(label)指明。考虑下面的汇编代码序列(完全是人为编造的)：

```
    movq $0,%rax              Set %rax to 0
    jmp .L1                   Goto .L1
    movq (%rax),%rdx          Null pointer dereference (skipped)
.L1:
    popq %rdx                 Jump target
```

指令 jmp .L1 会导致程序跳过 movq 指令，而从 popq 指令开始继续执行。在产生目标代码文件时，汇编器会确定所有带标号指令的地址，并将跳转目标(目的指令的地址)编码为跳转指令的一部分。

图 3-15 列举了不同的跳转指令。jmp 指令是无条件跳转。它可以是直接跳转，即跳转目标是作为指令的一部分编码的；也可以是间接跳转，即跳转目标是从寄存器或内存位置中读出的。汇编语言中，直接跳转是给出一个标号作为跳转目标的，例如上面所示代码中的标号“.L1”。间接跳转的写法是‘* ’后面跟一个操作数指示符，使用图 3-3 中描述的内存操作数格式中的一种。举个例子，指令

```
jmp *%rax
```

用寄存器%rax 中的值作为跳转目标，而指令

```
jmp *(%rax)
```

以%rax 中的值作为读地址，从内存中读出跳转目标。

指令	同义名	跳转条件	描述
jmp *Label*		1	直接跳转
jmp **Operand*		1	间接跳转
je *Label*	jz	ZF	相等/零
jne *Label*	jnz	~ZF	不相等/非零
js *Label*		SF	负数
jns *Label*		~SF	非负数
jg *Label*	jnle	~(SF ^ OF) & ~ZF	大于(有符号>)
jge *Label*	jnl	~(SF ^ OF)	大于或等于(有符号>=)
jl *Label*	jnge	SF ^ OF	小于(有符号<)
jle *Label*	jng	(SF ^ OF) | ZF	小于或等于(有符号<=)
ja *Label*	jnbe	~CF & ~ZF	超过(无符号>)
jae *Label*	jnb	~CF	超过或相等(无符号>=)
jb *Label*	jnae	CF	低于(无符号<)
jbe *Label*	jna	CF | ZF	低于或相等(无符号<=)

图 3-15　jump 指令。当跳转条件满足时，这些指令会跳转到一条带标号的目的地。有些指令有“同义名”，也就是同一条机器指令的别名

表中所示的其他跳转指令都是有条件的——它们根据条件码的某种组合，或者跳转，或者继续执行代码序列中下一条指令。这些指令的名字和跳转条件与 SET 指令的名字和设置条件是相匹配的(参见图 3-14)。同 SET 指令一样，一些底层的机器指令有多个名字。条件跳转只能是直接跳转。

3.6.4　跳转指令的编码

虽然我们不关心机器代码格式的细节，但是理解跳转指令的目标如何编码，这对第 7