Terminology

VM core

The precise composition of the VM core is open to discussion and debate. However, I think a safe, broad definition of it is that part of the VM which brings together the major components such as JITs, classloaders, scheduler, and GC. It's the hub in the wheel and is responsible for the main VM bootstrap (bootstrapping the classloader, starting the scheduler, memory manager, compiler etc).

VM init

The bootstrap of the VM has a number of elements to it, including gathering command line arguments, and starting the various components (above).

In the context of a Java-in-Java VM, the above is all written in Java.

VM boot image

The boot image is an image of a VM heap constructed ahead of time and populated with Java objects including code objects corresponding to the VM core and other elements of the VM necessary for the VM bootstrap (all written in Java, compiled ahead of time, packaged into Java objects and composed into a boot image). The boot image construction phase requires a working host VM (ideally the VM is self-hosting).

VM bootloader

In the case of Jikes RVM a dozen or so lines of assember and a few lines of C are required to basically do the job of any boot loader loader---mmap a boot image and throw the instruction pointer into it. It will also marshal argv and make it available to the VM core. This is technically interesting, but actually pretty trivial and has little to do with the VM core (aside from ensuring the instruction pointer lands a nice place within the boot image 😉

OS interface

The VM must talk to the OS (for file IO, signal handling, etc). There is not a whole lot to it, but a Java wrapper around OS functionality is required if the VM is java-in-java. This wrapper is pretty trivial and one half of it will (by necessity) be written in C.