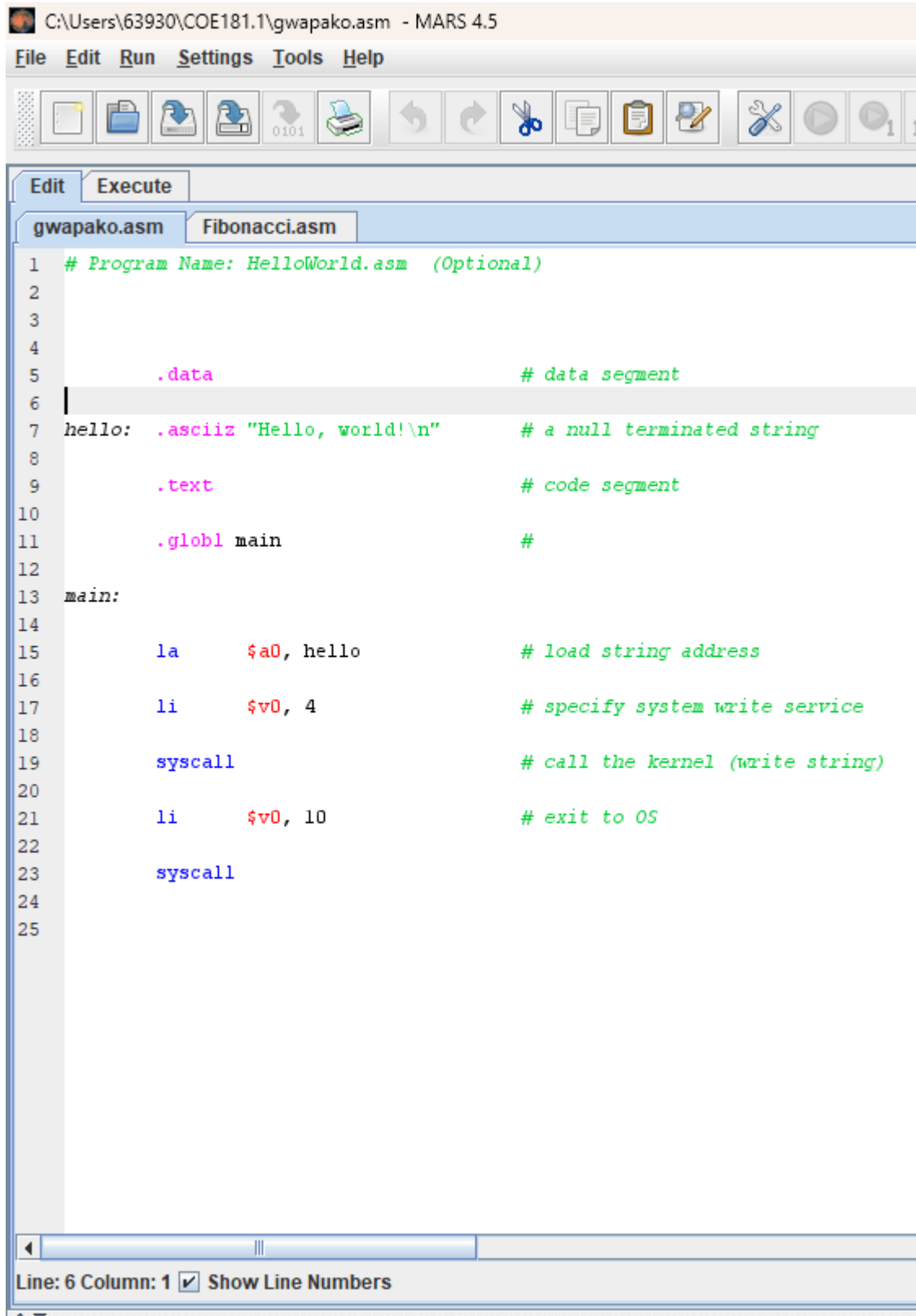


- b.

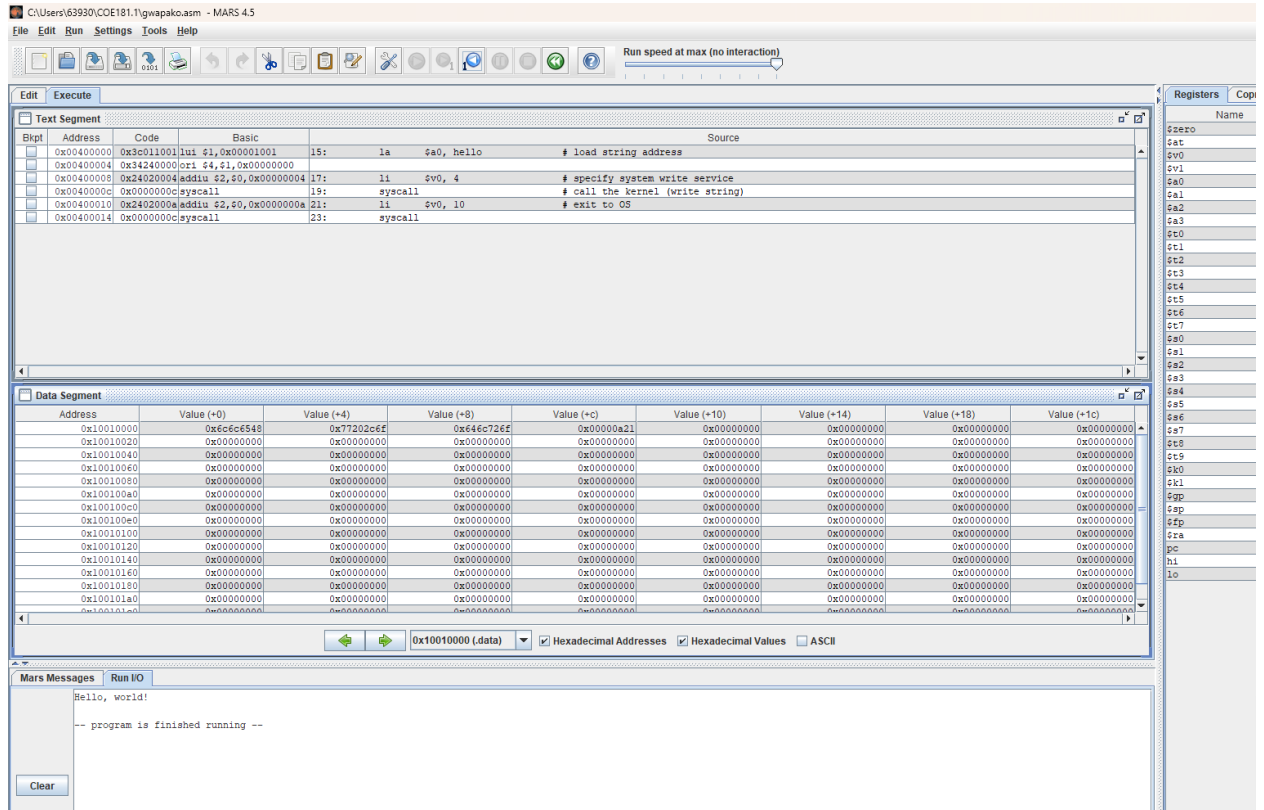


The screenshot shows the MARS 4.5 MIPS assembler interface. The title bar indicates the file path is C:\Users\63930\COE181.1\gwapako.asm - MARS 4.5. The menu bar includes File, Edit, Run, Settings, Tools, and Help. The toolbar contains icons for file operations (new, open, save, print), navigation (back, forward, search), and execution (run, step through, step over, step under). The main window has two tabs: gwapako.asm (active) and Fibonacci.asm. The assembly code is displayed with line numbers from 1 to 25. The code defines a data segment with a string 'Hello, world!\n' and a code segment with a main function that loads the string address, specifies the system write service, and calls the kernel to write the string and exit.

```
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

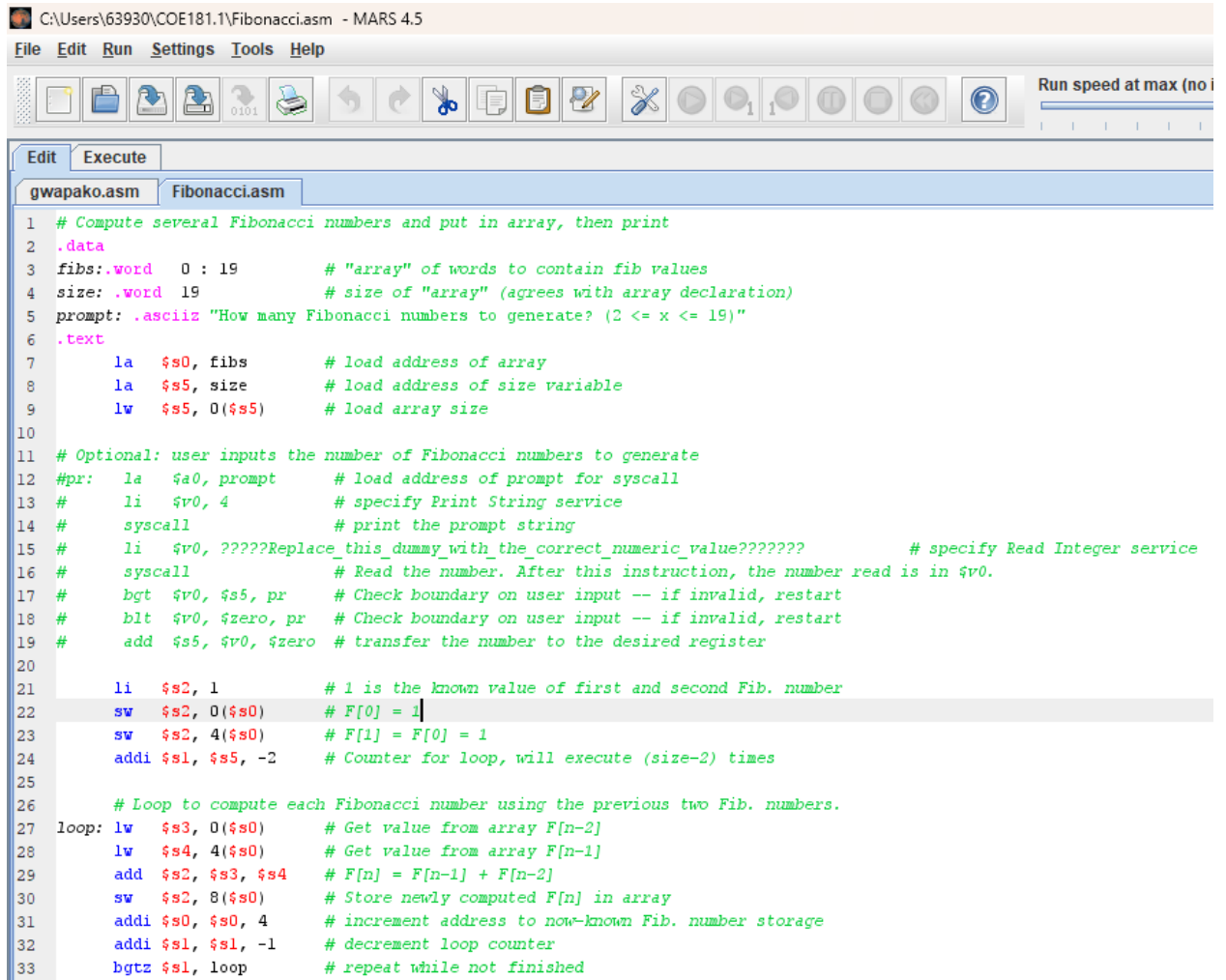
- *c.*



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

Task 2:

- a.



```
C:\Users\63930\COE181.1\Fibonacci.asm - MARS 4.5
File Edit Run Settings Tools Help

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
```

C:\Users\63930\COE181.1\Fibonacci.asm - MARS 4.5

File Edit Run Settings Tools Help

0101

Edit Execute

gwapako.asm Fibonacci.asm

```

35      # Fibonacci numbers are computed and stored in array. Print them.
36      la  $a0, fibs          # first argument for print (array)
37      add $a1, $zero, $s5    # second argument for print (size)
38      jal print              # call print routine.
39
40      # The program is finished. Exit.
41      li  $v0, 10            # system call for exit
42      syscall                # Exit!
43
44      #####
45      # Subroutine to print the numbers on one line.
46      .data
47      space: .asciiz " "      # space to insert between numbers
48      head:  .asciiz "The Fibonacci numbers are:\n"
49      .text
50      print: add $t0, $zero, $a0 # starting address of array of data to be printed
51             add $t1, $zero, $a1 # initialize loop counter to array size
52             la  $a0, head       # load address of the print heading string
53             li  $v0, 4          # specify Print String service
54             syscall            # print the heading string
55
56      out:  lw  $a0, 0($t0)      # load the integer to be printed (the current Fib. number)
57             li  $v0, 1          # specify Print Integer service
58             syscall            # print fibonacci number
59
60             la  $a0, space     # load address of spacer for syscall
61             li  $v0, 4          # specify Print String service
62             syscall            # print the spacer string
63
64             addi $t0, $t0, 4    # increment address of data to be printed
65             addi $t1, $t1, -1   # decrement loop counter
66             bgtz $t1, out       # repeat while not finished
67
68             jr  $ra            # return from subroutine
69      # End of subroutine to print the numbers on one line

```

- **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.

C:\Users\63930\COE181\Lab 1\Fibonacci.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Text Segment

Bkpt	Address	Code	Basic	Source
	4194304	0x3c011001	lui \$t0, 4097	7: la \$t0, fibs # load address of array
	4194308	0x34300000	ori \$t6, \$t0, 0	
	4194312	0x3c011001	lui \$t5, 4097	8: la \$t5, size # load address of size variable
	4194316	0x34350004	ori \$t1, \$t5, 76	
	4194320	0x8eb50000	lw \$t1, 0(\$t1)	9: lw \$t5, 0(\$t5) # load array size
	4194324	0x24120001	addiu \$t8, \$t0, 1	21: li \$t2, 1 # 1 is the known value of first and second Fib. number
	4194328	0xae120000	sw \$t8, 0(\$t6)	22: sw \$t2, 0(\$t0) # F[0] = 1
	4194332	0xae120004	sw \$t8, 4(\$t6)	23: sw \$t2, 4(\$t0) # F[1] = F[0] = 1
	4194336	0x2b1fffff	addi \$t1, \$t5, -2	24: addi \$t1, \$t5, -2 # Counter for loop, will execute (size-2) times
	4194340	0xe1300000	lw \$t3, 0(\$t0)	27: loop: lw \$t3, 0(\$t0) # Get value from array F[n-2]
	4194344	0xe1400004	lw \$t4, 4(\$t0)	28: lw \$t4, 4(\$t0) # Get value from array F[n-1]
	4194348	0x02749020	add \$t2, \$t3, \$t4	29: add \$t2, \$t3, \$t4 # F[n] = F[n-1] + F[n-2]
	4194352	0xae120008	sw \$t2, 8(\$t6)	30: sw \$t2, 8(\$t0) # Store newly computed F[n] in array
	4194356	0x21000004	addi \$t6, \$t6, 4	31: addi \$t0, \$t0, 4 # increment address to now-known Fib. number storage
	4194360	0x2b1fffff	addi \$t1, \$t1, -1	32: addi \$t1, \$t1, -1 # decrement loop counter
	4194364	0x1e20ffff	bgtz \$t1, loop	33: bgtz \$t1, loop # repeat while not finished
	4194368	0x3c011001	lui \$t0, 4097	36: la \$t0, fibs # first argument for print (array)

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	0	0	0	0	0	0	0	0
268501024	0	0	0	0	0	0	0	0
268501056	0	0	0	19	544698184	2037277037	1651066400	1667329647
268501088	1847617891	1700949365	1948283762	1701257327	1634887022	541025652	1008742952	544743485
268501120	824196412	167782713	1750335520	1766203493	1634627426	543777635	1651340654	544436837
268501152	979726945	10	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0
268501248	0	0	0	0	0	0	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0
268501440	0	0	0	0	0	0	0	0

0x1001000 (data) Hexadecimal Addresses Hexadecimal Values ASCII

• d.

C:\Users\63930\COE181\Lab 1\Fibonacci.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed 3 inst/sec

Text Segment

Bkpt	Address	Code	Basic	Source
	4194304	0x3c011001	lui \$t0, 4097	7: la \$t0, fibs # load address of array
	4194308	0x34300000	ori \$t6, \$t0, 0	
	4194312	0x3c011001	lui \$t5, 4097	8: la \$t5, size # load address of size variable
	4194316	0x34350004	ori \$t1, \$t5, 76	
	4194320	0x8eb50000	lw \$t1, 0(\$t1)	9: lw \$t5, 0(\$t5) # load array size
	4194324	0x24120001	addiu \$t8, \$t0, 1	21: li \$t2, 1 # 1 is the known value of first and second Fib. number
	4194328	0xae120000	sw \$t8, 0(\$t6)	22: sw \$t2, 0(\$t0) # F[0] = 1
	4194332	0xae120004	sw \$t8, 4(\$t6)	23: sw \$t2, 4(\$t0) # F[1] = F[0] = 1
	4194336	0x2b1fffff	addi \$t1, \$t5, -2	24: addi \$t1, \$t5, -2 # Counter for loop, will execute (size-2) times
	4194340	0xe1300000	lw \$t3, 0(\$t0)	27: loop: lw \$t3, 0(\$t0) # Get value from array F[n-2]
	4194344	0xe1400004	lw \$t4, 4(\$t0)	28: lw \$t4, 4(\$t0) # Get value from array F[n-1]
	4194348	0x02749020	add \$t2, \$t3, \$t4	29: add \$t2, \$t3, \$t4 # F[n] = F[n-1] + F[n-2]
	4194352	0xae120008	sw \$t2, 8(\$t6)	30: sw \$t2, 8(\$t0) # Store newly computed F[n] in array
	4194356	0x21000004	addi \$t6, \$t6, 4	31: addi \$t0, \$t0, 4 # increment address to now-known Fib. number storage
	4194360	0x2b1fffff	addi \$t1, \$t1, -1	32: addi \$t1, \$t1, -1 # decrement loop counter
	4194364	0x1e20ffff	bgtz \$t1, loop	33: bgtz \$t1, loop # repeat while not finished
	4194368	0x3c011001	lui \$t0, 4097	36: la \$t0, fibs # first argument for print (array)

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)
268500992	0x00000001	0x00000001	0x00000002	0x00000003
268501024	0x00000000	0x00000000	0x00000000	0x00000000
268501056	0x00000000	0x00000000	0x00000000	0x00000003
268501088	0x4e424943	0x4e424943	0x4277372	0x4547204f
268501120	0x4e424943	0x4e424943	0x4e424943	0x4e424943
268501152	0x4e424943	0x00000000	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
268501440	0x00000000	0x00000000	0x00000000	0x00000000

Registers

Register	Name	Number	Value
\$zero		0	0x00000000
\$at		1	0x00000000
\$v0		2	0x00000000
\$v1		3	0x00000000
\$a0		4	0x00000000
\$a1		5	0x00000000
\$a2		6	0x00000000
\$a3		7	0x00000000
\$t0		8	0x00000000
\$t1		9	0x00000000
\$t2		10	0x00000000
\$t3		11	0x00000000
\$t4		12	0x00000000
\$t5		13	0x00000000
\$t6		14	0x00000000
\$t7		15	0x00000000
\$t8		16	0x00000000
\$t9		17	0x00000000
\$s0		18	0x00000000
\$s1		19	0x00000000
\$s2		20	0x00000000
\$s3		21	0x00000000
\$s4		22	0x00000000
\$s5		23	0x00000000
\$s6		24	0x00000000
\$s7		25	0x00000000
\$s8		26	0x00000000
\$s9		27	0x00000000
\$k0		28	0x00000000
\$k1		29	0x00000000
\$k2		30	0x00000000
\$k3		31	0x00000000

- *e.*

C:\Users\63930\OneDrive\Lab 1\Fibonacci.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Test Segment

Reg	Address	Code	Basic	Source
	4194304	0x30c11001	lui \$1,4097	
	4194308	0x43400000	ori \$1,0	
	4194312	0x0a300000	lui \$2,4097	
	4194316	0x43400000	ori \$2,0	
	4194320	0x0a500000	lw \$3,0(\$0)	# load array elem
	4194324	0x41200010	addi \$1,\$0,1	# \$1 is the known value of first and second Fib. number
	4194328	0x0a200000	sw \$1,0(\$0)	# F(0) = 1
	4194332	0x0a200000	sw \$1,0(\$0)	# F(1) = F(0) = 1
	4194336	0x0a200000	sw \$1,0(\$0)	# Counter for loop, will execute (size-2) times
	4194340	0x0a200000	sw \$1,0(\$0)	# Get value from array F[n-2]
	4194344	0x0a200000	sw \$1,0(\$0)	# Get value from array F[n-1]
	4194348	0x0a200000	sw \$1,0(\$0)	# F[n] = F[n-1] + F[n-2]
	4194352	0x0a200000	sw \$1,0(\$0)	# Store newly computed Fib in array
	4194356	0x0a200000	sw \$1,0(\$0)	# Increment address to now-known Fib. number storage
	4194360	0x0a200000	sw \$1,0(\$0)	# Decrement loop counter
	4194364	0x0a200000	sw \$1,0(\$0)	# repeat while not finished
	4194368	0x0a200000	sw \$1,0(\$0)	# First argument for print (array)

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	
\$a0	1	26550992
\$a1	2	0
\$a2	3	0
\$a3	4	0
\$a4	5	19
\$a5	6	0
\$a6	7	0
\$a7	8	26550996
\$t0	9	18
\$t1	10	0
\$t2	11	0
\$t3	12	0
\$t4	13	0
\$t5	14	0
\$t6	15	0
\$t7	16	0
\$t8	17	0
\$t9	18	26551000
\$s0	19	4194304
\$s1	20	1947
\$s2	21	0
\$s3	22	2594
\$s4	23	0
\$s5	24	0
\$s6	25	0
\$s7	26	0
\$s8	27	0
\$s9	28	0
\$t0	29	0
\$t1	30	0
\$t2	31	0
\$t3	32	0
\$t4	33	0
\$t5	34	0
\$t6	35	0
\$t7	36	0
\$t8	37	0
\$t9	38	0
\$s0	39	0
\$s1	40	0
\$s2	41	0
\$s3	42	0
\$s4	43	0
\$s5	44	0
\$s6	45	0
\$s7	46	0
\$s8	47	0
\$s9	48	0
\$t0	49	0
\$t1	50	0
\$t2	51	0
\$t3	52	0
\$t4	53	0
\$t5	54	0
\$t6	55	0
\$t7	56	0
\$t8	57	0
\$t9	58	0
\$s0	59	0
\$s1	60	0
\$s2	61	0
\$s3	62	0
\$s4	63	0
\$s5	64	0
\$s6	65	0
\$s7	66	0
\$s8	67	0
\$s9	68	0
\$t0	69	0
\$t1	70	0
\$t2	71	0
\$t3	72	0
\$t4	73	0
\$t5	74	0
\$t6	75	0
\$t7	76	0
\$t8	77	0
\$t9	78	0
\$s0	79	0
\$s1	80	0
\$s2	81	0
\$s3	82	0
\$s4	83	0
\$s5	84	0
\$s6	85	0
\$s7	86	0
\$s8	87	0
\$s9	88	0
\$t0	89	0
\$t1	90	0
\$t2	91	0
\$t3	92	0
\$t4	93	0
\$t5	94	0
\$t6	95	0
\$t7	96	0
\$t8	97	0
\$t9	98	0
\$s0	99	0
\$s1	100	0
\$s2	101	0
\$s3	102	0
\$s4	103	0
\$s5	104	0
\$s6	105	0
\$s7	106	0
\$s8	107	0
\$s9	108	0
\$t0	109	0
\$t1	110</	

- 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 (fibs[2]).
 - Now, in line 31, \$s0 points to fibs[1].

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

• *f.*

CLib63930.COE181.TVFibonacci.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Text Segment

Inst	Address	Code	Basic
	0x00400000	0x00400000	7: 1a \$a0, fibs # load address of array
	0x00400004	0x00400000	8: 1a \$a0, size # load address of size variable
	0x00400008	0x00400000	9: 1w \$a0, 0(\$a0) # load array size
	0x0040000c	0x00400000	10: 1l \$a0, 1 # 1 is the known value of first
	0x00400010	0x00400000	11: mv \$a0, 0(\$a0) # F[0] = 1
	0x00400014	0x00400000	12: mv \$a0, 4(\$a0) # F[1] = F[0] = 1
	0x00400018	0x00400000	13: addi \$a1, \$a0, -2 # Counter for loop, will count
	0x0040001c	0x00400000	14: loopz lw \$a1, 0(\$a0) # Get value from array F[n-2]
	0x00400020	0x00400000	15: lw \$a2, 4(\$a0) # Get value from array F[n-1]
	0x00400024	0x00400000	16: add \$a2, \$a1, \$a2 # F[n] = F[n-1] + F[n-2]
	0x00400028	0x00400000	17: mv \$a0, 5(\$a0) # Store newly computed F[n] in
	0x0040002c	0x00400000	18: addi \$a0, \$a0, 4 # Increment address to now-know
	0x00400030	0x00400000	19: addi \$a1, \$a1, -1 # Decrement loop counter
	0x00400034	0x00400000	20: bnez \$a1, loop # Repeat while not finished
	0x00400038	0x00400000	21: 1a \$a0, fibs # First argument for print (a0)

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+C)
0x00100000	0x00000001	0x00000001	0x00000002	0x00000003
0x00100004	0x00000022	0x00000037	0x00000059	0x00000090
0x00100008	0x00000054	0x00000085	0x000000C9	0x00000134
0x0010000c	0x00000093	0x000000D9	0x00000134	0x00000195
0x00100010	0x000000D9	0x00000134	0x00000195	0x000001F6
0x00100014	0x00000134	0x00000195	0x000001F6	0x00000257
0x00100018	0x00000195	0x000001F6	0x00000257	0x000002B8
0x0010001c	0x000001F6	0x00000257	0x000002B8	0x00000319
0x00100020	0x00000257	0x000002B8	0x00000319	0x0000039A
0x00100024	0x000002B8	0x00000319	0x0000039A	0x000003FB
0x00100028	0x00000319	0x0000039A	0x000003FB	0x0000047C
0x0010002c	0x0000039A	0x000003FB	0x0000047C	0x000004DD
0x00100030	0x000003FB	0x0000047C	0x000004DD	0x0000053E
0x00100034	0x0000047C	0x000004DD	0x0000053E	0x0000059F
0x00100038	0x000004DD	0x0000053E	0x0000059F	0x000005FF
0x0010003c	0x0000053E	0x0000059F	0x000005FF	0x00000660
0x00100040	0x0000059F	0x000005FF	0x00000660	0x000006C1
0x00100044	0x000005FF	0x00000660	0x000006C1	0x00000722
0x00100048	0x00000660	0x000006C1	0x00000722	0x00000783
0x0010004c	0x000006C1	0x00000722	0x00000783	0x000007E4
0x00100050	0x00000722	0x00000783	0x000007E4	0x00000845
0x00100054	0x00000783	0x000007E4	0x00000845	0x000008A6
0x00100058	0x000007E4	0x00000845	0x000008A6	0x00000907
0x0010005c	0x00000845	0x000008A6	0x00000907	0x00000968
0x00100060	0x000008A6	0x00000907	0x00000968	0x000009C9
0x00100064	0x00000907	0x00000968	0x000009C9	0x00000A2A
0x00100068	0x00000968	0x000009C9	0x00000A2A	0x00000A8B
0x0010006c	0x000009C9	0x00000A2A	0x00000A8B	0x00000AEC
0x00100070	0x00000A2A	0x00000A8B	0x00000AEC	0x00000B4D
0x00100074	0x00000A8B	0x00000AEC	0x00000B4D	0x00000BAE
0x00100078	0x00000AEC	0x00000B4D	0x00000BAE	0x00000C0F
0x0010007c	0x00000B4D	0x00000BAE	0x00000C0F	0x00000C60
0x00100080	0x00000BAE	0x00000C0F	0x00000C60	0x00000CC1
0x00100084	0x00000C0F	0x00000C60	0x00000CC1	0x00000D22
0x00100088	0x00000C60	0x00000CC1	0x00000D22	0x00000D83
0x0010008c	0x00000CC1	0x00000D22	0x00000D83	0x00000DE4
0x00100090	0x00000D22	0x00000D83	0x00000DE4	0x00000E45
0x00100094	0x00000D83	0x00000DE4	0x00000E45	0x00000EA6
0x00100098	0x00000DE4	0x00000E45	0x00000EA6	0x00000F07
0x0010009c	0x00000E45	0x00000EA6	0x00000F07	0x00000F68
0x001000a0	0x00000EA6	0x00000F07	0x00000F68	0x00000FC9
0x001000a4	0x00000F07	0x00000F68	0x00000FC9	0x0000102A
0x001000a8	0x00000F68	0x00000FC9	0x0000102A	0x0000108B
0x001000ac	0x00000FC9	0x0000102A	0x0000108B	0x000010EC
0x001000b0	0x0000102A	0x0000108B	0x000010EC	0x0000114D
0x001000b4	0x0000108B	0x000010EC	0x0000114D	0x000011AE
0x001000b8	0x000010EC	0x0000114D	0x000011AE	0x0000120F
0x001000bc	0x0000114D	0x000011AE	0x0000120F	0x00001260
0x001000c0	0x000011AE	0x0000120F	0x00001260	0x000012C1
0x001000c4	0x0000120F	0x00001260	0x000012C1	0x00001322
0x001000c8	0x00001260	0x000012C1	0x00001322	0x00001383
0x001000cc	0x000012C1	0x00001322	0x00001383	0x000013E4
0x001000d0	0x00001322	0x00001383	0x000013E4	0x00001445
0x001000d4	0x00001383	0x000013E4	0x00001445	0x000014A6
0x001000d8	0x000013E4	0x00001445	0x000014A6	0x00001507
0x001000dc	0x00001445	0x000014A6	0x00001507	0x00001568
0x001000e0	0x000014A6	0x00001507	0x00001568	0x000015C9
0x001000e4	0x00001507	0x00001568	0x000015C9	0x0000162A
0x001000e8	0x00001568	0x000015C9	0x0000162A	0x0000168B
0x001000ec	0x000015C9	0x0000162A	0x0000168B	0x000016EC
0x001000f0	0x0000162A	0x0000168B	0x000016EC	0x0000174D
0x001000f4	0x0000168B	0x000016EC	0x0000174D	0x000017AE
0x001000f8	0x000016EC	0x0000174D	0x000017AE	0x0000180F
0x001000fc	0x0000174D	0x000017AE	0x0000180F	0x00001860
0x00100100	0x000017AE	0x0000180F	0x00001860	0x000018C1
0x00100104	0x0000180F	0x00001860	0x000018C1	0x00001922
0x00100108	0x00001860	0x000018C1	0x00001922	0x00001983
0x0010010c	0x000018C1	0x00001922	0x00001983	0x000019E4
0x00100110	0x00001922	0x00001983	0x000019E4	0x00001A45
0x00100114	0x00001983	0x000019E4	0x00001A45	0x00001AA6
0x00100118	0x000019E4	0x00001A45	0x00001AA6	0x00001B07
0x0010011c	0x00001A45	0x00001AA6	0x00001B07	0x00001B68
0x00100120	0x00001AA6	0x00001B07	0x00001B68	0x00001BC9
0x00100124	0x00001B07	0x00001B68	0x00001BC9	0x00001C2A
0x00100128	0x00001B68	0x00001BC9	0x00001C2A	0x00001C8B
0x0010012c	0x00001BC9	0x00001C2A	0x00001C8B	0x00001CEC
0x00100130	0x00001C2A	0x00001C8B	0x00001CEC	0x00001D4D
0x00100134	0x00001C8B	0x00001CEC	0x00001D4D	0x00001DAE
0x00100138	0x00001CEC	0x00001D4D	0x00001DAE	0x00001E0F
0x0010013c	0x00001D4D	0x00001DAE	0x00001E0F	0x00001E60
0x00100140	0x00001DAE	0x00001E0F	0x00001E60	0x00001EC1
0x00100144	0x00001E0F	0x00001E60	0x00001EC1	0x00001F22
0x00100148	0x00001E60	0x00001EC1	0x00001F22	0x00001F83
0x0010014c	0x00001EC1	0x00001F22	0x00001F83	0x00001FE4
0x00100150	0x00001F22	0x00001F83	0x00001FE4	0x00002045
0x00100154	0x00001F83	0x00001FE4	0x00002045	0x000020A6
0x00100158	0x00001FE4	0x00002045	0x000020A6	0x00002107
0x0010015c	0x00002045	0x000020A6	0x00002107	0x00002168
0x00100160	0x000020A6	0x00002107	0x00002168	0x000021C9
0x00100164	0x00002107	0x00002168	0x000021C9	0x0000222A
0x00100168	0x00002168	0x000021C9	0x0000222A	0x0000228B
0x0010016c	0x000021C9	0x0000222A	0x0000228B	0x000022EC
0x00100170	0x0000222A	0x0000228B	0x000022EC	0x0000234D
0x00100174	0x0000228B	0x000022EC	0x0000234D	0x000023AE
0x00100178	0x000022EC	0x0000234D	0x000023AE	0x0000240F
0x0010017c	0x0000234D	0x000023AE	0x0000240F	0x00002460
0x00100180	0x000023AE	0x0000240F	0x00002460	0x000024C1
0x00100184	0x0000240F	0x00002460	0x000024C1	0x00002522
0x00100188	0x00002460	0x000024C1	0x00002522	0x00002583
0x0010018c	0x000024C1	0x00002522	0x00002583	0x000025E4
0x00100190	0x00002522	0x00002583	0x000025E4	0x00002645
0x00100194	0x00002583	0x000025E4	0x00002645	0x000026A6
0x00100198	0x000025E4	0x00002645	0x000026A6	0x00002707
0x0010019c	0x00002645	0x000026A6	0x00002707	0x00002768
0x001001a0	0x000026A6	0x00002707	0x00002768	0x000027C9
0x001001a4	0x00002707	0x00002768	0x000027C9	0x0000282A
0x001001a8	0x00002768	0x000027C9	0x0000282A	0x0000288B
0x001001ac	0x000027C9	0x0000282A	0x0000288B	0x000028EC
0x001001b0	0x0000282A	0x0000288B	0x000028EC	0x0000294D
0x001001b4	0x0000288B	0x000028EC	0x0000294D	0x000029AE
0x001001b8	0x000028EC	0x0000294D	0x000029AE	0x00002A0F
0x001001bc	0x0000294D	0x000029AE	0x00002A0F	0x00002A60
0x001001c0	0x000029AE	0x00002A0F	0x00002A60	0x00002AC1
0x001001c4	0x00002A0F	0x00002A60	0x00002AC1	0x00002B22
0x001001c8	0x00002A60	0x00002AC1	0x00002B22	0x00002B83
0x001001cc	0x00002AC1	0x00002B22	0x00002B83	0x00002BE4
0x001001d0	0x00002B22	0x00002B83	0x00002BE4	0x00002C45
0x001001d4	0x00002B83	0x00002BE4	0x00002C45	0x00002CA6
0x001001d8	0x00002BE4	0x00002C45	0x00002CA6	0x00002D07
0x001001dc	0x00002C45	0x00002CA6	0x00002D07	0x00002D68
0x001001e0	0x00002CA6	0x00002D07	0x00002D68	0x00002DC9
0x001001e4	0x00002D07	0x00002D68	0x00002DC9	0x00002E2A
0x001001e8	0x00002D68	0x00002DC9	0x00002E2A	0x00002E8B
0x001001ec	0x00002DC9	0x00002E2A	0x00002E8B	0x00002EEC
0x001001f0	0x00002E2A	0x00002E8B	0x00002EEC	0x00002F4D
0x001001f4	0x00002E8B	0x00002EEC	0x00002F4D	0x00002FAE
0x001001f8	0x00002EEC	0x00002F4D	0x00002FAE	0x0000300F
0x001001fc	0x00002F4D	0x00002FAE	0x0000300F	0x00003060
0x00100200	0x00002FAE	0x0000300F	0x00003060	0x000030C1
0x00100204	0x0000300F	0x00003060	0x000030C1	0x00003122
0x00100208	0x00003060	0x000030C1	0x00003122	0x00003183
0x0010020c	0x000030C1	0x00003122	0x00003183	0x000031E4
0x00100210	0x00003122	0x00003183	0x000031E4	0x00003245
0x00100214	0x00003183	0x000031E4	0x00003245	0x000032AE
0x00100218	0x000031E4	0x00003245	0x000032AE	0x0000330F
0x0010021c	0x00003245	0x000032AE	0x0000330F	0x00003360
0x00100220	0x000032AE	0x0000330F	0x00003360	0x000033C1
0x00100224	0x0000330F	0x00003360	0x000033C1	0x00003422
0x00100228	0x00003360	0x000033C1	0x00003422	0x00003483
0x0010022c	0x000033C1	0x00003422	0x00003483	0x000034E4
0x00100230	0x00003422	0x00003483	0x000034E4	0x00003545
0x00100234	0x00003483	0x000034E4	0x00003545	0x000035AE
0x00100238	0x000034E4	0x00003545	0x000035AE	0x0000360F
0x0010023c	0x			

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

C:\Users\63930\COE181.1\Lab 1\Fibonacci.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	4194376	0x00152820	add \$5,\$0,\$21	37: add \$a1,\$zero,\$s5 # second argument for print (size)
	4194380	0x0c100016	jal 4194392	38: jal print # call print routine.
	4194384	0x2402000a	addiu \$2,\$0,10	41: li \$v0,10 # system call for exit
	4194388	0x0000000c	syscall	42: syscall # Exit!
	4194392	0x00044020	add \$8,\$0,\$4	50: print: add \$t0,\$zero,\$a0 # starting address of array of data to be printed
	4194396	0x00054820	add \$9,\$0,\$5	51: add \$t1,\$zero,\$a1 # initialize loop counter to array size
	4194400	0x3c011001	lui \$1,4097	52: la \$a0,head # load address of the print heading string
	4194404	0x3424000a	ori \$4,\$1,138	
	4194408	0x24020004	addiu \$2,\$0,4	53: li \$v0,4 # specify Print String service
	4194412	0x0000000c	syscall	54: syscall # print the heading string
	4194416	0x8d040000	lw \$4,0(\$8)	56: out: lw \$a0,0(\$t0) # load the integer to be printed (the current Fib. number)
	4194420	0x24020001	addiu \$2,\$0,1	57: li \$v0,1 # specify Print Integer service
	4194424	0x0000000c	syscall	58: syscall # print fibonacci number
	4194428	0x3c011001	lui \$1,4097	60: la \$a0,space # load address of spacer for syscall
	4194432	0x34240007	ori \$4,\$1,135	
	4194436	0x24020004	addiu \$2,\$0,4	61: li \$v0,4 # specify Print String service
	4194440	0x0000000c	syscall	62: syscall # print the spacer string

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	1	1	2	3	5	8	13	21
268501024	34	55	89	144	233	377	610	987
268501056	1597	2584	4181	15	544698184	2037277037	1651066400	1667329647
268501088	1847617891	1700949365	1948283762	1701257327	1634887022	541025652	1008742952	544743485
268501120	824196412	167782713	1750335520	1766203493	1634627426	543777635	1651340654	544436837
268501152	979726945	10	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0
268501248	0	0	0	0	0	0	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0
268501440	0	0	0	0	0	0	0	0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Mars Messages Run IO

The Fibonacci numbers are:
1

Reset: reset completed.

The Fibonacci numbers are:

Clear

- 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.



Run speed a

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	
<input type="checkbox"/>	4194304	0x3c011001	lui \$1,0x00001001	7: la \$s0, fibs # load address of array
<input type="checkbox"/>	4194308	0x34300000	ori \$16,\$1,0x00000000	
<input type="checkbox"/>	4194312	0x3c011001	lui \$1,0x00001001	8: la \$s5, size # load address of size variable
<input type="checkbox"/>	4194316	0x3435004c	ori \$21,\$1,0x0000004c	
<input type="checkbox"/>	4194320	0x8eb50000	lw \$21,0x00000000(\$21)	9: lw \$s5, 0(\$s5) # load array size
<input type="checkbox"/>	4194324	0x24120001	addiu \$18,\$0,0x0000...	21: li \$s2, 1 # 1 is the known value of first
<input type="checkbox"/>	4194328	0xae120000	sw \$18,0x00000000(\$16)	22: sw \$s2, 0(\$s0) # F[0] = 1
<input type="checkbox"/>	4194332	0xae120004	sw \$18,0x00000004(\$16)	23: sw \$s2, 4(\$s0) # F[1] = F[0] = 1
<input type="checkbox"/>	4194336	0x22b1ffff	addi \$17,\$21,0xffff...	24: addi \$s1, \$s5, -2 # Counter for loop, will execut
<input type="checkbox"/>	4194340	0x8e130000	lw \$19,0x00000000(\$16)	27: loop: lw \$s3, 0(\$s0) # Get value from array F[n-2]
<input type="checkbox"/>	4194344	0x8e140004	lw \$20,0x00000004(\$16)	28: lw \$s4, 4(\$s0) # Get value from array F[n-1]
<input type="checkbox"/>	4194348	0x02749020	add \$18,\$19,\$20	29: add \$s2, \$s3, \$s4 # F[n] = F[n-1] + F[n-2]
<input type="checkbox"/>	4194352	0xae120008	sw \$18,0x00000008(\$16)	30: sw \$s2, 8(\$s0) # Store newly computed F[n] in
<input type="checkbox"/>	4194356	0x22100004	addi \$16,\$16,0x0000...	31: addi \$s0, \$s0, 4 # increment address to now-know
<input type="checkbox"/>	4194360	0x2231ffff	addi \$17,\$17,0xffff...	32: addi \$s1, \$s1, -1 # decrement loop counter
<input type="checkbox"/>	4194364	0x1e20fff9	bgtz \$17,0xfffffff9	33: bgtz \$s1, loop # repeat while not finished
<input type="checkbox"/>	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	0x00000063d	0x000000a18	0x000001055	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
268501440	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.