

<pre> for (i=0; i&lt;10; i++) {     body } next statement </pre>	<pre> set r4 to zero label1: compare r4 to 10 branch to label2 if &gt;= code to perform body increment r4 branch to label1 label2: code for next statement </pre>
<b>(a)</b>	<b>(b)</b>

**Figure 8.3** (a) A *for* statement used in a high-level language, and (b) the equivalent assembly language code using register 4 as an index.

### 8.13 Assembly Code For Indefinite Iteration

In programming language terminology, *indefinite iteration* refers to a loop that executes zero or more times. Typically, a high-level language uses the keyword *while* to indicate indefinite iteration. Figure 8.4 shows the assembly language equivalent of a *while* statement.

<pre> while (condition) {     body } next statement </pre>	<pre> label1: code to compute condition branch to label2 if not true code to perform body branch to label1 label2: code for next statement </pre>
<b>(a)</b>	<b>(b)</b>

**Figure 8.4** (a) A *while* statement used in a high-level language, and (b) the equivalent assembly language code.

### 8.14 Assembly Code For Procedure Invocation

Architects use the term *procedure* or *subroutine* to refer to a piece of code that can be invoked, and the terms *procedure call* or *subroutine call* to refer to the invocation. The key idea is that when a subroutine is invoked, the processor records the location from which the call occurred, and resumes execution at that point once the subroutine completes. Thus, a given subroutine can be invoked from multiple points in a program because control always passes back to the location from which the invocation occurred.