**REPORT WEEK 11**

Họ và tên: Nguyễn Văn Hưng

MSSV: 20225634

Lớp: Thực hành kiến trúc máy tính – Mã lớp: 147799

1. **Assignment 1:**
2. Đề bài: Create a new project, type in, and build the program of Home Assignment 1. Upgrade the source code so that it could detect all 16 key buttons, from 0 to F.
3. Code:

.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 $t1, IN\_ADRESS\_HEXA\_KEYBOARD

li $t2, OUT\_ADRESS\_HEXA\_KEYBOARD

li $t3, 0x01

li $t4, 0x02

li $t5, 0x04

li $t6, 0x08

li $t0, 0

polling:

beq $t0, 100, exit

sb $t3, 0($t1 ) # must reassign expected row

lb $a0, 0($t2) # read scan code of key button

bne $a0, $zero, print

sb $t4, 0($t1 ) # must reassign expected row

lb $a0, 0($t2) # read scan code of key button

bne $a0, $zero, print

sb $t5, 0($t1 ) # must reassign expected row

lb $a0, 0($t2) # read scan code of key button

bne $a0, $zero, print

sb $t6, 0($t1 ) # must reassign expected row

lb $a0, 0($t2) # read scan code of key button

bne $a0, $zero, print

j continue

print:

li $v0, 34 # print integer (hexa)

syscall

continue:

addi $t0, $t0, 1

sleep:

li $a0, 1000 # sleep 1000ms

li $v0, 32

syscall

back\_to\_polling:

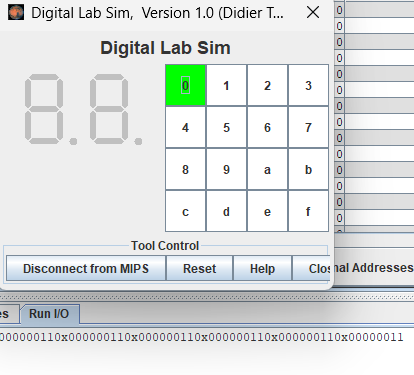
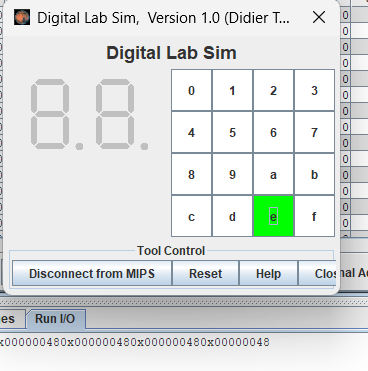
j polling # continue polling

exit:

1. Giải thích

Chương trình này liên tục kiểm tra trạng thái của các phím trên bàn phím ma trận. Khi phát hiện một phím đã được nhấn, chương trình sẽ in mã của phím đó dưới dạng số thập lục phân. Quá trình kiểm tra sẽ lặp lại đến khi giá trị đếm đạt 100 lần kiểm tra.

1. Kết quả

1. **Assignment 2:**
2. Đề bài: Create a new project, type in, and build the program of Home Assignment 2.
3. Code:

.eqv IN\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0012

.data

Message: .asciiz "ban da nhan mot nut \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

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

# Processing

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

IntSR:

addi $v0, $zero, 4 # show message

la $a0, Message

syscall

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

# Evaluate the return address of main routine

# epc <= epc + 4

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

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

1. Giải thích

Chương trình này kích hoạt ngắt từ bàn phím ma trận và liên tục chờ đợi ngắt trong vòng lặp vô hạn. Khi một phím được nhấn, ngắt sẽ được kích hoạt, và chương trình sẽ hiển thị thông báo rằng một phím đã được nhấn. Sau đó, nó cập nhật địa chỉ quay lại và tiếp tục chờ đợi trong vòng lặp vô hạn.

Trong chương trình, **$pc** thay đổi theo các bước sau:

1. **Thực thi tuần tự trong main:**
   * **$pc** tăng tuần tự qua các lệnh trong **main**.
2. **Vòng lặp vô hạn:**
   * Lệnh **b Loop** khiến **$pc** nhảy về nhãn **Loop**, tạo ra vòng lặp vô hạn.
3. **Khi xảy ra ngắt:**
   * **$pc** nhảy đến địa chỉ **0x80000180** để thực thi dịch vụ ngắt (**IntSR**).
4. **Trong dịch vụ ngắt (IntSR):**
   * **$pc** tiếp tục thực thi các lệnh trong **IntSR**.
5. **Trả về từ ngắt (eret):**
   * Lệnh **eret** khôi phục giá trị của **$pc** từ EPC và tiếp tục thực thi lệnh sau lệnh gây ra ngắt.
6. Kết quả

A screenshot of a computer

Description automatically generated

1. **Assignment 3:**
2. Đề bài: Create a new project, type in, and build the program of Home Assignment 3. Upgrade the source code so that it could detect all 16 key buttons, from 0 to F.
3. Code:

.eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv OUT\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014

.data

Message: .asciiz "Key scan code "

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# MAIN Procedure

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

.text

main:

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

# Bật các ngắt mong đợi

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

# Bật ngắt của bàn phím ma trận 4x4 trong Digital Lab Sim

li $t1, IN\_ADDRESS\_HEXA\_KEYBOARD

li $t3, 0x80 # bit 7 = 1 để bật ngắt

sb $t3, 0($t1)

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

# Vòng lặp và in số thứ tự

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

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 # in số thứ tự

syscall

prn\_eol:

addi $v0, $zero, 11

li $a0, '\n' # in ký tự xuống dòng

syscall

sleep:

addi $v0, $zero, 32

li $a0, 1000 # ngủ 300 ms

syscall

nop # CẢNH BÁO: nop là bắt buộc ở đây.

b Loop # Vòng lặp

end\_main:

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# ROUTINE PHỤC VỤ NGẮT CHUNG cho tất cả các ngắt

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

.ktext 0x80000180

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

# LƯU REG FILE hiện tại vào stack

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

IntSR:

addi $sp, $sp, -20

sw $at, 16($sp)

sw $v0, 12($sp)

sw $a0, 8($sp)

sw $t1, 4($sp)

sw $t3, 0($sp)

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

# Xử lý

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

prn\_msg:

addi $v0, $zero, 4

la $a0, Message

syscall

get\_cod:

# hang1

li $t1, IN\_ADDRESS\_HEXA\_KEYBOARD

li $t3, 0x81

sb $t3, 0($t1)

li $t1, OUT\_ADDRESS\_HEXA\_KEYBOARD

lb $a0, 0($t1)

bne $a0, $zero, prn\_cod

# hang2

li $t1, IN\_ADDRESS\_HEXA\_KEYBOARD

li $t3, 0x82

sb $t3, 0($t1)

li $t1, OUT\_ADDRESS\_HEXA\_KEYBOARD

lb $a0, 0($t1)

bne $a0, $zero, prn\_cod

# hang3

li $t1, IN\_ADDRESS\_HEXA\_KEYBOARD

li $t3, 0x84

sb $t3, 0($t1)

li $t1, OUT\_ADDRESS\_HEXA\_KEYBOARD

lb $a0, 0($t1)

bne $a0, $zero, prn\_cod

# hang4

li $t1, IN\_ADDRESS\_HEXA\_KEYBOARD

li $t3, 0x88 # kiểm tra hàng 4 và bật lại bit 7

sb $t3, 0($t1) # phải gán lại hàng mong đợi

li $t1, OUT\_ADDRESS\_HEXA\_KEYBOARD

lb $a0, 0($t1)

bne $a0, $zero, prn\_cod

prn\_cod:

li $v0, 34

syscall

li $v0, 11

li $a0, '\n' # in ký tự xuống dòng

syscall

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

# Đánh giá địa chỉ trả về của hàm chính

# epc <= epc + 4

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

next\_pc:

mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc

addi $at, $at, 4 # $at = $at + 4 (lệnh tiếp theo)

mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at

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

# PHỤC HỒI REG FILE từ STACK

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

restore:

lw $t3, 0($sp) # Phục hồi các thanh ghi từ stack

addi $sp, $sp, 4

lw $t1, 4($sp) # Phục hồi các thanh ghi từ stack

addi $sp, $sp, 4

lw $a0, 8($sp) # Phục hồi các thanh ghi từ stack

addi $sp, $sp, 4

lw $v0, 12($sp) # Phục hồi các thanh ghi từ stack

addi $sp, $sp, 4

lw $at, 16($sp) # Phục hồi các thanh ghi từ stack

addi $sp, $sp, 4

return:

eret # Trở về từ ngắt

1. Giải thích

* **Lưu lại các thanh ghi vào stack**:
  + Lưu các thanh ghi **$at**, **$v0**, **$a0**, **$t1**, **$t3** vào stack để bảo toàn giá trị hiện tại trước khi xử lý ngắt.
* **Xử lý ngắt**:
  + In ra thông báo "Key scan code".
  + Đọc mã phím từ bàn phím bằng cách kiểm tra từng hàng của ma trận bàn phím 4x4. Nếu có mã phím nào được phát hiện (**bne $a0, $zero, prn\_cod**), thì in mã phím đó ra.
* **Tính toán địa chỉ trở lại**:
  + Tính toán địa chỉ trở lại của lệnh tiếp theo sau khi xử lý ngắt (**epc + 4**).
* **Khôi phục lại các thanh ghi từ stack**:
  + Khôi phục lại giá trị các thanh ghi đã được lưu trước khi ngắt từ stack.
* **Trở lại từ ngoại lệ**:
  + Lệnh **eret** để trở lại từ chế độ ngoại lệ, kết thúc quá trình xử lý ngắt và quay

1. Kết quả

A screenshot of a computer code

Description automatically generated

1. **Assignment 4:**
2. Đề bài: Create a new project, type in, and build the program of Home Assignment 4.
3. Code:

.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

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

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

1. Giải thích

Chương trình này sử dụng ngắt để xử lý việc nhấn phím và thời gian. Khi có ngắt xảy ra, trình phục vụ ngắt (ISR) sẽ kiểm tra nguyên nhân ngắt và thực hiện các hành động tương ứng như in ra thông báo. Sau đó, chương trình tiếp tục lặp lại quá trình này.

1. Kết quả

A screenshot of a computer

Description automatically generated

1. **Assignment 5:**
2. Đề bài: Create a new project, type in, and build the program of Home Assignment 5.
3. Code:

.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

1. Giải thích

Đoạn mã này sử dụng ngắt để xử lý dữ liệu từ bàn phím và hiển thị nó trên màn hình. Dưới đây là cách hoạt động của chương trình:

* Chương trình sử dụng hai thanh ghi **$k0** và **$k1** để đọc mã ASCII từ bàn phím và kiểm tra xem có dữ liệu mới từ bàn phím hay không.
* Sử dụng hai thanh ghi **$s0** và **$s1** để hiển thị mã ASCII trên màn hình và kiểm tra xem màn hình đã sẵn sàng để hiển thị hay chưa.
* Trong vòng lặp, chương trình chờ đợi dữ liệu mới từ bàn phím. Nếu có dữ liệu mới, nó sẽ tạo ra một ngắt.
* Trong trình phục vụ ngắt, chương trình đọc mã ASCII từ bàn phím và chờ đợi cho đến khi màn hình sẵn sàng để hiển thị.
* Sau khi màn hình sẵn sàng, chương trình thực hiện các thao tác cần thiết để
* thay đổi dữ liệu và hiển thị nó lên màn hình.

1. Kết quả

