

Grupo ARCOS

**uc3m** | Universidad **Carlos III** de Madrid

# Objetivos y presentación del curso

Estructura de Computadores  
Grado en Ingeniería Informática



# Estructura de Computadores en la UC3M

- ▶ Asignatura obligatoria de segundo curso de:
  - ▶ Grado en Ingeniería Informática
  - ▶ Doble Grado en Ingeniería Informática y Administración de Empresas

# Estructura de Computadores en la UC3M

---

2º

Estructura de Computadores

# Estructura de Computadores en la UC3M

1º

Programación

Tecnología de Computadores

Estructuras de Datos y Algorit.

2º

Estructura de Computadores

# Estructura de Computadores en la UC3M

1º

Programación

Tecnología de Computadores

Estructuras de Datos y Algorit.

2º

Estructura de Computadores

Sistemas Operativos

# Estructura de Computadores en la UC3M

1º

Programación

Tecnología de Computadores

Estructuras de Datos y Algorit.

2º

Estructura de Computadores

Sistemas Operativos

3º

Arquitectura de Computadores

Redes de Ordenadores

Organización de Computadores

Sistemas Distribuidos

# Estructura de Computadores

1º Programación Tecnología de Computadores Estructuras de Datos y Algorit.

---

2º Estructura de Computadores

Sistemas Operativos

---

3º Arquitectura de Computadores Redes de Ordenadores

Organización de Computadores Sistemas Distribuidos

---

4º Sistemas de Tiempo Real Desarrollo de SW de sistemas Panorámica de las Com. digitales

---

# Página Web de la asignatura

<http://www.arcos.inf.uc3m.es/~infoec/>

- ❑ Toda la información sobre la asignatura estará en esta página
- ❑ También en [Aula Global](#)

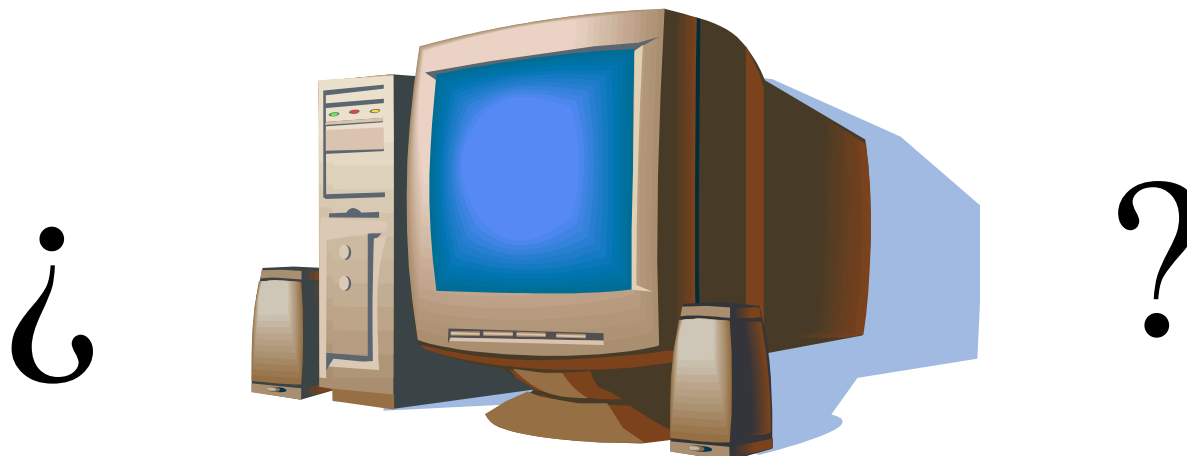
Profesor coordinador:

- ❑ Félix García Carballeira  
[felix.garcia@uc3m.es](mailto:felix.garcia@uc3m.es)



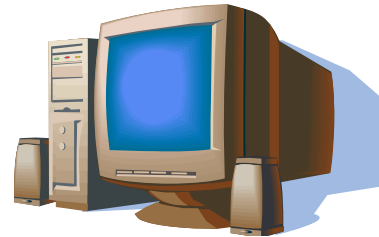


# Objetivos del curso



Conocer y entender los componentes y el funcionamiento básico de un computador

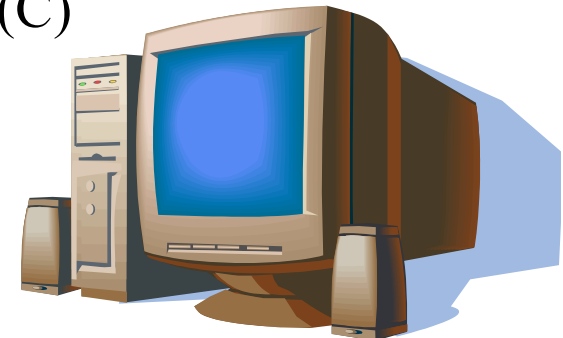
# Cualquier tipo de computador



# Entender cómo se ejecuta un programa

```
temp = v[k];  
v[k] = v[k+1];  
v[k+1] = temp;
```

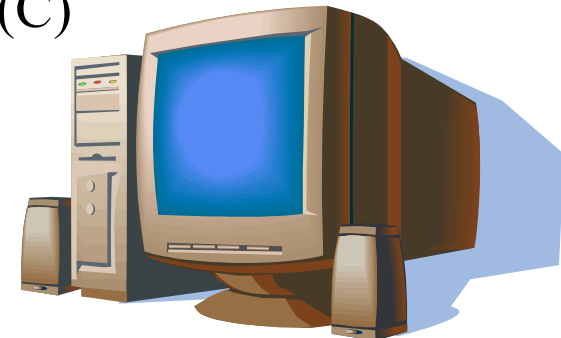
Lenguaje de alto nivel (C)



# Entender cómo se ejecuta un programa

```
temp = v[k];  
v[k] = v[k+1];  
v[k+1] = temp;
```

Lenguaje de alto nivel (C)



```
0000 1001 1100 0110 1010 1111 0101 1000  
1010 1111 0101 1000 0000 1001 1100 0110  
1100 0110 1010 1111 0101 1000 0000 1001  
0101 1000 0000 1001 1100 0110 1010 1111
```

Instrucciones máquina

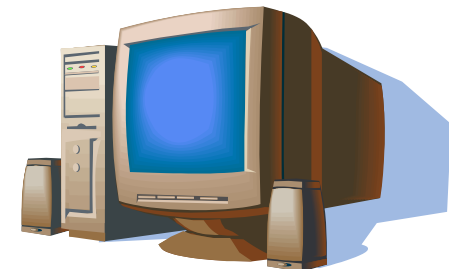
# Entender cómo se ejecuta un programa

```
temp = v[k];  
v[k] = v[k+1];  
v[k+1] = temp;
```

Lenguaje de alto nivel (C)

```
lw    $t0, 0($2)  
lw    $t1, 4($2)  
sw    $t1, 0($2)  
sw    $t0, 4($2)
```

Lenguaje ensamblador



```
0000 1001 1100 0110 1010 1111 0101 1000  
1010 1111 0101 1000 0000 1001 1100 0110  
1100 0110 1010 1111 0101 1000 0000 1001  
0101 1000 0000 1001 1100 0110 1010 1111
```

Instrucciones máquina

# Entender cómo se ejecuta un programa

```
temp = v[k];  
v[k] = v[k+1];  
v[k+1] = temp;
```

```
lw    $t0, 0($2)  
lw    $t1, 4($2)  
sw    $t1, 0($2)  
sw    $t0, 4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000  
1010 1111 0101 1000 0000 1001 1100 0110  
1100 0110 1010 1111 0101 1000 0000 1001  
0101 1000 0000 1001 1100 0110 1010 1111
```



Compilador

Ensamblador

# Entender cómo se ejecuta un programa

## Ejemplo 1

```
int n;  
n = 40000;  
printf( "%d \n", n *n );  
  
n = 50000;  
printf( "%d \n", n *n );
```

► ¿Es correcta esta salida?

1600000000

-1794967296

# Entender cómo se ejecuta un programa

## Ejemplo 2

```
float x, y , z;
```

```
x = 1.0e20;
```

```
y = -1.0e20;
```

```
z = 3.14;
```

```
printf("%f\n", (x + y) + z);
```

```
printf("%f\n", x + (y + z));
```

- ¿Se cumple  $(x+y) + z == x + (y+z)$ ?



# Entender cómo se ejecuta un programa

## Ejemplo 3

### ► Código 1

```
int a[N][N]
for (i=0; i < N; i++)
    for (j=0; j < N; j++)
        sum = sum + a[i][j];
```

### ► Código 2

```
int a[N][N]
for (j=0; j < N; j++)
    for (i=0; i < N; i++)
        sum = sum + a[i][j];
```

- ¿Hacen lo mismo?
- ¿Tardan lo mismo en ejecutarse?

# Entender cómo se ejecuta un programa

## Ejemplo 4

```
#include <stdio.h>

#define BLOCK_SIZE    512

void main(int argc, char **argv)
{
    int fde, fds;
    char buffer[BLOCK_SIZE];
    int n;

    fde = open(argv[1], 0);
    fds = creat(argv[2], 0666);

    while((n = read(fde, buffer, BLOCK_SIZE)) > 0)
        write(fds, buffer, n);

    close(fde);
    close(fds);

    return;
}
```

¿Qué ocurre si BLOCK\_SIZE = 8192?

# Entender cómo se ejecuta un programa

## Ejemplo 5

¿Dado el siguiente fragmento?

```
if (i == (int)((float) i))  
{  
    printf("true");  
}
```

¿Se ejecuta siempre la función `printf()`?

# Entender cómo se ejecuta un programa

## Ejemplo 6

- ▶ ¿Se puede intercambiar el valor de dos variables sin usar una variable intermedia?
- ▶ ¿Cómo se puede saber si el número de bits igual a 1 de una variable long de Java es par (de forma eficiente)?

# Entender cómo se ejecuta un programa

## Ejemplo 7

### ► ¿Son correctas las siguientes afirmaciones?

Un programa escrito en lenguaje máquina/ensamblador es más eficiente que un programa escrito en un lenguaje de alto nivel como C

Un programa siempre ejecutará más rápido cuanto más cores/núcleos tenga el procesador

# Ejemplo 8

¿Funciona correctamente este programa?

```
public class Stack {
    private          Object[] elements;
    private          int size = 0;
    private static final int DEFAULT_INITIAL_CAPACITY = 16;

    public Stack() {
        elements = new Object[DEFAULT_INITIAL_CAPACITY];
    }

    public void push(Object e) {
        ensureCapacity();
        elements[size++] = e;
    }

    public Object pop() {
        if (size == 0)
            throw new EmptyStackException();
        return elements[--size];
    }

    private void ensureCapacity() {
        if (elements.length == size)
            elements = Arrays.copyOf(elements, 2 * size + 1);
    }
}
```

# Ejemplo 8

¿Funciona correctamente este programa?

```
public class Stack {
    private          Object[] elements;
    private          int size = 0;
    private static final int DEFAULT_INITIAL_CAPACITY = 16;

    public Stack() {
        elements = new Object[DEFAULT_INITIAL_CAPACITY];
    }

    public void push(Object e) {
        ensureCapacity();
        elements[size++] = e;
    }

    public Object pop() {
        if (size == 0)
            throw new EmptyStackException();
        return elements[--size];
    }

    private void ensureCapacity() {
        if (elements.length == size)
            elements = Arrays.copyOf(elements, 2 * size + 1);
    }
}
```

Memory Leaks

# Ejemplo 8

¿Funciona correctamente este programa?

```
public class Stack {
    private          Object[] elements;
    private          int size = 0;
    private static final int DEFAULT_INITIAL_CAPACITY = 16;

    public Stack() {
        elements = new Object[DEFAULT_INITIAL_CAPACITY];
    }

    public void push(Object e) {
        ensureCapacity();
        elements[size++] = e;
    }

    public Object pop() {
        if (size == 0)
            throw new EmptyStackException();
        return elements[--size];
        elements[size] = null; // Eliminate obsolete reference
    }

    private void ensureCapacity() {
        if (elements.length == size)
            elements = Arrays.copyOf(elements, 2 * size + 1);
    }
}
```



# Ejemplo 9

- ¿Qué procesador es más rápido?



**Intel Atom x7-Z8700**

1.60 GHz

2.40 GHz

2.40 GHz

4

No

No

2 MB

—VS—



**Intel Core i3-4020Y**

1.50 GHz

Frequency

Turbo (1 Core)

Turbo (All Cores)

Cores

Hyperthreading

Overclocking ?

Cache

2

Yes

No

3 MB

# Temario

Tema 1. Introducción a los computadores

Tema 2. Representación de la información

Tema 3. Fundamentos de la programación en ensamblador

Tema 4. El procesador

Tema 5. Sistemas de memoria

Tema 6. Sistemas de Entrada/salida

# Desarrollo del curso

- ▶ 14 semanas
  - ▶ 1 sesión por semana de 100 minutos en grupo magistral
  - ▶ 1 sesión por semana de 100 minutos en grupo reducido
    - ▶ 4 de estas sesiones son de laboratorio
  - ▶ 1 sesión adicional de 100 minutos en la última semana

# Desarrollo del curso

- ▶ 14 semanas
  - ▶ 1 sesión por semana de 100 minutos en grupo magistral
  - ▶ 1 sesión por semana de 100 minutos en grupo reducido
    - ▶ 4 de estas sesiones son de laboratorio
  - ▶ 1 sesión adicional de 100 minutos en la última semana

100 m	100 m	4,5h (estudio personal)
-------	-------	-------------------------



dedicación media semanal (8 horas)

# Evaluación

60 % mediante evaluación continua

30% Prácticas

30% Pequeños exámenes

40 % mediante un examen final

# Evaluación continua

## ► **Prácticas: 30%**

- Se realizarán **DOS** prácticas obligatorias
  - Nota mínima de cada práctica: **2**
  - Nota mínima media de todas las prácticas: **4**
- Pesos de cada práctica:
  - Práctica 1: 15%
  - Práctica 2: 15%
- Se realizarán en grupos de **dos** alumnos
- **En caso de que se detecte copia de prácticas, a ambas partes implicadas (copiados y copiadores) se les calificará con un 0**

# Evaluación continua

- ▶ **Pequeños exámenes en cada grupo reducido: 30%**
  - ▶ Se realizarán **4 exámenes pequeños** (15-20 minutos)
  - ▶ En el que se evaluarán **todos** los conocimientos adquiridos por el alumno hasta ese momento
  - ▶ Se hará la media de las **3** mejores notas
- ▶ No se repetirá ningún examen.
- ▶ Salvo causa médica justificada con suficiente antelación, no se admitirá que un alumno realice el examen en un grupo distinto al que está matriculado.

# Evaluación continua

- ▶ Un alumno **sigue el proceso de evaluación continua** cuando:
  - ▶ Entrega **todas** las prácticas
    - ▶ Nota mínima de cada práctica: **2**
    - ▶ Nota mínima de todas las prácticas: **4**



# Examen final de la convocatoria ordinaria

- ▶ El **examen final** incluirá **todo** el contenido de la asignatura (tanto teórico como práctico).
  - ▶ **Nota mínima: 4**
- ▶ Para aquellos alumnos que hayan seguido el proceso de evaluación continua, el valor de este examen supondrá el 40% de la nota final. El otro 60% corresponderá a la nota obtenida en la evaluación continua.
- ▶ Para aquellos alumnos que no hayan completado el proceso de evaluación continua, el examen final tendrá un valor del 60% de la calificación total de la asignatura. Por tanto, para poder aprobar, el alumno deberá obtener una calificación superior a 8.33 sobre 10 en este examen

# Convocatoria extraordinaria

El **examen final** incluirá **todo** el contenido de la asignatura (tanto teórico como práctico).

❑ **Nota mínima: 4**

Si el estudiante siguió el proceso de evaluación continua, el examen final de esta convocatoria tendrá un peso del 40% y la calificación final tendrá en cuenta el otro 60% obtenido en el proceso de evaluación continua.

Si el estudiante no completó el proceso de evaluación continua el examen de esta convocatoria tendrá un valor del 100% de la calificación final de la asignatura.

Aunque el estudiante hubiera seguido el proceso de evaluación continua, tendrá derecho a ser calificado en la forma indicada en el apartado anterior cuando le resulte más favorable.

# Calificación como NO PRESENTADO

- ▶ Un estudiante aparecerá como **no presentado** cuando no realice el correspondiente examen final.
  - ▶ Aunque ya se encuentre aprobado por evaluación continua

# Nota final

- ▶ La nota final se incrementará en **1 punto** a aquellos alumnos que realicen:
  - ▶ **Todas** las pruebas de evaluación continua.
    - ▶ Todos los exámenes pequeños (**los 4**)
    - ▶ Todas las prácticas
  - ▶ Obtengan más de un 7 sobre 10 de calificación en la evaluación continua y al menos 4 puntos en el examen final.

# Importancia de la evaluación continua

	<b>2014- 2015</b>	<b>2015- 2016</b>	<b>2016- 2017</b>	<b>2017- 2018</b>	<b>2018- 2019</b>
Alumnos que siguen la evaluación continua	78%	74%	74%	71%	78%
Alumnos que aprueban la evaluación continua	74%	64%	67%	66%	76%
Alumnos que aprueban la evaluación continua respecto de los que la siguen	87%	87%	90%	89%	89%
Alumnos que aprobaron al final la asignatura aunque abandonaron la evaluación continua	< 1 %	< 1 %	< 1 %	< 1 %	<1%
Alumnos que aprobaron la evaluación continua y han aprobado la asignatura al final	92%	89%	85%	86%	94%
Alumnos que completaron y suspendieron la evaluación continua han aprobado la asignatura al final	6%	7%	4%	6%	2%
<b>Alumnos aprobados</b>	<b>67%</b>	<b>66%</b>	<b>65%</b>	<b>62%</b>	<b>78%</b>
<b>Alumnos no presentados</b>	<b>23%</b>	<b>20%</b>	<b>22%</b>	<b>26%</b>	<b>15%</b>
<b>Alumnos suspensos</b>	<b>10%</b>	<b>14%</b>	<b>13%</b>	<b>12%</b>	<b>7%</b>

# Bibliografía

- ▶ Problemas resueltos de Estructura de Computadores. 2ª edición  
F. García, J. Carretero, J. D. García  
D. Expósito  
2015



- ▶ Computer Organization and Design  
The Hardware/Software Interface  
D.A. Patterson, J. Hennessy  
Quinta edición  
2014



# Bibliografía

- ▶ Fundamentos de Sistemas Digitales  
Thomas L. Floyd  
Pearson 2016



- ▶ Computer Organization and Architecture  
Décima edición  
William Stallings  
Pearson 2016



# Materiales complementarios

- ▶ [Computer History Museum](#)
- ▶ [Museo histórico de la Informática, Universidad Politécnica de Madrid](#)
- ▶ [Museo virtual de la Informática. Universidad de Castilla-la Mancha](#)
- ▶ <https://www.computer.org/cms/Computer.org/Publications/timeline.pdf>
- ▶ [The EDSAC Simulator](#)
- ▶ [IBM Archives](#)
- ▶ [Charles Babbage Institute](#)



# ¿Por qué estudiar Estructura de Computadores? W. Stallings

El «IEEE/ACM Computer Curricula 2001» [JTF01], preparado por la *Joint Task Force* de currículo de computadores de la Sociedad de Computadores IEEE (*Institute of Electrical and Electronics Engineers*) y la ACM (*Association for Computing Machinery*), citan la arquitectura de computadores como uno de los temas troncales que debe estar en todos los currículos de todos los estudiantes de licenciatura e ingeniería informática. El informe dice lo siguiente:

«El computador está en el corazón de la informática. Sin él la mayoría de las asignaturas de informática serían hoy una rama de la matemática teórica. Para ser hoy un profesional en cualquier campo de la informática uno no debe ver al computador como una caja negra que ejecuta programas mágicamente. Todos los estudiantes de informática deben, en cierta medida, comprender y valorar los componentes funcionales de un computador, sus características, su funcionamiento y sus interacciones. También sus implicaciones prácticas. Los estudiantes necesitan comprender la arquitectura del computador para estructurar un programa de forma que este sea más eficiente en una máquina real. Seleccionando el sistema que se va a usar, debe ser capaz de comprender el compromiso entre varios componentes, como la velocidad del reloj de la CPU frente al tamaño de la memoria».

# ¿Por qué estudiar Estructura de Computadores? W. Stallings

En [CLEM00] se dan los siguientes ejemplos como razones para estudiar arquitectura de computadores:

1. Supóngase que un licenciado trabaja en la industria y se le pide seleccionar el computador con la mejor relación calidad precio para utilizarlo en una gran empresa. Comprender las implicaciones de gastar más en distintas alternativas, como una caché grande o una velocidad de reloj mayor, es esencial para tomar esta decisión.
2. Hay muchos procesadores que no forman parte de equipos PC o servidores, pero sí en sistemas embebidos. Un diseñador debe ser capaz de programar un procesador en C que esté embebido en algún sistema en tiempo real o sistema complejo, como un controlador electrónico de un coche inteligente. Depurar el sistema puede requerir utilizar un analizador lógico que muestre la relación entre las peticiones de interrupción de los sensores del sistema y el código máquina.
3. Los conceptos utilizados en arquitectura de computadores tienen aplicación en otros cursos. En particular, la forma en la que el computador ofrece un soporte arquitectural a los lenguajes de programación y funciones en principio propias del sistema operativo, refuerza los conceptos de estas áreas.