CS-35101

Thomas Moore

5/1/2022

Summary:

The first section of the code I modified and created was a loop for detecting and reversing the case of a character. After that I jump and link to the findMin section of code. There I use addition and several comparisons to attempt to find the minimum ASCIII character.

Conclusion:

This lab was rather challenging not just coding wise but comprehension wise too. This is the first time I had to complete a coding project in MIPS where the code was mostly completed but I had to create my own code to connect it all together. I was very confused about the minimum ASCII character section, but I think my code will work for the requested outputs. The only other problem I could not fix was an asterisk printing at the end of my reversed string.

Lab 5 Code:

1. # Starter code for reversing the case of a 30 character input.
2. # Put in comments your name and date please. You will be
3. # revising this code.
4. #
5. # Created by Dianne Foreback
6. # Students should modify this code
7. #
8. # This code displays the authors name (you must change
9. # outpAuth to display YourFirstName and YourLastName".
10. #
11. # The code then prompts the user for input
12. # stores the user input into memory "varStr"
13. # then displays the users input that is stored in"varStr"
14. #
15. # You will need to write code per the specs for
16. # procedures main, revCase and function findMin.
17. #
18. # revCase will to reverse the case of the characters
19. # in varStr. You must use a loop to do this. Another buffer
20. # varStrRev is created to hold the reversed case string.
21. #
22. # Please refer to the specs for this project, this is just starter code.
23. #
24. # In MARS, make certain in "Settings" to check
25. # "popup dialog for input syscalls 5,6,7,8,12"
26. #
27. .data # data segment
28. .align 2 # align the next string on a word boundary
29. outpAuth: .asciiz "This is Thomas Moore presenting revCaseMin.\n"
30. outpPrompt: .asciiz "Please enter 30 characters (upper/lower case mixed):\n"
31. .align 2 #align next prompt on a word boundary
32. outpStr: .asciiz "You entered the string: "
33. .align 2 # align users input on a word boundary
34. outpStrRev: .asciiz "\nYour string in reverse case is: "
35. .align 2 # align the output on word boundary
36. varStrRev: .space 32 # reserve 32 characters for the reverse case string
37. .align 2 # align on a word boundary
38. outpStrMin: .asciiz "\nThe min ASCII character after reversal is: "
39. varStr: .space 32 # will hold the user's input string thestring of 20 bytes
40. # last two chars are \n\0 (a new line and null char)
41. # If user enters 31 characters then clicks "enter" or hits the
42. # enter key, the \n will not be inserted into the 21st element
43. # (the actual users character is placed in 31st element). the
44. # 32nd element will hold the \0 character.
45. # .byte 32 will also work instead of .space 32
46. .align 2 # align next prompt on word boundary
47. myChar: .byte 'a'
48. #
49. .text # code section begins
50. .globl main
51. main:
52. #
53. # system call to display the author of this code
54. #
55. la $a0,outpAuth # system call 4 for print string needs address of string in $a0
56. li $v0,4 # system call 4 for print string needs 4 in $v0
57. syscall
58. #
59. # system call to prompt user for input
60. #
61. la $a0,outpPrompt # system call 4 for print string needs address of string in $a0
62. li $v0,4 # system call 4 for print string needs 4 in $v0
63. syscall
64. #
65. # system call to store user input into string thestring
66. #
67. li $v0,8 # system call 8 for read string needs its call number 8 in $v0
68. # get return values
69. la $a0,varStr # put the address of thestring buffer in $t0
70. li $a1,32 # maximum length of string to load, null char always at end
71. # but note, the \n is also included providing total len < 22
72. syscall
73. #move $t0,$v0 # save string to address in $t0; i.e. into "thestring"
74. #
75. # system call to display "You entered the string: "
76. #
77. la $a0,outpStr # system call 4 for print string needs address of string in $a0
78. li $v0,4 # system call 4 for print string needs 4 in $v0
79. syscall
80. #
81. # system call to display user input that is saved in "varStr" buffer
82. #
83. la $a0,varStr # system call 4 for print string needs address of string in $a0
84. li $v0,4 # system call 4 for print string needs 4 in $v0
85. syscall
86. #
87. # Your code to invoke revCase goes next
88. #
89. jal revCase #invoke revCase
90. # Exit gracefully from main()
91. li $v0, 10 # system call for exit
92. syscall # close file

95. ################################################################
96. # revCase() procedure can go next
97. ################################################################
98. # Write code to reverse the case of the string. The base address of the
99. # string should be in $a0 and placed there by main(). main() should also place into
100. # $a1 the number of characters in the string.
101. # You will want to have a label that main() will use in its jal
102. # instruction to invoke revCase, perhaps revCase:
103. #
104. revCase:
105. addi $t0, $zero,0 #loop var
106. la $t7, varStr
107. while:
108. bgt $t0, $a1, exit #loop while var is less than 31
109. lb $t2, 0($t7)
110. blt $t2, 96, upper #checks to see if lowercase branch if not
111. blt $t2, 65, skip
112. addi $t2, $t2, -32 #subtracts 32 from lowercase char
113. sb $t2, 0($t7)
114. addi $t0, $t0, 1 #increment loop
115. addi $t7, $t7, 1 # add 4 to index
116. j while
117. upper:
118. addi $t2, $t2, 32 #adds 32 to uppercase char
119. sb $t2, 0($t7)
120. addi $t0, $t0, 1 #increment loop var
121. addi $t7, $t7, 1 # add 4 to index
122. j while
123. skip:
124. addi $t0, $t0, 1 #increment loop var
125. addi $t7, $t7, 1 # add 4 to index
126. j while
128. exit:
129. move $t5, $ra
131. #
132. # After reversing the string, you may print it with the following code.
133. # This is the system call to display "Your string in reverse case is: "
134. la $a0,outpStrRev # system call 4 for print string needs address of string in $a0
135. li $v0,4 # system call 4 for print string needs 4 in $v0
136. syscall
137. # system call to display the user input that is in reverse case saved in the varRevStr buffer
138. la $a0,varStr # system call 4 for print string needs address of string in $a0
139. li $v0,4 # system call 4 for print string needs 4 in $v0
140. syscall
141. #
142. # Your code to invoke findMin() can go next
143. jal findMin
144. # Your code to return to the caller main() can go next
145. move $ra, $t5
146. jr $ra
147. ################################################################
148. # findMin() function can go next
149. ################################################################
150. # Write code to find the minimum character in the string. The base address of the
151. # string should be in $a0 and placed there by revCase. revCase() should also place into
152. # $a1 the number of characters in the string.
153. # You will want to have a label that revCase() will use in its jal
154. # instruction to invoke revCase, perhaps findMin:
155. #
156. #
157. findMin:
158. addi $t0, $zero,0 #loop var declared
159. la $t6, varStr #load the desired index
160. lb $t3, 0($t6)
162. while2:
163. bgt $t0, $a1, exit2 #loop while var is less than 31
164. lb $t2 0($t6) #load the desired index
165. blt $t2, 65 ,skip2
166. bgt $t3, $t2, minChar
167. addi $t0, $t0, 1 #increment loop var
168. addi $t6, $t6, 1 # add 4 to index
169. j while2
170. minChar:
171. move $t3, $t2
172. addi $t0, $t0, 1 #increment loop var
173. addi $t6, $t6, 1 # add 1 to index
174. j while2
175. #jr $ra
176. skip2:
177. addi $t0, $t0, 1 #increment loop var
178. addi $t7, $t7, 1 # add 4 to index
179. j while2
180. exit2:
181. sb $t3, myChar
182. # write use a loop and find the minimum character
184. #
185. # system call to display "The min ASCII character after reversal is: "
186. la $a0,outpStrMin # system call 4 for print string needs address of string in $a0
187. li $v0,4 # system call 4 for print string needs 4 in $v0
188. syscall
189. # write code for the system call to print the the minimum character
190. li $v0, 4
191. la $a0, myChar
192. syscall
193. # write code to return to the caller revCase() can go next
194. jr $ra

Example Output:

Text

Description automatically generated