Bài thực hành số 10

Lớp: 139365 – Học phần : Thực Hành Kiến Trúc Máy Tính

Họ và tên: Hạ Hồng Sơn MSSV: 20215130

Mục tiêu bài học:

-

Assignment 1: Giao tiếp với cổng LED

1. Đoạn 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, 0xbf # set value for segments
    jal SHOW_7SEG_LEFT # show
    nop
    li $a0, 0x3f # 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
```

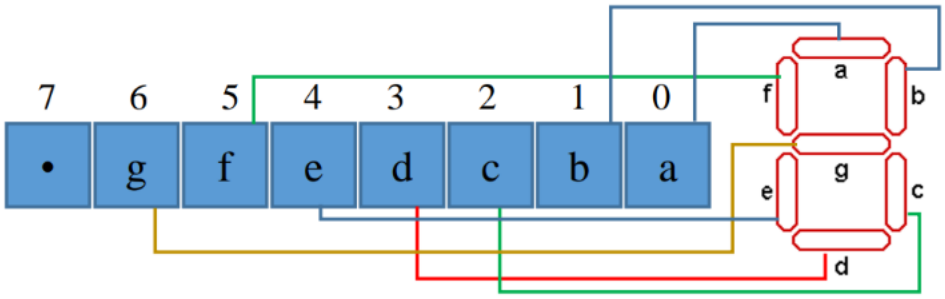
2. Giải thích:

- .eqv SEVENSEG_LEFT 0xFFFF0011: định nghĩa hằng số dùng để xác định địa chỉ của đèn LED 7 đoạn bên trái (0xFFFF0011).
- .eqv SEVENSEG_RIGHT 0xFFFF0010: định nghĩa hằng số để xác định địa chỉ của đèn LED 7 đoạn bên phải (0xFFFF0010).

Trong hàm main:

Thực hiện nhập giá trị in trên LED 7 thanh và in ra màn hình

- Để hiển thị giá trị số đó lên LED 7, ta xem số cần in cần những thanh sáng nào:



Vd: Với số 0: cần những thanh a, b, c, d, e, f. Ta có bảng:

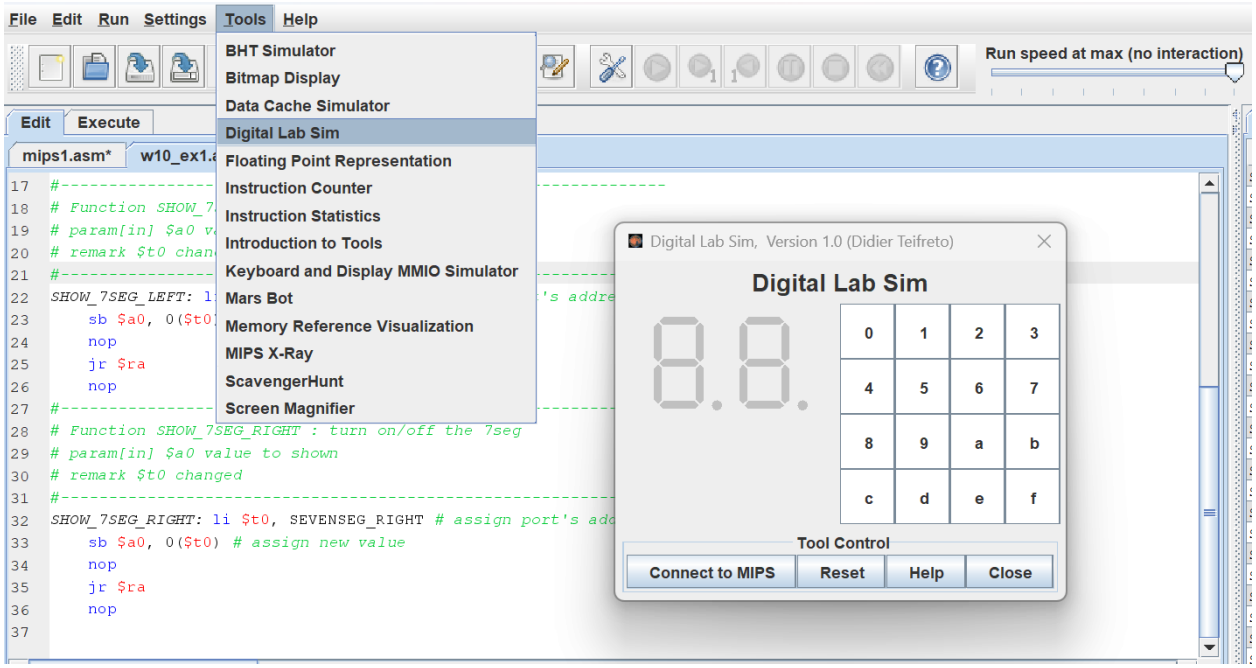
*	g	f	e	d	c	b	a
0	0	1	1	1	1	1	1

Từ đó ta được xâu nhị phân: $00111111_2 \rightarrow 0x3f_{hex}$

- $a0 : lưu giá trị hiển thị của LED 7 thanh
- Sau đó hiển thị ra màn hình thông qua SHOW_7SEG_LEFT, và SHOW_7SEG_RIGHT :
  - **SHOW_7SEG_LEFT** (hiển thị đèn LED 7 đoạn bên trái)
  - **SHOW_7SEG_RIGHT** (hiển thị đèn LED 7 đoạn bên phải)

Trong hàm **SHOW_7SEG_LEFT** và **SHOW_7SEG_RIGHT**:

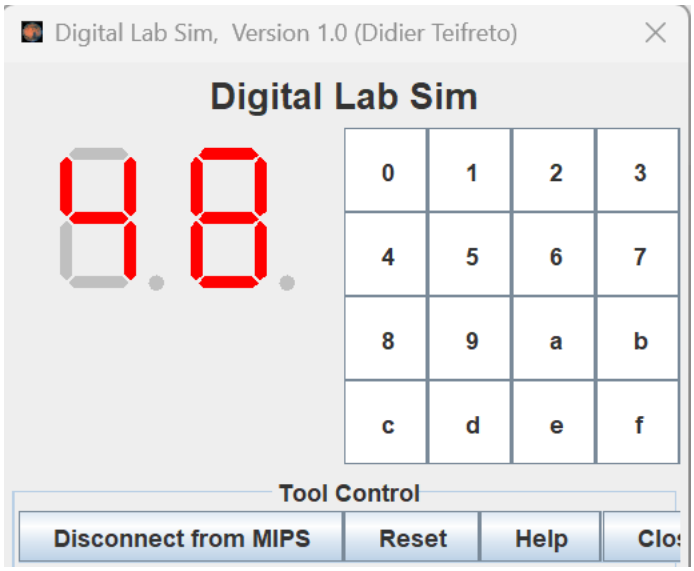
- Thực hiện đưa giá trị cần in ($a0) vào địa chỉ của LED 7 thanh ($t0) và hiển thị ra màn hình thông qua tools: **Digital Lab Sim**



3. Kết quả:

Cần in ra số 48: Ta có

- Số 4 cần những thanh: b, c, f, g → 01100110 → 0x66
- Số 8 cần những thanh: a, b, c, d, e, f, g → 01111111 → 0x7f



In số 48 trên 2 LED 7 thanh

Assignment 2: Hiển thị trên BITMAP

Đoạn code:

```
# Assignment 2 - Week 10

.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, RED
        sw      $t0, 72($k0)

        li      $t0, RED
        sw      $t0, 76($k0)

        li      $t0, GREEN
        sw      $t0, 80($k0)

        li      $t0, GREEN
        sw      $t0, 84($k0)

        li      $t0, RED
        sw      $t0, 104($k0)
```

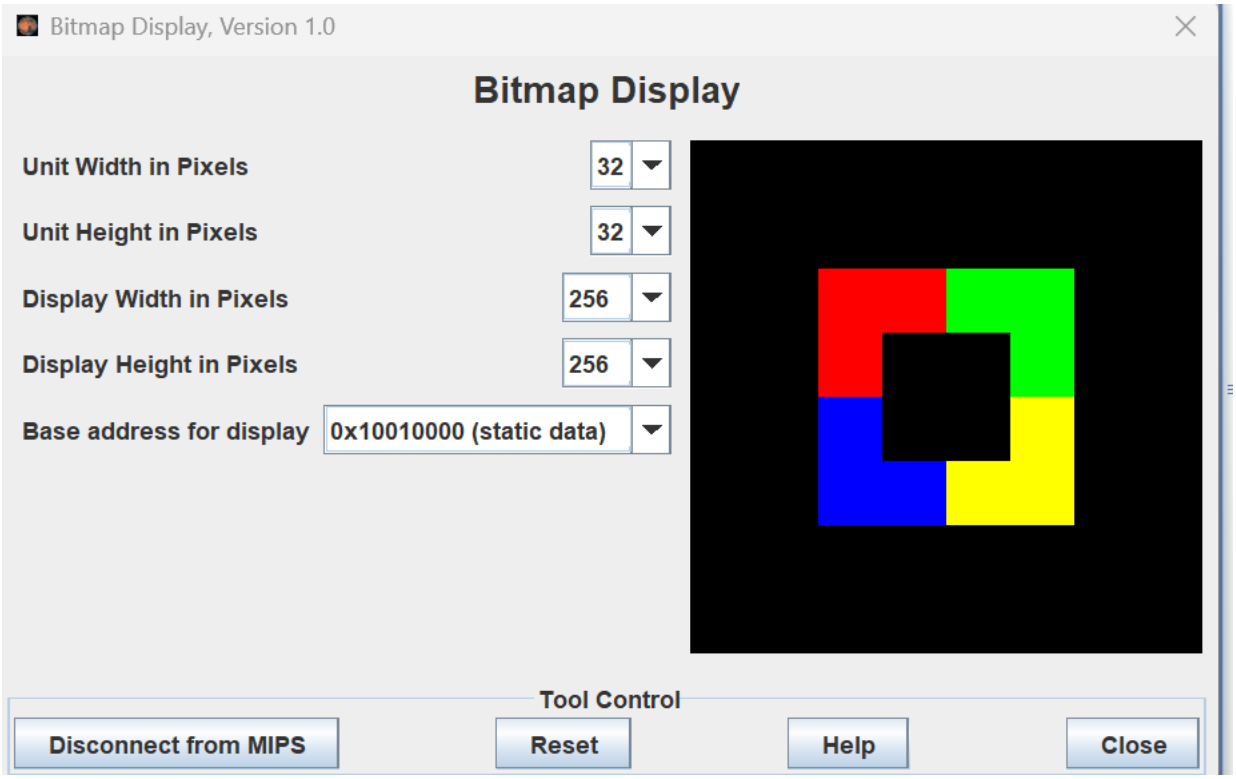
```
        li      $t0, BLUE
        sw      $t0, 136($k0)

        li      $t0, GREEN
        sw      $t0, 116($k0)

        li      $t0, YELLOW
        sw      $t0, 148($k0)

        li      $t0, BLUE
        sw      $t0, 168($k0)

        li      $t0, BLUE
        sw      $t0, 172($k0)

        li      $t0, YELLOW
        sw      $t0, 176($k0)

        li      $t0, YELLOW
        sw      $t0, 180($k0)
```

Giải thích:

- '.eqv MONITOR_SCREEN 0x10010000': Xác định địa chỉ bắt đầu của bộ nhớ màn hình MONITOR_SCREEN với giá trị 0x10010000.
- '.eqv RED 0x00FF0000': Lưu hằng số RED với giá trị 0x00FF0000 (đại diện cho giá trị màu đỏ)

    - tương tự với GREEN, BLUE, WHITE, YELLOW
- Hiển thị pixel trên màn hình 256x256, với mỗi pixel là 32x32 ta thu được bảng 8 hàng x 8 cột
- Để in mỗi pixel:
    - lưu giá trị màu vào thanh ghi $t0
    - Sau đó lưu giá trị của thanh ghi $t0 vào địa chỉ của màn hình in ($k0) để hiển thị lên màn hình
- Thực hiện in ra màn hình với tools: **Bitmap Display**

Kết quả:



Assignment 3: Vẽ tam giác cân với BOT MARS

Đoạn 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:
move_to_center:
    addi $a0, $zero, 135 # Marsbot rotates 135* and startrunning
    jal ROTATE
    nop
    jal GO
    nop
sleep0: addi $v0,$zero,32 # Keep running by sleeping in 5000 ms
    li $a0,5000
    syscall
    # Start to draw
    jal TRACK         # draw track line
    nop
    addi $a0, $zero, 60 # Marsbot rotates 60* and startrunning
    jal ROTATE
    nop
```

```
    jal GO
    nop
sleep1: addi $v0,$zero,32 # Keep running by sleeping in 5000 ms
    li $a0,5000
    syscall

    jal UNTRACK          # keep old track
    nop
    jal TRACK            # and draw new track line
    nop
goDOWN: addi $a0, $zero, 180 # Marsbot rotates 180*
    jal ROTATE
    nop

sleep2: addi $v0,$zero,32 # Keep running by sleeping in 5000 ms
    li $a0,5000
    syscall
    jal UNTRACK          # keep old track
    nop
    jal TRACK            # and draw new track line
    nop
```

```
goLEFT: addi $a0, $zero, 300 # Marsbot rotates 270*
    jal ROTATE
    nop

sleep3: addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
    li $a0,5000
```

```
    syscall
    jal UNTRACK          # keep old track
    nop
end_main:
    li $v0, 10
    syscall


#-------------------------------------------------------------
# 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
#-------------------------------------------------------------
# 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
```

Giải thích:

Lấy địa chỉ lưu vào các hằng số:

- **HEADING**: Biểu thị hướng hoặc góc hiện tại của MarsBot. (0xffff8010)
    - Nó là một giá trị nguyên từ 0 đến 359
    - 0 cho North (lên)
    - 90 cho East (phải)
    - 180 cho South (xuống)
    - 270 cho Tây (trái).
- **MOVING**: giá trị boolean xác định liệu MarsBot có nên di chuyển hay không (0xffff8050). Đó là một địa chỉ có thể lưu trữ giá trị 0 hoặc giá trị khác 0. Trong khi 0 có nghĩa là nó đứng yên.
- **LEAVETRACK**: giá trị boolean cho biết liệu MarsBot có nên rời khỏi track hay không (0xffff8020). Tương tự như MOVING, nó là một địa chỉ có thể chứa giá trị 0 hoặc khác 0.

- Các hàm điều khuyển:

- GO: Quy trình này khởi động BOT tự hành di chuyển:
    - Load địa chỉ của MOVING và thực hiện cập nhập nội dung là 1 để BOT bắt đầu chạy
- STOP: Làm BOT dừng di chuyển:
    - Trái với GO, STOP load địa chỉ của MOVING và cập nhập nội dung là 0 để BOT dừng lại
- TRACK: Quy trình này bắt đầu vẽ đường đi:
    - Load địa chỉ của LEAVETRACK (lưu vào $at), và lưu giá trị 1 vào địa chỉ để kịch hoạt vẽ đường đi của BOT
- UNTRACK: Quy trình này dừng xe tự hành vẽ đường đi:
    - Trái với TRACK, UNTRACK load địa chỉ của LEAVETRACK (lưu vào $at), và gán giá trị 0 vào địa chỉ để dừng vẽ đường đi của BOT.

- ROTATE: Quy trình này xoay máy dò theo một góc xác định:
    - Truy cập đến địa chỉ của HEADING, và thực hiện gán giá trị góc quay và địa chỉ ấy → Giúp BOT di chuyển theo góc đã định.

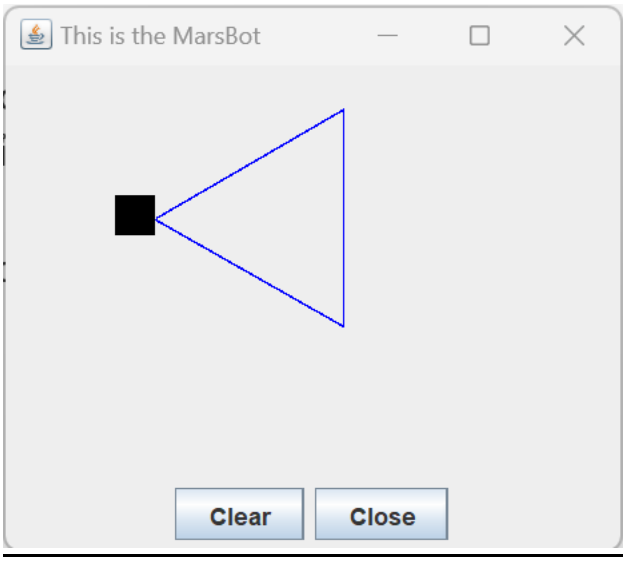
Vẽ hình tam giác cân:

+) b1: thực hiện di chuyển đến giữa mang hình;

+) b2 vẽ từng cạnh của tam giác với độ dài bằng nhau (bằng cách cho sleep cùng một khoảng thời gian), và thay đổi góc quay mỗi cạnh)

- Xoay BOT 60 độ.
- Di chuyển BOT về phía trước trong 5000 mili giây.
- Xoay BOT 180 độ.
- Di chuyển BOT về phía trước trong 5000 mili giây.
- Xoay BOT 300 độ.
- Di chuyển BOT về phía trước trong 5000 mili giây.
- Dừng xe.

Kết quả:



Tam giác cân vẽ bởi đường đi của BOT

Assignment 4: Nhập và hiển thị ký tự (‘exit’ → thoát chương trình)

Đoạn 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 has already to do
                                # Auto clear after sw

.data
TEXT: .word 10000

.text
li $k0, KEY_CODE
li $k1, KEY_READY
li $s0, DISPLAY_CODE
li $s1, DISPLAY_READY

la $t3, TEXT # Current address of the last letter in a 4-letter sequence (exit)
addi $t3, $t3, 12

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
#---------------------------------------------------
StoreKey:
        addi $t3, $t3, 4
        sw $t0, 0($t3)
#---------------------------------------------------
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
#---------------------------------------------------

Check_for_exit:
        lw $t4, -12($t3) # Load the first character in the 4 character sequences
        bne $t4, 101, loop # check for letter e
        nop
        lw $t4, -8($t3)
        bne $t4, 120, loop # check for letter x
        nop
        lw $t4, -4($t3)
        bne $t4, 105, loop # check for letter i
        nop
        lw $t4, 0($t3)
        bne $t4, 116, loop # check for letter t
        nop
end_check:
end:
        li $v0, 10
        syscall
```

Giải thích:

Định nghĩa các địa chỉ:

- **KEY_CODE**: Địa chỉ lưu trữ mã ASCII từ bàn phím, kích thước 1 byte.
- **KEY_READY**: Địa chỉ lưu trữ giá trị = 1 nếu có một mã ASCII mới từ bàn phím. Giá trị này tự động được xóa sau khi thực hiện lệnh lw (load word).
- **DISPLAY_CODE**: Địa chỉ lưu trữ mã ASCII để hiển thị trên màn hình, kích thước 1 byte.
- **DISPLAY_READY**: Địa chỉ lưu trữ giá trị = 1 nếu màn hình đã sẵn sàng để hiển thị mã ASCII mới. Giá trị này tự động được xóa sau khi thực hiện lệnh sw (store word).

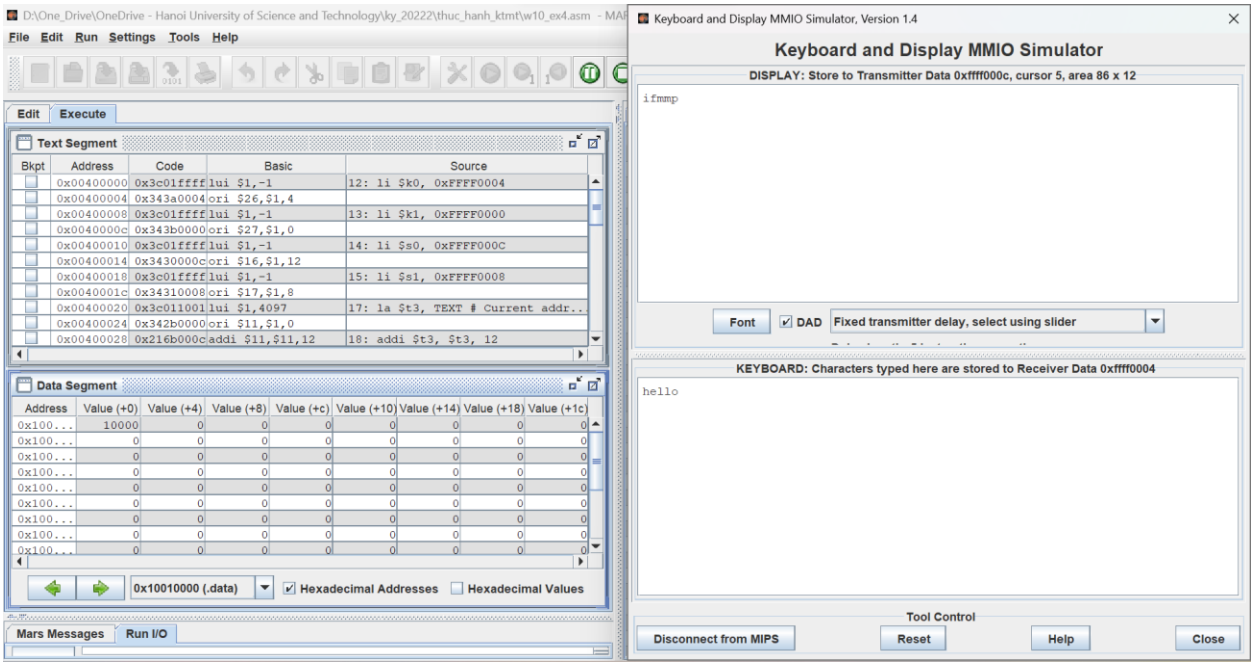
Vòng lặp **loop:**

- **WaitForKey**: Chờ đến khi có một mã ASCII mới từ bàn phím.
  - lw để load giá trị của địa chỉ KEY_READY vào thanh ghi $t1.
  - Kiểm tra xem giá trị trong $t1 có bằng 0 hay không. Nếu bằng 0, tiếp tục kiểm tra cho đến khi có mã ASCII mới từ bàn phím.
- **ReadKey**: Đọc mã ASCII từ bàn phím.
  - lw để load giá trị của KEY_CODE vào thanh ghi $t0.
- **StoreKey**: Lưu giá trị vừa đọc được liên tiếp vào thành ghi $(t3)
- **WaitForDis**: Chờ đến khi màn hình sẵn sàng hiển thị.
  - lw để load giá trị của DISPLAY_READY vào thanh ghi $t2.
  - Kiểm tra xem giá trị trong $t2 có bằng 0 hay không. Nếu bằng 0, tiếp tục kiểm tra cho đến khi màn hình sẵn sàng.
- **Encrypt**: Thực hiện mã hóa (thay đổi) mã ASCII đọc được từ bàn phím (tăng giá trị lên 1).
- **ShowKey**: Hiển thị mã ASCII đã mã hóa lên màn hình.

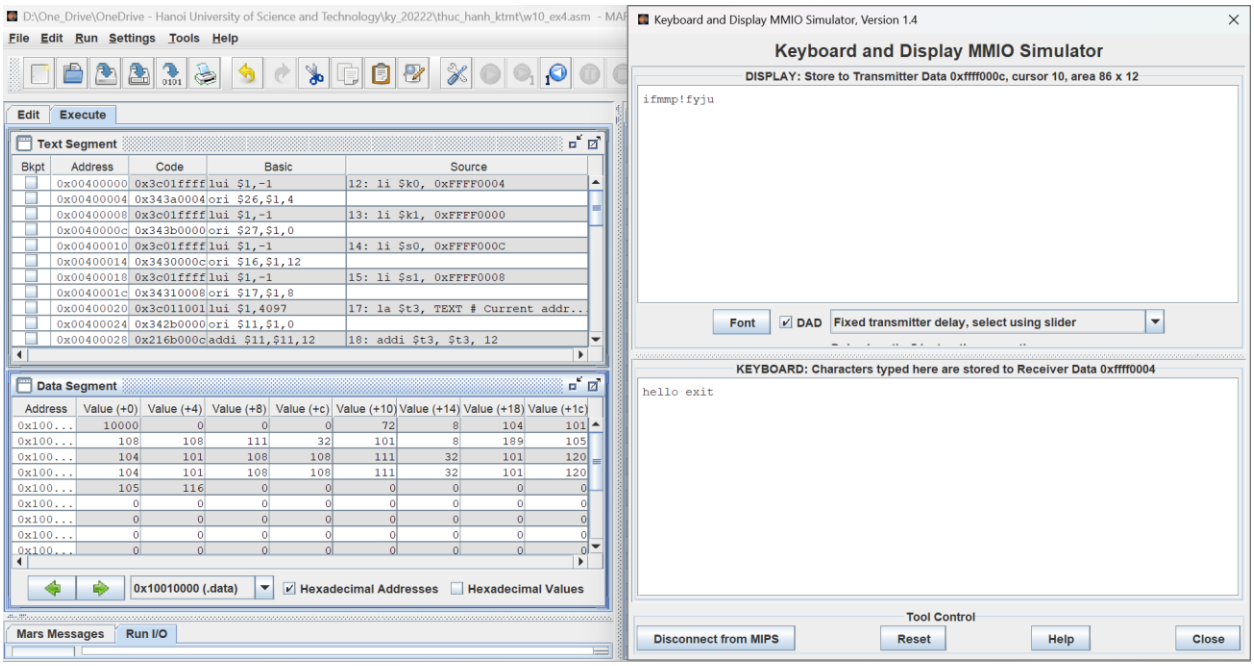
    - sw để lưu giá trị từ thanh ghi $t0 vào địa chỉ DISPLAY_CODE.
- **Check_for_exit:** Kiểm tra xem các ký tự vừa nhập xem 4 ký tự cuối cùng có phải là xâu ‘exit’ không. Nếu không tiếp tục lặp, nếu trùng với ‘exit’ thực hiện thoát chương trình.
- Quay lại vòng lặp loop.
- Hiển thị với tool: **Keyboard and Display MMIO Simulator**

Kết quả:

- Nhập từ bàn phím “hello”:



- Tiếp tục nhập xâu “exit” → Thoát chương trình


D:\One_Drive\OneDrive - Hanoi University of Science and Technology\ky_20222\thuc_hanh_ktmt\w10_ex4.asm - MAR
File Edit Run Settings Tools Help
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x3c01ffff lui $1,-1 12: li $k0, 0xFFFF0004
0x00400004 0x343a0004 ori $26,$1,4
0x00400008 0x3c01ffff lui $1,-1 13: li $k1, 0xFFFF0000
0x0040000c 0x343b0000 ori $27,$1,0
0x00400010 0x3c01ffff lui $1,-1 14: li $s0, 0xFFFF000C
0x00400014 0x3430000c ori $16,$1,12
0x00400018 0x3c01ffff lui $1,-1 15: li $s1, 0xFFFF0008
0x0040001c 0x34310008 ori $17,$1,8
0x00400020 0x3c011001 lui $1,4097 17: la $t3, TEXT # Current addr...
0x00400024 0x342b0000 ori $11,$1,0
0x00400028 0x216b000c addi $11,$11,12 18: addi $t3, $t3, 12
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x100... 10000 0 0 0 72 8 104 101
0x100... 108 108 111 32 101 8 189 105
0x100... 104 101 108 108 111 32 101 120
0x100... 104 101 108 108 111 32 101 120
0x100... 105 116 0 0 0 0 0 0
0x100... 0 0 0 0 0 0 0 0
0x100... 0 0 0 0 0 0 0 0
0x100... 0 0 0 0 0 0 0 0
0x100... 0 0 0 0 0 0 0 0
0x10010000 (.data)
Hexadecimal Addresses
Hexadecimal Values
Mars Messages
Run I/O
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c, cursor 10, area 86 x 12
ifmmp!fyju
Font
DAD
Fixed transmitter delay, select using slider
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
hello exit
Tool Control
Disconnect from MIPS
Reset
Help
Close