Descriptions:

In this code initializes three integers variables $t0, $t1 and $t2

With the values 300, 200, and 100 respectively. Then add the values of $t0,, $t1 and $t2 together and the sum of values stores in $t4. This sum of values stored on the stack at the address 12bytes above the stack pointer ($sp).

The program then enter a loop label “display” which repeatedly display the values of $t4 to the console using the “syscall” instruction with the code 1 ($v0 , set to 1).

Finally the program terminates when it reached the “exit” label and uses the “syscall” instruction with the code 10 ($v0 set to 10) to terminate the program.

Logic:

At start I initialize some variables , performs a calculation, stores the result and the display the result repeatedly until the program is terminate.

Code:

```
.data .text
.globl main
main:
li $t0,300
li $t1,200
li $t2,100
li $t3,0
li $s0,3
add $t4,$t0,$t1
add $t4,$t4,$t2
sw $t4,12($sp)
display:
move $a0,$t4
li $v0, 1
syscall
exit:
li $v0,10
syscall
```

Question no 2:

Description:

This code calculates the sum of elements in an array and passes the sum, array size, and array itself to a function called **Final_calculation**, which calculates the sum again and returns the value. The final sum is then displayed on the console.

The **.data** section initializes an array named **Array** with five elements and a variable named **size** with a value of 5. It also initializes a variable named **FinalResult** with a value of 0.

The **.text** section contains the **main** function, which loads the address of **Array** into **$t0**, loads the value of **size** into **$t1**, and initializes **$t2** and **$t3** to 0. The first loop (**FirstLoop**) adds up the elements of **Array** and stores the result in **$t2**. The sum, **size**, and address of **Array** are then passed as arguments to **Final_calculation** via **$a0**, **$a1**, and **$a2**, respectively, and the result is stored in **FinalResult**. The final result is then displayed on the console.

The **Final_calculation** function initializes **$t2** and **$t3** to 0 and loads the address of **Array** into **$t0**. It then calculates the sum of the elements in **Array** using a loop (**Second_Loop**) and stores the result in **$t2**. Finally, it returns the sum stored in **$s0**.

The code uses the following MIPS assembly language instructions: **la**, **lw**, **li**, **addi**, **beq**, **add**, **j**, **move**, **jal**, **sw**, **syscall**, **jr**.

Code:

data

Array: .word 1, 2, 3, 4, 5

size: .word 5

FinalResult: .word 0

```
.text
.globl main
main:
 la $t0, Array
 lw $t1, size
 li $t2, 0
 addi $t3, $zero, 0
FirstLoop:
 beq $t3, $t1, EndFirstLoop
 lw $t4, 0($t0)
 add $t2, $t2, $t4
 addi $t0, $t0, 4
 addi $t3, $t3, 1
 j FirstLoop
EndFirstLoop:
 move $a0, $t2
 move $a1, $t1
 la $a2, Array
 jal Final_calculation
 sw $v0, FinalResult
display:
 li $v0, 1
 lw $a0, FinalResult
 syscall
```

```
Exit:
 li $v0, 10
 syscall
Final_calculation:
 addi $sp, $sp, -8
 sw $ra, 0($sp)
 sw $s0, 4($sp)
 move $s0, $a0
 move $t0, $a2
 move $t1, $a1
 li $t2, 0
 addi $t3, $zero, 0
Second_Loop:
 beq $t3, $t1, EndSecond_Loop
 lw $t4, 0($t0)
 add $t2, $t2, $t4
 addi $t0, $t0, 4
 addi $t3, $t3, 1
 j Second_Loop
EndSecond_Loop:
 move $v0, $s0
 lw $ra, 0($sp)
 lw $s0, 4($sp)
 addi $sp, $sp, 8
 jr $ra
```