当任一操作数为 *NaN* 时，就会出现无序的情况。可以通过奇偶标志位发现这种情况。通常 jp(jump on parity)指令是条件跳转，条件就是浮点比较得到一个无序的结果。除了这种情况以外，进位和零标志位的值都和对应的无符号比较一样：当两个操作数相等时，设置 ZF；当 $S_2<S_1$ 时，设置 CF。像 ja 和 jb 这样的指令可以根据标志位的各种组合进行条件跳转。

来看一个浮点比较的例子，图 3-51a 中的 C 函数会根据参数 x 与 0.0 的相对关系进行分类，返回一个枚举类型作为结果。C 中的枚举类型是编码为整数的，所以函数可能的值为：0(NEG)，1(ZERO)，2(POS)和 3(OTHER)。当 x 的值为 *NaN* 时，会出现最后一种结果。

```
typedef enum {NEG, ZERO, POS, OTHER} range_t;

range_t find_range(float x)
{
    int result;
    if (x < 0)
        result = NEG;
    else if (x == 0)
        result = ZERO;
    else if (x > 0)
        result = POS;
    else
        result = OTHER;
    return result;
}
```

a）C代码

```
    range_t find_range(float x)
    x in %xmm0
1   find_range:
2     vxorps  %xmm1, %xmm1, %xmm1          Set %xmm1 = 0
3     vucomiss        %xmm0, %xmm1         Compare 0:x
4     ja      .L5                          If >, goto neg
5     vucomiss        %xmm1, %xmm0         Compare x:0
6     jp      .L8                          If NaN, goto posornan
7     movl    $1, %eax                     result = ZERO
8     je      .L3                          If =, goto done
9   .L8:                                 posornan:
10    vucomiss        .LC0(%rip), %xmm0    Compare x:0
11    setbe   %al                          Set result = NaN ? 1 : 0
12    movzbl  %al, %eax                    Zero-extend
13    addl    $2, %eax                     result += 2 (POS for > 0, OTHER for NaN)
14    ret                                  Return
15  .L5:                                 neg:
16    movl    $0, %eax                     result = NEG
17  .L3:                                 done:
18    rep; ret                             Return
```

b）产生的汇编代码

图 3-51　浮点代码中的条件分支说明