SIC Assembler 作業

做法：用字串陣列存放 optable，opcode，Loc 位址，symbol table，

object code，還有要組譯的 code。

1.先自訂了 optable 的指令以及對應的 opcode。

2.先算出每列指令前的 Loc 位址。

3.運用建好的 Loc 位址 建立 symbol table。

4.運用已知的 Loc 位址，symbol table，opcode 算出 object code。

5.印出所需要的內容。

其中因為操作需要建立了四個函數功能分別為

(1) 10 進制 轉 16 進制的字串 `char* DecToHex(int dec);`
(2) 16 進制的字串 轉 10 進制 `int HexToDec(const char* hexPoint);`
(3)字串相連 `char* stringAdd(const char* , const char* );`
(4)幫 16 進制的字串前面補零 `char* addZero(const char*, int );`

原程式碼：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <iostream>
#include <cstdlib>
//SIC Assembler
char* DecToHex(int dec);
int HexToDec(const char* hexPoint);
char* stringAdd(const char* , const char* );
char* addZero(const char*, int );
using namespace std;
```

```
int main(void)
{
	string optable[8] = {"LDA", "LDS", "ADDR", "STA", "RESW", "WORD", "FIRST", 
"END"};
	string opcode[8] = {	"14", "18" , "90"	, "23" , "52"	, "53"	, "06"	, "08" };
	string symtable[9][2] = {""};
	string Hexloc[9];
	string objcode[9];
	string h[3]={"Loc", "Source statement", "Object Code"}; //最上方標示欄位
	string zero4 = "0000", zero5 = "00000";
	string RS_code = "4", RA_code = "0";
	string codeLength_noZero, codeLength; //程式的長度
	int locStart,locEnd;
	//int a=4567;
	int i, sym = 0; // sym:symtable  的  index

	string a;
	string Addcode[9][3] = {"ADD"	, "START" , "1000"
				,"FIRST", "LDA"		, "FIVE"
				,""		, "LDS"		, "TWO"
				,""		, "ADDR"  , "S,A"
				,""		, "STA"		, "DATA"
				,"FIVE"	, "WORD","5"
				,"TWO"	, "WORD","2"
				,"DATA"	, "RESW"  ,"1"
				,""		, "END"		,"FIRST"};

	//Hecloc 位址
	for(i = 0; i<9; i++){

		if(Addcode[i][1] == "START")
		{
			Hexloc[i] = Addcode[i][2];
			locStart = i; //Hexloc 開始的 index
			Hexloc[i+1] = Addcode[i][2];
		}

		//計算 Hexloc
```

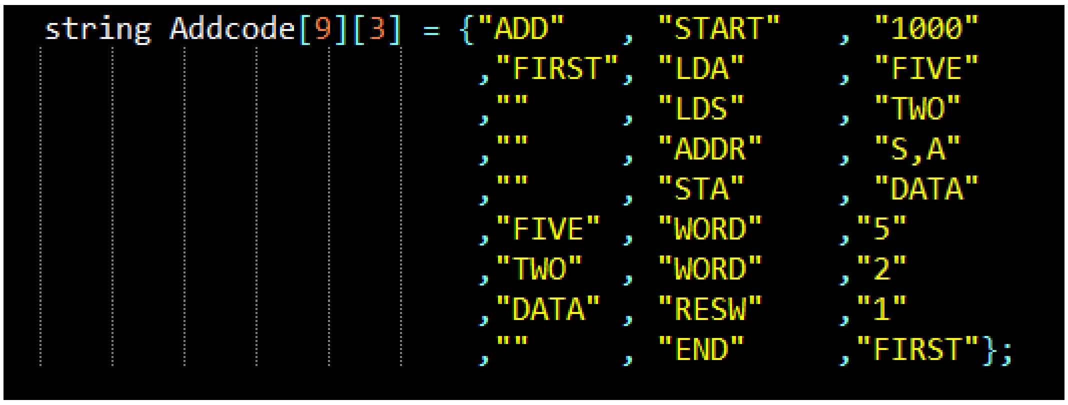
```
            if(Addcode[locStart][1] == "START")
            {
                  if(Addcode[i][1] == "LDA"){

                        Hexloc[i+1] = DecToHex(HexToDec(Hexloc[i].c_str())+3);
                  }
                  else if(Addcode[i][1] == "LDS"){

                        Hexloc[i+1] = DecToHex(HexToDec(Hexloc[i].c_str())+3);
                  }
                  else if(Addcode[i][1] == "ADDR"){

                        Hexloc[i+1] = DecToHex(HexToDec(Hexloc[i].c_str())+3);
                  }
                  else if(Addcode[i][1] == "STA"){

                        Hexloc[i+1] = DecToHex(HexToDec(Hexloc[i].c_str())+3);
                  }
                  else if(Addcode[i][1] == "WORD"){

                        Hexloc[i+1] = DecToHex(HexToDec(Hexloc[i].c_str())+3);
                  }
                  else if(Addcode[i][1] == "RESW"){

                        Hexloc[i+1] = DecToHex(HexToDec(Hexloc[i].c_str()) +
HexToDec(Addcode[i][2].c_str())*3);
                  }
            }
            if(Addcode[i][1] == "END"){

                  locEnd = i;
            }
      }
      //建立  symtable
      for(i = 0; i < 9 ; i++){

            if(Addcode[i][0] != ""){
```

```
            symtable[sym][0] = Addcode[i][0];
            symtable[sym][1] = Hexloc[i];
            sym++;
        }
    }
    //建立    object code
    for(i = 0; i < 9; i++)
    {
        if(Addcode[i][1] == "LDA"){
            for(int j = 0; j < 9; j++){

                if(Addcode[i][2] == symtable[j][0]){
                    objcode[i] = stringAdd(opcode[0].c_str(), symtable[j][1].c_str());
                }
            }
        }
        else if(Addcode[i][1] == "LDS"){
            for(int j = 0; j < 9; j++){

                if(Addcode[i][2] == symtable[j][0]){
                    objcode[i] = stringAdd(opcode[1].c_str(), symtable[j][1].c_str());
                }
            }
        }
        else if(Addcode[i][1] == "STA"){
            for(int j = 0; j < 9; j++){

                if(Addcode[i][2] == symtable[j][0]){
                    objcode[i] = stringAdd(opcode[3].c_str(), symtable[j][1].c_str());
                }
            }
        }
        else if(Addcode[i][1] == "ADDR"){

                if(Addcode[i][2] == "S,A"){
                    objcode[i] = stringAdd(opcode[2].c_str(), RS_code.c_str());
                    objcode[i] = stringAdd(opcode[2].c_str(), RA_code.c_str());
                }
                else if(Addcode[i][2] == "A,S"){
                    objcode[i] = stringAdd(opcode[2].c_str(), RA_code.c_str());
                    objcode[i] = stringAdd(opcode[2].c_str(), RS_code.c_str());
                }
            }
            else if(Addcode[i][1] == "WORD"){

                if(6 - Addcode[i][2].size() == 5){

                    objcode[i] = stringAdd(objcode[i].c_str(), zero5.c_str());
                    objcode[i] = stringAdd(objcode[i].c_str(), Addcode[i][2].c_str());
                }
                else if(6 - Addcode[i][2].size() == 4){

                    objcode[i] = stringAdd(objcode[i].c_str(), zero4.c_str());
                    objcode[i] = stringAdd(objcode[i].c_str(), Addcode[i][2].c_str());
                }

            }

        }
        //印出  Assembly Program with Object Code
        printf("%-8s%-24s%-16s\n",h[0].c_str(), h[1].c_str(), h[2].c_str());
        printf("---------------------------------------------------\n");
        for(i = 0; i < 9; i++){

            printf("%-8s%-8s%-8s%-8s%-6s\n",Hexloc[i].c_str(), Addcode[i][0].c_str(),
Addcode[i][1].c_str(), Addcode[i][2].c_str(), objcode[i].c_str());
        }
        printf("---------------------------------------------------\n");
        codeLength_noZero = DecToHex(HexToDec(Hexloc[locEnd].c_str()) -
HexToDec(Hexloc[locStart].c_str())); //  計算程式長度
        //補零
        codeLength = addZero(codeLength_noZero.c_str(), codeLength_noZero.size());
        Hexloc[locStart] = addZero( Hexloc[locStart].c_str(),    Hexloc[locStart].size());
        //印出  Object Program Fromat
        printf("H %-6s %-6s %-6s\n", Addcode[locStart][0].c_str(),
Hexloc[locStart].c_str(), codeLength.c_str());
```

```
        printf("T %-6s %-2s", Hexloc[locStart].c_str(), codeLength_noZero.c_str());
        for(i = locStart+1; i < locEnd; i++){

                printf(" %-6s", objcode[i].c_str());
        }
        printf("\n");
        printf("E %-6s\n", Hexloc[locStart].c_str());
        //cout<<HexToDec(Hexloc[locEnd].c_str()) -
HexToDec(Hexloc[locStart].c_str())<<endl;
}

//  幫字串補零
  char* addZero(const char* s_in, int length)
{
        string zero0 = "", zero1 = "0", zero2 = "00", zero3 = "000", zero4 = "0000", zero5
= "00000";
        char output[16];
        if(6 - length == 5){
                return stringAdd(zero5.c_str(), s_in);
        }
        else if(6 - length == 4){
                return stringAdd(zero4.c_str(), s_in);
        }
        else if(6 - length == 3){
                return stringAdd(zero3.c_str(), s_in);
        }
        else if(6 - length == 2){
                return stringAdd(zero2.c_str(), s_in);
        }
        else if(6 - length == 1){
                return stringAdd(zero1.c_str(), s_in);
        }
        else
            return stringAdd(zero0.c_str(), s_in);;
  }
//字串相連
char* stringAdd(const char* sa, const char* sb)
{
        char *a;
        char *b;
        char final[9];
        a=const_cast<char*>(sa); //consr char* To char*
        b=const_cast<char*>(sb);
        strcat(a,b); //字串相連
        return a;
}
//10 進制  轉  16 進制
char* DecToHex(int dec)
{
        char Hex[9];
        sprintf(Hex, "%X", dec);
        return Hex;
}
//16 進制  轉  10 進制
int HexToDec(const char* hexPoint)
{
        char hex[9];
        strncpy(hex, hexPoint, strlen(hexPoint) + 1);
        long long decimal, place;
        int i = 0, val, len;

        decimal = 0;
        place = 1;

        len = strlen(hex);
        len--;

        for(i=0; hex[i]!='\0'; i++){

        // Find the decimal representation of hex[i]
        if(hex[i]>='0' && hex[i]<='9')
        {
        val = hex[i] - 48;
        }
        else if(hex[i]>='a' && hex[i]<='f')
        {
```

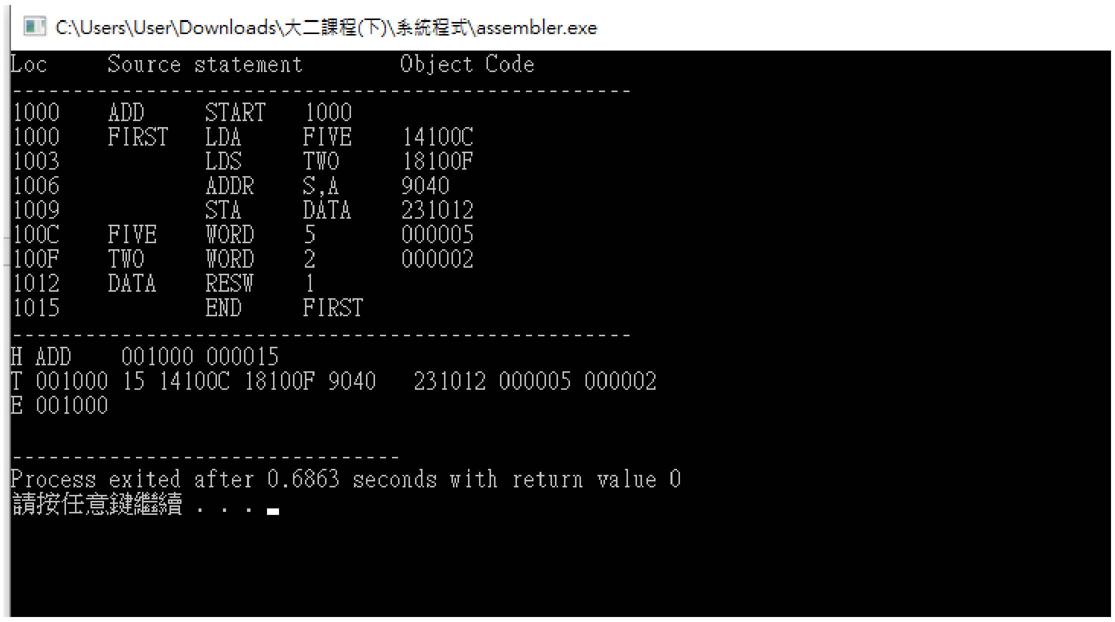
```
        val = hex[i] - 97 + 10;
        }
        else if(hex[i]>='A' && hex[i]<='F')
        {
        val = hex[i] - 65 + 10;
        }

        decimal += val * pow(16, len);
        len--;
        }

        return decimal;
}
```

測試資料如下圖：



```
string Addcode[9][3] = {"ADD"    , "START"    , "1000"
                       ,"FIRST", "LDA"      , "FIVE"
                       ,""      , "LDS"      , "TWO"
                       ,""      , "ADDR"     , "S,A"
                       ,""      , "STA"      , "DATA"
                       ,"FIVE"  , "WORD"     ,"5"
                       ,"TWO"   , "WORD"     ,"2"
                       ,"DATA"  , "RESW"     ,"1"
                       ,""      , "END"      ,"FIRST"};
```

結果輸出如下圖：



C:\Users\User\Downloads\大二課程(下)\系統程式\assembler.exe

```
Loc      Source statement         Object Code
----------------------------------------------------
1000     ADD      START    1000
1000     FIRST    LDA      FIVE     14100C
1003              LDS      TWO      18100F
1006              ADDR     S,A      9040
1009              STA      DATA     231012
100C     FIVE     WORD     5        000005
100F     TWO      WORD     2        000002
1012     DATA     RESW     1
1015              END      FIRST
----------------------------------------------------
H ADD     001000 000015
T 001000 15 14100C 18100F 9040   231012 000005 000002
E 001000

--------------------------------
Process exited after 0.6863 seconds with return value 0
請按任意鍵繼續 . . .
```

討論：在運算的過程中遇到 **string** 跟 **char*** 跟 **char[]** 跟 **const char*** 型態之間相互轉換的問題，了解相互轉換的方式花了許久的時間。

心得：這次的實作花了蠻久的時間，有空的話會再挑戰不用 **string** 陣列，只用 **char** 來完成。