即，这两条指令的最终效果是将原始的%xmm0低位4字节中的单精度值转换成双精度值，再将其两个副本保存到%xmm0中。我们不太清楚GCC为什么会生成这样的代码，这样做既没有好处，也没有必要在XMM寄存器中把这个值复制一遍。

对于把双精度转换为单精度，GCC会产生类似的代码：

```
    Conversion from double to single precision
1   vmovddup    %xmm0, %xmm0      Replicate first vector element
2   vcvtpd2psx  %xmm0, %xmm0      Convert two vector elements to single
```

假设这些指令开始执行前寄存器%xmm0保存着两个双精度值$[x_1, x_0]$。然后vmovddup指令把它设置为$[x_0, x_0]$。vcvtpd2psx指令把这两个值转换成单精度，再存放到该寄存器的低位一半中，并将高位一半设置为0，得到结果$[0.0, 0.0, x_0, x_0]$(回想一下，浮点值0.0是由位模式全0表示的)。同样，用这种方式把一种精度转换成另一种精度，而不用下面的单条指令，没有明显直接的意义：

```
vcvtsd2ss %xmm0, %xmm0, %xmm0
```

下面是一个不同浮点转换操作的例子，考虑以下C函数

```
double fcvt(int i, float *fp, double *dp, long *lp)
{
    float f = *fp; double d = *dp; long l = *lp;
    *lp = (long)    d;
    *fp = (float)   i;
    *dp = (double)  l;
    return (double) f;
}
```

以及它对应的x86-64汇编代码

```
    double fcvt(int i, float *fp, double *dp, long *lp)
    i in %edi, fp in %rsi, dp in %rdx, lp in %rcx
1   fcvt:
2     vmovss  (%rsi), %xmm0                  Get f = *fp
3     movq    (%rcx), %rax                   Get l = *lp
4     vcvttsd2siq     (%rdx), %r8            Get d = *dp and convert to long
5     movq    %r8, (%rcx)                    Store at lp
6     vcvtsi2ss       %edi, %xmm1, %xmm1     Convert i to float
7     vmovss  %xmm1, (%rsi)                  Store at fp
8     vcvtsi2sdq      %rax, %xmm1, %xmm1     Convert l to double
9     vmovsd  %xmm1, (%rdx)                  Store at dp
      The following two instructions convert f to double
10    vunpcklps       %xmm0, %xmm0, %xmm0
11    vcvtps2pd       %xmm0, %xmm0
12    ret                                    Return f
```

fcvt的所有参数都是通过通用寄存器传递的，因为它们既不是整数也不是指针。结果通过寄存器%xmm0返回。如图3-45中描述的，这是float或double值指定的返回寄存器。在这段代码中，可以看到图3-46～图3-48中的许多传送和转换指令，还可以看到GCC将单精度转换为双精度的方法。

练习题3.50　对于下面的C代码，表达式val1～val4分别对应程序值i、f、d和l：

```
double fcvt2(int *ip, float *fp, double *dp, long l)
```