Hacettepe University

Computer Engineering Department

BM234 Computer Organization - 2021 Spring

MIPS Project Report

April 17, 2021

Student name:
Şevval ATMACA

Student Number:
b21827115

1 Problem Definition

Our goal in this project is to get the similarity scores and change them as such encrypt the scores and calculate average for helping Külyutmaz detect cheaters.

2 Solution Implementation

```
1
2  addi $s2, $0, 0 # i = 0 : i value that for first function
3
4
5  .data
6  A: .word 720, 480, 80, 3, 1, 0 #input array
7  str1 : .asciiz "The average similarity score is: " #The string is average value to
       print to the console
8      .text
9      .globl main
10
11
12 main:
13     la $t1, A  # Load the address of A[0] to register t1
14     jal loop
15     #finish first function
16
17     #second function
18     la $a0, A # Load the address of A[0] to register a0
19     move $a1, $s0 # save the size of array to a1
20     jal average # call average
21
22     li $v0, 4 # to print string
23     la $a0, str1 # print str1 to the console
24     syscall
25
26     move $a0, $v1 # save the average value and print to the console
27     li $v0, 1
28     syscall
29
30     li $v0, 10 # exit
31     syscall
32
33
34 # $s0 = int Datasize;
35 # $s2 = i
36
37 loop:
38     slt $t0, $s2, $s0 #compare i and datasize. If i < datasize then 1. Otherwise 0.
39     beq $t0, $0, done #if i>= datasize then break the loop
40
41     sll $t0, $s2, 2 #4i
42     add $t0, $t0, $t1 # the adress of i = 4i + adress of data [0]
```

```
43      lw $t3, 0($t0) # t3 = data[i]
44
45      rem $t4, $t3, 2 # t4 = data[i] % 2
46      bne $t4,0, loop_else  # if data[i] % 2 != 0 then go to else block
47      sra $t3, $t3, 3 # otherwise t3 = data[i] * 8
48
49  L1:
50      sw $t3, 0($t0) # data[i] = t3
51      addi $s2 $s2 1 # increase the "i"
52      j loop # go back the loop the calculate other index
53
54  loop_else:
55      sll $t5, $t3, 2 # t5 = t3 * 4
56      sll $t6, $t3, 0 # t6 = t3 * 1
57      addu $t3, $t5, $t6 # t3 = t5 + t6 = 5 * t3
58      j L1
59
60
61  done:
62      jr $ra
63       # return main and execute orher function
64
65
66
67  # $a0 : int[] data
68  # $a1 : n
69  # $s0 : sum
70  # $s2: average
71
72
73  average:
74
75      addi $sp, $sp, -12 #make space on stack
76      sw $s3, 0($sp) # save s3(data[n-1]) on stack because overwrite
77      sw $a1, 8($sp) # save a1(n) on stack
78      sw $ra, 4($sp) # save $ra on stack
79
80      bne $a1, 1, else_average # if n != 1 go to else
81      lw $s0, 0($a0) # sum = data[0]
82      jal L2
83
84
85  else_average:
86
87      addi $t0, $a1, -1 # t0 = n - 1
88
89      sll $t0, $t0, 2
90      addu $t0, $t0, $a0
91      lw $s3 0($t0) # s3 = data[n-1]
92
93      addi $a1, $a1, -1 # n = n - 1
94
```

```
95      jal average # call recursive as a average_recursive(data, n - 1)
96
97      lw $a1, 8($sp) # save n to calculate sum
98
99      addi $t3, $a1, -1 # t3 = n - 1
100     mul $t5, $t3, $v0 #  t3 * v0 = (n - 1)* average_recursive(data, n - 1)
101     addu $s0, $s3, $t5 # data[n - 1] + (n - 1)* average_recursive(data, n - 1)
102
103
104 L2:
105     div $s2, $s0, $a1 # average = sum / n;
106     move $v0, $s2 # save as a return
107     move $v1, $s2 #save v1
108
109     lw $s3, 0($sp) # restore on stack
110     lw $a1, 8($sp)
111     lw $ra, 4($sp)
112     addi $sp, $sp, 12 #deallocate stack space
113     jr $ra # return to caller
```

2.1 Jal

I used "jal" for jumping and linking. Firstly, I used it when calling functions in main. Secondly, I used "jal" to calculate average in another block after going to each if and else for linking. Thirdly, I used "jal" to call the averagerecursive function recursively in the else block.

2.2 Stack

I used stack for store the elements into the stack and use after return and for data[n-1], n and ra values in each recursion. I keep these values to calculate the sum or average value after each recursive.

3 Results

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	720	480	80	3	1	0	543516756	1919252065
0x10010020	543516513	1768778099	1769103724	1931508084	1701998435	980642080	32	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0

Figure 1: Data Segment before Testing

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194304
hi		0
lo		0

Figure 2: Register before Testing

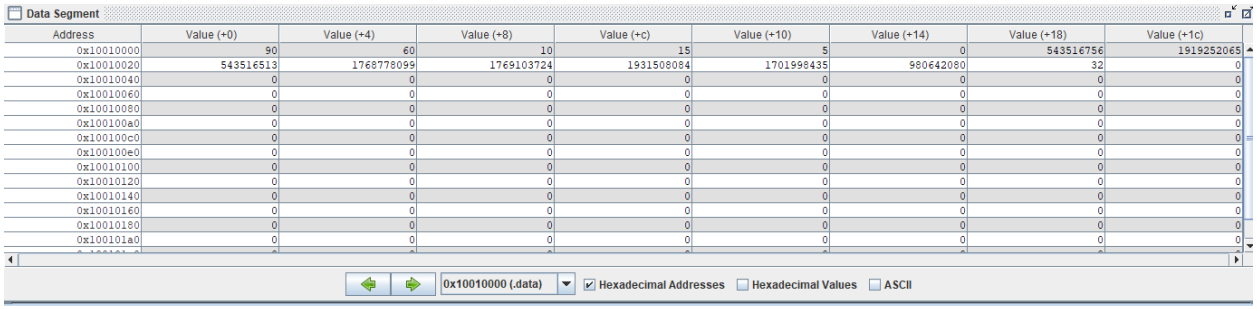

Figure 3: Data Segment after Testing

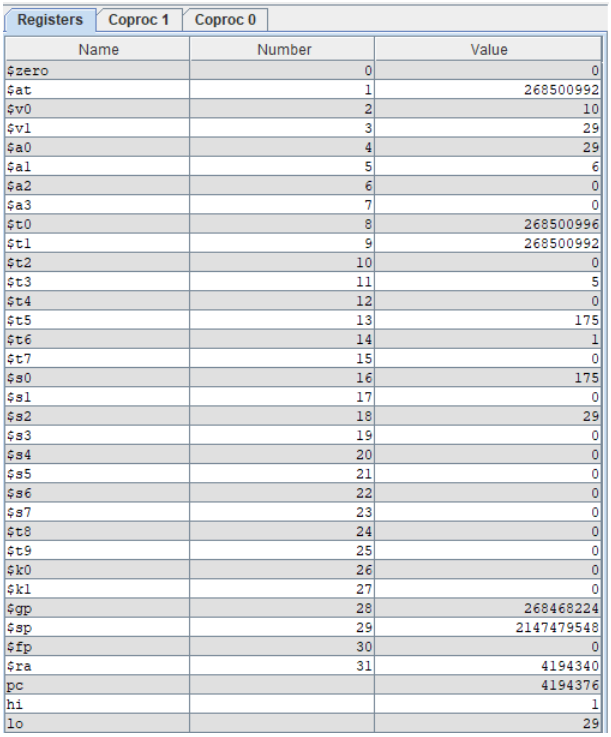

Figure 4: Register after Testing

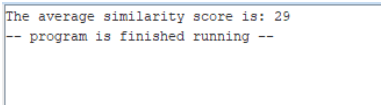

Figure 5: Output

- The first 6 values in the data segment represent the final state of each array after first function.
- In register, v1 and s2 is average, s1 is sum.

References

- https://slideplayer.com/slide/10869207/
- the exercises and examples in Digital Design and Computer Architecture