CISC ASSEMBLER USER MANUAL DOCUMENT

Tejasvi Chigurupati
Vivek Kommareddy
Isha Paliwal
Yashwanth Raj Varadharajan

We had run the project in IntelliJ and we recommend the same to use in your system with jdk version 17.

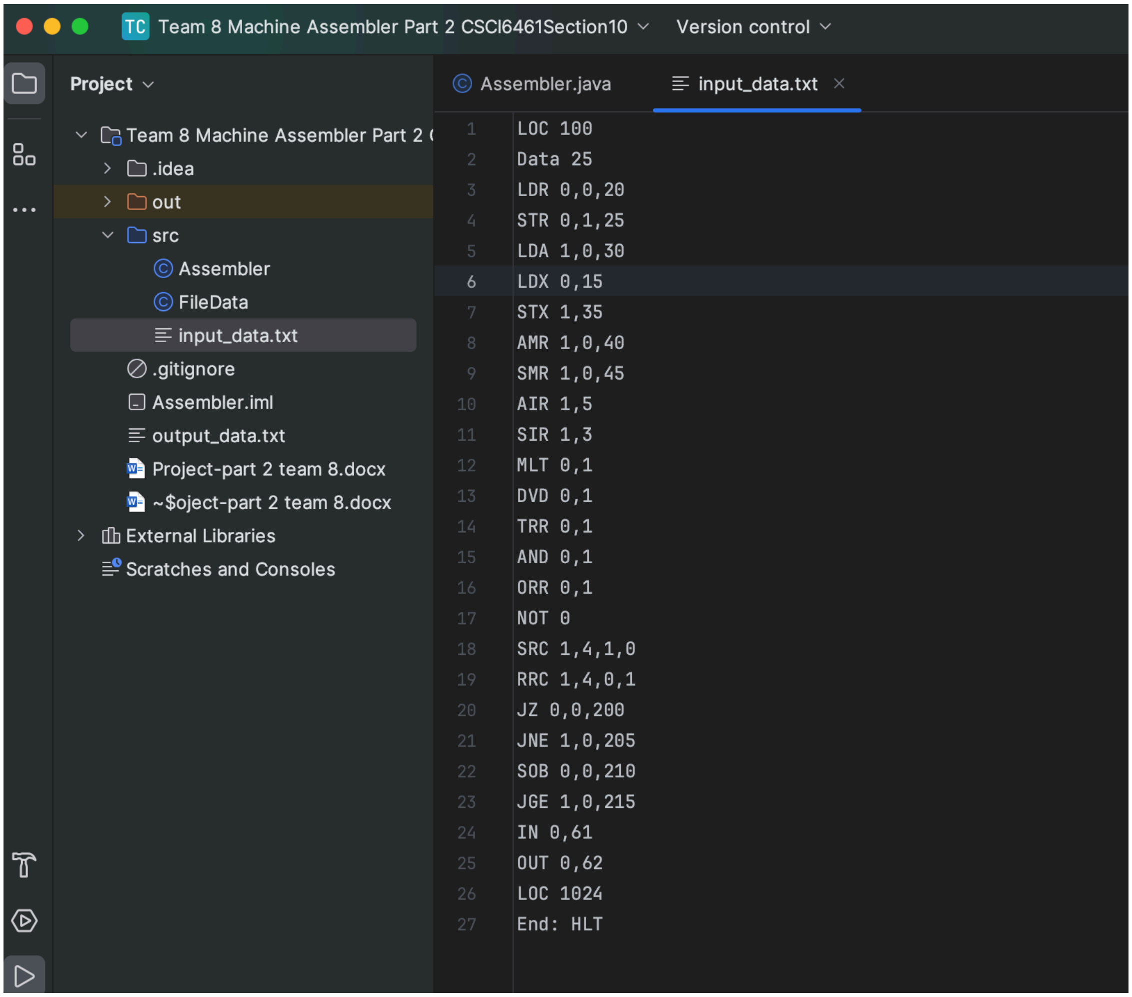
As previously indicated, the project folder "Team 8 Machine Simulator Part 1 CSCI6461Section10-1.zip" has already been uploaded. This folder contains the design notes and user manual that you may refer to.

In order to use the Assembler, you must first create an input file called input_data.txt. After that, you must run assembler.java to change the file into an output_data.txt file.

ASSEMBLER

This project's main objective is to extract data and translate it into hexadecimal code from the provided input text file. To do this conversion, each instruction in the input file must be represented using a syntax similar to XXXX YYYY, where XXXX denotes the program counter and YYYY denotes the decoded instruction.

As a point of reference, the original data may be found in the input_data.txt file, which is where the intended output is created.



```
LOC 100
Data 25
LDR 0,0,20
STR 0,1,25
LDA 1,0,30
LDX 0,15
STX 1,35
AMR 1,0,40
SMR 1,0,45
AIR 1,5
SIR 1,3
MLT 0,1
DVD 0,1
TRR 0,1
AND 0,1
ORR 0,1
NOT 0
SRC 1,4,1,0
RRC 1,4,0,1
JZ 0,0,200
JNE 1,0,205
SOB 0,0,210
JGE 1,0,215
IN 0,61
OUT 0,62
LOC 1024
End: HLT
```

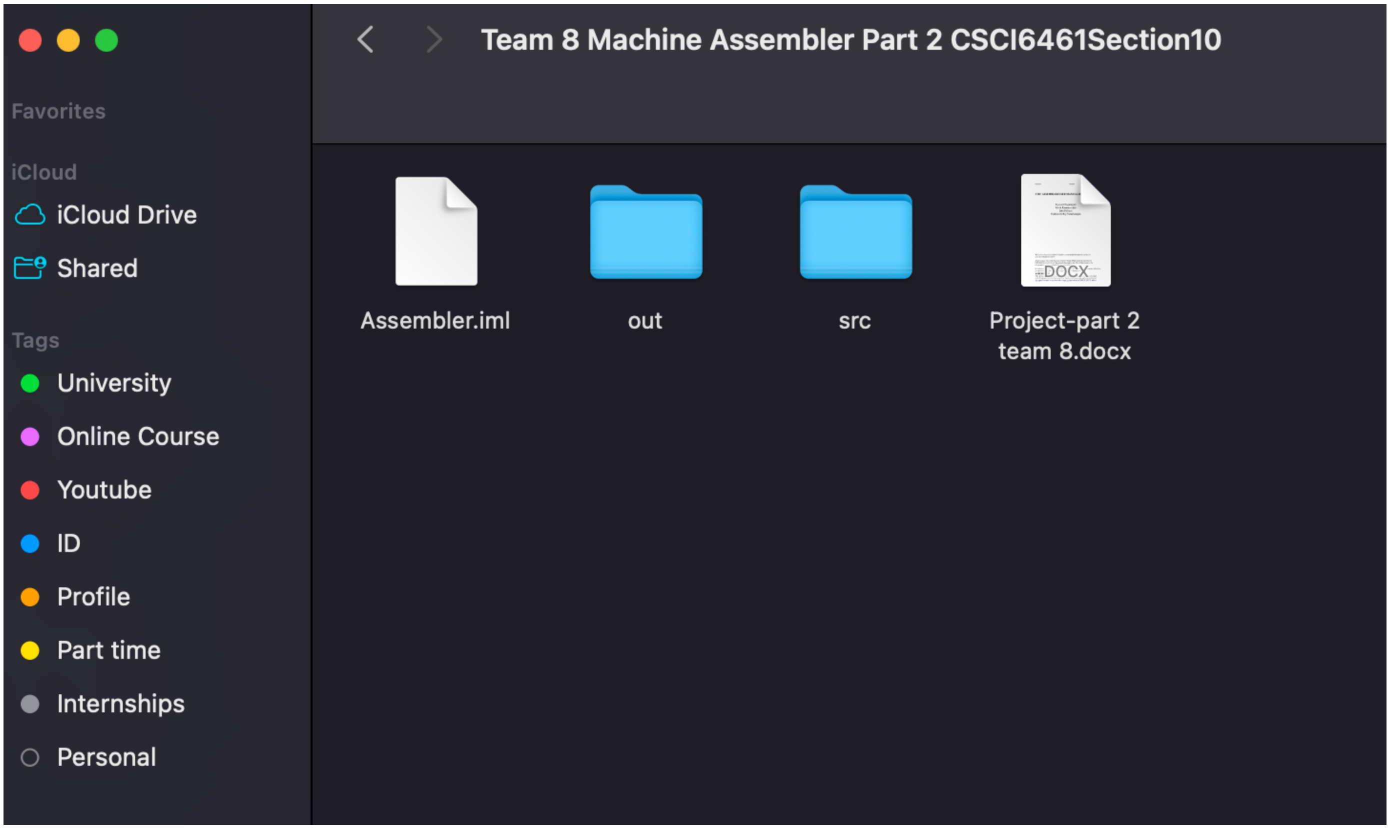
Overview of assembler.java:

This specialized Java application is tailored for a specific architectural framework, and it fulfills several vital roles within this context. Its capabilities include the retrieval of data from a file, the processing of commands, the generation of machine instructions, and the exportation of the final output to a separate file.
At the core of this process lies the pivotal generateInstruction function, responsible for handling diverse types of instructions. It meticulously configures the OpCode, R, IX, I, and Address attributes to ensure precise encoding.
The management of input rows and the integration of the resulting instructions into the outputData collection are skillfully overseen by the encodeInstructions function, which also keeps track of the LOC (location counter).

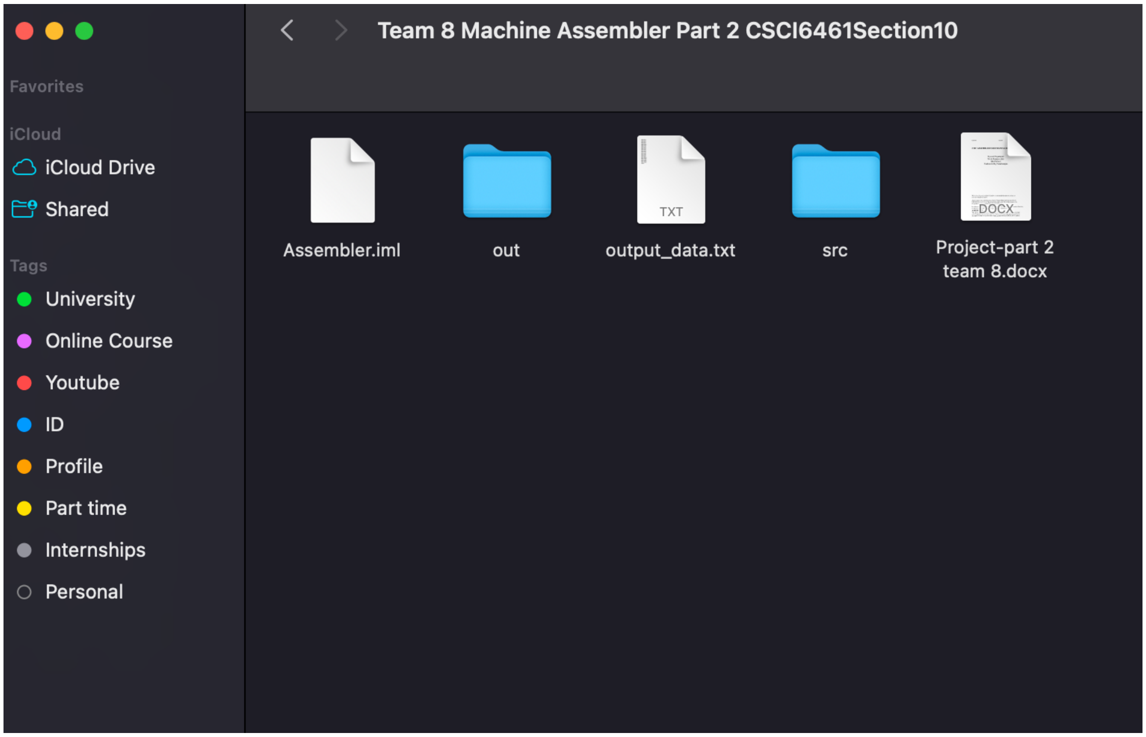
Impressively, the application exhibits proficiency in dealing with exceptional cases, such as instructions labeled as "End" and "Data." When supplied with valid input, it promptly generates the desired output and saves it in a file named "output_data.txt."

How to Run the Assembler?

We must unzip the Team 8 Machine Assembler Part 2 CSCI6461Section10.zip file before running the assembler. After that is finished, all we have to do is double-click the Assembler.jar application in the unzipped folder.



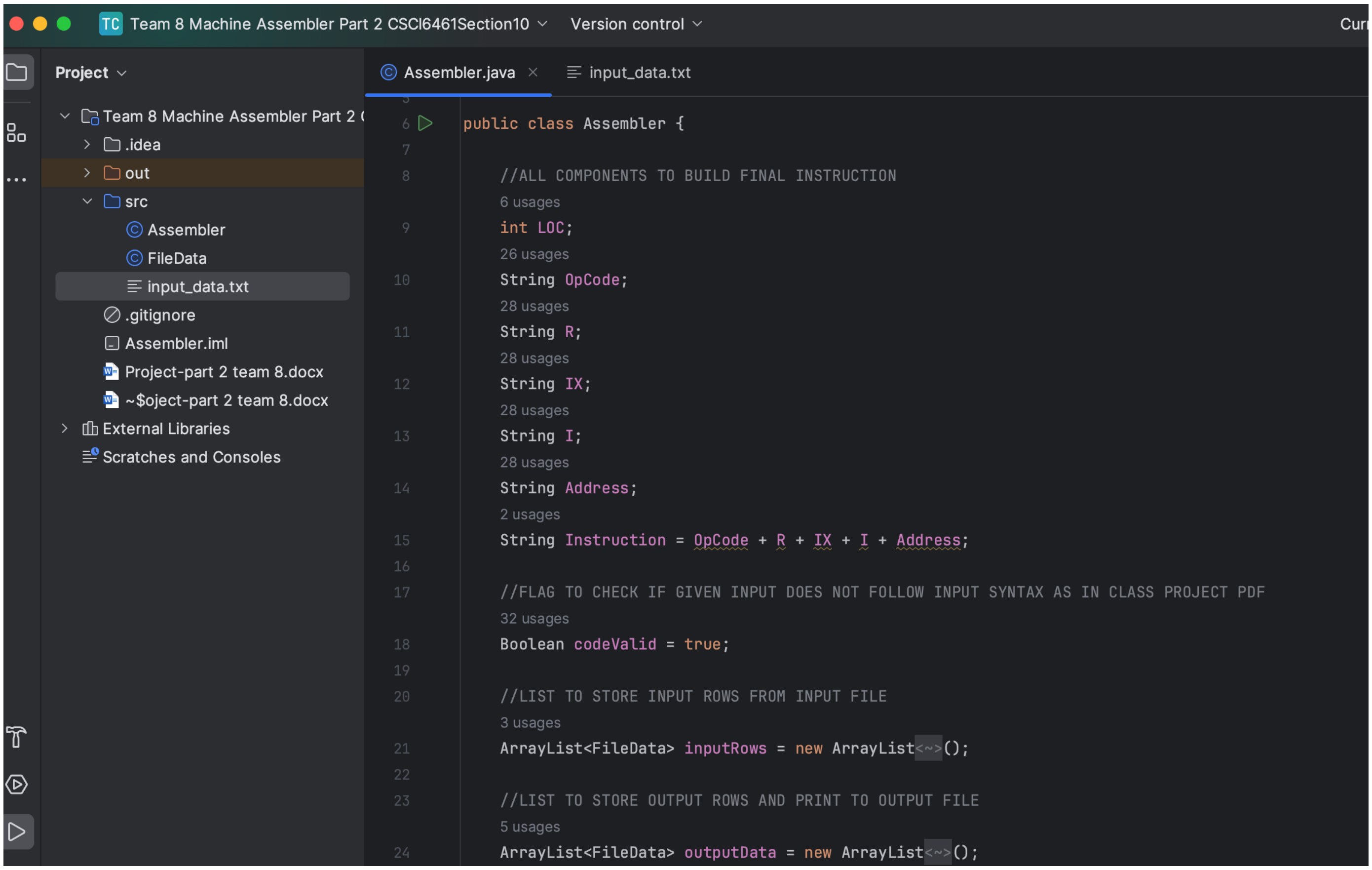
After the execution, the output_data.txt file will be found in the same folder.



How to run the Assembler using the Code?

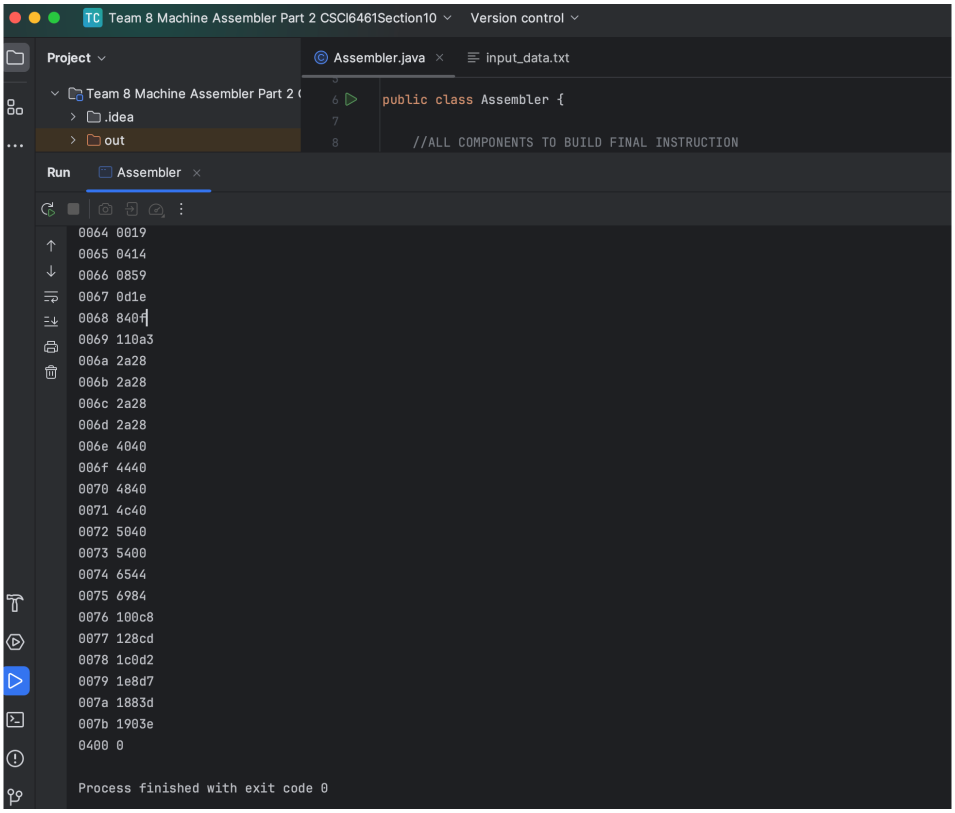
Access to the following files is available within the Assembler folder:

src -> Assembler.java in Assembler
src -> Assembler -> input_data.txt
Result.txt
Using IntelliJ IDEA, you may choose to run the Assembler.java file.



```
public class Assembler {

    //ALL COMPONENTS TO BUILD FINAL INSTRUCTION
    6 usages
    int LOC;
    26 usages
    String OpCode;
    28 usages
    String R;
    28 usages
    String IX;
    28 usages
    String I;
    28 usages
    String Address;
    2 usages
    String Instruction = OpCode + R + IX + I + Address;

    //FLAG TO CHECK IF GIVEN INPUT DOES NOT FOLLOW INPUT SYNTAX AS IN CLASS PROJECT PDF
    32 usages
    Boolean codeValid = true;

    //LIST TO STORE INPUT ROWS FROM INPUT FILE
    3 usages
    ArrayList<FileData> inputRows = new ArrayList<~>();

    //LIST TO STORE OUTPUT ROWS AND PRINT TO OUTPUT FILE
    5 usages
    ArrayList<FileData> outputData = new ArrayList<~>();
```

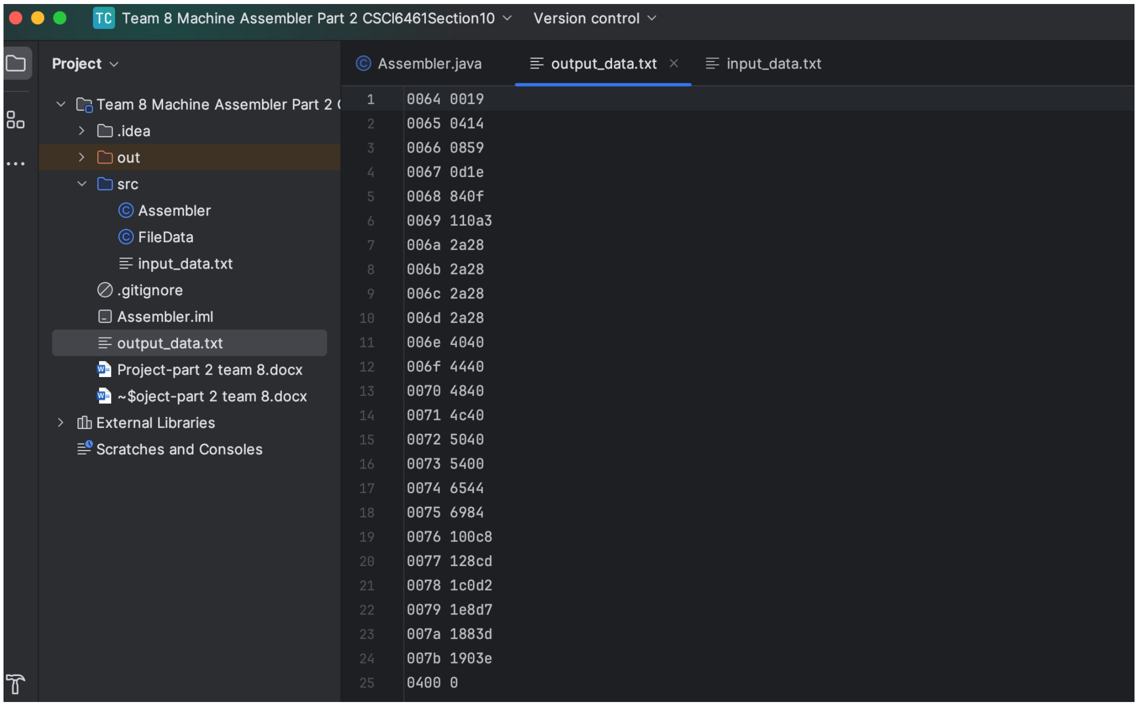
The application concurrently saves its result to a file called "output_data.txt" and displays it in the terminal window.


Team 8 Machine Assembler Part 2 CSCI6461Section10
Version control
Project
Assembler.java
input_data.txt
Team 8 Machine Assembler Part 2 (
.idea
out
Run
Assembler

```java
public class Assembler {

    //ALL COMPONENTS TO BUILD FINAL INSTRUCTION
```

```
0064 0019
0065 0414
0066 0859
0067 0d1e
0068 840f
0069 110a3
006a 2a28
006b 2a28
006c 2a28
006d 2a28
006e 4040
006f 4440
0070 4840
0071 4c40
0072 5040
0073 5400
0074 6544
0075 6984
0076 100c8
0077 128cd
0078 1c0d2
0079 1e8d7
007a 1883d
007b 1903e
0400 0

Process finished with exit code 0
```

In output_data.txt,



```
0064 0019
0065 0414
0066 0859
0067 0d1e
0068 840f
0069 110a3
006a 2a28
006b 2a28
006c 2a28
006d 2a28
006e 4040
006f 4440
0070 4840
0071 4c40
0072 5040
0073 5400
0074 6544
0075 6984
0076 100c8
0077 128cd
0078 1c0d2
0079 1e8d7
007a 1883d
007b 1903e
0400 0
```