Assembler Design Notes

CSCI 6461 Team 10 Project

February 10, 2025

1 Overview

This assembler translates assembly language instructions into machine-readable octal code following the instruction set architecture (ISA) specifications. It employs a two-pass approach to handle labels, symbol resolution, and binary conversion accurately.

2 Architecture

The assembler is structured into three main components:

- **Opcode Table:** A predefined mapping of mnemonics to binary opcodes.
- **Symbol Table:** A dictionary storing label names and their corresponding memory addresses.
- **Two-Pass Processing:**
  - **Pass One:** Constructs the symbol table by scanning labels and memory locations.
  - **Pass Two:** Converts assembly instructions into machine code and writes them to output files.

3 Processing Logic

3.1 Pass One: Symbol Table Construction

- Reads each line of the assembly source file.
- Identifies labels and stores their memory addresses in the symbol table.
- Tracks the location counter for each instruction.

3.2 Pass Two: Instruction Translation and Output

- Reads the assembly file again.
- Converts mnemonics to binary using the opcode table.
- Resolves label references using the symbol table.
- Formats the binary instruction and converts it to octal.
- Writes the results to `listing.txt` and `load.txt`.

4 Key Functions and Responsibilities

- **passOne():** Scans the source file and builds the symbol table.
- **passTwo():** Processes instructions, resolves labels, and generates output files.
- **parseOperands():** Converts instruction operands into binary format.
- **binaryToOctal():** Converts 16-bit binary values to octal representation.

5 Error Handling and Edge Cases

- Undefined labels trigger an error in `passTwo()`.
- Extra or missing operands are flagged as errors.
- All instructions are formatted to ensure an exact 16-bit representation before conversion.

6 Updates in the Latest Version

- Expanded opcode support, including arithmetic and logical operations.
- Improved operand parsing to correctly handle indirect addressing and index registers.
- Fixed issues related to octal conversion and memory location tracking.
- Adjusted instruction formatting to match ISA specifications.

7 Conclusion

This document provides an overview of the assembler's design, core logic, and processing workflow. The two-pass structure ensures correct translation and label resolution, making the assembler reliable for ISA-based execution.