# CS2610: Computer Organization and Architecture

## <u>Lab 7: Report</u>

Devansh Singh Rathore(111701011)

## Objective

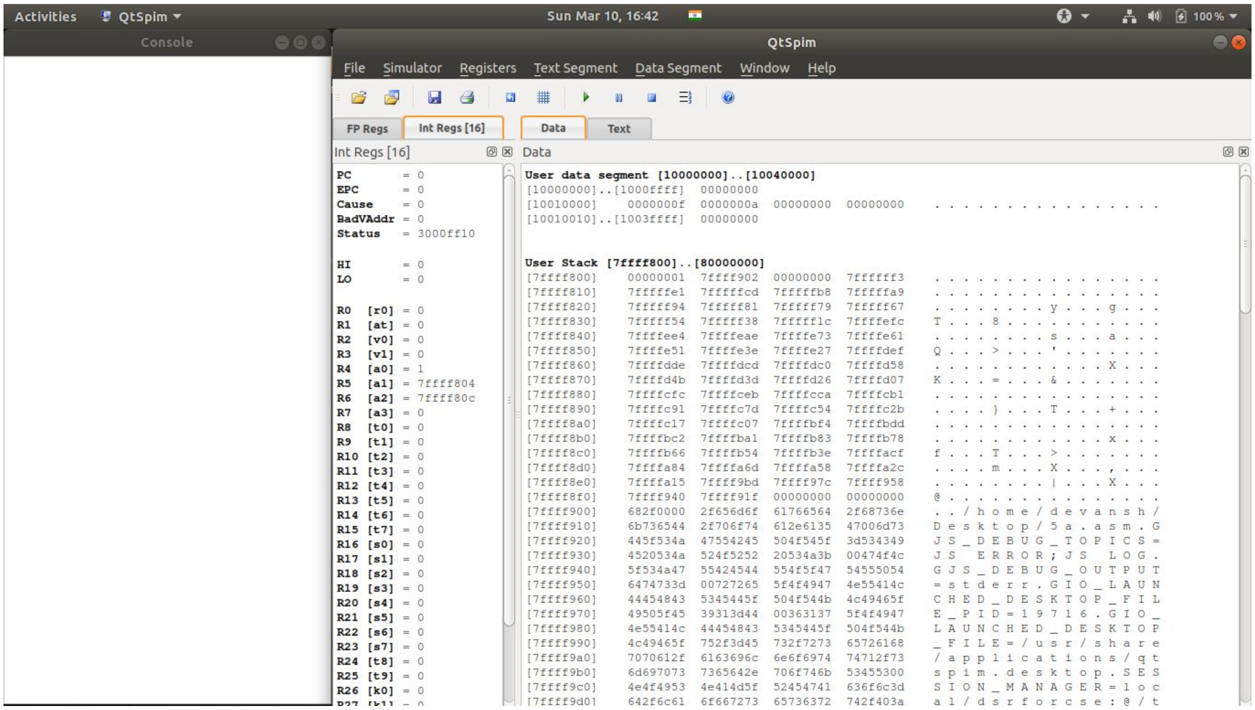
In this lab you will familiarize with MIPS relational instructions, branching instructions, arrays, stack operations, floating point operations using the QtSpim simulator.

## Problem

Illustrate the following using QtSpim:
(1) Compare two numbers A=0xFFFFFFFF, B=0x0000000F.
(2) Introduction to arrays
(3) Compute the total age in the array
(4) Introduction to stack operations
(5) Floating point arithmetic

## Implementation

*Initial Snapshot of Registers, Data Segments, and Console:*



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

HI        = 0
LO        = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 7ffff804
R6  [a2] = 7ffff80c
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
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
```

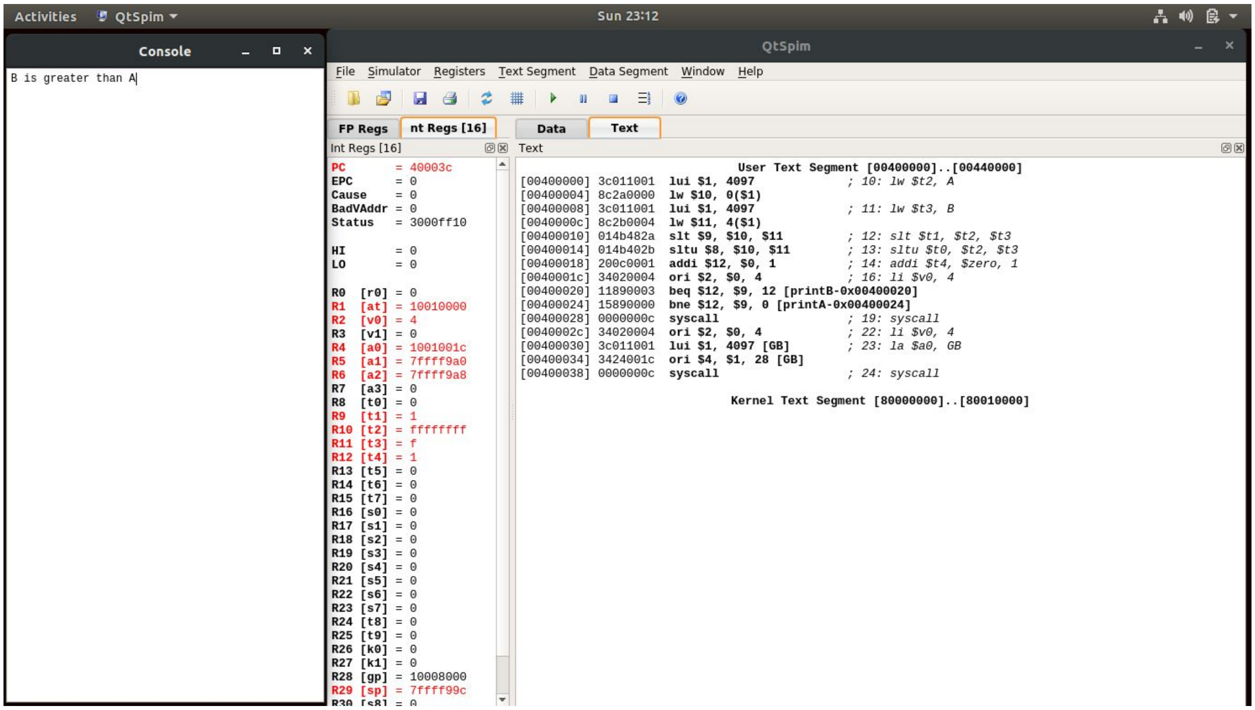
```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    0000000f  0000000a  00000000  00000000    . . . . . . . . . . . . . . . .
[10010010]..[1003ffff]  00000000


User Stack [7ffff800]..[80000000]
[7ffff800]    00000001  7ffff902  00000000  7ffffff3    . . . . . . . . . . . . . . . .
[7ffff810]    7fffffe1  7fffffcd  7fffffb8  7fffffa9    . . . . . . . . . . . . . . . .
[7ffff820]    7fffff94  7fffff81  7fffff79  7fffff67    . . . . . . . . y . . . g . . .
[7ffff830]    7fffff54  7fffff38  7fffff1c  7ffffefc    T . . . 8 . . . . . . . . . . .
[7ffff840]    7ffffee4  7ffffeae  7ffffe73  7ffffe61    . . . . . . . . s . . . a . . .
[7ffff850]    7ffffe51  7ffffe3e  7ffffe27  7ffffdef    Q . . . > . . . ' . . . . . . .
[7ffff860]    7ffffdde  7ffffdcd  7ffffdc0  7ffffd58    . . . . . . . . . . . . X . . .
[7ffff870]    7ffffd4b  7ffffd3d  7ffffd26  7ffffd07    K . . . = . . . & . . . . . . .
[7ffff880]    7ffffcfc  7ffffceb  7ffffcca  7ffffcb1    . . . . . . . . . . . . . . . .
[7ffff890]    7ffffc91  7ffffc7d  7ffffc54  7ffffc2b    . . . . } . . . T . . . + . . .
[7ffff8a0]    7ffffc17  7ffffc07  7ffffbf4  7ffffbdd    . . . . . . . . . . . . . . . .
[7ffff8b0]    7ffffbc2  7ffffba1  7ffffb83  7ffffb78    . . . . . . . . . . . . x . . .
[7ffff8c0]    7ffffb66  7ffffb54  7ffffb3e  7ffffacf    f . . . T . . . > . . . . . . .
[7ffff8d0]    7ffffa84  7ffffa6d  7ffffa58  7ffffa2c    . . . . m . . . X . . . , . . .
[7ffff8e0]    7ffffa15  7ffff9bd  7ffff97c  7ffff958    . . . . . . . . | . . . X . . .
[7ffff8f0]    7ffff940  7ffff91f  00000000  00000000    @ . . . . . . . . . . . . . . .
[7ffff900]    682f0000  2f656d6f  61766564  2f68736e    . . / h o m e / d e v a n s h /
[7ffff910]    6b736544  2f706f74  612e6135  47006d73    D e s k t o p / 5 a . a s m . G
[7ffff920]    445f534a  47554245  504f545f  3d534349    J S _ D E B U G _ T O P I C S =
[7ffff930]    4520534a  524f5252  20534a3b  00474f4c    J S   E R R O R ; J S   L O G .
[7ffff940]    5f534a47  55424544  554f5f47  54555054    G J S _ D E B U G _ O U T P U T
[7ffff950]    6474733d  00727265  5f4f4947  4e55414c    = s t d e r r . G I O _ L A U N
[7ffff960]    44454843  5345445f  504f544b  4c49465f    C H E D _ D E S K T O P _ F I L
[7ffff970]    49505f45  39313d44  00363137  5f4f4947    E _ P I D = 1 9 7 1 6 . G I O _
[7ffff980]    4e55414c  44454843  5345445f  504f544b    L A U N C H E D _ D E S K T O P
[7ffff990]    4c49465f  752f3d45  732f7273  65726168    _ F I L E = / u s r / s h a r e
[7ffff9a0]    7070612f  6163696c  6e6f6974  74712f73    / a p p l i c a t i o n s / q t
[7ffff9b0]    6d697073  7365642e  706f746b  53455300    s p i m . d e s k t o p . S E S
[7ffff9c0]    4e4f4953  4e414d5f  52454741  636f6c3d    S I O N _ M A N A G E R = l o c
[7ffff9d0]    642f6c61  6f667273  65736372  742f403a    a l / d s r f o r c s e : @ / t
```

## (1) Compare two numbers A=0xFFFFFFFF, B=0x0000000F::

**CODE:**

```
#I1::

.data
A: .word 0xFFFFFFFF
B: .word 0x0000000F
GA: .asciiz "A is greater than B"
GB: .asciiz "B is greater than A"
.text
main:
lw $t2, A
lw $t3, B
slt $t1, $t2, $t3
sltu $t0, $t2, $t3
addi $t4, $zero, 1

li $v0, 4
beq $t4, $t1, printB
bne $t4, $t1, printA
syscall

printB:
li $v0, 4
la $a0, GB
syscall
```

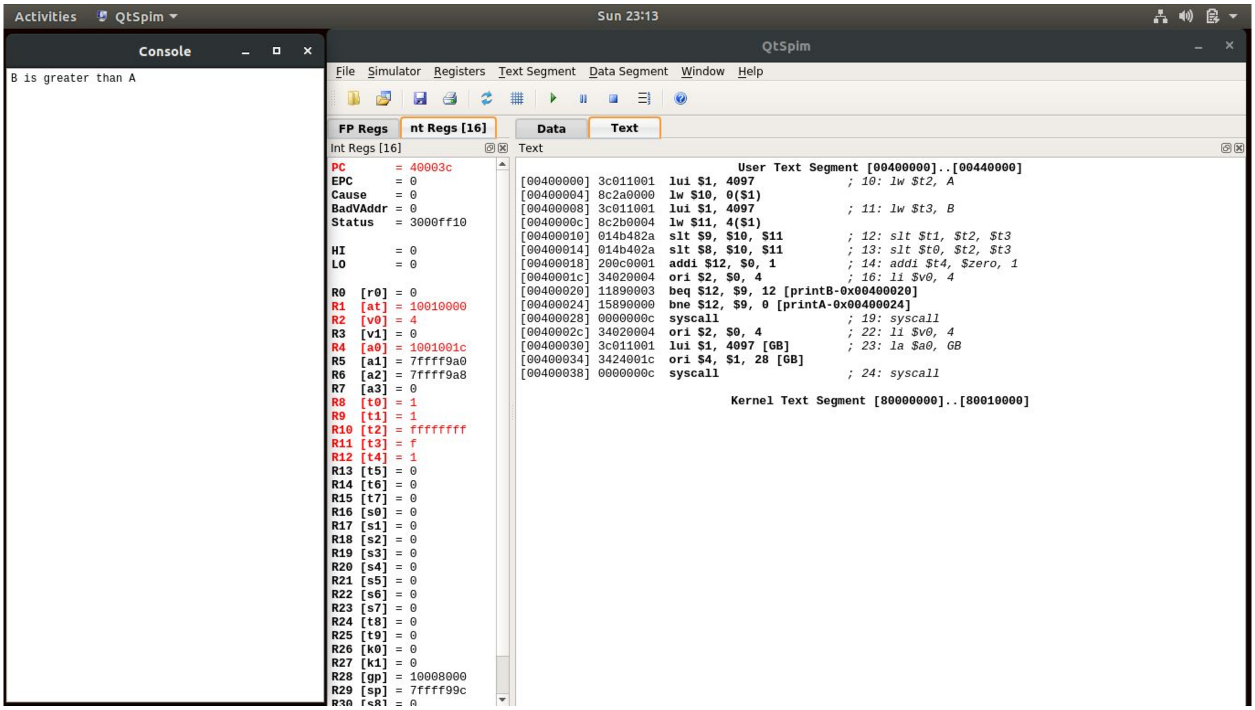
**CHANGES IN REGISTERS/DATA SEGMENTS(Final Snapshots after program execution):**



```
B is greater than A
```

```
PC       = 40003c
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 10010000
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 1001001c
R5  [a1] = 7ffff9a0
R6  [a2] = 7ffff9a8
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 1
R10 [t2] = ffffffff
R11 [t3] = f
R12 [t4] = 1
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
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff99c
R30 [s8] = 0
```

```
                              User Text Segment [00400000]..[00440000]
[00400000] 3c011001  lui $1, 4097               ; 10: lw $t2, A
[00400004] 8c2a0000  lw $10, 0($1)
[00400008] 3c011001  lui $1, 4097               ; 11: lw $t3, B
[0040000c] 8c2b0004  lw $11, 4($1)
[00400010] 014b482a  slt $9, $10, $11           ; 12: slt $t1, $t2, $t3
[00400014] 014b402b  sltu $8, $10, $11          ; 13: sltu $t0, $t2, $t3
[00400018] 200c0001  addi $12, $0, 1            ; 14: addi $t4, $zero, 1
[0040001c] 34020004  ori $2, $0, 4              ; 16: li $v0, 4
[00400020] 11890003  beq $12, $9, 12 [printB-0x00400020]
[00400024] 15890000  bne $12, $9, 0 [printA-0x00400024]
[00400028] 0000000c  syscall                    ; 19: syscall
[0040002c] 34020004  ori $2, $0, 4              ; 22: li $v0, 4
[00400030] 3c011001  lui $1, 4097 [GB]          ; 23: la $a0, GB
[00400034] 3424001c  ori $4, $1, 28 [GB]
[00400038] 0000000c  syscall                    ; 24: syscall

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



```
B is greater than A
```

```
PC       = 40003c
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 10010000
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 1001001c
R5  [a1] = 7ffff9a0
R6  [a2] = 7ffff9a8
R7  [a3] = 0
R8  [t0] = 1
R9  [t1] = 1
R10 [t2] = ffffffff
R11 [t3] = f
R12 [t4] = 1
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
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff99c
R30 [s8] = 0
```

```
                              User Text Segment [00400000]..[00440000]
[00400000] 3c011001  lui $1, 4097               ; 10: lw $t2, A
[00400004] 8c2a0000  lw $10, 0($1)
[00400008] 3c011001  lui $1, 4097               ; 11: lw $t3, B
[0040000c] 8c2b0004  lw $11, 4($1)
[00400010] 014b482a  slt $9, $10, $11           ; 12: slt $t1, $t2, $t3
[00400014] 014b402a  slt $8, $10, $11           ; 13: slt $t0, $t2, $t3
[00400018] 200c0001  addi $12, $0, 1            ; 14: addi $t4, $zero, 1
[0040001c] 34020004  ori $2, $0, 4              ; 16: li $v0, 4
[00400020] 11890003  beq $12, $9, 12 [printB-0x00400020]
[00400024] 15890000  bne $12, $9, 0 [printA-0x00400024]
[00400028] 0000000c  syscall                    ; 19: syscall
[0040002c] 34020004  ori $2, $0, 4              ; 22: li $v0, 4
[00400030] 3c011001  lui $1, 4097 [GB]          ; 23: la $a0, GB
[00400034] 3424001c  ori $4, $1, 28 [GB]
[00400038] 0000000c  syscall                    ; 24: syscall

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

**EXPLANATION:**

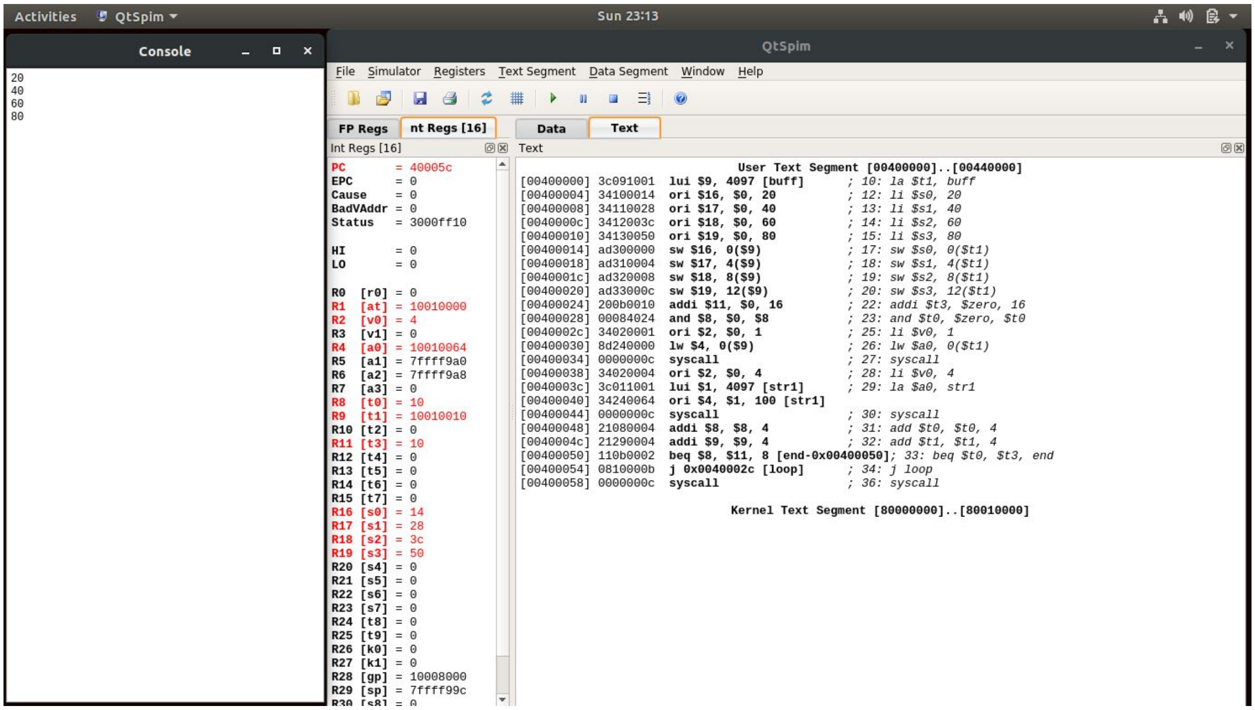
Firstly two numbers were compared as an unsigned numbers and in the later case they were compared as signed numbers. For that we need to make comparison operator from sltu to slt.

## (2) Introduction to arrays:

**CODE:**

```
#12::

.data

buff: .space 100
str1: .asciiz "\n"
.text

main:
la $t1, buff

li $s0, 20
li $s1, 40
li $s2, 60
li $s3, 80

sw $s0, 0($t1)
sw $s1, 4($t1)
sw $s2, 8($t1)
sw $s3, 12($t1)

addi $t3, $zero, 16
and $t0, $zero, $t0
loop:
li $v0, 1
lw $a0, 0($t1)
syscall
li $v0, 4
la $a0, str1
syscall
add $t0, $t0, 4
add $t1, $t1, 4
beq $t0, $t3, end
j loop
end:
syscall
```

**CHANGES IN REGISTERS/DATA SEGMENTS(Final Snapshots after program execution):**



```
20
40
60
80
```

```
PC       = 40005c
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 10010000
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 10010064
R5  [a1] = 7ffff9a0
R6  [a2] = 7ffff9a8
R7  [a3] = 0
R8  [t0] = 10
R9  [t1] = 10010010
R10 [t2] = 0
R11 [t3] = 10
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 14
R17 [s1] = 28
R18 [s2] = 3c
R19 [s3] = 50
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff99c
R30 [s8] = 0
```

```
                               User Text Segment [00400000]..[00440000]
[00400000] 3c091001  lui $9, 4097 [buff]      ; 10: la $t1, buff
[00400004] 34100014  ori $16, $0, 20          ; 12: li $s0, 20
[00400008] 34110028  ori $17, $0, 40          ; 13: li $s1, 40
[0040000c] 3412003c  ori $18, $0, 60          ; 14: li $s2, 60
[00400010] 34130050  ori $19, $0, 80          ; 15: li $s3, 80
[00400014] ad300000  sw $16, 0($9)            ; 17: sw $s0, 0($t1)
[00400018] ad310004  sw $17, 4($9)            ; 18: sw $s1, 4($t1)
[0040001c] ad320008  sw $18, 8($9)            ; 19: sw $s2, 8($t1)
[00400020] ad33000c  sw $19, 12($9)           ; 20: sw $s3, 12($t1)
[00400024] 200b0010  addi $11, $0, 16         ; 22: addi $t3, $zero, 16
[00400028] 00084024  and $8, $0, $8           ; 23: and $t0, $zero, $t0
[0040002c] 34020001  ori $2, $0, 1            ; 25: li $v0, 1
[00400030] 8d240000  lw $4, 0($9)             ; 26: lw $a0, 0($t1)
[00400034] 0000000c  syscall                  ; 27: syscall
[00400038] 34020004  ori $2, $0, 4            ; 28: li $v0, 4
[0040003c] 3c011001  lui $1, 4097 [str1]      ; 29: la $a0, str1
[00400040] 34240064  ori $4, $1, 100 [str1]
[00400044] 0000000c  syscall                  ; 30: syscall
[00400048] 21080004  addi $8, $8, 4           ; 31: add $t0, $t0, 4
[0040004c] 21290004  addi $9, $9, 4           ; 32: add $t1, $t1, 4
[00400050] 110b0002  beq $8, $11, 8 [end-0x00400050]; 33: beq $t0, $t3, end
[00400054] 0810000b  j 0x0040002c [loop]      ; 34: j loop
[00400058] 0000000c  syscall                  ; 36: syscall

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

**EXPLANATION:**

Following the simple process of loop initialization, repeated steps, and then the end procedural call.

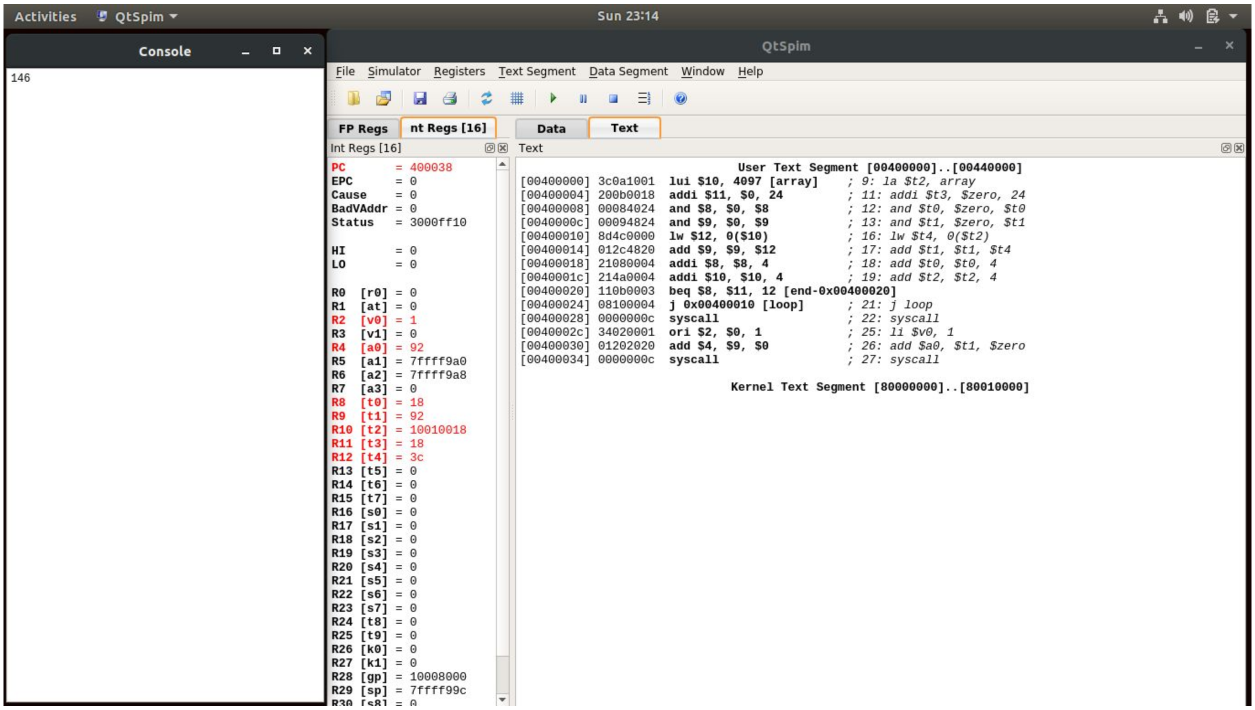
## (3) Compute the total age in the array:

**CODE:**

```
#l3::

.data

array: .word 1 3 20 30 32 60
.text

main:
la $t2, array

addi $t3, $zero, 24
and $t0, $zero, $t0
and $t1, $zero, $t1

loop:
lw $t4, 0($t2)
```

```
add $t1, $t1, $t4
add $t0, $t0, 4
add $t2, $t2, 4
beq $t0, $t3, end
j loop
syscall

end:
li $v0, 1
add $a0, $t1, $zero
syscall
```

**CHANGES IN REGISTERS/DATA SEGMENTS(Final Snapshots after program execution):**



```
146
```

```
PC       = 400038
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 1
R3  [v1] = 0
R4  [a0] = 92
R5  [a1] = 7ffff9a0
R6  [a2] = 7ffff9a8
R7  [a3] = 0
R8  [t0] = 18
R9  [t1] = 92
R10 [t2] = 10010018
R11 [t3] = 18
R12 [t4] = 3c
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
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff99c
R30 [s8] = 0
```

```
                        User Text Segment [00400000]..[00440000]
[00400000] 3c0a1001  lui $10, 4097 [array]    ; 9: la $t2, array
[00400004] 200b0018  addi $11, $0, 24         ; 11: addi $t3, $zero, 24
[00400008] 00084024  and $8, $0, $8           ; 12: and $t0, $zero, $t0
[0040000c] 00094824  and $9, $0, $9           ; 13: and $t1, $zero, $t1
[00400010] 8d4c0000  lw $12, 0($10)           ; 16: lw $t4, 0($t2)
[00400014] 012c4820  add $9, $9, $12          ; 17: add $t1, $t1, $t4
[00400018] 21080004  addi $8, $8, 4           ; 18: add $t0, $t0, 4
[0040001c] 214a0004  addi $10, $10, 4         ; 19: add $t2, $t2, 4
[00400020] 110b0003  beq $8, $11, 12 [end-0x00400020]
[00400024] 08100004  j 0x00400010 [loop]      ; 21: j loop
[00400028] 0000000c  syscall                  ; 22: syscall
[0040002c] 34020001  ori $2, $0, 1            ; 25: li $v0, 1
[00400030] 01202020  add $4, $9, $0           ; 26: add $a0, $t1, $zero
[00400034] 0000000c  syscall                  ; 27: syscall

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

**EXPLANATION:**

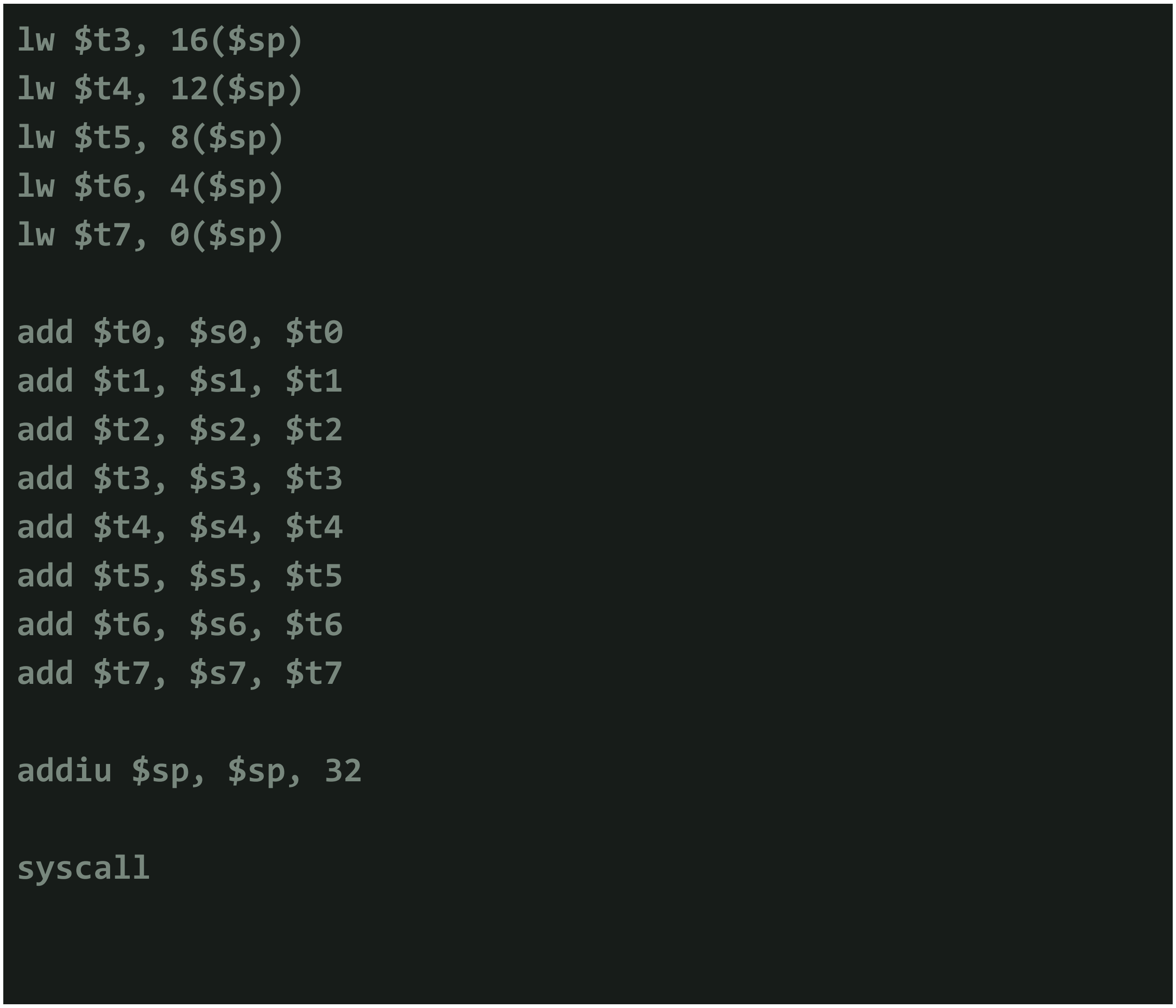
The idea is to store the elements as the data elements in .word format and extract them in a loop.

## (4) Introduction to stack operations:

**CODE:**

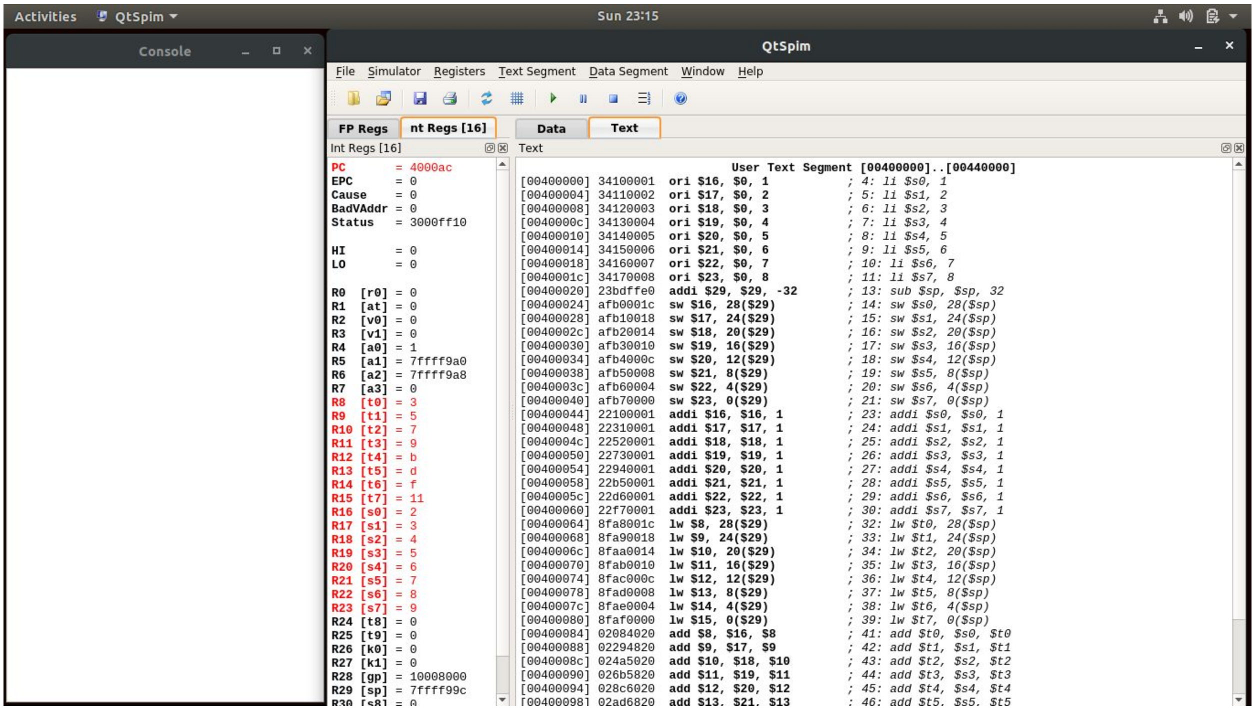
```
#14:
```

```
.text
li $s0, 1
li $s1, 2
li $s2, 3
li $s3, 4
li $s4, 5
li $s5, 6
li $s6, 7
li $s7, 8

sub $sp, $sp, 32
sw $s0, 28($sp)
sw $s1, 24($sp)
sw $s2, 20($sp)
sw $s3, 16($sp)
sw $s4, 12($sp)
sw $s5, 8($sp)
sw $s6, 4($sp)
sw $s7, 0($sp)

addi $s0, $s0, 1
addi $s1, $s1, 1
addi $s2, $s2, 1
addi $s3, $s3, 1
addi $s4, $s4, 1
addi $s5, $s5, 1
addi $s6, $s6, 1
addi $s7, $s7, 1

lw $t0, 28($sp)
lw $t1, 24($sp)
lw $t2, 20($sp)
```



```
lw $t3, 16($sp)
lw $t4, 12($sp)
lw $t5, 8($sp)
lw $t6, 4($sp)
lw $t7, 0($sp)

add $t0, $s0, $t0
add $t1, $s1, $t1
add $t2, $s2, $t2
add $t3, $s3, $t3
add $t4, $s4, $t4
add $t5, $s5, $t5
add $t6, $s6, $t6
add $t7, $s7, $t7

addiu $sp, $sp, 32

syscall
```

**CHANGES IN REGISTERS/DATA SEGMENTS(Final Snapshots after program execution):**



```
PC       = 4000ac
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 7ffff9a0
R6  [a2] = 7ffff9a8
R7  [a3] = 0
R8  [t0] = 3
R9  [t1] = 5
R10 [t2] = 7
R11 [t3] = 9
R12 [t4] = b
R13 [t5] = d
R14 [t6] = f
R15 [t7] = 11
R16 [s0] = 2
R17 [s1] = 3
R18 [s2] = 4
R19 [s3] = 5
R20 [s4] = 6
R21 [s5] = 7
R22 [s6] = 8
R23 [s7] = 9
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff99c
R30 [s8] = 0
```

```
                              User Text Segment [00400000]..[00440000]
[00400000] 34100001  ori $16, $0, 1            ; 4: li $s0, 1
[00400004] 34110002  ori $17, $0, 2            ; 5: li $s1, 2
[00400008] 34120003  ori $18, $0, 3            ; 6: li $s2, 3
[0040000c] 34130004  ori $19, $0, 4            ; 7: li $s3, 4
[00400010] 34140005  ori $20, $0, 5            ; 8: li $s4, 5
[00400014] 34150006  ori $21, $0, 6            ; 9: li $s5, 6
[00400018] 34160007  ori $22, $0, 7            ; 10: li $s6, 7
[0040001c] 34170008  ori $23, $0, 8            ; 11: li $s7, 8
[00400020] 23bdffe0  addi $29, $29, -32        ; 13: sub $sp, $sp, 32
[00400024] afb0001c  sw $16, 28($29)           ; 14: sw $s0, 28($sp)
[00400028] afb10018  sw $17, 24($29)           ; 15: sw $s1, 24($sp)
[0040002c] afb20014  sw $18, 20($29)           ; 16: sw $s2, 20($sp)
[00400030] afb30010  sw $19, 16($29)           ; 17: sw $s3, 16($sp)
[00400034] afb4000c  sw $20, 12($29)           ; 18: sw $s4, 12($sp)
[00400038] afb50008  sw $21, 8($29)            ; 19: sw $s5, 8($sp)
[0040003c] afb60004  sw $22, 4($29)            ; 20: sw $s6, 4($sp)
[00400040] afb70000  sw $23, 0($29)            ; 21: sw $s7, 0($sp)
[00400044] 22100001  addi $16, $16, 1          ; 23: addi $s0, $s0, 1
[00400048] 22310001  addi $17, $17, 1          ; 24: addi $s1, $s1, 1
[0040004c] 22520001  addi $18, $18, 1          ; 25: addi $s2, $s2, 1
[00400050] 22730001  addi $19, $19, 1          ; 26: addi $s3, $s3, 1
[00400054] 22940001  addi $20, $20, 1          ; 27: addi $s4, $s4, 1
[00400058] 22b50001  addi $21, $21, 1          ; 28: addi $s5, $s5, 1
[0040005c] 22d60001  addi $22, $22, 1          ; 29: addi $s6, $s6, 1
[00400060] 22f70001  addi $23, $23, 1          ; 30: addi $s7, $s7, 1
[00400064] 8fa8001c  lw $8, 28($29)            ; 32: lw $t0, 28($sp)
[00400068] 8fa90018  lw $9, 24($29)            ; 33: lw $t1, 24($sp)
[0040006c] 8faa0014  lw $10, 20($29)           ; 34: lw $t2, 20($sp)
[00400070] 8fab0010  lw $11, 16($29)           ; 35: lw $t3, 16($sp)
[00400074] 8fac000c  lw $12, 12($29)           ; 36: lw $t4, 12($sp)
[00400078] 8fad0008  lw $13, 8($29)            ; 37: lw $t5, 8($sp)
[0040007c] 8fae0004  lw $14, 4($29)            ; 38: lw $t6, 4($sp)
[00400080] 8faf0000  lw $15, 0($29)            ; 39: lw $t7, 0($sp)
[00400084] 02084020  add $8, $16, $8           ; 41: add $t0, $s0, $t0
[00400088] 02294820  add $9, $17, $9           ; 42: add $t1, $s1, $t1
[0040008c] 024a5020  add $10, $18, $10         ; 43: add $t2, $s2, $t2
[00400090] 026b5820  add $11, $19, $11         ; 44: add $t3, $s3, $t3
[00400094] 028c6020  add $12, $20, $12         ; 45: add $t4, $s4, $t4
[00400098] 02ad6820  add $13, $21, $13         ; 46: add $t5, $s5, $t5
```

**EXPLANATION:**

Following simple stack operations and extracting elements in reverse order.

## (5) Floating point arithmetic:

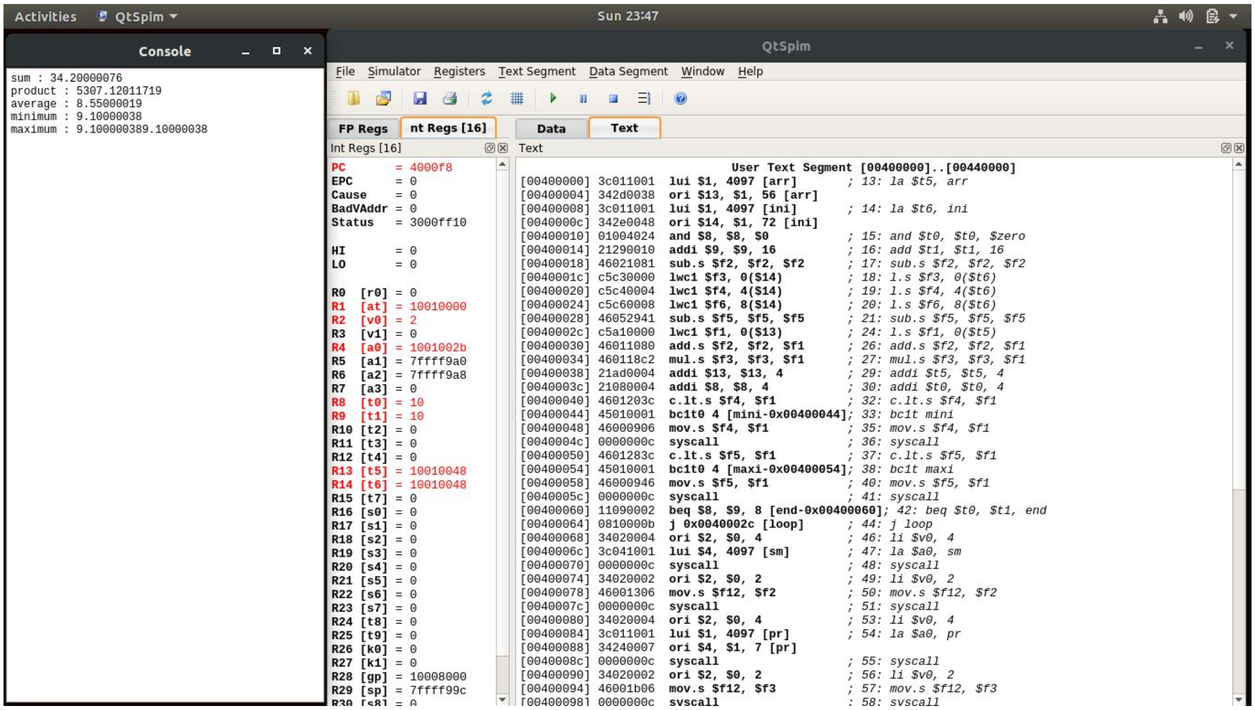
**CODE:**

```
#l5:

.data
sm: .asciiz "sum : "
pr: .asciiz "\nproduct : "
av: .asciiz "\naverage : "
mi: .asciiz "\nminimum : "
ma: .asciiz "\nmaximum : "
arr: .float 8.0 8.1 9.0 9.1
ini: .float 1.0 10.0 4.0

.text
la $t5, arr
la $t6, ini
and $t0, $t0, $zero
add $t1, $t1, 16
sub.s $f2, $f2, $f2
l.s $f3, 0($t6)
l.s $f4, 4($t6)
l.s $f6, 8($t6)
sub.s $f5, $f5, $f5

loop:
l.s $f1, 0($t5)

add.s $f2, $f2, $f1
mul.s $f3, $f3, $f1

addi $t5, $t5, 4
addi $t0, $t0, 4

c.lt.s $f4, $f1
bc1t mini
mini:
mov.s $f4, $f1
syscall

c.lt.s $f5, $f1
bc1t maxi
maxi:
mov.s $f5, $f1
syscall
beq $t0, $t1, end

j loop
```

```
end: li $v0, 4
la $a0, sm
syscall
li $v0, 2
mov.s $f12, $f2
syscall

li $v0, 4
la $a0, pr
syscall
li $v0, 2
mov.s $f12, $f3
syscall

li $v0, 4
la $a0, av
syscall
li $v0, 2
div.s $f2 ,$f2, $f6
mov.s $f12, $f2
syscall

li $v0, 4
la $a0, mi
syscall
li $v0, 2
mov.s $f12, $f4
syscall

li $v0, 4
la $a0, ma
```

```
syscall
li $v0, 2
mov.s $f12, $f5
syscall
syscall
```

**CHANGES IN REGISTERS/DATA SEGMENTS(Final Snapshots after program execution):**



```
sum : 34.20000076
product : 5307.12011719
average : 8.55000019
minimum : 9.10000038
maximum : 9.100000389.10000038
```

```
PC       = 4000f8
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 10010000
R2  [v0] = 2
R3  [v1] = 0
R4  [a0] = 1001002b
R5  [a1] = 7ffff9a0
R6  [a2] = 7ffff9a8
R7  [a3] = 0
R8  [t0] = 10
R9  [t1] = 10
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 10010048
R14 [t6] = 10010048
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff99c
R30 [s8] = 0
```

```
                        User Text Segment [00400000]..[00440000]
[00400000] 3c011001  lui $1, 4097 [arr]       ; 13: la $t5, arr
[00400004] 342d0038  ori $13, $1, 56 [arr]
[00400008] 3c011001  lui $1, 4097 [ini]       ; 14: la $t6, ini
[0040000c] 342e0048  ori $14, $1, 72 [ini]
[00400010] 01004024  and $8, $8, $0           ; 15: and $t0, $t0, $zero
[00400014] 21290010  addi $9, $9, 16          ; 16: add $t1, $t1, 16
[00400018] 46021081  sub.s $f2, $f2, $f2      ; 17: sub.s $f2, $f2, $f2
[0040001c] c5c30000  lwc1 $f3, 0($14)         ; 18: l.s $f3, 0($t6)
[00400020] c5c40004  lwc1 $f4, 4($14)         ; 19: l.s $f4, 4($t6)
[00400024] c5c60008  lwc1 $f6, 8($14)         ; 20: l.s $f6, 8($t6)
[00400028] 46052941  sub.s $f5, $f5, $f5      ; 21: sub.s $f5, $f5, $f5
[0040002c] c5a10000  lwc1 $f1, 0($13)         ; 24: l.s $f1, 0($t5)
[00400030] 46011080  add.s $f2, $f2, $f1      ; 26: add.s $f2, $f2, $f1
[00400034] 460118c2  mul.s $f3, $f3, $f1      ; 27: mul.s $f3, $f3, $f1
[00400038] 21ad0004  addi $13, $13, 4         ; 29: addi $t5, $t5, 4
[0040003c] 21080004  addi $8, $8, 4           ; 30: addi $t0, $t0, 4
[00400040] 4601203c  c.lt.s $f4, $f1          ; 32: c.lt.s $f4, $f1
[00400044] 45010001  bc1t0 4 [mini-0x00400044]; 33: bc1t mini
[00400048] 46000906  mov.s $f4, $f1           ; 35: mov.s $f4, $f1
[0040004c] 0000000c  syscall                  ; 36: syscall
[00400050] 4601283c  c.lt.s $f5, $f1          ; 37: c.lt.s $f5, $f1
[00400054] 45010001  bc1t0 4 [maxi-0x00400054]; 38: bc1t maxi
[00400058] 46000946  mov.s $f5, $f1           ; 40: mov.s $f5, $f1
[0040005c] 0000000c  syscall                  ; 41: syscall
[00400060] 11090002  beq $8, $9, 8 [end-0x00400060]; 42: beq $t0, $t1, end
[00400064] 0810000b  j 0x0040002c [loop]      ; 44: j loop
[00400068] 34020004  ori $2, $0, 4            ; 46: li $v0, 4
[0040006c] 3c041001  lui $4, 4097 [sm]        ; 47: la $a0, sm
[00400070] 0000000c  syscall                  ; 48: syscall
[00400074] 34020002  ori $2, $0, 2            ; 49: li $v0, 2
[00400078] 46001306  mov.s $f12, $f2          ; 50: mov.s $f12, $f2
[0040007c] 0000000c  syscall                  ; 51: syscall
[00400080] 34020004  ori $2, $0, 4            ; 53: li $v0, 4
[00400084] 3c011001  lui $1, 4097 [pr]        ; 54: la $a0, pr
[00400088] 34240007  ori $4, $1, 7 [pr]
[0040008c] 0000000c  syscall                  ; 55: syscall
[00400090] 34020002  ori $2, $0, 2            ; 56: li $v0, 2
[00400094] 46001b06  mov.s $f12, $f3          ; 57: mov.s $f12, $f3
[00400098] 0000000c  syscall                  ; 58: syscall
```

**EXPLANATION:**
**The process requires simple floating point operations in a loop and using special operations (.s) for single point float operations, bc1t for calling the procedural function if the above respective conditional statements are satisfied.**

# Observations

Key observations in performing the above executions in mips:

1. For comparing signed integers we need to use the slt operation, while for the unsigned once, we need to use sltu operation.

2. Asciiz is different from ascii, as asciiz is terminated by '\0' and hence it is often referred as C-string.

3. For extracting out elements and putting elements in the stack, we need to work in reverse index order.
4. Bc1t operation enables procedural call, provided the preceding condition is satisfied.
5. C.lt.s. Enables single precision floating point less than comparision.
6. “.s ” instruction enables single precision floating point arithmetic operations.

## Conclusions

We dealt with the basics of MIPS assembly programming using the QtSpim simulator and performed operations with arrays, floating point numbers, and arithmetic operations.