Documentation for Two-Pass Assembler GUI in Java

This documentation provides an overview of a Java Swing-based GUI application for a two-pass assembler simulation. The code is separated into two main classes: TwoPassAssemblerGUI, which handles the graphical user interface, and TwoPassAssembler, which performs the two-pass assembly process. An entry-point class Main is also provided to launch the application.

1. Overview

This Java application simulates a two-pass assembler used in computer systems to translate assembly language code into machine code (object code). It allows users to select an input file and an opcode table file (Optab), which contains opcode mappings. The assembler performs a "two-pass" process:

- **Pass One:** Establishes addresses for all labels and creates an intermediate file.
- **Pass Two:** Generates the object code based on instructions and labels defined in Pass One.

The GUI displays the contents of the intermediate file, symbol table, and generated object code.

---

2. Class Descriptions

TwoPassAssemblerGUI

The TwoPassAssemblerGUI class extends JFrame and creates the main user interface for interacting with the assembler.

Key Features:

- File selection for input and Optab files.
- Display areas for intermediate code, symbol table, and object code.
- Button to initiate assembly.

TwoPassAssembler

The TwoPassAssembler class performs the assembly in two passes:

- **Pass One:** Calculates memory locations and fills the symbol table.
- **Pass Two:** Translates instructions into object code using information from Pass One.

Main

The Main class contains the main method, which initiates the GUI by calling SwingUtilities.invokeLater(TwoPassAssemblerGUI::new);.

---

3. Detailed Code Explanation

GUI Components (TwoPassAssemblerGUI)

The GUI layout is configured in the TwoPassAssemblerGUI class.

Attributes

- **JTextField inputFileField, optabFileField**: Text fields for file paths.
- **JTextArea intermediateArea, symtabArea, outputArea**: Text areas to display the output.
- **JButton assembleBtn**: Button to trigger the assembler.

Methods

- **TwoPassAssemblerGUI() Constructor**: Sets up the GUI window, layout, and components.
- **createMainPanel()**: Configures the main panel layout, including labels, buttons, and scrollable areas for displaying output.

Action Listeners

- **browseFile(JTextField field)**: Opens a file chooser dialog to select files for the input and Optab.
- **runAssembler()**: Initiates the assembly process in a background thread using SwingWorker.

Pass One (TwoPassAssembler)

passOne() processes each line of the assembly source code:

1. **START Directive Handling**: Checks if the source code starts with a START directive. If so, it sets the location counter (locctr) to the specified start address.
2. **Symbol Table Creation**: Adds labels to symtab with their corresponding addresses.
3. **Intermediate Code Generation**: Stores each instruction and its location counter in intermediate.

Location counter (locctr) updates based on instruction types:

- **Standard OpCodes**: Increments locctr by 3.
- **Data Storage Directives**: Updates locctr based on the directive (WORD, BYTE, RESW, RESB).

Pass Two (TwoPassAssembler)

passTwo() generates the final object code:

1. **Header Record**: Created based on the START directive.
2. **Text Records**: Created for each instruction, translating assembly instructions into object code using the optab and symtab mappings.
3. **End Record**: Marks the end of the program.

Intermediate and Symbol Table Handling

The assembler uses multiple Map objects to hold data:

- **optab (Map<String, String>)**: Stores opcode mappings from the optab file.
- **symtab (Map<String, Integer>)**: Holds symbol names and their memory addresses.
- **intermediate (Map<Integer, String>)**: Stores each line of the assembly code with its memory address for use in Pass Two.
- **objectCode (Map<Integer, String>)**: Holds the final machine code for each instruction.

Helper Methods in TwoPassAssemblerGUI

- **displayFileContent(String filename, JTextArea textArea)**: Reads file content and displays it in the specified JTextArea.

---

4. Usage and Execution

Step 1: Prepare Files

Create two text files:

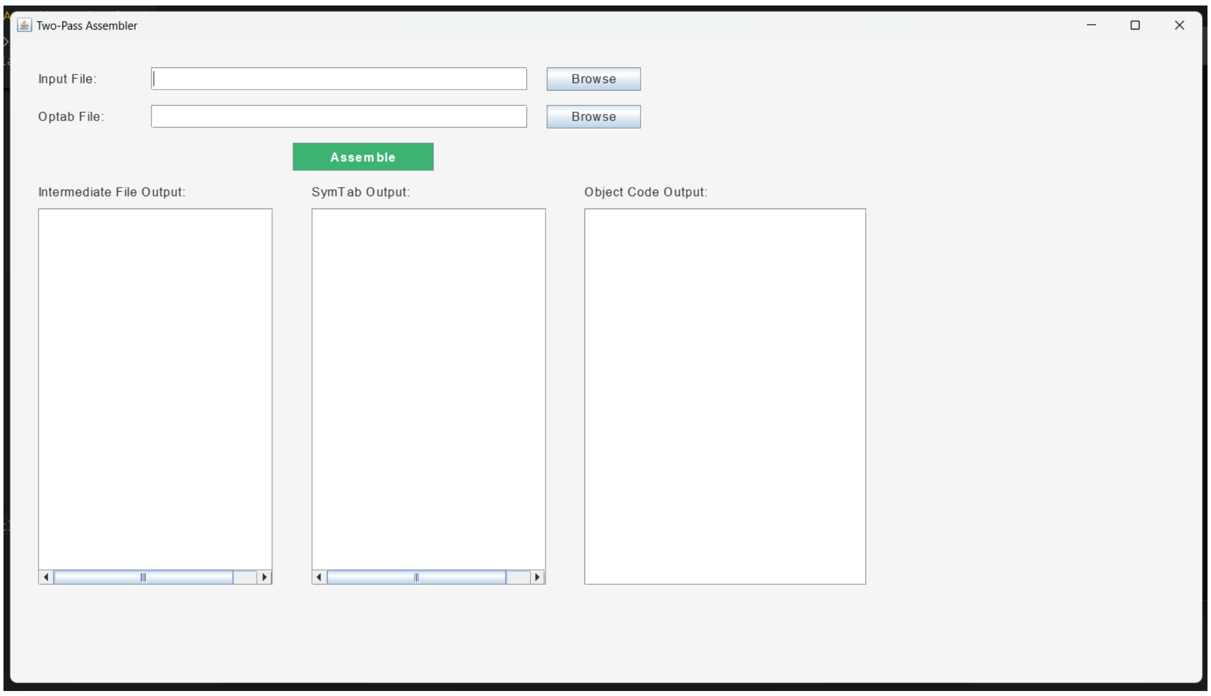
- **Input File**: Assembly source code containing instructions, labels, and directives.
- **Optab File**: Contains opcode mappings for instructions.

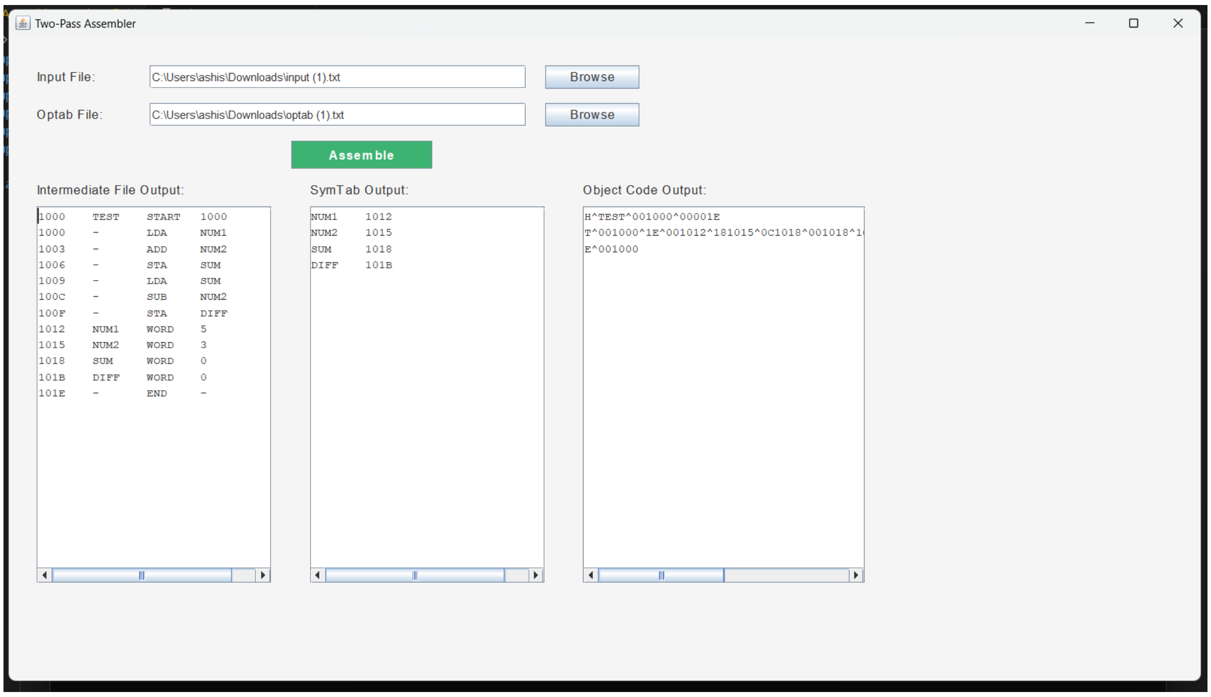
Step 2: Run the Program

- Run the Main class to open the GUI.
- Select the Input and Optab files using the "Browse" buttons.
- Click "Assemble" to initiate the two-pass assembly.

Step 3: View Results

- **Intermediate File Output**: Displays location and code for each instruction.
- **SymTab Output**: Shows symbols and their respective addresses.
- **Object Code Output**: Displays the generated machine code.





Input File: C:\Users\ashis\Downloads\input (1).txt

Optab File: C:\Users\ashis\Downloads\optab (1).txt

Intermediate File Output:

```
1000    TEST    START   1000
1000    -       LDA     NUM1
1003    -       ADD     NUM2
1006    -       STA     SUM
1009    -       LDA     SUM
100C    -       SUB     NUM2
100F    -       STA     DIFF
1012    NUM1    WORD    5
1015    NUM2    WORD    3
1018    SUM     WORD    0
101B    DIFF    WORD    0
101E    -       END     -
```

SymTab Output:

```
NUM1    1012
NUM2    1015
SUM     1018
DIFF    101B
```

Object Code Output:

```
H^TEST^001000^00001E
T^001000^1E^001012^181015^0C1018^001018^1
E^001000
```