练习题 3.56　考虑下面的 C 函数，其中 EXPR 是用 # define 定义的宏：

```
double simplefun(double x) {
    return EXPR(x);
}
```

下面，我们给出了为不同的 EXPR 定义生成的 AVX2 代码，其中，x 的值保存在%xmm0 中。这些代码都对应于某些对浮点数值有用的操作。确定这些操作都是什么。要理解从内存中取出的常数字的位模式才能找出答案。

A.
```
1     vmovsd  .LC1(%rip), %xmm1
2     vandpd  %xmm1, %xmm0, %xmm0
3   .LC1:
4     .long   4294967295
5     .long   2147483647
6     .long   0
7     .long   0
```

B.
```
1     vxorpd  %xmm0, %xmm0, %xmm0
```

C.
```
1     vmovsd  .LC2(%rip), %xmm1
2     vxorpd  %xmm1, %xmm0, %xmm0
3   .LC2:
4     .long   0
5     .long   -2147483648
6     .long   0
7     .long   0
```

3.11.6　浮点比较操作

AVX2 提供了两条用于比较浮点数值的指令：

指令	基于	描述
ucomiss S_1, S_2	S_2-S_1	比较单精度值
ucomisd S_1, S_2	S_2-S_1	比较双精度值

这些指令类似于 CMP 指令(参见 3.6 节)，它们都比较操作数 S_1 和 S_2(但是顺序可能与预计的相反)，并且设置条件码指示它们的相对值。与 cmpq 一样，它们遵循以相反顺序列出操作数的 ATT 格式惯例。参数 S_2 必须在 XMM 寄存器中，而 S_1 可以在 XMM 寄存器中，也可以在内存中。

浮点比较指令会设置三个条件码：零标志位 ZF、进位标志位 CF 和奇偶标志位 PF。3.6.1 节中我们没有讲奇偶标志位，因为它在 GCC 产生的 x86 代码中不太常见。对于整数操作，当最近的一次算术或逻辑运算产生的值的最低位字节是偶校验的(即这个字节中有偶数个 1)，那么就会设置这个标志位。不过对于浮点比较，当两个操作数中任一个是 NaN 时，会设置该位。根据惯例，C 语言中如果有个参数为 NaN，就认为比较失败了，这个标志位就被用来发现这样的条件。例如，当 x 为 NaN 时，比较 x==x 都会得到 0。

条件码的设置条件如下：

顺序 $S_2:S_1$	CF	ZF	PF
无序的	1	1	1
$S_2<S_1$	1	0	0
$S_2=S_1$	0	1	0
$S_2>S_1$	0	0	0