

Arquitectura del Computador - LCC

Examen recuperatorio Parcial 2 - 29 de noviembre de 2024

Nombre y apellido: _____

Legajo: _____

1. (60 puntos) Dado el siguiente código en lenguaje C:

```
#include <stdio.h>
int a[4]={0xaaaaaaaa, 0x11223344, 16, 4};

int determinante(int a, int b, int c, int d){
    int r = a*d - b*c;
    printf("Determinante: %d * %d - %d * %d = %d\n", a, d, b, c, r);
    return r;
}

int main(){
    int x=a[0], y=a[1], z=a[2], w=a[3];
    printf("Vamos a calcular el determinante de la matriz \n %d \t %d \n %d \t %d \n",x,y,z,w);
    int r=determinante(x,y,z,w);
    return 0;
}
```

(a) Completar el siguiente código equivalente en Assembler x86-64:

```
.data
a: .long 0xaaaaaaaa, 0x11223344, 16, 4
str: .asciz "Vamos a calcular el determinante de la matriz \n %d \t %d \n %d \t %d \n"
str1: .asciz "Determinante: %d * %d - %d * %d = %d\n"
```

```
.text
.global main
main:
```

```
    pushq    %rbp
    movq     %rsp, %rbp
    subq     $32, %rsp
    movl     0, %eax
    movl     %eax, -4(%rbp)
    movl     1, %eax
    movl     %eax, -8(%rbp)
    movl     16, %eax
    movl     %eax, -12(%rbp)
    movl     4, %eax
    movl     %eax, -16(%rbp)
    movl     -4(%rbp), %esi
    movl     -8(%rbp), %edx
    movl     -12(%rbp), %ecx
    movl     -16(%rbp), %r8d
    leaq     str, %rdi
    xorl     %eax, %eax
    call     printf
    movl     -4(%rbp), %edi
    movl     -8(%rbp), %esi
```

rsp = 0x7fffffffefb40 ✓

rsp = 0x7fffffffefb20 ✓

a[0]

a[1]

a[2]

a[3]

rsp = 0x7fffffffefb20 ✓

↳ (después de ejecutar printf,
ya se poppeó el rsp. la
dir. de retorno)


```

movl    -12(%rbp), %edx
movl    -16(%rbp), %ecx
call    determinante
pushq   %rax
xorl    %ecx, %eax
leave
ret

```

determinante:

```

pushq   %rbp
movq    %rsp, %rbp
subq    $32, %rsp
movl    %edi, -4(%rbp)
movl    %esi, -8(%rbp)
movl    %edx, -12(%rbp)
movl    %ecx, -16(%rbp)
movl    -4(%rbp), %eax
imull   -16(%rbp), %eax
movl    %eax, %edx
movl    -8(%rbp), %eax
imull   -12(%rbp), %eax
subl    %eax, %edx
movl    %edx, -20(%rbp)
movl    -4(%rbp), %esi
movl    -16(%rbp), %edx
movl    -8(%rbp), %ecx
movl    -12(%rbp), %ebx
movl    -20(%rbp), %ebx
leaq    -20(%rbp), %rdi
xorl    %eax, %eax
call    printf
movl    -20(%rbp), %eax
leave
ret

```

rsp = 0x7fffffffab48 ✓

rsp = 0x7fffffffefb10 ✓

rsp = 0x7fffffffef0 ✓

a

b

c

d

rax = ..., rdx = 0x0000000000000010 (c)

rax = ..., rdx = 0x0000000000000008

0x0000000012345678

a

d

b

c

r

rsp = 0x7fffffffef0 ✓

rsp = 0x7fffffffef13 ✓

(se empuja al stack por ser var. local)

↳ leave vuelve al %rsp al %rbp y por eso al %rbp orig. dejando al %rsp donde estaba al llegar a main

(el resultado es truncado)

↳ resultado probado anterior

(res. truncado)

- Mostrar cómo se guarda el arreglo de enteros en memoria, asumiendo que la etiqueta a corresponde a la dirección 0x404028.
- Elaborar un diagrama de la pila que detalle el contenido de la memoria junto con las direcciones correspondientes, asumiendo un valor inicial de `rsp=0x7fffffffefb48`.
- Indicar el valor del registro `rsp` tras la ejecución de las instrucciones correspondientes al comentario.
- Especificar el contenido de los registros `rax` y `rdx` después de la ejecución de las instrucciones `imull`.
- Justificar el uso de la instrucción `subq $32, %rsp` en la función `determinante`. ¿Por qué no se usó `subq $20, %rsp`?
- Analizar y justificar si el resultado obtenido será correcto.

2. (40 puntos) Escribir un programa en ensamblador x86-64 que calcule e imprima en pantalla el promedio de las notas ingresadas como argumentos por línea de comandos. Las notas deben ser números enteros y el promedio será un valor de tipo `double`. Finalmente, el programa debe retornar el valor "0".

Si se ingresan menos de dos argumentos, el programa debe imprimir en pantalla el mensaje "Ingrese al menos una nota" y retornar el valor "1".

Ayuda: La función `atoi` (ASCII to Integer) en C permite convertir una cadena de caracteres (de tipo `const char*`) que representa un número en formato de texto a un valor de tipo `int`. Su prototipo es:

```
int atoi(const char *str);
```

Ejercicio	1	2	Total
Puntos	60	40	100
Puntos obtenidos			