Báo cáo thực hành môn Kiến trúc máy tính

Họ tên: Nguyễn Thanh Hưng

MSSV: 20225633

1. Assignment 1

Code:

    .eqv HEADING, 0xffff8010                 # Integer: An angle between 0 and 359

    .eqv MOVING, 0xffff8050                  # Boolean: whether or not to move

    .eqv LEAVETRACK, 0xffff8020              # Boolean (0 or non-0):

    .eqv WHEREX, 0xffff8030                  # Integer: Current x-location of MarsBot

    .eqv WHEREY, 0xffff8040                  # Integer: Current y-location of MarsBot

.text

main:

    addi    $a0,        $zero,      90

    jal     ROTATE

    jal     GO

sleep1:

    addi    $v0,        $zero,      32      # Keep running by sleeping in 1000 ms

    li      $a0,        3000

    syscall

Turn1:

    addi    $a0,        $zero,      180

    jal     ROTATE

    jal     GO

Sleep2:

    addi    $v0,        $zero,      32      # Keep running by sleeping in 1000 ms

    li      $a0,        3000

    syscall

Draw1:

    jal     TRACK

    addi    $v0,        $zero,      32      # Keep running by sleeping in 1000 ms

    li      $a0,        3000

    syscall

    jal     UNTRACK

Turn2:

    addi    $a0,        $zero,      90

    jal     ROTATE

    jal     GO

Sleep3:

    addi    $v0,        $zero,      32      # Keep running by sleeping in 1000 ms

    li      $a0,        3000

    syscall

Turn3:

    addi    $a0,        $zero,      360

    jal     ROTATE

    jal     GO

Draw2:

    jal     TRACK

    addi    $v0,        $zero,      32      # Keep running by sleeping in 1000 ms

    li      $a0,        3000

    syscall

    jal     UNTRACK

Turn4:

    addi    $a0,        $zero,      270

    jal     ROTATE

    jal     GO

Sleep4:

    addi    $v0,        $zero,      32      # Keep running by sleeping in 1000 ms

    li      $a0,        3000

    syscall

Turn5:

    addi    $a0,        $zero,      180

    jal     ROTATE

    jal     GO

Sleep5:

    addi    $v0,        $zero,      32      # Keep running by sleeping in 1000 ms

    li      $a0,        1500

    syscall

Turn6:

    addi    $a0,        $zero,      90

    jal     ROTATE

    jal     GO

Draw3:

    jal     TRACK

    addi    $v0,        $zero,      32      # Keep running by sleeping in 1000 ms

    li      $a0,        3000

    syscall

    jal     UNTRACK

    jal     STOP

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

Kết quả chương trình:

A black and blue squares

Description automatically generated

1. Assignment 2

Code:

            .eqv    KEY\_CODE, 0xFFFF0004        # ASCII code from keyboard, 1 byte

            .eqv    KEY\_READY, 0xFFFF0000       # =1 if has a new keycode ?

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

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

.data

exit\_seq:   .byte   'e', 'x', 'i', 't'          # Sequence to check for

seq\_index:  .word   0                           # Index for sequence checking

.text

    li      $k0,    KEY\_CODE

    li      $k1,    KEY\_READY

    li      $s0,    DISPLAY\_CODE

    li      $s1,    DISPLAY\_READY

    la      $a3,    exit\_seq                    # Load address of exit sequence

    lw      $a2,    seq\_index                   # Load current index of sequence

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

    sw      $t0,    0($s0)                      # Show key regardless of match

    lb      $t3,    0($a3)                      # Load expected char from exit\_seq

    bne     $t0,    $t3,            reset\_seq   # If not matching, reset sequence

    addi    $a3,    $a3,            1           # Move to next character in sequence

    addi    $a2,    $a2,            1           # Increment sequence index

    li      $t4,    4

    beq     $a2,    $t4,            exit        # If index is 4, exit

    j       loop

    nop

reset\_seq:

    la      $a3,    exit\_seq                    # Reset to start of sequence

    sw      $zero,  seq\_index                   # Reset index to 0

    j       loop

    nop

exit:

    li      $v0,    10                          # Exit syscall

    syscall