

```

{
    int i = *ip; float f = *fp; double d = *dp;
    *ip = (int)    val1;
    *fp = (float)  val2;
    *dp = (double) val3;
    return (double) val4;
}


```

根据该函数如下的 x86-64 代码，确定这个映射关系：

```

double fcvt2(int *ip, float *fp, double *dp, long l)
ip in %rdi, fp in %rsi, dp in %rdx, l in %rcx
Result returned in %xmm0
1  fcvt2:
2      movl    (%rdi), %eax
3      vmovss  (%rsi), %xmm0
4      vcvtsd2si    (%rdx), %r8d
5      movl    %r8d, (%rdi)
6      vcvtsi2ss    %eax, %xmm1, %xmm1
7      vmovss  %xmm1, (%rsi)
8      vcvtsi2sdq    %rcx, %xmm1, %xmm1
9      vmovsd  %xmm1, (%rdx)
10     vunpcklps    %xmm0, %xmm0, %xmm0
11     vcvtps2pd    %xmm0, %xmm0
12     ret

```

 **练习题 3.51** 下面的 C 函数将类型为 `src_t` 的参数转换为类型为 `dst_t` 的返回值，这里两种数据类型都用 `typedef` 定义：

```

dest_t cvt(src_t x)
{
    dest_t y = (dest_t) x;
    return y;
}

```

在 x86-64 上执行这段代码，假设参数 `x` 在 `%xmm0` 中，或者在寄存器 `%rdi` 的某个适当的命名部分中（即 `%rdi` 或 `%edi`）。用一条或两条指令来完成类型转换，并把结果复制制到寄存器 `%rax` 的某个适当命名部分中（整数结果），或 `%xmm0` 中（浮点结果）。给出这条或这些指令，包括源和目的寄存器。

$T_s$	$T_d$	指令
long	double	<code>vcvtsi2sdq %rdi,%xmm0</code>
double	int	
double	float	
long	float	
float	long	

### 3.11.2 过程中的浮点代码

在 x86-64 中，XMM 寄存器用来向函数传递浮点参数，以及从函数返回浮点值。如图 3-45 所示，可以看到如下规则：

- XMM 寄存器 `%xmm0`~`%xmm7` 最多可以传递 8 个浮点参数。按照参数列出的顺序使用这些寄存器。可以通过栈传递额外的浮点参数。