Alexandria University
Faculty of Engineering
CC Program

Systems Programming
Spring 2019
Term Project

Final Term Project - SIC/XE Assembler
Phase 1

The term project is to implement SIC/XE assembler that produces code for the absolute loader used in the SIC/XE programming assignments.

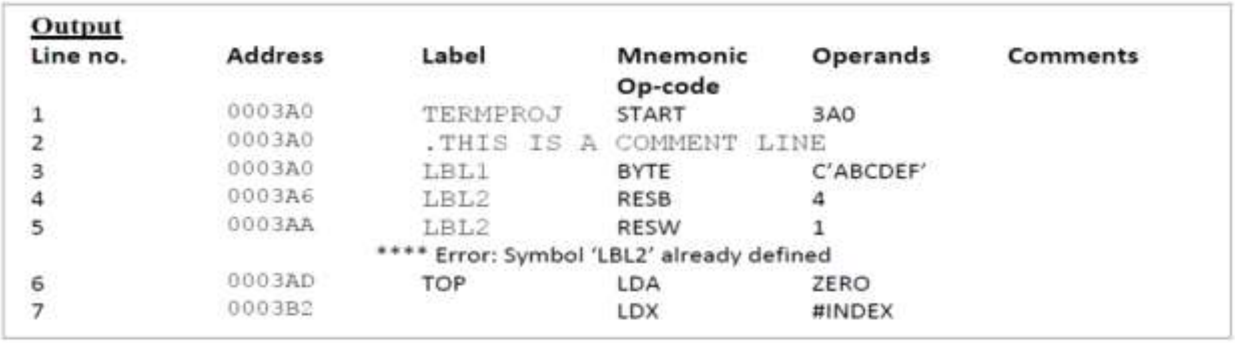
In phase 1 of the project, it is required to implement Pass1 of the assembler. The output of this phase should be used as input for subsequent phase.

Specifications

1. You should build a parser that is capable of handling source lines that are instructions, storage declaration, comments, and assembler directives (a directive that is not implemented should be ignored possibly with a warning)
2. For instructions, the parser is to minimally be 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
3. The parser is to handle all storage directives (BYTE, WORD, RESW, and RESB).
4. The output of this phase should contain (at least):
    a) The symbol table.
    b) The source program in a format similar to the listing file described in your text book except that the object code is not generated as shown below.
    c) A meaningful error message should be printed below the line in which the error occurred.

Sample Input

```
TERMPROJ START 3A0
.THIS IS A COMMENT LINE
LBL1 BYTE C'ABCDEF'
LBL2 RESB 4
LBL2 RESW 1
TOP LDA ZERO
LDX #INDEX
```



Output

```
Line no.    Address     Label       Mnemonic    Operands    Comments
                                    Op-code
1           0003A0      TERMPROJ    START       3A0
2           0003A0      .THIS IS A COMMENT LINE
3           0003A0      LBL1        BYTE        C'ABCDEF'
4           0003A6      LBL2        RESB        4
5           0003AA      LBL2        RESW        1
                    **** Error: Symbol 'LBL2' already defined
6           0003AD      TOP         LDA         ZERO
7           0003B2                  LDX         #INDEX
```

Bonus

Support free-formatted assembly language programs. In a free-formatted assembly program, statements are not restricted to begin at a given position in the line. Many consecutive white spaces or tabs should be treated as a single space. (You may use regular expressions)

Notes:

- You should work in groups of 4-5 members.
- All team members should work together. There is a grade on distributing the load evenly.
- All members should understand all components in the project, not just the parts they implemented.
- Cheating will be severely penalized. Both copies will be graded zero. So, delivering a partially functional implementation is much better than delivering a copy.
- You can use any language or your choice.

The implemented mini-language should include the following statements:

Statement		**Format**
RMO	r_1 , r_2	*2*
LDr	*m*	*3,4*
STr	*m*	*3,4*
LDCH	*m*	*3,4*
STCH	*m*	*3,4*
ADD	*m*	*3,4*
SUB	*m*	*3,4*
ADDR	r_1 , r_2	*3,4*
SUBR	r_1 , r_2	*2*
COMP	*m*	*3,4*
COMR	r_1 , r_2	*2*
J	*m*	*3,4*
JEQ	*m*	*3,4*
JLT	*m*	*3,4*
JGT	*m*	*3,4*
TIX	*m*	*3,4*
TIXR	r_1	*2*

Directive
START
END
BYTE
WORD
RESW
RESB
EQU
ORG
BASE

The assembler must detect the following errors:

error [01] : 'misplaced label'
error [02] : 'missing or misplaced operation mnemonic '
error [03] : 'missing or misplaced operand field '
error [04] : 'duplicate label definition '
error [05] : 'this statement can’t have a label '
error [06] : 'this statement can’t have an operand '
error [07] : 'wrong g operation prefix '
error [08] : 'unrecognized operation code '
error [09] : 'undefined symbol in operand '
error [10] : 'not a hexadecimal string'
error [11] : 'can’t be format 4 instruction’
error [12] : 'illegal address for a register '
error [13] : ‘missing END statement '

Deliverables:

- Source Code
- Report that contains:
    - Requirements specification.
    - Design
    - Main data structures
    - Algorithms description
    - Assumptions (if any)
    - Sample runs.
- You should submit the deliverables in a zipped file with the format:
  groupNumber_phase1.[rar/zip/...etc]. (for example: "1_phase1.rar")