Vicharak Internship Task: Writing a Simple Compiler for an 8-bit CPU

Objective: The objective is to create a simple compiler that translates a high-level language into assembly code for an 8-bit CPU. This task will help you understand the basics of compiler construction and the architecture of an 8-bit CPU.

Prerequisites:

- Basic understanding of computer architecture, especially 8-bit CPUs.
- Proficiency in C or C++ programming.
- Familiarity with assembly language and basic compiler concepts.

Task List:

1. **Setup the 8-bit CPU**

Simulator:
- Clone the 8-bit CPU repository.
- Read through the README.md to understand the CPU architecture and its instruction set.
- Run the provided examples to see how the CPU executes assembly code.

2. **Understand the 8-bit CPU Architecture:**
- Review the Verilog code in the rtl/ directory, focusing on key files such as machine.v.
- Identify the CPU's instruction set, including data transfer, arithmetic, logical, branching, machine control, I/O, and stack operations.

3. **Design a Simple High-Level Language:**
- Define a minimal high-level language with basic constructs like variables, arithmetic operations, conditionals, and loops.
- Example language constructs:
- Variables: int a;
- Arithmetic: a = b + c;
- Conditionals: if (a == b) {...}
- Loops: while (a < b) {...}

4. **Create a Lexer:**
- Write a lexer in C/C++ to tokenize the high-level language code.
- The lexer should recognize keywords, operators, identifiers, and literals.

5. **Develop a Parser:**
- Implement a parser to generate an Abstract Syntax Tree (AST) from the tokens.
- Ensure the parser handles syntax errors gracefully.

6. **Generate Assembly Code:**
• Traverse the AST to generate the corresponding assembly code for the 8-bit CPU.
• Map high-level constructs to the CPU's instruction set. For example, arithmetic operations map to add, sub, etc.

7. **Integrate and Test:**
• Integrate the lexer, parser, and code generator into a single compiler program.
• Test the compiler with simple high-level language programs and verify the generated assembly code by running it on the 8-bit CPU simulator.

8. **Documentation and Presentation:**
• Document the design and implementation of the compiler.
• Prepare a presentation to demonstrate the working of the compiler and explain the design choices.

Resources • 8-bit Computer Repository (https://github.com/lightcode/8bit-computer)This task list will guide you through the process of understanding the 8-bit CPU and writing a simple compiler for it, providing a practical experience with compiler construction and computer architecture.