列出的地址不同——链接器将这段代码的地址移到了一段不同的地址范围中。第二个不同之处在于链接器填上了 callq 指令调用函数 mult2 需要使用的地址(反汇编代码第 4 行)。链接器的任务之一就是为函数调用找到匹配的函数的可执行代码的位置。最后一个区别是多了两行代码(第 8 和 9 行)。这两条指令对程序没有影响，因为它们出现在返回指令后面(第 7 行)。插入这些指令是为了使函数代码变为 16 字节，使得就存储器系统性能而言，能更好地放置下一个代码块。

3.2.3　关于格式的注解

GCC 产生的汇编代码对我们来说有点儿难读。一方面，它包含一些我们不需要关心的信息，另一方面，它不提供任何程序的描述或它是如何工作的描述。例如，假设我们用如下命令生成文件 mstore.s。

```
linux> gcc -Og -S mstore.c
```

mstore.s 的完整内容如下：

```
        .file   "010-mstore.c"
        .text
        .globl  multstore
        .type   multstore, @function
multstore:
        pushq   %rbx
        movq    %rdx, %rbx
        call    mult2
        movq    %rax, (%rbx)
        popq    %rbx
        ret
        .size   multstore, .-multstore
        .ident  "GCC: (Ubuntu 4.8.1-2ubuntu1~12.04) 4.8.1"
        .section        .note.GNU-stack,"",@progbits
```

所有以'.'开头的行都是指导汇编器和链接器工作的伪指令。我们通常可以忽略这些行。另一方面，也没有关于指令的用途以及它们与源代码之间关系的解释说明。

为了更清楚地说明汇编代码，我们用这样一种格式来表示汇编代码，它省略了大部分伪指令，但包括行号和解释性说明。对于我们的示例，带解释的汇编代码如下：

```
    void multstore(long x, long y, long *dest)
    x in %rdi, y in %rsi, dest in %rdx
1   multstore:
2     pushq   %rbx             Save %rbx
3     movq    %rdx, %rbx       Copy dest to %rbx
4     call    mult2            Call mult2(x, y)
5     movq    %rax, (%rbx)     Store result at *dest
6     popq    %rbx             Restore %rbx
7     ret                      Return
```

通常我们只会给出与讨论内容相关的代码行。每一行的左边都有编号供引用，右边是注释，简单地描述指令的效果以及它与原始 C 语言代码中的计算操作的关系。这是一种汇编语言程序员写代码的风格。

我们还提供网络旁注，为专门的机器语言爱好者提供一些资料。一个网络旁注描述的是 IA32 机器代码。有了 x86-64 的背景，学习 IA32 会相当简单。另外一个网络旁注简要