


进位标志和零标志的组合。

注意到机器代码如何区分有符号和无符号值是很重要的。同 C 语言不同，机器代码不会将每个程序值都和一个数据类型联系起来。相反，大多数情况下，机器代码对于有符号和无符号两种情况都使用一样的指令，这是因为许多算术运算对无符号和补码算术都有一样的位级行为。有些情况需要用不同的指令来处理有符号和无符号操作，例如，使用不同版本的右移、除法和乘法指令，以及不同的条件码组合。


 **练习题 3.13** 考虑下列的 C 语言代码：

```
int comp(data_t a, data_t b) {
    return a COMP b;
}
```

它给出了参数 *a* 和 *b* 之间比较的一般形式，这里，参数的数据类型 *data\_t* (通过 `typedef` 被声明为表 3-1 中列出的某种整数类型，可以是有符号的也可以是无符号的 `comp` 通过 `#define` 来定义。

假设 *a* 在 `%rdi` 中某个部分，*b* 在 `%rsi` 中某个部分。对于下面每个指令序列，确定哪种数据类型 *data\_t* 和比较 `COMP` 会导致编译器产生这样的代码。(可能有多个正确答案，请列出所有的正确答案。)

- A. `cmpl %esi, %edi`  
`setl %al`
- B. `cmpw %si, %di`  
`setge %al`
- C. `cmpb %sil, %dil`  
`setbe %al`
- D. `cmpq %rsi, %rdi`  
`setne %a`

 **练习题 3.14** 考虑下面的 C 语言代码：

```
int test(data_t a) {
    return a TEST 0;
}
```

它给出了参数 *a* 和 0 之间比较的一般形式，这里，我们可以用 `typedef` 来声明 *data\_t*，从而设置参数的数据类型，用 `#define` 来声明 `TEST`，从而设置比较的类型。对于下面每个指令序列，确定哪种数据类型 *data\_t* 和比较 `TEST` 会导致编译器产生这样的代码。(可能有多个正确答案，请列出所有的正确答案。)

- A. `testq %rdi, %rdi`  
`setge %al`
- B. `testw %di, %di`  
`sete %al`
- C. `testb %dil, %dil`  
`seta %al`
- D. `testl %edi, %edi`  
`setne %al`

### 3.6.3 跳转指令

正常执行的情况下，指令按照它们出现的顺序一条一条地执行。跳转(`jump`)指令会导致执行切换到程序中一个全新的位置。在汇编代码中，这些跳转的目的地通常用一个标号