计算机科学与技术学院 2019-2020 学年第 2 学期　考试试卷

计算机系统基础（开卷）

2020.6.29

□ **考试纪律：　独立完成，如发现抄袭或作弊，将按纪处理！**

□ **诚信承诺：　自觉遵守考试纪律，独立完成考试**

不代考，不询问和抄袭他人答案。

□ **答题要求：　用 A4 白色打印纸作答**

一、（15 分，每小题 5 分）简单题

（1）什么是指令集体系结构（ISA）？为什么说 ISA 是划分计算机系统中软件和硬件的界面？说明理由。

（2）计算机在控制器的控制下完成程序的执行。在程序执行的过程中，IR 和 PC 两个寄存器的区别是什么？指令执行一般包含哪几个阶段？

（3）在 IEEE 754 浮点数运算中，为什么要对浮点数进行规格化？有哪两种规格化操作？

二、（18 分）数的表示和运算

设一个 4 位数正整数 B=204x（十进制），其中 x 是你学号的最后一位，如，若学号为 U201812345，x=5，则 B=2045。

根据你的学号生成的 B=_____

回答以下问题：

（1）请分别用 32 位补码整数和 IEEE 754 单精度浮点格式表示 B。

要求：写出必要的计算过程（如浮点数的符号位、阶码、尾数的计算过程，补码整数的换算过程，机器数的字节表示等。用**十六进制**表示。5 分）。

（2）在上述 B 的两种表示里，哪段二进制位序列在两种表示中完全相同？为什么会相同？（3 分）

（3）设程序中有三个变量 x、y、i，其中 x 和 y 是 float 型变量，i 是 16 位 short 型变量（用补码表示）。在程序执行的某一时刻有 x = 0.75、y = -B、i= B（注：B 就是上面根据你的学号生成的 4 位数正整数），它们都被写到按字节编址的主存中，其地址分别是 100、108 和 112。请在下表中分别填写在大端模式机器和小端模式机器上变量 x、y 和 i 中每个字节在主存存放位置的内容（与变量存储不相关的位置不需填写或填写‘/’，但若填写了其它带有错误信息的数据则视为填错，本小题 6 分）。

地址	大端机	小端机
100		
101		
102		
103		
104		
105		
106		
107		
108		
109		
110		
111		
112		
113		
114		

（4）对（3）中的 x 和 y，采用 IEEE 754 单精度浮点数格式计算 x+y 的值。要求按照 IEEE 754 单精度浮点数加减运算的规则写出具体的计算过程（4 分）。

三、（4 分）遵循 Lab1 的规则要求，回答以下问题。

Lab1 的基本规则：只能使用顺序程序结构；仅能使用限定类型和数量的 C 语言算术和逻辑操作（详见各函数说明）；不得使用超过 8 位表示的常量、强制类型转换、数组/结构/联合等数据类型、宏等；不得定义或调用其他函数等。

函数名	功能	约束条件	最多操作符数量
int isNonZero(int x)	如果 x 不等于 0 返回 1，否则返回 0	仅能使用 ~ & ^ | + << >>	10

写出函数 isNonZero 的实现

```
int isNonZero(int x)
{



}
```

四、（16 分）已知递归函数 refunc 的 C 语言代码框架及其对应的汇编代码如下所示：

```
int refunc(unsigned x)
{
    unsigned nx;
    int rv;
    if(      ①      )
        return    ②    ;
    nx=      ③      ;
    rv=refunc(nx);
    return        ④        ;
}

在main函数中对refunc函数调用：
int main(void)
{
    printf("%d",32-refunc(A));
    return 0;
}
```

```
1 refunc:
2    push   %ebp
3    mov    %esp,%ebp
4    sub    $0x18,%esp
5    cmpl   $0x0,0x8(%ebp)
6    jne    .L1
7    mov    $0x0,%eax
8    jmp    .L2
9    .L1;
10   mov    0x8(%ebp),%eax
11   shr    %eax
12   mov    %eax,B(%ebp)
13   sub    $0xc,%esp
14   pushl  B(%ebp)
15   call   refunc
16   add    $0x10,%esp
17   mov    %eax,C(%ebp)
18   mov    0x8(%ebp),%eax
19   and    $0x1,%eax
20   mov    %eax,%edx
21   mov    C(%ebp),%eax
22   add    %edx,%eax
23   .L2:
24   leave
25   ret
```

请回答以下问题：（填空题每空 1 分）

1）根据 refunc 函数对应的汇编代码填写其 C 代码中缺失的部分（注：汇编代码中的 **B**、**C** 是需要你在后面的第 3）问中计算的数据，你可以先看一下后面两问，然后再来做本小题）。

①：_____

②：_____

③：_____

④：_____

2）main 函数中的 **A** 是一个整数，是由你的学号的倒数第 3 位到倒数第 1 位共三个数字组成的整数，如，若学号是 U201812345，则 **A** 的值等于 345.

依据你的学号，你的 **A** = ⑤_____

3）汇编代码中 **B** 和 **C** 是**偏移量，**功能是按照(%ebp+偏移)方式分别访问 refunc 函数中非静态局部变量 nx、rv 在栈帧中的存储单元。设编译器的规则是：rv 的地址=nx 的地址+0x4。

此处 **B** 的绝对值=24-(**A**%5)*4。如，若学号是 U201812345，则|**B**|=24。

请根据你的学号，计算 **B** 和 **C** 的值，然后写出汇编代码中第 12 行和第 17 行的完整形式：

第 12 行：⑥_____

第 17 行：⑦_____

4）汇编代码中的第 4 行、第 16 行的作用分别是什么？

第 4 行：⑧_____

第 16 行：⑨_____

5）执行该程序（以你的 **A** 从 main 函数开始执行程序），则程序的输出是：

⑩_____

6）将 main 函数中对 refunc 函数的首次调用称为“第一次执行 refunc 函数”，并设此时刻（指**在 main 的汇编例程中执行 call refunc 指令的那一刻**）有：

R[ip]=0x08048431,　R[ebp]= 0xbc000a40,　R[esp]= 0xbc000a20。

称在“第一次执行 refunc 函数”中对 refunc 函数的(第一次)递归调用为“第二次执行 refunc 函数”。

基于上面计算出来的**A**、**B**、**C**和 ip、ebp、esp 寄存器内容的设定，并设 refunc 汇编例程里入口地址是 0x080483db，试画出“第二次执行 refunc 函数”时，**在执行完 refunc 汇编代码的第 3 行指令、但还未执行第 4 条指令时的那一刻**栈帧的内容。填写下表作答（6 分。对有内容的单元，要填写其地址和具体内容；对尚在等待填写内容的单元，填写其地址，并在内容中填问号“？”；对**没有内容的空闲单元**，可以不填。这里的地址均指的当前 4 字节单元的 LSB 的地址）。

序号	地址	单元内容（4 字节/单元）
1	/	
2	0xbc000a20	
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		

要求：

（1）main 函数的栈帧内容只需给出调用 refunc 函数时为 refunc 准备的参数和返回地址两个单元的内容即可；

（2）对“第一次执行 refunc 函数”，需要给出完整的栈帧内容；

（3）对“第二次执行 refunc 函数”，只需给出 refunc 汇编代码中第 4 条指令前（不包括第 4 条）的栈帧内容即可。

注：（1）以上表格的行数可以根据实际需要增加或减少。

五、（8 分）假设函数 sumij 的 C 代码如下，其中，M 和 N 是用#define 声明的常数。

```
1   int a[M][N], b[N][M];
2   int sumij(int i, int j)
3   {
4         return a[i][j] + b[j][i];
5   }
```

已知函数 sumij 的过程体对应的汇编代码如下：

```
…
1   movl      8(%ebp), %ecx
2   movl      12(%ebp), %edx
3   movl      %ecx, %eax
4   sall      $m, %eax
5   subl      %ecx, %eax
6   addl      %edx, %eax
7   movl      %edx, %ebx
8   leal      (%ebx, %ebx, n), %ebx
9   addl      %ebx, %edx
10  addl      %ecx, %edx
11  movl      a(,%eax, 4), %eax
12  addl      b(,%edx, 4), %eax
…
```

回答以下问题：

1）请使用第五题第 2)问中的 A 计算本题的 m 和 n 的值（十进制）。

m = 1+A%8 = _____

n = 1+A%11 = _____

2）根据 m 和 n 的值，确定汇编代码中的 M 和 N 的值，写出必要的推导过程。

六、（14 分）以下是缓冲区溢出攻击实验（lab3）的 fizz 阶段，fizz 函数的 C 语言代码以及由可执行目标代码 bufbomb 反汇编得到的 fizz 和 getbuf 例程的汇编代码（注：**这里的反汇编代码经过了一定的改造，请根据改造后的代码回答问题**）：

fizz 函数的 C 语言代码：

```
void fizz(int val)
{
    if (val == cookie) {
        printf("Fizz!: You called fizz(0x%x)\n", val);
        validate(1);
    }
    else printf("Misfire: You called fizz(0x%x)\n", val);
    exit(0);
}
```

fizz 例程的汇编代码：

```
1  08048bca <fizz>:
2   8048bca:  55                      push   %ebp
3   8048bcb:  89 e5                   mov    %esp,%ebp
4   8048bcd:  83 ec 18                sub    $0x18,%esp
5   8048bd0:  8b 45 08                mov    0x8(%ebp),%eax
6   8048bd3:  3b 05 20 c2 04 08       cmp    0x804c220,%eax
7   8048bd9:  75 1e                   jne    8048ce9 <fizz+0x2f>
8   8048bdb:  89 44 24 04             mov    %eax,0x4(%esp)
9   8048bdf:  c7 04 24 2e a1 04 08    movl   $0x804a12e,(%esp)
10  8048be6:  e8 f5 fb ff ff          call   80488d0 <printf@plt>
11  8048beb:  c7 04 24 01 00 00 00    movl   $0x1,(%esp)
12  8048bf2:  e8 5d 06 00 00          call   8049344 <validate>
13  8048bf7:  eb 10                   jmp    8048cf9 <fizz+0x3f>
14  8048bf9:  89 44 24 04             mov    %eax,0x4(%esp)
15  8048bfd:  c7 04 24 c4 a2 04 08    movl   $0x804a2c4,(%esp)
16  8048c04:  e8 d7 fb ff ff          call   80488d0 <printf@plt>
17  8048c09:  c7 04 24 00 00 00 00    movl   $0x0,(%esp)
18  8048c10:  e8 8b fc ff ff          call   8048990 <exit@plt>
```

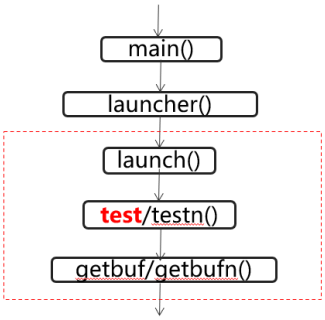
getbuf 例程的汇编代码：

```
1  080491ec <getbuf>:
2    80491ec:  55                    push   %ebp
3    80491ed:  89 e5                 mov    %esp,%ebp
4    80491ef:  83 ec 38              sub    $0x18,%esp
5    80491f2:  8d 45 d8              lea    -0x10(%ebp),%eax
6    80491f5:  89 04 24              mov    %eax,(%esp)
7    80491f8:  e8 55 fb ff ff        call   8048d52 <Gets>
8    80491fd:  b8 01 00 00 00        mov    $0x1,%eax
9    8049202:  c9                    leave
10   8049203:  c3                    ret
```

假设 makecookie 采用了一种新的算法生成 Cookie：取你的学号的后 8 位数字，当作一个十六进制的 32 位整数，以此作为 Cookie 来进行 fizz 阶段的实验。

如：makecookie U201812345 的输出是：0x01812345

提示：bufbomb 中函数之间的调用关系是：



回答以下问题：

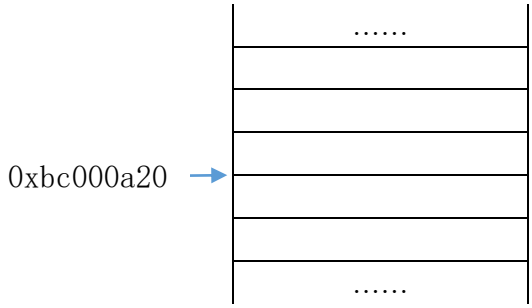
1）简述实现 fizz 阶段“攻击”的基本思想（5 分）。

2）根据以上汇编代码和 cookie 的值，构造一个可以实现 fizz 阶段“攻击”的攻击字符串（5 分）。

3）设某次执行，在执行到 getbuf()函数时，栈帧有以下情形：

请结合你的攻击字符串，给出控制转向 fizz 后，刚好执行完 fizz 汇编例程中的第 3 行指令（mov 指令）时的栈帧的主要内容（填写下面的表格，4 分）：

要求：只给出与“攻击”有关的栈帧单元的内容即可，其它单元的内容可以不管。



刚好执行完 fizz 第 3 行指令时栈帧的主要内容

此时 ebp 的内容是：

R[ebp]=_____

七、（10 分）现有两个程序模块 func1.c 和 func2.c，以及两个静态函数库 libx.a 和 liby.a，假设有如下的函数调用关系：

func1.o→func2.o,

func1.o→libx.a 中的函数 mylib1( )

func2.o→liby.a 中的函数 mylib2( )。

此外，libx.a 中的 mylib1→liby.a 中的函数 foo( );

liby.a 中的 mylib2→libx.a 中的函数 bar( )。

```
/*func1.c*/
# include < libx.h>
void main()
{
    func2();
    mylib1();
}
```

```
/*func2.c*/
# include <liby.h>
void func2()
{
    mylib2();
}
```

（1）假设需要链接生成的目标文件为 myfunc，请写出能够正确进行静态链接的命令（4 分）；

（2）请按步骤写出静态链接过程中符号解析的过程（6 分）。

八、假设一个 c 语言程序有两个模块: main.c 和 swap.c, 对它们单独编译，分别生成对应的可重定位目标文件 main.o 和 swap.o。

（1）main.c 和 swap.c 的代码如下：

main.c

```
int buf[2] = {1, -4};
extern void swap( );
int sum=0;
int main( )
{
        swap( );
        return 0;
}
int ave( ) {
    int a;
    static int count=2;
    sum=buf[0]+buf[1];
    a=sum/count;
    return a;
}
```

swap.c

```
extern int buf[];
extern int ave();
int *bufp0 = &buf[0];
static int *bufp1;
int sum;
void swap( ) {
        int temp;
        bufp1 = &buf[1];
        temp = *bufp0;
        *bufp0 = *bufp1;
        *bufp1 = temp;
       ave( );
}
```

对于 swap.o，填写下表中各符号的情况。首先说明各个符号是否出现在 swap.o 的符号表中，若是的话，再说明定义该符号的模块是 main.o 还是 swap.o？该符号的类型是全局、外部还是本地？该符号出现在 swap.o 中的哪个节（.text, .data 或.bss）？（5 分）

符号	是否在 swap.o 的符号表中	定义模块	符号类型	所在的节
buf				
bufp0				
bufp1				
sum				
swap				
temp				
ave				

（2）swap.o 的反汇编代码如下

```
00000000 <swap>:
   0:   55                      push   %ebp
   1:   89 e5                   mov    %esp,%ebp
   3:   83 ec 18                sub    $0x18,%esp
   6:   c7 05 00 00 00 00 04    movl   $0x4,0x0        # 重定位位置①
   d:   00 00 00
            8: R_386_32.bss
            c: R_386_32 buf
  10:   a1 00 00 00 00          mov    0x0,%eax        # 重定位位置②
            11: R_386_32   bufp0
  15:   8b 00                   mov    (%eax),%eax
  17:   89 45 f4                mov    %eax,-0xc(%ebp)
  1a:   a1 00 00 00 00          mov    0x0,%eax        # 重定位位置③
            1b: R_386_32   bufp0
  1f:   8b 15 00 00 00 00       mov    0x0,%edx
            21: R_386_32   .bss                      # 重定位位置④
  25:   8b 12                   mov    (%edx),%edx
  27:   89 10                   mov    %edx,(%eax)
  29:   a1 00 00 00 00          mov    0x0,%eax        # 重定位位置⑤
            2a: R_386_32   .bss
  2e:   8b 55 f4                mov    -0xc(%ebp),%edx
  31:   89 10                   mov    %edx,(%eax)
  33:   e8 fc ff ff ff          call   34 <swap+0x34>   # 重定位位置⑥
            34: _____⑥_____
  38:   90                      nop
  39:   c9                      leave
  3a:   c3                      ret
```

假定可执行目标文件里 swap 函数代码的起始地址是 0x0804842d，swap 函数代码紧接在 ave 函数代码的后面，且 ave 函数代码占 0x2e 个字节。对 swap.o 中的重定位位置⑥进行重定位。回答：此处待重定位的符号是什么、重定位类型是什么？重定位前的值及该值的含义是什么?重定位后的值是什么（需要给出计算过程）及重定位后最终指向的虚拟地址是多少？（10 分）