CS-35101  
Thomas Moore  
4/16/22

Summary:  
For lab 4 I used strings that correspond to each singular digit. These strings are used from lines 51 to 108 as each digit has its one branch that can print the names of the digits. I use several loops within the code, the second one on line 32 uses division to break down the numbers longer than one digit into individual numbers. While the loop on line 45 is used to detect negative numbers and return “Invalid Entry”. If number is longer than a single digit it will store the digits in correct order by using the stack.

Conclusion:

This lab challenged me by turning a basic program in most languages into a 100+ line program that required several forms of register storage and arithmetic. I was having a problem storing some of the data until I realized that I could store it on the stack using $sp. I discovered that $sp works on first in last out principles so I could easily access my data in an organized fashion. The second lesson I learned was the differences between mflo and mfhi. Using div stores either the quotient in the LO register and the remainder in the HI register.

Lab 4 Code:

1. #Thomas Moore
2. .data
3. zero: .asciiz "Zero "
4. one: .asciiz "One "
5. two: .asciiz "Two "
6. three: .asciiz "Three "
7. four: .asciiz "Four "
8. five: .asciiz "Five "
9. six: .asciiz "Six "
10. seven: .asciiz "Seven "
11. eight: .asciiz "Eight "
12. nine: .asciiz "Nine "
13. intro: .asciiz "Enter a positive number: "
14. error: .asciiz "Invaild Entry"
16. .text
17. la $a0, intro
18. li $v0, 4 #print string
19. syscall
20. li $v0, 5 #read in int
21. syscall
22. move $t0, $v0 #move input to $t0
23. blt $t0, $0,invalid #if negative branch to invalid
24. li $t1, 10 #storing int 10
25. li $v0, 4
26. li $t4, -1 #storing int -1
27. loop1:
28. bne $t0, $0, digit
29. beq $t2, $t0, check
31. digit:
32. div $t0, $t1
33. mflo $t3
34. mfhi $t2
35. move $t0, $t3
36. addi $t4, $t4, 1 #iterations
37. addi $sp, $sp, 4
38. sw $t2, 0($sp)
39. j loop1
40. check:
41. lw $t2, 0($sp)
43. loop2:
44. beq $t4, $0, terminate
45. addi $sp, $sp, -4
46. lw $t2, 0($sp)
47. addi $t4, $t4, -1
48. pone:
49. bne $t2, 1, ptwo
50. la $a0, one
51. syscall
52. j loop2
54. ptwo:
55. bne $t2, 2, pthree
56. la $a0, two
57. syscall
58. j loop2
60. pthree:
61. bne $t2, 3, pfour
62. la $a0, three
63. syscall
64. j loop2
66. pfour:
67. bne $t2, 4, pfive
68. la $a0, four
69. syscall
70. j loop2
72. pfive:
73. bne $t2, 5, psix
74. la $a0, five
75. syscall
76. j loop2
78. psix:
79. bne $t2, 6, pseven
80. la $a0, six
81. syscall
82. j loop2
84. pseven:
85. bne $t2, 7, peight
86. la $a0, seven
87. syscall
88. j loop2
90. peight:
91. bne $t2, 8, pnine
92. la $a0, eight
93. syscall
94. j loop2
96. pnine:
97. bne $t2, 9, pzero
98. la $a0, nine
99. syscall
100. j loop2
101. pzero:
102. la $a0, zero
103. syscall
104. j loop2
105. invalid:
106. la $a0, error
107. li $v0, 4
108. syscall
110. terminate:
111. li $v0, 10 #standard termination
112. syscall

Example Output:

Text

Description automatically generated