**Bài 11. LẬP TRÌNH XỬ LÝ NGẮT**

**Họ và tên:** Nguyễn Thành Duy

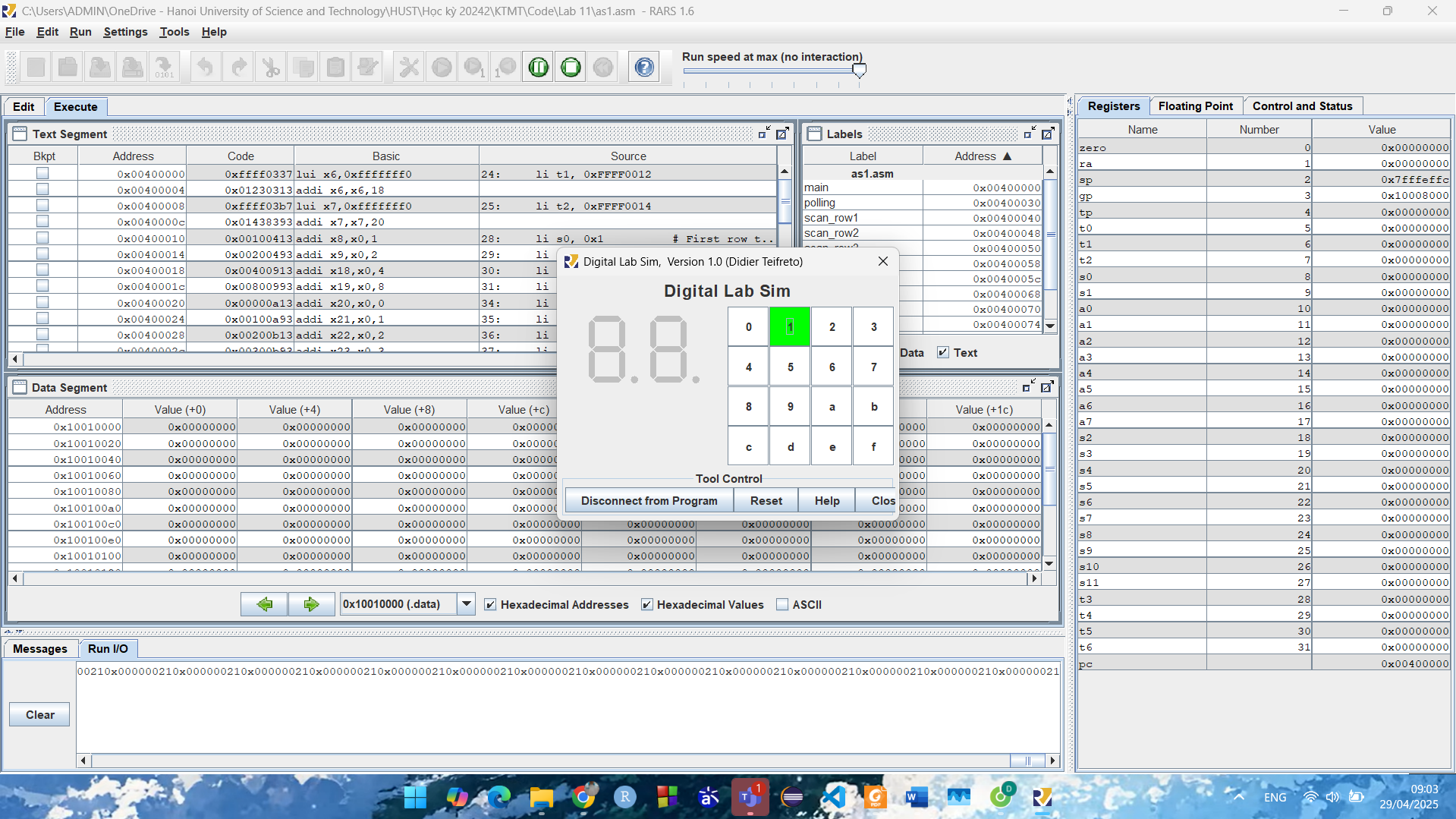
**MSSV:** 20235696

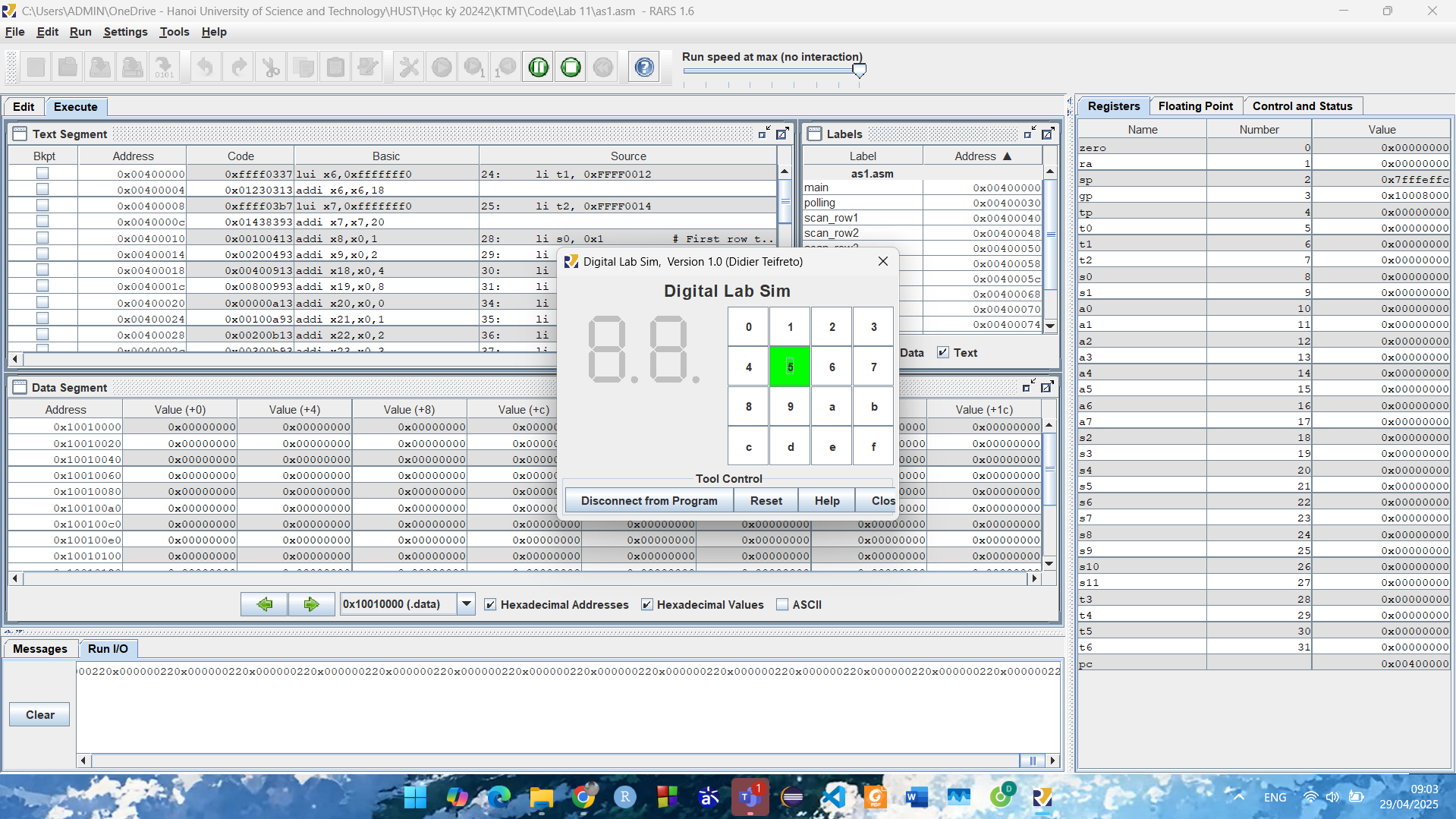
**Assignment 1**

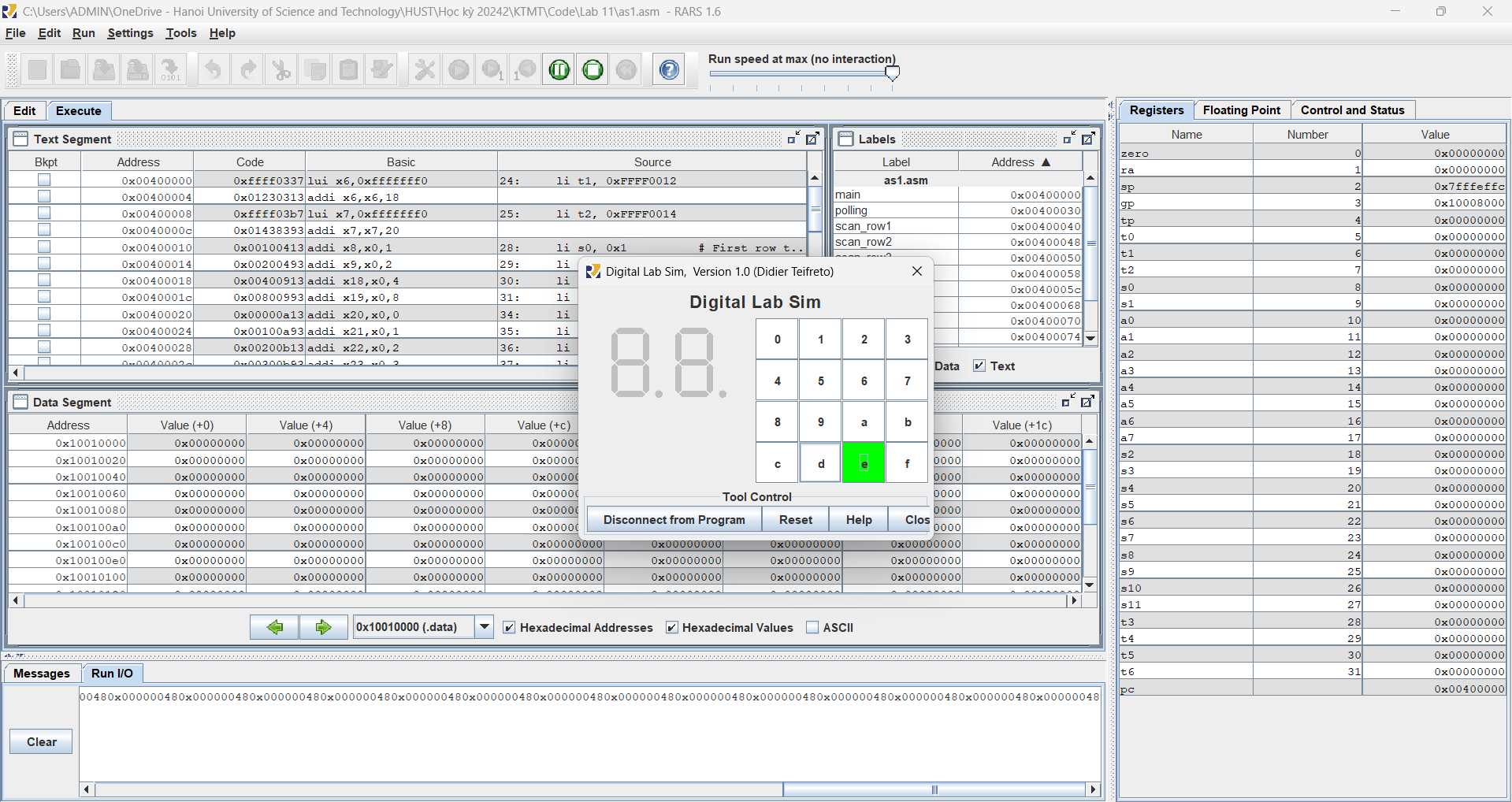
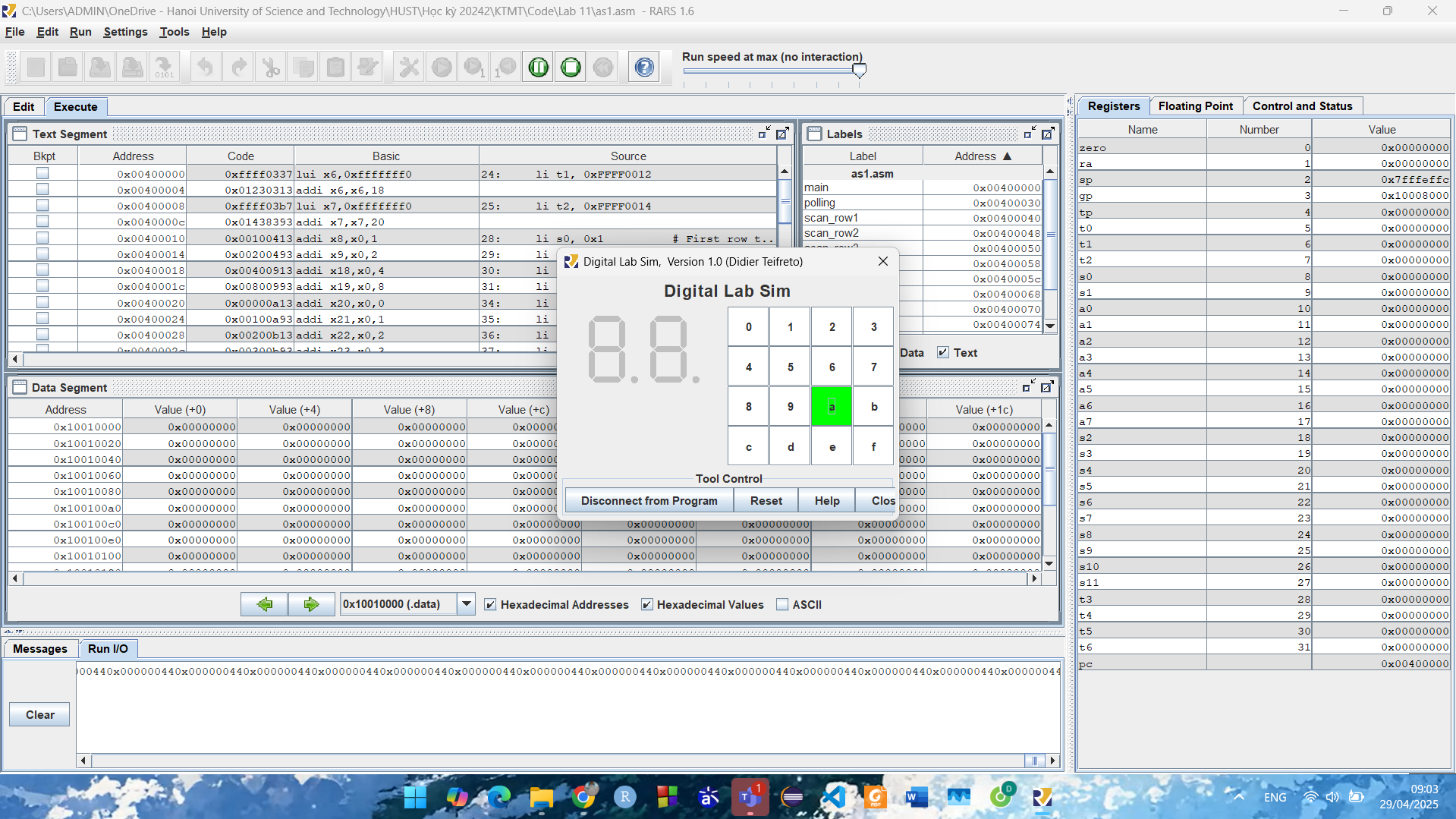
**Code:**

|  |
| --- |
| # ------------------------------------------------------  # col 0x1 col 0x2 col 0x4 col 0x8  # row 0x1 0 1 2 3  # 0x11 0x21 0x41 0x81  # row 0x2 4 5 6 7  # 0x12 0x22 0x42 0x82  # row 0x4 8 9 a b  # 0x14 0x24 0x44 0x84  # row 0x8 c d e f  # 0x18 0x28 0x48 0x88  # ------------------------------------------------------  # Command row number of hexadecimal keyboard (bit 0 to 3)  # Eg. assign 0x1, to get key button 0,1,2,3  # assign 0x2, to get key button 4,5,6,7  # NOTE must reassign value for this address before reading,  # eventhough you only want to scan 1 row  .eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012  # Receive row and column of the key pressed, 0 if not key pressed  # Eg. equal 0x11, means that key button 0 pressed.  # Eg. equal 0x28, means that key button D pressed.  .eqv OUT\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014  .text  main:  li t1, IN\_ADDRESS\_HEXA\_KEYBOARD  li t2, OUT\_ADDRESS\_HEXA\_KEYBOARD  # Initialize row values to scan (0x1, 0x2, 0x4, 0x8)  li s0, 0x1 # First row to scan  li s1, 0x2 # Second row to scan  li s2, 0x4 # Third row to scan  li s3, 0x8 # Fourth row to scan  # Initialize counter for row scanning  li s4, 0 # Current row index (0-3)  li s5, 1 # Constant 1 for comparison  li s6, 2 # Constant 2 for comparison  li s7, 3 # Constant 3 for comparison  polling:  # Select which row to scan based on s4  beq s4, zero, scan\_row1  beq s4, s5, scan\_row2  beq s4, s6, scan\_row3  beq s4, s7, scan\_row4  scan\_row1:  sb s0, 0(t1) # Scan row 1  j read\_key  scan\_row2:  sb s1, 0(t1) # Scan row 2  j read\_key  scan\_row3:  sb s2, 0(t1) # Scan row 3  j read\_key  scan\_row4:  sb s3, 0(t1) # Scan row 4  read\_key:  lb t3, 0(t2) # Read scan code of key button into t3  li t4, 0  beq t3, t4, no\_new\_key # If no key pressed, keep old key  new\_key\_pressed:  mv s8, t3 # Update last key pressed  j print\_key  no\_new\_key:  # Do not update s8, keep old value  nop  print\_key:  mv a0, s8 # Move last valid key to a0  li a7, 34 # Print integer (hexadecimal)  ecall  next\_row:  addi s4, s4, 1 # Increment row counter  li t0, 4  beq s4, t0, reset\_counter  sleep:  li a0, 100 # Sleep 100ms  li a7, 32  ecall  j polling  reset\_counter:  li s4, 0 # Reset row counter  j polling |

Kết quả khi chạy:







Giải thích:

* Bàn phím 16 phím (4 hàng x 4 cột).
* Mỗi lần chỉ quét **1 hàng**.
* Nếu **có phím nhấn**: ghi nhận và in ra phím đó.
* Nếu **không có phím nhấn**: vẫn **in lại** phím **cuối cùng đã nhấn**

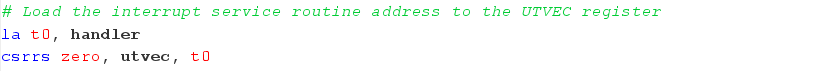
**Assignment 2**

**Code:**

|  |
| --- |
| .eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012  .data  message: .asciz "Someone's presed a button.\n"  # -----------------------------------------------------------------  # MAIN Procedure  # -----------------------------------------------------------------  .text  main:  # Load the interrupt service routine address to the UTVEC register  la t0, handler  csrrs zero, utvec, t0  # Set the UEIE (User External Interrupt Enable) bit in UIE register  li t1, 0x100  csrrs zero, uie, t1 # uie - ueie bit (bit 8)  # Set the UIE (User Interrupt Enable) bit in USTATUS register  csrrsi zero, ustatus, 1 # ustatus - enable uie (bit 0)  # Enable the interrupt of keypad of Digital Lab Sim  li t1, IN\_ADDRESS\_HEXA\_KEYBOARD  li t3, 0x80 # bit 7 = 1 to enable interrupt  sb t3, 0(t1)  # ---------------------------------------------------------  # No-end loop, main program, to demo the effective of interrupt  # ---------------------------------------------------------  loop:  nop  nop  nop  j loop  end\_main:  # -----------------------------------------------------------------  # Interrupt service routine  # -----------------------------------------------------------------  handler:  # ebreak # Can pause the execution to observe registers  # Saves the context  addi sp, sp, -8  sw a0, 0(sp)  sw a7, 4(sp)  # Handles the interrupt  # Shows message in Run I/O  li a7, 4  la a0, message  ecall  # Restores the context  lw a7, 4(sp)  lw a0, 0(sp)  addi sp, sp, 8  # Back to the main procedure  uret |

Giải thích:

* Đoạn code:



**la t0, handler**: nạp địa chỉ nhãn handler vào thanh ghi t0.

**csrrs zero, utvec, t0**:

* Ghi địa chỉ đó vào **thanh ghi CSR utvec**.
* utvec = **User Trap Vector**: chứa địa chỉ trình xử lý ngắt.
* Khi có ngắt xảy ra, CPU tự động nhảy tới handler.
* Đoạn code:



t1 = 0x100 (bit 8 = 1).

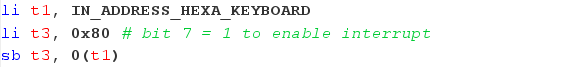
Ghi giá trị này vào **CSR uie** (User Interrupt Enable Register).

Bit 8 (UEIE) cho phép ngắt từ thiết bị ngoài (keypad).

* Cho phép ngắt ngoài cấp user (bàn phím).
* Đoạn code:



* Cho phép hệ thống nhận bất kỳ ngắt nào.
* Đoạn code:



Ghi 0x80 (bit 7 = 1) vào IN\_ADDRESS\_HEXA\_KEYBOARD.

Cụ thể: **bit 7 bật tính năng interrupt** của bàn phím.

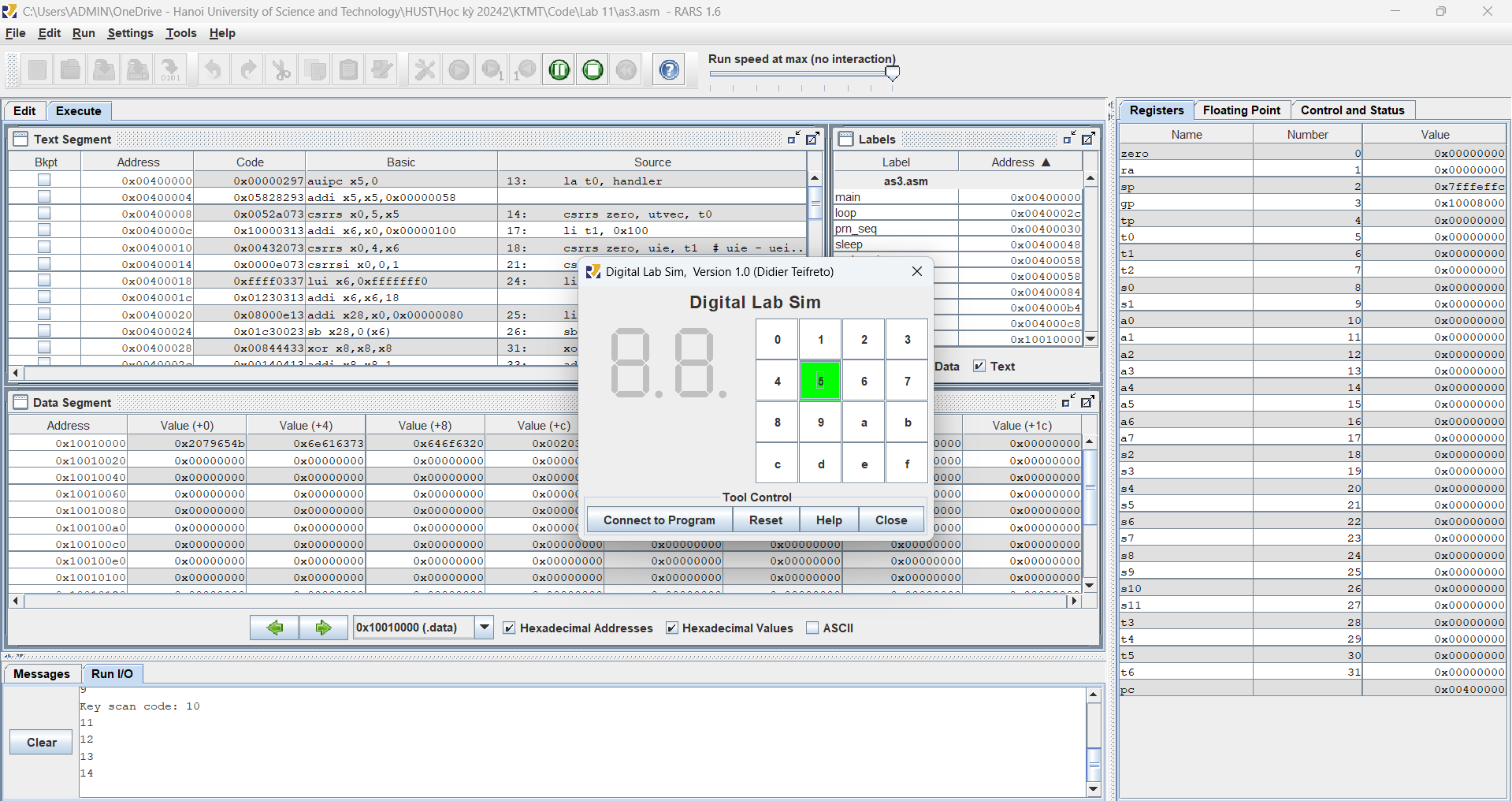
* Bàn phím sẽ gửi ngắt khi nhấn nút.

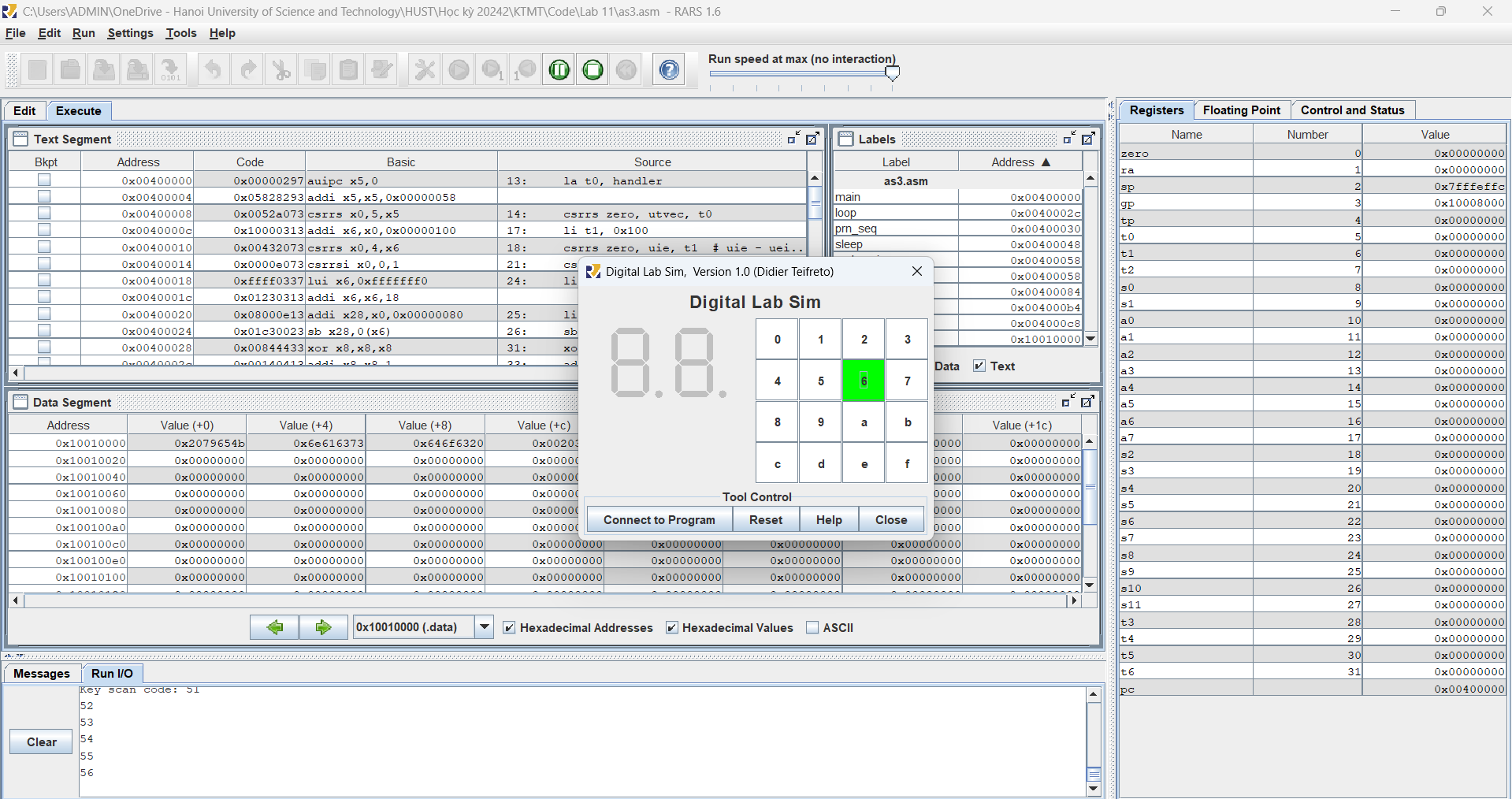
**Assignment 3**

**Code:**

|  |
| --- |
| .eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012  .eqv OUT\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014  .data  message: .asciz "Key scan code: "  # -----------------------------------------------------------------  # MAIN Procedure  # -----------------------------------------------------------------  .text  main:  # Load the interrupt service routine address to the UTVEC register  la t0, handler  csrrs zero, utvec, t0  # Set the UEIE (User External Interrupt Enable) bit in UIE register  li t1, 0x100  csrrs zero, uie, t1 # uie - ueie bit (bit 8)  # Set the UIE (User Interrupt Enable) bit in USTATUS register  csrrsi zero, ustatus, 1 # ustatus - enable uie (bit 0)  # Enable the interrupt of keypad of Digital Lab Sim  li t1, IN\_ADDRESS\_HEXA\_KEYBOARD  li t3, 0x80 # bit 7 = 1 to enable interrupt  sb t3, 0(t1)  # ---------------------------------------------------------  # Loop to print a sequence numbers  # ---------------------------------------------------------  xor s0, s0, s0 # count = s0 = 0  loop:  addi s0, s0, 1 # count = count + 1  prn\_seq:  addi a7, zero, 1  add a0, s0, zero # Print auto sequence number  ecall  addi a7, zero, 11  li a0, '\n' # Print EOL  ecall  sleep:  addi a7, zero, 32  li a0, 300 # Sleep 300 ms  ecall  j loop  end\_main:  # -----------------------------------------------------------------  # Interrupt service routine  # -----------------------------------------------------------------  handler:  # Save context  addi sp, sp, -16  sw a0, 0(sp)  sw a7, 4(sp)  sw t1, 8(sp)  sw t2, 12(sp)  # Print message  addi a7, zero, 4  la a0, message  ecall  # Scan all 4 rows to find which key is pressed  li t3, 0x1 # Start from row 0x1  li t4, 0x8 # Maximum row value is 0x8  scan\_rows:  li t1, IN\_ADDRESS\_HEXA\_KEYBOARD  li t5, 0x80 # Load immediate 0x80 vào thanh ghi t5  or t2, t3, t5 # t2 = row bit | enable interrupt bit  sb t2, 0(t1) # Select the row to scan  li t1, OUT\_ADDRESS\_HEXA\_KEYBOARD  lb a0, 0(t1) # Read key scan code  bne a0, zero, found\_key # If a key is detected, jump to print  slli t3, t3, 1 # Shift to next row (1->2->4->8)  ble t3, t4, scan\_rows  j end\_handler # No key found  found\_key:  # Print the scan code  li a7, 34 # Print integer (hex)  ecall  # Print newline  li a7, 11  li a0, '\n'  ecall  end\_handler:  # Restore context  lw t2, 12(sp)  lw t1, 8(sp)  lw a7, 4(sp)  lw a0, 0(sp)  addi sp, sp, 16  # Return from interrupt  uret |

Kết quả khi chạy chương trình:





Giải thích:

* **Chạy main** → **in ra một dãy số đếm** (74,75,76,...). <Với mục đích là để chương trình vẫn chạy liên tục mà không bị ngắt>
* Nếu  **nhấn nút** trên bàn phím keypad (hex keypad):
* **Tự động ngắt** chương trình chính (main),
* **Xử lý ngắt**: in ra mã số phím bấm theo định dạng: Key scan code: <địa chỉ>
* **Quay lại** chương trình chính.

**Bonus:**

**Code:**

|  |
| --- |
| .eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012  .eqv OUT\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014  .eqv MONITOR\_SCREEN 0x10010000 # Địa chỉ nơi Bitmap Display vẽ  .eqv RED 0xFF0000 # RGB Red  .eqv BLACK 0x000000 # RGB Black  .data  last\_red\_addr: .word 0 # Địa chỉ ô đã tô đỏ trước đó  .text  main:  # Thiết lập handler ngắt  la t0, handler  csrrs zero, utvec, t0  li t1, 0x100 # Bit 8 = UEIE  csrrs zero, uie, t1  csrrsi zero, ustatus, 1 # Bit 0 = UIE  # Bật ngắt keypad  li t1, IN\_ADDRESS\_HEXA\_KEYBOARD  li t2, 0x80  sb t2, 0(t1)  loop:  nop  nop  nop  j loop # Vòng lặp không làm gì cả, chờ ngắt  # ---------------------------------------------------------  # INTERRUPT HANDLER  # ---------------------------------------------------------  handler:  # Lưu context  addi sp, sp, -16  sw a0, 0(sp)  sw a7, 4(sp)  sw t1, 8(sp)  sw t2, 12(sp)  li t3, 0 # chỉ số hàng  scan\_rows:  li t1, IN\_ADDRESS\_HEXA\_KEYBOARD  li t2, 0x80 # Bật ngắt của bàn phím  li t4, 1  sll t4, t4, t3 # t4 = t4 \* 2^t3  or t2, t2, t4 # bật bit tương ứng với hàng cần quét  sb t2, 0(t1)  # Đọc mã phím  li t1, OUT\_ADDRESS\_HEXA\_KEYBOARD  lb a0, 0(t1)  beq a0, zero, next\_row # Không phím nào nhấn thì quét hàng tiếp theo    # Nếu phát hiện có phím được nhấn thì tìm toạ độ của phím  # a0 hiện tại đang chứa hàng và cột của phím đó  # 4 bit từ 0-3 chứa số hàng, chính là giá trị của t3  # 4 bit từ 4-7 chứa số cột, sẽ được tìm ra và lưu vào t5  li t5, 0 # chỉ số cột  find\_col:  li t6, 1  sll t6, t6, t5 # t6 = t6 \* 2^t5  slli t6, t6, 4 # Di chuyển để t6 đến bit từ 4-7 để check cột  and t6, t6, a0 # thực hiện and với a0 để tìm ra vị trí bit được đổi thành 1    # Nếu kq phép and = 0 nghĩa là tại vị trí bit tương ứng là 0, ngược lại thì bit tại đó bằng 1 và thực hiện vẽ.  bnez t6, draw\_square  addi t5, t5, 1 # Tăng chỉ số cột để tìm  li t0, 4  blt t5, t0, find\_col # Điều kiện thoát vòng lặp tìm cột  j end\_handler  draw\_square:  # Tính chỉ số ô  mul s0, t3, t0 # s0 = row \* 4  add s0, s0, t5 # s0 = index = row \* 4 + col  slli s0, s0, 2 # s0 = offset byte = index \* 4  li s1, MONITOR\_SCREEN  add s0, s0, s1 # s0 = địa chỉ ô cần tô màu đỏ  la a1, last\_red\_addr # Đọc địa chỉ ô được tô màu đỏ trước đó  lw a2, 0(a1)  beq a2, zero, skip\_clear # Nếu a2 = 0, tức đây là lần đầu nhấn phím thì bỏ qua bước tô đen ô cũ  li a3, BLACK # Tô đen ô đỏ cũ  sw a3, 0(a2)  skip\_clear:  li a4, RED # Tô đỏ ô mới  sw a4, 0(s0)    sw s0, 0(a1) # Lưu giá trị s0 vào bộ nhớ để dùng lần sau    next\_row:  addi t3, t3, 1  li t0, 4  blt t3, t0, scan\_rows # Kiểm tra hết thì quay về đầu bàn phím  end\_handler:  # Khôi phục ngữ cảnh  lw t2, 12(sp)  lw t1, 8(sp)  lw a7, 4(sp)  lw a0, 0(sp)  addi sp, sp, 16  uret |

**Kết quả:**

**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**