**CALCULATOR**

.data

cal: .asciiz "-----CALCULATOR-----\n"

prompt: .asciiz "\n-> Choose an Operation to Perform:\n"

opt: .asciiz "\n1. Addition\n2. Subtraction\n3. Multiplication\n4. Division\n5. Modulus\n6. Square\n7. Power\n8. Factorial\n9. Summation (Sigma)\n10. Boolean Operations\n11. Exit\n\nEnter Your Option: "

error: .asciiz "\nInvalid Option Entered.....Try Again\n"

line: .asciiz "\n"

add1: .asciiz "\n-----Addition-----\n"

sub1: .asciiz "\n-----Subtraction-----\n"

subop: .asciiz "\nEnter the Number of Operands you want to Subtract from A: "

divop: .asciiz "\nEnter the Number of Operands you want to Divide by A: "

mul1: .asciiz "\n-----Multiplication-----\n"

div1: .asciiz "\n-----Division-----\n"

mod1: .asciiz "\n-----Modulus-----\n"

sq: .asciiz "\n-----Square-----\n"

pow: .asciiz "\n-----Power-----\n"

fac: .asciiz "\n-----Factorial-----\n"

sum: .asciiz "\n-----Summation (Sigma)----\n"

error1: .asciiz "\nInvalid Number Eneterd.....Enter a Positive Number\n"

error2: .asciiz "\nCan't Divide by Zero.....Try Again\n"

error3: .asciiz "\nInvalid Input.....Input must be B > A.....Try again.\n"

entera: .asciiz "\nEnter A: "

enterb: .asciiz "Enter B: "

entern: .asciiz "\nEnter N: "

enterp: .asciiz "Enter Power: "

op: .asciiz "\nEnter the Number of Operands: "

enterop: .asciiz "Enter Operand: "

result: .asciiz "\nResult: "

bool: .asciiz "\n-----Boolean Operations-----\n\n1. AND\n2. OR\n3. XOR\n4. NOR\n5. Exit\n\nChoose an Option: "

and1: .asciiz "\n-----AND-----\n"

or1: .asciiz "\n-----OR-----\n"

xor1: .asciiz "\n-----XOR-----\n"

nor1: .asciiz "\n-----NOR-----\n"

.text

.globl main

main:

li $v0, 4

la $a0, cal

syscall

########## Code For Displaying Options ##########

option:

li $v0, 4

la $a0, prompt

syscall

li $v0, 4

la $a0, opt

syscall

li $v0, 5

syscall

move $t0, $v0

beq $t0, 1, addition

beq $t0, 2, subtraction

beq $t0, 3, multiplication

beq $t0, 4, division

beq $t0, 5, mod

beq $t0, 6, square

beq $t0, 7, power

beq $t0, 8, factorial

beq $t0, 9, sigma

beq $t0, 10, boolOp

beq $t0, 11, exit

li $v0, 4

la $a0, error

syscall

j option

########## Code For Addition ##########

addition:

li $v0, 4

la $a0, add1

syscall

li $v0, 4

la $a0, op

syscall

li $v0, 5

syscall

move $t0, $v0

li $t2 , 0

li $v0, 4

la $a0, line

syscall

addLoop:

li $v0, 4

la $a0, enterop

syscall

li $v0, 5

syscall

move $t1, $v0

add $t2, $t2, $t1

addi $t0, $t0, -1

bgt $t0, 0, addLoop

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t2

syscall

j option

########## Code For Subtraction ##########

subtraction:

li $v0, 4

la $a0, sub1

syscall

li $v0, 4

la $a0, entera

syscall

li $v0, 5

syscall

move $t2, $v0

li $v0, 4

la $a0, subop

syscall

li $v0, 5

syscall

move $t0, $v0

li $v0, 4

la $a0, line

syscall

subLoop:

li $v0, 4

la $a0, enterop

syscall

li $v0, 5

syscall

move $t1, $v0

sub $t2, $t2, $t1

addi $t0, $t0, -1

bgt $t0, 0, subLoop

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t2

syscall

j option

########## Code For Multiplication ##########

multiplication:

li $v0, 4

la $a0, mul1

syscall

li $v0, 4

la $a0, op

syscall

li $v0, 5

syscall

move $t0, $v0

li $t2 , 1

li $v0, 4

la $a0, line

syscall

mulLoop:

li $v0, 4

la $a0, enterop

syscall

li $v0, 5

syscall

move $t1, $v0

mult $t2, $t1

mflo $t2

addi $t0, $t0, -1

bgt $t0, 0, mulLoop

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t2

syscall

j option

########## Code For Division ##########

division:

li $v0, 4

la $a0, div1

syscall

multiDiv:

li $v0, 4

la $a0, entera

syscall

li $v0, 5

syscall

move $t2, $v0

li $v0, 4

la $a0, divop

syscall

li $v0, 5

syscall

move $t0, $v0

li $v0, 4

la $a0, line

syscall

divLoop:

li $v0, 4

la $a0, enterop

syscall

li $v0, 5

syscall

move $t1, $v0

beq $t1, 0, divZero

div $t2, $t1

mflo $t2

addi $t0, $t0, -1

bgt $t0, 0, divLoop

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t2

syscall

j option

divZero:

li $v0, 4

la $a0, error2

syscall

j multiDiv

########## Code For Modulus ##########

mod:

li $v0, 4

la $a0, mod1

syscall

li $v0, 4

la $a0, entera

syscall

li $v0, 5

syscall

move $t0, $v0

li $v0, 4

la $a0, enterb

syscall

li $v0, 5

syscall

move $t1, $v0

div $t0, $t1

mfhi $t2

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t2

syscall

j option

########## Code For Square ##########

square:

li $v0, 4

la $a0, sq

syscall

li $v0, 4

la $a0, entera

syscall

li $v0, 5

syscall

move $s0, $v0

mult $s0, $s0

mflo $s2

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $s2

syscall

j option

########## Code For Power ##########

power:

li $v0, 4

la $a0, pow

syscall

li $v0, 4

la $a0, entera

syscall

li $v0, 5

syscall

move $t0, $v0

li $v0, 4

la $a0, enterp

syscall

li $v0, 5

syscall

move $t1, $v0

li $t2, 1

powerLoop:

mult $t2, $t0

mflo $t2

addi $t1, $t1, -1

bgt $t1, 0, powerLoop

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t2

syscall

j option

########## Code For Factorial ##########

factorial:

li $v0, 4

la $a0, fac

syscall

factorialLoop:

li $v0, 4

la $a0, entern

syscall

li $v0, 5

syscall

move $t0, $v0

blt $t0, 0, invalid

blt $t0, 2, factorial01

li $t2, 1

calculateFactorial:

mult $t2, $t0

mflo $t2

addi $t0, $t0, -1

bgt $t0, 0, calculateFactorial

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t2

syscall

j option

invalid:

li $v0, 4

la $a0, error1

syscall

j factorialLoop

factorial01:

li $v0, 4

la $a0, result

syscall

li $v0, 1

li $a0, 1

syscall

j option

########## Code For Summation ##########

sigma:

li $v0, 4

la $a0, sum

syscall

sigmaLoop:

li $v0, 4

la $a0, entera

syscall

li $v0, 5

syscall

move $t0, $v0

li $v0, 4

la $a0, enterb

syscall

li $v0, 5

syscall

move $t1, $v0

ble $t1, $t0, invalidSum

li $t2, 0

sumLoop:

add $t2, $t2 , $t0

addi $t0, $t0, 1

ble $t0, $t1, sumLoop

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t2

syscall

j option

invalidSum:

li $v0, 4

la $a0, error3

syscall

j sigmaLoop

########## Code For Boolean Operations ##########

boolOp:

li $v0, 4

la $a0, bool

syscall

li $v0, 5

syscall

move $t0, $v0

beq $t0, 1, case1

beq $t0, 2, case2

beq $t0, 3, case3

beq $t0, 4, case4

beq $t0, 5, case5

li $v0, 4

la $a0, error

syscall

j boolOp

########## Code For AND Operation ##########

case1:

li $v0, 4

la $a0, and1

syscall

li $v0, 4

la $a0, entera

syscall

li $v0, 5

syscall

move $t0, $v0

li $v0, 4

la $a0, enterb

syscall

li $v0, 5

syscall

move $t1, $v0

and $t2, $t0, $t1

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t2

syscall

li $v0, 4

la $a0, line

syscall

j boolOp

########## Code For OR Operation ##########

case2:

li $v0, 4

la $a0, or1

syscall

li $v0, 4

la $a0, entera

syscall

li $v0, 5

syscall

move $t0, $v0

li $v0, 4

la $a0, enterb

syscall

li $v0, 5

syscall

move $t1, $v0

or $t2, $t0, $t1

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t2

syscall

li $v0, 4

la $a0, line

syscall

j boolOp

########## Code For XOR Operation ##########

case3:

li $v0, 4

la $a0, xor1

syscall

li $v0, 4

la $a0, entera

syscall

li $v0, 5

syscall

move $t0, $v0

li $v0, 4

la $a0, enterb

syscall

li $v0, 5

syscall

move $t1, $v0

xor $t2, $t0, $t1

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t2

syscall

li $v0, 4

la $a0, line

syscall

j boolOp

########## Code For NOR Operation ##########

case4:

li $v0, 4

la $a0, nor1

syscall

li $v0, 4

la $a0, entera

syscall

li $v0, 5

syscall

move $t0, $v0

li $v0, 4

la $a0, enterb

syscall

li $v0, 5

syscall

move $t1, $v0

nor $t2, $t0, $t1

li $v0, 4

la $a0, result

syscall

li $v0, 1

move $a0, $t2

syscall

li $v0, 4

la $a0, line

syscall

j boolOp

########## Code to Exit Boolean Operations ##########

case5:

j option

########## Code to Exit the Program ##########

exit:

li $v0, 10

syscall