

BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH TUẦN 6

Assignment 1

- Code:

```
.data
A: .word -2, 6, -1, 3, -2

.text
main: la $a0,A
      li $a1,5
      j mspfx
      nop
continue:
lock: j lock
      nop
end_of_main:
#-----

#Procedure mspfx
# @brief find the maximum-sum prefix in a list of integers
# @param[in] a0 the base address of this list(A) need to be processed
# @param[in] a1 the number of elements in list(A)
# @param[out] v0 the length of sub-array of A in which max sum reaches.
# @param[out] v1 the max sum of a certain sub-array
#-----

#Procedure mspfx
#function: find the maximum-sum prefix in a list of integers
#the base address of this list(A) in $a0 and the number of
#elements is stored in a1
mspfx: addi $v0,$zero,0 #initialize length in $v0 to 0
      addi $v1,$zero,0 #initialize max sum in $v1 to 0
      addi $t0,$zero,0 #initialize index i in $t0 to 0
```

addi \$t1,\$zero,0 #initialize running sum in \$t1 to 0

loop: add \$t2,\$t0,\$t0 #put 2i in \$t2

add \$t2,\$t2,\$t2 #put 4i in \$t2

add \$t3,\$t2,\$a0 #put 4i+A (address of A[i]) in \$t3

lw \$t4,0(\$t3) #load A[i] from mem(t3) into \$t4

add \$t1,\$t1,\$t4 #add A[i] to running sum in \$t1

slt \$t5,\$v1,\$t1 #set \$t5 to 1 if max sum < new sum

bne \$t5,\$zero,mdfy #if max sum is less, modify results

j test #done?

mdfy: addi \$v0,\$t0,1 #new max-sum prefix has length i+1

addi \$v1,\$t1,0 #new max sum is the running sum

test: addi \$t0,\$t0,1 #advance the index i

slt \$t5,\$t0,\$a1 #set \$t5 to 1 if i<n

bne \$t5,\$zero,loop #repeat if i<n

done: j continue

mspfx_end:

- Kết quả chạy chương trình:

The screenshot displays a MIPS simulator interface. The top window shows the assembly code with the following instructions highlighted:

- 9: lock: j lock
- 10: nop
- 24: mspfx: addi \$v0,\$zero,0 #initialize length in \$v0 to 0
- 25: addi \$v1,\$zero,0 #initialize max sum in \$v1 to 0
- 26: addi \$t0,\$zero,0 #initialize index i in \$t0 to 0
- 27: addi \$t1,\$zero,0 #initialize running sum in \$t1 to 0
- 29: loop: add \$t2,\$t0,\$t0 #put 2i in \$t2
- 30: add \$t2,\$t2,\$t2 #put 4i in \$t2
- 31: add \$t3,\$t2,\$a0 #put 4i+A (address of A[i]) in \$t3
- 32: lw \$t4,0(\$t3) #load A[i] from mem(t3) into \$t4
- 33: add \$t1,\$t1,\$t4 #add A[i] to running sum in \$t1
- 34: slt \$t5,\$v1,\$t1 #set \$t5 to 1 if max sum < new sum

The bottom window shows the Data Segment with memory addresses and values. The registers window on the right shows the following values:

Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	4
\$v1	3	6
\$a0	4	268500992
\$a1	5	5
\$a2	6	0
\$a3	7	0
\$t0	8	5
\$t1	9	4
\$t2	10	16
\$t3	11	268501008
\$t4	12	-2
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$s8	24	0
\$s9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194324
hi		0
lo		0

Nhận xét:

Chương trình load địa chỉ phần tử đầu tiên của mảng vào a0, rồi sau đó chạy vòng lặp từ đầu array đến cuối và tính tổng prefix vào running sum, lấy phần lớn nhất trong các tổng đã tính vào max sum, nếu running sum hiện tại lớn hơn max sum thì sẽ nhảy đến nhãn mdify để thay thế max sum bằng running sum.

Assignment 2

- Code:

```
.data
```

```
A: .word 7, -2, 5, 1, 5,6,7,3,6,8,8,59,5
```

```
Aend: .word
```

```
.text
```

```
main: la $a0,A #$a0 = Address(A[0])
```

```
la $a1,Aend
```

```
addi $a1,$a1,-4 #$a1 = Address(A[n-1])
```

```
j sort #sort
```

```
after_sort: li $v0, 10 #exit
```

```
syscall
```

```
end_main:
```

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

```
#procedure sort (ascending selection sort using pointer)
```

```
#register usage in sort program
```

```
#$a0 pointer to the first element in unsorted part
```

```
#$a1 pointer to the last element in unsorted part
```

```
#$t0 temporary place for value of last element
```

```
#$v0 pointer to max element in unsorted part
```

```
#$v1 value of max element in unsorted part
```

```

#-----
sort: beq $a0,$a1,done #single element list is sorted
      j max #call the max procedure
after_max: lw $t0,0($a1) #load last element into $t0
           sw $t0,0($v0) #copy last element to max location
           sw $v1,0($a1) #copy max value to last element
           addi $a1,$a1,-4 #decrement pointer to last element
           j sort #repeat sort for smaller list
done: j after_sort

#-----
#Procedure max
#function: find the value and address of max element in the list
#$a0 pointer to first element
#$a1 pointer to last element
#-----

max:
      addi $v0,$a0,0 #init max pointer to first element
      lw $v1,0($v0) #init max value to first value
      addi $t0,$a0,0 #init next pointer to first
loop:
      beq $t0,$a1,ret #if next=last, return
      addi $t0,$t0,4 #advance to next element
      lw $t1,0($t0) #load next element into $t1
      slt $t2,$t1,$v1 #(next)<(max) ?
      bne $t2,$zero,loop #if (next)<(max), repeat
      addi $v0,$t0,0 #next element is new max element
      addi $v1,$t1,0 #next value is new max value

```

j loop #change completed; now repeat

ret:

j after_max

- Kết quả chạy chương trình:

The screenshot shows the Mars MIPS simulator interface. The 'Text Segment' window displays assembly code for a bubble sort algorithm. The 'Registers' window on the right shows the state of MIPS registers. The 'Data Segment' window at the bottom shows memory addresses and their corresponding values.

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001 lui \$1,4097	5: main: la \$a0,A #a0 = Address(A[0])	
	0x00400004	0x34240000 ori \$4,\$1,0		
	0x00400008	0x3c011001 lui \$1,4097	6: la \$a1,Aend	
	0x0040000c	0x34250034 ori \$5,\$1,52		
	0x00400010	0x20a5fffc addi \$5,\$5,-4	7: addi \$a1,\$a1,-4 #a1 = Address(A[n-1])	
	0x00400014	0x08100008 j 0x00400020	8: j sort #sort	
	0x00400018	0x2402000a addiu \$2,\$0,10	9: after sort: li \$v0, 10 #exit	
	0x0040001c	0x0000000c syscall	10: syscall	
	0x00400020	0x10850006 beq \$a0,\$a1,done #single element list is sorted	21: sort: beq \$a0,\$a1,done #single element list is sorted	
	0x00400024	0x08100010 j 0x00400040	22: j max #call the max procedure	
	0x00400028	0x8ca80000 lw \$8,0(\$5)	23: after max: lw \$t0,0(\$a1) #load last element into \$t0	
	0x0040002c	0xac480000 sw \$8,0(\$2)	24: sw \$t0,0(\$v0) #copy last element to max location	

Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	10
\$v1	3	1
\$a0	4	268500992
\$a1	5	268500992
\$a2	6	0
\$a3	7	0
\$t0	8	-2
\$t1	9	-2
\$t2	10	1
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$s8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194336
hi		0
lo		0

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-2	1	3	5	5	5	6	6
0x10010020	7	7	8	8	59	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

- SORT TỪ LỚN HƠN ĐẾN NHỎ HƠN

.data

A: .word 7, -2, 5, 1, 5,6,7,3,6,8,8,59,5

Aend: .word

.text

main: la \$a0,A #a0 = Address(A[0])

la \$a1,Aend

addi \$a1,\$a1,-4 #a1 = Address(A[n-1])

j sort #sort

```

after_sort: li $v0, 10 #exit
            syscall
end_main:

#-----
#procedure sort (ascending selection sort using pointer)
#register usage in sort program
#$a0 pointer to the first element in unsorted part
#$a1 pointer to the last element in unsorted part
#$t0 temporary place for value of last element
#$v0 pointer to max element in unsorted part
#$v1 value of max element in unsorted part
#-----

sort: beq $a0,$a1,done #single element list is sorted
      j max #call the max procedure
after_max: lw $t0,0($a1) #load last element into $t0
           sw $t0,0($v0) #copy last element to max location
           sw $v1,0($a1) #copy max value to last element
           addi $a1,$a1,-4 #decrement pointer to last element
           j sort #repeat sort for smaller list
done: j after_sort

#-----
#Procedure max
#function: find the value and address of max element in the list
#$a0 pointer to first element
#$a1 pointer to last element
#-----

max:

```

```

addi $v0,$a0,0 #init max pointer to first element
lw $v1,0($v0) #init max value to first value
addi $t0,$a0,0 #init next pointer to first
loop:
beq $t0,$a1,ret #if next=last, return
addi $t0,$t0,4 #advance to next element
lw $t1,0($t0) #load next element into $t1
slt $t2,$v1,$t1 #(next)<(max) ?
bne $t2,$zero,loop #if (next)<(max), repeat
addi $v0,$t0,0 #next element is new max element
addi $v1,$t1,0 #next value is new max value
j loop #change completed; now repeat
ret:
j after_max

```

Assignment 3

bubble sort

```

.data
Arr: .word 1,2,3,4,5,6,7,8,9

.text
.globl main
main:
li $t1,6 # lay so phan tu cua mang

# main_loop -- di vong qua array nhieu lan
main_loop:
subi $a1,$t1,1
blez $a1,main_done

```

```

la    $a0,Arr
li    $t2,0

jal   pass_loop

      beqz    $t2,main_done      # trong vong lap hien tai k co swap la
xong
      subi   $t1,$t1,1          # tru di so lan lap con lai
      b      main_loop

main_done:
      j      end                # end ctr

pass_loop:
      lw     $s1,0($a0)         # load phan tu dau tien vao s1
      lw     $s2,4($a0)         # load phan tu 2 vao s2
      bgt    $s1,$s2,pass_swap  # if (s1 > s2) swap

pass_next:
      addiu   $a0,$a0,4          # di chuyen toi phan tu tiep theo
      subiu   $a1,$a1,1
      bgtz    $a1,pass_loop      # kiem tra xem swap chua, neu khong se
loop
      jr     $ra

pass_swap:
      sw     $s1,4($a0)         # cho gia tri cua [i+1] vao s1
      sw     $s2,0($a0)         # cho gia tri cua [i] vao s2
      li     $t2,1              # da swap
      j      pass_next

# End the program
end:
      li     $v0,10
      syscall

```


Assignment 4

.data

A: .word

.text

\$a0 store base address of A

\$s1 store value of n

\$s3 store value of i

main:

la \$a0, A # \$a0 = Address(A[0])

add \$s2, \$a0, \$0 # \$s2 -> add(A)

addi \$s1, \$0, 5 # n = 5

add \$s3, \$0, \$0 # i = 0

generate_arr:

beq \$s1, \$s3, end_generate # i == n end

li \$v0, 5 # read integer

syscall

sw \$v0, 0(\$s2) # A[i] = integer

addi \$s2, \$s2, 4 # next to A{i+1}

addi \$s3, \$s3, 1 # i++

j generate_arr

end_generate:

addi \$s3, \$0, 1 # set i = 1

j sort

```

# $s2 store value of j
# $t0 is tmp
# $t1 is address of A[i]
# $t2 is address of A[j]
# $t3 to compare A[i] and A[j]
# $t4 is value of key
# $t5 is value of A[j]
# $s0 is address of A[j+1]
# $s4 is value of A[j+1]
# -----
after_sort:
li $v0, 10 # exit
syscall
end_main:
sort:
loop_1:
beq $s3, $s1, after_sort # if i == n end sort
addi $s2, $s3, -1 # set j = i - 1
sll $t1, $s3, 2 # $t1 = i*4
add $t1, $a0, $t1 # $t1 -> A[i]
lw $t4, 0($t1) # key = A[i]
j loop_2
after_loop_2:
addi $s3, $s3, 1 # i++
sll $s6, $s2, 2
addi $s6, $s6, 4
add $s0, $s6, $a0

```

```

sw $t4, 0($s0) # A[j+1] = key
j loop_1
loop_2:
sll $t2, $s2, 2 # $t2 = j*4
add $t2, $a0, $t2 # $t2 -> A[j]
lw $t5, 0($t2) # $t5 = A[j]
addi $s0, $t2, 4 # $s0 -> A[j+1]
lw $s4, 0($s0) # $s4 = A[j+1]
sgt $t6, $s2, $0 # j >= 0
beq $s2, $0, setequal
next:
sgt $t7, $t5, $t4 # A[j] > key
and $t8, $t6, $t7 # if(j>=n && A[j] > key)
beq $t8, $0, after_loop_2
sw $t5, 0($s0) # A[j+1] = A[j]
addi $s2, $s2, -1 # j--
j loop_2
setequal:
seq $t6, $s2, $0
j next

```