Lab 9 (Part 1)

Interrupts & IO programming

Đinh Ngọc Khánh Huyền – 20225726

Assigment 1

Code

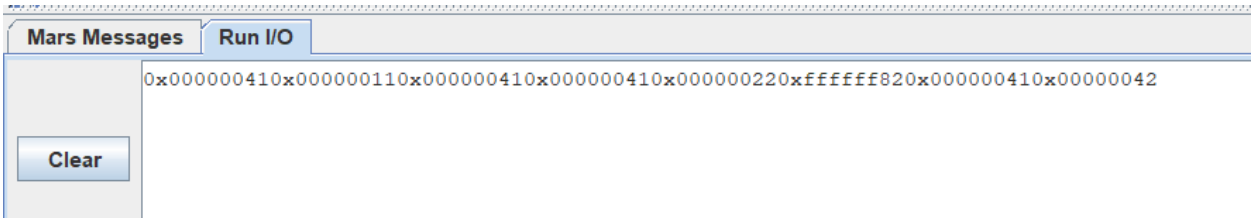
```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014



.text
main:
        li $t1, IN_ADRESS_HEXA_KEYBOARD
        li $t2, OUT_ADRESS_HEXA_KEYBOARD



start_polling_1:
        li $t3, 0x01 # check row 1 with key 0, 1, 2, 4
        sb $t3, 0($t1) # must reassign expected row
        jal polling

start_polling_2:
        li $t3, 0x02 # check row 2 with key 4, 5, 6, 7
        sb $t3, 0($t1) # must reassign expected row
        jal polling

start_polling_3:
```

```
        li $t3, 0x04 # check row 3 with key 8, 9, A, B
        sb $t3, 0($t1) # must reassign expected row
        jal polling

start_polling_4:
        li $t3, 0x08 # check row 4 with key C, D, E, F
        sb $t3, 0($t1) # must reassign expected row
        jal polling

check_after_polling_4:
        beq $a0, 0x0, print
        j start_polling_1

polling:
        lb $a0, 0($t2) # read scan code of key button
        bne $a0, 0x0, print
        jr $ra

print:
        li $v0, 34 # print integer (hexa)
        syscall

sleep:
        li $a0, 100 # sleep 100ms
        li $v0, 32
        syscall
```

back_to_start_polling:

j start_polling_1 # back to check row 1

Result



MSSV: 20225726

Explain

Yêu cầu: Check toàn bộ các ký tự từ 0 -> F

Giả sử với MSSV 20225726

0x41 là số 2;

0x11 là số 0;

0x41 là số 2;

0x41 là số 2;

0x22 là số 5;

0x82 là số 7;

0x41 là số 2;

0x42 là số 6;

⇨ MSSV: 20225726

Assigment 2

Code

.eqv SEVENSEG_LEFT 0xFFFF0011

.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012

```
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014

.text
main:
    li $t1, IN_ADRESS_HEXA_KEYBOARD
    li $t2, OUT_ADRESS_HEXA_KEYBOARD
    li $t4, 0x10

set:
    li $t3, 0x01

polling:
    beq $t3, $t4, print
    sb $t3, 0($t1) # must reassign expected row
    lb $a0, 0($t2) # read scan code of key button
    bne $a0, $0, print
    sll $t3, $t3, 1
    j polling

print:
    beq $a0, 0x11, zero
    beq $a0, 0x21, one
    beq $a0, 0x41, two
    beq $a0, 0x81, three
    beq $a0, 0x12, four
    beq $a0, 0x22, five
    beq $a0, 0x42, six
```

```
    beq $a0, 0x82, seven
    beq $a0, 0x14, eight
    beq $a0, 0x24, nine
    beq $a0, 0x44, Achar
    beq $a0, 0x84, Bchar
    beq $a0, 0x18, Cchar
    beq $a0, 0x28, Dchar
    beq $a0, 0x48, Echar
    beq $a0, 0x88, Fchar

zero:
    li $a0, 0x3f # set value for segments
    jal SHOW_7SEG_LEFT # show
    nop
    j sleep

one:
    li $a0, 0x06 # set value for segments
    jal SHOW_7SEG_LEFT # show
    nop
    j sleep

two:
    li $a0, 0x5b # set value for segments
    jal SHOW_7SEG_LEFT # show
    nop
    j sleep
```

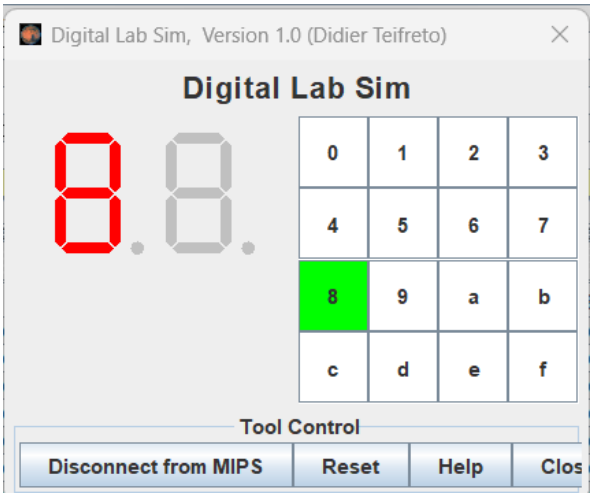
```
three:
    li $a0, 0x4f # set value for segments
    jal SHOW_7SEG_LEFT # show
    nop
    j sleep

four:
    li $a0, 0x66 # set value for segments
    jal SHOW_7SEG_LEFT # show
    nop
    j sleep

five:
    li $a0, 0x6d # set value for segments
    jal SHOW_7SEG_LEFT # show
    nop
    j sleep

six:
    li $a0, 0x7d # set value for segments
    jal SHOW_7SEG_LEFT # show
    nop
    j sleep

seven:
    li $a0, 0x7 # set value for segments
```

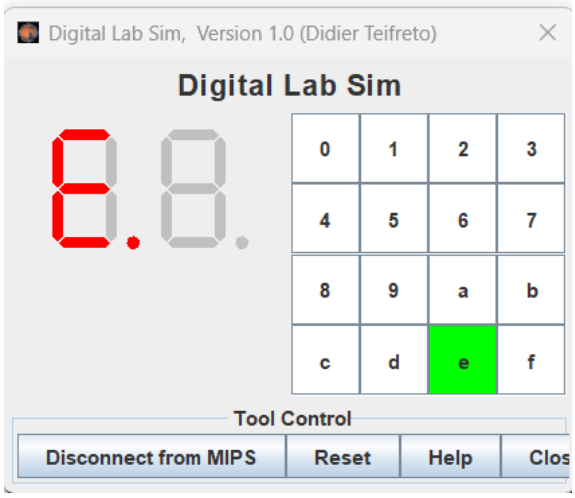
```
    jal SHOW_7SEG_LEFT # show
    nop
    j sleep

eight:
    li $a0, 0x7f # set value for segments
    jal SHOW_7SEG_LEFT # show
    nop
    j sleep

nine:
    li $a0, 0x6f # set value for segments
    jal SHOW_7SEG_LEFT # show
    nop
    j sleep

Achar:
    li $a0, 0xf7 # set value for segments
    jal SHOW_7SEG_LEFT # show
    nop
    j sleep

Bchar:
    li $a0, 0xff # set value for segments
    jal SHOW_7SEG_LEFT # show
    nop
    j sleep
```

```
Cchar:
    li $a0, 0xb9 # set value for segments
    jal SHOW_7SEG_LEFT # show
    nop
    j sleep

Dchar:
    li $a0, 0xbf # set value for segments
    jal SHOW_7SEG_LEFT # show
    nop
    j sleep

Echar:
    li $a0, 0xf9 # set value for segments
    jal SHOW_7SEG_LEFT # show
    nop
    j sleep

Fchar:
    li $a0, 0xf1 # set value for segments
    jal SHOW_7SEG_LEFT # show
    nop
    j sleep

sleep:
    li $a0, 2000 # sleep 100ms
```

```
    li $v0, 32
    syscall
    j set

back_to_polling:
    j polling # continue polling

#------------------------------

SHOW_7SEG_LEFT:
    li $t0, SEVENSEG_LEFT # assign port's address
    sb $a0, 0($t0) # assign new value
    nop
    jr $ra
    nop
```

Result





Quizzes

Ưu và nhược điểm của phương pháp Polling:

- **Ưu điểm:**
  + Đơn giản: dễ hiểu và dễ triển khai.
  + Kiểm soát: dễ dàng kiểm soát chính xác thời gian kiểm tra trạng thái, giúp quản lý tài nguyên hệ thống hiệu quả hơn.
  + Không yêu cầu phần cứng phức tạp: có thể triển khai trên nhiều loại phần cứng.
- **Nhược điểm:**
  + Tốn tài nguyên CPU: liên tục kiểm tra trạng thái, thậm chí khi không có sự kiện nào xảy ra.
  + Trễ: Do phải chờ đợi đến lượt kiểm tra, phương pháp này có thể tạo ra độ trễ không mong muốn trong việc phát hiện sự kiện.
  + Khả năng bỏ sót: có thể bỏ sót việc phát hiện sự kiện.

Ưu và nhược điểm của phương pháp dựa trên ngắt:

- **Ưu điểm:**
  + Phản ứng nhanh chóng: Ngắt cho phép phản ứng ngay lập tức khi sự kiện xảy ra, giảm thiểu độ trễ so với phương pháp polling.
  + Hiệu suất cao: CPU chỉ sử dụng tài nguyên khi có sự kiện xảy ra, giảm thiểu lãng phí tài nguyên.
  + Khả năng ưu tiên: Ngắt có thể được ưu tiên theo mức độ ưu tiên của các nguồn ngắt khác nhau.
- **Nhược điểm:**
  + Độ phức tạp cao hơn: Phải triển khai các trình xử lý ngắt và quản lý tài nguyên phức tạp hơn so với phương pháp polling.
  + Khả năng gây nghẽn: Nếu quá nhiều ngắt xảy ra cùng một lúc, có thể xảy ra tình trạng gây nghẽn, ảnh hưởng đến hiệu suất hệ thống.