


对 while 循环进行跳转到中间变换, 得到如下 goto 代码:


```
long fact_for_jm_goto(long n)
{
    long i = 2;
    long result = 1;
    goto test;
loop:
    result *= i;
    i++;
test:
    if (i <= n)
        goto loop;
    return result;
}
```

确实, 仔细查看使用命令行选项 -Og 的 GCC 产生的汇编代码, 会发现它非常接近于以下模板:

```
long fact_for(long n)
n in %rdi
fact_for:
    movl    $1, %eax           Set result = 1
    movl    $2, %edx           Set i = 2
    jmp     .L8                Goto test
.L9:
    imulq   %rdx, %rax          Compute result *= i
    addq    $1, %rdx            Increment i
.L8:
    cmpq    %rdi, %rdx          test:
                                Compare i:n
    jle     .L9                If <=, goto loop
    rep; ret                    Return
```

 **练习题 3.27** 先把 fact_for 转换成 while 循环, 再进行 guarded-do 变换, 写出 fact_for 的 goto 代码。

综上所述, C 语言中三种形式的所有的循环——do-while、while 和 for——都可以用一种简单的策略来翻译, 产生包含一个或多个条件分支的代码。控制的条件转移提供了将循环翻译成机器代码的基本机制。

 **练习题 3.28** 函数 fun_b 有如下整体结构:

```
long fun_b(unsigned long x) {
    long val = 0;
    long i;
    for ( ... ; ... ; ... ) {
        :
        :
    }
    return val;
}
```

GCC C 编译器产生如下汇编代码:

```
long fun_b(unsigned long x)
x in %rdi
1 fun_b:
2     movl    $64, %edx
3     movl    $0, %eax
4     .L10:
5     movq    %rdi, %rcx
```