Initial Final Project Design

tfischer, tim

Our project will be split into three main parts:

- Main GUI and Visual tie-in with the debugger
    - Simplified, Eclipse-like interface with ‘windows’ for main code view/editor, register view with ability to ‘lock’/’watch’ registers, memory view with ability to ‘lock’/’watch’ addresses, output console, possibly kate-like input console
    - Syntax highlighting, breakpoints, code editing on-the-fly during a pause in execution without restarting program
    - Possibly a very simple plug-in architecture for mapping syscalls to different events
- Core MIPS simulation engine
    - Basic Preprocessor (allowing #defines)
    - Parser and class hierarchy for different instructions
    - System state abstraction that all three core components know about at all times, including memory/register status, etc.
    - Interface to GUI to allow for modifying code during paused execution
- Reversible Debugger
    - Start out with a normal debugger and then concentrate on making it reversible
    - Closely tied in to the two main components above; with the GUI to allow for visual debugging, and to the simulation engine to ensure that every instruction is reversible, either by having an inverse routine for some instructions (add, for ex), or by having a snapshot taken right before executing that instruction (for non-deterministic instructions such as throwing an exception)

Our initial plan for distribution of the work is to have Travis work on the GUI, Tim work on the simulation engine, and jointly work on the debugger.