**THỰC HÀNH KIẾN TRÚC MÁY TÍNH BUỔI 12**

**Họ và tên:** Nguyễn Thành Duy

**MSSV:** 20235696

**Assignment 4**

**Code:**

|  |
| --- |
| .eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012  .eqv TIMER\_NOW 0xFFFF0018  .eqv TIMER\_CMP 0xFFFF0020  .eqv MASK\_CAUSE\_TIMER 4  .eqv MASK\_CAUSE\_KEYPAD 8  .data  msg\_keypad: .asciz "Someone has pressed a key!\n"  msg\_timer: .asciz "Time inteval!\n"  # -----------------------------------------------------------------  # MAIN Procedure  # -----------------------------------------------------------------  .text  main:  la t0, handler  csrrs zero, utvec, t0  li t1, 0x100  csrrs zero, uie, t1 # uie - ueie bit (bit 8) - external interrupt  csrrsi zero, uie, 0x10 # uie - utie bit (bit 4) - timer interrupt  csrrsi zero, ustatus, 1 # ustatus - enable uie - global interrupt  # ---------------------------------------------------------  # Enable interrupts you expect  # ---------------------------------------------------------  # Enable the interrupt of keypad of Digital Lab Sim  li t1, IN\_ADDRESS\_HEXA\_KEYBOARD  li t2, 0x80 # bit 7 of = 1 to enable interrupt  sb t2, 0(t1)  # Enable the timer interrupt  li t1, TIMER\_CMP  li t2, 1000  sw t2, 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:  # Saves the context  addi sp, sp, -16  sw a0, 0(sp)  sw a1, 4(sp)  sw a2, 8(sp)  sw a7, 12(sp)  # Handles the interrupt  csrr a1, ucause  li a2, 0x7FFFFFFF  and a1, a1, a2 # Clear interrupt bit to get the value  li a2, MASK\_CAUSE\_TIMER  beq a1, a2, timer\_isr  li a2, MASK\_CAUSE\_KEYPAD  beq a1, a2, keypad\_isr  j end\_process  timer\_isr:  li a7, 4  la a0, msg\_timer  ecall  # Set cmp to time + 1000  li a0, TIMER\_NOW  lw a1, 0(a0)  addi a1, a1, 1000  li a0, TIMER\_CMP  sw a1, 0(a0)  j end\_process  keypad\_isr:  li a7, 4  la a0, msg\_keypad  ecall  j end\_process  end\_process:  # Restores the context  lw a7, 12(sp)  lw a2, 8(sp)  lw a1, 4(sp)  lw a0, 0(sp)  addi sp, sp, 16  uret |

* Đoạn code lấy địa chỉ hàm handler (trình phục vụ ngắt), thiết lập địa chỉ xử lý ngắt vào thanh ghi utvec

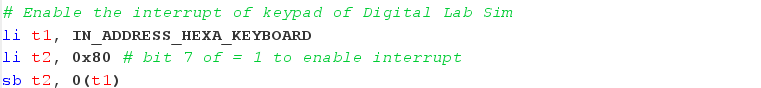
**A close up of a sign

AI-generated content may be incorrect.**

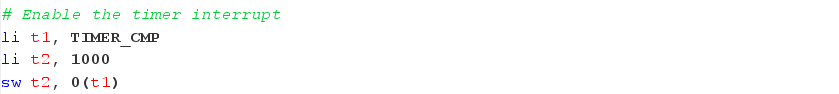
* Bật external interrupt (bit 8 = 0x100) trong thanh ghi uie. Bật **timer interrupt** (bit 4 = 0x10) trong thanh ghi uie.

****

* Gửi giá trị 0x80 vào địa chỉ keypad để bật ngắt thiết bị keypad (bit 7 = enable interrupt).

****

* Đặt giá trị TIMER\_CMP = 1000, tức là tạo mốc thời gian đầu tiên để timer sinh ngắt.

****

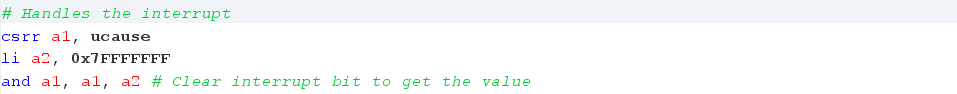
* Dành chỗ và lưu giá trị của các thanh ghi a0, a1, a2, a7 vào stack để bảo toàn trạng thái trước khi xử lý ngắt

**A white background with black text

AI-generated content may be incorrect.**

* Đọc mã nguyên nhân ngắt từ ucause.

 Dùng AND với 0x7FFFFFFF để xóa **interrupt bit** (bit 31), chỉ lấy cause code (0-30).

****

* So sánh a1 với MASK\_CAUSE\_TIMER (4) và MASK\_CAUSE\_KEYPAD (8).

 Nhảy đến timer\_isr hoặc keypad\_isr nếu phù hợp.

 Nếu không khớp ngắt nào thì đi tới end\_process.

**A close-up of a white background

AI-generated content may be incorrect.**

* Phục hồi giá trị của các thanh ghi a0, a1, a2, a7.

 Khôi phục lại con trỏ stack.

 Trở về chương trình chính bằng uret (User return from interrupt).

**A white background with black dots

AI-generated content may be incorrect.**

**Assignment 5**

**Code:**

|  |
| --- |
| .data  message: .asciz "Exception occurred.\n"  .text  main:  try:  la t0, catch  csrrw zero, utvec, t0 # Set utvec to the handler address  csrrsi zero, ustatus, 1 # Set interrupt enable bit in ustatus  lw zero, 0 # Trigger trap for Load access fault  finally:  li a7, 10 # Exit the program  ecall  catch:  # Show message  li a7, 4  la a0, message  ecall  # Since uepc contains address of the error instruction  # Need to load finally address to uepc  la t0, finally  csrrw zero, uepc, t0  uret |

* Load địa chỉ của nhãn catch vào thanh ghi t0.



* Bật bit 0 của ustatus (gọi là UIE – User Interrupt Enable).
* Cho phép nhận ngắt và exception ở chế độ người dùng (user-mode).



* Khi xảy ra trap, địa chỉ lệnh gây lỗi sẽ được lưu vào **uepc**.

 Để tiếp tục thực thi chương trình (bỏ qua lệnh lỗi), ta gán lại uepc thành địa chỉ của finally.

 csrrw zero, uepc, t0: ghi địa chỉ finally vào uepc



**Assignment 6: Ngắt mềm  
Code:**

|  |
| --- |
| .data  overflow\_msg: .asciz "Overflow occurred. Program terminated.\n"  .text  .globl main  main:  # ----- Đặt địa chỉ trình xử lý ngắt (trap handler)  la t0, handler  csrrw zero, utvec, t0 # Gán utvec = handler  csrrsi zero, ustatus, 1 # Bật global interrupt (UIE)  li t0, 0x1  csrrs zero, uie, t0 # Bật USIE (bit 0) trong uie → cho phép ngắt mềm  # ----- Cộng hai số nguyên có dấu (có thể thay đổi giá trị để kiểm thử)  li s1, 0x7FFFFFFF # MAX\_INT  li s2, 1  li t0, 0 # Mặc định: không tràn  add s3, s1, s2 # s3 = s1 + s2  xor t1, s1, s2 # Kiểm tra dấu của s1 và s2  blt t1, zero, end\_main # Nếu khác dấu → không thể tràn  blt s1, zero, NEGATIVE # Nếu s1 âm → kiểm tra theo nhánh âm  bge s3, s1, end\_main # Nếu s3 >= s1 → không tràn  j OVERFLOW  NEGATIVE:  bge s1, s3, end\_main # Nếu s1 >= s3 → không tràn  OVERFLOW:  li t0, 1 # Đánh dấu tràn  # Nếu t0 == 1 thì kích hoạt ngắt mềm  li t1, 1  csrrs zero, uip, t1 # Thiết lập USIP = 1  wait\_loop:  j wait\_loop # Đợi ngắt được xử lý  end\_main:  li a7, 10 # Không tràn → thoát chương trình  ecall  # -----------------------------  # Interrupt/trap handler  # -----------------------------  handler:  # Lưu context  addi sp, sp, -16  sw a0, 0(sp)  sw a1, 4(sp)  sw a7, 8(sp)  sw t0, 12(sp)  # Xác định nguyên nhân trap  csrr a0, ucause  li a1, 0x7FFFFFFF  and a0, a0, a1 # Xóa bit MSB (interrupt flag)  li a1, 0 # Mã ngắt mềm = 0 (USIP)  bne a0, a1, end\_handler # Nếu không phải ngắt mềm → bỏ qua  # ----- In thông báo lỗi  li a7, 4  la a0, overflow\_msg  ecall  # ----- Thoát chương trình  li a7, 10  ecall  end\_handler:  # Khôi phục context  lw a0, 0(sp)  lw a1, 4(sp)  lw a7, 8(sp)  lw t0, 12(sp)  addi sp, sp, 16  uret |

**Kết quả:**

Với s1 = 0x7FFFFFFF, s2 = 1 => s1 + s2 sẽ bị tràn số

A screenshot of a computer

AI-generated content may be incorrect.

Với s1 = 0x7FFFFFFF, s2 = -5 => s1 + s2 không tràn số nên không in ra thông báo và kết thúc chương trình

A screenshot of a computer

AI-generated content may be incorrect.

**Giải thích:**

* Kiểm tra tổng 2 số

+) Nếu như không xảy ra hiện tượng tràn số thì chương trình tiếp tục thực hiện các lệnh tiếp theo

+) Khi phát hiện tràn → **kích hoạt ngắt mềm** → in thông báo và kết thúc chương trình

**Bài tập bổ sung:**

**Code:**

|  |
| --- |
| .eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012 # địa chỉ truy cập thanh ghi làm việc với ngắt bàn phím  .eqv OUT\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014  .eqv TIMER\_NOW 0xFFFF0018 # địa chỉ truy cập thanh ghi lưu, đọc thời gian hiện tại  .eqv TIMER\_CMP 0xFFFF0020 # địa chỉ truy cập thanh ghi so sánh  .eqv MASK\_CAUSE\_TIMER 4 # mã nguyên nhân ngắt do bộ định thời  .eqv MASK\_CAUSE\_KEYPAD 8 # mã nguyên nhân ngắt do bàn phím  .eqv SEVENSEG\_LEFT 0xFFFF0011  .eqv SEVENSEG\_RIGHT 0xFFFF0010 # Gán địa chỉ của 2 đèn LED  .data  increase\_msg: .asciz "Đếm tăng dần.\n"  decrease\_msg: .asciz "Đếm giảm dần.\n"  slow\_down\_msg: .asciz "Giảm tốc. "  speed\_up\_msg: .asciz "Tăng tốc. "  speed\_info: .asciz "Thời gian 1 chu kỳ hiện tại là: "  msg1: .asciz " ms.\n"  led: .word 0x3F 0x06 0x5B 0x4F 0x66 0x6D 0x7D 0x07 0x7F 0x6F # Mảng mã của số hiện trên đèn LED từ 0-9  count: .word 0 # nơi lưu biến đếm hiện tại  cycle1: .word 1000 # nơi lưu chu kỳ bộ định thời  direction: .word 0 # nơi lưu hướng đếm hiện tại, 0 là tăng, 1 là giảm  .text  main:  la t0, handler  csrrs zero, utvec, t0 # lấy địa chỉ của hàm xử lý ngắt    li t1, 0x100  csrrs zero, uie, t1 # bật bit 8 của uie - ueie cho phép xử lý ngắt ngoài  csrrsi zero, uie, 0x10 # bật bit 4 của uie - utie cho phép xử lý ngắt thời gian    csrrsi zero, ustatus, 1 # bật bit 0 của ustatus cho phép xử lý ngắt ở User Mode    li t1, IN\_ADDRESS\_HEXA\_KEYBOARD  li t2, 0x80  sb t2, 0(t1) # bật ngắt của bàn phím    li t1, TIMER\_CMP  li t2, 1000  sw t2, 0(t1) # đặt thời gian ngắt mặc định là 1s    loop:  nop  nop  nop  j loop # Vòng lặp vô hạn  #--------------Trình xử lý ngắt-------------------  handler:  csrr a1, ucause  li a2, 0x7FFFFFFF  and a1, a1, a2 # loại bỏ bit 31    li a2, MASK\_CAUSE\_TIMER # nếu là bit 2, tức kq a1 = 4 thì là ngắt thời gian  beq a1, a2, timer\_isr  li a2, MASK\_CAUSE\_KEYPAD # nếu là bit 3, tức kq a1 = 8 thì là ngắt ngoài do bàn phím  beq a1, a2, keypad\_isr  j end\_handler # nếu không phải 2 loại ngắt này thì không xử lý    #----Xử lý ngắt thời gian, mỗi chu kỳ cập nhật lại LED 1 lần----------  timer\_isr:  la t1, count  lw t2, 0(t1) # đọc giá trị count từ bộ nhớ    li a7, 1  mv a0, t2  ecall # in count hiện tại ra màn hình    li a7, 11  li a0, 32 # thêm dấu cách giữa các số  ecall    li t1, 10  div t3, t2, t1 # lấy chữ số hàng chục  rem t4, t2, t1 # lấy chữ số hàng đơn vị  la t1, led  slli t3, t3, 2 # t3 = 4\*t3  add t3, t3, t1 # địa chỉ của mã trong mảng  lw t5, 0(t3) # Lấy mã LED của số hàng chục từ mảng led  li t6, SEVENSEG\_LEFT  sb t5, 0(t6) # hiện đèn bên trái    slli t4, t4, 2  add t4, t4, t1 # địa chỉ của mã trong mảng  lw t5, 0(t4) # Lấy mã LED của số đơn vị từ mảng led  li t6, SEVENSEG\_RIGHT  sb t5, 0(t6) # hiện đèn bên phải    la t1, direction  lw t3, 0(t1) # đọc hướng đếm hiện tại    beq t3, zero, increase\_count # hàm direction = 0, đang đếm tăng thì nhảy hàm tương ứng    decrease\_count:  addi t2, t2, -1 # direction khác 0 là bằng 1 thì giảm biến đếm  li t4, 0  bge t2, t4, update\_count # nếu count hiện tại >= 0 thì cập nhật luôn  li t2, 99 # nếu count < 0 thì cần đặt lại là 99 sau đó mới cập nhật  j update\_count    increase\_count:  addi t2, t2, 1 # tăng biến đếm  li t4, 99  ble t2, t4, update\_count # nếu count hiện tại <=99 thì cập nhật luôn  li t2, 0 # nếu đã lớn hơn 99 thì đặt về 0 và cập nhật  j update\_count # dòng hơi thừa nhưng nhét vào cho cân xứng :))    update\_count:  la t1, count  sw t2, 0(t1) # cập nhật lại count vào bộ nhớ    # Cập nhật chu kỳ của bộ định thời  li t1, TIMER\_NOW  lw t2, 0(t1)  la t3, cycle1  lw t4, 0(t3) # lấy tốc độ hiện tại  add t2, t2, t4 # đặt thời điểm ngắt kế tiếp  li t1, TIMER\_CMP  sw t2, 0(t1)    j end\_handler  #------Xử lý ngắt từ bàn phím, khi nhấn các phím 0, 1, 2, 3-----------  keypad\_isr:  li t1, IN\_ADDRESS\_HEXA\_KEYBOARD  li t2, 0x81  sb t2, 0(t1)  li t1, OUT\_ADDRESS\_HEXA\_KEYBOARD  lb t2, 0(t1) # đọc mã của phím được nhấn    # Kiểm tra phím được nhấn và thực hiện hành động tương ứng  li t3, 0x11 # Phím 0: Tăng dần  beq t2, t3, increase    li t3, 0x21 # Phím 1: Giảm dần  beq t2, t3, decrease    li t3, 0x41 # Phím 2: Giảm tốc độ (tăng chu kỳ)  beq t2, t3, slow\_down    li t3, 0xffffff81 # Phím 3: Tăng tốc độ (giảm chu kỳ)  beq t2, t3, speed\_up    j end\_handler  increase:  la t1, direction  li t2, 0  sw t2, 0(t1) # đặt giá trị direction về 0  li a7, 4  la a0, increase\_msg  ecall # in ra thông báo  j end\_handler  decrease:  la t1, direction  li t2, 1  sw t2, 0(t1) # đặt giá trị direction về 1  li a7, 4  la a0, decrease\_msg # in ra thông báo  ecall  j end\_handler  slow\_down:  la t1, cycle1  lw t2, 0(t1) # Load chu kỳ hiện tại  li t3, 2  mul t2, t2, t3 # Tăng thời gian chu kỳ gấp đôi  sw t2, 0(t1) # lưu lại vào bộ nhớ    li a7, 4  la a0, slow\_down\_msg # In ra thông báo  ecall    li a7, 4  la a0, speed\_info  ecall  li a7, 1  mv a0, t2  ecall  li a7, 4  la a0, msg1  ecall # Thông báo chu kỳ hiện tại  j end\_handler  speed\_up:  la t1, cycle1  lw t2, 0(t1) # Load chu kỳ hiện tại  li t3, 2  div t2, t2, t3 # Giảm thời gian chu kỳ một nửa  sw t2, 0(t1) # lưu lại vào bộ nhớ    li a7, 4  la a0, speed\_up\_msg # In ra thông báo  ecall    li a7, 4  la a0, speed\_info  ecall  li a7, 1  mv a0, t2  ecall  li a7, 4  la a0, msg1  ecall # Thông báo chu kỳ hiện tại  j end\_handler # Thêm vô cho đều chứ cũng không cần lắm  end\_handler:  uret |

* **Ý tưởng:**

- Thiết lập ngắt sau:

• Ngắt từ bộ định thời, 1 giây ngắt 1 lần.

• Ngắt từ bàn phím.

- Kiểm tra dữ liệu lấy được từ bàn phím để điều hướng tới hàm con để xử lý.

- Cách để 2 đèn LED chạy từ 00 đến 99. Khai báo ở .data một mảng chứa mã đèn LED của các số từ 0 – 9, sử dụng vòng lặp lấy biến đếm của vòng lặp làm chữ số hiển thị trên đèn LED.

- Xử lý phím được nhấn:

+ Nếu phím được nhấn là 0, biến đếm i = i + 1 sau mỗi vòng lặp. Nếu i = 100 đặt lại i = 0.

+ Nếu phím được nhấn là 1, biến đếm i = i - 1 sau mỗi vòng lặp. Nếu i = 0 thì đặt lại i = 99. + Nếu phím được nhấn là 2, tăng chu kỳ tại bộ định thời bằng cách tăng thời gian lên gấp đôi, tốc độ nhảy số sẽ giảm đi.

+ Nếu phím được nhấn là 3, giảm chu kỳ tại bộ định thời bằng cách giảm thời gian đi 1 nửa, tốc độ nhảy số sẽ tăng lên.

+ Để dễ dàng quan sát thông qua báo cáo, mỗi xử lý với phím được đưa ra 1 thông báo tương ứng, và mỗi chu kỳ sẽ in ra số trên đèn LED.

* Kết quả:

A white text with black numbers

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.