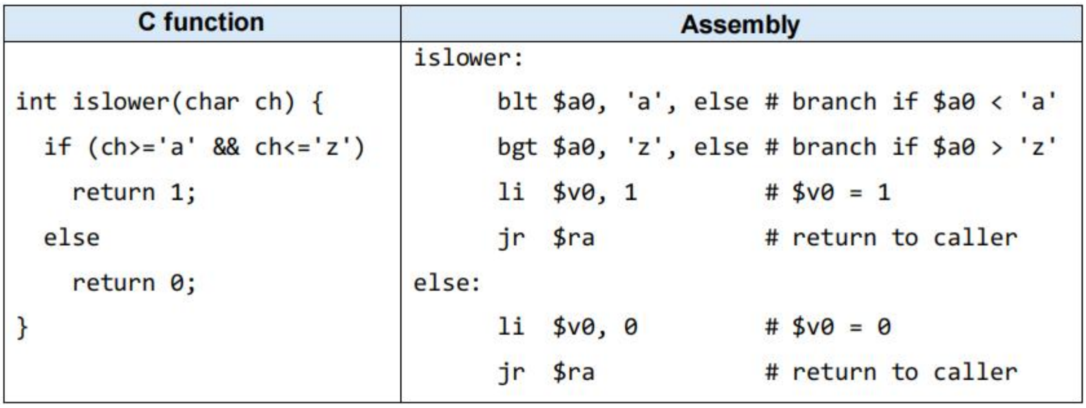
LAB #06 Report

Name:黄家睿　　　　ID number:202283890036　　　　Major:IOT

Task 1:

1. The function **islower** (shown in Fig. 1) tests whether a character ch is lowercase or not. Write the **main** function of a program that reads a character ch, calls the function islower, and then prints a message to indicate whether ch is a lowercase character or not.



C function	Assembly

`int islower(char ch) {`
`  if (ch>='a' && ch<='z')`
`    return 1;`
`  else`
`    return 0;`
`}`
	`islower:`
	`      blt $a0, 'a', else # branch if $a0 < 'a'`
	`      bgt $a0, 'z', else # branch if $a0 > 'z'`
	`      li  $v0, 1         # $v0 = 1`
	`      jr  $ra            # return to caller`
	`else:`
	`      li  $v0, 0         # $v0 = 0`
	`      jr  $ra            # return to caller`

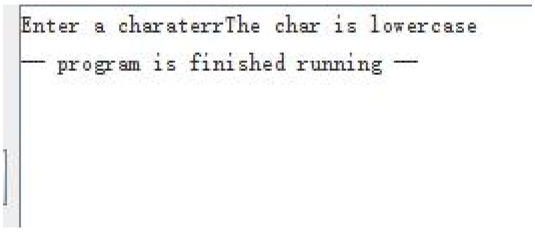
Solution

Code：

```
.data
str1: .asciiz "The char is lowercase"
str2: .asciiz "The char is upcase"
prompt:.asciiz "Enter a charater"

.text
main:
    # Print the prompt message
    li $v0, 4                 # syscall code for print string
    la $a0, prompt            # address of the prompt message
        syscall

      # Read a character
        li $v0, 12                # syscall code for read character
        syscall
        move $a0, $v0             # move the character to $a0
        jal islower

        beq $v0,1,print_lowercase
```

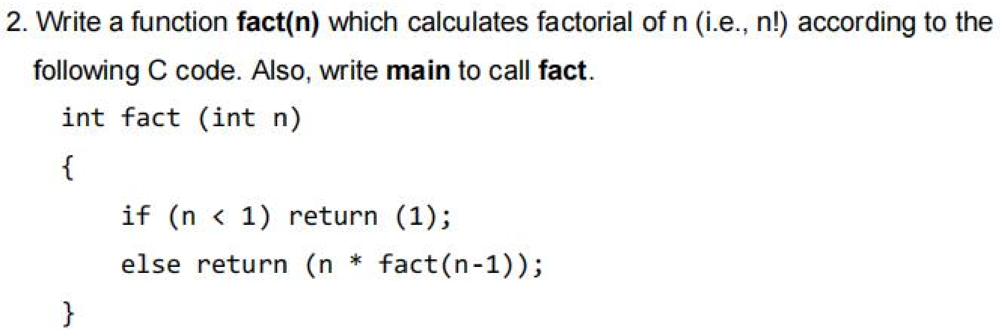
```
print_not_lowercase:
        li $v0, 4               # syscall code for print string
    la $a0, str2    # address of the not lowercase message
        syscall
    j end_program

print_lowercase:
        li $v0, 4               # syscall code for print string
    la $a0, str1   # address of the lowercase message
        syscall

end_program:
    li $v0, 10              # syscall code for exit
    syscall

islower:
    blt $a0,'a',else
    bgt $a0,'z',else
    li $v0,1
    jr $ra

else:
    li $v0,0
    jr $ra
```

output：



```
Enter a charaterrThe char is lowercase
— program is finished running —
```

Task 2:



2. Write a function **fact(n)** which calculates factorial of n (i.e., n!) according to the following C code. Also, write **main** to call **fact**.

```
int fact (int n)
{
    if (n < 1) return (1);
    else return (n * fact(n-1));
}
```

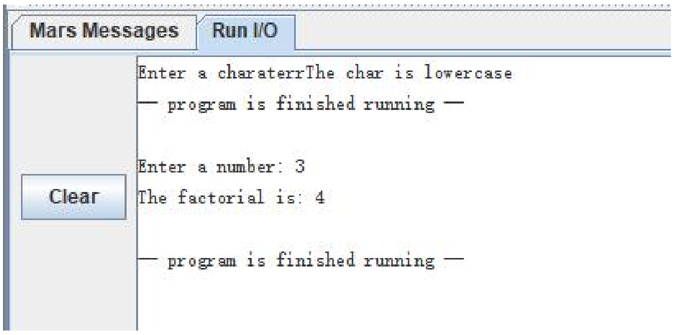
Solution:

Code:

```
.data
prompt: .asciiz "Enter a number: "
result_msg: .asciiz "The factorial is: "
newline: .asciiz "\n"

    .text
    .globl main

main:
    # Print the prompt message
    li $v0, 4                    # syscall code for print string
    la $a0, prompt                # address of the prompt message
    syscall

    # Read an integer from the user
    li $v0, 5                    # syscall code for read integer
    syscall
    move $a0, $v0                  # move the input number to $a0

    # Call the fact function
    jal fact

    # Print the result message
    li $v0, 4                    # syscall code for print string
    la $a0, result_msg          # address of the result message
    syscall
```

```
  # Print the factorial result
      move $a0, $v0               # move the result from $v0 to $a0
      li $v0, 1                  # syscall code for print integer
      syscall

      # Print a newline
      li $v0, 4                  # syscall code for print string
      la $a0, newline            # address of the newline character
      syscall

      # Exit the program
      li $v0, 10                 # syscall code for exit
      syscall

      fact:
      addi $sp, $sp, -8        # Create space on the stack
      sw $ra, 4($sp)            # Save return address
      sw $a0, 0($sp)            # Save argument n

      bge $a0, 1, recurse      # If n >= 1, recurse
      li $v0, 1                 # Base case: fact(0) = 1 or fact(n) for n < 1 = 1
      j end_fact                 # Return

recurse:
      addi $a0, $a0, -1        # Calculate fact(n-1)
      jal fact                  # Recursive call
      lw $a0, 0($sp)            # Restore argument n
      mul $v0, $a0, $v0         # Multiply n * fact(n-1)

end_fact:
      lw $ra, 4($sp)            # Restore return address
      addi $sp, $sp, 8         # Restore stack
      jr $ra                     # Return
```

Output:



```
Enter a charaterrThe char is lowercase
-- program is finished running --

Enter a number: 3
The factorial is: 4

-- program is finished running --
```