BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH

LAB 6

Họ và tên: Phạm Vân Anh

MSSV: 20214988

Mã lớp: 139365

ASSIGNMENT 1:

1. Code

ex1.asm | ex2.asm | ex3.asm | ex4.asm

```
#Laboratory 6, Home Assigment 1
#PhamVanAnh_20214988
.data
      A: .word -2, 6, -1, 3, -2, 10, 8, -2
      message1: .asciiz "Max sum prefic lenght is: "
      message2: .asciiz "\nPrefic is: "
.text
main:
      la $a0, A         #a0: dia chi cua A
      li $a1, 8         #a1: so phan tu cua mang
      j mspfx           #mspfx: max sum prefix: tong lon nhat cua day con
      nop
#---------------------------------------------------------------
#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

#reachs.
# @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
```

ex1.asm | ex2.asm | ex3.asm | ex4.asm

```
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
                 #tao ra A[i]
      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:
      move $t5, $v0
      j continue
```

```
continue:
      li   $v0, 4
      la   $a0, message1
      syscall

      li   $v0, 1
      la   $a0, 0($v1)
      syscall

      li    $v0, 4
      la    $a0, message2
      syscall

      li    $v0, 1
      la    $a0, ($t5)
      syscall
mspfx end:
```

Thực hiện gõ chương trình vào công cụ MARS

2. Giải thích

- Hàm main:
  - Gán địa chỉ của A vào $a0
  - Gán số phần tử của mảng A vào $a1
- Hàm mspfx: Tìm max sum prefix
  - Gán chiều dài của mảng con có tổng lớn nhất vào $v0
  - Gán tổng của mảng con có tổng lớn nhất vào $v1
  - Gán index i = $t0

- Gán tổng của i phần tử đầu tiên vào $t1

- Hàm loop:
  - Dòng 35 → 43: Thực hiện truy cập địa chỉ A[i]
  - Dòng 38: lw $t4, 0($t3): load giá trị của địa chỉ $t3 vào $t4
  - Dòng 40: tính tổng i phần tử đầu tiên, lưu vào thanh ghi $t1
  - So sánh với max vừa tìm được:
    - Nếu lớn hơn max → cập nhật max mới (mdfy):
      + Dòng 45: Tăng chiều dài của mảng con vừa tìm được thêm 1
      + Dòng 46: max = tổng hiện tại
    - Nếu nhỏ hơn max (test) → kiểm tra xem A[i] có phải phần tử cuối cùng trong mảng không. Nếu không thì tiếp tục vòng lặp
- In ra các giá trị: Dòng 55 → 69
  - Dòng 55 → 61: In ra tổng của dãy con có giá trị lớn nhất
  - Dòng 63 → 69: In ra số giá trị của dãy con

3. Thử với mảng A= { -2, 6, -1, 3, -2, 10, 8, -2}

```
Max sum prefic lenght is: 22
Prefic is: 7
-- program is finished running (dropped off bottom) --
```

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-2	6	-1	3	-2	10	8	-2
0x10010020	544760141	544044403	1717924464	1814061929	1751608933	1936269428	167780410	1717924432
0x10010040	1763730281	2112115	0	0	0	0	0	0

Kết quả:

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	1
$v1	3	22
$a0	4	7
$a1	5	8
$a2	6	0
$a3	7	0
$t0	8	8
$t1	9	20
$t2	10	28
$t3	11	268501020
$t4	12	-2
$t5	13	7

ASSIGNMENT 2:

1. Code

ex2.asm*

```
#Laboratory 6, Home Assigment 2
#PhamVanAnh_20214988

.data
      A: .word 10, -5, -4, -3, 33, -7, 12, 16, 27, -9
      Aend: .word
      tab: .asciiz " "
      message: .asciiz "Day sau khi duoc sap xep: "
.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:
      la    $s0, A
      la    $s1, Aend
      addi  $s1, $s1, -4
      li    $v0, 4
      la    $a0, message
      syscall
print_loop:
        li $v0, 1
        add $t2, $s0, 0
        lw  $a0, 0($t2)
        syscall

        beq $s0, $s1, out
        li  $v0, 4
        la  $a0, tab
        syscall
        add $s0, $s0, 4
        j print_loop
end main:
```

Line: 35 Column: 1 ☑ Show Line Numbers

```
35
36  #-----------------------------------------------------------------
37  #procedure sort (ascending selection sort using pointer)
38  #register usage in sort program
39  #$a0 pointer to the first element in unsorted part
40  #$a1 pointer to the last element in unsorted part
41  #$t0 temporary place for value of last element
42  #$v0 pointer to max element in unsorted part
43  #$v1 value of max element in unsorted part
44  #-----------------------------------------------------------------
45  sort:
46        beq $a0, $a1, done      #single element list is sorted
47        j   max                 #call the max procedure
48  after_max:
49        lw $t0, 0($a1)          #load last element into $t0
50        sw $t0, 0($v0)          #copy last element to max location
51        sw $v1, 0($a1)          #copy max value to last element
52        addi $a1, $a1, -4       #decrement pointer to last element
53        j sort                  #repeat sort for smaller list
54  done: j after_sort
55  #-----------------------------------------------------------------------
56
57  #Procedure max
58  #function: fax the value and address of max element in the list
59  #$a0 pointer to first element
60  #$a1 pointer to last element
61  #-----------------------------------------------------------------------
62
```

```
63  max:
64        addi  $v0, $a0, 0             #init max pointer to first element
65        lw    $v1, 0($v0)             #init max value to first value
66        addi  $t0, $a0, 0             #init next pointer to first
67  loop:
68        beq   $t0, $a1, ret           #if next=last, return
69        addi  $t0, $t0, 4             #advance to next element
70        lw    $t1, 0($t0)             #load next element into $t1
71        slt   $t2, $t1, $v1           #(next)<(max) ?
72        bne   $t2, $zero, loop        #if (next)<(max), repeat
73        addi  $v0, $t0, 0             #next element is new max element
74        addi  $v1, $t1, 0             #next value is new max value
75        j     loop                    #change completed; now repeat
76  ret:
77  j after_max
78  out:
79
```

Thực hiện gõ chương trình vào công cụ MARS

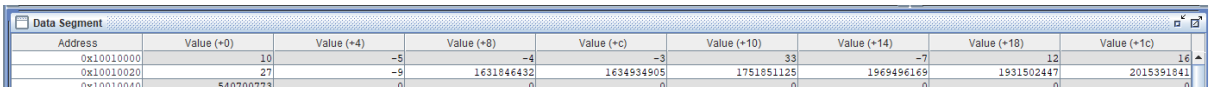
2. Giải thích code

- Injection Sort:
  - Mỗi lần duyệt tìm được phần tử lớn nhất → cho về cuối mảng
  - Gán $s0 trỏ tới phần tử đầu tiên của mảng
  - Gán $a1: trỏ tới phần tử cuối cùng của mảng

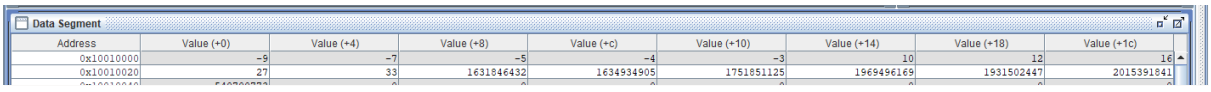
- Gán $v0: biến con trỏ lưu địa chỉ của phần tử có giá trị lớn nhất từ dãy chưa xếp
- Gán $v1: Giá trị của phần tử lớn nhất chưa được sắp xếp

- Hàm sort: Kiểm tra điều kiện dừng của vòng lặp: Dòng 46, 47
  - So sánh $a0 và $a1
    - Nếu a0 = a1 → done
    - Nếu a0 != a1 → max
- Hàm print_loop: In ra mảng sau khi sắp xếp
- Hàm after_max: Đổi chỗ phần tử lớn nhất hiện tại với phần tử cuối cùng của mảng
  - Lấy giá trị từ địa chỉ của a1 → t0
  - Lưu giá trị của t0 vào v0
  - Lưu giá trị của v1 vào a1
  - Sau đó, a1 giảm đi 4 để đến phần tử tiếp theo
  - → sort
- Hàm loop:
  - Kiểm tra con trỏ có nằm ở cuối không
    - Có: Vừa hoàn thành 1 vòng lặp → after_max
    - Không: Tăng con trỏ thêm 4 → sang phần tử tiếp. Kiểm tra xem phần tử đó có lớn hơn max không, nếu không → loop. Nếu có → cập nhật max → loop
- Hàm after_sort: Dòng 16 → 21
  - Gán địa chỉ của A vào $s0
  - Gán địa chỉ của Aend (đánh dấu kết thúc mảng) vào $a1
  - Lấy a1 = a1 – 4 (a1= address A[n-1])
  - Dòng 19 → 21: In ra message: “Chuoi sau khi duoc sap xep: “

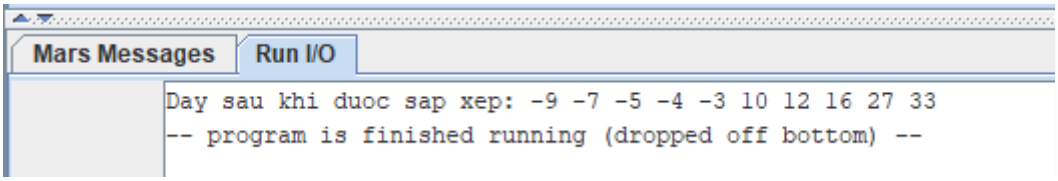
3. Kết quả:
   - Trước khi sắp xếp:



   - Sau khi sắp xếp:



   - Kết quả:



ASSIGNMENT 3:

1. Code

```
.data
      A: .word 5, 7, -3, 6, 3, -8, -4
      Aend: .word
      Message: .asciiz  "Mang sau khi duoc sap xep: "
      tab: .asciiz " "
.text
main:
      la $a0, A           # $a0 = Address(A[0])
      la $a1, Aend
      j sort              # sort
after_sort:
      j print_out
end_main:
# BUBBLE SORT
sort:
      beq $a0, $a1, done            # single element list is sorted
      j   travasal                  # call the travasal procedure
done:
      j     after_sort
travasal:
      addi $v0, $a0, 0          # init travasal pointer to first element
      addi $t0, $a0, 0          # init next pointer to first
      addi $a1, $a1, -4         # pointer to last element
      beq $a0, $a1, done        # completely sort
loop:
      beq  $v0, $a1, travasal          # if current=last, bubbleSort
      lw   $v1, 0($v0)
      addi $t0, $v0, 4          # advance to next element
      lw   $t1, 0($t0)          # load next element into $t1
      slt  $t2, $t1, $v1        # (next)<(current) ?
      bne  $t2, $zero, swap     # if (next)>(current), swap
      addi $v0, $v0, 4          # current point to next
      j    loop

```

```
35  swap:
36        add $t5, $v1, $zero      # store value has address ($v1) to $t5 (temp)
37        sw  $t1, 0($v0)          # copy next value to current
38        sw  $t5, 0($t0)          # copy temp value to next
39        addi $v0, $v0, 4         # current point to next
40        j   loop                 # change completed; now repeat
41
42  print_out:
43          li $v0, 4
44          la $a0, Message
45          syscall
46          la $a0, A
47          la $a1, Aend
48          add $t0, $a0, 0
49  print_el:                    # in ra tung phan tu trong mang
50          lw     $t1, 0($t0)   # lấy giá trị từ địa chỉ $t0
51          li     $v0, 1        # Thực hiện in giá trị ra màn hình
52          move   $a0, $t1
53          syscall
54
55          li $v0, 4            # in dấu cách tab khoảng cách giữu các phần tử
56          la $a0, tab
57          syscall
58
59          add  $t0, $t0, 4     # tiếp tục trỏ đến phần tử tiếp theo của mnagr
60          bne  $t0, $a1, print_el # kiểm tra xem có trỏ đến phần tử cuối cùng của mảng?
61
62          li $v0, 10              # exit()
63          syscall                 #
```

Line: 34 Column: 7 Show Line Numbers

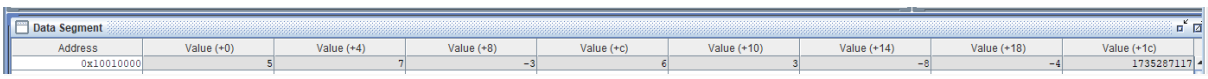
Thực hiện gõ chương trình vào công cụ MAR

2. Giải thích code:

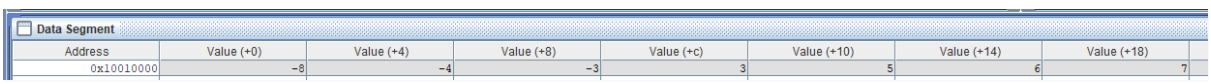
- Hàm sort: Duyệt từ phần tử đầu tiên tới phần tử cuối cùng
  - Dòng 25: Duyệt lại từ đầu mảng (sau khi tìm được phần tử lớn nhất đưa về cuối dãy)
  - So sánh 2 phần tử liền nhau. Nếu số sau < số hiện tại → swap
- Hàm swap:
  - Gán giá trị của $t1 vào $t5
  - Hoàn đổi giá trị của current và next thông qua $t5
  - Tăng $v0 (current) trỏ vào các phần tử tiếp theo

3. Kết quả:

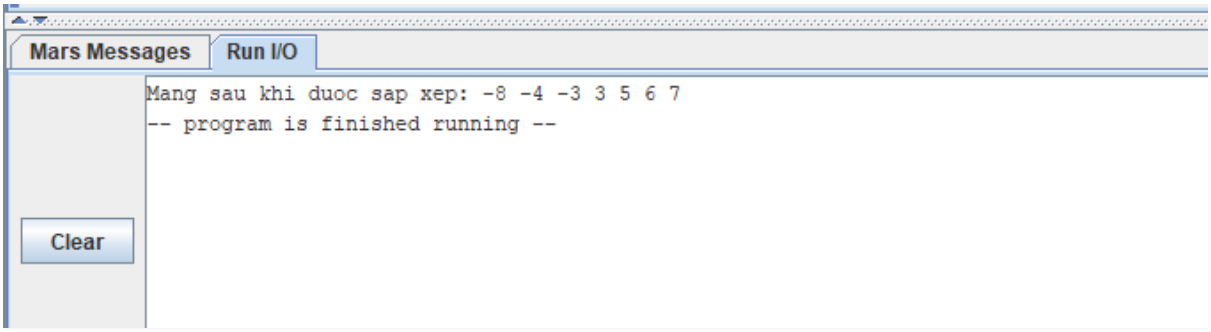
- Trước khi duyệt:



- Sau khi duyệt:



- Kết quả:



ASSIGNMENT 4:

1. Code

ex2.asm | ex3.asm | ex4.asm

```
#Laboratory 6, Home Assigment 4
#PhamVanAnh_20214988

.data
A: .word 10, 13, 6, 3, 7, 37, 8
Aend: .word
Message: .asciiz  "Mang sau khi duoc sap xep: "
tab: .asciiz " "

.text
main:   la $a0, A           # $a0 = Address(A[0])
        la $a1, Aend        # $a1 = Address(A[n-1]) + 4 //dieu kien dung vong lap
        j sort              # sort
after_sort:  j   print_out  # in ra man hinh mang sau khi sắp xếp
end_main:


# INSERTION SORT
sort:   beq $a0, $a1, done       # single element list is sorted
        j   travasal                  # call the travasal procedure
done:   j     after_sort

travasal:    addi $v0, $a0, 4    # init pointer to second element
             addi $t0, $v0, 0         # init previous pointer to $v0
loop:   beq  $v0, $a1, after_sort  # if next=last, return
        lw   $v1, 0($v0)         # init value to $v1
        addi $t0, $v0, -4        # advance to next element
        lw   $t1, 0($t0)         # load previous element into $t1
        sgt  $t2, $t1, $v1       # (pre)>(cur) ?
        bne  $t2, $zero, progress    # if (pre)<(cur), progress
        addi $v0, $v0, 4         # next element
        addi $t0, $t0, 4         #
        j    loop                # change completed; now repeat

```

Line: 34 Column: 1 ☑ Show Line Numbers

```
progress:       addi  $t5, $v1, 0   # value of current ($v1)
                addi  $t3, $v0, 0   # phan tu tinh tien
repeat:         sw    $t1, 0($t3)   # begin move back
                add   $t3, $t3, -4  # the next element move back
                beq   $t0, $a0, end   # check ($t0 == $a0) tránh trường hợp duyệt quá mảng
                add   $t0, $t0, -4
                lw    $t1, 0($t0)
                slt   $t2, $t1, $t5           # (pre)<(cur) ?
                beq   $t2, $zero, repeat      # if (pre)<(cur), repeat
end:            sw    $t5, 0($t3)
                add   $v0, $v0, 4
                j     loop
end_loop:
```

```
print_out:
        li $v0, 4
        la $a0, Message
        syscall
        la $a0, A
        la $a1, Aend
        add $t0, $a0, 0
print_el:                   # in ra tung phan tu trong mang
        lw      $t1, 0($t0)   # lấy giá trị từ địa chỉ $t0
        li      $v0, 1        # Thực hiện in giá trị ra màn hình
        move    $a0, $t1
        syscall

        li $v0, 4             # in dấu cách tab khoảng cách giữu các phần tử
        la $a0, tab
        syscall

        add   $t0, $t0, 4     # tiếp tục trỏ đến phần tử tiếp theo của mnagr
        bne   $t0, $a1, print_el # kiểm tra xem có trỏ đến phần tử cuối cùng của mảng?

        li $v0, 10              # exit()
        syscall                 #
```

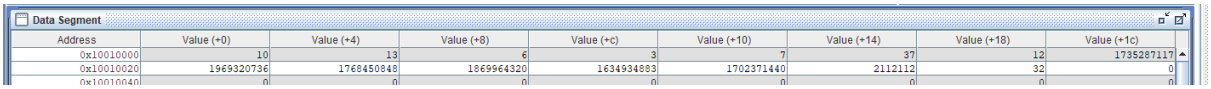
Thực hiện gõ chương trình vào công cụ MARS

2. Giải thích

- Gán $a0 mang địa chỉ của A
- Gán $s1 mang địa chỉ cuối mảng
- Hàm sort:
  - Duyệt từ phần tử A[1], so sánh với các phần tử trước đó của mảng. Dừng nếu $v0 = $a1
    - Nếu số trước lớn hơn → progress
    - Nếu số trước nhỏ hơn → done
- Hàm progress
  - Gán giá trị của phần tử cần sắp xếp lại vào $t5
  - Gán $t3 là phần tử bắt đầu đi về sau

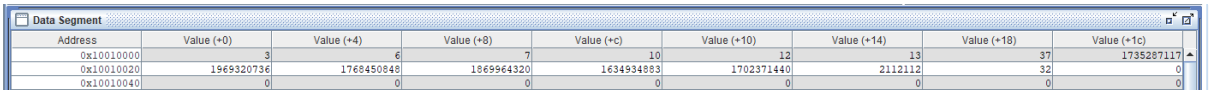
3. Kết quả:

- Trước khi xếp:


Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	10	13	6	3	7	37	12	1735287117
0x10010020	1969320736	1768450848	1869964320	1634934883	1702371440	2112112	32	0
0x10010040	0	0	0	0	0	0	0	0

- Sau khi xếp:


Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	3	6	7	10	12	13	37	1735287117
0x10010020	1969320736	1768450848	1869964320	1634934883	1702371440	2112112	32	0
0x10010040	0	0	0	0	0	0	0	0

- Kết quả: