Data Lab Solution

姓名：周泽龙
学号：2016013231

2018 年 10 月 6 日

目录

1. 实验感想

本次实验的限制无疑是十分严格的，通过开放有限的某些位运算符使用权限来实现一些简单的位级函数，无疑加深了同学们对于 C 中的位级表示和操作符对于位级表示的作用的理解。

刚开始写题时，受到各种规则的限制，让我感到束手束脚，不知从何下手。于是，我便先去重新熟悉了各种补码、操作符、IEEE 的规则。“磨刀不误砍柴工”，通过对于计算机内部运作方式的重新理解，重新做题时虽做不到马上或者一次性得到正确的解题思路，但却感觉不再受到操作符有限的各种约束了。

遇到实在想不出思路的题目，Google 始终是一个很好的开拓思路的工具。

2. 解题思路

2.1 bitAnd

根据摩根定律，

$$((x)|(y)) = x\&y$$

2.2 getByte

- x 算术右移 n*8 位
- “&”上 0x000000ff，取出低 8 位即可

2.3 logicalShift

- x 算术右移 n 位后，
- “&”上 temp，其中，temp 高 n 位为 0，其余位为 1

2.4 bitCount

采用分治法：

- 32 位二进制数分为 32 组
- 合并相邻两组，合并后数值为原来两组数值之和
- 重复第 2 步操作，直至只剩 1 组，其数值就是 1 的数量

2.5 bang

两种情况，0 和非 0：

- 0——(-0) = 0；
- 非 0——(-非 0)　非 0；

因此，方案如下：

- 取 x 的相反数（-x）
- 分别取 x 和（-x）的符号位，进行同或运算
- 0——1
- 非 0——0
- 即可得到答案

2.6 tmin

- 补码表示的整型最小值
- 即为 0x80000000

2.7 fitsBits

- 假设 x 可以用 n 位补码表示，那么将其左移 32-n 位后再算术右移 32-n 位后得到的值一定等于 x。
- 进一步简化思路，只需查看第 n-1 位与 MSB(32-n) 是否相等即可。

2.8 divpwr2

- 对于正数，直接算术右移 n 位即可
- 对于负数，算术右移后需适当的加上 bias（1）来向 0 取整
  - 整除时不需要加上 bias
  - 不能整除时需要加上 bias

2.9 negate

由补码规则，可知，

$$-x = \ x + 1$$

2.10 isPositive

- 检测 x 是否为 0
- 检测符号位是否为 0

2.11 isLessOrEqual

- 考虑上题“isPositive”思路，检测 y-x 是否大于等于 0
- 考虑溢出，分别取 x、y 的符号位
  - 若 x 正 y 反，返回 0
  - 若 x 反 y 正，返回 1

2.12 ilog2

- 找出位级表示中 1 的最高位的位数即可。
- 二分查找：
  - 若 x » 16 大于 0，则最高位的位数至少是 16，将结果的第 4 位，置 1；反之，置 0
  - 假设 x » 16 大于 0，则在第 24 位再进行二分查找
  - 不断二分查找直到不能再分

2.13 float_neg

- 对于一般浮点数（包括近 0 和无穷），只需取反符号位
- 对于 NaN，返回 NaN

2.14 float_i2f

- 对于负数，需将其绝对值后进行以下操作，记录下符号位 S。
- 取出从最高位开始算起第一个 1 以后的位数作为 M：
- 记录第一个 1 的位数序号，从而计算出 e
- E=e+127
- 按顺序存储 S、E、M 即可，其中右移 9 位时需舍入
- 舍入规则：
  - 若舍弃的 9 位最高位为 0，则表明舍去的数值小于原数第 9 位（以后称为最低位）的值的一半，那么不需舍入
  - 若舍弃的 9 位最高位为 1，那么,
    - 若后面位不全为 0，则表明舍去的数值大于最低位的值的一半，那么需要舍入，+1
    - 若后面位全为 0，则表明舍去的数值等于最低位的值的一半，那么,
      - 若最低位位 1，需舍入，+1
      - 若最低位为 0，不需舍入

2.15 float_twice

- 若浮点数是规格化的，只需 E+1
- 若浮点数的指数部分 E=0，则将基数部分左移 1 位，即 M«1
- 其余情况下返回原值即可，如无穷，NaN