BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH

LAB 11

Họ và tên: Phạm Vân Anh

MSSV: 20214988

Mã lớp: 139365

ASSIGNMENT 1:

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_ADRESS_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_ADRESS_HEXA_KEYBOARD 0xFFFF0014
```

```
.text
main:
	li $s1, IN_ADRESS_HEXA_KEYBOARD
	li $s2, OUT_ADRESS_HEXA_KEYBOARD
polling:
	li $t0, 0x01 # check row 1 with key 0, 1, 2, 3
check:
	beq $t0, 0x10, print
	sb $t0, 0($s1) # must reassign expected row
	lb $a0, 0($s2) # read scan code of key button
	bne $a0, 0, print
	sll $t0, $t0, 1
	j check
print:
	li $v0, 34 # print integer (hexa)
	syscall
	li $v0, 11
	li $a0, 10
	syscall
sleep:
	li $a0, 100 # sleep 100ms
	li $v0, 32
	syscall
back_to_polling:
	j polling # continue polling
```

Thực hiện gõ chương trình vào công cụ MARS

2. Giải thích

- Chương trình in ra màn hình mã scan của số được bấm trên Digital Lab Sim qua ma trận sau

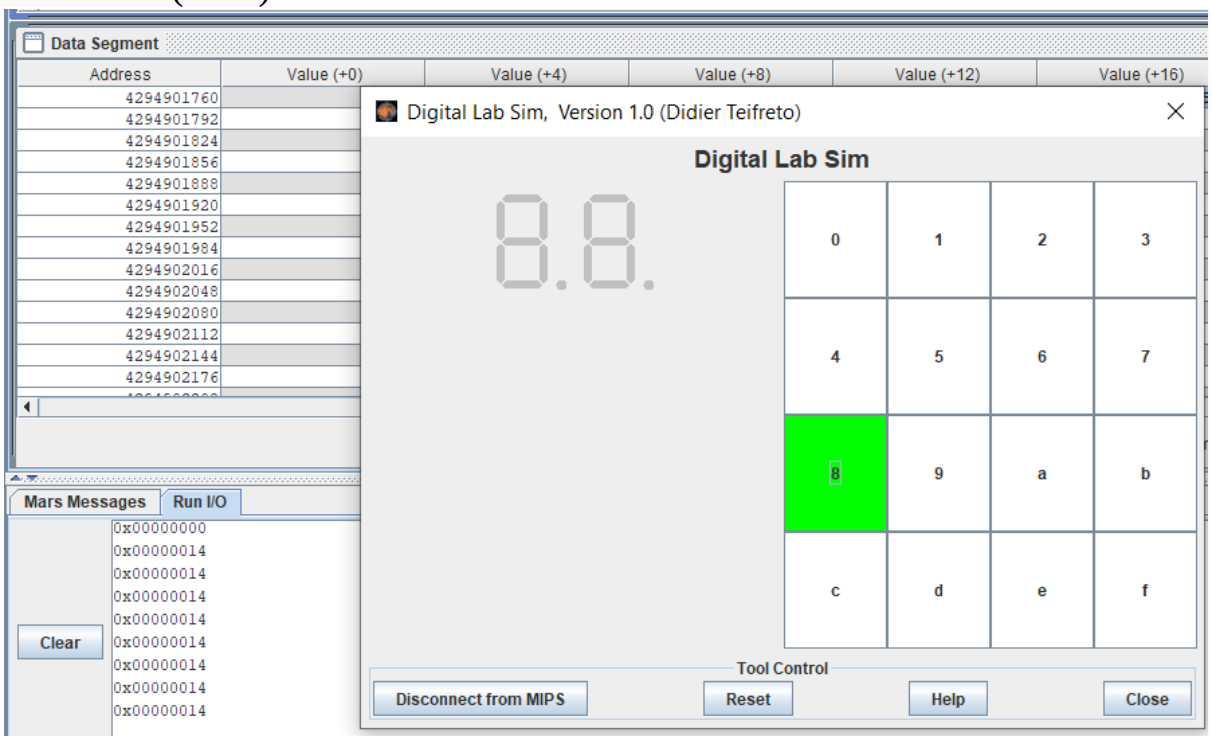
	0x10	0x20	0x30	0x40
0x01	0	1	2	3
0x02	4	5	6	7
0x03	8	9	A	B
0x04	C	D	E	F

- IN_ADDRESS_HEXA_KEYBOARD 0xFFFF0012: địa chỉ đầu vào
- OUT_ADDRESS_HEXA_KEYBOARD 0Xffff0014: địa chỉ đầu ra nhận giá trị hàng và cột của phím được bấm.

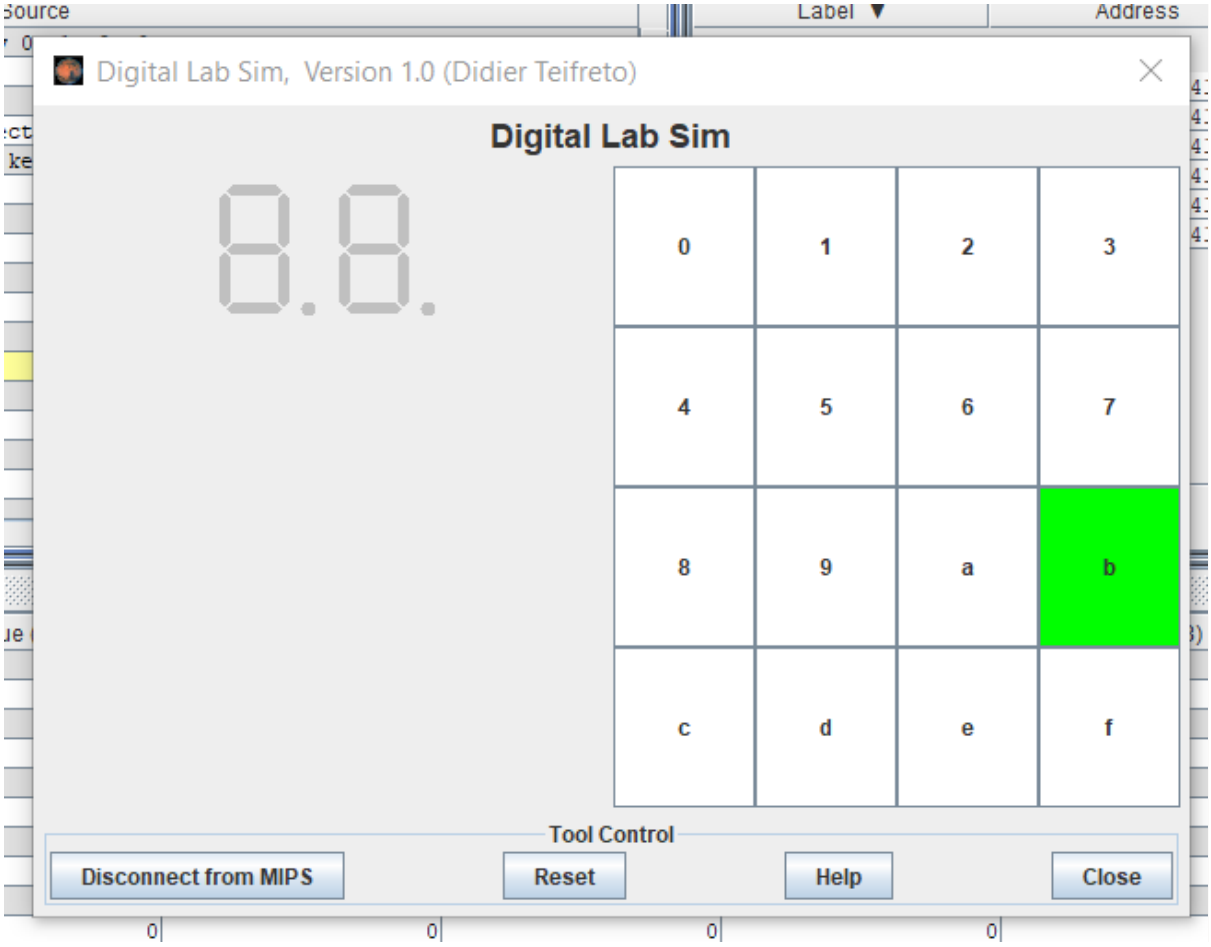
- Hàm *main:*
  - Gán địa chỉ vào các thanh ghi $s1 và $s2 tương ứng
  - Bắt đầu vòng lặp gán nhãn *polling* (kiểm tra)
- Hàm *polling:* vòng lặp kiểm tra từng phần tử trong hàng
- Hàm *check:* vòng lặp kiểm tra từng cột
  - Nếu $t0 = 0x10 (đã kiểm tra hết 4 cột) → *print*
  - Nếu không, nó ghi giá trị trong thanh ghi $t0 vào địa chỉ được lưu trong thanh ghi $t1 → gán hàng tiếp cho việc kiểm tra
  - Đọc giá trị từ địa chỉ lưu trong $t2 vào $t0 → đọc mã quét
  - Nếu $a0 != 0 (đã nhận được giá trị từ bàn phím) → print
  - Nếu không, nó dịch sang trái giá trị trong $t0 lên 1 (sang cột tiếp theo) → *check*
- Hàm *print:* in ra giá trị địa chỉ nút được bấm

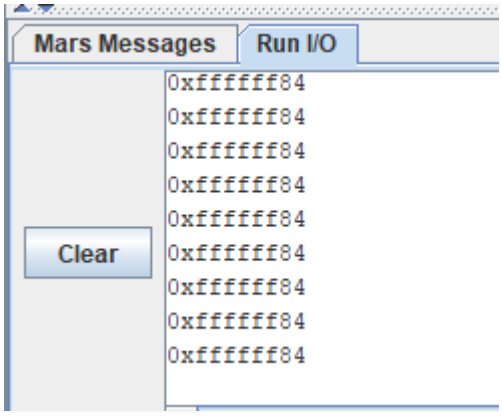
3. Kết quả

- Thử ấn số 8(0x14)



- Thử ấn b


Source
Label
Address
Digital Lab Sim, Version 1.0 (Didier Teifreto)
Digital Lab Sim
8.8.
0
1
2
3
4
5
6
7
8
9
a
b
c
d
e
f
Tool Control
Disconnect from MIPS
Reset
Help
Close


Mars Messages
Run I/O
Clear
0xffffff84
0xffffff84
0xffffff84
0xffffff84
0xffffff84
0xffffff84
0xffffff84
0xffffff84
0xffffff84

ASSIGNMENT 2:

1. Code

```
#Laboratory Exercise 11 Home Assignment 2
#PhamVanAnh_20214988
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.data
Message: .asciiz "Oh my god. Someone's presed a button.\n"
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# MAIN Procedure
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.text
main:
#-------------------------------------------------------------
# Enable interrupts you expect
#-------------------------------------------------------------
# Enable the interrupt of Keyboard matrix 4x4 of Digital Lab
#Sim
      li $t1, IN_ADRESS_HEXA_KEYBOARD
      li $t3, 0x80 # bit 7 of = 1 to enable interrupt
      sb $t3, 0($t1)
#-------------------------------------------------------------
# No-end loop, main program, to demo the effective of
#interrupt
#-------------------------------------------------------------
Loop:
      nop
      nop
      nop
      nop
      b     Loop # Wait for interrupt
end_main:
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# GENERAL INTERRUPT SERVED ROUTINE for all interrupts
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.ktext 0x80000180
#---------------------------------------------------------------
```

```
35  # Processing
36  #---------------------------------------------------------
37  IntSR:
38        addi $v0, $zero, 4 # show message
39        la $a0, Message
40        syscall
41  #---------------------------------------------------------
42  # Evaluate the return address of main routine
43  # epc <= epc + 4
44  #---------------------------------------------------------
45  next_pc:
46        mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
47        addi $at, $at, 4 # $at = $at + 4 (next instruction)
48        mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
49  return:
50        eret # Return from exception
```

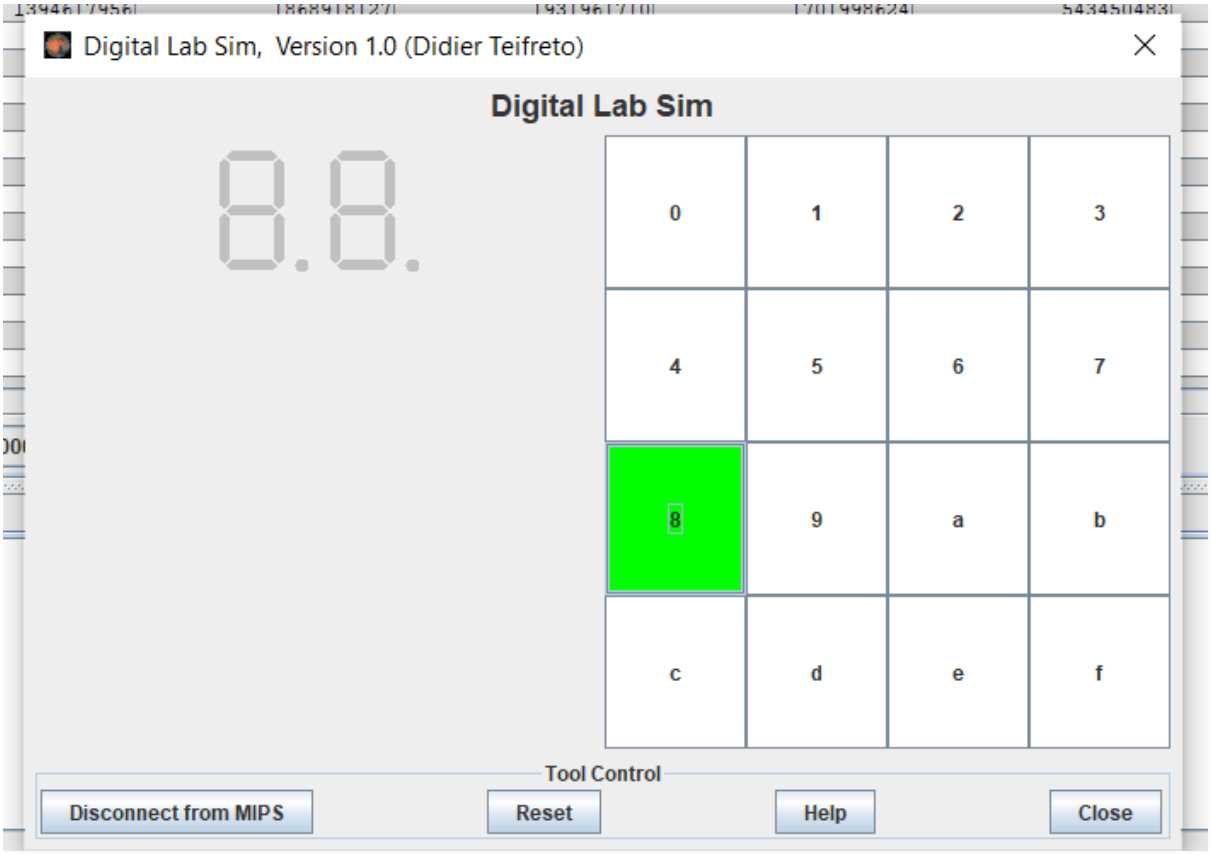
Thực hiện gõ chương trình vào công cụ MARS

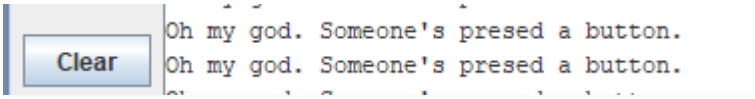
2. Giải thích

- Chương trình minh họa cho chương trình ngắt trên Digital Lab Sim
- Khi chương trình nhận tín hiệu ngắt, tự động nhảy tới địa chỉ *.ktext 8000180* để thực hiện chương trình con tự động ngắt. Sau khi thực hiện chương trình con, quay lại chương trình chính bằng cách nhảy tới địa chỉ của thanh ghi $14 (thanh ghi $co) nhưng lệnh tại địa chỉ của thanh ghi $14 đã thực hiện xong nên phải +4 để thực hiện lệnh tiếp theo.
- Hàm *main:*
    - Kích hoạt phím ngắt bằng cách ghi giá trị 0x80 vào thanh ghi $t3 và lưu vào địa chỉ được lưu trong $t1
- Hàm *loop:* Vòng lặp chờ sự kích hoạt ngắt từ bàn phím. Khi một nút được bấm (nhận được tín hiệu) → xử lí ngắt (Interript Servica Routine)

3. Kết quả

- Khi bấm số 8:





ASSIGNMENT 3:

1. Code

```
#Laboratory Exercise 11 Home Assignment 3
#PhamVanAnh_20214988
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014
.data
Message: .asciiz "Key scan code "
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# MAIN Procedure
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.text
main:
 #-----------------------------------------------------------
 # Enable interrupts you expect
 #-----------------------------------------------------------
 # Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim
      li    $t1, IN_ADRESS_HEXA_KEYBOARD
      li    $t3, 0x80 # bit 7 = 1 to enable
      sb    $t3, 0($t1)
#-----------------------------------------------------------
 # Loop an print sequence numbers
 #-----------------------------------------------------------
      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
```

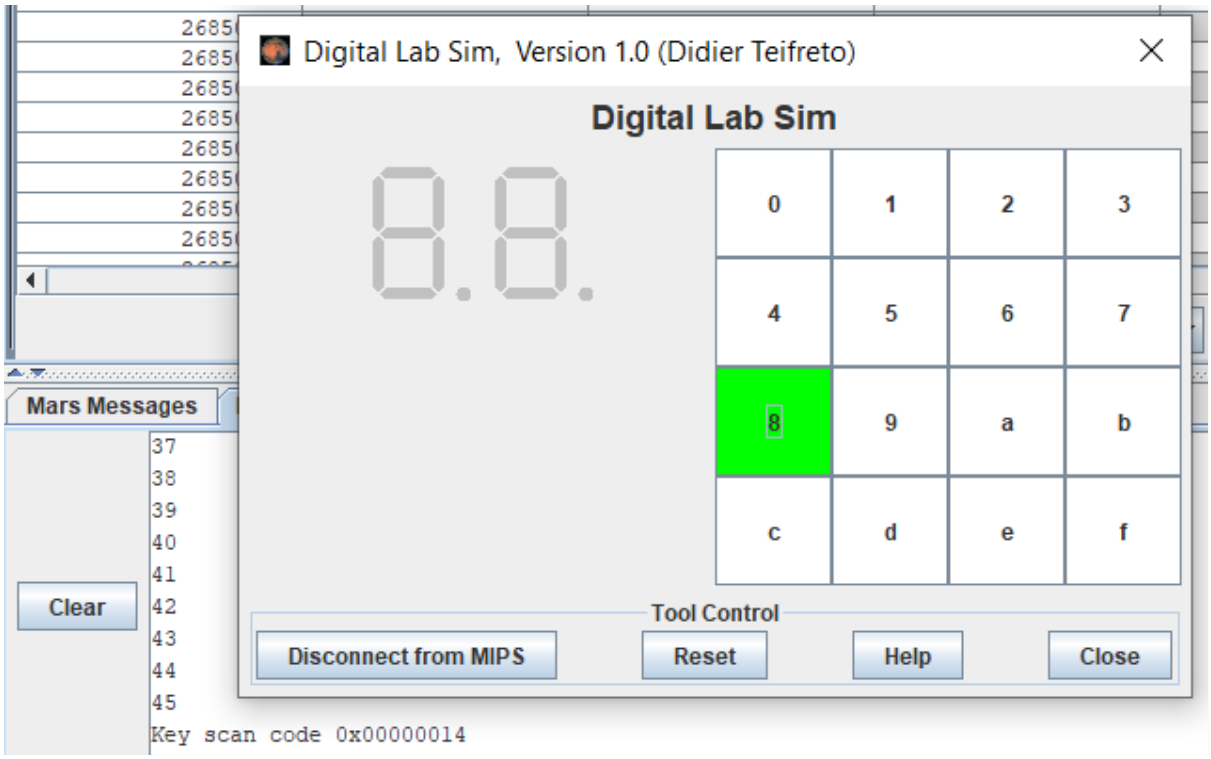
```
sleep:
      addi  $v0,$zero,32
      li    $a0,300 # sleep 300 ms
      syscall
      nop # WARNING: nop is mandatory here.
      b Loop # Loop
end_main:
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# GENERAL INTERRUPT SERVED ROUTINE for all interrupts
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.ktext 0x80000180
 #---------------------------------------------------------
 # SAVE the current REG FILE to stack
 #---------------------------------------------------------
IntSR:
      addi  $sp, $sp, 4 # Save $ra because we may change it later
      sw    $ra, 0($sp)
      sw    $at, 0($sp)
      addi  $sp, $sp, 4 # Save $v0 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)
 #---------------------------------------------------------
 # Processing
 #---------------------------------------------------------
prn_msg:
      addi  $v0, $zero, 4
      la    $a0, Message
      syscall
```

```
65        syscall
66        li    $t3, 0x81 # check row 1
67  get_cod:
68        li    $t1, IN_ADRESS_HEXA_KEYBOARD
69        sb    $t3, 0($t1) # must reassign expected row
70        li    $t1, OUT_ADRESS_HEXA_KEYBOARD
71        lb    $a0, 0($t1)
72        bne   $a0, 0, prn_cod
73        add   $t3, $t3, -128
74        sll   $t3, $t3, 1
75        add   $t3, $t3, 128
76        j get_cod
77  prn_cod:
78        add   $t3, $t3, 0x80
79        li    $v0,34
80        syscall
81        li    $v0,11
82        li    $a0,'\n' # print endofline
83        syscall
84  #---------------------------------------------------------
85  # Evaluate the return address of main routine
86  # epc <= epc + 4
87  #---------------------------------------------------------
88  next_pc:
89        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  #---------------------------------------------------------
93  # RESTORE the REG FILE from STACK
94  #---------------------------------------------------------
```

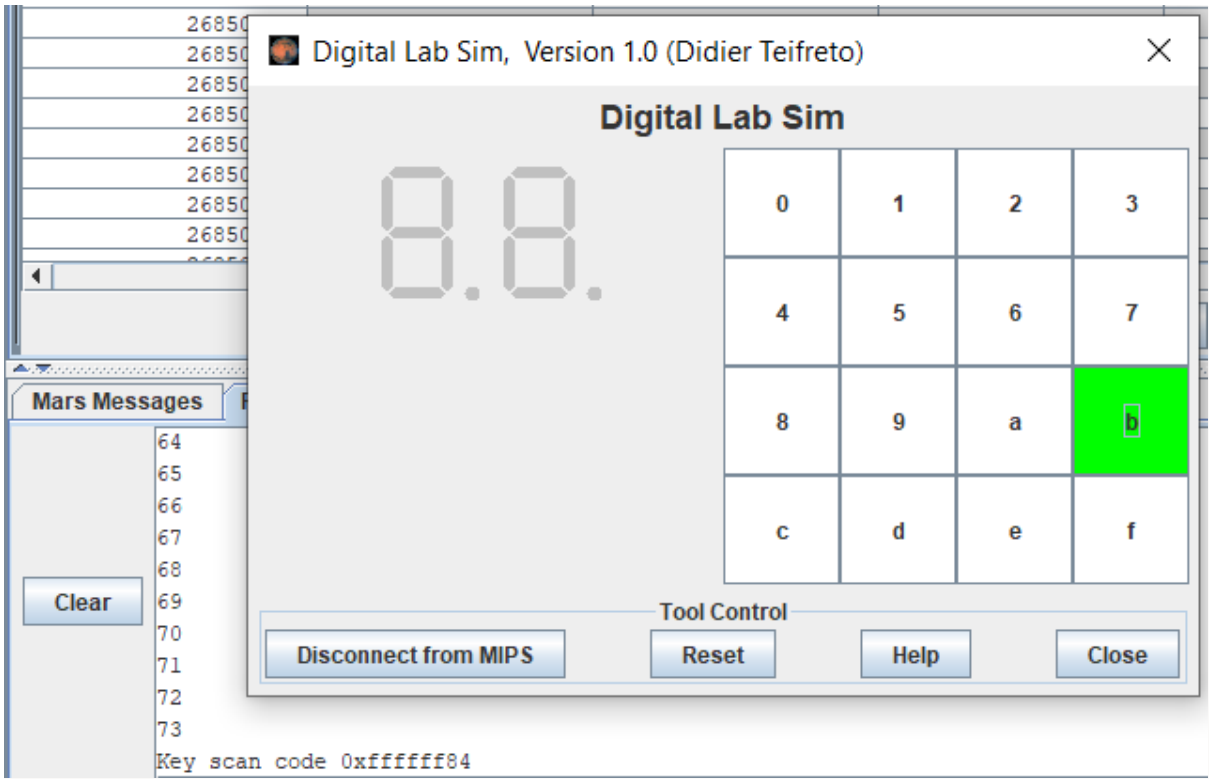
```
94  #---------------------------------------------------------
95  restore:
96        lw    $t3, 0($sp) # Restore the registers from stack
97        addi  $sp,$sp,-4
98        lw    $t1, 0($sp) # Restore the registers from stack
99        addi  $sp,$sp,-4
100       lw    $a0, 0($sp) # Restore the registers from stack
101       addi  $sp,$sp,-4
102       lw    $v0, 0($sp) # Restore the registers from stack
103       addi  $sp,$sp,-4
104       lw    $ra, 0($sp) # Restore the registers from stack
105       addi  $sp,$sp,-4
106 return:
107       eret # Return from exception
```

Thực hiện gõ chương trình vào công cụ MARS

2. **Giải thích**
   - Chương trình đếm và in ra màn hình các số mà ta bấm trên ma trận phím sử dụng ngắt.
   - Chương trình có 2 phần chính là .text để đếm và in ra màn hình, .ktext 0x80000180 xử lý chính khi gặp ngắt (khi bấm 1 phím trên bàn phím)
   - Khởi tạo IN_ADDRESS 0xFFFF0012 và OUT_ADDRESS 0xFFFF0014, gán IN_ADDRESS VÀO $t1
   - $s0 là biến đếm. Vòng lặp *loop* tăng đếm thêm 1, sau đó in ra màn hình → delay 300ms. Sau khi thực hiện xong ta sẽ nhảy lại về loop
   - Sau khi nhận được lệnh ngắt (ấn 1 phím ở trên ma trận), nó nhảy tới địa chỉ 0x80000180 và thực hiện chương trình con ngắt ở đó.
   - Lưu $ra, $at, $v0, $a0, $t1, $t3 vào trong stack ở *IntSR* vì ta sẽ có thể thay đổi chúng trong chương trình con → kết thúc ta trả lại giá trị cho chúng bằng cách pop các phần tử lần lượt ra.
   - In ra màn hình thông báo kèm mã của số ta vừa bấm vào. Xác định số vừa bấm vào giống ASSIGNMENT 1.
   - Thực hiện lấy địa chỉ quay về thanh ghi $14 vào $at, sau đó cộng thêm 4 và gán ngược lại vào thanh ghi $14 để quay lại chương trình chính và thực hiện lại từ đầu
3. **Kết quả**
   - Ấn số 8



- Ấn b


Digital Lab Sim, Version 1.0 (Didier Teifreto)
Digital Lab Sim
0
1
2
3
4
5
6
7
8
9
a
b
c
d
e
f
Tool Control
Disconnect from MIPS
Reset
Help
Close
Mars Messages
Clear
64
65
66
67
68
69
70
71
72
73
Key scan code 0xffffff84

ASSIGNMENT 4:

1. Code

```
#Laboratory Exercise 11 Home Assignment 4
#PhamVanAnh_20214988
.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"
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#MAIN Procedure
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.text
main:
#-------------------------------------------------------------
#Enable interrupts you expect
#-------------------------------------------------------------
#Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim
      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 300 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)
# no need to disable keyboard matrix interrupt
#-----------------------------------------------------------
```

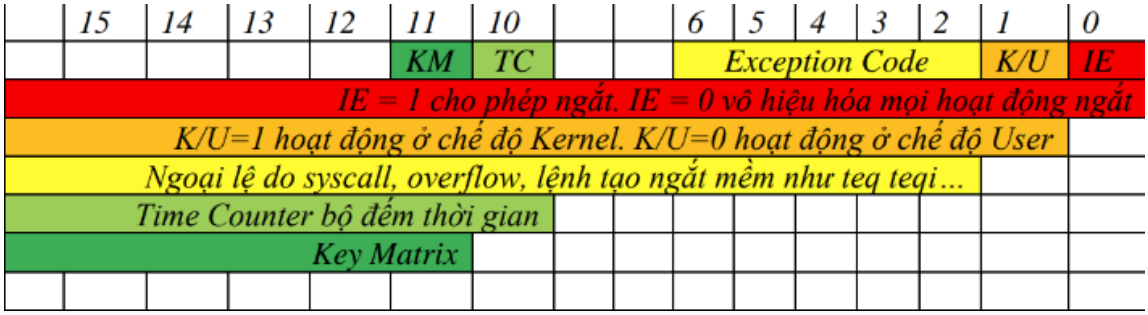
```
# Processing
#-----------------------------------------------------------
get_caus:
      mfc0 $t1, $13 # $t1 = Coproc0.cause
IsCount:
      li $t2, MASK_CAUSE_COUNTER# if Cause value confirm Counter..
      and $at, $t1, $t2
      beq $at, $t2, Counter_Intr
IsKeyMa:
      li $t2, MASK_CAUSE_KEYMATRIX # if Cause value confirm Key..
      and $at, $t1,$t2
      beq $at,$t2, Keymatrix_Intr
others: j end_process # other cases
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: #-----------------------------------------------------------
# Re-enable interrupt
#-----------------------------------------------------------
      li $t1, COUNTER
      sb $t1, 0($t1)
#-----------------------------------------------------------
# Evaluate the return address of main routine
# epc <= epc + 4
```

```
83  #---------------------------------------------------------
84  next_pc:
85        mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
86        addi $at, $at, 4 # $at = $at + 4 (next instruction)
87        mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
88  return:
89        eret # Return from exception
90
```

Thực hiện gõ chương trình vào công cụ MARS

2. Giải thích

- Chương trình mô tả cách minh họa bằng nhiều cách ngắt, ngắt bằng ma trận phím và Bộ đếm thời gian. Nguyên nhân ngắt nằm ở thanh ghi $13 trong CO



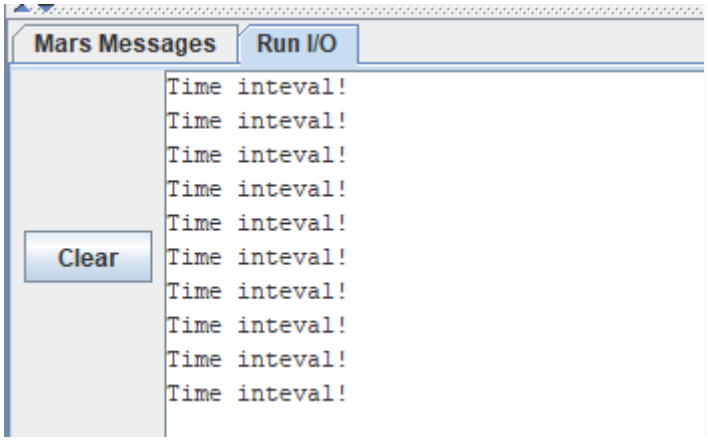
- Nếu ngắt bằng Ma trận phím thì thanh ghi $13 có giá trị 0x800 hay bit 10 là 1, ngắt bằng Time Counter thì thanh ghi $13 có giá trị 0x400 hay bit 11 là 1.
- Đầu tiên, ta khởi tạo giá trị MASK_CAUSE_COUNTER (ngắt bằng Time Counter) 0x00000400 và MASK_CAUSE_KEYMATRIX (ngắt bằng Ma trận phím) 0x00000800
- Hàm *main:*
  - Ngắt bằng Ma trận phím
    - Gán $t1 = IN_ADDRESS_HEXA KEYBOARD, gán $t3 = 0x80 (bit 7 = 1) cho phép chương trình ngắt bằng ma trận phím.
    - Gán giá trị của thanh ghi $t3 vào địa chỉ được trỏ bởi thanh ghi $t1
  - Ngắt bằng Time Counter
    - Gán $t1 = COUNTER, cho phép chương trình ngắt bằng bộ đếm thời gian
    - Gán giá trị ở $t1 vào chính địa chỉ được lưu ở $t1
- Sau đó, ta có vòng lặp *Loop* với vài lệnh **nop** và 1 lệnh **sleep,** sau đó quay lại vòng lặp *loop* để đợi cho ngắt xảy ra. Khi quá thời gian

đếm hay bấm phím nào trong ma trận, chương trình nhảy tới chương trình con phục vụ ngắt ở *.ktext* 0x80000180.

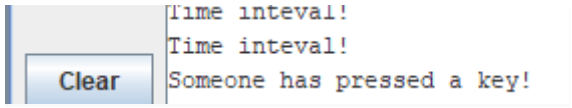
- Hàm *get_caus:* lấy ra giá trị của thanh ghi $13 để biết được nguyên nhân ngắt.
- Hàm *IsCount:* check xem có phải nguyên nhắn ngắt là do bộ đếm thời gian hay không
  - Gán $t2 = MASK_CAUSE_COUNTER 0x00000400 (nguyên nhân ngắt do Time Counter)
  - Làm phép toán AND giữa $t1 và $t2, lưu kết quả ra thành $at
  - Nếu $at = $t2 = MASK_CAUSE_COUNTER 0x00000400 → ngắt do bộ đếm thời gian → in ra kết quả
  - Nếu $at != $t2 → check tiếp
- Hàm *IsKeyMa:* check xem có phải nguyên nhân ngắt là do Ma trận phím.
  - Thực hiện tương tự như hàm *IsCount*
  - Nếu $at = $t2 = MASK_CAUSE_KEYMATRIX 0x00000800 → ngắt do Ma trận phím → In ra kết quả
  - Nếu $at != $t2 → *end_process*
- Sau khi in thông báo, ta sẽ xóa dữ liệu thanh ghi $13 bằng cách lưu $13 = 0. Sau đó, ta lưu lại $t1 vào địa chỉ ở $t1 để bật lại cho phép ngắt bằng thời gian.
- Cuối cùng, ta sẽ quay lại chương trình chính bằng cách trích xuất giá trị ở thanh ghi $14, cộng thêm 4 sau đó lưu lại → dùng lệnh **eret** để quay lại chương trình chính.

3. Kết quả

- Không bấm gì:



- Ấn số 8:



ASSIGNMENT 5:

1. Code

```
.eqv IN_ADDRESS_HEXA_KEYBOARD 0xFFFF0012
.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
      li $s2, IN_ADDRESS_HEXA_KEYBOARD
      li $t0, 0x80
      sb $t0, 0($s2)
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
#---------------------------------------------------------------
```

```
25  # Interrupt subroutine
26  #---------------------------------------------------------------
27  .ktext 0x80000180
28  get_caus: mfc0 $t1, $13 # $t1 = Coproc0.cause
29  IsCount:
30        li $t2, MASK_CAUSE_KEYBOARD# if Cause value confirm Keyboard..
31        and $at, $t1,$t2
32        beq $at,$t2, Counter_Keyboard
33        j end_process
34  Counter_Keyboard:
35  ReadKey: lw $t0, 0($k0) # $t0 = [$k0] = KEY_CODE
36  WaitForDis:
37        lw $t2, 0($s1) # $t2 = [$s1] = DISPLAY_READY
38        beq $t2, $zero, WaitForDis # if $t2 == 0 then Polling
39  Encrypt:
40        addi $t0, $t0, 1 # change input key
41  ShowKey:
42        sw $t0, 0($s0) # show key
43        nop
44  end_process:
45  next_pc:
46        mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
47        addi $at, $at, 4 # $at = $at + 4 (next instruction)
48        mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
49  return: eret # Return from exception
```

Thực hiện gõ chương trình vào công cụ MARS

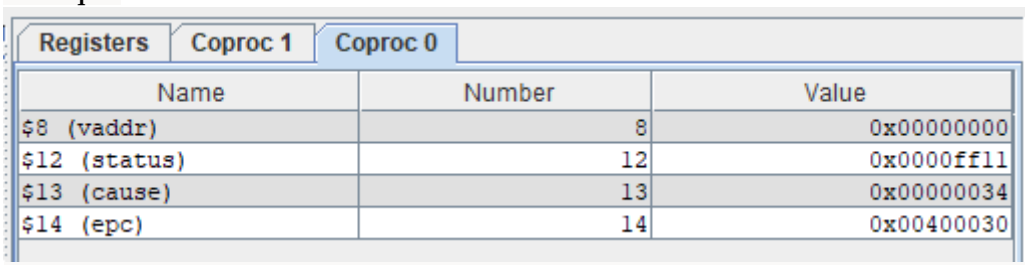
2. Giải thích

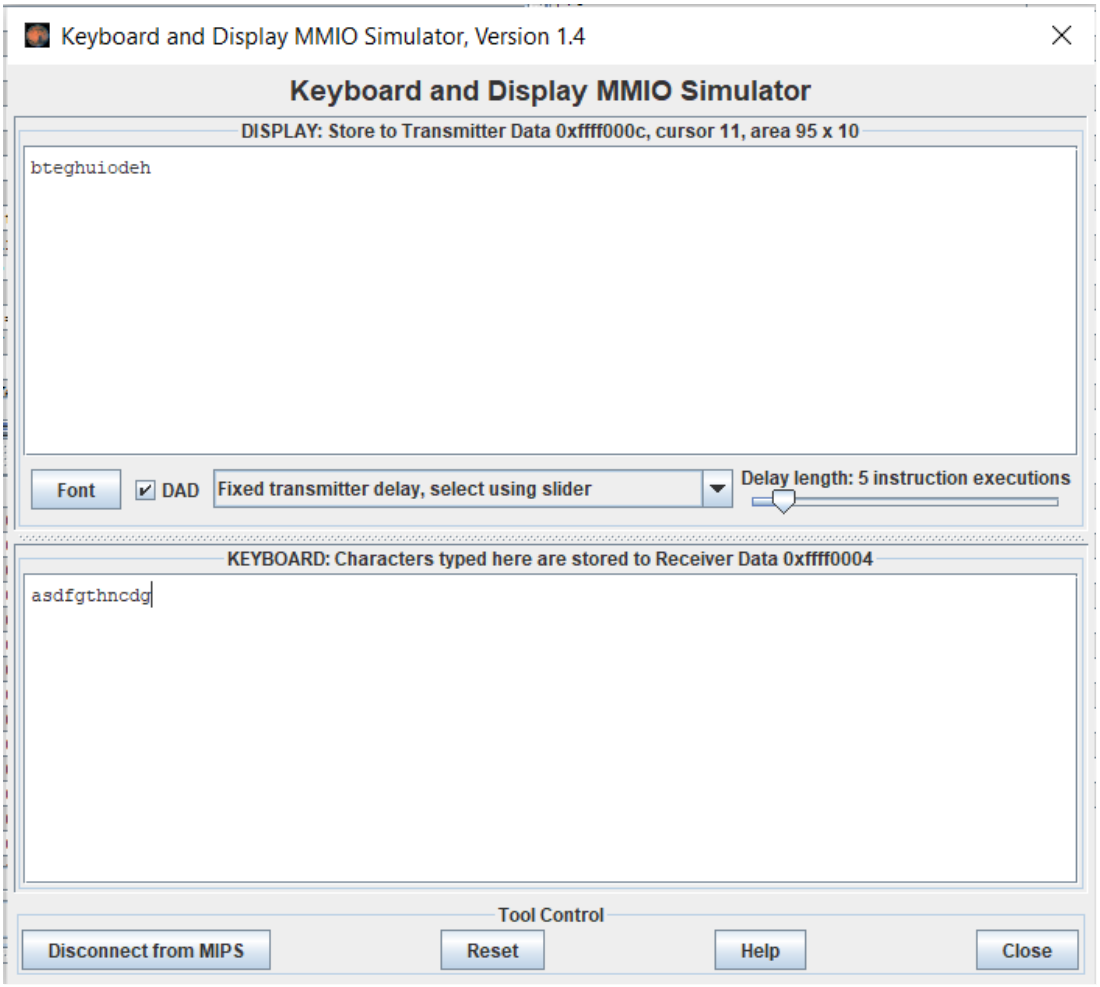
- Chương trình minh họa chương trình ngắt với bàn phím. Bàn phím không tự động ngắt mà phải dùng lệnh **teqi** để ngắt.
- Khởi tạo
  - KEY_CODE (0xFFFF0004) giá trị ASCII ký tự được nhập từ bàn phím → gán $k0
  - KEY_READY 0xFFFF0000 = 1 khi có ký tự được nhập và bằng 0 sau lệnh lw → gán $k1
  - DISPLAY_CODE (0xFFFF000C) giá trị ASCII của ký được được in ra màn hình → gán $s0
  - DISPLAY_READY (0xFFFF0008) = 1 khi chuẩn bị in và sẽ = 0 sau lệnh sw → gán $1
  - MASK_CAUSE_KEYBOARD: 0x00000034 BIT MASK: nguyên nhân ngắt tạo bởi keyboard.
- Hàm *WaitForKey:* vòng lặp đọc ký tự đầu vào
  - Nếu $t1 (KEY_READY) = 0 → chưa có ký tự nào được nhập → tiếp tục lặp đến khi $t1 = 1 (có ký tự từ bán phím được nhập)

- Sau khi xác nhận có ký tự được nhập, chương trình nhảy đến nhiệm vụ ngắt mềm bằng cách kiểm tra **teqi $1, 1** và nhảy đến địa chỉ ngắt *.ktext 0x80000180*.

- Cơ chế ngắt: khi có một ký tự được nhập từ bàn phím và KEY_READY = 1
  - Check nguyên nhân ngắt bằng cách kiểm tra thanh ghi $13 (được lưu vào $t1), và mặt nạ bit ngắt bàn phím (được lưu vào $t2). Sử dụng phép AND, nếu $1 trùng $t2 → ngắt do bàn phím
  - Đọc ký tự được ghi từ bán phím và kiểm tra DISPLAY_READY có sẵn sàng (=1) hay không (=0)
    - Nếu DISPLAY_READY = 1 → mã hóa ký tự (tăng mã ASCII lên 1) và sẵn sàng in ra màn hình
    - Nếu DISPLAY_READY = 0 → tiếp tục polling bằng vòng lặp
- Kết thúc, ta cập nhật giá trị $14 cộng thêm 4 (địa chỉ của câu lệnh tiếp theo), dùng lệnh **eret** để trở về chương trình chính

3. Kết quả




Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c, cursor 11, area 95 x 10
bteghuiodeh
Font
DAD
Fixed transmitter delay, select using slider
Delay length: 5 instruction executions
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
asdfgthncdg
Tool Control
Disconnect from MIPS
Reset
Help
Close