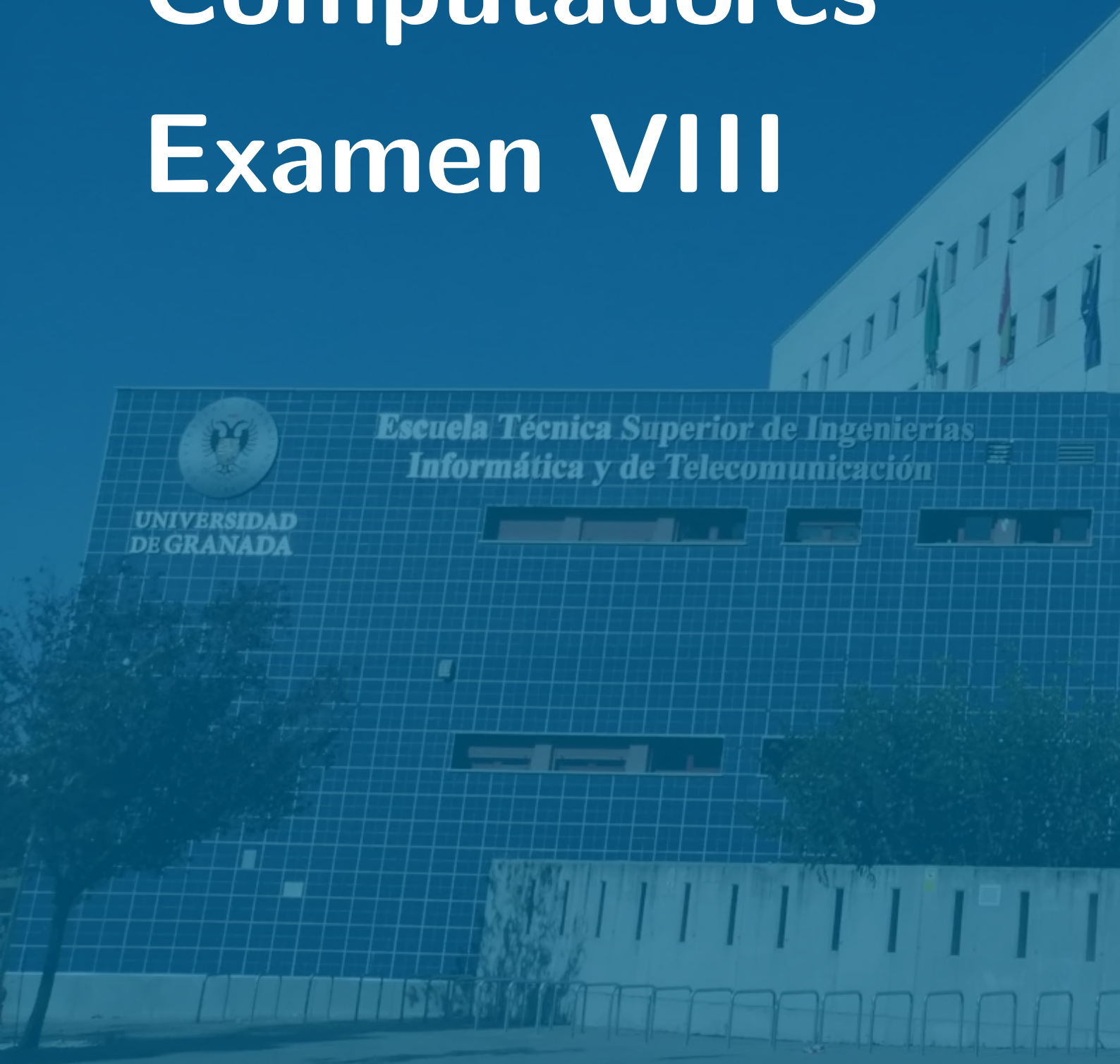
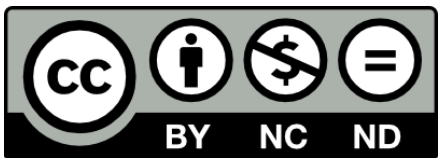
Estructura de Computadores Examen VIII



Los Del DGIIM, losdeldgiim.github.io

Doble Grado en Ingeniería Informática y Matemáticas
Universidad de Granada



Estructura de Computadores Examen VIII

Los Del DGIIM, `losdeldgiim.github.io`

Granada, 2025

Asignatura Estructura de Computadores.

Curso Académico 2024/25.

Grado Doble Grado en Ingeniería Informática y Matemáticas.

Grupo Único.

Profesor Ignacio Rojas Ruíz.

Descripción Examen de la Convocatoria Ordinaria.

Fecha 16 de enero del 2025.

Duración 2 horas.

Examen Tipo Test (3 puntos)

Todas las preguntas son de elección simple sobre 4 alternativas. Cada respuesta vale $^{3}/_{30}$ si es correcta, 0 si está en blanco o claramete tachada, $^{-1}/_{30}$ si es errónea.

Ejercicio 1. En la práctica "media" se pide como ejercicio previo sumar una lista de N enteros SIN signo produciendo un resultado `.quad` (doble precisión). El programa esqueleto ofrecido (`suma.s`, que suma la conocida lista de 9 elementos en simple precisión) no es válido. Se podría comprobar imprimiendo el resultado SIN signo y usando como contraejemplo las siguientes listas:

a) `0x4000 0000` y `0x8000 0000` (el usuario piensa que sus datos NO tienen signo).

b) `0x8000 0000` y `0xC000 0000`.

c) Ninguna de las dos.

d) Ambas.

Ejercicio 2. En la práctica "media" se pide como ejercicio previo sumar una lista de N enteros CON signo produciendo un resultado `.quad` (doble precisión). El programa esqueleto ofrecido (`suma.s`) no es válido. Se podría comprobar imprimiendo el resultado CON signo, y usando como contraejemplo las siguientes listas:

a) -1 y -1 (el usuario piensa que sus datos tienen signo).

b) `0x4000 0000` y `0x4000 0000`.

c) Ninguna de las dos.

d) Ambas.

Ejercicio 3. Suponga una memoria cache con las siguientes propiedades:

- Tamaño: 512 bytes.
- Política de reemplazo: LRU.
- Estado inicial: vacía (todas las líneas válidas).

Suponga que para la siguiente secuencia de direcciones enviadas a la cache: 0, 10, 16, 20, 30, 32, 40, 50, 60, 64, la tasa de acierto es 0,50. ¿Cuál es el tamaño de la línea de la cache?

a) 4 bytes.

b) 8 bytes.

c) 16 bytes.

d) Ninguo de los anteriores.

Ejercicio 4. Sea un computador de 64 bits con una memoria cache L1 para datos de 32 KB y líneas de 64 bytes asociativa por conjuntos de 8 vías. Dado el siguiente fragmento de código:

```
char v[262144];
for(i = 0; i < 262144; i+=8)
   v[i] = 9;
```

¿Cuál será la tasa de fallos aproximada que se obtiene en la ejecución del bucle anterior?

a) 1/2 (mitad aciertos, mitad fallos).

b) 1/4 (un fallo por cada 4 accesos).

c) 1/8 (un fallo por cada 8 accesos).

d) 1 (todo son fallos).

Ejercicio 5. En una bomba como las estudiadas en prácticas, del tipo:

```
0x080486e8: call 0x8048524 <strncmp>
0x080486ed: test %eax, %eax
0x080486ef: je 0x80486f6 <main+134>
0x080486f1: call 0x8048604 <boom>
0x080486f6: ...
```

la contraseña correcta es:

a) el `string` almacenado a partir de `0x8048524`.

b) el `string` almacenado a partir de `0x80486f6`.

c) el `string` almacenado a partir de `0x8048604`.

d) Ninguna de las otras opciones.

Ejercicio 6. La lectura de un elemento de un array anidado neceita los siguientes accesos a memoria:

a) Ninguna respuesta es correcta.

b) Uno.

c) Dos.

d) Tres.

Ejercicio 7. ¿En qué registro está contenido el último dato (o instrucción) leído de memoria, o el dato que se va a escribir en memoria?

a) Registro de propósito general.

b) MAR.

c) PC.

d) MBR/MDR.

Ejercicio 8. ¿Cuál de las siguientes instrucciones máquina es incorrecta en `x86-64`?

a) `movl %r8, %eax`.

b) `testl %edx, %edx`.

c) `addq $1, %rcx`.

d) `movl (%rdi,%rcx,4), %edx`.

Ejercicio 9. Si `ECX` vale 0, la instrucción `adc $0, %ecx`:

a) Pone `CF=0`.

b) Pone `CF=1`.

c) No cambia `CF`.

d) Cambia `CF`.

Ejercicio 10. Dada la siguiente definición de datos:

```
lista: .int 0x10000000, 0x50000000,
            0x10000000, 0x20000000
longlista: .int (.-lista)/4
resultado: .quad 0x123456789ABCDEF
formato: .ascii "suma=%llu=%llx hex\n\0"
```

la instrucción `movl longlista %ecx` copia el siguiente valor:

a) 16.

b) 8.

c) 32.

d) 4.

Ejercicio 11. Dado el siguiente fragmento de progama:

```
.section .data
lista: .int 1,2,0x10,3,-3
longlista: .int .-lista
resultad0: .quad 0

.section .text
main: .global main

xor %edx, %edx
mov $-12, %eax
cltd
mov longlista, %ebx

idiv %ebx
```

El valor de `%rdx` después de la división es:

a) `0x00000004`.

b) Ninguna de las soluciones es correcta.

c) `0x00000010`.

d) `0xFFFFFFF4`.

Ejercicio 12. Considere la siguiente declaración:

```
struct rec{
   int i;
   int j;
   int a[10];
   int *p;
};
```

y una función `void f(struct rec *r);` cuyo código en ensamblador es:

```
mov 0x4(%rdi), %eax
add (%rdi), %eax
cltq # RAX <- (long) EAX
lea 0x8(%rdi,%rax,4), %rax
mov %rax, 0x30(%rdi)
retq
```

¿Cuál es el código C de la función `f`?

a) `r->a[r->i] = r->j;`

b) `r->p = &(r->a[r->i + r->j]);`

c) `r->a[r->i] = r->a[r->j];`

d) `r->p = (int *)(long)(r->a[r->i] + r->a[r->j]);`

Ejercicio 13. Dada una matriz de 5×3 enteros, una posible traducción a ensamblador de una función que devuelve el elemento i, j:

```
int elemn(int A[5][3], size_t i, size_t j);
```

es:

a)
```
movq (%rdi, %rsi, 4), %rax
movl (%rax, %rdx, 4), %eax
ret
```

b)
```
leaq (%rdx, %rsi, 4), %rax
movl (%rdi, %rax, 4), %eax
ret
```

c)
```
leaq (%rsi, %rsi, 4), %rax
leaq (%rdx, %rdx, 2), %rdx
addq %rdx, %rax
movl (%rad, %rdi), %eax
ret
```

d)
```
leaq (%rsi, %rsi, 2), %rax
leaq (%rdi, %rax, 4), %rax
movl (%rax, %rdx, 4), %eax
ret
```

Ejercicio 14. ¿Cuál de las siguientes afirmaciones acerca de la memoria es FALSA?

a) La memoria dinámica usa señales de control `RAS#` y `CAS#`.

b) Las celdas de memoria dinámica están construidas por un transistor y un condensador.

c) Las celdas de memoria estática tienen que ser constantemente refrescadas.

d) La memoria estática se emplea en las caches L1 y L2.

Ejercicio 15. Al traducir la sentencia C

```
r->i = val;
```

`gcc` genera el código ASM:

```
movl %edx, 12(%rax)
```

Se puede deducir que:

a) `i` es un entero que vale 12.

b) El desplazamiento de `i` en `*r` es 12.

c) `r` es un puntero que apunta a la posición de memoria 12.

d) `val` es un entero que vale 12.

Ejercicio 16. ¿Cuántas líneas de dirección (patillas) son necesarias para direccionar un chip de memoria DRAM de 4096×8?

a) 6.

b) 10.

c) 11.

d) 12.

Ejercicio 17. ¿Cuántas líneas de dirección son necesarias en una memoria RAM de 256 K palabras dinámica? ¿Y estática?

a) 9/9.

b) 9/18.

c) 18/9.

d) 18/18.

Ejercicio 18. ¿Qué necesitamos para construir una memoria de 1K×8 bits?

a) 64 memorias de 128×1 bits.

b) 8 memorias de 512×2 bits.

c) 8 memorias de 256×4 bits y un decodificador de 2 a 4.

d) 4 memorias de 512×2 bits y un decodificador de 2 a 4.

Ejercicio 19. En un sistema con memoria direccionable por bytes, ¿cuál sería el tamaño de una línea de cache, si la cache del procesador fuera de 4 MB, asociativa por conjuntos de 16 vías, y contuviera 4096 conjuntos?

a) 16 B.

b) 32 B.

c) 64 B.

d) 128 B.

Ejercicio 20. En una caché con 64 bytes de longitud de línea, ¿cuál es la dirección de memoria de la primera palabra de la línea de caché que contenga la posición de memoria `0xBEE3DE72`?

a) `0xBEE3DE6E`.

b) `0xBEE3DE70`.

c) `0xBEE3DE40`.

d) `0x0EE3DE72`.

Ejercicio 21. Se definen:

```
struct S1 { int i[3]; char c[3]; doble v } p[3];
union U1 { int i[3]; char c[3]; doble v } q[3];
```

Indica qué afirmación es correcta.

a) `q` tiene un tamaño mayor que `p`.

b) `q` no está alineada y `p` sí.

c) `p` tiene un tamaño mayor que `q`.

d) `p` y `q` tienen el mismo tamaño.

Ejercicio 22. Dada la siguiente estructura:

```
struct W { int j[2]; char s[8]; short a[4]; long *j; };
```

Indique el tamaño total de la estructura:

a) 30.

b) 32.

c) 26.

d) 28.

F

Ejercicio 23. Dado un vector `W` de enteros (en `%rdx`), y un índice `j` (en `%rcx`). La expresión `*(W+j-4)` podría traducirse en una sentencia de ensamblador, del tipo:

a) `movl -16(%rdx, %rcx, 4), %eax`

b) `leaq 4(%rdx, %rdx, 4), %rax`

c) `movl (%rdx, %rcx, 4), %eax`

d) `movl -12(%rdx, %rcx, 4), %eax`

Ejercicio 24. Para leer un dato de un array multi-nivel, se requieren acceder a memoria

a) Ninguna respuesta es correcta.

b) Tres accesos a memoria.

c) Dos accesos a memoria.

d) Una sola vez.

Ejercicio 25. Dadas las siguientes estructuras:

```
struct W2 { char w[16]; char *c[2] };
struct W1 { short i; int c; int *j; short *d };
struct W { struct W2 q[2]; struct W1 z };
```

Calcule el tamaño necesario para almacenar `W`. Indique el tamaño total de la estructura.

a) Todas las respuestas son incorrectas.

b) 38.

c) 40.

d) 36.

Ejercicio 26. ¿Qué valor contendrá el registro `RDX` tras ejecutar las dos instrucciones siguientes?

```
movq $-1, %rdx
shr $16, %edx
```

a) `0xFFFF FFFF FFFF 0000`.

b) `0xFFFF FFFF FFFF FFFF`.

c) `0x0000 0000 0000 FFFF`.

d) `0xFFFF FFFF 0000 FFFF`.

Ejercicio 27. ¿Cuál de las siguientes instrucciones es errónea? (sale mensaje de error al intentar ensamblar):

a) `movb %sil, (%rax)`

b) `pushq $0xFF`

c) `movsbw (%rax), %dx`

d) `movzlq %edx, %rax`

Ejercicio 28. Se dispone de un procesador con una frecuencia de reloj de 1 GHz. Se le conecta un dispositivo que genera 100000 interrupciones por segundo. La rutina de servicio de interrupción ejecuta 500 instrucciones. El número medio de ciclos por instrucción es 2. ¿Qué porcentaje del tiempo dedica el procesador al dispositivo?

a) 1 %.

b) 10 %.

c) 50 %.

d) 90 %.

Ejercicio 29. ¿Cuántos conjuntos tendría una cache de 256 KB asociativa por conjuntos de 16 vías con línea de 64 bytes?

a) 4.

b) 16.

c) 64.

d) 256.

Ejercicio 30. ¿Qué componentes necesitamos para construir una memoria de 2K $\times$8 bits?

a) 64 memorias de 128 $\times$ 1 bits y un decodificador de 4 a 16.

b) 8 memorias de 512 $\times$ 2 bits y un decodificador de 1 a 2.

c) 8 memorias de 256 $\times$ 4 bits y un decodificador de 2 a 4.

d) 16 memorias de 512 $\times$ 2 bits y un decodificador de 2 a 4.

Examen de Problemas (3 puntos)

Ejercicio 1 (1 punto)**.** Una función, `switcher`, tiene la siguiente estructura general (Figura 2). Rellena las líneas en blanco del código C con ayuda del código ensamblador que se adjunta (Figura 1).

```
    movl   8(%ebp), %eax
    cmpl   $7, %eax
    ja     .L2
    jap    *.L7(,%eax,4)
.L2:
    movl   12(%ebp), %eax
    jmp    .L8
.L5:
    movl   $4, %eax
    jap    .L8
.L6:
    movl   12(%ebp), %eax
    xorl   $15, %eax
    movl   %eax, 16(%ebp)
.L3:
    movl   16(%ebp), %eax
    addl   $112, %eax
    jmp    .L8
.L4:
    movl   16(%ebp), %eax
    addl   12(%ebp), %eax
    sall   $2, %eax
.L8:
.L7:
    .long .L3
    .long .L2
    .long .L4
    .long .L2
    .long .L5
    .long .L6
    .long .L2
    .long .L4
```

Figura 1: Código de `switcher` en ensamblador.

```c
int switcher(int a, int b, int c)
{
   int answer;
   switch(a){
      case ______:   /* Case A */
         c = ______;
         /* Fall through */
      case ______:   /* Case B */
         answer = ______;
         break;
      case ______:   /* Case C */
      case ______:   /* Case D */
         answer = ______;
         break;
      case ______:   /* Case E */
         answer = ______;
         break;
      default:
         answer = ______;
   }
   return answer;
}
```

Figura 2: Código C a completar.

Ejercicio 2 (1.25 puntos)**.** La siguiente subrutina en C++ (Figura 3) implementa la función sumatoria. Al compilar el programa en un sistema de 64 bits da lugar al código ensamblador que podemos ver (Tabla 1). Supongamos que el procesador utiliza una **caché de correspondencia asociativa por conjuntos** con 8 marcos de bloque, 8 palabras por bloqe y 4 bloques por conjunto. Recuerde que se direcciona a nivel de bytes (1 palabra = 1 byte). El algoritmo de reemplazo es LRU (menos reciementemente usado).

- Simule la ejecución del programa y muestre el contenido de la caché en cada instante (primeras 7 instrucciones).
- ¿Cuál es la tasa de acierto de la caché tras la ejecución de las primeras 7 instrucciones del programa?

Condiciones iniciales:

- `$eax = 0x600d90$`.
- `$rbx = 0x600d90$`.
- `$edi = 0x3$`.
- `$rip = 0x400828$`.
- `$rsp = 0x7fffe16e4c18$`.

```
int sum(int x)
{
   if (x<2)
      return x;
   else
      return x + sum(x-1);
}
```

Figura 3: Programa en C++.

Dirección	Codificación	Instrución	Comentario
400828	83 ff 01	`cmp edi, 0x1`	
40082b	53	`push rbx`	
40082c	89 fb	`mov ebx, edi`	
40082e	7e 0a	`jle 40083a`	
400830	8d 7b ff	`lea edi, [rbd-1]`	edi = rbx - 1
400833	e8 f0 ff ff ff	`call 400828`	apila rip y salta
400838	01 c3	`add ebx, eax`	
40083a	89 d8	`mov eax, ebx`	
40083c	5b	`pop rbx`	
40083d	c3	`ret`	

Tabla 1: Código ensamblador.

Ejercicio 3 (0.75 puntos)**.** Configuración de memoria. Partiendo de circuitos DRAM de 1K ×1 bits, dibuje un sistema de memoria direccionable por palabras de 2 bytes, para llegar a direccionar 4 K palabras. Indique la primera y la última dirección en hexadecimal de cada fila de circuitos. Indique el número de patillas del bus de direcciones para la nueva memoria construida.