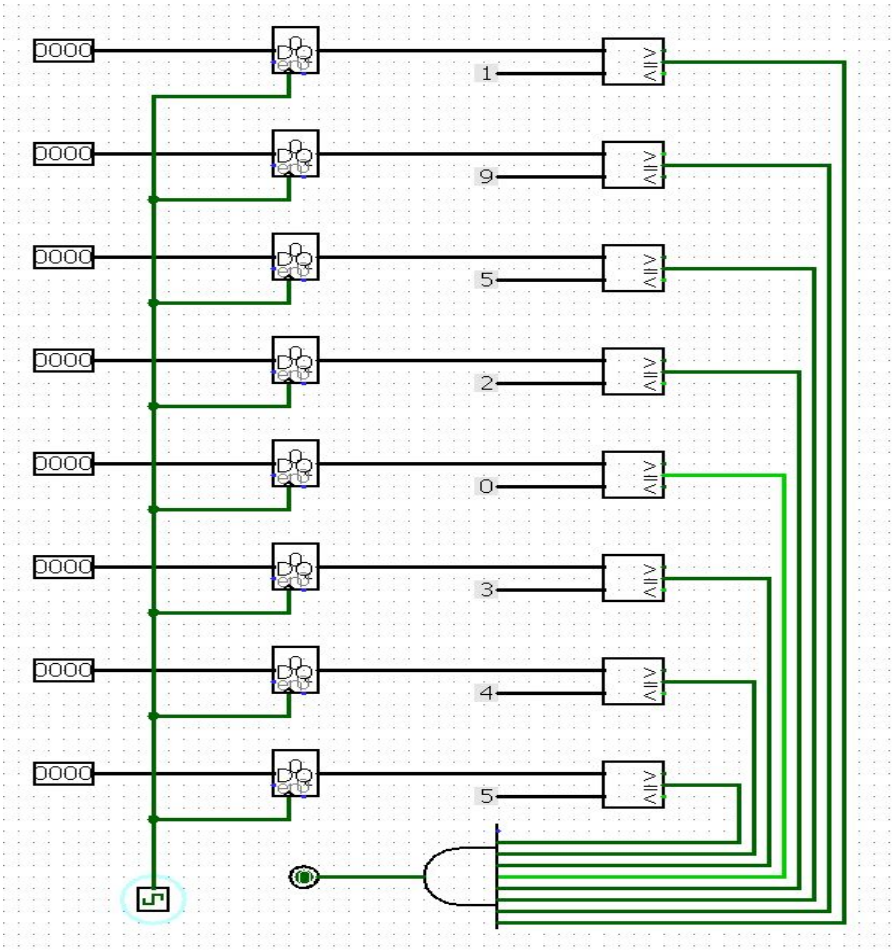
Họ tên : Đặng Vũ Phương Uyên
MSSV : 19520345
Lớp : IT012.L11

BÀI KIỂM TRA
THỰC HÀNH TỔ CHỨC VÀ CẤU TRÚC MÁY TÍNH II

Câu 1: Sinh viên sử dụng phần mềm Logisim để thiết kế mạch so sánh mã số sinh viên(8 số). Nếu sinh viên nhập vào 8 số theo thứ tự mã số sinh viên thì output bật lên 1.(2đ)



Vẽ mạch so sánh Mã số sinh viên có 8 chữ số:

Giả sử, ta có MSSV là 19520345, ta sẽ gán từng chữ số trong chuỗi số vào từng Constant trong Logisim như hình trên. Và có 8 giá trị input (đầu vào) - mỗi giá trị có 4 bits ứng với từng chữ số.

Sau đó, lưu từng giá trị vào thanh ghi cổng là 4 bits rồi nối với Comparator tiến hành so sánh với các giá trị Constant đã chọn trước.

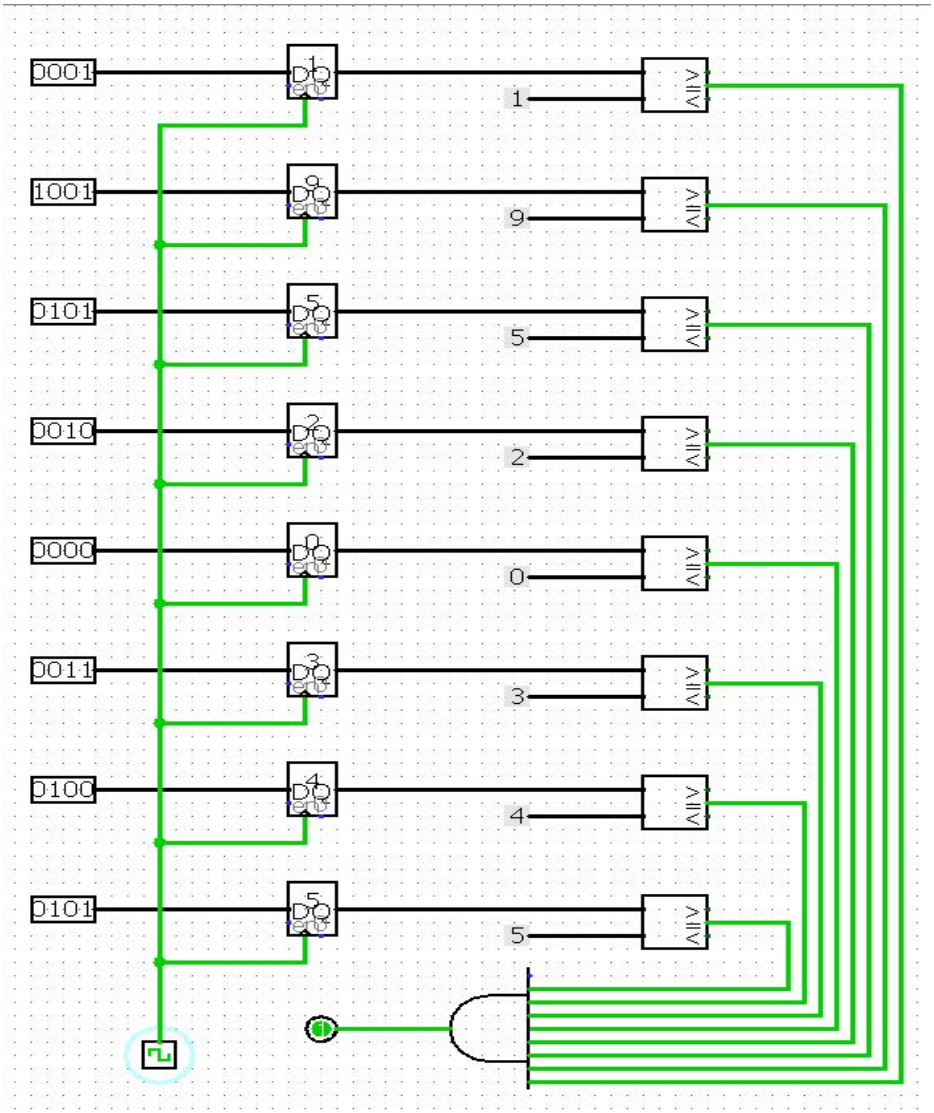
Khi thay đổi giá trị của các Inputs, t bật Clock để lưu lại giá trị rồi tiến hành so sánh. Nếu tất cả các giá trị đầu vào đều bằng Constant tương ứng với nó thì trả về giá trị là 1, ngược lại thì xuất 0.

Kết quả kiểm tra bằng Logisim:

TH1: Các input có thứ tự lần lượt là 1, 9, 5, 2, 0, 3, 4, 5.

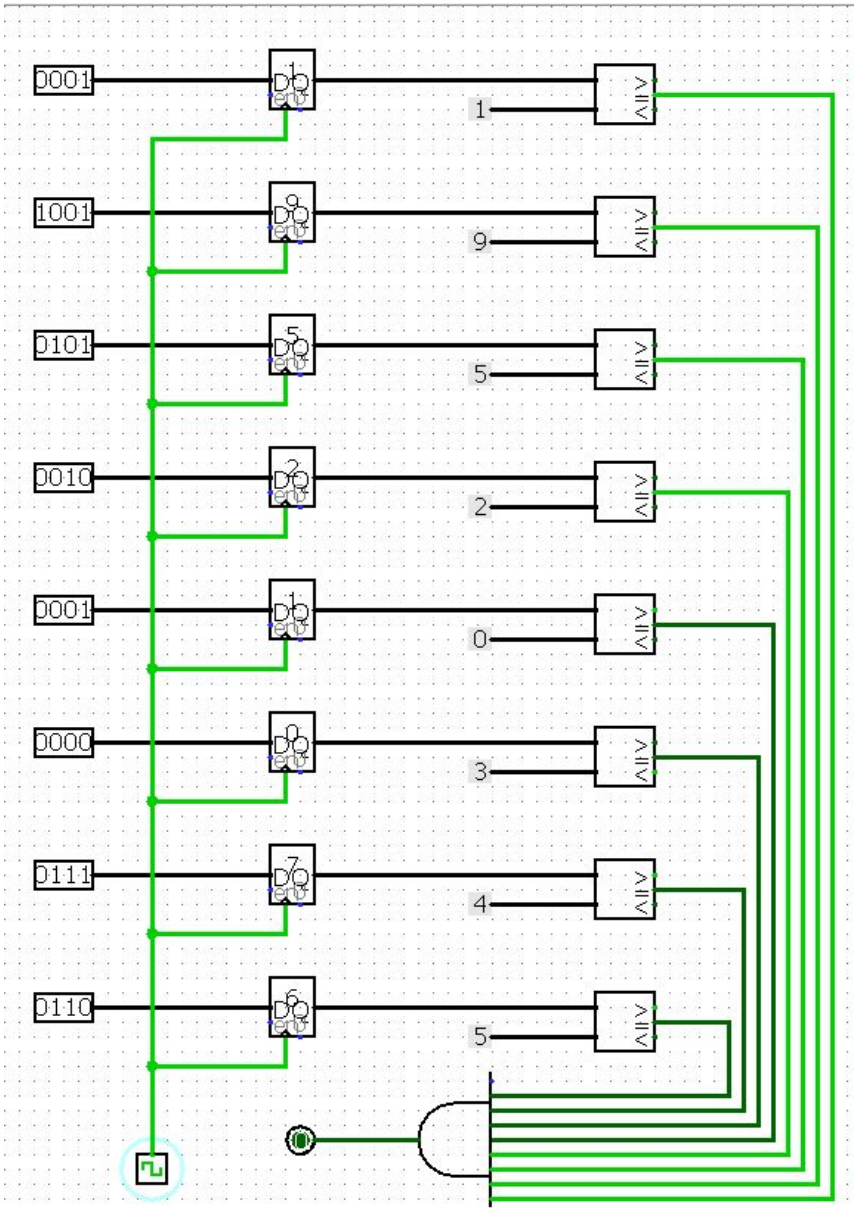
Các Constant theo thứ tự lần lượt là 1, 9, 5, 2, 0, 3, 4, 5.

→ Đầu ra mong muốn: 1.



⇒ Đầu ra thực tế giống như đầu ra mong muốn.

TH2: Các input có thứ tự lần lượt là 1, 9, 5, 2, 1, 0, 7, 6.
Các Constant theo thứ tự lần lượt là 1, 9, 5, 2, 0, 3, 4, 5.
→ Đầu ra mong muốn: 0.



⇒ Đầu ra thực tế giống như đầu ra mong muốn.

Câu 2: Sử dụng phần mềm Mars nhập một mảng gồm 8 nguyên dương:
a. Kiểm tra 8 số đó có hợp thành MSSV của bạn theo thứ tự. (3đ)

STT	*Chương trình*	*Ý nghĩa*

1	.data
	
	str1: .asciiz "\nNhap MSSV: "
	arr1: .space 8
	
	str5: .asciiz " => Tao thanh MSSV"
	str6: .asciiz " => Khong tao thanh MSSV"
	
	.text
	main:
	li $s0, 0 # s0 = mssv = 0
	jal nhap_mssv
	
	la $s1, arr1 # s1 = & arr1
	jal nhap_mang_va_xu_ly
	
	jal so_sanh
	
	j exit
	
	nhap_mssv:
	la $a0, str1
	li $v0, 4
	syscall
	li $v0, 5
	Syscall
	
	move $s0, $v0
	jr $ra
	
	nhap_mang_va_xu_ly:
	addi $t0, $0, 8 # n = 8
	li $t1, 0 # i = 0
	addi $t2, $0, 10000000 # t2 = 10000000
	
	addi $t3, $0, 10 # t3 = 10
	addi $t4, $0, 0 # sum = t4 = 0
	j nhap_phan_tu_mang_va_xu_ly
		● Khai báo các biến cho chương trình sau chỉ thị này:
		❖ str1 = "Nhap MSSV: ".
		❖ arr1: là một chuỗi có 8 bytes bộ nhớ, chưa được khởi tạo.
		❖ str5 = " => Tao thanh MSSV."
		❖ str6 = " =>Khong tao thanh MSSV".
		
		● Viết câu lệnh sau chỉ thị này.
		● Điểm bắt đầu chương trình chính:
		❖ Khởi tạo thanh ghi có giá trị là 0.
		❖ Lưu địa chỉ trở về vào $ra và nhảy tới hàm nhap_mssv.
		❖ Khởi tạo thanh ghi với địa chỉ của arr1.
		❖ Lưu địa chỉ trở về vào $ra và nhảy tới hàm nhap_mang_va_xu_ly.
		❖ Lưu địa chỉ trở về vào $ra và nhảy tới hàm so_sanh.
		❖ Nhảy tới hàm thoát chương trình.
		
		● Hàm nhap_mssv:
		❖ Khởi tạo thanh ghi với địa chỉ của str1.
		❖ Khởi tạo thanh ghi có giá trị là 4.
		❖ Dùng để xuất chuỗi str1.
		❖ Khởi tạo thanh ghi có giá trị là 5.
		❖ Dùng để nhập số nguyên dương có 8 chữ số.
		❖ Di chuyển giá trị $v0 vào $s0.
		❖ Nhảy đến địa chỉ trong thanh ghi $ra.
		
		● Hàm nhap_mang_va_xu_ly:
		❖ Đưa giá trị 8 vào thanh ghi $t0.
		❖ Đưa giá trị 0 vào thanh ghi $t1.
		❖ Đưa giá trị 10000000 vào thanh ghi $t2.
		❖ Đưa giá trị 10 vào thanh ghi $t3.
		❖ Đưa giá trị 0 vào thanh ghi $t4.
		❖ Nhảy tới hàm nhap_phan_tu_mang_va_xu_ly.
		● Hàm nhap_phan_tu_mang_va_xu_ly:

```
nhap_phan_tu_mang_va_xu_ly:
        beq $t1, $t0, exit_nhap       # if (i == n)

        li        $v0, 5

        syscall
        add $t5,$s1,$t1
        sb $v0, ($t5)

        mult $v0,$t2
        mflo $v0
        add $t4,$t4,$v0        # t4 = t4 + v0
        div $t2,$t3
        mflo $t2                       # gan ket
qua vao t2 => t2 = t2 // t3
        addi $t1, $t1, 1               # i = i + 1

        j nhap_phan_tu_mang_va_xu_ly   #
quay lai vong lap

exit_nhap:
        jr $ra

so_sanh:
        beq $s0, $t4, true



        li $v0,4

        la $a0,str6
        syscall
        jr $ra

true:
        li $v0,4
        la $a0,str5
        syscall
        jr $ra

exit:
        li $v0, 10
        syscall
```

- Nếu $t1 = $t0 thì nhảy tới hàm exit_nhap.
- Ngược lại, khởi tạo thanh ghi có giá trị là 5.
- Dùng để nhập giá trị phần tử thứ i.
- $t5 = $s1 + $t1.
- Lưu giá trị $v0 vào $t5 (lưu vào mảng).
- $v0 * $t2.
- Gán $v0 bằng giá trị tích trên.
- Gán $t4 bằng tổng của $t4 với $v0.
- Lấy $t2 chia 10.
- Gán $t2 là kết quả lấy phần nguyên của phép thương trên.
- Vị trí phần tử mảng tăng lên một đơn vị.
- Quay lại hàm nhap_phan_tu_mang_va_xu_ly.

- Hàm exit_nhap:
  - Nhảy đến địa chỉ trong thanh ghi $ra.

- Hàm so_sanh:
  - Nếu $t4 = $s0 (8 chữ số trong mảng hợp thành MSSV) thì nhảy tới hàm true.
  - Ngược lại, khởi tạo thanh ghi có giá trị là 4.
  - Khởi tạo thanh ghi với địa chỉ của str6.
  - Dùng để xuất chuỗi str6.
  - Nhảy đến địa chỉ trong thanh ghi $ra.

- Hàm true:
  - Khởi tạo thanh ghi có giá trị là 4.
  - Khởi tạo thanh ghi với địa chỉ của str5.
  - Dùng để xuất chuỗi str5.
  - Nhảy đến địa chỉ trong thanh ghi $ra.

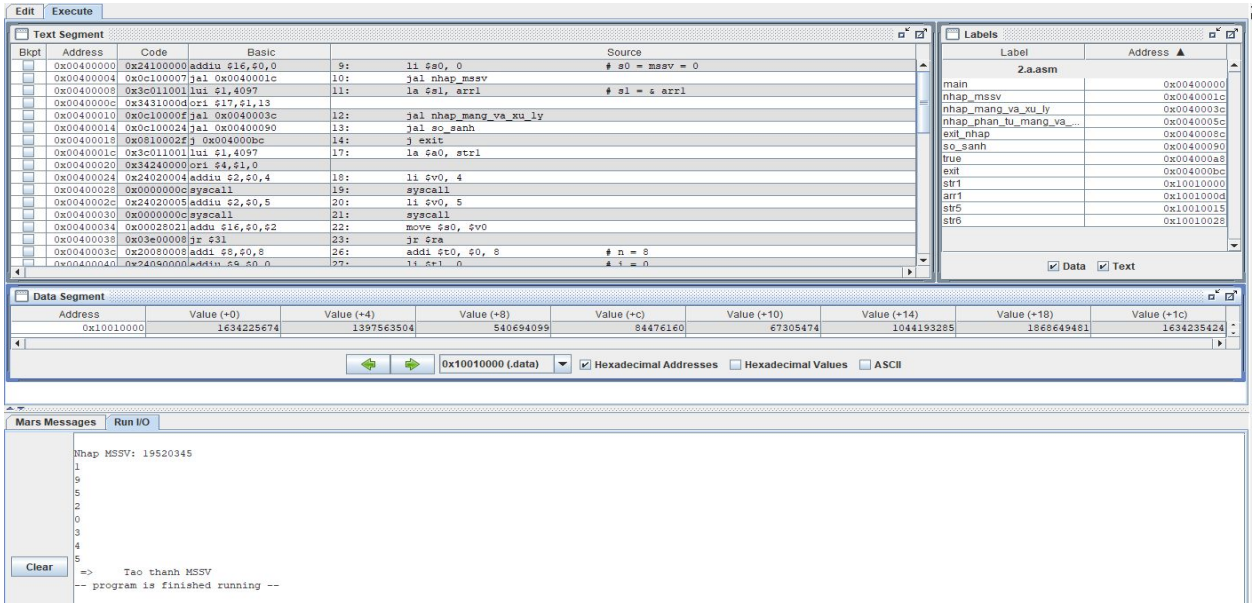
- Thoát chương trình

Kết quả kiểm tra bằng Mars:

TH1: MSSV : 19520345.

Chuỗi chữ số : 1, 9, 5, 2, 0, 3, 4, 5.

→ Kết quả mong đợi: "=>Tao thanh MSSV."

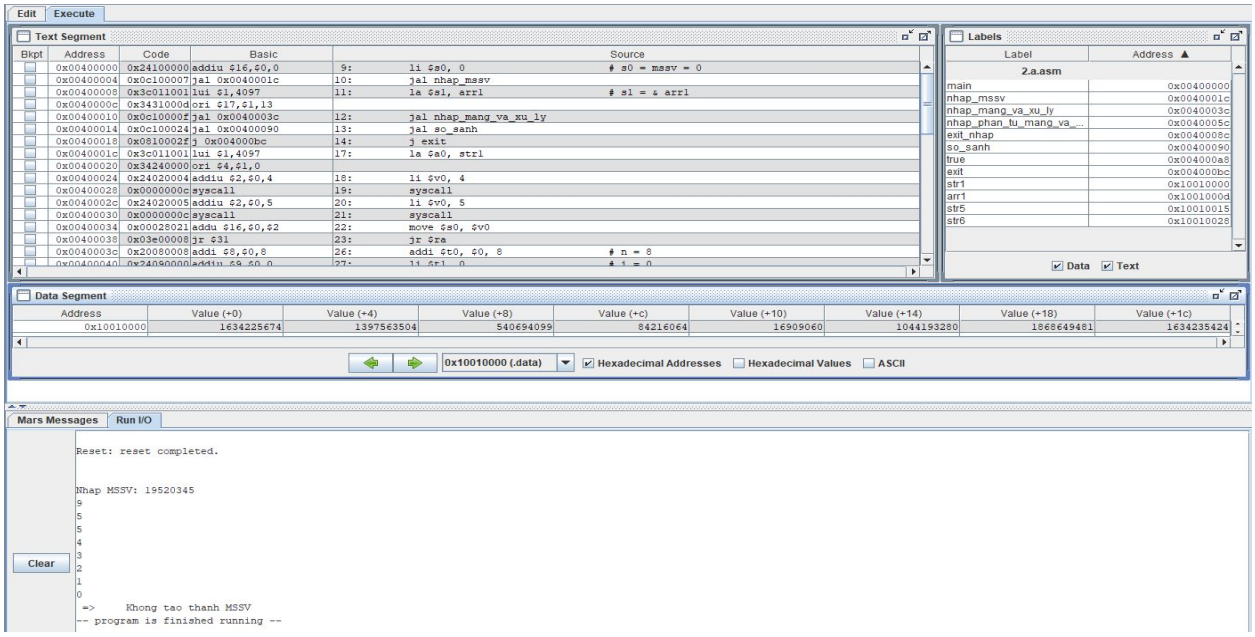


⇒ Đầu ra thực tế giống như đầu ra mong muốn.

TH2: MSSV: 19520345

Chuỗi chữ số: 9, 5, 5, 4, 3, 2, 1, 0.

→ Kết quả mong đợi: "=> Khong tao thanh MSSV."



⇒ Đầu ra thực tế giống như đầu ra mong muốn.

b. Kiểm tra 8 số đó có thể tạo thành MSSV của bạn.(3đ)

STT	***Chương trình***	***Ý nghĩa***
1	.data arr1: .space 8 arr2: .space 8 str1: .asciiz "=> Tao thanh MSSV" str2: .asciiz "=> Khong tao thanh MSSV" str3: .asciiz "\nNhap MSSV: " enter: .asciiz "\n" space: .asciiz " " .text main: li $s2, 0 # s2 = mssv = 0 jal nhap_mssv la $s0, arr1 # s0 = & arr1 jal nhap_mang_va_xu_ly jal tach_mssv_thanh_mang jal xu_ly_mang jal exit nhap_mssv: la $a0, str3 li $v0, 4 syscall li $v0, 5 syscall move $s2, $v0 jr $ra nhap_mang_va_xu_ly:	● Khai báo các biến cho chương trình sau chỉ thị này: ❖ arr1: là một chuỗi có 8 bytes bộ nhớ, chưa được khởi tạo. ❖ arr2: là một chuỗi có 8 bytes bộ nhớ, chưa được khởi tạo. ❖ str1 = " => Tao thanh MSSV.". ❖ str2 = " => Khong tao thanh MSSV". ❖ str3 = "Nhap MSSV: ". ❖ enter = '\n'. ❖ space = ' '. ● Viết câu lệnh sau chỉ thị này. ● Điểm bắt đầu chương trình chính: ❖ Khởi tạo thanh ghi có giá trị là 0. ❖ Lưu địa chỉ trở về vào $ra và nhảy tới hàm nhap_mssv. ❖ Khởi tạo thanh ghi với địa chỉ của arr1. ❖ Lưu địa chỉ trở về vào $ra và nhảy tới hàm nhap_mang_va_xu_ly. ❖ Lưu địa chỉ trở về vào $ra và nhảy tới hàm tach_mssv_thanh_mang. ❖ Lưu địa chỉ trở về vào $ra và nhảy tới hàm xu_ly_mang. ❖ Nhảy tới hàm thoát chương trình. ● Hàm nhap_mssv: ❖ Khởi tạo thanh ghi với địa chỉ của str3. ❖ Khởi tạo thanh ghi có giá trị là 4. ❖ Dùng để xuất chuỗi str3. ❖ Khởi tạo thanh ghi có giá trị là 5. ❖ Dùng để nhập số nguyên dương có 8 chữ số. ❖ Di chuyển giá trị $v0 vào $s2. ❖ Nhảy đến địa chỉ trong thanh ghi $ra. ● Hàm nhap_mang_va_xu_ly:

	li $t0, 0 # i = t0 = 0 addi $t1,$0,8 # n = t1 = 8 j nhap_phan_tu_mang_va_xu_ly	❖ Đưa giá trị 0 vào thanh ghi $t0. ❖ Đưa giá trị 8 vào thanh ghi $t1. ❖ Nhảy tới hàm nhap_phan_tu_mang_va_xu_ly.

	nhap_phan_tu_mang_va_xu_ly:
	beq $t0, $t1, exit_nhap # if (i == n)
	li $v0, 5 # read(arr1[i])
	
	syscall
	add $t2,$s0,$t0
	sb $v0, ($t2) # luu vao mang
	addi $t0, $t0, 1 # i = i + 1
	j nhap_phan_tu_mang_va_xu_ly
		● Hàm nhap_phan_tu_mang_va_xu_ly:
		❖ Nếu $t1 = $t0 thì nhảy tới hàm exit_nhap.
		❖ Ngược lại, khởi tạo thanh ghi có giá trị là 5.
		❖ Dùng để nhập giá trị phần tử thứ i.
		❖ Gán giá trị $t2 bằng tổng của $s0 với $t0.
		❖ Lưu giá trị $v0 vào $t2 (lưu vào mảng).
		❖ Vị trí phần tử mảng tăng lên một đơn vị.
		❖ Quay lại hàm nhap_phan_tu_mang_va_xu_ly.

	exit_nhap: jr $ra	● Hàm exit_nhap: ❖ Nhảy đến địa chỉ trong thanh ghi $ra.
	tach_mssv_thanh_mang: li $t0, 0 # gan i=0 addi $t5,$0,10 la $s1,arr2 # s1 = & arr2 j them_phan_tu_vao_mang	● Hàm tach_mssv_thanh_mang: ❖ Khởi tạo thanh ghi với giá trị là 0. ❖ Đưa giá trị 10 vào thanh ghi $t5. ❖ Khởi tạo thanh ghi với địa chỉ của ar2. ❖ Nhảy tới hàm them_phan_tu_vao_mang.

	them_phan_tu_vao_mang:
	beq $s2,$0,exit_them # if(mssv = 0)
	div $s2,$t5
	mflo $s2# s2 = mssv = mssv // 10
	
	mfhi $t4 # t4 = mssv % 10
	
	add $t2,$s1,$t0
	sb $t4,($t2) # luu vao mang
	addi $t0, $t0, 1 # i = i + 1
	j them_phan_tu_vao_mang
		● Hàm them_phan_tu_vao_mang:
		❖ Nếu $s2 = 0 thì nhảy tới hàm exit_them.
		❖ Ngược lại, lấy $s2 chia $t5.
		❖ Gán $s2 là phần nguyên sau khi thực hiện phép chia trên.
		❖ Gán $s4 là phần dư sau khi thực hiện phép chia trên.
		❖ Gán giá trị $t2 bằng tổng của $s1 với $t0.
		❖ Lưu giá trị vào mảng arr2.
		❖ Vị trí phần tử mảng tăng lên một đơn vị.
		❖ Quay lại hàm them_phan_tu_vao_mang.

	exit_them: jr $ra	● Hàm exit_them: ❖ Nhảy đến địa chỉ trong thanh ghi $ra.
	xu_ly_mang: li $t0, 0 # t0 = i = 0	● Hàm xu_ly_mang: ❖ Đưa giá trị 0 vào thanh ghi $t0.

```
        li $t1,8          # t1 = n = 8
        li $t7,-1         # t7 = -1
        addi $s3,$s3,0

        j kiem_tra

kiem_tra:
        beq $t0, $t1, exit_kiem_tra
# if (i == n) => ket thuc vong lap
        add $t2,$s0,$t0
        lb $t3, ($t2)
        li $t4,0          # t4 = i = 0
        j so_sanh

exit_kiem_tra:
        beq $s3,$t1, true
        li $v0,4

        la $a0,str2       # khong tao thanh mssv
        syscall
        jr $ra

true:             # tao thanh mssv
        li $v0,4
        la $a0,str1
        syscall
        jr $ra

so_sanh:
        beq $t4, $t1, exit_so_sanh
# if (i == n)
        add $t5,$s1,$t4
        lb $s2,($t5)
        beq $s2,$t3, tiep_tuc
        # neu s2 = t3 = mssv
        addi $t4, $t4, 1        # i = i + 1

        j so_sanh

tiep_tuc:
        sb $t7,($t5)
```

- ❖ Đưa giá trị 8 vào thanh ghi $t1.
- ❖ Đưa giá trị -1 vào thanh ghi $t7.
- ❖ Đưa giá trị của thanh ghi $s3 vào thanh ghi $s3.
- ❖ Nhảy tới hàm kiem_tra.

- Hàm kiem_tra:
  - ❖ Nếu $t0 = $t1 thì nhảy tới hàm exit_kiem_tra.
  - ❖ Gán giá trị $t2 bằng tổng của $s0 với $t0.
  - ❖ Lưu giá trị vào mảng.
  - ❖ Đưa giá trị 0 vào thanh ghi $t4.
  - ❖ Nhảy tới hàm so_sanh.

- Hàm exit_kiem_tra:
  - ❖ Nếu $s3 = $t1 thì nhảy qua hàm true.
  - ❖ Ngược lại, khởi tạo thanh ghi có giá trị là 4.
  - ❖ Khởi tạo thanh ghi với địa chỉ của str2.
  - ❖ Dùng để xuất chuỗi str2.
  - ❖ Nhảy đến địa chỉ trong thanh ghi $ra.

- Hàm true:
  - ❖ Khởi tạo thanh ghi có giá trị là 4.
  - ❖ Khởi tạo thanh ghi với địa chỉ của str1.
  - ❖ Dùng để xuất chuỗi str1.
  - ❖ Nhảy đến địa chỉ trong thanh ghi $ra.

- Hàm so_sanh:
  - ❖ Nếu $t4 = $t1 thì nhảy tới hàm exit_so_sanh.
  - ❖ Gán giá trị $t5 bằng tổng của $s1 với $t4.
  - ❖ Lưu giá trị vào mảng.
  - ❖ Nếu $s2 = $t3 thì nhảy tới hàm tiep_tuc.
  - ❖ Ngược lại, vị trí phần tử mảng tăng lên một đơn vị.
  - ❖ Quay lại ham so_sanh.

- Hàm tiep_tuc:
  - ❖ Nếu bằng nhau, ta gán số trong mảng bằng -1.

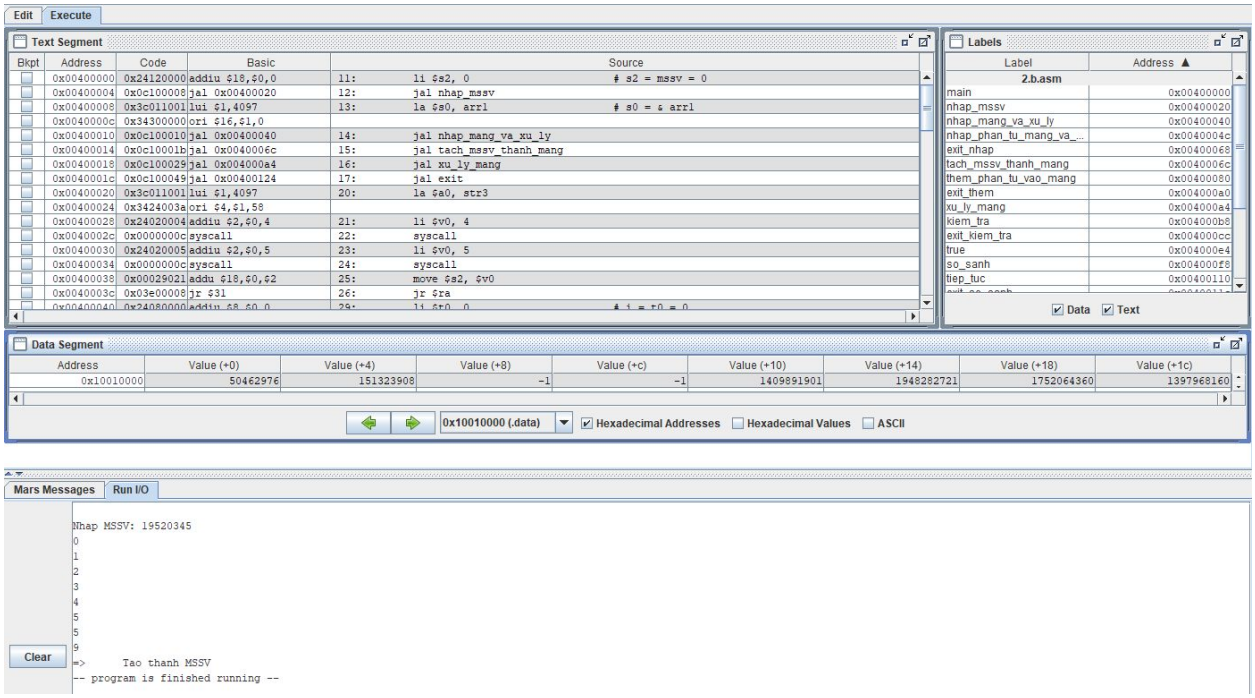
	add $s3,$s3,1 # s3 = s3 + 1
	j exit_so_sanh
	
	exit_so_sanh:
	addi $t0, $t0, 1 # t0 = t0 + 1
	j kiem_tra
	
	exit:
	li $v0, 10
	syscall
		❖ Tăng $s3 lên một đơn vị.
		❖ Nhảy tới hàm exit_so_sanh.
		
		● Hàm exit_so_sanh:
		❖ Tăng $t0 lên một đơn vị.
		❖ Nhảy tới hàm kiểm tra.
		
		● Hàm exit:
		❖ Thoát chương trình.

Kết quả kiểm tra bằng Mars:

TH1: MSSV : 19520345.

Chuỗi chữ số : 0, 1, 2, 3, 4, 5, 5, 9.

→ Kết quả mong đợi: "=>Tao thanh MSSV".

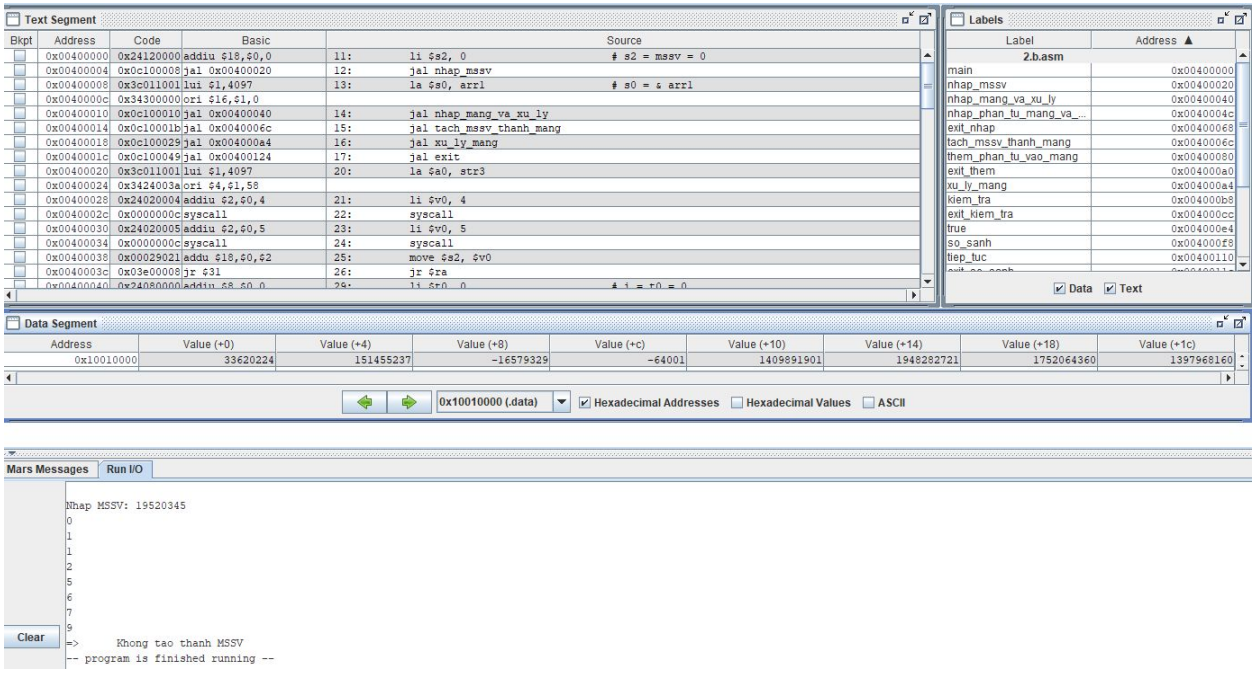


⇒ Đầu ra thực tế giống như đầu ra mong muốn.

TH2: MSSV: 19520345

Chuỗi chữ số: 0, 1, 1, 2, 5, 6, 7, 9.

→ Kết quả mong đợi: "=> Khong tao thanh MSSV".



⇒ Đầu ra thực tế giống như đầu ra mong muốn.

Câu 3: Nếu số nhập vào không là số nguyên dương, chương trình kết thúc với thông báo "invalid Entry"; nếu số nhập vào là nguyên dương, tên của từng chữ số được in ra và cách nhau một khoảng trắng. .(2đ)

Ví dụ: Nếu số nhập vào "728", in ra cửa sổ sẽ là "Seven Two Eight"

Ở câu này, để tiện cho việc kiểm tra số thì em sử dụng input là string thay vì nhập số, em ràng buộc chuỗi nhập vào chỉ tối đa 3 ký tự.

STT	*Chương trình*	*Ý nghĩa*
1	.data str1: .asciiz "Nhap so nguyen duong: " str2: .asciiz "\nKet qua: " str3: .asciiz "invalid Entry" n0: .asciiz "Zero " n1: .asciiz "One " n2: .asciiz "Two " n3: .asciiz "Three " n4: .asciiz "Four " n5: .asciiz "Five " n6: .asciiz "Six " n7: .asciiz "Seven "	● Khai báo các biến cho chương trình sau chỉ thị này: str1= "Nhap so nguyen duong: ". str2 = "Ket qua: ". str3 = "invalid Entry". n0 = "Zero". n1 = "One". n2 = "Two". n3 = "Three". n4 = "Four". n5 = "Five". n6 = "Six". n7 = "Seven".

```
n8:     .asciiz "Eight "
n9:     .asciiz "Nine "
input:  .space 4
        .text
main:


        la      $a0, str1
        li      $v0, 4
        syscall



        la      $a0, input
        li      $a1, 4
        li      $v0, 8
        syscall

        #Xuat "Ket qua: "
        la      $a0, str2
        li      $v0, 4
        syscall

        la      $s0, input
        li      $t4, 0
        li      $t5, 3

        jal     check_dieu_kien


check_dieu_kien:
        lb      $t0, 0($s0)
        beqz    $t0, exit_check
        addi    $t1, $t0, -10
        beqz    $t1, exit_check
        addi    $t4, $t4, 1
        addi    $s0, $s0, 1
        li      $t2, 47
        slt     $t6, $t2, $t0
        li      $t2, 58
        slt     $t7, $t0, $t2
        bne     $t6, $t7, exit_sai_input

        j       check_dieu_kien
```

n8 = “Eight”.
n9 = “Nine”.
input : là một chuỗi có 4 bytes bộ nhớ, chưa được khởi tạo

- Viết câu lệnh sau chỉ thị này.
- Điểm bắt đầu chương trình chính.
- Xuất câu lệnh str1:
  - ❖ Khởi tạo thanh ghi với địa chỉ của str1.
  - ❖ Khởi tạo thanh ghi có giá trị là 4.
  - ❖ Dùng để xuất chuỗi.

- Nhập chuỗi số:
  - ❖ Khởi tạo thanh ghi với địa chỉ của input
  - ❖ Khởi tạo thanh ghi có giá trị là 4.
  - ❖ Khởi tạo thanh ghi có giá trị là 8.
  - ❖ Dùng để nhập số nguyên dương.

- Xuất kết quả:
  - ❖ Khởi tạo thanh ghi với giá trị của str2.
  - ❖ Khởi tạo thanh ghi có giá trị là 4.
  - ❖ Dùng để xuất chuỗi str2.
- Kiểm tra điều kiện xử lý:
  - ❖ Khởi tạo thanh ghi với địa chỉ của input.
  - ❖ Khởi tạo thanh ghi có giá trị là 0.
  - ❖ Khởi tạo thanh ghi có giá trị là 3. (Nhập vào chỉ 3 ký tự).
  - ❖ Lưu địa chỉ trở về vào $ra và nhảy tới hàm check_dieu_kien.

- Hàm check_dieu_kien:
  - ❖ Lấy ký tự $t0 trong địa chỉ của $s0.
  - ❖ Nếu $t0 = ‘\0’ thì nhảy tới hàm exit_check.
  - ❖ Ngược lại, $t1 = $t0 - 10 (tức là $t1 = ‘\n’).
  - ❖ Nếu $t1 = ‘\n’ thì nhảy tới hàm exit_check.
  - ❖ Đếm bước nhảy ($t4 += 1).
  - ❖ Tăng vị trí địa chỉ tương ứng (&($s0) += 1).
  - ❖ Khởi tạo thanh ghi có giá trị là 47.
  - ❖ Nếu $t2 < $t0 thì $t6 = 1. Ngược lại, $t6 = 0.
  - ❖ Khởi tạo thanh ghi có giá trị là 58.
  - ❖ Nếu $t0 < $t2 thì $t7 = 1. Ngược lại, $t7 = 0.
  - ❖ Nếu $t7 != $t6 thì nhảy đến hàm exit_sai_input.
  - ❖ Quay lại hàm check_dieu_kien.

```
exit_check:
        la      $s0, input
        jal     in_chu_so


exit_sai_input:
        la      $a0, str3
        li      $v0, 4
        syscall

in_chu_so:
        bne     $t4, $t5, exit_sai_input


        lb      $t0, 0($s0)
        beqz    $t0, exit
        addi    $t1, $t0, -10
        beqz    $t1, exit
        move    $s1, $t0
        addi    $s0, $s0, 1
        jal     in_tung_so


in_tung_so:
        li      $t0, '0'
        beq     $s1, $t0, true_0
        li      $t0, '1'
        beq     $s1, $t0, true_1
        li      $t0, '2'
        beq     $s1, $t0, true_2
        li      $t0, '3'
        beq     $s1, $t0, true_3
        li      $t0, '4'
        beq     $s1, $t0, true_4
        li      $t0, '5'
        beq     $s1, $t0, true_5
        li      $t0, '6'
        beq     $s1, $t0, true_6
        li      $t0, '7'
        beq     $s1, $t0, true_7
        li      $t0, '8'
```

- Hàm exit_check:
  - ❖ Khởi tạo thanh ghi với địa chỉ của input.
  - ❖ Lưu địa chỉ trở về vào $ra và nhảy đến hàm in_chu_so.

- Hàm exit_sai_input:
  - ❖ Khởi tạo thanh ghi với địa chỉ của str3.
  - ❖ Khởi tạo thanh ghi có giá trị là 4.
  - ❖ Dùng để xuất chuỗi str3.

- Hàm in_chu_so:
  - ❖ Kiểm tra số vừa nhập có đủ 3 ký tự không? Nếu $t4 != $t5 thì nhảy tới hàm exit_sai_input.
  - ❖ Lấy ký tự $t0 trong &($s0).
  - ❖ Nếu $t0 = ‘\0’ thì nhảy tới hàm exit.
  - ❖ Ngược lại, $t1 = $t0 - 10 ($t1 = ‘\n’).
  - ❖ Nếu $t1 = ‘\n’ thì nhảy tới hàm exit.
  - ❖ Di chuyển giá trị ở $t0 vào $s1.
  - ❖ Tăng vị trí địa chỉ tương ứng (&($s0) += 1).
  - ❖ Lưu địa chỉ trở về vào $ra và nhảy đến hàm in_tung_so.

- Hàm in_tung_so:
  - ❖ Khởi tạo thanh ghi có giá trị là ‘0’.
  - ❖ Nếu $s1 = $t0 thì nhảy tới hàm true_0.
  - ❖ Ngược lại, khởi tạo thanh ghi có giá trị là ‘1’.
  - ❖ Nếu $s1 = $t0 thì nhảy tới hàm true_1.
  - ❖ Ngược lại, khởi tạo thanh ghi có giá trị là ‘2’.
  - ❖ Nếu $s1 = $t0 thì nhảy tới hàm true_2.
  - ❖ Ngược lại, khởi tạo thanh ghi có giá trị là ‘3’.
  - ❖ Nếu $s1 = $t0 thì nhảy tới hàm true_3.
  - ❖ Ngược lại, khởi tạo thanh ghi có giá trị là ‘4’.
  - ❖ Nếu $s1 = $t0 thì nhảy tới hàm true_4.
  - ❖ Ngược lại, khởi tạo thanh ghi có giá trị là ‘5’.
  - ❖ Nếu $s1 = $t0 thì nhảy tới hàm true_5.
  - ❖ Ngược lại, khởi tạo thanh ghi có giá trị là ‘6’.
  - ❖ Nếu $s1 = $t0 thì nhảy tới hàm true_6.
  - ❖ Ngược lại, khởi tạo thanh ghi có giá trị là ‘7’.
  - ❖ Nếu $s1 = $t0 thì nhảy tới hàm true_7.
  - ❖ Ngược lại, khởi tạo thanh ghi có giá trị là ‘8’.

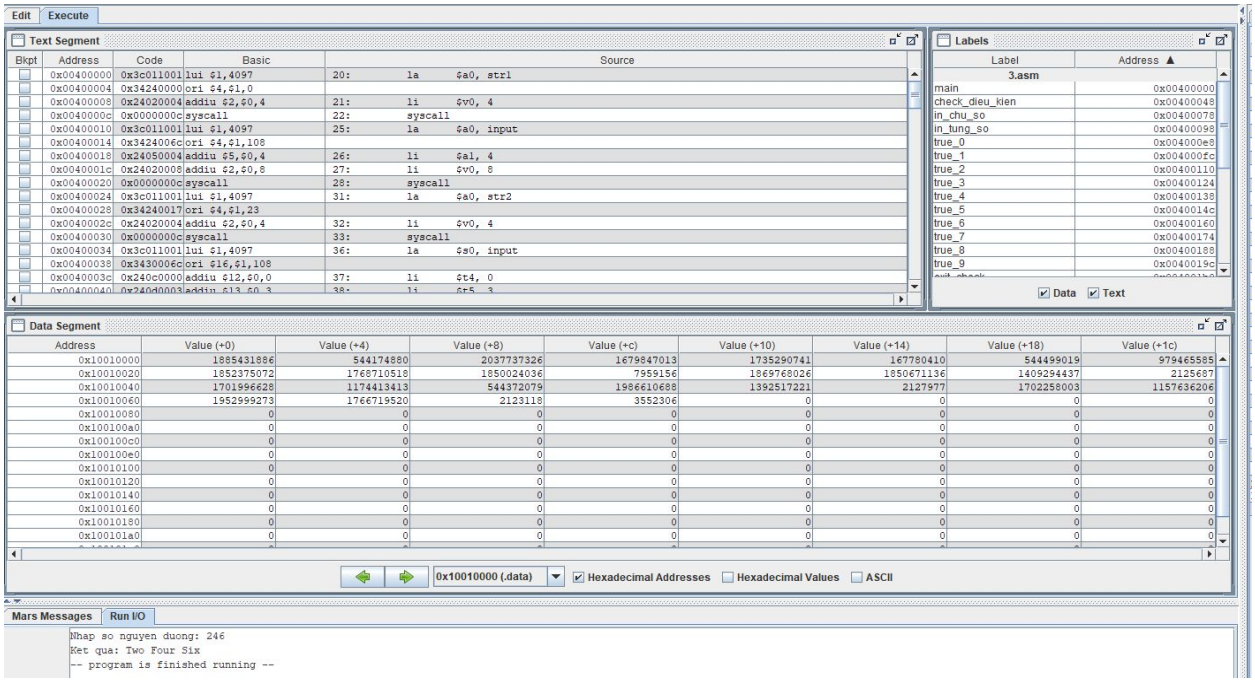
	beq $s1, $t0, true_8 li $t0, '9' beq $s1, $t0, true_9	❖ Nếu $s1 = $t0 thì nhảy tới hàm true_8. ❖ Ngược lại, khởi tạo thanh ghi có giá trị là '9'. ❖ Nếu $s1 = $t0 thì nhảy tới hàm true_9.
	true_0: la $a0, n0 li $v0, 4 syscall j in_chu_so	● Hàm true_0: ❖ Khởi tạo thanh ghi với giá trị của n0. ❖ Khởi tạo thanh ghi có giá trị là 4. ❖ Dùng để xuất chuỗi n0. ❖ Nhảy tới hàm in_chu_so.
	true_1: la $a0, n1 li $v0, 4 syscall j in_chu_so	● Hàm true_1: ❖ Khởi tạo thanh ghi với giá trị của n1. ❖ Khởi tạo thanh ghi có giá trị là 4. ❖ Dùng để xuất chuỗi n1. ❖ Nhảy tới hàm in_chu_so.
	true_2: la $a0, n2 li $v0, 4 syscall j in_chu_so	● Hàm true_2: ❖ Khởi tạo thanh ghi với giá trị của n2. ❖ Khởi tạo thanh ghi có giá trị là 4. ❖ Dùng để xuất chuỗi n2. ❖ Nhảy tới hàm in_chu_so.
	true_3: la $a0, n3 li $v0, 4 syscall j in_chu_so	● Hàm true_3: ❖ Khởi tạo thanh ghi với giá trị của n3. ❖ Khởi tạo thanh ghi có giá trị là 4. ❖ Dùng để xuất chuỗi n3. ❖ Nhảy tới hàm in_chu_so.
	true_4: la $a0, n4 li $v0, 4 syscall j in_chu_so	● Hàm true_4: ❖ Khởi tạo thanh ghi với giá trị của n4. ❖ Khởi tạo thanh ghi có giá trị là 4. ❖ Dùng để xuất chuỗi n4. ❖ Nhảy tới hàm in_chu_so.
	true_5: la $a0, n5 li $v0, 4 syscall j in_chu_so	● Hàm true_5: ❖ Khởi tạo thanh ghi với giá trị của n5. ❖ Khởi tạo thanh ghi có giá trị là 4. ❖ Dùng để xuất chuỗi n5. ❖ Nhảy tới hàm in_chu_so.
	true_6: la $a0, n6	● Hàm true_6: ❖ Khởi tạo thanh ghi với giá trị của n6.

	li $v0, 4 syscall j in_chu_so true_7: la $a0, n7 li $v0, 4 syscall j in_chu_so true_8: la $a0, n8 li $v0, 4 syscall j in_chu_so true_9: la $a0, n9 li $v0, 4 syscall j in_chu_so exit: li $v0, 10 syscall	❖ Khởi tạo thanh ghi có giá trị là 4. ❖ Dùng để xuất chuỗi n6. ❖ Nhảy tới hàm in_chu_so. ● Hàm true_7: ❖ Khởi tạo thanh ghi với giá trị của n7. ❖ Khởi tạo thanh ghi có giá trị là 4. ❖ Dùng để xuất chuỗi n7. ❖ Nhảy tới hàm in_chu_so. ● Hàm true_8: ❖ Khởi tạo thanh ghi với giá trị của n8. ❖ Khởi tạo thanh ghi có giá trị là 4. ❖ Dùng để xuất chuỗi n8. ❖ Nhảy tới hàm in_chu_so. ● Hàm true_9: ❖ Khởi tạo thanh ghi với giá trị của n9. ❖ Khởi tạo thanh ghi có giá trị là 4. ❖ Dùng để xuất chuỗi n9. ❖ Nhảy tới hàm in_chu_so. ● Thoát khỏi chương trình.

Kết quả kiểm tra bằng Mars:

TH1: Chuỗi nhập vào là 246

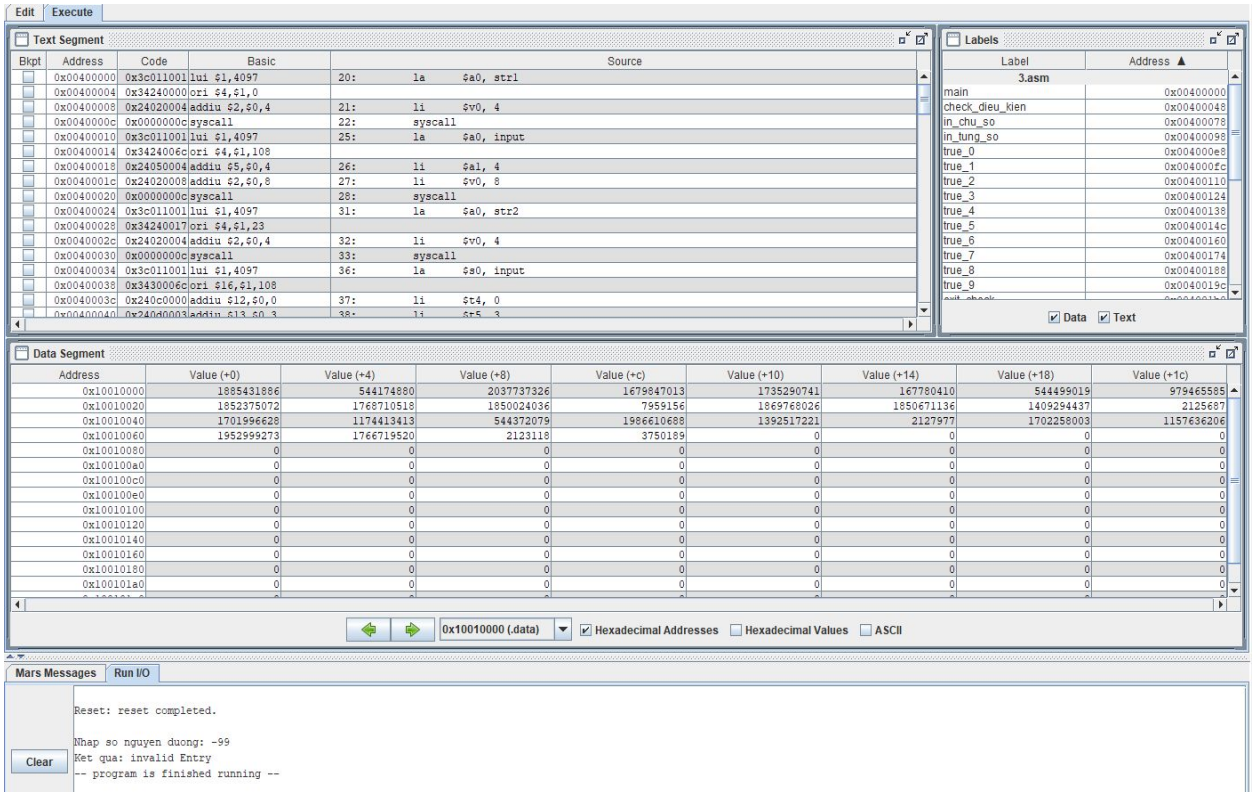
→Kết quả mong muốn: two four six



⇒ Đầu ra thực tế giống như đầu ra mong muốn.

TH2: Chuỗi nhập vào là -99

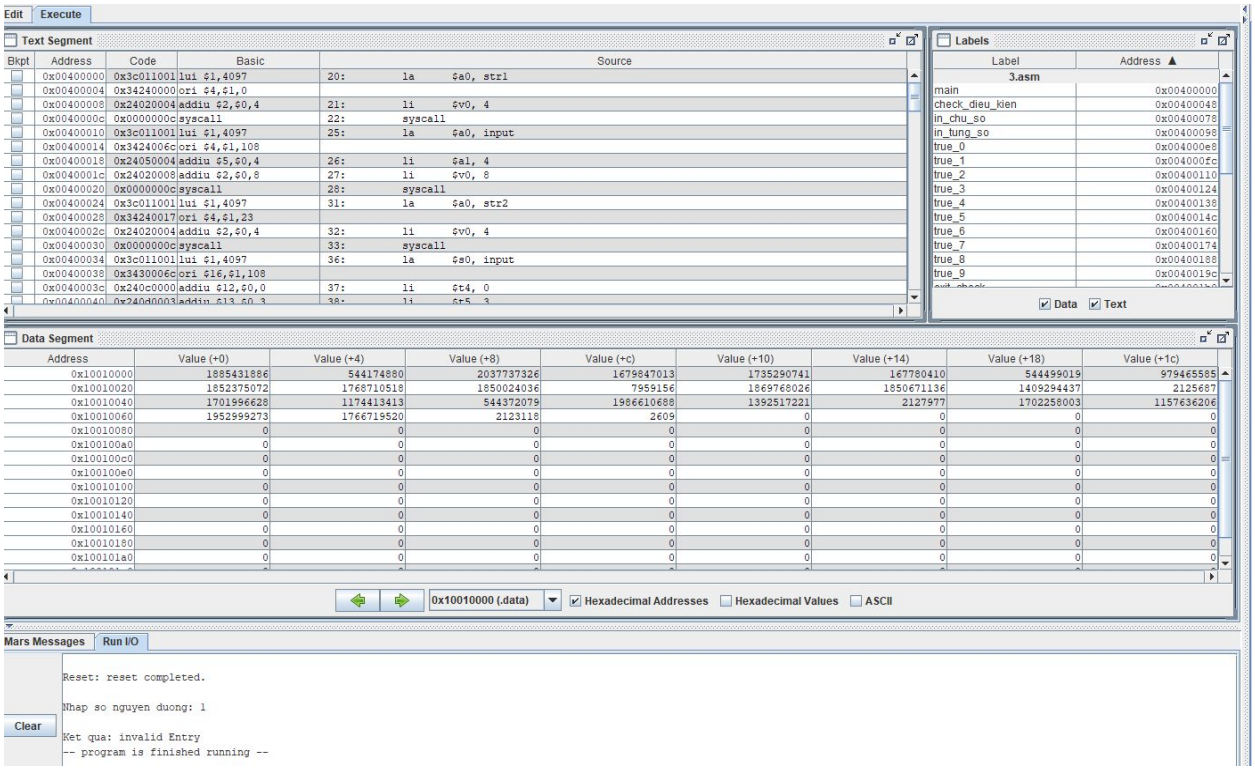
→Kết quả mong muốn : invalid Entry



⇒ Đầu ra thực tế giống như đầu ra mong muốn.

TH3: Chuỗi nhập vào là 1

Kết quả mong muốn : invalid Entry



⇒ Đầu ra thực tế giống như đầu ra mong muốn.