System Programming
Assignment #1

SIC Assembler

系級：　資工大三
學號：　410121021
　　410121058
姓名：　林育慈
　　陳勇安

2014.11.13

Assignment Description

Write an SIC assembler that reads an SIC assembly program, translates SIC statesments into their machine code equivalents, and generates and object file.

Highlight of the way you write the prgram

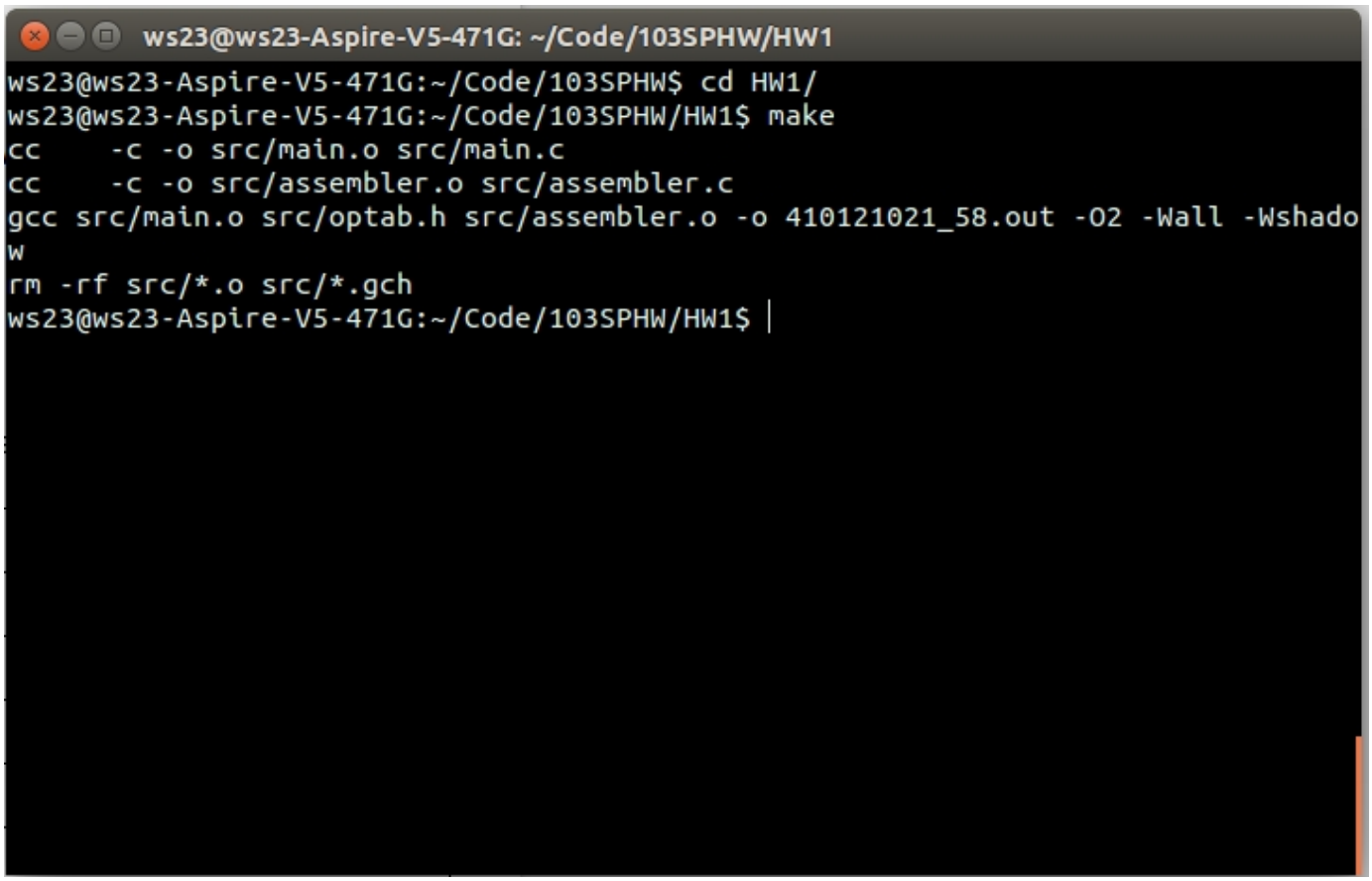
- assembler() -> To assembly the SIC code
    - pass1() -> To do first scanning to assembly
        - To give all instruction local address
        - To create symbol table
    - pass2() -> To do second scanning to assembly
        - To complete the object code
- toTarget() -> To make object code store into the target file
    - According to the SIC object code format
    - Comment at line 307, 368-375, 399 in assembler.c is optional to make user easy to read, but not the correct format for SIC object code

The program listing

- main.c
  - main file
  - do simple I/O & usage
- optab.h
  - define the opcode table
- assembler.h
  - the library of SIC assembler
    - void ini();
      - to initial the memory, prepare to do assembly
    - void addCode(char*);
      - to store a statement of SIC code, split them to four fields
    - void assembler();
      - the core of assembly
    - void toTarget(FILE*);
      - the core of make object code store to a output file
- assembler.c
  - define
    - symbol table structure
      - symbol, address
    - filed of instruction
      - lable, operation, operands, comment, locate address, object code
  - the source code of assembler.h
    - void showIns();
      - to show all instruction on the screen
    - void showSym();
      - to show symbol table on the screen
    - void addCode(char*);
    - void pass1();
      - Scanning first time
        - to give all instruction locall address
        - to create symbol table
    - void pass2();
      - Scanning second time
        - to complete the object code for each instruction
    - void ini();
    - void assembler();
    - void toTarget(FILE*);

Test run results

- Environment & Tools
  - OS: Ubuntu 14.04 LTS, i686, 3.13.0-40-generic, GNU/Linux
  - Compiler: gcc (Ubuntu 4.8.2-19ubuntu1) 4.8.2
  - Language: C89
  - Develope Tool: vim
- Execute
  - cd HW1
  - make
  - ./410121021_58 <your SIC code> [the output file name you want]



change director into it, and make it

```
ws23@ws23-Aspire-V5-471G: ~/Code/103SPHW/HW1
ws23@ws23-Aspire-V5-471G:~/Code/103SPHW/HW1$ ./410121021_58.out example.sic
-------To store the source code into Array
-------To assembly
----- PASS 1 ------
CLOOP      | 001003
ENDFIL     | 001015
EOF        | 00102A
THREE      | 00102D
ZERO       | 001030
RETADR     | 001033
LENGTH     | 001036
BUFFER     | 001039
RDREC      | 002039
RLOOP      | 00203F
EXIT       | 002057
INPUT      | 00205D
```

```
WRREC      | LDX        | ZERO       |   2061 | 041030
WLOOP      | TD         | OUTPUT     |   2064 | 102079
           | JEQ        | WLOOP      |   2067 | 302064
           | LDCH       | BUFFER,X   |   206A | 508000
           | WD         | OUTPUT     |   206D | DC2079
           | TIX        | LENGTH     |   2070 | 2C1036
           | JLT        | WLOOP      |   2073 | 382064
           | RSUB       |            |   2076 | 4C0000
OUTPUT     | BYTE       | X'05'      |   2079 | 05
           | END        | FIRST      |        |
-------To out
-----Complete
ws23@ws23-Aspire-V5-471G:~/Code/103SPHW/HW1$
```

Execute.

```
ws23@ws23-Aspire-V5-471G: ~/Code/103SPHW/HW1
          | WD         | OUTPUT     |   206D | DC2079
          | TIX        | LENGTH     |   2070 | 2C1036
          | JLT        | WLOOP      |   2073 | 382064
          | RSUB       |            |   2076 | 4C0000
OUTPUT    | BYTE       | X'05'      |   2079 | 05
          | END        | FIRST      |        |
-------To out
------Complete
ws23@ws23-Aspire-V5-471G:~/Code/103SPHW/HW1$ more example.sic.out
HCOPY  00100000107A
 ^     ^     ^
T0010001E141033482039001036281030301015482061 3C100300102A0C103900102D
 ^     ^ ^     ^     ^     ^     ^     ^     ^     ^     ^     ^
T001021150C1036482061081033 4C0000454F46000003000000
 ^     ^ ^     ^     ^     ^     ^     ^     ^
T0010181E041030001030102 05D30203FD8205D2810303020575480002C205E38203F
 ^     ^ ^     ^     ^     ^     ^     ^     ^     ^     ^     ^
T0010211C1010364C0000F1001000041030102079302064508000DC20792C1036
 ^     ^ ^     ^     ^ ^     ^     ^     ^     ^     ^     ^
T001021073820644C000005
 ^     ^ ^     ^     ^
E001000
 ^
ws23@ws23-Aspire-V5-471G:~/Code/103SPHW/HW1$
```

The output file content

```
ws23@ws23-Aspire-V5-471G: ~/Code/103SPHW/HW1
          | .          |            |        |
          | .          |            |        |
          | .          |            |        |
WRREC     | LDX        | ZERO       |   2061 | 041030
WLOOP     | TD         | OUTPUT     |   2064 | 102079
          | JEQ        | WLOOP      |   2067 | 302064
          | LDCH       | BUFFER,X   |   206A | 508000
          | WD         | OUTPUT     |   206D | DC2079
          | TIX        | LENGTH     |   2070 | 2C1036
          | JLT        | WLOOP      |   2073 | 382064
          | RSUB       |            |   2076 | 4C0000
OUTPUT    | BYTE       | X'05'      |   2079 | 05
          | END        | FIRST      |        |
-------To out
------Complete
HCOPY  00100000107A
T0010001E141033482039001036281030301015482061 3C100300102A0C103900102D
T00101E150C1036482061081033 4C0000454F46000003000000
T0000001E041030001030102 05D30203FD8205D2810303020575480002C205E38203F
T0020571C1010364C0000F1001000041030102079302064508000DC20792C1036
T002073073820644C000005
E001000
ws23@ws23-Aspire-V5-471G:~/Code/103SPHW/HW1$
```

Cancel the ^ sign

Discuss

其實SIC的組譯器演算規則並不複雜，且老師又說保證輸入的SIC Code會是合法的，因此這次的作業僅僅只是可以進行轉換的不完整組譯器。

但無奈由於對C的字串處理不甚熟悉，尤其在字串分割時不停的出現segmentation fault，實在是讓人很心煩，後來才知道錯誤的主因有兩個，一個是記憶體方面的 strtok 誤用，另一個則是 strcmp 字串比較時的回傳值誤解。

strtok，字串分割函式，它的執行方式是依照指定的起始位址向後尋找，找到指定的分割字元（在這次作業我們用到的字元分別有空白、單引號）後，用空字元（'\0'）將它取代，並回傳起始位置，而完成字串分割的目的。

strcmp，字串比較函式，它在兩字串比較後，若完全相同，會回傳0，而0在C當中代表的是false，因此過程中許多的判斷式全部相反了，造成整個程式出現各種意外。

經過了這次的練習，我們最大的收穫就是對於字串、字元的處理，而且我們的Coding Style並不算是非常好，有些變數的命名，隔天就忘了它的用處，註解也使用的不夠多，造成前一天寫的思考邏輯接不上來。