

|  |
| --- |
| Computer Architecture and Organization Report. |
|  |
| Submitted By:  Hasnain Khalid  Nausheen Kaleem  Areej Fatima  Syeda Zoya Hasan  Class : BSE-3B |

Table of Contents

[*Introduction:* 2](#_Toc62172178)

[*Abstract:* 4](#_Toc62172179)

[*Project Overview:* 4](#_Toc62172180)

[*Description:* 4](#_Toc62172181)

[*Code:* 5](#_Toc62172182)

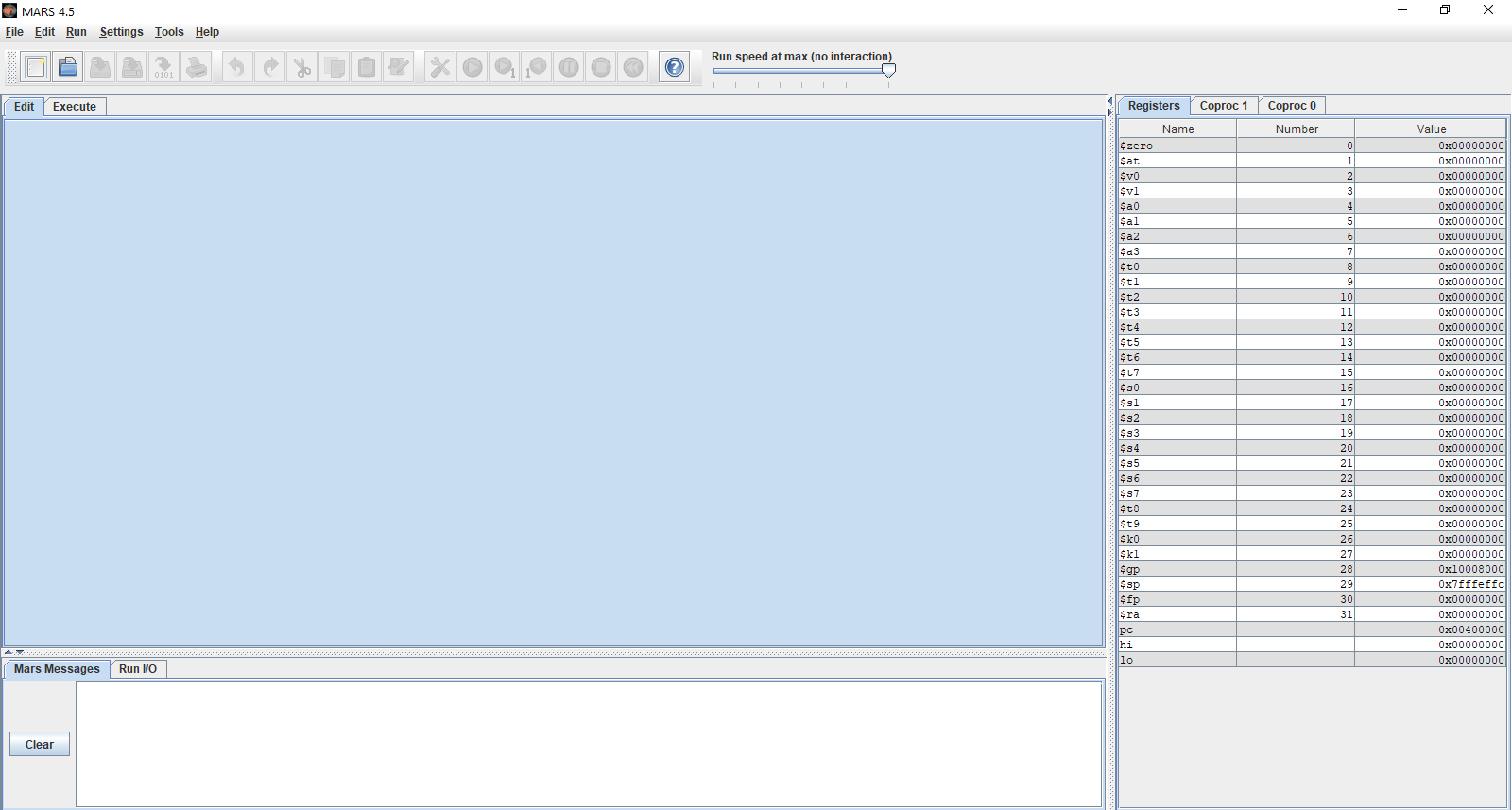
[*Output:* 17](#_Toc62172183)

[*Register value:* 18](#_Toc62172184)

[*Conclusion:* 19](#_Toc62172185)

# *Introduction:*

MIPS (originally an acronym for Microprocessor without Interlocked Pipeline Stages) is a Reduced Instruction Set Computer (RISC). MIPS is a register based architecture, meaning the CPU uses registers to perform operations on. Registers are memory just like RAM, except registers are much smaller than RAM, and are much faster. In MIPS the CPU can only do operations on registers, and special immediate values. MIPS processors have 32 registers, but some of these are reserved. A fair number of registers however are available for our use.



# *Abstract:*

A MIPS program implementing data structures to declare, search and sort in an array. In order to meet this end appropriate algorithms were designed and implemented. In HLL, an array is a multi-valued variable: there may be multiple values in a single array variable. Arrays would be identical in MIPS assembly; however, assembly abstraction of an array is very limited. An array is implemented in the MIPS assembly by storing several values in contiguous memory areas and accessing each value in the array as an array value offset.

# *Project Overview:*

This program consists of the following features:

1. Inputting An Array list
2. Displaying Array list
3. Sorting Array list
4. Binary Searching of an element

# *Description:*

The MIPS based program provides a way for the user to enter a number of integers in the form of an array. The user is able to sort these numbers accordingly and also search for a certain element in the array. The code contains several methods for different functions of the program such as printing the array, printing sorted array in ascending and descending form and also search methods so the user is able to search for an element in the array.

First the program asks the user the number of elements they want to store in the array. When the user provides a value, they enter the elements they want in the array list. The integers are then printed in the form of an array. Using bubble sort, the unsorted array is arranged in ascending and descending order and displayed in the program. The user is prompted to enter the element they want to search in the array. When the user enters the element, the program performs an iterative binary search among the elements of the array. If the element is present in the array, the program displays that the element is found. But if the element is not present in the array then the displays shows element not found and asks the user if they would want to search again for a different element. If the user chooses too search again the searching process is repeated and if the element is found the output displays element found and the program is ended.

# *Code:*

**Data Segment:**

.data

prompt: .asciiz " Welcome to Mips assembly Langauge !!!\n"

prompt1: .asciiz "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

prompt2: .asciiz "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_NUMERIC ARRAY\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_!!\n"

Length: .asciiz "How many elements you want to store in array list ? "

prompt3: .asciiz "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

items: .asciiz "Enter any integer value: "

prompt4: .asciiz "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

Unsorted\_array: .asciiz " !! UNSORTED ARRAY LIST !! \n"

nl: .asciiz "\n"

List: .asciiz "List: { "

tl: .asciiz " "

t2: .asciiz " "

B: .asciiz "}"

display\_descending: .asciiz " !! SORTED ARRAY IN DESCENDING ORDER !! \n"

display\_ascending: .asciiz " !! SORTED ARRAY IN ASCENDING ORDER !! \n"

searching: .asciiz " !! SEARCHING OF REQUIRED ELEMENT !! "

searching\_element: .asciiz "\nPlease insert a element which you want to search: "

f: .asciiz " was found at array index ["

cl\_bracket: .asciiz "]"

nf: .asciiz " was not found"

terminate: .asciiz "\nProgram complete"

again\_search: .asciiz "\nDo you want to search again ? if yes then press 1 :"

end: .asciiz "\nTHANK YOU !!!"

.align 2

array: .space 40

**Code Segment:**

.text

.globl main

main:

li $v0,4

la $a0,prompt

syscall

li $v0,4

la $a0,prompt1

syscall

li $v0,4

la $a0,prompt2

syscall

jal Enter\_Length

jal Enter\_items

jal Display\_Unsorted\_arrayV

jal Display\_Unsorted\_arrayH

jal sort\_array\_descending

jal sorted\_array\_descending\_displayV

jal sorted\_array\_descending\_displayH

jal sort\_array\_ascending

jal sorted\_array\_ascending\_displayV

jal sorted\_array\_ascending\_displayH

jal search\_element

jal search

jal search\_result

jal again

jal search

jal search\_result

j Exit

**Methods:**

***Enter\_Length:***

Enter\_Length:

la $a0,Length

li $v0, 4

syscall

li $v0, 5

syscall

move $t0, $v0

li $v0,4

la $a0,prompt3

syscall

li $t1, 0

li $t2, 1

jr $ra

***Enter\_items:***

addi $t2, $t2, 1

la $a0,items

li $v0, 4

syscall

li $v0, 5

syscall

sw $v0, array($t1)

addi $t1, $t1, 4

ble $t2, $t0, Enter\_items

li $v0,4

la $a0,prompt4

syscall

jr $ra

***Display\_Unsorted\_arrayV:***

li $t1, 0

li $t2, 1

la $a0,Unsorted\_array

li $v0, 4

syscall

***Print\_Unsorted\_ArrayV:***

ddi $t2, $t2, 1

lw $a0, array($t1)

li $v0, 1

syscall

la $a0, nl

li $v0, 4

syscall

addi $t1, $t1, 4

ble $t2, $t0,Print\_Unsorted\_ArrayV

jr $ra

***Display\_Unsorted\_arrayH:***

li $t1, 0

li $t2, 1

la $a0,List

li $v0, 4

syscall

***Print\_Unsorted\_ArrayH:***

addi $t2, $t2, 1

lw $a0, array($t1)

li $v0, 1

syscall

la $a0, tl

li $v0, 4

syscall

addi $t1, $t1, 4

ble $t2, $t0,Print\_Unsorted\_ArrayH

la $a0, B

li $v0, 4

syscall

li $v0,4

la $a0,prompt4

syscall

jr $ra

***sort\_array\_descending:***

li $t2, 0

outer:

addi $t2, $t2, 1

la $a1, array

li $t1, 0

sub $t3, $t0, 1

addi $t4, $t2, 1

ble $t2, $t3, inner

jr $ra

inner:

lw $t5, 0($a1)

lw $t6, 4($a1)

blt $t5, $t6, swap

j continue

swap:

sw $t6, 0($a1)

sw $t5, 4($a1)

continue:

addi $a1, $a1, 4

addi $t4, $t4, 1

bgt $t4, $t0, outer

j inner

***sorted\_array\_descending\_displayV:***

li $t1, 0

li $t2, 1

la $a0,display\_descending

li $v0, 4

syscall

***print\_sorted\_arrayV:***

addi $t2, $t2, 1

lw $a0, array($t1)

li $v0, 1

syscall

la $a0, nl

li $v0, 4

syscall

addi $t1, $t1, 4

ble $t2, $t0, print\_sorted\_arrayV

jr $ra

***sorted\_array\_descending\_displayH:***

li $t1, 0

li $t2, 1

la $a0,List

li $v0, 4

syscall

***print\_sorted\_arrayH:***

addi $t2, $t2, 1

lw $a0, array($t1)

li $v0, 1

syscall

la $a0,t2

li $v0, 4

syscall

addi $t1, $t1, 4

ble $t2, $t0, print\_sorted\_arrayH

la $a0, B

li $v0, 4

syscall

li $v0,4

la $a0,prompt4

syscall

jr $ra

***sort\_array\_ascending:***

li $t2, 0

out:

addi $t2, $t2, 1

la $a1, array

li $t1, 0

sub $t3, $t0, 1

addi $t4, $t2, 1

ble $t2, $t3, inn

jr $ra

inn:

lw $t5, 0($a1)

lw $t6, 4($a1)

bgt $t5, $t6, swp

j cont

swp:

sw $t6, 0($a1)

sw $t5, 4($a1)

cont:

addi $a1, $a1, 4

addi $t4, $t4, 1

bgt $t4, $t0, out

j inn

***sorted\_array\_ascending\_displayV:***

li $t1, 0

li $t2, 1

la $a0,display\_ascending

li $v0, 4

syscall

print\_sorted\_array:

addi $t2, $t2, 1

lw $a0, array($t1)

li $v0, 1

syscall

la $a0, nl

li $v0, 4

syscall

addi $t1, $t1, 4

ble $t2, $t0, print\_sorted\_array

jr $ra

***sorted\_array\_ascending\_displayH:***

li $t1, 0

li $t2, 1

la $a0,List

li $v0, 4

syscall

***print\_sorted\_array\_as:***

addi $t2, $t2, 1

lw $a0, array($t1)

li $v0, 1

syscall

la $a0,t2

li $v0, 4

syscall

addi $t1, $t1, 4

ble $t2, $t0, print\_sorted\_array\_as

la $a0, B

li $v0, 4

syscall

li $v0,4

la $a0,prompt4

syscall

jr $ra

***search\_element:***

li $v0, 4

la $a0, searching

syscall

li $v0, 4

la $a0, searching\_element

syscall

li $v0, 5

syscall

move $t0, $v0

li $t2, 0

li $t4, 2

li $t6, 4

jr $ra

***search:***

bgt $t2, $t3, return

***middle\_num:***

add $t7, $t2, $t3

div $t7, $t4

mflo $t8

mult $t8, $t6

mflo $t1

lw $t5, array($t1)

bgt $t5, $t0, lower

blt $t5, $t0, upper

li $s0, 1

jr $ra

***upper:***

add $t2, $t8, 1

j search

***lower:***

sub $t3, $t8, 1

j search

return:

li $s0, 0

jr $ra

***search\_result:***

beqz $s0, not\_found

***found:***

move $a0, $t0

li $v0, 1

syscall

la $a0, f

li $v0, 4

syscall

div $t1, $t6

mflo $a0

li $v0, 1

syscall

la $a0, cl\_bracket

li $v0, 4

syscall

jr $ra

***not\_found:***

move $a0, $t0

li $v0, 1

syscall

li $v0, 4

la $a0, nf

syscall

jr $ra

again:

la $a0, again\_search

li $v0, 4

syscall

li $v0,5

syscall

move $t0,$v0

li $t1,1

beq $t0,$t1,search\_element

***Exit:***

la $a0, terminate

li $v0, 4

syscall

la $a0, end

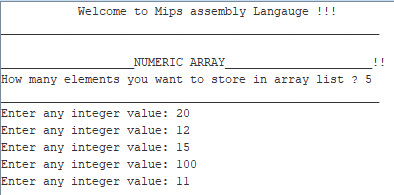
li $v0, 4

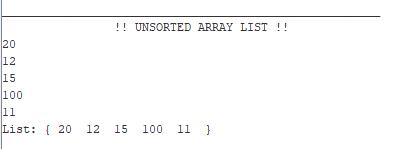
syscall

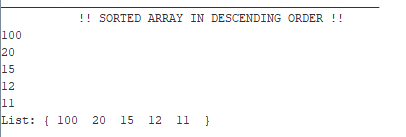
li $v0, 10

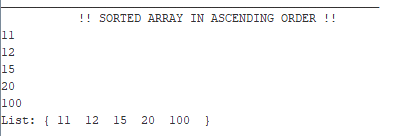
syscall

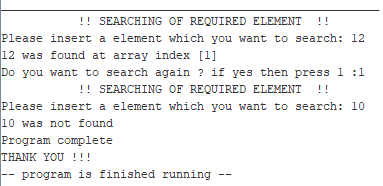
# *Output:*



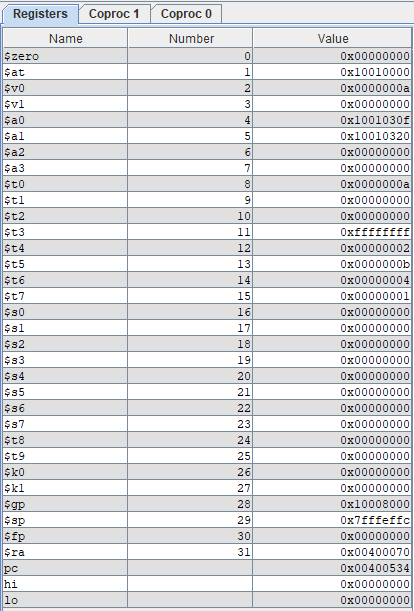








# *Register value:*



# *Conclusion:*

MIPS (Microprocessor without Interlocked Pipelined Stages) is a reduced instruction set computer (RISC) instruction set architecture. By using MIPs, we deign a program which will assist our user to store different elements in an array which he will be able to search in that particular array. We used array which is identical from MIPs assembly. The program will use bubble sort technique to print the provided list of elements in ascending and descending order. Further users can search element in respective list using binary search. The desired element which a user wants to search will be printed other than it will ask user that your input element is not available at any index of list. This process will be halted when user does not want to continue further. We use appropriate algorithms which will lead our program to be executed and implemented as per our expectations.