CS 2160 Fall 2017 Al Glock

Program 2

Write a program which uses two different methods to reverse a string. The results of both methods should compare, showing that the implementation is correct.

The StackReverse method has two calling parameters. $a0 is the address of the string containing the original, unreversed text. $a1 contains the address of a 40-character buffer where you can write the reversed string. There is no output value for this method, i.e., nothing in the $v registers. To do this task, we read through the input string, character by character, pushing each onto the stack. Even though we are reading the input string by bytes, we are storing words onto the stack . We could store bytes, but this would lead us to have to realign the stack pointer to a word boundary when we are done. This can be complicated. It’s easier to store each byte of the input string as a word on the stack. Once the characters are stored onto the stack, they are retrieved one at a time from the top of the stack. You will be reading words from the stack, but storing the low-order byte each time into the reversed-text buffer.

For the InPlaceReverse method, $a0 points to the string of text to be reversed. There is no output buffer and no output value from the method. This method operates by locating the end of the string (last character before the terminating 0) and then successively swapping characters at each end of the string, incrementing the beginning-of -string and decrementing the end-of-string pointers each time. This method exits when the pointers are equal or cross each other.

The boilerplate code then compares the results of the two reversal methods, presuming that if they match, you did the work correctly.

If your program works correctly, you will receive the following output:

Good job, looks like you're done!

-- program is finished running --

Add your code at the very bottom of the boiler plate code below. Do not change any of the code above that point in your submission. Altering the boilerplate code will result in a 0 on the assignment because it potentially changes the problem you’re trying to solve.

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 #the few items we need for this assignment

String1:

.asciiz "Here is a string to be reversed."

Success:

.asciiz "Good job, looks like you're done!\r\n"

Nonsuccess:

.asciiz "Oops. Still need to work on this.\r\n"

CR:

.asciiz "\r\n"

Buffer:

.space 40 #This buffer is where the reversed text is written for the stack reverse

.text

la $a0, String1 #load address of the string to be reversed

la $a1, Buffer #load the address where the string will be written for the stach reverse

jal StackReverse #do the stack reverse, leaving the calling arguments unchanged

jal InPlaceReverse #do the in-place reverse, writing the reversed string where the original string was

jal strcmp #compare the results of the two reverses

beq $v0, $0, OK #if they match, go to print the success message

la $a0, String1 #if they don't match, print out both results and the non-success messge

addi $v0, $0, 4

syscall

la $a0, CR

addi $v0, $0, 4

syscall

la $a0, Buffer

addi $v0, $0, 4

syscall

la $a0, CR

addi $v0, $0, 4

syscall

la $a0, Nonsuccess

j print

OK: #load the success message for printing

la $a0, Success

print: #print success or non-success message

addi $v0, $0, 4

syscall

addi $v0, $0, 10 #exit gracefully

syscall

strcmp: #String compare meethod

#$a0 and $a1 contain addresses of strings to be compared

#$v0 contains result, 0 if identical, non-zero if different

addi $t0, $a0, 0 #copy $a0 and $a1 so the copied can be changed

addi $t1, $a1, 0

strcmploop:

lbu $t2, ($t0) #load a character from each string

lbu $t3, ($t1)

sub $v0, $t2, $t3 #subtract to compare

beq $t2, $0, strcmpend #terminate compare if current haracter

beq $t3, $0, strcmpend # is zero in either string

addi $t0, $t0, 1 #increment address pointers

addi $t1, $t1, 1

beq $v0, $0, strcmploop #continue the compare if characcters were the same

strcmpend:

jr $ra #return with result in $v0

StackReverse: #this reverses the string by pusing characters onto the stack and popping them back off

#your code goes below here

#remember to also include the InPlaceReverse subroutine