SIC/XE Assembler Project Report

Name: 林宏昀
Student ID: 41147004S
Class: 台師大資工115
Grade: 大三

1. Project Overview

1.1 Objective

設計並實作一個SIC/XE組合語言編譯器，滿足以下要求：

- 能夠解析並執行測試資料，包含指令與組合語言指示碼。
- 產生的物件程式需符合SIC/XE架構，與教材中的格式保持一致。

1.2 Features Implemented

- **雙遍解析**：
  - 第一遍：建立符號表與計算地址。(包括Debug輸出)
  - 第二遍：以Sections為單位生成物件程式碼。
- **額外支援功能**：
  - Literals (文字常量)
  - Symbol-defining Statements (符號定義語句)
  - Program Blocks (程式區塊)
  - Control Sections (控制區段)

2. System Architecture

2.1 Modules and Functions

- `Assembler.py`：
  - **核心程式**：負責編譯與物件碼生成。
  - 提供符號檢查、位置計算與指令解析等功能。

    - 採用 `class` 提供模組化功能：
        - `pass_one`：建立符號表與指令地址並區分 `Sections` 。
        - `pass_two`：基於 `pass_one` 建立之符號表生成物件碼。
        - `write_output`：將結果寫入指定的輸出檔案。
- **`main.py`**：
    - **主程式**：負責與使用者接觸，解析命令行參數，調用 `Assembler.py` 完成組譯工作。
    - 提供多項命令列選項以便於測試與輸出管理。

3. Implementation Process

3.1 Development Log

Day 1 Fisrt 2 hours

- 設計組譯器架構，研究SIC/XE指令集與格式要求。
- 實作基本的符號表功能。

Day 1 3-4 hour

- 完成 `pass_one` ，支持計算指令位置並生成符號表。
- 增加對指令格式與標籤的檢查功能。

Day 1 5-8 hour

- 初步開發 `pass_two` 以生成物件碼，測試基礎指令格式。
- 實作文字常量與程式區塊的支援。
- 註:
    - 此時只支援單一 `Section` ，即一份程式碼只會有一個輸出檔
    - 尚未支援 `Symbol-defining Statements` (符號定義語句)

Day 2 1~3 hour

- 持續開發 `pass_two` 以生成物件碼，並測試多種指令格式。
- 實作符號定義語句的支援。

Day 2 4~6 hour

- 瘋狂Debug，檢查錯誤原因
    - 最後發現是在 `set_symbol` 函式時，迴圈判斷失誤多跑一次導致symbol value被錯誤編譯

Day 2 7~9 hour

- 基本組譯器大致完成，開始嘗試增加控制區段功能
- 增加控制區段（Control Sections）的處理功能。
- 測試不同測試資料，修正程式中的錯誤與邏輯問題。

Day3 1~5 hour

- 結構大量改動，組譯重心轉移至個別 `Section` 上
- 依照基本組譯器的方法，在Class `Section` 中時做assemble function
- 最終結構初步成形

Day3 6~10 hour

- 組譯器結構基本定型
- 大致修飾、完成 `Assemblyer.py`
- 開始設計Bonus測資及Debug

3.2 Key Challenges and Solutions

符號表前向參考問題

- 挑戰：SIC/XE 組合語言中，某些符號可能在使用時尚未定義，導致解析失敗。
- 解決方式：
    - 在 `set_symbol` 函數中實作多次迭代邏輯，逐步解析符號表中未解決的符號。
    - 嚴格檢測不合理的符號定義，並記錄解析過程中的每次更新。
    - 通過大量測試檢查前向參考處理的準確性，並不斷進行調試與修正。

程式區塊與控制區段的切換

- 挑戰：SIC/XE 支援多個程式區塊（ `Program Blocks` ）與控制區段（ `Control Sections` ），需要正確處理其間的切換與地址計算。
- 解決方式：
    - 利用專屬資料結構（如 `program_blocks` 與 `control_sections` ），分別儲存每個區塊與控制區段的指令與地址資訊。

    - 在 pass_one 中標記每個指令所屬的區塊，並在 pass_two 中依序處理。維持各區段之間的獨立性，確保組譯過程中彼此互不干擾。

如何獨立組譯個別控制區段

- 挑戰：SIC/XE 支援獨立組譯控制區段，這增加了組譯器的複雜性與維護難度。
- 解決方式：
    - 定義 `Section` 資料結構，每個區段的指令（instructions）與符號（symbols）分別儲存。
    - 在 `assemble` 中依序組譯每個區段，並生成對應的物件碼。獨立處理每個區段，便於排查錯誤與調試，且提升系統可擴展性。

4. Enhancements and Bonus Features

- **Literals處理**：支持以 = 開頭的文字常量，並在適當位置插入。
- **符號定義語句**: 支持如 `BUFFEND-BUFFER` 等符號定義語句
- **多控制區段支持**：能分別組譯多個區段，並生成相應的物件程式碼。
- **改進Debug輸出格式**：物件程式碼清晰分段，方便檢查與調試。

5. Lessons Learned

5.1 Skills Acquired

- 熟悉 SIC/XE 架構，包含指令格式、程式區塊、控制區段等核心概念。
- 使用 多遍解析技術 與符號表的管理方法，包括前向參考的解決方案。
    - 儘管可能並非完全按照真實組譯器的解決方式，但能有效解決前向參考問題
- 使用 Python 實現資料結構的設計與檔案操作，特別是在處理多層次的資料時。

5.2 Insights

- **模組化設計**：將功能劃分為多個模組，不僅提升了程式碼的可讀性與維護性，也使得不同部分的組譯邏輯更加清晰。
- **錯誤處理與測試**：在處理大型輸入資料時，嚴謹的錯誤檢測機制與豐富的測試用例，對於保障程式的穩定性至關重要。

- **可擴展性**：透過獨立的資料結構管理每個控制區段與程式區塊，實現了區段的獨立組譯與輸出功能，大幅提高程式的靈活性。

6. Test Cases and Results

以下是測試資料的執行結果：

1. **Basic.asm**

```
. from textbook Figure2.5
COPY    START   0
FIRST   STL     RETADR
        LDB     #LENGTH
        BASE    LENGTH
CLOOP   +JSUB   RDREC
        LDA     LENGTH
        COMP    #0
        JEQ     ENDFIL
        +JSUB   WRREC
        J       CLOOP
ENDFIL  LDA     EOF
        STA     BUFFER
        LDA     #3
        STA     LENGTH
        +JSUB   WRREC
        J       @RETADR
EOF     BYTE    C'EOF'
RETADR  RESW    1
LENGTH  RESW    1
BUFFER  RESB    4096

RDREC   CLEAR   X
        CLEAR   A
        CLEAR   S
        +LDT    #4096
RLOOP   TD      INPUT
        JEQ     RLOOP
        RD      INPUT
        COMPR    A,S
        JEQ     EXIT
        STCH    BUFFER,X
        TIXR    T
        JLT     RLOOP
EXIT    STX     LENGTH
        RSUB
INPUT   BYTE    X'F1'
.
.     SUBROUTINE
.
WRREC   CLEAR   X
        LDT     LENGTH
WLOOP   TD      OUTPUT
        JEQ     WLOOP
        LDCH    BUFFER,X
        WD      OUTPUT
        TIXR    T
        JLT     WLOOP
        RSUB
OUTPUT  BYTE    X'05'
        END     FIRST

```

output:

```
HCOPY  000000001077
T0000001D17202D69202D4B10103603202629000033200074B10105D3F2FEC032010
T00001D130F20160100030F200D4B10105D3E2003454F46
T0010361DB410B400B44075101000E32019332FFADB2013A00433200857C003B850
T0010531D3B2FEA1340004F0000F1B410774000E32011332FFA53C003DF2008B850
T001070073B2FEF4F000005
M00000705
M00001405
M00002705
E000000
```

2. **LiteralSDF.asm**

```
1  TEST    START   1000
2  BUFFER  RESW    5
3  BUFFEND EQU     BUFFER+10
4  L1      LDA     =C'EOF'
5          LDCH    =X'F1'
6          STA     BUFFER
7          END     TEST
8
```

Output:

```
HTEST  00100000001C
T00100F0D0320065320060F2FE8454F46F1
E001000
```

3. **Porgram_Block.asm**

```
. from textbook Fig2.11
COPY    START   0
FIRST   STL     RETADR
CLOOP   JSUB    RDREC
        LDA     LENGTH
        COMP    #0
        JEQ     ENDFIL
        JSUB    WRREC
        J       CLOOP
ENDFIL  LDA     =C'EOF'
        STA     BUFFER
        LDA     #3
        STA     LENGTH
        JSUB    WRREC
        J       @RETADR
        USE     CDATA
RETADR  RESW    1
LENGTH  RESW    1
            USE CBLKS
BUFFER  RESB    4096
BUFFEND EQU     *
MAXLEN  EQU     BUFEND-BUFFER

        USE
RDREC   CLEAR   X
        CLEAR   A
        CLEAR   S
        +LDT    #MAXLEN
RLOOP   TD      INPUT
        JEQ     RLOOP
        RD      INPUT
        COMPR   A,S
        JEQ     EXIT
        STCH    BUFFER,X
        TIXR    T
        JLT     RLOOP
EXIT    STX     LENGTH
        RSUB
        USE     CDATA
INPUT   BYTE    X'F1'
            USE
WRREC   CLEAR   X
            LDT LENGTH
WLOOP   TD      =X'05'
        JEQ     WLOOP
        LDCH    BUFFER,X
        WD      =X'05'
        TIXR    T
        JLT     WLOOP
        RSUB
        USE     CDATA
        LTORG
        END     FIRST
```

Output:

```
HCOPY  000000001071
T0000001E1720634B20210320602900003320064B203B3F2FEE0320550F2056010003
T00001E090F20484B20293E203F
T0000271DB410B400B44075101071E32038332FFADB2032A00433200857A02FB850
T000044093B2FEA13201F4F0000
T00006C01F1
T00004D19B410772017E3201B332FFA53A016DF2012B8503B2FEF4F0000
T00006D04454F4605
E000000
```

4. **CSECT.asm**

```
. from textbook Fig2.15
COPY    START   0
        EXTDEF  BUFFER,BUFEND,LENGTH
        EXTREF  RDREC,WRREC
FIRST   STL     RETADR
CLOOP   +JSUB   RDREC
            LDA LENGTH
            COMP        #0
            JEQ ENDFIL
            +JSUB       WRREC
            J   CLOOP
ENDFIL  LDA     =C'EOF'
            STA BUFFER
        LDA     #3
        STA     LENGTH
        +JSUB   WRREC
        J       @RETADR
RETADR  RESW    1
LENGTH  RESW    1
            LTORG
BUFFER  RESB    4096
BUFEND  EQU     *
MAXLEN  EQU     BUFEND-BUFFER

RDREC   CSECT
        EXTREF  BUFFER,LENGTH,BUFEND
        CLEAR   X
        CLEAR   A
        CLEAR   S
        LDT     MAXLEN
RLOOP   TD      INPUT
        JEQ     RLOOP
        RD      INPUT
        COMPR   A,S
        JEQ     EXIT
        +STCH   BUFFER,X
        TIXR    T
        JLT     RLOOP
EXIT    +STX    LENGTH
            RSUB
INPUT   BYTE    X'F1'
MAXLEN  WORD    BUFEND-BUFFER

WRREC   CSECT
        EXTREF  BUFFER,LENGTH
        CLEAR   X
        +LDT    LENGTH
WLOOP   TD      =X'05'
        JEQ     WLOOP
        +LDCH   BUFFER,X
        WD      =X'05'
        TIXR    T
        JLT     WLOOP
```

```
54          RSUB
55          END    FIRST
56
```

Output:

1.

```
HCOPY  000000001033
DBUFFER000033BUFEND001033LENGTH00002D
RRDREC WRREC
T0000001D1720274B10000003202329000033200074B1000003F2FEC0320160F2016
T00001D0D0100030F200A4B1000003E2000
T00003003454F46
M00000405+RDREC
M00001105+WRREC
M00002405+WRREC
E000000
```

2.

```
HRDREC 00000000002B
RBUFFERLENGTHBUFEND
T0000001DB410B400B440772FF7E3201B332FFADB2015A00433200957900000B850
T00001D0E3B2FE9131000004F0000F1000000
M00001805+BUFFER
M00002105+LENGTH
M00002906-BUFFER
M00002906+BUFEND
E
```

3.

```
HWRREC 00000000001C
RBUFFERLENGTH
T0000001CB41077100000E32012332FFA53900000DF2008B8503B2FEE4F000005
M00000305+LENGTH
M00000D05+BUFFER
E
```

5. **Testcase.asm**

```
MAIN    START   1000
        EXTDEF  LENGTH, BUFFER
        .Not recommanded,but we support blank after,
        EXTREF  DATA
        LDA     =C'EOF'
        STA     BUFFER
        LDA     BUFFER+10
        BASE    LENGTH
        LDCH    =X'F1'
        USE     BLOCK1
FIRST   STL     RETADR
        LDB     #LENGTH
        BASE    LENGTH
        USE     BLOCK2
SECOND  +LDA     DATA
        STA     BUFFER
        USE     BLOCK1
RETADR  RESW    1
LENGTH  WORD    30
BUFFER  RESB    10

WRREC   CSECT
        EXTREF  BUFFER,LENGTH
        EXTDEF DATA
        CLEAR   X
        +LDT    LENGTH
WLOOP   TD      =X'05'
        JEQ     WLOOP
        +LDCH   BUFFER,X
        WD      =X'05'
        TIXR    T
        JLT     WLOOP
DATA    WORD 20
        RSUB
        END     FIRST
```

Output:

1.

```
HMAIN  00100000002D
DLENGTH00001EBUFFER001018
RDATA
T0010000C03201F0F2012032019532019
T00100C06172003694000
T00102607031000000F2FEB
T0010150300001E
T00102204454F46F1
M00102705+DATA
E001000
```

2.

```
HWRREC 00000000001F
DDATA  000014
RBUFFERLENGTH
T0000001EB41077100000E32015332FFA53900000DF200BB8503B2FEE0000144F0000
T00001E0105
M00000305+LENGTH
M00000D05+BUFFER
E
```