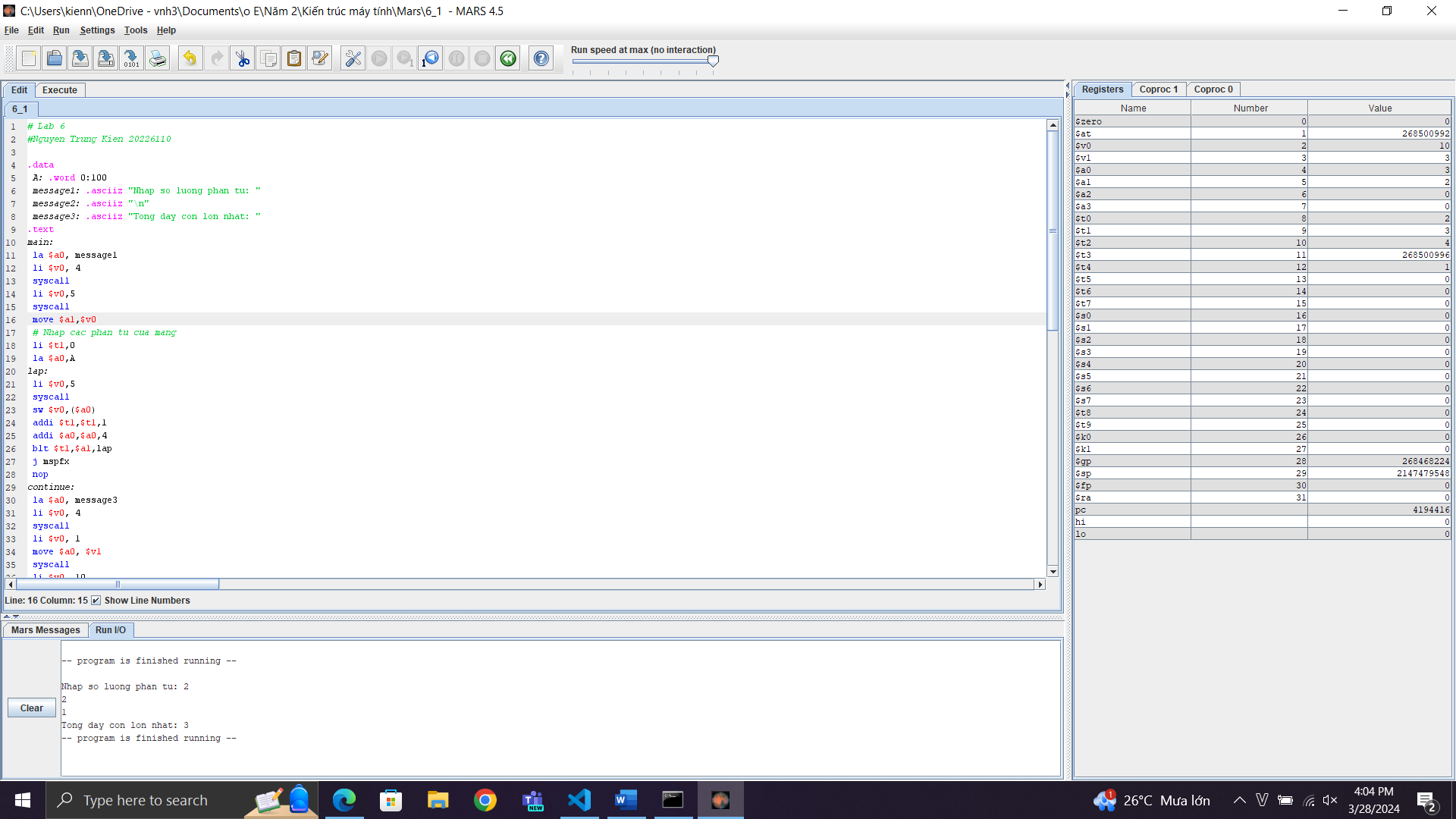
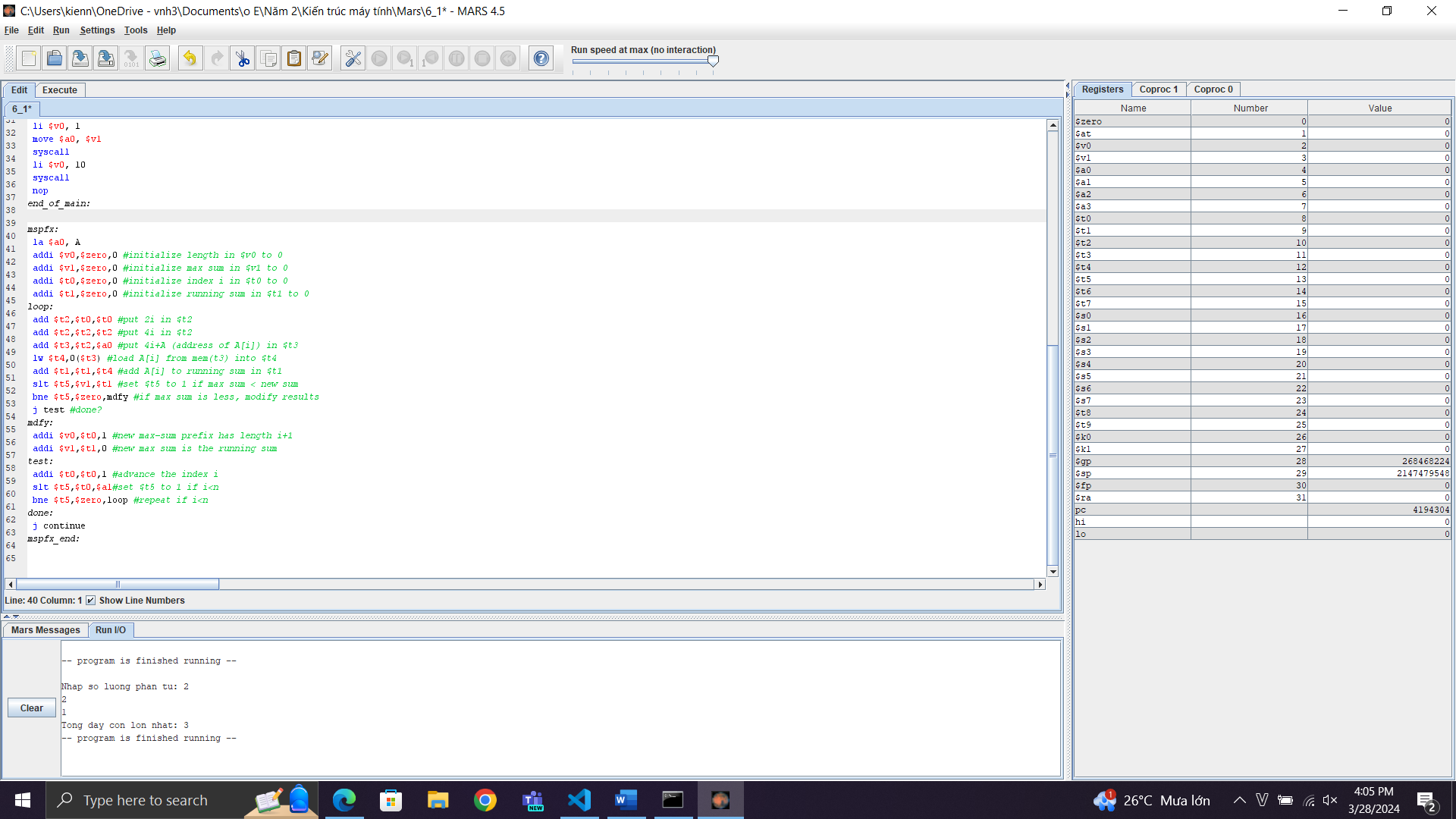
**Computer Architecture Lab Report Week 5**

**Full name: Nguyen Trung Kien**

**Student ID: 20226110**

**Assignment 1:**

***Code:***

# Lab 6

#Nguyen Trung Kien 20226110

.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

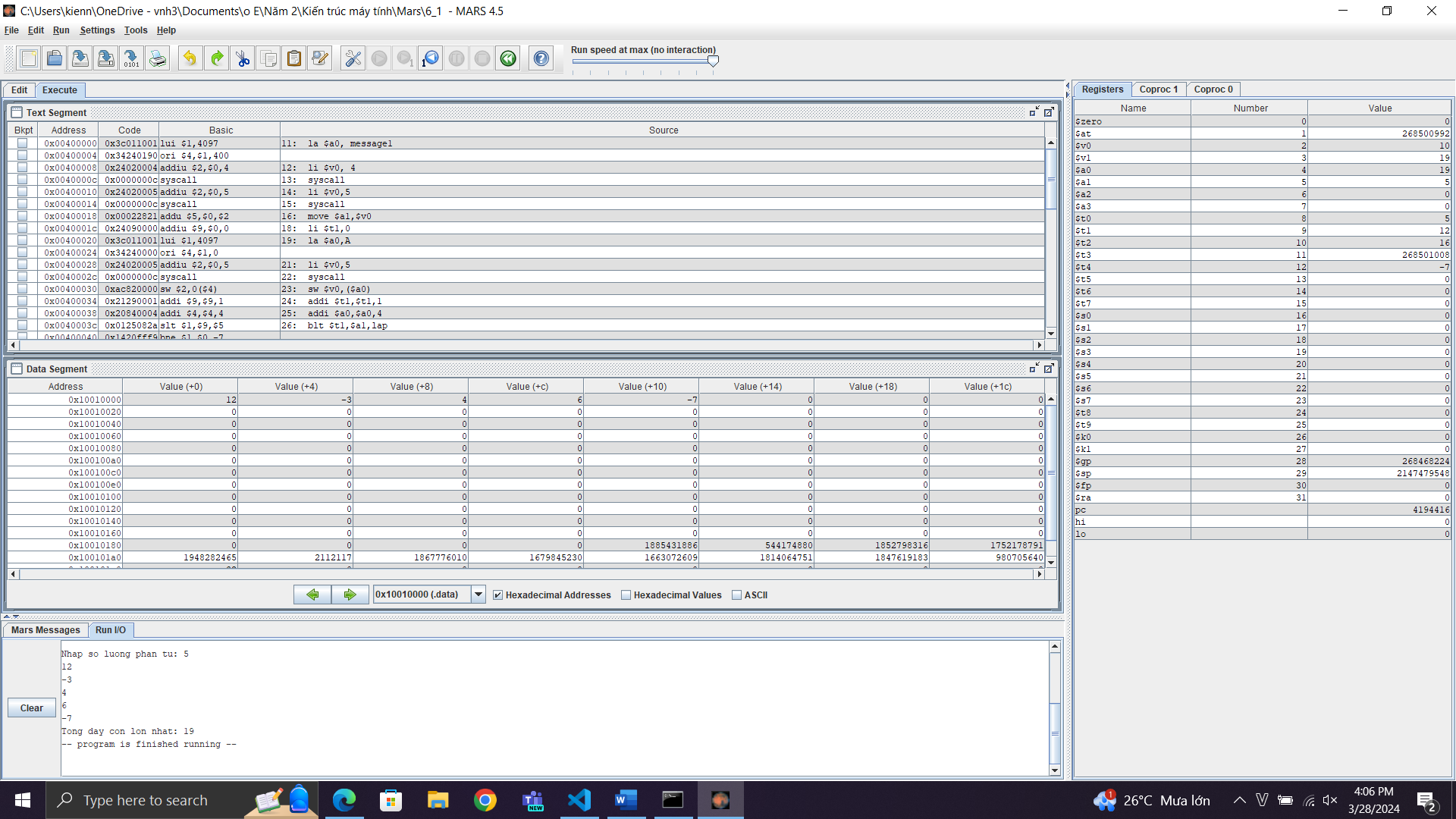
done:

j continue

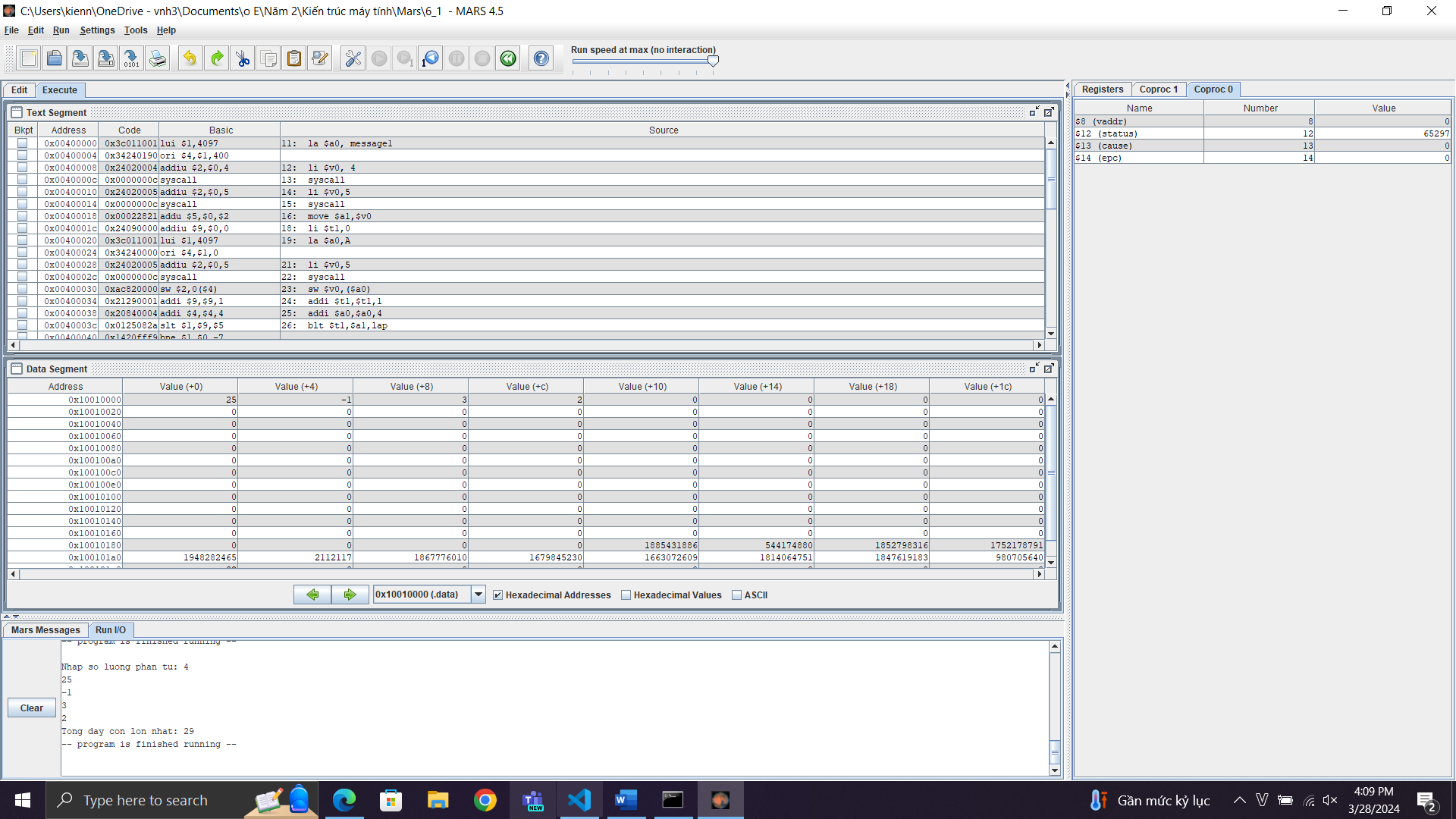
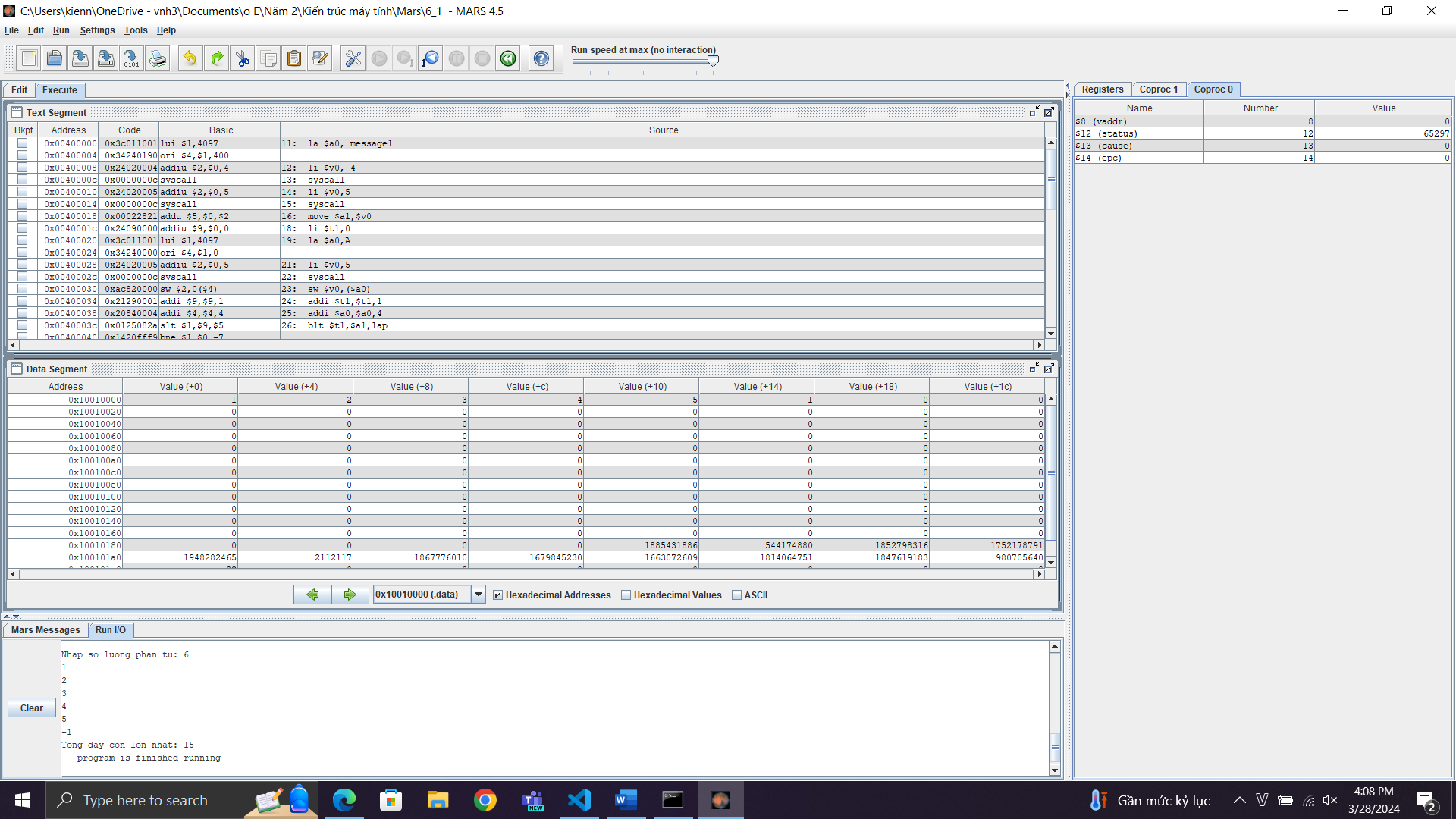
mspfx\_end:

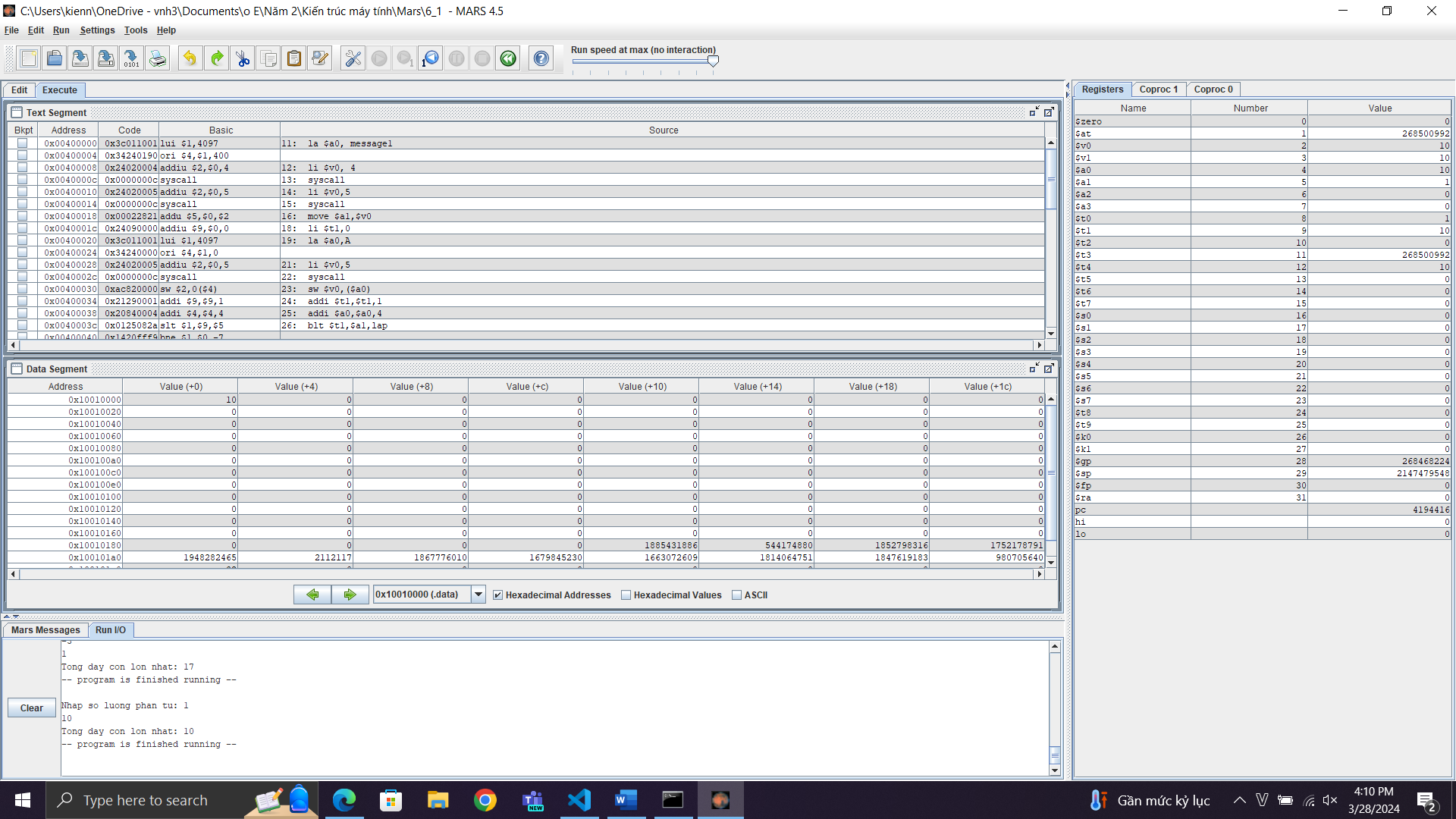
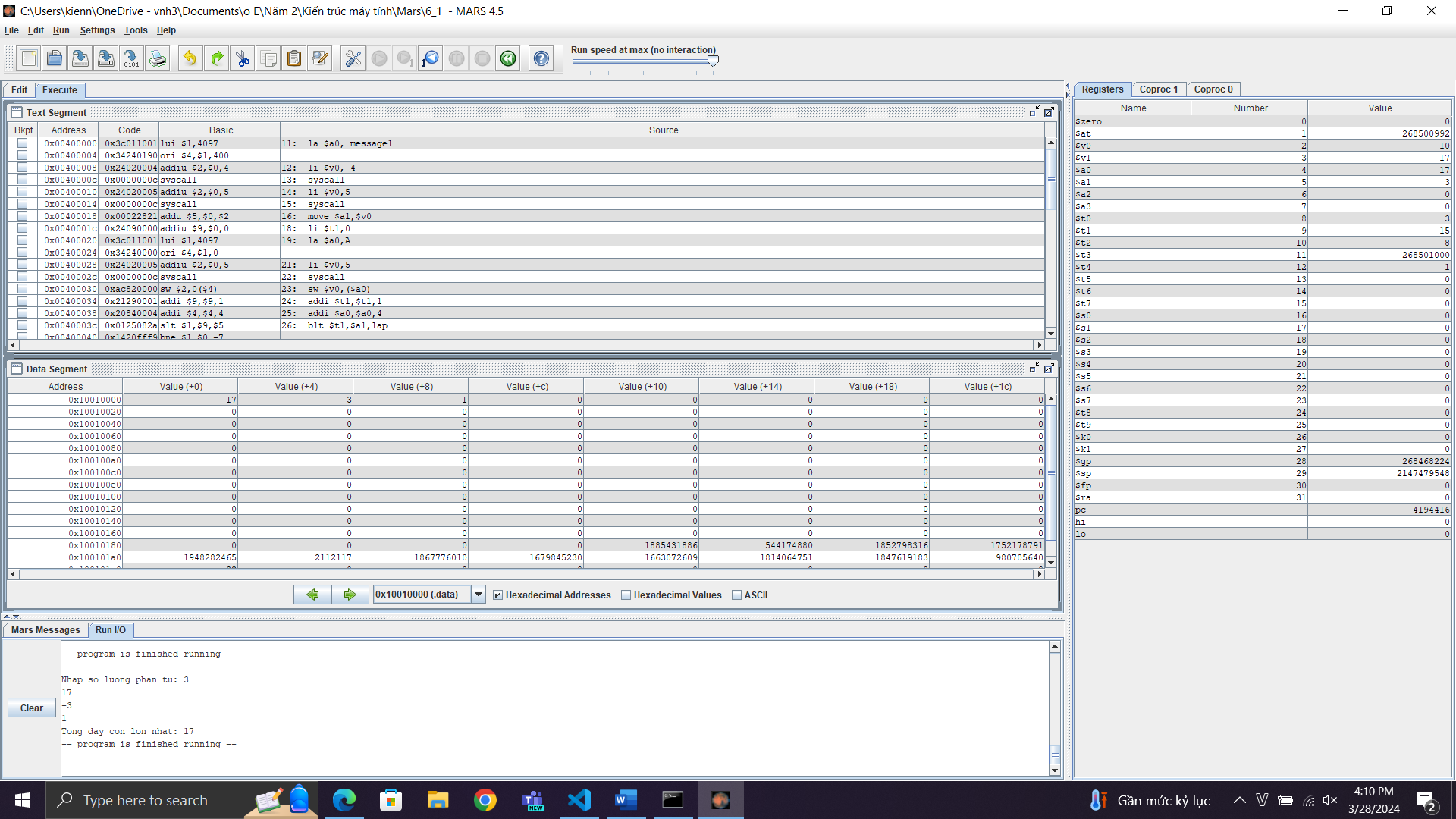
***Kết quả:***

* Ta thử nghiệm với dãy đầu tiên có 5 phần tử: 12 -3 4 6 -7

 => Đúng với mong đợi.

- Tiếp tục thử với một vài trường hợp khác: Ta nhận xét được chương trình hoạt động đúng.





**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.

***Code:***

# Lab 6

#Nguyen Trung Kien 20226110

.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

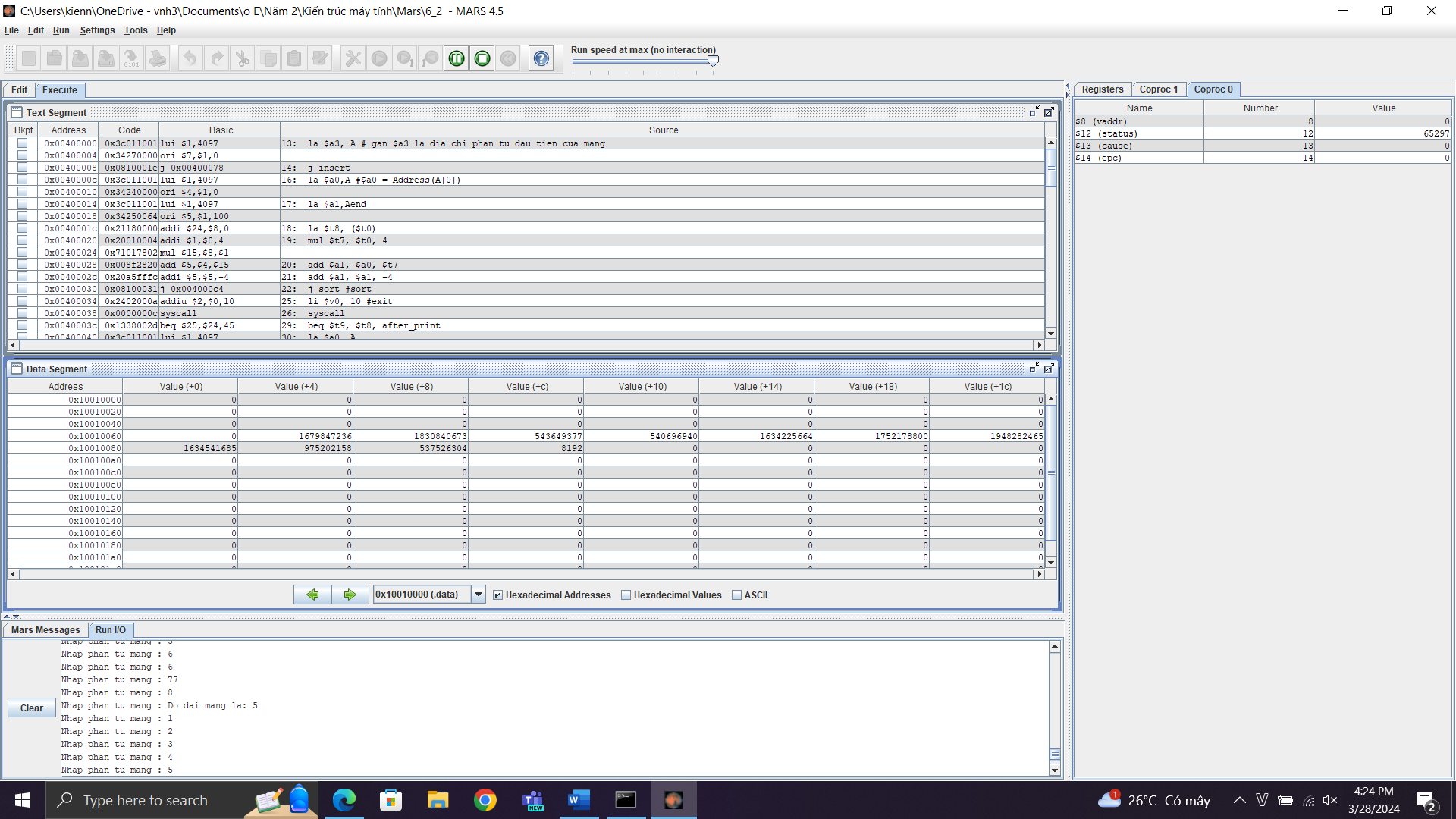
ret:

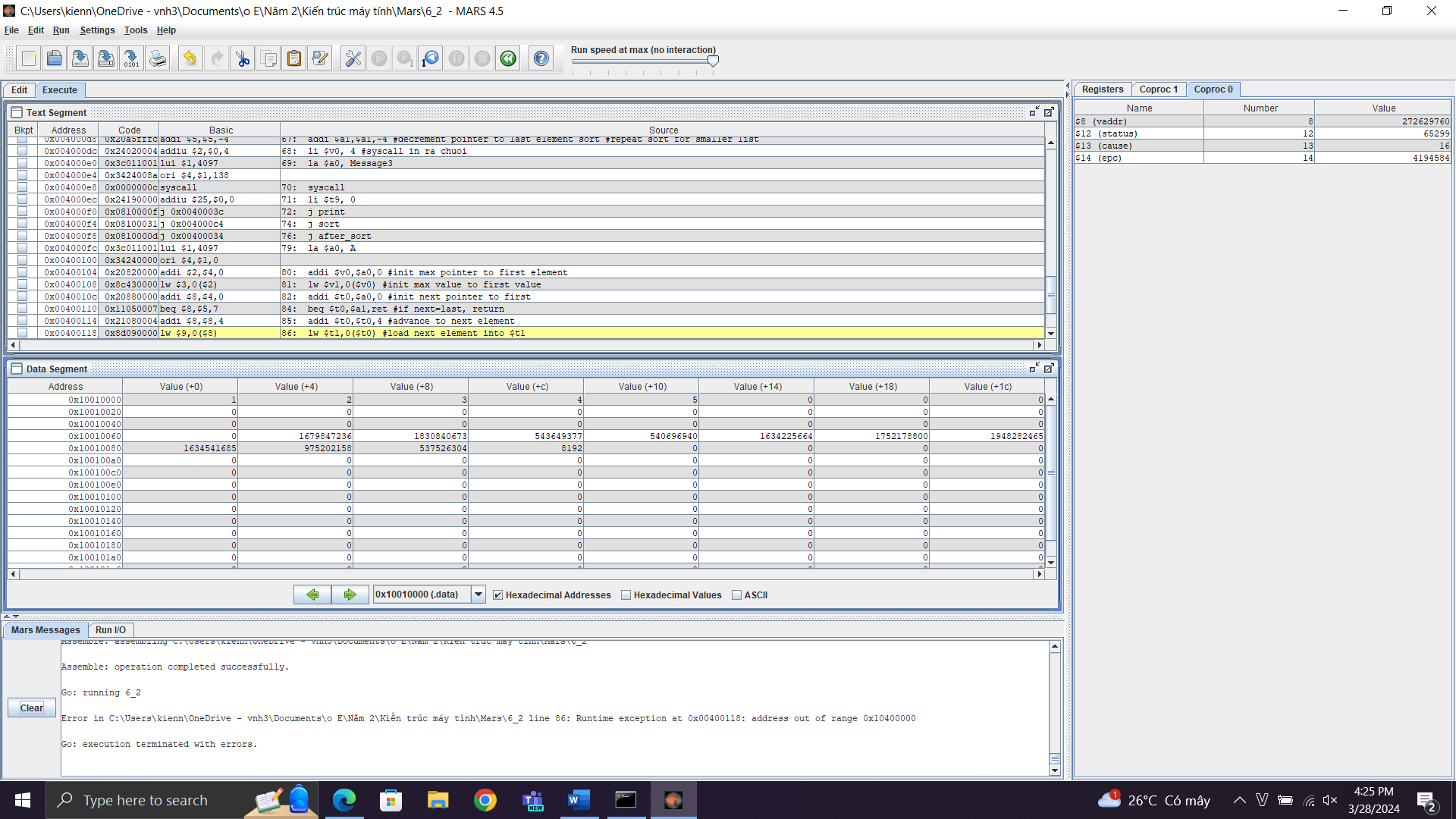
j after\_max

Kết quả:

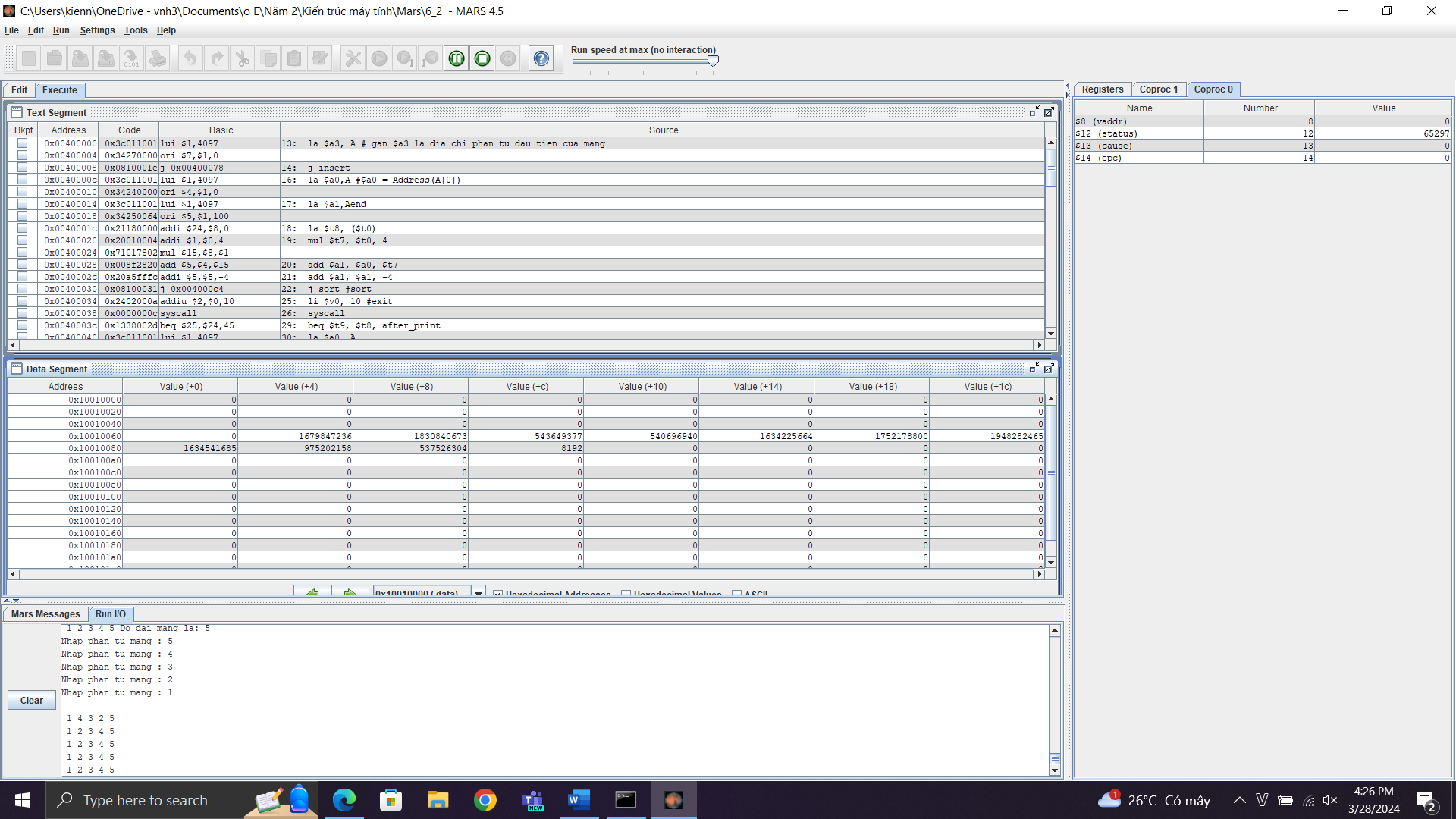
* Ta thử nghiệm các trường hợp sau:

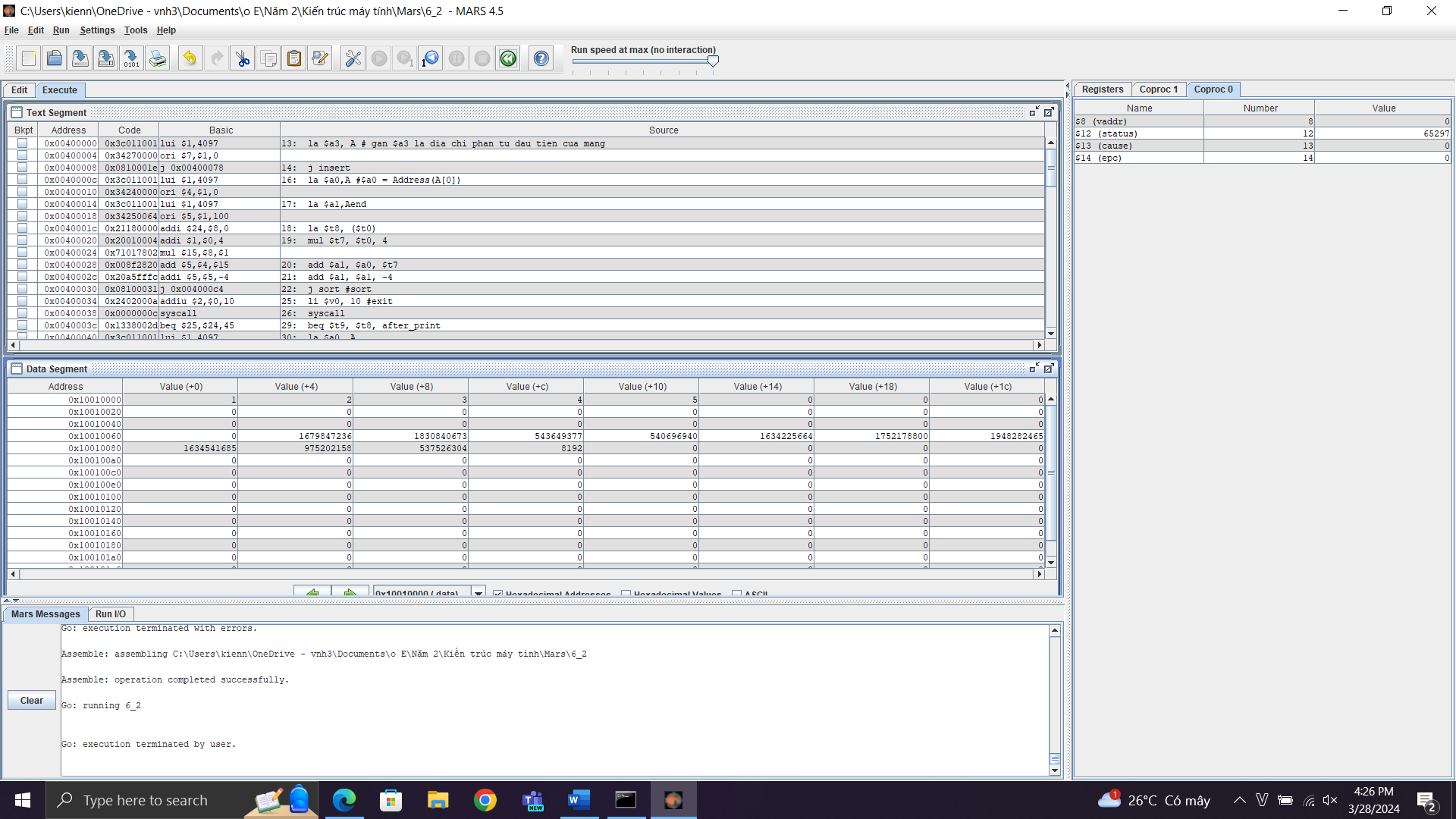
+ TH1: Dãy 1 2 3 4 5



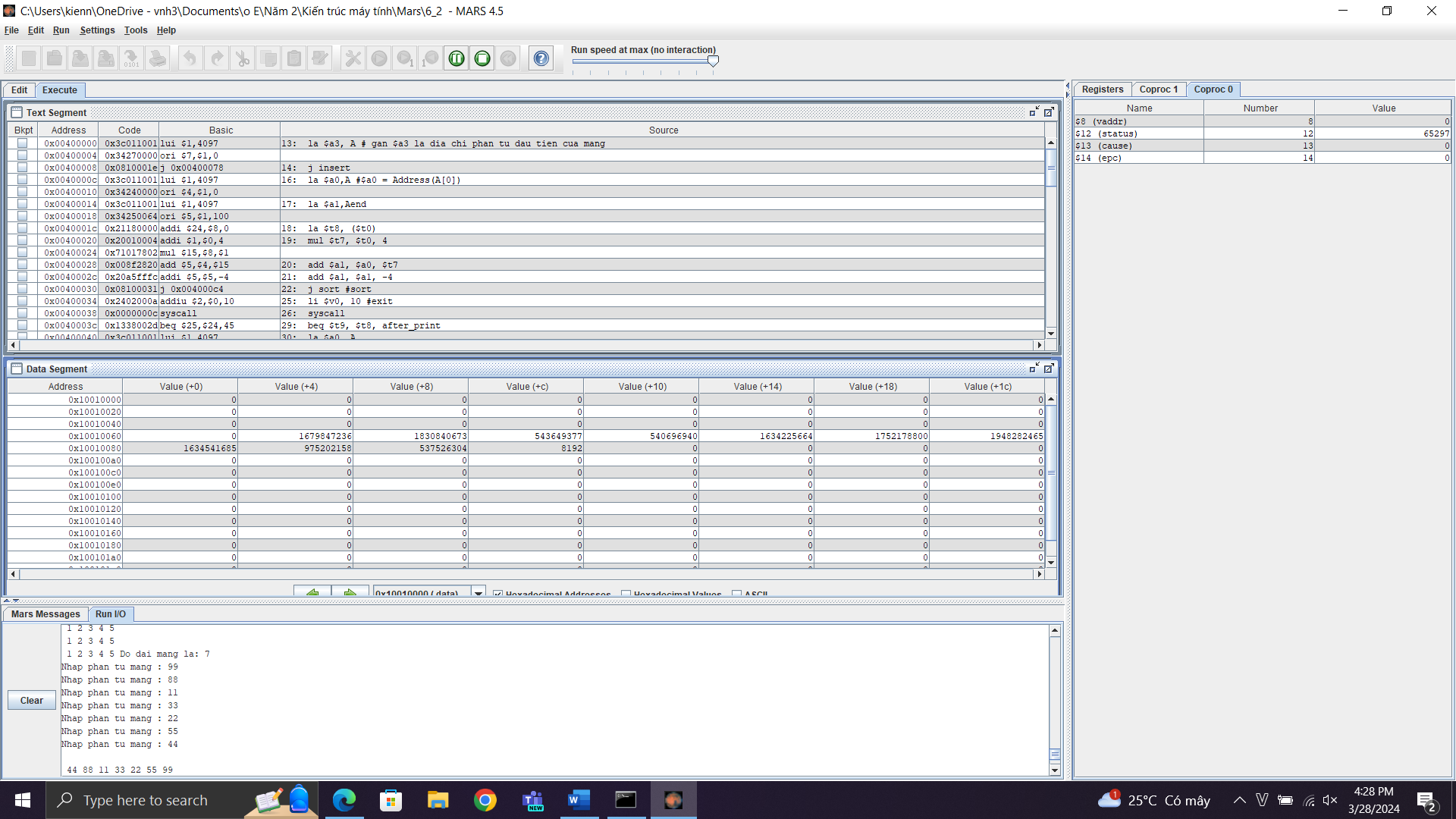


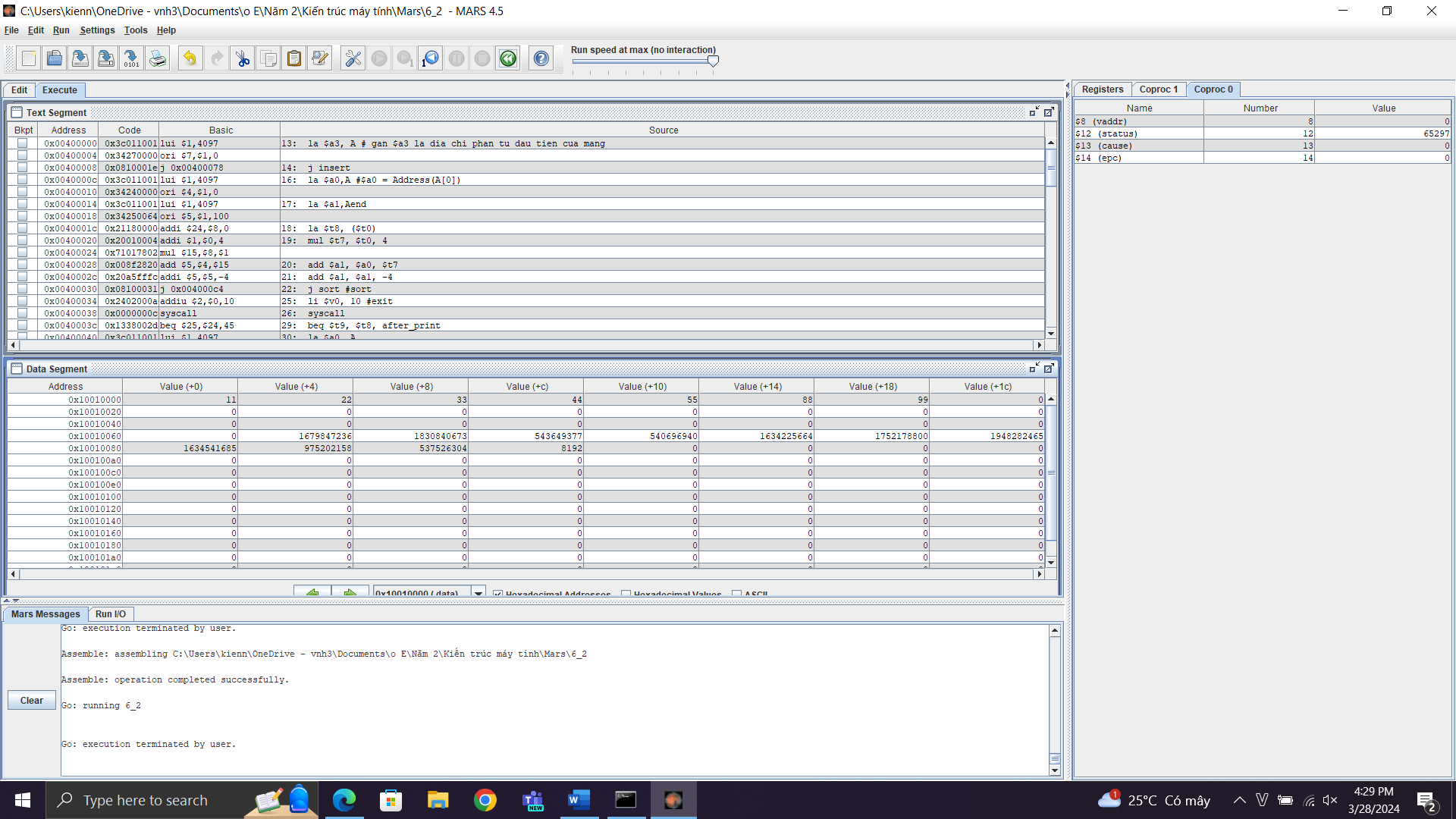
+ TH2: Dãy 5 4 3 2 1





+ TH3: Dãy 99 88 11 33 22 55 44





* Kết quả thu được 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.

***Code: (Ta nhập array sẵn và xem xét kết quả trên thanh ghi)***

.data

A: .word 28,-2,16,12,25,-9,24,12,18,-4 # Array

Aend: .word # Last element of the array

.text

la $a0, A # Base address

la $a1, Aend # End address

li $s0, 0 # Initialize counter = 0

li $s1, -1 # i-- (i in loopi)

count:

beq $a1, $a0, size # Compare A[0] with A[n]

addi $a1, $a1, -4 # A[n-1]

addi $s0, $s0, 1 # Increment element count

j count

size: addi $t0, $s0, -1 # t0 = n - 1

loopi: addi $s1, $s1, 1 # i++

li $s2, 0 # j = 0 (j in loopj)

beq $s1, $t0, exit # If i = n - 1 then exit

loopj: sub $t2, $t0, $s1 # t2 = (n - 1) - i

beq $s2, $t2, loopi # If j = (size - 1) - i then jump to loopi

if\_swap: sll $t3, $s2, 2 # Calculate distance of address A[j]

add $s3, $a0, $t3 # Calculate address A[j]

lw $v0, 0($s3) # Load value A[j]

addi $s3, $s3, 4 # Calculate address of A[j+1]

lw $v1, 0($s3) # Load value A[j+1]

sle $t4, $v0, $v1 # If A[j] <= A[j+1] then t4 = 1;

# If A[j] > A[j+1] then t4 = 0

beq $t4, $zero, swap # If t4 = 0 then jump to swap

addi $s2, $s2, 1 # j++

j loopj

swap: sw $v0, 0($s3) # Load A[j] into A[j+1]

addi $s3, $s3, -4 # Calculate address of A[j] = address of A[j+1] - 4

sw $v1, 0($s3) # Load A[j+1] into A[j]

addi $s2, $s2, 1 # j++

j loopj

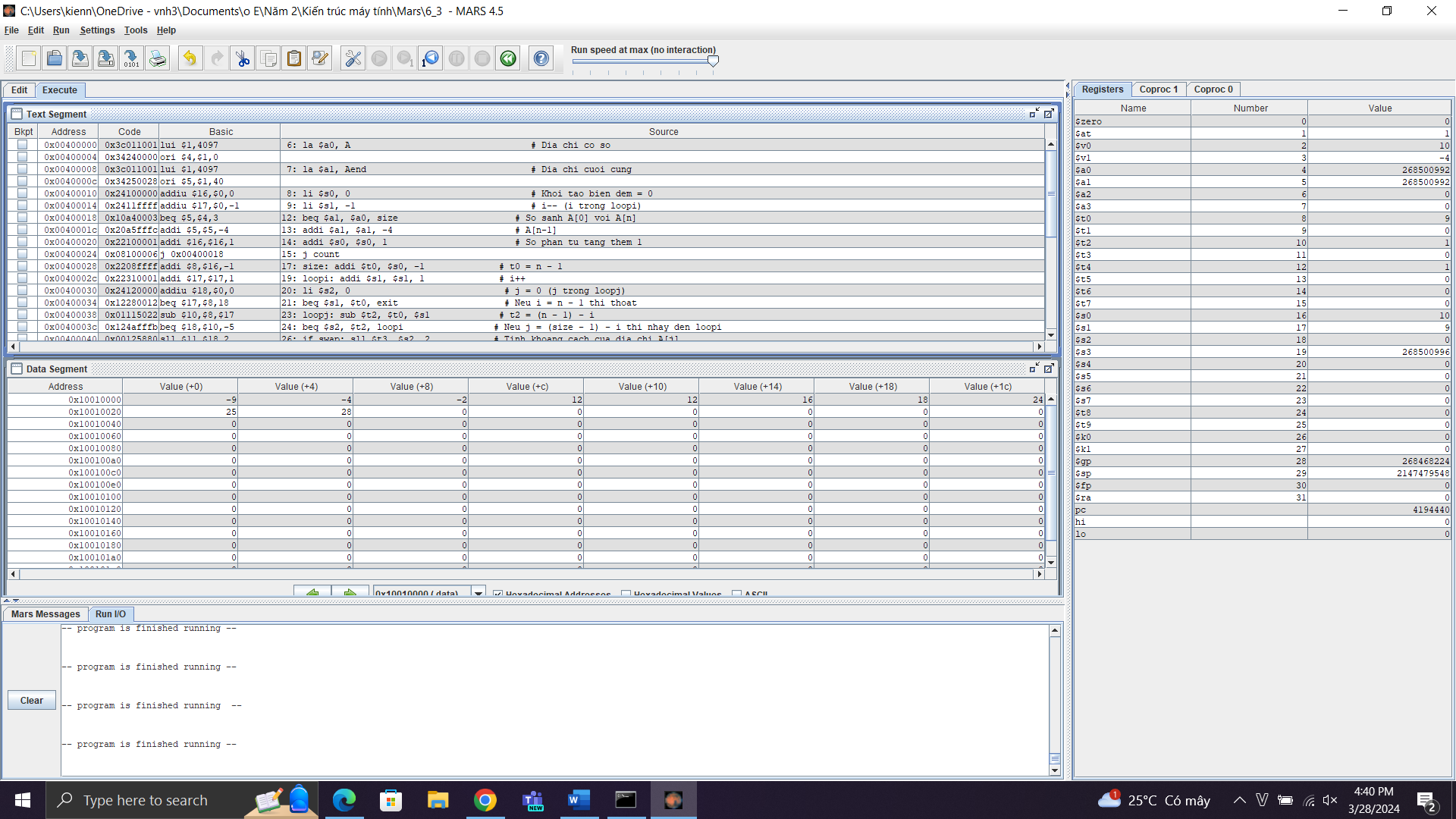
exit:

li $v0, 10

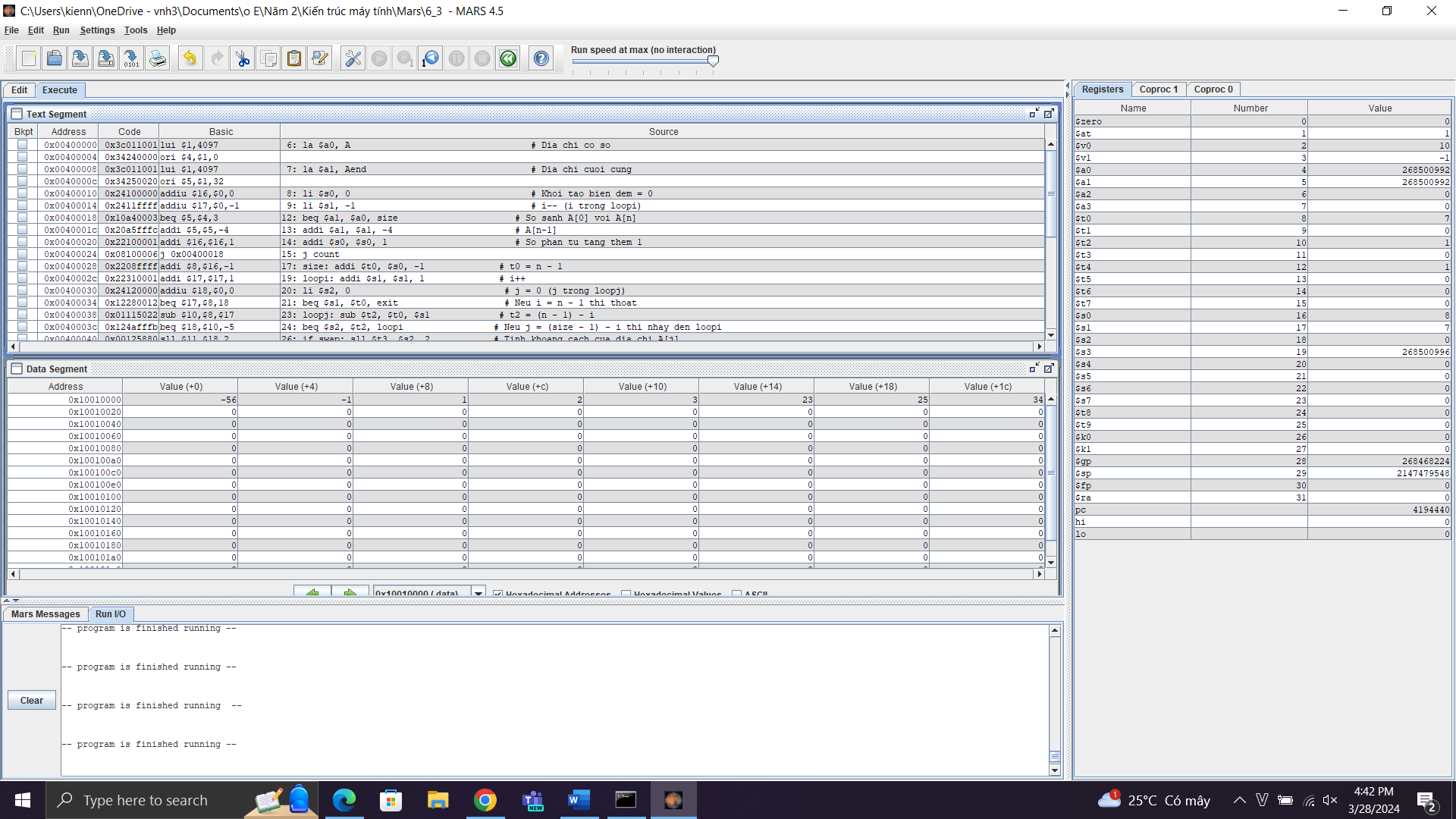
syscall

***Kết quả:***

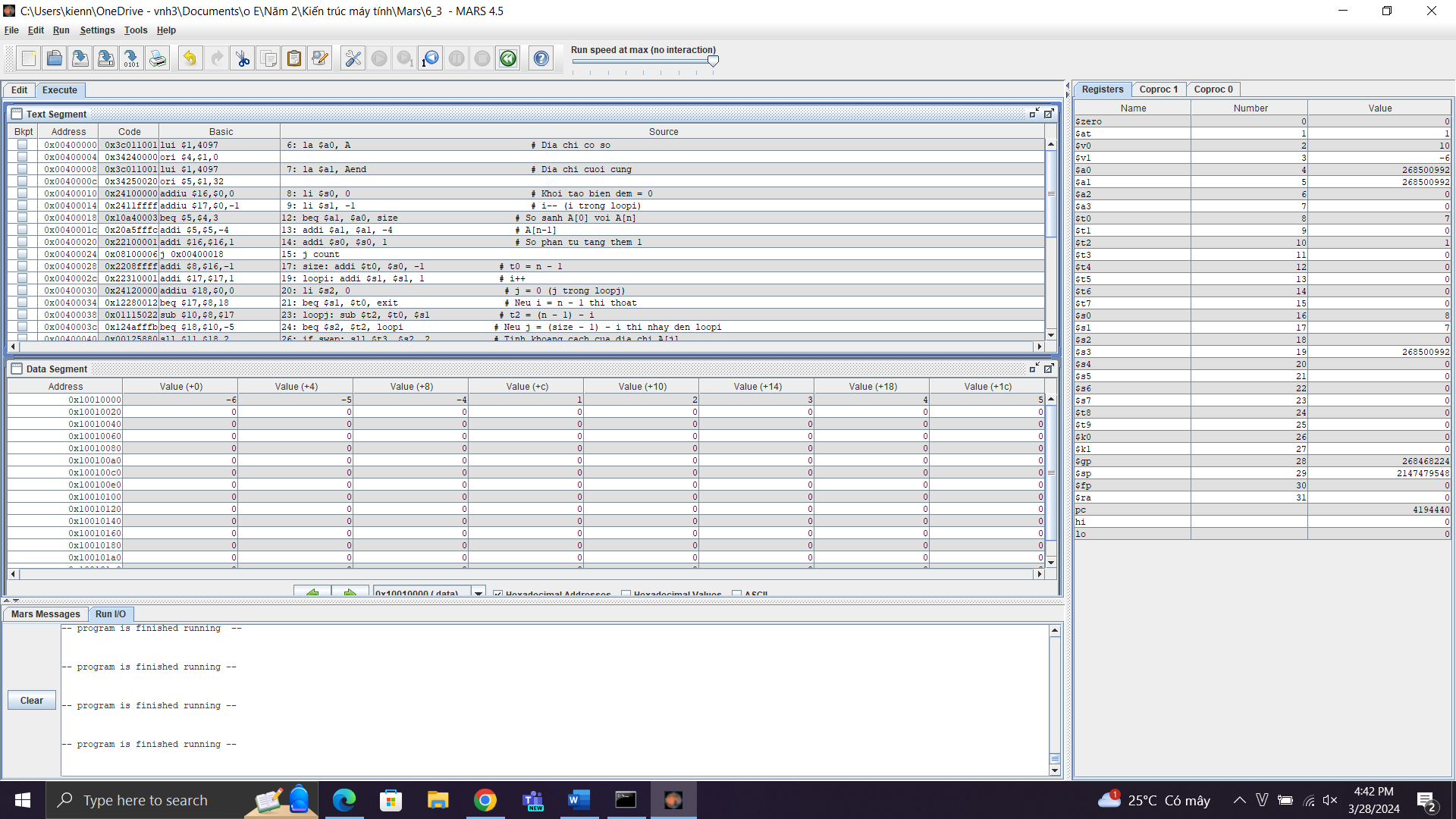
* **TH1**: Với mảng A: .word 28,-2,16,12,25,-9,24,12,18,-4



* **TH2**: Với mảng A: .word 25, -1, 23, -56, 2, 1, 34, 3



* **TH3**: Với mảng A: .word 5, 4, 3, 2, 1, -4, -5, -6



* Kết quả thu được 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.

***Code:***

.data

A: .word 28,-2,16,-12,18,4,25,-9,24,12

Aend: .word

.text

la $a0, A

la $a1, Aend

li $s0, 0

li $s1, 0

li $s2, 0

li $s3, 1

count:

beq $a1, $a0, loop

addi $a1, $a1, -4

addi $s0, $s0, 1

j count

loop:

beq $s3, $s0, exit

sll $t0, $s3, 2

add $s4, $a0, $t0

lw $s1, 0($s4)

addi $s2, $s3, -1

while:

slt $t1, $s2, $0

sll $t0, $s2, 2

add $s5, $a0, $t0

lw $t3, 0($s5)

sle $t4, $t3, $s1

add $t1, $t1, $t4

bne $t1, $0 , loop\_continue

addi $s5, $s5, 4

sw $t3, 0($s5)

addi $s2, $s2, -1

j while

loop\_continue:

addi $s5, $s5, 4

sw $s1, 0($s5)

addi $s3, $s3, 1

j loop

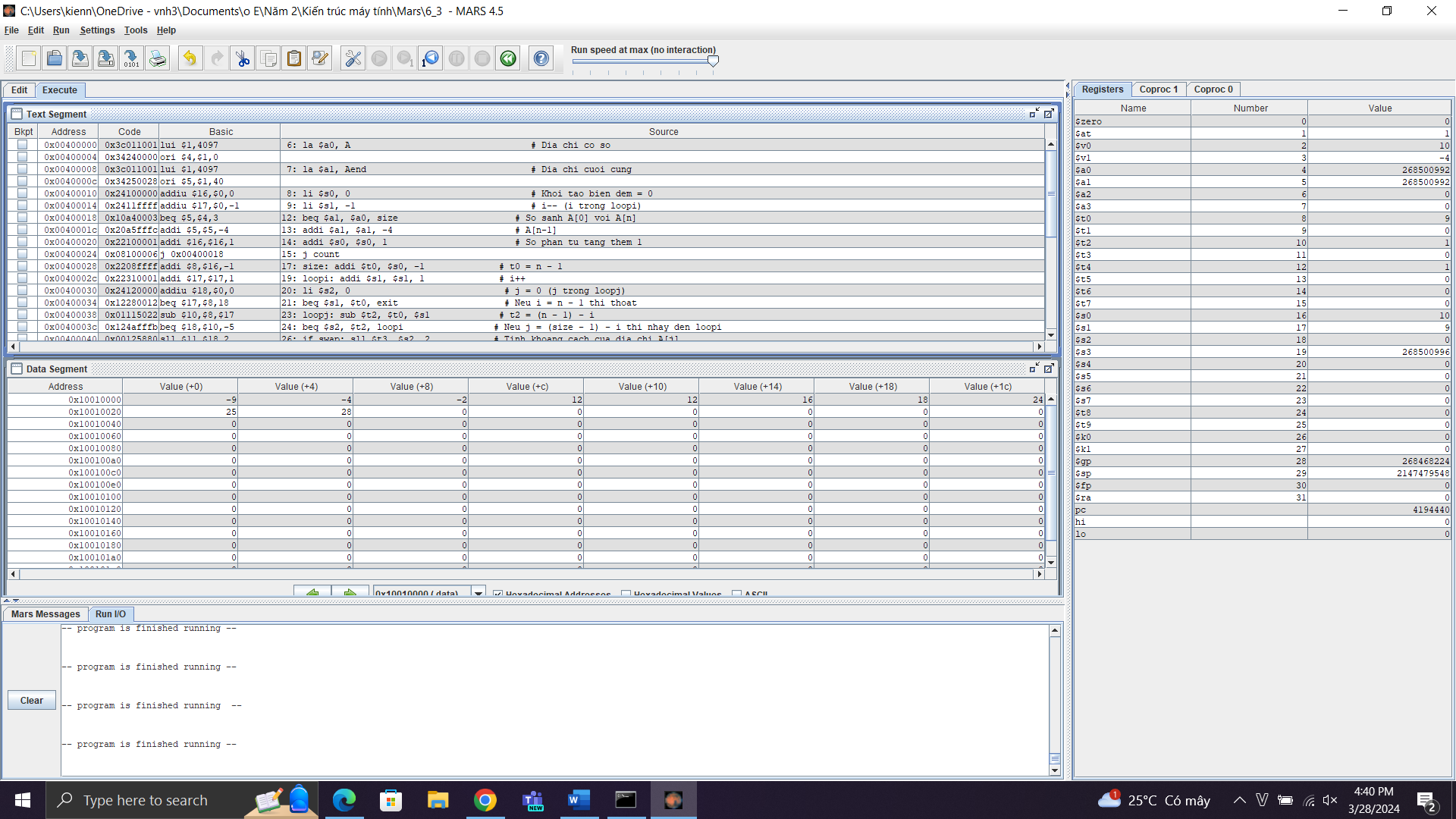
exit:

li $v0, 10

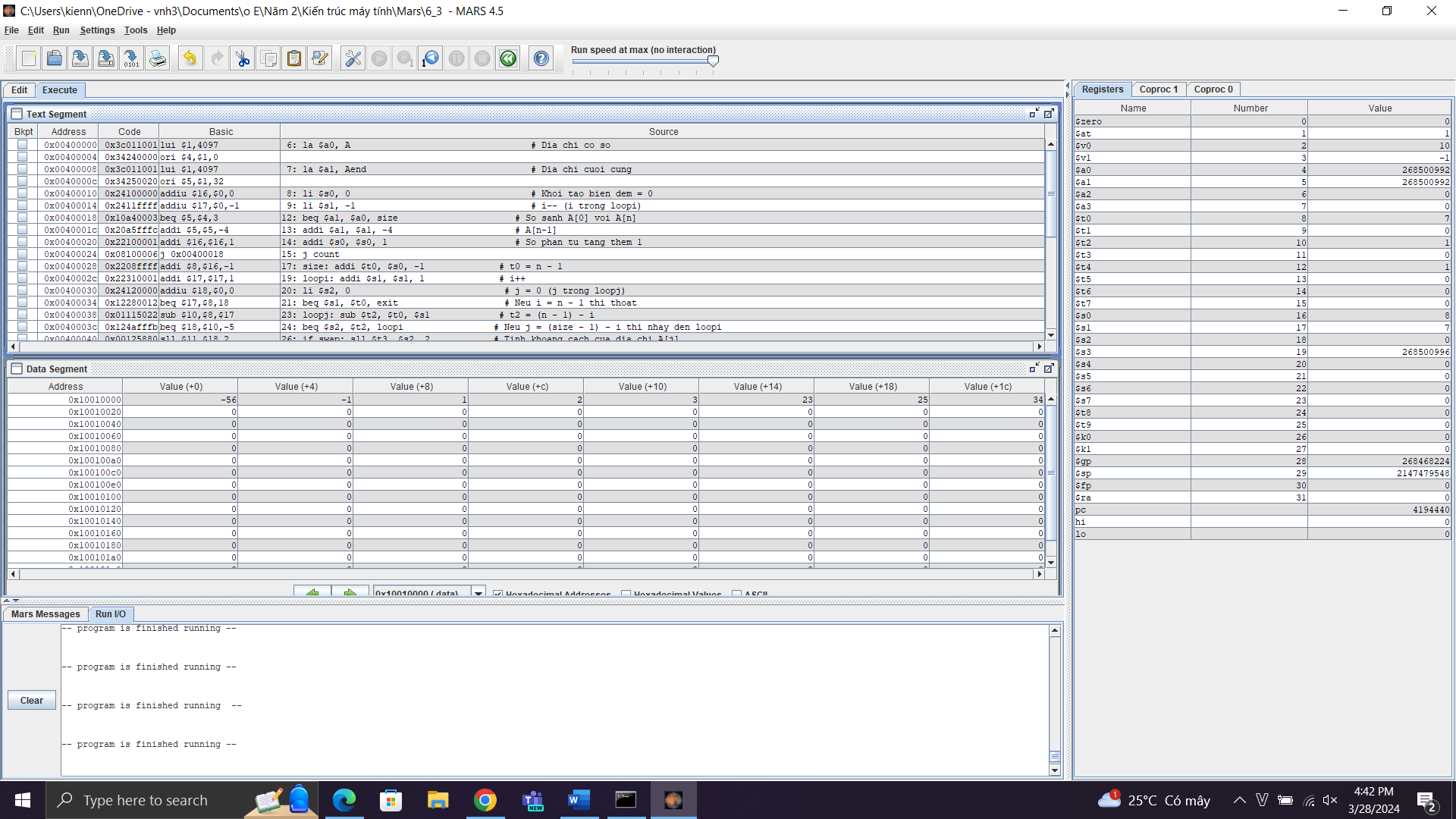
syscall

***Kết quả:***

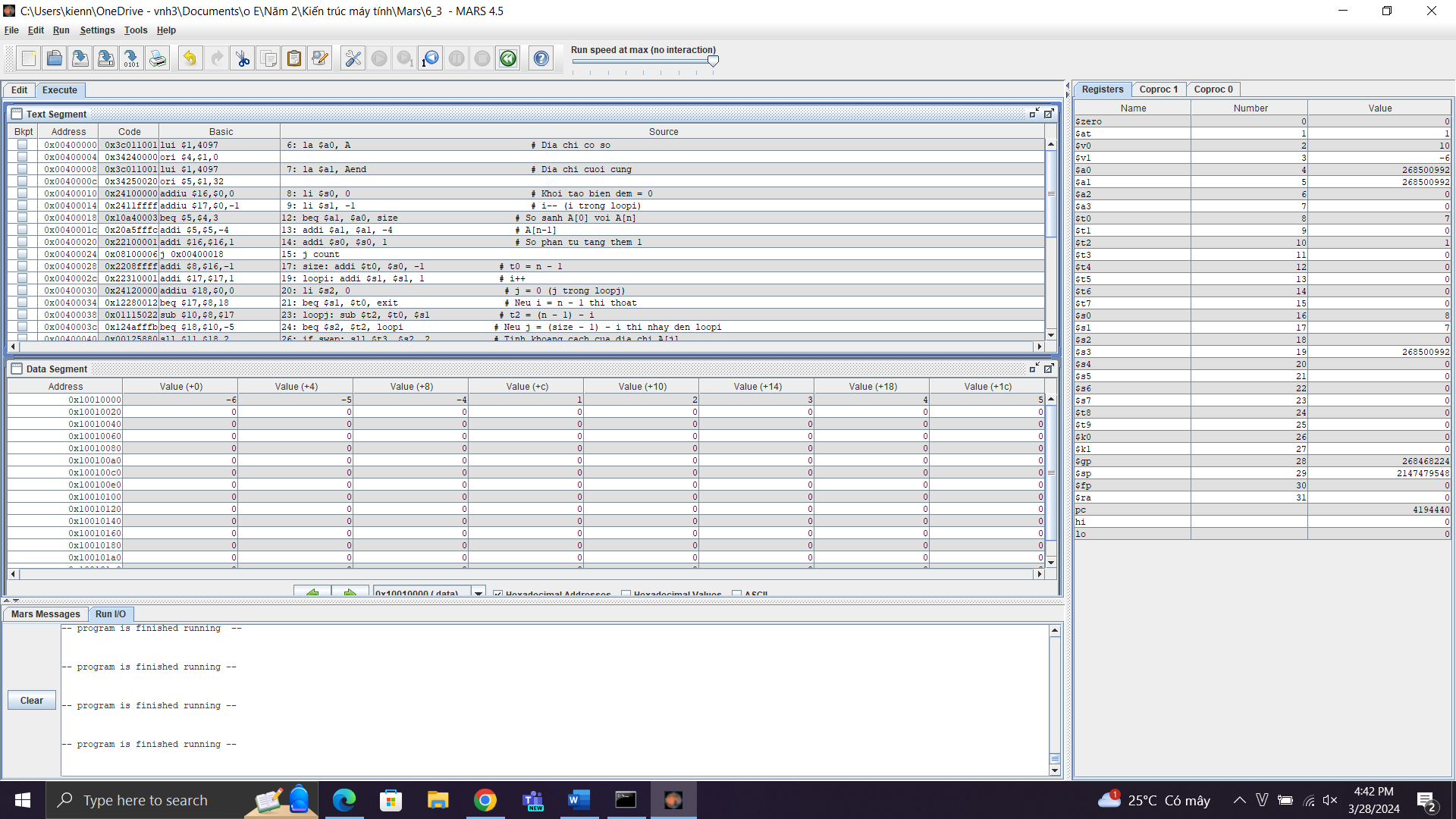
* **TH1**: Với mảng A: .word 28,-2,16,12,25,-9,24,12,18,-4



* **TH2**: Với mảng A: .word 25, -1, 23, -56, 2, 1, 34, 3



* **TH3**: Với mảng A: .word 5, 4, 3, 2, 1, -4, -5, -6



* Kết quả thu được đúng với lý thuyết.