**3.69 你负责维护一个大型的C程序，遇到下面的代码：

```
1  typedef struct {
2      int first;
3      a_struct a[CNT];
4      int last;
5  } b_struct;
6
7  void test(long i, b_struct *bp)
8  {
9      int n = bp->first + bp->last;
10     a_struct *ap = &bp->a[i];
11     ap->x[ap->idx] = n;
12 }
```

结构 - 示例7

i: %rdi, bp: %rsi

A. CNT 的值。

B. 结构 a_struct 的完整声明。

假设这个结构中只有字段 idx 和 x，这两个字段保存的都是 有符号值。

284<4+ 40*CNT <= 288

CNT = 7

idx为long

a_struct大小是40

288(%rsi)

```
void test(long i, b_struct *bp)
i in %rdi, bp in %rsi
1  0000000000000000 <test>:
2    0:  8b 8e 20 01 00 00     mov   0x120(%rsi),%ecx
3    6:  03 0e                 add   (%rsi),%ecx
4    8:  48 8d 04 bf           lea   (%rdi,%rdi,4),%rax
5    c:  48 8d 04 c6           lea   (%rsi,%rax,8),%rax
6   10:  48 8b 50 08           mov   0x8(%rax),%rdx
7   14:  48 63 c9              movslq %ecx,%rcx
8   17:  48 89 4c d0 10        mov   %rcx,0x10(%rax,%rdx,8)
9   1c:  c3                    retq
```

bp->last

bp->first + bp->last

5*i

bp+40*i

4字节转8字节，x[]为long

a_struct *ap : %rax

a_struct a[]

```
typedef struct {
    int first;
    a_struct a[CNT];
    int last;
} b_struct;

void test(long i, b_struct *bp)
{
    int n = bp->first + bp->last;
    a_struct *ap = &bp->a[i];
    ap->x[ap->idx] = n;
}
```

Test部分的第一个参数i存入寄存器rdi
第二个参数*bp的地址bp存入寄存器rsi

64位处理器按8字节作为处理单位

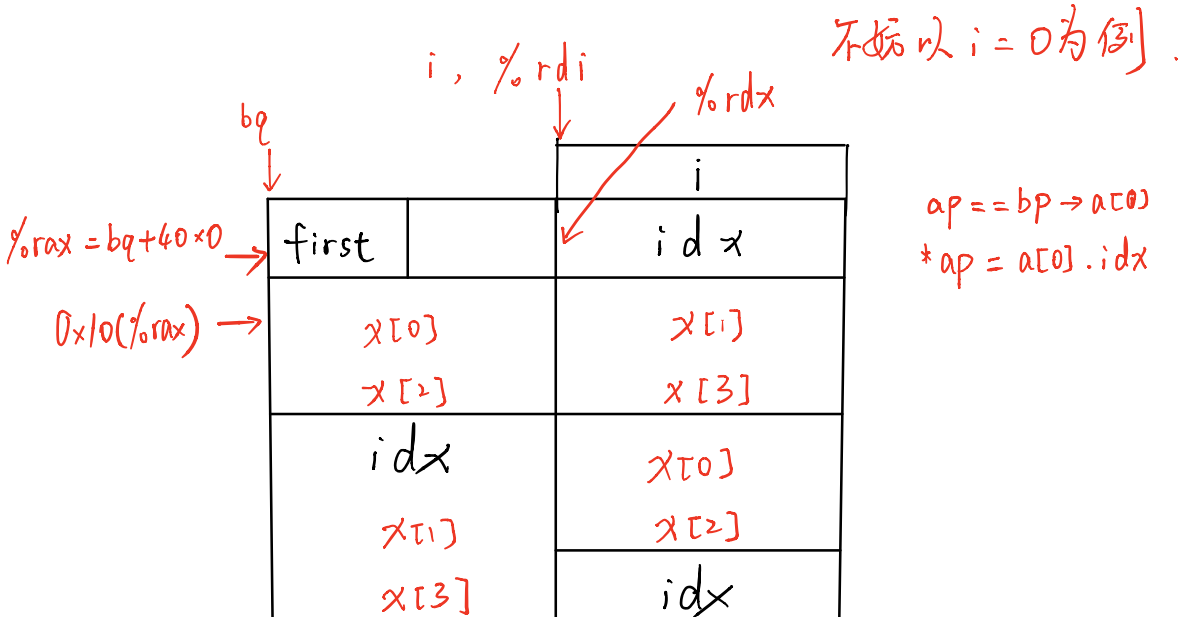
```
ict *bp)

:>:
) 00        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
```

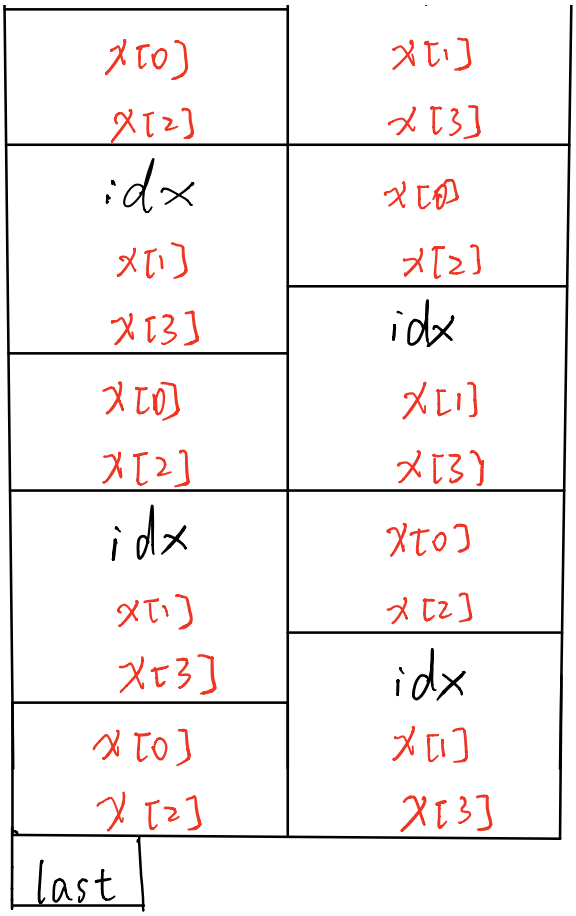
由该步操作，可知0x120(%rsi)所存值为bp->last
0x120对应10进制数为288，可知有284或280或276个比特用于存储a[CNT]. (因不能确定对齐之后first和last在哪个位置)

ecx的值为first+last
rax的值为bp+40i
于是可以推断出一个a-struct占用40个比特
40*7=280，故CNT=7

Rax的地址后8位开始的8比特被传送到rdx中，说明rax所存数据占8比特，故ap->Idx 为long型
bp+40i+8 处的值送至rdx，即

要用rcx来赋值，说明数据为8字节一组的数据，为long型；考虑到总长度为40，共有4个






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