**Matricula: 20180999 Nombre: Junior Hernández**

Calculadora

; Macros para la lectura e impresion

%macro Stprint 2 ; Se necesitan de 2 parametros

mov eax, 4 ; Aqui se escribe

mov ebx, 1 ;

mov ecx, %1 ; Aqui va la cadena

mov edx, %2 ; Aqui va la longitud de la cadena

int 0x80 ; Imprimir

%endmacro

; Ejemplo de uso de la macro con el salto de linea

%macro saltodelinea 0

Stprint salto, saltolen

%endmacro

global main

; Utilizar printf y scanf de la libreria de C para varias acciones que hacer.

extern printf, scanf

section .data

; Para tomar/capturar los numeros

scan\_format: db "%f",0

print\_formatA: db "Numero A: %d",0xA,0

print\_formatB: db "Numero B: %d",0xA,0

; Para mostrar/imprimir los numeros.

print\_formatA\_hex: db "A-hex: %x",0xA,0

print\_formatA\_oct: db "A-oct: %o",0xA,0

print\_formatB\_hex: db "B-hex: %x",0xA,0

print\_formatB\_oct: db "B-oct: %o",0xA,0

; Para imprimir resultados

; Con punto/coma flotante

print\_suma: db "Operacion de suma: %f", 0xA, 0

print\_resta: db "Operacion de resta: %f", 0xA, 0

print\_multiplicacion: db "Operacion de multiplicacion: %f", 0xA, 0

print\_division: db "Operacion de division: %f", 0xA, 0

; En Hexadecimal

print\_suma\_hex: db "Operacion de suma: %x", 0xA, 0

print\_resta\_hex: db "Operacion de resta: %x", 0xA, 0

print\_multiplicacion\_hex: db "Operacion multiplicacion: %x", 0xA, 0

print\_division\_hex: db "Operacion de division: %x", 0xA, 0

; En Octal

print\_suma\_oct: db "Operacion de suma: %o", 0xA, 0

print\_resta\_oct: db "Operacion de resta: %o", 0xA, 0

print\_multiplicacion\_oct: db "Operacion de multiplicacion: %o", 0xA, 0

print\_division\_oct: db "Operacion de division: %o", 0xA, 0

; Mensajes de salida y entrada.

ingresar\_a db "Ingrese Primer Numero(A): ", 0

ingresar\_a\_len equ $ - ingresar\_a

ingresar\_b db "Ingrese Segundo Numero(B): ", 0

ingresar\_b\_len equ $ - ingresar\_b

error db "El segundo numero no puede ser igual a 0 para dividir.", 0xA, 0

error\_len equ $ - error

msg1 db "Operaciones con flotantes: ", 0xA, 0

msg1\_len equ $ - msg1

msg2 db "Resultados: ", 0xA, 0

msg2\_len equ $ - msg2

msgdec db "Numeros decimales: ", 0xA, 0

msgdec\_len equ $ - msgdec

msghex db "Operaciones con hexadecimales: ", 0xA, 0

msghex\_len equ $ - msghex

msgoct db "Operaciones con octales: ", 0xA, 0

msgoct\_len equ $ - msgoct

; Para hacer un salto de linea

salto db 0xA, 0xD

saltolen equ $ - salto

section .bss

; Para reservar memoria con variables flotantes.

result\_num\_a: resb 4 ; Se guarda el numero A

result\_num\_b: resb 4 ; Se guarda el numero B

sum\_num: resb 8 ; Para guardar la suma

sub\_num: resb 8 ; Para guardar la resta

mul\_num: resb 8 ; Para guardar la multiplicacion

div\_num: resb 8 ; Para guardar la division

; Para reservar memoria para el resultado de la suma, resta, multiplicacion y division. De igual forma para las variables enteras

result\_num\_dub\_a: resb 4 ; Entero de result\_num\_a

result\_num\_dub\_b: resb 4 ; Entero de result\_num\_b

sum\_num\_dub: resb 4 ; Entero de la suma

sub\_num\_dub: resb 4 ; Entero de la resta

mul\_num\_dub: resb 4 ; Entero de la multiplicacion

div\_num\_dub: resb 4 ; Entero de la division

section .text

main:

Stprint ingresar\_a, ingresar\_a\_len ; Mensaje para que ingrese el primer numero o numero A.

; Accion para leer al Numero a

push result\_num\_a ; Se agrega el espacio en memoria al stack

push scan\_format ; SE agrega el formato de lectura

call scanf ; Se llama a scanf de C

add esp, 8 ; Se lee formato float de 32 bits

Stprint ingresar\_b, ingresar\_b\_len ; Mensaje para que ingrese el primer numero o numero B.

; Accion para leer al Numero a

push result\_num\_b

push scan\_format

call scanf

add esp, 8

saltodelinea

Stprint msg1, msg1\_len

; Para sumar

sub esp, 8

fld dword [result\_num\_a] ; Se carga mi valor A

fadd dword [result\_num\_b] ; Se le aplica la operacion con el valor B

fstp qword [sum\_num] ; Se guarda el valor float64 resultante en sum\_num

push dword[sum\_num+4] ; Se guarda la parte alta de sum\_num (MSB) en el stack para imprimir

push dword[sum\_num] ; Se guarda la parte baja de sum\_num (LSB)

push print\_suma ; Se procede a imprimir

call printf ; Se llama la funcion de printf

add esp, 12 ; 12 bits a esp para retornar flotante en un sistema de 32 bits,

; Para restar

sub esp, 8

fld dword [result\_num\_a]

fsub dword [result\_num\_b]

fstp qword [sub\_num]

push dword[sub\_num+4]

push dword[sub\_num]

push print\_resta

call printf

add esp, 12

; Para multiplicar

sub esp, 8

fld dword [result\_num\_a]

fmul dword [result\_num\_b]

fstp qword [mul\_num]

push dword[mul\_num+4]

push dword[mul\_num]

push print\_multiplicacion

call printf

add esp, 12

; Para dividir

; Ahora aqui comparo que si B es 0 entonces no se puede dividir

mov eax, [result\_num\_b] ; muevo el valor de b a eax

cmp eax, 0 ; si es 0 entonces

je ErrMensaje ; Hago un salto al mensaje

sub esp, 8

fld dword [result\_num\_a]

fdiv dword [result\_num\_b]

fstp qword [div\_num]

push dword[div\_num+4]

push dword[div\_num]

push print\_division

call printf

add esp, 12

jmp continuar ; Si no hago un salto a continuar

ErrMensaje: ; Muestra que es imposible dividir entre 0

Stprint error, error\_len

continuar:

;Aqui procedo a convertir e imprimir

; Convertir de flotante a entero

; A

fld dword [result\_num\_a] ; En priemr lugar procedo a cargar mi flotante

fistp dword [result\_num\_dub\_a] ; Despues convierto a entero concqyuda de fistp.

; B

fld dword [result\_num\_b]

fistp dword [result\_num\_dub\_b]

; Suma

fld qword[sum\_num]

fistp dword[sum\_num\_dub]

; Resta

fld qword[sub\_num]

fistp dword[sub\_num\_dub]

; Multiplicacion

fld qword[mul\_num]

fistp dword[mul\_num\_dub]

; Aqui vuelvo a comparar que si B es 0 entonces no se puede dividir

mov eax, [result\_num\_b]

cmp eax, 0

je cont

; Division

fld qword[div\_num]

fistp dword[div\_num\_dub]

cont:

; Imprimo el valor en decimal y los valores en enteros

saltodelinea

Stprint msg2, msg2\_len

Stprint msgdec, msgdec\_len

push dword [result\_num\_dub\_a] ; Guardo el numero en el stack

push print\_formatA ; Guardo el formato de impresion

call printf

add esp, 12 ; 12 bits a esp para retornar flotante en un sistema de 32 bits,

push dword [result\_num\_dub\_b]

push print\_formatB

call printf

add esp, 12

; Stprint/imprimir el valor en hexadecimal

saltodelinea

Stprint msghex, msghex\_len

push dword [result\_num\_dub\_a] ; Simplemente se toma el valor convertido de float a int, y imprimirlo

push print\_formatA\_hex ; Como se esta usando printf a un valor entero como %x lo imprimira como un hexadecimal

call printf

add esp, 12

push dword [result\_num\_dub\_b]

push print\_formatB\_hex

call printf

add esp, 12

; suma

push dword [sum\_num\_dub]

push print\_suma\_hex

call printf

add esp, 12

; resta

push dword [sub\_num\_dub]

push print\_resta\_hex

call printf

add esp, 12

; multiplicacion

push dword [mul\_num\_dub]

push print\_multiplicacion\_hex

call printf

add esp, 12

; Si B es 0 entonces no se puede dividir

mov eax, [result\_num\_b]

cmp eax, 0

je conta

; Si B es = 0 entonces esto nunca se muestra

; division

push dword [div\_num\_dub]

push print\_division\_hex

call printf

add esp, 12

conta:

; Stprint/imprimir valor en octal

saltodelinea

Stprint msgoct, msgoct\_len

push dword [result\_num\_dub\_a]

push print\_formatA\_oct ; Como se esta utilizando printf a un valor entero como %o lo imprimira como un octal

call printf

add esp, 12

push dword [result\_num\_dub\_b]

push print\_formatB\_oct

call printf

add esp, 12

; suma

push dword [sum\_num\_dub]

push print\_suma\_oct

call printf

add esp, 12

; resta

push dword [sub\_num\_dub]

push print\_resta\_oct

call printf

add esp, 12

; multiplicacion

push dword [mul\_num\_dub]

push print\_multiplicacion\_oct

call printf

add esp, 12

; Si B es 0 entonces no se puede dividir

mov eax, [result\_num\_a]

cmp eax, 0

je terminar

; Si B es = 0 entonces esto nunca se muestra

; division

push dword [div\_num\_dub]

push print\_division\_oct

call printf

add esp, 12

terminar:

; Terminar

xor ebx, ebx

mov eax, 1

int 0x80

ret

**Salida:**

