

Amrita Vishwa Vidyapeetham
Amrita School of Engineering, Coimbatore
Department of Computer Science and Engineering
Topic: MIPS ALP using QtSPIM

Sub Code: 19CS211

Sub Title: COA

Roll No: CB.EN.U4CSE19453

Name: R.ABHINAV

Lab Evaluation No: MIPS

Date: 03-05-2021

1. Write a MIPS ALP to find the minimum and maximum elements in an array and also find their position (5 Marks). Size and elements in the array can be decided by you.

Code :

```
.text
.globl main

main:
    la $a0, ask           #ask user for integer thats going
to be the size of the array
    li $v0, 4
    syscall

    li $v0, 5             #store that int
    syscall
    move $t1,$v0          #size of my array stored in $t1

    la $t0, array         #load address of our array
    li $t2, 0             #counter = 0
    lw $t3, ($t0)         #initialize min = array[0]
    lw $t4, ($t0)         #initialize max = array[0]

while:
    la $a0, intask        #ask user for integer
    li $v0, 4
    syscall

    li $v0, 5             #store that int
    syscall
    sw $v0, ($t0)         #store that int in the array
```

```

end:      add $t0, $t0, 4           #increment the array to the
next index
        add $t2, $t2, 1           #increment the counter by 1
        blt $t2, $t1,while        #branch to while if counter < size
of array

endw:
        la $a0,display            # Display "Array is: "
        li $v0,4
        syscall

        li $t0, 0                 # initilize array index value back
to 0
        li $t2, 0                 # initial size counter back to
zero
        la $t0, array             # load address of array back into
$t0

        sprint:
        lw $t6,($t0)              #load word into temp $t2
        move $a0, $t6             #store it to a safer place
        li $v0, 1                 #print it out
        syscall

        la $a0,space              # Display " "
        li $v0,4
        syscall

        add $t0, $t0, 4           #increment the array to the next
index
        add $t2, $t2, 1           #increment the counter by 1

        blt $t2, $t1,sprint       #branch to while if counter < size
of array

        li $t2, 0                 # initial size counter back to
zero
        la $t0, array             # load address of array back into
$t0
        add $t0, $t0, 4           #increment the array to the next
index
        add $t2, $t2, 1           #increment the counter by 1

        loop:  lw $t8,($t0)        # t8 = next element in array
        bge $t8, $t3, notMin      #if array element is >= min goto
notMin
        move $t3,$t8              #min = array[i]
        j notMax

```

```

    notMin: ble $t8,$t4, notMax          #if array element is <=
max goto notMax
    move $t4,$t8                        #max = array[i]

notMax:    add $t2,$t2,1                #incr counter
    add $t0,$t0, 4                      #go up in index
    blt $t2, $t1,loop                  #if counter < size, go to loop

eprint:
    la $a0,nextline                    # Display "\n"
    li $v0,4
    syscall

    la $a0,min                          # Display "min number is "
    li $v0,4
    syscall

    move $a0, $t3                      #displays min number in array
    li $v0,1
    syscall

    la $a0,nextline                    # Display "\n"
    li $v0,4
    syscall

    la $a0,max                          # Display "max number is "
    li $v0,4
    syscall

    move $a0, $t4                      #displays max number in array
    li $v0,1
    syscall

    li $v0,10                          # End Of Program
    syscall

.data
array: .space 100
ask: .asciiz "Size of the Array: "
intask: .asciiz "Enter an Integer: "
min: .asciiz "The minimum number is: "
max: .asciiz "The maximum number is: "
display: .asciiz "Array: "
space: .asciiz " "
newline: .asciiz "\n"

```

Data Segment:

FP Regs	Int Regs [16]	Data	Text
Int Regs [16]		Data	
PC	= 0	User data segment [10000000]..[10040000]	
EPC	= 0	[10000000]..[10010063] 00000000	
Cause	= 0	[10010064] 657a6953 20666f20 20656874	
BadVAddr	= 0	[10010070] 61727241 20203a79 746e4500 61207265	
Status	= 3000fff10	[10010080] 6e49206e 65676574 00203a72 20656854	
		[10010090] 69e6e96d 206d756d 626d756e 69207265	
HI	= 0	[100100a0] 00203a73 20656854 6978616d 206d756d	
LO	= 0	[100100b0] 626d756e 69207265 00203a73 61727241	
		[100100c0] 00203a79 000a0020 00000000 00000000	
		[100100d0]..[1003ffff] 00000000	
R0 [r0]	= 0	User Stack [7ffff724]..[80000000]	
R1 [at]	= 0	[7ffff724] 00000003 7ffff806 7ffff800	
R2 [v0]	= 0	[7ffff730] 7ffff7ec 00000000 7fffffe1 7fffffb0	
R3 [v1]	= 0	[7ffff740] 7ffff7f7 7fffff43 7fffff12 7fffff00	
R4 [a0] = 3		[7ffff750] 7ffff7dc 7ffff8b5 7ffff8b3 7ffff852	
R5 [a1] = 7ffff728		[7ffff760] 7ffff82a 7ffff81d 7ffff8d4 7ffff8d2	
R6 [a2] = 7ffff738		[7ffff770] 7ffff89e 7ffff8b8 7ffff874 7ffff849	
R7 [a3] = 0		[7ffff780] 7ffff83b 7ffff827 7ffff8e9 7ffff8cc	
R8 [t0] = 0		[7ffff790] 7ffff8a2 7ffff870 7ffff858 7ffff83d	
R9 [t1] = 0		[7ffff7a0] 7ffff8af 7ffff9f6 7ffff9d8 7ffff96d	
R10 [t2] = 0		[7ffff7b0] 7ffff956 7ffff942 7ffff933 7ffff91d	
R11 [t3] = 0		[7ffff7c0] 7ffff8f3 7ffff8ca 7ffff8ba 7ffff89b	
R12 [t4] = 0		[7ffff7d0] 7ffff882 7ffff85d 7ffff823 7ffff811	
R13 [t5] = 0		[7ffff7e0] 00000000 00000000 00000000 36387828	
R14 [t6] = 0		[7ffff7f0] 74512f29 6d697053 2e31712f 006d7361	
R15 [t7] = 0		[7ffff800] 656c6946 3a430073 6f72502f 6d617267	
R16 [s0] = 0		[7ffff810] 6e697700 3d726964 575c3a43 4f444e49	
R17 [s1] = 0		[7ffff820] 56005357 5f584f42 5f49534d 54534e49	
R18 [s2] = 0		[7ffff830] 5f4c4c41 48544150 5c3a433d 676f7250	
R19 [s3] = 0		[7ffff840] 206d6172 656c6946 724f5c73 656c6361	
R20 [s4] = 0		[7ffff850] 7269565c 6c617574 5c786f42 45535500	
R21 [s5] = 0			
...	...		
Memory and registers cleared			
SPIN Version 9.1.21 of January 17, 2020 Copyright 1990-2017 by James Larus. All Rights Reserved. SPIN is distributed under a BSD license. See the file README for a full copyright notice. QtSPIN is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.			

Text Segment :

FP Regs	Int Regs [16]	Data	Text
Int Regs [16]		Text	
PC	= 0	User Text Segment [00400000]..[00440000]	
EPC	= 0	[00400000] 8fa40000 lw \$4, 0(\$29) ; 183: lw \$a0 0(\$sp) # argc	
Cause	= 0	[00400004] 27a50004 addiu \$5, \$29, 4 ; 184: addiu \$a1 \$sp 4 # argv	
BadVAddr	= 0	[00400008] 24a60004 addiu \$6, \$5, 4 ; 185: addiu \$a2 \$a1 4 # envp	
Status	= 3000fff10	[0040000c] 00041080 sll \$2, \$4, 2 ; 186: sll \$v0 \$a0 2	
		[00400010] 00c23021 addu \$6, \$6, \$2 ; 187: addu \$a2 \$a2 \$v0	
HI	= 0	[00400014] 0c100009 jal 0x00400024 [main] ; 188: jal main	
LO	= 0	[00400018] 00000000 nop ; 189: nop	
		[0040001c] 3402000a ori \$2, \$0, 10 ; 191: li \$v0 10	
R0 [r0] = 0		[00400020] 0000000c syscall ; 192: syscall # syscall 10 (exit)	
R1 [at] = 0		[00400024] 3c011001 lui \$1, 4097 [ask] ; 5: la \$a0, ask #ask user for integer thats going to be the size of the array	
R2 [v0] = 0		[00400028] 34240064 ori \$4, \$1, 100 [ask]	
R3 [v1] = 0		[0040002c] 34020004 ori \$2, \$0, 4 ; 6: li \$v0, 4	
R4 [a0] = 3		[00400030] 0000000c syscall ; 7: syscall	
R5 [a1] = 7ffff728		[00400034] 34020005 ori \$2, \$0, 5 ; 9: li \$v0, 5 #store that int	
R6 [a2] = 7ffff738		[00400038] 0000000c syscall ; 10: syscall	
R7 [a3] = 0		[0040003c] 00024821 addu \$9, \$0, \$2 ; 11: move \$t1,\$v0 #size of my array stored in \$t1	
R8 [t0] = 0		[00400040] 3c011001 lui \$1, 4097 [array] ; 13: la \$t0, array #load address of our array	
R9 [t1] = 0		[00400044] 34280000 ori \$8, \$1, 0 [array]	
R10 [t2] = 0		[00400048] 340a0000 ori \$10, \$0, 0 ; 14: li \$t2, 0 #counter = 0	
R11 [t3] = 0		[0040004c] 8d0b0000 lw \$11, 0(\$8) ; 15: lw \$t3,(\$t0) #initialize min = array[0]	
R12 [t4] = 0		[00400050] 8d0c0000 lw \$12, 0(\$8) ; 16: lw \$t4,(\$t0) #initialize max = array[0]	
R13 [t5] = 0		[00400054] 3c011001 lui \$1, 4097 [intask] ; 20: la \$a0, intask #ask user for integer	
R14 [t6] = 0		[00400058] 34240079 ori \$4, \$1, 121 [intask]	
R15 [t7] = 0		[0040005c] 34020004 ori \$2, \$0, 4 ; 21: li \$v0, 4	
R16 [s0] = 0		[00400060] 0000000c syscall ; 22: syscall	
R17 [s1] = 0		[00400064] 34020005 ori \$2, \$0, 5 ; 24: li \$v0, 5 #store that int	
R18 [s2] = 0		[00400068] 0000000c syscall ; 25: syscall	
R19 [s3] = 0		[0040006c] ad020000 sw \$2, 0(\$8) ; 26: sw \$v0, (\$t0) #store that int in the array	
R20 [s4] = 0		[00400070] 21080004 addi \$8, \$8, 4 ; 29: add \$t0, \$t0, 4 #increment the array to the next index	
R21 [s5] = 0		[00400074] 214a0001 addi \$10, \$10, 1 ; 30: add \$t2, \$t2, 1 #increment the counter by 1	
..	..		
Memory and registers cleared			
SPIN Version 9.1.21 of January 17, 2020			
Copyright 1990-2017 by James Larus.			
All Rights Reserved.			
SPIN is distributed under a BSD license.			
See the file README for a full copyright notice.			
QtSPIN is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.			

Output:

```
Console
Size of the Array: 6
Enter an Integer: 9
Enter an Integer: 12
Enter an Integer: 5
Enter an Integer: 0
Enter an Integer: 53
Enter an Integer: 49
Array: 9 12 5 0 53 49
The minimum number is: 0
The maximum number is: 53
```

2. Write a MIPS ALP to find the average of various elements in a array and append the average value at the end of the array. Size and elements in the array can be decided by you.

Code :

```
.data
array: .word 10, 2, 9, 14, 19, 25
length: .word 6
sum: .word 0
average: .word 0
avgString: .asciiz "Average: "
arrayString: .asciiz "\nArray: "
space: .asciiz " "

.text

main:
    la $t0, array #Base address
    li $t1, 0 #i=0
    lw $t2, length #t2 = n
    li $t3, 0 #sum = 0

    sumLoop:
        lw $t4, ($t0) #t4 = arr[i]
        add $t3, $t3, $t4 #sum += arr[i]

        add $t1, $t1, 1 #i++
```

```

        add $t0, $t0, 4 #Increment array address

        bne $t1, $t2, sumLoop #i != n continue

sw $t3, sum

#Average
div $t5, $t3, $t2 #t5 = sum/n
sw $t5, average

#Print string system call
la $a0, avgString
li $v0, 4
syscall

#Print integer system call
li $v0, 1
add $a0, $t5, 0
syscall

#Add integer to array
sw $t5, ($t0)
add $t2, $t2, 1 #n+=1

#Print string system call
la $a0, arrayString
li $v0, 4
syscall

la $t0, array #baseaddress
li $t1, 0 #i=0
displayLoop:
    #Print integer system call
    li $v0, 1

    lw $t4, ($t0)
    add $a0, $t4, 0
    syscall

    #Print space to screen
    la $a0, space
    li $v0, 4
    syscall

    add $t1, $t1, 1 #i++
    add $t0, $t0, 4 #Increment array address
    bne $t1, $t2, displayLoop

#Exit program
li $v0, 10
syscall

```

Data Segment:

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [16] Data Text

Int Regs [16]

PC = 0
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0

R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 0
R3 [v1] = 0
R4 [a0] = 3
R5 [a1] = 7ffff728
R6 [a2] = 7ffff738
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0

User data segment [10000000]..[10040000]
[10000000]..[1000ffff] 00000000
[10010000] 0000000a 00000002 00000009 0000000e
[10010010] 00000013 00000019 00000003 00000000
[10010020] 00000000 72657641 3a656761 410a0020 Average : . . A
[10010030] 79617272 2000203a 00000000 00000000 r r a y :
[10010040]..[1003ffff] 00000000

User Stack [7ffff724]..[80000000]
[7ffff724] 00000003 7ffff806 7ffff800
[7ffff730] 7ffff7e7 00000000 7fffffe1 7fffffb0
[7ffff740] 7ffff7f7 7fffff43 7fffff12 7fffff00 C
[7ffff750] 7ffffedc 7ffffeb5 7ffffe83 7ffffe52 R
[7ffff760] 7ffffe2a 7fffffd1 7ffffdfd 7ffffdd2 *
[7ffff770] 7ffffd9e 7ffffdbb 7ffffd74 7ffffd49 t I
[7ffff780] 7ffffd3b 7ffffb27 7ffffae9 7ffffacc ;
[7ffff790] 7ffffa82 7ffffa70 7ffffa58 7ffffa3d p X =
[7ffff7a0] 7ffffa1f 7ffff9f6 7ffff9d8 7ffff96d m
[7ffff7b0] 7ffff956 7ffff942 7ffff933 7ffff91d V B 3
[7ffff7c0] 7ffff8f3 7ffff8ca 7ffff8ba 7ffff89b
[7ffff7d0] 7ffff882 7ffff85d 7ffff823 7ffff811 l #
[7ffff7e0] 00000000 28000000 29363878 5374512f (x 8 6) / O t S
[7ffff7f0] 2f6d6970 61626966 63636f6e 00732e69 p i m / f i b a n o c c i . s .
[7ffff800] 656c6946 3a430073 6f72502f 6d617267 F i l e s . C : / P r o g r a m
[7ffff810] 6e697700 3d726964 575c3a43 4f444e49 . w i n d i r = C : \ W I N D O
[7ffff820] 56005357 5f584f42 5f49534d 54534e49 W S . V B O X _ M S I _ I N S T
[7ffff830] 5f4c4c41 48544150 5c3a433d 676f7250 A L L _ P A T H = C : \ P r o g
[7ffff840] 206d6172 656c6946 724f5c73 656c6361 r a m F i l e s \ O r a c l e
[7ffff850] 7269565c 6c617574 5c786f42 45535500 \ v i r t u a l B o x \ . U S E
[7ffff860] 4f525052 454c4946 5c3a433d 72657355 R P O F I L E = C : \ U s e r
[7ffff870] 41525c73 4c4c4556 42412041 414e4948 s \ R A V E L L A A B H I N A
[7ffff880] 53550056 414e5245 523d454d 4c455641 V . U S E R N A M E = R A V E L

Memory and registers cleared

SPIM Version 9.1.21 of January 17, 2020
Copyright 1990-2017 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Text Segment :

FP Regs Int Regs [16] Data Text

Int Regs [16]

PC = 0
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0

R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 0
R3 [v1] = 0
R4 [a0] = 0
R5 [a1] = 0
R6 [a2] = 0
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0

User Text Segment [00400000]..[00440000]
[00400000] 8fa40000 lw \$4, 0(\$29) ; 183: lw \$a0 0(\$sp) # arg0
[00400004] 27a50004 addiu \$5, \$29, 4 ; 184: addiu \$a1 \$sp 4 # argv
[00400008] 24a60004 addiu \$6, \$5, 4 ; 185: addiu \$a2 \$a1 4 # envp
[0040000c] 00041080 sll \$2, \$4, 2 ; 186: sll \$v0 \$a0 2
[00400010] 00c23021 addu \$6, \$6, \$2 ; 187: addu \$a2 \$a2 \$v0
[00400014] 0c100009 jal 0x00400024 [main] ; 188: jal main
[00400018] 00000000 nop ; 189: nop
[0040001c] 3402000a ori \$2, \$0, 10 ; 191: li \$v0 10
[00400020] 0000000c syscall ; 192: syscall # syscall 10 (exit)
[00400024] 3c081001 lui \$8, 4097 [array] ; 13: la \$t0, array #Base address
[00400028] 34090000 ori \$9, \$0, 0 ; 14: li \$t1, 0 #i=0
[0040002c] 3c011001 lui \$1, 4097 ; 15: lw \$t2, length #t2 = n
[00400030] 8c2a0018 lw \$10, 24(\$1)
[00400034] 340b0000 ori \$11, \$0, 0 ; 16: li \$t3, 0 #sum = 0
[00400038] 8d0c0000 lw \$12, 0(\$8) ; 19: lw \$t4, (\$t0) #t4 = arr[i]
[0040003c] 016c5820 add \$11, \$11, \$12 ; 20: add \$t3, \$t3, \$t4 #sum += arr[i]
[00400040] 21290001 addi \$9, \$9, 1 ; 22: add \$t1, \$t1, 1 #i++
[00400044] 21080004 addi \$8, \$8, 4 ; 23: add \$t0, \$t0, 4 #Increment array address
[00400048] 152afffc bne \$9, \$10, -16 [sumLoop-0x00400048]
[0040004c] 3c011001 lui \$1, 4097 ; 27: sw \$t3, sum
[00400050] ac2b001c sw \$11, 28(\$1)
[00400054] 15400002 bne \$10, \$0, 8 ; 30: div \$t5, \$t3, \$t2 #t5 = sum/n
[00400058] 0000000d break
[0040005c] 016a001a div \$11, \$10
[00400060] 00006812 mflo \$13
[00400064] 3c011001 lui \$1, 4097 ; 31: sw \$t5, average
[00400068] ac2d0020 sw \$13, 32(\$1)
[0040006c] 3c011001 lui \$1, 4097 [avgString] ; 34: la \$a0, avgString
[00400070] 34240024 ori \$4, \$1, 36 [avgString]
[00400074] 34020004 ori \$2, \$0, 4 ; 35: li \$v0, 4

Memory and registers cleared

SPIM Version 9.1.21 of January 17, 2020
Copyright 1990-2017 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Output:

Input Array : 10, 2, 9, 14, 19, 25



```
Console
Average: 13
Array: 10 2 9 14 19 25 13 |
```

The screenshot shows a console window with a title bar that says "Console". Inside the window, the text "Average: 13" is displayed on the first line, and "Array: 10 2 9 14 19 25 13 |" is displayed on the second line. The cursor is at the end of the second line.