TWO PASS ASSEMBLER GUI

- AKHIL S NAMBIAR
CSE-A
ROLL NO:20

Overview

The Two Pass Assembler is a web-based application that facilitates the conversion of assembly language code into machine code using a two-pass assembly process. This tool helps users understand the workings of assemblers, including the generation of intermediate code, symbol tables, and object code. Designed primarily for educational purposes, it provides a hands-on experience for students and educators in computer science.

Functionality

Overview of Operations

1. **Input Handling**: Users enter assembly code and opcode definitions.
2. **Pass Execution**: Users trigger Pass 1 and Pass 2, which process the input to generate outputs.
3. **Output Display**: The generated intermediate code, symbol table, and object code are displayed in designated text areas.

Features

- **Dynamic Button Activation**: Buttons for running passes are enabled only when both inputs (assembly code and opcode table) are provided.
- **Intermediate Code Generation**: Displays a detailed breakdown of the assembly code and its corresponding location counter (locctr).
- **Symbol Table Creation**: Automatically generates a symbol table showing labels and their associated addresses.
- **Object Code Output**: Produces the final machine code output, formatted with headers and text records, ready for further processing or execution.
- **User-Friendly Interface**: Simplifies interaction through clear input areas, buttons, and output displays.

Code Structure

1. **HTML Structure**: The HTML document includes:
   - A `<head>` section for metadata and styles.
   - A `<body>` section containing the main interface elements.

2. **CSS Styles**: Internal styles define the appearance of the interface, including layout, colors, and button styles.
3. **JavaScript Logic**:
    - Event listeners for user interactions (input changes and button clicks).
    - Functions for processing assembly code and generating outputs during both passes.
    - Utilization of arrays and objects to store opcode mappings, symbol tables, and code outputs.

Extensibility

- The code can be modified to support additional assembly language features, such as more complex opcode handling or advanced error reporting.
- Additional validation can be implemented to enhance user input handling and feedback.

Testing and Debugging

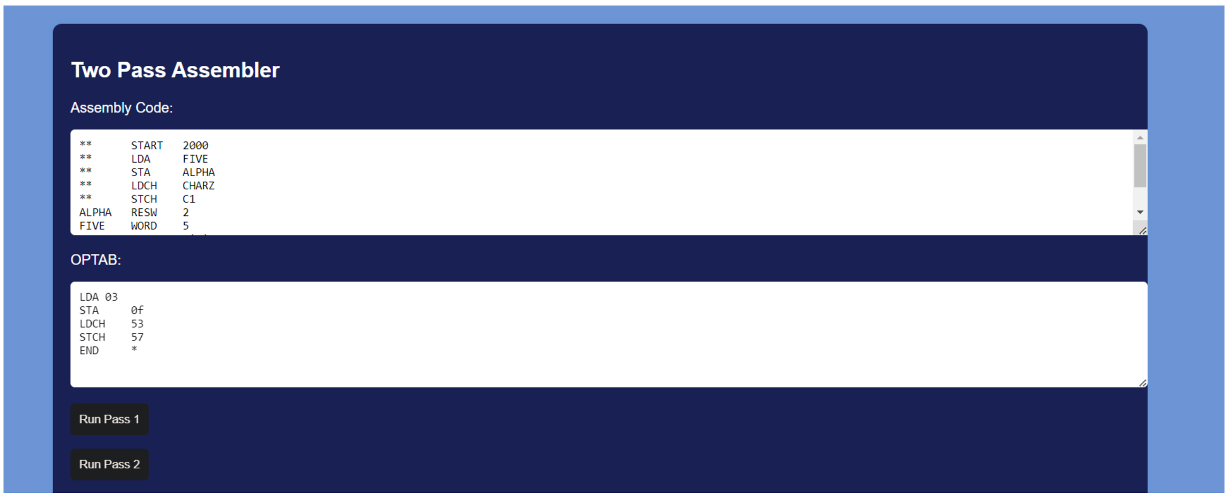
- Test in various web browsers to ensure compatibility.
- Use browser developer tools for debugging JavaScript functionality and fixing any UI issues.

User Interface

Layout

The user interface is designed to be intuitive and straightforward, featuring the following components:

1. **Container**: The main area containing all input and output elements, styled for a clean appearance.
2. **Assembly Code Input**:
    - **Label**: "Assembly Code:"
    - **Textarea**: A large text area where users can input their assembly code.
3. **OPTAB Input**:
    - **Label**: "OPTAB:"
    - **Textarea**: A text area for entering opcode definitions.



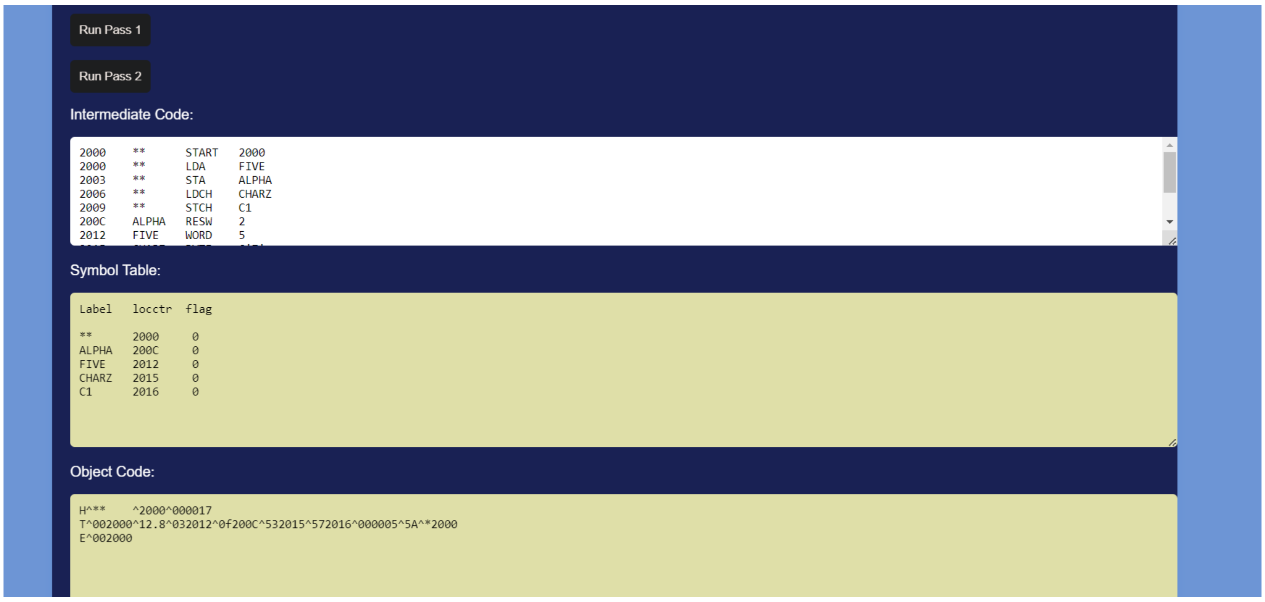
```
**      START   2000
**      LDA     FIVE
**      STA     ALPHA
**      LDCH    CHARZ
**      STCH    C1
ALPHA   RESW    2
FIVE    WORD    5
```

```
LDA 03
STA     0f
LDCH    53
STCH    57
END     *
```

4. **Action Buttons**:
   - **Run Pass 1 Button**: Executes the first pass of the assembler. Initially disabled until both input fields are filled.
   - **Run Pass 2 Button**: Executes the second pass of the assembler. Initially disabled until Pass 1 is executed successfully.
5. **Output Sections**:
   - **Intermediate Code**: A read-only text area displaying the intermediate code generated in Pass 1.
   - **Symbol Table**: A read-only text area showing the symbols and their addresses.
   - **Object Code**: A read-only text area displaying the final object code generated in Pass 2.

Visual Design

- **Color Scheme**: The application uses a blue background with white text and a dark-themed container for contrast.
- **Button Styles**: Buttons change color on hover for improved interactivity.
- **Readability**: Text areas have a consistent size and padding for easy reading



Intermediate Code:

```
2000    **      START   2000
2000    **      LDA     FIVE
2003    **      STA     ALPHA
2006    **      LDCH    CHARZ
2009    **      STCH    C1
200C    ALPHA   RESW    2
2012    FIVE    WORD    5
```

Symbol Table:

```
Label   locctr  flag

**      2000     0
ALPHA   200C     0
FIVE    2012     0
CHARZ   2015     0
C1      2016     0
```

Object Code:

```
H^**    ^2000^000017
T^002000^12.8^032012^0f200C^532015^572016^000005^5A^*2000
E^002000
```

Requirements

User Requirements

• Platform: The application should run in modern web browsers (Chrome, Firefox, Edge).

• Input: Users must be able to input assembly code and optab directly

Developer Requirements

• Programming Languages: HTML, CSS, and JavaScript

• Development Environment: Any text editor or IDE for web development (e.g., Visual Studio Code)

Access the Application

The web application can be accessed via https://github.com/akhilsn03/my-webpage

https://my-webpage-ivet.vercel.app/

Conclusion

This Two Pass Assembler provides an educational tool for learning about assembly language and assemblers. By following the user instructions, anyone can input assembly code and generate object code while developers can extend its functionality as needed.