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。

**答案： 1/(0.02 + 0.98/x) >= 7 x>= 7.9 取8**

**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;**

**参考答案：**

**可以看出x\*M和“int t = x; x << = 2; x-=t;”三句对应，这些语句实现了x乘4的功能（左移2位相当于乘以4，然后再减1），因此，M等于3；**

**y/N与“if ( y < 0 ) y += 15; y>>4;”两句对应，功能主要由第二句“y右移4位”实现，它实现了y除以16的功能，因此N=16。而第一句“if ( y < 0 ) y += 15;”主要用于对y为负数时进行调整。**

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

A 假。设 x 等于-2147483 648（TMin32），那么，我们有 x-1 等于 2147483647（TMax32）

B 假。当 x 为 65535（0xFFFF）时，x\*x 为-131071（0xFFFE0001）

C 真。如果 x 是非负数，则-x 是非正的

D 假。设 x 为-2 147 483 648（TMin32），那么 x 和-x 都为负数

E 真。补码和无符号数乘法有相同的位级行为，而且他们是可交换的

F 真。~y 等于-y-1。uy\*uy 等于 x\*y，因此等式左边等价于 x\*-y-x+x\*y

G 真。算术左移运算与乘法运算等价

H 真。因为右移总是向负无穷大方向取整

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;

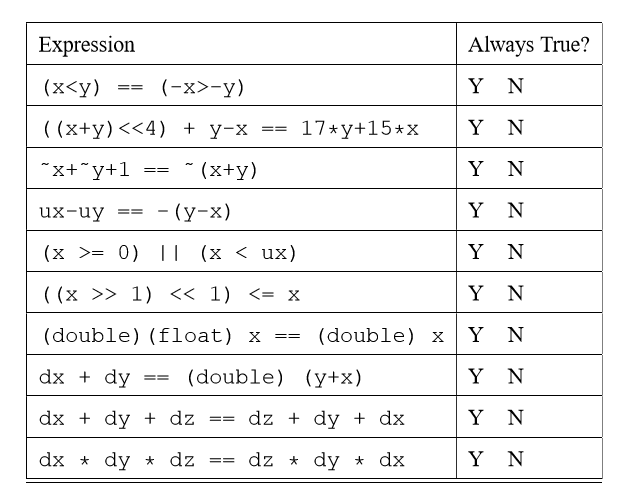
unsigned uy = (unsigned) y;

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。



Answers:

Expression Always True?

1. (x<y) == (-x>-y) No: Let x = Tmin, y = 0

2. ((x+y)<<4) + y-x == 17\*y+15\*x Yes: Associative, commutative, distributes

3. ~x+~y+1 == ~(x+y) Yes: (-x-1)+(-y-1)+1 == -(x+y)-1

4. ux-uy == -(y-x) Yes:

5. (x >= 0) || (x < ux) No: x = -1. Comparison x < ux is never true.

6. ((x >> 1) << 1) <= x Yes: x>>1 rounds toward minus infinity.

7. (double)(float) x == (double) x No: Try x = Tmax.

8. dx + dy == (double) (y+x) No: Try x=y=Tmin.

9. dx + dy + dz == dz + dy + dx Yes: Within range of exact representation by double's.

10. dx \* dy \* dz == dz \* dy \* dx No: Try dx=Tmax, dy=Tmax-1, dz=Tmax-2

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**

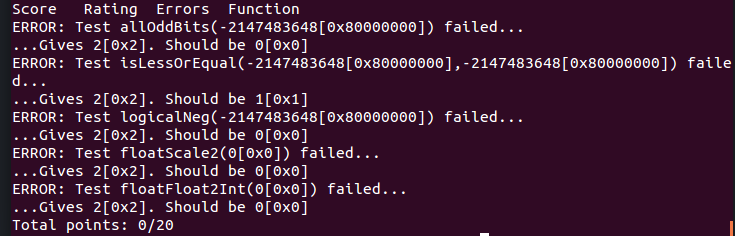
你可以看到这个数的float机器编码

**$ ./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：~ & ^ | + << >>

int allOddBits(int x) {

int mask = 0xAA+(0xAA<<8);

mask=mask+(mask<<16);

return !((mask&x)^mask);

}

//使用位级运算符实现判断整数x<= y

//可以使用的运算符：~ & ^ | + << >>

int isLessOrEqual(int x, int y) {

int sign = 1 << 31;

int signx = !(x & sign); // positive or zero is 1, negetive is 0

int signy = !(y & sign);

int diff = y + ~x + 1; // diff = y - x

int sameSign = !(signx ^ signy);

int lessEq = sameSign & !(diff & sign); // 符号相等 且 y-x >= 0

return (!signx & signy) | lessEq;

}

//使用位级运算求逻辑非 !

//可以使用的运算符：~ & ^ | + << >>

int logicalNeg(int x) {

return ((x|(~x+1))>>31)+1;

}

//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

unsigned floatScale2(unsigned uf) {

int sign = uf & 0x80000000;

int exp = uf & 0x7f800000;

int frag = uf & 0x007fffff;

if (exp == 0) {

//非规格化的数

return sign | frag << 1;

}

if (exp == 0x7f800000) { // inf or NaN

return uf;

}

// 规格化的数

exp += 0x0800000; // 指数加1，相当于 乘二

if (exp == 0x7f800000) {

// inf

frag = 0;

}

return sign | exp | frag;

}

// 将浮点数uf转换为整数的，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

int floatFloat2Int(unsigned uf)

{

int sign = uf & 0x80000000;

int exp = ((uf & 0x7f800000) >> 23) - 127; // 规格化数的指数的真值（非规格化数一律返回0）

int frag = (uf & 0x007fffff) | 0x00800000; // 补上前导1

int absval;

if (exp < 0) {

return 0;

}

if (exp > 30) {

return 0x80000000;

}

if (exp < 23) {

// 需要截断部分尾数

absval = frag >> (23 - exp);

} else {

absval = frag << (exp - 23);

}

return sign == 0x80000000 ? -absval : absval;

}

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：

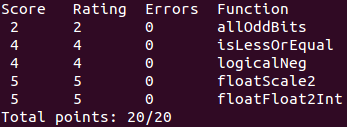
**$ make clean**

**$ 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。