SIC 流程說明
(由於程式碼太多,只挑重點部分說明)

(程式執行前,需有 opcode.txt 和 source.txt)

首先我定義了一個結構,每一行由一個 line 所組成
會將整行拆成 label ,mnemonic, operand,之後會比較好處理

```
struct line //store the info of each line
{
    string wholeString;

    string label;
    string mnemonic;
    string operand;

    int location;
    string myOPCODE;

};
```

symbolNode 會儲存 symbol 與對應的 location
(程式中的 location 我都會以 int 儲存,之後輸出才會轉成 16 進位)

```
struct symbolNode //store the info of the symbol and its address
{
    string symbol;
    int address;
};
```

各種 function:

將 16 進位的字串轉成對應的 int

```
int hexToDec(string s)//string to int
```

將 10 進位的字串轉成對應的 int

```
int dec_atoi(string s)//string to int
```

將 10 進位數字轉成 16 進位的字串

```
string decToHex(int num)//
```

將 10 進位數字轉成 16 進位的字串(不夠 4 位會補 0)

```
string decToHex_four(int num)//tu
```

這邊會用 hashMap 來儲存 mnemonic 所對應的 opcode,以及 symbol 所對應的 address

lineTable 儲存每一行的資訊

```
unordered_map<string,string> opTable;
unordered_map<string,int> symTable;

int symbolCount = 0;
symbolNode symbolTable[1000];//set the maximum symbol to 1000
int lineCount = 0;
line lineTable[1000];//set the maximum line to 1000


fstream inputFile;
inputFile.open("source.txt",ios::in); //read the soucre.txt

fstream opcodeFile;
opcodeFile.open("opcode.txt",ios::in); //read the opcode.txt
```

建立 opTable

```
while(opcodeFile >> s)//put the opcode to opTable
{
    opcodeFile >> codeNum;
    opTable[s] = codeNum;
}
```

(以下程式碼太多,沒辦法完整截圖,只能截一小部分)
開始 pass_1
一開始會先讀取第一行,來取得我的 starting address
途中會進行字串處理把一行切成 label,mnemonic,operand

```
//==================================pass_1==================================

//since the input is always correct, the first line is always START,we first
getline(inputFile,s); // get first line

line firstLine;
firstLine.wholeString = s;
unsigned int index = 0;//point the char from the first line
firstLine.myOPCODE = "";
firstLine.label = "";
while(true)//construct the label name till meet tab
{
    if(s[index] == ' ')
    {
        index++;
        continue;
    }
```

之後把 pc 設成第一行的 location

```
int pc = firstLine.location; //set the program counter
```

之後再繼續取得其他行

```
while(getline(inputFile,s))//get other lines
{
    line temp;
    temp.wholeString = s;
    temp.location = pc;//set the location to pc
    unsigned int i = 0;
    if(s[0] != '\t')//have label(the first character is not tab)
    {
        temp.label = "";
        while(true)
        {
            if(s[i] == ' ')//skip space
            {
                i++;
                continue;
            }
```

如果有 label 就把他加入 symbol table 內

```
symbolNode newSymbol; //create new symbol object
newSymbol.symbol = temp.label;
newSymbol.address = pc;

symbolTable[symbolCount] = newSymbol; //insert the
symTable[newSymbol.symbol] = newSymbol.address; //i
symbolCount++;
```

之後會依據我的 mnemonic 來增加 pc,並把 pc 填入該行的 location

```
//add the program counter
if(temp.mnemonic == "WORD")
{
    pc += 3;
}
else if(temp.mnemonic == "RESW")
{
    pc += 3*(dec_atoi(temp.operand));
}
else if(temp.mnemonic == "RESB")
{
    pc += dec_atoi(temp.operand);
}
```

BYTE 的話如果是 C 開頭,pc 要加入後面有幾個字元(一個字元代表一個 byte)
X 開頭的話固定一個 BYTE,加一

```
else if(temp.mnemonic == "BYTE")
{
    if((temp.operand)[0] == 'C')
    {
        pc += strlen((temp.operand).c_str()) - 3;//calculate how many character in the operand
    }
    else
    {
        pc += 1;
    }
}
```

如果都不是以上的情況就加 3(normal instruction)

```
}
else//normal instruction
{
    pc += 3;
}

lineTable[lineCount] = temp;
lineCount++;
```

輸出 pass1_locationAndSource

```
//write the locationAndSource.txt
fstream locationAndSource;
locationAndSource.open("pass1_locationAndSource.txt",ios::out);
locationAndSource << "Loc\t" << "Source statement" << endl;
locationAndSource << decToHex_four(firstLine.location) << "\t" << firstLine.wholeString << endl;
for(int i = 0; i < lineCount-1; i++)
{
    locationAndSource << decToHex_four(lineTable[i].location) << "\t" << lineTable[i].wholeString << endl;
}
locationAndSource << "\t" << lineTable[lineCount-1].wholeString << endl;
locationAndSource.close();
cout << "Pass 1 : " << endl << endl;
cout << "write the location of each source code to pass1_locationAndSource.txt!" << endl << endl;
```

再輸出 symbol table

```
//write the symbolTable.txt
fstream symbolTableFile;
symbolTableFile.open("pass1_symbolTable.txt",ios::out);
symbolTableFile << "Name\t" << "Address" << endl;
for(int i = 0; i < symbolCount; i++)
{
    symbolTableFile << symbolTable[i].symbol << "\t" << decToHex_four(symbolTable[i].address) << endl;
}
symbolTableFile.close();
cout << "write the symbol table to pass1_symbolTable.txt!" << endl << endl;
```

之後進入 pass_2

如果 mnemonic 是 RESW 或 REWB 的話直接略過,不需填 object code

```
//==================================pass_2==================================
for(int i = 0; i < lineCount-1; i++)
{
    if(lineTable[i].mnemonic == "RESW" || lineTable[i].mnemonic == "RESB")
    {
        lineTable[i].myOPCODE = "";
        continue;
    }
```

WORD 的話需要將 10 進位的 operand 改成 16 進位的數值(不夠 6 位要補 0)

```
if(lineTable[i].mnemonic == "WORD")
{
    string opcode = "";
    string s1 = decToHex(dec_atoi(lineTable[i].operand));
    for(unsigned int j = 0; j < (6-strlen(s1.c_str())); j++)//fill 0 to length 6
    {
        opcode += "0";
    }
    opcode += s1;
    lineTable[i].myOPCODE = opcode;
}
```

BYTE 的話可分為 C 開頭與 X 開頭

C 開頭代表後面有一些字元,會連續串在 opcode 上面

```
else if(lineTable[i].mnemonic == "BYTE")
{
    if((lineTable[i].operand)[0] == 'C')//C''
    {
        string opcode = "";
        unsigned int charNum = strlen((lineTable[i].operand).c_str()) - 3;//check how ma
        for(unsigned int j = 2; j < (strlen((lineTable[i].operand).c_str())-1); j++)//cc
        {
            string t = decToHex((int)((lineTable[i].operand)[j]));
            if(strlen(t.c_str()) == 1)
            {
                opcode += "0";
            }
            opcode += t;
        }
        lineTable[i].myOPCODE = opcode;
    }
```

X 開頭的話為一個 16 進位(1 Byte)

```
else//X''
{
    int len = strlen((lineTable[i].operand).c_str());
    string opcode = "";
    for(int j = 2; j < len-1; j++)
    {
        opcode += (lineTable[i].operand)[j];
    }
    lineTable[i].myOPCODE = opcode;
}
```

如果沒有 operand 後面要補 4 個 0

```
else if(lineTable[i].operand == "")//no operand
{
    string opcode = opTable[lineTable[i].mnemonic];//use optable to get opcode
    opcode += "0000";
    lineTable[i].myOPCODE = opcode;
}
```

其他的就是一般的 instruction 了

```
else//normal instruction
{
    string opcode = opTable[lineTable[i].mnemonic];//use optable to get opcode

    int x = -1;
    for(int j = 0; j < strlen((lineTable[i].operand).c_str()); j++)
    {
        if((lineTable[i].operand)[j] == ',')//indexed addressing
        {
            x = j;
            break;
        }
    }
    if(x == -1)//normal addressing
    {
        opcode += decToHex_four(symTable[lineTable[i].operand]);
```

這邊又可分為 index addressing 與非 index addressing
我會檢查 operand,如果裡面有逗號,代表是 index addressing
我會直接把 target address 加上 32768(第 16 個 bit)
之後再轉成 opcode
而非 index addressing 只要把 mnemonic 對應的 opcode 串上 symbol 對應的 address 就可以了

```
if(x == -1)//normal addressing
{
    opcode += decToHex_four(symTable[lineTable[i].operand]);
}
else//indexed addressing
{
    string realOperand = "";//to get the real operand without ',X'
    for(int j = 0; j < x; j++)
    {
        realOperand += (lineTable[i].operand)[j];
    }
    int targetAddress = symTable[realOperand];
    targetAddress += 32768;//to add the X
    opcode += decToHex_four(targetAddress);
}
```

之後輸出每行對應的 object code

```
cout << "Pass_2 : " << endl << endl;
 //write the source with location and objectcode to pass2_source_LocObj.txt
fstream source_LocObj;
source_LocObj.open("pass2_source_LocObj.txt",ios::out);
source_LocObj << decToHex_four(firstLine.location) << "\t" << firstLine.wholeString << endl;
for(int i = 0; i < lineCount-1; i++)
{
    if(strlen(lineTable[i].operand.c_str()) >= 8)
    {
        source_LocObj << decToHex_four(lineTable[i].location) << "\t" << lineTable[i].wholeString << "\t" << lineTab
    }
    else if(lineTable[i].operand == "")
    {
         source_LocObj << decToHex_four(lineTable[i].location) << "\t" << lineTable[i].wholeString << "\t\t\t" << li
    }
    else
    {
```

再輸出 text record

```
//out format of first line
pass2_textRecord << 'H';
pass2_textRecord << firstLine.label;
for(int i = 0; i < 6-strlen(firstLine.label.c_str()); i++)//fill space
{
    pass2_textRecord << ' ';
}
pass2_textRecord << "00" << decToHex_four(firstLine.location) << "00" << decToHex_four(lineTable[lineCount-1]

for(int i = 0; i < lineCount-1; )
{
    string opcodeBuffer = "";
    int recordLen = 0;
    bool first = true;
    int byteCount = 0;
    int startAddress;
```

結果:

pass1_locationAndSource

pass1_locationAndSource - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

Loc	Source statement		
1000	COPY	START	1000
1000	FIRST	STL	RETADR
1003	CLOOP	JSUB	RDREC
1006		LDA	LENGTH
1009		COMP	ZERO
100C		JEQ	ENDFIL
100F		JSUB	WRREC
1012		J	CLOOP
1015	ENDFIL	LDA	EOF
1018		STA	BUFFER
101B		LDA	THREE
101E		STA	LENGTH
1021		JSUB	WRREC
1024		LDL	RETADR
1027		RSUB	
102A	EOF	BYTE	C'EOF'
102D	THREE	WORD	3
1030	ZERO	WORD	0
1033	RETADR	RESW	1
1036	LENGTH	RESW	1
1039	BUFFER	RESB	4096
2039	RDREC	LDX	ZERO
203C		LDA	ZERO
203F	RLOOP	TD	INPUT
2042		JEQ	RLOOP
2045		RD	INPUT
2048		COMP	ZERO
204B		JEQ	EXIT

pass1_symbolTable

pass1_symbolTable - 記

檔案(F) 編輯(E) 格式(O)

Name	Address
FIRST	1000
CLOOP	1003
ENDFIL	1015
EOF	102A
THREE	102D
ZERO	1030
RETADR	1033
LENGTH	1036
BUFFER	1039
RDREC	2039
RLOOP	203F
EXIT	2057
INPUT	205D
MAXLEN	205E
WRREC	2061
WLOOP	2064
OUTPUT	2079

pass2_source_LocObj

pass2_source_LocObj - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

1000	COPY	START	1000	
1000	FIRST	STL	RETADR	141033
1003	CLOOP	JSUB	RDREC	482039
1006		LDA	LENGTH	001036
1009		COMP	ZERO	281030
100C		JEQ	ENDFIL	301015
100F		JSUB	WRREC	482061
1012		J	CLOOP	3C1003
1015	ENDFIL	LDA	EOF	00102A
1018		STA	BUFFER	0C1039
101B		LDA	THREE	00102D
101E		STA	LENGTH	0C1036
1021		JSUB	WRREC	482061
1024		LDL	RETADR	081033
1027		RSUB		4C0000
102A	EOF	BYTE	C'EOF'	454F46
102D	THREE	WORD	3	000003
1030	ZERO	WORD	0	000000
1033	RETADR	RESW	1	
1036	LENGTH	RESW	1	
1039	BUFFER	RESB	4096	
2039	RDREC	LDX	ZERO	041030
203C		LDA	ZERO	001030
203F	RLOOP	TD	INPUT	E0205D
2042		JEQ	RLOOP	30203F
2045		RD	INPUT	D8205D
2048		COMP	ZERO	281030
204B		JEQ	EXIT	302057
204E		STCH	BUFFER,X	549039

pass2_textRecord

pass2_textRecord - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明


```
HCOPY    00100000107A
T0010001E141033482039001036281030301015482061 3C100300102A0C103900102D
T00101E150C10364820610810334C0000454F46000003000000
T0020391E041030001030E0205D30203FD8205D2810303020575490392C205E38203F
T0020571C1010364C0000F1001000041030E02079302064509039DC20792C1036
T002073073820644C000005
E001000
```