```
6       andl    $1, %ecx
7       addq    %rax, %rax
8       orq     %rcx, %rax
9       shrq    %rdi            Shift right by 1
10      subq    $1, %rdx
11      jne     .L10
12      rep; ret
```

逆向工程这段代码的操作，然后完成下面的工作：

A. 根据汇编代码版本填写 C 代码中缺失的部分。

B. 解释循环前为什么没有初始测试也没有初始跳转到循环内部的测试部分。

C. 用自然语言描述这个函数是计算什么的。

练习题 3.29　在 C 语言中执行 continue 语句会导致程序跳到当前循环迭代的结尾。当处理 continue 语句时，将 for 循环翻译成 while 循环的描述规则需要一些改进。例如，考虑下面的代码：

```
/* Example of for loop containing a continue statement */
/* Sum even numbers between 0 and 9 */
long sum = 0;
long i;
for (i = 0; i < 10; i++) {
    if (i & 1)
        continue;
    sum += i;
}
```

A. 如果我们简单地直接应用将 for 循环翻译到 while 循环的规则，会得到什么呢？产生的代码会有什么错误呢？

B. 如何用 goto 语句来替代 continue 语句，保证 while 循环的行为同 for 循环的行为完全一样？

3.6.8　switch 语句

switch(开关)语句可以根据一个整数索引值进行多重分支(multiway branching)。在处理具有多种可能结果的测试时，这种语句特别有用。它们不仅提高了 C 代码的可读性，而且通过使用跳转表(jump table)这种数据结构使得实现更加高效。跳转表是一个数组，表项 i 是一个代码段的地址，这个代码段实现当开关索引值等于 i 时程序应该采取的动作。程序代码用开关索引值来执行一个跳转表内的数组引用，确定跳转指令的目标。和使用一组很长的 if-else 语句相比，使用跳转表的优点是执行开关语句的时间与开关情况的数量无关。GCC 根据开关情况的数量和开关情况值的稀疏程度来翻译开关语句。当开关情况数量比较多(例如 4 个以上)，并且值的范围跨度比较小时，就会使用跳转表。

图 3-22a 是一个 C 语言 switch 语句的示例。这个例子有些非常有意思的特征，包括情况标号(case label)跨过一个不连续的区域(对于情况 101 和 105 没有标号)，有些情况有多个标号(情况 104 和 106)，而有些情况则会落入其他情况之中(情况 102)，因为对应该情况的代码段没有以 break 语句结尾。

图 3-23 是编译 switch_eg 时产生的汇编代码。这段代码的行为用 C 语言来描述就是图 3-22b 中的过程 switch_eg_impl。这段代码使用了 GCC 提供的对跳转表的支持，这是