Documentation for GUITwoPass Java Project

A two-pass assembler is a program that translates assembly language into machine code in two distinct stages or "passes."

- Pass One: In this phase, the assembler reads the source code to create a symbol table, which maps labels to their corresponding memory addresses. It also determines the size of the program and generates intermediate code, but it does not produce machine code yet.

- Pass Two: During this phase, the assembler reads the intermediate code generated in Pass One. It uses the symbol table to resolve labels and generate the final machine code output, which can be directly executed by the computer.

Key Features

- Symbol Table Creation
- Intermediate Code Generation
- Address Resolution
- Machine Code Output
- Error Detection.

.

Project Overview

The GUITwoPass project is a Java-based graphical user interface application that implements a two-pass assembler. It converts assembly language source code into machine code, providing a user-friendly interface for assembly programming. The application facilitates the understanding of assembly language concepts through its interactive design, making it a valuable educational tool for students and developers alike.

Key Features

- Two-Pass Assembly Process: The application performs assembly in two passes. The first pass creates a symbol table and the second pass generates the object code.

- Graphical User Interface: Built with Java Swing, the interface is intuitive and user-friendly, allowing for easy file loading and output viewing.

- File Management: Users can load assembly input files and opcode tables using a file chooser dialog.

- Output Display: The application features multiple text areas to display intermediate code, symbol tables, program length, output, and object code dynamically.

- Error Handling: The application detects errors such as duplicate labels and issues during file loading, alerting users via dialog boxes.

- Reset Functionality: A reset button clears all fields, allowing users to start a new session without restarting the application.

Technology Stack

- Programming Language: Java

- Framework: Java Swing for GUI development

- Data Structures: Utilizes `HashMap` for managing the symbol and opcode tables

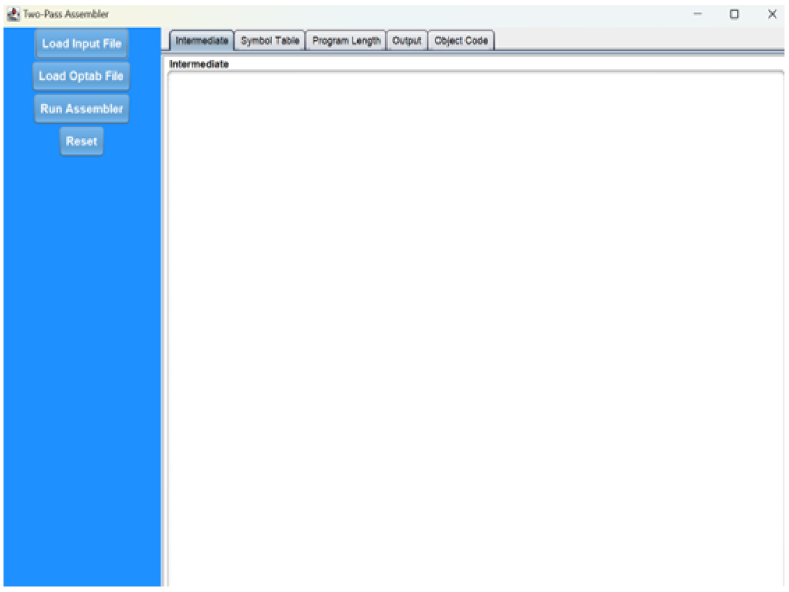
- File I/O: Java I/O classes for reading and writing files

- Development Environment: Compatible with any Java IDE (e.g., IntelliJ IDEA, Eclipse)

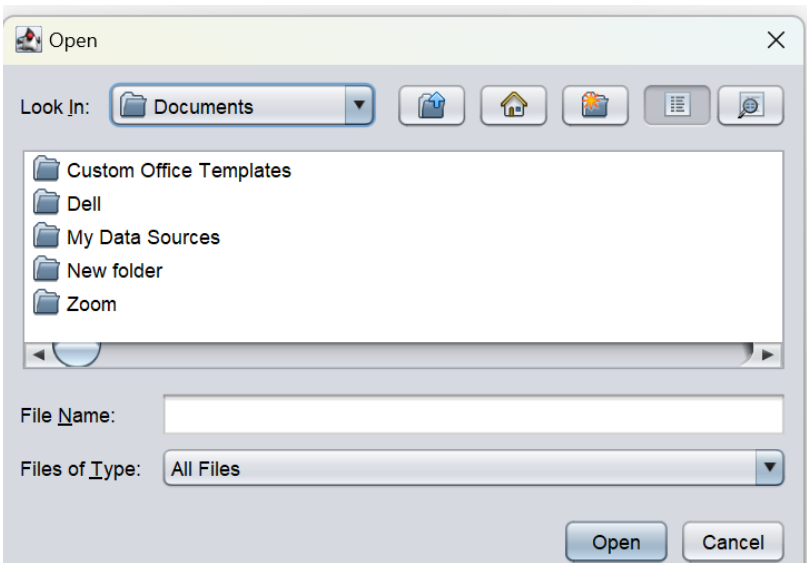
Screenshots

Insert relevant screenshots of the application here, showing different functionalities.

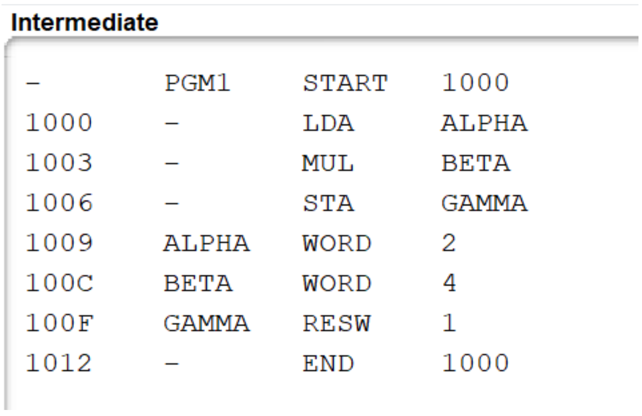
1. Main Interface



2. Load Input File

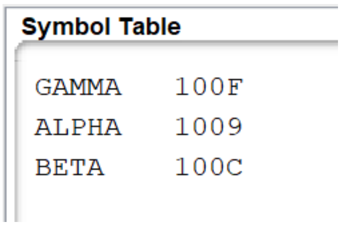


3. Output Display



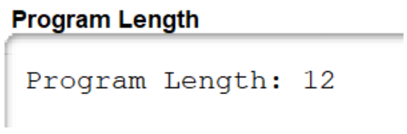
Intermediate

```
-       PGM1    START   1000
1000    -       LDA     ALPHA
1003    -       MUL     BETA
1006    -       STA     GAMMA
1009    ALPHA   WORD    2
100C    BETA    WORD    4
100F    GAMMA   RESW    1
1012    -       END     1000
```



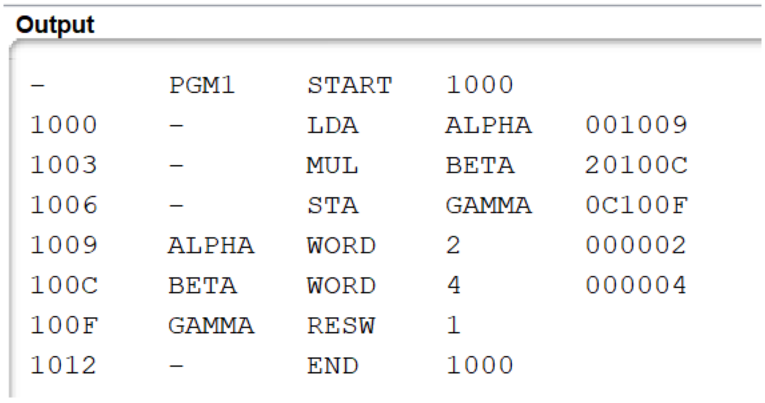
Symbol Table

```
GAMMA    100F
ALPHA    1009
BETA     100C
```



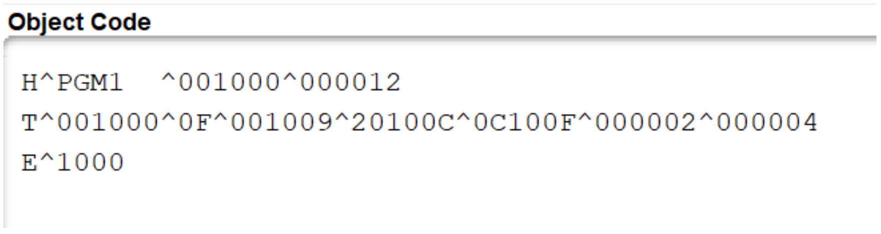
Program Length

```
Program Length: 12
```



Output

```
-       PGM1    START   1000
1000    -       LDA     ALPHA   001009
1003    -       MUL     BETA    20100C
1006    -       STA     GAMMA   0C100F
1009    ALPHA   WORD    2       000002
100C    BETA    WORD    4       000004
100F    GAMMA   RESW    1
1012    -       END     1000
```



Object Code

```
H^PGM1  ^001000^000012
T^001000^0F^001009^20100C^0C100F^000002^000004
E^1000
```

How to Use

1. Launch the Application: Compile and run the `GUITwoPass` class in your Java IDE or via the command line.

2. Load Input Files:

   - Click on Input File to choose the assembly source file.

   - Click on Load Optab File to choose the opcode table file.

3. Run the Assembler: After loading the required files, click on **Run Assembler** to begin the assembly process.

4. View Outputs: Navigate through the tabs to check:

   - Intermediate: Displays the intermediate code generated during the first pass.

   - Symbol Table: Shows the symbol table with labels and addresses.

  - Program Length: Indicates the total length of the program.

  - Output: Displays the complete output of the assembly.

  - Object Code: Shows the generated object code.

5. Reset the Application: Use the Reset button to clear all outputs and reset the application for a new assembly session.

Project Structure

```
GUITwoPass/
├── src/
│   └── GUITwoPass.java      # Main class containing the assembler logic and GUI
├── resources/
│   ├── input.asm           # Example assembly input file
│   ├── optab.txt           # Example opcode table file
└── README.md               # Documentation and project information
```

- src/: Contains the source code of the application.

- resources/: Includes sample files for testing the assembler, such as input assembly files and opcode tables.

- README.md: Provides an overview, setup instructions, and usage guidelines for the project.

Conclusion

The GUITwoPass project serves as an educational tool for those interested in assembly language and the two-pass assembly process. With its clear design and robust features, it provides an excellent platform for learning and experimentation in assembly programming.