

Thực hành kiến trúc máy tính

Tuần 6

Bùi Quang Hưng – 20225849

Assignment 1

Bài làm

Code:

.data

A: .word 2,0,2,2,5,8,4,9

.text

main:

la \$a0,A

li \$a1,8

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

msofx:

```
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
```

msofx_end:

- Nhận xét:

- Lệnh la được qui thành 2 lệnh là lui và ori, lưu địa chỉ đầu mảng A vào \$a0
- Độ dài mảng A=8 được lưu vào \$a1
- Nhảy đến hàm msofx
- Msofx: Khởi tạo các giá trị \$v0 (độ dài của prefix), \$v1 (max sum), \$t0 (index i), \$t1 (running sum) bằng 0

- Loop:
- Tính địa chỉ của A[i] theo index và địa chỉ A[0]
- Lấy giá trị của A[i] lưu vào \$t4
- Tính running sum hiện tại = running sum từ vòng lặp trước (\$t1) + A[i] (\$t4)
- So sánh \$v1 (max sum) và \$t1 (running sum), nếu \$v1 < \$t1 thì \$t5 = 1, ngược lại \$t5=0
- Nếu \$t5 khác 0 ó \$v1 < \$t1 thì nhảy đến mdify, nếu không thì thực hiện dòng lệnh tiếp theo nhảy đến test
- Mdfy:
Lưu độ dài của prefix có tổng lớn nhất (\$v0)
- Lưu max sum (\$v1)
- Nếu \$t5 khác 0 => i<n tiếp tục vòng lặp, ngược lại thì câu lệnh tiếp theo được thực hiện, nhảy đến hàm continue => lock => kết thúc chương trình

Kết thúc chương trình max sum được lưu tại \$v1 và độ dài của prefix được lưu ở \$v0

→ Kết quả đúng với lí thuyết.

Assignment 2

Bài làm

Code:

.data

A: .word 2,0,5,8,4,9

Aend: .word

Message: .asciiz "\n"

Message2: .asciiz " "

.text

main: la \$a3,A #\$a3 = Address(A[0])

la \$a1,Aend

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

la \$s7, Aend

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 $a3,$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
addi $t9, $a3,0
j in
#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
#-----

in : #in ra day sau moi lan sap xep
beq $t9, $s7, in2
li $v0, 1
lw $a0,0($t9)
addi $t9,$t9,4
syscall
li $v0,4
la $a0,Message2
syscall
j in

```

```

in2 : # in ra dau enter

li $v0,4

la $a0,Message

syscall

j sort

max:

addi $v0,$a3,0 #init max pointer to first element

lw $v1,0($v0) #init max value to first value

addi $t0,$a3,0 #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

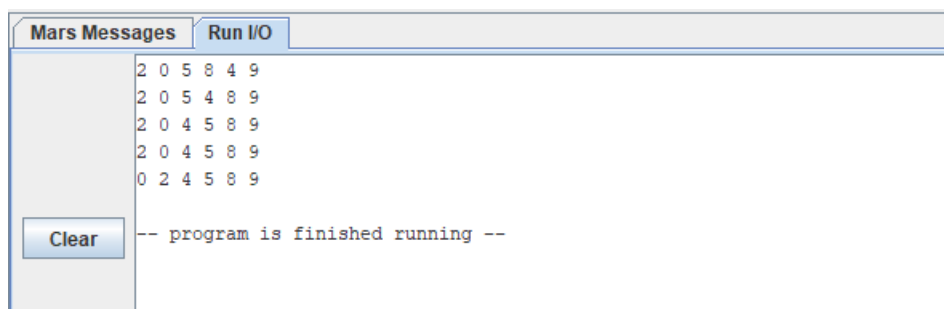
j loop #change completed; now repeat

ret:

j after_max

```

- Kết quả:



- Đầu tiên lưu địa chỉ A[0]
 - Lưu địa chỉ A[n-1] bằng cách lùi địa chỉ mảng kế tiếp mảng A (Aend) đi 4
 - Nhảy đến sort
 - Sort: Nếu \$a3 = \$a1, nhảy đến done => nhảy đến after_sort, ngược lại nhảy đến max
- Max:
 Khởi tạo địa chỉ của max
 Khởi tạo giá trị của max

- Khởi tạo con trỏ đến phần tử kế tiếp
- Loop
- Nếu địa chỉ kế tiếp là địa chỉ của A[n-1] thì nhảy đến ret
- Xét phần tử kế tiếp bằng cách tăng địa chỉ của con trỏ địa chỉ thêm 4
- Lấy giá trị của phần tử kế tiếp
- Nếu \$t1 < \$v1 thì \$t2=1, ngược lại \$t2=0
- Nếu \$t2 khác 0 => \$t1 < \$v1 ó phần tử kế tiếp không lớn hơn max, tiếp tục loop
- Nếu phần tử kế tiếp lớn hơn max, lưu địa chỉ và giá trị max mới, sau đó tiếp tục loop
- Ret: nhảy đến after_max
- After_max: đổi chỗ phần tử cuối của mảng con với max
- Lưu phần tử cuối vào \$t0
- Thế chỗ giá trị phần tử max = giá trị phần tử cuối
- Thế chỗ giá trị phần tử cuối = giá trị phần tử max
- Tiếp tục sort với mảng con nhỏ hơn
- After_sort: terminate execution

Assignment 3

Bài làm

.data

A: .word 7, 6, 5, 4, 3, 2, 1 # Mảng cần sắp xếp

Aend: .word # Địa chỉ kết thúc mảng (khởi tạo sau)

.text

la \$a0, A # \$a0 = Địa chỉ(A[0])

la \$a1, Aend

la \$a2, Aend

addi \$a1, \$a1, -4 # \$a1 = Địa chỉ(A[n-1]) = địa chỉ của phần tử cuối cùng của mảng A

move \$a3, \$a1 # \$a3 = \$a1 (để theo dõi cuối của phần đã sắp xếp)

j sort # Nhảy đến phần con sắp xếp

after_sort:

li \$v0, 10 # syscall để kết thúc chương trình

syscall

end_main:

sort:

li \$s0, 0 # Khởi tạo bộ đếm vòng lặp

move \$t7, \$a1 # \$t7 = \$a1 (con trỏ đến phần tử cuối)

loop:

beq \$t7, \$a0, end_loop # Thoát vòng lặp nếu \$t7 == \$a0 (đã đến đầu mảng)

Nạp các phần tử để so sánh

add \$t0, \$s0, \$s0 # Tính offset cho việc chỉ mục mảng

add \$t0, \$t0, \$t0

add \$t1, \$a0, \$t0 # Địa chỉ của phần tử hiện tại (\$t1 = &A[i])

lw \$t2, 0(\$t1) # Nạp A[i]

addi \$t3, \$t1, 4 # Địa chỉ của phần tử kế tiếp (\$t3 = &A[i+1])

lw \$t4, 0(\$t3) # Nạp A[i+1]

So sánh các phần tử

slt \$t5, \$t2, \$t4 # Kiểm tra nếu A[i] < A[i+1]

bne \$t5, \$zero, tang_\$s0 # Nhánh nếu A[i] < A[i+1]

Đổi chỗ các phần tử

move \$t6, \$t4

move \$t4, \$t2

move \$t2, \$t6

sw \$t2, 0(\$t1)

sw \$t4, 0(\$t3)

Tăng bộ đếm vòng lặp và con trỏ

addi \$s0, \$s0, 1

addi \$t7, \$t7, -4

j loop

tang_\$s0:

```
addi $s0, $s0, 1
```

```
addi $t7, $t7, -4
```

```
j loop
```

```
end_loop:
```

```
addi $a3, $a3, -4 # Di chuyển cuối phần đã sắp xếp về phía đầu
```

```
# Phần con in
```

```
print:
```

```
li $s1, 0      # Khởi tạo chỉ mục cho việc duyệt mảng
```

```
la $s4, A      # Nạp địa chỉ cơ sở của mảng A
```

```
print_char:
```

```
add $s2, $s1, $s1 # Tính offset cho việc chỉ mục mảng
```

```
add $s2, $s2, $s2
```

```
add $s3, $s4, $s2 # Tính địa chỉ của phần tử hiện tại
```

```
lw $s5, 0($s3)    # Nạp phần tử hiện tại
```

```
beq $s3, $a2, in_xuong_dong # Thoát vòng lặp nếu đến cuối mảng
```

```
li $v0, 1         # In số nguyên
```

```
move $a0, $s5
```

```
syscall
```

```
addi $s6, $a2, -4 # Tính địa chỉ của phần tử trước cuối
```

```
beq $s3, $s6, qua # Nếu không phải là phần tử cuối cùng, in một dấu cách
```

```
li $v0, 11
```

```
li $a0, ''
```

```
syscall
```


qua:

```
addi $s1, $s1, 1    # Tăng chỉ mục
```

```
j print_char
```

in_xuong_dong:

```
li $v0, 11          # In dấu xuống dòng
```

```
li $a0, '\n'
```

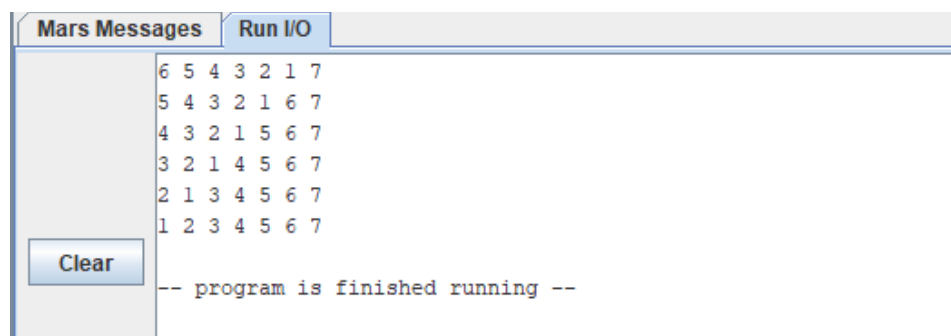
```
syscall
```

```
la $a0, A           # Nạp địa chỉ của mảng A
```

```
beq $a3, $a0, after_sort # Nếu cuối phần đã sắp xếp đến đầu mảng, thoát
```

```
j sort             # Ngược lại, tiếp tục sắp xếp
```

- Kết quả:



→ Kết quả đúng với lí thuyết

Assignment 4

Bài làm

Code:

.data

```
A: .word -1,-2,-3,-4,-5,-6    # Mảng cần sắp xếp
```

```
Aend: .word                  # Địa chỉ kết thúc mảng (khởi tạo sau)
```

.text

```
la $a0, A              # $a0 lưu địa chỉ của A[0]
```

```
la $a1, Aend
```

```
add $a2, $a0, 4        # $a2 lưu địa chỉ của A[1]
```

```
la $a3, Aend
```

```
add $a1, $a1, -4 # $a1 lưu địa chỉ của A[n-1]
```

```
j sort      # Nhảy đến phần con sắp xếp
```

```
after_sort:
```

```
li $v0, 10    # syscall để kết thúc chương trình
```

```
syscall
```

```
end_main:
```

```
sort:
```

```
li $s0, 1      # i (vị trí phần tử cần sắp xếp)
```

```
loop:
```

```
li $s1, 0      # Biến kiểm tra xem có cần chèn không?
```

```
add $t0, $s0, $s0
```

```
add $t0, $t0, $t0
```

```
add $t0, $a0, $t0 # Lấy địa chỉ của phần tử A[i]
```

```
addi $t2, $t0, -4 # Lấy địa chỉ của phần tử A[j] (j = i - 1)
```

```
move $t5, $t0    # $t5 lưu vị trí sẽ chèn (khởi tạo tại vị trí A[i])
```

```
loop1:
```

```
lw $t1, 0($t0)   # $t1 lưu giá trị của A[i]
```

```
lw $t3, 0($t2)   # $t3 lưu giá trị của A[j]
```

```
slt $t4, $t1, $t3 # Nếu A[j] < A[i]
```

```
beq $t4, $zero, tru_tiep # Thì nhảy đến j--
```

```
move $t5, $t2    # A[j] > A[j] thì vị trí chèn sẽ là $t2
```

```
li $s1, 1      # Biến kiểm tra = 1
```

```
tru_tiep:
```

```
beq $t2, $a0, chen # Nếu j-- đã đến vị trí A[0] thì bắt đầu chèn
```

```
addi $t2, $t2, -4 # j--
```

j loop1

chen:

beq \$s1, 0, print # Kiểm tra xem A[i] đã đúng vị trí chưa? Nếu \$s1 = 0 thì đã đúng và in
dãy luôn

loop_chen:

addi \$t0, \$t0, -4 # A[j]

lw \$s2, 0(\$t0) # Lưu A[j] vào vị trí A[i]

sw \$s2, 4(\$t0)

bne \$t0, \$t5, loop_chen # Nếu A[j] chưa đến vị trí cần chen thì tiếp tục

sw \$t1, 0(\$t5) # Nếu đã đến vị trí chen thì thực hiện chen A[i]

print:

li \$s4, 0 # Biến chỉ mục cho việc in

print_char:

la \$a0, A

add \$s5, \$s4, \$s4

add \$s5, \$s5, \$s5

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

lw \$s6, 0(\$s5) # Lưu A[j] vào vị trí A[i]

beq \$s5, \$a3, in_xuong_dong

li \$v0, 1 # In số nguyên

move \$a0, \$s6

syscall

addi \$s7, \$a3, -4

beq \$s5, \$s7, skip

li \$v0, 11 # In ký tự dấu cách

li \$a0, ''

```

syscall
skip:
addi $s4, $s4, 1 # Tăng biến chỉ mục
j print_char

```

in_xuong_dong:

```

li $v0, 11      # In ký tự xuống dòng
li $a0, '\n'
syscall
la $a0, A

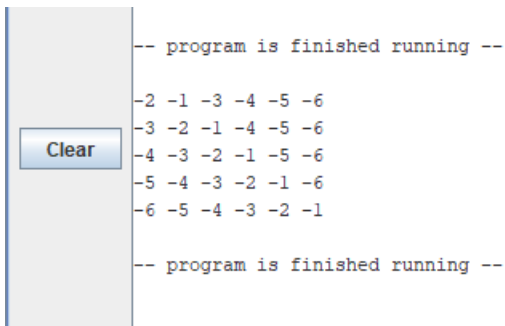
```

```

beq $a1, $a2, after_sort # Thực hiện vòng lặp đến A[n-1]
addi $s0, $s0, 1 # i++
addi $a2, $a2, 4 # Tăng $a2, thực hiện kiểm tra tiếp A[2]
j loop

```

- Kết quả:



```

-- program is finished running --
-2 -1 -3 -4 -5 -6
-3 -2 -1 -4 -5 -6
-4 -3 -2 -1 -5 -6
-5 -4 -3 -2 -1 -6
-6 -5 -4 -3 -2 -1
-- program is finished running --

```

Clear

→ Kết quả đúng với lý thuyết.