# Bài thực hành số 6

# Lớp: 139365 – Học phần: Thực hành Kiến Trúc Máy Tính

**Họ và tên: Đinh Thị Hồng Phúc** **MSSV: 20215118**

Bài 1.

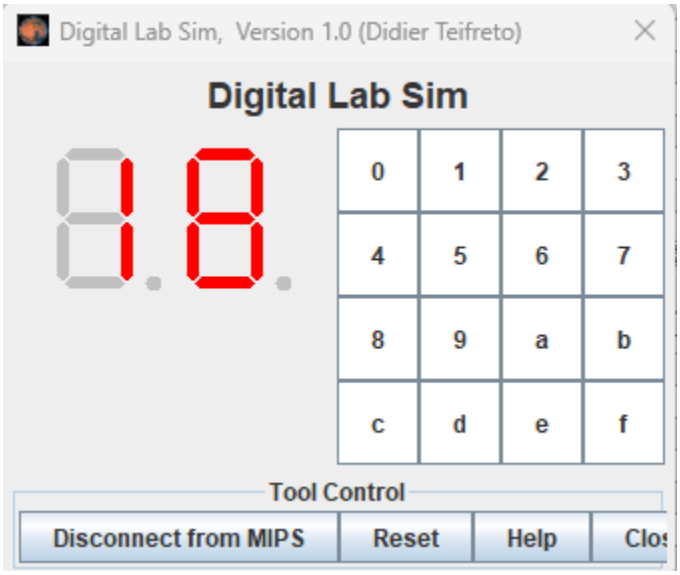
```
#Home Assignment 1
.eqv SEVENSEG_LEFT 0xFFFF0011
.eqv SEVENSEG_RIGHT 0xFFFF0010
.text
main:
        li $a0, 0x6             #set value for segments
        jal SHOW_7SEG_LEFT      #show
        nop
        li $a0, 0x7F            #set value for segments
        jal SHOW_7SEG_RIGHT     #show
        nop
exit:
        li $v0, 10
        syscall
endmain:

SHOW_7SEG_LEFT:
        li $t0, SEVENSEG_LEFT   # assign port's address
        sb $a0, 0($t0)          # assign new value
        nop
        jr $ra
        nop

SHOW_7SEG_RIGHT:
        li $t0, SEVENSEG_RIGHT  # assign port's address
        sb $a0, 0($t0)          # assign new value
        nop
        jr $ra
        nop
```

*Thực hiện gõ chương trình vào công cụ **MARS***

Giải thích:

- Đầu tiên dùng *.eqv* để định nghĩa 2 hằng số, để lưu địa chỉ của 2 số trên LED 7 thanh (bên trái và bên phải)

- *main:* dùng để gán giá trị đầu vào vào thanh ghi $a0 đại diện cho giá trị mà các thanh bên trái, phải sẽ hiển thị qua 2 chương trình con *SHOW_7SEG_LEFT* và *SHOW_7SEG_RIGHT*
- *exit:* thoát chương trình
- *SHOW_7SEG_LEFT* tương tự *SHOW_7SEG_RIGHT:* Đầu tiên gán địa chỉ của màn hình LED 7 thanh tương ứng (trái, phải) vào thanh ghi $t0 (dòng 18 và 25). Sau đó gán giá trị trong $a0 vào địa chỉ của màn hình hiển thị LED 7 thanh (dòng 19 và 26). Lệnh *jr $ra* được dùng để trở về địa chỉ gọi hàm (trong *main*)




*Thực hiện chạy chương trình với **MARS***

Cách hiển thị số 18 trên LED 7 thanh:

- Hiển thị 1: chỉ thanh b và c sáng → 0x6

| • | g | f | e | d | c | b | a |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

- Hiển thị 8: tất cả các thanh cùng sáng → 0x7F

| • | g | f | e | d | c | b | a |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

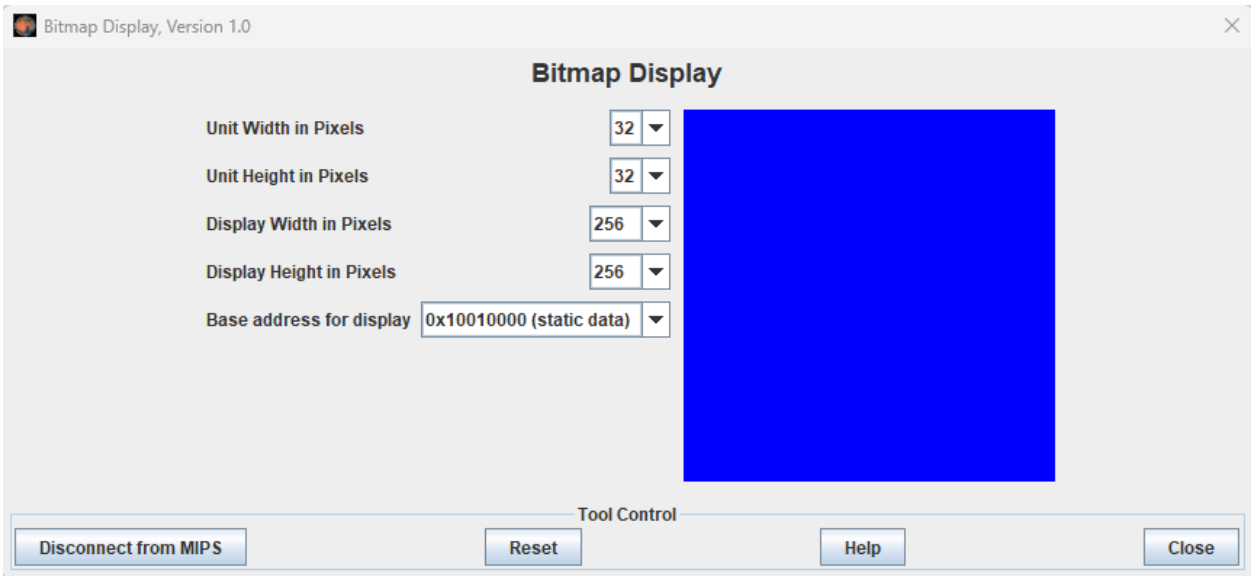
Bài 2.

```
1  #Home Assignment 2
2  .eqv MONITOR_SCREEN 0x10010000          #Dia chi bat dau cua bo nho man hinh
3  .eqv RED 0x00FF0000                     #Cac gia tri mau thuong su dung
4  .eqv GREEN 0x0000FF00
5  .eqv BLUE 0x000000FF
6  .eqv WHITE 0x00FFFFFF
7  .eqv YELLOW 0x00FFFF00
8  .text
9          li $k0, MONITOR_SCREEN
10         li $a0, BLUE
11         li $s0, 0
12         li $s1, 256
13 scan:
14         add $t0, $k0, $s0
15         sw $a0, 0($t0)
16         addi $s0, $s0, 4
17         ble $s0, $s1, scan
```

*Thực hiện gõ chương trình vào công cụ **MARS***

Giải thích:

- Dòng 1: Xác định địa chỉ bắt đầu của bộ nhớ màn hình, nơi các pixel được lưu trữ
- Các dòng lệnh *.eqv* tiếp theo dùng để định nghĩa các hằng số cho các giá trị màu sắc
- Dòng 9, 10: Đặt giá trị của địa chỉ màn hình vào thanh ghi $k0, giá trị màu *blue* vào thanh ghi $a0
- Thanh ghi $s0 được gán giá trị bằng 0 dùng để lưu địa chỉ của từng điểm ảnh trên màn hình, $s1 gán bằng 256 là giới hạn số điểm ảnh
- Vòng lặp *scan* thực hiện duyệt qua từng điểm ảnh bằng cách tính địa chỉ của điểm ảnh hiện tại (cộng địa chỉ bắt đầu của màn hình - $k0 với giá trị $s0), lưu giá trị màu vào điểm ảnh tại địa chỉ hiện tại. Sau đó tăng giá trị $s0 lên 4 để trỏ đến điểm ảnh tiếp theo. Khi $s0 vẫn nhỏ hơn hoặc bằng $s1 thì tiếp tục vòng lặp




*Thực hiện chạy chương trình với **MARS***

Bài 3.

```
#Home Assignmnet 3
.eqv HEADING 0xffff8010
.eqv MOVING 0xffff8050
.eqv LEAVETRACK 0xffff8020
.eqv WHEREX 0xffff8030
.eqv WHEREY 0xffff8040
.text
main:
        jal TRACK
        nop
        addi $a0, $zero, 90
        jal ROTATE
        nop
        jal GO
        nop
sleep1:
        addi $v0, $zero, 32
        li $a0, 5000
        syscall

        jal UNTRACK
        nop
        jal TRACK
        nop
```

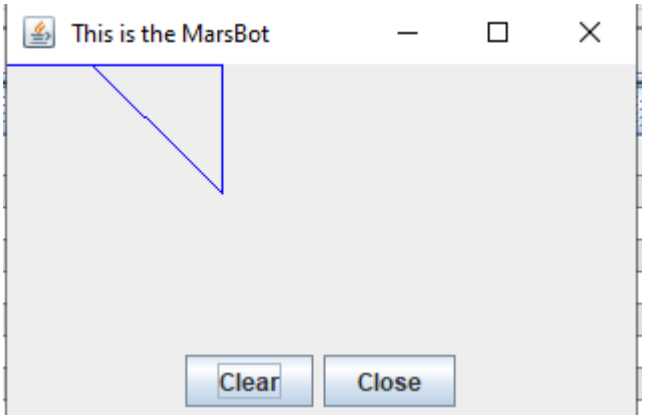
```
goDOWN:
        addi $a0, $zero, 180
        jal ROTATE
        nop

sleep2:
        addi $v0, $zero, 32
        li $a0, 3000
        syscall
        jal UNTRACK
        nop
        jal TRACK
        nop
goLEFT:
        addi $a0, $zero, 315
        jal ROTATE
        nop

sleep3:
        addi $v0, $zero, 32
        li $a0, 5000
        syscall
        jal UNTRACK
        nop
        jal TRACK
        nop
end_main:

GO:
        li $at, MOVING
        addi $k0, $zero, 1
        sb $k0, 0($at)
        nop
        jr $ra
        nop
STOP:
        li $at, MOVING
        sb $zero, 0($at)
        nop
        jr $ra
        nop
TRACK:
```

```
67          li $at, LEAVETRACK
68          addi $k0, $zero, 1
69          sb $k0, 0($at)
70          nop
71          jr $ra
72          nop
73  UNTRACK:
74          li $at, LEAVETRACK
75          sb $zero, 0($at)
76          nop
77          jr $ra
78          nop
79  ROTATE:
80          li $at, HEADING
81          sw $a0, 0($at)
82          nop
83          jr $ra
84          nop
```

*Thực hiện gõ chương trình vào công cu **MARS***

Giải thích:

- *.eqv* được dùng để định nghĩa các hằng số *HEADING, MOVING, LEAVETRACK, WHEREX, WHEREY* đại diện cho các địa chỉ trong bộ nhớ
- *main:* gọi chương trình con *TRACK*. Chương trình con này bật cờ *LEAVETRACK* để ghi lại các bước của bot vẽ hình tam giác trên màn hình. Dòng 11 thiết lập giá trị góc xoay là 90 độ → vẽ 1 đường kẻ ngang. Dòng 12 gọi chương trình con *ROTATE,* chương trình con này lưu giá trị góc xoay vào địa chỉ *HEADING*. Tiếp theo gọi chương trình con *GO*, chương trình con này đặt cờ *MOVING* cho phép bot di chuyển
- *sleep1:* Dòng 17, giá trị 32 được gán vào thanh ghi $v0 để thực hiện chức năng delay. Dòng 18 gán trị trị 5000 vào thanh ghi $a0 đại diện cho số lượng đơn vị thời gian (microseconds) muốn bot vẽ đường thẳng. Chương trình con *UNTRACK* sẽ tắt cờ *LEAVETRACK, TRACK* sẽ bật cờ *LEAVETRACK* để ghi lại đường kẻ
- Các hướng khác của tam giác (kẻ xuống – 180 độ, kẻ cạnh huyền – 315 độ) tương tự như đường kẻ ngang với thời gian chờ (hàm *sleep*) tương ứng
- Các chương trình con như *GO, STOP, TRACK, UNTRACK, ROTATE* sử dụng để tiếp tục di chuyển, dừng lại, chuyển sang dòng kẻ khác, xoay. Thao tác với các cờ và giá trị trong bộ nhớ để điều khiển bot




*Thực hiện chạy chương trình với **MARS***

Bài 4.

```
#Home Assignment 4
.eqv KEY_CODE 0xFFFF0004
.eqv KEY_READY 0xFFFF0000
.eqv DISPLAY_CODE 0xFFFF000C
.eqv DISPLAY_READY 0xFFFF0008
.text
        li $k0, KEY_CODE
        li $k1, KEY_READY

        li $s0, DISPLAY_CODE
        li $s1, DISPLAY_READY
loop:
        nop
WaitForKey:
        lw $t1, 0($k1)
        nop
        beq $t1, $zero, WaitForKey
        nop
ReadKey:
        lw $t0, 0($k0)
        addi $t9, $t8, 0
        addi $t8, $t7, 0
        addi $t7, $t6, 0
        addi $t6, $t0, 0
        nop
```

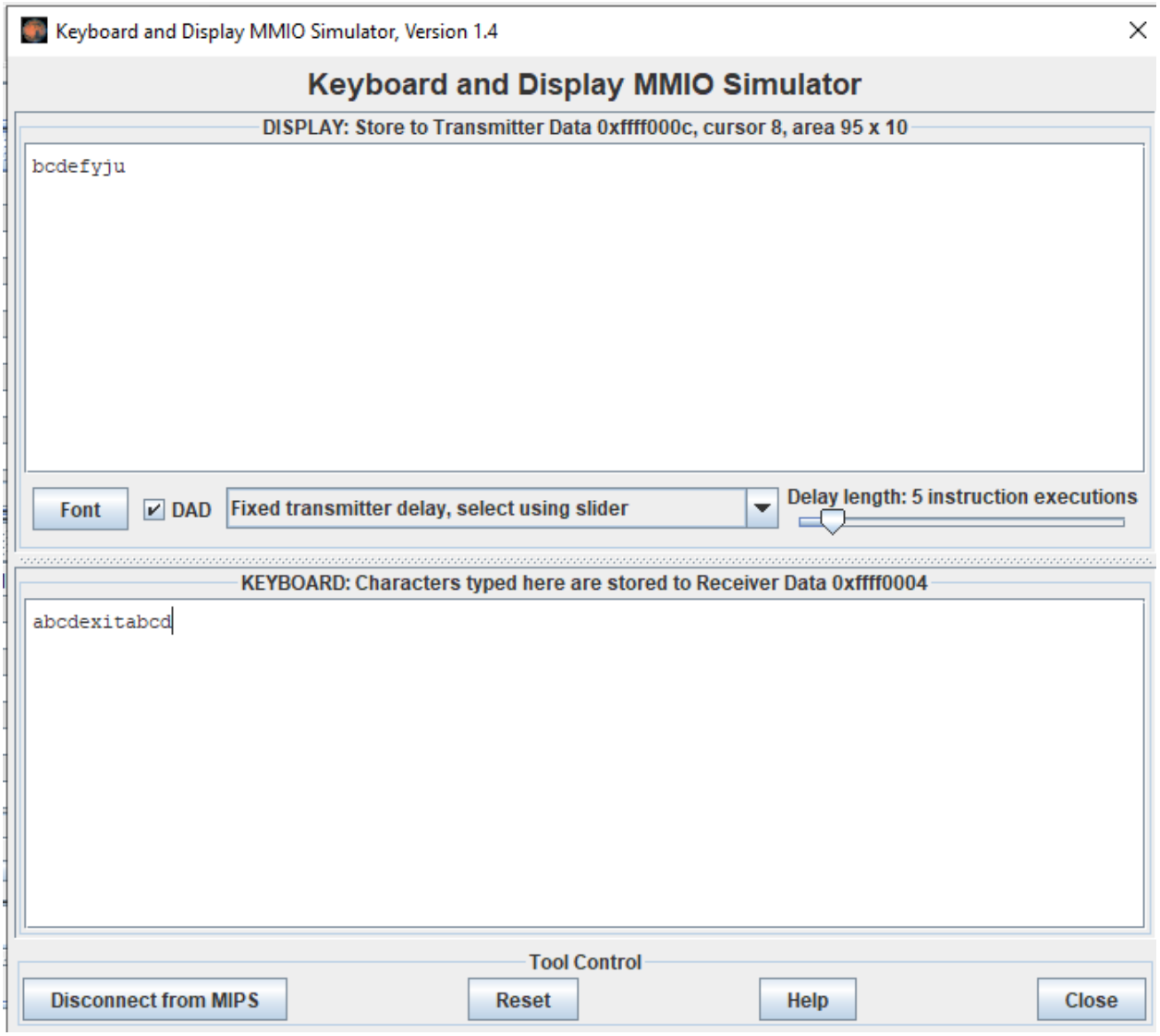
```
26  WaitForDis:
27          lw $t2, 0($s1)
28          nop
29          beq $t2, $zero, WaitForDis
30          nop
31  Encrypt:
32          addi $t0, $t0, 1
33  ShowKey:
34          sw $t0, 0($s0)
35          nop
36  CheckExit:
37          li $t5, 'e'
38          li $t4, 'x'
39          li $t3, 'i'
40          li $t2, 't'
41          bne $t9, $t5, loop
42          bne $t8, $t4, loop
43          bne $t7, $t3, loop
44          bne $t6, $t2, loop
45
46          li $v0, 10
47          syscall
```

*Thực hiện gõ chương trình vào công cụ **MARS***

Giải thích:

- Các dòng lệnh *.eqv* định nghĩa các hằng số để lưu địa chỉ của các thanh ghi điều khiển bàn phím và màn hình
- Dòng 7-11: Đặt giá trị của các hằng số tương ứng vào thanh ghi $k0, $k1 (địa chỉ và trạng thái sẵn sàng của thanh ghi điều khiển bàn phím), $s0, $s1 (địa chỉ và trạng thái sẵn sàng của thanh ghi điều khiển màn hình)
- Vòng lặp *loop* đọc kí tự từ bàn phím, mã hoá và hiển thị trên màn hình
- Vòng lặp *WaitForKey* kiểm tra xem có kí tự nào được nhận từ bàn phím hay chưa. Nếu không, vòng lặp sẽ tiếp tục chờ (tải giá trị từ ô nhớ tại địa chỉ được lưu trong thanh ghi $k1 vào $t1, địa chỉ này đại diện cho trạng thái của bàn phím. Sau đó kiểm tra xem giá trị trong $t1 có bằng 0 không. Nếu bằng 0 thì bàn phím chưa sẵn sàng và vòng lặp tiếp tục chờ)
- Sau khi nhận được kí tự từ bàn phím, vòng lặp *ReadKey* được thực hiện để thực hiện các bước mã hoá và hiển thị. Đọc giá trị của thanh ghi điều khiển bàn phím vào $t0, giá trị kí tự được nhập từ bàn phím. Sao chép giá trị từ $t0 sang $t6, $t7, $t8, $t9
- Vòng lặp *WaitForDis* tương tự như *WaitForKey* dùng để kiểm tra xem màn hình đã sẵn sàng để hiển thị kí tự mới hay chưa. Nếu không sẽ tiếp tục chờ

- Khi màn hình sẵn sàng, vòng lặp *Encrypt* được thực hiện để thay đổi giá trị của kí tự $t0 trước khi hiển thị lên màn hình
- *ShowKey:* Lưu giá trị của $t0 (kí tự đã được mã hoá) vào thanh ghi điều khiển màn hình
- Vòng lặp *CheckExit*: Kiểm tra xem chuỗi nhập vào có phải "exit" hay không. Nếu không, vòng lặp sẽ tiếp tục ở đầu vòng *loop* (đặt giá trị ASCII của chuỗi "exit" vào từng thanh ghi sau đó so sánh giá trị các thanh ghi $t9 → $t6 với từng kí tự. Nếu có bất kì sự khác nhau nào sẽ nhảy đến *loop*)
- Cuối cùng gán $v0 bằng 10 để kết thúc chương trình




*Thực hiện chạy chương trình với **MARS***