150117509 - Mustafa Abdullah Hakkoz
150115010 – Mahmut Aktaş

03.05.2020

COMPUTER ORGANIZATION PROJECT 1 REPORT

Question 1

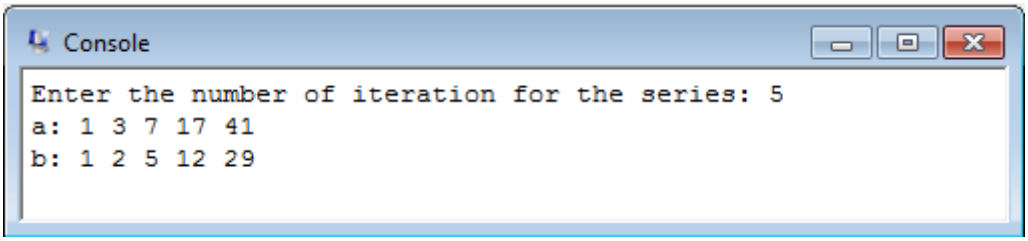
In this program we implement a number series for square root of 2 approximation based on given inputs. In order to achieve this goal firstly we need to use system call 4 to print a prompt string for user. After that we can read the input integer by using system call 5.

```
#prompt string
li $v0, 4
la $a0, prompt
syscall

#read the integer
li $v0, 5
syscall
move $t0, $v0
```

Then we created four integer to serve each specific goal. “temp” integer holds the b values. Therefore when b value changed, we can still the its previous value. After that we initialize our first loop to calculate a values. In this loop we calculated the values from the formula and print the a value. Then we initialize our second loop which is very similar to the first one. In this loop we calculate the b values with the same calculations and print it. By using two loops, we don’t have to convert integers to string in order to concatenate them.

```
    li $t1, 0 #t1 is temp
    li $t2, 1 #t2 is a
    li $t3, 1 #t3 is b
    li $t4, 0 #temp for counter

    add $t4, $t4, $t0

    li $v0, 4 # print "a: "
    la $a0, a_str
    syscall

while: blez $t0, endwhile

    move $a0, $t2
    li $v0, 1 # print "a" value
    syscall

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

    add $t1, $t3, $zero # temp = b

    add $t3, $t3, $t2 # b = a + b

    add $t1, $t1, $t1 #temp *= 2

    add $t2, $t1, $t2 #a = a + 2temp

    sub $t0, $t0, 1
    j while

endwhile:

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

Question 1 – output:



Question 2

As a start, we initiate variables, space alignments, printing strings to screen and reading matrices, then we convert them to integers and stored in arrays with **loop_string1**. Same process is repeted for second matrix in **loop_string2.**

After the both of loops, second dimension of the second matrix is calculated with **div** instruction.

After constructing arrays, we did matrix multiplication with three loops: **loop_i, loop_**j and **loop_k**. First loop iterates over rows of matrix1 **(i<a),** second loop iterates over columns of matrix2 **(j<c),** and third loop iterates over **b** which represents columns of matrix1 and rows of matrix2 at the same time **(k<b)**.

```
    ####################

    la $s4, matrix1                     #s4 holds matrix1 string
    la $s5, matrix2                     #s5 holds matrix2 string
    la $s6, spaceChar
    lb $s6, 0($s6)                      #s6 holds spaceChar

    #construct array1 and calculate matrix1 length
    li $t2, 0       #previous digit
    li $s3, 10      #$s3 represents 10 for digit calculations
    li $t4, 0       #array1 index
    li $t5, 0       #array1 length
    la $t7, newLine
    lb $t7, 0($t7)

loop_string1:
    lb $t3, 0($s4)
    beq $t3, $s6, space_condition1
    beq  $t3, $t7, end_string1
    sub $t3, $t3, 48
    mul $t2, $t2, $s3
    add $t2, $t2, $t3   #number= previous number * 10 + current digit
    addi $s4, $s4, 1
    j loop_string1
space_condition1:
    sw $t2, Array1($t4)
    addi $t4, $t4, 4
    addi $t5, $t5, 1                    #t5 holds length of array1
    li $t2, 0
    addi $s4, $s4, 1
    j loop_string1
end_string1:
    sw $t2, Array1($t4)
    addi $t5, $t5, 1
```

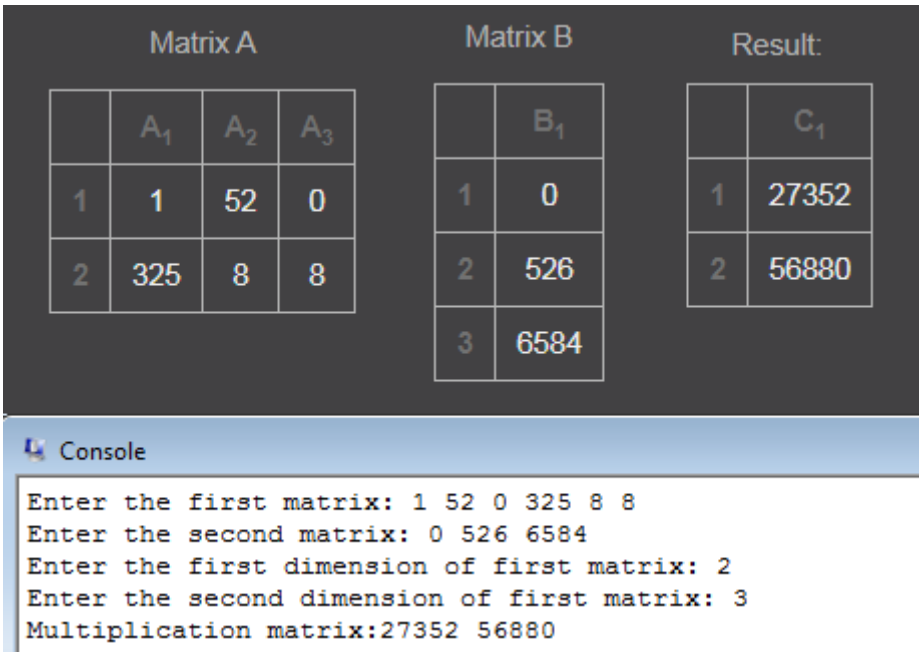
In **loop_k**, the most inner loop, we calculate indexes, fetch values from arrays by using them and multiply both of fetched values. In ending section of **loop_k** we summed all iterations of k. Basically, we did the calculation **sum += M[i,k] * N[k,j]** in this loop. In ending of **loop_k** (**loop_k_end**) we stored the **sum** in **Result** array (**R[i,j]**).

In **loop_j**, the intermediate loop, we initialize sum variable (**sum=0**) for calculations in **loop_k**.

And finally loop_i is most outer loop and it's only consist **loop_j.**

After matrix calculation steps, we read **Result** array and print it to screen.

Question 2 - output:



Question 3

```
    #find string length
    la $t0, buffer
loop:
    lb   $t1 0($t0)
    beq  $t1 $zero end

    addi $t0, $t0, 1
    j loop
end:
```

In this question we implement a program that checks if the given string is palindrome or not, case insensitive. We prompt the user the way we did in other two questions. After that we initialize a loop to find the given string's length. In order to achieve this goal, we iterate the given string's char by one until we hit the "zero" element at the end of the string.

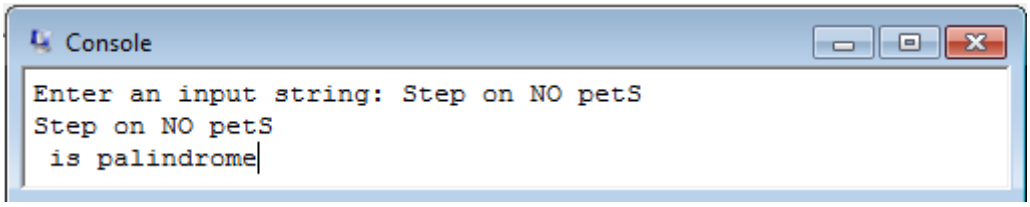
After finding the length of the string, now we can iterate our given string from start and the end points. Firstly we create a set of variables.

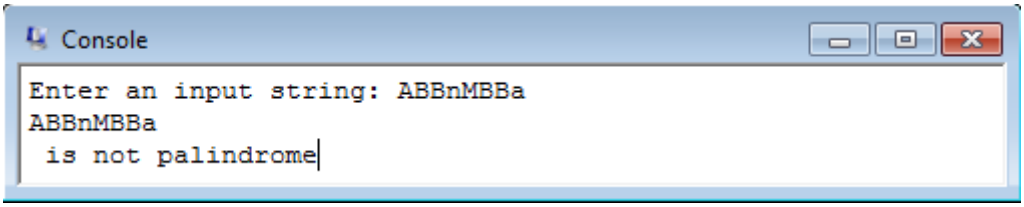
```
la $t1, buffer
sub $t3, $t0, $t1 #$t3 now contains the length of the string
sub $t3, $t3, 2

li $t2, 0 #lower bound counter

li $t4, 0 #lower bound char
li $t5, 0 #upper bound char

li $t6, 0 #lower bound char ascii
li $t7, 0 #upper bound char ascii
```

In this loop firstly we take the letters from the start. After getting a letter we checked if it is lower case or not by comparing it with 91. Lowercase letters start from 91 in the ASCII Table. If the current letter is not lowercase, we convert it to the lowercase by simply add it with 32. After conversion we do the same operation for letters from the end point. After having two ASCII value for the two letters, we compare them with each other. If they are not equal, then the input string is not palindrome. If the loop continues until both counter reach each other, then the given string is palindrome.

```
Loop:

    bgt $t2, $t3, endwhile


    add $t4, $t1, $t2 #checking letter from start
    lb $a0, 0($t4)
    li $t6, 0
    add $t6, $t6, $a0


    ble $t6, 91, lowercase1

continue1:
    add $t5, $t1, $t3 #checking letter from end
    lb $a1, 0($t5)
    li $t7, 0
    add $t7, $t7, $a1

    ble $t7, 91, lowercase2

continue2:
    bne $t6, $t7, not_palindrome

    add $t2, $t2, 1
    sub $t3, $t3, 1
    j Loop
```

Question 3 – Outputs:





Menu

In menu we have a main loop for printing items and reading selection. In the loop we call the code section for relevant function depending on selection number. All the codes for questions are included in **Menu.asm** file, in their seperate sections.

Menu – output:

```
Console
Welcome to our MIPS project!
Main Menu:
1. Square Root Approximate
2. Matrix Multiplication
3. Palindrome
4. Exit
Please select an option: 1
Enter the number of iteration for the series: 5
a: 1 3 7 17 41
b: 1 2 5 12 29

Welcome to our MIPS project!
Main Menu:
1. Square Root Approximate
2. Matrix Multiplication
3. Palindrome
4. Exit
Please select an option: 2
Enter the first matrix: 3 7 8 12 5 6 7 2 4 3 2 5
Enter the second matrix: 2 5 6 3 7 8 9 1
Enter the first dimension of first matrix: 3
Enter the second dimension of first matrix: 4
Multiplication matrix:212 112 113 101 85 50

Welcome to our MIPS project!
Main Menu:
1. Square Root Approximate
2. Matrix Multiplication
3. Palindrome
4. Exit
Please select an option: 3
Enter an input string: abGcdEEedcgba
abGcdEEedcgba
 is palindrome

Welcome to our MIPS project!
```