COE 181. 1 Lab Report

Lab Number: Lab 1

Lab Title: INTRODUCTION TO MARS

Student Name: Janna Joyce E. Jumao-as

Student Email: jannajoyce.jumao-as@g.msuiit.edu.ph

Student Phone Number: 09309321199

Objectives:

- Get familiar with the MARS simulator.
- Learn how to assemble, run, and debug a MIPS program

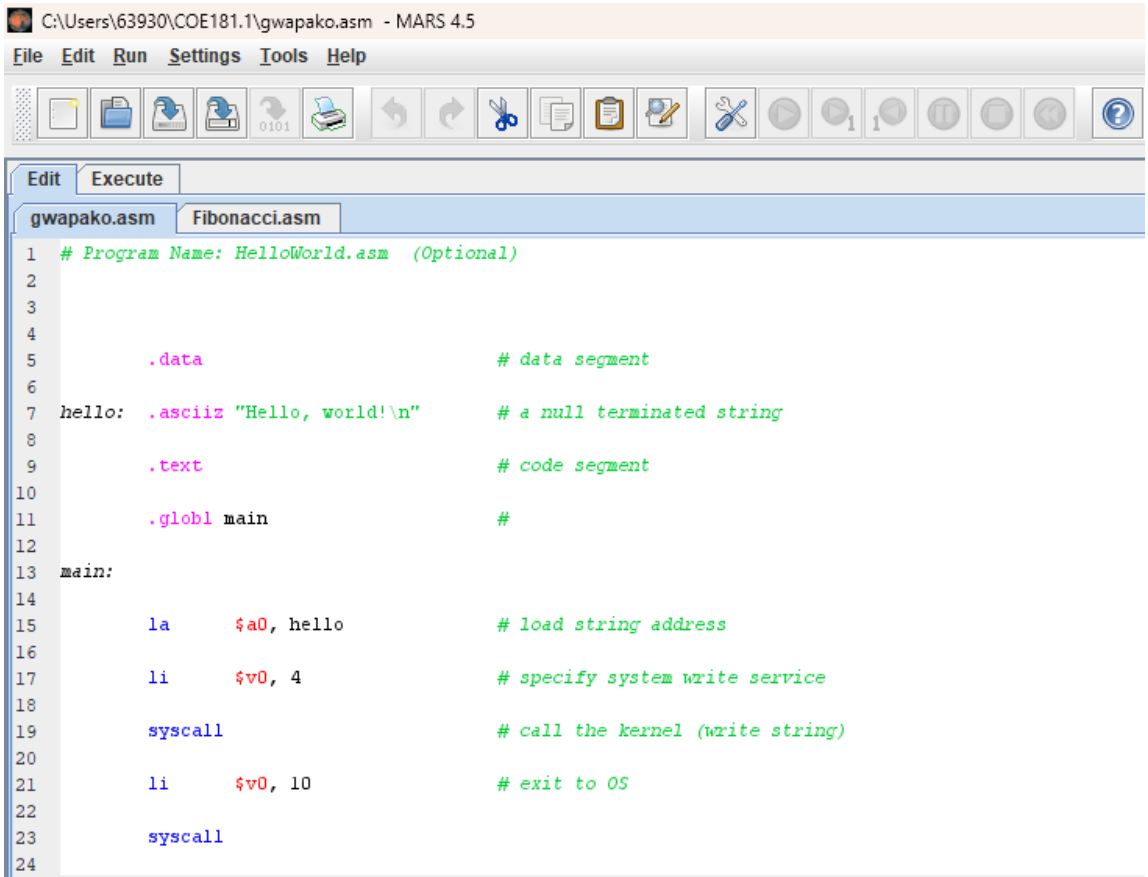
Procedures:

- Downloaded the MARS IDE.
- Familiarized the components of MARS Simulator.
- Learned to assemble, run and debug.

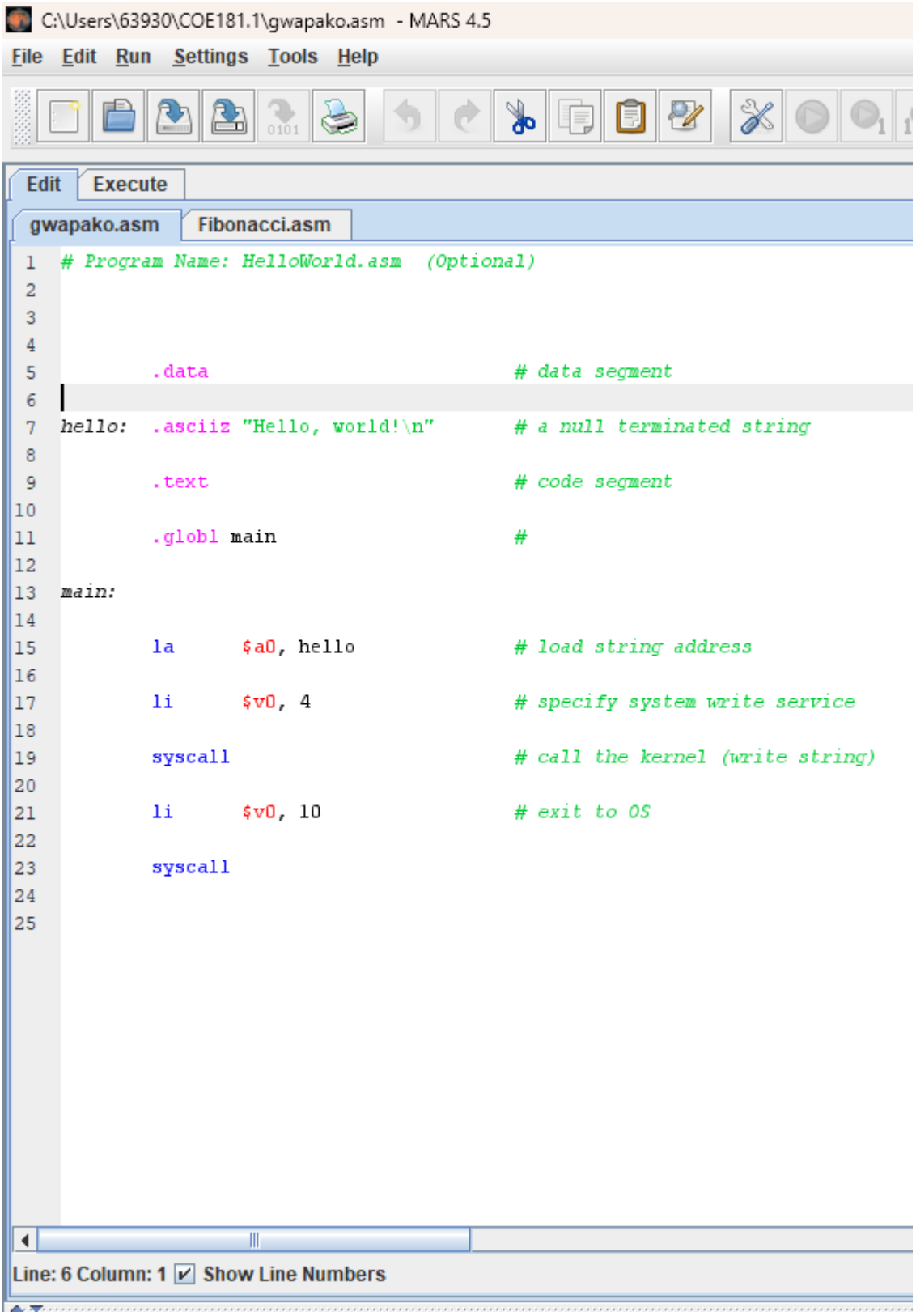
Results and Analysis:

Task 1:

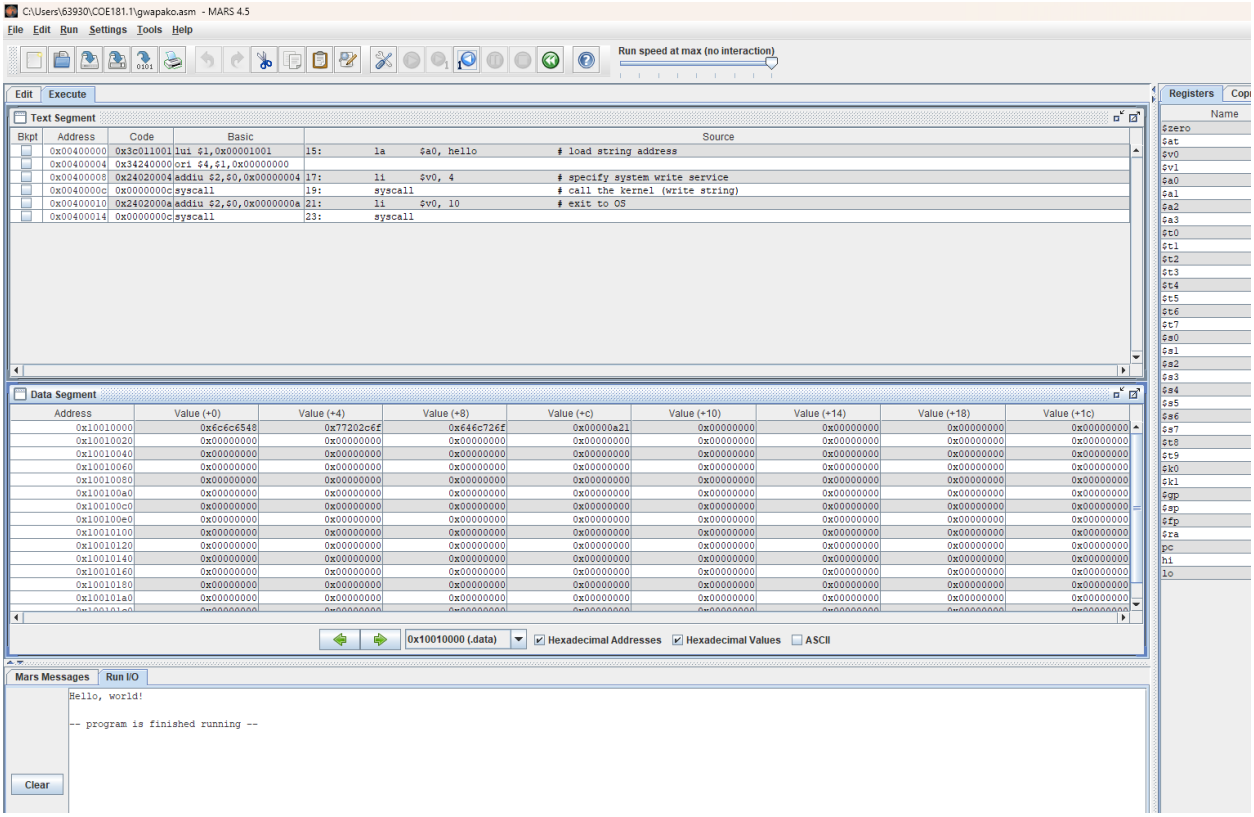
- ***a.***



- ***b.***


C:\Users\63930\COE181.1\gwapako.asm - MARS 4.5
File Edit Run Settings Tools Help
Edit Execute
gwapako.asm Fibonacci.asm
1 # Program Name: HelloWorld.asm (Optional)
2
3
4
5 .data # data segment
6
7 hello: .asciiz "Hello, world!\n" # a null terminated string
8
9 .text # code segment
10
11 .globl main #
12
13 main:
14
15 la $a0, hello # load string address
16
17 li $v0, 4 # specify system write service
18
19 syscall # call the kernel (write string)
20
21 li $v0, 10 # exit to OS
22
23 syscall
24
25
Line: 6 Column: 1 Show Line Numbers

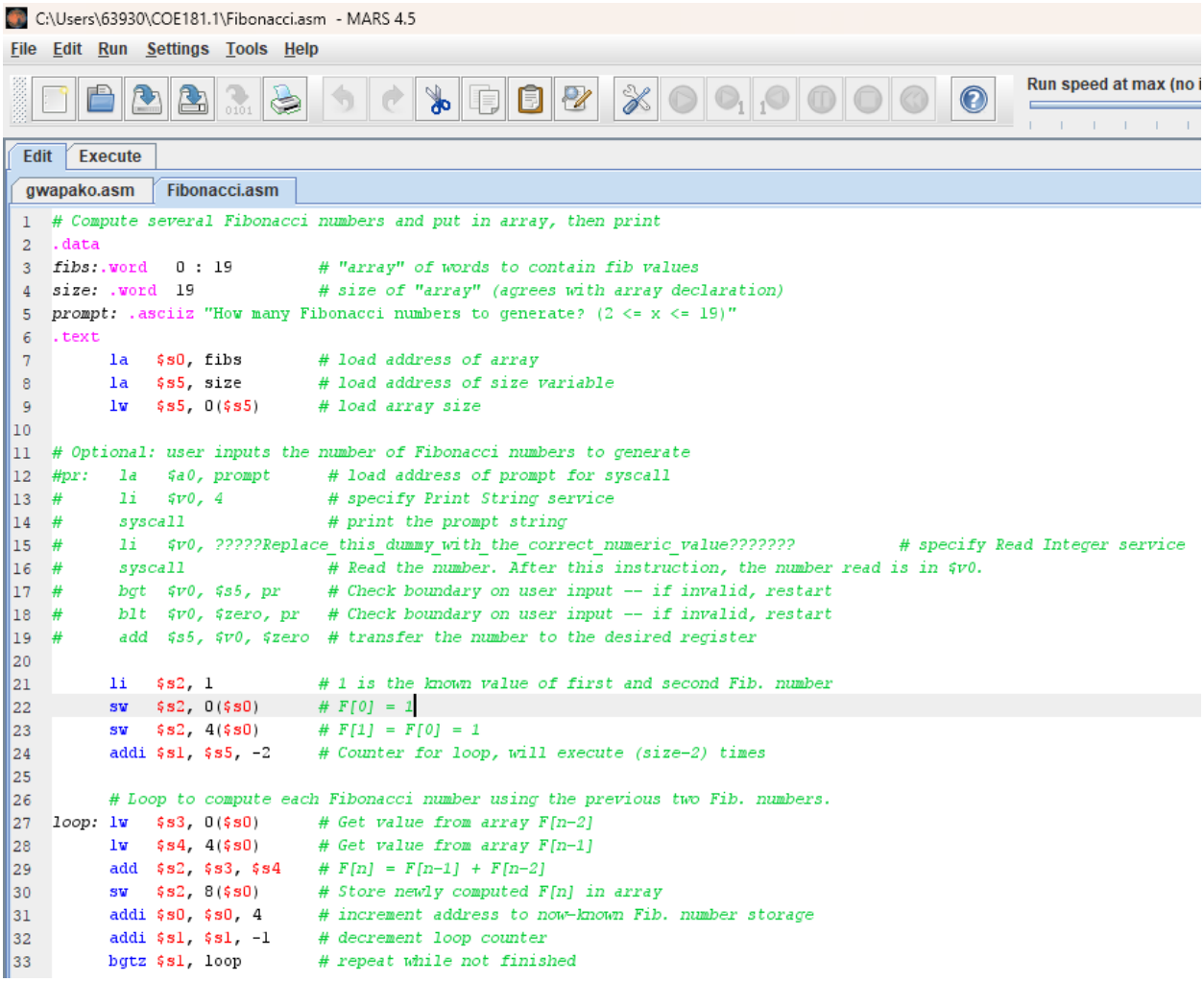
- ***c.***



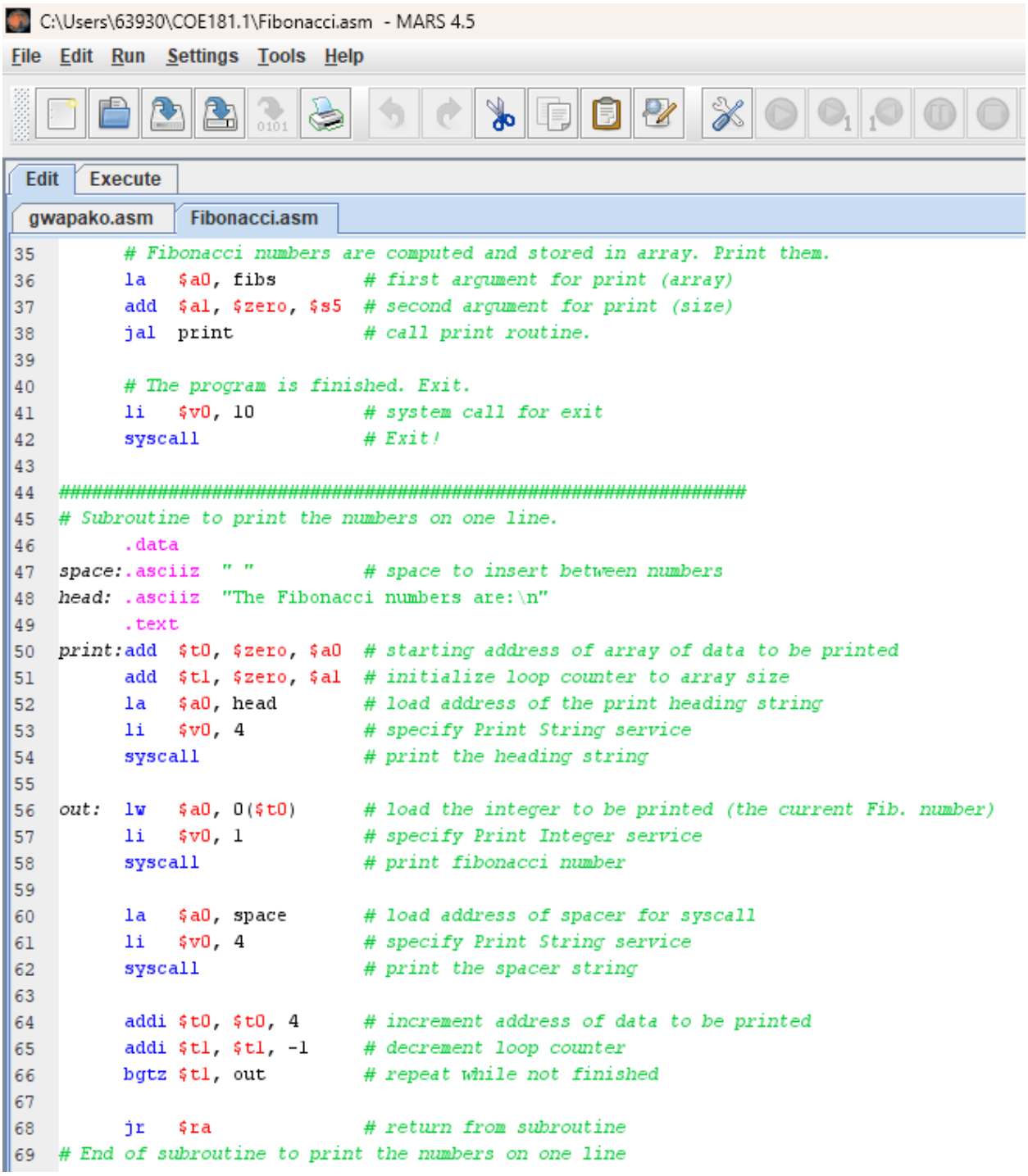
- ***d.*** *The output is Hello, world! And it appears in Run I/O display window.*

Task 2:

- ***a.***



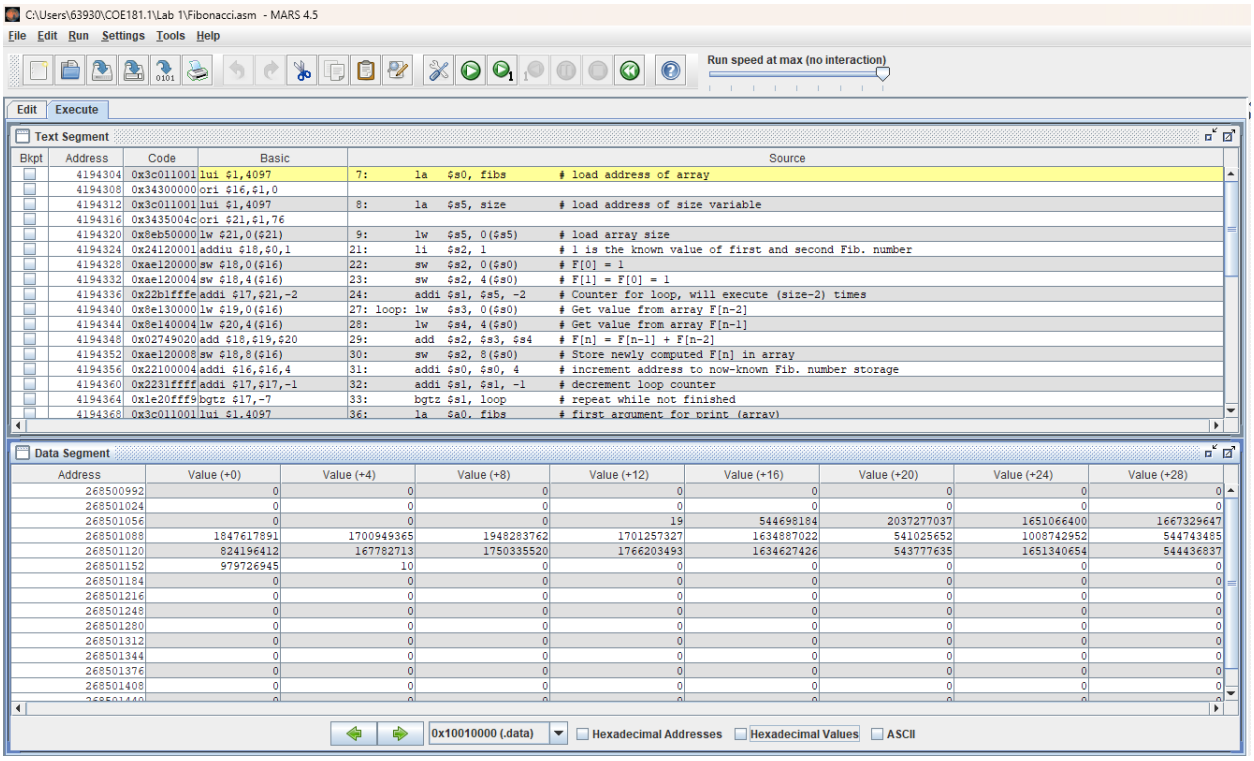

```
1  # Compute several Fibonacci numbers and put in array, then print
2  .data
3  fibs:.word   0 : 19         # "array" of words to contain fib values
4  size: .word  19             # size of "array" (agrees with array declaration)
5  prompt: .asciiz "How many Fibonacci numbers to generate? (2 <= x <= 19)"
6  .text
7        la   $s0, fibs        # load address of array
8        la   $s5, size        # load address of size variable
9        lw   $s5, 0($s5)      # load array size
10
11 # Optional: user inputs the number of Fibonacci numbers to generate
12 #pr:   la   $a0, prompt      # load address of prompt for syscall
13 #      li   $v0, 4           # specify Print String service
14 #      syscall              # print the prompt string
15 #      li   $v0, ?????Replace_this_dummy_with_the_correct_numeric_value???????          # specify Read Integer service
16 #      syscall              # Read the number. After this instruction, the number read is in $v0.
17 #      bgt  $v0, $s5, pr    # Check boundary on user input -- if invalid, restart
18 #      blt  $v0, $zero, pr  # Check boundary on user input -- if invalid, restart
19 #      add  $s5, $v0, $zero # transfer the number to the desired register
20
21       li   $s2, 1           # 1 is the known value of first and second Fib. number
22       sw   $s2, 0($s0)      # F[0] = 1
23       sw   $s2, 4($s0)      # F[1] = F[0] = 1
24       addi $s1, $s5, -2     # Counter for loop, will execute (size-2) times
25
26       # Loop to compute each Fibonacci number using the previous two Fib. numbers.
27 loop: lw   $s3, 0($s0)      # Get value from array F[n-2]
28       lw   $s4, 4($s0)      # Get value from array F[n-1]
29       add  $s2, $s3, $s4    # F[n] = F[n-1] + F[n-2]
30       sw   $s2, 8($s0)      # Store newly computed F[n] in array
31       addi $s0, $s0, 4      # increment address to now-known Fib. number storage
32       addi $s1, $s1, -1     # decrement loop counter
33       bgtz $s1, loop        # repeat while not finished
```



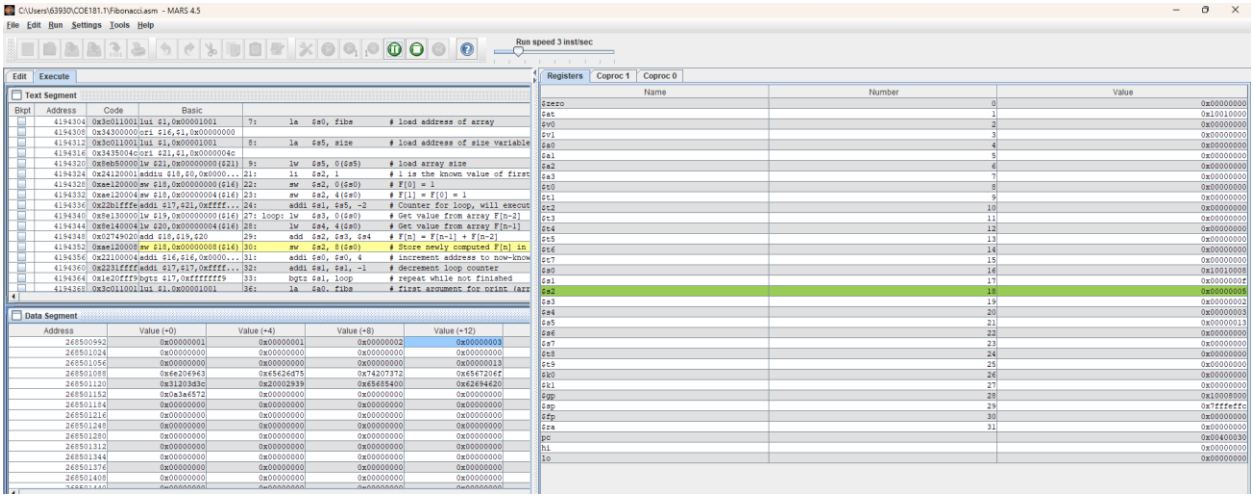
- ***b.*** *In the .data section, several key variables are defined for the program. The fibs array consists of 19 words, each initialized to zero, and is used to store the Fibonacci numbers as they are generated. The size variable holds a single word with the value 19, representing the size of the Fibonacci array. A string labeled prompt is included to prompt the user to input the number of Fibonacci numbers to generate. Additionally, the space variable contains a newline and space character, which are used to format the*

output when displaying the Fibonacci numbers. Finally, the head string is defined to print a heading before the Fibonacci sequence is shown.

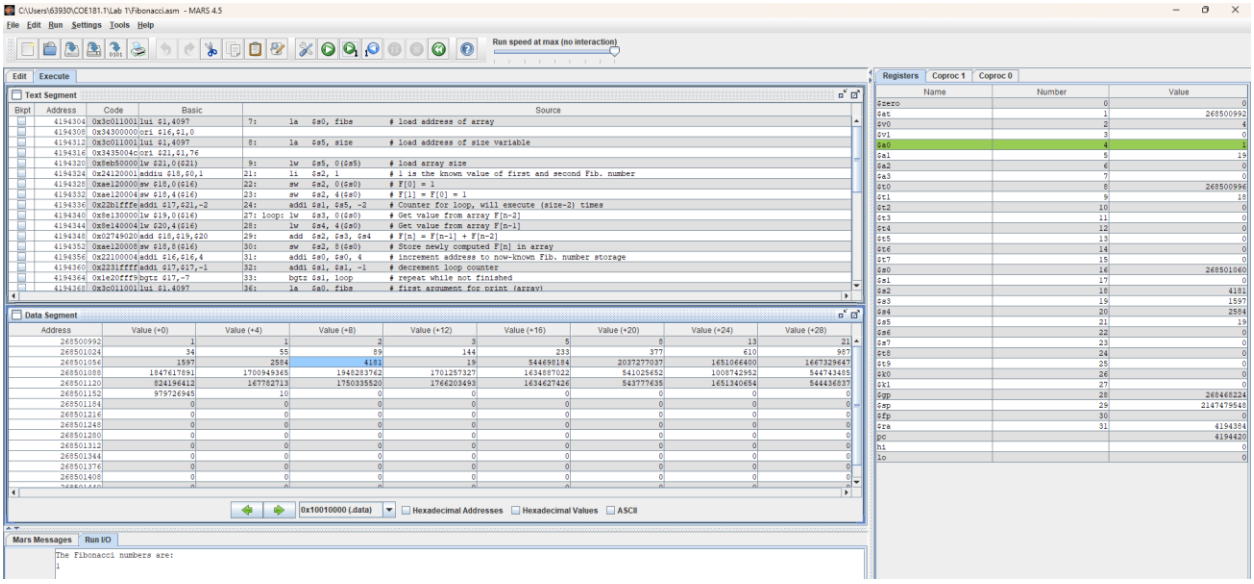
- ***c.***



- ***d.***



- ***e.***

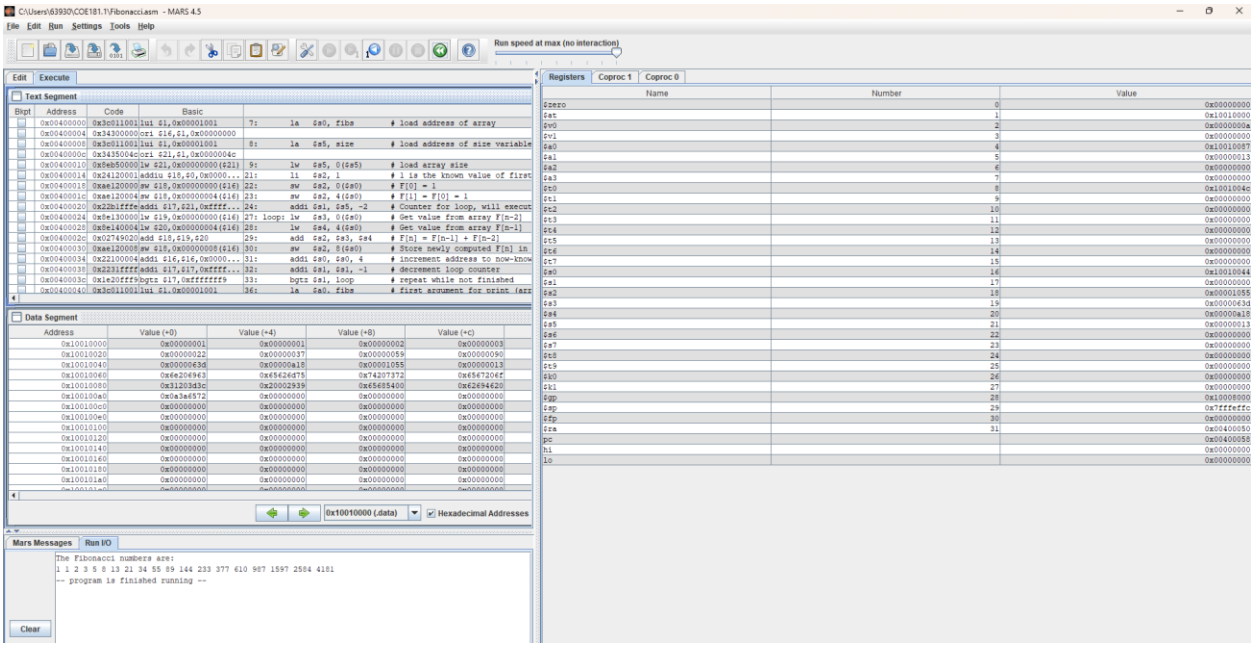


- *Line 7: $s0 points to the start of the fibs array.*
- *Line 8-9: $s5 holds the size of the array, which is 19.*
- *Line 21: $s2 initialized to 1.*
- *Line 21-22: fibs[0] =1 and fibs [1] = 1.*
- *For loop execution: fibs[2] = 2*

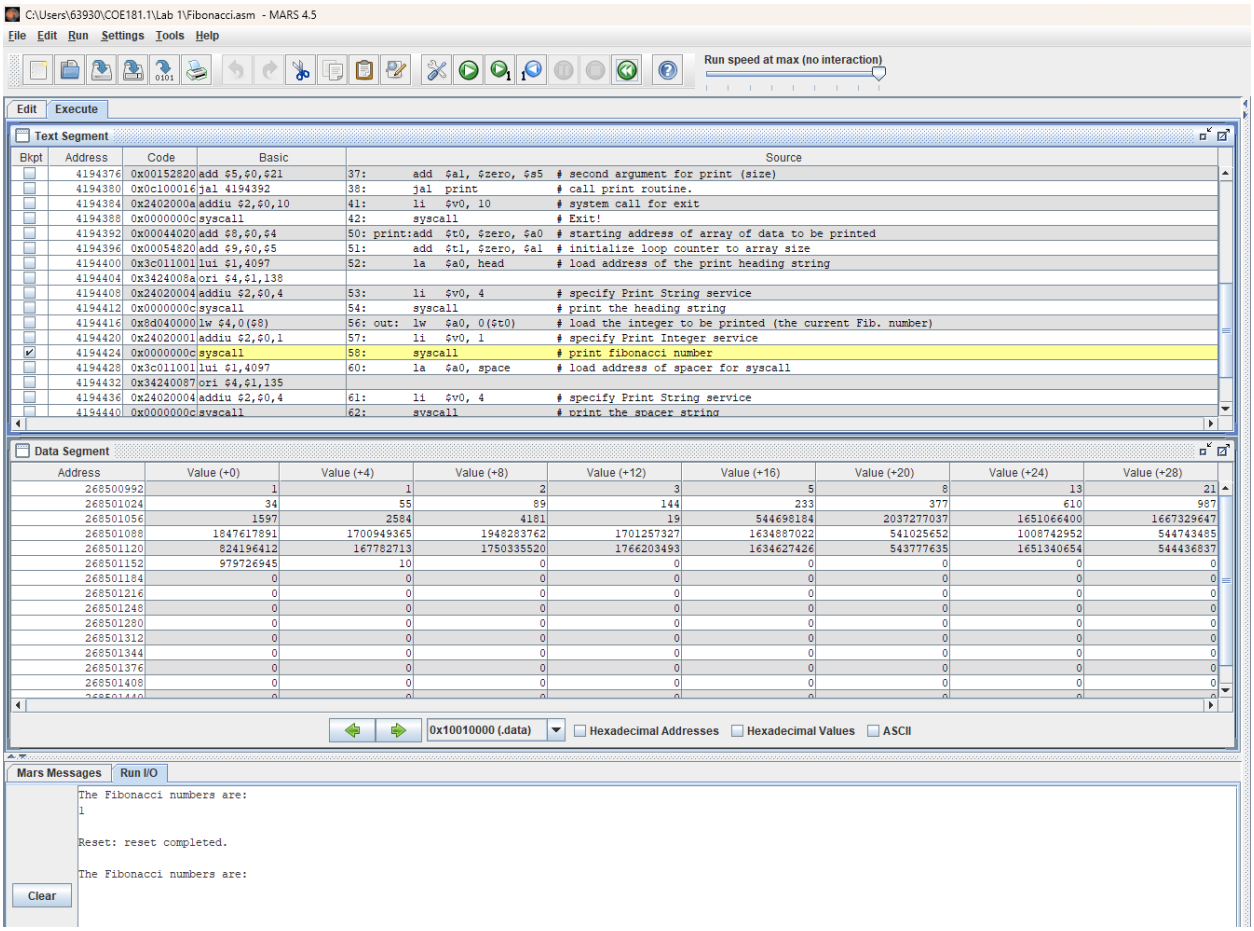
  since at line 27 $s3 = 1 (fibs[0]) and at line 28 $s4 = 1 (fibs[1]).
  At line 29, there is add instruction, $s2 = 1+1 = 2 (fibd[2]).
  Now, in line 31, $s0 points to fibs[1].

  Same process continues for the next iterations up to fibs[18].

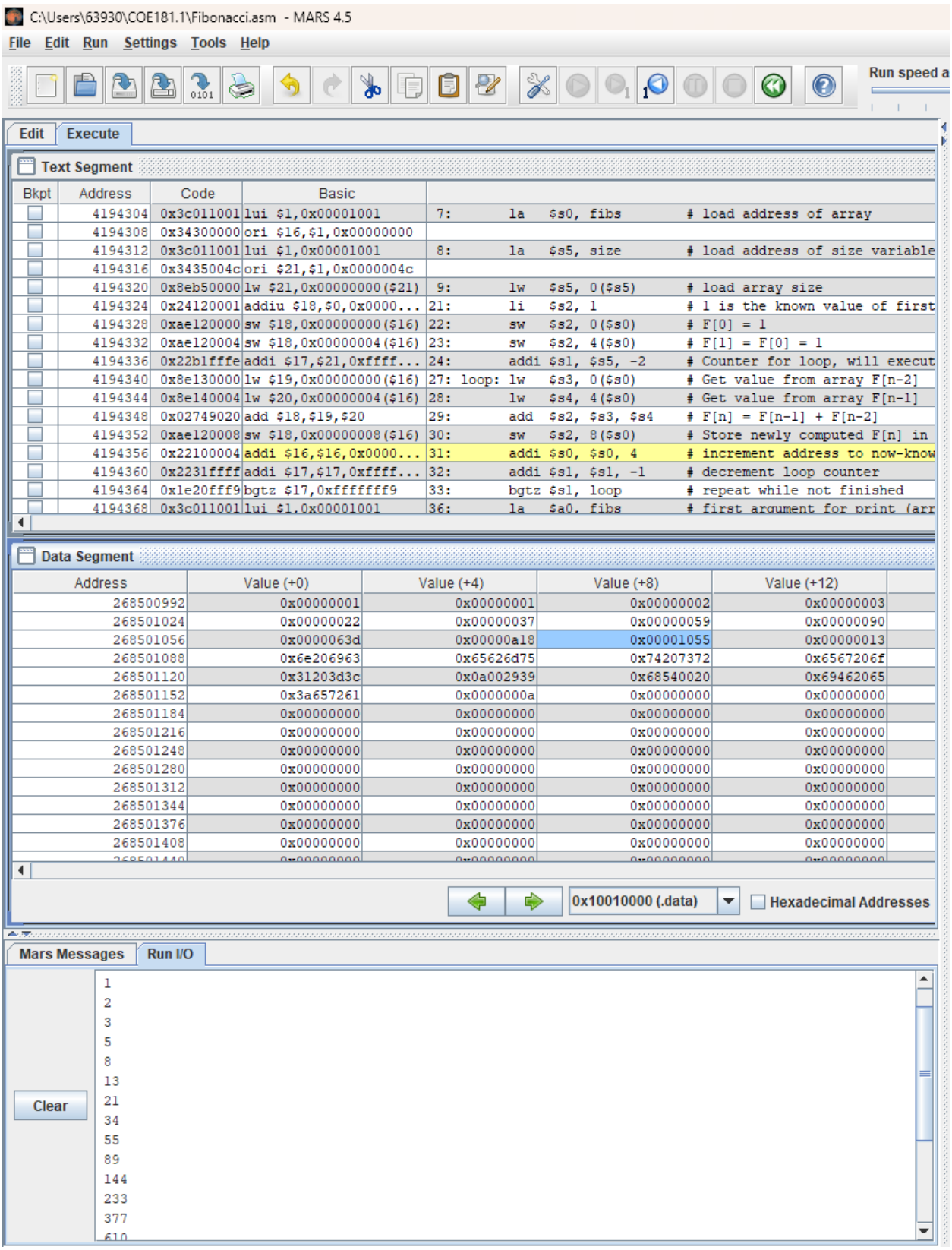
- *f.*


C:\Users\63930\COE181.1\Fibonacci.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic
0x00400000 0x3c011001 lui $1,0x00001001 7: la $a0, fibs # load address of array
0x00400004 0x34300000 ori $16,$1,0x00000000
0x00400008 0x3c011001 lui $1,0x00001001 8: la $a5, size # load address of size variable
0x0040000c 0x3435004c ori $21,$1,0x0000004c
0x00400010 0x8eb50000 lw $21,0x00000000($21) 9: lw $a5, 0($a5) # load array size
0x00400014 0x24120001 addiu $18,$0,0x0000... 21: li $s2, 1 # 1 is the known value of first
0x00400018 0xae120000 sw $18,0x00000000($16) 22: sw $s2, 0($a0) # F[0] = 1
0x0040001c 0xae120004 sw $18,0x00000004($16) 23: sw $s2, 4($a0) # F[1] = F[0] = 1
0x00400020 0x22b1fffe addi $17,$21,0xffff... 24: addi $s1, $a5, -2 # Counter for loop, will execut
0x00400024 0x8e130000 lw $19,0x00000000($16) 27: loop: lw $s3, 0($a0) # Get value from array F[n-2]
0x00400028 0x8e140004 lw $20,0x00000004($16) 28: lw $s4, 4($a0) # Get value from array F[n-1]
0x0040002c 0x02749020 add $18,$19,$20 29: add $s2, $s3, $s4 # F[n] = F[n-1] + F[n-2]
0x00400030 0xae120008 sw $18,0x00000008($16) 30: sw $s2, 8($a0) # Store newly computed F[n] in
0x00400034 0x22100004 addi $16,$16,0x0000... 31: addi $a0, $a0, 4 # increment address to now-know
0x00400038 0x2231ffff addi $17,$17,0xffff... 32: addi $s1, $s1, -1 # decrement loop counter
0x0040003c 0x1e20fff9 bgtz $17,0xfffffff9 33: bgtz $s1, loop # repeat while not finished
0x00400040 0x3c011001 lui $1,0x00001001 36: la $a0, fibs # first argument for print (arr
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c)
0x10010000 0x00000001 0x00000001 0x00000002 0x00000003
0x10010020 0x00000022 0x00000037 0x00000059 0x00000090
0x10010040 0x0000063d 0x00000a18 0x00001055 0x00000013
0x10010060 0x4e206963 0x65626d75 0x74207372 0x6567206f
0x10010080 0x31203d3c 0x20002939 0x65685400 0x62694620
0x100100a0 0x6e1a4572 0x00000000 0x00000000 0x00000000
0x100100c0 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000
0x10010100 0x00000000 0x00000000 0x00000000 0x00000000
0x10010120 0x00000000 0x00000000 0x00000000 0x00000000
0x10010140 0x00000000 0x00000000 0x00000000 0x00000000
0x10010160 0x00000000 0x00000000 0x00000000 0x00000000
0x10010180 0x00000000 0x00000000 0x00000000 0x00000000
0x100101a0 0x00000000 0x00000000 0x00000000 0x00000000
0x10010000 (.data)
Hexadecimal Addresses
Mars Messages Run I/O
The Fibonacci numbers are:
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
-- program is finished running --
Clear
Registers Coproc 1 Coproc 0
Name Number Value
$zero 0 0x00000000
$at 1 0x10010000
$v0 2 0x0000000a
$v1 3 0x00000000
$a0 4 0x10010087
$a1 5 0x00000013
$a2 6 0x00000000
$a3 7 0x00000000
$t0 8 0x1001004c
$t1 9 0x00000000
$t2 10 0x00000000
$t3 11 0x00000000
$t4 12 0x00000000
$t5 13 0x00000000
$t6 14 0x00000000
$t7 15 0x00000000
$s0 16 0x10010044
$s1 17 0x00000000
$s2 18 0x00001055
$s3 19 0x0000063d
$s4 20 0x00000a18
$s5 21 0x00000013
$s6 22 0x00000000
$s7 23 0x00000000
$t8 24 0x00000000
$t9 25 0x00000000
$k0 26 0x00000000
$k1 27 0x00000000
$gp 28 0x10008000
$sp 29 0x7fffeffc
$fp 30 0x00000000
$ra 31 0x00400050
pc 0x00400058
hi 0x00000000
lo 0x00000000

- ***g.***
  Address: 4154424 or 0X00400075
- ***h.***



- ***i.***
  If we change the line:
  ***space: .asciiz " " # space to insert between numbers** to:*
  space: .asciiz "\n" # space to insert between number
  the output would print vertically.


C:\Users\63930\COE181.1\Fibonacci.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed a
Edit Execute
Text Segment
Bkpt Address Code Basic
4194304 0x3c011001 lui $1,0x00001001 7: la $s0, fibs # load address of array
4194308 0x34300000 ori $16,$1,0x00000000
4194312 0x3c011001 lui $1,0x00001001 8: la $s5, size # load address of size variable
4194316 0x3435004c ori $21,$1,0x0000004c
4194320 0x8eb50000 lw $21,0x00000000($21) 9: lw $s5, 0($s5) # load array size
4194324 0x24120001 addiu $18,$0,0x0000... 21: li $s2, 1 # 1 is the known value of first
4194328 0xae120000 sw $18,0x00000000($16) 22: sw $s2, 0($s0) # F[0] = 1
4194332 0xae120004 sw $18,0x00000004($16) 23: sw $s2, 4($s0) # F[1] = F[0] = 1
4194336 0x22b1fffe addi $17,$21,0xffff... 24: addi $s1, $s5, -2 # Counter for loop, will execut
4194340 0x8e130000 lw $19,0x00000000($16) 27: loop: lw $s3, 0($s0) # Get value from array F[n-2]
4194344 0x8e140004 lw $20,0x00000004($16) 28: lw $s4, 4($s0) # Get value from array F[n-1]
4194348 0x02749020 add $18,$19,$20 29: add $s2, $s3, $s4 # F[n] = F[n-1] + F[n-2]
4194352 0xae120008 sw $18,0x00000008($16) 30: sw $s2, 8($s0) # Store newly computed F[n] in
4194356 0x22100004 addi $16,$16,0x0000... 31: addi $s0, $s0, 4 # increment address to now-know
4194360 0x2231ffff addi $17,$17,0xffff... 32: addi $s1, $s1, -1 # decrement loop counter
4194364 0x1e20fff9 bgtz $17,0xfffffff9 33: bgtz $s1, loop # repeat while not finished
4194368 0x3c011001 lui $1,0x00001001 36: la $a0, fibs # first argument for print (arr
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+12)
268500992 0x00000001 0x00000001 0x00000002 0x00000003
268501024 0x00000022 0x00000037 0x00000059 0x00000090
268501056 0x0000063d 0x00000a18 0x00001055 0x00000013
268501088 0x6e206963 0x65626d75 0x74207372 0x6567206f
268501120 0x31203d3c 0x0a002939 0x68540020 0x69462065
268501152 0x3a657261 0x0000000a 0x00000000 0x00000000
268501184 0x00000000 0x00000000 0x00000000 0x00000000
268501216 0x00000000 0x00000000 0x00000000 0x00000000
268501248 0x00000000 0x00000000 0x00000000 0x00000000
268501280 0x00000000 0x00000000 0x00000000 0x00000000
268501312 0x00000000 0x00000000 0x00000000 0x00000000
268501344 0x00000000 0x00000000 0x00000000 0x00000000
268501376 0x00000000 0x00000000 0x00000000 0x00000000
268501408 0x00000000 0x00000000 0x00000000 0x00000000
0x10010000 (.data)
Hexadecimal Addresses
Mars Messages Run I/O
Clear
1
2
3
5
8
13
21
34
55
89
144
233
377
610

Conclusion:

- This activity helped the student become familiar with the components of the MARS simulator.
- The student appreciated the well-structured manual, as it effectively guided them in exploring the MARS simulator and understanding how MIPS assembly language functions.

Additional Notes/Observations:

- So far, the student has not encountered any significant difficulties, except for the small font size, as the MARS simulator does not support zooming in and out.