作业1： Amdahl定律、数的表示

520030910281 肖真然

1. **思考题**

要求：下面的子问题需要书面完成，可以手写后扫描、也可以直接编辑本文件回答问题。

1、假定某个程序的代码可并行化部分是98%，提供给你多个处理器核(core)来将问题并行化。如果并行化后加速比要达到7以上，至少需要多少处理器核将问题并行化处理。

**解：**

解得，故至少需要8个处理器核将问题并行化处理才能使加速比达到7以上。

**2、以下是两段C语言代码，函数arith( )是直接用C语言写的，而optarith( )是对arith( )函数以某个确定的M和N编译生成的机器代码反编译生成的。根据optarith( )，可以推断函数arith( ) 中M和N的值各是多少？**

**#define M**

**#define N**

**int arith (int x, int y)**

**{**

**int result = 0 ;**

**result = x\*M + y/N;**

**return result;**

**}**

**int optarith ( int x, int y)**

**{**

**int t = x;**

**x << = 2;**

**x - = t;**

**if ( y < 0 ) y += 15;**

**y>>4;**

**return x+y;**

**}**

**解：**M = 3，N = 16.

3、假设我们在对有符号值使用补码运算的32位机器上运行代码。对于有符号值使用的是算术右移，对无符号值使用的是逻辑右移。变量的声明和初始化如下：

int x = foo(); //调用某某函数，给x赋值

int y = bar(); //调用某某函数，给y赋值

unsigned ux = x;

unsigned uy = y;

对于下面每个C表达式

证明对于所有的x和y 值，都为真（等于1）；或者（2）给出使得它为假的x和y值；

1. (x>0) || (x-1<0)
2. (x \* x) >= 0
3. x<0 || -x<=0
4. x>0 || -x>=0
5. x+y == uy+ux
6. x\*~y + uy\*ux == -x
7. x\*4 + y\*8 == (x<<2)+(y<<3)
8. ((x>>2)<<2)<=x

**解：**

A. 当x≤0且x-1≥0时为假，即x=1000 0000…（31个0），x-1=0111 1111 …（32个1）时为假。

B. 当x = 50000时，x\*x的结果的32位二进制码为1001 0101 0000 0010 1111 1001 0000 0000，符号位为1，故x\*x≥0为假。

C. 当x＜0时，原式为真；当x≥0时，x符号位为0，则-x要么符号位为1，要么-x=0，故-x≤0，原式为真。故原式永真。

D. 当x=-2^31时，x＜0，-x＜0，原式为假。

E. 因为x与ux，y与uy在物理层面上编码相同，且有符号数与无符号数共用一个加法器，所以原式永真。

F. x\*~y=x\*(-y-1)+ux\*uy=x\*(-y)-x+ux\*uy=-x，故原式永真。

G. 计算机内部编译器会将乘2^k解释为左移，所以原式永真。

H. 右移后再左移对高位无影响，对于低位，若右移损失的低位全为0，则再左移后所得结果还是与原x相等，若右移损失的低位中有1，则再左移后所得结果小于原x，故原式永真。

4、给定：

* int 、unsigned int长度为 32 bits.
* float 是32位 IEEE 754 单精度浮点数，double是64位 IEEE 754 双精度浮点数
* 变量之间的装换如下：

/\* Create some arbitrary values \*/

int x = random();

int y = random();

int z = random();

/\* Convert to other forms \*/

unsigned ux = (unsigned) x;

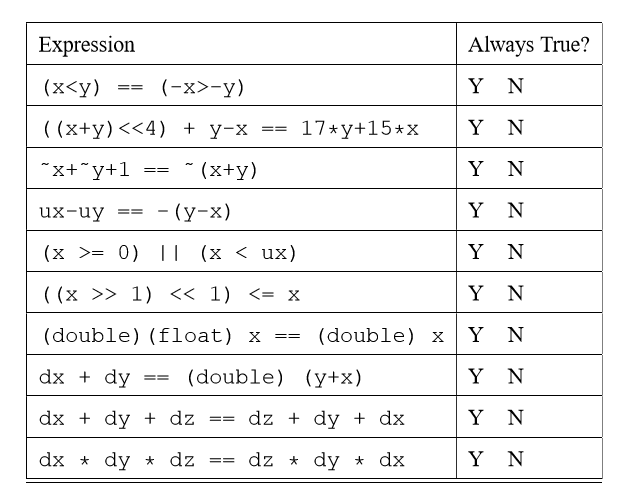
unsigned uy = (unsigned) y;

double dx = (double) x;

double dy = (double) y;

double dz = (double) z;

以下表达式，哪些恒定为true？是的话圈 “Y”，不是的话圈“N” ，并指出原因，给出反例。



**解：**

1. N，当x=2^31-1，y=-2^31时，原式左值为0，右值为1，原式为假。

2. Y，乘法与左移等价。

3. Y，[x]补 + [y]补 = [x + y]补 等价于 ~x + 1 + ~y + 1 = ~(x + y) + 1 等价于原式。

4. Y，无符号数与有符号数在内存中编码一致且共用一套加法器。

5. N，当x<0时，原式左值为0，右值为0，原式为假。

6. Y，右移后再左移对高位无影响，对于低位，若右移损失的低位全为0，则再左移后所得结果还是与原x相等，若右移损失的低位中有1，则再左移后所得结果小于原x。

7. N，当x=2^31-1时，原式为假。因为转换为float有精度损失。

8. N，当x=2^31-1，y=2^31-1时，原式为假。因为double类型的位数更多，dx+dy不会溢出，而y+x可能溢出，溢出后再转换成double类型就与dx+dy不一样了。

9. Y，double类型位数多，做加法不会溢出。

10. N，当x = 2147483647，y = 2199999999，z = 2148888888时，原式为假。因为乘法中间可能会有溢出与舍入，所以交换顺序后溢出与舍入可能不一样。计算机输出结果：左值=96550093426447996883 右值=96550093426447985888

1. **实践题：位级运算、数的编码**

要求：不需要提交代码，只需要在报告中，把你的运行结果、以及你实现的五个函数的源代码贴上来。

先下载datalab.tar 文件

#解压缩：

**$ tar -xvf datalab.tar**

#进入文件夹

**$ cd datalab**

查看文件夹下的文件

**$ ls**

执行命令

**$ make clean**

**$ make all**

运行：

**$ ./fshow 一个浮点数例如34.5**

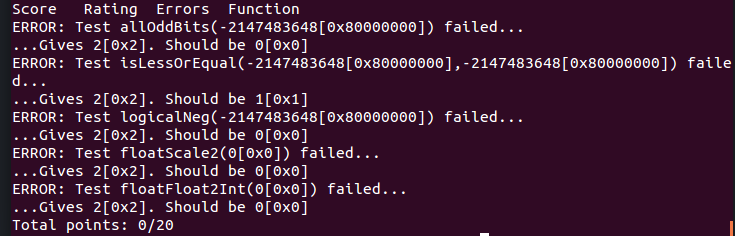
你可以看到这个数的float机器编码

**$ ./ishow 一个整数例如 20**

你可以看到这个数的十六进制的机器编码

**$ ./btest**

你会看到程序错误的提示：

****

你需要做的是：修改bit.c 文件中的几个函数，完成规定的功能，仔细阅读bit.c 中各函数前的注释，了解各函数应该能达到的功能。这些函数有：

// 判断整数x的所有奇数位是否都为1

// 可以使用的运算符：~ & ^ | + << >>,运算符最多可以用多少次参见程序注释

int allOddBits(int x) {

return 2;

}

//使用位级运算符实现判断整数x<= y

//可以使用的运算符：~ & ^ | + << >>,运算符最多可以用多少次参见程序注释

int isLessOrEqual(int x, int y) {

return 2;

}

//使用位级运算求逻辑非 !

//可以使用的运算符：~ & ^ | + << >>,运算符最多可以用多少次参见程序注释

int logicalNeg(int x) {

return 2;

}

//求2乘一个浮点数，

//可使用任意整数的合法运算符，例如: &, |, ^ , ||, &&, +，-， if, while

//运算符最多可以用多少次参见程序注释

unsigned floatScale2(unsigned uf) {

return 2;

}

// 将浮点数uf转换为整数，返回其32位的位级表达

// 可使用任意整数的合法运算符，例如: &, |, ^ , ||, &&, +，-， if, while

// 运算符最多可以用多少次参见程序注释

int floatFloat2Int(unsigned uf)

{

return 2;

}

关于浮点数的编码，[IEEE-754 Floating Point Converter (h-schmidt.net)](https://www.h-schmidt.net/FloatConverter/IEEE754.html) 这里有一个转换器，希望对你有帮助。

修改程序后重新编译：

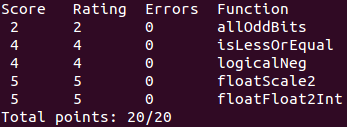
**$ make clean**

**$ make all**

执行：

**$ ./btest**

如果你的实现全部正确，应该得到以下结果，这5个函数的得分情况如下，满分为20分。



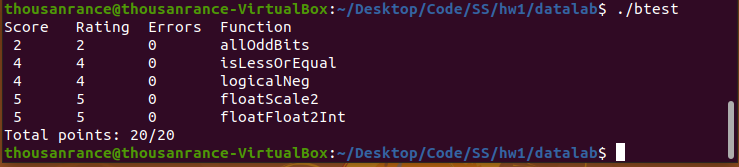
如果有扣分，说明函数实现没有符合要求，使用了不允许使用的运算符。使用命令

**$ ./dlc bits.c**

可以调用文件包中提供的规则检查器，检查哪个运算符是不合规定的。

注：本实验选自CMU CSAPP，如果你想了解CMU CSAPP：Datalab 的完整要求，可以查看：<http://csapp.cs.cmu.edu/3e/labs.html> 找到相关的文档和代码。

运行结果：



源代码：

