**SIC XE Assembler**

**系統程式書面報告**

**資訊二丙 座號 : 82 學號 : D0588813 姓名 : 曾玉鳳**

**資訊二丙 座號 : 54 學號 : D0542900 姓名 : 董育汝**

**資訊二丙 座號 : 18 學號 : D0511203 姓名 : 陳品樺**

**資訊二丙 座號 : 21 學號 : D0511305 姓名 : 范瑋軒**

**Assembler 開發語言及平台 :** 使用Python

**處理步驟 :**

Step1 : 輸入檔案名稱，將檔案一行一行讀入，如果中間讀到tab，則將Tab error寫入Lisfile檔並停止程式，其他則透過位置將字串分割出標籤、opcode、operand，並使用Python .strip()將空格清除，最後儲存至list中。

Step2 : 區分出虛擬指令。我們使用if-else來分別出START、RESB、RESW、BYTE、WORD、END、ORG做額外處理，並計算位址加入最前方欄位，及判斷有無+、X的符號，如遇END就結束Step2。

遇到「.」註解行、「BASE」即省略。

遇到「START」設定起始位置，計算Location時以10進位運算，算完再轉至16進位。

遇到「ORG」，若其後面有接標籤，則位置直接將位址跳至標籤位置，並儲存原本位置，若其後沒標籤，則跳回原本PC位址，若上述都不符合則顯示 ORG指令錯誤。

遇到label，則另外存至sym={ } (dictionary)。

遇到「WORD」轉為16進位，直接計算WORD及BYTE的object code，並計算下一個指令位址(+3)。

遇到「BYTE」中的字元'C'，則將其轉換成16進位ASCII碼，並將下一位址增加其字元長度。而16進位數字'X'，則將其數字直接存入，如不滿1byte則補空格。

遇到「RESB」、「RESW」，增加空欄位到最後一行，並計算下一個指令位址加入欄位中。

遇+格式指令，計算下一個指令位址加入欄位中(+4)。

遇到有X的opcode，則切割至' , '前之opcode，並放入該位址第一位加8，轉成16進位，放入該位址後3位。

剩餘指令依op dictionary內的指令長度計算下一個指令位址加入欄位中。

若上述都不符合，則顯示 WRONG OPCODE!

Step3 : 計算其餘的object code加入欄位，如遇END就結束Step3。

忽略ORG、BASE、註解。

遇到「BASE」將B設為此標籤位址。

遇到+格式指令，若為「索引模式,X」，將切割至' , '前之opcode指令+3第三位設為9加上位址，若為「立即模式#」，且為#數字，將指令+1第三位設為1加上位址，若為#標籤，則需另編寫M卡片，將位址+1加到M陣列裡，若不為上述，則將指令+1第三位設為1加上位址。

遇到索引模式,X，則切割至' , '前之opcode，將指令+3第三位設為9加上位址。

若指令在op={ } (dictionary)裡。遇到「立即模式#」，判斷為#數字，將指令+1加上位址，若為#標籤，判斷範圍適於減PC，則將指令+1第三位設為2加上(位址-PC)，判斷範圍適於減B，則將指令+1第三位設為4加上(位址-B)，非上述print錯誤訊息。遇到「間接模式@」，判斷範圍適於減PC，則將指令+2第三位設為2加上(位址-PC)，判斷範圍適於減B，則將指令+2第三位設為4加上(位址-B)，非上述print錯誤訊息。

若為format 2，將切割至' , '前之opcode加上暫存器編號，非上述print錯誤訊息。

非為上述則為一般，將指令+3第三位設為2加上(位址-PC)，判斷範圍適於減B，則將指令+3第三位設為4加上(位址-B)，非上述print錯誤訊息。

Step4 : 將內容依格式讀出寫成LISTJFILE.txt檔並輸出。

Step5 : 將內容讀出寫成OBJFILE.txt檔並輸出。

輸出H卡片。

將上面編好的object code依序寫入T卡片，若長度不滿30，則繼續 。

寫入若超過或下行object code為空，則換一張T卡片繼續編寫

如果遇到「.」、空字串(第一行)，表示是註解、RESB、RESW，省略不寫入。

如果遇到END，則先將前一行T卡片未完成字串輸出，從M陣列裡輸出M卡片並加上修改位置05，再輸出E卡片。

**輸入格式:**

不可使用tab

英文大、小寫皆可輸入

Label、指令及operand起始位置有固定要求(1-9，10-17，18- )

不一定要有label

註解可新的一整行，不可放於後面

**可處理的addressing modes和assembler directives:**

addressing modes : Direct addressing、Indirect addressing、

Extend format、Immediate addressing、

Index addressing、Relative addressing

assembler directives: START、END、WORD、BYTE、RESW、RESB、

BASE、ORG

**Function :**

hex( ) : 十進制轉16進制

ord( ) : 將字元轉為ASCII

append ( ): 將東西新增在列表後方

insert( ) : 將指定對象插入列表的指定位置

zfill( ) : 將字串前方補0，使之達指定字串長度

upper( ) : 將小寫轉大寫

split( ) : 通過指定分隔符對字串進行切割

strip( ) : 去除字頭字尾指令字元(默認為空白鍵)

math.ceil( ) : 取整數，無條件進位

**data structures :**

**xe list結構 :** 很多個list，每個list包含多個小list

[ [[10進位 , 16進位] , label , opcode , operand , object code] ,[…..] , ……. ]

Ex. [[[8192, '2000'], 'HW1SIC', 'START', '2000', ''],

[[8192, '2000'], 'FIRST', 'LDA', 'AA', '00203C'],

[[8195, '2003'], '', 'ADD,X', 'BB', '18A03F'],

[[8198, '2006'], '', 'STA,X', 'CC', '0CA042'],……]

**sym結構 :** {‘label名稱’:’location’,…… }

Ex. {'AA': '203C',

'BB': '203F',

'CC': '2042',

'DD': '2045',

'EE': '2048',

….. }

**op結構 :** {‘指令名稱’: [‘opcode’,’’] }

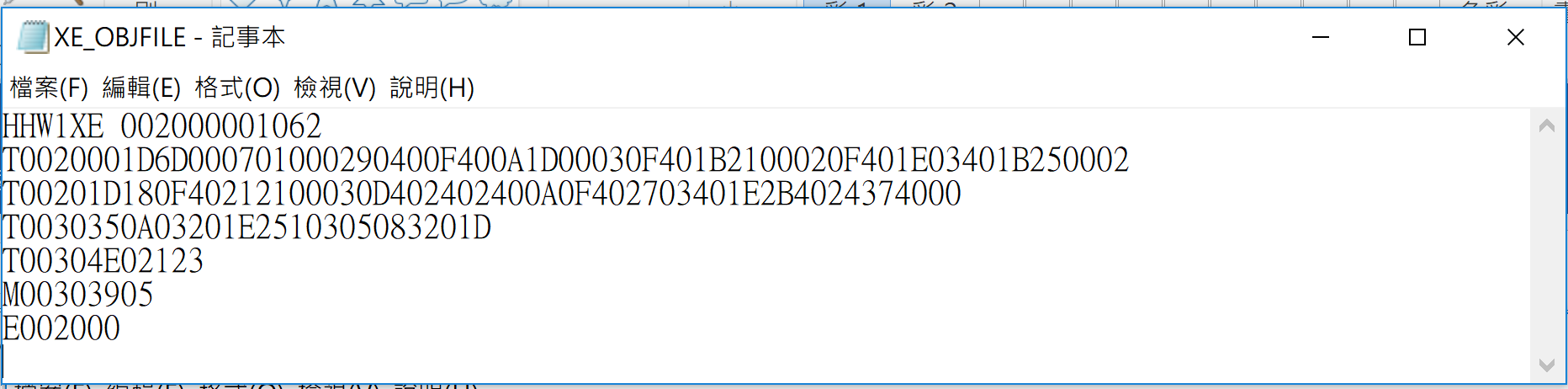
Ex. op={'ADD':['18',3],'ADDF':['58',3],'ADDR':['90',2],'AND':['40',3],……..}

**reg結構 :** {‘暫存器’: ‘暫存器編號’ }

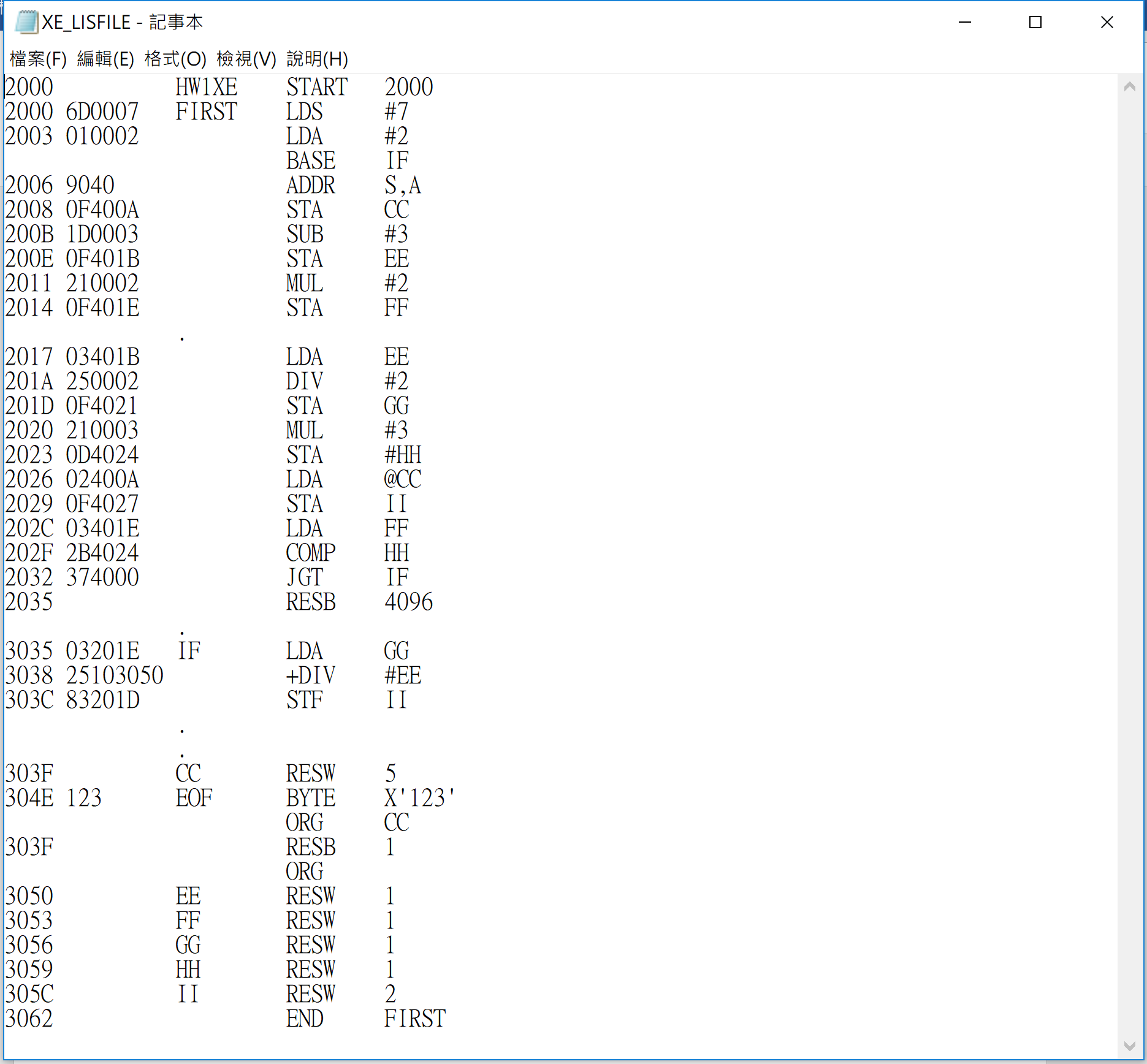
Ex.reg={'A':'0','X':'1','L':'2','B':'3','S':'4','T':'5','F':'6'}

**輸出結果 :**

OBJFILE :



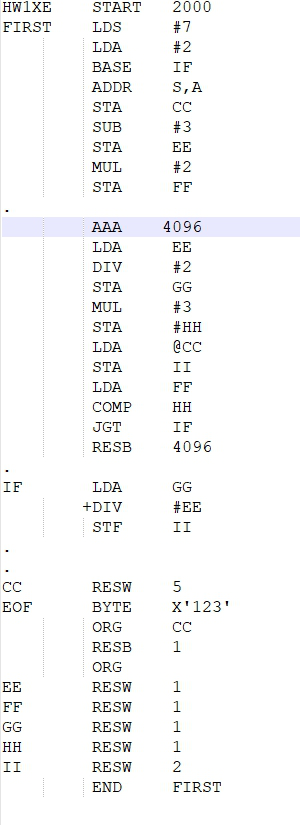
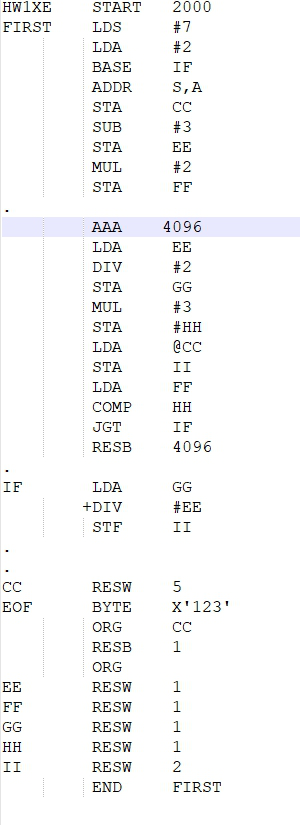
LISFILE :

****

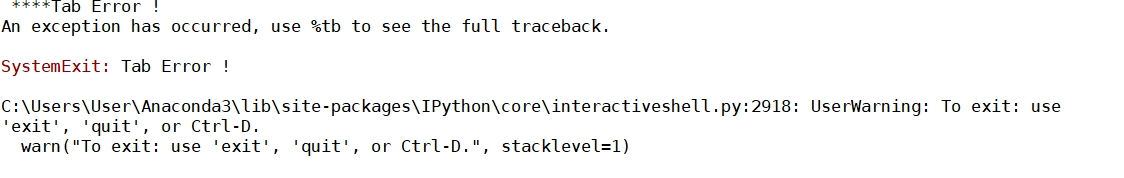
**錯誤訊息範例 :**

**Tab Error :**

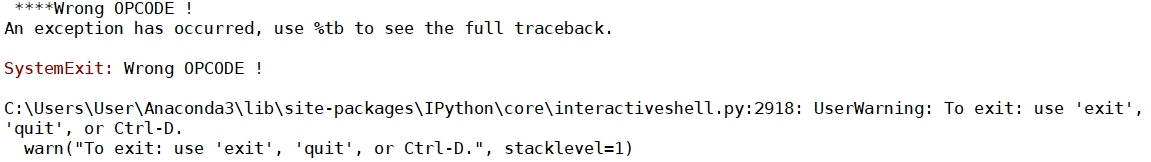
-測試程式碼 :



-測試結果 :

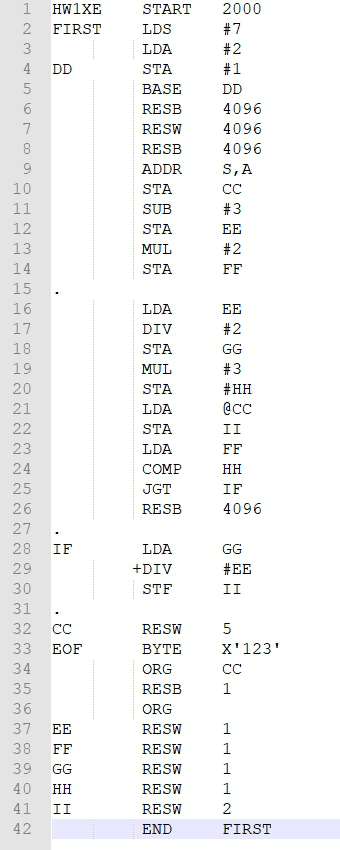
**Wrong Opcode :**

-測試結果 :



**Relative error :**

-測試程式碼 :



-測試結果 :

