Assignment 1

Bello Melido

03/15/2023

CSC 21000

Table of Contents:

Objective…………………………………………………………………….page 3

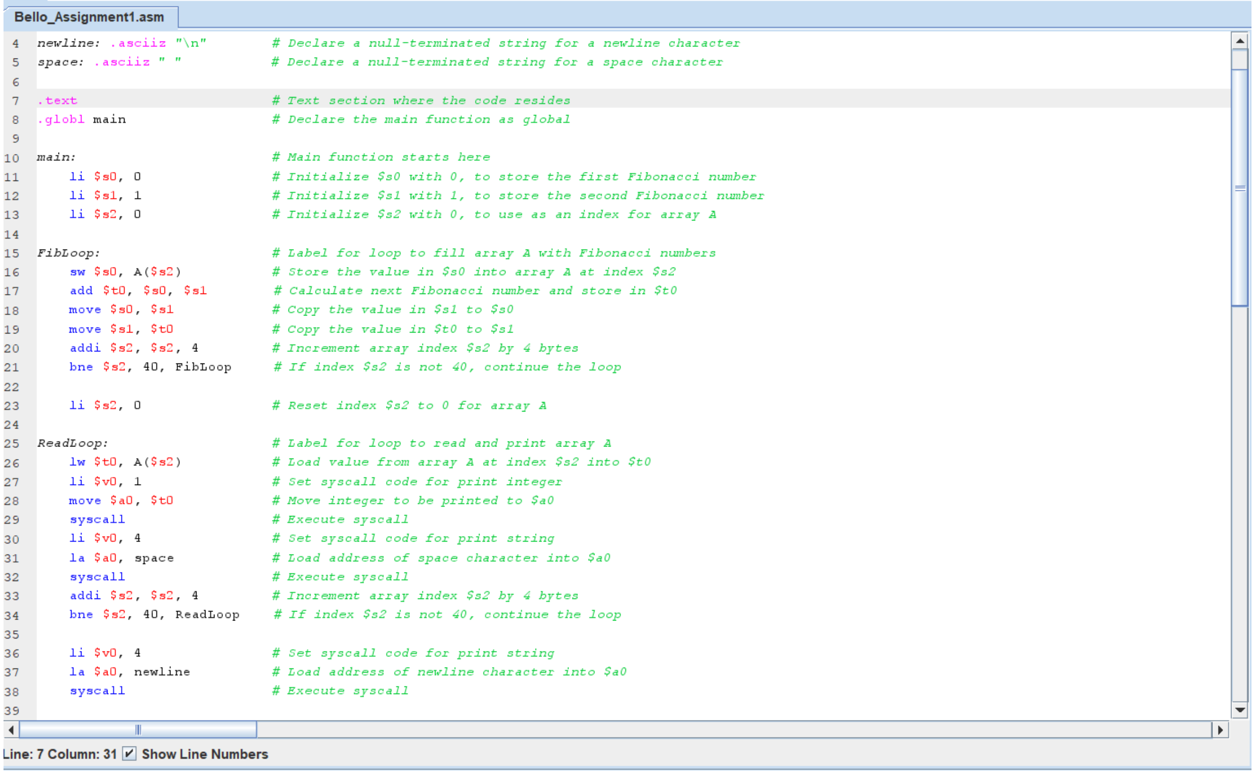
Code…………………………………………………………………………page 4-5

Screenshots and explanations………………………………………………………page 6-8

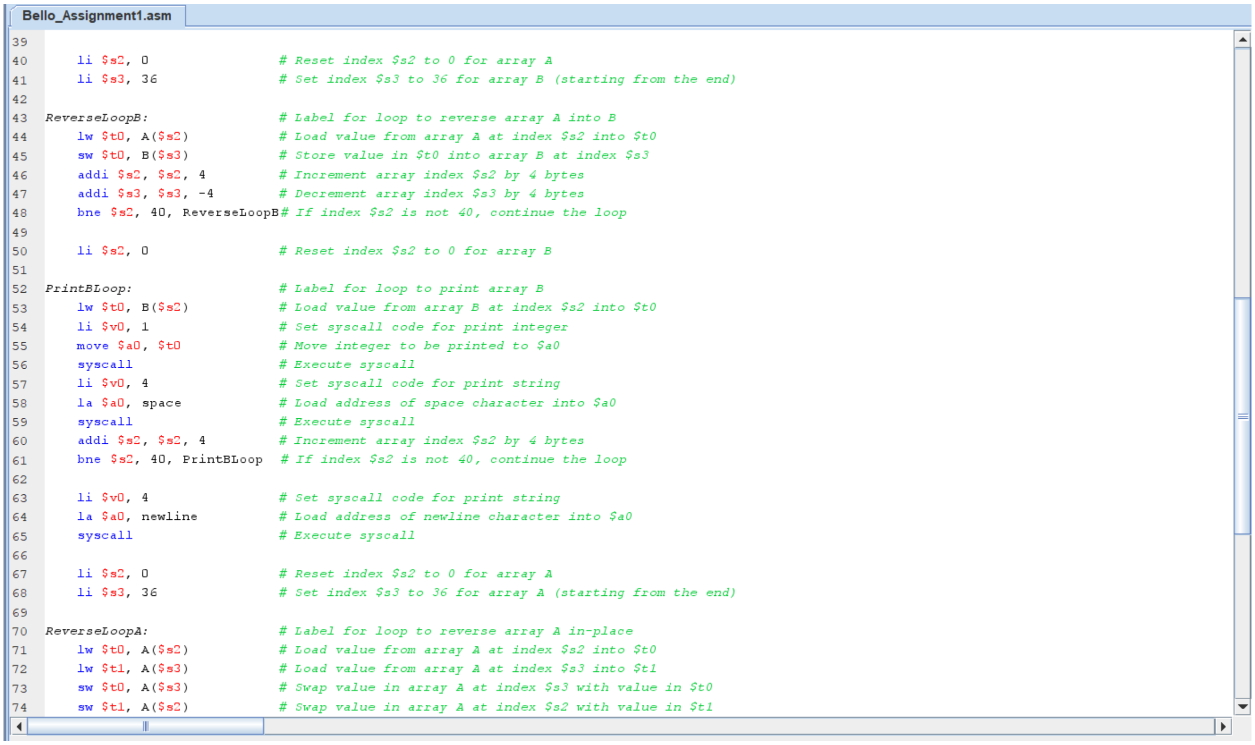
I. Objective

The objective of this assignment is to create a MIPS assembly program that performs specific tasks around data manipulation and memory handling. First, you will use the `LI` and `SW` instructions to initialize an array A[10] with the first 10 Fibonacci numbers. Second, the program will read this array from its original memory location. Third, you will write the elements of A[10] in reverse order to a different memory location and print the reversed array. Lastly, you will overwrite the original A[10] array with its elements in reverse order.

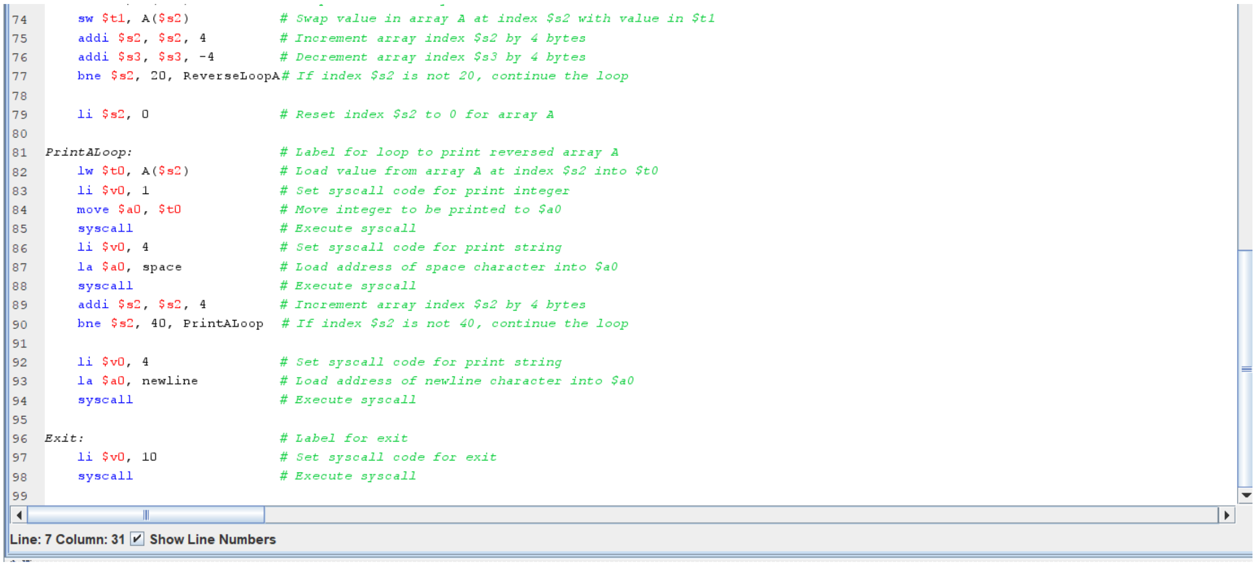
II. Code



```
newline: .asciiz "\n"        # Declare a null-terminated string for a newline character
space: .asciiz " "           # Declare a null-terminated string for a space character

.text                        # Text section where the code resides
.globl main                  # Declare the main function as global

main:                        # Main function starts here
    li $s0, 0                # Initialize $s0 with 0, to store the first Fibonacci number
    li $s1, 1                # Initialize $s1 with 1, to store the second Fibonacci number
    li $s2, 0                # Initialize $s2 with 0, to use as an index for array A

FibLoop:                     # Label for loop to fill array A with Fibonacci numbers
    sw $s0, A($s2)           # Store the value in $s0 into array A at index $s2
    add $t0, $s0, $s1         # Calculate next Fibonacci number and store in $t0
    move $s0, $s1            # Copy the value in $s1 to $s0
    move $s1, $t0            # Copy the value in $t0 to $s1
    addi $s2, $s2, 4         # Increment array index $s2 by 4 bytes
    bne $s2, 40, FibLoop      # If index $s2 is not 40, continue the loop

    li $s2, 0                # Reset index $s2 to 0 for array A

ReadLoop:                    # Label for loop to read and print array A
    lw $t0, A($s2)           # Load value from array A at index $s2 into $t0
    li $v0, 1                # Set syscall code for print integer
    move $a0, $t0            # Move integer to be printed to $a0
    syscall                  # Execute syscall
    li $v0, 4                # Set syscall code for print string
    la $a0, space            # Load address of space character into $a0
    syscall                  # Execute syscall
    addi $s2, $s2, 4         # Increment array index $s2 by 4 bytes
    bne $s2, 40, ReadLoop     # If index $s2 is not 40, continue the loop

    li $v0, 4                # Set syscall code for print string
    la $a0, newline          # Load address of newline character into $a0
    syscall                  # Execute syscall

```



```

    li $s2, 0                # Reset index $s2 to 0 for array A
    li $s3, 36               # Set index $s3 to 36 for array B (starting from the end)

ReverseLoopB:                # Label for loop to reverse array A into B
    lw $t0, A($s2)           # Load value from array A at index $s2 into $t0
    sw $t0, B($s3)           # Store value in $t0 into array B at index $s3
    addi $s2, $s2, 4         # Increment array index $s2 by 4 bytes
    addi $s3, $s3, -4        # Decrement array index $s3 by 4 bytes
    bne $s2, 40, ReverseLoopB# If index $s2 is not 40, continue the loop

    li $s2, 0                # Reset index $s2 to 0 for array B

PrintBLoop:                  # Label for loop to print array B
    lw $t0, B($s2)           # Load value from array B at index $s2 into $t0
    li $v0, 1                # Set syscall code for print integer
    move $a0, $t0            # Move integer to be printed to $a0
    syscall                  # Execute syscall
    li $v0, 4                # Set syscall code for print string
    la $a0, space            # Load address of space character into $a0
    syscall                  # Execute syscall
    addi $s2, $s2, 4         # Increment array index $s2 by 4 bytes
    bne $s2, 40, PrintBLoop   # If index $s2 is not 40, continue the loop

    li $v0, 4                # Set syscall code for print string
    la $a0, newline          # Load address of newline character into $a0
    syscall                  # Execute syscall

    li $s2, 0                # Reset index $s2 to 0 for array A
    li $s3, 36               # Set index $s3 to 36 for array A (starting from the end)

ReverseLoopA:                # Label for loop to reverse array A in-place
    lw $t0, A($s2)           # Load value from array A at index $s2 into $t0
    lw $t1, A($s3)           # Load value from array A at index $s3 into $t1
    sw $t0, A($s3)           # Swap value in array A at index $s3 with value in $t0
    sw $t1, A($s2)           # Swap value in array A at index $s2 with value in $t1
```

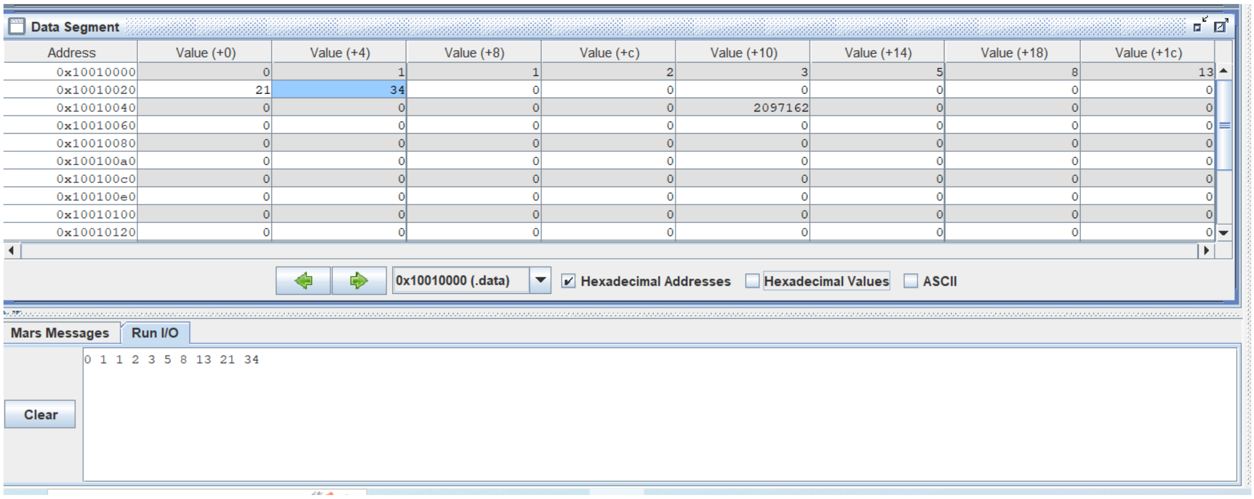


```
    sw $t1, A($s2)           # Swap value in array A at index $s2 with value in $t1
    addi $s2, $s2, 4         # Increment array index $s2 by 4 bytes
    addi $s3, $s3, -4        # Decrement array index $s3 by 4 bytes
    bne $s2, 20, ReverseLoopA# If index $s2 is not 20, continue the loop

    li $s2, 0                # Reset index $s2 to 0 for array A

PrintALoop:                  # Label for loop to print reversed array A
    lw $t0, A($s2)           # Load value from array A at index $s2 into $t0
    li $v0, 1                # Set syscall code for print integer
    move $a0, $t0            # Move integer to be printed to $a0
    syscall                  # Execute syscall
    li $v0, 4                # Set syscall code for print string
    la $a0, space            # Load address of space character into $a0
    syscall                  # Execute syscall
    addi $s2, $s2, 4         # Increment array index $s2 by 4 bytes
    bne $s2, 40, PrintALoop  # If index $s2 is not 40, continue the loop

    li $v0, 4                # Set syscall code for print string
    la $a0, newline          # Load address of newline character into $a0
    syscall                  # Execute syscall

Exit:                        # Label for exit
    li $v0, 10               # Set syscall code for exit
    syscall                  # Execute syscall

```

Line: 7 Column: 31 ☑ Show Line Numbers

III. Screenshots

Screenshot 1 - Write array A in different memory location in a reverse order.



Data Segment

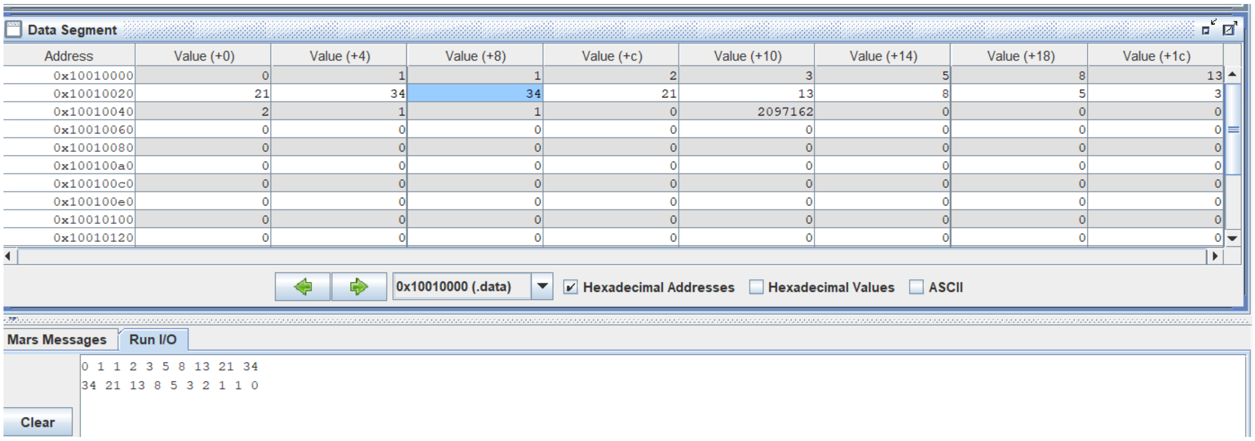
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	1	1	2	3	5	8	13
0x10010020	21	34	0	0	0	0	0	0
0x10010040	0	0	0	0	2097162	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

0x10010000 (.data) ☑ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

```
0 1 1 2 3 5 8 13 21 34
```

Explanation: In the Data Segment, 40 bytes are allocated for array 'A' starting at a base address, say '0x10010000'. During the 'ReadLoop', the program reads integers stored from this base address up to '0x10010024' in 4-byte increments. Each 'Value (+X)' shows a Fibonacci number corresponding to that memory location.

Screenshot 2 - Write array A in different memory location in a reverse order



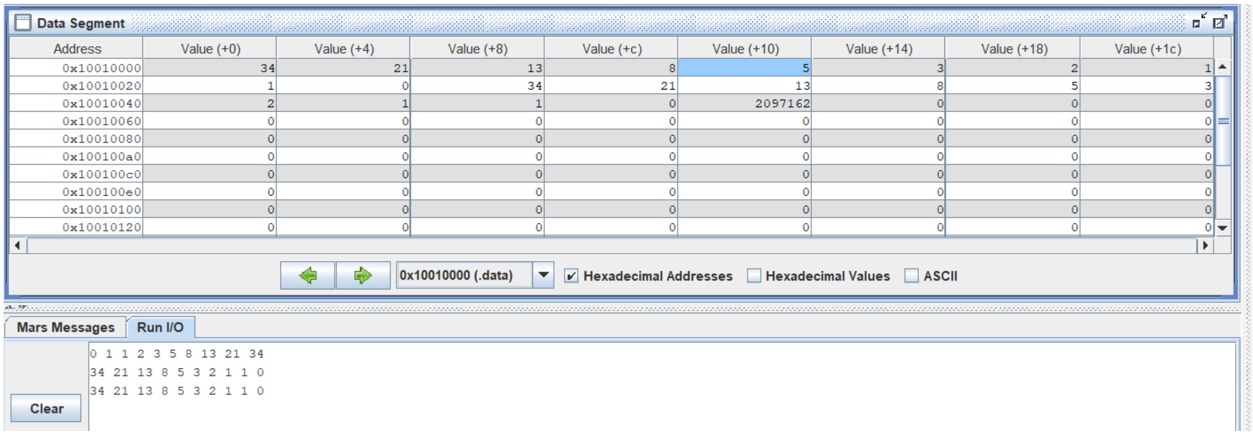
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	1	1	2	3	5	8	13
0x10010020	21	34	34	21	13	8	5	3
0x10010040	2	1	1	0	2097162	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

```
0 1 1 2 3 5 8 13 21 34
34 21 13 8 5 3 2 1 1 0
```

Explanation: Another 40 bytes are allocated for array 'B', possibly starting at another base address like '0x10010028'. The 'ReverseLoopB' reads from 'A' and writes these integers in reverse into 'B', filling memory addresses from '0x10010028' to '0x1001004C'. The Data Segment will show these reversed numbers at 'Value (+X)' corresponding to array 'B'.

Screenshot 3 - Write array A in different memory location in a reverse order



Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	34	21	13	8	5	3	2	1
0x10010020	1	0	34	21	13	8	5	3
0x10010040	2	1	1	0	2097162	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

```
0 1 1 2 3 5 8 13 21 34
34 21 13 8 5 3 2 1 1 0
34 21 13 8 5 3 2 1 1 0
```

Explanation: In the 'ReverseLoopA', the program swaps the elements of array 'A' in-place. For example, the Fibonacci numbers at addresses '0x10010000' and '0x10010024' will be swapped. This operation reverses the original array 'A' within its existing memory locations, ranging from '0x10010000' to '0x10010024'.