指令	同义名	效果	设置条件
sete D	setz	D ← ZF	相等/零
setne D	setnz	D ← ~ZF	不等/非零
sets D		D ← SF	负数
setns D		D ← ~SF	非负数
setg D	setnle	D ← ~(SF ^ OF) & ~ZF	大于（有符号>）
setge D	setnl	D ← ~(SF ^ OF)	大于等于（有符号>=）
setl D	setnge	D ← SF ^ OF	小于（有符号<）
setle D	setng	D ← (SF ^ OF) | ZF	小于等于（有符号<=）
seta D	setnbe	D ← ~CF & ~ZF	超过（无符号>）
setae D	setnb	D ← ~CF	超过或相等（无符号>=）
setb D	setnae	D ← CF	低于（无符号<）
setbe D	setna	D ← CF | ZF	低于或相等（无符号<=）

图 3-14　SET 指令。每条指令根据条件码的某种组合，将一个字节设置为 0 或者 1。有些指令有“同义名”，也就是同一条机器指令有别的名字

```
int comp(data_t a, data_t b)
a in %rdi, b in %rsi
1  comp:
2    cmpq    %rsi, %rdi      Compare a:b
3    setl    %al             Set low-order byte of %eax to 0 or 1
4    movzbl  %al, %eax       Clear rest of %eax (and rest of %rax)
5    ret
```

注意 cmpq 指令的比较顺序(第 2 行)。虽然参数列出的顺序先是%rsi(b)再是%rdi(a)，实际上比较的是 a 和 b。还要记得，正如在 3.4.2 节中讨论过的那样，movzbl 指令不仅会把%eax 的高 3 个字节清零，还会把整个寄存器%rax 的高 4 个字节都清零。

某些底层的机器指令可能有多个名字，我们称之为“同义名(synonym)”。比如说，setg(表示“设置大于”)和 setnle(表示“设置不小于等于”)指的就是同一条机器指令。编译器和反汇编器会随意决定使用哪个名字。

虽然所有的算术和逻辑操作都会设置条件码，但是各个 SET 命令的描述都适用的情况是：执行比较指令，根据计算 t =a-b 设置条件码。更具体地说，假设 a、b 和 t 分别是变量 a、b 和 t 的补码形式表示的整数，因此 $t=a-_w^t b$，这里 w 取决于 a 和 b 的大小。

来看 sete 的情况，即“当相等时设置(set when equal)”指令。当 $a=b$ 时，会得到 $t=0$，因此零标志置位就表示相等。类似地，考虑用 setl，即“当小于时设置(set when less)”指令，测试一个有符号比较。当没有发生溢出时(OF 设置为 0 就表明无溢出)，我们有当 $a-_w^t b<0$ 时 $a<b$，将 SF 设置为 1 即指明这一点，而当 $a-_w^t b\geqslant 0$ 时 $a\geqslant b$，由 SF 设置为 0 指明。另一方面，当发生溢出时，我们有当 $a-_w^t b>0$(负溢出)时 $a<b$，而当 $a-_w^t b<0$(正溢出)时 $a>b$。当 $a=b$ 时，不会有溢出。因此，当 OF 被设置为 1 时，当且仅当 SF 被设置为 0，有 $a<b$。将这些情况组合起来，溢出和符号位的 EXCLUSIVE-OR 提供了 $a<b$ 是否为真的测试。其他的有符号比较测试基于 SF ^ OF 和 ZF 的其他组合。

对于无符号比较的测试，现在设 a 和 b 是变量 a 和 b 的无符号形式表示的整数。在执行计算 t=a-b 中，当 $a-b<0$ 时，CMP 指令会设置进位标志，因而无符号比较使用的是