Arquitectura de Ordenadores

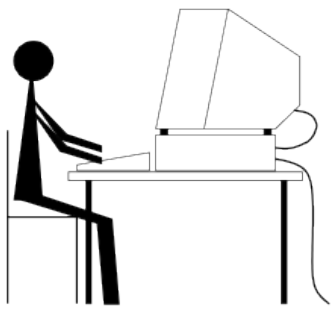
Programación en ensamblador

Abelardo Pardo
abel@it.uc3m.es

Universidad Carlos III de Madrid
Departamento de Ingeniería Telemática

Diferentes Perspectivas de un Ordenador

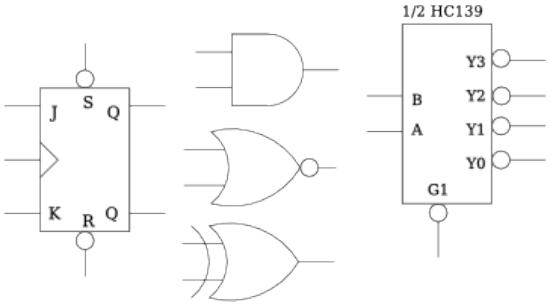
Perspectiva del Programador/Usuario



- Dispositivos de Entrada/Salida (Hardware).
  - Teclado.
  - Ratón.
  - Disco duro.
  - Conexión a Internet.
- Programas de Usuario (Software).
  - Navegador.
  - Lector de email.
  - Editor de texto.
  - Hoja de cálculo.
- Programas de Desarrollo.
  - Compilador de Java.
  - Ejecución de Programas en Java.

Diferentes Perspectivas de un Ordenador

Perspectiva del Diseñador



- Lógica Booleana.
- Circuitos Combinacionales.
  - Transistores.
  - Puertas lógicas.
- Circuitos Aritméticos.
  - Sumadores.
  - Multiplicadores.
- Circuitos Secuenciales.
- Autómatas Finitos.

Motivación

¿Por qué debo aprender a programar en lenguaje ensamblador del Pentium?

- Requiere un conjunto de técnicas propias de cualquier ingeniero.
- Entender el funcionamiento de un procesador facilita la comprensión de un **sistema completo**.
- Cada vez se diseñan más sistemas que incluyen un **procesador**.
- Es posible que alguien participe en el equipo de diseño de un **procesador**.
- Comprender del funcionamiento de un procesador permite desarrollar aplicaciones **eficientes** y **robustas**.
- Es el **único lenguaje** que entiende el hardware.

¿Qué es un programa?

- Un programa es un texto con cierta estructura que se escribe mediante un editor en un fichero.

```
public class Hello {
  public static void main(String[] args) {
    System.out.println("Hola mundo");
  }
}
```

- Se describe una **secuencia de pasos o instrucciones a realizar**.
- Estas instrucciones o pasos **manipulan un conjunto de datos**.
- Los datos a manipular **también necesitan ser definidos**.
- Una vez escrito el programa, este se **ejecuta** para obtener el resultado.

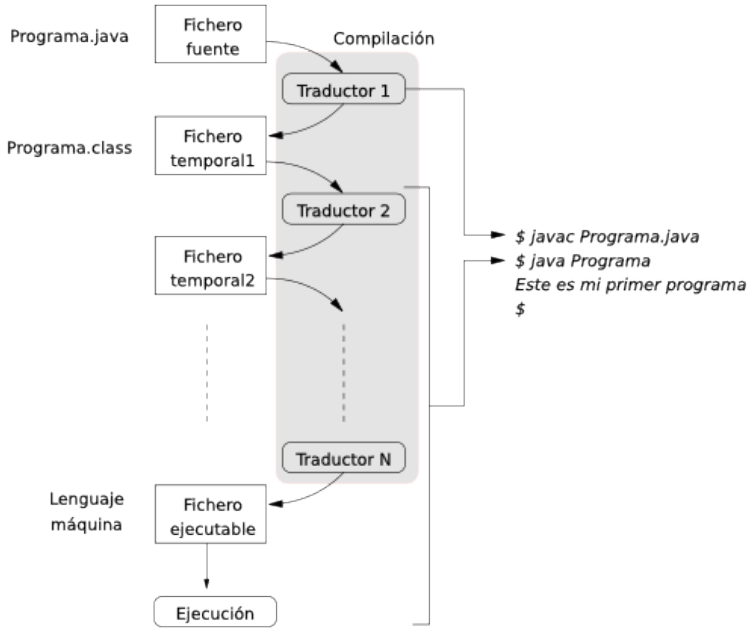
Un Programa en Ensamblador


```
        .data                   # Sección de datos
msg:    .asciz "Hola mundo\n" # Mensaje a imprimir

        .text                   # Sección de código
        .globl main             # main es un símbolo global
main:
        push $msg
        call printf             # printf es una función ya escrita que
                                # escribe por pantalla un argumento dado
        add $4, %esp
        ret
```

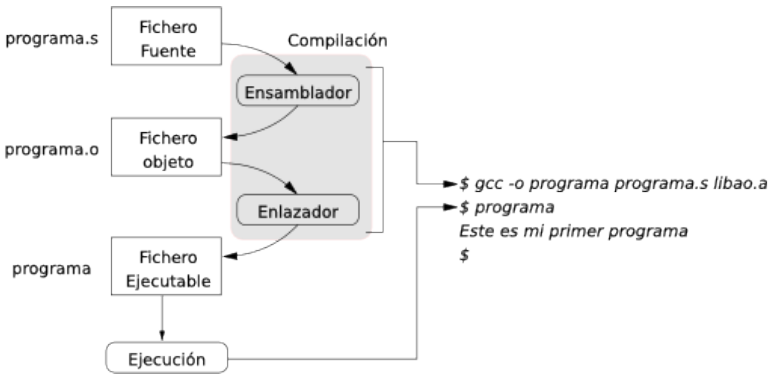
- El fichero tiene dos secciones: **datos y código**.
- La sección de datos comienza a partir de la palabra `.data`.
- En esta sección se define el **mensaje a imprimir**.
- La sección de código comienza a partir de la palabra `.text`.
- El código se organiza en **instrucciones**.
- El lenguaje máquina (o lenguaje ensamblador) es **menos intuitivo**.
- ¿Qué hace la instrucción `add $4, %esp`?

Ejecución de un programa en Java




```
unix> javac Hello.java
unix> java Hello
Hola mundo
unix>
```

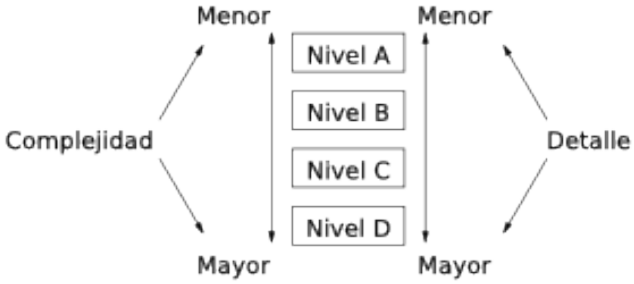

Ejecución de un programa en ensamblador



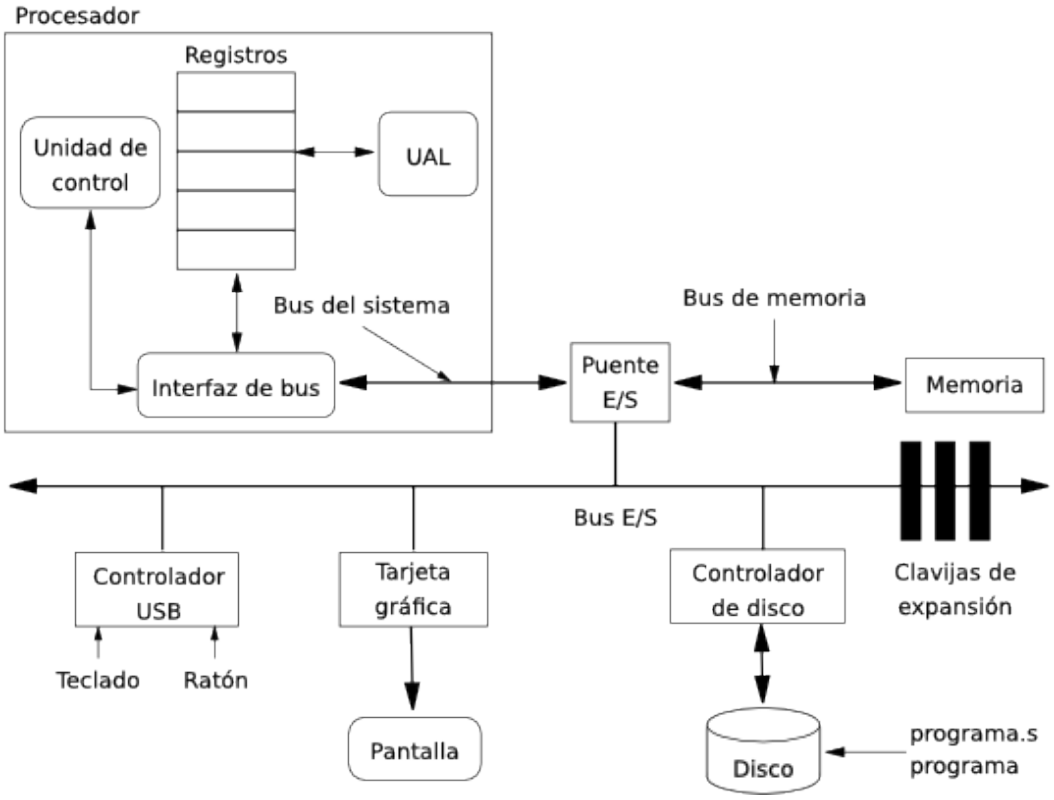

```
unix> hello
Hola mundo
unix>
```


¿Cómo se puede manipular el nivel de complicación que existe en un sistema?

- La complejidad se estructura en **Niveles de Abstracción**.
- Cada nivel sólo ve los aspectos proporcionados por el nivel inferior.
- Cada nivel simplifica la complejidad al nivel superior.



Ejemplo: Los mapas a escalas diferentes de un determinado lugar.



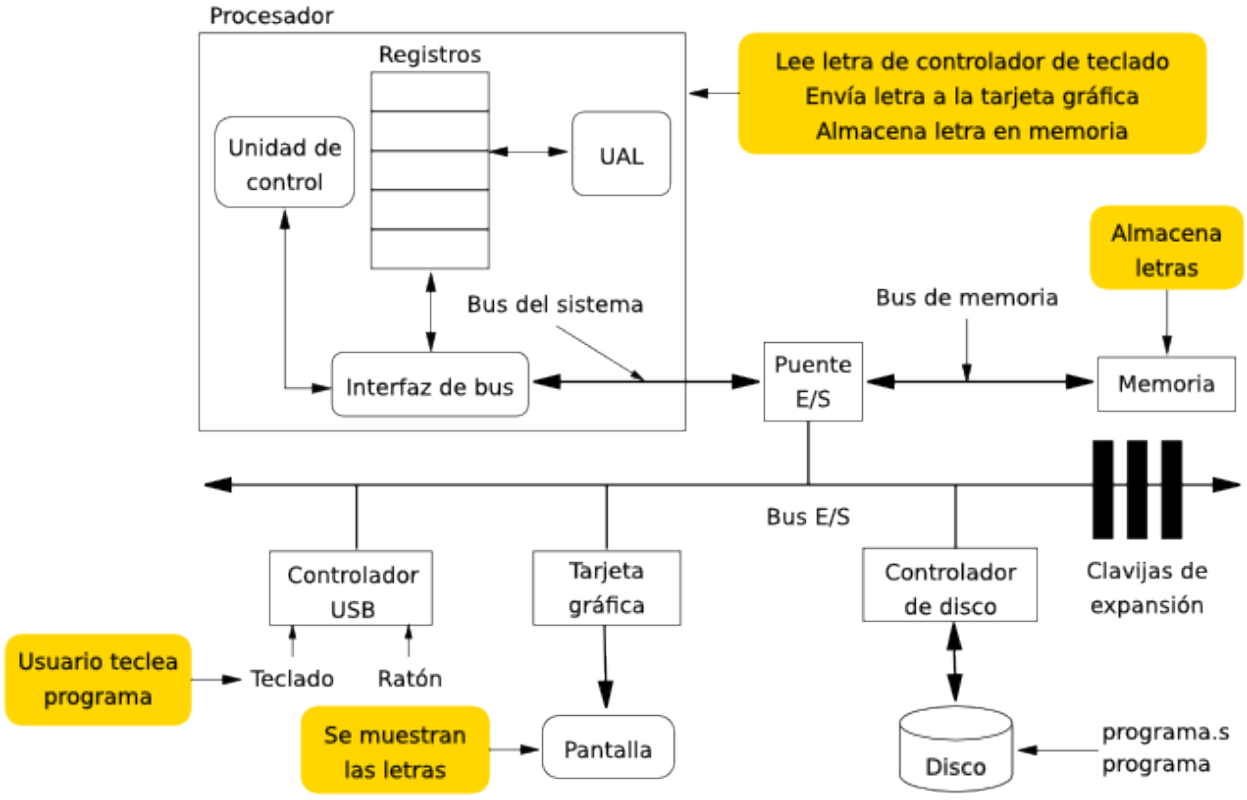
Ejecución de un Programa por el Procesador

- Al escribir el nombre del programa y pulsar “return”, el procesador comienza a ejecutar el programa escrito.

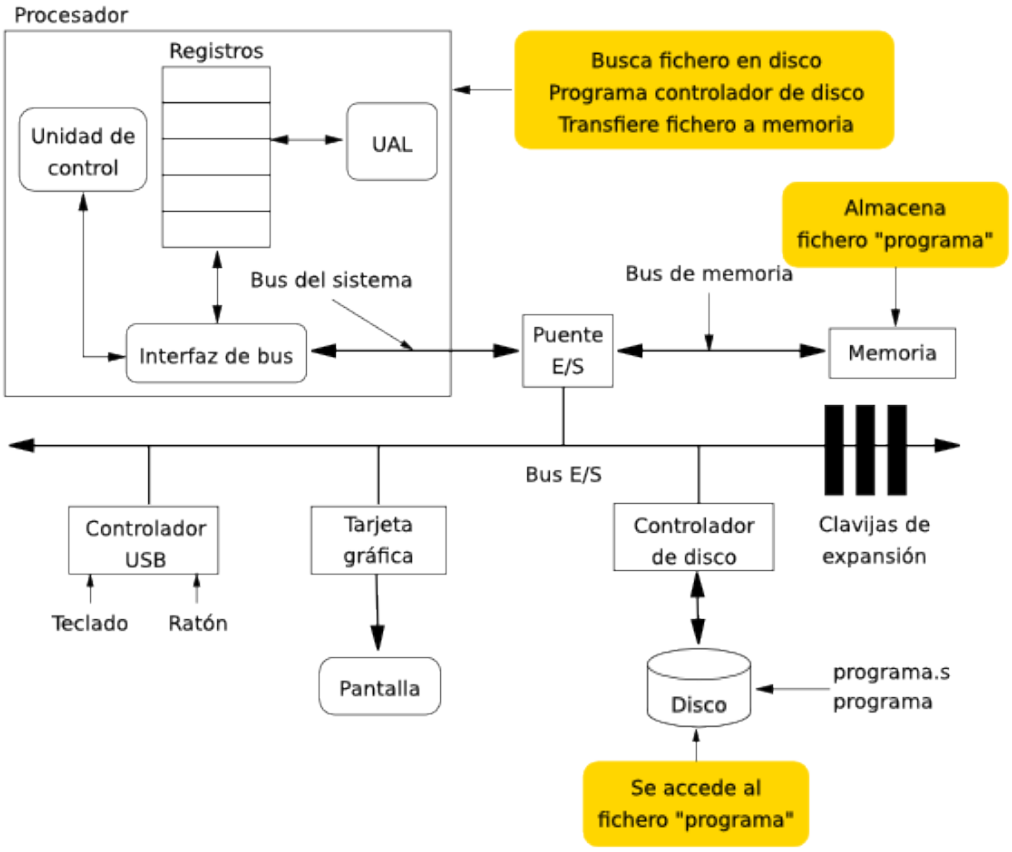
- Pasos para la ejecución de un programa:

1. Usuario teclea el nombre del ejecutable.
2. Se obtiene el programa del fichero ejecutable.
3. El programa se almacena en la memoria.
4. El procesador ejecuta una tras otra las instrucciones del programa.
5. Se borra el contenido del programa de memoria.

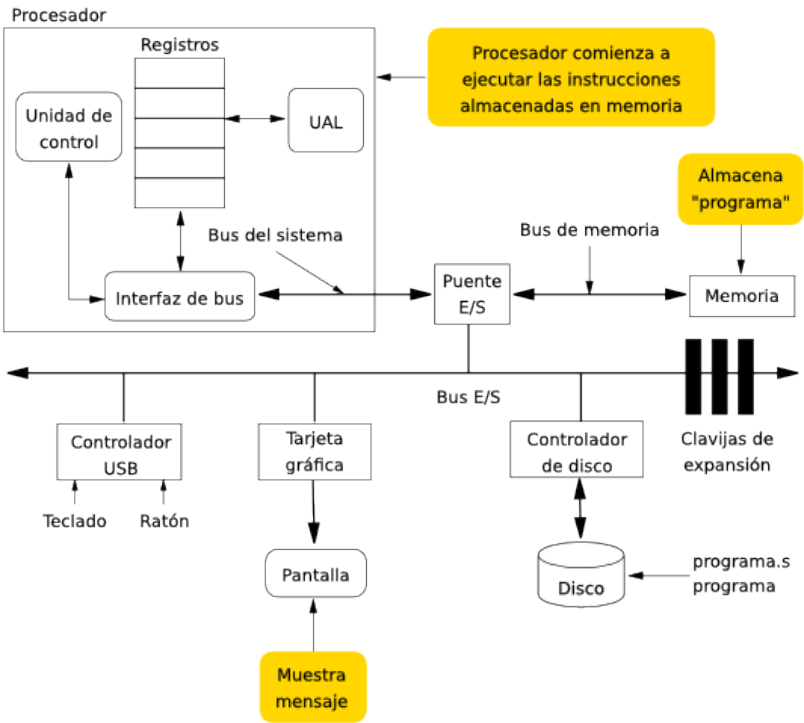
El Usuario teclea el nombre del ejecutable



El programa se transfiere de disco a memoria



El Procesador Ejecuta las Instrucciones



Codificación de un Programa

- Un programa manipula datos mediante **instrucciones máquina**.
- Tanto datos como instrucciones deben estar **codificados**.
- El procesador manipula **exclusivamente** información binaria.
- El conjunto de operaciones que el procesador es capaz de realizar es **fijo y finito**.
- ¿Cómo se pueden codificar **números enteros, reales, símbolos, letras e instrucciones** utilizando **únicamente** el código binario?

La Memoria

- ¿Qué es la **memoria** de un ordenador?
- ¿Qué tipo de **datos** se pueden almacenar?
- ¿Qué **tamaño** tiene?
- ¿Qué **operaciones** se realizan sobre la memoria?
- ¿Cómo está **organizada**?

- Estudio de las partes **internas** del procesador:
  - Nombre y número de **registros**.
  - **Propósito** de los diferentes registros.
  - Gestión de la **pila**.
  - Contador de programa, coma flotante, MMX.

- ¿Qué instrucciones es capaz de ejecutar el **Pentium**?
- ¿Cómo se especifican los **operandos** de una instrucción?
- **Ciclo de ejecución**: Pasos detallados en la ejecución de una instrucción máquina.
- Diferentes formas de obtener **operandos** para una instrucción.

Programas Complejos en Lenguaje Ensamblador

- ¿Cómo construir **programas complejos** mediante instrucciones máquina?
- Estructuras **Condicionales**.
- Iteración mediante **bucles**.
- **Comparaciones**.
- **Subrutinas**.