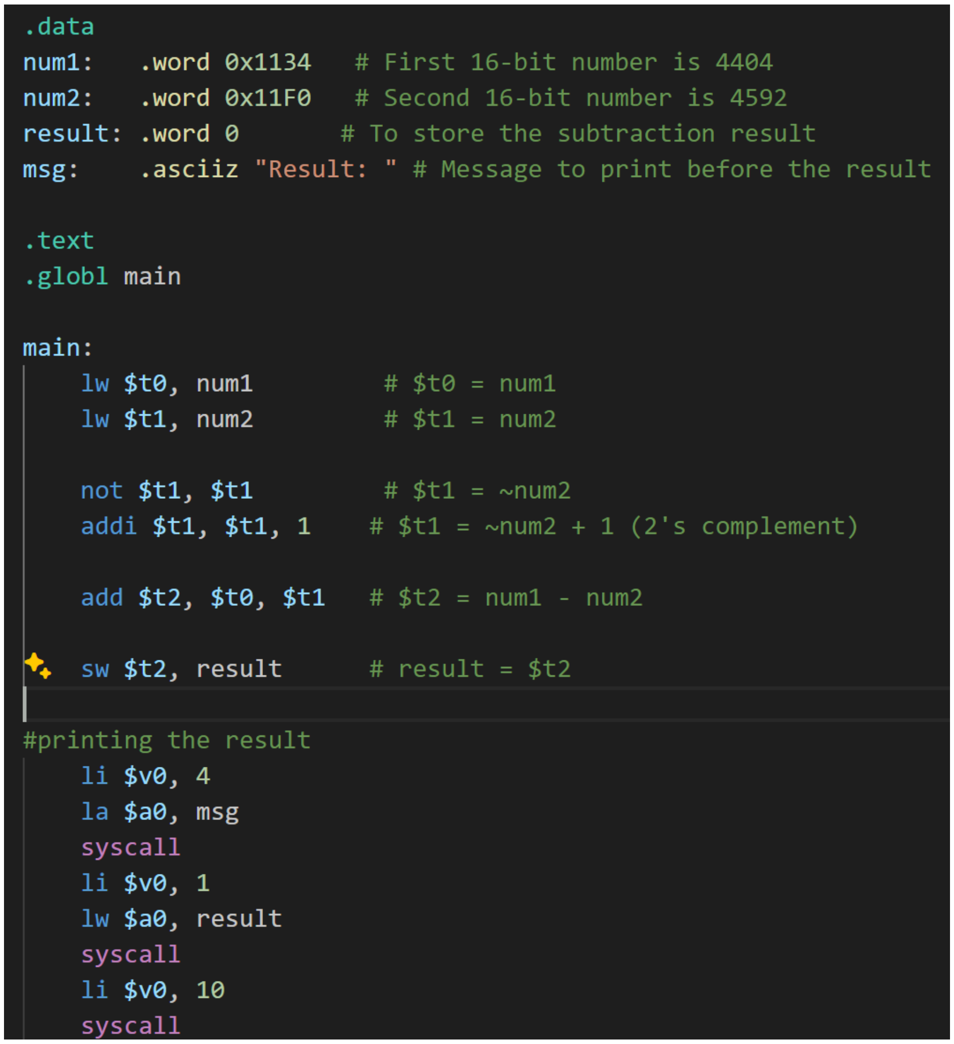
ES215: Assignment 3

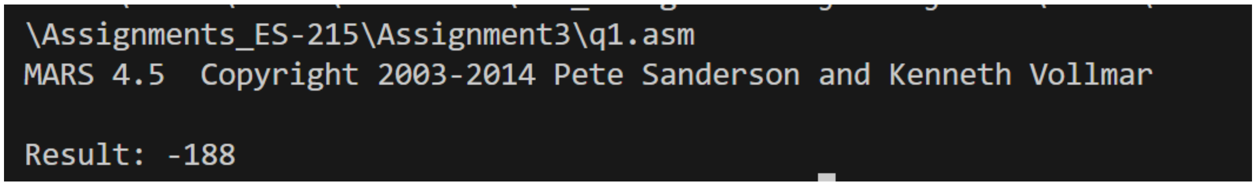
1.Write a program in assembly language to subtract two 16 bit numbers without using the subtraction instruction. Note: the numbers have to be fetched from the memory.

Program:



```
.data
num1:   .word 0x1134   # First 16-bit number is 4404
num2:   .word 0x11F0   # Second 16-bit number is 4592
result: .word 0       # To store the subtraction result
msg:    .asciiz "Result: " # Message to print before the result

.text
.globl main

main:
    lw $t0, num1         # $t0 = num1
    lw $t1, num2         # $t1 = num2

    not $t1, $t1         # $t1 = ~num2
    addi $t1, $t1, 1    # $t1 = ~num2 + 1 (2's complement)

    add $t2, $t0, $t1   # $t2 = num1 - num2

    sw $t2, result      # result = $t2

#printing the result
    li $v0, 4
    la $a0, msg
    syscall
    li $v0, 1
    lw $a0, result
    syscall
    li $v0, 10
    syscall
```

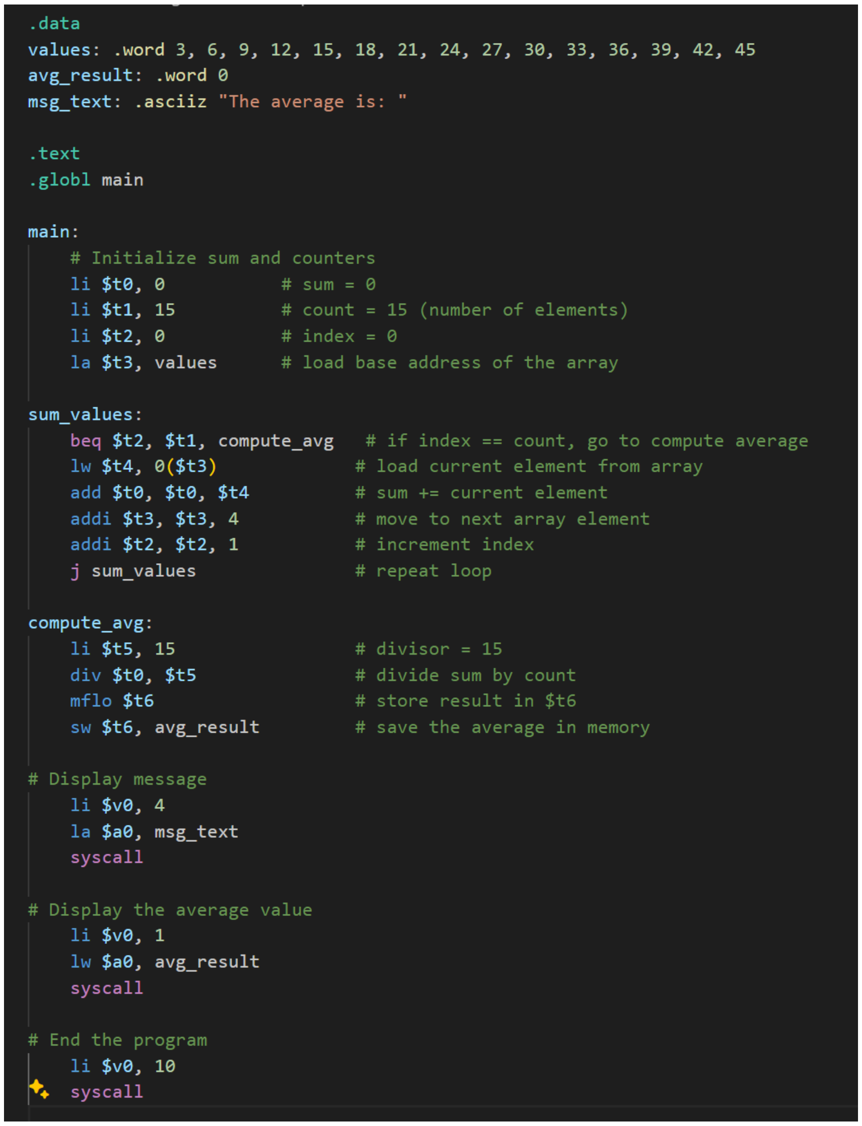
Output:



```
\Assignments_ES-215\Assignment3\q1.asm
MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar

Result: -188
```

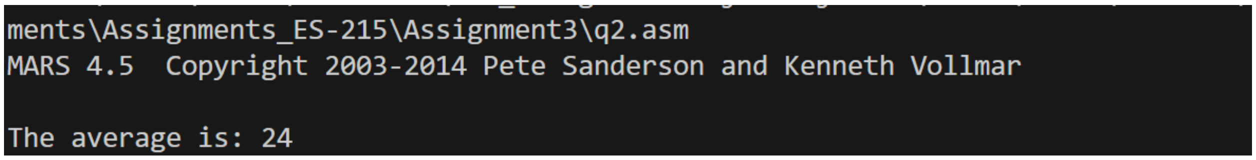
2.Write an assembly language program to find an average of 15 numbers stored at consecutive locations in memory.

Program:



```
.data
values: .word 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45
avg_result: .word 0
msg_text: .asciiz "The average is: "

.text
.globl main

main:
    # Initialize sum and counters
    li $t0, 0           # sum = 0
    li $t1, 15          # count = 15 (number of elements)
    li $t2, 0           # index = 0
    la $t3, values      # load base address of the array

sum_values:
    beq $t2, $t1, compute_avg   # if index == count, go to compute average
    lw $t4, 0($t3)             # load current element from array
    add $t0, $t0, $t4          # sum += current element
    addi $t3, $t3, 4           # move to next array element
    addi $t2, $t2, 1           # increment index
    j sum_values               # repeat loop

compute_avg:
    li $t5, 15                 # divisor = 15
    div $t0, $t5               # divide sum by count
    mflo $t6                   # store result in $t6
    sw $t6, avg_result         # save the average in memory

# Display message
    li $v0, 4
    la $a0, msg_text
    syscall

# Display the average value
    li $v0, 1
    lw $a0, avg_result
    syscall

# End the program
    li $v0, 10
    syscall
```

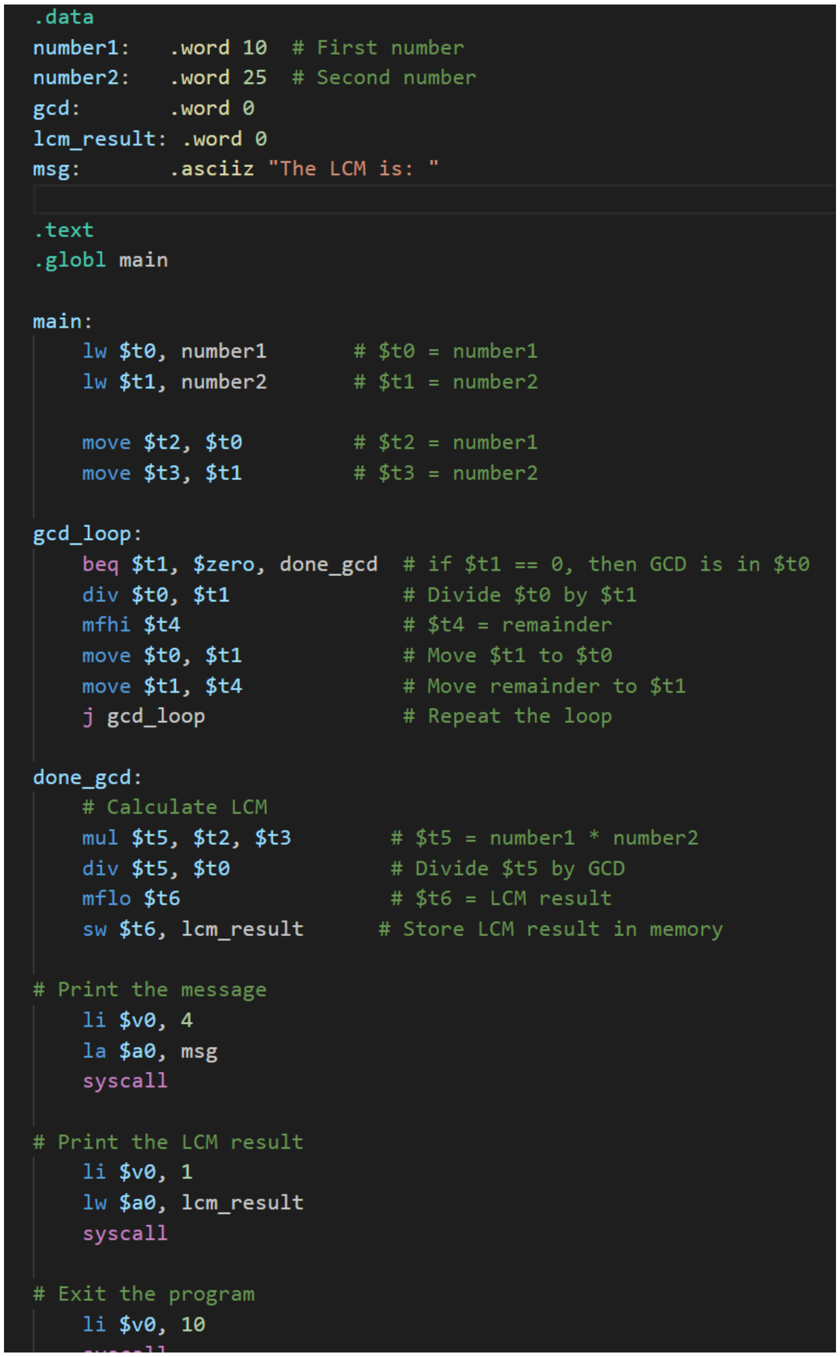
Output:



```
ments\Assignments_ES-215\Assignment3\q2.asm
MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar

The average is: 24
```

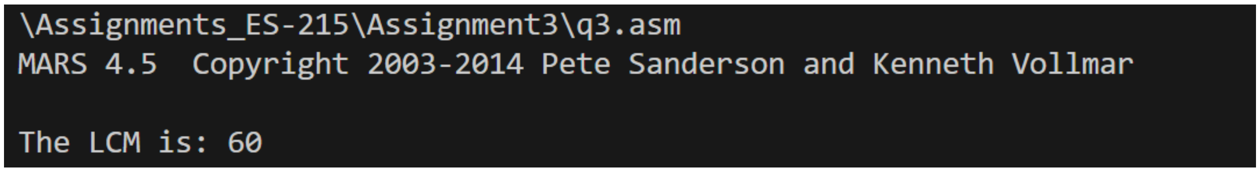
3. Write an assembly language program to find an LCM of two numbers stored at consecutive locations in memory.

Program:



```
.data
number1:    .word 10  # First number
number2:    .word 25  # Second number
gcd:        .word 0
lcm_result: .word 0
msg:        .asciiz "The LCM is: "

.text
.globl main

main:
    lw $t0, number1       # $t0 = number1
    lw $t1, number2       # $t1 = number2

    move $t2, $t0         # $t2 = number1
    move $t3, $t1         # $t3 = number2

gcd_loop:
    beq $t1, $zero, done_gcd  # if $t1 == 0, then GCD is in $t0
    div $t0, $t1              # Divide $t0 by $t1
    mfhi $t4                  # $t4 = remainder
    move $t0, $t1             # Move $t1 to $t0
    move $t1, $t4             # Move remainder to $t1
    j gcd_loop                # Repeat the loop

done_gcd:
    # Calculate LCM
    mul $t5, $t2, $t3        # $t5 = number1 * number2
    div $t5, $t0             # Divide $t5 by GCD
    mflo $t6                 # $t6 = LCM result
    sw $t6, lcm_result      # Store LCM result in memory

# Print the message
    li $v0, 4
    la $a0, msg
    syscall

# Print the LCM result
    li $v0, 1
    lw $a0, lcm_result
    syscall

# Exit the program
    li $v0, 10
```

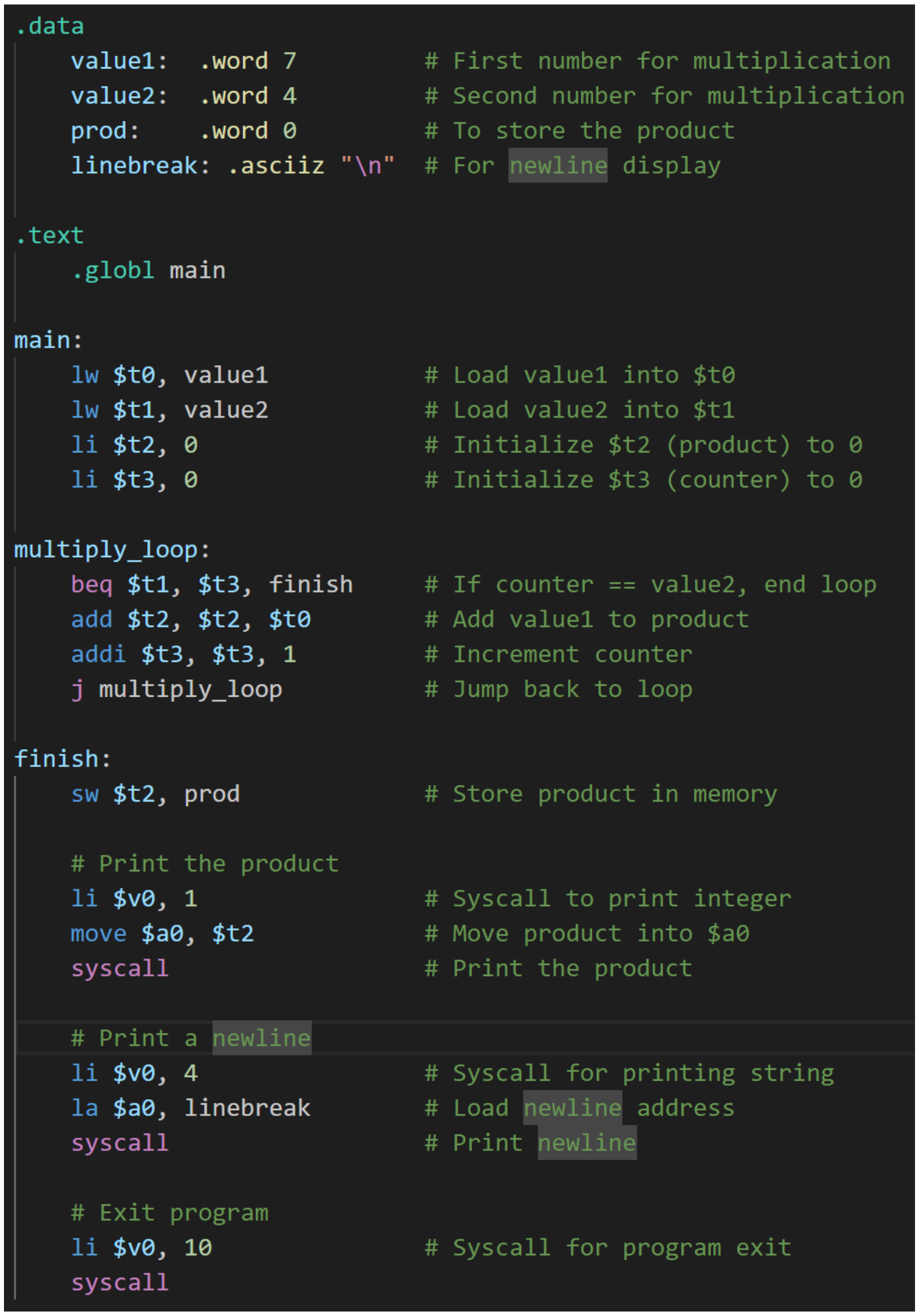
Output:



```
\Assignments_ES-215\Assignment3\q3.asm
MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar

The LCM is: 60
```

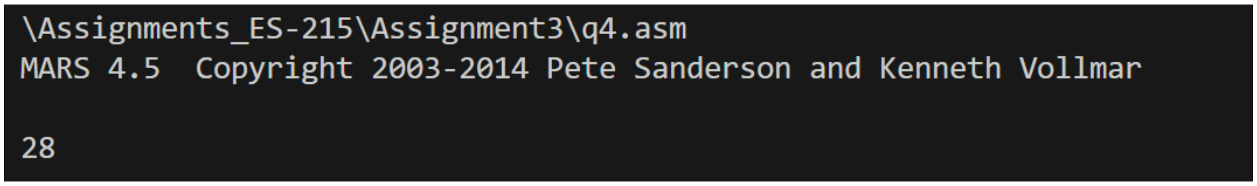
4. Write an assembly language program to calculate multiplication of two numbers without using MUL commands.

Program:



```
.data
    value1:   .word 7          # First number for multiplication
    value2:   .word 4          # Second number for multiplication
    prod:     .word 0          # To store the product
    linebreak: .asciiz "\n"    # For newline display

.text
    .globl main

main:
    lw $t0, value1             # Load value1 into $t0
    lw $t1, value2             # Load value2 into $t1
    li $t2, 0                  # Initialize $t2 (product) to 0
    li $t3, 0                  # Initialize $t3 (counter) to 0

multiply_loop:
    beq $t1, $t3, finish       # If counter == value2, end loop
    add $t2, $t2, $t0          # Add value1 to product
    addi $t3, $t3, 1           # Increment counter
    j multiply_loop            # Jump back to loop

finish:
    sw $t2, prod               # Store product in memory

    # Print the product
    li $v0, 1                  # Syscall to print integer
    move $a0, $t2              # Move product into $a0
    syscall                    # Print the product

    # Print a newline
    li $v0, 4                  # Syscall for printing string
    la $a0, linebreak          # Load newline address
    syscall                    # Print newline

    # Exit program
    li $v0, 10                 # Syscall for program exit
    syscall
```

Output:

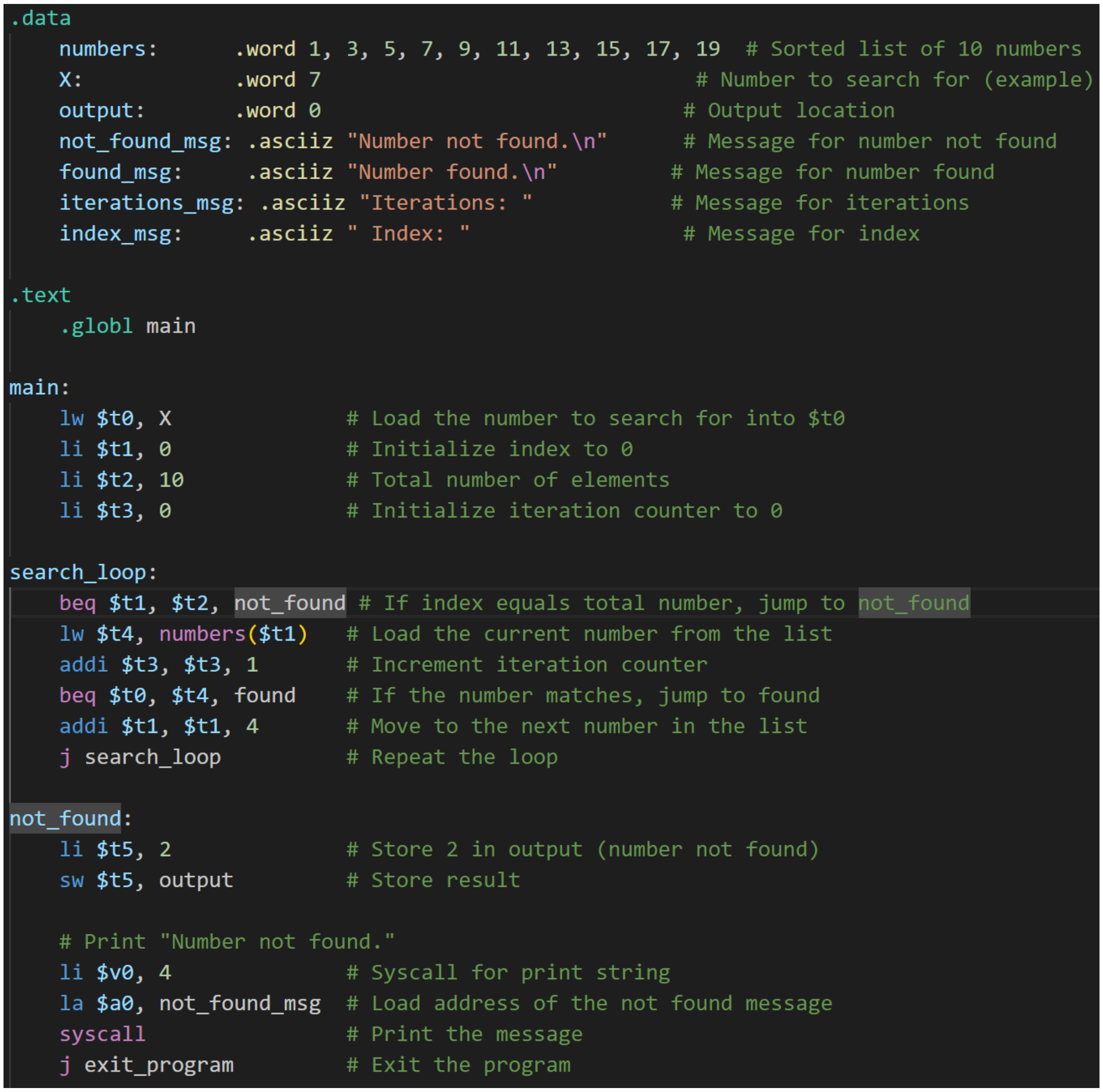


```
\Assignments_ES-215\Assignment3\q4.asm
MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar

28
```

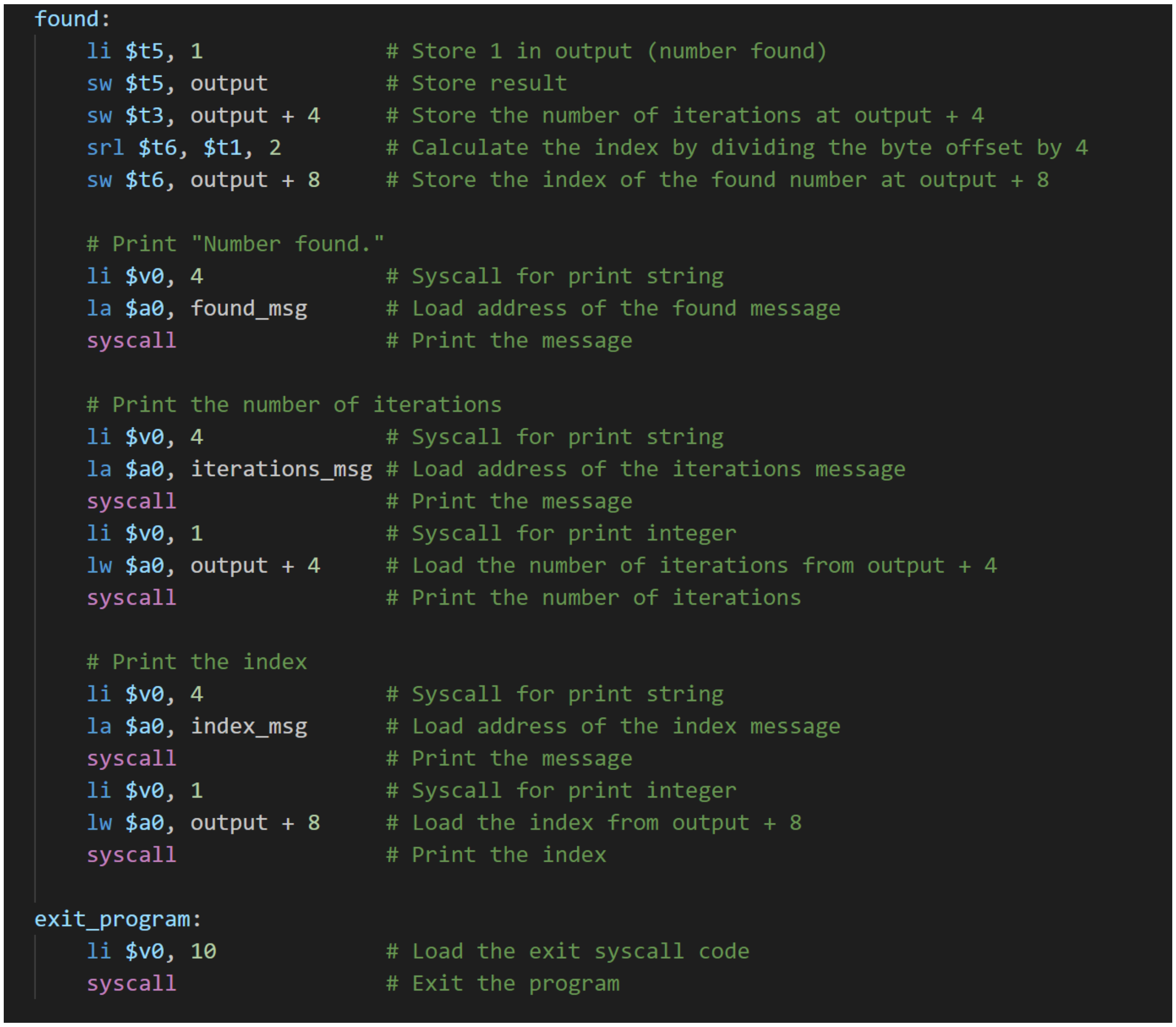
5. Write an assembly language program to find a given number in the list of 10 numbers (assuming the numbers are sorted). If found store 1 in output, else store 2 in output.The given number has been loaded from X location in memory, the

output has to be stored at the next location and if found store the number of iterations and the index of the element at the next consecutive locations.

Program:

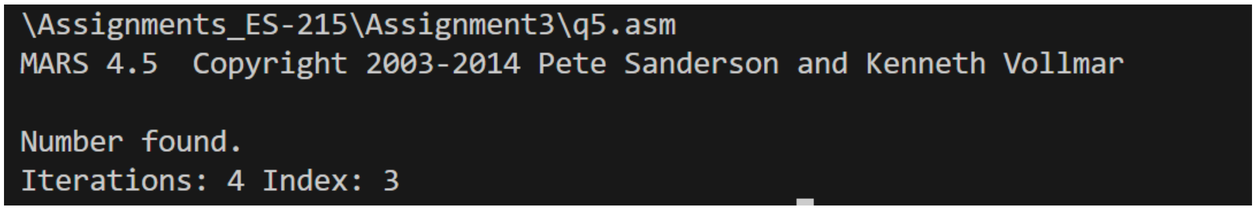


```
.data
    numbers:      .word 1, 3, 5, 7, 9, 11, 13, 15, 17, 19  # Sorted list of 10 numbers
    X:            .word 7                                # Number to search for (example)
    output:       .word 0                               # Output location
    not_found_msg: .asciiz "Number not found.\n"        # Message for number not found
    found_msg:     .asciiz "Number found.\n"           # Message for number found
    iterations_msg: .asciiz "Iterations: "             # Message for iterations
    index_msg:     .asciiz " Index: "                   # Message for index

.text
    .globl main

main:
    lw $t0, X              # Load the number to search for into $t0
    li $t1, 0              # Initialize index to 0
    li $t2, 10             # Total number of elements
    li $t3, 0              # Initialize iteration counter to 0

search_loop:
    beq $t1, $t2, not_found # If index equals total number, jump to not_found
    lw $t4, numbers($t1)   # Load the current number from the list
    addi $t3, $t3, 1       # Increment iteration counter
    beq $t0, $t4, found    # If the number matches, jump to found
    addi $t1, $t1, 4       # Move to the next number in the list
    j search_loop          # Repeat the loop

not_found:
    li $t5, 2              # Store 2 in output (number not found)
    sw $t5, output         # Store result

    # Print "Number not found."
    li $v0, 4              # Syscall for print string
    la $a0, not_found_msg  # Load address of the not found message
    syscall                # Print the message
    j exit_program         # Exit the program
```



```
found:
    li $t5, 1               # Store 1 in output (number found)
    sw $t5, output          # Store result
    sw $t3, output + 4      # Store the number of iterations at output + 4
    srl $t6, $t1, 2         # Calculate the index by dividing the byte offset by 4
    sw $t6, output + 8      # Store the index of the found number at output + 8

    # Print "Number found."
    li $v0, 4               # Syscall for print string
    la $a0, found_msg       # Load address of the found message
    syscall                 # Print the message

    # Print the number of iterations
    li $v0, 4               # Syscall for print string
    la $a0, iterations_msg # Load address of the iterations message
    syscall                 # Print the message
    li $v0, 1               # Syscall for print integer
    lw $a0, output + 4      # Load the number of iterations from output + 4
    syscall                 # Print the number of iterations

    # Print the index
    li $v0, 4               # Syscall for print string
    la $a0, index_msg       # Load address of the index message
    syscall                 # Print the message
    li $v0, 1               # Syscall for print integer
    lw $a0, output + 8      # Load the index from output + 8
    syscall                 # Print the index

exit_program:
    li $v0, 10              # Load the exit syscall code
    syscall                 # Exit the program
```

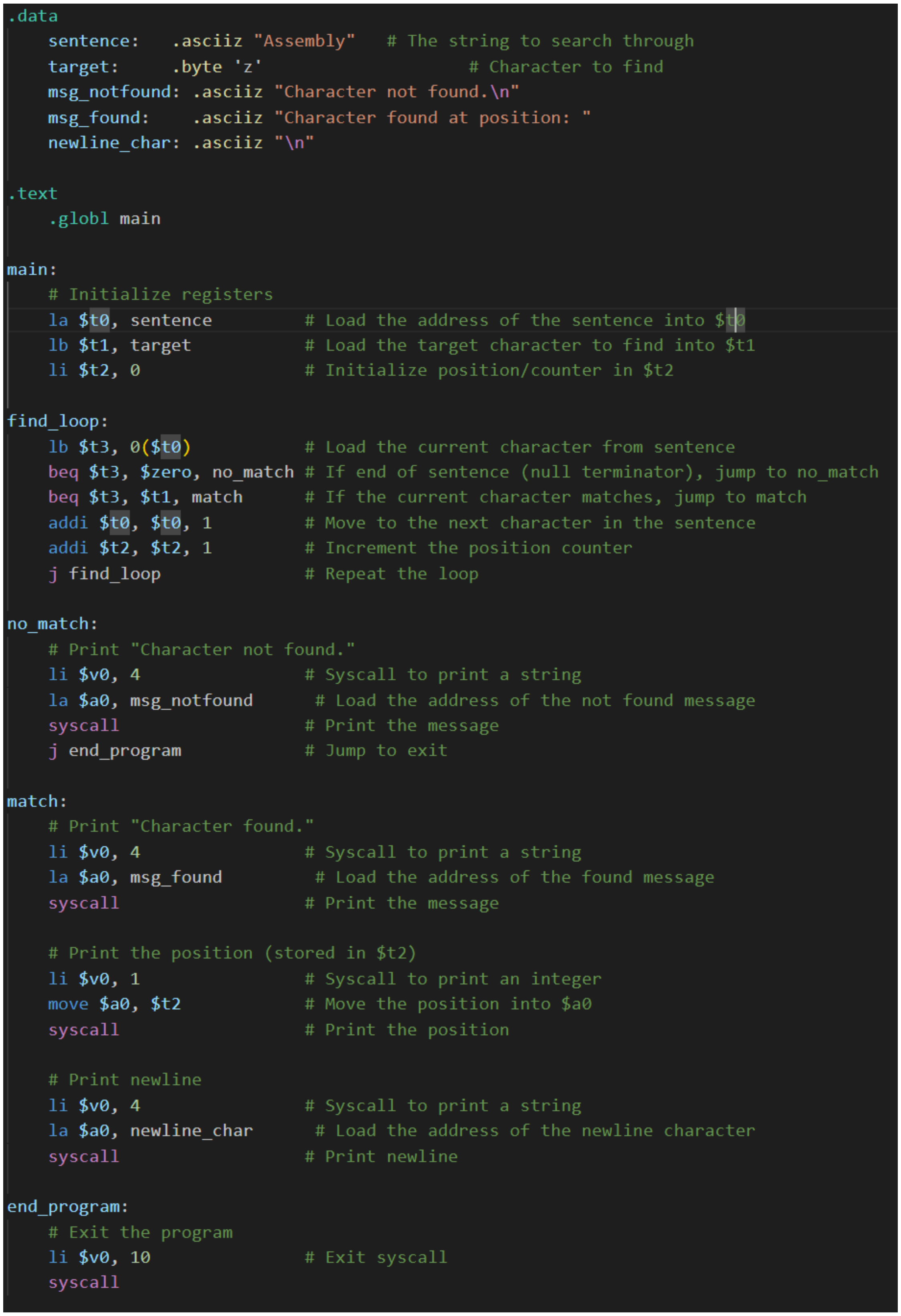
Output:



```
\Assignments_ES-215\Assignment3\q5.asm
MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar

Number found.
Iterations: 4 Index: 3
```

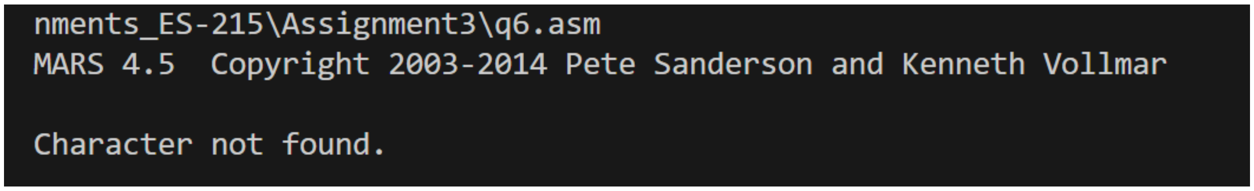
6. Write an assembly language program to find a character in a string.

Program:



```
.data
    sentence:   .asciiz "Assembly"   # The string to search through
    target:     .byte 'z'                    # Character to find
    msg_notfound: .asciiz "Character not found.\n"
    msg_found:    .asciiz "Character found at position: "
    newline_char: .asciiz "\n"

.text
    .globl main

main:
    # Initialize registers
    la $t0, sentence        # Load the address of the sentence into $t0
    lb $t1, target          # Load the target character to find into $t1
    li $t2, 0               # Initialize position/counter in $t2

find_loop:
    lb $t3, 0($t0)          # Load the current character from sentence
    beq $t3, $zero, no_match # If end of sentence (null terminator), jump to no_match
    beq $t3, $t1, match     # If the current character matches, jump to match
    addi $t0, $t0, 1        # Move to the next character in the sentence
    addi $t2, $t2, 1        # Increment the position counter
    j find_loop             # Repeat the loop

no_match:
    # Print "Character not found."
    li $v0, 4               # Syscall to print a string
    la $a0, msg_notfound     # Load the address of the not found message
    syscall                 # Print the message
    j end_program           # Jump to exit

match:
    # Print "Character found."
    li $v0, 4               # Syscall to print a string
    la $a0, msg_found        # Load the address of the found message
    syscall                 # Print the message

    # Print the position (stored in $t2)
    li $v0, 1               # Syscall to print an integer
    move $a0, $t2           # Move the position into $a0
    syscall                 # Print the position

    # Print newline
    li $v0, 4               # Syscall to print a string
    la $a0, newline_char     # Load the address of the newline character
    syscall                 # Print newline

end_program:
    # Exit the program
    li $v0, 10              # Exit syscall
    syscall
```

Output:



```
nments_ES-215\Assignment3\q6.asm
MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar

Character not found.
```