Bài thực hành số 11

Lớp: 139365 – Học phần: Thực Hành Kiến Trúc Máy Tính

Đào Minh Nhật – 20215107

Assignment 1:

Code:

```
#Lab 11, assignment 1
.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 $t3, 0x01    # row
                li $t4, 0x10

polling:        sb $t3, 0($t1) # must reassign expected row
                lb $a0, 0($t2) # read scan code of key button
                bne $a0, $0, print
                j next_row
print:          li $v0, 34 # print integer (hexa)
                syscall
sleep_10:       li $a0, 10 # sleep 100ms
                li $v0, 32
                syscall
next_row:       sll $t3, $t3, 1
                beq $t3, $t4, reset
                j polling
reset:          li $t3, 0x01
sleep_1000:     li $a0, 1000 # sleep 100ms
                li $v0, 32
                syscall
```

Giải thích:

Biến IN_ADRESS_HEXA_KEYBOARD lưu địa chỉ 0xFFFF0012 dùng để lưu địa chỉ của hàng cần duyệt.

Biến OUT_ADRESS_HEXA_KEYBOARD lưu địa chỉ 0xFFFF0014 để phát hiện nút nào đã được bấm.

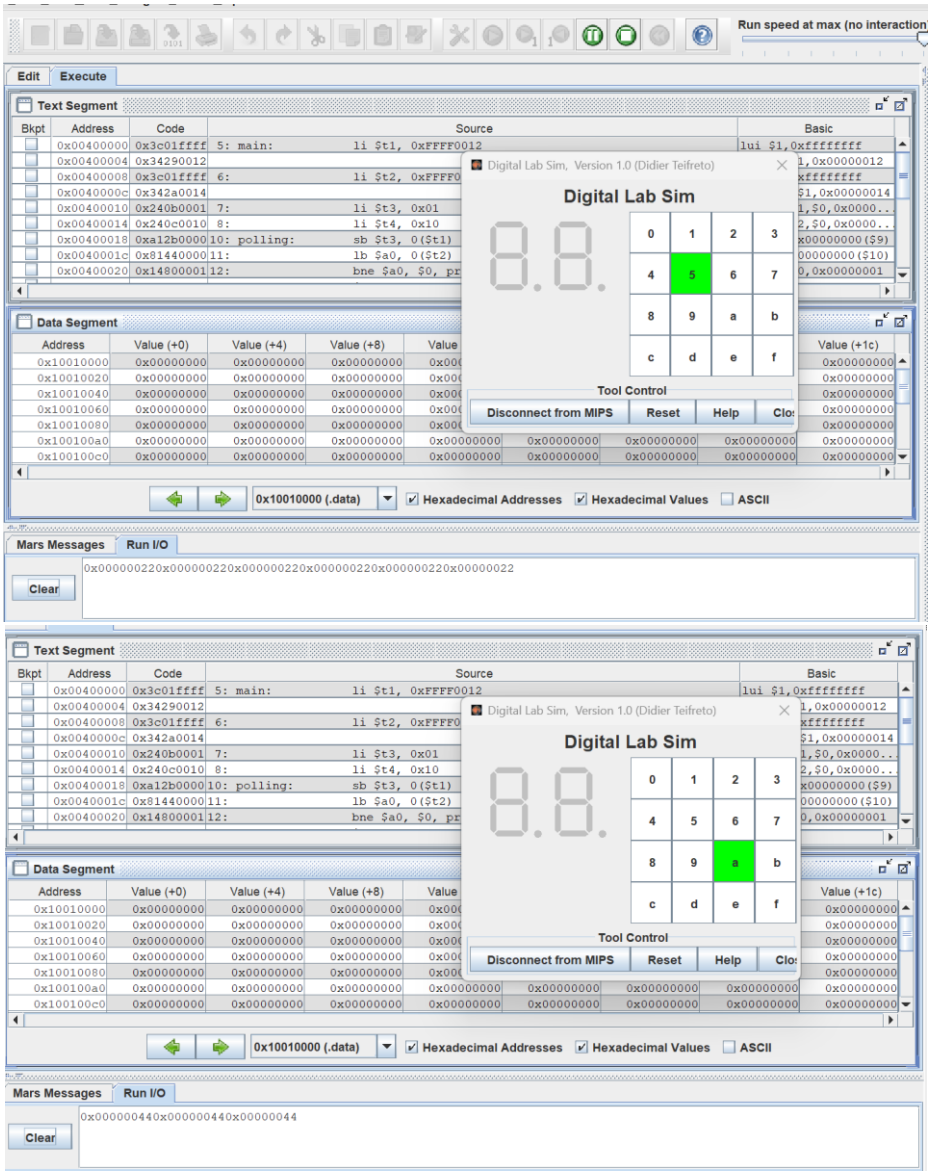
Cách hoạt động:

Thanh $t3 lưu địa chỉ của hàng cần duyệt

Duyệt lần lượt từng hàng 0x01, 0x02, 0x04, 0x08 bằng cách dùng lệnh dịch bit sll. Nếu $t3 = 0x10 thì sẽ reset lại $t3 = 0x01 để duyệt lại từ hàng đầu.

Nếu phát hiện ra nút bấm thì dùng lệnh lb từ địa chỉ 0xFFFF0014 để lấy ra nút đã được bấm và in ra.

Kết quả:



Assignment 2:

Code:

```
#Lab 11, assignment 2
.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 $t3, 0x80 # bit 7 of = 1 to enable interrupt
                sb $t3, 0($t1)

Loop:           nop
                nop
                nop
                nop
                b Loop # Wait for interrupt
end_main:

.ktext 0x80000180
IntSR:          li $t1, IN_ADRESS_HEXA_KEYBOARD
                li $t2, OUT_ADRESS_HEXA_KEYBOARD
                li $t3, 0x01    # row

                li $t4, 0x10
pol:            sb $t3, 0($t1) # must reassign expected row
                lb $a0, 0($t2) # read scan code of key button
                bne $a0, $0, print
                j next_row

print:          li $v0, 34 # print integer (hexa)
                syscall
sleep_10:       li $a0, 10 # sleep 100ms
                li $v0, 32
next_row:       sll $t3, $t3, 1
                beq $t3, $t4, end
                j pol
end:
                li $t3, 0x80 # bit 7 of = 1 to enable interrupt
                sb $t3, 0($t1)

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
```

Giải thích:

Gán thanh ghi $t3 = 0x80 đặt bit thứ 7 là 1 để kích hoạt ngắt trên bàn phím (interrupt).

Vòng lặp loop để chờ ngắt.

Sử dụng chỉ thị .ktext để viết code ở địa chỉ 0x80000180 để viết chương trình con phục vụ ngắt.

Hàm IntSR thực hiện in ra nút đã bấm.

Sau khi in được nút đã bấm thì gán lại thanh ghi $t3 = 0x80 vào địa chỉ IN_ADRESS_HEXA_KEYBOARD để chuẩn bị cho lệnh ngắt tiếp theo.

Hàm next_pc : Lấy giá trị của thanh ghi $at (Coproc0.$14) và cộng thêm 4 để chuyển đến lệnh kế tiếp.

Assignment 3:

Code:

```
#Lab 11, assignment 3
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014
.data
        Message: .asciiz "Key scan code "
.text
main:
                li $t1, IN_ADRESS_HEXA_KEYBOARD
                li $t3, 0x80 # bit 7 = 1 to enable
                sb $t3, 0($t1)

                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 $a0,300 # sleep 300 ms
                syscall
                nop # WARNING: nop is mandatory here.
                b Loop # Loop
end_main:

.ktext 0x80000180
IntSR:          addi $sp,$sp,4
                sw $ra,0($sp)
                addi $sp,$sp,4
                sw $at,0($sp)
                addi $sp,$sp,4
                sw $v0,0($sp)
                addi $sp,$sp,4
                sw $a0,0($sp)
                addi $sp,$sp,4
                sw $t1,0($sp)
                addi $sp,$sp,4
                sw $t3,0($sp)

prn_msg:        addi $v0, $zero, 4
```

```
43                  la $a0, Message
44                  syscall
45
46                  li $t3, 0x01
47                  li $t5, 0x10
48  get_cod:        li $t1, IN_ADRESS_HEXA_KEYBOARD
49                  ori $t4, $t3, 0x80
50                  sb $t4, 0($t1) # must reassign expected row
51                  li $t1, OUT_ADRESS_HEXA_KEYBOARD
52                  lb $a0, 0($t1)
53                  bne $a0, $0, prn_cod
54                  sll $t3, $t3, 1
55                  beq $t3, $t5, end_get
56                  j get_cod
57  end_get:        nop
58
59  prn_cod:        li $v0,34
60                  syscall
61                  li $v0,11
62                  li $a0,'\n' # print endofline
63                  syscall
64
65  next_pc:        mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
66                  addi $at, $at, 4 # $at = $at + 4 (next instruction)
67                  mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
68
69  restore:        lw $t3, 0($sp)
70                  addi $sp,$sp,-4
71                  lw $t1, 0($sp)
72                  addi $sp,$sp,-4
73                  lw $a0, 0($sp)
74                  addi $sp,$sp,-4
75                  lw $v0, 0($sp)
76                  addi $sp,$sp,-4
77                  lw $ra, 0($sp)
78                  addi $sp,$sp,-4
79  return:         eret # Return from exception
```

Giải thích:

Chương trình thực hiện đếm số lần lượt từ 0.

Nếu suất hiện ngắt bàn phím thì sẽ in ra nút được bấm.

Cách hoạt động:

Khởi tạo $t3 = 0x80 (khởi tạo bit 7 = 1) để kích hoạt ngắt từ bàn phím.

$s0 là đếm số.

Vòng lặp loop thực hiện đếm số chờ lệnh ngắt từ bàn phím.

Hàm prn_seq: in ra chuỗi số đếm.

Hàm prn_eol: in ra xuống dòng.

Hàm IntSR: thực hiện in ra số được bấm.

Lệnh addi và sw dùng để lưu trữ các giá trị $ra, $at, $v0, $a0, $t1, $t3 vào stack.

Hàm get_cod: thực hiện lấy giá trị của nút được bấm gán vào thanh ghi $a0. Bằng cách duyệt lần lượt từng hàng.

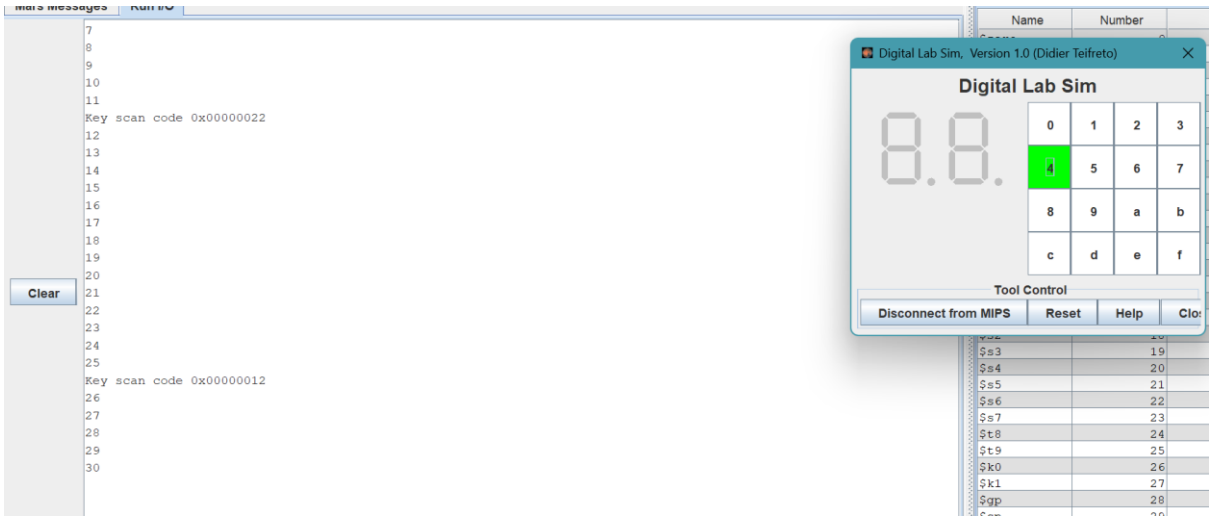
Hàm prn_cod: in ra nút bấm vừa tìm được và xuống dòng.

Hàm next_pc: Cộng thêm 4 vào thanh $at để giá trị trả về là địa chỉ của lệnh tiếp theo.

Hàm restore: lấy lại các giá trị đã được lưu trong stack ban đầu.

Hàm return: trở lại đếm số và chờ lệnh ngắt tiếp theo

Kết quả:



Assignment 4:

Code:

```
#Lab 11, assignment 4
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014
.eqv COUNTER 0xFFFF0013 # Time Counter
.eqv MASK_CAUSE_COUNTER 0x00000400 # Bit 10: Counter interrupt
.eqv MASK_CAUSE_KEYMATRIX 0x00000800 # Bit 11: Key matrix interrupt
.data
        msg_keypress: .asciiz "Someone has pressed a key : "
        msg_counter: .asciiz "Time inteval!\n"
.text
main:
        li $t1, IN_ADRESS_HEXA_KEYBOARD
        li $t3, 0x80 # bit 7 = 1 to enable
        sb $t3, 0($t1)
 # Enable the interrupt of TimeCounter of Digital Lab Sim
        li $t1, COUNTER
        sb $t1, 0($t1)

 #-----------------------------------------------------------
 # Loop an print sequence numbers
 #-----------------------------------------------------------
Loop:   nop
        nop
        nop
sleep:  addi $v0,$zero,32 # BUG: must sleep to wait for Time Counter
        li $a0,200 # sleep 200 ms
        syscall
        nop # WARNING: nop is mandatory here.
        b Loop
end_main:
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# GENERAL INTERRUPT SERVED ROUTINE for all interrupts
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.ktext 0x80000180
IntSR:
 # Temporary disable interrupt
 #-----------------------------------------------------------
dis_int:li $t1, COUNTER # BUG: must disable with Time Counter
        sb $zero, 0($t1)
 #-----------------------------------------------------------
 # Processing
 #-----------------------------------------------------------
get_caus:       mfc0 $t1, $13 # $t1 = Coproc0.cause
IsCount:        li $t2, MASK_CAUSE_COUNTER# if Cause value confirm Counter..
                and $at, $t1,$t2
                beq $at,$t2, Counter_Intr
IsKeyMa:        li $t2, MASK_CAUSE_KEYMATRIX # if Cause value confirm Key..
                and $at, $t1,$t2
                beq $at,$t2, Keymatrix_Intr
others:         j end process # other cases
```

```
51
52  Keymatrix_Intr: li $v0, 4 # Processing Key Matrix Interrupt
53                  la $a0, msg_keypress
54                  syscall
55
56                  li $t3, 0x01
57                  li $t5, 0x10
58  get_cod:        li $t1, IN_ADRESS_HEXA_KEYBOARD
59                  ori $t4, $t3, 0x80
60                  sb $t4, 0($t1) # must reassign expected row
61                  li $t1, OUT_ADRESS_HEXA_KEYBOARD
62                  lb $a0, 0($t1)
63                  bne $a0, $0, prn_cod
64                  sll $t3, $t3, 1
65                  beq $t3, $t5, end_get
66                  j get_cod
67  end_get:        nop
68
69  prn_cod:        li $v0,34
70                  syscall
71                  li $v0,11
72                  li $a0,'\n' # print endofline
73                  syscall
74
75                  j end_process
```

```
76
77  Counter_Intr:   li $v0, 4 # Processing Counter Interrupt
78                  la $a0, msg_counter
79                  syscall
80                  j end_process
81  end_process:
82                  mtc0 $zero, $13 # Must clear cause reg
83  en_int: #-----------------------------------------------------------
84   # Re-enable interrupt
85   #-----------------------------------------------------------
86                  li $t1, COUNTER
87                  sb $t1, 0($t1)
88   #-----------------------------------------------------------
89  next_pc:        mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
90                  addi $at, $at, 4 # $at = $at + 4 (next instruction)
91                  mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
92  return: eret # Return from exception
93
```

Giải thích:

Hàm ‘main’:

- Khởi tạo bit 7 = 1 và gán vào địa chỉ đầu vào của bàn phím để kích hoạt ngắt.
- Ghi giá trị trong thanh ghi $t1 (COUNTER - địa chỉ thanh ghi đếm thời gian) vào địa chỉ thanh ghi đếm thời gian để kích hoạt ngắt đếm thời gian.

- Vòng lặp loop và sleep để chờ ngắt.

Hàm ‘dis_int’: vô hiệu hóa ngắt bằng cách gán giá trị 0 vào địa chỉ thanh ghi đếm thời gian.

Hàm ‘get_caus’: sao chép giá trị của thanh ghi Cause vào thanh ghi $t1 để lấy nguyên nhân ngắt.

Hàm ‘IsCount’: kiểm tra ngắt đếm (Counter interrupt).

- $t2 lưu giá trị ngắt đếm MASK_CAUSE_COUNTER
- So sánh $t1(nguyên nhân) với $t2. Nếu bằng nhau tức là ngắt đếm thì nhảy tới hàm ‘Counter_Intr’ để thực hiện ngắt đếm.
- Ngược lại thì thực hiện hàm ‘IsKeyMa’ để kiểm tra ngắt bàn phím.

Hàm ‘IsKeyMa’ : kiểm tra ngắt bàn phím (Key matrix interrupt)

- $t2 lưu giá trị ngắt đếm MASK_CAUSE_KEYMATRIX
- So sánh $t1(nguyên nhân) với $t2. Nếu bằng nhau tức là ngắt bàn phím thì nhảy tới hàm ‘Counter_Intr’ để thực hiện ngắt đếm.
- Ngược lại thì tới hàm ‘end_process’ để kết thúc với nguyên nhân ngắt khác với 2 nguyên nhân trên.

Hàm ‘Counter_Intr’ : với nguyên nhân ngắt là ngắt đếm thì in ra ‘Time inteval!’.

Hàm ‘Keymatrix_Intr’: in ra nút đã được bấm.

Hàm ‘end_process’: kết thúc kiểm tra ngắt và gán giá trị 0 vào thanh ghi $13 (Cause) để xóa nguyên nhân ngắt.

Hàm ‘en_int’: kích hoạt lại ngắt bằng cách gán giá trị thanh ghi đếm thời gian vào địa chỉ của nó.

Hàm next_pc: Cộng thêm 4 vào thanh $at để giá trị trả về là địa chỉ của lệnh tiếp theo.

Assignment 5:

Code:

```
#Lab 11, assignment 5
.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
.eqv MASK_CAUSE_KEYBOARD 0x0000034 # Keyboard Cause
.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
                beq $t1, $zero, WaitForKey # if $t1 == 0 then Polling
MakeIntR:       teqi $t1, 1 # if $t0 = 1 then raise an Interrupt
                j loop
#---------------------------------------------------------------
# Interrupt subroutine
#---------------------------------------------------------------
.ktext 0x80000180
get_caus:       mfc0 $t1, $13 # $t1 = Coproc0.cause
IsCount:        li $t2, MASK_CAUSE_KEYBOARD# if Cause value confirm Keyboard..
                and $at, $t1,$t2
                beq $at,$t2, Counter_Keyboard
                j end_process
Counter_Keyboard:
ReadKey:        lw $t0, 0($k0) # $t0 = [$k0] = KEY_CODE
WaitForDis:     lw $t2, 0($s1) # $t2 = [$s1] = DISPLAY_READY
                beq $t2, $zero, WaitForDis # if $t2 == 0 then Polling
Encrypt:        addi $t0, $t0, 1 # change input key
ShowKey:        sw $t0, 0($s0) # show key
                nop
end_process:
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

```

Giải thích:

KEY_CODE 0xFFFF0004: Địa chỉ thanh ghi lưu trữ mã ASCII của phím từ bàn phím, kích thước 1 byte. ($k0)

KEY_READY 0xFFFF0000: Địa chỉ thanh ghi cho biết liệu có mã ASCII mới từ bàn phím hay không, có giá trị bằng 1 nếu có, tự động xóa sau lệnh lw, kích thước 1 byte. ($k1)

DISPLAY_CODE 0xFFFF000C: Địa chỉ thanh ghi lưu trữ mã ASCII để hiển thị trên màn hình, kích thước 1 byte. ($s0)

DISPLAY_READY 0xFFFF0008: Địa chỉ thanh ghi cho biết liệu màn hình đã sẵn sàng để hiển thị hay chưa, có giá trị bằng 1 nếu đã sẵn sàng, tự động xóa sau lệnh sw, kích thước 1 byte. ($s1)

MASK_CAUSE_KEYBOARD: Mặt nạ (mask) để xác định nguyên nhân ngắt từ bàn phím.

Hàm ‘WaitForKey’: nếu giá trị trong thanh ghi $t1 bằng 0, tức là không có mã ASCII mới, thì quay lại nhãn WaitForKey để tiếp tục kiểm tra

Hàm ‘MakeIntR’: so sánh giá trị trong thanh ghi $t1 với 1. Nếu bằng nhau, tức là có mã ASCII mới từ bàn phím, thì gây ra một ngắt.

Hàm ‘get_caus’: đọc giá trị từ thanh ghi Cause vào thanh ghi $t1 để xác định nguyên nhân của ngắt.

Hàm ‘IsCount’: kiểm tra xem nguyên nhân của ngắt có phải là từ bàn phím hay không bằng cách so sánh giá trị trong thanh ghi $t1 với MASK_CAUSE_KEYBOARD.

Hàm ‘ReadKey’: đọc giá trị từ địa chỉ thanh ghi $k0 vào thanh ghi $t0 để lưu trữ mã ASCII của phím từ bàn phím .

Hàm ‘WaitForDis’:

- Đọc giá trị từ địa chỉ thanh ghi $s1 vào thanh ghi $t2 để kiểm tra xem màn hình đã sẵn sàng để hiển thị hay chưa.

- Nếu giá trị trong thanh ghi $t2 bằng 0, tức là màn hình chưa sẵn sàng, thì quay lại nhãn ‘WaitForDis’ để tiếp tục kiểm tra.

Hàm ‘Encrypt’: Tăng input key thêm 1 đơn vị để mã hóa trước khi in ra màn hình.

Hàm ‘ShowKey’: Ghi giá trị trong thanh ghi $t0 vào địa chỉ thanh ghi $s0 để hiển thị mã ASCII của phím trên màn hình.

Hàm ‘next_pc’ và ‘return’ để quay lại lệnh tiếp theo của chương trình chính.

Kết quả: