微算機原理與實作

指導教授：蔡章仁

Project Report

系級:電機二 B
學號:112501561
姓名:陳冠義

※程式功能：

此程式會讀取 8051 組合語言指令，並使用查表法比對各種指令格式，自動轉換成對應的機器碼，最後輸出為純文字檔。

※流程介紹：

1. 建立可供查詢且具有彈性格式的表格。

(1)Data transfer instructions：

```
import re

# 定義 pattern 和對應的處理函式
patterns = [
    #Data transfer instructions
    {
        #MOV @Ri, #imm
        'pattern': r'^\s*MOV\s*+@R(\d)\s*,\s*+#0*([0-9A-Fa-f]+)H\s*$',
        'action': lambda m: f"{format(0x76 + int(m.group(1)), '02X')} {format(int(m.group(2), 16), '02X')} "
    },
    {
        #MOV @Ri, direct
        'pattern': r'^\s*MOV\s*+@R(\d)\s*,\s*+0*([0-9A-Fa-f]+)H\s*$',
        'action': lambda m: f"{format(0xA6 + int(m.group(1)), '02X')} {format(int(m.group(2), 16), '02X')} "
    },
    {
        #MOV direct, Rn
        'pattern': r'^\s*MOV\s*+0*([0-9A-Fa-f]+)H\s*,\s*+R(\d)\s*$',
        'action': lambda m: f"{format(0x88 + int(m.group(2)), '02X')} {format(int(m.group(1), 16), '02X')} "
    },
    {
        #MOV Rn, #immediate
        'pattern': r'^\s*MOV\s*+R(\d)\s*,\s*+#0*([0-9A-Fa-f]+)H\s*$',
        'action': lambda m: f"{format(0x78 + int(m.group(1)), '02X')} {format(int(m.group(2), 16), '02X')} "
    },
```

此部分程式對應資料移轉的 8051 指令，其中'pattern'為比對的標的，'action'為比對成功時要存入輸出的字串。為了進一步增加程式的可用範圍，對於所有 8051 指令元素的間隔中，我都放了\s*指令，以確保指令的 coding type 不同而讀取不了的情況不會發生。另外我也在讀取指令時進行了資料整理，也就是字元讀取成字元，數字讀取為 16 進位數字並且把第一個 0 去除以防止 0A2H 的情況。若比對成功的話，我會存對應機器語言到輸出字串中，並對不足 2 位數的 16 進位數字資料進行補 0，以彌補剛才去掉 0，但是卻如 02H 的情況。

(2)Arithmetic & logic instructions：

```
    #Arithmetic & logic instructions
    {
        #ADD A, #imm
        'pattern': r'^\s*ADD\s*+A\s*,\s*+#0*([0-9A-Fa-f]+)H\s*$',
        'action': lambda m: f"24 {format(int(m.group(1), 16), '02X')} "
    },
    {
        #SUBB A, direct
        'pattern': r'^\s*SUBB\s*+A\s*,\s*+0*([0-9A-Fa-f]+)H\s*$',
        'action': lambda m: f"95 {format(int(m.group(1), 16), '02X')} "
    },
    {
        #ANL direct, #imm
        'pattern': r'^\s*ANL\s*+0*([0-9A-Fa-f]+)H\s*,\s*+#0*([0-9A-Fa-f]+)H\s*$',
        'action': lambda m: f"53 {format(int(m.group(1), 16), '02X')} {format(int(m.group(2), 16), '02X')} "
    },
    {
        #XRL direct, A
        'pattern': r'^\s*XRL\s*+0*([0-9A-Fa-f]+)H\s*,\s*+A\s*$',
        'action': lambda m: f"62 {format(int(m.group(1), 16), '02X')} "
    },
```

此部分程式對應邏輯運算的 8051 指令。

(3)Branching instructions：

```
    #Branching instructions
    {
        #CJNE A, #imm, offset
        'pattern': r'^\s*CJNE\s*+A\s*,\s*+#0*([0-9A-Fa-f]+)H\s*,\s*+0*([0-9A-Fa-f]+)H\s*$',
        'action': lambda m: f"B4 {format(int(m.group(1), 16), '02X')} {format(int(m.group(2), 16), '02X')} "
    },
    {
        #DJNZ direct, offset
        'pattern': r'^\s*DJNZ\s*+0*([0-9A-Fa-f]+)H\s*,\s*+0*([0-9A-Fa-f]+)H\s*$',
        'action': lambda m: f"D5 {format(int(m.group(1), 16), '02X')} {format(int(m.group(2), 16), '02X')} "
    },
```

此部分程式對應分支跳轉的 8051 指令。

(4)Null line：

```
    #Null line
    {
        'pattern': r'^\s*$',
        'action': lambda m: f""
    },
]
```

此部分程式沒有對應 8051 指令，是為了防止輸入有空行所做的防呆設計。

2. 從 test01.txt 讀取資料轉換後輸出至 test01-out.txt

(1)讀取資料：

```
output_lines = []

# 讀取 test01.txt
with open('test01.txt', 'r') as f:
    lines = f.readlines()
```

使用 open()指令以及參數'r'對相同根目錄的 test01.txt 進行讀取，並存入 lines 串列。

(2)逐行比對：

```
#逐行分析
for line in lines:
    line = line.strip()
    matched = False
    for p in patterns:
        match = re.match(p['pattern'], line)
        if match:
            result = p['action'](match)
            output_lines.append(result)
            matched = True
            break
    if not matched:
        output_lines.append(f"\n未識別指令：{line}\n")
```

根據剛才的'patterns'進行比對，若比對成功的話就把'actions'對應的內容存到 output_lines 串列，對失敗的話就把比對失敗的指令前加上"為識別指令："，並存入 output_lines 串列，用這個方法可以在程式編寫的時候自動記錄錯誤部分。

(3)輸出資料：

```
# 輸出到 test01-out.txt
with open('test01-out.txt', 'w') as f:
    for out_line in output_lines:
        f.write(out_line)
```

和讀取資料類似，使用 open()指令打開 test01-out.txt，但是將參數改為'w'以進行寫入檔案的動作，也就是把剛才處理好 output_lines 串列的每個元素寫入檔案。