

**Nombre:****DNI:****Grupo:**

## Examen de Problemas (3.0p)

- 1. Acceso a arrays (0.5 puntos).** Considerar el código C mostrado abajo, donde H y J son constantes declaradas mediante #define.

```
int array1[H][J];
int array2[J][H];

void copy_array(int x, int y) {
    array2[y][x] = array1[x][y];
}
```

Suponer que ese código C genera el siguiente código ensamblador x86-64:

```
# A la entrada:
#   %edi = x
#   %esi = y
#
copy_array:
    movslq    %esi,%rsi
    movslq    %edi,%rdi
    movq      %rsi,%rdx
    salq      $3,%rdx
    subq      %rsi,%rdx
    addq      %rdi,%rdx
    leaq      (%rdi,%rdi,4),%rax
    addq      %rsi,%rax
    movl      array1(,%rax,4),%eax
    movl      %eax,array2(,%rdx,4)
    ret
```

¿Cuáles son los valores de H y J?

H = 7

J = 5

- 2. Representación y acceso a estructuras (0.5 puntos).** En el siguiente código C, las declaraciones de los tipos `type1_t` y `type2_t` se han hecho mediante `typedef`'s, y la constante `CNT` se ha declarado mediante `#define`.

```
typedef struct {
    type1_t y[CNT];
    type2_t x;
} a_struct;

void p1(int i, a_struct *ap) {
    ap->y[i] = ap->x;
}
```

La compilación del código para IA32 produce el siguiente código ensamblador:

```
# i en 8(%ebp), ap en 12(%ebp)
# movsbw = move signed byte to word (movw = move word)
p1:
    pushl    %ebp
    movl     %esp, %ebp
    movl     12(%ebp), %eax
    movsbw   28(%eax), %cx
    movl     8(%ebp), %edx
    movw     %cx, (%eax, %edx, 2)
    popl     %ebp
    ret
```

Indicar una combinación de valores para los dos tipos y CNT que pueda producir el código ensamblador mostrado arriba:

type1\_t:

type2\_t:

CNT:

**3. Memoria cache** (0.5 puntos). Los parámetros que definen la memoria de un computador son los siguientes:

- Tamaño de la memoria principal: 32 K palabras.
- Tamaño de la memoria cache: 4 K palabras.
- Tamaño de bloque: 64 palabras.

Dibujar el formato de una dirección de memoria desde el punto de vista del sistema cache, determinando el tamaño de cada campo, y anotar los cálculos realizados para obtener dichos tamaños, para las siguientes políticas de colocación:

A. Totalmente asociativa.

B. Por correspondencia directa.

C. Asociativa por conjuntos con 16 bloques por conjunto.

- 4. Diseño del sistema de memoria** (0.5 puntos). Disponemos de una CPU con buses de datos y direcciones de 16bit. Diseñar un sistema de memoria para la misma a partir de módulos SRAM de 16Kx8 y ROM de 8Kx4. La memoria ROM debe ocupar las direcciones 0x0000 a 0x3FFF y la SRAM 0x4000 a 0xFFFF. Se valorará la simplicidad del diseño.

- 5. Entrada/Salida** (0.5 puntos). Dibuje un módulo sencillo de E/S capaz de realizar la interfaz entre los buses de un procesador y una impresora usando direccionamiento en memoria (no independiente). El módulo contará con un decodificador que activará una señal SEL cuando la dirección de memoria sea 10111100 o 10111101. La primera de estas dos direcciones corresponderá a dos registros, uno de estado y otro de control ( $A0 = 0$ ), y la segunda al registro de datos ( $A0 = 1$ ). Los registros de estado y control pueden tener una única dirección asociada porque el primero sólo se lee (RD) y el segundo sólo se escribe (WR) desde el bus.

**6. Unidad de control microprogramada (0.5 puntos).** Sea la ruta de datos de la figura, donde la ALU puede realizar las siguientes operaciones: R, L, R+1, R-1, L+1, L-1, R+L y R-L. Todos los registros son de 8 bits.

- Indique las señales de control necesarias para esta arquitectura.
- Diseñe un formato de microinstrucción
- Indique la secuencia de señales de control necesaria para leer de memoria una instrucción máquina y saltar a su fase de ejecución.

