ASSEMBLEXPERIENCE

ABSTRACT

The goal of this project is to write an assembler, and simulator for the LC-3 instruction set that is easy to understand, easily modifiable, and open-source so it can become a powerful learning tool in the hands of professors and students. In order to make this project easy to understand, I've chosen to write my program in Python due to its easy to read syntax. Easy modification is also achieved through the use of Python because of its ease of use with UI libraries such as PyQt5 and its powerful network of built-in and third-party libraries. Lastly the open-source aspect will be achieved through public availability of code.

CHALLENGES

One Person Team:

- I was interested in pursuing a project that would expand my understanding of lower-level computing, this was not a popular project topic and so I set off on my own.

Reverse Engineering:

- One of my project requirements was to be able to generate a .obj file that is used by other proprietary LC3 software. In order to do this I had to reverse engineer the method of .obj file creation by creating them myself and examining their contents because no internet sources covered this topic.

Understanding the User:

- I made this software with a goal of enhancing user understanding of my topic, this meant that I also wanted students to be able to understand how an assembler and disassembler work by viewing the source code. I had to consider this in the way that I wrote and documented my code, making sure it could be understood by the average freshman to sophomore computer science student.

AUTHOR

TREVOR DARST

ADVISOR

JOHN GALLAGHER

ROADMAP

1 ASSEMBLER

The first step was creating an assembler, I wanted to tackle this first because I needed to be able to assemble LC-3 code in order to simulate it. Additionally, understanding how to assemble the LC-3 language would naturally deepen my understanding of it which would be needed for writing the simulator.

2 OBJECT FILE

I needed to be able to create a file that would follow some sort of encoding. All other proprietary simulators used the .obj file format so I had to reverse engineer this format by picking through created code

3 SIMULATOR

Now that I could assemble code into a .obj file, I needed to develop an ISA-level simulator that would allow sdents to step through the code, or just run it altogether.

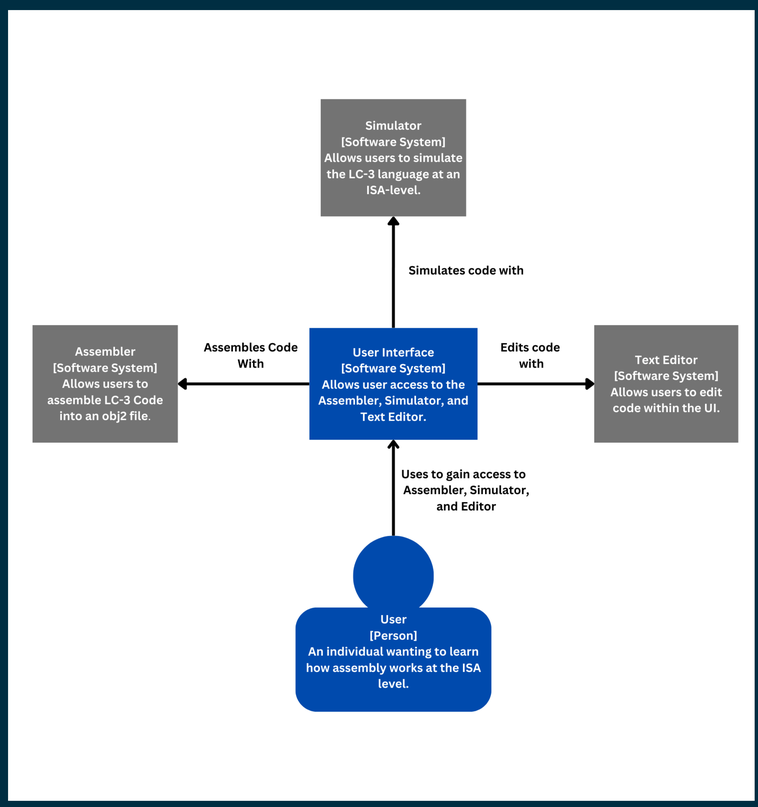
4 UI

In order to host all of my software components in a productive manner, I needed to create a UI. I achieved this through use of PyQt5 which I chose due to my familiarity with it. I also created a text editor for users to write code at their convenience.

5 TEST

I had already followed test-driven development fundamentals throughout the development of all of my software components, with over 90% unit test coverage. It was time to start doing heavy integration testing through actual development and simulation using the UI. I achieved this with help of my project advisor.

DESIGN



FUNCTIONALITY

AssembleXperience UI:

- **Assembler:**
  - Assembles LC3 code .asm file into .bin, .hex, and .obj files
  - Provides error output containing line location, location contents, and error type.
- **Simulator:**
  - Takes .obj file generated by assembler and simulates the processing of instructions at the ISA level.
  - Allows user to run or step through the program.
  - Stepping through the program displays the program counter, register, and flag states after each particular instruction execution.
- **Text Editor**
  - Contains a text editor tab that has a line display and console output, with options to create new, load, save, assemble, and disassemble.
- **Unit Tests**
  - Over 90% Unit Test coverage enables easier open-source development.