

Laboratory 11

Họ tên: Vũ Quốc Bảo
MSSV: 20225694

Assignment 1:

```
#-----
#          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_ADDRESS_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_ADDRESS_HEXa_KEYBOARD 0xFFFF0014

.text
main:
    li $t1, IN_ADDRESS_HEXa_KEYBOARD
    li $t2, OUT_ADDRESS_HEXa_KEYBOARD

polling:
    li $t3, 0x1 # check row 4 with key 0, 1, 2, 3
    sb $t3, 0($t1) # Must reassign expected row
    lb $a0, 0($t2) # Read scan code of key button
    bne $a0, 0x0, print
```

```
li $t3, 0x2 # check row 4 with key 4, 5, 6, 7
sb $t3, 0($t1) # Must reassign expected row
lb $a0, 0($t2) # Read scan code of key button
bne $a0, 0x0, print
```

```
li $t3, 0x4 # check row 4 with key 8, 9, A, B
sb $t3, 0($t1) # Must reassign expected row
lb $a0, 0($t2) # Read scan code of key button
bne $a0, 0x0, print
```

```
li $t3, 0x8 # check row 4 with key C, D, E, F
sb $t3, 0($t1) # Must reassign expected row
lb $a0, 0($t2) # Read scan code of key button
bne $a0, 0x0, print
```

```
j polling # Continue polling
```

print:

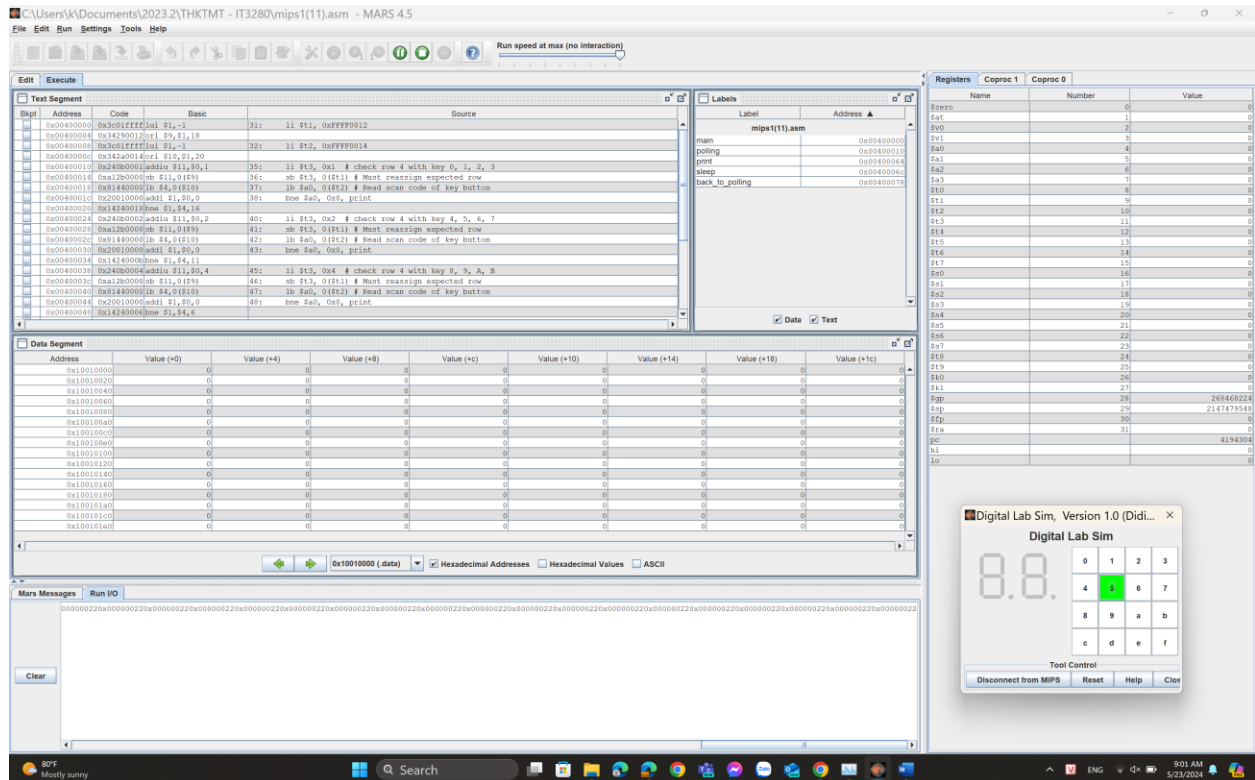
```
li $v0, 34 # Print integer (hexadecimal)
syscall
```

sleep:

```
li $a0, 100 # Sleep for 100ms
li $v0, 32
syscall
```

back_to_polling:

```
j polling # Continue polling
```



Assignment 2:

.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
#-----
# 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

```

The screenshot displays the MARS 4.5 MIPS simulator interface. The main window shows the assembly code for the provided program, with the following instructions visible:

```

0x00400000: 0x3c01ffff lui $t1, -1
0x00400004: 0x3420012c ori $t1, $t1, 18
0x00400008: 0x40000000 addiu $t1, $t1, 0x128
0x0040000c: 0xa12b0000 sb $t1, 0($t1)
0x00400010: 0x00000000 nop
0x00400014: 0x00000000 nop
0x00400018: 0x00000000 nop
0x0040001c: 0x00000000 nop
0x00400020: 0x0401ffff bneq $t0, -5
0x00400024: 0x20200004 addi $t2, $t0, 4
0x00400028: 0x3c011000 lui $t2, 4097
0x0040002c: 0x34280000 ori $t2, $t2, 0
0x00400030: 0x00000000 syscall
0x00400034: 0x00170000 mfc0 $t4, $14
0x00400038: 0x20200004 addi $t1, $t4, 4
0x0040003c: 0x40817000 mtc0 $t1, $14
0x00400040: 0x40000000 eret

```

The right panel shows the Registers window with the following values:

Register	Name	Number	Value
\$0	(zero)	0	0
\$12	(status)	12	65292
\$13	(cause)	13	0
\$14	(epc)	14	0

The bottom panel shows the Data Segment window with a table of memory addresses and values:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	330880399	1669029497	1394617956	166918127	1931961710	170399824	543450483	196345889
0x10010020	1852798068	2604	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0
0x100101c0	0	0	0	0	0	0	0	0
0x100101e0	0	0	0	0	0	0	0	0

The bottom right panel shows the Digital Lab Sim window, which displays a digital display showing the value 8.8 and a keypad with buttons 0-9, a, b, c, d, e, f, and a decimal point.

Assignment 3:

```
.eqv IN_ADDRESS_HEX_A_KEYBOARD 0xFFFF0012
.eqv OUT_ADDRESS_HEX_A_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_ADDRESS_HEX_A_KEYBOARD
    li $t3, 0x80 # bit 7 = 1 to enable
    sb $t3, 0($t1)
    #-----
    # Loop and 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

get_cod:
    li $t2, IN_ADDRESS_HEXKEYBOARD
    li $t3, 0x81          # check row 4 and re-enable bit 7
    sb $t3, 0($t2)        # must reassign expected row
    li $t1, OUT_ADDRESS_HEXKEYBOARD
    lb $a0, 0($t1)
    bne $a0, $0, prn_cod
    li $t3, 0x82          # check row 4 and re-enable bit 7
    sb $t3, 0($t2)        # must reassign expected row
    lb $a0, 0($t1)
    bne $a0, $0, prn_cod
    li $t3, 0x84          # check row 4 and re-enable bit 7
    sb $t3, 0($t2)        # must reassign expected row
    lb $a0, 0($t1)
    bne $a0, $0, prn_cod
    li $t3, 0x88          # check row 4 and re-enable bit 7
    sb $t3, 0($t2)        # must reassign expected row
    lb $a0, 0($t1)

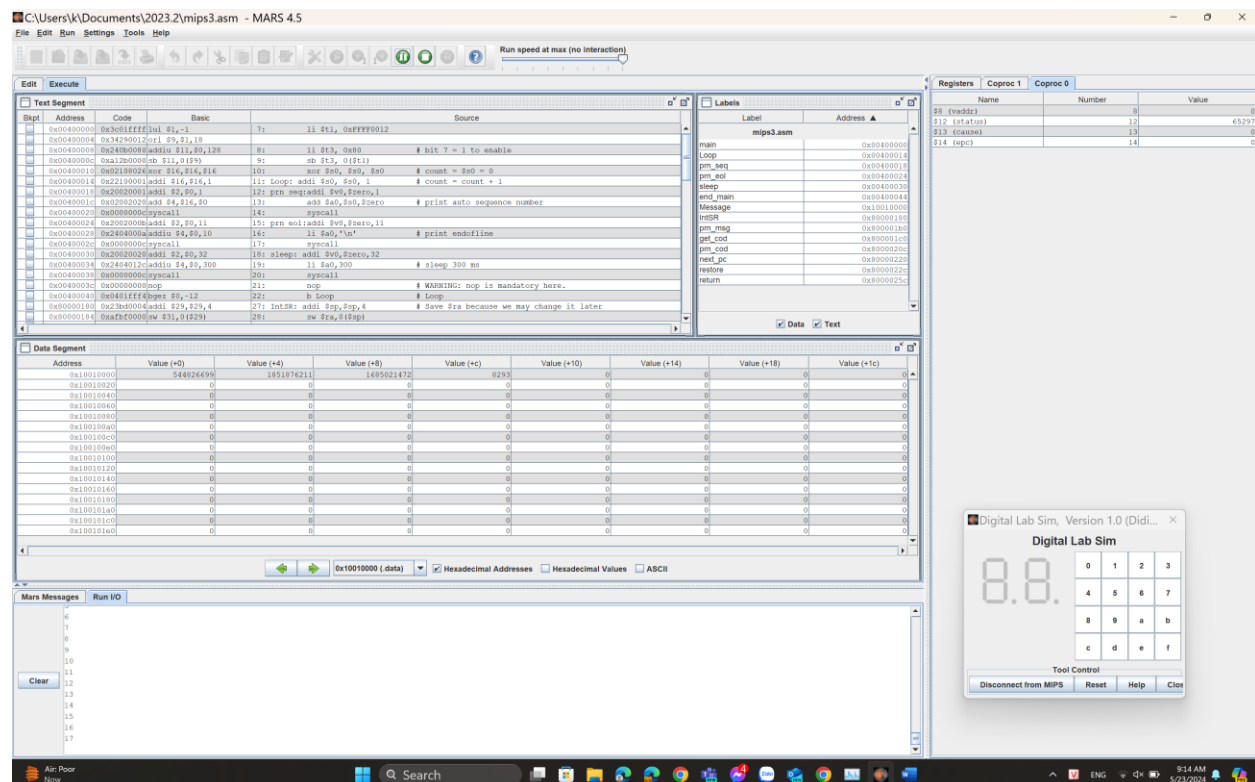
prn_cod:li $v0,34
    syscall
    li $v0,11
    li $a0,'\n' # print endofline
    syscall
    #-----
    # Evaluate the return address of main routine
    # epc <= epc + 4
    #-----

```

```

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

```



Assignment 4:

```
.eqv IN_ADRESS_HEX_A_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 interval!\n"
#~~~~~
# MAIN Procedure
#~~~~~
.text
main:
#-----
# Enable interrupts you expect
#-----
# Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim
li $t1, IN_ADRESS_HEX_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 and 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

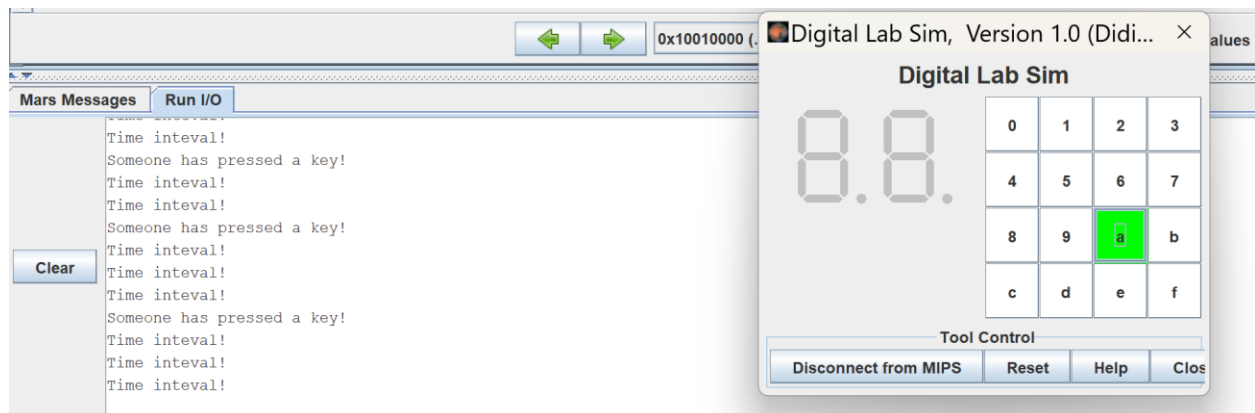
```



```

#-----
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

```



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

```

C:\Users\K\Documents\2023.2\vnips5.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Text Segment

Offset	Address	Code	Basic	Source
0x00400000	0x3c0fffff	lui \$t1, -1	9: li \$t0, 0xfffff004	
0x00400004	0x343a004c	rli \$t0, \$t1, 4		
0x00400008	0x3c0fffff	lui \$t1, -1	10: li \$t1, 0xfffff000	
0x0040000c	0x343b0000	rli \$t2, \$t1, 0	12: li \$s0, 0xfffff00c	
0x00400010	0x3c0fffff	lui \$t1, -1		
0x00400014	0x3430000c	rli \$t6, \$t1, 12	13: li \$s1, 0xfffff008	
0x00400018	0x3c0fffff	lui \$t1, -1		
0x0040001c	0x34310000	rli \$t1, \$t1, 0	14: loop: nop	
0x00400020	0x00000000		15: WaitForKey: lw \$t1, 0(\$t1) #	
0x00400024	0x3f600000	lw \$t0, 0(\$t1)	16: lmg \$t1, \$zero, WaitForKey: #	
0x00400028	0x3128ffff	hmg \$t0, \$t0, -2	17: MakeInt0: lmg \$t1, 3 # if \$t1	
0x0040002c	0x052c0001	lmg \$t1, 3	18: j loop	
0x00400030	0x00100001	0x00400020		
0x00400034	0x00000181	0x000000c0	\$t0, \$t1	21: get_caus: srl \$t1, \$t1 # \$t1
0x00400038	0x240a0014	asrui \$t0, \$t0, 52	24: faCount: li \$t2, 0x000034af	
0x0040003c	0x00000181	0x012a092d	and \$t0, \$t0, \$t2	25: and \$t1, \$t2
0x00400040	0x01000001	hmg \$t1, \$t1, 1	26: lmg \$t0, \$t0, Counter Keyboard	
0x00400044	0x00000000	0x000001ac		27: j end process
0x00400048	0x0f480000	lw \$s0, 0(\$t0)	29: ReadKey: lw \$s0, 0(\$s0) # \$s0	

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0	0	0
0x10010020	0	0	0
0x10010040	0	0	0
0x10010060	0	0	0
0x10010080	0	0	0
0x100100a0	0	0	0
0x100100c0	0	0	0
0x100100e0	0	0	0
0x10010100	0	0	0
0x10010120	0	0	0
0x10010140	0	0	0
0x10010160	0	0	0
0x10010180	0	0	0
0x100101a0	0	0	0
0x100101c0	0	0	0
0x100101e0	0	0	0

Registers

Name	Number	Value
\$0 (\$zero)	0	0
\$12 (\$t4)	12	61297
\$13 (\$t5)	13	0
\$14 (\$t6)	14	0

Labels

Label	Address
mips5.asm	
loop	0x00400020
WaitForKey	0x00400024
MakeInt0	0x0040002c
get_caus	0x00400034

Keyboard and Display MMIO Simulator, Version 1.4

Keyboard and Display MMIO Simulator

DISPLAY: Store to Transmitter Data 0xfffff00c, cursor 22, area 97 x 10

WV1hpdlCtpt1,131367:5

Font ☒ DAD Fixed transmitter delay, select using slider Delay length: 5 instruction executions

KEYBOARD: Characters typed here are stored to Receiver Data 0xfffff004

Vu Quoc Bao - 20225494

Tool Control

Disconnect from MIPS Reset Help Close

Mars Messages Run I/O

-- program is finished running --

Clear

-- program is finished running --

-- program is finished running --