```
4     addl    %r8d, %eax              Add to result
5     addq    $1, %rdx                j++
6     addq    %r9, %rcx               Bptr += n
7     cmpq    %rdi, %rdx              Compare j:n
8     jne     .L24                    If !=, goto loop
```

我们看到程序既使用了伸缩过的值 $4n$(寄存器%r9)来增加 Bptr，也使用了 n 的值(寄存器%rdi)来检查循环的边界。C 代码中并没有体现出需要这两个值，但是由于指针运算的伸缩，才使用了这两个值。

可以看到，如果允许使用优化，GCC 能够识别出程序访问多维数组的元素的步长。然后生成的代码会避免直接应用等式(3.1)会导致的乘法。不论生成基于指针的代码(图 3-37b)还是基于数组的代码(图 3-38b)，这些优化都能显著提高程序的性能。

3.9　异质的数据结构

C 语言提供了两种将不同类型的对象组合到一起创建数据类型的机制：结构(structure)，用关键字 struct 来声明，将多个对象集合到一个单位中；联合(union)，用关键字 union 来声明，允许用几种不同的类型来引用一个对象。

3.9.1　结构

C 语言的 struct 声明创建一个数据类型，将可能不同类型的对象聚合到一个对象中。用名字来引用结构的各个组成部分。类似于数组的实现，结构的所有组成部分都存放在内存中一段连续的区域内，而指向结构的指针就是结构第一个字节的地址。编译器维护关于每个结构类型的信息，指示每个字段(field)的字节偏移。它以这些偏移作为内存引用指令中的位移，从而产生对结构元素的引用。

给 C 语言初学者　将一个对象表示为 struct

C 语言提供的 struct 数据类型的构造函数(constructor)与 C++ 和 Java 的对象最为接近。它允许程序员在一个数据结构中保存关于某个实体的信息，并用名字来引用这些信息。

例如，一个图形程序可能要用结构来表示一个长方形：

```
struct rect {
    long llx;          /* X coordinate of lower-left corner */
    long lly;          /* Y coordinate of lower-left corner */
    unsigned long width;  /* Width (in pixels)                */
    unsigned long height; /* Height (in pixels)               */
    unsigned color;       /* Coding of color                  */
};
```

可以声明一个 struct rect 类型的变量 r，并将它的字段值设置如下：

```
struct rect r;
r.llx = r.lly = 0;
r.color = 0xFF00FF;
r.width = 10;
r.height = 20;
```

这里表达式 r.llx 就会选择结构 r 的 llx 字段。

另外，我们可以在一条语句中既声明变量又初始化它的字段：