

14 Building a Runnable Program

14.10 Explorations

- 14.20 Assuming you have access to `gcc`, run it with various of the compile-time flags that cause it to dump its RTL intermediate code. Recent versions of the compiler support about thirty such flags. Most have both a long descriptive name (e.g., `-fdump-rtl-cse` for a dump after common subexpression elimination) and a shorter abbreviation of the form `-dX`, where `X` is a single letter. Ask a local Unix guru to help you find and access the `gcc.info` files, which document RTL, the compile-time flags, and the various compiler phases.
- 14.21 Find out how linking works under your favorite operating system. Can code be dynamically linked? Can (nonprivileged) users create shared libraries? How does the loader find the libraries that need to be linked to a program? If your compiler can be instructed to generate position-independent code, how does this code compare (in size and run-time efficiency) with the non-position-independent equivalent?
- 14.22 Learn about *pointer swizzling* [Wil92a], originally developed to run programs on machines with insufficient virtual address space. Explain its connection to dynamic linking.

