Solution: HW#5 CSC390

HW#5_Q1:

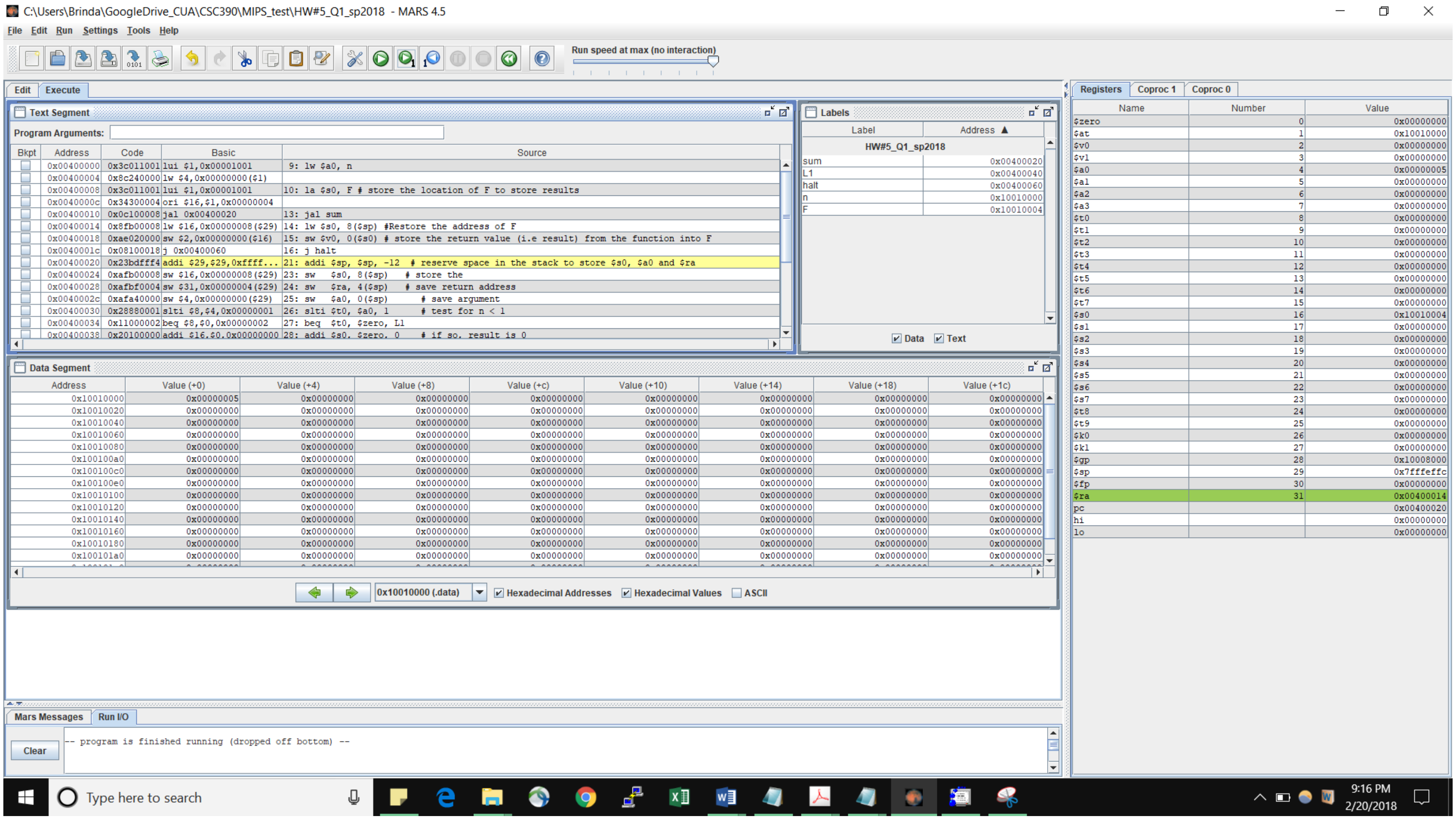
```
# Procedure Example (Nested Procedure)
# Call a function to perform n+(n-1)
.data
n: .word 5
F: .word  0 # store the result
.text
# put the data in the argument register.
# arguments resisters are used to pass-parameters in the procedure (function)
lw $a0, n
la $s0, F # store the location of F to store results

# Call the sum nested_procedure
jal sum
lw $s0, 8($sp) #Restore the address of F
sw $v0, 0($s0) # store the return value (i.e result) from the function into F
j halt

#the nested_procedure (sum)

```

```
20  sum:
21  addi $sp, $sp, -12  # reserve space in the stack to store $s0, $a0 and $ra
22                      # adjust stack for 3 items
23  sw   $s0, 8($sp)    # store the
24  sw   $ra, 4($sp)    # save return address
25  sw   $a0, 0($sp)        # save argument
26  slti $t0, $a0, 1        # test for n < 1
27  beq  $t0, $zero, L1
28  addi $s0, $zero, 0      # if so, result is 0
29  jr   $ra                #   return to line 32
30  L1: addi $a0, $a0, -1       # else decrement n
31  jal  sum          # recursive call
32  addi $sp, $sp, 12       # Increment stack by 12
33  lw   $ra, 4($sp)        # Restore $ra
34  lw   $a0, 0($sp)        # restore $a0
35  addu $s0, $a0, $s0      # Add to get result
36  add  $v0, $s0, $zero    # Copy the result to $V0
37
38  jr   $ra                # and return
39  halt:
40  nop
```

Word version of the code is also attached below:

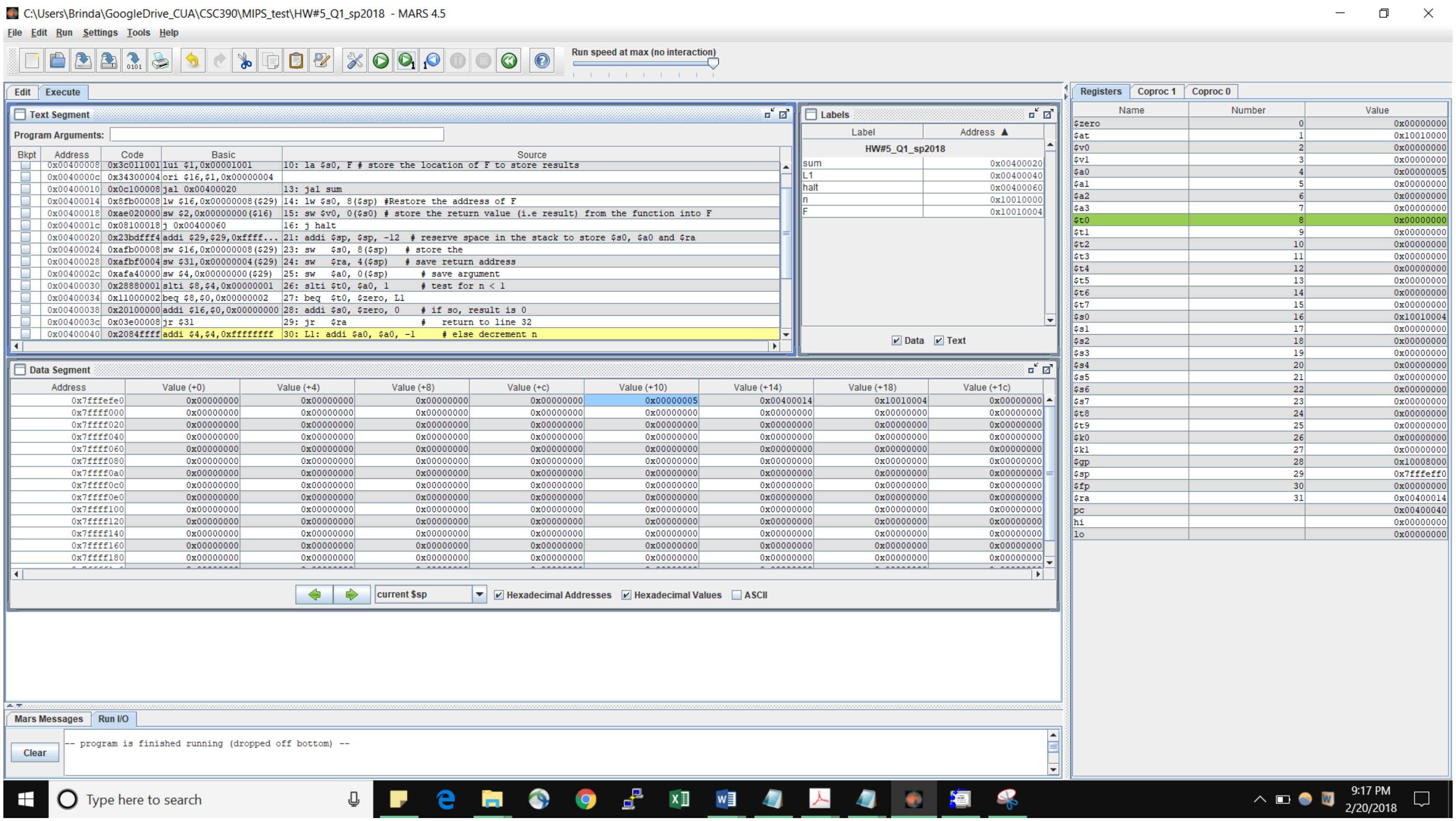
```
# Call a function to perform n+(n-1)
.data
n: .word 5
F: .word  0 # store the result
.text
# put the data in the argument register.
# arguments resisters are used to pass-parameters in the procedure (function)
lw $a0, n
la $s0, F # store the location of F to store results

# Call the sum nested_procedure
jal sum
lw $s0, 8($sp) #Restore the address of F
sw $v0, 0($s0) # store the return value (i.e result) from the function into F
j halt
#the nested_procedure (sum)
sum:
addi $sp, $sp, -12  # reserve space in the stack to store $s0, $a0 and $ra
                    # adjust stack for 3 items
sw   $s0, 8($sp)   # store the
sw   $ra, 4($sp)   # save return address
sw   $a0, 0($sp)      # save argument
slti $t0, $a0, 1      # test for n < 1
beq  $t0, $zero, L1
addi $s0, $zero, 0    # if so, result is 0
jr   $ra              #   return to line 32
L1: addi $a0, $a0, -1     # else decrement n
jal  sum       # recursive call
addi $sp, $sp, 12     # Increment stack by 12
lw   $ra, 4($sp)      # Restore $ra
lw   $a0, 0($sp)      # restore $a0
addu $s0, $a0, $s0    # Add to get result
add  $v0, $s0, $zero  # Copy the result to $V0
jr   $ra              # and return
halt:
nop
```

Sum is called for the first time:

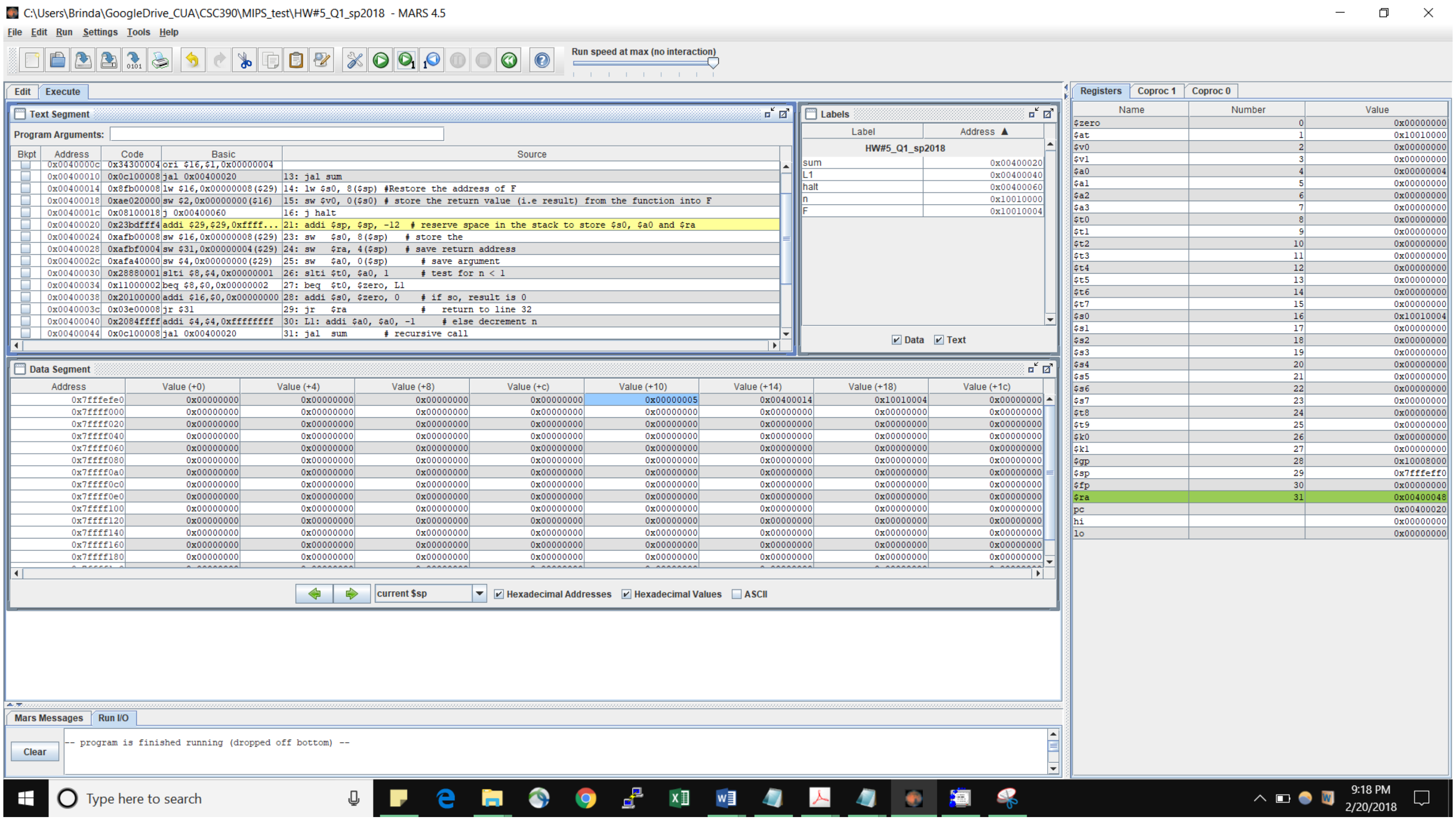


$pc has the current location, $ra contains the return address to the main program and $sp has the highest location of the memory.

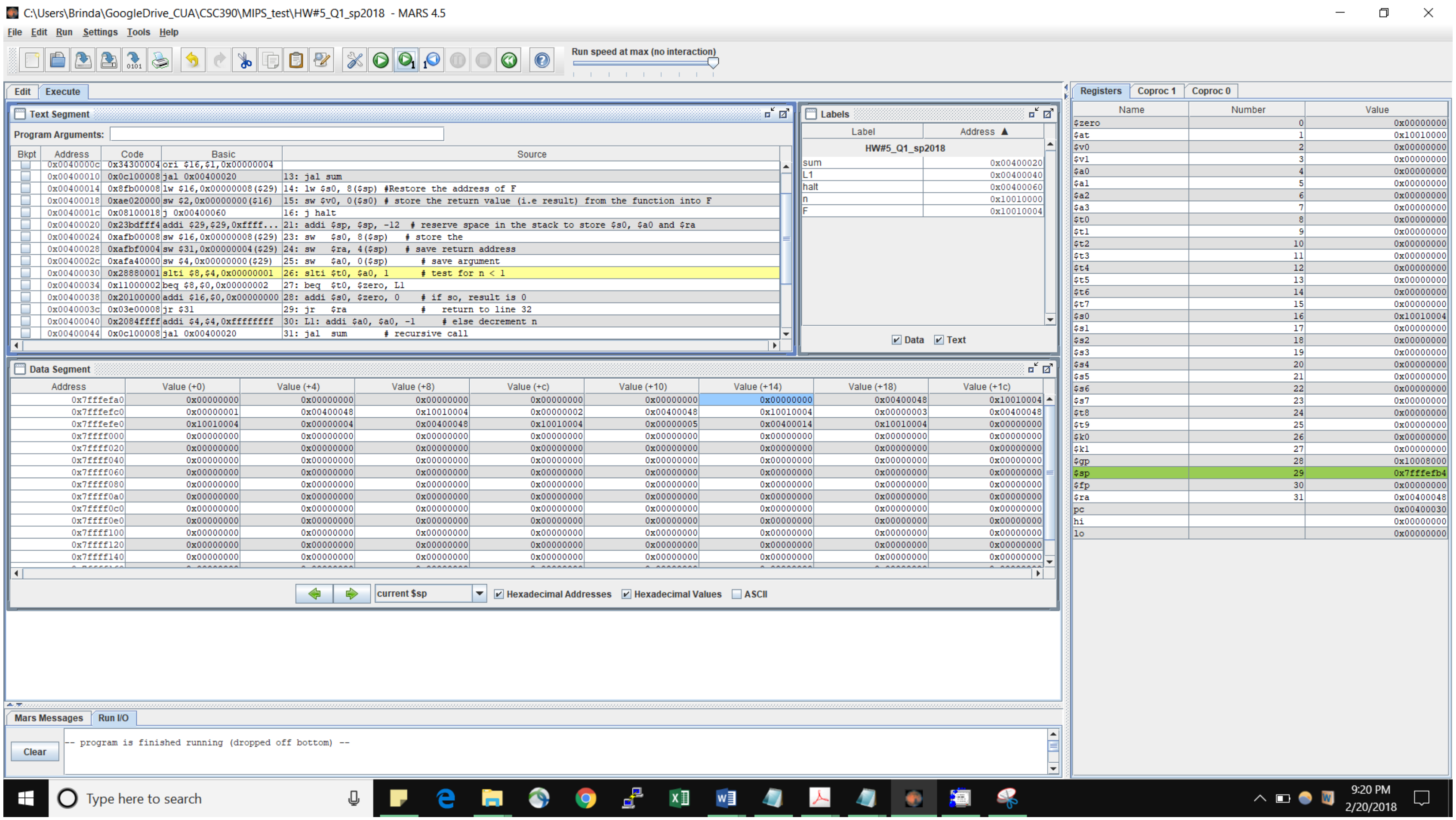
During the execution of sum procedure (1st iteration), the stack memory location is loaded the values of return address ($ra) to the main program and the current valu of n ($a0).



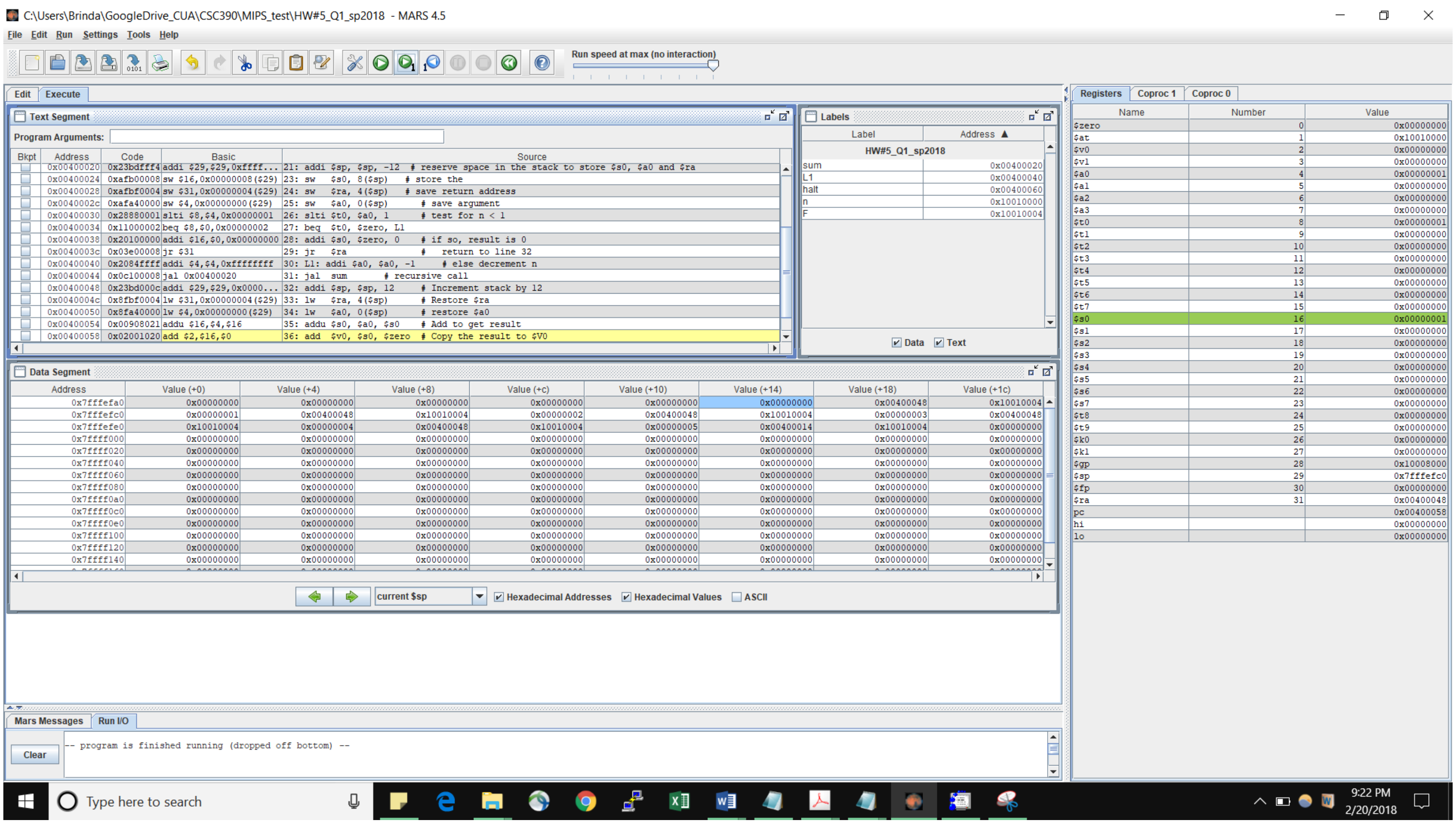
2nd Iteration: $ra has now the return address to the sum procedure location (recursive operation). $sp has the value decremented by 12.



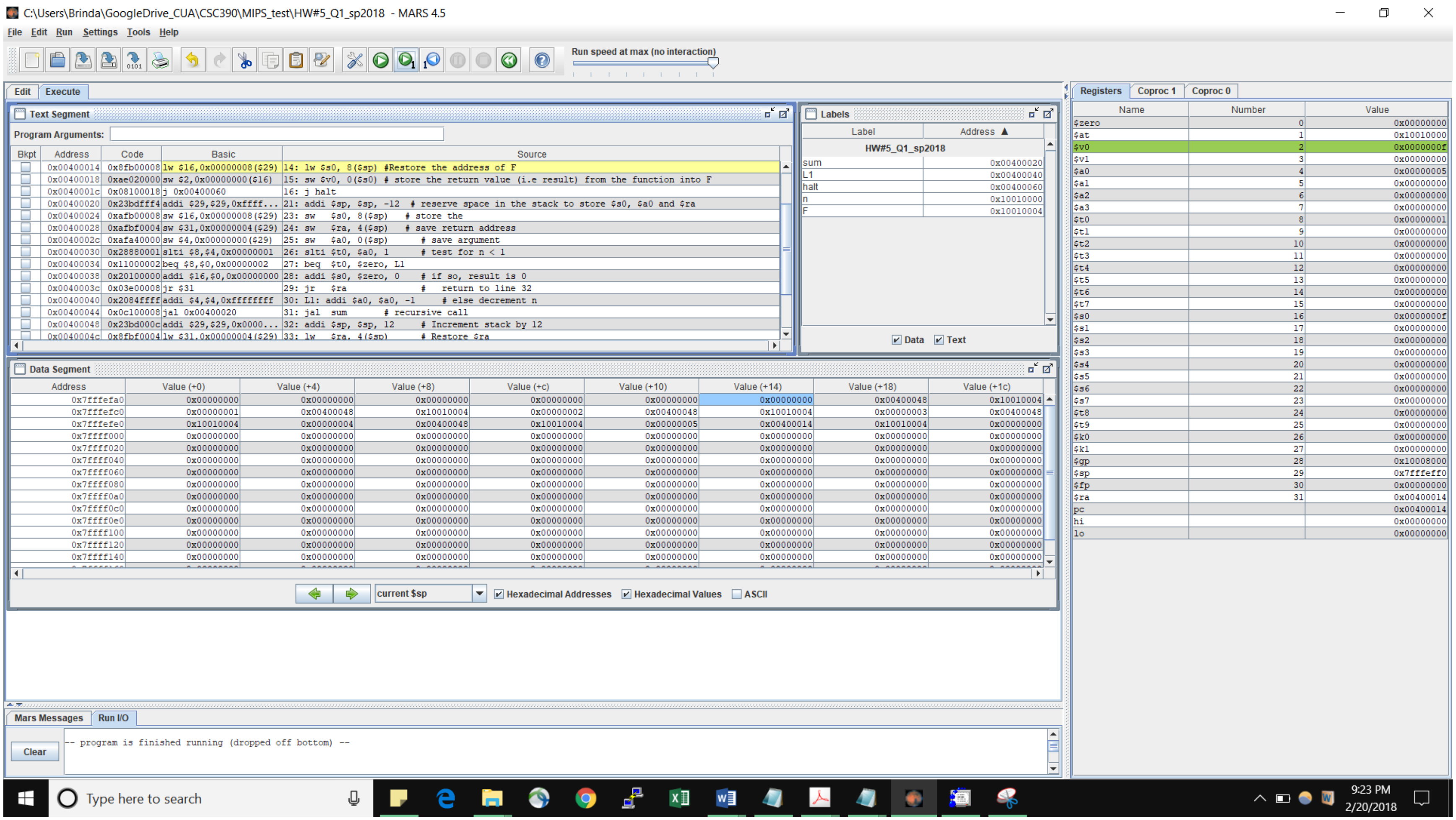
After 5 iterations: the stack memory locations showing the return addresses and the decremented values of n.



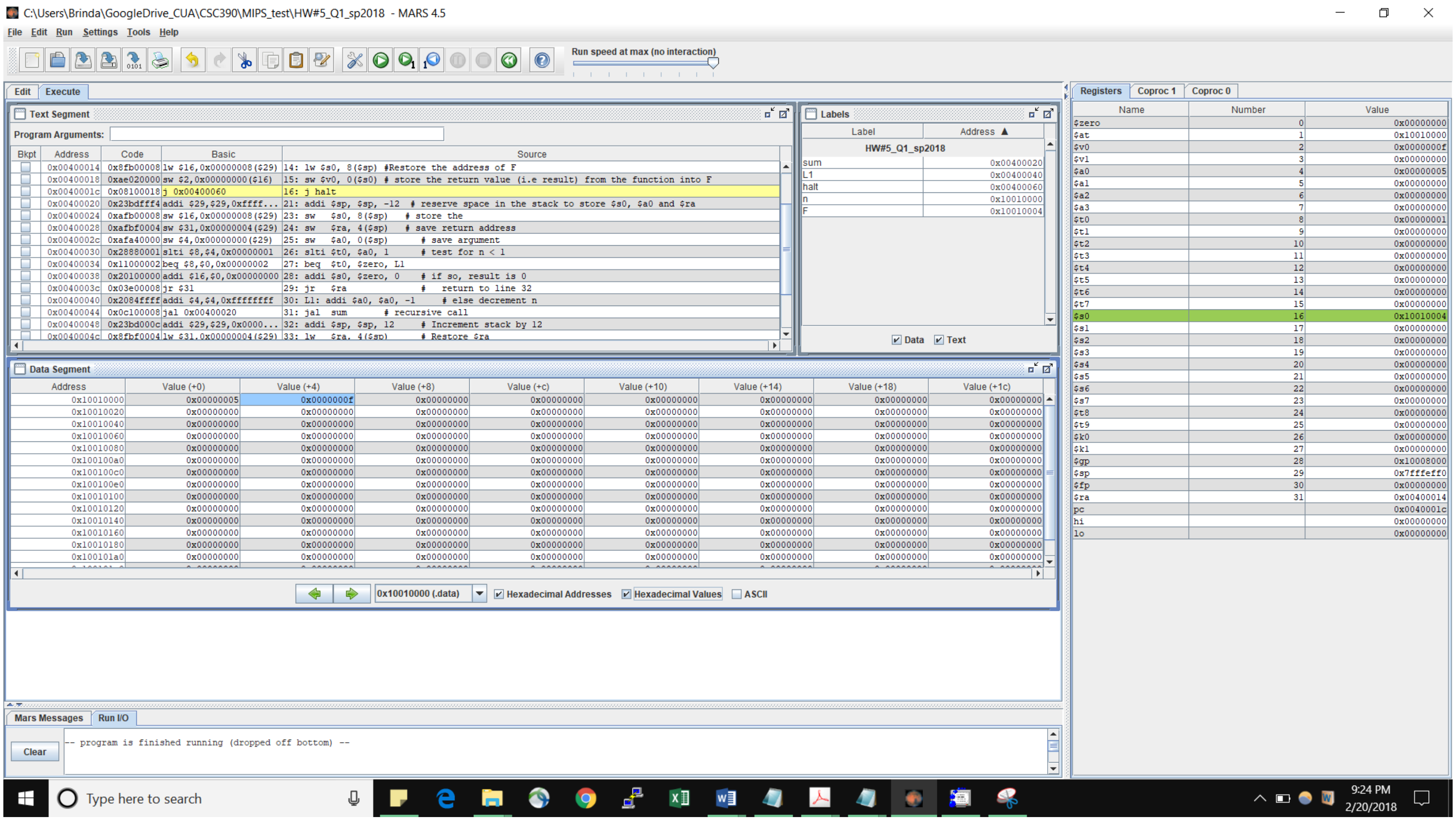
Values of the stack memory location when n decremented to 0



Contents of $ra, $pc, and $sp before returning to the main program:



After result is stored into the memory location of F. check that the memory location indicated by $S0 has the correct results 15 (i.e. f in Hex).



C:\Users\Brinda\GoogleDrive_CUA\CSC390\MIPS_test\HW#5_Q1_sp2018 - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Program Arguments:
Bkpt Address Code Basic Source
0x00400014 0x8fb00008 lw $16,0x00000008($29) 14: lw $s0, 8($sp) #Restore the address of F
0x00400018 0xae020000 sw $2,0x00000000($16) 15: sw $v0, 0($s0) # store the return value (i.e result) from the function into F
0x0040001c 0x08100018 j 0x00400060 16: j halt
0x00400020 0x23bdfff4 addi $29,$29,0xffff... 21: addi $sp, $sp, -12 # reserve space in the stack to store $s0, $a0 and $ra
0x00400024 0xafb00008 sw $16,0x00000008($29) 23: sw $s0, 8($sp) # store the
0x00400028 0xafbf0004 sw $31,0x00000004($29) 24: sw $ra, 4($sp) # save return address
0x0040002c 0xafa40000 sw $4,0x00000000($29) 25: sw $a0, 0($sp) # save argument
0x00400030 0x28880001 slti $8,$4,0x00000001 26: slti $t0, $a0, 1 # test for n < 1
0x00400034 0x11000002 beq $8,$0,0x00000002 27: beq $t0, $zero, L1
0x00400038 0x20100000 addi $16,$0,0x00000000 28: addi $s0, $zero, 0 # if so, result is 0
0x0040003c 0x03e00008 jr $31 29: jr $ra # return to line 32
0x00400040 0x2084ffff addi $4,$4,0xffffffff 30: L1: addi $a0, $a0, -1 # else decrement n
0x00400044 0x0c100008 jal 0x00400020 31: jal sum # recursive call
0x00400048 0x23bd000c addi $29,$29,0x0000... 32: addi $sp, $sp, 12 # Increment stack by 12
0x0040004c 0x8fbf0004 lw $31,0x00000004($29) 33: lw $ra, 4($sp) # Restore $ra
Labels
Label Address
HW#5_Q1_sp2018
sum 0x00400020
L1 0x00400040
halt 0x00400060
n 0x10010000
F 0x10010004
Data Text
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 0x00000005 0x0000000f 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010020 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010040 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010060 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010080 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100c0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010100 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010120 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010140 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010160 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010180 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100101a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII
Registers Coproc 1 Coproc 0
Name Number Value
$zero 0 0x00000000
$at 1 0x10010000
$v0 2 0x0000000f
$v1 3 0x00000000
$a0 4 0x00000005
$a1 5 0x00000000
$a2 6 0x00000000
$a3 7 0x00000000
$t0 8 0x00000001
$t1 9 0x00000000
$t2 10 0x00000000
$t3 11 0x00000000
$t4 12 0x00000000
$t5 13 0x00000000
$t6 14 0x00000000
$t7 15 0x00000000
$s0 16 0x10010004
$s1 17 0x00000000
$s2 18 0x00000000
$s3 19 0x00000000
$s4 20 0x00000000
$s5 21 0x00000000
$s6 22 0x00000000
$s7 23 0x00000000
$t8 24 0x00000000
$t9 25 0x00000000
$k0 26 0x00000000
$k1 27 0x00000000
$gp 28 0x10008000
$sp 29 0x7fffeff0
$fp 30 0x00000000
$ra 31 0x00400014
pc 0x0040001c
hi 0x00000000
lo 0x00000000
Mars Messages Run I/O
Clear
-- program is finished running (dropped off bottom) --
Type here to search
9:24 PM
2/20/2018

2.26

2.26.1 20

2.26.2

```
i = 10;
do {
    B += 2;
    i = i - 1;
} while ( i > 0)
```

2.26.3 **5*N+2**