ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

****

BÁO CÁO

Bài tập thực hành tuần 10

Học phần: Thực hành kiến trúc máy tính

Giảng viên hướng dẫn: Lê Bá Vui

Sinh viên thực hiện Nguyễn Bình An – 20225591

Hà Nội, tháng 5 năm 2024

Assignment 1: Hiển thị 2 chữ số cuối của mssv (20225591)

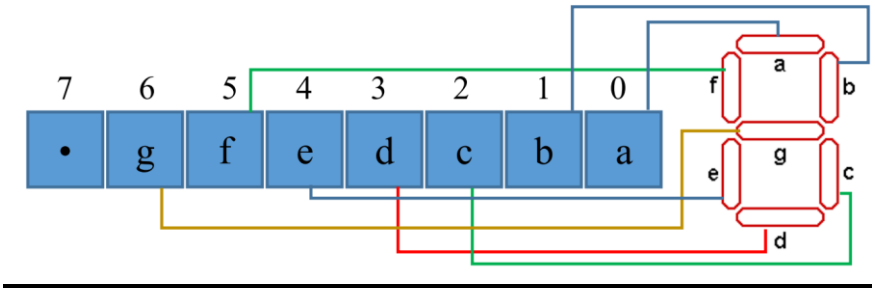
Code:

```
.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, 0xE7 # set value for left segments to display "9"
   jal SHOW_7SEG_LEFT # show "9" on left 7-segment display
   nop
   li $a0, 0x06 # set value for right segments to display "1"
   jal SHOW_7SEG_RIGHT # show "1" on right 7-segment display
   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 to display "9"
   nop
   jr $ra
   nop

SHOW_7SEG_RIGHT:
```

```
li $t0, SEVENSEG_RIGHT # assign port's address
sb $a0, 0($t0) # assign new value to display "1"
nop
jr $ra
nop
```

- Kết nối với địa chỉ của LED 7 thanh trái và phải
- Nhập 8 bit đầu vào thông qua thanh $a0
- Hiển thị kết quả

Chạy thử nghiệm:

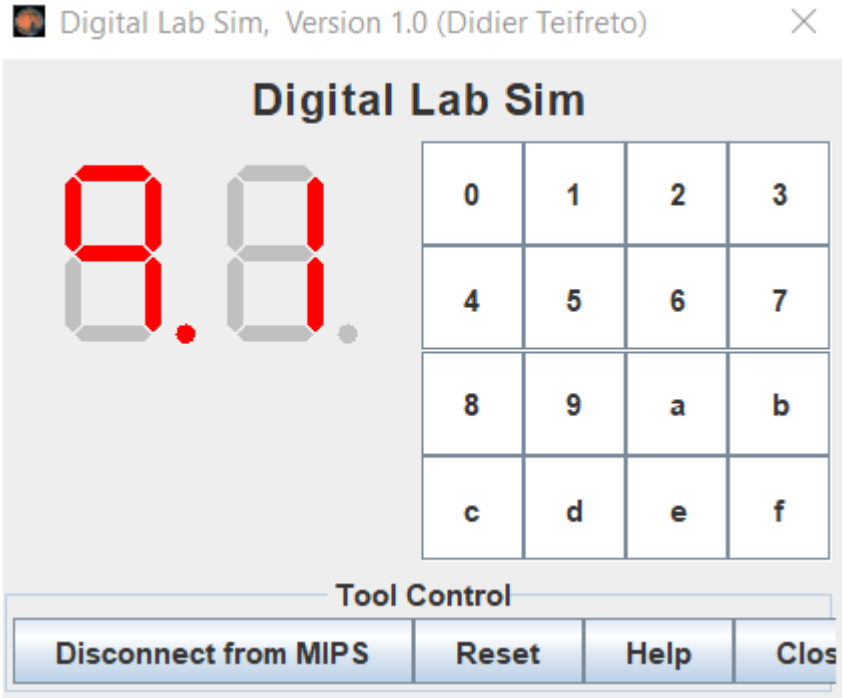


Với 0xE7 = 1 1 1 0 0 1 1 1 -> c,b,a,f,g bật

⇨ In ra số 9.

Với 0x3F = 0 0 0 0 0 1 1 0 -> b,c bật

⇨ In ra số 1



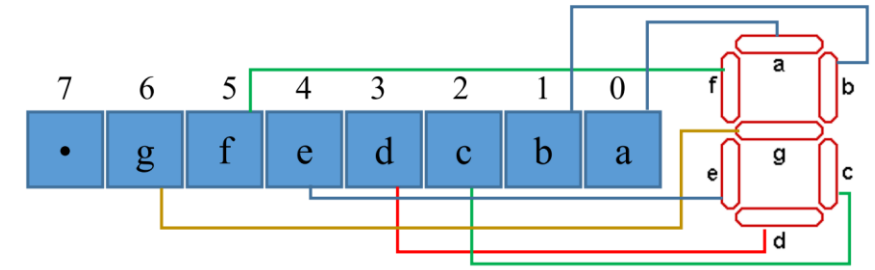
Assignment 2: Nhập vào một số, in ra 2 chữ số cuối ra màn hình LED

Code:

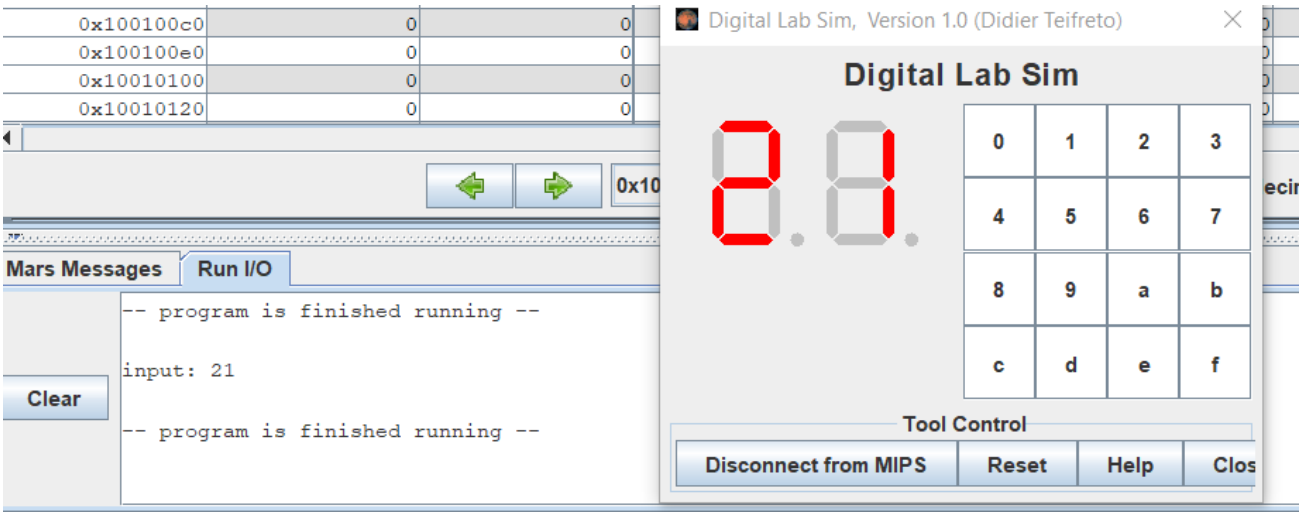
```
.data
prompt: .asciiz "Nhập một số: "
sevenseg_left: .word 0xFFFF0011 # Địa chỉ cho LED 7 đoạn bên trái
sevenseg_right: .word 0xFFFF0010 # Địa chỉ cho LED 7 đoạn bên phải

# Mẫu bit cho các chữ số 0-9 trên LED 7 đoạn bên trái
digit_patterns_left: .word 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x67

# Mẫu bit cho các chữ số 0-9 trên LED 7 đoạn bên phải (mẫu chuẩn)
digit_patterns_right: .word 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F
.text
.globl main
main:
    # In thông báo nhắc nhập
    li $v0, 4       # 4 = in chuỗi
    la $a0, prompt  # tải địa chỉ của chuỗi thông báo
    syscall         # in thông báo

    # Đọc số nguyên nhập vào
    li $v0, 5       # 5 = đọc số nguyên
    syscall         # đọc số nguyên từ người dùng
    move $t0, $v0   # lưu số nhập vào trong $t0

    # Trích xuất hai chữ số cuối
    li $t1, 100     # tải 100 vào $t1
    div $t0, $t1    # chia $t0 cho 100
```

```
    mfhi $t2        # số dư nằm trong $t2 (hai chữ số cuối)

    # Hiển thị chữ số hàng chục trên LED 7 đoạn bên trái
    li $t3, 10      # tải 10 vào $t3 (để làm phép modulo)
    div $t2, $t3    # chia hai chữ số cuối cho 10
    mflo $t4        # thương nằm trong $t4 (chữ số hàng chục)
    sll $t4, $t4, 2 # nhân chữ số hàng chục cho 4 (kích thước từ)
    lw $a0, digit_patterns_left($t4) # tải mẫu bit cho chữ số hàng chục
    jal SHOW_7SEG_LEFT # hiển thị chữ số hàng chục trên LED 7 đoạn bên trái

    # Hiển thị chữ số hàng đơn vị trên LED 7 đoạn bên phải
    mfhi $t4        # số dư nằm trong $t4 (chữ số hàng đơn vị)
    sll $t4, $t4, 2 # nhân chữ số hàng đơn vị cho 4 (kích thước từ)
    lw $a0, digit_patterns_right($t4) # tải mẫu bit cho chữ số hàng đơn vị
    jal SHOW_7SEG_RIGHT # hiển thị chữ số hàng đơn vị trên LED 7 đoạn bên phải
    li $v0, 10
    syscall         # thoát
SHOW_7SEG_LEFT:
    lw $t0, sevenseg_left # tải địa chỉ của LED 7 đoạn bên trái
    sb $a0, 0($t0) # lưu mẫu bit tại địa chỉ của LED 7 đoạn bên trái
    nop
    jr $ra
    nop
SHOW_7SEG_RIGHT:
    lw $t0, sevenseg_right # tải địa chỉ của LED 7 đoạn bên phải
    sb $a0, 0($t0) # lưu mẫu bit tại địa chỉ của LED 7 đoạn bên phải
    nop
    jr $ra
    nop
```

Chạy thử nghiệm:

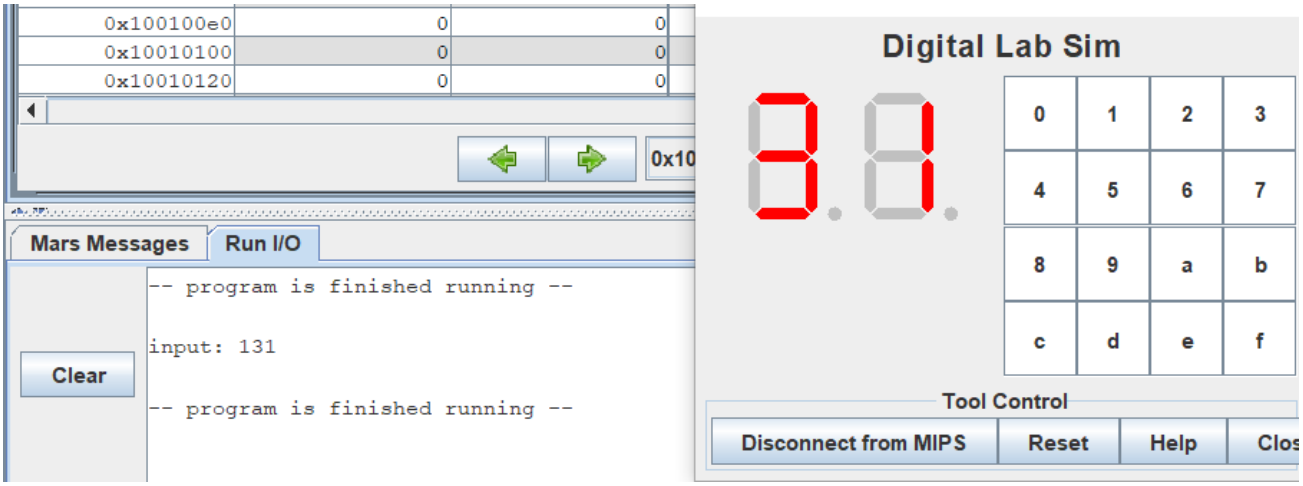


Lần 1: Với input = 21



LED trái: 0x5B: 01011011 -> g,e,d,b,a bật -> 2
LED phải: 0x3F = 0 0 0 0 0 1 1 0 -> b,c bật -> 21

Lần 1: Với input = 131



- Tách ra 2 chữ số cuối là 31
- LED trái:0x4F: 01001111 -> a,b,g,c,d bật -> 3
- LED phải: 0x3F = 0 0 0 0 0 1 1 0 -> b,c bật -> 21

Assignment 3: Nhập vào một ký tự, in ra 2 chữ số cuối của bảng mã ASCII của ký tự đó

Code:

```
.data
prompt: .asciiz "Nhập một ký tự: "
sevenseg_left: .word 0xFFFF0011 # Địa chỉ cho LED 7 đoạn bên trái
sevenseg_right: .word 0xFFFF0010 # Địa chỉ cho LED 7 đoạn bên phải
# Mẫu bit cho các chữ số 0-9 trên LED 7 đoạn bên trái
digit_patterns_left: .word 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x67
# Mẫu bit cho các chữ số 0-9 trên LED 7 đoạn bên phải (mẫu chuẩn)
digit_patterns_right: .word 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F

.text
main:
    # In thông báo nhắc nhập
    li $v0, 4       # 4 = in chuỗi
    la $a0, prompt  # tải địa chỉ của chuỗi thông báo
    syscall         # in thông báo

    # Đọc ký tự nhập vào
    li $v0, 12      # 12 = read_char
    syscall         # đọc ký tự từ người dùng
    move $t0, $v0   # lưu ký tự nhập vào trong $t0 (giá trị ASCII)

    # Trích xuất hai chữ số cuối của mã ASCII
```

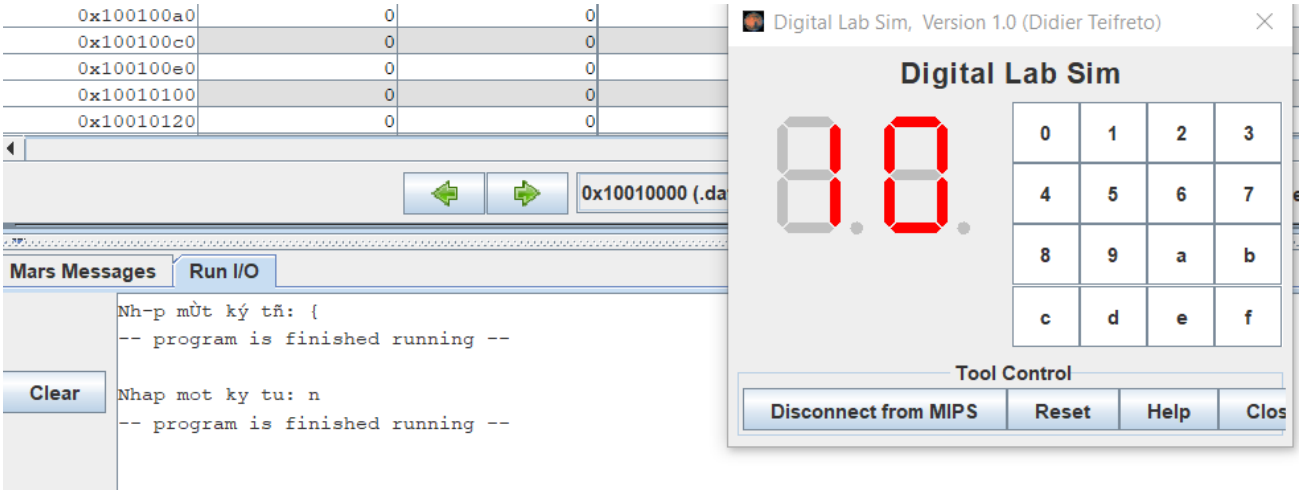
```
    li $t1, 100     # tải 100 vào $t1
    div $t0, $t1    # chia mã ASCII cho 100
    mfhi $t2        # số dư nằm trong $t2 (hai chữ số cuối)

    # Hiển thị chữ số hàng chục trên LED 7 đoạn bên trái
    li $t3, 10      # tải 10 vào $t3 (để làm phép modulo)
    div $t2, $t3    # chia hai chữ số cuối cho 10
    mflo $t4        # thương nằm trong $t4 (chữ số hàng chục)
    sll $t4, $t4, 2 # nhân chữ số hàng chục cho 4 (kích thước từ)
    lw $a0, digit_patterns_left($t4) # tải mẫu bit cho chữ số hàng chục
    jal SHOW_7SEG_LEFT # hiển thị chữ số hàng chục trên LED 7 đoạn bên trái

    # Hiển thị chữ số hàng đơn vị trên LED 7 đoạn bên phải
    mfhi $t4        # số dư nằm trong $t4 (chữ số hàng đơn vị)
    sll $t4, $t4, 2 # nhân chữ số hàng đơn vị cho 4 (kích thước từ)
    lw $a0, digit_patterns_right($t4) # tải mẫu bit cho chữ số hàng đơn vị
    jal SHOW_7SEG_RIGHT # hiển thị chữ số hàng đơn vị trên LED 7 đoạn bên phải

    # Thoát chương trình
    li $v0, 10      # 10 = thoát
    syscall         # thoát

SHOW_7SEG_LEFT:
    lw $t0, sevenseg_left # tải địa chỉ của LED 7 đoạn bên trái
    sb $a0, 0($t0) # lưu mẫu bit tại địa chỉ của LED 7 đoạn bên trái
    nop
    jr $ra
    nop

SHOW_7SEG_RIGHT:
```

```
lw $t0, sevenseg_right # tải địa chỉ của LED 7 đoạn bên phải
sb $a0, 0($t0) # lưu mẫu bit tại địa chỉ của LED 7 đoạn bên phải
nop
jr $ra
nop
```

Lưu ký tự vào $t0, chuyển đổi qua ASCII và tách lấy 2 số cuối bằng cách chia lấy dư cho 100. Sau đó thực hiện in lên LED như bài 2

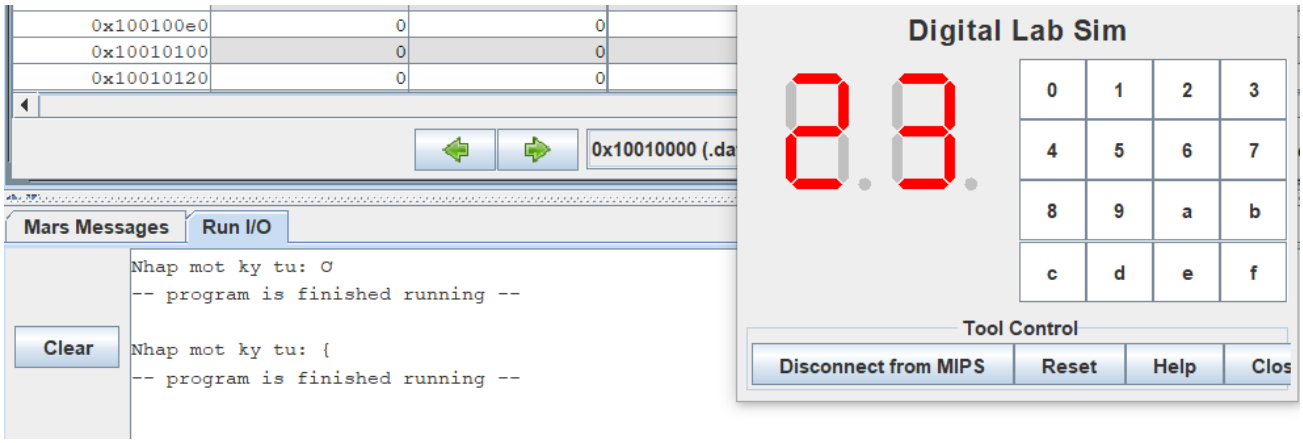
<u>Chạy thử nghiệm:</u>

Lần 1: Với input = n



- b = 110(ASCII) -> in ra số 10:
- LED trái: b,c bật
- LED phải: a,b,c,d,e,f bật

Lần 2: Với input = {



- { = 123 (ASCII) -> in ra số 23:
- LED trái: a,b,g,e,d bật
- LED phải: a,b,g,c,d bật

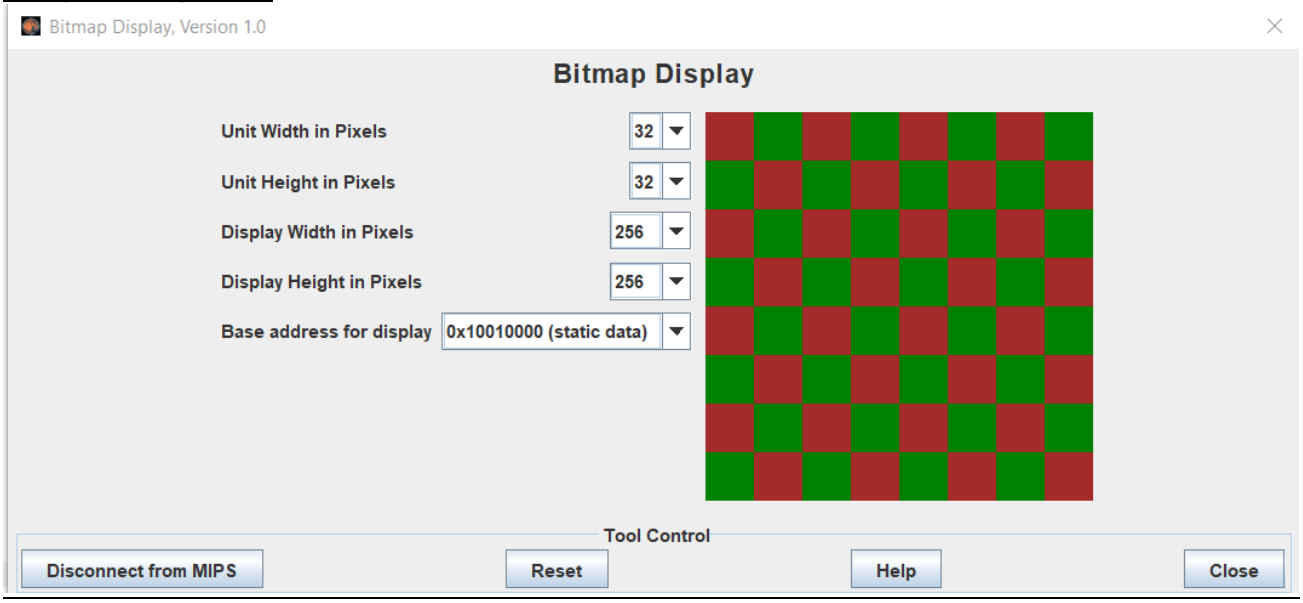
Assignment 4: Vẽ bàn cờ

Code:

```
.eqv MONITOR_SCREEN 0x10010000
.eqv BROWN 0xA52A2A
.eqv GREEN 0x008000

.text
    li $k0, MONITOR_SCREEN  # Đặt địa chỉ bắt đầu của màn hình vào $k0
    li $t1, 8  # Số hàng của bàn cờ
    li $t5, 0  # Khởi tạo biến $t5 để xác định màu sắc bắt đầu của mỗi hàng
row_loop:
    li $t2, 8  # Số cột của bàn cờ
    move $t3, $t5  # Copy giá trị màu bắt đầu hàng từ $t5 vào $t3
col_loop:
    beq $t3, 0, draw_brown  # Kiểm tra nếu $t3 là 0, vẽ màu nâu
    li $t0, GREEN  # Nếu không, set màu xanh lá
    j draw_color

draw_brown:
    li $t0, BROWN  # Set màu nâu

draw_color:
    sw $t0, 0($k0)  # Ghi màu vào địa chỉ hiện tại trên màn hình
    addi $k0, $k0, 4  # Tăng địa chỉ màn hình lên để vẽ pixel tiếp theo
```

```
not $t3, $t3  # Đảo bit của $t3 để thay đổi màu cho lần sau


addi $t2, $t2, -1
bnez $t2, col_loop  # Tiếp tục vòng lặp cột nếu chưa vẽ đủ 8 cột


not $t5, $t5  # Đảo màu bắt đầu hàng cho hàng tiếp theo
addi $t1, $t1, -1
bnez $t1, row_loop  # Tiếp tục vòng lặp hàng nếu chưa vẽ đủ 8 hàng


li $v0, 10  # Kết thúc chương trình
syscall
```

- $k0 chứa địa chỉ của màn hình in
- $t0 chứa mã của màu trắng
- In 8 hàng mỗi hàng 4 ô trắng bằng cách cộng $k0 với 8
- Khi in ô trắng dòng tiếp theo cần lùi vào 1 ô so dòng trước đó cần + thêm 4 vào $k0 trước khi bắt đầu vòng lặp loop2
- Sau khi in xong dòng lùi vào, cần $k0 – 4 để bắt đầu in ngay từ ô đầu tiên
- ***Chạy thử nghiệm***



Assignment 5:

Code:

```
.data
x1: .asciiz "Nhap x1: "
y1: .asciiz "Nhap y1: "
x2: .asciiz "Nhap x2: "
y2: .asciiz "Nhap y2: "
error1: .asciiz "Error: x2 phai khac x1. Moi nhap lai!\n"
error2: .asciiz "Error: y2 phai khac y1. Moi nhap lai!\n"

.eqv MONITOR_SCREEN 0x10010000
.eqv RED 0x00FF0000
.eqv GREEN 0x0000FF00

.text
main:
    # Nhập x1
    li $v0, 4
    la $a0, x1
    syscall
    li $v0, 5
    syscall
    move $s0, $v0

    # Nhập y1
```

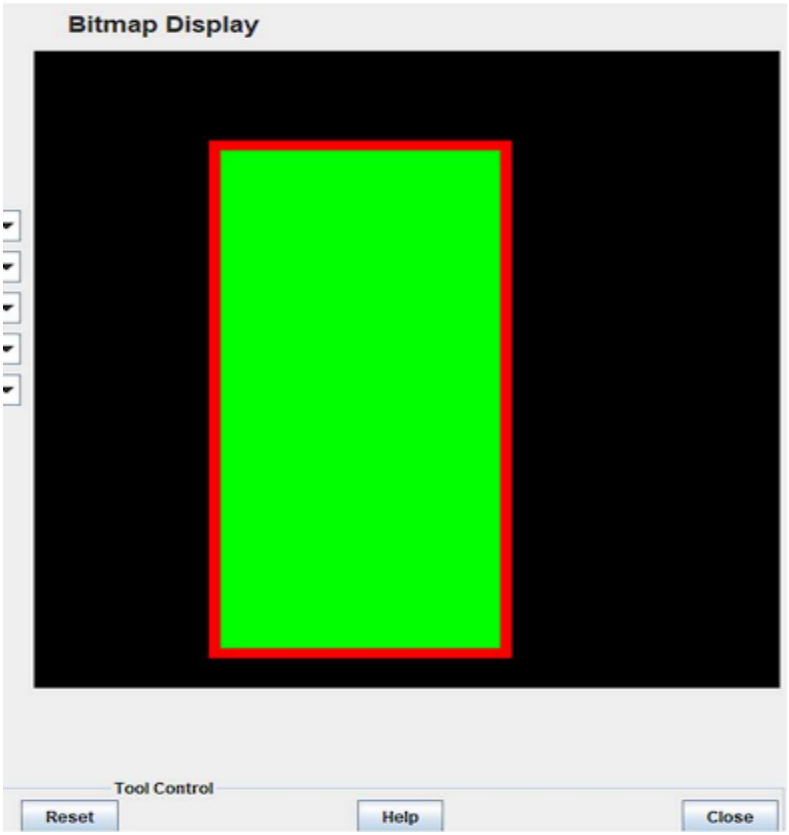
```
    li $v0, 4
    la $a0, y1
    syscall
    li $v0, 5
    syscall
    move $s1, $v0

NhapX2:
    li $v0, 4
    la $a0, x2
    syscall
    li $v0, 5
    syscall
    move $s2, $v0
    beq $s2, $s0, Error1

NhapY2:
    li $v0, 4
    la $a0, y2
    syscall
    li $v0, 5
    syscall
    move $s3, $v0
    beq $s3, $s1, Error2
    j Drawing
```

```
Error1:
    li $v0, 4
    la $a0, error1
    syscall
    j NhapX2

Error2:
    li $v0, 4
    la $a0, error2
    syscall
    j NhapY2

Drawing:
    # Thiết lập bắt đầu vẽ
    li $k0, MONITOR_SCREEN
    slt $t0, $s0, $s2
    slt $t1, $s1, $s3
    beq $t0, 0, ReverseX
    beq $t1, 0, ReverseY
    j DrawRect

ReverseX:
    move $t0, $s0
    move $s0, $s2
```

```
    move $s2, $t0
    j Drawing

ReverseY:
    move $t0, $s1
    move $s1, $s3
    move $s3, $t0
    j Drawing

DrawRect:
    move $a0, $s0
    move $a1, $s1
    move $a2, $s2
    move $a3, $s3
    jal FillRectangle
    j Exit

FillRectangle:
    move $t4, $a1
RectangleLoopY:
    bgt $t4, $a3, Return
    move $t5, $a0
RectangleLoopX:
    bgt $t5, $a2, EndLoopX
    beq $t4, $a1, DrawBorder
```

```
    beq $t4, $a3, DrawBorder
    beq $t5, $a0, DrawBorder
    beq $t5, $a2, DrawBorder
    li $a1, GREEN
    j DrawPixel
DrawBorder:
    li $a1, RED
DrawPixel:
    sll $t8, $t4, 8
    add $t8, $t8, $t5
    sll $t8, $t8, 2
    add $a2, $k0, $t8
    sw $a1, 0($a2)
    addi $t5, $t5, 1
    j RectangleLoopX
EndLoopX:
    addi $t4, $t4, 1
    j RectangleLoopY
Return:
    jr $ra


Exit:
    li $v0, 10
    syscall
```

Chạy thử nghiệm


Bitmap Display
Tool Control
Reset
Help
Close