Laboratory Exercise 11

|  |  |
| --- | --- |
| Tên | Đặng Hồng Minh |
| MSSV | 20225740 |

Assignment 1

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

.eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv OUT\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014

.data

prev\_key: .byte 0x00

.text

main:

li $t1, IN\_ADDRESS\_HEXA\_KEYBOARD

li $t2, OUT\_ADDRESS\_HEXA\_KEYBOARD

la $t4, prev\_key

li $t3, 0x1

polling:

sb $t3, 0($t1)

lb $a0, 0($t2)

lb $t5, 0($t4)

beq $a0, $t5, next\_row

beqz $a0, next\_row

sb $a0, 0($t4)

li $v0, 34

syscall

li $a0, 100

li $v0, 32

syscall

next\_row:

sll $t3, $t3, 1

li $t6, 0x10

beq $t3, $t6, reset\_rows

j polling

reset\_rows:

li $t3, 0x1

j polling

exit:

li $v0, 10

syscall

Assignment 2

A screenshot of a computer

Description automatically generated

.eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012

.data

Message: .asciiz "Oh my god. Someone's presed a button.\n"

.text

main:

# Enable interrupts you expect

# Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim

li $t1, IN\_ADDRESS\_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

Assignment 3

A screenshot of a computer

Description automatically generated

.eqv IN\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv OUT\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0014

.data

Message: .asciiz "Key scan code "

.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)

addi $sp, $sp, 4 # Save $at because we may change it later

sw $at, 0($sp)

addi $sp, $sp, 4 # Save $sp 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

li $t6, 0x1

li $t3, 0x81 # check row 4 and re-enable bit 7

get\_cod:li $t1, IN\_ADRESS\_HEXA\_KEYBOARD

bgt $t3, 0x88, reset\_getcod # check row 4 and re-enable bit 7

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

li $t1, OUT\_ADRESS\_HEXA\_KEYBOARD

lb $a0, 0($t1)

bnez $a0, prn\_cod

mul $t6, $t6, 2

add $t3, $t6, 0x80

j get\_cod

prn\_cod:li $v0,34

syscall

li $v0,11

li $a0,'\n' # print endofline

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

# RESTORE the REG FILE from STACK

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

lw $ra, 0($sp) # Restore the registers from stack

addi $sp, $sp, -4

return: eret # Return from exception

reset\_getcod:

li $t3, 0x81

li $t6, 0x1

j get\_cod