TWOPASS ASSEMBLER- DOCUMENTATION

BY : AGNES JAMES

ROLL NO:13

Introduction

The Two-Pass Assembler is a graphical user interface (GUI) application that simulates the functionality of a two-pass assembler, which translates assembly language code into object code for a hypothetical machine. The assembler takes input for the assembly code and generates the corresponding object code using a symbol table (SYMTAB) and an operation table (OPTAB). This manual provides a step-by-step guide for using the program.

Requirements to Develop the Code

To develop and run this Two-Pass Assembler program, you need the following:

1. Development Environment

- **Java Development Kit (JDK):**
  - **Ensure that JDK 8 or above is installed, as the program uses javax.swing.* for the graphical user interface (GUI).**
- **Integrated Development Environment (IDE):**
  - **You can use an IDE like Visual Studio Code, Eclipse, IntelliJ IDEA, or NetBeans to write, compile, and run the code.**
  - **If using Visual Studio Code, ensure that you have the Java Extension Pack installed, which includes tools for Java development.**

2. Dependencies and Libraries

- **The code uses the javax.swing.* package for building the GUI and java.awt.event.* for handling action events.**
  - **Ensure that you have the Swing and AWT libraries available, which are included by default in JDK 8 and above.**

Requirements to Run the Java Application

The requirements focus on what's needed to run the Two-Pass Assembler program once it has been developed and packaged.

1. Java Runtime Environment (JRE)

- Users must have JRE (Java Runtime Environment) installed on their system. JRE allows users to run Java programs without needing the full development kit (JDK).
- Download JRE from the Oracle website or use OpenJDK's version of the runtime.

2. User Guide to Run and Use the Application

- Once the executable is ready, users need to:
  - Install Java Runtime Environment (JRE): If not already installed, users must have JRE.
  - Open the Application: Run the .exe or .jar file by double-clicking the executable or using the terminal/command prompt.
  - Provide Input:
    - Enter the operation table (OPTAB) and assembly instructions in the provided text areas in the GUI.
    - Click "Pass One" to generate the symbol table and intermediate code.
    - Click "Pass Two" to generate the object code and final assembly.
  - View the Results: The output, including the symbol table, intermediate code, and machine/object code, will appear in the output text area.

3. System Requirements for Users:

- Operating System:
  - Any operating system that supports Java, including Windows, macOS, and Linux.
- Java Version:
  - Ensure Java 8 or above is installed, which supports Swing and AWT components for GUI

Data Structures:

- HashMap<String, String> symtab: The symbol table storing labels and their addresses.
- HashMap<String, String> optab: The operation table storing opcodes and their machine code equivalents.
- StringBuilder intermediate: Used to store the intermediate code during Pass One.
- StringBuilder symtabOutput: Used to store the symbol table output.

- **int finalLocctr: Stores the final value of the location counter, representing the total memory used by the program.**

Running the Application

1. Compile the Program:

- **Use a Java IDE or command line to compile the TwoPassAssembler.java file**

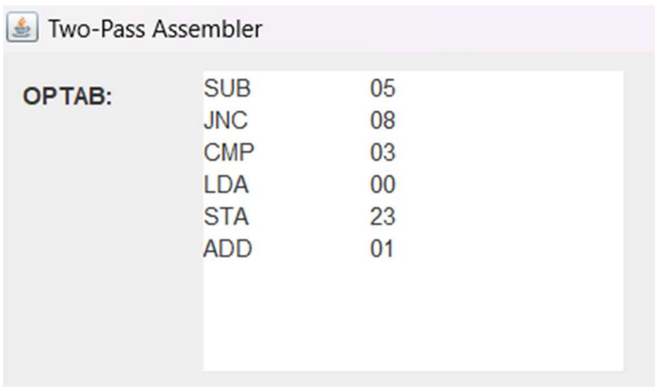
2. Run the Program:

- **Execute the compiled class file**

3. The GUI window will appear with options to input assembly code, operation table, and buttons for performing both passes of the assembler.

Understanding the GUI Layout:

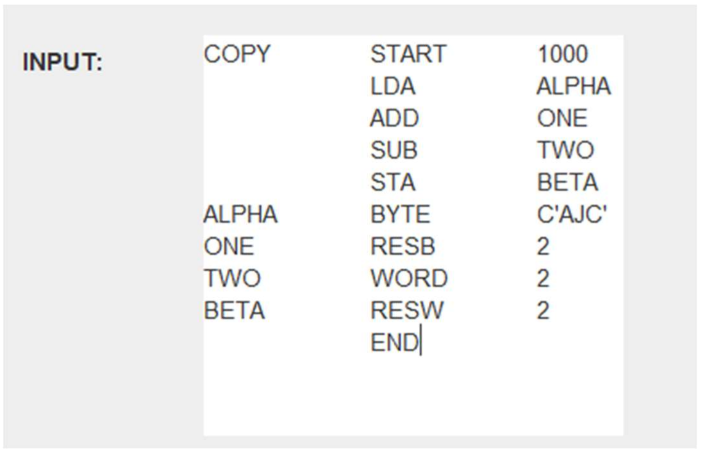
- ### OPTAB Input Area:
  - **Location: Top left.**
  - **Purpose: To input the operation table (OPTAB), which contains the list of mnemonics and their corresponding opcode values.**



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

. INPUT Area:

- **Location: Middle left.**
- **Purpose: To input the assembly language code for the assembler to process.**



INPUT:

```
COPY    START   1000
        LDA     ALPHA
        ADD     ONE
        SUB     TWO
        STA     BETA
ALPHA   BYTE    C'AJC'
ONE     RESB    2
TWO     WORD    2
BETA    RESW    2
        END
```

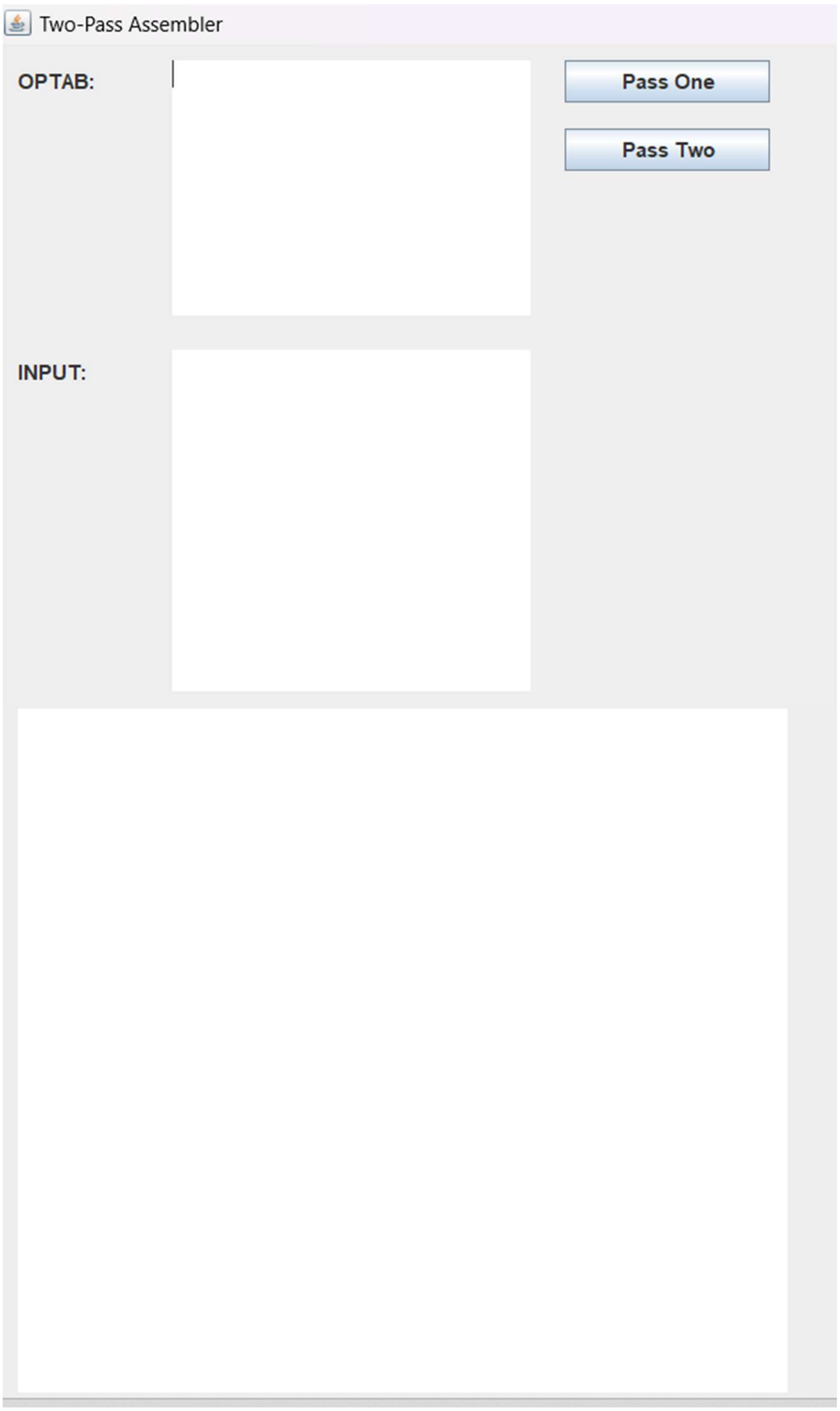
Buttons:

- **Pass One:**
  - **Location: Top center.**
  - **Function: Executes the first pass of the assembler, which generates the SYMTAB (Symbol Table) and the intermediate representation.**
- **Pass Two:**
  - **Location: Below Pass One button.**
  - **Function: Executes the second pass, which generates the object code.**

Output Area:

- **Location: Bottom.**

- **Displays the results, including SYMTAB, intermediate code, and object code for both passes.**



Using the Program

Step 1: Input the OPTAB

- Enter the **operation table** in the top text area labeled "OPTAB." Each instruction should be followed by its opcode. For example:

Step 2: Input the Assembly Code

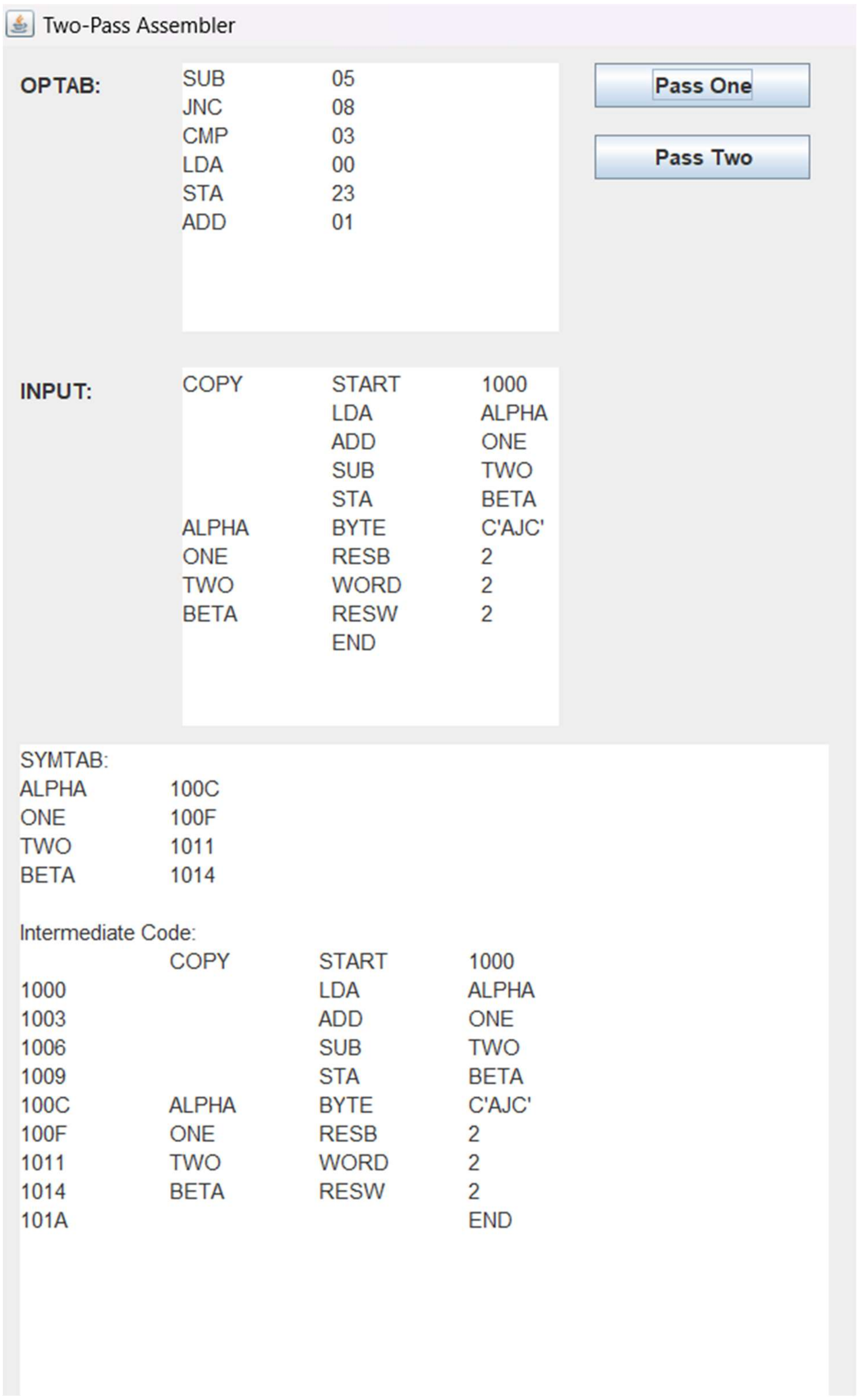
- **In the "INPUT" text area, input your assembly code. This code should contain labels, instructions, and operands in the standard format. Each line should consist of a label (optional), an opcode, and an operand.**

Step 3: Perform Pass One

- **Click the "Pass One" button to execute the first pass of the assembler.**
- **The following will be generated and displayed in the output area:**
  - **SYMTAB: Contains the symbol addresses from the assembly code.**
  - **Intermediate Code: The intermediate representation with calculated addresses and mnemonics.**

Step 4: Perform Pass Two

- **After running Pass One, click the "Pass Two" button to generate the object code for the assembly program.**
- **This includes:**
  - **Object code in the format suitable for the machine.**
  - **Text records and header records based on the addresses calculated in Pass One.**



OPTAB:

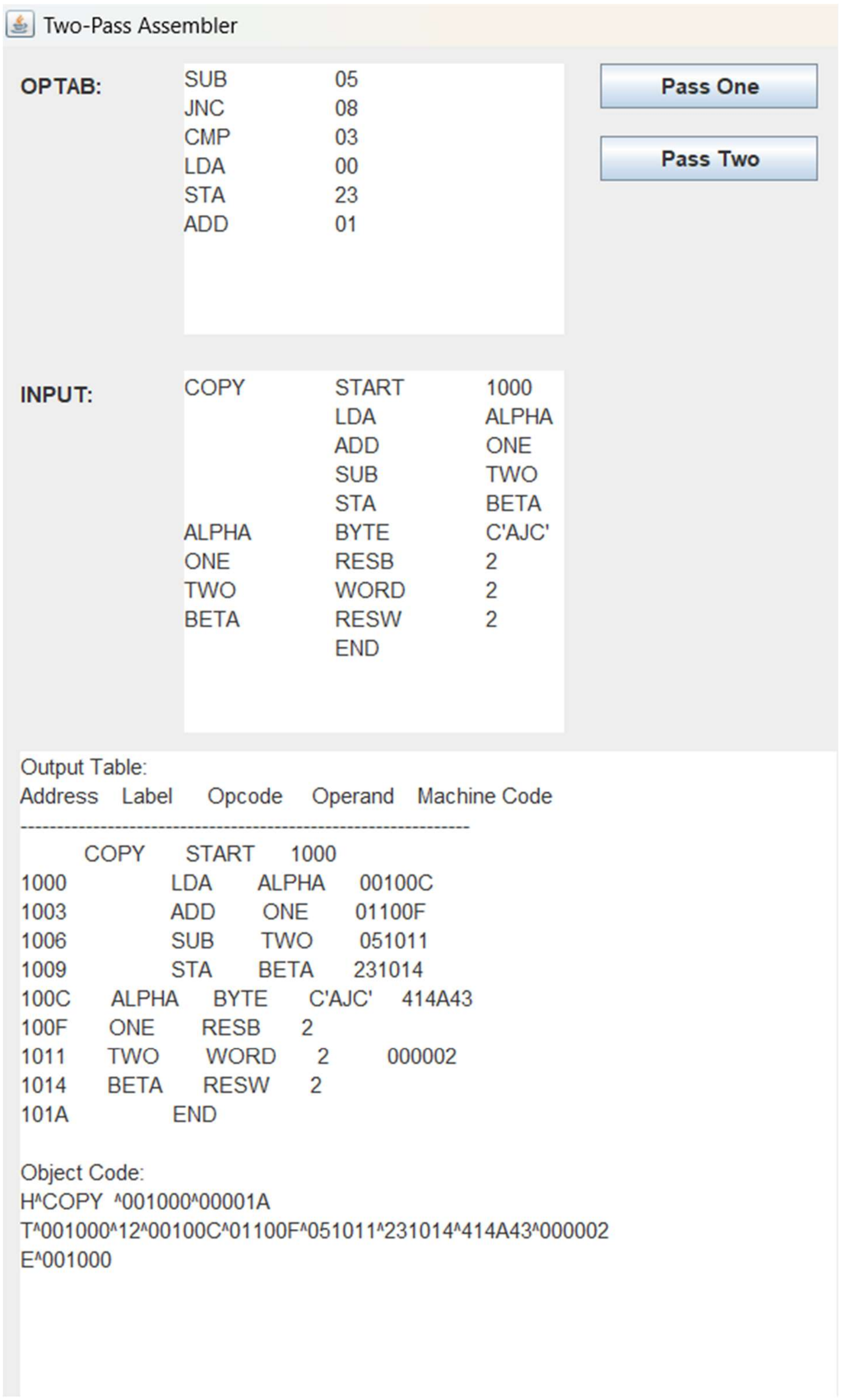
```
SUB         05
JNC         08
CMP         03
LDA         00
STA         23
ADD         01
```

INPUT:

```
COPY        START       1000
            LDA         ALPHA
            ADD         ONE
            SUB         TWO
            STA         BETA
ALPHA       BYTE        C'AJC'
ONE         RESB        2
TWO         WORD        2
BETA        RESW        2
            END
```

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

Intermediate Code:
            COPY        START       1000
1000                    LDA         ALPHA
1003                    ADD         ONE
1006                    SUB         TWO
1009                    STA         BETA
100C        ALPHA       BYTE        C'AJC'
100F        ONE         RESB        2
1011        TWO         WORD        2
1014        BETA        RESW        2
101A                                END
```



OPTAB:

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

INPUT:

```
COPY      START     1000
          LDA       ALPHA
          ADD       ONE
          SUB       TWO
          STA       BETA
ALPHA     BYTE      C'AJC'
ONE       RESB      2
TWO       WORD      2
BETA      RESW      2
          END
```

```
Output Table:
Address   Label    Opcode    Operand   Machine Code
-------------------------------------------------------------
         COPY     START     1000
1000              LDA       ALPHA     00100C
1003              ADD       ONE       01100F
1006              SUB       TWO       051011
1009              STA       BETA      231014
100C     ALPHA    BYTE      C'AJC'    414A43
100F     ONE      RESB      2
1011     TWO      WORD      2         000002
1014     BETA     RESW      2
101A              END

Object Code:
H^COPY  ^001000^00001A
T^001000^12^00100C^01100F^051011^231014^414A43^000002
E^001000
```

Github link: https://github.com/agnesjames-2026/TwoPass-Assembler-GUI