ASSIGNMENT WEEK 6

-Họ tên: Hồ Tuấn Huy
-MSSV: 20225856

Assignment 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.

```
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  end_of_main:
10
11 mspfx:
12 addi $v0,$zero,0 #initialize length in $v0 to 0
13 addi $v1,$zero,0 #initialize max sum in $v1to 0
14 addi $t0,$zero,0 #initialize index i in $t0 to 0
15 addi $t1,$zero,0 #initialize running sum in $t1 to 0
16 loop:
17 add $t2,$t0,$t0 #put 2i in $t2
18 add $t2,$t2,$t2 #put 4i in $t2
19 add $t3,$t2,$a0 #put 4i+A (address of A[i]) in $t3
20 lw $t4,0($t3) #load A[i] from mem(t3) into $t4
21 add $t1,$t1,$t4 #add A[i] to running sum in $t1
22 slt $t5,$v1,$t1 #set $t5 to 1 if max sum < new sum
23 bne $t5,$zero,mdfy #if max sum is less, modify results
24 j test #done?
25 mdfy:
```

```
25  mdfy:
26  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:
29  addi $t0,$t0,1 #advance the index i
30  slt $t5,$t0,$a1 #set $t5 to 1 if i<n
31  bne $t5,$zero,loop #repeat if i<n
32  done: j mspfx_end
33  mspfx_end:
```

- Đưa địa chỉ bắt đầu của mảng A vào thanh ghi $a0, đưa giá trị 5 là số phần tử của mảng vào thanh ghi $a1.
- Đến phần mspfx (hàm tính toán dãy con liên tục có tổng lớn nhất):
  +Các lệnh addi và add để tính toán tổng và chiều dài dãy con liên tục có tổng lớn nhất.
  +lw $t4,0($t3)=>Lấy giá trị tại địa chỉ được lưu trong $t3 (địa chỉ của A[i]) và đưa vào thanh ghi $t4 để tính tổng.
  +slt $t5,$v1,$t1=>So sánh nếu giá trị trong $v1 (tổng lớn nhất hiện tại) bé hơn giá trị trong $t1 (tổng hiện tại của dãy con) để quyết định có cần cập nhật kết quả không.
  +bne $t5,$zero,mdfy=>Nhảy tới nhãn "mdfy" nếu điều kiện so sánh trên không đúng.
- Hàm mdfy: Tăng độ dài mảng lên i+1 và giá trị lớn nhất sẽ là giá trị hiện tại.
- Hàm test: Kiểm tra phần tử A[i] có phải phần tử cuối cùng hay không. Nếu không thì thực hiện tiếp vòng lặp còn nếu có thì kết thúc hàm mspfx và kết thúc chương trình.

=>***Kết quả:*** $v0 chứa độ dài của dãy con liên tục có tổng lớn nhất còn $v1 là tổng:

$v0	2	0x00000004
$v1	3	0x00000006

Assignment 2:

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.

```
1  .data
2  A: .word 9, 0, 6, 10, 9, 4, -4, 7, 1
3  Aend: .word
4  space: .asciiz " "
5  .text
6  main:
7  la $a0,A #$a0 = Address(A[0])
8  la $a1,Aend
9  li $s0, 0 #i=0
10 addi $a1,$a1,-4 #$a1 = Address(A[n-1])
11 j sort #sort
12 end_main:
13 sort:
14 beq $a0, $a1, done #single element list is sorted
15 j max #call the max procedure
16
17 after_max:
18 lw $t0,0($a1) #load last element into $t0
19 sw $t0,0($v0) #copy last element to max location
20 sw $v1,0($a1) #copy max value to last element
21 addi $a1,$a1,-4 #decrement pointer to last element
22 j sort #repeat sort for smaller list
23 done: j print_array
24 max:
25 addi $v0,$a0,0 #init max pointer to first element
```

```
26  lw $v1,0($v0) #init max value to first value
27  addi $t0,$a0,0 #init next pointer to first
28  loop:
29  beq $t0,$a1,ret #if next=last, return
30  addi $t0,$t0,4 #advance to next element
31  lw $t1,0($t0) #load next element into $t1
32  slt $t2,$t1,$v1 #(next)<(max) ?
33  bne $t2,$zero,loop #if (next)<(max), repeat
34  addi $v0,$t0,0 #next element is new max element
35  addi $v1,$t1,0 #next value is new max value
36  j loop #change completed; now repeat
37  ret:
38  j after_max
39
40  print_array:
41  la $a1, Aend
42  add $a1, $a1, -4
43  loop_after:
44  add $s2, $s0, $s0 #put 2i in $s2
45  add $s2, $s2, $s2 #put 4i in $s2
46  add $s3, $s2, $a0 #put 4i+A (address of A[i] in $s3)
```

```
47  slt $s1, $a1, $s3
48  bne $s1, $zero, end
49  lw $s4, 0($s3)
50  add $s0, $s0, 1
51  j print
52  print:
53  li $v0, 1
54  addi $a0, $s4, 0
55  syscall
56  li $v0, 4
57  la $a0, space
58  syscall
59  la $a0, A
60  j loop_after
61  end:
```

- Hàm main: Gán địa chỉ đầu của mảng vào thanh ghi $a0 và địa chỉ cuối vào thanh ghi $a1.
- Hàm sort: Điều kiện dừng là $a0 = $a1, rồi sau có chạy xuống hàm print để in ra mảng còn nếu không thì chạy xuống hàm max để thực thi chương trình tiếp.
- Hàm max: Kiểm tra xem phần tử tiếp theo có phải là phần tử lớn nhất trong mảng chưa sắp hay không, nếu có thì lưu địa chỉ lớn nhất tại $v0 và giá trị lớn nhất tại $v1.
- Hàm loop: Kiểm tra xem phần tử tiếp theo trong mảng có phải lớn nhất trong mảng chưa sắp xếp hay chưa, nếu có thì lưu địa chỉ vào $v0 và lưu giá trị vào $v1. Sau đó thực hiện hoán đổi giá trị của max với giá trị con trỏ $a1, rồi dịch lên địa chỉ tiếp theo và tiếp tục thực hiện sắp xếp.
- Hàm print: Sử dụng vòng lặp để duyệt từng phần tử của mảng A khi đã sắp xếp sau đã sử dụng mã nguồn là 1 để in ra số nguyên và 4 để in ra ký tự (ở đây là space).

=>Kết quả:

```
-- program is finished running (dropped off bottom) --

-4 0 1 4 6 7 9 9 10
-- program is finished running (dropped off bottom) --
```

Assignment 3:

Assignment 3

Write a procedure to implement bubble sort algorithm.

```
1  .data
2  A: .word 3, 9, 2, 4, 3, 5
3  Aend: .word
4  .text
5  la $a0, A
6  la $a1, Aend
7  li $s0, 0
8  li $s1, -1
9  dem:
10 beq $a1, $a0, size
11 addi $a1, $a1, -4
12 addi $s0, $s0, 1
13 j dem
14 size:
15 addi $t0, $s0, -1
16 loop1:
17 addi $s1, $s1, 1
18 li $s2, 0
19 beq $s1, $t0, Exit
20 loop2:
21 sub $t2, $t0, $s1
22 beq $s2, $t2, loop1
23
24 if_swap:
25 sll $t3, $s2, 2
```

```
25 sll $t3, $s2, 2
26 add $s3, $a0, $t3
27 lw $v0, 0($s3)
28 addi $s3, $s3, 4
29 lw $v1, 0($s3)
30 sle $t4, $v0, $v1
31 beq $t4, $zero, swap
32 addi $s2, $s2, 1
33 j loop2
34 swap:
35 sw $v0, 0($s3)
36 addi $s3, $s3, -4
37 sw $v1, 0($s3)
38 addi $s2, $s2, 1
39 j loop2
40 Exit:
41 li $v0, 10
42 syscall
```

- Đưa địa chỉ bắt đầu của mảng A vào thanh ghi $a0, địa chỉ cuối của mảng A vào thanh ghi $a1.
- Khởi tạo biến $s0 làm chỉ số lặp (i=0), biến $s1 để theo dõi số lần đã sắp xếp.
- dem=>vòng lặp để đếm số lượng phần tử trong mảng:

+beq $a1, $a0, size=>Kiểm tra nếu con trỏ cuối mảng đã đến vị trí con trỏ đầu mảng thì chuyển đến size.
+addi $a1, $a1, -4=>địa chỉ $a1 giảm để đến địa chỉ của từng phần tử trong mảng.
+addi $s0, $s0, 1=>số lượng phần tử tăng 1 (tăng biến đếm i lên 1)
=>Sau đó quay lại kiểm tra tiếp.

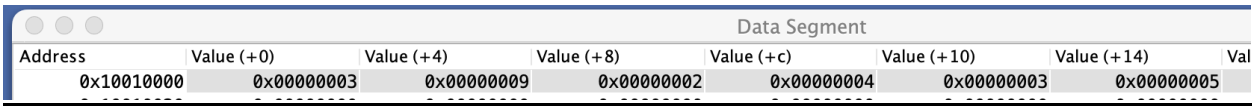
- size=>xác định kích thước mảng: khởi tạo $t0 là biến đếm cho việc duyệt từ phải sang trái.
- loop1=>Vòng lặp ngoài để duyệt từng phẩn tử trong mảng:

+ Tăng biến đếm $s1 lên 1 sau mỗi vòng lặp (để giảm số lần so sánh ở vòng lặp tiếp theo).
+ Khởi tạo biến đếm $s2 cho việc duyệt từ trái sang phải để so sánh.
+Kiểm tra nếu đã duyệt qua tất cả các phần tử thì thoát vòng lặp (beq $s1, $t0, Exit).

- loop2: Vòng lặp trong để so sánh từng cặp phần tử:
  + sub $t2, $t0, $s1=>Tính chỉ số phần tử phải so sánh.
  + beq $s2, $t2, loop1=>Kiểm tra nếu đã so sánh hết các cặp thì quay ra vòng lặp ngoài.
- If_swap=>xác định điều kiện cần hoán đổi:
  +sll $t3, $s2, 2=>tính địa chỉ phần tử hiện tại bằng cách nhân chỉ số của phần tử với kích thước của 1 từ (4 byte).
  +add $s3, $a0, $t3=>tính địa chỉ phần tử tiếp theo trong mảng để so sánh.
  +lw $v0, 0($s3)=>load giá trị của phần tử hiện tại từ bộ nhớ.
  +addi $s3, $s3, 4=>di chuyển địa chỉ để load giá trị phần tử tiếp theo.
  +lw $v1, 0($s3)=>load giá trị phần tử tiếp theo từ bộ nhớ.
  +sle $t4, $v0, $v1=>so sánh nếu giá trị phần tử hiện tại bé hơn hoặc bằng giá trị phần tử tiếp theo. Nếu không thoả mãn thì chuyển đến swap để hoán đổi 2 phần tử.
- swap=>hoán đổi 2 phần tử:
  +sw $v0, 0($s3) =>lưu giá trị phần tử hiện tại vào vị trí phần tử tiếp theo.
  +addi $s3, $s3, -4=>di chuyển địa chỉ để lưu giá trị phần tử tiếp theo vào vị trí phần tử hiện tại.
  +sw $v1, 0($s3)=>lưu giá trị phần tử tiếp theo vào vị trí phần tử hiện tại.
  +addi $s2, $s2, 1 =>tăng biến đếm để chuyển sang phần tử tiếp theo trong vòng lặp.

⇨ ***<u>Kết quả:</u>***

Ban đầu:



Sau khi sort: