武汉大学计算机学院

《计算机系统基础》期末考试试卷

姓名_____ 学号_____

（注：① 闭卷考试；② 考试时间为 120 分钟；③ 所有解答必须写在答题纸上。）

注意：
本次试卷全部题目为计算题或论述题，在给出结论的同时，必须给出详细分析和计算过程，否则不能得分。

本试卷上的所有代码，如未特别说明，均在x86-64/Linux机器上编译、链接和运行。

一、 （6分）

某同学用C语言位运算实现了如下函数：

```
1  int funx(int x, int y) {
2      int z=x+y;
3      int a=(x>>31)&0x1;
4      int b=(y>>31)&0x1;
5      int c=(z>>31)&0x1;
6      return !(((!(a^b))&(a^c))&(b^c));
7  }
```

(1) 请问该函数的功能是什么？（3分）

(2) 请使用更少的运算符来实现同样的功能，要给出C语言代码及使用的操作符数量。（3分）（**要求**：(i) **允许**使用的操作符为：**!**、~、**&**、^、|、+、<<、>>；(ii) **不允许**使用控制结构（例如if，while，do～while，for，switch等）；其他要求与Datalab相同，此处不一一详述）

二、 (4分)

有如下所示C语言代码，编译执行结果为输出`0.000000`和`1.000000`。请解释输出这两个值的原因。

```
1  int main ()
2  {
3      int a = 1;
4      float *p = (float*) &a;
5      printf("%f\n", *p);
6      printf("%f\n", (float)(a));
7      return 0;
8  }
```

三、 （16分）

请根据部分C语言源代码（如下图左列所示）和对应的汇编代码（如下图右列所示）回答下列问题。

```
1  typedef struct {
2      ____①____ index;
3      ____②____ x[____③____];
4  } a_struct;

5  typedef struct {
6      int first;
7      a_struct a[____④____];
8      int last;
9  } b_struct;

10 void fun(long i, b_struct *bp) {
11     int n = bp->first + bp->last;
12     a_struct *ap = &bp->a[i];
13     ap->x[ap->index] = n;
14 }
```

```
1  fun:
2      movl    368(%rsi), %edx
3      addl    (%rsi), %edx
4      leaq    0(,%rdi,8), %rax
5      leaq    (%rax,%rdi), %rcx
6      movq    %rcx, %rax
7      addq    8(%rsi,%rcx,8), %rax
8      movslq  %edx, %rdx
9      movq    %rdx, 16(%rsi,%rax,8)
10     ret
```

(1)参数i和bp分别放在哪两个寄存器中？（2分）

(2)movslq指令的功能是什么？（2分）

(3)代码中①～④处分别是什么内容？(①②处为变量类型，③④处为常量)（4分）

(4)sizeof(a_struct)和sizeof(b_struct)的值分别为多少？（4分）

(5)汇编代码中哪些指令会访问内存？（以行号表示，多答、错答均相应扣分）（4分）

四、 （16分）

阅读下面给出的汇编代码和跳转表，填写C语言代码中缺失的部分。

```
1   int switcher (int x, int y, int z) {
2       int result = 0;
3       switch (x) {
4           case ____①____
5               result = x - y;
6               break;
7           ____②____
8           ____③____
9               result = ____④____;
10          ____⑤____
11              result = ____⑥____;
12              break;
13          ____⑦____
14              result = y;
15              break;
16          default:
17              result = ____⑧____;
18              break;
19          }
20          return result;
21  }
```

```
1   switcher:
2       leal    -17(%rdi), %eax
3       cmpl    $6, %eax
4       ja      .L8
5       movl    %eax, %eax
6       jmp     *.L4(,%rax,8)
7   .L4:
8       .quad   .L7
9       .quad   .L6
10      .quad   .L5
11      .quad   .L8
12      .quad   .L5
13      .quad   .L8
14      .quad   .L3
15  .L3:
16      movl    %esi, %eax
17      ret
18  .L6:
19      movl    $0, %edx
20  .L5:
21      leal    (%rdx,%rdi), %eax
22      ret
23  .L7:
24      movl    %edi, %eax
25      subl    %esi, %eax
26      ret
27  .L8:
28      movl    $20, %eax
29      ret
```

五、 （20分）

根据C语言代码（如下图左列所示）及其运行时反汇编代码（如下图右列所示），回答下列问题。

```
int series(int n)
{
    if (    ①    )
      return    ②    ;
    else if (    ③    )
        return    ④    ;
    else
        return    ⑤    ;
}
```

```
Dump of assembler code for function series:
1  0x55555555473a <+0>:   test   %edi,%edi
2  0x55555555473c <+2>:   je     0x55555555476f <series+53>
3  0x55555555473e <+4>:   cmp    $0x1,%edi
4  0x555555554741 <+7>:   jne    0x555555554749 <series+15>
5  0x555555554743 <+9>:   mov    $0x1,%eax
6  0x555555554748 <+14>:  retq
7  0x555555554749 <+15>:  push   %rbp
8  0x55555555474a <+16>:  push   %rbx
9  0x55555555474b <+17>:  sub    $0x8,%rsp
10 0x55555555474f <+21>:  mov    %edi,%ebx
11 0x555555554751 <+23>:  lea    -0x1(%rdi),%edi
12 0x555555554754 <+26>:  callq  0x55555555473a <series>
13 0x555555554759 <+31>:  lea    (%rax,%rax,1),%rbp
14 0x55555555475d <+35>:  lea    -0x2(%rbx),%edi
15 0x555555554760 <+38>:  callq  0x55555555473a <series>
16 0x555555554765 <+43>:  add    %rbp,%rax
17 0x555555554768 <+46>:  add    $0x8,%rsp
18 0x55555555476c <+50>:  pop    %rbx
19 0x55555555476d <+51>:  pop    %rbp
20 0x55555555476e <+52>:  retq
21 0x55555555476f <+53>:  mov    $0x0,%eax
22 0x555555554774 <+58>:  retq
```

(1) 汇编代码第8行为什么要将%rbx入栈？可以把第8行和第18行一起删除吗？（2分）

(2) 汇编代码中%rbp寄存器是作为帧指针使用的吗？（2分）

(3) C语言代码中缺失的①～⑤处的内容分别是什么？（10分）

(4) 假设以n=3调用函数series(),进入函数时rsp的值为0x7fffffffe7a8，画出第二次调用series(1)时（即执行到0x55555555473a处指令时）的栈状态。（6分）（**要求：**标识出rsp当前指向的位置、从0x7fffffffe7a8到当前栈顶的地址及对应的内容（给出值的含义或数值均可））

六、 （16分）

张三同学写了下面的prefixSum代码，但是好像效率不太高。请你帮他重写代码并满足以下条件：

（1）不要重复从内存中检索p[i]的值。

（2）使用3次循环展开来减少每个循环中的操作。

（3）使用三路并行。

```
1   /*Compute prefix sum of vector v*/
2   void prefixSum(float v[], float p[], long int n)
3   {
4       long int i
5       p[0] = v[0];
6       for( i = 1; i < n; i++ )
7           p[i] = p[i-1] + v[i];
8   }
23
```

七、 （22分）

有如下两个C语言程序float.c和infloat.c。

```
1   /* float.c */
2   #include <stdio.h>
3   extern void infloat(float *);
4   float f0=3.0;
5   int   i0;
6   int main()
7   {
8       static float f1=1.0;
9       float f2;
10      infloat(&f2);
11      printf("%f\n", f2);
12      return 0;
13  }
```

```
1   /* infloat.c */
2   void infloat(float *f)
3   {
4       *f = 2.0;
5   }
```

(1)运行gcc -c float.c得到float.o文件。请在下表中说明每个标识符是否出现在该 .o 文件的符号表（.symtab节）中，如果不在，后面的TYPE、BIND和Section以"-"标识；如果在，则说明它的Type、Bind，以及它所存放的位置Section。(12分)

标识符	.symtab条目？(Y/N)	Type (NOTYPE、OBJECT或FUNC)	Bind (LOCAL或GLOBAL)	Section (.text、.data、.bss或UND)
f0				
f1				
f2				
i0				
main				
infloat				

(2)将float.o和infloat.o链接后得到可执行程序a.out，main函数的部分汇编代码如下：

```
1  main:
2      push   %rbp
3      mov    %rsp,%rbp
4      sub    $0x10,%rsp
5      mov    %fs:0x28,%rax
6      mov    %rax,-0x8(%rbp)
7      xor    %eax,%eax
8      lea    -0xc(%rbp),%rax
9      mov    %rax,%rdi
10     callq  0x5555555546aa <infloat>
```

假设在执行第2行语句之前，rsp的值为0x7fffffffe768，那么执行到第10行语句时，变量f2对应的浮点数存放地址是什么？（以十六进制表示）infloat()函数返回之后，从该地址开始的连续4个字节空间的内容分别是多少？（要求给出每个字节地址的内容，以十六进制表示）（6分）

(3)运行时用户空间的布局如下，请说明变量f0和f2对应的数据的存放位置，函数printf()和infloat()对应的代码的存放位置。（4分）