“Min & Max Number using Recursion”

Microprocessor & Assembly Language

Project

SUBMITTED BY :	Syed Ali Raza Naqvi
STUDENT ID :	SP19-BSCS-0100
SECTION :	AM
SUBMITTED TO:	Sir Muhammad Adil Rao

INRODUCTION

MIPS assembly language simply refers to the assembly language of the MIPS processor. The term MIPS is an acronym which stands for Microprocessor without Interlocked Pipeline Stages, and it is a reduced-instruction set architecture which was developed by an organization called MIPS Technologies.

Project Working:-

In this program we are taking the input of Array Size and then values from user and stored in the Array and then tell the user **Minimum Number** and **Maximum Number** using Recursion in the Array by comparing each and every Number.

This Project covers following topics**:**

1) **Arrays**
2) **Function**
3) **Recursion**
4) **Loop**

Language Used:-

- **Mips**

“Min & Max Number in Array”

Source code:

```
.data

ArraySizeMsg:          .asciiz "Insert the Array Size : "

InputArray:            .asciiz "Insert the Array Elements (one per line)\n"

MinMsg:                .asciiz "The Array Min = : "

MaxMsg:                .asciiz "The Array Max = : "

Nextline:              .asciiz "\n"

ThankxMsg:             .asciiz "\t\t<----------Thank You---------->"


.text

.globl main

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

main:

la $a0, ArraySizeMsg       # Print the array sixa msg

li $v0, 4

syscall


li $v0, 5             # Get the array size(n) and and put it in$v0

syscall


move $s2, $v0         # $s2=n
move $t7,$s2          #t7=n

sll $s0, $v0,2        # $s0=n*4

sub $sp, $sp, $s0     # This instruction creates a stack

                    # form large enough to contain the array

la $a0, InputArray

li $v0, 4             # ask for input

syscall


move $s1, $0           # i=0

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

for_get: bge

$s1, $s2, exit_get     # if i>=n go to exit_for_get

sll $t0, $s1, 2        # $t0=i*4

add                    $t1, $t0, $sp      # $t1=$sp+i*4

li$v0, 5               # Get one element of the array

syscall

sw $v0, 0($t1)          #The element is stored  at the address $t1


la $a0, Nextline        # print next line

li $v0, 4

syscall

addi $s1, $s1, 1            # i=i+1

j for_get
```

```
exit_get:

          move $a0, $sp          # $a0=base address af the array

          move $a1, $s2           #$a1=size of the array

jal isort                          # isort(a,n)

                                   # In this moment the array has been

                                    # sorted and is in the stack frame


la $a0, MinMsg                     # Print of str3 Array min

li $v0, 4

syscall

 move $s1, $zero                   # i=0

mul $t0, $s1, 4                    # $t0=i*4

add $t1, $sp, $t0                  # $t1=address of a[i]

lw $a0, 0($t1)

li $v0, 1                          # print of the element a[i]

syscall

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

for_print_max:

                bge $s1, $s2, exit_print        # if i>=n go to exit_print

sll $t0, $s1, 2                    # $t0=i*4

add $t1, $sp, $t0                  # $t1=address of a[i]

addi $s1, $s1, 1                   # i=i+1

j for_print_max
```

```
               #----------------------------------------------------------------------
exit_print:

la $a0, Nextline            # Print of line space
li $v0, 4
syscall
la $a0, MaxMsg          # Print of Array mAX
li $v0, 4
syscall
lw $a0, 0($t1)
li $v0, 1                      # print of the element a[i]
syscall
 add $sp, $sp, $s0      # elimination of the stack frame
 la $a0, Nextline        # Print of line space
li $v0, 4
syscall
la $a0,ThankxMsg     # Print the Thnakx msg
li $v0, 4
syscall

li $v0, 10                     # EXIT
syscall
                 #----------------------------------------------------------------------
```

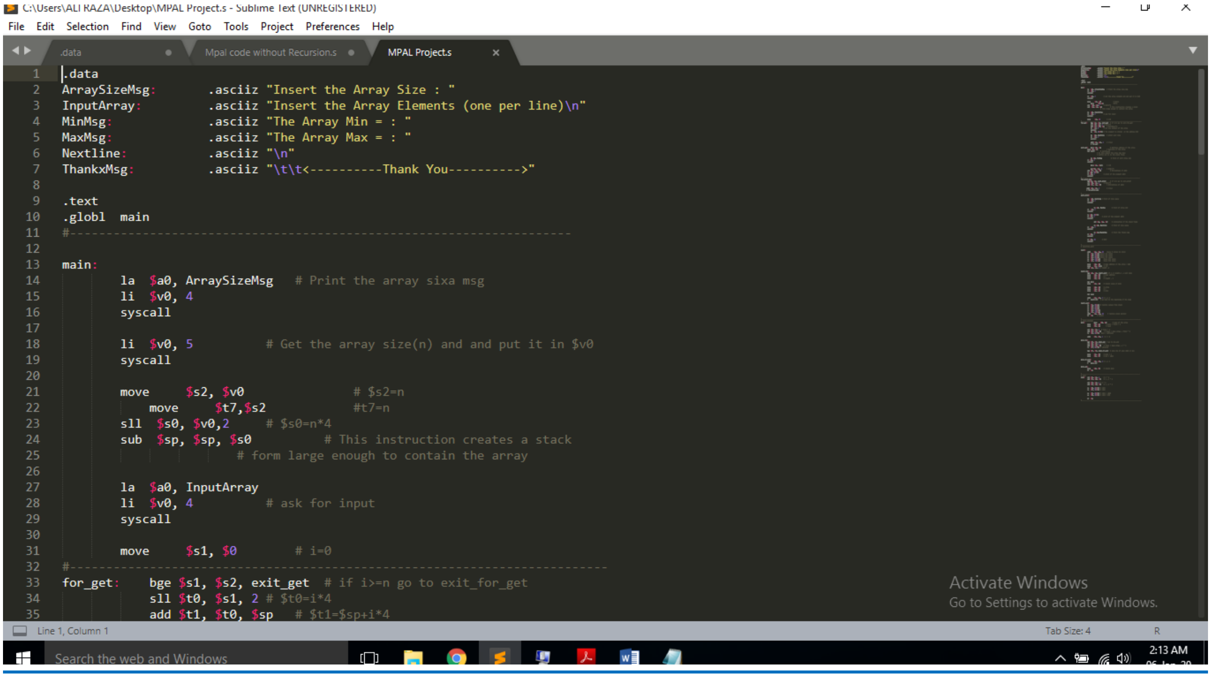
```
# selection_sort

isort:

addi$sp, $sp,-20        #save 5 values on stack

sw $ra, 0($sp)          #save 1st value

sw $s0, 4($sp)          #save 2nd value

sw $s1, 8($sp)          #save 3rd value

sw $s2, 12($sp)         #save 4th value

sw $s3, 16($sp)         #save 5th value


move  $s0, $a0          # base address of the array = $s0

move $s1, $zero         # i=0

sub $s2, $a1, 1         # lenght -1


isort_for:

bge $s1, $s2, isort_exit      # if i >= length-1 -> exit loop

move $a0, $s0           # base address

move $a1, $s1           # i

move $a2, $s2           # length – 1

jal mini

move $s3, $v0           # return value of mini

move $a0, $s0           # array

move $a1, $s1           # i

move $a2, $s3           # mini
```

```
jal swap
addi $s1, $s1, 1        # i += 1
j isort_for             # go back to the beginning of the loop

isort_exit:
lw $ra, 0($sp)          # restore values from stack
lw $s0, 4($sp)
lw $s1, 8($sp)
lw $s2, 12($sp)
lw $s3, 16($sp)
addi $sp, $sp, 20      # restore stack pointer
jr $ra                  # return

        #----------------------------------------------------------------
# index_minimum
mini:
move $t0, $a0           # base of the array
move $t1, $a1           # mini = first = i
move $t2, $a2           # last
sll $t3, $t1, 2         # first * 4
add $t3, $t3, $t0       # index = base array + first * 4
lw $t4, 0($t3)          # min = v[first]
addi $t5, $t1, 1        # i = 0
```

```
mini_for:

        bgt $t5, $t2, mini_end     # go to min_end

        sll $t6, $t5, 2            # i * 4

        add $t6, $t6, $t0      # index = base array + i * 4

        lw $t7, 0($t6)               # v[index]


        bge $t7, $t4, mini_if_exit     # skip the if when v[i] >= min

        move $t1, $t5                # mini = i

        move $t4, $t7              # min = v[i]


mini_if_exit:

addi $t5, $t5, 1                # i += 1

j mini_for


mini_end:

move  $v0, $t1                  # return mini

jr $ra

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

# swap

            swap: sll $t1, $a1, 2    # i * 4

            add $t1, $a0, $t1       # v + i * 4

            sll $t2, $a2, 2           # j * 4

           add $t2, $a0, $t2        # v + j * 4
```

```
          lw $t0, 0($t1)          # v[i]

          lw $t3, 0($t2)          # v[j]

          sw $t3, 0($t1)          # v[i] = v[j]

          sw $t0, 0($t2)          # v[j] = $t0

jr $ra

      #-------------------------------------------------------------------------
```

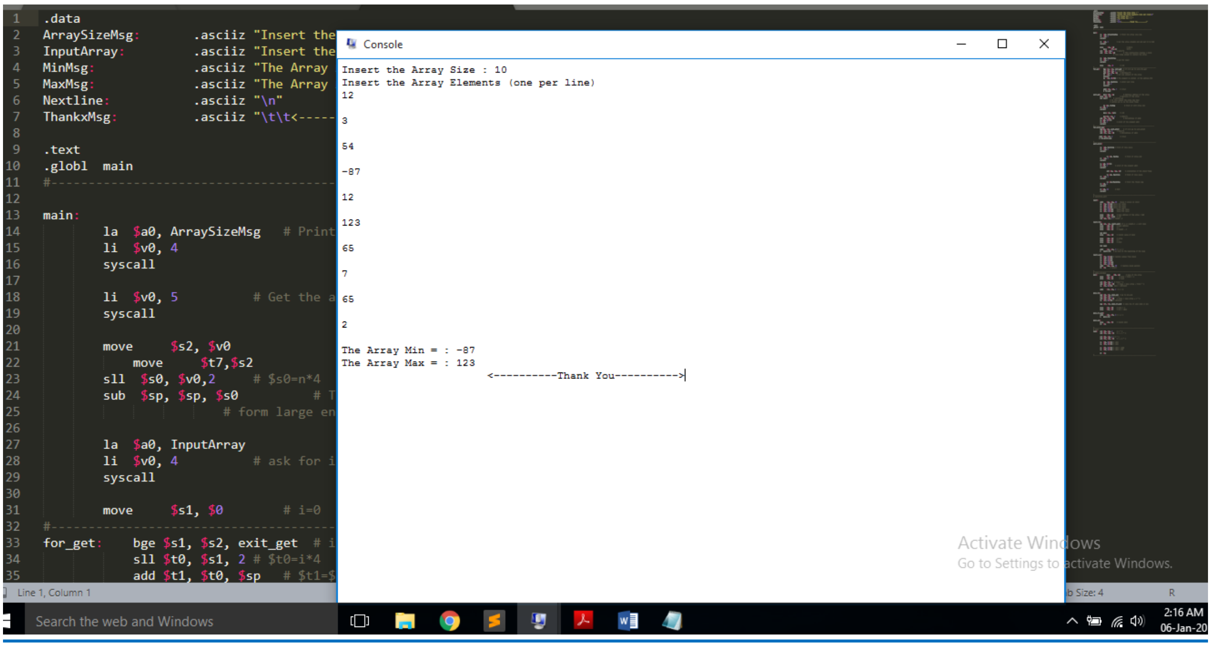
“Code Screenshot”



C:\Users\ALI RAZA\Desktop\MPAL Project.s - Sublime Text (UNREGISTERED)

```
.data
ArraySizeMsg:        .asciiz "Insert the Array Size : "
InputArray:          .asciiz "Insert the Array Elements (one per line)\n"
MinMsg:              .asciiz "The Array Min = : "
MaxMsg:              .asciiz "The Array Max = : "
Nextline:            .asciiz "\n"
ThankxMsg:           .asciiz "\t\t<---------Thank You--------->"

.text
.globl  main
#---------------------------------------------------------------------

main:
        la  $a0, ArraySizeMsg   # Print the array sixa msg
        li  $v0, 4
        syscall

        li  $v0, 5          # Get the array size(n) and and put it in $v0
        syscall

        move     $s2, $v0               # $s2=n
            move     $t7,$s2            #t7=n
        sll  $s0, $v0,2     # $s0=n*4
        sub  $sp, $sp, $s0          # This instruction creates a stack
                        # form large enough to contain the array

        la  $a0, InputArray
        li  $v0, 4          # ask for input
        syscall

        move     $s1, $0        # i=0
#--------------------------------------------------------------------------
for_get:    bge $s1, $s2, exit_get  # if i>=n go to exit_for_get
            sll $t0, $s1, 2 # $t0=i*4
            add $t1, $t0, $sp   # $t1=$sp+i*4
```

“Result Screenshot”



Console

```
Insert the Array Size : 10
Insert the Array Elements (one per line)
12

3

54

-87

12

123

65

7

65

2

The Array Min = : -87
The Array Max = : 123
                <----------Thank You---------->
```