对C语言的扩展。数组jt包含7个表项，每个都是一个代码块的地址。这些位置由代码中的标号定义，在jt的表项中由代码指针指明，由标号加上‘&&’前缀组成。(回想运算符&创建一个指向数据值的指针。在做这个扩展时，GCC的作者们创造了一个新的运算符&&，这个运算符创建一个指向代码位置的指针。)建议你研究一下C语言过程switch_eg_impl，以及它与汇编代码版本之间的关系。

```
void switch_eg(long x, long n,
               long *dest)
{
    long val = x;

    switch (n) {

    case 100:
        val *= 13;
        break;

    case 102:
        val += 10;
        /* Fall through */

    case 103:
        val += 11;
        break;

    case 104:
    case 106:
        val *= val;
        break;

    default:
        val = 0;
    }
    *dest = val;
}
```

a) switch语句

```
1   void switch_eg_impl(long x, long n,
2                       long *dest)
3   {
4       /* Table of code pointers */
5       static void *jt[7] = {
6           &&loc_A, &&loc_def, &&loc_B,
7           &&loc_C, &&loc_D, &&loc_def,
8           &&loc_D
9       };
10      unsigned long index = n - 100;
11      long val;
12
13      if (index > 6)
14          goto loc_def;
15      /* Multiway branch */
16      goto *jt[index];
17
18   loc_A:    /* Case 100 */
19      val = x * 13;
20      goto done;
21   loc_B:    /* Case 102 */
22      x = x + 10;
23      /* Fall through */
24   loc_C:    /* Case 103 */
25      val = x + 11;
26      goto done;
27   loc_D:    /* Cases 104, 106 */
28      val = x * x;
29      goto done;
30   loc_def:  /* Default case */
31      val = 0;
32   done:
33      *dest = val;
34  }
```

b) 翻译到扩展的C语言

图3-22　switch语句示例以及翻译到扩展的C语言。该翻译给出了跳转表jt的结构，以及如何访问它。作为对C语言的扩展，GCC支持这样的表

原始的C代码有针对值100、102-104和106的情况，但是开关变量n可以是任意整数。编译器首先将n减去100，把取值范围移到0和6之间，创建一个新的程序变量，在我们的C版本中称为index。补码表示的负数会映射成无符号表示的大正数，利用这一事实，将index看作无符号值，从而进一步简化了分支的可能性。因此可以通过测试index是否大于6来判定index是否在0～6的范围之外。在C和汇编代码中，根据index的值，有五个不同的跳转位