Compiler Final Project

40647027S 陳冠頴

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。

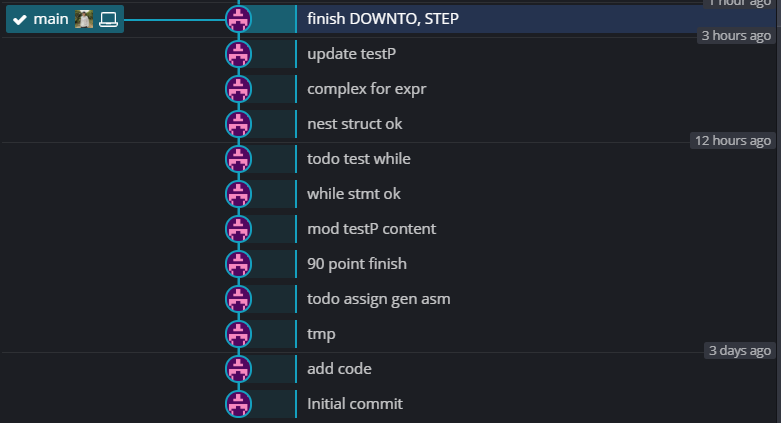
開發過程

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:



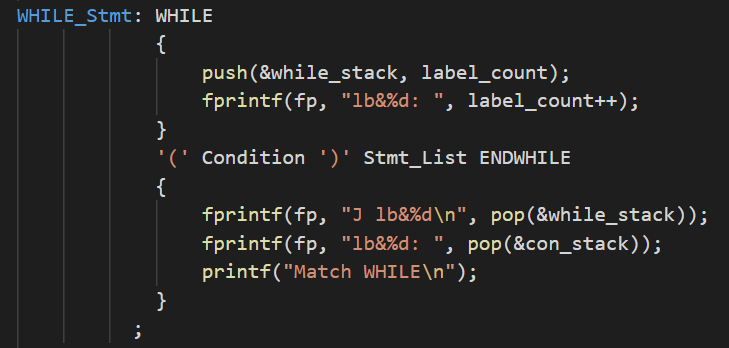
Bonus加分題部分

1. To support more complex FOR construct



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。

1. To support WHILE construct

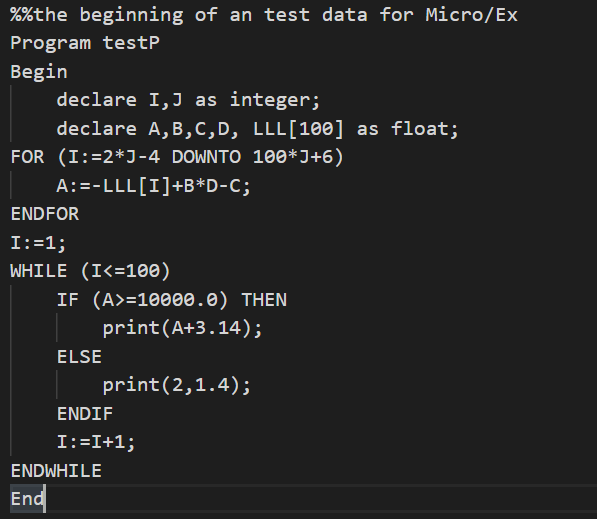


WHILE的製作比較容易，IF做好就很容易達成，因為都使用到Condition的文法。

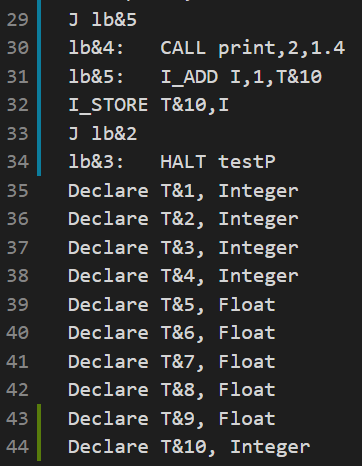
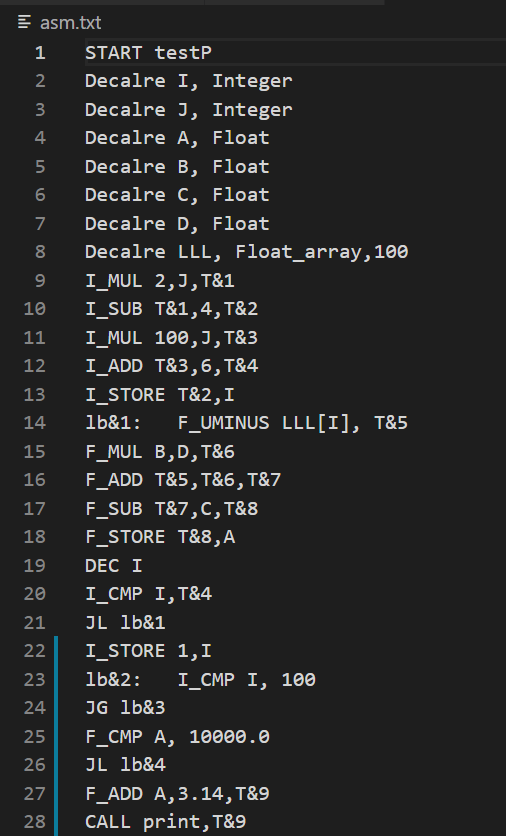
1. To support nested structure

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:



輸出asm.txt:



Copyright Claim

Copyright (C) 2021 Guan-Ying Chen