Mystery Readme

Paul Jones
Computer Architecture (01:198:211)
School of Arts and Sciences
Rutgers University

April 4, 2013

1 My Process

The way that I worked out what `mystery.s` does by first creating a `mystery.c` file, and commenting in all of `mystery.s`. This allowed me to interact directly with the assembly code, adding comments as insights came to me and tracing through it.

I also compiled an executable and gathered information about the function calls and logic by performing an `objdump` and `readelf` on them. This allowed me to see some of the logic flow and function names.

With this information and my own understanding of the `mystery.s` file from my `mystery.c` comment (which have been handed in for reference) allowed me to fully reproduce the assembly code in C ("Decompiling").

2 Compiler Optimization

While it is impossible to irrefutable ascertain what the original C code was from the compiled assembly, it is highly likely that there were more variables than there were blocks of memory claimed for use by the program.

The reason this is possible is because the compiler knows ahead of time how many registers will be required, and can (usually) make the number of registers less than the total number of variables in the program, and move the literals on and off as they are needed.

This made it frustrating to read because sometimes registers previously used for some other purpose suddenly become re-commissioned for a new purpose (and then back again).