

## Datalab

姓名：汪俊轩

学号：21307130169

终端执行 `./dlc -e bits.c`后的截图

```
wjx@Wang-Junxuan1: ~/datalab-handout
wjx@Wang-Junxuan1:~$ ls
asm.txt  bomb:Zone.Identifier  datalab-handout  datalab-handout.tar  datalab-handout.tar:Zone.Identifier  lab2
wjx@Wang-Junxuan1:~$ cd datalab-handout
wjx@Wang-Junxuan1:~/datalab-handout$ ls
Driverhdrs.pm  Makefile  bits.c  btest  btest.h  dlc  fshow  ishow  tests.c
Driverlib.pm  README  bits.h  btest.c  decl.c  driver.pl  fshow.c  ishow.c
wjx@Wang-Junxuan1:~/datalab-handout$ ./dlc -e bits.c
dlc:bits.c:150:bitNor: 3 operators
dlc:bits.c:161:tmax: 2 operators
dlc:bits.c:173:isTmin: 6 operators
dlc:bits.c:184:minusOne: 1 operators
dlc:bits.c:198:absVal: 7 operators
dlc:bits.c:212:leastBitPos: 3 operators
dlc:bits.c:233:byteSwap: 20 operators
dlc:bits.c:249:logicalShift: 8 operators
dlc:bits.c:262:isLessOrEqual: 20 operators
dlc:bits.c:277:multFiveEighths: 10 operators
dlc:bits.c:298:bitCount: 36 operators
dlc:bits.c:318:greatestBitPos: 31 operators
dlc:bits.c:330:bang: 7 operators
dlc:bits.c:351:bitReverse: 40 operators
dlc:bits.c:377:mod3: 71 operators
dlc:bits.c:404:float_neg: 10 operators
dlc:bits.c:454:float_i2f: 30 operators
dlc:bits.c:487:float_twice: 19 operators
wjx@Wang-Junxuan1:~/datalab-handout$
```

终端执行 `./btest`后的截图

```
wjx@Wang-Junxuan1: ~/datalab-handout
dlc:bits.c:318:greatestBitPos: 31 operators
dlc:bits.c:330:bang: 7 operators
dlc:bits.c:351:bitReverse: 40 operators
dlc:bits.c:377:mod3: 71 operators
dlc:bits.c:404:float_neg: 10 operators
dlc:bits.c:454:float_i2f: 30 operators
dlc:bits.c:487:float_twice: 19 operators
wjx@Wang-Junxuan1:~/datalab-handout$ ./dlc bits.c
wjx@Wang-Junxuan1:~/datalab-handout$ ./btest
Score  Rating  Errors  Function
1      1      0      bitNor
1      1      0      tmax
1      1      0      isTmin
1      1      0      minusOne
2      2      0      absVal
2      2      0      leastBitPos
2      2      0      byteSwap
3      3      0      logicalShift
3      3      0      isLessOrEqual
3      3      0      multFiveEighths
4      4      0      bitCount
4      4      0      greatestBitPos
4      4      0      bang
4      4      0      bitReverse
4      4      0      mod3
2      2      0      float_neg
4      4      0      float_i2f
4      4      0      float_twice
Total points: 49/49
wjx@Wang-Junxuan1:~/datalab-handout$
```

每个函数的思路：

`bitNor ()`

通过分析每一位的真值情况即可获得，只在全为0时结果为1。

`tmax ()`

最大的数除了符号位其他全为1，只需要得到符号位为1的数再取反即可。

`isTmin ()`

只有最小的二进制数和0满足 $x^{(\sim x+1)}$ 为0，利用和！x位或排除0的情况之后再取反即可。

`minusOne ()`

-1是所有位都为1的数，故对0取反即可。

`absVal ()`

取出符号位用于判断是否取反加一，表达式设为 $((\sim \text{flag}) \& x) | (\text{flag} \& ((\sim x) + 1))$ ，用到了flag是否为0来调节究竟是返回x还是 $((\sim x) + 1)$ 。

**leastBitPos ()**

想要得到最后一位1，应该进行某些只会影响到最后一位1的操作，经过思考后发现取反+1的操作会影响到最后一位1的位置，完善后可得 $((\sim x) + 1) \& x$ 。

**byteSwap ()**

调换第m，n位只需要分别用只在这两位上是1的数把这两位提取出来，再移位后填入即可完成交换动作。

**logicalShift ()**

得到一个前n位全为0的数与移位后的数位与，如此把即前n为置位0。

**isLessOrEqual ()**

判断y-x是否非负，同时排除掉因为溢出（如y为负而x为正的情况）即可。

**multFiveEighths()**

可拆解为除以二加上除以八。为了防止移位的时候位的丢失，将后三位和前面29位分开处理。

**bitCount ()**

原本思路是一个一个相加，但如果一个一个相加，写出代码后发现符号数完全不够用，因此必须使得一个+号完成多次加法，因此想到把奇数位和偶数位相加，再把01位和23位相加（以此类推），可以做到一个+号完成了多次加法，最终得出了答案。

**bang ()**

本题是要将0与非0数的差别在某一位具体的数字上体现出来，因此思考后发现可以利用任何非0数都满足本身或者本身加上最大二进制数之后首位总有一个会为1的特点，将0与非0数区分开来，得到最终结果。

**bitReverse()**

先相邻两位调换，再相邻四位调换，再相邻八位调换，最后相邻16位调换，即可实现32位数的从头到尾的调换。

**mod3 ()**

利用数学知识发现，从低位到高位，出现在第奇数位上的1总会给该数贡献余数1，出现在第偶数位上的1总会给该数贡献余数2，如此将它们分别相加即可化简，重复多次即可得到最终结果，最后将3转化为0即可。

**float\_neg()**

将尾数部分取出，用if结构讨论是否需要将符号位取反即可。

**float\_i2f()**

同样分情况讨论，现将0和最小负数的情况取出，之后再判断最高位的位置自后进行移位即可，注意一下进位。

**float\_twice ()**

将0和最小负数情况取出后讨论即可