出条件传送指令的操作数长度，所以对所有的操作数长度，都可以使用同一个的指令名字。

指令	同义名	传送条件	描述
cmove S, R	cmovz	ZF	相等/零
cmovne S, R	cmovnz	~ZF	不相等/非零
cmovs S, R		SF	负数
cmovns S, R		~SF	非负数
cmovg S, R	cmovnle	~(SF ^ OF) & ~ZF	大于（有符号>）
cmovge S, R	cmovnl	~(SF ^ OF)	大于或等于（有符号>=）
cmovl S, R	cmovnge	SF ^ OF	小于（有符号<）
cmovle S, R	cmovng	(SF ^ OF) | ZF	小于或等于（有符号<=）
cmova S, R	cmovnbe	~CF & ~ZF	超过（无符号>）
cmovae S, R	cmovnb	~CF	超过或相等（无符号>=）
cmovb S, R	cmovnae	CF	低于（无符号<）
cmovbe S, R	cmovna	CF | ZF	低于或相等（无符号<=）

图 3-18　条件传送指令。当传送条件满足时，指令把源值 S 复制到目的 R。有些指令是“同义名”，即同一条机器指令的不同名字

同条件跳转不同，处理器无需预测测试的结果就可以执行条件传送。处理器只是读源值(可能是从内存中)，检查条件码，然后要么更新目的寄存器，要么保持不变。我们会在第 4 章中探讨条件传送的实现。

为了理解如何通过条件数据传输来实现条件操作，考虑下面的条件表达式和赋值的通用形式：

```
v = test-expr ? then-expr : else-expr;
```

用条件控制转移的标准方法来编译这个表达式会得到如下形式：

```
    if (!test-expr)
        goto false;
    v = then-expr;
    goto done;
false:
    v = else-expr;
done:
```

这段代码包含两个代码序列：一个对 *then-expr* 求值，另一个对 *else-expr* 求值。条件跳转和无条件跳转结合起来使用是为了保证只有一个序列执行。

基于条件传送的代码，会对 *then-expr* 和 *else-expr* 都求值，最终值的选择基于对 *test-expr* 的求值。可以用下面的抽象代码描述：

```
v  = then-expr;
ve = else-expr;
t  = test-expr;
if (!t) v = ve;
```

这个序列中的最后一条语句是用条件传送实现的——只有当测试条件 t 满足时，vt 的值才会被复制到 v 中。

不是所有的条件表达式都可以用条件传送来编译。最重要的是，无论测试结果如何，