Oindril Dutta (CHANGE WITH YOUR OWN NAME AND CHANGE FILE ITSELF PLEASE)

CS 47

Extra Credit

***Problem #1 Code***

#======================================

#$t0 is general use number

#======================================

#$t1 is the value at the sorted barrier

#$t2 is the current value

#======================================

#$a1 is the index of the array

#$a2 is the sorted barrier of the sort

#======================================

#$s1 is length of var\_a, var\_m

#======================================

#$v1 is the index of the current min

#$s2 is the current min

#======================================

.include "cs47\_macro.asm"

.data

.align 2

var\_a: .word 8 6 11 17 20 3 2 3 5 5 20 5 6 7 8 15 8 10 11 14 15 17 1

var\_m: .word 23

done: .asciiz "Sorted!\n"

swpng: .asciiz "Swapping!\n"

sepe: .asciiz ": "

newl: .asciiz "\n"

.text

.globl start

start:

#Initizlize 0

add $a2, $zero, 0 #sort barrier index of var\_a

#Get the length

lw $s1, var\_m #length of var\_a

loopcheck:

#Check if the sorted index reached the length yet

sub $t0, $s1, $a2 #If sorted index of a < length of a...

blez $t0, bbye #Jump to second check, else...

#assign the sorted barrier as the new min

move $v1, $a2

#set the index to 0

move $a1, $a2

looproutine:

#Check to see if the index is greater than the length, and if it is swap+restart

sub $t0, $s1, $a1 #If sorted index of a < length of a...

blez $t0, swap #Jump to second check, else...

#Get the current min

la $s2, var\_a #Load var\_a array into $a2.

mul $t0, $v1, 4 #Get pointer reference by mult index by 4.

add $t0, $s2, $t0 #Go to the index from pointer in array.

lw $s2, ($t0) #Get value from index in var\_a, store into $s2 as min.

#Get the number at the current index

la $t2, var\_a #Load var\_a array into $t1.

mul $t0, $a1, 4 #Get pointer reference by mult index by 4.

add $t0, $t2, $t0 #Go to the index from pointer in array.

lw $t2, ($t0) #Get value from index in var\_a, store into $t1.

#See if the current index is smaller than the min

sub $t0, $s2, $t2

blez $t0, nomin #If the current number is bigger, then skip assigning the new min

#Set the min to the current index

move $v1, $a1

nomin:

#Add the index and keep moving

addi $a1, $a1, 1

j looproutine

swap:

#Get the current min

la $s2, var\_a #Load var\_a array into $a2.

mul $t0, $v1, 4 #Get pointer reference by mult index by 4.

add $t0, $s2, $t0 #Go to the index from pointer in array.

lw $s2, ($t0) #Get value from index in var\_a, store into $s2 as min.

#Get the val at the sorted varrier

la $t1, var\_a #Load var\_a array into $a2.

mul $t0, $a2, 4 #Get pointer reference by mult index by 4.

add $t0, $t1, $t0 #Go to the index from pointer in array.

lw $t1, ($t0) #Get value from index in var\_a, store into $s2 as min.

#Put the min at the sorted barrier

la $t2, var\_a

mul $t0, $a2, 4

add $t0, $t2, $t0

sw $s2, ($t0) #stores the value $s2 from the merge logic into ($t0)

#Put the val at the sorted barrier at the min pos

la $t2, var\_a

mul $t0, $v1, 4

add $t0, $t2, $t0

sw $t1, ($t0) #stores the value $t1 from the merge logic into ($t0)

add $a2, $a2, 1 #Add to the sorted barrier

j loopcheck

bbye:

add $a1, $zero, $zero

print\_str(done)

print:

sub $t0, $s1, $a1 #If sorted index of a < length of a...

blez $t0, exit #Jump to second check, else...

la $a2, var\_a #Load var\_a array into $a2.

mul $t0, $a1, 4 #Get pointer reference by mult index by 4.

add $t0, $a2, $t0 #Go to the index from pointer in array.

lw $a2, ($t0) #Get value from index in var\_a, store into $s2 as min.

print\_reg\_int($a1)

print\_str(sepe)

print\_reg\_int($a2)

print\_str(newl)

add $a1, $a1, 1

j print

exit:

li $v0, 10

syscall

***Problem #2 Code***

#======================================

#$t0 is general use number

#======================================

#$t1 is var\_a

#$t2 is the previous val of var\_a

#======================================

#$a1 is the sorted barrier of the sort

#======================================

#$s1 is length of var\_a, var\_m

#======================================

#$v1 is the temp value of var\_a

#======================================

.include "cs47\_macro.asm"

.data

.align 2

var\_a: .word 8 6 11 17 20 3 2 3 5 5 20 5 6 7 8 15 8 10 11 14 15 17 1

var\_m: .word 23

done: .asciiz "Sorted!\n"

sepe: .asciiz ": "

newl: .asciiz "\n"

.text

.globl start

start:

add $a1, $zero, 1 #sort barrier index of var\_a

lw $s1, var\_m #length of var\_a

loopcheck:

blez $a1, begofarray

sub $t0, $s1, $a1

blez $t0, bbye

la $t1, var\_a #Load var\_a array into $t1.

mul $t0, $a1, 4 #Get pointer reference by mult index $a1 by 4.

add $t0, $t1, $t0 #Go to the index from pointer in array.

lw $t1, ($t0) #Get value from index in var\_a, store into $t1

#Loaded val of var\_b at current index of $a1 into $t1

add $a1, $a1, -1 #Decrease the index to get the previous element

la $t2, var\_a #Load var\_a array into $t2.

mul $t0, $a1, 4 #Get pointer reference by mult index by 4.

add $t0, $t2, $t0 #Go to the index from pointer in array.

lw $t2, ($t0) #Get value from index in var\_a, store into $t2.

#Loaded val of var\_a at current index of $a1 into $t2

sub $t0, $t2, $t1 #If num at a is greater than a-1...

bgtz $t0, swap #Jump to swap, else...

j moveup #Moveup to the next index to check

swap:

la $t3, var\_a

mul $t0, $a1, 4

add $t0, $t3, $t0

sw $t1, ($t0)

addi $a1, $a1, 1

la $t3, var\_a

mul $t0, $a1, 4

add $t0, $t3, $t0

sw $t2, ($t0)

addi $a1, $a1, -2

begofarray:

addi $a1, $a1, -1

moveup:

add $a1, $a1, 2

j loopcheck

bbye:

add $a1, $zero, $zero

print\_str(done)

print:

sub $t0, $s1, $a1 #If sorted index of a < length of a...

blez $t0, exit #Jump to second check, else...

la $a2, var\_a #Load var\_a array into $a2.

mul $t0, $a1, 4 #Get pointer reference by mult index by 4.

add $t0, $a2, $t0 #Go to the index from pointer in array.

lw $a2, ($t0) #Get value from index in var\_a, store into $s2 as min.

print\_reg\_int($a1)

print\_str(sepe)

print\_reg\_int($a2)

print\_str(newl)

add $a1, $a1, 1

j print

exit:

li $v0, 10

syscall

***Problem #3 Code***

#======================================

#$t0 is general use number

#======================================

#$v0 & $v1 are returns for functions

#$a0 & $a1 are arguments for functions

#======================================

.include "cs47\_macro.asm"

.data

.align 2

var\_a: .word 2 3 5 5 5 6 7 8 8 10 11 14 15 17 17 18 20

#var\_a: .word 15 18 21

var\_m: .word 17

#var\_m: .word 3

gcdstr: .asciiz "GCD of array is: "

lcmstr: .asciiz "LCM of array is: "

newl: .asciiz "\n"

.text

.globl main

main:

add $a0, $zero, $zero #index of var\_a

lw $a1, var\_m #length of var\_a

jal gcd\_lcm\_set

move $t0, $v0

move $t1, $v1

print\_str(gcdstr)

print\_reg\_int($t0)

print\_str(newl)

print\_str(lcmstr)

print\_reg\_int($t1)

print\_str(newl)

li $v0, 10

syscall

#------------------------------------------------------------------------------

# Function: gcd\_lcm\_set

# Argument:

# $a0 : +ve integer number n1, beginning of array

# $a1 : +ve integer number n2, number of elements of array

# Returns

# $v0 : gcd of array from n1 to n2

# $v1 : lcm of array from n1 to n2

#

# Purpose: Implementing the recursive gcd function to

# get the Greatest Common Divisor of n1 and n2

#

# $s0 is the index of the array

# $s1 is the length of the array

# $s2 is the latest GCD of the array

# $s3 is the latest LCM of the array

# $s4 is the value at current index of array

# $s6 is the call save for GCD\_LCM\_SET

# $s7 is the call save for LCM

#------------------------------------------------------------------------------

gcd\_lcm\_set:

move $s6, $ra

move $s0, $a0

move $s1, $a1

la $s2, var\_a #Load var\_a array into $a1.

mul $t0, $s0, 4 #Get pointer reference by mult index $a3 by 4.

add $t0, $s2, $t0 #Go to the index from pointer in array.

lw $s2, ($t0) #Get value from index in var\_a, store into $a1

move $s3, $s2

loopcheck:

addi $s0, $s0, 1

sub $t0, $s1, $s0 #If index of a < length of a...

bgtz $t0, looproutine #Jump to second check, else...

move $v0, $s2 #Since array is empty...

move $v1, $s3 #Move values into correct position...

jr $s6 #And exit.

looproutine:

la $s4, var\_a #Load var\_a array into $s7.

mul $t0, $s0, 4 #Get pointer reference by mult index $s0 by 4.

add $t0, $s4, $t0 #Go to the index from pointer in array.

lw $s4, ($t0) #Get value from index in var\_a, store into $s7

#Loaded val of var\_a at current index of $s0 into $s7

move $a0, $s2 #Put last GCD into arg 1

move $a1, $s4 #Put new array num into arg 2

jal gcd

move $s2, $v0 #Put new GCD into last GCD

move $a0, $s3 #Put last LCM into arg 1

move $a1, $s4 #Put new array num into arg 2

jal lcm

move $s3, $v0 #Put new GCD into last GCD

j loopcheck

#------------------------------------------------------------------------------

# Function: gcd

# Argument:

# $a0 : +ve integer number n1

# $a1 : +ve integer number n2

# Returns

# $v0 : gcd(n1, n2)

#

# Purpose: Implementing the recursive gcd function to

# get the Greatest Common Divisor of n1 and n2

#------------------------------------------------------------------------------

gcd:

sub $t0, $a0, $a1

beqz $t0, gcdA

bgtz $t0, gcdB

bltz $t0, gcdC

gcdA:

move $v0, $a0

jr $ra

gcdB:

sub $a0, $a0, $a1

j gcd

gcdC:

sub $a1, $a1, $a0

j gcd

#------------------------------------------------------------------------------

# Function: lcm

# Argument:

# $a0 : +ve integer number n1

# $a1 : +ve integer number n2

# Returns

# $v0 : lcm(n1, n2)

#

# Purpose: Implementing the iterative lcm function to

# get the Least Common Multiple of n1 and n2

#

# $t2 is a hold for n1

# $t3 is a hold for n2

# $t4 is a hold for gcd(n1, n2)

#------------------------------------------------------------------------------

lcm:

move $s7, $ra

move $t2, $a0

move $t3, $a1

jal gcd

move $t4, $v0

mul $t2, $t2, $t3

div $t2, $t2, $t4

move $v0, $t2

jr $s7