# this is a full working factorial version that I found on the internet and is complete

#reference- https://gist.github.com/Peng-YM/be70d28079833bd701b05a5ce7772ff1

.data

prompt: .asciiz "Input an integer:\n" #storing an output message to be called

result: .asciiz "Fact(x) = " #storing an output message to be called

.text

# show prompt

li $v0, 4 #syscall 4

la $a0, prompt #load prompt into $a0

syscall #system call,

# read x

li $v0, 5 #syscall 5

syscall #system call

# function call

move $a0, $v0 # $a0 = $v0, $a0 will be the input

jal factorial # jump factorial and save position to $ra

move $t0, $v0 # $t0 = $v0

# show prompt

li $v0, 4 #syscall 4

la $a0, result #save result prompt into $a0

syscall #system call

# print the result

li $v0, 1 # system call #1 - print int

move $a0, $t0 # $a0 = $t0

syscall # system call

# return 0

li $v0, 10 # $v0 = 10

syscall #system call

.text

factorial:

# base case -- still in parent's stack segment

# adjust stack pointer to store return address and argument

addi $sp, $sp, -8

# save $s0 and $ra

sw $s0, 4($sp) #save $s0 in stack

sw $ra, 0($sp) #save $ra in stack

bne $a0, 0, else #perform else, if $a0 is not zero

addi $v0, $zero, 1 # return 1

j fact\_return #jump to fact\_return

else:

# backup $a0

move $s0, $a0 #$s0 = $a0

addi $a0, $a0, -1 # x -= 1

jal factorial #call factorial until condition does not hold

# when we get here, we already have Fact(x-1) store in $v0

multu $s0, $v0 # return x\*Fact(x-1), this will execute on final

mflo $v0 #store into LO

fact\_return:

lw $s0, 4($sp) #restore $s0

lw $ra, 0($sp) #restore $ra

addi $sp, $sp, 8 #adjust the stack

jr $ra #return to main

#this is just the function for factorial that I made

fact:

addi $sp, $sp, -8

sw $ra, 4($sp)

sw $a0, 0($sp)

slti $t0, $a0, 1

beq $t0, $zero, else

addi $v0, $zero, 1

addi $sp, $sp, 8

jr $ra

else:

addi $a0, $a0, -1

jal fact

lw $a0, 0($sp)

lw $ra, 4($sp)

addi $sp, $sp, 8

mul $v0, $a0, $v0

jr $ra

# this is a full working Fibonacci version that I found on the internet and is complete

#reference - https://gist.github.com/libertylocked/068b118354539a8be992

.data

prompt1: .asciiz "Enter the sequence index\n" #storing an output message to be called

prompt2: .asciiz "The Fibonacci value is:\n" #storing an output message to be called

message: .asciiz "The Fibonacci value is:\n0" #storing an output message to be called

.text

# Print prompt1

li $v0, 4 #syscall 4

la $a0, prompt1 #store promt into $a0

syscall #system call

# Read integer

li $v0, 5 #syscall 5

syscall #system call

beq $v0, 0, equalToZero #branch if fib is zero to avoid excessive calulation, fib will be 0

# Call fibonacci

move $a0, $v0 # $a0 = $v0,

jal fibonacci #jump to fibinochi function

move $a1, $v0 # save return value to a1

# Print prompt2

li $v0, 4 #syscall 4

la $a0, prompt2 #store prompt2 into $a0

syscall #system call

# Print result

li $v0, 1 #syscall 1- read int

move $a0, $a1 #$a0 = $a1

syscall #system call

# Exit/ return 0

li $v0, 10

syscall

## Function int fibonacci (int n)

fibonacci:

# Prologue

addi $sp, $sp, -12 #adjust stack

sw $ra, 8($sp) #save $ra in stack

sw $s0, 4($sp) #save $s0 in stack

sw $s1, 0($sp) #save $s1 in stack

move $s0, $a0 #$s0 = $a0

li $v0, 1 # return value for terminal condition

ble $s0, 0x2, fibonacciExit # check terminal condition, jump to fibonacciexit

addi $a0, $s0, -1 # set args for recursive call to f(n-1)

jal fibonacci #restart fibinacci

move $s1, $v0 # store result of f(n-1) to s1

addi $a0, $s0, -2 # set args for recursive call to f(n-2)

jal fibonacci #restart fibinochi

add $v0, $s1, $v0 # add result of f(n-1) to it

fibonacciExit:

# Epilogue

lw $ra, 8($sp) #restore $ra

lw $s0, 4($sp) #restore $s0

lw $s1, 0($sp) #restore $s1

addi $sp, $sp, 12 #adjust stack

jr $ra #jump to return address

## End of function fibonacci

equalToZero:

li $v0, 4 #syscall 4

la $a0, message #load message to $a0

syscall #system call

#this is just the function for fibinchi I made

fibinochi:

addi $sp, $sp, -12

sw $ra, 8($sp)

sw $s0, 4($sp)

sw $s1, 0($sp)

addi $s0, $a0, 0

ble $s0, 1 ,Exit

addi $a0, $s0, -1

jal fibinochi

addi $s1, $v0, 0

addi $a0, $s0, -2

jal fibonacci

add $v0, $s1, $v0

Exit:

lw $ra, 8($sp)

lw $s0, 4($sp)

lw $s1, 0($sp)

addi $sp, $sp, 12

jr $ra