Tema 1: Introducción a la Informática

Contenidos

1 ¿Qué es un ordenador?

Un ordenador es un dispositivo electrónico que puede interpretar y ejecutar comandos programados para operaciones de entrada, salida y computación.

Un sistema basado en PC (Personal Computer) tiene cuatro tipos de componentes básicos (ver Figura 1):

5. Entrada
    - Teclado
    - Ratón
    - Micrófono
6. Salida
    - Monitor
    - Altavoces
    - Impresora
7. Procesamiento
    - Microprocesador
8. Almacenamiento
    - Temporal: Memoria RAM (Random Access Memory)
    - Permanente: disco duro, CD, DVD...

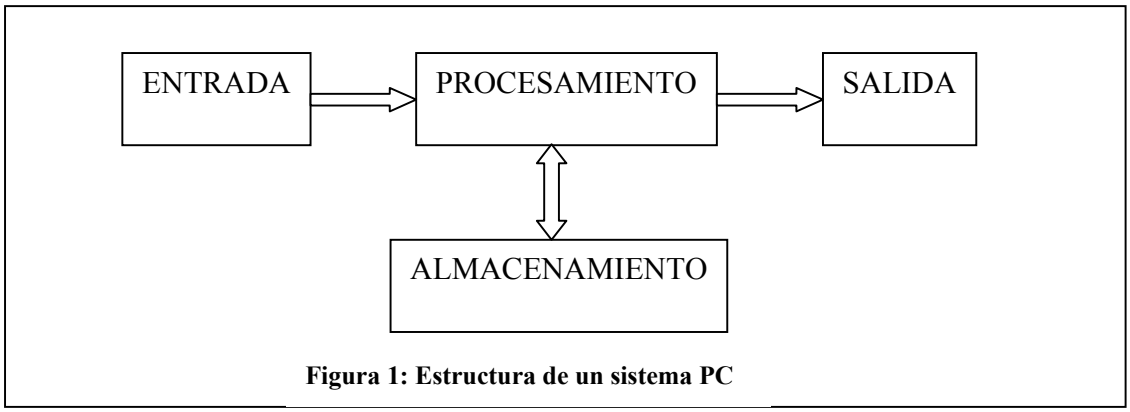

Figura 1: Estructura de un sistema PC

1.1 Elementos del PC

Los elementos principales del PC son el procesador y la memoria (ver Figura 2).

Procesador

El procesador, también llamado CPU (Central Processing Unit) es el núcleo de un ordenador. Tiene dos partes principales: La unidad de control (CU: Control Unit) y la unidad aritmético-lógica (ALU: Arithmetic-Logic Unit). El procesador trabaja en contacto directo con la memoria.

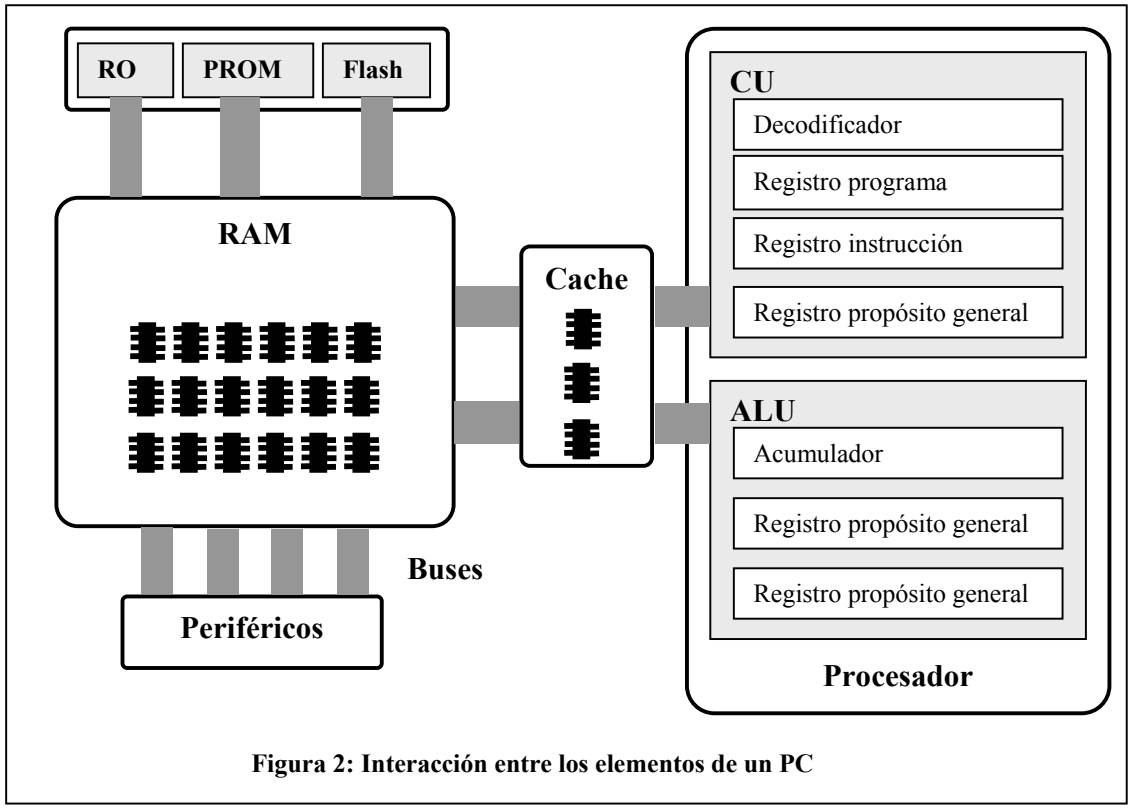

Figura 2: Interacción entre los elementos de un PC

Unidad de control (CU)
Funciones:
- leer e interpretar instrucciones de programa
- dirigir la operación de los componentes de procesado interno
- controlar el flujo de programa y los datos de entrada y salida en la RAM

Los programas se encuentran normalmente almacenados en un fichero en la memoria permanente (disco duro u otros discos externos). Cuando se arranca el programa, lo primero que se hace es pasarlo de la memoria permanente a la memoria RAM.

Al ejecutar un programa (secuencia de instrucciones), la primera de esas instrucciones se mueve de la memoria RAM a la unidad de control, donde se decodifica en el decodificador. La unidad de control utiliza además registros (de acceso muy rápido) para almacenar información y facilitar el intercambio de datos entre la memoria RAM y el procesador. El registro de instrucción contiene la instrucción que está siendo ejecutada. El registro de programa contiene la dirección en la RAM de la próxima instrucción a ser ejecutada. Los registros de propósito general almacenan datos para su procesamiento inmediato.

Unidad aritmético-lógica (ALU)
Funciones:
- computaciones aritméticas (suma, resta, multiplicación y división)
- computaciones lógicas (comparaciones)

Los resultados se almacenan en el acumulador.

RAM

La RAM (Random Access Memory) es una memoria de lectura y escritura. Los chips de la RAM tienen puntos cargados electrónicamente que representan un bit (unidad mínima de información, de valor 1 ó 0). Los bits se agrupan en bytes (conjunto de 8 bits). La información en la RAM se distribuye en direcciones: cada instrucción o dato de un programa está en una dirección.

La memoria se accede mediante buses: el bus de direcciones (selecciona la posición de la que se va a leer/escribir) y el bus de datos (datos leídos/escritos).

La memoria RAM es volátil: cuando se quita la corriente eléctrica, se pierden los datos. Los programas y los datos se transfieren a la RAM desde dispositivos de entrada o desde el disco duro. Cuando se deja de usar un programa, el espacio en memoria se asigna a otro.

Otro tipo de memorias
Memoria caché: más rápida que la RAM. Guarda datos o instrucciones recientemente accedidos (es probable que se vuelvan a acceder).
Memoria ROM (Read-Only Memory): Memoria de sólo lectura. Suele utilizarse en el arranque del ordenador.

1.2 Ciclo de ejecución de una instrucción

El procesador interactúa con la memoria para ejecutar las instrucciones de los programas. El ciclo de ejecución de una instrucción se divide en dos fases principales:

1. Fase de instrucción
   - Captura de la instrucción desde la RAM al registro de instrucción de la CU
   - Decodificación e interpretación de la instrucción
2. Fase de ejecución
   - Ejecución de la instrucción (normalmente usando la ALU)
   - Almacenamiento de resultados en memoria

2 ¿Qué es un sistema operativo?

Un Sistema Operativo (SO) es un programa que gestiona el hardware de un ordenador, proporcionando la base para los programas de aplicación y sirviendo como intermediario entre el usuario del ordenador y el hardware del ordenador.

- El hardware (CPU, memoria y dispositivos de entrada/salida (I/O: input/output) proporciona los recursos básicos de computación.
- Los programas de aplicación (procesadores de texto, hojas de cálculo, compiladores, correo electrónico, navegadores, etc.) definen la forma en la que estos recursos se utilizan para resolver los problemas de computación de los usuarios.
- El sistema operativo controla y coordina el uso del hardware entre las distintas aplicaciones de los distintos usuarios.

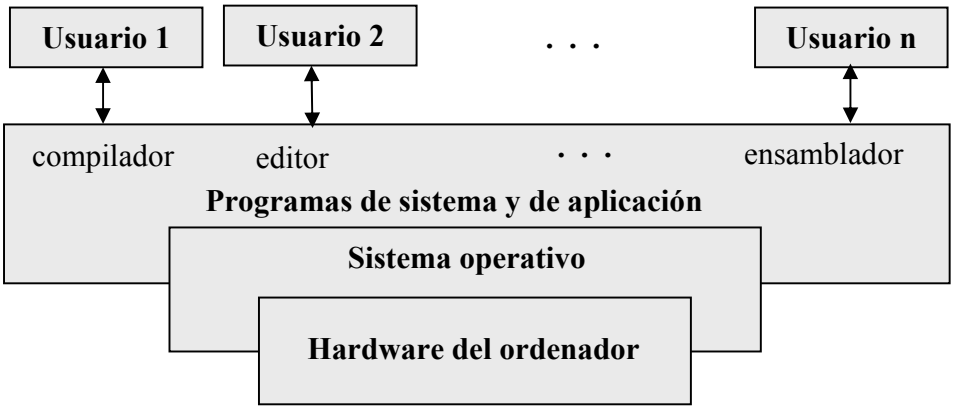

Figura 3: Visión abstracta de los componentes de un sistema de computación

Los dos objetivos principales de un SO son:

- eficiencia en el uso de recursos
- facilidad de uso para el usuario

2.1 Tareas principales de un sistema operativo

- Gestión de los programas: planificación de los procesos, carga, inicialización y supervisión de su ejecución
- Gestión de la memoria
- Gestión de ficheros
- Gestión de las operaciones de entrada/salida
- Interfaz con el usuario

3 ¿Cómo se representa la información?

En un ordenador, la información se representa de forma digital, también llamada binaria. El Sistema Binario se fundamenta en la aritmética en base 2.

- Un bit es la unidad mínima de información en el Sistema Binario y puede tomar los valores 0 ó 1.
- Un byte es un conjunto ordenado de 8 bits.
- Una palabra es el número de bytes que pueden ser manejados por la unidad de control (CU) del ordenador de una sola vez.
- Un KByte (kilobyte) = 1024 bytes
- Un MByte (megabyte) = 1024 KBytes
- Un GByte (gigabyte) = 1024 MBytes

Ejemplo de representación de número decimales en binario:

$52 = 110100 = 1*2^5+1*2^4+0*2^3+1*2^2+0*2^1+0*2^0$

4 Lenguajes de programación

4.1 Lenguaje de programación

Conjunto de normas «lingüísticas» que permiten escribir un programa y que éste sea entendido por el ordenador y pueda ser trasladado a ordenadores similares para su funcionamiento en otros sistemas.

- alfabeto: qué elementos léxicos están permitidos
- sintaxis: cómo se construyen las frases
- semántica: qué significan las frases

4.2 Programa, datos y algoritmos

Un *programa* es un conjunto de instrucciones escritas en un lenguaje de programación susceptibles de ser ejecutadas por ordenador para realizar una determinada función.
Un programa es un conjunto de datos y de algoritmos.

Los *datos* son estructuras algebraicas que consisten, en general, en un conjunto de valores, con sus propiedades y con las operaciones que soportan.

Un algoritmo es una forma de describir una solución a un problema. Se representa mediante una serie *ordenada* y finita de instrucciones elementales que trabaja sobre os datos, posiblemente modificándolos.

4.3 Evolución de los lenguajes de programación

Lenguajes de Bajo Nivel
Lenguaje máquina: En un principio, el lenguaje de programación estaba directamente orientado al tipo de instrucciones que el procesador del ordenador era capaz de entener y ejecutar (escrito en binario!).
Lenguaje ensamblador: versión "humanizada" del lenguaje máquina que utiliza nombres de instrucciones (ADD, LOAD, etc). Todavía muy cercano a la ejecución de la máquina.

Lenguajes de Alto Nivel
Abstraen el problema a resolver de la máquina y procesador concretos que lo van a ejecutar el programa.
Ejemplos: C, Pascal, C++, Java, etc.

4.4 Paradigmas de programación

Todos los lenguajes de programación proporcionan abstracciones. La complejidad de los problemas que se pueden resolver depende del tipo y la calidad de la abstracción.

El programador debe establecer una asociación entre el "espacio de la solución" (el lugar donde se modela el problema, en este caso el ordenador) y el "espacio del problema" (por ejemplo, si se quiere programar una aplicación bancaria, los elementos que conforman el entorno bancario)

Por ejemplo, el lenguaje ensamblador es una abstracción de la máquina subyacente.
Un paradigma de programación es un modelo básico de diseño y desarrollo de programas. Según el tipo de abstracción que se realice, tendremos los distintos tipos de *paradigmas de programación*.

Programación imperativa
Un programa en un lenguaje imperativo aparece como una lista de instrucciones u órdenes elementales que han de ejecutarse una tras otra, en el orden en que aparecen en el programa. Las instrucciones de un programa imperativo utilizan datos almacenados en la memoria del computador, llamados variables. Para realizar algún cálculo, se parte de ciertos datos almacenados y se realizan diversas operaciones (instrucciones); al final, el resultado está almacenado en alguna celda de memoria.

Los elementos principales de la programación imperativa son:

- *Concepto de variable*. El componente principal es la memoria, compuesto por un gran número de celdas donde se almacenan los datos y que son referenciadas por medio de su nombre (variable). El conjunto de valores de todas las variables del programa en un momento dado representa el estado del programa.
- *Operaciones de asignación*. Cada valor calculado debe ser asignado a la variable mediante operaciones de asignación. De esta forma se modifica el estado del programa.
- *Repetición*. Un programa imperativo, normalmente realiza su tarea ejecutando repetidamente una secuencia de pasos elementales.

El paradigma imperativo es una abstracción del lenguaje ensamblador. En los términos mencionados anteriormente, el modelado se realiza más cerca del "espacio de la solución" que del "espacio del problema".

Algunos de los lenguajes de programación que siguen el paradigma imperativo son BASIC, Pascal, Modula y C.

Ejemplo: programa en C para conversión de temperaturas.

```
#include <stdio.h>
  /* imprime la tabla Fahrenheit-Celsius
     para fahr=0,20,...,3000*/
main ()
{
  int fahr, Celsius;
  int lower, upper, step;

  lower = 0;
  upper = 300;
  step = 20;

  fahr = lower;
  while (fahr <= upper){
    Celsius = 5*(fahr-32)/9;
    printf("%d\t%d\n", fahr, Celsius);
    fahr = fahr + step;
  }
}
```

Programación funcional

El paradigma funcional se basa en el concepto matemático de función. Una función es una regla de correspondencia que asocia a cada elemento de un conjunto origen un elemento de un conjunto destino. Los conjuntos origen y destino suelen llamarse, respectivamente, dominio y rango de la función. Otro elemento básico es la composición funcional (el resultado de un cálculo es el argumento del siguiente).

Los elementos principales de la programación imperativa son:

- Tipos de datos
- Expresiones condicionales
- Recursión

El paradigma funcional, en lugar de abstraer la máquina subyacente, trata de abstraer el problema a solucionar, modelándolo como una composición de funciones.

En el caso de los programas funcionales, se modela el problema que se quiere resolver, y no una abstracción de la máquina que lo va a resolver. Es decir, estamos más cercanos al "espacio del problema" que al "espacio de la solución". Este paradigma tiene una visión particular del problema a resolver basada en funciones.

Algunos ejemplos de lenguajes de programación funcional son LISP, Haskell, Gofer, etc.

Ejemplo: implementación del algoritmo quicksort en Haskell

```
qsort []     = []
qsort (x:xs) = qsort elts_lt_x ++ [x] ++ qsort elts_greq_x
                 where
                   elts_lt_x   = [y | y <- xs, y < x]
                   elts_greq_x = [y | y <- xs, y >= x]
```

Programación orientada a objetos

La programación orientada a objetos (POO) permite representar directamente en el programa los elementos del espacio del problema. Los elementos del espacio del problema y su representación en el espacio de solución se denominan *objetos*. La idea principal es que le código que describe la solución se escribe en los términos del problema.

Cada objeto tiene un estado, representado por unos atributos (parecido a las variables) y unas operaciones que puede realizar.

Las principales características de la programación orientada a objetos son las siguientes:

1. *Todo es un objeto.* Cualquier elemento conceptual de problema (una persona, una cuenta bancaria, un circuito electrónico, un servicio...) puede representarse como un objeto en el programa.
2. *Un programa es un conjunto de objetos que se hacen peticiones los unos a los otros mandándose mensajes*. Mandarse un mensaje es equivalente a hacer una llamada a una de las operaciones ofrecidas por el objeto.
3. *Cada objeto tiene su propia memoria compuesta de otros objetos*. Esos otros objetos son los atributos del objeto (por ejemplo, un coche tiene una matrícula). De esa forma se pueden crear agrupaciones complejas que resulten sencillas de manejar a través del objeto.
4. *Cada objeto es de una clase determinada*. Una clase es como un tipo. Los objetos son instancias de la clase. La clase determina cómo está compuesto el objeto (qué atributos tiene) y cómo se comporta (qué mensajes se le pueden mandar, es decir, qué operaciones ofrece).
5. *Todos los objetos de una clase pueden recibir el mismo tipo de mensajes*.

Ejemplos de lenguajes orientados a objeto son Smalltalk, C++ y Java.

El próximo tema profundizará en los conceptos de orientación a objeto.

4.5 ¿Cómo se ejecuta un programa?

Los programas escritos en un lenguaje de alto nivel (código fuente) se tienen que traducir a un lenguaje ejecutable por el procesador de la máquina. Existen dos variantes principales:

- *Compilador*: traduce el código fuente a las instrucciones básicas del ordenador (código máquina), que luego se ejecutan
- *Intérprete*: toma una instrucción del código fuente, la traduce a código máquina, la ejecuta, y realiza lo mismo con las siguientes instrucciones

Java es un lenguaje que se caracteriza por ser independiente de la plataforma. Por plataforma se entiende el conjunto de hardware (ordenador con su procesador) y sistema operativo. Esto se consigue realizando una mezcla entre compilación e interpretación (ver Figura 4).

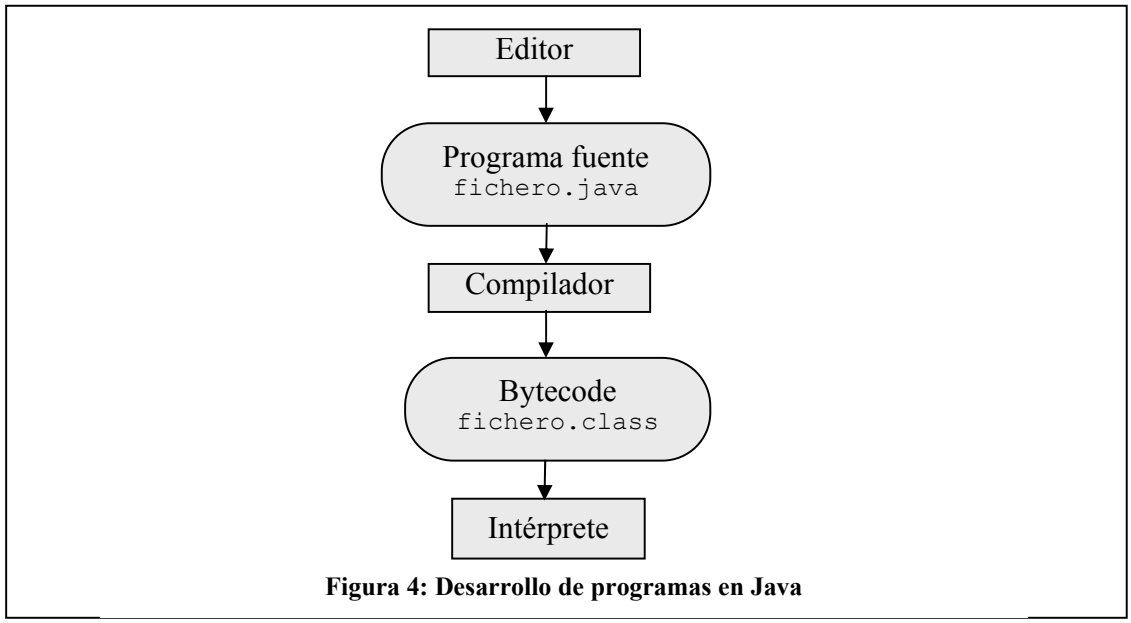

Figura 4: Desarrollo de programas en Java

Primero se compila el programa fuente a un lenguaje intermedio (`fichero.java` en la figura), llamado *bytecode* (códigos de byte) (`fichero.class` en la figura), que es parecido al lenguaje máquina, pero independiente de una máquina concreta.

Cada máquina sobre la que se ejecute Java va a tener una máquina virtual capaz de interpretar el bytecode. Por ello, el mismo fichero bytecode va a poder interpretarse / ejecutarse en distintas máquinas.