This model has been applied to other parts of domain zero. The latest versions of Xen include *stub domains*, which provide device support for "hardware virtualized" domains (described in the following section). Moving this code into isolated domains allows better performance isolation, improves robustness, and—somewhat surprisingly—improves raw performance. As development continues, more features may be moved out of domain zero, especially where doing so might improve security.

Changing Hardware, Changing Xen

Up to this point, our discussion has concentrated on paravirtualization. However, between Xen versions 2.0 and 3.0, Intel and AMD introduced distinct but similar support in their processors for *hardware virtual machines*. It became possible to run unmodified operating systems, including Microsoft Windows or native Linux, in virtual machines. So did this spell the end for paravirtualization?

First of all, let's look at how hardware virtual machines are implemented. Both Intel and AMD introduced a new mode (*nonroot mode* on Intel and *guest mode* on AMD) in which attempting to execute a privileged operation, even at the highest (virtual) privilege level, generates an exception that notifies the hypervisor. Therefore it is no longer necessary to scan the code and replace these instructions (either at runtime or in advance through paravirtualization). The hypervisor can use shadow page tables to provide the virtual machine with an illusion of contiguous memory, and it can trap I/O operations in order to emulate physical devices.

Xen added support for hardware virtual machines in version 3.0. The transition was aided greatly by open source development. Since Xen is an open source project, it was possible for developers from Intel and AMD to contribute low-level code that supports the new processors. Furthermore, thanks to its GPL status, Xen could incorporate code from other open source projects. For example, the new hardware virtual machines required an emulated BIOS and emulated hardware devices; implementing either of these would require a huge development effort. Fortunately, Xen could call on the open source BIOS from the Bochs project and emulated devices from QEMU.

EMULATION VERSUS VIRTUALIZATION

The latest version of Xen includes code from Bochs and QEMU, which are both *emulators*. What is the difference between emulation and virtualization, and how can the two combine?

Bochs provides an open source implementation, in software, of the x86 family of processors, as well as the supporting hardware. QEMU emulates several architectures, including the x86. Both can be used to run unmodified x86 operating systems and applications. Moreover, because they include a full implementation of the hardware—including the CPU—they can run on hardware that uses an incompatible instruction set.