格式翻译成 do-while 循环：

```
t = test-expr;
if (!t)
    goto done;
do
    body-statement
    while (test-expr);
done:
```

相应地，还可以把它翻译成 goto 代码如下：

```
t = test-expr;
if (!t)
    goto done;
loop:
    body-statement
    t = test-expr;
    if (t)
        goto loop;
done:
```

利用这种实现策略，编译器常常可以优化初始的测试，例如认为测试条件总是满足。

再来看个例子，图 3-21 给出了图 3-20 所示阶乘函数同样的 C 代码，不过给出的是 GCC 使用命令行选项-O1 时的编译。图 3-21c 给出实际生成的汇编代码，图 3-21b 是这个汇编代码更易读的 C 语言表示。根据 goto 代码，可以看到如果对于 n 的初始值有 $n \leqslant 1$，那么将跳过该循环。该循环本身的基本结构与该函数 do-while 版本产生的结构(图 3-19)一样。不过，一个有趣的特性是，循环测试(汇编代码的第 9 行)从原始 C 代码的 $n>1$ 变成了 $n \neq 1$。编译器知道只有当 $n>1$ 时才会进入循环，所以将 n 减 1 意味着 $n>1$ 或者 $n=1$。因此，测试 $n \neq 1$ 就等价于测试 $n \leqslant 1$。

```
long fact_while(long n)
{
    long result = 1;
    while (n > 1) {
        result *= n;
        n = n-1;
    }
    return result;
}
```

a）C代码

```
long fact_while_gd_goto(long n)
{
    long result = 1;
    if (n <= 1)
        goto done;
 loop:
    result *= n;
    n = n-1;
    if (n != 1)
        goto loop;
 done:
    return result;
}
```

b）等价的goto版本

图 3-21　使用 guarded-do 翻译方法的阶乘算法的 while 版本的 C 代码和汇编代码。函数 fact_while_gd_goto 说明了汇编代码版本的操作