ĐẠI HỌC BÁCH KHOA HÀ NỘI

BÁO CÁO BÀI TẬP LỚN

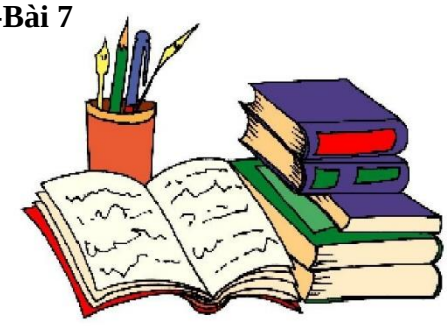
Học phần:THỰC HÀNH KIẾN TRÚC MÁY TÍNH
Mã lớp:1309378 Kỳ học:20212

Giảng viên hướng dẫn: Lê Bá Vui

Sinh viên thực hiện:

Nguyễn Hữu Đức-20204951-Bài 1

Lê Minh Vũ-20205050-Bài 7



Bài 1:
Đề bài:

1. Curiosity Marsbot

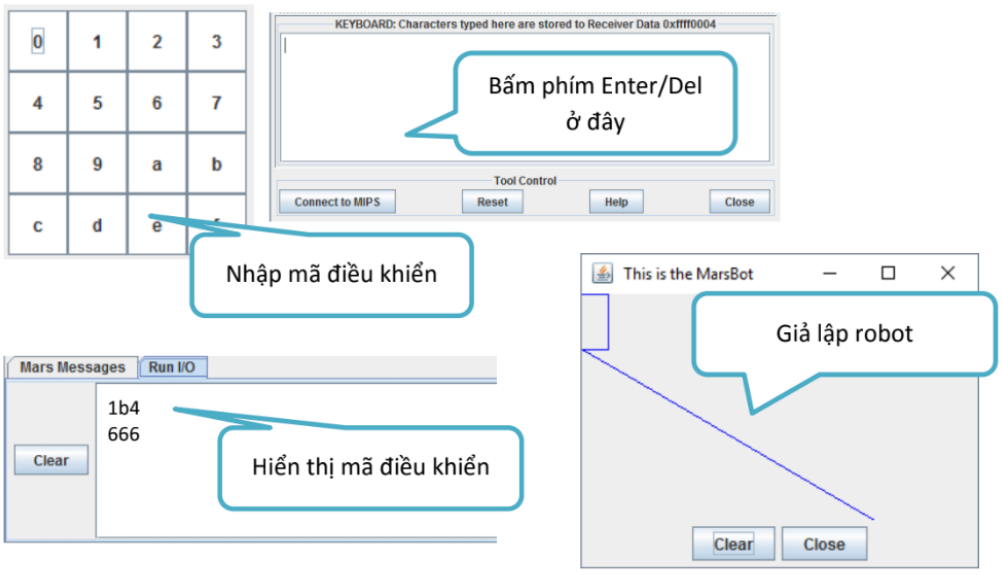
Xe tự hành Curioisity Marsbot chạy trên sao Hỏa, được vận hành từ xa bởi các lập trình viên trên Trái Đất. Bằng cách gửi đi các mã điều khiển từ một bàn phím ma trận, lập trình viên điều khiển quá trình di chuyển của Marbot như sau:

Mã điều khiển	Ý nghĩa
1b4	Marbot bắt đầu chuyển động
c68	Marbot đứng im
444	Rẽ trái 90* so vơi phương chuyển động gần đây và giữ hướng mới
666	Rẽ phải 90* so vơi 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 phát (dù có thể lệch một chút do hàm syscall sleep không thực sự thời gian thực)

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 Marbot thực thi
Phím Del	Xóa toàn bộ mã điều khiển đang nhập dở dang.

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.



***Ý tưởng:**

- Tạo chương trình nhận mã lệnh từ digital lab sim sau đó được kich hoạt từ Keyboard & Display MMIO Simulator. Đọc key nhận được từ Keyboard & Display MMIO Simulator và kiểm tra xem đó là phím spacebar, delete hay enter sau đó sẽ kích hoạt

tương ứng: thực hiện lệnh(Enter), xóa mã đang nhập(Delete), lặp lại lệnh trước đó(Spacebar). Sau đó so sánh mã lệnh nhận được với mã lệnh format đã cho có đúng format hay không rồi cho marsbot thực thi.

***Chi tiết thuật toán:**

1. **Khởi tạo data:**

```
.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

.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
#0-3
.eqv KEY_0 0x11
.eqv KEY_1 0x21
.eqv KEY_2 0x41
.eqv KEY_3 0x81
#4-7
.eqv KEY_4 0x12
.eqv KEY_5 0x22
.eqv KEY_6 0x42
.eqv KEY_7 0x82
#8-b
.eqv KEY_8 0x14
.eqv KEY_9 0x24
.eqv KEY_a 0x44
.eqv KEY_b 0x84
#c-f
.eqv KEY_c 0x18
.eqv KEY_d 0x28
.eqv KEY_e 0x48
.eqv KEY_f 0x88

#Function code
ChuyenDong: .asciiz "1b4"           #Marsbot bat dau chuyen dong
Dung: .asciiz "c68"           #Marsbot dung im
ReTrai: .asciiz "444"         #Re trai
RePhai: .asciiz "666"         #Re phai
DeVet: .asciiz "dad"          #De lai vet tren duong
DungDeVet: .asciiz "cbc"            #Cham dut de lai vet tren duong
DiNguoc: .asciiz "999"        #Di lo trinh nguoc lai
MaLoi: .asciiz "Ma khong hop le!"

InputCode: .space 50
```

```
InputCode1: .space 50
CodeLong: .word 0
CodeLong1: .word 0
HuongDi: .word 0

# duong di cua masbot 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
PathLong: .word 12            #bytes
```

- Khởi tạo các mảng cần thiết.
- Gán giá trị cho các biến phím nhận từ digital lab sim để tạo lệnh ngắt.
- Gán các mã lệnh vào các biến để thực hiện so sánh.

2. Các chương trình con

- Chương trình đợi nhập một phím bất kỳ từ bàn phím vào keyboard & display MMIO simulator.

Các chương trình chức năng:
-CheckInput: Thực hiện check mã lệnh nhập vào có đúng với format được cho hay không. Nếu đúng sẽ đi đến bước thực thi còn nếu sai sẽ chay tới Error để báo lỗi.
-Print: In lệnh vừa thực thi(InputCode) ra và sao lưu vào một mảng phụ(InputCode1) để thực hiện lặp.
-Delete: Thực hiện xóa mã lệnh đang nhập chứa trong InputCode.
-Repeat: Thực hiện sao lưu lại lệnh vừa thực hiện trước đó đang chứa trong InputCode1 vào InputCode để chạy lại một lần nữa.
-StorePath: Lưu lại vị trí hiện tại và hướng đi của marsbot ngay khi được gọi tới.
-Equal: thực hiện check xem mã lệnh đã nhập có đúng với format đã cho hay không(chương trình con của CheckInput). Nếu đúng thì sẽ trả về 1 và nếu sai sẽ trả về 0.
-Error: Báo lỗi khi được gọi tới.

Các chương trình chức năng theo mã điều khiển:
-CodeReturn: Thực hiện đảo ngược lại quãng đường đã đi.
-CodeTrack: Nhảy đến track để thực hiểnhiện để lại vết.
-CodeUntrack: Dừng để lại vết.
-CodeGo: Bắt đầu chuyển động.
-CodeStop: Dừng chuyển động.
-CodeRight: tăng hướng đi thêm 90 độ để hướng của marsbot quay sang phải.
-CodeLeft: Tương tự quay trái 90 độ.

Các code chức năng của marsbot: GO,STOP,TRACK,UNTRACK,ROTATE.

Code nhận input từ digital lab sim

Ở mỗi chương trình con đều thêm backup và sau khi thực hiện chương trình con xong sẽ restore vì tránh để các thanh ghi bị ghi đè và thay đổi dữ liệu làm lỗi lệnh vì việc phải sử dụng nhiều thanh ghi có thể gây nhầm lẫn.

3. Mô tả quá trình

B1: Nhập mã lệnh từ Digital Lab Sim
B2: Chờ nhận tín hiệu phím từ bàn phím vào Keyboard & Display MMIO Simulator
+ Nếu nhấn phím Enter sẽ nhảy đến bước thực thi lệnh
+ Nếu nhấn phím Delete sẽ xóa hết mã lệnh đang nhập.
+ Nếu nhấn phím Spacebar sẽ thực hiện lại lệnh trước đó(trong trường hợp mới mở tools lên chưa có lệnh gì khi ấn spacebar sẽ nhận là lệnh rỗng => báo lỗi).
B3: Nếu ấn Enter và đến bước thực thi lệnh, Marsbot sẽ thực hiện theo yêu cầu sau đó sẽ in ra mã lệnh đã nhập xuống màn hình hiển thị.

Nếu thực hiện báo lỗi xong sẽ thực hiện xóa hết mã lệnh đang chứa trong CodeInput để nhận lệnh tiếp theo và sau khi thực thi lệnh xong và in lệnh ra sẽ thực hiện xóa hết lệnh đang chứa trong CodeInput để chờ lệnh tiếp theo.

4. Mã nguồn

```
.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

.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
#0-3
.eqv KEY_0 0x11
.eqv KEY_1 0x21
.eqv KEY_2 0x41
.eqv KEY_3 0x81
#4-7
.eqv KEY_4 0x12
.eqv KEY_5 0x22
.eqv KEY_6 0x42
.eqv KEY_7 0x82
#8-b
.eqv KEY_8 0x14
.eqv KEY_9 0x24
.eqv KEY_a 0x44
.eqv KEY_b 0x84
```

```
#c-f
.eqv KEY_c 0x18
.eqv KEY_d 0x28
.eqv KEY_e 0x48
.eqv KEY_f 0x88

#Function code
ChuyenDong: .asciiz "1b4"        #Marsbot bat dau chuyen dong
Dung: .asciiz "c68"        #Marsbot dung im
ReTrai: .asciiz "444"      #Re trai
RePhai: .asciiz "666"      #Re phai
DeVet: .asciiz "dad"       #De lai vet tren duong
DungDeVet: .asciiz "cbc"          #Cham dut de lai vet tren duong
DiNguoc: .asciiz "999"            #Di lo trinh nguoc lai
MaLoi: .asciiz "Ma khong hop le!"

InputCode: .space 50
InputCode1: .space 50
CodeLong: .word 0
CodeLong1: .word 0
HuongDi: .word 0

# duong di cua masbot 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
PathLong: .word 12         #bytes



.text
main:
li $k0, KEY_CODE
li $k1, KEY_READY
#-----------------------------------------------------------
# Enable the interrupt 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 , Delete      #if $t6 == delete key then remove input
#127 is delete key in ascii
beq $t6, 32, repeat

beq $t6, 13 , loop         #if $t6 != '\n' then Polling
nop
beq $t6, 13 , loop

#Cac code chuc nang
#-----------------------------------------------------------------------------
```

```
------------------------
   #Chuong trinh con thuc hien ma code vua nhan tu digital lab sim
   CheckInput:
   la $s2, CodeLong
   lw $s2, 0($s2)
   bne $s2, 3, Error

   la $s3, ChuyenDong
   jal Equal
   beq $t0, 1, CodeGO

   la $s3, Dung
   jal Equal
   beq $t0, 1, CodeStop

   la $s3, ReTrai
   jal Equal
   beq $t0, 1, CodeLeft

   la $s3, RePhai
   jal Equal
   beq $t0, 1, CodeRight

   la $s3, DeVet
   jal Equal
   beq $t0, 1, CodeTrack

   la $s3, DungDeVet
   jal Equal
   beq $t0, 1, CodeUntrack

   la $s3, DiNguoc
   jal Equal
   beq $t0, 1, CodeReturn

   beq $t0, 0, Error


   #Thuc hien in ra code vua duoc nhan va thuc thi
   #sau do sao luu ma code do sang mot bien khac de dung cho code repeat
   Print:
   li $v0, 4
   la $a0, InputCode
   syscall
   nop
   la $s1, InputCode
   la $s2, InputCode1
   strcpy1:
   add $s0, $zero, $zero # s0 = i = 0
   L1:
   add $t1,$s0, $s1 # t1 = s0 + a1 = dia chi y[i]
   lb $t2, 0($t1) # t2 = gia tri tai dia chi t1 = gia tri cua y[i]
   add $t3, $s2, $s0 # t3 = dia chi bat dau cua xau dich + index = dia chi cua
x[i]
   sb $t2, 0($t3) # Gan gia tri cua y[i] cho thanh ghi co dia chi t3 (x[i])
   beq $t2, $zero, end_of_strcpy1 #Neu ki tu vua doc duoc la KI TU KET THUC
CHUOI, KET THUC
   nop
   addi $s0, $s0, 1 # s0 = s0+1 = i+1
   j L1
   nop
```

```
   end_of_strcpy1:
   la $s1, CodeLong
   la $s2, CodeLong1
   strcpy3:
   add $s0, $zero, $zero # s0 = i = 0
   L3:
   add $t1,$s0, $s1 # t1 = s0 + a1 = dia chi y[i]
   lb $t2, 0($t1) # t2 = gia tri tai dia chi t1 = gia tri cua y[i]
   add $t3, $s2, $s0 # t3 = dia chi bat dau cua xau dich + index = dia chi cua
x[i]
   sb $t2, 0($t3) # Gan gia tri cua y[i] cho thanh ghi co dia chi t3 (x[i])
   beq $t2, $zero, end_of_strcpy3 #Neu ki tu vua doc duoc la KI TU KET THUC
CHUOI, KET THUC
   nop
   addi $s0, $s0, 1 # s0 = s0+1 = i+1
   j L3
   nop
   end_of_strcpy3:


   #Delete code cho nhan phim Delete
   Delete:
   #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, CodeLong
   lw $t3, 0($s2)                          #$t3 = CodeLong
   addi $t1, $zero, -1                     #$t1 = -1 = i
   addi $t2, $zero, 0                      #$t2 = '\0'
   la $s1, InputCode
   addi $s1, $s1, -1
   Delete_loop: addi $t1, $t1, 1                #i++
   add $s1, $s1, 1                  #$s1 = InputCode + i
   sb $t2, 0($s1)                   #InputCode[i] = '\0'

   bne $t1, $t3, Delete_loop  #if $t1 <=3 Delete loop
   nop
   bne $t1, $t3, Delete_loop

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

   j loop
   nop
   j loop

   #Repeat code cho nhan phim space
   repeat:
   #backup
   addi $sp,$sp,4
   sw $s1, 0($sp)
   addi $sp,$sp,4
   sw $s2, 0($sp)
   addi $sp,$sp,4
   sw $s0, 0($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)

   #processing
   la $s1, InputCode1
   la $s2, InputCode
   strcpy2:
   add $s0, $zero, $zero # s0 = i = 0
   L2:
   add $t1,$s0, $s1 # t1 = s0 + a1 = dia chi y[i]
   lb $t2, 0($t1) # t2 = gia tri tai dia chi t1 = gia tri cua y[i]
   add $t3, $s2, $s0 # t3 = dia chi bat dau cua xau dich + index = dia chi cua
x[i]
   sb $t2, 0($t3) # Gan gia tri cua y[i] cho thanh ghi co dia chi t3 (x[i])
   beq $t2, $zero, end_of_strcpy2 #Neu ki tu vua doc duoc la KI TU KET THUC
CHUOI, KET THUC
   nop
   addi $s0, $s0, 1 # s0 = s0+1 = i+1
   j L2
   nop
   end_of_strcpy2:
   la $s1, CodeLong1
   la $s2, CodeLong
   strcpy4:
   add $s0, $zero, $zero # s0 = i = 0
   L4:
   add $t1,$s0, $s1 # t1 = s0 + a1 = dia chi y[i]
   lb $t2, 0($t1) # t2 = gia tri tai dia chi t1 = gia tri cua y[i]
   add $t3, $s2, $s0 # t3 = dia chi bat dau cua xau dich + index = dia chi cua
x[i]
   sb $t2, 0($t3) # Gan gia tri cua y[i] cho thanh ghi co dia chi t3 (x[i])
   beq $t2, $zero, end_of_strcpy4 #Neu ki tu vua doc duoc la KI TU KET THUC
CHUOI, KET THUC
   nop
   addi $s0, $s0, 1 # s0 = s0+1 = i+1
   j L4
   nop
   end_of_strcpy4:
   #restore
   lw $s1, 0($sp)
   addi $sp,$sp,-4
```

```
lw $s2, 0($sp)
addi $sp,$sp,-4
lw $s0, 0($sp)
addi $sp,$sp,-4
lw $t1, 0($sp)
addi $sp,$sp,-4
lw $t2, 0($sp)
addi $sp,$sp,-4
lw $t3, 0($sp)
addi $sp,$sp,-4

j CheckInput
nop
j CheckInput

#luu lai vi tri hien tai va huong di cua marsbot
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, HuongDi
lw $s4, 0($s4)        #s4 = now heading

la $t3, PathLong
lw $s3, 0($t3)        #$s3 = PathLong (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 PathLong
#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


   #Code check xem ma code nhan duoc va ma code theo dau bai cho co dung
format hay khong
   #Luc nay $s3 se chua dia chi co cac code chuc nang theo format da cho
   #$t0 la output neu code nhan vao dung format se tra ve 1, nguoc lai la 0

   Equal:
   #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 $s1, InputCode                 #$s1 = InputCode
   Equal_loop: addi $t1, $t1, 1                  #i++

   add $t2, $s1, $t1                 #$t2 = InputCode + i
   lb $t2, 0($t2)                    #$t2 = InputCode[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, Equal_loop     #if $t1 <=2 Delete loop
   nop
   bne $t1, 2, Equal_loop
   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


   #Code bao loi
   Error: li $v0, 4
   la $a0, InputCode
   syscall
   nop

   li $v0, 55
   la $a0, MaLoi
   syscall
   nop
   nop
   j Delete
   nop
   j Delete
   #-------------------------------------------------------------------------
-------



   #Code thuc hien cac chuc nang theo yeu cau
   #-------------------------------------------------------------------------
-----------
   #Di theo lo trinh nguoc lai
   CodeReturn:    la $s7, Path
   la $s5, PathLong
   lw $s5, 0($s5)
   add $s7, $s7, $s5
   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
   #sub $s6, $zero, $s6

   la $t8, HuongDi#marsbot quay nguoc lai
   sw $s6, 0($t8)
```

```
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

beq $s5, 0, finish

j begin

finish:  jal STOP
la $t8, HuongDi
add $s6, $zero, $zero
sw $s6, 0($t8)        #update heading
la $t8, PathLong
sw $s5, 0($t8)        #update PathLong = 0
jal ROTATE
j Print

#De lai vet tren duong
CodeTrack:     jal TRACK
j Print

#Dung de lai vet tren duong
CodeUntrack:   jal UNTRACK
j Print

#Bat dau chuyen dong
CodeGO:  jal GO
j Print

#Marsbot dung im
CodeStop:      jal STOP
j Print

#Re phai 90 do
CodeRight:     la $s5, HuongDi
lw $s6, 0($s5) #$s6 is heading at now
addi $s6, $s6, 90    #increase heading by 90*
sw $s6, 0($s5) # update HuongDi
jal storePath
jal ROTATE
j Print

#Re trai 90 do
CodeLeft:la $s5, HuongDi
lw $s6, 0($s5) #$s6 is heading at now
addi $s6, $s6, -90   #increase heading by 90*
sw $s6, 0($s5) # update HuongDi
jal storePath
jal ROTATE
j Print
```

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


   #Cac chuc nang cua Marsbot
   #-------------------------------------------------------------------------
---

   # 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
```

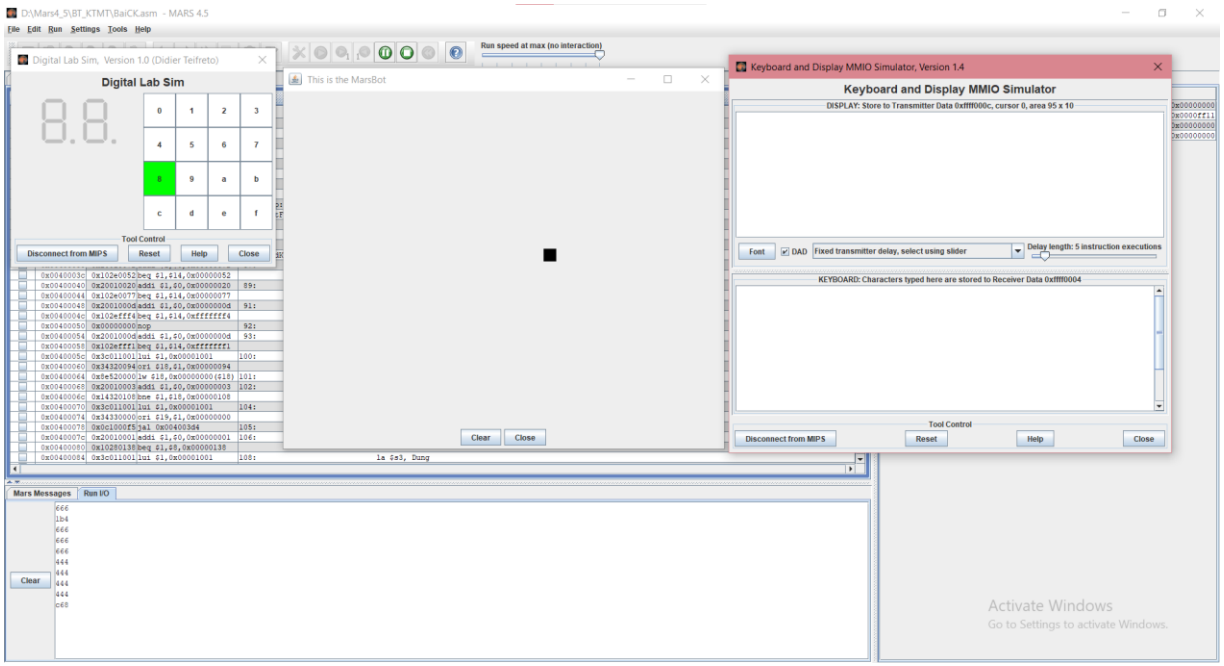
```
   lw $at, 0($sp)
   addi $sp,$sp,-4

   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] HuongDi 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, HuongDi
   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
   #-------------------------------------------------------------------


   #Nhan code input
   #-----------------------------------------------------------
   #===========================================================================
=====
   # GENERAL INTERRUPT SERVED ROUTINE for all interrupts
   #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   .ktext 0x80000180
```

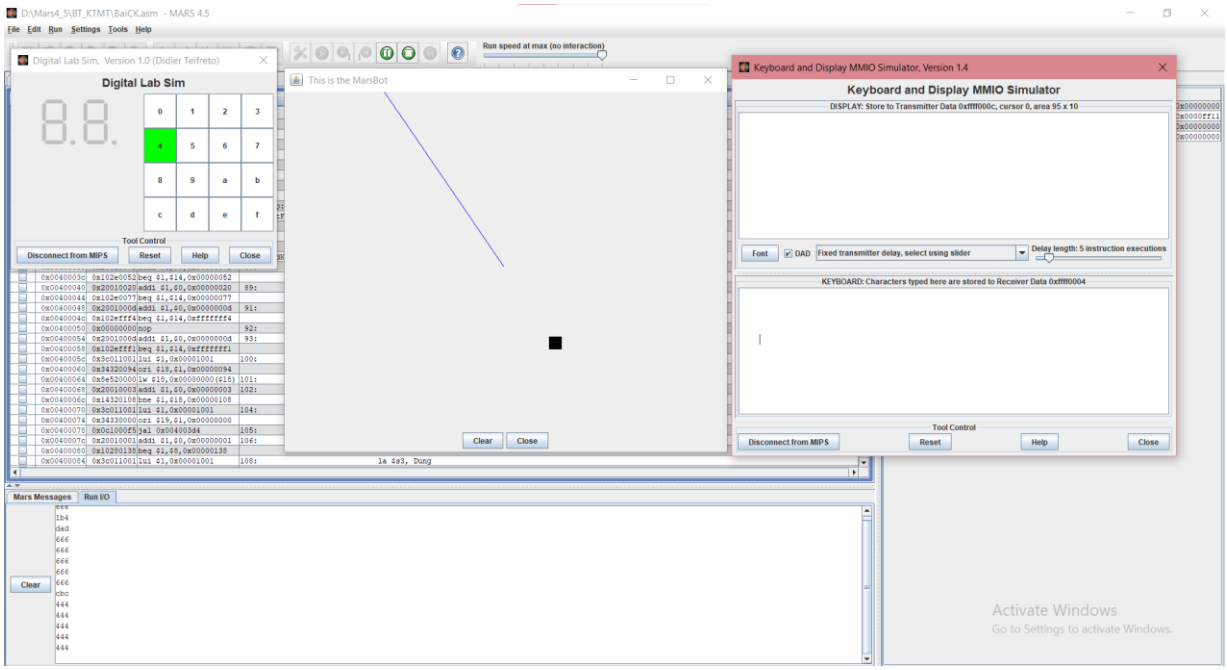
```
#-------------------------------------------------------
# SAVE the current REG FILE to stack
#-------------------------------------------------------
backup:
addi $sp,$sp,4
sw $ra,0($sp)
addi $sp,$sp,4
sw $t1,0($sp)
addi $sp,$sp,4
sw $t2,0($sp)
addi $sp,$sp,4
sw $t3,0($sp)
addi $sp,$sp,4
sw $a0,0($sp)
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 $s4,0($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, 0x11
sb $t3, 0($t1)
lbu $a0, 0($t2)
bnez $a0, get_code_in_char
scan_row2:
li $t3, 0x12
sb $t3, 0($t1)
lbu $a0, 0($t2)
bnez $a0, get_code_in_char
scan_row3:
li $t3, 0x14
sb $t3, 0($t1)
lbu $a0, 0($t2)
bnez $a0, get_code_in_char
scan_row4:
li $t3, 0x18
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, InputCode
la $s2, CodeLong
lw $s3, 0($s2)                  #$s3 = strlen(InputCode)
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 = InputCode + i
sb  $s0, 0($s1)            #InputCode[i] = $s0

addi $s0, $zero, '\n'      #add '\n' character to end of string
addi $s1, $s1, 1           #add '\n' character to end of string
sb  $s0, 0($s1)            #add '\n' character to end of string


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 $s4, 0($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. Kết quả (Mã 999 vẫn bị lỗi đôi khi được đôi khi không)


Digital Lab Sim
Keyboard and Display MMIO Simulator
This is the MarsBot
Clear
Close
Reset
Help
Disconnect from MIPS
Activate Windows
Go to Settings to activate Windows.


Digital Lab Sim
Keyboard and Display MMIO Simulator
This is the MarsBot
Clear
Close
Reset
Help
Disconnect from MIPS
Activate Windows
Go to Settings to activate Windows.

Bài 7:

***Đề bài:**

7. Chương trình kiểm tra cú pháp lệnh MIPS

Trình biên dịch của bộ xử 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*
- *K*iểm 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.*

Gợi ý: nên xây dựng một cấu trúc chứa khuôn dạng của từng lệnh với tên lệnh, kiểu của toán hạng 1, toán hạng 2, toán hạng 3.

***Ý Tưởng**:

Xây dựng một cấu trúc chứa khuôn dạng của từng lệnh với tên lệnh(opcode),kiểu toán hạng1, kiểu toán hạng 2, kiểu toán hạng 3. Và một số cấu trúc chứa tên thanh ghi và các ký tự in hoa và in thường. Sau đó lần lượt kiểm tra xâu nhập vào có tuân thủ theo các cấu trúc mà mình đặt ra hay không.

***Chi tiết thuật toán:**

1. Khởi tạo data:

```
.data
      command: .space 100
      opcode:  .space 10
      token:   .space 20
      number:  .space 15
      ident:   .space 30
      Nhap:    .asciiz "Nhap vao mot lenh hop ngu: "
      hopLe1:  .asciiz "Opcode: "
      hopLe2:  .asciiz "Toan hang: "
      hopLe3:  .asciiz "Hop le.\n"
      lamlai:  .asciiz "Ban muon thuc hien lai?(0.Dong y/1.De sau)"
      lenhloi: .asciiz "Lenh hop ngu khong hop le. Loi cu phap!\n"
      lenhloi1: .asciiz "Khong tim duoc khuon dang lenh nay!\n"
      ketthuc:  .asciiz "\nHoan thanh! Lenh phu hop voi cu phap!\n"
      # Quy luat cua khuon dang lenh: opcode co do dai = 5 byte
      # moi lenh co 3 toan hang va chi co 4 loai la: thanh ghi = 1, hang
so nguyen =2, dinh danh = 3 hoac khong co = 0.
khuondang:.asciiz"or***111;xor**111;lui**120;jr***100;jal**300;addi*11;ad
d**111;sub**111;ori**112;and**111;beq**113;bne**113;j****300;nop**000;"
      charGroup: .asciiz
"qwertyuiopasdfghjklmnbvcxzQWERTYUIOPASDFGHJKLZXCVBNM_"
      tenthanhghi: .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   $fp   $ra
$0    $1    $2    $3    $4    $5    $7    $8    $9    $10   $11   $12
$13   $14   $15   $16   $17   $18   $19   $20   $21   $22   $21   $22
$23   $24   $25   $26   $27   $28   $29   $30   $31   "
```

- Khởi tạo các mảng cần dùng (sẽ giải thích công dụng của từng cái khi dùng đến).
- Tạo những xâu cần dùng.
- Khởi tạo cấu trúc.

- Ta sẽ tạo một xâu khuondang gồm các chuỗi 8 bit được ngăn cách nhau bằng dấu phảy với 5 bit đầu là opcode và 3 bit sau tương ứng với dạng của 3 toán hạng với (0 là không có, 1 là thanh ghi, 2 là số nguyên và 3 là nhãn).
- Xâu charGroup là các chữ cái từ a đến z viết hoa và viết thường để tiện cho việc kiểm tra nhãn
- Xâu tenthanhghi là chuỗi các thanh ghi được ngăn cách với nhau bằng dấu space dùng cho việc kiểm tra tên thanh ghi có đúng hay không.

2.Các chương trình con:

*Chương trình nhập lệnh từ bàn phím:

Lưu phần xâu tính từ ký tự đầu tiên đến dấu cách thì dừng vào mảng Opcode
Chức năng:Lưu lệnh vào mảng Opcode

```
readData:
      li $v0, 4
      la $a0, Nhap
      syscall
      li $v0, 8
      la $a0, command
      li $a1, 100
      syscall
Chức năng:Lưu địa chỉ của lệnh vào $a0
li $t2, 0 # i
readOpcode:
      la $a1, opcode # luu cac ki tu doc duoc vao opcode
      add $t3, $a0, $t2 # dich bit
      add $t4, $a1, $t2
      lb $t1, 0($t3) # doc tung ki tu cua command
      sb $t1, 0($t4)
      beq $t1, 32, done # gap ki tu '  '  luu ki tu nay vao opcode de xu
ly
      beq $t1, 0, done # ket thuc chuoi command
      addi $t2, $t2, 1
      j readOpcode
```

Chức năng:

*Xử lý Opcode: beq $s1,2,ad j abc

Đầu tiên khởi tạo biến $t7 làm con trỏ cho khuondang(vì khuondang chứa các chuỗi 8 bit và dấu , nên nó sẽ đc khởi tạo ở -9.)
xuLyOpcode:
Lần lượt lấy ra 9 bit từ khuondang để đối chiếu với Opcode
compare:
Lần lượt đối chiếu 5 đầu bit trong khuondang với Opcode(xử dụng thuật toán so sánh 2 xâu nếu giống nhau thì báo hợp lệ còn không thì sẽ chuyển sang khuôn lệnh tiếp theo)

check:
Kiểm tra xem Opcode đã nhập có thực sự giống hay chỉ là giống mỗi đoạn đầu(Vd nếu ta nhập addd khi so sánh vs add** thì nó sẽ vẫn hiểu là 3 bit đầu giống nhau opcode hợp lệ vì thế sau khi so sánh đến khi gặp dấu * ta sẽ kiểm tra xem sau đó có phải kí tự space nếu phải kí tự hợp lệ còn không chuyển đến khuôn dạng tiếp theo.
.
checkContinue:
Lưu vị trí để xử lý token trong comand thông báo opcode có hệp lệ hay không
check2:
Dành cho những lệnh chỉ có opcode nếu sau đó không phải ký tự xuống dòng thì không hợp lệ

- Xử lý toán hạng

readToanHang1:
kiểm tra bit số 5 xem là 1,2,3 sau đó tiến hành kiểm tra tương ứng
Nếu là 0 thì nhảy đến nhãn checkNT:
1 thì nhảy đến nhãn checkTokenReg:
2 thì nhảy đến nhãn checkHSN:
3 thì nhảy đến nhãn checkIdent:
checkTokenReg: lưu xâu vào mảng token để so sánh lần lượt với các thanh ghi ở trong mảng

thanh ghi.vẫn áp dụng thuật toán so sánh xâu nếu gặp dấu “,” ngừng so sánh nếu gặp dấu space bỏ qua
checkHSN:lưu xâu vào mảng number xem nó có giống trong numberGroup hay không nếu số không năm trong đoạn từ 0 đến 9 thì nhảy đến nhãn kết thúc và số không hợp lệ
checkIdent:lưu nhãn vào để so sánh xem nó có giống trong charGroup hay không nếu số không
với thuật toán so sanh xâu.
Kèm theo một biến $v1 để kiểm tra xem đang là toán hạng thứ mấy.để nhảy đến các nhãn readToanHang2: hoặc readToanHang3:

Hàm toán hạng 2 và 3 giống toán hạng 1.
*nhãn continue: và resetAll: nếu người dùng chọn làm lại thì reset toàn bộ các biến về ban đầu và chạy lại

***Code chính thức:**

```
.data
      command: .space 150
      opcode: .space 15
      token: .space 25
      number: .space 20
      ident: .space 50
      Nhap: .asciiz "Nhap vao mot lenh hop ngu: "
      hopLe1: .asciiz "Opcode: "
      hopLe2: .asciiz "Toan hang: "
      hopLe3: .asciiz "Hop le.\n"
      lamlai: .asciiz "Ban muon thuc hien lai?(0.Dong y/1.De sau)"
      lenhloi: .asciiz "Lenh hop ngu khong hop le. Loi cu phap!\n"
      lenhloi1: .asciiz "Khong tim duoc khuon dang lenh nay!\n"
      ketthuc: .asciiz "\nHoan thanh! Lenh phu hop voi cu phap!\n"
      # Quy luat cua libray: opcode co do dai = 5 byte
      # moi lenh co 3 toan hang va chi co 4 loai la: thanh ghi = 1, hang so
nguyen =2, dinh danh = 3 hoac khong co = 0.
      khuondang: .asciiz
"or***111;xor**111;lui**120;jr***100;jal**300;addi*112;add**111;sub**111;ori*
*112;and**111;beq**113;bne**113;j****300;nop**000;"
      charGroup: .asciiz
"qwertyuiopasdfghjklmnbvcxzQWERTYUIOPASDFGHJKLZXCVBNM_"
      tenthanhghi: .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   $fp   $ra   $0    $1
$2    $3    $4    $5    $7    $8    $9    $10   $11   $12   $13   $14   $15
$16   $17   $18   $19   $20   $21   $22   $21   $22   $23   $24   $25   $26
$27   $28   $29   $30   $31   "

.text

readData: # Doc lenh nhap vao tu ban phim
      li $v0, 4
      la $a0, Nhap
      syscall
      li $v0, 8
      la $a0, command # chua dia chi cua lenh nhap vao
      li $a1, 100
      syscall
#main
      li $t2, 0 # i
readOpcode:
      la $a1, opcode # luu cac ki tu doc duoc vao opcode
```

```
      add $t3, $a0, $t2 # dich bit
      add $t4, $a1, $t2
      lb $t1, 0($t3) # doc tung ki tu cua command
      sb $t1, 0($t4)
      beq $t1, 32, done # gap ki tu ' ' -> luu ki tu nay vao opcode de xu ly
      beq $t1, 0, done # ket thuc chuoi command
      addi $t2, $t2, 1
      j readOpcode

#<--xu ly opcode-->
done:
      li $t7,-9
      la $a2, khuondang
xuLyOpcode:
      li $t1, 0 # i
      li $t2, 0 # j
      addi $t7,$t7,9 # buoc nhay = 9 de den vi tri opcode trong library
      add $t1,$t1,$t7 # cong buoc nhay

      compare:
      add $t3, $a2, $t1 # t3 tro thanh con tro cua library
      lb $s0, 0($t3)
      beq $s0, 0, notFound # khong tim thay opcode nao trong library
      beq $s0, 42, check # gap ki tu '*' -> check xem opcode co giong nhau
tiep ko?.
      add $t4, $a1, $t2
      lb $s1, 0($t4)
      bne $s0,$s1,xuLyOpcode # so sanh 2 ki tu. dung thi so sanh tiep, sai
thi nhay den phan tu chua khuon danh lenh tiep theo.
      addi $t1,$t1,1 # i+=1
      addi $t2,$t2,1 # j+=1
      j compare

      check:
      add $t4, $a1, $t2
      lb $s1, 0($t4)
      bne $s1, 32, check2 # neu ki tu tiep theo khong phai ' ' => lenh khong
hop le. chi co doan dau giong.
      checkContinue:
      add $t9,$t9,$t2 # t9 = luu vi tri de xu ly token trong command
      li $v0, 4
      la $a0, hopLe1 # opcode hop le
      syscall
      li $v0, 4
      la $a0, opcode
      syscall
      li $v0, 4
      la $a0, hopLe3
      syscall
      j readToanHang1

      check2: # neu ki tu tiep theo khong phai '\n' => lenh khong hop le. chi
co doan dau giong.
      bne $s1, 10, notFound
      j checkContinue

# <!--ket thuc xu ly opcode -->

#<--xu li toan hang-->

readToanHang1:
```

```
	# xac dinh kieu toan hang trong library
	# t7 dang chua vi tri khuon dang lenh trong library
	li $t1, 0
	addi $t7, $t7, 5 # chuyen den vi tri toan hang 1 trong library
	add $t1, $a2, $t7 # a2 chua dia chi library
	lb $s0, 0($t1)
	addi $s0,$s0,-48 # chuyen tu char -> int
	li $t8, 1 # thanh ghi = 1
	beq $s0, $t8, checkTokenReg
	li $t8, 2 # hang so nguyen = 2
	beq $s0, $t8, checkHSN
	li $t8, 3 # dinh danh = 3
	beq $s0, $t8, checkIdent
	li $t8, 0 # khong co toan hang = 0
	beq $s0, $t8, checkNT
	j end

#<--check Token Register-->
checkTokenReg:
	la $a0, command
	la $a1, token # luu ten thanh ghi vao token de so sanh
	li $t1, 0
	li $t2, -1
	addi $t1, $t9, 0
	readToken:
		addi $t1, $t1, 1 # i
		addi $t2, $t2, 1 # j
		add $t3, $a0, $t1
		add $t4, $a1, $t2
		lb $s0, 0($t3)
		add $t9, $zero, $t1 # vi tri toan hang tiep theo trong command
		beq $s0, 32, checkTokenReg # gap dau ' '
		beq $s0, 44, readTokenDone # gap dau ','
		beq $s0, 0, readTokenDone # gap ki tu ket thuc
		sb $s0, 0($t4)
		j readToken

	readTokenDone:
		sb $s0, 0($t4) # luu them ',' vao de compare
		li $t1, -1 # i
		li $t2, -1 # j
		li $t4, 0
		li $t5, 0
		add $t2, $t2, $k1
		la $a1, token
		la $a2, tenthanhghi
		j compareToken

compareToken:
	addi $t1,$t1,1
	addi $t2,$t2,1
	add $t4, $a1, $t1
	lb $s0, 0($t4)
	beq $s0, 0, end
	add $t5, $a2, $t2
	lb $s1, 0($t5)
	beq $s1, 0, notFound
	beq $s1, 32, checkLengthToken
	bne $s0,$s1, jump
	j compareToken
```

```
      checkLengthToken:
            beq $s0, 44, compare1
            beq $s0, 10, compare1
            j compare2
      jump:
            addi $k1,$k1,6
            j readTokenDone
      compare1:
            la $a0, hopLe2 # opcode hop le
            syscall
            li $v0, 4
            la $a0, token
            syscall
            li $v0, 4
            la $a0, hopLe3
            syscall
            addi $v1, $v1, 1 # dem so toan hang da doc.
            li $k1, 0 # reset buoc nhay
            beq $v1, 1, readToanHang2
            beq $v1, 2, readToanHang3
            j end
      compare2:
            j notFound
#<!--ket thuc check Token Register-->

#<--check toan hang la hang so nguyen-->
checkHSN: # kiem tra co phai hang so nguyen hay ko
      la $a0, command
      la $a1, number # luu day chu so vao number de so sanh tung chu so co
thuoc vao numberGroup hay khong.
      li $t1, 0
      li $t2, -1
      addi $t1, $t9, 0
      readNumber:
            addi $t1, $t1, 1 # i
            addi $t2, $t2, 1 # j
            add $t3, $a0, $t1
            add $t4, $a1, $t2
            lb $s0, 0($t3)
            add $t9, $zero, $t1 # vi tri toan hang tiep theo trong command
            beq $s0, 32, checkHSN # gap dau ' '
            beq $s0, 44, readNumberDone # gap dau ','
            beq $s0, 0, readNumberDone # gap ki tu ket thuc
            sb $s0, 0($t4)
            j readNumber
      readNumberDone:
            sb $s0, 0($t4) # luu them ',' vao de compare
            li $t1, -1 # i
            li $t4, 0
            la $a1, number
            j compareNumber
compareNumber:
      addi $t1, $t1, 1
      add $t4, $a1, $t1
      lb $s0, 0($t4)
      beq $s0, 0, end
      beq $s0, 45, compareNumber # bo dau '-'
      beq $s0, 10, compareNum1
      beq $s0, 44, compareNum1
      li $t2, 48
      li $t3, 57
```

```
      slt $t5, $s0, $t2
      bne $t5, $zero, compareNum2
      slt $t5, $t3, $s0
      bne $t5, $zero, compareNum2
      j compareNumber

      compareNum1:
            la $a0, hopLe2
            syscall
            li $v0, 4
            la $a0, number
            syscall
            li $v0, 4
            la $a0, hopLe3
            syscall
            addi $v1, $v1, 1 # dem so toan hang da doc.
            li $k1, 0 # reset buoc nhay
            beq $v1, 1, readToanHang2
            beq $v1, 2, readToanHang3
            j end
      compareNum2:
            j notFound
#<!--ket thuc check toan hang la hang so nguyen-->

#<--check Indent-->
checkIdent:
      la $a0, command
      la $a1, ident # luu ten thanh ghi vao indent de so sanh
      li $t1, 0
      li $t2, -1
      addi $t1, $t9, 0
      readIndent:
            addi $t1, $t1, 1 # i
            addi $t2, $t2, 1 # j
            add $t3, $a0, $t1
            add $t4, $a1, $t2
            lb $s0, 0($t3)
            add $t9, $zero, $t1 # vi tri toan hang tiep theo trong command
            beq $s0, 32, checkIdent # gap dau ' '
            beq $s0, 44, readIdentDone # gap dau ','
            beq $s0, 0, readIdentDone # gap ki tu ket thuc
            sb $s0, 0($t4)
            j readIndent
      readIdentDone:
            sb $s0, 0($t4) # luu them ',' vao de compare
            loopj:
            li $t1, -1 # i
            li $t2, -1 # j
            li $t4, 0
            li $t5, 0
            add $t1, $t1, $k1
            la $a1, ident
            la $a2, charGroup
            j compareIdent
compareIdent:
      addi $t1,$t1,1
      add $t4, $a1, $t1
      lb $s0, 0($t4)
      beq $s0, 0, end
      beq $s0, 10, compareIdent1
      beq $s0, 44, compareIdent1
```

```
      loop:
      addi $t2,$t2,1
      add $t5, $a2, $t2
      lb $s1, 0($t5)
      beq $s1, 0, compareIdent2
      beq $s0, $s1, jumpIdent # so sanh ki tu tiep theo trong ident
      j loop # tiep tuc so sanh ki tu tiep theo trong charGroup

      jumpIdent:
            addi $k1,$k1,1
            j loopj

      compareIdent1:
            la $a0, hopLe2 # opcode hop le
            syscall
            li $v0, 4
            la $a0, ident
            syscall
            li $v0, 4
            la $a0, hopLe3
            syscall
            addi $v1, $v1, 1 # dem so toan hang da doc.
            li $k1, 0 # reset buoc nhay
            beq $v1, 1, readToanHang2
            beq $v1, 2, readToanHang3
            j end
      compareIdent2:
            j notFound
#<!--ket thuc check Indent-->

#<--kiem tra khong co toan hang-->
checkNT:
      la $a0, command
      li $t1, 0
      li $t2, 0
      addi $t1, $t9, 0
      add $t2, $a0, $t1
      lb $s0, 0($t2)
      addi $v1, $v1, 1 # dem so toan hang da doc.
      li $k1, 0 # reset buoc nhay
      beq $v1, 1, readToanHang2
      beq $v1, 2, readToanHang3
#<!--ket thuc kiem tra khong co toan hang-->

#<--check Token Register 2-->
readToanHang2:
      # xac dinh kieu toan hang trong khuondang
      # t7 dang chua vi tri khuon dang lenh trong khuondang
      li $t1, 0
      la $a2,khuondang
      addi $t7, $t7, 1 # chuyen den vi tri toan hang 2 trong kuondang
      add $t1, $a2, $t7 # a2 chua dia chi library
      lb $s0, 0($t1)
      addi $s0,$s0,-48 # chuyen tu char -> int
      li $t8, 1 # thanh ghi = 1
      beq $s0, $t8, checkTokenReg
      li $t8, 2 # hang so nguyen = 2
      beq $s0, $t8, checkHSN
      li $t8, 3 # dinh danh = 3
      beq $s0, $t8, checkIdent
      li $t8, 0 # khong co toan hang = 0
```

```
      beq $s0, $t8, checkNT
      j end
#<!--ket thuc check Token Register 2-->

#<--check Token Register 3-->
readToanHang3:
      # xac dinh kieu toan hang trong khuondang
      # t7 dang chua vi tri khuon dang lenh trong khuondang
      li $t1, 0
      la $a2, khuondang
      addi $t7, $t7, 1 # chuyen den vi tri toan hang 3 trong khuondang
      add $t1, $a2, $t7 # a2 chua dia chi khuondang
      lb $s0, 0($t1)
      addi $s0,$s0,-48 # chuyen tu char -> int
      li $t8, 1 # thanh ghi = 1
      beq $s0, $t8, checkTokenReg
      li $t8, 2 # hang so nguyen = 2
      beq $s0, $t8, checkHSN
      li $t8, 3 # dinh danh = 3
      beq $s0, $t8, checkIdent
      li $t8, 0 # khong co toan hang = 0
      beq $s0, $t8, checkNT
      j end
#<!--ket thuc check Token Register 3-->
#<--check Token Register 3-->
continue: # lap lai chuong trinh.
      li $v0, 4
      la $a0, lamlai
      syscall
      li $v0, 5
      syscall
      add $t0, $v0, $zero
      beq $t0, $zero, resetAll
      j TheEnd
resetAll:
      li $v0, 0
      li $v1, 0
      li $a0, 0
      li $a1, 0
      li $a2, 0
      li $a3, 0
      li $t0, 0
      li $t1, 0
      li $t2, 0
      li $t3, 0
      li $t4, 0
      li $t5, 0
      li $t6, 0
      li $t7, 0
      li $t8, 0
      li $t9, 0
      li $s0, 0
      li $s1, 0
      li $s2, 0
      li $s3, 0
      li $s4, 0
      li $s5, 0
      li $s6, 0
      li $s7, 0
      li $k0, 0
      li $k1, 0
```

```
      j readData
notFound:
      li $v0, 4
      la $a0, lenhloi1
      syscall
      j TheEnd
error:
      li $v0, 4
      la $a0, lenhloi
      syscall
      j TheEnd
end:
      li $v0, 4
      la $a0, ketthuc
      syscall
      j continue
TheEnd:
```

***Chạy thử**

Case 1: chạy các lệnh bình thường chạy thử với 5 lệnh add,addi,beq,j,

```
Nhap vao mot lenh hop ngu: addi $t1, $t1, 1
Opcode: addi Hop le.
Toan hang: $t1,Hop le.
Toan hang: $t1,Hop le.
Toan hang: 1
Hop le.

Hoan thanh! Lenh phu hop voi cu phap!
Ban muon thuc hien lai?(0.Dong y/1.De sau)
```

```
Nhap vao mot lenh hop ngu: addi $t1, $t1, 1
Opcode: addi Hop le.
Toan hang: $t1,Hop le.
Toan hang: $t1,Hop le.
Toan hang: 1
Hop le.

Hoan thanh! Lenh phu hop voi cu phap!
Ban muon thuc hien lai?(0.Dong y/1.De sau)
```

```
Nhap vao mot lenh hop ngu: beq $s0,$t1,abc
Opcode: beq Hop le.
Toan hang: $s0,Hop le.
Toan hang: $t1,Hop le.
Toan hang: abc
Hop le.

Nhap vao mot lenh hop ngu: j abcyu
Opcode: j Hop le.
Toan hang: abcyu
Hop le.

Hoan thanh! Lenh phu hop voi cu phap!
Ban muon thuc hien lai?(0.Dong y/1.De sau)
```

Case2:lệnh có các dấu cách lung tung

```
Nhap vao mot lenh hop ngu: beq         $s1,       34,   aaaa1
Opcode: beq Hop le.
Toan hang: $s1,Hop le.
Toan hang: 34,Hop le.
Toan hang: aaaa1
Hop le.

Hoan thanh! Lenh phu hop voi cu phap!
Ban muon thuc hien lai?(0.Dong y/1.De sau)
```

Case 3: lệnh có chú thích

```
Nhap vao mot lenh hop ngu: addi $s1,$s2,45 #
Opcode: addi Hop le.
Toan hang: $s1,Hop le.
Toan hang: $s2,Hop le.
Toan hang: 45Hop le.

Hoan thanh! Lenh phu hop voi cu phap!
Ban muon thuc hien lai?(0.Dong y/1.De sau)
```