USER MANUAL

By : Aravind R Nair

Roll No : 49

Assembler Application Documentation

Overview

The Assembler Application is a web-based tool built with Django that simulates the first and second passes of an assembler. It allows users to upload input files and an operation table (OPTAB) to generate intermediate and object code based on assembly language instructions. The application provides a user-friendly interface for running passes and viewing results, including symbol tables and program lengths.

Features

- **File Upload:** Users can upload input files containing assembly instructions and a corresponding OPTAB file.
- **Pass 1 Logic:** Processes the input file to generate intermediate file content and a symbol table.
- **Pass 2 Logic:** Uses the intermediate content and symbol table to produce the final object code.
- **Clear Functionality:** Allows users to clear all session data and reset the page.
- **Error Handling:** Provides user feedback if Pass 1 is not completed before running Pass 2.

Installation

To set up the Assembler Application, follow these steps:

Prerequisites

- Python 3.x
- Django

Steps

1. **Create a Virtual Environment:**

python -m venv env

source env/bin/activate # On Windows use `env\Scripts\activate`

2. **Install Required Packages:** pip

install django

3. **Run Database Migrations:**

python manage.py migrate

4. **Run the Development Server:**

python manage.py runserver

5. **Access the Application:**

Open your web browser and navigate to http://127.0.0.1:8000/ to access the Assembler Application.

Usage

1. **Upload Files:**
    - Use the "Input File" field to upload your assembly language file.
    - Use the "OPTAB File" field to upload the operation table.
2. **Run Pass 1:**

- Click the "Run Pass 1" button to process the uploaded files. ○ The results, including the intermediate file content and symbol table, will be displayed on the page.

3. **Run Pass 2:**
    - If Pass 1 is successful, click the "Run Pass 2" button to generate the object code.
4. **Clear Data:**
    - Click the "Clear" button to reset the application, clearing all session data.
5. **Error Handling:**
    - If you attempt to run Pass 2 without completing Pass 1, an error message will prompt you to run Pass 1 first.

Code Explanation

Views

The main functionality of the application resides in the single_page_view function within the views.py file. Here's a breakdown of the logic:

- **Pass 1 Logic:**
    - Reads the uploaded input and OPTAB files. ○ Processes the assembly instructions to generate intermediate content and a symbol table. ○ Stores results in the session.
- **Pass 2 Logic:**
    - Retrieves intermediate content and symbol table from the session.
    - Generates object code based on the retrieved data.
- **Clear Functionality:**
    - Clears all session data when the "Clear" button is clicked.

HTML Template

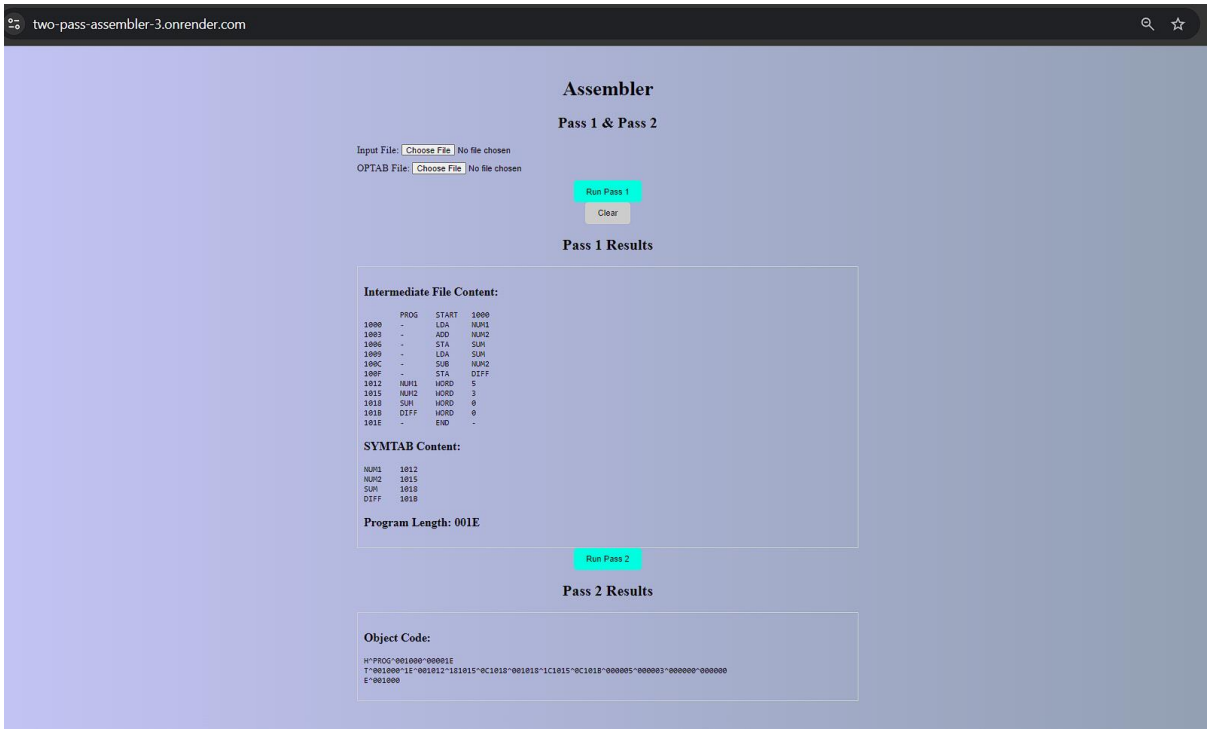
The single_page.html template provides the user interface, allowing users to upload files and view results. The key elements include:

- **File Input Fields:** For uploading input and OPTAB files.
- **Buttons:** For running passes and clearing data.
- **Conditional Rendering:** Displays results only if Pass 1 or Pass 2 has been completed.

Session Management

Session management is crucial for maintaining state across requests. The application uses Django's built-in session framework to store intermediate content, symbol tables, program lengths, and object code.

Working Diagrams



Hosting on Render

The Two Pass Assembler is hosted on Render, a cloud platform that makes it easy to deploy web applications. Render offers a range of features, including:

- Automatic Deployments: Automatically deploy your app with each push to your GitHub repository.
- Custom Domains: Set up a custom domain for your application.
- SSL Certificates: Free SSL certificates for secure HTTPS access.

Links

- **GitHub Repository**: Two Pass Assembler
- **Live Application**: Two Pass Assembler on Render

Conclusion

The Assembler Application provides an interactive platform for users to simulate assembly language processing. With its simple design and clear functionality, users can effectively experiment with assembly code compilation and understand the underlying mechanics.