Systems Programming

Final Project – SIC/XE Assembler

Phase 1

Prepared by:

1. Eman Rafik ID:11
2. Toka Alaa ID:14
3. Nada Salama ID:55
4. Yomna Gamal ID:60

➢ Requirements Specifications:

Implementing pass 1 of two pass SIC/XE assembler with the following specifications:

1. Building a parser to parse each of source code lines and capable of:
   - ✓ Decoding 2, 3 and 4-byte instructions as follows:

     a) 2-byte with 1 or 2 symbolic register reference (e.g., TIXR A, ADDR S,A)

     b) RSUB (ignoring any operand or perhaps issuing a warning)

     c) 3-byte PC-relative with symbolic operand to include immediate, indirect, and indexed addressing.

     d) 3-byte absolute with non-symbolic operand to include immediate, indirect, and indexed addressing.

     e) 4-byte absolute with symbolic or non-symbolic operand to include immediate, indirect, and indexed addressing.

   - ✓ Handling all storage directives (BYTE, WORD, RESW, and RESB).
   - ✓ Handling comment lines.
   - ✓ Ignoring unimplemented directives and operations with a warning.

2. The output of this phase is:

   - The symbol table.
   - The source program code with location counter before each line.
   - A meaningful error message printed below the line in which the error occurred.

❖ Extra Features:

1. The parser is capable of parsing free format lines.
2. The program deals with literals and print them in the output source code.
3. Expressions are allowed as operands for EQU directive.

➢ Design:

The assembler source code is mainly composed of six main classes:

- **Class Program:**
  It is a singleton and the main class in the program having reference to all other main classes.
  It saves the program attributes (ex: name, starting address, …).
  It has pass1 method which from which pass 1 launches and it controls the program till the end of pass 1.
- **Class Address:**
  It deals with location counter and update it throughout the program.
  It has three methods:
  1. startCounter: to initialize the location counter to certain position.
  2. updateCounter: to update counter after each line according to its format.
  3. getAddress: to get the current location counter.
- **Class SymbolTable:**
  It deals with symbol table and add labels to the table whenever found throughout the program.
  It has one parameter: map symbtable to save the symbol table.
  Its main methods are:
  1. addlabel: to add label to the map and return a string of error if occurred (empty string in case no error).
  2. getSymTab: to get a reference to the symbol table map.
- **Class LiteralTable:**
  It deals with literals throughout the program.
  It has a main parameter: litTab map to save the literal table.
  It has two methods:
  1. addLiterals: to add literals to the map.
  2. setLiterals: to be called at LTORG or END to set address to previous literals.
- **Class Parser:**
  It is responsible to parse each line of the input code.
  It has three main functions:
  1. parseStart: to parse line with start directive if found at the beginning of the input program.
  2. parseLine: to parse each input line of the input file.
  3. parseEnd: to parse the end line of the program.

- **Class Line:**
  Saves the parameters of each line after parsing.
  It has parameters with all fields of the line, in addition to string to for the error if occurred, and two boolean parameters to determine if there is an error and if the line is a comment line.
  It has two main methods:
  1. executer: to perform line updates (ex: update counter, …).
  2. write: to write the line to output file.

  In addition to some helping functions (ex: method to handle special cases of operation codes, …)

<u>In addition, there are three other helping classes:</u>

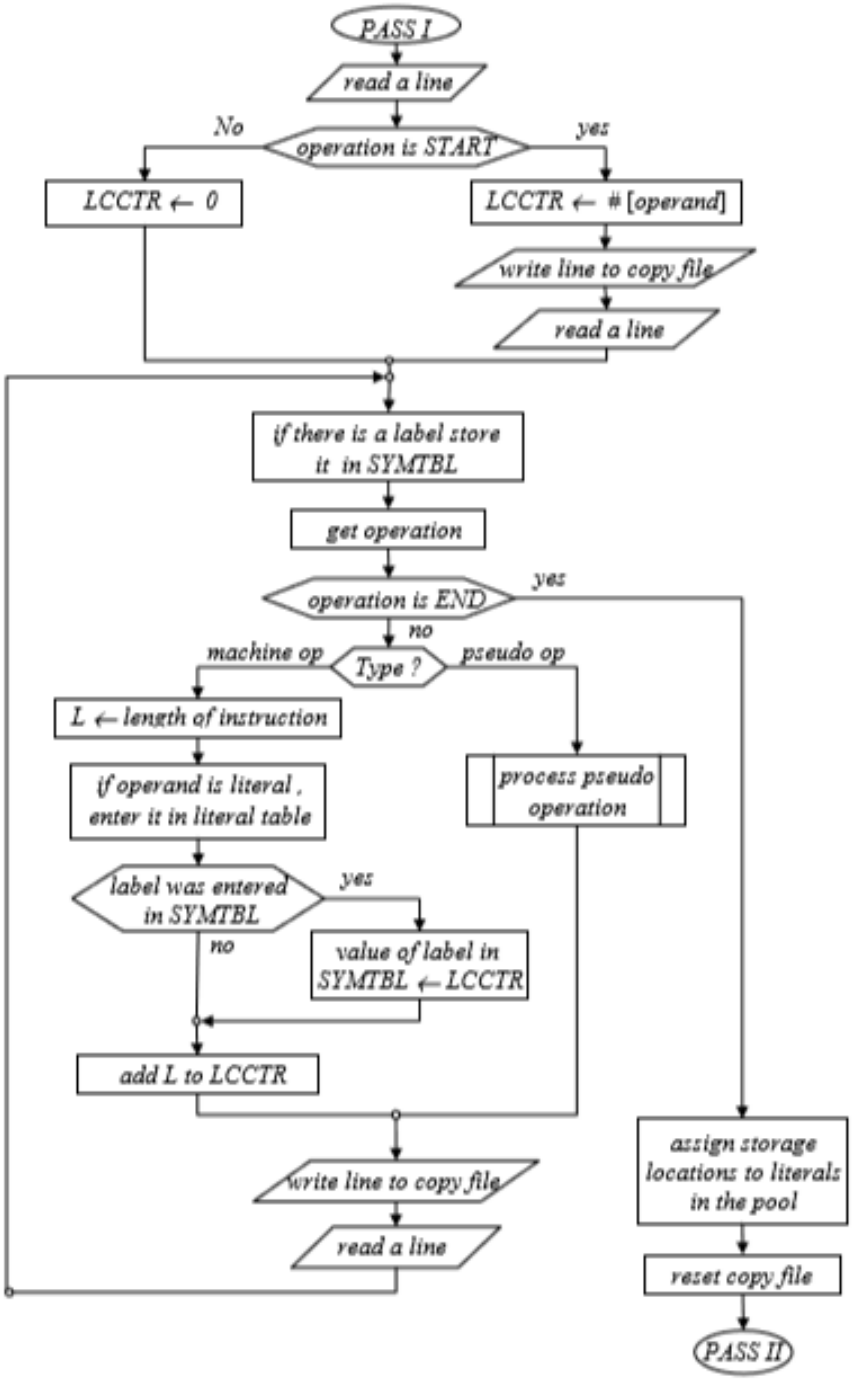
- **Class Output:**
  Responsible for writing the output file.
  It has two main methods:
  1. makeLine: to re-format the fields of the line and print it to the output file.
  2. printSymbolTable: to print the symbol table at the end of output file.

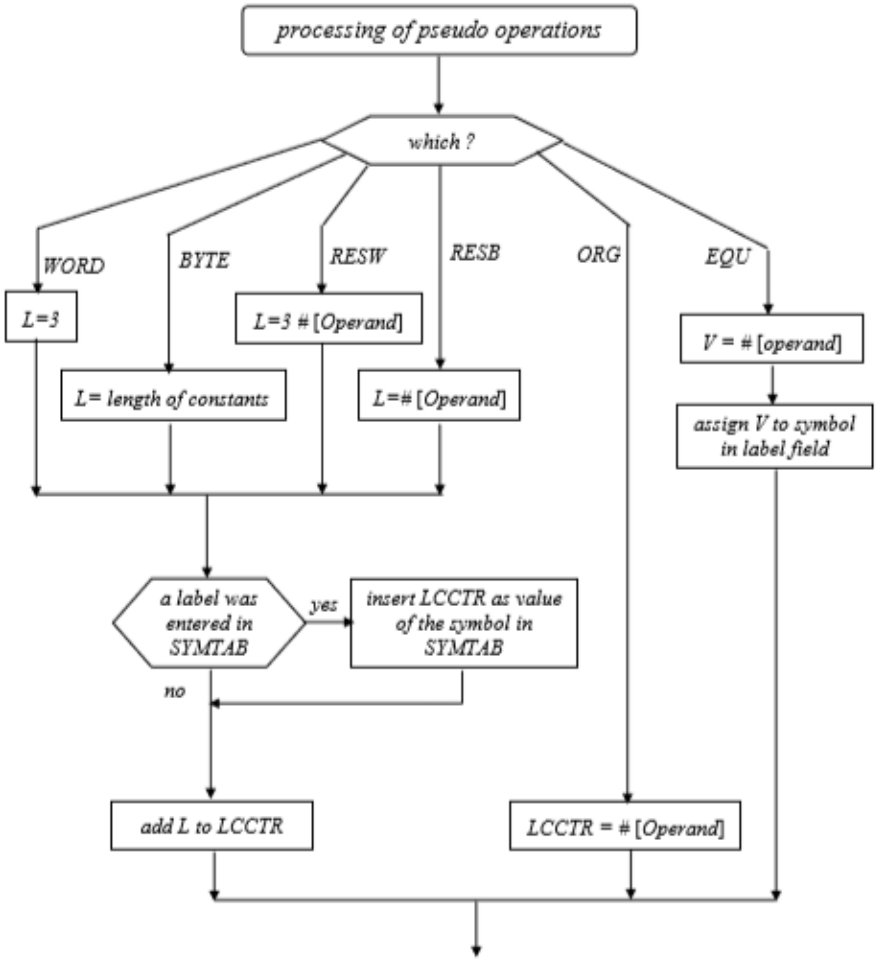
  In addition to a helping function (writeLine) to print a line immediately to the output file.
- **Class FormatChecker:**
  Responsible to check and validate format of operands according to the operation code.
  It has four main methods:
  1. formatThree:
  2. formatTwo: to check operands of format two and return a boolean accordingly.
  3. storageDirectives: to check format of storage directives and return a Boolean accordingally.
- **Class OperationCodes:**
  Save all operation codes with the expected format of operands and also save directives to validate the operation code of each line.

➢ Main Data Structures:

1. Three maps for symbol table, literal table and operation codes.
2. Two arrays for storage directives and assembler directives.

➢ Algorithms Description:





➢ Assumptions:

1. Each code line must all be entered in an individual line.
2. The program supports free format. i.e. Fields are not restricted to specific position in the line.
3. Comment of a line must be preceded by (.).
4. Any comment line must be preceded by(.) at the very beginning of the line.
5. Labels and operands must much the naming conventions (begins with a letter and contains only letters, numbers and (_).
6. Operands length are not limited, however if they exceed the memory allowed an error will occur during running the program.
7. Output is printed in a text file named “Output.txt” in a path relative to the executable file of the program.
8. Labels and operands are illegal to be one of the assembly keywords.

➢ Sample Runs:

Sample 1:

- Input:

trial.txt - Notepad

File Edit Format View Help

```
.2345678901234567890
START   0000
  LDA  #0
         STA      DIGIT
LOOP     TD       DEVICE
         JEQ      LOOP
.        J        ENDL .comment
START RD DEVICE
       SUBT     #48
     MUL      #10
       STA      DIGIT
WLOOP    TD       DEVICE
JEQ     WLOOP
.        J        ENDL
         ADD      DIGIT
   STA      DIGIT
ENDL     LDA      DIGIT
    J        *
.
DEVICE   BYTE     X'F3'
DIGIT    RESW     1
         END
```

- Output:

Output.txt - Notepad

File Edit Format View Help

```
Address     label    OPCode  Operand            Comment
       .2345678901234567890
0000                 start
0000                 LDA     #0
0003                 STA     DIGIT
0006        LOOP     TD      DEVICE
0009                 JEQ     LOOP
0009   .        J       ENDL .comment
000c        START    RD      DEVICE
       SUBT    #48
        Undefined operation code ..

000f                 MUL     #10
0012                 STA     DIGIT
0015        WLOOP    TD      DEVICE
0018                 JEQ     WLOOP
0018   .        J       ENDL
001b                 ADD     DIGIT
001e                 STA     DIGIT
0021        ENDL     LDA     DIGIT
0024                 J       *
0024   .
0027        DEVICE   BYTE    X'F3'
0028        DIGIT    RESW    1

002b                 END
*******************************************************
Symbol    Location    type
device    0027        R
digit     0028        R
endl      0021        R
loop      0006        R
start     000c        R
wloop     0015        R
```

Sample 2:

- Input:

trial.txt - Notepad

File Edit Format View Help

```
.2345678901234567890
          START   0000
          LDA     #1
          STA     LENGTH
1LOOP      TDE     DEVICE1
          JEQ     LOOP
                  LDX     LENGTH
          RD      DEVICE1
                  COMP    ENDF
                  JEQ     ENDLOOP
          STCH            ALPHA,X
.          LDCH            ALPHA,X  comment line
          LDA     LENGTH
          ADD         #1
                  STA           LENGTH
          J           LOOP
ENDLOOP   TD          DEVICE2
          JEQ       ENDLOOP
                  LDX     LENGTH
          LDCH       ALPHA,X
                  WD      DEVICE2
          LDA                LENGTH
                  SUB       #1
          STA     LENGTH
.          LDA          LENGTH
                  COMP    #0
          JGT     ENDLOOP
          J       *
.
EndF      Word    4
DEVICE1   BYTE    X'F3'
DEVICE2   BYTE    X'04'
ALPHA     RESB    100
LENGTH    RESW    1
          END
```

- Output:

Output.txt - Notepad

File Edit Format View Help

```
Address     label    OPCode  Operand             Comment
       .2345678901234567890
0000                start
0000                LDA     #1
0003                STA     LENGTH
1LOOP     TDE     DEVICE1
Undefined opCode ..

0006                JEQ     LOOP
0009                LDX     LENGTH
000c                RD      DEVICE1
000f                COMP    ENDF
0012                JEQ     ENDLOOP
0015                STCH    ALPHA,X
0015   .         LDCH            ALPHA,X  comment line
0018                LDA     LENGTH
001b                ADD     #1
001e                STA     LENGTH
0021                J       LOOP
0024        ENDLOOP  TD      DEVICE2
0027                JEQ     ENDLOOP
002a                LDX     LENGTH
002d                LDCH    ALPHA,X
0030                WD      DEVICE2
0033                LDA     LENGTH
0036                SUB     #1
0039                STA     LENGTH
0039   .         LDA         LENGTH
003c                COMP    #0
003f                JGT     ENDLOOP
0042                J       *
0042   .
0045        EndF     Word    4
0048        DEVICE1  BYTE    X'F3'
0049        DEVICE2  BYTE    X'04'
004a        ALPHA    RESB    100
00ae        LENGTH   RESW    1

00b1                END
*****************************************************
.....................................................

Symbol    Location    type
alpha     004a        R
device1   0048        R
device2   0049        R
endf      0045        R
endloop   0024        R
length    00ae        R
```