练习题 3.4　假设变量 sp 和 dp 被声明为类型

```
src_t  *sp;
dest_t *dp;
```

这里 src_t 和 dest_t 是用 typedef 声明的数据类型。我们想使用适当的数据传送指令来实现下面的操作

```
*dp = (dest_t) *sp;
```

假设 sp 和 dp 的值分别存储在寄存器%rdi 和%rsi 中。对于表中的每个表项，给出实现指定数据传送的两条指令。其中第一条指令应该从内存中读数，做适当的转换，并设置寄存器%rax 的适当部分。然后，第二条指令要把%rax 的适当部分写到内存。在这两种情况中，寄存器的部分可以是%rax、%eax、%ax 或%al，两者可以互不相同。

记住，当执行强制类型转换既涉及大小变化又涉及 C 语言中符号变化时，操作应该先改变大小(2.2.6 节)。

src_t	dest_t	指令
long	long	movq(%rdi),%rax movq %rax,(%rsi)
char	int	
char	unsigned	
unsigned char	long	
int	char	
unsigned	unsigned char	
char	short	

给 C 语言初学者　指针的一些示例

函数 exchange(图 3-7a)提供了一个关于 C 语言中指针使用的很好说明。参数 xp 是一个指向 long 类型的整数的指针，而 y 是一个 long 类型的整数。语句

```
long x = *xp;
```

表示我们将读存储在 xp 所指位置中的值，并将它存放到名字为 x 的局部变量中。这个读操作称为指针的间接引用(pointer dereferencing)，C 操作符* 执行指针的间接引用。

语句

```
*xp = y;
```

正好相反——它将参数 y 的值写到 xp 所指的位置。这也是指针间接引用的一种形式(所以有操作符*)，但是它表明的是一个写操作，因为它在赋值语句的左边。

下面是调用 exchange 的一个实际例子：

```
long a = 4;
long b = exchange(&a, 3);
printf("a = %ld, b = %ld\n", a, b);
```