**Computer Organization and Architecture**

**Post-Lab Report**

**Lab 04**



|  |  |
| --- | --- |
| Group Members Name & Reg #: | **Muhammad Haris Irfan**  **(FA18-BCE-090)** |
|  |  |
| Class | Computer Organization and Architecture CPE343(**BCE-5B**) |
| Instructor’s Name | Dr. Adeel Israr |

POST LAB

Question:

Write a program that first inputs total no. of student. It then gets student roll no. and marks secured by all students in 5 subjects, and stores both in memory. After that it gives Maximum, Minimum and Median of all subjects.

Solution:

I am attaching my commented code below,

#Title : Lab 4 Postlab. Filename: Lab4Postlab.asm

#Author: Muhammad Haris Irfan. Date: 11-10-20

#Roll Number: FA18-BCE-090 Description: Maximum, Minimum and Median

#Registers: $t1, $t2, $t3,$t4, $t5, $v0, $a0

########################### Data Segment ############################################################

.data

co: .word 0:10

mc: .word 0:10

re: .word 0:10

os: .word 0:10

db: .word 0:10

msg1: .asciiz "Enter the Number of Students: "

msg2: .asciiz "Enter the marks for CO: "

msg3: .asciiz "\nresult: "

msg4: .asciiz "\nMIN is: "

msg5: .asciiz "\nMAX is:"

msg6: .asciiz "\n\nEnter the marks for CO: "

msg7: .asciiz "Enter the marks for DE: "

msg8: .asciiz "Enter the marks for RAES: "

msg9: .asciiz "Enter the marks for OS: "

msg10: .asciiz "Enter the marks for DB: "

msg11: .asciiz "\n\n FOR COMP ORGAN: "

msg12: .asciiz "\n\n FOR MULTI CAL: "

msg13: .asciiz "\n\n FOR RE: "

msg14: .asciiz "\n\n FOR Operating Systems: "

msg15: .asciiz "\n\n FOR Data Bases: "

msg16: .asciiz "\nMEDIAN is: "

################### Code Segment ######################################################

.text

main:

la $a0,msg1 #Load msg1

li $v0,4

syscall

li $v0,5

syscall

move $t0,$v0 #t0 stores the number of students

move $s7,$v0 #$s7 will be used in median calculation

div $s7,$s7,2

mul $s7,$s7,4 #mul and div performed to get to the mid of the sorted array

li $t1, 0 #for loops

li $s0,0 #(pointer position)

###############################################

loop:

beq $t1,$t0,here #loop would run for t0 number of times

####################

# THE NEXT FEW STEPS ENTER DATA INTO THE ARRAYS #

###################

la $a0,msg6

li $v0,4

syscall

li $v0,5

syscall

sw $v0, co($s0)

la $a0,msg7

li $v0,4

syscall

li $v0,5

syscall

sw $v0, mc($s0)

la $a0,msg8

li $v0,4

syscall

li $v0,5

syscall

sw $v0, re($s0)

la $a0,msg9

li $v0,4

syscall

li $v0,5

syscall

sw $v0, os($s0)

la $a0,msg10

li $v0,4

syscall

li $v0,5

syscall

sw $v0, db($s0)

addiu $s0,$s0,4 #array pointer increminted with 4

addiu $t1,$t1,1 #$t1++

j loop

here:

##########################################################################

jal initialize #initializes the values of array pointer, and loop variables hich were changed during the course of last loop

loop1:

beq $t0,$t1,here1 #loop runs for t0 times

lw $t4,co($s0)

jump1:

bgt $t5,$t4,min1 #bgt branch greater than if t5 < t4 min1 calculates min and stores it in t5

bgt $t4,$t6,max1 #calculates max and stores in t6

addiu $s0,$s0,4

addiu $t1,$t1,1

j loop1

here1:

la $a0,msg11

move $t1,$t0 #for sorting

la $t9, co #t9 is the array pointer here

jal sort #to jump to sort function

jal output #to jump to output function

jal output1 #to jump to output1 function to pront median

jal initialize

loop2:

beq $t0,$t1,here2

lw $t4,mc($s0)

jump2:

bgt $t5,$t4,min2

bgt $t4,$t6,max2

addiu $s0,$s0,4

addiu $t1,$t1,1

j loop2

here2:

la $a0,msg12

la $t9, mc

jal sort

jal output

jal output2

jal initialize

loop3:

beq $t0,$t1,here3

lw $t4,re($s0)

jump3:

bgt $t5,$t4,min3

bgt $t4,$t6,max3

addiu $s0,$s0,4

addiu $t1,$t1,1

j loop3

here3:

la $a0,msg13

la $t9, re

jal sort

jal output

jal output3

jal initialize

loop4:

beq $t0,$t1,here4

lw $t4,os($s0)

jump4:

bgt $t5,$t4,min4

bgt $t4,$t6,max4

addiu $s0,$s0,4

addiu $t1,$t1,1

j loop4

here4:

la $a0,msg14

la $t9, os

jal sort

jal output

jal output4

jal initialize

loop5:

beq $t0,$t1,here5

lw $t4,db($s0)

jump5:

bgt $t5,$t4,min5

bgt $t4,$t6,max5

addiu $s0,$s0,4

addiu $t1,$t1,1

j loop5

here5:

la $a0,msg15

la $t9, db

jal sort

jal output

jal output5

li $v0,10

syscall

###########################################

## FUNCTIONS AND JUMPS ##

###########################################

### so many min and max functions were made so the program would jump back to the accurate position.###

min1:

move $t5,$t4

j jump1

max1:

move $t6,$t4

j jump1

min2:

move $t5,$t4

j jump2

max2:

move $t6,$t4

j jump2

min3:

move $t5,$t4

j jump3

max3:

move $t6,$t4

j jump3

min4:

move $t5,$t4

j jump4

max4:

move $t6,$t4

j jump4

min5:

move $t5,$t4

j jump5

max5:

move $t6,$t4

j jump5

####################################################

########### OUTPUTS ##############

####################################################

initialize: #intializes all variables to be used in a loop

li $t1,0

li $s0,0 #(pointer position)

li $t5,101

li $t6,0

jr $ra

output:

li $v0,4

syscall

la $a0,msg4

li $v0,4

syscall

li $v0,1

move $a0,$t5

syscall

la $a0,msg5

li $v0,4

syscall

li $v0,1

move $a0,$t6

syscall

la $a0,msg16

li $v0,4

syscall

jr $ra

output1:

lw $t7, co($s7)

li $v0,1

move $a0,$t7

syscall

jr $ra

output2:

lw $t7, mc($s7)

li $v0,1

move $a0,$t7

syscall

jr $ra

output3:

lw $t7, re($s7)

li $v0,1

move $a0,$t7

syscall

jr $ra

output4:

lw $t7, os($s7)

li $v0,1

move $a0,$t7

syscall

jr $ra

output5:

lw $t7, db($s7)

li $v0,1

move $a0,$t7

syscall

jr $ra

########################################

## TRANSLATED FROM A C++ PROGRAM ##

########################################

sort:

addiu $sp, $sp, -4

sw $ra, 0($sp) # store $ra into the stack

sll $t1, $t1, 2 # shifing to left or mul by 2

li $v0, 0 # int i = 0

looop:

slt $t3, $v0, $t1 # if (i < n) => $t3 = 1

beq $t3, $zero, end # while (i < n) {

bne $v0, $zero, compare # if (i == 0)

addiu $v0, $v0, 4 # i = i + 1

compare:

addu $t2, $t9, $v0 # $s2 = &arr[i]

lw $t7, -4($t2) # $t4 = arr[i-1]

lw $t8, 0($t2) # $t5 = arr[i]

blt $t8, $t7, swap # swap if (arr[i] < arr[i-1])

addiu $v0, $v0, 4 # i = i+ 1

j looop

swap:

sw $t7, 0($t2) # swap (arr[i], arr[i-1])

sw $t8, -4($t2)

addiu $v0, $v0, -4 # i = i - 1

j looop

end:

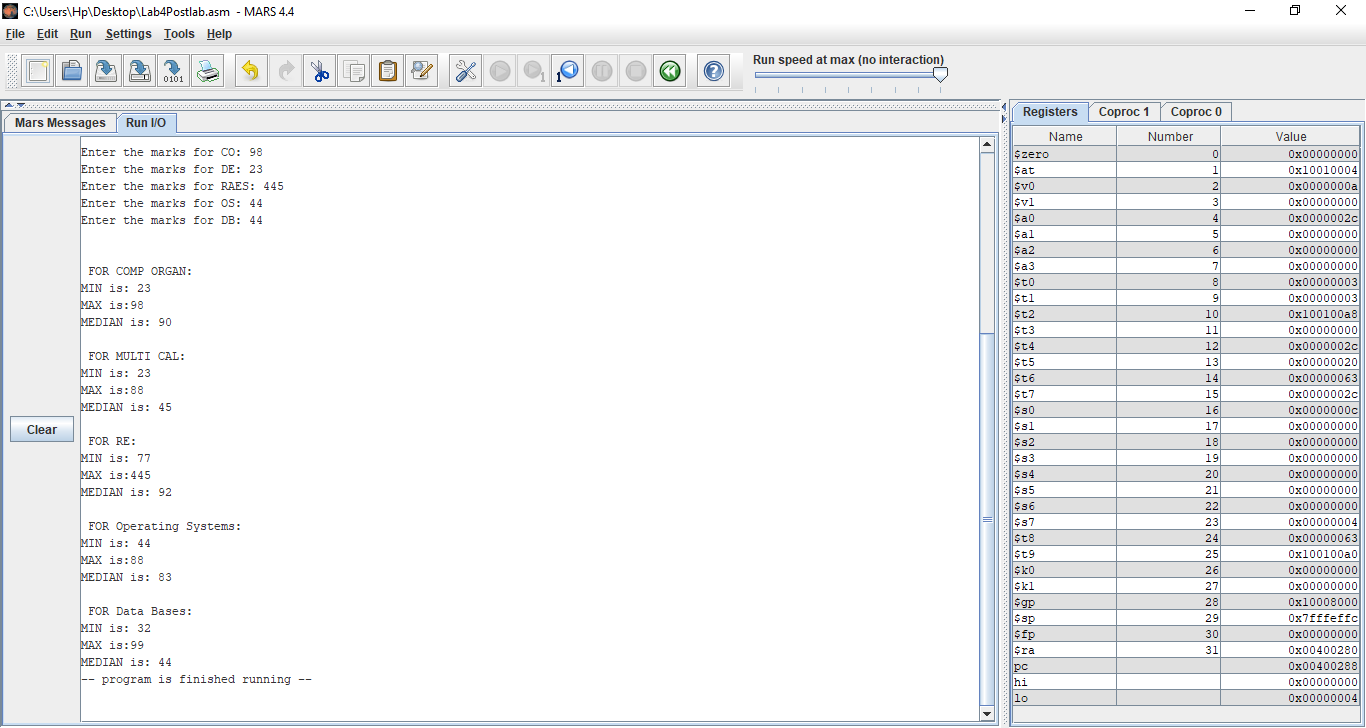
srl $t1, $t1, 2

lw $ra, ($sp) # copy from stack to $ra

addi $sp, $sp, 4 # increment stack pointer by 4

jr $ra # return to main

The result for this program is shown below,



Flow chart of thtask:

Diagram, schematic

Description automatically generated

Critical Analysis/ Conclusion:

In This lab, we learnt about accessing Arrays, through implementation we learnt how to use lw( Load word) and sw(Store word) commands to put and get data from an Array. We also implemented these commands in Different tasks. Moreover, we made handwritten Flowcharts for each task, depicting the working of our code.

\_\_\_\_\_\_THE END\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_