**华东师范大学期中考试试卷**

2018－2019学年第二学期

课程名称： 计算机系统 课程性质：专业必修

学生姓名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 学 号：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

专 业：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 年级/班级：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

一、选择题 (20分)

1. 代码和数据被加载到( )，处理器就开始执行main程序中的机器语言指令。

(a) 主存 (b) 寄存器 (c) 程序计数器 (d) 磁盘

2. 当调用malloc这样的C标准库函数时，( )可以在运行时动态的扩展和收缩。

(a) 共享库 (b) 栈 (c) 堆 (d) 虚拟存储器

3. CPU中有一个程序计数器（又称指令计数器）。它用于存储（ ）

(a) 保存将要提取的下一条指令的地址

(b) 保存当前CPU所要访问的内存单元地址

(c) 暂时存放ALU运算结果的信息

(d) 保存当前正在执行的一条指令

4. 当一个有符号数和一个无符号数执行运算后的结果是( ) 。

(a) 有符号数 (b) 无符号数 (c) 非法运算 (d) 不一定

5. 变量x、f、d分别为int，float，double类型的，除了f和d不能等于无穷大和NAN以外值可以任意选取，下面表达式为真的是（ ） 。

(a) x==(int)(float)x

(b) d==(double)(float)d

(c) x==(int)(double)x

(d) (f+d)-f==d

6. testq %rax, %rax

jne 3d<function+0x3d>

当%rax为（ ）时，会引起程序跳转。

(a) 1 (b) 0 (c) %rax任意值 (d) %rax没有合适值能满足指令跳转

7. 使用小端法的机器中，int x=0x123456;语句的高地址字节是（ ）。

(a) 0x12 (b) 0x00 (c) 0x34 (d) 0x56

8. 在X86 64 Linux系统中，（ ）占用内存中字节最多

(a) int \*c (b) char a[7] (c) short b[3] (d) float d

9. %rsp的值为0x108, popq %rax指令后%rsp的值为（ ）

(a)0x108 (b)0x100 (c)0x110 (d)0x104

10. 下面指令中错误的是( )。

(a) movl 8(%rdi),%eax

(b) movq 0(%rax,%rcx,4),%rdx

(c) movq %rax,8(%rsp)

(d) movl 4(%rdx),(%rax)

二、（6分）假设一种整数为8位字长，根据第一列的描述，填写其对应的值。

|  |  |
| --- | --- |
| 描述 | 值(十进制表示) |
| Umax | 255 |
| Tmin | -128 |
| (unsigned) ((int) -7) | 249 |
| (((unsigned) 0x81) << 1) & 0xff) | 2 |
| 12 && 4 | 1 |
| (! 0x15) > 16 | 0 |

三、(20分)考虑基于IEEE浮点格式标准的8位浮点表示（符号位1位，阶码3位，尾数4位）。

该8位浮点数数值为V＝（-1）S×1.M×2E，其中E是偏置后的阶码值，M是尾数值，S为符号位。

给出以下十进制值，请写出其对应的浮点数二进制表示，并请给出其舍入值（采用round-to-even舍入规则）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 描述 | 浮点数位模式 | M | E | 实际值 |
| -0 | 1 000 0000（1） | 0（2） | -2（3） | -0.0 |
| — | 0 100 0101 | 21/16（4） | 1（5） | 21/8（6） |
| 最小的非规格化数（负） | 1 000 1111（7） | 15/16（8） | -2（9） | -15/64（10） |
| 最大的规格化数（正） | 0 110 1111（11） | 31/16（12） | 3（13） | 31/2（14） |
| 1 | 0 011 0000（15） | 1（16） | 0（17） | 1.0 |
| — | 0 101 0110（18） | 11/8（19） | 2（20） | 5.5 |

四、（12分）a，b为整型变量，将左侧的每个描述与右侧的一行代码匹配。

1. ˜ a f ;

2. a b ;

3. a & b a ;

4. a \* 7 c ;

5. a / 4 e ;

6. (a < 0) ? 1 : -1 h .

a. ˜(˜a | (b ˆ (MIN\_INT + MAX\_INT)))

b. ((a ˆ b) & ˜b) | (˜(a ˆ b) & b)

c. 1 + (a << 3) + ˜a

d. (a << 4) + (a << 2) + (a << 1)

e. ((a < 0) ? (a + 3) : a) >> 2

f. a ˆ (MIN\_INT + MAX\_INT)

g. ˜((a | (˜a + 1)) >> 31) & 1

h. ˜((a >> 31) << 1)

i. a >> 2

五、(12分)考虑如下C函数,请根据对应的汇编代码，填写C代码中缺失的表达式。

int looper(int n, int \*a) {

int i;

int x = \_\_\_\_\_\_（1）\_0\_\_\_\_\_\_\_;

for(i = \_\_\_\_\_\_\_（2）0\_\_\_\_\_;

\_\_\_\_\_\_\_（3）i < n \_\_\_\_\_\_\_\_\_;

i++) {

if (\_\_\_\_\_\_\_\_（4）\_\_ a[i] > x \_\_\_\_\_\_\_\_\_)

x = \_\_\_\_\_\_（5）a[i]\_\_\_\_\_\_\_\_\_\_\_;

\_\_\_\_\_\_\_\_（6）x++\_\_\_\_\_\_\_\_\_\_\_\_;

}

return x;

}

其对应的汇编代码如下：

int looper(int n, int \*a)

n in %edi, a in %rsi

looper:

xorl %eax,%eax

xorl %ecx,%ecx

jmp .L25

.L27:

movl (%rsi,%rcx,4),%edx

cmpl %eax,%edx

jle .L28

movl %edx,%eax

.L28:

incl %eax

incl %ecx

.L25:

cmpl %edi,%ecx

jl .L27

ret

六、 (8分)考虑如下C程序。

#include <stdio.h>

/\* Read a string from stdin into buf \*/

long evil\_read\_string() {

long buf[2];

scanf("%s",buf);

return buf[0];

}

int main() {

printf("0x%x\n", evil\_read\_string()); }

* scanf("%s", buf)：从标准输入流（stdin）中读入字符串，并存储在buf开始的存储器中（字符串以’\0’结束，不检查buf存储器的长度）。
* printf("0x%x", i)：以“0x”开始打印输出整型变量i的十六进制值。
* Linux/x86-64机器采用小端法。
* 部分字符的ASCII码如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字符 | ASCII | 字符 | ASCII | 字符 | ASCII |
| ‘d’ | 0x64 | ‘i’ | 0x69 | ‘e’ | 0x65 |
| ‘v’ | 0x76 | ‘.’ | 0x2e | ‘s’ | 0x73 |
| ‘r’ | 0x72 | ‘l’ | 0x6c | ‘\0’ | 0x00 |

A．上述程序运行在Linux/x86-64机器, 输入字符串dr.evils.lives，针对如下所示的栈结构，请填写buf[0]和buf[1]中存储的数据（以十六进制表示）

Stack Frame

For evil\_read\_string

**65**

**2e**

**72**

**64**

**Buf[0]**

20 bytes unused

**73**

**6c**

**69**

**76**

**76**

**69**

**6c39**

**2e**

**35**

**00**

**73**

**65**

**39**

**38**

**37**

**36**

**33**

**32**

**31**

**30**

**Buf[1]**

B．程序的输出结果是什么？ 0x\_\_\_\_\_\_\_\_\_\_\_

0x736c6976652e7264

七、 (12分)考虑x86-64 机型下定义的如下结构：

struct my\_struct {

int \*p;

struct {

int x;

short y;

}s;

struct my\_struct \*next;

};

1. （4分）试确定下列字段的偏移量是多少？（以字节为单位）

p: (1) 0 ;

s.x: (2) 8 ;

s.y: (3) 12 ;

next: (4) 16 ;

1. （2分）这个结构总共需要的字节数为 （5） 24 。
2. （6分）下面的过程是对这个结构进行操作的C代码及其编译后的汇编代码，试填写过程中缺失部分：

void my\_struct\_init(struct my\_struct \*sp){

sp->s.x= (6) sp->s.y ;

sp->p= (7) &(sp->s.x) ;

sp->next= (8) sp ;

}

其汇编代码如下：

void my\_struct\_init(struct my\_struct \*sp)

sp in %rdi

my\_struct\_init:

movl 12(%rdi),%eax

movl %eax, 8(%rdi)

leaq 8(%rdi),%rax

movq %rax,(%rdi)

movq %rdi, 16(%rdi)

ret

八、(10 points):某程序中包含switch语句，采用跳转表实现。其跳转表如下：

0x4004b7: jmpq \*0x400600(,%rax,8)

Using GDB, we extract the 8-entry jump table:

0x400600: 0x00000000004004d1 0x00000000004004c80x400610: 0x00000000004004c8 0x00000000004004be0x400620: 0x00000000004004c1 0x00000000004004d70x400630: 0x00000000004004c8 0x00000000004004be

包含switch语句的C语言程序的反汇编代码如下：

# on entry: %rdi = x, %rsi = y, %rdx = z

0x4004b0: cmp $0x7,%edx0x4004b3: ja 0x4004c80x4004b5: mov %edx,%eax

0x4004b7: jmpq \*0x400600(,%rax,8)0x4004be: mov %edi,%eax

0x4004c0: retq

0x4004c1: mov $0x3,%eax0x4004c6: jmp 0x4004da0x4004c8: mov %esi,%eax0x4004ca: nopw 0x0(%rax,%rax,1) // 空指令

0x4004d0: retq

0x4004d1: mov %edi,%eax0x4004d3: and $0x19,%eax0x4004d6: retq

0x4004d7: lea (%rdi,%rdi,1),%eax0x4004da: add %esi,%eax0x4004dc: retq

请分析反汇编程序以及跳转表，填写下列C程序中的缺失部分：

int test(int x, int y, int z){

int result = 3; switch(z){

case \_（1）0\_\_:

\_\_\_\_\_（2）\_ result= x & 25\_\_\_\_;

break;

case \_（3）\_ 3\_:

case \_（4）\_7\_:

result = \_\_ x \_\_\_（5）\_\_\_\_\_\_\_\_;

break;

case \_（6）\_5\_:

result = \_ 2 \* x \_\_\_（7）\_\_\_\_\_\_\_\_\_;

case \_（8）4\_\_:

result = result + y \_\_\_\_\_（9）\_\_\_\_\_\_\_\_;

break;

default:

result = \_\_\_\_\_\_（10）\_ y \_\_\_\_\_\_; }

return result; }