Super Cool Disassembler That Does Cool Things

1.1 Problem Description

Malware, or harmful and malicious software, is a major problem in the modern era. According to a PBS article about ransomware, a type of malware that demands money from the user for the return of their data, attacks rose by 158% in North America[1] in one year. To learn more about malware and how it attacks a device, it must be reverse engineered from binary into a programming language that can be understood. To successfully combat malware attacks, many reverse engineers will use a disassembler.

Other applications of a disassembler are that it can be used to modify existing programs, or to combine features of two or more programs. These have practical applications outside of malware and are useful to programmers because binary files are harder to read and typically have hardware values encoded, while assembly files have generic values and could be run on multiple machines.

Our disassembler will be a program that takes a binary file and translates it into a MIPS assembly file. It is our expectation that the produced assembly file will have a small margin of error when compared to the original file. The disassembler will be available via a website where you can upload a text file with the binary and it will output the MIPS assembly.

1.2 Methodology

Specific Tasks:

- Research about MIPS Instructions Set/Architecture
- Develop algorithm to determine instruction type and translate machine language to assembly
- Research Web Development
- Create Deliverables
    - Program Files (stored in code repository: Github)
    - Presentation/Demo
    - Final Report

Research

- Learn more about MIPS Instruction Set
    - Encoding and Decoding (disassembly) of Instructions

---

1 https://www.pbs.org/newshour/nation/why-ransomware-attacks-are-on-the-rise-and-what-can-be-done-to-stop-them

        - Floating Point Registers and Instructions
        - Pseudoinstructions: how they are implemented
    - Learn more about other available disassemblers:
        - ODA Disassembler

Tools

- MIPS simulator: to create working assembly code and compare with disassembler output, so that we are able to:
    - Debug the disassembler output
    - Analyze the discrepancies between translated code
        - Translations between languages (back and forth) will have them
- MIPS Converter: inputting an assembly instruction to observe the machine language output, so that we are able to:
    - Debug disassembler output
    - Confirm the accuracy of the disassembler output
- Github: code repository

1.3 Deliverables

- A website that translates the machine language into assembly code.
    - This is for making what the machine is doing more human-readable.
        - From .txt input file to assembly language
    - This file (or these files) will be put in a github repository
- A Powerpoint
    - This powerpoint will be run through during our final presentation in class that describes what our problem was, our solution, and how we went about it
    - This powerpoint will most likely include some sort of example/demonstration to show the input and the resulting output (what our assembler does will be explained during the presentation while displaying the example)
- A Final Report
    - A report similar to this one that outlines our problem, the finalized methods we used to find our solution, and the final results of our project

1.4 Team

Project Proposal Assignments

Emily: Problem Description
Haley: Deliverables
SJ: Methodology (Algorithm/Pseudocode/Directions) & Team
Saide: Methodology (Algorithm & Tools)

Project Deliverable Assignments

Frontend

Lead: Saide
Supporting: SJ
Description: Dealing with the interface/flow of the webpage, sending the input to the "brain" of the website, and displaying the output.

Logic

Lead: SJ
Supporting: Saide
Description**:** Converting the ASCII 0s and 1s to MIPS instructions.

Preparation

Lead: Emily
Supporting: Haley
Description: Converting the object file to a human readable format/file.

Testing
Lead: Haley
Supporting: Emily
Description: Testing extraneous output/input cases and testing.

Final Report

Report Lead: Haley
Supporting: SJ, Saide, Emily

Code & Documentation

Lead: SJ
Supporting: Saide, Emily, Haley
Description: Demonstration with screenshots and a short explanation of the disassembler's purpose, interface, and how the components communicate with each other.