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

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include"linked_list.h"

//存放指令和對應opcode的結構，用來查詢比對
typedef struct opcode{
    char *command;
    int code;
}opcode;

//總共有26個指令
opcode sic[26];

//紀錄掃描到的標籤及地址的單向鏈結串列的節點
typedef struct label{
  char *label_name;
    int addres;
  llNode_t node;
}label;

//暫存object code的結構(使用位域)
typedef struct sic24_t{
    unsigned int addres : 16;
    unsigned int opcode : 8;
    }sic24_t;

//顯示一個單向鏈結串列所有節點的資料(用在開發中途測試串列是否正常運作)
int printf_all_list(llNode_t *head)
{
  llNode_t *current = head;
  while(current->next != NULL )
  {
    current = current->next;
    printf("%-8s " , return_to_user_struct_pointer(label, node, current)->label_name);
        printf("%X\n" , return_to_user_struct_pointer(label, node, current)->addres);
  }
  printf("end\n");
    return 0;
}



//將指數乘開的函數
int exponent_Int(const int base, int n)
{
    int p = base;
    if(n == 0)
    {
        p = 1;
    }
    else
    {
        for (int i = 1; i < n; i++)
        {
            p *= base;
        }
    }
```

```c
    return p;
}

//將讀取的字串視為16進位，並轉換為10進位數字的函數
int hex_to_dex(char *hex)
{
    char *char_temp = (char *)malloc(strlen(hex));
    strcpy(char_temp, hex);
    char_temp = strtok(char_temp, "\n");
    char temp[2];
    int total = 0;
    int count = strlen(char_temp);

    while(count-- && count >= 0)
    {
        sprintf(temp, "%c", *char_temp);
        if(strcmp(temp, "A") == 0){total += exponent_Int(16, count)*10;}
        if(strcmp(temp, "B") == 0){total += exponent_Int(16, count)*11;}
        if(strcmp(temp, "C") == 0){total += exponent_Int(16, count)*12;}
        if(strcmp(temp, "D") == 0){total += exponent_Int(16, count)*13;}
        if(strcmp(temp, "E") == 0){total += exponent_Int(16, count)*14;}
        if(strcmp(temp, "F") == 0){total += exponent_Int(16, count)*15;}
        else{total += exponent_Int(16, count)*atoi(temp);}
        char_temp++;
    }

    return total;
}

//取得命令中的目標檔名做為參數
int main(int argc, char *argv[])
{
    //初始化每種指令及其對應的opcode
    sic[0].command = "ADD";
        sic[0].code = 24;

        sic[1].command = "DIV";
        sic[1].code = 36;

        sic[2].command = "JGT";
        sic[2].code = 52;

        sic[3].command = "LDA";
        sic[3].code = 0;

        sic[4].command = "LDX";
        sic[4].code = 4;

        sic[5].command = "RD";
        sic[5].code = 216;

        sic[6].command = "STCH";
        sic[6].code = 84;

        sic[7].command = "STX";
        sic[7].code = 16;

        sic[8].command = "TIX";
        sic[8].code = 44;

```

```c
        sic[9].command = "AND";
        sic[9].code = 64;

        sic[10].command = "J";
        sic[10].code = 60;

        sic[11].command = "JLT";
        sic[11].code = 56;

        sic[12].command = "LDCH";
        sic[12].code = 80;

        sic[13].command = "MUL";
        sic[13].code = 32;

        sic[14].command = "RSUB";
        sic[14].code = 76;

        sic[15].command = "STL";
        sic[15].code = 20;

        sic[16].command = "SUB";
        sic[16].code = 28;

        sic[17].command = "WD";
        sic[17].code = 220;

        sic[18].command = "COMP";
        sic[18].code = 40;

        sic[19].command = "JEQ";
        sic[19].code = 48;

        sic[20].command = "JSUB";
        sic[20].code = 72;

        sic[21].command = "LDL";
        sic[21].code = 8;

        sic[22].command = "OR";
        sic[22].code = 68;

        sic[23].command = "STA";
        sic[23].code = 12;

        sic[24].command = "STSW";
        sic[24].code = 232;

        sic[25].command = "TD";
        sic[25].code = 224;

//******************************   第一階段 檔案處理 ********************************************//


    for (int i = 1; i < argc; i++)
    {
        //打開來源檔案，同時組合新的檔名並以它新增空白檔案
        FILE *ass = fopen(argv[i],"r");
        printf("[%d] [source] %s\n", i, argv[i]);
```

```c
        char *obj_file_name = strtok(argv[i], ".");
        char *obj= ".o";
        strcat(obj_file_name, obj);
        FILE *obj_file = fopen(obj_file_name,"w");
        printf("[%d] [target] %s\n\n", i, obj_file_name);


        //宣告組譯過程中會用到的buffer及變數
        char buffer[50];
        char obj_temp[100];
        char obj_cat_temp[60];
        char record_head_temp[50];
        char temp1[50];
        char temp2[50];
        char temp3[50];
        char temp4[50];
        char temp5[50];
        int code_address = 0;
        llNode_t *label_list = LL_init();
        llNode_t *label_list_temp;
        label *new = NULL;
        char program_name[10];
        int START_address = 0;
        int END_address = 0;
        sic24_t obj_code;
        char *BYTE_temp;
        int arg_get = 0;
        int T_count = 0;
        int T_start_count = 0;
        llNode_t *free_temp;
        label *free_top_temp;


        //第一次掃描開始，針對不同的狀況及規則算出地址
        while(fgets(buffer, 50, ass))
        {
            memset(temp1, 0, 50);
            memset(temp2, 0, 50);
            memset(temp3, 0, 50);
            memset(obj_cat_temp, 0, 50);

            sscanf(buffer, "%s %s %s", temp1, temp2, temp3);
            if(strcmp(temp1, ".") != 0 )
            {
                if(strcmp(temp2, "START") != 0)
                {
                    if(strcmp(temp1, "END") != 0)
                    {

                        for(int i = 0; i < 26; i++)
                        {
                            if(strcmp(temp1, sic[i].command) == 0)
                            {
                                code_address+=3;
                            }
                        }

                        for(int i = 0; i < 26; i++)
                        {
                            if(strcmp(temp2, sic[i].command) == 0)
```

```c
                    {
                        //如果是標籤就在串列尾端新增一個節點紀錄標籤名及地址
                        new = malloc(sizeof(label));
                        new->label_name = malloc(strlen(temp1));
                        strcpy(new->label_name, temp1);
                        new->addres = code_address;
                        LL_add_tail(&new->node, label_list);

                        code_address+=3;
                    }
                }

                if(strcmp(temp2, "RESB") == 0)
                {
                    //如果是標籤就在串列尾端新增一個節點紀錄標籤名及地址
                    new = malloc(sizeof(label));
                    new->label_name = malloc(strlen(temp1));
                    strcpy(new->label_name, temp1);
                    new->addres = code_address;
                    LL_add_tail(&new->node, label_list);

                    code_address += atoi(temp3);
                }

                if(strcmp(temp2, "RESW") == 0)
                {
                    //如果是標籤就在串列尾端新增一個節點紀錄標籤名及地址
                    new = malloc(sizeof(label));
                    new->label_name = malloc(strlen(temp1));
                    strcpy(new->label_name, temp1);
                    new->addres = code_address;
                    LL_add_tail(&new->node, label_list);

                    code_address += 3*atoi(temp3);
                }

                if(strcmp(temp2, "BYTE") == 0)
                {
                    //如果是標籤就在串列尾端新增一個節點紀錄標籤名及地址
                    new = malloc(sizeof(label));
                    new->label_name = malloc(strlen(temp1));
                    strcpy(new->label_name, temp1);
                    new->addres = code_address;
                    LL_add_tail(&new->node, label_list);

                    sscanf(temp3, "%[^']'%[^']", temp4, temp5);

                    if(*temp4 == 'C')
                    {
                        code_address += strlen(temp5);
                    }

                    if(*temp4 == 'X')
                    {
                        code_address += strlen(temp5)/2 + strlen(temp5)%2;
                    }

                }

                if(strcmp(temp2, "WORD") == 0)
```

```c
                        {
                            //如果是標籤就在串列尾端新增一個節點紀錄標籤名及地址
                            new = malloc(sizeof(label));
                            new->label_name = malloc(strlen(temp1));
                            strcpy(new->label_name, temp1);
                            new->addres = code_address;
                            LL_add_tail(&new->node, label_list);

                            code_address += 3;
                        }
                    }
                    else
                    {
                        END_address = code_address;
                    }

                }
                else
                {
                    code_address += hex_to_dex(temp3);
                    START_address = code_address;
                    strcpy(program_name, temp1);
                }
            }

        }

        //印出object file 的檔頭片段
        printf("H^%-6s^%06X^%06X\n", program_name, START_address, END_address - START_address);
        sprintf(record_head_temp, "H^%-6s^%06X^%06X\n", program_name, START_address, END_address - START_address);
        fputs(record_head_temp, obj_file);

        //重置檔案資料流指針回起點
        rewind(ass);
        code_address = 0;

        //第二次掃描開始
        while(fgets(buffer, 50, ass))
        {

            memset(temp1, 0, 50);
            memset(temp2, 0, 50);
            memset(temp3, 0, 50);


            sscanf(buffer, "%s %s %s", temp1, temp2, temp3);
            if(strcmp(temp1, ".") != 0 )
            {
                if(strcmp(temp2, "START") != 0)
                {
                    if(strcmp(temp1, "END") != 0)
                    {
                        for(int i = 0; i < 26; i++)
                        {
                            //比對第一段字串
                            if(strcmp(temp1, sic[i].command) == 0)
                            {
                                obj_code.opcode = sic[i].code;
```

```c
                                label_list_temp = label_list;

                                while(label_list_temp->next != NULL)
                                {
                                    label_list_temp = LL_next_node(label_list_temp);
                                    if(strcmp(temp2, return_to_user_struct_pointer(label, node, label_list_temp)->label_name) == 0)
                                    {
                                        obj_code.addres = return_to_user_struct_pointer(label, node, label_list_temp)->addres;
                                        arg_get = 1;
                                    }

                                }

                                if(strcmp(temp1,"RSUB") != 0 && arg_get == 0)
                                {
                                    sscanf(temp2, "%[^,],%s",temp4, temp5);

                                    label_list_temp = label_list;
                                    //比對參數是否為標籤
                                    while(label_list_temp->next != NULL)
                                    {
                                        label_list_temp = LL_next_node(label_list_temp);
                                        if(strcmp(temp4, return_to_user_struct_pointer(label, node, label_list_temp)->label_name) == 0)
                                        {
                                            obj_code.addres = return_to_user_struct_pointer(label, node, label_list_temp)->addres;
                                        }

                                    }

                                    if(*temp5 == 'X')
                                    {
                                        obj_code.addres += 32768;
                                    }
                                }

                                //如果片段加上這次的opject code會超過30個Byte
                                if(T_count + 3 > 30)
                                {
                                    //顯示片段起始訊息，同時在輸出片段到檔案前先把起始訊息寫進檔案中一行的起頭
                                    printf("T^%06X^%02X", T_start_count, T_count);
                                    sprintf(record_head_temp, "T^%06X^%02X", T_start_count, T_count);
                                    fputs(record_head_temp, obj_file);

                                    //輸出整段object code 到終端機和檔案
                                    printf("%s\n", obj_temp);
                                    fputs(obj_temp, obj_file);
                                    fputs("\n", obj_file);
                                    memset(obj_temp, 0, 100);
                                    T_count = 0;
                                }

                                if(T_count == 0)
                                {
```

```c
                                sprintf(obj_temp, "^%02X%04X", obj_code.opcode, obj_code.addres);
                                T_start_count = code_address;
                            }
                            else
                            {
                                sprintf(obj_cat_temp, "^%02X%04X", obj_code.opcode, obj_code.addres);
                                strcat(obj_temp, obj_cat_temp);
                            }

                            T_count += 3;

                            //如果片段剛好裝滿30個Byte
                            if(T_count == 30)
                            {
                                //顯示片段起始訊息，同時在輸出片段到檔案前先把起始訊息寫進檔案中一行的起頭
                                printf("T^%06X^%02X", T_start_count, T_count);
                                sprintf(record_head_temp, "T^%06X^%02X", T_start_count, T_count);
                                fputs(record_head_temp, obj_file);

                                //輸出整段object code 到終端機和檔案
                                printf("%s\n", obj_temp);
                                fputs(obj_temp, obj_file);
                                fputs("\n", obj_file);
                                memset(obj_temp, 0, 100);

                                T_count = 0;
                            }

                            code_address+=3;
                            obj_code.opcode = 0;
                            obj_code.addres = 0;
                        }
                    }

                    //比對第二段字串
                    for(int i = 0; i < 26; i++)
                    {
                        if(strcmp(temp2, sic[i].command) == 0)
                        {

                            obj_code.opcode = sic[i].code;

                            label_list_temp = label_list;
                            while(label_list_temp->next != NULL)
                            {
                                label_list_temp = LL_next_node(label_list_temp);
                                if(strcmp(temp3, return_to_user_struct_pointer(label, node, label_list_temp)->label_name) == 0)
                                {
                                    obj_code.addres = return_to_user_struct_pointer(label, node, label_list_temp)->addres;
                                }
                            }


                            //如果片段加上這次的opject code會超過30個Byte
```

```c
                                if(T_count + 3 > 30)
                                {
                                    //顯示片段起始訊息，同時在輸出片段到檔案前先把起始訊息寫進檔案中一行的起頭
                                    printf("T^%06X^%02X", T_start_count, T_count);
                                    sprintf(record_head_temp, "T^%06X^%02X", T_start_count, T_count);
                                    fputs(record_head_temp, obj_file);

                                    //輸出整段object code 到終端機和檔案
                                    printf("%s\n", obj_temp);
                                    fputs(obj_temp, obj_file);
                                    fputs("\n", obj_file);
                                    memset(obj_temp, 0, 100);
                                    T_count = 0;
                                }

                                if(T_count == 0)
                                {
                                    T_start_count = code_address;
                                    sprintf(obj_temp, "^%02X%04X", obj_code.opcode, obj_code.addres);
                                }
                                else
                                {
                                    sprintf(obj_cat_temp, "^%02X%04X", obj_code.opcode, obj_code.addres);
                                    strcat(obj_temp, obj_cat_temp);
                                }

                                T_count += 3;

                                //如果片段剛好裝滿30個Byte
                                if(T_count == 30)
                                {
                                    //顯示片段起始訊息，同時在輸出片段到檔案前先把起始訊息寫進檔案中一行的起頭
                                    printf("T^%06X^%02X", T_start_count, T_count);
                                    sprintf(record_head_temp, "T^%06X^%02X", T_start_count, T_count);
                                    fputs(record_head_temp, obj_file);

                                    //輸出整段object code 到終端機和檔案
                                    printf("%s\n", obj_temp);
                                    fputs(obj_temp, obj_file);
                                    fputs("\n", obj_file);
                                    memset(obj_temp, 0, 100);

                                    T_count = 0;
                                }

                                code_address+=3;
                                obj_code.opcode = 0;
                                obj_code.addres = 0;
                            }
                        }

                        //命令以外的其他狀況處理
                        if(strcmp(temp2, "RESB") == 0)
                        {
```

```c

                            //如果片段加上這次的opject code會超過30個Byte
                            if(T_count + atoi(temp3) > 30)
                            {
                                //顯示片段起始訊息，同時在輸出片段到檔案前先把起始訊息寫進檔案中一行的起頭
                                printf("T^%06X^%02X", T_start_count, T_count);
                                sprintf(record_head_temp, "T^%06X^%02X", T_start_count, T_count);
                                fputs(record_head_temp, obj_file);

                                //輸出整段object code 到終端機和檔案
                                printf("%s\n", obj_temp);
                                fputs(obj_temp, obj_file);
                                fputs("\n", obj_file);
                                memset(obj_temp, 0, 100);
                                T_count = 0;
                            }

                            //如果片段剛好裝滿30個Byte
                            if(T_count == 30)
                            {
                                //顯示片段起始訊息，同時在輸出片段到檔案前先把起始訊息寫進檔案中一行的起頭
                                printf("T^%06X^%02X", T_start_count, T_count);
                                sprintf(record_head_temp, "T^%06X^%02X", T_start_count, T_count);
                                fputs(record_head_temp, obj_file);

                                //輸出整段object code 到終端機和檔案
                                printf("%s", obj_temp);
                                fputs(obj_temp, obj_file);
                                fputs("\n", obj_file);
                                memset(obj_temp, 0, 100);

                                T_count = 0;
                            }
                            code_address += atoi(temp3);

                        }

                        if(strcmp(temp2, "RESW") == 0)
                        {
                            code_address += 3*atoi(temp3);
                        }

                        if(strcmp(temp2, "BYTE") == 0)
                        {
                            sscanf(temp3, "%[^']'%[^']", temp4, temp5);

                            if(*temp4 == 'C')
                            {
                                //如果片段加上這次的opject code會超過30個Byte
                                if(T_count + strlen(temp5) > 30)
                                {
                                    //顯示片段起始訊息，同時在輸出片段到檔案前先把起始訊息寫進檔案中一行的起頭
                                    printf("T^%06X^%02X", T_start_count, T_count);
                                    sprintf(record_head_temp, "T^%06X^%02X", T_start_count, T_count);
```

```c
                    fputs(record_head_temp, obj_file);

                    //輸出整段object code 到終端機和檔案
                    printf("%s\n", obj_temp);
                    fputs(obj_temp, obj_file);
                    fputs("\n", obj_file);
                    memset(obj_temp, 0, 100);
                    T_count = 0;
                }

                if(T_count == 0)
                {
                    T_start_count = code_address;
                    strcat(obj_temp, "^");

                    BYTE_temp = temp5;
                    for(int i = 0 ; i < strlen(temp5) ; i++)
                    {
                        sprintf(obj_cat_temp, "%X" , *BYTE_temp);
                        strcat(obj_temp, obj_cat_temp);
                        BYTE_temp++;
                    }
                }
                else
                {
                    strcat(obj_temp, "^");
                    BYTE_temp = temp5;
                    for(int i = 0 ; i < strlen(temp5) ; i++)
                    {
                        sprintf(obj_cat_temp, "%X" , *BYTE_temp);
                        strcat(obj_temp, obj_cat_temp);
                        BYTE_temp++;

                    }
                }

                T_count += strlen(temp5);

                //如果片段剛好裝滿30個Byte
                if(T_count == 30)
                {
                    //顯示片段起始訊息，同時在輸出片段到檔案前先把起始訊息寫進檔案中一行的起頭
                    printf("T^%06X^%02X", T_start_count, T_count);
                    sprintf(record_head_temp, "T^%06X^%02X", T_start_count, T_count);
                    fputs(record_head_temp, obj_file);

                    //輸出整段object code 到終端機和檔案
                    printf("%s\n", obj_temp);
                    fputs(obj_temp, obj_file);
                    fputs("\n", obj_file);
                    memset(obj_temp, 0, 100);

                    T_count = 0;
                }
                code_address += strlen(temp5);
            }

            if(*temp4 == 'X')
```

```c
                        {
                            //如果片段加上這次的opject code會超過30個Byte
                            if(T_count + strlen(temp5)/2 + strlen(temp5)%2 > 30)
                            {
                                //顯示片段起始訊息，同時在輸出片段到檔案前先把起始訊息寫進檔案中一行的起頭
                                printf("T^%06X^%02X", T_start_count, T_count);
                                sprintf(record_head_temp, "T^%06X^%02X", T_start_count, T_count);
                                fputs(record_head_temp, obj_file);

                                //輸出整段object code 到終端機和檔案
                                printf("%s\n", obj_temp);
                                fputs(obj_temp, obj_file);
                                fputs("\n", obj_file);
                                memset(obj_temp, 0, 100);
                                T_count = 0;
                            }

                            if(T_count == 0)
                            {
                                T_start_count = code_address;

                                sprintf(obj_temp, "^%s", temp5);
                            }
                            else
                            {
                                sprintf(obj_cat_temp, "^%s", temp5);
                                strcat(obj_temp, obj_cat_temp);
                            }

                            T_count += strlen(temp5)/2 + strlen(temp5)%2;

                            //如果片段剛好裝滿30個Byte
                            if(T_count == 30)
                            {
                                //顯示片段起始訊息，同時在輸出片段到檔案前先把起始訊息寫進檔案中一行的起頭
                                printf("T^%06X^%02X", T_start_count, T_count);
                                sprintf(record_head_temp, "T^%06X^%02X", T_start_count, T_count);
                                fputs(record_head_temp, obj_file);

                                //輸出整段object code 到終端機和檔案
                                printf("%s\n", obj_temp);
                                fputs(obj_temp, obj_file);
                                fputs("\n", obj_file);
                                memset(obj_temp, 0, 100);

                                T_count = 0;
                            }
                            code_address += strlen(temp5)/2 + strlen(temp5)%2;
                        }

                    }

                    if(strcmp(temp2, "WORD") == 0)
                    {
                        //如果片段加上這次的opject code會超過30個Byte
                        if(T_count + 3 > 30)
```

```
                            {
                                //顯示片段起始訊息，同時在輸出片段到檔案前先把起始訊息寫進檔案中一行的起頭
                                printf("T^%06X^%02X", T_start_count, T_count);
                                sprintf(record_head_temp, "T^%06X^%02X", T_start_count, T_count);
                                fputs(record_head_temp, obj_file);

                                //輸出整段object code 到終端機和檔案
                                printf("%s\n", obj_temp);
                                fputs(obj_temp, obj_file);
                                fputs("\n", obj_file);
                                memset(obj_temp, 0, 100);
                                T_count = 0;
                            }

                            if(T_count == 0)
                            {
                                T_start_count = code_address;
                                sprintf(obj_temp, "^%06X", atoi(temp3));
                            }
                            else
                            {
                                sprintf(obj_cat_temp, "^%06X", atoi(temp3));
                                strcat(obj_temp, obj_cat_temp);
                            }

                            T_count += 3;

                            //如果片段剛好裝滿30個Byte
                            if(T_count == 30)
                            {
                                //顯示片段起始訊息，同時在輸出片段到檔案前先把起始訊息寫進檔案中一行的起頭
                                printf("T^%06X^%02X", T_start_count, T_count);
                                sprintf(record_head_temp, "T^%06X^%02X", T_start_count, T_count);
                                fputs(record_head_temp, obj_file);

                                //輸出整段object code 到終端機和檔案
                                printf("%s\n", obj_temp);
                                fputs(obj_temp, obj_file);
                                fputs("\n", obj_file);
                                memset(obj_temp, 0, 100);

                                T_count = 0;
                            }
                            code_address += 3;
                        }
                    }
                }
                else
                {
                    code_address += hex_to_dex(temp3);
                }
            }
            arg_get = 0;
        }

        //如果翻譯完有剩餘的片段，把剩下的印出來
```

```c
        if(strcmp(obj_temp, "") != 0)
        {
            printf("T^%06X^%02X", T_start_count, T_count);
            sprintf(record_head_temp, "T^%06X^%02X", T_start_count, T_count);
            fputs(record_head_temp, obj_file);

            printf("%s\n", obj_temp);
            fputs(obj_temp, obj_file);
            fputs("\n", obj_file);
        }

        //印出object file 的終止片段
        printf("E^%06X\n", START_address);
        sprintf(record_head_temp, "E^%06X", START_address);
        fputs(record_head_temp, obj_file);

        //關閉來源檔案及目標檔案
        fclose(ass);
        fclose(obj_file);

        //釋放紀錄標籤的鏈結串列
        while(!LL_isEmpty(label_list))
      {
            free_temp = LL_next_node(label_list);
            free_top_temp = return_to_user_struct_pointer(label, node, free_temp);
        LL_delete_next(label_list);
            memset(free_top_temp->label_name, 0, strlen(free_top_temp->label_name));
            free(free_top_temp->label_name);
            free_top_temp->label_name = NULL;
            free(free_top_temp);
            free_top_temp = NULL;
            free_temp = NULL;
      }
      LL_free_head(label_list);
    }

}
```

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)
$
```

```
assembler.c    test.o
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儘量做測試，避免一開始的小錯誤在後期耗費大量的時間來尋找源頭