

Ejercicio /

Dado el siguiente programa que se compila para un procesador de 32 bits:

```
#include<stdio.h>
#include<string.h>

int son_iguales(char *param1, char *param2){
    int igualdad;

    if(strcmp(param1, param2)==0)
        igualdad = 1; //iguales
    else
        igualdad = 0; //distintos

    return igualdad;
}

1 int main(int argc, char *argv[]){
2     char s_linea_comando[4];
3     char s_scanf[4];
4     1 byte
5     s_linea_comando[0] = argv[1][0]; = a0
6     s_linea_comando[1] = argv[1][1]; = a1
7     s_linea_comando[2] = argv[1][2]; = a2
8     s_linea_comando[3] = 0;

9     printf("Ingrese 3 letras consecutivas:\n");
10    scanf("%c%c%c", &s_scanf[0], &s_scanf[1], &s_scanf[2]);
11    s_scanf[3] = 0;

12    if(son_iguales(s_linea_comando, s_scanf))
13        printf("Son iguales: %s", s_scanf);
14    else
15        printf("Son distintos: %s y %s", s_linea_comando, s_scanf);

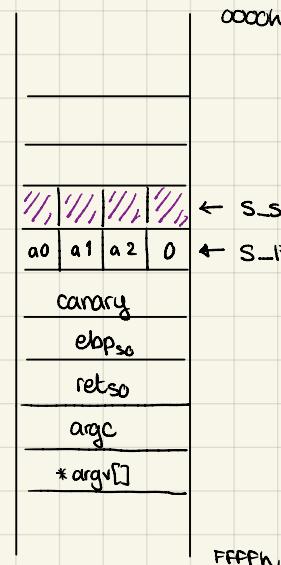
16    return 0;
}
```

1. Explique con dibujos respectivos el contenido de la pila durante toda la ejecución del programa.
2. Muestre los registros del microprocesador que intervienen en los dibujos de la pila.

① línea 1 a 3



② línea 4 a 7



→ ingresa primeros 3 caracteres
del segundo argumento de
línea de comando

e.g. %exc blah



③ Línea 8

	0000h
canary	
ebp_main	
ret_main	
LC0	
MMMMMM	← "Ingres ..."
/// /// \\ \\ \\ \\	← s_scanf[4]
'b' 'L' 'a' 0	← S_linea_Commando[4]
canary	
ebp_so	
retso	
argc	
*argv[]	
	FFFFFh

printf devuelve cantidad de bytes escritos con éxito
 $cax = \text{strlen}(\text{"Ingres ..."})$

← "Ingres ..."
 ← s_scanf[4]
 ← S_linea_Commando[4]

asumir que se ejecuta

add esp, 4*4

para liberar "parámetros"
 de printf

"%c%c%c%c"

digamos que
 ingresamos
 "bla"

④ Línea 9

	0000h
canary	
ebp_main	
ret_main	
LC1	
\\$s_scan[0]	
\\$s_scan[1]	
\\$s_scan[2]	
MMMMMM	
'b' 'L' 'a' [3]	← s_scanf[4]
'b' 'L' 'a' 0	← S_linea_Commando[4]
canary	
ebp_so	
retso	
argc	
*argv[]	
	FFFFFh

scanf devuelve número
 de bytes leidos y
 asignados

↓

eax = 3

⑤ Línea 10 y 11

igualdad	
canary	
ebp_main	
ret_main	
s_linea_Commando	
s_scanf	
MMMMMM	son_iguales como usamos
'b' 'L' 'a' 0	de ejemplo "bla" y
'b' 'L' 'a' 0	asumimos que da igual
canary	eax = 1
ebp_so	
retso	
argc	
*argv[]	
	FFFFFh

son_iguales como usamos
 de ejemplo "bla" y
 asumimos que da igual

eax = 1

← s_scanf[4]

← S_linea_Commando[4]

en asm

el if se representa por como

esta planteada en C

[cmp eax, 1
 je .iguales
 jmp .no_iguales]

asumir que se ejecuta

add esp, 4*6

para liberar "parámetros"
 de son_iguales

⑥ Línea 12

	0000h
canary	
ebp_main	
ret_main	
LC2	
*s_scanf	
MMMMMM	← "son_iguales" [3] %s \n" [17]
/// /// \\ \\ \\ \\	← gcc guarda espacio de más
'b' 'L' 'a' 0	← s_scanf[4]
canary	
ebp_so	
retso	
argc	
*argv[]	
	FFFFFh

EAX_{previo} = 1

0000h

printf devuelve cantidad de bytes escritos con éxito

cax = 17

← "son_iguales" [3] %s \n" [17]

← s_scanf[4]

← S_linea_Commando[4]

⑦ supongamos que en ⑤ daba
 son_iguales (s_linea_Commando, s_scanf) = 0
 entonces se ejecuta Línea 13 osea
 que ⑥ no sucede.

↓
 osea en paso ⑤ eax = 0

⑦ linea 13

EAXprevio = 0

0000h	
canary	
ebp_main	
ret_main	
LC3	
*s_linea_comando	
*s_scanf	
M M M M M M	← gcc guarda espacio de más
'a' 'b' 'c' 0	← s_scant[4]
'b' 'c' 'a' 0	← S_linea_Commando[4]
canary	
ebp_so	
ret_so	
argc	
*argv[]	

FFFFh

⑧ linea 14

asume que se ejecuta
add esp, 4*6
para liberar "parametros"
de print f ⑦

sino para caso ⑥
add esp, 4*5

0000h	
M M M M M M	← gcc guarda espacio de más
'a' 'b' 'c' 0	← s_scant[4]
'b' 'c' 'a' 0	← S_linea_Commando[4]
canary	
ebp_so	
ret_so	
argc	
*argv[]	

FFFFh

eax = 0
↑
return 0;



↓
se termina el programa
se desarma el stack frame
y se "liberan" parametros del
main re-estableciendo el esp=ret_so

② Dado el programa

```
#include <stdio.h>
int suma (char *nombre, int cantidad) {
    int i, resultado;
    for (i=0; i<cantidad; i++)
        resultado += nombre[i];
    return resultado;
}

int main () {
    char cadena [10];
    int i;
    for (i=0; i<10; i++)
        cadena [i] = 2;
    printf ("suma de cadena %s \n", suma (cadena, 10));
    return 0;
}
```

- a) Expresa con dibujos el contenido de la pila durante toda la ejecución. Muestra todo lo que ocurre.
- b) Muestra los valores y retorna cada función y lo forma en que lo hace.
- c) Muestra la salida en pantalla del programa.

cadena[10]	i=10			
	2	2	2	2
	2	2	2	2
	2	2		
	canary			
	ebpso			
	retso			