A tool suited for learning LC3

In 210, we learn the model computer LC3 (Little Computer 3); this is a purely theoretical computer that teaches a simplified version of how computers work. The model shows the basics of how computers process instructions and IO. My learning experience was hindered by the outdated tooling around LC3, with the main simulator the course uses not fully conforming to the model we learnt in class.

Many tools exist for LC3, but they are all some combination of:

- Hard to use.
- Highly outdated.
- Opinionated away from the course textbook.
- Not available outside of windows.

So what do I offer?

In my free time, I have created a simulator for the LC3 computer equipped with many tools to input and debug machine code and understand how the internals respond to input.

I have shared this tool online via a website: 210tools.github.io, and many people have already found benefits in using this tool to study. It is fully cross-platform with desktop apps for Mac, Linux, windows, etc., and a website you can access on your phone.

Of course, this tool is incomplete, and I have a list of over 80 feature ideas that I could implement, ranging from trivial to more work than everything I have done. I have, however, already done a large portion of the work.

Here is a non-exhaustive list of some of the current key features of the tool:

- Full emulation of the LC3 system conforming to the course textbook.
- A handwritten OS designed to be easy for students to read and understand.
- A tokenizer and parser for the high-level assembly language offering detailed errors so students can quickly correct syntax issues.
- Compilation to machine code and flashing emulator memory in one button in < 10 microseconds[1].
- Complete interactive documentation on the LC3 assembly language.
- Breakpoints to help with debugging.
- So many other minor things.

As the tool stands, it already boasts a better feature set than most other LC3 tools.

I would like to see these features in a final product:

- An easy mode that would remove some of the detailed parts of the simulation that got in the way of learning the basics of the assembly (EG: The OS layer)
- History for every value and rollback debugging at any time
  - Historical value tracking with time graphs
  - Tool to sort and filter changes.
- Themes creation tool
- Store program state
  - except for the emulator (only the panels and stuff)
- Emulator snapshot and restore
- Make use of the spare OS traps to implement util functions (like mult)

[1] On my machine, 500 lines of code.

As strange as it may sound, these goals are achievable, and a week of work is enough to sort them out.

Implementation

The entire tool is written in Rust, a systems language like C++, so it's pretty damn fast. I have ensured that it targets WASM and has tests for macOS, Linux, Windows and the web. Because it is a wasm binary, no special hosting is required, and a static page works just fine (GitHub provides unlimited static hosting for FREE!).

I have kept correctness and conformity with the textbook in mind when designing the emulator, and every op has a unit test.

I have set up on the GitHub repository automatic checks for code correctness so that on the occasion I break something, I will be alerted that a test failed, so I am very sure that the core emulator code is good; the UI code is where all the bugs are to be found.

Thanks

I hope that this tool can be of use to future semesters teaching 210 and mabye outside the university of auckland in the future.