数据的机器级表示

- 不管以什么形态出现，在计算机内部，数据最终都由机器指令来处理。
- 机器级数据分两大类：
  - 数值数据:无符号/带符号整数、浮点数中实数)、十进制数
  - 非数值数据:逻辑数(包括位串)、西文字符和汉字
- 真值:现实中世界中数的值
- 机器数:计算机内部的0/1序列

数值的表示格式

进位计数制

> B 二进制 O八进制 D十进制 H十六进制

- 整数转换规则:除2取余，直至商为0，先得低位(除基取余法)
- 小数转换规则:乘2取整，至小数部分为0或取近似值(乘基取整法)

定点数

定点数:数据格式中小数点的位置固定不变

计算机中的定点数只采用**纯整数**或者**纯小数**表示

三种定点编码方式：原码、补码、反码

原码

定义

- 最高位为**符号位0/1+数值的绝对值形式**

表示范围

- 正数有$2^{n-1}-1$个（去掉00000......)
- 负数有$2^{n-1}-1$个（去掉11111......)
- 整数表示范围:$-(2^n-1)$~2^n-1
- 小数表示范围:$-（1-2^{-n}）$~$1-2^{-n}$

特点

- 值[+0]，[-0]的原码分别为00000、10000，形式不唯一；
- 运算时需对符号位进行处理。

补码

定义

（1）当 X_T 为正数时，$[X_T]_{补} = X_T = M + X_T \pmod{M}$；

（2）当 X_T 为负数时，$[X_T]_{补} = M - |X_T| = M + X_T \pmod{M}$。

[X]补=2^n+X （MOD 2^n）

表示规律(书本例题P30-P32)

0正1负；

正数的补码就是原码，负数的补码按照从右至左，见1后反

1. 设机器数有8位，求123和-123的补码表示。

解: 123 = 127- 4 = 01111111B - 100B = 01111011B

-123= - 01111011B

$[01111011]_{补}$= 2^8 + 01111011 = 100000000 + 01111011

= 01111011 (mod 2^8)，即 7BH。

$[-01111011]_{补}$= 2^8 – 01111011 = 10000 0000 - 01111011

= 1111 1111 – 0111 1011 +1

= 1000 0100 +1 ← 各位取反，末位加1

= 1000 0101，即 85H。

表示范围

- 正数有$2^{n-1}-1$个（去掉00000......)
- 负数有2^{n-1}个(100000用来表示-2^{n-1})
- 整数表示范围:-2^n~2^n-1
- 小数表示范围:-1~$1-2^{-n}$

特点:

补码正是利用补数概念，把负数映射到正数域中(平移模值，小数的模为2，n位整数的模为2^n)从而将数的正负符号数码化，将减法运算转换为加法运算。

反码

X是正数，[X]反=[X]原；X是负数，符号+数值取反。

0正1负，各位取反；可利用反码来求补码，即在末尾加一。

移码

> 主要用来表示浮点数阶码

0负1正，数码位同补码

如：　$x = +10101$　　$[x]_{移} = 1,10101$

　　$x = -10101$　　$[x]_{移} = 0,01011$

便于浮点数加减运算时对阶操作

例：

如机器字长为8，则

- 真值19 原码：00010011 反码：00010011 补码：00010011 移码：10010011
- 真值-19 原码：10010011 反码：11101100 补码：11101101 移码：01101101
- 真值+0.75 原码：01100000 反码：0.1100000 补码：0.1100000
- 真值-0.75 原码：11100000 反码：1.0011111 补码：1.1000000

整数

无符号符号数

编码中没有符号位，能表示的最大值大于位数相同的带符号整数

有无符号数的差别：

- 扩充操作有差别

  无符号数一般使用0扩展，有符号数使用符号扩展

- 数的比较有差异

  无符号数不用考虑最高位

- 溢出判断有差异

浮点数

对**尾数**和**指数**分别编码，就可表示一个浮点数

$$N = M \times r^E$$

IEEE 754

IEEE754尾数带有一个隐藏位，偏置常数与其他相比也少一。

例1： IEEE 754 单精度浮点数 BEE00000H的十进制？

1	0111 1101	110 0000 0000 0000 0000 0000

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

- 符号s: 1 => 负数
- 阶码:
  - 0111 $1101_{two} = 125_{ten}$
  - 偏移值调整计算: e=125 - 127 = -2
- 尾数: 1.75 即：0.11 + 1(隐藏) = 1.75

故，十进制表示：$-1.75_{ten} \times 2^{-2} = -0.4375$

	符号	指数	尾数
单精度	1 bit	8 bits	23 bits
双精度	1 bit	11 bits	52 bits

规格化数：+/-1.xxxxxxxxxx x 2^Exponent

- 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）

指数Exponent范围：

SP： (-126~127),偏移127

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

DP：(-1022~1023)偏移1023

0000 … 0001 (-1022) ~ 1111 … 1110 (1023)

浮点数的分类：

- 0的表示

  阶码与尾数都为零，符号位有效表示正负
- 无穷的表示

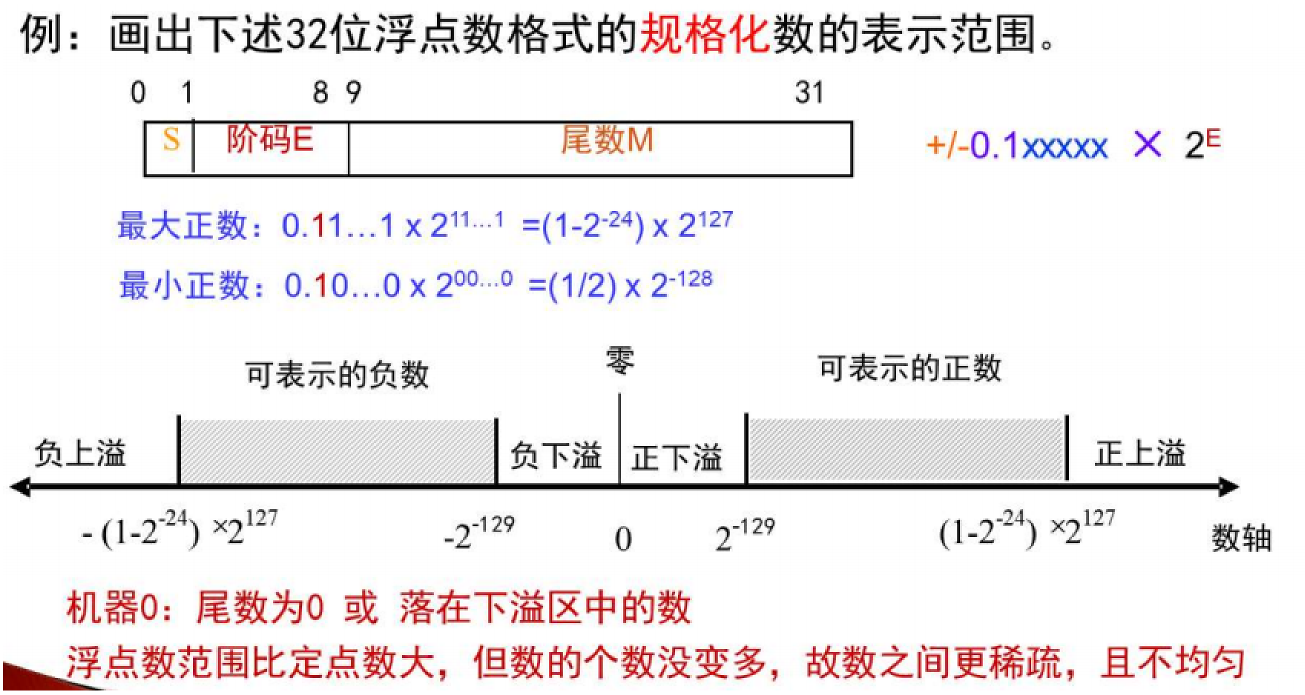
  阶码全1，尾数全0，符号位有效表示正负
- 非数

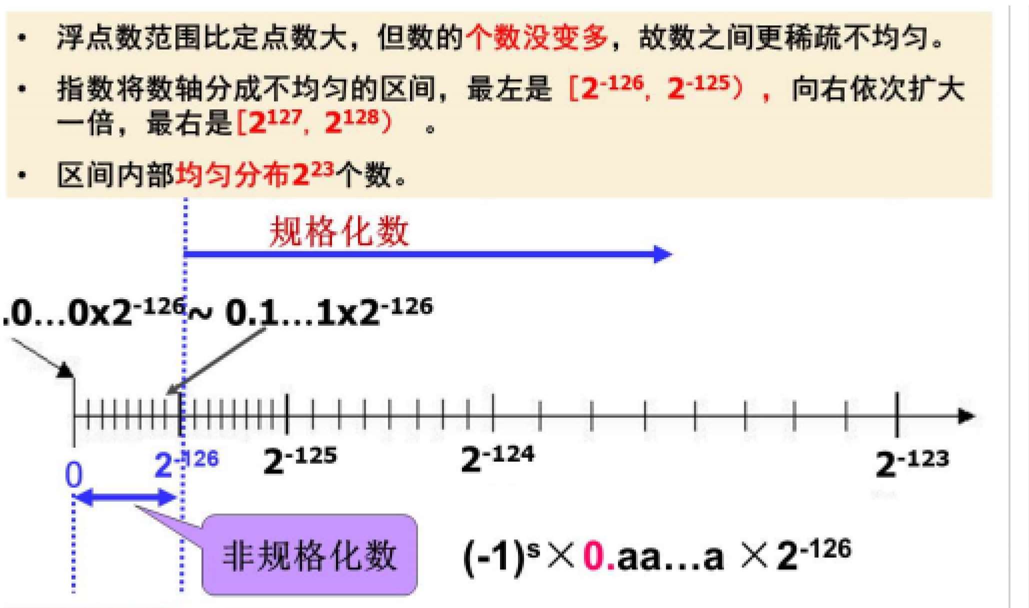
  阶码全1，尾数不全为0
- 非规格化数

  阶码全0，尾数高位为0但不全为0

  非规格化数处理阶码下溢，是的出现比最小规格化数还小的数时程序也能继续运行下去。

浮点数的表示范围：





最大的单精度数：$+1.11111111 \times 2^{127}$

浮点数的大数吃小数问题ppt上

十进制数的表示

- ASCII码
- BCD码

非数值数据的表示

逻辑数据

用一位表示。例： 真：1/假：0.

可参考flag寄存器

西文字符

常用7位ASCII码表示

汉字

GB2312国标码->汉字机内码

数据宽度

- 字长：即机器字长吗，指定点运算数据通路的宽度
- 字：表示呗处理信息的单位，用来度量数据类型的宽度

字和字长的宽度可以一阿姨那个，也可不同

数据的存储和排列

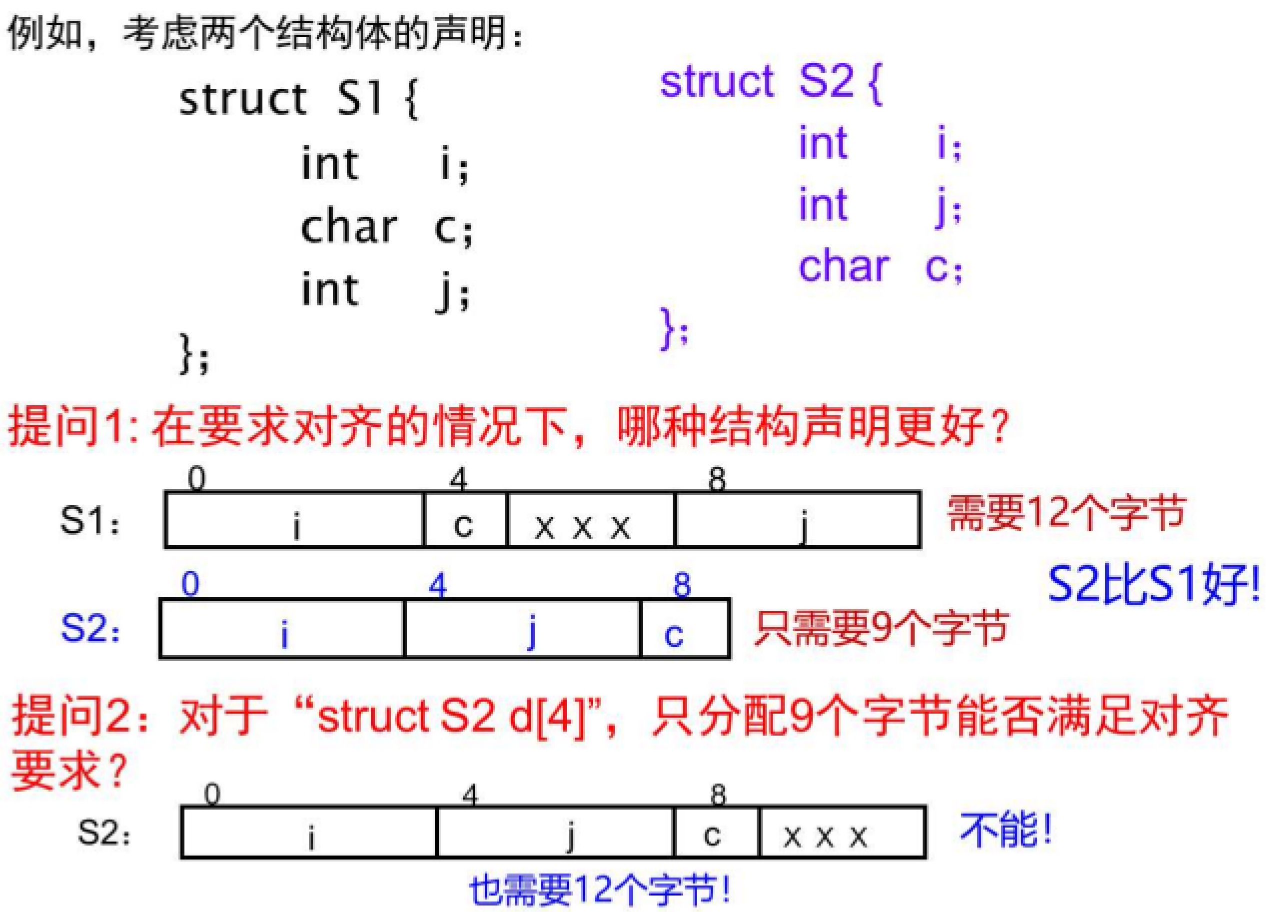
- 大端方式：将数据的最高有效字节MSB存放在低地址单元中
- 小端方式：将数据的最低有效字节LSB存放在低地址单元中。

例：在100—103存放FFFF0001H：

103	102	101	100	
FF	FF	00	01	little endian
01	00	FF	FF	big endian

对齐

> 要求数据段地址是相应边界的地址



提问2：S2是将结构体与结构体之间也要对其，不对齐的地址为s，s+9,s+18，s+27，只有第一个元素满足四字节对齐要求。

数据校验码

码距

任意两个合法码之间不相同的二进制位数的最小值

- 要具有差错能力，则码距>1
- 合理增大码距，就能提高发现错误的能力
    - 若码距d为奇数，则能发现d-1位错，或能纠正(d-1)/2位错若
    - 码距d为偶数，则能发现d/2位错，并能纠正(d/2-1)位错

奇偶校验码

> 基本思想:**增加一位校验位**，根据接收端的数据求出新校验位，根据新校验位确定是否发生了错误。常用于存储器读写检查，或ASCII字符传送过程中的检查。

当实际数据中“1”的个数为偶数的时候，校验位是“0”，反之校验位是“1”，换句话说，**数据位（N位）和 校验位（1位）组成的 编码数据（N+1位）中，将总共包含偶数个1。**

检查 **编码数据（N+1位）**，如果其包含**偶数**个 **1**，则 **校验通过**

奇偶校验码只能发现奇数位错，且不能确定出错位置，因此不具有纠错能力

若计算最终校验位得到P*

- 若P*=1，奇数个错
- 若P*=0，正确或偶数个错

海明校验码

> 海明校验全码实际上就是一种多重奇偶校验码

校验位数的确定：

2^k>=1(无错)+n(数据位错)+k(校验位错)

说明：M7（第11位）：1011，故由P1,P2,P4识别。

校验位Pi的位置在2的i-1次方，但是除了最高位

- 如果故障字各位全部是0，则表示没有发生错误
- 如果故障字中有且仅有一位为1,则表示校验位中有一位出错,不需要纠正（即故障位为检验码）
- 如果故障字中多位为1（即故障位为数据位）,则表示有一个教据位出错,其在码字中的出错位由故障字的数值来决定所正时只要将出错位取反即可

校验过程例：

(2) 数据位M’= 01111010，校验位P’’=0011。

用M’生成新的校验位P’为：

$P_4' = M_5' \oplus M_6' \oplus M_7' \oplus M_8' = 1 \oplus 1 \oplus 1 \oplus 0 = 1$

$P_3' = M_2' \oplus M_3' \oplus M_4' \oplus M_8' = 1 \oplus 0 \oplus 1 \oplus 0 = 0$

$P_2' = M_1' \oplus M_3' \oplus M_4' \oplus M_6' \oplus M_7' = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 = 1$

$P_1' = M_1' \oplus M_2' \oplus M_4' \oplus M_5' \oplus M_7' = 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 0$

故障字S为：

$S_4 = P_4' \oplus P_4'' = 1 \oplus 0 = 1$

$S_3 = P_3' \oplus P_3'' = 0 \oplus 0 = 0$

$S_2 = P_2' \oplus P_2'' = 1 \oplus 1 = 0$

$S_1 = P_1' \oplus P_1'' = 0 \oplus 1 = 1$

因此，错误位是第9位，排列的是数据位M_5，即M_5错，纠错时只要将码字的第9位（M_5）取反即可。

(循环冗余校验)CRC码

假设M(X)为-个n位的二进制数据

- 将 M(x)左移K位,用一个K+1位的生成多项式G(x)去除M(x)得到一个K位余数
- 将K位余数(校验位)拼接到M(x)后面一起传输
- 将接收到的数据与校验位用同样的生成多项式G(w)相除,,或余数为0，则无错

具有纠错能力:

- 不同出错位置的余数不同。
- 只要出错位置相同余数一定相同，与码字无关。
- 通信中一般只检错不纠错

运算方法和运算部件

ALU的设计与实现

定点数的加法

- 计算机中，常用补码进行加减运算
- 补码可将减法变加法进行运算
- 补码运算特点:符号位数值位一同运算

- [X]补+[Y]补=[X+Y]补

- $[X]_{补}-[Y]_{补}=[X]_{补}+[-Y]_{补}=[X-Y]_{补}$