Report of Assignment III
ES 215
Computer Organization and Architecture

Chandrabhan Patel
Roll No. 22110057

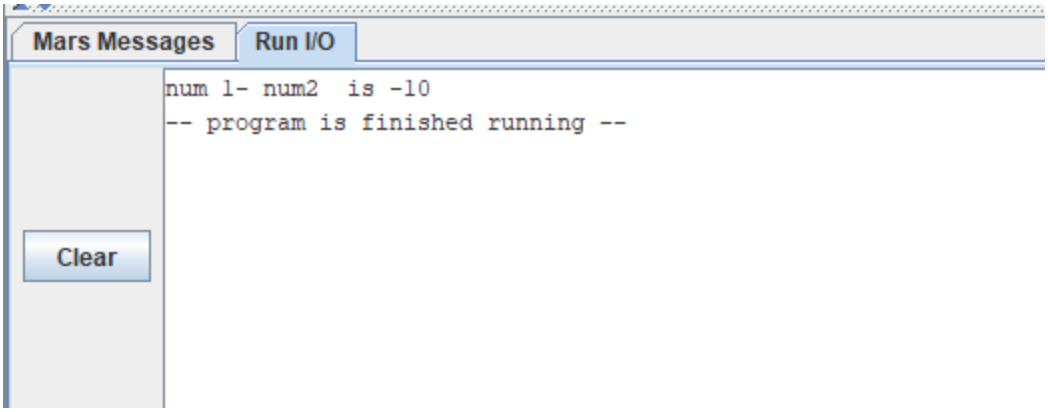
All codes are in assembly language for MIPS Processors.
I have used the [MARS MIPS simulator] to execute the code.

Q1.

Assumption: Instead of loading using lhw, I have used li, and loaded number in 16 bit range.
Also I have printed the value to the I/O

```
Unset
.data
    result: .asciiz "num 1- num2  is " # string for printing

.text

    li $t0, 10  #10
    li $t1, 20  # 20


    not $t2, $t1
    add $t2, $t2,1 # 2's complement for subtraction
    la $a0, result
    li $v0, 4 #syscall code 4 for printing string
    syscall

    add $t3, $t0, $t2
    move $a0, $t3 #storing answer in argument register.
    li $v0, 1 #syscall code 4 for printing string
    syscall


    li $v0, 10  #syscall code 10 for printing
    syscall
```

Output:



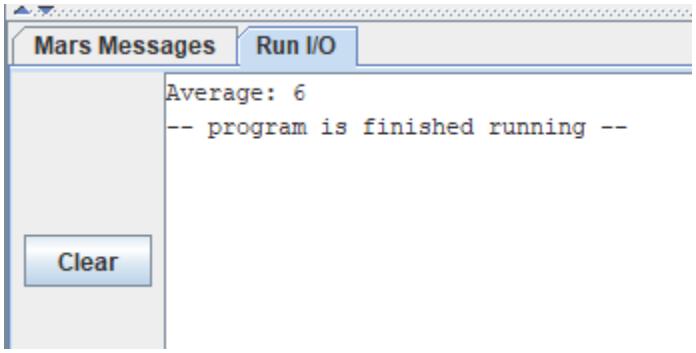
Q2.

Unset

```
.data
    numbers:    .word 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
    count:      .word 11
    result:     .asciiz "Average: "

.text
.globl main
main:
    la $t0, numbers      #load address of  numbers to reg. t0
    lw $t1, count         #load word count to  reg. t1

    li $t2, 0
    li $t3, 0            #loop counter

find_sum:
    lw $t4, 0($t0)        # indexing the array
    add $t2, $t2, $t4     # t2 is final sum
    addi $t0, $t0, 4      # index++
    addi $t3, $t3, 1      #loop counter++
    bne $t3, $t1, find_sum  # branch if not equal


    div $t2, $t1          # sum/count
    mflo $t5              # move quotient to $t5


    li $v0, 4
```

```
    la $a0, result
    syscall

    li $v0, 1
    move $a0, $t5
    syscall

    li $v0, 10
    syscall
```

Output:



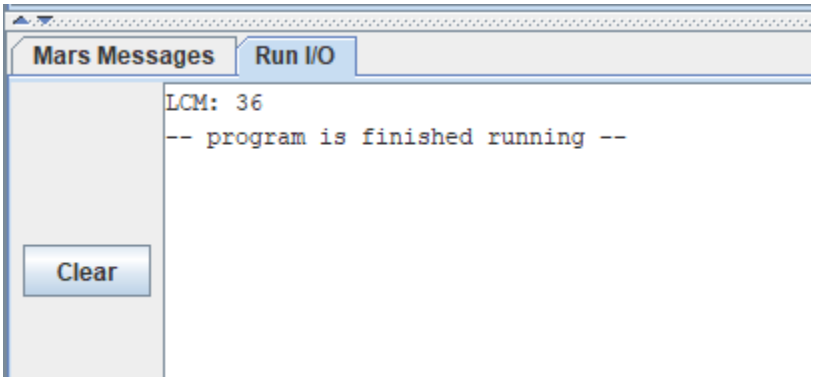
Q3.

Unset

```
.data
    num1:    .word 12
    num2:    .word 18
    msg:     .asciiz "LCM: "    # Message

.text
.globl main
main:

    lw $t0, num1
    lw $t1, num2
    move $t2, $t0        #copy of n1
    move $t3, $t1        # copy of n2

    # euclidian algo.
```

```
find_gcd:
    beq $t1, $zero, done_gcd  # branch if equal
    rem $t4, $t0, $t1        # remainder of $t0 / $t1
    move $t0, $t1           # divisor -> dividend
    move $t1, $t4            # remainder -> divisor
    j find_gcd

done_gcd:
    # Calculate LCM = (num1 * num2) / GCD
    mul $t5, $t2, $t3         # Multiply original num1 and num2
    move $t6, $t0             # $t6 now holds GCD
    div $t5, $t6              # Divide (num1 * num2) by GCD
    mflo $t7                  # Move quotient (LCM) to $t7

    li $v0, 4
    la $a0, msg
    syscall

    li $v0, 1
    move $a0, $t7
    syscall
#exit
    li $v0, 10
    syscall
```

Output:

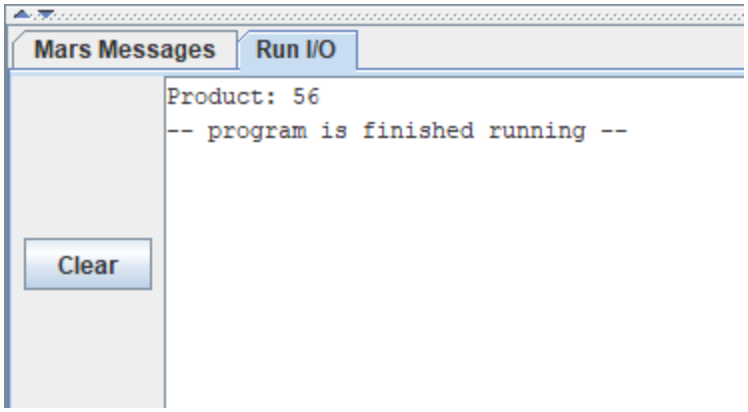


Q4.

Unset

```
.data
    num1:    .word 7
    num2:    .word 8
    msg:     .asciiz "Product: "

.text
.globl main
main:

    lw $t0, num1
    lw $t1, num2
    li $t2, 0            # result register

    #multiplication done by adding the n1, n2 times
multiply_loop:
    beq $t1, $zero, done_multiply   # If num2 is zero, exit loop
    add $t2, $t2, $t0              # adding n1 to result register
    subi $t1, $t1, 1               #  $t1
    j multiply_loop          # jump register

done_multiply:

    # Print "Product: "
    li $v0, 4
    la $a0, msg
    syscall

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

    # Exit
    li $v0, 10
    syscall
```

Output:



Q5.

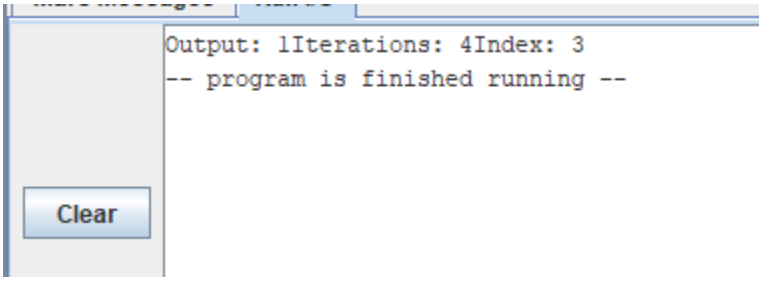
Unset

```
.data
    numbers:  .word 1, 3, 5, 7, 9, 11, 13, 15, 17, 19    # List of sorted
numbers
    target:   .word 7      # Target number to search
    output:   .word 0      # Output location (1 if found, 2 if not found)
    iterations: .word 0    # To store the number of iterations (if found)
    index:    .word 0      # To store the index of the found element
    msg_output: .asciiz "Output: "
    msg_iterations: .asciiz "Iterations: "
    msg_index: .asciiz "Index: "

.text
.globl main
main:
    # Initialize registers
    la $t0, numbers        # Load the base address of the numbers array into
$t0
    lw $t1, target         # Load the target number into $t1
    li $t2, 10             # Number of elements in the list
    li $t3, 0              # Initialize iteration counter
    li $t4, 2              # Assume output to be "2" (not found) initially

search_loop:
    beqz $t2, not_found    # If no more elements, go to not_found
    lw $t5, 0($t0)         # Load the current number from the list
    addi $t3, $t3, 1       # Increment iteration counter
    beq $t1, $t5, found    # If target equals current element, go to found
    addi $t0, $t0, 4       # Move to the next element in the array
    subi $t2, $t2, 1       # Decrement the number of remaining elements
    j search_loop          # Repeat the loop
```

```
found:
    li $t4, 1              # Store 1 in output if found
    subi $t6, $t3, 1       # Index = iterations - 1
    la $t7, output         # Load address of output
    sw $t4, 0($t7)         # Store 1 (found) in output location
    sw $t3, 4($t7)         # Store the number of iterations
    sw $t6, 8($t7)         # Store the index of the found element
    j print_result         # Jump to print the results

not_found:
    la $t7, output         # Load address of output
    sw $t4, 0($t7)         # Store 2 (not found) in output location
    j print_result         # Jump to print the results

# Printing the results
print_result:
    # Print "Output: "
    li $v0, 4              # System call for printing a string
    la $a0, msg_output      # Load address of the output message
    syscall

    # Print output value
    lw $a0, output          # Load the output value
    li $v0, 1               # System call for printing an integer
    syscall

    # Print "Iterations: "
    li $v0, 4               # System call for printing a string
    la $a0, msg_iterations  # Load address of the iterations message
    syscall

    # Print iterations value
    lw $a0, iterations      # Load the iterations value
    li $v0, 1               # System call for printing an integer
    syscall

    # Print "Index: "
    li $v0, 4               # System call for printing a string
    la $a0, msg_index       # Load address of the index message
    syscall

    # Print index value
    lw $a0, index           # Load the index value
```

```
    li $v0, 1               # System call for printing an integer
    syscall

exit:
    li $v0, 10              # System call for exit
    syscall
```

Output:



Given condition in the question:

It was asked to store the output and iterations to contiguous memory addresses, the assembler automatically stores them in contiguous address.

```
addiu $11,$0,0x00000000   18:    li $t3, 0          # Initialize iteration counter
addiu $12,$0,0x00000002   19:    li $t4, 2          # Assume output to be "2" (not found) initially
```

Q6.

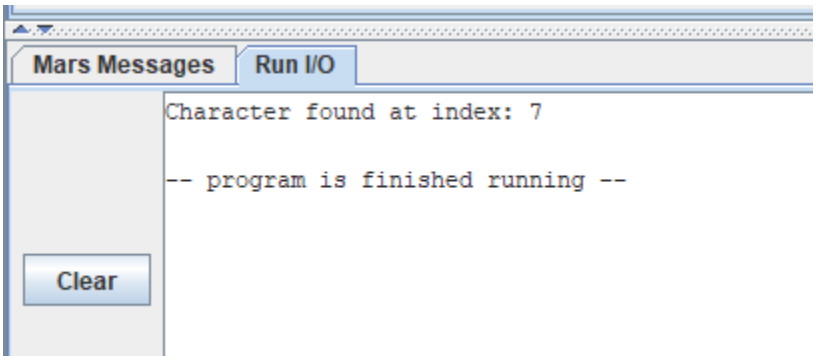
Unset

```
.data
    string:  .asciiz "Hello, MIPS World!"
    charToFind: .asciiz "M"                # Character to search for
    msg_found: .asciiz "Character found at index: "
    msg_not_found: .asciiz "Character not found"
    result_index: .word 0                  # To store the index of the found
character
    newline: .asciiz "\n"

.text
.globl main
main:

    la $t0, string
```

```
    la $t1, charToFind
    lb $t2, 0($t1)
    li $t3, 0
    li $t4, -1

search_loop:
    lb $t5, 0($t0)        # Load current character from the string
    beqz $t5, not_found   # If end of string (null terminator), go to not_found
    beq $t5, $t2, found   # If character matches, go to found
    addi $t0, $t0, 1      # Move to the next character in the string
    addi $t3, $t3, 1      # Increment index counter
    j search_loop         # Repeat the loop

found:
    li $t4, 0             # Store index in result index (found)
    sw $t3, result_index  # Store the found index
    j print_result        # Jump to print the result

not_found:
    li $t4, 1             # Character not found (set result index to -1)

print_result:
    # Print message based on the result
    beq $t4, 0, print_found  # If result index is 0, print found message
    li $v0, 4              # Print string syscall
    la $a0, msg_not_found  # Load address of "Character not found" message
    syscall
    j end_program

print_found:
    li $v0, 4              # Print string syscall
    la $a0, msg_found      # Load address of "Character found at index: "
message
    syscall

    # Print the index
    li $v0, 1              # Print integer syscall
    lw $a0, result_index   # Load the result index
    syscall

    # Print newline
    li $v0, 4              # Print string syscall
    la $a0, newline        # Load address of newline
    syscall
```

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

Output:



References:

Reference sheet for mips:

https://uweb.engr.arizona.edu/~ece369/Resources/spim/MIPSReference.pdf

Referred to LLM's for learning purposes as well.