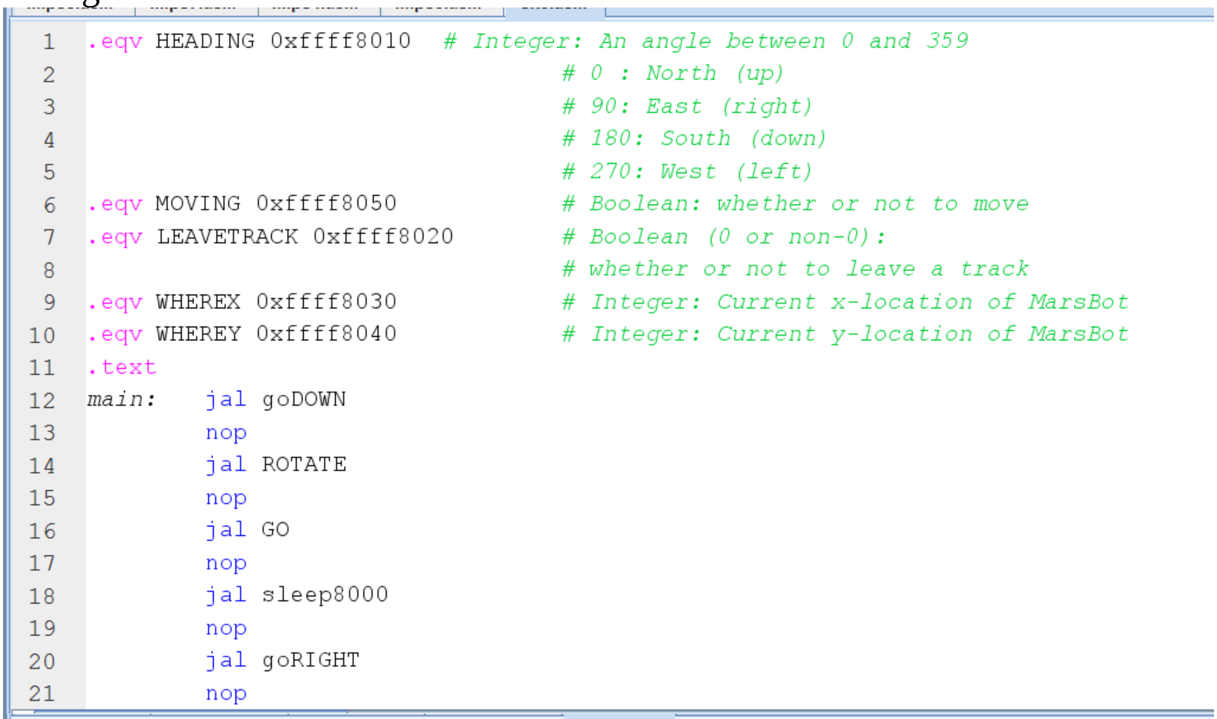
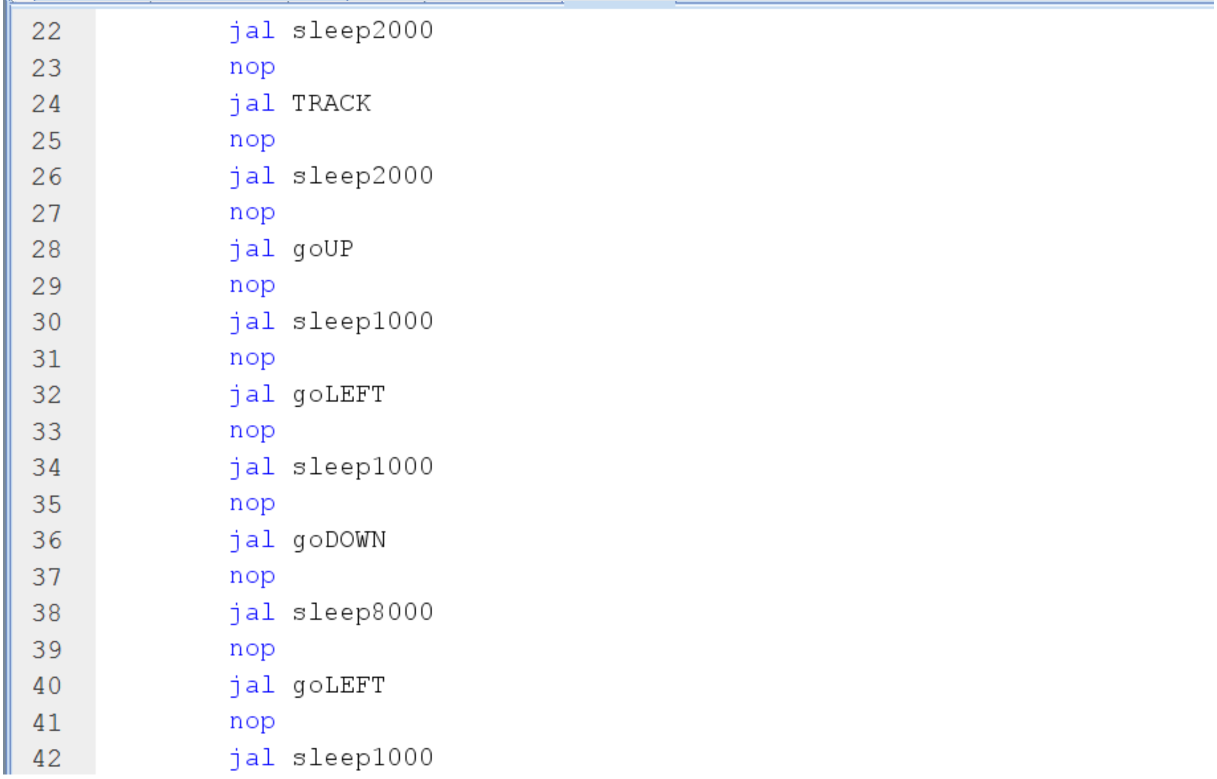
Báo cáo tuần 11

Bài 3:

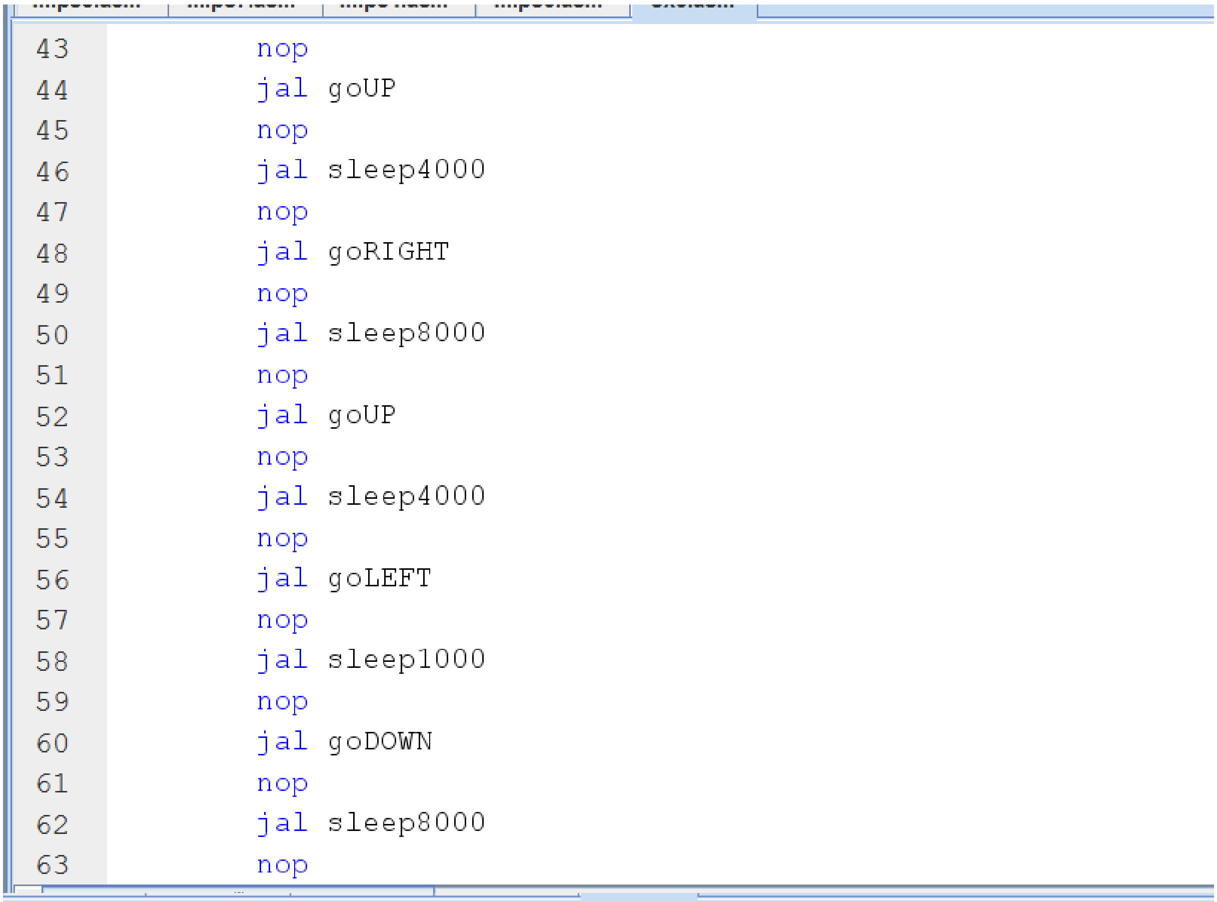
- Mã nguồn:



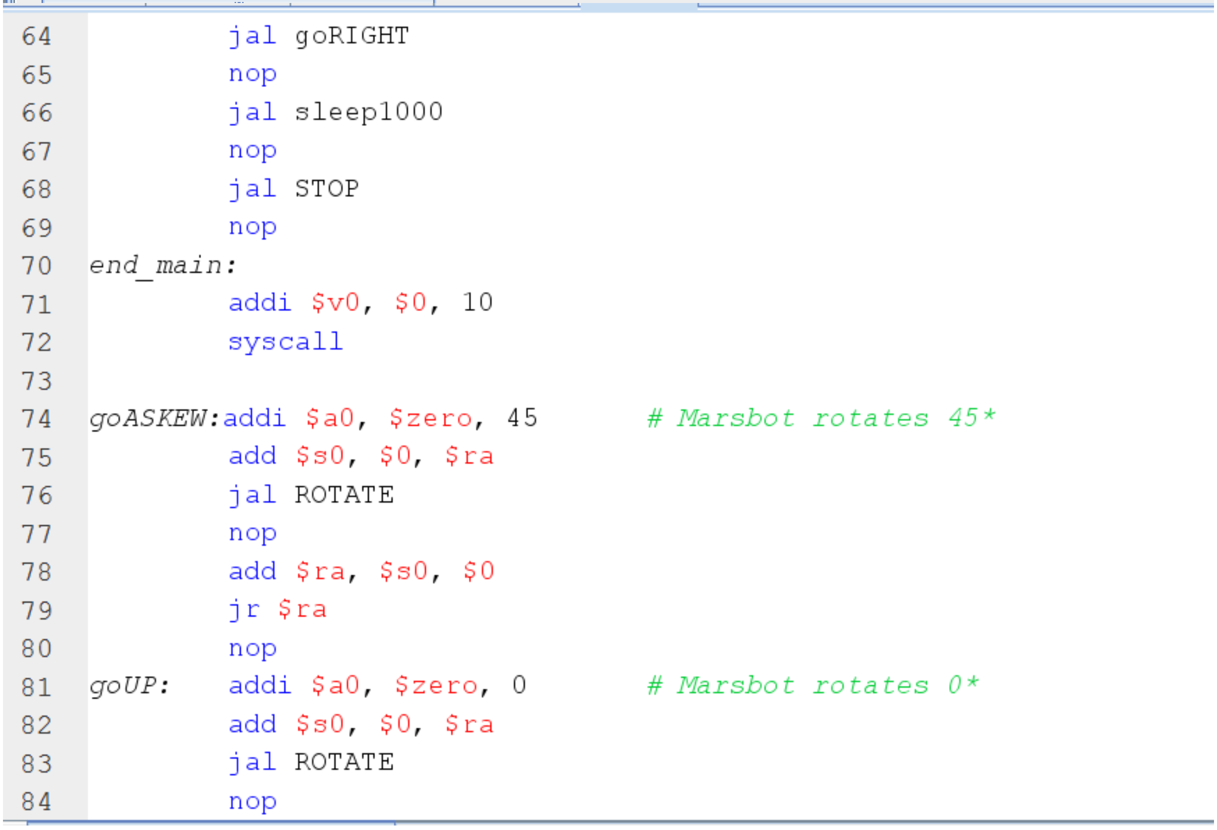
```
.eqv HEADING 0xffff8010  # Integer: An angle between 0 and 359
                                  # 0 : North (up)
                                  # 90: East (right)
                                  # 180: South (down)
                                  # 270: West (left)
.eqv MOVING 0xffff8050            # 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 goDOWN
        nop
        jal ROTATE
        nop
        jal GO
        nop
        jal sleep8000
        nop
        jal goRIGHT
        nop
```



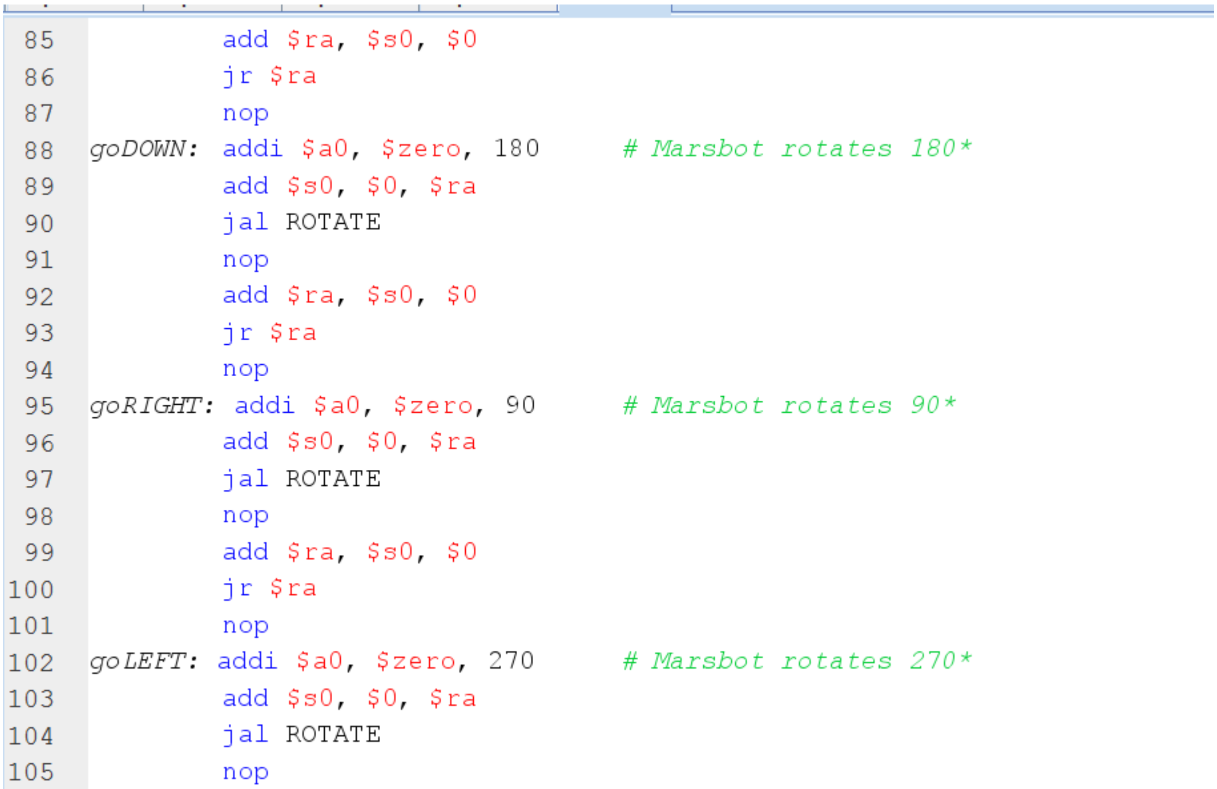
```
        jal sleep2000
        nop
        jal TRACK
        nop
        jal sleep2000
        nop
        jal goUP
        nop
        jal sleep1000
        nop
        jal goLEFT
        nop
        jal sleep1000
        nop
        jal goDOWN
        nop
        jal sleep8000
        nop
        jal goLEFT
        nop
        jal sleep1000
```



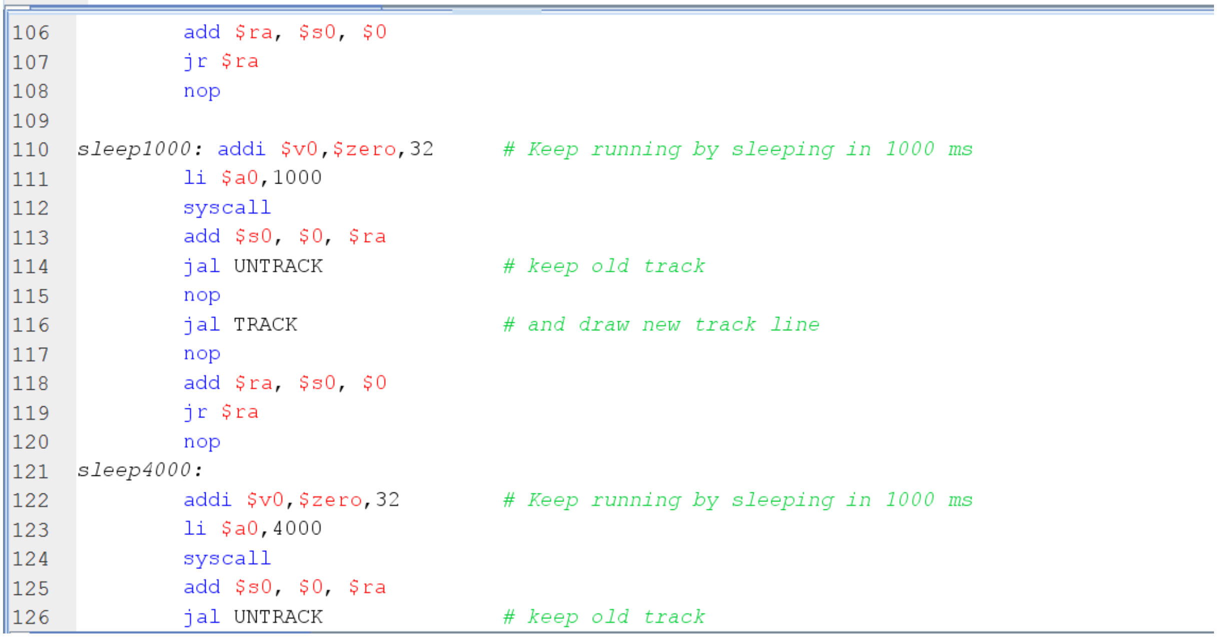
```
        nop
        jal goUP
        nop
        jal sleep4000
        nop
        jal goRIGHT
        nop
        jal sleep8000
        nop
        jal goUP
        nop
        jal sleep4000
        nop
        jal goLEFT
        nop
        jal sleep1000
        nop
        jal goDOWN
        nop
        jal sleep8000
        nop
```



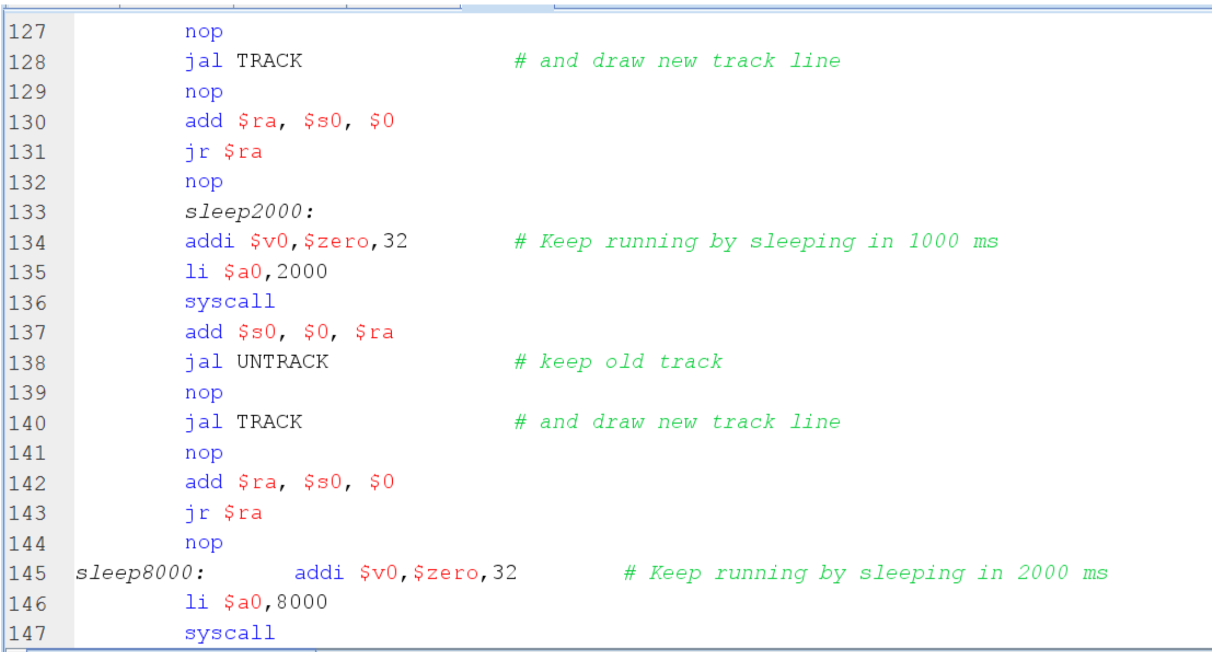
```
        jal goRIGHT
        nop
        jal sleep1000
        nop
        jal STOP
        nop
end_main:
        addi $v0, $0, 10
        syscall

goASKEW:addi $a0, $zero, 45       # Marsbot rotates 45*
        add $s0, $0, $ra
        jal ROTATE
        nop
        add $ra, $s0, $0
        jr $ra
        nop
goUP:   addi $a0, $zero, 0        # Marsbot rotates 0*
        add $s0, $0, $ra
        jal ROTATE
        nop
```



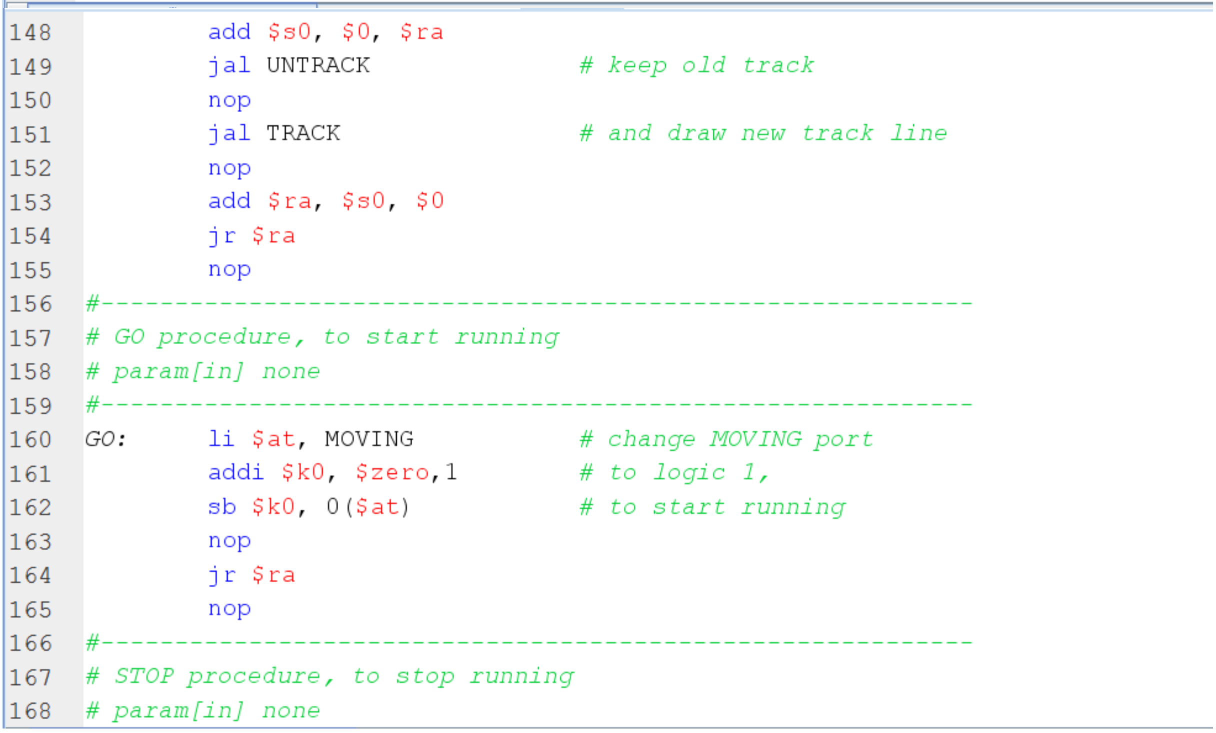
```
          add $ra, $s0, $0
          jr $ra
          nop
goDOWN:   addi $a0, $zero, 180     # Marsbot rotates 180*
          add $s0, $0, $ra
          jal ROTATE
          nop
          add $ra, $s0, $0
          jr $ra
          nop
goRIGHT:  addi $a0, $zero, 90      # Marsbot rotates 90*
          add $s0, $0, $ra
          jal ROTATE
          nop
          add $ra, $s0, $0
          jr $ra
          nop
goLEFT:   addi $a0, $zero, 270     # Marsbot rotates 270*
          add $s0, $0, $ra
          jal ROTATE
          nop
```



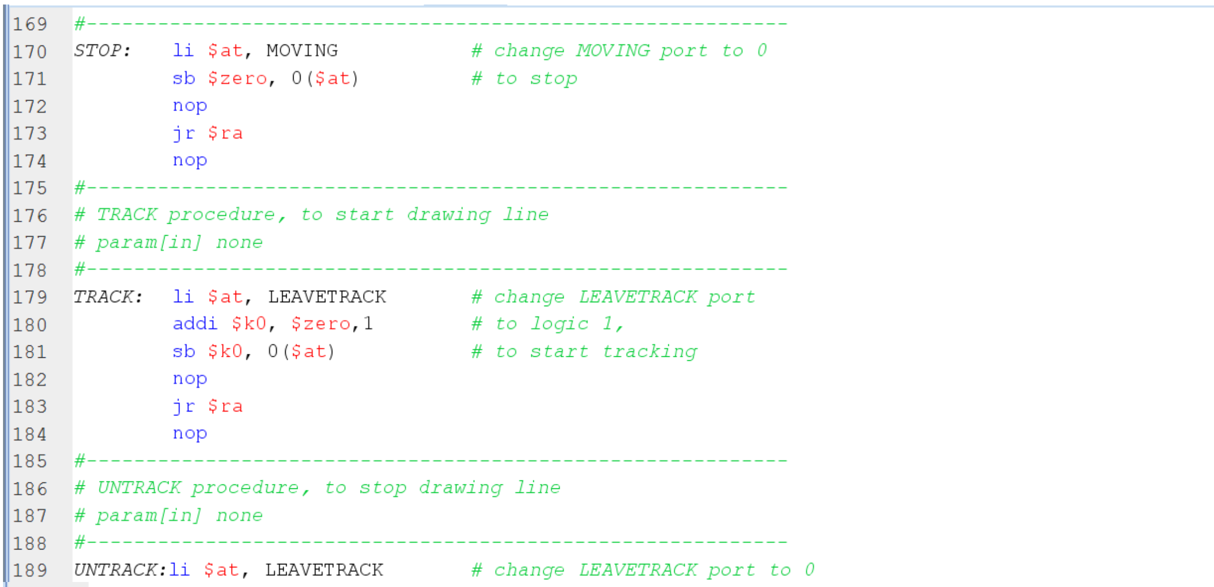
```
          add $ra, $s0, $0
          jr $ra
          nop

sleep1000: addi $v0,$zero,32      # Keep running by sleeping in 1000 ms
          li $a0,1000
          syscall
          add $s0, $0, $ra
          jal UNTRACK             # keep old track
          nop
          jal TRACK               # and draw new track line
          nop
          add $ra, $s0, $0
          jr $ra
          nop
sleep4000:
          addi $v0,$zero,32       # Keep running by sleeping in 1000 ms
          li $a0,4000
          syscall
          add $s0, $0, $ra
          jal UNTRACK             # keep old track
```



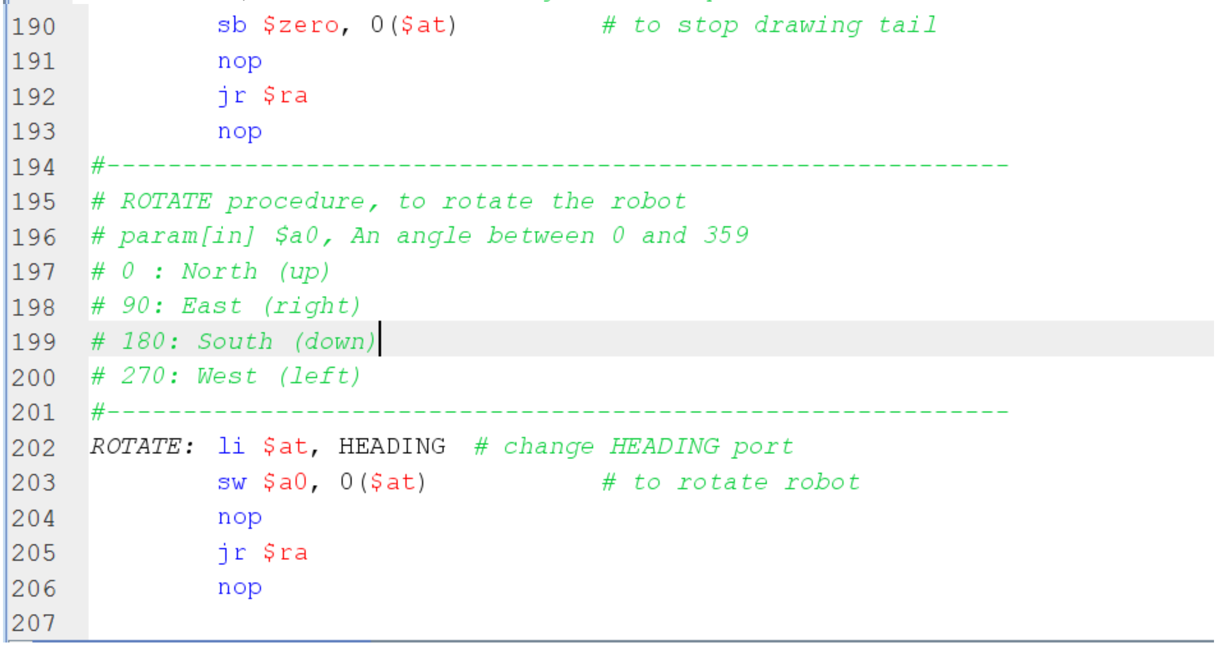
```
		nop
		jal TRACK			# and draw new track line
		nop
		add $ra, $s0, $0
		jr $ra
		nop
		sleep2000:
		addi $v0,$zero,32		# Keep running by sleeping in 1000 ms
		li $a0,2000
		syscall
		add $s0, $0, $ra
		jal UNTRACK			# keep old track
		nop
		jal TRACK			# and draw new track line
		nop
		add $ra, $s0, $0
		jr $ra
		nop
sleep8000:	addi $v0,$zero,32		# Keep running by sleeping in 2000 ms
		li $a0,8000
		syscall
```



```
		add $s0, $0, $ra
		jal UNTRACK			# keep old track
		nop
		jal TRACK			# and draw new track line
		nop
		add $ra, $s0, $0
		jr $ra
		nop
#---------------------------------------------------------------
# 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
```

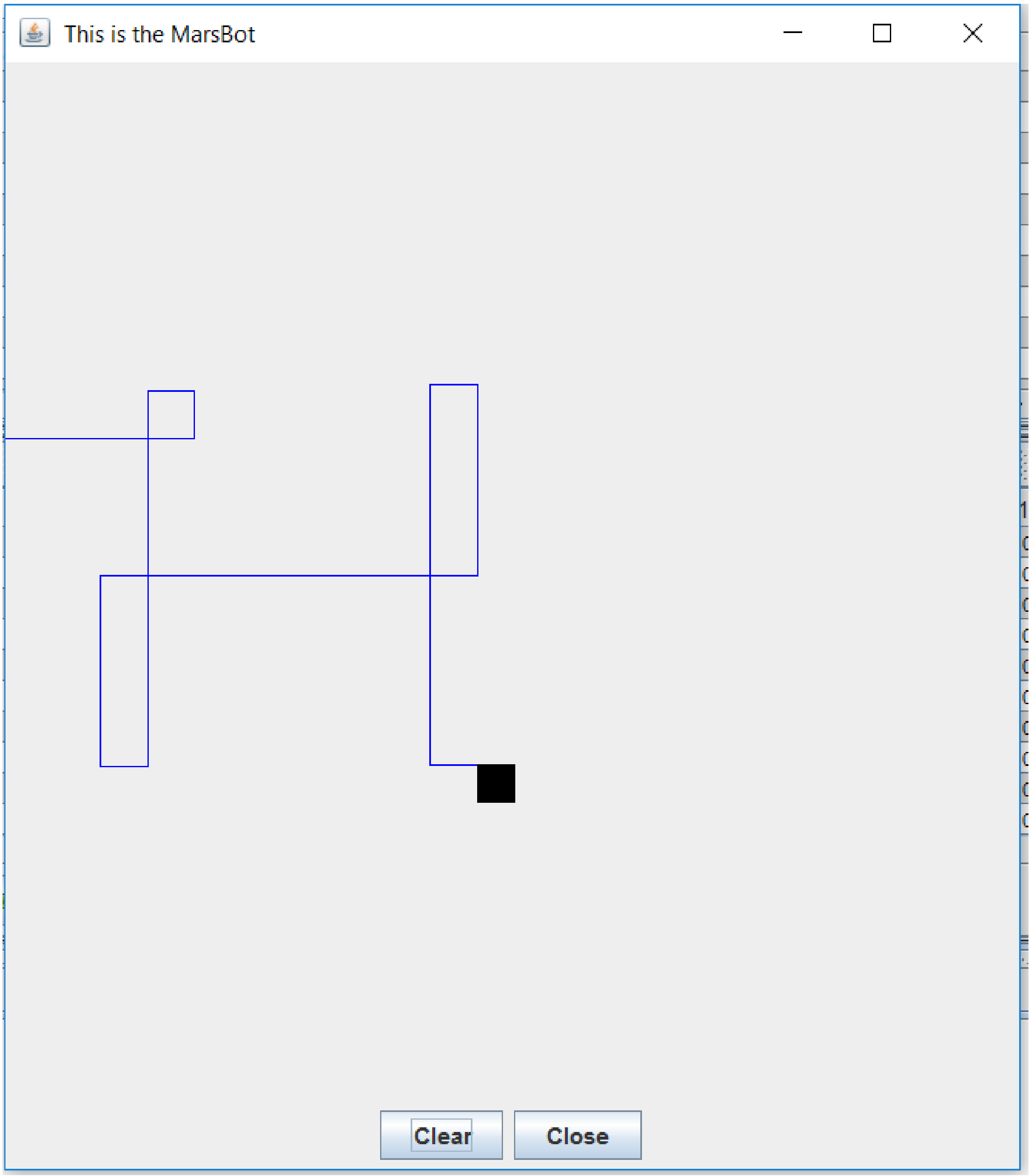


```
#-------------------------------------------------------------
STOP:    li $at, MOVING             # change MOVING port to 0
         sb $zero, 0($at)           # to stop
         nop
         jr $ra
         nop
#-------------------------------------------------------------
# TRACK procedure, to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



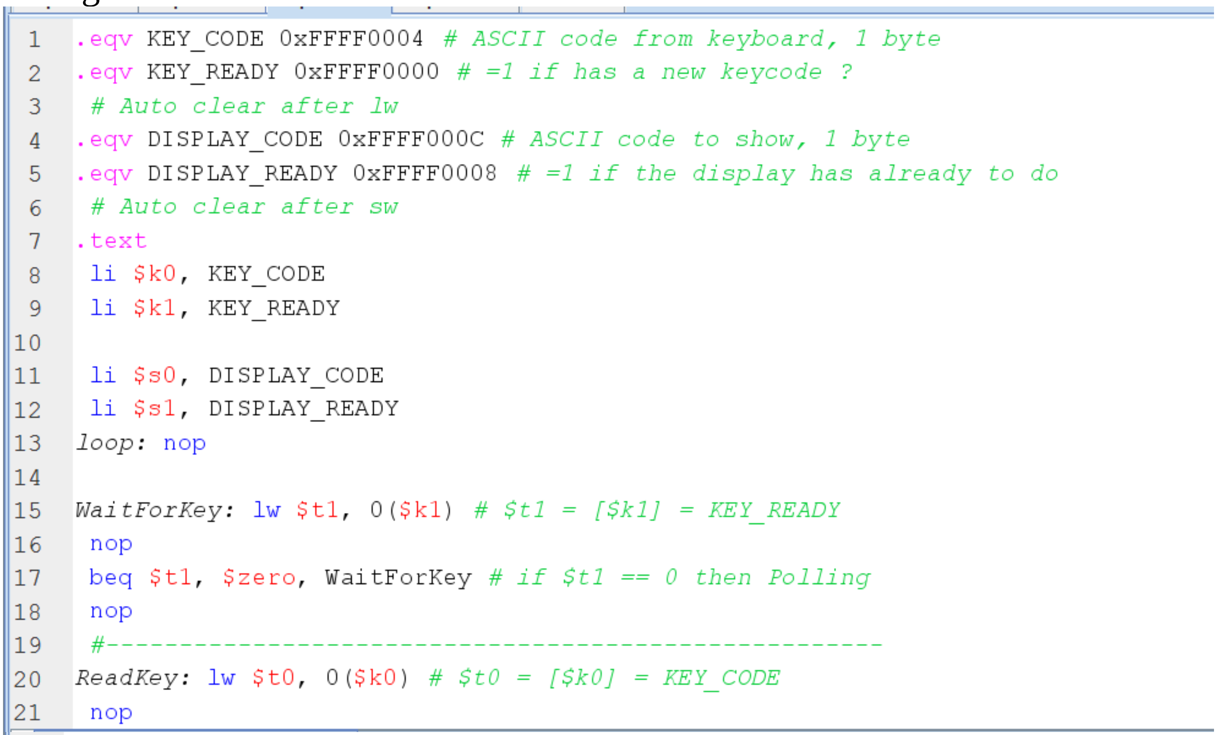
```
         sb $zero, 0($at)           # to stop drawing tail
         nop
         jr $ra
         nop
#-------------------------------------------------------------
# ROTATE procedure, to rotate the robot
# param[in] $a0, An angle between 0 and 359
# 0 : North (up)
# 90: East (right)
# 180: South (down)
# 270: West (left)
#-------------------------------------------------------------
ROTATE:  li $at, HEADING  # change HEADING port
         sw $a0, 0($at)             # to rotate robot
         nop
         jr $ra
         nop
```

Vẽ chữ H:

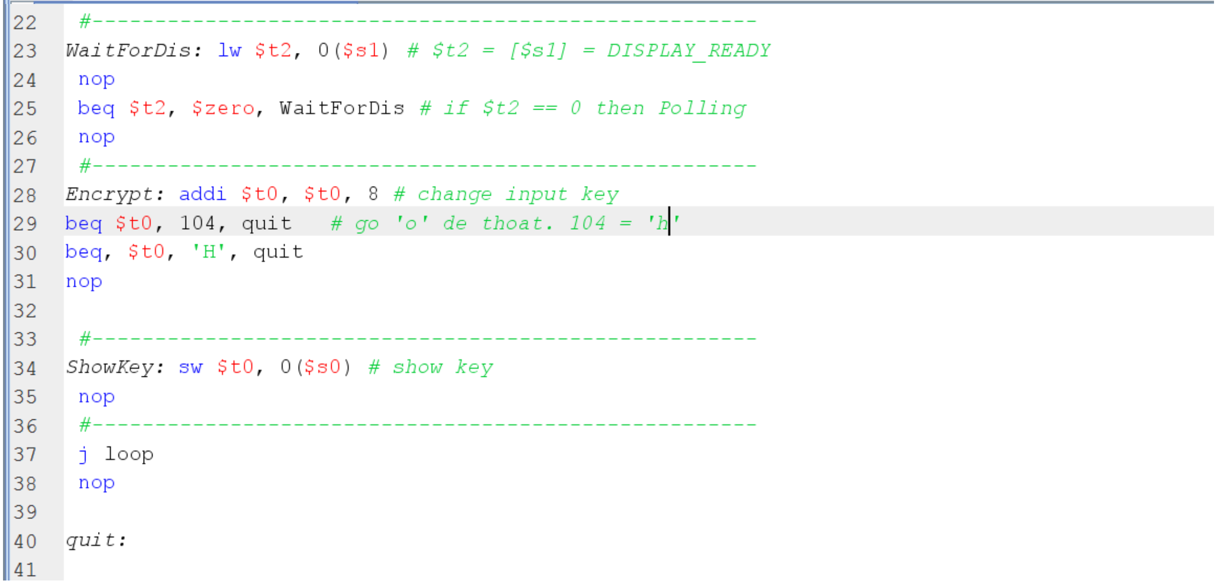


Bài 4:

- Mã nguồn:

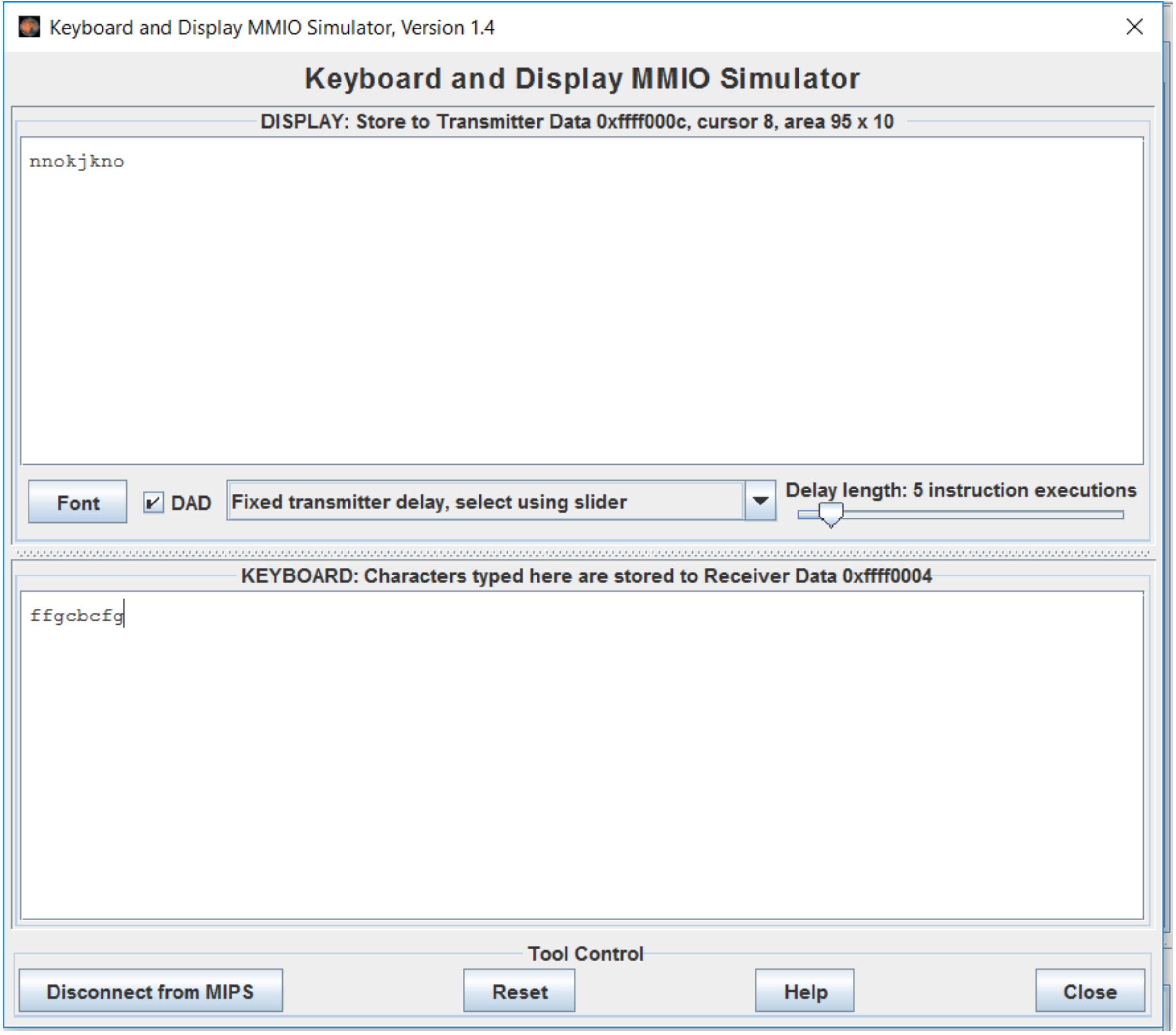


```
.eqv KEY_CODE 0xFFFF0004 # ASCII code from keyboard, 1 byte
.eqv KEY_READY 0xFFFF0000 # =1 if has a new keycode ?
 # Auto clear after lw
.eqv DISPLAY_CODE 0xFFFF000C # ASCII code to show, 1 byte
.eqv DISPLAY_READY 0xFFFF0008 # =1 if the display has already to do
 # Auto clear after sw
.text
 li $k0, 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
```



```
 #-----------------------------------------------------
WaitForDis: lw $t2, 0($s1) # $t2 = [$s1] = DISPLAY_READY
 nop
 beq $t2, $zero, WaitForDis # if $t2 == 0 then Polling
 nop
 #-----------------------------------------------------
Encrypt: addi $t0, $t0, 8 # change input key
beq $t0, 104, quit   # go 'o' de thoat. 104 = 'h'
beq, $t0, 'H', quit
nop

 #-----------------------------------------------------
ShowKey: sw $t0, 0($s0) # show key
 nop
 #-----------------------------------------------------
 j loop
 nop

quit:

```

- Kết quả:
  Dịch 8 bit:



Keyboard and Display MMIO Simulator, Version 1.4

Keyboard and Display MMIO Simulator

DISPLAY: Store to Transmitter Data 0xffff000c, cursor 8, area 95 x 10

```
nnokjkno
```

☑ DAD

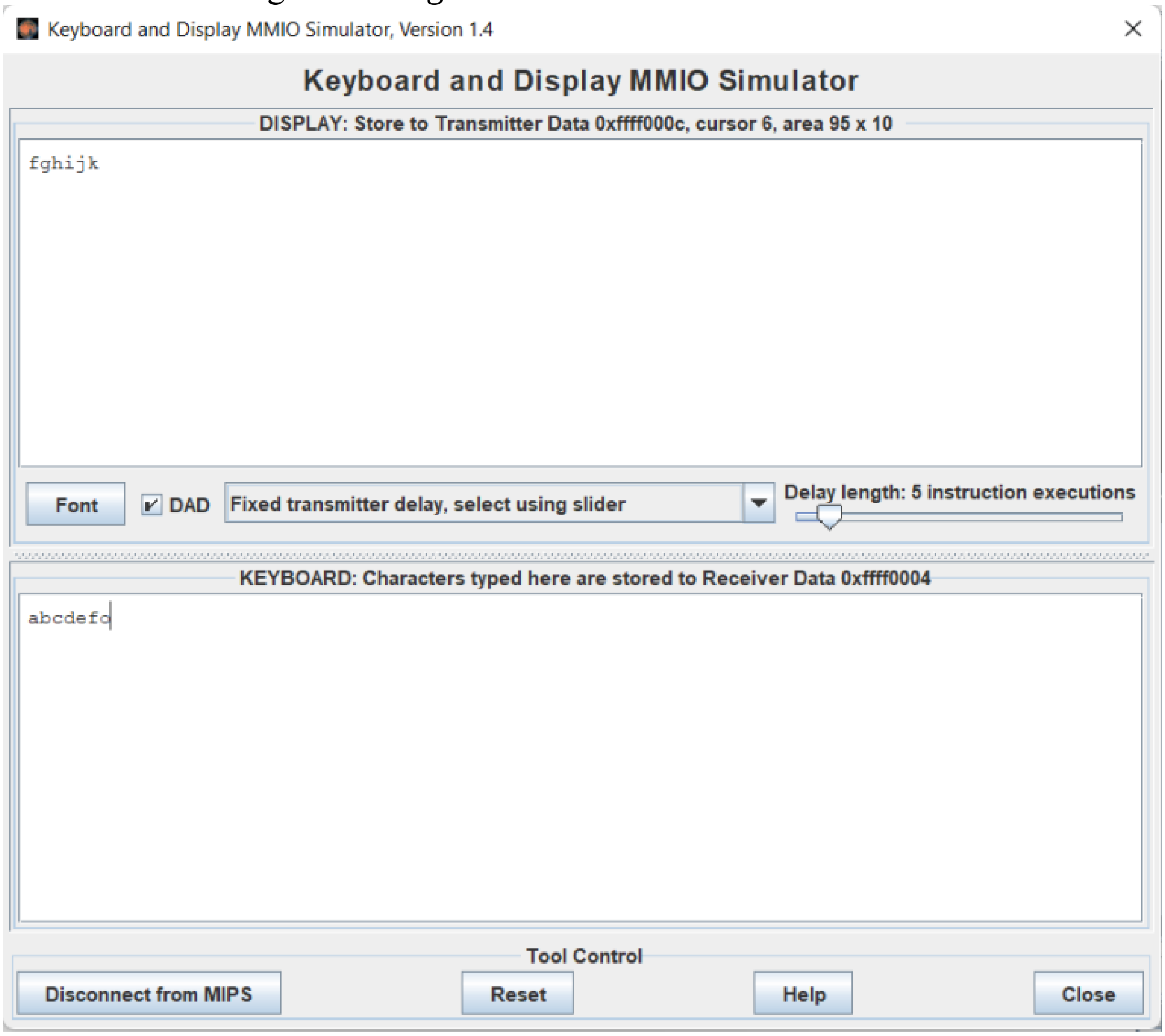
Fixed transmitter delay, select using slider

Delay length: 5 instruction executions

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

```
ffgcbcfg
```

Bấm ‘o’ thì chương trình dừng:


Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c, cursor 6, area 95 x 10

```
fghijk
```

Font
☑ DAD
Fixed transmitter delay, select using slider
Delay length: 5 instruction executions
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004

```
abcdefo
```

Tool Control
Disconnect from MIPS
Reset
Help
Close

- Giải thích:
  - Dòng 8: Gán k0 là key_code
  - Dòng 9: gán k1 là key_read
  - Dòng 11: gán s0 là display_code
  - Dòng 12: gán s1 là display_ready
  - Dòng 15: nhãn WaitForKey: đọc ký tự key_ready vào t1
  - Dòng 17: nếu t1 = 0 thì chờ ký tự tiếp theo
  - Dòng 20: nhãn ReadKey: đọc ký tự nhập từ bàn phím vào t0
  - Dòng 23->26: nhãn WaitForDis: hiển thị ra màn hình
  - Dòng 28->30: nhãn Encrypt: để mã hóa dịch. Ký tự nhập vào sẽ được dịch đi 5 . Ví dụ nhập ‘1’ thành ‘6’. Khi nhập đến ký tự ‘o’ (mã ASCII là 116) hoặc ‘O’. Tức sau khi mã hóa thành ‘t’ hoặc ‘T’ thì nhảy đến ‘quit’ để thoát chương trình.
  - Dòng 34->38: lưu t0 vào s0 để hiện thị
  - Dòng 39: nhãn quit: thoát chương trình