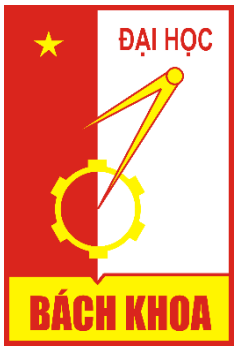
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY



FINAL PROJECT REPORT

COMPUTER ARCHITECTURE LAB

Supervisor: Le Ba Vui

Group 17

Tran Viet Anh 20226012

Vo Chien Huy 20215208

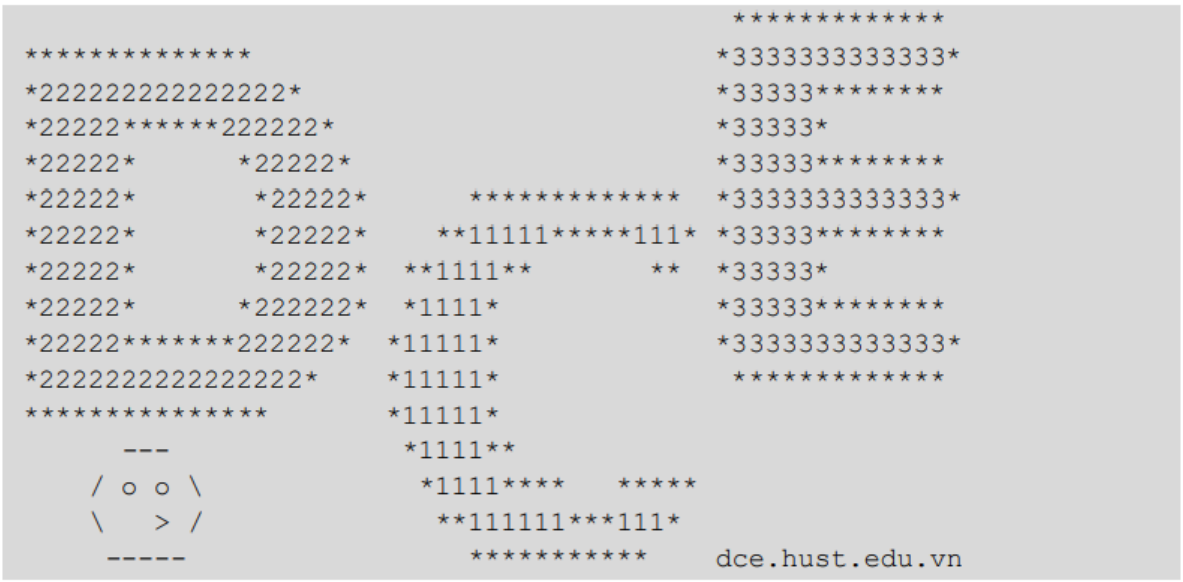
Hanoi, 2024

Contents

Project 9 : Drawing shape ussing ASCII characters

1. Problem

Given a picture translated to ASCII characters as follows, this is the shapes of DCE with border * and colors are digits.



- Show this picture in the console window.
- Change the picture so that DCE has only a border without color inside.
- Change the order of DCE to ECD.
- Enter the new color number from the keyboard, update the picture with new colors.

Note: Except the memory used to store the picture in source code, do not use any extra memory space.

2. Explain this code

Set up a board size 64x16: each word E,C,D has the width of 21 characters; the 64th character is the 'new line' character.

Data segment:

+ The string to print the board.
+ A Menu.

+ An error message to be printed if the choice is not valid.

Text segment:

Main

- Print Menu using ‘syscall function’ with $v0 = 4.
- Input choice using ‘syscall function’ with $v0 = 5.
- If choice = 1,2,3,4 then do task 1,2,3,4; if choice = 5 then exit.
- If choice is not 1-5, then print error message and jump back to main.

Task 1:

Print the board by printing ‘string’ using ‘syscall function’ with $v0 = 4.

Task 2:

The idea is to go through each character (1024 characters), if it is a number then convert it into ‘space’ and print, else just print it.

- Set $s0 to be the address of ‘string’, $t0 to be the order of character.
- Set up a loop (loop_2):
  - Check if the order exceeds 1024, then all characters are printed and jump to main.
  - Load the first byte of $s0 into $a0.
- In ASCII, number if from 48 – 57, so check if $a0 is not a number then just print, else convert $a0 into 32 (the ‘space’ character).

Task 3:

The idea is to print characters 42 – 63 (letter E), then characters 21 – 21 (letter C) then characters 0 – 20 (letter D).

- Set $s0 to be the order of line.
- In each loop, check if $s0 reaches 16, then end task and jump back to main.
- Set $s1 to be the first character of each line by set $s1 = 16 x (the order of line) + (address of the first character of board).
- Set $s2 to be the first character of each letter in each line by set $s2 = 0 + $s1 (letter D), $s2 = 21 + $s1 (letter C), $s2 = 42 + $s1 (letter E).
- Print letters E, C, D sequentially by using ‘jal’ (use ‘jal’ so that it can return and print the next letter).
- Print ‘new line’ character.
- Advance to next line by add 1 to $s0.
- Inside ‘print_21_character’ function:
  - Set $t0 to be the order of character.
  - If $t0 exceeds 21, then jump back and print the characters of the next letter in that line.
  - Print characters by using ‘syscall function’ with $v0 = 11.
  - Advance to next byte by adding 1 to $s2, advance the order of characters by adding 1 to $t0.

Task 4:
The idea is to build an input function to choose color for each letter, the rest is almost similar to task 3.

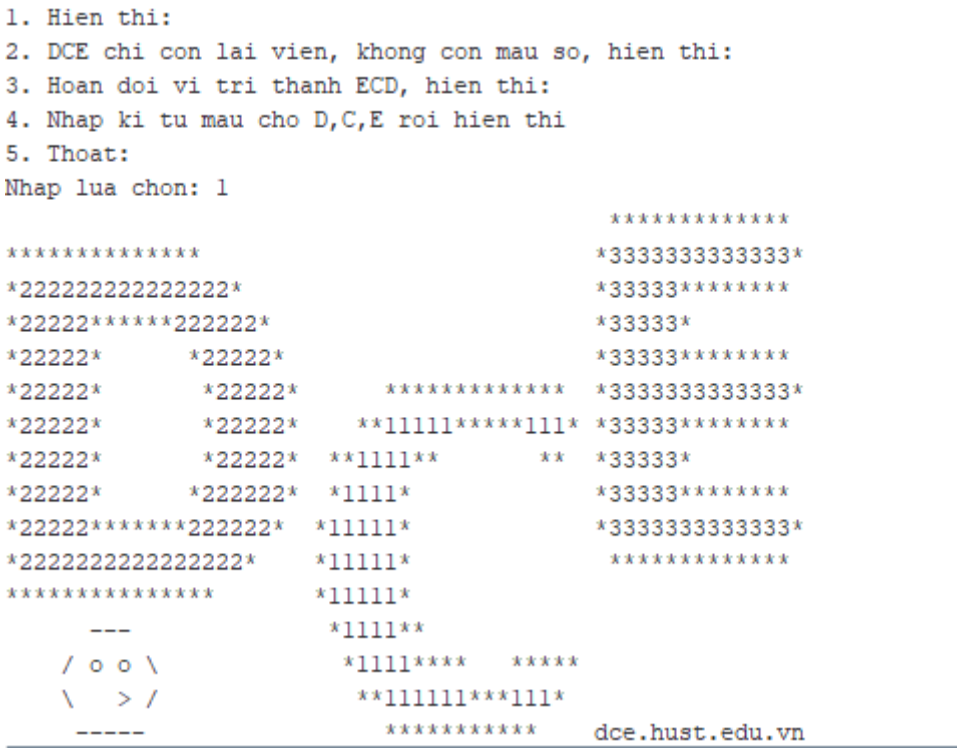
- In ‘input’ function:
  - Print input message.
  - Using ‘syscall’ function to input color of each letter.
- ‘output’ function is like a combination of task 2 and task 3: the main idea is similar to task 3; except that if the character if a number, then convert it into the number which is inputted in ‘input’ function.

Task 5 (exit):

- Exit just by jumping to the end of the program.

1. Result

- Task 1



- Task 2

```
1. Hien thi:
2. DCE chi con lai vien, khong con mau so, hien thi:
3. Hoan doi vi tri thanh ECD, hien thi:
4. Nhap ki tu mau cho D,C,E roi hien thi
5. Thoat:
Nhap lua chon: 2
                                          *************
***************                          *             *
*                *                       *      ********
*      ******       *                     *      *
*      *       *      *                   *      ********
*      *        *      *       *************  *             *
*      *        *      *     **      *****   * *      ********
*      *        *      *   **     **        **  *      *
*      *       *       *   *     *             *      ********
*      *******       *   *      *              *             *
*                  *     *      *               *************
****************        *      *
       ---               *     **
     / o o \              *     ****   *****
     \   > /               **       ***   *
      -----                  ***********     dce.hust.edu.vn
```

- Task 3

```
1. Hien thi:
2. DCE chi con lai vien, khong con mau so, hien thi:
3. Hoan doi vi tri thanh ECD, hien thi:
4. Nhap ki tu mau cho D,C,E roi hien thi
5. Thoat:
Nhap lua chon: 3
 *************
*3333333333333*                          **************
*33333********                           *222222222222222*
*33333*                                  *22222******222222*
*33333********                           *22222*      *22222*
*3333333333333*            *************  *22222*       *22222*
*33333********           **llllll*****lll* *22222*       *22222*
*33333*                 **llll**       **  *22222*       *22222*
*33333********          *llll*             *22222*      *222222*
*3333333333333*        *lllll*             *22222*******222222*
 *************         *lllll*             *2222222222222222*
                        *lllll*             ****************
                         *llll**                  ---
                          *llll****   *****     / o o \
                           **llllll***lll*      \   > /
dce.hust.edu.vn              ***********         -----
```

- Task 4

```
Nhap lua chon: 4
Nhap lai mau so cua D,C,E:
1
2
3
                                                 ************
**************                                  *3333333333333*
*11111111111111*                                *33333********
*111111******111111*                            *33333*
*111111*      *111111*                          *33333********
*111111*       *111111*      ************       *3333333333333*
*111111*       *111111*    **22222*****222*     *33333********
*111111*       *111111*  **2222**       **      *33333*
*111111*      *111111*  *2222*                   *33333********
*111111*******111111*  *22222*                  *3333333333333*
*11111111111111111*    *22222*                   ************
***************       *22222*
      ---               *2222**
    / o o \              *2222****   *****
    \   > /               **222222***222*
     -----                  **********     dce.hust.edu.vn
```

- Task 5

```
MENU:
1. Hien thi:
2. DCE chi con lai vien, khong con mau so, hien thi:
3. Hoan doi vi tri thanh ECD, hien thi:
4. Nhap ki tu mau cho D,C,E roi hien thi
5. Thoat:
Nhap lua chon: 5

-- program is finished running (dropped off bottom) --
```

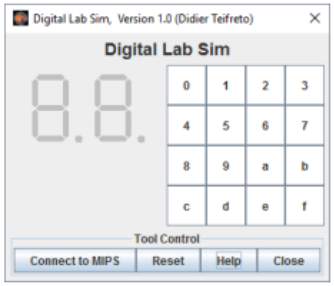
Project 3 : Typing Test

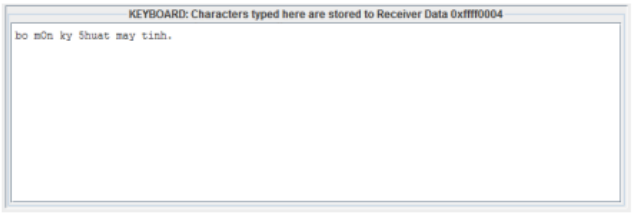
1. Problem

3. Typing test

Create a program to measure typing speed and display results using 2 7-segment LEDs with the following requirements:

- Given a sample text, hardcoded in the source code.
- Use the timer to create measurement intervals. This is the time between two consecutive interruptions.
- The user enters text from the keyboard, the program will count the number of correct characters and display it with LEDs.
- The program also needs to estimate typing speed as the number of words per unit of time.





2. Explain this code

- Constans and data section

```
.eqv SEVENSEG_LEFT 0xFFFF0011                  #Address of the left 7-segment LED
.eqv SEVENSEG_RIGHT 0xFFFF0010                 #Address of the right 7-segment LED
.eqv IN_ADDRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv MASK_CAUSE_COUNTER 0x00000400             #Bit 10: Counter interrupt
.eqv COUNTER 0xFFFF0013                        #Time Counter
.eqv KEY_CODE   0xFFFF0004                     #ASCII code from the keyboard, 1 byte
.eqv KEY_READY  0xFFFF0000                     #Non-zero if there is a new keycode
.data
number_array: .byte     63, 6,  91, 79, 102, 109 ,125, 7, 127, 111       #from 0 to 9
string: .asciiz "Bo mon ki thuat may tinh"
message1: .asciiz "Elapsed time: "
message2: .asciiz "(s) \nAverage typing speed: "
message3: .asciiz " words/minute\n"
Continue: .asciiz "Continue entering?"
```

In this section, constants are defined using .eqv statements, and memory space is reserved for initialized data, such as arrays (number_array), a string (string), and messages to be displayed later in the program.

- Main procedure

```
.text                                          #global variables: k0, k1, s0, s1, s2, s3, s4, s5, a1
MAIN:   li      $k0,  KEY_CODE
        li      $k1,  KEY_READY
        li      $t1, COUNTER                   #Initialize the timer
        sb      $t1, 0($t1)
        addi    $s0, $0, 0                     #Count the number of characters in 1 second
        addi    $s1, $0, 0                     #Count the total number of correct characters
        addi    $s2, $0, 0                     #Count the total number of entered words
        addi    $s3, $0, 0                     #Count the number of counter_intr occurrences
        addi    $s4, $0, 0                     #Store the previous character
        addi    $s5, $0, 0                     #Count the time (seconds)
        la      $a1, string
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#ENDLESS LOOP TO WAIT FOR INTERRUPTS
loop:
        lw      $t1, 0($k1)                    #$t1 = [$k1] = KEY_READY
        bne     $t1, $zero, make_Keyboard_Intr  #Generate an interrupt when a key is pressed on the keyboard
        addi    $v0, $0, 32
        li      $a0, 5
        syscall
        b       loop                           #The number of instructions in one loop is 6 => after 5 iterations, generate a counter interrupt
        nop
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
make_Keyboard_Intr:
        teqi    $t1, 1
        b       loop                           #Return to the loop to wait for the next interrupt event
        nop

end_Main:
```

This is the main procedure of the program. It initializes registers and variables, then enters an infinite loop waiting for interrupts. It checks for keyboard interrupts (make_Keyboard_Intr) and generates counter interrupts after a certain number of iterations.

- Interrupt service routine

```
.ktext 0x80000180

dis_int:li      $t1, COUNTER                   #BUG: must disable with Time Counter
        sb      $zero, 0($t1)
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#GET THE VALUE OF THE CP0.CAUSE REGISTER TO CHECK THE TYPE OF INTERRUPT
get_Caus:mfc0   $t1, $13                       #$t1 = Coproc0.cause
isCount:li      $t2, MASK_CAUSE_COUNTER        #if Cause value confirms Counter..
         and    $at, $t1,$t2
         bne    $at,$t2, keyboard_Intr
```

This section defines the interrupt service routine. It disables the timer (COUNTER) and checks the type of interrupt by examining the CP0.CAUSE register.

- Counter interrupt handling

```
counter_Intr:
        blt     $s3, 40, continue               #If the interrupt count is less than 40: 1 second has passed -> reset
        jal     hien_thi
        addi    $s3, $0, 0                      #Reset $s3
        addi    $s5, $s5, 1                     #Increase the time counter (seconds)
        j       en_int
        nop
continue:
        addi    $s3, $s3, 1                     #If less than 1 second, increase the interrupt count
        j       en_int
        nop
```

If the interrupt is a counter interrupt, it checks the interrupt count ($s3). If it's less than 40, it proceeds to continue (1 second has passed), otherwise, it displays typing speed using hien_thi.

- Keyboard interrupt handling

```
keyboard_Intr:
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#KEYBOARD INTERRUPT
test_char:                                      #Check the entered character
        lb      $t0, 0($a1)                     #Get the i-th character from the given string
        lb      $t1, 0($k0)                     #Get the entered character from the keyboard
        beq     $t1, $0, en_int                 #Error
        beq     $t1, '\n', end_Program          #If the character is '\n', check and print
        bne     $t0, $t1, kiem_tra_dau_cach     #If the entered character and the i-th character in the given
        nop
        addi    $s1, $s1, 1                     #Increase the count of correct characters
test_space:                                     #Check if the entered character is ' '
        bne     $t1, ' ', end_Process           #If the entered character == ' ' && the previous character !=
        nop
        beq     $s4, ' ', end_Process
        nop
        addi    $s2, $s2, 1                     #Increase the count of entered words
end_Process:
        addi    $s0, $s0, 1                     #Increase the number of characters in 1 second
        addi    $s4, $t1, 0                     #Update the previous character
        addi    $a1, $a1, 1                     #Increase the pointer by 1 <=> string+i
        j en_int
```

In the keyboard interrupt section, it checks the entered character, counts correct characters ($s1), and checks for spaces to count entered words ($s2).

- Display function

```
display:
        addi    $sp, $sp, -4
        sw      $ra, ($sp)
        addi    $t0, $0, 10
        div     $s0, $t0
        mflo    $v1                             #Get the tens place
        mfhi    $v0                             #Get the units place
        la      $a0, mang_so
        add     $a0, $a0, $v1
        lb      $a0, 0($a0)                     #Set value for segments
        jal     SHOW_7SEG_LEFT                  #Display

        la      $a0, mang_so
        add     $a0, $a0, $v0
        lb      $a0, 0($a0)                     #Set value for segments
        jal     SHOW_7SEG_RIGHT                 #Display

        addi    $s0, $0, 0                      #After displaying on the screen, reset the counter
        lw      $ra, ($sp)
        addi    $sp, $sp, 4
        jr      $ra
SHOW_7SEG_LEFT:
        li      $t0, SEVENSEG_LEFT              #Assign port's address
        sb      $a0, 0($t0)                     #Assign a new value
        jr      $ra
SHOW_7SEG_RIGHT:
        li      $t0, SEVENSEG_RIGHT             #Assign port's address
        sb      $a0, 0($t0)                     #Assign a new value
        jr      $ra
        nop
```

The **display** function converts the character count to 7-segment display values and calls

functions to display them.

- End Program

```
end_Program:
        addi    $v0, $0, 4
        la      $a0, message1
        syscall
        addi    $v0, $0, 1
        addi    $a0, $s5, 0
        syscall
        addi    $v0, $0, 4
        la      $a0, message2
        syscall
        addi    $v0, $0, 1
        addi    $a0, $0, 60
        mult    $s2, $a0
        mflo    $s2
        div     $s2, $s5
        mflo    $a0
        syscall
        addi    $v0, $0, 4
        la      $a0, message3
        syscall
        addi    $s0, $s1, 0
        jal     hien_thi
CONTINUE:
       li $v0, 50
       la $a0, Continue
       syscall
       beq $a0, 0, MAIN
       li $v0, 10
       syscall
```

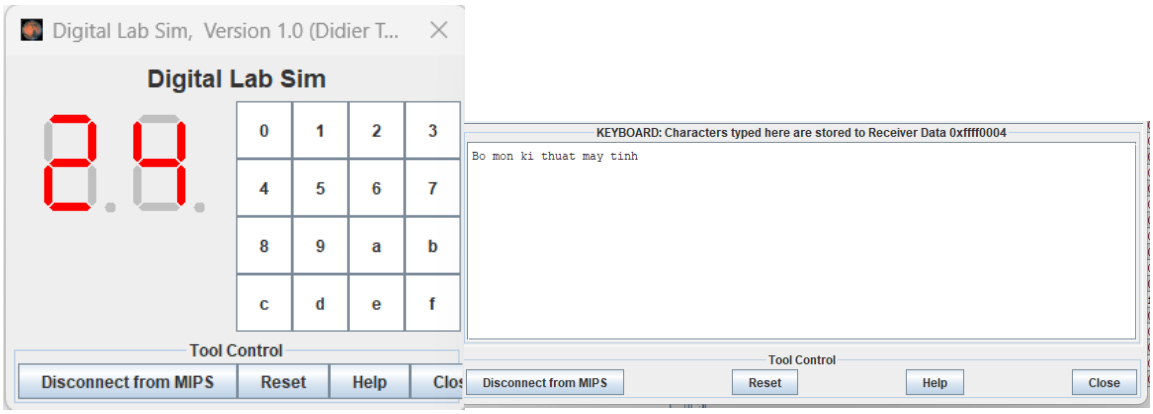
At the end of the program, it displays the elapsed time and average typing speed, then asks if the user wants to continue.

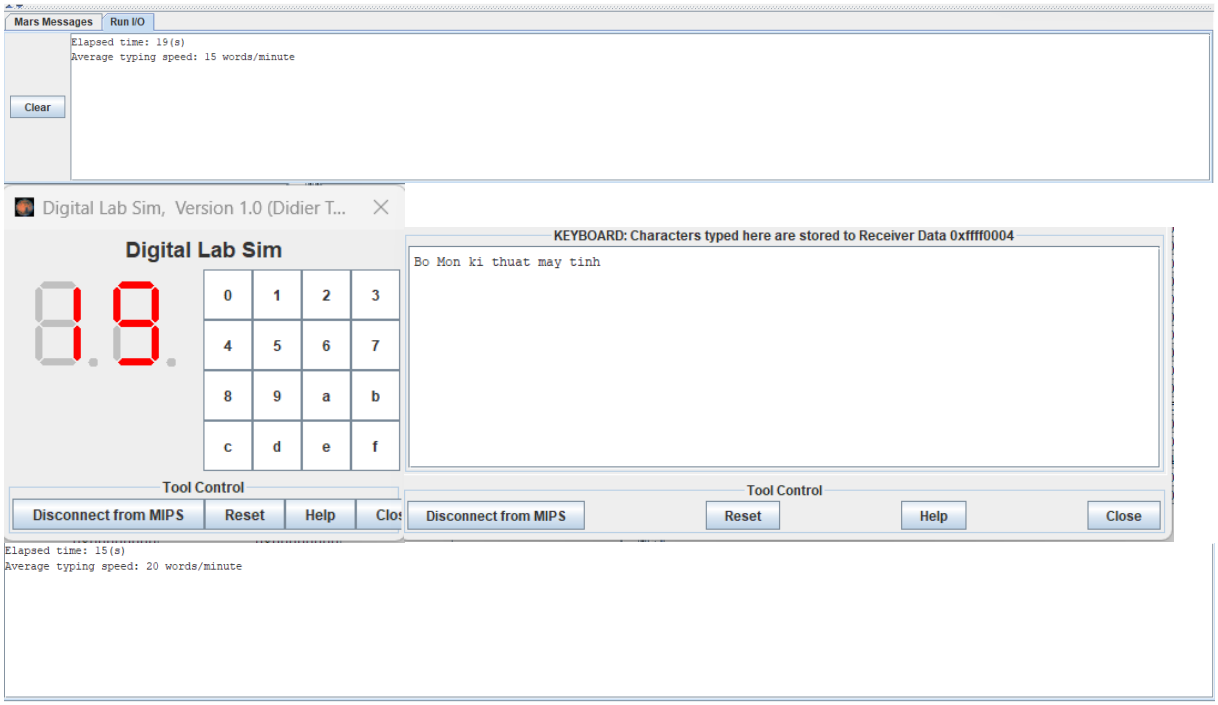
- End of interrupt processing

```
en_int:
       li      $t1, COUNTER
       sb      $t1, 0($t1)
       mtc0    $zero, $13                  #Must clear the cause register
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 the exception
```

This section handles the end of the interrupt, clearing the cause register and returning from the exception.

3. Result




Mars Messages
Run I/O
Elapsed time: 19(s)
Average typing speed: 15 words/minute
Clear
Digital Lab Sim, Version 1.0 (Didier T...
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
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
Bo Mon ki thuat may tinh
Tool Control
Disconnect from MIPS
Reset
Help
Close
Elapsed time: 15(s)
Average typing speed: 20 words/minute