计算机系统概论（2023 秋） 作业 2

1. 使用不超过 3 条 x86 指令实现如下函数：其中 x,y,z,w 分别存储于 %rdi, %rsi, %rdx, %rcx。 返回值存储于 %rax。

```
long add(long x, long y, long z, long w) {
    return 32 * x + 8 * y + 4 * z + w;
}
```

```
add:
    _____
    _____
    _____
    ret
```

2. X86-64 体系结构中的条件跳转指令 jg 是用于符号数比较还是无符号数比较的？其产生跳转的成立条件是~(SF^OF)&~ZF 为真，请解释为何是这一条件。

3. 有如下对应的 C 代码与汇编代码（x86-64），请对照着填上代码中缺失的部分（数字请用十进制表示）

```
call_swap:
        subq    $24, %rsp
        movl    ①, 12(%rsp)
        movl    $91125, 8(%rsp)
        leaq    8(%rsp), %rsi
        leaq    12(%rsp), ④
        movl    $0, %eax
        call     swap
```

```
void swap(int *a, int *b);
void call_swap()
{
  int zip1 = 15213;
  int zip2 = ②;
  ③;
}
```

① ：_____

② ：_____

③ ：_____

④ ：_____

4. 一个 C 语言的 for 循环代码（部分）及其 64 位 Linux 汇编如下所示，请对照汇编填充 C 语言里的缺失部分。

```
int looper(int n, int *a) {
  int i;
  int x = ____________;
  for(i = ____________;
    _________;
    i++)
  {
    if (_________)
          x = _________;
    else_________;
  }
  return x;
}
```

```
looper:
        movl    $0, %eax
        movl    $0, %edx
        jmp     .L2
.L4:
        movslq  %edx, %rcx
        movl    (%rsi,%rcx,4), %ecx
        addl    $1, %eax
        cmpl    %eax, %ecx
        jle     .L3
        leal    (%rcx,%rcx), %eax
.L3:
        addl    $1, %edx
.L2:
        cmpl    %edi, %edx
        jl      .L4
        ret
```

5. 对于如下代码

```
long v2permute(long *array, long x, long y, long z) {
    long t1 = 8253 * x;
    long t2 = array[t1 + 2 * y];
    long t3 = array[t2 * 16 + z];
    long t4 = t1 + t2 + t3;
    long t5 = array[0] * t1;
    long ret = t3 & t5;
    return ret;
}
```

对应如下汇编指令，请写出每条指令之后目标寄存器存储的变量/临时变量值

```
v2permute:
    movq    %rdx, %r8          _____
    movq    %rcx, %rdx         _____
    imulq   $8253, %rsi, %rax  _____
    leaq    (%rax,%r8,2), %rcx _____
    movq    (%rdi,%rcx,8), %rcx _____
    salq    $4, %rcx           _____
    addq    %rdx, %rcx         _____
    imulq   (%rdi), %rax       _____
    andq    (%rdi,%rcx,8), %rax _____
```

6. 请对照下面的 C 语言代码与相应汇编（Linux X86-64），给出 M、N 的值。

```
copy_element:
```

```
        movslq  %edi, %rdi
        movslq  %esi, %rsi
        leaq    (%rsi,%rsi,2), %rax
        leaq    (%rsi,%rax,4), %rax
        addq    %rdi, %rax
        movl    mat2(,%rax,4), %edx
        leaq    0(,%rdi,8), %rax
        subq    %rdi, %rax
        addq    %rax, %rsi
        movl    %edx, mat1(,%rsi,4)
        ret
```

```
#define M _____
#define N _____
int mat1[M][N];
int mat2[N][M];
int copy_element(int i, int j)
{
    mat1[i][j] = mat2[j][i];
}
```