Evan Smith

CS231

3/7/18

1)

000100 01001 01010 0000000000000100

100011 01000 10001 0000000000000000

000000 10001 10000 10000 00000 100000

001000 01001 01001 0000000000000001

000010 00000010001100101100100101

2)

#Program Description: Assignment 1 calls for a program that asks the user to enter a value for array

# size 'n' and fills up the array with 'n' integers. The array is then reversed

# within itself and printed to the screen.

#Author: Evan Smith

#Creation Date: 03/03/2018

.data

arr: .word 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

str1: .asciiz "\nPlease select a value for 'n': "

str2: .asciiz "\nPlease enter values for array size 'n': \n"

str3: .asciiz "\nThe array of 'n' elements: \n\n"

str4: .asciiz " | "

str5: .asciiz "\n"

str6: .asciiz "\n\n\*\*\*\*\*\*\*\*\*\*\*\* REVERSING ARRAY \*\*\*\*\*\*\*\*\*\*\*\*\n\n"

errLo: .asciiz "\nERROR: You must choose a value greater than 0"

errHi: .asciiz "\nERROR: You must choose a value less than 20 "

err2: .asciiz "\nERROR: You must enter a positive number "

err3: .asciiz "\nERROR: You must enter a number divisible by 3, try again \n"

.text

main:

begin:

#call readNum function and save return

# arraysize = readNum()

# num 'n' saved to s3

jal readNum

add $s3, $t9, $0

#call createArray function

# OK = verifySize(n)

add $a0, $s3, $0

jal verifySize

#if (OK == 1) call createArray else go to 'begin:'

add $a0, $s7, $0

beq $s7, $0, begin

add $a0, $s3, $0

jal createArray

#call printArray function

add $a0, $s3, $0

jal printArray

#call reverseArray function

add $a0, $s3, $0

jal reverseArray

#call printArray function

add $a0, $s3, $0

jal printArray

#program exit from main()

li $v0, 10

syscall

readNum:

#ask the user to enter a number for array size

li $v0, 4

la $a0, str1

syscall

#get user input

li $v0, 5

syscall

#return 'n'

add $t9, $v0, $0

jr $ra

verifySize:

#param 'n' entered in readNum

add $s3, $a0, $0

#store constants for error checking

li $t1, 0

li $t2, 20

#error check to see if amount of items is <= 20

ble $s3, $t1, err\_1

bge $s3, $t2, err\_2

#set flag (OK == 1)

li $s7, 1

#send 'n' back to main

add $v0, $s3, $0

jr $ra

err\_1:

#set flag (OK == 0)

li $s7, 0

li $v0, 4

la $a0, errLo

syscall

#return int 'OK == 0'

add $v0, $s7, 0

jr $ra

err\_2:

#set flag (OK == 0)

li $s7, 0

li $v0, 4

la $a0, errHi

syscall

#return int 'OK == 0'

add $v0, $s7, 0

jr $ra

createArray:

add $s3, $a0, $0

#loop counter and function constant

li $t0, 0

li $t9, 3

#declare array and memory register for first element

la $s4, arr

#ask the user to enter values for array size 'n'

li $v0, 4

la $a0, str2

syscall

loop1:

#run loop until we reach user input

beq $t0, $s3, exit1

#enter user input 'x'

li $v0, 5

syscall

#if x is negative throw error

blt $v0, $0, err\_3

#if x is not divisble by 3 throw error

add $t1, $v0, $0

div $t1, $t9

mfhi $t1

bne $t1, $0, err\_4

#store input into array

sw $v0, 0($s4)

#increase counter

add $s4, $s4, 4

add $t0, $t0, 1

j loop1

exit1:

#void return type

jr $ra

printArray:

#loop counter

li $t0, 0

#declare array and memory register for first element

la $s4, arr

#program announcement to print array

li $v0, 4

la $a0, str3

syscall

loop2:

#run loop until we reach user input

beq $t0, $s3, exit2

#store input into array

lw $t1, 0($s4)

#print array cells

li $v0, 4

la $a0, str4

syscall

#print x and index i

add $a0, $t1, $0

li $v0, 1

syscall

#increase counter

add $s4, $s4, 4

add $t0, $t0, 1

j loop2

exit2:

#close storage container and return to main

li $v0, 4

la $a0, str4

syscall

#print newline

li $v0, 4

la $a0, str5

syscall

#void return type

jr $ra

reverseArray:

#load the array size into $t8

add $t8, $s3, $0

# counter and offset num

li $t6, 0

li $t7, 4

#head = arr[head]

la $s4, arr

#to find the last element preform arr[(n\*4) - 1]

mult $t8, $t7

mflo $t9

#tail = n \* 4

add $t2, $t9, $s4

#subtract tail by 4 to account for offset

sub $t2, $t2, 4

#set up swap loop bounds (floor of n/2)

li $t7, 2

div $t8, $t7

mflo $t9

#program announcement to print array

li $v0, 4

la $a0, str6

syscall

swap:

#run loop until we reach floor of n/2

beq $t9, $t6, exit3

#load temps for swap

lw $t3, 0($s4)

lw $t4, 0($t2)

#preform swap

sw $t3, 0($t2)

sw $t4, 0($s4)

#increment and decrement counters

add $s4, $s4, 4

sub $t2, $t2, 4

add $t6, $t6, 1

j swap

exit3:

jr $ra

# error messages

err\_3:

li $v0, 4

la $a0, err2

syscall

j loop1

err\_4:

li $v0, 4

la $a0, err3

syscall

j loop1