Compiler Final Project

40647027S 陳冠穎

檔案說明

```
compiler-final-project/
├── a.out
├── asm.txt
├── compile.sh
├── final.l # lex code
├── final.y # yacc code
├── lex.yy.c
├── LICENSE.md
├── Program_Exercise_yacc_2021.pdf
├── README.md
├── stack.c # custom stack
├── stack.h # custom stack
├── symtab.c # symbol table
├── symtab.h # symbol table
├── testP   # test program
├── y.tab.c
└── y.tab.h
```

- final.l: lex 程式碼。
- final.y: yacc 程式碼
- stack.c .h: 自己寫的簡易 stack。
- symtab.c .h: 跟 symbol table 有關的函式。
- testP: 測試用的 MICRO 程式碼。

- compile.sh: 使用 sh compile.sh 指令可以編譯原始碼。

開發環境

OS: Ubuntu 20.04

Gcc: gcc version 9.3.0

安裝: sudo apt-get install flex bison libbison-dev

編譯、執行

```
sh compile.sh
```

```
./a.out < testP
```

最後輸出的 assembly code 會在 asm.txt 檔案中

說明: 其中 testP 為要編譯的程式碼檔案。

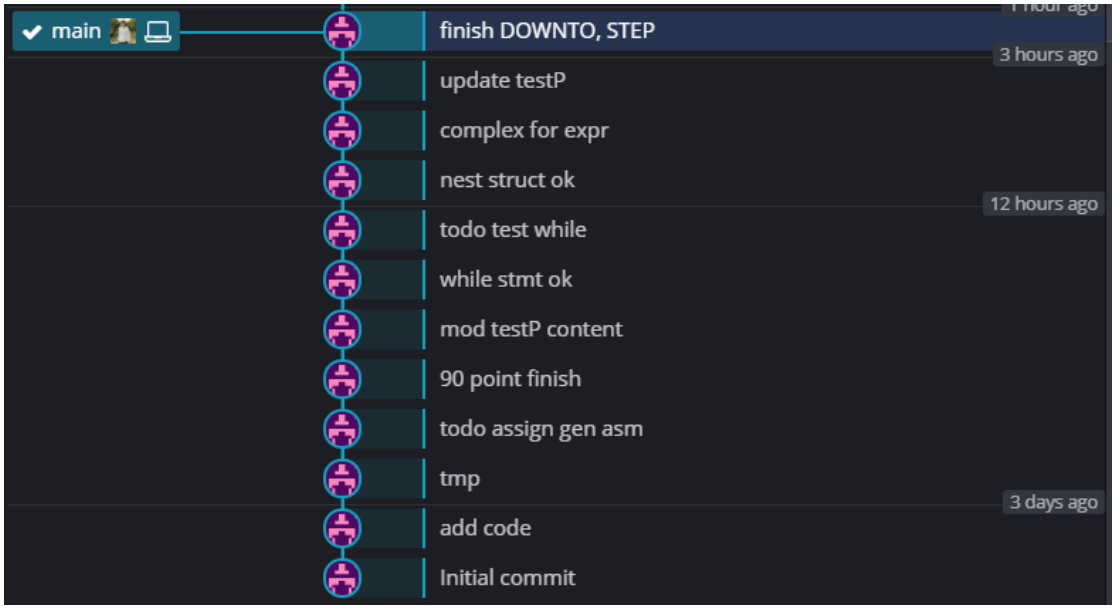
開發過程

6/7: 做好只能處理 DECALRE 的基礎部分。

6/15: 開始製作可以 Parse IF、FOR、WHILE，Parse 部分完成之後要開始產生 assembly code 其中遇到的問題是 expression 要轉換為 assembly code 比較麻煩，後來參考老師的 ch3-05.y 發現有使用到額外的 symbol table code，因此寫的 symbol table 的一些支援函式。

6/16: 做到可以產生 expression 的 assembly code，再來就分別完成 IF、FOR、WHILE 的產生 assembly code 功能，其中 label 的產生花了比較多時間除錯，完成後接著順便做加分題的部分。

以下是 commit 紀錄截圖：



Bonus 加分題部分

1. To support more complex FOR construct

```
FOR_Stmt: FOR '(' VarName Assign_Op Expr FOR_To Expr FOR_Step')'
          {
              fprintf(fp, "I_STORE %s,%s\n", ($5->name), $3);
              push(&for_stack, label_count);
              fprintf(fp, "lb&%d:    ", label_count++);
          }
          Stmt_List ENDFOR
          {
              if(!strcmp($8, "1")){
                  fprintf(fp, "%s %s\n", !strcmp($6, "TO")?"INC":"DEC", $3);
              }
              else{
                  struct symtab_struct *p = creatTmp(1);
                  fprintf(fp, "%s %s,%s,%s\n", !strcmp($6, "TO")?"I_ADD":"I_SUB", $3,
                  fprintf(fp, "I_STORE %s,%s\n", p->name, $3);
              }
              fprintf(fp, "I_CMP %s,%s\n", $3, ($7->name));
              fprintf(fp, "JL lb&%d\n", pop(&for_stack));
          }
        ;

FOR_Step: STEP Expr {$$ = $2->name; printf("STEP %s\n", $2->name);}
        | {$$ = "1"; }
        ;

FOR_To: TO   {$$ = "TO";}
      | DOWNTO   {$$ = "DOWNTO";}
      ;
```

使用 Expr 來對應複雜的表達式，以及 FOR_To、FOR_Step 來判斷是否使用 TO 或 DOWNTO，以及判斷使否使用 STEP 語法。

2. To support WHILE construct

```
WHILE_Stmt: WHILE
            {
                push(&while_stack, label_count);
                fprintf(fp, "lb&%d: ", label_count++);
            }
            '(' Condition ')' Stmt_List ENDWHILE
            {
                fprintf(fp, "J lb&%d\n", pop(&while_stack));
                fprintf(fp, "lb&%d: ", pop(&con_stack));
                printf("Match WHILE\n");
            }
          ;
```

WHILE 的製作比較容易，IF 做好就很容易達成，因為都使用到 Condition 的文法。

3. To support nested structure

這邊製作上遇到的問題是 label 的編號會出錯，所以實作了簡單的 Stack，寫在 stack.c stack.h 檔中，來輔助儲存 label 的編號，最後可以正確執行。

測試用 testP:

```
%%the beginning of an test data for Micro/Ex
Program testP
Begin
    declare I,J as integer;
    declare A,B,C,D, LLL[100] as float;
FOR (I:=2*J-4 DOWNTO 100*J+6)
    A:=-LLL[I]+B*D-C;
ENDFOR
I:=1;
WHILE (I<=100)
    IF (A>=10000.0) THEN
        print(A+3.14);
    ELSE
        print(2,1.4);
    ENDIF
    I:=I+1;
ENDWHILE
End
```

輸出 asm.txt:

asm.txt

```
1   START testP
2   Decalre I, Integer
3   Decalre J, Integer
4   Decalre A, Float
5   Decalre B, Float
6   Decalre C, Float
7   Decalre D, Float
8   Decalre LLL, Float_array,100
9   I_MUL 2,J,T&1
10  I_SUB T&1,4,T&2
11  I_MUL 100,J,T&3
12  I_ADD T&3,6,T&4
13  I_STORE T&2,I
14  lb&1:   F_UMINUS LLL[I], T&5
15  F_MUL B,D,T&6
16  F_ADD T&5,T&6,T&7
17  F_SUB T&7,C,T&8
18  F_STORE T&8,A
19  DEC I
20  I_CMP I,T&4
21  JL lb&1
22  I_STORE 1,I
23  lb&2:   I_CMP I, 100
24  JG lb&3
25  F_CMP A, 10000.0
26  JL lb&4
27  F_ADD A,3.14,T&9
28  CALL print,T&9
29  J lb&5
30  lb&4:   CALL print,2,1.4
31  lb&5:   I_ADD I,1,T&10
32  I_STORE T&10,I
33  J lb&2
34  lb&3:   HALT testP
35  Declare T&1, Integer
36  Declare T&2, Integer
37  Declare T&3, Integer
38  Declare T&4, Integer
39  Declare T&5, Float
40  Declare T&6, Float
41  Declare T&7, Float
42  Declare T&8, Float
43  Declare T&9, Float
44  Declare T&10, Integer
```

Copyright Claim