

User Manual for Two-Pass Assembler GUI

Introduction

The **Two-Pass Assembler GUI** is a Java application designed to assemble assembly language programs into machine code. It reads a source assembly program, processes it in two passes, and generates intermediate code, symbol tables, and the final object code. This user-friendly interface allows users to load assembly files and opcode tables, view intermediate results, and generate object code in an intuitive and modern interface.

System Requirements

- **Operating System:** Windows, macOS, or Linux with a Java Runtime Environment (JRE).
 - **Java Version:** Java 8 or higher.
 - **Memory:** 512MB of RAM minimum.
 - **Files Required:**
 - Assembly input file (source code).
 - Opcode table (optab) file for the instruction set.
-

Installation and Setup

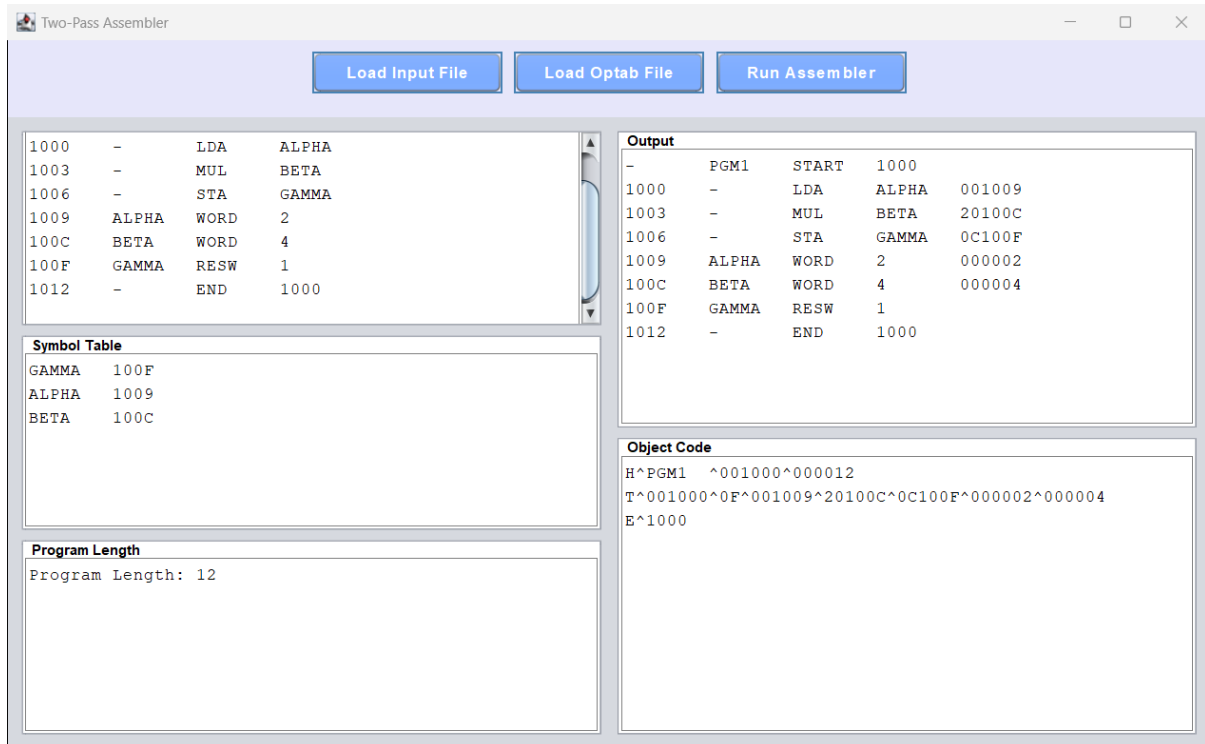
1. **Download and Install Java:**
 - If Java is not installed, download and install the latest version of the JDK (Java Development Kit)
2. **Download the Program:**
 - Ensure that you have the TwoPassAssemblerGUI.java file in your working directory.
3. **Compile the Program:**
 - Open the terminal or command prompt in the directory where the Java file is located.
 - Run the following command to compile the program:

```
javac TwoPassAssemblerGUI.java
```

4. **Run the Program:**
 - Once compiled, execute the program with the following command:

```
java TwoPassAssemblerGUI
```

Application Layout



The Two-Pass Assembler GUI consists of several sections to help you interact with the assembler.

1. Main Window

- The main window contains the following components:
 - Input/Optab Buttons:** Buttons to load the input assembly program and the opcode table.
 - Run Assembler Button:** Executes the assembler, running **Pass 1** and **Pass 2**.
 - Text Areas:** Five sections display intermediate, symbol table, program length, output, and object code.

2. Text Areas:

- Intermediate Area:** Displays the intermediate code generated after **Pass 1**.
- Symbol Table (Symtab) Area:** Shows the symbol table with labels and their corresponding addresses.
- Program Length:** Displays the total length of the program in hexadecimal.
- Output Area:** Displays the assembled program, including addresses, opcodes, and labels.
- Object Code Area:** Displays the final object code, formatted for loading into a machine or simulator.

Step-by-Step Guide

1. Loading the Assembly Input File

- Click on the **Load Input File** button at the top of the window.
- A file chooser window will appear. Navigate to the folder containing your assembly source file and select the file.
- The file is now loaded into the assembler and ready for processing.

2. Loading the Opcode Table (Optab)

- Click on the **Load Optab File** button.
- A file chooser will appear. Select your optab file, which contains the opcode-instruction mappings.
- Once loaded, the assembler will use this table to generate the object code in **Pass 2**.

3. Running the Assembler

- After loading both the input file and the opcode table, click on the **Run Assembler** button.
- The assembler will perform **Pass 1** and **Pass 2** of the assembly process:
 - **Pass 1:** Generates the intermediate code and builds the symbol table.
 - **Pass 2:** Generates the final object code using the symbol and opcode tables.
- The results will be displayed in the text areas:
 - **Intermediate Area:** Shows the intermediate code, including addresses and parsed instructions.
 - **Symtab Area:** Displays the symbol table with label names and their respective addresses.
 - **Program Length:** Displays the total length of the program.
 - **Output Area:** Displays the fully assembled code with labels, addresses, and object code.
 - **Object Code Area:** Displays the final object code in a format suitable for use in loaders or simulators.

Understanding the Output

1. Intermediate File:

- Displays each instruction with its address, label, opcode, and operand.
- Example:

1000 LOOP LDA VALUE

1003 - STA RESULT

2. Symbol Table:

- Displays labels (symbol names) with their corresponding addresses.
- Example:

LOOP 1000

VALUE 1050

3. Program Length:

- Shows the total length of the program in hexadecimal.
- Example:

Program Length: 0066

4. Output:

- Shows the address, label, opcode, operand, and the corresponding machine code (object code).
- Example:

1000 LOOP LDA VALUE 032050

1003 - STA RESULT 0F2000

5. Object Code:

- Displays the final object code in standard format with headers, text records, and an end record.
- Example:

H^COPY ^001000^000045

T^001000^1E^141033^481039^000036

E^001000

Files Format

1. Input Assembly File:

The assembly file is a simple text file with one instruction per line, consisting of a label (optional), opcode, and operand. Example:

COPY START 1000

LDA ALPHA

ADD BETA

STA GAMMA

END

2. Opcode Table (Optab) File:

The opcode table file is a text file where each line consists of an opcode and its corresponding machine code, separated by spaces. Example:

r

Copy code

LDA 00

ADD 18

STA 0F

Contact and Support

If you encounter any issues, need further assistance, or would like to report a bug, please contact the developer at:

- **GitHub:** <https://github.com/Adarshsuresh11/two-pass-assembler.git>

This user manual provides detailed instructions on how to effectively use the Two-Pass Assembler GUI to assemble your assembly language programs. If you follow the steps closely, you'll be able to generate intermediate files, symbol tables, and final object code successfully.