CAPÍTULO 1

CONCEPTO DE COMPUTADOR

La Informática, definida por la Real Academia como el conjunto de conocimientos científicos y técnicos que hacen posible el tratamiento automático de la información por medio de computadoras, requiere previamente contestar a determinadas preguntas sobre el funcionamiento de estas máquinas.

En este primer capítulo se pretende dar los conceptos mínimos e imprescindibles, para que el lector tenga una visión panorámica del contenido del libro, de forma que cuando se adentre en los próximos capítulos sepa enmarcar el sentido e importancia de cada uno de ellos.

1.1. UNA PRIMERA APROXIMACIÓN AL COMPUTADOR

De forma genérica y en una primera aproximación, un computador puede definirse como una "**máquina digital, electrónica, programable para el procesamiento de información**". Notemos por tanto, que el computador[1], en cuanto a "**máquina**" se refiere, está en la categoría de los molinos de vientos, telares, etc., que pueden funcionar correcta e incorrectamente. "**Digital**" significa que estas máquinas trabajan almacenando información, en forma de códigos, que representan las letras o *dígitos* de los números. "**Electrónica**" sugiere que está construida usando componentes electrónicos de estado sólido, conocidos por circuitos integrados. **Programable** significa que admite la posibilidad de ejecutar instrucciones a demanda de una sucesión de órdenes preestablecidas. Un ejemplo de máquina programable, son las lavadoras automáticas actuales ó los más tradicionales carrillones.

[1] Al definir computador como un tipo de máquina (femenino), podríamos utilizar la palabra computadora, como se hace en Sudamérica. Por las funcionalidades de la máquina, el término "computador/a", que proviene del latín computare, debería ser preferido al más popular de ordenador (ordenar es sólo una de las múltiples funciones que la máquina es capaz de llevar a cabo). En este libro usaremos libremente cualquiera de estos tres sinónimos.

El concepto "**procesamiento de información**", es menos obvio. **Información** es un término general que abarca hechos y representaciones que pueden estar o no relacionadas entre sí. A dichas entidades codificadas, en condiciones de ser introducidas en el computador, los llamaremos **datos** que pueden usarse de forma individual o como grandes conjuntos, dotados de una estructura (por ejemplo una matriz); de hecho los datos son significativos en sí y en relación con la estructura a la que pertenecen (un número de teléfono no es de gran utilidad si no se sabe a quien pertenece).

La unidad más elemental de información es una variable binaria, conocida como BIT (contracción de las palabras BInary y digiT), que toma el valor 0 ó 1. El bit representa la mínima información posible, ya que corresponde a la ocurrencia de un suceso, de entre dos posibilidades distintas. Por ejemplo, un bit es la cantidad de información correspondiente a un mensaje anunciando si determinado caballo ha ganado (1) o perdido (0) una carrera.

1.1.1 OPERACIONES BÁSICAS DEL PROCESADO DE DATOS

Procesar los datos, consiste en someterlos a un conjunto de operaciones tales como ordenación, selección, ejecución de cálculos, etc. de forma que nos permita extraer conclusiones de los datos que manipulamos. Por tanto, procesar información es transformar datos primarios en información organizada, significativa y útil, que a su vez está compuesta de datos (Ver Figura 1.1.)

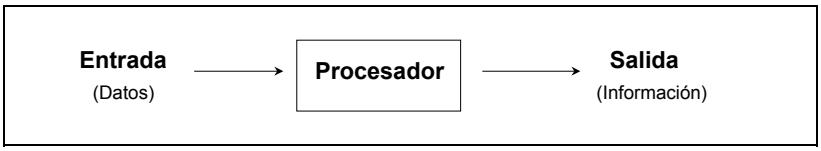

Fig. 1.1. *Proceso de datos*

En el concepto de procesar, entran ejemplos tan diversos como la obtención de las raíces de un polinomio a partir de los datos de los valores de sus coeficientes, o el control de un cohete en vuelo, a partir de todos los datos físicos del sistema. Nótese, que el concepto de procesador es más amplio que el de computador, ya que existen procesadores que no son computadores: Un termostato a partir del dato de la temperatura tiene como salida el control del aire acondicionado y nuestro propio sistema cardiovascular, a partir de los datos sensoriales reacciona adaptándose a las distintas circunstancias.

De acuerdo con lo anterior, para procesar datos, el computador debe ser capaz de realizar ciertos tipos de operaciones:

- Entrada de datos: Suministrar información al computador desde su entorno exterior (ej. pulsar el teclado, o leer un código de barras).
- Salida de datos: Obtener información de un computador. (ej. visualizar los resultados en una pantalla o impresora).
- Almacenamiento: Hacer una copia permanente de la información con el objetivo, que el computador pueda emplearla de nuevo (ej. copiar en cintas y discos magnéticos).
- Recuperación: Leer de nuevo la información almacenada (en cinta o discos magnéticos).
- Transmisión: Transferir la información a otro computador a través de una red de comunicación de datos.
- Recepción: Recibir la información enviada por otro computador.
- Tratamiento: Operaciones sobre datos, tales como la ordenación, selección, combinación, reclasificación, así como la ejecución de cálculos, que permita obtener la información deseada.

1.1.2 ALGORITMOS Y PROGRAMAS

Llamaremos **algoritmo** a un conjunto de pasos y acciones, que especifican de forma no ambigua y finita, la secuencia de operaciones a realizar para procesar datos con un determinado objetivo. (Este concepto abarca, desde un método de resolución de ecuaciones, hasta la ejecución de una receta de cocina). Hagamos dos aclaraciones previas: a) La algorítmica existe con anterioridad al desarrollo de los computadores actuales (el propio origen de la palabra proviene de un matemático árabe que describió en el siglo VIII el método manual con el cual dividimos dos números). b) Una vez encontrada este conjunto de instrucciones que resuelven el problema, éstas se pueden ejecutar sin que sea necesaria la comprensión de los principios en los que se basa el algoritmo. Así cuando dividimos dos números o construimos una pajarita a partir de una cuartilla, nos limitamos a ejecutar una serie de acciones en un determinado orden, sin preocuparnos demasiado por sus correspondientes bases teóricas.

Para el caso en que el procesador sea un computador, la forma de expresión de este algoritmo se llama **programa**. Cada paso del algoritmo se expresa en el programa, por medio de un conjunto de **instrucciones** escritas en un determinado lenguaje. Uno de los aspectos más llamativos de los computadores es el hecho de que con un reducido número de instrucciones básicas, pueda llevarse a cabo una gran cantidad de tareas.

El conjunto de pasos que constituyen el programa debe escribirse de forma muy precisa. La ambigüedad inmanente a los lenguajes naturales (español, inglés, etc.) los invalida para expresar algoritmos, por ello existe un cierto tipo de lenguajes que

reúne la suficiente precisión como para expresar algoritmos: son los llamados **lenguajes de programación**. Estos lenguajes tienen, como todo lenguaje una **sintaxis** (cómo agrupar las símbolos y elementos propios del lenguaje para formar frases válidas) y una **semántica** (qué significa cada frase válida de este lenguaje) diferenciándose de los lenguajes naturales en cuatro características:

1) Su vocabulario y su sintaxis son sencillos y limitados. Esto los hace, inadecuados para describir cualquier tipo de prosa no algorítmica.
2) El vocabulario de un lenguaje de programación contiene solamente aquellos tipos de acciones básicas que puede ejecutar un ordenador (operaciones aritméticas, lógicas, entrada/salida, etc.) y no otras.
3) La sintaxis es muy rígida y no permite excepciones ni muchas variaciones. Por ejemplo para calcular una división de "X" por "A", hay que escribir obligatoriamente "X/A".
4) Su semántica es estricta y la ambigüedad no tiene cabida en ellos.

Un mismo algoritmo puede expresarse en diferentes lenguajes y ejecutarse en distintos ordenadores. Contar con un algoritmo es una condición necesaria para la obtención de un programa, aunque no es suficiente, ya que su realización requiere además, creatividad y conocimientos sobre los recursos físicos y lógicos del ordenador. La fase de conversión del algoritmo a programa se denomina **codificación**, ya que el algoritmo escrito en un lenguaje específico de programación se denomina **código.**

Para que se puedan procesar los datos por medio de un programa, el conjunto de instrucciones que lo constituyen debe ser accesible para la máquina y en consecuencia, ésta debe tener la capacidad de almacenarlo internamente, al objeto que pueda efectuar el control de las operaciones que constituyen el algoritmo. Aunque todavía no hayamos profundizado en el concepto de computador, sí que podemos deducir que éste tiene que ser capaz de trabajar y reconocer simultáneamente tanto las instrucciones, que determinan lo que hay que hacer, como los datos que se van a procesar. Este requisito es tan fundamental, que la forma como éste se ha venido plasmando, conceptual y técnicamente a lo largo de la historia, permite rastrear los orígenes de la informática.

1.2. ANTECEDENTES HISTÓRICOS

En 1982, la revista TIME nombró "hombre del año" a la computadora. La noticia supuso un impacto cultural importante y oficializó el hecho de que esta máquina es un personaje más de la historia de la humanidad. Aunque considerada como un paradigma de la etapa moderna y con sólo 30 años de verdadera historia

(hasta mediados de los años 60 las computadoras eran caras y para usos muy específicos que sólo las grandes instituciones y universidades podían permitirse) su consecución, de una u otra forma, ha ocupado y preocupado a muchas de las cabezas que jalonan la historia de la ciencia. El hecho de que nuestro estado actual de avance tecnólogico haya permitido su construcción, no debe ocultar los esfuerzos hechos desde los inicios del pensamiento científico.

Ábacos

La necesidad de contar con un elemento mecánico que ayude a las personas en sus tareas de manipulación numérica, o lo que es lo mismo una máquina que ejecute algoritmos de forma más o menos automática es realmente antigua. Todos los historiadores coinciden en citar como antecedente de lo que hoy llamamos computador al ábaco; instrumento utilizado para la suma y la resta, desde hace 4000 años. Este consiste en un conjunto de alambres que llevan ensartados unas cuentas móviles, colocadas en un marco rectangular cruzado por una barra horizontal que atraviesa los alambres (Figura 1.2).

Cada una de las cuentas situadas por encima de la barra horizontal vale cinco, y cada cuenta situada bajo la barra vale uno. Si comenzamos por la parte de la derecha y nos vamos moviendo hacia la izquierda, el primer alambre representa el dígito de las unidades; el segundo, el de las decenas; y sucesivamente las centenas, los millares, etc. Borramos o ponemos a cero el ábaco separando todas las cuentas de la barra horizontal. Introducimos los números moviendo las cuentas correspondientes hacia esa barra, comenzando por el dígito situado más hacia la derecha, y moviéndonos hacia la izquierda.

Este artilugio resulta de especial interés para ayudar a sumar o restar dos números. La entrada de datos se hace a través de una persona, que también se encarga de la ejecución y control del algoritmo de suma o resta correspondiente, la salida de datos es consecuencia de la observación de las cuentas, por lo que esta máquina seria un sistema de almacenamiento, que necesita de un experto humano para procesar información (suma y resta).

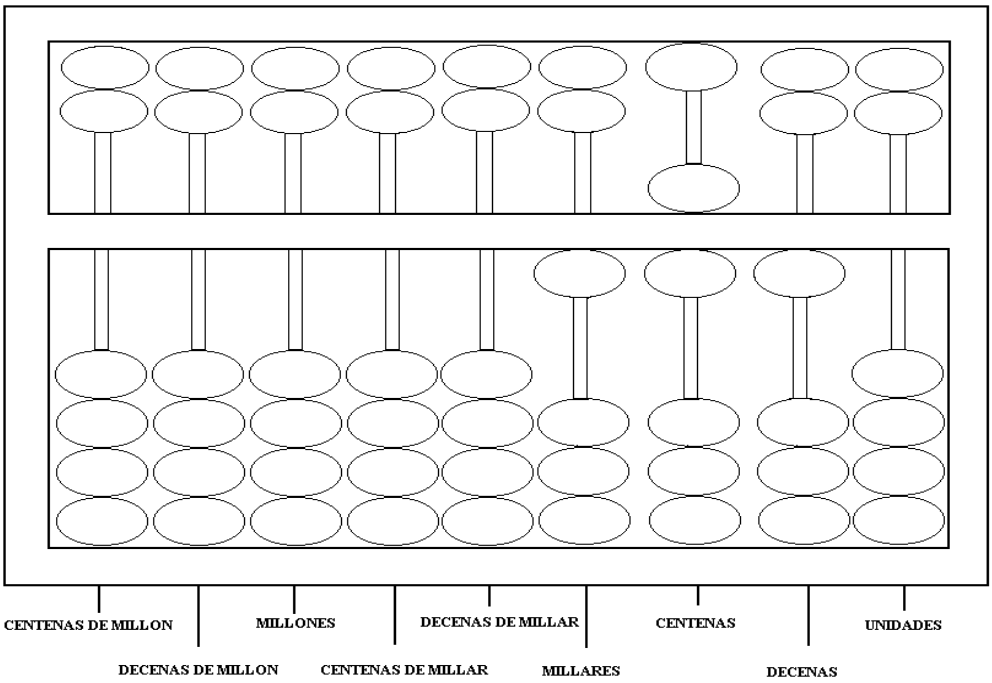

Fig. 1.2 *Representación en un ábaco del número 1610*

Calculadoras mecánicas

A medida que las distintas otras ramas científicas se desarrollaron, se puso en evidencia la necesidad de optimizar la capacidad de cálculo habida cuenta de lo dificultoso de las operaciones que se tenían que llevar a cabo. Ello hizo que muchos matemáticos se plantearan el desarrollo de instrumentos que facilitaran al menos teóricamente esta labor. Así, John Napier, que recordamos por los logaritmos neperianos, publicó un estudio en 1617 que describía la utilización de cuentas con una marca especial, para realizar multiplicaciones y divisiones (la obra matemática de Napier condujo mucho más adelante al desarrollo de la regla de cálculo, venerable instrumento que durante muchos años fue para algunos de nosotros la "máquina" primaria para llevar a cabo cálculos complejos). En 1642, Blaise Pascal inventó la primera "sumadora" real, parecida a las calculadoras mecánicas que se popularizarian en nuestros años sesenta. Se trataba de una compleja combinación de ruedas, engranajes y ventanas a través de las cuales aparecían los números. A finales del siglo XVII, otro famoso matemático, Gottfried Leibnitz, desarrolló una máquina parecida, pero más avanzada; podía sumar, restar, multiplicar y dividir mecánicamente, e incluso sacar raíces cuadradas.

A pesar de la ingeniosidad de estos planteamientos mecánicos, hasta 1820 no se pudo disponer de la tecnología que permitiera la aparición de las primeras máquinas comerciales capaces de efectuar las cuatro operaciones matemáticas básicas. Esta es una constante en la historia de las máquinas de calcular, el decalaje existente entre los diseños originales y las disponibilidades tecnológicas para llevarlos a la práctica. En la máquina de Leibnitz, se produce un avance respecto al ábaco, puesto que el algoritmo esta ya incorporado en la propia estructura de la máquina y además respecto a la de Pascal supone un incremento en flexibilidad, pues el usuario puede seleccionar, la operación que desea llevar a cabo. Sin embargo, en su tiempo no pasaron de ser meras curiosidades, sin posibilidad de aplicación práctica.

Las tarjetas perforadas y los computadores mecánicos

A comienzos del siglo XIX se producen aportaciones, curiosamente ligadas a la resolución de problemas de naturaleza no numérica. En 1801, Joseph Jacquard, inventó un telar controlado mediante instrucciones almacenadas según un código representado en tarjetas perforadas; de esta forma, el algoritmo que segue la máquina podrá cambiarse fácilmente para conseguir un dibujo distinto sobre la tela. La máquina de Jacquard incorporó varios elementos básicos que constituyen los computadores actuales.

La idea de usar en tarjetas perforadas, para guardar tanto números como instrucciones datos indujo, en 1835, a Charles Babbage, a inventar un computador digital matemático de tipo mecánico que recibió el nombre de máquina analítica. Babbage utilizó las tarjetas perforadas para programar su máquina, que podía utilizar los resultados de un cálculo como entrada del siguiente y que era capaz de manejar cálculos repetitivos, que más adelante llamaremos bucles. Un logro, aun más significativo, de la máquina analítica fue que, en vez de seguir las instrucciones del programa en la secuencia prefijada, podía saltar de una a otra (los programadores actuales hablamos de bifurcaciones condicionales). Aunque los computadores actuales están basados en muchos de los principios que Babbage utilizó, en su tiempo no existía ningún procedimiento que pudiera mover su cada vez más complejo artilugio mecánico. Una vez más, la tecnología no era la adecuada para el desarrollo teórico propuesto y el trabajo de Babbage se saldó con un fracaso, sólo mitigado por el apoyo de Ada Lovelace, hija de Lord Byron, considerada como la primera programadora de la historia y en cuyo honor el lenguaje de programación ADA lleva su nombre.

El concepto de la tarjeta perforada para almacenar programas y datos llegó a prender, y Herman Hollerith para procesar el censo de 1890 en EEUU la recuperó con notable éxito (consiguió un ahorro de tiempo de varios meses) al incorporarla a máquinas alimentadas eléctrica y no mecánicamente. Hollerith puso las bases de

una de las empresas que más adelante se integraría en la International Business Machines (IBM).

En paralelo, George Boole (1815-1864), el fundador de la teoría de la lógica matemática, nos legó un álgebra para representar cantidades lógicas e investigó las operaciones que se pueden realizar con estas variables (álgebra de Boole). La lógica booleana es la base teórica tanto para el diseño de circuitos electrónicos como para muchas técnicas de programación, aunque difícilmente, pudo ser Boole consciente de la trascendencia práctica de su teoría.

Los computadores electromecánicos

Coincidiendo con la aparición de las máquinas eléctricas y bajo le presión del esfuerzo bélico de la Segunda Guerra Mundial (resolución de claves secretas, etc,) se desarrollaron computadores electromecánicos, como elMark I (1944) de la Universidad de Harvard. Se trataba de una máquina de grandes dimensiones, 15,5 metros de largo por 2,4 metros de altura, en la que las instrucciones se introducían mediante cinta de papel, y los datos mediante tarjeta perforada y un teletipo iba escribiendo los resultados. El Mark I podía multiplicar dos números en unos tres segundos. En 1947, su sucesor el Mark II podía llevar a cabo la misma multiplicación en un cuarto de segundo aproximadamente. Era doce veces más rápido y suponía, un gran avance; sin embargo su obsolescencia fue inmediata, como veremos a continuación, con el advenimiento de la electrónica.

Computadores electrónicos

La electrónica empieza con la válvula o tubo de vacío (un bulbo de vidrio, donde una placa de metal calentada por un filamento emite electrones que se desplazan en el vacío debido a una diferencia de potencial entre el cátodo y el ánodo). El primer computador digital electrónico utilizaba tubos de vacío y podía realizar una multiplicación en unos 2,8 milisegundos. Fue desarrollado en 1946 por un equipo de la Universidad de Pennsylvania. Esta máquina recibió el nombre de ENIAC, (Electronic Numerical Integrator and Computer) su primera aplicación, a pesar de la finalización de la guerra, fue el cálculo de las tablas de tiro de la artillería. El ENIAC se programaba cambiando manualmente los conectores y manipulando conmutadores, lo que, como podemos imaginar, requería gran cantidad de tiempo, por lo que su uso casi no traspasó el ámbito académico.De hecho, éstas máquinas no tuvieron ningún impacto comercial, aunque sentaron claramente las bases del diseño de los computadores actuales. Cuando en 1947 tiene lugar, en los laboratorios Bell, la invención del **transistor** (más pequeño que la válvula, menor consumo y mayor fiabilidad) se posibilita el paso del computador, de pieza experimental de laboratorio, a dispositivo con ciertas posibilidades comerciales.

Otro avance tecnológico posterior, **los circuitos integrados**, permitieron integrar en un único sustrato de silicio cientos de transistores (actualmente esta integración es ya del orden de millones, en los llamados chips[2]), con lo que quedaron sentadas las bases de los computadores actuales. A pesar de que quedaban muchos problemas por resolver, a principios de los años sesenta, las ideas y las tecnologías estaban ya maduras y se trataba de seguir una evolución, cuyos resultados no han dejado de sorprendernos hasta ahora. El computador es uno de los elementos claves de la actual revolución científico-industrial y su papel, en la historia de la humanidad, no será menor que el que jugaron en su día, la rueda, el motor de explosión o la energía nuclear.

1.3. ORGANIZACIÓN DE UN COMPUTADOR

1.3.1 LA ARQUITECTURA DE VON NEUMANN

John von Neumann[3] (1903-1957), que había colaborado en el diseño de varios computadores durante la Segunda Guerra Mundial, publica en 1946 un artículo que esboza los principios generales de diseño de un computador en su sentido actual:

- La máquina es controlada por un conjunto de instrucciones, con un pequeño número de elementos centrales de proceso.

- El programa se almacena en el computador de forma que en su representación interna no se hacen distinciones entre datos e instrucciones, ambos almacenados en código binario.

De esta forma el programa pasa a estar "dentro" del computador, y así el cambio de un programa a otro sólo implica un cambio en el valor de posiciones de memoria, en contraposición con lo hecho hasta la fecha que requería cambios manuales de clavijas. Estos principios, enriquecidos con importantes aportaciones tecnológicas,

---

[2]. La forma que adoptan estos circuitos encapsulados con sus conectores, los asemeja a las pulgas (*chip* en inglés), por lo que esta denominación es la que ha terminado por imponerse.

[3] Von Neumann y sus colaboradores describieron una máquina constituida por dos órganos, uno constituido por las unidades de control, de aritmética y de entrada salida y otro formado por la memoria. En esta máquina la memoria ocupaba 4096 palabras o posiciones de memoria cada una de 40 bits. Cuando una palabra se interpretaba como dato, estos 40 bits representaban un número en notación binaria y cuando lo hacía como instrucción, cada palabra contenía 2 instrucciones de 20 bits. Los datos eran todos enteros y las operaciones aritméticas que podía efectuar eran suma, resta, multiplicación, división y valor absoluto, de forma que el resultado de cualquiera de ellas se situaba en un registro llamado acumulador. A cada posición de memoria podía asignársele el valor contenido en el acumulador, de forma que se reemplazaba el valor anterior almacenado en esta posición de memoria. El flujo de control del programa era el de pasar de una instrucción a la siguiente, pudiendo ser interrumpida esta secuencialidad a través de un salto incondicional (goto) de forma que la siguiente instrucción a ser ejectuada era la de la palabra de memoria que se mencionaba en la instrucción del salto incondicional.

se han mantenido vigentes hasta nuestros días en lo que llamamos máquina de von Neumann, y cuyo esquema general es el de la Figura 1.3.

1.3.2 UNIDADES FUNCIONALES

Las unidades funcionales de una máquina de von Neumann son cinco: **Unidad de Control** (UC), **Unidad Aritmético-Lógica** (ALU), **Unidad de Entrada**, **Unidad de Salida** y **Memoria (principal y secundaria)**. Por su disposición y función, se llama **unidad de central de proceso (CPU = Central Process Unit)** al conjunto formado por la UC y la ALU, donde reside la 'inteligencia' de la máquina, es decir, la capacidad de procesamiento de la información. La **memoria** es el lugar donde se almacenan los datos y las instrucciones de forma permanente o transitoria. Un vistazo al esquema de la Figura 1.3., justifica que llamemos **periféricos** al conjunto de unidades de E/S y de memoria masiva ó auxiliar.

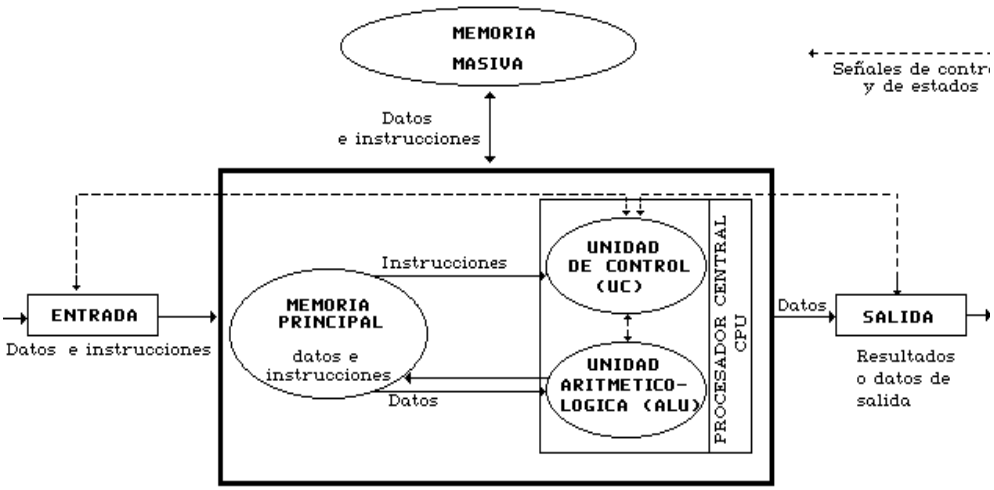

Fig. 1.3 *Esquema de una máquina de von Neumann*

Veamos ahora con más detalle las unidades funcionales:

UNIDAD DE ENTRADA (E). Son los dispositivos por donde se introducen los datos e instrucciones. En estas unidades se transforman las informaciones de entrada, en señales binarias de naturaleza eléctrica. Son unidades de entrada: el

teclado, un terminal de ventas, un digitalizador, una lectora de tarjetas de crédito, etc.

UNIDAD DE SALIDA (S). Son los dispositivos por donde se obtienen los resultados de los programas ejecutados en el computador. La mayor parte de estas unidades transforman las señales eléctricas binarias en caracteres, escritos o visualizados, inteligibles por un humano o por otra máquina. Son dispositivos de salida: un monitor, una impresora, un registrador gráfico, el visor de un casco de realidad virtual, etc.

LA UNIDAD DE PROCESAMIENTO CENTRAL (CPU). La CPU se encarga de manipular los datos, gracias a su capacidad para ejecutar una serie de tareas básicas que se corresponde con las operaciones incluidas dentro de sus circuitos, contenidos, normalmente, en un solo chip llamado *microprocesador* Estos están constituidos, al menos, por dos unidades básicas, la UC y la ALU.

LA UNIDAD DE CONTROL. Es la encargada de administrar y coordinar los recursos y actividades del computador. Se puede pensar en la UC, como si fuera un guardia de tráfico, dirigiendo el flujo de datos dentro de la máquina, esto es, un coordinador que controla la ejecución "tirando de la cuerda adecuada, en el momento adecuado". Por ello la UC tiene misiones relacionadas con: identificar instrucciones, supervisar su ejecución (enviando bits a las líneas de señal adecuadas en los momentos adecuados), y detectar señales eléctricas, procedentes del resto de unidades. Durante la ejecución de un programa la UC de forma sucesiva, debe seguir el siguiente ciclo: a) Captar de memoria una a una las instrucciones. b) Identificar las unidades involucradas en las operaciones asociadas a la ejecución de la misma. c) Generar, de acuerdo con el código de operación y con las señales de estado, las correspondiente señales de control.
Más específicamente, las funciones de la UC son:

1. Interpretar el código y generar las señales de control que lo ejecutan.
2. Controlar la secuencia en que se ejecutan las operaciones.
3. Controlar el acceso a la memoria principal
4 Enviar y recibir señales de control relacionadas con las operaciones que ejecuta la ALU
5. Regular las temporizaciones de las unidades de E/S

UNIDAD ARITMÉTICO-LÓGICA. Esta unidad contiene los circuitos electrónicos con los que se llevan a cabo las operaciones básicas de tipo aritmético (suma, resta, multiplicación, división, etc.), y de tipo lógico (comparar dos números, hacer operaciones del álgebra de Boole, etc). Este aparentemente limitado conjunto de circuitos es más que suficiente, ya que todas las operaciones se pueden llevar a cabo en términos de un gran número de pequeños pasos, involucrando cada uno de ellos, uno o dos circuitos.

La ALU aparte de los circuitos operativos, contiene un grupo de **registros**, es decir, de pequeñas memorias construidas directamente en la CPU, que se usan para guardar los datos que van a ser procesados por la instrucción actual. Por ejemplo, la UC puede cargar dos números de memoria en los registros de la ALU, para luego ordenarle, que los divida (una operación aritmética) o que verifique si son iguales (una operación lógica), dejando el resultado correspondiente en otro registro de la unidad.

MEMORIA

La CPU contiene la lógica y los circuitos para que pueda funcionar la computadora, sin embargo carece de espacio para guardar programas y datos, esta es precisamente la misión de la unidad de MEMORIA: almacenar de forma eficiente tanto datos como programas. Notemos que, aunque la CPU contiene registros para datos e instrucciones, éstas son áreas de memoria muy pequeñas, que sólo pueden guardar unos cuantos bits a la vez; cuando hay necesidad de grandes cantidades de espacio de almacenamiento se hace necesaria la memoria externa a la CPU. Existen dos tipos básicos de memoria, diferenciadas tanto por sus funciones como por su velocidad: la principal y la secundaria. En ambas, la información se codifica en un alfabeto binario formado por bits, generalmente 8, llamado **octeto o byte,** que es la unidad utilizada para medir la capacidad de almacenamiento de una memoria. Como el byte es relativamente pequeño, es usual utilizar múltiplos:

1 Kbyte (o KB) = 2^{10} bytes = 1024 bytes ### 10^3 bytes
1 Megabyte (o MB) = 2^{10} Kbytes = 2^{20} bytes = 1048 576 bytes ### 10^6 bytes
1 Gigabyte (o GB) = 2^{10} Mbyte = 2^{30} bytes = 1 073 741 824 bytes ### 10^9 bytes
1 Terabyte (o TB) = 2^{10} Gbyte = 2^{40} bytes ### 10^{12} bytes

Memoria principal (central o interna). Su función es la de almacenar los datos y el programa que va ejecutarse en cada momento, por ello es la más rápida y tiene optimizada sus transferencias, con la CPU Físicamente la memoria consiste en una serie de chips conectados con el microprocesador de la forma más integrada posible. La memoria está dividida en celdas o posiciones denominadas también **palabras de memoria** (Ver Figura 1.4.), estas celdas tienen como longitud, un número determinado de bits, normalmente coincidiendo con un múltiplo del byte: 8, 16, 32, 64, etc.

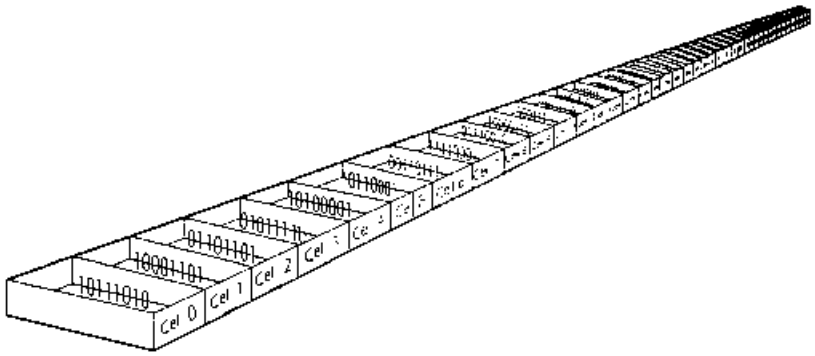

Fig. 1.4. *Disposición de las celdas o posiciones de una memoria*

Existen distintos tipos de memorias y más de una manera de clasificarlas. Una manera de hacerlo es por la permanencia de la información en ellas. Algunos chips de memoria siempre conservan los datos que tienen aun cuando la computadora esté apagada; esta memoria se llama **no volátil**. Otros chips, que forman la mayor parte de la memoria, sí pierden su contenido cuando la computadora se apaga, la memoria de éstos es **volátil**.

Una segunda clasificación de la memoria es por la manera en que puede usarse. Algunos chips de memoria siempre guardan los mismos datos, esto es, además de ser no volátiles sus datos no pueden ser cambiados. De hecho, cuando los datos se guardan en este tipo de memoria se llama "grabación permanente de los datos". ya que sólo pueden leerse y usarse, nunca cambiarse. Este tipo de memoria se llama **memoria de sólo lectura** (Read Only Memory, **ROM**). Una de las razones por las que una computadora necesita ROM es para disponer de instrucciones en el momento del encendido, ya que sin datos grabados de forma permanente, no se podría producir todas las tareas que constituyen el proceso de arranque. La memoria cuyo contenido puede cambiarse se llama **memoria de acceso aleatorio** (Random-Access Memory, **RAM**). El propósito de la memoria RAM es guardar los distintos programas y datos, que vamos a utilizar en cada momento. Utilizamos el término acceso aleatorio porque la CPU accede a su memoria, utilizando una dirección de memoria, que es un número que indica un lugar en el chip de memoria, (de la misma forma que si fuera un número de apartado postal, que indica en qué buzón deberá ponerse el correo). De esta manera, la computadora no tiene que reconocer en toda la memoria para encontrar los datos que necesita; puede buscar la dirección e ir directamente a ella.

En ambos tipos de memoria, la lectura es no destructiva ya que el contenido de una posición de memoria no se altera por muchas veces que se lea. Sin embargo, la escritura en una memoria RAM (recordemos que la memoria ROM no permite la

escritura) es destructiva, es decir, cuando se escribe algo en una posición de memoria automáticamente se destruye el contenido anterior.

Memoria auxiliar masiva, (secundaria o externa). La memoria principal, aunque es muy rápida, no tiene gran capacidad de almacenamiento. Para guardar grandes cantidades de información, se utilizan otros tipos de memoria, tales como discos y cintas magnéticas, que siendo más lentas en la transferencia de datos, tienen mucha más capacidad que la memoria principal (del orden de mil veces más lentas pero más capaces). Frecuentemente los datos y programas se graban (introduciéndolos por las unidades de entrada) en la memoria masiva, de forma que si se ejecuta varias veces el programa o se utilizan repetidamente los datos, no es necesario introducirlos de nuevo por el dispositivo de entrada, sino que basta leerlos desde el disco o la cinta.

Las memorias masivas han heredado el concepto de **fichero** de los sistemas manuales de tratamiento de la información, de forma que mucha de su terminología deriva de la tradicional. Así hablaremos de **ficheros o archivos** como una colección ordenada de datos relacionados entre sí. Existen distintos tipos de archivos, algunos de ellos tienen como unidad elemental el **registro**, que se correspondería con una ficha en un fichero manual. Los registros están formados por **campos**, que constituyen unidades de información (la forma más común de identificar un registro es eligiendo un campo dentro del mismo que llamaremos **clave**). En la Figura 1.5. puede verse un ejemplo de este tipo de fichero.

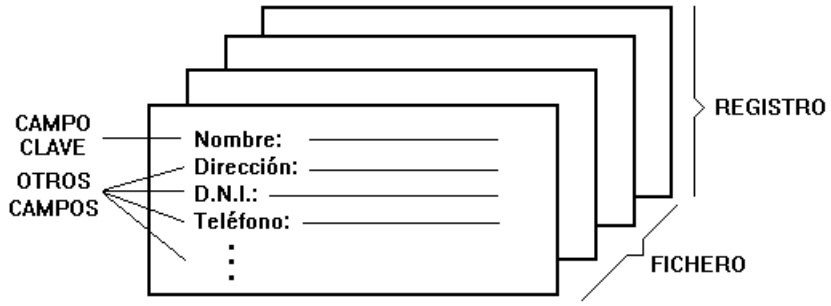

Fig. 1.5. *Elementos de un fichero*

Jerarquía de memoria

Una vez revisadas, las unidades funcionales, hay que hacer notar que la mayor parte del tratamieto de la información se lleva a cabo en la ALU, por lo que resulta importante aumentar la eficacia de su trabajo. Esta eficiencia depende de sus registros, que como sabemos ,son pequeñas memorias que se utilizan para el almacenamiento provisional de datos en el momento de ser objeto de procesamiento. Un registro de datos debe ser lo suficientemente grande como para almacenar los datos que puede manejar la ALU, es decir, si el dato estándar tiene una longitud de m bits, el registro tiene que ser capaz de almacenar los m bits.

Notemos que los datos transitan por los registros constantemente: cuando van a introducirse en la memoria, al acabar de ser extraídos de ella y cuando se obtienen resultados intermedios durante operaciones de la ALU. En cuanto una palabra, contenida en la memoria principal, se transfiere al registro adecuado, es cuando actúa como instrucción o como dato de un programa. Así el programa, que está en la memoria principal, al ejecutarse va captando, bien las diferentes instrucciones bien los datos sobre los cuales se va a actuar, para situarlos en los registros adecuados, consiguiendo con ello facilitar la implementación de las operaciones de la máquina y aumentar su eficacia.

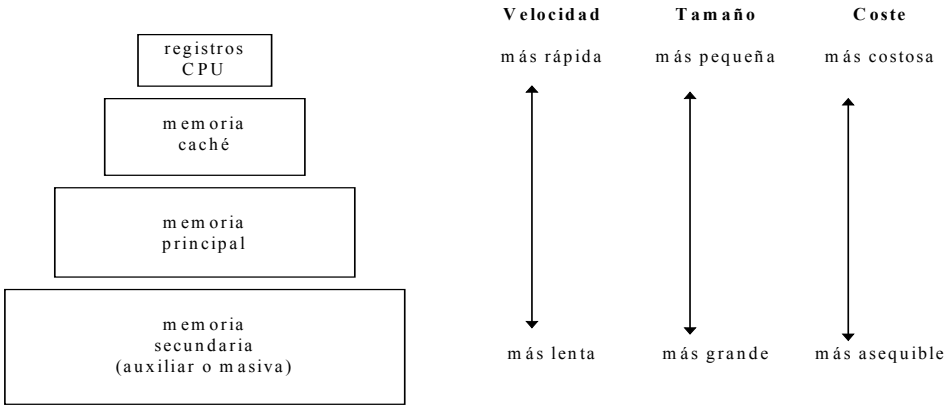

Fig. 1.6. *Estructura de una jerarquía de memoria*

El conjunto de los tipos de memorias se organizan por niveles como una **jerarquía de memorias** (ver Figura 1.6). Los registros se usan para manejar los datos utilizados en la operación que se esta ejecutando, la memoria principal se utiliza para soportar los datos que serán utilizados en el futuro próximo y la memoria secundaria se usa para aquellos datos que normalmente no van a ser manipulados en el corto plazo. En muchas máquinas existe un nivel adicional, llamado **memoria cache**, consistente en una memoria de gran velocidad con tiempos de respuestas semejantes a los registros, normalmente situada en la propia CPU. La máquina trata de utilizar esta área de memoria como copia de la porción de memoria que le interesa en cada momento. En este caso la transferencia de datos en lugar de hacerse entre memoria y registros se hace entre memoria caché y registros, y resultando mucho más rápida.

BUSES

Un requisito para el buen funcionamiento de la máquina, es la conexión, para el intercambio de información, entre unas unidades y otras. Este intercambio entre unidades se tiene que lleva a cabo en paralelo (varios bits de información al

mismo tiempo) a través de una canal de información o **bus** que contiene un determinado número de hilos conductores.

La palabra es la unidad de transferencia de información, al estar ligada a la estructura interna de la computadora, su longitud, en bits, indica tanto el tamaño de los datos con que opera la ALU, como la de los datos transferidos entre la memoria y la CPU. Por tanto, esta longitud acaba determinando las características físicas de los buses[4]. Así, por ejemplo, se se trabaja con palabras de memoria de 16 bits la información desde una unidad a otra debe transmitirse por un bus de 16 hilos conductores.

A través del bus, la CPU puede extraer (leer) datos de la memoria principal, proporcionando tanto la dirección de la posición de memoria como su contenido y viceversa en el proceso de escritura en memoria. Al bus empleado para intercambiar información entre los distintos subsistemas se le llama **bus de datos**, mientras que si un bus es específicamente utilizado para indicar posiciones de memoria se conoce como **bus de direcciones**.

1.4. EL CONCEPTO DE PROGRAMA ALMACENADO

La idea de von Neumann de que tanto los datos como el programa residan en la memoria principal, supone que una máquina esta diseñada para reconocer ciertos patrones de bits que representan ciertas instrucciones, que ahora vamos a definir con mayor precisión: llamaremos **instrucción** a un conjunto de símbolos que representan una orden de operación, que suele realizarse con o sobre datos y en consecuencia **programa** es un conjunto ordenado de instrucciones, que indican al ordenador los procedimientos que se desean.

1.4.1 TIPOS DE INSTRUCCIONES

Puesto que los circuitos de la UC reconocen y ejecutan un determinado repertorio de instrucciones, propias de su CPU, cuando se ejecuta un programa, la computadora interpreta las ordenes contenidas en las instrucciones y las ejecuta sucesivamente. Un programa se escribe, utilizando una sucesión de instrucciones y salvo indicación contraria, su ejecución coincide con el orden en que las instrucciones han sido escritas (Figura 1.7):

[4]Nótese que las longitudes de palabra de la CPU y de la memoria pueden no coincidir, pues la longitud de la palabra de CPU es el número de bits máximo de los datos con los que opera la ALU, mientras que la longitud de la palabra de memoria es el número de bits que conforman cada posición de memoria (número de bits que se pueden leer o escribir simultáneamente); este número suele coincidir con el número de bits (integrantes de datos o instrucciones) que se transmiten simultáneamente entre las distintas unidades en un instante dado. Y longitud de palabra debe ser tenida en cuenta a la hora de interconectar las distintas unidades.

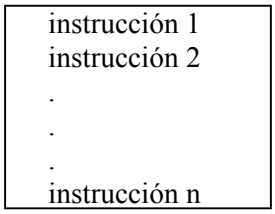

Fig. 1.7. *Orden secuencial de las instrucciones de un Programa*

Sin embargo, es básico, para que la flexibilidad del programa sea tal, que la UC pueda interrumpir el orden normal, mediante instrucciones de **bifurcación**, que permiten que el control del programa pase a una instrucción que no sea necesariamente la inmediata. **Con ello el desarrollo líneal de un programa se interrumpe**.(Figura 1.8)

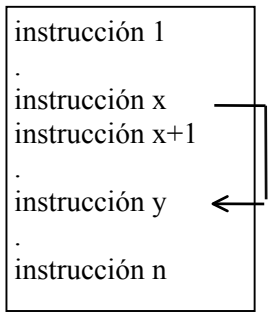

Fig. 1.8. *Ruptura del orden secuencial de las instrucciones de un programa con una bifurcación*

Una instrucción de bifurcación debe indicar el punto del programa hacia a donde se bifurca. Desde el punto de vista del 'sentido' del salto, una bifurcación puede ser hacia delante (en general puede considerarse que el computador deja de ejecutar determinadas instrucciones), o hacia atrás (con lo cual pasa a ejecutar instrucciones que pueden haberse ejecutado previamente).

Desde el punto de vista de la causa que la produce, la bifurcación puede ser: **incondicional**, (el salto se da siempre que el flujo del programa pase por la instrucción), ó **condicional**, caso que la bifurcación dependa del cumplimiento de una determinada condición; si esta se cumple se produce el salto, si no, el programa continúa en la instrucción siguiente a la de bifurcación.

Las instrucciones ejecutables por cualquier CPU, se pueden clasificar en los siguientes grupos:

- Instrucciones de transferencias de datos como por ejemplo: de entrada o lectura (llevar un dato de una unidad de entrada a la memoria o a la ALU), de salida o escritura (llevar un dato de la memoria o de la ALU a una unidad de salida), transferencia de un dato de la memoria a la ALU o viceversa, etc.
- Instrucciones de tratamiento como por ejemplo, sumar dos datos, comparar dos datos para comprobar si son iguales o uno mayor que otro; en particular incluye las operaciones aritmético-lógicas que ejecuta la ALU.
- Instrucciones de bifurcación y saltos. Que permiten alterar el orden secuencial de ejecución.
- Otras instrucciones. Tal como detener el funcionamiento de la computadora a la espera de una acción del operador, cambiar el modo de funcionamiento de la CPU, etc.

Cada modelo de CPU, tiene su propio conjunto de instrucciones, sin embargo, los fabricantes tienden a agrupar las CPU en "familias", con conjuntos de instrucciones semejantes. Cuando se desarrolla una nueva CPU, su conjunto de instrucciones, por lo general, contiene las mismas instrucciones que tenia su predecesor, además de algunas nuevas. Esto permite que un programa, desarrollado para una cierta CPU, pueda utilizarse con nuevas CPU de la misma familia (esta estrategia de fabricación es conocida como **escalabilidad**).

1.4.2 LENGUAJE MÁQUINA Y LENGUAJE ENSAMBLADOR

La colección de instrucciones ejecutables para cada CPU, junto con su sistema de codificación, se llama **lenguaje máquina**. Este lenguaje es el primer lenguaje de programación que se utilizó y el único que entiende directamente el computador, aunque, al estar ligado a la circuitería, queda muy alejado de la forma en que solemos expresar los problemas. Sus principales características son:

- Las instrucciones son cadenas de ceros y unos.
- Los datos se utilizan por medio de las direcciones de memoria donde están almacenados y no admite identificadores, sino que el programador debe hacer una *asignación de direcciones de memoria* para todas las variables y constantes del programa.
- Las instrucciones realizan operaciones muy simples, dependientes de las posibilidades de la circuitería del procesador.

Los primeros programadores se comunicaban con los computadores interconexionando cables, más adelante lo hicieron a través de números binarios, pero éstas tareas eran tan plúmbeas, que pronto inventaron notaciones simbólicas

que facilitasen este trabajo. Sin embargo, estas notaciones tenían que ser traducidas manualmente a binario, por lo que el siguiente paso fue conseguir programas que tradujeran la versión simbólica de las instrucciones a la versión binaria. Así el programador escribe:

```
CAR A , M(16)
```

y el programa traductor convierte esta notación en:

```
000000010000
```

indicándole al computador que cargue en el registro A de la ALU, el contenido de la posición de memoria 16.

Este programa traductor se llama **ensamblador** y el lenguaje simbólico utilizado se conoce como **lenguaje ensamblador**. Este lenguaje requiere que el programador escriba una línea por cada instrucción que seguirá la máquina. Observemos que ambos lenguajes (ensamblador y máquina) fuerzan al programador a pensar directamente en las funcionalidades primarias de la propia máquina.

Siendo importante la aparición de los lenguajes ensambladores, la tarea de programación, con ellos, seguía siendo ardua (aunque para ciertas aplicaciones aun hoy, hay que recurrir a estos lenguajes). De hecho, los llamados ordenadores de primera generación se desarrollaron, a pesar de su buen funcionamiento electrónico, en medio de un cierto escepticismo, pues, al no estar conceptualizadas abstracciones más próximas al modo de razonar de las personas, se pensaba, no sin razón, que ni siquiera el propio creador de un programa alcanzaría su total comprensión, una vez escrito en términos binarios o en lenguaje ensamblador. Afortunadamente; los primeros programadores superaron tan negras perspectivas e idearon los lenguajes de programación de alto nivel (Fortran, Basic, Pascal, Lisp, Prolog, C, C++, etc.) mucho más amistosos, que se tratarán en un próximo apartado.

1.4.3 EJECUCIÓN DE UN PROGRAMA

Llegados aquí, vamos a profundizar un poco más en el funcionamiento de la máquina de von Neumann, analizando la forma como se ejecuta un programa ya almacenadoen memoria. Supongamos que está almacenado a partir de la posición *i* de memoria y que la ejecución indica “pasar el control a la posición *i* de memoria”. A partir de ese momento, la UC repite sucesivamente las siguientes fases:

a) Lleva de la memoria a la UC la instrucción que está en la posición *i*, y cambiar el valor de *i* por *i*+1.

b) Interpreta el código de operación de la instrucción y, según sea éste y las señales de estado, envía señales de control a las unidades y circuitos que deben

intervenir para ejecutar la instrucción. Y efectua las operaciones (*microoperaciones*) que ésta implica. Volver a la fase *a*)

La fase *a)* es igual en todas las instrucciones, siendo la fase de captación de instrucción, y la *b),* es específica de cada instrucción, siendo la fase de ejecución de la misma. En el caso de que la ejecución de la instrucción implique saltar a otra, a la posición *m* por ejemplo, (alterándose por tanto el orden secuencial), la UC hace, en la fase de ejecución de la instrucción de salto, que se cambie *i* por *m*, de forma que en la siguiente fase de captación se ejecuta la instrucción que está en *m*, por ser éste el valor actual de *i*.

A modo de ejemplo, vamos analizar un caso sencillo. Supóngase que se dispone de un computador, con un teclado como unidad de entrada, una impresora como unidad de salida (Ver Figura 1.9), y una CPU cuyo lenguaje máquina contiene, entre otras, las instrucciones siguientes (validas para cualquier CPU):

- ENTrada de información que se transfiere a la posición *m* de memoria, que se indica en el campo de dirección de la propia instrucción. Abreviadamente: ENT M(m),E.

- SALida de información procedente de la posición *m* de memoria.:
  SAL S,M(m).

- MEMorizar en la posición m de memoria el contenido de la ALU. Abreviadamente: MEM M(m),A.

- CARgar en el registro A de la ALU, el contenido de la posición m de memoria.: CAR A,M(m).
- SUMar el contenido del registro A, con el contenido de la posición *m* de memoria, almacenándose el resultado en A.: SUM A,M(m).
- FIN indica a la UC que debe acabar el procesamiento de ese programa.

Se desea escribir un programa que sume dos números. Analizando el repertorio de instrucciones de que se dispone, hay que ejecutar los siguientes pasos: 1) Leer ("dar entrada") los dos números llevándolos a la memoria, 2) Llevar uno de los sumandos a la ALU, y sumarlo con el otro , 3) El resultado almacenado en la ALU, llevarlo a la memoria y 4) Escribirlo a través de la unidad de salida.
(Ver diagrama de la Figura 1.9.)

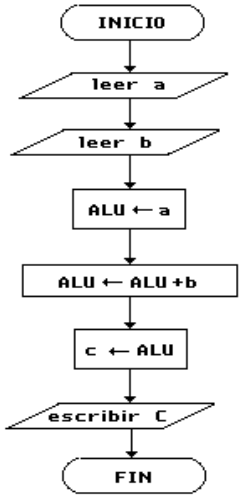

Fig. 1.9. *Diagrama de las operaciones a realizar para el programa de sumar*

Además, el programador en lenguaje máquina, ha de determinar las posiciones de memoria que va a utilizar. En el ejemplo vamos a seleccionar:

Primer sumando,	en posición 16
Segundo sumando,	en posición 17
Suma,	en posición 18

El programa es el siguiente:

(1) Leer el primer dato y llevarlo a la posición 16.
ENT M(16),E 00100 001 0000

(2) Leer el segundo dato y llevarlo a la posición 17.
ENT M(17),E 00100 001 0001

(3) Llevar a la ALU el primer dato (que está en *m*=16).
CAR A,M(16) 00000 001 0000

(4) Sumar el contenido de la ALU con el segundo sumando (que está en *m*=17).
SUM A,M(17) 11000 001 0001

(5) Llevar el resultado de la ALU a la posición *m*=18.
MEM M(18),A 00010 001 0010

(6) Escribir el resultado que está en *m*=18.
SAL S,M(18) 00110 001 0010

(7) FIN

Después de describir cada instrucción hemos escrito su abreviación y su correspondiente instrucción máquina, cuyo procesador suponemos esta en condiciones de manejar instrucciones de 12 bits: los cinco primeros son el código de operación y los siete restantes constituyen el campo de dirección ó la dirección de memoria que interviene en la instrucción.

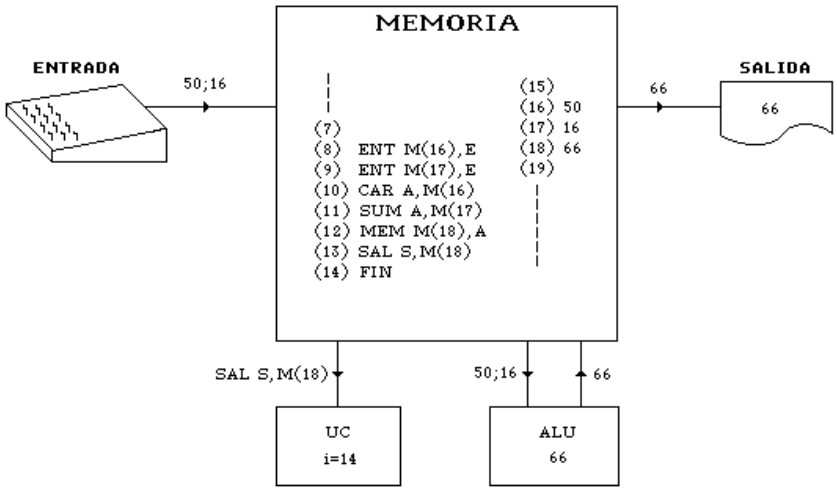

Fig. 1.10 *Contenido de la memoria después de ejecutado el programa en la computadora del ejemplo (Por simplicidad , se representan las instrucciones abreviadamente y los datos en decimal, aunque realmente son ceros y unos).*

Supóngase que el programa se carga en memoria a partir de la dirección *i*=8 , y se indica a la UC que pase su control a la instrucción que está en i=8, con el objetivo de sumar 50 + 26. Los pasos de la ejecución pueden seguirse fácilmente con ayuda de la Figura 1.10:

(1.a) La UC capta la instrucción que está en $i = 8$ y cambia a $i = 8 + 1 = 9$.
(1.b) La UC interpreta el código de la instrucción, 00100. y genera las señales de control para que el dispositivo de entrada lea un dato y sea escrito en la posición m = 001 0000 o 16 en decimal. Si el dato tecleado es 50, al final de la ejecución de la instrucción este valor en binario (0000 0011 0010) queda almacenado en la posición 16.

(2.a) La UC capta la instrucción que está en $i = 9$ y cambia a $i = 9 + 1 = 10$
(2.b) La UC interpreta el código de la instrucción captada, 00100. y genera las mismas señales de control que en la instrucción anterior. En este caso m =

001 0001 (17 en decimal) y el segundo dato, 26 queda grabado en binario (0000 0001 1010) en la posición 17.

(3.a) La UC capta la instrucción que está en $i = 10$ y cambia a $i = 10 + 1 = 11$.
(3.b) La UC interpreta el código de operación de la instrucción 00000, generando las señales de control necesarias para que se lea el contenido de la posición m = 001 0000 (16 en decimal) y sea llevado a un registro de la ALU. Ningún contenido de la memoria cambia.

(4.a) La UC capta la instrucción que está en $i = 11$ y cambia a $i = 11 + 1 = 12$.
(4.b) La UC interpreta el código de operación de la instrucción captada, en este caso 11000 y genera las señales de control para sumar el contenido de la ALU (0000 0011 0010, en decimal 50) con el contenido de la posición m = 001 0001 de la memoria (que es 0000 0001 1010, en decimal 26). El resultado de la suma (0000 0011 0010 + 0000 0001 1010 = 0000 0100 1100; en decimal 76), queda en el mismo registro de la ALU.

(5.a) La UC capta la instrucción que está en $i = 12$ y cambia i a $i = 12 + 1 = 13$.
(5.b) El código de operación, en este caso 00010, es interpretado por la UC, dando ésta las señales de control para que el contenido de la ALU (0000 0100 1100) se grabe en la posición m = 0000 0001 0010 (18, en decimal) de la memoria. El resultado de la suma queda, en la posición 18

(6.a) La UC capta la instrucción que está en $i = 13$. y cambia a $i = 13 + 1 = 14$.
(6.b) El código de operación, ahora 00110, se interpreta por la UC y ésta genera las señales de control para leer de memoria, el contenido de la posición m = 0000 0001 0010 (18 en decimal) y llevarlo a la unidad de salida. Allí el valor transferido, es convertido de forma que en la impresora se disparan los elementos necesarios para escribir 76.

(7) La UC capta la instrucción de FIN y acaba el procesamiento (La Figura 1.10 muestra el contenido de la memoria, al finalizar la ejecución).

Hay que insistir en que este problema, escrito en un lenguaje de alto nivel, habría resultado mucho más sencillo, aunque una vez traducido el ordenador trabajaría con un código muy parecido al que acabamos de usar.

1.5. CONCEPTO ACTUAL DEL COMPUTADOR

1.5.1 DEFINICIÓN ACTUAL

Aunque no está en la óptica de este texto, el profundizar en las nuevas arquitecturas que presentan los computadores en la actualidad, si parece necesario poner en evidencia que el concepto de máquina de von Neumann tal como lo hemos tratado hasta ahora, (como un dispositivo de procesamiento individual controlado por un único programa), aunque adecuado, debe ser objeto de una definición más amplia, para llegar al actual concepto de computador. A modo de reflexión, hay que hacer notar que muchos computadores, contienen más de un procesador, requieren para obtener varios programas trabajando de forma aparentemente simultánea y que una computadora moderna puede tener gran cantidad (cientos) de unidades de entrada o salida, próximas o remotas.Vamos a dar una definición operativa de computador que amplie la máquina de von Neumann:

Un **computador** es una colección de recursos, que incluyen: dispositivos de proceso electrónico digital, programas almacenados y conjuntos de datos, que, bajo el control de programas, produce salidas, almacena, recupera y procesa datos, pudiendo también transmitirlos y recibirlos hacia y desde otros computadores. En todo computador hay que distinguir dos soportes:

- El **soporte físico** (hardware en inglés) que es la máquina en sí: el conjunto de circuitos electrónicos, cable, armario, dispositivos electrónicos, etc.

- El **soporte lógico** (o software, neologismo inglés que contrapone soft (blando) con hard (duro), - logicielle en francés) que es el conjunto de programas que dirigen el funcionamiento del computador: sistema operativo, programa de usuario, etc.

Notemos que en la práctica la distinción física/lógica es más difusa de lo que aparentemente parece, así en algunos computadores las multiplicaciones se hacen directamente por hardware, mientras que en otros se hace en forma de sumas múltiples, controladas por software. Además, parte del software está almacenado permanentemente en forma de ROM, que se conoce como **firmware** y que es un ente intermedio entre el hardware y el software.

1.5.2 PARÁMETROS BÁSICOS DE LA MÁQUINA

Existen varias características, relacionadas con el funcionamiento de las distintas unidades funcionales.que determinan el tipo y velocidad del ordenador central. En este sentido, hay que saber que la UC contiene un reloj o generador de pulsos, que sincroniza todas las operaciones elementales de la computadora. El período de esta señal se denomina **tiempo de ciclo**, cuyo orden de magnitud es de nanosegundos. La frecuencia del reloj (que suele darse en millones de

ciclos/segundo o Mhz) es un parámetro que en parte determina la velocidad de funcionamiento de la computadora.

La **longitud de palabra** determina, la precisión de los cálculos, la característica de los buses y la variedad de instrucciones respecto a esta velocidad, sabemos que cada CPU tiene un **conjunto de instrucciones,** determinado. Hoy en día, hay dos grandes familias: Procesadores con un amplio conjunto de instrucciones (CISC = Complex Instruction Set Computer) y Procesadores con un conjunto de instrucciones reducido, pero optimizadas para que éstas se ejecuten más rápido (RISC = Reduced Instruction Set Computer).

Es frecuente, en computadores convencionales, que la gestión de las operaciones con números reales (que arrastran un gran numero de decimales y por tanto requieren bastantes ciclos de ejecución en la ALU) se realicen por hardware con circuitos integrados auxiliares conocidos como **coprocesador matemático**. El que un computador disponga o no de coprocesador matemático varía enormemente su velocidad en aplicaciones que requieran gran número de operaciones con números reales. La tendencia actual es que el coprocesador matemático se integre dentro de la propia CPU y no se hable explícitamente del mismo (por ejemplo el 486 y el Pentium). Igualmente el tipo de **bus** (o buses) que conecta los distintos subsistemas del computador condiciona su potencia, especialmente en la adecuación del número de líneas del bus con el tamaño de palabra.

Otros de los parámetros significativos son las capacidades de memoria principal y secundaria. La **capacidad de la memoria principal**, indica el tamaño de los programas (instrucciones+datos) que el computador puede procesar simultáneamente. La memoria secundaria sirve tanto como 'ampliación' de la memoria principal (dentro de la concepción de jerarquía de memorias ya indicada), como almacenamiento permanente de programas y datos. Al considerar todos estos parámetros, es importante recordar, que para acceder a instrucciones o datos, es imprescindible el poder direccionar todas y cada una de las palabras de la memoria principal. En una primera aproximación, esto se hace asignando a cada posición un número en base 2; así con n bits podemos direccionar 2^n palabras. Sin embargo, hay que analizar lo que supone esta función de direccionamiento, a medida que la memoria crece; así para direccionar 64 Kpalabras necesitamos 16 bits y para 1 Mpalabra son necesarios 20 bits, con lo que al crecer la memoria se hace imprescindible acudir a técnicas de direccionamiento, para poder acceder a un número grande de palabras con un número relativamente pequeño de bits.

Junto con la **capacidad de la memoria secundaria**. de los distintos periféricos, el **tiempo de acceso** (para lectura y para escritura en el periférico) es un parámetro importante para determinar la potencia de un computador. Además de cuantos y cuáles sean los **periféricos** que puede incorporar un determinado computador (p.e.

los computadores personales disponen de *slots* o ranuras sobre el bus para 'clavar' nuevos componentes; el número y tipo de los *slots* establece las posibilidades de ampliación de periféricos).

1.6. DEL COMPUTADOR A LA PROGRAMACIÓN

Una vez visto el funcionamiento del computador, hay dos consecuencias a sacar sobre la naturaleza de esta máquina:

1) Desde el punto de vista de los datos, éstos se presentan en forma de estructuras binarias de tamaño predefinido. El computador sólo trabaja con estructuras aparentemente muy limitadas, que son las que incorporan los registros y las palabras de la memoria en las que se almacenan valores binarios.

2) Desde el punto de vista algorítmico, el número de operaciones que puede llevar a cabo viene prefijado y limitado por un conjunto de instrucciones dado,materializables a través de circuitos electrónicos.

La pregunta que surge es inmediata, ¿cómo puede una sola máquina representar datos tan complejos y diferentes como números, textos,vectores y otras estructuras más elaboradas, con solo la utilización de bits y al tiempo, resolver un espectro tan amplio de problemas con la única utilización del repertorio del lenguaje máquina?. En otras palabras, ¿cómo supera el computador éstas limitaciones, tanto en materia de estructura de datos como de algoritmos?
La respuesta la indicó el propio von Neumann: El computador es una máquina de propósito general, cuya naturaleza queda transformada por el programa que se le proporciona, de forma que en un momento dado, un caudal de información constituye un conjunto de datos procesados por el programa, y a continuación otro caudal de información se interpreta como instrucciones ejecutables.

1.6.1 LOS DATOS

Los datos son los objetos sobre los que opera un ordenador; sabemos que éste tiene capacidad de procesar datos de tres tipos básicos: numéricos, lógicos y caracteres alfanuméricos (no deja de ser sorprendente que, estando el cálculo científico en el origen del ordenador, sus actuales aplicaciones sean mucho más amplias que las derivadas del manejo de números). Los primeros representan las diferentes clases de números, enteros y reales, con distintas posibilidades de rango de magnitud y precisión. Los datos lógicos o booleanos son aquellos que pueden tomar valores ciertos o falsos y que nos servirán para tomar decisiones en función de que se cumplan o no determinadas condiciones. Los datos alfanuméricos son, a

grandes rasgos, los que puede producir un teclado habitual. Retengamos la idea, de que el ordenador está en condiciones de representar y diferenciar internamente estos tipos de datos simples, que a su vez pueden aparecer de forma aislada o agrupados en forma de una estructura. Como tendremos ocasión de ver, existen una importante variedad de datos estructurados, como por ejemplo:

cadenas de caracteres.
vectores y matrices (habitualmente de datos numéricos).
registros y ficheros para el almacenamiento secundario.

1.6.2 LENGUAJES DE PROGRAMACIÓN DE ALTO NIVEL

Como ya anunciamos, ante los serios inconvenientes de los lenguajes máquina y ensamblador desde los inicios de los computadores actuales, se han desarrollado los llamados **lenguajes de alto nivel**, con algunas características muy apreciadas: están más cercanos a nuestra forma de resolver problemas, son relativamente independientes de la máquina, (una sola de sus instrucciones equivale a varias instrucciones de lenguaje máquina), facilitan enormemente el proceso de programación, acercándolo, en la medida de lo posible, a los lenguajes que los humanos utilizamos habitualmente, etc.

Las sentencias o frases que se pueden construir a partir de la sintaxis de un lenguaje de alto nivel son de dos tipos: *imperativas (o instrucciones*) que indican acciones a tomar por el computador, o *declarativas*, que proporcionan información sobre determinadas circunstancias del programa. Así la sentencia:

```
float Base
```

es declarativa, ya que no implica una acción visible dentro del contexto del programa, sino que indica al programa traductor, que la variable Base se va a utilizar como un dato numérico en coma flotante, de simple precisión para que lo tenga en cuenta, a efectos de representación interna en la memoria. Sin embargo, la sentencia:

```
Area = Base * Altura
```

es una instrucción pues produce la ejecución de una operación, en este caso, multiplicar el valor de Base por el de Altura, y el valor del resultado asignárselo a la variable Area.

Notemos que comparados con los lenguajes máquina, los de alto nivel permiten utilizar variables, símbolos y términos más parecidos a los usados por los humanos, en la resolución de problemas. Este proceso de abstracción nos permite pensar con independencia de las particularidades con las que trabaja la máquina, sin ello resultaría casi imposible crear programas de tamaño no trivial. El programador a la hora de resolver un problema, establece una jerarquía de abstracciones, hasta que el

computador puede encargarse de su traducción y posterior ejecución. Con ello tenemos una doble ventaja, por un lado que el programador piense en una determinada estructura de datos sin tener que preocuparse en cómo se organizan éstos realmente en memoria y por otro, que pueda utilizar operaciones, habituales para nosotros, que serán finalmente ejecutadas por el procesador. Por tanto los lenguajes de alto nivel deben permitir especificar en forma abstracta tanto estructuras de datos complejas como los distintos pasos que constituyen un algoritmo que implementa un programa. Esta idea constituirá el hilo conductor de este libro, obedeciendo a una de las claves de la Programación.

PROGRAMA = ESTRUCTURA DE DATOS + ALGORITMOS

Esto significa que todo proceso de programación, además de contar con un algoritmo que resuelva el problema planteado, ha de tener conceptualizada la forma como organizar los datos antes y después de ser procesados. De hecho, ambas cuestiones están interrelacionadas: una estructura de datos adecuada puede simplificar el diseño de un algoritmo, y un algoritmo menos adecuado que otro puede implicar manejar estructuras de datos más complejas.

La evolución del arte y la tecnología de la programación es muy intensa y sigue en pleno desarrollo, en la actualidad. Veamos algunos hitos: **programación clásica**, representada por las primeras versiones de FORTRAN y BASIC, se caracteriza por el mero uso de una secuencia de ordenes y bifurcaciones; **programación modular**, añade un método de diseño que permite resolver un problema, mediante su descomposición en problemas más simples o **módulos**; **programación estructurada** (representada por PASCAL y C entre otros) supone. además un conjunto de técnicas que permiten desarrollar programa fáciles de escribir, verificar, leer y mantener. La naturaleza de todos ellos es **procedural**, es decir, está basada en procedimientos que hacen algo concreto, como escribir un mensaje en pantalla, obtener datos desde el teclado o ejecutar un proceso algorítmico, de forma que un programa puede incorporar fácilmente cientos de procedimientos individuales. En otros capítulos, nos extenderemos sobre esta variedad de técnicas de programación, limitémonos ahora, a añadir la llamada **programación orientada a objetos**, cuyos programas se construyen ensamblando partes denominadas objetos, que es un tipo especial de dato, en un intento de emular el mundo real, que se compone de objetos que interaccionan entre sí. La programación orientada a objetos presenta ventajas importantes, pero sus conceptos no son fáciles y en ningún caso accesibles, sin un buen dominio de la programación estructurada.

Ante la actual babel de lenguajes de programación, el objetivo de este libro es ser compatible con la mayor parte de los procedurales, sin tomar partido por ninguno en particular, (es en otros textos y cursos donde se debe aprender a dominar algún lenguaje de programación específico). No obstante, como ejemplos significativos,

(desde la doble óptica de los años noventa y de unos estudios de ciencias e ingeniería), vamos a tratar de rastrear, en la medida de lo posible, cuatro de los más populares lenguajes, (alguno de los cuales debe de ser aprendido en paralelo con este curso), que por orden cronológico son:

FORTRAN. Pasa por ser el primer lenguaje de alto nivel que contó con un compilador eficiente. Su propio nombre, acronismo de "FORmula TRANslation", señala que fue diseñado con el objetivo de ser utilizado para calculo científico. Desarrollado en 1957 por un pionero de la informática John Backus y un equipo de científicos de IBM, su primera estandarización data de 1966 y desde entonces han aparecido dos mejoras sustantivas en 1977 y 1990.
Aunque su panorámica se reduce casi exclusivamente al calculo científico y técnico, es tal la cantidad de software escrito en este veterano lenguaje, que su uso en los ambientes de calculo intensivo esta muy extendido y es difícil hacer un pronóstico acerca de su sustitución por otros lenguajes de alto nivel más sofisticados.

BASIC. Fue desarrollado en 1964 por John Kemeny y Thomas Kurtz en el Darmouth College, comenzó siendo una herramienta para enseñar programación como indica su acronismo (Beginners All-purpose Symbolic Instruction Code). Su simplicidad le hizo especialmente popular, siendo el primer lenguaje de alto nivel que se utilizó con la aparición de los primeros ordenadores personales, incluso anteriores al primer PC de IBM . Existen versiones del Basic especialmente potentes y accesibles como Turbo-Basic, Quick-Basic, GW-Basic y Visual-Basic (que ya incorpora alguna de las características y métodos orientados a objetos).
A pesar de su popularidad, el BASIC no ha cuajado completamente a nivel científico y profesional, ya que no tiene un gran repertorio de herramientas y sus compiladores no producen archivos ejecutables que sean tan compactos, veloces y eficientes como los producidos por otros lenguajes. Sin embargo, su simplicidad, puede llegar a satisfacer las necesidades de muchos usuarios.

PASCAL. Desarrollado por el suizo Niklaus Wirth en 1971, en honor del sabio francés del siglo XVII del mismo nombre, es el primer lenguaje específicamente pensado para la programación estructurada. Sus puntos más interesantes son sus impecables medios para revisar el tipo de datos y para controlar el flujo de ejecución del programa. Son muchos los libros y cursos que lo utilizan como lenguaje de primera elección para aprender programación estructurada. Como no podía ser menos, existen muchas extensiones de PASCAL y en particular las últimas, orientadas a objetos. Los puntos negativos para los profesionales de la programación hay que buscarlos en su filosofía excesivamente académica, lo que conduce a hacerlo un poco tedioso.

C y C++. El lenguaje C está considerado una especie de pura sangre por muchos programadores, ya que los programas escritos en C producen un código ejecutable

especialmente veloz y eficiente, pues con él la máquina puede hacer casi todo lo que su hardware le permite realizar. Fue desarrollado por Brian Kernighan y Dennis Ritchie a principios de la década de los 70 en los Laboratorios Bell; el segundo de sus autores tiene también la importante paternidad del sistema operativo UNIX. De hecho C y UNIX están íntimamente ligados ya que el código fuente C se puede trasladar de una máquina UNIX a otra. El costo de un lenguaje tan útil reside en que no es de los de más fácil aprendizaje, lo que puede llevar a cierto grado de desmoralización a los principiantes, que deben superar rápidamente este estado de ánimo.

A principios de los 80, otro científico de los Laboratorios Bell, Bjarne Stroustrup, introdujo la orientación a objetos en C. El resultado fue un lenguaje poderoso y eficiente pero no sencillo de aprender, cosa que sólo se puede hacer una vez dominado el C. Hoy, a mitades de los noventa, el C++ aparece como un lenguaje candidato a ser utilizado en muchas aplicaciones informáticas.

1.6.3 ELEMENTOS BÁSICOS DE UN LENGUAJE DE ALTO NIVEL

Un programa consta de una sucesión de sentencias, escritas en lineas sucesivas.Veámos cuales son los **elementos** que están presentes en cualquier programa escrito en un lenguaje de programación de alto nivel.

Palabras reservadas: Conjunto de colecciones de caracteres con significado especial, que el traductor del lenguaje interpreta según su significado. Estas palabras, permitirán formar sentencias, tanto declarativas como imperativas. Cada lenguaje tiene un conjunto de palabras reservadas. Veamos algunos ejemplos de palabras reservadas tomadas de cada lenguaje:

FORTRAN	BASIC
PROGRAM, DATA COMPLEX, IF	dim, step loop, select
PASCAL	**C**
program, procedure begin, end	`main, switch` `void, float`

Constantes y Variables: Un programa contiene ciertos objetos o conjuntos de datos que no deben cambiar durante su ejecución, estos valores se llaman **constantes**, por contra una **variable** es un objeto o conjunto de datos cuyo valor puede cambiar durante el desarrollo del algoritmo o la ejecución del programa. Las variables tienen un nombre, conocido como **identificador**, que suele constar de varios caracteres alfanuméricos de los cuales el primero normalmente es una letra. Una variable por tanto se identifica por dos atributos: **nombre** y **tipo** Este último

define el conjunto de posibles valores que puede tomar la variable,a la vez que determina el tipo de dato que soporta su correspondiente representación interna.

Expresiones: Combinaciones de constantes, variables, símbolos de operación, caracteres especiales, etc., análogas a las utilizadas en notación matemática. Cada expresión toma un valor que se determina tomando los valores de las variables y constantes implicadas y la ejecución de las operaciones indicadas. Una expresión consta de operandos y operadores. Según sea el tipo de objetos que manipulan, pueden ser:

- Aritméticas. Donde las variables y constantes son numéricas y las operaciones son aritméticas, una vez ejecutadas, su valor numérico es:

  Ej.: x + (b+5) + z * z

- Relacionales: Utiliza signos de comparación (<, >, =, etc.) con valores de tipo numérico o carácter para producir un valor lógico (verdadero o falso),

  Ej.: (A-2) < (B-4)

- Lógicas: Combina operadores lógicos (AND, OR, NOT) que operarán sobre valores lógicos, de acuerdo con sus tablas de verdad.

  Ej.: (A<B) OR (A>B)

- De Carácteres: Que involucra cadenas de caracteres alfanuméricos.

  Ej.: Unión de dos cadenas.

Etiquetas: Ciertas instrucciones de un programa utilizan **etiquetas** para referirse a una línea determinada de un programa, bien para realizar un salto a dicha línea, bien porque dicha línea contiene información adicional para una instrucción. La etiqueta puede tener un nombre (situado al principio de la línea que está etiquetando) o ser simplemente el número de la línea en cuestión.

Comentarios: Con el único objetivo, de hacer más comprensible la lectura del programa por parte de un humano, esta prevista la inclusión de comentarios a la largo de su desarrollo, que hay que marcar debidamente, para que el ordenador las ignore.

FORTRAN	**BASIC**
C comentario * comentario	´ comentario REM comentario
PASCAL	**C**
{comentario} (*comentario*)	/* comentario */ // comentario en algunos C

Asignación: Es la acción por la cual le damos (asignamos) valores a una variable, la representaremos por el signo ←. Al escribir A←5 significamos que A toma el valor de 5 como real. La acción de asignar es destructiva, ya que el posible valor que tuviera la variable antes de la asignación se pierde y se reemplaza por el nuevo valor. Se debe pensar en la variable como en una posición de memoria, cuyo contenido puede variar mediante instrucciones de asignación (un símil es un marcador -de un campo de fútbol-, donde su contenido variará según el movimiento del partido; al poner un número en uno de los casilleros, desaparece el que hubiera anteriormente). Puesto que cada variable tiene un tipo determinado ,se entenderá que no se pueda (o no se deba) asignar valores a una variable de un tipo diferente del suyo y así se presentará un error si se trata de asignar valores de tipo carácter a una variable numérica o un valor numérico a una variable tipo carácter (¡ imaginemos que en nuestro marcador del campo de fútbol pusiéramos una letra A en el casillero del tanteo del equipo local !).

El computador ejecuta la acción de asignar en dos pasos, en el primero de ellos, el valor de la expresión al lado derecho del operador se calcula, obteniendo un valor de un tipo específico. En el segundo paso este valor se almacena en la variable, cuyo nombre aparece a la izquierda.

Veamos un ejemplo:
¿Cuál es el valor de las variables A, B y AUX antes y después de cada paso?

1. A ← 10
2. B ← 13 + 7
3. AUX ← A + 15
4. A ← B
5. B ← A + 4
6. AUX ← AUX + 1

Antes de la ejecución de las instrucciones, el valor de A, B, y AUX es indeterminado, es decir, el contenido de las posiciones de memoria correspondientes a ellas es desconocido.

1. A toma el valor 10
2. se evalúa la expresión 13+7 y B toma el valor resultante, 20
3. se evalúa la expresión con el valor de A en ese momento (10) más 15, AUX toma el valor 25 (A no se ve afectada)
4. A toma el valor de B en ese momento, o sea 20 (B no cambia)
5. se evalúa la expresión con el valor de A en ese momento (20), más 4, 24; B toma el valor resultante 24 (A no se ve afectada)

6. se evalúa la expresión con el valor de AUX en ese momento (25) más 1; AUX toma el valor resultante 26.

Nótese que la operación de asignación, pese a que en algunos lenguajes se exprese con el signo = no es una igualdad matemática. Por ejemplo, la instrucción 4 del ejemplo anterior escrita en lenguaje C sería:

```
A=B;
```

lo cual no quiere decir que a partir de ese momento A y B vayan a ser siempre iguales. Igualmente, la instrucción 6 del ejemplo en lenguaje C

```
AUX=AUX+1;
```

tiene sentido como asignación, pero sería un completo contrasentido como igualdad matemática.Veámos la asignación en los distintos lenguajes:

FORTRAN	BASIC
V = E	V = E
PASCAL	**C**
v := e	v = e

Entrada: Las acciones de entrada permiten obtener determinados valores a partir de un periférico y asignarlos a determinadas variables. Esta entrada se conoce como operación de lectura. Y se representa por el formato:

leer (lista de variables de entrada)

Por ejemplo, la instrucción: leer NUMERO, HORAS, TASA.

Lee del terminal los valores NUMERO, HORAS y TASAS, almacenándolos en la memoria; con la correspondiente acción de asignación a las tres variables. Si los tres números que se teclean en respuesta a la instrucción son: 12325, 32,1200, y equivale a la ejecución de las instrucciones.

NUMERO ← 12325
HORAS ← 32
TASA ← 1200

Veamos como se realiza la entrada por teclado en algunos lenguajes:

FORTRAN	BASIC
READ f, numero, horas, tasa	INPUT numero, horas, tasa

f: etiqueta de formato (* = formato libre)	
PASCAL	**C**
readln(numero, horas, tasa)	`scanf("%d %d %d", &numero, &horas, &tasa)`

Salida: Las acciones de salida, permiten transferir a un periférico los resultados obtenidos por el computador. Esta operación se conoce como escritura y se representa por el siguiente formato:

escribir (lista de variables de salida)

Como ejemplo, veamos el resultado de la ejecución de las siguientes instrucciones:

A ← 100
B ← 200
C ← 300
escribir A, B, C

Se visualizarán en pantalla o imprimirán en impresora los valores 100, 200 y 300

Veámos como se realiza la salida por pantalla en algunos lenguajes:

FORTRAN	**BASIC**
WRITE f, numero, horas, tasa PRINT f, numero, horas, tasa f: etiqueta de formato (* = formato libre)	PRINT numero, horas, tasa
PASCAL	**C**
writeln(numero, horas, tasa)	`printf("%d %d %d",numero, horas,tasa)`

1.6.4 ORGANIZACIÓN DE UN PROGRAMA

Un programa escrito en un lenguaje procedural de alto nivel consta de un **encabezamiento** (opcional en algunos lenguajes) donde se indica información general del programa, tal como su nombre y algunas circunstancias que el traductor debe saber, y un **cuerpo**, donde se desarrolla el programa. En el cuerpo, primeramente debe hacerse la **declaración de variables** por la cual se indica que

variables van a emplearse en el subprograma y cual es el nombre y el tipo de cada una de ellas. A continuación aparecen las sentencias imperativas que, por medio de las variables que hayamos declarado representan el algoritmo.

La estructura general que se espera que adopte un programa, es la siguiente:

Identificador del programa o módulo
{sección de declaraciones}
inicio
    {datos de entrada}
    {sentencias imperativas, que ejecutan el algoritmo correspondiente}
    {datos de salida}
fin

Aunque es ligeramente prematuro, el lector deberá ser capaz de seguir sin dificultad la organización de los programas resultantes de codificar el problema de hallar el volumen de un cilindro a partir de su radio y altura en los cuatro lenguajes utilizados.

FORTRAN

```
  PROGRAM volcilin
C Hallar volumen de un cilindro
  REAL radio, altura, volumen
  DATA PI /3.14.16/
  READ radio,altura
  volumen=PI*radio*radio*altura
  WRITE volumen
  END
```

BASIC

```
REM radio, altura, volumen
REM si no se declaran son SINGLE
PRINT "Radio y altura : "
INPUT radio, altura
volumen=3.1416* radio^2 *altura
PRINT "el volumen es";volumen
END
```

PASCAL

```
program volcilin(input,output);
const PI = 3.1416;
(*Hallar volumen de un cilindro *)
var radio, altura, volumen: real;
begin
  write("Radio y altura : ");
  readln(radio, altura);
  volumen=PI*radio*radio*altura;
  writeln("el volumen es",volumen);
end.
```

C

```
#include <stdio.h>
#define PI 3.1416
/*Hallar volumen del cilindro */
main(){
  float,radio, altura, volumen;
  printf("Radio y altura : ");
  scanf("%f%f",&radio, &altura);
  volumen=PI*radio*radio*altura;
printf("volumen=%d\n",volumen);
}
```

1.6.5 TRADUCCIÓN DE PROGRAMAS

Una vez escrito un programa en lenguaje de alto nivel, para lo que se habrá empleado un **editor de texto**, tenemos lo que se llama un **programa fuente**. Para poderlo ejecutar, debemos recurrir a un programa traductor, que lo convierta en lenguaje máquina. Aunque el proceso de traducción puede hacerse de varias formas, lo más frecuente es hacerlo por medio de un compilador. Un **compilador** es un programa que traduce en su totalidad un **programa fuente**, escrito en un lenguaje de alto nivel, a un **programa objeto**, escrito en lenguaje máquina.

Una vez obtenido el programa objeto, raramente puede ejecutarse por sí mismo, pues habitualmente presenta algunos cabos sin atar, que deben conectarse a otros programas del sistema para poder ejecutarse. Esta tarea corre a cargo a su vez de un programa llamado **montador** (linker) que genera el llamado **programa ejecutable**.

Finalmente hay que introducir la totalidad del programa en la memoria. De esta tarea, se encarga un programa denominado **cargador** (loader) que sitúa las instrucciones en posiciones consecutivas de la memoria principal a partir de una determinada posición (con lo que tenemos el concepto ya conocido de programa almacenado) El proceso completo desde el código fuente hasta el programa almacenado se muestra en la Figura 1.11.

Fig. 1.11. *Preparación de un programa para su ejecución*

Las distintas versiones del programa (fuente, objeto y ejecutable), se suelen guardar en archivos, en un dispositivo de almacenamiento masivo, para ser usado posteriormente sin necesidad de volver a realizar la traducción. Por ello se habla de archivos fuente, archivos objeto y archivos ejecutables. Si queremos ejecutar un programa y disponemos de su archivo ejecutable, en la mayoría de casos será suficiente con escribir su nombre: el programa se cargará en la memoria como programa almacenado y podrá empezar a ejecutarse, sin necesidad de pasar por la compilación y el montaje.

1.7. SISTEMA OPERATIVO Y PROGRAMAS DEL SISTEMA

A medida que maduraba la programación, se vió lo ventajoso de reutilizar programas, se comenzó a almacenar programas o subprogramas cuya utilización es frecuente y básica para el trabajo del ordenador, (ej: los programas relacionados con la gestión de los distintos tipos de periféricos). Este fue el germen de los

primeros sistemas operativos: programas que gestionan las funciones básicas del computador, simplificando la tarea del programador que puede soslayar ciertos detalles de la máquina que quedan a cargo del sistema operativo. Hoy en día sería imposible o extremadamente difícil utilizar un computador sin disponer de un sistema operativo (S.O.).

Los **sistemas operativos** son programas que gestionan los recursos de un computador en beneficio de los programas que se ejecutan en esa máquina. Un sistema operativo es en sí mismo un programa, pero forma parte integral de la máquina y es tan importante conocerlo como conocer sus componentes físicos.
Sus tareas principales son:

- Proporcionar al usuario una interfaz que le permita comunicarse con el computador.
- Administrar los dispositivos de hardware del computador
- Administrar y mantener los sistemas de archivo de disco.
- Dar soporte a otros programas y a los programadores aislando los detalles físicos

Cuando se enciende un computador, éste ejecuta primero una autoprueba para identificar los dispositivos que tiene conectados, contar la cantidad de memoria disponible y hacer una rápida comprobación de la misma. A continuación, busca un sistema operativo que le indique como interactuar con el usuario y como usar los dispositivos. Cuando lo encuentra lo carga en memoria y mantendrá parte del mismo en su memoria hasta que se apague el computador.

Junto al sistema operativo hay un conjunto de programas conocidos como **programas del sistema** necesarios para el funcionamiento habitual de la máquina (algunos de estos programas se suministran con el sistema operativo y en ocasiones se consideran parte integrante del mismo). El software del sistema está constituido, además del sistema operativo, por:

- Utilidades generales de programación, que contiene programas o utilidades que facilitan la construcción de las aplicaciones de los usuarios, sea cual sea la naturaleza de éstas. Incluye herramientas tales como:

- Traductores (ensambladores, compiladores, etc.).
- Editores de textos
- Rastreadores/depuradores de errores de programación.
- Sistemas de Gestión de archivos
- Administrador de bibliotecas de programas.
- etc.

- Programas de diagnóstico, generación y mantenimiento. Que utilizan los responsables del mantenimiento y puesta al día del hardware y del software (incluida la generación y mantenimiento del propio SO). Con estos programas se

pretende por ejemplo localizar automáticamente las averías de un determinado dispositivo ó circuito, ó las causas de un mal funcionamiento de algún modulo del SO.

1.8. TIPOS DE COMPUTADORES

Hoy en día existe una gran variedad de computadores de uso general, que varían en tamaño capacidad y rendimiento. La propia evolución tecnológica dificulta la clasificación de los diferentes tipos de computadores, aunque se siguen distinguiendo una serie de grandes familias:

SUPERCOMPUTADORES. Un **supercomputador** es el computador más potente disponible en un momento dado. Formado por procesadores múltiples, trabajan en paralelo accediendo a memorias de grandes dimensiones. Para el control de la entrada/salida de datos y de los canales de comunicaciones externos utilizan procesadores especiales, puesto que los requisitos de velocidad son muy elevados (los llamados front-end). Se suelen utilizar para efectuar cálculos complejos que requieren grandes cantidades de datos, tal como ocurre en astronomía, predicción meteorológica y en la simulación y modelado de procesos hidrodinámicos, aerodinámicos, químicos y biológicos.

MACROCOMPUTADORES (mainframes). Son los ordenadores más potentes de uso común en aplicaciones comerciales. Están formados por grandes unidades independientes, con distintos procesadores centrales y otros procesadores que controlan la entrada/salida, los medios de almacenamiento masivo y los canales de comunicación de datos, provenientes de un gran número de terminales. Los mainframe permiten el trabajo concurrente de un gran numero de aplicaciones y usuarios, siendo su papel, ser el centro de sistemas transaccionales, tales como los que necesitan bancos, compañías aéreas, organismos de la administración, etc.

SISTEMAS DE TAMAÑO MEDIO (minicomputadores). Se enmarcan, en un nivel intermedio entre los macrocomputadores y los computadores personales: Suelen disponer de varios módulos de proceso y de unidades de entrada/salida, que trabajan en paralelo, todos ellos montados en un bastidor central. Pueden soportar varias aplicaciones simultáneas, proporcionando servicio a varios usuarios, pudiendo hacer de enlace de información con otros sistemas de información remotos. Son computadores adecuados para laboratorios, control de la producción y organizaciones administrativas de tamaño medio; aunque cada vez están más en competencia con sistemas de menor coste.

ESTACIONES DE TRABAJO. Son sistemas pensados para ser utilizado por un solo usuario, basados en microprocesadores muy potentes y dotados de gran

capacidad de memoria, diseñados desde el principio para soportar sistemas operativos capaces de ejecutar más de un programa de forma simultánea. Suelen estar conectados entre si, compartiendo recursos potentes como discos duros de alta velocidad, impresoras, y vías de comunicación con el exterior. Son especialmente adecuadas para diseño asistido por computador, cálculo científico y aplicaciones gráficas de alta resolución. Sin embargo, con la actual evolución, parece que en el futuro, tendremos dificultades para distinguir entre una estación de trabajo y un computador personal.

COMPUTADORES PERSONALES. Diseñados inicialmente para trebajar de forma aislada e individual, han evolucionado de forma extraordinaria en prestaciones y reducción de costes. También llamados microcomputadores, son los ordenadores más difundidos, pudiéndose encontrar comúnmente tanto en oficinas y laboratorios como en aulas y hogares. La gama de aplicaciones que puede ejecutar es enorme y cada día penetra más en actividades básicas de nuestra vida cotidiana. La posibilidad de trabajar en red, incrementa sus prestaciones sin que pueda saberse con exactitud cuál es su futuro, sus dos representantes son los PC compatibles y los Apple.

CALCULADORAS DE BOLSILLO (calculadoras programables). Con el aspecto tradicional de una calculador digital, incorporan las unidades funcionales que constituyen todo computador; con un teclado elemental para introducir datos y un sistema de presentación de una o pocas líneas, cada vez incluyen más cantidad de memoria, más posibilidades de programación y posibilidades de comunicación, con ordenadores próximos o lejanos. Su evolución es difícil de prever, al desaparecer las diferencias con los computadores personales portátiles.

1.9. COMUNICACIÓN DE DATOS Y REDES

La posibilidad de interconectar computadoras brinda tantos beneficios que se ha convertido en una de las áreas con mayor crecimiento en la actualidad. A medida que las máquinas se dispersan, su interconexión es más deseable (a mayor numero de usuarios con máquina propia, se incrementa la necesidad de interconectarlas). Cuando tenemos un conjunto de ordenadores conectados entre si, decimos que forman una **red** informática. Con ello, la comunicación de datos y la transferencia electrónica de información, se ha convertido en un tema básico.

1.9.1 TRANSMISIÓN DE DATOS DENTRO DEL COMPUTADOR

Antes de seguir con temas de comunicación, hay que resaltar, que cualquier tipo de transmisión de datos a larga distancia, funcionalmente, no debe

diferir de otras comunicaciones de datos, que se dan en el funcionamiento habitual del ordenador; tales como las que se producen dentro de los propios circuitos integrados (cuyo orden de magnitud es el mm) o las que se tienen lugar en una misma tarjeta, a través de los buses, entre procesador y memoria (del orden de cms).

Incluso, tenemos conexiones de mayor longitud, en la comunicación entre el procesador y sus periféricos, que habitualmente requieren un soporte físico distinto, en forma de cables del orden de las decenas de centímetros. Estos cables constan de varios hilos, entonces se esta en condiciones de transmitir varios bits simultáneamente, uno por hilo, que ya definimos como transmisión de **datos en paralelo**, cuando hablamos de buses. Sin embargo cuando los datos tienen que transmitirse a cierta distancia, el uso de este tipo de cable resulta inviable; entonces se utiliza un conductor único. En estas circunstancias se transmiten los datos bit a bit, lo que se denomina transmisión de datos **en serie**.

Las distintas posibilidades de intercomunicación implican que los dos soportes del computador (el físico y el lógico) tengan que adaptarse, a nuevos equipos y a procedimientos preestablecidos. Dados dos componentes, conectados entre si, llamaremos **Interfaz,** al conjunto de informaciones y equipos utilizadas para interconectarlos, que nos indican las secuencias de operaciones permitidas entre ellos y **Protocolo**, al conjunto de reglas y de procedimientos que permiten que desde uno de los componentes y a través de las interfaces respectivas, que el otro componente realice correctamente una función determinada. Por tanto las interfaces y protocolos especifican la forma de interconectar dos o más componentes de un sistema informático. Dado que hay dos tipos de transmisión, serie y paralelo, existen estas mismas dos clases genéricas de interfaces: y en cada uno de ellos se distinguen distintos tipos. En la Figura 1.12. se puede ver un ejemplo de un interfaz paralelo (tipo centronics) comúnmente utilizado para conectar un computador a una impresora.

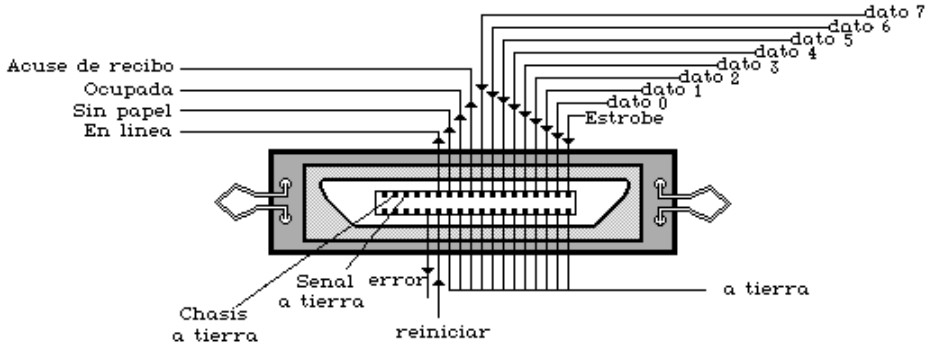

Fig. 1.12. *Ejemplo de una conexión paralelo (tipo centronics)*

1.9.2 COMUNICACIÓN DE DATOS A LARGA DISTANCIA

Para llevar a cabo la comunicación entre dispositivos informáticos, se necesita un medio físico por donde transmitir las señales eléctricas, que van desde el simple es el par de cables trenzados hasta la fibra óptica, además de la posibilidad de conexión inalámbrica, a través de enlaces vía radio y vía satélite. Para transmisiones del orden de kilómetros, lo más común, es recurrir a la red telefónica ya existente, que está pensada para transmitir voz, por lo que su rango de frecuencias, de 300 a 3400 Hz, no la hace totalmente adecuada para la transmisión digital.

Para poder transmitir datos mediante la red telefónica, debemos recurrir a una señal analógica, que codifique de cierta manera, los diferentes valores discretos, que caracterizan la información digital. Este proceso se lleva a cabo por un dispositivo llamado *modem* (que toma su nombre de la expresión **mo**dulador/**dem**odulador). Los *modems* se conectan a la red telefónica de forma que se pueden utilizar la infraestructura de la misma, incluidas las conexiones celulares, de los teléfonos portátiles, para intercambiar información. Las velocidades de transmisión de datos varían considerablemente, dependiendo de las aplicaciones y el medio de comunicación. La unidad de medida del ritmo de transmisión de datos es el **baudio**. cuya definición precisa es realmente complicada, pero puede considerarse para propósitos prácticos como la velocidad de transmisión (bit/seg).

La necesidad de interconectar dispositivos separados por grandes distancias, ha propiciado la aparición de una nueva disciplina técnica la **Telemática**, como rama interdisciplinar entre la Informática y las Telecomunicaciones, que versa sobre la utilización de equipos informáticos interconectados a través de líneas o redes de telecomunicación, bien sea periférico-computador, o computador-computador. Cuando introducimos estos nuevos elementos, aparecen una serie de nuevos factores que no se apreciaban en el tipo de interconexión que define el bus: el medio de transmisión -que va a requerir un tipo de circuitería especial y la aparición de nuevos errores de transmisión, debidos a la larga distancia y que no se dan en el interior del ordenador. Será de nuevo, tarea de la interfaz y del protocolo correspondiente, el establecer entre las máquinas, el convenio de transmisión de datos (piénsese que en una máquina una palabra puede ser de 1 byte y en otra de dos), la sincronización entre ambas, la velocidad de transmisión, la comprobación de errores de transmisión, etc.

Con independencia del medio de comunicación entre dispositivos, analógico o digital, aparece la cuestión de gestionar la intercomunicación de múltiples y diferentes sistemas informáticos (*networking* en inglés), de forma que los distintos recursos puedan ser compartidos y utilizados de forma remota. En el caso de que estos recursos estén situados en un mismo edificio o en edificios próximos,

hablaremos de una LAN (Local Area Network), mientras que si hablamos de áreas geográficas de mayores dimensiones nos referiremos a WAN (Wide Area Network).

1.9.3 REDES DE ORDENADORES

Entenderemos como red una manera de interconectar ordenadores de tal forma que cada uno de ellos sea consciente de esta circunstancia y de los recursos que puede compartir. La red viene a superar la utilización por múltiples usuarios de una misma máquina, habida cuenta que resulta mucho más funcional y económico, el dotar a cada usuario de su propio computador y conectarlos entre si, que poner en servicio complicados sistemas operativos que resuelvan este mismo problema. Las redes, en principio, consiguen aumentar la relación prestación/coste, proporcionando las siguientes posibilidades:

- Aumentar la seguridad (fiabilidad) del sistema (si una computadora de la red falla, se puede utilizar otra de la misma red).

- Si el equipo local, al que se tiene acceso directo, no dispone de las prestaciones adecuadas (poca memoria, está muy cargado de trabajo, no dispone de impresoras rápidas, etc.), el usuario puede conectarse a otro equipo de la red que reúna las características pertinentes y utilizar sus recursos o repartir su tarea entre varias máquinas.

- Un servicio remoto para la utilización de aplicaciones, sin necesidad de que el usuario disponga realmente de ellas.

- Permitir el acceso a bancos de datos ubicados a grandes distancias.

- Posibilitan la existencia de sistemas de control industrial constituidos, por ejemplo, por varias computadoras de uso específico de una fábrica, interconectadas entre sí.

- Utilización de la red de computadoras como medio de comunicación: correo electrónico.

- etc.