a complex but fast hard disk controller (or drive). A simple and reliable hard disk emulation will easily suffice, and the same holds true for all other hardware components. Even when looking at the processor, you can choose a Pentium II as the target, and there is no need to emulate instruction set extensions that are present in the latest chips. Because modern software, including operating systems, typically work perfectly well without such instructions to ensure backward compatibility, this decision is not significantly limiting.

Nevertheless, emulation speed remains the major challenge. The obvious way to get the best performance is to use some form of dynamic binary translation to move from a laborious step-by-step software emulation to a compiled mode where the raw speed of the underlying hardware is more efficiently exploited. This technique is used in many different guises; the modern x86 processor breaks the x86 instructions on first use into smaller microcodes, which it then caches for quick repeated execution. Just-In-Time-compiled Java environments similarly compile blocks of bytecodes into native code to improve the speed of execution for code that is repeated many times. The effectiveness of such techniques is due to the fact that in almost all software, 10% of the code represents 90% of the execution time.§ Finding simple ways of improving the speed of repeating this 10% of "hot" code can increase the overall speed dramatically. Also note that applying this technique does not imply compiling all the code; rather, selective optimization can result in a massive performance gain without the unacceptable delay that would ensue if all code were optimized.

The fact that virtually all software, whether compiled into native x86 from C/C++ or Java bytecode, is subjected to various amounts of dynamic binary translation by modern computers prior to execution shows the power and effectiveness of this technique. Thus, when approaching the task of creating JPC, there is hope for reasonable speed if similar techniques can be applied.

By gaining reassurance from existing emulators that this work is plausible, and by selecting a simple PC architecture as the initial target, we reduced the original scope to an achievable level. We then needed to assess the potential of dynamic binary translation to improve speed and ensure that a realistic performance could be achieved, even in principle. If the best outcome for JPC with all programming tricks applied was to run at 1%, then the project would not have been worth it.

Potential Processor Performance Tests

In order to evaluate the potential performance level achievable using various emulation tactics, a "Toy" processor was invented as a simple model. The Toy processor has 13 instructions,

§ This apocryphal rule of thumb was actually verified using JPC during the boot sequence of DOS and when playing numerous DOS games. With the total control of an emulator, it is easy to compile such statistics on program execution. See also Donald E. Knuth's "An empirical study of FORTRAN programs." (*Software Practice and Experience*, 1: 105-133. Wiley, 1971.)