BRANCHING OPERATIONS

LAB # 02

Fall 2023

CSE-304L Computer Organization and Architecture Lab

Submitted by: **Ali Asghar**

Registration No.: **21PWCSE2059**

Class Section: **C**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Submitted to:

Dr. Bilal Habib

Date:

14th October 2023

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

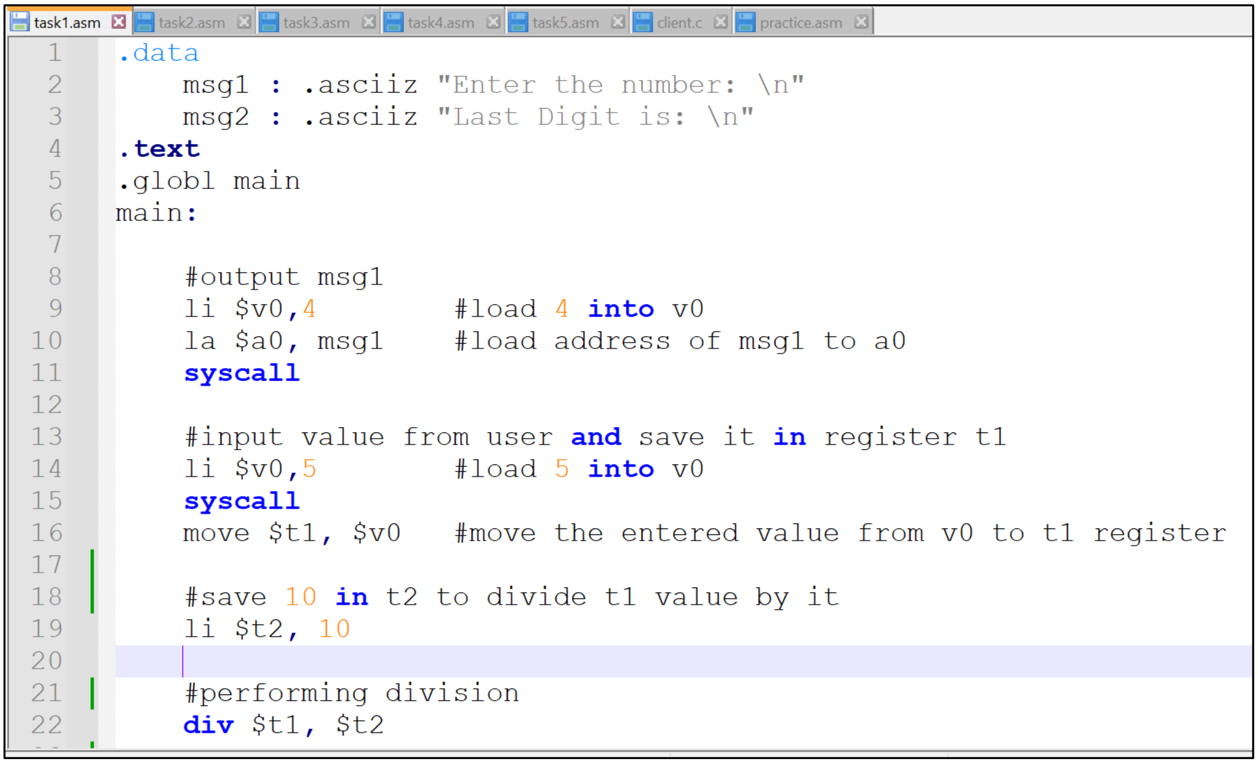
ASSESSMENT RUBRICS COA LABS

LAB REPORT ASSESSMENT				
Criteria	**Excellent**	**Average**	**Nill**	**Marks Obtained**
1. Objectives of Lab	All objectives of lab are properly covered [Marks 10]	Objectives of lab are partially covered [Marks 5]	Objectives of lab are not shown [Marks 0]	
2. MIPS instructions with Comments and proper indentations.	All the instructions are well written with comments explaining the code and properly indented [Marks 20]	Some instructions are missing are poorly commented code [Marks 10]	The instructions are not properly written [Marks 0]	
3. Simulation run without error and warnings	The code is running in the simulator without any error and warnings [Marks 10]	The code is running but with some warnings or errors. [Marks 5]	The code is written but not running due to errors [Marks 0]	
4. Procedure	All the instructions are written with proper procedure [Marks 20]	Some steps are missing [Marks 10]	steps are totally missing [Marks 0]	
5. OUTPUT	Proper output of the code written in assembly [Marks 20]	Some of the outputs are missing [Marks 10]	No or wrong output [Marks 0]	
6. Conclusion	Conclusion about the lab is shown and written [Marks 20]	Conclusion about the lab is partially shown [Marks 10]	Conclusion about the lab is not shown[Marks0]	
7. Cheating			Any kind of cheating will lead to 0 Marks	
Total Marks Obtained:__________ Instructor Signature: ______________________				

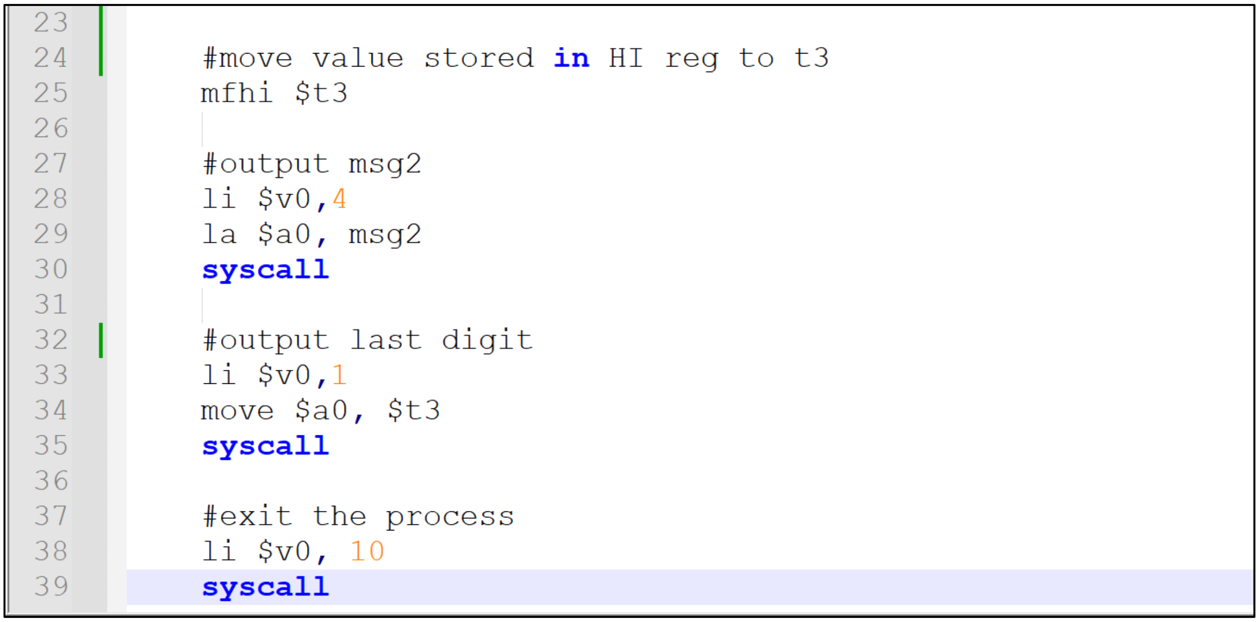
Task 1:

Enter a number 5432 from user and then display the last digit in the console. (hint: use mfhi).

Code:

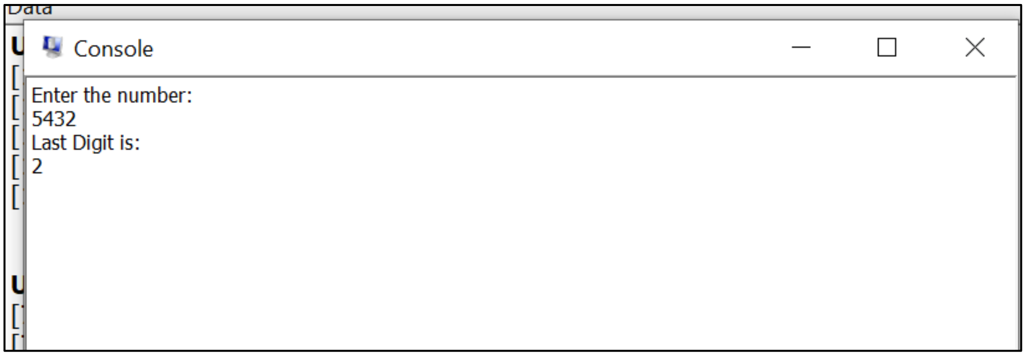


```
.data
    msg1 : .asciiz "Enter the number: \n"
    msg2 : .asciiz "Last Digit is: \n"
.text
.globl main
main:

    #output msg1
    li $v0,4        #load 4 into v0
    la $a0, msg1    #load address of msg1 to a0
    syscall

    #input value from user and save it in register t1
    li $v0,5        #load 5 into v0
    syscall
    move $t1, $v0   #move the entered value from v0 to t1 register

    #save 10 in t2 to divide t1 value by it
    li $t2, 10

    #performing division
    div $t1, $t2
```



```

    #move value stored in HI reg to t3
    mfhi $t3

    #output msg2
    li $v0,4
    la $a0, msg2
    syscall

    #output last digit
    li $v0,1
    move $a0, $t3
    syscall

    #exit the process
    li $v0, 10
    syscall
```

Output:

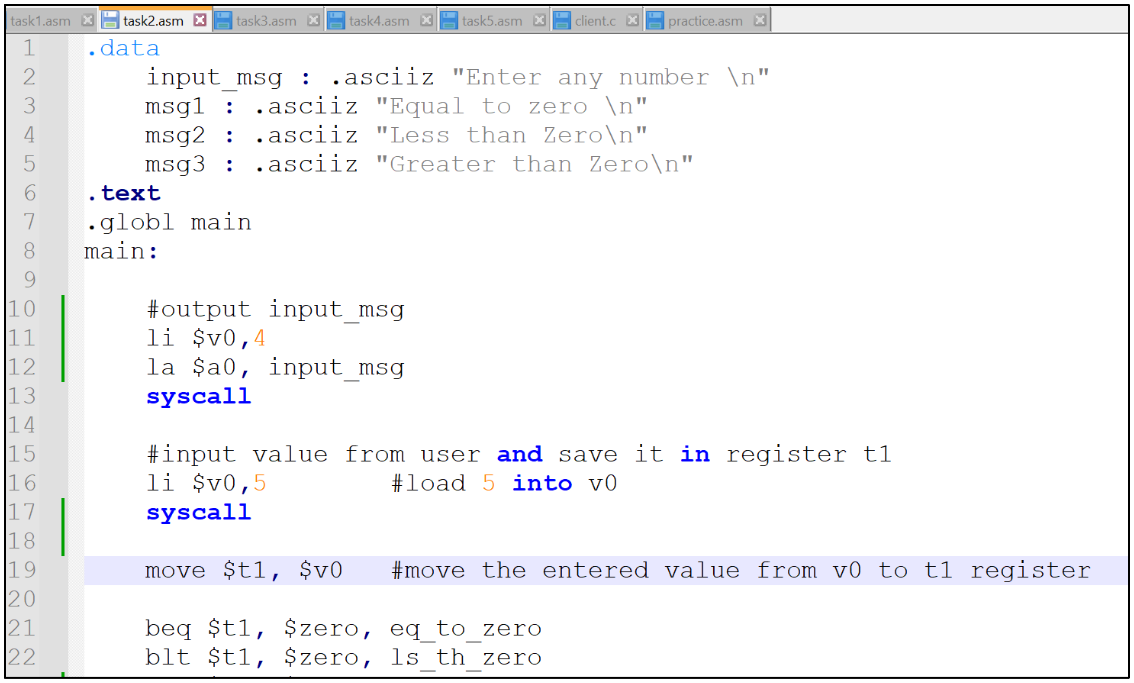


```
Enter the number:
5432
Last Digit is:
2
```

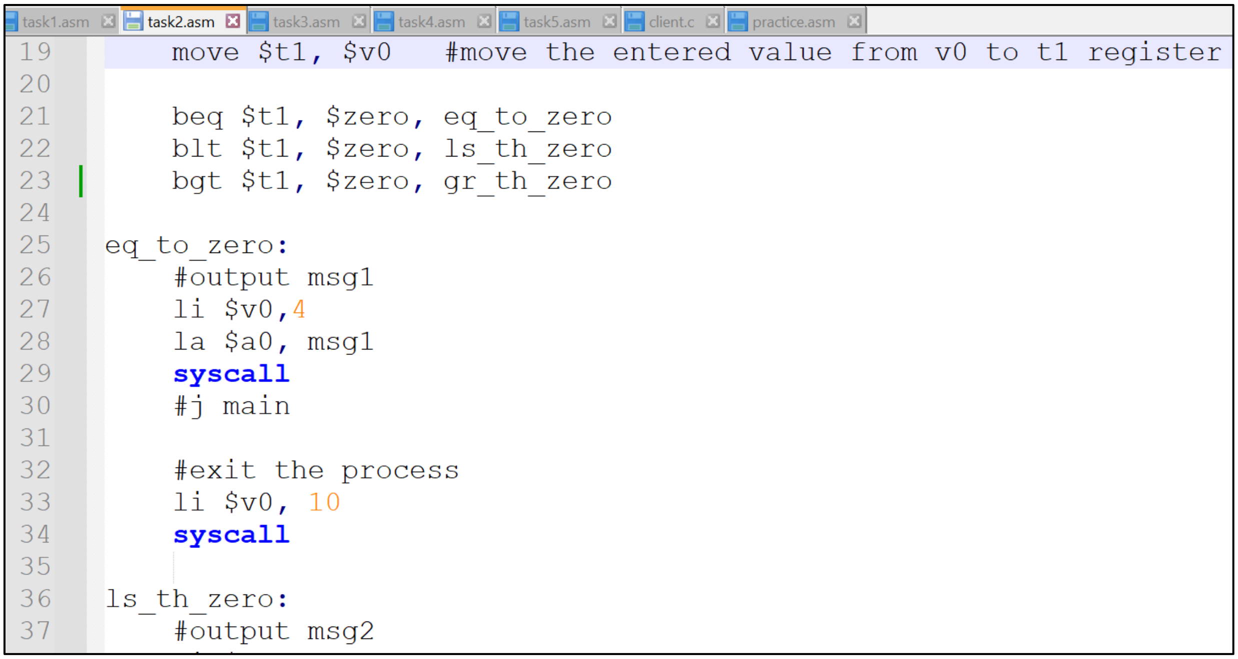
Task 2:

Check whether a number input by user is negative or equal to zero or greater then zero using branching (Use bgt or ble).

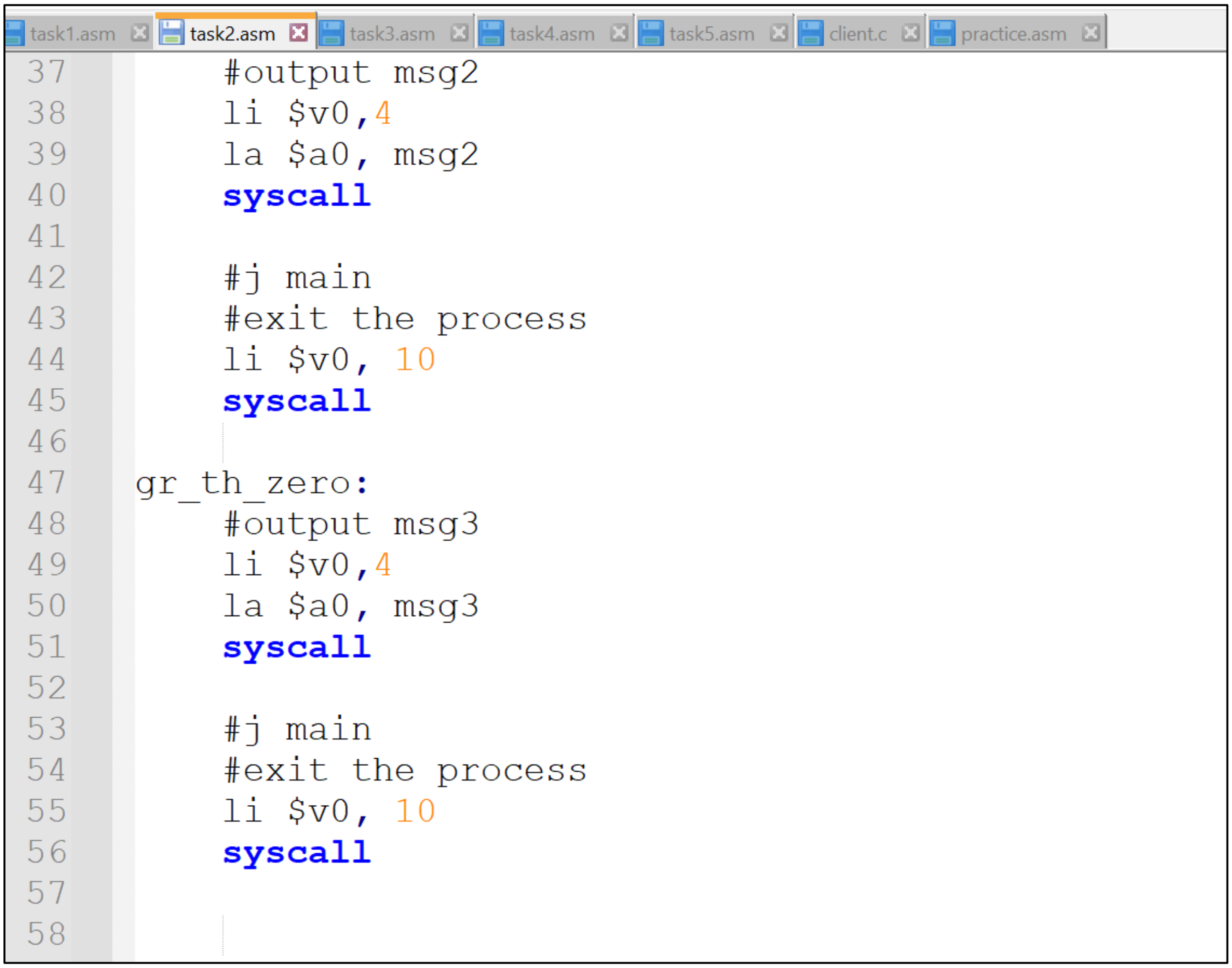
Code:



```
.data
    input_msg : .asciiz "Enter any number \n"
    msg1 : .asciiz "Equal to zero \n"
    msg2 : .asciiz "Less than Zero\n"
    msg3 : .asciiz "Greater than Zero\n"
.text
.globl main
main:

    #output input_msg
    li $v0,4
    la $a0, input_msg
    syscall

    #input value from user and save it in register t1
    li $v0,5        #load 5 into v0
    syscall

    move $t1, $v0   #move the entered value from v0 to t1 register

    beq $t1, $zero, eq_to_zero
    blt $t1, $zero, ls_th_zero
```

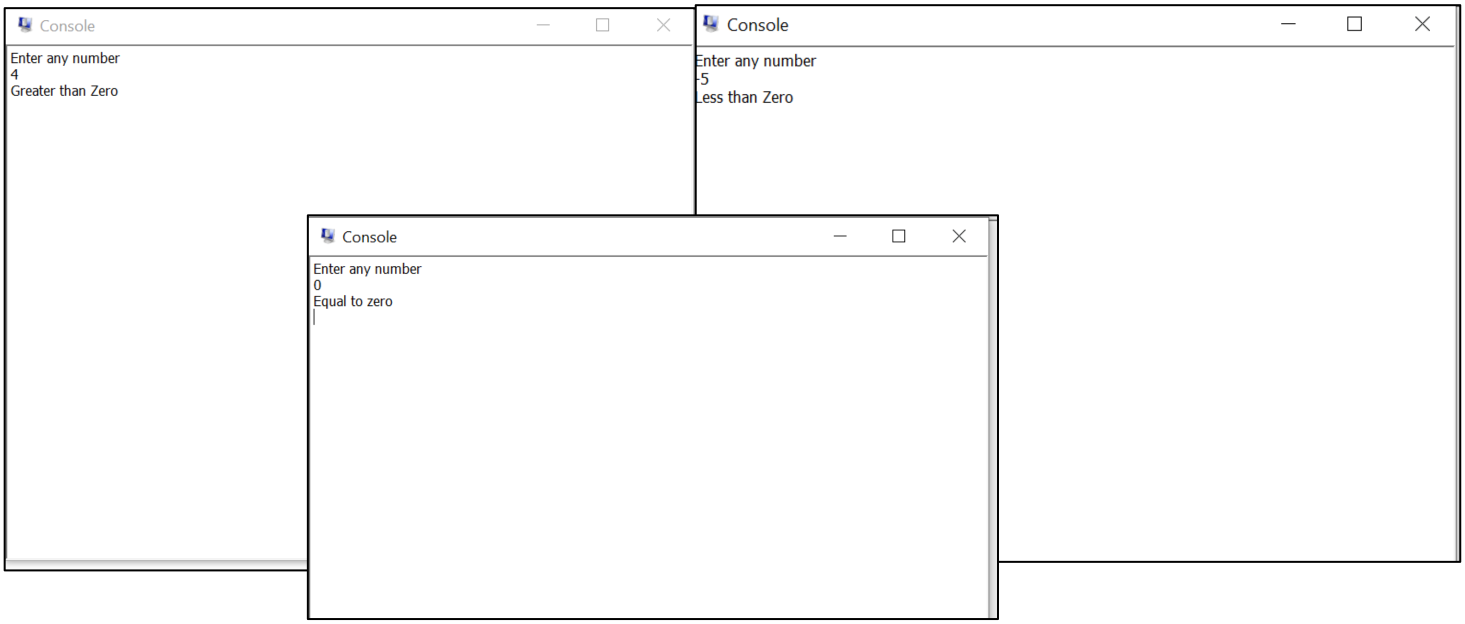


```
    move $t1, $v0   #move the entered value from v0 to t1 register

    beq $t1, $zero, eq_to_zero
    blt $t1, $zero, ls_th_zero
    bgt $t1, $zero, gr_th_zero

eq_to_zero:
    #output msg1
    li $v0,4
    la $a0, msg1
    syscall
    #j main

    #exit the process
    li $v0, 10
    syscall

ls_th_zero:
    #output msg2
```



```
    #output msg2
    li $v0,4
    la $a0, msg2
    syscall

    #j main
    #exit the process
    li $v0, 10
    syscall

 gr_th_zero:
    #output msg3
    li $v0,4
    la $a0, msg3
    syscall

    #j main
    #exit the process
    li $v0, 10
    syscall


```

Output:



```
Enter any number
4
Greater than Zero
```

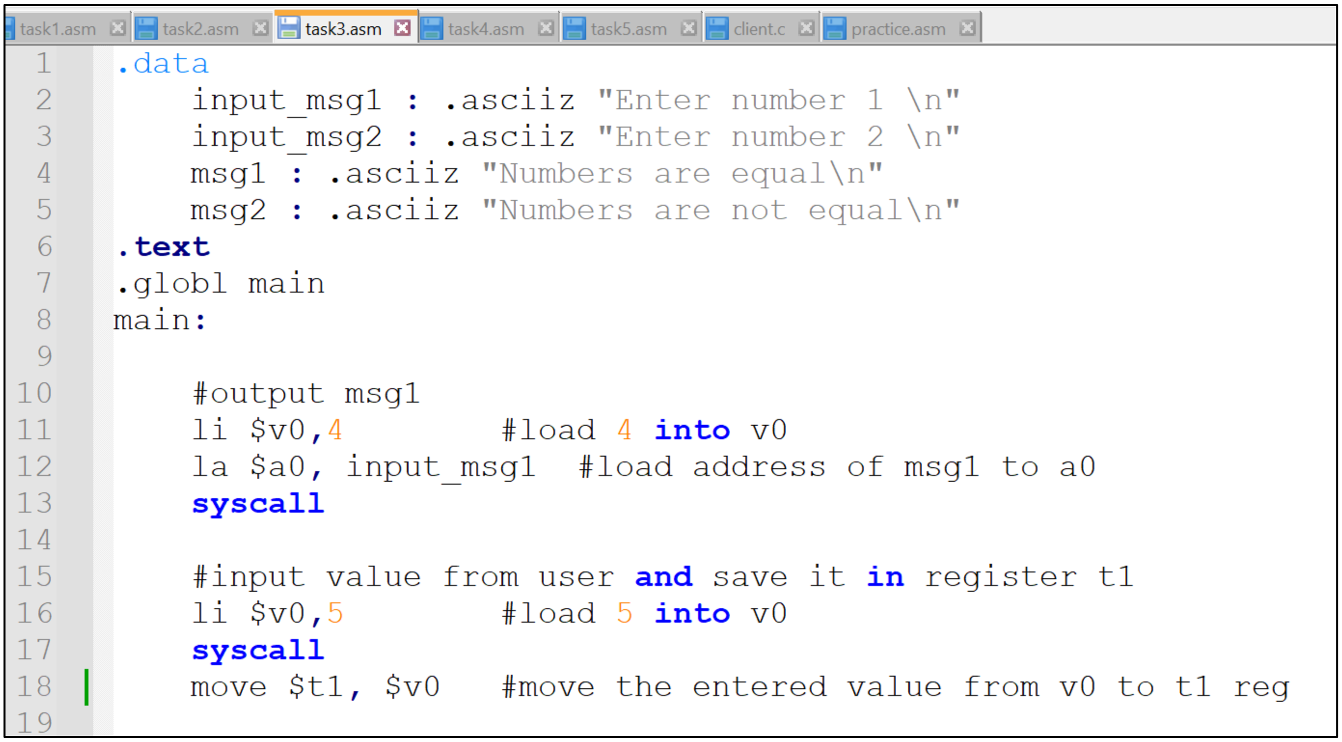
```
Enter any number
-5
Less than Zero
```

```
Enter any number
0
Equal to zero
```

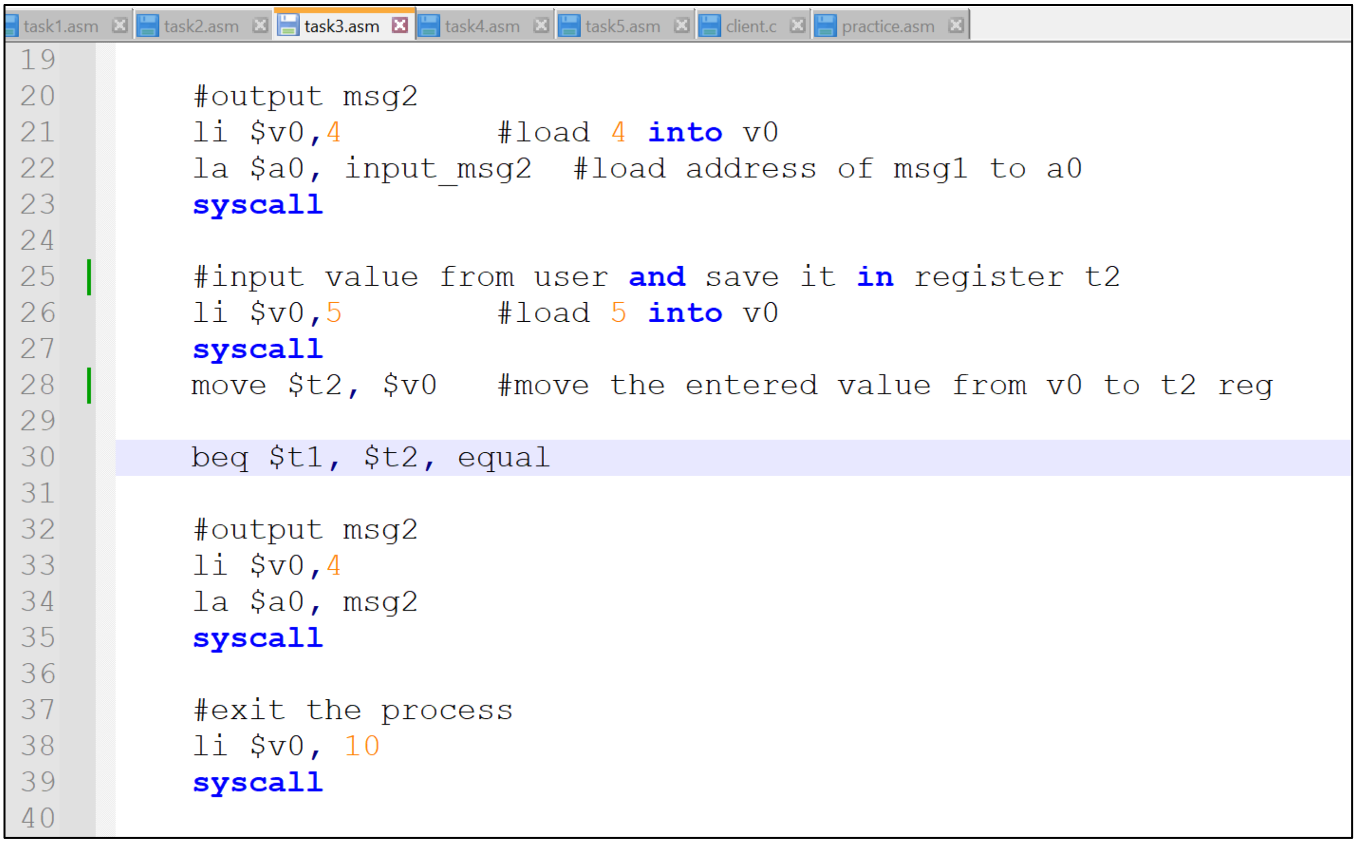
Task 3:

Check using branch whether the number input by user are equal or not (Use beq).

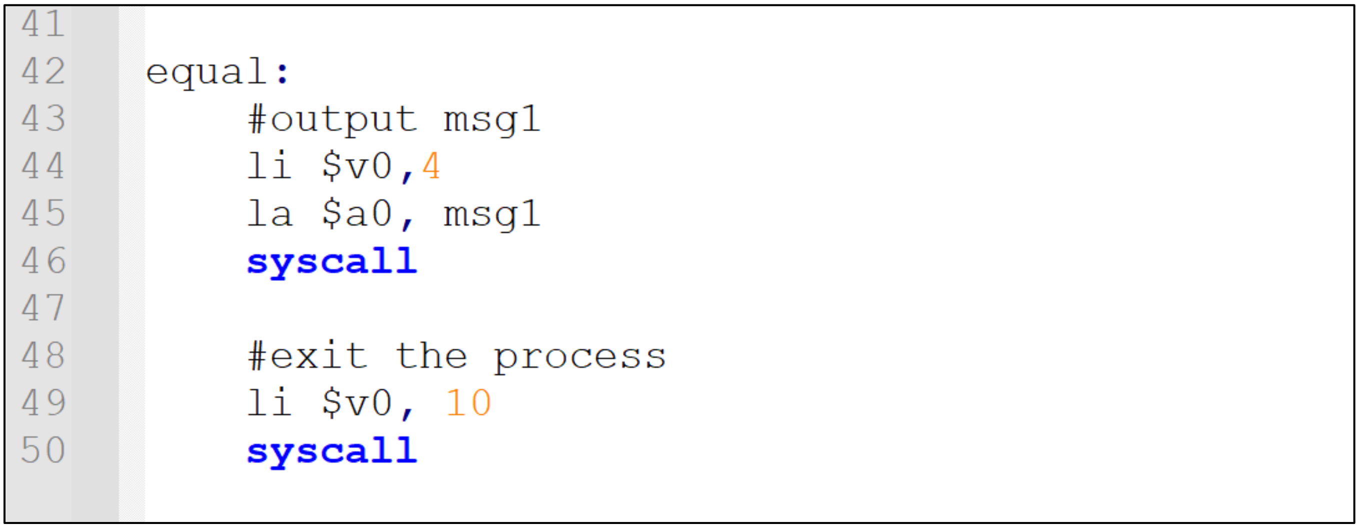
Code:



```
.data
    input_msg1 : .asciiz "Enter number 1 \n"
    input_msg2 : .asciiz "Enter number 2 \n"
    msg1 : .asciiz "Numbers are equal\n"
    msg2 : .asciiz "Numbers are not equal\n"
.text
.globl main
main:

    #output msg1
    li $v0,4          #load 4 into v0
    la $a0, input_msg1  #load address of msg1 to a0
    syscall

    #input value from user and save it in register t1
    li $v0,5          #load 5 into v0
    syscall
    move $t1, $v0    #move the entered value from v0 to t1 reg

```

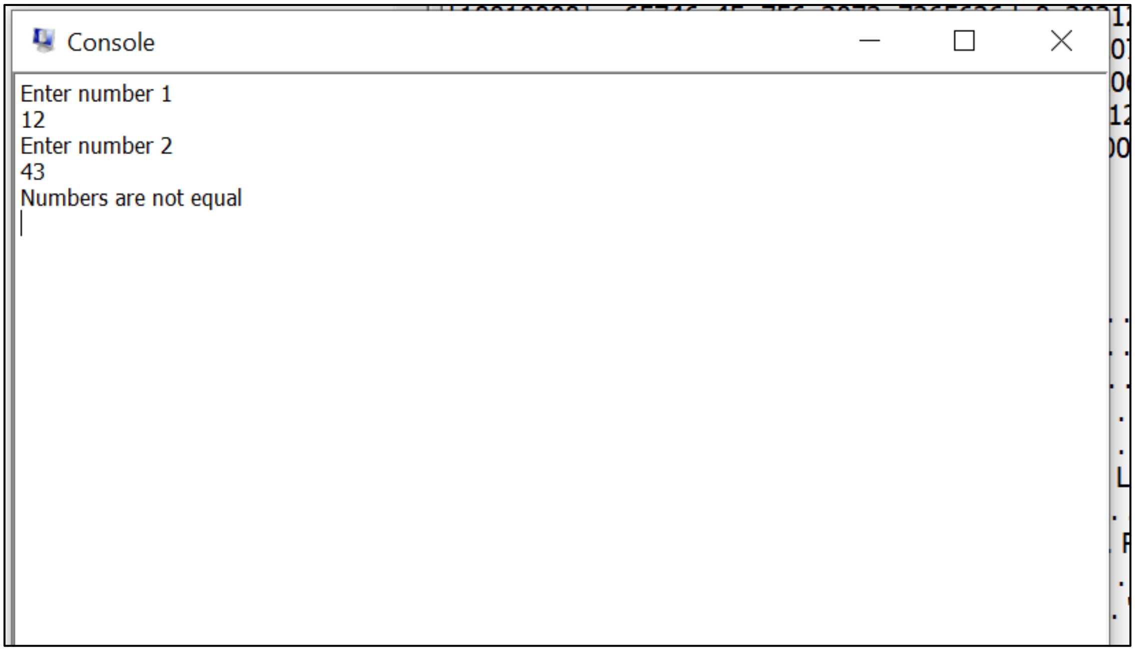


```

    #output msg2
    li $v0,4        #load 4 into v0
    la $a0, input_msg2  #load address of msg1 to a0
    syscall

    #input value from user and save it in register t2
    li $v0,5        #load 5 into v0
    syscall
    move $t2, $v0   #move the entered value from v0 to t2 reg

    beq $t1, $t2, equal

    #output msg2
    li $v0,4
    la $a0, msg2
    syscall

    #exit the process
    li $v0, 10
    syscall

```

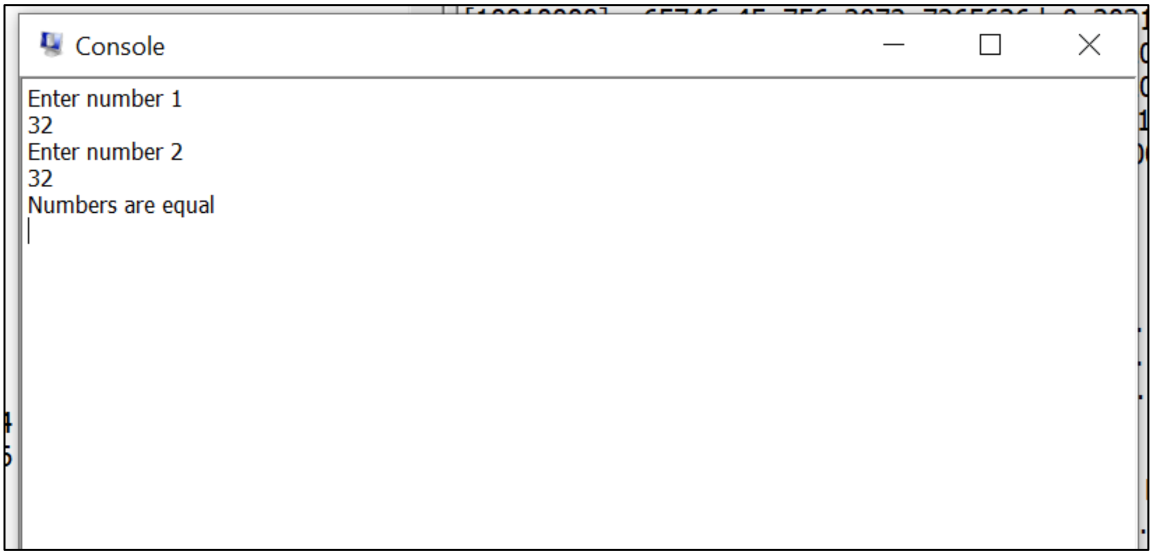


```

equal:
    #output msg1
    li $v0,4
    la $a0, msg1
    syscall

    #exit the process
    li $v0, 10
    syscall
```

Output:



```
Enter number 1
12
Enter number 2
43
Numbers are not equal
```



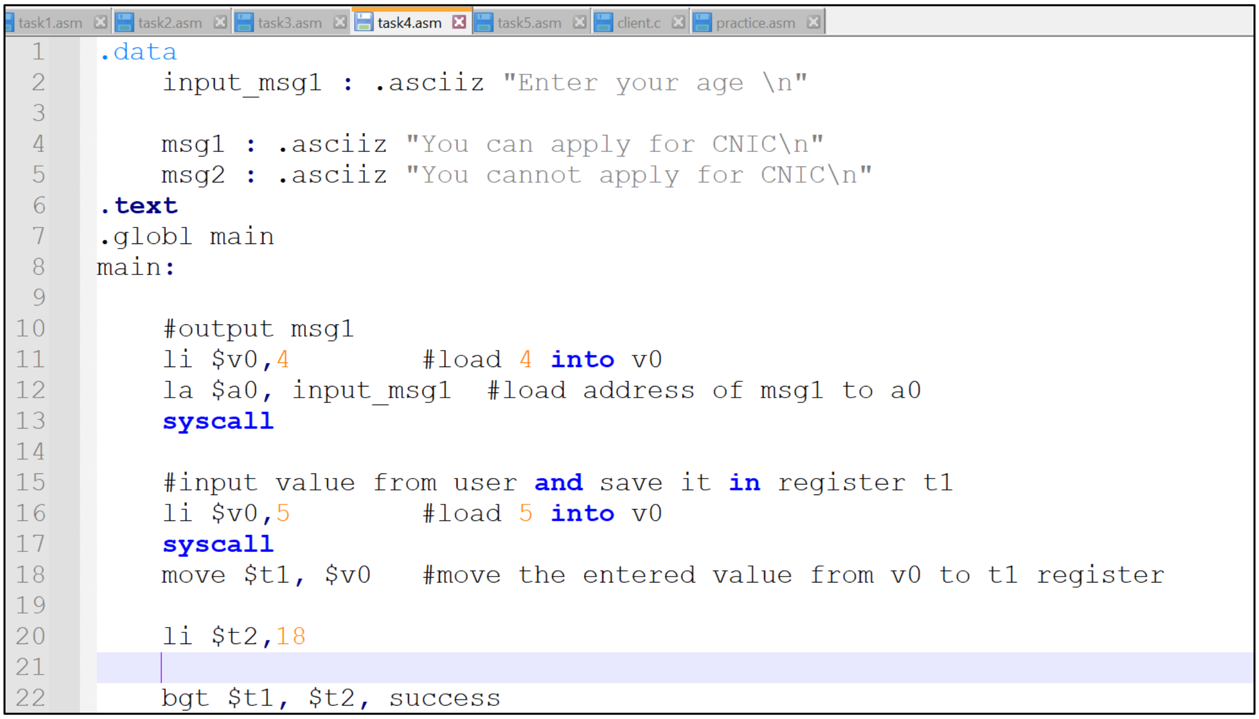
```
Enter number 1
32
Enter number 2
32
Numbers are equal
```

Task 4:

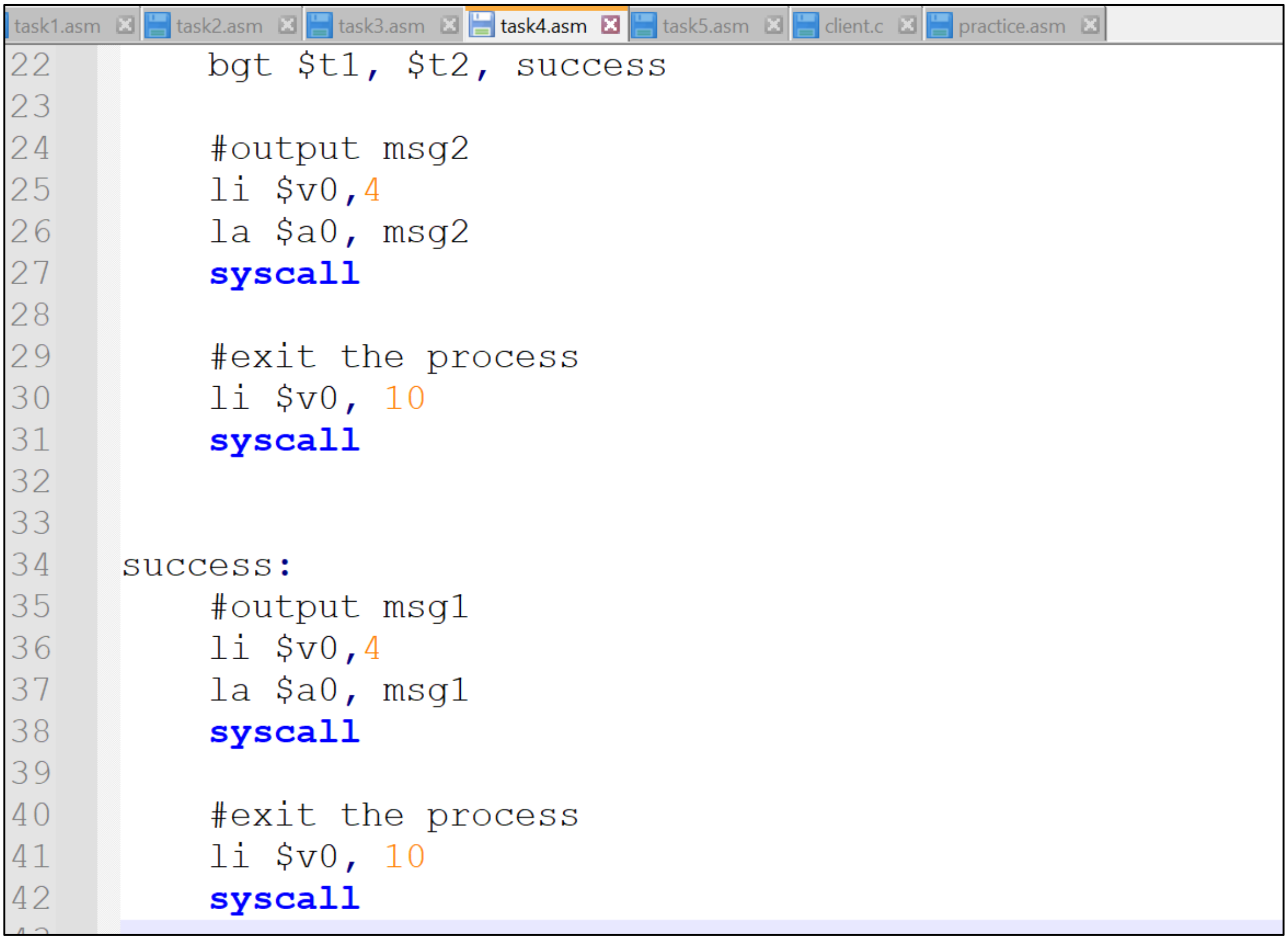
Write the assembly of the below C++ code:

```
Int age;
Cout<<”enter your age”<<endl;
Cin>>age;
If(age > 18)
{
Cout<<”you can apply for CNIC”<<endl;
}
Else
{
Cout<<”you cannot apply for CNIC”<<endl;
}
```

Code:

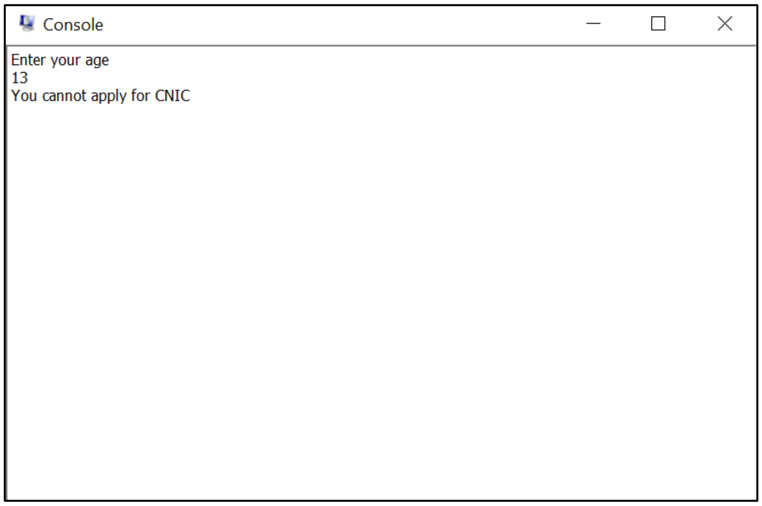


```
.data
    input_msg1 : .asciiz "Enter your age \n"

    msg1 : .asciiz "You can apply for CNIC\n"
    msg2 : .asciiz "You cannot apply for CNIC\n"
.text
.globl main
main:

    #output msg1
    li $v0,4         #load 4 into v0
    la $a0, input_msg1  #load address of msg1 to a0
    syscall

    #input value from user and save it in register t1
    li $v0,5         #load 5 into v0
    syscall
    move $t1, $v0    #move the entered value from v0 to t1 register

    li $t2,18

    bgt $t1, $t2, success
```

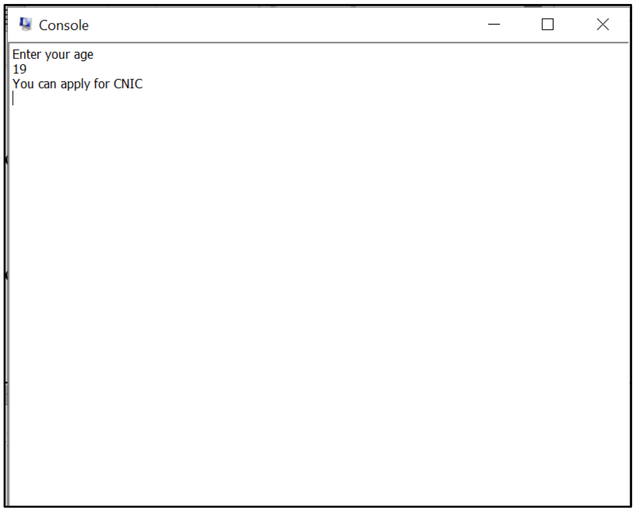


```
    bgt $t1, $t2, success

    #output msg2
    li $v0,4
    la $a0, msg2
    syscall

    #exit the process
    li $v0, 10
    syscall


success:
    #output msg1
    li $v0,4
    la $a0, msg1
    syscall

    #exit the process
    li $v0, 10
    syscall
```

Output:



```
Enter your age
13
You cannot apply for CNIC
```



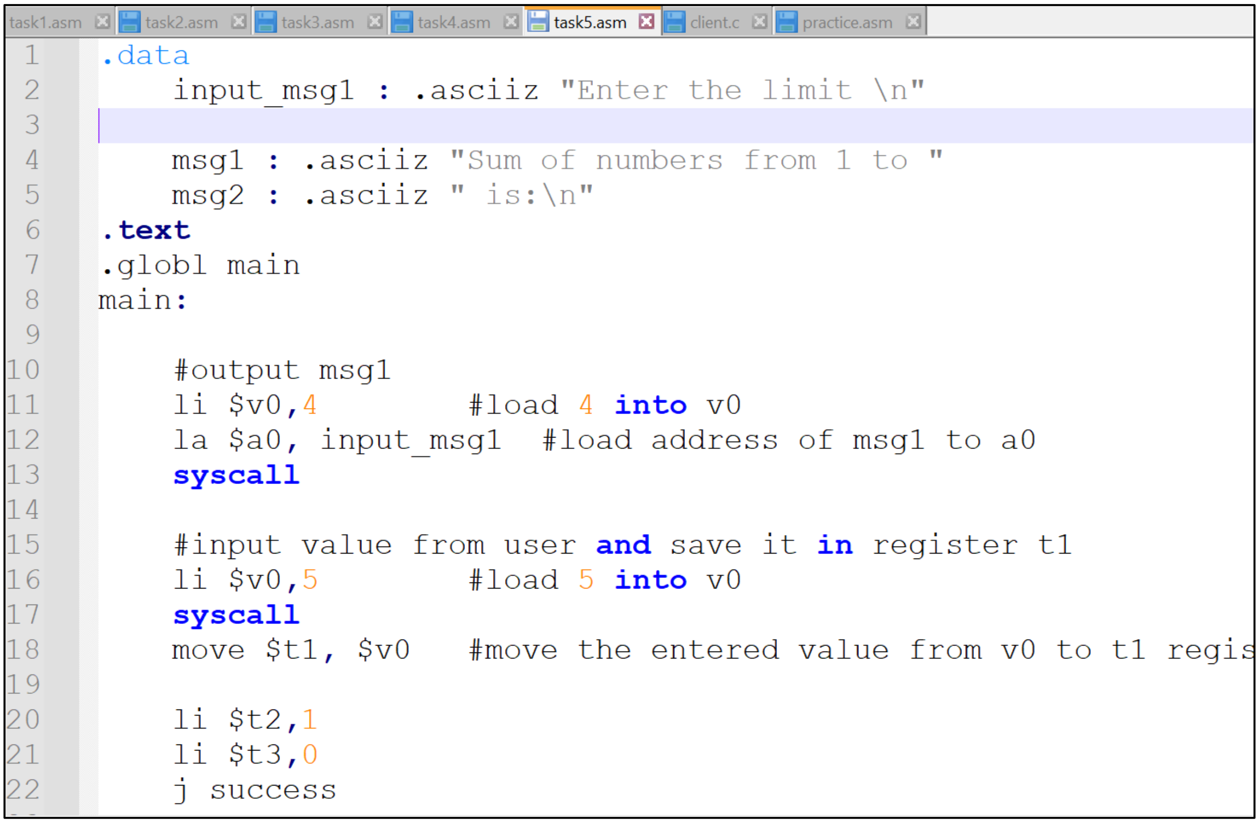
```
Enter your age
19
You can apply for CNIC
```

Task 5:

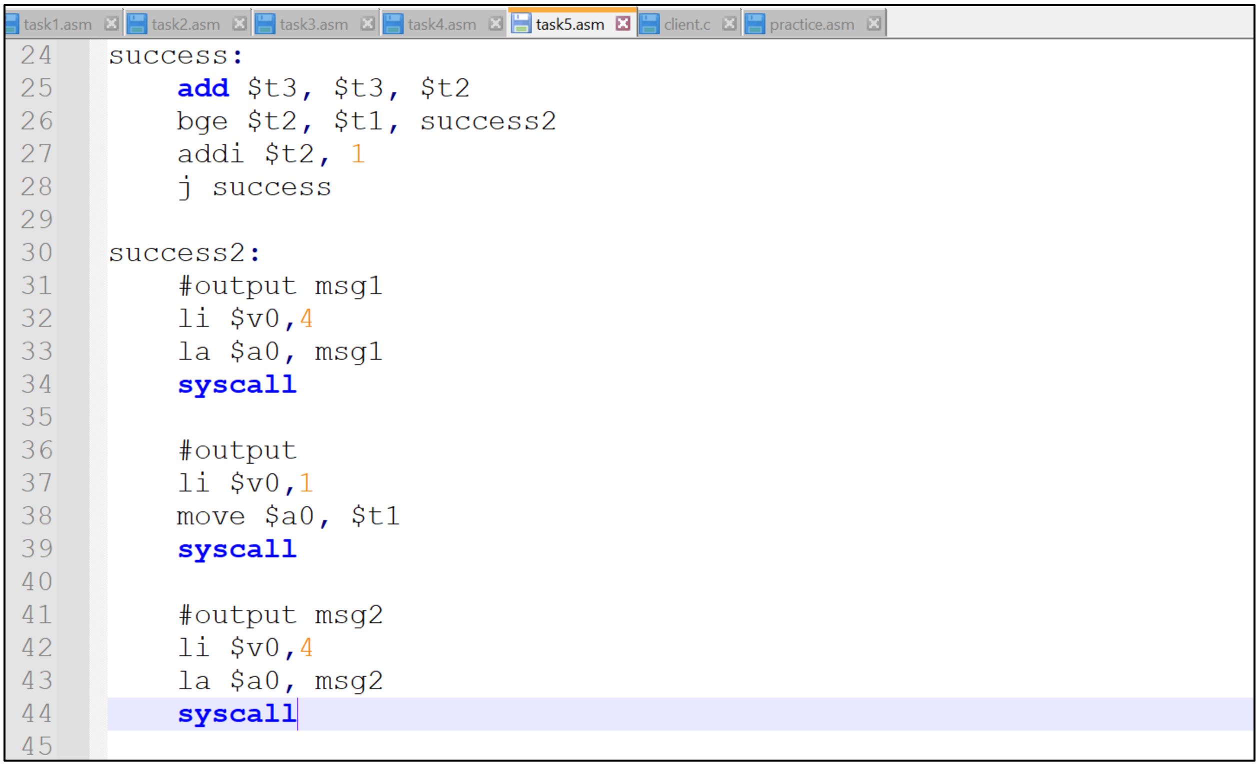
Write a program which take a limit from user and compute the sum of numbers from 0 to the limit (Use bqe, add, addi, and J (jump)). Below is the C++ language code:

```
Int limit;

Int sum;

Cout<<"Enter a number"<<endl;

Cin>>limit;

   for (int i = 1; i <= limit; ++i) {
      sum += i;
   }
   Cout<<"sum of numbers from 1 to <<limit<<"is"<<sum<<endl;
```

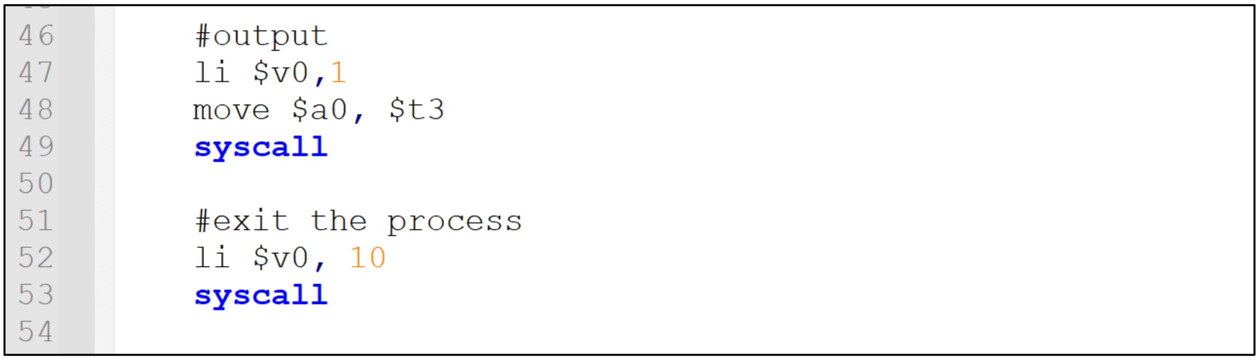
Code:



```
.data
    input_msg1 : .asciiz "Enter the limit \n"

    msg1 : .asciiz "Sum of numbers from 1 to "
    msg2 : .asciiz " is:\n"
.text
.globl main
main:

    #output msg1
    li $v0,4        #load 4 into v0
    la $a0, input_msg1  #load address of msg1 to a0
    syscall

    #input value from user and save it in register t1
    li $v0,5        #load 5 into v0
    syscall
    move $t1, $v0   #move the entered value from v0 to t1 regis

    li $t2,1
    li $t3,0
    j success
```

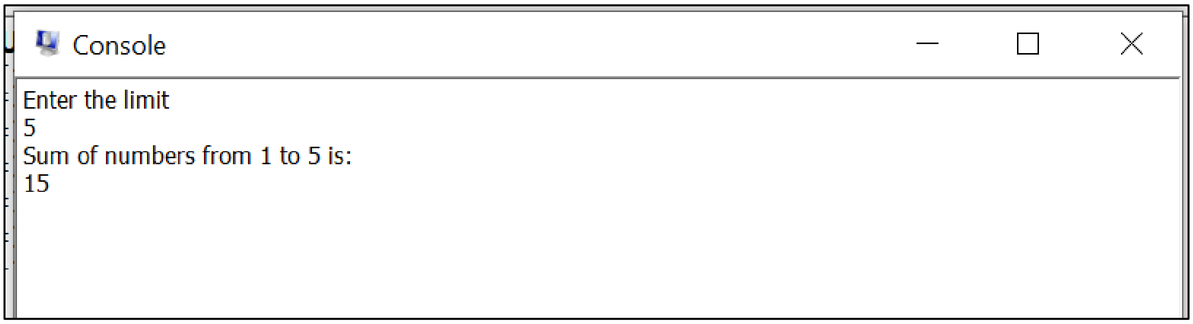


```
success:
    add $t3, $t3, $t2
    bge $t2, $t1, success2
    addi $t2, 1
    j success

success2:
    #output msg1
    li $v0,4
    la $a0, msg1
    syscall

    #output
    li $v0,1
    move $a0, $t1
    syscall

    #output msg2
    li $v0,4
    la $a0, msg2
    syscall

```



```
    #output
    li $v0,1
    move $a0, $t3
    syscall

    #exit the process
    li $v0, 10
    syscall

```

Output:



```
Enter the limit
5
Sum of numbers from 1 to 5 is:
15
```

Conclusion:

In this lab, I learned about the branching instructions(Control Structures) in MIPS Assembly.