CS 2160 Fall 2017 Al Glock

Program 1

Write a subroutine to add the numbers from 1 to 100. The limit number (100 in this case) is in the $a0 register. When your assignment is graded, this number may be changed, so your code should work regardless of what the limit number is.

Copy the boilerplate code in the box below into MARS and save it as <lastname>.asm (e.g., “glock.asm”) Please do not use .txt or nothing as an extent as these will get lost when I download the code from Blackboard. Do not submit your code as a word document or a pdf. You will be graded as late if you do.

Add your code at the very bottom of the code below. Do not change any of the code above that point in your submission. You may, when you test your code, substitute different numbers for the “100” in line 3, but reset that number to 100 when you submit the assignment. Altering the boilerplate code will result in a 0 on the assignment because it potentially changes the problem you’re trying to solve.

The boilerplate code is commented, so you are encouraged to study it to see how it works. This may help you when you add in your code. Basically, the number 100 is loaded into the $a0 register and a separate routine is called to compute the sum of the numbers. Its result is saved in register $s0. Notice that this routine does not alter the calling argument in $a0. Your summation routine is called next and its result is compared with the previously stored result. If the two results compare, you get the success message. If not, you don’t.

Correct output looks like this:

Congratulations, your code appears to work!

-- program is finished running --

You will lose points on this assignment if you 1) submit code you didn’t write yourself, 2) alter the boilerplate code in the file you turn in, 3) name the file something besides <last name>.asm, 4) don’t get the right answer, 5) alter the values in $a0 or $s0, 6) do not return properly from your subroutine, 7) have side effects in your code (i.e., dependencies on other code apart from $a and $v register values), and 8) can’t explain your code if I ask you about it.

.data

Limit: #sum from 1 to this number

.word 100

Success: #Happy message string

.asciiz "Congratulations, your code appears to work!\r\n"

Nonsuccess: #Other message string

.asciiz "You still have work to do...\r\n"

.text #running program starts here

lw $a0, Limit #provide the argument to the subroutine call

jal GaussianSum #calculate the sum based on Gauss' formula: n(n+1)/2

addi $s0, $v0, 0 #save the result for later

jal Sum #Execute student's code

beq $v0, $s0, Good #compare values, go to Good if the same

la $a0, Nonsuccess #load the address of the unsuccessful message

j AlmostDone #jump to place where it's printed out

Good:

la $a0, Success #load the success message

AlmostDone:

addi $v0, $0, 4 #tell the OS to print the string whose address is in $a0

syscall #kernel call

addi $v0, $0, 10 #tell the OS we want to end the program

syscall #call the OS

GaussianSum:

addi $v0, $a0, 1 #add 1 to the calling argument and load into return register

mul $v0, $a0, $v0 #multiply the n+1 in $v0 times the n in $a0

srl $v0, $v0, 1 #divide $v0 by 2 by shifting right 1 bit

jr $ra

Sum:

#your code goes below here. Do not change anything above this point when submitting.