一. 选择题

1. 某 C 语言程序中对数组变量 a 的声明为 `long a[10][10];` , 有如下一段代码:

```
for (i=0; i<10; i++)
    for (j=0; j<10; j++)
        sum += a[i][j];
```

假设执行到`sum+=a[i][j];`时, sum 的值在`%rax` 中, `a[i][0]` 所在的地址在`%rdx` 中, j 在`%rsi` 中, 则`sum+=a[i][j];` 所对应的指令是 _____

A. `addl 0(%rdx, %rsi, 8), %eax`

B. `addl 0(%rsi, %rdx, 8), %eax`

C. `addl 0(%rdx, %rsi, 4), %eax`

D. `addl 0(%rsi, %rdx, 4), %eax`

2. 下列关于数据传送指令的说法中, 正确的是 _____

A. 常规的`movq` 指令只能以表示为 32 位补码数字的立即数作为源操作数 (注: 这里的"只能" 强调的是不能以 64 位数作为源操作数), 然后把这个值零扩展到 64 位的值, 放到目的位置.`movabsq` 指令能够以任意 64 位立即数值作为源操作数, 并且只能以寄存器为目的.

B. 在`Imm(rb,ri,s)` 这一寻址模式中, `s` 必须是`1,2,4` 或者`8`, 基址和变址寄存器必须是 64 位寄存器.

C. `cqto` 指令不需要额外操作数, 它的作用是把`%eax` 符号扩展到`%rax`.

D. CPU 执行指令`movl -1,%eax` 后, `%rax` 的值为`0x00000000ffffffff`.

3. 下列关于通用目的寄存器和条件码的说法中, 正确的是 _____

A. `%rbx,%rbp,%r12-r15` 是被调用者保存寄存器, 其他寄存器是调用者保存寄存器.

B. `xorq %rax,%rax` 可以用于清空`%rax`; `cmpq %rax,%rax` 可以用于判断`%rax` 的值是否为零.

C. 调用一个过程时, 最多可以通过寄存器传递 6 个整型参数, 依次存储在 (假设参数都是 64 位的)`%rdi,%rsi,%rdx,%rcx,%r8,%r9` 中. 当需要更多的参数时, 调用者过程可以在自己的栈帧里按照地址由低到高的顺序依次存储这些参数.

D. `leaq` 指令不改变条件码; 逻辑操作 (如 XOR) 会将进位标志和溢出标志都设置为零; 移位操作会把溢出标志设置为最后一个被移出 (shift out) 的位, 而把进位标志设置为零.

4. 假设某条 C 语言 `switch` 语句编译后产生了如下的汇编代码及跳转表:

```
movl 8(%ebp), %eax
subl $48, %eax
cmpl $8, %eax
ja   .L2
```

```
5  jmp  *.L7(, %eax, 4)

   .L7:
   .long .L3
   .long .L2
10 .long .L2
   .long .L5
   .long .L4
   .long .L5
   .long .L6
15 .long .L2
   .long .L3
```

在源程序中, 下面的哪些 (个) 标号出现过_____

A. '2', '7' B. 1 C. '3' D. 5

5. x86 体系结构中, 下面哪个说法是正确的?

   A. leal 指令只能够用来计算内存地址

   B. x86-64 机器可以使用栈来给函数传递参数

   C. 在一个函数内, 改变任一寄存器的值之前必须先将其原始数据保存在栈内

   D. 判断两个寄存器中值大小关系, 只需要 SF 和 ZF 两个条件

6. 下列的指令组中, 哪一组指令只改变条件码, 而不改变寄存器的值?

   A. CMP, SUB B. TEST, AND C. CMP, TEST D. LEAL, CMP

7. 在下列关于条件传送的说法中, 正确的是

   A. 条件传送可以用来传送字节、字、双字、和 4 字的数据

   B. C 语言中的"?:" 条件表达式都可以编译成条件传送

   C. 使用条件传送总可以提高代码的执行效率

   D. 条件传送指令不需要用后缀 (例如 b, w, l, q) 来表明操作数的长度

8. 以下关于 x86-64 指令的描述, 说法正确的有_____

   (a) 有符号除法指令 idivq S 将%rdx(高 64 位) 和%rax(低 64 位) 中的 128 位数作为被除数, 将操作数 S 的值作为除数, 做有符号除法运算; 指令将商存在%rdx 寄存器中, 将余数存在%rax 寄存器中.

   (b) 我们可以使用指令 jmp %rax 进行间接跳转, 跳转的目标地址由寄存器%rax 的值给出.

   (c) 算术右移指令 shr 的移位量既可以是一个立即数, 也可以存放在单字节寄存器%cl 中.

   (d) leaq 指令不会改变任何条件码.

9. x86-64 指令提供了一组条件码寄存器; 其中 ZF 为零标志,ZF=1 表示最近的操作得出的结构为 0;SF 为符号标志,SF=1 表示最近的操作得出的结果为负数;OF 为溢出标志,OF=1 表示最近的操作导致一个补码溢出 (正溢出或负溢出). 当我们在一条 cmpq 指令后使用条件跳转指令 jg 时, 那么发生跳转等价于以下哪一个表达式的结果为 1?

A. `~(SF ^ OF) & ~ZF`

B. `~(SF ^ OF)`

C. `SF ^ OF`

D. `(SF ^ OF) | ZF`

二. 填空将下列汇编代码翻译成 C 代码

```
func:
    movl $1, %eax
    jmp .L2
.L4:
    testb $1, %sil
    je .L3
    imulq %rdi, %rax
.L3:
    sarq %rsi
    imulq %rdi, %rdi
.L2:
    testq %rsi, %rsi
    jg .L4
    rep ret
```

```
// a in %rdi, b in %rsi
long func(long a, long b) {
    long ans = ___________;
    while (___________) {
        if (___________)
            ans = __________;
        b = ________;
        a = ________;
    }
    return ans;
}
```

三. 以下提供了一段代码的 C 语言、汇编语言以及运行到某一时刻栈的情况

I. 互相翻译 C 语言代码和汇编代码，补充缺失的空格（标号相同的为同一格）。

```
0000000000400596 <func>:
400596: sub $0x28,%rsp
40059a: mov (4)_____,%rax
4005a3: mov %rax,0x18(%rsp)
4005a8: xor %eax,%eax
4005aa: mov (%rdi),%rax
4005ad: mov 0x8(%rdi),%rdx
4005b1: cmp %rdx,%rax
4005b4: jge (1)_____
4005b6: mov %rdx,(%rdi)
4005b9: mov %rax,0x8(%rdi)
4005bd: mov 0x8(%rdi),%rax
4005c1: test %rax,%rax
4005c4: jne 4005cb <func+0x35>
4005c6: mov (%rdi),%rax
4005c9: jmp (2)_____
4005cb: mov (%rdi),%rdx
4005ce: sub %rax,%rdx
4005d1: mov %rdx,(%rsp)
4005d5: mov %rax,0x8(%rsp)
4005da: mov (3)_____ ,%rdi
4005dd: callq 400596 <func>
4005e2: mov 0x18(%rsp),%rcx
4005e7: xor %fs:0x28,%rcx
4005f0: (5)_____ 4005f7 <func+0x61↩
    >
4005f2: callq 400460 <↩
    __stack_chk_fail@plt>
```

```
4005f7: add (6)____,%rsp
4005fb: retq
00000000004005fc <main>:
4005fc: sub $0x28,%rsp
400600: mov %fs:0x28,%rax
400609: mov %rax,0x18(%rsp)
40060e: xor %eax,%eax
400610: movq 0x69,(%rsp)
400618: movq 0xfc,0x8(%rsp)
400621: mov %rsp,%rdi
400624: callq 400596 <func>
400629: mov %rax,%rsi
40062c: mov $0x4006e4,%edi
400631: mov $0x0,%eax
400636: callq 400470 <printf@plt>
40063b: mov 0x18(%rsp),%rdx
400640: xor (4)____,%rdx
400649: (5)____ 400650 <main+0x54↩
   >
40064b: callq 400460 <↩
   __stack_chk_fail@plt>
400650: mov $0x0,%eax
400655: add (6)____ ,%rsp
400659: retq
```

```
typedef struct{ long a; long b; } pair_type;
long func(pair_type *p) {
    if (p->a < p->b) {
        long temp = p->a;
        p->a = p->b;
        p->b = temp;
    }
    if ((7)____ ) return p->a;
    pair_type np;
    np.a = (8)____;
    np.b = (9)____;
    return func(&np);
}
int main(int argc, char* argv[]) {
    pair_type np;
    np.a = (10)____;
    np.b = (11)____ ;
    printf("%ld", func(&np));
    return 0;
}
```

(1) _____ (5) _____ (9) _____

(2) _____ (6) _____ (10) _____

(3) _____ (7) _____ (11) _____

(4) _____ (8) _____

附: 一些可能用到的字符的 ASCII 码表

\n	space	"	%	(	)	,	0	A	a
0x0a	0x20	0x22	0x25	0x28	0x29	0x2c	0x30	0x41	0x61

II. 补充栈的内容。使用 16 进制，可以不写前导多余的 0；对于给定已知条件后仍无法确定的值，填写“不确定”；已知程序运行过程中寄存器`%fs` 的值没有改变

```
......
0x0000000000000000
0xc76d5add7bbeaa00
0x00007fffffffdf60
(a)_________________
(b)_________________
0x0000000000400629
(c)_________________
(d)_________________
0x0000000000000001
0x0000000000000069
0x0000000000000093
(e)_________________
0x00000000ff000000
(f)_________________
0x0000000000000000
(g)_________________
(h)_________________
(i)_________________
0x0000000000000000
(j)_________________
(k)_________________
0x000000000000002a
0x000000000000003f
0x00000000004005e2
// stack top (low address)
```

III. 程序运行结果为_____.