

北京大学信息科学技术学院考试试卷

考试科目：计算机系统导论 姓名：_____ 学号：_____

考试时间：2025 年 10 月 13 日 大班号：_____ 小班号：_____

题号	2	3	4	5	6	7	8	总分
分数								
阅卷人								

北京大学考场纪律

1、考生进入考场后，按照监考老师安排隔位就座，将学生证放在桌面上。无学生证者不能参加考试；迟到超过 15 分钟不得入场。在考试开始 30 分钟后方可交卷出场。

2、除必要的文具和主考教师允许的工具书、参考书、计算器以外，其它所有物品（包括空白纸张、手机、或有存储、编程、查询功能的电子用品等）不得带入座位，已经带入考场的必须放在监考人员指定的位置。

3、考试使用的试题、答卷、草稿纸由监考人员统一发放，考试结束时收回，一律不准带出考场。若有试题印制问题请向监考教师提出，不得向其他考生询问。提前答完试卷，应举手示意请监考人员收卷后方可离开；交卷后不得在考场内逗留或在附近高声交谈。未交卷擅自离开考场，不得重新进入考场答卷。考试结束时间到，考生立即停止答卷，在座位上等待监考人员收卷清点后，方可离场。

4、考生要严格遵守考场规则，在规定时间内独立完成答卷。不准交头接耳，不准偷看、夹带、抄袭或者有意让他人抄袭答题内容，不准接传答案或者试卷等。凡有违纪作弊者，一经发现，当场取消其考试资格，并根据《北京大学本科考试工作与学术规范条例》及相关规定严肃处理。

5、考生须确认自己填写的个人信息真实、准确，并承担信息填写错误带来的一切责任与后果。

学校倡议所有考生以北京大学学生的荣誉与诚信答卷，共同维护北京大学的学术声誉。

注意事项:

(1) 本试卷基本按照大班课程讲授次序分组, 不是按照题目难度排序。题量较大, 注意合理安排时间。

(2) 除特别说明外, 默认技术前提是本课程课件和对应教材内容, 例如: 基于 x86-64 体系结构、Linux 操作系统、C 语言程序等; 汇编程序是对应 C 程序主体功能实现, 不做高级别优化, 不一定是完整代码。

(3) 除特别说明外, 汇编程序代码按每行计分, 其它按每空计分。

(4) 在试卷上指定位置答题, 包括冒号后、等号后、箭头后、横线上、括号里等。

第 2 讲 (每空 2 分, 12 分) 得分: _____

1. 两个 8 位二进制数 10110101 和 01011100, 分别记为 a 和 b。问:

(1) a 转换为十进制表示为:

(2) b 转换为十六进制表示为:

(3) a 和 b 按位与操作, $a \& b$ (用十六进制表示):

(4) 用 C 语言中的“非”操作, $!!b$ (用十六进制表示):

2. 将 C 语言中的两个常量比较大小: -10 和 10U。关系是 (大于、小于、等于):

3. 分析下面的 C 语言代码

```
short int y = -12;
```

```
int iy = (int) y;
```

iy 首字节的值为 (用十六进制表示):

第 3 讲 (每空 1 分, 10 分) 得分: _____

4. 在 IEEE 754 标准中, 单精度浮点数 (C 语言中的 float, 32 位) 的编码结构要点为: 符号位 (1-bit)、阶码 (8-bit)、尾数 (23-bit)。据此, 问:

(1) 正无穷的编码为 (用十六进制表示):

(2) 将编码 0xA1100000 对应到表达式 $v = (-1)^s \times M \times 2^E$, 则:

$s =$ _____, $M =$ _____, $E =$ _____ (均用十进制表示)

5. 按照“向偶数舍入”的规则, 将下面的二进制数保留小数点后三位:

101.111100 →

101.100100 →

6. 在一个 int 类型为 32 位补码表示的机器上运行程序。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, 该机器上没有 80 位扩展精度。以下 C 代码产生随机整数 x、y、z, 并把它们转换成 double 类型的值:

```

int x = random();
int y = random();
int z = random();
double dx = (double) x;
double dy = (double) y;
double dz = (double) z;

```

对于下列的每个 C 表达式, 指出表达式是否总是为 True。若为“否”, 给出简要说明。

- A. $(\text{float})\ x == (\text{float})\ dx \rightarrow$
- B. $(dx + dy) + dz == dx + (dy + dz) \rightarrow$
- C. $(dx * dy) * dz == dx * (dy * dz) \rightarrow$
- D. $dx/dx == dz/dz \rightarrow$

第 4 讲 (每空 1 分, 6 分) 得分: _____

7. x86-64 的通用寄存器中, 对应 rbx 低 32 位的寄存器名称为: _____; 对应 rcx 最低字节的寄存器名称为: _____。

8. 以下代码中, 有 _____ 次通用寄存器访问, 有 _____ 次内存访问 (均不含取指令相关操作)。

swap:

```

movq (%rdi), %rax
movq (%rsi), %rdx
movq %rdx, (%rdi)
movq %rax, (%rsi)

```

9. 若寄存器内容为 $rdx:0xc000, rcx:0x0200$, 则计算下列地址的值 (十六进制表示):

$0x10(\%rdx, \%rcx, 4):$

$0x30(, \%rdx, 8):$

第 5 讲 (每空 2 分, 28 分) 得分: _____

10. 对于以下汇编指令, 给出与之内在操作流程相同、只是会修改目的寄存器的指令:

$\text{cmpq } \%rbx, \%rax \rightarrow$

$\text{testq } \%rbx, \%rax \rightarrow$

11. 下面的 C 语言代码与汇编语言代码对应

<pre> int cmpxy(long x, long y) { return x > y; } </pre>	<pre> cmpq %rsi, %rdi setg %al movzbl %al, %eax </pre>
---	---

}	ret
---	-----

(1) 修改 C 语言代码如下，相应补全汇编语言代码

int cmpxy(long x, long y)	cmpq %rsi, %rdi
{	_____
return x <= y;	movzbl %al, %eax
}	ret

(2) movzbl 指令本身的功能是:

(3) 在 x86-64 中，这条 movzbl 隐含操作还有:

12. 下面的 C 语言代码与汇编语言代码对应，补全汇编代码

long absdiff(long x, long y)	absdiff:
{	cmpq %rsi, %rdi
long result;	_____
if (x <= y)	movq %rdi, %rax
result = x-y;	subq %rsi, %rax
else	_____
result = y-x;	.L4:
return result;	movq %rsi, %rax
}	subq %rdi, %rax
	ret

(2) 如果用条件传送指令实现上述功能，对应汇编语言代码如下，补全缺失的代码，(注意要符合编译器生成代码的常规情况):

```
absdiff:
    movq    %rdi, %rax
    subq    %rsi, %rax
    movq    %rsi, %rdx
    subq    %rdi, %rdx
    _____
    cmovg   %rdx, %rax
    ret
```

(4) 有些情况不适合使用条件传送指令，因为条件传送指令需要:

13. 考虑下面的汇编代码:

```
1 loop:
2     movl   %esi, %ecx
```

```

3    movl    $1, %edx
4    movl    $0, %eax
5    jmp     .L2
6  .L3:
7    movq    %rdi, %r8
8    andq    %rdx, %r8
9    orq     %r8, %rax
10   salq    %cl, %rdx
11  .L2:
12   testq   %rdx, %rdx
13   jne     .L3
14   rep; ret

```

补全下面对应的 C 语言代码:

```

long loop(long x, int n)
{
    long result = ____;
    long mask;
    for (mask = ____; mask != ____; mask = ____ ) {
        result |= ____;
    }
    return result;
}

```

14. 考虑下面汇编代码片段:

```

.section .rodata
    .align 8
.L1:
    .quad .L11
    .quad .L12
    .quad .L13
    .quad .L14
...
jmp *.L1(,%rdi,8)

```

若执行到 jmp 指令时, rdi 的值为 2, 则该指令会跳转到_____指向的位置。

第 6 讲 (每空 1 分, 16 分) 得分: _____

15. x86-64 中, 执行 `popq %rbx` 的操作依次为如下步骤:

- (1) 从寄存器_____中读出数据 x , 以 x 为地址, 读出对应内存中的数据 y ;
- (2) 将 (1) 中提到的寄存器 (加减多少): _____;
- (3) 将 y 存到寄存器_____中。

16. 以下两组对应的 C 语言代码和汇编语言代码:

<pre>void multstore (long x, long y, long *dest) { long t = mult2(x, y); *dest = t; }</pre>	<pre>0000000000400640 <multstore>: 400640: push %rbx 400641: mov %rdx,%rbx 400644: callq 400650 <mult2> 400649: mov %rax,(%rbx) 40064c: pop %rbx 40064d: retq</pre>
<pre>long mult2 (long a, long b) { long s = a * b; return s; }</pre>	<pre>0000000000400650 <mult2>: 400650: mov %rdi,%rax 400653: imul %rsi,%rax 400657: retq</pre>

(1) 假设准备执行 `callq` 指令时, `%rsp` 内容为 `0x110`, 那执行完 `callq` 指令后, `%rsp` 内容为_____, `%rsp` 指向的内存中的数值为_____. 随后执行完地址 400657 处的 `retq` 指令后, `%rsp` 内容为_____. (均为十六进制)

(2) C 代码中的下列变量分别保存在哪个寄存器中:

`x:` `y:` `a:` `b:` `s:`

(3) 汇编代码中, 地址 400641 的指令修改了 `%rbx`, 地址 400649 的指令使用了 `%rbx`, 而二者中间有过程调用 `call`, 是什么机制保证了 `%rbx` 不会被修改:

(4) 如果把 C 代码中的 `mult2(x, y)` 改为 `mult2(y, x)`, 汇编代码会有什么变化:

17. 考虑以下 C 语言代码:

```
long bit_not1(long a) {
    long x = ~a;
    return x;
}

long bit_not2(long *p) {
```

```

    long x = *p;
    long y = ~x
    return y;
}
long call_bitnot() {
    long v0 = 1010;
    long v1 = 1010;
    long v2 = bit_not1(v0);
    long v3 = bit_not2(&v1);
    return v2+v3;
}

```

下列变量分别存放在寄存器还是内存中？

v0: v1: v2: v3:

第 7 讲 (每空 2 分, 20 分) 得分: _____

18. 考虑下面的 C 语言代码：

```

#define CNT 4
int arr_data[CNT][CNT] =
    {{900, 901, 902, 903},
     {910, 911, 912, 913},
     {920, 921, 922, 923},
     {930, 931, 932, 933}};
int get_digit(int x, int y)
{
    return arr_data[x][y];
}

```

补全对应的汇编代码：

```

get_digit:
    shlq    $4,
    movl    arr_data(rdi,    ,    ),
    ret

```

19. C 语言中传参 $n \times n$ 二维数组时，如果 n 是变量，应该如何传递？补全下面的汇编代码。

```

# int var_ele(size_t n, int a[n][n], size_t i, size_t j)
# { return a[i][j]; }
# hint: %rdi,%rsi,%rdx,%rcx

```

```

var_ele:
    movq    %rdx, %rax
    _____, %rax
    _____, %rax
    movl    (%rsi,%rax,4), %eax
    ret

```

20. 考虑下面的代码，输出结果是: _____

```

#define ZLEN 5
#define PCOUNT 4
typedef int zip_dig[ZLEN];

int main() {
    zip_dig pgh[PCOUNT] =
        {{1, 5, 2, 0, 6},
         {1, 5, 2, 1, 3},
         {1, 5, 2, 1, 7},
         {1, 5, 2, 2, 1}};
    int *linear_zip = (int *) pgh;
    int *zip1 = (int *) pgh[1];
    int result =
        pgh[2][3] +
        linear_zip[14] +
        *(linear_zip + 16) +
        zip1[4];
    printf("result: %d\n", result);
}

```

21. 考虑下面的代码，A 和 B 是用 #define 定义的常数:

```

typedef struct {
    int x[A][B];    /* Unknown constants A and B */
    long y;
} str1;

typedef struct {
    char array[B];
    int t;
    short s[A];
}

```



```

    long u;
} str2;

void setVal(str1 *p, str2 *q) {
    long v1 = q->t;
    long v2 = q->u;
    p->y = v1 + v2;
}

```

GCC 为 setVal 产生如下汇编代码 (x86-64 System V 调用约定):

```

# p in %rdi, q in %rsi
setVal:
    movslq 8(%rsi), %rax
    addq 32(%rsi), %rax
    movq %rax, 184(%rdi)
    ret

```

求 A 和 B 的值是多少?

A: _____ B: _____

22. 考虑下面的 C 语言代码和对应的汇编语言代码:

```

double mix_compute(long *a, double *b, float c) {
    double result = *a + *b + c;
    return result;
}

mix_compute:
    movsd (%rsi), %xmm1
    movq (%rdi), %rax
    cvtsi2sd %rax, %xmm2
    addsd %xmm1, %xmm2
    cvtss2sd %xmm0, %xmm0
    addsd %xmm2, %xmm0
    ret

```

C 代码中的下列变量, 分别在哪个寄存器中:

b: _____ c: _____ result: _____

第 8 讲 (每空 2 分, 8 分) 得分: _____

23. 考虑下面的 C 语言代码:

```

union {
    unsigned char c[8];
    unsigned int i[2];
} dw;

int j;
for (j = 0; j < 8; j++)
    dw.c[j] = 0xC0 + j;

printf("Ints 0-1 == [0x%x,0x%x]\n", dw.i[0], dw.i[1]);

```

如果在 大端模式的计算机（如 Sun）上运行，输出结果是：

Ints 0-1 == [_____]

24. 课上讲授了防御或者避免缓冲区攻击的几种主要方法，分析它们的特点。以普遍性情况为准，不用考虑少见的软硬件需求。（每行算作一空）

	是否需要修改用户程序源代码？	是否需要重新编译用户程序？	是否需要更换新的 CPU 硬件？
使用 fgets() 等安全的库函数	是	是	否
栈初始位置随机化			
在内存访问权限上设置“可执行”位			
“金丝雀”（哨兵）机制			