南京邮电大学 2023/24 学年 第 2 学期

《计算机系统基础 I》期末试卷（A）

院(系)_____ 班级_____ 学号_____ 姓名_____

注意：答案必须写在答题纸上，写在本试卷上无效

得分

一、单项选择题(每题两分，共二十分）

1. 下列选项中，属于指令集体系结构名称的是（）

A. Linux B.x86-64 C.Ubuntu D.Windows

2. 以下是关于补码特点的描述，错误的是（）

A. 0 的表示是唯一的。

B. 加法运算时，符号位可以和数值位一起参加运算

C. -a 和 a 的编码仅符号位不一样。

D. 补码所对应的真值范围是不对称的，负数个数比正数多一个。

3. C 语言代码：

Unsigned short usi=32768;

Short si=usi;

执行上述程序段后，si 的值是（）

A.65530 B.-65530 C.32768 D.-32768

4.下面舍入方式中，（ ）方式更能避免偏向偏差。

A. 向偶数舍入

B. 向零舍入

C. 向下舍入

D. 向上舍入

5.-1028.0 采用 IEEE 754 单精度浮点格式表示的结果（十六进制）是（ ）。

A. C4808000H

B. C4C04000H

C. 44804000H

D. 44C04000H

6.在可重定位目标文件和可执行目标文件描述中，错误的是（ ）。

A. 可执行目标文件有程序头表，而可重定位目标文件没有
B. 可重定位目标文件可能有重定位节，而可执行目标文件没有
C. 可执行目标文件的 .text 和可重定位目标文件的 .text 节内容完全相同
D. 可执行目标文件和可重定位目标文件都有 ELF Header

7. 以下是一段 C 语言过程调用代码：

```
        Long caller()
{
        Long a=1;
        Float b=12.34;
        Test(a,&a,b,&b);
        Return a*b;
}
```

当程序执行到调用程序 test 时，实参 a,&a,b,&b 分别存储在哪些寄存器中（）
A. rdi ,rsi ,rdx ,rex
B. Rdi ,rsi ,xmm0, xmm1
C. Rdi ,rsi ,xmm0, rdx
D. Rdi, rsi ,xmm0 ,rex

8. 设 p.o →libx.a →liby.a→libx.a→p.o，其中 a→b 表示 a 依赖于 b ，下列命令中，使得静态链接器能解析所有的符号引用, 并且是最小的命令行（即一个含有最少数量的目标文件和库参数的命令)的是( )。
A. linux> gcc p.o libx.a liby.a
B. linux> gcc p.o libx.a liby.a p.o
C. linux> gcc p.o libx.a liby.a libx.a p.o
D. linux> gcc p.o libx.a liby.a libx.a

9 指令序列如下（左边为指令地址，中间为机器代码，右边为汇编指令）；

```
804857b：48 39 c2          cmpq    %rax, %rdx

804857e:    76 f8              jbe xxxxxxxx
```

若执行到上述 cmpq.指令时，rdx 寄存器值是 80，rax 寄存器值是 68，则 jbe 指令执行后将会执行()处指令。
A.8048572 B.8048576 C.8048578 D.8048580

10. 设小端法规则机器中，有一个 int 型变量 x 的地址为 0x8000fff0，x=0x80901020，则地址为 0x8000fff2 单元中存放的内容的十六进制表示为( )
A.20 B.10 C.90 D.80

得分

二、分析题（每题 5 分，共 20 分）

1. 假设 ax 的值=0xffd0， bx 的值=0x8005，执行指令“addw %bx, %ax”后，寄存器的内容和标志寄存器备标志位 ZF，CF，OF，SF 的值分别是 0x_____、_____、_____、_____和_____。

2.假设某源文件 p.c 中定义“static int buf[4]=(50, 20, -1, 8), static int *bufp1= &buf[2]”，则生成的可重定位文件 p.o.中 buf 被定义在_____节中，共占_____个字节，bufp1 共占_____个字节，bufp1 的重定位信息在_____节中。设 buf 起始地址为 0x4011C0，则&buf[2]是 0x_____。

3.假设下面的值存放在指定的内存单元和寄存器中，填写下面各条指令执行结果。

内存单元地址	值
0x100	0xff
0x104	0xab

寄存器	值
rax	0x100
rbx	0xf0

指令	指令执行结果
leaq 7(%rax,%rbx,4),%rcx	rcx =_____
movq $0x100,%rax	Rax=_____
addl 0x100,%ebx	ebx =_____
salb $2,%bl	bl =_____
orw $0xc003,%bx	bx =_____

4. 请分析整数除 0 会发生异常，而浮点数除 0 不会出现异常的原因。

得分

三、综合应用题

一个 C 语言程序有两个源文件：main.c 和 sum.c,他们的内容如下图所示。
设在 x86-64/linux 平台上（采用小端法规则），用 GCC 编译驱动程序处理。

```
/*main.c/
#include <stdio.h>
int sum(int *a, int n);
static int array[10] = { 3, -7, 6, 14, -5, 9, 1, 3, 16, 22 };
int result;

void main()
```

```
{
    int num;
    scanf("%d", &num);
    int val = sum(array, num);
    printf("val = %d result = %d\n", val, result);
}
```

```
/*sum.c*/
extern int result;

int sum(int *a, int n)
{
    int i, s = 0;
    result = 1;
    for (i = 0; i < n; i++) {
        s += a[i];
        result *= a[i];
    }
    return s;
}
```

Main.c 和 sum.c 的可重定位目标文件名分别是 main.o 和 sum.o，生成的可执行文件名为 sum。
使用”objdump-d sum”得到可执行目标文件 sum 的反汇编部分结果如下。

```
0000000000401176 <main>:
  401176:   f3 0f 1e fa                endbr64
  40117a:   48 83 ec 18                sub    $0x18,%rsp
  40117e:   64 48 8b 04 25 28 00       mov    %fs:0x28,%rax
  401185:   00 00
  401187:   48 89 44 24 08             mov    %rax,0x8(%rsp)
  40118c:   31 c0                      xor    %eax,%eax
  40118e:   48 8d 74 24 04             lea    0x4(%rsp),%rsi
  401193:   48 8d 3d 6a 0e 00 00 lea    0xe6a(%rip),%rdi
  40119a:   e8 e1 fe ff ff             call   401080 <__isoc99_scanf@plt>
  40119f:   8b 74 24 04                mov    0x4(%rsp),%esi
  4011a3:   48 8d 3d b6 2e 00 00 lea    0x2eb6(%rip),%rdi        # 404080 <array>
  4011aa:   e8 _ _ _ _                 call   4011e7 <sum>
  4011af:   89 c2                      mov    %eax,%edx
    ...........
00000000004011e7 <sum>:
  4011e7:   f3 0f 1e fa                endbr64
  4011eb:   c7 05 97 2e 00 00 01 movl   $0x1,0x2e97(%rip)        # 40408c <result>
  4011f2:    00 00 00 00
  4011f5:    b9 00 00 00 00             mov    $0x0,%ecx
  4011fa:    ba 00 00 00 00             mov    $0x0,%edx
  4011ff:    eb 18                      jmp    401219 <sum+0x32>
```

```
401201:    48 63 c2                 movslq %edx,%rax
401204:    8b 04 87                 mov    (%rdi,%rax,4),%eax
401207:    01 c1                    add    %eax,%ecx
401209:    0f af 05 7c 2e 00 00     imul   0x2e7c(%rip),%eax     # 40408c <result>
401210:    89 05 76 2e 00 00        mov    %eax,0x2e76(%rip)     # 40408c <result>
401216:    83 c2 01                 add    $0x1,%edx

401219:    39 f2                    cmp    %esi,%edx
40121b:    7c e4                    jl     401201 <sum.loop>

40121d:    89 c8                    mov    %ecx,%eax
40121f:    c3                       ret
```

采用 gdb 调试程序，当 CPU 准备运行地址 4011aa 处的 call 指令，此时部分调试信息如下:
(gdb) i r rip rsp rdi rsi (查看寄存器内容)。
Rip 0x4011aa rsp 0x7fffffde00 rdi 0x404080 rsi 0x4。

基于给出的代码以及给出调试信息，回答下列问题或完成下列任务。

1.写出从 C 语言源程序到生成可执行文件 sum 的命令行，并回答从 C 语言源程序到可执行文件的转换需要经过哪些步骤?。

2.填写下表中各标识符的情况，说明每个标识符是否出现在 main.o 的符号表(.symtab 节)中，如果是的话，进一步说明定义该符号的模块是 main.o 还是 sum.o、该符号的类型是全局、外部还是局部符号。

标识符	是否在 main.o 的符号表中？	定义模块	符号类型
Sum			
array			
result			
num			

3 库函数 scanf 和 printf 中格式符(如“%d”)在哪个节定义？

4.程序调试执行时，输入的 num 的值是多少？程序屏幕输出的结果是什么？

5. 计算机按字节编址，0x404080 开始的 8 个连续地址单元中存放的内容是什么？（4 分）
(gdb)x/8xb 0x404080 (按 8 位一个字节显示）
0x404080: _____

6. 4011aa 处 call 指令的操作码是 0xe8,其余字节是偏移量，偏移量采用补码表示，转移的目标地址采用相对寻址方式。重定位表如下所示。

readelf -r main.o

Relocation section 'rela.text' contains 6 entries:

Offset Info Type Sym. Value Sym. Name + Addend

000000000035 000900000004 R _X86_ 64_ _PLT32 000000000000000 fun-4

分析该 call 指令的机器指令有几个字节?写出该 call 指令的完整机器指令。

4011aa: e8___ ___ ____ ____ call4011e7 <sum>

7.当 CPU 运行 4011aa 处的 call 指令后，此时 rip 和 rsp.寄存器的值是多少?此时栈顶单元内容是什么? (均用 十六进制表示)。

(gdb)x/xg $rsp

8. 当 i=2 时，执行 401204 处的“mov (%rdi,%rax,4),%eax”指令后，eax 寄存器中的内容是什么?

9. CPU 正在执行 401219 处“cmp %esi,%edx”，假设此时 edx 寄存器的值是 2,则该指令执行后，esi 和 ecx 寄存器中的内容分别是什么?

10. sum 函数中哪两条指令实现了 result*= a[i]的功能?。

11.40117e 和 401187 两条指令的作用是什么?。

得分

四、实验题

甲同学正在完成缓冲区溢出实验，下面是可执行目标文件 target 的反汇编部分结果：

```
0000000401de1 <getbuf>:
401de1:          f3 0f 1e fa              endbr64
```

```
401de5:          48 83 ec 28              sub    $0x18%rsp

401de9:          48 89 e7                 mov    %rsp,%rdi

401dec:         e8 ad 02 00 00            call     40209e <Gets>

401df1:          b8 01 00 00 00           mov    $0x1 ,%eax

401df6:         48 83 c4 28               add     $0x18,%rsp

401dfa:         c3                                ret



000000000401 dfb <bomb>:
401dfb:        f3 0f 1e fa                        endbr64

401dff:         50                                  push       %rax
......
```

1. 帮助甲同学设计字符串输入给 target,造成缓冲区溢出，进而使程序转向执行 bomb 子程序。输入字符串可以用十六进制表示字符的 ASCII 码，字节间用空格隔开，
如：68 ef cd ab 00 83 c0。
2. 简述有哪些方法可以对抗缓冲区溢出攻击。