Bài thực hành số 6

Lớp: 139365 – Học phần: Thực Hành Kiến Trúc Máy Tính
Đào Minh Nhật – 20215107

Assignment 1:

Code:

```
#Lab 6, assginment 1
.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:

mspfx:  addi $v0,$zero,0 #initialize length in $v0 to 0
        addi $v1,$zero,0 #initialize max sum in $v1to 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:

```

Giải thích:

a0: địa chỉ của mảng A

a1: số lượng phần tử của mảng A. (n)

v0: chiều dài của tổng thỏa mãn

v1: kết quả của tổng cuối cùng

t0: chỉ số i

t1: tổng hiện tại

Dòng 15-18: Khởi tạo các giá trị v0, v1, t0, t1.

Dòng 20-21: Khởi tạo giá trị t2 = 5.t0 = 4i

Dòng 22 : Khởi tạo giá trị t3 = 4i + A = địa chỉ của A[i]

Dòng 23: Gán giá trị A[i] vào thanh t4 từ địa chỉ được lưu trong thanh t3.

Dòng 24: t1 = t1 + A[i]

Dòng 25, 26: Nếu tổng vừa tính mà lớn hơn tổng trước đó đã lưu trong v1 thì thực hiện jum tới mdfy để cập nhật lại tổng v1 và chiều dài v0.

Dòng 32, 33, 34: Cho tới khi nào i < n thì thực hiện jum tới loop để lặp lại vòng lặp

Assignment 2

Code:

```
#Lab 6, assignment 2
.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:

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

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

```

Giải thích:

a0: địa chỉ của A[0]

a1: địa chỉ của A[n-1]

v0: lưu địa chỉ của vị trí lơn nhất

v1: lưu giá trị lớn nhất

t0: lưu địa chỉ của vị trí hiện tại

t1: lưu giá giá trị của vị trí A[i] hiện tại

Dòng 15, 16: Kiểm tra nếu chỉ còn 1 phần tử thì jum tới done để kết thúc còn không thì jum tới max để tìm số lớn nhất trong dãy.

Dòng 18: Thực hiện gán giá trị của phần tử cuối cùng vào thanh ghi t0: t0 = A[n-1]

Dòng 19: Giá trị t0 (hay giá trị của phần tử cuối cùng) được lưu trữ trong thanh ghi có địa chỉ được lưu trữ tại thanh ghi v0(vị trí tìm được số lớn nhất trong dãy)

Dòng 20: Giá trị lớn nhất được tìm thấy là v1 được lưu trữ tại thanh ghi có địa chỉ được lưu trữ tại thanh ghi a1(vị trí cuối cùng)

⇨ Thực hiện đổi chỗ phần tử lớn nhất vừa tìm thấy với phần tử cuối cùng của dãy.

Dòng 21: Thanh a1 giảm đi 4 byte con trỏ sẽ trỏ đến phần tử trước của phần tử hiện tại

Dòng 26-28: Khởi tạo giá trị của v0 = a0 = địa chỉ của phần tử đầu tiên; v1 = A[0]; t0 = a0

Dòng 30: Nếu t0 = a1 (chỉ có 1 phần tử) thì có nghĩa là đã tìm được số lớn nhất trong dãy => jum tới ret

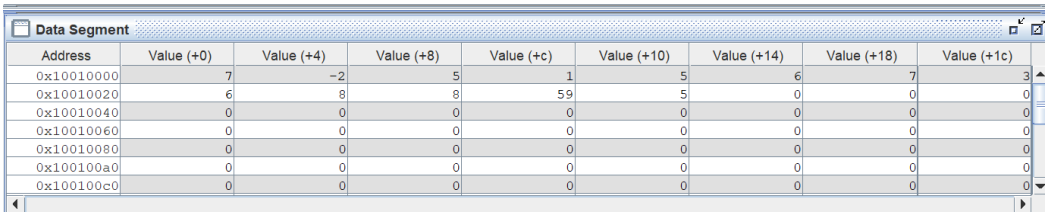
Dòng 31: Xét phần tử tiếp theo: t0 += 4

Dòng 32: Tải giá trị tại địa chỉ t0 vào thanh ghi t1

Dòng 33-36: Nếu t1 < v1 (chưa tìm được số lớn hơn số trước đó tìm được) thì thực hiện lại vòng lặp loop, ngược lại thực hiện thì cập nhật giá trị max mới và địa chỉ của nó lần lượt tại v1 và v0

Kết quả:

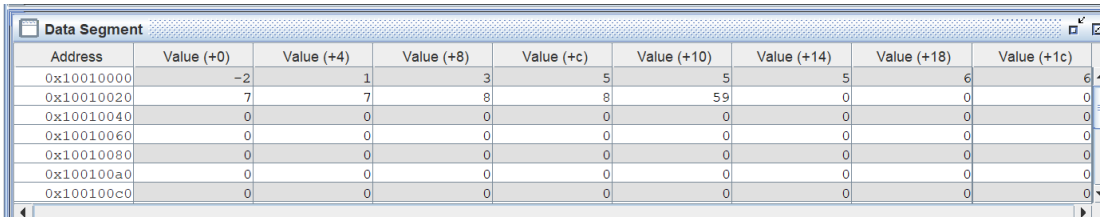
Trước khi sắp xếp:



Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	7	-2	5	1	5	6	7	3
0x10010020	6	8	8	59	5	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

Sau khi sắp xếp:



Data Segment

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

Assignment 3

Code:

```
1  # Lab 6, assignment 3
2  .data
3          A: .word 8,2,4,6,1,3,10,7,9,5,12,13,11
4          Aend: .word
5  .text
6  main:
7          li $a2, 13              #a2 = n
8          la $a0, A               #$a0 = 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         li $s0, 0       # i = 0
19         addi $t0, $a0, 0        # s0 = j = 0
20
21    loop_j:
```

```
22        mul $s1, $s0, 4
23        sub $t3, $a1, $s1      #t3 = n-i-1
24        beq $t0, $t3, end_loop_j       #nếu t0 = t3 ( j = n-i-1 ) thì kết thúc vòng lặp j, tăng i
25
26        lw $t1, 0($t0) # t1 = A[j]
27        lw $t2, 4($t0) # t2 = A[j+1]
28
29        slt $t5, $t1, $t2 # nếu t1 < t2 ( a[j] < a[j+1] => t5 = 1
30        bne $t5, $zero, no_swap
31
32        #swap
33        sw $t1, 4($t0)
34        sw $t2, 0($t0)
35
36        no_swap:
37        addi $t0, $t0, 4 #j++
38        j loop_j
39    end_loop_j:
```

```
40
41    loop_i:
42         addi $s0, $s0, 1        # i += 1
43         slt $t4, $s0, $a2       # nếu i < n thì t4 = 1
44         beq $t4, $zero, end_loop_i
45         addi $t0, $a0, 0        # s0 = j = 0
46         j loop_j
47    end_loop_i:
48
49         j after_sort
50
```

Giải thích:

a0: địa chỉ của A[0]

a1: địa chỉ của A[n-1]

a2: n-1

Dòng 18: Khởi tại s0 = 0 (i = 0)

Dòng 19: Khởi tạo t0 = địa chỉ của A[0]

Dòng 22-23: Tính t3 là địa chỉ của A[n-i-1]

Dòng 24 : Kiểm tra nếu t0 = t3 tức là j = n-i-1 thì sẽ kết thúc vòng lặp j. Ngược lại thì gán t1 = A[j] (dòng 26) và t2 = A[j+1] (dòng 27).

Dòng 29-30 : Kiểm tra nếu t1 < t2 tức là A[j] < A[j+1] thì không thực hiện đổi chỗ mà tăng j rồi thực hiện tiếp vòng lặp tiếp theo. Ngược lại thì thực hiện dòng 33, 34 để đổi chỗ.

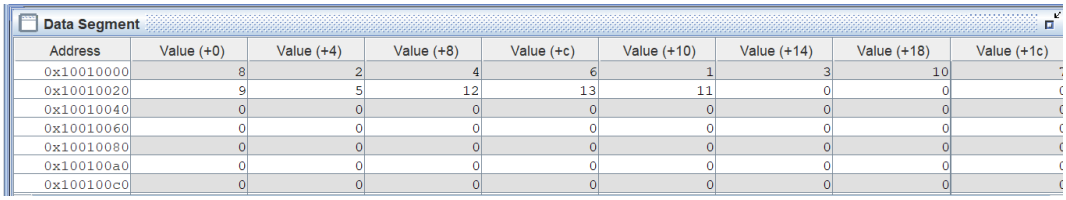
Dòng 33-34 : Đổi chỗ A[j] và A[j+1].

Dòng 43, 44 : Kiểm tra nếu i > n thì dừng vòng i, kết thúc bubble sort

Dòng 45-46: Set lại giá trị t0 = địa chỉ của A[0]. Rồi jum tới loop_j để bắt đầu vòng lặp j.

Kết quả:

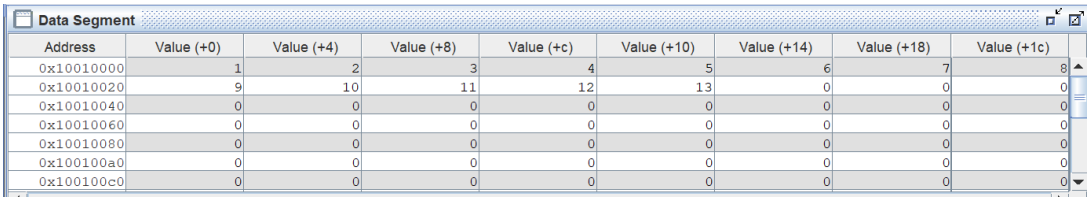
Trước khi sắp xếp:



Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	8	2	4	6	1	3	10	7
0x10010020	9	5	12	13	11	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

Sau khi sắp xếp:



Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	1	2	3	4	5	6	7	8
0x10010020	9	10	11	12	13	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

Assiginment 4

Code

```
# Lab 6, assignment 4
.data
        A: .word 8,2,4,6,1,3,10,7,9,5,12,13,11
.text
#void insertionSort(int *a, int n){
#    for(int i=1; i < n; i++){
#        int x = a[i], pos = i-1;
#        while(pos >= 0 && x < a[pos]){
#            a[pos+1] = a[pos];
#            pos--;
#        }
#        a[pos+1] = x;
#    }
#}

main:
        li $a2, 13              #a2 = n
        la $a0, A               #$a0 = Address(A[0])
        j sort                  #sort
after_sort:
        li $v0, 10 #exit
       syscall
end_main:

sort:
       li $t1, 0      # t1 = i = 0
       addi $t0, $a0, 0 # t0 = *a[0]
for:
       addi $t1, $t1, 1 # i++
       addi $t0, $t0, 4
       beq $t1, $a2, endFor   # i = n => end

       lw $t2, 0($t0) # t2 = A[i]     x = a[i]
       addi $t3, $t1, -1 #t3 = i-1    pos = i-1
       addi $t5, $t0, 0
  while:
       blt $t3, $zero, endWhile #pos < 0 -> end

       lw $t4, -4($t5)  #t4 = A[pos]

       # x < a[pos]-> s1 = 1
       slt $s1, $t2, $t4

        beq $s1, $zero, endWhile

        #a[pos+1] = a[pos]
        sw $t4, 0($t5)
        addi $t5, $t5, -4
        addi $t3, $t3, -1
        j while
   endWhile:
        sw $t2, 0($t5)
        j for
endFor:
        j after_sort
```

Giải thích:

a0: địa chỉ của A[0]

a2: n

t1: i

t0: địa chỉ của A[i]

t2 : A[i]

t3 : pos = i-1

t4 : A[pos]

Thực hiện insertion sort như đoạn code từ dòng 5-14 :

Dòng 26-27 : Khởi tạo i = 0 và t0 = địa chỉ a[0]

Dòng 31 : Kiểm tra nếu i = n thì kết thúc vòng lặp

Dòng 33 : t2 = A[i]

Dòng 34 : t3 = pos = i-1

Dòng 37 và 43 là điều kiện vòng lặp while.

Nếu pos < 0 hay x >= A[pos] thì dừng vòng while

Dòng 39: t4 = A[pos]

Dòng 46: A[pos+1] = A[pos]

Dòng 51: A[pos+1] = x = A[i].