**COMPUTER ARCHITECTURE LAB**

**ASSEMBLY LANGUAGE**

STUDENT NAME: PHI QUOC HUNG

STUDENT ID: 20184270

REPORT LAB 10

**ASSIGNMENT 1:**

- Code:

.eqv SEVENSEG\_LEFT 0xFFFF0011 # Dia chi cua den led 7 doan trai.

# Bit 0 = doan a;

# Bit 1 = doan b; ...

# Bit 7 = dau .

.eqv SEVENSEG\_RIGHT 0xFFFF0010 # Dia chi cua den led 7 doan phai

.text

main:

li $a0, 0x7F # set value for segments

jal SHOW\_7SEG\_LEFT # show

nop

li $a0, 0x7F # set value for segments

jal SHOW\_7SEG\_RIGHT # show

nop

exit:

li $v0, 10

syscall

endmain:

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

# Function SHOW\_7SEG\_LEFT : turn on/off the 7seg

# param[in] $a0 value to shown

# remark $t0 changed

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

SHOW\_7SEG\_LEFT:

li $t0, SEVENSEG\_LEFT # assign port's address

sb $a0, 0($t0) # assign new value

nop

jr $ra

nop

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

# Function SHOW\_7SEG\_RIGHT : turn on/off the 7seg

# param[in] $a0 value to shown

# remark $t0 changed

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

SHOW\_7SEG\_RIGHT:

li $t0, SEVENSEG\_RIGHT # assign port's address

sb $a0, 0($t0) # assign new value

nop

jr $ra

nop

- The solution observed:

Table

Description automatically generated with medium confidence

- Explanation:

The instruction li $a0,0x7F (16) equal to 01111111 (2), so all the led will light except the dot.

**ASSIGNMENT 2:**

- Code:

.eqv MONITOR\_SCREEN 0x10010000 #Dia chi bat dau cua bo nho man hinh

.eqv RED 0x00FF0000 #Cac gia tri mau thuong su dung

.eqv GREEN 0x0000FF00

.eqv BLUE 0x000000FF

.eqv WHITE 0x00FFFFFF

.eqv YELLOW 0x00FFFF00

.text

li $k0, MONITOR\_SCREEN #Nap dia chi bat dau cua man hinh

li $t0, WHITE

sw $t0, 0($k0)

nop

li $t0, WHITE

sw $t0, 4($k0)

nop

li $t0, WHITE

sw $t0, 8($k0)

nop

li $t0, YELLOW

sw $t0, 12($k0)

nop

li $t0, WHITE

sw $t0, 72($k0)

nop

li $t0, WHITE

sw $t0, 128($k0)

nop

li $t0, WHITE

sw $t0, 64($k0)

nop

li $t0, YELLOW

sw $t0, 68($k0)

nop

li $t0, YELLOW

sw $t0, 132($k0)

nop

li $t0, YELLOW

sw $t0, 76($k0)

nop

li $t0, YELLOW

sw $t0, 136($k0)

nop

li $t0, YELLOW

sw $t0, 140($k0)

- The result observed:

A picture containing text, screenshot, monitor

Description automatically generated

**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

.text

main:

jal TRACK # draw track line

nop

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

jal ROTATE

nop

jal GO

nop

sleep1:

addi $v0,$zero,32 # Keep running by sleeping in 2000 ms

li $a0,2000

syscall

jal UNTRACK # keep old track

nop

jal TRACK # and draw new track line

nop

goUpRight:

addi $a0, $zero, 30 # Marsbot rotates 30\*

jal ROTATE

nop

sleep2:

addi $v0,$zero,32 # Keep running by sleeping in 2000 ms

li $a0,2000

syscall

jal UNTRACK # keep old track

nop

jal TRACK # and draw new track line

nop

goDownRight:

addi $a0, $zero, 150 # Marsbot rotates 150\*

jal ROTATE

nop

sleep3:

addi $v0,$zero,32 # Keep running by sleeping in 2000 ms

li $a0,2000

syscall

jal UNTRACK # keep old track

nop

jal TRACK # and draw new track line

nop

goLeft:

addi $a0, $zero, 270 # Marsbot rotates 120\*

jal ROTATE

nop

sleep4:

addi $v0,$zero,32 # Keep running by sleeping in 2000 ms

li $a0,4000

syscall

jal UNTRACK # keep old track

nop

jal TRACK # and draw new track line

nop

j STOP

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

- The result observed:

Graphical user interface, application, Word

Description automatically generated

**ASSIGNMENT 4:**

- 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 is already to do

# Auto clear after sw

.eqv e\_KEYCODE 0x65 #ASCII code of each char of 'exit'

.eqv x\_KEYCODE 0x78

.eqv i\_KEYCODE 0x69

.eqv t\_KEYCODE 0x74

.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

nop

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

nop

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

ReadKey:

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

nop

j CheckE

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

WaitForDis:

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

nop

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

nop

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

Encrypt:

addi $t0, $t0, 1 # change input key

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

ShowKey:

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

nop

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

j loop

nop

CheckE: beq $t3, e\_KEYCODE, CheckX # check if exist e, jump check x

bne $t0, e\_KEYCODE, WaitForDis # if current char is not e, continue

add $t3, $t0, $zero # save e to $t3 if receive e

j WaitForDis

CheckX: beq $t4, x\_KEYCODE, CheckI # check if exist x, jump check for i

bne $t0, x\_KEYCODE, Reset # if current char is not x, reset and continue

add $t4, $t0, $zero # save x to $t4 if receive x

j WaitForDis

CheckI: beq $t5, i\_KEYCODE, CheckT # check if exist i, jump check for t

bne $t0, i\_KEYCODE, Reset # if current char is not i, reset then continue

add $t5, $t0, $zero # save i to $t5 if receive i

j WaitForDis

CheckT: beq $t0, t\_KEYCODE, Exit # check if meet t, exit

j Reset # if current char is not t, reset then continue

Reset: li $t3, 0 # dequeue 'e'

li $t4, 0 # dequeue 'x'

li $t5, 0 # dequeue 'i'

j WaitForDis

Exit:

li $v0, 10

syscall