

# 系統程式期中作業

## SIC assembler

李奕承 611121212

## 目標概述

這個作業的目標是要做出一個組譯器，它可以讀取一個用SIC組合語言寫成的程式，並將它翻譯成object code存成另一個檔案。

## 運作原理

建立結構，暫存每行object code的值

建立結構，存放指令和它的opcode，用來查詢

建立鏈結串列的head，用來存放掃描到的標籤及它的地址

開檔、讀取

第一次掃描:

算出每行的地址

紀錄下有定義標籤的指令所在地址(使用鏈結串列)

重置檔案資料流指針回到開頭

第二次掃描:

將讀取到的字串與指令做比對，若查到吻合的指令，就將其opcode寫入object code暫存結構中的opcode成員

將不是指令的部分與標籤鏈結串列比對，若吻合就將其地址寫入object code暫存結構中的address成員

將不是指令的部分與標籤鏈結串列比對，若吻合就將其地址寫入object code暫存結構中的address成員

判讀剩下的部分，若有 X 模式

則把address成員加上32768(二進位是 1000 0000 0000 0000 )

判讀剩下的部分，若有資料

則把資料轉化為16進位顯示

寫入、關檔

釋放紀錄標籤的鏈結串列

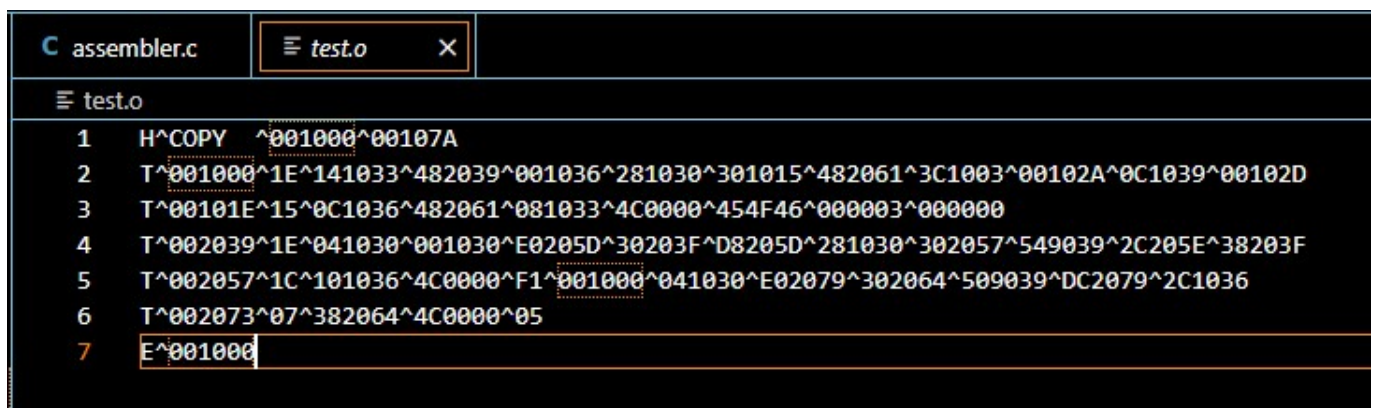
## 編譯測試

```
Lyciih@DESKTOP-CR5NUFU MINGW64 ~/Desktop/github/System-Programming/SIC assembler (main)
$ make
gcc assembler.c -o assembler -Wall -L . -l linked_list

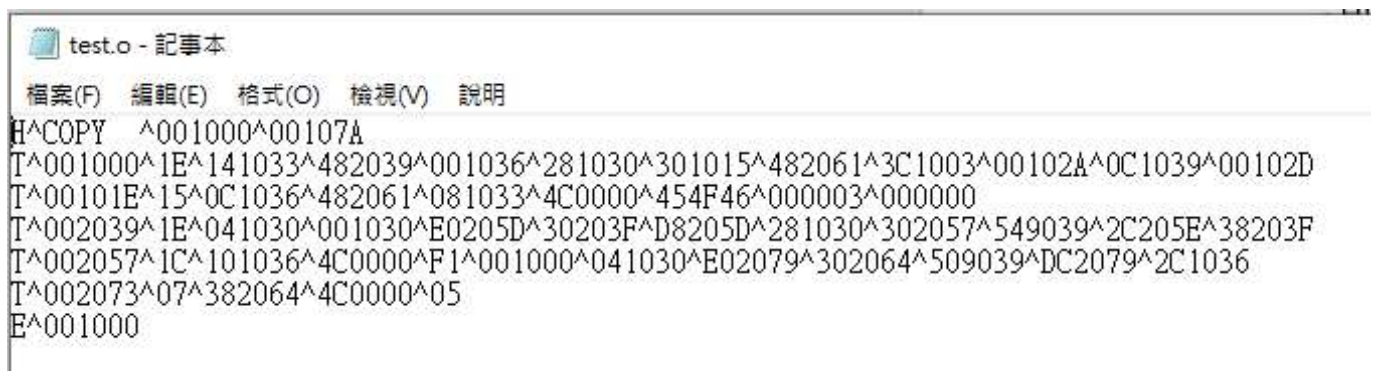
Lyciih@DESKTOP-CR5NUFU MINGW64 ~/Desktop/github/System-Programming/SIC assembler (main)
$ make run
[1] [source] test.s
[1] [target] test.o

H^COPY ^001000^00107A
T^001000^1E^141033^482039^001036^281030^301015^482061^3C1003^00102A^0C1039^00102D
T^00101E^15^0C1036^482061^081033^4C0000^454F46^000003^000000
T^002039^1E^041030^001030^E0205D^30203F^D8205D^281030^302057^549039^2C205E^38203F
T^002057^1C^101036^4C0000^F1^001000^041030^E02079^302064^509039^DC2079^2C1036
T^002073^07^382064^4C0000^05
E^001000
>>> The head of the source list has been freed (response from LL_free_head )

Lyciih@DESKTOP-CR5NUFU MINGW64 ~/Desktop/github/System-Programming/SIC assembler (main)
$
```



```
test.o
1 H^COPY ^001000^00107A
2 T^001000^1E^141033^482039^001036^281030^301015^482061^3C1003^00102A^0C1039^00102D
3 T^00101E^15^0C1036^482061^081033^4C0000^454F46^000003^000000
4 T^002039^1E^041030^001030^E0205D^30203F^D8205D^281030^302057^549039^2C205E^38203F
5 T^002057^1C^101036^4C0000^F1^001000^041030^E02079^302064^509039^DC2079^2C1036
6 T^002073^07^382064^4C0000^05
7 E^001000
```



```
test.o - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明
H^COPY ^001000^00107A
T^001000^1E^141033^482039^001036^281030^301015^482061^3C1003^00102A^0C1039^00102D
T^00101E^15^0C1036^482061^081033^4C0000^454F46^000003^000000
T^002039^1E^041030^001030^E0205D^30203F^D8205D^281030^302057^549039^2C205E^38203F
T^002057^1C^101036^4C0000^F1^001000^041030^E02079^302064^509039^DC2079^2C1036
T^002073^07^382064^4C0000^05
E^001000
```

## 討論

在這個作業的過程中我瞭解到編譯的過程需要經過兩次掃描，否則在不知道標籤位址的情況下根本無法把命令後方作為參數的標籤換成正確的位址。同時在字串切割處理的每一個階段都應該盡量做測試，避免一開始的小錯誤在後期耗費大量的時間來尋找源頭