13 Postscript: More Fun to Go

We shall not cease from exploration, and at the end we will arrive where we started, and know the place for the first time.
—T. S. Eliot (1888–1965)

Congratulations! You have finished the construction of a complete computing system. We hope that you enjoyed this journey. Let us, the authors of this book, share a secret with you: We suspect that we enjoyed writing the book even more. After all, we got to *design* this computing system, and design is often the ''funnest'' part of every project. We are sure that some of you, adventurous readers, would like to get in on some of that design action. Maybe you would like to improve the architecture; maybe you have ideas for adding new features here and there; maybe you envision a wider system. And then, maybe, you just want to be in the navigator's seat and decide *where* to go, not only *how* to get there.

Many alternative design elements can be implemented by modifying and extending the software that you have written in the various projects. For example, the assembly language, the Jack language, and the operating system can be modified and extended at will, by changing their specifications and rewriting portions of your respective assembler, compiler, and OS implementations. Other changes would likely require modification of the software *supplied by us*. For example, if you change the VM specification or the hardware specification, then you would probably want to change the respective emulators as well. Or if you want to add a new input or output device to the Hack computer, you would probably need to model them as built-in chips in the hardware simulator.

In order to allow complete flexibility of modifications and extensions, we are making all the source code of the software associated with the book publicly available. All our code is 100 percent Java, expect for the batch files used for starting the software on the Windows and Linux platforms. The software and its documentation

are available from the book's Web site at http://www.idc.ac.il/tecs. You are welcome to modify and extend all our tools as you deem desirable for your latest idea—and then share them with others, if you want. We hope that our code is written and documented well enough to make modification a satisfying experience. In particular, we wish to mention that the supplied hardware simulator has a simple and well-documented interface for adding new "built-in" chips. This interface can be used for extending the simulated hardware platform with, say, disk storage or communications devices.

While we cannot even start to imagine what *your* design improvements may be, we can briefly sketch some of the ones *we* were thinking of.

13.1 Hardware Realizations

Every hardware module presented in the book was software-based and HDL-simulated. This, in fact, is how hardware is actually designed. However, at some point the HDL designs are committed to silicon, becoming "real computers." Wouldn't it be nice to make Hack or Jack also run on some "real platform," made from some "real stuff"? Several different approaches may be taken towards this goal. On one extreme, you can attempt to nearly directly fabricate a real chip using the existing HDL design of Hack, and then deal with implementation issues related to the RAM, ROM, and I/O devices. Another extreme approach may be to attempt emulation (of either Hack, the VM, or even the Jack platform) on some existing hardware device like a cell phone or a PDA. It seems that any such project would want to reduce the size of the Hack screen as to keep the cost of the hardware resources reasonable.

13.2 Hardware Improvements

Although Hack is a *stored program computer*, the program that it runs must be prestored in its ROM device. In the present Hack architecture, there is no way of loading another program into the computer under user control, except for simulating the replacement of the entire physical ROM chip. Adding a "load program" capability in a balanced way would likely involve changes at several levels of the hierarchy. The Hack hardware can be modified to allow loaded programs to reside in a writable RAM rather than in the existing ROM. Some type of permanent storage (e.g., a

disk-on-chip) can probably be added to the hardware, to allow storage of programs. The operating system can be extended to handle this permanent storage device, as well as new logic for loading and running programs. At this point some kind of an OS user interface (“shell” or “DOS window”) would come in handy.

13.3 High-Level Languages

Like all professionals, programmers have strong feelings about their tools—the programming languages they use—and like to personalize them. And indeed, the Jack language, which leaves much to be desired, can be significantly improved or completely replaced (e.g., how about *Scheme*?). Some changes are simple, some are more involved, and some would likely require modifying the VM specification (e.g., adding real inheritance).

13.4 Optimizations

The book has almost completely sidestepped optimization issues (except for chapter 12, which introduced some efficiency measures). Optimization is a great playfield for every hacker. You can start with local optimizations in the existing compiler or hardware (or, in our platform, the best bang for the buck will probably come from optimizing the VM translator). Ambitious optimizations on a more global scale will involve changing specifications of interfaces such as the machine language or the VM language.

13.5 Communications

Wouldn’t it be nice to connect the Hack computer to the Internet? This could probably be done by adding a built-in communication chip to the hardware and writing some OS code to deal with it and to handle higher-level communication protocols. Some other programs would need to “talk” with the simulated communication chip, providing an interface to the Internet. For example, an HTTP-speaking Web browser in Jack seems like a feasible and worthy project.

These are some of our design itches—what are yours?