**武汉大学计算机学院 2021-2022 学年第一学期《 系统级程序设计 》期末考试试卷 A 答案**

1. 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 |
|---|---|---|---|
| 0x200 | 0xBE | %rax | 0x204 |
| 0x202 | 0x37 | %rdi | 0x3 |
| 0x204 | 0x5C | %rsi | 0x1 |
| 0x206 | 0x13 | | |
| 0x208 | 0x24 | | |
| 0x20A | 0xAA | | |
| 0x20C | 0x6D | | |

填写下表，给出所示操作数的值（10 分）

| 操作数 | 值 |
|---|---|
| %rax | |
| $0x200 | |
| 4(%rax) | |
| 262(%rcx,%rdx) | |
| 0x2(%rax, %rcx, 3) | |

**答案：**

| 操作数 | 值 | 注释 |
|---|---|---|
| %rax | 0x204 | 寄存器 |
| $0x20A | 0x20A | 立即数 |
| 4(%rax) | 0x24 | 地址 0x208 |
| 514(%rsi,%rdi) | 0x13 | 地址 0x206 |
| 0x5(%rax, %rsi, 3) | 0x6D | 地址 0x20C |

2. 假设在一个 int 类型为 32 位长度的机器上运行程序。Float 类型的值使用 32 位 IEEE 格式，而 double 类型的值使用 64 位 IEEE 格式。如下左图所示，产生随机数 x, y 和 z，并把它们转换为 float/double 类型的值。对于下右图中每个 C 表达式，请指出表达式是否总是为 1。如果它总是为 1，请描述其中的数学原理。否则，列举出一个使它为 0 的情况示例。（10 分）

```
/* Create some arbitrary values */
int x = random();
int y = random();
int z = random();
/* Convert to unsigned */
unsigned ux = (unsigned) x;
unsigned uy = (unsigned) y;
/* Convert to float/double */
float fx = (float) x;
double dx = (double) x;
double dy = (double) y;
double dz = (double) z;
```

```
A. ux - x == 0

B. (x-y) > 0 => -x < -y

C. (double)(float) x == (double) x

D. dx + dy == (double) (y+x)

E. dx + dy + dz == dz + dy + dx
```

答案：

A. 正确。表达式中有无符号数 ux，x 也会自动转换为 ux，所以结果恒为 0.

B. 不正确。反例：x = -1， y = TMin

C. 不正确。Int 转换成 float 时可能有精度损失，但 int 转换为 double 时不会。反例：x=Tmax

D. 不正确。y+x 可能发生溢出，但 dy+dx 不会溢出。反例：x=y=Tmax

E. 不正确。浮点数不满足结合律，例如：(3.14+1e10)-1e10 = 0, 3.14+(1e10-1e10) = 3.14

3. 我们在一个 int 类型值为 32 位的机器上（64 位）运行程序，这些值以补码形式表示。右移操作都是算术右移。MAX_INT 表示最大的整数，MIN_INT 表示最小的整数，W= 31。假设 x 是有符号整数，且 $x < 0$。请对以下表达式的求值。如果结果是常数，请给出其 16 进制表示。否则，请以包含 x 的表达式予以表示。请给出求解过程的简要说明。（10 分）

   示例：

   **x << 2　　答案：x * 4**

   **A. ( x << 4 ) + ( x << 2 ) + ( x << 1 )**

   **B. 1 + ( x << 3 ) + ~ x**

   **C. x ^ ( MIN_INT + MAX_INT)**

   **D. ~ ( ( x >> W) << 1 )**

   E. **~ ( ( x | ( ~ x + 1) ) >> W ) & 1**

答案:

A. 22*x

B. 7*x

C. ~ x （或者 x ^ 0xFFFFFFFF）

D. x<0，结果恒为 1，即 0x000000001;

E. x 不等于 0， 结果恒为 0，即 0x00000000.

4. 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 | 1. 有一个符号位 |
| 2. 有 k=3 个阶码位。阶码偏置值是 3. | 2. 有 k=5 个阶码位。阶码偏置值是 15. |
| 3. 有 n=5 个小数位 | 3. 有 n=3 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，你要向偶数舍入。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。（10 分）

| 格式 A | | 格式 B | |
|---|---|---|---|
| 位 | 值 | 位 | 值 |
| 0 011 0000 | 1 | 0 01111 000 | 1 |
| | | | -15 |
| | $\frac{53}{16}$ | | |
| | | 0 10100 110 | (不用填) |

答案：

| 格式 A | | 格式 B | |
|---|---|---|---|
| 位 | 值 | 位 | 值 |
| 0 011 0000 | 1 | 0 01111 000 | 1 |
| 1 110 11100<br>（1 分） | -15<br>（1 分） | 1 10010 111<br>（1 分） | -15 |
| 0 100 10101<br>（1 分） | $\frac{53}{16}$ | 0 10000 101 （2 分） | 13/4<br>（2 分） |
| 0 111 00000<br>（1 分） | +INF<br>（1 分） | 0 10100 110 | (不用填) |

5、请分析下面的汇编代码及对应的 C 语言函数框架回答问题（10 分）

```
arith:
.LFB0:
        leaq    (%rdi,%rdi), %rax
        leaq    4(%rsi), %rdi
        movl    $0, %ecx
        jmp  .L2
.L3:
        addq    %rdx,  %rax
        addq    %rdx,  %rdi
        addl    $1,  %ecx
.L2:
        cmpl    $4, %ecx
        jle  .L3
        imulq   %rdi, %rax
        ret
```

对应的 C 语言函数框架

```
long arith(long x, long y, long z)
{
```

```
    long t1 = 2 * x;
    long t2 = y + 4;
    for( int i=0;  _______;  _______;){
       _____________;
       _____________;
    }
    long rval = t1 * t2;
    return rval;
}
```

（1） 解释汇编代码中下划线语句的具体含义（尽量使用 C 语言的变量来解释）；

（2） 在 C 语言函数框架中填写下划线部分的 C 程序语句。

答案：

（1）

.L3:

```
addq    %rdx,  %rax      计算 t1=t1+z  （%rax 中保存变量 t1 的值）
addq    %rdx,  %rdi      计算  t2=t2+z     （%rdi 中保存变量 t2 的值）
addl    $1,  %ecx        i=i+1   （ecx 中保存变量 i 的值）
```

.L2:

```
cmpl    $4, %ecx         比较是否 i<=4  （ecx 中保存变量 i 的值）
jle  .L3                 如果 i<=4，跳转到.L3，进入 for 循环
imulq   %rdi, %rax       计算%rax=%rax*%rdi，等价于将 t1*t2 的值放入%rax 中
```

（2）

```
i <= 4
i++
t1 = t1 + z
t2 = t2 + z
```

6、请分析下面的汇编代码及对应的 C 语言函数框架回答问题（10 分）

```
rfact:
.LFB23:
    cmpq    $2, %rdi
    jg   .L8
    movl    $1, %eax
    ret
.L8:
    pushq   %rbp
    pushq   %rbx
    subq$8, %rsp
    movq    %rdi, %rbx
    leaq    -1(%rdi), %rdi
```

```
call    rfact
movq    %rax, %rbp
leaq    -2(%rbx), %rdi
call    rfact
leaq    0(%rbp,%rax,2), %rax
addq    %rbx, %rax
addq    $8, %rsp
popq    %rbx
popq    %rbp
ret
```

对应的 C 语言函数框架

```
long rfact(long n)
{
long result;
    if(n <= 2){
        result = 1;
    }else{
        result =rfact(n-1)+_____+_____;
    }
return result;
}
```

(1) 解释汇编代码中下划线语句的具体含义（尽量使用 C 语言的变量来解释）；

(2) 在 C 语言函数框架中填写下划线部分的 C 程序语句。

答案：

（1）

```
movq    %rdi, %rbx          将%rdi(变量 n 的值)保存到%rbx
leaq    -1(%rdi), %rdi      %rdi=n-1
call    rfact               计算 rfact(n-1)
movq    %rax, %rbp          将计算结果%rax 中 rfact(n-1)的值保存到%rbp
leaq    -2(%rbx), %rdi      %rdi=n-2
call    rfact               计算 rfact(n-2)
leaq    0(%rbp,%rax,2), %rax    将计算结果%rax*2+%rbp，%rax 中保存 2*rfact(n-2)+rfact(n-1)
addq    %rbx, %rax          %rax=%rax+%rbp，因此%rax=2*rfact(n-2)+rfact(n-1)+n
```

（2）2 * rfact(n - 2)

n

7、请分析下面的汇编代码及对应的 C 语言函数框架，填写 C 代码中缺失的部分（10 分）

switch_eg:

```
.LFB0:
        subq$56, %rsi
        cmpq    $5, %rsi
        ja  .L9
        jmp *.L4( ,%rsi, 8)
        .section    .rodata
.L4:
        .quad   .L3
        .quad   .L9
        .quad   .L5
        .quad   .L6
        .quad   .L7
        .quad   .L7
        .text
.L3:
        leaq    (%rdi,%rdi,2), %rax
        leaq    (%rdi,%rax,4), %rdi
        jmp  .L2
.L5:
        cmpq    $10, %rdi
        jle .L8
        subq$10, %rdi
.L6:
        addq$11, %rdi
.L2:
        movq    %rdi, (%rdx)
        ret
.L8:
        addq$10, %rdi
        jmp  .L6
.L7:
        imulq   %rdi, %rdi
        jmp  .L2
.L9:
        movl    $0, %edi
        jmp  .L2
```

对应的 C 语言函数框架：

```
void switch_eg (long x, long n,long *dest)
{
        long val = x;
        switch (n)
        {
                case 56:
```

```
            val *= 13;
            break;
        case __________:
            if(val >__________){
                val -= 10;
            }
            else{
                __________;
            }
        case 59:
            __________;
            break;
        case 60:
        case 61:
            val *= val;
            break;
        default:
            __________;
    }
    *dest = val;
}
```

答案：

58

10

val += 10

val += 11

val = 0

8、某 C 代码在 X86-64，使用命令 gcc -O0 -fno-stack-protector 编译后，得到的汇编代码如下，请回答下列问题。（14 分）

| C 语言代码 | foo 函数对应的汇编代码 |
| --- | --- |

```
void foo()
{
        char buf[4]="abc";
        int a = 0x31323334;
        gets(&a);
        printf("a=%s", &a);
}

int main()
{
        foo();
        return 0;
}
```

提示：

1. ‘a'对应的 ascii 码为 0x61。
2. ‘0’对应的 ascii 码为 0x30。
3. X86-64 为小端模式。
4. 通用寄存器长度为 64 位。
5. gets 为标准库函数。

```
0000000000400537 <foo>:
  400537:       55                      push   %rbp
  400538:       48 89 e5                mov    %rsp,%rbp
  40053b:       48 83 ec 10             sub    $0x10,%rsp
  40053f:       c7 45 fc 61 62 63 00    movl   $0x636261,-0x4(%rbp)
  400546:       c7 45 f8 34 33 32 31    movl   $0x31323334,-0x8(%rbp)
  40054d:       48 8d 45 f8             lea    -0x8(%rbp),%rax
  400551:       48 89 c7                mov    %rax,%rdi
  400554:       b8 00 00 00 00          mov    $0x0,%eax
  400559:       e8 e2 fe ff ff          callq  400440 <gets@plt>
  40055e:       48 8d 45 f8             lea    -0x8(%rbp),%rax
  400562:       48 89 c6                mov    %rax,%rsi
  400565:       48 8d 3d a8 00 00 00    lea    0xa8(%rip),%rdi
  40056c:       b8 00 00 00 00          mov    $0x0,%eax
  400571:       e8 ba fe ff ff          callq  400430 <printf@plt>
  400576:       90                      nop
  400577:       c9                      leaveq
  400578:       c3                      retq

0000000000400579 <main>:
  400579:       55                      push   %rbp
  40057a:       48 89 e5                mov    %rsp,%rbp
  40057d:       b8 00 00 00 00          mov    $0x0,%eax
  400582:       e8 b0 ff ff ff          callq  400537 <foo>
  400587:       b8 00 00 00 00          mov    $0x0,%eax
  40058c:       5d                      pop    %rbp
  40058d:       c3                      retq
  40058e:       66 90                   xchg   %ax,%ax
```

（1）假设用户输入为“0123456789012345”，下列数据位置在 gets 函数执行前后的值是多少？（如果信息不够，请填写“不确定”，并简洁分析）（6 分）

| **数据** | **gets 执行前**<br>**（16 进制或字符表示都行）** | **gets 执行后**<br>**（16 进制或字符表示都行）** |
|---|---|---|
| A | 0x31323334 或“4321” | |
| Buf | “abc” 或 0x00636261 | |
| foo 函数中，寄存器 rbp 在栈上的保存备份 | | |
| foo 函数执行的返回地址 | | |

（2）该程序在上述输入时，可能发生“段错误”，试分析原因？（2 分）

（3）该程序在上述输入时，也有可能正常结束，试分析原因？这样有何风险？如何解决？（6 分）

答案：

（1）

| **数据** | **gets 执行前**<br>**（16 进制或字符表示都行）** | **gets 执行后**<br>**（16 进制或字符表示都行）** |
|---|---|---|
| a | 0x31323334 或“4321” | “0123” |
| buf | “abc” 或 0x00636261 | “4567” |
| 寄存器 rbp 的在栈上的保存备份 | 不确定 | “89012345” |
| foo 函数执行的返回地址 | 0x400587 | 0x400036 |

（2）

有可能。当返回地址 0x400036 不是进程中合法的代码地址，就会发生段错误。

（3）

有可能。当返回地址 0x400036 是进程中的合法代码地址（但是意料之外的），就会跳转到该地址开始执行，并有可能正常结束。

这样有巨大的风险。一个程序计算过程是错误的，但是正常结束执行；用户却以为是正确执行的。可以利用金丝雀值进行栈保护，比如关闭 -fno-stack-protector 选项。

9、你参加了一个开发团队，试图开发世界上性能最快的计算阶乘的程序。（10 分）

（1）第 1 版使用的是递归调用实现；第 2 版采用了循环实现，如下表（a），发现性能得到了显著提升。试简要分析原因。（3 分）

（2）有一位程序员为了进行优化，尝试了第 3 版，如下表（b），发现性能没有得到提升。试简要分析原因。（3 分）

（3）根据教材第 5 章所学，请提供 2 个正确、且比第 3 版性能更优的版本。（4 分）

| （a） | （b) |
|---|---|
| int fact(int n)<br>{<br>int i;<br>int result = 1;<br><br>for (i = n; i > 1; i = i --)<br>result = result * i ;<br>} | int fact(int n)<br>{<br>int i;<br>int result = 1;<br><br>for (i = n; i > 1; i = i -2)<br>result = result * i * (i-1);<br>} |

**答案：**

（1）因为递归调用涉及大量的函数调用，而相比循环，函数调用有很多额外的性能 overhead，包括控制转移、参数和返回值传递、内存分配等。

（2）虽然做了循环展开，但是两个乘法之间仍然构成串行依赖，所以不能并行改进。

(3.a) Result = result *( i * (i-1));

(3.b) res1 = result *i; res2 = res2 * (i-1);

**10、**下面是不完整的 C 语言代码，以及该代码在 **IA-32** 的 Linux 上编译后的汇编代码。在该机器上，数据类型的大小和对齐要求如下表所示。结合这些信息以及 IA-32 上指针长度为 32 位，请将下列 C 语言补充完整。（6 分）

```
typedef struct node {
  _____ x;
  _____ y;
  struct node *next;
  struct node *prev;
} node_t;
```

```
node_t n;
void func() {
  node_t *m;
  m = _____;
  m->y /= 16;
  return;
}
```

| 对应的汇编代码 | 类型及对齐要求 |
|---|---|

```
func:
  pushl %ebp
  movl n+12,%eax
  movl 16(%eax),%eax
  movl %esp,%ebp
  movl %ebp,%esp
  shrw $4,8(%eax)
  popl %ebp
  ret
```

| Type | Size (bytes) | Alignment (bytes) |
|---|---|---|
| char | 1 | 1 |
| short | 2 | 2 |
| unsigned short | 2 | 2 |
| int | 4 | 4 |
| unsigned int | 4 | 4 |
| double | 8 | 4 |

**答案：**

1）x 占 8 个字节，double 或者 int[2]类型；

2）y 占 2 个字节，short 类型；

3）m = n.next->prev