**Bahria University,**

**Karachi Campus**

****

**COURSE:** CEN-221 COMPUTER ARCHITECTURE AND ORGANIZATION

**TERM: Fall 2021, CLASS: BSE- 3(B)**

**PROJECT REPORT**

**Project Title:**

**“ARRAY MANIPULATION”**

**Submitted By:**

**Fizza Mukhtar (65156)**

**Iqra Afzaal (65161)**

**Kinza Noor (65141)**

**Aqsa Shamsher (65148)**

**Submitted to:**

Engr. Muhammad Rehan Baig

**Signed: Remarks: Score:**

**Table of Contents**

**CHAPTER 1**3

INTRODUCTION03

BACKGROUND 04

CHAPTER 25

**CONCEPTS**05

**CODE**06

**OUTPUT SCREEN**18

**REGISTER VALUS**22

End page 26

**INTRODUCTION**

Our project is based on algorithm applied to sort integer in an array. User can input the number in an array and then has the functionality to perform sorting, searching and displaying the input array.

**BACKGROUND**

Searching and sorting is useful while handling different data. In previous time different algorithm has been introduce by developers and coders that uses different techniques to search and sort data and integers. In this project we are using one of the technique to sort and search data in efficient way using MIPS language.

**CONCEPTS**

**Following are the important concepts of MIPs Assembly Language used in this project:**

1. **Array Processing**
2. **Memory Access**
3. **For Loops; Control Structures**
4. **If/Else; Control Structures**
5. **Unconditional Structures**
6. **Addu Instructions**
7. **Remainder/Quotient Instructions**

**CODE & DESCRIPTION**

**CODE**

.data

amount\_num: .asciiz "How many numbers do you have? "

num: .asciiz "Enter a number: "

nl: .asciiz "\n"

array\_contains: .asciiz "\nThe array contains the following elements: \n"

complete: .asciiz "\nProgram complete!!!"

.align 2

array: .space 40

search\_num: .asciiz "\nEnter number to search for: "

f: .asciiz " was found at array["

cl\_bracket: .asciiz "]"

nf: .asciiz " was not found"

c: .asciiz "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_COMPUTER ARCHITECTURE AND ORGANIZATION CEN -221 PROJECT\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

APP:.asciiz "PROJECT:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_INTEGERS SEARCHING AND SORTING APPLICATION\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n"

GM:.asciiz "-----------------------GROUP MEMBERS--------------------------\n"

GM1: .asciiz "1. IQRA AFZAAL (02-131192-048)\n"

GM2: .asciiz "2. KINZA NOOR (02-131192-025)\n"

GM3: .asciiz "3. FIZZAH MUKHTAR (02-131192-042)\n"

GM4: .asciiz "4. AQSA SHAMSHER (02-131192-033)\n\n"

M: .asciiz "Dear User! Do you want to store data in Collection?\n\_\_\_\_\_Enter 1 if you want or 0 if you don't want to store:\n"

nd: .asciiz "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*HAVE A NICE DAY! :)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

st: .asciiz "YOUR DATA HAS BEEN SUCCESSFULLY STORED!!!\n"

so: .asciiz "\nSorted Collection of Numbers is given below!\n"

se: .asciiz "Do you want to search any element in your collection?\n\_\_\_\_\_\_Enter 1 if you want or 0 if you don't want to search any element:\t"

.text

.globl main

main:

li $v0, 4

la $a0, c

syscall

li $v0, 4

la $a0, APP

syscall

li $v0, 4

la $a0, GM

syscall

li $v0, 4

la $a0, GM1

syscall

li $v0, 4

la $a0, GM2

syscall

li $v0, 4

la $a0, GM3

syscall

li $v0, 4

la $a0, GM4

syscall

li $v0, 4

la $a0, M

syscall

li $v0, 5

syscall

move $a0, $v0

beq $a0, 1 , start

bne $a0, 1, exit

start:

jal enter\_amount # Prompts the user to enter the amount of integers to be entered.

jal enter\_num # Enter the integers into the array.

jal contains # Prints the contents of the array.

jal co

jal sort # Sorts the integers in the array using a bubble sort.

jal contains # Prints the sorted contents of the array.

jal ASEAR

jal search\_for # Enter the integer you're searching for.

jal search # Searches the array using a iterative binary search.

jal search\_results # Prints the results of the search for the integer.

j done # Exit

enter\_amount:

la $a0, amount\_num # Asks the user to input the number of elements for the array.

li $v0, 4 # Prints the message.

syscall

li $v0, 5 # Reads the user's input (number of elements).

syscall

move $t0, $v0 # Stores the user's input (number of elements) in $t0.

li $t1, 0 # Offset for the array: array[x] = array[0]

li $t2, 1 # Counter for the number of integers printed: i = 1

jr $ra # Return to main.

enter\_num:

addi $t2, $t2, 1 # Increment the counter: i++

la $a0, num # Asks the user to enter an integer.

li $v0, 4 # Prints the message.

syscall

li $v0, 5 # Reads the integer that the user inputted.

syscall

sw $v0, array($t1) # Stores the inputted integers into the array.

addi $t1, $t1, 4 # Increments the address of the array: array[x+1].

ble $t2, $t0, enter\_num # if $t2 (the counter) <= $t0 (total # of integers): enter\_num

jr $ra # Return to main.

contains:

li $t1, 0 # Resets the offset for the array: array[x] = array[0]

li $t2, 1 # Resets the counter for print the integers: i = 0

la $a0, array\_contains # Prints "The array contains the following: "

li $v0, 4 # Prints the message.

syscall

print\_array:

addi $t2, $t2, 1 # Increment the counter: i++

lw $a0, array($t1) # Loads the integer into $a0: a = array[x].

li $v0, 1 # Prints the integer.

syscall

la $a0, nl # nl = newline (The next integer will be printed on a newline).

li $v0, 4 # Prints a newline.

syscall

addi $t1, $t1, 4 # Increments the offset of the array: x+1

ble $t2, $t0, print\_array # if $t2 (integer counter) <= $t0 (total # of integers): print\_array

jr $ra # Return to main.

co:

li $v0, 4

la $a0, st

syscall

jr $ra

ASEAR: li $v0, 4

la $a0, se

syscall

li $v0, 5

syscall

move $a0,$v0

bne $a0, 1, exit

jr $ra

* **SORTING:**

sort:

li $t2, 0 # i = 0

outer:

addi $t2, $t2, 1 # Increments i: i++

la $a1, array # Load array address into $a1

li $t1, 0 # array[x] = array[0]

sub $t3, $t0, 1 # $t3 = n - 1 (n = total # of integers)

addi $t4, $t2, 1 # j = i + 1

ble $t2, $t3, inner # if i <= (n-1): inner

li $v0, 4

la $a0, so

syscall

jr $ra # Return to main.

inner:

lw $t5, 0($a1) # Ex: a = 6

lw $t6, 4($a1) # Ex: b = 5

bgt $t5, $t6, swap # if a > b: swap

j continue # else: continue

swap:

sw $t6, 0($a1) # Ex: a = 5

sw $t5, 4($a1) # Ex: b = 6

continue:

addi $a1, $a1, 4 # Array[x+1]

addi $t4, $t4, 1 # j + 1

bgt $t4, $t0, outer # if (j = i + 1) > n: outer

j inner # else: inner

* **ELEMENTS TO SEARCH FOR:**

search\_for: # Enter the integer you're looking for.

la $a0, search\_num # Prompts the user to enter an integer.

li $v0, 4 # Prints the message.

syscall

li $v0, 5 # Reads the integer.

syscall

move $t0, $v0 # Stores the integer to search for in $t0

li $t2, 0 # First element in the array: array[0]

li $t4, 2 # Used to divide the amount of elements in the array by two.

li $t6, 4 # Used to increment the array

jr $ra # Return to main.

* **SEARCHING OF ENTRIES:**

search:

bgt $t2, $t3, return # if $t2(first element) > $t3(last element (n-1)): The number wasn't found and return to main

middle\_num:

add $t7, $t2, $t3 # Store the sum of $t2(first element) and $t3(last element(n-1)) in $t7.

div $t7, $t4 # $s0 # Get the middle element: $t7 / $t4 = (first element + last element) / 2.

mflo $t8 # Store the middle element in $t8.

mult $t8, $t6 # Multiply the middle element by 4 to get the index for the array.

mflo $t1 # Store the product in $t1.

lw $t5, array($t1) # Load the value of array[(middle) \* 4] in $t5.

# Ex: middle = 5, x = 5 \* 4 = 20, y = array[x]

bgt $t5, $t0, lower # if middle element > x (User's desired integer): check the lower half of the array

blt $t5, $t0, upper # if middle element < x (User's desired integer): check the upper half of the array

li $s0, 1 # The integer was found (1 = true)

jr $ra # if middle element == x (User's desired integer): return to main

upper: # Upper half of the array.

add $t2, $t8, 1 # $t2 (first element) = $t8 (middle element) + 1

j search # Jump to search and check the rest of the array.

lower: # Lower half of the array.

sub $t3, $t8, 1 # $t3 (last element (n-1)) = $t8 (middle element) - 1

j search # Jump to search and check the rest of the array.

return:

li $s0, 0 # The integer wasn't found (0 = false)

jr $ra # Return to main.

search\_results:

beqz $s0, not\_found # if $s0 == 0: not\_found

found:

move $a0, $t0 # Store the user's desired integer in $a0 to print.

li $v0, 1 # Print the user's integer.

syscall

la $a0, f # Prompt the user that the integer was found.

li $v0, 4 # Print the message.

syscall

div $t1, $t6 # Divide the offset of the array by 4 to get the

mflo $a0 # location of the number in the array and store it in $a0.

li $v0, 1 # Print the location of the number.

syscall

la $a0, cl\_bracket # Print ] for the location of the number.

li $v0, 4 # Print the bracket.

syscall

jr $ra # Return to main.

not\_found:

move $a0, $t0 # Store the user's desired integer in $a0 to print.

li $v0, 1 # Print the user's integer.

syscall

la $a0, nf # Prompt the user that the integer wasn't found.

li $v0, 4 # Print the message.

syscall

jr $ra # Return to main.

done:

la $a0, complete # Tells the user that the program is complete.

li $v0, 4 # Prints the message.

Syscall

* **EXIT:**

exit:

li $v0, 4

la $a0, nd

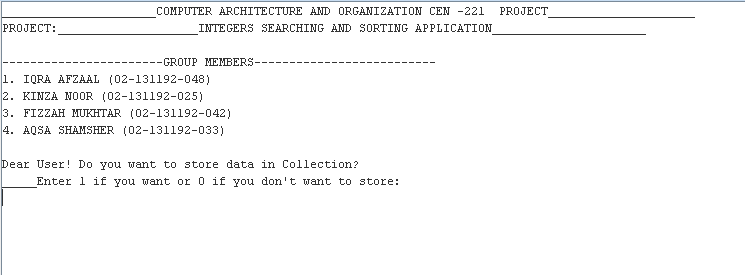
syscall

li $v0, 10 # Ends the program.

syscall

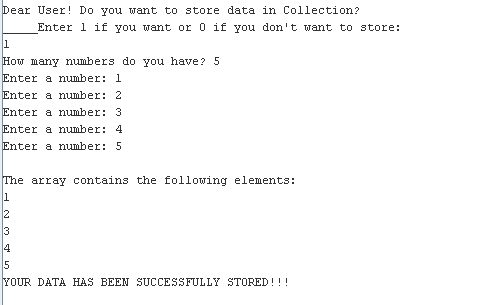
**OUTPUT SCREENS**

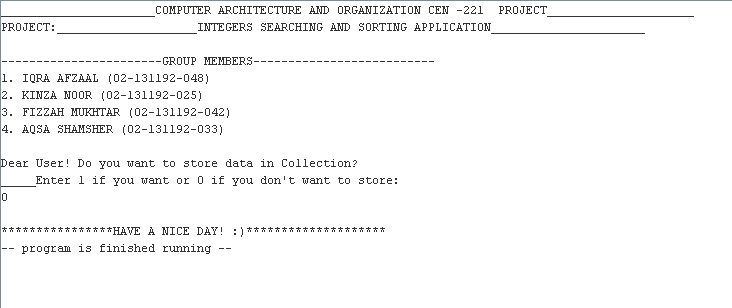
* **MAIN SCREEN:**

****

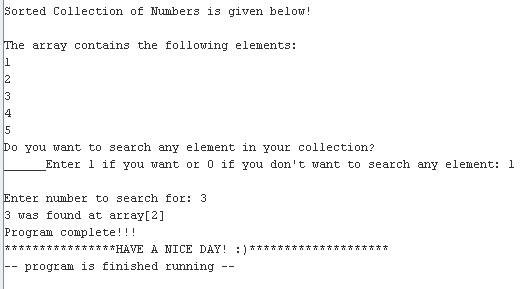
* **STORING DATA INTO AN ARRAY:**

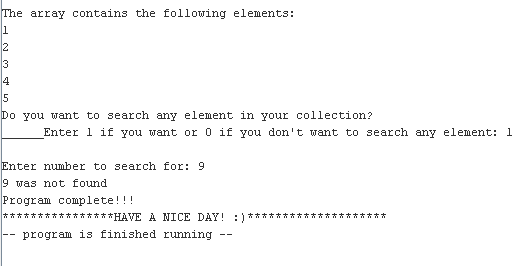
1. **IF USER SELECTS 1 (Wants to store):**

****

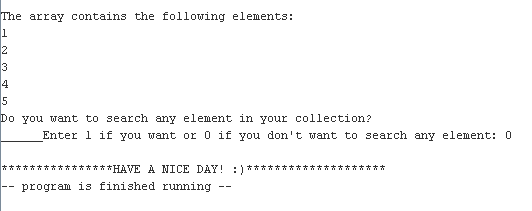
1. **IF USER SELECTS 0 (Does not want to store):**

* **SEARCHING DATA:**

1. **IF USER SELECTS 1 (Wants to search):**
2. **SEARCHED ITEM FOUND:**
3. **SEARCHED ITEM NOT FOUND:**

****

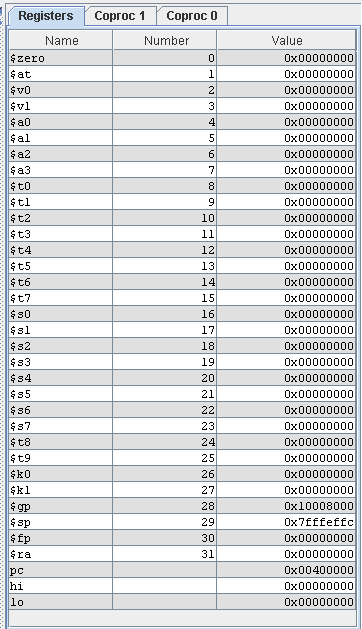
1. **IF USER SELECTS 0 (Does not want to search):**

****

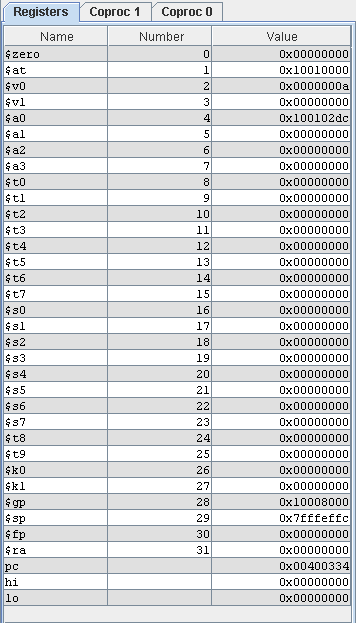
**REGISTER VALUES**

* **STORING DATA INTO AN ARRAY:**

1. **IF USER SELECTS 1 (Wants to store):**

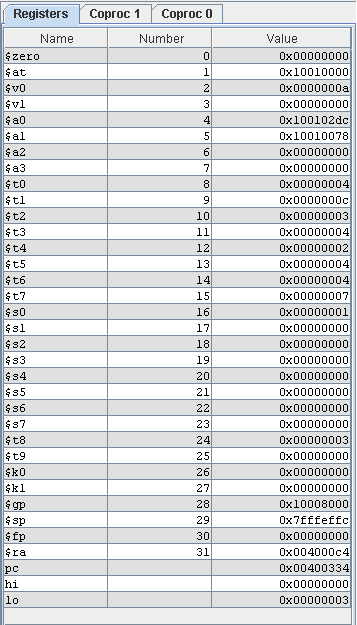


1. **IF USER SELECTS 0 (Does not want to store):**

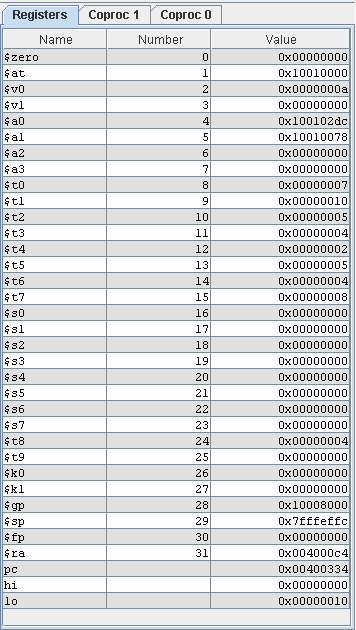


* **SEARCHING DATA:**

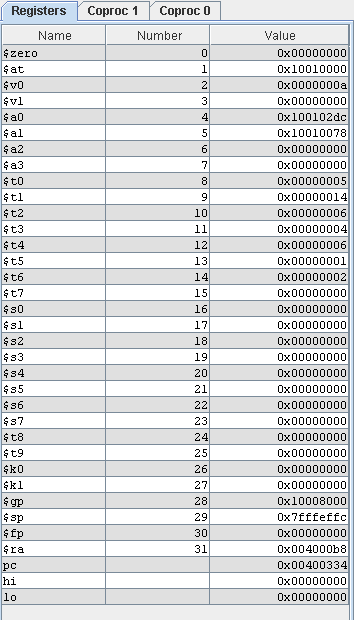
1. **IF USER SELECTS 1 (Wants to search):**
2. **SEARCHED ITEM FOUND:**



1. **SEARCHED ITEM NOT FOUND:**



1. **IF USER SELECTS 0 (Does not want to search):**



|  |
| --- |
|  |