计算机系统基础 Lab1 实验报告

1. 实验内容:

   完成计算机位运算相关函数。

2. 实验步骤:

   1. 利用 ssh 从远程服务器获得任务文件。
   2. 对 bit.c 文件内缺失的函数进行补全。
   3. 在 Ubuntu 虚拟机内对程序进行编译测试。
   4. 上传程序至测试平台。
   5. 撰写实验报告。

3. 函数详解:

   1. bitXor 利用公式及德摩根律。
   2. thirdBits 按位寻找规律并依此进行移位，通过一次移动更多 1 来节约运算符。
   3. fitsShort 通过移位与异或判断前十六位是否为空。
   4. isTmax 利用 t_max+1 变为 t_min 的性质变换判断条件。
   5. fitsBits fitsShort 的可变版本。
   6. upperBits 利用 int 右移扩展符号位的特性（注意 0 的判断）
   7. anyOddBit 使用多个 1010 1010 作为掩码，进一步异或进行判断。
   8. byteSwap 通过位移 1111 1111 掩码选出字节，再进行交换。
   9. absVal 利用一个数异或自身符号位会变为正数的性质。
   10. divpwr2 基本操作通过移位实现，主要注意负数判断及小数部分的修正。

11. float_neg　　取反并且判断特殊情况。

12. logicalNeg　　通过右移出零、异或操作判断是否为 0。

13. bitmask　　通过取反构造 1，左右移改变位置。

14. isGreater　　利用取反加一构造减法，对差进行正负判断。

15. logicalShift　在判断 n 是否过大（32）或过小（0）后直接进行按位右移。

16. satMul2　　在判断是否越界后直接进行按位左移并利用掩码处理越界情况。

17. subOK　　原理与 isGreater 相近。

18. trueThreeFourths　右移两位获得原数四分之一，与原数做差后进行舍入。

19. isPower2　　找到规律（为二次方的 int 值必为正且仅有一个 1）后利用与自身减一按位与是否为零判断 1 的个数。

```
491
492  /*
493   * isPower2 - returns 1 if x is a power of 2, and 0 otherwise
494   *   Examples: isPower2(5) = 0, isPower2(8) = 1, isPower2(0) = 0
495   *   Note that no negative number is a power of 2.
496   *   Legal ops: ! ~ & ^ | + << >>
497   *   Max ops: 20
498   *   Rating: 4
499   */
500  int isPower2(int x) {
501    int ifnegativeis0 = !(x>>31);
502    int if0is0 = !(!x);
503    int neg1 = ~0;
504    int Pnowifonlyone1is1 = !(x & (x+neg1));
505
506    return ifnegativeis0 & if0is0 & Pnowifonlyone1is1;
507  }
508
```

20. float_i2f

首先取符号位 s

再逐步右移计算 e

根据阶码 e 与原数求得 f

最后按位或合并 s，e，f 得到结果

21. howManyBits　先行处理数据为正数，再利用逐步右移判断哪个高位首先为 1，即可得出结果。（使用二分法可以提升复杂度为代价减少运算符的使用）

```
int howManyBits(int x) {
    int ALLS = x >> 31;         //右移出s
    int PisPNisOp = x ^ ALLS;   //正数不变负数按位取反
    int Zero0El1 = !(!PisPNisOp);

    //printf("first PisPNisOp:%x\n",PisPNisOp);

    int bit_16,bit_8,bit_4,bit_2,bit_1;   //有为1
    bit_16=!(!(PisPNisOp >> 16)) << 4;
    PisPNisOp=PisPNisOp >> bit_16;  //有就看前面没有看后面
    bit_8=!(!(PisPNisOp >> 8)) << 3;
    PisPNisOp=PisPNisOp >> bit_8;
    bit_4=!(!(PisPNisOp >> 4)) << 2;
    PisPNisOp=PisPNisOp >> bit_4;
    bit_2=!(!(PisPNisOp >> 2)) << 1;
    PisPNisOp=PisPNisOp >> bit_2;
    bit_1=!(!(PisPNisOp >> 1));

    //printf("x:%x alls %x, PisPNisOp %x, bit16 %x bit8 %x bit4 %x bit2 %x bit1 %x\n",x,ALLS,Pi
    //printf("zero0el1:%x \n",Zero0El1);
    return (bit_16 | bit_8 | bit_4 | bit_2 | bit_1) + 1 + (Zero0El1 & 0x1);
}
```

22. float_half　通过选用恰当的掩码分别获得输入值的符号位，阶码，尾码，其次进行无穷大与 0 的判断，再根据阶码判断此数除以 2 后能否规格化表示。若可，则通过阶码减一实现除二操作；若不可，则尾码右移。

5. 实验心得:

注意特殊数据的处理（无穷，0，负数）；注意 int 右移符号位扩展的影响与利用；注意浮点数尾码右移导致的舍入问题（四舍六入五取偶）；注意位移 0/32 的 undefined behavior。

使用 Ubuntu 运行文件时注意 make clean。