当用优化等级-O1编译时，GCC为这个函数的内循环产生下面的代码：

```
.L6:
  movq    (%rdx), %rcx
  movq    (%rax), %rsi
  movq    %rsi, (%rdx)
  movq    %rcx, (%rax)
  addq    $8, %rdx
  addq    $120, %rax
  cmpq    %rdi, %rax
  jne     .L6
```

我们可以看到GCC把数组索引转换成了指针代码。

A. 哪个寄存器保存着指向数组元素A[i][j]的指针？

B. 哪个寄存器保存着指向数组元素A[j][i]的指针？

C. M的值是多少？

*3.66 考虑下面的源代码，这里NR和NC是用#define声明的宏表达式，计算用参数n表示的矩阵A的维度。这段代码计算矩阵的第j列的元素之和。

```
long sum_col(long n, long A[NR(n)][NC(n)], long j) {
    long i;
    long result = 0;
    for (i = 0; i < NR(n); i++)
        result += A[i][j];
    return result;
}
```

编译这个程序，GCC产生下面的汇编代码：

```
long sum_col(long n, long A[NR(n)][NC(n)], long j)
n in %rdi, A in %rsi, j in %rdx
sum_col:
  leaq    1(,%rdi,4), %r8
  leaq    (%rdi,%rdi,2), %rax
  movq    %rax, %rdi
  testq   %rax, %rax
  jle     .L4
  salq    $3, %r8
  leaq    (%rsi,%rdx,8), %rcx
  movl    $0, %eax
  movl    $0, %edx
.L3:
  addq    (%rcx), %rax
  addq    $1, %rdx
  addq    %r8, %rcx
  cmpq    %rdi, %rdx
  jne     .L3
  rep; ret
.L4:
  movl    $0, %eax
  ret
```

运用你的逆向工程技术，确定NR和NC的定义。

**3.67 这个作业要查看GCC为参数和返回值中有结构的函数产生的代码，由此可以看到这些语言特性通常是如何实现的。

下面的C代码中有一个函数process，它用结构作为参数和返回值，还有一个函数eval，它调用process：

```
typedef struct {
    long a[2];
```