南京航空航天大学计算机科学与技术学院计算机组成原理

实验 Datalab 数据表示实验报告

实验信息

姓名：赵维康
学号：161630220
班级：1616302
指导教师：李博涵
实验日期：2018/6/26

第 1 题：bitNor

题目要求理解

初步判断这一题的函数实现的功能是什么

两个操作数按位或再按位取反。

编码思路分析

解释一下你是怎么想到这样写的，然后简述一下这样写的正确性

由所给测试用例 bitNor(0x6, 0x5) = 0xFFFFFFF8，0x5、0x6 对应的二进制分别为 00…0000 0101、00…0000 0110 分别对其按位取反得 11…1111 1010、11…1111 1001，再按位与得 0xFFFFFFF8，所以返回值为（~x）&（~y）。事实上，可以由德摩根定律得：~(x|y)=(~x)&(~y)。第一题的语法检查及正确性检查如图

```
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc bits.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc -e bits
.c
dlc:bits.c:175:bitNor: 3 operators
dlc:bits.c:185:bitXor: 0 operators
dlc:bits.c:195:allEvenBits: 0 operators
dlc:bits.c:207:byteSwap: 0 operators
dlc:bits.c:221:multFiveEighths: 0 operators
dlc:bits.c:231:conditional: 0 operators
dlc:bits.c:241:isGreater: 0 operators
dlc:bits.c:252:localAnd: 0 operators
dlc:bits.c:263:logicalNeg: 0 operators
dlc:bits.c:277:float_abs: 0 operators
dlc:bits.c:289:float_i2f: 0 operators
dlc:bits.c:303:float_half: 0 operators
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# make
gcc -O -Wall -m32 -lm -o btest bits.c btest.c decl.c tests.c
gcc -O -Wall -m32 -o fshow fshow.c
gcc -O -Wall -m32 -o ishow ishow.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./btest -f bi
tNor
Score   Rating  Errors  Function
 1      1       0       bitNor
Total points: 1/1
```

解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

```
int bitNor(int x, int y) {
return (~x)&(~y);
}
```

第 2 题：bitXor

题目要求理解

初步判断这一题的函数实现的功能是什么

实现两个操作数的按位异或操作

编码思路分析

解释一下你是怎么想到这样写的，然后简述一下这样写的正确性

由于题目要求只能用~、&两个操作符 ，由离散数学代数部分知识得x^y=((x&~y)|~x&y)= ~(~(~x&y)&~(x&~y))。最后一个等号后边正好使用了第一题的德摩根定理。第二题的语法检查及正确性检查如图

```
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc bits.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc -e bits
.c
dlc:bits.c:175:bitNor: 3 operators
dlc:bits.c:185:bitXor: 8 operators
dlc:bits.c:195:allEvenBits: 0 operators
dlc:bits.c:207:byteSwap: 0 operators
dlc:bits.c:221:multFiveEighths: 0 operators
dlc:bits.c:231:conditional: 0 operators
dlc:bits.c:241:isGreater: 0 operators
dlc:bits.c:252:localAnd: 0 operators
dlc:bits.c:263:logicalNeg: 0 operators
dlc:bits.c:277:float_abs: 0 operators
dlc:bits.c:289:float_i2f: 0 operators
dlc:bits.c:303:float_half: 0 operators
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# make
gcc -O -Wall -m32 -lm -o btest bits.c btest.c decl.c tests.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./btest -f bi
tXor
Score   Rating  Errors  Function
 1      1       0       bitXor
Total points: 1/1
```

解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

```
int bitXor(int x, int y) {
return ~(~(~x&y)&~(x&~y));
}
```

第 3 题：allEvenBits

题目要求理解

初步判断这一题的函数实现的功能是什么

如果该操作数的所有偶数位都为 1，返回 1，否则，返回 0。

编码思路分析

解释一下你是怎么想到这样写的，然后简述一下这样写的正确性

首先找一个偶数位都是 1 的数，例如 0xaa，然后将它左移 8 位，再加它自身，得到 s（0xaa）,再让 s 左移 16 位，再加 s，得到 t（值还是 0xaa，不过位数变为 16，并且所有偶数位为 1），x 与 t 按位或再加 1，然后取反，返回取反结果的低八位的值即为所求。第三题的语法检查及正确性检查如图

```
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc bits.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc -e bits
.c
dlc:bits.c:175:bitNor: 3 operators
dlc:bits.c:185:bitXor: 8 operators
dlc:bits.c:201:allEvenBits: 7 operators
dlc:bits.c:213:byteSwap: 0 operators
dlc:bits.c:227:multFiveEighths: 0 operators
dlc:bits.c:237:conditional: 0 operators
dlc:bits.c:247:isGreater: 0 operators
dlc:bits.c:258:localAnd: 0 operators
dlc:bits.c:269:logicalNeg: 0 operators
dlc:bits.c:283:float_abs: 0 operators
dlc:bits.c:295:float_i2f: 0 operators
dlc:bits.c:309:float_half: 0 operators
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# make
gcc -O -Wall -m32 -lm -o btest bits.c btest.c decl.c tests.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./btest -f al
lEvenBits
Score   Rating  Errors  Function
 2      2       0       allEvenBits
Total points: 2/2
```

解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

```
int allEvenBits(int x) {
int s;
int t;
s = (0xaa << 8) + 0xaa;
t = (s << 16) + s;
x = (x | t) + 1;

return !x;
}
```

第 4 题：byteSwap

题目要求理解

初步判断这一题的函数实现的功能是什么

将一个 32 位操作数的第 n 个字节与第 m 个字节互相交换

编码思路分析

解释一下你是怎么想到这样写的，然后简述一下这样写的正确性

解题源代码可以详解为如下代码：

int shift_n = n << 3;// 以题目所给用例 byteSwap(0x12345678, 1, 3) = 0x56341278，n=1,m=3,此时，shift_n=8

int shift_m = m << 3;// shift_m=24

int n_x = (x >> shift_n) & 0xff;//获取第一个字节的信息

int m_x = (x >> shift_m) & 0xff;//获取第三个字节的信息

return (x ^ (n_x << shift_n) ^ (m_x << shift_m)) | (n_x << shift_m) | (m_x << shift_n);//第一个|运算符前面的结果仍为 x，第一个|运算符后面的整体结果为第一个字节与第三个字节相互交换后的结果，两部分综合即为题目所求。第四题的语法检查及正确性检查如图

```
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc bits.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc -e bits
.c
dlc:bits.c:175:bitNor: 3 operators
dlc:bits.c:185:bitXor: 8 operators
dlc:bits.c:201:allEvenBits: 7 operators
dlc:bits.c:220:byteSwap: 10 operators
dlc:bits.c:234:multFiveEighths: 0 operators
dlc:bits.c:244:conditional: 0 operators
dlc:bits.c:254:isGreater: 0 operators
dlc:bits.c:265:localAnd: 0 operators
dlc:bits.c:276:logicalNeg: 0 operators
dlc:bits.c:290:float_abs: 0 operators
dlc:bits.c:302:float_i2f: 0 operators
dlc:bits.c:316:float_half: 0 operators
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./btest -f by
teSwap
Score   Rating  Errors  Function
 2      2       0       byteSwap
Total points: 2/2
```

解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

```
int byteSwap(int x, int n, int m) {
    int shift_n;
    int shift_m;
    int s;
    shift_n = n << 3;
    shift_m = m << 3;
    s = ((x >> shift_m) ^ (x >> shift_n)) & 0xff;

    return x ^ ((s << shift_m) | (s << shift_n));
}
```

第 5 题：multFiveEighths

题目要求理解

初步判断这一题的函数实现的功能是什么

求一个操作数乘以 5 除以 8 之后的结果。若结果是非整数，向下取整，同时也对溢出结果进行处理。

编码思路分析

解释一下你是怎么想到这样写的，然后简述一下这样写的正确性

首先对该操作数进行乘以 5 的操作，通过(x << 2) + x（x*5）即可实现。然后进行除以 8 的操作，通过 x>>3（x/8）即可实现。通过 x>>31 可获得其符号为，便于进行溢出判断。第五题的语法检查及正确性检查如图

```
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc bits.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc -e bits
.c
dlc:bits.c:175:bitNor: 3 operators
dlc:bits.c:185:bitXor: 8 operators
dlc:bits.c:201:allEvenBits: 7 operators
dlc:bits.c:220:byteSwap: 10 operators
dlc:bits.c:238:multFiveEighths: 6 operators
dlc:bits.c:248:conditional: 0 operators
dlc:bits.c:258:isGreater: 0 operators
dlc:bits.c:269:localAnd: 0 operators
dlc:bits.c:280:logicalNeg: 0 operators
dlc:bits.c:294:float_abs: 0 operators
dlc:bits.c:306:float_i2f: 0 operators
dlc:bits.c:320:float_half: 0 operators
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# make
gcc -O -Wall -m32 -lm -o btest bits.c btest.c decl.c tests.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./btest -f mu
ltFiveEighths
Score   Rating  Errors  Function
 3      3       0       multFiveEighths
Total points: 3/3
```

解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

```
int multFiveEighths(int x) {
  int s;
  x = (x << 2) + x;
  s = (x >> 31) & 0x7;

  return (x + s) >> 3;
}
```

第 6 题：conditional

题目要求理解

初步判断这一题的函数实现的功能是什么

实现 x ？ y ： z 的功能。

编码思路分析

解释一下你是怎么想到这样写的，然后简述一下这样写的正确性

首先由返回结果为 y 或 z 知有表达式（y &__）|（z & __），当 x 为 1 时返

回 y,当 x 为 0 时返回 z。对于第一个空，当 x=1 时，必须为 0xffffffff,所以得第一空为~(~!x+1)，同理，对于第二空，当 x=0 时，必须为 0x00000000，所以得第一空为~!x+1。第六题的语法检查及正确性检查如图

```
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc bits.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc -e bits
.c
dlc:bits.c:175:bitNor: 3 operators
dlc:bits.c:185:bitXor: 8 operators
dlc:bits.c:201:allEvenBits: 7 operators
dlc:bits.c:220:byteSwap: 10 operators
dlc:bits.c:238:multFiveEighths: 6 operators
dlc:bits.c:251:conditional: 7 operators
dlc:bits.c:261:isGreater: 0 operators
dlc:bits.c:272:localAnd: 0 operators
dlc:bits.c:283:logicalNeg: 0 operators
dlc:bits.c:297:float_abs: 0 operators
dlc:bits.c:309:float_i2f: 0 operators
dlc:bits.c:323:float_half: 0 operators
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# make
gcc -O -Wall -m32 -lm -o btest bits.c btest.c decl.c tests.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./btest -f co
nditional
Score   Rating  Errors  Function
 3      3       0       conditional
Total points: 3/3
```

解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

```
int conditional(int x, int y, int z) {
  int s;
  s = ~!x + 1;

  return (y & ~s)|(z & s);
}
```

第 7 题：isGreater

题目要求理解

初步判断这一题的函数实现的功能是什么

判断第一个操作数是否大于第二个操作数，若大于，返回 1，否则，返回 0。

编码思路分析

解释一下你是怎么想到这样写的，然后简述一下这样写的正确性

返回值为 1（x>y）可分为 x>0，y<0 或 x，y 同号且 x-y>=0。返回值为 0 的可分为 x 为负数（符号为 1），y 为非负（符号为 0）或者 x，y 同号且 x-y<0（((x+~y) >>31)。第七题的语法检查及正确性检查如图

```
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc bits.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc -e bits
.c
dlc:bits.c:175:bitNor: 3 operators
dlc:bits.c:185:bitXor: 8 operators
dlc:bits.c:201:allEvenBits: 7 operators
dlc:bits.c:220:byteSwap: 10 operators
dlc:bits.c:238:multFiveEighths: 6 operators
dlc:bits.c:251:conditional: 7 operators
dlc:bits.c:271:isGreater: 20 operators
dlc:bits.c:282:localAnd: 0 operators
dlc:bits.c:293:logicalNeg: 0 operators
dlc:bits.c:307:float_abs: 0 operators
dlc:bits.c:319:float_i2f: 0 operators
dlc:bits.c:333:float_half: 0 operators
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# make
gcc -O -Wall -m32 -lm -o btest bits.c btest.c decl.c tests.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./btest -f is
Greater
Score   Rating  Errors  Function
 3      3       0       isGreater
Total points: 3/3
```

解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

```
int isGreater(int x, int y) {
  int s;
  int sign_x;
  int sign_y;
  int p;
  int q;
  s = (~(((1 << 31)&(x + ~y)) >> 31))& 0x1;
  sign_x = x >> 31;
  sign_y = y >> 31;
  p = s &(!(sign_x ^ sign_y));
  q = (!sign_x)&(!sign_x)&(!!(sign_x ^sign_y));
  return p | q;
}
```

第 8 题：localAnd

题目要求理解

初步判断这一题的函数实现的功能是什么

第一个操作数与第二个操作数的第 n 个字节相与，将结果保存在第一个操作数中，并返回。

编码思路分析

解释一下你是怎么想到这样写的，然后简述一下这样写的正确性

首先提取出 x、y 的第 n 个字节，然后相与，然后将相与的结果与 x 的第 n 个字节交换（利用 byteSwap()函数的原理）。第八题的语法检查及正确性检查如图

```
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc bits.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc -e bits
.c
dlc:bits.c:175:bitNor: 3 operators
dlc:bits.c:185:bitXor: 8 operators
dlc:bits.c:202:allEvenBits: 7 operators
dlc:bits.c:222:byteSwap: 10 operators
dlc:bits.c:241:multFiveEighths: 6 operators
dlc:bits.c:255:conditional: 7 operators
dlc:bits.c:276:isGreater: 20 operators
dlc:bits.c:297:localAnd: 10 operators
dlc:bits.c:311:logicalNeg: 7 operators
dlc:bits.c:331:float_abs: 4 operators
dlc:bits.c:374:float_i2f: 27 operators
dlc:bits.c:417:float_half: 21 operators
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# make
gcc -O -Wall -m32 -lm -o btest bits.c btest.c decl.c tests.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./btest -f lo
calAnd
Score   Rating  Errors  Function
 3      3       0       localAnd
Total points: 3/3
```

解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

```
int localAnd(int x, int y, int n) {
   int s;
   int t;
   int m;
   int p;
   int shift;
   shift = n << 3;
   s = (x >> shift)&0xff;
   t = (y >> shift)&0xff;
   m = s & t;
   p = m << shift;
   return ((s << shift) ^ p) ^x ;
}
```

第 9 题：logicalNeg

题目要求理解

初步判断这一题的函数实现的功能是什么

逻辑取负（取补）。

编码思路分析

解释一下你是怎么想到这样写的，然后简述一下这样写的正确性

取反（也称取补），即对操作数按位取反，末尾加一。设返回值为 y，$y=\sim x+1$。当 x 为 0 时，y 的符号位也为 0；当 x 取最小（0x8000 0000）时，x、y 的符号位都是 1；x、y 的符号位是 0-1、1-0。第九题的语法检查及正确性检查如图

```
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc bits.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc -e bits
.c
dlc:bits.c:175:bitNor: 3 operators
dlc:bits.c:185:bitXor: 8 operators
dlc:bits.c:201:allEvenBits: 7 operators
dlc:bits.c:220:byteSwap: 10 operators
dlc:bits.c:238:multFiveEighths: 6 operators
dlc:bits.c:251:conditional: 7 operators
dlc:bits.c:271:isGreater: 20 operators
dlc:bits.c:282:localAnd: 0 operators
dlc:bits.c:295:logicalNeg: 7 operators
dlc:bits.c:309:float_abs: 0 operators
dlc:bits.c:321:float_i2f: 0 operators
dlc:bits.c:335:float_half: 0 operators
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# make
gcc -O -Wall -m32 -lm -o btest bits.c btest.c decl.c tests.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./btest -f lo
gicalNeg
Score   Rating  Errors  Function
 4      4       0       logicalNeg
Total points: 4/4
```

解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

```
int logicalNeg(int x) {
  int y;
  y = ~x + 1;
  return ((~x & ~y)>> 31)& 0x1;
}
```

第 10 题：float_abs

题目要求理解

初步判断这一题的函数实现的功能是什么

求一个单精度浮点数的绝对值。若此浮点数为 NaN（非数），返回浮点数本身，否则，返回其绝对值。

编码思路分析

解释一下你是怎么想到这样写的，然后简述一下这样写的正确性

当 uf=NaN 时，即阶码全为 1 或者尾数全为 0，直接返回 uf;当 uf!=NaN 时，

符号位取反，返回-uf 即可。第十题的语法检查及正确性检查如图

```
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc bits.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc -e bits
.c
dlc:bits.c:175:bitNor: 3 operators
dlc:bits.c:185:bitXor: 8 operators
dlc:bits.c:201:allEvenBits: 7 operators
dlc:bits.c:220:byteSwap: 10 operators
dlc:bits.c:238:multFiveEighths: 6 operators
dlc:bits.c:251:conditional: 7 operators
dlc:bits.c:271:isGreater: 20 operators
dlc:bits.c:282:localAnd: 0 operators
dlc:bits.c:295:logicalNeg: 7 operators
dlc:bits.c:315:float_abs: 4 operators
dlc:bits.c:328:float_i2f: 0 operators
dlc:bits.c:342:float_half: 0 operators
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# make
make: Nothing to be done for 'all'.
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./btest -f fl
oat_abs
Score   Rating  Errors  Function
 2      2       0       float_abs
```

解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

```
unsigned float_abs(unsigned uf) {
  int s;
  s=uf&(~(1<<31));
  if(s>0x7f800000) {
  return uf;
  }
  else
  return s;
}
```

第 11 题：float_i2f

题目要求理解

初步判断这一题的函数实现的功能是什么

求一个定点数的浮点数（单精度）表示形式。

编码思路分析

解释一下你是怎么想到这样写的，然后简述一下这样写的正确性

首先我们 int 型的范围为-2^31~2^31-1。分为两种情况：用科学计数法表示 int 型数时，尾数位数<=23，例如 0x00008001，此时将 0x8001 左移 8 位得到 frac，而 exp 为 127+16-1；当尾数位数>23 时，找到尾数最末一位，记为 x[i]，然后对尾数的舍去分 3 种情况考虑:要偶端舍入的情况;当 x[i-1]=1 且 x[i-2]、x[i-3]…x[0]不都为 0 的情况;c=0 的情况；其他特殊情况。第十一题的语法检查及正确性检查如图

```
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc bits.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc -e bits
.c
dlc:bits.c:175:bitNor: 3 operators
dlc:bits.c:185:bitXor: 8 operators
dlc:bits.c:202:allEvenBits: 7 operators
dlc:bits.c:222:byteSwap: 10 operators
dlc:bits.c:241:multFiveEighths: 6 operators
dlc:bits.c:255:conditional: 7 operators
dlc:bits.c:276:isGreater: 20 operators
dlc:bits.c:287:localAnd: 0 operators
dlc:bits.c:301:logicalNeg: 7 operators
dlc:bits.c:321:float_abs: 4 operators
dlc:bits.c:364:float_i2f: 27 operators
dlc:bits.c:378:float_half: 0 operators
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# make
gcc -O -Wall -m32 -lm -o btest bits.c btest.c decl.c tests.c
gcc -O -Wall -m32 -o fshow fshow.c
gcc -O -Wall -m32 -o ishow ishow.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./btest -f fl
oat_i2f
Score   Rating  Errors  Function
 4      4       0       float_i2f
```

解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

```
unsigned float_i2f(int x) {
   int _s;
    int i;
    int _exp;
    int _frac;
    int _del;
    int _frac_mk;
    _s=x>>31&1;
    _frac = 0;
    if(x==0)//||x==0x8000000)
        return x;
    else if(x==0x80000000)
        _exp=158;
    else{
        if (_s)
        x = -x;
        i = 30;
        while ( !(x >> i) )
            i--;
            _exp = i + 127;
        x = x << (31 - i);
        _frac_mk = 0x7fffff;
        _frac = _frac_mk & (x >> 8);
        x = x & 0xff;
        _del = x > 128 || ((x == 128) && (_frac & 1));
        _frac += _del;
        if(_frac >> 23) {
            _frac &= _frac_mk;
            _exp += 1;
        }
    }
    return (_s<<31)|(_exp<<23)|_frac;
}
```

第 12 题：float_half

题目要求理解

初步判断这一题的函数实现的功能是什么

求一个浮点数*0.5 的结果，当结果为 NaN 时，返回参数。

编码思路分析

解释一下你是怎么想到这样写的，然后简述一下这样写的正确性

此题同上一题，还是根据浮点数的特点，分成几部分来求。分别对符号位、指数部分、小数部分、偏差、小数部分的掩码等情况进行处理。第十二题的语法检查及正确性检查如图

```
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# gedit bits.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc bits.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc -e bits
.c
dlc:bits.c:175:bitNor: 3 operators
dlc:bits.c:185:bitXor: 8 operators
dlc:bits.c:202:allEvenBits: 7 operators
dlc:bits.c:222:byteSwap: 10 operators
dlc:bits.c:241:multFiveEighths: 6 operators
dlc:bits.c:255:conditional: 7 operators
dlc:bits.c:276:isGreater: 20 operators
dlc:bits.c:287:localAnd: 0 operators
dlc:bits.c:301:logicalNeg: 7 operators
dlc:bits.c:321:float_abs: 4 operators
dlc:bits.c:364:float_i2f: 27 operators
dlc:bits.c:407:float_half: 21 operators
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# make
gcc -O -Wall -m32 -lm -o btest bits.c btest.c decl.c tests.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./btest -f fl
oat_half
Score   Rating  Errors  Function
 4      4       0       float_half
Total points: 4/4
```

解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

```
unsigned float_half(unsigned uf) {
  int _round;
  int _s;
  int _e;
  int _mke;
  int _mkm;
  int _mks;
  int _mkem;
  int _mksm;
  int _tmp;
  _round = !((uf&3)^3);
  _mks = 0x80000000;
  _mke = 0x7F800000;
  _mkm = 0x007FFFFF;
  _mkem= 0x7FFFFFFF;
  _mksm= 0x807FFFFF;
  _e = uf&_mke;
  _s = uf&_mks;
  if (_e==0x7F800000) return uf;
  if (_e==0x00800000){
    return _s | (_round + ((uf & _mkem)>>1)) ;
  }
   if (_e==0x00000000) {
    _tmp = (uf&_mkm)>>1;
    return _s | (_tmp + _round);
  }
   return (((_e>>23)-1)<<23) | (uf & _mksm);
}
```

本地检查情况

语法检查情况

使用`./dlc bits.c`命令对实现代码进行检查，并将检查情况截图粘贴在下方。

```
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./dlc bits.c
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./btest bits.
c
```

结果检查情况

使用`./btest bits.c`命令对实现代码的运行情况进行检查，并将检查情况截图粘贴在下方。

```
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./btest bits.
c
Score   Rating  Errors  Function
 1      1       0       bitNor
 1      1       0       bitXor
 2      2       0       allEvenBits
 2      2       0       byteSwap
 3      3       0       multFiveEighths
 3      3       0       conditional
 3      3       0       isGreater
 3      3       0       localAnd
 4      4       0       logicalNeg
 2      2       0       float_abs
 4      4       0       float_i2f
 4      4       0       float_half
Total points: 32/32
```

完整检查情况

使用`./driver.pl`命令对实现代码进行完全检查和打分，并将检查情况截图粘贴在下方。

```
root@zhaoweikang:/home/zhaoweikang/lab1-handout-rjgc/lab1-handout# ./driver.pl
1. Running './dlc -z' to identify coding rules violations.

2. Compiling and running './btest -g' to determine correctness score.
gcc -O -Wall -m32 -lm -o btest bits.c btest.c decl.c tests.c

3. Running './dlc -Z' to identify operator count violations.

4. Compiling and running './btest -g -r 2' to determine performance score.
gcc -O -Wall -m32 -lm -o btest bits.c btest.c decl.c tests.c


5. Running './dlc -e' to get operator count of each function.

Correctness Results     Perf Results
Points  Rating  Errors  Points  Ops     Puzzle
1       1       0       2       3       bitNor
1       1       0       2       8       bitXor
2       2       0       2       7       allEvenBits
2       2       0       2       10      byteSwap
3       3       0       2       6       multFiveEighths
3       3       0       2       7       conditional

3       3       0       2       20      isGreater
3       3       0       2       10      localAnd
4       4       0       2       7       logicalNeg
2       2       0       2       4       float_abs
4       4       0       2       27      float_i2f
4       4       0       2       21      float_half

Score = 56/56 [32/32 Corr + 24/24 Perf] (130 total operators)
```

思考与体会

简单说一说自己在完成的过程中遇到的困难或者技巧等，或者学到了什么

要熟练掌握位运算、逻辑运算的知识，还要学会使用提供的工具进行语法正确性的检查、结果正确性的检查。对一些特殊情况也要进行处理。