BÁO CÁO THỰC HÀNH
IT3280 –THỰC HÀNH KIẾN TRÚC MÁY TÍNH
Tuần 03: LOAD/ STORE , JUMP & BRANCH INSTRUCTIONS.

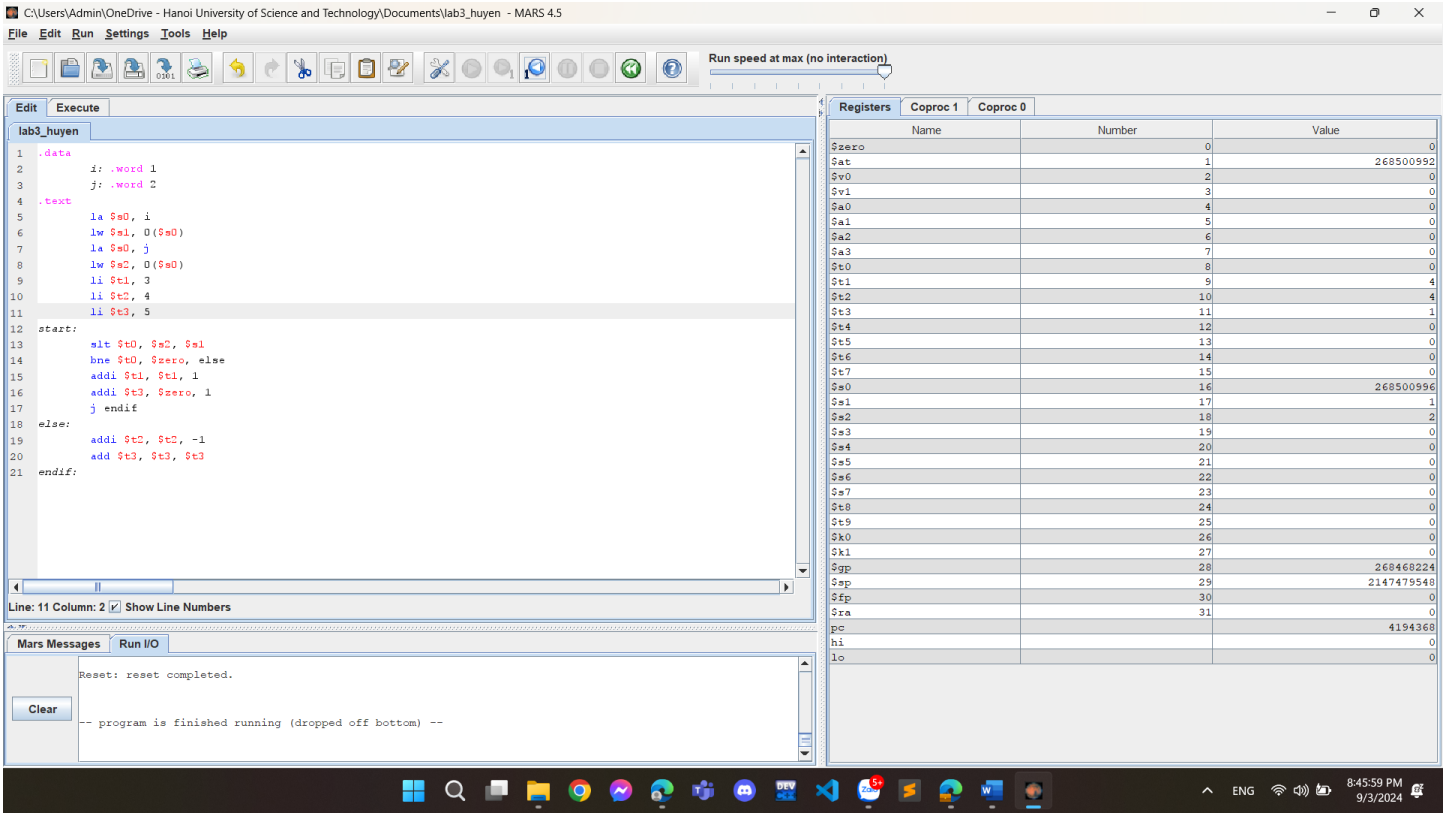
Họ và tên	Đinh Ngọc Khánh Huyền
Mã số sinh viên	20225726

Assignment 1

a. i <= j

* Khởi tạo với i=1, j=2 (i <= j) , x=3, y=4, z=5

- Mã nguồn:



- Nhận xét:

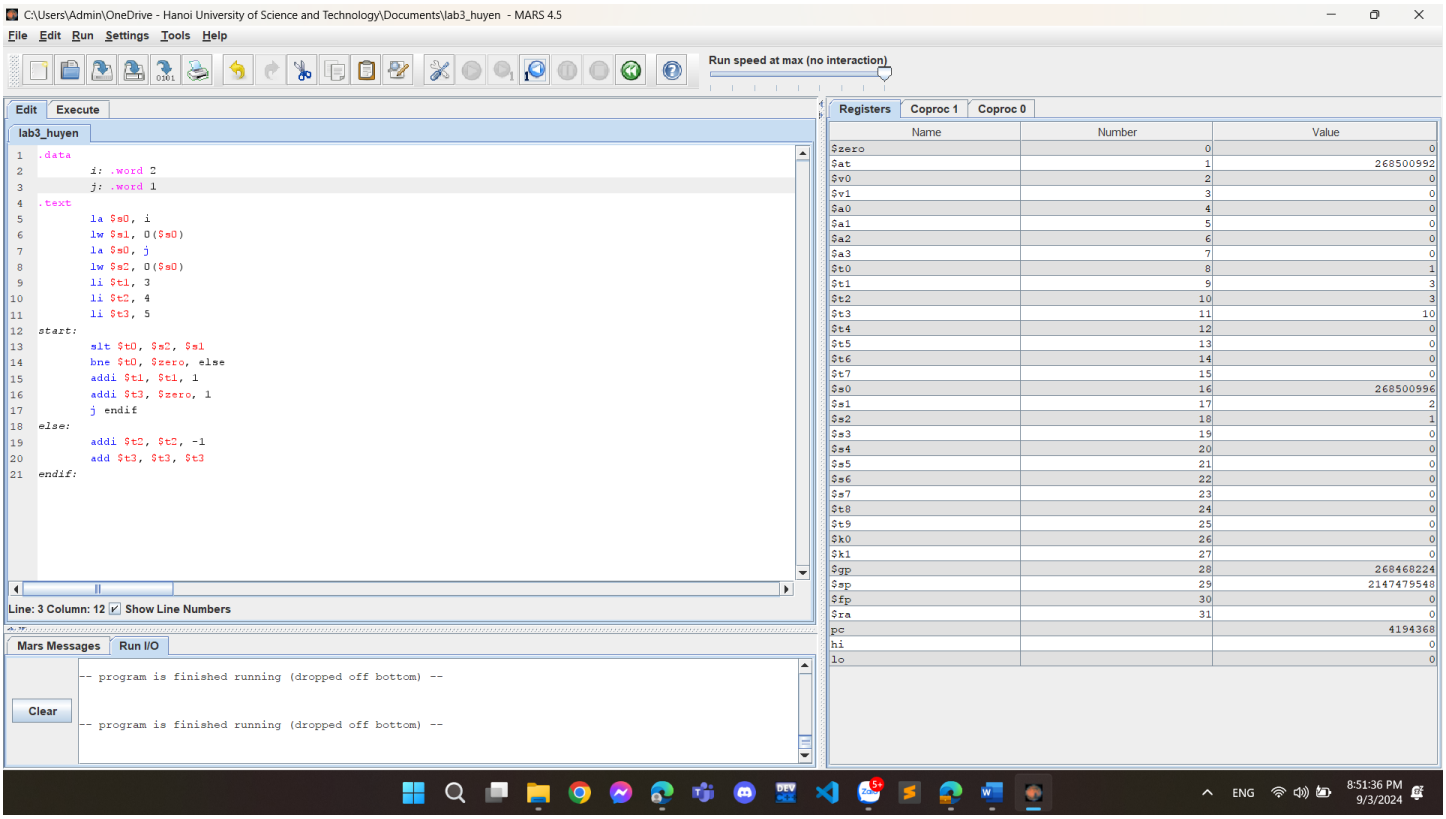
- Khởi tạo giá trị i=1 ứng với thanh ghi $s1và giá trị của=2 ứng với thanh ghi $s2
- Khởi tạo giá trị x, y, z ứng với thanh ghi $t1, $t2, $t3
- Ở câu lệnh “slt $t0,$s2,$s1”, điều kiện j<i không đúng nên thanh ghi $t0= 0 xuyên suốt chương trình.
- Khi đến câu lệnh bne (**B**ranch If **N**ot **E**qual), giá trị của thanh ghi $t0 bằng với giá trị của $zero nên không thực hiện rẽ nhánh đến else, vì vậy, chương trình sẽ tiếp tục thực hiện phần code ở start.
- x = x+1, z = 1 => $t1 = 4; $t3 = 1; $t2 giữ nguyên giá trị

=> Đúng với lý thuyết đưa ra

- Nhảy xuống endif và kết thúc

*Khởi tạo với i=2, j=1 (i>j), x=3, y=4, z=5

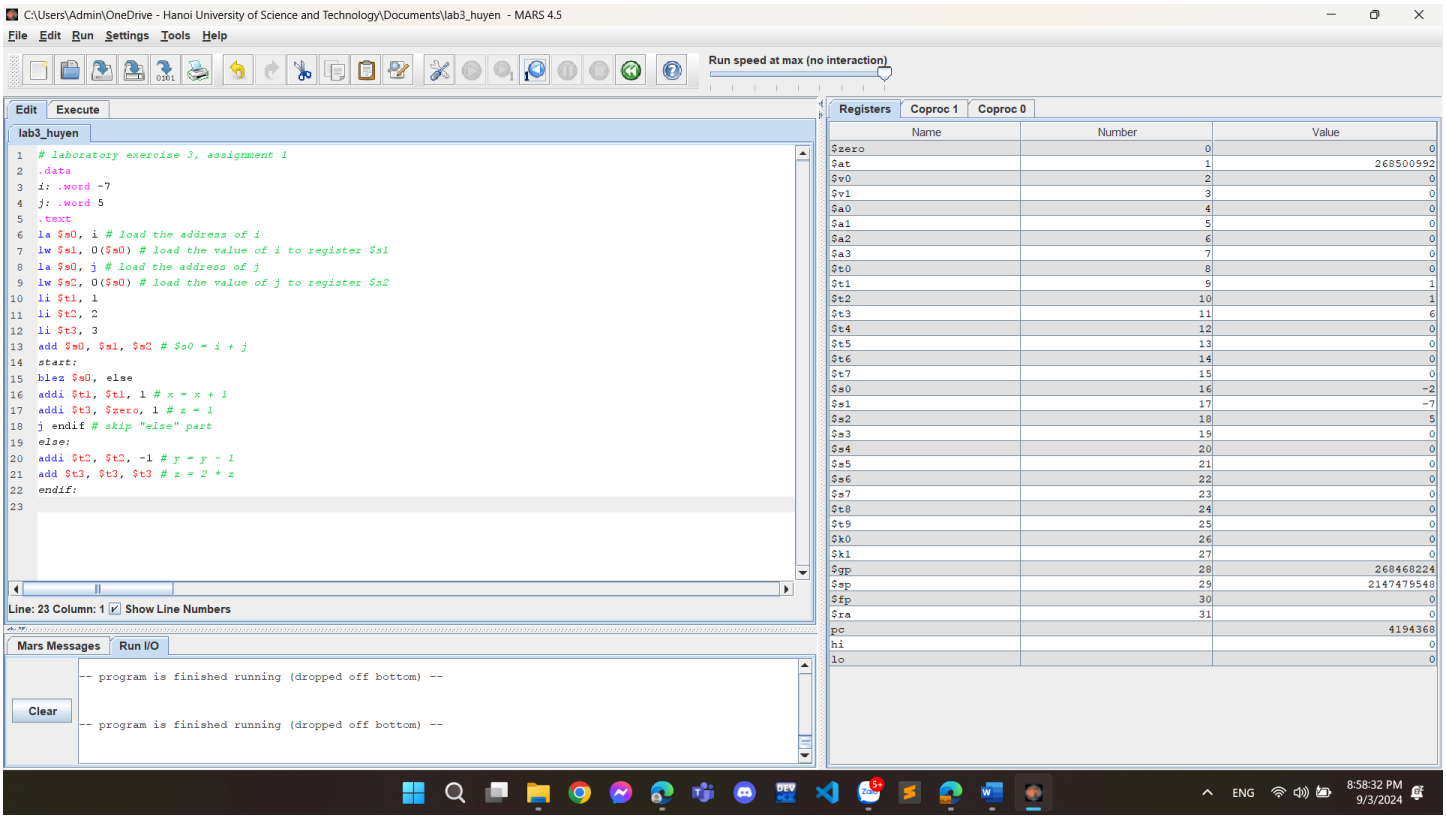
- Mã nguồn:



- Nhận xét:

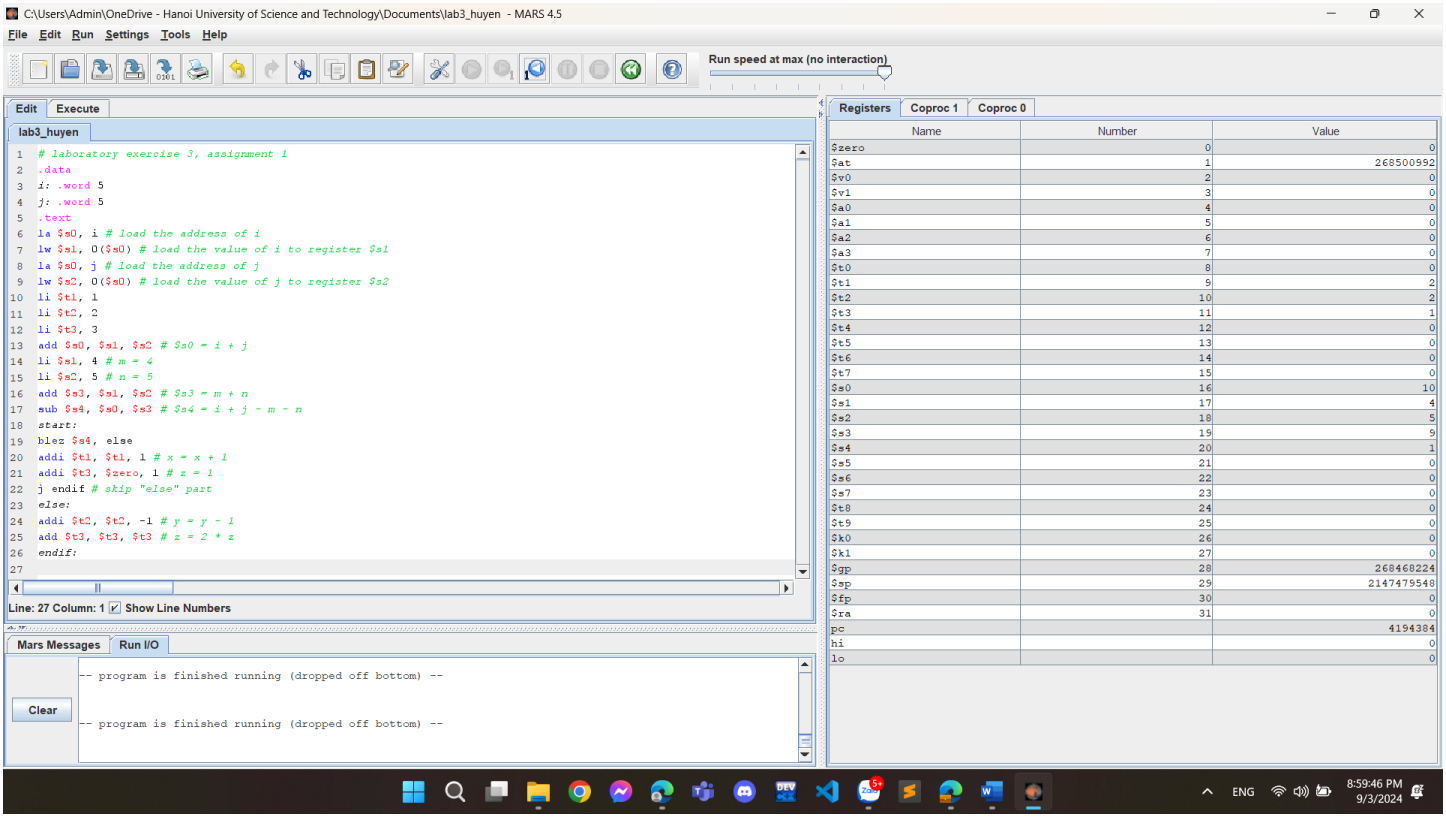
- Ở câu lệnh “slt $t0,$s2,$s1”, điều kiện j<I đúng nên thanh ghi $t0= 1
- Khi đến câu lệnh bne, giá trị của thanh ghi $t0 khác với giá trị của $zero nên thực hiện rẽ nhánh đến else
- y=y-1, z=2*z; $t1 giữ nguyên giá trị, $t2=3, $t3=10
  => Đúng với lý thuyết đưa ra

b, i+j>=0



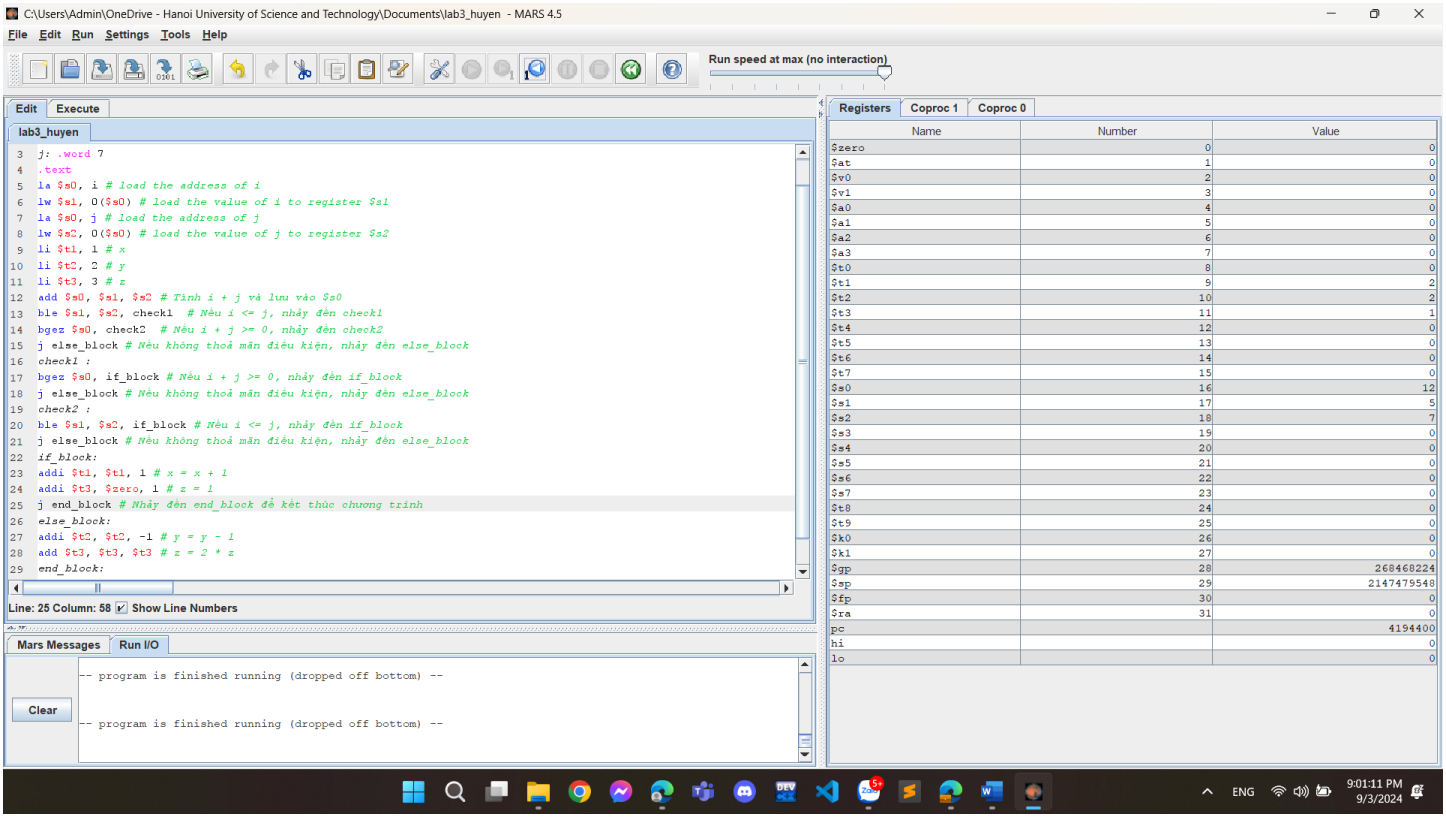
c, j+j>m+n

- Khởi tạo với i=5, j=5, m=4, n=5



d, *(i <= j) AND (i+j >= 0)*

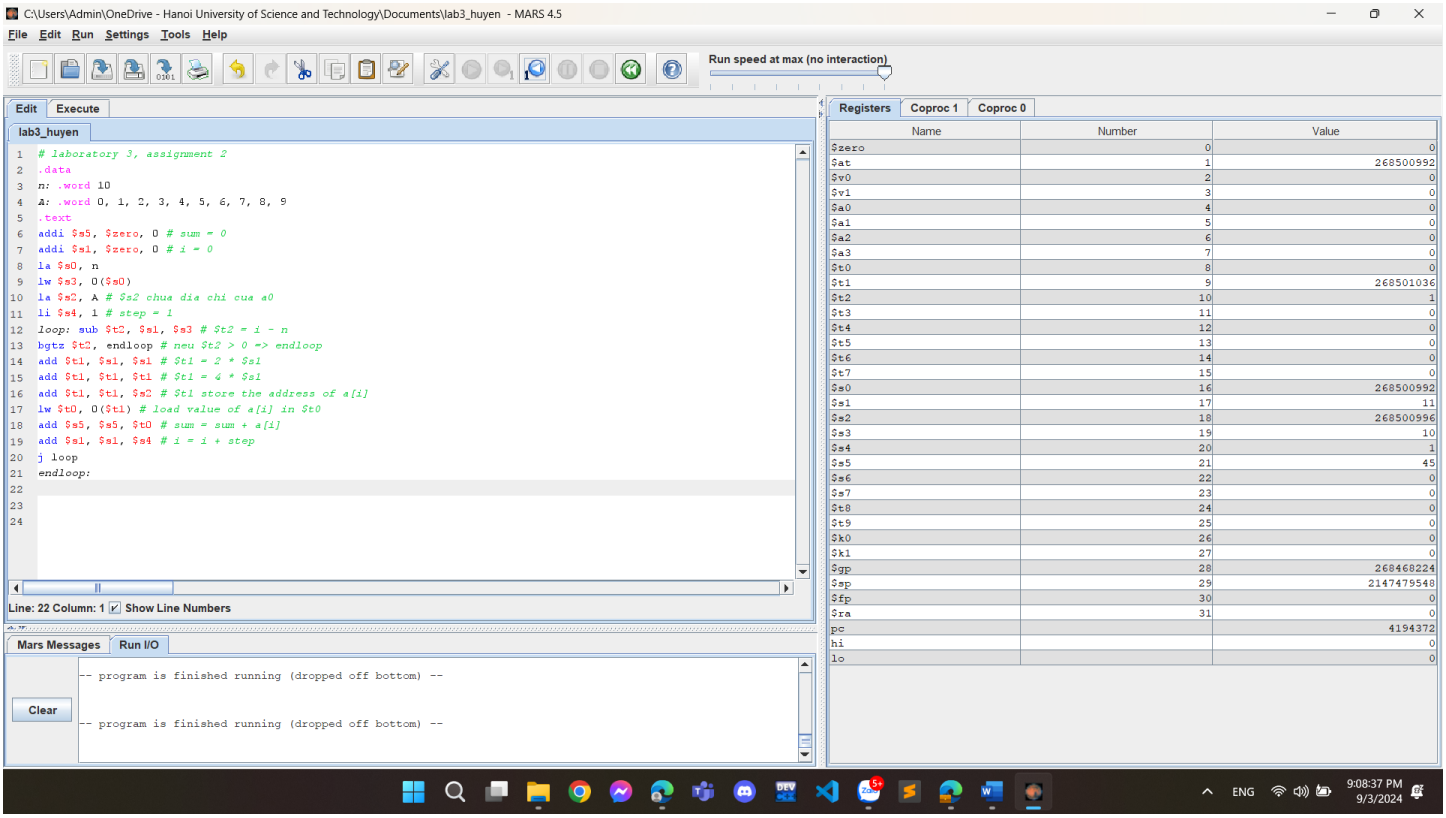
- Khởi tạo với i=5, j=7



Assignment 2

Khởi tạo mảng A: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

a, i <= n



- Nhận xét:

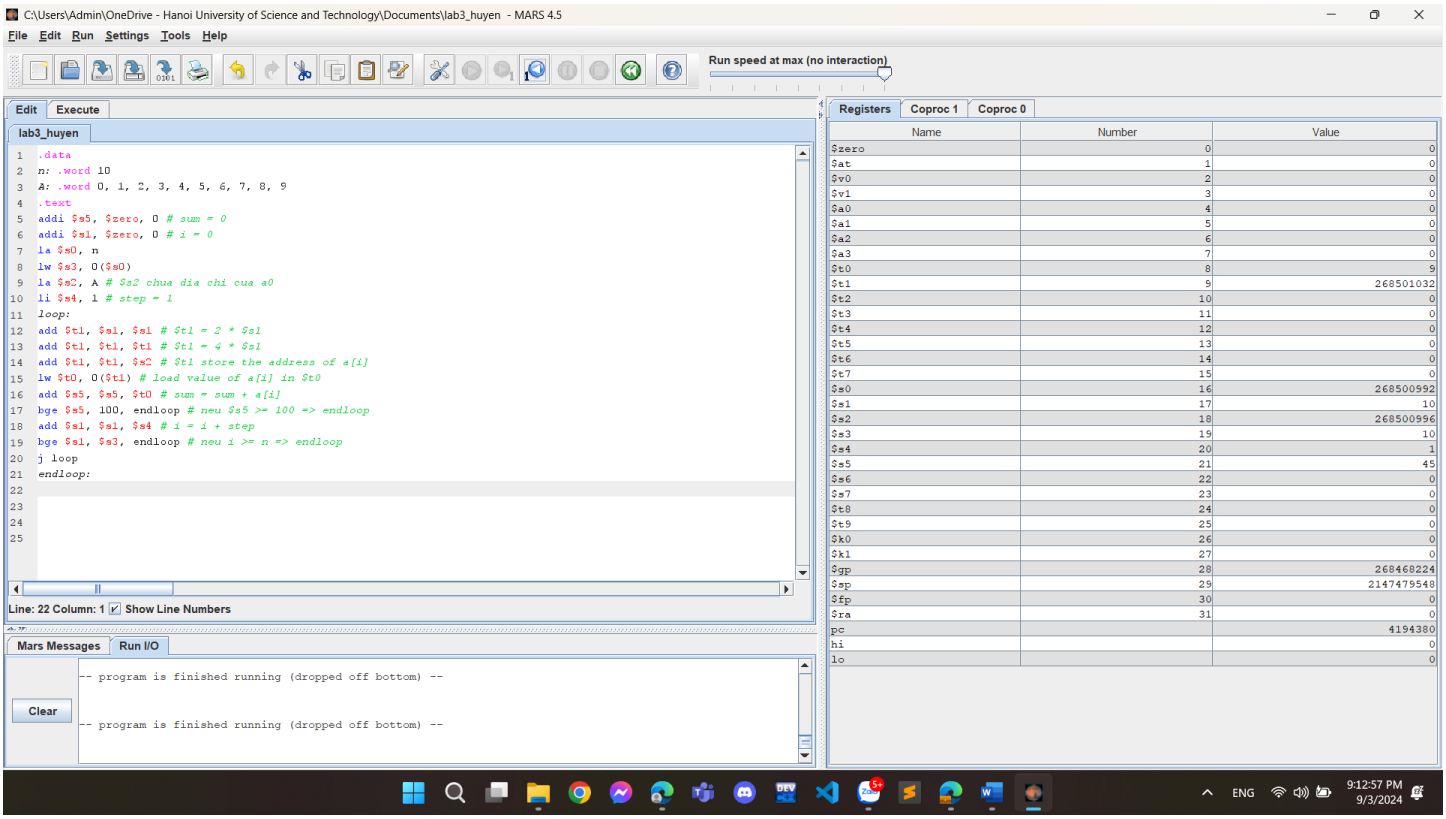
- Đoạn code trên thực hiện tính tổng các phần tử trong mảng arr

  - Câu lệnh ‘add $s1, $s1, $s4’ để tăng chỉ số của mảng theo step
  - Câu lệnh ‘add $t1, $s1, $s1’ và ‘add $t1, $t1, $t1’ được sử dụng để tính 4*i, vì các giá trị của mảng arr có độ dài 4 bytes.
  - Câu lệnh ‘add $t1, $t1, $s2’ câu lệnh này được sử dụng để lấy địa chỉ của phần tử thứ i ở trong mảng và lưu vào t1. Phải sử dụng câu lệnh này vì ta sẽ truy cập vào các phần tử trong mảng dựa trên địa chỉ của các phần tử.
  - Câu lệnh ‘lw $t0,0($t1)’ lưu giá trị của phần tử thứ i trong mảng vào thanh ghi $t0.
  - Câu lệnh ‘add $s5, $s5, $t0’ cộng dần các giá trị các phần tử trong mảng vào sum.
  - Câu lệnh ‘bne $s1, $s3, loop’ là điều kiện vòng lặp so sánh với n thì sẽ tiến hành rẽ nhánh về nhãn loop, nếu không, thì không thực hiện rẽ nhánh.

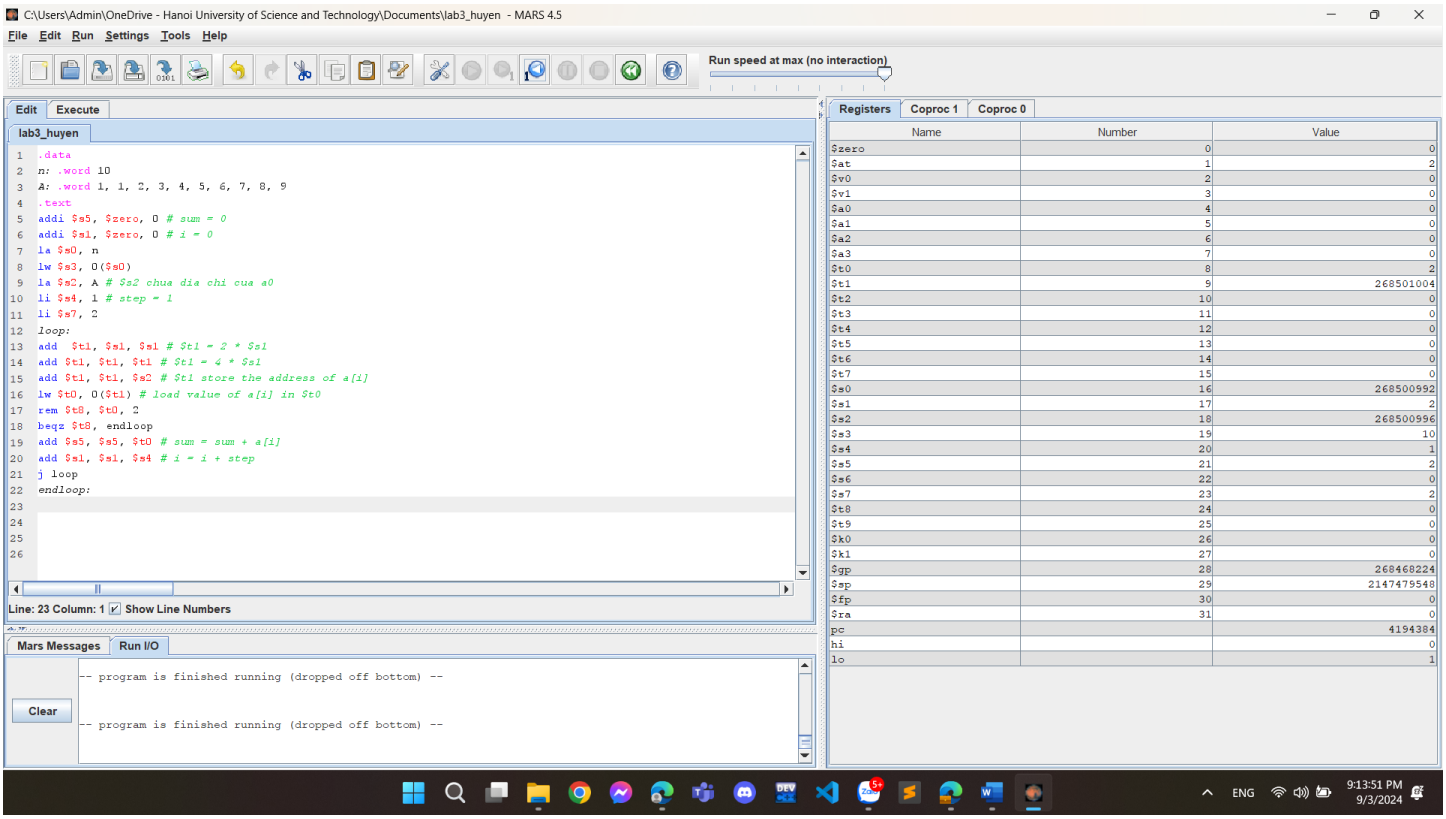
- Kết quả: Kết quả thanh ghi $s5: sum = 45

  => Đúng với lý thuyết đưa ra

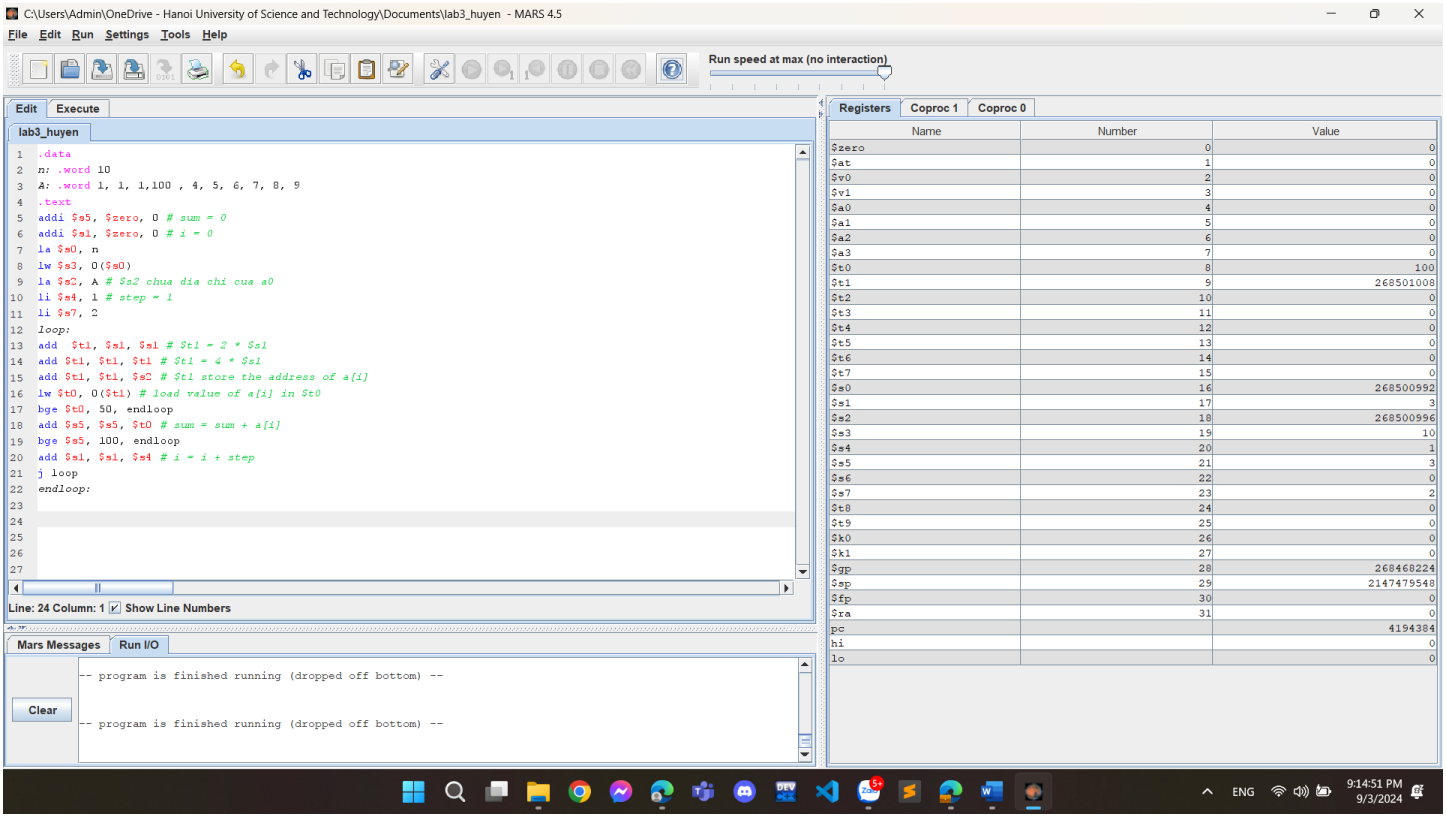
b, sum >= 100



c, *A[i] % 2 = 0*



d, *sum >= 100 OR A[i] >= 50*



Assignment 3

Sample Program 1:

- Các thanh ghi bị ảnh hưởng bởi các lệnh jump/branch:
  - $t0, $t1, $t2, $t3: các thanh ghi này được thay đổi dựa trên các điều kiện và nhánh được thực hiện trong chương trình.
- Giải thích về cách tính toán địa chỉ đích:
  - j: lệnh j thực hiện một nhảy vô điều kiện, địa chỉ đích là địa chỉ được chỉ định trong lệnh.
  - bne: lệnh bne nhảy nếu hai toán hạng không bằng nhau. Nếu điều kiện đúng, sẽ xảy ra nhánh; ngược lại, lệnh tiếp theo được thực hiện. Địa chỉ đích là địa chỉ hiện tại cộng với độ lệch được chỉ định trong lệnh.
  - else: trong trường hợp này, else là một nhãn, và địa chỉ đích của lệnh bne là địa chỉ của nhãn else.

Sample Program 2:

- Các thanh ghi bị ảnh hưởng bởi các lệnh jump/branch:

$s1, $s4, $s5, $t0, $t1: các thanh ghi này được thay đổi trong quá trình thực thi của chương trình.

- Giải thích về cách tính toán địa chỉ đích:
  - bne: tương tự như Chương trình Mẫu 1, lệnh bne nhảy nếu hai toán hạng không bằng nhau. Địa chỉ đích được tính dựa trên địa chỉ hiện tại cộng với độ lệch được chỉ định trong lệnh.
  - loop: địa chỉ đích của lệnh bne là địa chỉ của nhãn loop.

Sample Program 3:

- Các thanh ghi bị ảnh hưởng bởi các lệnhjump/branch:
  - $s0, $s1, $s2, $s3, $t0, $t1, $t2: các thanh ghi này được thay đổi trong quá trình thực thi của chương trình.
- Giải thích về cách tính toán địa chỉ đích:
  - beq: lệnh beq nhảy nếu hai toán hạng bằng nhau. Địa chỉ đích được tính dựa trên địa chỉ hiện tại cộng với độ lệch được chỉ định trong lệnh.
  - case_0, case_1, case_2, default: đây là các nhãn được sử dụng để nhánh. Các địa chỉ đích của các lệnh beq là địa chỉ của những nhãn này.

Assignment 4

- Mã nguồn

```
.data
arr: .word

#Generate a random number in the range of [0, 100]

.text
la $s0, arr #load address of arr into $s0
addi $s1,$zero,0 #count i
addi $s2,$zero, 40 #arr with 40 elements

loop_1: #initialize elements for arr
        slt $t0, $s1, $s2 # i < n?
        beq $t0 $zero, endloop_1 #brand to endloop if i >= n
        add $s3,$s1,$s1 #$s3=2*$s1
        add $s3,$s3,$s3 #$s3=4*$s1
        add $s4,$s0,$s3 #$s4 stores the address of A[i]
        li $v0, 42 # System call code to generate random integer
        li $a1, 200 # The upper bound is set in $a1
        syscall # The random number is stored in $a0
```

```
        sw $a0, 0($s4) #stores value of A[i] in $a0 register into address memory, that was saved
in $s4
        addi $s1,$s1,1 #i = i + 1(step)
        j loop_1

endloop_1:
        addi $s1,$zero,0 #load value 0 into $s1 (reset count i = 0)

#largest element in $t1 and address in $t4
#second largest element in $t2 ddress in $t5
#third          largest element in $t3 ddress in $t6

loop_2: #find 3 largest elements in arr
        slt $t0, $s1, $s2 # i < n?
        beq $t0 $zero, endloop_2 #brand to endloop if i >= n
        add $s3,$s1,$s1 #$s3=2*$s1
        add $s3,$s3,$s3 #$s3=4*$s1
        add $s4,$s0,$s3 #$s4 stores the address of A[i]
        lw $s5, 0($s4) #load value of A[i] in $s5

        slt $s6, $t1, $s5  #largest number now < A[i]?
        beq $s6,$zero, else_1
        add $t1,$zero,$s5 #update largest number
        add $t4, $zero, $s4 #update address of largest number
        j endif

        else_1:
        slt $s6, $t2, $s5  #second largest number now < A[i]?
        beq $s6,$zero, else_2
        add $t2,$zero,$s5 #update second largest number
        add $t5, $zero, $s4 # #update address of second largest number
        j endif

        else_2:
        slt $s6, $t3, $s5 #third largest number now < A[i]?
        beq $s6,$zero, endif
        add $t3,$zero,$s5 #update third largest number
        add $t6, $zero, $s4 #update address of third largest number

        endif:
        addi $s1, $s1, 1
        j loop_2

endloop_2:
```

```
li $v0, 1 # System call code to print an integer on the console
add $a0,$t1,$zero #$a0 = largest number
syscall # The integer needs to be printed out is stored in $a0
add $a0, $zero, 10 #ma ascii de xuong dong
li $v0,11
syscall

li $v0, 1
add $a0,$t4, $zero #$a0 = address of largest number
syscall # The integer needs to be printed out is stored in $a0
add $a0, $zero, 10  #ma ascii de xuong dong
li $v0,11
syscall

li $v0, 1
add $a0,$t2, $zero #$a0 = second largest number
syscall # The integer needs to be printed out is stored in $a0
add $a0, $zero, 10 #ma ascii de xuong dong
li $v0,11
syscall

li $v0, 1
add $a0,$t5, $zero #$a0 = address of second largest
syscall # The integer needs to be printed out is stored in $a0
add $a0, $zero, 10 #ma ascii de xuong dong
li $v0,11
syscall

li $v0, 1
add $a0,$t3, $zero #$a0 = third largest number
syscall # The integer needs to be printed out is stored in $a0
add $a0, $zero, 10 #ma ascii de xuong dong
li $v0,11
syscall

li $v0, 1
add $a0,$t6, $zero #$a0 = address of third largest
syscall # The integer needs to be printed out is stored in $a0
```

- Giải thích:
  - la $s0, arr: Load địa chỉ của mảng vào thanh ghi $s0.
  - addi $s1, %zero, 0: Khởi tạo biến i (vị trí trong mảng) bằng 0.
  - addi $s2, zero, 40: Gán giá trị 40 cho n (số phần tử của mảng).
  - loop_1: Khởi tạo giá trị cho các phần tử của mảng.

    - Lệnh slt $t0, $s1, $s2 kiểm tra xem i<n hay không.
    - Nếu không (beq %t0, $zero, endloop_1) nhảy đến nhãn endloop_1.
    - Cập nhật địa chỉ của phần tử tiếp theo trong mảng và lấy một số ngẫu nhiên từ 0 đến 199 để lưu vào địa chỉ đó.
    - Tăng I lên 1 và quay lại loop_1
- loop_2: Tìm ba số lớn nhất và địa chỉ của chúng trong mảng
    - Lệnh slt $t0, $s1, $s2 kiểm tra xem i<n hay không.
    - Nếu không (beq %t0, $zero, endloop_2) nhảy đến nhãn endloop_2.
    - Đọc giá trị của phần tử tại vị trí i trong mảng.
    - So sánh giá trị này với ba số lớn nhất hiện tại và cập nhật chúng nếu cần.
- endif: Tiếp tục tăng i lên 1 và quay lại loop_2.
- endloop_2: In ra ba số lớn nhất và địa chỉ của chúng.

- Kết quả: