武汉大学计算机学院 2020-2021 学年第一学期期末考试试卷

课程名称：《 系统级程序设计 》（ A 卷）　　授课教师：_____

年级：_____　　专业：_____层次：本科

姓名：_____学号：_____

说明：1、答案一律书写在答题纸上，书写在试卷上或其他地方一律无效。

2、请准确规范书写姓名和学号，否则视为答卷作废。

1. 假设在一个 int 类型为 32 位长度的机器上运行程序。Float 类型长度 32 位，double 类型长度 64 位。我们产生随机数 x, y 和 z，并把它们转换为 float/double 类型的值。对于右图中每个 C 表达式，请指出表达式是否总是成立。如果总是成立，那么请描述其中的数学原理；否则，列举出一个使它不成立的反例。(10 分)

int x = random();
int y = random();
int z = random();
unsigned ux = (unsigned) x;
unsigned uy = (unsigned) y;
float fx = (float) x;
double dx = (double) x;
double dy = (double) y;
double dz = (double) z;
	1) (x >=0) || (x < ux)
	2) ((x >> 1) << 1) $\leq$ x
	3) x = = (int) (float) x
	4) (dx * dy) * dz == dx * (dy * dz)
	5) dy > fx $\Longrightarrow$ - fx > - dy

答案：

1. 错. 当 x>=0 时表达式值为 1；当 x=Tmin 时，x 被 casting 为 Tmax+1>0，所以 x<ux 也为假，整个表达式取值为 0

2. 对. 有符号数算术右移向负无穷舍入，导致结果变小. (有符号数除法舍入问题)

3. 错，整数转换为 float 可能会有舍入

4. 错（dx=Tmax32, dy=tmax32-1, dz=tmax32-2），

5. 对

2. 假设一个 int 类型长度 32 位的机器上，整型值以补码表示。MAX_INT 是最大的整数，MIN_INT 是最小的整数。假设 x 和 y 都是有符号整数，x=56, y= -82。请填写下表，指明各表达式的字节值。注：在该机器上整数 17 的字节值表示为 0x00000011。(8 分）

表达式	值	表达式	值
x & y		x && y	
x | y		x || y	
x & !y		x && ~y	
1 + (x << 3) + ~x		~(~x |(y ^ (MIN_INT+ MAX_INT)))	

答案

X = 0x38, Y=0xFFFFFFAE

表达式	值	表达式	值
x & y	0x00000028	x && y	0x00000001
x | y	0xffffffbe	x || y	0x00000001
x & !y	0x00000000	x && ~y	0x00000001
1 + (x << 3) + ~x	0x00000188	~(~x |(y ^ (MIN_INT+ MAX_INT)))	0x00000028

3. 下面两个基于 IEEE 浮点格式的 9 位表示：

格式 A	格式 B
1. 有一个符号位 2. 有 k=5 个阶码位。阶码偏置值是 15. 3. 有 n=3 个小数位	1. 有一个符号位 2. 有 k=4 个阶码位。阶码偏置值是 7. 3. 有 n=4 个小数位

请将下面给出的格式 A 的位模式，转换为最接近的格式 B 的值。如果需要舍入，你要向+∞舍入。另外，给出用格式 A 和格式 B 表示的位模式对应的值：要么是整数，要么是小数（例如 17/64）。请写清楚格式转换的计算过程。（共 12 分）

格式 A		格式 B	
位	值	位	值
1 01110 001	-9/16	1 0110 0010	-9/16
0 01000 011			
1 11000 000			

答案：

格式 A		格式 B	
位	值	位	值
0 01000 011	11/1024	0 0000 1011	11/1024
1 11000 000	-512	1 1110 1111	-248

4. 假设某些内存地址和寄存器中的值如下表所示。

内存地址	值	寄存器	值
0x100	0x67	%rax	0x100
0x102	0xAB	%rcx	0x2
0x104	0x1F	%rdx	0x4
0x106	0xBD	---	---
0x108	0x49		
0x10A	0xEF		
0x10C	0x8C		

填写下表，给出所示操作数的值（共 10 分）

操作数	值
%rax	
$0x10A	
4(%rax)	

262(%rcx,%rdx)	
0x2(%rax, %rcx, 3)	

操作数	值	注释
%rax	0x100	寄存器
$0x10A	0x10A	立即数
4(%rax)	0x1F	地址 0x104
262(%rcx,%rdx)	0x8C	地址 0x10C
0x2(%rax, %rcx, 3)	0x49	地址 0x108

5. 请根据汇编代码及 C 语言函数框架写出对应的函数内部 C 语言程序语句（12 分）：

汇编代码	C 语言函数框架

arith:
leaq (%rdi,%rsi), %rax
addq %rdx, %rax
leaq (%rsi,%rsi,2), %rdx
salq $3, %rdx
leaq 6(%rdi,%rdx), %rcx
imulq %rcx, %rax
ret
	long arith(long x, long y, long z)
	{
	.....
	return rval;
	}

答案：

```
long arith(long x, long y, long z)
{
  long t1 = x+y;
  long t2 = z+t1;
  long t3 = x+6;
  long t4 = y * 24;
  long t5 = t3 + t4;
  long rval = t2 * t5;
  return rval;
}
```

6. 请根据汇编代码及 C 语言函数框架写出对应的函数内部 C 语言程序语句（12 分）：

汇编代码	C 语言函数框架

absdiff:
pushq %rbp
movq %rsp, %rbp
movq %rdi, -24(%rbp)
movq %rsi, -32(%rbp)
	long absdiff(long x, long y)
	{
	long result;
	.....
	return result;

movq -24(%rbp), %rax
cmpq -32(%rbp), %rax
jle .L2
movq -24(%rbp), %rax
subq -32(%rbp), %rax
movq %rax, -8(%rbp)
jmp .L3
.L2:
movq -32(%rbp), %rax
subq -24(%rbp), %rax
movq %rax, -8(%rbp)
.L3:
movq -8(%rbp), %rax
popq %rbp
ret
 	
sum_absdiff:
pushq %rbp
movq %rsp, %rbp
subq $32, %rsp
movq %rdi, -24(%rbp)
movq %rsi, -32(%rbp)
movq $0, -16(%rbp)
movq $0, -8(%rbp)
.L6:
movq -24(%rbp), %rax
andl $1, %eax
movq %rax, -8(%rbp)
movq -32(%rbp), %rdx
movq -8(%rbp), %rax
movq %rdx, %rsi
movq %rax, %rdi
call absdiff
addq %rax, -16(%rbp)
sarq -24(%rbp)
cmpq $0, -24(%rbp)
jne .L6
movq -16(%rbp), %rax
leave
ret
	}
	
	long sum_absdiff(long x， long y)
	{
	long result = 0;
	long temp=0;
	
	.....
	return result;
	}

答案：

```
long absdiff(long x, long y)
{
```

```
    long result;
    if (x > y)
        result = x-y;
    else
        result = y-x;
    return result;
}

long sum_absdiff(long x，long y) {
  long result = 0;
  long temp=0;
  do {
    temp=x & 0x1;
    result +=absdiff(temp, y);
    x >>= 1;
  } while (x);
  return result;
}
```

7. 右图是一个通用结构的 C 函数 switcher：请根据以上汇编代码和跳转表填写 C 代码中缺失的部分。除了情况标号 C 和 D 的顺序之外，将不同情况填入这个模板的方式是唯一的（10 分）。

汇编代码	跳转表	C 语言函数框架

void switcher(long a, long b,
long c, long *dest)
// a in %rdi, b in %rsi,
// c in %rdx, dest in %rcx
switcher：
cmpq $7, %rdi
ja .L2
jmp *.L4(,%rdi, 8)
.section .rodata
.L7：
xorq $14, %rsi
movq %rsi, %rdx
.L3:
leaq 122(%rdx), %rdi
jmp .L6
.L5:
leaq (%rdx, %rsi), %rdi
salq $1, %rdi
jmp .L6
.L2:
movq %rsi, %rdi
	.L4:
	.quad .L3
	.quad .L2
	.quad .L5
	.quad .L7
	.quad .L6
	.quad .L2
	.quad .L2
	.quad .L5
		void switcher(long a, long b,
		long c, long* dest)
		{
		long val;
		switch (a)
		{
		case _____: /*Case A*/
		c = _____;
		/*Fall through*/
		case _____: /*Case B*/
		val = _____;
		break;
		case _____: /*Case C*/
		case _____: /*Case D*/
		val = _____;
		break;
		case _____: /*Case E*/
		val = _____;
		break;
		default:
		val = _____;

.L6:
movq %rdi, (%rcx)
ret
		}
		*dest = val;
		}

答案：

```
void switcher(long a, long b, long c, long* dest)
{
    long val;
    switch (a)
    {
    case 3:                     /*Case A*/
        c = b ^ 14;
        /*Fall through*/
    case 0:                     /*Case B*/
        val = c + 122;
        break;
    case 2:                     /*Case C*/
    case 7:                     /*Case D*/c
        val = (c + b) << 1;
        break;
    case 4:                     /*Case E*/
        val = a;
        break;
    default:
        val = b;
    }
    *dest = val;
}
```

8. 右图是数组操作的部分 C 语言代码，左图是对应的汇编代码。在汇编代码中，最终读取 P[i][j] 和写入 Q[j][i]的汇编指令对应的行号分别是第几行和第几行？M 和 N 分别是多少？（6 分）

汇编代码	C 语言函数框架

```
 6 swapInc:
 7 .LFB0:
 8         .cfi_startproc
 9         movslq  %edi, %rdi
10         movslq  %esi, %rsi
11         leaq    Q(%rip), %rdx
12         leaq    (%rdi,%rdi,4), %rax
13         leaq    (%rdi,%rsi,2), %rcx
14         addq    %rax, %rsi
15         leaq    P(%rip), %rax
16         movl    (%rax,%rsi,4), %eax
17         addl    $1, %eax
18         movl    %eax, (%rdx,%rcx,4)
19         xorl    %eax, %eax
20         ret
21         .cfi_endproc
```

```
#define M _
#define N _

int P[M][N];
int Q[N][M];

int swapInc(int i, int j)
{
        Q[j][i] = P[i][j] + 1;
        return 0;
}
```

分值：1+1+2+2

16，18

M：2

N：5

9. 请介绍缓冲区溢出攻击的基本原理，并分别阐述课本上所提及的三种方法能够限制缓冲区溢出攻击的理由。（12 分）

分值：3+3+3+3

原理：精心设计 shell code（exploit code + padding + address of exploit code， AEC）；通过字符串读取函数，将 shell code 传递给缓冲区，覆盖函数的返回地址并将其更新为 exploit code 的地址；函数返回时，会跳转到 exploit code。

方法 1：栈地址随机化。让 AEC 难以确定。

方法 2：栈地址没有执行权限。让 exploit code 无法执行。

方法 3：canary value。通过检查 canary value 是否被修改，来检查返回地址是否被修改。

10. 作为软件工程系学生，请结合本课程的学习，从程序员的角度讨论改善代码性能的编程原则。请阐述 2 条以上的高性能编程原则，并分别阐述这些原则能够改善性能的理由。（8 分）

分值：4+4

参考（多选 2）

1. 数组元素替换成临时变量。

2. 将循环中的函数调用提到循环外面。

3. 右结合展开。

4. 多个临时变量循环展开。