SIC/XE Assembler

Producing code for the absolute loader used in the SIC/XE programming assignments using One Pass Assembler.

Names:

Aya Gamal Mohamed	1
Enas Morsy Mohamed	20
Sara Mohamed Youssef	31
Linh Ahmed	50
Nada Fathy Ali	68

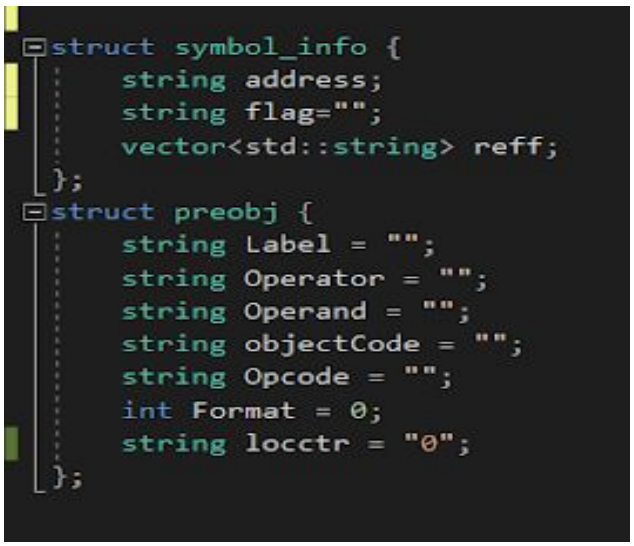
Specifications

- Build a parser that is capable of handling source lines that are instructions, storage declaration, comments, and assembler directives.
- For instructions, the parser is to minimally be capable of decoding 2, 3 and 4-byte instructions as follows:
  - ➢ 2-byte with 1 or 2 symbolic register reference (e.g., TIXR A, ADDR S,A)
  - ➢ RSUB (ignoring any operand or perhaps issuing a warning)
  - ➢ 3-byte PC-relative with symbolic operand to include immediate, indirect, and indexed addressing
  - ➢ 3-byte absolute with non-symbolic operand to include immediate, indirect, and indexed addressing
  - ➢ 4-byte absolute with symbolic or non-symbolic operand to include immediate, indirect, and indexed addressing
- The parser is to handle all storage directives (BYTE, WORD, RESW, and RESB).
- The assembler should support:
  - ➢ EQU and ORG statements.
  - ➢ Simple expression evaluation.
  - ➢ A simple expression includes simple (A B) operand arithmetic, where is one of +, -, *, / and no spaces surround the operation, eg. A+B.

Main data structures

➢ vector<preobj> table: which contains all the instructions with their locctr and objectCode .

➢ Map<string,symbol_info>:SYMTAB: which represents the symbol table.



```
struct symbol_info {
    string address;
    string flag="";
    vector<std::string> reff;
};
struct preobj {
    string Label = "";
    string Operator = "";
    string Operand = "";
    string objectCode = "";
    string Opcode = "";
    int Format = 0;
    string locctr = "0";
};
```

Design

- operations.h/operations.cpp

  Contain functions to get format,opCode and no. of operand of each operator.

- structures.h/structures.cpp

  Contain the two main structures in the program.

- Conversions.h/Conversions.cpp/Convert.h/Convert.cpp

  Contain all the conversions used in the program

  Ex: from bin to Hex,from dec to Hex,.....etc

- ObjectCode.h/ObjectCode.cpp

  Contain functions

  - ➢ to store registers and labels with their address in the symbol table.
  - ➢ to form the object code using the symbol table and by calling the Formats class.
- Formats.h/Formats.cpp

  Contains functions to form object code of format 3 or 4 by calculating its displacement and the nixbpe.

- WriteFile.h/WriteFile.cpp

  To write the object file using the symbol table and Table map.

Algorithms description

The algorithm is to implement one pass assembler which translates mnemonic operation codes to their machine language equivalents.

- Read the file line by line.
- For each line check its validation and print error if a wrong line is entered.
- If the line is correct , remove extra spaces and split the instruction to its label,operator and operand and store them.
- Check if the instruction is the START one save the program name and set the locctr to the operand.

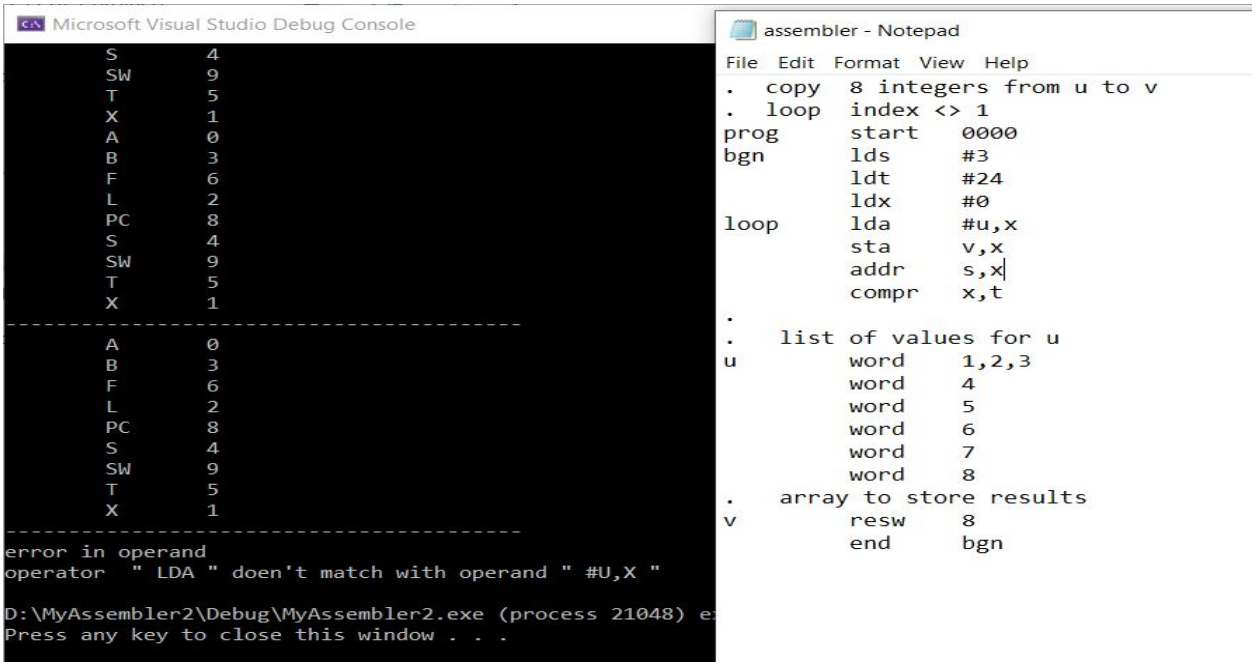
- Check if the instruction is one from the storage directives(WORD,RESW,...) ,store the label in the symbol table and increment the locctr according to its case .
- Otherwise get the format and opcode of the operator and form the objectCode and add format to the locctr.
- Check if all the labels are defined ,write the object file .Otherwise,print error message.

Assumptions

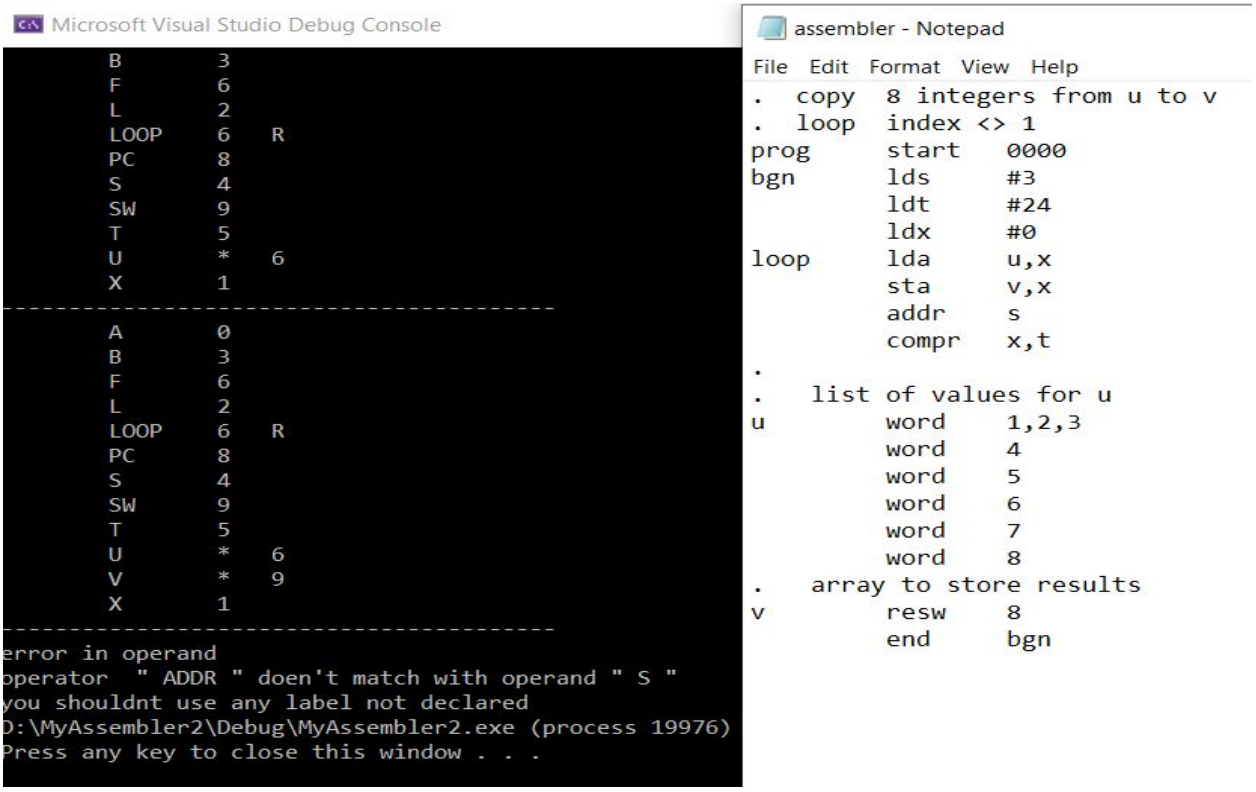
- If the line is the EQU instruction the locctr doesn't change.”as not mentioned in the lectures or text book”
- If the label is used before it was defined ,put 0000/000000 in the objectCode and when the label is found write in the object file each modification should be done.

Sample runs.

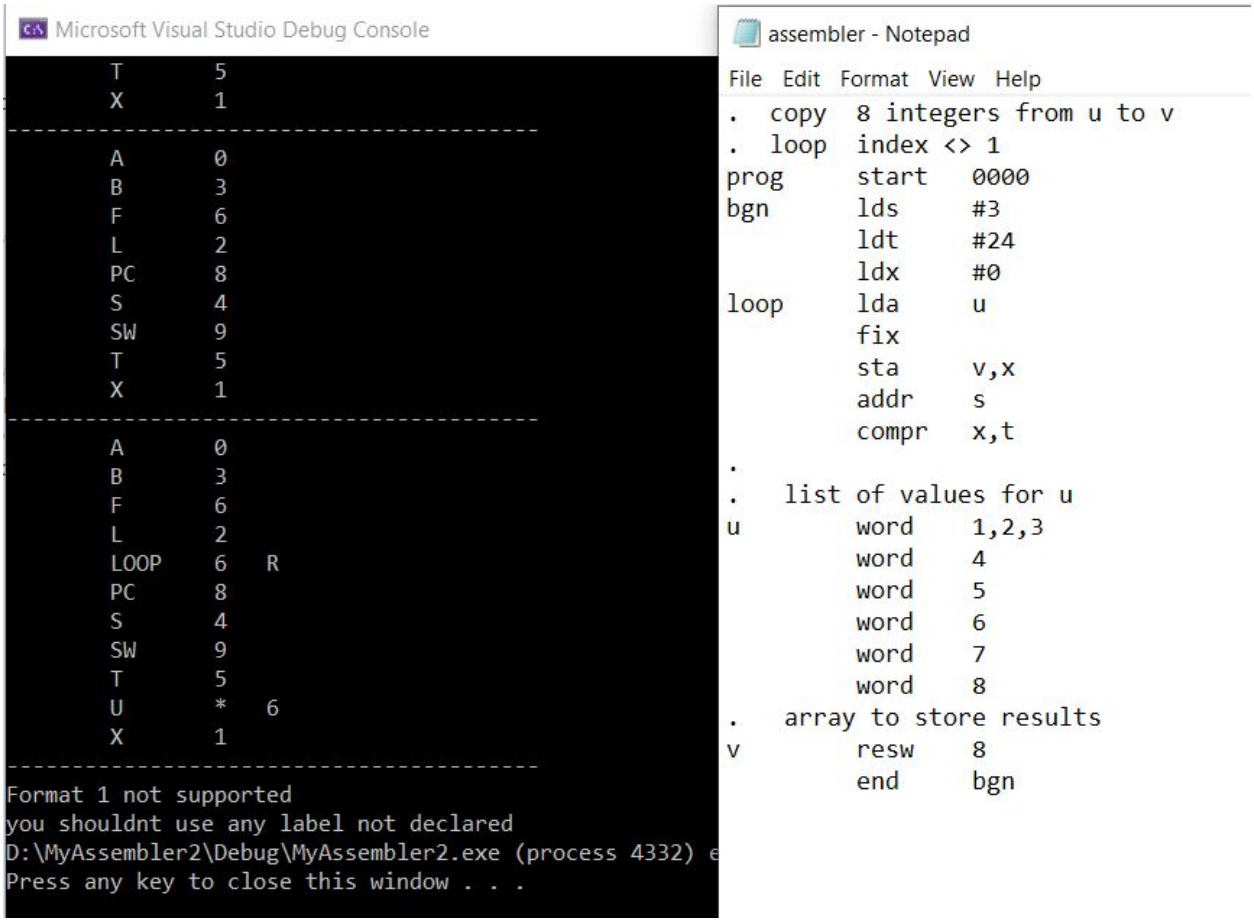
- Not allow to use immediate addressing and indexing at the same time.



- Number of operands of some operators such as ADDR must be 2.

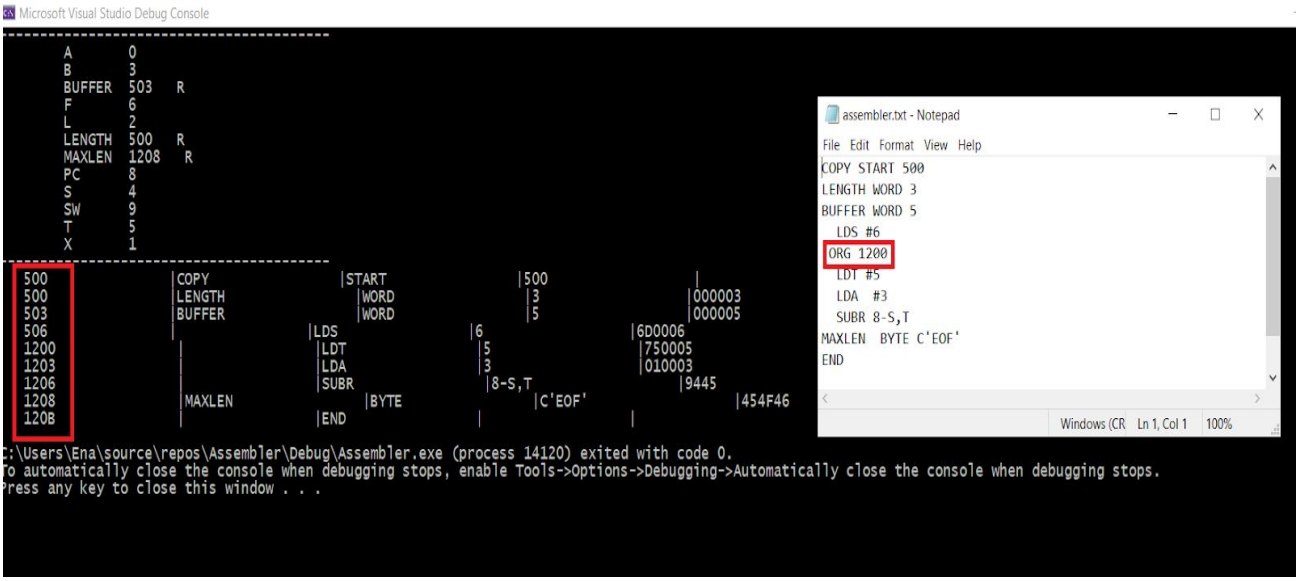


Format 1 not supported.

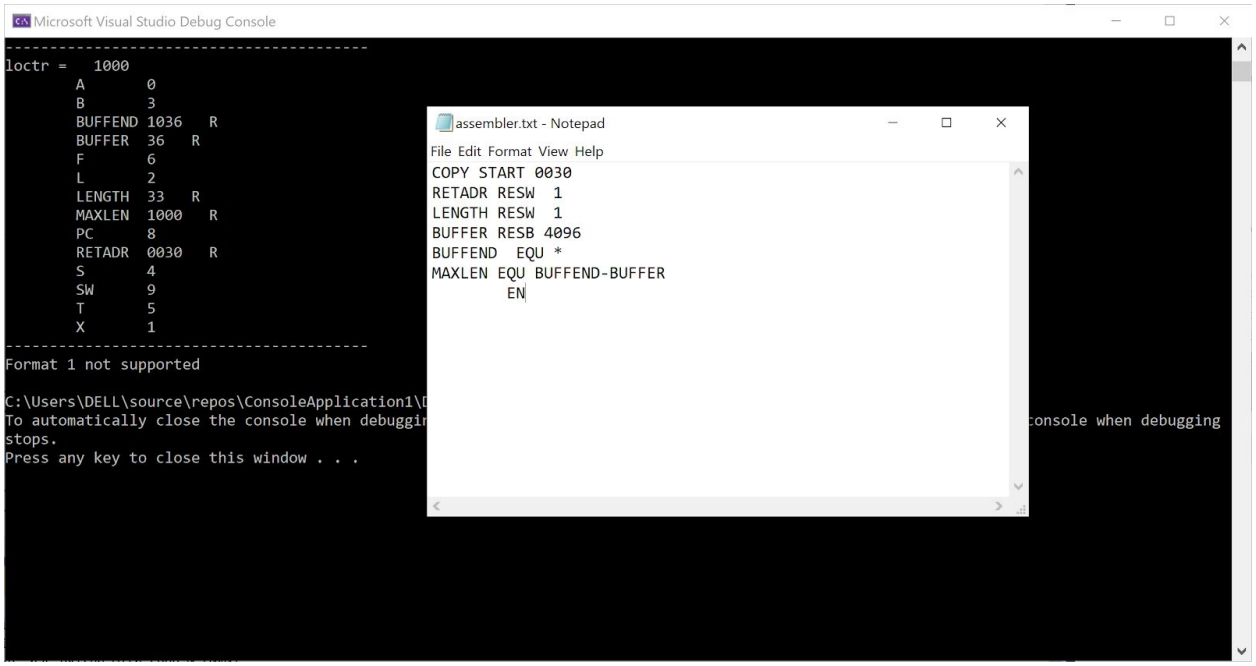


Symbol statement

- ORG

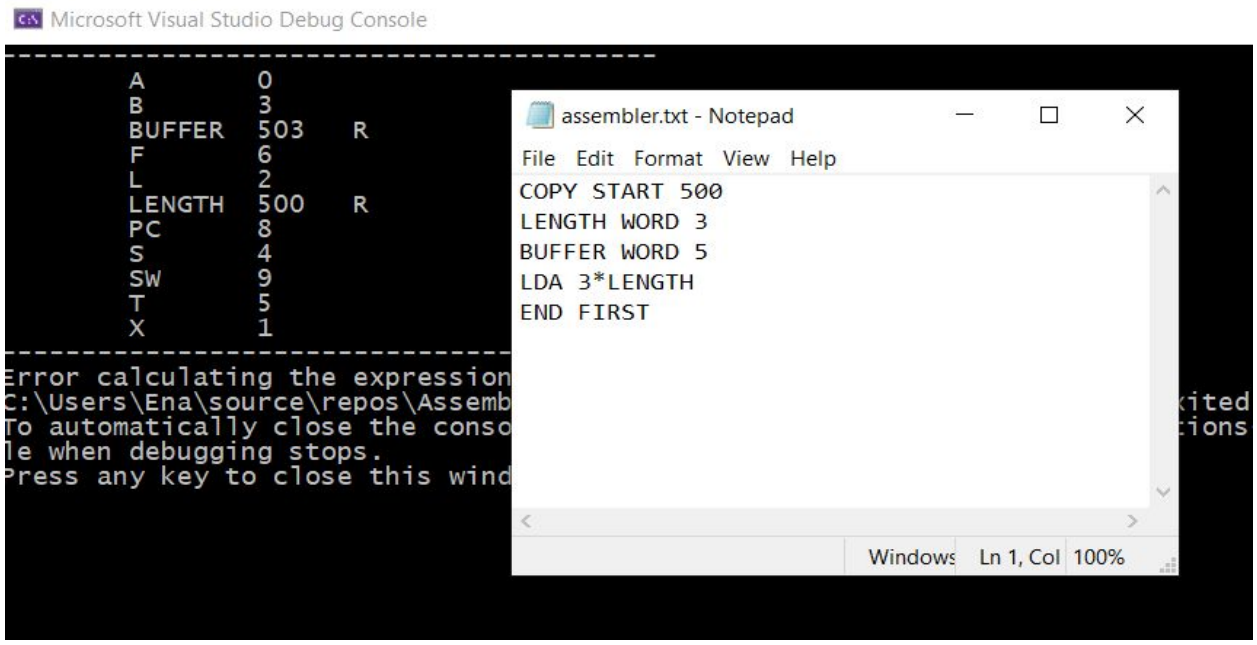


- EQU

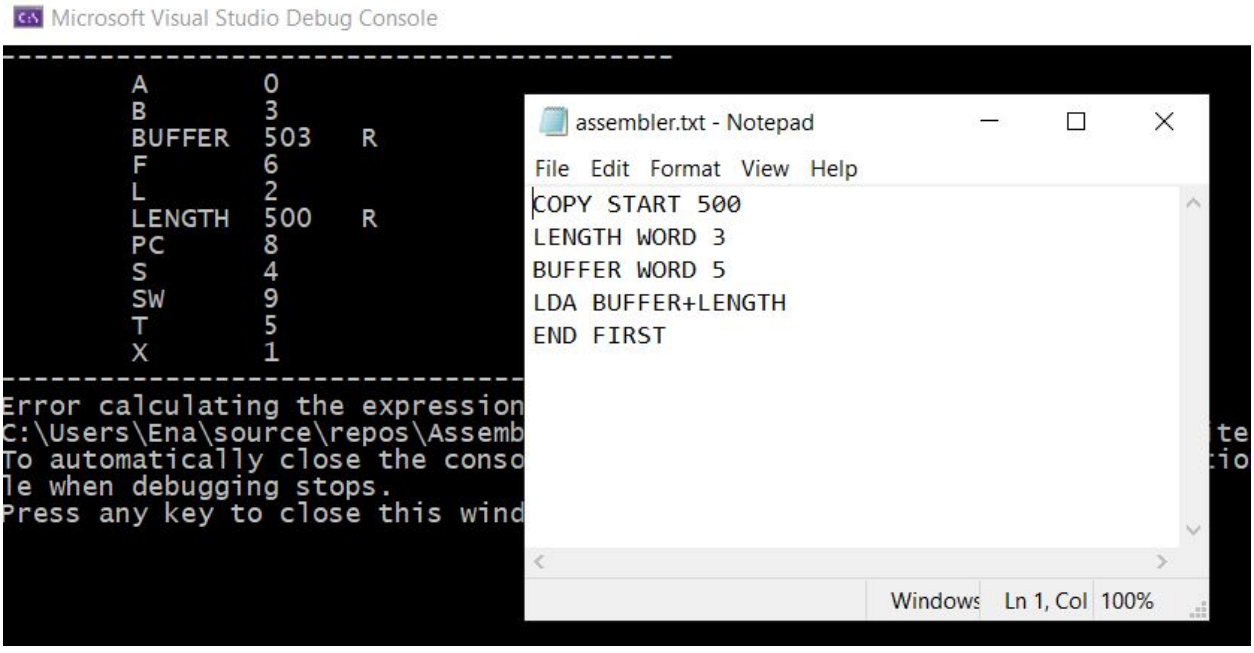


Expressions:

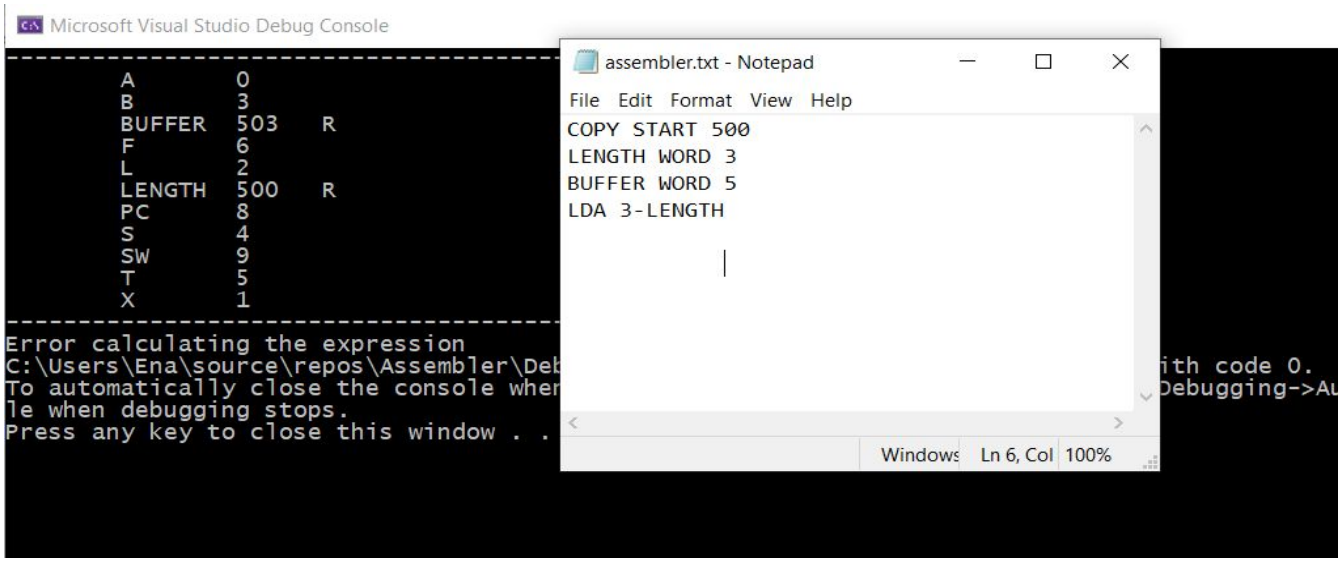
➢ Not allowed to use relative in multiplication or division.



➢ Relative+relative =ERROR



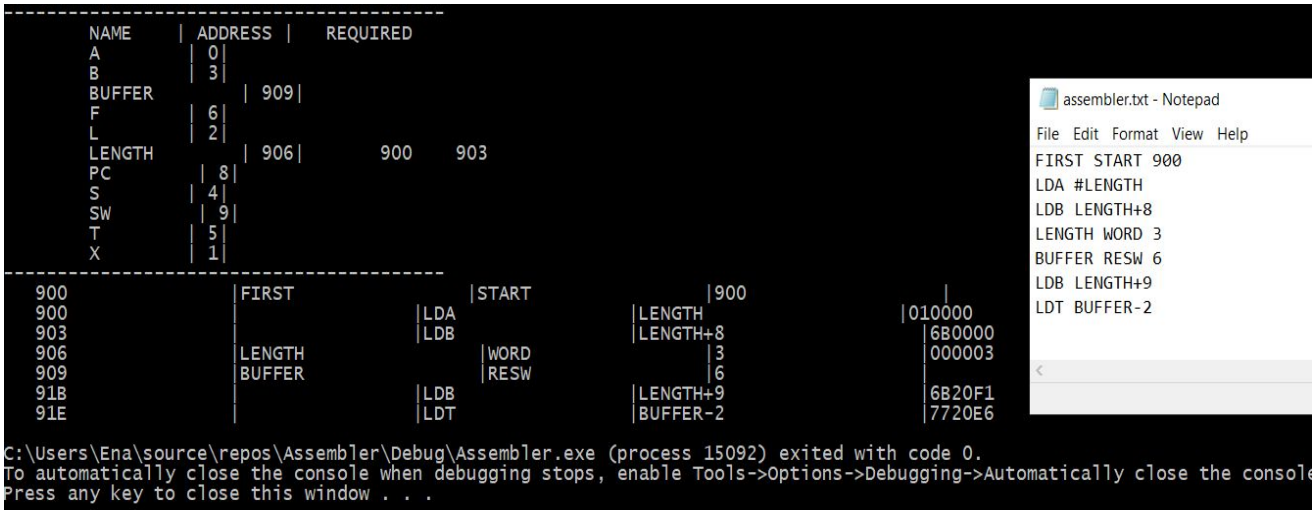
➢ Absolute-Relative=ERROR



➢ Right expression.

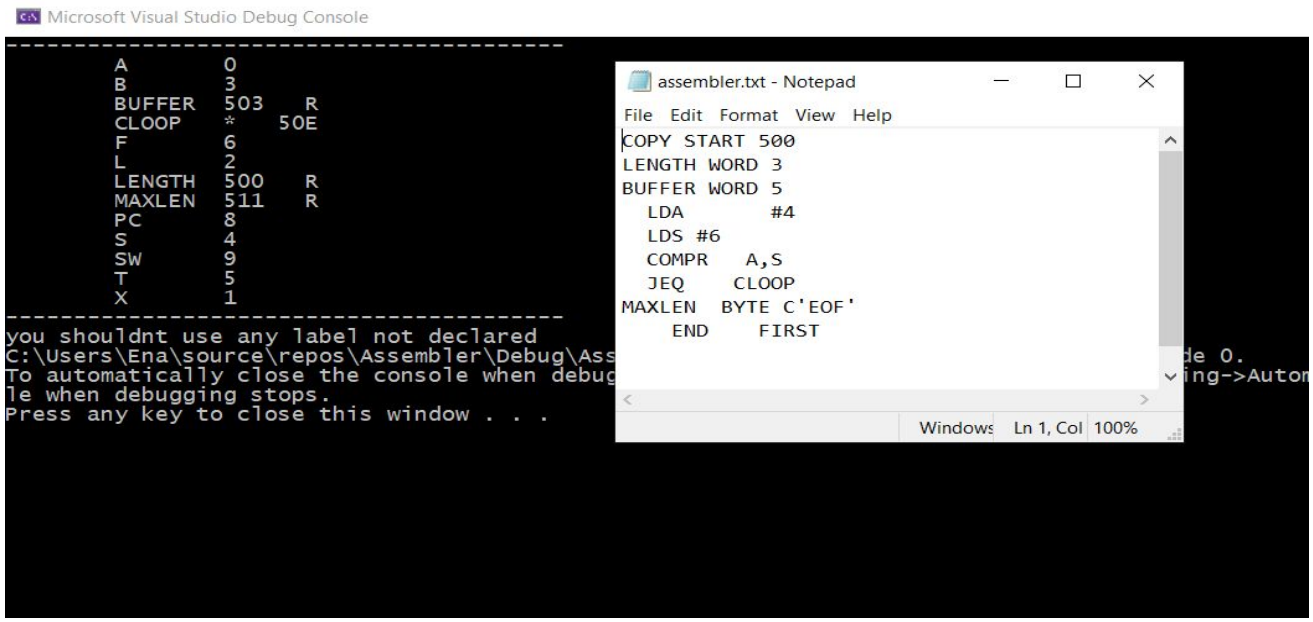
If the label is not defined, put zeros as in line 2,3.

Otherwise calculate it as in line 6,7.

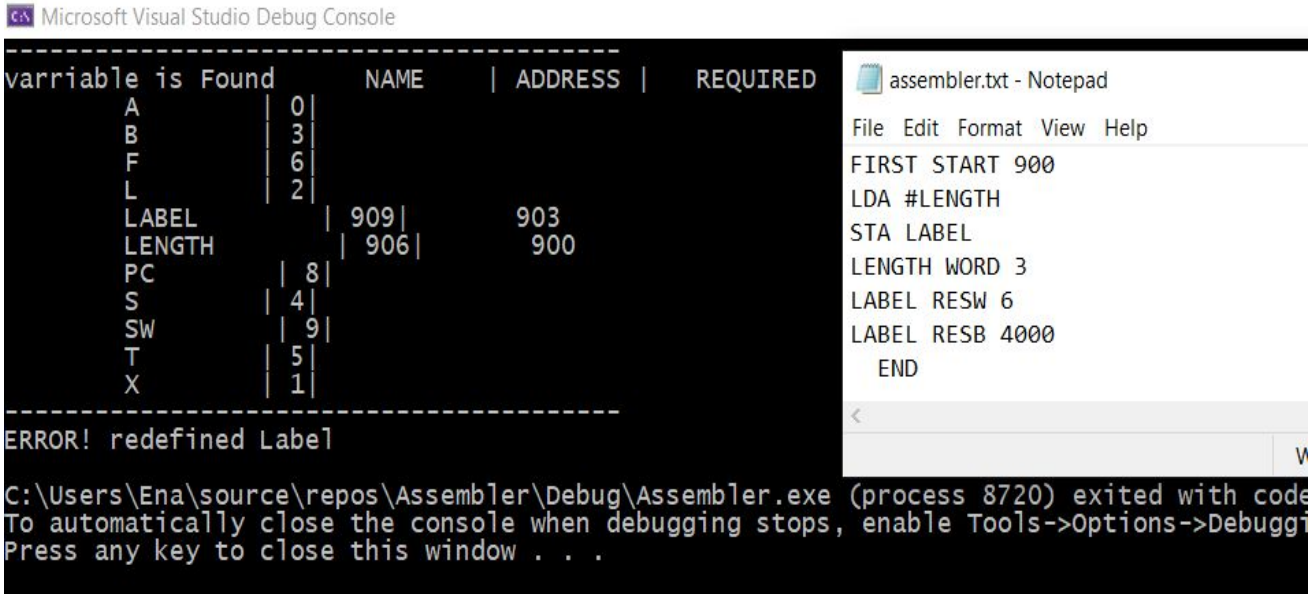


UNDEFINED LABEL:

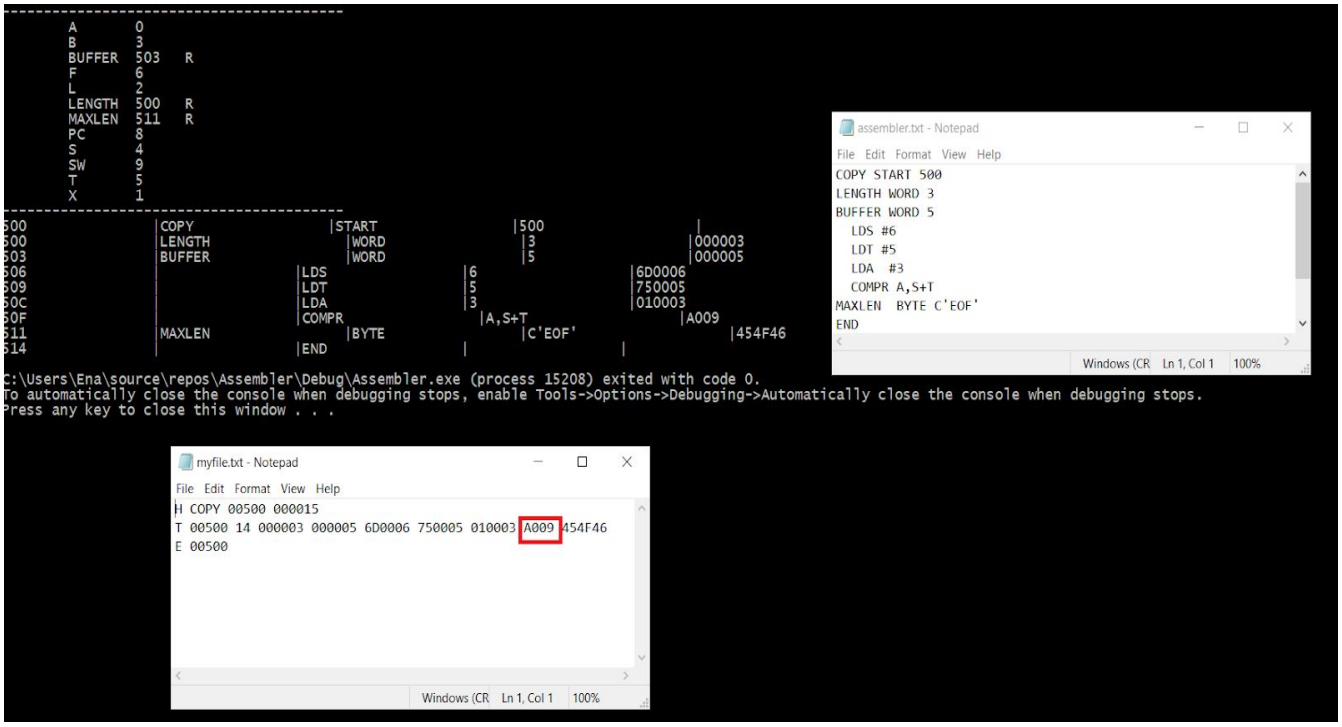
Cloop is used in line 7 but the program is ended without defining it.

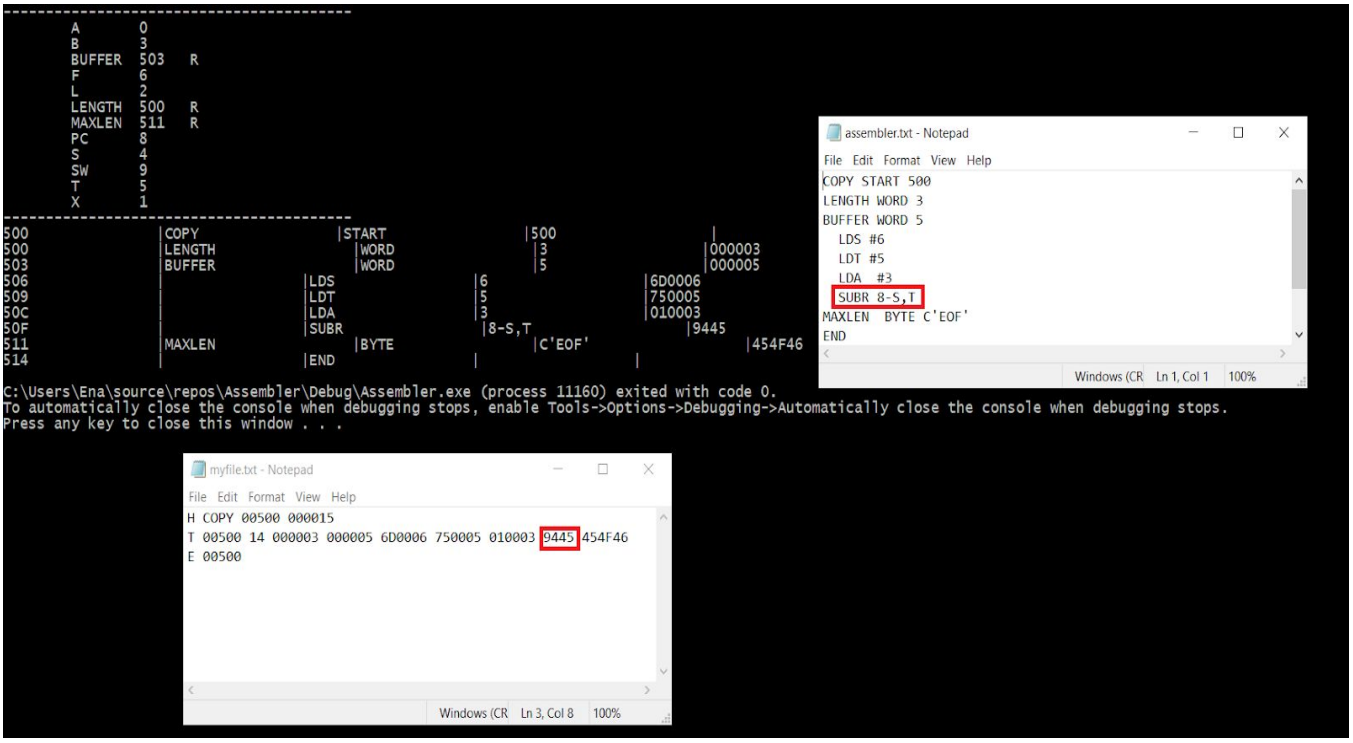


REDEFINED LABEL: LABEL is redefined.



Sample Run on Format2:





CORRECT SAMPLE RUNS: