章研究链接非常重要。此外，它也能帮助理解反汇编器的输出。在汇编代码中，跳转目标用符号标号书写。汇编器，以及后来的链接器，会产生跳转目标的适当编码。跳转指令有几种不同的编码，但是最常用都是 PC 相对的(PC-relative)。也就是，它们会将目标指令的地址与紧跟在跳转指令后面那条指令的地址之间的差作为编码。这些地址偏移量可以编码为 1、2 或 4 个字节。第二种编码方法是给出“绝对”地址，用 4 个字节直接指定目标。汇编器和链接器会选择适当的跳转目的编码。

下面是一个 PC 相对寻址的例子，这个函数的汇编代码由编译文件 branch.c 产生。它包含两个跳转：第 2 行的 jmp 指令前向跳转到更高的地址，而第 7 行的 jg 指令后向跳转到较低的地址。

```
1     movq    %rdi, %rax
2     jmp     .L2
3   .L3:
4     sarq    %rax
5   .L2:
6     testq   %rax, %rax
7     jg      .L3
8     rep; ret
```

汇编器产生的“.o”格式的反汇编版本如下：

```
1      0:   48 89 f8              mov    %rdi,%rax
2      3:   eb 03                 jmp    8 <loop+0x8>
3      5:   48 d1 f8              sar    %rax
4      8:   48 85 c0              test   %rax,%rax
5      b:   7f f8                 jg     5 <loop+0x5>
6      d:   f3 c3                 repz retq
```

右边反汇编器产生的注释中，第 2 行中跳转指令的跳转目标指明为 0x8，第 5 行中跳转指令的跳转目标是 0x5(反汇编器以十六进制格式给出所有的数字)。不过，观察指令的字节编码，会看到第一条跳转指令的目标编码(在第二个字节中)为 0x03。把它加上 0x5，也就是下一条指令的地址，就得到跳转目标地址 0x8，也就是第 4 行指令的地址。

类似，第二个跳转指令的目标用单字节、补码表示编码为 0xf8(十进制-8)。将这个数加上 0xd(十进制 13)，即第 6 行指令的地址，我们得到 0x5，即第 3 行指令的地址。

这些例子说明，当执行 PC 相对寻址时，程序计数器的值是跳转指令后面的那条指令的地址，而不是跳转指令本身的地址。这种惯例可以追溯到早期的实现，当时的处理器会将更新程序计数器作为执行一条指令的第一步。

下面是链接后的程序反汇编版本：

```
1      4004d0:   48 89 f8              mov    %rdi,%rax
2      4004d3:   eb 03                 jmp    4004d8 <loop+0x8>
3      4004d5:   48 d1 f8              sar    %rax
4      4004d8:   48 85 c0              test   %rax,%rax
5      4004db:   7f f8                 jg     4004d5 <loop+0x5>
6      4004dd:   f3 c3                 repz retq
```

这些指令被重定位到不同的地址，但是第 2 行和第 5 行中跳转目标的编码并没有变。通过使用与 PC 相对的跳转目标编码，指令编码很简洁(只需要 2 个字节)，而且目标代码可以不做改变就移到内存中不同的位置。