SIC Assember

資訊二丙　82　D0588813　曾玉鳳

資訊二丙　54　D0542900　董育汝

資訊二丙　18　D0511203　陳品樺

資訊二丙　21　D0511305　范瑋軒

Assembler 開發語言及平台： 使用 Python

處理步驟：

Step1：將檔案讀入，使用 Python 分割字串的特定語法，將字串分割並存取，如果中間讀到 tab，則將 tab error 寫入 Lisfile 檔並停止程式

Step2：區分出虛擬指令，我們使用 if-else 來分別出 START、RESB、RESW、BYTE、WORD 及 END，計算 Location 時以 10 進位運算，算完再轉至 16 進位，如遇到 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。

若遇到有 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。

Step3：將其餘的 object code 編出，

Step4：將內容讀出寫成 LISFILE.txt 檔並輸出，若有錯誤，亦會顯示於內

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，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

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

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 :

sic list : 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 dict. : {`label 名稱’ : `location’ ,..... }

Ex. {'AA': '203C',
'BB': '203F',
'CC': '2042',
'DD': '2045',
'EE': '2048',
..... }

輸出格式範例:

```
========== RESTART: D:\課程\SIC.py ==========
輸入檔案名稱(SIC.txt): SIC.txt

輸入之SIC程式碼:
HW1SIC    START   2000
FIRST     LDA     AA
          ADD,X   BB
          STA,X   CC
          SUB     DD
          STA     EE
          MUL     AA
          STA     FF
.
          LDA     EE
          DIV     AA
          STA     GG
          MUL     THREE
          STA     HH
          LDA     ZERO
          STA     II
          LDA     FF
          COMP    HH
          JGT     IF
.
IF        LDA     GG
          DIV     SIX
          STA     II
.
.
AA        WORD    2
BB        WORD    7
CC        RESW    1
DD        WORD    3
EE        RESW    1
FF        RESW    1
GG        RESW    1
HH        RESW    1
II        RESW    1
THREE     WORD    3
SIX       WORD    6
ZERO      WORD    0
          END     FIRST

輸出之LISFILE:
2000        HW1SIC    START   2000
2000 00203C FIRST     LDA     AA
2003 18A03F           ADD,X   BB
2006 0CA042           STA,X   CC
2009 1C2045           SUB     DD
200C 0C2048           STA     EE
200F 20203C           MUL     AA
2012 0C204B           STA     FF
                .
2015 002048           LDA     EE
2018 24203C           DIV     AA
201B 0C204E           STA     GG
201E 202057           MUL     THREE
2021 0C2051           STA     HH
2024 00205D           LDA     ZERO
2027 0C2054           STA     II
202A 00204B           LDA     FF
202D 282051           COMP    HH
2030 342033           JGT     IF
                .
2033 00204E IF        LDA     GG
2036 24205A           DIV     SIX
2039 0C2054           STA     II
                .
                .
203C 000002 AA        WORD    2
203F 000007 BB        WORD    7
2042        CC        RESW    1
2045 000003 DD        WORD    3
2048        EE        RESW    1
204B        FF        RESW    1
204E        GG        RESW    1
2051        HH        RESW    1
2054        II        RESW    1
2057 000003 THREE     WORD    3
205A 000006 SIX       WORD    6
205D 000000 ZERO      WORD    0
2060                  END     FIRST

輸出之OBJFILE:
HHW1SIC002000000060
T0020001E00203C18A03F0CA0421C20450C204820203C0C204B00204824203C0C204E
T00201E1E2020570C205100205D0C205400204B282051342033000204E24205A0C2054
T00203C06000002000007
T00204503000003
T0020570900000300000600000O
E002000
```

程式碼:

```
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 30 13:24:11 2018

@author: YURU
"""
import math
import sys

sic=[]  #程式碼儲存
op={'ADD':['18',3],'AND':['40',3],'COMP':['28',3],'DIV':['24',3],'FIX':['C4',1],
    'J':['3C',3],'JEQ':['30',3],'JGT':['34',3],'JLT':['38',3],'JSUB':['48',3],
    'LDA':['00',3],'LDCH':['50',3],'LDL':['8',3],'LDX':['4',3],'MUL':['20',3],'OR':['44',3],'RD':['D8',3],'RSUB':['4C',3],
    'STA':['0C',3],'STCH':['54',3],'STL':['14',3],'STX':['10',3],'SUB':['1C',3],'TD':['E0',3],'TIX':['2C',3],'WD':['DC',3]}
sym={}  #SYMTAB
locctr=0

file=input('輸入檔案名稱(SIC.txt): ')
#file='SIC.txt'
srcfile=open(file)  #開檔
lisfile=open(file.split('.')[0]+'_LISFILE.txt','w')
objfile=open(file.split('.')[0]+'_OBJFILE.txt','w')

print('\n輸入之SIC程式碼:')
for line in srcfile:
    print(line,end='')
    if '\t' in line:  #需增加錯誤行
        print(' ****Tab error !')
        lisfile.write(' ****Tab error !\n')
        sys.exit('Tab error !')
    if '.' in line:
        sic.append(['.',line.split('.')[1].strip()])
    else:
        sic.append([line[0:9].strip().upper(),line[9:17].strip().upper(),line[17:].strip().upper()])  #分割儲存字串
print()
#print(sic)

for code in sic:  #計算位址
    if code[0]=='.':  #省略註解行
        continue
    if code[1]=='START':  #設定起始位置
        locctr=int(code[2],16)  #將16進位轉10進位
        code.insert(0,[locctr,str(hex(locctr))[2:].zfill(4).upper()])  #將位址(10&16進位)差在List最前方
        code.append('')
        continue
    code.insert(0,[locctr,str(hex(locctr))[2:].zfill(4).upper()])  #將前一次加完之位置存在陣列前方
    if code[1].strip() and code[1]!='.':  #如果標籤不為空，儲存至SYMTAB
        sym[code[1]]=code[0][1]
    if code[2]=='END':  #遇到END則結束
        code.append(sym[code[3]].zfill(6))
        break
    elif code[2]=='WORD':  #直接寫出opcode並加在List最後面
        code.append(str(hex(int(code[3])))[2:].zfill(6).upper())  #轉換數字成16進位並補0
        locctr+=3  #每個WORD位址加3
    elif code[2]=='BYTE':  #直接寫出opcode並加在List最後面
        if code[3][0]=='C':  #將字元轉換成ASCII hex(ord())
            c=''
            for x in code[3][2:-1]:  #字元一一轉換
                c+=str(hex(ord(x))[2:])
            code.append(c)
            locctr+=len(code[3][2:-1])
        elif code[3][0]=='X':
            locctr+=math.ceil(len(code[3][2:-1])/2)  #math.ceil 無條件進位
    elif code[2]=='RESW':
        code.append('')
        locctr+=3*int(code[3])
    elif code[2]=='RESB':
        code.append('')
        locctr+=int(code[3])
    elif code[2].find(',X') != -1:  #有X之OPCODE
            if code[2].split(',')[0] in op:  #切割至','前
                locctr+=op[code[2].split(',')[0]][1]
    elif code[2] in op:
        locctr+=op[code[2]][1]
    else:
        print(code,'error')

for code in sic:
    if code[0]=='.':  #省略註解行
        continue
    if code[2].find(',X') != -1:  #有X之OPCODE
        if code[2].split(',')[0] in op:
            #切割至','前之OPCODE + 放入之位址第一位加8並轉成16進位 + 加入放入之位址後3位
            code.append(op[code[2].split(',')[0]][0]+hex(int(sym[code[3]][0])+8)[2:].upper()+sym[code[3]][1:])
    if code[2] in op:
            code.append(op[code[2]][0]+sym[code[3]]) #切割至','前之OPCODE + 加入放入之位址

'''
for code in sic:
    print(code)
'''

print('\n輸出之LISFILE: ')
```

```
    if code[0]=='.':  #顯示整行註解
        lisfile.write(' '*12+'.'+code[1]+'\n')
        print(' '*12+'.'+code[1])
    elif code[2]=='END':
        lisfile.write('{0:<4}'.format(code[0][1])+' '*8+'{0:<9}{1:<8}{2}\n' .format(code[1],code[2],code[3]))
        print('{0:<4}'.format(code[0][1])+' '*8+'{0:<9}{1:<8}{2}' .format(code[1],code[2],code[3]))
    else:
        lisfile.write('{0:<4} {1:<6} {2:<9}{3:<8}{4}\n' .format(code[0][1],code[4],code[1],code[2],code[3]))
        print('{0:<4} {1:<6} {2:<9}{3:<8}{4}' .format(code[0][1],code[4],code[1],code[2],code[3]))

print('\n輸出之OBJFILE: ')
length=str(hex(sic[-1][0][0]-sic[0][0][0]))[2:].zfill(6).upper()  #計算總長度
#輸出 H卡片
print('H'+'{0:<6}'.format(sic[0][1])+sic[0][3].zfill(6)+length)
objfile.write('H'+'{0:<6}'.format(sic[0][1])+sic[0][3].zfill(6)+length+'\n')

objf=[]
l=-1
t=''
#輸出 T,E卡片
for i,code in enumerate(sic):
    if code[0]=='.' or code[4]=='':  #省略註解行 及 RESW RESB
        continue
    if code[2]=='END':  #遇到END輸出E卡片
        if t!='':  #前一行有剩餘未完成字串則輸出
            objfile.write(str(hex(l))[2:].zfill(2).upper()+t+'\n')
            print(str(hex(l))[2:].zfill(2).upper()+t)
            t=''
        objfile.write('E'+sym[code[3]].zfill(6)+'\n')
        print('E'+sym[code[3]].zfill(6))
        break
    elif l==-1:  #T卡片開頭
        objfile.write('T'+code[0][1].zfill(6))
        print('T'+code[0][1].zfill(6),end='')
        t+=code[4]
        l+=1+math.ceil(len(code[4])/2)
    elif l<=30: #卡片長度不超過30則繼續寫入
        l+=math.ceil(len(code[4])/2)
        t+=code[4]
    while sic[i+1][0]=='.':  #遇到註解行跳過
        i=i+1
    if math.ceil(len(sic[i+1][4])/2)+l>30 or sic[i+1][4]=='':  #如果加入下行object code即超過30 or 下行object code為空 則換卡片
        objfile.write(str(hex(l))[2:].zfill(2).upper()+t+'\n')
        print(str(hex(l))[2:].zfill(2).upper()+t)
        t=''
        l=-1
```