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/

任课教师：赵备
邮　　箱：zhaobei@hdu.edu.cn

杭州電子科技大学
HANGZHOU DIANZI UNIVERSITY

第三章　信息编码与数据表示

- 计算机信息分类：
  - 数据信息
    - 数值数据
    - 非数值数据
  - 地址信息
  - 控制信息

第三章　信息编码与数据表示

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

一、进位计数制

- 则数值N的实际值为：

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

- 例如：R＝10，即十进制数。它的每一位上的数码D_i只能取0，1，2，……9；各个数码的权为10^i，i指示数码所处的位置，个位i＝0，十位i＝1，百位i＝2，依此类推。
- 思考：二进制、八进制、十六进制？

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。（按式3.1）
  - 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（Hexadecimal）数分别用D、B、Q、H来标志。
- 例如：$(1011)_2 \rightarrow (1011)_B \rightarrow 1011B \rightarrow 1011b$
  - $(123.45)_{10} \rightarrow (\ 123.45\ )_D \rightarrow 123.45D \rightarrow 123.45$
  - $(2B.D)_{16} = (2B.D)_H = (43.8125)_{10} = (53.64)_Q$

（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
- （a）十进制转化为R进制
- （b）二进制转化为八、十六进制

（a）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。
- 例：（123.75）$_{10}$=（ 1111011.11 ）$_{2}$
  - （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。

- 例如：$(0.35)_{10}=(\ ?\ )_2$无法整除
  - $(0.6875)_{10}=(\ ?\ )_2$位数太长

- 若要求精度大于10%，则表示“=”左右两边的十进制值的差的绝对值<10%。
  - 则我们只需取4位二进制小数即可满足要求，因为$10\%>2^{-4}$。

（b）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
  - 以小数点为中心分别向两边分组，每四位一组，写出对应的十六进制符号。（不够位数则在两边加0补足4位 ）

- 例：（1011111.11）$_2$

=（ **137.6** ）$_8$=（ **5F.C** ）$_{16}$

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，最高位和最低位的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，最高位和最低位的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、比较容易找到具有二值状态的物理器件来表示数据和实现存储。
- 2、二值性使二进制数的存储具有抗干扰能力强、可靠性高等优点。
- 3、二进制的运算规则简单，运算过程中的输入状态和输出状态较少，便于硬件实现。
- 4、二进制数的0和1与逻辑推理中的“真”和“假”相对应，为实现逻辑运算和逻辑判断提供了便利。

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

- 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。
- 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。
- 余3码：对应的8421码加上0011构成的。

（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，大大减少了十→二和二→十转换，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

- 计算机中参与运算的数据有两种：
  - **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。计算机中的带符号数据又称为**机器数**。

3.2 数据格式

- 1、机器数与真值
  - **机器数**：把“+”、“–”符号代码化，并保存在计算机中的数据。
  - **真值**：是指机器数所真正表示的数值，用数值并冠以“+”、“–”符号的方法来表示。
  - 机器数的编码方法：原码、反码、补码、移码。

3.2 数据格式

- 计算机硬件如何区分它们是无符号数据还是带符号数据的呢？（存储时，硬件无须区分，但是在运算或执行比较、跳转操作时，则通过程序中不同的指令加以区分。）
- 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。
- 程序A
  - 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
  - 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）$_{10}$，所以执行结果是不转移，顺序执行。

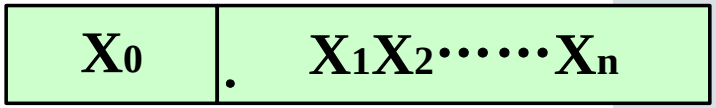
3.2 数据格式

2、小数点的表示方法

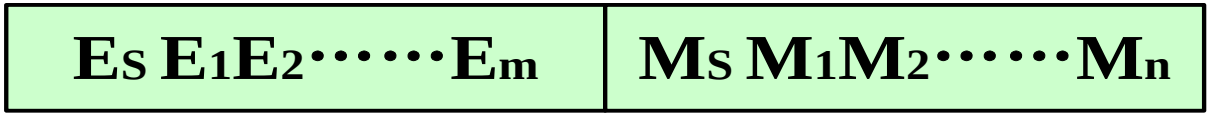
- 在机器数中，小数点及其位置是隐含规定的；有两种隐含方式：
  - 定点数：小数点的位置是固定不变的
  - 浮点数：小数点的位置是浮动的
- 定点机器数分为定点小数、定点整数两种。
- 浮点机器数中小数点的位置由阶码规定，因此是浮动的。

3.2 数据格式

（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。
- 1、原码表示法
- 2、补码表示法
- 3、反码表示法
- 4、移码表示法
- 5、定点机器数转换

1、原码表示法

- （1）表示方法：最高位表示数的符号，其余表示数值位。
  - 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$[X]_{原}=1.X_1X_2\cdots\cdots X_n$。

“，”和“.”只用于助记，在计算机中并无专用部件来表示

1、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：$[0]_{原}$＝？

1、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

2、补码表示法

- （1）模（mod）的概念：把一个计量单位称之为模或模数。
  - 例如，时钟是以12进制进行计数循环的，即以12为模。在时钟上，时针加上（正拨）12的整数倍或减去（反拨）12的整数倍，时针的位置不变。14点钟在舍去模12后，成为（下午）2点钟（14=14-12=2）。从0点出发逆时针拨10格即减去10小时，也可看成从0点出发顺时针拨2格（加上2小时），即2点（0-10=-10=-10+12=2）。因此，在模12的前提下，-10可映射为+2。由此可见，对于一个模数为12的循环系统来说，加2和减10的效果是一样的；因此，在以12为模的系统中，凡是减10的运算都可以用加2来代替，这就把减法问题转化成加法问题了（注：计算机的硬件结构中只有加法器，所以大部分的运算都必须最终转换为加法）。10和2对模12而言互为补数。

  - 同理，计算机的运算部件与寄存器都有一定字长的限制（假设字长为8），因此它的运算也是一种模运算。当计数器计满8位也就是256个数后会产生溢出，又从头开始计数。产生溢出的量就是计数器的模，显然，8位二进制数，它的模数为2^8=256。在计算中，两个互补的数称为“补码”。

2、补码表示法

- （2）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$[X]_{补}=1.\overline{X}_1\overline{X}_2\cdots\cdots\overline{X}_n+0.00\cdots\cdots1$。

2、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：$[0]_{补}$＝？

2、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0
  - $[+0]_{补}$= 00…0　　$[-0]_{补}$= 00…0
  - 10…0用来表示-2^n（定点整数）和-1（定点小数）
- （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：$-1 \leqslant X \leqslant 1-2^{-n}$
- 计算机中的整型数据（int）均用补码来表示。

包括1位符号位，n位数值位

3、反码表示法

- （1）**表示方法**：最高位表示数的符号，其他位表示数值位。
  - 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$[X]_{反}=1.\overline{X_1}\,\overline{X_2}\cdots\cdots\overline{X_n}$。

3、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：$[0]_{反}$＝？

3、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

4、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，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$[X]_{移}=1,X_1X_2\cdots\cdots X_n$；
  - 若$X=-X_1X_2\cdots\cdots X_n$，则$[X]_{移}=0,\overline{X_1}\,\overline{X_2}\cdots\cdots\overline{X_n}+1$。
- 对于定点小数：
  - 若$X=+0.X_1X_2\cdots\cdots X_n$，则$[X]_{移}=1.X_1X_2\cdots\cdots X_n$；
  - 若$X=-0.X_1X_2\cdots\cdots X_n$，则$[X]_{移}=0.\overline{X_1}\,\overline{X_2}\cdots\cdots\overline{X_n}+0.00\cdots\cdots1$。

移码表示：即为补码的符号位取反

4、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：$[0]_{移}$＝？

4、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　　$[-0]_{移}$= 1
  - 00…0用来表示-2^n（定点整数）和-1（定点小数）
- （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：$-2^n \leqslant X \leqslant 2^n-1$
  - 定点小数：$-1 \leqslant X \leqslant 1-2^{-n}$
- 移码通常作为浮点数的阶码。

包括1位符号位，n位数值位

5、定点机器数转换

- 机器数转换为真值
  - 四种定点机器数转换为真值的方法要点是：首先根据机器数的符号位确定真值的正负，然后对照机器数的定义和表示，反方向求出真值的绝对值
- 机器数之间的相互转换
  - 原码、补码、反码和移码之间的相互转换，最简单的方法是先求出它们的真值，然后再转换为另一种表示方法。

3.4 浮点机器数的表示方法

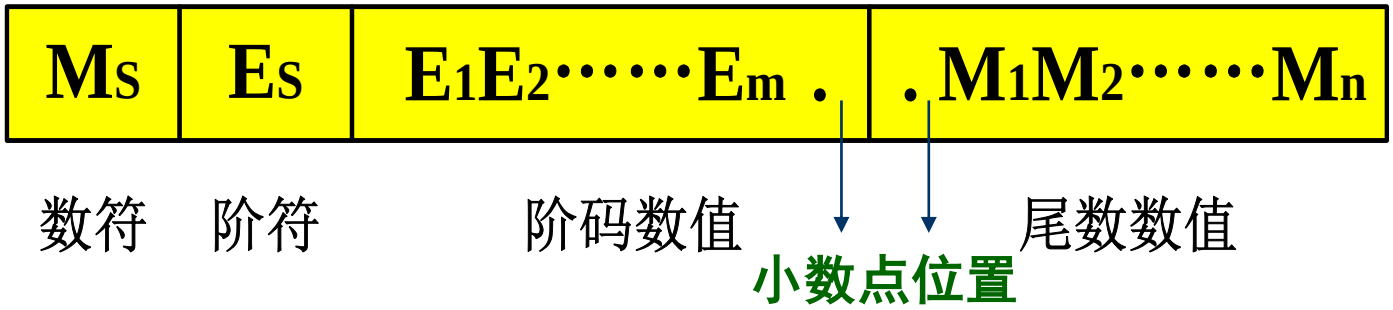
1. 浮点机器数的格式
2. 浮点机器数的规格化表示
3. 浮点数的表示范围

1、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。**

1、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。

1、浮点机器数的格式

- 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。

表3.3 IEEE 754 浮点数标准格式

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	$N=(-1)^{M}\times(1.M_1M_2\cdots\cdots M_n)\times 2^{E-127}$
长实数	64	53	11	$N=(-1)^{M}\times(1.M_1M_2\cdots\cdots M_n)\times 2^{E-1023}$
临时实数	80	65	15	——

2、浮点机器数的规格化表示

- 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：最高有效位为1，即

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

2、浮点机器数的规格化表示

- 为便于计算机硬件对尾数的机器数形式的规格化判断，通常采用下列方法实现判定：
  - （1）对于原码表示的尾数，当最高有效位（M1）为1时，浮点数为规格化，即尾数为×.1×……×形式；
  - （2）对于补码表示的尾数，当符号位（MS）与最高有效位（M1）相异时，浮点数为规格化，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**。

2、浮点机器数的规格化表示

- 若尾数进行右移实现的规格化，则称为**右规**；若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：
  - （1）提高了浮点数据的精度；
  - （2）使程序能够更方便地交换浮点数据；
  - （3）可以使浮点数的运算更为简化。

2、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。写出X与Y的规格化浮点数。
  - （1）X＝－123.25
  - （2）Y＝34/128
- （1）$X=(-123.25)_{10}$
  $=(-1111011.01)_{2}$
  $=-0.111101101\times 2^{+7}$

2、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＝（34/128 ）$_{10}$

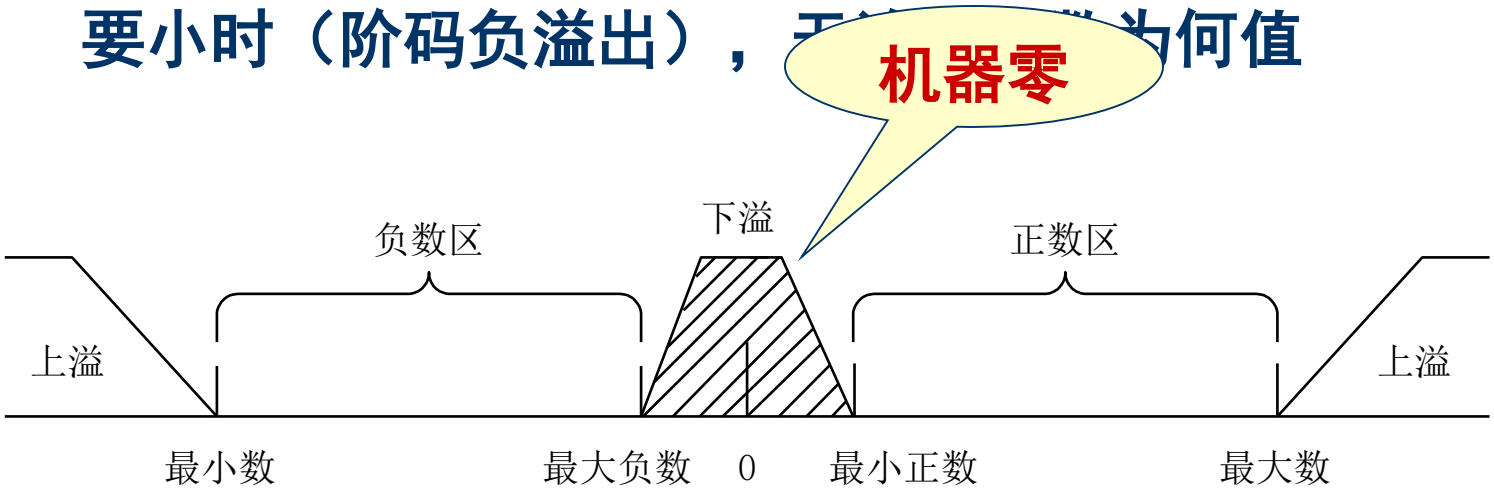
＝（0.010001）$_2$

＝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

3、浮点数的表示范围

- 在浮点数的表示范围中，有两种情况被称为机器零：
  - （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），无论尾数为何值



3、浮点数的表示范围

- 浮点数的表示范围通常：最小（负）数、最大负数、最小正数、最大（正）数。
  - 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，称为下溢。对于下溢的处理，计算机直接将其视为机器零。
  - 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。

3.5 非数值数据的表示

- 非数值数据：没有数值大小概念的信息，如文字和符号（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

- 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。
- 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

ASCII码分类

- 95个可打印或显示的字符：称为图形字符，有确定的结构形状，可在打印机和显示器等输出设备上输出；而且这些字符均可在计算机键盘上找到相应的键，按键后就可以将相应字符的二进制编码送入计算机内。

  包括52个引文大写和小写字母（A~Z、a~z）、10个十进制数字(0~9)、33个通用的运算符及标点符号。

- 33个控制字符：不可打印或显示，用于控制某些外设的工作特性、通讯协议或某些软件的运行情况。包括清屏、退格、换行等。

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、钱码等。

二、汉字编码

2. 汉字交换码

- 汉字交换码是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。
- 目前国内计算机系统所采用的标准信息处理交换码，是基于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。
- 该字符集共收录了6763个汉字和682个图形符号。6763个汉字按其使用频率和用途，又可分为一级常用汉字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，这样可对128×128＝16384个字符进行编码。

二、汉字编码

❖两种典型的数字编码：

- 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”。
- 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

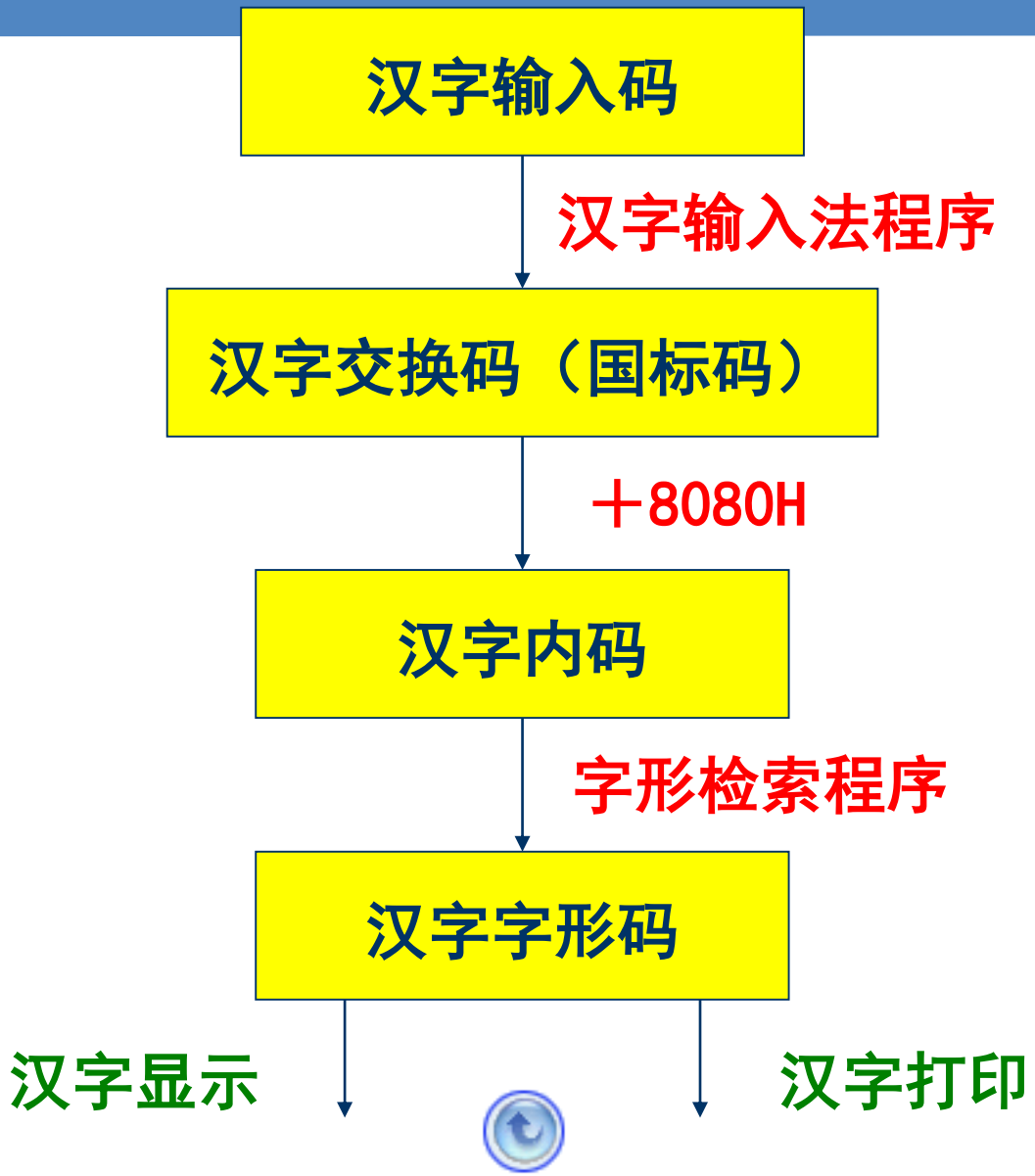
3、汉字内码

- 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。
- 英文字符的机内代码是七位的ASCII码，当用一个字节表示时，最高位为“0”。为了与英文字符能相互区别，汉字机内代码中两个字节的最高位均规定为“1”。
- 机内码等于汉字国标码加上8080H。例如“中”字的机内码为D6D0H。

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。


汉字输入码
汉字输入法程序
汉字交换码（国标码）
＋8080H
汉字内码
字形检索程序
汉字字形码
汉字显示
汉字打印

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

3.6 校验码

- 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。

  计算机中的错误（不包括软件的错误）类型：

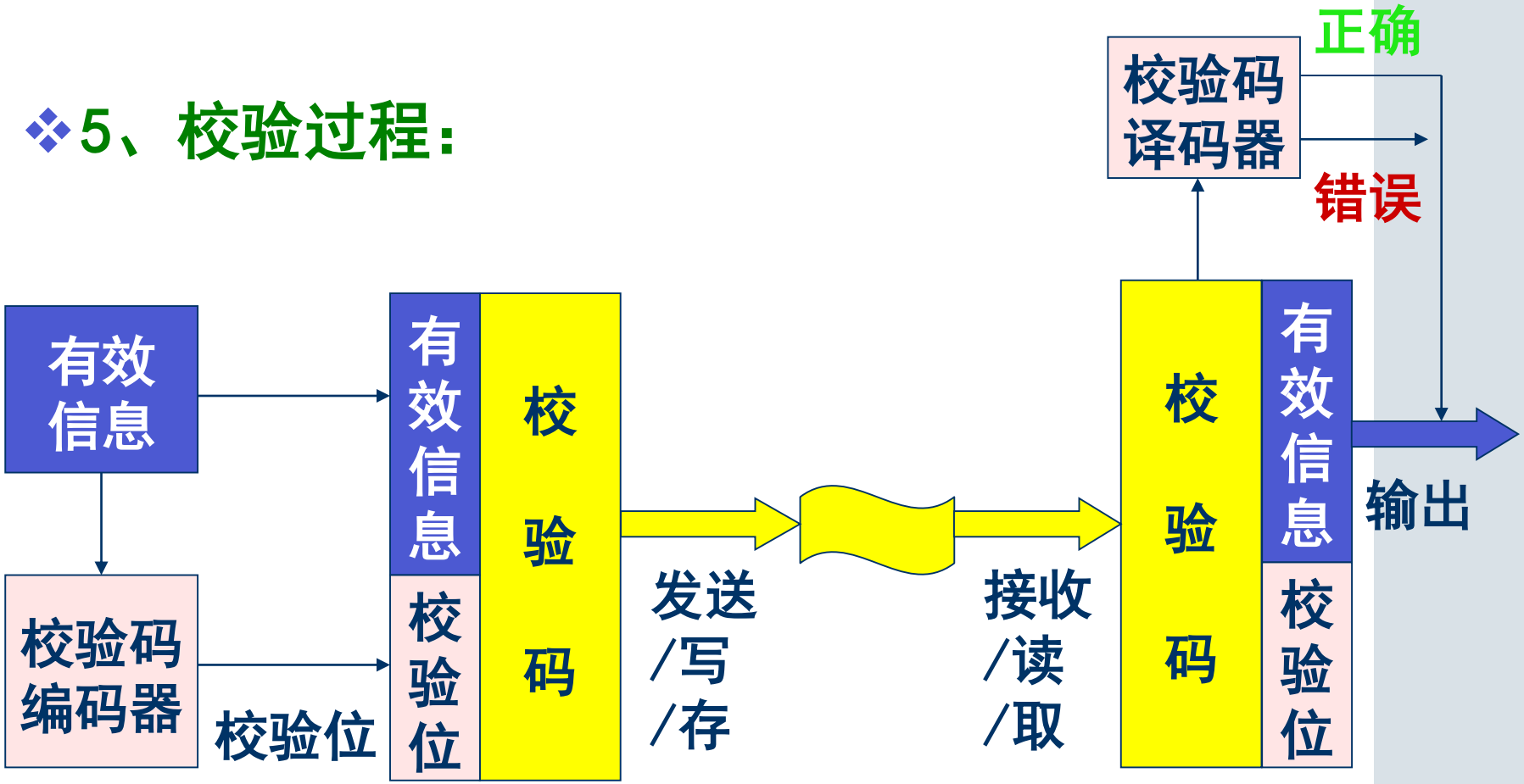
  （1）固定性错误 ---元器件故障；

  （2）突发性错误 ---噪声干扰。

- 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）

3.6 校验码

❖5、校验过程：


正确
校验码
译码器
错误
有效
信息
校验码
编码器
校验位
有
效
信
息
校
验
位
校
验
码
发送
/写
/存
接收
/读
/取
校
验
码
有
效
信
息
校
验
位
输出

3.6 校验码

❖ 6、校验码原理

- **码距（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。也就是两个代码C1与C2之间不同的比特数。如：1100与1010的码距为2;1111与0000的码距为4。
- 一种码制的码距是指该码制中任意（所有）两个代码之间的**最小距离**。

  如：一个编码系统有四种编码分别为：0000，0011，1100，1111，此编码系统中0000与1111的码距为4;0000与0011的码距为2，是此编码系统的最小码距。因此该编码系统的码距为2。

3.6 校验码

- **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**。
  - 校验码的检错纠错能力与码距的关系如下：
    - 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$\frac{d-1}{2}$位错误。
    - 若码距d为偶数，则可以发现$\frac{d}{2}$位错误，并能够纠正（$\frac{d}{2}$-1）位错误。

3.6 校验码

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。

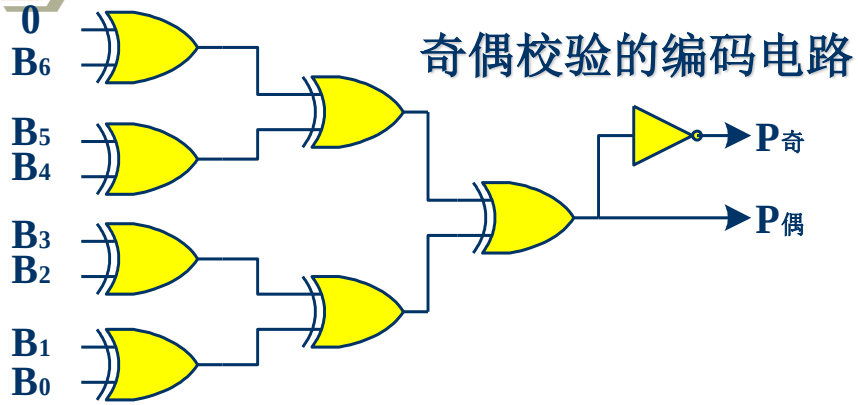
  奇（偶）校验码的编码和译码在硬件上通常采用异或非门（异或门）实现。

  - 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。
    - 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：

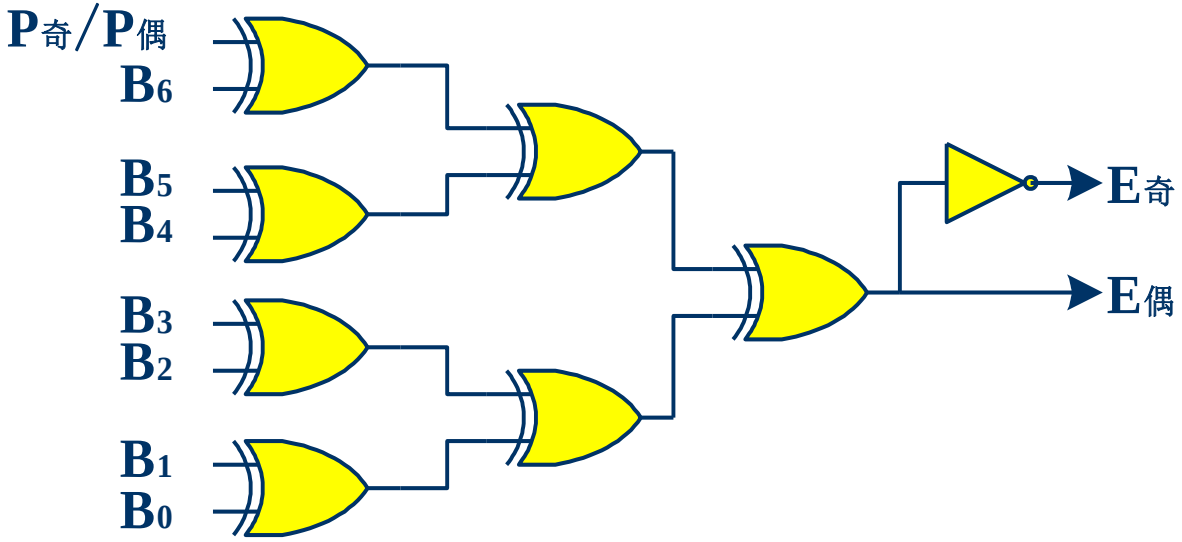
字符“A”的ASCII码为41H，奇校验码为C1H，偶校验码为41H。

- 2、译码
  - 在接收端，必须检验接收到的校验码信息的奇偶性，对于奇校验码，校验码中“1”的个数应该为奇数，否则出错；对于偶校验码，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

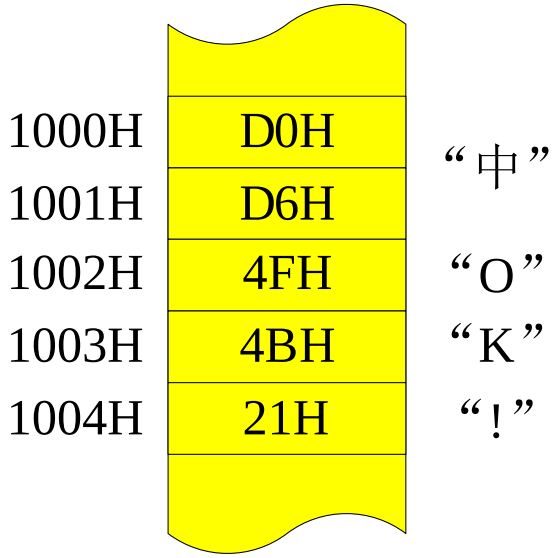


例如：

字符“A”的ASCII码为41H，奇校验码为C1H，偶校验码为41H。接收正确的情况下，$E_{奇}$=0，$E_{偶}$=0。

3.7 现代计算机系统的数据表示

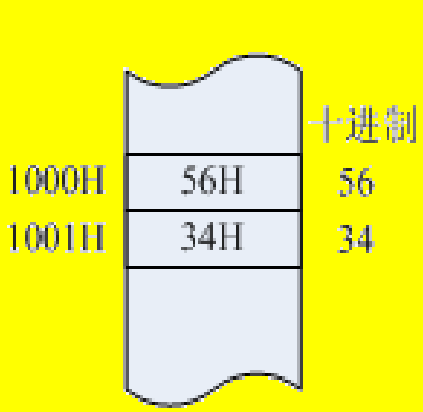
- 几种类型的数据在Pentium系列CPU中的表示形式 ：
- 1、字符串
  - 由字符的ASCII码或者文字的Unicode编码组成，按顺序存放在内存或寄存器中。每个ASCII码字符占1个字节，每个Unicode编码占用2个字节



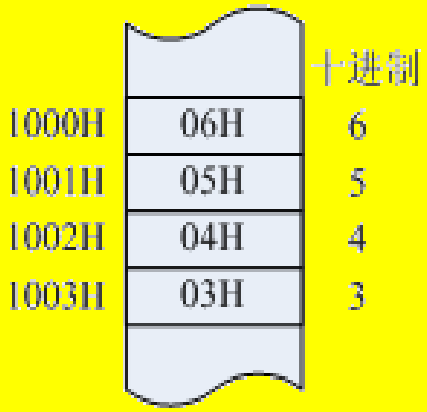
3.7 现代计算机系统的数据表示

BCD

- 在基于Pentium系列CPU的IA构架中，BCD数据分为压缩的（packed）BCD码和非压缩（unpacked）的BCD码两种，前者以每字节2位BCD数字的形式存储，后者以每字节1位BCD数字的方法存储



（a）压缩 BCD 码



（b）非压缩 BCD 码

图3.12 BCD 数据在内存的表示

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，
  - （1）一种是32位的近指针，用于定义段内偏移量和段内访问；
  - （2）一种是48位的全指针，又称远指针，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。
  - 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。

本章小结

- 真值转化为定点机器数时，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。
- 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。
- 计算机中使用校验码来检错和纠错。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。

课后作业

3.3 写出下列各数的原码、反码和补码，机器数长度为8位。

（1）0　　(4) —19/128

3.4 写出下列各机器数的二进制真值X。

（2）$[X]_{补}$=1.1001　　（4）$[X]_{原}$=1.1101

（6）$[X]_{反}$=1.1011　　（8）$[X]_{移}$=1,1001

3.10 将下列十进制数转换为IEEE754 单精度浮点数格式：

（1）＋36．75

3.11 求下列各IEEE754 单精度浮点数的十进制真值：

（1）43990000H

3.12 在汉字系统中，有哪几种编码？它们各自有什么作用？

The End!