## **PREGUNTAS AUTOEVALUACIÓN:**

### **Módulo syscall\_write\_puts.s:**

syscall\_write\_puts.s:

#MACROS

.equ SYS\_EXIT, 1

.equ SUCCESS, 0

#DATOS

.data

buffer: .ascii "Hola\n"

#INSTRUCCIONES

.global \_start

.text

\_start:

#PUTS

push $buffer

call puts

#WRITE

#METIENDO LOS ARGUMENTOS EN REGISTROS

mov $4,%eax

mov $1,%ebx

mov $buffer,%ecx

mov $5,%edx

int $0x80

#PONIENDO LOS ARGUMENTOS EN LA PILA

push $buffer

push $1

push $4

call write

mov $SYS\_EXIT, %eax

mov $SUCCESS, %ebx

fin:int $0x80

.end

### **Módulo sumMtoN\_aviso.c:**

#include <stdio.h>

int main()

{

int argumento1 = 1,argumento2 = 5,diferencia,suma = 0;

if (argumento2 < argumento1)

printf("El argumento2 debe ser mayor que el argumento1\n");

diferencia = argumento2-argumento1;

printf("diferencia %d",diferencia);

while(diferencia >= 0)

{

suma = suma+argumento1;

argumento1++;

diferencia--;

}

return suma;

}

### Módulo sumMtoN\_aviso.s:

## MACROS

.equ SYS\_EXIT, 1

## DATOS

.section .data

error: .ascii "El argumento2 debe ser mayor que el argumento1\n"

## INSTRUCCIONES

.section .text

.globl \_start

\_start:

## Paso los argumentos a la subrutina a través de la pila

pushl $1 #push second argument

pushl $5 #push first argument

##COMPROBACION AVISO

movl %esp,%eax

movl 4(%esp),%ebx

sub %eax,%ebx

jns continua

push $error

call printf

## Llamada a la subrutina sum1toN

continua: call sum1toN

## Paso la salida de sum11toN al argumento a la llamada al sistema exit()

mov %eax, %ebx

## Código de la llamada al sistema operativo

movl $SYS\_EXIT, %eax #exit (%ebx is returned)

## Interrumpo la rutina y llamo al S.O.

int $0x80

### Subrutina: sum1toN

### Descripción: calcula la suma de números enteros en secuencia desde el 1º sumando hasta el 2º sumando

### Argumentos de entrada: 1º sumando y 2º sumando

### : los argumentos los pasa la rutina principal a través de la pila:

### 1º se apila el último argumento y finalmente se apila el 1º argumento.

### Argumento de salida: es el resultado de la suma y se pasa a la rutina principal a través del registro EAX.

### Variables locales: se implementa una variable local en la pila pero no se utiliza

###

.type sum1toN, @function # declara la etiqueta sum1toN

sum1toN:

## Próĺogo: Crea el nuevo frame del stack

pushl %ebp #salvar el frame pointer antiguo

movl %esp, %ebp #actualizar el frame pointer nuevo

## Reserva una palabra en la pila como variable local

## Variable local en memoria externa: suma

subl $4, %esp

## Captura de argumentos

movl 8(%ebp), %ebx #1º argumento copiado en %ebx

movl 12(%ebp), %ecx #2º argumento copiado en %ecx

## suma la secuencia entre el valor del 1ºarg y el valor del 2ºarg

## 1º arg < 2ºarg

## utilizo como variable local EDX en lugar de la reserva externa para variable local: optimiza velocidad

## Inicializo la variable local suma

movl $0,%edx

## Número de iteracciones

mov %ecx,%eax

sub %ebx,%eax

bucle:

add %ebx,%edx

inc %ebx

sub $1,%eax

jns bucle

## Salvo el valor de retorno

movl %edx, %eax

## Epílogo: Recupera el frame antiguo

movl %ebp, %esp #restauro el stack pointer

popl %ebp #restauro el frame pointer

## Retorno a la rutina principal

ret

.end

## **MEMORIA:**

Historial de gdb:

+b \_start

Punto de interrupción 1 at 0x1000: file sumMtoN.s, line 18.

+run

Starting program: /home/ubuntu20/Escritorio/sumMtoN

Breakpoint 1, \_start () at sumMtoN.s:18

+x $esp

0xffffd120: 0x00000001

+x $ebp

0x0: No se puede acceder a la memoria en la dirección 0x0

+x /xw $sp

0xffffd120: 0x00000001

+p /s \*(char \*\*)($sp+4)

$1 = 0xffffd2f8 "/home/ubuntu20/Escritorio/sumMtoN"

+n

+n

+step

+x /i \*(int \*)$sp

0x56556009 <\_start+9>: mov %eax,%ebx

+x /i $sp

0xffffd114: or %esp,0x55(%eax)

+n

+n

sum1to () at sumMtoN.s:46

+p $fp

$2 = (void \*) 0xffffd110

+p $sp

$3 = (void \*) 0xffffd110

+x /i \*(int \*)($fp+4)

0x56556009 <\_start+9>: mov %eax,%ebx

+p $eax

$4 = 28

+n

+n

+n

+n

+n

+n

bucle () at sumMtoN.s:63

+n

+n

+n

+n

+n

+n

+n

+n

+n

+n

+n

+n

+n

+n

+n

+n

+n

+n

+n

+n

+p eax

No symbol "eax" in current context.

+p $eax

$5 = -1

+n

+p $eax

$6 = 15

+n

bucle () at sumMtoN.s:73

+n

bucle () at sumMtoN.s:76

+p $fp

$7 = (void \*) 0x0

+p $sp

$8 = (void \*) 0xffffd114

+x \*(int \*)$sp

0x56556009 <\_start+9>: mov %eax,%ebx

+x (int \*)$fp

0x0: No se puede acceder a la memoria en la dirección 0x0

+x \*(int \*)$fp

No se puede acceder a la memoria en la dirección 0x0

+x $fp

0x0: No se puede acceder a la memoria en la dirección 0x0

+p $sp

$9 = (void \*) 0xffffd114

+n

\_start () at sumMtoN.s:25

+p $sp

$10 = (void \*) 0xffffd118

+p $fp

$11 = (void \*) 0x0

+q

Al frame pointer de una rutina que lleve el SO no se puede acceder porque es información que el sistema operativo no nos proporciona.