Laboratory Exercise 10 (p2)

|  |  |
| --- | --- |
| Tên | Đặng Hồng Minh |
| MSSV | 20225740 |

A screen shot of a computer

Description automatically generatedAssignment 1

.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

.text

main: jal TRACK # draw track line

nop

addi $a0, $zero, 180 # Marsbot rotates to 180\* and start running

jal ROTATE

nop

jal GO

nop

sleep1: addi $v0, $zero, 32 # Keep running by sleeping in 5000 ms

li $a0, 5000

syscall

jal UNTRACK # keep old track

nop

jal TRACK # and draw new track line

nop

goUP: addi $a0, $zero, 0 # Marsbot rotates to 0\*

jal ROTATE

nop

sleep2: addi $v0, $zero, 32 # Keep running by sleeping in 5000 ms

li $a0, 5000

syscall

jal UNTRACK # keep old track

nop

jal TRACK # and draw new track line

nop

goASKEWDOWN: addi $a0, $zero, 135 # Marsbot rotates to 135\*

jal ROTATE

nop

sleep3: addi $v0, $zero, 32 # Keep running by sleeping in 2500 ms

li $a0, 2500

syscall

jal UNTRACK # keep old track

nop

jal TRACK # and draw new track line

nop

goASKEWUP: addi $a0, $zero, 45 # Marsbot rotates to 45\*

jal ROTATE

nop

sleep4: addi $v0, $zero, 32 # Keep running by sleeping in 2500 ms

li $a0, 2500

syscall

jal UNTRACK # keep old track

nop

jal TRACK # and draw new track line

nop

goDOWN: addi $a0, $zero, 180 # Marsbot rotates to 180\*

jal ROTATE

nop

sleep5: addi $v0, $zero, 32 # Keep running by sleeping in 5000 ms

li $a0, 5000

syscall

jal UNTRACK # keep old track

nop

end\_main:

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

# GO procedure, to start running

# param[in] none

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

GO: li $at, MOVING # change MOVING port

addi $k0, $zero, 1 # to logic 1,

sb $k0, 0($at) # to start running

nop

jr $ra

nop

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

# STOP procedure, to stop running

# param[in] none

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

STOP: li $at, MOVING # change MOVING port to 0

sb $zero, 0($at) # to stop

nop

jr $ra

nop

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

# TRACK procedure, to start drawing line

# param[in] none

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

TRACK: li $at, LEAVETRACK # change LEAVETRACK port

addi $k0, $zero, 1 # to logic 1,

sb $k0, 0($at) # to start tracking

nop

jr $ra

nop

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

# UNTRACK procedure, to stop drawing line

# param[in] none

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

UNTRACK:li $at, LEAVETRACK # change LEAVETRACK port to 0

sb $zero, 0($at) # to stop drawing tail

nop

jr $ra

nop

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

# ROTATE procedure, to rotate the robot

# param[in] $a0, An angle between 0 and 359

# 0 : North (up)

# 90: East (right)

# 180: South (down)

# 270: West (left)

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

ROTATE: li $at, HEADING # change HEADING port

sw $a0, 0($at) # to rotate robot

nop

jr $ra

nop

Assignment 2

.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 DISPLAY\_CODE 0xFFFF000C # ASCII code to show, 1 byte

.eqv DISPLAY\_READY 0xFFFF0008 # =1 if the display has already to do

# Auto clear after sw

.text

li $k0, KEY\_CODE

li $k1, KEY\_READY

li $s0, DISPLAY\_CODE

li $s1, DISPLAY\_READY

loop: nop

WaitForKey:

lw $t1, 0($k1) # $t1 = [$k1] = KEY\_READY

beq $t1, $zero, WaitForKey # if $t1 == 0 then Polling

ReadKey:

lw $t0, 0($k0) # $t0 = [$k0] = KEY\_CODE

j check\_e

WaitForDis:

lw $t2, 0($s1) # $t2 = [$s1] = DISPLAY\_READY

beq $t2, $zero, WaitForDis # if $t2 == 0 then Polling

Encrypt: addi $t0, $t0, 3 # change input key

ShowKey:

sw $t0, 0($s0) # show key

nop

j loop

check\_e:

beq $t3, 'e', check\_x # if character e exist then check x

bne $t0, 'e', WaitForDis # if character != e then continue

add $t3, $t0, $0 # else $t3 = 'e'

j WaitForDis

check\_x:

beq $t4, 'x', check\_i # if character x exist then check i

bne $t0, 'x', reset # if character != i then continue

add $t4, $t0, $0 # else $t4 = 'x'

j WaitForDis

check\_i:

beq $t5, 'i', check\_t # if character i exist then check t

bne $t0, 'i', reset # if character != t then continue

add $t5, $t0, $0 # else $t5 = 't'

j WaitForDis

check\_t:

beq $t0, 't', exit # if character t exist then exit

j reset # if character != t then continue

reset: li $t3, 0 # reset 'e' to 0

li $t4, 0 # reset 'x' to 0

li $t5, 0 # reset 'i' to 0

j WaitForDis

exit: li $v0, 10

syscall

