Final Project Report

> Copyright : 41047902S 鄭淮薰

Introduction

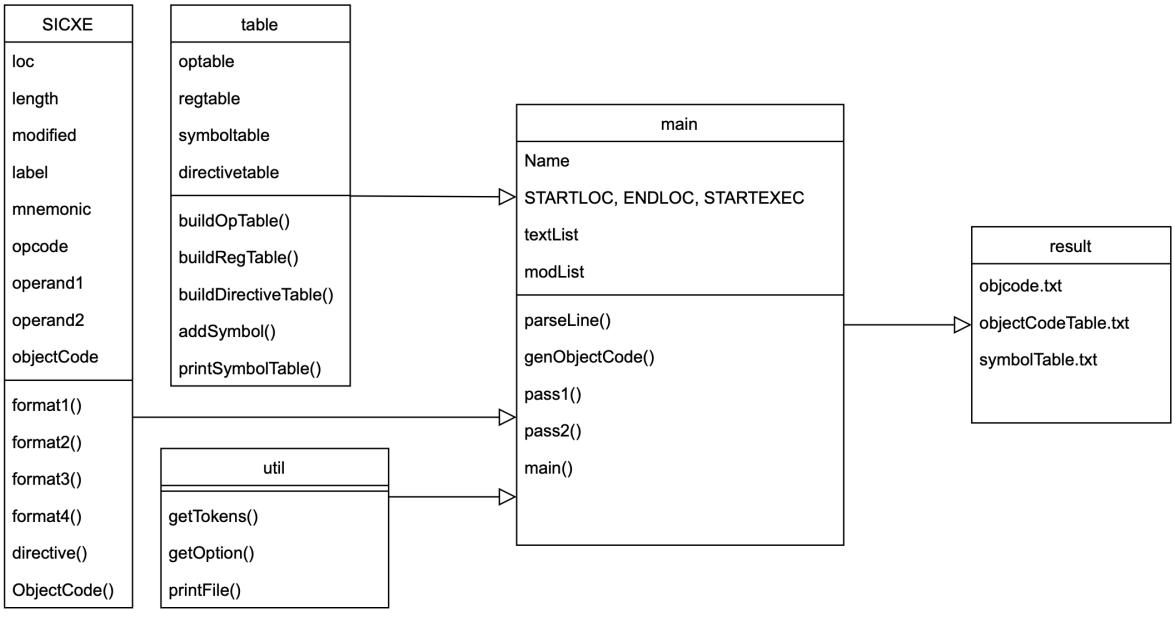
This is an SICXE assembler written by c++

Feature

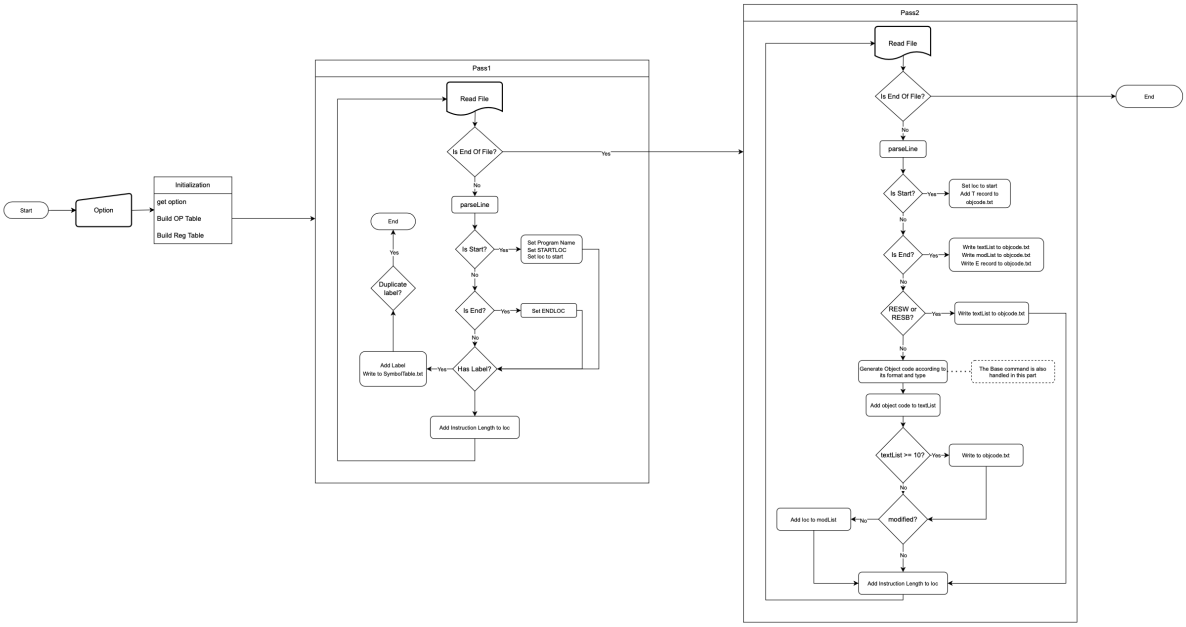
- Compile SIC/XE program to object code
- Show the symbol table after assembling

Implementation

Program Architecture



Program Flow



Programming Process

Pass1

我嘗試將指令一行一行讀入。將每一行指令分成 `label` 、 `mnemonic` 、 `operand` 三個一部分拆開來。因為 pass1 的目標是要產生 symbol table，所以首要目標是知道每行指令的大小才能計算 location ，進而取得每個 symbol 的位置。首先我先創建了 opcode table 、register table 等結構，以紀錄每個 assembler 指令的 format 及參數等資訊，利用 opcode table 查詢指令的 format 就能知道大部分指令的大小。接著處理 directive (`BYTE` 、 `WORD` 、 `RESB` 、 `RESW`) 的部分，利用 operand 計算需保留的大小。如此一來就能完成基本的 pass1 流程。

Pass2

pass2 的流程跟 pass1 有些接近，同樣將指令一行一行讀出來做處理。這部分的目標是要產生 object code，因此我定義了一個 SICXE 的 class，將每一行 source code 都打包成一個 SICXE 的型態，裡面會紀錄這行程式的所有資訊，如 location、mnemonic、operand、opcode等等，且包含一產生 object code 的方法。我將每種 format 生成 object code 的方式都包在此結構中，這樣 pass2 的主程式就會變得更簡潔，邏輯也能更清楚。最後在產生 object file 時有特別在 `RESW` 及 `RESB` 的地方做換行，以節省空間。

Others

完成以上兩部分主程式後，我也有加一些錯誤判斷，如 label 重複報錯，以及新增 option 讓程式更人性化且有彈性。

Experience

類似這樣的專案雖然乍看之下很複雜，但實際動手去做，把要做的事分成很多獨立的小部分來實作時，就變得容易許多。尤其實作 pass1 時，其實就是將檔案讀入、做字串切割再一行一行把地址計算出來，不用非常艱深的技巧就完成這個 50 % 的專案了！當然其中還是有很多小細節要注意，但是拆成小任務來實作時，便能更專注在每個功能上，細節也比較不會遺漏。在撰寫專題的過程中我還有一個很大的體悟那就是規劃程式架構的重要性，因為架構除了會影響程式的可讀性外，也會大幅影響整個專案的可變動性。我在這個專案中有嘗試的將程式碼封裝，雖然還不是封裝的非常好，有時還是會需要判斷例外的狀況，但我認為這是一個很好的練習與嘗試，希望未來可以在這方面更上手。

Feedback

我除了修習這門課外，也有修您的程式語言結構課程，我很喜歡老師您讓我們動手實作的上課方式。尤其到這後半學期，每次都很期待課堂的小實驗，讓我們自己動手操作看看，非常有有趣！而且也讓我對於那部分的知識更有感覺，而不只是將課本的知識背誦下來。感謝您這學期的教導，不僅學到課本上的知識，更是學習如何將這些內容活用。