Background

CS31 students learn MIPS assembly using a program called *SPIM*. SPIM is 'MIPS' spelled backwards, and it is backwards. The interface is obtuse; the debugger is difficult to use; there's no good preprocessor, and the entire experience seems designed to discourage experimentation.

Yet, overall, SPIM is a success. Students learn a real-world assembly language while avoiding many hardware-level distractions. Even SPIM's pathetic debugger is enough to hunt down most problems. The develop-run-test loop is reasonably comfortable.

Our Idea

We will implement a better SPIM using C++ and QT. Our application will be a MIPS IDE, incorporating a MIPS simulator and debugger, as well as a text-editor. It will be CS31's DrScheme.

The interface and the debugger/executor will be in separate threads, so we'll have synchronization to deal with. We will heavily use the STL and QT. We'll support Windows, OS X, and Linux. We'll stick up a sourceforge site.

The core features we will support are:

- A MIPS simulator.
  - Our simulator will read and run MAL assembly source. We will not support TAL, nor compiled binaries.
  - A very extensible (possibly plugin-based) architecture for parsing, executing, and reversing[1] instructions. Ideally, it will be extendable to support other assembly languages. These other back-ends will be written by Open Source developers in far-off lands.
- A reversible, very graphical debugger
  - Users can set breakpoints, set watchpoints, step through the program, and edit code while program execution has paused.
  - If the user sees a problem, he/she can back-up to before the problem occurred, fix the error in the code, then continue.
- A great interface.
  - We'll use QT to create an Eclipse-ish interface. Running, editing, and debugging will all be available in one window.

[1] Our reversible debugger will need every instruction to 'know' how to reverse itself.