MIPS PROJECT

BBM234
COMPUTER ORGANIZATION

Name and Surname : Meltem Kaya
Student ID : 21827555
Due Date : 17/04/2021

HACETTEPE
University

Department of Computer Engineering

–

Instructor
Prof. Dr. Süleyman Tosun

Teaching Assistant
Selma Dilek

Table of Contents

Problem Explanation

Hababam Class was a very cunning and copy-prone class. That's why Külyutmaz Necmi decided to make a mips project to prevent cheating in exams.

The features will be as follows:

1. Students will be registered in the system with an id instead of their names.
2. Similarity scores will be encrypted in such a way that the even numbers will be multiplied by 8, while the odd numbers will be divided by 5.
3. Finally, the encrypted similarity scores will be stored as a row-major 1-D implementation of an upper triangular matrix which holds the similarity scores between each pair of students

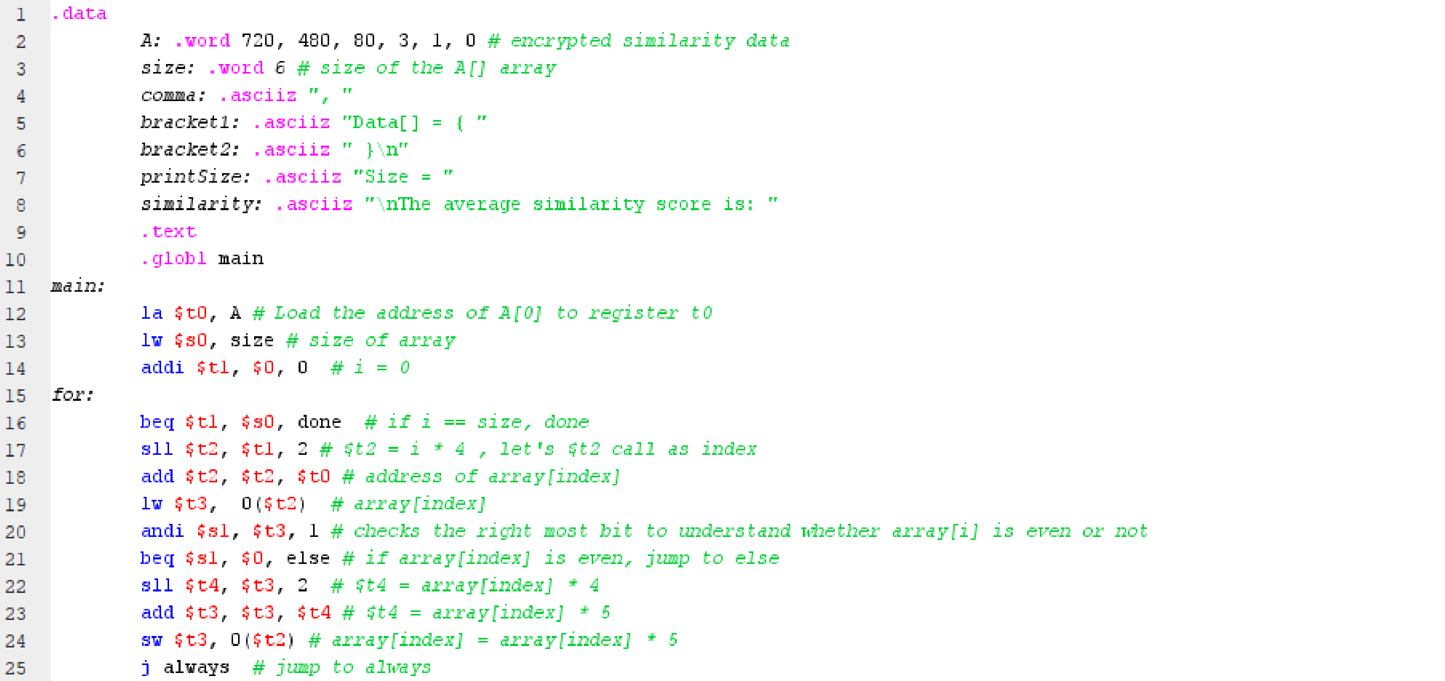
The mission is to tackle these problems one at a time and detect cheaters.
In this project, we will calculate the average similarity score of students.

The steps followed for calculation is as follows:

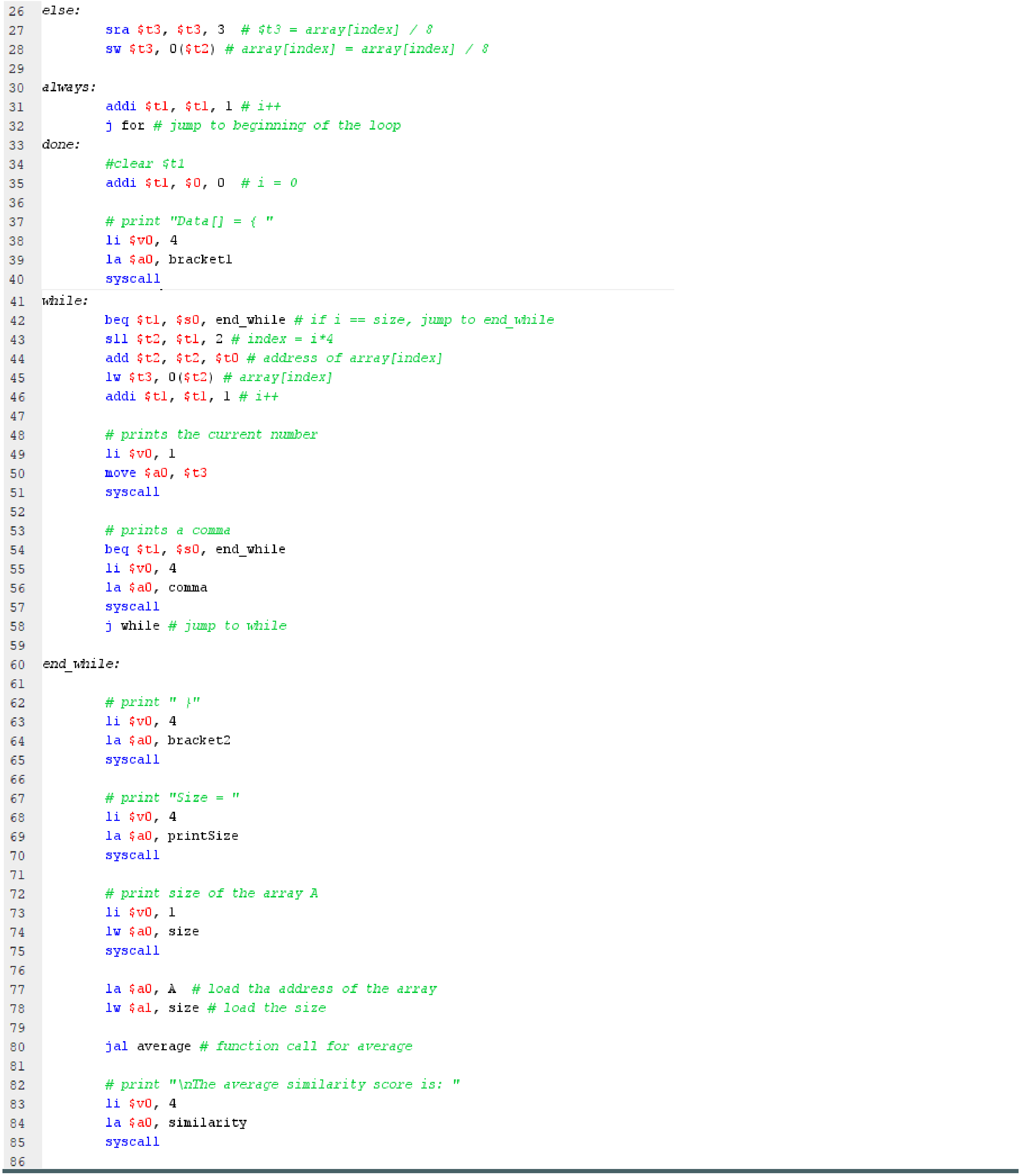
1. Decrypt each similarity score of students.
2. Calculate the average similarity score by given formula below:

$$AVG(n) = \frac{(AVG(n-1) \times (n-1)) + n^{th}element}{n}$$

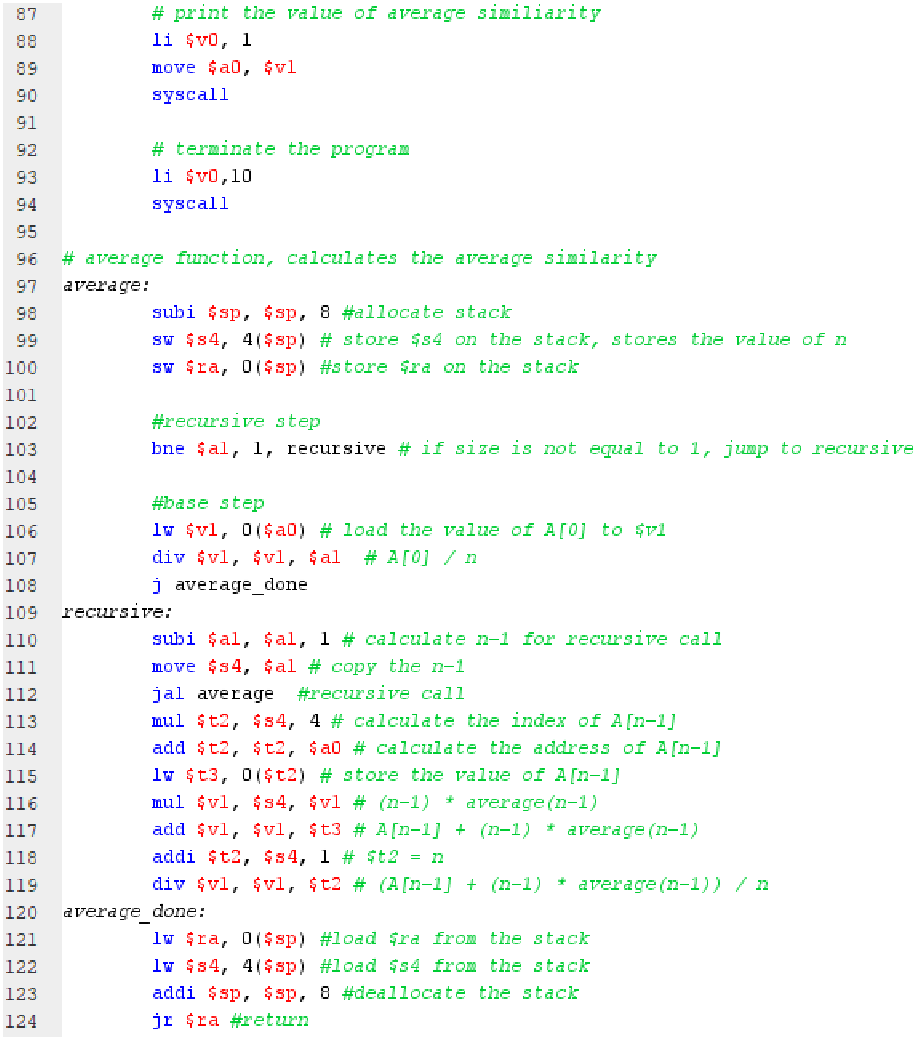
Code Fragments



```
.data
        A: .word 720, 480, 80, 3, 1, 0 # encrypted similarity data
        size: .word 6 # size of the A[] array
        comma: .asciiz ", "
        bracket1: .asciiz "Data[] = { "
        bracket2: .asciiz " }\n"
        printSize: .asciiz "Size = "
        similarity: .asciiz "\nThe average similarity score is: "
        .text
        .globl main
main:
        la $t0, A # Load the address of A[0] to register t0
        lw $s0, size # size of array
        addi $t1, $0, 0  # i = 0
for:
        beq $t1, $s0, done  # if i == size, done
        sll $t2, $t1, 2 # $t2 = i * 4 , let's $t2 call as index
        add $t2, $t2, $t0 # address of array[index]
        lw $t3,  0($t2)  # array[index]
        andi $s1, $t3, 1 # checks the right most bit to understand whether array[i] is even or not
        beq $s1, $0, else # if array[index] is even, jump to else
        sll $t4, $t3, 2  # $t4 = array[index] * 4
        add $t3, $t3, $t4 # $t4 = array[index] * 5
        sw $t3, 0($t2) # array[index] = array[index] * 5
        j always  # jump to always
```



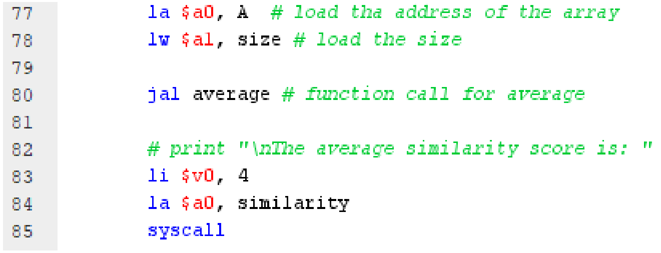
```
else:
        sra $t3, $t3, 3  # $t3 = array[index] / 8
        sw $t3, 0($t2) # array[index] = array[index] / 8

always:
        addi $t1, $t1, 1 # i++
        j for # jump to beginning of the loop
done:
        #clear $t1
        addi $t1, $0, 0  # i = 0

        # print "Data[] = { "
        li $v0, 4
        la $a0, bracket1
        syscall
while:
        beq $t1, $s0, end_while # if i == size, jump to end_while
        sll $t2, $t1, 2 # index = i*4
        add $t2, $t2, $t0 # address of array[index]
        lw $t3, 0($t2) # array[index]
        addi $t1, $t1, 1 # i++

        # prints the current number
        li $v0, 1
        move $a0, $t3
        syscall

        # prints a comma
        beq $t1, $s0, end_while
        li $v0, 4
        la $a0, comma
        syscall
        j while # jump to while

end_while:

        # print " }"
        li $v0, 4
        la $a0, bracket2
        syscall

        # print "Size = "
        li $v0, 4
        la $a0, printSize
        syscall

        # print size of the array A
        li $v0, 1
        lw $a0, size
        syscall

        la $a0, A  # load tha address of the array
        lw $a1, size # load the size

        jal average # function call for average

        # print "\nThe average similarity score is: "
        li $v0, 4
        la $a0, similarity
        syscall

```



```
        # print the value of average similiarity
        li $v0, 1
        move $a0, $v1
        syscall

        # terminate the program
        li $v0,10
        syscall

# average function, calculates the average similarity
average:
        subi $sp, $sp, 8 #allocate stack
        sw $s4, 4($sp) # store $s4 on the stack, stores the value of n
        sw $ra, 0($sp) #store $ra on the stack

        #recursive step
        bne $a1, 1, recursive # if size is not equal to 1, jump to recursive

        #base step
        lw $v1, 0($a0) # load the value of A[0] to $v1
        div $v1, $v1, $a1  # A[0] / n
        j average_done
recursive:
        subi $a1, $a1, 1 # calculate n-1 for recursive call
        move $s4, $a1 # copy the n-1
        jal average  #recursive call
        mul $t2, $s4, 4 # calculate the index of A[n-1]
        add $t2, $t2, $a0 # calculate the address of A[n-1]
        lw $t3, 0($t2) # store the value of A[n-1]
        mul $v1, $s4, $v1 # (n-1) * average(n-1)
        add $v1, $v1, $t3 # A[n-1] + (n-1) * average(n-1)
        addi $t2, $s4, 1 # $t2 = n
        div $v1, $v1, $t2 # (A[n-1] + (n-1) * average(n-1)) / n
average_done:
        lw $ra, 0($sp) #load $ra from the stack
        lw $s4, 4($sp) #load $s4 from the stack
        addi $sp, $sp, 8 #deallocate the stack
        jr $ra #return
```

Explanation of Jal Instructions

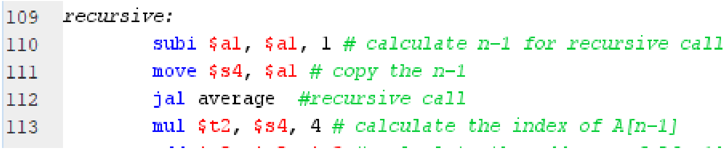
1. Usage



```
        la $a0, A  # load tha address of the array
        lw $a1, size # load the size

        jal average # function call for average

        # print "\nThe average similarity score is: "
        li $v0, 4
        la $a0, similarity
        syscall
```

The first usage of jump and link instruction is to calculate the average similarity score. That calls average function parameters with $a0 which stores the address of the data array and $a1 which stores size of the data array. Average function returns the average similarity score by $v1 register.

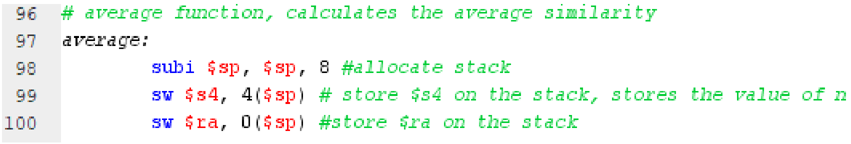
2. Usage



```
recursive:
        subi $a1, $a1, 1 # calculate n-1 for recursive call
        move $s4, $a1 # copy the n-1
        jal average  #recursive call
        mul $t2, $s4, 4 # calculate the index of A[n-1]
```

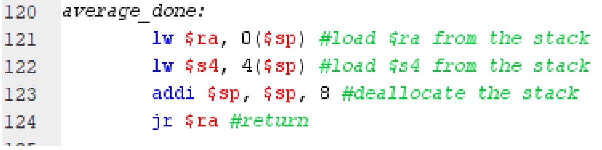
The second usage of jump and link instruction is a recursive call of average function. If the function parameters do not provide the base step, then call average function recursively. This call will be ended if the base step is provided.

Explanation of Stack Usage



```
# average function, calculates the average similarity
average:
        subi $sp, $sp, 8 #allocate stack
        sw $s4, 4($sp) # store $s4 on the stack, stores the value of n
        sw $ra, 0($sp) #store $ra on the stack
```

Firstly, I allocated the stack. Each register costs 4 bytes and total amount of the stack size which would be allocated is 8. Then I stored the values which are $ra and $s4 by giving them an address. When the function is called, it creates a new frame onto the stack, which will be used for local storage. I used the stack for saving passing parameters and return values.
$ra is the return value, it returns the final value stored in $v1.



```
average_done:
        lw $ra, 0($sp) #load $ra from the stack
        lw $s4, 4($sp) #load $s4 from the stack
        addi $sp, $sp, 8 #deallocate the stack
        jr $ra #return
```

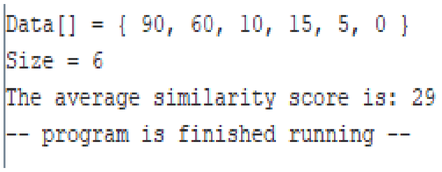
Lastly, I deallocated the stack before the average function returns because before the function returns, it must pop its stack frame, to restore the stack to its original state.

Output Tests

Test Case:

Number of students = 4
Data size = 6
Data[] = {720, 480, 80, 3, 1, 0}

Output:

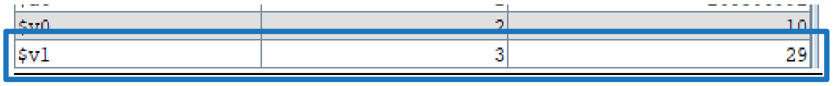


```
Data[] = { 90, 60, 10, 15, 5, 0 }
Size = 6
The average similarity score is: 29
-- program is finished running --
```

The number of students is 4 and the number of similiarity data is 6. Similarity scores are given in an array as stated in above.
After decrypting the input data, Actual similarity values are = "90, 60, 10, 15, 5, 0"
The average similarity score is 29.

Registers:

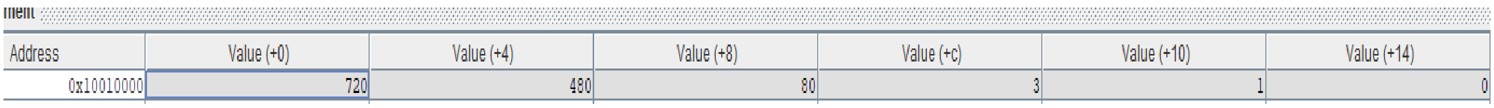
Return Register



$v0	2	10
$v1	3	29

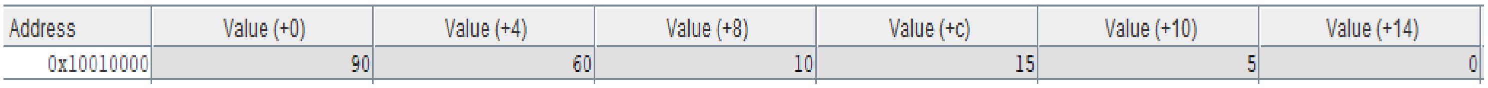
$v1 is used for returning the average similarity score.
The final result of the recursive function is stored in register v1, and output to the console.

Before Decrypting the Data[] array:



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x10010000	720	480	80	3	1	0

After Decrypting the Data[] array:



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x10010000	90	60	10	15	5	0

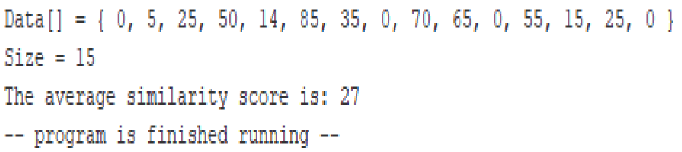
As we can see, encrypted values are stored after decryption in the same memory location. They has the same values and same order with inputs and outputs.

Test Case:

Number of students = 6
Data size = 15
Data[] = {0, 1, 5, 400, 112, 17, 7, 0, 560, 13, 0, 11, 3, 5, 0}

The number of students is 6 and the number of similiarity data is 15. Similarity scores are given in an array as stated in above.

Output:

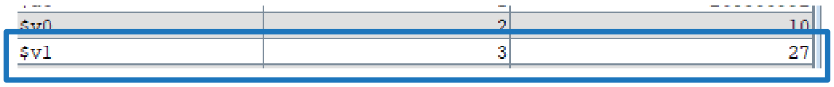


```
Data[] = { 0, 5, 25, 50, 14, 85, 35, 0, 70, 65, 0, 55, 15, 25, 0 }
Size = 15
The average similarity score is: 27
-- program is finished running --
```

After decrypting the input data, Actual similarity values are stated above.
The average similarity score is 27.

Registers:

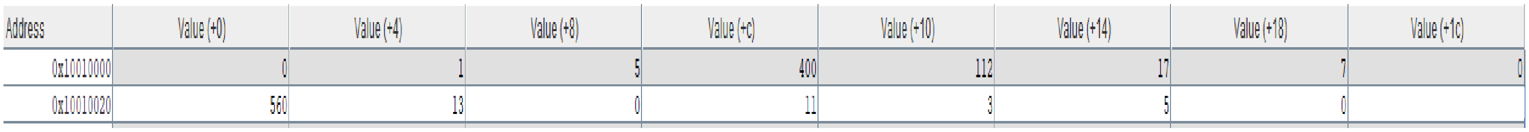
Return Register



$v0	2	10
$v1	3	27

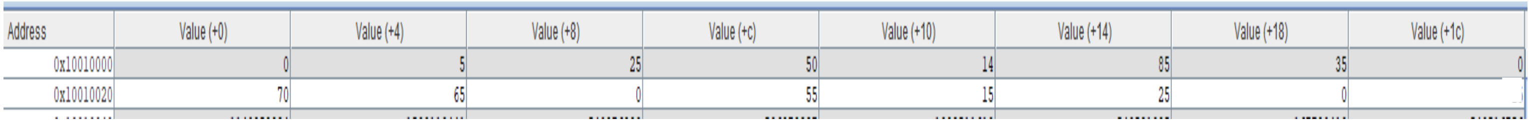
$v1 is used for returning the average similarity score.
The final result of the recursive function is stored in register v1, and output to the console.

Before Decrypting the Data[] array:



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	1	5	400	112	17	7	0
0x10010020	560	13	0	11	3	5	0	

After Decrypting the Data[] array:



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	5	25	50	14	85	35	0
0x10010020	70	65	0	55	15	25	0	

As we can see, encrypted values are stored after decryption in the same memory location. They has the same values and same order with inputs and outputs.

References

Hacettepe University Computer Engineering Lesson: BBM234 slides
Hacettepe University Computer Engineering Lesson: Video records

Youtube videos:
https://www.youtube.com/watch?v=3napwKvocSU&list=RDCMUCPZ473Q4kbG98JmL71PgXTA&start_radio=1&t=771
https://www.youtube.com/watch?v=_KLfGJRI5_Q&list=RDCMUCPZ473Q4kbG98JmL71PgXTA&index=5
https://www.youtube.com/watch?v=0aexcR9CNcE&list=RDCMUCPZ473Q4kbG98JmL71PgXTA&index=7
https://www.youtube.com/watch?v=bQC1PqGLwmo&t=407s
https://www.youtube.com/watch?v=B6ky4Weahm4&t=909s