Name: Antonio Santos

Lab Section: B03

Course: ENCM 369

Lab Assignment: Lab 2

Exercise C

# array-sum.asm

# ENCM 369 Winter 2019 Lab 2 Exercise C Part 3

# Start-up and clean-up code copied from stub1.asm

# BEGINNING of start-up & clean-up code. Do NOT edit this code.

.data

exit\_msg\_1:

.asciiz "\*\*\*About to exit. main returned "

exit\_msg\_2:

.asciiz ".\*\*\*\n"

main\_rv:

.word 0

.text

# adjust $sp, then call main

addi $t0, $zero, -32 # $t0 = 0xffffffe0

and $sp, $sp, $t0 # round $sp down to multiple of 32

jal main

nop

# when main is done, print its return value, then halt the program

sw $v0, main\_rv

la $a0, exit\_msg\_1

addi $v0, $zero, 4

syscall

nop

lw $a0, main\_rv

addi $v0, $zero, 1

syscall

nop

la $a0, exit\_msg\_2

addi $v0, $zero, 4

syscall

nop

addi $v0, $zero, 10

syscall

nop

# END of start-up & clean-up code.

# Global variables

.data

# int abc[ ] = {-32, -8, -4, -16, -128, -64}

.globl abc

abc: .word -32, -8, -4, -16, -128, -64

# Hint for checking that the original program works:

# The sum of the six array elements is -252, which will be represented

# as 0xffffff04 in a MIPS GPR.

# Hint for checking that your final version of the program works:

# The mimimum of the four array elements is -128, which will be represented

# as 0xffffff80 in a MIPS GPR.

# int main(void)

#

# local variable register

# int \*p $s0

# int \*end $s1

# int sum $s2

# int min $s3 (to be used when students enhance the program)

.text

.globl main

main:

la $s0, abc # p = abc

addi $s1, $s0, 24 # end = p + 6

add $s2, $zero, $zero # sum = 0

add $s3, $zero, $zero # min = 0

L1:

beq $s0, $s1, L2 # if (p == end) goto L2

lw $t9, ($s0) # $t9 = \*p

add $s2, $s2, $t9 # sum += $t9

addi $s0, $s0, 4 # p++

slt $t0, $t9, $s3 # $t0 = if($t9 < $s3)

bne $t0, $zero, L3 # if($t0 != 0) goto L3

j L1

L2:

add $v0, $zero, $zero # return value from main = 0

jr $ra

L3:

add $s3, $t9, $zero # min = $t9

j L1

Exercise D

# BEGINNING of start-up & clean-up code. Do NOT edit this code.

.data

exit\_msg\_1:

.asciiz "\*\*\*About to exit. main returned "

exit\_msg\_2:

.asciiz ".\*\*\*\n"

main\_rv:

.word 0

.text

# adjust $sp, then call main

addi $t0, $zero, -32 # $t0 = 0xffffffe0

and $sp, $sp, $t0 # round $sp down to multiple of 32

jal main

nop

# when main is done, print its return value, then halt the program

sw $v0, main\_rv

la $a0, exit\_msg\_1

addi $v0, $zero, 4

syscall

nop

lw $a0, main\_rv

addi $v0, $zero, 1

syscall

nop

la $a0, exit\_msg\_2

addi $v0, $zero, 4

syscall

nop

addi $v0, $zero, 10

syscall

nop

# END of start-up & clean-up code.

# Below is the stub for main. Edit it to give main the desired behaviour.

.data

.globl foo

foo: .word 0xd3, 0xe3, 0xf3, 0xc3, 0x83, 0x93, 0xa3, 0xb3

.globl bar

bar: .word 0x80, 0x70, 0x60, 0x50, 0x40, 0x30, 0x30, 0x10

# int main(void)

#

# local variable register

# int \*p $s0

# int \*q $s1

# int \*stop $s2

# int max $s3

# int j $s4

# int \*m $s5

# int n $s6

.text

.globl main

main:

la $s5, foo # m = foo

addi $s4, $zero, 4 # j = 1

addi $t0, $zero, 32 # $t0 = 8

add $s3, $zero, $zero # max = 0

for\_start:

slt $t1, $s4, $t0 # $t1 = if($s4 < $t0)

beq $t1, $zero, for\_end # if($t1 == 0) goto for\_end

add $t2, $s5, $s4 # $t2 = foo + j

lw $t4, ($t2) # $t4 = foo[j]

slt $t3, $s3, $t4 # $t3 = if(max < foo[j])

beq $t3, $zero, if\_end # if($t3 == 0) goto if\_end

add $s3, $zero, $t4 # max = foo[j]

if\_end:

addi $s4, $s4, 4 # j++

j for\_start # goto for\_start

for\_end:

la $s0, bar # p = bar

addi $s1, $s5, 32 # q = foo + 8

addi $s2, $s0, 32 # stop = p + 8

do\_start:

addi $s1, $s1, -4 # q--

lw $t4, ($s0) # $t4 = \*p

sw $t4, ($s1) # \*q = \*p

addi $s0, $s0, 4 # p++

beq $s0, $s2, do\_end # if(p == stop) goto do\_end

j do\_start # goto do\_start

do\_end:

add $v0, $zero, $zero # return value from main = 0

jr $ra