1.開發平台: Dev-C++(Windows 10)
2.使用開發環境: 整合式開發環境(IDE)
3.使用的程式語言: C++
4.所選擇的組合語言: SIC
5.程式設計:
(1)功能: SIC 的 Lexical Analysis，將原始程式轉成 Token 並且以指令、假指令、暫存器、符號、Symbol、地址和字串作為區分，並找出所在的分類代碼與序號。

(2)流程:
1.首先先將 Table1 至 Table4 以 vector 宣告動態陣列並儲存成資料庫，以方便後續比對。初始化 hashing table，給定 100 個位置。

2.輸入欲讀取的檔案名稱，一行一行依序讀取。

3.將一整行分解，先記錄空白的位置，並以空白作為間隔切割出字串。(文字與符號混在一起切割)

4.將這些字串依序存在一個 TokenList(vector)中，以便印出答案。

5.從 TokenList 中每一個元素依序搜尋是否含有符號。先找有無分號(若有分號，分號右側全部不用儲存)。若為撇號(')，則先找出下一個撇號位置，並把兩個撇號中間的字串放入 string table 中，並在 TokenList 中放入兩個撇號(不含字串)。每一個元素會搜尋 Table4 中所有符號，並分為三個情況: 1.符號在中間 2.符號在最左側 3.符號在最右側。優先判斷順序為 2->3->1。每一次皆把字串分為兩半，將新的一半插入 vector 中，含有兩個字元以上的字串繼續搜尋是否還有符號在內。
ex:
1.'123' -> ',123' -> ', 123, '
2.abc'123' -> abc'123, ' -> abc, '123, ' -> abc, ', 123, '
3.'123'abc -> ', 123'abc -> ', 123, 'abc -> ', 123, ', abc
4.abc'123'def -> abc, '123'def -> abc, ', 123' def
-> abc, ', 123, 'def -> abc, ', 123, ', def

6.在 TokenList 中每一個元素皆比對 Table1 至 Table4，若相符則記錄分別所在的 table 與序列，第一次若都不符合則會把所有小寫 a 至 z 轉為大寫(ASCII CODE 十進位減 32)，再搜尋一次。若不在 Table1 至 Table4，則需判別為 Symbol 或地址。

7.先確定是否為16進位，字尾需有H，且其餘只能有0至9和A至F。若不是16進位，則判斷是否為10進位，字串中只能含有數字0至9，若都不是則為Symbol。將這些字串加入對應的table中，並在TokenList紀錄對應的table與序列。
8.將TokenList的資料輸出建檔。

(3)使用的data structure: 動態陣列、hashing function、字串處理