华中科技大学 计算机学院

[计算机系统基础]

[实验任务书 V1.0]

实验一 数据表示

1. 实验简介

本实验的目的是更好地熟悉和掌握计算机中整数和浮点数的二进制编码表示。实验中，你需要解开一系列编程“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，在此过程中你将加深对数据二进制编码表示的了解。

实验语言：c； 实验环境： linux

2. 实验数据

实验所需要的代码和相关文件已打包成一个 ZIP 文件（[[File:ICS-lab1.zip]]）供下载，其中包含下列文件：

- README 有关实验细节的说明文件，请在开始实验前仔细阅读
- bits.c 包含一组用于完成指定功能的函数的代码框架，需要你按要求补充完成其函数体代码并“作为实验结果提交”。函数的功能与实现要求详细说明在相应函数和文件首部的注释中（务必认真阅读和遵照说明完成实验）。
- bits.h 头文件
- 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。
- btest.h, decl.c, tests.c 生成 btest 程序的源文件
- 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 等工具的 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

3. 实验内容

需要完成 bits.c 中下列函数功能，具体分为三大类：**位操作、补码运算和浮点数操作**。

1）位操作

表 1 列出了 bits.c 中一组操作和测试位组的函数。其中，“级别”栏指出各函数的难度等级（对应于该函数的实验分值），“功能”栏给出函数应实现的输出（即功能），“约束条件”栏指出你的函数实现必须满足的编码规则（具体请查看 bits.c 中相应函数注释），“最多操作符数量”指出你的函数实现中允许使用的操作符的最大数量。

你也可参考 tests.c 中对应的测试函数来了解所需实现的功能，但是注意这些测试函数并不满足目标函数必须遵循的编码约束条件，只能用做关于目标函数正确行为的参考。

表 1 位操作题目列表

级别	函数名	功能	约束条件	最多操作符数
1	lsbZero	将 x 的最低有效位（LSB）清零	仅能使用! ~ & ^ | + << >>	5
2	byteNot	将 x 的第 n 个字节取反（字节从 LSB 开始到 MSB 依次编号为 0-3）	仅能使用! ~ & ^ | + << >>	6
2	byteXor	比较 x 和 y 的第 n 个字节（字节从 LSB 开始到 MSB 依次编号为 0-3），若不同，则返回 1；若相同，则返回 0	仅能使用! ~ & ^ | + << >>	20
3	logicalAnd	x && y	仅能使用! ~ & ^ | + << >>	20
3	logicalOr	x || y	仅能使用! ~ & ^ | + << >>	20
3	rotateLeft	将 x 循环左移 n 位	仅能使用!~ & ^ | + << >>	25
4	parityCheck	若 x 有奇数个 1，则返回 1；否则，返回 0	仅能使用! ~ & ^ | + << >>	20

2）补码运算

表 2 列出了 bits.c 中一组使用整数的补码表示的函数。可参考 bits.c 中注释说明和 tests.c 中对应的测试函数了解其更多具体信息。

表 2 补码运算题目列表

级别	函数名	功能	约束条件	最多操作符数
2	mul2OK	计算 2*x，如果不溢出，则返回 1，否则，返回 0	仅能使用 ~ & ^ | + << >>	20

2	mult3div2	计算(x*3)/2，朝零方向取整	仅能使用! ~ & ^ | + << >>	12
3	subOK	计算 x – y，如果不溢出，则返回 1，否则，返回 0	仅能使用! ~ & ^ | + << >>	20
4	absVal	求 x 的绝对值	仅能使用! ~ & ^ | + << >>	10

3）浮点数操作

表 3 列出了 bits.c 中一组浮点数二进制表示的操作函数。可参考 bits.c 中注释说明和 tests.c 中对应的测试函数了解其更多具体信息。**注意 float_abs 的输入参数和返回结果（以及 float_f2i 函数的输入参数）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。**

表 3 浮点数操作题目列表

级别	函数名	功能	约束条件	最多操作符数
2	float_abs	返回浮点数‘|f|’的二进制表示，当输入参数是 NaN 时，返回 NaN	仅能使用任何整型/无符号整型操作，包括||，&&以及 if，while 控制结构	10
4	float_f2i	返回浮点数‘f’的强制整型转换“(int)f”表示	仅能使用任何整型/无符号整型操作，包括||，&&以及 if，while 控制结构	30

4. 实验要求

实验前请认真阅读本文档和 bits.c 中的代码及注释，然后根据要求相应完成 bits.c 中的各函数代码。

实验中实现的函数代码必须满足下述基本条件（更多具体要求见函数的注释）：

- 除关于浮点数的函数实现外，只能使用顺序程序结构，不得使用循环或条件分支控制程序结构，例如 if, do, while, for, switch 等。
- 仅能使用有限类型和数量的 C 语言算术和逻辑操作，例如如下的操作符，但注意每个题目可能有不同的可用操作符列表，详见具体函数说明。！ ~ & ^ | + << >>
- 不得使用超过 8 位表示的常量（即其值必须位于[0,255]中）。

- 不得使用任何形式的强制类型转换。
- 不得使用除整型外的任何其它数据类型，如数组、结构、联合等。
- 不得定义和使用宏。
- 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。
- 具体的函数功能和实现要求可参看 bits.c 各函数框架的注释，以注释为准。
- 特定于浮点数操作函数的额外限制条件：
  - 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）；
  - 不使用任何浮点数据类型、操作及常量。
  - 关于浮点数的函数实现可以使用标准的程序结构(选择、循环均可使用)，可以使用 int 和 unsigned 两种整型数据，不得使用浮点数据类型、struct、union 或数组结构。关于浮点数的函数均使用 unsigned 型数据表示浮点数据。
  - float_abs 和 float_half 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。

5. 代码检查

如前所述，实验数据包中包含两个工具程序可帮助检查你的代码的正确性。

1）使用 dlc 检查函数实现代码是否符合实验要求的编码规则

完成 bits.c 后，调用如下命令进行检查：

```
$ ./dlc bits.c
```

如果 dlc 发现了错误，例如出现不允许使用的操作符、过多数量的操作符或者非顺序的代码结构，则将返回错误信息。如果程序代码满足要求，dlc 将不输出任何提示。

使用-e 选项调用 dlc

```
$ ./dlc -e bits.c
```

可使 dlc 打印出每个函数使用的操作符数量。

输入“ ./dlc -help”可打印出 dlc 的可用命令行选项列表。

2）使用 btest 检查函数实现代码的功能正确性

首先使用如下命令编译生成 btest 可执行程序：

```
$ make
```

如下调用 btest 命令检查 bits.c 中所有函数的功能正确性：

```
$ ./btest
```

注意每次修改 bits.c 后都必须使用 make 命令重新编译生成 btest 程序。

为方便依次检查测试每一函数的正确性，可如下在命令行使用“-f”选项跟上函数名，以要求 btest 只测试所指定的函数：

```
$ ./btest -f byteNot
```

进一步可如下使用“-1, -2, -3”等选项在函数名后输入特定的函数参数：

```
$ ./btest -f byteNot -1 0xf -2 1
```

（README 文件中有关于 btest 程序的使用说明）

6. 建议与提示

1）如果你的代码不能完全满足相应函数的操作符使用限制，你可以获得部分得分，但是往往这样的次优解总能找到改进它的方法，从而获得正确解答。

2）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。注意尽管未包含<stdio.h>头文件，你仍然可以在 bits.c 中调用 printf 函数进行调试，gcc 将打印警告信息但你可以忽略它们。

3）注意 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 */
}
```

你必须类似如下代码将变量声明放在最前：

```
int foo(int x)
{    int a = x;
     int b;
     a *= 3;
     b = a;
}
```

7. 实验结果提交

请将完成函数体代码后的 bits.c 按以下命名规范命名后作为实验结果提交。**本次实验不需要提交实验报告，但各函数实现中应有详细注释描述解题思路**。按班为单位提交电子版本压缩文件。

- **专业命名规范**

  信安 IS　物联网 IT　计算机 CS　卓越班 ZY　ACM 班 ACM

- **文件命名规范**

  CS1201_U201214795_姓名.c

- **电子版按班为单位集中打包发送至 130757@qq.com 谭志虎老师处归档**