

SIC Assembler Bot - Project Report/User Manual

- Benjamin Francis Thottam

- Roll no. 65

This document provides an overview and detailed analysis of the SIC Assembler Bot project.

1. Introduction

The **SIC Assembler Bot** is a Telegram bot developed to simulate a two-pass assembler for SIC (Simplified Instructional Computer) assembly language. This bot allows users to input SIC assembly code and generate the object code and symbol table, along with intermediate output, through a conversation-based interface on Telegram.

The bot leverages Python's `python-telegram-bot` library for bot functionality and follows a two-pass methodology for assembling the input code.

This document outlines the bot's features, design, and functionality, along with instructions on how to use it effectively.

2. Overview of SIC Assembler Bot

The bot processes assembly code using the traditional two-pass assembler technique:

- **Pass 1:** Constructs the symbol table by reading the source code, tracking all labels and calculating their addresses.
- **Pass 2:** Uses the symbol table to resolve addresses and generate the final object code based on the instruction set (optab).

Users interact with the bot by sending SIC assembly code and the operation table (optab) via Telegram messages. The bot then replies with the results of the assembly process.

3. Design and Implementation

The bot follows a modular architecture, with the following components:

- **Pass 1:** Parses the input SIC code, generates the symbol table, and produces an intermediate file with line addresses.
- **Pass 2:** Uses the intermediate file and symbol table to generate the final object code.
- **Error Handling Module:** Detects errors such as duplicate labels, invalid opcodes, and missing symbols. Error messages are sent back to the user when issues are encountered.
- **Telegram Integration Module:** Manages user interaction through a Telegram bot, built using `python-telegram-bot`. Handles input messages and sends results back to the user.

The bot was developed using Python and relies on asynchronous communication (with `aiohttp` and `asyncio`) to handle Telegram updates efficiently.

4. Features

- **Symbol Table Generation:** The bot builds a symbol table from the input SIC code during Pass 1.
- **Object Code Generation:** The final machine code is generated during Pass 2.
- **Error Detection and Reporting:** The bot detects common assembly errors (e.g., invalid opcodes, undefined symbols) and reports them to the user.
- **Telegram Integration:** Users can interact with the bot using simple commands and text inputs through the Telegram app.
- **Intermediate Code Display:** The bot shows intermediate results like the symbol table and location counters for better debugging and understanding.
- **File Management:** The bot allows users to upload files containing their input code and optab via Telegram.

5. User Interface

The Telegram bot features a simple, user-friendly conversational interface. It consists of the following elements:

- **Input Code Section:** Users can either type or upload their SIC assembly code in plain text. The bot expects code formatted in a standard SIC assembly format.
- **Operation Table (Optab) Section:** Users provide an optab containing the opcode-to-machine code mappings. This can be input manually or uploaded as a `.txt` file.
- **Output Section:** After the two-pass assembly process, the bot responds with:
 - **Intermediate Output:** Displays the intermediate file, including location counters.
 - **Symbol Table:** Lists all defined symbols and their memory addresses.
 - **Final Object Code:** Outputs the assembled machine code in a format ready for execution on a SIC machine.

Bot Commands:

- `/start`: Begins a new session with the user and requests input code.

Conversation Flow:

1. **Start:** The bot welcomes the user and requests input code.
2. **Input Code:** The user sends SIC assembly code.
3. **Optab:** The bot asks for the operation table (optab) to be sent.
4. **Results:** After processing, the bot responds with the intermediate output, symbol table, and final object code.

6. Screenshots

Close

SIC Assembler bot



What can this bot do?

SIC Assembler bot is built to convert assembly language code into machine-readable object code using a two-pass methodology.

⌵ Tap here to use this bot

Start

Close

SIC Assembler

bot

What can this bot do?

SIC Assembler bot is built to convert assembly language code into machine-readable object code using a two-pass methodology.

Today

/start 10:06 PM ✓

Hi Ben! Welcome to the SIC Assembler bot!

Enter Input Code to Assemble

10:06 PM

```
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
```

10:08 PM ✓

Enter Optab 10:08 PM

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

10:08 PM ✓

Intermediate Output:

```
- 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
```

Symtab Output:

```
ALPHA 100C
ONE 100F
TWO 1011
BETA 1014
```

Final Output:

```
- COPY START 1000
```

Menu



Message



Close

SIC Assembler

bot



```
- STA BETA
ALPHA BYTE C'CSE'
ONE RESB 2
TWO WORD 2
BETA RESW 2
- END 1000
```

10:08 PM ✓

Enter Optab

10:08 PM

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

10:08 PM ✓

Intermediate Output:

```
- 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
```

Symtab Output:

```
ALPHA 100C
ONE 100F
TWO 1011
BETA 1014
```

Final Output:

```
- 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
```

Object Code:

```
H^COPY^001000^000014
T^001000^12^^00100C^01100F^051011^231014^43534
5^000002
E^001000
```

10:08 PM

Menu



Message



7. Requirements

User Requirements:

- **Platform:** Telegram app (available on Android, iOS, Windows, macOS, or any web browser).
- **Input:** Users must input SIC assembly code and an optab (operation table) either by typing or uploading files through the Telegram chat interface.

Developer Requirements:

- **Programming Language:** Python 3.x is used for the development of the bot.
- **Development Environment:** The project can be developed using any Python-friendly IDE, such as PyCharm or VSCode.
- **Libraries/Frameworks:**
 - **python-telegram-bot:** Used for handling Telegram bot interactions.
 - **aiohttp:** For managing asynchronous HTTP connections.
 - **asyncio:** To handle asynchronous execution and message polling.
 - **Logging:** To record bot activity and errors.
 - **File Handling Modules:** For reading and processing uploaded files (e.g., `os` for environment variables).

8. Application

The SIC Assembler Bot successfully implements the two-pass assembly process. It converts SIC assembly language into object code and provides users with symbol tables and error handling through an intuitive conversational interface in Telegram.

The bot is hosted on a cloud platform and can be accessed by adding the bot username **@sicassemblerbot** in Telegram. The source code is available for review and further development on [GitHub](#).