

Laboratory Exercise 6

Array and Pointer

1. **Assignment 1:** Create a new project to implement procedure in Home Assignment 1. Add code the main program and initialize data for the integer list. Compile and upload to simulator. Run this program step by step, observe the process of explore each element of the integer list using indexing method.

```
Edit Execute
mips1.asm
1  .data
2  A: .word -2, 6, -1, 3, -2
3  .text
4  main:
5      la $a0,A
6      li $a1,5
7      j mspfx
8      nop
9  continue:
10 lock: j lock
11      nop
12 end_of_main:
13
14 mspfx: addi $v0,$zero,0      #initialize length in $v0 to 0
15      addi $v1,$zero,0      #initialize max sum in $v1 to 0
16      addi $t0,$zero,0      #initialize index i in $t0 to 0
17      addi $t1,$zero,0      #initialize running sum in $t1 to 0
18  loop: add $t2,$t0,$t0      #put 2i in $t2
19      add $t2,$t2,$t2        #put 4i in $t2
20      add $t3,$t2,$a0        #put 4i+A (address of A[i]) in $t3
21      lw $t4,0($t3)          #load A[i] from mem(t3) into $t4
22      add $t1,$t1,$t4        #add A[i] to running sum in $t1
23      slt $t5,$v1,$t1        #set $t5 to 1 if max sum < new sum
24      bne $t5,$zero,mdfy # if max sum is less, modify results
25      j test                 #done?
26  mdfy: addi $v0,$t0,1        #new max-sum prefix has length i+1
27      addi $v1,$t1,0          #new max sum is the running sum
28  test: addi $t0,$t0,1        #advance the index i
29      slt $t5,$t0,$a1        #set $t5 to 1 if i<n
30      bne $t5,$zero,loop     #repeat if i<n
31  done: j continue
32 mspfx_end:
```

Text Segment

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400014	0x08100005	j 0x00400014	10: lock: j lock
<input type="checkbox"/>	0x00400018	0x00000000	nop	11: nop
<input type="checkbox"/>	0x0040001c	0x20020000	addi \$2,\$0,0x00000000	14: mspfx: addi \$v0,\$zero,0 #initialize length in \$v..
<input type="checkbox"/>	0x00400020	0x20030000	addi \$3,\$0,0x00000000	15: addi \$v1,\$zero,0 #initialize max sum in \$v..
<input type="checkbox"/>	0x00400024	0x20080000	addi \$8,\$0,0x00000000	16: addi \$t0,\$zero,0 #initialize index i in \$v..
<input type="checkbox"/>	0x00400028	0x20090000	addi \$9,\$0,0x00000000	17: addi \$t1,\$zero,0 #initialize running sum ..
<input type="checkbox"/>	0x0040002c	0x01085020	add \$10,\$8,\$8	18: loop: add \$t2,\$t0,\$t0 #put 2i in \$t2
<input type="checkbox"/>	0x00400030	0x014a5020	add \$10,\$10,\$10	19: add \$t2,\$t2,\$t2 #put 4i in \$t2
<input type="checkbox"/>	0x00400034	0x01445820	add \$11,\$10,\$4	20: add \$t3,\$t2,\$a0 #put 4i+A (address of A[...
<input type="checkbox"/>	0x00400038	0x8d6c0000	lw \$12,0x00000000(\$...	21: lw \$t4,0(\$t3) #load A[i] from mem(t3) ..
<input type="checkbox"/>	0x0040003c	0x012c4820	add \$9,\$9,\$12	22: add \$t1,\$t1,\$t4 #add A[i] to running sum..
<input type="checkbox"/>	0x00400040	0x0069682a	slt \$13,\$9,\$9	23: slt \$t5,\$v1,\$t1 #set \$t5 to 1 if max sum..
<input type="checkbox"/>	0x00400044	0x15a00001	bne \$13,\$0,0x00000001	24: bne \$t5,\$zero,mdfy # if max sum is less, modi..
<input type="checkbox"/>	0x00400048	0x08100015	j 0x00400054	25: j test #done?
<input type="checkbox"/>	0x0040004c	0x21020001	addi \$2,\$8,0x00000001	26: mdfy: addi \$v0,\$t0,1 #recalculate prefix has

Labels

Label	Address
mips1.asm	
main	0x00400000
continue	0x00400014
lock	0x00400014
end of main	0x0040001c
mspfx	0x0040001c
loop	0x0040002c
mdfy	0x0040004c
test	0x00400054
done	0x00400060
mspfx_end	0x00400064
A	0x10010000

☒ Data
☒ Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0xffffffff	0x00000006	0xffffffff	0x00000003	0xffffffff	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

☒ Hexadecimal Addresses
☒ Hexadecimal Values
☐ ASCII

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x00000004
\$v1	3	0x00000006
\$a0	4	0x10010000
\$a1	5	0x00000005
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000005
\$t1	9	0x00000004
\$t2	10	0x00000010
\$t3	11	0x10010010
\$t4	12	0xffffffff
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7fffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400014
hi		0x00000000
lo		0x00000000

- Assignment 2:** Create a new project to implement procedure in Home Assignment 2. Add code the main program and initialize data for the integer list. Compile and upload to simulator. Run this program step by step, observe the process of explore each element of the integer list using pointer updating method. Write a procedure to print array after each round.

mips1.asm

mips2.asm

```
1  .data
2  A: .word 7, -2, 5, 1, 5,6,7,3,6,8,8,59,5
3  Aend: .word
4  Message: .asciiz "\n"
5  Message1: .asciiz " "
6  .text
7  main:   la $a0,A # $a0 = Address(A[0])
8         la $s5,A # $s5 = Address(A[0])
9         la $a1,Aend
10        addi $a1,$a1,-4 # $a1 = Address(A[n-1])
11        j sort #sort
12  after_sort:
13        li $v0, 10 #exit
14        syscall
15  end_main:
16
17  sort:
18        slt $s4,$a0,$a1
19        beq $a0,$a1,done #single element list is sorted
20        addi $a0,$a0,4
21        addi $s6,$a0,0
22        j loop1
23  after_loop:
24        la $t0,A
25        la $t1,Aend
26  print:
27        lw $at,0($t0)
28        li $v0,1 #service 1 is print integer
29        move $a0,$at
30        syscall #execute
31        li $v0, 4
32        la $a0, Message1
```

```

33     syscall
34     addi $t0,$t0,4
35     slt $t3,$t0,$t1
36     bne $t3,$zero,print
37     li $v0,4
38     la $a0,Message
39     syscall
40     addi $a0,$s6,0  #$s6 = Address(A[i])
41     j sort
42 done: j after_sort
43 loop1:
44     addi $v0,$a0,0  #init pointer to current element
45     lw $v1,0($v0)  #init value to current value
46     addi $t0,$a0,0  #init previous pointer to previous element
47 loop:
48     beq $t0,$s5,ret  #if next=last, return
49     addi $t0,$t0,-4  #advance to previous element
50     lw $t1,0($t0)  #load previous element into $t1
51     addi $s0,$v0,0
52     addi $s1,$v1,0
53     addi $v0,$t0,0  #previous element is new current element
54     addi $v1,$t1,0  #previous value is new current value
55     slt $t2,$t1,$s1  $(previous)<(current) ?
56     bne $t2,$zero,loop  #if (previous)<(current), repeat
57     lw $s3,0($t0)
58     sw $s3,0($s0)
59     sw $s1,0($t0)  #swap 2 elements
60     addi $v1,$s1,0
61     j loop  #change completed; now repeat
62 ret:
63     j after loop

```

Mars Messages

Run I/O

Clear

```

-2 7 5 1 5 6 7 3 6 8 8 59 5
-2 5 7 1 5 6 7 3 6 8 8 59 5
-2 1 5 7 5 6 7 3 6 8 8 59 5
-2 1 5 5 7 6 7 3 6 8 8 59 5
-2 1 5 5 6 7 7 3 6 8 8 59 5
-2 1 5 5 6 7 7 3 6 8 8 59 5
-2 1 3 5 5 6 7 7 6 8 8 59 5
-2 1 3 5 5 6 6 7 7 8 8 59 5
-2 1 3 5 5 6 6 7 7 8 8 59 5
-2 1 3 5 5 6 6 7 7 8 8 59 5

```

Mars Messages

Run I/O

Clear

```

-2 1 3 5 5 6 6 7 7 8 8 59 5
-2 1 3 5 5 6 6 7 7 8 8 59 5
-2 1 3 5 5 6 6 7 7 8 8 59 5
-2 1 3 5 5 6 6 7 7 8 8 59 5
-2 1 3 5 5 6 6 7 7 8 8 59 5
-2 1 3 5 5 5 6 6 7 7 8 8 59
-- program is finished running --

```

3. **Assignment 3:** Write a procedure to implement ***bubble sort*** algorithm

```

1  .data
2  A: .word 7, -2, 5, 1, 5,6,7,3,6,8,8,59,5
3  Aend: .word
4  Message: .asciiz "\n"
5  Message1: .asciiz " "
6  .text
7  main:    la $a0,A           #$a0 = Address(A[0])
8          la $a1,Aend
9          addi $a1,$a1,-4     #$a1 = Address(A[n-1])
10         j sort             #sort
11  after_sort:
12         li $v0, 10          #exit
13         syscall
14  end_main:
15
16  sort:
17         slt $s4,$a0,$a1
18         beq $s4,$zero,done   #single element list is sorted
19         j loop1
20  after_loop:
21         addi $a1,$a1,-4      #decrement pointer to last element
22         la $t0,A
23         la $t1,Aend
24  print:
25         lw $at,0($t0)
26         li $v0, 1            # service 1 is print integer
27         move $a0, $at
28         syscall              # execute
29         li $v0, 4
30         la $a0, Message1
31         syscall
32         addi $t0,$t0,4

```

```

33         slt $t3,$t0,$t1
34         bne $t3,$zero,print
35         li $v0, 4
36         la $a0, Message
37         syscall
38         la $a0,A           #$a0 = Address(A[0])
39         j sort             #repeat sort for smaller list
40 done:    j after_sort
41
42 loop1:
43         addi $v0,$a0,0      #init pointer to first element
44         lw $v1,0($v0)       #init value to first value
45         addi $t0,$a0,0      #init next pointer to first
46 loop:
47         beq $t0,$a1,ret     #if next=last, return
48         addi $t0,$t0,4      #advance to next element
49         lw $t1,0($t0)       #load next element into $t1
50         addi $s0,$v0,0
51         addi $s1,$v1,0
52         addi $v0,$t0,0      #next element is new current element
53         addi $v1,$t1,0      #next value is new current value
54         slt $t2,$s1,$t1     $(current)< $(next) ?
55         bne $t2,$zero,loop  #if $(current)<$(next), repeat
56         lw $s3,0($t0)       #
57         sw $s3,0($s0)        #
58         sw $s1,0($t0)        #swap 2 elements
59         addi $v1,$s1,0
60         j loop             #change completed; now repeat
61 ret:
62         j after_loop

```

Mars Messages

Run I/O

Clear

-2 5 1 5 6 7 3 6 7 8 8 5 59
-2 1 5 5 6 3 6 7 7 8 5 8 59
-2 1 5 5 3 6 6 7 7 5 8 8 59
-2 1 5 3 5 6 6 7 5 7 8 8 59
-2 1 3 5 5 6 6 5 7 7 8 8 59
-2 1 3 5 5 6 5 6 7 7 8 8 59
-2 1 3 5 5 5 6 6 7 7 8 8 59
-2 1 3 5 5 5 6 6 7 7 8 8 59
-2 1 3 5 5 5 6 6 7 7 8 8 59

Clear

- - - - -
-2 1 3 5 5 5 6 6 7 7 8 8 59
-2 1 3 5 5 5 6 6 7 7 8 8 59
-2 1 3 5 5 5 6 6 7 7 8 8 59

-- program is finished running --

4. Assignment 4: Write a procedure to implement **insertion sort** algorithm


```

1  .data
2  A: .word 7, -2, 5, 1, 5,6,7,3,6,8,8,59,5
3  Aend: .word
4  Message: .asciiz "\n"
5  Message1: .asciiz " "
6  .text
7  main:    la $a0,A           #$a0 = Address(A[0])
8          la $s5,A           #$s5 = Address(A[0])
9          la $a1,Aend
10         addi $a1,$a1,-4      #$a1 = Address(A[n-1])
11         j sort              #sort
12  after_sort:
13         li $v0, 10           #exit
14         syscall
15  end_main:
16
17  sort:
18         slt $s4,$a0,$a1
19         beq $s4,$zero,done    #single element list is sorted
20         addi $a0,$a0,4
21         addi $s6,$a0,0
22         j loop1
23  after_loop:
24         la $t0,A
25         la $t1,Aend
26  print:
27         lw $at,0($t0)
28         li $v0, 1             # service 1 is print integer
29         move $a0, $at
30         syscall               # execute
31         li $v0, 4
32         la $a0, Message1
33         syscall

```

```

34      addi $t0,$t0,4
35      slt $t3,$t0,$t1
36      bne $t3,$zero,print
37      li $v0, 4
38      la $a0, Message
39      syscall
40      addi $a0,$s6,0      #$s6 = Address(A[i])
41      j sort              #repeat sort
42 done: j after_sort
43
44 loop1:
45      addi $v0,$a0,0      #init pointer to current element
46      lw $v1,0($v0)      #init value to current value
47      addi $t0,$a0,0      #init previous pointer to previous element
48 loop:
49      beq $t0,$s5,ret      #if current=first, return
50      addi $t0,$t0,-4      #advance to previous element
51      lw $t1,0($t0)      #load previous element into $t1
52      addi $s0,$v0,0
53      addi $s1,$v1,0
54      addi $v0,$t0,0      #previous element is new current element
55      addi $v1,$t1,0      #previous value is new current value
56      slt $t2,$t1,$s1      $(previous)< $(current) ?
57      bne $t2,$zero,loop   #if $(previous)<$(current), repeat
58      lw $s3,0($t0)      #
59      sw $s3,0($s0)      #
60      sw $s1,0($t0)      #swap 2 elements
61      addi $v1,$s1,0
62      j loop              #change completed; now repeat
63 ret:
64      j after_loop

```

Mars Messages

Run I/O

Clear

-2 7 5 1 5 6 7 3 6 8 8 59 5

-2 5 7 1 5 6 7 3 6 8 8 59 5

-2 1 5 7 5 6 7 3 6 8 8 59 5

-2 1 5 5 7 6 7 3 6 8 8 59 5

-2 1 5 5 6 7 7 3 6 8 8 59 5

-2 1 5 5 6 7 7 3 6 8 8 59 5

-2 1 3 5 5 6 7 7 6 8 8 59 5

-2 1 3 5 5 6 6 7 7 8 8 59 5

Clear

-2 1 3 5 5 6 6 7 7 8 8 59 5

-2 1 3 5 5 6 6 7 7 8 8 59 5

-2 1 3 5 5 6 6 7 7 8 8 59 5

-2 1 3 5 5 5 6 6 7 7 8 8 59

-- program is finished running --