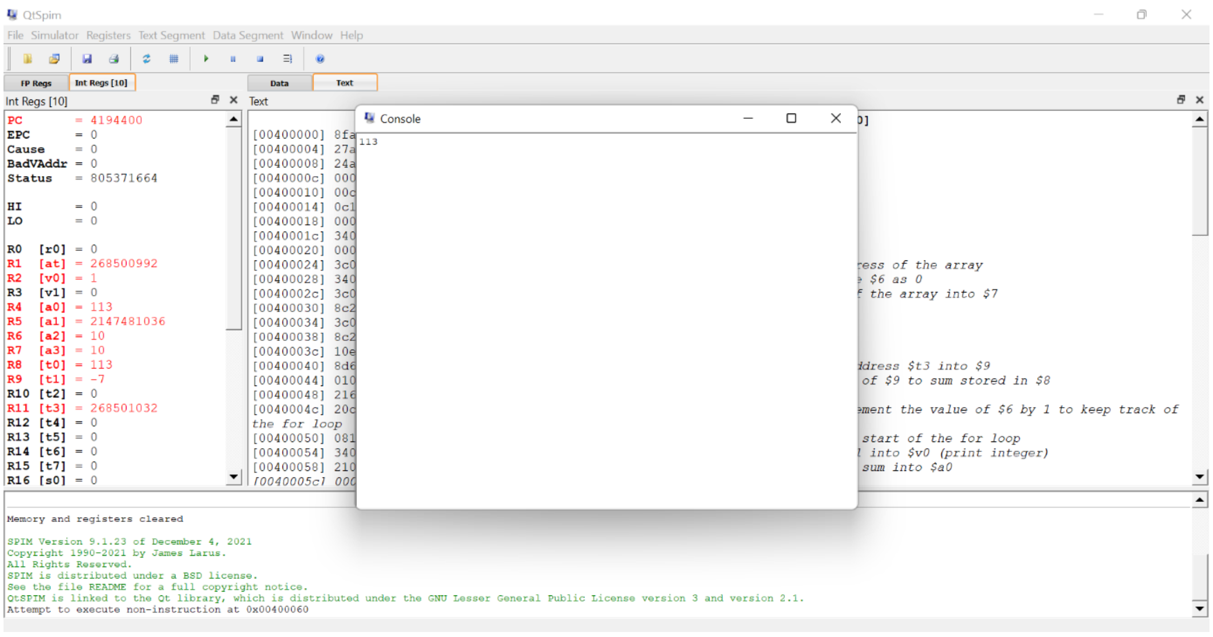
CS 211
LAB 2: System Calls and Control Flow

Name – Gautam Kumar Mahar
Roll No. – 2103114
Branch – Computer Science Engineering

PART B

In this part, you will write MIPS assembly language programs to understand control flow instructions.
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 #array={10,12,15,-10,13,82,-9,4,3,-7}
length: .word 10 #load the length of the array as 10
sum: .word 0 #initialise sum to 0
.text
main:
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.
#Add your code here
Count the total number of machine instructions executed to complete this task.In this part,

In Decimal –


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

```
PC       = 4194400
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 1
R3  [v1] = 0
R4  [a0] = 113
R5  [a1] = 2147481036
R6  [a2] = 10
R7  [a3] = 10
R8  [t0] = 113
R9  [t1] = -7
R10 [t2] = 0
R11 [t3] = 268501032
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
```

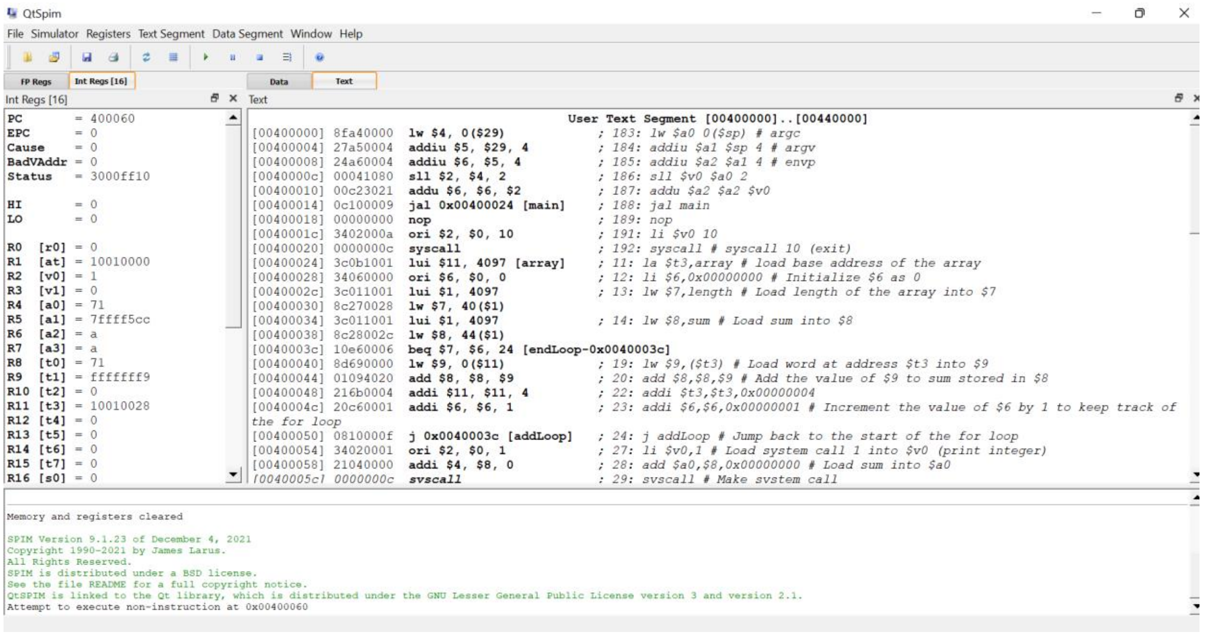
```
[00400000] 8fa
[00400004] 27a
[00400008] 24a
[0040000c] 000
[00400010] 00c
[00400014] 0c1
[00400018] 000
[0040001c] 340
[00400020] 000
[00400024] 3c0
[00400028] 340
[0040002c] 3c0
[00400030] 8c2
[00400034] 3c0
[00400038] 8c2
[0040003c] 10e
[00400040] 8d6
[00400044] 010
[00400048] 216
[0040004c] 20c
the for loop
[00400050] 081
[00400054] 340
[00400058] 210
[0040005c] 000
```

```
0]

ress of the array
e $6 as 0
f the array into $7

ddress $t3 into $9
 of $9 to sum stored in $8

ement the value of $6 by 1 to keep track of

 start of the for loop
l into $v0 (print integer)
 sum into $a0
```

```
113
```

```
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.
Attempt to execute non-instruction at 0x00400060
```

In Hexa Decimal -



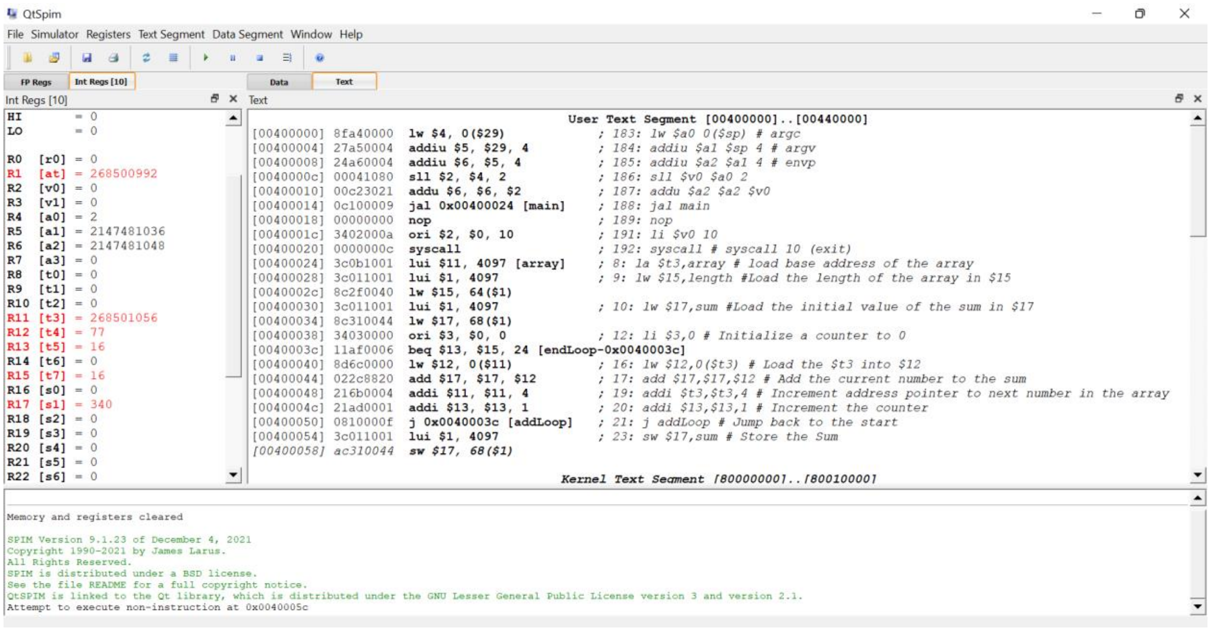
```
PC       = 400060
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 10010000
R2  [v0] = 1
R3  [v1] = 0
R4  [a0] = 71
R5  [a1] = 7ffff5cc
R6  [a2] = a
R7  [a3] = a
R8  [t0] = 71
R9  [t1] = fffffff9
R10 [t2] = 0
R11 [t3] = 10010028
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 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] 3c0b1001  lui $11, 4097 [array]     ; 11: la $t3,array # load base address of the array
[00400028] 34060000  ori $6, $0, 0             ; 12: li $6,0x00000000 # Initialize $6 as 0
[0040002c] 3c011001  lui $1, 4097              ; 13: lw $7,length # Load length of the array into $7
[00400030] 8c270028  lw $7, 40($1)
[00400034] 3c011001  lui $1, 4097              ; 14: lw $8,sum # Load sum into $8
[00400038] 8c28002c  lw $8, 44($1)
[0040003c] 10e60006  beq $7, $6, 24 [endLoop-0x0040003c]
[00400040] 8d690000  lw $9, 0($11)             ; 19: lw $9,($t3) # Load word at address $t3 into $9
[00400044] 01094020  add $8, $8, $9            ; 20: add $8,$8,$9 # Add the value of $9 to sum stored in $8
[00400048] 216b0004  addi $11, $11, 4          ; 22: addi $t3,$t3,0x00000004
[0040004c] 20c60001  addi $6, $6, 1            ; 23: addi $6,$6,0x00000001 # Increment the value of $6 by 1 to keep track of
the for loop
[00400050] 0810000f  j 0x0040003c [addLoop]    ; 24: j addLoop # Jump back to the start of the for loop
[00400054] 34020001  ori $2, $0, 1             ; 27: li $v0,1 # Load system call 1 into $v0 (print integer)
[00400058] 21040000  addi $4, $8, 0            ; 28: add $a0,$8,0x00000000 # Load sum into $a0
[0040005c] 0000000c  syscall                   ; 29: syscall # Make system call
```

```
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.
Attempt to execute non-instruction at 0x00400060
```

B) Include the following numbers in the array data segment of question 1.
10,20,30,40,50,77
Now you have 16 numbers residing in the array (data memory). Add these numbers and display the
result.
Count the total number of instructions.
Compare and analyse the relation between the number of data elements and total number of machine
instructions executed.

In Decimal –



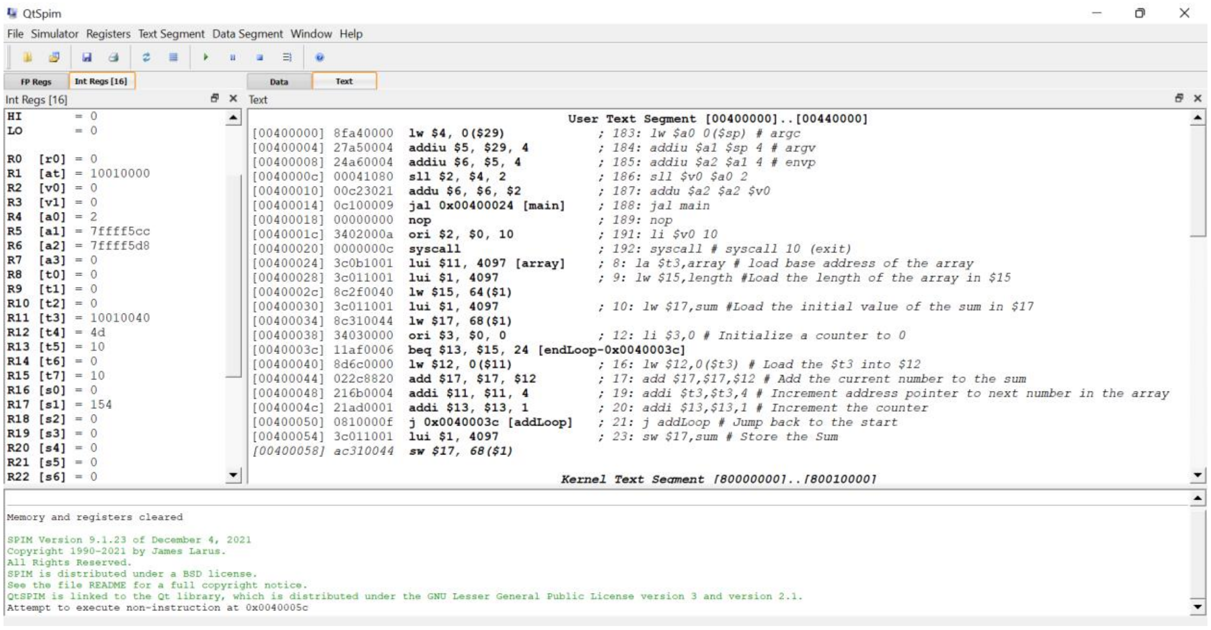
```
Int Regs [10]
HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 2
R5  [a1] = 2147481036
R6  [a2] = 2147481048
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 268501056
R12 [t4] = 77
R13 [t5] = 16
R14 [t6] = 0
R15 [t7] = 16
R16 [s0] = 0
R17 [s1] = 340
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 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] 3c0b1001  lui $11, 4097 [array]    ; 8: la $t3,array # load base address of the array
[00400028] 3c011001  lui $1, 4097             ; 9: lw $15,length #Load the length of the array in $15
[0040002c] 8c2f0040  lw $15, 64($1)
[00400030] 3c011001  lui $1, 4097             ; 10: lw $17,sum #Load the initial value of the sum in $17
[00400034] 8c310044  lw $17, 68($1)
[00400038] 34030000  ori $3, $0, 0            ; 12: li $3,0 # Initialize a counter to 0
[0040003c] 11af0006  beq $13, $15, 24 [endLoop-0x0040003c]
[00400040] 8d6c0000  lw $12, 0($11)           ; 16: lw $12,0($t3) # Load the $t3 into $12
[00400044] 022c8820  add $17, $17, $12        ; 17: add $17,$17,$12 # Add the current number to the sum
[00400048] 216b0004  addi $11, $11, 4         ; 19: addi $t3,$t3,4 # Increment address pointer to next number in the array
[0040004c] 21ad0001  addi $13, $13, 1         ; 20: addi $13,$13,1 # Increment the counter
[00400050] 0810000f  j 0x0040003c [addLoop]   ; 21: j addLoop # Jump back to the start
[00400054] 3c011001  lui $1, 4097             ; 23: sw $17,sum # Store the Sum
[00400058] ac310044  sw $17, 68($1)

                                       Kernel Text Segment [80000000]..[80010000]
```

```
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.
Attempt to execute non-instruction at 0x0040005c
```

In Hexa Decimal –

-



```
Int Regs [16]
HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 10010000
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 2
R5  [a1] = 7ffff5cc
R6  [a2] = 7ffff5d8
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 10010040
R12 [t4] = 4d
R13 [t5] = 10
R14 [t6] = 0
R15 [t7] = 10
R16 [s0] = 0
R17 [s1] = 154
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 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] 3c0b1001  lui $11, 4097 [array]    ; 8: la $t3,array # load base address of the array
[00400028] 3c011001  lui $1, 4097             ; 9: lw $15,length #Load the length of the array in $15
[0040002c] 8c2f0040  lw $15, 64($1)
[00400030] 3c011001  lui $1, 4097             ; 10: lw $17,sum #Load the initial value of the sum in $17
[00400034] 8c310044  lw $17, 68($1)
[00400038] 34030000  ori $3, $0, 0            ; 12: li $3,0 # Initialize a counter to 0
[0040003c] 11af0006  beq $13, $15, 24 [endLoop-0x0040003c]
[00400040] 8d6c0000  lw $12, 0($11)           ; 16: lw $12,0($t3) # Load the $t3 into $12
[00400044] 022c8820  add $17, $17, $12        ; 17: add $17,$17,$12 # Add the current number to the sum
[00400048] 216b0004  addi $11, $11, 4         ; 19: addi $t3,$t3,4 # Increment address pointer to next number in the array
[0040004c] 21ad0001  addi $13, $13, 1         ; 20: addi $13,$13,1 # Increment the counter
[00400050] 0810000f  j 0x0040003c [addLoop]   ; 21: j addLoop # Jump back to the start
[00400054] 3c011001  lui $1, 4097             ; 23: sw $17,sum # Store the Sum
[00400058] ac310044  sw $17, 68($1)

                                       Kernel Text Segment [80000000]..[80010000]
```

```
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.
Attempt to execute non-instruction at 0x0040005c
```

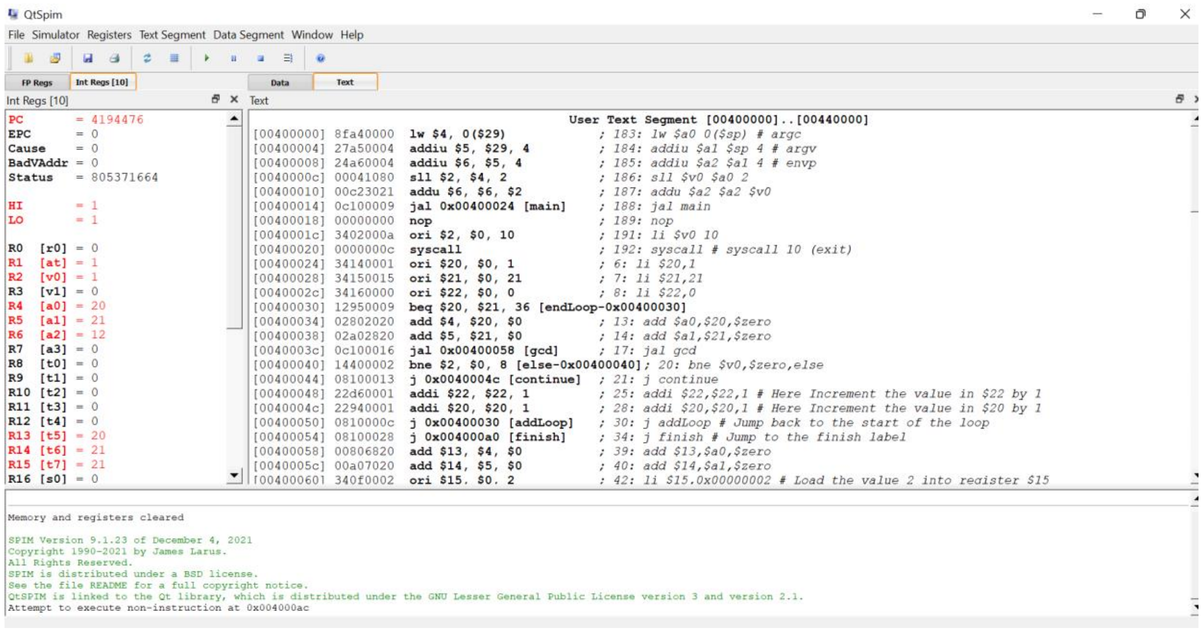
3) Compute the Euler Phi function for the number 21.
Euler's Phi function for an input n, denoted as (phi(n)) is the count of numbers in {1, 2, 3, …, n} that
are relatively prime to n, i.e, the numbers whose GCD (Greatest Common Divisor) with n is 1.
Examples:
phi(1)=1, (gcd(1,1)=1)
phi(2)=1, (gcd(1,2)=1, but gcd(2,2)=2)
phi(3)=2, (gcd(1,3)=1, gcd(2,3)=1, gcd(3,3)=3)
phi(4)=2 , (gcd(1,4)=1, gcd(2,4)=2, gcd(3,4)=1, gcd(4,4)=4)
phi(5)=4, (gcd(1,5)=1, gcd(2,5)=1, gcd(3,5)=1, gcd(4,5)=1, gcd(5,5)=5)

(Hint: The logic for your code would be as follows

```
phi = 0;
trial = 1;
while ( trial &lt; N) #where N is the number under consideration (given : 21)
{
if ( gcd(N,trial) == 1 ) phi++;
} )
```

The computed value, which is phi(21), should be printed on the screen.

In Decimal –

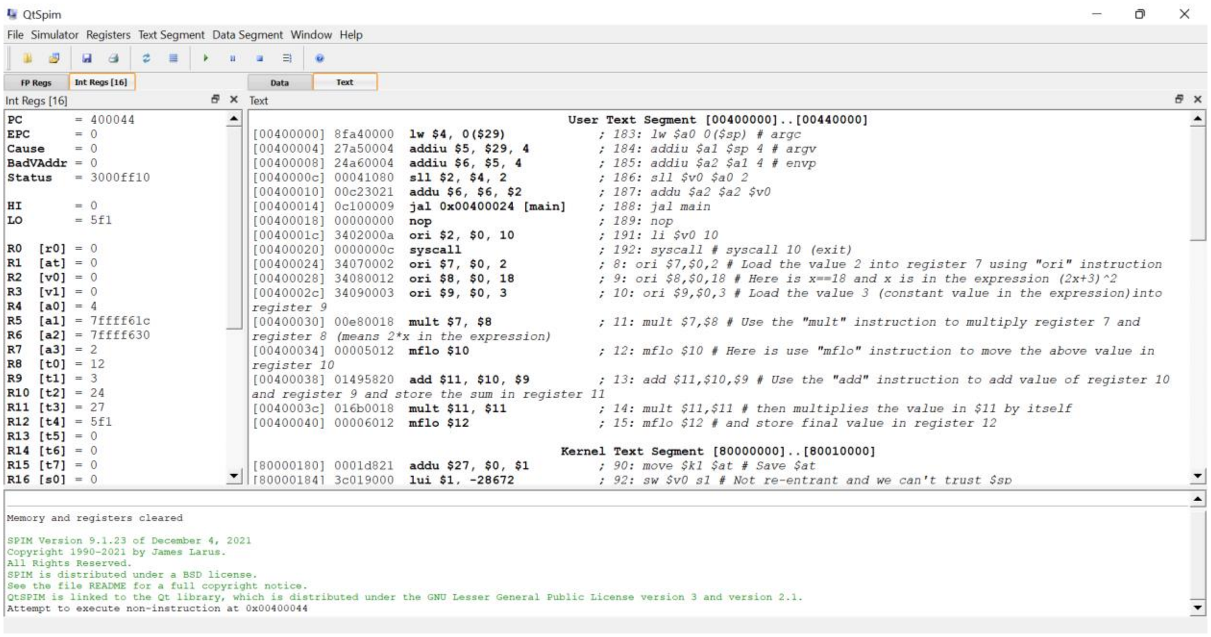

QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [10]
Data
Text
Int Regs [10]
Text

```
PC       = 4194476
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 1
LO       = 1

R0  [r0] = 0
R1  [at] = 1
R2  [v0] = 1
R3  [v1] = 0
R4  [a0] = 20
R5  [a1] = 21
R6  [a2] = 12
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 20
R14 [t6] = 21
R15 [t7] = 21
R16 [s0] = 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] 34140001  ori $20, $0, 1           ; 6: li $20,1
[00400028] 34150015  ori $21, $0, 21          ; 7: li $21,21
[0040002c] 34160000  ori $22, $0, 0           ; 8: li $22,0
[00400030] 12950009  beq $20, $21, 36 [endLoop-0x00400030]
[00400034] 02802020  add $4, $20, $0          ; 13: add $a0,$20,$zero
[00400038] 02a02820  add $5, $21, $0          ; 14: add $a1,$21,$zero
[0040003c] 0c100016  jal 0x00400058 [gcd]     ; 17: jal gcd
[00400040] 14400002  bne $2, $0, 8 [else-0x00400040]; 20: bne $v0,$zero,else
[00400044] 08100013  j 0x0040004c [continue]  ; 21: j continue
[00400048] 22d60001  addi $22, $22, 1         ; 25: addi $22,$22,1 # Here Increment the value in $22 by 1
[0040004c] 22940001  addi $20, $20, 1         ; 28: addi $20,$20,1 # Here Increment the value in $20 by 1
[00400050] 0810000c  j 0x00400030 [addLoop]   ; 30: j addLoop # Jump back to the start of the loop
[00400054] 08100028  j 0x004000a0 [finish]    ; 34: j finish # Jump to the finish label
[00400058] 00806820  add $13, $4, $0          ; 39: add $13,$a0,$zero
[0040005c] 00a07020  add $14, $5, $0          ; 40: add $14,$a1,$zero
[00400060] 340f0002  ori $15, $0, 2           ; 42: li $15,0x00000002 # Load the value 2 into register $15
```

```
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.
Attempt to execute non-instruction at 0x004000ac
```

In Hexa Decimal –


QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [16]
Data
Text
Int Regs [16]
Text

```
PC       = 400044
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 5f1

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 4
R5  [a1] = 7ffff61c
R6  [a2] = 7ffff630
R7  [a3] = 2
R8  [t0] = 12
R9  [t1] = 3
R10 [t2] = 24
R11 [t3] = 27
R12 [t4] = 5f1
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 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] 34070002  ori $7, $0, 2           ; 8: ori $7,$0,2 # Load the value 2 into register 7 using "ori" instruction
[00400028] 34080012  ori $8, $0, 18          ; 9: ori $8,$0,18 # Here is x==18 and x is in the expression (2x+3)^2
[0040002c] 34090003  ori $9, $0, 3           ; 10: ori $9,$0,3 # Load the value 3 (constant value in the expression)into
register 9
[00400030] 00e80018  mult $7, $8             ; 11: mult $7,$8 # Use the "mult" instruction to multiply register 7 and
register 8 (means 2*x in the expression)
[00400034] 00005012  mflo $10                ; 12: mflo $10 # Here is use "mflo" instruction to move the above value in
register 10
[00400038] 01495820  add $11, $10, $9        ; 13: add $11,$10,$9 # Use the "add" instruction to add value of register 10
and register 9 and store the sum in register 11
[0040003c] 016b0018  mult $11, $11           ; 14: mult $11,$11 # then multiplies the value in $11 by itself
[00400040] 00006012  mflo $12                ; 15: mflo $12 # and store final value in register 12

                               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
```

```
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.
Attempt to execute non-instruction at 0x00400044
```

Thank You.