# 組合語言與系統程式作業二報告

102502559 資工2B 吳承霖

## 程式流程及原理

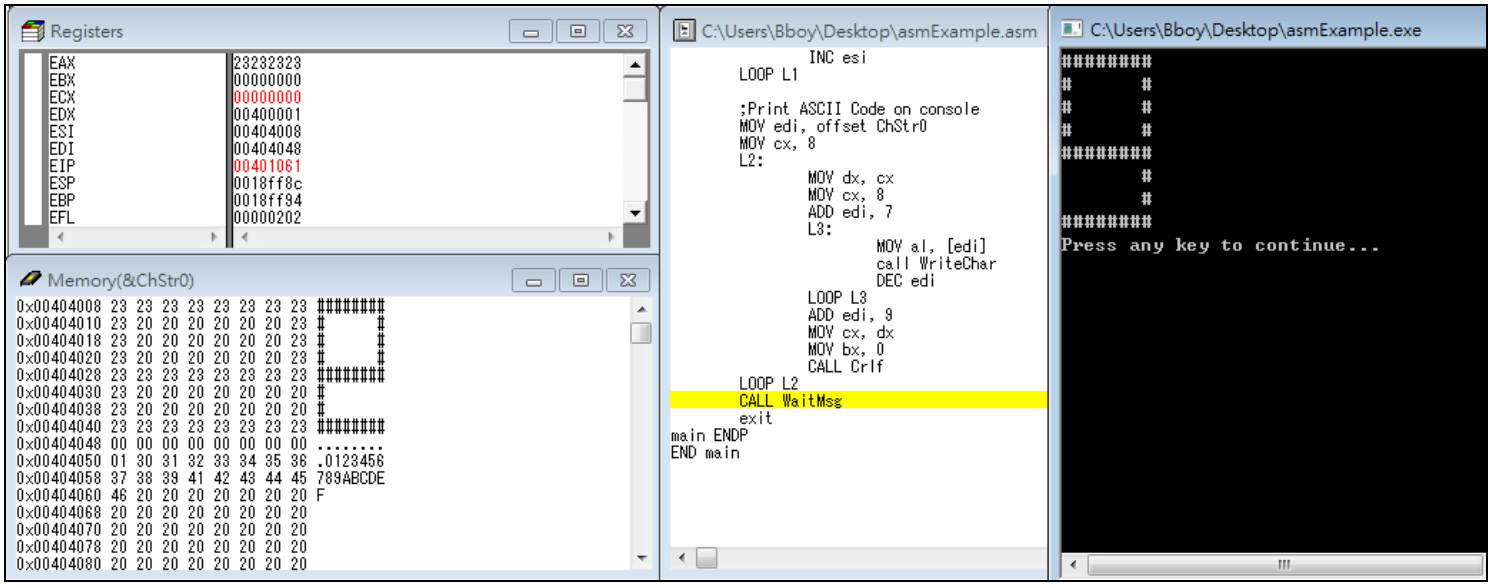
這次作業的流程與作業二大致相同，唯一不同處在於我們使用Macro取代Procedure來做有關字元轉換的處理。Macro與Procedure的主要差別，在於Macro在組譯的時候，組譯器僅僅將Macro的程式碼複製貼上在呼叫處，其效果等同於我們在呼叫處打上Macro內的程式碼，而組譯產生的Machine Code並沒有達到縮減的效果(每個呼叫處都有一份相同的Machine code)。至於Procedure，程式執行到呼叫Procedure時會執行Jump的動作，所以組譯成Machine Code時，Procedure內的Machine code在整份程式碼內只有一份。

```
change MACRO
    LOCAL L1, L2, L3
    mov cx, 8
    mov ebx, 0
    L1:
        MOV eax, [esi]
        BT eax, ebx
        JC L2
        MOV al, 20h
        MOV [edi], al
        JMP L3
    L2:
        MOV al, 23h
        MOV [edi], al
    L3:
        INC ebx
        INC edi
    LOOP L1
ENDM
```

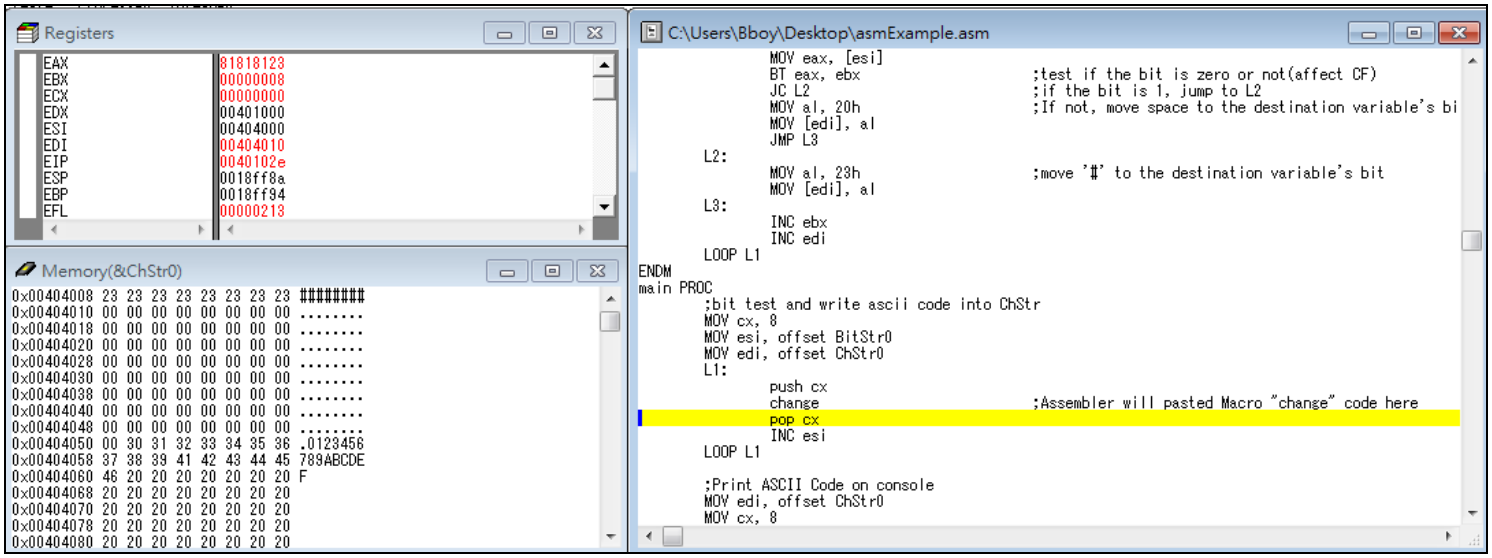
右圖程式碼中，我們可以發現Macro的第一行出現LOCAL這個指令，他限定了L1、L2、L3這三個標籤的scope只在於change這個Macro內，如此一來重複呼叫同一個Macro便不會產生重複的同名Label。

程式一開始宣告初始值(學號末一碼)進入BitStr，以1和0分別代表數字圖案的網格 為＃或是空白，那是如何判斷一個BYTE內的所有Bit為0或1呢？在change這個 Macro中，我使用了”BT”這個指令，右圖為此次作業核心處理的程式碼片段，以”BT eax, ebx”解釋，程式會執行”檢查eax暫存器的第ebx，如果該bit值為1，則改變CF為1”。執行完BT後我緊接著依照CF做條見判斷，如果CF為1，表示此 變數的該bit為1，會跳至L2並且把＃寫進ChStr中。

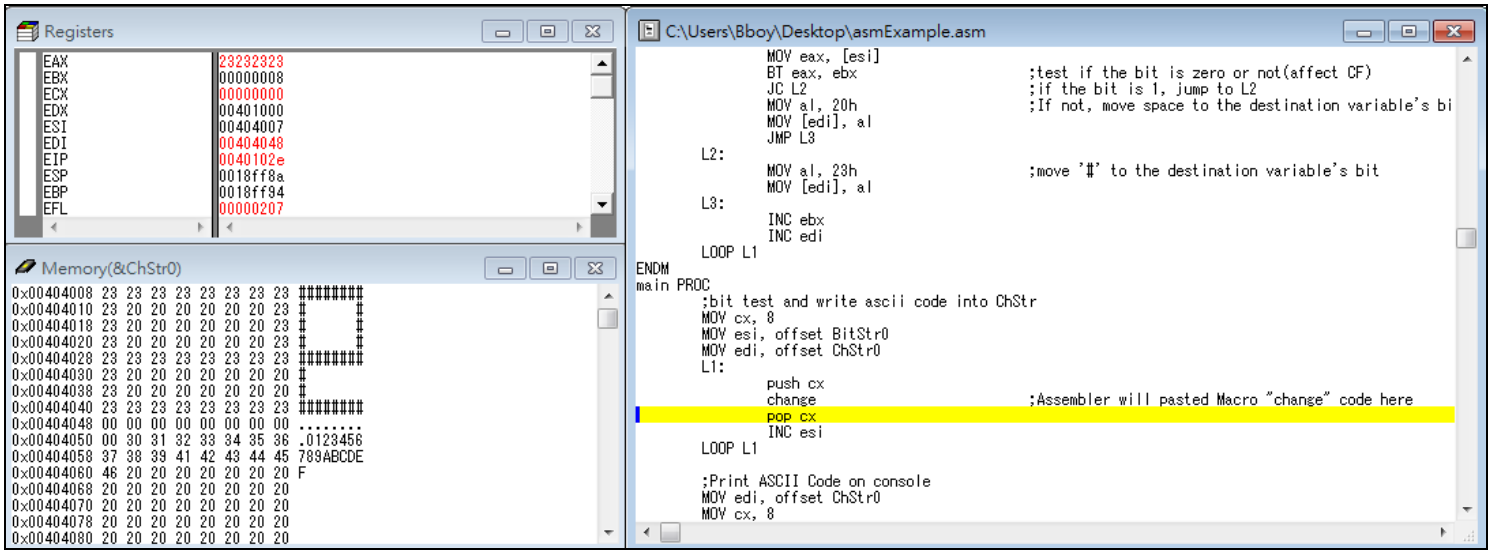
## 完成畫面截圖



## 記憶體＆暫存器狀態圖




這是第一次呼叫change這個Macro結束後的截圖，可惜的是Windbg並沒有辦法讓暫存器和記憶體狀態在MAcro內逐行顯示，所以接下來各個狀態直接來到Macro結束後的數值，我們可以看到記憶體位置404008h到40400fh皆存入23，在ascii顯示成'#'，這是第一次Macro內處理完的結果。




這是所有字元皆經過轉換並存入ChStr後的狀態圖，我們可以發現在記憶體中，9為左右顛倒的，所以在Write到Console的時候必須使用倒敘的方式印出。印出的結果請見"程式完成截圖"。

## 心得與感想

陸續學到Macro以及Procedure，剛開始對兩者的差別有些困惑，也不懂他在高階程式語言的應用有哪些。經過釐清後發現Macro有點像高階程式語言的"Include"，至於Procedure則有點像高階程式語言的Function，寫組語的我們也必須知道Macro雖然有助於縮短組合語言程式碼的長度，但組譯產生的Machine Code長度卻沒有縮短，因此，相較於Procedure，並沒有有效率地減少程式所消耗的資源。當然他也確實有其方便處，若能靈活且適當地運用這兩者，我相信將有助於我們寫出來的程式品質。