**Lab Report  
Lab 10: Peripheral devices**

|  |  |
| --- | --- |
| Student ID | 20235890 |
| Student Name | Nguyễn Đức Anh |

**Assignment 1:**

**Code:**

# Lab 10 - Assignment 1

# Last 2 digit of Student ID

# My Student ID is 20235890 -> last 2 digits is 90

.eqv 7\_SEG\_LEFT 0xFFFF0011  # Address of the 7-SEG LED on the left

.eqv 7\_SEG\_RIGHT 0xFFFF0010     # Address of the 7-SEG LED on the right

.text

    main:

        li a1, 0x6F # Set value for 7 segments to Display 9

        jal SHOW\_7SEG\_LEFT # Show the result

        li a1, 0x3F # Set value for 7 segments to display 0

        jal SHOW\_7SEG\_RIGHT # Show the result

    exit:

        li a7, 10

        ecall

    end\_main:

    SHOW\_7SEG\_LEFT:

        li t1, 7\_SEG\_LEFT # Assign port's address

        sb a1, 0(t1) # Assign new value

        jr ra

    SHOW\_7SEG\_RIGHT:

        li t1, 7\_SEG\_RIGHT # Assign port's address

        sb a1, 0(t1) # Assign new value

        jr ra

**Result:**

A screenshot of a computer

Description automatically generated

**Assignment 2:**

**Code:**

# Lab 10 - Assignment 2

# Print last 2 digits ASCII Code of a letter input from keyboard

.eqv SEV\_SEG\_LEFT 0xFFFF0011    # Address of the 7-SEG LED on the left

.eqv SEV\_SEG\_RIGHT 0xFFFF0010   # Address of the 7-SEG LED on the right

.data

    num\_data: .byte 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F # 7-segment codes for 0-9

.text

main:

    input:

        # Input a Character from keyboard

        li a7, 12            # Syscall to read a character

        ecall

    process:

        # Calculate the last two digits of the ASCII code

        li t0, 10            # Base 10

        rem t1, a0, t0       # t1 = a0 % 10 (last digit)

        div a0, a0, t0       # a0 = a0 / 10

        rem t2, a0, t0       # t2 = a0 % 10 (second last digit)

        # Load 7-segment representation for the digits

        la t3, num\_data      # Load base address of num\_data

        add t4, t3, t1       # t4 = address of num\_data[t1] (last digit code)

        lb t4, 0(t4)         # Load value at num\_data[t1] into t4

        add t5, t3, t2       # t5 = address of num\_data[t2] (second last digit code)

        lb t5, 0(t5)         # Load value at num\_data[t2] into t5

    display:

        # Display the digits on the 7-segment LEDs

        li t6, SEV\_SEG\_RIGHT

        sb t4, 0(t6)         # Write the last digit to the right display

        li t6, SEV\_SEG\_LEFT

        sb t5, 0(t6)         # Write the second last digit to the left display

end:

    # End of program

    li a7, 10            # Exit syscall

    ecall

**Explaination:**

* I use an array num\_data to represent the digits on 7 LED Segment.
* Load the base num\_data start from 0 (num\_data[0] = 0) and add the value of the digit to take the value of the num\_data and represented in 7 LED Segment.

**Result:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Assignment 3:**

**Code:**

**Explaination:** (Idea, how your code work, task reponse, etc…)

**Result:** (Input, output for each case; is the result same as the theory, etc…)

**Assignment 4:**

**Code:**

# Lab 10 - Assignment 4

.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

.text

    li a0, KEY\_CODE

    li a1, KEY\_READY

    li s0, DISPLAY\_CODE

    li s1, DISPLAY\_READY

loop:

    WaitForKey:

        lw t1, 0(a1) # t1 = [a1] = KEY\_READY

        beq t1, zero, WaitForKey # if t1 == 0 then Polling

    ReadKey:

        lw t0, 0(a0) # t0 = [a0] = KEY\_CODE

    WaitForDis:

        lw t2, 0(s1) # t2 = [s1] = DISPLAY\_READY

        beq t2, zero, WaitForDis # if t2 == 0 then polling

    Encrypt:

        LowercaseCheck:

            # Check if it's a lowercase character (a-z)

            li t3, 97      # ASCII 'a'

            li t4, 122     # ASCII 'z'

            blt t0, t3, UppercaseCheck  # Skip if t0 < 'a'

            bgt t0, t4, UppercaseCheck  # Skip if t0 > 'z'

            addi t0, t0, -32              # Convert to uppercase

            j ShowKey                   # Jump to show the key

        UppercaseCheck:

            # Check if it's an uppercase character (A-Z)

            li t3, 65      # ASCII 'A'

            li t4, 90      # ASCII 'Z'

            blt t0, t3, DigitCheck   # Skip if t0 < 'A'

            bgt t0, t4, DigitCheck  # Skip if t0 > 'Z'

            addi t0, t0, 32              # Convert to lowercase

            j ShowKey                   # Jump to show the key

        DigitCheck:

            # Check if it's a digit (0-9)

            li t3, 48      # ASCII '0'

            li t4, 57      # ASCII '9'

            blt t0, t3, OtherChar  # Skip if t0 < '0'

            bgt t0, t4, OtherChar  # Skip if t0 > '9'

            j ShowKey               # Digit is displayed as is

        OtherChar:

            # Display '\*' for other characters

            li t0, 42      # ASCII '\*'

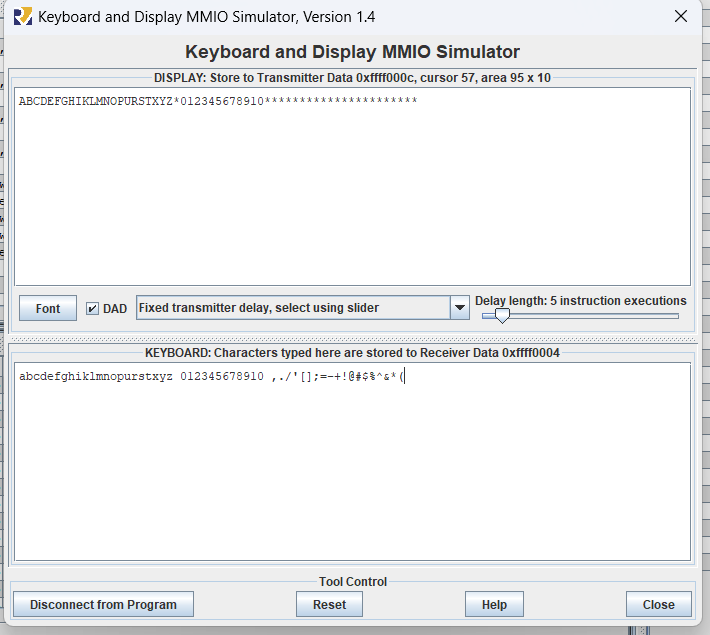
        j ShowKey          # Jump to show the key

    ShowKey:

        sw t0, 0(s0) # show key

        j loop

**Result:**



**Assignment 5:**

**Code:**

**Explaination:** (Idea, how your code work, task reponse, etc…)

**Result:** (Input, output for each case; is the result same as the theory, etc…)