

期末程式專題報告

衛教 114 41005001E 徐偉翔

程式架構：

一、資料結構：

運用 python 的 dictionary 功能建立 op_table 及 register_table 存放 assembler 所需要用到的內容

二、初始化：

1. symbol_table：用來儲存 label 與對應的記憶體位置
2. op_list：詳細儲存每一行程式的 format、address...等方便 pass 2 使用

三、Pass 1：

1. 找到"START"，將其位置設定為起始位置，並更新程式名稱
2. 逐行讀取直到讀取到"END"，每次讀取更新現在位置：
 - ▲需判斷"BYTE"、"WORD"、"RESB"、"RESW"前面有沒有符號名稱
 - ▲需針對重複命名進行除錯（有建立一個 symbol set 專門存放所有 label，用來判斷有沒有重複）
 - (1)當遇到"BYTE"時：針對不同形式（"C"、"X"、一般整數）設計不同的處理方式，同時新增 symbol、address 到 symbol_table 中，並將其內容存放到對應指令的存放區
 - (2)當遇到"WORD"時：新增 symbol、address 到 symbol_table 中，並將其內容存放到對應指令的存放區
 - (3)當遇到"RESB"、"RESW"時：新增 symbol、address 到 symbol_table 中
 - (4)當遇到"BASE"時：更新 base label
 - (5)當遇到"END"時：更新 end label，同時設定結束位置
 - (6)其他：以"+"判斷是否為 format 4，同時如果該行程式的第一個片段不是指令，而是 symbol，新增到 symbol_table 中，並將其內容存放到對應指令的存放區
3. 將程式名稱、起始位置、結束位置、程式長度及 symbol table 打印出來，即完成 pass 1

四、Pass 2：

結合全部 object code 產生 object program，並儲存在 output.txt 中

1. H record：

"H" + 程式名稱 + 起始位置(6bytes) + 程式長度(6bytes)
2. T record：

"T" + 程式起始位置 + 程式長度 + 每一行程式的 object code

▲程式長度不可超過 hex("1E")

根據相對或絕對位置及 opcode 產生 object code (需要判斷 n, l, x, b, p, e 的值):

(1)當遇到"RESB"、"RESW" 或程式長度過長時:

將目前 T record 印出, 換行並重新創造一個 T record

(2)當遇到 format 2 時:

● Register 的值可以看 register_table

opcode (8 bits) + first register (4 bits) + second register (4 bits)

(3)當遇到 format 3 時:

n, i: 由定址法判斷

("#"=>n=0, i=1; "@"=>n=1, i=0; " "=>n=1, i=1)

x: 根據該行程式後面有無",x" 判斷

b, p: 根據哪個 relative 判斷 (計算 displacement)

e=0



(4)當遇到 format 4 時:

n, i: 由定址法判斷

("#"=>n=0, i=1; "@"=>n=1, i=0; " "=>n=1, i=1)

x: 根據該行程式後面有無",x" 判斷

b=0, p=0

e=1



3. M record:

如果該行程式是 format 4 且不是立即地址, 則印出

"H" + 程式位置+1(6bytes) + "05"

4. E record:

"E" + 起始位置(6bytes)

程式功能:

僅完成一般 2 pass 的 assembler, 並無製作任何 bonus

執行方式:

列於 README 中

組譯結果：

```
Please enter the file: test.sic

process name: COPY
process start address: 0x000000
process end address: 0x001077
process length: 4215 Bytes

SYMBOL ADDRESS
FIRST 000000
CLOOP 000006
ENDFIL 00001A
EOF 00002D
RETADR 000030
LENGTH 000033
BUFFER 000036
RDREC 001036
RLOOP 001040
EXIT 001056
INPUT 00105C
WRREC 00105D
WLOOP 001062
OUTPUT 001076

HCOPY 000000001077
T0000001D17202D69202D4B1010360320262900003320074B10105D3F2FEC032010
T00001D130F20160100030F200D4B10105D3E2003454F46
T0010361DB410B400B44075101000E32019332FFADB2013A00433200857C003B850
T0010531D3B2FEA1340004F0000F1B410774000E32011332FFA53C003DF2008B850
T001070073B2FEF4F000005
M00000705
M00001405
M00002705
E0000000

Press Enter to quit.
```

專題日誌：

- 11/30 - 12/4：python 複習與 assembler 架構設計
- 12/5 - 12/11：pass 1 製作與 debug，同時針對架構作調整（這裡花了最多時間，一開始設計的架構過於簡陋，很多東西都沒有判斷到，所以基本上重做）
- 12/20 - 12/22：pass 2 製作與 debug，同時針對架構作調整

面臨挑戰：

1. 專題 deadline 鄰近期末考，所以無法完全專心投入製作專題，還須兼顧考試，因此花了大量時間
2. 查找資料發現 python 對於字串的處理功能非常強大，因此選擇使用 python 來製作此次專題，但是對於 python 並不是太熟悉，花了一段時間重新複習
3. Assembler 製作需要反覆在字串與數值之間來回轉換，腦袋容易打結

參考資料：

一些基本的 python 功能與 assembler 限制有詢問 ChatGPT 該如何執行，架構與邏輯判斷皆是自己建構與規劃