|  |  |
| --- | --- |
| 实验题目：datalab | |
| 实验目的：**替换bits.c中各个函数中的return，补全函数代码** | |
| 实验环境：个人电脑、linux发行版本 | |
| **实验内容及操作步骤：**  **填写bits.c中实验需填写的内容，并进行简单的解释：**   1. **简单取反运算**     2.问题：大小端对移位运算有没有影响？        大小端是内存存储数据高位和数据低位的不同顺序；而移位移的是寄存器中的内容，与内存是无关的。移位就按照二进制数来左右移。    3.因为int是有符号类型，对其进行逻辑移位。 法一：不防先把符号位保存下来再赋0。移位之后再把符号位赋回去。 法二：先把符号位复制下了，一起移位。利用取反操作，将算术移位补的位全取反，再&回去，使得x补的位必定为0。同时巧妙利用低位为0，取反为1，&x保留原值。    4.因为符号个数限制，不能一位一位左移&0x80000000来算(不用右移是因为算术移位)。利用分治法。    5.利用0的补码是自己，其他有符号数（不论正负）的补码都表示取负（若原来为负数，取补后表示绝对值）。原理就是取补就是x’=2^k-x. 2^x最高位的1溢出，整体0。那么x’+x=0.  不论x是正是负。  利用这一特点，即取补后符号位会变，那么将两个符号位或一下，非0数的符号位肯定为1！从来判断是否为非零数！ 最后return也要注意，返回的是相反值。利用有符号数算术右移，非0数的符号位右移31位，x变成全1.这样返回neg+1满足题意。    6.最小就是-2^(k-1)    7.减法可由加补码表示！ 判断高位是否一致。因为正常来讲，先只剩符号位和补的高位（与符号位相同）。即x是否在-2(n-1)到2(n-1)-1范围内 正常全0或全1，再加1，1或0，使高位全0. 不然n不够的话，都是大数不够，特征就是最高位和符号位不一样！ 抓住这一特点。    8.若单纯右移，是向下取整。而当x为负数时，就需要+1. 这样我们根据符号位来确定加0还是加1.    9.相反数即取补    10.即判断符号位。为0返回1，为1返回0. 0为特殊情况，返回0.用&来讨论。    11.即判断是否x-y<=0。恰巧借鉴T10. 将=x-y代入T10并取非即可。    12. 得到一个数是由多少二进制位表示，倒数第二行表示当x为0的时候，还需要进行减一才能得到正确值，是一个类似于分治法，逐步化简。    13.与1异或是取反，与0异或是保持不变。 该题就是翻转符号位，并判断是否为NAN。 NAN是阶码全1，frac不为0. 将符号位拿掉，判断是否大于0x7f800000    14. 将整形转化为无符号浮点数，即求浮点数。先取的符号位，再将剩余部分全部取为正数形式，即absx，即可以得到无符号的数值。然后将有数字的部分直接移动到最高位，记录移动的位数，再将其移动9位（因尾数只要23即可）。对于阶码部分，由于记录的是小数点从31位右数到第一个1，但实际上需要处理的是从第0位到第一位，所以E=32-shiftleft，bias为127，加上为159，if部分做舍入处理(因为int为32位，去掉符号位是32位。整数化浮点数是先看成二进制科学计数法。这样尾数最多有30位。而float的尾数最多23位。所以要舍入)      15. 将无符号浮点数乘2，对无阶码小数，对其尾部乘2即可，即直接左移一位，但要提前记录符号位。对于规格化数，直接对其阶码+1即可    **实验结果及分析：**  通过使用dlc的-e选项    使用dlc检测bits.c是否有错误，代码行为：./dlc bits.c    调用btest命令检查bits.c中的所有函数的功能的正确性，以便下一步查找错误原因，指令行为：./btest bits.c    **收获与体会**：  回顾了不少原理的运算知识。同时数据思维更熟练了。还是该多练习，才能保持状态。Pwn和逆向也能快速审计代码。之后记得整理笔记写博客，多复习。 | |
|  | |
| 实  验成绩 |  |

实验报告撰写说明

1．实验题目和目的

请从实验指导资料中获取。

2．实验步骤和内容

包括：

（1）本次实验的要求；

（2）源程序清单或者主要伪代码；

（3）预期结果；

（4）上机执行或调试结果：包括原始数据、相应的运行结果和必要的说明（截图）；

3．实验体会

调试中遇到的问题及解决办法；若最终未完成调试，要试着分析原因；调试程序的心得与体会；对课程及实验的建议等。