

## BÁO CÁO THỰC HÀNH

# Laboratory Exercise 10

Học phần: Thực hành Kiến trúc máy tính

Họ tên sinh viên: Nguyễn Thị Hoài Linh

MSSV: 20205231

Lớp: Công nghệ thông tin Việt – Pháp 01 K65

### Assignment 1

Create a new project, type in, and build the program of Home Assignment 1. Show different values on LED.

```

Edit  Execute
as1.asm  as2.asm  as3.asm  as4.asm
#Laboratory Exercise 10, Assignment 1
#Nguyen Thi Hoai Linh - 20205231
.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, 0x4F          # set value for segments
    jal SHOW_7SEG_LEFT    # show
    nop
    li $a0, 0x6           # 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

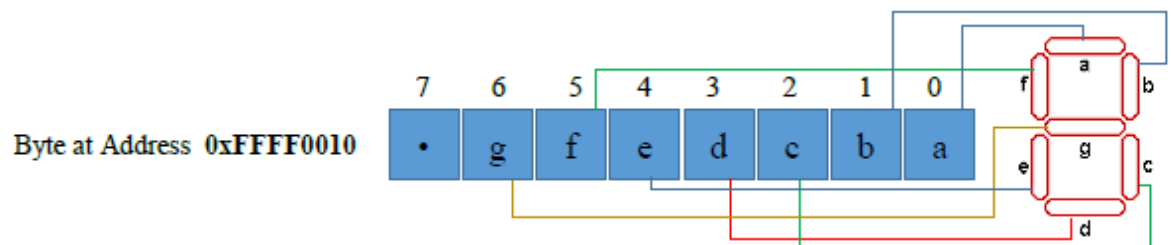
#-----
# Function SHOW_7SEG_RIGHT : turn on/off the 7seg
# param[in]      $a0 value to shown
# remark         $t0 changed
#-----
SHOW_7SEG_RIGHT:
    li $t0, SEVENSEG_RIGHT # assign port's address
    sb $a0, 0($t0) # assign new value
    nop
    jr $ra
    nop

```

Giả sử ta muốn hiển thị hai chữ số cuối trong MSSV 20205231 là 3 và 1 trên hai LED 7 thanh. Ta thực hiện như sau:

1. Xác định giá trị cho các đoạn của LED

- Các đoạn và dấu chấm của LED 7 đoạn tương ứng với các bit theo thứ tự trong một số 8 bit như sau :



Đoạn/Dấu	Bit trọng số
a	0
b	1
c	2
d	3
e	4
f	5
g	6
.	7

- Ta muốn đèn/chấm nào sáng thì cho giá trị bit tương ứng bằng 1 và ngược lại.
- Giả sử không hiện dấu chấm (.), tức bit trọng số lớn nhất bằng 0, ta cần hiển thị:

- Số 3: các đoạn cần sáng là a, b, c, d, g → số nhị phân tương ứng là 01001111 = 0x4F
- Số 1: các đoạn cần sáng là b, c → số nhị phân tương ứng là 00000110 = 0x6
- Như vậy, ta cần gán giá trị cho hai LED là 0x4F và 0x6.
- 2. Truyền giá trị cho LED 7 đoạn vào địa chỉ của nó
  - Ta thực hiện việc này bằng hàm SHOW\_7SEG\_LEFT và SHOW\_7SEG\_RIGHT.
  - Biết địa chỉ của đèn LED 7 đoạn trái là 0xFFFF0011 và đèn LED 7 đoạn phải là 0xFFFF0010. Ta lưu giá trị đã xác định ở trên vào địa chỉ này để bật/tắt từng đoạn như mong muốn.

Ví dụ ở đây ta muốn LED 7 đoạn trái hiển thị số 3:

- Đầu tiên ta xác định giá trị cho từng đoạn và thu được giá trị cần gán cho LED là 0x4F như trên. Ta gán giá trị này vào thanh ghi \$a0: li \$a0, 0x4F
- Ta thực hiện truyền giá trị cho LED 7 đoạn trái bằng hàm SHOW\_7SEG\_LEFT:
  - Ghi địa chỉ của đèn LED 7 đoạn trái là 0xFFFF0011 vào thanh ghi \$t0: li \$t0, SEVENSEG\_LEFT
  - Lưu giá trị 0x4F vào địa chỉ LED 7 đoạn trái: sb \$a0, 0(\$t0)

Chạy lần lượt từng lệnh thực hiện các bước trên, ta thấy giá trị tại địa chỉ 0xffff0011 trong bộ nhớ thay đổi thành 4f và đèn LED 7 đoạn trái sẽ hiển thị số 3 như sau:

The screenshot displays a digital logic simulator interface. The top window shows the MIPS assembly code being executed:

Bkpt	Address	Code	Basic	Source
	0x00400000	0x2404004f	addiu \$4,\$0,0x00000...	10: li \$a0, 0x4F # set value for segments
	0x00400004	0x0c100008	jal 0x00400020	11: jal SHOW 7SEG LEFT # show
	0x00400008	0x00000000	nop	12: nop
	0x0040000c	0x24040006	addiu \$4,\$0,0x00000...	13: li \$a0, 0x6 # set value for segments
	0x00400010	0x0c10000e	jal 0x00400038	14: jal SHOW 7SEG RIGHT # show
	0x00400014	0x00000000	nop	15: nop
	0x00400018	0x2402000a	addiu \$2,\$0,0x00000...	17: li \$v0, 10
	0x0040001c	0x0000000c	syscall	18: syscall
	0x00400020	0x3c01ffff	lui \$1,0xffffffffff	26: li \$t0, 0xFFFF0011 # assign port's address
	0x00400024	0x34280011	ori \$8,\$1,0x00000011	
	0x00400028	0xa1040000	sb \$4,0x00000000(\$8)	27: sb \$a0, 0(\$t0) # assign new value
	0x0040002c	0x00000000	nop	28: nop

The bottom window shows the Data Segment memory values:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0xffff0000	0x00000000	0x00000000	0x00000000	0x00000000	0x00004f00	0x00000000
0xffff0020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xffff0040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xffff0060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xffff0080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xffff00a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xffff00c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xffff00e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xffff0100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0xffff0120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

The Digital Lab Sim window shows a 7-segment display displaying the number 3. The tool control buttons at the bottom include Disconnect from MIPS, Reset, Help, and Close.

Tương tự với LED 7 đoạn phải, ta được kết quả như sau:

The screenshot displays a MIPS assembly simulator interface. The main window is divided into several panels:

- Text Segment:** A table showing assembly instructions with their addresses, codes, basic forms, and source comments. The instructions include `nop`, `addiu $2,$0,0x00000000`, `syscall`, `lui $1,0xffffffff`, `ori $8,$1,0x00000011`, `sb $4,0x00000000($8)`, `jr $31`, and `sb $a0, 0($t0)`.
- Labels:** A table listing labels and their addresses, including `main` at `0x00400000`, `exit` at `0x00400018`, `endmain` at `0x00400020`, `SHOW_TSEG_LEFT` at `0x00400020`, and `SHOW_TSEG_RIGHT` at `0x00400038`.
- Data Segment:** A table showing memory addresses and their values in various offsets (+0, +4, +8, +c, +10, +14, +18, +1c).
- Digital Lab Sim:** A window showing a digital display with the number "8.8" and a keypad with digits 0-9 and letters a-f.
- Mars Messages:** A panel showing messages such as "Reset: reset completed." and "-- program is finished running --".

## Assignment 2

Create a new project, type in, and build the program of Home Assignment 2. Draw something.

Home Assignment 2:

```

1  #Laboratory Exercise 10, Assignment 2
2  #Nguyen Thi Hoai Linh - 20205231
3  .eqv MONITOR_SCREEN 0x10010000  #Dia chi bat dau cua bo nho man hinh
4  .eqv RED 0x00FF0000  #Cac gia tri mau thuong su dung
5  .eqv GREEN 0x0000FF00
6  .eqv BLUE 0x000000FF
7  .eqv WHITE 0x00FFFFFF
8  .eqv YELLOW 0x00FFFF00
9  .text
10     li $k0, MONITOR_SCREEN  #Nap dia chi bat dau cua man hinh
11
12     li $t0, RED
13     sw $t0, 0($k0)
14     nop
15
16     li $t0, GREEN
17     sw $t0, 4($k0)
18     nop
19
20     li $t0, BLUE
21     sw $t0, 8($k0)
22     nop
23
24     li $t0, WHITE
25     sw $t0, 12($k0)
26     nop
27
28     li $t0, YELLOW
29     sw $t0, 32($k0)
30     nop
31
32     li $t0, WHITE
33     lb $t0, 42($k0)
34     nop

```

Bitmap Display, Version 1.0

**Bitmap Display**


Unit Width in Pixels: 32

Unit Height in Pixels: 32

Display Width in Pixels: 256

Display Height in Pixels: 256

Base address for display: 0x10010000 (static data)



Tool Control: Disconnect from MIPS, Reset, Help, Close

Line: 3 Column: 11 ☒ Show Line Numbers

Mars Messages: Run I/O

Reset: reset completed.

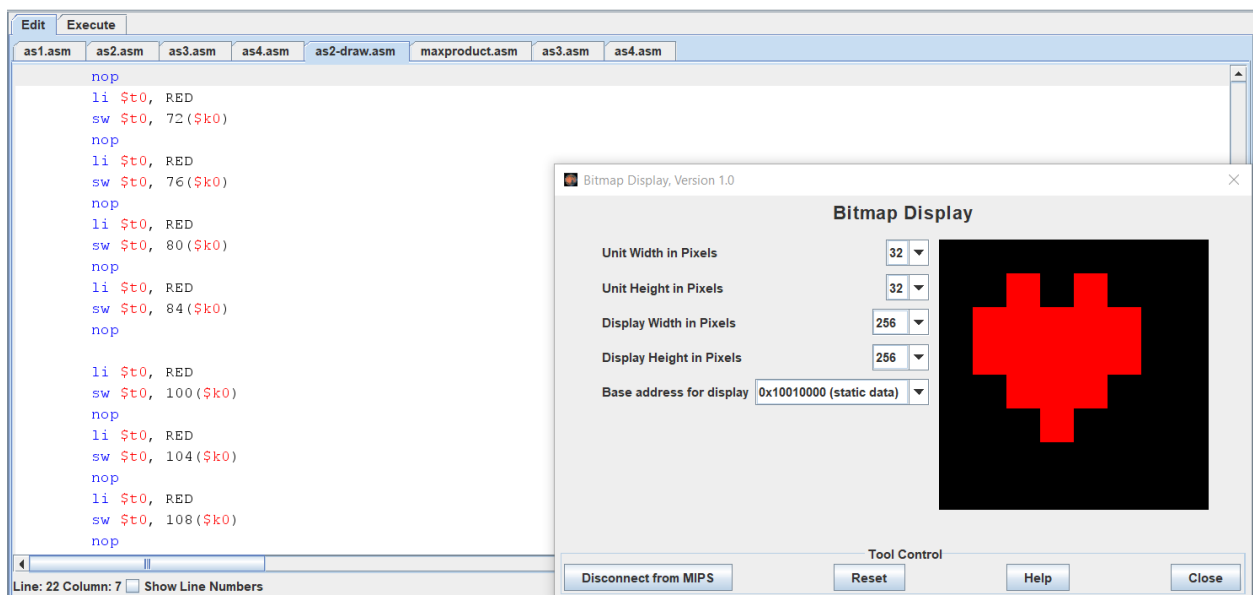
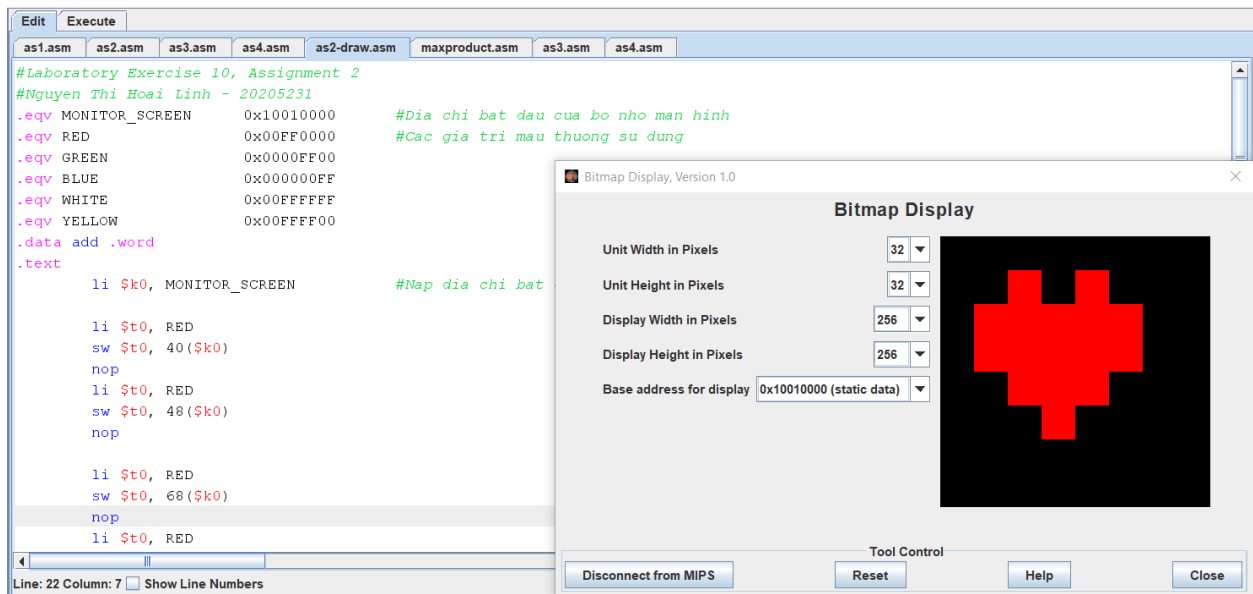
Clear

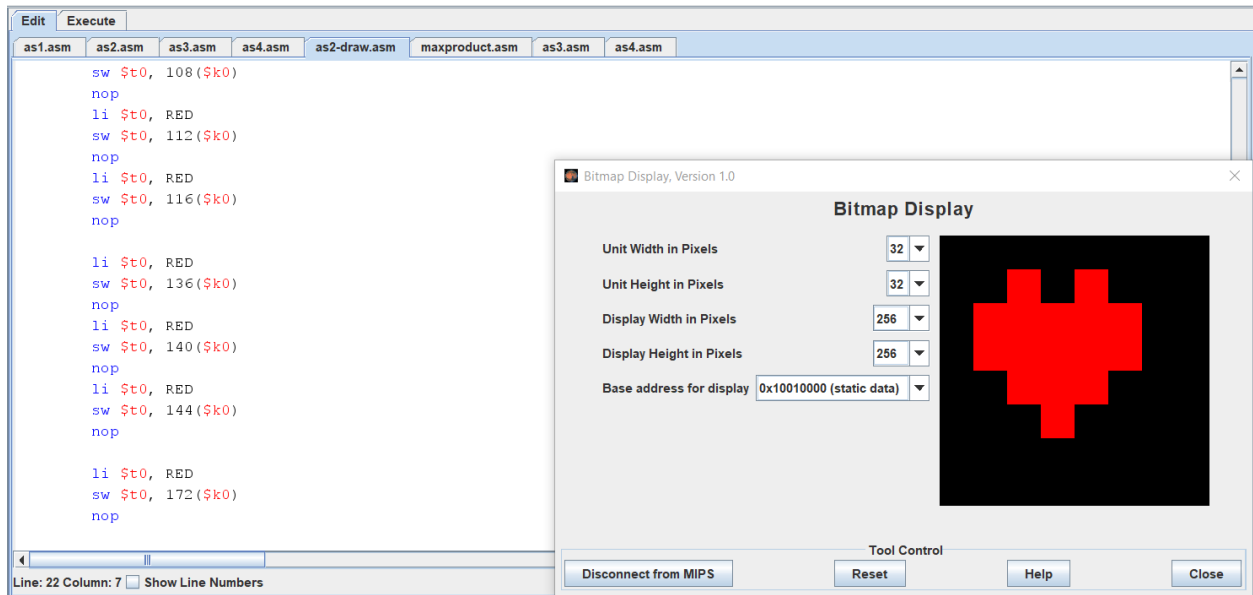
-- program is finished running

Để vẽ trên Bitmap, ta lưu các giá trị màu vào bộ nhớ màn hình tương ứng với từng pixel, biết trước địa chỉ cơ sở của màn hình (ở bài này là 0x10010000). Mỗi pixel bitmap tương ứng với 4 bytes trong bộ nhớ. Muốn thay đổi màu cho từng pixel, ta thay đổi giá trị trong bộ nhớ theo giá trị màu ta muốn. Ví dụ ta muốn pixel đầu tiên ở góc trên bên trái có màu đỏ, ta thực hiện như sau:

- li \$t0, RED : lưu giá trị màu đỏ vào thanh ghi \$t0
- sw \$t0, 0(\$k0) : lưu giá trị màu đỏ từ thanh ghi \$t0 vào địa chỉ pixel được chứa trong thanh ghi \$k0

Áp dụng cách trên, ta có thể vẽ một hình trái tim như sau:





### Assignment 3

Create a new project, type in, and build the program of Home Assignment 3. Make the Bot run and draw a triangle by tracking.

### Assignment 4

Create a new project, type in, and build the program of Home Assignment 4. Read key char and terminate the application when receiving "exit" command.