**BÁO CÁO THỰC HÀNH**

**KIẾN TRÚC MÁY TÍNH LAB 11**

**Assignment 1:**

**Code:**

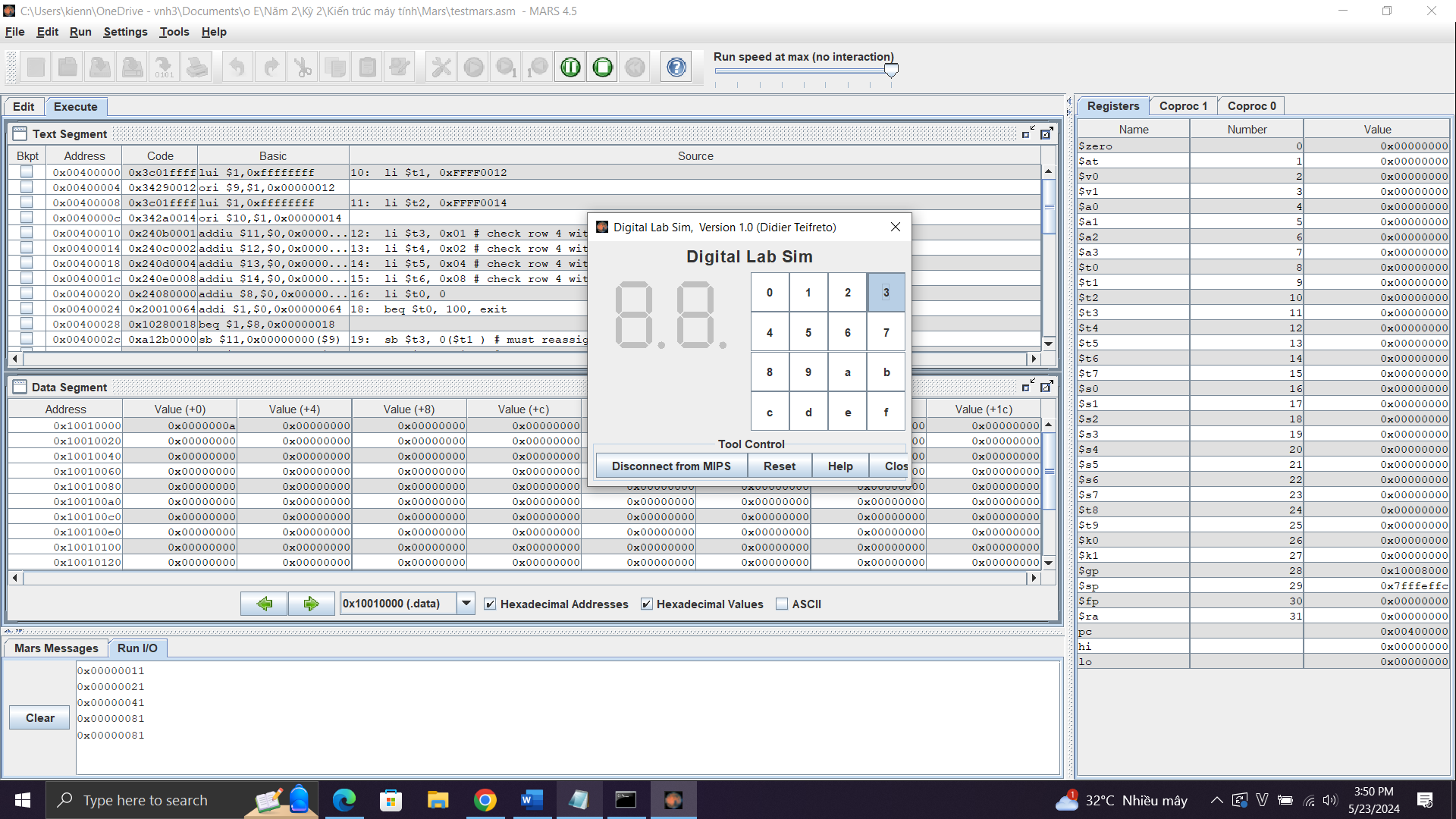
|  |
| --- |
| .eqv IN\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0012  .eqv OUT\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0014  .data  nl: .asciiz "\n"  .text  main:  li $t1, IN\_ADRESS\_HEXA\_KEYBOARD  li $t2, OUT\_ADRESS\_HEXA\_KEYBOARD  li $t3, 0x01 # check row 4 with key C, D,E, F  li $t4, 0x02 # check row 4 with key C, D,E, F  li $t5, 0x04 # check row 4 with key C, D,E, F  li $t6, 0x08 # check row 4 with key C, D,E, F  li $t0, 0  polling:  beq $t0, 100, exit  sb $t3, 0($t1 ) # must reassign expected row  lbu $a0, 0($t2) # read scan code of key button  bne $a0, $zero, print  sb $t4, 0($t1 ) # must reassign expected row  lbu $a0, 0($t2) # read scan code of key button  bne $a0, $zero, print  sb $t5, 0($t1 ) # must reassign expected row  lbu $a0, 0($t2) # read scan code of key button  bne $a0, $zero, print  sb $t6, 0($t1 ) # must reassign expected row  lbu $a0, 0($t2) # read scan code of key button  bne $a0, $zero, print    j continue  print:  li $v0, 34 # print integer (hexa)  syscall  la $a0, nl  li $v0, 4  syscall  continue:  addi $t0, $t0, 1  sleep:  li $a0, 1000 # sleep  li $v0, 32  syscall  back\_to\_polling:  j polling # continue polling  exit: |

**Nhận xét:**

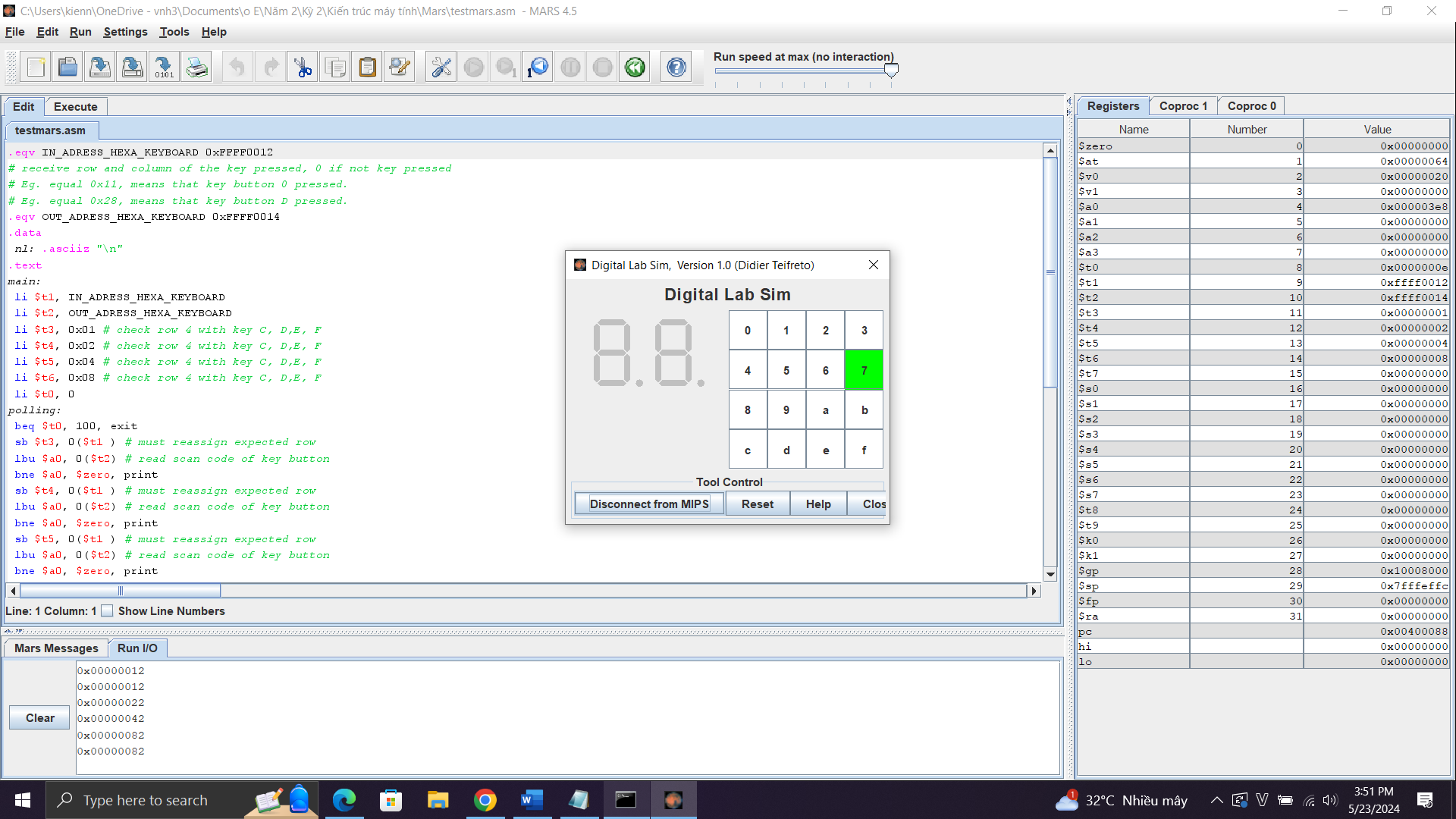
* Chương trình này được viết để kiểm tra một bàn phím hexa sử dụng phương thức polling. Đây là một cách thức để liên tục kiểm tra trạng thái của một thiết bị ngoại vi mà không cần ngắt.
* Dùng lệnh “lbu” thay cho “lb” để tránh việc khi mở rộng dấu làm sai lệch kết quả số ở cột thứ 4.
* Ta thu được kết quả là 16 số hexa ở dưới cùng tương ứng với 16 phím được bấm.

**Kết quả:**  (vì thực hiện chương trình khá lag nên em chia nhỏ bấm từng dòng)

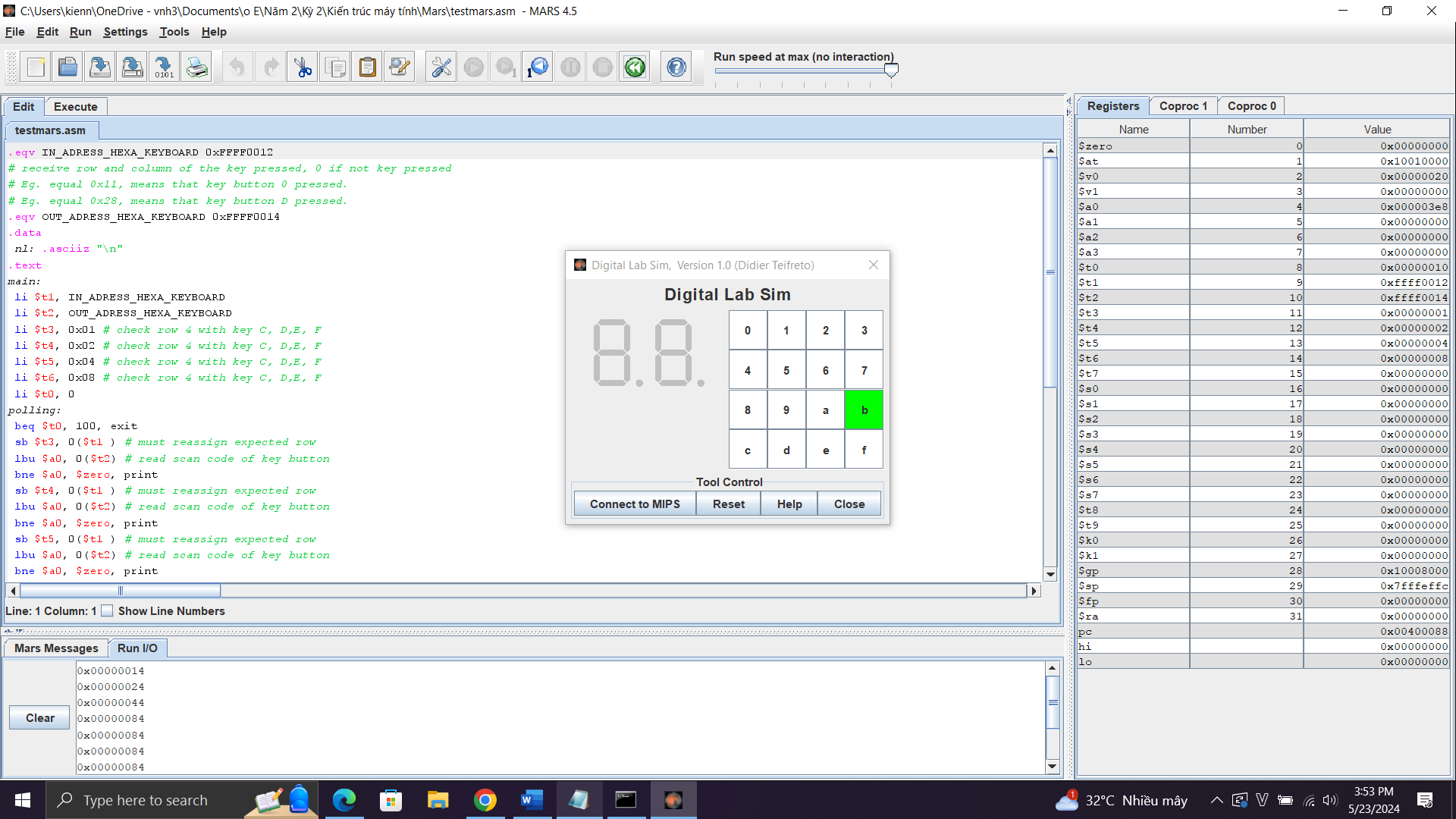
Dòng 1:



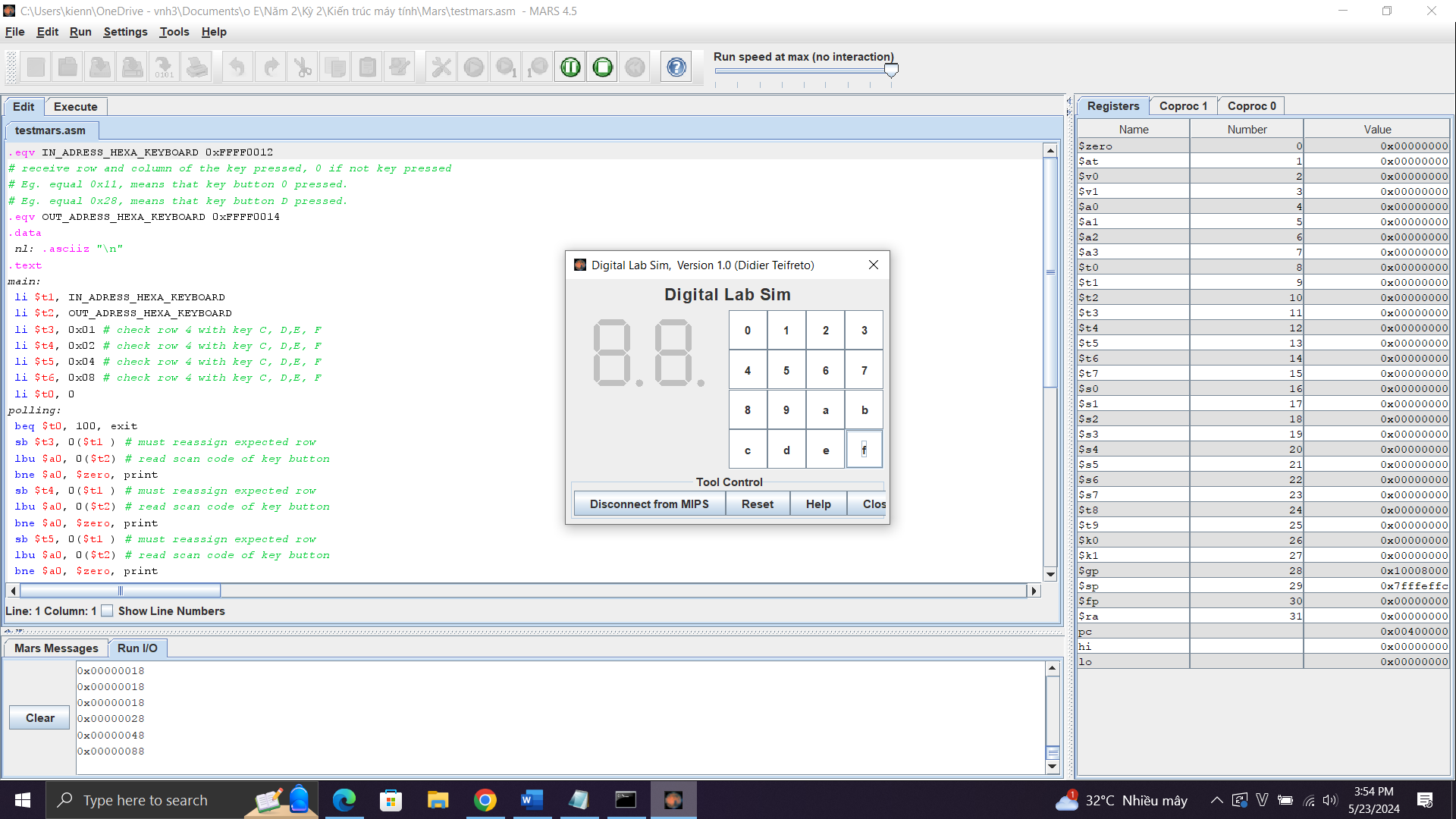
Dòng 2:



Dòng 3:



Dòng 4:



**Assignment 2:**

**Code:**

|  |
| --- |
| .eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012  .data  Message: .asciiz "Oh my god. Someone's presed a button.\n"  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  # MAIN Procedure  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  .text  main:  #---------------------------------------------------------  # Enable interrupts you expect  #---------------------------------------------------------  # Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim  li $t1, IN\_ADDRESS\_HEXA\_KEYBOARD  li $t3, 0x80 # bit 7 of = 1 to enable interrupt  sb $t3, 0($t1)  #---------------------------------------------------------  # No-end loop, main program, to demo the effective of interrupt  #---------------------------------------------------------  Loop: nop  nop  nop  nop  b Loop # Wait for interrupt  end\_main:  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  # GENERAL INTERRUPT SERVED ROUTINE for all interrupts  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  .ktext 0x80000180  #--------------------------------------------------------  # Processing  #--------------------------------------------------------  IntSR: addi $v0, $zero, 4 # show message  la $a0, Message  syscall  #--------------------------------------------------------  # Evaluate the return address of main routine  # epc <= epc + 4  #--------------------------------------------------------  next\_pc:mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc  addi $at, $at, 4 # $at = $at + 4 (next instruction)  mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at  return: eret # Return from exception |

**Nhận xét:**

* Cài đặt và Khởi động:

+ Đầu tiên, mã thiết lập địa chỉ nhớ cho bàn phím và một chuỗi thông báo, được sử dụng để thông báo khi có ngắt phím.

+ Chương trình tiếp tục bằng cách ghi giá trị vào địa chỉ đã định nghĩa để kích hoạt ngắt từ bàn phím.

* Vòng Lặp Chính:

+ Chương trình đi vào một vòng lặp vô hạn, nơi nó chỉ thực hiện các lệnh nop (no operation). Mục đích của vòng lặp này là để giữ cho chương trình chạy và chờ đợi một ngắt xảy ra.

* Xử Lý Ngắt:

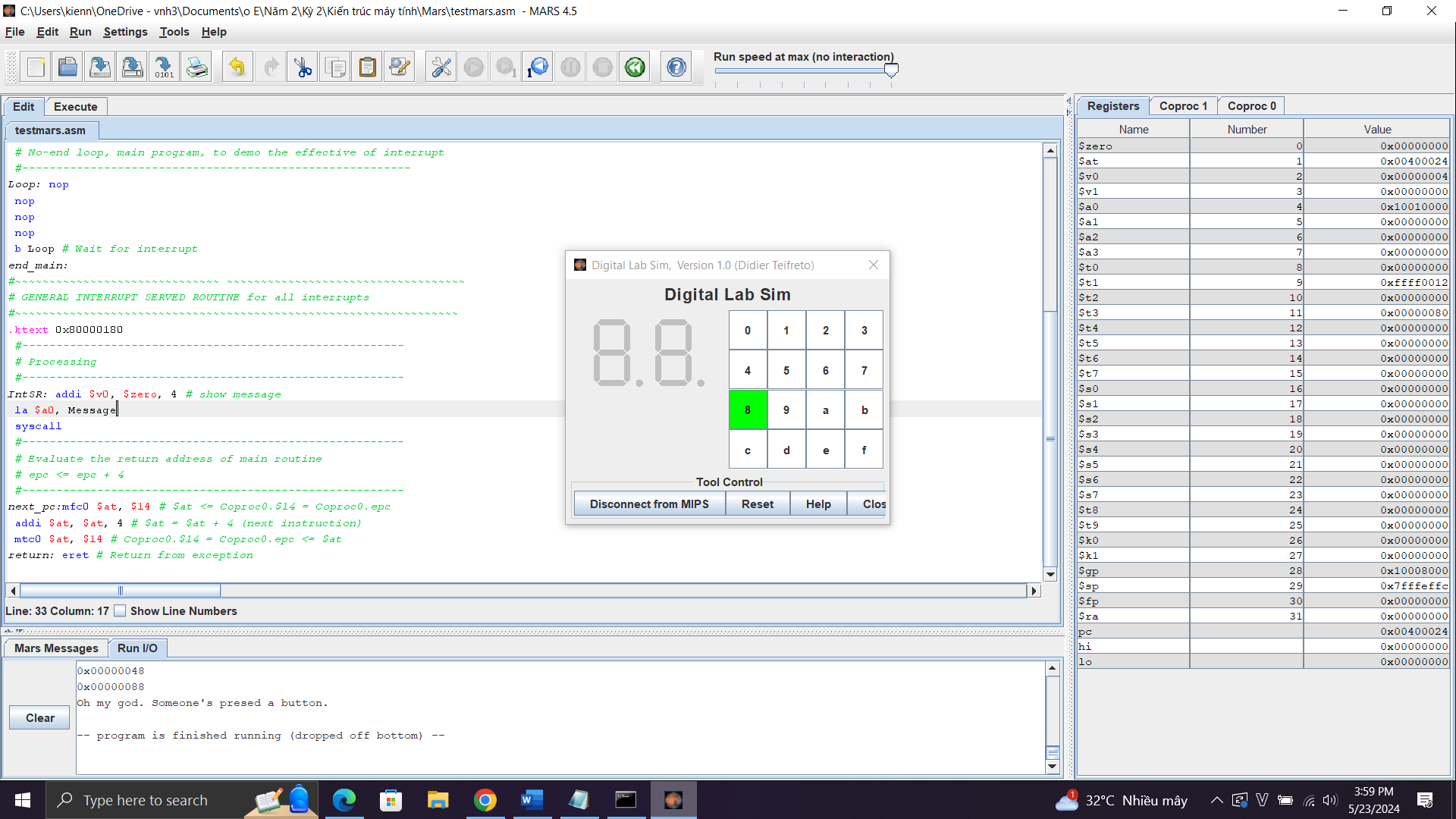
+ Khi có ngắt xảy ra, chương trình sẽ nhảy tới trình xử lý ngắt được định tại địa chỉ 0x80000180.

+Trong trình xử lý ngắt, chương trình in ra thông điệp đã được khởi tạo ở phần đầu. Điều này được thực hiện thông qua một lệnh syscall để hiển thị chuỗi trên màn hình.

+ Sau đó, mã cập nhật giá trị của bộ đếm chương trình (Program Counter - PC) để đảm bảo rằng khi trở về từ ngắt, chương trình sẽ tiếp tục tại lệnh tiếp theo chứ không phải lặp lại lệnh đã xử lý ngắt.

* Kết thúc Ngắt: Sau khi cập nhật PC, lệnh eret (exception return) được sử dụng để kết thúc xử lý ngắt và trả lại điều khiển cho chương trình chính.

**Kết quả:**



**Assignment 3:**

**Code:**

|  |
| --- |
| .eqv IN\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0012  .eqv OUT\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0014  .data  Message: .asciiz "Key scan code "  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  # MAIN Procedure  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  .text  main:  #---------------------------------------------------------  # Enable interrupts you expect  #---------------------------------------------------------  # Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim  li $t1, IN\_ADRESS\_HEXA\_KEYBOARD  li $t3, 0x80 # bit 7 = 1 to enable  sb $t3, 0($t1)  #---------------------------------------------------------  # Loop an print sequence numbers  #---------------------------------------------------------  xor $s0, $s0, $s0 # count = $s0 = 0  Loop:  addi $s0, $s0, 1 # count = count + 1  prn\_seq:  addi $v0,$zero,1  add $a0,$s0,$zero # print auto sequence number  syscall  prn\_eol:  addi $v0,$zero,11  li $a0,'\n' # print endofline  syscall  sleep:  addi $v0,$zero,32  li $a0,500 # sleep 0,5 s  syscall  nop # WARNING: nop is mandatory here.  b Loop # Loop  end\_main:  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  # GENERAL INTERRUPT SERVED ROUTINE for all interrupts  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  .ktext 0x80000180  #-------------------------------------------------------  # SAVE the current REG FILE to stack  #-------------------------------------------------------  IntSR:  addi $sp,$sp,4 # Save $ra because we may change it later  sw $ra,0($sp)  addi $sp,$sp,4 # Save $at because we may change it later  sw $at,0($sp)  addi $sp,$sp,4 # Save $sp because we may change it later  sw $v0,0($sp)  addi $sp,$sp,4 # Save $a0 because we may change it later  sw $a0,0($sp)  addi $sp,$sp,4 # Save $t1 because we may change it later  sw $t1,0($sp)  addi $sp,$sp,4 # Save $t3 because we may change it later  sw $t3,0($sp)  #--------------------------------------------------------  # Processing  #--------------------------------------------------------  prn\_msg:  addi $v0, $zero, 4  la $a0, Message  syscall  get\_cod:  li $t1, IN\_ADRESS\_HEXA\_KEYBOARD  li $t3, 0x81 # check row 4 and re-enable bit 7  sb $t3, 0($t1) # must reassign expected row  li $t1, OUT\_ADRESS\_HEXA\_KEYBOARD  lbu $a0, 0($t1)  bne $a0, $zero, prn\_cod    li $t1, IN\_ADRESS\_HEXA\_KEYBOARD  li $t3, 0x82 # check row 4 and re-enable bit 7  sb $t3, 0($t1) # must reassign expected row  li $t1, OUT\_ADRESS\_HEXA\_KEYBOARD  lbu $a0, 0($t1)  bne $a0, $zero, prn\_cod    li $t1, IN\_ADRESS\_HEXA\_KEYBOARD  li $t3, 0x84 # check row 4 and re-enable bit 7  sb $t3, 0($t1) # must reassign expected row  li $t1, OUT\_ADRESS\_HEXA\_KEYBOARD  lbu $a0, 0($t1)  bne $a0, $zero, prn\_cod  li $t1, IN\_ADRESS\_HEXA\_KEYBOARD  li $t3, 0x88 # check row 4 and re-enable bit 7  sb $t3, 0($t1) # must reassign expected row  li $t1, OUT\_ADRESS\_HEXA\_KEYBOARD  lbu $a0, 0($t1)  bne $a0, $zero, prn\_cod    prn\_cod:  li $v0,34  syscall  li $v0,11  li $a0,'\n' # print endofline  syscall  #--------------------------------------------------------  # Evaluate the return address of main routine  # epc <= epc + 4  #--------------------------------------------------------  next\_pc:  mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc  addi $at, $at, 4 # $at = $at + 4 (next instruction)  mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at  #--------------------------------------------------------  # RESTORE the REG FILE from STACK  #--------------------------------------------------------  restore:  lw $t3, 0($sp) # Restore the registers from stack  addi $sp,$sp,-4  lw $t1, 0($sp) # Restore the registers from stack  addi $sp,$sp,-4  lw $a0, 0($sp) # Restore the registers from stack  addi $sp,$sp,-4  lw $v0, 0($sp) # Restore the registers from stack  addi $sp,$sp,-4  lw $ra, 0($sp) # Restore the registers from stack  addi $sp,$sp,-4  lw $ra, 0($sp) # Restore the registers from stack  addi $sp,$sp,-4  return:  eret # Return from exception |

**Nhận xét:**

* Chương trình in Dãy Số Tuần Tự: Trong chương trình chính, một số được in ra màn hình theo một chu kỳ liên tục. Đây là một phần của vòng lặp chính của chương trình, nơi mỗi lần lặp tăng giá trị số lên và sau đó in số đó ra. Điều này sẽ xảy ra liên tục cho đến khi có sự kiện ngắt.
* Ngắt và In Mã Quét Phím: Khi bạn nhấn một trong các phím C, D, E, hoặc F, chương trình sẽ kích hoạt một ngắt. Trong trình xử lý ngắt, chương trình sẽ đọc mã quét của phím đã được nhấn từ bàn phím và in mã đó ra màn hình. Sau khi in mã quét phím, chương trình trở lại vòng lặp chính và tiếp tục in số tuần tự.

**Kết quả:** Nhập lần lượt các số 2, 5, 0, 1 (ngày sinh Nguyễn Trung Kiên) ta thu được kết quả sau:

