Alexandria University

Final Project – Phase I

SIC/XE assembler

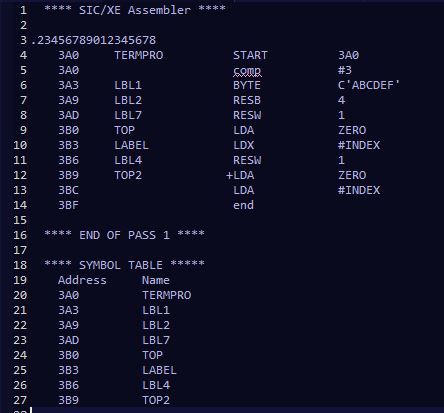
Zahraa Emara – 4558

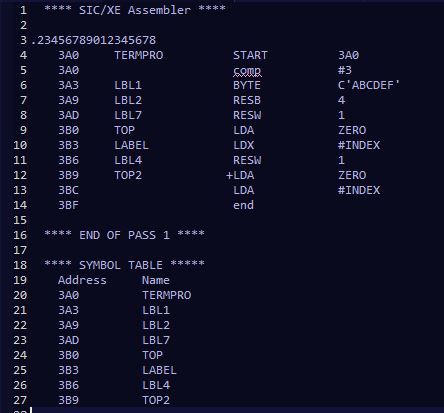
Sarah Yousry – 4582

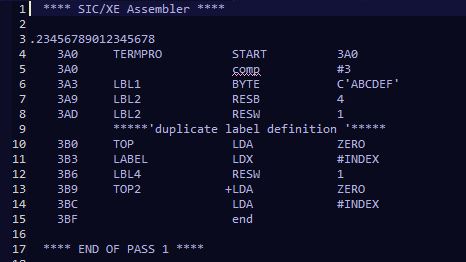
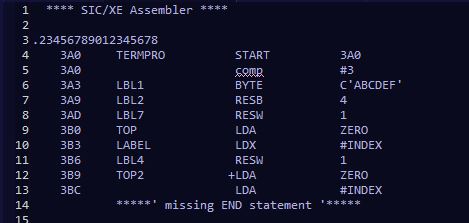
Mayar El Mahdy – 4639

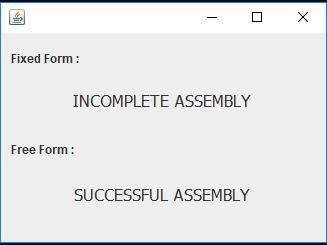
Mariam Beltagy – 5109

1. **Requirement specification**
2. The parser is to handle all storage directives.  
     
     
   The program counter is calculated relevant to each directive.
3. The symbol table.



1. List file.  
     
   
2. A meaningful error message should be printed below the line in which the error occurred.   
   Case I : The user declares 2 duplicate labels

  
  
 Case II: The user forgets adding the END statement   
 

1. **Design**we used a simple GUI to display a message to show whether the assembling was successful or not. ****
2. **Main Data structures**

* Arrays

1. **Algorithms Description**

It’s implemented in the (Fixed Controller) class. It scans the source file Instruction by instruction.

First the instruction is read and stored in a string, then this string is divided into words and each word is identified either as a label, an opcode or an operand and stored in its specified array accordingly as demonstrated below, if a comment is detected the dividing step is skipped and instead it’s stored in the comments array.

For example, if the instruction **PROG START 1000** is read the algorithm stores it in the arrays as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| index | Label Array | Opcode Array | Operands Array |
| 0 | **PROG** | **START** | **1000** |

But, if the instruction **ADDR A,X** is read to preserve the sequence of instructions an empty space is stored in the label array as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| index | Label Array | Opcode Array | Operands Array |
| 1 | NULL | **ADDR** | **A,X** |

After scanning the entire file and populating the 4 main arrays we reconstruct the instruction by adding the first element in each array then we validate them together. If there’s an error a relevant message is stored in the error array then the instruction is printed in the list file after that the second instruction is reconstructed and validated in the same manner then printed and so on.

1. **Assumptions**

* In the START instruction if the hexadecimal address was written in a wrong format the assembler assumes it to be equal to zero to avoid interrupting the program counter from calculating the addresses of the rest of the instructions.

1. **Sample Runs**

### Implementation of the fixed format:

### **Source File:** **List File:**

### Implementation of the free format:

### **Source File:** **List File:**