

A Comprehensive Guide to the Two-Pass Assembler

Introduction

Welcome to the **Two-Pass Assembler Web Tool**, designed to streamline the process of converting assembly language into machine code using the two-pass assembly process. This website allows users to upload their assembly source file and corresponding opcode table file (optab), process them through Pass 1 and Pass 2 algorithms, and generate intermediate output, symbol table, program length and object codes.

Whether you're a student or a professional working with assembly language, this tool provides a clear, organized, and accessible interface for managing assembly code compilation.

Features of the Website

1. File Uploads:

- Allows uploading of two critical files: the assembly input source file and the opcode table (optab) file.

2. Pass One Processing:

- Executes the pass one of the two-pass assembler to generate the intermediate output, SYMTAB (symbol table), and program length.

3. Pass Two Processing:

- Executes the pass two, producing the final overall output and object code, which is the machine code for the input assembly program.

4. Downloadable Outputs:

- Users can download generated outputs like intermediate files, symbol tables, program lengths, and object codes as .txt files.

5. Clear Segregation of Pass One and Pass Two:

- The site is structured to distinctly handle Pass 1 and Pass 2 operations through individual tabs for clarity.

7. The Two-Pass Assembler Algorithms:

- There is a separate section that displays algorithm for each pass one and pass two of two pass assembler.
-

How to Use This Website

Step-by-Step Guide:

1. Visit the Home Page:

- Upon loading the website, you'll be greeted with a simple, user-friendly interface. The navbar allows you to navigate between the **Home**, **Pass One**, and **Pass Two** sections.

2. Uploading Files:

- In the "Upload Files" section, two file inputs are required:
 - **Input File:** Upload a .txt file containing the assembly code you want to process.
 - **Optab File:** Upload a .txt file containing the opcode table (optab), which maps operation codes to machine instructions.

3. Running the Assembler:

- Once both files are uploaded, click the **Run** button to initiate the two-pass assembler process.

4. Viewing Pass One Outputs:

- Navigate to the **Pass One** tab to view the intermediate outputs. The three sections within Pass One are:
 - **Intermediate Output:** Displays the intermediate file generated during Pass 1.
 - **SYMTAB Output:** Shows the symbol table with the addresses of the symbols used.
 - **Program Length:** Displays the total length of the program after Pass 1.

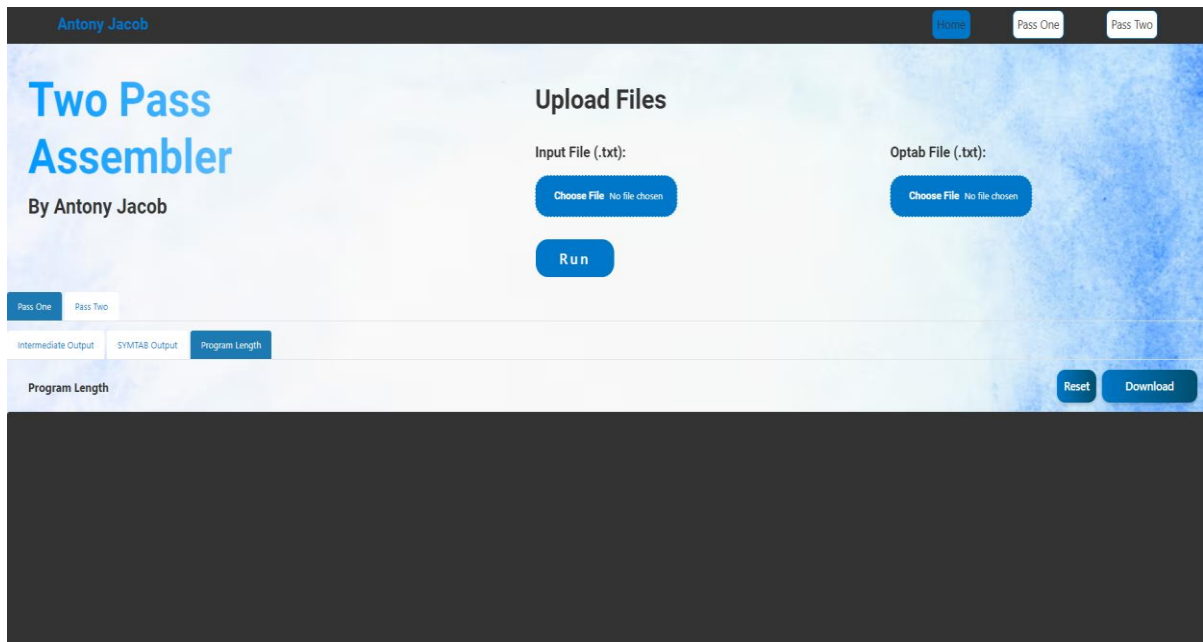
5. Viewing Pass Two Outputs:

- Go to the **Pass Two** tab to check the final outputs. You'll find:
 - **Output:** The overall machine code generated after Pass 2.
 - **Object Code:** The final object code for each instruction in header record, text record and end record (HTE).

6. Downloading Results:

- You can download the results in .txt format for further analysis or use by clicking the **Download** buttons next to each output section.
-

User Interface



The screenshot shows the 'Two Pass Assembler' web interface. At the top, the name 'Antony Jacob' is displayed on the left, and navigation tabs for 'Home', 'Pass One', and 'Pass Two' are on the right. The main header area contains the title 'Two Pass Assembler' and the author 'By Antony Jacob'. Below this, there are two sections for file uploads: 'Input File (.txt):' and 'Optab File (.txt):', each with a 'Choose File' button. A 'Run' button is positioned below the input file section. A progress bar at the bottom indicates the current stage, with 'Pass One' selected and 'Pass Two' disabled. Below the progress bar, there are tabs for 'Intermediate Output', 'SYMTAB Output', and 'Program Length'. The 'Program Length' tab is active, showing a large dark area for the output, with 'Reset' and 'Download' buttons on the right.

Example Input and Optab

Sample Input File (input.txt):

```
PGM1    START    1000
-        LDA      ALPHA
-        MUL      BETA
-        STA      GAMMA
ALPHA    WORD     2
BETA     WORD     4
GAMMA    RESW     1
-        END      1000
```

Sample Optab File (optab.txt):

```
LDA      00
STA      0C
MUL      20
```

Outputs

Intermediate Output:

This file contains information about each line of code and its associated address, generated after Pass 1.

SYMTAB Output:

This output shows the symbol table (SYMTAB) which includes labels used in the input file and their assigned addresses.

Program Length:

Displays the total length of the program.

Output:

The overall machine code generated after Pass 2.

Object Code:

The final object code for each instruction, generated after Pass 2.

Add Ons

Extra Features

- **Reset or Restart:** After finishing a process, you can restart the workflow by refreshing the page or clicking the restart option.
- **Compatibility:** Works on modern browsers (Chrome, Firefox, Edge).

Tips for Effective Use

- **Ensure correct formatting** of the input files.
- **Review the intermediate code** thoroughly before moving to Pass 2 to avoid errors.
- **Use the download feature** to keep a local copy of all outputs.

Troubleshooting

- **File Upload Issues:** Ensure the files are correctly formatted and are of the correct type.
- **No Output in Pass 1:** Check if the input assembly file is correctly structured.
- **Object Code Errors:** If Pass 2 generates incorrect results, revisit Pass 1 and ensure the intermediate code and symbol table were properly generated.

Conclusion

The Two-Pass Assembler web tool simplifies the task of assembling programs by providing an intuitive interface for performing both Pass 1 and Pass 2 operations. It generates crucial outputs such as intermediate files, symbol tables, and object codes, all available for download. Whether you are learning assembly language or working with real-world projects, this tool serves as a vital asset in understanding and generating machine code from assembly instructions.

Take advantage of the structured Pass One and Pass Two processes to turn your assembly language programs into executable object code, all from the comfort of your web browser!