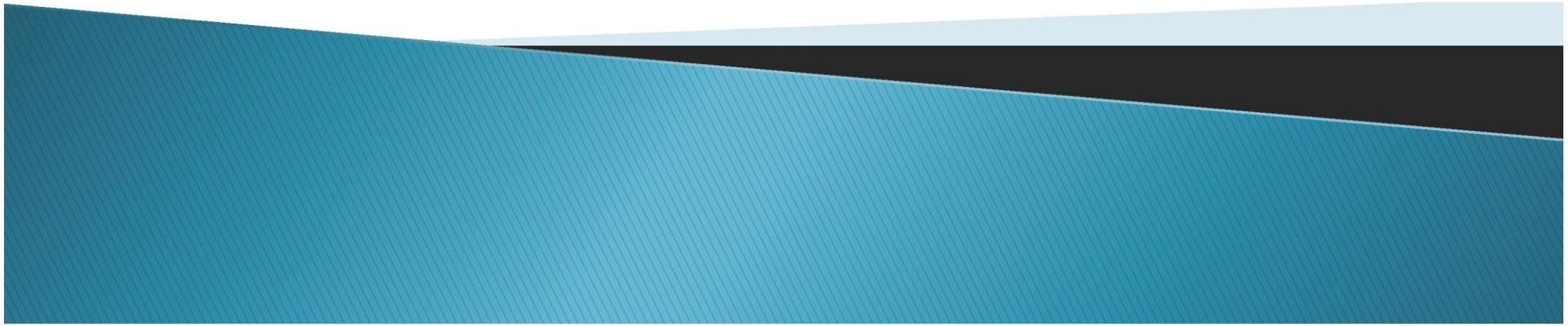
Labl 数据表示



实验概述

- CMU-ICS课程原版实验；教材：《深入理解计算机》
- 实验目的：更好地熟悉和掌握计算机中定点数补码表示和浮点数IEEE754表示，加深对数据二进制编码表示的理解。
- 实验内容：解开一系列编程“难题”——使用有限类型和数量的运算操作实现一组给定功能的函数。然后将完成函数体代码后的bits.c作为实验结果提交。
- 实验环境： linux 32位 gcc32

代码框架

- README　实验细节的说明文件，请仔细阅读。
- bits.c　包含一组用于完成指定功能的函数的代码框架，需按要求补充完成其函数体代码并“作为实验结果提交”。函数的功能和实现要求详见该文件中的注释部分。
- btest.c　实验结果测试工具，用于检查作为实验结果的bits.c中函数实现是否满足实验的功能正确性要求。
- dlc　实验结果检查工具，用于判断作为实验结果的bits.c中函数实现是否满足实验的语法规则要求。
- Makefile　生成btest、fshow、ishow的Make文件。
- ishow.c　整型数据表示查看工具。
- fshow.c　浮点数据表示查看工具。

实验内容

- bits.c函数分类
  - 位操作函数
  - 补码运算函数
  - 浮点数表示函数
- 函数难度分级
  - 1,2,3,4

实验要求

- 除浮点数函数实现外，只能使用顺序程序结构，禁用if, do, while, for, switch等。
- 有限操作类型，！ ~ & ^ | + << >> 各函数不一样
- 常量值范围　0~255 (0x00~0xff)
- 禁用强制类型转换
- 禁用整型外的任何其它数据类型
- 禁用定义和宏，可以定义变量
- 不得定义除已给定的框架函数外的其他函数，不得调用任何函数
- 更多具体要求可参看bits.c各函数框架的注释，以注释为准

浮点数函数规则

- 可以使用循环和条件控制
- 可以使用整型和无符号整型常量及变量（取值不受[0,255]限制）
- 禁用浮点数据类型、struct、union或数组结构
- 浮点数函数均使用unsigned型数据表示浮点数据
- float_neg和float_twice等函数必须能处理全范围的变量值，包括(NaN)和infinity
- 为简化问题，若要返回NaN值，可使用0x7FC0000表示
- 上述要求促使我们必须从二进制位的角度考虑数据，进而能够更清楚地理解数据的二进制表示。

函数原型

```
int Funct(arg1, arg2, ...) {
    int var1 = Expr1;//可以定义变量
       ...
    int varM = ExprM;

    varJ = ExprJ;
       ...
    varN = ExprN;
    return ExprR;
}
```

例子

```
/*   sign - return 1 if positive, 0 if zero,
and -1 if negative
 *   Examples: sign(130) = 1
 *             sign(-23) = -1
 *   Legal ops: ! ~ & ^ | + << >>
 *   Max ops: 10
 *   Rating: 2 */
int sign(int x) {
    return (x>>31) | (!!x);
}
```

代码检查

- 语法检查：使用dlc检查函数实现代码是否符合实验要求中的编码规则（必须通过，否则无法评分）
  - $ ./dlc bits.c　　#简单语法检查
  - $ ./dlc -e bits.c　　#检查操作运算符是否符合需求
  - dlc使用的是开源编译器，能通过gcc编译不一定能通过dlc检查
- 编译生成可执行文件
  - make
  - 修改bits.c必须make，make完成编译，链接，执行文件生成
- 正确性检查：使用btest检查函数实现代码的功能正确性
  - $ ./btest　　#检查bits所有函数功能，失败给出测试用
  - $ ./btest -f byteNot　#检查单个函数，失败给出测试用例
  - $ ./btest -f byteNot -1 0xf -2 1　#规定测试用例检查

代码检查

- 一键检查：

```
$ ./driver.pl
```

- 做的事情：

1.判定编程风格要求；

2.验证执行结果；

3.判定有无不允许使用的操作符；

4.计算性能得分；

5.计算操作数量；

6.展示最终结果。

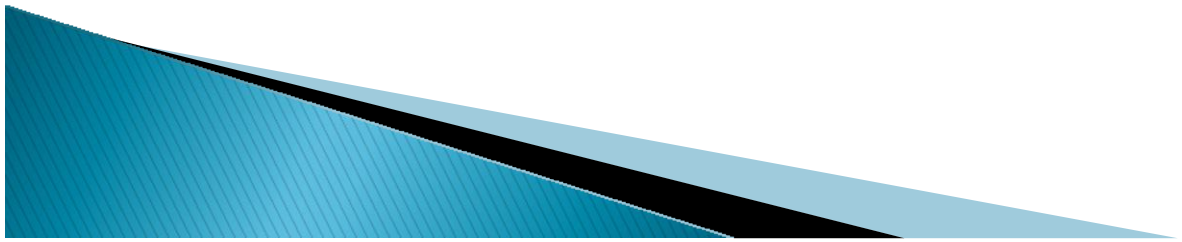
注意事项

- 在bits.c文件中不要包含<stdio.h>头文件，因为这样将给dlc程序造成困难并产生一些难以理解的错误信息。虽然未包含<stdio.h>头文件，但是仍然可以在bits.c中调用printf函数进行调试，gcc给出的警告信息可忽略。
- dlc程序使用比gcc和C++更严格的C变量声明形式。在由“{}”包围的一个代码块中，所有变量声明必须出现在任何非声明语句之前。例如，针对下述代码，dlc将报错：

```
int foo(int x)
{   int a = x;
    a *= 3;      /* Statement that is not a declaration */
    int b = a;   /* ERROR: Declaration not allowed here */
}
```

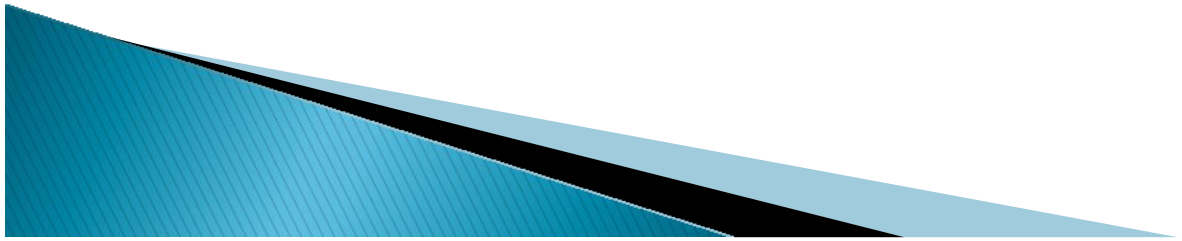
注意事项

- float_neg和float_twice函数的输入参数和返回结果、float_i2f函数的返回结果均为unsigned int类型，但应作为单精度浮点数解释其32 bit二进制表示对应的值。



挑战教授

- 地址： http://49.235.111.14:8080/
- 提交方式：在你的目录下面，运行如下命令
  - ./driver.pl –u “你的学号”
- 注意必须是以你的学号命名，最后提交的报告中要有相应的截图；
- 服务器有点不稳定，如果服务器有故障请在群里说。



结果提交

- 及时备份bits.c
- 最终提交文件必须能通过dlc,btest检查
- 最终提交的项目：

1.学号-姓名-lab1.c（直接传送出虚拟机并更名即可）

2.学号-姓名-lab1.pdf（每个函数需写明解题思路）

3. 学号-姓名-lab1.md （查重备用）

- 排版要求：MarkDown格式
- 提交时间：具体时间确定后群里通知（应该不会晚于10周）
- 注意：采用专门工具进行查重。抄袭风险大，借鉴需谨慎