3. 61

```
long cread(long *xp) {
    return (xp ? *xp : 0);
}

long cread_alt(long *xp) {
    return (!xp ? 0 : *xp);
}
```

3. 65

我们先假设M为4，我们假设矩阵A为：

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

那么在用函数transpose处理之后，矩阵变成了

1 5 9 13

2 6 10 14

3 7 11 15

4 8 12 16

可以看出对矩阵沿着对角线进行了转换。我们继续看汇编代码

下边的汇编代码只是函数中内循环中的代码

.L6:

movq (%rdx), %rcx

movq (%rax), %rsi

movq %rsi, (%rdx)

movq %rcx, (%rax)

addq $8, %rdx

addq $120, %rax

cmpq %rdi, %rax

jne .L6

我们很容易就发现，指向A[i][j]的寄存器为%rdx,指向A[j][i]的寄存器为%rax

求M最关键的是找出%rax寄存器移动的规律，因为%rdx也就是A[i][j] + 8 就表示右移一位

而%rax则要移动M * 8位

因此M = 120 / 8 = 15

上边的寄存器%rdi应该放的就是i == j时的A[i][j]的地址

3. 69

i in %rdi, bp in %rsi

test:

    mov 0x120(%rsi), %ecx

    add (%rsi), %ecx

    lea (%rdi,%rdi,4), %rax

    lea (%rsi,%rax,8), %rax

```
    mov 0x8(%rax), %rdx
    movslq %ecx, %rcx
    mov %rcx, 0x10(%rax,%rdx,8)
    retq
```

由 %rdx = (5 * i * 8 + bp) + 8 可以推导出 a_struct a[CNT] 每个元素占40个字节，first占8个字节

==>

CNT = (288 - 8) / 40 ==> CNT = 7

本题重点理解%rax 和 %rdx中保存的是什么的值，

%rax中保存的是ap的值，而%rdx中保存的是ap->idx的值，理解了这一层接下来就简单了

说明ap->idx保存的是8字节的值，根据 &(16 + %rax + 8 * %rdx) = %rcx 可以得出idx应该是结构体的第一个变量long idx

如果结构体占用了40个字节 ， 那么数组x应该占用 40 - 8 也就是32个字节，每个元素占8个，可以容纳4个元素

```
typedef struct {
    long idx;
    long x[4];
}a_struct;
```

这个题目最重要的地方是理解mov 0x8(%rax), %rdx 这段代码，它是求ap->idx的值。