SIMULATOR REPORT

Introduction

The provided C++ assembler code is designed as a RISC assembly to machine code converter, forming the backend for an anticipated web-based simulator. This report outlines both the successes and challenges encountered during the development, with a focus on improvements made to label and register conversions. It also acknowledges the groundwork laid for the web interface, although specific components have not been added.

Successes

1. Comprehensive Instruction Set:

The assembler code successfully supports a comprehensive set of RISC assembly instructions, covering arithmetic operations, memory access, control flow, and data directives. The mapping of assembly instructions to machine code is well-defined, facilitating the accurate translation of assembly programs.

2. Register Mapping:

The initial implementation provides an extensive mapping of general-purpose registers, including floating-point (f) registers. The register-to-binary mapping ensures that each register is correctly represented in machine code, supporting a wide range of register-based instructions.

3. Scalable Symbol Table:

The code incorporates a symbol table to manage labels and their corresponding addresses during the conversion process. The symbol table allows for the proper referencing of labels within the assembly code, facilitating accurate conversion to machine code.

Handling Errors and Resolutions

1. Label Handling:

The original code encountered challenges in accurately converting labels to machine code. The revised code now identifies labels with a trailing ':' character and calculates their addresses based on the generated machine code. This enhancement ensures correct referencing and conversion of labels during instruction processing.

2. Register Handling:

Issues related to the conversion of registers, specifically excluding certain general-purpose registers, were identified. The revised register-to-binary mapping now accurately represents the intended set of registers, addressing the initial inaccuracies and providing a correct mapping for machine code generation.

Web Interface Integration

While the current version focuses on the backend logic for assembly to machine code conversion, an exciting aspect is the envisioned web interface for a simulator. The code lays the foundation for a user-friendly web-based simulator, and the architecture is in place to seamlessly integrate frontend components, enhancing the overall user experience.

Conclusion

The assembler code has successfully achieved its primary objectives, demonstrating competence in handling a wide range of RISC assembly instructions and providing accurate register mappings. The improvements made to label and register conversions enhance the reliability of the code. Furthermore, the groundwork for a web interface indicates a forward-looking approach, anticipating future enhancements to deliver a comprehensive RISC assembly simulator. Users are encouraged to explore the code and anticipate the evolution of the web-based interface in subsequent updates.