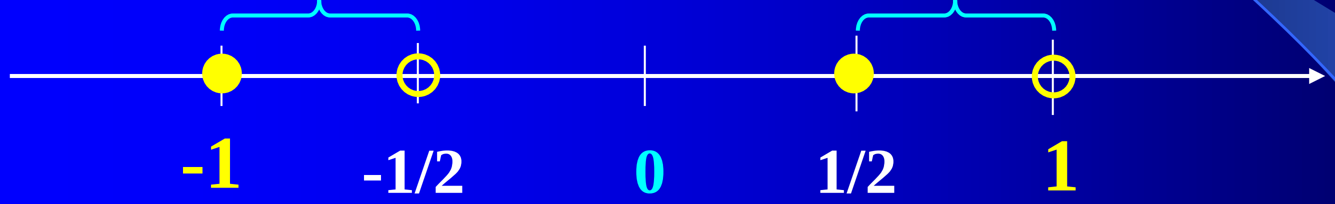
→尾数M的“规格”化

① 浮点数用原码表示时

$1/2 \le |M| < 1$

② 浮点数用补码表示时

$-1 \le M < -1/2$ 或 $1/2 \le M < 1$



对于原码： 规格化以后尾数的最高有效位为“1”

$M_{原}=0.\underline{1}000$，$1.\underline{1}010$ ~~$M_{原}=0.\underline{0}010$, $1.\underline{0}110$~~ 非规格化

对于补码：

正数，规格化后最高数值位为“1” 如$0.\underline{1}010$，$0.\underline{1}110$

0.625 0.875

负数，规格化后最高数值位为“0” 如$1.\underline{0}010$，$1.\underline{0}000$

-0.875 -1

[例1] 阶符1位、阶码m位，数符1位、尾数n位，如果表示成规格化补码，分析各真值对应的M、E取值情况。

$$N = M \times 2^{E} = F(M, E) \begin{cases} -1 \le M < -1/2 \text{或} 1/2 \le M < 1 \\ -2^{m} \le E \le 2^{m}-1 \end{cases}$$

最小浮点数	E→	为最大正数：$2^{m}-1$
	M→	为最小负数：-1
最大浮点数	E→	为最大正数：$2^{m}-1$
	M→	为最大正数：$1-2^{-n}$
最小浮点正数	E→	为最小负数：-2^{m}
	M→	为最小正数：2^{-1}
最大浮点负数	E→	为最小负数：-2^{m}
	M→	为最大负数：$-(1/2+2^{-n})$

[例2]某个用规格化补码表示的浮点数，其中阶码6位（含1位阶符），尾数10位（含1位数符）。

表示范围：011111 100…00 ~ 011111 011…11

$2^{31} \times (-1)$ $2^{31} \times (1-2^{-9})$

绝对精度（即最小的非0正数）：

100000 010…00 $=2^{-32} \times 0.5=2^{-33}$

相对精度（只与尾数的数值位数有关）：

xxxxxx 000…01 $=2^{-9}$

[思考]某十六进制浮点数$A3680000_{16}$，将其表示成补码，字长32位，阶码8位（含1位阶符），尾数24位（含1位数符），求该浮点数十进制的真值。

3、※IEEE754格式的浮点数

有32位浮点数（单精度）和64位浮点数（双精度）

① 32位短浮点数：

31	30 ~ 23	22 ~ 0
S	E	M

在上述的表示格式中：

- S =浮点数的符号位，0表示正数，1表示负数；
- E =阶码，8位，采用移码表示，阶符隐含；
- M=尾数，23位，纯小数表示，且真值=1+M；
- 阶码E采用移码形式，但只偏移2^7-1（不是2^7）。

② 64位长浮点数：

63	62 ~ 52	51 ~ 0
S	E	M

在上述表示格式中：

- S =浮点数的符号位，0表示正数，1表示负数；
- E =阶码，11位，采用移码方式，阶符隐含；
- M=尾数，52位，用纯小数表示；
- 对阶码E编码时，只偏移2^{10}-1（标准移码偏移2^{10}）。

※补充说明：

（1）为了确保浮点数表示的唯一性，约定 $0 \le M < 1$；

（2）E为全0且M非全0：非规范浮点数（E偏移126），则：$F_{真}=(-1)^S \times M \times 2^{E-126}$

（3）E为全0且M为全0：表示浮点数 0

（4）$1 \le E \le 254$：数是规范浮点数（E偏移127），则：$F_{真}=(-1)^S \times (1+M) \times 2^{E-127}$

（5）E为全1（255）：M为全0，则$F_{真}=\pm\infty$

（6）E为全1（255）：M非全0时，代码无效(NaN)；

［例］将十进制数20.59375转换成IEEE754的32位标准浮点数的2进制格式，并写出相应的16进制数。

［解答］首先分别将整数和小数部分转换成二进制：

20.59375＝10100.10011

然后移动小数点，使其在第1、2位之间

$10100.10011=1.\underline{010010011}\times 2^4$

小数点被左移了4位，于是得到：$e=4$

尾符S=0，阶码$E=4+\underline{127}=131$，尾数M=010010011

最后得到32位浮点数的二进制代码：

0100 0001 1010 0100 1100 0000 0000 0000

＝$(41A4C000)_{16}$