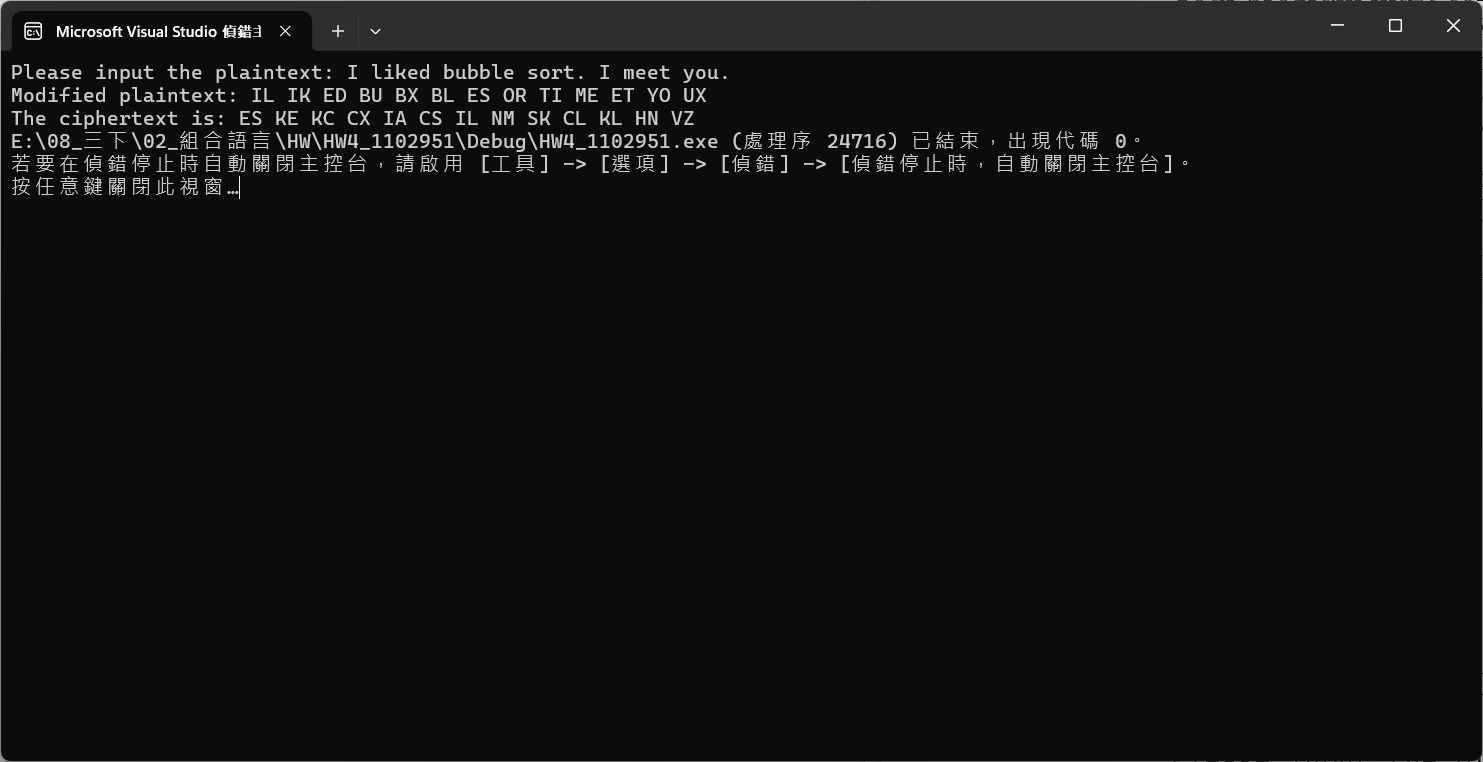
Assembly Language and Laboratory Homework 4

學號：1102951 姓名：劉宗翰

1. 程式執行畫面：



1. 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。

* 1. 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：
     + 1. 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。
       2. 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。
       3. 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）。
       4. 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）。
       5. findPlayfairKey：吃BX參數，並將結果存於AL，利用Row\*5+Column來得知該位置在PlayfairKey中的元素為何，並且將其存於AL中。

1. 程式碼：
2. INCLUDE Irvine32.inc
3. BUFMAX = 128
4. .data
5. prompt1 BYTE "Please input the plaintext: ",0
6. prompt2 BYTE "Modified plaintext: ",0
7. prompt3 BYTE "The ciphertext is: ",0
8. plaintext BYTE BUFMAX+1 DUP(0)
9. ciphertext BYTE BUFMAX+1 DUP(0)
10. 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'
11. .code
12. main PROC
13. call inputPlaintext
14. call checkCharacter
15. call addX
16. lea ESI,[ciphertext]
17. lea EDI,[PlayfairKey]
18. push ESI
19. push EDI
20. call PlayFair
21. exit
22. main ENDP
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. checkCharacter PROC
33. mov EDX,OFFSET prompt2 ; print "Modified plaintext: "
34. call WriteString
35. mov ESI,OFFSET plaintext
36. mov EDI,OFFSET ciphertext
37. charCheck:
38. mov AL,[ESI]
39. cmp AL,41h ; skip if the character is below then 'A'
40. jb skip
41. cmp AL,5bh ; copy if the character is between 'A' to 'Z'
42. jb copy
43. cmp AL,61h ; skip if the character is between '[' to '`'
44. jb skip
45. cmp AL,7bh

; Change to Uppercase if the character is between 'a' to 'z'

1. jb toUpperCase
2. cmp AL,7fh ; skip if the character is between '{' to '~'
3. jb skip
4. toUpperCase:
5. sub AL,20h
6. copy:
7. mov [EDI],AL
8. inc EDI
9. skip:
10. inc ESI
11. loop charCheck
12. invoke str\_length,ADDR ciphertext
13. mov ECX,EAX
14. invoke str\_copy,ADDR ciphertext,ADDR plaintext
15. ret
16. checkCharacter ENDP
17. addX PROC
18. mov ESI,OFFSET plaintext
19. mov EDI,OFFSET ciphertext
20. twinCheck:
21. mov AL,[ESI]
22. mov AH,[ESI+1]
23. mov [EDI],AL
24. call WriteChar
25. cmp AL,0 ; if the AL = 0, then finish
26. je theAddXEnd
27. cmp AH,0 ; if the AH = 0, then needs to add 'X' into ciphertext
28. je addLastX
29. cmp AL,AH; if the AL != AH, then write it into ciphertext
30. jne notTwin
31. mov AL,'X'
32. mov [EDI+1],AL
33. call WriteChar
34. jmp finishTwinCheck
35. notTwin:
36. mov [EDI+1],AH
37. mov AL,AH
38. call WriteChar
39. inc ESI
40. finishTwinCheck:
41. mov AL,' '
42. call WriteChar
43. inc ESI
44. add EDI,2
45. loop twinCheck
46. addLastX:
47. mov AL,'X'
48. mov [EDI+1],AL
49. call WriteChar
50. theAddXEnd:
51. invoke str\_length,ADDR ciphertext
52. mov ECX,EAX
53. invoke str\_copy,ADDR ciphertext,ADDR plaintext
54. call Crlf
55. ret
56. addX ENDP
57. PlayFair PROC
58. mov EDX,OFFSET prompt3
59. call WriteString
60. mov ESI,[EBP-16] ; ciphertext starting address
61. push ECX
62. changeJToI:

; changing J to I in the ciphertext, using loop due to there may have more then one 'J' in the string

1. mov AL,[ESI]
2. cmp AL,'J'
3. jne notJ
4. dec AL
5. mov [ESI],AL

; changing J to I if it equals to J and store it into modifiedPlaintext

1. notJ:
2. inc ESI ; do nothing if it isn't J
3. loop changeJtoI
4. mov ESI,[EBP-16] ; ciphertext starting address
5. mov ECX,[EBP-28] ; load old ECX from stack
6. shr ECX,1 ; do 2 bytes at a time
7. encrypt:
8. mov DL,5 ; each playfair key row has five elements
9. inc ESI
10. call findRowCol

; do second byte first and store quotient and remainder in BL and BH

1. mov BX,AX
2. dec ESI
3. call findRowCol

; than do first byte and store quotient and remainder in AL and AH

1. cmp AL,BL

; if AL = BL(same Quotient), indicate that they are at same row

1. jne notSameRow
2. mov DL,AH
3. mov DH,BH
4. call addColumnOrRow
5. mov AH,DL
6. mov BH,DH
7. jmp finishEncrypt
8. notSameRow:
9. cmp AH,BH

; if AH = BH(same Remainder), indicate that they are at same column

1. jne differentRowAndCol
2. mov DL,AL
3. mov DH,BL
4. call addColumnOrRow
5. mov AL,DL
6. mov BL,DH
7. jmp finishEncrypt
8. differentRowAndCol:
9. xchg AH,BH ; if they are not above, exchange them
10. finishEncrypt:
11. push EAX
12. inc ESI
13. call findPlayfairKey
14. mov EBX,[EBP-32]
15. dec ESI
16. call findPlayfairKey
17. mov AL,[ESI]
18. call WriteChar
19. mov AL,[ESI+1]
20. call WriteChar
21. add ESI,2
22. mov AL,' '
23. call WriteChar
24. pop EAX
25. loop encrypt
26. pop ECX
27. ret
28. PlayFair ENDP
29. findRowCol PROC
30. mov AL,[ESI]
31. mov EDI,[EBP-20] ; PlayfairKey starting address
32. push ECX
33. mov ECX,25
34. repne scasb ; compare to playfair key
35. pop ECX
36. sub EDI,[EBP-20]
37. dec EDI ; the offset of it in playfair key
38. mov EAX,EDI
39. idiv DL; division to know the row and column of it in playfair key
40. ret
41. findRowCol ENDP
42. addColumnOrRow PROC
43. inc DL
44. cmp DL,5
45. jne nextRowOrColumn
46. mov DL,0
47. nextRowOrColumn:
48. inc DH
49. cmp DH,5
50. jne finishRowOrColumn
51. mov DH,0
52. finishRowOrColumn:
53. ret
54. addColumnOrRow ENDP
55. findPlayfairKey PROC
56. mov EDI,[EBP-20]
57. mov AL,BL
58. mov DL,5
59. mul DL
60. add AL,BH
61. add EDI,EAX
62. mov AL,[EDI]
63. mov [ESI],AL
64. ret
65. findPlayfairKey ENDP
66. END main