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

****

BÁO CÁO PROJECT
THỰC HÀNH KIẾN TRÚC MÁY TÍNH

Giáo viên hướng dẫn: ThS. Lê Bá Vui

Mã lớp: 139363 – Kỳ 2022.2

Nhóm: 5

Sinh viên thực hiện: Nguyễn Văn Cường (20215006)
Bùi Anh Đức (20210195)

Hà Nội, ngày 07 tháng 07 năm 2023

Contents

I. Project 1 *(Người thực hiện: Nguyễn Văn Cường – 20215006)*

1. <u>Đề bài:</u> Curiosity Marsbot

- Xe tự hành Curiosity Marsbot chạy trên sao Hỏa, được vận hành từ xa bởi các lập trình viên trên Trái Đất.
- Bằng cách gửi đi các mã điều khiển từ một bàn phím ma trận, lập trình viên điều khiển quá trình di chuyển của Marbot như sau:

Mã điều khiển	**Ý nghĩa**
1b4	Marbot bắt đầu chuyển động
c68	Marbot đứng im
444	Rẽ trái 90 độ so với phương chuyển động gần đây và giữ hướng mới
666	Rẽ phải 90 độ so với phương chuyển động gần đây và giữ hướng mới
dad	Bắt đầu để lại vết trên đường
cbc	Chấm dứt để lại vết trên đường
999	Tự động quay trở lại theo lộ trình ngược lại. Không vẽ vết, không nhận mã khác cho tới khi kết thúc lộ trình ngược. Mô tả: Marsbot được lập trình để nhớ lại toàn bộ lịch sử các mã điều khiển và khoảng thời gian giữa các lần đổi mã. Vì vậy, nó có thể đảo ngược lại lộ trình để quay về điểm xuất

- Sau khi nhận mã điều khiển, Curioisity Marsbot sẽ không xử lý ngay, mà phải đợi lệnh kích hoạt mã từ bàn phím Keyboard & Display MMIO Simulator. Có 2 lệnh như vậy:

Kích hoạt mã	**Ý nghĩa**
Phím Enter	Kết thúc nhập mã và yêu cầu Marsbot thực thi
Phím Del	Xóa toàn bộ mã điều khiển đang nhập dở dang
Phím Space	Lặp lại lệnh đã thực hiện trước đó

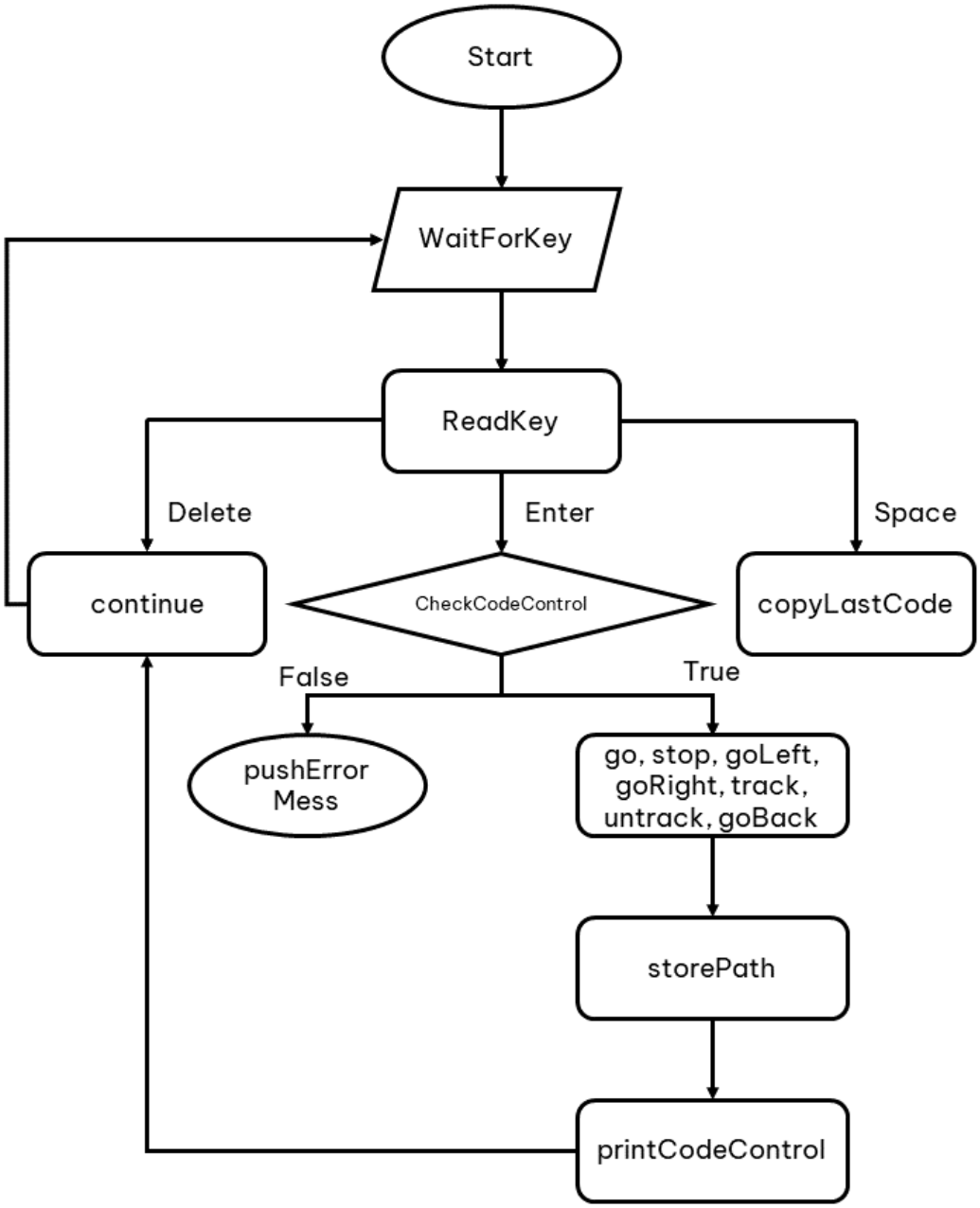
- Hãy lập trình để Marsbot có thể hoạt động như đã mô tả.
- Đồng thời bổ sung thêm tính năng: mỗi khi gửi một mã điều khiển cho Marsbot, hiển thị mã đó lên màn hình console để người xem có thể giám sát lộ trình của xe.

2. <u>Phân tích cách thực hiện:</u>

a) Mô tả

- Bước 1: Mỗi khi người dùng nhập 1 kí tự từ Digital Lab Sim sẽ lưu kí tự được nhập vào bộ nhớ (***inputControlCode*** và ***latestCode***).
- Bước 2: Kiểm tra liên tục xem kí tự Enter có được nhập ở Keyboard & Display MMIO Simulator hay không. Nếu người dùng nhập Del, chuyển tới bước 5. Nếu người dùng nhập Enter, chuyển tới bước 4. Nếu người dùng nhập Space, chuyển tới bước 3.
- Bước 3: Sao chép ***latestCode*** vào ***inputControlCode***. Chuyển tới bước 4.
- Bước 4: Kiểm tra xem đoạn code điều khiển có hợp lệ không (gồm 3 kí tự), nếu không sẽ thông báo code lỗi và sang bước 5. Nếu hợp lệ thì chuyển sang bước 6.
- Bước 5: Xóa toàn bộ mã điều khiển đang nhập. Quay trở lại bước 1.
- Bước 6: Marsbot thực hiện các yêu cầu theo lệnh. Lưu lại đường đi (nếu cần).
- Bước 7: In ra console code điều khiển đã nhập, đồng thời xóa luôn mã điều khiển. Quay trở lại bước 1.

b) Lưu đồ thuật toán



3. <u>Ý nghĩa các hàm trong mã nguồn:</u>

a) CheckControlCode

Ý nghĩa: Kiểm tra ***inputControlString*** có trùng với các Control Code hay không:

- Bước 1: Kiểm tra ***lengthControlCode***, nếu bằng 3, chuyển tới bước 2. Ngược lại, hiển thị lỗi "*Wrong Control Code*".
- Bước 2: Kiểm tra nội dung ***inputControlCode*** có trùng với các Control Code (địa chỉ lưu trong thanh ghi $s3) hay không bằng hàm ***isEqualString***. Nếu có, thực thi các hàm điều khiển Marsbot tương ứng. Ngược lại, thông báo lỗi.

b) *isEqualString*

Ý nghĩa: so sánh 2 xâu

Lần lượt so sánh các kí tự trong 2 xâu này. Nếu 2 xâu bằng nhau thì gán thanh ghi $t0 giá trị là 1, ngược lại là 0.

c) *pushErrorMess*

Ý nghĩa: hiện thông báo dialog khi người dùng nhập code điều khiển không đúng.

Sử dụng các hàm syscall 4, 55.

d) *printControlCode*

Ý nghĩa: mỗi khi gửi một mã điều khiển cho Marsbot, hiển thị mã đó lên màn hình console để người xem có thể giám sát lộ trình của xe.

e) *removeControlCode*

Ý nghĩa: xóa ***inputControlString***

Sử dụng vòng lặp, lần lượt gán các kí tự từ 0 tới ***lengthControlCode*** (độ dài của xâu hiện tại) bằng '\0'.

Sau đó update ***lengthControlCode*** = 0

f) *copyLastCode*

Copy các ký tự của ***latestCode*** vào trong ***InputControlCode***. Được gọi khi ấn phím Space để lặp lại lệnh đã thực hiện trước đó

g) *GO* và *STOP*

Ý nghĩa: điều khiển Marsbot bắt đầu chuyển động (GO) hoặc dừng lại (STOP).

Load 1 vào địa chỉ MOVING (0xffff8050) nếu muốn Marsbot chuyển động và load 0 nếu muốn Marsbot dừng lại.

h) *goRight* và *goLeft*

Ý nghĩa: điều khiển Marsbot quay và di chuyển sang phải (với hàm goRight) hoặc trái (goLeft) một góc 90 độ.

Đầu vào: biến ***nowHeading***

Muốn di chuyển sang phải 90 độ so với hướng hiện tại ta chỉ cần tăng biến nowHeading thêm 90 dộ, đối với bên trái là giảm đi 90 độ. Sau đó gọi hàm ROTATE để thực hiện thay đổi.

i) *ROTATE*

Ý nghĩa: quay Marsbot theo hướng được lưu trong nowHeading

Đầu vào: biến ***nowHeading***

Load biến nowHeading và lưu giá trị vào địa chỉ HEADING (0xffff8010) để Marsbot chuyển hướng.

j) TRACK và UNTRACK

Ý nghĩa: điều khiển Marsbot bắt đầu để lại vết (TRACK) hoặc kết thúc để lại vết (UNTRACK)

Load 1 vào địa chỉ LEAVETRACK (0xffff8020) nếu muốn để lại vết và load 0 nếu muốn kết thúc vết.

k) goBack

Ý nghĩa: điều khiển Marsbot đi ngược lại theo lộ trình nó đã đi và không để lại vết

Đầu vào: mảng path lưu thông tin đường đi, biến ***lengthPath*** lưu kích cỡ của mảng history theo byte.

Mảng path: lưu thông tin về cạnh đường đi. Mỗi cạnh gồm 3 số nguyên: tọa độ x và y và hướng đi z. Do đó mỗi thông tin đường đi sẽ chiếm 12 byte (3 words x 4 byte). Do đó ***lengthPath*** sẽ có giá trị là bội của 12.

Mỗi khi muốn quay ngược lại và đi về điểm đầu tiên của 1 cạnh trên đường đi, ta sẽ đảo ngược hướng đã thực hiện (bằng cách tăng thêm 180 độ) và di chuyển đến khi nào gặp điểm có tọa độ đã lưu thì kết thúc việc đi ngược trên cạnh đó, tiếp tục trên cạnh khác. Dừng lại khi Marbot quay lại vị trí xuất phát.

l) storePath

Ý nghĩa: Lưu lại thông tin về đường đi của Marsbot vào mảng path.

Đầu vào: biến ***nowHeading***, ***lengthPath***, ***WHEREX***, ***WHEREY***.

Mảng path lưu thông tin về đường đi hay đúng hơn là thông tin về các cạnh của đường đi của Marsbot. Mỗi một cạnh gồm 3 thông tin: tọa độ x (***WHEREX***) và y (***WHEREY***) của điểm đầu tiên, z (***nowHeading***) là hướng đi của cạnh đó.

m) Các thao tác trong phần xử lí interrupt (địa chỉ cố định 0x80000180)

Lần lượt quét các hàng của Digital Lab Sim để xem phím nào được bấm bằng cách so sánh với các giá trị được lưu trong ***Key value***. Tiếp đó dựa vào mã được trả về ghi kí tự tương ứng vào bộ nhớ.

Sau khi kết thúc chương trình ngắt, sử dụng lệnh eret để quay trở lại chương trình chính. Lệnh eret sẽ gán nội dung thanh ghi PC bằng giá trị trong thanh ghi $14 (epc).

Vì thanh ghi PC vẫn chứa địa chỉ của lệnh mà ngắt xảy ra, tức là lệnh đã thực hiện xong, chứ không chứa địa chỉ của lệnh kế tiếp. Bởi vậy cần lập trình để tăng địa chỉ chứa trong thanh ghi epc.

4. <u>Mã nguồn:</u> nằm trong file ***n01_g05_NguyenVanCuong.asm***

```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014
.eqv KEY_CODE 0xFFFF0004   # ASCII code from keyboard, 1 byte
.eqv KEY_READY 0xFFFF0000  # =1 if has a new keycode ?
                           # Auto clear after lw
#-----------------------------------------------------------
-----------------
# Marsbot
.eqv HEADING 0xffff8010 # Integer: An angle between 0 and 359
                        # 0 : North (up)
                        # 90: East (right)
                        # 180: South (down)
                        # 270: West (left)
.eqv MOVING 0xffff8050 # Boolean: whether or not to move
.eqv LEAVETRACK 0xffff8020 # Boolean (0 or non-0):
                        # whether or not to leave a track
.eqv WHEREX 0xffff8030 # Integer: Current x-location of MarsBot
.eqv WHEREY 0xffff8040 # Integer: Current y-location of MarsBot

#===========================================================
============================
#===========================================================
============================
.data
# Key value
    .eqv KEY_0 0x11
    .eqv KEY_1 0x21
    .eqv KEY_2 0x41
    .eqv KEY_3 0x81
    .eqv KEY_4 0x12
    .eqv KEY_5 0x22
    .eqv KEY_6 0x42
    .eqv KEY_7 0x82
    .eqv KEY_8 0x14
    .eqv KEY_9 0x24
```

```
	.eqv KEY_a 0x44
	.eqv KEY_b 0x84
	.eqv KEY_c 0x18
	.eqv KEY_d 0x28
	.eqv KEY_e 0x48
	.eqv KEY_f 0x88
#-------------------------------------------------------------
-----------------
#Control code
	MOVE_CODE: .asciiz "1b4"
	STOP_CODE: .asciiz "c68"
	GO_LEFT_CODE: .asciiz "444"
	GO_RIGHT_CODE: .asciiz "666"
	TRACK_CODE: .asciiz "dad"
	UNTRACK_CODE: .asciiz "cbc"
	GO_BACK_CODE: .asciiz "999"
	WRONG_CODE: .asciiz "Wrong control code!"
#-------------------------------------------------------------
-----------------
	inputControlCode: .space 50    #input
	lengthControlCode: .word 0
	latestCode: .space 50
	nowHeading: .word 0
#-------------------------------------------------------
# duong di cua marsbot duoc luu tru vao mang path
# moi 1 canh duoc luu tru duoi dang 1 structure
# 1 structure co dang {x, y, z}
# trong do:	x, y la toa do diem dau tien cua canh
#		z la huong cua canh do
# mac dinh:	structure dau tien se la {0,0,0}
# do dai duong di ngay	khi bat dau la 12 bytes (3x 4byte)
#-------------------------------------------------------
	path: .space 600
	lengthPath: .word 12		#bytes

#=============================================
============================
#=============================================
============================
.text
```

```
main:
	li $k0, KEY_CODE
	li $k1, KEY_READY
#-----------------------------------------------------------
# Enable the interruption of Keyboard matrix 4x4 of Digital Lab Sim
#-----------------------------------------------------------
	li $t1, IN_ADRESS_HEXA_KEYBOARD
	li $t3, 0x80 # bit 7 = 1 to enable
	sb $t3, 0($t1)
#-----------------------------------------------------------
loop:	nop
WaitForKey:	lw $t5, 0($k1)		#$t5 = [$k1] = KEY_READY
		beq $t5, $zero, WaitForKey	#if $t5 == 0 then Polling
		nop
		beq $t5, $zero, WaitForKey
ReadKey:	lw $t6, 0($k0)		#$t6 = [$k0] = KEY_CODE
		beq $t6, 127 , continue		#if $t6 == delete key then remove input
					#127 is delete key in ascii

		beq $t6, 32, copyLastCode	#if $t6 == space key then repeat the last action

		bne $t6, '\n' , loop		#if $t6 != '\n' then Polling
		nop
		bne $t6, '\n' , loop
		j CheckControlCode		#if $t6 = '\n'(enter) check Control Code

CheckControlCode:
		la $s2, lengthControlCode
		lw $s2, 0($s2)
		#---------------
		bne $s2, 3, pushErrorMess


		la $s3, MOVE_CODE
		jal isEqualString
		beq $t0, 1, go

		la $s3, STOP_CODE
```

```
            jal isEqualString
            beq $t0, 1, stop


            la $s3, GO_LEFT_CODE
            jal isEqualString
            beq $t0, 1, goLeft

            la $s3, GO_RIGHT_CODE
            jal isEqualString
            beq $t0, 1, goRight

            la $s3, TRACK_CODE
            jal isEqualString
            beq $t0, 1, track


            la $s3, UNTRACK_CODE
            jal isEqualString
            beq $t0, 1, untrack


            la $s3, GO_BACK_CODE
            jal isEqualString
            beq $t0, 1, goBack

            beq $t0, 0, pushErrorMess

printControlCode:
        li $v0, 4
        la $a0, inputControlCode
        syscall
        nop
        li      $a0, '\n'
        li      $v0, 11
        syscall
        nop


continue:
```

```
        jal removeControlCode
        nop
        j loop
        nop
        j loop

#
================================================================
================================
# -------------- procedure to copy the last code to re-execute --------
-----------
#
================================================================
================================
copyLastCode:
        # ----- store register -----
        addi $sp, $sp, 4
        sw    $a0, 0($sp)
        addi $sp, $sp, 4
        sw    $v0, 0($sp)
        addi $sp, $sp, 4
        sw    $t1, 0($sp)
        addi $sp, $sp, 4
        sw    $t2, 0($sp)
        addi $sp, $sp, 4
        sw    $t3, 0($sp)
        # ----- procedure -----
        la     $t1, inputControlCode
        la     $t2, latestCode
        lb     $t3, 0($t2)       # $t3 = code[0]
        sb     $t3, 0($t1)       # code[0] = lastcode[0]
        lb     $t3, 1($t2)       # $t3 = code[1]
        sb     $t3, 1($t1) # code[1] = lastcode[1]
        lb     $t3, 2($t2)       # $t3 = code[2]
        sb     $t3, 2($t1)       # code[2] = lastcode[2]
        la     $t1, lengthControlCode
        li     $t3, 3
        sw    $t3, 0($t1)               # length = $t3 = 3
        # ----- restore register -----
        lw     $t3, 0($sp)
```

```
	addi $sp, $sp, -4
	lw	$t2, 0($sp)
	addi $sp, $sp, -4
	lw	$t1, 0($sp)
	addi $sp, $sp, -4
	lw	$v0, 0($sp)
	addi $sp, $sp, -4
	lw	$a0, 0($sp)
	addi $sp, $sp, -4

	nop
	j	CheckControlCode

#-------------------------------------------------------------
# storePath procedure, store path of marsbot to path variable
# param[in]	nowHeading variable
#		lengthPath variable
#-------------------------------------------------------------


storePath:
	#backup
	addi $sp,$sp,4
	sw $t1, 0($sp)
	addi $sp,$sp,4
	sw $t2, 0($sp)
	addi $sp,$sp,4
	sw $t3, 0($sp)
	addi $sp,$sp,4
	sw $t4, 0($sp)
	addi $sp,$sp,4
	sw $s1, 0($sp)
	addi $sp,$sp,4
	sw $s2, 0($sp)
	addi $sp,$sp,4
	sw $s3, 0($sp)
	addi $sp,$sp,4
	sw $s4, 0($sp)

	#processing
```

```
li $t1, WHEREX
lw $s1, 0($t1)          #s1 = x
li $t2, WHEREY
lw $s2, 0($t2)          #s2 = y

la $s4, nowHeading
lw $s4, 0($s4)          #s4 = now heading

la $t3, lengthPath
lw $s3, 0($t3)          #$s3 = lengthPath (dv: byte)

la $t4, path
add $t4, $t4, $s3       #position to store

sw $s1, 0($t4)          #store x
sw $s2, 4($t4)          #store y
sw $s4, 8($t4)          #store heading

addi $s3, $s3, 12       #update lengthPath
                    #12 = 3 (word) x 4 (bytes)
sw $s3, 0($t3)

#restore
lw $s4, 0($sp)
addi $sp,$sp,-4
lw $s3, 0($sp)
addi $sp,$sp,-4
lw $s2, 0($sp)
addi $sp,$sp,-4
lw $s1, 0($sp)
addi $sp,$sp,-4
lw $t4, 0($sp)
addi $sp,$sp,-4
lw $t3, 0($sp)
addi $sp,$sp,-4
lw $t2, 0($sp)
addi $sp,$sp,-4
lw $t1, 0($sp)
addi $sp,$sp,-4
```

```
        jr $ra
        nop
        jr $ra
#-----------------------------------------------------------
# goBack procedure, control marsbot go back
# param[in]      path array, lengthPath array
#-----------------------------------------------------------

goBack:    li $v0, 4
        la $a0, inputControlCode
        syscall
        nop
        jal UNTRACK
        la $s7, path
        la $s5, lengthPath
        lw $s5, 0($s5)
        add $s7, $s7, $s5      #vi tri hien tai
begin:      addi $s5, $s5, -12     #lui lai 1 structure

        addi $s7, $s7, -12     #vi tri cua thong tin ve canh cuoi cung
        lw $s6, 8($s7)         #huong cua canh cuoi cung
        addi $s6, $s6, 180     #nguoc lai huong cua canh cuoi cung


        la $t8, nowHeading  #nowHeading = S6
        sw $s6, 0($t8)         #nowHeading = s6
        jal ROTATE

go_to_first_point_of_edge:
        lw $t9, 0($s7)         #toa do x cua diem dau tien cua canh
        li $t8, WHEREX         #toa do x hien tai
        lw $t8, 0($t8)

        bne $t8, $t9, go_to_first_point_of_edge



        lw $t9, 4($s7)         #toa do y cua diem dau tien cua canh
        li $t8, WHEREY         #toa do y hien tai
        lw $t8, 0($t8)
```

```
        bne $t8, $t9, go_to_first_point_of_edge


        #Kiem tra xem Marbot da di ve diem dau cua canh hay chua

        beq $s5, 0, finish      #kiem tra xem marbot da o vi tri ban dau chua
        #nop
        #beq $s5, 0, finish

        j begin
        #nop
        #j goBack

finish:     jal STOP
        la $t8, nowHeading
        add $s6, $zero, $zero
        sw $s6, 0($t8)          #update nowHeading = 0
        la $t8, lengthPath
        sw $s5, 0($t8)          #update lengthPath = 0
        jal ROTATE
        j continue
#-----------------------------------------------------------
# track procedure, control marsbot to track and print control code
# param[in] none
#-----------------------------------------------------------

track:      jal TRACK
        j printControlCode
#-----------------------------------------------------------
# untrack procedure, control marsbot to untrack and print control code
# param[in] none
#-----------------------------------------------------------

untrack: jal UNTRACK
        j printControlCode
#-----------------------------------------------------------
# go procedure, control marsbot to go and print control code
# param[in] none
```

```
#-----------------------------------------------------------

go:   jal GO
      j printControlCode
#-----------------------------------------------------------
# stop procedure, control marsbot to stop and print control code
# param[in] none
#-----------------------------------------------------------

stop:       jal STOP
      j printControlCode
#-----------------------------------------------------------
# goRight procedure, control marsbot to go left and print control code
# param[in] none
#-----------------------------------------------------------

goRight:la $s5, nowHeading
      lw $s6, 0($s5)  #$s6 is heading at now
      addi $s6, $s6, 90 #increase heading by 90*
      sw $s6, 0($s5) # update nowHeading
      jal storePath
      jal ROTATE
      j printControlCode
#-----------------------------------------------------------
# goLeft procedure, control marsbot to go left and print control code
# param[in] none
#-----------------------------------------------------------

goLeft:     la $s5, nowHeading
      lw $s6, 0($s5)  #$s6 is heading at now
      addi $s6, $s6, -90 #increase heading by 90*
      sw $s6, 0($s5) # update nowHeading
      jal storePath
      jal ROTATE
      j printControlCode
#-----------------------------------------------------------
# removeControlCode procedure, to remove inputControlCode string
#                         inputControlCode = ""
# param[in] none
```

```
#-----------------------------------------------------------

removeControlCode:
	#backup
	addi $sp,$sp,4
	sw $t1, 0($sp)
	addi $sp,$sp,4
	sw $t2, 0($sp)
	addi $sp,$sp,4
	sw $s1, 0($sp)
	addi $sp,$sp,4
	sw $t3, 0($sp)
	addi $sp,$sp,4
	sw $s2, 0($sp)

	#processing
	la $s2, lengthControlCode
	lw $t3, 0($s2)				#$t3 = lengthControlCode
	addi $t1, $zero, -1			#$t1 = -1 = i
	addi $t2, $zero, 0			#$t2 = '\0'
	la $s1, inputControlCode
	addi $s1, $s1, -1
for_loop_to_remove:
	addi $t1, $t1, 1		#i++
	add $s1, $s1, 1			#$s1 = inputControlCode + i
	sb $t2, 0($s1)			#inputControlCode[i] = '\0'

	bne $t1, $t3, for_loop_to_remove	#if $t1 <=3 continue loop
	nop
	bne $t1, $t3, for_loop_to_remove

	add $t3, $zero, $zero
	sw $t3, 0($s2)				#lengthControlCode = 0

	#restore
	lw $s2, 0($sp)
	addi $sp,$sp,-4
	lw $t3, 0($sp)
	addi $sp,$sp,-4
	lw $s1, 0($sp)
```

```
        addi $sp,$sp,-4
        lw $t2, 0($sp)
        addi $sp,$sp,-4
        lw $t1, 0($sp)
        addi $sp,$sp,-4

        jr $ra
        nop
        jr $ra
#-------------------------------------------------------------
# isEqualString procedure, to check inputControlCode string
#                       is equal with string s (store in $s3 )
#                       Length of two string is the same
# param[in] $s3, store address of a string
# param[out] $t0, 1 if equal, 0 is not equal
#-------------------------------------------------------------

isEqualString:
        #backup
        addi $sp,$sp,4
        sw $t1, 0($sp)
        addi $sp,$sp,4
        sw $s1, 0($sp)
        addi $sp,$sp,4
        sw $t2, 0($sp)
        addi $sp,$sp,4
        sw $t3, 0($sp)

        #processing
        addi $t1, $zero, -1                     #$t1 = -1 = i
        add $t0, $zero, $zero
        la $s1, inputControlCode                #$s1 = inputControlCode
for_loop_to_check_equal:
        addi $t1, $t1, 1                #i++
        add $t2, $s1, $t1                       #$t2 = inputControlCode + i
        lb $t2, 0($t2)                          #$t2 = inputControlCode[i]

        add $t3, $s3, $t1                       #$t3 = s + i
        lb $t3, 0($t3)                          #$t3 = s[i]
```

```
        bne $t2, $t3, isNotEqual          #if $t2 != $t3 -> not equal

        bne $t1, 2, for_loop_to_check_equal       #if $t1 <=2 continue loop
        nop
        bne $t1, 2, for_loop_to_check_equal
isEqual:
        #restore
        lw $t3, 0($sp)
        addi $sp,$sp,-4
        lw $t2, 0($sp)
        addi $sp,$sp,-4
        lw $s1, 0($sp)
        addi $sp,$sp,-4
        lw $t1, 0($sp)
        addi $sp,$sp,-4

        add $t0, $zero, 1                         #update $t0
        jr $ra
        nop
        jr $ra
isNotEqual:
        #restore
        lw $t3, 0($sp)
        addi $sp,$sp,-4
        lw $t2, 0($sp)
        addi $sp,$sp,-4
        lw $s1, 0($sp)
        addi $sp,$sp,-4
        lw $t1, 0($sp)
        addi $sp,$sp,-4

        add $t0, $zero, $zero                     #update $t0
        jr $ra
        nop
        jr $ra
#-------------------------------------------------------------
# pushErrorMess procedure, to announce the inputed control code is wrong
# param[in] none
#-------------------------------------------------------------
```

```
pushErrorMess: li $v0, 4
        la $a0, inputControlCode
        syscall
        nop

        li $v0, 55
        la $a0, WRONG_CODE
        syscall
        nop
        nop
        j continue
        nop
        j continue
#-----------------------------------------------------------
# GO procedure, to start running
# param[in] none
#-----------------------------------------------------------
GO:  #backup
        addi $sp,$sp,4
        sw $at,0($sp)
        addi $sp,$sp,4
        sw $k0,0($sp)
        #processing
        li $at, MOVING          # change MOVING port
        addi $k0, $zero,1       #to logic 1,
        sb $k0, 0($at)          # to start running
        #restore
        lw $k0, 0($sp)
        addi $sp,$sp,-4
        lw $at, 0($sp)
        addi $sp,$sp,-4

        jr $ra
        nop
        jr $ra
#-----------------------------------------------------------
# STOP procedure, to stop running
# param[in] none
#-----------------------------------------------------------
STOP:       #backup
```

```
        addi $sp,$sp,4
        sw $at,0($sp)
        #processing
        li $at, MOVING          # change MOVING port to 0
        sb $zero, 0($at)        # to stop
        #restore
        lw $at, 0($sp)
        addi $sp,$sp,-4

        jr $ra
        nop
        jr $ra
#-----------------------------------------------------------
# TRACK procedure, to start drawing line
# param[in] none
#-----------------------------------------------------------
TRACK:    #backup
        addi $sp,$sp,4
        sw $at,0($sp)
        addi $sp,$sp,4
        sw $k0,0($sp)
        #processing
        li $at, LEAVETRACK  # change LEAVETRACK port
        addi $k0, $zero,1       # to logic 1,
        sb $k0, 0($at)          # to start tracking
        #restore
        lw $k0, 0($sp)
        addi $sp,$sp,-4
        lw $at, 0($sp)
        addi $sp,$sp,-4

        jr $ra
        nop
        jr $ra
#-----------------------------------------------------------
# UNTRACK procedure, to stop drawing line
# param[in] none
#-----------------------------------------------------------
UNTRACK:#backup
        addi $sp,$sp,4
```

```
        sw $at,0($sp)
        #processing
        li $at, LEAVETRACK  # change LEAVETRACK port to 0
        sb $zero, 0($at)        # to stop drawing tail
        #restore
        lw $at, 0($sp)
        addi $sp,$sp,-4

        jr $ra
        nop
        jr $ra
#-----------------------------------------------------------
# ROTATE_RIGHT procedure, to control robot to rotate
# param[in] nowHeading variable, store heading at present
#-----------------------------------------------------------
ROTATE:
        #backup
        addi $sp,$sp,4
        sw $t1,0($sp)
        addi $sp,$sp,4
        sw $t2,0($sp)
        addi $sp,$sp,4
        sw $t3,0($sp)
        #processing
        li $t1, HEADING  # change HEADING port
        la $t2, nowHeading
        lw $t3, 0($t2)          #$t3 is heading at now
        sw $t3, 0($t1)          # to rotate robot
        #restore
        lw $t3, 0($sp)
        addi $sp,$sp,-4
        lw $t2, 0($sp)
        addi $sp,$sp,-4
        lw $t1, 0($sp)
        addi $sp,$sp,-4

        jr $ra
        nop
        jr $ra
```

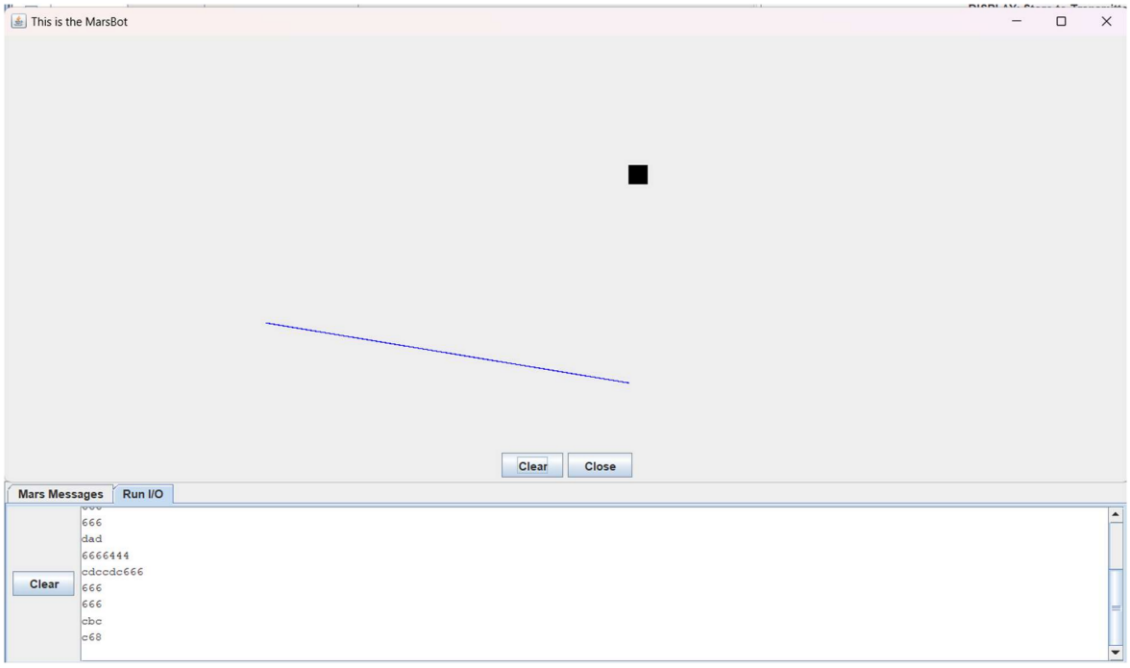
```
#============================================================
============================
# GENERAL INTERRUPT SERVED ROUTINE for all interrupts
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.ktext 0x80000180
#------------------------------------------------------
# SAVE the current REG FILE to stack
#------------------------------------------------------
backup:
        addi $sp,$sp,4
        sw $ra,0($sp)
        addi $sp,$sp,4
        sw $t1,0($sp)
        addi $sp,$sp,4
        sw $t2,0($sp)
        addi $sp,$sp,4
        sw $t3,0($sp)
        addi $sp,$sp,4
        sw $a0,0($sp)
        addi $sp,$sp,4
        sw $at,0($sp)
        addi $sp,$sp,4
        sw $s0,0($sp)
        addi $sp,$sp,4
        sw $s1,0($sp)
        addi $sp,$sp,4
        sw $s2,0($sp)
        addi $sp,$sp,4
        sw $t4,0($sp)
        addi $sp,$sp,4
        sw $s3,0($sp)
#------------------------------------------------------
# Processing
#------------------------------------------------------
get_cod:
        li $t1, IN_ADRESS_HEXA_KEYBOARD
        li $t2, OUT_ADRESS_HEXA_KEYBOARD
scan_row1:
        li $t3, 0x81
        sb $t3, 0($t1)
```

```
        lbu $a0, 0($t2)
        bnez $a0, get_code_in_char
scan_row2:
        li $t3, 0x82
        sb $t3, 0($t1)
        lbu $a0, 0($t2)
        bnez $a0, get_code_in_char
scan_row3:
        li $t3, 0x84
        sb $t3, 0($t1)
        lbu $a0, 0($t2)
        bnez $a0, get_code_in_char
scan_row4:
        li $t3, 0x88
        sb $t3, 0($t1)
        lbu $a0, 0($t2)
        bnez $a0, get_code_in_char
get_code_in_char:
        beq $a0, KEY_0, case_0
        beq $a0, KEY_1, case_1
        beq $a0, KEY_2, case_2
        beq $a0, KEY_3, case_3
        beq $a0, KEY_4, case_4
        beq $a0, KEY_5, case_5
        beq $a0, KEY_6, case_6
        beq $a0, KEY_7, case_7
        beq $a0, KEY_8, case_8
        beq $a0, KEY_9, case_9
        beq $a0, KEY_a, case_a
        beq $a0, KEY_b, case_b
        beq $a0, KEY_c, case_c
        beq $a0, KEY_d, case_d
        beq $a0, KEY_e, case_e
        beq $a0, KEY_f, case_f

        #$s0 store code in char type
case_0:    li $s0, '0'
        j store_code
case_1:    li $s0, '1'
        j store_code
```

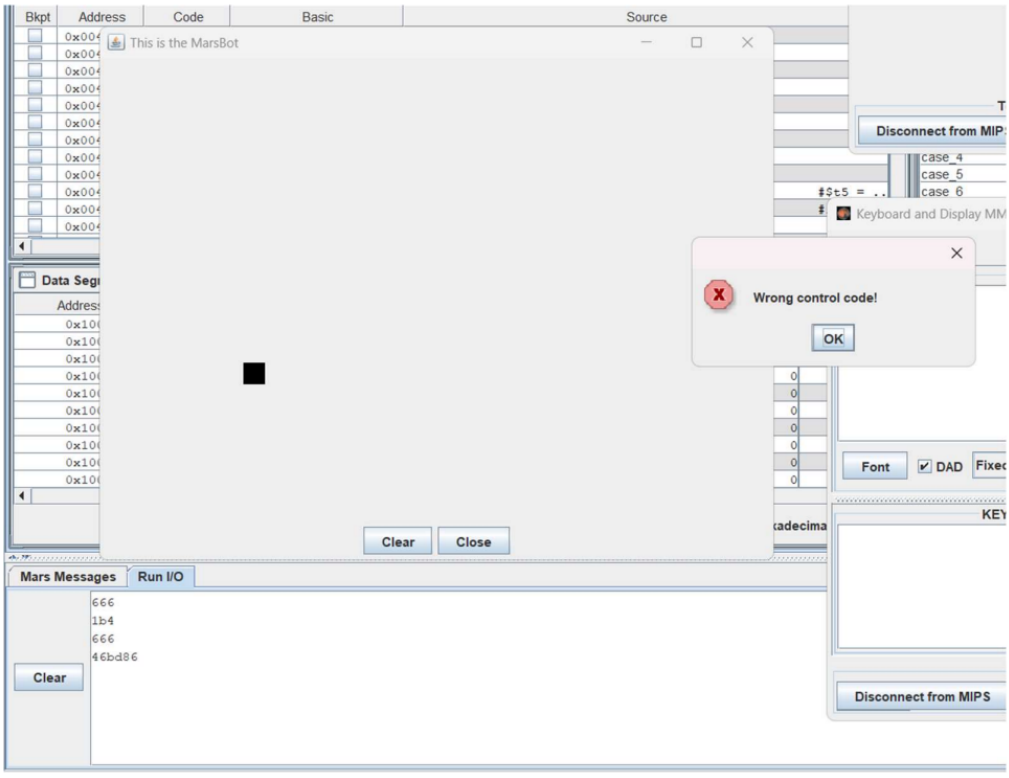
```
case_2:    li $s0, '2'
        j store_code
case_3:    li $s0, '3'
        j store_code
case_4:    li $s0, '4'
        j store_code
case_5:    li $s0, '5'
        j store_code
case_6:    li $s0, '6'
        j store_code
case_7:    li $s0, '7'
        j store_code
case_8:    li $s0, '8'
        j store_code
case_9:    li $s0, '9'
        j store_code
case_a:    li $s0, 'a'
        j store_code
case_b:    li $s0, 'b'
        j store_code
case_c:    li $s0, 'c'
        j store_code
case_d:    li $s0, 'd'
        j store_code
case_e:    li $s0,       'e'
        j store_code
case_f:    li $s0, 'f'
        j store_code
store_code:
        la $s1, inputControlCode
        la $s2, lengthControlCode
        la $s7, latestCode
        lw $s3, 0($s2)                  #$s3 = strlen(inputControlCode)
        addi $t4, $t4, -1               #$t4 = i
        for_loop_to_store_code:
                addi $t4, $t4, 1
                bne $t4, $s3, for_loop_to_store_code
                add $s1, $s1, $t4       #$s1 = inputControlCode + i
                add $s7,$s7,$t4                 #$s7 = latestCode + i
                sb  $s0, 0($s1)         #inputControlCode[i] = $s0
```

```
            sb  $s0, 0($s7)              #lastestCode[i]= $s0

            addi $s3, $s3, 1
            sw $s3, 0($s2)              #update length of input control code

#-----------------------------------------------------------
# Evaluate the return address of main routine
# epc <= epc + 4
#-----------------------------------------------------------
next_pc:
        mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
        addi $at, $at, 4 # $at = $at + 4 (next instruction)
        mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
#-----------------------------------------------------------
# RESTORE the REG FILE from STACK
#-----------------------------------------------------------
restore:
        lw $s3, 0($sp)
        addi $sp,$sp,-4
        lw $t4, 0($sp)
        addi $sp,$sp,-4
        lw $s2, 0($sp)
        addi $sp,$sp,-4
        lw $s1, 0($sp)
        addi $sp,$sp,-4
        lw $s0, 0($sp)
        addi $sp,$sp,-4
        lw $at, 0($sp)
        addi $sp,$sp,-4
        lw $a0, 0($sp)
        addi $sp,$sp,-4
        lw $t3, 0($sp)
        addi $sp,$sp,-4
        lw $t2, 0($sp)
        addi $sp,$sp,-4
        lw $t1, 0($sp)
        addi $sp,$sp,-4
        lw $ra, 0($sp)
        addi $sp,$sp,-4
return: eret # Return from exception
```

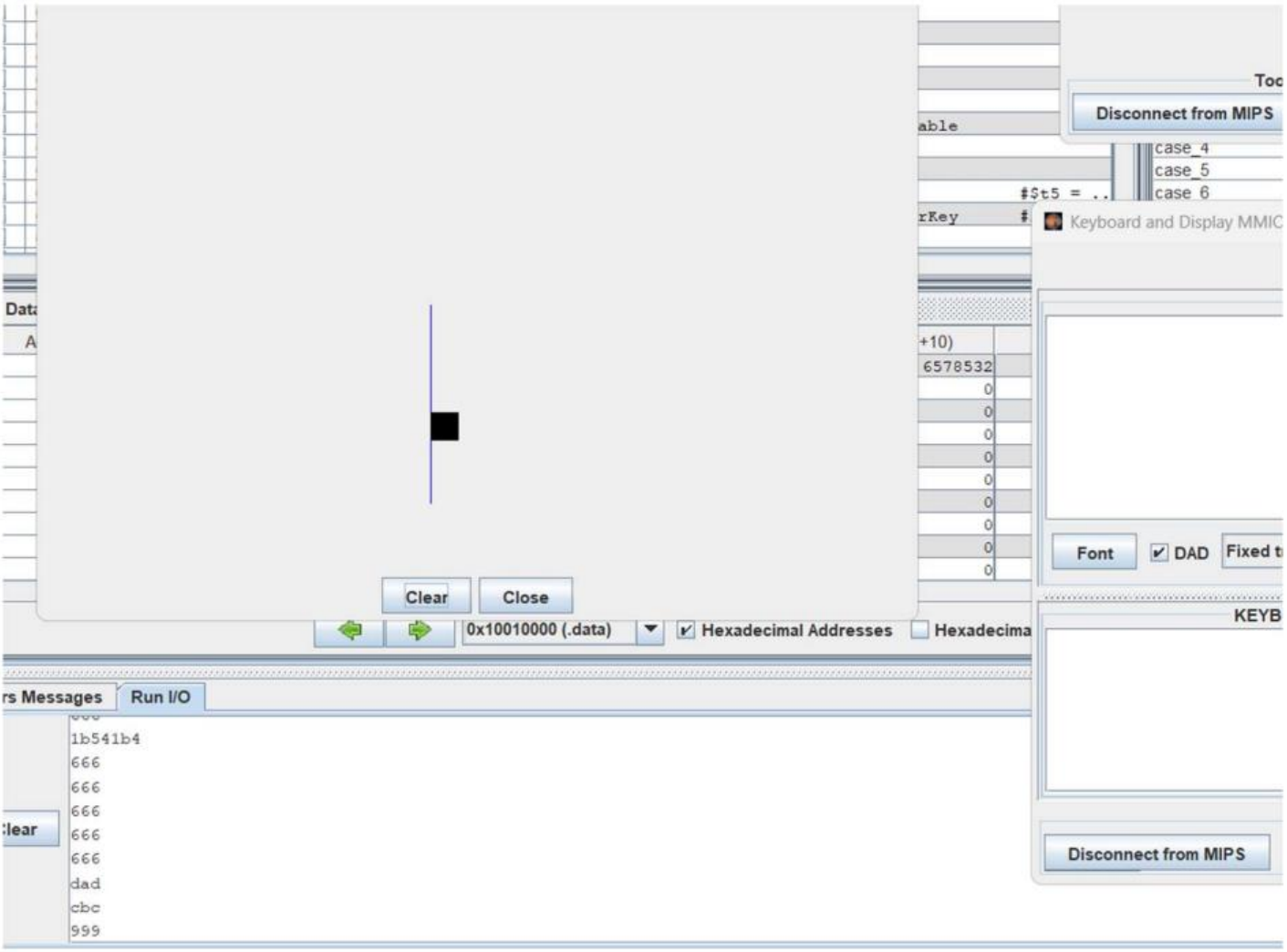
5. Mô phỏng:



Hình 1. Điều khiển Marsbot bằng các câu lệnh và in các câu lệnh ra console



Hình 2. In ra dialog thông báo control code nhập vào bị lỗi



Hình 3. Marsbot đi ngược lại trở về vị trí ban đầu

II. Project 7 *(Người thực hiện: Bùi Anh Đức – 20210195)*

1. Đề bài: Chương trình kiểm tra cú pháp lệnh MIPS

Trình biên dịch của bộxử lý MIPS sẽ tiến hành kiểm tra cú pháp các lệnh hợp ngữ trong mã nguồn, xem có phù hợp về cú pháp hay không, rồi mới tiến hành dịch các lệnh ra mã máy. Hãy viết một chương trình kiểm tra cú pháp của 1 lệnh hợp ngữ MIPS bất kì (không làm với giả lệnh) như sau:

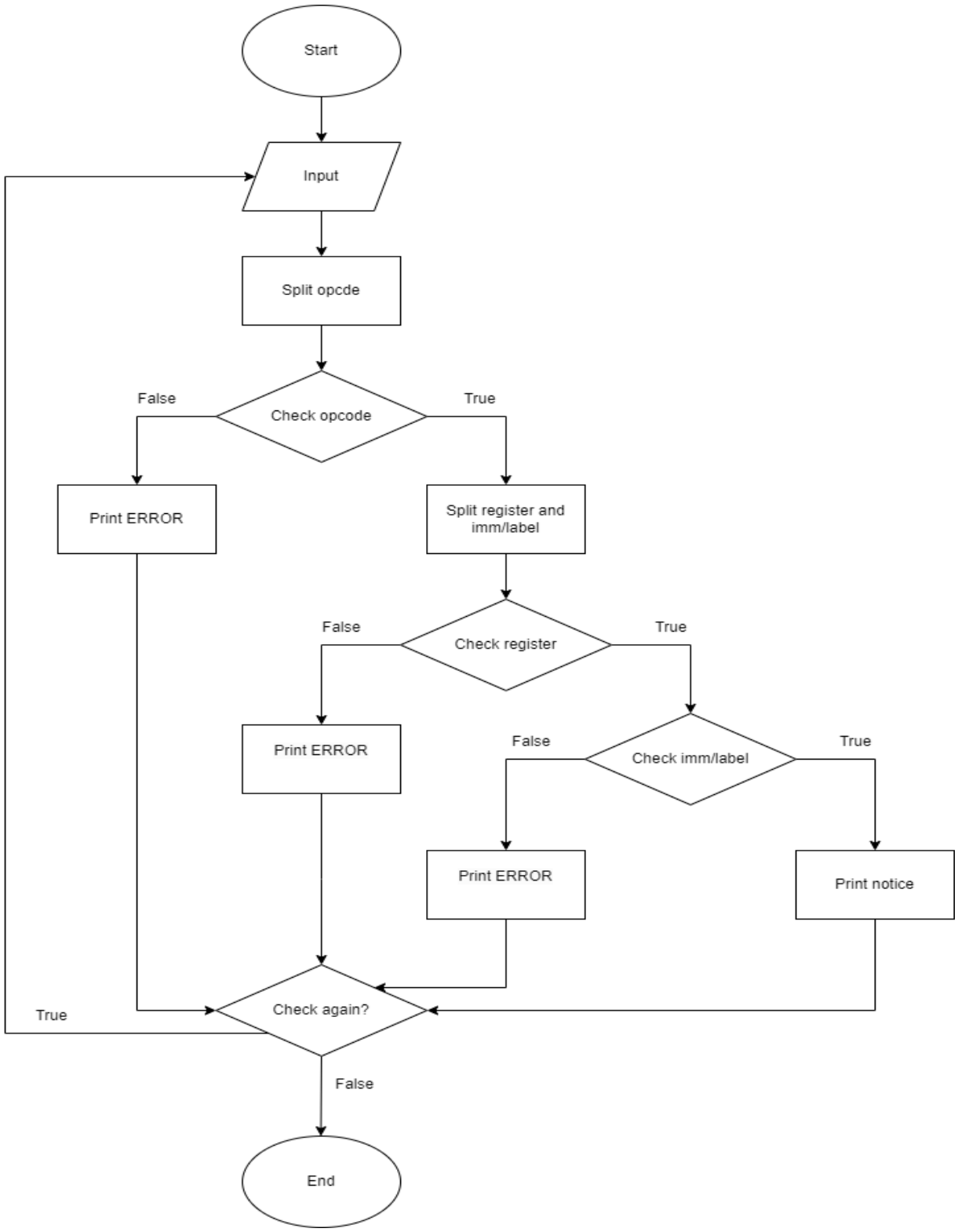
- Nhập vào từ bàn phím một dòng lệnh hợp ngữ. Ví dụ: *beq s1,31,t4*
- Kiểm tra xem mã opcode có đúng hay không? Trong ví dụ trên, opcode là beq là hợp lệ thì hiện thị thông báo *"opcode: beq, hợp lệ"*
- Kiểm tra xem tên các toán hạng phía sau có hợp lệ hay không? Trong ví dụ trên, *toán hạng s1 là hợp lệ, 31 là không hợp lệ, t4 thì khỏi phải kiểm tra nữa vì toán hạng trước đã bị sai rồi*.

2. Phân tích cách thực hiện:

a) Mô tả

- B1: Người dùng nhập một câu lệnh.
- B2: Tách câu lệnh ra để lấy opcode của câu lệnh này.
- B3: Kiểm tra opcode này có đúng không. Nếu đúng thì kiểm tra xem nó là loại opcode nào. Nếu sai thì in thông báo câu lệnh không chính xác ra màn hình.
- B4: Tách câu lệnh ra để lấy được thanh ghi thứ nhất và thanh ghi thứ 2/imm/label.
- B5: Kiểm tra thanh ghi đầu tiên có đúng không. Nếu đúng thì chuyển đến Bước 6, còn nếu sai thì in thông báo câu lệnh không chính xác ra màn hình.
- B6: Kiểm tra thanh ghi thứ 2/imm/label có đúng không. Nếu đúng thì in thông báo câu lệnh chính xác, còn nếu sai thì in thông báo câu lệnh không chính xác ra màn hình.
- B7: Kiểm tra xem người dùng có muốn tiếp tục kiểm tra hay không. Nếu có thì quay lại Bước 1, còn nếu không thì kết thúc chương trình.

b) Lưu đồ thuật toán



3. <u>Ý nghĩa các hàm trong mã nguồn:</u>

a) Split_opcode:

Ý nghĩa: chia một chuỗi (lệnh nhập vào) thành các chuỗi nhỏ (opcode, các thanh ghi, ...)

- Bước 1: khởi tạo các biến và thanh ghi

- Bước 2: Vòng lặp Loop1 lấy các ký tự từ chuỗi ban đầu (lệnh nhập vào) và kiểm tra xem nếu nó là NULL thì kết thúc vòng lặp, nếu nó là space thì bỏ qua và kiểm tra ký tự tiếp theo, còn nếu không thì tiếp tục chạy
- Bước 3: Vòng lặp Loop2 cũng tương tự như Loop 1 nhưng nó tiếp tục từ vị trí kết thúc Loop1 và kết thúc khi gặp newline.

b) Check_opcode:

Ý nghĩa: kiểm tra xem câu lệnh đầu vào có thuộc loại câu lệnh nào không (quy ước có các loại câu lệnh như: R, R1, R2, I, I1, J, J1, L, L1 và câu lệnh đặc biệt)

Lần lượt kiểm tra đầu vào có trùng với các opcode đúng không. Nếu có thì câu lệnh đầu vào thuộc loại đó, còn nếu không thì câu lệnh sai.

c) Check_Register_and_Number

Ý nghĩa: kiểm tra câu lệnh đầu vào có đúng cú pháp hay không

Nếu đã check_opcode và câu lệnh đầu vào thuộc loại nào, ta jump đến đoạn check của loại câu lệnh tương ứng, sau đó check các thanh ghi và imm/label sau đó. Nếu tất cả đều đúng thì câu lệnh đầu vào là đúng cú pháp, còn nếu không thì câu lệnh sai.

d) Kết thúc

Ý nghĩa: In thông báo ứng với các trường hợp và hỏi xem người dùng có muốn kiểm tra câu lệnh tiếp theo hay không. Nếu có thì reset các thanh ghi và quay lại đầu chương trình, nếu không thì kết thúc chương trình.

4. <u>Mã nguồn:</u> nằm trong file ***n07_g05_BuiAnhDuc.asm***

```
.data
Message1: .asciiz "Nhap dong lenh can check: "
Message2: .asciiz "Opcode: "
Message3: .asciiz ", hop le!"
Message4: .asciiz " khong hop le!"
Message5: .asciiz " \nCau lenh dung!\n--------------------\n"
Message6: .asciiz " \nCau lenh sai!\n--------------------\n"
Message7: .asciiz " \n"
Message8: .asciiz "Thanh ghi "
Message9: .asciiz "So "
Message10: .asciiz "Nhan "
Message11: .asciiz "Ban muon kiem tra tiep khong?"
```

```
Message12: .asciiz "\nLenh can kiem tra: "
string: .space 100
#Luu cac opcode can check vao mang
Opcode_R_Check: .asciiz "/add/sub/addu/subu/and/or/slt/sltu/nor/srav/srlv/movn/movz/mul/ "
Opcode_R_Check_1: .asciiz "/beq/bne/ "
Opcode_R_Check_2: .asciiz "/div/divu/mfc0/mult/multu/clo/clz/move/negu/not/madd/maddu/msub/msubu/ "
Opcode_I_Check: .asciiz "/addi/addiu/andi/ori/slti/sltiu/sll/srl/sra/ "
Opcode_I_Check_1: .asciiz "/li/lui/ "
Opcode_J_Check: .asciiz "/j/jal/ "
Opcode_J_Check_1: .asciiz "/jr/mfhi/mthi/mflo/mtlo/ "
Opcode_L_Check: .asciiz "/lb/lbu/lhu/ll/lw/sb/sc/sh/sw/lwc1/ldc1/swc1/sdc1/ "
Opcode_L_Check_1: .asciiz "/la/ "
Special_command: .asciiz "/syscall/nop/ "
Register_Check: .asciiz "/$zero/$at/$v0/$v1/$a0/$a1/$a2/$a3/$t0/$t1/$t2/$t3/$t4/$t5/$t6/$t7/$s0/$s1/$s2/$s3/$s4/$s5/$s6/$s7/$t8/$t9/$k0/$k1/$gp/$sp/$sp/$fp/$ra/$0/$1/$2/$3/$4/$5/$6/$7/$8/$9/$10/$11/$12/$13/$14/$15/$16/$17/$18/$19/$20/$21/$22/$23/$24/$25/$26/$27/$28/$29/$30/$31/ "
chain_check: .word   #Chua xau ki tu ðang xet
.text
start:
	la	$s2, chain_check #Dia chi chua chain_check
	li	$s6, 32		#s6=space
	li	$s7, 47		#s7 = '/'
#Nhap dong lenh can check
	li	$v0, 54
	la	$a0, Message1
	la	$a1, string
	la	$a2, 100
	syscall
	la	$s1, string
#-----------------------------------------------
#main
	jal	Print_Input
	jal	Split_opcode
	jal	Check_opcode
```

```
	beq	$s4, $zero, False_opcode #Opcode false
	addi $t0, $zero, 5			#Syscall, nop->Right code
	beq	$s4, $t0, Right_code
	addi $t5, $zero, 1
	beq	$s4, $t5, R_Check_Register_and_Number
	addi $t5, $zero, 2
	beq	$s4, $t5, R_1_Check_Register_and_Number
	addi $t5, $zero, 3
	beq	$s4, $t5, I_Check_Register_and_Number
	addi $t5, $zero, 4
	beq	$s4, $t5, J_Check_Register_and_Number
	addi $t5, $zero, 6
	beq	$s4, $t5, R_2_Check_Register_and_Number
	addi $t5, $zero, 7
	beq	$s4, $t5, I_1_Check_Register_and_Number
	addi $t5, $zero, 8
	beq	$s4, $t5, J_1_Check_Register_and_Number
	addi $t5, $zero, 9
	beq	$s4, $t5, L_Check_Register_and_Number
	addi $t5, $zero, 10
	beq	$s4, $t5, L_1_Check_Register_and_Number
	j	End_main
#-----------------------------------------------
#Tach ma opcode
Split_opcode:
	li	$s5, 0		#Vi tri load ban dau cua lenh nap vao
	li	$s0, 0		#Vi tri phan tu cuoi cua mang chain_check
	li	$t1, 0	#i=0
Loop1:
	add	$a2, $s1, $t1	#a2 = Dia chi cua ky tu dang load
	add	$a3, $s2, $s0	#a3 = Dia chi dang nap vao hang doi
	lb	$t0, 0($a2)
	beq	$t0, $zero, EndLoop	#Gap null => ket thuc vong lap 1
	beq	$t0, $s6, Loop1_them
	sb	$t0, 0($a3)		#Nap ky tu vao hang doi
	addi $s0, $s0, 1	#Dich chuyen vi tri cuoi cua hang doi sang phai
	addi $t1, $t1, 1
	addi $s5, $s5, 1
Loop2:
	add	$a2, $s1, $t1	#a2 = Dia chi cua ky tu dang load
```

```
        add  $a3, $s2, $s0    #a3 = Dia chi dang nap vao hang doi
        lb     $t0, 0($a2)
        beq  $t0, $zero, EndLoop  #Gap null => ket thuc vong lap 1
        beq  $t0, $s6, EndLoop    #Gap space => ket thuc vong lap 1
        li     $t5, 10                #t5=newline
        beq  $t0, $t5, EndLoop    #Gap newline => ket thuc vong lap 1
        sb     $t0, 0($a3)            #Nap ky tu vao hang doi
        addi $s0, $s0, 1       #Dich chuyen vi tri cuoi cua hang doi sang phai
        addi $t1, $t1, 1
        addi $s5, $s5, 1
        j       Loop2
EndLoop:
        #Chen ky tu NULL cho hang doi
        sb     $zero, 0($a3)
        #add        $s5, $s0, $zero         #Luu vi tri ki tu dang doc vao s5
        addi $s0, $s0, -1
        jr      $ra

#-------------------------------------------------
#Tach ma thanh ghi va so
Split_Register_and_Number:
        li      $s0, 0              #Vi tri phan tu cuoi cua mang chain_check
        add  $t1, $s5, $zero  #i=vi tri dang doc trong cau lenh=s5
Loop1_Split:
        add  $a2, $s1, $t1     #a2 = Dia chi cua ky tu dang load
        add  $a3, $s2, $s0    #a3 = Dia chi dang nap vao hang doi
        lb     $t0, 0($a2)      #t0 = Ky tu dang Load
        add  $t9, $zero, $t0  #t9 = Ky tu cuoi cung duoc load
        beq  $t0, $zero, EndLoop_Split#Check_Reg_and_Num   #Gap  null  =>
ket thuc vong lap 1
        beq  $t0, $s6, Loop1_Split_them      #Gap Space -> Chay qua Space
        li      $t5, 44             #t5=44~'dau phay,'
        beq  $t0, $t5, False_code
        sb     $t0, 0($a3)             #Nap ky tu vao hang doi
        addi $s0, $s0, 1       #Dich chuyen vi tri cuoi cua hang doi sang phai
        addi $t1, $t1, 1
Loop2_Split:
        add  $a2, $s1, $t1     #a2 = Dia chi cua ky tu dang load
        add  $a3, $s2, $s0    #a3 = Dia chi dang nap vao hang doi
        lb     $t0, 0($a2)
```

```
	add  $t9, $zero, $t0  #t9 = Ky tu cuoi cung duoc load
	beq  $t0, $zero, EndLoop_Split#Check_Reg_and_Num   #Gap  null  =>
ket thuc vong lap 1
	beq  $t0, $s6, Loop3_Split       #Gap space => Chay qua Space
	li   $t5, 10                     #t5=newline
	beq  $t0, $t5, EndLoop_Split   #Check_Reg_and_Num   #Gap  newline
=> ket thuc vong lap 1
	li   $t5, 44                     #t5=44~'dau phay,'
	beq  $t0, $t5, EndLoop_Split   #Gap dau phay => ket thuc vong lap 1
	sb   $t0, 0($a3)                 #Nap ky tu vao hang doi
	addi $s0, $s0, 1        #Dich chuyen vi tri cuoi cua hang doi sang phai
	addi $t1, $t1, 1
	j    Loop2_Split
Loop3_Split:
	add  $a2, $s1, $t1      #a2 = Dia chi cua ky tu dang load
	add  $a3, $s2, $s0      #a3 = Dia chi dang nap vao hang doi
	lb   $t0, 0($a2)        #t0 = Ky tu dang Load
	add  $t9, $zero, $t0  #t9 = Ky tu cuoi cung duoc load
	beq  $t0, $zero, EndLoop_Split#Check_Reg_and_Num   #Gap  null  =>
ket thuc vong lap 1
	beq  $t0, $s6, Loop3_Split_them     #Gap Space -> Chay qua Space
	li   $t5, 44             #t5=44~'dau phay,'
	beq  $t0, $t5, EndLoop_Split
	li   $t5, 10             #t5=10~'New line'
	beq  $t0, $t5, EndLoop_Split
	j    False_code
EndLoop_Split:
	#Chen ky tu NULL cho hang doi
	sb   $zero, 0($a3)
	addi $s5, $t1, 1 #Luu vi tri ki tu dang doc vao s5
	addi $s0, $s0, -1
	jr   $ra
#-------------------------------------------------
#Tach Sign ExtImm
Split_Sign_ExtImm:
	li   $s0, 0              #Vi tri phan tu cuoi cua mang chain_check
	add  $t1, $s5, $zero  #i=vi tri dang doc trong cau lenh=s5
Loop1_Sign:
	add  $a2, $s1, $t1      #a2 = Dia chi cua ky tu dang load
	add  $a3, $s2, $s0      #a3 = Dia chi dang nap vao hang doi
```

```
	lb	$t0, 0($a2)	#t0 = Ky tu dang Load
	add	$t9, $zero, $t0	#t9 = Ky tu cuoi cung duoc load
	beq	$t0,	$zero,	EndLoop_Sign_them_2#Check_Reg_and_Num
	#Gap null => ket thuc vong lap 1
	li	$t5, 10		#t5=10~'New line'
	beq	$t0, $t5, EndLoop_Sign_them_2
	beq	$t0, $s6, Loop1_Sign_them	#Gap Space -> Chay qua Space
	li	$t5, 44		#t5=44~'dau phay,'
	beq	$t0, $t5, False_code
	sb	$t0, 0($a3)		#Nap ky tu vao hang doi
	li	$t5, 40		#Thay dau ( thi ket thuc
	beq	$t0, $t5, EndLoop_Sign_them
	li	$t5, 41		#Thay dau ) thi ket thuc
	beq	$t0, $t5, EndLoop_Sign_them_3
	addi	$s0, $s0, 1	#Dich chuyen vi tri cuoi cua hang doi sang phai
	addi	$t1, $t1, 1
Loop2_Sign:
	add	$a2, $s1, $t1	#a2 = Dia chi cua ky tu dang load
	add	$a3, $s2, $s0	#a3 = Dia chi dang nap vao hang doi
	lb	$t0, 0($a2)
	add	$t9, $zero, $t0	#t9 = Ky tu cuoi cung duoc load
	beq	$t0,	$zero,	EndLoop_Sign_them_2#Check_Reg_and_Num
	#Gap null => ket thuc vong lap 1
	li	$t5, 10		#t5=10~'New line'
	beq	$t0, $t5, EndLoop_Sign_them_2
	beq	$t0, $s6, EndLoop_Sign	#Gap space => Chay qua Space
	li	$t5, 10		#t5=newline
	beq	$t0, $t5, EndLoop_Sign	#Check_Reg_and_Num	#Gap	newline
=> ket thuc vong lap 1
	li	$t5, 44		#t5=44~'dau phay,'
	beq	$t0, $t5, EndLoop_Sign	#Gap dau phay => ket thuc vong lap 1
	li	$t5, 40		#Thay dau ( thi ket thuc
	beq	$t0, $t5, EndLoop_Sign_them_1
	li	$t5, 41		#Thay dau ) thi ket thuc
	beq	$t0, $t5, EndLoop_Sign_them_1
	sb	$t0, 0($a3)		#Nap ky tu vao hang doi
	addi	$s0, $s0, 1	#Dich chuyen vi tri cuoi cua hang doi sang phai
	addi	$t1, $t1, 1
	j	Loop2_Sign
EndLoop_Sign:
```

```
        #Chen ky tu NULL cho hang doi
        sb      $zero, 0($a3)
        addi    $s5, $t1, 0
        addi    $s0, $s0, -1
        jr      $ra

#-------------------------------------------------
#Check Opcode
Check_opcode:
        li      $s4, 0      #s4 bieu thi cho khuon dang lenh: Saiopcode: 0, R: 1, R_1:
2, I: 3, J: 4, Dac biet: 5

        #Check_R
        la      $s3, Opcode_R_Check
        li      $t1, 0      #i=0
Loop1_R:
        add   $a3, $s3, $t1     #load byte cua opcode mau
        lb      $t3, 0($a3)
        addi  $t1, $t1, 1
        bne   $t3, $s7, Loop1_R
        li      $t0, 0              #So ki tu cua opcode mau
Loop2_R:
        add   $a3, $s3, $t1     #load byte cua opcode mau
        lb      $t3, 0($a3)
        add   $a2, $s2, $t0     #Load byte cua opcode can check
        lb      $t2, 0($a2)
        beq   $t3, $s7, Check_R
        beq   $t3, $s6, End_Loop_R
        bne   $t2, $t3, Loop1_R_them    #Kiem tra xem opcode check va opcode
mau co giong nhau khong
        beq   $t2, $t3, Loop2_R_them
End_Loop_R:

        #Check_R_2
        la      $s3, Opcode_R_Check_2
        li      $t1, 0      #i=0
Loop1_R_2:
        add   $a3, $s3, $t1     #load byte cua opcode mau
        lb      $t3, 0($a3)
        addi  $t1, $t1, 1
```

```
	bne	$t3, $s7, Loop1_R_2
	li	$t0, 0			#So ki tu cua opcode mau
Loop2_R_2:
	add	$a3, $s3, $t1	#load byte cua opcode mau
	lb	$t3, 0($a3)
	add	$a2, $s2, $t0	#Load byte cua opcode can check
	lb	$t2, 0($a2)
	beq	$t3, $s7, Check_R_2
	beq	$t3, $s6, End_Loop_R_2
	bne	$t2, $t3, Loop1_R_2_them	#Kiem tra xem opcode check va opcode mau co giong nhau khong
	beq	$t2, $t3, Loop2_R_2_them
End_Loop_R_2:



	#Check_I
	la	$s3, Opcode_I_Check
	li	$t1, 0		#i=0
Loop1_I:
	add	$a3, $s3, $t1	#load byte cua opcode mau
	lb	$t3, 0($a3)
	addi	$t1, $t1, 1
	bne	$t3, $s7, Loop1_I
	li	$t0, 0			#So ki tu cua opcode mau
Loop2_I:
	add	$a3, $s3, $t1	#load byte cua opcode mau
	lb	$t3, 0($a3)
	add	$a2, $s2, $t0	#Load byte cua opcode can check
	lb	$t2, 0($a2)
	beq	$t3, $s7, Check_I
	beq	$t3, $s6, End_Loop_I
	bne	$t2, $t3, Loop1_I_them	#Kiem tra xem opcode check va opcode mau co giong nhau khong
	beq	$t2, $t3, Loop2_I_them
End_Loop_I:

	#Check_I_1
	la	$s3, Opcode_I_Check_1
	li	$t1, 0		#i=0
Loop1_I_1:
```

```
        add  $a3, $s3, $t1     #load byte cua opcode mau
        lb     $t3, 0($a3)
        addi $t1, $t1, 1
        bne  $t3, $s7, Loop1_I_1
        li     $t0, 0              #So ki tu cua opcode mau
Loop2_I_1:
        add  $a3, $s3, $t1     #load byte cua opcode mau
        lb     $t3, 0($a3)
        add  $a2, $s2, $t0     #Load byte cua opcode can check
        lb     $t2, 0($a2)
        beq  $t3, $s7, Check_I_1
        beq  $t3, $s6, End_Loop_I_1
        bne  $t2, $t3, Loop1_I_1_them #Kiem tra xem opcode check va opcode
mau co giong nhau khong
        beq  $t2, $t3, Loop2_I_1_them
End_Loop_I_1:

        #Check_J
        la     $s3, Opcode_J_Check
        li     $t1, 0        #i=0
Loop1_J:
        add  $a3, $s3, $t1     #load byte cua opcode mau
        lb     $t3, 0($a3)
        addi $t1, $t1, 1
        bne  $t3, $s7, Loop1_J
        li     $t0, 0              #So ki tu cua opcode mau
Loop2_J:
        add  $a3, $s3, $t1     #load byte cua opcode mau
        lb     $t3, 0($a3)
        add  $a2, $s2, $t0     #Load byte cua opcode can check
        lb     $t2, 0($a2)
        beq  $t3, $s7, Check_J
        beq  $t3, $s6, End_Loop_J
        bne  $t2, $t3, Loop1_J_them    #Kiem tra xem opcode check va opcode
mau co giong nhau khong
        beq  $t2, $t3, Loop2_J_them
End_Loop_J:

        #Check_J_1
        la     $s3, Opcode_J_Check_1
```

```
        li      $t1, 0        #i=0
Loop1_J_1:
        add  $a3, $s3, $t1    #load byte cua opcode mau
        lb     $t3, 0($a3)
        addi $t1, $t1, 1
        bne   $t3, $s7, Loop1_J_1
        li      $t0, 0              #So ki tu cua opcode mau
Loop2_J_1:
        add  $a3, $s3, $t1    #load byte cua opcode mau
        lb     $t3, 0($a3)
        add  $a2, $s2, $t0    #Load byte cua opcode can check
        lb     $t2, 0($a2)
        beq  $t3, $s7, Check_J_1
        beq  $t3, $s6, End_Loop_J_1
        bne  $t2, $t3, Loop1_J_1_them       #Kiem tra xem opcode check va
opcode mau co giong nhau khong
        beq  $t2, $t3, Loop2_J_1_them
End_Loop_J_1:

        #Check Special Command
        la     $s3, Special_command
        li      $t1, 0        #i=0
Loop1_Sc:
        add  $a3, $s3, $t1    #load byte cua opcode mau
        lb     $t3, 0($a3)
        addi $t1, $t1, 1
        bne   $t3, $s7, Loop1_Sc
        li      $t0, 0              #So ki tu cua opcode mau
Loop2_Sc:
        add  $a3, $s3, $t1    #load byte cua opcode mau
        lb     $t3, 0($a3)
        add  $a2, $s2, $t0    #Load byte cua opcode can check
        lb     $t2, 0($a2)
        beq  $t3, $s7, Check_Sc
        beq  $t3, $s6, End_Loop_Sc
        bne  $t2, $t3, Loop1_Sc_them #Kiem tra xem opcode check va opcode
mau co giong nhau khong
        beq  $t2, $t3, Loop2_Sc_them
End_Loop_Sc:
```

```
        #Check_L
        la      $s3, Opcode_L_Check
        li      $t1, 0          #i=0
Loop1_L:
        add  $a3, $s3, $t1     #load byte cua opcode mau
        lb      $t3, 0($a3)
        addi $t1, $t1, 1
        bne   $t3, $s7, Loop1_L
        li      $t0, 0              #So ki tu cua opcode mau
Loop2_L:
        add  $a3, $s3, $t1     #load byte cua opcode mau
        lb      $t3, 0($a3)
        add  $a2, $s2, $t0     #Load byte cua opcode can check
        lb      $t2, 0($a2)
        beq  $t3, $s7, Check_L
        beq  $t3, $s6, End_Loop_L
        bne  $t2, $t3, Loop1_L_them    #Kiem tra xem opcode check va opcode
mau co giong nhau khong
        beq  $t2, $t3, Loop2_L_them
End_Loop_L:
        #Check_L_1
        la      $s3, Opcode_L_Check_1
        li      $t1, 0          #i=0
Loop1_L_1:
        add  $a3, $s3, $t1     #load byte cua opcode mau
        lb      $t3, 0($a3)
        addi $t1, $t1, 1
        bne   $t3, $s7, Loop1_L_1
        li      $t0, 0              #So ki tu cua opcode mau
Loop2_L_1:
        add  $a3, $s3, $t1     #load byte cua opcode mau
        lb      $t3, 0($a3)
        add  $a2, $s2, $t0     #Load byte cua opcode can check
        lb      $t2, 0($a2)
        beq  $t3, $s7, Check_L_1
        beq  $t3, $s6, End_Loop_L_1
        bne  $t2, $t3, Loop1_L_1_them       #Kiem tra xem opcode check va
opcode mau co giong nhau khong
        beq  $t2, $t3, Loop2_L_1_them
End_Loop_L_1:
```

```
#Check_R_1
	la	$s3, Opcode_R_Check_1
	li	$t1, 0		#i=0
Loop1_R_1:
	add	$a3, $s3, $t1	#load byte cua opcode mau
	lb	$t3, 0($a3)
	addi	$t1, $t1, 1
	bne	$t3, $s7, Loop1_R_1
	li	$t0, 0		#So ki tu cua opcode mau
Loop2_R_1:
	add	$a3, $s3, $t1	#load byte cua opcode mau
	lb	$t3, 0($a3)
	add	$a2, $s2, $t0	#Load byte cua opcode can check
	lb	$t2, 0($a2)
	beq	$t3, $s7, Check_R_1
	beq	$t3, $s6, End_Loop_R_1
	bne	$t2, $t3, Loop1_R_1_them	#Kiem tra xem opcode check va opcode mau co giong nhau khong
	beq	$t2, $t3, Loop2_R_1_them
End_Loop_R_1:
	jr	$ra
#-----------------------------------------------------
#Check cac thanh ghi va so
R_Check_Register_and_Number:
	jal	Right_opcode
	jal	Split_Register_and_Number
	jal	Check_Register
	#jal	Check_Number
	jal	Split_Register_and_Number
	jal	Check_Register
	jal	Split_Register_and_Number
	jal	Check_Register
	addi	$t5, $zero, 10
	beq	$t9, $t5, Right_code
	addi	$t5, $zero, 0
	beq	$t9, $t5, Right_code
	j	False_code
R_1_Check_Register_and_Number:
	jal	Right_opcode
```

```
        jal    Split_Register_and_Number
        jal    Check_Register
        jal    Split_Register_and_Number
        jal    Check_Register
        jal    Split_Register_and_Number
        addi $t5, $zero, 10
        beq  $t9, $t5, R_1_Check_Label
        addi $t5, $zero, 0
        beq  $t9, $t5, R_1_Check_Label
        j      False_code
        R_1_Check_Label:
        jal    Check_Label
R_2_Check_Register_and_Number:
        jal    Right_opcode
        jal    Split_Register_and_Number
        jal    Check_Register
        jal    Split_Register_and_Number
        jal    Check_Register
        addi $t5, $zero, 10
        beq  $t9, $t5, Right_code
        addi $t5, $zero, 0
        beq  $t9, $t5, Right_code
        j      False_code
I_Check_Register_and_Number:
        jal    Right_opcode
        jal    Split_Register_and_Number
        jal    Check_Register
        #jal  Check_Number
        jal    Split_Register_and_Number
        jal    Check_Register
        jal    Split_Register_and_Number
        jal    Check_Number
        addi $t5, $zero, 10
        beq  $t9, $t5, Right_code
        addi $t5, $zero, 0
        beq  $t9, $t5, Right_code
        j      False_code
I_1_Check_Register_and_Number:
        jal    Right_opcode
        jal    Split_Register_and_Number
```

```
        jal    Check_Register
        #jal   Check_Number
        jal    Split_Register_and_Number
        jal    Check_Number
        addi $t5, $zero, 10
        beq  $t9, $t5, Right_code
        addi $t5, $zero, 0
        beq  $t9, $t5, Right_code
        j      False_code
J_Check_Register_and_Number:
        jal    Right_opcode
        jal    Split_Register_and_Number
        addi $t5, $zero, 10
        beq  $t9, $t5, J_Check_Label
        addi $t5, $zero, 0
        beq  $t9, $t5, J_Check_Label
        j      False_code
        J_Check_Label:
        jal    Check_Label
J_1_Check_Register_and_Number:
        jal    Right_opcode
        jal    Split_Register_and_Number
        jal    Check_Register
        addi $t5, $zero, 10
        beq  $t9, $t5, Right_code
        addi $t5, $zero, 0
        beq  $t9, $t5, Right_code
        j      False_code
L_Check_Register_and_Number:
        jal    Right_opcode
        jal    Split_Register_and_Number
        jal    Check_Register
        jal    Check_Sign_ExtImm
L_1_Check_Register_and_Number:
        jal    Right_opcode
        jal    Split_Register_and_Number
        jal    Check_Register
        jal    Split_Register_and_Number
        addi $t5, $zero, 10
        beq  $t9, $t5, L_1_Check_Label
```

```
        addi $t5, $zero, 0
        beq  $t9, $t5, L_1_Check_Label
        j       False_code
        L_1_Check_Label:
        jal     Check_Label

#-------------------------------------------------
Loop1_them:
        addi $t1, $t1, 1
        addi $s5, $s5, 1
        j       Loop1
Loop1_Split_them:
        addi $t1, $t1, 1
        j       Loop1_Split
Loop2_Split_them:
        addi $t1, $t1, 1
        j       Loop2_Split
Loop3_Split_them:
        addi $t1, $t1, 1
        j       Loop3_Split
Loop1_Sign_them:
        addi $s5, $s5, 1
        addi $t1, $t1, 1
        j       Loop1_Sign
Loop2_Sign_them:
        addi $t1, $t1, 1
        j       Loop2_Sign
EndLoop_Sign_them:
        addi $a3, $a3, 1
        sb     $zero, 0($a3)
        addi $s5, $s5, 1
        jr      $ra
        #addi       $s0, $s0, -1
        #addi       $t1, $t1, 1
        #add        $a3, $s2, $s0          #Cap nhat moi dia chi dang load cua
hang doi
        #j     EndLoop_Sign
EndLoop_Sign_them_1:
        add  $a3, $s2, $s0          #Cap nhat moi dia chi dang load cua hang doi
        j       EndLoop_Sign
```

```
EndLoop_Sign_them_2:
        add  $s0, $s0, 1
        j       EndLoop_Sign
EndLoop_Sign_them_3:                    #load cac ki tu sau dau ) de kiem tra dung
sai
        addi $a2, $a2, 1
        lb      $t9, 0($a2)
        li      $t5, 0                  #t5 = NULL
        beq  $t9, $t5, Right_code
        li      $t5, 10                 #t5 = new line
        beq  $t9, $t5, Right_code
        li      $t5, 32                 #t5 = space
        beq  $t9, $t5, EndLoop_Sign_them_3
        j       False_code
Loop_Number_them:
        addi $t1, $t1, 1
        j       Loop_Number
Loop_Number_them_1:
        addi $t1, $t1, 1
        j       Loop_Number_1
Check_Mark_them:
        addi $t1, $t1, 1
        j       Check_Mark_done

#Check thanh ghi R
Check_R:
        addi $t0, $t0, -1
        beq  $s0, $t0, R_True
        j       Loop1_R
Loop1_R_them:
        addi $t1, $t1, 1
        j       Loop1_R
Loop2_R_them:
        addi $t1, $t1, 1
        addi $t0, $t0, 1
        j       Loop2_R
R_True:
        li      $s4, 1
        jr      $ra
```

```
#Check thanh ghi R_2
Check_R_2:
        addi $t0, $t0, -1
        beq  $s0, $t0, R_2_True
        j       Loop1_R_2
Loop1_R_2_them:
        addi $t1, $t1, 1
        j       Loop1_R_2
Loop2_R_2_them:
        addi $t1, $t1, 1
        addi $t0, $t0, 1
        j       Loop2_R_2
R_2_True:
        li      $s4, 6
        jr      $ra

#Check thanh ghi I
Check_I:
        addi $t0, $t0, -1
        beq  $s0, $t0, I_True
        j       Loop1_I
Loop1_I_them:
        addi $t1, $t1, 1
        j       Loop1_I
Loop2_I_them:
        addi $t1, $t1, 1
        addi $t0, $t0, 1
        j       Loop2_I
I_True:
        li      $s4, 3
        jr      $ra

#Check thanh ghi I_1
Check_I_1:
        addi $t0, $t0, -1
        beq  $s0, $t0, I_1_True
        j       Loop1_I_1
Loop1_I_1_them:
        addi $t1, $t1, 1
        j       Loop1_I_1
```

```
Loop2_I_1_them:
        addi $t1, $t1, 1
        addi $t0, $t0, 1
        j       Loop2_I_1
I_1_True:
        li      $s4, 7
        jr      $ra

#Check thanh ghi J
Check_J:
        addi $t0, $t0, -1
        beq  $s0, $t0, J_True
        j       Loop1_J
Loop1_J_them:
        addi $t1, $t1, 1
        j       Loop1_J
Loop2_J_them:
        addi $t1, $t1, 1
        addi $t0, $t0, 1
        j       Loop2_J
J_True:
        li      $s4, 4
        jr      $ra

#Check thanh ghi J_1
Check_J_1:
        addi $t0, $t0, -1
        beq  $s0, $t0, J_1_True
        j       Loop1_J_1
Loop1_J_1_them:
        addi $t1, $t1, 1
        j       Loop1_J_1
Loop2_J_1_them:
        addi $t1, $t1, 1
        addi $t0, $t0, 1
        j       Loop2_J_1
J_1_True:
        li      $s4, 8
        jr      $ra
```

```
#Check thanh ghi Sc - Special Command
Check_Sc:
        addi $t0, $t0, -1
        beq  $s0, $t0, Sc_True
        j       Loop1_Sc
Loop1_Sc_them:
        addi $t1, $t1, 1
        j       Loop1_Sc
Loop2_Sc_them:
        addi $t1, $t1, 1
        addi $t0, $t0, 1
        j       Loop2_Sc
Sc_True:
        li      $s4, 5
        jr      $ra

#Check thanh ghi R_1
Check_R_1:
        addi $t0, $t0, -1
        beq  $s0, $t0, R_1_True
        j       Loop1_R_1
Loop1_R_1_them:
        addi $t1, $t1, 1
        j       Loop1_R_1
Loop2_R_1_them:
        addi $t1, $t1, 1
        addi $t0, $t0, 1
        j       Loop2_R_1
R_1_True:
        li      $s4, 2
        jr      $ra
#Check thanh ghi L
Check_L:
        addi $t0, $t0, -1
        beq  $s0, $t0, L_True
        j       Loop1_L
Loop1_L_them:
        addi $t1, $t1, 1
        j       Loop1_L
Loop2_L_them:
```

```
        addi $t1, $t1, 1
        addi $t0, $t0, 1
        j       Loop2_L
L_True:
        li      $s4, 9
        jr      $ra

#Check thanh ghi L_1
Check_L_1:
        addi $t0, $t0, -1
        beq  $s0, $t0, L_1_True
        j       Loop1_L_1
Loop1_L_1_them:
        addi $t1, $t1, 1
        j       Loop1_L_1
Loop2_L_1_them:
        addi $t1, $t1, 1
        addi $t0, $t0, 1
        j       Loop2_L_1
L_1_True:
        li      $s4, 10
        jr      $ra

#-------------------------------------------------
#Check Register
Check_Register:
        la      $s3, Register_Check
        li      $t1, 0        #i=0
Loop1_Reg:
        add  $a3, $s3, $t1    #load byte cua thanh ghi mau
        lb      $t3, 0($a3)
        addi $t1, $t1, 1
        bne  $t3, $s7, Loop1_Reg
        li      $t0, 0              #So ki tu cua thanh ghi mau
Loop2_Reg:
        add  $a3, $s3, $t1    #load byte cua thanh ghi mau
        lb      $t3, 0($a3)
        add  $a2, $s2, $t0    #Load byte cua thanh ghi can check
        lb      $t2, 0($a2)
        beq  $t3, $s7, Check_Reg
```

```
	beq	$t3, $s6, False_code
	bne	$t2, $t3, Loop1_Reg_them	#Kiem tra xem thanh ghi check va thanh ghi mau co giong nhau khong
	beq	$t2, $t3, Loop2_Reg_them
End_Loop_Reg:

#-----------------------------------------------

Check_Reg:
	addi	$t0, $t0, -1
	beq	$s0, $t0, Reg_True
	j	Loop1_Reg
Loop1_Reg_them:
	addi	$t1, $t1, 1
	j	Loop1_Reg
Loop2_Reg_them:
	addi	$t1, $t1, 1
	addi	$t0, $t0, 1
	j	Loop2_Reg
Reg_True:
	add	$t8, $zero, $ra
	jal	Right_Register
	jr	$t8

#-----------------------------------------------
#Check Number
Check_Number:
	li	$t1, 0		#i = 0
	j	Check_Mark
Check_Mark_done:
	add	$a2, $s2, $t1	#Kiem tra so dau tien
	lb	$t2, 0($a2)
	li	$t5, 10		#t5 = newline
	beq	$t2, $t5, False_code
	beq	$t2, $zero, False_code
	li	$t5, 48		#t5 = zero
	bne	$t2, $t5, Loop_Number_1
	slti	$t4, $t2, 48
	bne	$t4, $zero, False_code
	slti	$t4, $t2, 58
```

```
        beq  $t4, $zero, False_code
        addi $t1, $t1, 1   #Kiem tra so thu hai(co the la x trong so hexa)
        add  $a2, $s2, $t1
        lb   $t2, 0($a2)
        beq  $t2, $zero, Right_Number
        li   $t5, 120
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 88
        beq  $t2, $t5, Loop_Number_them
        slti $t4, $t2, 48
        bne  $t4, $zero, False_code
        slti $t4, $t2, 58
        beq  $t4, $zero, False_code
Loop_Number:
        add  $a2, $s2, $t1
        lb   $t2, 0($a2)
        beq  $t2, $zero, Right_Number
        li   $t5, 48
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 49
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 50
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 51
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 52
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 53
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 54
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 55
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 56
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 57
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 65
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 66
```

```
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 67
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 68
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 69
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 70
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 97
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 98
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 99
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 100
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 101
        beq  $t2, $t5, Loop_Number_them
        li   $t5, 102
        beq  $t2, $t5, Loop_Number_them
        j    False_code
Loop_Number_1:
        add  $a2, $s2, $t1
        lb   $t2, 0($a2)
        beq  $t2, $zero, Right_Number
        li   $t5, 48
        beq  $t2, $t5, Loop_Number_them_1
        li   $t5, 49
        beq  $t2, $t5, Loop_Number_them_1
        li   $t5, 50
        beq  $t2, $t5, Loop_Number_them_1
        li   $t5, 51
        beq  $t2, $t5, Loop_Number_them_1
        li   $t5, 52
        beq  $t2, $t5, Loop_Number_them_1
        li   $t5, 53
        beq  $t2, $t5, Loop_Number_them_1
        li   $t5, 54
        beq  $t2, $t5, Loop_Number_them_1
```

```
        li      $t5, 55
        beq     $t2, $t5, Loop_Number_them_1
        li      $t5, 56
        beq     $t2, $t5, Loop_Number_them_1
        li      $t5, 57
        j       False_code
#-------------------------------------------------
Right_Number:
        add     $t8, $zero, $ra
        jal     Print_Right_Number
        jr      $t8
#-------------------------------------------------
Check_Mark:    #Ham kiem tra dau cua imm
        add     $a2, $s2, $t1     #Kiem tra xem ki tu dau tien cua Imm co phai dau +
hay - khong?
        lb      $t2, 0($a2)
        li      $t5, 43           #t5 =43 ~ '+'
        beq     $t2, $t5, Check_Mark_them
        li      $t5, 45           #t5 =45 ~ '-'
        beq     $t2, $t5, Check_Mark_them
        j       Check_Mark_done
#-------------------------------------------------
#Check Sign_ExtImm
Check_Sign_ExtImm:
        add     $t8, $zero, $ra         #Luu dia chi tro ve chuong trinh vao -> t8
        jal     Split_Sign_ExtImm
        jal     Check_Number
        jal     Split_Sign_ExtImm
        jal     Check_Parentheses_1
        jal     Split_Sign_ExtImm
        jal     Check_Register
        jal     Split_Sign_ExtImm
        jal     Check_Parentheses_2
        addi    $t5, $zero, 10
        beq     $t9, $t5, Right_code
        addi    $t5, $zero, 0
        beq     $t9, $t5, Right_code
        addi    $t5, $zero, 41          #t5 ~ ')'
        beq     $t9, $t5, Right_code
        j       False_code
```

```
#Check_Parentheses_1  Kiem tra dau (
Check_Parentheses_1:
        li      $t1, 0          #i = 0
        add   $a2, $s2, $t1
        lb      $t2, 0($a2)
        li      $t5, 40
        bne   $t2, $t5, False_code
        addi $t1, $t1, 1
        add   $a2, $s2, $t1
        lb      $t2, 0($a2)
        bne   $zero, $t2, False_code
        jr      $ra

#Check_Parentheses_2  Kiem tra dau )
Check_Parentheses_2:
        li      $t1, 0          #i = 0
        add   $a2, $s2, $t1
        lb      $t2, 0($a2)
        li      $t5, 41
        bne   $t2, $t5, False_code
        addi $t1, $t1, 1
        add   $a2, $s2, $t1
        lb      $t2, 0($a2)
        bne   $zero, $t2, False_code
        jr      $ra
#-----------------------------------------------
#Check Label
Check_Label:
        li      $t1, 0                  #i = 0
        add   $a2, $s2, $t1
        lb      $t2, 0($a2)       #lay ki tu tu hang doi
        beq   $t2, $zero, False_code
        li      $t5, 10                 #t5 = 'New line'
        beq   $t2, $t5, False_code
        slti    $t4, $t2, 65
        bne   $t4, $zero, False_code
        li      $t5, 91
        beq   $t2, $t5, False_code
        li      $t5, 92
```

```
        beq  $t2, $t5, False_code
        li   $t5, 93
        beq  $t2, $t5, False_code
        li   $t5, 94
        beq  $t2, $t5, False_code
        li   $t5, 96
        beq  $t2, $t5, False_code
        slti $t4, $t2, 123
        beq  $t4, $zero, False_code
        addi $t1, $t1, 1
Loop_Label:
        add  $a2, $s2, $t1
        lb   $t2, 0($a2)
        beq  $t2, $zero, True_Label
        li   $t5, 10            #t5 = 'New line'
        beq  $t2, $t5, True_Label
        slti $t4, $t2, 48
        bne  $t4, $zero, False_code
        li   $t5, 58
        beq  $t2, $t5, False_code
        li   $t5, 59
        beq  $t2, $t5, False_code
        li   $t5, 60
        beq  $t2, $t5, False_code
        li   $t5, 61
        beq  $t2, $t5, False_code
        li   $t5, 62
        beq  $t2, $t5, False_code
        li   $t5, 63
        beq  $t2, $t5, False_code
        li   $t5, 64
        beq  $t2, $t5, False_code
        li   $t5, 91
        beq  $t2, $t5, False_code
        li   $t5, 92
        beq  $t2, $t5, False_code
        li   $t5, 93
        beq  $t2, $t5, False_code
        li   $t5, 94
        beq  $t2, $t5, False_code
```

```
        li      $t5, 96
        beq     $t2, $t5, False_code
        slti    $t4, $t2, 123
        beq     $t4, $zero, False_code
        addi    $t1, $t1, 1
        j       Loop_Label

#-----------------------------------------------------
True_Label:
        jal     Print_Right_Label
        j       Right_code
#-----------------------------------------------------
#-----------------------------------------------------
#Output
Print_Input:
        #Print "Lenh can kiem tra"
        li      $v0, 4
        la      $a0, Message12
        syscall
        nop
        #Print lenh nguoi dung da nhap
        li      $v0, 4
        add     $a0, $zero,$s1
        syscall
        nop
        jr      $ra

False_opcode:
        #Print "Opcode"
        li $v0, 4
        la $a0, Message2
        syscall
        nop
        #Print Opcode Input
        li $v0, 4
        add $a0, $zero, $s2
        syscall
        nop
        #Print "Khong hop le!"
        li $v0, 4
```

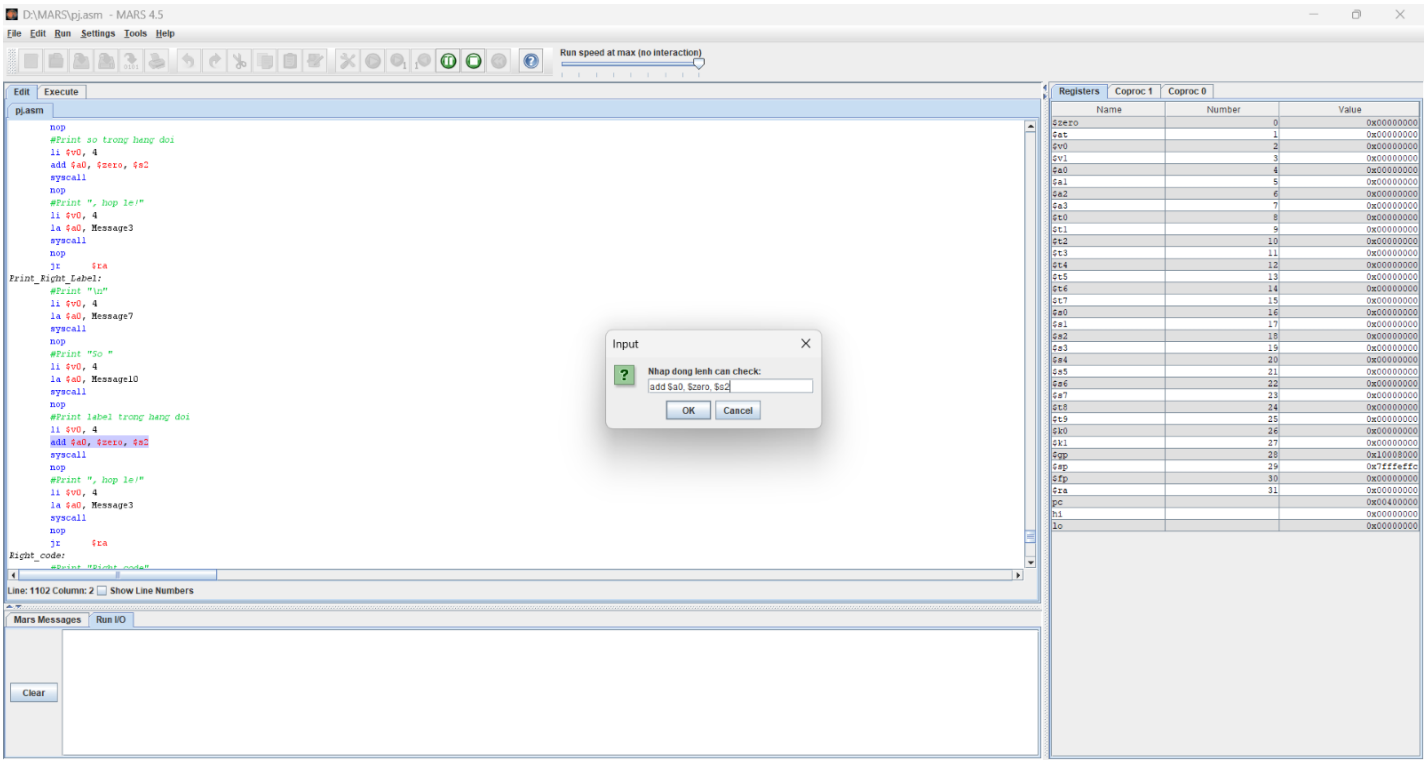
```
        la $a0, Message4
        syscall
        nop
        jal     False_code
        j       End_main
Right_opcode:
        #Print "Opcode"
        li $v0, 4
        la $a0, Message2
        syscall
        nop
        #Print Opcode Input
        li $v0, 4
        add $a0, $zero, $s2
        syscall
        nop
        #Print ", hop le!"
        li $v0, 4
        la $a0, Message3
        syscall
        nop
        jr      $ra
Right_Register:
        #Print "\n"
        li $v0, 4
        la $a0, Message7
        syscall
        nop
        #Print "Thanh ghi"
        li $v0, 4
        la $a0, Message8
        syscall
        nop
        #Print Register Input
        li $v0, 4
        add $a0, $zero, $s2
        syscall
        nop
        #Print ", hop le!"
        li $v0, 4
```

```
        la $a0, Message3
        syscall
        nop
        jr      $ra
Print_Right_Number:
        #Print "\n"
        li $v0, 4
        la $a0, Message7
        syscall
        nop
        #Print "So "
        li $v0, 4
        la $a0, Message9
        syscall
        nop
        #Print so trong hang doi
        li $v0, 4
        add $a0, $zero, $s2
        syscall
        nop
        #Print ", hop le!"
        li $v0, 4
        la $a0, Message3
        syscall
        nop
        jr      $ra
Print_Right_Label:
        #Print "\n"
        li $v0, 4
        la $a0, Message7
        syscall
        nop
        #Print "So "
        li $v0, 4
        la $a0, Message10
        syscall
        nop
        #Print label trong hang doi
        li $v0, 4
        add $a0, $zero, $s2
```

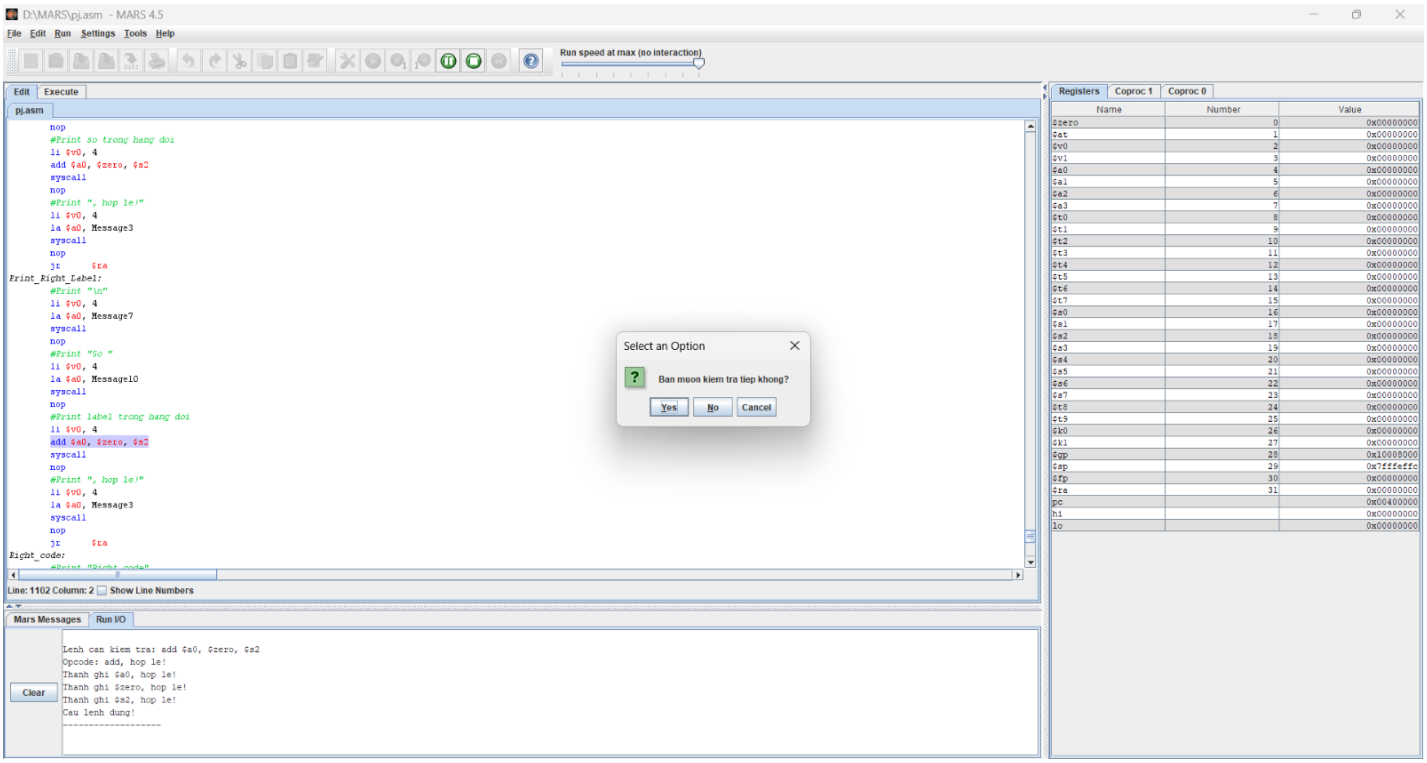
```
        syscall
        nop
        #Print ", hop le!"
        li $v0, 4
        la $a0, Message3
        syscall
        nop
        jr      $ra
Right_code:
        #Print "Right code"
        li $v0, 4
        la $a0, Message5
        syscall
        nop
        j       End_main
False_code:
        #Print "False code"
        li $v0, 4
        la $a0, Message6
        syscall
        nop
        j       End_main
End_main:
Run_Again:      li $v0, 50
                la $a0, Message11
                syscall
                nop
                beq $a0, $zero, clear
                nop
                j exit
                nop
# clear: dua string chua lenh ve trang thai ban dau de thuc hien lai qua trinh
clear:          add  $s3, $zero, $s1
Loop_Null:
                lb     $t3, 0($s3)
                li     $t5, 10
                beq  $t3, $t5, Loop_Null_them
                nop
                sb     $zero, 0($s3)
                addi $s3, $s3, 1
```

```
            j       Loop_Null
Loop_Null_them:
            sb      $zero, 0($s3)
            j start
            nop

exit:       li $v0, 10
            syscall
```

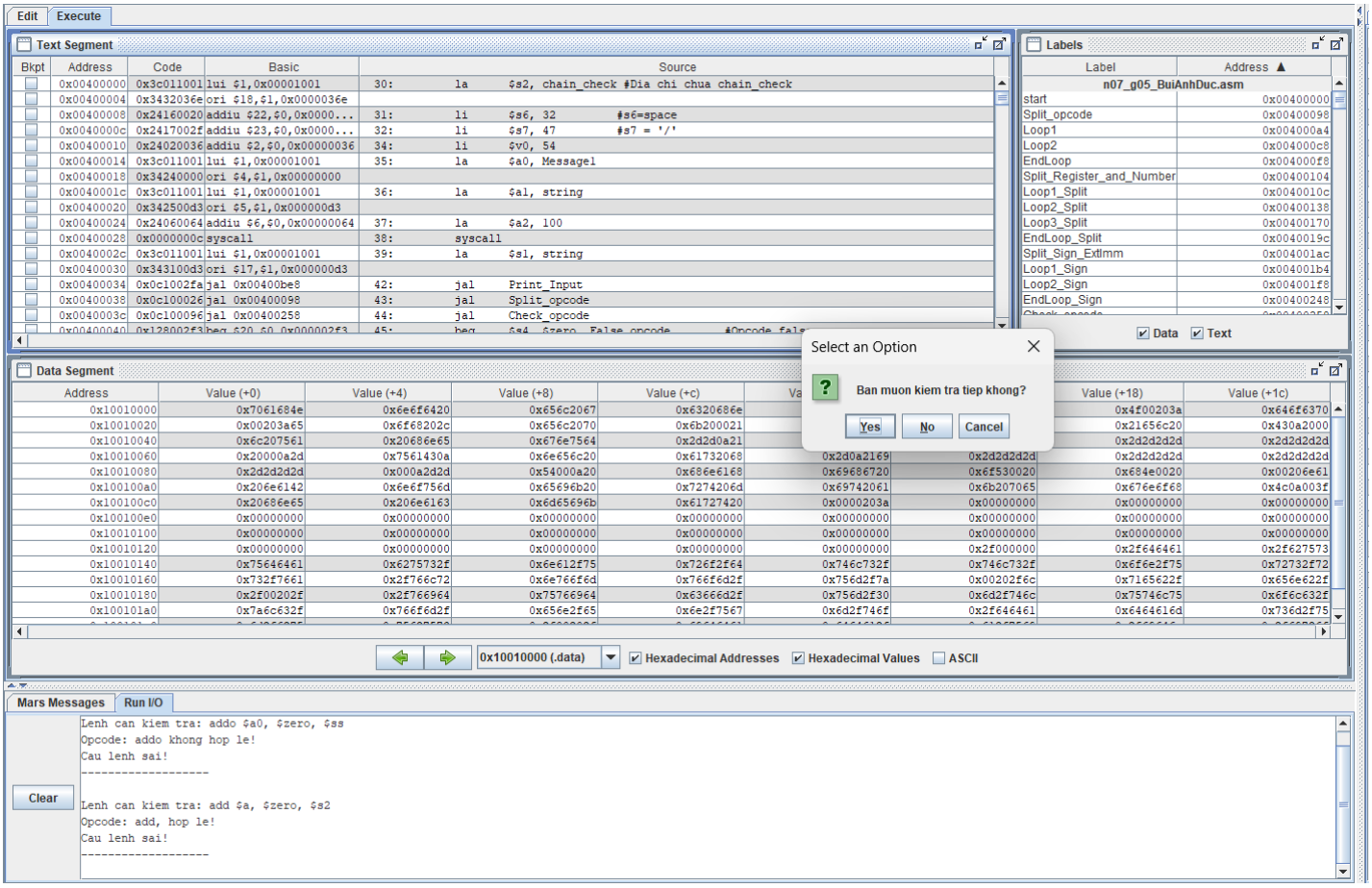
5. Mô phỏng:



Hình 1. Nhập câu lệnh



Hình 2. In kết quả và hỏi xem người dùng có kiểm tra tiếp không



Hình 3. Một vài trường hợp câu lệnh sai