Báo cáo thực hành KTMT tuần 6

Họ và tên: Đỗ Gia Huy

MSSV: 20215060

Assignment 1

1. Code

```
.data
        A: .word 0:100
        message1: .asciiz "Nhap so luong phan tu: "
        message2: .asciiz "\n"
        message3: .asciiz "Tong day con lon nhat: "
.text
main:
        la      $a0, message1
        li      $v0, 4
        syscall

        li      $v0,5
        syscall
        move $a1,$v0
        #       Nhap cac phan tu cua mang

        li      $t1,0
        la      $a0,A
lap:
        li      $v0,5
        syscall
```

```
        sw      $v0,($a0)
        addi    $t1,$t1,1
        addi    $a0,$a0,4
        blt     $t1,$a1,lap
        j       mspfx
        nop

continue:
        la      $a0, message3
        li      $v0, 4
        syscall

        li      $v0, 1
        move $a0, $v1
        syscall

        li      $v0, 10
        syscall
        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 reachs.
```

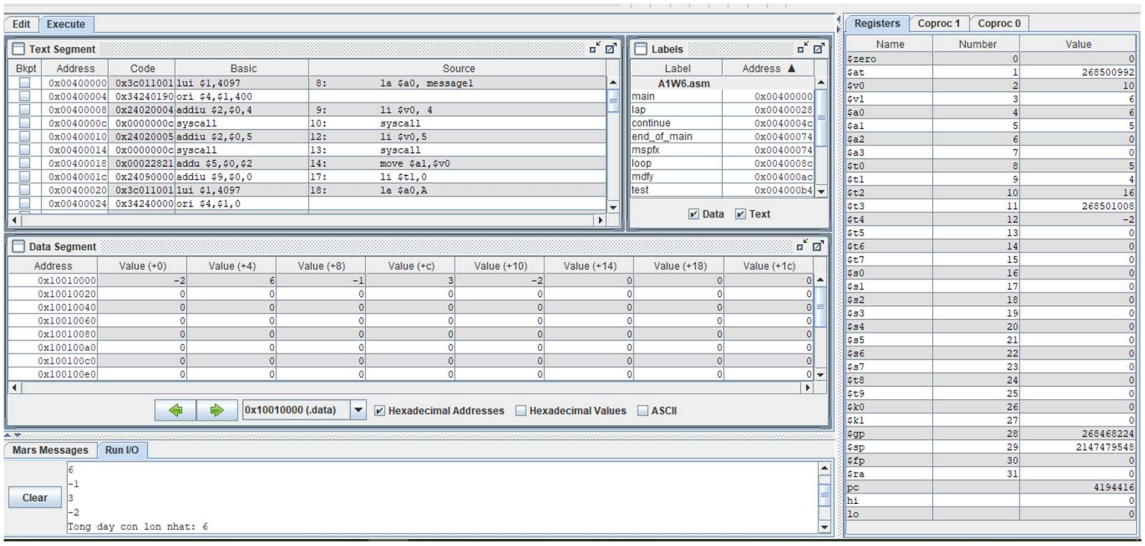
```
# @param[out] v1 the max sum of a certain sub-array
#---------------------------------------------------------------
#Procedure mspfx
#function: find the maximum-sum prefix in a list of integers
#the base address of this list(A) in $a0 and the number of
#elements is stored in a1
mspfx:
        la      $a0, A
        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

mspfx_end:

2. Kết quả



Khi nhập 5 phần tử -2, 6, -1, 3, -2 thì kết quả cho ra màn hình là 6

Tổng $v1 = 6$

⇨ **Kết quả trên đúng với lý thuyết**

Assignment 2

Selection sort là một thuật toán sắp xếp đơn giản, tìm kiếm và chọn phần tử nhỏ nhất hoặc lớn nhất trong danh sách và đổi chỗ nó với phần tử đầu tiên của danh sách. Tiếp tục quá trình này cho đến khi danh sách được sắp xếp hoàn toàn.

Ý tưởng của selection sort:

1. Xác định phần tử nhỏ nhất (hoặc lớn nhất) trong danh sách.
2. Hoán đổi phần tử nhỏ nhất (hoặc lớn nhất) với phần tử đầu tiên của danh sách.
3. Tiếp tục sắp xếp danh sách con còn lại (trừ phần tử đã sắp xếp) bằng cách lặp lại bước 1 và 2 cho đến khi danh sách được sắp xếp hoàn toàn.

Điều này tạo ra một danh sách con được sắp xếp ở đầu danh sách ban đầu, danh sách con này tiếp tục được lặp lại đến khi danh sách được sắp xếp hoàn toàn. Thuật toán selection sort hoạt động với tốc độ O(n^2) trong trường hợp xấu nhất.

1- Selection sort (Tăng dần)

1.1- Code

```
.data
    A: .space 100 #khai bao mang A
    Aend: .word
    Message1: .asciiz "Do dai mang la: "
    Message2: .asciiz "Nhap phan tu mang : "
    Message3: .asciiz "\n "
    ms: .asciiz " "
.text
main:
    la    $a3, A # gan $a3 la dia chi phan tu dau tien cua mang
    j     insert
after_insert:
    la    $a0,A #$a0 = Address(A[0])
    la    $a1,Aend
    la    $t8, ($t0)
    mul   $t7, $t0, 4
    add   $a1, $a0, $t7
    add   $a1, $a1, -4
    j     sort #sort
after_sort:
```

```
        li      $v0, 10 #exit
        syscall
end_main:

print:
        beq     $t9, $t8, after_print
        la      $a0, A
        mul     $t6, $t9, 4
        add     $t7, $a0, $t6
        lw      $a0, ($t7)
        li      $v0, 1
        syscall
        li      $v0, 4
        la      $a0, ms
        syscall
        addi    $t9, $t9, 1
        j       print

insert:
        li      $v0, 4 #syscall in ra chuoi
        la      $a0, Message1
        syscall
        li      $v0, 5
        syscall
        la      $t0, ($v0) #luu tam thoi do dai mang vao $t0
        li      $t1, 0
loop_insert:
        beq     $t1, $t0, after_insert #quay tro lai main
```

```
        li      $v0, 4 #syscall in ra chuoi
        la      $a0, Message2
        syscall
        li      $v0, 5
        syscall
        sw      $v0, 0($a3)
        addi    $t1, $t1, 1
        add     $a3, $a3, 4
        j       loop_insert


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 sort #repeat sort
for smaller list
        li      $v0, 4 #syscall in ra chuoi
        la      $a0, Message3
        syscall
        li      $t9, 0
        j       print
after_print:
        j       sort
done:


        j       after_sort
```

```
max:
        la      $a0, A
        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
```

1.2- Kết quả

Nhập số phần tử mảng n = 13

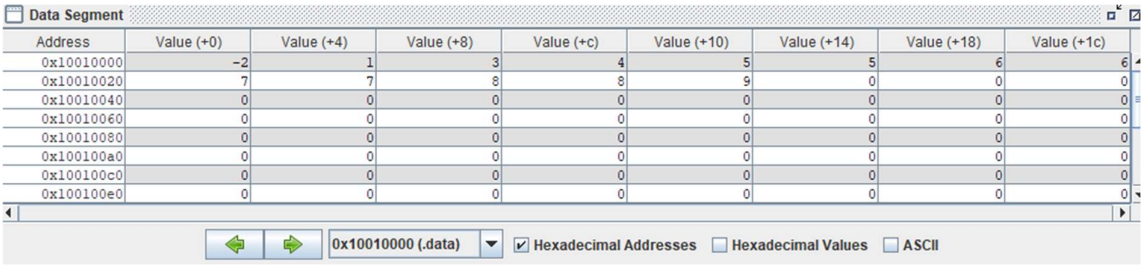
Mảng A ban đầu: 7,-2,5,1,4,6,7,3,6,8,8,9,5

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	7	-2	5	1	4	6	7	3
0x10010020	6	8	8	9	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
0x100100e0	0	0	0	0	0	0	0	0

0x10010000 (.data) ☑ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

Mảng A sau khi sắp xếp: -2,1,3,4,5,5,6,6,7,7,8,8,9



⇨ **Kết quả này đúng với lý thuyết**

2- Selection sort (Giảm dần) (Làm thêm)

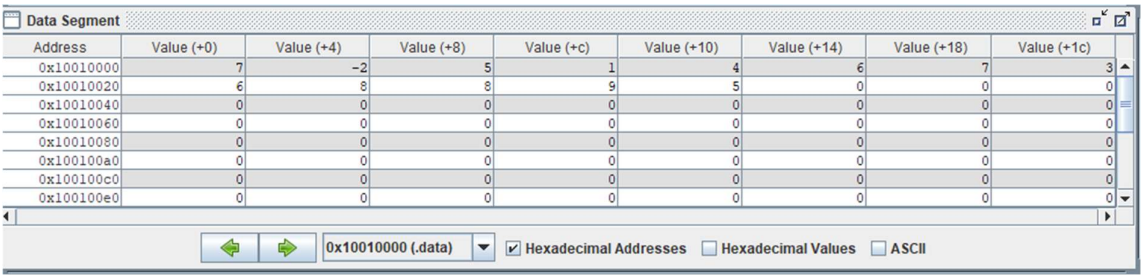
2.1- Code

```
.data
        A: .word 7,-2,5,1,4,6,7,3,6,8,8,9,5
        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
            li      $s6, 0
after_sort: li      $v0, 10 #exit
            syscall
end_main:
#---------------------------------------------------------
#procedure sort (Descending selection sort using pointer)
#register usage in sort program
#$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
#$v0 pointer to min element in unsorted part
#$v1 value of min element in unsorted part
#---------------------------------------------------------
```

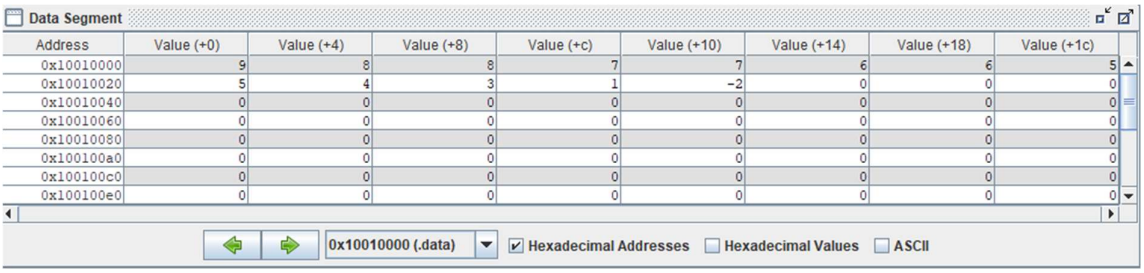
```
sort:       beq   $a0,$a1,done #single element list is sorted
            j     min #call the min procedure
after_min:  lw    $t0,0($a1) #load last element into $t0
            sw    $t0,0($v0) #copy last element to min location
            sw    $v1,0($a1) #copy min value to last element
            addi  $a1,$a1,-4 #decrement pointer to last element
            j     sort #repeat sort for smaller list
done:       j     after_sort
#---------------------------------------------------------------------
#Procedure min
#function: fax the value and address of min element in the list
#$a0 pointer to first element
#$a1 pointer to last element
#---------------------------------------------------------------------
min:        addi  $v0,$a0,0 #init min pointer to first element
            lw    $v1,0($v0) #init min 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   $t2,$v1,$t1 #(next)>(min) ?
            bne   $t2,$zero,loop #if (next)>(min), repeat
            addi  $v0,$t0,0 #next element is new min element
            addi  $v1,$t1,0 #next value is new min value
            j     loop #change completed; now repeat
ret:        j     after_min
```

2.2- Kết quả

Mảng A ban đầu: 7,-2,5,1,4,6,7,3,6,8,8,9,5



Mảng A sau khi sắp xếp: 9,8,8,7,7,6,6,5,5,4,3,1,-2



⇨ **Kết quả trên đúng với lý thuyết**

Assignment 3

Bubble sort là một thuật toán sắp xếp đơn giản, lặp lại việc so sánh và đổi chỗ các phần tử liên tiếp nhau nếu chúng không đúng thứ tự mong muốn. Trong mỗi lần lặp, phần tử lớn nhất sẽ được đưa lên đầu danh sách và tiếp tục sắp xếp danh sách con còn lại.

Ý tưởng của bubble sort như sau:

1. Bắt đầu từ đầu danh sách, so sánh phần tử thứ i với phần tử thứ i+1.
2. Nếu phần tử thứ i lớn hơn (hoặc nhỏ hơn) phần tử thứ i+1, hoán đổi chúng.
3. Tiếp tục lặp lại bước 1 và 2 cho đến khi đi qua tất cả các phần tử trong danh sách.
4. Lặp lại quá trình trên cho đến khi không có phần tử nào được hoán đổi nữa.

Trong mỗi lần lặp, phần tử lớn nhất sẽ được đưa lên đầu danh sách, cho đến khi danh sách được sắp xếp hoàn toàn. Thuật toán bubble sort hoạt động với tốc độ $O(n^2)$ trong trường hợp xấu nhất. Mặc dù thuật toán này đơn giản, nhưng nó thường chỉ được sử dụng cho các danh sách nhỏ hoặc đã gần sắp xếp.

1- Bubble sort (Tăng dần)

1.1- Code

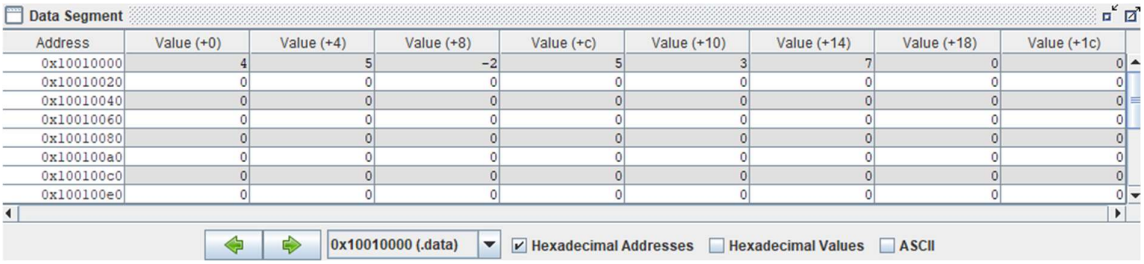
```
.data
    A: .word 4, 5, -2, 5, 3, 7
    Aend: .word
.text
    la $a0, A
    la $a1, Aend
    li $s0, 0       # count = 0 (count la bien dem phan tu)
    li $s1, -1      # i = -1 (i trong loopi)

DemPhanTu:  beq  $a1, $a0, Size     # So sanh dia chi hien tai trong a1 voi dia chi co so cua mang A
            addi $a1, $a1, -4       # dia chi a1 giam de den tung dia chi cua tung phan tu trong mang
            addi $s0, $s0, 1        # So luong phan tu tang thêm 1
            j    DemPhanTu
Size:   addi $t0, $s0, -1       # t0 = So luong phan tu cua mang A - 1
loop1:  addi $s1, $s1, 1        # i++
        li   $s2, 0             # j = 0 (j trong loop 2)
        beq  $s1, $t0, Exit     # Neu i = size - 1 thì thoat
loop2:  sub  $t2, $t0, $s1      # t2 = (size - 1) - i
        beq  $s2, $t2, loop1    # Neu j = (size - 1) - i thi nhay den loop1
if_swap: sll $t3, $s2, 2        # Tính offset cua dia chi A[j]
        add  $s3, $a0, $t3      # Tính dia chi A[j]
        lw   $v0, 0($s3)        # Load giá tri A[j]
        addi $s3, $s3, 4        # Tính dia chi cua A[j+1]
        lw   $v1, 0($s3)        # Load giá tri A[j+1]
        sle  $t4, $v0, $v1      # Neu A[j] <= A[j+1] thì t4 = 1;
```

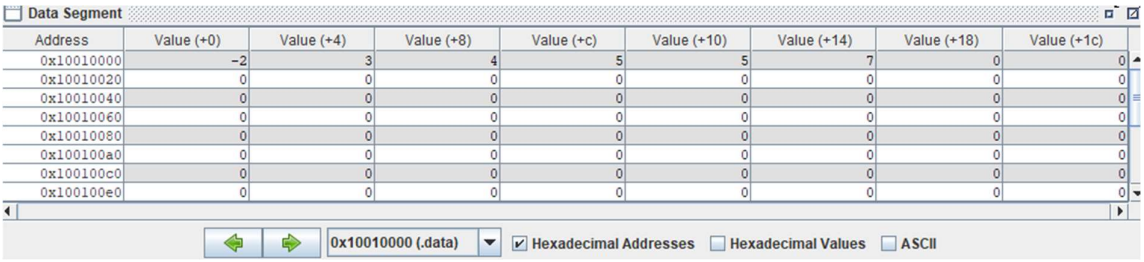
```
                                    # A[j] > A[j+1] thì t4 = 0
            beq   $t4, $zero, swap  # t4 = 0 thì nhay den swap
            addi  $s2, $s2, 1       # j++
            j     loop2
swap:       sw    $v0, 0($s3)       # Ghi A[j] vào A[j+1]
            addi  $s3, $s3, -4  # Tính dia chi cua A[j] = dia chi cua A[j+1] - 4
            sw    $v1, 0($s3)       # Ghi A[j+1] vào A[j]
            addi  $s2, $s2, 1       # j++
            j     loop2
Exit:  li $v0, 10
       syscall
```

1.2- Kết quả

Mảng A ban đầu: 4,5,-2,5,3,7



Mảng A sau khi sắp xếp: -2,3,4,5,5,7



⇨ **Kết quả trên đúng với lý thuyết**

2- Bubble sort (Giảm dần)

2.1- Code

```
.data
        A: .word 4, 5, -2, 5, 3, 7
        Aend: .word
```

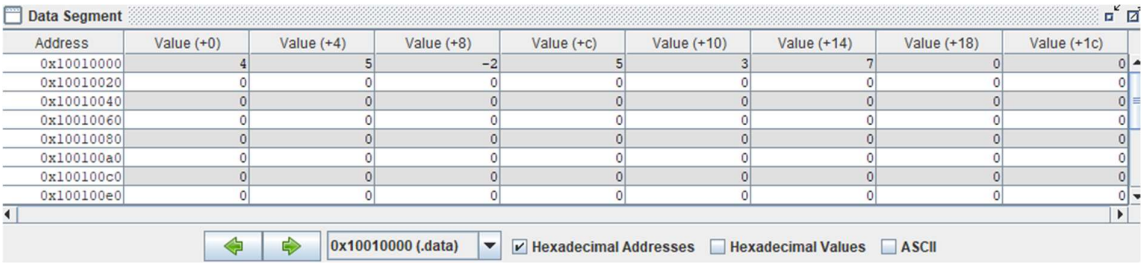
```
.text
        la      $a0, A
        la      $a1, Aend
        li      $s0, 0 # count = 0 (count la bien dem phan tu)
        li      $s1, -1         # i = -1 (i trong loopi)

DemPhanTu:      beq     $a1, $a0, Size      # So sanh dia chi hien tai trong a1 voi dia chi co so cua mang A
                addi    $a1, $a1, -4 # Dia chi a1 giam de den tung dia chi cua tung phan tu trong mang
                addi    $s0, $s0, 1         # So luong phan tu tang them 1
                j       DemPhanTu
Size:           addi    $t0, $s0, -1  # t0 = So luong phan tu cua 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 nhay den loop1
if_swap:        sll     $t3, $s2, 2         # Tính offset cua dia chi A[j]
                add     $s3, $a0, $t3       # Tính dia chi A[j]
                lw      $v0, 0($s3)         # Load giá tri A[j]
                addi    $s3, $s3, 4         # Tính dia chi cua A[j+1]
                lw      $v1, 0($s3)         # Load giá tri A[j+1]
                sle     $t4, $v1, $v0       # Neu A[j+1] <= A[j] thì t4 = 1;
                                            # A[j+1] > A[j] thì t4 = 0
                beq     $t4, $zero, swap    # t4 = 0 thì nhay den swap
                addi    $s2, $s2, 1         # j++
                j       loop2
```

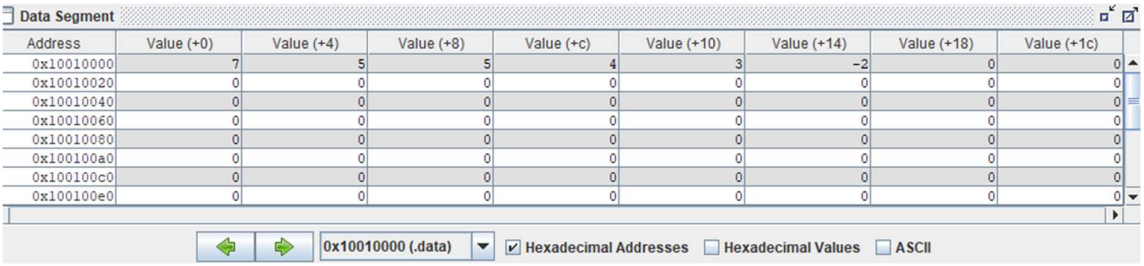
```
swap:   sw   $v0, 0($s3)        # Ghi A[j] vào A[j+1]
        addi $s3, $s3, -4  # Tính dia chi cua A[j] = dia chi cua A[j+1] - 4
        sw   $v1, 0($s3)        # Ghi A[j+1] vào A[j]
        addi $s2, $s2, 1        # j++
        j    loop2
Exit:   li $v0, 10
        syscall
```

2.2- Kết quả

Mảng A ban đầu: 4,-2,5,3,5,7



Mảng A sau khi sắp xếp: 7,5,5,4,3,-2



⇨ **Kết quả trên đúng với lý thuyết**

Assignment 4

Insertion sort là một thuật toán sắp xếp đơn giản, sắp xếp danh sách bằng cách chèn phần tử vào vị trí thích hợp trong danh sách con đã được sắp xếp trước đó.

Ý tưởng của insertion sort như sau:

1. Bắt đầu với phần tử đầu tiên của danh sách, coi như rằng danh sách con đầu tiên chỉ chứa phần tử đó.
2. Lặp lại cho đến khi tất cả các phần tử được sắp xếp.

3. Trong mỗi lần lặp, chọn một phần tử trong danh sách chưa được sắp xếp và chèn nó vào vị trí đúng trong danh sách con đã được sắp xếp trước đó.
4. Sau khi chèn, danh sách con đã được sắp xếp được mở rộng một phần tử.

Thuật toán insertion sort hoạt động với tốc độ $O(n^2)$ trong trường hợp xấu nhất. Tuy nhiên, nó có thể hoạt động nhanh hơn các thuật toán sắp xếp khác cho các danh sách nhỏ hoặc đã gần sắp xếp.

1- Insertion sort (Tăng dần)

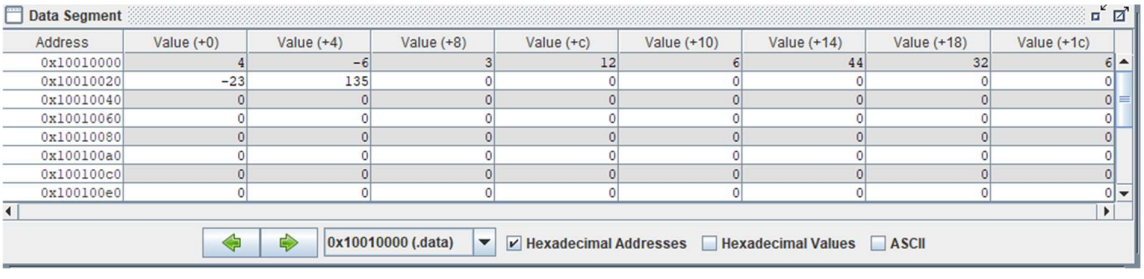
1.1- Code

```
.data
    A: .word 4, -6, 3, 12, 6, 44, 32, 6, -23, 135
    Aend: .word
.text
    la      $a0, A
    la      $a1, Aend
    li      $s0, 0 # count = 0 (count la bien dem phan tu)
    li      $s1, 0 # key = 0
    li      $s2, 0 # j = 0
    li      $s3, 1 # i = 1
DemPhanTu:      beq     $a1, $a0, Loop      # So sanh dia chi hien tai trong a1 voi dia chi co so cua mang A
                addi    $a1, $a1, -4 # Dia chi a1 giam de den tung dia chi cua tung phan tu trong mang
                addi    $s0, $s0, 1         # So luong phan tu tang thêm 1
                j       DemPhanTu
Loop: beq     $s3, $s0, Exit        # Neu i = So luong phan tu có trong mang thì thoát
                sll     $t0, $s3, 2         # Tính Offset cua dia chi A[i]
                add     $s4, $a0, $t0       # Tính dia chi cua A[i]
                lw      $s1, 0($s4)         # Load giá tri A[i] = key
```

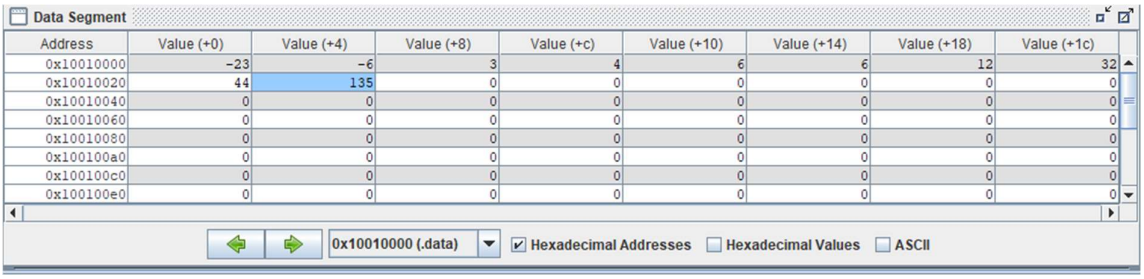
```
        addi  $s2, $s3, -1  # j = i - 1
While:  slt   $t1, $s2, $zero     # Neu j >= 0 thì t1 = 0
        sll   $t0, $s2, 2         # Tính offset cua dia chi A[j]
        add   $s5, $a0, $t0       # Tính dia chi cua A[j]
        lw    $t3, 0($s5)         # Load giá tri A[j] = thanh ghi t3
        sle   $t4, $t3, $s1# Neu key >= t3 thì t4 = 0
        add   $t1, $t1, $t4
        bne   $t1, $zero,  loop_continue # Neu t1 = 0 thì dung while
        addi  $s5, $s5, 4         # Tính dia chi cua A[j+1]
        sw    $t3, 0($s5)         # Ghi giá tri A[j] vào A[j+1]
        addi  $s2, $s2, -1  # j = j - 1
        j     While
loop_continue:
        addi  $s5, $s5, 4         # Tính ??a ch? c?a A[j+1]
        sw    $s1, 0($s5)         # Ghi giá tr? key vào A[j+1]
        addi  $s3, $s3, 1         # i++
        j     Loop
Exit:   li $v0, 10
        syscall
```

1.2- Kết quả

Mảng A ban đầu: 4, -6, 3, 12, 6, 44, 32, 6, -23, 135



Mảng A sau khi sắp xếp: -23,-6, 3, 4, 6, 6, 12, 32, 44, 135



⇨ **Kết quả trên đúng với lý thuyết**

2- Insertion sort (Giảm dần)

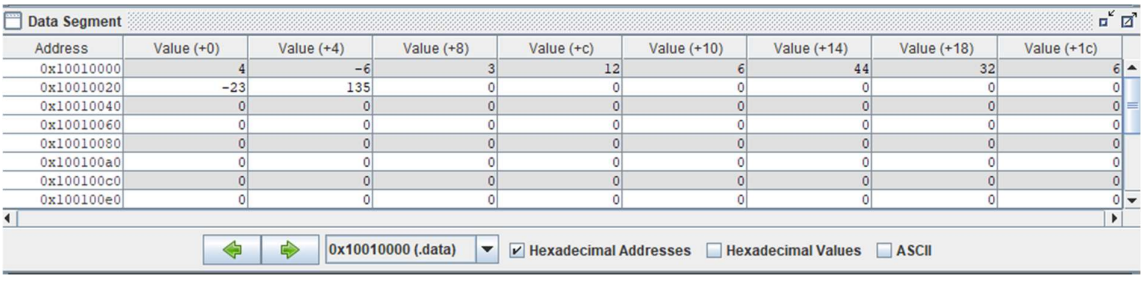
2.1- Code

```
.data
    A: .word 4, -6, 3, 12, 6,  44, 32, 6, -23, 135
    Aend: .word
.text
    la      $a0, A
    la      $a1, Aend
    li      $s0, 0 # count = 0 (count la bien dem phan tu)
    li      $s1, 0 # key = 0
    li      $s2, 0 # j = 0
    li      $s3, 1 # i = 1
DemPhanTu:      beq     $a1, $a0, Loop      # So sanh dia chi hien tai trong a1 voi dia chi co so cua mang A
                addi    $a1, $a1, -4 # Dia chi a1 giam de den tung dia chi cua tung phan tu trong mang
                addi    $s0, $s0, 1         # So luong phan tu tang thêm 1
                j       DemPhanTu
Loop: beq       $s3, $s0, Exit      # Neu i = So luong phan tu có trong mang thì thoát
                sll     $t0, $s3, 2         # Tính Offset cua dia chi A[i]
                add     $s4, $a0, $t0       # Tính dia chi cua A[i]
                lw      $s1, 0($s4)         # Load giá tri A[i] = key
                addi    $s2, $s3, -1  # j = i - 1
```

```
While:    slt   $t1, $s2, $zero     # Neu j >= 0 thì t1 = 0
          sll   $t0, $s2, 2         # Tính offset cua dia chi A[j]
          add   $s5, $a0, $t0       # Tính dia chi cua A[j]
          lw    $t3, 0($s5)         # Load giá tri A[j] = thanh ghi t3
          sle   $t4, $s1, $t3# Neu key >= t3 thì t4 = 0
          add   $t1, $t1, $t4
          bne   $t1, $zero,  loop_continue # Neu t1 = 0 thì dung while
          addi  $s5, $s5, 4         # Tính dia chi cua A[j+1]
          sw    $t3, 0($s5)         # Ghi giá tri A[j] vào A[j+1]
          addi  $s2, $s2, -1  # j = j - 1
          j     While
loop_continue:
          addi  $s5, $s5, 4         # Tính dia chi cua A[j+1]
          sw    $s1, 0($s5)         # Ghi giá tri key vào A[j+1]
          addi  $s3, $s3, 1         # i++
          j Loop
Exit:  li $v0, 10
       syscall
```

2.2- Kết quả

Mảng A ban đầu: 4, -6, 3, 12, 6, 44, 32, 6, -23, 135



Mảng A sau khi sắp xếp: 135, 44, 32, 12, 6, 6, 4, 3, -6, -23

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	135	44	32	12	6	6	4	3
0x10010020	-6	-23	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

⇨ **Kết quả trên đúng với lý thuyết**

~THE END~