**CSE 338 - COMPUTER ORGANIZATION**

**PROJECT #1**

**REPORT**



Emre ATEŞ 150112059

Oğuz ÖZTÜRK 150112060

**Question 1**

In .data part,we define the asking message and error message.Also we define an array with the size of 100.

In .text part, we ask a number from user with the help of system calls.First we print the string that ask the user for enter a number.Then we read the number.

Then we check the errors.If an error will occur a message will shown on the console screen.After the error message,again the asking message appear.This will continue until user enters an acceptable input.

After that we send the number *to func1*.In *func1* first we put $ra in stack.Then we initalize the variables which are i=0,a1=0,s1=array address

Then we create two for loops.In the *loop1* we call *func2* with current i value.Now we are in *func2*.In *func2* first we put $ra in stack.Then we and input value with 1.

MIPS -> andi $t0,$a1,1 after this operation last bit will 0 if k%2=0

Then check if $t0 equals to zero.

If $t0 equals zero,means number is even,then operate this statement ,return (k \* k + 2) ,in mips.

Which in mips => mul $t2,$a1,$a1 # k\*k

addi $t2,$t2,2 # k\*k+2Then

addi $v0,$t2,0 # return (k\*k+2)

Then copy the $ra value in the stack and return the value to *loop1.*

If $t0 not equal zero,means number is odd,then go to *else* label and operate, return k \* (-1) , statement.In *else* label we find the negative value of input (which comes func2 from func1)

MIPS code to statment above => nor $t1,$a1,0

addi $t1,$t1,1

addi $v0,$t1,0

Then copy the $ra value in the stack and return the value to *loop1*.

Now we turn back to *loop1* (in *func1*) and we got return value from *func2*.Then we perform this operation arr[i] = func2(i).First we must multiply the i value with 4 because integers has 4 bytes in addressing.Then we add the i\*4 value to the current address of array.Then we are ready to do arr[i] = func2(i) operation.

MIPS => sll $t1,$s2,2 # $t1= i\*4

add $t1,$t1,$s1 # $t1=v+[i\*4]

sw $v0,0($t1) # arr[i]=func2(i)

This operations (go to *func2*,assign the return values in array) continue until *loop1* ends. (until i=k-1).After *loop1* ends again initalize variables. (i=0,s=0)

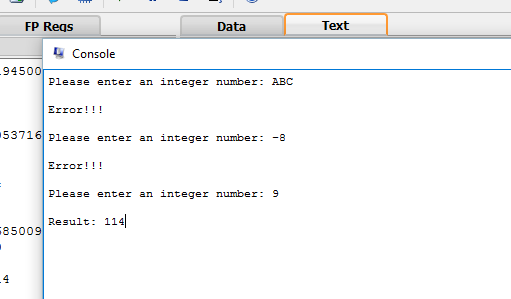
Then we go *loop2*.In *loop2* we calculate the sum of all array variables.

MIPS => lw $t4,0($t1) # $t4= arr[i]

add $s4,$s4,$t4 # s = s+arr[i]

This calculations continue until loop ends.Finally we return s variable to *main* and print it on console screen.

**Screenshot of program with errors and correct result:**

****

**Question 2**

In .data part,we define the asking message and error message.Also we define newline and white-space charachter.

In .text part, we ask a number from user with the help of system calls.First we print the string that ask the user for enter a positive number.Then we read the number .

Then we check the errors.If an error will occur a message will shown on the console screen.After the error message,again the asking message appear.This will continue until user enters an acceptable input.

Then,we design a nested for loop.First we initalize values.( i=0, j=1 , x=2)

i in $t1, j in $t3, x in $s0 , number in $t0

Our pseudo code:

for(i=0;i<number;i++){

for(j=1;j<x;j++){

print j;

}

x=x+1;

j=0;

j=y+1;

}

We transform this code to the MIPS code.

i<number;i++ -> addi $t1,$t1,1

slt $t2,$t1,$t0

bne $t2,$zero*,outer\_loop*

j<x;j++ -> addi $t3,$t3,1

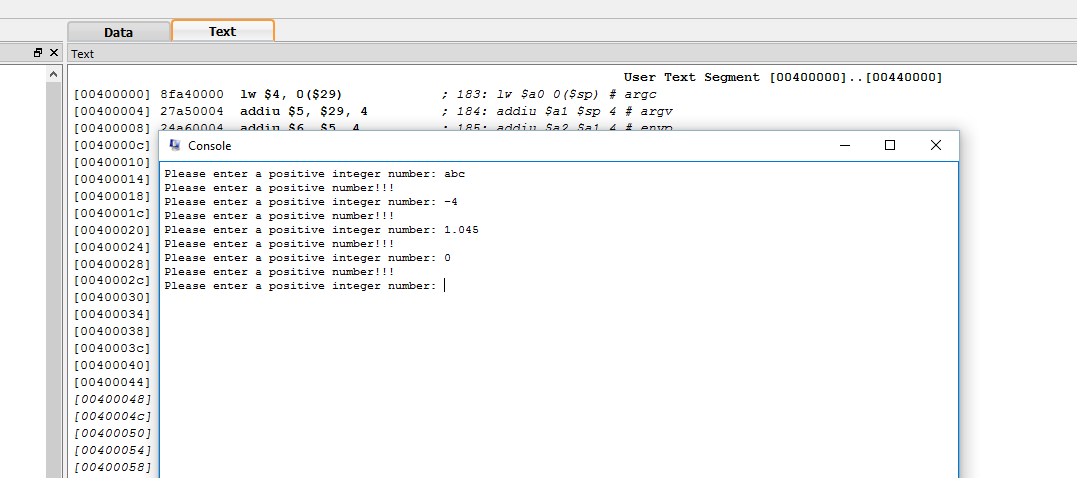
blt $t3,$s0,*inner\_loop*

x=x+1; -> addi $s0,$s0,1

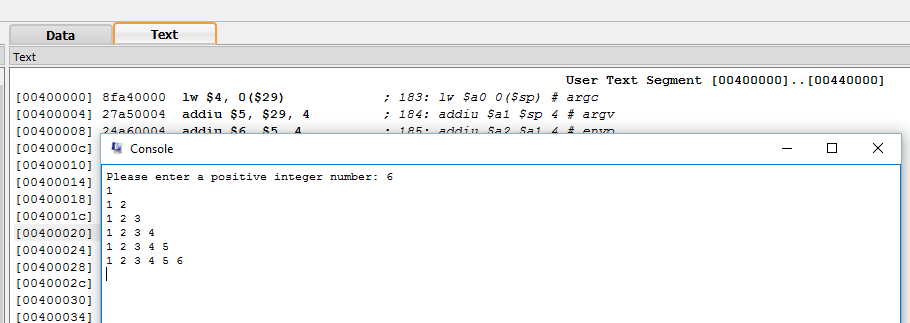
j=0; -> move $t3,$zero

j=y+1; -> addi $t3,$t3,1

**Screenshot for errors:**



**Screenshot for acceptable input:**



**Question 3**

In .data part,we define the strings and a string array called buffer.

In .text part, we ask user to enter a sentence and we load it in buffer.

We initalize buffer size to 250 and i=0.Then for checking we initalize special characters according to their ascii values.

MIPS => addi $t5,$t5,47 # character=,space()'.-

addi $t6,$t6,63 # character=?

addi $t7,$t7,95 # character=\_

->**Assume that character is not a special character**:

We create a *loop*.In the loop we find ascii value of first and second character and store in $a0 and $a1.After that we check first character is ? or not.

If it is equal to ? => jump label L 4

If it is not => jump label L2

Now we are in *L2*.In *L2* we check the first character is equal \_ (underscore) symbol or not.

If it is equal to \_ symbol => jump label L4

If it is not => jump label L3

Now we are in *L3*.In *L3* we check the first character is equal to remaining symbols (space,comma… etc.) or not.

If it equals => jump label L4

If it is not => jump label L1

Now we are in *L1*.In *L1*,we again check all conditions and if there is match with special characters jump the lable called *elsecon*.

If there is no match with special characters the continue with *printc* label.In there print the character and jump the *exit* label.

In *exit* label we increase i by 0 and check i < buffersize.If so loop will continue and turn back for check next character.

So if a character is not special character the program follows this way:

Loop -> L2 -> L3 -> L1 -> printc -> exit

->**Assume that character is one of the special character**:

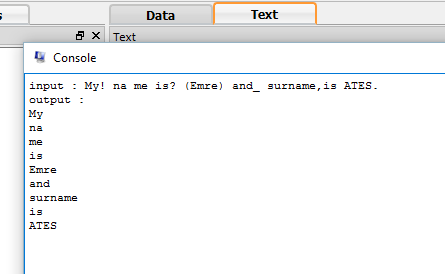
If there is match with a special character,program jump to label *L4*.

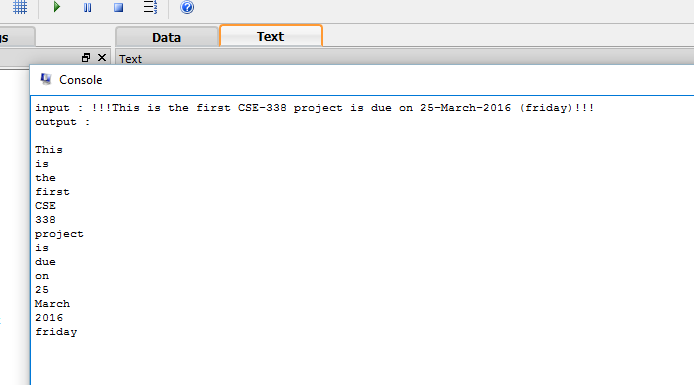
In *L4*,we check $a1,next character,because next character will be a normal character or a special character.

If $a1 is also a special character jump the *exit* label for searching next characters.

If $a1 is not a special character jump the *elsecon* label.In *elsecon* label program print newline character because the next characters should be a word.

**Screenshots:**





**Question 4**

In .data part,we define the asking message and error message.Also we define an array with the size of 100.

In .text part, we ask a number from user with the help of system calls.First we print the string that ask the user for enter a positive number.Then we read the number and we use the number as a array size. (addi $t2,$v0,0 #array size )

Then we check the errors.If an error will occur a message will shown on the console screen.After the error message,again the asking message appear.This will continue until user enters an acceptable input.

Then we initialize first and second element of array.First element is equal to 0 and second element is equal to 1.

MIPS =>

1-) la $s1,arr 2- ) move $t0 ,$zero 3-)sw $t0,0($s1) 4-) addi $t0,$t0,1 5-) sw $t0,4($s1)

Then we jump the *function1*.In the *function1* we initialize i=0 and we substract 2 from array size because we already initialize first two element of array.

Then we create a *loop*.In the *loop* ,first we multiply i value by 4 because integers hold 4 bytes.Then we add i value to array address value.After that we assign $t5 to current address value and $t6 to next element value.Then we multiply $t6 by 4 and add **-**$t5 value on it.We store the result in memory (2 next position on array).Then we increase i by 1 and check loop condition.

MIPS =>

sll $t4,$t1,2 # $t4=i\*4 integer hold 4 byte so we multiply by 4

add $t4,$t4,$s1 # $t4=v+[i\*4] find address of the element

lw $t5,0($t4) # assign to $t5 register value of in $t4 address location

lw $t6,4($t4) # assign to $t6 register value of next element

mul $t7,$t6,4 # $t6 register multiply by 4

nor $t5,$t5,$zero # !$t5

addi $t5,$t5,1 # take -$t5 value

add $t8,$t7,$t5 # get result of the operations(Result like this : A(x) = 4A(x-1) - A(x-2) )

sw $t8,8($t4) # store result to memory after 2 next element

addi $t1, $t1, 1 # i = i + 1

slt $t3, $t1, $t2 # i < size

bne $t3, $zero, loop

After *loop* ends we turn back to main.Then we go *printloop* and print all numbers in the array.At the same time in this way we just check correctness of our operations and results.

Lastly,we print the number in requested position.

**Screenshots:**

