指令 0x2 的地址处。减去这个值就得到地址 0x400545。注意，ja 指令的编码需要 2 个字节，它一定位于地址 0x400543 处。检查原始的反汇编代码也证实了这一点：

```
400543: 77 02                   ja     400547
400545: 5d                      pop    %rbp
```

D. 以相反的顺序来读这些字节，我们看到目标偏移量是 0xffffff73，或者十进制数 −141。0x4005ed(nop 指令的地址)加上这个值得到地址 0x400560：

```
4005e8: e9 73 ff ff ff          jmpq   400560
4005ed: 90                      nop
```

3.16　对汇编代码写注释，并且模仿它的控制流来编写 C 代码，是理解汇编语言程序很好的第一步。本题是一个具有简单控制流的示例，给你一个检查逻辑操作实现的机会。

A. 这里是 C 代码：

```
void goto_cond(long a, long *p) {
    if (p == 0)
        goto done;
    if (*p >= a)
        goto done;
    *p = a;
 done:
    return;
}
```

B. 第一个条件分支是 && 表达式实现的一部分。如果对 p 为非空的测试失败，代码会跳过对 a>*p 的测试。

3.17　这个练习帮助你思考一个通用的翻译规则的思想以及如何应用它。

A. 转换成这种替代的形式，只需要调换一下几行代码：

```
long gotodiff_se_alt(long x, long y) {
    long result;
    if (x < y)
        goto x_lt_y;
    ge_cnt++;
    result = x - y;
    return result;
 x_lt_y:
    lt_cnt++;
    result =  y - x;
    return result;
}
```

B. 在大多数情况下，可以在这两种方式中任意选择。但是原来的方法对常见的没有 else 语句的情况更好一些。对于这种情况，我们只用简单地将翻译规则修改如下：

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

基于这种替代规则的翻译更麻烦一些。

3.18　这个题目要求你完成一个嵌套的分支结构，在此你会看到如何使用翻译 if 语句的规则。大部分情况下，机器代码就是 C 代码的直接翻译。

```
long test(long x, long y, long z) {
    long val = x+y+z;
    if (x < -3) {
        if (y < z)
            val = x*y;
```