Tema 1: Introducción. Computadores y Programas

1 La Computadora y sus componentes

1.1 Computadoras

El término Informática proviene del francés como contracción de las palabras "information" y "automatique"; en inglés "Computer Science".

La **computadora digital** también llamada ordenador (del francés "*ordinateur*"), es un instrumento de gran importancia en la sociedad actual, constituyéndose en una herramienta habitual en todos los entornos de trabajo.

Históricamente, las computadoras se clasifican en tres grandes grupos:

- **Microcomputadoras**: Computadoras personales que poseen un microprocesador en su CPU.
- **Minicomputadoras**: Computadora de pequeña o mediana escala que funciona como una única estación de trabajo.
- **Macrocomputadoras**: Computadora de gran escala. Se trata normalmente de grandes ordenadores que proveen distintos servicios a una red.

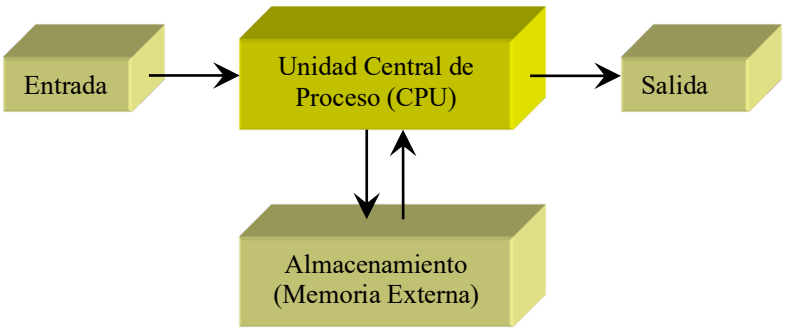
Actualmente, este término ha perdido relevancia. La evolución tecnológica y el abaratamiento de los costes de producción han llevado a que los tres tipos posean características de precio y rendimiento similares.

Sea cual sea su tipo, todas las computadoras poseen los siguientes componentes:

- **Dispositivos de entrada**. Se encargan de capturar información de entrada a la computadora: teclado, ratón, micrófono, etc.
- **Dispositivos de salida**: Se encargan de la salida de información de la computadora: monitor, impresora, altavoces, etc.
- **Dispositivos de almacenamiento**: Se encargan de almacenar los datos para su uso posterior: disco duro, disquete, CD-ROM, etc.
- **Unidad de Procesamiento**: Realiza las operaciones con los datos.

A los dispositivos de entrada, salida y almacenamiento se les denomina **Elementos Periféricos**. Por otro lado, la Unidad de Procesamiento junto a la Memoria Central, forman la **Unidad Central de Proceso** (CPU). La Memoria Central es donde

se almacenan temporalmente los datos con los que la Unidad de Procesamiento trabaja en cada momento.



1.2 Programas

Un programa es el conjunto ordenado de instrucciones que indican a la computadora las acciones que debe realizar para cumplir una tarea específica. En este sentido, con un programa especificamos a la computadora qué tiene que hacer, en el orden en el que lo tiene que hacer y los datos que ha de usar para ello.

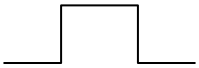
Los programas están escritos en un determinado lenguaje formal libre de ambigüedades, de forma que la especificación de lo que queremos hacer sea unívoca. Con respecto al nivel de abstracción de estos lenguajes, los podemos clasificar en dos grandes grupos:

- **Lenguajes Máquina**: Como su propio nombre indica, se trata de los lenguajes "nativos" de la computadora, y por ello es capaz de procesarlos directamente. Como veremos más adelante resultan sumamente costosos para el programador.
- **Lenguajes de Alto Nivel** (Pascal, C, C++, Java, etc.): Se trata de lenguajes mucho más cercanos al lenguaje humano, por lo que son más comprensibles y menos engorrosos de usar para el programador. Los programas construidos con estos lenguajes han de ser traducidos a Lenguaje Máquina mediante el uso de un compilador para que la computadora los pueda procesar.

2 Código Máquina y Código Ensamblador

2.1 Representación de la Información

La información se representa en una computadora mediante el uso de dos estados: ausencia y presencia de corriente. Esto se debe a la naturaleza eléctrica de la misma:

	Estado 0	Estado 1
Impulsos eléctricos		

Esta dualidad nos permite adoptar un sistema binario para representar la información. Así, la presencia de señal se representa por el 1 y la ausencia por el 0. Esto nos lleva al concepto de **bit**, que es la unidad mínima de información. Un bit puede adoptar los valores 0 ó 1.

Un único bit aporta poca cantidad de información. Es por ello que los bits se agrupan de 8 en 8 formando un **byte**. Un byte permite representar $2^8 = 256$ combinaciones diferentes.

A su vez, los bytes se agrupan formando **palabras**, que dependiendo de cada computadora pueden ser de 8, 16, 32 ó 64 bytes.

2.2 Programación de la Computadora

La computadora por sí misma no realiza ninguna tarea. Necesita de un conjunto de programas que le digan lo que tiene que hacer. A este conjunto de programas se le denomina **software** (parte inmaterial o modificable). En contrapartida, a la parte visible y material (circuitería, periféricos…) se le denomina **hardware**.

No todas las computadoras poseen el mismo hardware, por tanto, tampoco el mismo lenguaje nativo. Desde este punto de vista, entendemos por:

Instrucción: cadena de ceros y unos que permiten al procesador reconocer una operación elemental y realizarla.

Lenguaje de máquina: conjunto de instrucciones que puede ejecutar el procesador y las reglas para su codificación.

En este sentido, cada computadora posee un conjunto de instrucciones distinto. Por tanto, cada computadora únicamente podrá trabajar con el software escrito en su lenguaje máquina.

2.3 Lenguaje Máquina

Como hemos visto en apartados anteriores, el lenguaje máquina es el lenguaje nativo de cada computadora. Por tanto, dicho lenguaje está escrito en el sistema binario (ceros y unos). Aunque el objetivo de este curso es aprender el uso de lenguajes de alto nivel, veamos un ejemplo constructivo de cómo realizar un programa en este lenguaje para una computadora imaginaria:

Cada instrucción de la computadora consta de una palabra de 8 bits, en la que en los 3 primeros bits se especifica el código de operación y en los 5 siguientes la dirección del operando en la memoria.

Cada operación normalmente consta de dos operandos y un resultado y, como vemos, solamente podemos especificar uno. El segundo operando y el resultado de la operación están en un registro llamado *acumulador*.

El juego de instrucciones de la computadora es el siguiente:

Código	**Operación**	**Descripción**
001	Cargar	Copia en el acumulador la palabra cuya dirección de memoria se indica.
010	Sumar	Suma el operando con el acumulador guardando el resultado en el acumulador.
011	Restar	Resta el operando del acumulador y guarda el resultado en el acumulador.
100	Bifurcar	Si el contenido del acumulador es cero salta a la dirección que indica el operando.
101	Comparar	Si el contenido del acumulador es igual al operando iguala a cero el acumulador.
110	Almacenar	Copia el acumulador la dirección de memoria que indica el operando.
111	Parar	Detiene la ejecución del programa sea cual sea el operando.

Para realizar un programa en lenguaje máquina necesitamos realizar tres etapas:

- **Descomponer** el problema en operaciones elementales
- **Asignar** un lugar en la memoria central para cada instrucción y dato
- **Codificar** cada instrucción como una serie de dígitos binarios según la tabla de instrucciones, máquina del procesador.

El siguiente programa, en lenguaje máquina, resta un número de otro y el resultado lo almacena en una posición de memoria:

Memoria Central		
Dirección	**Instrucción**	**Contenido**
00000	00100101	Programa
00001	01100110	
00010	11000100	
00011	11100000	
00100	00000000	Datos
00101	00000111	
00110	00000010	

Como podemos imaginar, la programación en lenguaje máquina resulta excesivamente costosa e improductiva. Esto se debe a que la codificación de un algoritmo a este lenguaje resulta ilegible, lo que hace que la depuración de un error resulte altamente engorrosa.

2.4 Lenguaje ensamblador

Para salvar los problemas del lenguaje máquina surgió el lenguaje ensamblador. Este lenguaje, aun siendo dependiente de cada computadora, utiliza nemotécnicos para las operaciones y etiquetas para las direcciones de memoria con los datos. Esto hace que resulte mucho más sencillo que el lenguaje máquina. De cada instrucción en ensamblador existe una traducción directa al lenguaje máquina.

Veamos cómo quedaría el ejemplo anterior en lenguaje ensamblador. El conjunto de instrucciones del lenguaje sería:

Código	Abreviatura	Operación
001	CAR	Cargar
010	ALM	Almacenar
011	SUM	Sumar
100	RES	Restar
101	BC	Bifurcar si es cero
110	BNC	Bifurcar si no es cero
111	PAR	Parar

El programa anterior en ensamblador sería:

```
        CAR   A
        RES   B
        ALM   C
        STOP
C:      0
A:      7
B:      2
```

Como vemos, al utilizar nemotécnicos para las operaciones no es necesario recordar el código de cada una. Lo mismo ocurre con las etiquetas de las direcciones de memoria.

3 Lenguajes de Alto Nivel

3.1 Introducción

Como hemos visto, el lenguaje ensamblador constituye un gran avance en lo que a facilidad de comprensión se requiere a la hora de programar. De todas formas, este lenguaje sigue teniendo varios inconvenientes: Por un lado, cada computadora posee un lenguaje ensamblador distinto, por lo que para utilizar un programa en otra computadora distinta habría que rescribirlo. Por otro lado, el lenguaje sigue siendo demasiado cercano al lenguaje máquina, lo que dificulta su comprensión y por tanto encarece el desarrollo de programas de gran extensión.

Los lenguajes de alto nivel se caracterizan por no ser dependientes de máquina. Mediante la compilación (traducción) de un programa escrito en lenguaje de alto nivel podremos ejecutarlo en cualquier máquina. Para ello, necesitaremos un compilador de ese lenguaje para cada máquina en la que queramos ejecutar nuestro programa.

Otra característica de estos lenguajes es que la gramática de los mismos, pese a ser sumamente estricta, es más natural y comprensible, lo que hace que se puedan construir programas grandes e igualmente entendibles.

Por ejemplo, en el lenguaje COBOL, para realizar una resta de dos números únicamente se realiza una instrucción:

SUBSTRACT I MPUESTOS FROM SUELDO GIVING LIQUIDO

En este ejemplo hemos calculado el líquido eliminando los impuestos del sueldo bruto. Otra forma de expresarlo sobre el mismo lenguaje sería:

LIQUIDO = SUELDO-IMPUESTOS

Como vemos, los lenguajes de alto nivel poseen numerosas ventajas, entre las que podemos destacar:

- Son más productivos
- Más fáciles de aprender
- Autodocumentados. Las variables tienen un nombre que imponemos dependiendo del contexto.
- No es necesario conocer en profundidad la máquina
- Son independientes de máquina. Portabilidad

3.2 Clasificación de los lenguajes de programación

Existen cientos de lenguajes de programación, desarrollados a lo largo del tiempo para cubrir distintas necesidades. De la misma forma, existen distintos modelos de clasificación de los mismos, atendiendo a determinadas características.

El modelo de clasificación clásico consiste en diferenciar los lenguajes con respecto al **paradigma** que siguen. Podríamos definir el paradigma como la naturaleza o filosofía de programación que sigue el lenguaje. Distinguimos entre los siguientes paradigmas:

- **Imperativo** o procedimental (ej: C, Pascal, etc.)
- **Declarativo**, que a su vez se divide en funcional o aplicativo (ej: Lisp) y lógico (ej: Prolog)
- **Orientado a objetos** (ej: Java, C++, etc.)

Aparte de ésta, existen otras clasificaciones menos relevantes, como por ejemplo, con respecto al proceso de traducción. En este sentido, hablaremos de:

- **Lenguajes compilados**: A través de un programa denominado *compilador*, se traduce el programa completo a código máquina una

sola vez. A continuación se ejecuta dicho código independientemente del compilador.

- **Lenguajes interpretados**: Cada instrucción se traduce y ejecuta de forma individual a través de un programa denominado intérprete. Cada vez que ejecutemos el programa necesitaremos utilizar el intérprete.

A lo largo de este curso nos centraremos en el paradigma imperativo, y en concreto en el lenguaje C. Se trata de un lenguaje muy extendido, debido a su gran potencia y versatilidad. Es un lenguaje básicamente compilado, aunque existen determinadas versiones del mismo interpretadas.