Homework 2 + swap

Bello Melido

09/17/2023

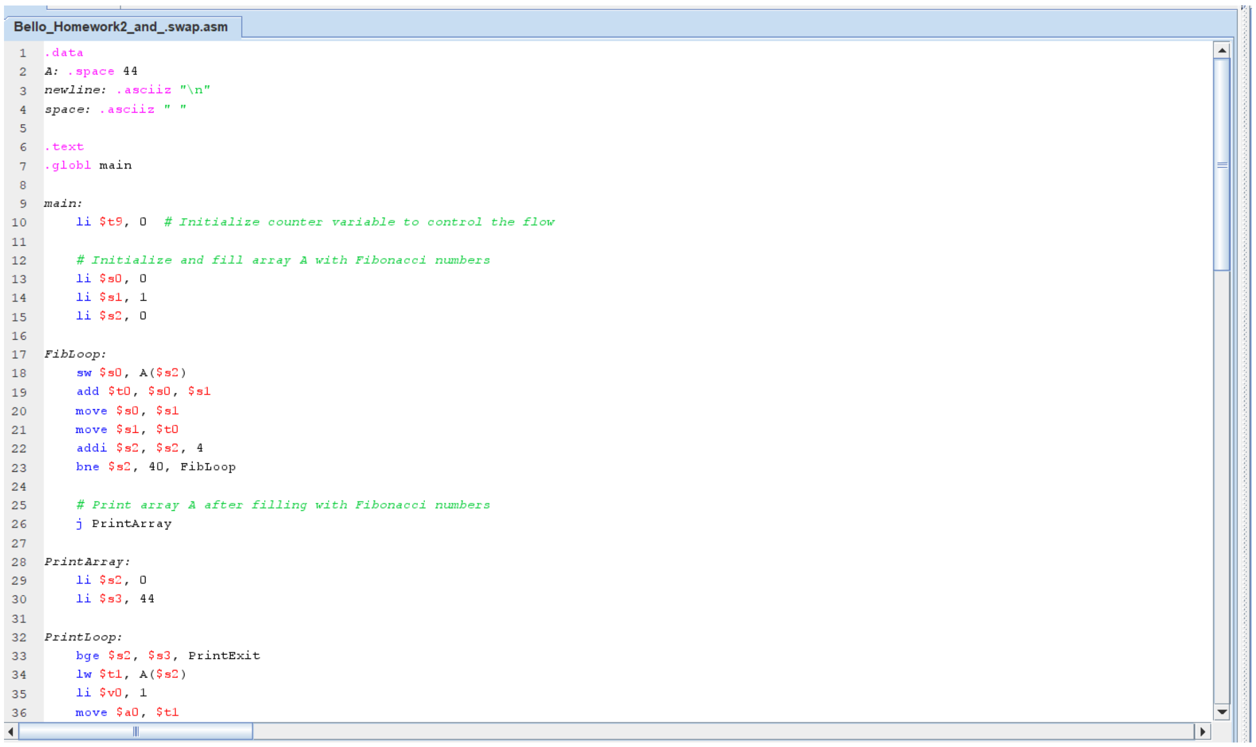
CSC 21000

Table of Contents:

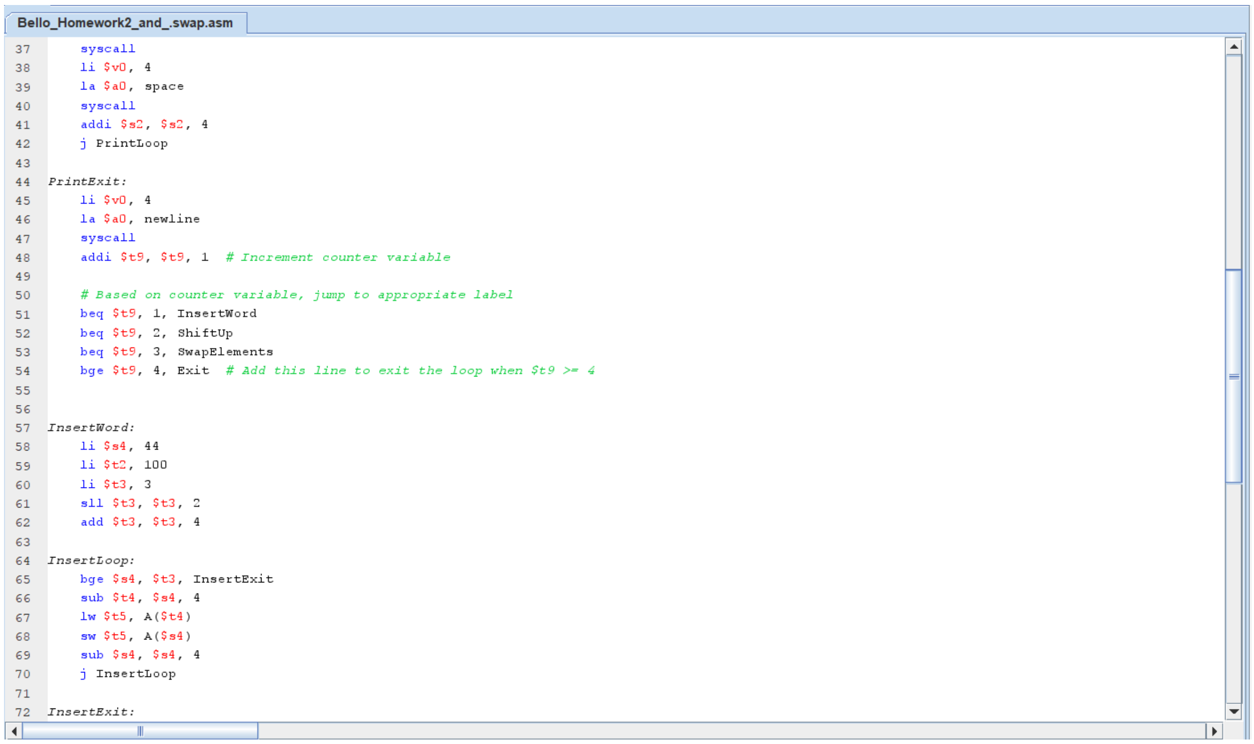
I. Objective

The objective of this assignment is to create a MIPS assembly program that focuses on array manipulation and memory operations. Initially, the program populates an 11-element array with the first 10 Fibonacci numbers, leaving the 11th element uninitialized. The program then performs a series of tasks: inserting a new word (100) after a specific index in the array, shifting all elements upwards by one position, and swapping two specific elements. After each operation, the program prints the current state of the array to demonstrate the impact of these manipulations.

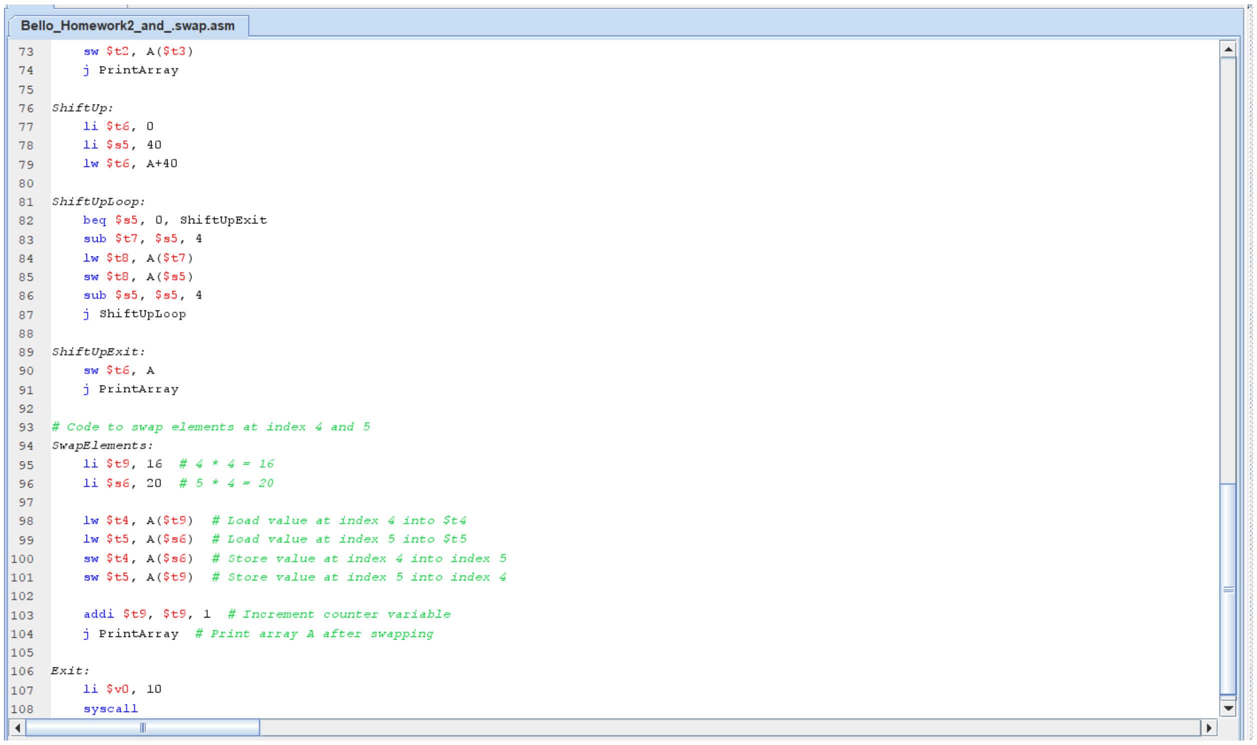
II. Code



```
.data
A: .space 44
newline: .asciiz "\n"
space: .asciiz " "

.text
.globl main

main:
    li $t9, 0  # Initialize counter variable to control the flow

    # Initialize and fill array A with Fibonacci numbers
    li $s0, 0
    li $s1, 1
    li $s2, 0

FibLoop:
    sw $s0, A($s2)
    add $t0, $s0, $s1
    move $s0, $s1
    move $s1, $t0
    addi $s2, $s2, 4
    bne $s2, 40, FibLoop

    # Print array A after filling with Fibonacci numbers
    j PrintArray

PrintArray:
    li $s2, 0
    li $s3, 44

PrintLoop:
    bge $s2, $s3, PrintExit
    lw $t1, A($s2)
    li $v0, 1
    move $a0, $t1
```



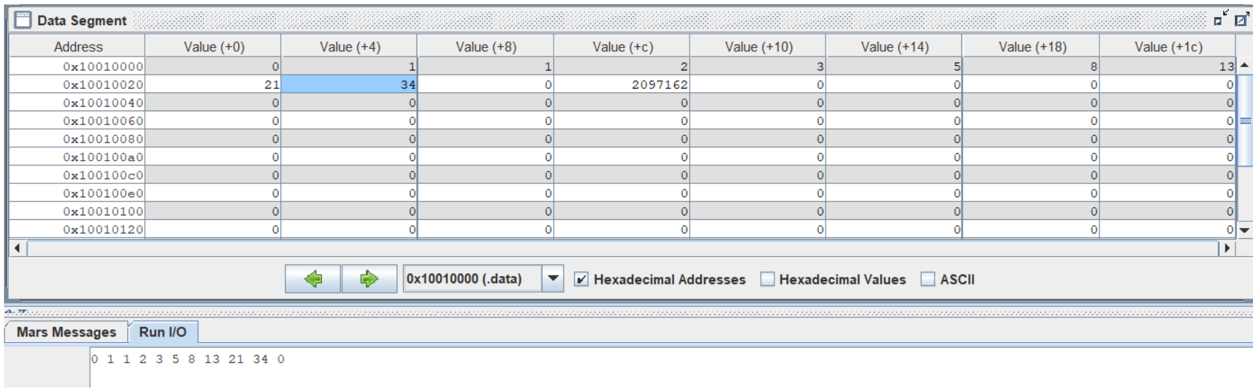
```
    syscall
    li $v0, 4
    la $a0, space
    syscall
    addi $s2, $s2, 4
    j PrintLoop

PrintExit:
    li $v0, 4
    la $a0, newline
    syscall
    addi $t9, $t9, 1  # Increment counter variable

    # Based on counter variable, jump to appropriate label
    beq $t9, 1, InsertWord
    beq $t9, 2, ShiftUp
    beq $t9, 3, SwapElements
    bge $t9, 4, Exit  # Add this line to exit the loop when $t9 >= 4


InsertWord:
    li $s4, 44
    li $t2, 100
    li $t3, 3
    sll $t3, $t3, 2
    add $t3, $t3, 4

InsertLoop:
    bge $s4, $t3, InsertExit
    sub $t4, $s4, 4
    lw $t5, A($t4)
    sw $t5, A($s4)
    sub $s4, $s4, 4
    j InsertLoop

InsertExit:
```


Bello_Homework2_and_.swap.asm

```
    sw $t2, A($t3)
    j PrintArray

ShiftUp:
    li $t6, 0
    li $s5, 40
    lw $t6, A+40

ShiftUpLoop:
    beq $s5, 0, ShiftUpExit
    sub $t7, $s5, 4
    lw $t8, A($t7)
    sw $t8, A($s5)
    sub $s5, $s5, 4
    j ShiftUpLoop

ShiftUpExit:
    sw $t6, A
    j PrintArray

# Code to swap elements at index 4 and 5
SwapElements:
    li $t9, 16  # 4 * 4 = 16
    li $s6, 20  # 5 * 4 = 20

    lw $t4, A($t9)  # Load value at index 4 into $t4
    lw $t5, A($s6)  # Load value at index 5 into $t5
    sw $t4, A($s6)  # Store value at index 4 into index 5
    sw $t5, A($t9)  # Store value at index 5 into index 4

    addi $t9, $t9, 1  # Increment counter variable
    j PrintArray  # Print array A after swapping

Exit:
    li $v0, 10
    syscall
```

III. Screenshots

Screenshot 1 - Array A with an extra space



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	2097162	0	0	0	0
0x10010040	0	0	0	0	0	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

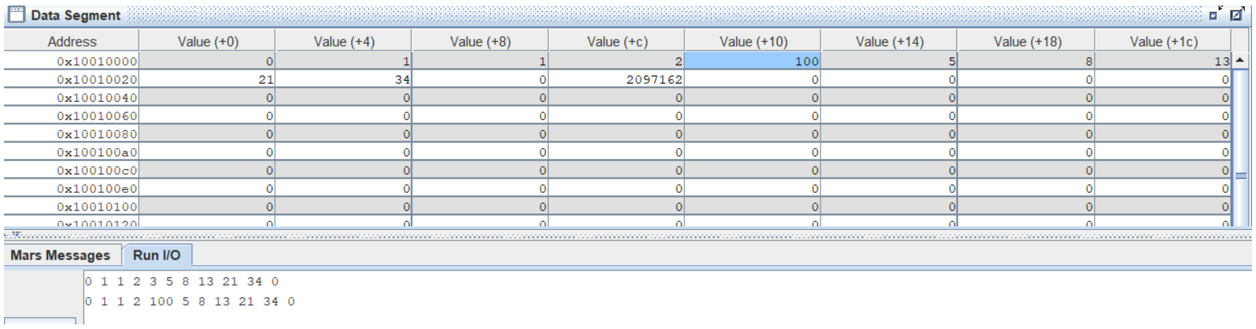
0x10010000 (.data) [✓ Hexadecimal Addresses] Hexadecimal Values ASCII

Mars Messages | Run I/O

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

Explanation: This is the initial state of array A. It's filled with the first 10 Fibonacci numbers. The last '0' is an uninitialized slot in the array. The array has 11 slots but only 10 are filled initially.

Screenshot 2 - Array A After InsertWord Operation



Data Segment

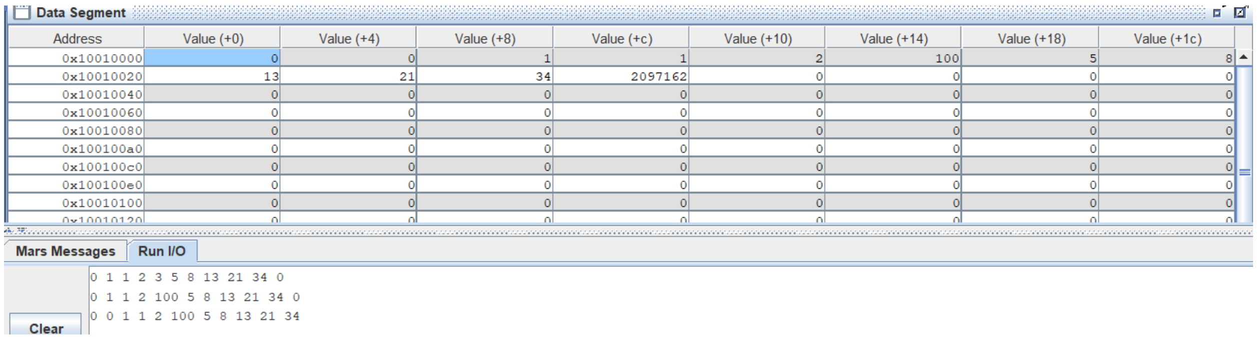
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	1	1	2	100	5	8	13
0x10010020	21	34	0	2097162	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	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

Mars Messages | Run I/O

```
0 1 1 2 3 5 8 13 21 34 0
0 1 1 2 100 5 8 13 21 34 0
```

Explanation: The number '100' is inserted after the element at index 3 (value '2'), pushing all subsequent elements one position to the right. The last '0' remains the same.

Screenshot 3 - Array A After ShiftUp Operation



Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	1	1	2	100	5	8
0x10010020	13	21	34	2097162	0	0	0	0
0x10010040	0	0	0	0	0	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

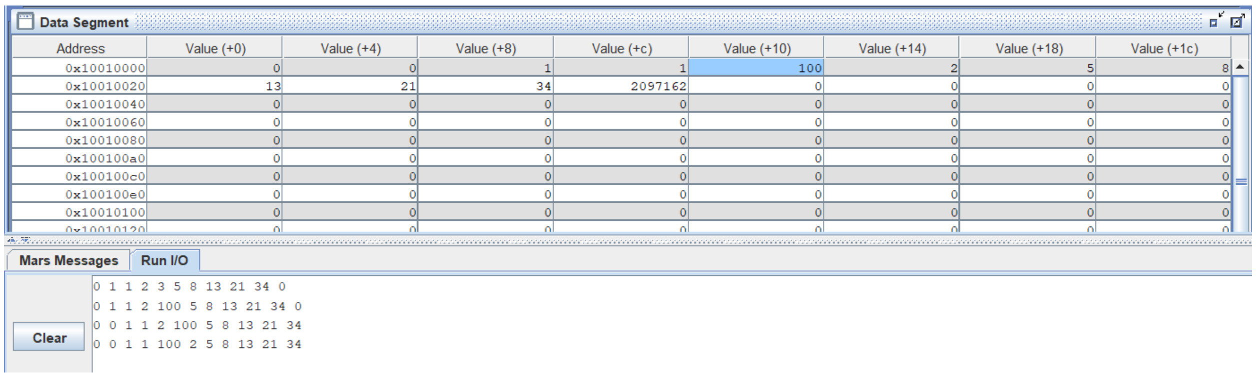
Mars Messages | Run I/O

```
0 1 1 2 3 5 8 13 21 34 0
0 1 1 2 100 5 8 13 21 34 0
0 0 1 1 2 100 5 8 13 21 34
```

Clear

Explanation: All elements are shifted up by one position. The last element (which was '0') takes the place of the first element, effectively rotating the array.

Screenshot 4 - Array A After SwapElements Operation



Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	1	1	100	2	5	8
0x10010020	13	21	34	2097162	0	0	0	0
0x10010040	0	0	0	0	0	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

Mars Messages | Run I/O

```
0 1 1 2 3 5 8 13 21 34 0
0 1 1 2 100 5 8 13 21 34 0
0 0 1 1 2 100 5 8 13 21 34
0 0 1 1 100 2 5 8 13 21 34
```

Clear

Explanation: The elements at index 4 and 5 (values '100' and '2') are swapped.