CS-35101

Thomas Moore

3/25/22

Summary:

For lab 3 I used a loop from lines 11 to 20. This loop continually prints the input string on line 4 and reads in a number on line 15. The key portion of the code that breaks the loop is on line 17 where it checks if $v0 is equal to zero. If this is statement is true, then it will branch to line 22 and start the termination process where it prints a string and the sum of all the entered integers.

Conclusion:

The main problem I faced in lab 3-1 was figuring out how to store the integer properly after it was read. I originally tried using $a0 to store it via add $a0, $a0, $v0 but I don’t think I initialized a value to $a0 correctly. Every time I tried to use it as the register I would get “268501230” instead of the actual value. This instead led me to use $s0 on line 7 where I stored a base value of 0. The lesson I learned from this is that though it is possible to use $a0 to store what I wanted I needed to format the code differently and use $a0 as a properly as an argument. Instead I adapted and used what I knew about $s registers and stored the values as non-temporary data.

Lab 3-1 Code:

1. #Thomas Moore
2. .data
3. Intro: .asciiz "\nEnter 0 to exit and print sum: "
4. Input: .asciiz "\nEnter an integer: "
5. Sum: .asciiz "\nThe Sum is: "
6. .text
7. li $s0, 0 # $s0 = 0
8. li $v0, 4 # Print string
9. la $a0, Intro
10. syscall
11. Loop:
12. li $v0, 4 # Print string
13. la $a0, Input
14. syscall
15. li $v0, 5 # Read integer
16. syscall
17. beq $v0, $0, Terminate #Terminate if ($v0 == 0)
18. add $s0, $s0, $v0 # add integer to current sum
20. j Loop
21. Terminate:
22. li $v0, 4 # Print string
23. la $a0, Sum
24. syscall
26. li $v0, 1 # Print Int sum
27. move $a0, $s0
28. syscall
29. li $v0, 10
30. syscall

lab 3-1 Results:

Text

Description automatically generated