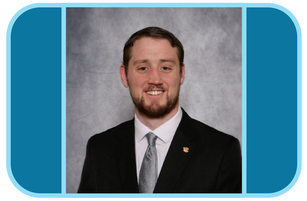
HACKADEMIA

Chloe Belleti



Preston Buterbaugh

Project Advisor

Dr. Giovani Abuaitah

Madilyn Coulson

Isabella Hall

Background

- **Project for Dr. Giovani Abuaitah (Computer Science professor)**: Focuses on teaching assembly code by visualizing how C code translates to assembly and affects the CPU.
- **Continuation of previous x86 assembly emulator**: The existing tool statically displayed C code, its assembly translation, and allowed step-by-step execution to show stack and register updates.
- **Goal**: Enable users to write custom C code, compile it to assembly in real time, and dynamically interpret assembly instructions to update registers and the stack accordingly.

Technologies

- **Client-server model**: User interacts with a web app that sends C code to the server via HTTP.
- **Compilation process**: Server uses GCC and Linux commands to compile C code into binary machine code, returning it in an HTTP response.
- **Disassembly**: Client uses the Iced x86 library to convert machine code into assembly.
- **Tech stack**:
  - **Frontend**: React (TypeScript)
  - **Backend**: Node.js

Obstacles & Challenges

- **C-to-Assembly conversion**: Gaining a deeper understanding of the translation process.
- **Low-level programming**: Re-familiarizing ourselves with assembly and system-level concepts.
- **Feature dependencies**: Planning ahead to manage interdependent functionalities.
- **Disassembler choice**: Deciding between building a custom disassembler or using an open-source library.
- **Hex extraction**: Extracting machine code from C object files.
- **Frontend design**: Developing new styled components for the module.

Achievements

- **C code input**: Users enter C code, including a main function and helper functions, in the UI.
- **Compilation**: The server compiles the C code into a hex string.
- **Disassembly**: The client receives the hex string and uses ICED to convert it into assembly language.
- **Display**: The assembly code is shown in the UI.
- **Stack & register visualization**: Initial implementation of correspondence and animations has begun.

Future Work

- **Academic impact**: Provides students and professors with an interactive way to visualize and teach how C code transforms into assembly.
- **Broad usability**: Beneficial for anyone looking to understand and practice C-to-Assembly conversion.
- **Future expansion**: Can be extended to support larger, more complex programs.
- **Bridging knowledge gaps**: Helps students grasp low-level programming concepts, stack operations, and register management.
- **Inspiring interest**: Encourages students to explore computer science through an engaging, hands-on tool.
- **Potential model**: Could inspire future developer tools for understanding language translation.