Let's first consider what's needed for an exception. Every exception needs to create a list of methods that are on the stack at the point of the exception (known as the stack trace) and to pass control to a catch-block that handles the exception. Null pointer exceptions are efficiently handled by allowing memory protection failures to occur—for example, by reading or writing to a page of memory that doesn't exist. When a fault is generated, the address of the faulting instruction is provided to a handler. From this, the stack trace and handler information must be determined.

A VM that is written in C may try to use the C calling conventions for the Java JIT compiled code. A problem here is that the C calling conventions don't record what code is running. To solve this problem, a C VM can try to use the return address to compute what method is running. Every method within the VM must be searched to see whether its corresponding compiled code contains a given return address. This must be repeated for every method that is on the stack.

As Jikes RVM controls its memory layout, the stack is laid out to contain extra information to handle exceptions quickly. In particular, Jikes RVM places an identifier on the stack that is just a simple lookup away from providing method information. To determine the bytecode where the exception occurred and the handler locations, a computation is made using the return address or faulting instruction address. This means handling an exception's speed is proportional to the depth of the stack, and not dependent on the number of methods inside the VM.

In common with other VMs, Jikes RVM will optimize away exceptions when possible, in the best case reducing them to branches. It will also inline many methods into one method, so placing an identifier on the stack occurs infrequently, only when going between noninlined methods.

Magic, Annotations, and Making Things Go Smoothly

The code generated by the compiler has instructions that directly access memory. On other occasions memory needs to be accessed directly—for example, to implement garbage collection routines or access the object header for locking. To allow strongly typed memory accesses, Jikes RVM uses a library called *VM Magic*, which has been taken as a standard interface and used in other JVM projects. VM Magic defines classes that are either compiler pragma annotations, which extend the Java language, or unboxed types, which represent access to the underlying memory system. The unboxed type `Address` is used to directly access memory.

The VM magic unboxed types are handled specially within Jikes RVM's compilers. All the method calls upon an unboxed type are treated as manipulating a word-sized value directly. For example, the `plus` method is treated as an addition of the underlying word size of the machine, and directly acts upon what would normally be the `this` pointer of the object. As the value being manipulated isn't an object reference, the compiler records that the places in which