CAPÍTULO 1

INTRODUCCIÓN Y CONCEPTOS GENERALES

El objetivo de este capítulo primero es introducir algunas palabras de uso habitual entre quienes se ven en la necesidad de programar: léxico común, de poca complejidad, pero que es necesario conocer bien para poder luego saber de qué se está hablando. Y presentar también muy sucintamente una descripción de la arquitectura de un ordenador, que nos permitirá comprender (de forma intuitiva: no se pretende ahora llegar más allá) cómo trabaja un ordenador con las instrucciones y datos que recibe al ejecutar un programa; y cómo logra un programador hacerse entender con una máquina que tiene un sistema de comunicación y de interpretación completamente distinto al humano.

Estructura funcional de las computadoras

Un esquema muy sencillo que representa la estructura básica de una

computadora queda recogido en la figura 1.1.

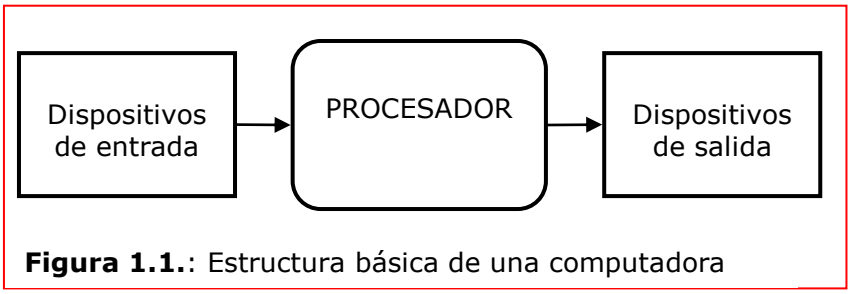

Figura 1.1.: Estructura básica de una computadora

El procesador recibe los datos desde los dispositivos de entrada (por ejemplo, teclado o ratón) y los muestra, o muestra los resultados del procesamiento, en los dispositivos de salida (por ejemplo pantalla o impresora). Los dispositivos de memoria (disco duro, lápiz óptico, disquete,…) son dispositivos de entrada y de salida de información: el procesador puede leer la información grabada en ellos y puede almacenar en ellos nueva información. A estos dispositivos de memoria les llamamos de memoria masiva.

La estructura básica del procesador queda esquematizada en la figura 1.2.

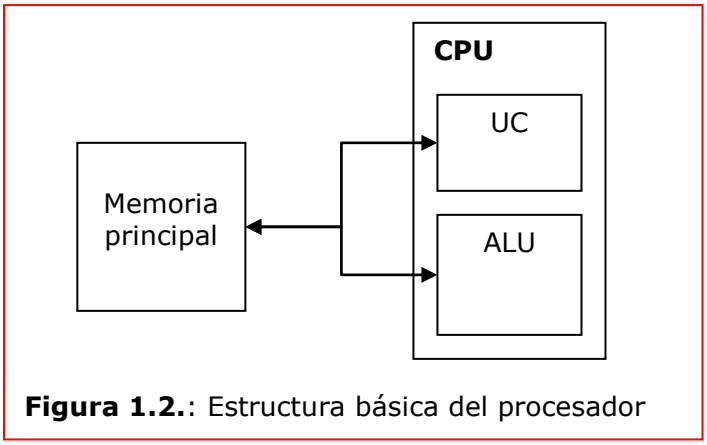

Figura 1.2.: Estructura básica del procesador

La **memoria principal** es el lugar donde se almacenan los datos a procesar o ya procesados y los resultados obtenidos. También se almacenan en ella el conjunto de instrucciones, o programas, que se

desean ejecutar para el procesamiento de la información. Todo programa que se ejecute debe estar almacenado (cargado) en esta memoria principal.

La memoria principal está construida mediante circuitos electrónicos muy sencillos que alcanzan dos estados estables posibles. Habitualmente se llaman a estos estados el estado cero y el estado uno. Es lo que se conoce como BIT (BInary digiT: dígito binario). El **bit** es la unidad básica de información. No sólo desde el punto de vista informático. Con un bit es posible decir verdadero o falso; sí o no.

Los circuitos electrónicos que forman la memoria son circuitos integrados, en los que se logra acumular una cantidad enorme de bits. Se podría hablar de la capacidad de memoria de una computadora, indicando el número de bits de qué dispone. Pero habitualmente no se hace así, sino que se agrupan los bits para formar bloques de mayor capacidad de información. El agrupamiento más conocido de bits es el llamado BYTE. Un **byte** es una agrupación de 8 bits. Con bits juntos y agrupados en una única unidad de información ya se pueden codificar muchos más que los dos valores posibles que alcanzábamos a codificar con un bit. En concreto, con un byte es posible obtener hasta $256 = 2^8$ combinaciones distintas: 00000000; 00000001; 00000010; 00000011; …; 11111101; 11111110; 11111111.

Podemos decir, por tanto, que toda la memoria de la computadora está dividida en bloques (bytes). Estos bloques se disponen ordenadamente, de forma que se puede hacer referencia a uno u otro bloque concreto dentro del circuito integrado de memoria. Se puede hablar, por tanto, de posiciones dentro de la memoria; cada posición está formada por un byte. Así es posible crear un índice de posiciones de memoria. Cada posición tiene su índice, al que llamamos **dirección de memoria**.

La capacidad de almacenamiento de una computadora (o de cualquier soporte de información) se mide por el número de bytes de que dispone. De forma habitual se toman múltiplos de byte para esa cuantificación.

Entendemos por…

Kilobyte: 2^{10} bytes, es decir, 1.024 bytes, que es un valor cercano a 10^3 bytes.

Megabyte: 2^{20} bytes, es decir 1.048.576 bytes, que es un valor cercano a 10^6 bytes.

Gigabyte: 2^{30} bytes, es decir 1.073.741.824 bytes, que es un valor cercano a 10^9 bytes.

Terabyte: 2^{40} bytes, es decir 1.099.511.627.776 bytes, que es un valor cercano a 10^{12} bytes.

Petabyte: 2^{50} bytes, es decir 1.125.899.906.842.624 bytes, que es un valor cercano a 10^{15} bytes.

La **memoria masiva**, distinta cualitativamente de la memoria principal y a la que antes nos hemos referido al presentar los dispositivos de entrada y salida de información, y también llamada memoria auxiliar, o secundaria, o externa, no goza de la misma velocidad que la principal, pero sí logra ofrecer cantidades enormes de memoria donde almacenar datos de forma masiva. Ejemplos de esta memoria son el disco de una computadora, un DVD o un CD, o las cintas magnéticas.

Hemos llamado con el acrónimo **UC** a la **Unidad de Control** de la computadora. Es la encargada de interpretar cada instrucción del programa recibida desde la memoria, y de controlar su ejecución una vez interpretada. Capta también las señales de estado que proceden de las distintas unidades de la computadora y que le informan (a la UC) de la situación o condición actual de funcionamiento (por ejemplo, informan de si un determinado periférico está listo para ser usado, o de si un dato concreto está ya disponible). Y a partir de las instrucciones interpretadas y de las señales de estado recibidas, genera las señales de control que permitirán la ejecución, instrucción tras instrucción, de todo el programa.

El último elemento del esquema del procesador es la **ALU** (**Unidad Aritmético Lógica**). Este bloque de nuestro procesador está formado por una serie de operadores aritméticos y operadores lógicos que producen resultados en su salida a partir de la información almacenada en sus entradas. La ALU, como los demás elementos de la computadora, trabaja a las órdenes de las señales de control generadas en la UC.

Al conjunto de la UC y la ALU se le llama **CPU** (**Unidad Central de Proceso**).

Existen dos elementos más, para recoger la estructura general de una computadora, y que completan el cuadro presentado en la figura 1.1. Uno son los llamados **controladores** de entrada y salida, que permiten la correcta comunicación entre el procesador y los diferentes dispositivos de entrada y salida. El otro elemento es el **bus del sistema**, que comunica todas las unidades, y permite el trasiego de datos (bus de datos), de direcciones de memoria (bus de direcciones) donde deben leerse o escribirse los datos, y de las diferentes sentencias de control generadas por la UC (bus de control).

Para terminar esta rápida presentación del procesador, conviene referir algunos elementos básicos que componen la UC, la ALU, y la memoria principal.

La Unidad de Control dispone, entre otros, de los siguientes elementos:

Registro de instrucción, que contiene la instrucción que se está ejecutando.

Contador de programa, que contiene permanentemente la dirección de memoria de la siguiente instrucción a ejecutar y la envía por el bus de direcciones.

Decodificador, que se encarga de extraer el código de operación de la instrucción en curso.

Secuenciador, que genera las microórdenes necesarias para ejecutar la

instrucción decodificada.

Reloj, que proporciona una sucesión de impulsos eléctricos constantes.

A su vez, la Unidad Aritmético Lógica, dispone de los siguientes elementos:

Registros de Entrada, que contienen los operandos de la operación que se va a realizar.

Registro acumulador, donde se almacenan los resultados de las operaciones.

Registro de estado, que registra las condiciones de la operación anterior.

Circuito Operacional, que realiza las operaciones con los datos de los registros de entrada y del registro de estado, deja el resultado en el registro acumulador y registra, para próximas operaciones, en el registro de estado, las condiciones que ha dejado la operación realizada.

Y finalmente, la memoria dispone también de una serie de elementos, que se recogen a continuación:

Registro de direcciones, que contiene la dirección de la posición de memoria a la que se va a acceder.

Registro de intercambio, que recibe los datos en las operaciones de lectura y almacena los datos en las operaciones de escritura.

Selector de memoria, que se activa cada vez que hay que leer o escribir conectando la celda de memoria a la que hay que acceder con el registro de intercambio.

Señal de control, que indica si una operación es de lectura o de escritura.

Instrucciones, Lenguajes, Compiladores.

Una **instrucción** es un conjunto de símbolos que representa (que codifica) una orden para el computador, que indica una operación o tratamiento sobre datos. Y un programa es un conjunto ordenado de instrucciones que se le dan a la computadora y que realizan, todas ellas, un determinado proceso.

Tanto las instrucciones, como los datos a manipular con esas instrucciones, se almacenan en la memoria principal, en lugares distintos de esa memoria.

Las instrucciones se envían desde la memoria hacia la Unidad de Control donde son interpretadas y donde se generan las señales de control que gobiernan los restantes elementos del sistema.

Los datos pueden intervenir como operandos en un procedimiento (información inicial) o ser el resultado de una secuencia determinada de instrucciones (información deducida). En el primer caso, esos datos van de la memoria a la Unidad Aritmético Lógica, donde se realiza la operación indicada por la presente instrucción en ejecución. Si el dato es el resultado de un proceso, entonces el camino es el inverso, y los nuevos datos obtenidos son escritos en la memoria.

El tipo de instrucciones que se pueden codificar en la UC depende, entre otros factores de la ALU de que dispone el procesador en el que se trabaja. De forma general esas instrucciones se pueden clasificar en los siguientes tipos: de transferencia de información (por ejemplo, copiar un valor de una posición de la memoria a otra posición); aritméticas, que básicamente se reducen a la suma y la resta; lógicas, como operaciones relacionales (comparar valores de distintos datos) o funciones lógicas (AND, OR y XOR).

Las instrucciones a ejecutar deben ser codificadas de forma que la máquina las "entienda". Ya hemos visto que todo en una computadora se codifica con bits, a base de ceros y unos. Con ceros y unos se debe

construir toda información o sentencia inteligible para la computadora. Lograr comunicarse con la máquina en ese lenguaje de instrucciones que ella entiende (código máquina) es una tarea difícil y compleja.

Un lenguaje así está sujeto a frecuentes errores de trascripción. Y resulta completamente inexpresivo. Además, otro problema, no pequeño, de trabajar en el lenguaje propio de una máquina es que un programa sólo resulta válido para esa máquina determinada, puesto que ante máquinas con distinta codificación se tendrá lógicamente lenguajes diferentes. El único lenguaje que entiende directamente una máquina es su propio lenguaje máquina.

Resulta mucho más sencillo poder expresar las instrucciones que debe ejecutar la computadora en un lenguaje semejante al utilizado habitualmente por el hombre en su comunicación. Esa es la finalidad de los lenguajes de programación. Pero un lenguaje así, desde luego, no lo entiende una máquina que sólo se codifica con ceros y unos.

Un lenguaje de programación no puede tener la complejidad de un lenguaje natural de comunicación entre personas. Un buen lenguaje de programación debe permitir describir de forma sencilla los diferentes datos y estructuras de datos. Debe lograr expresar las distintas instrucciones de forma sencilla y precisa. Ha de resultar fácil escribir programas con él. Conocer un lenguaje es tanto como saber su léxico (semántica) y su sintaxis; y tener acotados el conjunto de símbolos permitidos para expresarse en él.

Así han surgido los distintos lenguajes de programación, capaces de expresar instrucciones en unas sentencias que quedan a mitad de camino entre el lenguaje habitual y el código máquina.

Dependiendo del grado de semejanza con el lenguaje natural, o de la cercanía con el lenguaje de la máquina, los lenguajes pueden clasificarse en distintas categorías:

1. El lenguaje de bajo nivel, o ensamblador, muy cercano y parecido al

lenguaje máquina. Cada instrucción máquina se corresponde con una instrucción del lenguaje ensamblador, codificada, en lugar de con ceros y unos, con una agrupación de tres o cuatro letras que representan, abreviadamente, la palabra (habitualmente inglesa) que realiza la operación propia de esa instrucción.

2. Lenguajes de alto nivel, que disponen de instrucciones diferentes a las que la máquina es capaz de interpretar. Habitualmente, de una instrucción del lenguaje de alto nivel se derivan varias del lenguaje máquina, y la ejecución de una instrucción de alto nivel supone la ejecución de muchas instrucciones en código máquina. Normalmente esas instrucciones se pueden expresar de una forma cómoda y comprensible. Así, por ejemplo, un lenguaje de alto nivel podría disponer de una instrucción para dividir enteros, que se ejecutaría mediante múltiples instrucciones de suma y resta del lenguaje máquina.

Para resolver este problema de comunicación entre la máquina y su lenguaje máquina y el programador y su lenguaje de programación, se emplean programas que traducen del lenguaje de programación al lenguaje propio de la máquina. Ese programa traductor va tomando las instrucciones escritas en el lenguaje de programación y las va convirtiendo en instrucciones de código máquina.

Gracias a la capacidad de traducir un programa escrito en lenguaje de alto nivel al lenguaje máquina, se puede hablar de portabilidad en los lenguajes de alto nivel: la posibilidad de que un mismo programa pueda ser ejecutado en computadoras diferentes gracias a que cada una de ellas dispone de su correspondiente traductor desde el lenguaje en que va escrito el programa hacia el propio lenguaje máquina.

Se disponen de dos diferentes tipos de traductores. Unos, llamados **intérpretes**, van traduciendo el programa a medida que éste se ejecuta. Cada vez que un usuario quiera ejecutar ese programa deberá disponer del intérprete que vaya dictando a la computadora las

instrucciones en código máquina que logran ejecutar las sentencias escritas en el lenguaje de alto nivel. Un intérprete hace que un programa fuente escrito en un lenguaje vaya, sentencia a sentencia, traduciéndose y ejecutándose directamente por el ordenador. No se crea un archivo o programa en código máquina. La ejecución del programa debe hacerse siempre supervisada por el intérprete.

Otro tipo de traductor se conoce como **compilador**: un compilador traduce todo el programa antes de ejecutarlo, y crea habitualmente un programa redactado en código máquina, que puede ser ejecutado tantas veces como se quiera, sin necesidad de disponer del código en el lenguaje de alto nivel y sin necesidad tampoco de tener el compilador, que una vez ha creado el nuevo programa en lenguaje máquina ya no resulta necesario para su ejecución. Una vez traducido el programa al correspondiente lenguaje o código máquina, su ejecución es independiente del compilador.

Soporte físico (hardware) y soporte lógico (software). Sistemas Operativos.

Se habla de **hardware** cuando nos referimos a cualquiera de los componentes físicos de una computadora: la cpu, la memoria, un dispositivo de entrada…

Se habla de **software** para referirse a los diferentes programas que hacen posible el uso de la computadora para unas aplicaciones concretas.

Dependiendo del hardware de una computadora, ésta se puede clasificar en función de su capacidad y potencia. Muy extendidos están los llamados PC (computadoras profesionales o personales). Habitualmente trabajaremos en ellos y, de ahora en adelante, cuando queramos referirnos a una máquina o computadora pensaremos en un PC, al que llamaremos, sencillamente, ordenador.

Un **sistema operativo** es un programa que actúa de interfaz o conexión entre el usuario de un ordenador y el propio hardware del ordenador. Ofrece al usuario el entorno necesario para la ejecución de los distintos programas.

Un sistema operativo facilita el manejo del sistema informático, logrando un uso eficiente del hardware del ordenador. Facilita al usuario la correcta ejecución de los programas, las operaciones de entrada y salida de datos, la manipulación de la información almacenada en la memoria, la detección de errores,… Permite que el sistema tenga una racional y correcta asignación de los recursos.

Sistemas operativos conocidos son, por ejemplo, el Unix, o su versión para PC llamada Linux, y el comercialmente extendido Windows, de Microsoft.

Recapitulación.

Hemos introducido algunas nociones necesarias para poder luego enfrentarnos al estudio de la programación y, en concreto, de una lenguaje (como el lenguaje C, o el lenguaje Java): qué es un lenguaje de programación y cómo se logra que un ordenador sea capaz de interpretar correctamente las instrucciones que el programador le indica con un determinado lenguaje.

Y hemos presentado los conceptos básicos necesarios para conocer cómo está construido y cómo funciona internamente un ordenador, y cómo es posible que realice una serie de operaciones perfectamente definidas y en correcta secuencia.