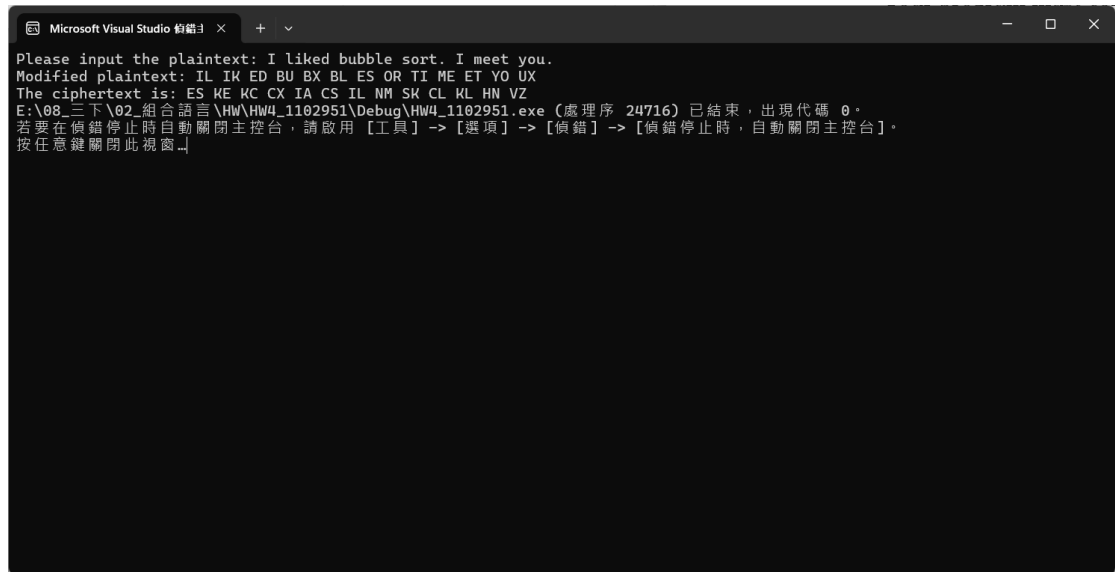


Assembly Language and Laboratory

Homework 4

學號：1102951 姓名：劉宗翰

1. 程式執行畫面：



```
Microsoft Visual Studio 解編 3 x + v
Please input the plaintext: I liked bubble sort. I meet you.
Modified plaintext: IL IK ED BU BX BL ES OR TI ME ET YO UX
The ciphertext is: ES KE KC CX IA CS IL NM SK CL KL HN VZ
E:\08_二下\02_組合語言\HW\HW4_1102951\Debug\HW4_1102951.exe (處理序 24716) 已結束，出現代碼 0。
若要在偵錯停止時自動關閉主控台，請啟用【工具】->【選項】->【偵錯】->【偵錯停止時，自動關閉主控台】。
按任意鍵關閉此視窗
```

2. 程式說明：

程式中刻意只宣告兩個字串，以降低記憶體的使用量。利用在 main 使用 lea 指令宣告兩字串的起始位置，並且儲存於 stack 中，再於 PlayFair procedure 中利用 EBP 的相對位置，取用於 stack 中兩字串的起始位置，以達成 passing by reference 的題目要求。利用在 PlayFair procedure 多個 push 和 pop 指令，將 local variable 存於 Stack 中，以達成 using local variable 的題目要求。接下來介紹所有 Procedure 之用途：

(1) inputPlaintext：輸入字串，存入 plaintext 中。

- (2) `checkCharacter`: 將字串除英文字母以外的字元刪除，修改完的字串利用 `str_copy` 與 `str_length` 複製並計算長度存入 `plaintext` 中。
- (3) `addX`: 一次輸出兩個字元，首先將第一個字元載入 `AL`，第一個字元載入 `AH`，若兩字元相同，則加入 `X`，並使 `ESI+1` (`ESI` 為刪除英文字母以外的字元後的字串起始位置)；若 `AL` 為 0，則代表字串 (`ESI`) 剛好到終點位置；若 `AH` 為 0，則代表字串 (`ESI`) 差一個 byte 到終點位置，所以須加上 0，又因 `AL=0` 時，`AH` 必然為 0，所以三個判斷式的前後順序必須為 `cmp AL,0 -> cmp AH,0 -> cmp AL,AH`，修改完的存入 `ciphertext` 字串中。
- (4) `PlayFair`: 利用上述修改完的字串進行加密，接下來介紹重要的 label 與判斷條件：
- i. `changeJToI`: 因為 `PlayfairKey` 為 5x5 的矩陣，且因英文字母有 26 個，所以 `I` 和 `J` 必須共用矩陣中的位置，相應的字串中的 `J` 也先需轉換為 `I` (相反亦可)，所以我利用 `loop` 字串的長度來檢視字串中的所有字元，誘因後續還須利用 `loop` 字串的長度做加密，因此將 `ECX` 值 (為字串長度) 在此 `loop` 前 `push` 進 `stack` 中。

ii. encrypt: 因在 changeJToI 的 loop 中,ESI 值(ciphertext 的起始位置)與 ECX 值有遭到變動,所以利用先前存進 stack 中的地址與數值恢復,在 encrypt 中,每次處理兩個字元,因此利用 shr ECX,1 除二以調整至適當的 loop 次數。在此 loop 中,我會先找到兩字元各自在 PlayfairKey 中的位置,但因為我的設定是將第一個字元存至 AX 中,第二個字元存至 BX 中,若先做第一個字元,再做第二個字元,會造成 AX 的數值在運算的過程中遭到覆蓋,因此我會先處理第二個字串,在處理第一個字串(寫這個文件想到其實可以將結果 push 至 stack 中,但懶得改了 XD),這個功能是調用了 findRowCol 去達成的。之後我們對兩字元在 PlayfairKey 中的相對位置,去判斷應該要做什麼動作,例如若為同行或同列,則調用 addColumnOrRow 去做加密結果位置的運算;若都沒有同行或同列,則交換兩字元之餘數。在之後調用 findPlayfairKey 來去找出該結果位置在 PlayfairKey 中的元素為何,並且存於 ciphertext 中。最後再印出兩個字元和一個空白格。

- iii. findRowCol：吃 ESI 參數，並將結果存於 AX。利用儲存 ECX 進 stack 並將其重新設定為 25 (PlayfairKey 的元素數量)，並利用 repne scasb 找出該字元在 PlayfairKey 中的位置，並運算得出其 Row 和 Column 的位置，並存於 AX (AL 存 Row，AH 存 Column)。
- iv. addColumnOrRow：吃 DX 參數，並將結果存於 DX。對 DL 和 DH+1，並判斷是否超過 5 (PlayfairKey 矩陣為 5x5)。
- v. findPlayfairKey：吃 BX 參數，並將結果存於 AL，利用 $\text{Row} \times 5 + \text{Column}$ 來得知該位置在 PlayfairKey 中的元素為何，並且將其存於 AL 中。

3. 程式碼：

```
1.  INCLUDE Irvine32.inc
2.  BUFMAX = 128
3.  .data
4.  prompt1                BYTE "Please input the plaintext: ",0
5.  prompt2                BYTE "Modified plaintext: ",0
6.  prompt3                BYTE "The ciphertext is: ",0
7.  plaintext              BYTE BUFMAX+1 DUP(0)
8.  ciphertext              BYTE BUFMAX+1 DUP(0)
9.  PlayfairKey             BYTE    'M', 'O', 'N', 'A', 'R', 'C', 'H', 'Y',
    'B', 'D', 'E', 'F', 'G', 'I', 'K', 'L', 'P', 'Q', 'S', 'T', 'U', 'V',
    'W', 'X', 'Z'
10. .code
11. main PROC
12. call inputPlaintext
13. call checkCharacter
14. call addX
15. lea ESI,[ciphertext]
16. lea EDI,[PlayfairKey]
17. push ESI
18. push EDI
19. call PlayFair
20. exit
21. main ENDP
22.
23. inputPlaintext PROC
24. mov EDX,OFFSET prompt1          ; print "Please input the plaintext:
    "
25. call WriteString
26. mov ECX,BUFMAX
27. mov EDX,OFFSET plaintext        ; input string
28. call ReadString
29. mov ECX,EAX                    ; loop the size of input string
30. ret
31. inputPlaintext ENDP
32.
33. checkCharacter PROC
```

```

34. mov EDX,OFFSET prompt2          ; print "Modified plaintext: "
35. call WriteString
36. mov ESI,OFFSET plaintext
37. mov EDI,OFFSET ciphertext
38. charCheck:
39. mov AL,[ESI]
40. cmp AL,41h          ; skip if the character is below then 'A'
41. jb skip
42. cmp AL,5bh          ; copy if the character is between 'A' to 'Z'
43. jb copy
44. cmp AL,61h          ; skip if the character is between '[' to '`'
45. jb skip
46. cmp AL,7bh
    ; Change to Uppercase if the character is between 'a' to 'z'
47. jb toUpperCase
48. cmp AL,7fh          ; skip if the character is between '{' to '~'
49. jb skip
50. toUpperCase:
51. sub AL,20h
52. copy:
53. mov [EDI],AL
54. inc EDI
55. skip:
56. inc ESI
57. loop charCheck
58. invoke str_length,ADDR ciphertext
59. mov ECX,EAX
60. invoke str_copy,ADDR ciphertext,ADDR plaintext
61. ret
62. checkCharacter ENDP
63.
64. addX PROC
65. mov ESI,OFFSET plaintext
66. mov EDI,OFFSET ciphertext
67. twinCheck:
68. mov AL,[ESI]
69. mov AH,[ESI+1]
70. mov [EDI],AL

```

```

71. call WriteChar
72. cmp AL,0 ; if the AL = 0, then finish
73. je theAddXEnd
74. cmp AH,0 ; if the AH = 0, then needs to add 'X' into ciphertext
75. je addLastX
76. cmp AL,AH; if the AL != AH, then write it into ciphertext
77. jne notTwin
78. mov AL,'X'
79. mov [EDI+1],AL
80. call WriteChar
81. jmp finishTwinCheck
82. notTwin:
83. mov [EDI+1],AH
84. mov AL,AH
85. call WriteChar
86. inc ESI
87. finishTwinCheck:
88. mov AL,' '
89. call WriteChar
90. inc ESI
91. add EDI,2
92. loop twinCheck
93. addLastX:
94. mov AL,'X'
95. mov [EDI+1],AL
96. call WriteChar
97. theAddXEnd:
98. invoke str_length,ADDR ciphertext
99. mov ECX,EAX
100. invoke str_copy,ADDR ciphertext,ADDR plaintext
101. call Crlf
102. ret
103. addX ENDP
104.
105. PlayFair PROC
106. mov EDX,OFFSET prompt3
107. call WriteString
108. mov ESI,[EBP-16] ; ciphertext starting address

```

```

109. push ECX
110. changeJtoI:
    ; changing J to I in the ciphertext, using loop due to there may have
    more then one 'J' in the string
111. mov AL,[ESI]
112. cmp AL,'J'
113. jne notJ
114. dec AL
115. mov [ESI],AL
    ; changing J to I if it equals to J and store it into modifiedPlaintext
116. notJ:
117. inc ESI                ; do nothing if it isn't J
118. loop changeJtoI
119. mov ESI,[EBP-16]      ; ciphertext starting address
120. mov ECX,[EBP-28]      ; load old ECX from stack
121. shr ECX,1             ; do 2 bytes at a time
122. encrypt:
123. mov DL,5              ; each playfair key row has five elements
124. inc ESI
125. call findRowCol
    ; do second byte first and store quotient and remainder in BL and BH
126. mov BX,AX
127. dec ESI
128. call findRowCol
    ; than do first byte and store quotient and remainder in AL and AH
129. cmp AL,BL
    ; if AL = BL(same Quotient), indicate that they are at same row
130. jne notSameRow
131. mov DL,AH
132. mov DH,BH
133. call addColumnOrRow
134. mov AH,DL
135. mov BH,DH
136. jmp finishEncrypt
137. notSameRow:
138. cmp AH,BH
    ; if AH = BH(same Remainder), indicate that they are at same column
139. jne differentRowAndCol

```



```

140. mov DL,AL
141. mov DH,BL
142. call addColumnOrRow
143. mov AL,DL
144. mov BL,DH
145. jmp finishEncrypt
146. differentRowAndCol:
147. xchg AH,BH ; if they are not above, exchange them
148. finishEncrypt:
149. push EAX
150. inc ESI
151. call findPlayfairKey
152. mov EBX,[EBP-32]
153. dec ESI
154. call findPlayfairKey
155. mov AL,[ESI]
156. call WriteChar
157. mov AL,[ESI+1]
158. call WriteChar
159. add ESI,2
160. mov AL,' '
161. call WriteChar
162. pop EAX
163. loop encrypt
164. pop ECX
165. ret
166. PlayFair ENDP
167.
168. findRowCol PROC
169. mov AL,[ESI]
170. mov EDI,[EBP-20] ; PlayfairKey starting address
171. push ECX
172. mov ECX,25
173. repne scasb ; compare to playfair key
174. pop ECX
175. sub EDI,[EBP-20]
176. dec EDI ; the offset of it in playfair key
177. mov EAX,EDI

```

```
178. idiv DL; division to know the row and column of it in playfair key
179. ret
180. findRowCol ENDP
181.
182. addColumnOrRow PROC
183. inc DL
184. cmp DL,5
185. jne nextRowOrColumn
186. mov DL,0
187. nextRowOrColumn:
188. inc DH
189. cmp DH,5
190. jne finishRowOrColumn
191. mov DH,0
192. finishRowOrColumn:
193. ret
194. addColumnOrRow ENDP
195.
196. findPlayfairKey PROC
197. mov EDI,[EBP-20]
198. mov AL,BL
199. mov DL,5
200. mul DL
201. add AL,BH
202. add EDI,EAX
203. mov AL,[EDI]
204. mov [ESI],AL
205. ret
206. findPlayfairKey ENDP
207. END main
```