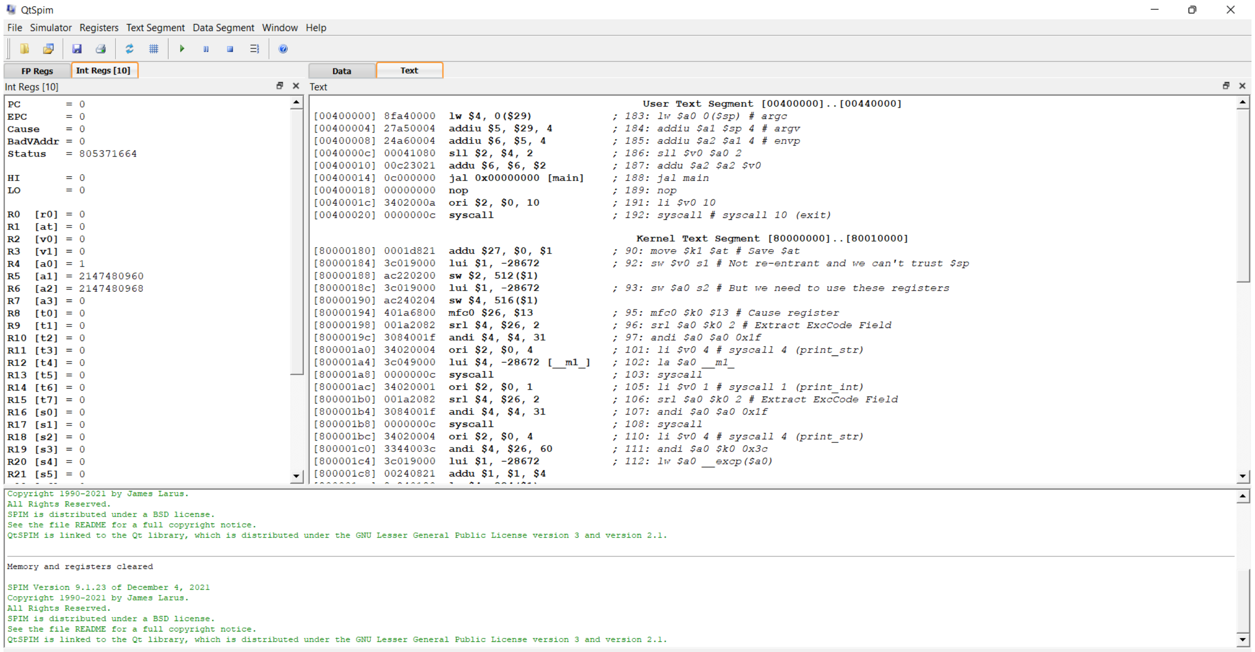
Lab2
CS211

Name-Pranav Tambe
Roll No-2106339


QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [10]
Data
Text

Int Regs [10]

```
PC       = 0
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147480960
R6  [a2] = 2147480968
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
```

Text

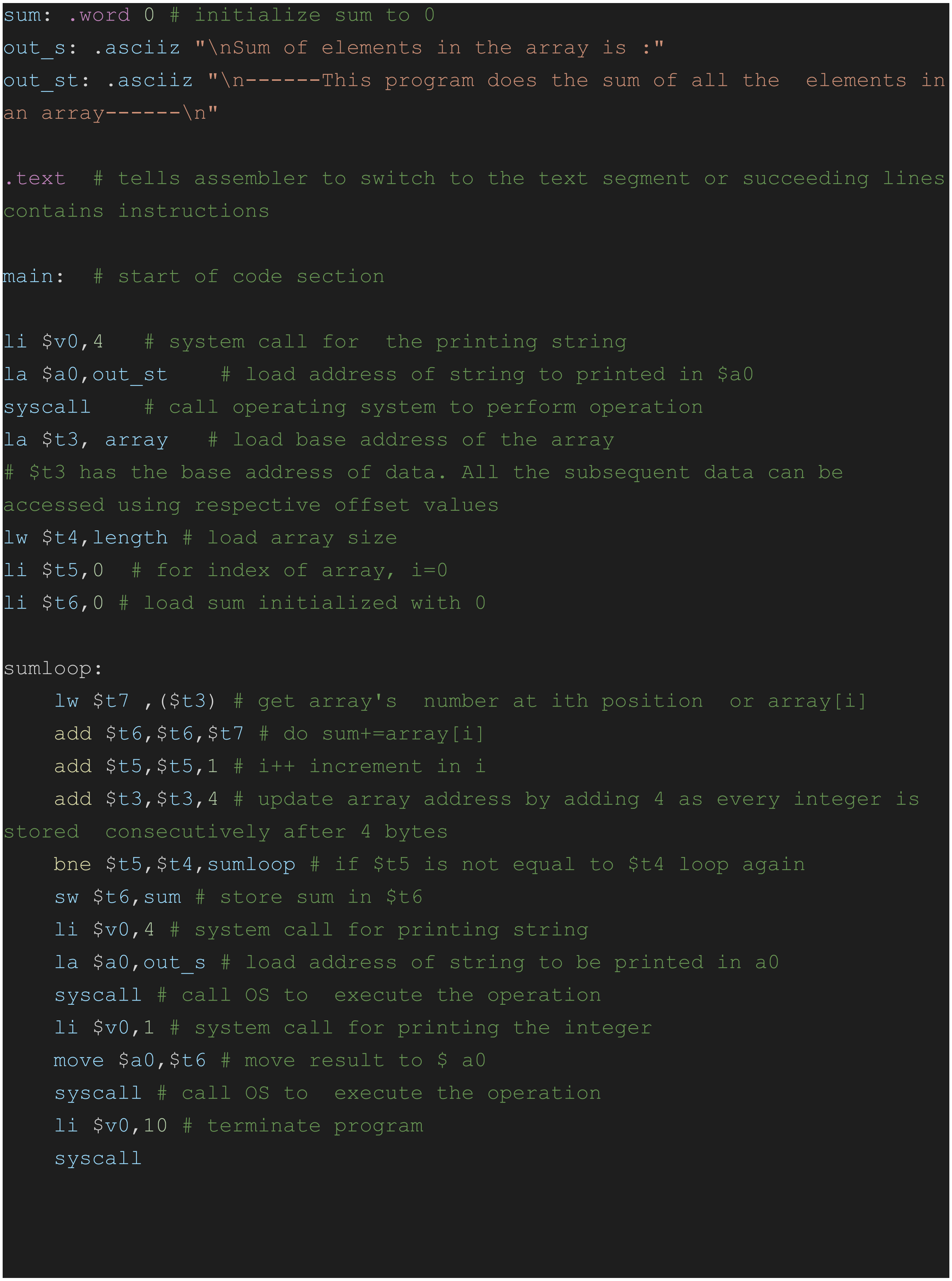
```
                                                  User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)             ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4          ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4           ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2             ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2           ; 187: addu $a2 $a2 $v0
[00400014] 0c000000  jal 0x00000000 [main]     ; 188: jal main
[00400018] 00000000  nop                       ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10            ; 191: li $v0 10
[00400020] 0000000c  syscall                   ; 192: syscall # syscall 10 (exit)

                                                 Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1          ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672            ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672            ; 93: sw $a0 s2 # But we need to use these registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13             ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082  srl $4, $26, 2            ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f  andi $4, $4, 31           ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4             ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000  lui $4, -28672 [__m1_]    ; 102: la $a0 __m1_
[800001a8] 0000000c  syscall                   ; 103: syscall
[800001ac] 34020001  ori $2, $0, 1             ; 105: li $v0 1 # syscall 1 (print_int)
[800001b0] 001a2082  srl $4, $26, 2            ; 106: srl $a0 $k0 2 # Extract ExcCode Field
[800001b4] 3084001f  andi $4, $4, 31           ; 107: andi $a0 $a0 0x1f
[800001b8] 0000000c  syscall                   ; 108: syscall
[800001bc] 34020004  ori $2, $0, 4             ; 110: li $v0 4 # syscall 4 (print_str)
[800001c0] 3344003c  andi $4, $26, 60          ; 111: andi $a0 $k0 0x3c
[800001c4] 3c019000  lui $1, -28672            ; 112: lw $a0 __excp($a0)
[800001c8] 00240821  addu $1, $1, $4
```

```
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

Interface before loading any file

Q.1) Complete the following code snippet to add 10 numbers stored consecutively in data memory.
Print the result

```
.data
array : .word 10, 12, 15, -10, 13, 82, -9, 4, 3, -7  # load the array
length: .word 10 # load the length of the array as 10
```



```
sum: .word 0 # initialize sum to 0
out_s: .asciiz "\nSum of elements in the array is :"
out_st: .asciiz "\n------This program does the sum of all the  elements in
an array------\n"

.text  # tells assembler to switch to the text segment or succeeding lines
contains instructions

main:  # start of code section

li $v0,4   # system call for  the printing string
la $a0,out_st    # load address of string to printed in $a0
syscall    # call operating system to perform operation
la $t3, array   # load base address of the array
# $t3 has the base address of data. All the subsequent data can be
accessed using respective offset values
lw $t4,length # load array size
li $t5,0  # for index of array, i=0
li $t6,0 # load sum initialized with 0

sumloop:
    lw $t7 ,($t3) # get array's  number at ith position  or array[i]
    add $t6,$t6,$t7 # do sum+=array[i]
    add $t5,$t5,1 # i++ increment in i
    add $t3,$t3,4 # update array address by adding 4 as every integer is
stored  consecutively after 4 bytes
    bne $t5,$t4,sumloop # if $t5 is not equal to $t4 loop again
    sw $t6,sum # store sum in $t6
    li $v0,4 # system call for printing string
    la $a0,out_s # load address of string to be printed in a0
    syscall # call OS to  execute the operation
    li $v0,1 # system call for printing the integer
    move $a0,$t6 # move result to $ a0
    syscall # call OS to  execute the operation
    li $v0,10 # terminate program
    syscall
```

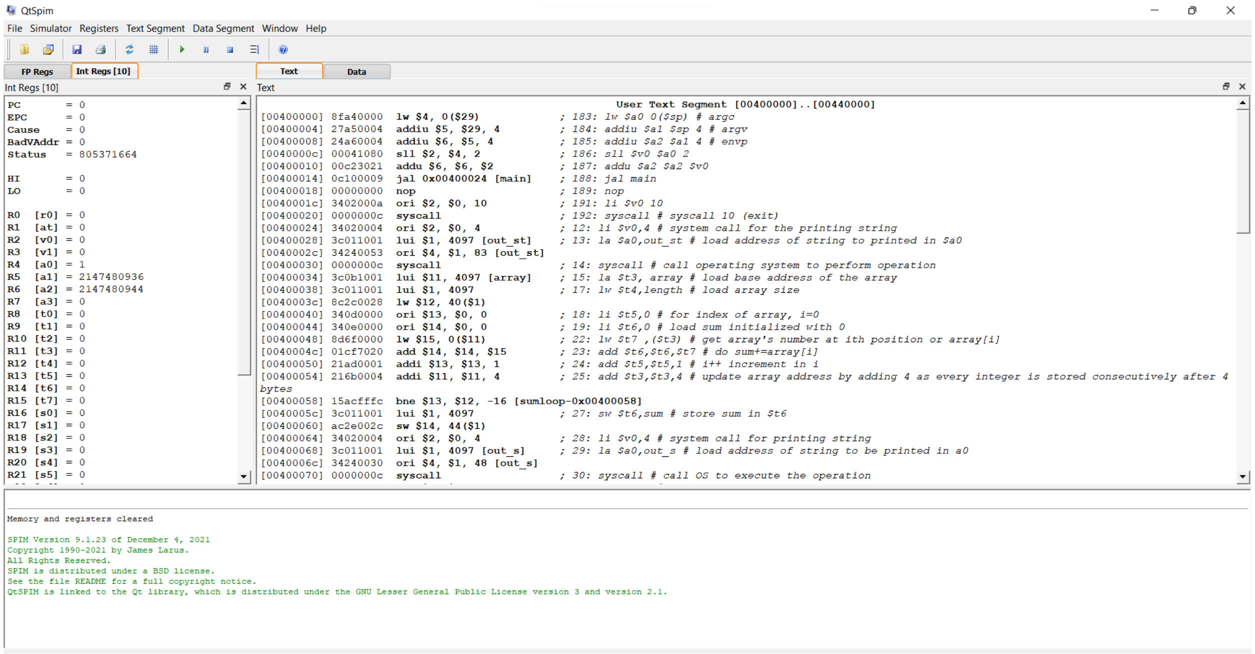
Brief overview of the code section

Array contains 10 integers stored in 16 bit word format ,the starting address is represented by name array
Array length is 10
Sum is initiated with value 0
Some strings to be printed on the console

In main section array address is stored in $t3 by la(load address)
 $t4 contains length of the array
$t5 for index of array, i=0
$t6 to store sum initialized with 0

Loop section
Array address is loaded in $t7
then it is added with $t6
Index is incremented by 1
Address is incremented by adding 4 in the previous address as integers are 4 bits long
Then if index is not equal to length we again branch to the loop by bne instruction
And at last the result is stored and printed on the console.

After loading the first file

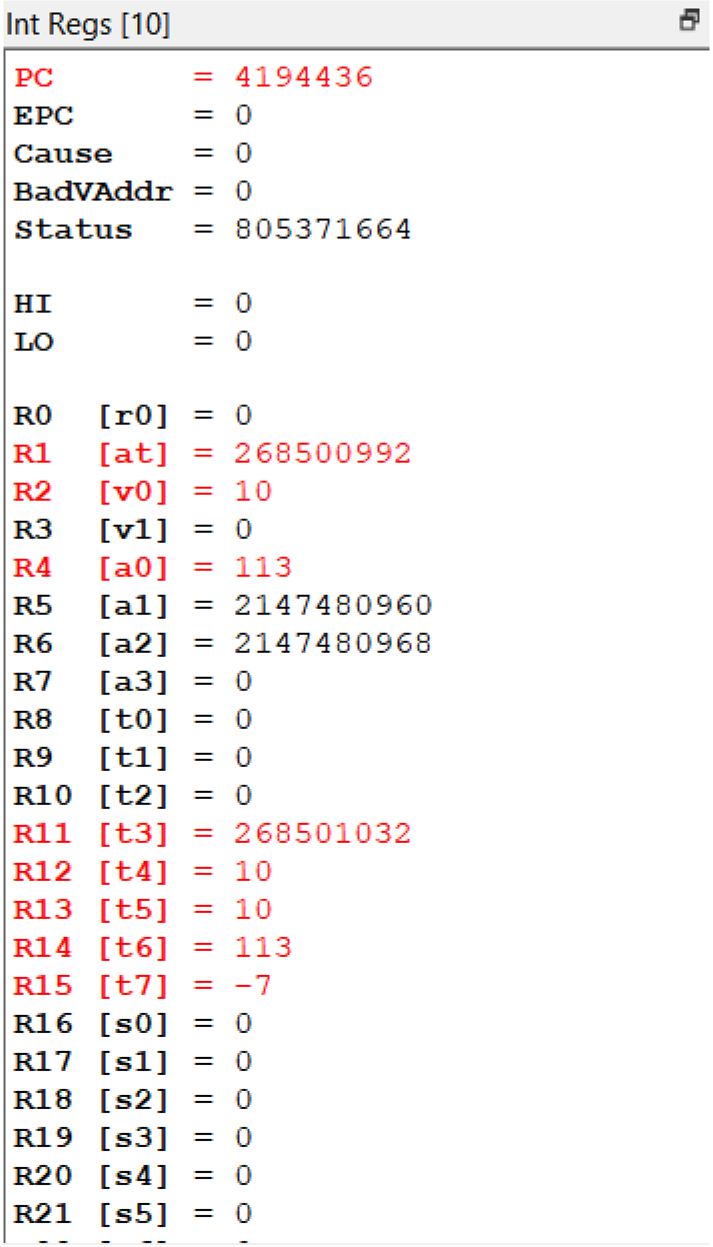


```
PC       = 0
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147480936
R6  [a2] = 2147480944
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
```

```
                              User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)            ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4         ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4          ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2            ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2          ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]    ; 188: jal main
[00400018] 00000000  nop                      ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10           ; 191: li $v0 10
[00400020] 0000000c  syscall                  ; 192: syscall # syscall 10 (exit)
[00400024] 34020004  ori $2, $0, 4            ; 12: li $v0,4 # system call for the printing string
[00400028] 3c011001  lui $1, 4097 [out_st]    ; 13: la $a0,out_st # load address of string to printed in $a0
[0040002c] 34240053  ori $4, $1, 83 [out_st]
[00400030] 0000000c  syscall                  ; 14: syscall # call operating system to perform operation
[00400034] 3c0b1001  lui $11, 4097 [array]    ; 15: la $t3, array # load base address of the array
[00400038] 3c011001  lui $1, 4097             ; 17: lw $t4,length # load array size
[0040003c] 8c2c0028  lw $12, 40($1)
[00400040] 340d0000  ori $13, $0, 0           ; 18: li $t5,0 # for index of array, i=0
[00400044] 340e0000  ori $14, $0, 0           ; 19: li $t6,0 # load sum initialized with 0
[00400048] 8d6f0000  lw $15, 0($11)           ; 22: lw $t7 ,($t3) # get array's number at ith position or array[i]
[0040004c] 01cf7020  add $14, $14, $15        ; 23: add $t6,$t6,$t7 # do sum+=array[i]
[00400050] 21ad0001  addi $13, $13, 1         ; 24: add $t5,$t5,1 # i++ increment in i
[00400054] 216b0004  addi $11, $11, 4         ; 25: add $t3,$t3,4 # update array address by adding 4 as every integer is stored consecutively after 4
bytes
[00400058] 15acfffc  bne $13, $12, -16 [sumloop-0x00400058]
[0040005c] 3c011001  lui $1, 4097             ; 27: sw $t6,sum # store sum in $t6
[00400060] ac2e002c  sw $14, 44($1)
[00400064] 34020004  ori $2, $0, 4            ; 28: li $v0,4 # system call for printing string
[00400068] 3c011001  lui $1, 4097 [out_s]     ; 29: la $a0,out_s # load address of string to be printed in a0
[0040006c] 34240030  ori $4, $1, 48 [out_s]
[00400070] 0000000c  syscall                  ; 30: syscall # call OS to execute the operation
```

```
Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

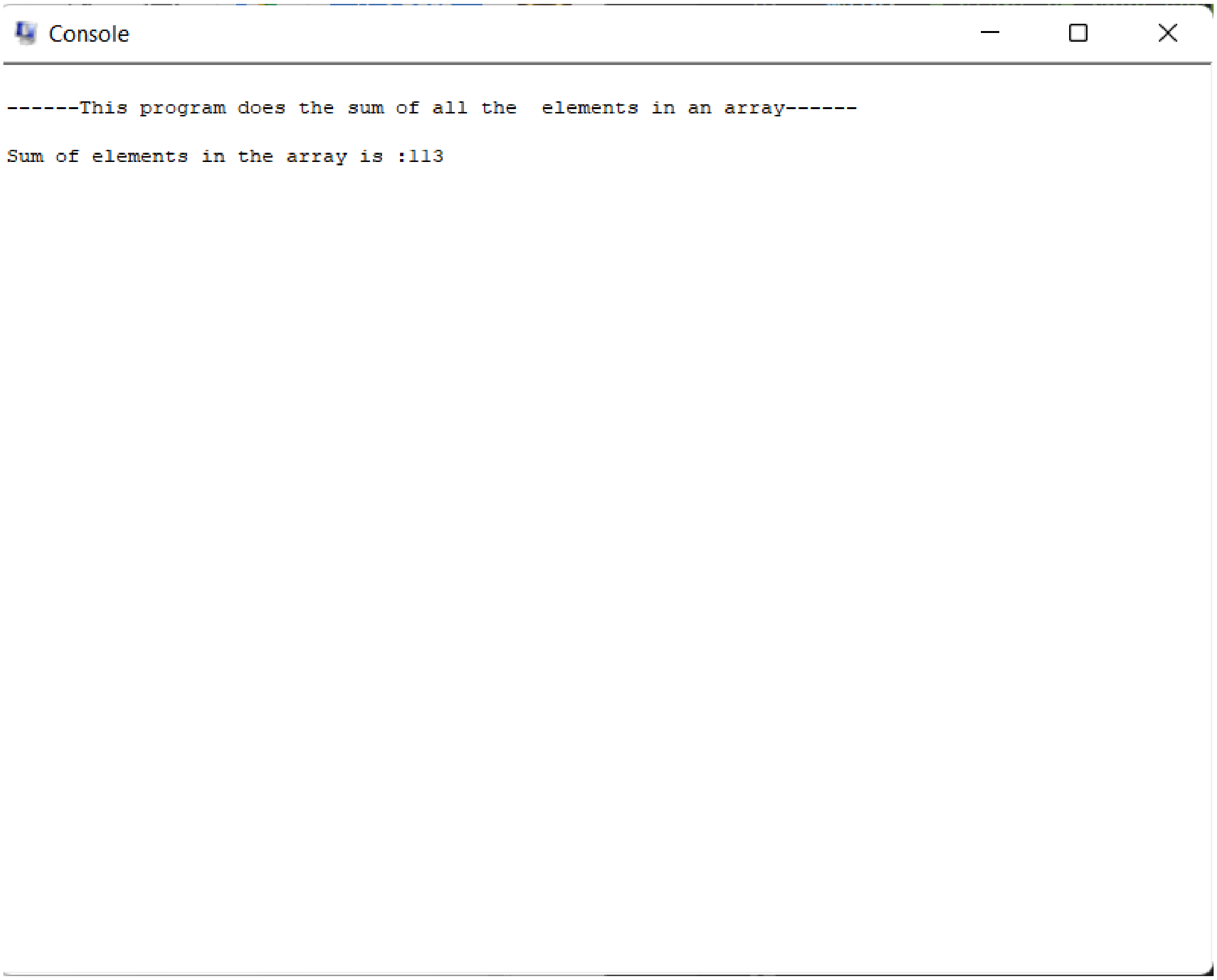
Registers after execution of the code



Int Regs [10]

```
PC        = 4194436
EPC       = 0
Cause     = 0
BadVAddr  = 0
Status    = 805371664

HI        = 0
LO        = 0

R0  [r0]  = 0
R1  [at]  = 268500992
R2  [v0]  = 10
R3  [v1]  = 0
R4  [a0]  = 113
R5  [a1]  = 2147480960
R6  [a2]  = 2147480968
R7  [a3]  = 0
R8  [t0]  = 0
R9  [t1]  = 0
R10 [t2]  = 0
R11 [t3]  = 268501032
R12 [t4]  = 10
R13 [t5]  = 10
R14 [t6]  = 113
R15 [t7]  = -7
R16 [s0]  = 0
R17 [s1]  = 0
R18 [s2]  = 0
R19 [s3]  = 0
R20 [s4]  = 0
R21 [s5]  = 0
```

Console after execution of the program



```
------This program does the sum of all the  elements in an array------

Sum of elements in the array is :113
```

Loop instructions

1) lw $t7 ,($t3) # get array's number at ith position or array[i]

2) add $t6,$t6,$t7 # do sum+=array[i]

3) add $t5,$t5,1 # i++ increment in i

4)add $t3,$t3,4 # update array address by adding 4 as every integer is stored consecutively after 4 bytes

5)bne $t5,$t4,sumloop # if $t5 is not equal to $t4 loop again

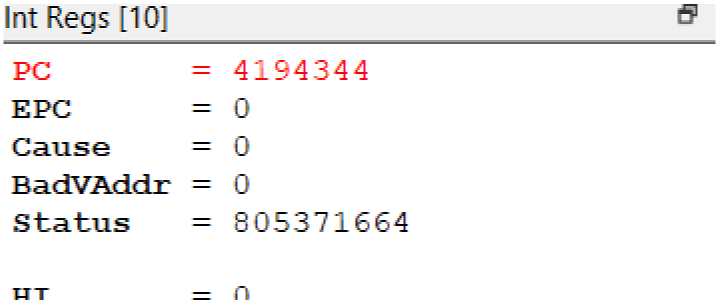
5 loop instructions for each number

So total loop instructions to be executed =length X instructions for a single number= 10 X 5=50

other instructions for printing strings or integers on console loading them etc. for once

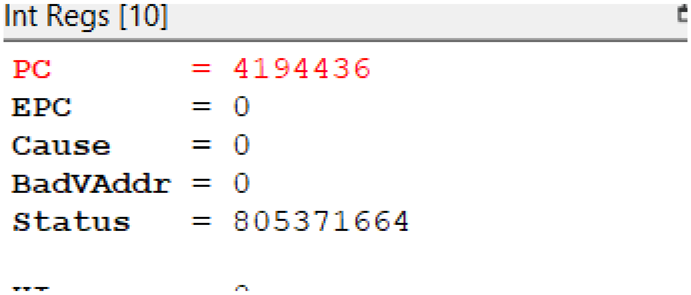
Can be calculated by pc value(program counter)

Pc value at the start of the program



```
PC         = 4194344
EPC        = 0
Cause      = 0
BadVAddr   = 0
Status     = 805371664

HI         = 0
```

Pc value at the end of the program



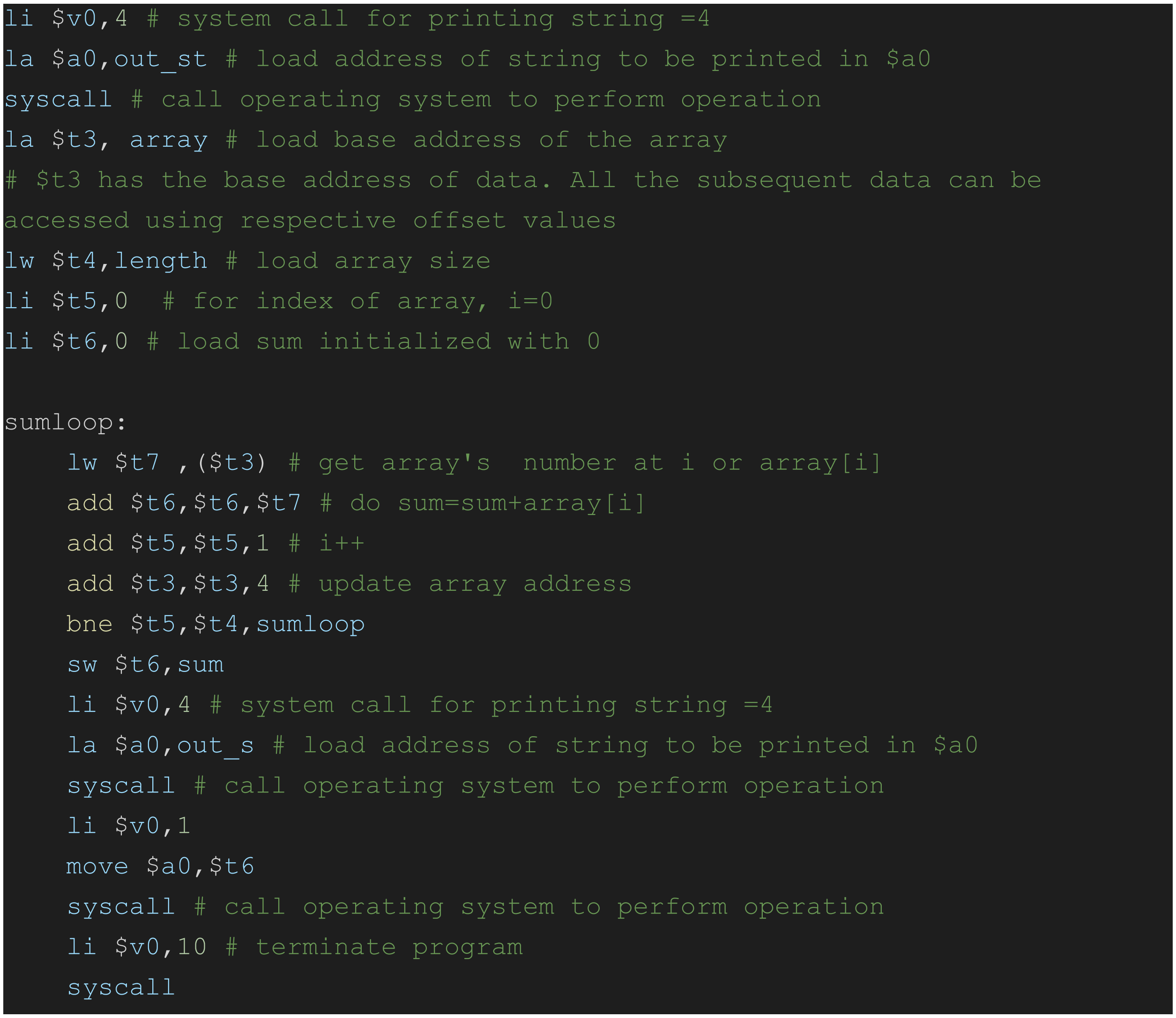
```
PC         = 4194436
EPC        = 0
Cause      = 0
BadVAddr   = 0
Status     = 805371664

HI         = 0
```

Other instructions =(4194436-4194344)/4 - loop instructions =92/4-5=23-5=18

So **total instructions = total loop instructions + other instructions =50 +18= 68**

Q.2) Include the following numbers in the array data segment of question 1. 10,20,30,40,50,77

```
.data
array : .word 10, 12, 15, -10, 13, 82, -9, 4, 3, -7   # load the array
        .word 10, 20, 30, 40, 50, 77
length: .word 16 # load the length of the array as 10
sum: .word 0 # initialize sum to 0
out_s: .asciiz "\nSum of elements in the array is :"
out_st: .asciiz "\n------This program does the sum of all the  elements in
an array------\n"

.text  # tells assembler to switch to the text segment or succeeding lines
contains instructions

main: # start of code section
```



```
li $v0,4 # system call for printing string =4
la $a0,out_st # load address of string to be printed in $a0
syscall # call operating system to perform operation
la $t3, array # load base address of the array
# $t3 has the base address of data. All the subsequent data can be
accessed using respective offset values
lw $t4,length # load array size
li $t5,0  # for index of array, i=0
li $t6,0 # load sum initialized with 0

sumloop:
    lw $t7 ,($t3) # get array's  number at i or array[i]
    add $t6,$t6,$t7 # do sum=sum+array[i]
    add $t5,$t5,1 # i++
    add $t3,$t3,4 # update array address
    bne $t5,$t4,sumloop
    sw $t6,sum
    li $v0,4 # system call for printing string =4
    la $a0,out_s # load address of string to be printed in $a0
    syscall # call operating system to perform operation
    li $v0,1
    move $a0,$t6
    syscall # call operating system to perform operation
    li $v0,10 # terminate program
    syscall
```

Brief overview of the code section

Array size is made to 16 and 6 new integers are added at the back of the array rest of the code is exact same as Q.1

Now the total loop instructions to be executed =length X instructions for a single number= 16 X 5=80

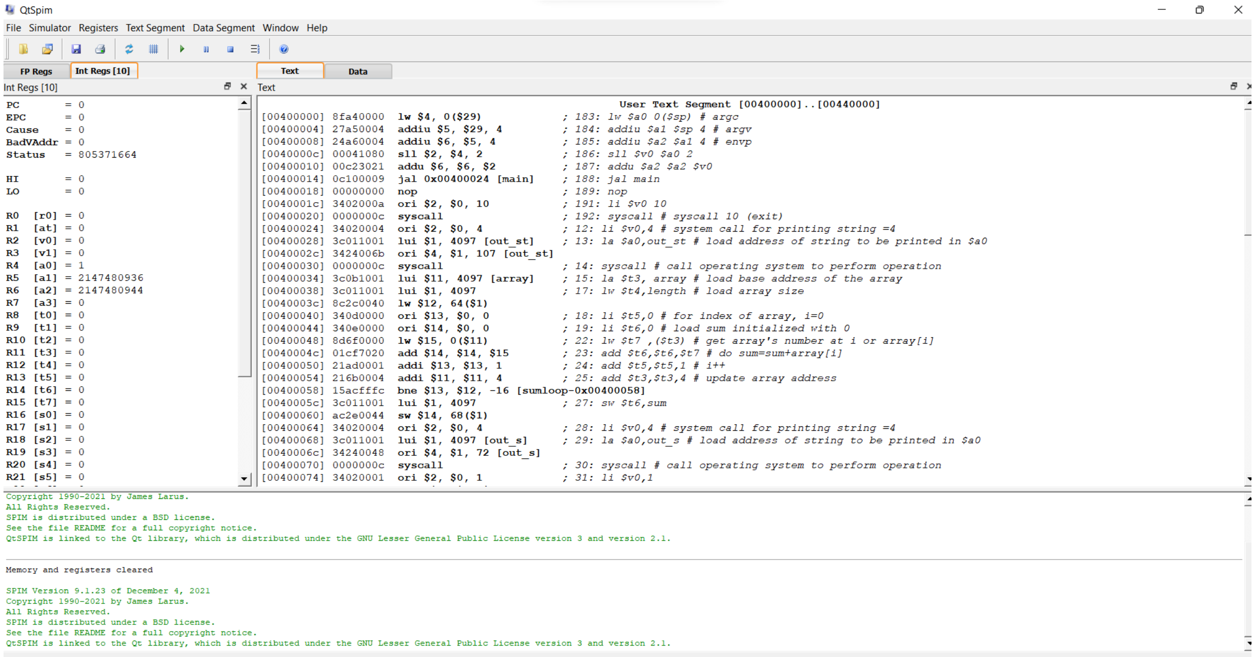
18 other instructions for printing strings or integers on console loading them etc. for once ,same as Q.1

So total instructions = total loop instructions + other instructions =80 +18= 98

The total loop instructions are now increased by 30 from before as we have added 6 elements at the back of the array

So 6 X loop instructions for a single entity = 6 X 5 =30 are increased from before as Those corresponds to increased array size

After loading the second file



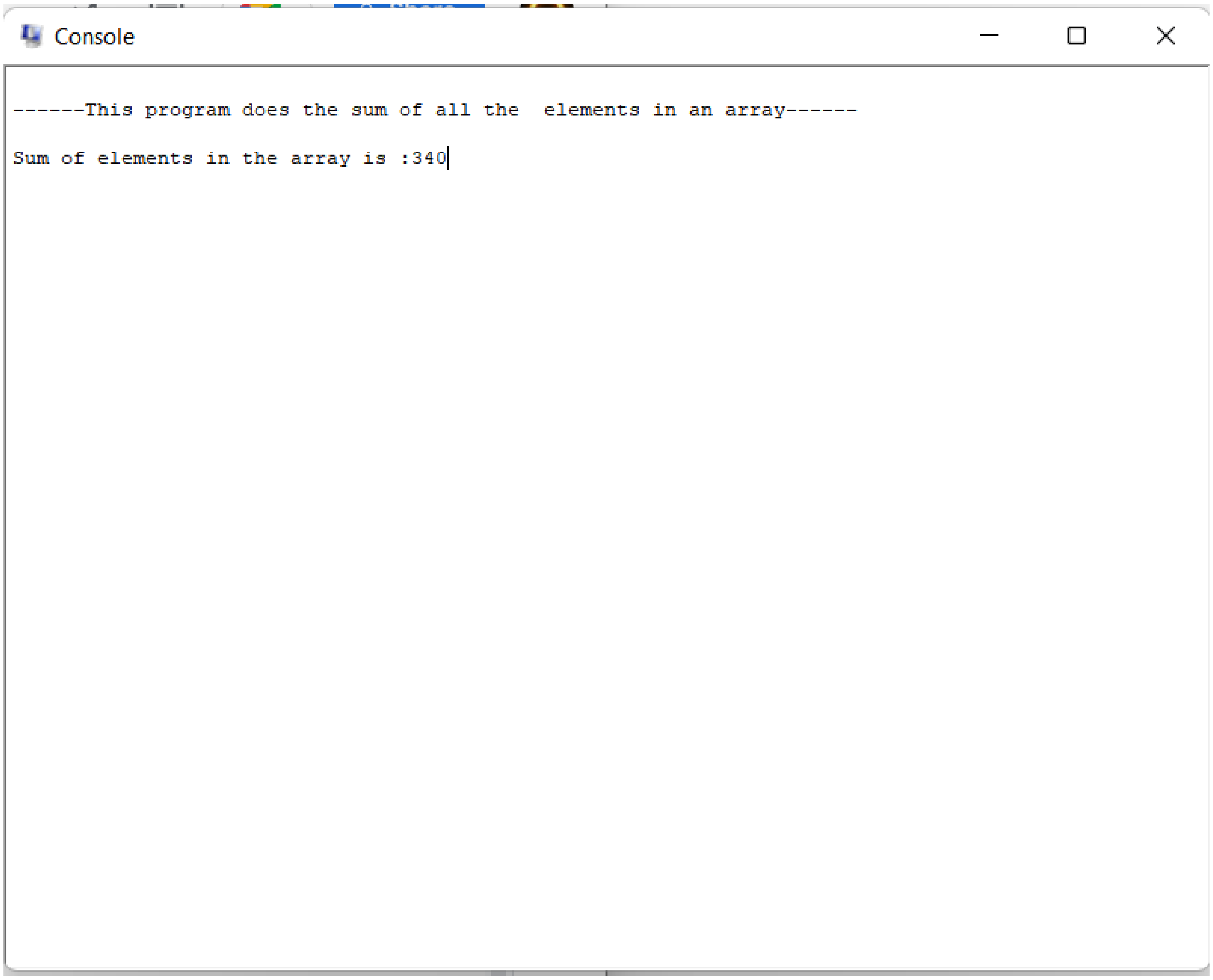
QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [10]
Text
Data
Int Regs [10]
Text

```
PC       = 0
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147480936
R6  [a2] = 2147480944
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
```

```
                                        User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)           ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4        ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4         ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2           ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2         ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]   ; 188: jal main
[00400018] 00000000  nop                     ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10          ; 191: li $v0 10
[00400020] 0000000c  syscall                 ; 192: syscall # syscall 10 (exit)
[00400024] 34020004  ori $2, $0, 4           ; 12: li $v0,4 # system call for printing string =4
[00400028] 3c011001  lui $1, 4097 [out_st]   ; 13: la $a0,out_st # load address of string to be printed in $a0
[0040002c] 3424006b  ori $4, $1, 107 [out_st]
[00400030] 0000000c  syscall                 ; 14: syscall # call operating system to perform operation
[00400034] 3c0b1001  lui $11, 4097 [array]   ; 15: la $t3, array # load base address of the array
[00400038] 3c011001  lui $1, 4097            ; 17: lw $t4,length # load array size
[0040003c] 8c2c0040  lw $12, 64($1)
[00400040] 340d0000  ori $13, $0, 0          ; 18: li $t5,0 # for index of array, i=0
[00400044] 340e0000  ori $14, $0, 0          ; 19: li $t6,0 # load sum initialized with 0
[00400048] 8d6f0000  lw $15, 0($11)          ; 22: lw $t7 ,($t3) # get array's number at i or array[i]
[0040004c] 01cf7020  add $14, $14, $15       ; 23: add $t6,$t6,$t7 # do sum=sum+array[i]
[00400050] 21ad0001  addi $13, $13, 1        ; 24: add $t5,$t5,1 # i++
[00400054] 216b0004  addi $11, $11, 4        ; 25: add $t3,$t3,4 # update array address
[00400058] 15acfffc  bne $13, $12, -16 [sumloop-0x00400058]
[0040005c] 3c011001  lui $1, 4097            ; 27: sw $t6,sum
[00400060] ac2e0044  sw $14, 68($1)
[00400064] 34020004  ori $2, $0, 4           ; 28: li $v0,4 # system call for printing string =4
[00400068] 3c011001  lui $1, 4097 [out_s]    ; 29: la $a0,out_s # load address of string to be printed in $a0
[0040006c] 34240048  ori $4, $1, 72 [out_s]
[00400070] 0000000c  syscall                 ; 30: syscall # call operating system to perform operation
[00400074] 34020001  ori $2, $0, 1           ; 31: li $v0,1
```

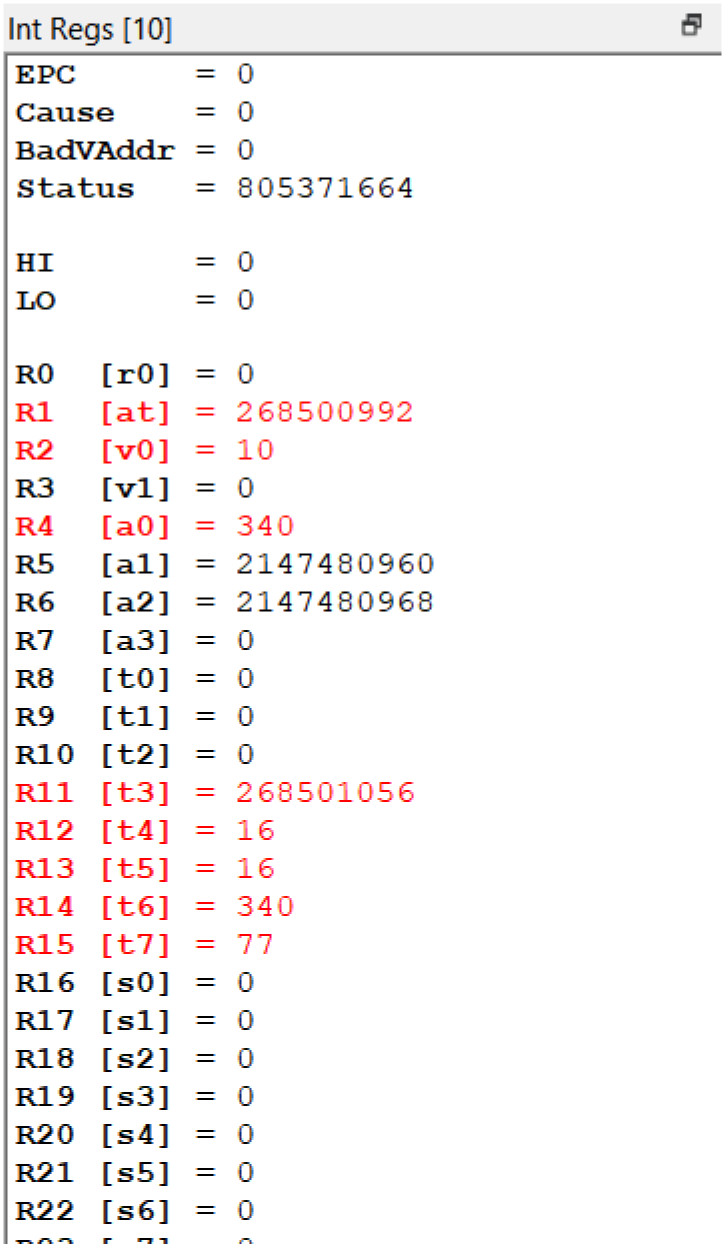
```
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

Console after execution of the program



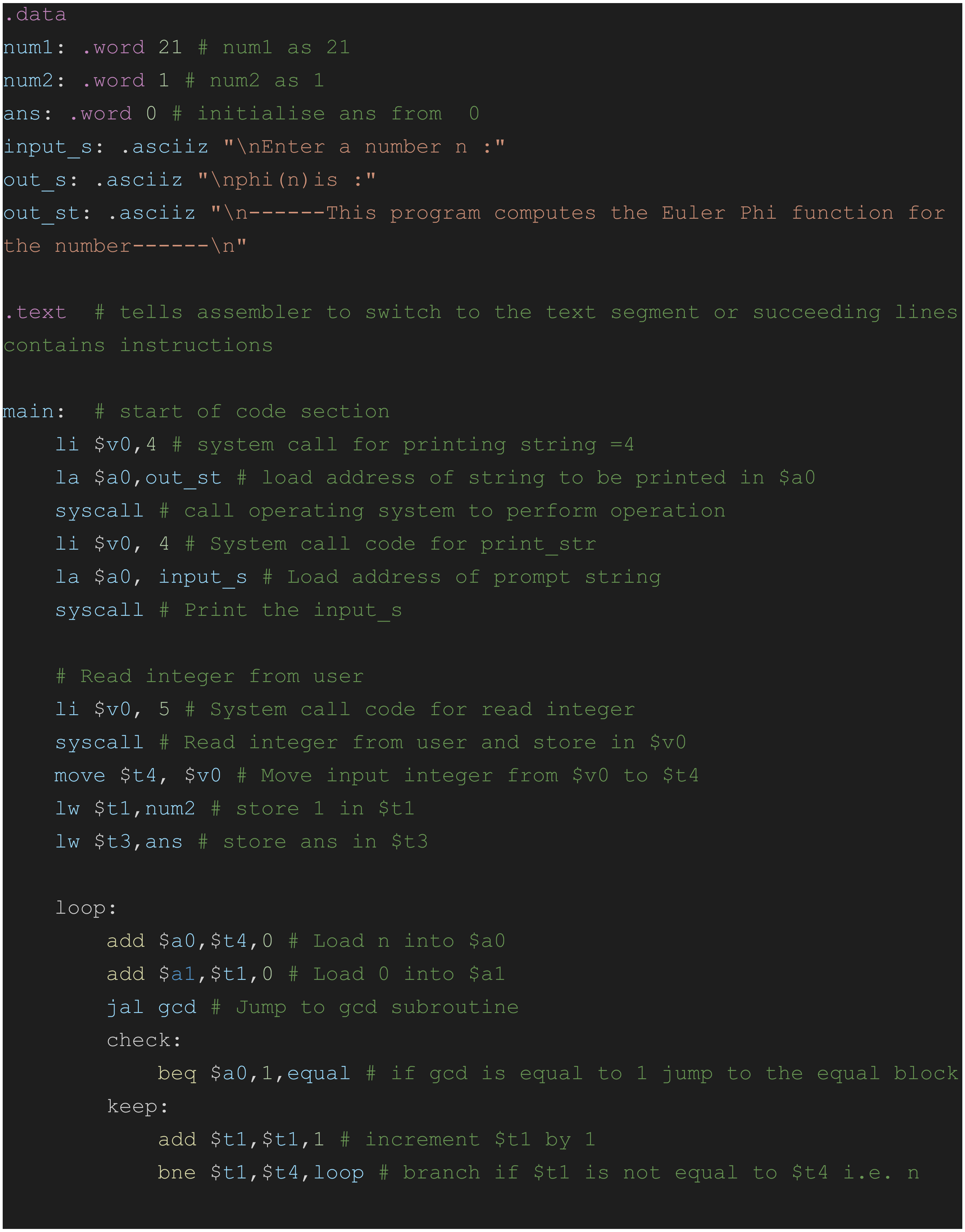
```
------This program does the sum of all the  elements in an array------

Sum of elements in the array is :340
```

Registers after execution of the code



```
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 340
R5  [a1] = 2147480960
R6  [a2] = 2147480968
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 268501056
R12 [t4] = 16
R13 [t5] = 16
R14 [t6] = 340
R15 [t7] = 77
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
```

Q.3) Euler's Phi function for an input n.
Compute the Euler Phi function for the number 21.



```
.data
num1: .word 21 # num1 as 21
num2: .word 1 # num2 as 1
ans: .word 0 # initialise ans from  0
input_s: .asciiz "\nEnter a number n :"
out_s: .asciiz "\nphi(n)is :"
out_st: .asciiz "\n------This program computes the Euler Phi function for
the number------\n"

.text  # tells assembler to switch to the text segment or succeeding lines
contains instructions

main:  # start of code section
    li $v0,4 # system call for printing string =4
    la $a0,out_st # load address of string to be printed in $a0
    syscall # call operating system to perform operation
    li $v0, 4 # System call code for print_str
    la $a0, input_s # Load address of prompt string
    syscall # Print the input_s

    # Read integer from user
    li $v0, 5 # System call code for read integer
    syscall # Read integer from user and store in $v0
    move $t4, $v0 # Move input integer from $v0 to $t4
    lw $t1,num2 # store 1 in $t1
    lw $t3,ans # store ans in $t3

    loop:
        add $a0,$t4,0 # Load n into $a0
        add $a1,$t1,0 # Load 0 into $a1
        jal gcd # Jump to gcd subroutine
        check:
            beq $a0,1,equal # if gcd is equal to 1 jump to the equal block
        keep:
            add $t1,$t1,1 # increment $t1 by 1
            bne $t1,$t4,loop # branch if $t1 is not equal to $t4 i.e. n
```

```
      li $v0,4 # system call for printing string = 4
      la $a0,out_s # load address of string to be printed in $a0
      syscall # call operating system to perform operation

      li $v0, 1 # System call code for print_int
      move $a0, $t3 # Load result into $a0 from $t3
      syscall # Print the result

      li $v0, 10 # System call code for exit
      syscall # Exit program

    #
----------------------------------------------------------------------------
---------------------------------------------------------#
    # gcd function
    # Recursive definition
    # gcd(m,n)==gcd(n,(m%n))
    #
----------------------------------------------------------------------------
---------------------------------------------------------#
    gcd:
      # Base case
      beq $a1, $0, end # If second number is 0, return first number
      move $t0, $a0 # Save first number
      move $a0,$a1 # move second  number in $a0 it is now our first number
      div $t0, $a1 # Divide first number by second number
      mfhi $a1 # Store the remainder in $a1(remainder is now second
number)

      j gcd # Recursively call the gcd function

    end:
      j check # Return to check block  our gcd is stored in $a0
    equal:
          add $t3,$t3,1 # increment $t3 by 1
          j keep # jump to keep block
```

Brief overview of the code section

With system call and li $v0, 5 read an integer n from the user move the stored value to $t4
Load 1 in $t1

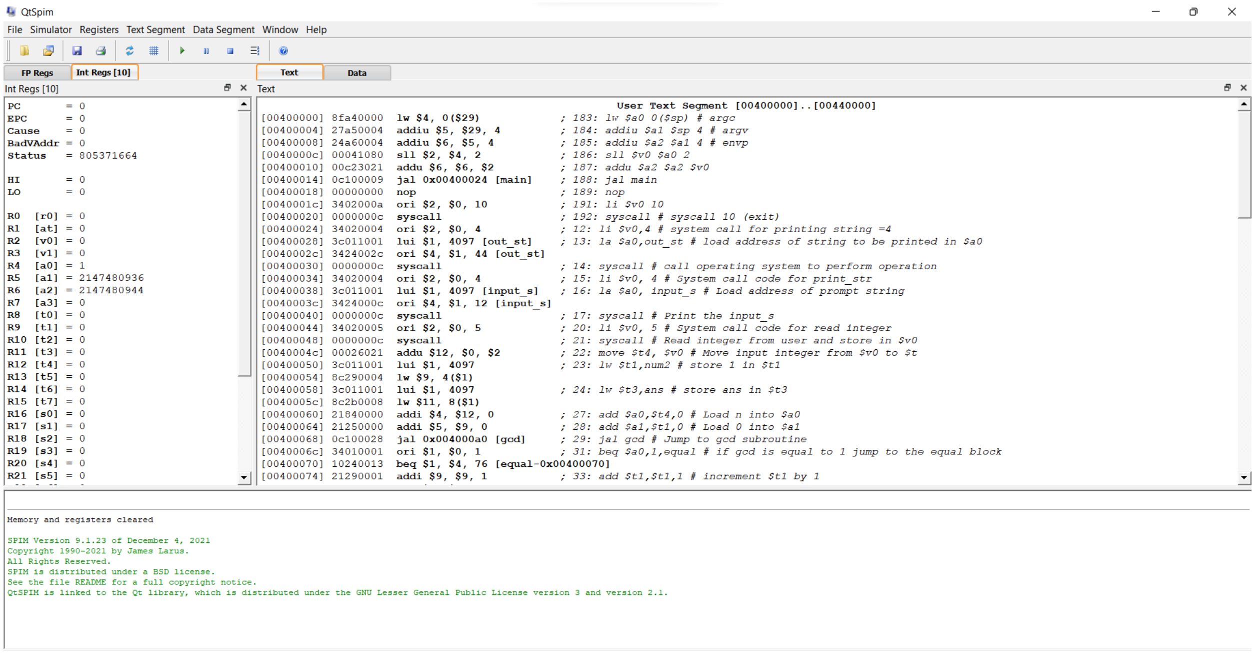
Loop section

```
 Add $t4 with 0 and store in $a0
 Store value in $t1 into $a1
 Call gcd with $a0 and $a1

   Now in  the gcd block
        Check for the base case if $a1 equal to zero then jump to the end block and gcd is
        stored  in $a0
    else
          store $a0 in $t0
          Move $a1  in $a0
          Div  $t0 by $a1
          Store the remainder in $a1
 And call gcd function recursively
```

Here in gcd function gcd is calculated by Euclidean method
gcd(m,n)==gcd(n,(m%n))

From the end block we jump to check where code checks whether gcd is equal to 1 or not, if it is then ans in $t3 is incremented by one as we have found a number whose gcd with n is 1 and keep moving in the loop
Else we keep moving in the loop without incrementing ans in $t3
Increment $t1 by one
Iterate through the loop till $t1 is less than equal n for each $t1 incremented by 1 each time
Print the ans on the console

After loading the second file

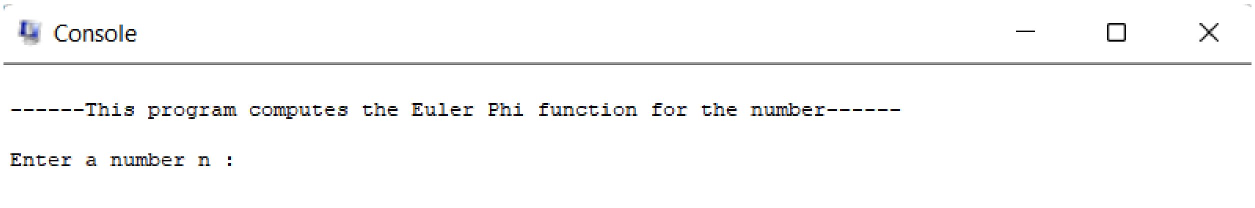

QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [10]
Text
Data
Int Regs [10]
Text

```
PC       = 0
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147480936
R6  [a2] = 2147480944
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
```

```
                                    User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)          ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4       ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4        ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2          ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2        ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]  ; 188: jal main
[00400018] 00000000  nop                    ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10         ; 191: li $v0 10
[00400020] 0000000c  syscall                ; 192: syscall # syscall 10 (exit)
[00400024] 34020004  ori $2, $0, 4          ; 12: li $v0,4 # system call for printing string =4
[00400028] 3c011001  lui $1, 4097 [out_st]  ; 13: la $a0,out_st # load address of string to be printed in $a0
[0040002c] 3424002c  ori $4, $1, 44 [out_st]
[00400030] 0000000c  syscall                ; 14: syscall # call operating system to perform operation
[00400034] 34020004  ori $2, $0, 4          ; 15: li $v0, 4 # System call code for print_str
[00400038] 3c011001  lui $1, 4097 [input_s] ; 16: la $a0, input_s # Load address of prompt string
[0040003c] 3424000c  ori $4, $1, 12 [input_s]
[00400040] 0000000c  syscall                ; 17: syscall # Print the input_s
[00400044] 34020005  ori $2, $0, 5          ; 20: li $v0, 5 # System call code for read integer
[00400048] 0000000c  syscall                ; 21: syscall # Read integer from user and store in $v0
[0040004c] 00026021  addu $12, $0, $2       ; 22: move $t4, $v0 # Move input integer from $v0 to $t
[00400050] 3c011001  lui $1, 4097           ; 23: lw $t1,num2 # store 1 in $t1
[00400054] 8c290004  lw $9, 4($1)
[00400058] 3c011001  lui $1, 4097           ; 24: lw $t3,ans # store ans in $t3
[0040005c] 8c2b0008  lw $11, 8($1)
[00400060] 21840000  addi $4, $12, 0        ; 27: add $a0,$t4,0 # Load n into $a0
[00400064] 21250000  addi $5, $9, 0         ; 28: add $a1,$t1,0 # Load 0 into $a1
[00400068] 0c100028  jal 0x004000a0 [gcd]   ; 29: jal gcd # Jump to gcd subroutine
[0040006c] 34010001  ori $1, $0, 1          ; 31: beq $a0,1,equal # if gcd is equal to 1 jump to the equal block
[00400070] 10240013  beq $1, $4, 76 [equal-0x00400070]
[00400074] 21290001  addi $9, $9, 1         ; 33: add $t1,$t1,1 # increment $t1 by 1
```

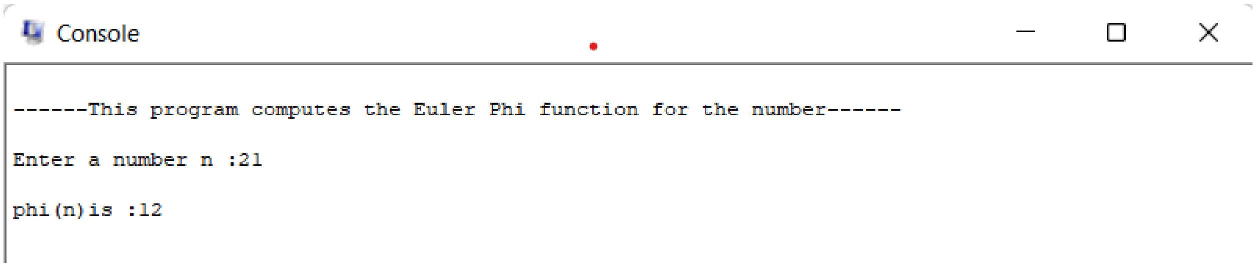
```
Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

Console waiting for the user input for number n



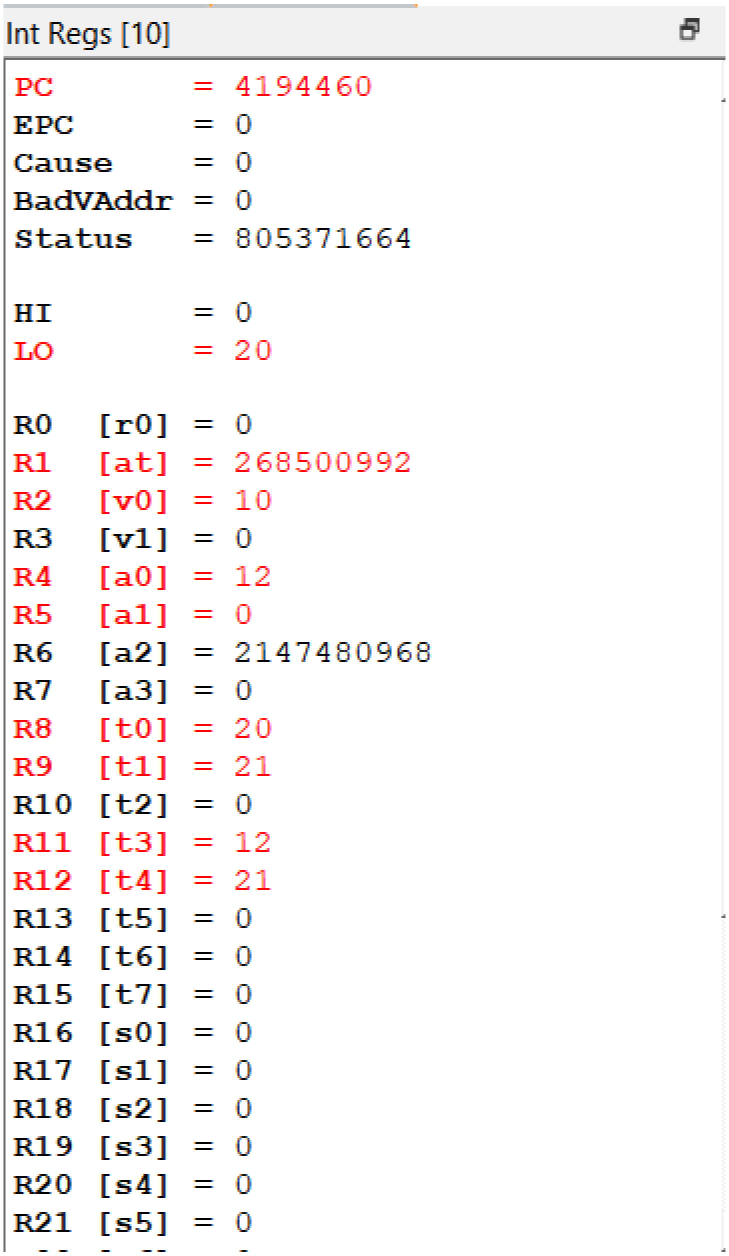
```
------This program computes the Euler Phi function for the number------

Enter a number n :
```

Result after putting 21



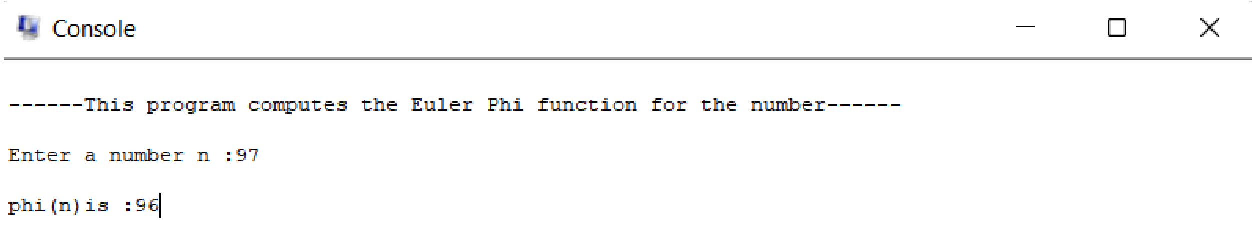
```
------This program computes the Euler Phi function for the number------

Enter a number n :21

phi(n)is :12
```

Register values



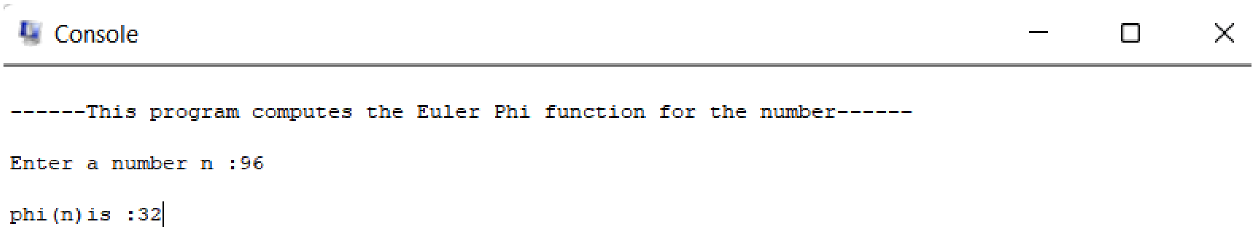
```
PC         = 4194460
EPC        = 0
Cause      = 0
BadVAddr   = 0
Status     = 805371664

HI         = 0
LO         = 20

R0   [r0]  = 0
R1   [at]  = 268500992
R2   [v0]  = 10
R3   [v1]  = 0
R4   [a0]  = 12
R5   [a1]  = 0
R6   [a2]  = 2147480968
R7   [a3]  = 0
R8   [t0]  = 20
R9   [t1]  = 21
R10  [t2]  = 0
R11  [t3]  = 12
R12  [t4]  = 21
R13  [t5]  = 0
R14  [t6]  = 0
R15  [t7]  = 0
R16  [s0]  = 0
R17  [s1]  = 0
R18  [s2]  = 0
R19  [s3]  = 0
R20  [s4]  = 0
R21  [s5]  = 0
```

Let’s find Euler Phi function for some other integers



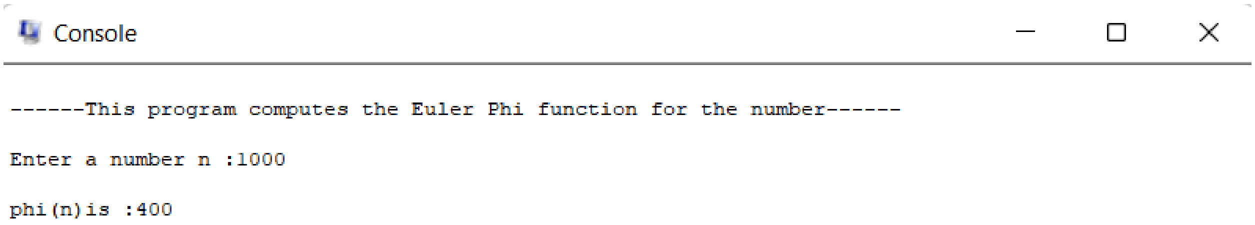
```
------This program computes the Euler Phi function for the number------

Enter a number n :97

phi(n)is :96
```

We know phi(n) if n is prime is n-1 , 97 is a prime number so phi(97)==96



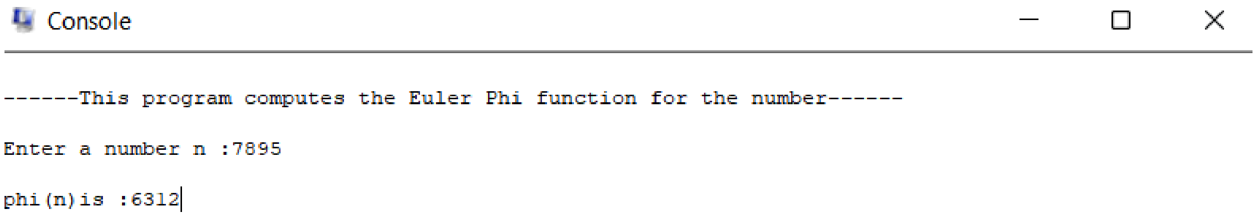
```
------This program computes the Euler Phi function for the number------

Enter a number n :96

phi(n)is :32
```

phi(96)=32



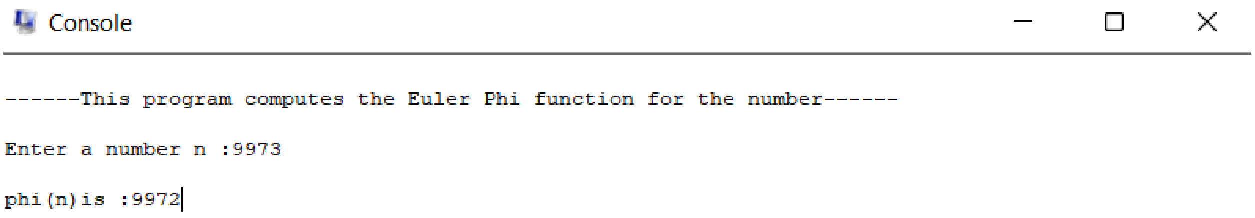
```
------This program computes the Euler Phi function for the number------

Enter a number n :1000

phi(n)is :400
```

phi(1000)=400



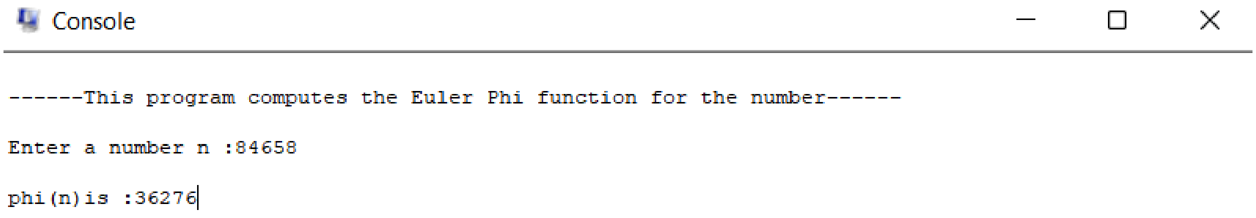
```
------This program computes the Euler Phi function for the number------

Enter a number n :7895

phi(n)is :6312
```

phi(7895)=6312



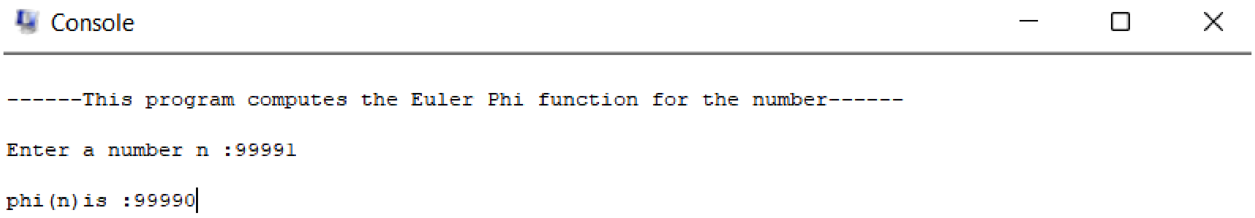
```
------This program computes the Euler Phi function for the number------

Enter a number n :9973

phi(n)is :9972
```

phi(9973)=9972



```
------This program computes the Euler Phi function for the number------

Enter a number n :84658

phi(n)is :36276
```

phi(84658)=36276



```
------This program computes the Euler Phi function for the number------

Enter a number n :99991

phi(n)is :99990
```

phi(99991)=99990

Note -we can also use this program to find whether the given number is prime or not
If phi(n)=n-1 then prime else composite