Assembly Language Homework 3

學號：1102951 姓名：劉宗翰

1. 程式運行截圖：



1. 設計概念：Morse Code的陣列依ASCII Code的順序由小到大排列，並且由”Space”（ASCII Code = 20h）開始，至”\_”（ASCII Code = 5h）結束，在陣列中，若有Morse Code無法表示的字元，則以”@”儲存於陣列中，若有特殊要求，如Space，則以”/”儲存於陣列中，且陣列都以DWORD形式儲存，這是為了後續調用此陣列的方便性，但是，以此方式儲存，需要將Morse Code顛倒，如A原本的Morse Code為'.-'，在陣列中就需以'-.'的樣式儲存，最後，因為Output Morse Code字元間需要用Space做區隔，因此需要調用到WriteChar，而WriteChar需調用到AL來印出指定字元，但題目中要求Input String的OFFSET需儲存於EAX，這導致了EAX有兩個用途卻存在於同個Procedure中，因此，我在陣列中將所有元素都添加了” ”來取代WriteChar需調用到AL的問題。而作業中要求製作兩個Procedure分別為lowerToCap和MorseTran，在lowerToCap中，判定Input String的個別字元是否大於5Fh，若大於5Fh，則減20h對應到大寫的ASCII Code或是錯誤”@”，這是因為40h~5Fh和60h~7Fh在ASCII Code轉為Morse Code中的陣列結構是一樣的，因此以大於5Fh減20h作為判定標準；而在MorseTran中，先將lowerToCap利用ESI已儲存的Input String起始位置恢復到EAX，再進行運算，運算方式為，因為Morse Code陣列中以ASCII Code = 20h作為起始位置，所以先將Input String陣列中的數值減去20h以對應到Morse Code陣列的個別元素，又因陣列以DWORD形式儲存，每個Morse Code間相隔三個DWORD，所以需乘上12，最後再加上Morse Code陣列的起始位址來對應到該字元在Morse Code陣列中的正確位址，並且印出，再進行迴圈，最後跳出迴圈，並回到Main Procedure詢問是否再次輸入。
2. 程式碼：
3. INCLUDE Irvine32.inc
4. BUFMAX = 128
5. .data
6. introText BYTE "This is a Morse Code Translator!", 0
7. enterText BYTE "Please enter your string:", 0
8. outputText BYTE "Morse Code:", 0
9. askStr BYTE "Would you like to proceed another translation (y/n)? ", 0
10. inputStr BYTE BUFMAX+1 DUP(0)
11. morseCode DWORD ' /', null, 0, ' @', null, 0, '..-.', ' .-', 0, ' @', null, 0, '-...', ' -..', 0, ' @', null, 0, ' @', null, 0, '---.', ' .-', 0, '--.-', ' -.', 0, '--.-', ' -.', 0, ' @', null, 0, ' @', null, 0, '..--', ' --', 0 ; SP ~ ,
12. DWORD '...-', ' -.', 0, '-.-.', ' -.', 0, '-..-', ' .', 0, '----', ' -', 0, '---.', ' -', 0, '--..', ' -', 0, '-...', ' -', 0, '....', ' -', 0, '....', ' .', 0, '...-', ' .', 0, '..--', ' .', 0, '.---', ' .', 0, '----', ' .', 0 ; - ~ 9
13. DWORD '.---', ' ..', 0, '.-.-', ' .-', 0, ' @', null, 0, ' @', null, 0, ' @', null, 0, '--..', ' ..', 0, ' @', null, 0 ; : ~ @
14. DWORD ' -.', null, 0, '...-', ' ', 0, '.-.-', ' ', 0, ' ..-', null, 0, ' .', null, 0, '.-..', ' ', 0, ' .--', null, 0, '....', ' ', 0, ' ..', null, 0, '---.', ' ', 0, ' -.-', null, 0, '..-.', ' ', 0, ' --', null, 0 ; A ~ M
15. DWORD ' .-', null, 0, ' ---' , null, 0, '.--.', ' ', 0, '-.--', ' ', 0, ' .-.', null, 0, ' ...', null, 0, ' -', null, 0, ' -..', null, 0, '-...', ' ', 0, ' --.', null, 0, '-..-', ' ', 0, '--.-', ' ', 0, '..--', ' ', 0 ; N ~ Z
16. DWORD ' @', null, 0, ' @', null, 0, ' @', null, 0, ' @', null, 0, ' @', null, 0 ; [~\_
17. .code
18. main PROC
19. mov EDX,OFFSET introText ; print "This is a Morse Code Translator!"
20. call WriteString
21. keepGoing:
22. call Crlf
23. mov EDX,OFFSET enterText ; print "Please enter your string:"
24. call WriteString
25. call Crlf
26. mov EDX,OFFSET inputStr ; input string
27. mov ECX,BUFMAX
28. call ReadString
29. mov ECX,EAX ; store string size
30. call lowerToCap ; change lowercase to uppercase
31. mov EDX,OFFSET outputText ; print "Morse Code:"
32. call WriteString
33. call Crlf
34. call MorseTran ; change string to morse code
35. call Crlf
36. mov EDX,OFFSET askStr ; print "Would you like to proceed another translation (y/n)? "
37. call WriteString
38. call ReadChar
39. cmp AL,79h
40. je keepGoing
41. exit
42. main ENDP
43. lowerToCap PROC
44. push ECX
45. mov EAX,OFFSET inputStr
46. mov ESI,EAX ; store the starting position of input string to ESI
47. lowercaseCheck:
48. mov EBX,0 ; clean up EBX
49. mov BL,[EAX]
50. cmp BL,60h
51. jb noChange
52. sub BL,20h ; lowercase change to uppercase
53. noChange:
54. mov [EAX],BL
55. inc EAX
56. loop lowercaseCheck
57. pop ECX
58. ret
59. lowerToCap ENDP
60. MorseTran PROC
61. push ECX
62. mov EAX,ESI ; restore the starting position of input string to EAX
63. TransferToMorse:
64. mov EBX,0
65. mov BL,[EAX]
66. sub BL,20h
67. imul BX,12
68. mov EDX,OFFSET morseCode
69. mov ESI,EDX
70. add ESI,EBX
71. mov EDX,ESI
72. call WriteString
73. inc EAX
74. loop TransferToMorse
75. pop ECX
76. ret
77. MorseTran ENDP
78. END main