计算机组成原理与系统结构

第三章

信息编码与数据表示

http://jpkc.hdu.edu.cn/computer/zcyl/dzkjdx/

第三章　信息编码与数据表示

3.1 数值数据的表示

3.2 数据格式

3.3 定点机器数的表示方法

3.4 浮点机器数的表示方法

3.5 非数值数据的表示

3.6 校验码

3.7 现代计算机系统的数据表示

本章小结

BACK

3.1 数值数据的表示

一 进位计数制

二 不同数制之间的相互转换

三 十进制数的编码

一、进位计数制

- 数制的两大要素：
  - 基数R：指在这种进位制中允许使用的基本数码个数。
  - 权W_i：权也称位权，指某一位i上的数码的权重值，即权与数码所处的位置i有关。 $W_i = R^i$。
- 基数为R的数制称为R进制数。
- R进制数的主要特点就是逢R进1。

一、进位计数制

- 思考：何谓十进制、二进制、八进制、十六进制？

进制	基数R	权W_i	数码符号
十进制	R=10	10^i	0～9
二进制	R=2	2^i	0、1
八进制	R=8	8^i	0～7
十六进制	R=16	16^i	0～9、A～F

一、进位计数制

- 假设任意数值N用R进制数来表示，形式为：
  $N=(D_{m-1}D_{m-2}\ldots D_0 . D_{-1}D_{-2}\ldots D_{-k})_R$
  - 其中，D_i为该进制的基本符号，$D_i \in [0, R-1]$，$i = -k, -k+1, \ldots\ldots, m-1$；小数点在$D_0$和$D_{-1}$之间。
- 则数值N的**实际值**为：加权求和

$$N=\sum_{i=-k}^{m-1}(D_i \times R^i)$$

十进制

一、进位计数制

- 例1：$(2345.459)_{10}=2\times10^3+3\times10^2+4\times10^1+5\times10^0+4\times10^{-1}+5\times10^{-2}+9\times10^{-3}$

- 例2：$(11011.011)_2=1\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^{-0}+0\times2^{-1}+1\times2^{-2}+1\times2^{-3}=(27.375)_{10}$

- 例3：$(123.67)_8=1\times8^2+2\times8^1+3\times8^0+6\times8^{-1}+7\times8^{-2}=(83.859375)_{10}$

二、不同数制之间的相互转换

1. 常用的几种数制的对应关系
2. 二、八、十六进制转换为十进制
3. 十进制转换为二、八、十六进制

1、常用的几种数制的对应关系

十进制	二进制	八进制	十六进制	十进制	二进制	八进制	十六进制
0	0000	0	0	8	1000	10	8
1	0001	1	1	9	1001	11	9
2	0010	2	2	10	1010	12	A
3	0011	3	3	11	1011	13	B
4	0100	4	4	12	1100	14	C
5	0101	5	5	13	1101	15	D
6	0110	6	6	14	1110	16	E
7	0111	7	7	15	1111	17	F
				16	10000	20	10

2、二、八、十六进制转换为十进制

- **转换方法：加权求和。**

$$N=\sum_{i=-k}^{m-1}(D_i\times R^i)$$

  - 例：$(5AC.E6)_{16}= 5\times16^2+ 10\times16^1 +12\times16^0 +14\times16^{-1}+ 6\times16^{-2} = (1452.8984375)_{10}$

- 十进制（Decimal）、二进制（Binary）、八进制（Octal）、十六进制（Hexdecimal）数分别用D、B、Q、H来标志。
- 例如：$(1011)_2$→$(1011)_B$→1011B→1011b
  - $(123.45)_{10}$→$(\ 123.45\ )_D$→ 123.45D→123.45
  - $(2B.D)_{16}=(2B.D)_H=(43.8125)_{10}=(53.64)_Q$

3、十进制转换为二、八、十六进制

- **转换方法**：可以分为以下两种方法
  - **直接转换**：十进制 → 二、八、十六进制
  - **间接转换**：十进制 → 二进制 → 八、十六进制

- （1）十进制转化为R进制
- （2）二进制转化为八、十六进制

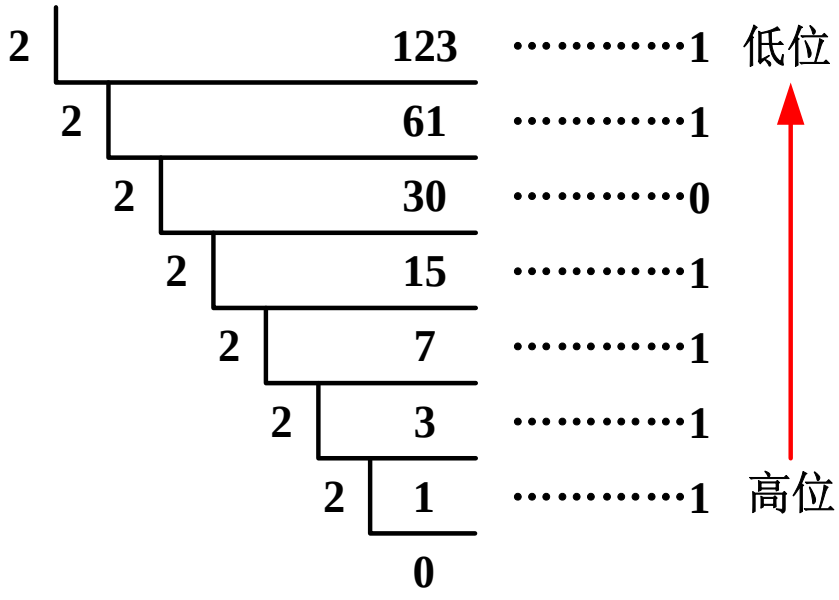
（1）十进制转化为R进制

- 转换方法
  - **整数部分：**除以R取余，**先得低位**，直到商为0。
  - **小数部分：**乘R取整，**先得高位**，直到积为0或者达到精度要求为止。

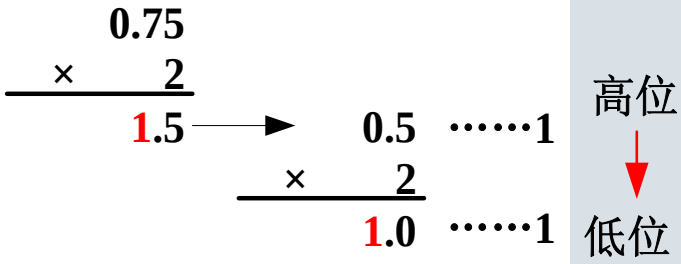
（1）十进制转化为R进制

❖ 例：（123.75）$_{10}$=（ 1111011.11 ）$_2$

❖ 整数部分



❖ 小数部分



❖ 练习：（123.75）$_{10}$=（ 173.6 ）$_8$

小数部分的精度要求

- 当小数部分不能整除为二进制时，则乘以2取整的过程中，积不会为0；或者当小数部分转化为二进制位数很长，这时由精度来决定二进制位数。
  - $(114.35)_{10}=(\ ?\ )_2$无法整除
  - $(0.6875)_{10}=(\ ?\ )_2$位数太长
- 例：$(114.35)_{10}=(\ ?\ )_2$，要求精度大于10%。
  - 要求“=”左右两边的十进制值的差的绝对值<10%；因为$10\%>2^{-4}$，所以只需取4位二进制小数即可满足要求。
  - $(114.35)_{10}=(1110010.\ 0101)_2$
- 思考：若要求转换精度大于1%，则二进制小数取几位？

（2）二进制转化为八、十六进制

- 二进制→八进制
  - 以小数点为中心分别向两边分组，每三位一组，写出对应的八进制数字。（不够位数则在两边加0补足3位 ）
- 二进制→十六进制

001 011 111. 110

心分别向两边分组，每四位一组，写出对的十六进制符号。（不够位数则在两边加0补足位 ）

- 例：$(1011111.11)_2 = (\quad 137.6 \quad)_8$

  $(1011111.11)_2 = (\quad 5F.C \quad)_{16}$

0101 1111. 1100

思考1：八、十六进制如何转化为二进制？

- **八进制→二进制**：将每位八进制数展开为3位二进制数，整数的最高位0和小数的最低位0可以略去。
- **十六进制→二进制**：将每位十六进制数展开为4位二进制数，整数的最高位0和小数的最低位0可以略去。

- 例：（765.23）$_8$=（ 111 110 101.010 011 ）$_2$
- 例：（765.23）$_{16}$=（ 111 0110 0101.0010 0011 ）$_2$

思考2：计算机中为什么采用二进制表示数据？

- ① 具有二值状态的物理器件容易实现。
- ② 二进制数据的抗干扰性强，可靠性高。
- ③ 二进制的运算规则简单，硬件实现容易。
- ④ 具有逻辑特性，可代表“真假”、“是非”。

三、十进制数的编码

- 提出的问题：如何在计算机内使用二进制来表示十进制数据？
- 1、二－十进制码（BCD码）
- 2、十进制数串的表示方法

1、二一十进制码（BCD码）

- BCD（Binary Coded Decimal）码：使用二进制来编码十进制数字0～9。
- **编码方法：**一般使用4位二进制编码来表示1位十进制数字，在16个编码中选用10个来表示数字0～9。不同的选择构成不同的BCD码。
- **分类：**
  - **有权码：**编码的每一位都有固定的权值，加权求和的值即是表示的十进制数字。如8421码、2421码、5211码、4311码、84 -2-1码等。
  - **无权码：**编码的每一位并没有固定的权，主要包括格雷码、余3码等。

1、二—十进制码（BCD码）

十进制数	8421码	2421码	5211码	4311码	84-2-1码	格雷码	余3码
0	0000	0000	0000	0000	0000	0000	0011
1	0001	0001	0001	0001	0111	0001	0100
2	0010	0010	0011	0011	0110	0011	0101
3	0011	0011	0101	0100	0101	0010	0110
4	0100	0100	0111	1000	0100	0110	0111
5	0101	1011	1000	0111	1011	1110	1000
6	0110	1100	1010	1011	1010	1010	1001
7	0111	1101	1100	1100	1001	1000	1010
8	1000	1110	1110	1110	1000	1100	1011
9	1001	1111	1111	1111	1111	0100	1100

几种常见的BCD码

① 8421码：

- 特点：4位二进制数位的权从高到低依次是8、4、2、1；8421码实际上就是十进制数字0～9的二进制编码本身。
- 是最常用的一种BCD码，在没有特别指出的一般情况下，所提到的BCD码通常就是指8421码。

② 余3码：对应的8421码加上0011构成的。是一种自补码，即任何两个相加之和等于9的编码，互为反码 。

几种常见的BCD码

③格雷码：

- 特点：又叫循环码，它的任何相邻的两个编码（例如2和3、7和8、9和0等）之间只有一位二进制位不同。
- 优点：是用它构成计数器时，在从一个编码变到下一个编码时，只有一个触发器翻转即可，波形更完美、可靠。
- 格雷码的编码方案有许多种。

2、十进制数串的表示方法

- **字符串形式：**用ASCII码来表示十进制数字或符号位，即1个字节存放1位十进制数字或符号位。
- **压缩的十进制数串形式：**用BCD码来表示十进制数字，即1个字节存放2个十进制的数字；符号位放在最低位数字位之后，一般用C（12）表示正号，用D（13）表示负号。
  - 例如 ＋258被表示成258CH，占用两个字节，－34被表示为034DH，也占用两个字节。
- **共同点：必须给出它在主存中的首地址和位长。**

2、十进制数串的表示方法

- 采用十进制表示数据的优点是：
  - 对于需要大量地进行输入输出数据而运算简单的场合，大大减少了二-十进制转换，提高了机器的运行效率；
  - 十进制数串的位长可变，许多机器中规定该长度从0到31，有的甚至更长。不受定点数和浮点数统一格式的约束，从而提高了数据的表示范围和运算精度。

3.2 数据格式

一 机器数

二 小数点的表示方法

一、机器数

- **机器数**：数值数据在计算机中的表示形式。
- **特点**：
  - 表示的**数值范围受**计算机字长的**限制**；
  - 机器数的**符号位**必须被数值化为二进制**0和1**；
  - 机器数的**小数点**是用**隐含规定**的方式来表达的。
- **真值**：机器数所真正表示的数值，一般使用数值（二进制或十进制）前冠以“+”、“—”符号这种方法来书写。
- **机器数的编码方法**：**原码、反码、补码、移码。**

一、机器数

- 计算机中参与运算的数值数据有两种：
  - 无符号数据（Unsigned ）：所有的二进制数据位数均用来表示数值本身，没有正负之分。
  - 带符号数据（Signed）：则其二进制数据位，包括符号位和数值位。
- 思考：计算机硬件如何区分无符号数据和带符号数据呢？
- 例： Intel X86系列CPU
- 假设AX＝（1111111111111111）$_2$，BX＝（0000000000000001）$_2$，那么执行下面两段程序时，计算机硬件将把AX和BX中的数据看成是不同的数据。

一、机器数

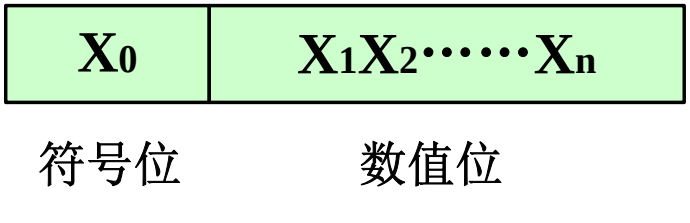
- 程序A： AX=0FFFFH，BX=0001H
  - CMP　　AX,BX　　;结果影响标志位
  - JL　　L1　　;有符号数小于转移
  - 执行JL指令时，操作数AX和BX被当作有符号数据，AX＝（－1）$_{10}$，BX＝（＋1）$_{10}$，所以执行结果是转移到L1标号处执行。
- 程序B： AX=0FFFFH，BX=0001H
  - CMP　　AX,BX
  - JB　　L1　　;无符号数小于转移
  - 执行JB指令时，操作数AX和BX被当作无符号数据，AX＝（65535）$_{10}$，BX＝1，所以执行结果是不转移，顺序执行。

计算机硬件不区分无符号数据和带符号数据，由程序（指令）来区分。

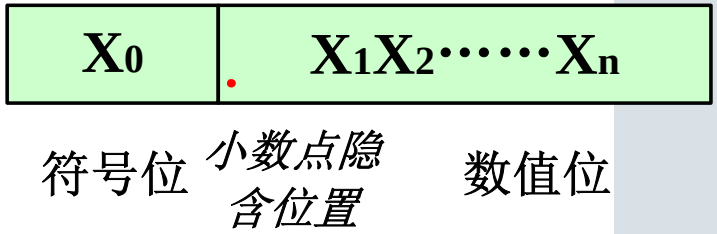
二、小数点的表示方法

- 在机器数中，小数点及其位置是隐含规定的；有两种隐含方式：
  - 定点机器数：小数点的位置是固定不变的
  - 浮点机器数：小数点的位置是浮动的
- 定点机器数分为定点小数、定点整数两种。
- 浮点机器数中小数点的位置由阶码规定，因此是浮动的。

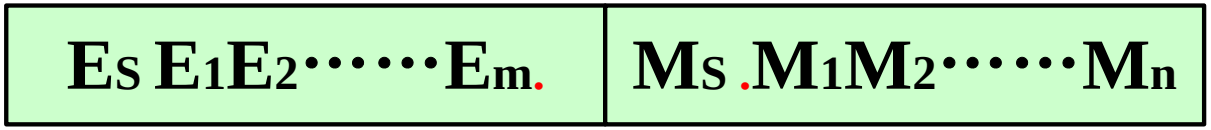
二、小数点的表示方法



（a） 定点整数格式



（b）定点小数格式



（c）浮点数格式

3.3 定点机器数的表示方法

- 定点机器数的小数点的位置是固定不变的，可以分为两种：
  - 定点小数：用于表示纯小数，小数点隐含固定在最高数据位的左边，**整数位则用于表示符号位**。
  - 定点整数：用于表示纯整数，小数点位置隐含固定在最低位之后，**最高位为符号位**。

- 一、原码表示法
- 二、补码表示法
- 三、反码表示法
- 四、移码表示法
- 五、定点机器数转换

对比

一、原码表示法

“，”和“．”只用于助记，在计算机中并无专用部件来表示

- 1、**表示方法**：最高位为符号位，其他
  - 符号位：0—正数，1—负数。
  - 数值位：与绝对值相同。
- 对于定点整数：
  - 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$[X]_{原}=1,X_1X_2\cdots\cdots X_n$ 。
- 对于定点小数：
  - 若$X=+0.X_1X_2\cdots\cdots X_n$ ，则$[X]_{原}=0.X_1X_2\cdots\cdots X_n$ ；
  - 若$X=-0.X_1X_2\cdots\cdots X_n$ ，则$[X]_{原}=1.X_1X_2\cdots\cdots X_n$ 。

一、原码表示法

- 例1：X=1011，Y＝－1011，则：

  $[X]_{原}$＝ 0,1011 ；$[Y]_{原}$＝ 1,1011 ；

- 例2：X=0.1101，Y＝- 0.1101，则：

  $[X]_{原}$＝ 0.1101 ；$[Y]_{原}$＝ 1.1101 ；

- 例3： X=1011， Y＝- 0.1101，求X和Y的8位原码机器数。

  $[X]_{原}$＝ 0,0001011 ；$[Y]_{原}$＝1. 1101000；

- 例4：$[0]_{原}$＝？

一、原码表示法

- 2、 0的表示：0 的原码表示有两种形式，即分别按照正数和负数表示。
  - $[+0]_{原}$= 00…0　　$[-0]_{原}$= 10…0
- 3、表示范围：对于n＋1位原码机器数X，它所能表示的数据范围为：
  - 定点整数：$-(2^n-1) \leqslant X \leqslant 2^n-1$
  - 定点小数：$-(1-2^{-n}) \leqslant X \leqslant 1-2^{-n}$

包括1位符号位，n位数值位

•思考：16位定点整数的原码表示范围？

一、原码表示法

❖ **4、数学表示：编码与真值之间的数学关系**

定点整数

$$[X]_{原}=\begin{cases}X & X\geq 0\\ 2^{n}-X & X\leq 0\end{cases}$$

定点小数

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

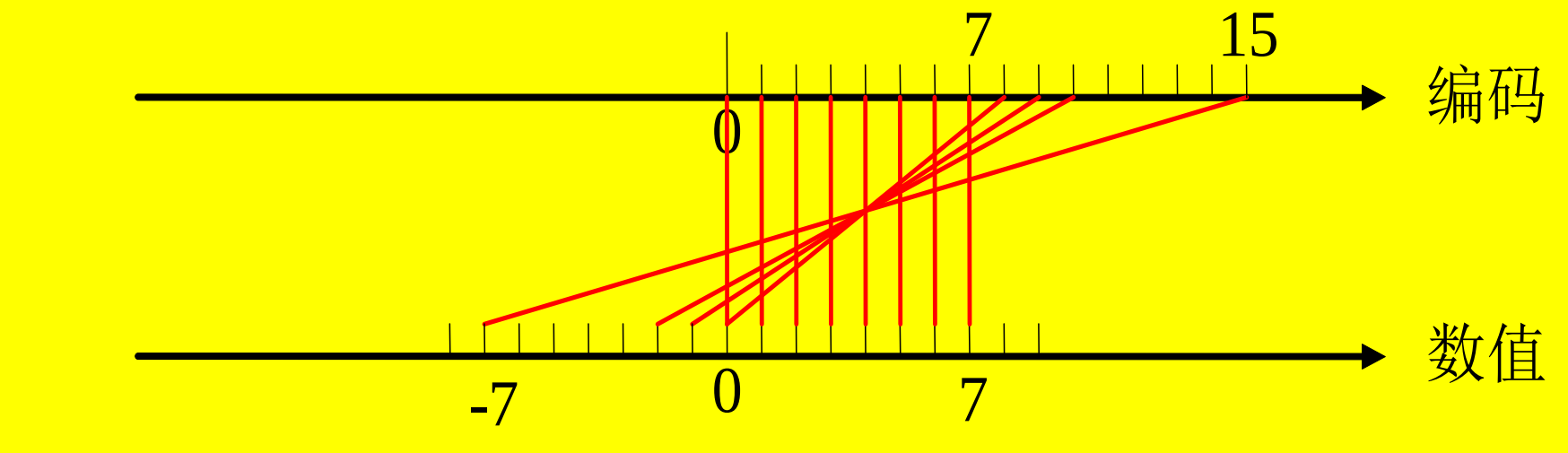
一、原码表示法

- 4位原码机器数（整数）对应的真值

机器数	原码对应真值
0000	0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

一、原码表示法

- 4位原码机器数（整数）在数轴上的表示
  - 原码机器数编码与真值的对应



二、补码表示法

- 1、**表示方法：**最高位为符号位，其他位为数值位。
  - 符号位：0－正数，1－负数。
  - 数值位：正数时，与绝对值相同；**负数时，为绝对值取反后，末位加1。**
- 对于定点整数：
  - 若$X=+X_1X_2\cdots\cdots X_n$，则$[X]_{补}=0,X_1X_2\cdots\cdots X_n$；
  - 若$X=-X_1X_2\cdots\cdots X_n$，则$[X]_{补}=1,\overline{X_1}\,\overline{X_2}\cdots\cdots\overline{X_n}+1$。
- 对于定点小数：
  - 若$X=+0.X_1X_2\cdots\cdots X_n$，则$[X]_{补}=0.X_1X_2\cdots\cdots X_n$；
  - 若$X=-0.X_1X_2\cdots\cdots X_n$，则$[X]_{补}=1.\overline{X_1}\,\overline{X_2}\cdots\cdots\overline{X_n}+0.00\cdots\cdots1$。

二、补码表示法

- 例1：X=1011，Y＝－1011，则：

  $[X]_{补}$＝ 0,1011 ；$[Y]_{补}$＝ 1,0101 ；

- 例2：X=0.1101，Y＝- 0.1101，则：

  $[X]_{补}$＝ 0.1101 ；$[Y]_{补}$＝ 1.0011 ；

- 例3： X=1011， Y＝- 0.1101，求X和Y的8位补码机器数。

  $[X]_{补}$＝ 0,0001011 ；$[Y]_{补}$＝ 1. 0011000 ；

- 例4：$[0]_{补}$＝？

二、补码表示法

- 2、 0的表示：0 的补码表示形式是唯一的，即分别按照正数和负数表示均一致，为全零。
  - $[+0]_{补}=$ 00…0　　$[-0]_{补}=$ 00…0
- 3、表示范围：对于n＋1位补码机器数X，它所能表示的数据范围为：
  - 定点整数：$-2^n \leqslant X \leqslant 2^n-1$
  - 定点小数：$-1 \leqslant X \leqslant 1-2^{-n}$

包括1位符号位，n位数值位

- 计算机中的整型数据（int）均用补码来表示。

思考题：

- 32位微机中，C程序定义了两个变量x，y：
- Int x;
- Unsigned int y;
- 问：x，y的数据取值范围？

- x是32位补码表示的整数：$-2^{31} \le X \le 2^{31}-1$
- y是32位无符号整数：$0 \le X \le 2^{32}-1$

二、补码表示法

❖4、数学表示：

定点整数

$$[X]_{补}=\begin{cases}X & X\geq 0\\ 2^{n+1}+X & X<0\end{cases}$$

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

定点小数

$$[X]_{补}=\begin{cases}X & X\geq 0\\ 2+X & X<0\end{cases}$$

模2补码

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

二、补码表示法

- 4位补码机器数（整数）对应的真值

$[X]_{补}= X_0 \quad X_1 \quad X_2 \quad X_3$

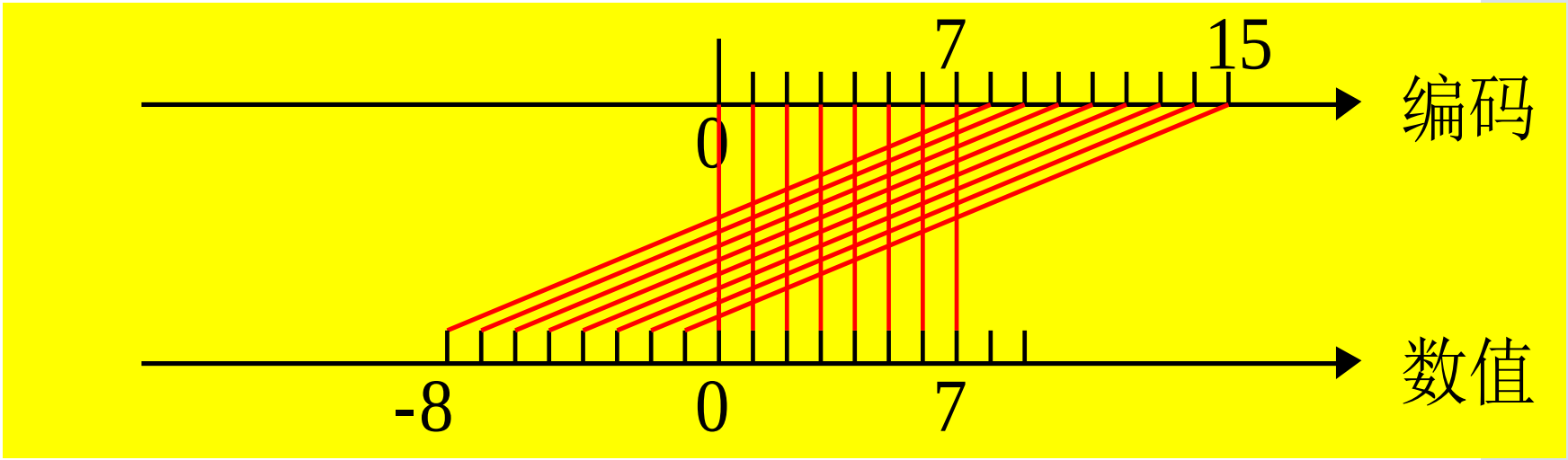
权： $2^{-3} \quad 2^2 \quad 2^1 \quad 2^0$

真值=加权求和

机器数	补码真值	原码真值
0000	0	0
0001	+1	+1
0010	+2	+2
0011	+3	+3
0100	+4	+4
0101	+5	+5
0110	+6	+6
0111	+7	+7
1000	-8	-0
1001	-7	-1
1010	-6	-2
1011	-5	-3
1100	-4	-4
1101	-3	-5
1110	-2	-6
1111	-1	-7

二、补码表示法

- 4位补码机器数（整数）在数轴上的表示
  - 补码机器数编码与真值的对应



三、反码表示法

- **1、表示方法**：最高位为符号位，其他位为数值位。
  - 符号位：0—正数，1—负数。
  - 数值位：正数时，与绝对值相同；**负数时，为绝对值取反**。
- **对于定点整数：**
  - 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$[X]_{反}=1,\overline{X_1}\,\overline{X_2}\cdots\cdots\overline{X_n}$。
- **对于定点小数：**
  - 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$[X]_{反}=1.\overline{X_1}\,\overline{X_2}\cdots\cdots\overline{X_n}$

三、反码表示法

- 例1：X=1011，Y＝－1011，则：

  $[X]_{反}$＝ 0,1011 ；$[Y]_{反}$＝ 1,0100 ；

- 例2：X=0.1101，Y＝- 0.1101，则：

  $[X]_{反}$＝ 0.1101 ；$[Y]_{反}$＝ 1.0010 ；

- 例3： X=1011， Y＝- 0.1101，求X和Y的8位反码机器数。

  $[X]_{反}$＝ 0,0001011 ；$[Y]_{反}$＝ 1. 0010111 ；

- 例4：$[0]_{反}$＝?

三、反码表示法

- 2、 0的表示：0 的反码表示有两种形式，即分别按照正数和负数表示。
  - $[+0]_{反}$= 00…0　　$[-0]_{反}$= 11…1
- 3、表示范围：对于n＋1位反码机器数X，它所能表示的数据范围为：
  - 定点整数：$-(2^n-1) \leqslant X \leqslant 2^n-1$
  - 定点小数：$-(1-2^{-n}) \leqslant X \leqslant 1-2^{-n}$

包括1位符号位，n位数值位

•思考：16位定点整数的反码表示范围？

三、反码表示法

❖4、数学表示：

定点整数

$$[X]_{反}=\begin{cases}X & X\geq 0\\ 2^{n+1}-1+X & X\leq 0\end{cases}$$

定点小数

$$[X]_{反}=\begin{cases}X & X\geq 0\\ 2-2^{-n}+X & X\leq 0\end{cases}$$

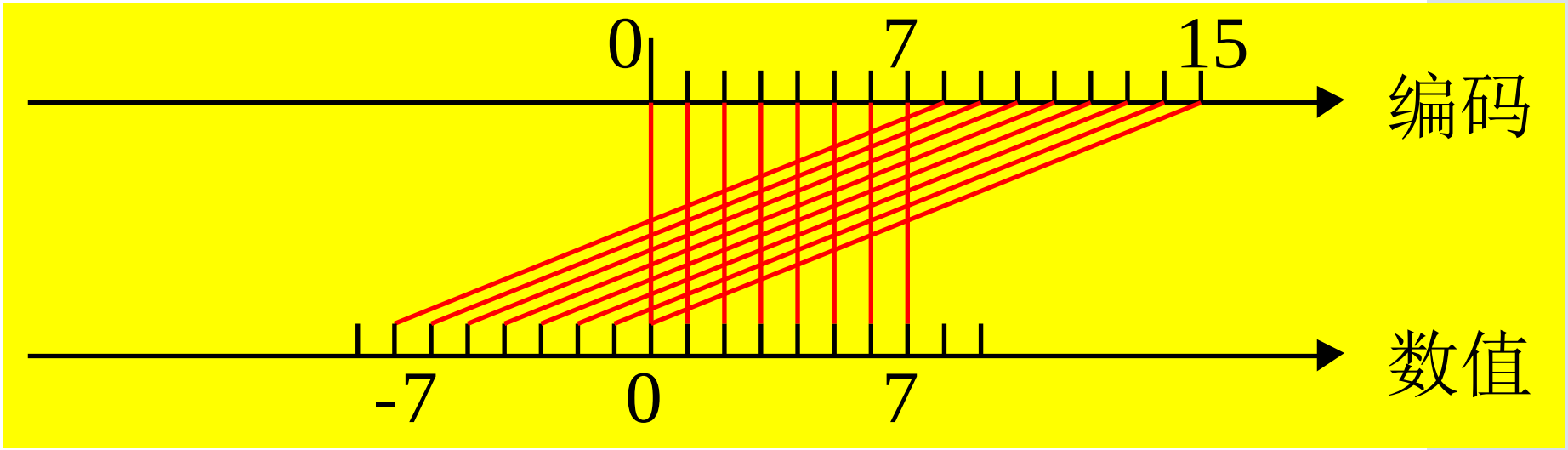
三、反码表示法

- 4位反码机器数（整数）对应的真值

机器数	原码真值	补码真值	反码真值
0000	0	0	0
0001	+1	+1	+1
0010	+2	+2	+2
0011	+3	+3	+3
0100	+4	+4	+4
0101	+5	+5	+5
0110	+6	+6	+6
0111	+7	+7	+7
1000	-0	-8	-7
1001	-1	-7	-6
1010	-2	-6	-5
1011	-3	-5	-4
1100	-4	-4	-3
1101	-5	-3	-2
1110	-6	-2	-1
1111	-7	-1	-0

三、反码表示法

- 4位反码机器数（整数）在数轴上的表示
  - 反码机器数编码与真值的对应



四、移码表示法

- **1、表示方法：最高位为符号位，其他位为数值位。**
  - 符号位：1—正数，0—负数。
  - 数值位：正数时，与绝对值相同；负数时，为绝对值取反后，末位加1。

移码表示：即为补码的符号位取反

- **对于定点整数：**
  - 若$X=+X_1X_2\cdots\cdots X_n$，则$[X]_{移}=1,X_1X_2\cdots\cdots X_n$；
  - 若$X=-X_1X_2\cdots\cdots X_n$，则$[X]_{移}=0,\overline{X_1}\,\overline{X_2}\cdots\cdots\overline{X_n}+1$。

四、移码表示法

- 例1：X=1011，Y＝－1011，则：

  $[X]_{移}$＝ 1,1011 ；$[Y]_{移}$＝ 0,0101 ；

- 例2：X=0.1101，Y＝- 0.1101，则：

  $[X]_{移}$＝ 1.1101 ；$[Y]_{移}$＝ 0.0011 ；

- 例3： X=1011， Y＝- 0.1101，求X和Y的8位移码机器数。

  $[X]_{移}$＝ 1,0001011 ；$[Y]_{移}$＝ 0. 0011000 ；

- 例4：$[0]_{移}$＝？

四、移码表示法

❖2、 0的表示：0 的移码表示形式是唯一的，即分别按照正数和负数表示均一致。

- $[+0]_{移}$= 10…0　　$[-0]_{移}$= 10…0

❖3、表示范围：对于n＋1位移码机器数X，它所能表示的数据范围为：

- 定点整数：$-2^n \leqslant X \leqslant 2^n-1$

包括1位符号位，n位数值位

❖移码通常作为浮点数的阶码。

•思考：16位定点整数的移码表示范围？

四、移码表示法

❖4、数学表示：

定点整数

$$[X]_{移} = 2^n + X$$

❖又称增码

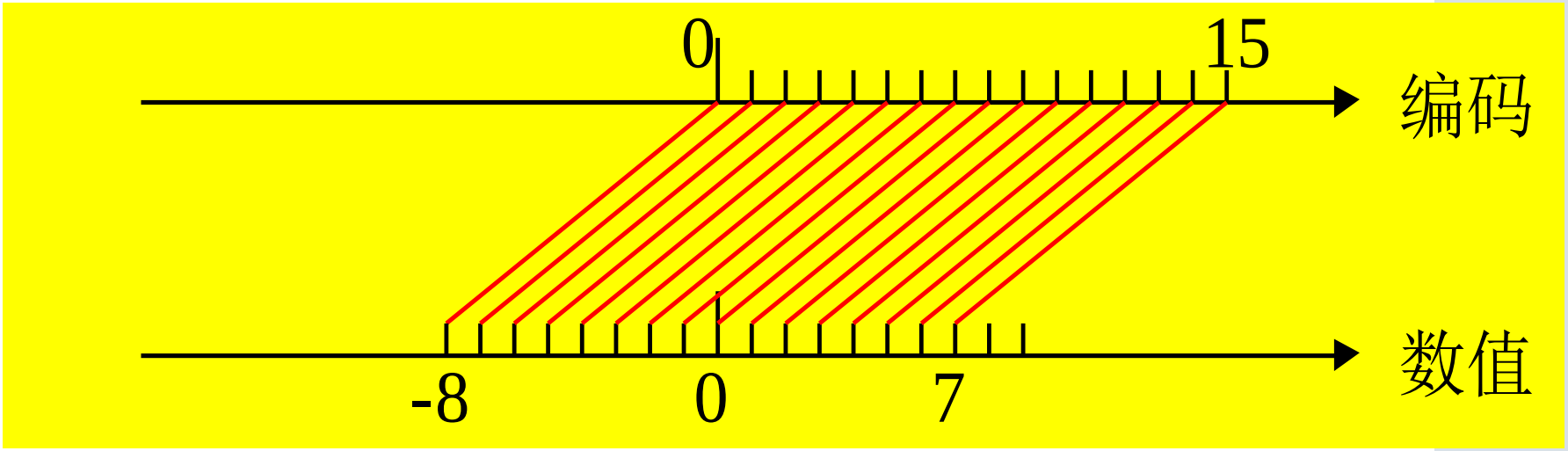
四、移码表示法

- 4位移码机器数（整数）对应的真值

机器数	原码真值	补码真值	反码真值	移码真值
0000	0	0	0	-8
0001	+1	+1	+1	-7
0010	+2	+2	+2	-6
0011	+3	+3	+3	-5
0100	+4	+4	+4	-4
0101	+5	+5	+5	-3
0110	+6	+6	+6	-2
0111	+7	+7	+7	-1
1000	-0	-8	-7	0
1001	-1	-7	-6	+1
1010	-2	-6	-5	+2
1011	-3	-5	-4	+3
1100	-4	-4	-3	+4
1101	-5	-3	-2	+5
1110	-6	-2	-1	+6
1111	-7	-1	-0	+7

四、移码表示法

- 4位移码机器数（整数）在数轴上的表示
  - 移码机器数编码与真值的对应



四种定点机器数的表示

真值	$X=+\ X_1X_2\cdots\cdots X_n$	$X=-\ X_1X_2\cdots\cdots X_n$
原码	$[X]_{原}=\ 0\ X_1X_2\cdots\cdots X_n$	$[X]_{原}=\ 1\ X_1X_2\cdots\cdots X_n$
反码	$[X]_{反}=\ 0\ X_1X_2\cdots\cdots X_n$	$[X]_{反}=\ 1\ \overline{X_1}\,\overline{X_2}\cdots\cdots\overline{X_n}$
补码	$[X]_{补}=\ 0\ X_1X_2\cdots\cdots X_n$	$[X]_{补}=\ 1\ \overline{X_1}\,\overline{X_2}\cdots\cdots\overline{X_n}+1$
移码	$[X]_{移}=\ 1\ X_1X_2\cdots\cdots X_n$	$[X]_{移}=\ 0\ \overline{X_1}\,\overline{X_2}\cdots\cdots\overline{X_n}+1$

四种定点机器数，0的表示

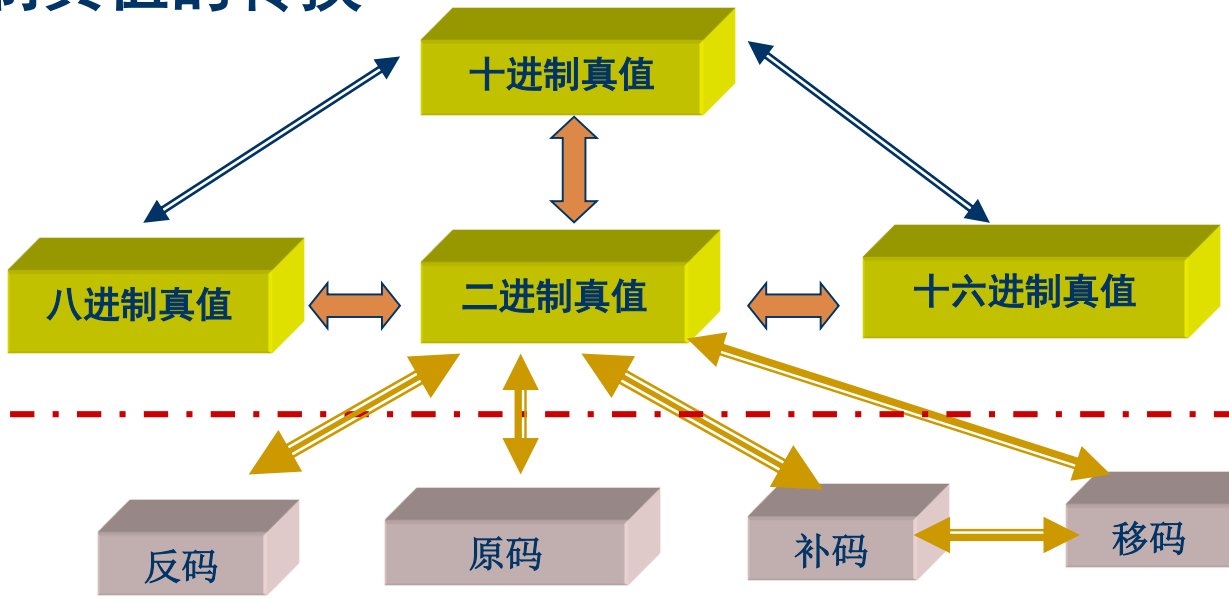
真值	X=+ 0	X=− 0
原码	$[X]_{原}$= 0 00……0	$[X]_{原}$= 1 00……0
反码	$[X]_{反}$= 0 00……0	$[X]_{反}$= 1 11……1
补码	$[X]_{补}$= 0 00……0	
移码	$[X]_{移}$= 1 00……0	

四种定点机器数的表示范围（n+1位机器数）

	定点整数	定点小数
原码	$-(2^n-1) \leqslant X \leqslant 2^n-1$	$-(1-2^{-n}) \leqslant X \leqslant 1-2^{-n}$
反码	$-(2^n-1) \leqslant X \leqslant 2^n-1$	$-(1-2^{-n}) \leqslant X \leqslant 1-2^{-n}$
补码	$-2^n \leqslant X \leqslant 2^n-1$	$-1 \leqslant X \leqslant 1-2^{-n}$
移码	$-2^n \leqslant X \leqslant 2^n-1$	$-1 \leqslant X \leqslant 1-2^{-n}$

五、定点机器数转换

不同进制真值的转换



机器码的转换关系

五、定点机器数转换

- 机器数转换为真值
  - ①机器数的符号位→真值的正负
  - ②机器数的定义和表示→ 真值的绝对值
- 机器数之间的相互转换
  - 最简单的方法：先求出它们的真值，然后再转换为另一种表示方法。

课堂练习

- 1、已知$[X]_{补}$=1.1010，求X= —0.0110

  $[X]_{原}$= 1.0110

  $[X]_{反}$= 1.1001

  $[X]_{移}$= 0.1010

- 2、求以下各机器数的十进制真值：

课堂练习

$[X]_{原}$ = 1,0000000 ， 则X = ?
$[X]_{补}$ = 1,0000000 ， 则X = ?
$[X]_{反}$ = 1,0000000 ， 则X = ?
$[X]_{移}$ = 1,0000000 ， 则X = ?
$[X]_{原}$ = 1,1101 ， 则X = ?
$[X]_{补}$ = 1,1101 ， 则X = ?
$[X]_{反}$ = 1,1101 ， 则X = ?
$[X]_{移}$ = 1,1101 ， 则X = ?
$[X]_{原}$ = 0,1000 ， 则X = ?
$[X]_{补}$ = 1,1000 ， 则X = ?
$[X]_{反}$ = 0,1000 ， 则X = ?
$[X]_{移}$ =0,1000 ， 则X = ?

$-(1111111 + 1)_2$

X = - 0
X = （- 128）$_{10}$
X = （-127）$_{10}$
$-(1111111)_2$
X = 0
X = - 1101B
X = - 0011B
X = - 0010 B
X = +1101B
X = +1000 B
X = - 1000 B
X = +1000B
X = -1000 B

3.4 浮点机器数的表示方法

一 浮点机器数的格式

二 浮点机器数的规格化表示

三 浮点数的表示范围

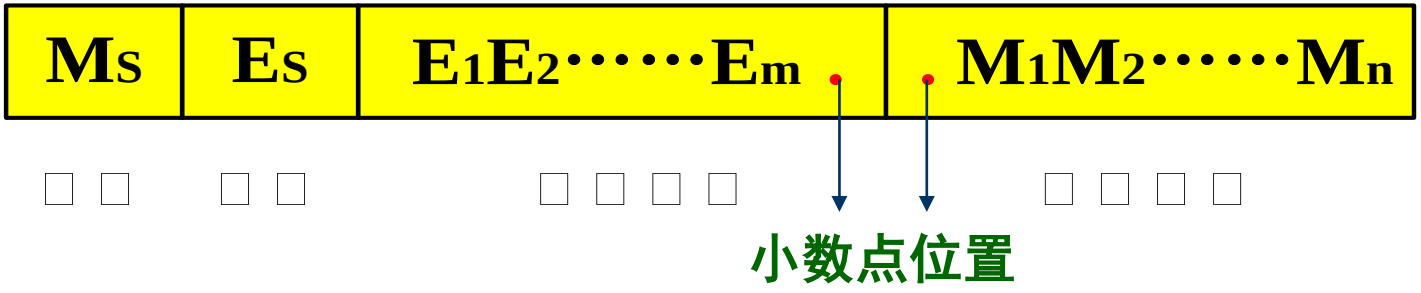
四 浮点数与定点数比较

一、浮点机器数的格式

- 浮点机器数用于表示实数，其小数点的位置由其中的阶码规定，因此是浮动的。
- 浮点数N的构成：

$$N = M \times R^{E}$$

- 浮点数的格式：阶码的底是隐含规定的。



  - 在机器中，为了方便浮点数大小的比较，通常将数符放置在浮点数的首位。

一、浮点机器数的格式

- 尾数M：为定点小数，尾数的位数决定了浮点数有效数值的精度，尾数的符号代表了浮点数的正负，因此又称为数符。尾数一般采用原码和补码表示。
- 阶码E：为定点整数，阶码的数值大小决定了该浮点数实际小数点位置与尾数的小数点位置（隐含）之间的偏移量。阶码的位数多少决定了浮点数的表示范围。阶码的符号叫阶符。阶码一般采用移码和补码表示。
- 阶码的底R：一般为2、8或16 ，且隐含规定。

IEEE 754 浮点数标准

- 根据IEEE 754 国际标准，常用的浮点数格式有3种，阶码的底隐含为2。
- 短实数又称为单精度浮点数，长实数又称为双精度浮点数，临时实数主要用于进行浮点数运算时保存临时的计算结果。格式：

M_S	E		M
数符	阶符	阶码	尾数
0：正数 1：负数	2^n-1的移码		原码表示； 单、双精度的整数位“1”隐藏； 临时实数无隐藏位

IEEE 754 浮点数标准

- 位数：

隐藏位,位于整数位

类型	总位数	尾数位数（含1位数符）	阶码位数（含1位阶符）	真值计算
短实数	32	24	8	$N=(-1)^{MS}\times(1.M_1\ M_2\ldots\ldots\ M_n)\times 2^{E-127}$
长实数	64	53	11	$N=(-1)^{MS}\times(1.\ M_1\ M_2\ldots\ldots\ M_n)\times 2^{E-1023}$
临时实数	80	65	15	

隐藏位,位于整数位

二、浮点机器数的规格化表示

1. **浮点数的规格化表示：**为了充分利用尾数的二进制数位来表示更多的有效数字，将尾数的绝对值限定在某个范围之内。

- 例如：R＝2，则规格化浮点数的**尾数M应满足条件：绝对值最高有效位为1**，即

$$\frac{1}{2} \le |M| \le 1$$

没有前导零

二、浮点机器数的规格化表示

2. 计算机硬件对浮点数规格化判断方法：

- 原码表示的尾数：
  - M1=1：规格化，即尾数为×.1×……×形式；
- 补码表示的尾数：
  - MS ⊕ M1=1：规格化，即尾数为0.1×……×形式或者为1.0×……×形式。

❖ 对于非规格化浮点数，可以通过修改阶码和左右移尾数的方法来使其变为规格化浮点数，这个过程叫做规格化。

二、浮点机器数的规格化表示

- 尾数进行右移实现的规格化，则称为**右规**；
- 尾数进行左移实现的规格化，则称为**左规**。

- 使用规格化的浮点数表示数据的优点：
  - 提高了浮点数据的精度；
  - 使程序能够更方便地交换浮点数据；
  - 可以使浮点数的运算更为简化。

二、浮点机器数的规格化表示

- 例：一浮点数的阶码为6位（包括一位阶符），尾数为10位（包括一位数符），阶码与尾数均采用补码表示，阶码的底为2，**数符在浮点数的最高位**。写出X与Y的规格化浮点数。
  - （1）X＝－123.25
  - （2）Y＝17/64
- （1）$X=(-123.25)_{10}$

  $=(-1111011.01)_{2}$

  $=-0.111101101\times2^{+7}$

二、浮点机器数的规格化表示

- E_X=+7=(+00111)$_2$，M_X=－0. 111101101
- $[E_X]_{补}$=000111，　　$[M_X]_{补}$=1. 000010011
- 则：$[X]_{浮}$= 1 000111 000010011

❖ （2）Y＝（17/64）$_{10}$

＝（0.010001）$_2$

＝0. 10001×2^{-1}

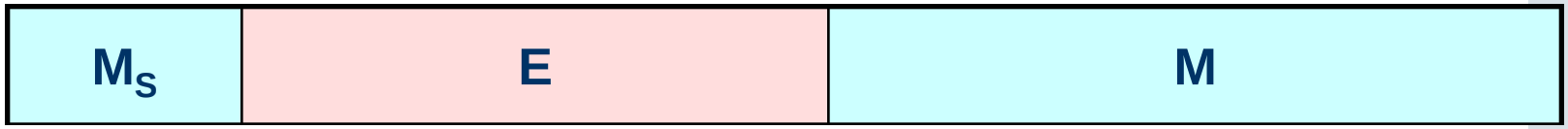
- E_Y=－00001，M_Y=0. 100010000
- $[E_Y]_{补}$=111111，　$[M_Y]_{补}$=0. 100010000
- 则：$[Y]_{浮}$= 0 111111 100010000

总结：

- 规格化浮点数表示方法：
- （1）写出数据的二进制真值。
- （2）转换为$M\times 2^{E}$的形式，其中M为没有前导零的定点小数，E为整数。
- （3）按照格式写出M和E的规定机器数编码。
- （4）按照格式要求排列E和M。

IEEE754浮点数表示方法

- 例：X=17/64，写出X的IEEE754**单精度**浮点数。
- 单精度浮点数格式：



- X=（0.010001）$_2$
- X =（1. 0001）$_2 \times 2^{-2}$
- 正数，Ms=0
- M=0001000 00000000 0000000
- 【E】$_{移}$=127-2=125，【E】$_{移}$=0111 1101
- 【X】$_{浮}$= 0 0111 1101 0001000 00000000 0000000

整数位1，隐藏

尾数数值位23位原码

阶码8位，$2^n-1=127$的移码

IEEE754浮点数表示方法

- 例：Y=00000000H，是[illegible]单精[illegible]数，求Y的十进制真值。
- Y=00000000H = 0000 0000 0000 0000 0000 0000 0000 0000 B
- Ms=0，表明正数
- 【E】移= 000 0000 0 B = 0 =127+E
- 则：阶码的真值E=0-127=-127
- M=1.000 0000 0000 0000 0000 0000B=1.0
- Y =+1.0×2^{-127}
- Y = 2^{-127}

数符，+

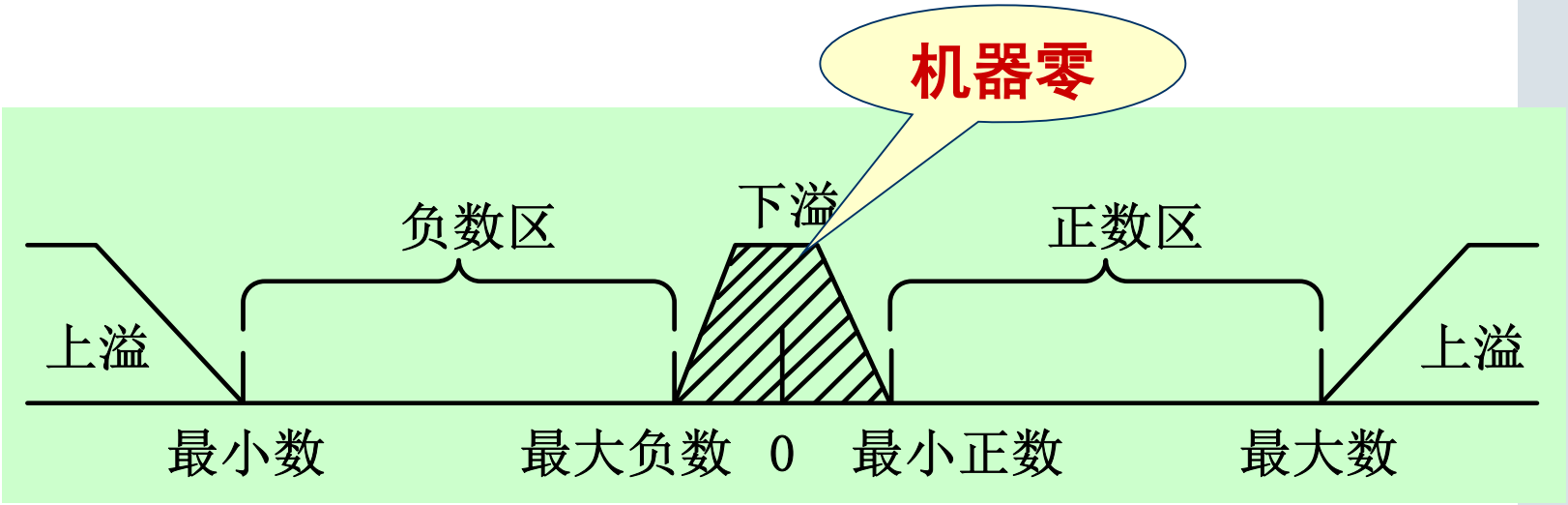
阶码8位，2^n-1的移码

尾数数值位23位原码

添加隐藏位1

三、浮点数的表示范围

- 浮点数的表示范围通常由4个点界定：最小（负）数、最大负数、最小正数、最大（正）数。
- **范围**：最小负数～最大负数，最小正数～最大正数



三、浮点数的表示范围

- **下溢**：位于最大负数和最小正数之间的数据（除0外），机器无法表示。
  - 处理：计算机直接将其视为机器零。

正溢出

- **上溢**：当一个数据大于最大（正）数，或者小于最小（负）数时，机器也无法表示，称为上溢，上溢又称溢出。
  - 处理：计算机置溢出标志位，或者报警。

负溢出

- **机器零**：有两种情况
  - （1）若浮点数的尾数为零，无论阶码为何值；
  - （2）当阶码的值遇到比它能表示的最小值还要小时（阶码负溢出），无论其尾数为何值

三、浮点数的表示范围

$$N = M \times R^{E}$$

浮点数	尾数M	阶码E
最小数	最小数	最大数
最大负数	最大负数	最小数
最小正数	最小正数	最小数
最大数	最大数	最大数

三、浮点数的表示范围

- 一浮点数的阶码为6位，尾数为10位，阶码与尾数均采用补码表示，阶码的底为2，写出浮点数格式的规格化和非规格化表示范围。
- 解：（1）规格化表示范围：尾数必须规格化

	真值	浮点数表示
最小数	$-1\times2^{31} = -2^{31}$	1，11111　1.000000000
最大负数	$-(2^{-1}+2^{-9})\times2^{-32}$	0，00000　1.011111111
最小正数	$2^{-1}\times2^{-32} = 2^{-33}$	0，00000　0.100000000
最大数	$(1-2^{-9})\times2^{31}$	1，11111　0.111111111

三、浮点数的表示范围

- （2）非规格化表示范围：

	真值	浮点数表示
最小数	$-1\times2^{31} = -2^{31}$	1，11111　1.000000000
最大负数	$-2^{-9}\times2^{-32} = -2^{-41}$	0，00000　1.111111111
最小正数	$2^{-9}\times2^{-32} = 2^{-41}$	0，00000　0.000000001
最大数	$(1-2^{-9})\times2^{31}$	1，11111　0.111111111

四、定点数与浮点数比较

- 相同点：
  - 1、计算机所能表示的数据都是一系列离散的点
    - 问题：存在于两个点之间的数据？
    - 解决：计算机通常采用合适的舍入操作，选取最近似的值来替代。
  - 2、计算机硬件的字长是有限的
    - 问题：数据超出了机器数所能表示的最大界限？
    - 解决：计算机必须能够产生“溢出”的异常报告。

四、定点数与浮点数比较

- **不同点：**
  - 定点数可表示的点在数轴上是均匀的，距离是等长的1（定点整数）或者2^{-n}（定点小数）；
  - 浮点数则是分布不均匀、距离不相等的。

3.5 非数值数据的表示

- 非数值数据：文字和符号（字符）、图像、声音等
- 非数值数据的表示：对其进行二进制编码

一 字符编码

二 汉字编码

一、字符编码

1. **字符的表示：**采用字符编码，即用规定的二进制数表示文字和符号的方法。
2. **ASCII码：**美国标准信息交换码，为国际标准，在全世界通用 。

- 常用的**7位ASCII码**的每个字符都由7个二进制位b6～b0 表示，有128个编码，最多可表示128种字符；其中包括：
  - 10个数字 ‘0’～ ‘9’：30H～39H，顺序排列
  - 26个小写字母 ‘a’～ ‘z’：61H～7AH ，顺序排列
  - 26个大写字母 ‘A’～ ‘Z’：41H～5AH ，顺序排列
  - 各种运算符号和标点符号等。

ASCII码分类

- 95个可打印或显示的字符：称为图形字符，可在打印机和显示器等输出设备上输出；可在计算机键盘上找到相应的键。
- 33个控制字符：不可打印或显示，分成5类：
  - ① 10个传输类控制字符：用于数据传输控制；
  - ② 6个格式类控制字符，用于控制数据的位置
  - ③ 4个设备类控制字符，用于控制辅助设备；
  - ④ 4个信息分隔类控制字符，用于分隔或限定数据
  - ⑤ 9个其他控制字符、空格字符和删除字符。

ASCII码编码表

	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	SO	RS	.	〉	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

基于IBM ProPrinter打印机的扩展ASCII码

$B_7B_6B_5B_4$ $B_3B_2B_1B_0$	0000	0001	1000	1001	1010	1011	1100	1101	1110	1111
0000		▸	Ç	É	á	▒	└	╨	α	≡
0001	☺	◂	ü	æ	í	▒	┴	╤	ß	±
0010	☻	↕	é	Æ	ó	▓	┬	╥	Γ	≥
0011	♥	‼	â	ô	ú	│	├	╙	π	≤
0100	♦	¶	ä	ö	ñ	┤	─	Ô	Σ	⌠
0101	♣	§	à	ò	Ñ	╡	┼	╒	σ	⌡
0110	♠	▬	å	û	ª	╢	╞	╓	µ	÷
0111	●	↨	ç	ù	º	╖	╟	╫	τ	≈
1000	◘	↑	ê	ÿ	¿	╕	╚	╪	Φ	≈
1001	○	↓	ë	Ö	⌐	╣	╔	┘	Θ	·
1010	■	→	è	Ü	¬	║	╩	┌	Ω	·
1011	♂	←	ï	¢	½	╗	╦	█	δ	√
1100	♀	∟	î	£	¼	╝	╠	▄	∞	ⁿ
1101	♪	↔	ì	¥	¡	╜	═	▌	φ	²
1110	♫	▲	Ä	₧	«	╛	╬	▐	ε	■
1111	☼	▼	Å	ƒ	»	┐	╧	▀	∩	

二、汉字编码

对于汉字，计算机的处理技术必须解决三个问题：汉字输入、汉字储存与交换、汉字输出，它们分别对应着**汉字输入码、交换码、内码、字形码**的概念。

1. 汉字输入码

汉字输入码也称**外码**，是为了将汉字输入计算机而编制的代码，**是代表某一汉字的一串键盘符号。**

- 汉字输入码种类：
  - **数字编码**：如区位码、国标码、电报码等。
  - **拼音编码**：如全拼码、双拼码、简拼码等。
  - **字形编码**：如王码五笔、郑码、大众码等。
  - **音形编码**：如表形码、钱码、智能ABC等。

二、汉字编码

2、汉字交换码：指不同的具有汉字处理功能的计算机系统之间在交换汉字信息时所使用的代码标准。

- 目前国内标准信息处理交换码：基于1980年制定的国家标准《信息交换用汉字编码字符集·基本集》（GB2312-80）修订的国标码。
- 共收录了6763个汉字和682个图形符号。6763个汉字分为一级常用汉字3755个，二级次常用汉字3008个。其中一级汉字按拼音字母顺序排列，二级汉字按偏旁部首排列。
- 采用两个字节对每个汉字进行编码，每个字节各取七位，可对128×128＝16384个字符进行编码。

二、汉字编码

两种典型的数字编码作为交换码：

① **区位码**：是将国家标准局公布的6763个两级汉字分为94个区，每个区分94位，实际上把汉字表示成二维数组，每个汉字在数组中的下标就是区位码。例如“中”字位于54区48位，“中”字的区位码即为“5448”。

② **国标码**：将区位码加2020H，占用两个字节。例如“中”字的国标码为区位码5448的区码和位码转化为16进制，为3630H，再加2020H得国标码5650H。

二、汉字编码

3、汉字内码

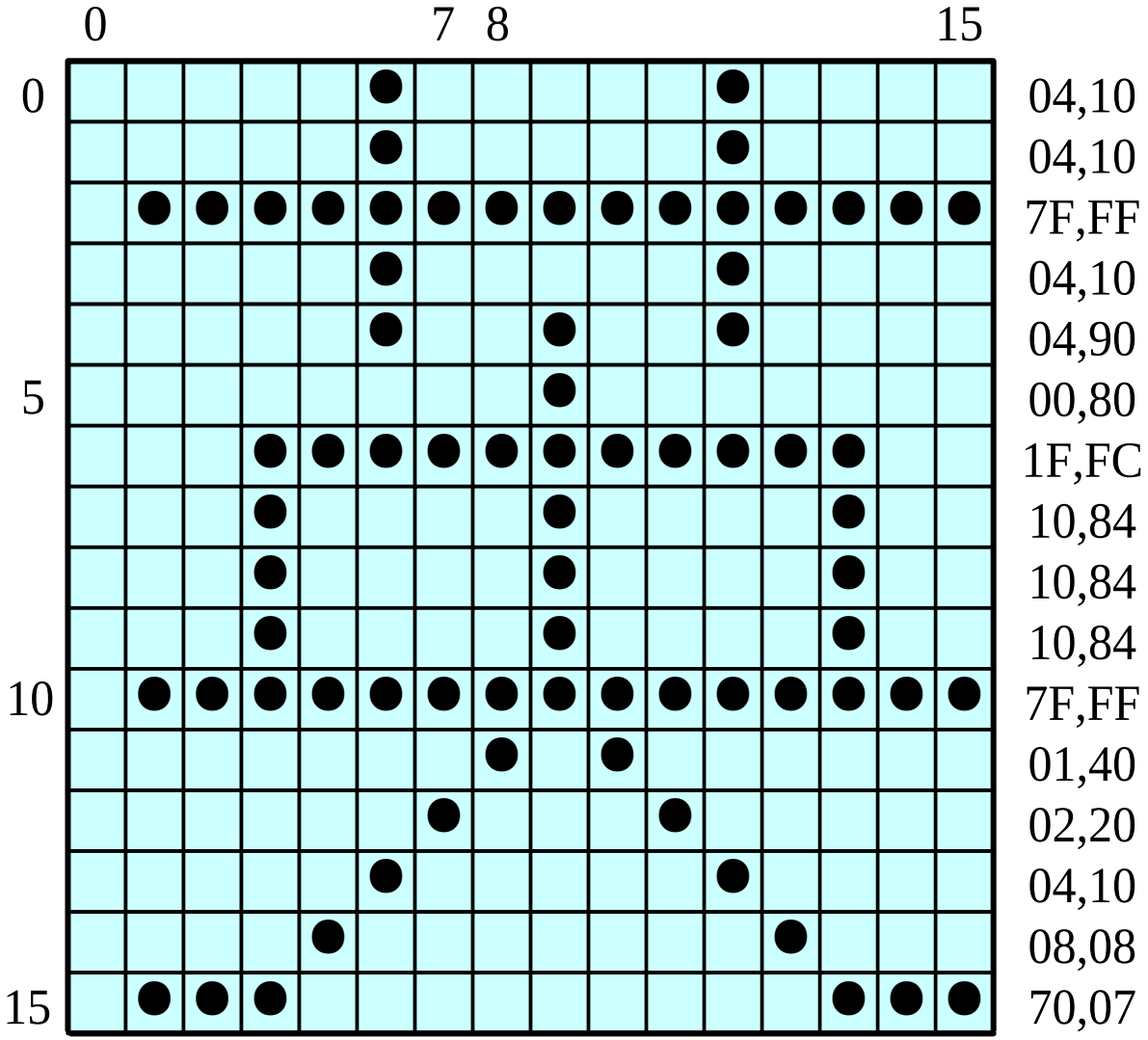
- 汉字内码是用于汉字信息的存储、交换、检索等操作的机内代码，一般采用两个字节表示。
- 汉字可以通过不同的输入法输入，但其内码在计算机中是唯一的。
- 英文字符：七位的ASCII码，字节的最高位为“0”。
- 汉字机内代码：2个字节的最高位均为“1”。
- 汉字机内码=汉字国标码+8080H。例如“中”字的机内码为D6D0H。
- 文本文件中储存的是汉字内码。

二、汉字编码

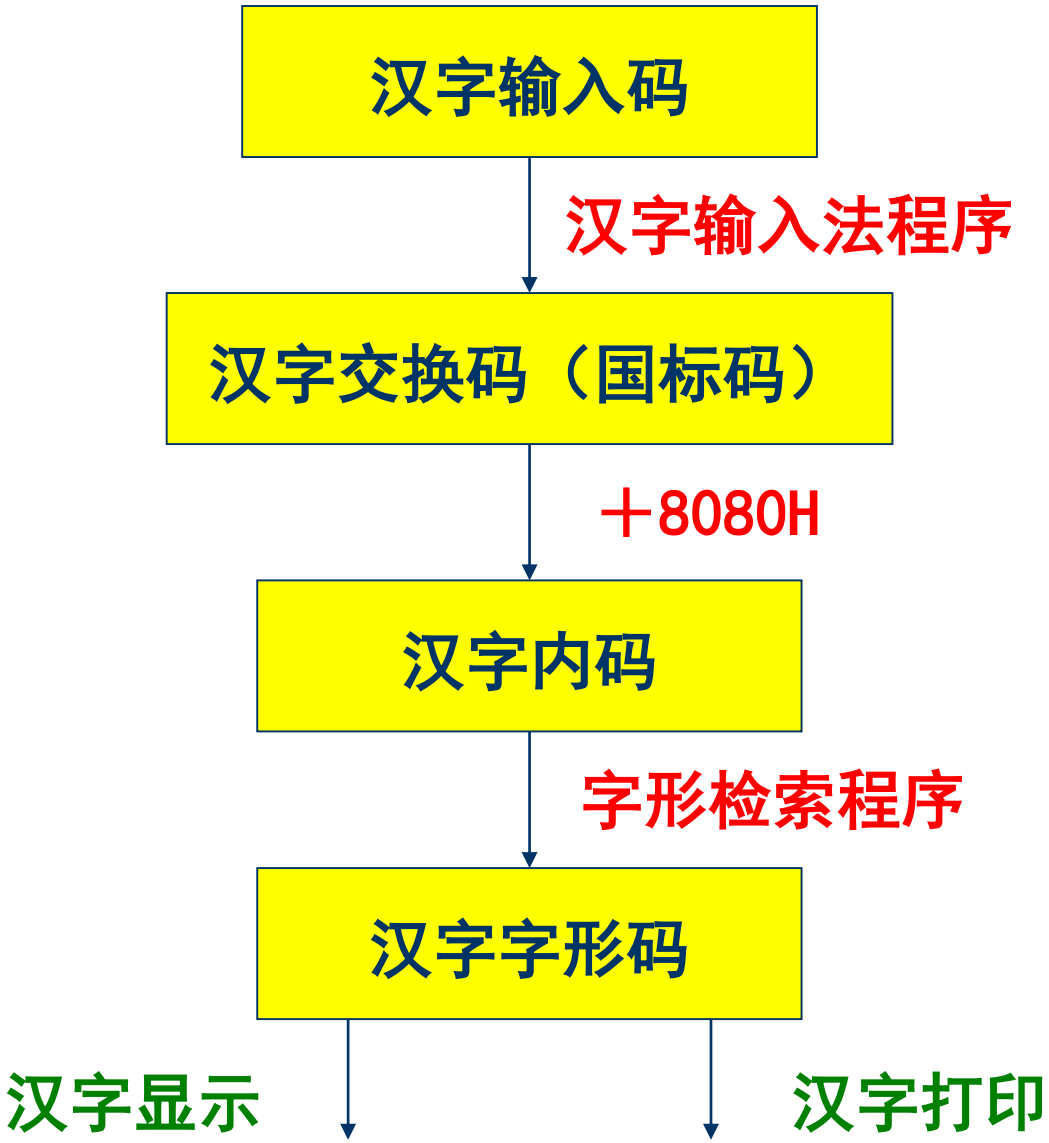
4、汉字字形码

- 汉字字形码是将汉字字形经过点阵数字化后形成的一串二进制数，用于汉字的显示和打印。
- 根据汉字输出的要求不同，点阵有以下几种：
  - 简易型汉字：16×16，　32字节/汉字
  - 普通型汉字：24×24，　72字节/汉字
  - 提高型汉字：32×32，128字节/汉字。
- 汉字字库：将所有汉字的字模点阵代码按内码顺序集中起来，构成了汉字库。

字模码：16×16点阵，需要16×16b=32B/汉字



汉字编码之间的关系



Unicode

- Unicode是国际组织制定的可以容纳世界上所有文字和符号的字符编码方案，又称统一码、万国码、单一码，是一种在可在计算机上使用的字符编码。
- 它为每种语言中的每个字符设定了统一并且唯一的二进制编码，以满足跨语言、跨平台进行文本转换、处理的要求。
- 目前普遍采用的是UCS-2，即Unicode 16，它用2个字节来编码一个字符；包含了GB18030里面的所有汉字（27484个字）。 “中”的Unicode 16编码是4E2DH。
- UCS-4（Unicode 32bit）：预备纳入康熙字典的所有汉字。

3.6 校验码

一 校验码概述

二 奇偶校验码

三 海明校验码

四 CRC校验码

一、校验码概述

1、校验码定义：是一种具有发现某些错误或自动改正错误能力的一种数据编码方法。

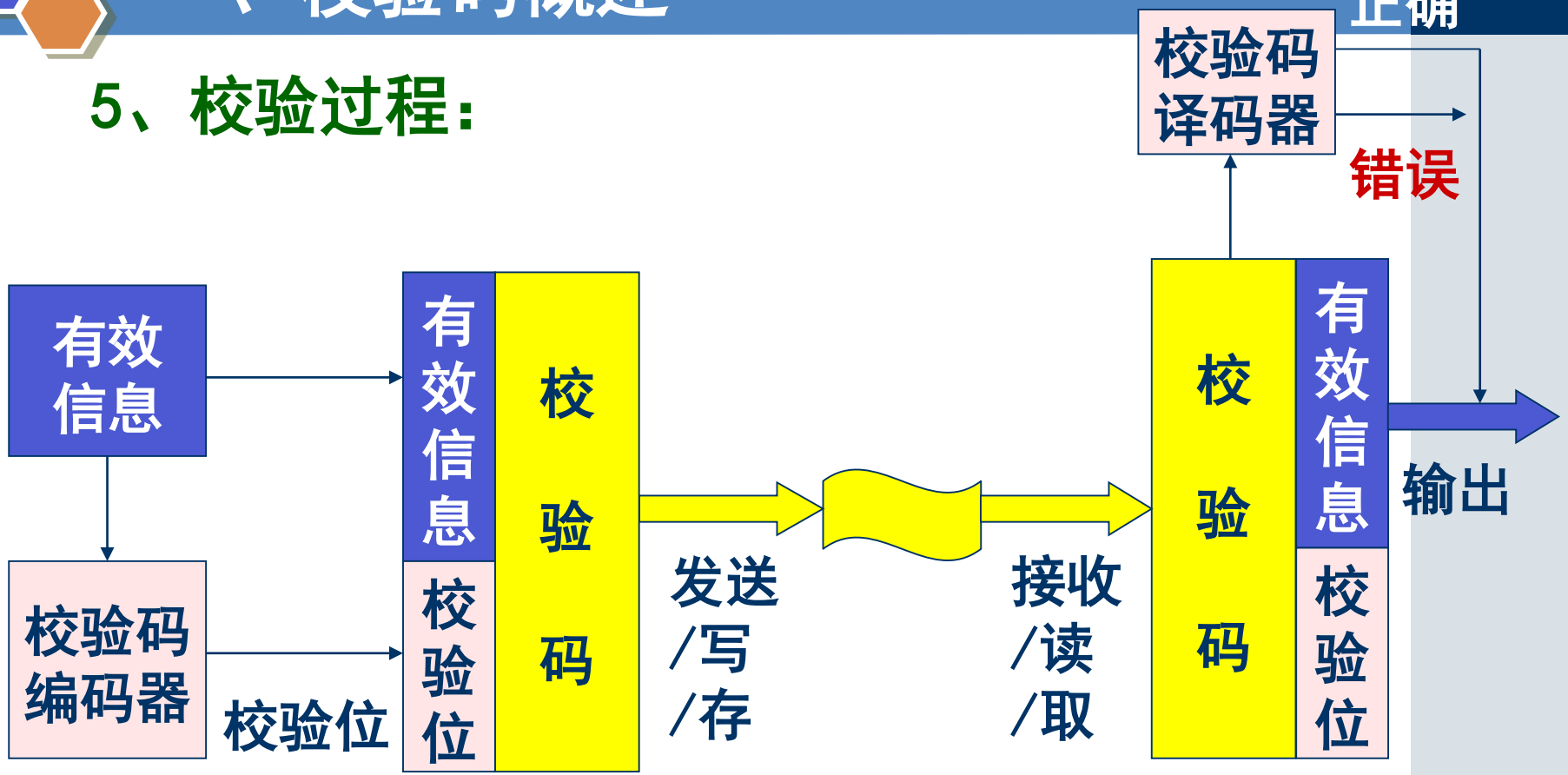
2、校验码目的：用于检查或纠正在存取、读写和传送数据的过程中可能出现的错误。

3、校验码的基本思想：“冗余校验”，即通过在有效信息代码的基础上，添加一些冗余位来构成整个校验码。

4、校验码的构成：有效信息＋校验位（由有效信息产生的冗余位）

一、校验码概述

5、校验过程：



一、校验码概述

6、**校验码原理**：通过判断代码的**合法性**来检错的。

- 只有当合法码之间的**码距d≥2**时，校验码才具有**检错能力**，当码距d≥3时，校验码才具有**纠错能力**。
- **码距**：一种码制的码距是指该码制中所有代码之间的最小**距离**。
- **两个代码之间的距离**：在一种编码中，在任何两个代码之间逐位比较，对应位值不同的个数。

一、校验码概述

6、**校验码原理**：通过判断代码的**合法性**来检错的。

- 校验码的**检错纠错能力与码距的关系**如下：

① 若码距d为奇数，**如果**只用来检查错误，则可以发现d－1位错误；**如果**用来纠正错误，则能够纠正（d-1）/2位错误。

② 若码距d为偶数，则可以发现 d/2 位错误，**并**能够纠正（d/2 -1）位错误。

一、校验码概述

7、常见校验码：

- 奇偶校验码：码距d=2，检错码，能检验奇数位错误；通常用于磁带或者串行通信中。
- 海明校验码：码距d>=3，纠错码，能纠正1位或多位错误；通常用于磁盘冗余阵列中。
- CRC校验码：码距d=3，纠错码，能纠正1位错误；通常用于磁盘或数据块的校验。

二、奇偶校验码

- 在有效信息位的前面或者后面添加一位奇（偶）校验位就组成了奇（偶）校验码。
- 奇（偶）校验码的编码和译码在硬件上通常采用异或非门（异或门）实现。

1、编码

- 奇校验位的取值应该使整个奇校验码中“1”的个数为奇数，偶校验位的取值应该使整个偶校验码中“1”的个数为偶数。

二、奇偶校验码

1、编码

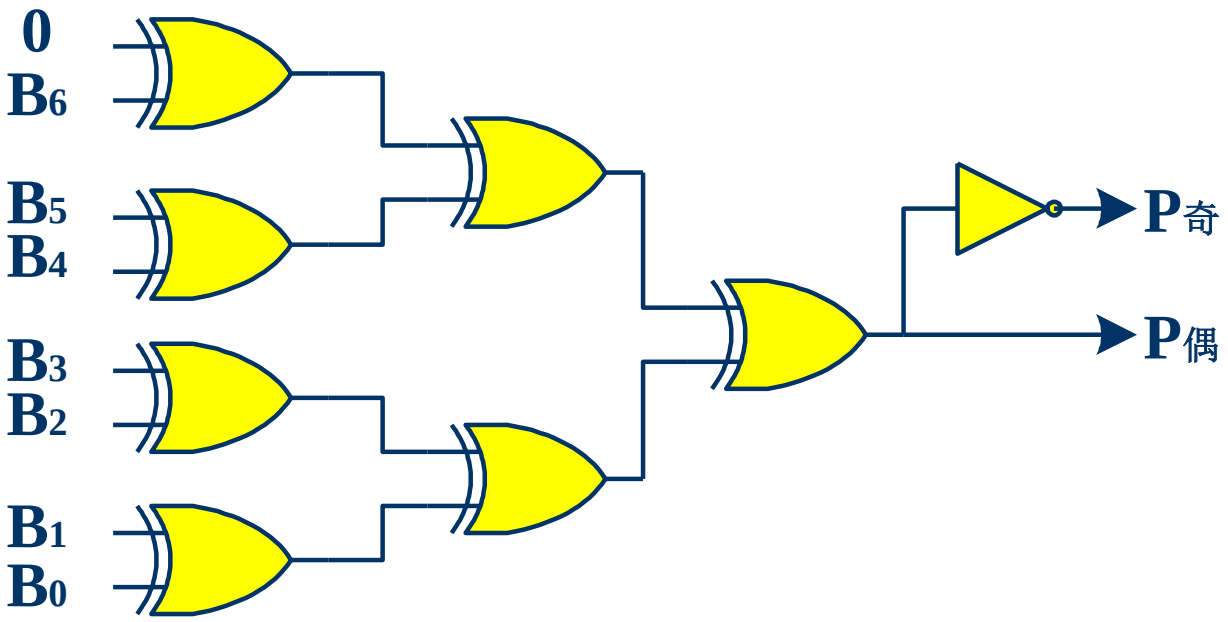
- 假设在发送端，要发送七位ASCII码（B6 B5 B4 B3 B2 B1 B0），在ASCII码前面添加一位奇校验位$P_{奇}$或偶校验位$P_{偶}$变为一个字节的奇偶校验码，则它们的生成表达式为

$$P_{奇} = \overline{B_6 \oplus B_5 \oplus B_4 \oplus B_3 \oplus B_2 \oplus B_1 \oplus B_0}$$

$$P_{偶} = B_6 \oplus B_5 \oplus B_4 \oplus B_3 \oplus B_2 \oplus B_1 \oplus B_0$$

二、奇偶校验码

奇偶校验的编码电路



- 例如：字符“A”的ASCII码为41H = 100 0001B
- 奇校验码为C1H = 1100 0001
- 偶校验码为41H = 0100 0001

二、奇偶校验码

2、译码

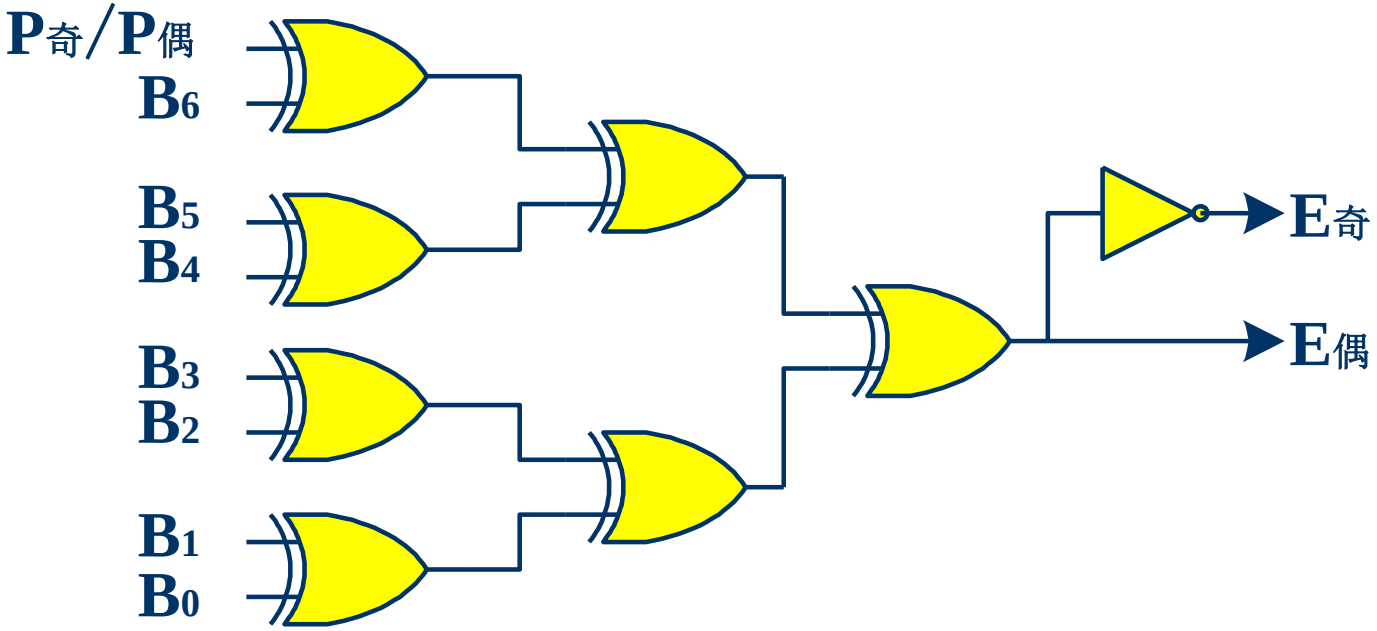
- 接收端检验接收到的校验码信息的奇偶性，奇校验码中“1”的个数应该为奇数；偶校验码中“1”的个数应该为偶数;否则出错。
- 设$E_{奇}$为奇校验码出错信号，$E_{偶}$为偶校验码出错信号，为1出错，为0正确，则它们的表达式为

$$E_{奇}=\overline{B_6\oplus B_5\oplus B_4\oplus B_3\oplus B_2\oplus B_1\oplus B_0\oplus P_{奇}}$$

$$E_{偶}=B_6\oplus B_5\oplus B_4\oplus B_3\oplus B_2\oplus B_1\oplus B_0\oplus P_{偶}$$

二、奇偶校验码

奇偶校验的译码电路



三、海明码

- 1、编码：步骤如下：
  - （1）计算校验位的位数
  - （2）确定有效信息和校验位的位置
  - （3）分组
  - （4）进行奇偶校验，合成海明码
- 2、译码

（1）计算校验位的位数

- 假设信息位为k位，增加r位校验位，构成n=k+r位海明码字。若要求海明码能纠正一位错误，用r位校验位产生的r位指误字来区分无错状态及码字中n个不同位置的一位错误状态，则要求r满足：
- $2^r \geqslant k + r + 1$
- 计算出k位有效信息时，必须添加的能纠错一位的海明校验码的校验位的位数r

k值	r最小值
1～4	3
5～11	4
12～26	5
27～57	6
58～120	7

（2）确定有效信息和校验位的位置

- 假设k位有效信息从高到低为$D_k\ D_{k-1}\ \ldots D_2\ D_1$
- 添加的r位校验位为$P_r\ P_{r-1}\ \ldots P_2\ P_1$
- 它们构成n=k+r位的海明码为$H_n\ H_{n-1}\ \ldots H_2\ H_1$
- H的下标被称为海明位号，则第i位的校验位P_i必须位于位号为2^{i-1}的位置，即：
- $P_i=H_j$，$j=2^{i-1}$，其中，i=1，2，…，r；
- 有效信息则在其余的海明码位置上顺序排列。
- 例如：k=8，则r=？其海明码的排列？

H_{12}	H_{11}	H_{10}	H_9	H_8	H_7	H_6	H_5	H_4	H_3	H_2	H_1
D_8	D_7	D_6	D_5	P_4	D_4	D_3	D_2	P_3	D_1	P_2	P_1

（3）分组

- 海明码是分组进行奇偶校验的，每一组通过一个监督表达式来监督有效信息的变化
- 分组必须使得监督表达式得出的指误字能够反映出错位的位号。
- **分组的原则**：
  - 校验位只参加一组奇偶校验，有效信息则参加至少两组的奇偶校验。
  - 若 $D_i = H_j$ ，则D_i参加那些位号之和等于j的校验位的分组校验。

k=8，r=4的海明码分组

序号	H_{12}	H_{11}	H_{10}	H_9	H_8	H_7	H_6	H_5	H_4	H_3	H_2	H_1
分组	D_8	D_7	D_6	D_5	P_4	D_4	D_3	D_2	P_3	D_1	P_2	P_1
P_4	√	√	√	√	√							
P_3	√					√	√	√	√			
P_2		√	√			√	√			√	√	
P_1		√		√		√		√		√		√

（4）进行奇偶校验，合成海明码

① 按照分组和奇偶校验的规律将每个校验位的生成表达式写出

② 带入有效信息的值，依次得出校验位的取值

③ 将校验位按各自的位置插入，与有效信息一起合成海明码。

例如：有效信息为11101001，则可以纠错一位的海明码为1 1 1 0 1 1 0 0 0 1 0 1

$$P_4 = D_8 \oplus D_7 \oplus D_6 \oplus D_5$$

$$P_3 = D_8 \oplus D_4 \oplus D_3 \oplus D_2$$

$$P_2 = D_7 \oplus D_6 \oplus D_4 \oplus D_3 \oplus D_1$$

$$P_1 = D_7 \oplus D_5 \oplus D_4 \oplus D_2 \oplus D_1$$

2、译码

- 在接收端收到每个海明码后，也必须按上述分组检验每组的奇偶性有无发生变化；方法：
  - 按照监督关系式算出指误字S_r S_{r-1} …S_2 S_1
  - 若为全零，则说明各组奇偶性全部无变化，信息正确，将相应的有效信息位析取出来使用；
  - 若不为零，则指误字的十进制值，就是出错位的海明位号。

2、译码

- **例如：**收到的海明码为110011000101，得到S_4 S_3 S_2 S_1=（1010）$_2$=（10）$_{10}$ ，则表明是H_{10}（D_6）出错，将H_{10}取反，得正确海明码为111011000101。

$$S_4 = P_4 \oplus D_8 \oplus D_7 \oplus D_6 \oplus D_5$$
$$S_3 = P_3 \oplus D_8 \oplus D_4 \oplus D_3 \oplus D_2$$
$$S_2 = P_2 \oplus D_7 \oplus D_6 \oplus D_4 \oplus D_3 \oplus D_1$$
$$S_1 = P_1 \oplus D_7 \oplus D_5 \oplus D_4 \oplus D_2 \oplus D_1$$

四、循环冗余码CRC

- 循环冗余码CRC（Cyclic Redundancy Code），又称为多项式码。

1、编码

- k位要发送的有效信息位可对应于一个k-1次多项式M（x），r位冗余校验位对应于一个r-1次多项式R（x）。由k位信息位后面加上r位冗余位组成的n=k+r位CRC码字则对应于一个n-1次多项式C（x），即：

$$C(x) = x^r \cdot M(x) + R(x)$$

该CRC码称为（n，k）循环码。

四、循环冗余码CRC

- 由信息位产生冗余位的编码过程，就是已知M（x）求R（x）的过程。在CRC码中可以通过找到一个特定的多项式G（x）来实现。用G（x）去除x^r • M（x）得到的余式就是R（x），假设商的多项式为Q（x），编码过程：

$$\frac{x^r \cdot M(x)}{G(x)} = Q(x) + \frac{R(x)}{G(x)}$$

$$x^r \bullet M(x) = G(x) \bullet Q(x) + R(x)$$

四、循环冗余码CRC

- 生成多项式G（x）应满足以下条件：

① 必须是r次多项式，最高项x^r 和x^0的系数为1，即它对应的二进制编码是r+1位的。

② CRC校验码的任何一位发生错误，余数不为零；且不同位发生错误，余数不同。

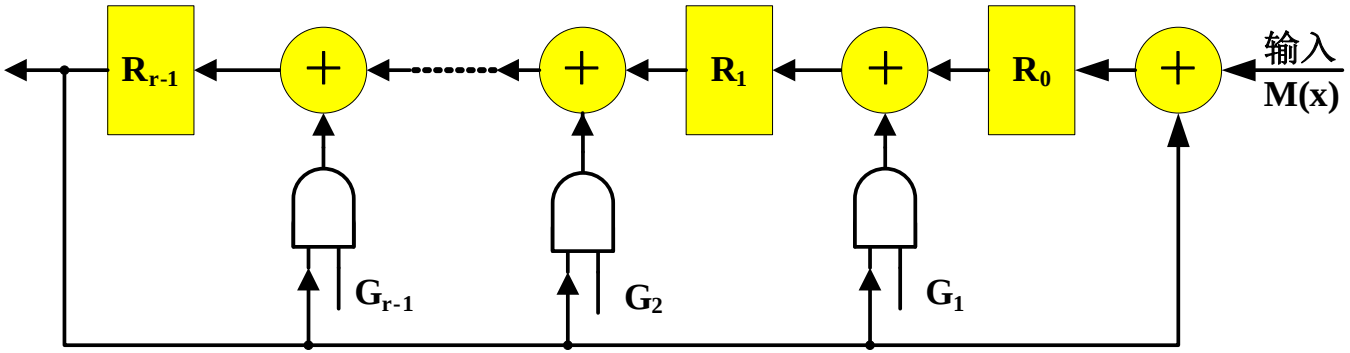
③ 对余数继续模2除，应使余数循环。

- 目前已经有多种生成多项式被列入国际标准中，如：CRC-4、CRC-12、CRC-16、CCITT-16、CRC-32等。

四、循环冗余码CRC

- 目前常用的CRC-16多项式为$x^{16}+x^{12}+x^5+1$（记为1021），CCITT-16多项式为$x^{16}+x^{15}+x^2+1$（记为8005）。

CRC（n,k）校验码串行生成电路原理图



图中Gi代表生成多项式G（x）各项的系数

四、循环冗余码CRC

2、译码

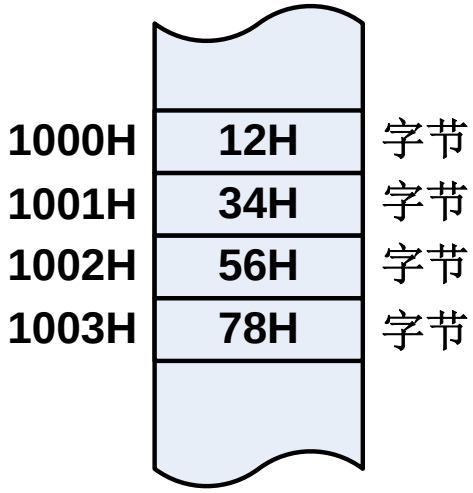
- 接收端的校验过程就是用G（x）来除接收到的码字多项式的过程。
  - 若余式为零则认为传输无差错；
  - 若余式不为零则传输有差错。出错的位置与余数值是一一对应的关系，通过查找出错模式表，即可以确定是那一位出错。

3.7 现代计算机系统的数据表示

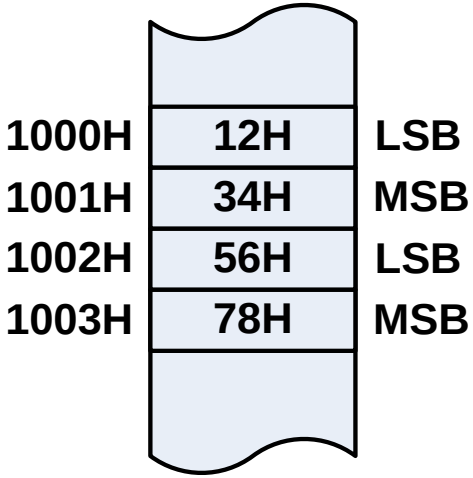
- 以Pentium系列的微处理器为例：
  - 编址方式：X86CPU是按字节编址的，即每8位就有一个编码地址。
  - 数据存取的尺寸：字节数据、字数据、双字数据、四字数据等。
  - 数据在内存中的存放位置：低字节（LSB）存放在低地址，高字节（MSB）存放在高地址。

3.7 现代计算机系统的数据表示

- 字节数据、字数据、双字数据在内存中的存储



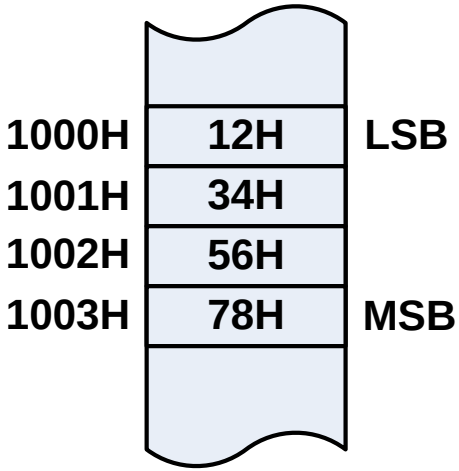
字节数据



字数据

MOV AX, [1000H]

AX= 3412H



双字数据

MOV EAX, [1000H]

EAX= 78563412H

3.7 现代计算机系统的数据表示

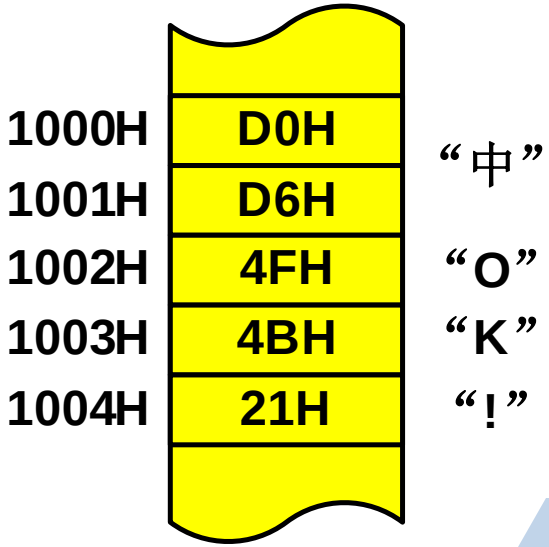
- 在Pentium系列CPU中，无论是字节数据、字数据还是双字、四字数据，它们都可以是无符号数、带符号数、或者BCD。
- 那么，如何区分内存某个单元存放的数据尺寸？如何分辨它们的数据类型？仍旧取决于访问它的指令的操作数尺寸和指令本身的功能。
- 当数据为有符号数时，默认它是补码表示的机器数。例如，假如某个有符号字节数据的二进制代码是90H，则它的值为－70H。

3.7 现代计算机系统的数据表示

❖ 几种类型的数据在X86系列CPU中的表示形式 ：

1、字符串

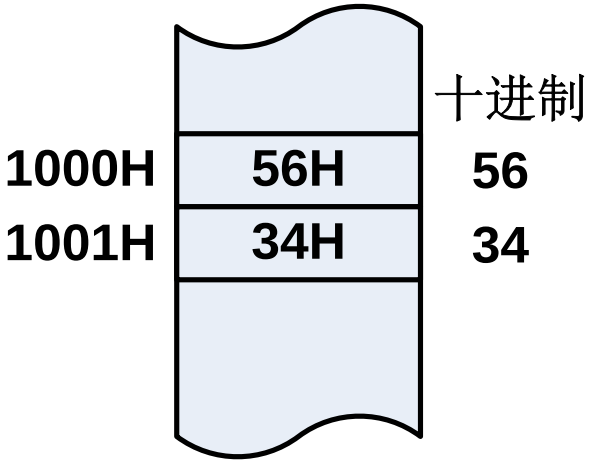
- 由字符的ASCII码（1B）或者文字的Unicode编码（2B）组成，按顺序存放在内存或寄存器中。
- Windows下以0做结束符



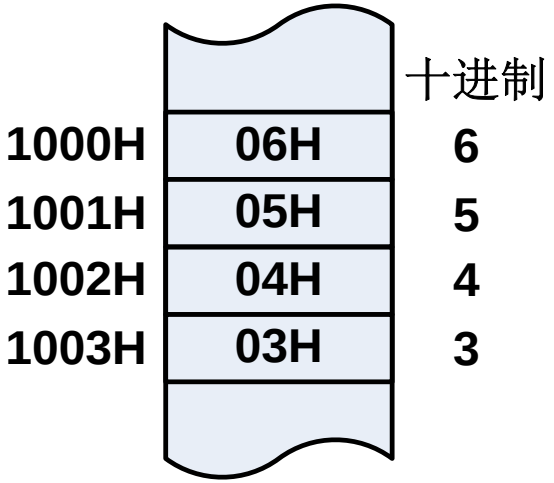
3.7 现代计算机系统的数据表示

2、BCD

- BCD数据分为压缩的（packed）BCD码（4位）和非压缩（unpacked）的BCD码（8位）两种。



（a）压缩BCD码



（b）非压缩BCD码

3.7 现代计算机系统的数据表示

3、指针

- 指针实际上是内存单元的地址，因此是无符号数据。IA构架中定义了两种类型的指针：
  - 近指针：32位，用于定义段内偏移量和段内访问；
  - 全指针：又称远指针，48位，用于段间访问。

4、浮点数

- Pentium系列CPU支持IEEE 754标准的3种浮点数格式：单精度、双精度和扩展精度浮点数。

本章小结

- **数制有两个要素**：基数R与位权W。计算机中的信息均由二进制来表示，即R=2，$W=2^i$。用于表示十进制数值的二进制编码被称为BCD码，4位二进制编码表示一个十进制数字。
- 机器数是数值数据在机器中的表示形式，根据小数点的位置是否浮动，可以分为**定点数和浮点数**。
  - 定点机器数根据小数点的隐含位置又分为**定点小数**和**定点整数**两种。
  - 定点机器数有四种表示形式：**原码、反码、补码和移码**。移码主要用于表示浮点数的阶码。

本章小结

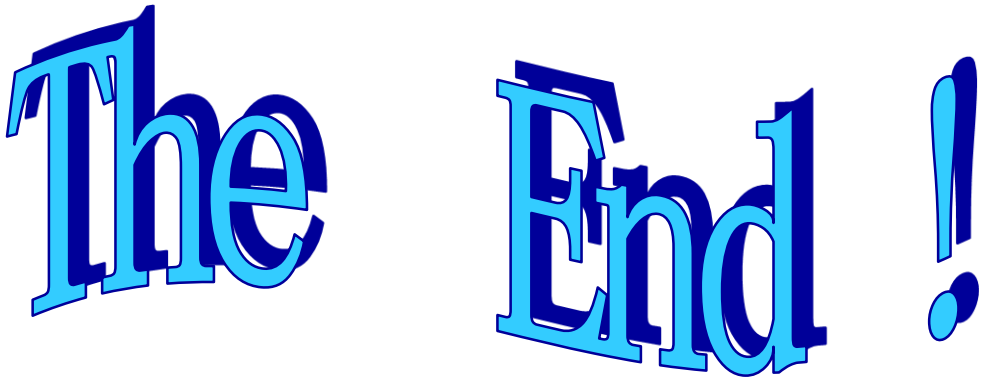
- **浮点机器数由阶码E和尾数M两部分构成，**阶码是定点整数，尾数是定点小数；阶码E（即指数）的底，一般隐含为2。浮点机器数的小数点的位置随阶码数值而变化。
- **IEEE754标准的浮点数**有单精度、双精度、临时浮点数3种格式，分别为32位、64位和80位。
- 浮点数的规格化表示方法。

本章小结

- 计算机中的非数值数据的表示：
  - 字符数据通常采用7位的ASCII码来表示。
  - 汉字的输入编码用于使用西文标准键盘输入汉字，汉字的机内码则用于汉字的存储、检索和处理，汉字的字模码则用于汉字的显示和打印输出。
- 校验码：用于检错和纠错。
  - 奇偶校验码是最简单的一种检错码，它可以检查出一位或奇数位错误。海明校验码是一种多重奇偶校验码，具有纠错能力，而CRC校验码则是一种目前广泛使用的纠错码，可以纠错一位。
- 本章重点为定点机器数和浮点机器数的表示方法。

作业

- P101：1，2，3，4，5，8，9（1）～（4），14, 17


The End !