置：loc_A(在汇编代码中标识为.L3)，loc_B(.L5)，loc_C(.L6)，loc_D(.L7)和 loc_def(.L8)，最后一个是默认的目的地址。每个标号都标识一个实现某个情况分支的代码块。在 C 和汇编代码中，程序都是将 index 和 6 做比较，如果大于 6 就跳转到默认的代码处。

```
    void switch_eg(long x, long n, long *dest)
    x in %rdi, n in %rsi, dest in %rdx
1   switch_eg:
2     subq    $100, %rsi              Compute index = n-100
3     cmpq    $6, %rsi                Compare index:6
4     ja      .L8                     If >, goto loc_def
5     jmp     *.L4(,%rsi,8)           Goto *jt[index]
6   .L3:                            loc_A:
7     leaq    (%rdi,%rdi,2), %rax     3*x
8     leaq    (%rdi,%rax,4), %rdi     val = 13*x
9     jmp     .L2                     Goto done
10  .L5:                            loc_B:
11    addq    $10, %rdi               x = x + 10
12  .L6:                            loc_C:
13    addq    $11, %rdi               val = x + 11
14    jmp     .L2                     Goto done
15  .L7:                            loc_D:
16    imulq   %rdi, %rdi              val = x * x
17    jmp     .L2                     Goto done
18  .L8:                            loc_def:
19    movl    $0, %edi                val = 0
20  .L2:                            done:
21    movq    %rdi, (%rdx)            *dest = val
22    ret                             Return
```

图 3-23　图 3-22 中 switch 语句示例的汇编代码

执行 switch 语句的关键步骤是通过跳转表来访问代码位置。在 C 代码中是第 16 行，一条 goto 语句引用了跳转表 jt。GCC 支持计算 goto(computed goto)，是对 C 语言的扩展。在我们的汇编代码版本中，类似的操作是在第 5 行，jmp 指令的操作数有前缀'*'，表明这是一个间接跳转，操作数指定一个内存位置，索引由寄存器%rsi 给出，这个寄存器保存着 index 的值。(我们会在 3.8 节中看到如何将数组引用翻译成机器代码。)

C 代码将跳转表声明为一个有 7 个元素的数组，每个元素都是一个指向代码位置的指针。这些元素跨越 index 的值 0～6，对应于 n 的值 100～106。可以观察到，跳转表对重复情况的处理就是简单地对表项 4 和 6 用同样的代码标号(loc_D)，而对于缺失的情况的处理就是对表项 1 和 5 使用默认情况的标号(loc_def)。

在汇编代码中，跳转表用以下声明表示，我们添加了一些注释：

```
1     .section        .rodata
2     .align 8          Align address to multiple of 8
3   .L4:
4     .quad   .L3       Case 100: loc_A
5     .quad   .L8       Case 101: loc_def
6     .quad   .L5       Case 102: loc_B
7     .quad   .L6       Case 103: loc_C
8     .quad   .L7       Case 104: loc_D
9     .quad   .L8       Case 105: loc_def
10    .quad   .L7       Case 106: loc_D
```