Thực hành Kiến trúc máy tính tuần 10.2

Họ tên: Đỗ Hoàng Minh Hiếu

MSSV: 20225837

BÀI 1:

1. Vẽ hình tam giác đều

Code:

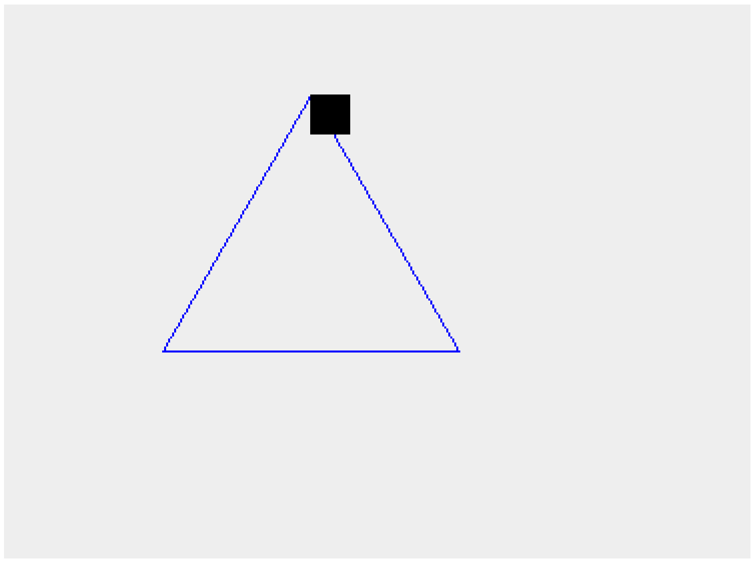
```
.eqv HEADING 0xffff8010 # Integer: An angle between 0 and 359
.eqv MOVING 0xffff8050 # Boolean: whether or not to move
.eqv LEAVETRACK 0xffff8020 # Boolean (0 or non-0):
 # whether or not to leave a track
.eqv WHEREX 0xffff8030 # Integer: Current x-location
.eqv WHEREY 0xffff8040 # Integer: Current y-location
.text
main:
	jal UNTRACK # draw track line
	nop
	li $a0, 110
	jal ROTATE
	nop
	jal GO
	nop
	li $a0, 17000
	jal SLEEP
	nop
	jal TRACK
	nop
Cheo1:
	li $a0, 150
```

```
        jal ROTATE
        nop
        li $a0, 7000
        jal SLEEP
        nop
        jal UNTRACK
        nop
        jal TRACK
        nop

Ngang:
        li $a0, 270
        nop
        jal ROTATE
        nop
        li $a0, 7000
        jal SLEEP
        nop
        jal UNTRACK
        nop
        jal TRACK
        nop

Cheo2:
        li $a0, 30
        jal ROTATE
        nop
        li $a0, 7000
```

```
            jal SLEEP
            nop
            jal UNTRACK
            nop
            jal STOP
            nop
li $v0, 10
syscall
end_main:

# Function from here
SLEEP:
            li $v0, 32
            syscall

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: li $at, MOVING # change MOVING port to 0
 sb $zero, 0($at) # to stop
nop
 jr $ra
 nop
```

```
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:li $at, LEAVETRACK # change LEAVETRACK port to 0
 sb $zero, 0($at) # to stop drawing tail
 nop
 jr $ra
 nop

ROTATE: li $at, HEADING # change HEADING port
 sw $a0, 0($at) # to rotate robot
 nop
 jr $ra
 nop
```

Kết quả:



2. Vẽ hình vuông

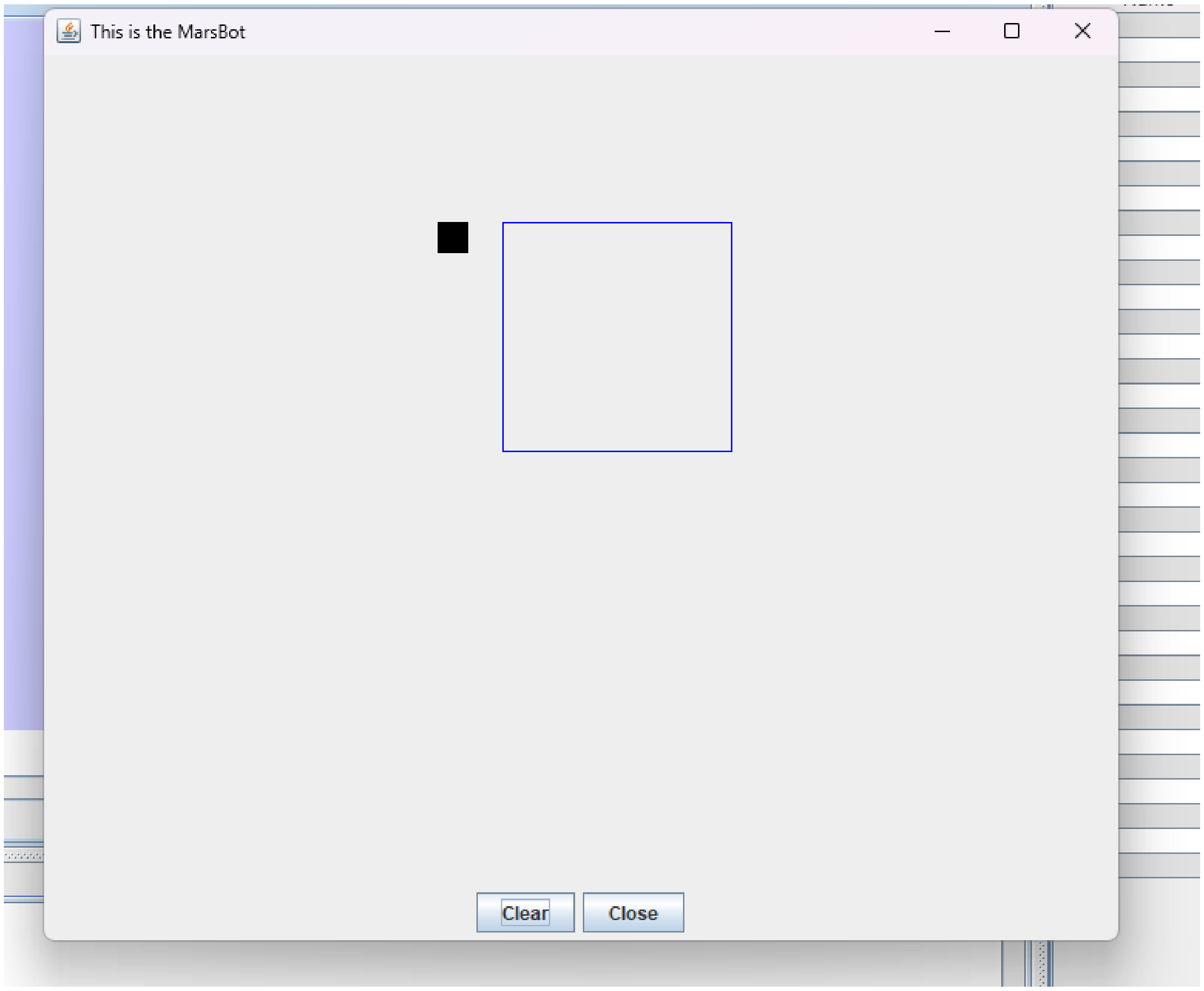
Code:

```
.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):
 # whether or not to leave a track
.eqv WHEREX 0xffff8030 # Integer: Current x-location
.eqv WHEREY 0xffff8040 # Integer: Current y-location
.text
main:
        jal UNTRACK # draw track line
        nop
        li $a0, 110
        jal ROTATE
        nop
        jal GO
        nop
        li $a0, 15000
        jal SLEEP
        nop
        jal TRACK
        nop
Doc1:
        li $a0, 180
        jal ROTATE
        nop
        li $a0, 7000
        jal SLEEP
        nop
        jal UNTRACK
        nop
        jal TRACK
        nop

Ngang1:
        li $a0, 90
        nop
        jal ROTATE
        nop
        li $a0, 7000
        jal SLEEP
        nop
        jal UNTRACK
        nop
        jal TRACK
        nop

Doc2:
        li $a0, 0
        jal ROTATE
        nop
        li $a0, 7000
        jal SLEEP
        nop
        jal UNTRACK
        nop
        jal TRACK
        nop

Ngang2:
        li $a0, 270
        nop
        jal ROTATE
        nop
        li $a0, 7000
        jal SLEEP
        nop
        jal UNTRACK
        nop
        li $a0, 270
        nop
        jal ROTATE
        nop
```

```
            li $a0, 2000
            jal SLEEP
            nop
            jal STOP
            nop
li $v0, 10
syscall
end_main:

# Function from here
SLEEP:
            li $v0, 32
            syscall

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: li $at, MOVING # change MOVING port to 0
 sb $zero, 0($at) # to stop
nop
 jr $ra
 nop

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:li $at, LEAVETRACK # change LEAVETRACK port to 0
 sb $zero, 0($at) # to stop drawing tail
 nop
 jr $ra
 nop

ROTATE: li $at, HEADING # change HEADING port
 sw $a0, 0($at) # to rotate robot
 nop
 jr $ra
 nop
```

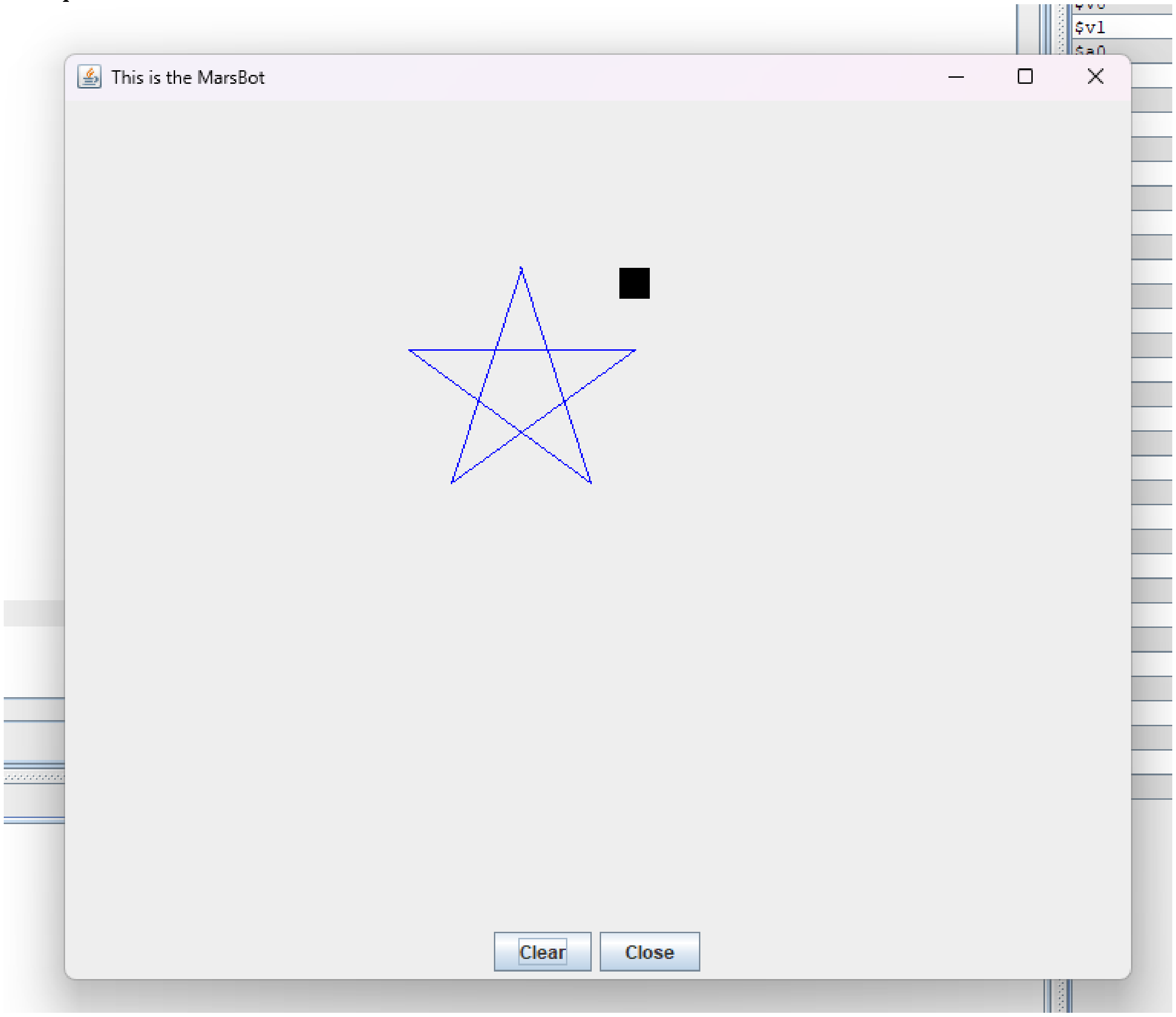
Kết quả:



3. Vẽ hình ngôi sao 5 cánh

```
.eqv HEADING 0xffff8010
.eqv MOVING 0xffff8050
.eqv LEAVETRACK 0xffff8020
.eqv WHEREX 0xffff8030
.eqv WHEREY 0xffff8040
.text
main:
 addi $a0, $zero, 110
 jal ROTATE
 jal GO


 addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
 li $a0,15000
 syscall


sleep1:
 addi $a0, $zero, 162
 jal ROTATE
 jal GO
 jal UNTRACK # keep old track
 jal TRACK # and draw new track line
 addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
 li $a0,7000
 syscall
sleep2:
 addi $a0, $zero, 306
 jal ROTATE
 jal GO
 jal UNTRACK # keep old track
 jal TRACK # and draw new track line
 addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
 li $a0,7000
 syscall
sleep3:
 addi $a0, $zero, 90
 jal ROTATE
 jal GO
 jal UNTRACK # keep old track
 jal TRACK # and draw new track line
 addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
 li $a0,7000
 syscall
sleep4:
 addi $a0, $zero, 234
 jal ROTATE
 jal GO
 jal UNTRACK # keep old track
 jal TRACK # and draw new track line
 addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
 li $a0,7000
 syscall
sleep5:
 addi $a0, $zero, 18
 jal ROTATE
 jal GO
 jal UNTRACK # keep old track
 jal TRACK # and draw new track line
 addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
 li $a0,7000
 syscall
end_main:
 jal UNTRACK # keep old track
 addi $a0, $zero, 90
 jal ROTATE
 jal GO
 addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
 li $a0,3000
 syscall
 jal STOP
 li $v0, 10
 syscall

GO:
 li $at, MOVING # change MOVING port
 addi $k0, $zero,1 # to logic 1,
 sb $k0, 0($at) # to start running
 jr $ra

ROTATE:
 li $at, HEADING # change HEADING port
 sw $a0, 0($at) # to rotate robot
 jr $ra
STOP:
 li $at, MOVING # change MOVING port to 0
 sb $zero, 0($at) # to stop
 jr $ra

TRACK:
 li $at, LEAVETRACK # change LEAVETRACK port
 addi $k0, $zero,1 # to logic 1,
 sb $k0, 0($at) # to start tracking
 jr $ra
UNTRACK:
 li $at, LEAVETRACK # change LEAVETRACK port to 0
 sb $zero, 0($at) # to stop drawing tail
 jr $ra
```

Kết quả:



BÀI 2:

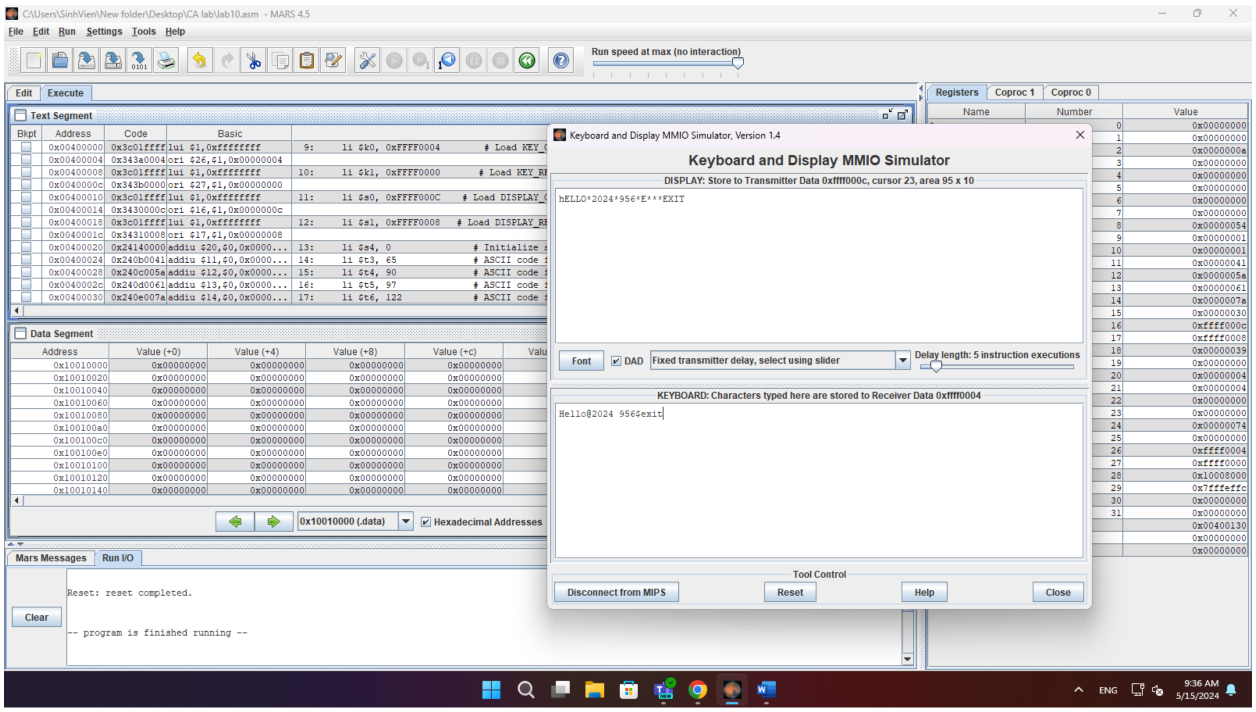
Code:

```
.eqv KEY_CODE 0xFFFF0004      # Address to read the ASCII code from the keyboard, 1 byte
.eqv KEY_READY 0xFFFF0000     # Address to check if a new keycode is available
.eqv DISPLAY_CODE 0xFFFF000C  # Address to write the ASCII code to the display, 1 byte
.eqv DISPLAY_READY 0xFFFF0008 # Address to check if the display is ready

.text
```

```
main:
   li $k0, KEY_CODE        # Load KEY_CODE address into $k0
   li $k1, KEY_READY       # Load KEY_READY address into $k1
   li $s0, DISPLAY_CODE    # Load DISPLAY_CODE address into $s0
   li $s1, DISPLAY_READY   # Load DISPLAY_READY address into $s1
   li $s4, 0               # Initialize state to 0
   li $t3, 65              # ASCII code for 'A'
   li $t4, 90              # ASCII code for 'Z'
   li $t5, 97              # ASCII code for 'a'
   li $t6, 122             # ASCII code for 'z'
   li $t7, 48              # ASCII code for '0'
   li $s2, 57              # ASCII code for '9'

loop:
   nop

WaitForKey:
   lw $t1, 0($k1)          # Load KEY_READY status
   nop
   beq $t1, $zero, WaitForKey # Poll until a key is ready
   nop

ReadKey:
   lw $t0, 0($k0)          # Load the key code
   nop

WaitForDis:
   lw $t2, 0($s1)          # Load DISPLAY_READY status
   nop
   beq $t2, $zero, WaitForDis # Poll until the display is ready
   nop

CheckKey:
   li $s5, 1
   beq $s4, $zero, CheckE  # If state is 0, check for 'e'
   beq $s4, $s5, CheckX    # If state is 1, check for 'x'
   li $s5, 2
   beq $s4, $s5, CheckI    # If state is 2, check for 'i'
   li $s5, 3
   beq $s4, $s5, CheckT    # If state is 3, check for 't'
   j Encrypt               # Otherwise, go to Encrypt

CheckE:
   li $t8, 101             # ASCII code for 'e'
   beq $t0, $t8, agree     # If key is 'e', go to agree
   j Minus                 # Otherwise, reset state

CheckX:
   li $t8, 120             # ASCII code for 'x'
   beq $t0, $t8, agree     # If key is 'x', go to agree
   j Minus                 # Otherwise, reset state

CheckI:
   li $t8, 105             # ASCII code for 'i'
   beq $t0, $t8, agree     # If key is 'i', go to agree
   j Minus                 # Otherwise, reset state

CheckT:
   li $t8, 116             # ASCII code for 't'
   beq $t0, $t8, agree  # If key is 't', terminate
   j Minus                 # Otherwise, reset state

Minus:
   li $s4, 0              # Reset state to 0

Encrypt:
   # Check if uppercase letter
   blt $t0, $t3, not_uppercase
   bgt $t0, $t4, not_uppercase
   # Convert to lowercase
   addi $t0, $t0, 32
   j continue

not_uppercase:
   # Check if lowercase letter
   blt $t0, $t5, not_number
   bgt $t0, $t6, not_number
   # Convert to uppercase
   addi $t0, $t0, -32
   j continue

not_number:
   # Check if number
```

```
   blt $t0, $t7, else
   bgt $t0, $s2, else
   j continue

else:
   li $t0, 42              # Replace with '*'
   j continue

continue:
   # Display the key code
   sw $t0, 0($s0)         # Show key
   nop
          li $s5, 4
          beq $s4, $s5, end_main
   j loop

agree:
   addi $s4, $s4, 1        # Increment state
   j Encrypt
   nop

end_main:
   li $v0, 10              # Exit program
   syscall
```

Kết quả:


C:\Users\SinhVien\New folder\Desktop\CA lab\lab10.asm - MARS 4.5

Text Segment

Bkpt	Address	Code	Basic	
	0x00400000	0x3c01ffff	lui $1,0xffffffff	9: li $k0, 0xFFFF0004 # Load KEY_C
	0x00400004	0x343a0004	ori $26,$1,0x00000004	
	0x00400008	0x3c01ffff	lui $1,0xffffffff	10: li $k1, 0xFFFF0000 # Load KEY_RE
	0x0040000c	0x343b0000	ori $27,$1,0x00000000	
	0x00400010	0x3c01ffff	lui $1,0xffffffff	11: li $s0, 0xFFFF000C # Load DISPLAY_C
	0x00400014	0x3430000c	ori $16,$1,0x0000000c	
	0x00400018	0x3c01ffff	lui $1,0xffffffff	12: li $s1, 0xFFFF0008 # Load DISPLAY_RE
	0x0040001c	0x34310008	ori $17,$1,0x00000008	
	0x00400020	0x24140000	addiu $20,$0,0x0000...	13: li $s4, 0 # Initialize s
	0x00400024	0x240b0041	addiu $11,$0,0x0000...	14: li $t3, 65 # ASCII code f
	0x00400028	0x240c005a	addiu $12,$0,0x0000...	15: li $t4, 90 # ASCII code f
	0x0040002c	0x240d0061	addiu $13,$0,0x0000...	16: li $t5, 97 # ASCII code f
	0x00400030	0x240e007a	addiu $14,$0,0x0000...	17: li $t6, 122 # ASCII code f

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses

Keyboard and Display MMIO Simulator, Version 1.4

Keyboard and Display MMIO Simulator

DISPLAY: Store to Transmitter Data 0xffff000c, cursor 23, area 95 x 10

```
hELLO*2024*956*E***EXIT
```

☑ DAD Fixed transmitter delay, select using slider — Delay length: 5 instruction executions

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

```
Hello@2024 956$exit
```

Number	Value
0	0x00000000
1	0x00000000
2	0x0000000a
3	0x00000000
4	0x00000000
5	0x00000000
6	0x00000000
7	0x00000000
8	0x00000054
9	0x00000001
10	0x00000001
11	0x00000041
12	0x0000005a
13	0x00000061
14	0x0000007a
15	0x00000030
16	0xffff000c
17	0xffff0008
18	0x00000039
19	0x00000000
20	0x00000004
21	0x00000004
22	0x00000000
23	0x00000000
24	0x00000074
25	0x00000000
26	0xffff0004
27	0xffff0000
28	0x10008000
29	0x7fffeffc
30	0x00000000
31	0x00000000
	0x00400130
	0x00000000
	0x00000000

Mars Messages / Run I/O

```
Reset: reset completed.

-- program is finished running --
```

File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Registers Coproc 1 Coproc 0
Font
Tool Control
Disconnect from MIPS
Reset
Help
Close
Clear
9:36 AM
5/15/2024

Bài 3:

Code:

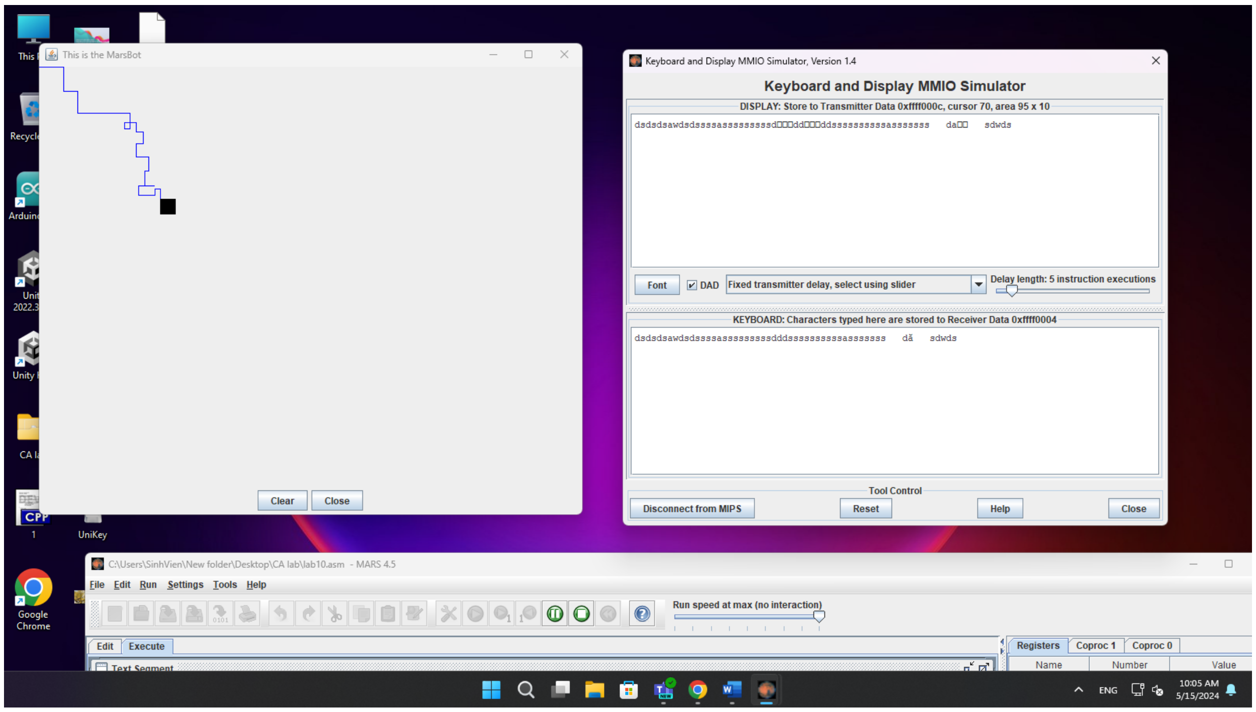
```
.eqv KEY_CODE 0xFFFF0004      # Address to read the ASCII code from the keyboard, 1 byte
.eqv KEY_READY 0xFFFF0000     # Address to check if a new keycode is available
.eqv DISPLAY_CODE 0xFFFF000C  # Address to write the ASCII code to the display, 1 byte
.eqv DISPLAY_READY 0xFFFF0008 # Address to check if the display is ready
.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): 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:
        li $t8, KEY_CODE

        li $t9, KEY_READY

        li $s0, DISPLAY_CODE # chua ky tu can in ra man hinh

        li $s1, DISPLAY_READY

loop: nop

WaitForKey:

        lw $t1, 0($t9) # $t1 = [$k1] = KEY_READY

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

ReadKey:

        lw $t0, 0($t8) # $t0 = [$k0] = KEY_CODE

WaitForDis:

        lw $t2, 0($s1) # $t2 = [$s1] = DISPLAY_READY

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

Encrypt:

        beq $t0, 65, sleepA

        beq $t0, 97, sleepA

        beq $t0, 87, sleepW

        beq $t0, 119, sleepW

        beq $t0, 68, sleepD

        beq $t0, 100, sleepD

        beq $t0, 83, sleepS

        beq $t0, 115, sleepS

        beq $t0, 32, Nghiem

ShowKey:

  # Display the key code

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

  nop

  j loop
```

```
sleepW:
        addi $a0, $zero, 0
        jal ROTATE
        jal GO
        jal UNTRACK # keep old track
        jal TRACK # and draw new track line
        j ShowKey
sleepS:
        addi $a0, $zero, 180
        jal ROTATE
        jal GO
        jal UNTRACK # keep old track
        jal TRACK # and draw new track line
        j ShowKey
sleepD:
        addi $a0, $zero, 90
        jal ROTATE
        jal GO
        jal UNTRACK # keep old track
        jal TRACK # and draw new track line
        j ShowKey
sleepA:
        addi $a0, $zero, 270
        jal ROTATE
        jal GO
        jal UNTRACK # keep old track
        jal TRACK # and draw new track line
        j ShowKey
```

```
Nghiem:
	jal STOP
	j ShowKey
Ditiep:
	jal GO
	j ShowKey
end_main:
  li $v0, 10		# Exit program
  syscall
GO:
	li $at, MOVING # change MOVING port
	li $k0, 1 # to logic 1,
	sb $k0, 0($at) # to start running
	jr $ra

ROTATE:
	li $at, HEADING # change HEADING port
	sw $a0, 0($at) # to rotate robot
	jr $ra
STOP:
	li $at, MOVING # change MOVING port to 0
	sb $zero, 0($at) # to stop
	jr $ra

TRACK:
	li $at, LEAVETRACK # change LEAVETRACK port
	li $k0, 1 # to logic 1,
	sb $k0, 0($at) # to start tracking
```

```
        jr $ra
UNTRACK:
        li $at, LEAVETRACK # change LEAVETRACK port to 0
        sb $zero, 0($at) # to stop drawing tail
        jr $ra
```

Kết quả:



This is the MarsBot
Clear
Close
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
Font
DAD
Tool Control
Disconnect from MIPS
Reset
Help
Close
C:\Users\SinhVien\New folder\Desktop\CA lab\lab10.asm - MARS 4.5
File
Edit
Run
Settings
Tools
Help
Run speed at max (no interaction)
Edit
Execute
Text Segment
Registers
Coproc 1
Coproc 0
Name
Number
Value

DISPLAY: Store to Transmitter Data 0xffff000c, cursor 70, area 95 x 10

```
dsdsdsawdsdsssasssssssssd□□□dd□□□ddssssssssssasssssss    da□□    sdwds
```

[✓ DAD] Fixed transmitter delay, select using slider

Delay length: 5 instruction executions

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

```
dsdsdsawdsdsssassssssssssdddssssssssssasssssss    dă    sdwds
```