中国海洋大学 计算机科学与技术系

实验报告

姓名：陈岳阳　学号：21020007009　专业：计算机科学与技术

科目：计算机系统基础　　题目：lab1

实验时间: 2022/9/30

实验成绩:　　实验教师：范浩

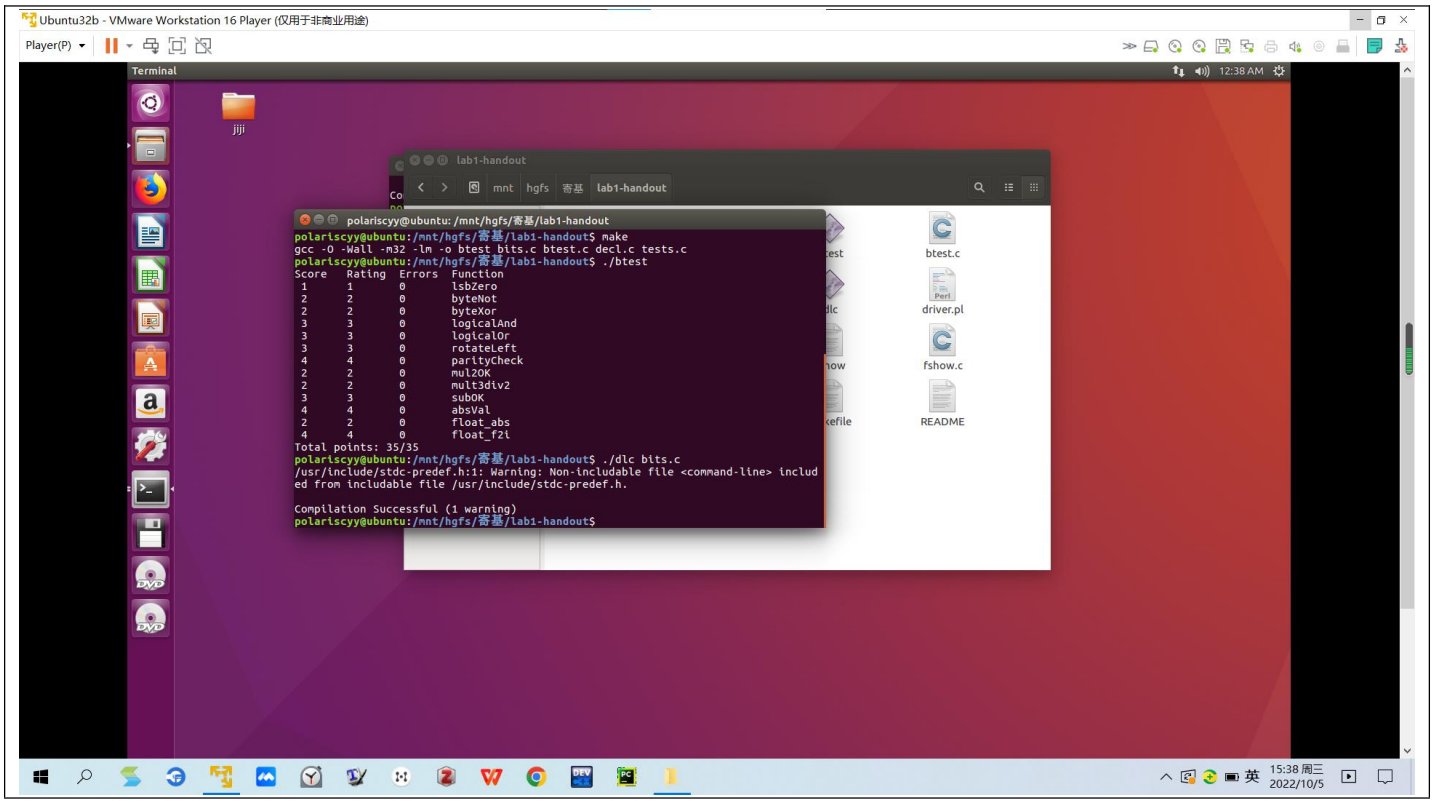
一、实验目的

1. 深刻理解位运算。
2. 深刻理解浮点数的 IEEE 754 标准。
3. 了解 vim 等工具的使用。
4. 15213.

二、实验要求

补充 bits.c 中所有函数的代码，并通过 dlc 检测，在 btest 中拿到所有 point！

三、实验内容（所修改函数代码，功能以及重要代码的解释）：



lsbZero(int x)：将 x 最低位置 0。

解法：将 x 右移一位后左移一位即可。

```
int lsbZero(int x) {
    x = x >> 1;
    x = x << 1;
    return x;
}
```

byteNot(int x, int n)：将 x 的第 n 个字节取反
解法：<{0, 1}, ^>构成群，0 为幺元，1 为逆元。
0xff<<(n<<3)可以得到一个第 n 个字节中位全为 1，其余全为 0 的数

```
int byteNot(int x, int n) {
    n = n << 3;
    n = 0xff << n;
    x = x ^ n;
    return x;
}
```

byteXor(int x, int y, int n): 如果 x 与 y 的第 n 字节相同，返回 0，否则返回 1
解法：将 x 与 y 取异或，然后将第 n 字节移到最低字节，按位与 0xff。此时，如果 x 与 y 第 n 字节相同，结果的最低字节应该全 0，理应返回 0，否则应该返回 1.

```
int byteXor(int x, int y, int n) {
    x = x ^ y;
    n = n << 3;
    x = x >> n;
    x = x & 0xff;
    return !!x;
}
```

logicalAnd(int x, int y): x&&y
解法：用！将 x 和 y 转换成逻辑值。

```
int logicalAnd(int x, int y) {
    x = !!x;
    y = !!y;
    return x & y;
}
```

logicalOr(int x, int y): x||y
解法：同 logicalAnd

```
int logicalOr(int x, int y) {
    x = !!x;
    y = !!y;
    return x | y;
}
```

rotateLeft(int x, int n): 将 x 向左旋转 n 位！(左边超出的移到右边）
解法：重点是怎么取出 x 的最左边 n 位。tp2 定义的前半部位将最高 n 位移到最低 n 位，后半部分用于去除算术右移时补的 1。

```
int rotateLeft(int x, int n) {
    int tp1 = x << n;
    int tp2 = (x >> (32 + ~n + 1)) & ~((~0) << n);
    return tp1 | tp2;
}
```

parityCheck(int x): 若 x 有奇数个位是 1，返回 1，否则返回 0
解法：同样运用了群的性质，这里还利用了结合律。(CSAPP 中显式地提到了这点)
重点是怎么将 32 位合到 1 位。这里使用右移，每次将有效位数缩小到原来的一半，并放在低位。

```
int parityCheck(int x) {
    x = x ^ (x >> 16);
    x = x ^ (x >> 8);
    x = x ^ (x >> 4);
    x = x ^ (x >> 2);
    x = x ^ (x >> 1);
    return x & 0x1;
}
```

mul2OK(int x): 如果*2 不溢出，返回 1，否则返回 0
解法：OF=最高位 CF ^ 次高位 CF。返回值为!OF
x*2 情况下，这两位的 CF 仅与这两位的值有关，如果是 10 或 01，即取异或结果为 1，则溢出。

```
int mul2OK(int x) {
    return (((x >> 31) & 1)^((x >> 30) & 1))^1;
}
```

mult3div2(int x): 返回 x*3/2，向 0 舍入
解法：首先求出 x*3。如果是负数，还要加上 2^k-1 = 1 的 offset。最后除以 2。

```
int mult3div2(int x) {
    x = x + (x << 1);
    return (x + (1 & x >> 31)) >> 1;
}
```

subOK(int x, int y): 如果 x-y 不溢出，返回 1，否则返回 0。
解法：如果 x-y 溢出，x 与 y 一定不同符号，先判断符号位是否相同，不同时为 0xffffffff(为了节省两个 op，要排名的)，否则为 0。
如果符号位不同，判断 x-y 的符号位是否与 x 的符号位相同。如果不同，结果为 1，说明发生截断，溢出。
由于不溢出返回 1，因此还要对结果取反。

```
int subOK(int x, int y) {
    int sig_x = x >> 31;
    int sig_y = y >> 31;
    return !( (sig_x ^ sig_y) & (sig_x ^ ( x + ~y + 1) >> 31 ) );
}
```

absVal(int x): 返回 x 的绝对值
解法：通过符号位右移 31 位，得到 sig_x=0xffffffff 或 0。sig_x^x，当 x 是负数，相当与取反；x 是

正数，不变。然后再对取反后的负数+1，正数+0.

```
int absVal(int x) {
    int sig_x = x>>31;
    return (sig_x ^ x) +(sig_x & 1);

}
```

float_abs(unsigned uf)：用 **uf** 表示浮点数。如果 **uf** 表示的浮点数是 **NaN**，返回 **uf**，否则返回 **uf** 的绝对值。
解法：**IEEE 754** 标准的浮点数正负仅由符号位控制，与其它位无关。
如果阶码 **E==0xff** 且 **frac != 0**，**uf** 是 **NaN**，返回 **uf**，否则将符号位置 **0** 即可。

```
unsigned float_abs(unsigned uf) {
    int E = uf >> 23 & 0xff;
    if((E == 0xff) && (uf & 0x7fffff))
        return uf;
    return (uf & 0x7fffffff);
}
```

float_f2i(unsigned uf)：将 **uf** 转换成 **int** 并返回。如果 **uf** 超过 **int** 表示范围，或是 **NaN** 或 **Inf**，返回 **0x80000000u**
解法：首先取出 **uf** 的符号位、阶码和 **frac**
如果 **E>158**，由于位数部分自带 **1**，这个值的绝对值已经不小于 **INT_MAX+1**，直接返回 **0x80000000u**。
如果 **E<158**，这个值已经小于 **1**，且转换时向 **0** 舍入，直接返回 **0**。
E 在上述两种情况以外的范围中时，舍入后一定在整数表示范围内，首先将尾数部分添上隐含的 **1**，然后根据 **E** 和 **bias** 计算出 **exp**(同样为了省 **op**，没写 **exp**)，对尾数进行操作。注意到此时尾数是 **24** 位 **2** 进制数而非小数，因此要根据 **exp** 判断左移或是右移。
最后，根据 **uf** 的符号位，判断是否取反。

```
int float_f2i(unsigned uf) {
    int S = uf >> 31;
    int E = uf >> 23 & 0xff;
    int frac = uf && 0x7fffff;
    if(E > 158){
        return 0x80000000u;
    }else if(E < 127){
        return 0;
    }else{
        frac = frac | 0x800000;
        if(E > 150){
            frac = frac << (E - 150);
        }else{
            frac = frac >> (150 - E);
        }
        if(S){
            frac = ~frac+1;
        }
        return frac;
```

```
    }
}
```

四、实验总结

很好的巩固了第二章所学内容，深刻了解了位运算和浮点数、整数的表示。

但是早就出了新版 **lab** 了，可以更新一下。