Computer Organisation and Architecture (ES 215)

Assignment - 3

Name: Pavan Deekshith Doddi
Roll no: 22110190

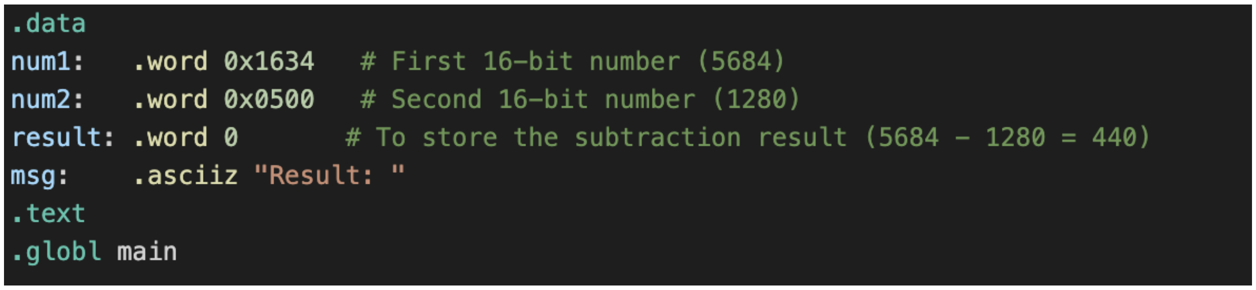
Question-1: Write a program in assembly language to subtract two 16 bit numbers without using the subtraction instruction. Note: the numbers have to be fetched from the memory.

Memory Locations:
The 16-bit numbers, referred to as `num1` and `num2`, are stored in memory as 32-bit words and are accessed using `lw` (load word) instructions.

Number Representation:
Both numbers are treated as 16-bit integers, and subtraction is performed using two's complement arithmetic by negating num2 and adding it to num1.

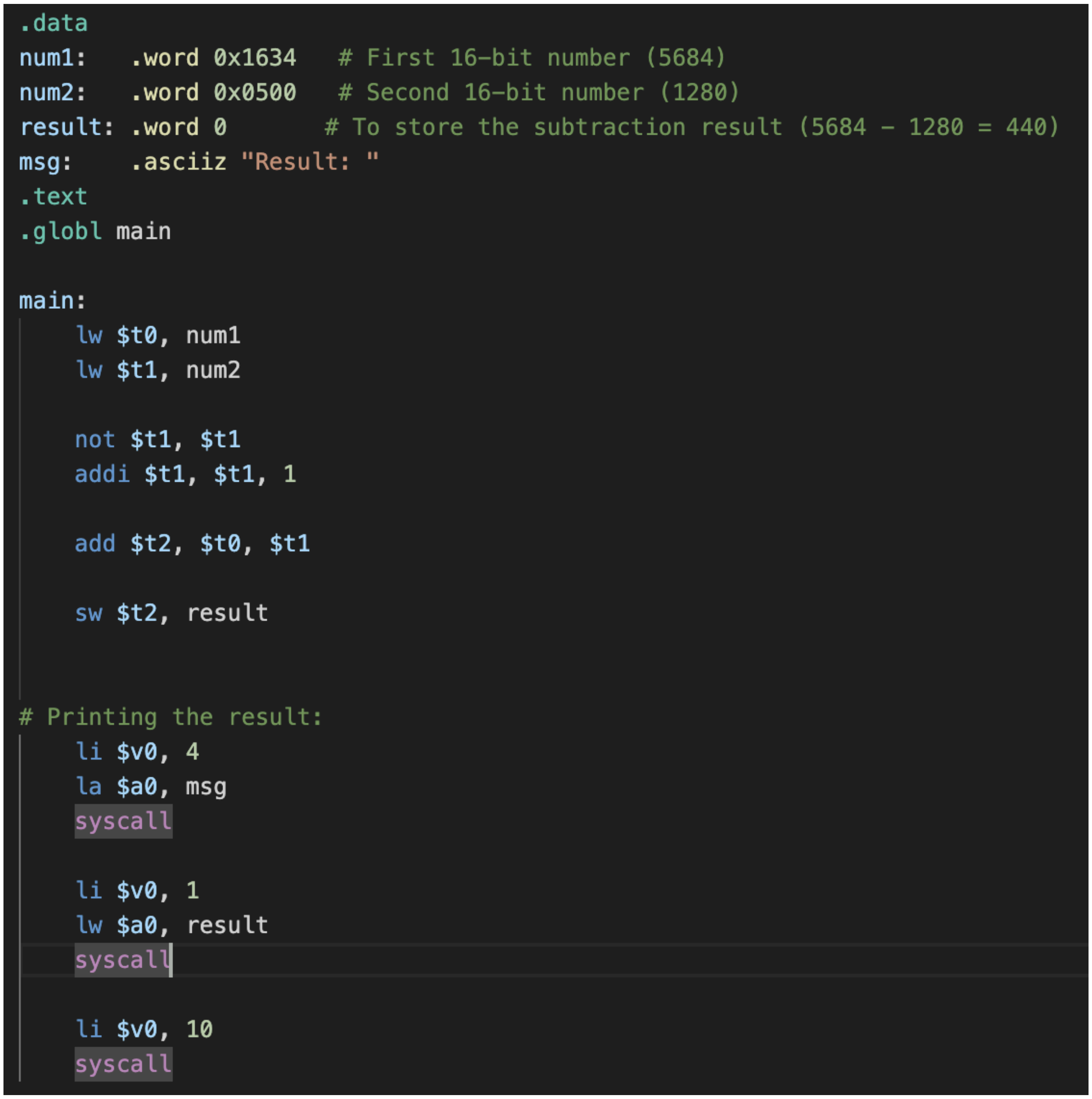
Result Storage:
The result of the subtraction is stored in a memory location labeled result.

Example Values: For this example, `num1` is set to `0x1634` (which is 5684 in decimal) and `num2` is `0x0500` (which is 1280 in decimal), though any values can be used.



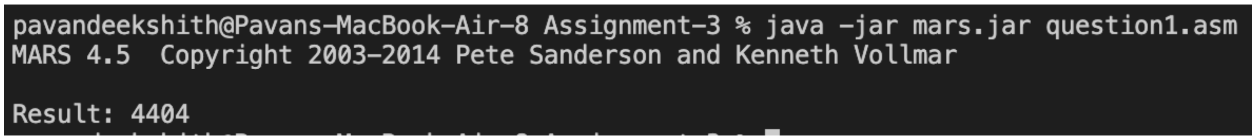
```
.data
num1:   .word 0x1634   # First 16-bit number (5684)
num2:   .word 0x0500   # Second 16-bit number (1280)
result: .word 0       # To store the subtraction result (5684 - 1280 = 440)
msg:    .asciiz "Result: "
.text
.globl main
```

Code:



```
.data
num1:    .word 0x1634    # First 16-bit number (5684)
num2:    .word 0x0500    # Second 16-bit number (1280)
result: .word 0        # To store the subtraction result (5684 - 1280 = 440)
msg:     .asciiz "Result: "
.text
.globl main

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

    not $t1, $t1
    addi $t1, $t1, 1

    add $t2, $t0, $t1

    sw $t2, result


# Printing the result:
    li $v0, 4
    la $a0, msg
    syscall

    li $v0, 1
    lw $a0, result
    syscall

    li $v0, 10
    syscall
```

Output:



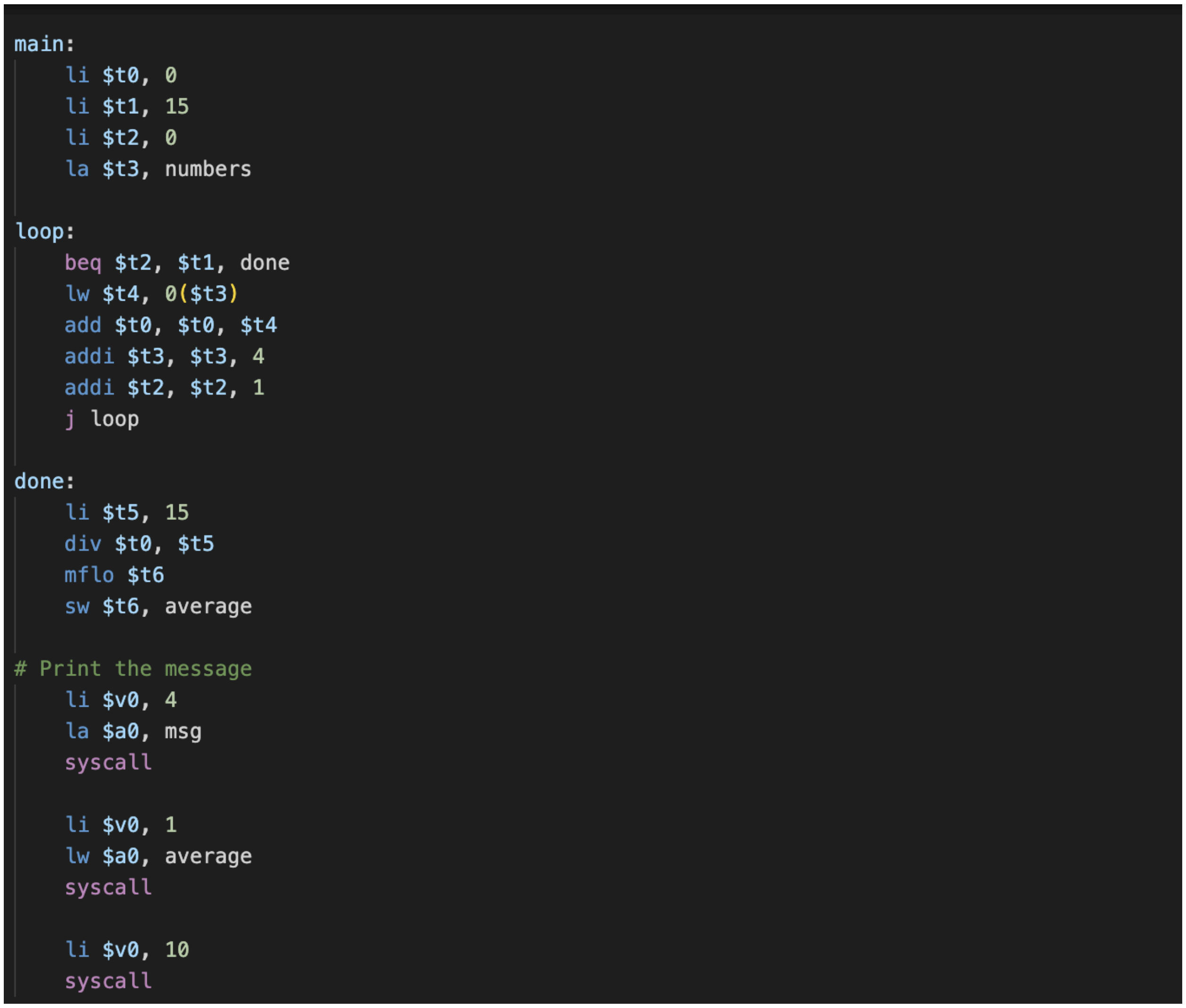
```
pavandeekshith@Pavans-MacBook-Air-8 Assignment-3 % java -jar mars.jar question1.asm
MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar

Result: 4404
```

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

Memory Layout:
The 15 numbers are stored as 32-bit words in consecutive memory locations starting from the label `numbers`.

Loop Counter:
The loop iterates exactly 15 times, corresponding to the number of elements in the `numbers` array.

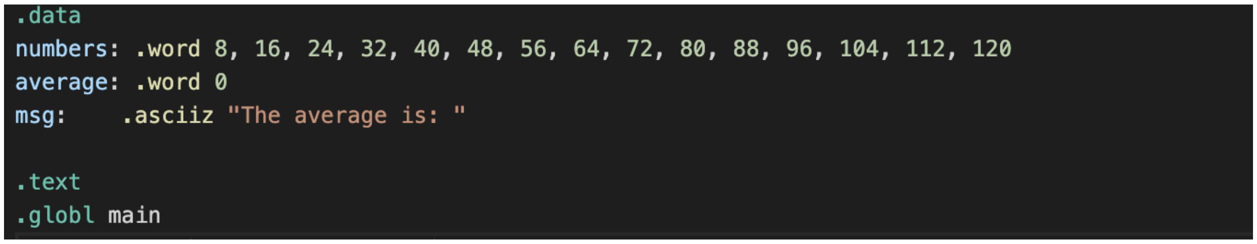
Code:



```
main:
    li $t0, 0
    li $t1, 15
    li $t2, 0
    la $t3, numbers

loop:
    beq $t2, $t1, done
    lw $t4, 0($t3)
    add $t0, $t0, $t4
    addi $t3, $t3, 4
    addi $t2, $t2, 1
    j loop

done:
    li $t5, 15
    div $t0, $t5
    mflo $t6
    sw $t6, average

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

    li $v0, 1
    lw $a0, average
    syscall

    li $v0, 10
    syscall
```

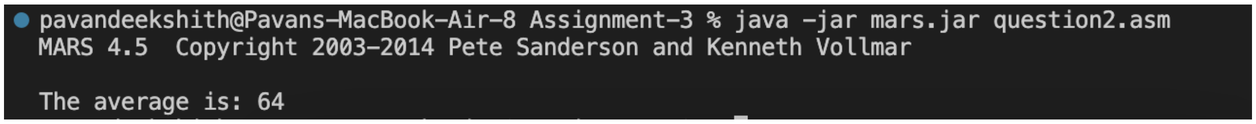
Result Storage:
The computed average is saved in a memory location labeled `average`, and it is treated as a 32-bit word.

Example Values:
The 15 numbers used are: 8, 16, 24, 32, 40, 48, 56, 64, 72, 80, 88, 96, 104, 112, 1 The average of these numbers is 64.



```
.data
numbers: .word 8, 16, 24, 32, 40, 48, 56, 64, 72, 80, 88, 96, 104, 112, 120
average: .word 0
msg:    .asciiz "The average is: "

.text
.globl main
```

Output:



```
pavandeekshith@Pavans-MacBook-Air-8 Assignment-3 % java -jar mars.jar question2.asm
MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar

The average is: 64
```

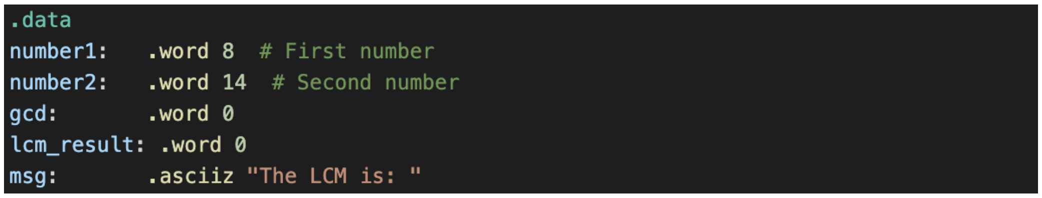
Question-3: Write an assembly language program to find an LCM of two numbers stored at consecutive locations in memory.

Memory Layout:
The two numbers for which the Least Common Multiple (LCM) is to be computed are stored in memory at locations labeled `number1` and `number2`, each occupying 32 bits.

Result Storage:
The calculated LCM is stored at the memory location labeled `lcm_result`, which is also a 32-bit word.

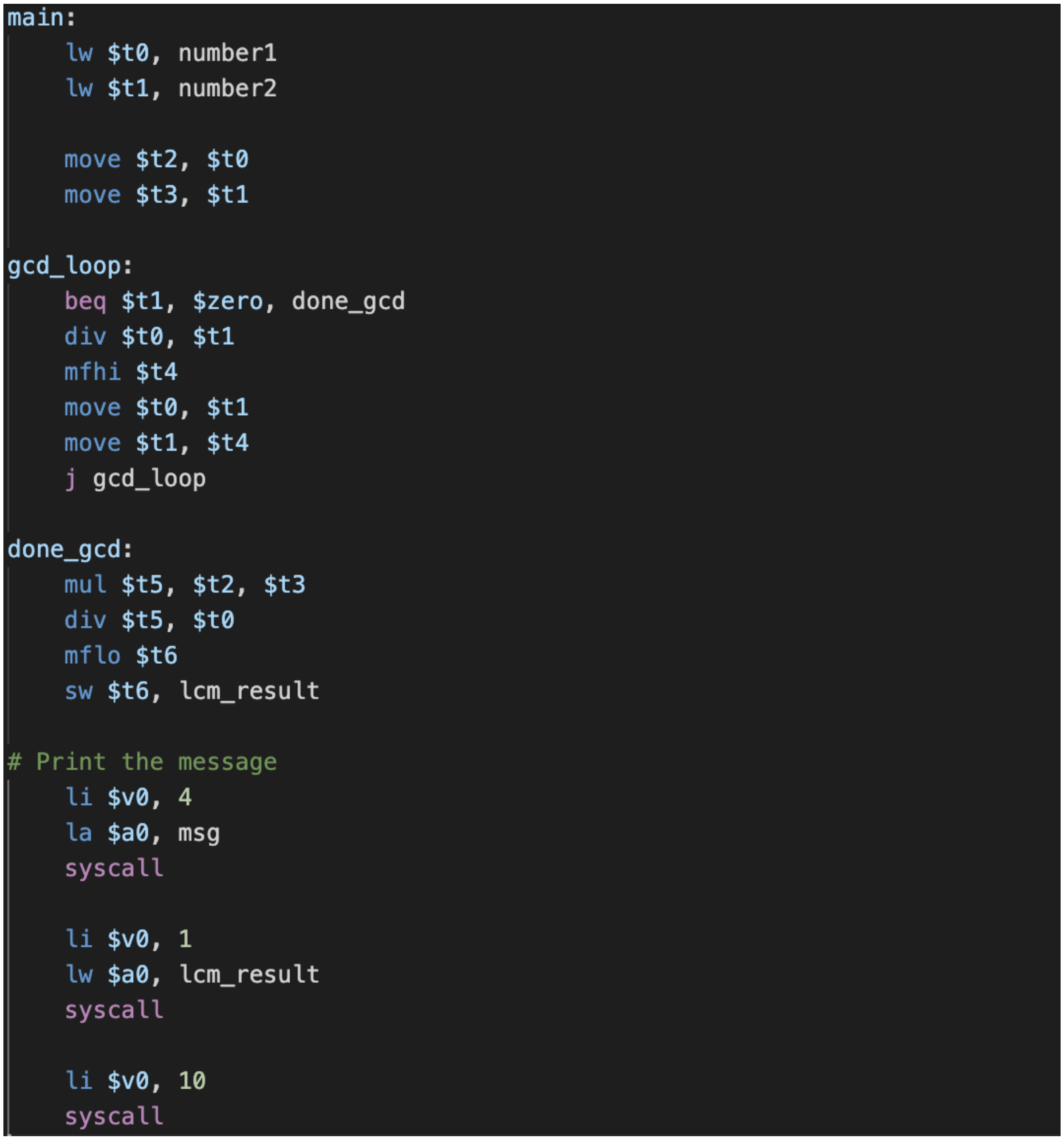
Formula used:
The LCM is calculated using the formula: LCM = (number1 * number2) / GCD, where GCD is determined using the Euclidean algorithm.

Assumptions:
The numbers chosen are 8 and 14, resulting in an LCM of 56.



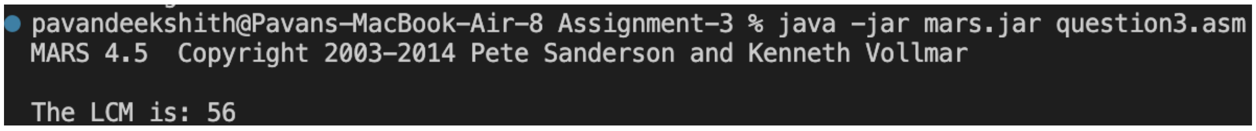
```
.data
number1:    .word 8  # First number
number2:    .word 14  # Second number
gcd:        .word 0
lcm_result: .word 0
msg:        .asciiz "The LCM is: "
```

Code:



```
main:
    lw $t0, number1
    lw $t1, number2

    move $t2, $t0
    move $t3, $t1

gcd_loop:
    beq $t1, $zero, done_gcd
    div $t0, $t1
    mfhi $t4
    move $t0, $t1
    move $t1, $t4
    j gcd_loop

done_gcd:
    mul $t5, $t2, $t3
    div $t5, $t0
    mflo $t6
    sw $t6, lcm_result

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

    li $v0, 1
    lw $a0, lcm_result
    syscall

    li $v0, 10
    syscall
```

Output:



```
pavandeekshith@Pavans-MacBook-Air-8 Assignment-3 % java -jar mars.jar question3.asm
MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar

The LCM is: 56
```

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

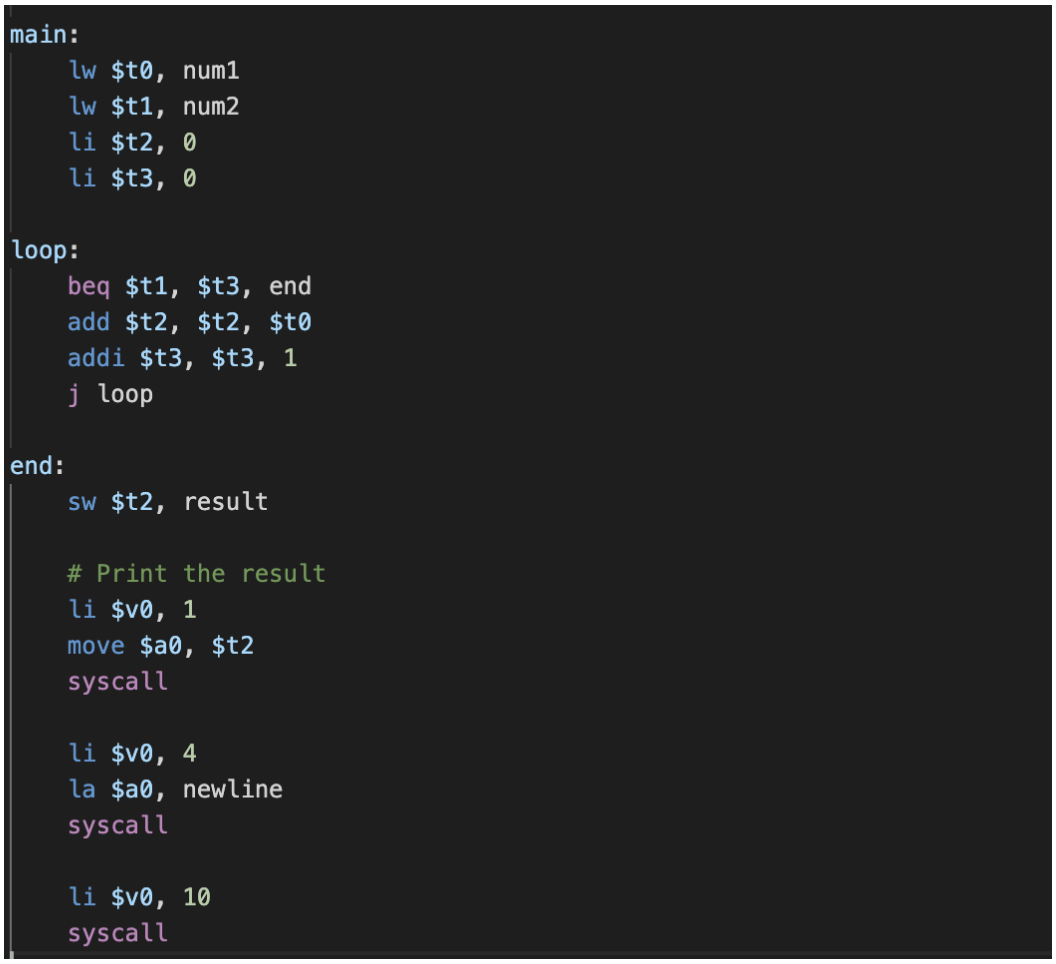
Memory Layout:
Two numbers are stored in consecutive memory locations labeled num1 and num2.

Result Storage:
The calculated result is stored in a memory location labeled product.

Formula used:
Multiplication is the same as repeated addition, I used the same principle.

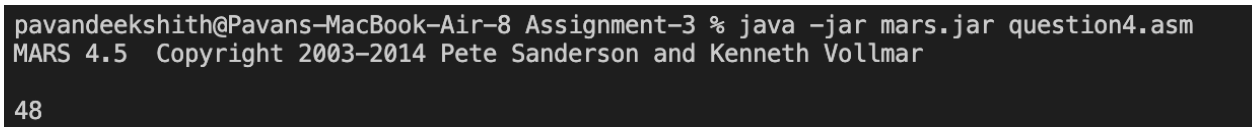
Assumptions:
For testing purposes, the numbers chosen are 4 and 12, whose product is 48.

Code:



```
main:
    lw $t0, num1
    lw $t1, num2
    li $t2, 0
    li $t3, 0

loop:
    beq $t1, $t3, end
    add $t2, $t2, $t0
    addi $t3, $t3, 1
    j loop

end:
    sw $t2, result

    # Print the result
    li $v0, 1
    move $a0, $t2
    syscall

    li $v0, 4
    la $a0, newline
    syscall

    li $v0, 10
    syscall
```

Output:



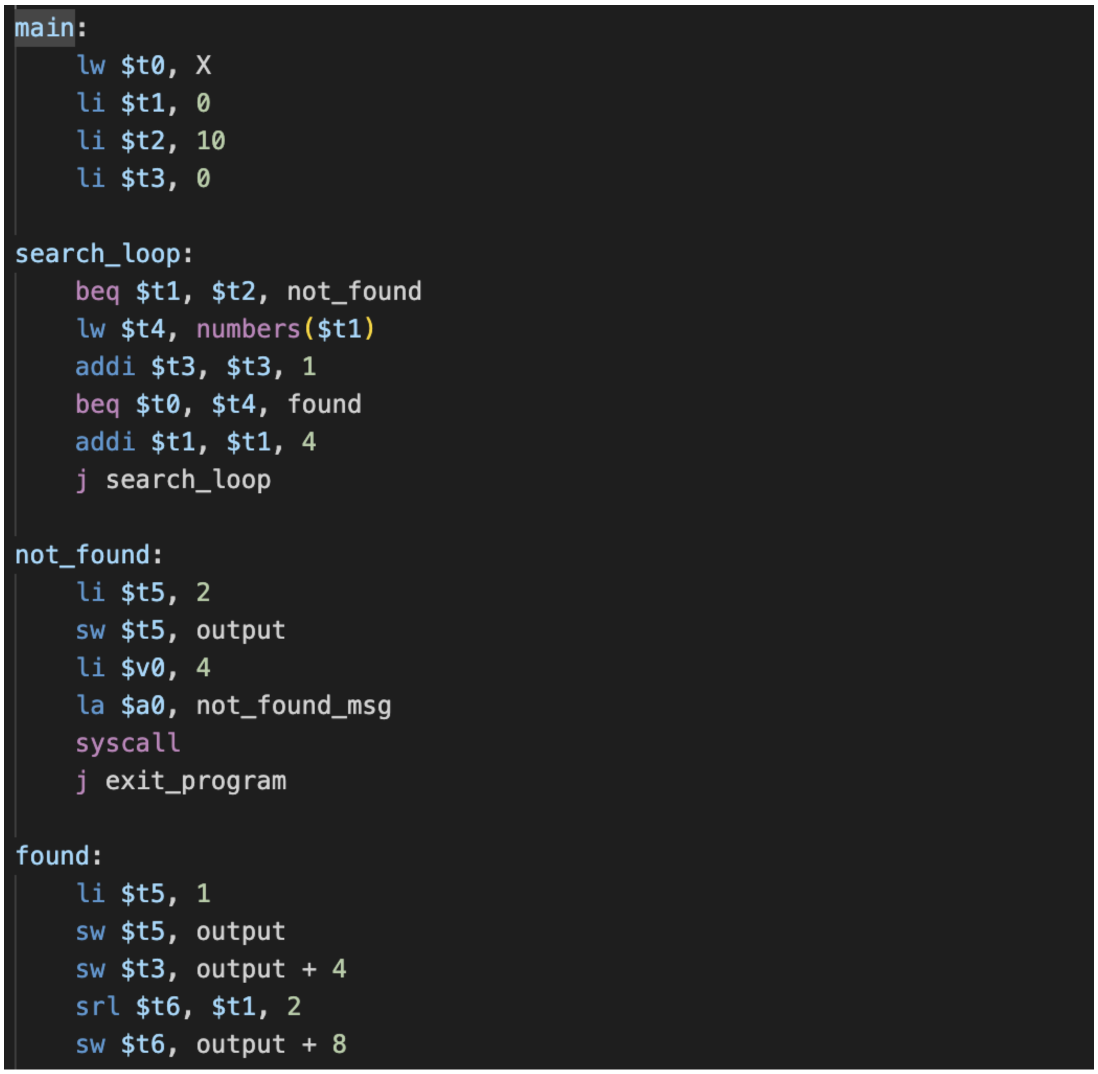
```
pavandeekshith@Pavans-MacBook-Air-8 Assignment-3 % java -jar mars.jar question4.asm
MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar

48
```

Question-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 at the next consecutive locations, if found.

Memory Setup:

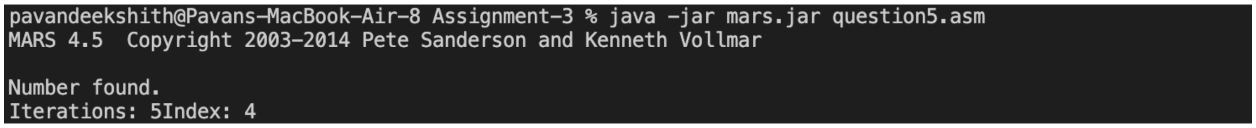
The array of 10 sorted numbers begins at the label `numbers`. The target number to search for is stored at the label `X`. The outcome of the search is recorded at `output`; a value of 1 indicates the number was found, while 2 means it was not found. The count of iterations performed is stored at `output + 4`. If the number is located, its index is stored at `output + 8`.

Code:



```
main:
    lw $t0, X
    li $t1, 0
    li $t2, 10
    li $t3, 0

search_loop:
    beq $t1, $t2, not_found
    lw $t4, numbers($t1)
    addi $t3, $t3, 1
    beq $t0, $t4, found
    addi $t1, $t1, 4
    j search_loop

not_found:
    li $t5, 2
    sw $t5, output
    li $v0, 4
    la $a0, not_found_msg
    syscall
    j exit_program

found:
    li $t5, 1
    sw $t5, output
    sw $t3, output + 4
    srl $t6, $t1, 2
    sw $t6, output + 8
```

Output:



```
pavandeekshith@Pavans-MacBook-Air-8 Assignment-3 % java -jar mars.jar question5.asm
MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar

Number found.
Iterations: 5Index: 4
```

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

Method Used:
String and Character: The string to be searched is located at the memory address labeled `string`, while the target character to find is stored at `char`.

Initialization: An index counter is initialized to zero.

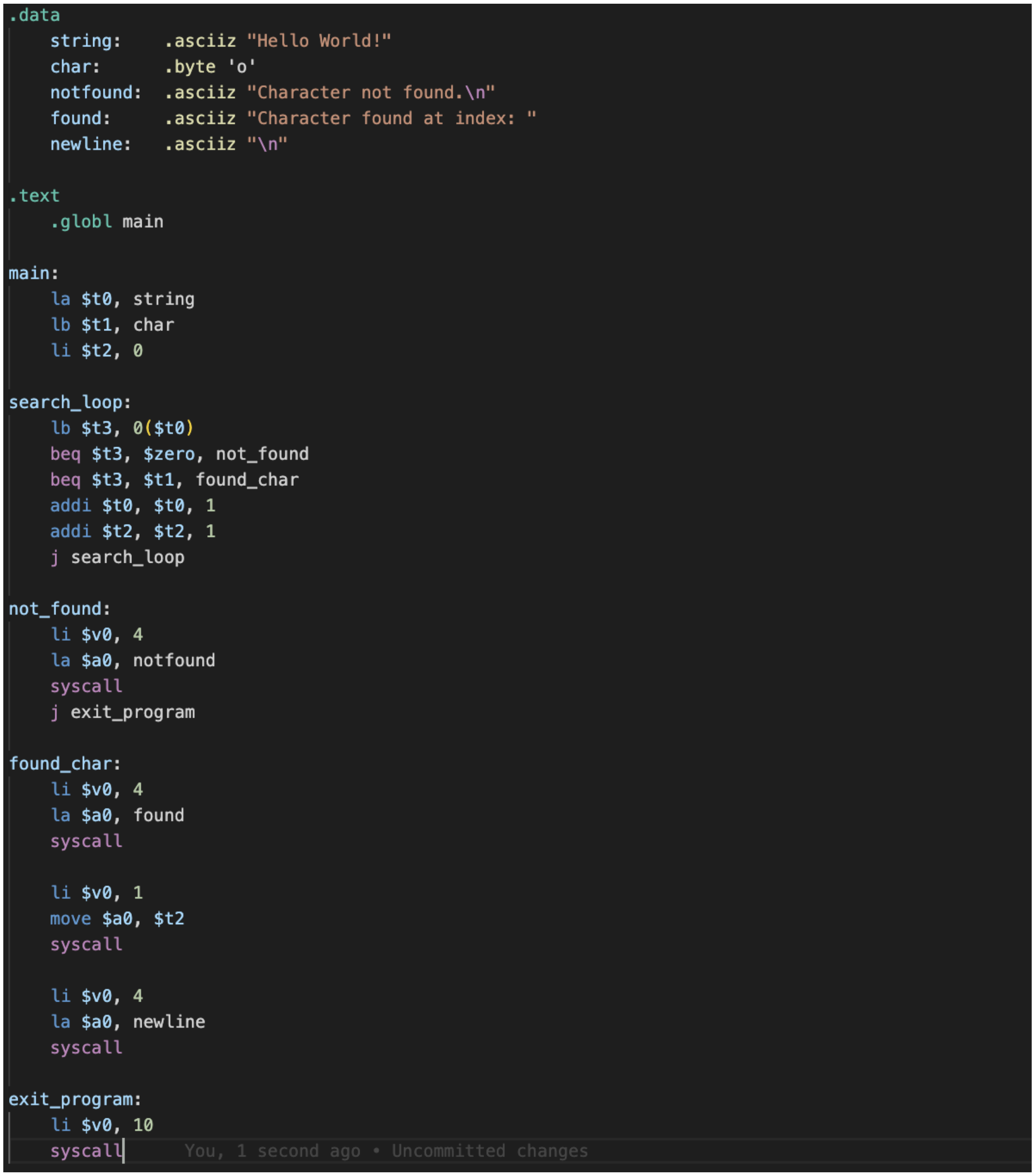
Character Search:
- Traverse each character in the string.
- Load the current character from the string.
- If the character matches the target, jump to the "found" section.
- If the end of the string (null terminator) is reached without finding the character, jump to the "not found" section.

Character Found: If the character is located, print a message indicating the index where it was found.

Character Not Found: If the character is not present after scanning the entire string, print a message indicating that the character was not found.

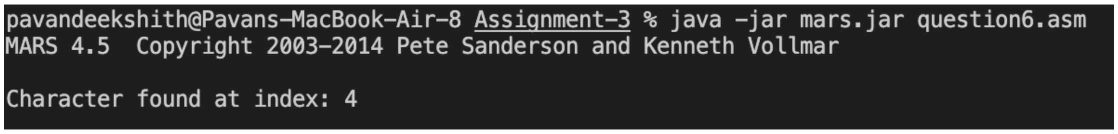
Assumptions:
The string is taken as “Hello World!” and the character to search for is “o”, which is at a position of 4.

Code:



```
.data
    string:    .asciiz "Hello World!"
    char:      .byte 'o'
    notfound:  .asciiz "Character not found.\n"
    found:     .asciiz "Character found at index: "
    newline:   .asciiz "\n"

.text
    .globl main

main:
    la $t0, string
    lb $t1, char
    li $t2, 0

search_loop:
    lb $t3, 0($t0)
    beq $t3, $zero, not_found
    beq $t3, $t1, found_char
    addi $t0, $t0, 1
    addi $t2, $t2, 1
    j search_loop

not_found:
    li $v0, 4
    la $a0, notfound
    syscall
    j exit_program

found_char:
    li $v0, 4
    la $a0, found
    syscall

    li $v0, 1
    move $a0, $t2
    syscall

    li $v0, 4
    la $a0, newline
    syscall

exit_program:
    li $v0, 10
    syscall      You, 1 second ago • Uncommitted changes
```

Output:



```
pavandeekshith@Pavans-MacBook-Air-8 Assignment-3 % java -jar mars.jar question6.asm
MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar

Character found at index: 4
```