2.58

int is\_little\_endian() {

int test\_num = 1;

return \*(char\*)&test\_num;

}

2.71

A.错在忽略了int为一个有符号的数，而原函数获得的数前3个byte的bit pattern均为0。

B.

法一：可以将获得的一个byte的数据通过char型数据保存，之后强制类型转换为int(系统函数会自动根据转换后类型是否为有符号数来自动选择合适的补全方式)

int xbyte(pack\_t word, int bytenum) {

assert(bytenum >= 0 && bytenum < 4);

char byte = (word >> (byte << 3)) & 0xff;

return (int)byte;

}

法二：换一种移位的方式，由于无论是否是有符号数，向左的移位的补充的数均为0，将那个byte的数据移动到最高位，保存在一个int型数据中，然后通过>>运算，系统自动识别运算符左边的operand是否为有符号数，来确定补全的方式

int xbyte(pack\_t word, int bytenum) {

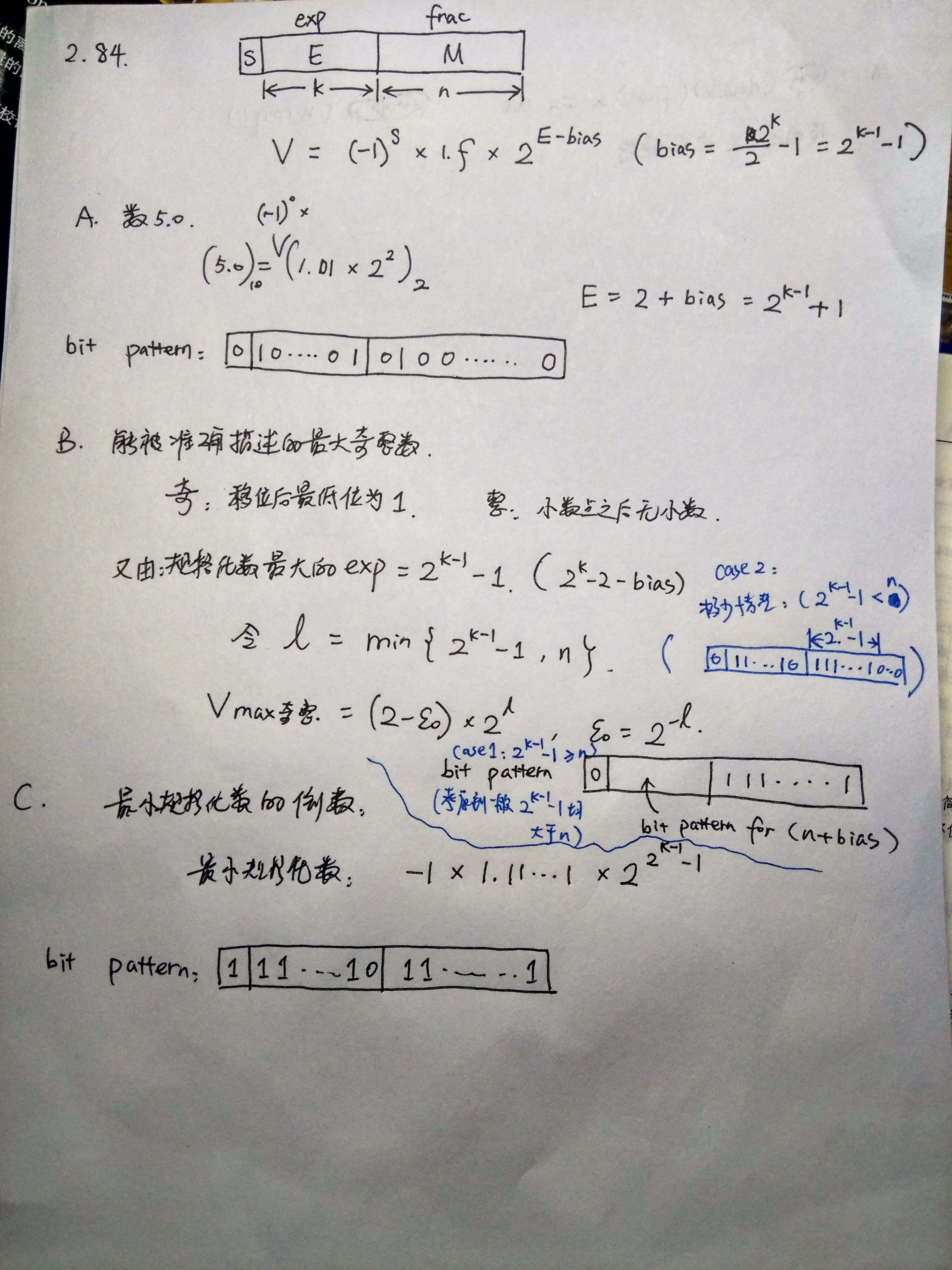
assert(bytenum >= 0 && bytenum < 4);

int num = (word << ((3 - bytenum) << 3));

return num >> 24;

}

2.84



2.88

A.(double)(float)x == dx

错，32bit的int型x在强制转化为float型数据之之后，由于float小数位长度小于30(只有23)，因此无法准确表示所有的int型数。

例子：x为bit pattern均为1的int型数

B.dx + dy == (double)(x + y)

错，x + y可能发生溢出，而左边则不会出现溢出。

例子：x，y均为bit pattern均为1的int型数

C.dx + dy + dz == dz + dx + dy

对，对int型整数，double型均可正确表示，且加法移码对齐的时候也不会发生舍入。

D.dx \* dy \* dz == dz \* dx \* dy

错，两个int型值相乘，有效位数超出了double型小数可以表示的范围，会发生舍入，导致不精确，结合律失效(不过到不会发生溢出)。

例子：

当dx与dy相乘向上舍入，再与dz相乘又是向上舍入；而dz与dx相乘向下舍入，再与dy相乘向下舍入的时候，其运算结果一定不一致（具体例子很难说，但是一般都是不对的，极少数可能是对的）

E.dx/dx == dy/dy

错：x/x，当x为0.0时，结果为NaN，并不为1。

例子：见上述

2.92

#include <stdio.h>

typedef int float\_bits;

float\_bits bit\_pattern(float f) {

return \*(int\*)&f;

}

int isNaN(float\_bits fb) {

printf("%x\n", fb);

printf("%x\n", (fb & 0x7f800000));

if (((fb & 0x7f800000) == 0x7f800000 ) && (fb & 0x007fffff)) { //指数部分全为1且尾数部分不全为0

return 1;

} else {

return 0;

}

}

float float\_negate(float\_bits fb) {

if (isNaN(fb)) {

return \*(float\*)&fb;

} else {

int temp = fb ^ 0x80000000;

return \*(float\*)(&temp);

}

}

int main() {

int a = 0x7f800700; //bit pattern of NaN

float f1 = \*(float\*)&a;

float f2 = 5.3;

printf("%f", float\_negate(bit\_pattern(f1)));

printf("%f", float\_negate(bit\_pattern(f2)));

return 0;

}

基本的思想是将float型转换为bit pattern完全相同的int型数据，然后可以通过位与，异或等操作查看内部具体每个bit的值，并根据float编码原则判断是否是NaN，如果是，将其转换为bit pattern完全相同的float型数据并返回，不是，最高位取反，并转换为float返回。