Lab Report 10.2

## Assignment 1

**Code:**

.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:

triangle:

jal UNTRACK

jal GO

li $a0, 90

jal ROTATE

li $v0, 32

li $a0, 1000

syscall

li $a0, 180

jal ROTATE

li $v0, 32

li $a0, 1000

syscall

li $a0, 90

jal ROTATE

li $v0, 32

li $a0, 500

syscall

li $a0, 150

jal ROTATE

jal TRACK

li $v0, 32

li $a0, 1000

syscall

jal UNTRACK

li $a0, 270

jal ROTATE

jal TRACK

li $v0, 32

li $a0, 1000

syscall

jal UNTRACK

li $a0, 30

jal ROTATE

jal TRACK

li $v0, 32

li $a0, 1000

syscall

jal UNTRACK

square:

li $a0, 90

jal ROTATE

li $v0, 32

li $a0, 1000

syscall

jal TRACK

li $v0, 32

li $a0, 1000

syscall

jal UNTRACK

li $a0, 180

jal ROTATE

jal TRACK

li $v0, 32

li $a0, 1000

syscall

jal UNTRACK

li $a0, 270

jal ROTATE

jal TRACK

li $v0, 32

li $a0, 1000

syscall

jal UNTRACK

li $a0, 0

jal ROTATE

jal TRACK

li $v0, 32

li $a0, 1000

syscall

jal UNTRACK

star:

li $a0, 90

jal ROTATE

li $v0, 32

li $a0, 2000

syscall

li $a0, 162

jal ROTATE

jal TRACK

li $v0, 32

li $a0, 1000

syscall

jal UNTRACK

li $a0, 306

jal ROTATE

jal TRACK

li $v0, 32

li $a0, 1000

syscall

jal UNTRACK

li $a0, 90

jal ROTATE

jal TRACK

li $v0, 32

li $a0, 1000

syscall

jal UNTRACK

li $a0, 234

jal ROTATE

jal TRACK

li $v0, 32

li $a0, 1000

syscall

jal UNTRACK

li $a0, 18

jal ROTATE

jal TRACK

li $v0, 32

li $a0, 1000

syscall

jal UNTRACK

li $a0, 90

jal ROTATE

li $v0, 32

li $a0, 1000

syscall

end\_main:

jal STOP

li $v0, 10

syscall

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

# 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

jr $ra

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

# STOP procedure, to stop running

# param[in] none

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

STOP:

li $at, MOVING # change MOVING port to 0

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

jr $ra

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

# 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

jr $ra

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

# 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

jr $ra

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

# 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

jr $ra

**Results:**

Graphical user interface, text

Description automatically generated

## Assignment 2

**Code:**

.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

WaitForDis:

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

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

Check:

add $t8, $zero, $t7

add $t7, $zero, $t6

add $t6, $zero, $t5

add $t5, $zero, $t0

bne $t8, 'e', NotExit

bne $t7, 'x', NotExit

bne $t6, 'i', NotExit

bne $t5, 't', NotExit

j EndLoop

NotExit:

sge $t3, $t0, 'a'

sle $t4, $t0, 'z'

and $t3, $t3, $t4

beq $t3, 1, IsLower

sge $t3, $t0, 'A'

sle $t4, $t0, 'Z'

and $t3, $t3, $t4

beq $t3, 1, IsUpper

sge $t3, $t0, '0'

sle $t4, $t0, '9'

and $t3, $t3, $t4

beq $t3, 1, ShowKey

j IsOther

IsLower:

add $t0, $t0, -32

j ShowKey

IsUpper:

add $t0, $t0, +32

j ShowKey

IsOther:

li $t0, '\*'

j ShowKey

ShowKey:

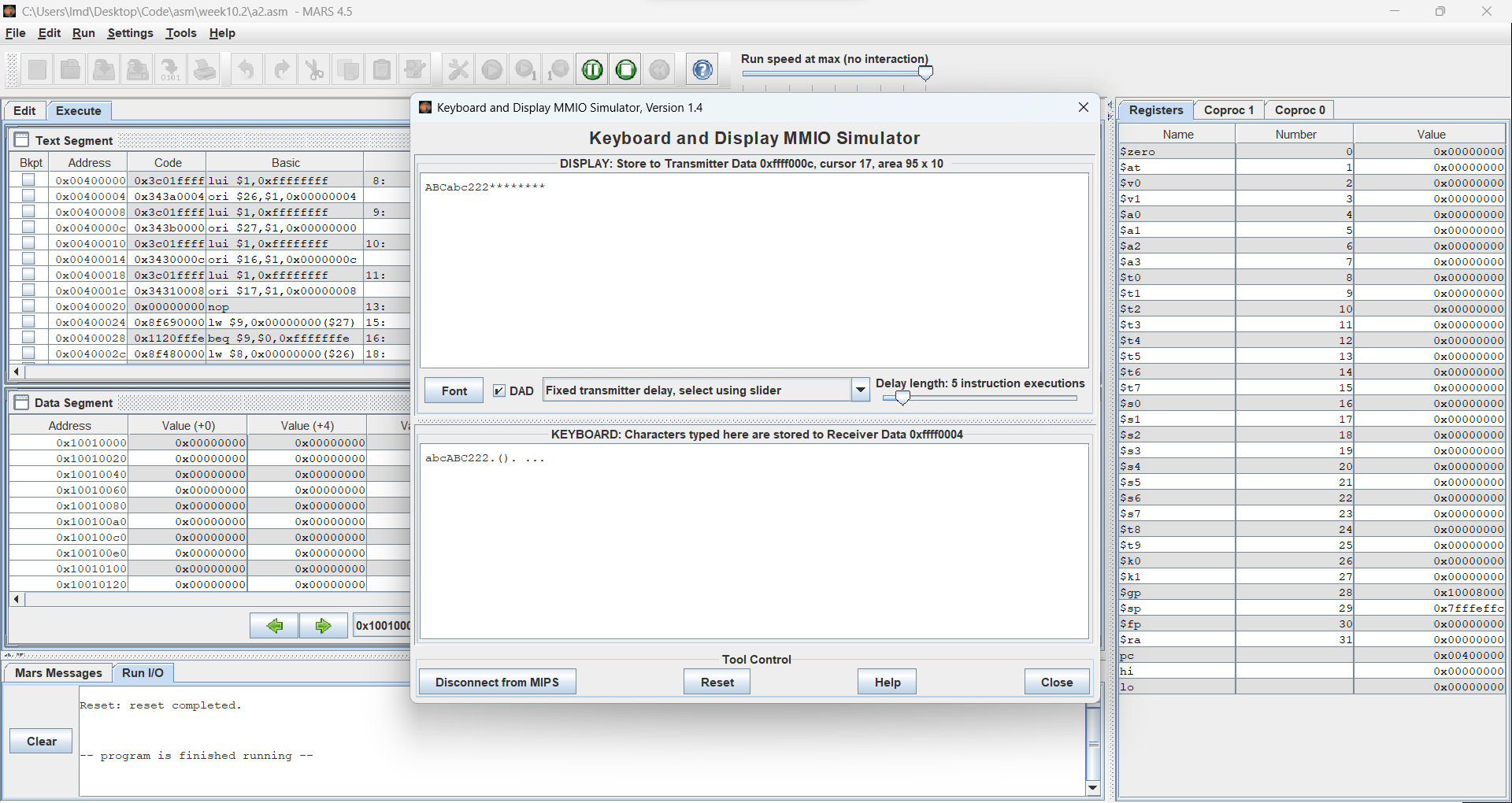
sw $t0, 0($s0) # show key

nop

j Loop

EndLoop:

**Results:**



## Assignment 3

**Code:**

.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

.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

.text

li $t6, KEY\_CODE

li $t7, KEY\_READY

li $s0, DISPLAY\_CODE

li $s1, DISPLAY\_READY

li $t8, 0 # 0: STOP, 1: GO

li $t9, 0 # 0: UNTRACK, 1: TRACK

jal UNTRACK

li $a0, 180

Loop:

nop

WaitForKey:

lw $t1, 0($t7) # $t1 = [$t7] = KEY\_READY

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

WaitForDis:

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

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

ReadKey:

lw $t0, 0($t6) # $t0 = [$t6] = KEY\_CODE

Check:

jal ROTATE

beq $t0, 32, ToggleGo

beq $t0, '\n', ToggleTrack

beq $t0, 'w', TurnUp

beq $t0, 'W', TurnUp

beq $t0, 's', TurnDown

beq $t0, 'S', TurnDown

beq $t0, 'a', TurnLeft

beq $t0, 'A', TurnLeft

beq $t0, 'd', TurnRight

beq $t0, 'D', TurnRight

j ShowKey

ToggleGo:

beq $t8, 0, RunGo

beq $t8, 1, RunStop

RunGo:

li $t8, 1

jal GO

j ShowKey

RunStop:

li $t8, 0

jal STOP

j ShowKey

ToggleTrack:

beq $t9, 0, RunTrack

beq $t9, 1, RunUntrack

RunTrack:

li $t9, 1

jal TRACK

j ShowKey

RunUntrack:

li $t9, 0

jal UNTRACK

j ShowKey

TurnUp:

li $a0, 0

jal ROTATE

j ShowKey

TurnDown:

li $a0, 180

jal ROTATE

j ShowKey

TurnLeft:

li $a0, 270

jal ROTATE

j ShowKey

TurnRight:

li $a0, 90

jal ROTATE

j ShowKey

ShowKey:

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

nop

j Loop

EndLoop:

jal STOP

li $v0, 10

syscall

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

# 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

jr $ra

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

# STOP procedure, to stop running

# param[in] none

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

STOP:

li $at, MOVING # change MOVING port to 0

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

jr $ra

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

# 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

jr $ra

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

# 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

jr $ra

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

# 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

jr $ra

**Results:**

A screenshot of a computer

Description automatically generated