Bài thực hành số 11

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

Họ và tên: Đinh Thị Hồng Phúc **MSSV: 20215118**

Bài 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 $t4, 0x10
set:    li $t3, 0x01    #scan from row1
polling:
        sb $t3, 0($t1 ) # must reassign expected row
        lb $a0, 0($t2)  # read scan code of key button
        bne $a0, $0, print
        j sleep
print:  li $v0, 34      # print integer (hexa)
        syscall
sleep:  li $a0, 1000    # sleep 1000ms
        li $v0, 32
        syscall
nextrow:
        sll $t3, $t3, 1
        beq $t3, $t4, reset
        j polling
reset:  li $t3, 0x01
        j polling
```

*Thực hiện gõ chương trình vào công cụ **MARS***

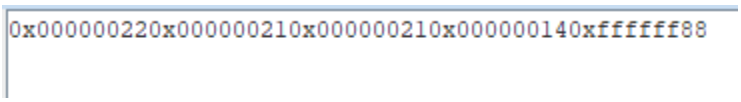
Giải thích: Chương trình thực hiện quá trình đọc và hiển thị mã quét của các phím trên bàn phím

- Đầu tiên, dùng *.eqv* để định nghĩa địa chỉ bắt đầu của đầu vào và đầu ra của bàn phím
- *main:* Gán giá trị đầu vào và đầu ra lần lượt vào thanh ghi $t1, $t2. Gán giá trị đại diện cho số lượng dòng (rows) vào thanh ghi $t4
- *set:* Gán giá trị đại diện cho dòng đầu tiên (0x01) vào thanh ghi $t3
- *polling:* Ban đầu ghi giá trị của thanh ghi $t3 vào ô nhớ tại địa chỉ $t1 (quét từng hàng một). Sau đó, lấy giá trị từ địa chỉ $t2 lưu vào thanh ghi $a0.

Kiểm tra giá trị trong thanh ghi $a0 có khác 0 hay không. Nếu khác nghĩa là 1 phím đã được nhấn, sau đó chương trình nhảy tới *print* để hiển thị mã quét của phím. Nếu không thì không có phím nào được nhấn và chương trình nhảy tới *sleep*, bỏ qua phần in mã quét

- *print:* Thực hiện in mã quét của phím ra màn hình. Thanh ghi $v0 được gán giá trị 34 để in số nguyên hệ hexa
- *sleep:* Gán giá trị 1000 vào vào thanh ghi $a0, là thời gian chờ (đơn vị là ms) trước khi chương trình tiếp tục chạy dòng tiếp theo. Gán giá trị 32 vào thanh ghi $v0 thực hiện hành động ngủ với thời gian được chỉ định trong thanh ghi $a0
- *nextrow:* Dịch trái giá trị trong thanh ghi $t3 để chuyển sang dòng tiếp theo. Kiểm tra giá trị trong $t3 có bằng giá trị trong $t4 hay không. Nếu bằng nghĩa là đã quét hết tất cả các dòng, chương trình sẽ nhảy đến *reset* để đặt lại giá trị ban đầu cho $t3 và tiếp tục quét từ dòng đầu tiên. Nếu không bằng, chương trình nhảy đến *polling* để tiếp tục quá trình quét và đọc mã phím từ các hàng

Nhập 5118f từ bàn phím:



*Thực hiện chạy chương trình với **MARS***

Bài 2.

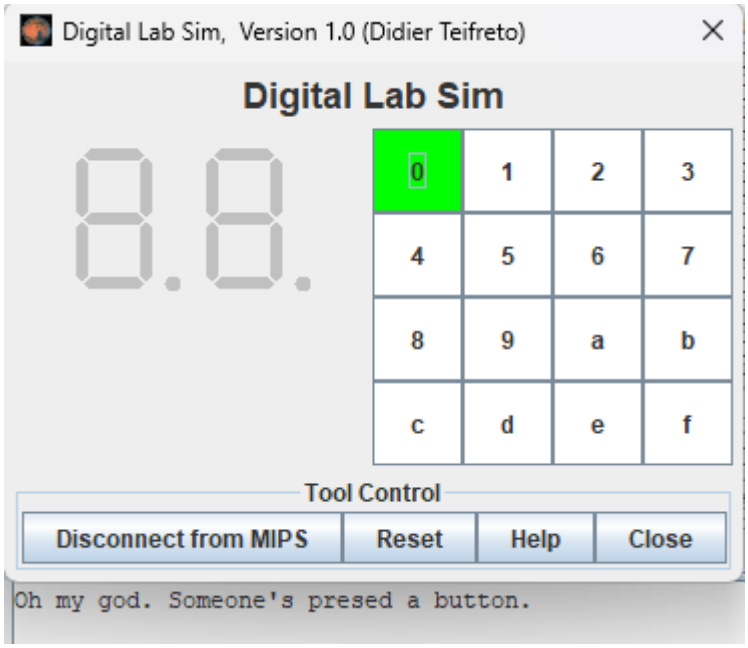
```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.data
        Message: .asciiz "Oh my god. Someone's presed a button.\n"
.text
main:   li $t1, IN_ADRESS_HEXA_KEYBOARD
        li $t3, 0x80    # bit 7 of = 1 to enable interrupt
        sb $t3, 0($t1)
Loop:   nop
        nop
        nop
        nop
        b Loop  # Wait for interrupt
end_main:
.ktext 0x80000180
IntSR:  addi $v0, $zero, 4      # show message
        la $a0, Message
        syscall
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
```

*Thực hiện gõ chương trình vào công cụ **MARS***

Giải thích: Đoạn code trên sử dụng ngắt (interrupt) để phát hiện khi nhấn nút trên bàn phím và hiển thị chuỗi *Message* khi có ngắt xảy ra.

- Dùng *.eqv* để định nghĩa hằng số, lưu địa chỉ cổng đầu vào của bàn phím
- *.data:* Khai báo chuỗi *Message* là *Oh my god. Someone's presed a button.\n*
- *main:* Gán giá trị địa chỉ cổng đầu vào vào thanh ghi $t1. Gán giá trị 0x80 vào thanh ghi $t3, giá trị này đại diện cho bit thứ 7 được đặt thành 1 để kích hoạt ngắt khi nhấn 1 nút. Sau đó ghi giá trị trong thanh ghi $t3 vào cổng đầu vào của bàn phím (bật chế độ kích hoạt ngắt trên bàn phím).
- *Loop:* Vòng lặp vô hạn để chờ xảy ra ngắt. Khi ngắt sẽ nhảy đến chương trình con xử lý ngắt *.ktext* ở địa chỉ bắt đầu là 0x80000180.
- *IntSR:* Gán giá trị 4 vào thanh ghi $v0 để hiển thị chuỗi kí tự *Message* ra màn hình.
- *next_pc:* Lấy địa chỉ lệnh tiếp sau khi xử lý ngắt. Đầu tiên sao chép giá trị thanh ghi $14 vào thanh ghi $at. Tăng giá trị $at lên 4, tương ứng với lệnh tiếp theo. Sau đó sao chép giá trị trong thanh ghi $at vào thanh ghi $14.
- Lệnh *eret* là lệnh trả về từ xử lý ngắt và tiếp tục chương trình ở lệnh tiếp theo sau ngắt.

Khi ấn nút sẽ hiển thị chuỗi kí tự *"Oh my god. Someone's presed a button."*



*Thực hiện chạy chương trình với **MARS***

Bài 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        # Save $ra because we may change it later
        sw $ra, 0($sp)
        addi $sp, $sp, 4        # Save $ra because we may change it later
        sw $at, 0($sp)
        addi $sp, $sp, 4        # Save $ra 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)
prn_msg:
        addi $v0, $zero, 4
        la $a0, Message
        syscall
        li $t3, 0x01
get_cod:
        li $t1, IN_ADRESS_HEXA_KEYBOARD
        ori $t4, $t3, 0x80
        sb $t4, 0($t1)  # must reassign expected row
        li $t1, OUT_ADRESS_HEXA_KEYBOARD
        lb $a0, 0($t1)
        bne $a0, $0, prn_cod
        sll $t3, $t3, 1
        j get_cod
```

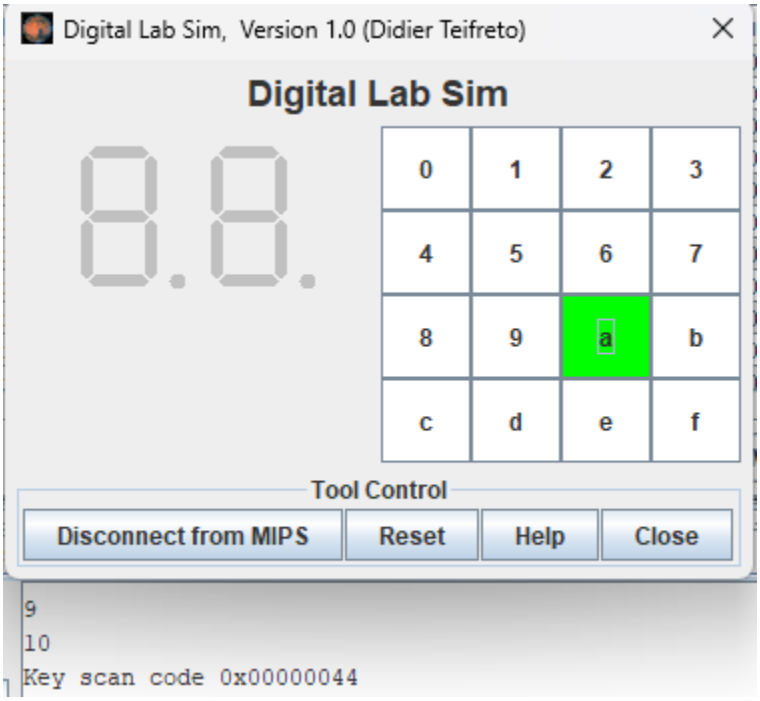
```
prn_cod:
        li $v0, 34
        syscall
        li $v0, 11
        li $a0, '\n'    # print endofline
        syscall
next_pc:
        mfc0 $at, $14   # $at <= Coproc0.$14 = Coproc0.epc
        addi $at, $at, 4        # $at = $at + 4 (next instruction)
        mtc0 $at, $14   # Coproc0.$14 = Coproc0.epc <= $at
restore:
        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
return: eret    # Return from exception
```

*Thực hiện gõ chương trình vào công cu **MARS***

Giải thích: Đoạn code trên sử dụng ngắt (interrupt) để phát hiện khi nhấn nút trên bàn phím và hiển thị mã quét (scan code) của nút đó.

- Dùng *.eqv* để định nghĩa hằng số, lưu địa chỉ cổng đầu vào (IN_ADRESS_HEXA_KEYBOARD là 0Xffff0012) và đầu ra (OUT_ADRESS_HEXA_KEYBOARD là 0Xffff0014) của bàn phím.
- *.data:* Khai báo chuỗi *Message* là *Key scan code*
- *main:* Gán giá trị địa chỉ cổng đầu vào của bàn phím vào thanh ghi $t1. Gán giá trị 0x80 vào thanh ghi $t3, giá trị này đại diện cho bit thứ 7 được đặt thành 1 để kích hoạt ngắt khi nhấn 1 nút. Sau đó ghi giá trị trong thanh ghi $t3 vào cổng đầu vào của bàn phím (bật chế độ kích hoạt ngắt trên bàn phím). Xóa giá trị thanh ghi $s0, thiết lập giá trị biến đếm ($s0) bằng 0.
- *Loop:* Tăng biến đếm $s0 thêm 1.
- *prn_seq:* Gán giá trị 1 vào thanh ghi $v0, để in ra số thứ tự của vòng lặp hiện tại (giá trị biến đếm).
- *prn_eol:* In ra kí tự xuống dòng '\n'.
- *sleep:* Gán giá trị 32 vào thanh ghi $v0 để tạm dừng chương trình trong 1 khoảng thời gian $a0 tính bằng ms (trong bài là dừng trong 300ms). Sau đó chương trình quay lại vòng *Loop* để tiếp tục quét mã phím và in ra số lần quét.
- *.ktext:* Chương trình con xử lý ngắt ở địa chỉ bắt đầu là 0x80000180.
- *IntSR:* Lưu trữ giá trị các thanh ghi trong ngăn xếp để đảm bảo không bị ghi đè trong quá trình xử lý ngắt: Dịch chuyển con trỏ ngăn xếp $sp lên 4 byte, để lưu trữ thanh ghi $ra vào địa chỉ con trỏ ngăn xếp hiện tại. Tương tự với các thanh ghi $at, $v0, $a0, $t1, $t3.
- *prn_msg:* In ra chuỗi *Message* và gán giá trị 0x01 vào thanh ghi $t3, là giá trị ban đầu để quét các hàng của bàn phím.
- *get_cod:* Thực hiện quét mã phím: Gán giá trị 0x80 vào thanh ghi $t4 sau đó ghi vào cổng đầu vào của bàn phím. Đọc và lưu giá trị từ cổng đầu ra vào thanh ghi $a0. Nếu giá trị đọc được khác 0, nghĩa là có phím được nhấn thì chương trình sẽ chuyển đến *prn_cod*. Nếu là 0 thì dịch trái thanh ghi $t3 để quét hàng tiếp theo (lặp lại vòng *get_cod*).
- *prn_cod:* In giá trị mã phím đã quét, sau đó in kí tự xuống dòng '\n'
- *next_pc:* Lấy địa chỉ của lệnh kế tiếp và gán vào thanh ghi $14
- *restore:* Lấy lại giá trị các thanh ghi từ ngăn xếp
- Dòng 73: Kết thúc xử lý ngắt, tiếp tục thực hiện chương trình sau ngắt.

Nhấn 1 nút màn hình sẽ hiện ra mã quét của nút đó



*Thực hiện chạy chương trình với **MARS***

Bài 4.

```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.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!\n"
        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)
        li $t1, COUNTER
        sb $t1, 0($t1)
Loop:   nop
        nop
        nop
sleep:  addi $v0, $zero, 32     # BUG: must sleep to wait for Time Counter
        li $a0, 300             # sleep 300 ms
        syscall
        nop                     # WARNING: nop is mandatory here.
        b Loop
end_main:
```

```
.ktext 0x80000180
IntSR:  #-----------------------------------------------------
        # Temporary disable interrupt
        #-----------------------------------------------------
dis_int:li $t1, COUNTER # BUG: must disable with Time Counter
        sb $zero, 0($t1)
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
Keymatrix_Intr:
        li $v0, 4       # Processing Key Matrix Interrupt
        la $a0, msg_keypress
        syscall
        j end_process

Counter_Intr:
        li $v0, 4       # Processing Counter Interrupt
        la $a0, msg_counter
        syscall
        j end_process
end_process:
        mtc0 $zero, $13 # Must clear cause reg
en_int: li $t1, COUNTER
        sb $t1, 0($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
```

*Thực hiện gõ chương trình vào công cụ **MARS***

Giải thích: Thực hiện việc xử lý ngắt từ bàn phím và từ bộ đếm thời gian

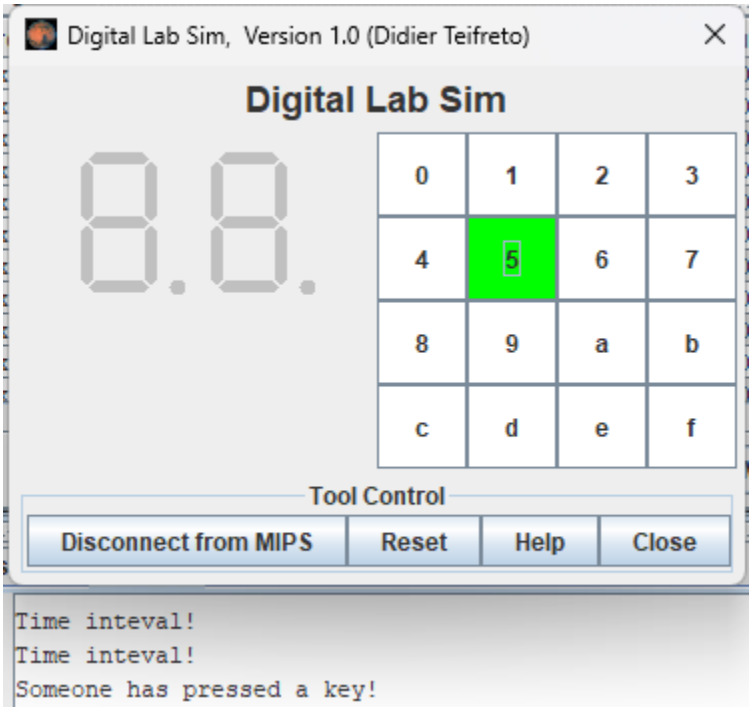
Thanh ghi số 13, status trong bộ đồng xử lý C0, chứa các thiết lập về tình trạng ngắt.

	15	14	13	12	11	10			6	5	4	3	2	1	0
					KM	TC			Exception Code					K/U	IE
IE = 1 cho phép ngắt. IE = 0 vô hiệu hóa mọi hoạt động ngắt															
K/U=1 hoạt động ở chế độ Kernel. K/U=0 hoạt động ở chế độ User															
Ngoại lệ do syscall, overflow, lệnh tạo ngắt mềm như teq teqi…															
Time Counter bộ đếm thời gian															
Key Matrix															

- Dùng *.eqv* để định nghĩa các hằng số địa chỉ của cổng đầu vào của bàn phím, địa chỉ của thanh ghi đếm thời gian, các bit trong thanh ghi cause để kiểm tra nguyên nhân ngắt (bit 10 cho ngắt từ bộ đếm và 11 cho ngắt từ bàn phím).
- *.data:* Khai báo 2 chuỗi để hiển thị khi ngắt từ bộ đếm và ngắt từ bàn phím.
- *main:* Gán giá trị địa chỉ cổng đầu vào của bàn phím vào thanh ghi $t1. Gán giá trị 0x80 vào thanh ghi $t3, giá trị này đại diện cho bit thứ 7 được đặt thành 1 để kích hoạt ngắt khi nhấn 1 nút. Sau đó ghi giá trị trong thanh ghi $t3 vào cổng đầu vào của bàn phím (bật chế độ kích hoạt ngắt trên bàn phím). Giá trị hằng số COUNTER gán vào thanh ghi $t1, lưu giá trị của $t1 vào $t1 (để kích hoạt time counter)
- *sleep:* Gán giá trị 32 vào thanh ghi $v0 để tạm dừng chương trình trong 1 khoảng thời gian $a0 tính bằng ms (trong bài là dừng trong 300ms). Sau đó chương trình quay lại vòng *Loop*.
- *.ktext:* Chương trình con xử lý ngắt bắt đầu ở địa chỉ 0x80000180.
- *dis_int:* Thực hiện tắt ngắt từ bộ đếm thời gian bằng cách ghi giá trị 0 vào thanh ghi đếm thời gian ($t1).
- *get_caus:* Kiểm tra nguyên nhân ngắt bằng cách đọc giá trị của thanh ghi $13 vào thanh ghi $t1
- *IsCount:* Kiểm tra xem ngắt có phải từ bộ đếm thời gian hay không. Nếu đúng sẽ chuyển đến *Counter_Intr*. Nếu không tiếp tục kiểm tra có phải từ bàn phím hay không bằng chương trình *IsKeyMa*. Nếu là ngắt từ bàn phím thì chuyển đến *Keymatrix_Int*. Nếu cũng không phải ngắt từ bàn phím thì nhảy đến *end_process*.
- *Keymatrix_Int, Counter_Int:* In ra chuỗi ngắt bằng thời gian và bàn phím tương ứng, sau đó chuyển đến *end_process*.
- *end_process:* Xóa thanh ghi $13 (đặt giá trị 0) để chuẩn bị cho lần ngắt tiếp theo.

- *en_int:* Thực hiện kích hoạt lại ngắt từ bộ đếm thời gian
- *next_pc:* Cập nhật giá trị thanh ghi $14 để chỉ đến lệnh tiếp theo.
- *return:* Trở về từ xử lý ngắt và tiếp tục chương trình từ địa chỉ tiếp theo.

Khi bình thường, chương trình sẽ báo ngắt do bộ đếm thời gian, khi nhấn nút chương trình sẽ báo ngắt do bàn phím.



*Thực hiện chạy chương trình với **MARS***

Bài 5.

```
1  .eqv KEY_CODE 0xFFFF0004        # ASCII code from keyboard, 1 byte
2  .eqv KEY_READY 0xFFFF0000       # =1 if has a new keycode ?
3                                  # Auto clear after lw
4  .eqv DISPLAY_CODE 0xFFFF000C    # ASCII code to show, 1 byte
5  .eqv DISPLAY_READY 0xFFFF0008   # =1 if the display has already to do
6                                  # Auto clear after sw
7  .eqv MASK_CAUSE_KEYBOARD 0x0000034      # Keyboard Cause
8  .text
9          li $k0, KEY_CODE
10         li $k1, KEY_READY
11
12         li $s0, DISPLAY_CODE
13         li $s1, DISPLAY_READY
14 loop:   nop
15 WaitForKey:
16         lw $t1, 0($k1)  # $t1 = [$k1] = KEY_READY
17         beq $t1, $zero, WaitForKey      # if $t1 == 0 then Polling
18 MakeIntR:
19         teqi $t1, 1     # if $t0 = 1 then raise an Interrupt
20         j loop
```

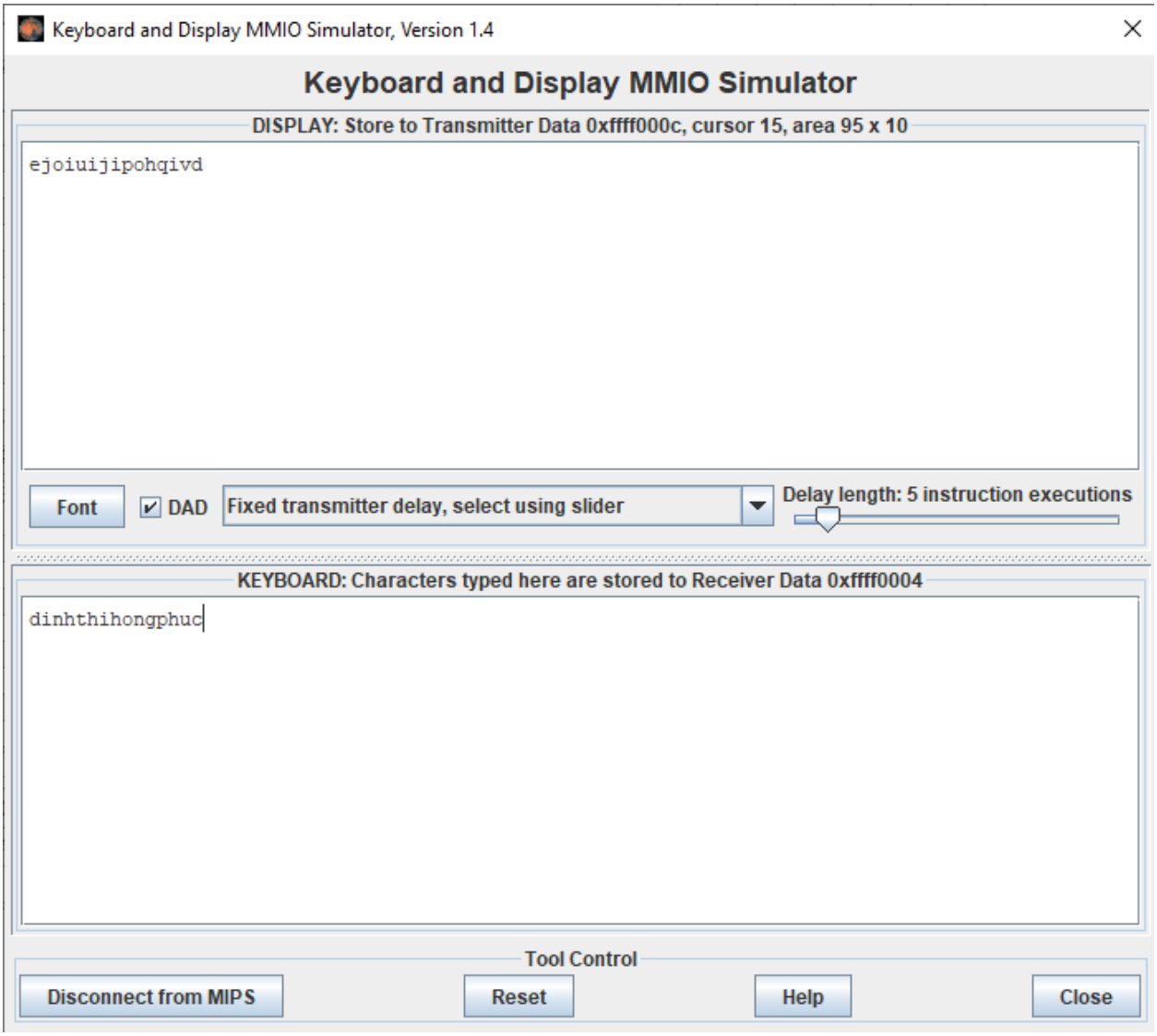
```
21 .ktext 0x80000180
22 get_caus:
23         mfc0 $t1, $13   # $t1 = Coproc0.cause
24 IsCount:
25         li $t2, MASK_CAUSE_KEYBOARD# if Cause value confirm Keyboard..
26         and $at, $t1, $t2
27         beq $at, $t2, Counter_Keyboard
28         j end_process
29 Counter_Keyboard:
30 ReadKey:
31         lw $t0, 0($k0)  # $t0 = [$k0] = KEY_CODE
32 WaitForDis:
33         lw $t2, 0($s1)  # $t2 = [$s1] = DISPLAY_READY
34         beq $t2, $zero, WaitForDis      # if $t2 == 0 then Polling
35 Encrypt:
36         addi $t0, $t0, 1        # change input key
37 ShowKey:
38         sw $t0, 0($s0)  # show key
39         nop
40 end_process:
```

```
41 next_pc:
42         mfc0 $at, $14   # $at <= Coproc0.$14 = Coproc0.epc
43         addi $at, $at, 4        # $at = $at + 4 (next instruction)
44         mtc0 $at, $14   # Coproc0.$14 = Coproc0.epc <= $at
45 return: eret    # Return from exception
```

*Thực hiện gõ chương trình vào công cụ **MARS***

Giải thích: Thực hiện xử lý ngắt từ bàn phím và hiển thị kí tự lên màn hình

- Dùng *.eqv* định nghĩa các hằng số tương ứng với địa chỉ của thanh ghi lưu mã ASCII từ bàn phím, thanh ghi kiểm tra sẵn sàng nhận mã ASCII từ bàn phím, thanh ghi lưu mã ASCII để hiển thị lên màn hình và thanh ghi kiểm tra sẵn sàng hiển thị. Và định nghĩa hằng số tương ứng với giá trị trong thanh ghi cause để kiểm tra nguyên nhân ngắt từ bàn phím.
- *WaitForKey:* Kiểm tra có mã ASCII mới từ bàn phím không: Đọc giá trị từ thanh ghi $k1 vào thanh ghi $t1. Nếu giá trị này bằng 0, nghĩa là không có kí tự được nhập, chương trình tiếp tục đợi. Sau khi có mã ASCII mới từ bàn phím, chương trình chuyển đến *MakeIntR.* Dòng 19 được dùng để so sánh giá trị thanh ghi $t1 với 1, nếu bằng nhau chương trình sẽ gây ra 1 ngắt. Nếu không chương trình quay lại vòng *loop* để tiếp tục chờ mã từ bàn phím.
- *.ktext:* Chương trình con xử lý ngắt bắt đầu ở địa chỉ 0x80000180.
- *get_caus:* Dùng để đọc giá trị thanh ghi cause, xác định nguyên nhân của ngắt.
- *IsCount:* Kiểm tra ngắt có phải từ bàn phím hay không bằng cách so sánh giá trị của thanh ghi $t1 với *MASK_CAUSE_KEYBOARD*. Nếu là ngắt từ bàn phím, chương trình sẽ chuyển đến *Counter_Keyboard.*
- *Counter_Keyboard:* Đọc giá trị mã ASCII từ bàn phím và đợi màn hình sẵn sàng để hiển thị bằng cách kiểm tra giá trị của thanh ghi $s1 (*DISPLAY_READY*). Nếu giá trị này bằng 0, nghĩa là màn hình chưa sẵn sàng, chương trình tiếp tục đợi bằng lệnh nhảy đến *WaitForDis.*
- Sau khi màn hình sẵn sàng, chương trình thực hiện các bước xử lý dữ liệu, như mã hoá (*Encrypt*) và hiển thị mã ASCII lên màn hình (*ShowKey*).
- Sau khi hoàn thành xử lý, chương trình quay lại *end_process* để chuẩn bị cho lần ngắt tiếp theo. *next_pc* được sử dụng để cập nhập giá trị của thanh ghi $14 và lệnh *eret* được sử dụng để trở về từ xử lý ngắt, tiếp tục chương trình từ câu lệnh kế tiếp.



*Thực hiện chạy chương trình với **MARS***