系統程式報告

作業一

(Programming Assignment #1)

系級：資工三

學號：410921202

姓名： 林芷萱

一、 Brief description of the assignment problem

此次作業是要使用 C 語言完成一個 SIC assembler，他會讀取一個 SIC assembly program，將其轉成 machine code，再生成一個 object file。

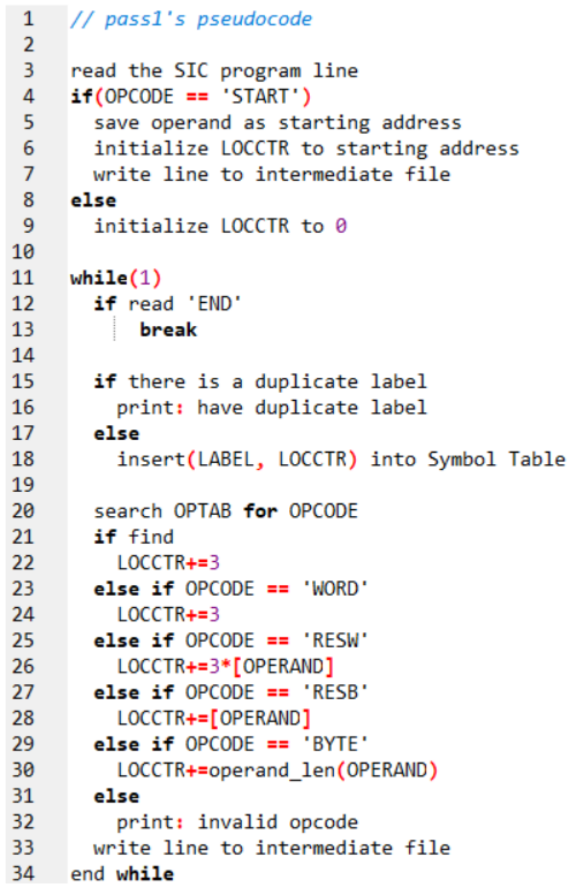
二、 Highlights on how you write the program

當 source program 進入 assembler 時，assembler 會對 source program 做兩次掃描，簡單來講就是第一次掃描(即 Pass 1)會產生一個中間檔案，而第二次掃描(即 Pass 2)會讀取此中間檔案並生成 object code。

Pass1

這個步驟的目的是計算 label 及 address 並建立 Symbol Table。

下圖他的 pseudocode：



```
// pass1's pseudocode

read the SIC program line
if(OPCODE == 'START')
  save operand as starting address
  initialize LOCCTR to starting address
  write line to intermediate file
else
  initialize LOCCTR to 0

while(1)
  if read 'END'
      break

  if there is a duplicate label
    print: have duplicate label
  else
    insert(LABEL, LOCCTR) into Symbol Table

  search OPTAB for OPCODE
  if find
    LOCCTR+=3
  else if OPCODE == 'WORD'
    LOCCTR+=3
  else if OPCODE == 'RESW'
    LOCCTR+=3*[OPERAND]
  else if OPCODE == 'RESB'
    LOCCTR+=[OPERAND]
  else if OPCODE == 'BYTE'
    LOCCTR+=operand_len(OPERAND)
  else
    print: invalid opcode
  write line to intermediate file
end while
```

Pass2

這個步驟的目的是產生 object code。

下圖為他的 pseudocode：

```
// pass2's pseudocode

read the SIC program line
if(OPCODE == 'START')
  write listing line

if OPCODE == 'START'
  write Header record to object program
else
  LOCCTR =0

initialize first text record
while(1)
  initialize objcode
  read line

  search OPTAB for OPCODE
  if find
    if there is a symbol in OPERAND
       search SYMTAB for OPERAND
       if find
         store symbol value as operand address
        else
          store 0 as operand address
  else if OPCODE == 'WORD' or 'BYTE'
    convert constant to object code
  else if OPCODE == 'RESW'
    LOCCTR+=3*[OPERAND]
    write text record to object program
    initialize new test record
  else if OPCODE == 'RESB'
    LOCCTR+=[OPERAND]
    write text record to object program
    initialize new test record
end while
```

三、 Program listing

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

//全域變數
#define MAXOP 27

char name[16];
FILE *f;
FILE *fobj;

char line[80];
char label[7];
char op[7];
char operand[10];
char new_operand[7];
int indexed = 0;
char hex[16];

char prog_name[7];
int  start_addr = 0;
int  start_text=0;//記 text 部分的起使位置
int  end_text=0;//記 text 部分的結束位置
int  prog_len = 0;
char obj_line[70];
char obj_code[7];
int  locctr = 0;
int  textpos = 0;

const char a_start[] = "START";
const char a_end[] = "END";
const char a_byte[] = "BYTE";
const char a_word[] = "WORD";
const char a_resb[] = "RESB";
const char a_resw[] = "RESW";

const char optab[26][2][6] = {{"ADD", "18"}, {"AND", "40"}, {"COMP", "28"}, {"DIV",
"24"}, {"J", "3C"}, {"JEQ", "30"}, {"JGT", "34"}, {"JLT", "38"}, {"JSUB", "48"},
{"LDA", "00"}, {"LDCH", "50"}, {"LDL", "08"}, {"LDX", "04"}, {"MUL", "20"}, {"OR",
"44"}, {"RD", "D8"}, {"RSUB", "4C"}, {"STA", "0C"}, {"STCH", "54"}, {"STL", "14"},
{"STSW", "E8"}, {"STX", "10"}, {"SUB", "1C"}, {"TD", "E0"}, {"TIX", "2C"}, {"WD",
"DC"}};
const char hex_c[16] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B',
'C', 'D', 'E', 'F'};

void deci_to_hex(int num)//10 進位轉成 16 進位
{
    char temp[16];
    int i = 0;
    int j = 0;
    while(num)
    {
        temp[i] = hex_c[num % 16];//對十進位制數求餘並最終與 hexc 陣列中的字元匹配
        i++;
        num=num/16;
    }
    j=i-1;
    for (i=0; j >= 0; j--, i++)//倒過來放好
        hex[i]=temp[j];
    hex[i]='\0';
}

typedef struct sym_node
```

```
{
    char l[7];
    int v;
    struct sym_node * next;
} symNODE;

symNODE *symtab = NULL;

symNODE* insert(char *s, int r)
{
    symNODE *t = NULL;
    t = (struct sym_node *)malloc(sizeof(struct sym_node));
    if( t )
    {
        strcpy(t->l, s);
        t->v = r;
        t->next=symtab;
    }
    return t;
}

symNODE* search(symNODE *t, char *s)//有起點
{
    if(t)
    {
        if (strcmp(s, t->l) == 0)
        {
            return t;
        }
        else
        {
            return search(t->next, s);
        }
    }
    else
    {
        return NULL;
    }
}

symNODE* new_search( char *s)//從頭找
{
    int i;
    strcpy(new_operand, s );
    for (i=strlen(new_operand); i<6; i++)
        new_operand[i] = ' ';
    new_operand[6] = '\0';
    return search( symtab, new_operand );
}

char* lookup (char *s)//找 op code
{
    int i = 0;
    int nf = 1;
    while ((i < MAXOP) && (nf))
    {
        if (strcmp(s, optab[i][0]) == 0)
            nf=0;
        else i++;
    }
    if (i >= MAXOP)
        return NULL;
    else
        return (char*)optab[i][1];
}
```

```
int operand_len ()//operand 長度
{
    int i, l;
    l = strlen(operand);
    if (operand[0] == 'C')
        l -= 3;
    else if (operand[0] == 'X')
        l = (l-3) / 2;
    return l;
}

int readline()//讀一行
{
    int i, j, l, x;

    fgets(line, 80, f);
    l = strlen(line);
    if ((l>0) && (line[0]!='.'))
    {
        for (i = 0; i < 6; i++)//記 label
        {
            label[i] = line[i];
        }
        label[i] = '\0';
        while(line[i]==' ') i++;
        j = 0;
        while ((line[i]!=' ') && (line[i]!='\0') && (line[i]!='\n') && (i < l))//記
operation
        {
            op[j] = line[i];
            i++;
            j++;
        }
        op[j] = '\0';
        while(line[i]==' ') i++;
        j = 0;
        while ((line[i]!=' ') && (line[i]!='\0') && (line[i]!='\n') && (i < l))//記
operand
        {
            operand[j] = line[i];
            i++;
            j++;
        }
        operand[j] = '\0';
        indexed = 0;
        x = strlen(operand);
        if((x>2) && (operand[x-2]==',') && (operand[x-1]=='X'))
        {
            operand[x-2] = '\0';
            indexed = 1;
        }
        return 1;
    }
    else
    {
        return 0;
    }
}

void pass1 ()
{
    readline();
    int l;
    if(strcmp(op,"START")==0)//從 START 開始
    {
        l = strlen(operand);
        int i, T;
        for(i=l-1, T=1; i>=0; i--, T*=16)
            locctr+=(operand[i]-'0')*T;
        start_addr=locctr;
    }
    else
        locctr=0;

    while(1)
    {
        readline();
        if(strcmp(op,"END")==0)//一直讀到 END
            break;
        if(line[0]!='.')//跳過有.的那幾行
        {
            symNODE *temp=search(symtab, label);//檢查是否有重複的 label
            if(line[0]!=' ')//label 有東西的話
            {
                if(temp!=NULL)
                printf("have duplicate label [%s]\n", label);
            else
                symtab=insert(label, locctr);//加入 label 表格
            }
            char *n=lookup(op);//找 op 表
            if(n!=NULL)//正常的 op
                locctr+=3;
            else if(strcmp(op,"WORD")==0)
                locctr+=3;
            else if(strcmp(op,"RESW")==0)
                locctr+=3*atoi(operand);//operand 轉成數字
            else if(strcmp(op,"RESB")==0)
                locctr+=atoi(operand);
            else if(strcmp(op,"BYTE")==0)
```

```
            {
                locctr+=operand_len(operand);
            }
            else
                printf("invalid opcode [%s]\n", op);
        }
    }
    prog_len=locctr-start_addr;
}

void print_symtab (symNODE * t)//印 symtab
{
    if (t)
    {
        print_symtab( t->next );
        printf("[%s] = [%5X]\n", t->l, t->v);
    }
}

void init_obj_line()//初始化 obj_line
{
    int i;
    for (i=0; i<70; i++)
        obj_line[i] = ' ';
    obj_line[i] = '\0';
}

void wr_header()//寫 Head
{
    init_obj_line();
    obj_line[0]='H';//0
    deci_to_hex(prog_len);
    int i, j;
    for(i=1, j=0; i<=6; i++, j++)//1-6 程式名稱
        obj_line[i]=label[j];
    for(i=7, j=0; i<=12; i++)//7-12 起始位置
    {
        if(i<=12-operand_len())//前面補 0
            obj_line[i]='0';
        else
        {
            obj_line[i]=operand[j];
            j++;
        }
    }
    deci_to_hex(prog_len);
    for(i=13, j=0; i<=18; i++)//13-18 總長度
    {
        if(i<=18-strlen(hex))//前面補 0
            obj_line[i]='0';
        else
        {
            obj_line[i]=hex[j];
            j++;
        }
    }
    printf(obj_line);
    printf("\n");
    fprintf(fobj, obj_line);
    fprintf(fobj, "\n");
}

void init_obj_code ()
{
    int i;
    for (i=0; i<6; i++)
```

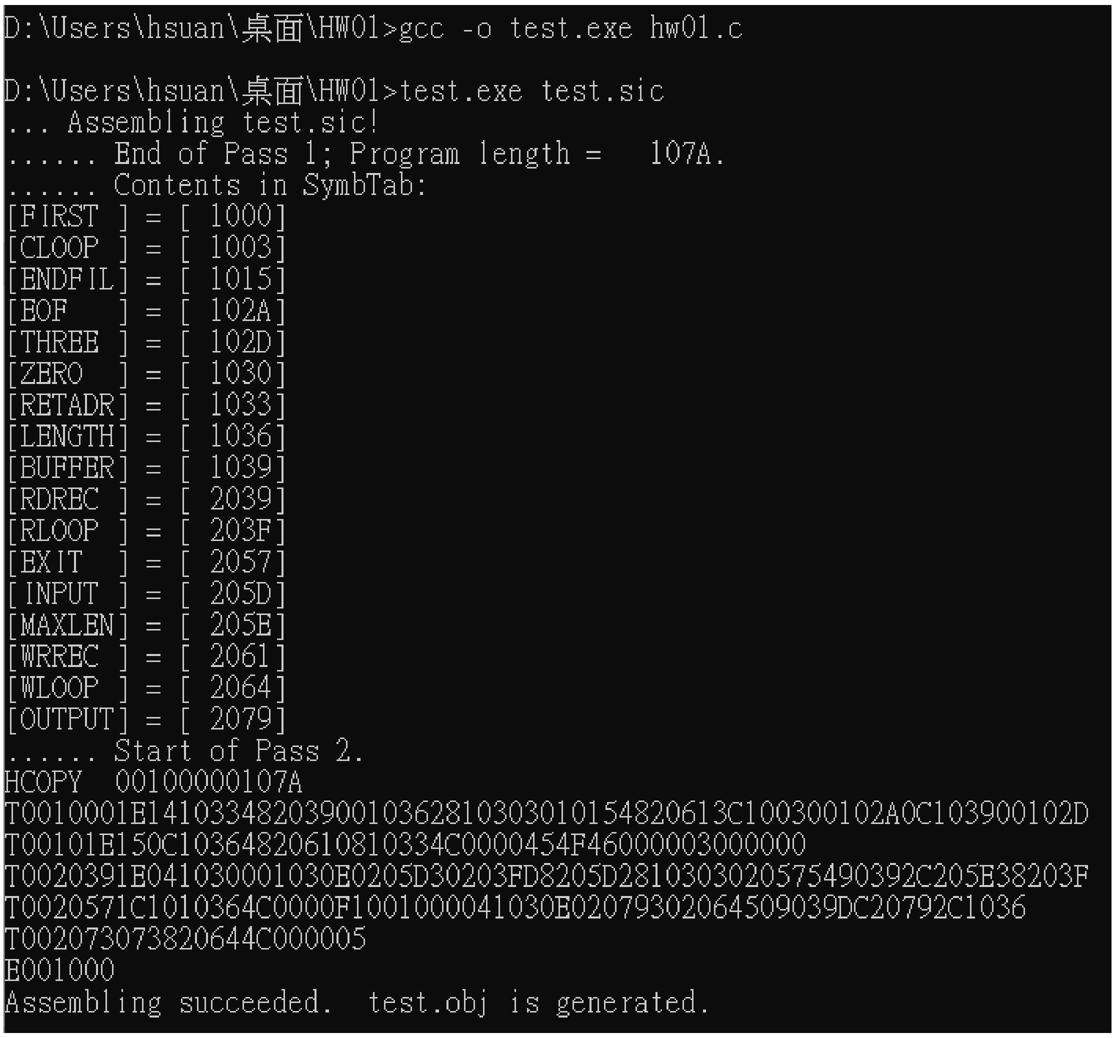
```
        obj_code[i] = ' ';
    obj_code[6] = '\0';
}

void conv_byte ( int l, char *p, char *q )
{
    int i, j, k, max, c, m, n;
    if (p[0] == 'X')
    {
        max = 2 * l;
        for (i=2, j=0, k=0; k < max&&p[i]!='\''; i++, j++, k++)
            q[j] = p[i];

        q[j] = '\0';
    }
    else if (p[0] == 'C')
    {
        max = l;
        for (i=2, j=0, k=0; k < max; i++, k++)
        {
            c = (int)p[i];
            m = c / 16;
            q[j++] = hex_c[m];
            n = c % 16;
            q[j++] = hex_c[n];
        }
        q[j] = '\0';
    }
    else
    {
        printf("Error: wrong operand of BYTE!\n");
    }
}
```

```
void init_text ()
{
    init_obj_line();
    sprintf( obj_line, "T%6X  ", start_text);//obj_line 的格式:T+起始位置
    int i;
    for (i=1; i<7; i++)
        if (obj_line[i] == ' ') obj_line[i] = '0';//空白補 0
    textpos = 9;
}

void wr_text ()
{
    if((end_text-start_text)>=0)//若長度為正
    {
        deci_to_hex(end_text-start_text);
        if(textpos<999)
            start_text=end_text;//現在的結尾是下一次的開頭
        if(strlen(hex)==2)
        {
            obj_line[7]=hex[0];
            obj_line[8]=hex[1];
        }
        else//長度不足 2 的話要補 0
        {
            obj_line[7]='0';
            obj_line[8]=hex[0];
        }
        printf(obj_line);
        printf("\n");
        fprintf(fobj, obj_line);
        fprintf(fobj, "\n");
    }
```

```
}

void add_text (int k)
{

    int const max = 69;
    int i;
    if ((textpos+k)>max)//當無法再加的時候就開始寫入Text
    {
        wr_text();
        init_text();
    }
    for (i=0; i<k; i++)//把code家道陣列尾端
    {
        obj_line[textpos] = obj_code[i];
        textpos++;
    }
    end_text=locctr;//更新結束位置
}

void wr_end ()
{
    init_obj_line();
    obj_line[0]='E';//0
    deci_to_hex(start_addr);
    int i, j;
    for(i=1, j=0; i<=6; i++)
    {
        if(i<=6-strlen(hex))
            obj_line[i]='0';
        else
        {
            obj_line[i]=hex[j];
            j++;
        }
    }
    printf(obj_line);
    printf("\n");
    fprintf(fobj, obj_line);
    fprintf(fobj, "\n");
}


void pass2 ()
{
    readline();
    if(strcmp(op,"START")==0)//START 開始
    {
        locctr=start_addr;
        start_text=locctr;
        wr_header();
    }
    else
        locctr=0;

    init_text ();
    while(1)
    {
        init_obj_code();
        readline();
        if(line[0]!='.')
        {
            char *n=lookup(op);//找 op 表
            if(n!=NULL)
            {
                obj_code[0]=n[0];
                obj_code[1]=n[1];
                if(strlen(line)>=16)//operand 是有東西的
                {
                    int i;
                    if(indexed==1)//有","的是間接定址模式
                        deci_to_hex(new_search(operand)->v+32768);
                    else
                        deci_to_hex(new_search(operand)->v);
                    int j;
                    for(i=2, j=0; i<6; i++, j++)
                        obj_code[i]=hex[j];
                    obj_code[i]=’\0’;

                }
                else
                {
                    int i, j;
                    for(i=2, j=0; i<6; i++, j++)
                        obj_code[i]=’0’;
                    obj_code[i]=’\0’;
                }
                locctr+=3;
                add_text(strlen(obj_code));//累加進陣列裡
            }
            else if(strcmp(op,"WORD")==0)
            {
                obj_code[0]=’0’;
                obj_code[1]=’0’;
                int i, j;
                int nn=atoi(operand);//轉成整數
                deci_to_hex(nn);
                for(i=2, j=0; i<6; i++)
                {
                    if(i>=6-strlen(hex))
                    {
                        obj_code[i]=hex[j];
                        j++;
                    }
                    else
                        obj_code[i]='0';
                }
                obj_code[i]='\0';
                locctr+=3;
                add_text(strlen(obj_code));//累加進陣列裡
            }
            else if(strcmp(op,"RESW")==0)
            {
                locctr+=3*atoi(operand);
                if(textpos!=999)
                {
                    wr_text();
                    init_text();
                }
                textpos=999;//直接開新的一行 Text
                start_text=locctr;
            }
            else if(strcmp(op,"RESB")==0)
            {
                locctr+=atoi(operand);
                if(textpos!=999)
                {
                    wr_text();
                    init_text();
                }
                textpos=999;
                start_text=locctr;
```

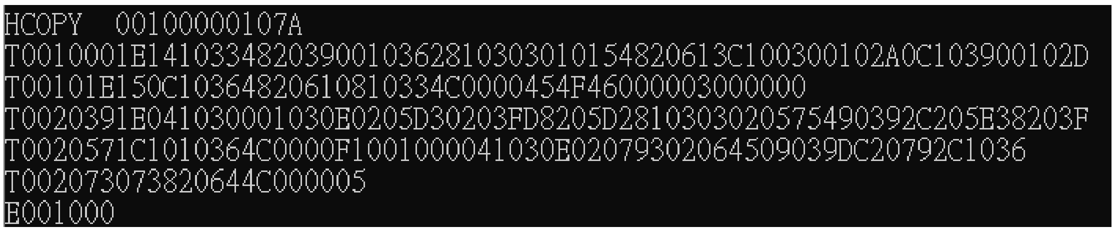
```
            }
            else if(strcmp(op,"BYTE")==0)
            {
                obj_code[0]='0';
                obj_code[1]='0';
                conv_byte(strlen(operand)-3, operand, obj_code);
                locctr+=operand_len(operand);
                add_text(strlen(obj_code));
            }
            else if(strcmp(op,"END")==0)
            {
                wr_text();
                break;
            }
            else
            {
                printf("invalid opcode [%s]\n", op);
            }
        }
    }
    wr_end();
}

int main(int argc, char*argv[])
{
    int t = argc;
    char fname[20];
    int i = 0;
    if (t == 2 )
    {
        f = fopen(argv[1], "r");
        if (f)
        {
            printf("... Assembling %s!\n", argv[1]);
            pass1();
            printf("...... End of Pass 1; Program length = %6X.\n", prog_len);
            printf("...... Contents in SymbTab:\n");
            print_symtab(symtab);
            fclose( f );
            strcpy( fname, argv[1] );
            for (i=0; (fname[i]!='.') && (fname[i]!='\0'); i++);
            fname[i++] = '.';
            fname[i++] = 'o';
            fname[i++] = 'b';
            fname[i++] = 'j';
            fname[i] = '\0';
            f = fopen(argv[1], "r");
            fobj = fopen("ans.txt", "w+");
            printf("...... Start of Pass 2.\n");
            pass2();
            printf("Assembling succeeded.  %s is generated.\n", fname);
            fclose( f );
            fclose( fobj );
        }
        else
        {
            printf("Assemble syntax: [assemble soure_file_name]\n");
        } // f
    }
    else
    {
        printf("Assemble syntax: [assemble soure_file_name]\n");
    } // t
}
```

四、 Test run snapshots



```
D:\Users\hsuan\桌面\HW01>gcc -o test.exe hw01.c

D:\Users\hsuan\桌面\HW01>test.exe test.sic
... Assembling test.sic!
...... End of Pass 1; Program length =   107A.
...... Contents in SymbTab:
[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]
...... Start of Pass 2.
HCOPY  00100000107A
T0010001E1410334820390010362810303010154820613C100300102A0C103900102D
T00101E150C10364820610810334C0000454F46000003000000
T0020391E041030001030E0205D30203FD8205D2810303020575490392C205E38203F
T0020571C1010364C0000F1001000041030E02079302064509039DC20792C1036
T002073073820644C000005
E001000
Assembling succeeded.  test.obj is generated.
```

對 object code 的輸出解釋：



```
HCOPY  00100000107A
T0010001E1410334820390010362810303010154820613C100300102A0C103900102D
T00101E150C10364820610810334C0000454F46000003000000
T0020391E041030001030E0205D30203FD8205D2810303020575490392C205E38203F
T0020571C1010364C0000F1001000041030E02079302064509039DC20792C1036
T002073073820644C000005
E001000
```

H header：程式名稱、Program 的起始 address(2-7 行)、Program length(14-19 行)

T text：紀錄 object code 的起始 address(2-7 行)、紀錄 object code length(8-9 行)、object code(10-69 行)

E end：第一個可執行指令的位置(2-7 行)

五、 Discussion

在這次的作業中學習到在 source code 進入 assembler 後是如何轉換成 object code，我覺得很有趣，以前接觸過的程式大部分都是像 C、Python 這類語言，原來電腦背後處理的是這樣的一個過程。上學期的程式語言與編譯器這堂課在作業一中學習了 compiler 的 front-end，而這學期的系統程式在作業一則是學習了 compiler 的 back-end，比起在教科書上學習理論，實做讓我更加了解 compiler 實際上是如何運行的。