

ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



BÁO CÁO

Bài tập thực hành tuần 6

Học phần: Thực hành kiến trúc máy tính

Giảng viên hướng dẫn: Lê Bá Vui

Sinh viên thực hiện: Phạm Huy Cảnh - 20194490

Mã lớp: 130938

Hà Nội, tháng 5 năm 2022

1. Assignment 1:

D:\Courses\20212\ThucHanhKienTrucMayTinh\Assignments\Week5\HomeAssignment1 - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	4
\$v1	3	6
\$a0	4	268500992
\$a1	5	5
\$a2	6	0
\$a3	7	0
\$t0	8	5
\$t1	9	4
\$t2	10	16
\$t3	11	268501000
\$t4	12	-2
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$t8	16	0
\$t9	17	0
\$s0	18	0
\$s1	19	0
\$s2	20	0
\$s3	21	0
\$s4	22	0
\$s5	23	0
\$s6	24	0
\$s7	25	0
\$s8	26	0
\$s9	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
\$PC		4194324
\$HI		0
\$LO		0

Mars Messages Run I/O

Go: running HomeAssignment1

Clear

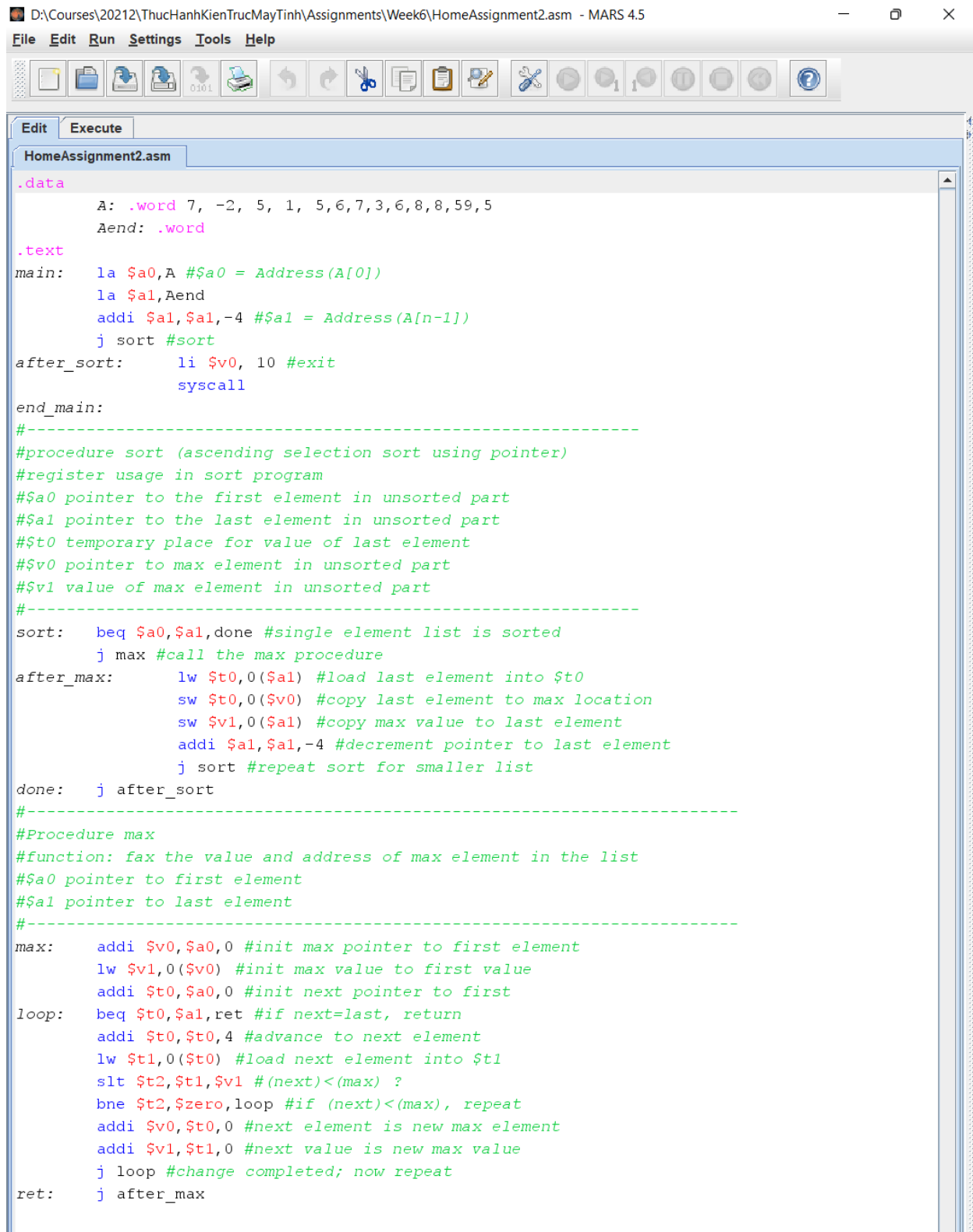
Go: execution terminated by user.

Kết quả: Sum $\sum_{i=0}^5 i = 6$

⇒ Kết quả này đúng với tính toán

2. Assignment 2: Selection Sort

I. Selection sort (Tăng dần):



The screenshot shows the MARS MIPS simulator window. The title bar indicates the file path: D:\Courses\20212\ThucHanhKienTrucMayTinh\Assignments\Week6\HomeAssignment2.asm - MARS 4.5. The menu bar includes File, Edit, Run, Settings, Tools, and Help. The toolbar contains various icons for file operations and simulation control. The main window is titled 'HomeAssignment2.asm' and contains the following assembly code:

```
.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:
#-----
#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 $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
#-----
#Procedure max
#function: fax the value and address of max element in the list
# $a0 pointer to first element
# $a1 pointer to last element
#-----
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
```

Kết quả được hiển thị trong Data Segment:

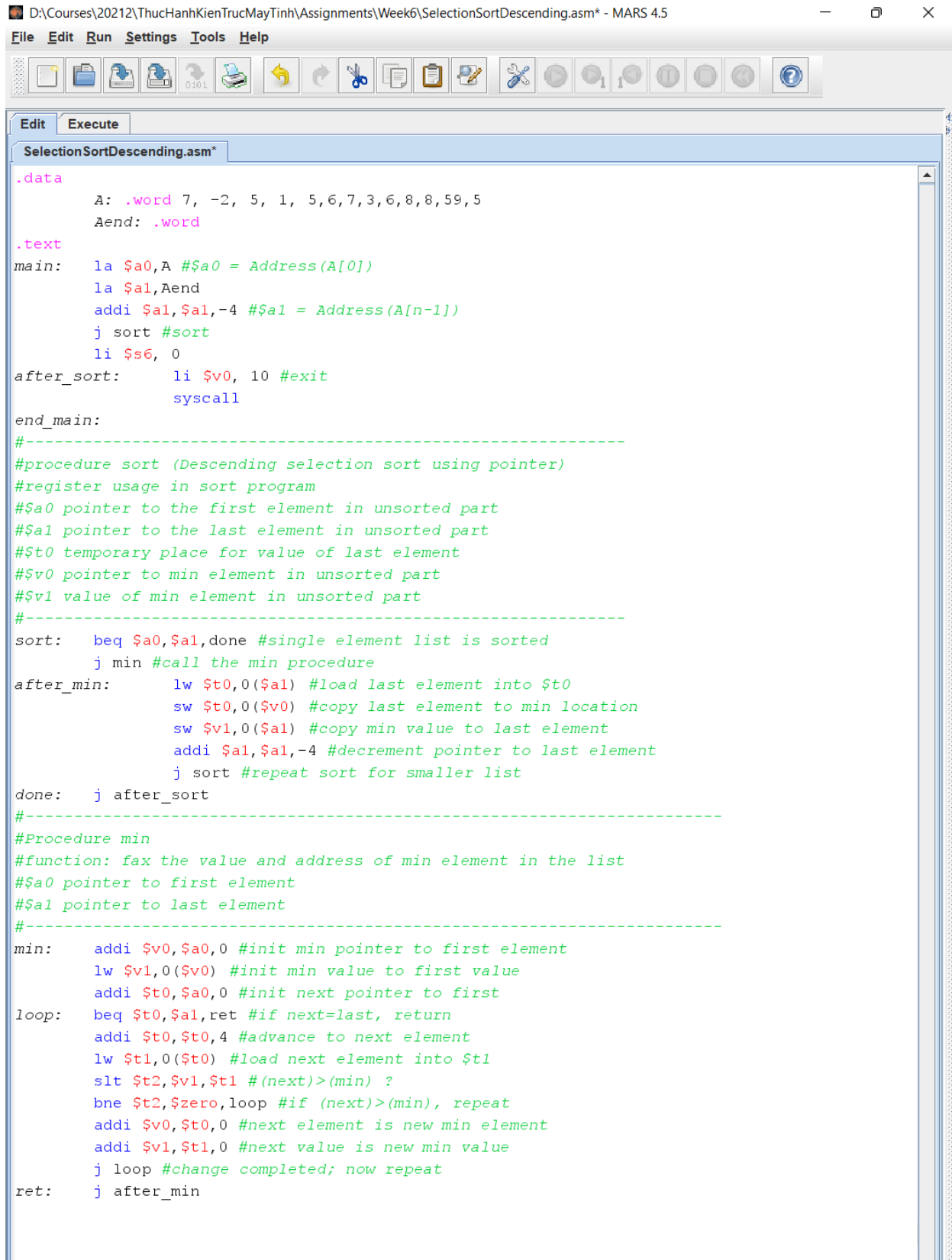
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	7	-2	5	1	5	6	7	3
0x10010020	6	8	8	59	5	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

Mảng A ban đầu gồm các phần tử: 7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5

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
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

Mảng A sau khi chạy xong chương trình: -2, 1, 3, 5, 5, 5, 6, 6, 7, 7, 8, 8, 59

II. Selection sort (Giảm dần):



The screenshot shows the MARS 4.5 MIPS assembler interface. The title bar indicates the file path: D:\Courses\20212\ThucHanhKienTrucMayTinh\Assignments\Week6\SelectionSortDescending.asm* - MARS 4.5. The menu bar includes File, Edit, Run, Settings, Tools, and Help. The toolbar contains icons for file operations, execution, and debugging. The main window displays the assembly code for SelectionSortDescending.asm.

```
.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
        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
```

Kết quả được hiển thị trong 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
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

Mảng A ban đầu gồm các phần tử: 7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	59	8	8	7	7	6	6	5
0x10010020	5	5	3	1	-2	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
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

Mảng A sau khi chạy xong chương trình: 59, 8, 8, 7, 7, 6, 6, 5, 5, 5, 3, 1, -2

3. Assignment 3: Bubble Sort

I. Bubble sort (Tăng dần):

```
.data
    A: .word 4, 5, -2, 5, 3, 45
    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 loop1)

DemPhanTu:      beq $a1, $a0, Size      # So sanh địa chỉ hiện tại trong a1 với
                                                # địa chỉ cơ sở của mảng A
                addi $a1, $a1, -4      # Địa chỉ a1 giảm để đến từng địa chỉ của từng
                                                # phần tử trong mảng
                addi $s0, $s0, 1      # Số lượng phần tử tăng thêm 1
                j DemPhanTu

Size:           addi $t0, $s0, -1      # t0 = Số lượng phần tử của mảng A - 1
loop1:          addi $s1, $s1, 1      # i++
                li $s2, 0             # j = 0 (j trong loop2)
                beq $s1, $t0, Exit     # Nếu i = size - 1 thì thoát
loop2:          sub $t2, $t0, $s1      # t2 = (size - 1) - i
                beq $s2, $t2, loop1    # Nếu j = (size - 1) - i thì nhảy đến loop1
if_swap:        sll $t3, $s2, 2       # Tính offset của địa chỉ A[j]
                add $s3, $a0, $t3     # Tính địa chỉ A[j]
                lw $v0, 0($s3)        # Load giá trị A[j]
                addi $s3, $s3, 4      # Tính địa chỉ của A[j+1]
                lw $v1, 0($s3)        # Load giá trị A[j+1]
                sle $t4, $v0, $v1     # Nếu A[j] <= A[j+1] thì t4 = 1;
                                                # A[j] > A[j+1] thì t4 = 0
                beq $t4, $zero, swap   # t4 = 0 thì nhảy đến 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 địa chỉ của A[j] = Địa chỉ của 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
```

Kết quả được hiển thị trong Data Segment:

Mảng A ban đầu gồm các phần tử: 4, 5, -2, 5, 3, 45

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	4	5	-2	5	3	45	0	0
0x10010020	0	0	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
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

Mảng A sau khi chạy xong chương trình: -2, 3, 4, 5, 5, 45

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-2	3	4	5	5	45	0	0
0x10010020	0	0	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
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

II. Bubble sort (Giảm dần):

```
D:\Courses\20212\ThucHanhKienTrucMayTinh\Assignments\Week6\BubbleSortDescending.asm - MARS 4.5
File Edit Run Settings Tools Help

BubbleSortDescending.asm

.data
    A: .word 4, 5, -2, 5, 3, 45
    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 địa chỉ hiện tại trong a1 với
                                                         # địa chỉ cơ sở của mảng A
    addi $a1, $a1, -4      # Địa chỉ a1 giảm để đến từng địa chỉ của từng
                                                         # phần tử trong mảng
    addi $s0, $s0, 1      # Số lượng phần tử tăng thêm 1
    j DemPhanTu

Size:    addi $t0, $s0, -1      # t0 = Số lượng phần tử của mảng A - 1
loop1:   addi $s1, $s1, 1      # i++
    li $s2, 0      # j = 0 (j trong loop2)
    beq $s1, $t0, Exit      # Nếu i = size - 1 thì thoát
loop2:   sub $t2, $t0, $s1      # t2 = (size - 1) - i
    beq $s2, $t2, loop1      # Nếu j = (size - 1) - i thì nhảy đến loop1
if_swap: sll $t3, $s2, 2      # Tính offset của địa chỉ A[j]
    add $s3, $a0, $t3      # Tính địa chỉ A[j]
    lw $v0, 0($s3)      # Load giá trị A[j]
    addi $s3, $s3, 4      # Tính địa chỉ của A[j+1]
    lw $v1, 0($s3)      # Load giá trị A[j+1]
    sle $t4, $v1, $v0      # Nếu A[j+1] <= A[j] thì t4 = 1;
                                                         # A[j+1] > A[j] thì t4 = 0
    beq $t4, $zero, swap    # t4 = 0 thì nhảy đến 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 địa chỉ của A[j] = Địa chỉ của 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
```

Kết quả được hiển thị trong Data Segment:

Mảng A ban đầu gồm các phần tử: 4, 5, -2, 5, 3, 45

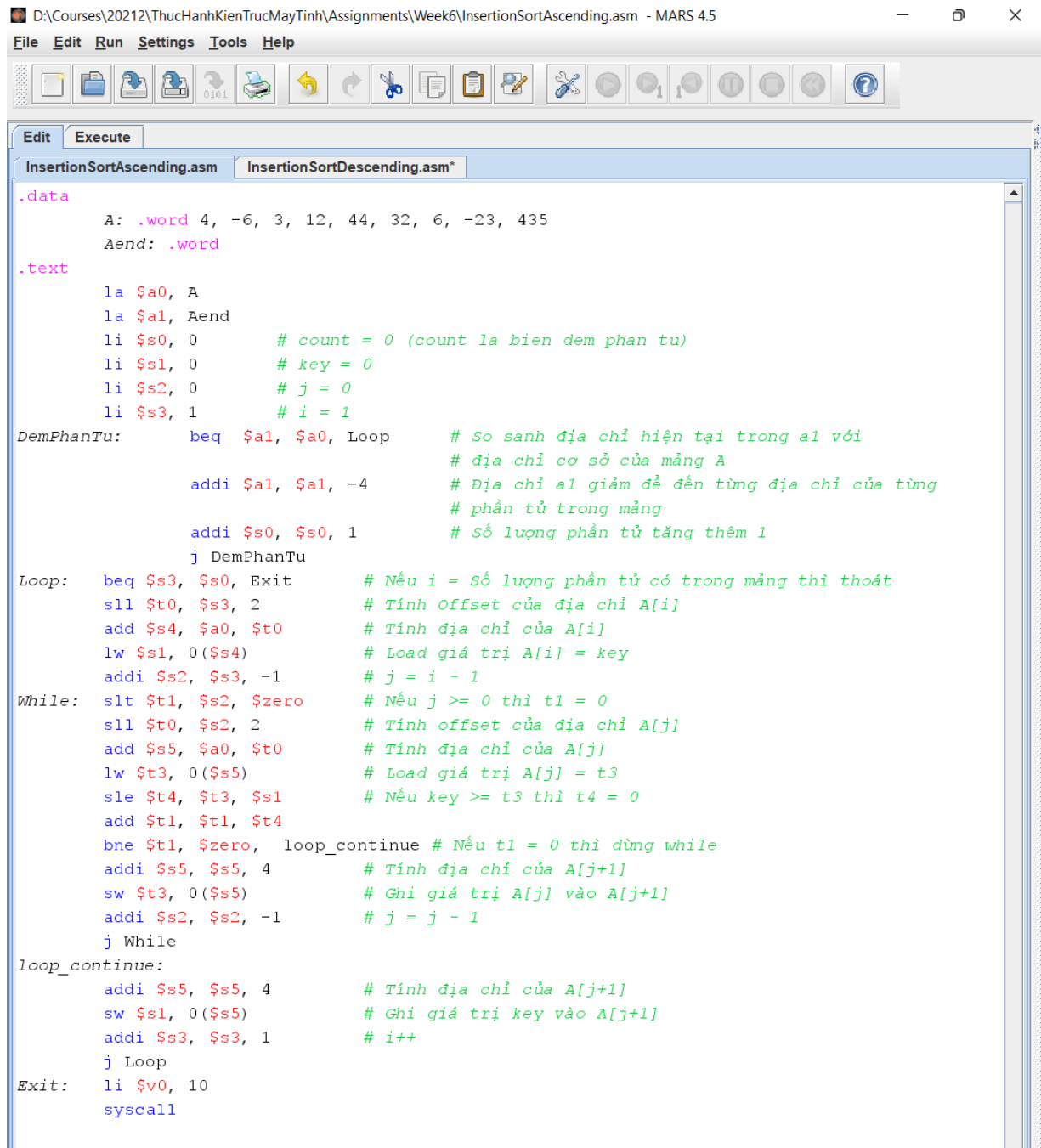
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	4	5	-2	5	3	45	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

Mảng A sau khi chạy xong chương trình: 45, 5, 5, 4, 3, -2

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	45	5	5	4	3	-2	0	0
0x10010020	0	0	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
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

4. Assignment 4: Insertion Sort

I. Insertion sort (Tăng dần):



```
.data
    A: .word 4, -6, 3, 12, 44, 32, 6, -23, 435
    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 địa chỉ hiện tại trong a1 với
                                         # địa chỉ cơ sở của mảng A
                addi $a1, $a1, -4      # Địa chỉ a1 giảm để đến từng địa chỉ của từng
                                         # phần tử trong mảng
                addi $s0, $s0, 1      # Số lượng phần tử tăng thêm 1
                j DemPhanTu

Loop:          beq $s3, $s0, Exit      # Nếu i = Số lượng phần tử có trong mảng thì thoát
                sll $t0, $s3, 2      # Tính Offset của địa chỉ A[i]
                add $s4, $a0, $t0      # Tính địa chỉ của A[i]
                lw $s1, 0($s4)        # Load giá trị A[i] = key
                addi $s2, $s3, -1      # j = i - 1

While:         slt $t1, $s2, $zero      # Nếu j >= 0 thì t1 = 0
                sll $t0, $s2, 2      # Tính offset của địa chỉ A[j]
                add $s5, $a0, $t0      # Tính địa chỉ của A[j]
                lw $t3, 0($s5)        # Load giá trị A[j] = t3
                sle $t4, $t3, $s1      # Nếu key >= t3 thì t4 = 0
                add $t1, $t1, $t4      # Nếu t1 = 0 thì dừng while
                bne $t1, $zero, loop_continue
                addi $s5, $s5, 4      # Tính địa chỉ của A[j+1]
                sw $t3, 0($s5)        # Ghi giá trị A[j] vào A[j+1]
                addi $s2, $s2, -1      # j = j - 1
                j While

loop_continue:
                addi $s5, $s5, 4      # Tính địa chỉ của A[j+1]
                sw $s1, 0($s5)        # Ghi giá trị key vào A[j+1]
                addi $s3, $s3, 1      # i++
                j Loop

Exit:          li $v0, 10
                syscall
```

Kết quả được hiển thị trong Data Segment:

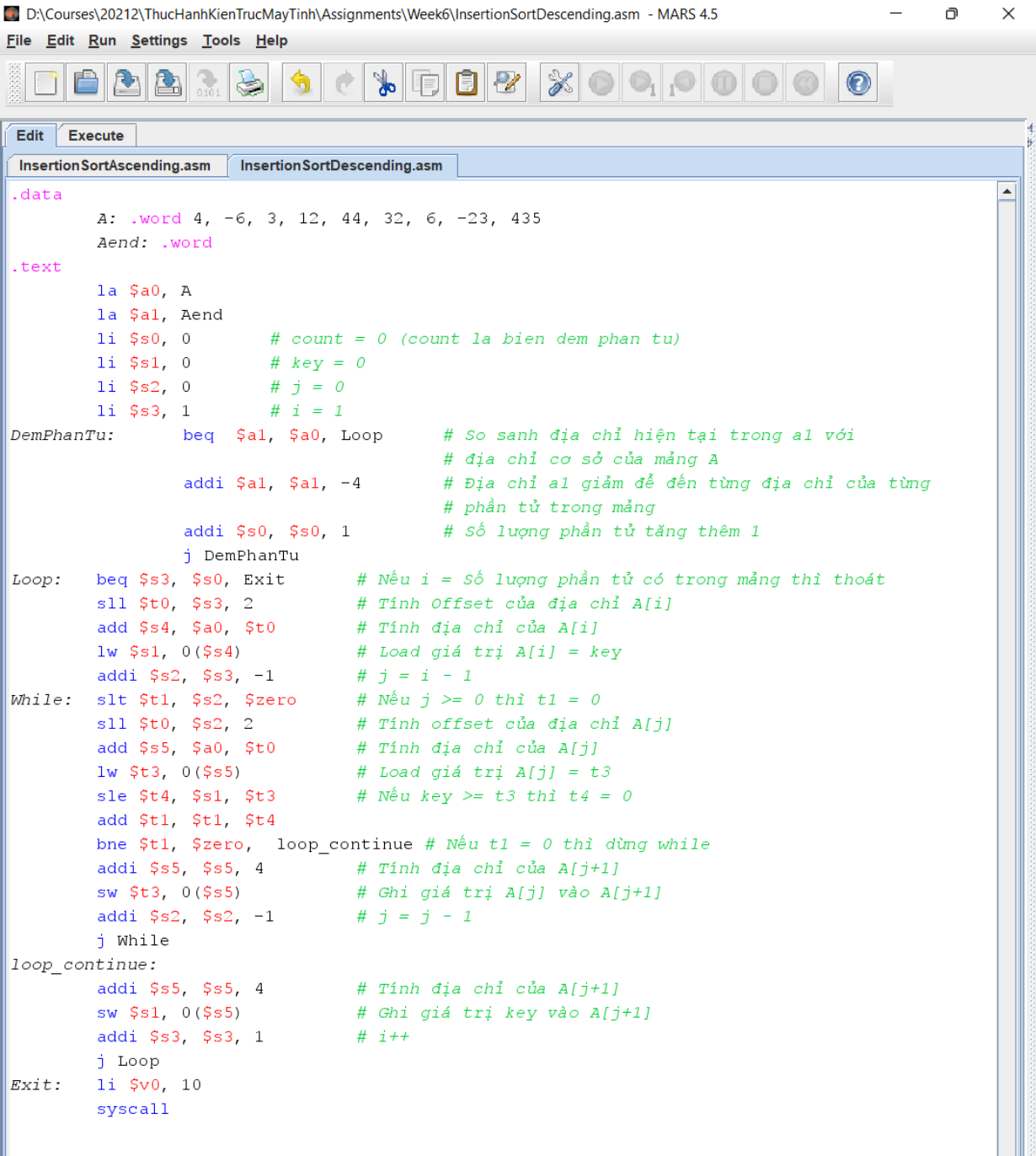
Mảng A ban đầu gồm các phần tử: 4, -6, 3, 12, 44, 32, 6, -23, 435

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	4	-6	3	12	44	32	6	-23
0x10010020	435	0	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
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

Mảng A sau khi chạy xong chương trình: -23, -6, 3, 4, 6, 12, 32, 44, 435

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-23	-6	3	4	6	12	32	44
0x10010020	435	0	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
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

II. Insertion sort (Giảm dần):



```
.data
    A: .word 4, -6, 3, 12, 44, 32, 6, -23, 435
    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 them 1
    j DemPhanTu
Loop:             beq $s3, $s0, Exit      # Nieu i = So luong phan tu co trong mang thi thoat
    sll $t0, $s3, 2      # Tinh Offset cua dia chi A[i]
    add $s4, $a0, $t0    # Tinh dia chi cua A[i]
    lw $s1, 0($s4)      # Load gia tri A[i] = key
    addi $s2, $s3, -1    # j = i - 1
While:           slt $t1, $s2, $zero      # Nieu j >= 0 thi t1 = 0
    sll $t0, $s2, 2      # Tinh offset cua dia chi A[j]
    add $s5, $a0, $t0    # Tinh dia chi cua A[j]
    lw $t3, 0($s5)      # Load gia tri A[j] = t3
    sle $t4, $s1, $t3    # Nieu key >= t3 thi t4 = 0
    add $t1, $t1, $t4
    bne $t1, $zero, loop_continue # Nieu t1 = 0 thi dung while
    addi $s5, $s5, 4      # Tinh dia chi cua A[j+1]
    sw $t3, 0($s5)      # Ghi gia tri A[j] vao A[j+1]
    addi $s2, $s2, -1    # j = j - 1
    j While
loop_continue:   addi $s5, $s5, 4      # Tinh dia chi cua A[j+1]
    sw $s1, 0($s5)      # Ghi gia tri key vao A[j+1]
    addi $s3, $s3, 1      # i++
    j Loop
Exit:           li $v0, 10
    syscall
```

Kết quả được hiển thị trong Data Segment:

Mảng A ban đầu gồm các phần tử: 4, -6, 3, 12, 44, 32, 6, -23, 435

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	4	-6	3	12	44	32	6	-23
0x10010020	435	0	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
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

Mảng A sau khi chạy xong chương trình: 435, 44, 32, 12, 6, 4, 3, -6, -23

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	435	44	32	12	6	4	3	-6
0x10010020	-23	0	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
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

Source code:

<https://drive.google.com/drive/folders/1IoxxWCIEpDWDFUGyMhBIx8QG8QsUPQ2i?usp=sharing>