9、

1. FFFF8000H
2. 020AH
3. 0000FFFAH
4. 40H
5. BF8CCCCDH
6. 4025000000000000H

10、

（1）-65530

（2）-8196

（3）2^32-6

（4）字符“\*”

（5）-800

（6）-10.25

17、

因为小端方式，低位放低端，因此数值左端为低位

（1）440

（2）20

（3）-424

（4）-396

（5）68

（6）-312

（7）16

（8）12

（9）-276

（10）32

21、

M=15

N=4

24、

4098=+1 0000 0000 0010=1.0000 0000 001\*2^12

所以32位补码：00001002H

单精度浮点：45801000H

重合：0000 0000 0010，因为这是浮点数除隐藏位外的有效数字

28、

X=BE00 0000H

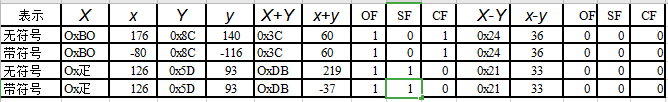
Y=40F0 0000H

i=0064H

大端：BE 00 00 00 40 F0 00 00 00 64

小端：00 00 00 BE 00 00 F0 40 64 00

29、如下图所示



31、

当arraysize超过32位能表达的值，使用malloc函数仍会有溢出，因为malloc函数是32位的，因此需要先判断arraysize是否在32位以内，若超过32位，返回-1，否则再申请空间进行复制。

通过强制转换来判断是否超过32位，如果超过32位，会发生截断，因此不会等于原来的值。

34、

(2)(x-l<0）||x>0

非永真。当x=-2147483648时,显然,x<0,机器数为8000000H,x-1的机器数为7FFFFFFFH(这是因为在运算器上不去分有符号还是无符号,直接计算),符号位为0,因而x-1>0。此时,

(x-1<0)和 x>0两者都不成立。

（4）x>0|-x>=0

非永真。当x-214748364S时,x<0,且x和-x的机器数都为8000000H,即-x<0。此时,x>0|和-x>=O 两者都不成立

（6）(x>y)=(-x<-y)

非永真。当x=-2147483648、y任意（除-2147483648外),或者 y=-2147483648、x任意(除-2147483 648外)时不等。因为int型负数-2147483648是最小负数,该数取负后结果仍为-2147483648（机器数）,而不是2147483648。所以仍是最小负数,和任意其它数比还是小。

8)(int)(ux-uy)==-(y-x)

永真。(int)(ux-uy）=[x-y]=[x]+[-y]=[-y+x]=[-(y-x)]

(10）x\*4+y\*8==(x<<2)+(y<<3)

|永真。因为带符号整数x乘以2完全等于x左移k位,无论结果是否溢出。

(12）x\*y=ux\*uy

永真。根据第2.7.5节P65页内容可知,x\*y的低32位和 ux\*uy 的低 32位是完全一样的位序列。

(14）x\*~y+ux\*uy==-x

永真。这里~y 是按位取反,不是负运算。所以,[-y=~y+1],即[~y=-y-1]。而根据(12)

X\*y=ux\*uy,所以x\*~y+ux\*uy==-x

35、

（1）永真，因为运算不会影响符号位

（3）非永真，x+y会溢出，而dx+dy不会

（5）非永真，浮点乘法有舍入，所以不满足交换律

36、

(1) 对于结果为±1x .xx……x的情况，需要进行右规。右规时，尾数右移一位，阶码加1。右规操作可以表示为：M b<-M b ×2 -1，Eb<-Eb+1。

(2) 对于结果为±0.00……01x……x的情况，需要进行左规。左规时，数值位逐次左移，阶码逐次减1，直到将第一位“1”移到小数点左边。假定k为结果中“±”和左边第一个1之间连续0的个数，则左规操作可以表示为：M b<-M b ×2k，Eb<-Eb–k。

39、

[x]浮 = 0 01111110 10...0 ：阶码E x= 01111110，尾数Mx= 0(1). 1...0

[y]浮 = 1 10000101 000001010...0 ：阶码 Ey= 10000101，尾数 My =1 (1).000001010...0

（1）0.75+ (– 65.25）

① 对阶：[ΔE]补=E x+ [–E y]补 (mod 2n) = 0111 1110 + 0111 1011 = 1111 1001，即ΔE = –7。

根据对阶规则可知需对 x 进行对阶，结果为：Ex = E y = 10000101 Mx = 00.000000110...000

② 尾数相加：Mb = Mx + My = 00.000000110...000+ 11.000001010 ...000 （注意小数点在隐藏位后） 根据原码加/减法运算规则，得：00.000000110...000+ 11.000001010...000 = 11.000000100…000

③ 规格化：尾数数值部分最高位为 1，因此不需要进行规格化。

④ 舍入：把结果的尾数 Mb 中最后两位附加位舍入掉，从本例来看，不管采用什么舍入法，结果都一样，都是把最后两个 0 去掉，得：Mb = 11.000000100…0

⑤ 溢出判断：在上述阶码计算和调整过程中，没有发生“阶码上溢”和“阶码下溢”的问题。因此，阶码Eb = 10000101。

最后结果为Eb = 10000101，Mb = 1(1).00000010…0，即：– 64.5。

（2） 0.75–(– 65.25）

① 对阶：[ΔE]补 = E x + [–E y]补 (mod 2n) = 0111 1110 + 0111 1011 = 1111 1001，ΔE = -7。

根据对阶规则可知需要对x 进行对阶，结果为：Ex = E y = 10000110，Mx = 00.000000110...000

② 尾数相加：Mb = Mx – My = 00.000000110...000 – 11.000001010...000 （注意小数点在隐藏位后） 根据原码加/减法运算规则，得：00.000000110...000 – 11.000001010...000=01.00001000…000

③ 规格化：尾数数值部分最高位为 1，不需要进行规格化。

④ 舍入：把结果的尾数 Mb 中最后两位附加位舍入掉，从本例来看，不管采用什么舍入法，结果都一样，都是把最后两个 0 去掉，得：Mb = 01.00001000…0

⑤ 溢出判断：在上述阶码计算和调整过程中，没有发生“阶码上溢”和“阶码下溢”的问题。因此，阶码Eb = 10000101。

最后结果为 Eb = 10000101，Mb = 0(1).00001000…0，即：+66。