## Part 1:

Make a fibonacci series program in MIPS assembly, take user input for the number of terms to be added in fibonnaci series. Make sure to validate user input that it is greater than 1, prompt problem and ask again otherwise, or print series and sum of all terms if it is all right. Good prompts for user and proper comments in code are mandatory for this program.

Fibonnaci Series is calculated by



An Example of Series is:  


The example input / output prompts and validation for program is demonstrated below:

|  |
| --- |
| Enter an Integer for Number to terms to be added in Fibonacci Series: 0  Kindly enter an integer greater than 1  Enter an Integer for Number to terms to be added in Fibonacci Series: -123  Kindly enter an integer greater than 1  Enter an Integer for Number to terms to be added in Fibonacci Series: 1  Kindly enter an integer greater than 1  Enter an Integer for Number to terms to be added in Fibonacci Series: 2  Fibonacci Series: 0, 1  The sum of Fibonacci sequence is: 1  Enter an Integer for Number to terms to be added in Fibonacci Series: 13  Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144  The sum of Fibonacci sequence is: 376 |

|  |
| --- |
| Solution:  #Homework 5-1 Michael Bros  .data  prompt1: .asciiz"Please enter an Integer for terms to be added to Fibonacci Series(-1 to exit): "  msg1: .asciiz "\nFibonacci Series: "  msg2: .asciiz "\nThe sum of the Fibonacci sequence is: "  error: .asciiz "Please enter an integer greater than 1\n"  comma: .asciiz ", "  space: .asciiz " "  newLine: .asciiz "\n"  .text  main:  li $s1, 0 #term 1  li $s2, 1 #term 2  li $s3, 0 #next term  li $s4, 0 #sum  li $s5, 3 #init i    li $v0, 4 #ask user to input number  la $a0, prompt1  syscall    li $v0, 5 #get number and store in $s0  syscall  move $s0,$v0    li $t0, -1  beq $t0, $s0, Exit    li $t0,1  beq $t0, $s0, Error  slt $t1, $s0, $t0 #if number < 1  bnez $t1, Error #number < 1 branch to numCheck    li $v0, 4 #print msg1  la $a0, msg1  syscall    li $v0, 1 #print term 1  addi $a0, $s1, 0  syscall  add $s4, $s4, $s1 #add to sum    li $v0, 4 #print comma  la $a0, comma  syscall    li $v0, 1 #print term 2  addi $a0, $s2, 0  syscall  add $s4, $s4, $s2 #add to sum    j For    For:  li $t0, 2  beq $t0, $s0, Sum    add $s3, $s1, $s2  move $s1, $s2  move $s2, $s3    li $v0, 4 #print comma  la $a0, comma  syscall    li $v0, 1 #print next term  addi $a0, $s3, 0  syscall  add $s4, $s4, $s3 #add to sum    addi $s5, $s5, 1 #i++  beq $s5, $s0, For # i == n  slt $t1, $s5, $s0 # i < n  bnez $t1, For    j Sum  Sum:  #print sum  li $v0, 4 #print msg2  la $a0, msg2  syscall    li $v0, 1 #print sum  addi $a0, $s4, 0  syscall    li $v0, 4 #print newLine  la $a0, newLine  syscall  syscall  j main  Error:  li $v0, 4 #print error  la $a0, error  syscall    j main  Exit:  li $v0, 10  syscall  Screenshot of program running:  A screenshot of a social media post  Description automatically generated |

## Part 2:

The following MIPS program was written by ~~Paul.~~ ~~Not Paul~~. Bob. It was supposed to take an integer x from the user and then return x^2. Unfortunately they were ~~too lazy~~. ~~too important~~ too busy to finish so they've asked you to finish it for them. Finish their code by writing the function call for their getSq function. (Note the function only uses mflo so you need to find a way to re-prompt the user for input if they enter a number greater than 46340). You will also need to find a way to output the results of the function to the user.

|  |
| --- |
| .data  inputmsg: .asciiz "Please enter a number less than 46341\n"  output: .asciiz "Square of that is: "  .text  main:  li $t0 46341  getIn:#takes input from the user  li $v0 4  la $a0 inputmsg  syscall    li $v0 5  syscall  #if greater than 46341 branch back to getln  Slt $t1, $a0, $t0  Beqz $t1, getln  Jal getSq  #output message  li $v0 4  la $a0 output  syscall  #print output value  addi $a0, $v0, 0  li $v0 5  syscall  li $v0 10 #exit the program  syscall  getSq:#totally flawless function! Make sure you use it! Also don’t edit it  mult $a0 $a0  mflo $v0  jr $ra |