Term Project (1)
SIC/XE Assembler

The project is to implement a 2-pass assembler for SIC/XE machine, written in **C/C++, JAVA, Python** producing code for the relocating 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. The pass1 is to execute by entering
pass1 <source-file-name>
2. The source file for the main program for this phase is to be named pass1.c
3. 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)
For instructions, the parser is to be minimally capable of decoding 1, 2, 3 and 4-byte instructions as follows:
a. 1-byte (e.g. NORM)
b. 2-byte with 1 or 2 symbolic register reference (e.g., TIXR A, ADDR S,A)
c. RSUB (ignoring any operand or perhaps issuing a warning)
d. 3-byte PC-relative with symbolic operand to include immediate, indirect, and indexed addressing
e. 3-byte absolute with non-symbolic operand to include immediate, indirect, and indexed addressing
f. 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), in addition to START and END directives.

- Hexadecimal addresses that would begin with 'A' through 'F' must start with a leading '0' to distinguish them from labels.
- Instructions and assembler directives in the source program may be written using either uppercase or lowercase letters.
- The source program to be assembled must be in fixed format as follows:

1. bytes 1–8 label
2. 9 blank
3. 10–15 operation code
4. 16–17 blank
5. 18–35 operand
6. 36–66 comment

• If a source line contains "." in the first byte, the entire line is treated as a comment

• A list of required instructions along with their op-codes are found in Slides of Lecture 1. (also in the text book)

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. A meaningful error message is printed below the line in which the error occurred.

Example 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 Op-code	Operands	Comments
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	

In phase 2 of the project

It is required to implement **Pass2** of the assembler. The output of this phase should contain the object code formatted as H T M E records where
H stands for Header record.
T stands for Text record.
M stands for Modification record.
E stands for End record.

Full description of these records is found in slides of lectures 2 and 3 (and as in the text book).

Note: You should handle the errors occurred in pass 1 and pass 2.

e.g. Duplicate labels,
displacement exceeds the limits in pc-relative and base-relative addressing mode, unknown instruction, etc.

Bonus

Handling literals