There are too many contributions to Jikes RVM to mention, but we thank the Jikes RVM development community for all their work. Just under 100 people have contributed code to Jikes RVM, and 19 people have served as members of the Jikes RVM core team. For a list of full credits, it is worth going to the Jikes RVM website. More details about the early history of Jikes RVM and the growth of its open source community can be found in a 2005 *IBM System Journal* paper (Alpern et al. 2005).

Bootstrapping a Self-Hosting Runtime

Compared to the bootstrap of a traditional compiler (Figure 10-1), the bootstrap of a metacircular runtime involves a few more tricks. Figure 10-2 shows a T-diagram depicting the process.

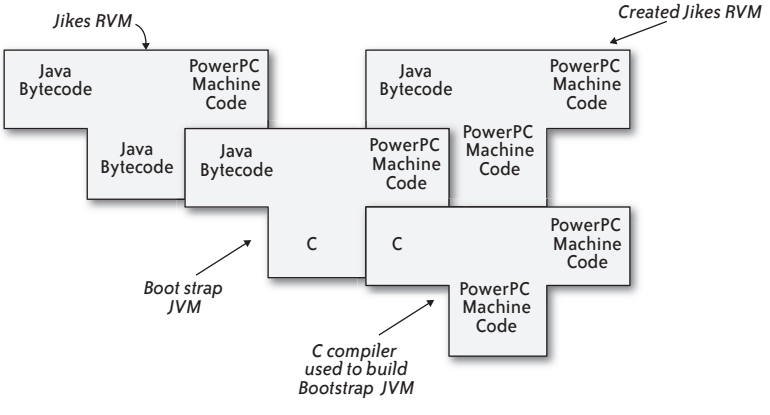

FIGURE 10-2. A T-Diagram showing the bootstrapping of Jikes RVM on an existing JVM written in C

The *boot image* contains several files that represent memory when the system is hosting itself (the rightmost T in Figure 10-2). The contents of the boot image are code and data, similar to what is found in a regular compiler's object file. An extra section in Jikes RVM's boot image contains the root map, which is created by the garbage collector. We describe the root map later in "Garbage Collection." The *boot image writer* is a program that uses Jikes RVM's compilers to create the boot image files, executing on a bootstrap JVM. A loader is responsible for loading the boot image into the correct area of memory, and in Jikes RVM, the loader is known as the *boot image runner*.

Object Layout

The boot image writer must lay out the objects on disk as they will be used in the running Jikes RVM. The object model in Jikes RVM is configurable, allowing different design alternatives to be evaluated while keeping the rest of the system fixed. An example of a design alternative is