Thực hành kiến trúc máy tính – Tuần 10

Trần Khánh Quỳnh – 20225762

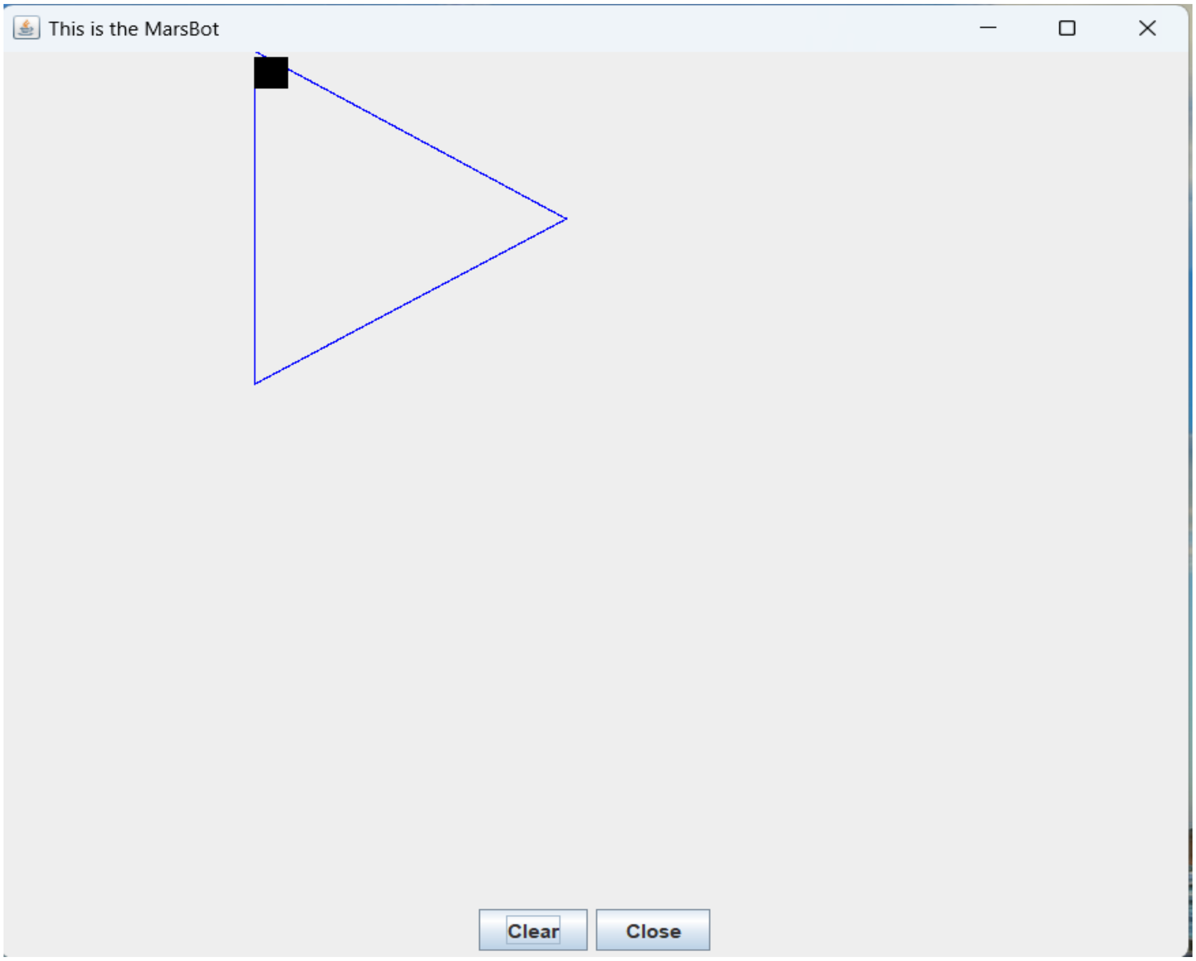
1. Assignment 3

```
.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:
        addi    $a0, $zero, 90  # Marsbot rotates 90* and start
running:
        jal     ROTATE
        nop
        jal     GO
        nop
sleep1: addi    $v0,$zero,32    # Keep running by sleeping in 1000 ms
        li      $a0,7000
        syscall
        li $t0, 2

draw:  beq $t0, $zero, end_main
        jal     UNTRACK         # keep old track
        nop
        jal     TRACK           # and draw new track line
        nop
```

```
goDOWN: addi    $a0, $zero, 120 # Marsbot rotates 120*
        jal     ROTATE
        nop

sleep2: addi    $v0,$zero,32    # Keep running by sleeping in 10000 ms
        li      $a0,10000
        syscall

        jal     UNTRACK         # keep old track
        nop
        jal     TRACK           # and draw new track line
        nop

goLEFT: addi  $a0, $zero, 240   # Marsbot rotates 240*
        jal   ROTATE
        nop

sleep3: addi    $v0,$zero,32    # Keep running by sleeping in 10000 ms
        li      $a0,10000
        syscall

        jal     UNTRACK         # keep old track
        nop
        jal     TRACK           # and draw new track line
        nop

goASKEW:addi  $a0, $zero, 0  # Marsbot rotates 0*
        jal   ROTATE
        nop

sleep4: addi    $v0,$zero,32    # Keep running by sleeping in 10000 ms
        li      $a0,10000
        syscall
        addi $t0, $t0, -1
        j draw

end_main:
      jal STOP
      nop
      j end



#-----------------------------------------------------------
# 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
end:
```

-Kết quả chạy:

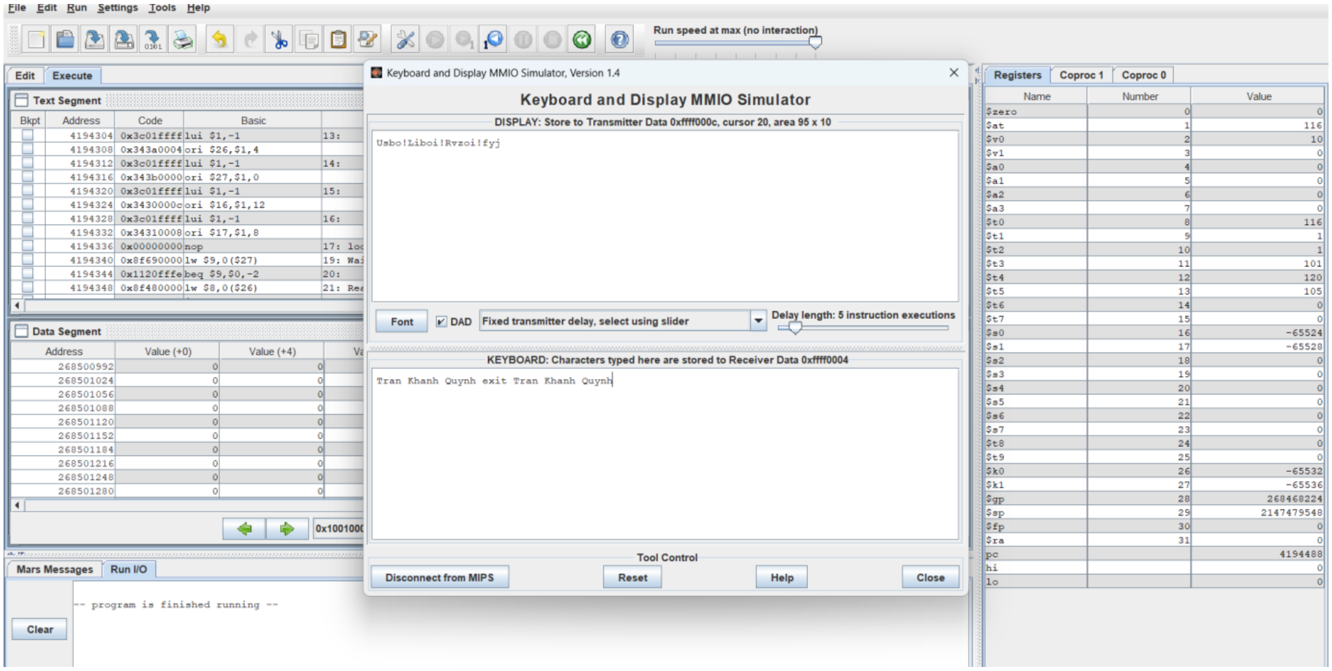


2. Assignment 4

-Mã nguồn:

```
.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
.eqv e_Char   0x65
.eqv x_Char   0x78
.eqv i_Char   0x69
.eqv t_Char   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
             beq $t1, $zero, WaitForKey        # if $t1 == 0 then Polling
ReadKey:     lw $t0, 0($k0)                    # $t0 = [$k0] = KEY_CODE
             j checkE
WaitForDis:  lw $t2, 0($s1)                    # $t2 = [$s1] = DISPLAY_READY
             beq $t2, $zero, WaitForDis        # if $t2 == 0 then Polling
Encrypt:     addi $t0, $t0, 1                  # change input key
ShowKey:     sw $t0, 0($s0)                    # show key
             nop
             j loop
checkE:      beq $t3, e_Char, checkX           # check if exist e in queue, checkX
             bne $t0, e_Char, WaitForDis       # if current char is not e, continue
             add $t3, $t0, $zero       # save 'e' to $t3
             j WaitForDis
checkX:      beq $t4, x_Char, checkI           # check if exist x in queue, checkI
             bne $t0, x_Char, reset            # if current char is not x, reset then
continue
             add $t4, $t0, $zero       # save 'x' to $t4
             j WaitForDis
checkI:              beq $t5, i_Char, checkT           # check if exist i in queue, checkT
             bne $t0, i_Char, reset            # if current char is not i, reset then
continue
             add $t5, $t0, $zero       # save 'i' to $t5
             j WaitForDis
checkT:              beq $t0, t_Char, terminate # check if meet t, terminate (exit word
complete)
             j reset                           # if current char is not t, reset then
continue
reset:       li $t3, 0                 # set 'e' to unspecified
             li $t4, 0                 # set 'x' to unspecified
             li $t5, 0                 # set 'i' to unspecified
             j WaitForDis
terminate:   li $v0, 10
             syscall
```

-Kết quả chạy:



Bkpt	Address	Code	Basic	
□	4194304	0x3c01ffff	lui $1,-1	13:
□	4194308	0x343a0004	ori $26,$1,4	
□	4194312	0x3c01ffff	lui $1,-1	14:
□	4194316	0x343b0000	ori $27,$1,0	
□	4194320	0x3c01ffff	lui $1,-1	15:
□	4194324	0x3430000c	ori $16,$1,12	
□	4194328	0x3c01ffff	lui $1,-1	16:
□	4194332	0x34310008	ori $17,$1,8	
□	4194336	0x00000000	nop	17: loc
□	4194340	0x8f690000	lw $9,0($27)	19: Wai
□	4194344	0x1120fffe	beq $9,$0,-2	20:
□	4194348	0x8f480000	lw $8,0($26)	21: Rea

Address	Value (+0)	Value (+4)
268500992	0	0
268501024	0	0
268501056	0	0
268501088	0	0
268501120	0	0
268501152	0	0
268501184	0	0
268501216	0	0
268501248	0	0
268501280	0	0

Keyboard and Display MMIO Simulator

DISPLAY: Store to Transmitter Data 0xffff000c, cursor 20, area 95 x 10

```
Usbo!Liboi!Rvzoi!fyj
```

☑ DAD Fixed transmitter delay, select using slider

Delay length: 5 instruction executions

KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004

```
Tran Khanh Quynh exit Tran Khanh Quynh
```

```
-- program is finished running --
```

Name	Number	Value
$zero	0	0
$at	1	116
$v0	2	10
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	116
$t1	9	1
$t2	10	1
$t3	11	101
$t4	12	120
$t5	13	105
$t6	14	0
$t7	15	0
$s0	16	-65524
$s1	17	-65528
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	-65532
$k1	27	-65536
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194488
hi		0
lo		0

-Giải thích:

+Kiểm tra xem có phải là ‘e’ không, nếu là ‘e’ thì kiểm tra tiếp, lưu ‘e’ vào $t3; còn nếu không thì quay lại WaitForDis.

+Kiểm tra xem có phải là ‘x’ không, nếu là ‘x’ thì kiểm tra tiếp, lưu ‘x’ vào $t4, nếu không thì xuống reset để đưa $t3 về 0 rồi quay lại WaitForDis.

+Kiểm tra xem có phải là ‘i’ không, nếu là ‘i’ thì kiểm tra tiếp, lưu ‘i’ vào $t5; còn nếu không thì xuống reset để thiết lập lại $t3, $t4 rồi quay lại WaitForDis

+Kiểm tra xem có phải là ‘t’ không, nếu là ‘t’ thì kết thúc chương trình; còn nếu không thì xuống reset thiết lập lại $t3, $t4, $t5 rồi quay lại WaitForDis