

<b>Nombre:</b>	
<b>DNI:</b>	<b>Grupo:</b>

## Examen de Problemas (3,0 p)

**1. Ensamblador.** (0.8 puntos). La siguiente es una función genérica para intercambiar los bytes de una variable numérica cuyo tipo se especifica mediante typedef. En este ejemplo concreto `number_t` es un alias del tipo `long` (8 bytes en x86-64).

```
#include <stddef.h>

typedef long number_t;
#define NUMBER_SIZE sizeof(number_t)

number_t big2little (number_t n) {
    union {
        number_t n;
        char b[NUMBER_SIZE];
    } src, dst;

    src.n = n;
    for (size_t i = 0; i < NUMBER_SIZE; i++)
        dst.b[i] = src.b[NUMBER_SIZE-1 - i];

    return dst.n;
}
```

Escriba el código de una función en ensamblador de x86-64 que realice la misma operación (devolver el número de 8 bytes pasado como argumento con los bytes intercambiados).

*Ayuda:*

- 1. `size_t` es equivalente a `unsigned long`*
- 2. los dos campos `n` y `b[NUMBER_SIZE]` de la unión están solapados en memoria (ocupan el mismo espacio)*
- 3. `sizeof(src)` es 8, `sizeof(dst)` es 8*
- 4. para almacenar `src` y `dst` se puede usar la zona roja por debajo de `rsp` sin tener que cambiar el valor de `rsp`, es decir, que se puede acceder directamente a `-8(%rsp)`, `-16(%rsp)`, etc.*

**2. Ensamblador** (0.2 puntos). Una función devuelve el producto escalar entre dos vectores, definido como:

$$A \cdot B = A_1B_1 + A_2B_2 + \dots + A_nB_n$$

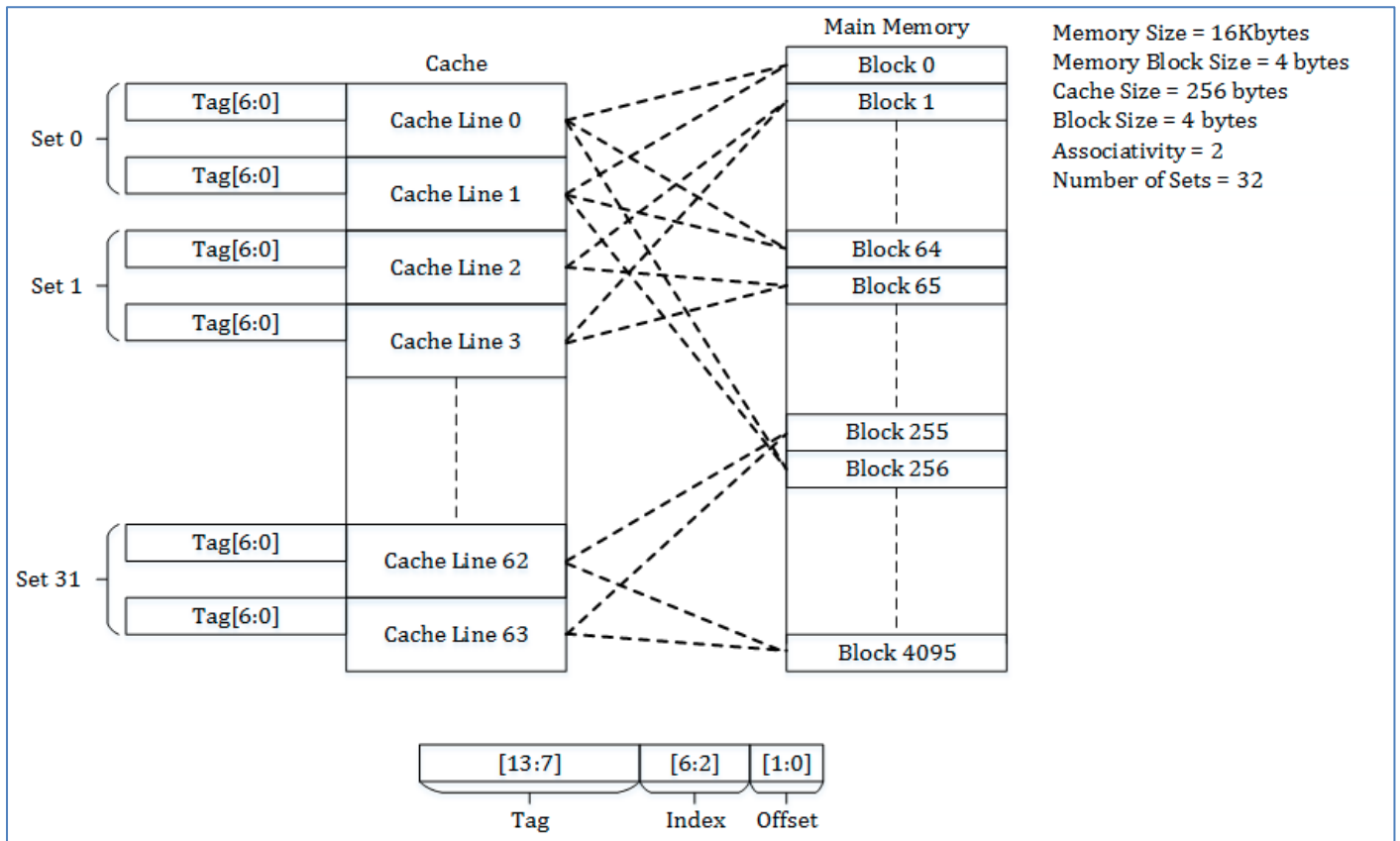
Al compilar dicha función se obtiene el siguiente código en ensamblador:

```
producto_escalar:
    testq    %rdx, %rdx
    je       .L4
    xorl     %eax, %eax
    xorl     %r8d, %r8d
.L3:
    movq     (%rdi,%rax,8), %rcx
    imulq    (%rsi,%rax,8), %rcx
    addq     $1, %rax
    addq     %rcx, %r8
    cmpq     %rax, %rdx
    jne      .L3
    movq     %r8, %rax
    ret
.L4:
    xorl     %r8d, %r8d
    movq     %r8, %rax
    ret
```

- a) (0.1p) ¿De qué tipo es cada uno de los elementos de los vectores A y B?
- b) (0.1p) ¿De cuántas dimensiones es el espacio vectorial, es decir, cuántos elementos tiene cada uno de los vectores A y B?

- 3. Unidad de control** (0.5 puntos). Dibuje un camino de datos de un único bus para una arquitectura de registros de propósito general con dos operandos para las instrucciones del tipo suma, resta, and, etc.
- 4. Entrada/Salida** (0.5 puntos). Disponemos de un microprocesador de 8 bits (bus de datos de 8 bits y bus de direcciones de 16 bits) con E/S independiente. Diseñe un sistema de E/S que permita acceder a los siguientes puertos: puerto 0x240 de entrada y 0x241 de salida. Utilice lógica de decodificación distribuida. No emplee decodificadores.
- 5. Configuración de memoria** (0.5 puntos). Un módulo de memoria DIMM tiene un tamaño de 8 GB en configuración de 1G×64 y está constituido por módulos de memoria DRAM de 1G×4. El módulo DIMM tiene, entre otras líneas, las siguientes: A14-A0, D63-D0, RAS#, CAS# y WE#. Dibuje un esquema del módulo DIMM incluyendo cada uno de sus chips y las conexiones de patillas.

**6. Cache (0.5 puntos).** El siguiente esquema de Wikimedia Commons muestra la estructura general de una cache asociativa por conjuntos:



[https://commons.wikimedia.org/wiki/File:Set-Associative\\_Cache\\_Snehal\\_Img.png](https://commons.wikimedia.org/wiki/File:Set-Associative_Cache_Snehal_Img.png)

Dibuje un esquema similar al anterior, pero con los números correctos para la cache L3 del procesador Intel Core i7-10710U, que reúne las siguientes características y usa un tamaño de línea (= bloque) de 64 bytes:

Level 1 cache size ?	6 x 32 KB 8-way set associative instruction caches 6 x 32 KB 8-way set associative data caches
Level 2 cache size ?	6 x 256 KB 4-way set associative caches
Level 3 cache size	12 MB 12-way set associative shared cache
Physical memory	64 GB