数字逻辑

信息编码

2019年3月18日

数的机器码表示

一、进位计数制及其转换

1、进位计数制

进位计数制：用少量的数字符号，按先后次序把它们排成数位，由低到高进行计数，计满进位，这样的方法称为进位计数制。

基数：进位制基本特征数，即所用到的数字符号个数。

例如：10进制 ：0～9 十个数码表示，基数为10。

权：进位制中各位“1”所表示的值为该位的权。

常见的进位制： 2，8，10，16进制。

进制表示

$$N = \sum_{i=-k}^{m} D_i * r^i$$

- N　代表一个数值
- r　是这个数制的基(Radix)
- i　表示这些符号排列的位号
- D_i 是位号为i的位上的一个符号
- r^i 是位号为i的位上的 1 代表的值
- $D_i * r^i$ 是第i位的所代表的实际值
- Σ　表示m+k+1位的值求累加和

1、十进制（Decimal）

基数：10

符号：0, 1, 2, 3, 4, 5, 6, 7, 8, 9

计算规律：“逢十进一 ”或“借一当十”

十进制数的多项式表示：

$$N_{10}=d_{n-1}\times10^{n-1}+d_{n-2}\times10^{n-2}+\cdots\cdots d_1\times10^{1}+d_0\times10^{0}+d_{-1}\times10^{-1}+d_{-2}\times10^{-2}+\cdots\cdots d_{-m}\times10^{-M}$$

式中：

m，n——正整数。n为整数位数，m为小数位数。

Di——第i位的系数。10^i称为该位的权。

例如：一个十进制数123.45的表示：

$$123.45=1\times10^{2}+2\times10^{1}+3\times10^{0}+4\times10^{-1}+5\times10^{-2}$$

注：等式左边为并列表示法，等式右边为多项式表示法。

2、二进制（Binary）

基数：2

符号：0,1

计算规律：“逢二进一”或“借一当二”

二进制的多项式表示：

$$N_2=d_{n-1}\times2^{n-1}+d_{n-2}\times2^{n-2}+\cdots\cdots d_1\times2^1+d_0\times2^0+d_{-1}\times2^{-1}+d_{-2}\times2^{-2}+\cdots\cdots d_{-m}\times2^{-m}$$

式中：

n为整数位数，m为小数位数。

di表示第i位的系数，2^i称为该位的权。

3、十六进制（Hexadecimal）

基数：16

符号：0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

计算规律：“逢十六进一”或“借一当十六”

十六进制的多项式表示：

$$N_{16}=d_{n-1}\times16^{n-1}+d_{n-2}\times16^{n-2}+\cdots\cdots d_1\times16^{1}+d_0\times16^{0}+d_{-1}\times16^{-1}+d_{-2}\times16^{-2}+\cdots\cdots d_{-m}\times16^{-m}$$

式中：

n为整数位数；m为小数位数。

Di表示第i位的系数，16^i称为该位的权。

例如：十六进制数 $(2C7.1F)_{16}$的表示：

$$(2C7.1F)_{16}=2\times16^{2}+12\times16^{1}+7\times16^{0}+1\times16^{-1}+15\times16^{-2}$$

4 、进位计数制之间的转换

1）R进制转换成十进制的方法

按权展开法：先写成多项式,然后计算十进制结果.

$$N=d_{n-1}d_{n-2}\cdots\cdots d_1d_0d_{-1}d_{-2}\cdots\cdots d_{-m}$$

$$=d_{n-1}\times R^{n-1}+d_{n-2}\times R^{n-2}+\cdots\cdots d_1\times R^1+d_0\times R^0+d_{-1}\times R^{-1}+d_{-2}\times R^{-2}$$

$$+\cdots\cdots d_{-m}\times R^{-m}$$

例：写出$(1101.01)_2$，$(237)_8$，$(10D)_{16}$的十进制数

$(1101.01)_2=1\times2^3+1\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}=8+4+1+0.25=13.25$

$(237)_8=2\times8^2+3\times8^1+7\times8^0=128+24+7=159$

$(10D)_{16}=1\times16^2+13\times16^0=256+13=269$

2）十进制转换成二进制方法

一般分为两个方法：

方法1——整数部分的转换

除2取余法（基数除法）

小数部分的转换

乘2取整法（基数乘法）

方法2——减权定位法

除基取余法：

把给定的数除以基数，取余数作为最低位的系数，然后继续将商部分除以基数，余数作为次低位系数，重复操作直至商为0。

例如：用基数除法将(327)10转换成二进制数。

		余数
2	327	
2	163	1
2	81	1
2	40	1
2	20	0
2	10	0
2	5	0
2	2	1
2	1	0
2	0	1

$(327)_{10} = (101000111)_{2}$

乘基取整法(小数部分的转换)

把给定的十进制小数乘以 2 ,取其整数作为二进制小数的第一位,然后取小数部分继续乘以2,将所的整数部分作为第二位小数,重复操作直至得到所需要的二进制小数

例如：将$(0.8125)_{10}$转换成二进制小数

整数部分	0.
2 ×0.8125=1.625	1
2 ×0.625=1.25	1
2 × 0.25=0.5	0
2 ×0.5=1	1

$(0.8125)_{10}=(0.1101)_{2}$

例：将$(0.2)_{10}$转换成二进制小数

整数部分	0.
0.2 × 2 = 0.4	0
0.4 × 2 = 0.8	0
0.8 × 2 = 1.6	1
0.6 × 2 = 1.2	1
0.2 × 2 = 0.4	0
0.4 × 2 = 0.8	0
0.8 × 2 = 1.6	1
0.6 × 2 = 1.2	1

$(0.2)_{10}=[\ 0.00110011001\cdots\ ]_{2}$

减权定位法

将十进制数依次从二进制的最高位权值进行比较，若够减则对应位置1，减去该权值后再往下比较，若不够减则对应位为0，重复操作直至差数为0。

512　256　128　64　32　16　8　4　2　1

例如：将 $(327)_{10}$ 转换成二进制数　256<327<512

327 - 256=71	1	256
71 < 128	0	128
71 - 64 =7	1	64
7 < 32	0	32
7 < 16	0	16
7 < 8	0	8
7 - 4 =3	1	4
3 - 2 =1	1	2
1 - 1 =0	1	1

3）其它进制之间的直接转换法

- 二进制(B)转换成八进制(Q)

例：$(10110111.01101)_2$

二进制：10，110，111．011，01

二进制：010，110，111．011，010

八进制：2　6　7　.　3　2

$(10110111.01101)_2=(267.32)_8$

- 八进制(Q)转换二进制(B)

例：$(123.46)_8$

$=(001,010,011.100,110)_2$

$=(1010011.10011)_2$

- 二进制(B)转换成十六进制(H)

例：$(110110111.01101)_2$

二进制：　1，1011，0111．0110，1

二进制：0001，1011，0111．0110，1000

十六进制：　1　B　7　．　6　8

$(10110111.01101)_2=(1B7.68)_{16}$

10110111.01101B =1B7.68H

- 十六进制(H)转换成二进制(B)

例：$(7AC.DE)_{16}$

$=(0111,1010,1100.1101,1110)_2$

$=(11110101100.1101111)_2$

几个简化运算的例子

133=128+5=10000101

65538=65536+2=1 0000 0000 0000 0010

2003=2047-44=1111 1111 1111-32-8-4

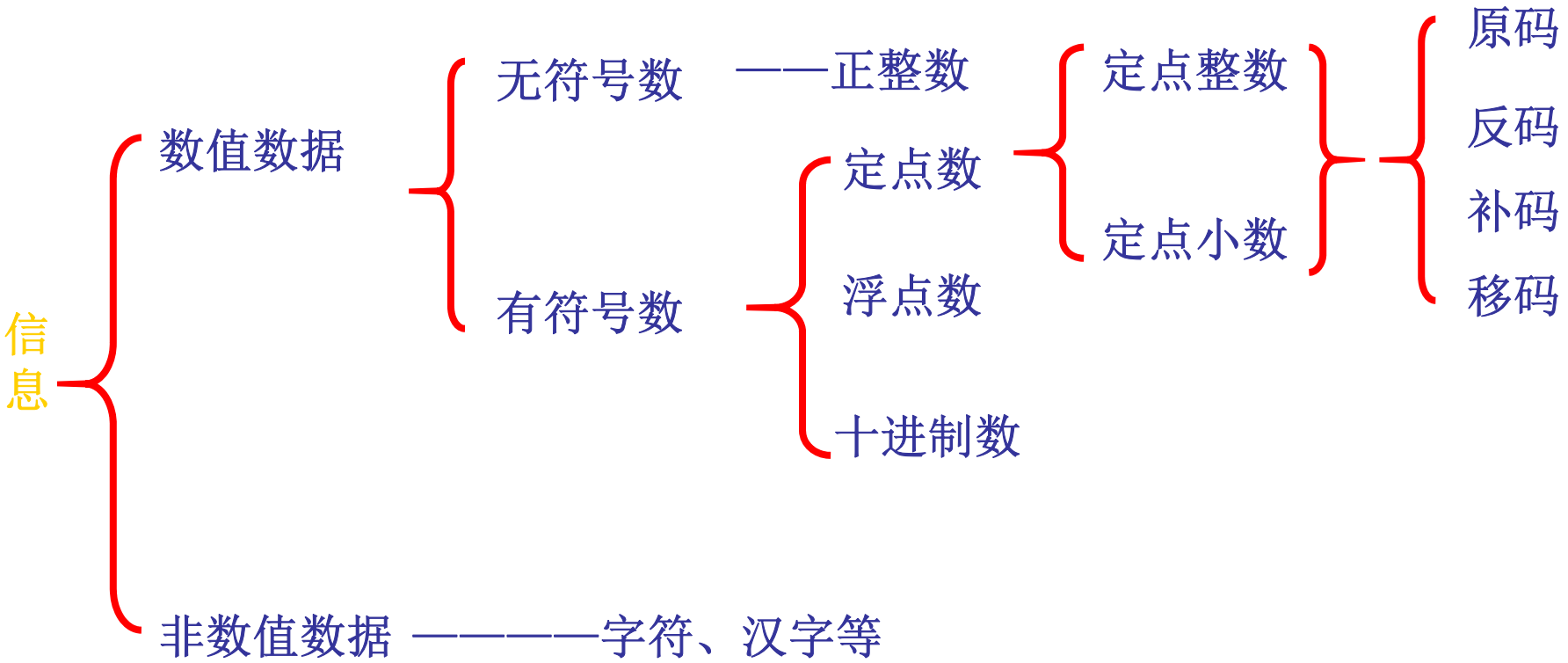
1111111110111 = 2^{12}-1-8

1111 1111 1110=1111 1111 1111-1

=2^{12}-1-1=4046

-17/128=-0.0010001

常用信息分类及表示：


信息
数值数据
无符号数
——正整数
有符号数
定点数
浮点数
十进制数
定点整数
定点小数
原码
反码
补码
移码
非数值数据 ————字符、汉字等

名词解释——真值和机器数

真值：正、负号加某进制数绝对值的形式称为真值。如+3, -5等，即实际值。
（书写用）

机器数：将符号和数值一起编码表示的二进制数称为机器数：

X=01011　　Y=11011

即真值在机器中的表示，称为机器数。（机器内部使用）

数据表示

计算机中常用的数据表示格式有两种：

定点格式——容许的数值范围有限，但要求的处理硬件比较简单。

浮点格式——容许的数值范围很大，但要求的处理硬件比较复杂。

1. 定点数的表示方法

定点表示：约定机器中所有数据的小数点位置是固定不变的。

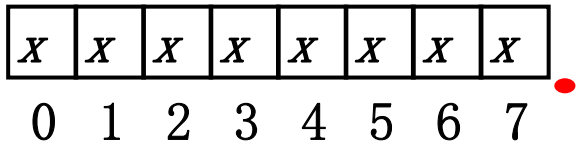
（由于约定在固定的位置，小数点就不再使用记号“.”来表示。）

通常将数据表示成纯小数或纯整数。

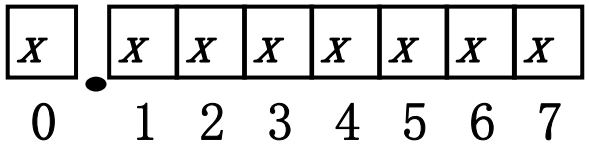
定点数：小数点位置固定不变的数

定点整数：小数点固定在最低位数的右面

定点小数：小数点固定在最高位数的后面，即纯小数表示



(a) 定点整数



(b) 定点小数

数值范围：

纯小数　　$0 \leqslant |x| \leqslant 1 - 2^{-n}$

纯整数　　$0 \leqslant |x| \leqslant 2^{n} - 1$

无符号数的编码

正整数

数值表示：

$$x = x_0x_1x_2\cdots x_n \quad x_i=\{0,1\}, \quad 0\leqslant i\leqslant n$$

$$x_0 2^n + x_1 2^{n-1} + \cdots + x_{n-1}2^1 + x_n$$

数值范围

$$0\leqslant x\leqslant 2^{n+1}-1$$

例如：

X = 010101

其数值$=2^4+2^2+2^0=21$

在数据处理的过程中，如不需要设置符号位可用全部字长来表示数值大小。如8位无符号数的取值范围是0~255（2^8-1）。

数的机器码表示

1、原码表示法　（Signed magnitude ）

(1)定点小数

若定点小数的原码形式为 $x0.\ x1\ x2\ \cdots\ xn$，（共n+1位）则原码表示的定义是：

$$[x]_{原} = \begin{cases} x & 0 \le x < 1 \\ 1 - x = 1 + |x| & -1 < x \le 0 \end{cases}$$

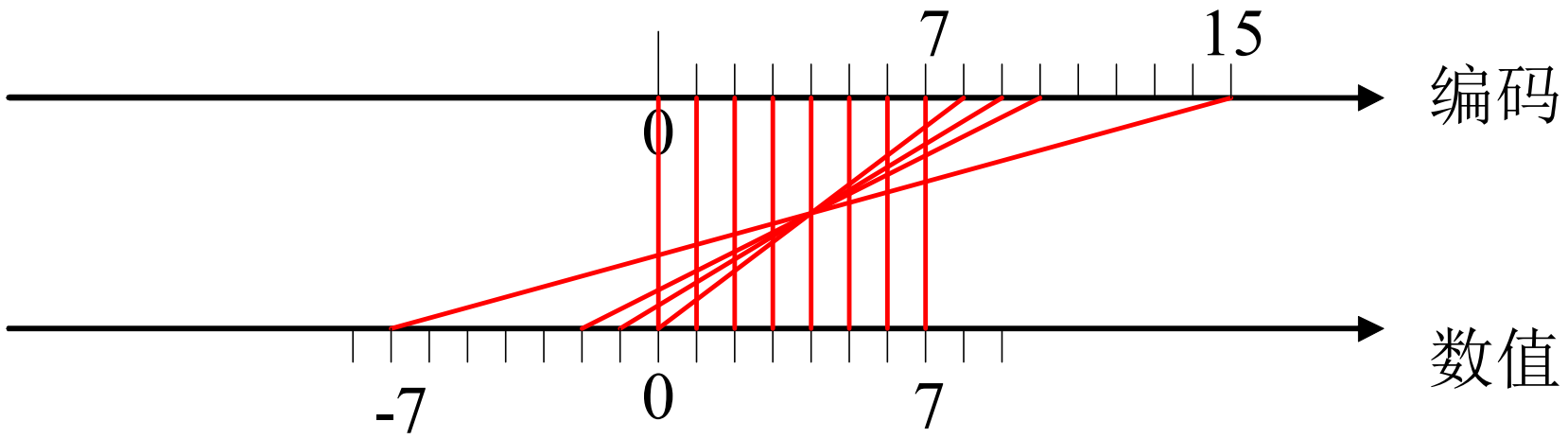
式中$[x]_{原}$是机器数，x是真值。

(2)定点整数

若定点整数的原码形式为 $x0\ x1\ x2\ \cdots\ xn$，则原码表示的定义是：

$$[x]_{原} = \begin{cases} x & 0 \le x < 2^n \\ 2^n - x = 2^n + |x| & -2^n < x \le 0 \end{cases}$$

原码在数轴上的表示



-7～+7　7个正数，7个负数，两个零

$-(2^n-1)$ ～（2^n-1） 数据位共n位

例1： $x = +0.1001$， 则 $[x]_{原} = 0.1001$

$x = -0.1001$， 则 $[x]_{原} = 1 + |x| = 1.1001$

对于0，原码机器中往往有“+0”、“-0”之分，故有两种形式：

$[+0]_{原} = 0.000...0$

$[-0]_{原} = 1.000...0$

例2： $x =$ 0.10110 ； -0.10110； 0.0000

$[x]_{原} =$ 0.10110； 1.10110； 0.0000 1.0000

例3： $x = +1011$ 总共用5位表示，n=4

$[x]_{原} =01011$

$x = -1011$

$[x]_{原} = 2^n + |x| = 10000 + |-1011| = 11011$

原码小数的表示范围：

最大值 ： $1-2^{-n}$

最小值:$-(1-2^{-n})$

若原码小数的位数是8位,其该数表示的最大值、最小值：

8位： 127/128，-127/128

原码整数的表示范围：

最大值 ： $2^{n}-1$ 最小值:$-(2^{n}-1)$

若原码整数的位数是8位,其表示的最大值、最小值 8位： 127，-127

(3)结论

原码为符号位加上数的绝对值，0正1负;

原码零有两个编码，+0和 -0编码不同;

原码加减运算复杂，乘除运算规则简单;

原码表示简单，易于同真值之间进行转换。

2、补码表示法 （ Two’s complement ）

同余的概念

假定有两个数a和b，若用某一个整数m去除，所得的余数相同，就称a,b两个数对m同余,记作：

$$a \equiv b \pmod{m}$$

假设X,Y,Z三个数，满足下列关系：Z=nX+Y (n为整数),则称Z和Y对模X是同余的，记作：

$$Z \equiv Y \pmod{X} \qquad Y \equiv Z \pmod{X}$$

(1)模的概念

假设两位十进制数计算:

- 77 - 38 = ?
- 77 + 62 = 139

77 - 38 = 77 + 62 = 39 （mod 100）

由此可以看出，减38和加62是等价的，前提是说对模为100是正确的。

这个100在数学上称为模数。

模的概念

计算机中运算器、寄存器、计数器都有一定的位数，不可能容纳无限大的任意数。当运算结果超出实际的最大表示范围，就会发生溢出，此时所产生的溢出量就是模（module）。

因此，可以把模定义为一个计量器的容量。如：一个4位的计数器，它的计数值为0--15。当计数器计满15之后再加1，这个计数器就发生溢出，其溢出量为16，也就是模等于16。

- 定点小数的溢出量为2，即模为2 ；
- 一个字长为n+1位的定点整数的溢出量为2^{n+1}，即以模为2^{n+1} 。

补码定义：

任意一个数X的补码记为$[x]_{补}$，

$$[x]_{补} = X + M \quad (\text{Mod } M)$$

当X>0时　X + M >M 自动丢失，

$$[x]_{补} = X \quad (\text{Mod } M)$$

当X<0时　X + M = M − | X | < M，

$$[x]_{补} = X + M \quad (\text{Mod } M)$$

(2) 定点小数

若定点小数的补码形式为 $x0.\ x1\ x2\ \cdots\ xn$, 则补码表示的定义是：

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

例： $x = +0.1011$, 则 $[x]_{补} = 0.1011$

$x = -0.1011$, 则 $[x]_{补} = 10+x = 10.0000-0.1011 = 1.0101$

对于0，$[+0]_{补} = [-0]_{补} = 0.0000$ (mod 2)

注意：0的补码表示只有一种形式。

(3) 定点整数

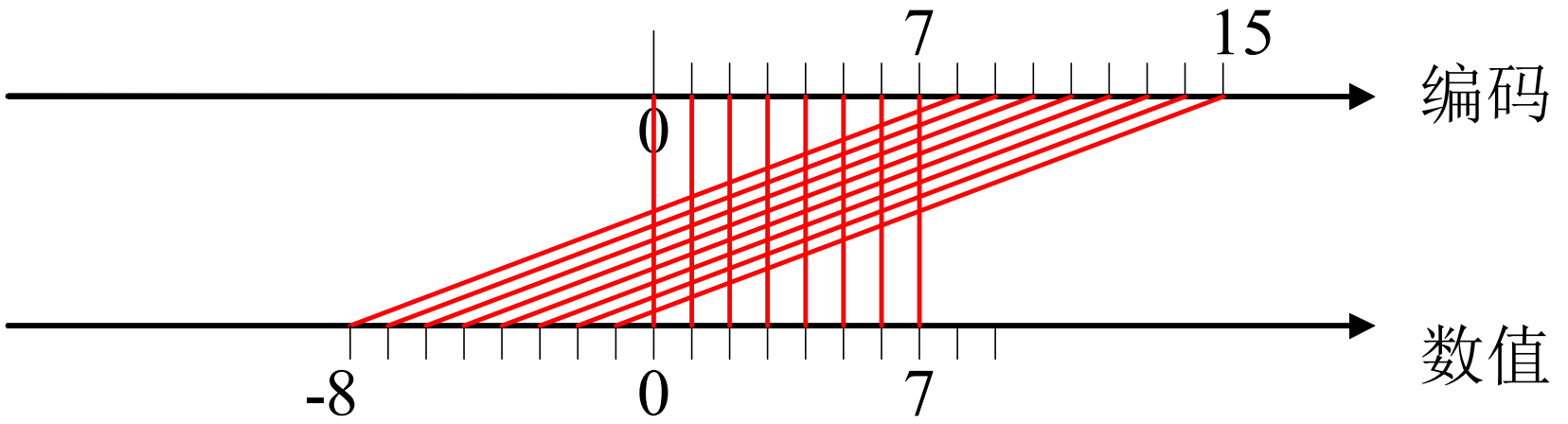
若定点整数的补码形式为 $x0\ x1\ x2\ \cdots\ xn$, 则补码表示的定义是：

$$[x]_{补} = \begin{cases} x & 0 \le x < 2^n \\ 2^{n+1} + x = 2^{n+1} - |x| & -2^n \le x \le 0 \end{cases} \quad (\mathrm{mod}\ 2^{n+1})$$

例：$x = +0111$, 则 $[x]_{补} = 00111$

$x = -0111$, 则 $[x]_{补} = 2^{4+1} - |-0111| = 100000 - 0111 = 11001$

补码在数轴上的表示



-8 ~ +7 正数7个，负数8个，零1个

-2^n ~ 2^n-1 数据位n位

补码的表数范围：

n+1位补码整数： 2^n-1 — -2^n

n+1 位补码小数：$1-2^{-n}$ — -1

若补码整数的位数是8位，其表示的最大值、最小值：-128 -- 127

若补码小数的位数是8位时，其该数表示的最大值、最小值：

-1 — $1-2^{-7}$ 即 -1 —127/128

(4) 特点

补码最高一位为符号位，0正1负；

补码零有唯一编码；

补码能很好用于加减运算。

补码满足 $[-x]_{补}+[x]_{补}=0$ $[+7]_{补}=00111$ $[-7]_{补}=11001$

最高位参与演算，与其它位一样对待。

扩展方便。5位的补码扩展为8位 00111 → 00000111

11001 → 11111001

算术移位。假设 $[x]_{补}= x0.\ x1\ x2\ \cdots\ xn$,

$[x/2]_{补}= x0.\ x0\ x1\ x2\ \cdots\ xn-1$

原符号位不变，符号位与数值位均右移一位，

[X]补 =10010 则 [X/2]补 =11001

最大的优点就是将减法运算转换成加法运算。

$[X+Y]_{补}= [X]_{补} + [Y]_{补}$

$[X-Y]_{补}= [X]_{补} + [-Y]_{补}$

例如：$X=(11)_{10}=(1011)_2$ ，$Y=(5)_{10}=(0101)_2$ ，已知字长n=5位，则

$[X]_{补}+ [-Y]_{补}= 01011 + 11011 = 100110 = 00110 = (6)_{10}$

注： 最高1位已经超过字长故应丢掉。

$[X - Y]_{补}= [0110]_{补}=00110$

补码编码的简便方法：

正数的补码在其二进制代码前加上符号位0；

负数的补码是将二进制代码前加0后，再全部按位取反，然后在最低位上加1。

例：设x=1010，y=-1010，求$[x]_{补}$和$[y]_{补}$。

解：根据补码的编码方法，正数的补码与它的二进制表示相同，所以加上符号位0后得：

$[x]_{补}$=01010　　$[x]_{补}$=00001010

负数的补码的编码方法：

1）将二进制代码前加0　0 1 0 1 0

2）再全部按位取反　1 0 1 0 1

3）然后在最低位上加1　1 0 1 1 0　　$[y]_{补}$=10110

例子

X=+0.11111111　　[X]补 =0.11111111

X=-0.11111111

[X]补 =1.00000000 + 0.00000001 = 1.00000001

X=-0.00000000

$[X]_{补}$ =1.11111111 + 0.00000001 = 10.00000000

=0.00000000

X=-1.0000

$[X]_{补}$= 0.1111+0.0001 =1.0000

原码与补码之间的转换：

已知原码求补码

正数　　$[X]_{补}=[X]_{原}$

负数　　符号除外，各位取反，末位加1

例：X= -1001001

$[X]_{原}=11001001$

$[X]_{补}=10110110+1=10110111$

$[X]_{补}= 2^{7+1}+X=100000000-1001001= 10110111$

$$\begin{array}{r} 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ -\quad\ 1\ 0\ 0\ 1\ 0\ 0\ 1 \\ \hline 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1 \end{array}$$

由$[X]_{补}$求$[-X]_{补}$

运算过程是：将$[X]_{补}$连同符号一起将各位取反，末位再加1。

例：设字长N=8位

X= +100 1001

$[X]_{补}$ = 0100 1001

各位取反 1011 0110

末位再加1 1011 0111

即：$[-X]_{补}$= 1011 0111

补码与真值之间的转换：

(1)

求值方法

$$x = -x_0 2^n + x_1 2^{n-1} + \cdots + x_{n-1} 2 + x_n$$

例如：　10000100　　的真值为　-128+4=-124

(2)

补码

- 符号位为“1”--负，余下求补为数值部分
- 符号位为“0”--正，余下为数值部分

例：$[X]_{补}$ = 0100 1001　　X= 0100 1001

例：$[X]_{补}$ = 1000 0000　　X= -1000 0000B = 80H =-128

3、反码表示法 （ One's complement ）

所谓反码，就是二进制的各位数码0变为1，1变为0。

(1)定点小数定义

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

一般情况下，

对于正数 $x = +0.x1x2\cdots xn$，有：

$$[x]_{反} = 0.x1x2\cdots xn$$

对于负数 $x = -0.x1x2\cdots xn$，有：

$$[x]_{反} = 1.\overline{x1}\ \overline{x2}\ \cdots\ \overline{xn}$$

例： x = 0.10110　　-0.10110　　0.0000

$[x]_{反}$= 0.10110　　1.01001　　0.0000　　1.1111

(2) 由反码求补码的公式

由反码与补码的定义

$[x]_{反} = (2-2^{-n}) + x$

$[x]_{补} = 2 + x$

得：$[x]_{补} = [x]_{反} + 2^{-n}$

即：若要一个负数变补码，其方法是符号位置1，其余各位0变1，1变0，然后在最末位(2^{-n})上加1。

(3) 定点整数定义

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

(4) 结论

负数反码为符号位跟每位数的反，0正1负；

反码零有两个编码，+0 和 -0 的编码不同；

反码难以用于加减运算；

反码的表数范围与原码相同。

4、移码表示法 （ Biased notation ）

保持数据原有大小顺序，便于进行比较操作。

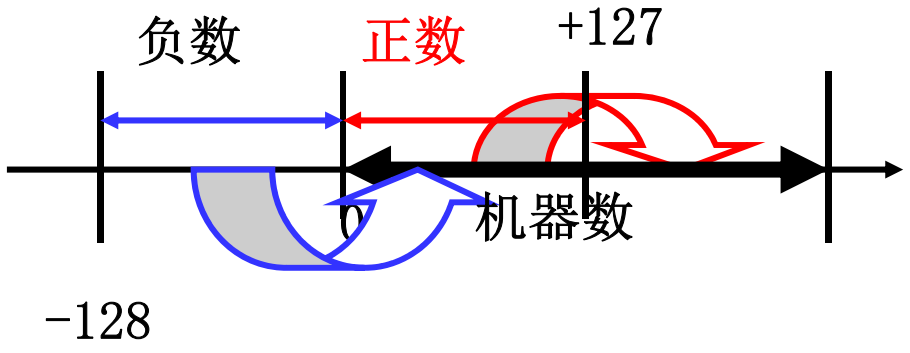
(1) 移码定义

移码通常用于表示浮点数的阶码。

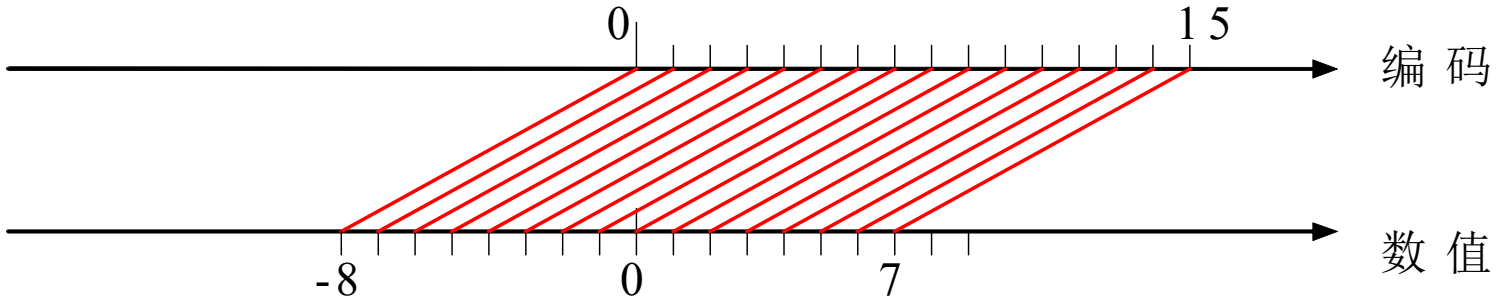
假设定点整数移码形式为 *x0 x1 x2 • • • xn*时，移码的定义是：

$$[x]_{移} = 2^n + x \qquad -2^n \le x < 2^n$$

8 位移码表示的机器数为数的真值在数轴上向右平移了 128 个位置。



表示范围： 00000000～11111111



例1：当正数 $x = +10101$ 时，

$$[x]_{移} = 2^5 + 10101 = 1,10101 \ ;$$

例2：当负数 $x = -10101$ 时，

$$[x]_{移} = 2^5 + x = 2^5 - 10101 = 0,01011$$

例3：0的移码是唯一的，即：

$$[+0]_{移} = [0]_{移} = 100\cdots00$$

[注意]：移码中符号位 $x0$表示的规律与原码、补码、反码相反

——“1”正，“0”负。

移码、补码和真值之间的关系

真值 （十进制）	真值 （二进制）	$[x]_{补}$ （补码）	$[x]_{移}$ （移码）
-128	-1000, 0000	1000, 0000	0000, 0000
-127	-0111, 1111	1000, 0001	0000, 0001
…	…	…	…
-1	-0000, 0001	1111, 1111	0111, 1111
0	0000, 0000	0000, 0000	1000, 0000
1	0000, 0001	0000, 0001	1000, 0001
…	…	…	
127	0111, 1111	0111, 1111	1111, 1111

(3) 移码的特点

- 在移码中，最高位为0表示负数，最高位为1表示正数，这与原码、补码、反码的符号位取值正好相反。
- 移码为全0时所对应的真值最小，为全1时所对应的真值最大！

因此，移码的大小直观地反映了真值的大小，这将有助于两个浮点数进行阶码大小比较。

- 真值0在移码中的表示形式是唯一的，即：

$$[+0]_{移} = [0]_{移} = 100\cdots00$$

- 移码把真值映射到一个正数域，所以可将移码视为无符号数，直接按无符号数规则比较大小。
- 同一数值的移码和补码除最高位相反外，其他各位相同。

几种机器编码简便方法对比

机器码	真值为正数	真值为负数
原码	符号位为零, 等于真值本身	符号位为一, 数值位为真值本身 简便编码方法：加符号位
补码	符号位为零, 等于真值本身	符号位为一, 逐位取反, 末位加一
反码	符号位为零, 等于真值本身	符号位为一, 逐位取反
移码	符号为一, 数值位为真值本身	符号位为零, 数值位逐位取反, 末位加一

例 子

机器码	+00001111	−00001111	+0.00001111	−0.00001111
原码	000001111	100001111	0.00001111	1.00001111
补码	000001111	111110001	0.00001111	1.11110001
反码	000001111	111110000	0.00001111	1.11110000
移码	100001111	011110001	小数无移码	小数无移码

码制表示法小结

[X]原、[X]反 、[X]补用“0”表示正号，用“1”表示负号；

[X]移用“1”表示正号，用“0”表示负号。

如果X为正数，则[X]原=[X]反=[X]补。

如果X为0，则 [X]补 、[X]移有唯一 编码，

[X]原、[X]反 有两种编码。

移码与补码的形式相同，只是符号位相反。

5、小结

数据四种机器表示法中：

(1)移码表示法主要用于表示浮点数的阶码。

(2)补码表示对加减法运算十分方便，因此目前机器中广泛采用补码表示法。

(3)在一些机器中,数用补码表示,补码存储,补码运算。

在有些机器中，数用原码进行存储和传送，运算时改用补码。

还有些机器在做加减法时用补码运算，在做乘除法时用原码运算。

例： 设机器字长16位，定点表示，尾数15位，数符1位，问：

(1)定点原码整数表示时，最大正数是多少?最小负数是多少?

(2)定点原码小数表示时，最大正数是多少?最小负数是多少?;

解：(1)定点原码整数表示

最大正数值$=(2^{15}-1)_{10}=(+32767)_{10}$

0111 1111 1111 1111

最小负数值$=-(2^{15}-1)_{10}=(-32767)_{10}$

1111 1111 1111 1111

(2)定点原码小数表示

最大正数值$=(1-2^{-15})_{10}=(+0.111...11)_2$

最小负数值$=-(1-2^{-15})_{10}=(-0.111..11)_2$

十进制数和数串的表示

十进制是人们最常用的数据表示方法，一些通用性较强的计算机上设有十进制数据的表示，可以直接对十进制数进行运算和处理。

十进制数的编码

用四位二进制数来表示一位十进制数，称为(Binary coded decimal)二进制编码的十进制数，简称BCD码。

四位二进制数可以组合出16种代码，能表示16种不同的状态，我们只需要使用其中的10种状态，就可以表示十进制数的0～9十个数码，而其他的六种状态为冗余状态。由于可以取任意的10种代码来表示十个数码，所以就可能产生多种BCD编码。BCD编码既具有二进制数的形式，又保持了十进制数的特点。

几种常见的BCD码

十进制	8421 码	2421 码	余 3 码
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100

8421码又称为NBCD码，其主要特点是：

(1) 它是一种有权码，四位二进制代码的位权从高到低分别为8、4、2、1。

(2) 简单直观。每个代码与它所代表的十进制数之间符合二进制数和十进制数相互转换的规则。

(3) 不允许出现1010～1111。这6个代码在8421码中是非法码。

8421码　　(8*X3+4*X2+2*X1+1*X0)

2421码的主要特点是：

(1)它也是一种有权码，四位二进制代码的位权从高到低分别为2、4、2、1。

(2) 它又是一种对9的自补码。即某数的2421码，只要自身按位取反，就能得到该数对9之补的2421码。

例如： 3的2421码是0011。3对9之补是6，而6的2421码是1100。

(3) 不允许出现0101～1010。这6个代码在2421码中是非法码。

2421码　　(2*X3+4*X2+2*X1+1*X0)

余3码的主要特点是：

(1) 这是一种无权码，但也可看作是一种特殊的有权码，即在8421码的基础上加+3（+0011）形成的，故称余3码。在这种编码中各位的“1”不表示一个固定的十进制数值，因而不直观。

(2) 它也是一种对9的自补码。

(3) 不允许出现0000～0010、1101～1111。这6个代码在余3码中是非法码。

余三码　　(8*X3+4*X2+2*X1+1*X0)+0011

十进制数串

1. 非压缩的十进制数串

非压缩的十进制数串中一个字节存放一个十进制数或符号的ASCII-7码。

非压缩的十进制数串又分成前分隔式数字串和后嵌入式数字串两种格式。在前分隔式数字串中，符号位占用单独一个字节，放在数值位之前，正号对应的ASCII码为2BH，负号对应的ASCII码为2DH。在后嵌入式数字串中，符号位不单独占用一个字节，而是嵌入到最低一位数字里边去。若数串为正，则最低一位数字0～9的ASCII码不变（30H～39H）；若数串为负，把负号变为40H，并将其与最低数值位相加，此时数字0～9的ASCII码变为70H～79H。

2. 压缩的十进制数串

压缩的十进制数串，一个字节可存放两位BCD码表示的十进制数，既节省了存储空间，又便于直接进行十进制算术运算。

在主存中，一个压缩的十进制数串占用连续的多个字节，每位数字仅占半个字节，其值常用8421码表示。符号位也占半个字节，并存放在最低数值位之后，通常用CH表示正号，DH表示负号。在这种表示中，规定数字的个数加符号位之和必须为偶数；当和为奇数时，应在最高数值位之前补0H（即第一个字节的高半字节为“0000”）。

非数值数据的表示

非数值数据，又称为字符数据，通常是指字符、字符串、图形符号和汉字等各种数据，它们不用来表示数值的大小，一般情况下不对它们进行算术运算。

1、ASCII码

“美国标准信息交换代码”(American Standard Code for Information Interchange)，简称ASCII码。7位二进制编码，可表示2^7=128个字符。

ASCII码中，编码值0～31不对应任何可印刷（或称有字形）字符，通常称它们为控制字符，用于通信中的通信控制或对计算机设备的功能控制。编码值为32的是空格（或间隔）字符SP。编码值为127的是删除控制DEL码。其余的94个字符称为可印刷字符。

在计算机中，通常用一个字节来存放一个字符。在ASCII码表中，数字和英文字母都是按顺序排列的，只要知道其中一个的二进制代码，不要查表就可以推导出其他数字或字母的二进制代码。

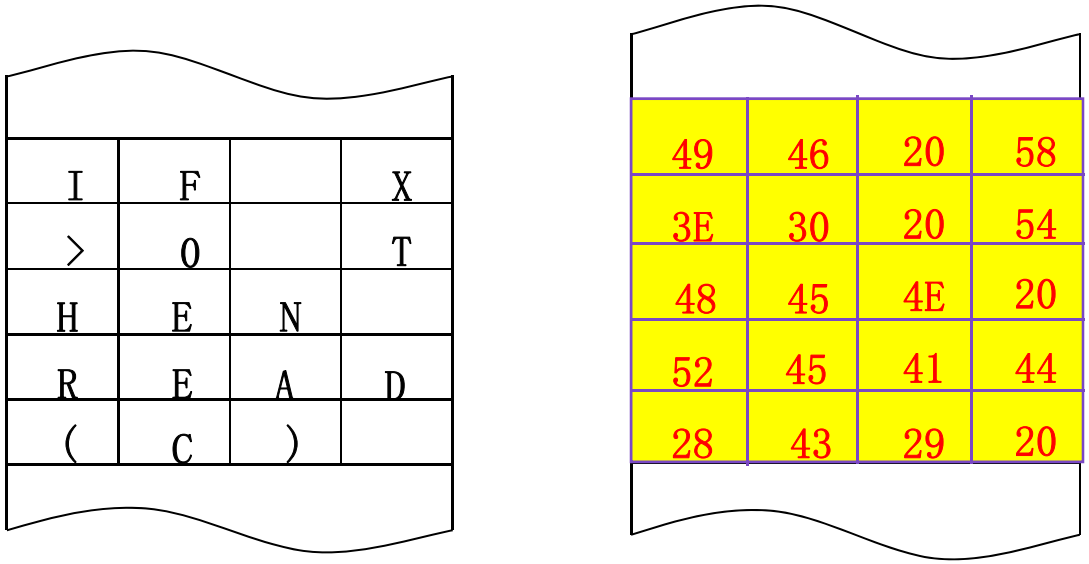
ASCII字符编码表

$b_6b_5b_4$ / $b_3b_2b_1b_0$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	RO	RS	.	>	N	↑	n	~
1111	SI	US	/	?	O	_	o	DEL

2、字符串的存放

字符串是指一串连续的字符。例如，字符串IF X>0 THEN READ (C)。

向量存放法在存储器中占用一片连续的空间，每个字节存放一个字符代码，字符串的所有元素（字符）在物理上是邻接的。在字长为32位的存储器，每一个主存单元可存放4个字符，整个字符串需5个主存单元。在每个字节中实际存放的是相应字符的ASCII码。



汉字标准

GB2312-80国家标准

- 1981年，GB2312-80国家标准，包括6763个汉字/682个非汉字字符，称为国标码或国际交换码
- GB2312字符集的构成：
  - 一级常用汉字3755个，按汉语拼音排列
  - 二级常用汉字3008个，按偏旁部首排列
  - 非汉字字符682个

- GB13000-1993
  - 20902个汉字 (Unicode 1.1版本)
- 汉字扩展规范GBK1.0 标准1995（非国家标准）
  - 21003个字符（兼容GB2312）
- GB18030-2000(1/2/4字节编码)
  - 27484汉字 （向下兼容GB2312 GBK，GB13000）

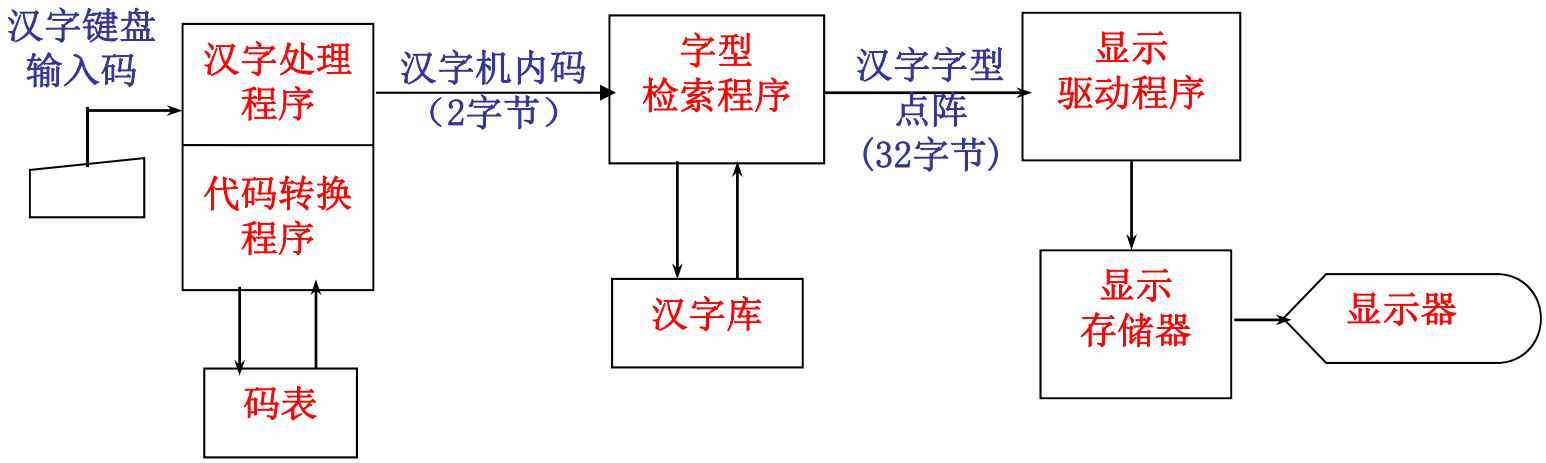
GBK2K从根本上解决了字位不够，字形不足的问题。

汉字的表示方法

1、涉及多种编码：首先将汉字转换成计算机能接收的编码，称为汉字输入码（外码），输入码进入计算机后必须转换成汉字内码才能进行处理。为了显示输出汉字或打印输出汉字，需要经过一个变换，将汉字内码转换成汉字字形码。此外，为了使不同的汉字处理系统之间能够交换信息，还应存在汉字交换码。

注意：汉字的输入编码、汉字内码、字形码是计算机中用于输入、内部处理、输出三种不同用途的编码，不要混为一谈。

汉字处理



汉字的输入码

1）数字编码

数字编码是用数字串代表一个汉字的输入，常用：区位码

例：中—5448　国—2590　→—0190　Σ—0138　か—0411

2）拼音码

用汉语拼音输入汉字。例：ZHONG—中　GUO—国

3）字型编码

以汉字的形式确定编码。常用：五笔字型

4）其它输入方法

郑码、智能等

汉字机内码

机内码——用于汉字的存储、交换、查询等

汉字在计算机内部其内码是唯一的。因为汉字处理系统要保证中西文的兼容，当系统中同时存在ASCII码和汉字国标码时，将会产生二义性。例如：有两个字节的内容为30H和21H，它既可表示汉字“啊”的国标码，又可表示西文“0”和“!”的ASCII码。为此，汉字机内码应对国标码加以适当处理和变换。

GB码的机内码为二字节长的代码，它是在相应GB码的每个字节最高位上加“1”，即

汉字机内码＝汉字国标码＋8080H

例如：上述“啊”字的国标码是3021H，其汉字机内码则是B0A1H。

需要注意的是：汉字区位码并不等于汉字国标码，它们两者之间的关系可用以下公式表示：

国标码＝区位码（十六进制）＋2020H

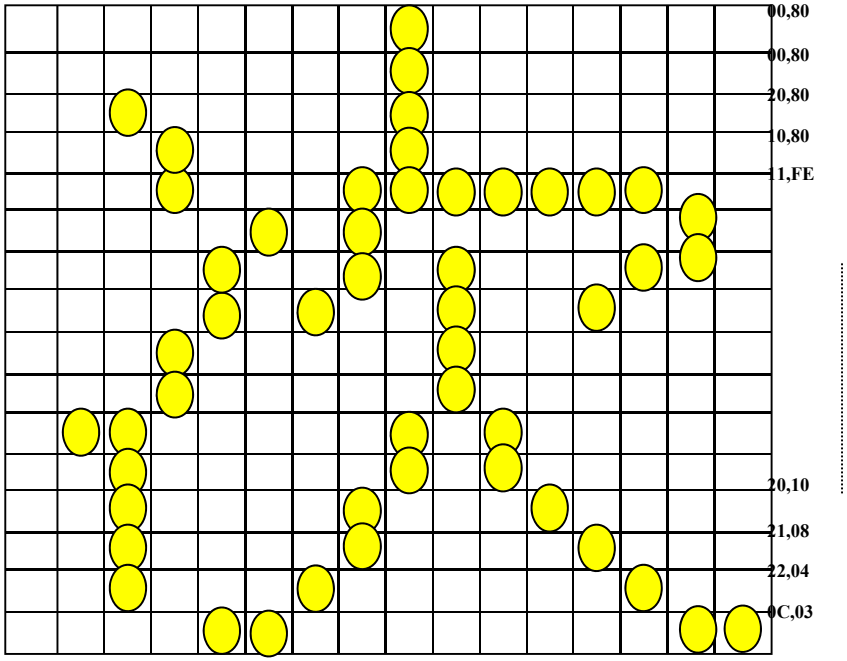
例：汉字“春”的区位码为“20-26”，它的国标码。

国标码：　34H 3AH

汉字字形码

主要显示用于汉字输出或汉字打印输出

- 字模码是用点阵表示的汉字字型代码，是汉字的输出形式。
- 字模点阵的信息量是很大的，所占存储空间也很大。以16*16为例，每个汉字要占用32个字节，
- 因此字模点阵只能用来构成汉字库，而不能用于机内存储。

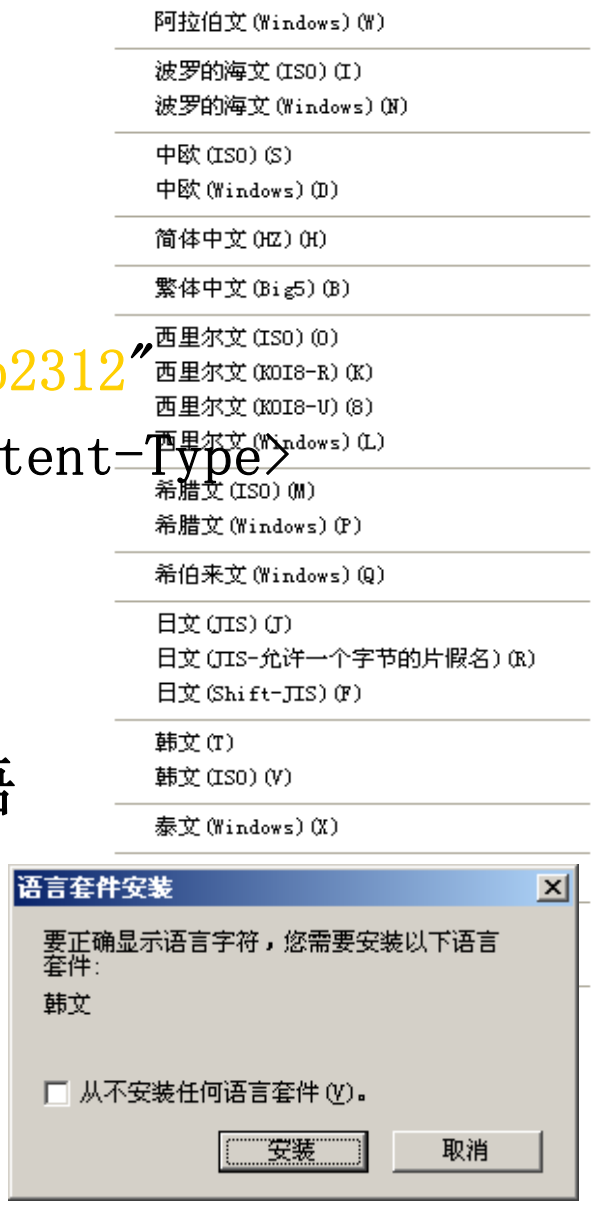


Unicode

- 用于克服字符数字的限制
- 为所有语言中的字符分配唯一的代码
- 16 bit 字符集，65536 Unicode 字符
- 提供唯一的代码
  - 不论任何平台
  - 不论任何程序
  - 不论任何语言

Charset

- <META content="text/html; charset=gb2312"
  … http-equiv=Content-Type>
- charset=gb2312 简体中文
  charset=big5 繁体中文
  charset=EUC_KR 韩语
  charset=Shift_JIS 或 EUC_JP 日语
  charset=KOI8-R/Windows-1251俄语
  charset=iso-8859-2 中欧语系
  charset=utf-8 unicode多语言



Universal Character Set ISO

- UCS
  - ISO 10646
  - UCS-2 UCS-4
- UTF (Unicode Transform format)
  - UTF-7
  - UTF-8
  - UTF-16

统一代码

Unicode是对国际标准ISO10646编码的一种称谓（ISO/IEC10646是一个国际标准，亦称大字符集，它是ISO于1993年颁布的一项重要国际标准，其宗旨是全球所有文种统一编码）

Unicode是两字节的全编码，对于ASCII字符它也使用两字节表示。UNICODE则一律使用两个字节表示一个字符，最明显的好处是它简化了汉字的处理过程。

Unicode的基本方法是用一个16位的数来表示每个符号，这种符号集可表示65536个不同的字符或符号。被称为基本多语言平面（BMP）。这个空间已经非常大了，但设计者考虑到将来某一天它可能也会不够用，所以采用了一种可使这种表示法使用得更远的方法。

当用两字节来表示Unicode字符时，使用的是UCS-2编码，但尽管如此，也允许在UCS-2文本中插入一些UCS-4字符。为此，在BMP中，保留了两个大小为1024的块，这两个块中任何位置都不能用来表示任何符号。UCS-4的两个16位字每个表示一个数，这个数是UCS-2 BMP中1024个数值中的一个。这两个数的组合可以表示多达100多万个自定义的UCS-4字符。