常难以阅读和调试。本文中使用 goto 语句，是为了构造描述汇编代码程序控制流的 C 程序。我们称这样的编程风格为“goto 代码”。

在 goto 代码中(图 3-16b)，第 5 行中的 goto x_ge_y 语句会导致跳转到第 9 行中的标号 x_ge_y 处(当 $x \geqslant y$ 时会进行跳转)。从这一点继续执行，完成函数 absdiff_se 的 else 部分并返回。另一方面，如果测试 x>=y 失败，程序会计算 absdiff_se 的 if 部分指定的步骤并返回。

汇编代码的实现(图 3-16c)首先比较了两个操作数(第 2 行)，设置条件码。如果比较的结果表明 x 大于或者等于 y，那么它就会跳转到第 8 行，增加全局变量 ge_cnt，计算 x-y 作为返回值并返回。由此我们可以看到 absdiff_se 对应汇编代码的控制流非常类似于 gotodiff_se 的 goto 代码。

```
long lt_cnt = 0;
long ge_cnt = 0;

long absdiff_se(long x, long y)
{
    long result;
    if (x < y) {
        lt_cnt++;
        result = y - x;
    }
    else {
        ge_cnt++;
        result = x - y;
    }
    return result;
}
```

a) 原始的C语言代码

```
1  long gotodiff_se(long x, long y)
2  {
3      long result;
4      if (x >= y)
5          goto x_ge_y;
6      lt_cnt++;
7      result = y - x;
8      return result;
9   x_ge_y:
10     ge_cnt++;
11     result = x - y;
12     return result;
13 }
```

b) 与之等价的goto版本

```
     long absdiff_se(long x, long y)
     x in %rdi, y in %rsi
1    absdiff_se:
2      cmpq    %rsi, %rdi          Compare x:y
3      jge     .L2                 If >= goto x_ge_y
4      addq    $1, lt_cnt(%rip)    lt_cnt++
5      movq    %rsi, %rax
6      subq    %rdi, %rax          result = y - x
7      ret                         Return
8    .L2:                        x_ge_y:
9      addq    $1, ge_cnt(%rip)    ge_cnt++
10     movq    %rdi, %rax
11     subq    %rsi, %rax          result = x - y
12     ret                         Return
```

c) 产生的汇编代码

图 3-16　条件语句的编译。a)C 过程 absdiff_se 包含一个 if-else 语句；b)C 过程 gotodiff_se 模拟了汇编代码的控制；c)给出了产生的汇编代码