

- 函数使用寄存器 %xmm0 来返回浮点值。
- 所有的 XMM 寄存器都是调用者保存的。被调用者可以不用保存就覆盖这些寄存器中任意一个。

当函数包含指针、整数和浮点数混合的参数时，指针和整数通过通用寄存器传递，而浮点值通过 XMM 寄存器传递。也就是说，参数到寄存器的映射取决于它们的类型和排列的顺序。下面是一些例子：

```
double f1(int x, double y, long z);
```


这个函数会把 x 放在 %edi 中， y 放在 %xmm0 中，而 z 放在 %rsi 中。

```
double f2(double y, int x, long z);
```

这个函数的寄存器分配与函数 $f1$ 相同。

```
double f1(float x, double *y, long *z);
```

这个函数会将 x 放在 %xmm0 中， y 放在 %rdi 中，而 z 放在 %rsi 中。

 **练习题 3.52** 对于下面每个函数声明，确定参数的寄存器分配：

- `double g1(double a, long b, float c, int d);`
- `double g2(int a, double *b, float *c, long d);`
- `double g3(double *a, double b, int c, float d);`
- `double g4(float a, int *b, float c, double d);`

3.11.3 浮点运算操作

图 3-49 描述了一组执行算术运算的标量 AVX2 浮点指令。每条指令有一个 (S_1) 或两个 (S_1, S_2) 源操作数，和一个目的操作数 D 。第一个源操作数 S_1 可以是一个 XMM 寄存器或一个内存位置。第二个源操作数和目的操作数都必须是 XMM 寄存器。每个操作都有一条针对单精度的指令和一条针对双精度的指令。结果存放在目的寄存器中。

单精度	双精度	效果	描述
vaddss	vaddsd	$D \leftarrow S_2 + S_1$	浮点数加
vsubss	vsubsd	$D \leftarrow S_2 - S_1$	浮点数减
vmulss	vmulsd	$D \leftarrow S_2 \times S_1$	浮点数乘
vdivss	vdivsd	$D \leftarrow S_2 / S_1$	浮点数除
vmaxss	vmaxsd	$D \leftarrow \max(S_2, S_1)$	浮点数最大值
vminss	vminsd	$D \leftarrow \min(S_2, S_1)$	浮点数最小值
sqrtps	sqrtsd	$D \leftarrow \sqrt{S_1}$	浮点数平方根

图 3-49 标量浮点算术运算。这些指令有一个或两个源操作数和一个目的操作数

来看一个例子，考虑下面的浮点函数：

```
double funct(double a, float x, double b, int i)
{
    return a*x - b/i;
}
```

x86-64 代码如下：