**191220154 张涵之 第3章作业**

1. C1 = A1·B1 + A1·C0 + B1·C0

C2 = A2·B2 + A2·C1 + B2·C1

C3 = A3·B3 + A3·C2 + B3·C2

C4 = A4·B4 + A4·C3 + B4·C3

1. 10的原码和补码均为001010，-6的原码为100110，补码111010，6均为000110

（1）[x + y]补 = [10]补 + [-6]补 (mod 26) = 001010 + 111010 = 000100，真值为4

[x – y]补 = [10]补 + [6]补 (mod 26) = 001010 + 000110 = 010000，真值为16

（2）对原码001010和100110符号位有0⊕1 = 1，数值部分为01010和00110

C P Y 说明

0 0 0 0 0 0 0 0 1 1 0 P0 = 0

+ 0 0 0 0 0 Y5 = 0，不做加法（加0）

0 0 0 0 0 0 0 0 1 1 0 C、P和Y同时右移一位

0 0 0 0 0 0 0 0 0 1 1 得P1

+ 0 1 0 1 0 Y4 = 1，+X

0 0 1 0 1 0 C、P和Y同时右移一位

0 0 0 1 0 1 0 0 0 0 1 得P2

+ 0 1 0 1 0 Y3 = 1，+X

0 0 1 1 1 1 C、P和Y同时右移一位

0 0 0 1 1 1 1 0 0 0 0 得P3

+ 0 0 0 0 0 Y2 = 0，不做加法（加0）

0 0 0 1 1 1 1 0 0 0 0 C、P和Y同时右移一位

0 0 0 0 1 1 1 1 0 0 0 得P4

+ 0 0 0 0 0 Y1 = 0，不做加法（加0）

0 0 0 0 1 1 1 1 0 0 0 C、P和Y同时右移一位

0 0 0 0 0 1 1 1 1 0 0 得P5

可见[x×y]原 = 100000 111100，是12位机器数原码表示，真值为-60

（3）[x]补 = 001010，[y]补 = 111010，[-x]补 = 110110

P Y y-1 说明

0 0 0 0 0 0 1 1 1 0 1 0 0 设y-1 = 0，[P0]补 = 0

y0y-1 = 00，P、Y直接右移一位

0 0 0 0 0 0 0 1 1 1 0 1 0 得[P1]补

+ 1 1 0 1 1 0 y1y0 = 10，+[-x]补

1 1 0 1 1 0 P、Y同时右移一位

1 1 1 0 1 1 0 0 1 1 1 0 1 得[P2]补

+ 0 0 1 0 1 0 y2y1 = 01，+[x]补

0 0 0 1 0 1 P、Y同时右移一位

0 0 0 0 1 0 1 0 0 1 1 1 0 得[P3]补

+ 1 1 0 1 1 0 y3y2 = 10，+[-x]补

1 1 1 0 0 0 P、Y同时右移一位

1 1 1 1 0 0 0 1 0 0 1 1 1 得[P4]补

y4y3 = 11，P、Y直接右移一位

1 1 1 1 1 0 0 0 1 0 0 1 1 得[P5]补

y5y4 = 11，P、Y直接右移一位

1 1 1 1 1 1 0 0 0 1 0 0 1 得[P6]补

可见[x×y]补 = 111111 000100，是12位机器数补码表示，真值为-60

（4）[x]原数值部分扩展为10位00000 01010，[y]原数值部分00110，[-y]补取11010

A Q 说明

0 0 0 0 0 0 1 0 1 0 开始R0 = X

+ 1 1 0 1 0 R1 = X – Y

1 1 0 1 0 0 1 0 1 0 R1 < 0，则q6 = 0，没有溢出

1 0 1 0 0 1 0 1 0 \_ 2R1（R和Q同时左移，空出一位商）

+ 0 0 1 1 0 R2 = 2R1 + Y

1 1 0 1 0 1 0 1 0 0 R2 < 0，则q5 = 0

1 0 1 0 1 0 1 0 0 \_ 2R2（R和Q同时左移，空出一位商）

+ 0 0 1 1 0 R3 = 2R2 + Y

1 1 0 1 1 0 1 0 0 0 R3 < 0，则q4 = 0

1 0 1 1 0 1 0 0 0 \_ 2R3（R和Q同时左移，空出一位商）

+ 0 0 1 1 0 R4 = 2R3 + Y

1 1 1 0 0 1 0 0 0 0 R4 < 0，则q3 = 0

1 1 0 0 1 0 0 0 0 \_ 2R4（R和Q同时左移，空出一位商）

+ 0 0 1 1 0 R5 = 2R4 + Y

1 1 1 1 1 0 0 0 0 0 R5 < 0，则q2 = 0

1 1 1 1 0 0 0 0 0 \_ 2R5（R和Q同时左移，空出一位商）

+ 0 0 1 1 0 R6 = 2R5 + Y

0 0 1 0 0 0 0 0 0 1 R6 > 0，则q1 = 1

商的最高位为0，说明没有溢出，数值部分为00001，符号位为0⊕1 = 1

则[x/y]原的商为100001原码表示，真值为-1，余数为000100，真值为4

1. 用一位乘法计算要8 \* 1ns + 8 \* 0.5ns = 12ns，两位乘法要4 \* 1ns + 4 \* 0.5ns = 6ns
2. 不能，尽管使用unsigned long long之后，实参arraysize的表示范围增大了，但由于标注库函数函数malloc的形参定义为unsigned int，则实际分配空间时，arraysize仍会被转换成unsigned int传入，则按例3.8中取count = 230 + 1，arraysize超出unsigned int能表示的最大范围，malloc函数还是只会分配4个字节的空间，同样造成整数溢出。

标准库函数不能修改，则程序员要么手动实现自己的形参定义为unsigned long long的与malloc功能相同的函数用于内存分配。如果还要使用C语言提供的malloc，则应该在调用该函数前检查arraysize是否超出unsigned int表示范围，若超出，则输出提示信息告知用户数组过大，元素复制失败，并终止程序，否则再进行正常复制。

具体实现为将例3.8中第五行改为：

unsigned long long arraysize = count\* (unsigned long long) sizeof (int);

unsigned int myarraysize = (unsigned int) arraysize;

if (myarraysize != arraysize) {

printf(“Failure: array size too large\n”);

return -1;

}

int myarray = (int \*) malloc (myarraysize);

1. 使用移位（左移n位相当于乘以2n）和加减来实现乘法比直接进行乘法操作更合算。

55 × x = 64 × x – 9 × x = 64 × x – 8 × x – x

55 × x = 32 × x + 23 × x = 32 × x + 16 × x + 8 × x – x

则前一种更合算，进一步可表示为55 × x = x << 6 – x << 3 – x

只需要进行两次位移、两次减法操作即可，共需要4个时钟周期就可以完成

1. IEEE 754标准单精度和双精度浮点数格式的尾数分别为23位和52位

则加上隐藏位，能表示的最大有效位数分别为24位和53位

则不能精确表示的最小正整数分别为224 + 1和253 + 1

1. ①对于结果形如+/-1b.bb…b的情况，需要进行右归：尾数右移一位，阶码加1。注意右移后得到的隐藏位为1。最后一位移出时，要考虑舍入。

②对于结果形如+/-0.00…01bb..b的情况，需要进行左归：尾数逐次左移，每移一位价码减1，直到阶码全为0或第一位1移到小数点左边。其中当阶码全为0时，尾数不再左移，结果为非规格化数形式。进行尾数相加时，默认小数点位置在第一个数值位（即隐藏位）之后，所以小数点右移k位后被移到了第一位1后面，这个1就是隐藏位。

1. x = 0.75 = 0.11B = 1.1B×2-1，机器数为0 0111 1110 100 0000 0000 0000 0000 0000

y = -65.25 = 1000001.01B = 1.00000101B×26，

机器数为1 1000 0101 000 0010 1000 0000 0000 0000。

1. 对阶。[Ex]移 = 0111 1110，[Ey]移 = 1000 0101，[Ex – Ey]补 = [Ex]移 + [-[Ey]移]补 = 0111 1110 + 0111 1011 = 1111 1001，Ex – Ey = -111B = -7，x的尾数右移7位，对阶后x的阶码为1000 0101，尾数为0.000 000**1** 1000 0000 0000 0000 **00**，其中粗体“1”为右移的隐藏位，最低两位粗体“0”为移出时保留的附加位。
2. 尾数相加。0.000 0001 1000 0000 0000 0000 00 – 1.000 0010 1000 0000 0000 0000 = -1.000 0001 0000 0000 0000 0000 00。
3. 尾数规格化。尾数相加后的结果已经是规格化结果。
4. 舍入。针对第（3）步得到的结果，对小数点右边第23位后的数字进行舍入，得到最终的尾数部分。此处舍去的两位数字均为0，直接丢弃即可。

x + y的机器数为1 1000 0101 000 0001 1000 0000 0000 0000

x + y的真值为-1.0000001B×26 = -1000000.1B = -64.5

* 1. 对阶，同上，阶码为1000 0101。

1. 尾数相加。0.000 0001 1000 0000 0000 0000 00 + 1.000 0010 1000 0000 0000 0000 = -1.000 0100 0000 0000 0000 0000 00。
2. 尾数规格化。尾数相加后的结果已经是规格化结果。
3. 舍入。针对第（3）步得到的结果，对小数点右边第23位后的数字进行舍入，得到最终的尾数部分。此处舍去的两位数字均为0，直接丢弃即可。

x + y的机器数为0 1000 0101 000 0100 0000 0000 0000 0000

x + y的真值为1.00001B×26 = 1000010B = 66