

Báo cáo Thực hành KTMT buổi 6

Họ và tên: Nguyễn Đức Phú

MSSV: 20215116

Assignment 1: Maximum-sum prefix

- Bổ sung đoạn code nhập dữ liệu:

```
.data
A: .word
message1: .ascii "Nhap so phan tu: "
message2: .ascii "Nhap lan luot cac phan tu:\n"

.text
main:
    li $v0, 4
    la $a0, message1
    syscall
    li $v0, 5
    syscall

    add $a1,$zero,$v0      # $a1 chua so phan tu

    li $v0, 4
    la $a0, message2
    syscall
    la $a0,A
    addi $t0,$zero,0

input:
    beq $t0,$a1,end_input
    add $t2,$t0,$t0
    add $t2,$t2,$t2
    add $t3,$t2,$a0      # address of A[i] in $t3
    li $v0, 5
    syscall
    sw $v0,0($t3)
    addi $t0,$t0,1
    j input

end_input:
```

- Với bộ dữ liệu đầu vào như sau:

Mars Messages
Run I/O

Nhập số phần tử: 5
 Nhập lần lượt các phần tử:
 -2
 6
 3
 9
 -27

Clear

- Kết quả cho ra tại thanh ghi \$v1 = 16 và chỉ số tại \$v0 = 4 là kết quả đúng

Text Segment					Registers		
Bkpt	Address	Code	Basic	Source	Name	Number	Value
<input type="checkbox"/>	0x0040004c	0x0000000c	syscall	26: syscall	\$zero	0	0
<input type="checkbox"/>	0x00400050	0xad620000	sw \$2,0(\$11)	27: sw \$v0,0(\$t3)	\$at	1	268500992
<input type="checkbox"/>	0x00400054	0x21080001	addi \$8,\$8,1	28: addi \$t0,\$t0,1	\$v0	2	4
<input type="checkbox"/>	0x00400058	0x0810000e	j 0x00400038	29: j input	\$v1	3	16
<input type="checkbox"/>	0x0040005c	0x0810001b	j 0x0040006c	31: j mspfx	\$a0	4	268500992
<input type="checkbox"/>	0x00400060	0x00000000	nop	32: nop	\$a1	5	5
<input type="checkbox"/>	0x00400064	0x08100019	j 0x00400064	34: lock: j lock	\$a2	6	0
<input type="checkbox"/>	0x00400068	0x00000000	nop	35: nop	\$a3	7	0
<input type="checkbox"/>	0x0040006c	0x20020000	addi \$2,\$0,0	38: mspfx: addi \$v0,\$zer..	\$t0	8	5
					\$t1	9	-11
					\$t2	10	16

Assignment 2: Selection Sort

- Code:

`.data`

A: `.word` -2,5,7,-23,45,-6,34,2

Aend: `.word`

message1: `.ascii` " "

message2: `.ascii` "\n"

`.text`

main:

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

la \$a1,Aend

la \$t7,Aend #Use in print array

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

j sort #sort

```

after_sort:
    li $v0, 10          #exit
    syscall
end_main:
sort:
    beq $a0,$a1,done    #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
print_arr:
    la $t6,A
    j show_arr
end_print:
    j sort               #repeat sort for smaller list
done:
    j after_sort
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

```

show_arr:

```
li $v0, 1
lw $a0, 0($t6)
syscall
li $v0, 4
la $a0, message1
syscall
addi $t6, $t6, 4
bne $t6, $t7, show_arr
li $v0, 4
la $a0, message2
syscall
la $a0, A
j end_print
```

- Kết quả:
- In ra array sau mỗi lần đổi chỗ:

```
-- program is finished running --
```

- Kết quả tại Data Segment:

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-23	-6	-2	2	5	7	34	45
0x10010020	655392	0	0	0	0	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

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Assignment 3: Bubble Sort

- Code:

.data

A: **.word** 6, 9, -3, 27, 20, -5, -12, 8

Aend: **.word**

message1: **.ascii**z " "

message2: **.ascii**z "\n"

.text

la \$a0, A

la \$a1, Aend

la \$t7, Aend

li \$s0, 0 # count = 0 (count la bien dem phan tu)

li \$s1, -1 # i = -1 (i trong loopi)

DemPhanTu:

beq \$a1, \$a0, Size

addi \$a1, \$a1, -4

addi \$s0, \$s0, 1

j DemPhanTu

Size:

addi \$t0, \$s0, -1 # t0 = So luong phan tu mang A - 1

loop1:

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

li \$s2, 0 # j = 0 (j trong loop2)

beq \$s1, \$t0, Exit # Neu i = size - 1 thì thoát

loop2:

sub \$t2, \$t0, \$s1 # t2 = (size - 1) - i

beq \$s2, \$t2, loop1 # Neu j = (size - 1) - i thì nhảy loop1

if_swap:

sll \$t3, \$s2, 2

add \$s3, \$a0, \$t3

lw \$v0, 0(\$s3)

addi \$s3, \$s3, 4

lw \$v1, 0(\$s3)

sle \$t4, \$v0, \$v1

```
    beq $t4, $zero, swap
    addi $s2, $s2, 1
    j loop2
```

swap:

```
    sw $v0, 0($s3)
    addi $s3, $s3, -4
    sw $v1, 0($s3)
    addi $s2, $s2, 1
```

print_arr:

```
    la $t6, A
    j show_arr
```

end_print:

```
    j loop2
```

show_arr:

```
    li $v0, 1
    lw $a0, 0($t6)
    syscall
    li $v0, 4
    la $a0, message1
    syscall
    addi $t6, $t6, 4
    bne $t6, $t7, show_arr
    li $v0, 4
    la $a0, message2
    syscall
    la $a0, A
    j end_print
```

Exit:

```
    li $v0, 10
    syscall
```

- Kết quả chạy:
 - Hiển thị array sau mỗi lần swap:

Mars MessagesRun I/O

Clear

```

6 -3 9 27 20 -5 -12 8
6 -3 9 20 27 -5 -12 8
6 -3 9 20 -5 27 -12 8
6 -3 9 20 -5 -12 27 8
6 -3 9 20 -5 -12 8 27
-3 6 9 20 -5 -12 8 27
-3 6 9 -5 20 -12 8 27
-3 6 9 -5 -12 20 8 27
-3 6 9 -5 -12 8 20 27
-3 6 -5 9 -12 8 20 27
-3 6 -5 -12 9 8 20 27
-3 6 -5 -12 8 9 20 27
-3 -5 6 -12 8 9 20 27
-3 -5 -12 6 8 9 20 27
-5 -3 -12 6 8 9 20 27
-5 -12 -3 6 8 9 20 27
-12 -5 -3 6 8 9 20 27

-- program is finished running --

```

- Kết quả tại Data Segment:

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+...	Value (+14)	Value (...)	Value (...)
0x10010000	-12	-5	-3	6	8	9	20	27
0x10010020	655392	0	0	0	0	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

←

→

0x10010000 (.data)

☒ Hexadecimal Addresses
 ☐ Hexadecimal Values

Assignment 4: Insertion Sort

- Code:

.data

A: **.word** 6,9,-3,27,20,-5,-12,8

Aend: **.word**

message1: **.ascii**z " "

message2: **.ascii**z "\n"

.text

la \$a0, A

la \$a1, Aend

la \$t7, Aend **#Use to Print Array**

li \$s0, 0 **# count = 0 (dem phan tu)**

li \$s1, 0 **# key = 0**

li \$s2, 0 **# j = 0**

li \$s3, 1 **# i = 1**

DemPhanTu:

beq \$a1, \$a0, Loop

addi \$a1, \$a1, -4

addi \$s0, \$s0, 1

j DemPhanTu

Loop:

beq \$s3, \$s0, Exit

sll \$t0, \$s3, 2

add \$s4, \$a0, \$t0

lw \$s1, 0(\$s4)

addi \$s2, \$s3, -1

While:

slt \$t1, \$s2, \$zero

sll \$t0, \$s2, 2

add \$s5, \$a0, \$t0

lw \$t3, 0(\$s5)

sle \$t4, \$t3, \$s1

add \$t1, \$t1, \$t4

bne \$t1, \$zero, loop_continue

addi \$s5, \$s5, 4


```

        sw $t3, 0($s5)
        addi $s2, $s2, -1
        j While
loop_continue:
        addi $s5, $s5, 4
        sw $s1, 0($s5)
        addi $s3, $s3, 1
print_arr:
        la $t6,A
        j show_arr
end_print:
        j Loop
show_arr:
        li $v0,1
        lw $a0,0($t6)
        syscall
        li $v0, 4
        la $a0, message1
        syscall
        addi $t6,$t6,4
        bne $t6,$t7,show_arr
        li $v0, 4
        la $a0, message2
        syscall
        la $a0,A
        j end_print
Exit:
        li $v0, 10
        syscall

```

- Kết quả chạy:
 - Hiển thị Array sau mỗi vòng:

Mars Messages

Run I/O

Clear

```

6 9 -3 27 20 -5 -12 8
-3 6 9 27 20 -5 -12 8
-3 6 9 27 20 -5 -12 8
-3 6 9 20 27 -5 -12 8
-5 -3 6 9 20 27 -12 8
-12 -5 -3 6 9 20 27 8
-12 -5 -3 6 8 9 20 27

-- program is finished running --

```

- Kết quả tại cửa sổ Data Segment:

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+...	Value (+...	Value ...	Value ...
0x10010000	-12	-5	-3	6	8	9	20	27
0x10010020	655392	0	0	0	0	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

←

→

0x10010000 (.data)

☒ Hexadecimal Addresses
 ☐ Hexadecimal Values