**BÁO CÁO THỰC HÀNH**

**KIẾN TRÚC MÁY TÍNH LAB**

**Assignment 4:**

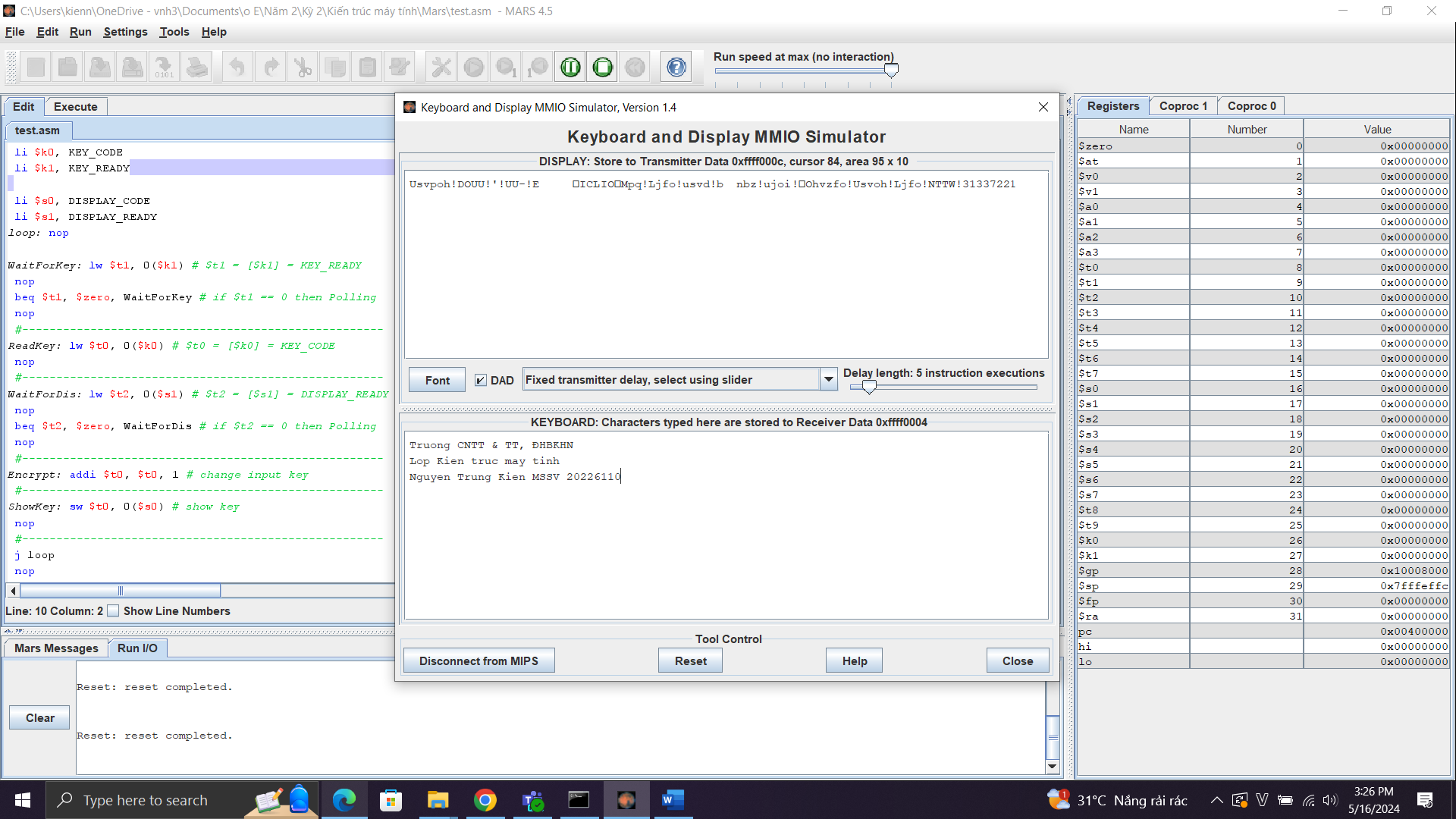
**Code:**

|  |
| --- |
| .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  .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, 1 # change input key  #-----------------------------------------------------  ShowKey: sw $t0, 0($s0) # show key  nop  #-----------------------------------------------------  j loop  nop |

**Nhận xét:**

* Sử dụng Memory-Mapped I/O (MMIO) cho thiết bị nhập từ bàn phím và thiết bị hiển thị ký tự. Trong khi chương trình MIPS đang được kết nối, mỗi lần nhấn phím sẽ được mã hóa thành mã ASCII tương ứng và được đặt vào thanh ghi Receiver Data.
* Vòng lặp chính kiểm tra trạng thái sẵn sàng của dữ liệu từ bàn phím, đọc dữ liệu đó, và sau đó chuyển đến xử lý việc hiển thị.

**Kết quả:**



**Assignment 5:**

**Code:**

|  |
| --- |
| .data  .eqv SEVENSEG\_LEFT, 0xFFFF0010 # Địa chỉ của đèn LED 7 đoạn bên trái  .eqv SEVENSEG\_RIGHT, 0xFFFF0011 # Địa chỉ của đèn LED 7 đoạn bên phải  # Mảng các giá trị segment cho từng số từ 0 đến 9  digitValues: .byte 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F  digitValues2: .byte 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F  .text  .globl main  main:  li $t0, 0 # Khởi tạo biến đếm cho LED trái (0 đến 9)  li $t1, 9 # Khởi tạo biến đếm cho LED phải (9 về 0)  li $t4, 0x80 # Trạng thái ban đầu cho dấu chấm nhấp nháy  loop:  la $t2, digitValues # Địa chỉ của mảng giá trị segment cho LED trái  add $t3, $t2, $t0 # Địa chỉ của giá trị hiện tại cho LED trái  lb $a0, 0($t3) # Load giá trị segment cho LED trái  xor $a0, $a0, $t4 # Nhấp nháy dấu chấm  jal SHOW\_7SEG\_LEFT # Hiển thị giá trị trên LED trái  xori $t4, $t4, 0x80 # Đảo trạng thái nhấp nháy    la $t2, digitValues2 # Địa chỉ của mảng giá trị segment cho LED phải  add $t3, $t2, $t1 # Địa chỉ của giá trị hiện tại cho LED phải  lb $a0, 0($t3) # Load giá trị segment cho LED phải  jal SHOW\_7SEG\_RIGHT # Hiển thị giá trị trên LED phải  jal DELAY # Thực hiện trễ thời gian  addi $t0, $t0, 1 # Tăng giá trị đèn trái  addi $t1, $t1, -1 # Giảm giá trị đèn phải  bge $t0, 10, reset\_left # Kiểm tra và đặt lại giá trị nếu cần  bge $t1, 0, loop # Nếu $t1 >= 0, tiếp tục lặp  reset\_left:  li $t0, 0 # Đặt lại giá trị đèn trái  li $t1, 9 # Đặt lại giá trị đèn phải  j loop # Quay lại vòng lặp  exit:  li $v0, 10  syscall # Kết thúc chương trình  #---------------------------------------------------------------  # Function: SHOW\_7SEG\_LEFT  # Mô tả: Hiển thị giá trị trên đèn LED 7 đoạn bên trái  #---------------------------------------------------------------  SHOW\_7SEG\_LEFT:  li $t5, SEVENSEG\_LEFT  sb $a0, 0($t5) # Gửi giá trị tới địa chỉ đèn LED  jr $ra # Trở về chương trình chính  #---------------------------------------------------------------  # Function: SHOW\_7SEG\_RIGHT  # Mô tả: Hiển thị giá trị trên đèn LED 7 đoạn bên phải  #---------------------------------------------------------------  SHOW\_7SEG\_RIGHT:  li $t5, SEVENSEG\_RIGHT  sb $a0, 0($t5) # Gửi giá trị tới địa chỉ đèn LED  jr $ra # Trở về chương trình chính  #---------------------------------------------------------------  # Function: DELAY  # Mô tả: Tạo trễ thời gian  #---------------------------------------------------------------  DELAY:  li $t3, 200000 # Đặt số lần lặp cho vòng lặp trễ  delay\_loop:  addi $t3, $t3, -1 # Giảm biến đếm  bnez $t3, delay\_loop # Nếu $t3 không bằng 0, tiếp tục lặp  jr $ra # Trở về chương trình chính |

**Nhận xét:**

Cấu Trúc Chương Trình Phần Khai Báo:

+ Địa chỉ các đèn LED: Chương trình bắt đầu với việc khai báo địa chỉ bộ nhớ cho đèn LED 7 đoạn bên trái và bên phải. Đây là thông tin cần thiết để có thể gửi dữ liệu tới các đèn LED từ bộ xử lý.

+ Mảng Giá Trị Segment: Tiếp theo, chương trình khai báo hai mảng chứa giá trị cho từng số từ 0 đến 9 dựa trên cách các segment được bật tắt. Mỗi byte trong mảng tương ứng với một cách hiển thị số trên đèn LED.

+ Hàm Main:

* Khởi Tạo Biến: Được sử dụng để thiết lập các giá trị ban đầu cho các biến đếm, và trạng thái nhấp nháy cho dấu chấm.
* Vòng Lặp Chính: Chứa logic để hiển thị số lên đèn LED dựa trên giá trị trong mảng đã khai báo. Đồng thời, cập nhật giá trị đếm tăng cho đèn trái và giảm cho đèn phải.
* Điều Kiện Kiểm Tra: Đảm bảo rằng khi đến số cuối cùng của chuỗi đếm, biến đếm được đặt lại để tiếp tục vòng lặp.

+ Các Hàm Hỗ Trợ:

* SHOW\_7SEG\_LEFT và SHOW\_7SEG\_RIGHT: Những hàm này dùng để gửi giá trị từ bộ xử lý tới đèn LED tương ứng.
* Hàm nhận giá trị segment từ mảng và gửi nó tới địa chỉ bộ nhớ của đèn LED.
* DELAY: Hàm này tạo một độ trễ trong chương trình, cho phép người dùng quan sát sự thay đổi số trên đèn LED. Độ trễ được thực hiện thông qua một vòng lặp giảm dần.

**Kết quả:**

Link video kết quả chương trình: <https://drive.google.com/file/d/1z46KoGr7fYVTS72Oqk7sWeQexpSDnEzK/view?usp=sharing>

(do làm ko cẩn thận nên em nhầm dot sáng là dot bên trái. Em đã sửa lại ở phần code nhưng chưa kịp sửa video và ảnh, mong thầy & anh thông cảm ạ).

