HACETTEPE UNIVERSITY

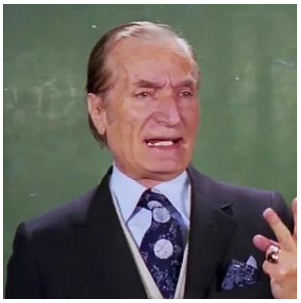
Department of Computer Engineering

BBM234 – MIPS Project

Date	Name	Number
15/04/2021	Murat Çelik	21827263

Hababam

Problem Definition



He is the last stronghold in defense of justice. He is the one who thinks teaching is the secret of the world. He is **Külyutmaz Necmi.**

He, who thinks that his students are cheating, wants to find a solution.

Hababam Sınıfı - the Chaos Class Tries to Cheat

For this, he creates a system using Mips technology. The teacher, who encrypts the notes, first wants to decrypt the notes. Even notes are divided by 8, single notes are multiplied by 5.

```
int datasize; // May vary with each different input
int data[datasize]; // Will be given for each run. Must be stored in memory
for(int i = 0; i < datasize; i++){
    if(data[i]%2 == 0)
        data[i] = data[i]/8; // It is forbidden to use div instructions
    else
        data[i] = data[i]*5; // It is forbidden to use mul/mult instructions
}
```

After that, it finds the average of the grades with its own system.

```
int average_recursive(int *data, int n){
    int sum, avg;
    if(n == 1)
        sum = data[0];
    else
        sum = data[n-1] + ((n-1) * average_recursive(data, n-1));
    avg = sum / n;
    return avg;
}
```

Külyutmaz thinks that students who get 80 and above are cheating and wants to find them.

We will help him.

MIPS Project

```
        .data
num_student :   .asciiz         "num_students = "
datasize :      .asciiz         "datasize     = "
# Test case 1  array = 720, 480, 80, 3, 1, 0                                    ----
# Test case 2  array = 0, 1, 5, 400, 112, 17, 7, 0, 560, 13, 0, 11, 3, 5, 0     ----
A:              .word           0, 1, 5, 400, 112, 17, 7, 0, 560, 13, 0, 11, 3, 5, 0
string:         .asciiz         "data[] = "
newline:        .asciiz         "\n"
result:         .asciiz         "average = "
bonus:          .asciiz         "Bonus: Cheaters IDs  "
space:          .asciiz         " "
and:            .asciiz         " and "
one :           .asciiz         "Inek Saban"
two:            .asciiz         "Güdük Necmi"
three:          .asciiz         "Damat Ferit"
four:           .asciiz         "Domdom Ali"
five:           .asciiz         "Tulum Hayri"
six :           .asciiz         "Hayta İsmail"

        .text
        .globl main
main:
```

This is the beginning part of my project. I have defined all the strings here. A statement is where I define array. In the lower part, I have defined the names of the students.

This part is the beginning of the main function. In chunks, this function continues. I get the number of students and data size from the command line here.

I have defined a counter for the for loop. I printed some words on the command line for the next process.

```
main:
        # this block takes the value num_student from the user
        li      $v0, 4
        la      $a0, num_student
        syscall
        li      $v0, 5
        syscall
        move $s0, $v0
        #-----------------------------------------------------
        # this block takes the value data size from the user
        li      $v0, 4
        la      $a0, datasize
        syscall
        li      $v0, 5
        syscall
        move $s1, $v0
        #-----------------------------------------------------

        la      $s2, A                  # $s2 =  array base address
        add     $t0, $0, $0             # $t0 = counter for "for loop"
        add     $t1, $0, $s2            # temporary base address
        # display string expression to terminal to make the answer readable
        li $v0, 4
        la $a0, newline
        syscall
        li $v0, 4
        la $a0, string
        syscall
        #-------------------------------------------------------------------
```

Part 1

```
# this is Part 1
for:
        slt     $t2, $t0, $s1           # for loops control statement => Less Than Comparision(use substraction)
        beq     $t2, $0, continue_main  # if counter - datasize is not equal 0, move on to the next instruction; otherwise go main
        lw      $s3, 0($t1)             # s3 holds array base value to navigate within array
        andi    $t3, $s3, 1             # holds "LSB value" of value for number is odd or even
        beq     $t3, $0, if_even        # if $t1 = 0, jump to if_even
        j       if_odd                  # else $t1 = 1, jump to if_odd
if_even:
        srl     $s3, $s3, 3             # shift to right for dividing 8
        j       inside_loop             # continue from loop body
if_odd:
        add     $t4, $s3, $0            # temporarily keep the initial of the value
        sll     $s3, $s3, 2             # shift to the left 2 times, so the value will be multiplied by 4
        add     $s3, $s3, $t4           # add temp valua and the value will be multiplied by 5
        j       inside_loop             # continue from loop body
inside_loop:
        sw      $s3, 0($t1)             # replace the changed value in array
        addi    $t0, $t0, 1             # counter += 1
        addi    $t1, $t1, 4             # switch the next value address of the array
        j               for             # continue for loop
#-------------------------------------------------------------------------------------------------
```

In this section I have deciphered the notes. Divide the even notes by 8 and multiply the odd notes by 5. While doing this, I did not use functions such as mul or div.

Test Case 1

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x10010000	1601009006	1685419123	1937010277	2112800	1635017060	1702521203
0x10010020	720	480	80	3	1	0

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x10010000	1601009006	1685419123	1937010277	2112800	1635017060	1702521203
0x10010020	90	60	10	15	5	0

Test Case 2

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	1601009006	1685419123	1937010277	2112800	1635017060	1702521203	538976288	2112800
0x10010020	0	1	5	400	112	17	7	0
0x10010040	560	13	0	11	3	5	0	1635017060

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	1601009006	1685419123	1937010277	2112800	1635017060	1702521203	538976288	2112800
0x10010020	0	5	25	50	14	85	35	0
0x10010040	70	65	0	55	15	25	0	1635017060

```
continue_main:
        sub     $t0, $t0, $t0           # empty register
        add     $t8, $0, $s2            # temporary base address of array
        j       loop1


# this part display all variable in array
loop1:
    bge     $t0, $s1, main_continue
    # load word from array and go next array
    lw      $t2, 0($t8)
    addi    $t8, $t8, 4
    # syscall to print value
    li      $v0, 1
    move    $a0, $t2
    syscall
    # spaces between numbers
    li      $a0, 32
    li      $v0, 11
    syscall
    #increment counter
    addi    $t0, $t0, 1
    j       loop1
#---------------------------------------
```

I printed the output on the screen by visiting the whole array

```
main_continue:
        # newline
        li $v0, 4
        la $a0, newline
        syscall
        #---------
        sub     $t8, $t8, $t8           # empty register
        add     $t1, $0, $s2            # temp base address ,again
        add     $t0, $0, $s1            # temp N
        addi    $t4, $t0, -1            # N-1
        sll     $t2, $t4, 2             # hold N-1 stack location value  **stack is used in this section **
        sub     $sp, $sp, $t2           # hold N-1 stack location       **Stack was used to remember where we came from **
        add     $a0, $0, $t1            #argument 1 = data
        add     $a1, $0, $t0            #argument 2 = N
        jal     AVG                     # function call
        add     $v1, $v0, $0            # save return(result) value to register
        j       main_continue_2
```

This is the preparatory part for Part 2. I updated some registers. I have defined the arguments. I used stack to use it in recursive function. I recorded them on the other side so as not to forget the places of return. I called the function. I saved the result at the end of the function in "$ v1".

Part 2

```
# this is Part 2
AVG:
        add     $s5, $0, $0             # $s5 = sum
        add     $s6, $0, $0             # $s6 = avg
        addi    $t3, $0, 1              # 1 for equality
        addi    $t4, $a1, -1            #N-1
        bne     $a1, $t3, else_AVG      #condition
        j       if_AVG
if_AVG:
        lw      $t5, 0($a0)
        add     $s5, $s5, $t5           # sum equal to first index of array
        j       inside_AVG              # continue loop
else_AVG:
        addi    $a1, $a1, -1            # recursive argument
        sw      $ra, 0($sp)             # remember called address
        addi    $sp, $sp, 4             # next stack address
        jal     AVG                     # recursion called
        addi    $sp, $sp, -4            # pevious stack address
        lw      $ra, 0($sp)             # remember called address
        add     $t8, $0, $v0            # take recursive return value
        addi    $t4, $t4, 1             # fix N-1
        mul     $t2, $t8, $t4           # temp value => recurs * (n-1)
        add     $s5, $s5, $t2           # add value to sum (complete else summing)
        add     $t6, $0, $a0            # temp data base
        sll     $t7, $t4, 2             # value for data[n-1] index
        add     $t6, $t6, $t7           # data[n-1]
        lw      $t9, 0($t6)             # value of data[n-1]
        add     $s5, $s5, $t9           # add data[n-1] to sum
        addi    $a1, $a1, 1
        j       inside_AVG
inside_AVG:

        div     $s6, $s5, $a1           # avg = sum / N
        sub     $s5, $s5, $s5           # sum = 0 for other function
        move    $v0, $s6                # save return value
        jr      $ra                     # go back

#-------------------------------------------------------------------------------------------------
```

I defined my recursive function according to the given system. With the help of the stack, I did not have any errors in "jr $ ra" instruction. If I hadn't used it, it would be overwritten and I wouldn't be able to access it.

Test 1

```
num_students = 4
datasize     = 6

data[] = 90 60 10 15 5 0
average = 29
Bonus: Cheaters IDs  1 Inek Saban and 2 Güdük Necmi
-- program is finished running --
```

$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	29

Test 2

```
num_students = 6
datasize     = 15

data[] = 0 5 25 50 14 85 35 0 70 65 0 55 15 25 0
average = 27
Bonus: Cheaters IDs  2 Güdük Necmi and 3 Damat Ferit
-- program is finished running --
```

$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	27

```
main_continue_2:
        # display result (final average)
        li $v0, 4
        la $a0, result
        syscall
        li $v0, 1
        move $a0, $v1
        syscall
        #--------------
```

In this section I printed the result on the screen.

Part 3 Bonus

```
#this is Part 3 ----- Bonus Part
        addi    $s6, $0, 1              # first cheater
        addi    $s7, $0, 2              # second cheater
        add     $t3, $0, $s2            # temp base address
        addi    $t4, $0, 0              # counter
        addi    $t6, $0, 79             # control score  (80 or higher)

for_loop_1:
        slt     $t5, $t4, $s1           # for loops control statement => Less Than Comparision(use substraction)
        beq     $t5, $0, done           # actually this is never execute; maybe if no one is cheater, it will go to finish point.
        lw      $t7, 0($t3)             # array value
        bge     $t7, $t6, print         # if array value > 79, go print else continue
        # this is helpful for find index of cheater
        addi    $t4,$t4,1
        addi    $t3, $t3, 4
        slt     $t5, $s7, $s0
        beq     $t5, $0, else
if:
        addi    $s7, $s7, 1
        j for_loop_1
else:
        addi    $s6, $s6, 1
        add     $s7, $s6, 1
        j for_loop_1
        #----------------------------------------
```

Using Array, I found the result above 79 and recorded that they have their indexes. I saved cheater index in $s6 and $s7.

```
print:
        li $v0, 4
        la $a0, newline # newline for readability
        syscall

        li $v0, 4
        la $a0, bonus   # bonus title
        syscall

        li $v0, 1
        move $a0, $s6   # index of first cheater
        syscall

        li $v0, 4
        la $a0, space   # space
        syscall

        add $a0, $0, $s6# for find name of first cheater
        jal equal

        li $v0, 4
        la $a0, and     # and
        syscall

        li $v0, 1
        move $a0, $s7   # index of second cheater
        syscall

        li $v0, 4
        la $a0, space   # space
        syscall

        add $a0 , $0, $s7
        jal equal       # for find name of second cheater

        j done  #finish task
# this part is helpful for find which index have which name
```

```
# this part is helpful for find which index have which name
equal:
        addi $t8, $0, 1
        beq $t8, $a0, print_inek
        addi $t8, $t8, 1
        beq $t8, $a0, print_necmi
        addi $t8, $t8, 1
        beq $t8, $a0, print_ferit
        addi $t8, $t8, 1
        beq $t8, $a0, print_ali
        addi $t8, $t8, 1
        beq $t8, $a0, print_hayri
        addi $t8, $t8, 1
        beq $t8, $a0, print_ismail
        jr $ra

print_inek:
        li $v0, 4
        la $a0, one
        syscall
        jr $ra
print_necmi:
        li $v0, 4
        la $a0, two
        syscall
        jr $ra
```

Test 1

$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	29
$a0	4	268501112
$a1	5	6
$a2	6	0
$a3	7	0
$t0	8	6
$t1	9	268501024
$t2	10	175
$t3	11	268501024
$t4	12	0
$t5	13	1
$t6	14	79
$t7	15	90
$s0	16	4
$s1	17	6
$s2	18	268501024
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	1
$s7	23	2
$t8	24	2
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479528
$fp	30	0
$ra	31	4194948
pc		4195132
hi		1
lo		29

Test 2

$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	27
$a0	4	268501160
$a1	5	15
$a2	6	0
$a3	7	0
$t0	8	15
$t1	9	268501024
$t2	10	406
$t3	11	268501044
$t4	12	5
$t5	13	1
$t6	14	79
$t7	15	85
$s0	16	6
$s1	17	15
$s2	18	268501024
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	2
$s7	23	3
$t8	24	3
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479492
$fp	30	0
$ra	31	4194948
pc		4195132
hi		1
lo		27

```
num_students = 4
datasize     = 6

data[] = 90 60 10 15 5 0
average = 29
Bonus: Cheaters IDs  1 Inek Saban and 2 Güdük Necmi
-- program is finished running --
```

```
num_students = 6
datasize     = 15

data[] = 0 5 25 50 14 85 35 0 70 65 0 55 15 25 0
average = 27
Bonus: Cheaters IDs  2 Güdük Necmi and 3 Damat Ferit
-- program is finished running --
```

Finish

```
done:
        li $v0, 10
        syscall
```

References

MIPS Tutorial - https://chortle.ccsu.edu/assemblytutorial/index.html

Amell Peralta - https://www.youtube.com/watch?v=B0GAXDjfdbQ

Quick Tutorial - https://minnie.tuhs.org/CompArch/Resources/mips_quick_tutorial.html

Stackoverflow - https://stackoverflow.com/questions/19748054/reading-and-printing-an-integer-in-mips