OptiSIC

Web-based SIC Assembler (Pass 1 & Pass 2) - Tutorial

1. Introduction

This tutorial covers the use of a web-based **SIC assembler** designed to handle both **Pass 1** and **Pass 2** of the assembly process for the **SIC** (Simplified Instructional Computer) architecture. The assembler is implemented using **HTML**, **Tailwind CSS**, and **JavaScript**, providing a simple and interactive interface for users to upload files, assemble the code, and view the output directly on the webpage. The project is hosted on **Vercel**, a cloud platform that enables fast, automated deployments and ensures optimized performance for users accessing the site globally.

2. Features of the Assembler

2.1 Two-Pass Assembly Process

- **Pass 1**: The assembler reads the source code and generates a **symbol table** that maps labels (e.g., ALPHA, ONE) to their corresponding memory addresses. It also creates an **intermediate file** with the parsed instructions, leaving placeholders for unresolved addresses or values.
- **Pass 2**: The assembler uses the symbol table and OP tab (operation codes) to fill in the placeholders in the intermediate file, resolving addresses and generating the **final object code**.

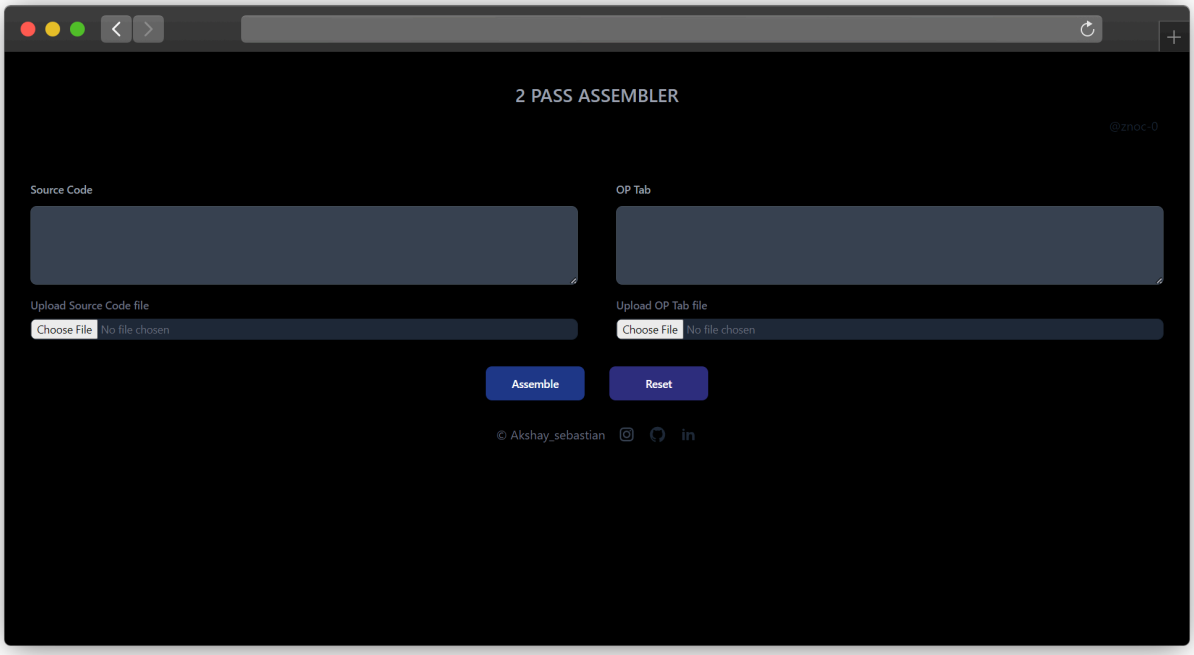
2.2 Key Functionalities

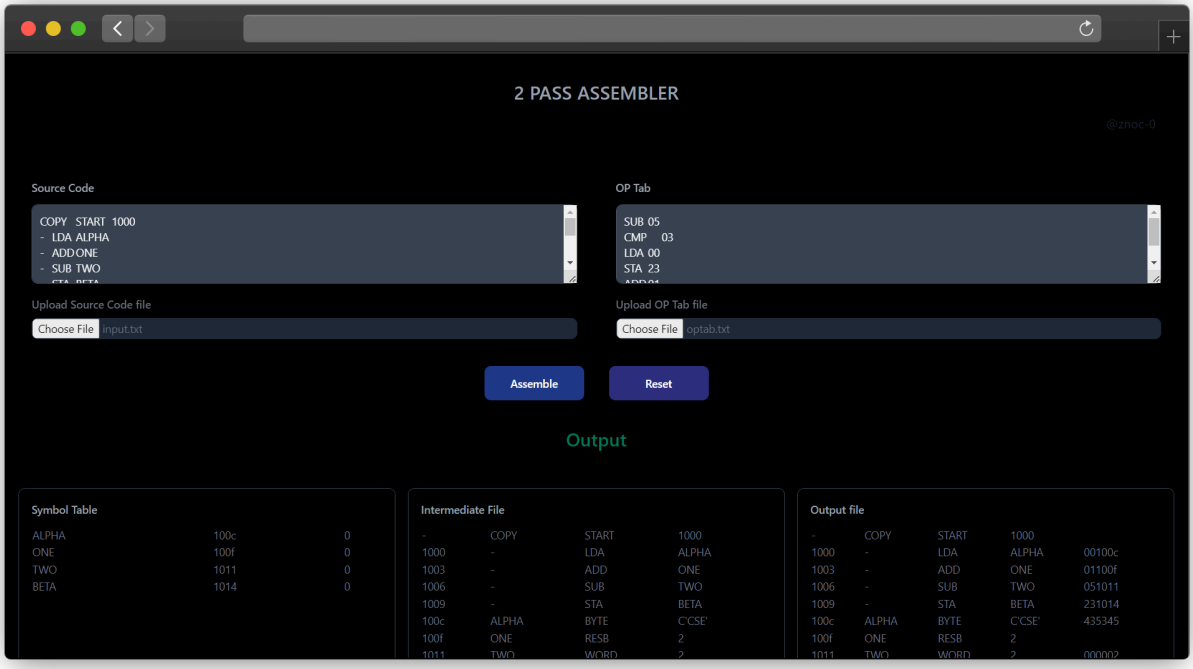
- **Source Code Upload**: Users can upload their SIC assembly source code through a simple file upload system.
- **OP Tab Upload**: An operation (OP) table is uploaded alongside the source code, mapping mnemonic instructions (e.g., LDA, STA) to machine codes.
- **Output Generation**: The assembler generates three outputs: the **symbol table**, **intermediate file**, and **object code**.
- **User-Friendly Interface**: The assembler features a straightforward interface for interacting with the system, offering buttons to assemble the code and reset the inputs.

3. Tech Stack

- **HTML**: Used for structuring the web interface, defining the layout of file uploads, buttons, and output displays.
- **Tailwind CSS**: Employed for styling and creating a responsive design, ensuring the assembler interface is visually appealing and user-friendly.
- **JavaScript**: Handles the assembly logic, processes file uploads, and generates outputs like the symbol table, intermediate file, and object code.
- **Vercel**: A cloud platform used to deploy and host the SIC assembler, enabling fast, automated deployments and optimized global performance.

4. ScreenShots.





5. How to Use the Assembler

[Go to site]

Step 1: Upload Source Code and OP Tab Files

- **Source Code**: Upload the source code file (e.g., `input.txt`) containing the assembly instructions and data definitions.
- **OP Tab**: Upload the operation code table file (e.g., `optab.txt`), mapping instructions to

their corresponding opcode.

Example Source Code:

```
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
```

Example OP Tab:

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

Step 2: Click 'Assemble'

Once both files are uploaded, click the "Assemble" button to begin the assembly process. The assembler processes both files in two passes:

- **Pass 1**: Generates the **symbol table** and **intermediate file** with placeholders for any unresolved addresses or labels.
- **Pass 2**: Resolves addresses and labels, producing the **final object code**.

Step 3: View the Output

The output is displayed in three sections:

- **Symbol Table**: Lists the labels (e.g., ALPHA, ONE) and their memory addresses.
- **Intermediate File**: Displays the processed assembly code with corresponding addresses and placeholders for unresolved labels.
- **Object Code**: Displays the final machine code that can be executed by the SIC machine.

Step 4: Reset (Optional)

If you wish to upload new files or start over, click the "Reset" button to clear the current files and output.

6.Requirements

6.1. Developer Requirements

- Basic knowledge of **HTML**, **CSS**, and **JavaScript**.
- Familiarity with web development frameworks and tools, particularly **Tailwind CSS** for styling.
- Understanding of assembly language concepts, particularly the SIC architecture and its assembly process.
- Access to a code editor and a modern web browser for testing and debugging.

6.2. General User Requirements

- A modern web browser (e.g., Chrome, Firefox, Safari) to access the assembler.
- Basic familiarity with assembly language concepts for effective use of the tool.
- Ability to upload source code and OP tab files in the required format.
- A stable internet connection for using the web-based application hosted on **Vercel**.

7. Conclusion

This web-based SIC assembler provides an intuitive platform for performing two-pass assembly for SIC architecture. It simplifies the process of converting assembly language to machine code, enabling users to upload files, process them, and view the output with ease.

Akshay_Sebastian