Besides the assignment operators, the only operators that modify their operands are increment and decrement. When using these operators, be careful that your expressions don't depend on a particular order of evaluation. In the following example, j may be assigned one of two values:

```
i = 2;
j = i * i++;
```

It's natural to assume that j is assigned the value 4. However, the effect of executing the statement is undefined, and j could just as well be assigned 6 instead. Here's the scenario: (1) The second operand (the original value of i) is fetched, then i is incremented. (2) The first operand (the new value of i) is fetched. (3) The new and old values of i are multiplied, yielding 6. "Fetching" a variable means to retrieve the value of the variable from memory. A later change to the variable won't affect the fetched value, which is typically stored in a special location (known as a ***register***) inside the CPU.

registers ➤ *18.2*

Undefined Behavior

According to the C standard, statements such as `c = (b = a + 2) - (a = 1);` and `j = i * i++;` cause ***undefined behavior,*** which is different from implementation-defined behavior (see Section 4.1). When a program ventures into the realm of undefined behavior, all bets are off. The program may behave differently when compiled with different compilers. But that's not the only thing that can happen. The program may not compile in the first place, if it compiles it may not run, and if it does run, it may crash, behave erratically, or produce meaningless results. In other words, undefined behavior should be avoided like the plague.

4.5 Expression Statements

C has the unusual rule that *any* expression can be used as a statement. That is, any expression—regardless of its type or what it computes—can be turned into a statement by appending a semicolon. For example, we could turn the expression ++i into a statement:

```
++i;
```

When this statement is executed, i is first incremented, then the new value of i is fetched (as though it were to be used in an enclosing expression). However, since ++i isn't part of a larger expression, its value is discarded and the next statement executed. (The change to i is permanent, of course.)

Q&A

Since its value is discarded, there's little point in using an expression as a statement unless the expression has a side effect. Let's look at three examples. In