浙江大学

《运算器集成实验报告》

队　名　四个赛艇

姓名与学号　杨璞　3140102308

陈铮　3150000101

李自乐　3140104024

张昌琳　3140103541（本次实验项目经理）

指导教师　楼老大

年级与专业　14 级　软件工程

目录

一、实验目的和要求 1
二、原理证明（算法推导） 2
1 整数运算 2
1.1 原码 2
1.2 补码 9
1.3 移码 12
1.4 帅码 14
2 浮点数运算 17
2.1 浮点数加减法 17
2.2 浮点数乘除法 18
三、软件流程图 19
1 通用加减法 19
2 通用乘除法 20
四、程序结构/函数调用关系 20
五、使用手册 21
1、运行环境 21
2、使用方法 21
3、输入输出 21
六、使用实例 22
1、整数运算 22
1.1 加法 22
1.2 减法 23
1.3 乘法 23
1.4 除法 24
2、浮点数运算 24
七、展示 PPT（见附件） 26

一、实验目的和要求

了解并掌握整数和浮点数在计算机中的表示方式和运算方法，并通过程序模拟来实现整数和浮点数的数转码、码转数、加减乘除四则运算。

具体如下：

1、整数运算

比较补码、原码、移码及帅码制的表示方法与四则算术运算算法，分析各种码制的优缺点。

基本要求实现六个函数：

word atom(char*)：字符串转换成对应的二进制。

char* mtoa(word)：二进制转换成字符串。
word madd(word,word)：二进制所表示数的加法。
word msub(word,word)：减法。
word mmul(word,word)：乘法。
word mdiv(word,word)：除法。
word mmod(word,word)：取余。
扩展功能：
位扩展(8-16、16-32 位)方法，溢出判断，大小比较。
2、浮点数运算
基本要求实现六个函数：
dwrd atom(char*)：字符串转换成对应的二进制。
char* mtoa(dwrd)：二进制转换成字符串。
dwrd madd(dwrd,dwrd)：二进制所表示数的加法。
dwrd msub(dwrd,dwrd)：减法。
dwrd mmul(dwrd,dwrd)：乘法。
dwrd mdiv(dwrd,dwrd)：除法。
dwrd mmod(dwrd,dwrd)：取余。
扩展功能：
特殊操作数正确返回结果，溢出判断，大小比较。

二、原理证明（算法推导）

1 整数运算

1.1 原码

1.1.1 数转码

根据原码的定义，当 x>=0 时 X=x，当 x<0 时 $X=2^{n-1}+|x|$。则 $X=2^{n-1}-x$，综上所述：

$$X=\begin{cases} x & (x\geq 0) \\ 2^{n-1}-x & (x<0) \end{cases}$$

。之后证明都在概念的基础上进行。

1.1.2 码转数

依据 1 给出的定义，码转数是数转码的逆运算。

由上可知当 X>2^{n-1}时 x<0，则带入下面的式子进行转化，而当 X<=2^{n-1}时 x>=0，则带入上面的式子进行计算，综上所述：$x=\begin{cases} X & (X \le 2^{n-1}) \\ 2^{n-1}-X & (X > 2^{n-1}) \end{cases}$。

1.1.3 加法运算

我们设加法的加数为 x,y，和为 z。加法运算要分四种情况考虑。其中涉及的绝对值大小的比较在后面大小比较的模块会详细论述。

（1）x>=0,y>=0

在这种情况下，z=x+y，由于所有数都是正数，所以原码与数相等。Z=X+Y,z=X+Y。这种情况可能会产生溢出，这在后续会有

（2）x<0,y<0

在这种情况下，z=x+y，x,y,z 都是负数，故这种情况存在溢出的可能性，我们在下面讨论溢出时会进一步说明。

$$
\begin{aligned}
&x = 2^{n-1} - X \\
&y = 2^{n-1} - Y \\
&z = 2^{n-1} - Z \\
&\because z = x + y \\
&\therefore 2^{n-1} - Z = 2^{n-1} - X + 2^{n-1} - Y \\
&\therefore Z = 2^{n-1} + X + Y \\
&\therefore z = -(X + Y)
\end{aligned}
$$

（3）x>=0,y<0

$$x = X$$
$$y = 2^{n-1} - Y$$
$$(1)\,|x| \geq |y|$$
$$\therefore z \geq 0$$
$$\therefore z = Z$$
$$\because z = x + y$$
$$\therefore Z = X + 2^{n-1} - Y$$
$$\therefore z = X + 2^{n-1} - Y$$
$$(2)\,|x| < |y|$$
$$\therefore z < 0$$
$$\therefore z = 2^{n-1} - Z$$
$$\because z = x + y$$
$$\therefore 2^{n-1} - Z = X + 2^{n-1} - Y$$
$$\therefore Z = Y - X$$
$$\therefore z = X + 2^{n-1} - Y$$

（4）x<0, y>=0

$$x = 2^{n-1} - X$$
$$y = Y$$
$$(1)\,|x| > |y|$$
$$\therefore z < 0$$
$$\therefore z = 2^{n-1} - Z$$
$$\because z = x + y$$
$$\therefore 2^{n-1} - Z = Y + 2^{n-1} - X$$
$$\therefore Z = X - Y$$
$$z = Y + 2^{n-1} - X$$
$$(2)\,|x| \leq |y|$$
$$\therefore z \geq 0$$
$$\therefore z = Z$$
$$\because z = x + y$$
$$\therefore Z = Y + 2^{n-1} - X$$
$$\therefore z = Y + 2^{n-1} - X$$

综上所述

$$Z=\begin{cases} X+Y & (x,y\geq 0) \\ 2^{n-1}+X+Y & (x,y<0) \\ 2^{n-1}+X-Y & (x\geq 0,y<0,|x|\geq|y|) \\ Y-X & (x\geq 0,y<0,|x|<|y|) \\ 2^{n-1}+Y-X & (x<0,y\geq 0,|x|\leq|y|) \\ X-Y & (x<0,y\geq 0,|x|>|y|) \end{cases}$$

$$z=\begin{cases} X+Y & (x,y\geq 0) \\ -(X+Y) & (x,y<0) \\ 2^{n-1}+X-Y & (x\geq 0,y<0) \\ 2^{n-1}+Y-X & (x<0,y\geq 0) \end{cases}$$

1.1.4 减法运算

减法运算本质上与加法运算是相同的，可以把减号看作是被减数带的符号我们设被减数与减数分别为 x, y，得到的差为 z。减法运算要分四种情况考虑。

（1）x>=0, y>=0

$$x=X$$
$$y=Y$$
$$(1)|x|\geq|y|$$
$$\therefore z\geq 0$$
$$\therefore z=Z$$
$$\because z=x-y$$
$$\therefore Z=X-Y$$
$$\therefore z=X-Y$$
$$(2)|x|<|y|$$
$$\therefore z<0$$
$$\therefore z=2^{n-1}-Z$$
$$\because z=x-y$$
$$\therefore 2^{n-1}-Z=X-Y$$
$$\therefore Z=2^{n-1}+Y-X$$
$$\therefore z=X-Y$$

（2）x<0, y<0

在这种情况下，z=x+y，x, y, z 都是负数，故

$$x = 2^{n-1} - X$$
$$y = 2^{n-1} - Y$$
$$(1)|x|>|y|$$
$$\therefore z < 0$$
$$\therefore z = 2^{n-1} - Z$$
$$\because z = x - y$$
$$\therefore Z = X + 2^{n-1} - Y$$
$$z = Y - X$$
$$(2)|x| \leq |y|$$
$$\therefore z \geq 0$$
$$\therefore z = Z$$
$$\because z = x - y$$
$$\therefore Z = Y - X$$
$$\therefore z = Y - X$$

（3）x>=0, y<0

在这种情况下，由于 z=x-y，则 z>0，这种情况有可能溢出，详见下面对溢出的讨论。

$$x = X$$
$$y = 2^{n-1} - Y$$
$$z = Z$$
$$\because z = x - y$$
$$\therefore Z = X - 2^{n-1} + Y$$
$$\therefore Z = X + Y - 2^{n-1}$$
$$\therefore z = X + Y - 2^{n-1}$$

（4）x<0, y>=0

在这种情况下，由于 z=x-y，则 z<0，这种情况有可能溢出，详见下面对溢出的讨论。

$$x = 2^{n-1} - X$$
$$y = Y$$
$$z = 2^{n-1} - Z$$
$$\because z = x - y$$
$$\therefore 2^{n-1} - Z = 2^{n-1} - X - Y$$
$$\therefore Z = X + Y$$
$$\therefore z = 2^{n-1} - X - Y$$

综上所述

$$Z=\begin{cases} X-Y & (x,y\geq 0,|x|\geq|y|) \\ 2^{n-1}+Y-X & (x,y\geq 0,|x|<|y|) \\ 2^{n-1}+X-Y & (x,y<0,|x|>|y|) \\ Y-X & (x,y<0,|x|\leq|y|) \\ X+Y-2^{n-1} & (x\geq 0,y<0,|x|\geq|y|) \\ X+Y & (x<0,y\geq 0,|x|>|y|) \end{cases}$$

$$z=\begin{cases} X-Y & (x,y\geq 0) \\ Y-X & (x,y<0) \\ X+Y-2^{n-1} & (x\geq 0,y<0) \\ 2^{n-1}-X-Y & (x<0,y\geq 0) \end{cases}$$

1.1.5 乘法运算

原码的乘法的运算的核心在于绝对值的乘法，我们设两个乘数分别为 x,y，得到的结果为 z。原码的计算大体分三步。

（1）取绝对值

如果 x,y 可以通过“与”上 7FFF 的操作得到绝对值（这里认为原码是 16 位码）。并且通过 x,y 正负来判断 z 的正负，通过 flag 记录 z 的正负。

（2）绝对值的乘法操作

绝对值的乘法操作是通过移位和加法两种操作共同完成的。

$$\begin{aligned} &设|X|=a_0*2^0+a_1*2^1+\ldots\ldots+a_{n-1}*2^{n-1} \\ &|Y|=b_0*2^0+b_1*2^1+\ldots\ldots+b_{n-1}*2^{n-1} \\ &|Z|=|X|*|Y|=|X|*b_0*2^0+|X|*b_1*2^1+\ldots\ldots+|X|*b_{n-1}*2^{n-1} \\ &=|X|*b_0+|X|*b_1<<1+\ldots\ldots+|X|*b_{n-1}<<n-1 \end{aligned}$$

这些步骤都是可以通过循环实现的。

（3）得到最终结果 Z

通过第一步对符号的判断得到 Z 的最终结果，可以通过之前提到的码转数将 Z 转化为 z 进行输出。

1.1.6 除法运算以及取余操作

我们设被除数与除数分别为 x,y，得到的商为 z，余数为 w。

$$\because x = z * y + w$$
$$(1) x \geq 0, y > 0$$
$$\text{则} z \geq 0$$
$$w = x - z * y = |x| - |z| * |y|$$
$$(2) x < 0, y > 0$$
$$\text{则} z < 0$$
$$w = x - z * y = -|x| + |z| * |y| = -(|x| - |z| * |y|)$$
$$(3) x \geq 0, y < 0$$
$$\text{则} z < 0$$
$$w = x - z * y = |x| - |z| * |y|$$
$$(4) x < 0, y < 0$$
$$\text{则} z > 0$$
$$w = x - z * y = -|x| + |z| * |y| = -(|x| - |z| * |y|)$$
$$\text{故} |w| = |x| - |z| * |y|, \text{并且与} x \text{一致}$$

故原码的除的运算的核心在于绝对值的除法，则原码除法计算大体分三步。

（1）取绝对值

如果 x,y 可以通过“与”上 7FFF 的操作得到绝对值（这里认为原码是 16 位码），记 x,y,z,w 的绝对值的原码分别为 X,Y,Z,W，由于都是正数所以数与码相等。并且通过 x,y 正负来判断 z 的正负以及 w 的正负。

（2）绝对值的除法操作

绝对值的除法操作是通过多次减法操作共同完成的。 根据余数与除法的定义，先比较 X 与 Y 的大小，若 X<Y，则运算结束，Z 为 0，W=X。若不是我们可以用 X 减去 Y，若得到的结果的绝对值小于 Y 则除法运算结束，若比 Y 大，则用该结果减去 Y，如此循环直到得到的结果的绝对值小于 Y。做减法的次数即为 Z 的值，而最终结果就是 W。

（3）得到最终结果 z，w

通过第一步对符号的判断得到 Z 的最终结果，可以通过之前提到的码转数将 Z 转化为 z 进行输出。

1.1.7 溢出

在我们上述运算中有三种情况可能造成溢出，一是同号相加，二是异号相减，三是乘法运算。

（1）同号相加：

判断是否溢出只需要通过判断得到的和是否与加数同号即可，若不同号（即最高位的数不同）即为溢

出

（2）异号相减：

减法是加法的逆运算，判断是否溢出只要通过判断得到的差是否与被减数同号即可，若不同号（即最高位的数不同）即为溢出。

（3）乘法操作：

在进行乘法操作时需在绝对值运算时即进行溢出判断。若两个绝对值的积为负数（即最高位为 1）即为溢出。

1.1.8 扩展

（1）8 位到 16 位：

原码的扩展即在符号位与表示数值的位的第一个‘1’之间添 0。具体操作可以是再取一个 8 位的数来表示原数的最高位。先判断数值正负，若为正则不变，若为负，则在原码取绝对值后其高八位与 0x001 取“或”操作。

（2）16 位到 32 位：

这个操作与（1）类似，也是再取一个 16 位的数来表示原数的最高位，先判断数值正负，若正则不变，若为负，则在原码取绝对值后与 0x001 取“或”操作。

1.1.9 大小比较

原码的大小比较首先是符号位的比较，然后是绝对值的比较，通过符号位得知正负，若同正或同负再进行绝对值的比较。绝对值的比较则可以通过移位来实现，通过移位找到两个数不同的位数，以高位为准，然后再通过之前由符号位得到的正负判断两个数的大小。

1.2 补码

1.2.1 数码转换

atom 函数与 mtoa 函数在数学上互逆的，即

(mtoa◦atom)(x)≡x,x 是数(atom◦mtoa)(x)≡x,x 是码

然而在计算机中受到机器精度限制，并不能好好地做运算。

考虑到这点，我们可以给函数加上定义域与值域的限制。

特别地，对于 32 位补码，定义：

atom:[−231,231)→[0,232)

mtoa:[0,232)→[−231,231)

atom(x)={x,232+x,0≤x<231−231≤x<0

构成双射函数，于是求这个函数的反函数得到：

mtoa(x)={x,−232+x,0≤x<231231≤x<232

atom，mtoa 的定义是等价的，都可以称为补码的定义

接下来看一下在 0-1 串内部的一些位运算：

二进制码按位取反函数的代数代换

设有 0-1 串

m=(m1m2...mn)2∈[0,2n)

则有 Not(m)=(Not(m1)Not(m2)...Not(mn))2

显然有 m+Not(m)=(11...1)2=2n−1

于是 Not(m)=2n−1−m∈[0,2n)

按位取反运算：Not(m)在[0,2n) 中封闭。

补码：取相反数的算法证明

设 m≠0 是一个 n 位的 0-1 串，证明：mtoa(Not(m)+1)=−mtoa(m)。

证:

代入按位取反公式：

mtoa(Not(m)+1)=mtoa(2n−1−m+1)

简单化简：

mtoa(2n−1−m+1)=mtoa(2n−m)

（1）若 m∈(0,2n−1)→2n−m∈(2n−1,2n)

根据补码的定义

−mtoa(m)=−mmtoa(2n−m)=−2n+(2n−m)=−m

所以：

mtoa(2n−m)=−mtoa(m)

（2）若 m∈[2n−1,2n)→2n−m∈(0,2n−1]

根据补码的定义

−mtoa(m)=−(−2n+m)=2n−mmtoa(2n−m)=2n−m

所以：

mtoa(2n−m)=−mtoa(m)

综上所述，

mtoa(Not(m)+1)=−mtoa(m)

证毕。

所以，非 0 数的补码按位取反再加一即为其相反数的补码。

特别地，0 的相反数的补码是其本身，这个可以单独验证。

1.2.2 补码加法

对于 n 位 0-1 串 m1,m2，证明：

mtoa(m1)+mtoa(m2)≡mtoa((m1+m2)mod2n)(mod2n)。

引理 2：mtoa(x)≡x(mod2n),x∈[0,2n)

证明：

1. 若 x∈[0,2n−1)

mtoa(x)=x≡x(mod2n)

2. 若 x∈[2n−1,2n)

mtoa(x)=−2n+x≡x(mod2n)

证毕。

证：

由引理 2：

mtoa(m1)≡m1(mod2n)mtoa(m2)≡m2(mod2n)mtoa((m1+m2)mod2n)≡m1+m2(mod2n)

由同余定理得

mtoa(m1)+mtoa(m2)≡m1+m2(mod2n)

所以，

mtoa(m1)+mtoa(m2)≡mtoa((m1+m2)mod2n)(mod2n)

证毕。

这个部分证明了：两数的和的补码与两数的补码的和(高位丢弃)对 2n 同余。

1.2.3 补码减法

对于 n 位 0-1 串 m1,m2，证明 mtoa(m1)−mtoa(m2)≡mtoa((m1−m2)mod2n)(mod2n)。

证：

由引理 2：

mtoa(m1)≡m1(mod2n)mtoa(m2)≡m2(mod2n)mtoa((m1−m2)mod2n)≡m1−m2(mod2n)

由同余定理：

mtoa(m1)−mtoa(m2)≡m1−m2(mod2n)

所以，

mtoa(m1)−mtoa(m2)≡mtoa((m1−m2)mod2n)(mod2n)

这个部分证明了：两数的差的补码与两数的补码的差对 2n 同余。

1.2.4 补码乘法

这个部分的数学部分就比较麻烦了，二进制串(向量)乘法本质是一个离散卷积。关于其算法，普通可以使用 O(n2)的朴素做法，也可以用 FFT1 算法进行优化到 O(nlog(n))。

1.3 移码

1.3.1 数码转换

X 为纯整数，他的移码为 2n-1 + X

范围：[-2n-1,2n-1)

整数转化为移码：

X≧0: X[移] = 2n-1 + |x|

X < 0: X[移] = 2n-1 - |x|

溢出判断：

X≧0: |x| ≧2n-1 溢出

X < 0: |x| > 2n-1 溢出

1.3.2 移码加法

同号：

X, Y ≧ 0：

X[移]去除最高位为 x，Y[移]去除最高位为 y，如果 x+y ≧ 2n-1 则溢出，否则(X+Y)[移] = x+y + 2n-1 相当于 X[移] + Y[移] - 2n-1 ，因为 X,Y 都为正， X[移]， Y[移]大于 2n-1

X, Y < 0：

如果 X[移] + Y[移] < 2n-1, 则 2n-1 - |X |+ 2n-1 -|Y |= 2n -|X |-|Y |< 2n-1, |X| + |Y|>2n-1,溢出

否则，(X+Y)[移] = X[移]+Y[移] - 2n-1 = 2n-1 - |X| - |Y|

异号：

不会溢出：

X[移]去除最高位为 x， Y[移]去除最高位为 y，(X+Y)[移] = x+y, 因为 X, Y 中有且仅有一个>2n-1

1.3.3 移码减法

加相反数

移码取相反数：

X[移] = 0：溢出

否则， (-X)[移] = 2n - X[移] = 2n-1 + |X|(X < 0) = 2n-1 - |X|(X≧0)

1.3.4 比较大小

直接比较

1.4 帅码

1.4.1 帅码简介

余三码是一种 BCD 码，它是由 8421 码加 3 后形成的（即余三码是在 8421 码基础上每位十进制数 BCD 码再加上二进制数 0011 得到的）。因为 8421 码中无 1010～1111 这 6 个代码，所以余三码中无 0000～0010、1101～1111 这 6 个代码。余三码不具有有权性，但具有自补性，余三码是一种“对 9 的自补码”。

帅码在余三码的基础上使用最高四位表示符号，若最高四位为 0000 则为正数，若最高四位为 1000 则为负数。

码表如下：

十进制数	8421 码	余三码
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

1.4.2 数码转换

1、将十进制字符串 … 每一位分别转化为对应的余三码（即每一位的原码加三）。

2、读入低位时，为了将之前读入的码置于高位，需要将之前读入的码左移四位。

3、不足七位的十进制字符串需要在高位补 0（0 的余三码为 0011）。

4、根据是否有负号，将最高位置为 0000 或者 1000。

例如：

123 的帅码为 0x03333456

-123 的帅码为 0x73333456

1.4.3 码数转换

1、先取出帅码的最高四位判断数的符号，若为负号输出负号。

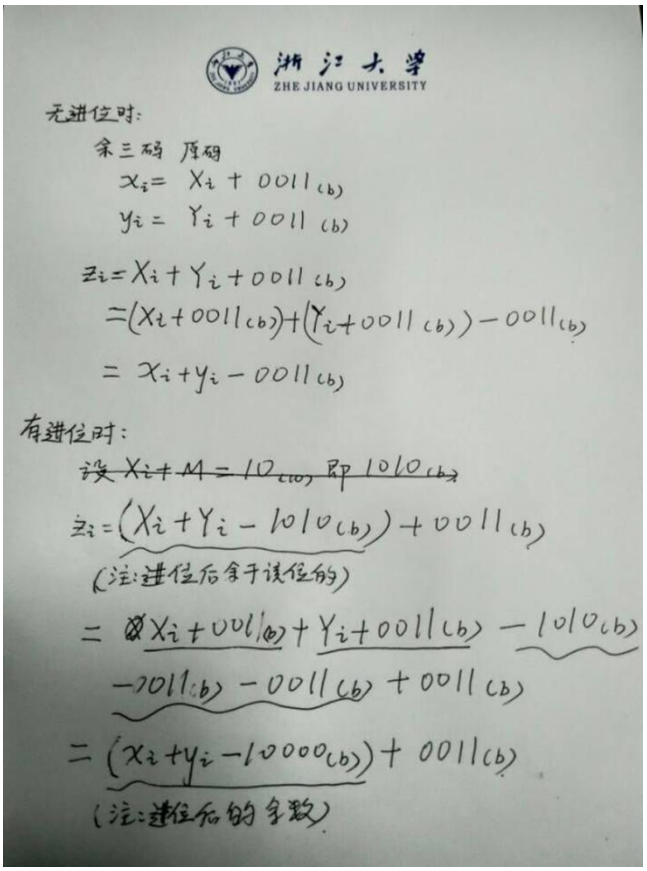
2、将每四位二进制转为十进制，并从第一个非零数开始输出对应字符。

1.4.4 加法

这里把加法看作是同号相加。（异号相减转化为同号相加）

每一位用其对应的原码做加法，再依如下规则进行修正：

如有进位，该位加 3；如无进位，该位减 3。



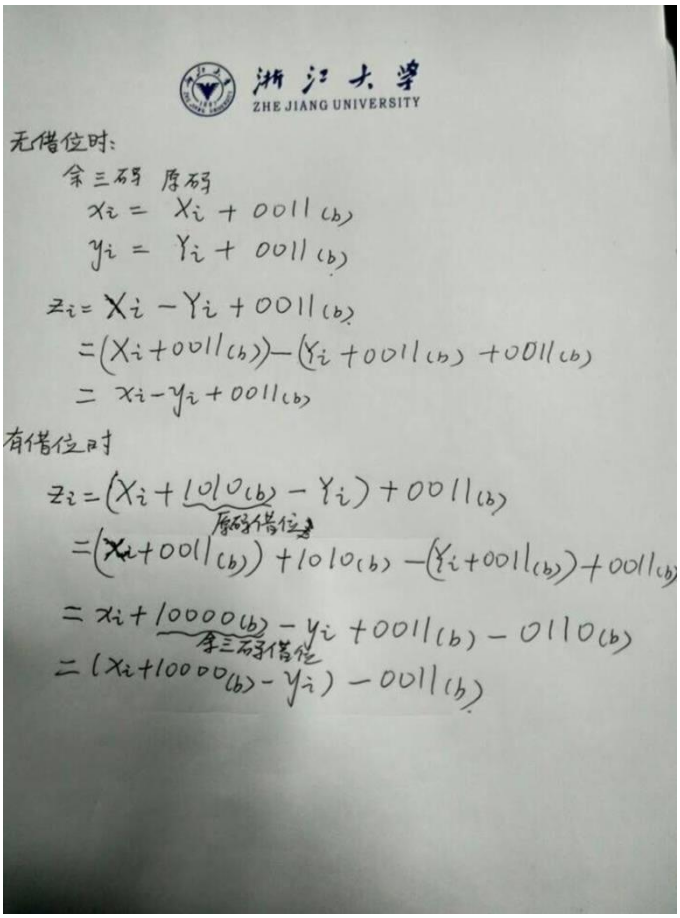
1.4.5 减法

这里把减法看作是同号相减。（异号相加转化为同号相减）

方法：

每一位用其对应的原码做减法，再依如下规则进行修正：

如有借位，该位减 3；如无借位，该位加 3。



1.4.6 大小比较

1、判断两数符号，若异号，正数大于负数

2、若同号，两数相减，判断结果正负。若为正，被减数大；若为负，减数大；若为 0，两数一样大。

1.4.7 溢出判断

判断 32 位二进制码中除去符号位四位的 28 位运算是否产生进位，若有进位，则溢出。

1.4.8 位扩展(8-16、16-32 位)

符号位之后补零（余三码为 0011）

8 位扩展到 16 位

$a_1\ a_2\ a_3\ a_4\ \ a_5\ a_6\ a_7\ a_8 \Rightarrow$

$a_1\ a_2\ a_3\ a_4\ $ 0011 0011 $\ a_5\ a_6\ a_7\ a_8$

16 位扩展到 32 位

$a_1\ a_2\ a_3\ a_4\ \ a_5\ a_6\ a_7\ a_8 \cdots a_{13}\ a_{14}\ a_{15}\ a_{16} \Rightarrow$

$a_1\ a_2\ a_3\ a_4\ $ 0011 0011 0011 0011 $\ a_5\ a_6\ a_7\ a_8 \cdots a_{29}\ a_{30}\ a_{31}\ a_{32}$

2 浮点数运算

2.1 浮点数加减法

1、对 0、Infinity 和 NaN 操作数作检查

若有一个操作数为 NaN 则直接返回 NaN;

若有一个操作数为 0 则直接返回另一个操作数;

若有一个操作数为 Infinity，

若另一个操作数也是 Infinity 且符号相同则返回第一个操作数;

若另一个操作数也是 Infinity 且符号不同则返回 NaN;

若其他情况则返回 Infinity。

2、对阶，若两操作数的阶码不相等，则对阶（小数点对齐）。

由于尾数右移所损失的精度比左移的小，因此小阶向大阶看齐。

3、符号位+尾数（包含隐藏位）执行加减法。

按有符号整数加减法进行运算，并采用双符号法判断是否溢出。

4、规格化。

向右规格化：若步骤 3 出现溢出，则尾数右移 1 位，阶码+1;

向左规格化：若步骤 3 没有出现溢出，且数值域最高位与符号位数值相同，则尾数左移 1 位且阶码-1，直到数值域最高位为 1 为止。

5、舍入处理

6、溢出判断

由于规格化时可能会导致阶码发生溢出

若无发生溢出，则运算正常结束。

若发生上溢出，则返回 Infinity。

若发生下溢出，则返回 0。

2.2 浮点数乘除法

1、对 0、Infinity 和 NaN 操作数作检查

乘法：

其中一个为 0 则结果一定是 0;

否则，一个数为非数即结果为非数;

否则，一个数为无穷大即结果为无穷大。

除法：

一个数为非数即结果为非数；

否则，除数为 0 则结果为非数；

否则，被除数为 0 则结果为 0；

否则，被除数、除数均为无穷大，结果为非数；

否则，被除数为无穷大即结果为无穷大。

2、阶码相加减

按照定点整数的加减法运算方法对两个浮点数的阶码进行加减运算。

3、尾数相乘或相除

按照定点小数的阵列乘除法运算方法对两个浮点数的尾数进行乘除运算。为了保证尾数相除时商的正确性，必须保证被除数尾数的绝对值一定小于除数尾数的绝对值。若被除数尾数的绝对值大于除数尾数的绝对值，需对被除数进行调整，即被除数的尾数每右移 1 位，阶码加 1，直到被除数尾数的绝对值小于除数尾数的绝对值。

4、规格化

向右规格化：若步骤 3 出现溢出，则尾数右移 1 位，阶码+1;

向左规格化：若步骤 3 没有出现溢出，且数值域最高位与符号位数值相同，则尾数左移 1 位且阶码-1，直到数值域最高位为 1 为止。

5、舍入处理

6、溢出判断

由于规格化时可能会导致阶码发生溢出

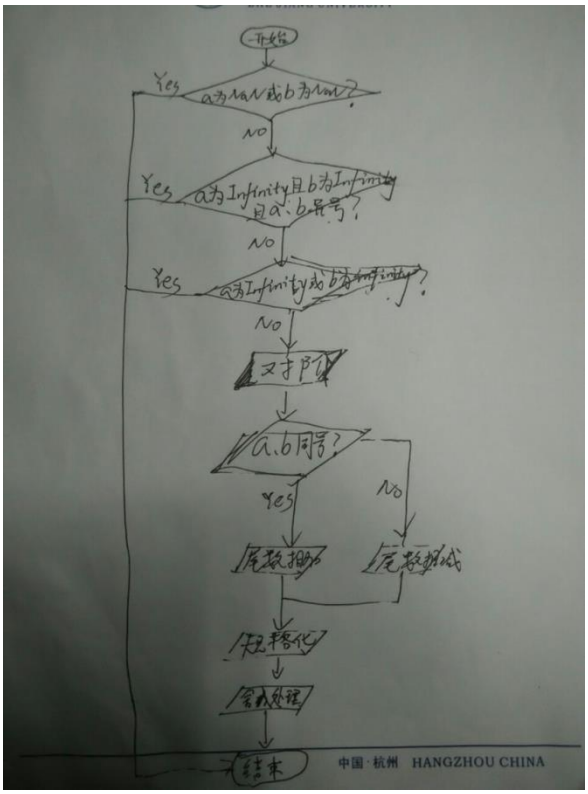
若无发生溢出，则运算正常结束。

若发生上溢出，则返回 Infinity。

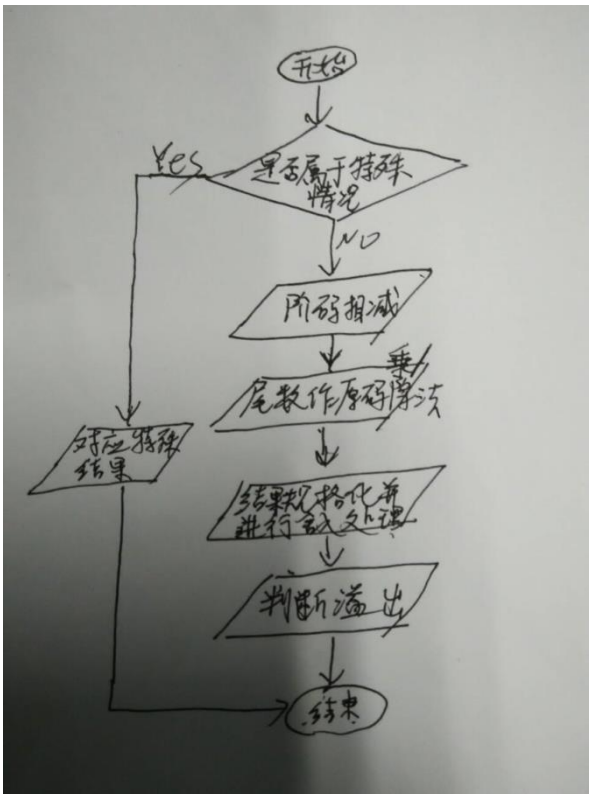
若发生下溢出，则返回 0。

三、软件流程图

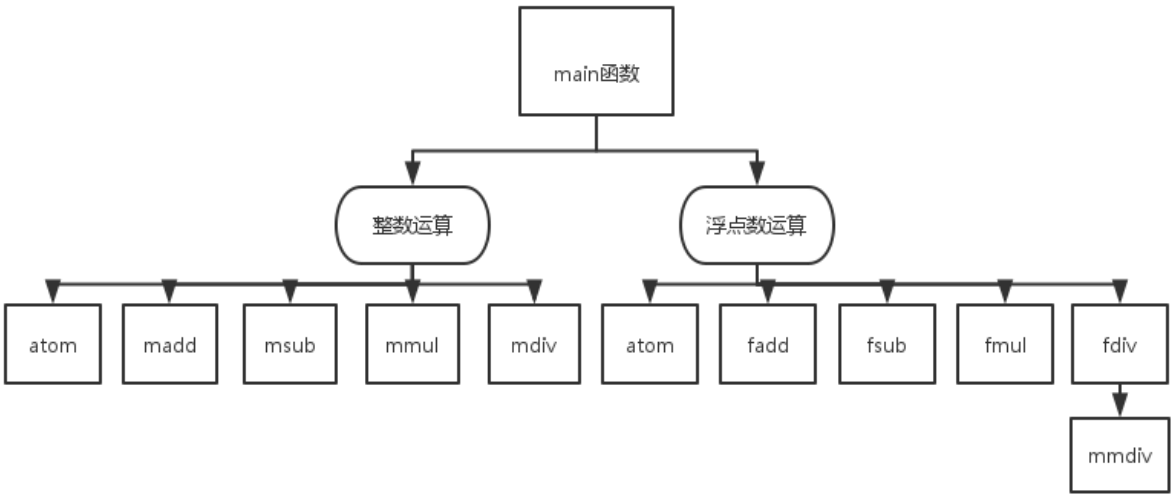
1 通用加减法



2 通用乘除法



四、程序结构/函数调用关系



五、使用手册

1、运行环境

PC 及其兼容机上。

2、使用方法

点击运行“运算器.exe”文件。

3、输入输出

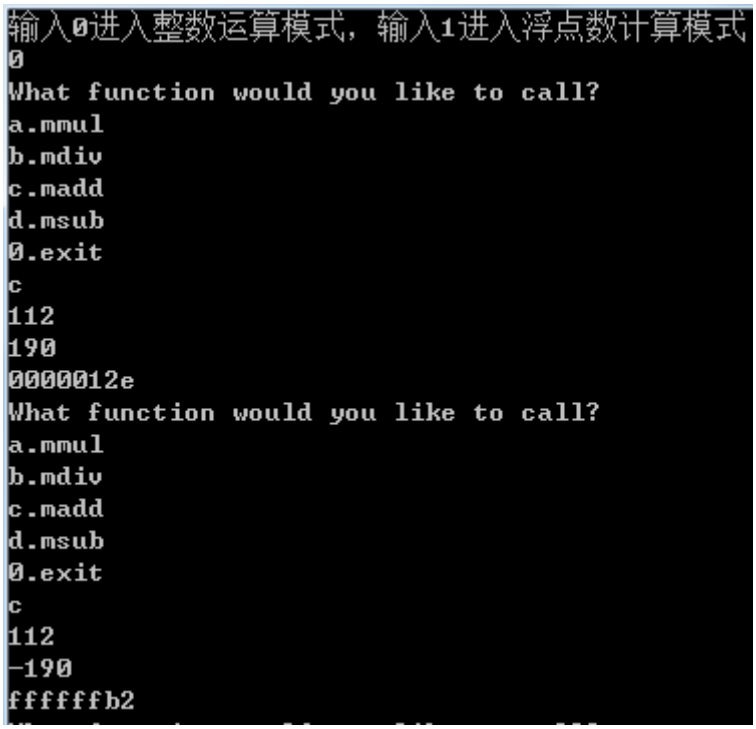
输入	功能
0/1	选择整数运算模式/浮点数运算模式
a/b/c/d	选择进行的整数运算加/减/乘/除
x,y	输入操作数

	输出
整数运算	
浮点数运算	

六、使用实例

1、整数运算

1.1 加法



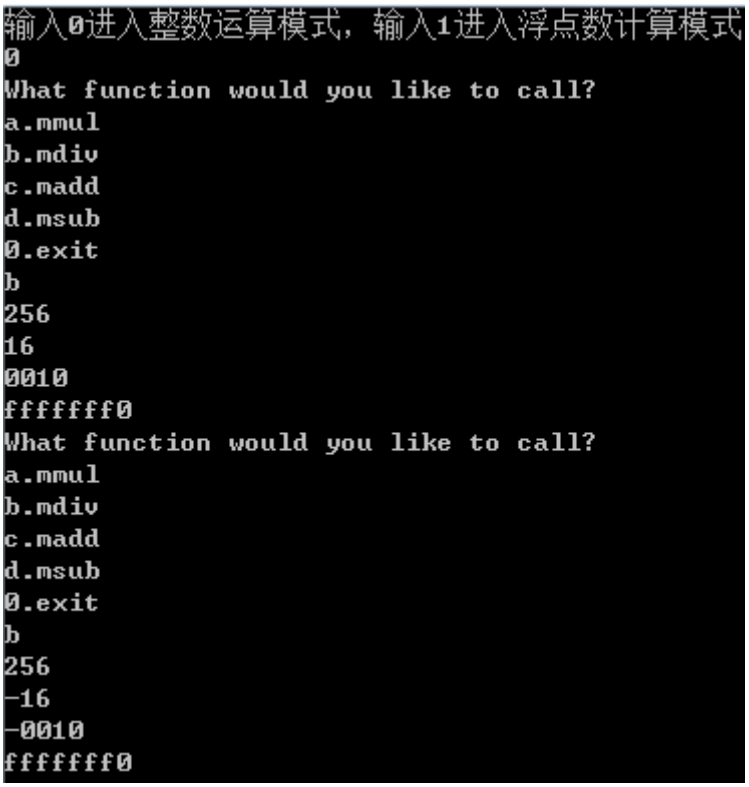
1.2 减法

```
输入0进入整数运算模式，输入1进入浮点数计算模式
0
What function would you like to call?
a.mmul
b.mdiv
c.madd
d.msub
0.exit
d
180
12
000000a8
What function would you like to call?
a.mmul
b.mdiv
c.madd
d.msub
0.exit
d
180
-12
000000c0
```

1.3 乘法

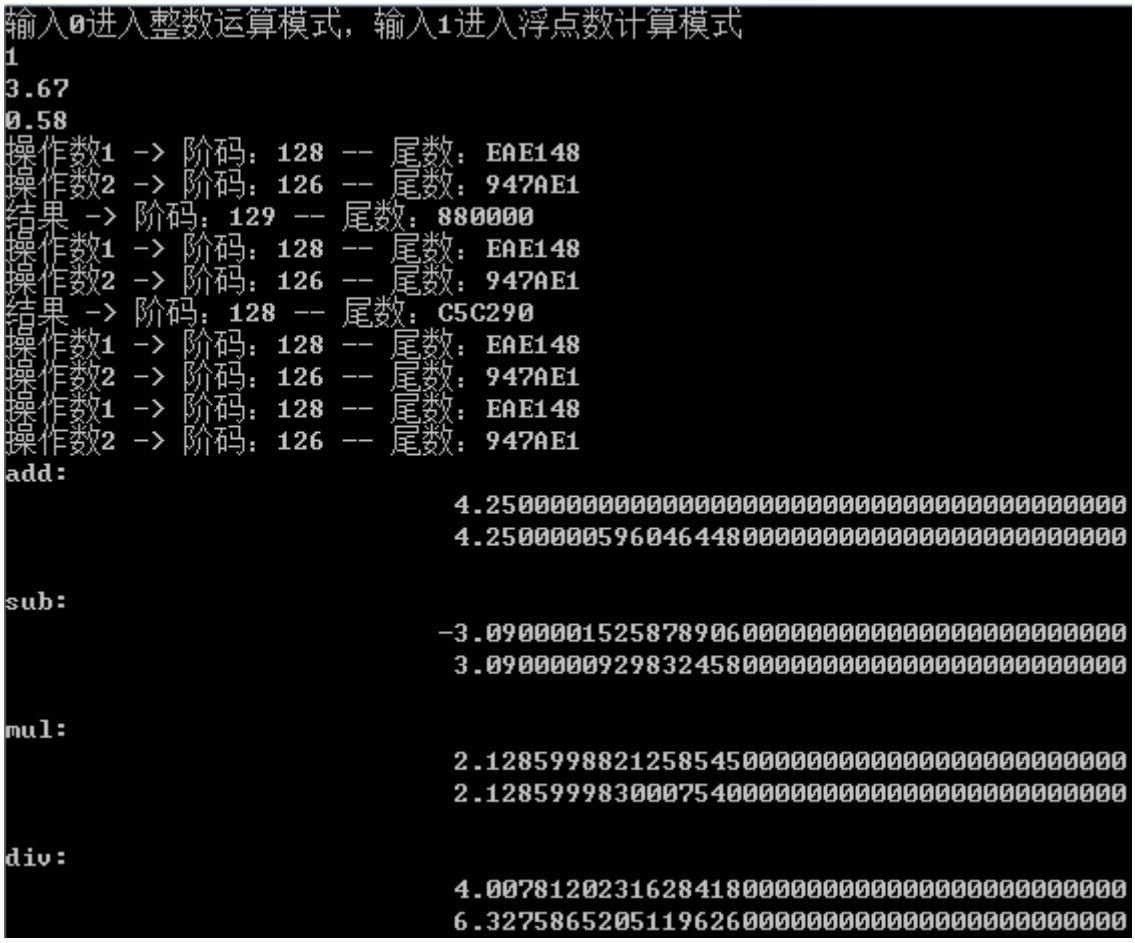
```
输入0进入整数运算模式，输入1进入浮点数计算模式
0
What function would you like to call?
a.mmul
b.mdiv
c.madd
d.msub
0.exit
a
13
14
00000000 000000b6
What function would you like to call?
a.mmul
b.mdiv
c.madd
d.msub
0.exit
a
13
-14
ffffffff ffffff4a
```

1.4 除法



2、浮点数运算

输出两个操作数加减乘除的四个结果：
示例 1


输入0进入整数运算模式，输入1进入浮点数计算模式
1
3.67
0.58
操作数1 -> 阶码：128 -- 尾数：EAE148
操作数2 -> 阶码：126 -- 尾数：947AE1
结果 -> 阶码：129 -- 尾数：880000
操作数1 -> 阶码：128 -- 尾数：EAE148
操作数2 -> 阶码：126 -- 尾数：947AE1
结果 -> 阶码：128 -- 尾数：C5C290
操作数1 -> 阶码：128 -- 尾数：EAE148
操作数2 -> 阶码：126 -- 尾数：947AE1
操作数1 -> 阶码：128 -- 尾数：EAE148
操作数2 -> 阶码：126 -- 尾数：947AE1
add:
4.250000000000000000000000000000000000000000
4.250000059604644800000000000000000000000000
sub:
-3.090000152587890600000000000000000000000000
3.090000092983245800000000000000000000000000
mul:
2.128599882125854500000000000000000000000000
2.128599983000754000000000000000000000000000
div:
4.007812023162841800000000000000000000000000
6.327586520511962600000000000000000000000000

示例 2

```
输入0进入整数运算模式，输入1进入浮点数计算模式
1
3.67
-0.58
操作数1 -> 阶码：128 -- 尾数：EAE148
操作数2 -> 阶码：126 -- 尾数：947AE1
结果 -> 阶码：128 -- 尾数：C5C290
操作数1 -> 阶码：128 -- 尾数：EAE148
操作数2 -> 阶码：126 -- 尾数：947AE1
结果 -> 阶码：129 -- 尾数：880000
操作数1 -> 阶码：128 -- 尾数：EAE148
操作数2 -> 阶码：126 -- 尾数：947AE1
操作数1 -> 阶码：128 -- 尾数：EAE148
操作数2 -> 阶码：126 -- 尾数：947AE1
add:
                          -3.090000152587890600000000000000000000000000
                           3.090000092983245800000000000000000000000000

sub:
                           4.250000000000000000000000000000000000000000
                           4.250000059604644800000000000000000000000000

mul:
                          -2.128599882125854500000000000000000000000000
                          -2.128599983000754000000000000000000000000000

div:
                          -4.007812023162841800000000000000000000000000
                          -6.327586520511962600000000000000000000000000
```

七、展示 PPT（见附件）