Two Pass Assembler

Project Report

Aiswarya Manoj

Roll no. 16

This document provides a detailed overview of the Two Pass Assembler web application, developed to convert assembly language into machine-readable code using a two-step processing approach. The purpose of this document is to guide users and developers on the features, functionality, and usage of the website.

1. Introduction

The Two Pass Assembler is a web-based tool that converts assembly code into machine code using a two-pass process. The website offers a simple interface where users can input assembly code manually or upload files from their device. It allows users to download the results and clear the workspace for new inputs.

This document provides an overview of the tool's features, architecture, and usage, along with technical details for developers.

2. Overview of the Two Pass Assembler

The assembler processes the assembly code in two distinct phases:

- **Pass 1:** The assembler scans the code, identifying and recording all symbols and their associated addresses in a symbol table.
- **Pass 2:** The assembler uses the symbol table and resolves addresses, generating final object code ready for execution.

This website-based tool also supports basic error detection, providing feedback for invalid assembly instructions or syntax errors. Users can view intermediate outputs or directly download the symbol table and object code.

3. Design and Implementation

The web application is modular and organized into distinct functional components:

- **Pass 1 Module:** This module handles the creation of the symbol table and intermediate outputs.
- **Pass 2 Module:** Using the symbol table, this module resolves addresses and generates the object code, presenting it to the user.

- **Error Handling Module:** Provides feedback on issues such as undefined symbols, syntax errors, or incorrect instructions.
- **File Handling Module:** The web application allows users to upload assembly files and download symbol tables and object code results. Data can also be cleared or reset as needed.

The application was built with **HTML**, **CSS** for layout and styling, and **JavaScript** for dynamic content and processing. LocalStorage is used to store input data temporarily, ensuring user data persists until it's manually cleared.

4. Features

The web-based Two Pass Assembler provides the following core features:

- **Symbol Table Generation:** The website displays the generated symbol table, listing symbols and their corresponding memory addresses.
- **Object Code Generation:** Users can view and download the final machine-readable object code.
- **Error Detection:** The tool includes built-in validation and displays relevant error messages when assembly instructions are incorrect.
- **User-friendly Interface:** Designed to be simple, intuitive, and responsive for a smooth user experience.
- **File Upload and Download:** Users can upload assembly files and download both the symbol table and object code files.
- **LocalStorage Integration:** Data is stored locally in the browser during a session, preserving the input and output until manually cleared.
- **Clear Functionality:** Users can easily reset the workspace, clearing both the input and the generated output files.

5. User Interface

The user interface of the Two Pass Assembler website is minimalistic and divided into the following sections:

- **Header Section:** Displays the title "Two Pass Assembler" with a link to the GitHub repository or project documentation.
- **Input Section:** Users can either manually input their assembly code into a large text area or load a .txt file. The input area supports multiline code, with the option to load large programs. Basic syntax highlighting may be applied for better readability.
- **Button Section:**

  - **Run:** Starts the two-pass assembly process. The symbol table and object code will be generated and displayed to the user.
  - **Clear:** Clears the text input and resets the workspace to start a fresh session.
  - **Download Results:** Allows users to download the generated symbol table and object code as separate .txt files for further use.

All files, including symbol tables and object code, can be downloaded in .txt format directly from the website.

6. Requirements

User Requirements:

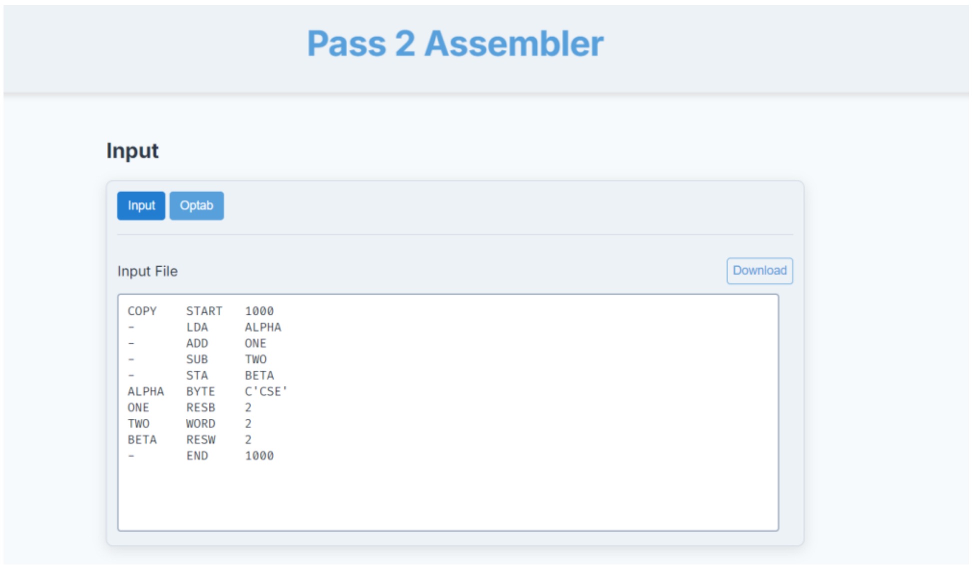
These requirements outline what the users can expect from the Two Pass Assembler:

- **Platform:** The web app runs on modern web browsers (Chrome, Firefox, Safari, etc.) on any device—desktop, tablet, or mobile.
- **Input:** Users can manually input assembly code or upload files from their device in .txt format.
- **Local Data Storage:** The website saves input and output in LocalStorage, meaning data is preserved until it is manually cleared by the user.

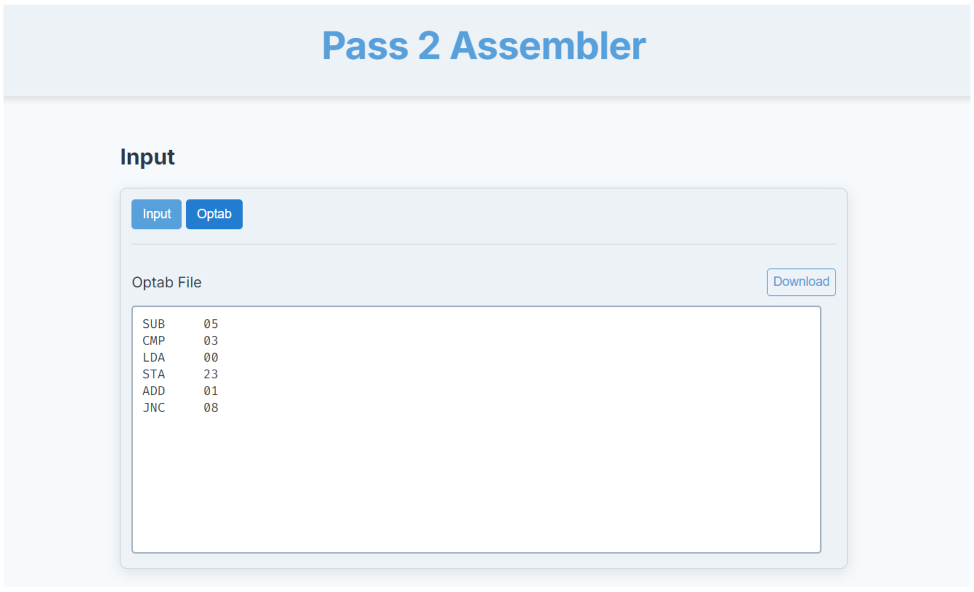
Developer Requirements:

These requirements outline the developer needs for setting up or modifying the web app:

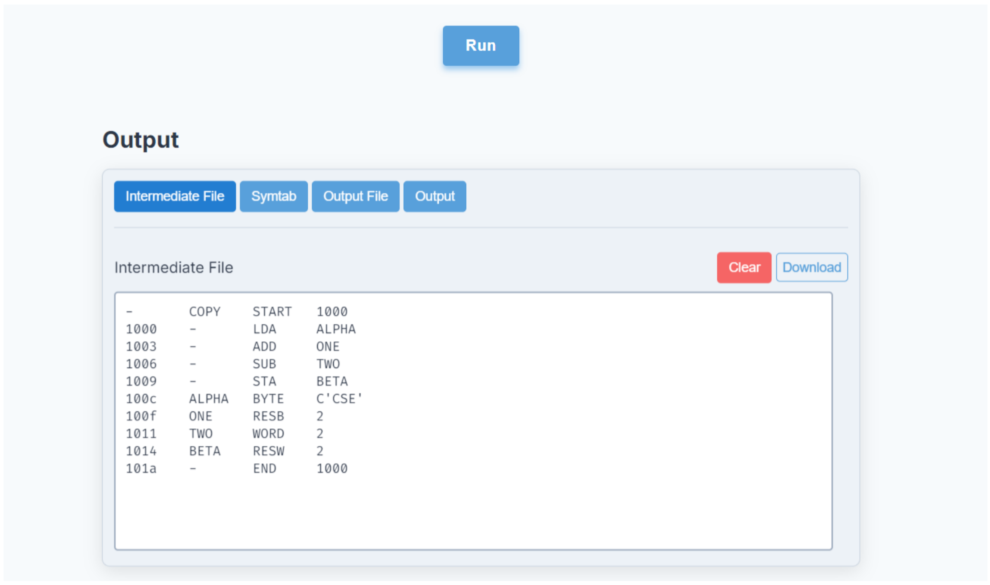
- **Languages Used:**
  - HTML: For structuring the content.
  - CSS: For layout and visual styling.
  - JavaScript: For the core logic of the assembler and file handling.
- **Development Environment:**
  - A code editor (VS Code, Sublime Text, etc.) is recommended for development.
  - A local web server or browser-based testing setup.
- **Browser Support:** Ensure compatibility with the latest versions of major browsers like Chrome, Firefox, and Safari.
- **LocalStorage:** Used to persist input and output data temporarily during a session.
- **Error Handling:** JavaScript should include sufficient checks to ensure valid assembly instructions and provide useful error messages when errors occur.



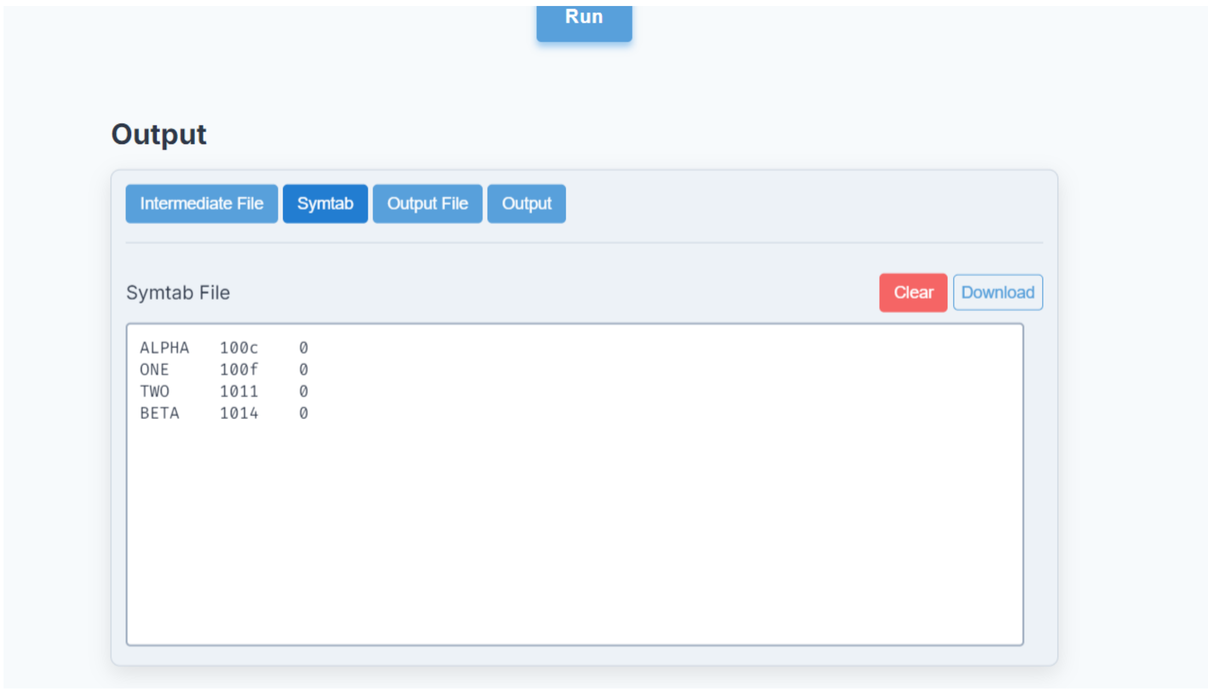
```
COPY    START   1000
-       LDA     ALPHA
-       ADD     ONE
-       SUB     TWO
-       STA     BETA
ALPHA   BYTE    C'CSE'
ONE     RESB    2
TWO     WORD    2
BETA    RESW    2
-       END     1000
```



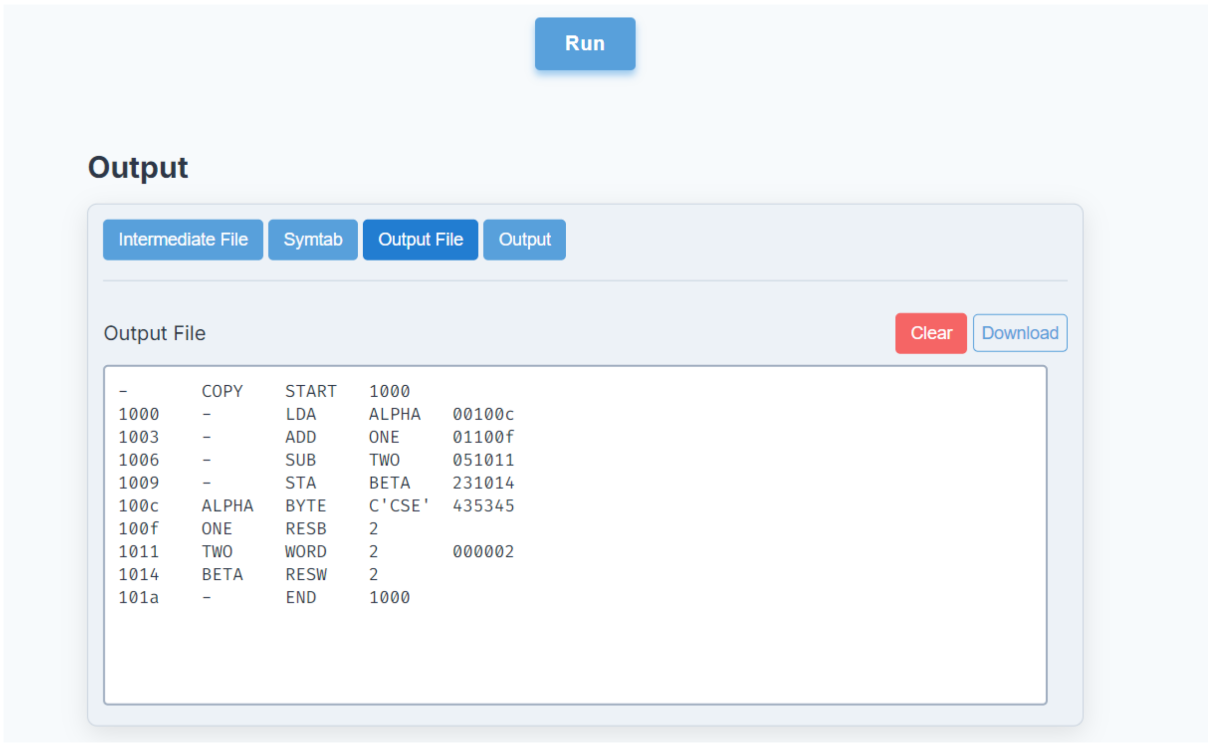
```
SUB     05
CMP     03
LDA     00
STA     23
ADD     01
JNC     08
```



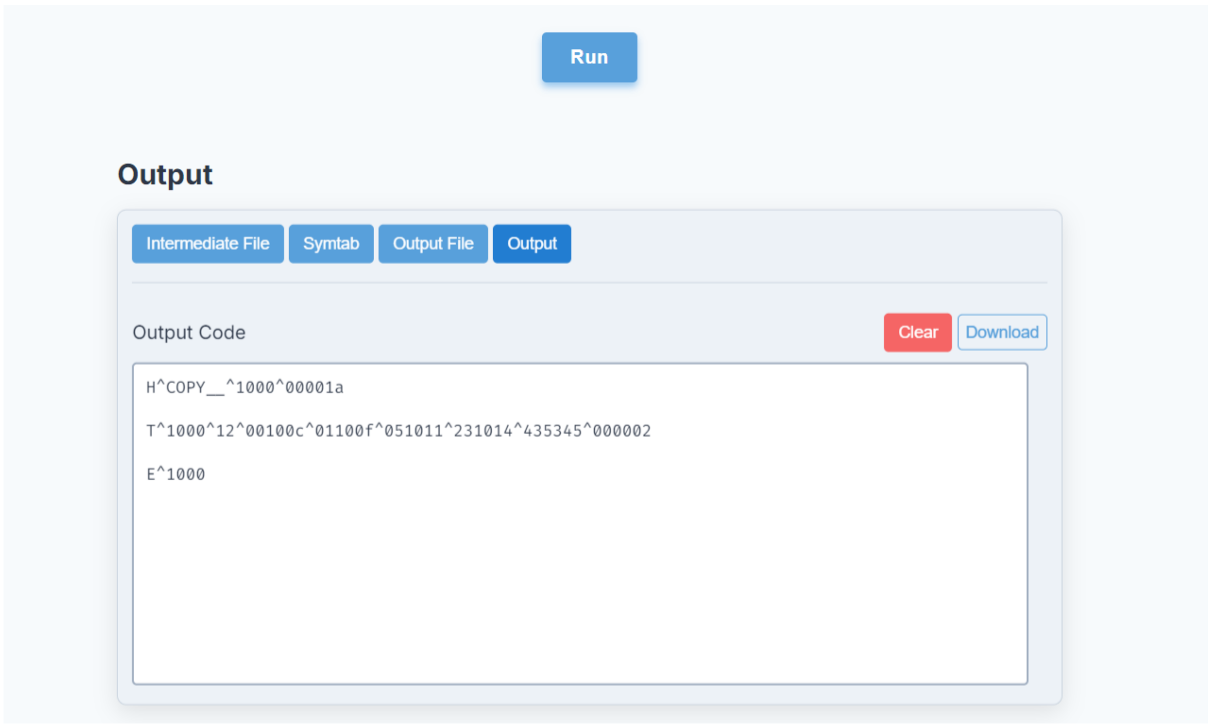
```
-       COPY    START   1000
1000    -       LDA     ALPHA
1003    -       ADD     ONE
1006    -       SUB     TWO
1009    -       STA     BETA
100c    ALPHA   BYTE    C'CSE'
100f    ONE     RESB    2
1011    TWO     WORD    2
1014    BETA    RESW    2
101a    -       END     1000
```



```
ALPHA   100c    0
ONE     100f    0
TWO     1011    0
BETA    1014    0
```



```
-       COPY    START   1000
1000    -       LDA     ALPHA   00100c
1003    -       ADD     ONE     01100f
1006    -       SUB     TWO     051011
1009    -       STA     BETA    231014
100c    ALPHA   BYTE    C'CSE'  435345
100f    ONE     RESB    2
1011    TWO     WORD    2       000002
1014    BETA    RESW    2
101a    -       END     1000
```



```
H^COPY__^1000^00001a

T^1000^12^00100c^01100f^051011^231014^435345^000002

E^1000
```

7. Project Outcome

The **Two Pass Assembler** web application effectively implements the two-pass assembly process for converting assembly language into machine-readable code. The tool is user-friendly, supports file uploading and downloading, and provides accurate assembly functionality, including symbol management, error detection, and object code generation. This project offers a versatile and accessible solution for users seeking to work with assembly language in a browser environment.

To download the app, visit GitHub Repository for access to the source code and installation instructions.