2 运算方法和运算器

提纲

2.1 数据与文字的表示方法

提纲

2.1.1 数据格式

- 一、复习

- 10进制和R进制之间的转换
  - R进制到10进制：

$$\sum_{i=n}^{-m} k_i \times r^i$$

  - 10进制到R进制：
    - ✓ 整数部分：除r取余，r为进制基数
    - ✓ 小数部分：乘r取整

2.1.1 数据格式

- 二、数值数据

- 计算机在数据、文字的表示方式时，应该考虑以下几个因素：
  - 表示的数据类型（符号、小数点、数值）
  - 数值的范围
  - 数值精度
  - 存储、处理、传送的硬件代价

2.1.1 数据格式

- 三、计算机常用的数据表示格式有两种：

- 定点表示：小数点位置固定
- 浮点表示：小数点位置不固定

2.1.1 数据格式

- 四、定点表示法

- 所有数据的小数点位置固定不变
- 理论上位置可以任意，但实际上将数据表示有两种方法（小数点位置固定-定点表示法/定点格式）：
  - 纯小数
  - 纯整数
- 定点数表示：
  - 带符号数
  - 不带符号数

2.1.1 数据格式

- ①定点纯小数

$$\underline{x_0}\ \underline{x_1\ x_2\ x_3 \cdots x_{n-1}\ x_n}$$

符号

小数点固定于符号位之后，不需专门存放位置

量值

表示数的范围是 $0 \le |x| \le 1 - 2^{-n}$

（最小数、最大数、最接近0的正数、最接近0的负数）

2.1.1 数据格式

- ②纯小数的表示范围

x=0.00…0 x=1.00…0	x=0	正0和负0都是0
x=0.11…1	$x=1-2^{-n}$	最大
x=0.00…01	$x=2^{-n}$	最接近0的正数
x=1.00…01	$x=-2^{-n}$	最接近0的负数
x=1.11…1	$x=-(1-2^{-n})$	最小

2.1.1 数据格式

- ③定点纯整数

$x_0\ x_1\ x_2\ x_3\ \dots\ x_{n-1}\ x_n$

符号

量值

小数点固定于最后一位之后，不需专门存放位置

表示数的范围是 $0 \le |x| \le 2^n - 1$

最小数、最大数、最接近0的正数、最接近0的负数呢

2.1.1 数据格式

- ④定点表示法的特点

- 定点数表示数的范围受字长限制，表示数的范围有限;
- 定点表示的精度有限

- 如果用定点表示，则如何表示实数（包括小数和整数）呢?
  - 引入浮点

2.1.1 数据格式

- 五、浮点表示：小数点位置随阶码不同而浮动

①格式： $N = R^{E}.M$

基数R,取固定的值,比如10,2等

指数E

尾数M

②机器中表示

阶符	阶码	数符	尾数

2.1.1 数据格式

- ③IEEE754标准(规定了浮点数的表示格式,运算规则等)
  - 规则规定了单精度(32)和双精度(64)的基本格式.
  - 规则中,尾数用原码,指数用移码(便于对阶和比较)

2.1.1 数据格式

- ③IEEE754标准(规定了浮点数的表示格式,运算规则等)

- 基数R=2，基数固定，采用隐含方式来表示它。
- 32位的浮点数：
  - S数的符号位，1位，在最高位，“0”表示正数，“1”表示负数。
  - M是尾数，23位，在低位部分，采用纯小数原码表示
  - E是阶码，8位，采用移码表示，移码比较大小方便。
- 规格化：若不对浮点数的表示作出规定，同一个浮点数的表示不惟一
  - 尾数域最左位(最高有效位)总是1，故这一位经常不予存储，而认为隐藏在小数点的左边。
  - 采用这种方式时，将浮点数的指数真值e变成阶码E时，应将指数e加上一个固定的偏移值127(01111111)，即E=e+127

2.1.1 数据格式

- ③IEEE754标准(规定了浮点数的表示格式,运算规则等)

- 当E=0且M≠0时，表示非规格化数
  - $(-1)^{s}\times 0.M\times 2^{-126}$
  - $(-1)^{s}\times 0.M\times 2^{-1022}$
- 当E=0且M=0时，表示x=0，结合S位，有+0和-0之分
- 当E为全1且M=0时，表示x为∞，，结合S位，有+∞和-∞之分
- 当E为全1且M≠ 0时，表示x为非数（NaN）

2.1.1 数据格式

- ③IEEE754标准(规定了浮点数的表示格式,运算规则等)

- 64位的浮点数中符号位1位，阶码域11位，尾数域52位，指数偏移值是1023。因此规格化的64位浮点数x的真值为：

$$x=(-1)^S\times(1.M)\times2^{E-1023}$$

$$e=E-1023$$

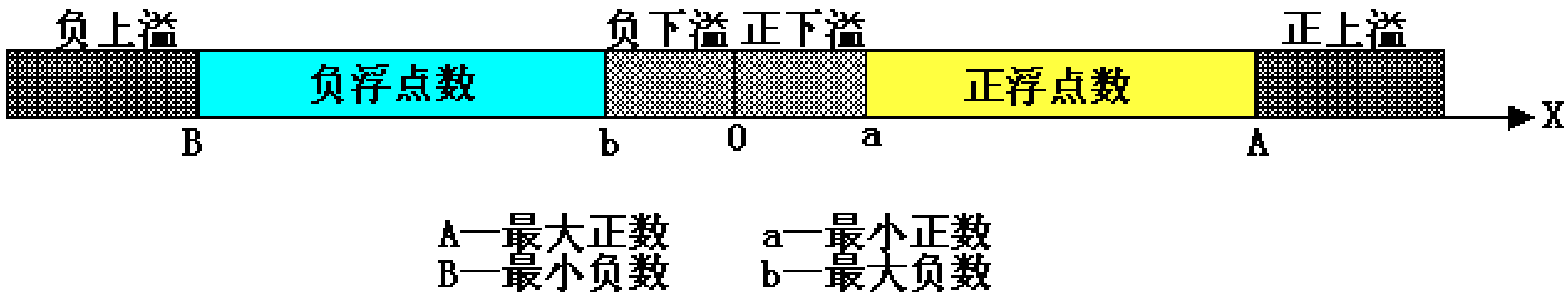
- 一个规格化的32位浮点数x的真值表示为

$$x=(-1)^S\times(1.M)\times2^{E-127}$$

$$e=E-127$$

2.1.1 数据格式

- ③IEEE754标准(规定了浮点数的表示格式,运算规则等)
- 浮点数表示范围如下图所示



- 最大正数：0 11111110 111 1111 1111 1111 1111 1111=$2^{127}*(1+(1-2^{-23}))$
- 最小正数：0 00000001 000 0000 0000 0000 0000 0000=$2^{-126}*1$
- 最小负数：1 11111110 111 1111 1111 1111 1111 1111=$-2^{127}*(1+(1-2^{-23}))$
- 最大负数：1 00000001 000 0000 0000 0000 0000 0000=$-2^{-126}*1$

2.1.1 数据格式

- ③IEEE754标准(规定了浮点数的表示格式,运算规则等)

- 例：若浮点数x的754标准存储格式为$(41360000)_{16}$，求其浮点数的十进制数值。
- 解：将16进制数展开后，可得二制数格式

  0　100 00010 011 0110 0000 0000 0000 0000

  S　阶码(8位)　　尾数(23位)

  指数e=阶码-127=10000010-01111111=00000011=$(3)_{10}$

  包括隐藏位1的尾数

  1.M=1.011 0110 0000 0000 0000 0000=1.011011

  于是有：$x=(-1)S\times 1.M\times 2^{e}=+(1.011011)\times 2^{3}=+1011.011=(11.375)_{10}$

2.1.1 数据格式

- ③IEEE754标准(规定了浮点数的表示格式,运算规则等)

- 例：将数$(20.59375)_{10}$转换成754标准的32位浮点数的二进制存储格式。
- 解：首先分别将整数和分数部分转换成二进制数：

$$20.59375=10100.10011$$

然后移动小数点，使其在第1，2位之间

$10100.10011=1.010010011\times2^4$

e=4于是得到：S=0, E=4+127=131, M=010010011

最后得到32位浮点数的二进制存储格式为：

01000001101001001100000000000000=(41A4C000)16

2.1.1 数据格式

- 六、十进制数串的表示
- 字符串形式
- BCD(压缩)：4位二进制表示一个十进制
- 编码方式
  - 有权码：（8421码、2421码、5211码）
    - 数字表示相应的权值
  - 无权码：
    - 余三码：每个字符编码比相应的8421码多3
    - 格雷码：任意两个相邻的代码只有一位二进制数不同
- 自定义数据表示

十进制	格雷码	十进制	格雷码
0	0000	8	1100
1	0001	9	1101
2	0011	10	1111
3	0010	11	1110
4	0110	12	1010
5	0111	13	1011
6	0101	14	1001
7	0100	15	1000

2.1.2 数的机器码表示

- 一、数的机器码表示

- 真值：一般书写的数
- 机器码：机器中表示的数, 要解决在计算机内部数的正、负符号和小数点运算问题
  - 原码
  - 反码
  - 补码
  - 移码

2.1.2 数的机器码表示

- ①原码表示法

- 定点小数x0.x1x2…xn

$$[x]_{原}=\begin{cases} x & 1>x\geq 0 \\ 1-x & 0\geq x>-1 \end{cases} \qquad 符号\begin{cases} 0，正 \\ 1，负数 \end{cases}$$

  - 有正0和负0之分
  - 范围$2^{-n}-1\sim 1-2^{-n}$

例：$x=+0.11001110$

$[x]_{原}=0.11001110 \qquad [-x]_{原}=1.11001110$

2.1.2 数的机器码表示

- ①原码表示法

- 定点整数X0X1X2...Xn

$$[x]_{原}=\begin{cases} x & 2^n > x \geq 0 \\ 2^n - x & 0 \geq x > -2^n \end{cases} \qquad 符号\begin{cases} 0，正数 \\ 1，负数 \end{cases}$$

说明：

  - 有正0和负0之分
  - 范围 $1 - 2^n \sim 2^n - 1$

- 例：x=+11001110

  $[x]_{原}=011001110 \qquad [-x]_{原}=111001110$

2.1.2 数的机器码表示

- ①原码表示法

- 原码特点：
  - 表示简单，易于同真值之间进行转换，实现乘除运算规则简单。
  - 进行加减运算十分麻烦。

2.1.2 数的机器码表示

- ②补码表示法

- 定义：正数的补码就是正数的本身，负数的补码是原负数加上模。

- 计算机运算受字长限制,属于有模运算.
  - 定点小数x0.x1x2.....xn溢出量为2，以2为模
  - 定点整数x0x1x2.....xn溢出量为2^{n+1}，以2^{n+1}为模

2.1.2 数的机器码表示

- ②补码表示法
- 定点小数x0.x1x2...xn

$$[x]_{补}=\begin{cases} x & 1>x\geq 0 \\ 2+x & 0>x>-1 \end{cases}$$

符号 $\begin{cases} 0，正数,0 \\ 1，负数 \end{cases}$

- 例：x= -0.1011
  - $[x]_{补}=10+x=10.0000-0.1011=1.0101$
  - y=-0.01111
  - $[y]_{补}=10+y=10.00000-0.01111=1.10001$

2.1.2 数的机器码表示

- ②补码表示法
- 定点整数x0x1x2…xn

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

符号 $\begin{cases} 0，正数,0 \\ 1，负数 \end{cases}$

- 例：x= -1011
  - $[x]_{补}$=100000+x= 100000-1011=10101

2.1.2 数的机器码表示

- ②补码表示法
- 补码性质
  - 高位表明正负
  - 正数补码，尾数与原码相同
  - 范围$-2^n \sim 2^n-1$(定点整数)
- 变相补码(双符号补码)
  - 为了防止溢出而设定

2.1.2 数的机器码表示

- ②补码表示法

- 最大的优点就是将减法运算转换成加法运算

  $[X]_{补}-[Y]_{补}=[X]_{补}+[-Y]_{补}$

  例如 $X=(11)_{10}=(1011)_2$ $Y=(5)_{10}=(0101)_2$

  已知字长n=5位

  $[X]_{补}-[Y]_{补}=[X]_{补}+[-Y]_{补}$

  $=01011+11011=100110=00110=(6)_{10}$

  注： 最高1位已经超过字长故应丢掉

- 无正零和负零之分

- 但是，在求补码还要减法,电路繁琐，下面的反码表示解决这个问题。

2.1.2 数的机器码表示

- ③反码表示法

- 定义：正数的表示与原、补码相同，负数的符号位为1，数值位是将原码的数值按位取反，就得到该数的反码表示

- 电路容易实现，触发器的输出有正负之分

2.1.2 数的机器码表示

- ③反码表示法

- 对尾数求反，它跟补码的区别在于末位少加一个1，所以可以推出反码的定义
  - 定点小数x0.x1x2…xn

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

X1=+0.1011011，[X1] 反 =0.1011011

X2= -0.1011011，[X2] 反 =1.0100100

2.1.2 数的机器码表示

- ③反码表示法

- [x]补=[x]反+2^{-n}
- 反码表示有正0和负0之分
- 上述公式解决了前边的问题（求补码还要减法）

2.1.2 数的机器码表示

- ④移码表示法

- 移码表示法（用在阶码中）
- 定点整数定义 $[x]_移=2^n+x \quad 2^n > x \geq -2^n$
- 00000000~11111111(-2^n~2^n-1)
- 例+1011111 原码为01011111

  补码为01011111 反码为01011111

  移码为 11011111

2.1.2 数的机器码表示

- ④移码表示法

- 例-1011111 原码为11011111

  补码为10100001 反码为10100000

  移码为：10000000-1011111=00100001
- 特点：移码和补码尾数相同，符号位相反
- 范围:$-2^n \sim 2^n-1$

2.1.2 数的机器码表示

- 例：以定点整数为例,用数轴形式说明原码、反码、补码表示范围和可能的数码组合情况。

- 补码10000000表示-128，而原码10000000表示-0

2.1.2 数的机器码表示

- 例：将十进制真值(- 127, - 1,0, + 1, + 127)列表表示成二进制数及原码、反码、补码、移码值。

真值x（十进制）	真值x（二进制）	[x]原	[x]反	[x]补	[x]移
-127	-01111111	11111111	10000000	10000001	00000001
-1	-00000001	10000001	11111110	11111111	01111111
		00000000	00000000		
0	00000000			00000000	10000000
		10000000	11111111		
+1	+00000001	00000001	00000001	00000001	10000001
+127	+01111111	01111111	01111111	01111111	11111111

2.1.2 数的机器码表示

- 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?

- (1)定点原码整数表示

  最大正数值 $=(2^{15}-1)_{10}=(+32767)_{10}$

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

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

2.1.2 数的机器码表示

- 例：假设由S,E,M三个域组成的一个32位二进制字所表示的非零规格化浮点数 x ,真值表示为（非IEEE754标准）：
  $x = (-1)^{s} \times (1.M) \times 2^{E-128}$
  问：它所表示的规格化的最大正数、最小正数、最大负数、最小负数是多少?
- (1)最大正数：0 1111 1111 111 1111 1111 1111 1111 1111
  $x = [1 + (1 - 2^{-23})] \times 2^{127}$
- (2)最小正数：000 000 000000 000 000 000 000 000 000 00
  $x = 1.0 \times 2^{-128}$
- (3)最小负数：111 111 111111 111 111 111 111 111 111 11
  $x = -[1 + (1 - 2^{-23})] \times 2^{127}$
- (4)最大负数100 000 000000 000 000 000 000 000 000 00
  $x = -1.0 \times 2^{-128}$

2.1.3 字符和字符串(非数值)的表示方法

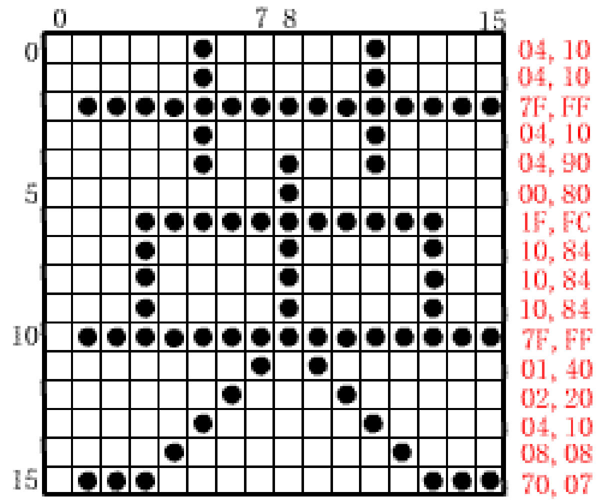
- 符号数据：字符信息用数据表示，如ASCII等；
- 字符表示方法ASCII:用一个字节来表示,低7位用来编码(128),最高位为校验位,参见教材P24表2.1
- 字符串的存放方法

2.1.4 汉字的存放

- 汉字的表示方法（一级汉字3755个，二级汉字3008个）
  - 输入码：数字编码、拼音码、字形编码
    - 国标码
      - ✓ 一级（16~55）*94：常用汉字计3755个，按汉语拼音字母/笔形顺序排列
      - ✓ 二级（56~87）*94：次常用汉字计3008个，置于56-87区，按部首/笔画顺序排列
      - ✓ 图形符号（682个）（01~09）*94
    - 拼音、五笔
  - 汉字内码：汉字信息的存储，交换和检索的机内代码，两个字节组成，每个字节高位都为1（区别于英文字符）

2.1.4 汉字的存放

- 汉字字模码：汉字字形
  - 点阵



  - 汉字库

2.1.5 校验码

- 校验码（只介绍奇偶校验码）
  - 引入：信息传输和处理过程中受到干扰和故障，容易出错。
  - 解决方法：是在有效信息中加入一些冗余信息（校验位）
  - 奇偶校验位定义
  - 设 $x = (x_0 x_1 \ldots x_{n-1})$是一个n位字,则奇校验位C定义为:$\bar{C} = x_0 \oplus x_1 \oplus \ldots \oplus x_{n-1}$,式中⊕代表按位加,表明只有当x中包含有奇数个1时,才使$\bar{C} = 1$,即$C = 0$。同理可以定义偶校验。
  - 只能检查出奇数位错；不能纠正错误。
  - p26例10自己看一下。
  - 其它还有Hamming,CRC

- 奇校验码，奇数个1使结果为0；偶校验码，偶数个1使结果为0

2.1.5 校验码

- 奇校验：所有传送的位(含字符的各位和校验位的1的个数为奇数.如:8位数据:0110101 1的个个数为偶数,我们加一个1,变为奇数,所以校验位为1.如:8位数据:0110001 1的个个数为奇数,我们加一个0仍为奇数,所以校验位为0.

- 偶校验：所有传送的位(含字符的各位和校验位的1的个数为偶数.如:8位数据:0110101 1的个个数为偶数,我们加一个0,仍为偶数,所以校验位为0.如:8位数据:0110001 1的个个数为奇数,我们加一个1,变为偶数,所以校验位为1.