

Nombre:

DNI:

Grupo:

Sobre 10, cada respuesta vale 2 si es correcta, 0 si está en blanco o claramente tachada, y -2/3 si es errónea.
Anotar las respuestas (**a, b, c o d**) en la siguiente tabla.

1	2	3	4	5
c	d	a	c	b

1. La instrucción push \$0xff...

- a. Equivale a pushb \$0xff, mete un byte en la pila error, no existe esa instr.
- b. Equivale a pushw \$0x00ff, mete 2 bytes en la pila eso existe, pero no equivale
- c. Equivale a pushl \$0x000000ff, mete 4 bytes en la pila Ej.3.2.5, p.208 libro
- d. Da un error de ensamblado, hace falta indicar el tamaño del operando usando b/w/l no da error, asume pushl

2. Indicar cuál es la dirección de la instrucción mov en el siguiente desensamblado, donde se ha borrado parte de la dirección

```
0804xxxx: 74 12          je 08048391
0804xxxx: b8 00 00 00 00  mov $0, %eax
```

- a. $08048391 + 12 = 08048403$ no, sería -
- b. $08048391 - 12 = 08048379$ no, es 0x12 hex, no decimal
- c. 0804837d sería la de je, no la de mov
- d. 0804837f Ej.3.3.15c, p.226 libro y Tema2.3, tr.7

3. Respecto a funciones recursivas y reentrantes...

- a. Toda función recursiva es reentrante, aunque no toda función reentrante es recursiva Tema2.3, tr.10
- b. Toda función reentrante es recursiva, aunque no toda función recursiva es reentrante
- c. Ambos calificativos son sinónimos, se prefiere llamarlas “funciones recursivas” en lenguajes de alto nivel, y “subrutinas reentrantes” en el contexto de lenguaje ensamblador, contadores de programa y direcciones de retorno
- d. Los dos calificativos no tienen nada que ver, son características independientes (ortogonales), pueden ponerse ejemplos de funciones que sean de un tipo, del otro, de ambos y de ninguno

4. El ajuste de marco de pila que gcc (Linux/IA-32) prepara para todas las funciones consiste en las instrucciones...

- a. `movl %ebp, %esp`
`pushl %ebp`
- b. `movl %esp, %ebp`
`pushl %esp`
- c. `pushl %ebp`
`movl %esp, %ebp` Tema2.3, tr.28
- d. `pushl %esp`
`movl %ebp, %esp`

5. Al traducir de lenguaje C a ensamblador, gcc en máquinas Linux/IA-32 almacena (reserva espacio para) una variable “var” local a una función “fun” en una dirección de memoria referenciable (en lenguaje ensamblador) como...

- a. var (el nombre de la variable representa su posición de memoria)
- b. `-k(%ebp)`, siendo k un número constante relativamente pequeño Tema2.3, tr.46-48
- c. `k(%ebp)`, siendo k un número constante relativamente pequeño
- d. `k(%esp)`, siendo k un número constante relativamente pequeño