APUNTES DE ELEMENTOS DE INFORMÁTICA

Ezequiel Moyano
Adriana Urciuolo
Esteban Alejandro Czelada
Colaboración especial:
Emilio Izarra

ediciones UNTDF
COLECCIÓN
EN CARRERA

APUNTES DE ELEMENTOS DE INFORMÁTICA

APUNTES DE ELEMENTOS DE INFORMÁTICA

Autores:

Lic. Ezequiel Moyano
Mgs. Adriana Urciuolo
Ing. Esteban Alejandro Czelada

Colaboración especial:
Ing. Emilio Izarra

COLECCIÓN
EN CARRERA

ediciones UNTDF

Moyano, Ezequiel
Apuntes de elementos de informática / Ezequiel Moyano ; Adriana Urciuolo ; Esteban Alejandro Czelada ; contribuciones de Emilio Izarra. - 1a ed. - Ushuaia : Ediciones UNTDF, 2023.
Libro digital, PDF - (En Carrera / Daniela Stagnaro ; 2)
Archivo Digital: descarga
ISBN 978-987-46807-4-7

1. Tecnología Informática. 2. Ingeniería de Sistemas. I. Urciuolo, Adriana. II. Czelada, Esteban Alejandro. III. Izarra, Emilio, colab. IV. Título.
CDD 004.02

Colección EN CARRERA

Fuegia Basket 251, Ushuaia (9410), Tierra del Fuego, Antártida e Islas del Atlántico Sur, Argentina. Tel.: (0054) 2901-430892. ediciones@untdf.edu.ar
http://www.untdf.edu.ar/extension/ediciones

Diseño y maquetación: Daniel Vidable
Corrección de estilo: Miram Camponovo

ISBN: 978-987-46807-4-7
Hecho el depósito que marca la Ley 11723

ÍNDICE

EVOLUCIÓN HISTÓRICA DE LAS COMPUTADORAS

Procesamiento electrónico de datos

Comúnmente, escuchamos hablar de "procesamiento de datos" vinculando erróneamente este concepto al de "cálculo con computadoras". Este error surge de confundir "procesamiento de datos" con "procesamiento electrónico de datos"; la palabra "electrónica" indica que nos referimos a computadoras.

Otras formas de procesar datos son las siguientes:

- Por medio de máquinas registradoras, calculadoras mecánicas, etcétera; lo que llamaremos "proceso mecánico de datos".
- Por medio de personas como los administrativos de las oficinas, que escriben sobre formularios o facturas; en este caso utilizaremos el término "proceso manual de datos".

A continuación, trataremos de definir en forma general el procesamiento de datos.

En primer lugar, diremos que los datos son elementos susceptibles de observación directa, es decir, elementos de conocimiento.

Veamos el ejemplo de un proceso de pedidos en el cual una empresa ofrece descuentos a sus clientes, dependiendo del importe del pedido; si este pasa los 510.000, se descuenta un 5%, si pasa los 525.000, un 10%.

La secuencia de operaciones a realizar por el empleado sería la siguiente:

- Tomar la factura de la bandeja "entradas".
- Decidir si tiene derecho a un descuento. (proceso).
- En caso afirmativo, calcular el importe del descuento y el importe neto de la factura. (proceso).

• Escribir en la factura el descuento y el importe neto y ponerlo en la bandeja de "salidas".

Aunque como datos entraron el nombre, teléfono y dirección del cliente, el pedido y su importe, el empleado que calculó el importe a pagar, seleccionó qué datos le interesaba utilizar, en este caso, solo el importe del pedido. Por lo tanto, de un conjunto de datos, organizándolos y procesándolos convenientemente, podemos obtener la información necesaria para la efectivización de un objetivo deseado.

Lo que hizo el empleado fue procesar datos para obtener información. La diferencia básica entre dato e información consiste en que los datos no son útiles o significativos como tales, sino hasta que son procesados y convertidos a una forma útil llamada información.

Podemos, entonces, ampliar la idea que tenemos de procesamiento de datos diciendo que:

Procesar significa transformar insumos a fin de obtener un producto.

Procesamiento de datos es planificar una serie de acciones y operaciones sobre determinados datos con el fin de llegar a la información deseada.

El procesamiento se puede esquematizar en etapas como en la figura 1.1.

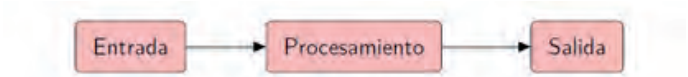


Figura 1.1: Etapas del procesamiento de datos

Estamos ligando la palabra *dato* a los conceptos de *entrada* y de *proceso*, y la palabra *información* al concepto de *salida*, como resultado del proceso.

En todo procesamiento se transforman datos para elaborar los resultados que originarán la información, esta permitirá la toma de decisiones. Podrán existir resultados parciales (informaciones) que vuelven al proceso antes de transformarse en información final.

En todo tipo de procesamiento encontramos que muchos se realizan con datos (un remito, un precio, etcétera) e informaciones (una cuenta, un nuevo

saldo, etcétera). Es decir, en muchos casos la información se transforma en dato para dar origen a una nueva información.

Además, esta debe responder a lo que se requiere, es decir, *ser oportuna, completa y actual*.

La ciencia que estudia la obtención de información por métodos automáticos es la *Informática*.

Decimos que la Informática es una ciencia porque constituye un conjunto de conocimientos de validez universal y porque utiliza el método científico para el logro de sus objetivos.

La Asociación Francesa de Normalización (AFNOR), miembro de la Organización Internacional de Normalización, define a la Informática como "el conjunto de disciplinas científicas y de técnicas específicamente aplicables al procesamiento de la información, especialmente por medios automáticos".

Podemos decir, entonces, que la Informática se encarga del estudio de la circulación, transmisión, tratamiento y difusión de la información. Comprende disciplinas tales como el procesamiento electrónico de datos, análisis de sistemas y la programación.

Las computadoras

Para el creador de la palabra informática, esta no se concibe sin las computadoras.

Una computadora procesa datos en forma automática bajo control de un programa, para producir la información deseada (ver figura 1.2).

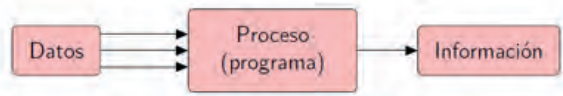


Figura 1.2: Datos, proceso e información

Una computadora hace uso de dispositivos que le permiten leer datos, (realizar la entrada) y presentar resultados (ofrecer una salida); componentes que le permiten realizar cálculos, operaciones lógicas, organizar cuándo funciona cada dispositivo. A su vez, posee una memoria que almacena cada

dato que se va procesando, el resultado de los cálculos y el programa; dispositivos que permiten guardar datos históricos en archivos, etcétera.

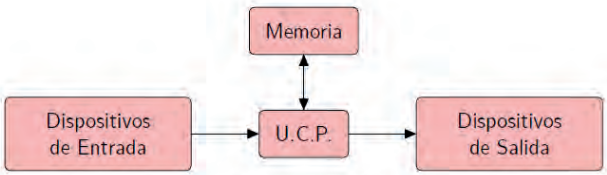


Figura 1.3: Esquema del procesamiento electrónico de datos

Las ventajas del uso de computadoras para la obtención de información son, entre otras, las siguientes:

- Altas velocidades de proceso.
- Gran confiabilidad.
- Menos errores humanos.
- Alta precisión.
- Permiten archivar grandes volúmenes de datos con una gran velocidad de acceso a los mismos.

Una computadora puede realizar tareas tales como las que se detallan:

- Operaciones aritméticas.
- Operaciones lógicas.
- Recuperación de datos.
- Presentación de resultados.
- Toma de decisiones.

Antecedentes

A lo largo de la historia, el hombre fue resolviendo sus necesidades de registración para llevar la cuenta de sus bienes y efectuar las operaciones necesarias para su permuta o venta. Fue ideando métodos ágiles de cálculo, como contar con los dedos (dígitos), piedras, nudos en la soga, etcétera.

Así, los egipcios utilizaban el contador de arena, que consistía en surcos en la arena donde colocaban piedras (se agrupaban guijarros en montones de a 10). Partiendo de la idea de contar con los dedos, los pueblos primitivos tomaron como base de sus cálculos el número 10; algunos pueblos, como los mayas, calculaban en base 20; los esquimales, en base 5.

Un mejoramiento en las técnicas de cálculo surge con la aparición del ábaco en el año 3000 a. C., una tabla con una serie de hendiduras; en la primera se colocan tantas piedras como unidades se quieren representar; en la segunda, decenas, y así sucesivamente. Esto supuso un gran avance para las culturas que lo adoptaron. Es considerado como la primera máquina típicamente digital que el hombre haya utilizado para ayudarse a resolver sus problemas aritméticos. Conocido por los aztecas y los rusos, permitía sumar, restar y multiplicar por medio de sumas sucesivas.

Más tarde apareció en China y Japón donde fue llamado *suanpan* y *soroban,* respectivamente.

Después de este avance, pasaron más de tres mil años antes del siguiente progreso importante, ya que el uso de los números romanos obstaculizó la invención de sistemas mecánicos de cálculo.

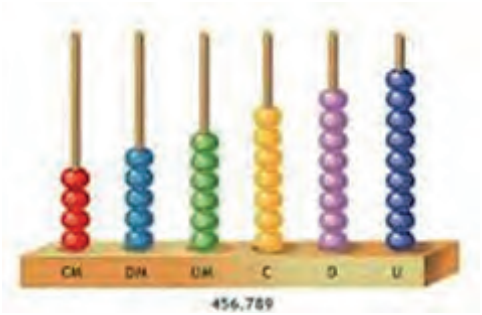


Figura 1.4: Abaco

Fueron necesarios miles de años para lograr una simbología práctica de las magnitudes que permitiera realizar fácilmente las operaciones. El primer método conocido consistía en representar cada magnitud con una marca; los griegos utilizaron letras del alfabeto. La mayor dificultad que ofrecen estos sistemas es la inexistencia del cero. En el siglo I o II, los matemáticos indios introdujeron el concepto del cero, así como la ordenación de los números en posiciones consecutivas que indican las unidades, decenas, centenas, y demás. Este sistema llegó a la civilización europea a través de los matemáticos árabes.

La aritmética basada en la notación indoarábiga, en lugar de la romana, se conoció como algoritmia. Alrededor del año 1200, con la aceptación del

sistema arábigo se favorecieron los avances. Pero no apareció ningún aparato mecánico hasta el siglo XVII.

En 1617, Neper desarrolló los logaritmos a través de un sistema tabular que proporcionó un método conveniente para abreviar los cálculos, de allí se deriva la invención de la Regla de Cálculo, la que consta de una escala fija y otra móvil, con la cual las operaciones aritméticas se resuelven por medio de sumas y restas de distancias geométricas. La Regla de Cálculo se le atribuye a Ougthre (1632). Cada escala está marcada de modo tal que las distancias reales desde el comienzo de la regla son proporcionales a los logaritmos de los números impresos sobre ella. Deslizando las escalas, se puede multiplicar y dividir fácilmente.

Figura 1.5: Sistemas de Numeration

Precursores

Las máquinas precursoras de las computadoras datan del siglo XVII. Las partes que realizan cálculos en las actuales computadoras son el resultado de la evolución lógica de las máquinas de sumar y las calculadoras mecánicas.

En 1642, Blas Pascal, en Francia, fue el primero en construir una máquina de sumar. Este es el primer eslabón que registra la historia sobre el tratamiento automático de la información: la calculadora de ruedas numéricas o la rueda contador decimal. La concepción primaria que esta máquina introdujo fue la

mecanización del traslado de cifras. La máquina consiste en una serie de ruedas numeradas o discos, los cuales contienen los números de 0 a 9 y se encuentran dispuestos de manera que puedan leerse de izquierda a derecha. Cuando uno de los discos pasa de 9 a 0, un trinquete hace que la rueda de la izquierda se mueva una unidad hacia delante para tomar en cuenta el arrastre producido. La máquina suma y resta (por complemento) directamente, pero la multiplicación y división son ejecutadas por medio de sumas y restas repetidas. Los resultados se proporcionan en forma directa enseñando un número a través de una ventana.

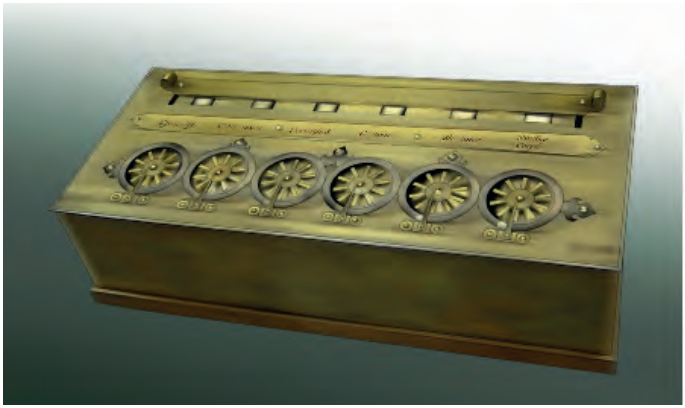


Figura 1.6: Calculadora de ruedas

En 1673, Leibnitz construyó su calculadora de rueda de pasos, que efectuaba las cuatro operaciones matemáticas elementales. La suma y la resta se realizaban igual que en la máquina de Pascal; la multiplicación se realizaba directamente gracias a unos engranajes adicionales que se le incorporaron a la máquina.

Se puede considerar que la calculadora de mesa y la caja registradora son descendientes directos del invento de Leibnitz.

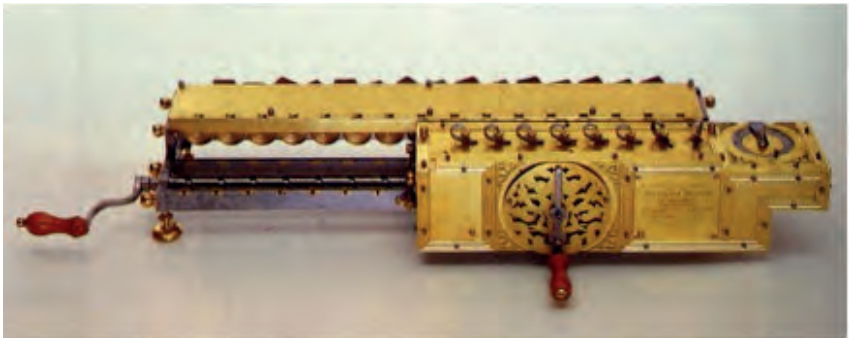


Figura 1.7: Calculadora de ruedas de pasos

Tanto los inventos de Pascal como los de Leibnitz fueron hechos a mano, y por lo tanto, su fabricación en serie hubiera sido costosa. Faltaba la tecnología.

En la evolución de la computación influyeron, en gran medida, las técnicas de las tarjetas perforadas que surgen de la industria textil. En 1801 Joseph Jacquard, un tejedor francés, perfeccionó una máquina que empleaba una secuencia de tarjetas perforadas, cuyas perforaciones controlaban la selección de los hilos y la ejecución del diseño; solamente podían entrar en el patrón aquellos hilos cuyo gancho guía encontraba un hueco en la tarjeta.

Figura 1.8: Máquina- telar de tarjeta perforada

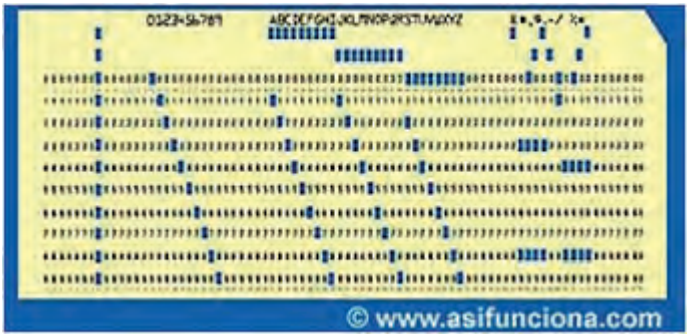


Figura 1.9: Tarjeta perforada

Los ingenios citados anteriormente no pueden considerarse como máquinas automáticas pues requieren la continua intervención del operador para introducir nuevos datos, y anotar los resultados intermedios. El deseo de evitar estas engorrosas maniobras condujo a la necesidad de discutir la posibilidad de crear una máquina que sea capaz de realizar cálculos automáticamente, es decir, sin la intervención humana durante el proceso.

El matemático inglés Charles Babbage, alrededor de 1830, reflexionó en lo inútil de aumentar la velocidad de las máquinas mecánicas si las operaciones debían seguir siendo preparadas manualmente. Era preciso automatizar el paso de una operación a la siguiente. Pero, la secuencia de operaciones a ejecutar no eran siempre las mismas. Entonces, tuvo la idea de recoger un concepto precedentemente utilizado por Jacquard para la automatización de los telares, el de programa exterior.

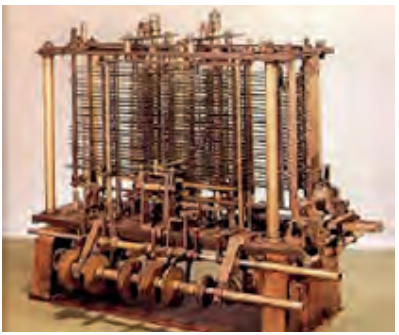


Figura 1.10: Máquina analítica de Babbage

La máquina de Babbage debía tener una tarjeta perforada que contuviera la definición de la operación a ejecutar; debía ejecutar la operación, leer la ficha siguiente, y repetir el proceso. En el terreno de los principios, la máquina de Babbage es la antecesora de las máquinas eléctricas de relés, construidas por algunas empresas y universidades americanas en el curso de la última guerra mundial.

Estas máquinas son designadas como "máquinas de programa exterior", subrayando con ello que el programa no forma parte de la máquina, sino que se ejecuta paso a paso a partir de un soporte exterior intercambiable.

La "máquina analítica" de Babbage, diseño que nunca fue llevado a la práctica, contenía todos los elementos que constituyen una computadora moderna y que la diferencian de una calculadora.

Estaba dividida funcionalmente en dos grandes partes: una que ordenaba y otra que ejecutaba. Esta última era una versión muy ampliada de la máquina de Pascal, una máquina con ruedas contadoras decimales que fuera capaz de ejecutar una operación de suma en un segundo. La otra era la parte clave. El usuario podía, cambiando las especificaciones del control, lograr que la misma máquina ejecutara operaciones complejas, diferentes de las que había hecho antes. Babbage había diseñado su máquina con capacidad de acumular datos, operar y controlar la ejecución de las instrucciones. Para ello, debía disponer de lo siguiente:

1. Dispositivos de entrada: una sección de recepción de los datos con los que iba a trabajar y de las instrucciones necesarias para las operaciones. Esta computadora "leía" los datos de entrada por medio de tarjetas perforadas.
2. Memoria: para almacenar datos introducidos y los resultados de las operaciones intermedias.
3. Unidad de control: para vigilar la ejecución de las operaciones según la secuencia adecuada.
4. Unidad aritmético-lógica: a fin de efectuar las operaciones para las que ha sido programada la máquina.
5. Dispositivos de salida: para transmitir al exterior los resultados del cálculo llevado a cabo.

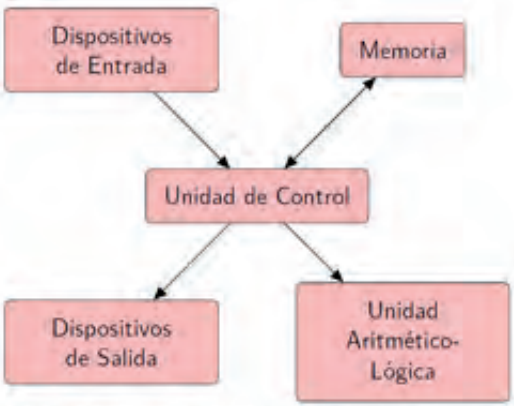


Figura 1.11: Esquema de la "Máquina Analítica de Babbage"

Los conceptos desarrollados por Babbage lo convirtieron en el precursor del requerimiento de velocidad tanto para los datos como para las instrucciones, demostró que era necesario tener ambos almacenados. Planteó tener las instrucciones requeridas para una serie de operaciones, preparadas con anterioridad y colocadas en el almacenamiento de instrucciones en el orden que debían ser utilizadas. Estas ideas se usaron con cierto éxito cien años más tarde.

Entre 1842 y 1843, la matemática Augusta Ada King (condesa de Lovelace), en su trabajo sobre los conceptos de la máquina analítica de Babbage, dejaría el primer algoritmo a ser procesado por una máquina, por lo que a menudo se la menciona como la primera programadora.

En 1854, George Boole, matemático inglés, presentó la lógica en los símbolos matemáticos, a lo que se llamó álgebra booleana o álgebra lógica.

Esto tuvo gran importancia para las computadoras, ya que están construidas mediante redes muy complejas de circuitos y el álgebra booleana proporciona un método para su representación mucho más eficiente y sistemático que las representaciones geométricas o electrónicas convencionales.

El espectacular avance de la Revolución Industrial durante el siglo XIX, así como la creciente complejidad de la organización social, planteó un nuevo problema: el tratamiento de grandes masas de información.

En las últimas décadas del siglo XIX, la oficina de censos de los Estados Unidos se enfrentaba con un problema prácticamente insoluble: las leyes americanas ordenaban efectuar un censo de la población cada 10 años y en 1886

todavía se trabajaba con los datos de 1880. Era evidente que no se podría terminar su clasificación en el momento de realizar el censo de 1890.

Herman Hollerith, funcionario de la citada oficina, entendió que la única solución residía en la mecanización de las operaciones de recuento y clasificación; se dio cuenta de que en la mayor parte de las preguntas del censo, se respondía mediante un "sí" o un "no" y, conocedor del mecanismo de las tarjetas perforadas del telar de Jacquard, comprendió que en estas se podía representar la respuesta "sí" mediante una perforación en un lugar determinado y la respuesta "no" con la ausencia de dicha perforación. Además, Hollerith ideó la posibilidad de detectar dichas respuestas mediante contactos eléctricos establecidos a través de las perforaciones: "el paso de la corriente representaba un sí y la falta de corriente un no". Las máquinas ideadas por Hollerith para el tratamiento de tarjetas perforadas fueron ya utilizadas para el censo de 1890.

Estas innovaciones aumentaron la velocidad, versatilidad y utilidad de las máquinas de tarjetas perforadas. Esto dio por resultado que se usaran cada vez más estos dispositivos para procesamiento de datos de negocios, así como computaciones científicas y cálculos estadísticos.

Después que Hollerith paseó sus inventos por el mundo durante veinte años sin mucho éxito, se vio obligado a vender, primero, una parte de sus patentes y, después, la sociedad misma en 1912. Un grupo de financistas la volvió a comprar y puso a la cabeza de la nueva sociedad denominada "Tabulating Machine Corporation" (TMC), al Sr. Thomas Watson, quien era el director de la "National Cash Register" (NCR). Dos años más tarde, Watson, convertido en propietario de TMC, cambió el nombre por "International Business Machine" (IBM).

Evolución de las computadoras

Después de la Primera Guerra Mundial se comenzaron a instalar máquinas electrónicas como ordenadores.

En 1936, Konrad Zuse desarrolló en Alemania los conceptos fundamentales referentes a los ordenadores automáticos: utilizó el sistema binario y lenguajes de programa muy simples.

Por orden del Centro Experimental Alemán de Aviación, construyó la primera calculadora-ordenador con programa: El ZUSE 23. Esta máquina

trabajaba con 2600 relés y era capaz de realizar en un minuto hasta veinte operaciones aritméticas.

Mientras Aiken trabajaba en la construcción de un ordenador, Claude Shannon inventaba en EE. UU. los circuitos de conmutación eléctricos que cumplían los "principios de Boole", y el inglés Alan Turing desarrollaba el fundamento teórico para técnicas de programación.

En 1944, el profesor Howard Aiken de la Universidad de Harvard, con la colaboración de IBM, puso en marcha la primera computadora de la historia llamada MARK I, en cuya construcción invirtió más de cinco años. Este ordenador fue construido sobre la base de elementos electromecánicos, careciendo de los electrónicos. El esquema lógico del MARK I se adaptaba perfectamente al propuesto por Babbage, es decir, constaba de unidades de entrada y salida, memoria, unidad de cálculo y unidad de control. Las 200 mil piezas y los 800 mil cables empleados dan una idea de la magnitud del proyecto. Era una máquina lenta con gran cantidad de relés y lámparas eléctricas capaz de efectuar las operaciones aritméticas, cálculos de trigonometría y logaritmos.

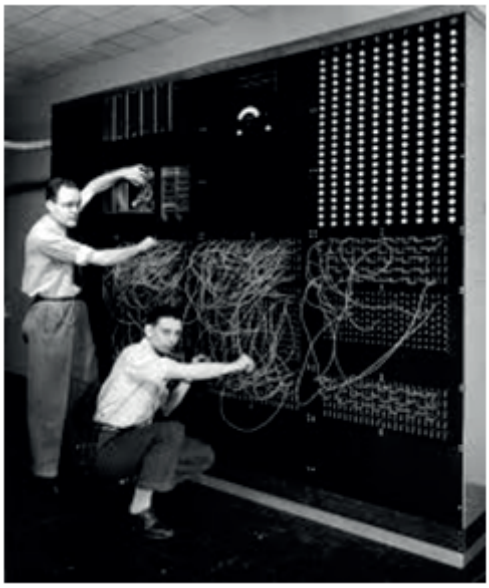


Figura 1.12: Mark II

El profesor Aiken perfeccionó su prototipo durante la guerra e inmediatamente después puso en marcha el MARK II.

En 1946, los profesores Eckert y Mauchly, de la Universidad de Pennsylvania, pusieron en marcha el Integrador y Computador Numérico Electrónico (ENIAC), primer ordenador electrónico de la historia. A diferencia del MARK I, las operaciones de almacenamiento, cálculo y control de secuencia de operaciones eran efectuadas por circuitos electrónicos. El ENIAC disponía de unos 18 mil tubos de vacío, ocupaba todo el sótano de la Universidad, consumía 200 kW de energía y liberaba tanto calor que requería de un sistema de aire acondicionado industrial. Era capaz de realizar cinco mil sumas por segundo (contra una por segundo del MARK I).

El ENIAC era incapaz de almacenar distintos programas. Para pasar de uno a otro, los ingenieros tenían que modificar parte de los circuitos de la máquina con el fin de que esta efectuara las operaciones requeridas para la solución de cada problema específico.

Figura 1.13: ENIAC

El doctor Von Neumann se planteó la posibilidad de construir un ordenador en el que no hubiera que variar los circuitos internos al cambiar el programa.

Promovió el paso decisivo hacia la mecanización del tratamiento digital de la información con la invención de dos nuevos conceptos que se detallan:

1. El Programa Registrado: las máquinas de la época poseían elementos de memoria capaces de conservar en el curso de la ejecución, resultados parciales para su posterior utilización. Von Neumann tuvo la idea de utilizar las memorias del calculador para almacenar también el programa. En lugar de ejecutar las operaciones al compás de su lectura en una cinta perforada, la nueva máquina supone almacenado el programa en memoria, antes de la ejecución de las operaciones.

2. La Ruptura de Secuencia: la máquina de programa exterior necesitaba de la intervención humana cada vez que se planteaba una toma de decisión; en otras palabras, que el tratamiento consiguiente dependía de los resultados que iban obteniéndose. Sobre la base de las nuevas posibilidades de las máquinas de programa registrado, Von Neumann concibió la idea de hacer automáticas las operaciones de decisión lógica, dotando a la máquina de una instrucción llamada: "salto condicional" o también "ruptura condicional de secuencia". Según el valor de un resultado obtenido, la máquina ejecutaría una u otra parte del programa.

La mayoría de las computadoras funcionan de acuerdo al modelo de Von Neumann.

Las computadoras de programa almacenado señalaron el siguiente gran avance en el desarrollo de las computadoras electrónicas. La primera que se construyó fue la EDSAC.

Figura 1.14: EDSAC

- 1949 (Computadora Automática Electrónica de Almacenamiento Diferido), puesta en marcha en la Universidad de Cambridge, Reino Unido. El primero en ser planeado fue el EDVAC que se construyó en la Universidad de Pennsylvania, y se retrasó por la marcha de los profesores Eckert y Mauchly, que decidieron fundar su propia empresa para la fabricación de computadoras electrónicas.
  En 1947 se fundó la Eckert - Mauchly Computer Corporation y como fruto de sus trabajos, en 1951 apareció el UNIVAC I. Esta fue una de las primeras máquinas en utilizar cinta magnética como dispositivo de entrada y salida. Disponía de gran velocidad, confiabilidad y capacidad de memoria.

Figura 1.15: UNIVAC I

Generaciones de computadoras

Se suelen distinguir tres generaciones de computadoras desde el momento de su comercialización.

Esta distinción se funda habitualmente en criterios de índole tecnológica, pero las mutaciones de una a otra generación son mucho más interesantes y significativas.

Generalmente se hace corresponder a la primera generación (1944-1958) con las válvulas de vacío, a la segunda generación (1958-1965) con los transistores, y a la tercera generación (1965-1980, aprox.) con los circuitos integrados. Existe una cuarta generación, cuyo período se desarrolla en la actualidad, que comenzó alrededor de 1980.

Trataremos de describir de una manera más amplia y completa, los avances producidos en cada generación enfocándolos no solo desde el punto de vista de la electrónica, sino del tipo de almacenamiento interno y auxiliar utilizado y de las técnicas de organización y explotación.

Primera generación (1944-1958)

La primera generación de computadoras constituye la continuación inmediata de los prototipos construidos en las universidades estadounidenses e inglesas. Respecto de las técnicas constructivas, las computadoras de la primera generación se caracterizaban por la utilización de válvulas de vacío.

En ellas se utilizó como almacenamiento interno el tambor magnético, el cual almacenaba datos en pistas circulares sobre un cilindro recubierto con material magnetizable.

El adelanto más importante que surgió fue el Programa de almacenamiento interno. Los datos para los cálculos se almacenaban con las instrucciones en la "memoria" del equipo.

La programación se realizaba en lenguaje de máquina.

Otros avances importantes fueron la memoria intermedia y el acceso al azar. El almacenamiento intermedio es un dispositivo que sirve como memoria transitoria. El acceso al azar de la información permite acceder en forma directa a alguna unidad de almacenamiento para la lectura y escritura de datos.

Como almacenamiento auxiliar se utilizaba la cinta magnética.

La programación de cada serie de trabajos era lenta y compleja, utilizaba códigos cifrados difíciles de manejar.

En cuanto a las técnicas de organización y explotación, las computadoras de la primera generación ejecutaban sus trabajos de manera puramente secuencial, cada uno en tres tiempos:

1. El programa perforado en tarjetas o en cintas de papel era leído y registrado en memoria gracias a un programa cargador.
2. El programa era ejecutado.
3. Se imprimían los resultados.

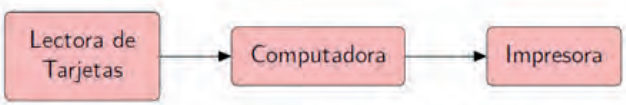


Figura 1.16: Procesamiento secuencial de trabajos

En el programa era posible incluir la lectura de nuevos datos u obtener impresiones de los resultados parciales, pero las operaciones de procesamiento, de entrada o de salida, tenían que encadenarse en el tiempo, con lo cual la duración de todo el proceso era la suma de todas y cada una de las operaciones.

Las computadoras destacadas de la primera generación fueron el UNIVAC I y las series 600 y 700 de IBM.

Segunda generación (1958-1965)

Comienza a utilizarse el transistor, sustituyendo a la válvula.

Un transistor es un pequeño paralelepípedo de silicio con base cuadrada, de algunas décimas de milímetros de lado y de 150 micrones de altura. Los transistores, cada uno montado en una cápsula, eran ensamblados con otros componentes (diodos, resistencias, capacitores) sobre placas de tamaño reducido. Las interconexiones entre componentes se realizaban por el procedimiento de impresión metálica.

El uso de transistores de tamaño reducido trajo como consecuencia mayor confiabilidad, menor generación de calor y consumo de energía, aumentando la eficiencia y disminuyendo el costo. Permitió acrecentar la potencia y la velocidad de las viejas computadoras de la primera generación.

En cuanto al almacenamiento interno, se reemplazaron los tambores magnéticos por núcleos magnéticos.

Como almacenamiento externo o auxiliar, se utilizaron los discos magnéticos.

Las nuevas posibilidades de la unidad central permitieron multiplicar y aumentar la potencia de las unidades de entrada y salida, por una parte; y por la otra, simplificar los códigos de programación volviéndola más fácil. Esto se consiguió con el desarrollo de los lenguajes de "alto nivel".

Durante esta época, además, se estaba definiendo una nueva ciencia: la comunicación entre la computadora y el usuario.

En cuanto a las técnicas de organización y explotación, la computadora de la segunda generación ofrecía posibilidades de simultaneidad entre el cálculo interno y las operaciones de entrada/salida. Sin embargo, el encadenamiento de los trabajos seguía siendo secuencial, ya que las posibilidades de simultaneidad solo se daban dentro de un mismo programa. En particular, la unidad

central permanecía inactiva cuando se cargaban nuevos programas en memoria.

La desproporción entre la velocidad de cálculo y las velocidades de lectura de tarjetas o de impresión, ocasionaba que la unidad central no fuera utilizada realmente más que durante un pequeño porcentaje de tiempo.

El problema se solucionó asignando a las cintas magnéticas, más rápidas que las lectoras de tarjetas e impresoras, las operaciones de entrada y salida, siguiendo el proceso que se ilustra en la figura 1.17.

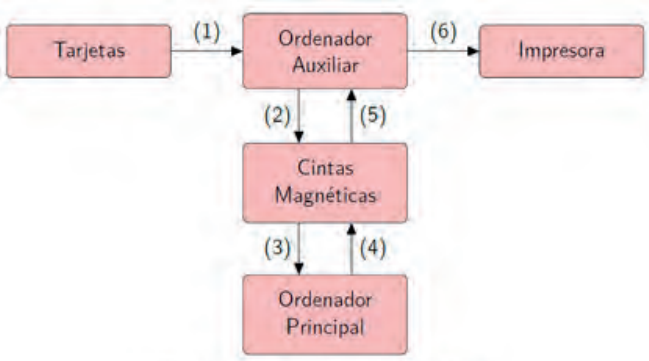


Figura 1.17: Procesamiento por lotes

Este procedimiento de explotación se conoce con el nombre de procesamiento por lotes, para indicar que es necesario esperar a que el lote de trabajos cargados en la cinta magnética haya sido totalmente procesado, para poder cargar uno nuevo.

Dentro de la segunda generación podemos citar las series "400" y "700" de IBM, el "1107", de Sperry Raud, entre otros.

Tercera generación (a partir de 1965-comienzos de los 1980)

Se inauguró con la serie 360 de IBM.

En esta generación se comenzó a trabajar con circuitos integrados usando la tecnología del "estado sólido" que incorpora diversos circuitos en un solo bloque. Su tamaño es similar al del transistor y en uno de ellos se encontraban hasta una docena de componentes elementales interconectados entre sí. Esta red de circuitos integrados está formada por escamas de silicio muy delgadas.

Aumentan las velocidades internas de procesamiento y reducen los costos de energía.

Como almacenamiento interno o principal se utilizaron núcleos magnéticos, estado sólido, y como almacenamiento auxiliar se comenzó a utilizar el disco magnético rígido y los disquetes.

Uno de los rasgos más característicos de esta generación es el gran desarrollo de *software*, creando un conglomerado de técnicas y lenguajes para un uso más fácil de la máquina. Llegó la generalización de los lenguajes "evolucionados" y universales; es decir, utilizables en todas las máquinas de igual potencia, sea cual fuere la marca, FORTRAN, COBOL, BASIC, Pascal, C, etcétera. Se comercializan programas utilitarios y paquetes de utilidades que permiten reducir el tiempo de trabajo de los programadores y que puedan tener acceso a una máquina usuarios ajenos a temas específicos de computación.

En cuanto a las técnicas de organización y explotación, el ordenador de la tercera generación permite explotar eficazmente las simultaneidades latentes ya en la segunda. Varios programas pueden residir en la memoria en un instante dado, solo uno de ellos utiliza la unidad central, mientras los otros pueden simultáneamente efectuar operaciones de entrada y salida al mismo tiempo.

Cuando el programa que utiliza la unidad central se detiene en espera de una operación de entrada o salida, otro programa toma su lugar y esto evita los tiempos muertos en la unidad central.

Este método de explotación se llama multiprogramación. Permite mejorar el empleo del conjunto de recursos de un sistema informático.

La carga por lotes se sustituyó por la carga continua de los trabajos a medida que se presentan, utilizando el disco magnético como memoria auxiliar.

Con la tercera generación surge la comunicación y transmisión de datos que permite conectar un centro de cómputos con unidades terminales remotas y de representación visual.

Esta posibilidad de trabajar a distancia se denomina teleprocesamiento.

Se desarrollaron también los sistemas conversacionales, los cuales permiten a los usuarios seguir el desarrollo de las diferentes etapas de sus programas, así como intervenir sobre este desarrollo por medio de terminales adaptados al diálogo.

A fin de servir a varios usuarios, el computador puede operar en tiempo compartido; es decir, que dedica a cada usuario una parte de su tiempo con

una periodicidad determinada para que tenga la impresión de poseer enteramente la máquina.

Cuarta Generación (aprox. 1980-?)

No está bien definido el ingreso a una cuarta generación, incluso algunos autores hablan de una quinta, pero, de todos modos, lo que haremos es enunciar algunos avances que podrían ser los que caracterizaran una nueva generación. En los últimos tiempos se ha desarrollado la tecnología de circuitos integrados a gran escala (LSI) de modo muy avanzado, se redujo considerablemente el tamaño físico de las máquinas.

Se integró en un único chip, el procesador.

Las microcomputadoras, y su correspondiente *software*, inundaron el mercado y permitieron definitivamente el ingreso de la computación en los hogares.

En cuanto al almacenamiento auxiliar, discos de gran capacidad de almacenamiento que utilizan la tecnología láser de grabación: los CD-ROM, los discos ópticos de altísima capacidad, los tape backup (cintas magnéticas de alta densidad de almacenamiento), han sido desarrollos tecnológicos muy importantes relacionados al uso de microcomputadoras.

Se introdujeron conceptos como el de "máquina virtual", el cual tiende a simplificar la labor del programador: este no conoce más que una máquina ficticia o "virtual" que no presenta ni las limitaciones de la configuración del ordenador utilizado (sobre todo en cuanto a memoria central) ni las limitaciones debidas a la compartición del ordenador con otros usuarios (especialmente en el tiempo compartido). Se desarrollaron gran cantidad de sistemas a tiempo real de gran aplicación en la industria, ingeniería y temas vinculados al uso de este método de manejo de la información.

En cuanto a las metodologías de programación, a principios de los 80 se comenzó a utilizar la programación estructurada en un intento de organizar en módulos adecuadamente los programas que hasta entonces se realizaban sin metodologías que garantizaran la eficiencia, facilidad de cambio, depuración, etcétera.

El continuo desarrollo de las técnicas de diseño estructurado permitió que, a finales de los 80, se implementaran metodologías de Análisis Estructurado de Sistemas y con el fin de solucionar la crisis del *software* existente que se estaba produciendo se otorgó gran dedicación a la Ingeniería de *Software* y a

las herramientas de automatización de análisis, diseño y programación asistidos por computadora (CASE).

Se generalizó el uso de las bases de datos y los lenguajes apropiados para su manejo.

Hacia la misma época se desarrollaron sistemas expertos que manejan inteligencia artificial utilizando lenguajes como PROLOG o LISP, pero que hasta la fecha no son de uso generalizado.

A principios de la década del 90 en las universidades se comenzó a utilizar la programación orientada a objetos (OOP) y poco tiempo después se avanzó en cuanto al diseño, análisis y bases de datos orientadas a objetos. Si bien el precursor de la OOP fue el Smalltalk, enfoque puramente orientado a objetos, permitió que otros lenguajes (menos puristas) como el C++ o el Pascal orientado a objetos, lograran imponerse a mayor cantidad de usuarios y aplicaciones.

En cuanto a las técnicas de organización y explotación, las redes de microcomputadoras, en sus diferentes topologías, han inundado el mercado, suplendo en gran cantidad de aplicaciones el uso de las "minis" y de las "mainframes", logrando una espectacular optimización del uso de los recursos de los sistemas de cómputo.

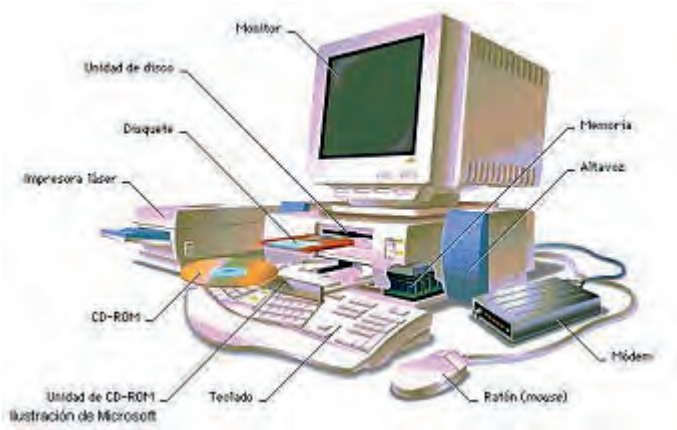


Figura 1.18: Computadora actual

Resumen cronológico

De una a otra generación, a lo largo de los años, se han conseguido progresos, en algunos casos considerables, en las características de los circuitos: miniaturización, fiabilidad, complejidad y velocidad.

- *La miniaturización*: se aprecia al advertir que la misma función lógica necesitaba un armario en la generación de las válvulas, un cajón —o parte de uno— en la generación de los transistores, una placa impresa para los circuitos integrados a pequeña escala, una cápsula de silicio en la generación de los circuitos integrados a gran escala.
- *La fiabilidad*: tal progreso en el terreno de la fiabilidad se debe a dos causas: el progreso en la fiabilidad del componente y la reducción, gracias a la integración, del número de interconexiones.
- *La complejidad*: la posibilidad de concebir conjuntos electrónicos cada vez más complejos es un corolario directo de la ganancia en fiabilidad.
- *La velocidad*: los tiempos de conmutación de los circuitos lógicos han pasado de los varios microsegundos de la primera generación a varios nanosegundos en la tercera. Esto ha permitido pasar de máquinas de un millar de instrucciones por segundo a máquinas, de complejidad equivalente, que ejecutan un millón de instrucciones por segundo.

En cuanto a los sistemas de explotación o sistemas operativos, reducidos al cargador y al traductor de lenguaje de la primera generación, han tomado a su cargo las funciones de encadenamiento de los trabajos y de gestión de las entradas y salidas a partir de la segunda y, posteriormente, a lo largo de la tercera la gestión de la multiprogramación, tiempo compartido, teleprocesamiento y la posibilidad eventual de trabajar en memoria virtual.

Antecedentes y precursores

- Ábaco (3000 a. C.)
- Desarrollo de logaritmos. Neper, 1617.
- Calculador de ruedas numéricas. Pascal, 1642.
  Máquina de sumar y restar.
- Rueda de pasos. Leibnitz, 1694.
  Máquina que realizaba las cuatro operaciones elementales.
- Utilización de tarjetas perforadas en los telares. Jacquard 1810.

- Máquina analítica. Babbage, 1830.
  Plantea la idea de ingresar datos e instrucciones por medio de tarjetas perforadas. Plantea una máquina que tenga memoria, unidad de control, unidad aritmética lógica, dispositivos de entrada y salida. Plantea concepto de programa exterior.
- Primer algoritmo. Augusta Ada King, 1843.
- Algebra booleana o lógica. Boole, 1854.
- Máquina tabuladora. Hollerith, 1890.
  Utilización de tarjetas perforadas y contadores electromecánicos para tabulación del censo poblacional de EE. UU.

Evolución de las computadoras

- ZUSE 23. Koward Zuse, 1936.
  Calculadora - ordenador (electromecánico con programa).
  Funcionamiento: relés
- Fundamento teórico para técnicas de programación. Alan Turing. 1936.
- Circuitos de conmutación eléctricos. Shannon. 1938.
- MARK I. Howard Aiken, 1944.
  Construida con elementos electromecánicos.
  Esquema lógico según propuesta de Babbage.
- ENIAC. Eckert y Mauchly, 1946.
  Primer ordenador electrónico, utilizaba tubos de vacío.
  Se utilizaban interruptores y tablero de conexiones de alambres para programar operaciones.
  Era capaz de realizar 5 mil operaciones por segundo.
- EDSAC. John Von Neumann, 1949.
  Primera computadora electrónica de programa almacenado.
- EDVAC.
- UNIVAC I. Eckert y Mauchly, 1951.
  Computadoras de programa almacenado.
  Gran velocidad, capacidad de memoria. Utilización de cintas magnéticas.
- Mercury. Ferranty, 1957.
- Clementina (Mercury). Primera computadora científica argentina, 1961-1971.
- Cristina (Cluster de 560 procesadores Intel Xeon 5420). Instalada en Córdoba, 2010. *Es 500 veces más poderosa que una computadora personal.*

Clementina: primera computadora en Argentina

Clementina fue la primera computadora en la Argentina, llegó al país en noviembre de 1960 y funcionó desde del año siguiente hasta 1971 cuando fue desmantelada en el Pabellón I de la Ciudad Universitaria de la UBA. La compra de Clementina fue la mayor inversión realizada en ciencia y tecnología en el país hasta ese momento y fue bien capitalizada por el Instituto de Cálculo que comenzó a funcionar orgánicamente en 1960. Reglamentado en 1962 como el primer instituto de la UBA, Manuel Sadosky lo dirigió desde su fundación hasta el golpe de estado de 1966

Básicamente estaba compuesta por 18 gabinetes, por lo que medía 18 metros de largo y ocupaba toda una habitación. Tardaba dos horas en encenderse y era 3.400.000.000.000 veces más lenta que una computadora de hoy. Por su gran tamaño y el sistema de refrigeración que necesitaba, se tuvo que modificar el edificio de la Ciudad Universitaria en el cual se instaló.

Clementina fue intensamente utilizada para aplicaciones de economía matemática, investigación operativa, estadística, análisis numérico y además facilitó la enseñanza de programación en la primera carrera universitaria de computación de toda América del Sur.

La computadora tenía ciclos (frecuencia reloj) de 1 MHz que le permitían hacer unos 30 mil cálculos cada segundo, tenía una memoria operativa de anillos de ferrita de 1024×40 bits (5 kB), un tiempo de acceso de 10 microsegundos para una palabra de 10 bits, memoria de archivo de cilindros magnéticos de 4×4×4900×40 bits (80 kB) y se programaba en un lenguaje propio de alto nivel, Mercury Autocode.

El origen del nombre fue la única característica multimedia del equipo: cuando terminaba un proceso de cálculo comenzaba a sonar la música de la popular canción estadounidense *Oh My Darling, Clementine*. Luego fue programada para que ejecutara un tango, se ignora cuál. A pesar de que su sonido fue modificado y adaptado al tango argentino conservó su nombre originario.

Clementina siguió en funcionamiento hasta mediados del año 1971, cuando su mantenimiento se hizo imposible por falta de piezas. Luego de su desmantelamiento, los restos se deshicieron como simples residuos. Tan solo unas pocas partes de la computadora fueron rescatadas por personal técnico de la facultad y aún los conservan como piezas de colección.

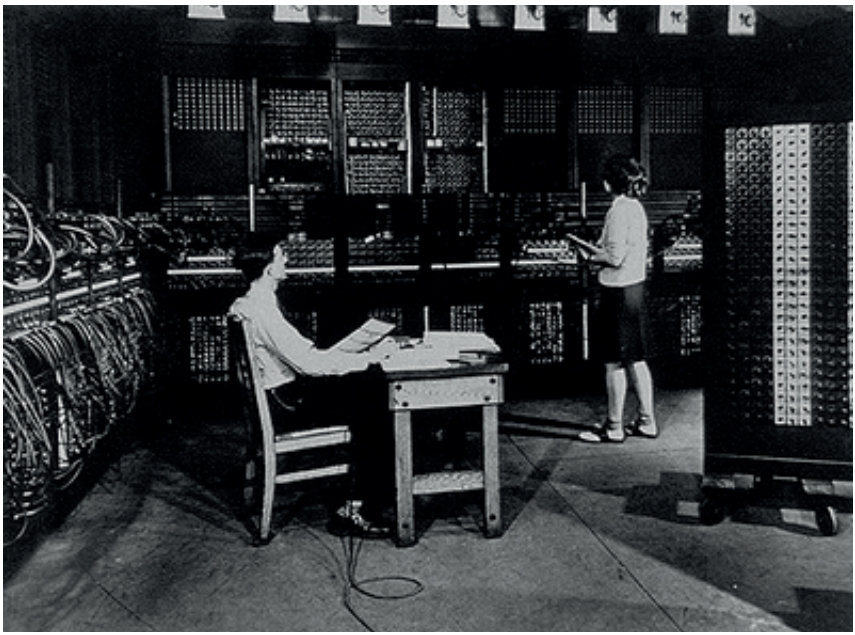


Figura 1.19: Clementina 1ra. Computadora Argentina

Manuel Sadosky

Manuel Sadosky fue un matemático argentino considerado por muchos como el padre de la computación en nuestro país.

Trajo la computadora Clementina al país y fue el creador de la carrera de Computador Científico (actual Licenciatura en Ciencias de la Computación).

En 1940 se graduó como Doctor en Ciencias Físico-Matemáticas de la UBA con una tesis en Matemática aplicada y enseguida comenzó a ejercer la docencia en esa Universidad.

Trajo la computadora Clementina al país, y fue el creador de la carrera de Computador Científico (Actual Licenciatura en Ciencias de la Computación)

Falleció el 18 de junio de 2005. Desde la Fundación Sadosky se realizaron diferentes acciones tendientes a hacer conocer su figura y mantener vivos sus ideales, buscando posicionar a la ciencia y la tecnología como motores fundamentales del desarrollo y crecimiento de la Nación.

FUNDAMENTOS MATEMÁTICOS

Definiciones

En este capítulo se presentarán la nomenclatura y los mecanismos matemáticos utilizados.

- La base de numeración de una expresión se indicará: $N_{(base)}$ o $N(_{base)}$.
- La sumatoria es un operador que permite definir la suma de varios términos de forma concisa. Utiliza la simbología $\sum_a^b c$, donde por lo general el campo *a* para indicar los índices y los valores iniciales, el campo *b* para los valores finales, y en *c* se indica el término, en función de los índices indicados en *a*.

$$a_0 + a_1 + a_2 + \ldots + a_n = \sum_{i=0}^{n} a_i$$

$$a_0 \times x^0 + a_1 \times x^1 + \ldots, a_n \times x^n = \sum_{i=0}^{n} a_i \times x^n$$

$$a \times b = \sum_{i=1}^{b} a$$

- Los logaritmos permiten conocer el valor de x en expresiones de la forma $B^x = N$, es decir, a qué número se debe elevar una base dada (B) para obtener el número N.

  Otra forma de expresarlo es $\log_B x = n \Leftrightarrow x = Bn$.
  Los logaritmos más notables son el logaritmo en base 10, log10 o log, y el logaritmo natural (ln) donde la base es e = 2.71828182846..., conocido el número de Euler o la constante de Napier.

- Algunas propiedades de los logaritmos son las siguientes:

$$\log_B(x * y) = \log_B(x) + \log_B(y)$$
$$\log_B(x^n) = n \times \log_B(x)$$
$$\log_B(x/y) = \log_B(x) - \log_B(y)$$
$$\log_B(B) = 1 \; \forall B \equiv 1 \text{ para todo } B$$
$$\log_B(x) = \frac{\log_K(x)}{\log_K(B)}$$
$$\log_B(1) = \frac{\log_K(k)}{\log_K(B)} = \frac{1}{\log_K(B)}$$

- Es conveniente tener presente las propiedades de las potencias para poder agilizar las operaciones:

$$a^x \times b^x = (a \times b)^x$$
$$a^x \times a^y = a^{x+y}$$
$$a^{-x} = 1/a^x$$
$$a^x/b^x = \left(\frac{a}{b}\right)^x$$

Ecuaciones

Una posible definición de ecuación sería la siguiente: "Una igualdad entre dos expresiones que contiene una o más variables".

Las ecuaciones se vuelven útiles cuando uno es capaz de operar sin modificar la igualdad a fin de especificar una relación contenida en ella u obtener el valor de una de las variables (procedimiento denominado "despejar"). El proceso siempre consiste en aplicar una operación que no modifique la igualdad de la expresión. Usaremos la expresión $A = B$ para ilustrar la operatoria.

- Multiplicar por la unidad.

$$
\begin{aligned}
A &= B \times 1 \\
A &= B \times \frac{\alpha}{\alpha} \\
\frac{A}{\alpha} &= \frac{B}{\alpha} && \text{(Dividir ambos miembros.)} \\
\alpha \times A &= B \times \alpha && \text{(Multiplicar ambos miembros.)}
\end{aligned}
$$

- Sumar cero.

$$
\begin{aligned}
A &= B + 0 \\
A &= B + \beta - \beta \\
\beta + A &= B + \beta && \text{(Sumar a ambos miembros.)} \\
-\beta + A &= B - \beta && \text{(Restar a ambos miembros.)}
\end{aligned}
$$

- La forma general, aplicar la misma función a ambos miembros.

$$
\begin{aligned}
f(A) &= f(B) \\
cte + A &= B + cte && f(x) = x + cte \text{ (Sumar una constante.)} \\
0 + A &= B + 0 \\
cte \times A &= B \times cte && f(x) = x \times cte \text{ (Multiplicar por una constante.)} \\
1 \times A &= B \times 1 \\
\log(A) &= \log(B) && f(x) = \log(x) \\
\ln(A) &= \ln(B) && f(x) = \ln(x) \\
2^A &= 2^B && f(x) = 2^x
\end{aligned}
$$

Teoría de conjuntos

La teoría de conjuntos permite generalizar una grámatica para el agrupamiento de objetos.

Indicaremos que un conjunto *A* tiene los elementos *a*, *b* y *c*, con la forma: *A* = {*a*,*b*,*c*}. Es decir, con letras mayúsculas los conjuntos y con minúsculas los elementos de cada conjunto. En un conjunto no existen repetidos.

Además, se puede definir un conjunto mínimo de operadores:

- Proposiciones lógicas: $p \wedge q$ (p y q), $p \vee q$ (p o q), $\neg p$ (no p), $\underline{\vee}$ (p ó q, excluyente).
- Existenciales: $a \in A$ (a existe en A).
- Agrupadores: $\forall$ (para todos), $\exists$ (existe), $\exists!$ (existe solo uno).
- Definir: $exp_2 := exp_1$, define la expresión 2 en base a la expresión 1.

Luego se pueden utilizar los operadores anteriores para definir construcciones más elaboradas:

- Implicación: $p \Longrightarrow q := (p \vee \neg q)$
- Subconjunto: los elementos de A están contenidos en B.
  $A \subset B := x_i \in B \, \forall x_i \in A$
- Elementos no contenidos en un conjunto. Si el conjunto U corresponde al universo, y se cumple que A es un subconjunto de U.
  $a \notin A := a \in U \wedge \neg(a \in A)$
- Intersección: el conjunto con los elementos comunes a dos conjuntos.
  Dados $A = \{a_i\}$ y $B = \{b_i\}$; $x_i \in A \cap B \Leftrightarrow x_i \in A \wedge x_i \in B$.
- Unión: el conjunto con los elementos de ambos conjuntos.
  Dados A = $\{a_i\}$ y $B = \{b_i\}$; $x_i \in A \cup B \Leftrightarrow x_i \in A \vee x_i \in B$.
- Conjunto Complementario: todos los elementos del universo que no están contenidos en A, a este conjunto lo expresaremos con $\overline{A}$. Y siempre se cumplirá $a \in A \cap \overline{A}$, que es otra forma de expresar el conjunto universo.
- El conjunto vacío es muy utilizado, a tal punto que amerita tener su propio símbolo: $\emptyset := \{\}$.

Prefijos de unidades

El Sistema Internacional de Unidades solo define los prefijos multiplicadores de la base decimal.

Pero recomienda el uso de los prefijos del estándar *ISO/IEC 80000-13* para indicar multiplicadores en potencias de 2, para evitar confusiones y errores. En los cuadros [tab:prefijosSI] y [tab:prefijosBinarios] se muestran algunos de los prefijos y las magnitudes que representan.

Para cuantificar el error de utilizar *k* en lugar *Ki*, podemos usar la relación $1024/1000 = 1,024$ que corresponde al $2,4\%$.

En cambio, sí cometemos el mismo error con *G* y *Gi* superaremos el 15% ($1024^3/1000^3 = 1,024^3 \approx 1,153$).

Cuadro 2.1: Prefijos del Sistema Internacional de Unidades

PREFIJO	NOMBRE	MAGNITUD
n	nano	10^{-9}
μ	micro	10^{-6}
m	mili	10^{-3}
c	centi	10^{-2}
d	deci	10^{-1}
		$10^{0} = 1$
da	deca	10^{1}
h	hecto	10^{2}
k	kilo	$10^{3} = 1000^{1}$
M	mega	$10^{6} = 1000^{2}$
G	giga	$10^{9} = 1000^{3}$
T	tera	$10^{12} = 1000^{4}$

Cuadro 2.2: Prefijos del estándar ISO/IEC 80000-13.

PREFIJO	NOMBRE	MAGNITUD
		$2^{0} = 1$
Ki	kibi	$2^{10} = 1024$
Mi	mibi	$2^{20} = 1024^{2}$
Gi	gibi	$2^{30} = 1024^{3}$
Ti	tibi	$2^{40} = 1024^{4}$

SISTEMAS DE NUMERACIÓN

Historia de los sistemas de numeración

La necesidad de contar es tan antigua como el hombre. El ser humano aprendió a contar con los medios a su alcance, utilizando, por ejemplo, sus dedos, piedras, o cualquier otro conjunto de objetos, que las distintas culturas trataron de representar simbolizando de diversas maneras su valor numérico.

Una manera de representar el número de elementos que constituyen un cierto conjunto es establecer una correspondencia con un número igual de símbolos.

Transcurrió un largo tiempo hasta que el hombre intentara asociar, con éxito, a cada cantidad un símbolo, así nacieron los *sistemas de numeración*.

Un *sistema de numeración* es un conjunto de símbolos y reglas que se utilizan para representar un número.

$$SN = (S, R)$$

Llamamos número a la representación de una cantidad cualquiera.

Figura 3.1: Algunos sistemas de numeración.

Para que un sistema de numeración llegue a ser de uso generalizado debe ser práctico y contemplar algunas de las siguientes características:

- Conjunto finito de elementos: debe contener pocos símbolos (las cantidades grandes se expresarán con la combinación de ellos); por esto, cuando fue necesario representar gráficamente muchos elementos, se trató de utilizar la menor cantidad de signos posibles, entre los cuales existían operaciones implícitas. Así, los romanos utilizaron un sistema en el cual los signos crecientes I, V, X, L, C, D, M se iban agregando de derecha a izquierda y se sumaban entre sí:

CXVII = cien + diez + cinco + uno + uno = 117

Y se restaban entre sí dos signos que en el conjunto no siguieran el orden creciente:

MCMV = mil + (mil - cien) + cinco = 1905

Esta notación tiene el inconveniente de que se necesitan nuevos símbolos a medida que los números se hacen mayores y, además, los cálculos resultan sumamente engorrosos. Por otra parte, no se utilizaba el cero. Que un sistema de numeración disponga de pocos símbolos es un obstáculo para la representación de cantidades grandes, esto se consigue combinando adecuadamente los símbolos. Como la naturaleza ha dotado al hombre de las manos como su mejor herramienta, siempre tuvo la tendencia a utilizarlas para contar; por lo tanto, es natural que el sistema numérico decimal que usamos se base en el número de dedos que poseemos

- Ser práctico: esto significa que las reglas que lo componen deben ser claras y concisas.
- Libre de ambigüedades: debe existir una correspondencia biunívoca entre símbolo y cantidad. Una cantidad no puede referir a más de un símbolo, ni un símbolo, representar más de una cantidad.
- Existencia del elemento nulo o vacío.

Acostumbrados, como estamos, al uso del sistema decimal, como dijimos, llamamos *número* a la representación en este sistema de una cantidad cualquiera. Sin embargo, corresponde decir que *número* es una cantidad abstracta que no depende del sistema de representación elegido.

Los sistemas de numeración se pueden clasificar en tres tipos:

- No posicionales: son aquellos en los que, indistintamente de la ubicación del símbolo de una cifra, siempre se toma su valor atómico. Ejemplos son sistemas como el egipcio, el griego o el azteca.

Figura 3.2: Sistema de numeración no posicional.

Si tomamos el sistema de la figura las cifras donde ∩ equivale a 10 y I equivale a 1:

II∩ es igual a 12.
I ∩ *I* es igual a 12.
∩*II* es igual a 12.

- Semiposicionales: son aquellos en los que, si bien lo más importante es el valor del símbolo, en algunas cifras el lugar que ocupa el símbolo tiene su importancia. El sistema romano es un ejemplo, ya que, si bien las cifras se arman según el valor del símbolo, no es lo mismo el valor que ocupa la I a la izquierda o a la derecha de la V, en el número 4 (IV) que en el 6 (VI). Sistema de numeración semiposicional.

- Posicionales: son aquellos en los que, tanto el valor del símbolo como su ubicación en el número tienen igual importancia. El sistema indoarábigo o maya, como los actuales sistemas de numeración, son posicionales.

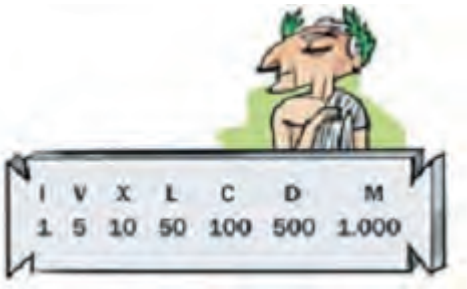


Figura 3.3: Sistema de numeración posicional.

En el sistema decimal de numeración, pueden representarse diez números (incluyendo el cero) con un solo dígito: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Este sistema fue ideado en la India, hacia el siglo VI a. C. y llevado a Europa por los árabes en la Edad Media. Diremos que su base es diez (10), ya que este es el número de símbolos distintos que utiliza el sistema. Desde el punto de vista aritmético se puede establecer un sistema, de igual naturaleza que el decimal, tomando como base cualquier número mayor que 1.

Sistemas posicionales

Los pueblos orientales desarrollaron los *sistemas posicionales*, los cuales, a partir de un número limitado de símbolos, llamado *raíz* o *base* del sistema, permiten representar cualquier número; existe el 0 para representar ausencia de elementos. Estos sistemas ofrecen una notación compacta y relativamente cómoda para expresar números grandes destacándose la facilidad para realizar operaciones aritméticas y computacionales.

Cada símbolo, además del valor que posee aisladamente, tiene un significado o peso según la posición que ocupa dentro del número del que forma parte. Las potencias de la base aumentan de derecha a izquierda, comenzando por la potencia 0 (cero), y disminuyen de derecha a izquierda (decimales):

$$\ldots 10^3, 10^2, 10^1, 10^0; 10^{-1}, 10^{-2}, etc.$$

De esta manera es posible representar, sistemáticamente, cualquier número empleando, en forma combinada, un conjunto limitado de caracteres. Es importante el valor de cada dígito, el cual se determina por su posición; por ejemplo, el 2 en 2.000 tiene un valor diferente del 2 en 20.

La cantidad de símbolos permitidos en un sistema de numeración posicional se conoce como *base*. Un sistema con X cantidad de símbolos se dice que está en base X. Cualquier entero $Z > 1$ puede constituirse en una base.

Sistema decimal

En el sistema decimal pueden representarse diez números con un dígito. Fue ideado en la India en el siglo VI a. C. y llevado a Europa por los árabes en la Edad Media. Al escribir un número en el sistema decimal observamos lo siguiente:

$$9402 = 9 \cdot 10^3 + 4 \cdot 10^2 + 0 \cdot 10^1 + 2 \cdot 10^0$$

- Existen solo diez símbolos {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} para ocupar cualquier posición, por lo cual, la base del sistema es 10.
- Los diez símbolos combinados permiten representar los números conforme a una convención que atribuye un valor individual y otro posicional a cada símbolo.
- Cada dígito indica cuántos subconjuntos {0, 1, … ,9}, que agrupan igual número de elementos (uno, diez, cien, mil, …), se han utilizado para formar un número decimal. Existen tantos tipos de subconjuntos como dígitos contenga un número. En el ejemplo se observan 9 subconjuntos que agrupan mil + 4 subconjuntos que agrupan cien + 0 subconjuntos que agrupan diez + 2 subconjuntos que agrupan uno.
- El número está formado por los dígitos que representan los agrupamientos de cada tipo, ordenados (a partir del dígito correspondiente a los mayores agrupamientos que pueden formarse) todos los que le siguen en tamaño hasta llegar a las unidades.
- En un sistema posicional, por lo tanto, el valor de cada dígito depende de su posición en el número. A cada posición le corresponde siempre una misma potencia de la base o "peso".
- Las potencias de la base aumentan de derecha a izquierda, comenzando por la potencia cero: 10^3, 10^2, 10^1, 10^0, etcétera.

- El número representado supone la suma de los productos de cada dígito por el peso correspondiente.

Entonces, la regla general para la representación de números en el sistema decimal, utilizando la notación posicional, es la que sigue:

$$N_{(10)} = d_{m-1} \cdot 10^{m-1} + d_{m-2} \cdot 10^{m-2} + \ldots + d_2 \cdot 10^2 + d_1 \cdot 10^1 + d_0$$

donde *m* indica el número de dígitos a izquierda del punto decimal.

Números de bases arbitrarias

Cualquier entero positivo ($r > 1$) se puede emplear como la base de un sistema de numeración posicional que sea semejante al sistema decimal, cuya base es $r = 10$. Los dígitos en el sistema de numeración de base *r* son los números {0, 1, 2, 3, 4, *r*-1}.

En general, para cualquier base con *r* símbolos, denominados dígitos (d_i), un número $N_{(r)}$ en esa base es una expresión del tipo:

$$N_{(r)} = \sum_{i=0}^{m-1} d_i \cdot r^i = d_{m-1} \cdot r^{m-1} + d_{m-2} \cdot r^{m-2} + \ldots + d_2 \cdot r^2 + d_1 \cdot r^1 + d_0$$

El subíndice *i* denota las *m* posiciones del número; d_{m-1}, d_{m-2}, d_1, d_0 son dígitos.

El número puede representarse también como sigue:

$$N_{(r)} = (d_{m-1}\, d_{m-2}\, d_1\, d_0)$$

Como vimos entonces, en el sistema decimal se representan los números como suma de potencias de 10.

El mismo criterio se aplica con igual validez a las fracciones decimales, usando potencias negativas de 10, las cuales crecen en valor absoluto, de izquierda a derecha, a partir de la coma decimal.

$$0,573 = 5 \cdot 10^{-1} + 7 \cdot 10^{-2} + 3 \cdot 10^{-3}$$

Genéricamente:

$$Nf_{(r)} = \sum_{i=-1}^{n-1} d_i \cdot r^i = d_{-1} \cdot r^{-1} + d_{-2} \cdot r^{-2} + ... + d_{-n} \cdot r^n$$

Los números con parte entera y fraccionaria combinan ambas expresiones:

$$1372,25 = 1 \cdot 10^3 + 3 \cdot 10^2 + 7 \cdot 10^1 + 2 \cdot 10^0 + 2 \cdot 10^{-1} + 5 \cdot 10^{-2}$$

Resulta, genéricamente:

$$N_{(r)} = \sum_{i=-n}^{m-1} d_i \cdot r^i$$
$$= d_{m-1} \cdot r^{m-1} + d_{m-2} \cdot r^{m-2} + ... + d_0 + d_{-1} \cdot r^{-1} + d_{-2} \cdot r^{-2} + ... + d_{-n} \cdot r^n$$

Sistemas más usados en computación

Sistema binario

Las máquinas digitales funcionan mediante dispositivos de dos posiciones; es decir, pueden considerarse en dos estados: encendido (1) o apagado (0) como los relés e interruptores.

Haciendo una semejanza entre el uso de dos estados para los circuitos electrónicos con el 0 y el 1, vemos la conveniencia de utilizar un sistema binario de numeración, es decir de base 2.

Los dígitos del sistema binario son 0 y 1. Los valores posicionales de estos dígitos o bits (binary digits) en la representación binaria de un número, son potencias de 2. De lo dicho se desprende que un número escrito en un sistema binario será una sucesión de ceros y unos.

Características:

- Base: 2
- Símbolos: {0, 1}

Cada bit es una pieza de información digital. Se requieren más dígitos binarios que decimales para simbolizar un mismo número de elementos; la simplicidad y velocidad de los circuitos digitales electrónicos compensa ampliamente esta desventaja relativa de la representación binaria.

El método empleado en la representación es el mismo que el visto para la base 10. Solo varía el tamaño de los agrupamientos (que existen infinitos), cada grupo es el doble que el anterior. En este caso está permitido formar un solo agrupamiento de cada tamaño (en base 10 hasta nueve agrupamientos).

Para formar un número en la base 2, partiremos de considerar cómo lo hacemos en el sistema decimal: una vez que se escribieron todos los símbolos correspondientes a las unidades (del 0 al 9), se comienzan a combinar de a dos en forma creciente (del 10 al 99), luego se combinan de a tres (del 100 al 999), etcétera. De manera análoga, en base 2, luego de escribir los símbolos del sistema (0, 1) se deben combinar de a dos (10 y 1). El 11 sería como el 99, por lo cual, después del 11 viene el 100, sigue el 101, 110 y 111.

Sistemas octal y hexadecimal

Existen otros dos sistemas numéricos, muy útiles en computación: el octal y el hexadecimal. Ambos permiten representar rápidamente un número binario con menos dígitos; es decir, de una manera más compacta. (Hay que recordar que 8 y 16, bases de estos sistemas, son potencias enteras de 2).

En el interior de una computadora solo existe información digital binaria, que forma combinaciones de unos y ceros, compuestos de 8, 16, 32, 64 bits. Estas largas sucesiones de unos y ceros son difíciles de leer y pueden llevar a error al manipularlo las personas. Por lo dicho, estos sistemas facilitan lo siguiente:

- La representación de números, dado que se emplea un número menor de símbolos que en binario (3ª y 4ª parte), redunda en escritura más veloz y con menos errores.
- El pasaje en forma directa a binario y viceversa, por ser una base potencia entera de la otra.

Sistema octal

- Base: 8
- Símbolos: {0, 1, 2, 3, 4, 5, 6, 7}
- Valores absolutos: 0 a 7 respectivamente, siendo el número:

$$8_{(10)} = 10_{(8)}; 9_{(10)} = 11_{(8)}; 10_{(10)} = 12_{(8)}$$

- Los valores posicionales de un número octal para calcular su equivalente decimal serán los que siguen: 8^0, 8^1, 8^2, 8^3, etcétera.

Sistema hexadecimal

- Base: 16
- Símbolos: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}
- Valores absolutos: 0 a 15, respectivamente.
- El valor decimal de un número escrito en hexadecimal se calcula teniendo en cuenta los valores posicionales: $16^0, 16^1, 16^2, 16^3$, etcétera.

Cuadro 3.1: Tabla de equivalencias entre sistemas de numeración

DECIMAL	BINARIO	OCTAL	HEXADECIMAL
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Es importante tener bien presente esta equivalencia puesto que estos sistemas son extensamente usados en computación.

De la tabla se deduce que, cuanto menor es la base, mayor es el número de símbolos necesarios para expresar el mismo número. Se observa además que, en cualquier base, una vez que se escribieron ordenadamente todos los símbolos que la constituyen, sigue la combinación de símbolos 10 que representan un conjunto de elementos igual al valor de dicha base. A partir de 10, comienzan a escribirse de a dos los símbolos de un sistema numérico.

El sistema binario es el único que utiliza internamente la máquina para efectuar los cálculos, ya que, según lo dicho, maneja dos estados: apagado/

prendido, 0/1. Los sistemas octal y hexadecimal se emplean para representar los números en forma más compacta cuando necesitamos interpretar el contenido interno de la memoria o de los registros que utiliza el computador.

El sistema hexadecimal se estructura formando agrupaciones tales que cada una sea dieciséis veces mayor que la anterior, mientras que el sistema octal lo hace formando agrupaciones tales que cada una sea ocho veces mayor que la anterior.

Síntesis de sistemas posicionales

A modo de conclusión podemos citar las siguientes características de un sistema posicional:

- Consta de un número finito de símbolos distintos que define la base del sistema o raíz.
- Cada símbolo aislado representa un número especificado de unidades.
- Existe un elemento 0 para indicar la ausencia de elementos.
- Un mismo símbolo tiene una significación o "peso" distinto según su posición.
- La posición extrema derecha corresponde a unidades (peso 1). A partir de ella, cada posición tiene el peso de la que está a su derecha multiplicada por la base.

Cambio de base

Cambiar de base significa encontrar el valor equivalente de un número en una base diferente de la que está expresado.

Recordando lo visto para sistemas posicionales, un número puede expresarse como:

$$\begin{aligned} N_{(r)} &= \sum_{i=0}^{m-1} d_i \cdot r^i \\ &= d_{m-1} \cdot r^{m-1} + d_{m-2} \cdot r^{m-2} + d_{m-3} \cdot r^3 + \ldots + d_2 \cdot r^2 + d_1 \cdot r^1 + d_0 \end{aligned}$$

Si se tratara de un número decimal la expresión sería así:

$$\begin{aligned} N_{(10)} &= d_{m-1} d_{m-2} \ldots d_2 d_1 d_0 \\ &= d_{m-1} \cdot 10^{m-1} + d_{m-2} \cdot 10^{m-2} + \ldots + d_2 \cdot 10^2 + d_1 \cdot 10^1 + d_0 \end{aligned}$$

Si estuviera en otra base, por ejemplo, la base 2, la expresión sería de la siguiente manera:

$$N_{(2)} = \sum_{i=0}^{k-1} b_i \cdot 10^i$$
$$= b_{k-1} \cdot 10^{k-1} + b_{k-2} \cdot 10^{k-2} + \ldots + b_2 \cdot 10^2 + b_1 \cdot 10^1 + b_0$$

Tener en cuenta que para una base cualquiera *r*; $r_{(10)} = 10_{(r)}$. En particular para el sistema binario, $2_{(10)} = 10_{(2)}$. Siempre y cuando 0 y 1 se correspondan al conjunto vacío y a la unidad, respectivamente.

Como vemos, pueden expresarse los números como sumatorias de productos cuyos factores y potencias estén representadas en la base de que se trate. Pero es común escribir por comodidad un número dado en una base *r* como una sumatoria cuyos factores y productos están en base 10.

Como en general no estamos acostumbrados a hacer operaciones complejas en sistemas distintos del decimal, se recurre a algoritmos que permiten pasar de un sistema a otro mediante operaciones simples.

En general, la equivalencia en base 10 de un número binario podrá expresarse:

$$N_d = (b_{k-1}\, b_{k-2} \ldots b_2\, b_1\, b_0)_{(2)}$$
$$N_d = \left(\sum_{i=0}^{k-1} b_i \cdot 2^i\right)_{(10)} = \left(b_{k-1} \cdot 2^{k-1} + b_{k-2} \cdot 2^{k-2} + \ldots + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0\right)_{(10)}$$

Si, en cambio, conocemos el número en base 10, el problema consiste en determinar el valor de las incógnitas $\{b_{k-1}, b_{k-2}, \ldots, b_2, b_1, b_0\}$ y cuál es su longitud (valor de k).

Pasaje entre dos bases r_1 y r_2

Dado un número definido en la base r_1:

$$N_{(r_1)} = b_k \cdot r_2^k + b_{k-1} \cdot r_2^{k-1} + \ldots + b_2 \cdot r_2^2 + b_1 \cdot r_2^1 + b_0$$

Con los valores $\{b_i\}$ y r_2 expresados en la base r_1.

El algoritmo general que permite hallar los dígitos de una nueva base r_2 a la que se quiere pasar, lo que permite deducir:

Sacando r_2 factor común:

$$N_{(r_1)} = \underbrace{r_2 \cdot \left(b_{k-1} \cdot r_2^{k-2} + \ldots + b_2 \cdot r_2 + b_1\right)}_{N1_{(r_1)}} + b_0 = N1_{(r_1)} + b_0$$

De acuerdo a la definición de cociente, b_0 será el resto de dividir el número originario $N_{(r1)}$ por r_2 en la base r_1.

$$\begin{array}{l|l} N_{(r_1)} & r_2 \\ b_0 & N1_{(r_1)} \\ 0 & \\ \vee & \end{array}$$

Por lo tanto, la incógnita b_0 es el resto de dividir el $N_{(r1)}$ por r_2 y $N1_{(r1)}$ es el cociente.

$$N1_{(r_1)} = \left(b_{k-1} \cdot r_2^{k-2} + \ldots + b_2 \cdot r_2 + b_1\right)$$

Si nuevamente r_2 se utiliza de factor común:

$$N1_{(r_1)} = \underbrace{r_2 \cdot \left(b_{k-2} \cdot r_2^{k-1} + \ldots + b_3 \cdot r_2 + b_2\right)}_{N2_{(r_1)}} + b_1 = N2_{(r_1)} + b_1$$

De lo que se observa que la incógnita b_1 es el resto de dividir el polinomio $N1_{(r1)}$ por r_2 y $N2_{(r1)}$ es el cociente.

Si se continúa dividendo sucesivamente por r_2 se arribará a un cociente nulo:

$$Nn_{(r_1)} = r_2 \cdot 0 + b_k = b_k$$

De esta manera, se obtuvieron los dígitos $\{b_0, b1, \ldots, b_k\}$ de la base r_2 que se querían encontrar y el número binario obtenido es:

$$N_{(b)} = b_{k-2}\, b_{k-1}\, \ldots\, b_{11}\, b_{10}\, b_1\, b_0$$

Cambio de base de un número fraccionario

En un sistema posicional, un número fraccionario que se conoce en una base r_1 y quiere pasarse a una base r_2 puede representarse como:

$$Nf_{(r_1)} = b_1 \cdot r_2^{-1} + b_2 \cdot r_2^{-2} + \ldots + b_{-n} \cdot r_2^{-n}$$

Según el algoritmo general.

$\{b_{-1}, b_{-2}, \ldots, b_{-n}\}$= dígitos del número en la nueva base, que se quieren determinar.

Multiplicando ambos miembros por r_2:

$$Nf_{(r_1)} = b_1 + \left(b_2 \cdot r_2^{-1} + \ldots + b_{-n} \cdot r_2^{-n+1}\right) = b_{-1} + Nf1_{(r_1)}$$

Siendo b_1 un número entero y $Nf1_{(r1)}$ su parte fraccionaria, por contener potencias negativas de r_2.

A $Nf1_{(r1)}$ se le puede multiplicar por r_2 nuevamente, obteniéndose b_2 y una nueva fracción $Nf2_{(r1)}$.

El proceso puede continuar hasta obtener una parte fraccionaria nula o hasta una precisión determinada.

Pasar el número decimal 0, 83510 a base 2:

$$\left.\begin{array}{l} 0,835 \times 2 = 1,670 \implies d_1 = 1 \\ 0,670 \times 2 = 1,340 \implies d_2 = 1 \\ 0,340 \times 2 = 0,680 \implies d_3 = 0 \\ 0,680 \times 2 = 1,360 \implies d_4 = 1 \\ 0,360 \times 2 = 0,720 \implies d_5 = 0 \\ 0,720 \times 2 = 1,440 \implies d_6 = 1 \end{array}\right\} \implies 0,835_{(10)} = 0,110101_{(2)}$$

Si un número tiene parte entera y parte fraccionaria, se cambia de base, separadamente cada una de ellas, y se escriben ambos resultados, separados por la coma correspondiente.

Operaciones entre números naturales de una base cualquiera

Definido un sistema de numeración posicional con una base r_1 y un conjunto de símbolos, es posible construir, tal como lo hacemos con los números decimales, tablas para las operaciones de suma y multiplicación que indiquen el resultado respectivo para cada par de dígitos:

+	0	1
0	0	1
1	1	1

×	0	1
0	0	0
1	0	1

Cuadro 3.2: Tabla de suma y multiplicación binaria

+	0	1	2	3
0	0	1	2	3
1	1	2	3	10
2	2	3	10	11
3	3	10	11	12

×	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	10	12
3	0	3	12	21

Cuadro 3.3: Tabla de suma y multiplicación para base $r = 4$

Con estos elementos es posible resolver sumas, restas, multiplicaciones y divisiones de dos o más números en una determinada base, utilizando los mismos algoritmos que estamos habituados a utilizar cuando operamos con base 10.

Suma y resta binaria

La suma binaria se realiza de la misma manera que la suma decimal. Las *cifras que se llevan* se consideran igual, pero como 1 es el dígito más grande en el sistema binario, cualquier suma mayor de 1 requiere que se traslade un dígito a la siguiente posición.

Ejemplo: de suma decimal y binaria

Decimal	Binario	Decimal	Binario	Decimal	Binario
5	101	15	1111	3,25	11,01
+ 6	+ 110	+ 20	+ 10100	+ 5,75	+ 101,11
11	1011	35	100011	9,00	1001,00

Según vemos: 1 + 1 = 0 más un 1 que se traslada; 1 + 1 + 1 = 1 "llevando" 1, ya que 1 + 1 = 10 y 1 + 1 + 1 = 11.

La resta es la operación inversa a la suma. Para restar es necesario establecer un procedimiento para sustraer un dígito grande de uno menor. El único caso que se presenta en los números binarios es cuando 1 se resta de 0; el residuo es 1, pero es necesario pedir prestado un 1 de la siguiente columna de la izquierda:

Por ejemplo, en la próxima resta se tomará prestado 1 de la tercera columna, debido a la diferencia 0 - 1 en la segunda columna:

111101	110001
-10010	-1010
101011	100111

Ejemplo: de resta decimal y binaria

Decimal	Binario	Decimal	Binario	Decimal	Binario
9	1001	16	10000	6,25	110,01
- 5	- 101	- 3	- 11	- 4,50	- 100,10
4	100	13	1101	1,75	1,11

Multiplicación y división binarias

Los siguientes ejemplos de multiplicación binaria ilustran la sencillez de cada operación. Solamente es necesario copiar el multiplicando si el dígito del multiplicador es 1 y copiar todos los 0 (o correr un lugar a la izquierda), si el dígito del multiplicador es 0.

Ejemplo: de multiplicación binaria

Decimal		Binario		Decimal		Binario
12		1100		102		1100110
× 10	×	1010	×	8	×	1000
120		0		816		0
		1100				0
		0				0
		1100				0
						1100110

Una definición posible de la división binaria es:

÷	0	1
0	-	0
1	-	1

Cuadro 3.4: Tabla de división binaria

Ejemplo: de división binaria

```
10010 | 110
 110  | 11
00110
 110
   0
   V
```

Suma en hexadecimal

Es muy frecuente en la práctica que se necesite operar con números hexadecimales, puesto que el contenido de la memoria y los registros del procesador se representan externamente en forma hexadecimal.

La suma en este sistema obedece a las mismas reglas que los demás sistemas numéricos.

Hay que tener en cuenta que, al efectuar la suma de dos dígitos hexadecimales, dicha suma puede exceder de 16; si ello ocurre se escribe en esa

columna la diferencia entre el número resultante y 16 y se arrastra un "1" a la columna siguiente:

Ejemplo: de sumas hexadecimales

```
  9654      6AE      8A0F
+ 4528    + 1FA    + 9CD1
------    -----    ------
  DB7C      8A8     126E0
```

Resta en hexadecimal

Se procede como de costumbre, tomando una unidad de la columna siguiente, en caso de ser necesario

```
Ejemplo:   COA8
         -  31E
         ------
           BD8A
```

Complementos

La operación de complementación es importante, puesto que permite reducir el problema de la resta a una simple suma, aprovechando los circuitos sumadores existentes, así como la representación de los números negativos. Veremos cómo se resuelve el problema de la resta mediante la complementación del sustraendo.

Complemento auténtico o a la base

En primer lugar, definiremos qué es el complemento de un número.

Debe tenerse presente que las máquinas, en general, operan con un conjunto de números enteros, los cuales poseen un número fijo de dígitos. Los números menores completan con ceros a la izquierda el formato dado de n dígitos.

Por ejemplo, en base 10, trabajando con un número de $n = 3$ dígitos, el complemento a $1000 = 10^3$ de $B = 975$ es $B' = 025$, o sea, lo que le falta a 975 para ser 1000 (975+025=1000).

Se acostumbra a decir que 025 es el "complemento a la base" = B' del número $B = 975$. En este caso, en realidad es el complemento a la base 10 a la potencia 3.

Se podría calcular como: $B'_{975} = 1000 - 975 = 025$

Recíprocamente, 975 es el complemento a la base de B = 025; $B_{025}{}' = 975$.

En general, dada una base cualquiera en la que se opera con un conjunto de m dígitos, y dado un número B perteneciente al mismo, su complemento a la base llamado B', será otro número de ese conjunto tal que:

$B' = r^m - B$ se denomina "complemento a la base" de B o sea $B + B' = r^m$.

Si se trabaja en base 2 = 10_2, por ejemplo, con números de 4 dígitos, el complemento a la base (10^4 = 10000) del número B = 1001, resulta B' = 0111.

En este caso, el complemento a la base se denomina "complemento a dos" del mismo modo que puede llamarse complemento a diez en el sistema decimal.

Complemento restringido o a la base menos uno

En cualquier base, dado un número B de m dígitos, su complemento a la base menos uno es otro número tal que:

$\bar{B} = r^m - 1 - B$ con $\bar{B}$ el "complemento restringido de B", o sea $B + \bar{B} = r^m - 1$.

Es más sencillo, por lo general, hallar el complemento restringido (o a la base menos uno) de un número que el complemento auténtico. Por este motivo, es frecuente hallar el primero y, a partir de este, el segundo.

Realizaremos un ejemplo hallando en base 10 el complemento restringido del número B = 975. Será el complemento a la base 10 elevada a la potencia n = 3 menos 1 (complemento a 9 para el caso de esta base).

$10^3 - 1 = 1000 - 1 = 999$, por lo tanto, $B + \overline{B} = 999$

Luego, $\overline{B} = 999 - 975 = 024$

Si le sumamos 1 a $\overline{B}$ = 024 obtenemos el complemento auténtico B' = 025, sin necesidad de hacer la resta 1000 - 975.

Por lo tanto, el complemento a la base o auténtico puede obtenerse sumando 1 al complemento restringido. Luego, para cualquier base será:

$$B' = \bar{B} + 1 \qquad (3.1)$$

En base 2, la obtención del complemento restringido es directa, ya que este se obtiene permutando los unos por ceros y ceros por unos.

Resta mediante el complemento

Sea la diferencia $d = A - B$ entre dos números naturales en base r y el mayor de ellos posee m dígitos. Si a ambos miembros se le suma r_m, se tiene:

$$r^m + d = r^m + (A - B) = A + \overbrace{(r^m - B)}^{B'}$$
$$r^m + d = A + B' \qquad (3.2)$$

El problema consiste en hallar d, conocida la suma $A + B'$.

Ejemplo de resta utilizando complemento en base 10:
Supongamos que queremos efectuar la siguiente resta:

$$\underbrace{54}_{A} - \underbrace{B}_{B} = \underbrace{22}_{d}$$

Utilizando complemento, si a ambos miembros le sumamos r^m o sea:

$$2^1 0 = 100\ 54 + \underbrace{100 - 32}_{r^m - B} = 22 + 100$$

Efectuando la diferencia 100 - 32 = 68, tenemos:

$$\underbrace{54}_{A} + \underbrace{68}_{B'} = \underbrace{22}_{d} + \underbrace{1000}_{r^m}$$
$$54 + 68 - 100 = 22$$
$$122 - 100 = 22$$

Vemos que eliminando el 1 de la izquierda, obtenemos el resultado de la resta inicial.

A continuación, veremos otro ejemplo de cómo obtener el resultado de una resta utilizando la suma de un número más el complemento del otro (en base $r = 2$, teniendo que A tiene $m = 5$ dígitos):

$$A - B = 11101 - 1110$$
$$r^m = 2^5 = 10000_2$$

Para evitar hacerla resta $B' = r^m - B = 100000 - 001110$, que puede resultar engorrosa. Por lo que se utiliza el siguiente artificio:

Sumando y restando 1:

$$r^m - B + 1 - 1 = \overbrace{(r^m - 1)}^{\bar{B}} - B + 1 = B'$$

Mediante lo cual obtenemos el complemento auténtico de B.

O sea, se calcula en primer lugar $(r^m - 1) - B$, denominado "complemento a la base -1" o "complemento restringido" de B (esto facilita la resta, como veremos en el ejemplo) y al resultado se le suma 1:

$$\begin{aligned} r^m - 1 &= 2^5 - 1 = 100000 - 1 = 11111 \\ \underbrace{(r^m - 1)}_{\bar{B}} - B &= 11111 - 01110 = 10001 \end{aligned}$$

Se observa que el "complemento a la base menos uno" de B puede obtenerse, directamente, permutando los unos por ceros y los ceros por unos de B, respectivamente:

$$B = 01110 \; \bar{B} = 10001$$

De allí la conveniencia de trabajar con complemento restringido en el sistema binario, debido a la facilidad de los cálculos.

Por lo tanto, el complemento restringido de B es:

$$\bar{B} = r^m - 1 - B \qquad (3.3)$$

Recordando [eq:RestaPorComplemento], se puede expresar $r^m + d = A + B'$, a continuación se realizará $A + B'$.

Basándonos en la ecuación [eq:BComplementoAutentico], podemos expresar:

$$B' = \bar{B} + 1$$
$$A + B' = A + (r^m - 1 - B) + 1 = A\bar{B} + 1$$

$$\begin{array}{rcl} A & = & 11101 \\ & + & \\ \bar{B} & = & 10001 \\ \hline A + \bar{B} & = & 101110 \\ & + & \quad 1 \\ \hline A + B' & = & 101111 \end{array}$$

Aplicando la expresión 3.2:

$$r^m + d = r^m + (A - B)$$
$$= 1 \cdot 2^5 + \left(0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^1 + 1\right)$$

Puesto que:

r^m = 100000 = $1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^3 + 0$, se deduce que el único aporte de r^m en la fórmula $r^m + (A - B)$ es el término $1 \cdot 2^5$, o sea, el dígito 1 en la posición de mayor peso, separado por la raya punteada.

Por consiguiente, el resultado buscado, es decir, la diferencia $A - B$, serán simplemente los dígitos que están al lado derecho de la línea punteada:

$A - B$ = 01111 de la misma forma que en la fórmula. (-3-)

Se ha supuesto $A > B$. Ahora veremos qué sucede si $A < B$, o sea, si la diferencia entre A y B es $A - B = -d$. Se hace la operación $A - B = -d$, según la regla anterior es:

$$A + B = A + (r^m - 1 - B) = r^m - 1 + (A - B) = r^m - 1 - d$$

Por lo que $A + B = r^m - 1 - d$, se observa, según (-4-) que el resultado obtenido es el complemento restringido del valor deseado.

Ejemplo:

d= 1110 - 11101

Con un procedimiento análogo al anterior:

A = 1110 B = 11101

Dado B = 11101, podemos hallar el complemento restringido permutando ceros por unos y unos por ceros $\overline{B}$ = 00010

$$\begin{array}{rcl} A & = & 0\ 1110 \\ & + & \\ \bar{B} & = & 0\ 0010 \\ A+\bar{B} & = & 1\ 0000 \\ & + & \quad 1 \\ A+B' & = & 10\,1111 \end{array}$$

Dado que $A + \overline{B} = \overline{d}$, para hallar d, se debe hallar el complemento del resultado hallado:

$\overline{d}$ = 10000 $\Longrightarrow$ d = -01111

En ambos casos, se ha obtenido una diferencia, mediante una complementación y una suma, que era el objetivo buscado. En síntesis, para restar dos números naturales cualquiera, $A - B$, se debe hallar primero $\overline{B}$ luego realizar la suma $A + \overline{B}$.

Si aparece en $A + \overline{B}$ el dígito 1 en la posición r^m, o sea, con un dígito más que los sumandos, resulta $A > B$, y para determinar $A - B$, se debe sumar 1 a $A + \overline{B}$ y el resultado se obtiene desechando el dígito 1 de la posición r^m.

Si no aparece dicho dígito, resulta $A < B$, y la diferencia $A - B$ se obtiene complementando el resultado de $A + \overline{B}$.

Veremos mediante ejemplos, paso a paso, el procedimiento práctico para hallar la diferencia $A - B$ en ambos casos ($A > B$ y $A < B$).

Conviene siempre, para evitar equivocaciones, igualar la cantidad de dígitos de A y B, agregando ceros a la izquierda del número que tenga menos dígitos.

Caso en que $A > B$

$$A = 23_{10} = 10111_2\ B = 12_10 = 1100_2$$

1. Se debe realizar la diferencia $A - B$

   Se halla el complemento restringido de B (recordar que se debe rellenar con ceros a la izquierda si la cantidad de dígitos es

menor que m). Dado $\overline{B}$=01100 $\Longrightarrow \overline{B}$ = 10011$. Permutando los ceros por unos y viceversa. Como A, que es el mayor de los dos, tiene 5 dígitos, m = 5, por ello, se debe rellenar B con un cero hacia la izquierda.

2. Se realiza la suma $A + \overline{B}$.

$$\begin{array}{rcl} A & = & 10111 \\ & + & \\ \bar{B} & = & 10011 \\ A+\bar{B} & = & 1\,101010 \end{array}$$

3. Como apareció un dígito 1 en la posición r^m, o sea, un dígito más que los sumandos, resulta $A > B$ y se debe sumar 1 a $A + \overline{B}$.

$$\begin{array}{rcl} A & = & 10111 \\ & + & \\ \bar{B} & = & 10011 \\ A+\bar{B} & = & 1\,101010 \\ & + & 1 \\ & & 1\,01011 \end{array}$$

4. Se desecha el dígito 1 de la posición r^m y el resultado es:
$A - B = 1011_2 = 11_{10}$

Caso en que $A < B$

$$B = 23_{10} = 10111_2 \; A = 12_10 = 1100_2$$

1. Se halla el complemento restringido de B, permutando ceros y unos:

$$B = 10111 \implies \bar{B} = 01000$$

2. Se realiza la suma $A + \overline{B}$

$$\begin{array}{rcl} A & = & 01100 \\ & + & \\ \overline{B} & = & 01000 \\ A+\overline{B} & = & 10100 \end{array}$$

3. Como no apareció un dígito en la posición r^m, o sea, un dígito más que los sumandos, resulta $A < B$ y se debe complementar el resultado de $A + \overline{B}$.

$$\overline{d} = A + \overline{B} = 10100$$
$$d = -01011 = A - B$$

4. Verificación: $12_{10} - 23_{10} = -11_{10}$.

Ejemplo de resta utilizando complemento, en base 10:

$$A - B = 173210 - 47910;\ A = 1732;\ B = 479$$

1. Se halla el complemento restringido de B. Teniendo en cuenta que $m = 4$ y $r = 10$.

$$\begin{aligned} \bar{B} &= r^m - 1 - B \\ \bar{B} &= 10000 - 1 - B \\ \bar{B} &= 9999 - B \\ \bar{B} &= 9999 - 0479 \\ \bar{B} &= 9250 \end{aligned}$$

2. Se realiza la suma de $A + \overline{B}$.

$$\begin{array}{rcl} A & = & 1732 \\ & + & \\ \bar{B} & = & 9520 \\ A+\bar{B} & = & 11252 \\ & + & 1 \\ & & 11253 \end{array}$$

3. La operación de sumar 1 se debe a que apareció un 1 en la posición r^m, indicando que $A > B$.

4. Se desecha el dígito 1 de la posición r^m dando como resultado:

$$A - B = 1253$$

Ejemplo de resta utilizando complemento, en base 16:

$$A - B = 2DA416_{16} - AF57C16_{16}$$

1. Se halla el complemento restringido de B

$$\bar{B} = r^m - 1 - B \; ; m = 5 \; ; r = 16$$
$$\bar{B} = \mathrm{FFFFF} - B$$
$$\bar{B} = \mathrm{FFFFF} - \mathrm{AF57C}$$
$$\bar{B} = \mathrm{50A83}$$

2. Se realiza la suma $A + \bar{B}$

$$\begin{array}{rcl} A & = & 02DA4 \\ & + & \\ \bar{B} & = & 50A83 \\ \hline A + \bar{B} & = & 53827 \end{array}$$

3. Como no apareció un dígito 1 en la posición r^m resulta $A < B$.
4. Se debe complementar el resultado de $A + \bar{B} = \bar{d}$; d.

$$\begin{array}{rcl} \bar{d} & = & FFFFF \\ & - & 53827 \\ \hline d & = & -AC7D8 \\ & & \\ A - B & = & -AC7D8 \end{array}$$

Este método permite aprovechar un mismo circuito sumador para sumar o restar. Puesto que se resta sumando, se evita el "pedir prestado".

REPRESENTACIÓN DE DATOS

Según ya vimos, las limitaciones de la electrónica convencional imponen una sola forma práctica de representación: la lógica de dos estados: "0" y "1".

Un circuito electrónico digital conoce habitualmente dos estados: *apagado/prendido*, *conectado/desconectado*, los cuales se corresponden con las representaciones lógicas "0" y "1".

La "representación binaria de datos" significa representar datos por medio de estos ceros y unos, denominados "dígitos binarios" o "bits" (contracción de binary digits).

Cualquier dato puede ser representado mediante una lógica binaria. Por ejemplo, el Código Morse utilizado en telegrafía, consiste en puntos y rayas; solo existe una posible elección binaria. Combinando estos puntos y rayas se pueden representar, después, números, caracteres, etcétera.

El bit es la unidad mínima de información. El problema a resolver es cómo los bits pueden representar los datos de una manera útil y cómo recuperar, posteriormente, esos datos.

Unidades de medida

8 bits = 1 byte
1024 bytes = 1 kilobyte
1024 kilobytes = 1 megabyte
1024 megabytes = 1 gigabyte
1024 gigabytes = 1 terabyte
1024 terabytes = 1 petabyte
1024 petabytes = 1 exabyte
1024 exabytes = 1 zettabyte
1024 zettabytes = 1 yottabyte

El método que se emplea en computación es el siguiente: dividir la memoria de la máquina en celdas de longitud fija y predeterminada de bits. Utilizando combinaciones de bits se pueden representar los símbolos deseados. A cada grupo de bits se le asigna una posición dentro de la memoria; es decir, una dirección. Esto permitirá encontrar dicho grupo cada vez que se necesite procesarlo.

Si vamos a representar datos de forma normal y que resulte comprensible, necesitaremos grupos de bits que identifiquen al menos lo siguiente:

- Los números del "0" al "9" (10 grupos de bits).
- Las letras del alfabeto (26 grupos de bits).
- Otros símbolos: {$, #, &, *, @, %, ^, <, = }.

Harán falta por lo menos 36 combinaciones de bits para representar las letras y números:

- 1 bit da dos combinaciones: $\{0, 1\}$, 2^1 combinaciones.
- 2 bits dan cuatro combinaciones: $\{00, 01, 10, 11\}$, 2^2 combinaciones.
- 3 bits dan ocho combinaciones: $\{000, 001, 010, 011, 100, 101, 110, 111\}$, 2^3 combinaciones.

De esta manera, utilizando grupos de 6 bits se obtienen $2^6 = 64$ combinaciones posibles de bits que nos permiten representar 10 números, 26 letras, 28 símbolos. Así, durante cierto tiempo se utilizaron códigos de 6 bits, a ese conjunto se le llamó carácter.

Todo esto significa que cuando los datos externos son trasmitidos a la computadora deben ser, primeramente, convertidos a una "representación interna" para que el computador pueda procesarlos. Después de que este proceso es llevado a cabo, los datos deben ser convertidos nuevamente a una "representación externa" para que el usuario reciba datos provistos de significado. Ver figura 4.1.

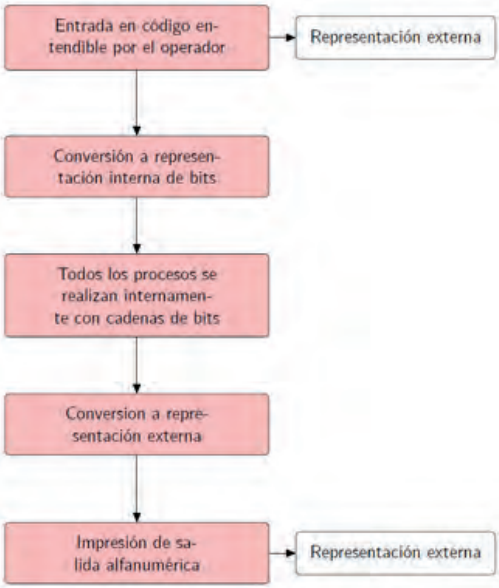


Figura 4.1: Procesamiento electrónico de datos.

Todo este proceso implica una codificación y posterior decodificación de la información.

Codificación

Codificar es aplicar un conjunto de reglas inequívocas que especifiquen la forma en que pueden representarse los datos, de forma tal que sea posible la decodificación ulterior.

Al margen de esta definición podemos decir que el propósito que tendría una codificación es posibilitar un manejo de la información que, de otro modo, no sería posible. Hoy, el hombre debe valerse de la codificación para decirle a una máquina qué es lo que quiere que haga, cómo, y con qué datos deberá manejarse. Como vimos, para ello utiliza el sistema binario, combinándolo y reformándolo con el objetivo de crear en base a él, distintas codificaciones que le permitan representar distintas cosas, sean números, letras, instrucciones, o estados internos de la máquina (señales de habilitación de un canal, de un periférico, etcétera).

La elección de un sistema de codificación adecuado es importante, ya que de esto dependerá la fiabilidad, universalidad y velocidad del sistema.

La velocidad de reconocimiento de la información varía según la potencia operativa del código. Por ejemplo, para transmitir una información necesitaremos una cantidad determinada de caracteres codificados, pero si ese sistema es menos potente, necesitaremos más caracteres con el consiguiente aumento en el tiempo de transferencia de los datos y de la posibilidad de que se pierda una porción de la información.

Existen en la actualidad una infinidad de códigos, algunos que son propios de una máquina o marca específica, y otros que son universalmente aceptados. Algunos ejemplos de estos códigos internacionales son los siguientes: BCD, EBCDIC, ASCII 8, etcétera.

La codificación consiste, entonces, en establecer una ley de correspondencia llamada código, entre las informaciones por representar y las posibles configuraciones binarias, de tal manera que a cada información le corresponda una y solo una configuración binaria.

Códigos

Representación de datos alfanuméricos

En el almacenamiento de un computador se pueden reconocer dos clases de datos: en forma de caracteres (es decir, alfanuméricos: nombres, direcciones, etcétera) y numéricos.

Dijimos que con grupos de 6 bits se podían representar $2^6 = 64$ combinaciones de bits, es decir, 64 símbolos diferentes. Tales códigos de 6 bits existieron, pero la mayoría de las computadoras de 3ra y 4ta generación trabajan con códigos de 8 bits. Con estos es posible representar los 36 caracteres alfanuméricos y símbolos más usados como $, =, >, <, entre otros.

A cada grupo de 8 bits se lo denomina byte (del inglés *binary terms*), 1 byte = 8 bits. Los caracteres, entonces, se traducen a un código de 8 bits.

En informática hay dos códigos de uso general para representación de caracteres:

1. EBCDIC: *Extended Binary Coded Decimal Interchange Code,* que usan principalmente algunos modelos de IBM.
2. ASCII: *American Standard Code for Information Interchange*, de uso universal en microprocesadores.

Esquema de representación de un byte en ambos códigos: la representación de cada caracter se divide en dos partes, una de zona a la izquierda, y una, numérica, a la derecha.

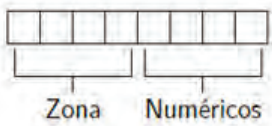


Figura 4.2: Zonas de un número de 8 bits.

1. ASCII

Este código utiliza 7 bits que proporciona 2^7 = 128 combinaciones diferentes. El octavo bit o bit más significativo, es decir, el de la izquierda, es el bit de paridad; sirve para garantizar la conservación del contenido de un byte.

- Cada dígito (0 al 9) tiene 011 como la parte de zona del byte y su representación binaria como la parte numérica.
- Los caracteres alfabéticos (mayúsculas) están distribuidos en dos grupos según que sus bytes tengan 100 o 101 como la parte de zona.
  Se adjunta la tabla de códigos ASCII de 7 bits; se utiliza "tal cual", es decir, sin paridad, añadiendo un cero en la posición izquierda, o con paridad, añadiendo en esa posición el bit adecuado.
  Existe también el código ASCII - 8; este código utiliza los 8 bits para almacenar un carácter y el computador almacena un bit extra o noveno bit llamado bit de paridad.
- Cada dígito (0 al 9) tiene 0101 como la parte de zona del byte y su representación binaria como la parte numérica.
- Los caracteres alfabéticos (mayúsculas) están distribuidos en dos grupos según que sus bytes tengan 1010 o 1011 como la parte de zona.

2. EBCDIC

El código ampliado decimal codificado en binario hace uso de 8 bits, presente principalmente en equipos IBM y compatibles.

Se forma:

- Cada *dígito* tiene *1111* como la parte de zona del byte y su representación binaria, como la parte numérica.
- Los *caracteres alfabéticos* (mayúsculas) están distribuidos en tres grupos, ya sea que los bytes tengan 1100, 1101 o 1110 como bits de zona.

Ejemplo de codificación EBCDIC.

CARÁCTER	ZONA	NUMÉRICA
A	1100	0001
B	1100	0010
1	1111	0001
2	1111	0010

La codificación se comprende observando el siguiente diagrama

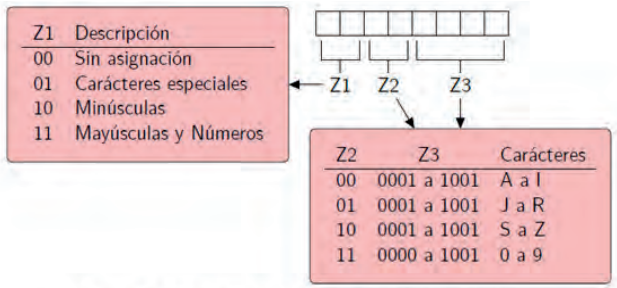


Figura 4.3: Partes de un número en formato EBCDIC.

Representación de datos numéricos

Representar números no es una operación sencilla y es preciso diferenciar varios casos. Los dispositivos de las computadoras en los que se efectúan las operaciones aritméticas solo podrán procesar números que les sean presentados según uno o más formatos bien definidos.

Existen dos métodos fundamentales para representar un dato numérico: codificar los dígitos decimales individualmente o codificar el número completo. En el último método tendremos, a su vez, dos tipos de formatos: los enteros de punto fijo y los números de punto flotante.

Codificación del número completo

1. Representación en formato de punto fijo

El primer componente de un formato es su longitud; el segundo, es el convenio escogido en la representación o codificación del número.

La representación en formato fijo es la manera más natural de escribir un número en una palabra de memoria.

Para representar enteros grandes, como veremos, necesitamos varios bytes, pero para hacer operaciones aritméticas con eficacia hay que emplear un número de bytes fijo, no variable; por tanto, la determinación del número de bytes supone la del mayor número representable.

Recordemos, además, que en electrónica no es posible representar directamente los signos "+", "-", sino que se deben utilizar ciertas convenciones. En todas ellas para un formato de trabajo de *n* bits (en general, 8, 16, 32, 64) se tiene que:

- El bit reservado para el signo es "0" si el número es positivo.
- El bit reservado para el signo es "1" si el número es negativo.

Forma binaria directa: para representar enteros puede usarse a la forma binaria directa, que es la representación decimal del número en el sistema binario. Con 8 bits pueden representarse directamente los números comprendidos entre 00000000 y 11111111, es decir, entre 0 y 255 de la base decimal. Se plantean dos dificultades que se detallan a continuación: la primera, que solo representamos números positivos; la segunda, que la magnitud de esos números queda limitada a 255, si trabajamos con solo 8 bits. A continuación, exploraremos las soluciones más comunes a estos inconvenientes.

Binario con signo: en un número representado en notación binaria con signo, según dijimos, este viene indicado por el bit de la izquierda llamado Bit Más Significativo (*BMS*), tradicionalmente "0" si es positivo, y "1" si es negativo; por lo tanto, el rango que se puede representar con 8 bits va desde:

$$\underbrace{1}_{-}\ \underbrace{1111111}_{127} = -127 \qquad \underbrace{0}_{+}\ 1111111 = +127$$

Figura 4.4: Bit más significativo

De esta forma ya se pueden representar números negativos, pero a costa de reducir la magnitud de 255 a 127. Además, se pierde un número ya que tenemos +0 y -0. Considerando el problema de la magnitud, para representar

números más grandes no hay más remedio que utilizar un número mayor de bits; es decir, usar una "palabra de memoria" mayor.

Se considera "palabra de memoria" a la longitud de la cadena de bits direccionada como un solo bloque en la memoria de la computadora (8, 16, 32, 64 bits). Utilizando, por ejemplo, palabras de 16 bits podemos representar todos los números comprendidos entre -32 Ki y +32 Ki. (1 kibyte = 1024 bytes; 2^{15} bytes = 32 kibytes). El bit 15 lleva el signo y los 15 restantes (los comprendidos entre el 0 y el 14) expresan hasta la magnitud 2^{15} = 32767. Representación binaria:

Ejemplo: Representación binaria:

$$(+17)_{(10)} = \underbrace{0}_{signo\ +}\ \underbrace{00\ 0000\ 0001\ 0001}_{17\ en\ binario}$$

$$(-17)_{(10)} = \underbrace{1}_{signo\ -}\ \underbrace{00\ 0000\ 0001\ 0001}_{17\ en\ binario}$$

$$(+127)_{(10)} = \underbrace{0}_{signo\ +}\ 00\ 0000\ 0111\ 1111$$

$$(-129)_{(10)} = \underbrace{1}_{signo\ -}\ 00\ 0000\ 1000\ 0001$$

$$(+256)_{(10)} = \underbrace{0}_{signo\ +}\ 00\ 0001\ 0000\ 0000$$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
±	Valor absoluto														

Figura 4.5: Partes de un número binario con signo de 16 bits.

Esta representación impone un tratamiento especial del signo y, por el mismo hecho, circuitos diferenciales para el procesamiento de sumas y restas, inconveniente que desaparece si se representan los números bajo la forma complementada.

Complemento a uno: la representación en binario se puede obtener de la siguiente manera:

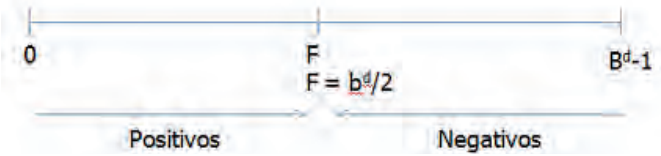


Figura 4.6

Dado un número entero, hallar el natural (la representación):

$$n = e + \delta(e) * b^d \qquad \begin{cases} \delta(e) = 0 \text{ si } e >= 0 \\ \delta(e) = 1 \text{ si } e < 0 \end{cases}$$

Figura 4.7

Si queremos saber cuál es la representación (en 8 bits) del número 35: e = 35 y la función devuelve 0 porque es entero, es mayor que cero.

b_d = 256

Por lo cual, n = 35 y en binario será la presentación.

$$(+35) = 0000\ 0000\ 0010\ 0011$$

Dado un natural, hallar el entero representado. Dado un número entero, hallar el natural (la representación).

$$e = n - \varphi(n) * b^d \qquad \begin{cases} \varphi(n) = 0 \text{ si } n < f \\ \varphi(n) = 1 \text{ si } n >= f \end{cases}$$

Figura 4.8

Si queremos saber cuál es el número entero representado (en 8 bits) en 0110 0010:

n (el decimal del número de la representación) = 98 y la función devuelve 0 porque la frontera es 127

b_d = 98

Por lo cual, e = 35 y en binario será la presentación.

$$(+35) = 0000\ 0000\ 0010\ 0011$$

Una manera más sencilla establece que en complemento a uno, todos los enteros positivos, se representan en su formato binario correcto. Para representar un número negativo se complementan cada uno de los bits de la representación original; es decir, se transforman todos los ceros en uno y los unos en cero.

Ejemplo:

$$(+3)_{(10)} = 0000\ 0000\ 0000\ 0011$$
$$(-3)_{(10)} = 1111\ 1111\ 1111\ 1100$$
$$(+127)_{(10)} = 0000\ 0000\ 0111\ 1111$$
$$(-127)_{(10)} = 1111\ 1111\ 1000\ 0000$$

Complemento a dos: los números positivos se representan como binarios con signo exactamente igual que se hacía en complemento a uno. La diferencia estriba en la representación de los números negativos, que se hace determinando, primero, el complemento a uno y sumando uno a continuación; es decir, obteniendo el complemento auténtico.

Veamos un ejemplo:

$$(+10)_{(10)} = 0000\ 0000\ 0000\ 1010$$
$$(-10)_{(10)} = 1111\ 1111\ 1111\ 0110$$
$$(+125)_{(10)} = 0000\ 0000\ 0111\ 1101$$
$$(-125)_{(10)} = 1111\ 1111\ 1000\ 0111$$

El complemento a dos constituye la forma de representación más adecuada para los procesadores más simples, como los microprocesadores. En

procesadores complejos puede recurrirse a otras formas de representación, como el complemento a uno, usando un circuito especial para corregir el resultado.

Cero desplazado: es otra técnica de punto flotante más sencilla.

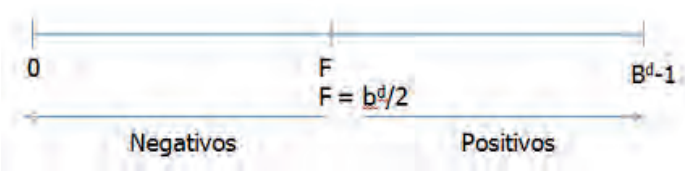


Figura 4.9

Para obtener el entero o el natural las fórmulas son las siguientes:

N = E + F, donde F es la frontera o el punto medio y se obtiene como $b^d/2$

E = N - F

Si queremos saber cuál es el número entero representado (en 8 bits) en 0110 0010:

N (el decimal del número de la representación) = 98

E = N - F, o sea, E = 98 - 128, entonces, el entero es E = - 30

Ejemplo: Si queremos saber cuál es la representación del número 72

N = E + F, o sea, N = 72 + 128, entonces, la representación del binario de 200 es:

N = 1100 1000

En la aritmética de punto fijo se le deja al programador el cuidado de situar la coma y se ve obligado a conocer y hacer evolucionar el lugar de la coma a lo largo de las operaciones. Por lo tanto, se usa la aritmética entera (punto fijo) cuando las operaciones se realizan con números que están restringidos a tomar valores enteros solamente. Cuando las operaciones se realizan con números que toman valores decimales, la aritmética usada es la de punto flotante.

2. Representación de punto flotante.

Los números fraccionarios decimales se pueden representar en almacenamiento, mediante su expresión en punto flotante; es decir, mediante sus representaciones en forma exponencial normalizada.

A continuación explicaremos cómo se realiza esto.

La condición básica es que los decimales deben representarse en algún formato fijo. Para no desperdiciar bits, la representación utilizada pasará por la normalización de todos los números.

Supongamos el número "0,000123"; al escribirlo, se desperdician 3 lugares correspondientes a los ceros a la izquierda del número, ceros que solo sirven para indicar la posición del punto. El número se normaliza haciendo: $0,123x10^3$.

Es decir, el número se coloca en una escala tal que el punto decimal queda inmediatamente a la izquierda del dígito más significativo distinto de cero.

Ejemplo: Otros Casos:

Número	Notación exponencial normalizada
13,76	$0,1376 \cdot 10^2$
978,0	$0,978 \cdot 10^3$
0,00742	$0,742 \cdot 10^2$

En el número normalizado: $0,123x10^3$, ".123" se llama **mantisa normalizada**; y -3 se denomina exponente. La normalización ha consistido en la eliminación de todos los ceros situados a la izquierda y en el cálculo del exponente. La forma general es:

$$SM \times \&^E$$

Donde:

- Indica el signo del número.
- Corresponde a la mantisa del número normalizado.
- Es el exponente del número normalizado.
- Es la base del sistema de numeración.

La notación que se usa en aritmética de punto flotante para representar internamente los números en el computador es, básicamente, una adaptación

de la notación científica. Por lo tanto, un número de punto flotante se compone de tres partes:

- El signo del número.
- Los dígitos del número, llamados *mantisa*.
- El *exponente* expresado como una característica de dos dígitos.

La figura muestra cómo están distribuidos los bits en el almacenamiento de un número binario de punto flotante, en forma de una palabra de 32 bits.

Figura 4.10: Partes de la representación de punto flotante para 32 bits

Como un exponente puede ser negativo, tenemos que incluir el signo del exponente en el campo reservado para él. Mejor que reservar un bit para el signo, empleamos la característica del exponente en lugar de su verdadero valor. Cuando se reserva un campo de 7 bits para el exponente, la característica de un exponente se obtiene agregando $2^6 = 64$ al exponente. Entonces *C=E+64. Esta forma de representar enteros se denomina de "Cero desplazado".* Exponente expresado con la técnica de cero desplazado.

Exponente verdadero	-64	-63	-62	···	-1	0	+1	···	+63
Característica	0	1	2	···	63	64	65	···	127

Cuadro 4.2: Exponente expresado con la técnica de cero desplazado.

Existen varios sistemas de representación de punto flotante. Es importante aclarar que las actuales técnicas utilizan palabras de 64 bits, pero a los fines prácticos utilizaremos palabras de 32 bits, veremos dos técnicas.

IBM-360

Una de las primeras técnicas de codificación de punto flotante es IBM-360.

Esta técnica divide la palabra de 32 bits de la siguiente manera:

Figura 4.11: Partes de la representación IBM-360

Para codificar el exponente se utiliza una técnica de punto fijo *cero desplazado* y para la mantisa una técnica de punto fijo de *Signo Valor Absoluto.*

Veamos un ejemplo del número a la representación:

Dado el siguiente número 123, 456 x 10^2, cómo obtenemos la representación.

1- Resolvemos la forma científica para poder convertir a Hexa: 12345,6

2- Convertimos el número a Hexa con el método ya visto: 3039,9999

3- Normalizamos el número: 0,303999 x 10^4 (como la mantisa utiliza 6 dígitos Hexa, solo tomo los primeros 6).

Tengo el exponente, faltan calcular los dos primeros dígitos hexa de la representación:

N= e + f, entonces, n = 4 + 64, por lo cual, n = 68 que en binario en 7 bits es 100 0100.

El signo adelante es 0 por ser positivo, entonces, los 8 bits son 0100 0100, que en hexa es 44.

Por lo que la representación final es $(44303999)_{16}$

Veamos otro ejemplo de la representación al número:

$(C3AB7261)_{16}$

Tomamos los dos primeros dígitos hexa que tienen el signo y el exponente.

C3 en binario es 1100 0011, por lo cual, en signo es NEGATIVO.

El exponente será 100 0011 en decimal 67, o sea, E = N - F, por lo cual, E = -61.

Ahora hay que calcular el número de la siguiente manera:

$0,AB7261x(10)_{16}{}^{-61}$,$(10x16^{-1} + 12x16^{-2} + 7x16^{-3} + 2x16^{-4} + 6x16^{-5} + 1x16^{-6})x16^{-61}$

Resolvemos la expresión y obtenemos el resultado que, recordemos, es Negativo.

PDP-11

Una técnica mucho más moderna y más utilizada es PDP-11.

Esta técnica divide la palabra de 32 bits de la siguiente manera:

Figura 4.12: Partes de la representación de PDP-11

Esta técnica usa un bit más para el exponente, lo cual permite un rango aún más grande de representación, esto lo logra porque a diferencia de IBM-360 que utiliza una base hexa para la representación, PDP-11 utiliza la base 2.

(A modo de ser más práctico, los ejemplos y ejercicios de PDP-11 se expresan en base 16, pero su fundamento es la base 2, que es fácil de convertir por ser potencias enteras una de la otra).

Entonces, al utilizar el sistema binario, todo número binaro normalizado siempre, después de la coma, lleva un 1 (uno).

Veamos un ejemplo del número a la representación:

Dado el siguiente número 48, 654562 x 10^4, cómo obtenemos la representación.

1- Resolvemos la forma científica para poder convertir a hexa: 486545,62

2- Convertimos el número a binario con el método ya visto: 1110110110010010001,10011. Debemos tratar de ver que, luego de normalizar, tengamos 24 dígitos binarios detrás de la coma.

3- Normalizamos el número: 0,11101101100100100011001 1 x $(10)_2^{19}$. RECORDAR que el primer 1 después de la coma NO va.

Tengo el exponente, falta calcular los dos primeros dígitos hexa de la representación:

n= e + f, entonces, n = 19 + 128, por lo cual, n = 147, que en binario en 8 bits es 1001 0011.

El signo adelante es 0 por ser positivo, entonces, la cadena de bits es:

sigo 0, exponente 1001 0011, y la mantisa sin el primer 1 es 11011011001001000110011.

Todo junto, pasado a hexa, queda $(49ED9233)_{16}$

Veamos otro ejemplo de la representación al número:

$(124EC82A)_{16}$

En este caso tomamos los tres primeros dígitos hexa que tienen el signo y el exponente.

124 en binario es 0001 0001 0100, por lo cual, el signo es positivo.

El exponente será tomado después del signo; es decir, los siguientes 8 bits 001 0001 0 en decimal 34, o sea, E = N - F, por lo cual, E = -94.

Ahora hay que calcular el número de la siguiente manera: el tercer bit hexa era el 4 en binario 0100, el primer dígito (0) forma parte del exponente, por lo que nos falta uno, ¿cuál? es el 1 implícito de la representación, por lo que queda 1100 y es E

$0,EEC82A \text{ x } (10)_2^{-94}$,$(14x16^{-1} + 14x16^{-2} + 12x16^{-3} + 8x16^{-4} + 2x16^{-5} + 10x16^{-6}) x2^{-98}$ Resolveremos la expresión y obtendremos el resultado que, recordemos, es un número negativo.

Ahora bien, en los ejemplos que hemos visto (cuando pasamos del número a la representación) observamos que los números a representar tienen en su forma científica un exponente chico; es decir, elevado a la -3, -2, -1 o 1, 2, 3.

Pero qué pasa cuando queremos, por ejemplo, representar valores tales como la masa del electrón 9 x 10^{-28} gramos o la del sol 2 x 10^{33} gramos, lo que supone una gama (rango) de números mayor a 10^{60}.

En estos casos no podemos resolver la forma científica como hemos visto, pues la cantidad de ceros sería enorme.

Nosotros partimos de la siguientes fórmula:

M x $b^e = B^x$, donde:

M es la mantisa del número a convertir,

b es la base del número,

e es el exponente,

B es la base a la que se quiera pasar,

x es la incógnita, que será el exponente a la que se elevará la B.

Para lo cual, esa etapa la resolvemos usando el logaritmo natural.

x es nuestra incógnita $x = (ln(M)+e \text{ x } ln(b))/ln(B)$

Luego, se debe operar para hallar una mantisa que quedará multiplicada por la base B y un exponente entero. Habremos obtenido el exponente de nuestra representación. Solo quedará normalizar (de ser necesario y con la técnica que se utilice) y seguir los pasos ya vistos.

Veamos un ejemplo.

Si tenemos que representar $123,45x10^{30}$

Por fórmula $x = (ln(M)+e \text{ x } ln(b))/ln(B)$, o sea,

$x = (ln123,45 + 30 \text{ x } ln10)/ln16$ (Base 2 si trabajo con PDP-11 o 16 en IBM-360)

$x = 29,73029692$. Volviendo a nuestra fórmula original, en su parte derecha $16^{29,73029692}$

Por propiedad de potencia puedo escribir como $16^{29} \text{ x } 16^{0,73029692}$;

si resolvemos la parte de exponente fraccionario, obtenemos $7,5746943 \text{ x } 16^{29}$;

y continuamos con los pasos descriptos del procedimiento natural; es decir, cambio de base a base 16, normalizo, etcétera.

Codificación de caracteres numéricos

Cada dígito del número es codificado individualmente por el código de caracteres del computador, por ejemplo, EBCDIC o ASCII-8, excepto que la zona del último dígito se reserva para el signo del número.

Los códigos de caracteres numéricos no se emplean para realizar operaciones aritméticas con los datos.

Estas cadenas de caracteres son generalmente de longitudes variables.

De acuerdo con esto, la mayoría de los computadores convierten al código de procesamiento más conveniente antes de llevar a cabo las operaciones aritméticas con los datos. Los resultados del procesamiento aritmético se vuelven a convertir, entonces, en el código de caracteres para la salida potencial.

La codificación del signo se refleja en la siguiente tabla y ejemplo:

Zona	Signo
1111	sin signo
1100	positivo
1101	negativo

Cuadro 4.3: Significado del 1/2 Byte de zona

Número	6	3	±7
+637	1111 0110	1111 0011	1100 0111
-637	1111 0110	1111 0011	1101 0111

Cuadro 4.4: Códificación en EBCDIC de los números +637 y −637.

Un código de procesamiento muy conocido es el sistema DECIMAL CODIFICADO en BINARIO o código BCD (*binary coded decimal*).

Representación BCD: esta técnica consiste en codificar cada una de las cifras decimales de un número mediante su representación binaria utilizando 4 bits, reservando para el signo el último campo de 4 bits.

Para codificar las diez cifras decimales comprendidas entre el 0 y el 9 hacen falta 4 bits (que dan dieciséis combinaciones).

Símbolo	0	1	2	3	4	5	6	7	8	9
Código	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Cuadro 4.5: Representación binaria de 4 bits de los dígitos decimales.

Como solo hacen falta 4 bits para codificar una cifra en BCD, pueden representarse dos cifras en cada byte y se forma el BCD empaquetado. Se utilizan los bits de zona reservados para el signo como últimos 4 bits (1100 para los positivos y 1101 para los negativos).

Retomando el ejemplo de ±637.

Número	6	3	7	±
+637	0110	0011	0111	1100
-637	0110	0011	0111	1101

Obsérvese que es posible convertir datos numéricos del código EBCDIC (o ASCII-8) de 8 bits en el BCD, mediante la eliminación de los unos (1) del medio byte izquierdo de cada dígito (bits de zona) y "empaquetando" o comprimiendo los bytes restantes.

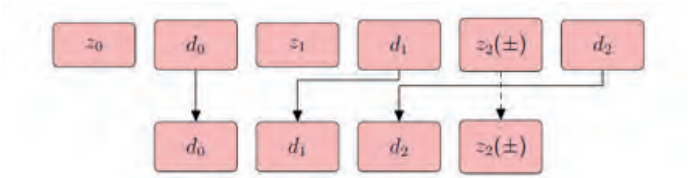


Figura 4.13: Empaquetamiento.

En general, en formato EBCDIC, en *n* bytes se puede representar un número de *n* dígitos decimales, mientras que, en el formato decimal empaquetado, en *n* bytes pueden representarse cifras de $2 \times n - 1$ dígitos decimales.

Control de paridad

Cuando la computadora trasmite información de un lado a otro lo hace a altísima velocidad. Esto trae aparejado el riesgo de perder algún bit en el camino, esto ocurre cuando el receptor de la información recibe caracteres equivocados.

Para tratar de disminuir este riesgo se utiliza el control de paridad que funciona de la siguiente manera:

Se agrega un bit a la cantidad de bits que tenga el código utilizado. Por ejemplo, si trabajamos con el código EBCDIC vamos a tener el byte, conformado por 8 bits, más este bit adicional denominado de redundancia. El nombre se debe a que se produce una redundancia en la codificación, esto es, se agregan más bits de los necesarios para representar la información.

La función del bit adicional es detectar un error de transmisión, lo que se logra haciendo que la suma de todos los "1" de un byte sea par (paridad par) o impar (paridad impar).

De esta forma, si se opera con paridad par y en un byte a transmitir hay tres "1", se enciende el bit de paridad (toma valor "1"), entonces, la suma es 4 (par).

Al contrario, si la suma de los bits del byte que se va a trasmitir es par se deja el bit de paridad con valor "0" (cero).

De esta forma, al transmitir información el receptor puede tener el control de paridad, y si detecta que algún byte tiene una cantidad incorrecta de "1", es porque hubo un error y solicita la retransmisión del mismo.

Este método detecta cuando hay un solo error en la transmisión de un byte, porque si son dos se compensan.

Si se debe transmitir el byte N = 1111 0101.

- Se debe notar que tiene 6 bits en "1".
- Es necesario definir la ubicación del bit de paridad, para este ejemplo, primero enviaremos el bit de paridad (p), y luego N.
- Si se utiliza paridad impar es necesario que p = 1.
- Si optamos por p*aridad par* usaremos p = 0.

Paridad	p	N	"1" transmitidos
Impar	1	11110101	7
Par	0	11110101	6

Cuadro 4.6: Paridad en números binarios.

Formatos de representación de números de punto flotante

Los siguientes cuadros resumen el proceso de representar e interpretar números de punto flotante en IBM-360 y PDP-11, con las siguientes consideraciones.

- Las variables σ, ε y μ corresponden al número natural representado por la cadena de bits indicada.

- Las variables S, E, M corresponden a los campos *Signo*, *Exponente* y *Mantisa* del número a ser representado (indicándose la base como subíndice).

- Las variables s, e, m corresponden a los campos *Signo*, *Exponente* y *Mantisa* del número interpretado a partir de la cadena de bits en cada formato.

- Al momento de calcular la variable *m* se realiza la división por 16^6 o 2^{24}. En ambos casos equivale a mover la coma antes del bit 24.

- En el caso particular del proceso mostrado en la figura [fig:InterpretacionMantisa], el valor de la mantisa ($\mu_{(16)}$) en ambos formatos serán iguales sí en el formato IBM-360 $X_1 \geq 8_{(16)}$ (es decir, que el bit 23 es 1).

- Se debe tener en cuenta que la misma cadena de bits representa números diferentes en cada formato.

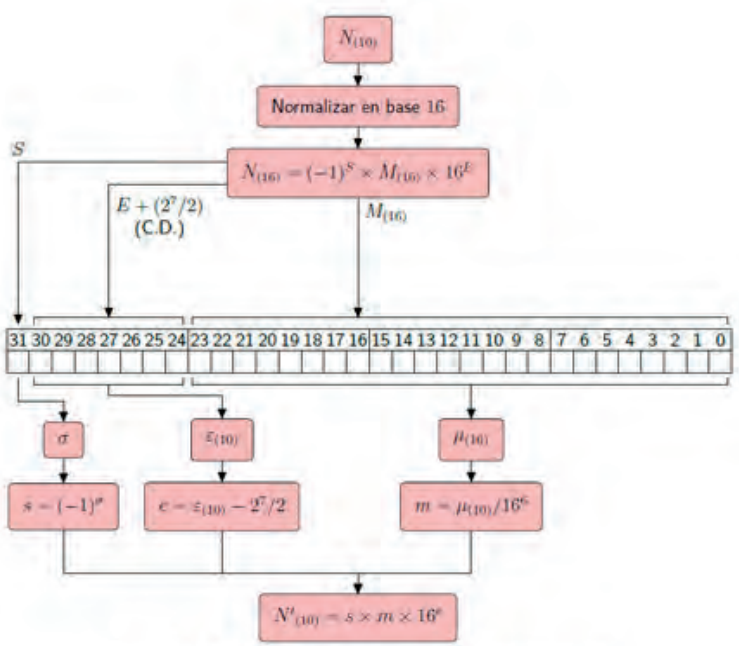


Figura 4.14: IBM-360. Representación e Interpretación.

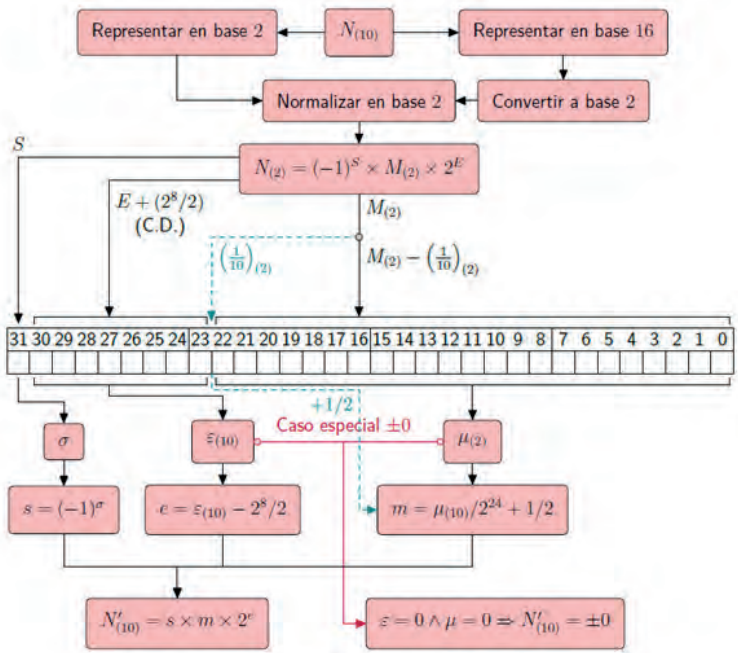


Figura 4.15: PDP-11. Representación e Interpretación.

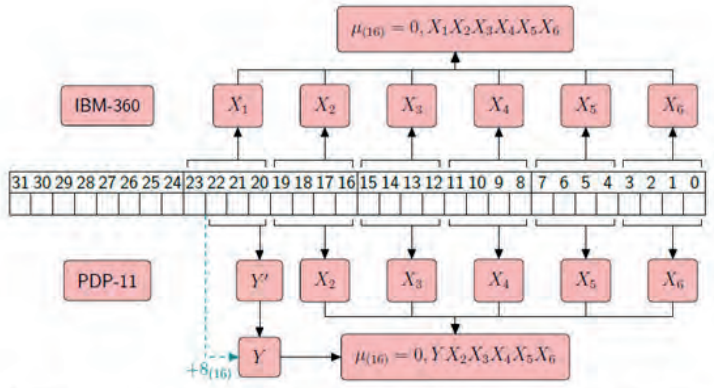


Figura 4.16: IBM-360 y PDP-11. Interpretar la mantisa usando base hexadecimal.

Procedimiento para representar en punto flotante

Usaremos el número $N_{(10)} = 153,48487 \times 10^2$ para dar un procedimiento para representar N en ambos formatos mencionados (IBM-360 y PDP-11).

$$
\begin{array}{lll}
N_{(10)} = & (153,48487 \times 10^2)_{10} & \text{(a)} \\
N_{(10)} = & +1 \times (15348,487 \times 10^0)_{10} & \text{(b)} \\
N_{(10)} = & +1 \times (15348 + 0,487)_{10} & \text{(c)} \\
N_{(16)} = & +1 \times (\texttt{3BF4} + 0,\texttt{7CAC083}\ldots)_{16} & \text{(d)} \\
N_{(16)} = & +1 \times (\texttt{3BF4},\texttt{7CA})_{16} & \text{(e)} \\
\hline
N_{(16)} = & -1^0 \times 0,\texttt{3BF47CA} \times 10^4 & \text{(f)} \\
N_{(2)} = & -1^0 \times 0,\texttt{0011|1011|1111|0100|0111|1100|1010} \times 10^{16} & \text{(g)}
\end{array}
$$

Es importante resaltar que el número N es siempre el mismo, solo es presentado en un formato o base diferente.

1. El número original es positivo y de exponente pequeño.
2. Desarrollamos la forma científica e identificamos el signo del número.
3. Se debe dividir el número en parte entera y fraccionaria, para su conversión.
4. Representar cada parte del número en base 16.
5. Obtener el número completo en base 16.
6. Normalizar la mantisa en base 16. Usar $(-1)^{signo}$ para el signo.
7. Representar en base 2 a partir de la representación en base 16. El signo es el mismo, la mantisa se transforma por el método de los agrupamientos, en el caso del exponente se debe tener en cuenta que $16 = 2^4$, por lo que $16^n = 2^{4 \boxtimes n}$.

Representación en el formato de IBM-360

Una vez que hemos normalizado N en base 16, podemos construir la cadena de bits de su representación (o la cadena de símbolos hexadecimales).

1. Primero debemos representar el exponente en el formato de cero desplazado de 7 bits (previa verificación de que sea representable). Aplicando la ecuación:

$$
\begin{aligned}
natural = entero + frontera \Rightarrow n = e + f &= e + 2^7/2 \\
n &= 4 + 64 = 68 \\
n_{en\ 7\ bits} &= 1000100
\end{aligned}
$$

2. Completamos el primer byte con el bit de signo.

$$\text{Primer Byte a izquierda} = (\underbrace{0}_{signo}\ \underbrace{1000100}_{exponente})_{(2)} = 44_{(16)}$$

3. Completamos con los seis dígitos hexadecimales más significativos de la mantisa. Completando con ceros a la derecha de ser necesario.

$$\text{N en IBM-360, en base } 16 = 44|3B|F4|7C$$
$$\text{N en IBM-360, en base } 2 = 01000100|00111011|11110100|01111100$$

Representación en el formato PDP-11

De manera similar podemos partir de la expresión (g) para representar el número en formato PDP-11.

1. Es necesario asegurar que el número de la expresión (g) este normalizado en base 2.

$$N = -1 \times 0,0011|1011|1111|0100|0111|1100|1010 \times 10^{16}$$
$$N = -1 \times 0,11101111101000111111001010 \times 10^{16-2}$$
$$N = -1 \times 0,1110|1111|1101|0001|1111|0010|1000 \times 10^{14}$$

2. Una vez normalizado en base 2, debemos representar el exponente en el formato de cero desplazado de 8 bits. Aplicando la ecuación:

$$natural = entero + frontera \Rightarrow n = e + f = e + 2^8/2$$
$$n = 14 + 128 = 142$$
$$n_{en\ 8\ bits} = 10001110$$

3. Completamos los primeros 9 bits a izquierda.

$$\text{Primeros 9 bits a izquierda} = (\underbrace{0}_{signo}\ \underbrace{1001110|0}_{exponente})_{(2)}$$

4. En el caso del ejemplo no es necesario, pero debemos completar la mantisa con ceros a la derecha hasta completar los 24 bits. El formato PDP-11 no codifica el primer "1" significativo pues está implícito en la técnica. Por lo que los bits en el campo de la mantisa son como los que se muestran a continuación:

$$1110|1111|1101|0001|1111|0010$$

5. Falta concatenar los 9 bits de la característica (el signo y el exponente) con los 23 bits de la mantisa.

$$\text{N en PDP-11, en base 2} = \overbrace{0}^{signo}\ \overbrace{100|111|0}^{exponente}\ \overbrace{110|1111|1101|0001|1111|0010}^{mantisa}$$
$$\text{N en PDP-11, en base 16} = 4|E|6|F|E|1|F|2$$

Usando el logaritmo natural

Los procedimientos anteriores no definen ningún procedimiento matemático en particular para la conversión de *N* a la base hexadecimal o binaria. Los métodos más simples pueden resultar demasiado extensos (y propensos a tener errores de cálculo), por ello, se aplica ln para generar una forma de *N* más útil.

El ln permite generar la representación $N = -1^S \times B^x$.

$$\begin{aligned} N &= -1^S \times M \times 10^E \\ \cancel{-1^S} \times B^x &= \cancel{-1^S} \times M \times 10^E && \text{Eliminando el signo} \\ \ln(B^x) &= \ln(M \times 10^E) && \text{Aplicando ln a ambos miembros} \\ x \times \ln(B) &= \ln(M) + E \times \ln(10) && \text{Aplicando propiedades de ln} \\ x &= \frac{\ln(M) + E \times \ln(10)}{\ln(B)} && \text{Despejando x} \end{aligned}$$

En el caso de $N = +1 \times 153,48487 \times 10^2$ y aplicando la ecuación para obtener x para la representación en la base 16 ($B = 16$).

$$
\begin{aligned}
x &= \frac{\ln(153,48487) + 2 \times \ln(10)}{\ln(16)} \\
x &= \frac{5,03360199539 + 2 \times 2,30258509299}{2,77258872224} \\
x &= \frac{5,03360199539 + 4,60517018599}{2,77258872224} \\
x &= \frac{5,03360199539 + 4,60517018599}{2,77258872224} \\
x &= 3,47645220658
\end{aligned}
$$

Por lo que se puede escribir $B^x = 16^{3,47645220658} = 16^3 \times 16^{0,47645220658}$.

$$
\begin{aligned}
N_{(10)} &= +1 \times 16^{0,47645220658} \times 16^3 \\
N_{(10)} &= +1 \times 3,74718920894 \times 16^3 \\
N_{(16)} &= +1 \times 3,BF47CAC0 \times (10_{(16)})^3 \\
N_{(16)} &= +1 \times 0,3BF47CAC0 \times (10_{(16)})^4
\end{aligned}
$$

En la última expresión hemos obtenido N normalizado en base 16, obsérvese que los valores obtenidos son iguales que los obtenidos con el método anterior (además de con más decimales).

COMPONENTES DE UN SISTEMA DE CÓMPUTOS

Un sistema de cómputos o sistema de procesamiento electrónico de datos se divide en cuatro (4) componentes básicos:

1. *Hardware*.
2. Sistema operativo.
3. Programación de sistemas.
4. Programas de aplicaciones.

Figura 5.1: Vista abstracta de un sistema de cómputos - Esquema de Peterson

Hardware

Es el nivel de máquina real, el inferior del sistema; es un nivel de lógica digital o biestable en el cual el sistema solo abre y cierra compuertas, existen solamente dos estados: CONECTADO - DESCONECTADO.

Sistema operativo

Es un conjunto de programas que administra los recursos del equipo y sirve de interfaz entre los usuarios y el sistema.

Programación de sistemas

Constituye un conjunto de programas que tienen funciones "cercanas" a las de un sistema operativo, pero no forman parte de él; en este grupo se incluyen los linkeadores, compiladores, editores, macroensambladores, etcétera.

Programas de aplicaciones

Este grupo de programas está integrado por diferentes tipos de aplicaciones y utilidades manejadas directamente por los usuarios.

Hardware

De acuerdo al modelo presentado, en primer lugar, trataremos el tema del *HARDWARE* y sus componentes, para lo cual planteamos la siguiente clasificación:

Figura 5.2

Figura 5.3

Figura 5.4

En este capítulo nos centraremos en los componentes internos, específicamente en la Unidad Central de Proceso de la computadora.

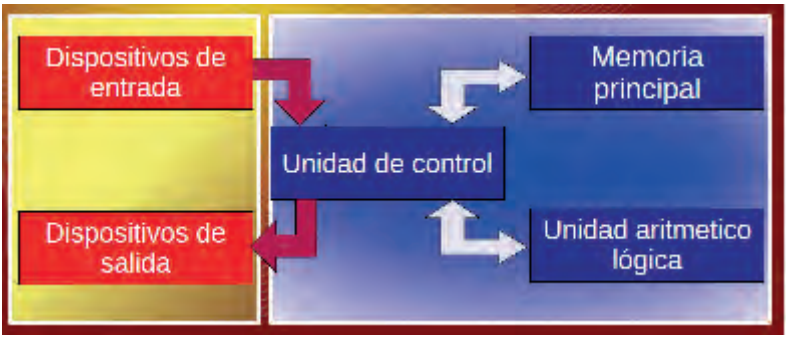


Figura 5.5: Esquema de los componentes del *hardware*.

Comenzaremos con el estudio del Procesador Central o de la CPU.

El procesador

Recordemos lo visto en los primeros módulos.

Una computadora es una máquina capaz de aceptar datos a través de un medio de entrada, procesarlos automáticamente bajo el control de un programa que antes fue almacenado en la memoria principal, y proporcionar la información resultante a través de un medio de salida.

Una computadora comprende, en consecuencia y según el esquema visto, tres partes básicas a saber:

- Una memoria central (almacena los programas y datos).
- Una unidad central de proceso (ejecuta los programas).
- Unidades de entrada y salida (E/S) para intercambios con el exterior.

Estos bloques están comunicados entre sí mediante conjuntos de líneas que transportan información binaria del mismo tipo. Dichos conjuntos se denominan buses.

La Unidad Central de Procesamiento es el componente central de una computadora digital. Su objetivo consiste en interpretar códigos de instrucción que se reciben de la memoria y realizar operaciones aritméticas, lógicas y de control con datos almacenados en registros internos, palabras de memoria o unidades de interfaz de E/S. En el exterior, la CPU tiene un sistema de buses

que transfiere instrucciones, datos e información de control a los módulos conectados a ella y desde estos últimos.

Una CPU típica suele estar dividida en dos partes: la unidad de control y la unidad aritmética-lógica. Consta, además, de registros y buses internos que generan las trayectorias de datos para la transferencia de información.

Funciones del procesador

- Ejecutar la secuencia de instrucciones contenidas en un programa almacenado en la memoria principal, luego de su decodificación.
- Desarrollar las operaciones aritméticas y lógicas que sean necesarias para procesar los datos.
- Leer y escribir contenidos en la memoria.
- Llevar y traer datos entre las celdas de memoria y los registros especiales.
- Controlar y supervisar el sistema integral de la computadora.
- Controlar el envío y recepción de datos desde las unidades periféricas a la unidad de memoria.

Para realizar sus funciones, la CPU se sirve de lo siguiente:

- a) Unidad de control.
- b) Unidad aritmético-lógica.
- c) Registros, buses.

Unidad de control

La *Unidad de control (UC)* dirige todas las actividades de la computadora. Para ello dispone de un sincronizador, que es un reloj electrónico que a intervalos regulares genera impulsos eléctricos que marcan un "ciclo de base", el ciclo de máquina. La ejecución de todas las operaciones elementales requiere un tiempo múltiplo de este ciclo de máquina. El ciclo de máquina suele ser de unos pocos nanosegundos (un nanosegundo equivale a la milmillonésima parte de un segundo).

La función de la unidad de control es interpretar las instrucciones y, luego, generar la secuencia de señales necesarias hacia las unidades correspondientes de la computadora que ejecutarán la instrucción. En base a la sincronización que le proporciona el clock y utilizando los ciclos de máquina necesarios, la unidad de control realiza los siguientes pasos:

- Determina la secuencia en que las instrucciones deben ejecutarse.
- Obtiene de la memoria la próxima instrucción a ejecutar.
- Interpreta la instrucción a ejecutar.
- Encarga la materialización de la instrucción a la ALU, si esta es de tipo aritmética o lógica; a un canal, si es de entrada o salida, y transfiere datos desde la memoria y hacia ella.
- Notificar al sistema operativo cualquier falla o avería que se detecte.
- Establecer la comunicación entre la ALU y la memoria principal, a través de la utilización de registros.

La Sección de Control está compuesta de la siguiente manera:

Registro de instrucción (IR)

Recibe la instrucción a ejecutarse proveniente de la memoria y la almacena temporalmente durante su ejecución.

Decodificador de instrucciones (ID)

Realiza la decodificación de la instrucción almacenada en el IR y envía señales en consecuencia al controlador-secuenciador.

Controlador-Secuenciador

Interpreta la información proveniente del ID; determina la ejecución de la instrucción dentro del procesador y fuera de él, generando señales de control hacia los componentes del sistema; estas señales se transmiten a través del bus de control y permiten el intercambio de información entre los distintos componentes.

Contador de programa (PC)

Contiene la dirección de la próxima instrucción a ejecutarse, es decir, que la "apunta". La ejecución de un programa es normalmente secuencial, para acceder a la instrucción siguiente es necesario extraerla previamente de la memoria donde se encuentra almacenado el programa. Para ello, se entrega el contenido del PC al bus de direcciones, que lo transmite a la memoria. Cada vez que se ejecuta una instrucción, el contenido del PC es automáticamente incrementado en una, dos y tres unidades, dependiendo del tipo de instrucción. Pero también puede darse el caso de que su contenido sea totalmente

modificado, tal es el caso de las instrucciones de ruptura de secuencia (saltos condicionales e incondicionales).

Puntero de pila (SP)

Identifica el elemento superior de la pila dentro de la memoria. La pila es una porción de memoria dinámica de tipo LIFO (último en entrar, primero en salir). Está formada por un conjunto de posiciones de memoria adscriptos a esa estructura de datos. El primer elemento introducido en la pila ocupa siempre su fondo, mientras que la introducción más reciente está siempre en la parte superior.

Una memoria de tipo LIFO trabaja de la siguiente manera: al momento de almacenar el DATO1, el SP apunta a una determinada dirección, por ejemplo, 10000; entonces, el DATO1 se ubica en este lugar. Una vez almacenado, el SP es decrementado automáticamente, listo para una nueva operación de almacenaje. Si se desea almacenar en una nueva dirección, el DATO2, este se ubicará en la posición a la cual quedó apuntando el SP, o sea, 9999; una vez realizada esta operación, el SP es nuevamente decrementado, y así sucesivamente.

Se ha ido conformando una pila de datos dentro del mapa de memoria; la función del SP es exclusivamente la de indicar en qué lugar han de colocarse los datos. De la igual manera, si queremos extraer un dato de la pila, este será el DATO2 (pues, el último que entra es el primero que sale).

Registros de índice (IX)

El empleo de índices permite acceder con una sola instrucción a bloques completos de datos contenidos en la memoria. El registro de índice suele contener un valor de desplazamiento que se suma automáticamente a una base (o viceversa). De esta forma, el índice da acceso a cualquiera de las palabras de un bloque de datos.

La interfaz primaria entre la memoria y la CPU se realiza por medio del MBR (Registro de datos de memoria) y del MAR (Registro de direcciones de memoria).

Registro de datos de memoria (MBR)

Es un registro temporal que se halla ubicado entre la memoria principal y los distintos componentes internos de la CPU. Todo dato que va a ser

procesado en la CPU o almacenado en la memoria pasa temporalmente por este registro.

Registro de direcciones de memoria (MAR)

Al igual que el anterior, este registro se halla ubicado entre la memoria central y la CPU. A través de él se direcciona la memoria central, se almacena temporalmente la dirección en cuestión.

Ciclo de la instrucción

Toda UC se encuentra, en todo momento, en uno de los dos estados del "Ciclo de procesamiento" que se exponen a continuación:

1. Semiciclo de búsqueda (FETCH): durante esta etapa del procesamiento se generan aquellos impulsos necesarios para leer o escribir instrucciones y datos en memoria. Se subdivide en lo siguiente:

   - Direccionamiento: la dirección correspondiente al dato (o instrucción) que se desea leer o escribir, se coloca en el MAR y, a través de este, pasa al BUS DE DIRECCIONES.
   - Memoria: el dato (o instrucción) ya seleccionado en la dirección de memoria indicada, se transfiere de la memoria central a la CPU (lectura); se emplea el BUS DE DATOS.

2. Semiciclo de ejecución (EXECUTE): en esta etapa se generan aquellos impulsos necesarios para ejecutar la instrucción almacenada en el Registro de Instrucción (IR). Previo a la ejecución se realiza una descodificación de la instrucción (en el Decodificador de Instrucciones), a través de la cual el procesador puede discriminar qué tipo de operación debe realizar.

Figura 5.6: Ciclo de la instrucción

Figura 5.7: Diag. de flujo del Ciclo de la instrucción

Buses

Un bus es un conjunto de conexiones que permite la circulación de la información entre los diferentes componentes del sistema. Existen tres buses: bus de direcciones, bus de datos, bus de control. La comunicación entre la CPU y los módulos externos se realiza por medio de los buses de direcciones y de datos.

Bus de datos

Transporta datos de unos elementos del sistema a otros; a través de él circulan las instrucciones y datos almacenados en la memoria hacia la CPU y los resultados para ser almacenados en la memoria. Según vemos, circula información en uno y otro sentido, por lo tanto, es bidireccional.

Bus de direcciones

A través de él circulan las direcciones que corresponden a los datos o instrucciones que se quieren leer o escribir; el sentido de la circulación es de la CPU hacia la memoria, es decir, el bus es unidireccional.

Bus de control

Conduce las diversas señales de control y sincronización que gobiernan el funcionamiento del sistema. Es bidireccional, por ejemplo, las líneas de control de lectura y escritura que llegan a la UC.

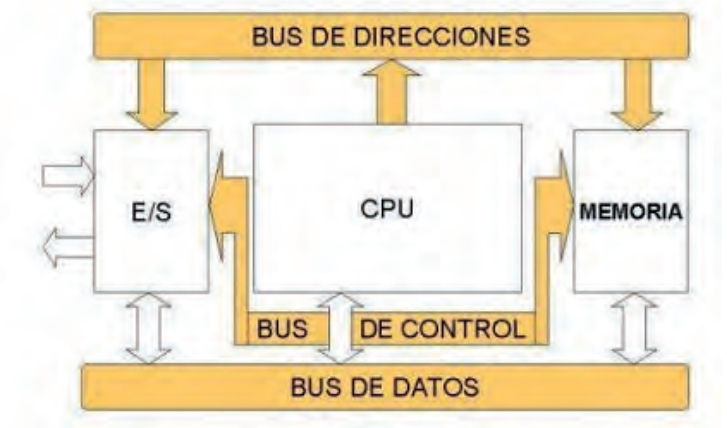


Figura 5.8: Buses

Unidad aritmético-lógica

También llamada UAL o ALU, es un sistema combinacional que bajo el gobierno de la UC se encarga de realizar las operaciones con los datos de acuerdo con el programa en curso; las operaciones que puede realizar son las siguientes: aritméticas elementales (suma, resta, y multiplicación y división en los más modernos), comparación, complementación, desplazamiento y rotación de bits; y lógicas (AND, OR, NOT, XOR, etcétera). Una unidad típica es capaz de hacer tan solo un número reducido de operaciones; la ejecución de las operaciones complejas se lleva a cabo descomponiéndolas en pasos elementales que se ejecutan a gran velocidad.

Esta sección de la máquina puede ser relativamente pequeña y consiste en uno o más circuitos de integración a gran escala (LSI). En los computadores más orientados hacia el cálculo científico puede estar formada por un gran conjunto de componentes lógicos de alta velocidad.

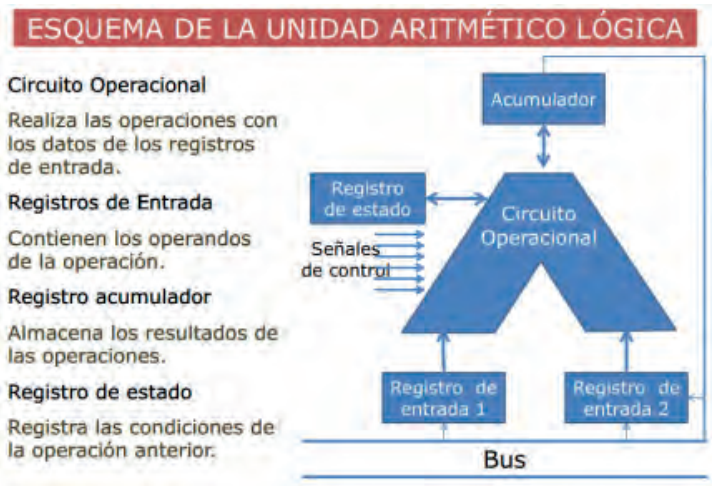


Figura 5.9: Unidad aritmético-lógica típica

La instrucción concreta a realizar viene indicada por la señal que envía la UC, aunque la UAL la ejecuta de manera autónoma. Busca los datos con los cuales operar de los registros convenientes y proporciona el resultado en un registro al efecto. La estructura de la UAL está formada por un conjunto de registros asociados a ella en los que se puede almacenar información y un

conjunto de circuitos lógicos que hacen posible la realización de operaciones sobre la información almacenada en los registros y operaciones entre registros.

Operadores

Circuitos electrónicos que realizan las funciones aritméticas o lógicas.

Acumulador

Es un registro temporal cuya función es almacenar los operandos con los cuales va a realizarse la operación que indique la UC, una vez producida almacena el resultado originado por los operadores.

Indicadores de estado (Registro de Estado) o banderas (Flags)

Es un conjunto de indicadores (biestables) asociados a la UAL que se activa cuando, al concluir una determinada operación, se detecta una cierta condición.

Existen diversos señalizadores o "bits de estado" entre los que se encuentran:

- *Bit de cero:* se pone a 1 si el resultado de la última operación ha sido cero.
- *Bit de acarreo:* se pone a 1 si el resultado de la última operación produjo acarreo.
- *Bit de signo:* se pone a 1 si el resultado de la última operación ha sido negativo.
- *Bit de desborde:* se pone a 1 cuando el resultado de la última operación excede la cantidad de dígitos representables por la UAL.

Un banco de registros de tipo general

Su función es almacenar datos. Son dispositivos electrónicos (registros biestables MOS - –ipo de dispositivo metal - –xido - –emiconductor) capaces de almacenar una pequeña cantidad de bits (8, 16, 32). La cantidad de estos registros que puede contener una CPU depende de su arquitectura interna. Los registros auxiliares son también unidades de memoria y difieren de la memoria central en que son de más rápido y fácil acceso (se utilizan para propósitos de tipo general, tales como el almacenamiento temporal de direcciones). Algunos de estos registros son accesibles al programador, y otros, no.

Un dato almacenado en un registro flip-flop puede manipularse de la siguiente manera:

- El registro puede ponerse a 0.
- El contenido del registro puede complementarse para obtener su complemento a 1 o a 2.
- El contenido del registro puede desplazarse a izquierda o derecha.
- El contenido del registro puede incrementarse o decrementarse.

En síntesis

Tanto las instrucciones como los datos que se van a procesar se hallan almacenados en la memoria central. El programa es registrado en la memoria antes de comenzar su ejecución.

A través del *bus de datos* circulan las instrucciones y los datos almacenados en la memoria hacia la CPU, y los resultados para ser almacenados, de la CPU a la memoria. El registro de instrucciones (IR) recibe la instrucción proveniente de la memoria central y la almacena temporalmente. El decodificador de instrucciones (ID) realiza la decodificación de la instrucción almacenada en el IR y envía el resultado obtenido al controlador-secuenciador, que interpreta el contenido y ejecuta la instrucción en consecuencia; es decir, genera las señales de control hacia los distintos componentes del sistema que se transmiten a través del *bus de control*.

El contador de programa (PC) sirve para direccionar la posición de la memoria de instrucciones donde se encuentra el programa y almacena la dirección que apunta la próxima instrucción a ejecutarse. El registro de direcciones (MAR) actúa como interfaz guardando el código de la dirección de memoria que se va a enviar a través del *bus de direcciones*.

Los registros auxiliares son unidades de memoria, pero de más rápido acceso que la memoria central. Si existe registro índice, es utilizado para llevar a cabo el direccionamiento indexado de la memoria. La UAL opera con los datos que recibe siguiendo órdenes de la UC, que analiza la instrucción y establece las conexiones eléctricas correspondientes dentro de la UAL. En el acumulador almacena los operandos con los cuales va a realizar la operación que indique la UC y, una vez producida esta, almacena el resultado. Los indicadores o flags del registro de estado (RE) se activan cuando al concluir una

operación, se cumplió una condición. El resultado de la operación de la UAL se almacena temporalmente en el acumulador o en la memoria central.

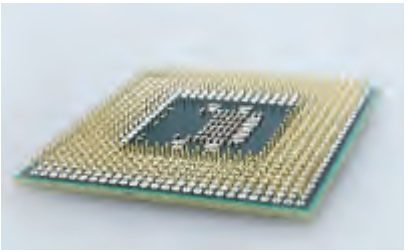


Figura 5.10: Procesador actual

MEMORIA PRINCIPAL

Llamamos memoria a todo dispositivo electrónico capaz de almacenar información binaria de la cual se puede obtener información para ser procesada cuando se necesite.

La totalidad de las memorias emplean el almacenamiento binario, es decir, que la información más elemental registrada es el bit, cuyo soporte físico llamamos "punto de memoria". Las características tecnológicas de la memoria de una computadora quedan determinadas por las características inherentes a su celda básica de almacenamiento o punto de memoria.

Clasificación de memorias

La clasificación de las unidades de memoria puede hacerse a partir de diferentes conceptos de referencia. Comúnmente se establece una primera clasificación general atendiendo la jerarquía que corresponde a la unidad de memoria dentro del sistema de proceso.

La jerarquía es un concepto de clasificación que obedece a algunas propiedades de las memorias: velocidad de trabajo, capacidad de almacenamiento, función dentro del sistema. Como sabemos, la memoria de una computadora no está concentrada en un solo sitio, los dispositivos de almacenamiento están dispersos por toda la máquina.

Los diversos tipos de memorias catalogados en los sucesivos órdenes jerárquicos son los siguientes:

1. Los *registros* de propósito general de los microprocesadores, de tipo flip-flop utilizados en la ALU y en la UC. Se trata de memorias de baja capacidad y alta velocidad; actúan habitualmente como memorias auxiliares en los procesos de transferencia de información.
2. La *memoria principal,* o *memoria central*, es un conjunto de celdas direccionables en donde la computadora almacena toda la información (datos y programas) que va a usar mientras esté

encendida. Cuando se realiza el procesamiento de datos, la información de memoria se transfiere, en primer lugar, a registros seleccionados de la CPU. Los resultados intermedios y finales que se obtienen en la CPU se vuelven a transferir a la memoria. La información binaria que se recibe de un dispositivo de entrada se almacena, primero, en la memoria, y la información que se transfiere a un dispositivo de salida, se toma de la memoria.

3. Las *memorias de masa* son de alta capacidad y se emplean como almacenamiento auxiliar. Para que la CPU pueda tratar determinada información, esta debe pasar inicialmente al interior de la memoria central del sistema.

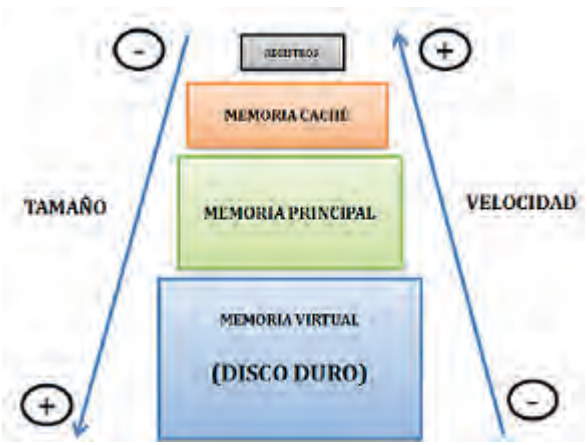


Figura 6.1: Relación Velocidad y Capacidad de almacenamiento

Memoria central o principal

Características de las memorias

- *Volatilidad*: en estas memorias, gobernadas por conmutación electrónica, se dice que la información almacenada es volátil si se ve alterada por falta de suministro eléctrico.

- *Direccionamiento*: técnica para localizar una información dentro de la memoria.
- *Modo de acceso*: aleatorio en este tipo de memorias.
- *Tiempo de acceso*: tiempo transcurrido desde que se solicita un dato a la unidad de memoria hasta que esta lo entrega.
- *Capacidad de almacenamiento*: número total de bytes que puede alojar.

Clasificación básica de las memorias centrales

Dentro del campo de las memorias centrales se establecen varios conceptos que dan lugar a diversas clasificaciones. Tomaremos los siguientes dos conceptos de referencia: modo de lectura y retención de la información almacenada.

1. Según el *modo de lectura*:

- *Memorias de lectura destructiva*: al leer determinada posición de memoria, la información almacenada desaparece. Este tipo de memorias precisan una regeneración del contenido después de efectuada la operación de lectura.
- *Memorias de lectura no destructiva*: las operaciones de lectura no provocan la pérdida de la información almacenada.

2. Según el *modo de retener la información*:

- *Memorias volátiles o no volátiles*: las memorias volátiles son aquellas que requieren la presencia de una fuente de alimentación, al desconectarlas de esta se pierde la información.
- *Memorias estáticas o dinámicas*: la información almacenada en una memoria estática permanece inalterable mientras no se modifique por actuación externa. La información almacenada en una memoria dinámica sufre una degradación con el tiempo, de tal forma que llega a desaparecer después de un intervalo más o menos prolongado. Para evitar esta pérdida de información deben enviarse periódicamente unos pulsos denominados "de refresco" que renuevan la información almacenada.

Clasificación de las memorias según el tipo de soporte

Las computadoras de la primera generación se caracterizaban por disponer de muy pocas celdas de memoria, ya que eran muy costosas y difíciles de

construir. La tecnología de las de la primera y de la segunda generación estuvo dominada por las memorias de ferrita (core memory), de las cuales luego explicaremos el funcionamiento.

En las máquinas de tercera generación, las ferritas fueron reemplazadas por memorias de semiconductores hechas con circuitos integrados a base de transistores. Se pueden construir por métodos industriales con las consiguientes ventajas de precio y cantidad.

Memorias de semiconductores

Existen dos tipos de memoria que se comunican directamente con la CPU: la *memoria de acceso aleatorio* (RAM) y la *memoria de solo lectura* (ROM). La RAM permite operaciones de *escritura*, o sea almacenamiento de nueva información y de *lectura*, es decir, transferencias de información desde la memoria. La ROM solo admite operaciones de *lectura*, por lo tanto, la información almacenada puede ser recuperada, pero no puede ser alterada ya que no se admite la escritura.

Memoria de acceso aleatorio (RAM)

Es una memoria de almacenamiento temporal de acceso aleatorio. Esto significa que se puede acceder a las celdas para transferir información hacia o desde cualquier ubicación adecuada deseada. Una memoria de este tipo consta de un conjunto de celdas de almacenamiento junto con circuitos asociados que se necesitan para transferir información dentro del dispositivo y fuera de este.

El contenido de estas memorias permanece mientras está presente la tensión eléctrica que las alimenta; es decir, mientras esté conectado el equipo.

Las computadoras usan invariablemente este tipo de memoria como principal o de trabajo, ya que permite la lectura/escritura.

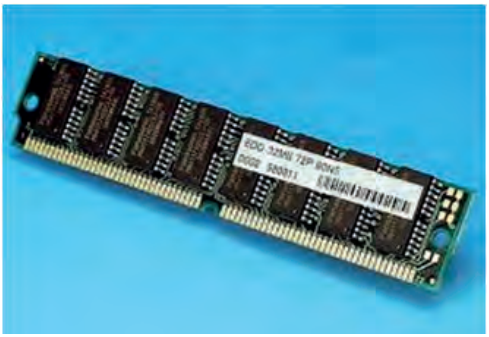


Figura 6.2: Memoria RAM

Una unidad de memoria almacena información en grupos de bits llamados *palabras*. Una palabra es una entidad de bits que entran y salen del espacio de almacenamiento como una unidad direccionable. Los unos y ceros de una palabra de memoria pueden representar un dato, una instrucción o cualquier información codificada en binario. La mayoría de las computadoras utilizan palabras de memoria que son múltiplo de un byte (8 bits), por ejemplo, 16 bits, 32 bits, etcétera.

Físicamente, las células se disponen en una matriz de dos dimensiones. El direccionamiento de la matriz, es decir el acceso a cualquier célula, se logra mediante las direcciones y señales de control adecuadas.

Analizaremos el siguiente esquema para representar cómo se comunica la memoria con su entorno: a través de líneas de entrada y salida de datos, líneas de direcciones y líneas de control.

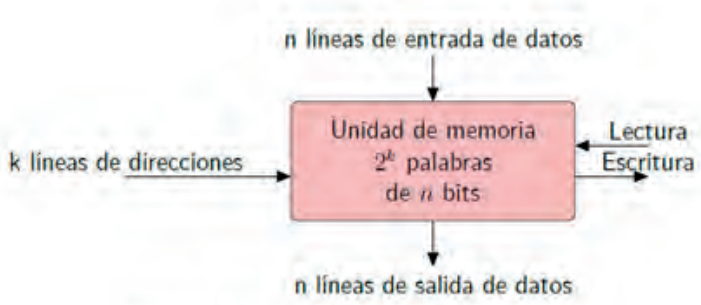


Figura 6.3: Bloque de una unidad de memoria.

Las *n líneas de entrada de datos* proporcionan la información que se almacenará en la memoria y las *n líneas de salida de datos*, la información que se transfiere desde la memoria. Las *k líneas de direcciones* especifican la palabra elegida entre las disponibles.

Las dos *entradas de control* especifican la dirección de la transferencia deseada: la de *lectura* hace que se transfieran datos fuera de la memoria; la de *escritura* hace que se transfieran datos binarios a la memoria.

La unidad de memoria se especifica por el número de palabras que contiene y el número de bits que hay en cada palabra. Cada palabra contenida en la memoria tiene un número que la identifica llamado *dirección*. Las direcciones van desde 0 hasta 2k - 1, donde k es el número de líneas de direcciones a las cuales se aplica la dirección binaria de k bits a la cual se quiere acceder. Un decodificador de direcciones integrado a la memoria acepta esa dirección y abre las trayectorias necesarias para seleccionar la palabra.

Estructura de la memoria

Como dijimos, la memoria consta de diversos bloques: matriz de memoria, decodificador de direcciones, lógica de control y registro de información. Analizaremos en primer lugar, la matriz de memoria.

Matriz de memoria

Agrupando un determinado número de puntos de memoria llegamos a obtener una celda de memoria. Si trabajamos con palabras de 1 byte, cada celda cuenta de 8 puntos de memoria, con respecto a los cuales definimos los siguientes conceptos:

- Dirección: es un número virtual que identifica la celda; está relacionada con la ubicación de la celda dentro de la matriz de memoria.

- Contenido: es la información que en cada instante se halla almacenada en una celda de memoria.

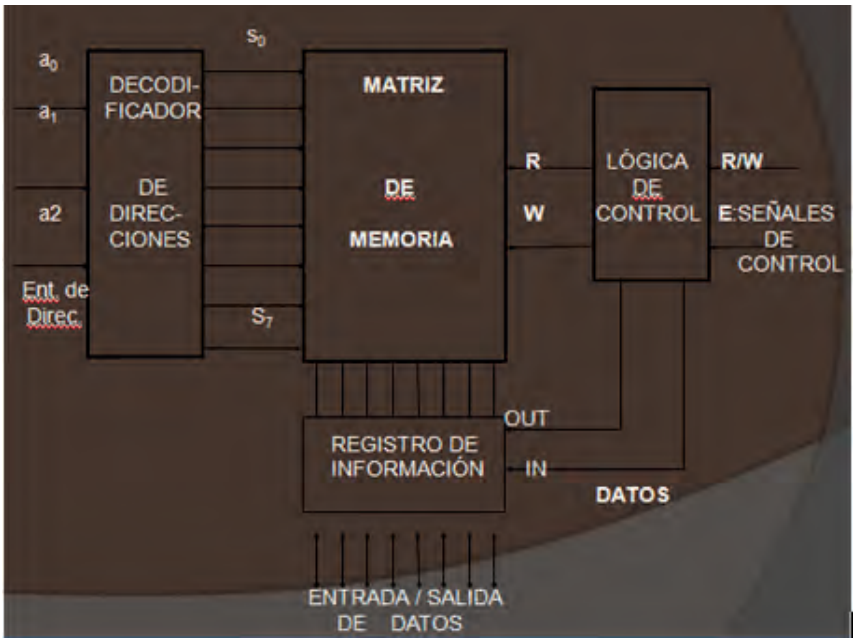


Figura 6.4: Funcionamiento de una memoria

Decodificador de direcciones

El circuito de direcciones tiene como tarea seleccionar la celda de memoria cuya dirección ingresa en la unidad de memoria a través de las líneas de direccionamiento.

Observando el modelo, vemos que las posibilidades de direccionamiento se concretaron en 8 celdas que almacenan sendas palabras de información. Para direccionar cualquiera de las 8 celdas son necesarias tres líneas de bit que transmitirán las configuraciones binarias correspondientes. Recordemos que con 3 bits pueden generarse hasta 2 configuraciones binarias distintas que, en este caso, cubren el margen de direccionamiento disponible. Las líneas de direccionamiento proceden de la CPU, a través del bus de direcciones.

Lógica de control

El circuito de control, o lógica de control, genera las órdenes de gobierno internas a la unidad de memoria a partir de dos comandos exteriores. Estos son los que se detallan a continuación:

- E (Enable) Autorización. Al llevar la entrada E a posicionamiento activo, la unidad de memoria queda autorizada o habilitada para efectuar sobre ella operaciones de lectura/escritura.
- R / W (Read / Write). Lectura / Escritura. Hallándose la unidad de memoria habilitada, la actuación del comando R/W es la siguiente:
    - R/W = 1 Lectura
    - R/W = 0 Escritura

A partir de dichas entradas, el circuito de control sintetiza las órdenes internas de lectura y escritura que acceden a los diversos puntos de memoria, así como las órdenes de entrada / salida (In / Out) para los amplificadores de líneas asociados al registro de información.

Registro de información

Está constituido por un número de biestables igual a la longitud de cada celda de memoria. Su tarea consiste en memorizar temporalmente las palabras de información que van a ser almacenadas o que han sido extraídas de la celda de memoria seleccionada por las líneas de direccionamiento.

Operaciones de lectura y escritura

La señal de escritura especifica una operación de transferencia de entrada y la señal de lectura especifica una operación de transferencia de salida. Al aceptar una de las señales de control, los circuitos internos de la memoria generan la función deseada.
Los pasos a seguir con el fin de transferir una nueva palabra a memoria son los que se detallan:

1. Transferencia de la dirección binaria de la palabra deseada a las líneas de direcciones.
2. Transferencia de los bits de datos que deben almacenarse en la memoria a las líneas de entrada de datos.
3. Activación de la entrada de *escritura*.
4. Entonces, la unidad de memoria tomará los bits de las líneas de datos de entrada y los almacenará en la palabra especificada por las líneas de direcciones.

Los pasos que deben seguirse para transferir una palabra almacenada fuera de la memoria son los que siguen a continuación:

1. Transferencia de la dirección binaria de la palabra deseada a las líneas de direcciones.
2. Activación de la entrada de *lectura*.
3. Luego, la unidad de memoria tomará los bits de la palabra que se haya seleccionado a través de la dirección y los aplicará a las líneas de datos de salida. El contenido de la palabra seleccionada no cambia después de la lectura.

Consideraremos un ejemplo de una memoria con una capacidad de 1K palabras de 16 bits cada una. Como 1K = 1024 = 2^{10} y 16 bits son dos bytes, podemos decir que esta memoria tipo puede contener *2 K bytes = 2048 bytes*.

Cuadro 6.1: Direcciones de memoria

BINARIO	DECIMAL	CONTENIDO
00 0000 0000	0	1011 0101 0001 1101
00 0000 0001	1	1100 1110 0011 0010
00 0000 0010	2	0001 0111 0001 0111
11 1111 1101	1.021	1100 1100 0001 1101
11 1111 1110	1.022	0000 0000 0111 0000
11 1111 1111	1.023	1000 1000 1000 1000

Cada palabra tiene 16 bits que se pueden dividir en dos bytes. Las palabras se identifican por sus direcciones decimales de 0 a 1023 (en binario = 1111111111). Cuando se lee o escribe una palabra, la memoria opera con los diez bits como una unidad.

Memorias estáticas y dinámicas

La memoria física de las computadoras actuales está formada por dispositivos de memoria de semiconductor. Hay dos tipos en el mercado: las *memorias de acceso aleatorio dinámicas (DRAM) y las estáticas (SRAM)*.

Por motivos de consumo y precio, ambos dispositivos se fabrican con una tecnología denominada CMOS (semiconductor de óxido metálico con salida

complementaria). Esta tecnología, aunque proporciona menor densidad de integración que su predecesora, la NMOS (semiconductor de óxido metálico de tipo "n"), tiene la ventaja fundamental de que su consumo de corriente es prácticamente nulo, salvo cuando hay conmutación de estado lógico. Por lo tanto, su consumo siempre es proporcional a la frecuencia de trabajo, pero siempre varios órdenes de magnitud por debajo de la tecnología NMOS, siendo, además mucho más fiable.

Entre los tipos DRAM y SRAM, hay también unas diferencias importantes a saber:

- La celda básica de memoria de una RAM dinámica se consigue con la integración de un único transistor; esta simplicidad permite que se puedan integrar varios millones de celdas de memoria en un único circuito integrado y que el coste de cada uno sea muy bajo.
- Las RAM estáticas utilizan el biestable como celda de memoria; el más sencillo que puede construirse con tecnología CMOS utiliza cuatro transistores. Por lo tanto, los niveles de integración son mucho más reducidos y el coste de memoria estática es más elevado.

Memorias RAM dinámicas

Estas memorias son más lentas que las estáticas; actualmente se sitúan en una velocidad de 60 ns, mientras que en las estáticas se pueden conseguir tiempos de acceso de 10 ns, siempre en tecnología CMOS.

Por ser la celda de memoria un condensador (y dado que en ellos siempre hay fugas), el dato almacenado en una memoria de este tipo se pierde si pasa cierto tiempo sin que se regenere, es decir, hay que leer el dato de cada celda periódicamente y volverlo a escribir. A este periodo de tiempo entre lecturas regeneradoras se le llama período de *refresco de memoria*. Se refresca la información contenida con una nueva escritura de un valor idéntico al almacenado.

Los períodos de refresco de los chips actuales son del orden de los 10 ms, aunque ya los hay de 20 ms. Todo esto se realiza de manera transparente, de tal forma que ni el procesador ni el usuario se enteran. Además, las memorias electrónicas verifican constantemente que la información almacenada no se altere ni se degrade, por medio de una técnica conocida como "control de paridad".

El diseño de un sistema de memoria usando DRAMs resulta más complicado que si se usaran SRAMs, pero la diferencia en costo y espacio es tan importante, que los fabricantes optan por el uso de las primeras.

Memorias RAM estáticas

Retienen su contenido durante el tiempo que se mantienen las tensiones de alimentación de corriente eléctrica. La RAM estática consta básicamente de flip-flop internos que almacenan la información binaria. La información almacenada sigue siendo válida en tanto se aplique energía a la unidad.

Tipos de memorias RAM

SRAM: Memoria estática de acceso aleatorio.

- *NVRAM* Memoria de acceso aleatorio no volátil.
- *MRAM* Memoria de acceso aleatorio magnética.

DRAM RAM dinámica, memoria dinámica de acceso aleatorio.

- *DRAM Asincrónica* Memoria de acceso aleatorio dinámica asincrónica
- *SDRAM* Memoria de acceso aleatorio dinámica sincrónica.
- *SDR SDRAM* SDRAM de tasa de datos simple.
- *DDR SDRAM* SDRAM de tasa de datos doble.
- *DDR2 SDRAM* SDRAM de tasa de datos doble de tipo dos.
- *DDR3 SDRAM* SDRAM de tasa de datos doble de tipo tres.
- *DDR4 SDRAM* SDRAM de tasa de datos doble de tipo cuatro.

Memoria de solo lectura (ROM)

Una memoria de solo lectura es básicamente un dispositivo en el cual se almacena información binaria permanente. La información binaria debe ser especificada por el diseñador y después incorporada en la unidad para formar el patrón de interconexión requerido. Las ROM vienen con fusibles electrónicos internos especiales que se pueden programar para generar una configuración específica. Una vez establecido el patrón, este permanece dentro de la unidad aun cuando se interrumpe el suministro de energía y se vuelve a activar.

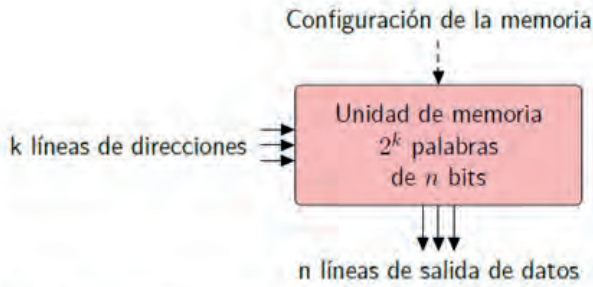


Figura 6.5: Diagrama simplificado de la unidad de memoria.

En la figura 6.5 se ilustra una ROM que consta de k entradas y n salidas. Las entradas proporcionan la dirección de memoria y las salidas dan los bits de datos de la palabra almacenada que selecciona la dirección. El número de palabras de una ROM se determina a partir de que se necesitan k líneas de entrada de direcciones para especificar 2^k palabras. La memoria no acepta operaciones de escritura, pero cada tecnología define el proceso por el cual se cargan los valores almacenados en esta. Ese proceso está indicado en la figura como el proceso de configuración.

El almacenamiento binario interno de una ROM lo especifica una tabla de verdad que muestra el contenido de palabras en cada dirección. El procedimiento de *hardware* que programa la ROM ocasiona que se fundan fusibles internos de acuerdo con una tabla de verdad dada. Todo 0 presentado en la tabla de verdad especifica un fusible que se fundirá, y todo 1 citado especifica una trayectoria que se obtiene a través de un fusible intacto. Por ejemplo, una tabla especifica una palabra de 8 bits 10110010 para su almacenamiento permanente en la dirección de entrada 00011. Los ceros de la palabra se programan fundiendo los fusibles entre la salida 3 del decodificador y las entradas de las compuertas asociadas con las salidas. Los cuatro unos de la palabra se marcan en el diagrama con una cruz para designar un fusible intacto.

Figura 6.6: Memoria ROM

Tipos de ROM

Las trayectorias que se requieren en una ROM se pueden programar en formas diferentes:

- La primera se denomina programación con máscara y la realiza la compañía de semiconductores durante el último proceso de fabricación de la unidad. Básicamente, un *ROM de máscara* es una matriz de conmutadores que conservan de modo permanente su estado abierto o cerrado. El proceso de fabricar una ROM requiere que el comprador llene la tabla de verdad que desee que la ROM satisfaga. El fabricante crea la "máscara" correspondiente de las trayectorias a fin de producir los unos y los ceros de acuerdo con la tabla de verdad del comprador.
  La programación con máscara es conveniente solo si se ordena una gran cantidad de la misma configuración de ROM; de otro modo resulta costosa.

- Un segundo tipo de ROM que resulta más económico para pequeñas cantidades, se denomina *PROM* (memoria permanente y programable de solo lectura). Es más flexible que el anterior ya que puede programarse fuera del ámbito de fabricación. Cuando se solicitan, las unidades PROM contienen todos los fusibles intactos, lo que hace que haya exclusivamente unos en los bits de las palabras almacenadas. Los fusibles de la PROM se funden por la aplicación de pulsos de corriente a través de las terminales de salida de cada dirección. Un fusible fundido define un estado 0 binario y el fusible intacto genera un estado 1

binario. Esto permite al usuario programar la PROM en su laboratorio para obtener la configuración deseada.
Para programar una memoria de este tipo se suelen usar dispositivos especiales llamados PROM burners. El procedimiento de *hardware* para programar ROM o PROM es irreversible y una vez que se programa el patrón fijo es permanente, por lo que, si se quiere cambiar el patrón de bits, la unidad tiene que desecharse.

- Un tercer tipo de ROM disponible recibe el nombre de *EPROM* (memoria permanente y reprogramable de solo lectura). La EPROM se puede reestructurar al valor inicial, aun cuando sus fusibles se hayan fundido previamente. Cuando la EPROM se coloca bajo una luz ultravioleta especial por un espacio de tiempo dado, la radiación de onda corta descarga las compuertas internas que sirven como fusibles. Después del borrado, la EPROM regresa a su estado inicial y se puede reprogramar a un nuevo conjunto de palabras. Suelen recibir el nombre de *UVPROM*.

Figura 6.7: Memoria EPROM

- Ciertas PROM pueden borrarse con señales eléctricas en lugar de luz ultravioleta, por lo cual, reciben el nombre de *EEPROM* (memoria permanente de solo lectura y borrado eléctrico). Aparecieron a principios de los 80. Una variedad de este tipo de memoria es la EAPROM

(memoria permanente alterable de solo lectura y borrado eléctrico), más moderna que las anteriores; permite borrar palabras por separado y tiene una permanencia de varios años.

Memoria caché

Definición de memoria caché

Literalmente, se trata de una palabra en francés que quiere decir "escondido" u "oculto". Pero tiene un uso en la informática que le ha dado nombre a un tipo particular de memoria.

La memoria caché de un procesador es un tipo de memoria volátil (como la memoria RAM), pero muy rápida. Su función es almacenar instrucciones y datos a los que el procesador debe acceder continuamente.

¿Cuál es su finalidad? Que este tipo de datos sea de acceso instantáneo para el procesador, ya que se trata de información relevante que debe estar a la mano de manera muy fluida. Los sistemas de *hardware* y *software* llamados caché, almacenan este tipo de datos de manera duplicada y por esta razón su acceso es tan veloz.

En resumen, se trata de aquella cantidad de datos que permanece de manera temporal en un sistema, lo que ayuda a que el rescate de datos se haga de manera más eficiente y veloz. En palabras simples, la memoria caché está diseñada para hacer más organizado el almacenamiento de datos en un sistema, entiéndase computador, celular o cualquier otro dispositivo que contenga un procesador.

Funcionamiento de la memoria caché

Cada vez que el sistema quiere acceder a un nuevo dato, este es almacenado en la memoria caché. Entonces, cuando se necesita recurrir nuevamente a ese dato, el sistema se dirigirá directamente al caché, haciendo así el proceso mucho más rápido. Este ciclo de almacenamiento y rescate de datos obliga a la memoria caché a estar en continua renovación.

Su función, entonces, es mantener de manera temporal y accesible aquellos datos que son requeridos por el sistema para realizar determinadas funciones o tareas. Así, cada vez que abras una app en tu smartphone, esta tendrá acceso inmediato a la información que necesita para subir el nivel de eficiencia de sus funciones.

INSTRUCCIONES Y MODOS DE DIRECCIONAMIENTO

Ciclo de ejecución de las instrucciones

Recordando lo visto sobre el ciclo de la instrucción (Fetch-Execute), analizaremos cómo el procesador realiza estas funciones:

1- Tomar la siguiente instrucción.
2- Decodificar la instrucción.
3- Ejecutar la instrucción.

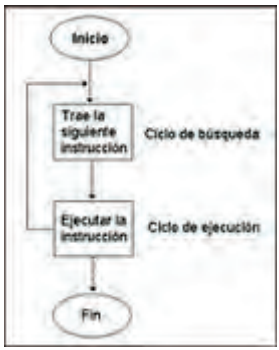


Figura 7.1: Ciclo de Ejecución Fetch-Execute

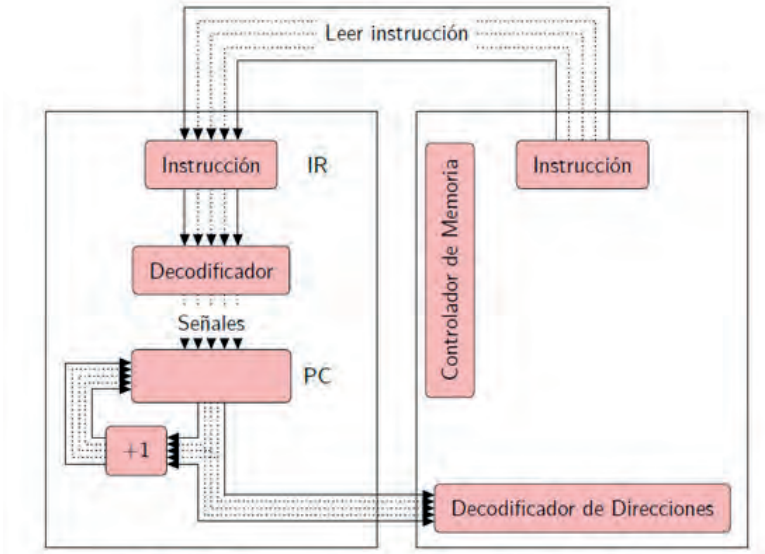


Figura 7.2: Comunicación entre el procesador y la memoria.

Búsqueda de la instrucción

- En la primera parte del ciclo, el contenido del contador de programa se deposita en el bus de direcciones (utilizando una interfaz primaria, el MAR: registro de direcciones de memoria) que lo conduce hasta la memoria.
- Simultáneamente, se emite una señal de lectura a lo largo del bus de control del sistema.
- La memoria recibe la dirección que especifica una de sus posiciones.
- Al recibir la señal de lectura, decodifica la dirección por medio de su propio decodificador y selecciona la posición indicada en ella.
- Algunos nanosegundos después, la memoria deja el dato correspondiente a la dirección pedida en el bus de datos (se utiliza como interfaz con la CPU el MBR: registro de datos de memoria).
- El microprocesador lee el bus de datos y deposita su contenido en un registro interno llamado IR: registro de instrucción, que almacena la instrucción que se acaba de tomar de la memoria.
- La fase de tomar del ciclo de ejecución de una instrucción ha terminado.

Decodificación y ejecución

- Una vez alojada la instrucción en el IR, la UC decodifica su contenido en el decodificador de instrucciones.
- El secuenciador genera las microórdenes o secuencia de señales externas e internas necesarias para ejecutar la instrucción; es decir, activar y gobernar los distintos elementos constitutivos del sistema.
- El tiempo de ejecución de cada tipo de instrucción es diferente; ese tiempo se mide en ciclos de reloj, ya que su duración es variable.

Siguiente instrucción a ejecutar

- Durante la ejecución de un programa, las instrucciones se toman en secuencia de la memoria; por ello es preciso prever un mecanismo automático que adopta la forma de un sumador conectado al PC: contador de programa.
- Esto funciona de la siguiente forma: cada vez que se coloca en el bus de direcciones el contenido del PC, dicho contenido se incrementa en una unidad y se vuelve a escribir en el registro contador (mecanismo automático de secuenciación de instrucciones). Este mecanismo es válido excepto en el caso de las instrucciones de salto donde el PC toma el valor de la dirección de la instrucción hacia donde bifurca el programa.
- La descripción anterior es una simplificación de la realidad, porque algunas instrucciones pueden tener más de una palabra de memoria, lo cual obliga a tomarlas una tras otra; sin embargo, el mecanismo fundamental es idéntico; el contador de programa se usa tanto para tomar bytes sucesivos de una instrucción como para tomar sucesivas instrucciones. Por tanto, el dispositivo formado por el contador y el sumador señala posiciones sucesivas de la memoria.

Instrucciones de computadora

La estructura física y lógica de las computadoras se describe normalmente en manuales de consulta que se proporcionan con el sistema. Dichos manuales explican la construcción interna de la computadora e incluyen los registros del procesador. Se presenta una lista de todas las instrucciones que se aplican en *hardware*, su código binario y una definición exacta de lo que hace cada instrucción.

El formato de una instrucción se representa en una figura rectangular que simboliza los bits del código de instrucción.

Los bits del código de una instrucción binaria se dividen en grupos que subdividen la instrucción en partes llamadas campos; estos constituyen información de interés para el secuenciador como son el código de operación, las condiciones de direccionamiento, la dirección de registros o de unidades periféricas, y demás.

Cada campo, por tanto, especifica diferentes funciones para la instrucción y cuando se muestran juntos constituyen el formato de la instrucción. Los más comunes son los siguientes:

1. Campo de código de operación: especifica la operación que se realizará con los datos.
2. Campo de dirección u operando: puede designar una dirección de la memoria, un código para elegir un registro del procesador o un dato.
3. Campo de modo: especifica la forma en que se interpretará el campo de operando.

El campo de código de operación de una instrucción es un grupo de bits que definen diversas operaciones del procesador, como la adición, sustracción, complemento y corrimiento. Los manuales de cada procesador presentan una lista de todas las instrucciones que se aplican al *hardware*, especifican su formato en código binario y ofrecen una definición exacta de cada instrucción. La función de la *UC* será interpretar cada código de instrucción y proporcionar las señales de control necesarias para procesar cada instrucción.

El campo de operando: las operaciones especificadas por instrucciones se ejecutan con algunos datos almacenados en memoria o en registros del procesador. Los operandos que residen en memoria se especifican por medio de sus direcciones; los que residen en registros, por medio de una dirección de registro, que es un código binario de *n* bits (dependiendo de la cantidad de registros del procesador).

Formatos de instrucciones

Las computadoras pueden tener instrucciones de longitudes diferentes. El número de campos de dirección en el formato de instrucción de una computadora depende de la organización interna de sus registros.

El formato de una instrucción puede representarse en una figura rectangular que simboliza los bits de la instrucción conforme aparecen en las palabras de la memoria o en un registro de control. A modo de ejemplo, considere los tres formatos de códigos de instrucciones que se representan en las siguientes figuras:

1. Operando implicado o implícito. [Código de operación]
   Este formato de instrucción consta de un código de operación que implica un registro de la unidad procesadora; se puede utilizar, por ejemplo, para especificar instrucciones como "incrementar un registro del procesador", "complementar el registro acumulador", etcétera.
2. Operando inmediato. [Código de operación | Operando]
   Este formato de instrucción tiene un código de operación seguido de un operando; a esta se la denomina de "operando inmediato" porque este está inmediatamente después del código de operación; se puede utilizar para especificar instrucciones como "sumar el operando al contenido de un registro" o "transferir el operando al registro acumulador", o cualquier otra operación a efectuar entre un operando dado y un registro del procesador.
3. Dirección directa. [Código de operación | Dirección de Operando]
   Este formato de instrucción indica la dirección del operando en la memoria. Es decir que, la operación especificada por el código de operación se realiza entre un registro del procesador y un operando que está almacenado en una celda de memoria; la dirección de dicha celda está incluida como parte de la instrucción (es la segunda palabra).

Suponiendo que se trabaja con una pequeña computadora que tiene una unidad de memoria de 8 bits por palabra y que un código de operación contiene 8 bits, analizaremos en la siguiente figura cómo se almacenarían en la memoria instrucciones como las vistas anteriormente:

Dirección (decimal)	Contenido de Memoria	Cód. Op.	Descripción
80	0000 0001	1	$R \leftarrow R + 1$
	...		
100	0000 0011	3	$R \leftarrow 3$
101	0010 1100		$Operando = 44$
	...		
112	0000 1001	9	$R \leftarrow R(dir)$
113	0111 1111		$dir = 127$
	...		
127	0001 0001		$Operando = 17$

Figura 7.3: Ejemplo de estado de memoria.

Lo que debemos reconocer es la relación que existe entre una operación de la computadora y una microoperación de *hardware*. Una operación está especificada por una instrucción almacenada en la memoria de la computadora, la cual es un código binario que indica a la computadora una operación específica. La unidad de control recupera la instrucción de la memoria e interpreta los bits del código de operación; emite una secuencia de funciones de control para realizar las microoperaciones requeridas para la ejecución de la instrucción.

Campo de dirección

Según vimos, las operaciones especificadas por instrucciones de computadora se ejecutan con algunos datos almacenados en la memoria o en registros del procesador:

- Los operandos que residen en memoria se especifican por medio de sus direcciones.
- Los operandos que residen en registros del procesador se especifican mediante una dirección de registro.

El número de campos de dirección en el formato de instrucción de una computadora depende de la organización interna de sus registros. La mayoría de las instrucciones se pueden clasificar en uno de los siguientes tipos de organización: de un solo acumulador, de registros múltiples, de pila.

1. Organización de un solo acumulador.
   En el primer caso, todas las operaciones se realizan con el registro acumulador implicado. El formato de instrucción en este caso utiliza un campo de dirección de la memoria y su representación es: [Código de operación | Dirección de Operando]
   Por ejemplo, una instrucción de suma como la siguiente tiene un solo campo de dirección que es X:

   ADD X
   ADD = código de operación
   X = dirección del operando en memoria.

   Mediante esta instrucción se realiza la siguiente operación:
   $AC \leftarrow AC + M[X]$
   AC = registro acumulador
   M[X] = palabra de memoria en la dirección X.

2. Organización de registros múltiples.
   El formato de instrucción en este tipo de computadora necesita de más de un registro. Por lo tanto, la instrucción de adición aritmética puede escribirse en forma simbólica como:

   ADD R1, R2, R3
   Denota la operación: sumar R1 + R2 y depositar en R3.

   $R3 \leftarrow R1 + R2$

   El número de campos de dirección de registro en la instrucción se puede reducir de tres a dos si el registro destino es el mismo que uno de los registros fuente. Por lo cual, la instrucción:

   ADD R1, R2
   Denota la operación sumar R1 y R2 y depositar el resultado en R2.

   $R2 \leftarrow R2 + R1$
   R1 y R2 son registros fuente

R2 también es registro destino.

Las computadoras de múltiples registros emplean dos o tres campos de dirección en el formato de instrucción. Cada campo de dirección puede especificar un registro del procesador o bien una dirección de memoria.

Una instrucción simbolizada como:
ADD X, R1
Denota la operación sumar al registro R1 el contenido de la posición de memoria X y depositar el resultado en R1.

$R1 \leftarrow R1 + M[X]$ utiliza dos campos de dirección: uno para R1, otro para X.

Modos de direccionamiento

Se llama direccionamiento a la especificación dentro de una instrucción de la posición del operando sobre el que actuará la misma.

Según vimos, una operación se debe realizar con algunos datos almacenados en registros de la computadora o en palabras de memoria. La forma en que se escogen los operandos durante la ejecución del programa depende del modo de direccionamiento de la instrucción.

En esta sección analizaremos las técnicas de direccionamiento más comunes:

- Inmediato.
- Directo.
- Indirecto.
- Registro.
- Indirecto con registro.
- Con desplazamiento.

Hay que resaltar que la mayoría de las arquitecturas de computadores ofrecen más de uno de estos modos de direccionamiento. La unidad de control determina qué modo de direccionamiento se está empleando en cada instrucción. Se pueden adoptar varias alternativas; en algunas se utilizan códigos de operación para indicar el tipo de direccionamiento que se usa; en otras, se utilizan uno o más bits del formato de instrucción para indicar el modo de

direccionamiento (campo de modo). El valor del campo de modo determina el modo de direccionamiento va a utilizarse.

Ejemplo: addinm (suma inmediata), adddir (suma directa), addreg (suma con registro), etcétera.

Tomemos como ejemplos palabras de 32 bits.

Modo inmediato

- El operando está en la instrucción.
- El operando es una constante a tiempo de ejecución.
- No hay referencias adicionales a memoria después de la búsqueda de la instrucción.
- El tamaño del operando está limitado al tamaño del campo de dirección.

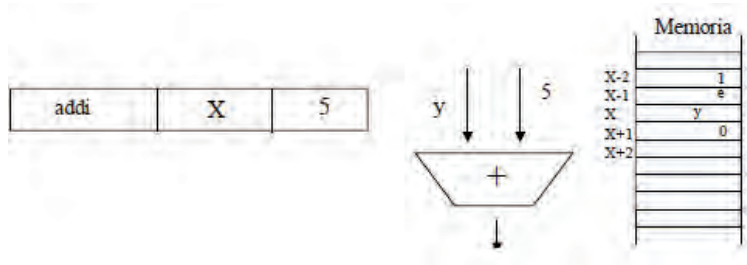


Figura 7.4: Modo de Direcc. Inmediato.

Modo directo

Una forma sencilla de direccionamiento es el directo en el que el campo de dirección contiene la dirección efectiva del operando.

Este modo fue común en las primeras generaciones de computadores y aún se encuentra en un número pequeño de sistemas.

Ventaja: solo requiere una referencia a memoria para obtener el operando y no necesita ningún cálculo especial.

Desventaja: la limitación directa es que proporciona un espacio limitado de direcciones.

La instrucción contiene la dirección efectiva del operando, el acceso adicional es requerido para obtener el operando. El rango de direcciones está limitado por el ancho del campo que contiene la dirección del operando y la

dirección es una constante a tiempo de ejecución, pero el operando en sí puede variar a tiempo de ejecución.

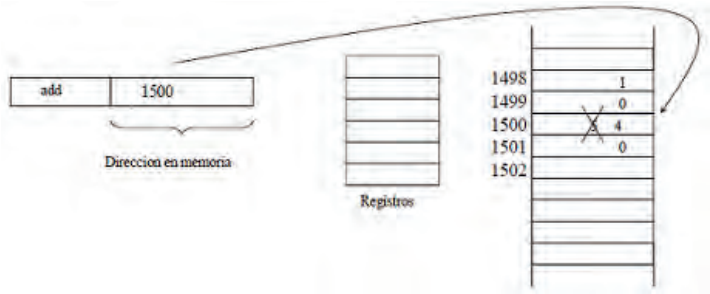


Figura 7.5: Modo de Direcc. Directo.

Modo indirecto

Ventaja: permite acceder a todo el espacio de direcciones. No hay limitación, como en el caso anterior de modo directo.

Desventaja: la ejecución de la instrucción requiere de dos referencias a memoria para capturar el operando; una, para captar su dirección, y otra, para obtener su valor.

Una variante, raramente utilizada, es el modo indirecto de varios niveles. Esta aproximación no presenta realmente una ventaja significativa, y su desventaja principal es que pueden requerirse dos o más referencias para captar un operando.

La instrucción contiene la dirección de memoria que indica la dirección del operando. Se requieren dos accesos a memoria para obtener el operando. El rango de direcciones efectivas es de 2^n, donde n es el ancho de la palabra. El número de direcciones de memoria que puede ser usado para almacenar direcciones efectivas es de 2^k, donde k es el ancho del campo dirección en la instrucción.

Figura 7.6: Modo de Direcc. Indirecto.

Modo registro

El campo de dirección referencia un registro en lugar de una dirección de memoria principal.

Ventaja: solo es necesario un campo pequeño de direcciones en el acceso a un registro interno al CPU, es menor que el tiempo necesario para acceder memoria principal.

Desventaja: el espacio de direcciones está muy limitado dado al pequeño número de registros que poseen las arquitecturas.

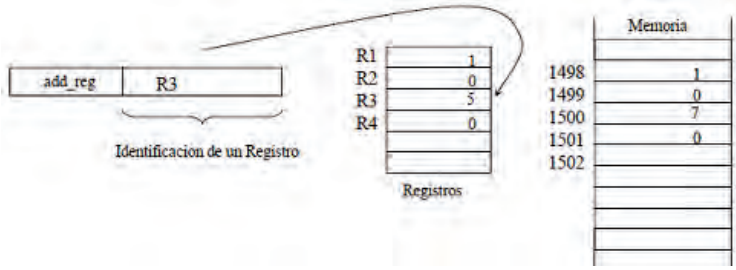


Figura 7.7: Modo registro

Modo indirecto con registro

Es similar al indirecto. A diferencia del indirecto, el campo de dirección hace referencia a una posición de memoria que contiene la dirección del operando, mientras que en el indirecto con registro, el campo de dirección hace referencia a un registro que contiene la dirección efectiva del operando.

Ventaja: emplea una referencia menos a memoria que el modo indirecto.

Desventaja: la ejecución de la instrucción requiere una referencia a memoria para capturar el operando.

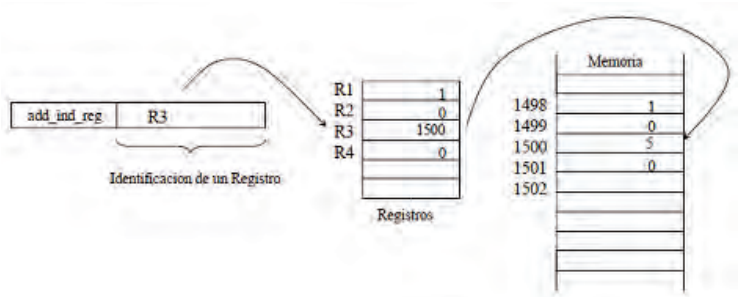


Figura 7.8: Modo indirecto con registro

Modo con desplazamiento

Este modo combina los modos directo e indirecto con registro. Necesita que las instrucciones tengan dos campos de dirección donde, al menos, uno de ellos es explícito. El valor contenido en uno de los campos de dirección se utiliza directamente, mientras que el otro campo de dirección o referencia implícita definida por el código de operación se refiere a un registro cuyo contenido se suma al primer campo de dirección para generar la dirección efectiva del operando.

Hay tres formas distintas de modos con desplazamiento:

• **Desplazamiento relativo**

El registro referenciado implícitamente es el contador de programa (PC).

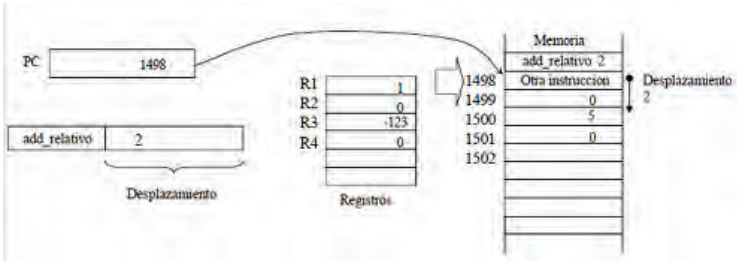


Figura 7.9: Modo con desplazamiento a PC

• **Desplazamiento con registro base**

El registro referenciado implícitamente es un Registro.

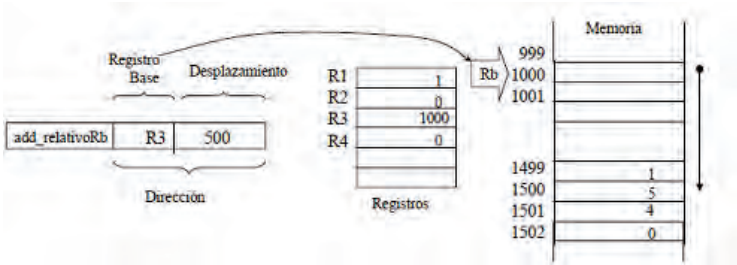


Figura 7.10: Modo con desplazamiento a registro

• **Indexado (o relativo a un registro índice)**

Este modo de direccionamiento es empleado especialmente para el recorrido de estructuras.

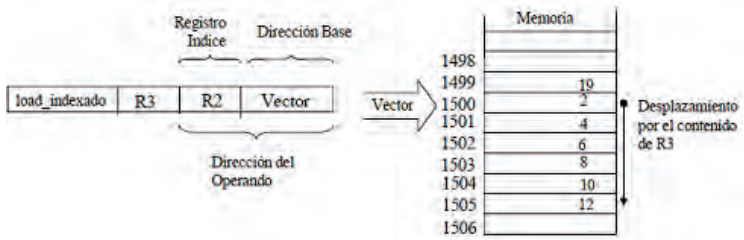


Figura 7.11: Modo con desplazamiento

Se han visto diferentes modos de direccionamiento, y conviene comentar brevemente acerca de cómo el *hardware* sabe qué tipo de direccionamiento debe utilizar con la instrucción que acaba de ser leída desde la memoria principal.

Existen diferentes maneras de solucionar este problema. Una puede ser tener un código de operación distinto para cada método, es decir, tener diferentes códigos de operación; otra forma consiste en hacer que el modo de direccionamiento quede descripto por un campo del formato de instrucción.

MEMORIA AUXILIAR

La memoria secundaria, memoria auxiliar, memoria periférica o memoria externa, es el conjunto de dispositivos y soportes de almacenamiento de datos que conforman el subsistema de memoria de la computadora, junto con la memoria primaria o principal, la memoria caché y los registros del CPU.

Subsistema de memoria de la computadora

La memoria secundaria es un tipo de almacenamiento masivo y permanente (no volátil) con mayor capacidad para almacenar datos e información que la memoria primaria, que es volátil y rápida. Puede denominarse periférico de almacenamiento o "memoria práctica", en contraposición a la "memoria central", porque en ocasiones puede considerarse como periférico de Entrada/Salida.

Clasificación de las memorias auxiliares

Deben diferenciarse los "dispositivos o unidades de almacenamiento" de los "soportes o medios de almacenamiento", porque los primeros son los aparatos que leen o escriben los datos almacenados en los soportes. El proceso de transferencia de datos a un equipo de cómputo o sistema informático se llama "procedimiento de lectura". El proceso de transferencia de datos desde la computadora hacia el almacenamiento se denomina "procedimiento de escritura" o grabación.

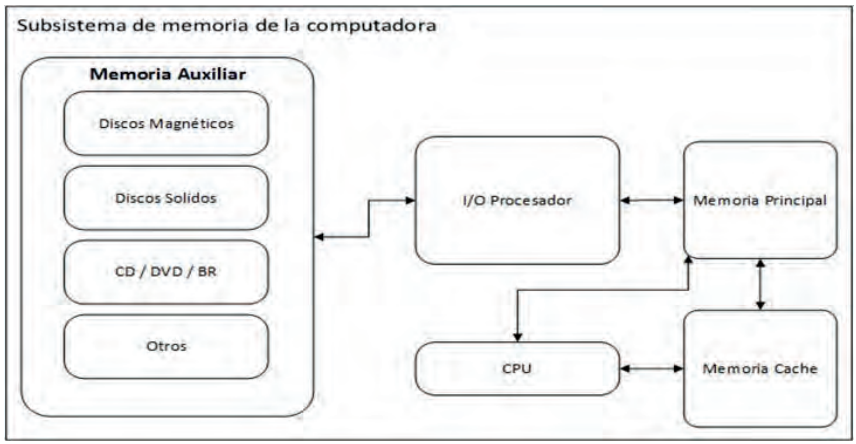


Figura 8.1: Subsistema de Memoria de la computadora

Para almacenar información se pueden usar los siguientes tipos de tecnología:

Magnética: disquete, disco duro, cinta magnética.

Óptica: CD, DVD, BD.

Magneto-óptica: disco zip, floptical, minidisc. (No desarrollada)

Estado sólido, almacenamiento electrónico, o memoria flash: memoria USB o pendrive; tarjetas de memoria: SD, miniSD, microSD, MS, MMC, CF, SM.

La mayoría de los dispositivos y medios de almacenamiento emplean una tecnología magnética o tecnología óptica; algunos, llamados híbridos (almacenamiento magneto-óptico), utilizan ambas. Otra categoría de almacenamiento, como el dispositivo de estado sólido, se utiliza con mayor frecuencia en las computadoras portátiles (netbooks, notebooks, ultrabooks), así como también en cámaras digitales, teléfonos inteligentes, tabléfonos y reproductores multimedia.

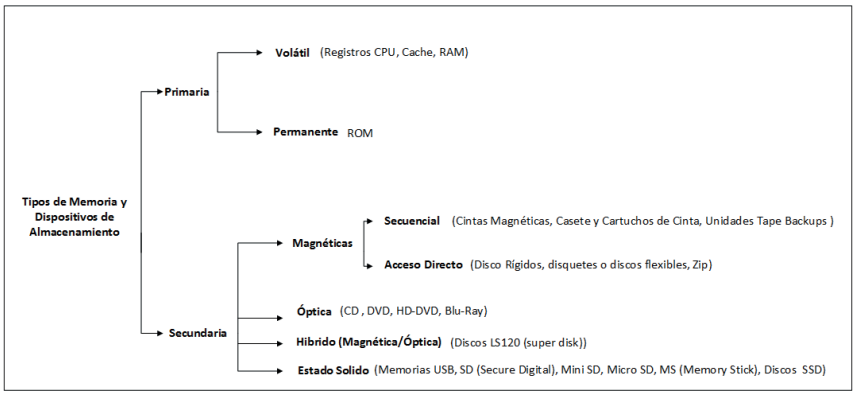


Figura 8.2: Clasificación de la memoria

Tecnología de soporte magnético

El soporte magnético es uno de los tipos de medios o soportes de almacenamiento de datos en los que se usan las propiedades magnéticas de los materiales para almacenar información digital.

La lectura y grabación de la información en un dispositivo de almacenamiento magnético se da por la manipulación de partículas magnéticas presentes en la superficie del medio magnético. Para la grabación, el cabezal de lectura y grabación del dispositivo genera un campo magnético que magnetiza las partículas magnéticas, presentando así dígitos binarios (bits) de acuerdo a la polaridad utilizada. Para la lectura, el cabezal de lectura y grabación genera un campo magnético, que cuando entra en contacto con las partículas magnéticas del medio verifica si estas atraen o repelen al campo magnético, de esta manera saben si el polo encontrado en la molécula es positivo o negativo.

Clasificación del soporte magnético según tipo de acceso

Existen básicamente dos tipos de unidades periféricas magnéticas:

1. Aquellas donde la información se lee/graba de manera secuencial. En el caso de lectura, se debe recorrer todo el dispositivo desde el principio hasta ubicar la información deseada. Para escribir algún dato nuevo se lo ubica al final.

2. Aquellas donde el acceso a los datos se hace de manera directa o aleatoria. Se puede acceder a la información para lectura o escritura en forma directa.

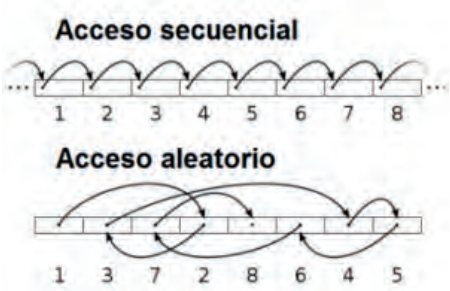


Figura 8.3: Tipos de acceso

Acceso secuencial

Cintas magnéticas

La UNIVAC I (1951) inició una nueva tendencia en el almacenamiento de datos: la cinta magnética. IBM pronto comenzó a usar carretes de cinta magnética (similar a la cinta de audio de ese entonces) para el almacenamiento de datos de las computadoras, y el resto de la industria la siguió. La cinta de computadora, generalmente guardada en carretes abiertos, usualmente consistía en delgadas tiras de plástico cubiertas con una sustancia sensible al magnetismo en las que las computadoras escribían y leían por medio de cabezales electrónicos incorporados en un drive especial de cinta.

Numerosos modelos de computadoras de producción (especialmente los mainframes y las minicomputadoras) usaron las cintas de carrete abierto como medio de almacenamiento masivo hasta los años 70 y 80, cuando los diseñadores cambiaron a los cartuchos de cintas.

Originalmente fueron los sistemas principales de almacenamiento masivo. Hoy su función principal se reduce a la de seguridad o respaldo de la información en disco (backup). Es el medio más popular actualmente para almacenar archivos históricos o grandes volúmenes de información que no se utilizan diariamente. Debido a que el tiempo de acceso es muy lento, la cinta no es

recomendable como almacenamiento principal, pero por el bajo costo su función básica se transformó en la de respaldo. Se diferencian por su capacidad y velocidad de lectura/escritura.

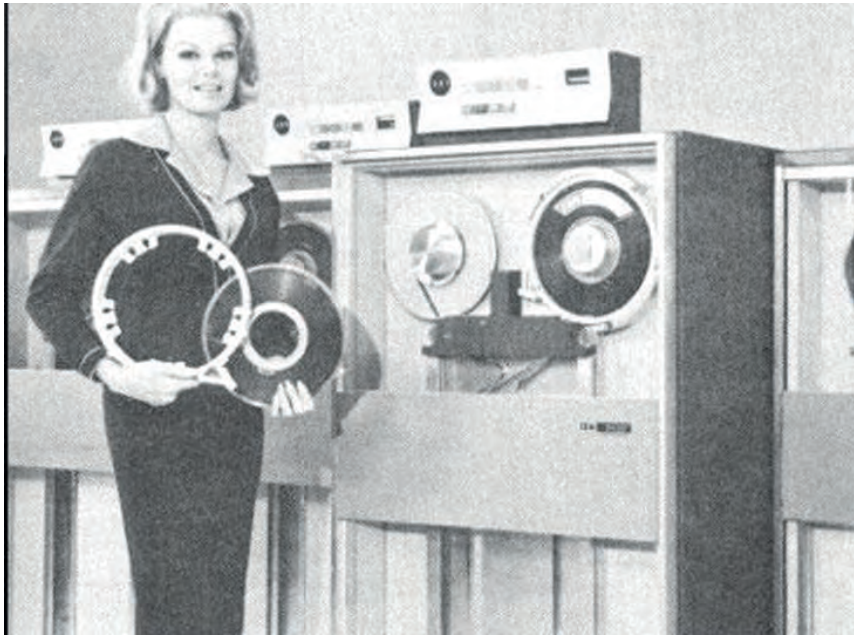


Figura 8.4: UNICAC I - FOTO: IBM

Grabación de los datos en cinta magnética

La información se escribe (o almacena) en la cinta de forma secuencial por medio de un dispositivo denominado controlador de cinta, el cual también permite la lectura de los datos allí almacenados. Para la lectura o escritura se hace uso de un elemento denominado cabezal que, en el caso de la escritura, a partir de la corriente que circula sobre la bobina que contiene, genera un campo magnético que actúa sobre las partículas metálicas de la cinta (que se comportan como pequeños imanes) haciendo que se orienten. En el caso de la lectura estas partículas metálicas orientadas inducen una corriente en el cabezal que luego es analizada para ver si se trata de un bit (0 o 1) o no.

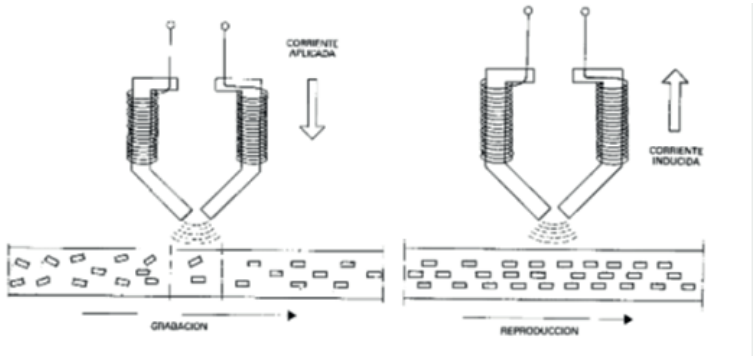


Figura 8.5: Grabación en cinta

La cinta es un material sintético recubierto por una capa que puede ser de óxido de hierro, óxido de cromo o partículas de metal. La cinta se divide a su vez en nueve "subcintas" denominadas pistas. Cada pista está asociada a un cabezal, por tanto, el controlador de cinta dispone de nueve cabezales agrupados. Ocho segmentos de pista alineados verticalmente representan un carácter, así, cada segmento de cinta puede considerarse como un bit, donde el último bit, el segmento de pista nueve, es un bit de paridad. Un grupo de bytes 1 se denomina registro. Entre registro y registro se deja un espacio denominado IRG (inter record gaps) para que sean independientes. Cada registro cuenta con una región en la que se dispone de información que permite identificarlo.

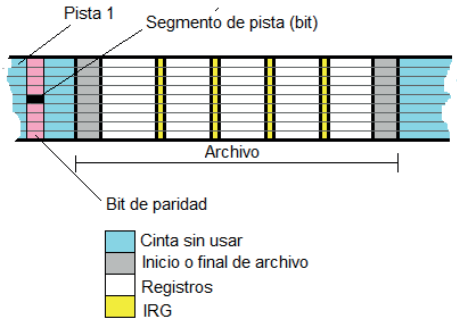


Figura 8.6: Regiones cinta

Un grupo de registros e IRG se denominan bloque; así, un archivo puede considerarse como un grupo de bloques. De forma similar a los registros, un archivo cuenta con una región inicial que tiene información que lo identifica, y otra final, que indica el final del archivo. Las cintas son medios de almacenamiento lentos, además de incómodos, a la hora de actualizar un archivo o realizar una lectura, ya que debe volverse a grabar todo el disco o recorrer toda la cinta hasta llegar al archivo deseado, respectivamente. El tiempo de retención de los datos depende del uso que se le dé, o de agentes como campos magnéticos externos y el calor. El número de bytes que se puede almacenar en una pulgada de cinta se llama densidad de grabación. Esta se mide en cpi (caracteres por pulgada) o bpi (bits por pulgada) vistos a lo ancho de la cinta. Las densidades típicas de grabación en las cintas de carrete son de 800, 1600, 6250 bpi.

Casete y cartuchos de cinta

En los años 60, los fabricantes de computadoras comenzaron a colocar carretes de cintas magnéticas en miniatura dentro de cartuchos de plástico. Estos cartuchos eran más durables, portables y convenientes que los carretes abiertos de cintas magnéticas, y su popularidad como medio de backup para los cada vez mayores discos duros se incrementó en los años 70 y 80. Al igual que los primeros sistemas de carretes abiertos, la capacidad de los sistemas de cartuchos de cinta tenía la ventaja de la flexibilidad. Cuando las necesidades de almacenamiento crecían, los fabricantes de cintas simplemente crearon cartuchos que tenían más cinta. Más cinta significaba más espacio.

Hoy, los cartuchos de cinta como el LTO Ultrium (abajo a la izquierda) de 800 GB siguen en uso para el backup de servidores de gran tamaño, aunque su popularidad ha disminuido en la pasada década, debido a que las transferencias de hard drive a hard drive han ganado aceptación.

Los casetes de cinta intercambiable usados en los grabadores domésticos son un medio barato y prácticamente en desuso, para almacenar datos. No son de calidad para el uso científico o de negocios. Hay cartuchos de cinta que contienen largas cintas magnéticas. Estos cartuchos brindan una forma más conveniente de empacar la cinta y simplifican el montaje de los carretes de cinta. Permiten protección para la suciedad, dado que la cinta está sellada en el cartucho. Los cartuchos son un medio de almacenamiento en cinta de alto desempeño.

Figura 8.7: Casetes y cartuchos de cinta

Unidades de tape backup

El soporte físico empleado es parecido a un casete, pero en dimensiones mayores. Las unidades de lectura-escritura son del tamaño de una disquetera.

Dentro de un cartucho de cinta hay una tira delgada plástica con superficie magnética, similar a la encontrada en cintas para audio y cámaras de video. Cuando se inserta el cartucho en la unidad, este se mueve a través de cabezas de lectura-escritura que leen y registran datos.

Su principal función es la referente a la realización de copias de seguridad y/o almacenamiento masivo de archivos. Para usar archivos respaldados en cinta se deberá, primeramente recuperar los archivos deseados y guardarlos en el disco rígido del sistema. La mayor ventaja de estos sistemas es su bajo costo relativo por MB (megabytes) respaldado, así como también su alta capacidad de respaldo. Las cintas magnéticas son un medio ideal para archivar datos que no es necesario mantener permanentemente en línea (datos "históricos" que puedan ser necesario recuperar en algún momento), dado que son un medio compacto que ocupa relativamente poco espacio.

La selección del tipo de cinta se determinará por los requerimientos de capacidad y velocidad de transferencia de los distintos dispositivos. Así, para backup de servidores lo común serán los dispositivos DAT (para bajos

requerimientos) y DLT (para altos requerimientos), y para los equipos PC tendremos dispositivos QIC y DAT.

Acceso directo

Un dispositivo de almacenamiento de acceso directo (*direct access storage device* o DASD, en inglés) es cualquier tipo de dispositivo de almacenamiento secundario que tiene un tiempo de acceso bajo en proporción a su capacidad. Las unidades de acceso directo en general, son más caras que las de acceso secuencial porque los circuitos electrónicos requeridos para el movimiento de las cabezas lectoras/grabadoras es complejo y de gran precisión. IBM introdujo el primer *hard drive* con discos removibles en la IBM 1311 en 1963. Esta usaba paquetes de discos intercambiables, cada uno de los cuales se componía de seis discos de 14 pulgadas de diámetro. Cada paquete de discos almacenaba alrededor de 2 MB de datos. Muchos *hard drives* de los años 70, como el DEC RK05, tenían paquetes de discos acomodados que las compañías de minicomputadoras generalmente usaban para la distribución de *software*.

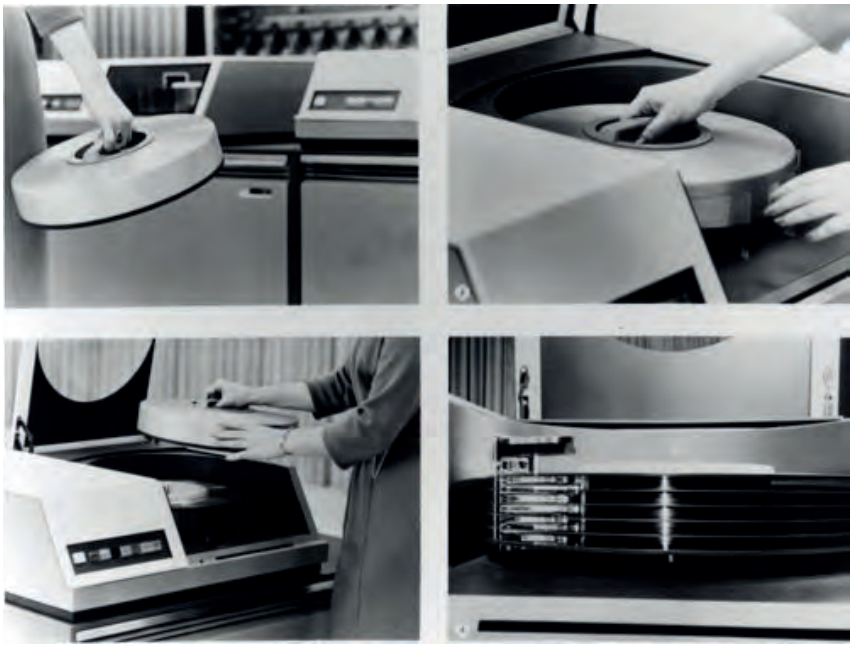


Figura 8.8: DEC RK05

La funcionalidad de acceso directo, ahora llamada acceso aleatorio, de esos dispositivos era el opuesto al acceso secuencial usado en cintas magnéticas, mucho más lento al acceder a un punto distante en el dispositivo. Tanto los tambores como las células han desaparecido como productos en sí, de modo que DASD es ahora un sinónimo de dispositivo de disco. Los discos modernos usados en computadoras centrales raramente son discos individuales, son más bien grandes conjuntos de discos usando esquemas RAID. Los discos magnéticos resultan un medio conveniente para tener acceso a grandes cantidades de datos. Son útiles como soportes de información en aplicaciones que van desde bases de datos a utilización del sistema por varios usuarios en tiempo compartido y memoria virtual. (Prolongación de la memoria principal utilizando el disco).

Actualmente hay varios tipos de discos, son el principal exponente de esta forma de grabar/leer información. Se presentan en los siguientes tipos: discos rígidos fijos, discos rígidos removibles y discos flexibles. Cada uno está diseñado para trabajar con determinado tipo de unidad de arrastre. No obstante, todos trabajan de forma similar.

Disco flexible floppy

El disco flexible nació en IBM, a inicios de la década del 70. Ha sufrido una serie de evoluciones tanto en dimensión como en capacidad; de 8 pulgadas a 3 ½ pulgadas y de 100 kB a 1,44 MB. El soporte magnético está constituido por un material magnético depositado sobre un soporte circular de plástico llamado Mylar. El material magnético puede cubrir una cara del soporte o las dos. Este conjunto se introduce en una funda cuadrada de cartón y se cierra, tiene dos orificios pasantes, de los cuales, cada uno, tiene una función específica. El central permite el arrastre del disco a la unidad mecánica. Cuando se ha introducido en la boca de la unidad el disco gira a una velocidad de más de 300 vueltas por minuto.

Figura 8.9: Disquete

Existe además una hendidura alargada que permite a las cabezas de lectura y grabación acceder a la superficie del soporte magnético. Para posibilitar el control de la velocidad del disco flexible y el inicio de cada pista de grabación, existe un pequeño orificio que, mediante una célula fotoeléctrica detecta el inicio de la pista. Todos los disquetes tienen el orificio de índice. En cada revolución, el agujero deja pasar la luz que emite un diodo luminoso (LED). Así se genera una señal eléctrica que indica el comienzo de la pista.

Componentes de la unidad de grabación de un disco flexible:

- Bloque lógico: lo integran los circuitos de control y los circuitos de lectura/escritura.
- Bloque mecánico: motores de desplazamiento, los sensores, cabezas de lectura y el motor de arrastre del disco flexible.

Disco rígido

En informática, la unidad de disco duro o unidad de disco rígido (en inglés, *hard disk drive*, HDD) es el dispositivo de almacenamiento de datos que emplea un sistema de grabación magnética para almacenar archivos digitales. Se compone de uno o más platos o discos rígidos, unidos por un mismo eje que gira a gran velocidad dentro de una caja metálica sellada. Sobre cada plato, y en cada una de sus caras, se sitúa un cabezal de lectura/escritura que flota sobre una delgada lámina de aire generada por la rotación de los discos. Es memoria no volátil.

Figura 8.10: HD

Estructura lógica

Dentro del disco se encuentra la siguiente estructura:

- El registro de arranque principal (*Master Boot Record*, MBR), en el bloque o sector de arranque, que contiene la tabla de particiones.
- Las particiones de disco, necesarias para poder colocar los sistemas de archivos.

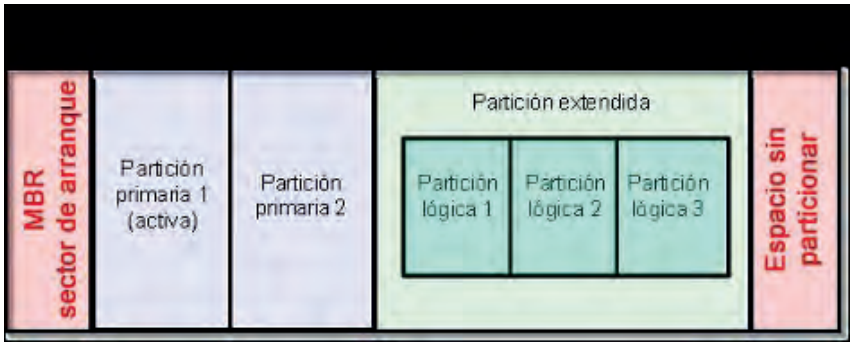


Figura 8.11: HD Estructura lógica

Estructura física

Dentro de la unidad de disco duro hay uno o varios discos (de aluminio o cristal) concéntricos llamados platos (normalmente entre 2 y 4, aunque pueden ser hasta 6 o 7 según el modelo), que giran todos a la vez sobre el mismo eje, al que están unidos. El cabezal (dispositivo de lectura y escritura) está formado por un conjunto de brazos paralelos a los platos, alineados verticalmente y que también se desplazan de forma simultánea, en cuya punta están las cabezas de lectura/escritura. Por norma general hay una cabeza de lectura/escritura para cada superficie de cada plato. Los cabezales pueden moverse hacia el interior o el exterior de los platos, lo cual combinado con su rotación, permite que los cabezales puedan alcanzar cualquier posición de la superficie de los platos.

Cada plato posee dos "ojos", y es necesaria una cabeza de lectura/escritura para cada cara. Si se observa el esquema cilindro-cabeza-sector, a primera vista se ven cuatro brazos, uno para cada plato. En realidad, cada uno de los brazos es doble, y contiene dos cabezas: una para leer la cara superior del plato, y otra para leer la cara inferior. Por tanto, hay ocho cabezas para

leer cuatro platos, aunque por cuestiones comerciales, no siempre se usan todas las caras de los discos y existen discos duros con un número impar de cabezas, o con cabezas deshabilitadas. Los cabezales de lectura/escritura no tocan el disco, sino que pasan muy cerca (hasta a tres nanómetros), debido a una finísima película de aire que se forma entre los cabezales y los platos cuando los discos giran (algunos discos incluyen un sistema que impide que los cabezales pasen por encima de los platos hasta que alcancen una velocidad de giro que garantice la formación de esta película). Si alguna de las cabezas llega a tocar una superficie de un plato, causaría muchos daños en él, rayándolo gravemente, debido a lo rápido que giran los platos (uno de 7200 revoluciones por minuto se mueve a 129 km/h en el borde de un disco de 3,5 pulgadas).

Direccionamiento

Hay varios conceptos para referirse a las zonas del disco:

- Plato: cada uno de los discos que hay dentro de la unidad de disco duro.
- Cara: cada uno de los dos lados de un plato.
- Cabezal: número de cabeza o cabezal por cada cara.
- Pista: una circunferencia dentro de una cara; la pista cero (0) está en el borde exterior.
- Cilindro: conjunto de varias pistas, son todas las circunferencias que están alineadas verticalmente (una de cada cara).
- Sector: cada una de las divisiones de una pista. El tamaño del sector no es fijo, siendo el estándar actual 512 bytes.
- Sector geométrico: son los sectores contiguos pero de pistas diferentes.
- Clúster: es un conjunto contiguo de sectores.

El primer sistema de direccionamiento que se usó fue el cilindro-cabezal-sector (*cylinder-head-sector*, CHS), ya que con estos tres valores se puede situar un dato cualquiera del disco. Más adelante se creó otro sistema más sencillo, que actualmente se usa: direccionamiento de bloques lógicos (*logical block addressing*, LBA), que consiste en dividir el disco entero en sectores y asignar a cada uno un único número.

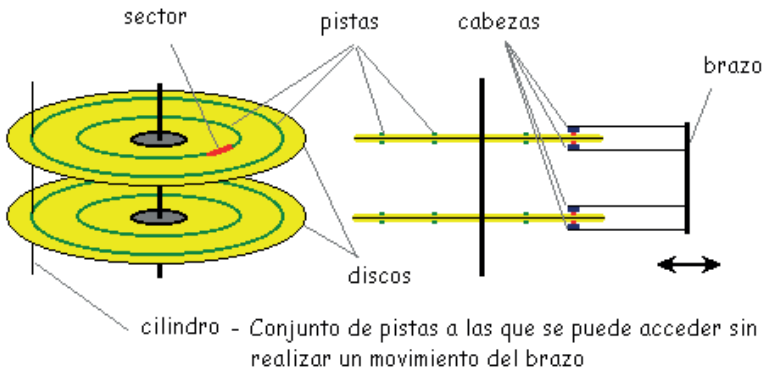


Figura 8.12: HD Estructura

Características de un disco rígido

Las características que se deben tener en cuenta en un disco duro son las que se indican a continuación:

- Tiempo medio de acceso: tiempo medio que tarda la aguja en situarse en la pista y el sector deseado; es la suma del Tiempo medio de búsqueda (situarse en la pista), Tiempo de lectura/escritura y la Latencia media (situarse en el sector).
- Tiempo medio de búsqueda: tiempo medio que tarda la aguja en situarse en la pista deseada; es la mitad del tiempo empleado por la aguja en ir desde la pista más periférica hasta la central del disco.
- Tiempo de lectura/escritura: tiempo medio que tarda el disco en leer o escribir nueva información: depende de la cantidad de información que se quiere leer o escribir, el tamaño de bloque, el número de cabezales, el tiempo por vuelta y la cantidad de sectores por pista.
- Latencia media: tiempo medio que tarda la aguja en situarse en el sector deseado; es la mitad del tiempo empleado en una rotación completa del disco.
- Velocidad de rotación: es la velocidad a la que gira el disco duro, más exactamente, la velocidad a la que giran el/los platos del disco, que es donde se almacenan magnéticamente los datos. La regla es que a mayor velocidad de rotación, más alta será la transferencia de datos,

pero también mayor será el ruido y mayor será el calor generado por el disco duro. Se mide en número revoluciones por minuto (r. p. m.). No debe comprarse un disco duro IDE de menos de 5400 r. p. m. (ya hay discos IDE de 7200 r. p. m.), a menos que esté a un muy buen precio, ni un disco SCSI de menos de 7200 r. p. m. (los hay de 10.000 r. p. m.). Una velocidad de 5400 r. p. m. permitirá una transferencia entre 10 Mb y 16 Mb por segundo con los datos que están en la parte exterior del cilindro o plato, algo menos en el interior de los platos. A mayor velocidad de rotación, menor latencia media.

- Tasa de transferencia: velocidad a la que puede transferir la información a la computadora una vez que la aguja está situada en la pista y sector correctos. Puede ser velocidad sostenida o de pico.

Otras características son las siguientes:

- Caché de pista: es una memoria tipo flash dentro del disco duro.
- Interfaz: medio de comunicación entre el disco duro y la computadora. Pueden ser IDE/ATA, SCSI, SATA, USB, Firewire, Serial Attached SCSI.

Tecnología de soporte óptico

Discos ópticos

Los discos ópticos presentan una capa interna protegida, donde se guardan los bits mediante distintas tecnologías los que se leen merced a un rayo láser incidente. Este, al ser reflejado, permite detectar variaciones microscópicas de propiedades óptico-reflectivas ocurridas como consecuencia de la grabación realizada en la escritura. Un sistema óptico con lentes encamina el haz luminoso, y lo enfoca como un punto en la capa del disco que almacena los datos.

Las tecnologías de grabación (escritura) a desarrollar son las que se detallan a continuación:

- Por moldeado durante la fabricación, mediante un molde de níquel (CD-ROM y DVD ROM).
- Por la acción de un haz láser (CD-R y CD-RW, también llamado CD-E).

- Por la acción de un haz láser en conjunción con un campo magnético (discos magneto-ópticos - MO).

En el almacenamiento de datos sobre medios giratorios se diferencian dos procesos: el CAV (*constant angular velocity*) y el CLV (*constant linear velocity*). En el primero de ellos (CAV, usado en discos duros y disquetes) el medio gira a una velocidad angular constante (siempre el mismo número de vueltas por unidad de tiempo) bajo la cabeza lectora independientemente de que esta esté en la parte más central del disco o en la más alejada. Las pistas sobre las que leen son circulares (y se dividen en sectores radiales), así, las pistas interiores son más cortas que las exteriores.

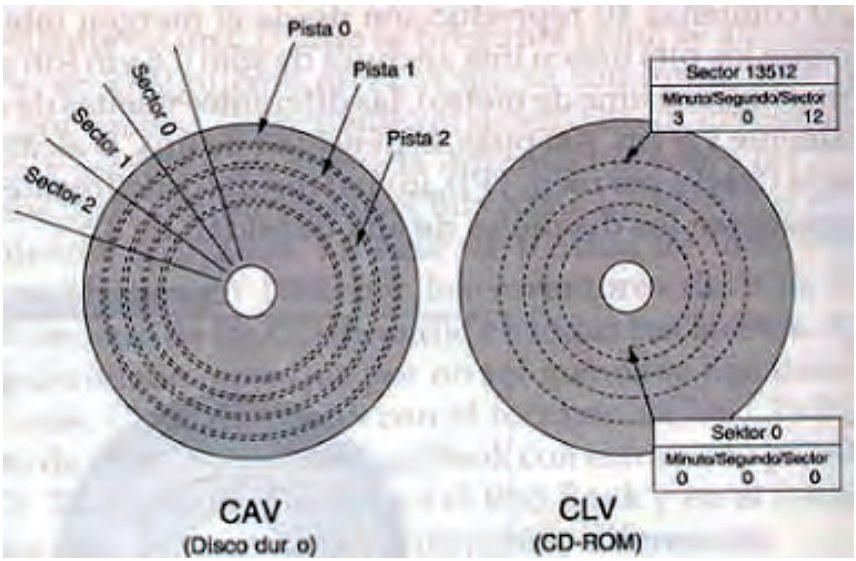


Figura 8.13: CAV

En el procedimiento CLV (usado en los CD) el movimiento de rotación es variable dependiendo de la posición de la cabeza lectora con respecto del centro del disco. En este caso la pista es única en espiral y la cabeza recorre la misma longitud de pista por unidad de tiempo. Así, cuando la cabeza lectora se encuentra más cerca del centro el disco gira más rápido que cuando su posición es más excéntrica, esto ocasiona que continuamente se tenga que estar

regulando la velocidad de rotación respecto a la posición de la cabeza. Esta es una de las razones por las que las unidades CD-ROM, en comparación con los discos duros, presentan velocidades de acceso mucho menores ya que las aceleraciones y deceleraciones consumen un tiempo innecesario para la lectura de datos. Además de que es mucho más difícil encontrar un sector a lo largo de una espiral que en un medio limpiamente dividido en pistas y sectores. La superficie grabable de un CD se divide en tres partes: El *lead-in*, la zona de datos y el *lead-out*. El *lead-in* (o encabezamiento) ocupa los cuatro primeros milímetros del disco desde el margen interior. A continuación le sigue la zona de datos que, dependiendo del nivel de ocupación, ocupa más o menos, hasta un máximo de 33 milímetros. Por último, la parte final la constituye la zona *lead-out* que es como una marca de terminación. Se encuentra inmediatamente después de la zona de datos y tiene una anchura de 1 milímetro. El *lead-in* contiene la información acerca de cómo se disponen los archivos en el disco.

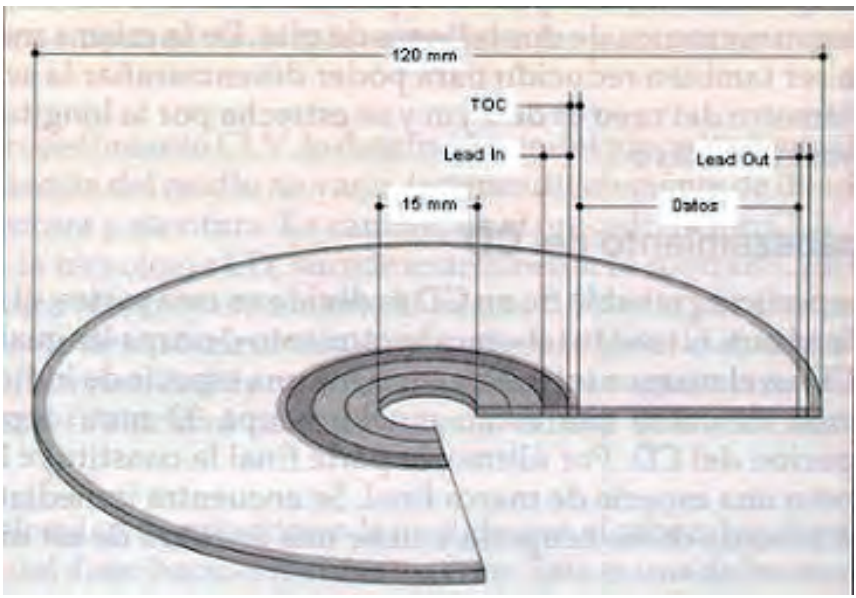


Figura 8.14: Estructura CD

Ventajas y desventajas

- La comparación del lector óptico con los lectores de soportes magnéticos estriba en que aquí no existe contacto físico entre cabeza y disco, por tanto, no hay rozamiento. En consecuencia, tendremos menor desgaste, superior duración y mayor seguridad en los datos. La cabeza móvil —que porta la fuente láser y la óptica asociada— por estar separada a 1mm de la superficie del disco, nunca puede tocarla, por lo que no produce desgaste por rozamiento ni existe riesgo de aterrizaje, como en el disco rígido con cabezas flotantes.

- Los discos ópticos, además de ser portables, son más seguros en la conservación de los datos, dado que la capa que los almacena es inmune a los campos magnéticos caseros y está protegida de la corrosión ambiental, etcétera.

- Puede estimarse entre 10 y 15 años la permanencia de la información en un CD ROM común, dado que la superficie de aluminio que contiene la información se oxida muy lentamente en ese lapso.

- La mayor capacidad de los discos ópticos frente a los magnéticos se debe al carácter puntual del haz láser incidente y a la precisión del enfoque óptico del láser. Ello permite, por un lado, que en una pista los bits estén más juntos (mayor densidad lineal); y por otro, que las pistas estén más próximas. En los medios magnéticos, donde el tamaño de las cabezas lecto-grabadoras es más grande, impide que las pistas sean más pequeñas.

- La óptica requerida para los discos borrables es extremadamente complicada y todavía muy cara.

- Para acceso random real, los discos ópticos tienen una performance relativamente pobre y más aún los borrables. Como es sabido, los soportes magnéticos están organizados por pistas y sectores, por lo que el acceso a la información es muy rápido, ya que la cabeza lectora-grabadora se desplaza inmediatamente a la dirección de pista/sector que se requiera, y ello ocurre a una velocidad de giro constante. Sin embargo, en los discos ópticos, por ser un producto procedente del disco de sonido, el acceso es

longitudinal y la búsqueda se efectúa a lo largo de toda la grabación en espiral —análogamente a lo que sucedía en los tradicionales discos LP de vinilo—, por lo que la recuperación es más lenta, ya que la cabeza lectora debe recorrer todo el camino hasta encontrar el bloque solicitado y, como también el disco gira a velocidad lineal constante, el tiempo resultante de acceso es considerablemente superior al de un disco duro convencional, lo que constituye un apreciable inconveniente.

CD-ROM

Ante todo digamos que el CD-ROM procede, según sus propias iniciales originales, de la expresión *compact disk-read only memory* (disco compacto de memoria solamente de lectura) y, lo que en términos habituales denominamos ROM, es equivalente a almacenamiento de datos permanente no modificable.

Cuando los discos de vinilo y los casetes (sistema de almacenamiento de datos analógico) desgastados por el uso empezaban a hacer demasiado ruido, Sony y Philips, a los mediados de los 80, planearon la posibilidad de crear un sistema de almacenamiento digital de datos (en principio ideado para el almacenamiento de música), fácilmente transportable. Requerían, para ello, un soporte que permitiera guardar en forma de bits a 44,1 kHz con una resolución de 16 bits, y por supuesto, en estéreo (es decir, información independiente para dos canales, derecho e izquierdo). Si traducimos esto a unidades de almacenamiento de datos supondría 44100*60*16 = 1.411.200 bits por segundo. En definitiva, con los sistemas de almacenamiento digital de entonces habrían necesitado 1 disco duro de 10 MB para guardar un minuto de música. Otra condición que tenía que cumplir el soporte a inventar era que la velocidad de transmisión de datos del dispositivo de lectura fuera exactamente 1.411.200 bits por segundo. Es entonces cuando crean el CD.

Se trata, por tanto, de un disco compacto de gran almacenamiento, regrabable y/o no regrabable, hoy en día, que se conecta al computador como un periférico más. Apareció a mediados de los 80 y fue el primer formato de almacenamiento óptico. La capacidad de un CD-ROM es de 600 MB, equivalente a 100.000 páginas tipo A4, lo cual lo hace ideal para almacenar grandes bases de datos.

Este medio de almacenamiento tiene la desventaja de no poder reescribir en ellos, lo cual si bien no permite su uso para guardar datos propios, es ideal

para distribuir *software*. Todos los CD-ROMs existentes en el mercado comparten el mismo formato físico, pero manejan diversos estándares de archivos y directorios. La principal característica de la unidad de CD-ROM es el almacenamiento masivo de datos pues en un solo CD se puede tener la información equivalente a la de un disco de 600 MB.

Características técnicas

Un CD, cualquiera sea su formato, tiene un diámetro de 12 cm (a excepción de los *electronic book CD* que tienen 8 cm). El agujero interior es de 15 mm. Tiene una capa metálica reflectante recubierta de un barniz transparente. Desde el centro del CD parte una espiral (cuya longitud total es de aproximadamente 6 km), de 0,6 micras de grosor, que es la que contiene los datos en forma de raya discontinua (protuberancias y cavidades llamados pits y lands). Cada vuelta de la espiral está separada 1,6 micras de la vuelta siguiente. Así, la densidad de un CD alcanza las 16.000 pistas o, mejor dicho, vueltas por pulgada (TPI = tracks per inch), difícilmente comparable con las 135 TPI que presentan los disquetes de 3,5 pulgadas de alta densidad. La cabeza lectora está formada por un emisor de luz (cuyo haz tiene un diámetro de 1 micra) y un receptor, de manera que los cambios entre pits y lands producen diferencias de luz.

Figura 8.15: Espiral

Llevando esta terminología de uso común señalaremos que su capacidad oscila, según las marcas de los distintos fabricantes, entre 500 y 600 MB, que es aproximadamente la misma que 400 disquetes de 31/2 pulgadas (de 1,4 MB), mientras que su equivalente en disco duro sería de 27 discos de 20 MB, aproximadamente. Comparando estas cifras con el contenido de los textos de un libro diremos que en un CD-ROM se pueden almacenar unas 250.000 páginas.

Tecnología CD-ROM

Los datos son grabados mediante un haz de láser. Cada disco es un plato de 120 mm de diámetro que, en su superficie, tiene depresiones (pits) y zonas planas (lands), que se combinan para determinar la presencia de ceros y unos (indican el cambio de cero a uno o de uno a cero). Estos dos accidentes presentan diferentes capacidades ópticas, muy fáciles de detectar por la cabeza lectora. Ancho de pistas: 0,6 micrómetros. Densidad: 15.875 pistas por pulgada. Capacidad: 600 MB. Los datos deben ser puestos en un master mediante un proceso especial. A partir de ese master se reproducen los discos. Los tres pasos para hacer un CD-ROM son los siguientes: conversión de datos, premastering y mastering. En la etapa de conversión de datos, estos se organizan y se formatean. El premastering agrega los códigos de error y los índices necesarios para hacer eficiente el uso del CD-ROM. El master, usado para estampar los pits en los discos de producción, es creado en la etapa de mastering.

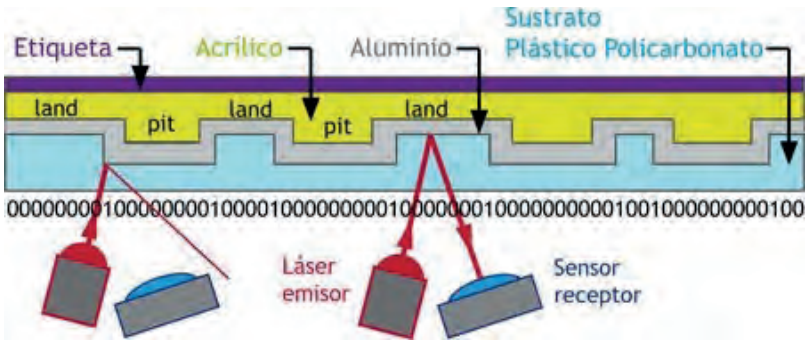


Figura 8.16: Estructura CD

CD-R

Un CD-R (*CD recordable*, o sea grabable) puede grabarse por cualquier usuario que tenga conectado en su computadora el periférico unidad grabadora de CD. En esta, un haz láser graba en una espiral parcialmente pregrabada de fábrica, construida en una capa de material orgánico. Dicha espiral ya viene formateada por *hardware* con las direcciones de los sectores y sirve de guía para el láser. Sobre dicha capa orgánica con la espiral, que es translúcida, el CD-R presenta otra capa de oro para reflejar el haz láser en cada lectura. Estas dos capas están protegidas por otras de policarbonato. La capa orgánica translúcida es de resina o pigmento verde (generalmente cianina). Después de ser grabado, un CD-R se convierte, de hecho, en un CD-ROM, que puede leerse en cualquier unidad lectora de estos discos —de la forma antes descripta— sin posibilidad de ser regrabado. No es necesario grabar toda la espiral de un CD-R de una sola vez (sesión). Es factible hacerlo en tantas "sesiones" como archivos se quieran incorporar a lo largo del tiempo, hasta completar la capacidad del CD-R (como ser, 650 MB). Una vez grabada una porción de la espiral, no puede borrarse y ser regrabada. Por tal motivo, los CD-R también se denominan CD-WO (*write once*, o sea, de una escritura). Esta imposibilidad de regrabación ha motivado su uso en el ámbito contable y financiero, pues garantiza datos no borrables para auditorias. Por lo general, los CD-R se reconocen a primera vista por el color dorado de su etiqueta. Los primeros 4 mm de ancho radial de una espiral de un CD-R o de un CD-ROM constituyen el "lead in" que antecede a la zona de datos. Esta es de unos 29 mm de ancho, y le sigue el "lead out" de 1 mm. En un CD-R, el "lead-in" es precedido por dos áreas necesarias para alinear el haz láser a fin de poder grabar lo que sigue. Cada sesión de grabado de la espiral debe comenzar con la escritura de un "lead in", y terminar con la de un "lead out". A su vez, cada "lead in" debe contener la tabla de contenidos ("*tabla of contents*" TOC), índice de los datos grabados en la sesión correspondiente. Debe mencionarse que un CD-R grabado en "multisesiones" debe ser leído por un lector de CD-ROM apropiado, como los actuales, de no ser así, solo leerá la primera sesión. Existen grabadoras/lectoras de CD-R de varias velocidades (x1, x2, x4...). A mayor velocidad debe usarse un láser más potente para producir más calor, de forma tal que permita atacar adecuadamente los puntos requeridos en la espiral. Existen discos vírgenes CD-R para distintas velocidades, cuyo sustrato disipa distinta cantidad de calor en

correspondencia con su velocidad de grabación. Debido a su capa orgánica, los CD-R no deben ser expuestos a excesivo calor (por ejemplo, dentro de un automóvil o al sol directo) o humedad, pues pueden reducir su vida útil, o ser inutilizables por filtraciones de cianina. También se debe cuidar de no escribir con bolígrafo su etiqueta, dado que la presión ejercida puede dañarlos. Una unidad CD-R puede leer un CD-ROM, y viceversa.

CD-RW

CD-RW son las siglas de *CD Rewritable*, o sea, CD re-escribible, asociado a la tecnología de regrabación por cambio de fase. También se denominan CD-E (*CD-Erasable*), o sea, CD borrable. Esta tecnología se basa en la propiedad que posee una capa de material como el teluro (mezclado con germanio o antimonio), de cambiar del estado amorfo (0) al cristalino (1) si se alcanza la "temperatura transición" (100 ºC o más); y de volver de cristalino a amorfo si se alcanza la temperatura de fusión y se deja enfriar. Resulta fácil distinguir un CD grabable, ya que al contrario de sus primos, los CD-ROM (solo lectura), su brillo no es plateado, sino dorado. En esta capa, en lugar de los pits y lands del CD-ROM, se encuentra una sustancia de color cuya propiedad de reflexión se determina vía láser, simulando los pits y lands, consiguiendo que las unidades normales de CD-ROM sean capaces de leer estos CD. Para escribir un uno en un punto de una pista del disco, un láser con baja potencia lo calienta rápidamente hasta la temperatura de transición. Si el estado físico del punto era amorfo, pasa a cristalino; y si ya está en este estado, quedará igual. Un cero se escribe calentando el punto hasta la temperatura de fusión, usando el láser con alta potencia. Al enfriarse pasa al estado amorfo, y si estaba en ese estado volverá al mismo. La lectura de las pistas así grabadas se realiza con el mismo cabezal, recorriéndoles con el láser de potencia diez veces menor. Al ser censada la luz láser reflejada permite detectar, por diferencias de reflectividad, los cambios de un estado físico al otro a lo largo de la pista. Un punto en estado cristalino refleja el 70 por ciento de la luz incidente, y en estado amorfo el 18 por ciento. Obsérvese que esta tecnología es puramente óptica, sin magnetismo, requiriéndose una sola pasada para escribir. Para escribir o leer este tipo de discos se requiere de grabadoras y lectoras apropiadas para su tecnología. Se estimaba hace poco que un CD-E puede regrabarse unas 100.000 veces. Realizando 50 reescrituras diarias, duraría 5 años (de 365 días). Hubo

avances al respecto. Las unidades CD-RW también pueden leer los CD-ROM y CD-R, siendo además que estos CD cumplen con el formato UDF (*universal disc format*) normalizado por la Asociación Tecnológica de Almacenamiento Óptico que facilita a los sistemas operativos el acceso a discos.

DVD

Los DVD-ROM (*digital versatil disk*) de "simple capa" tienen igual tamaño que un CD-ROM de 680 MB, y se basan en la misma tecnología de grabación y lectura que estos, pero pueden almacenar 4,7 GB de datos (7 veces más), video o audio. Típicamente pueden transferir unos 0,6 MB/seg (como un CDx4) para entretenimientos, y 1,3 MB/seg para computación (como un CDx10). Esto se ha logrado disminuyendo a la mitad la longitud de los pits en relación a un CD-ROM y llevando al doble el número de vueltas por pulgada radial de la espiral. El DVD estándar que se comercializa en el mercado es fruto del acuerdo entre Philips, Sony (creadores del "Multimedia CD", MMCD), y Toshiba (que con otros grupos desarrolló el *Super Density,* SD). Este DVD puede almacenar 2 horas de video de calidad, con títulos y sonido. Asimismo, los 4,7 GB permiten guardar 135 minutos de filmes (duración típica de una película de cine) en reemplazo de una cinta de video. Esto es así, dado que con compresión MPEG2 se requiere, para transferir imagen, sonido y títulos, cerca de 0,5 Mb/seg. Si calculamos: 135 min x 60 seg/min x 0,5 Mb/s, resulta un valor cercano a 4,7 GB. Los DVD-ROM de "doble capa" presentan una capa semitransparente reflectiva con oro (que puede guardar 3,8 GB), la cual se encuentra debajo de la capa reflectora (4,7 GB) metalizada con plata. Sumando ambas capacidades resulta un total 8,5 GB. Para leer la capa semitransparente, el haz láser es enfocado en ella con baja potencia, mientras que la lectura de la capa reflectiva se realiza enfocando en esta el haz, ahora con mayor potencia, para que atraviese la capa semitransparente al incidir, y cuando se refleja. También se están fabricando DVD-ROM de "simple capa" y "doble cara", para ser leídos en ambas caras, con lo cual se logra 4,7 GB x 2 = 9,4 GB; y DVD-ROM de "doble capa" y "doble cara", de 8,5 x 2 = 17 GB. Estos CD están muy expuestos a las rayaduras, por ser más finas las capas protectoras transparentes. Un DVD-RW es análogo a un CD-RW re-escribible antes descripto, pero tiene mayor capacidad, merced al empleo de un láser de menor longitud de onda que los usados. Debido a las limitaciones de fabricación masiva de láseres azules de potencia de corta longitud de onda, la capacidad de los DVD-RAM

es de 2,6 GB frente a los 4,7 GB de los DVD-ROM. Potencialmente, los DVD-RW pueden ser competidores de las cintas magnéticas para "backups" si el costo por byte almacenado lo justifica.

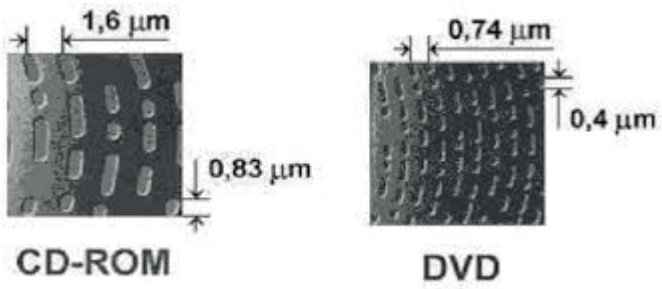


Figura 8.17: CD vs. DVD pits

BLU-RAY

El disco Blu-ray, también conocido como BD (en inglés, *Blu-ray disc*), es un formato de disco óptico de nueva generación desarrollado por la *Blu-ray Disc Association* (BDA), que se emplea para video de alta definición (HD), 3D y UltraHD, con mayor capacidad de almacenamiento de datos de alta densidad que la del DVD. El disco Blu-ray tiene 12 cm de diámetro, igual que el CD y el DVD. Guardaba 25 GB por capa, por lo que Sony y Panasonic han desarrollado un nuevo índice de evaluación (i-MLSE) que permitiría ampliar un 33 por ciento la cantidad de datos almacenados, desde 25 a 33,4 GB por capa. Este disco hace uso de un rayo láser de color azul con una longitud de onda de 405 nanómetros, a diferencia del láser rojo utilizado en lectores de DVD, que tiene una longitud de onda de 650 nanómetros. Esto, junto con otros avances tecnológicos, permite almacenar sustancialmente más información que el DVD en un disco de iguales dimensiones y aspecto externo. Blu-ray obtiene su nombre del color azul del rayo láser (*blue ray* significa 'rayo azul'). La letra 'e' de la palabra original 'blue' fue eliminada debido a que, en algunos países, no se puede registrar para un nombre comercial una palabra común.

Figura 8.18: Medios ópticos

Tecnología memoria en estado sólido

El almacenamiento de estado sólido (*Solid State Storage*, SSS) es un medio de almacenamiento de cómputo hecho a base de microchips de silicio. SSS almacena datos de forma electrónica en lugar de hacerlo magnéticamente, como lo hacen las unidades giratorias del disco duro (HDD) o la cinta de óxido magnético. Una ventaja importante del almacenamiento en estado sólido es que

no contiene partes mecánicas, lo que permite que la transferencia de datos hacia y desde los medios de almacenamiento tenga lugar a una velocidad mucho mayor, brindando una vida más predecible para estos últimos. Debido a que no hay partes móviles, las *Solid State Drive* (SSD) producen mucho menos calor que las HDD. Además de proporcionar tiempos de entrada/salida (E/S) más rápidos y consistentes, los medios de almacenamiento de estado sólido ofrecen los mismos niveles de integridad y resistencia de datos que otros dispositivos electrónicos, y requieren menos energía y refrigeración que sus equivalentes electromecánicos. Generalmente también pesan menos.

Tipos de sistemas de almacenamiento de estado sólido

Hay dos tipos de sistemas de estado sólido: sistemas basados en memoria flash y sistemas basados en RAM. Al principio, se construían con una memoria volátil DRAM y, más adelante, se empezaron a fabricar con una memoria no volátil NAND flash.

Basados en DRAM

Las SSD basadas en este tipo de almacenamiento proporcionan una rauda velocidad de acceso a datos, en torno a 10 ms y se utilizan principalmente para acelerar aplicaciones que de otra manera serían mermadas por la latencia del resto de sistemas. Estas SSD incorporan una batería o bien un adaptador de corriente continua, además de un sistema de copia de seguridad de almacenamiento para desconexiones abruptas que al restablecerse vuelve a volcarse a la memoria no volátil, algo similar al sistema de hibernación de los sistemas operativos. Estas SSD son generalmente equipadas con los mismos módulos de memoria RAM que cualquier ordenador corriente, permitiendo su sustitución o expansión. Sin embargo, las mejoras de las unidades basadas en flash están haciendo las SSD basadas en DRAM no tan efectivas y acortando la brecha que los separa en términos de rendimiento. Además los sistemas basados en DRAM son tremendamente más caros.

Basados en memoria flash NAND

Casi la totalidad de los fabricantes comercializan sus SSD con memorias no volátiles NAND para desarrollar un dispositivo no solo veloz y con una vasta capacidad, sino robusto y a la vez lo más pequeño posible tanto para el mercado de consumo como el profesional. Al ser memorias no volátiles, no

requieren ningún tipo de alimentación constante ni pilas para no perder los datos almacenados, incluso en apagones repentinos, aunque cabe destacar que las SSD NAND son más lentas que las que se basan en DRAM. Son comercializadas con las dimensiones heredadas de los discos duros, es decir, en 3.5 pulgadas, 2.5 pulgadas y 1.8 pulgadas, aunque también ciertas SSD vienen en formato tarjeta de expansión. En algunos casos, las SSD pueden ser más lentas que los discos duros, en especial con controladoras antiguas de gamas bajas, pero dado que los tiempos de acceso de una SSD son inapreciables, al final resultan más rápidos. Este tiempo de acceso tan corto se debe a la ausencia de piezas mecánicas móviles, inherentes a los discos duros. El rendimiento de las SSD se incrementa añadiendo chips NAND en paralelo. Un solo chip NAND es relativamente lento, dado que la interfaz de entrada y salida es de 8 o 16 bits asíncrona y también por la latencia adicional de las operaciones básicas de E/S (típica de los SLC NAND, aproximadamente 25 ms para buscar una página de 4 KB de la matriz en el búfer de E/S en una lectura, aproximadamente 250 ms para una página de 4 KB de la memoria intermedia de E/S a la matriz de la escritura y sobre 2 ms para borrar un bloque de 256 KB). Cuando varias unidades con NAND operan en paralelo dentro de un SSD, las escalas de ancho de banda se incrementan y las latencias de alta se minimizan, siempre y cuando suficientes operaciones estén pendientes y la carga se distribuya uniformemente entre los dispositivos. Las SSD de Micron e Intel fabricaron unidades flash mediante la aplicación de los datos de creación de bandas (similar a RAID 0) e intercalado. Esto permitió la creación de SSD ultrarápidas con 250 Mb/s de lectura y escritura. Las controladoras serie SF 1000 de Sandforce consiguen tasas de transferencia cercanas a la saturación de la interfaz SATA II (rozando los 300 Mb/s simétricos tanto en lectura como en escritura). La generación sucesora, las de la serie SF 2000 de Sandforce, permiten más allá de los 500 Mb/s simétricos de lectura y escritura secuencial, requiriendo de una interfaz SATA III si se desea alcanzar estos registros.

Tendencias de almacenamiento de estado sólido

Aunque la tecnología de almacenamiento de estado sólido no es nueva, el interés radica en cómo la tecnología se puede atribuir a reducciones en el precio y al rendimiento del *hardware*, especialmente en el ahorro de energía. Desde el cambio de siglo, las velocidades del procesador han seguido aumentando drásticamente, mientras que los tiempos de lectura y escritura para discos

duros mecánicos no han aumentado. Las CPU de hoy en día pueden procesar datos mucho más rápido de lo que el almacenamiento HDD puede suministrarlo. El tiempo de demora resultante se conoce como latencia, y los administradores de infraestructuras IT generalmente lidian con la alta latencia de almacenamiento mediante el manejo de las unidades de disco. Se realizan pausas cortas al limitar deliberadamente la capacidad de la unidad de disco, por lo que el actuador de la unidad de disco debe mover las cabezas a través de un número menor de pistas, lo que reduce el tiempo de búsqueda. Los entornos que implementan recorridos cortos normalmente tienen que compensar la capacidad reducida utilizada en cada unidad de disco al aumentar la cantidad de unidades de disco en estas configuraciones. Por el contrario, los dispositivos de almacenamiento de estado sólido no tienen tiempo de búsqueda. Esto reduce considerablemente sus latencias, lo que las hace más rápidas que las HDD, especialmente para operaciones aleatorias de lectura/escritura. Los dispositivos SSS tienen menos ventajas de rendimiento cuando se trata de operaciones secuenciales de lectura/escritura. Al ser inmune a las vibraciones externas es especialmente apto para vehículos, computadoras portátiles, celulares, tabletas, etcétera. En la actualidad, la tecnología de almacenamiento de estado sólido se utiliza para el almacenamiento primario y también como caché frente a los discos giratorios tradicionales, introduciendo una nueva capa entre el procesador y el almacenamiento. Algunos expertos de la industria predicen que el almacenamiento en estado sólido eventualmente reemplazará el almacenamiento en el disco duro si las fábricas de silicio pueden satisfacer la creciente demanda de productos y el precio del SSS continúa disminuyendo.

Memoria flash

La memoria flash permite la lectura y escritura de múltiples posiciones de memoria en la misma operación. Gracias a ello, mediante impulsos eléctricos la tecnología flash permite velocidades de funcionamiento muy superiores frente a la tecnología EEPROM (*Electrically Erasabale Programmable Read Only Memory*) primigenia, que solo permitía actuar sobre una única celda de memoria en cada operación de programación. Se trata de la tecnología empleada en las memorias USB, tarjetas de memorias y unidades de estado sólido. Económicamente hablando, el precio en el mercado cumple la ley de Moore aumentando su capacidad y disminuyendo el precio. Entre sus ventajas se

encuentra una gran resistencia a los golpes, tiempos de accesos más rápidos, bajo consumo de energía y un funcionamiento silencioso, ya que no contiene actuadores mecánicos ni partes móviles comparados con un disco duro convencional. Su pequeño tamaño también es un factor determinante a la hora de escoger para un dispositivo portátil, así como su ligereza y versatilidad para todos los usos hacia los que está orientado. En vista de ello, comienzan a popularizarse las unidades SSD que usan memoria flash en lugar de platos. Sin embargo, todos los tipos de memoria flash solo permiten un número limitado de escrituras y borrados, generalmente entre 10.000 y un millón, dependiendo de la celda, de la precisión del proceso de fabricación y del voltaje necesario para su borrado. Además, su relación costo capacidad es menos favorable respecto a otros medios como los discos ópticos y los discos duros.

Memoria flash de tipo NOR

NOR, así llamada por la tecnología de asignación de datos específicos (en inglés, *Not OR*), es una tecnología de memoria flash de alta velocidad. La memoria flash NOR proporciona capacidades de acceso aleatorio de alta velocidad, pudiendo leer y escribir datos en lugares específicos de la memoria sin tener que acceder a la memoria en modo secuencial. A diferencia de la memoria flash NAND, la memoria flash NOR permite la recuperación de datos desde un solo byte. La memoria flash NOR es excelente en aplicaciones donde los datos se recuperan o se escriben de manera aleatoria. NOR se encuentra más frecuentemente integrada en teléfonos celulares (para almacenar el sistema operativo del teléfono) y asistentes digitales personales; también se utiliza en computadoras para almacenar el programa BIOS que se ejecuta para proporcionar la función de arranque. Este tipo de memoria tiene sus transistores conectados en paralelo. A diferencia de la memoria NAND, la memoria NOR no tiene ninguna tolerancia con las células defectuosas, por lo que cada célula ha de ser siempre perfecta. A diferencia también de la NAND, para borrar las células de memoria NOR hay que borrar la célula entera, en lugar de poder borrar solo un bloque de la información almacenada, como sí permite la memoria NAND. Así como la memoria NAND es rápida a la hora de grabar datos, la memoria NOR es rápida a la hora de leerlos. Estas características hacen que la memoria NOR sea la más empleada en los chips de memoria donde se instala la BIOS de las placas base y de las tarjetas gráficas.

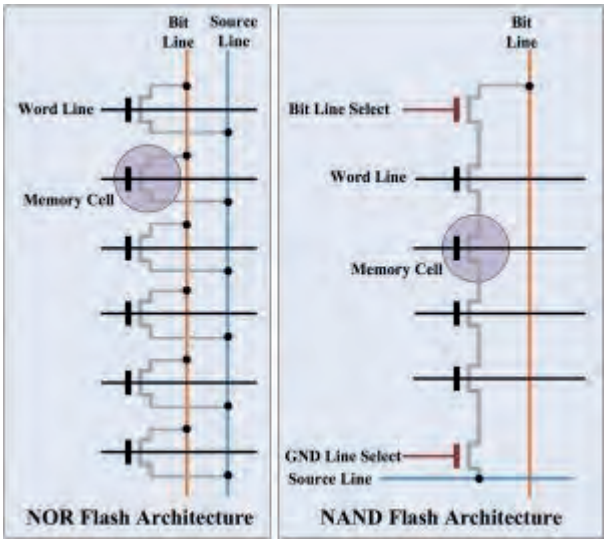


Figura 8.19: Flash NOR y NAND

Memoria flash de tipo NAND

La memoria flash NAND fue inventada después de la memoria flash NOR y tomó su nombre de la tecnología de asignación específica utilizada para datos (en inglés, *Not AND*). La memoria flash NAND lee y escribe a alta velocidad, en modo secuencial, manejando datos en bloques de tamaño pequeño ("páginas"). La memoria flash NAND puede recuperar o escribir datos como páginas únicas, pero no puede recuperar bytes individuales como la memoria flash NOR. En la memoria flash NAND, los transistores se conectan en serie entre ellos, como se puede ver en la figura anterior. Este tipo de memoria accede a las direcciones de memoria de las células en el orden de "página", "palabra" y, finalmente "bit". El hecho de ser un tipo de memoria flash más sencilla de construir, permite aumentar la densidad de transistores por célula de memoria y, por tanto, células de memoria que son capaces de almacenar muchos más datos. Esto también le hace tener una capacidad de grabar datos a una velocidad sensiblemente rápida. También es un tipo de memoria flash más permisiva si una célula de memoria no es completamente perfecta. La

memoria flash NAND se encuentra comúnmente en unidades de disco duro de estado sólido, dispositivos flash de medios digitales de audio y video, decodificadores de televisión, cámaras digitales, teléfonos celulares (para almacenamiento de datos), y otros dispositivos donde los datos se escriben o leen, generalmente de manera secuencial.

SSD

La unidad de estado sólido, SSD (del inglés *solid-state drive*) es un tipo de dispositivo de almacenamiento secundario de datos que utiliza memoria no volátil, como la memoria flash, para almacenar datos, en lugar de los platos o discos magnéticos de las unidades de discos duros (HDD) convencionales. Están hechos con componentes electrónicos en estado sólido pensado para utilizarse en equipos informáticos en sustitución de una unidad de disco duro convencional, como memoria auxiliar o para crear unidades híbridas compuestas por SSD y disco duro. Consta de una memoria no volátil, en vez de los platos giratorios y cabezal de las unidades de disco duro convencionales. Al no tener piezas móviles, una unidad de estado sólido reduce drásticamente el tiempo de búsqueda, latencia y otros, diferenciándose así de los discos duros magnéticos.

Al ser inmune a las vibraciones externas, es especialmente apto para vehículos, computadoras portátiles, teléfonos inteligentes, tabletas, etcétera. En comparación con los discos duros tradicionales, las unidades de estado sólido son menos sensibles a los golpes al no tener partes móviles, son prácticamente inaudibles, y poseen un menor tiempo de acceso y de latencia, lo que se traduce en una mejora del rendimiento exponencial en los tiempos de carga de los sistemas operativos. En contrapartida, su vida útil es muy inferior, ya que tienen un número limitado de ciclos de escritura, pudiendo producirse la pérdida absoluta de los datos de forma inesperada e irrecuperable. Las SSD hacen uso de la misma interfaz SATA que los discos duros, por lo que son fácilmente intercambiables sin tener que recurrir a adaptadores o tarjetas de expansión para compatibilizarlos con el equipo. A partir de 2010, la mayoría de las SSD utilizan memoria flash basada en puertas NAND, que retiene los datos sin alimentación eléctrica. Para aplicaciones que requieren acceso rápido, pero no necesariamente la persistencia de datos después de la pérdida de potencia, las SSD pueden ser construidas a partir de memoria de acceso aleatorio (RAM). Estos dispositivos pueden emplear fuentes de alimentación independientes,

como baterías, para mantener los datos después de la desconexión de la corriente eléctrica. Se han desarrollado dispositivos que combinan ambas tecnologías, discos duros con memorias flash, que se denominan unidades de estado sólido híbridas (SSHD), que intentan aunar capacidad y velocidad a precios inferiores a las SSD.

Una SSD se compone principalmente de los siguientes componentes:

- Controladora: es un procesador electrónico que se encarga de administrar, gestionar y unir los módulos de memoria NAND con los conectores en entrada y salida. Ejecuta *software* a nivel de firmware y es, con toda seguridad, el factor más determinante para las velocidades del dispositivo.
- Caché: un SSD utiliza un pequeño dispositivo de memoria DRAM similar al caché de los discos duros. El directorio de la colocación de bloques y el desgaste de nivelación de datos también se mantiene en la memoria caché mientras la unidad está operativa.
- Condensador: es necesario para mantener la integridad de los datos de la memoria caché el tiempo suficiente para que se puedan enviar los datos retenidos hacia la memoria no volátil si la alimentación eléctrica se ha detenido inesperadamente.

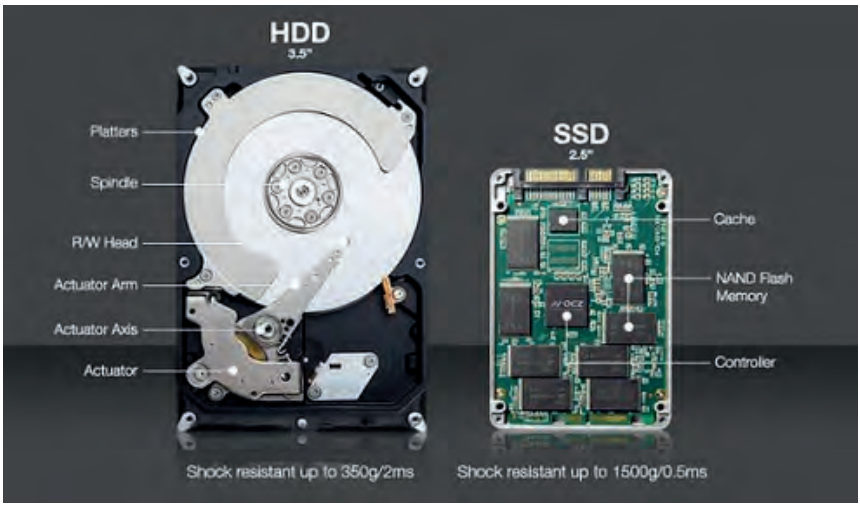


Figura 8.20: HDD vs. SSD

Memoria USB

La memoria USB (*Universal Serial Bus*), denominada, también, lápiz de memoria, lápiz USB, memoria externa, pen drive o pendrive, es un tipo de dispositivo de almacenamiento de datos que utiliza memoria flash para guardar datos e información.

Primera generación

Las empresas Trek Technology e IBM comenzaron a vender las primeras unidades de memoria USB en el año 2000. Trek vendió un modelo bajo el nombre comercial de Thumbdrive e IBM vendió las primeras unidades en Norteamérica bajo la marca DiskOnKey, desarrolladas y fabricadas por la empresa israelí M-Systems en capacidades de 8 MB, 16 MB, 32 MB y 64 MB. Estos fueron promocionados como los «verdaderos reemplazos del disquete», y su diseño continuó hasta los 256 MB. Los modelos anteriores de este dispositivo utilizaban copias de baterías, en vez de la alimentación de la PC 2.

Segunda generación

Dentro de esta generación de dispositivos existe conectividad con la norma USB 2.0. Sin embargo, no usan en su totalidad la tasa de transferencia de 480 Mb/s que soporta la especificación USB 2.0 Hi-Speed debido a las limitaciones técnicas de las memorias flash basadas en NAND. Los dispositivos más rápidos de esta generación usan un controlador de doble canal, aunque todavía están muy lejos de la tasa de transferencia posible de un disco duro de la actual generación, o el máximo rendimiento de alta velocidad USB. Las velocidades de transferencia de archivos varían considerablemente. Se afirma que las unidades rápidas típicas pueden leer a velocidades de hasta 480 Mb/s y escribir a cerca de la mitad de esa velocidad. Esto es aproximadamente 20 veces más rápido que en los dispositivos USB 1.1, que poseen una velocidad máxima de 24 Mb/s.

Tercera generación

La norma USB 3.0 ofrece tasas de cambio de datos mejoradas enormemente en comparación con su predecesor, además de compatibilidad con los puertos USB 2.0. La norma USB 3.0 fue anunciada a finales de 2009, pero los dispositivos de consumo no estuvieron disponibles hasta principios de 2010. La interfaz USB 3.0 dispone las tasas de transferencia de hasta 4,8 Gb/s, en

comparación con los 480 Mb/s de USB 2.0. A pesar de que la interfaz USB 3.0 permite velocidades de datos muy altas de transferencia, a partir de 2011 la mayoría de las unidades USB 3.0 flash no utilizan toda la velocidad de la interfaz USB 3.0 debido a las limitaciones de sus controladores de memoria, aunque algunos controladores de canal de memoria llegan al mercado para resolver este problema. Algunas de estas memorias almacenan hasta 256 GB de memoria (lo cual es 1024 veces mayor al diseño inicial de M-Systems). También hay dispositivos, que aparte de su función habitual, poseen una memoria USB como aditamento incluido, como algunos ratones ópticos inalámbricos o memorias USB con aditamento para reconocer otros tipos de memorias (microSD, m2, etcétera). En agosto de 2010, Imation anuncia el lanzamiento al mercado de la nueva línea de USB de seguridad Flash Drive Defender F200, con capacidades de 1 GB, 2 GB, 4 GB, 8 GB, 16 GB y 32 GB. Estas unidades de almacenamiento cuentan con un sensor biométrico ergonómico basado en un *hardware* que corrobora las coincidencias de las huellas dactilares de identificación, antes de permitir el acceso a la información.

Figura 8.21: USB

Tarjetas de memoria

La tarjeta de memoria, o tarjeta de memoria flash, es el medio o soporte de almacenamiento de datos que conserva los datos transferidos y guardados de forma correcta, en el tipo de memoria flash. Es un tipo de memoria no volátil, es decir, que conserva los datos incluso con la pérdida de energía eléctrica. Los dispositivos de almacenamiento que leen y graban este tipo de tarjetas, se llaman lectores de tarjetas de memoria. Las tarjetas de memoria flash son usadas para almacenar fotos y videos en las cámaras digitales, teléfonos celulares, tabletas o para para ampliar su capacidad de almacenamiento.

Ejemplos:

- Secure Digital (SD), MiniSD microSD.
- Memory Stick (MS).
- MultiMediaCard (MMC).
- CompactFlash (CF).
- SmartMedia (SM).

Figura 8.22: Tarjetas memoria

Almacenamiento en la nube

El almacenamiento en la nube es un concepto completamente diferente al de las memorias externas de las que hemos hablado anteriormente. En esta ocasión nos referimos a un sistema de almacenamiento externo, pero virtual, no físico. En definitiva, el almacenamiento en la nube es un servicio que se apoya en algún soporte de almacenamiento ya mencionado, magnético, óptico o sólido en un centro de cómputos del mundo. El servicio de almacenamiento en la nube se ofrece a través de un proveedor externo, se ofrece bajo demanda y a un determinado costo en función de la capacidad que se necesite.

Las características principales del almacenamiento en la nube son las siguientes:

- No es necesario tener ningún dispositivo externo, ya que todo se va almacenando en una web a la que se tendrá acceso de forma instantánea desde donde se prefiera.
- Se pueden compartir los recursos almacenados con otros usuarios.
- El sistema de seguridad que ofrece el almacenamiento en la nube es excelente.
- Se integra muy bien con otras aplicaciones.
- Se actualiza de forma periódica.

Figura 8.23: Almacenamiento en la nube

PERIFÉRICOS

La computadora es una herramienta esencial capaz de procesar datos, que ayuda a la mejora y excelencia del trabajo haciéndolo mucho más fácil y práctico. Se han integrado de tal manera a nuestra vida cotidiana, puesto que han transformado los procesos laborales complejos y de gran dificultad hacia una manera más eficiente de resolver los problemas difíciles, buscándole una solución práctica.

El papel que juegan los dispositivos periféricos de la computadora es esencial, ya que sin ellos esta no sería útil a los usuarios.

Los dispositivos periféricos permiten introducir los datos para que la computadora ayude a la resolución de problemas y, por consiguiente, obtener el resultado de dichas operaciones; es decir, estos dispositivos ayudan a comunicarnos con la computadora, para que esta, a su vez, nos ayude a resolver los problemas y realice las operaciones que nosotros no podamos realizar manualmente.

La computadora necesita de entradas para poder generar salidas y estas se dan a través de dos tipos de dispositivos periféricos.

Los periféricos son instrumentos o unidades utilizadas para la comunicación entre la CPU y el mundo exterior. La CPU procesa datos que hay que introducir en su memoria de alguna forma y genera resultados que debe comunicarnos a través de algún medio. Cada periférico suele estar formado por dos partes claramente diferenciadas en cuanto a su misión y a su funcionamiento, que poseen una parte mecánica y una parte electrónica.

Definición

Se denominan periféricos tanto a las unidades o dispositivos a través de los cuales la computadora se comunica con el mundo exterior, como a los sistemas que almacenan o archivan la información, sirviendo de memoria auxiliar de la memoria principal.

La parte mecánica está formada básicamente por dispositivos electromecánicos (conmutadores manuales, motores, electroimanes, servomecanismos,

etcétera) controlados por los elementos electrónicos. La velocidad de funcionamiento de un periférico y el tiempo medio transcurrido entre averías, suelen venir impuestos por los elementos mecánicos.
La parte electrónica o controlador del periférico se encarga de interpretar las órdenes que llegan de la CPU para la recepción o transmisión de datos, dependiendo de que se trate de un periférico de entrada, respectivamente, y de generar señales de control para la activación de los elementos electromagnéticos del periférico que producen o captan los datos en el soporte de información correspondiente (pantalla, impresora, disco magnético, etcétera). En la parte electrónica de los periféricos es común usar elementos optoelectrónicos que actúan como detectores o generadores de la información de entrada o salida, respectivamente. También estos dispositivos se usan como detectores de posición de los elementos mecánicos móviles del periférico.

Según la función que realice cada dispositivo pueden clasificarse de la siguiente manera:

- Unidades de entrada: sirven como medio para introducir información en la memoria del ordenador.
- Unidades de memoria masiva auxiliar: son los medios usados como memoria auxiliar de la memoria principal.
- Unidades de salida: sirven como medio para extraer información de la memoria del ordenador.
- Las unidades funcionales del ordenador, así como estas con los periféricos, se comunican por conjuntos o grupos de hilos denominados buses. Existen periféricos que actúan, en comparación con las unidades centrales, muy lentamente y además pueden estar muy alejados del ordenador central necesitándose hilos muy largos para realizar la conexión.
- Los periféricos se interconectan al bus del sistema directamente, o bien a través de unos circuitos denominados interfaces.
- Existe una gran diversidad de periféricos con distintas características eléctricas y velocidades de funcionamiento. Las interfaces son para adaptar las características de los periféricos a las del bus del sistema.

Dispositivos de entrada

Estos dispositivos permiten al usuario la introducción de datos, comandos y programas. La información introducida es transformada por el ordenador en modelos reconocibles. Los datos se leen de los dispositivos de entrada y se almacenan en la memoria central o interna. Los dispositivos de entrada convierten la información en señales eléctricas que se almacenan en la memoria central.

Teclado

El teclado es similar al de una máquina de escribir, correspondiendo cada tecla a uno o varios caracteres, funciones u órdenes; es el dispositivo de entrada más usado para la comunicación de los usuarios con el computador. Para seleccionar uno de los caracteres de una tecla puede ser necesario pulsar simultáneamente dos o más teclas, una de ellas la correspondiente al caracter.

Al pulsar una tecla se cierra un conmutador que hay en el interior del teclado, estos hacen que unos circuitos codificadores del controlador del teclado generan el código correspondiente al caracter seleccionado, almacenándolo en la memoria intermedia del teclado, apareciendo este en la pantalla si no es un caracter de control.

El teclado contiene los siguientes tipos de teclas:

Teclado principal: contiene los caracteres alfabéticos, numéricos y especiales, como en una máquina de escribir convencional con alguno adicional. También incluyen caracteres gráficos. Teclas de desplazamiento del cursor: permiten desplazar el cursor a izquierda, derecha, arriba y abajo, borrar un carácter o parte de una línea.

Teclado numérico: contiene teclas de funciones cuya función es definible por el usuario o está predefinida en un programa. Las teclas de funciones locales controlan funciones propias del terminal, como impresión del contenido de imagen cuando la computadora está conectada a una impresora.

El teclado se conecta a la computadora por medio de una ficha denominada PS/2, pero también puede hacerlo vía USB o ser inalámbrico.

Figura 9.1: Teclado

Ratón o mouse

El ratón es un pequeño periférico de entrada que utiliza un sistema óptico (como el láser) de ubicación espacial en el plano. Es un dispositivo electrónico que permite dar instrucciones a la computadora a través de un cursor que aparece en la pantalla, haciendo clic para que se lleve a cabo una acción determinada a medida que el ratón rueda sobre el escritorio, el cursor (puntero) en la pantalla hace lo mismo. Tal procedimiento permitirá controlar, apuntar, sostener y manipular varios objetos gráficos y de texto en un programa.

Al momento de activar el ratón, se asocia su posición con la del cursor en la pantalla. Si se lo desplaza sobre una superficie, el cursor seguirá dichos movimientos. Es muy empleado en aplicaciones dirigidas por menús o entornos gráficos, como por ejemplo, Windows ya que con un pulsador adicional en cualquier instante se pueden obtener en programa las coordenadas (x, y) donde se encuentra el cursor en la pantalla, seleccionando de esta forma una de las opciones de un menú.

El ratón ha sido uno los elementos que más variaciones ha sufrido en su diseño. Existen modelos en los que la transmisión se hace por infrarrojos por tanto se elimina la necesidad de cableado.

Tipos de ratón:

Mecánico: es poco preciso, estaba basada en contactos físicos eléctricos a modo de escobillas que en poco tiempo comenzaban a fallar.

Óptico: es el más utilizado y fabricado en la actualidad.

Optomecánico: es muy preciso, pero demasiado caro y falla a menudo.

Trackball: es un dispositivo en el cual se mueve una bola con la mano, en lugar de estar abajo y arrastrarla por una superficie.

Óptico mouse trackball: es una superficie del tamaño de una tarjeta de visita por la que se desliza el dedo para manejar el cursor, es estático, ideal para cuando no se dispone de mucho espacio.

Touch: muy usado en las notebooks.

Al igual que los teclados, los ratones se pueden conectar a través de un cable (RS-232, PS/2, USB), por *bluetooth* o ser inalámbricos.

Figura 9.2: Mouse - Trackball

Lector óptico o lápiz óptico

Un lector óptico tiene la misión de transformar en datos binarios todo tipo de información manuscrita o imágenes ya impresas, que puede ser procesada por la computadora. Existen distintas tecnologías que permiten contar con esta lectura. Una de ellas es la utilización de una especie de lápiz óptico en el cual se debe predefinir que áreas deben ser leídas por el lector, para poder realizar un posterior procesamiento eficiente, a alta velocidad y con gran rendimiento.

El dispositivo de lector óptico tiene que tener la aptitud necesaria para poder detectar aquellas áreas que difieren del fondo de un papel y aquellas que están tratadas con un elemento escritor que ha dejado una impronta de tinta. Lo común es que esta información binaria sea transformada en un archivo de imagen, aunque hoy en día las tecnologías y las aplicaciones en el *software* permiten, inclusive, que se puedan obtener documentos digitales de fuentes en papel.

Los códigos de barras se leen utilizando estos dispositivos. Otros similares son el lector de caracteres, lector de bandas magnéticas, lector de marcas, etcétera.

Figura 9.3: Lector código de barra - lápiz óptico

Lector de tarjetas (Smart Card)

Cuando estudiamos los dispositivos de almacenamiento o memorias auxiliares comentamos y describimos las tarjetas de memorias. Un lector de tarjetas de memoria es un dispositivo de almacenamiento de datos para acceder (leer) los datos en una tarjeta de memoria, como por ejemplo, CompactFlash (CF), Secure Digital (SD) o MultiMediaCard (MMC). Es un periférico de entrada. La mayoría de los lectores de tarjetas también ofrecen capacidad de escritura, y junto con la tarjeta, esto puede funcionar como una memoria USB o pendrive. Algunas impresoras y computadoras personales tienen un lector de tarjetas incorporado.

Un lector de tarjetas múltiple se utiliza para la comunicación con más de un tipo de tarjeta de memoria flash. Los multilectores de tarjetas no se han incorporado en la capacidad de memoria, pero son capaces de aceptar varios tipos y estilos de las tarjetas de memoria.

Reconocedor de voz

Representa uno de los campos de investigación actual más relevante, donde se pretende una comunicación directa del hombre con la computadora, sin necesidad de transcribir la información a través de un teclado u otros soportes de información.

Este dispositivo trata de identificar fonemas o palabras dentro de un repertorio o vocabulario muy limitado. Un sistema capaz de reconocer, suponga-mos, siete palabras, lo que hace al detectar un sonido es extraer características

o parámetros físicos inherentes a dicho sonido, y compararlos con los parámetros (previamente memorizados) de las siete palabras que es capaz de reconocer.

Existen dos tipos de unidades de reconocimiento de la voz:

Dependientes del usuario: en estos sistemas es necesario someter el dispositivo a un período de aprendizaje o programación, al cabo del cual puede reconocer ciertas palabras del usuario en el cual el sistema memoriza las características de los sonidos emitidos por el locutor, que luego tendrá que identificar.

Independientes del usuario: están más difundidos. Los parámetros de las palabras que identifican vienen ya memorizados al adquirir la unidad.

Joystick o palanca manual de control

Este dispositivo está constituido por una caja de la que sale una palanca o mando móvil. Un joystick o palanca de juegos tiene normalmente una base de plástico redonda o rectangular, a la que está acoplada una palanca vertical. Los botones de control se localizan sobre la base y algunas veces en la parte superior de la palanca, que puede moverse en todas las direcciones para controlar el movimiento de un objeto en la pantalla. Los botones activan diversos elementos de *software*, generalmente produciendo un efecto en la pantalla. El usuario puede actuar sobre el extremo de la palanca exterior a la caja, y a cada posición de ella le corresponde sobre la pantalla un punto de coordenadas (x, y). La caja dispone de un pulsador que debe ser presionado para que exista una interacción entre el programa y la posición de la palanca. La información que transmite es analógica, no es digital.

Su uso mayoritariamente esta dado en juegos que requieran este tipo de mandos.

Figura 9.4: Joystick

Digitalizador o tabla gráfica

Es un dispositivo de entrada que permite transferir, directamente a la computadora, gráficos, figuras, planos, mapas, o dibujos en general. Esto se realiza manualmente, una pieza móvil pasa por encima de la línea a digitalizar y automáticamente se transfieren las coordenadas (x, y) de los distintos puntos que forman la imagen, unas detrás de otras; es decir, partiendo de un dibujo se obtiene su representación digital en el interior. Los más difundidos son los que se utilizan en terminales como bancos, cajeros automáticos o terminales turísticas.

Un digitalizador consta de tres elementos:

Tabla: donde se ubica el dibujo a digitalizar (puede ser opaca o transparente).

Mando: con el que el usuario debe recorrer el dibujo (lápiz o cursor). El cursor tiene una ventana cerrada con una lupa, en cuyo interior se encuentra embebida una retícula en forma de cruz para señalar o apuntar con precisión el punto a digitalizar. El mando puede disponer de uno o varios pulsadores para controlar la modalidad de funcionamiento, forma de transmisión y selección de opciones del programa que gestiona la digitalización.

Circuitos electrónicos: que controlan el funcionamiento de la unidad.

Figura 9.5: Pantalla táctil

Escáner o rastreador

Es un dispositivo que recuerda a una fotocopiadora que se emplea para introducir imágenes. Las imágenes que se capturan deben estar correctamente iluminadas para evitar brillo y tonos no deseados. Es un dispositivo de entrada de datos de propósito especial que se emplea conjuntamente con paquetes *software* para gráficos y pantallas de alta resolución. La mayor parte de los escáneres capturan imágenes en color. Dada la cantidad de espacio de almacenamiento que se necesita para una imagen no suelen capturarse imágenes en movimiento.

La información se almacena en archivos en forma de mapas de bits (bit maps), o en otros formatos más eficientes como Jpeg o Gif, existen escáneres que codifican la información gráfica en blanco y negro, y a colores.

Hay escáneres de plataforma plana fija (cama plana) con apariencia muy similar a una fotocopiadora, y de barrido manual. Los primeros pueden verificar una página entera a la vez, mientras que los portátiles solo pueden revisar franjas de alrededor de 4 pulgadas. Reconocen imágenes, textos y códigos de barras, convirtiéndolos en código digital.

Figura 9.6: Escáner

Micrófono

El micrófono es el transductor encargado de transformar energía acústica en energía eléctrica, permitiendo el registro, almacenamiento, transmisión y procesamiento electrónico de las señales de audio. Es un dispositivo dual del altoparlante, constituyendo, ambos transductores, los elementos más

significativos en cuanto a las características sonoras que sobreimponen a las señales de audio.

Existe el micrófono de diadema que se adhiere a la cabeza del usuario como una diadema cualquiera, lo que brinda mayor comodidad ya que no necesita sostenerlo con las manos y le permite realizar otras actividades.

Figura 9.7: Micrófono

Dispositivos de salida

Estos dispositivos permiten al usuario ver los resultados de los cálculos o de las manipulaciones de datos en la computadora.

Impresora

Es el periférico que la computadora utiliza para presentar información impresa en papel. Las primeras impresoras nacieron muchos años antes que el PC, e incluso antes que los monitores, es el método más usual para presentar los resultados de los cálculos en aquellos primitivos ordenadores.

Fundamento del sistema de impresión: hay impresoras que realizan la impresión por impacto de martillos o piezas móviles mecánicas, y otras, sin impacto mecánico. El fundamento de las impresoras por impacto es similar al de las máquinas de escribir, sobre la superficie de la línea a imprimir en el papel se desliza una cinta entintada, y delante de esta pasa una pieza metálica donde está moldeado el juego de tipos de impresión. Cuando pasa el tipo a grabar sobre su posición en el papel, se dispara un martillo que golpea la cinta

contra el papel, quedando impreso en tinta sobre el papel el carácter en cuestión.

Las impresoras de impacto son muy ruidosas y tradicionalmente han sido las más utilizadas. Entre ellas se encuentran las impresoras de rueda, bola, margarita, matriciales, cilindro, cadena.

Las impresoras sin impacto forman los caracteres sin necesidad de golpes mecánicos y utilizan otros principios físicos para transferir las imágenes al papel. Son impresoras sin impacto las térmicas, de inyección de tinta, las impresoras láser.

Hay varios tipos que se mencionarán a continuación.

Matriciales:

Ofrecen mayor rapidez pero una calidad muy baja. Consiste en un conjunto de agujas dispuestas verticalmente, puede ser proyectado contra la cinta entintada y el papel al aplicar una corriente eléctrica a sus respectivos electroimanes, volviendo a la posición inicial por mediación de un muelle. Imprime caracteres compuestos por puntos empleando un cabezal de impresión formado por agujas accionadas electromagnéticamente. Los parámetros principales de calidad de impresión de una impresora matricial son el número de puntos de la matriz de agujas y su velocidad. El número de agujas del cabezal de impresión suele ser 9, 18 o 24. La calidad de la impresión es bastante mala ya que se distinguen fácilmente los puntos separados que conforman cada letra.

Figura 9.8: Impresora Matricial

Inyección:

La tecnología de inyección a tinta es la que ha alcanzado un mayor éxito en las impresoras de uso doméstico o para pequeñas empresas, gracias a su relativa velocidad, calidad y, sobre todo, precio reducido, que suele ser la décima parte de una impresora de iguales características. Claro está que hay razones de peso que justifican estas características, pero para imprimir algunas cartas, facturas y pequeños trabajos, el rendimiento es similar y el costo muy inferior. Hablamos de impresoras de color porque la tendencia del mercado es que la informática en conjunto sea en color. Esta tendencia empezó hace una década con la implantación de tarjetas gráficas y monitores en color. Todavía podemos encontrar algunos modelos en blanco y negro, pero ya no son recomendables.

El descubrimiento de esta tecnología fue fruto del azar. Al acercar accidentalmente el soldador, por parte de un técnico, a un minúsculo cilindro lleno de tinta, salió una gota de tinta proyectada, naciendo la inyección de tinta por proceso térmico. La primera de este tipo de impresión data del año 1951.

Actualmente hay varias tecnologías, aunque son muy pocos los fabricantes a nivel mundial que las producen; la mayoría es de un mismo fabricante cuya marca la pone el que las vende.

El fundamento físico es emitir un chorro de gotas de tinta ionizadas que en su recorrido es desviado por unos electrodos según la carga eléctrica de las gotas. El carácter se forma con la tinta que incide en el papel. Cuando no se debe escribir, las gotas de tinta se desvían hacia un depósito de retorno. Estas impresoras son bidireccionales y hay modelos que imprimen en distintos colores.

Figura 9.9: Impresora inyección

Láser:

Ofrece rapidez y una mayor calidad, pero tiene un alto costo y solo se suele utilizar en la mediana y gran empresa. Por medio de un haz de láser imprime sobre el material que corresponde las imágenes que le haya enviado la CPU.

Otro tipo de impresora es la térmica que utiliza un papel especial, termosensible, que se ennegrece al aplicar calor. Este se transfiere desde el cabezal por una matriz de pequeñas resistencias en las que, al pasar una corriente eléctrica, se calientan y forman los puntos en el papel. Estas impresoras pueden ser de caracteres o de líneas.

Las impresoras láser utilizan un papel especial eléctricamente conductor (de color gris metálico). La forma de los caracteres se produce por medio de cargas eléctricas que se fijan en el papel a través de una hilera de plumillas que abarcan el ancho del papel. Luego de estar formada eléctricamente la línea, se la hace pasar, avanzando el papel, por un depósito donde se la pulveriza con un líquido que contiene partículas de tóner (polvo de carbón) suspendidas. Las partículas son atraídas en los puntos que conforman el carácter. Estas impresoras de línea son muy rápidas. En la actualidad tienen gran importancia, no solo por su elevada velocidad, sino por la calidad de impresión, relativo bajo precio y la utilización de tamaños de papel A4 carta u oficio.

Su fundamento es muy parecido al de las máquinas de fotocopiar. La página a imprimir se transfiere al papel por contacto, desde un tambor que contiene la imagen impregnada en tóner. La impresión se realiza mediante radiación láser, dirigida sobre el tambor cuya superficie tiene propiedades electrostáticas (se trata de un material fotoconductor, tal que, si la luz incide sobre su superficie, la carga eléctrica de esa superficie cambia).

Figura 9.10: Impresora Láser

Impresora LED:

Es análoga a la de láser, la única diferencia radica en que la imagen se genera desde una hilera de diodos, en vez de un láser. Al ser un dispositivo fijo, es más compacta y barata, aunque la calidad es peor. Algunas de las que se anuncian como láser a precio barato, son de esta tecnología.

Impresora térmica:

Similar a la impresora de aguja, utiliza un papel especial termosensible que se ennegrece al aplicar calor, este se transfiere desde el cabezal por una matriz de pequeñas resistencias en las que al pasar una corriente eléctrica por ellas se calientan formando los puntos en el papel. Esta puede ser de caracteres o de líneas.

Figura 9.11: Impresora térmica

Forma de imprimir los caracteres:

Impresora de caracteres: realizan la impresión por medio de un cabezal que va escribiendo la línea carácter a carácter. El cabezal se desplaza a lo largo de la línea que se está imprimiendo, solo de izquierda a derecha (impresora unidireccional), o bien, para conseguir mayor velocidad, de izquierda a derecha y de derecha a izquierda, sucesivamente (impresora bidireccional).

Impresora de líneas: imprime simultáneamente todos o varios de los caracteres correspondientes a una línea de impresión.

Impresora de página: imprime de forma muy similar a las máquinas fotocopiadoras. Se caracteriza por contener un tambor rotativo donde se forma

con tinta o polvillo especial (tóner) la imagen de la página a imprimir. Esta imagen, por contacto y un proceso de fijación se transfiere al papel.

Velocidad de escritura:

Normalmente la velocidad de impresión se da en las unidades que se detallan. Impresora de caracteres: caracteres por segundo, impresora de líneas: líneas por minuto o impresora de páginas: páginas por minuto.

Trazador de gráficos o plotter

El trazador de gráficos o plotter es un dispositivo de salida que realiza dibujos sobre papel. Este periférico tiene gran importancia ya que permite salidas en forma de planos, mapas, dibujos, gráficos, esquemas e imágenes en general. El funcionamiento de un plotter se controla desde un programa. El usuario puede incluir en su programa instrucciones para realizar las representaciones que desee con sus datos.

El registrador gráfico se fundamenta en el desplazamiento relativo de un cabezal con el elemento de escritura, con respecto al papel. Dependiendo del tipo de gráfico moverá solo la cabeza o la cabeza y el papel.

Según la forma en que se realiza el dibujo existen tres tipos de registradores: de pluma, electrostático o de inyección.

El registrador electrostático es una impresora electrostática. El sistema de tracción de papel es similar al de una impresora convencional. El dibujo se realiza línea a línea. El elemento de escritura está constituido por una serie de agujas cuya densidad puede variar. El de inyección trabaja de forma análoga a una impresora de inyección de tinta, tal como se describe en el apartado correspondiente.

Existen plotters para diferentes tamaños máximos de hojas (A0, A1, A2, A3 y A4), para diferentes calidades de hojas de salida (bond, calco, acetato), para distintos espesores de línea de dibujo (diferentes espesores de rapidógrafos), y para distintos colores de dibujo (distintos colores de tinta en los rapidógrafos).

Figura 9.12: Plotter

Sintetizador de voz

El sintetizador de voz es un dispositivo que emite sonidos (fonemas o palabras) similares al habla humana, a través de programas. Este periférico de salida suele incluir un microprocesador, memoria ROM con programas y datos, un conversor D/A, un amplificador de audiofrecuencia y altavoz. La mayor parte de los dispositivos sintetizadores de voz tienen memorizados digitalmente cada uno de los fonemas o palabras que son capaces de emitir. Los datos que recibe un sintetizador, procedentes de la computadora, corresponden a la identificación de los fonemas o palabras a emitir. Una vez que se analiza el dato, se activa una rutina encargada de generar el sonido correspondiente.

Los sonidos resultan muy metálicos. Por lo general, estos sistemas incluyen programas que los enriquecen, como por ejemplo, generar frases o combinaciones de palabras, incluso hay sistemas que traducen cantidades.

Visualizador

El visualizador es una pequeña unidad de salida que permite al usuario leer una instrucción, un dato o un mensaje. Los caracteres se forman partiendo de estructuras en módulos, cada uno de los cuales sirve para visualizar un carácter. Cada módulo contiene una serie de segmentos, siendo los más habituales de 7. Un carácter concreto se visualiza activando determinados segmentos,

dependiendo de la forma del carácter. Un ejemplo típico es la calculadora de bolsillo y del reloj digital.

Figura 9.13: Visualizador

Monitor

El monitor es un dispositivo electrónico de salida de la computadora en el que se muestran las imágenes y textos generados por medio de un adaptador gráfico o de video. El término monitor se refiere, normalmente, a la pantalla de video, y su función principal y única es la de permitir al usuario interactuar con la computadora.

Una computadora típica presenta un monitor con tecnología CRT (tubos de rayos catódicos), la misma que emplean los televisores; sin embargo, hoy en día existe la tecnología TFT (transistor de película fina) que reduce significativamente el volumen del monitor. También se encuentra la tecnología LCD (dispositivos de cristal líquido), plasma, EL (electroluminiscencia) o FED (dispositivos de emisión de campo).

En cualquiera de esas tecnologías, la imagen mostrada consta de píxeles (pequeños puntos o elementos de imagen). La resolución de un monitor se refiere a la cantidad de píxeles que se muestran en un determinado espacio. A mayor resolución, más amplitud tendrá el monitor.

El monitor también es un aparato o programa dedicado a gestionar información de algún tipo como datos visuales o sonoros. Se utiliza mucho en medicina, por ejemplo, cuando se realiza un ecosonograma en una mujer embarazada, a través del monitor se muestra la imagen del feto y también se pueden escuchar los latidos de su corazón. En campo de la educación y sociocultural, el monitor es el encargado de guiar y orientar a un grupo de personas o estudiantes, o de enseñar una actividad deportiva o cultural. Tiene como

funciones motivar, movilizar, sensibilizar a las personas, ayuda a asumir responsabilidades, descubre en los individuos sus aspiraciones, sus necesidades, sus esperanzas, sus trabajos, etcétera.

Atendiendo al color

Monitor color: su pantalla está formada internamente por tres capas de material de fósforo, una por cada color básico (rojo, verde y azul). También consta de tres cañones de electrones, que al igual que las capas de fósforo, hay uno por cada color. Para formar un color en pantalla que no sea ninguno de los colores básicos, se combinan las intensidades de los haces de electrones de los tres colores básicos.

Monitor monocromático: muestra un solo color: negro sobre blanco o ámbar, o verde sobre negro. Uno de estos monitores con una resolución equivalente a la de un monitor color, si es de buena calidad, generalmente es más nítido y más legible.

Atendiendo a la tecnología

Monitor de cristal líquido: el cristal líquido es una sustancia transparente con cualidades propias de líquidos y de sólidos. La luz que atraviesa un cristal líquido sigue el alineamiento de las moléculas aplicando una carga eléctrica a estos cristales, se produce un cambio en la alineación de las moléculas, y por tanto, en el modo en que la luz pasa a través de ellas. Una pantalla LCD está formada por dos filtros polarizantes con filas de cristales líquidos alineados perpendicularmente entre sí, de modo que al aplicar o dejar de aplicar una corriente eléctrica a los filtros, se consigue que la luz pase o no pase a través de ellos, según el segundo filtro bloquee o no el paso de la luz que ha atravesado el primero. El color se consigue añadiendo tres filtros adicionales de color (uno rojo, uno verde, uno azul).

En la actualidad coexisten varios tipos: Dual Scan (DSTN), HPA y Matriz Activa (TFT), que permiten una visualización perfecta sean cuales fuesen las condiciones de iluminación exterior.

La principal diferencia entre el monitor LCD y el plasma, es que el primero utiliza, básicamente, moléculas de cristal líquido, las cuales cambian su orientación en función al voltaje que se les aplica, así cambian de forma y de color y pueden generar imágenes. Mientras que el segundo está integrado por miles

de celdas de vidrio, las cuales se estimulan por la carga de plasma que se ejerce sobre ellas.

Monitor con tubo de rayos catódicos: las señales digitales del entorno son recibidas por el adaptador de VGA. Este lleva las señales a través de un circuito llamado convertidor analógico digital (DAC). Generalmente, el circuito de DAC está contenido dentro de un chip especial que realmente contiene tres DAC, uno para cada color básico. Los circuitos DAC comparan los valores digitales enviados por la PC en una tabla que contiene los niveles de voltaje coincidentes con los tres colores básicos necesarios para crear el color de un único píxel. El adaptador envía señales a los tres cañones de electrones localizados detrás del tubo de rayos catódicos del monitor (CRT). Cada cañón de electrones expulsa una corriente de electrones, una cantidad por cada uno de los tres colores básicos.

La imagen está formada por una multitud de puntos de pantalla, uno o varios puntos de pantalla forman un punto de imagen (píxel), una imagen se constituye en la pantalla del monitor por la activación selectiva de una multitud de puntos de imagen.

Después de que los haces hagan un barrido horizontal de la pantalla, las corrientes de electrones son apagadas cuando el cañón de electrones enfoca las trayectorias de los haces en el borde inferior izquierdo de la pantalla en un punto exactamente debajo de la línea de barrido anterior, este proceso es llamado refresco de pantalla. Los barridos a través de la superficie de la pantalla se realizan desde la esquina superior izquierda de la pantalla a la esquina inferior derecha. Un barrido completo de la pantalla es llamado campo. La pantalla es normalmente redibujada, o refrescada, unas 60 veces por segundo, haciéndolo imperceptible para el ojo humano.

Figura 9.14: Monitor de Tubo - Cristal Líquido

Tamaño

Los monitores tienen diversos tamaños, generalmente van desde las 15 pulgadas a 24 pulgadas entre los más comunes, con resoluciones desde 800 x 600 75 Hz a 1280 x 1024 a 85 Hz.

Resolución

Se trata del número de puntos que puede representar el monitor por pantalla, en horizontal por vertical. Así, un monitor cuya resolución máxima es de 1024 x 768 puntos, quiere decir que es capaz de representar hasta 768 líneas horizontales de 1024 puntos cada una, además de otras resoluciones inferiores, como 640 x 480 u 800 x 600. Cuanto mayor sea la resolución de un monitor, mejor será la calidad de la imagen en pantalla, y mayor será la calidad del monitor. La resolución debe ser apropiada, además, al tamaño del monitor.

Refresco de pantalla

Es el número de veces que se escribe la información en pantalla por unidad de segundo. También se llama "frecuencia de refresco vertical". Se puede comparar al número de fotogramas por segundo de una película de cine, por lo que deberá ser lo mayor posible. Se mide en Hz (hercios) y debe estar por encima de 60 Hz, preferiblemente 70 Hz u 80 Hz. A partir de esta cifra, la imagen en la pantalla es sumamente estable, sin parpadeos apreciables, con lo que la vista sufre mucho menos.

Antiguamente los monitores solo podían presentar imágenes con unos refrescos determinados y fijos, por ejemplo, los monitores CGA o EGA y algunos VGA; hoy en día todos los monitores pueden presentar varios refrescos dentro de un rango determinado (multiscan). La tarjeta gráfica es la que proporciona estos refrescos, pero quien debe presentarlos es el monitor. Si se pone un refresco de pantalla que el monitor no soporta, se podría dañar, por lo que debemos conocer su rango de velocidades de refresco para no tener ningún problema, por eso lo mejor es leer con detenimiento el manual o mirar otro parámetro denominado "frecuencia horizontal", que debe ser lo mayor posible, entre unos 30 kHz a 80 kHz.

Conexiones

En lo que respecta a las conexiones, no debe faltar el típico conector mini D-sub de 15 pines (VGA) y el S-Video. En monitores de 17 pulgadas o más, es

interesante que existan, además, conectores BNC que presentan la ventaja de separar los tres colores básicos; también en los monitores más modernos debe estar presente otra conexión digital, la DVI. De cualquier modo, esto solo importa si la tarjeta gráfica aun los incorpora y si la precisión en la representación del color resulta determinante en el uso del monitor.

Figura 9.15: Conectores de Monitor

Dispositivos mixtos

Son algunos dispositivos que tienen la propiedad de poder ser usados tanto como de entrada de datos como para la salida de ellos.

Pantalla sensible al tacto

Es una pantalla que puede detectar las coordenadas (x, y) de la zona de la propia pantalla donde se acerca algo (por ejemplo, un dedo). Este es un sistema muy sencillo para dar entradas o elegir opciones sin utilizar el teclado. Se utiliza para la selección de opciones dentro del menú o como ayuda en el uso de editores gráficos. Con frecuencia se ve en los denominados kioscos informativos, cada vez más difundido en grandes empresas, bancos y en puntos de información urbana. Existen pantallas con toda su superficie sensible, y otras en las que solo una parte de ella lo es. Dependiendo de su tecnología la información se muestra como las descriptas en los monitores.

Lectora/perforadora de tarjetas

Se encuentra entre los primeros dispositivos que existieron como entrada/salida de datos.

La tarjeta perforada es una cartulina dura, rectangular, con una esquina cortada para identificar de forma visual o mecánica su posición y su cara correctas. Es un sistema obsoleto desde hace años.

La información se representa por medio de caracteres que se graban a través de perforaciones en determinadas posiciones de la tarjeta. Esta operación

se realiza en máquinas auxiliares, denominadas perforadoras, parecidas a la máquina de escribir. En el teclado de la perforadora se pulsan los caracteres a grabar y automáticamente se efectúan los taladros en las posiciones correspondientes. También se puede obtener información en tarjetas perforadas en ordenadores que tienen una unidad de salida de perforación de tarjetas que funciona en línea con la computadora.

El formato de tarjeta más utilizado corresponde al de 80 caracteres, también denominado tarjeta de Hollerith. Posteriormente, se crearon unas nuevas tarjetas más pequeñas en las que se puede grabar más información, de 96 caracteres.

Durante la primera y la segunda generación de ordenadores, las tarjetas perforadas fueron el principal soporte de información utilizado como entrada. No obstante, su uso era engorroso y caro.

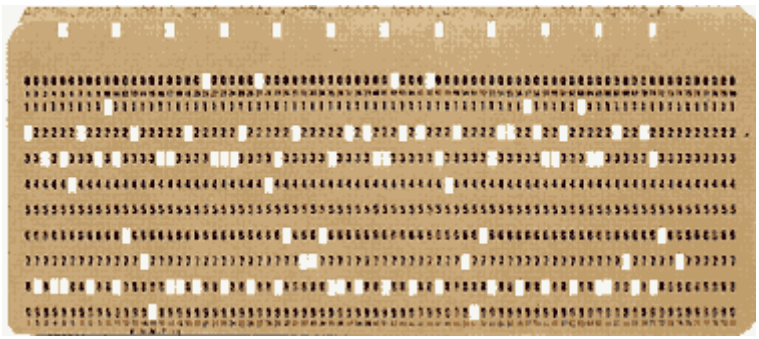


Figura 9.16: Tarjeta Perforada

Lectora de tarjetas perforadas

Los sistemas de lectura de tarjetas perforadas son unidades de entrada al ordenador que transforman en señales eléctricas binarias a la información contenida en forma de perforaciones en la tarjeta.

Están constituidos fundamentalmente por tres partes a detallar.

Cajón o depósito de alimentación: donde se deposita el bloque de tarjetas a leer.

Estación de lectura: que suele estar constituida por una serie de fotocélulas, y cada fotocélula detecta las perforaciones de una fila. Las tarjetas son arrastradas una a una de forma mecánica del cajón de alimentación a la estación de lectura. Las tarjetas pasan longitudinalmente entre las células fotoeléctricas y una fuente de luz. Las fotocélulas detectan la presencia de un orificio al captar la luz procedente del otro lado de la tarjeta. De esta forma, y con un circuito decodificador adecuado, se rellena la memoria intermedia de la lectora con la información de una tarjeta en un código de E/S. Una vez llena la memoria intermedia, se transmite su contenido al ordenador central mientras se procede al arrastre de la siguiente tarjeta.

Cajón o depósito de salida: en el que se depositan las tarjetas, después de ser leídas, llegando en el mismo orden en que se ubicaron en el depósito de alimentación.

Unidad de perforación de tarjetas

Está constituida por las mismas partes fundamentales de la lectura, pero añadiendo una estación de perforación. Como es lógico también es un dispositivo fuera de uso.

Ahora se usarán tarjetas vírgenes y entre el cajón inicial y la estación de lectura existe una estación de perforación constituida por una hilera de punzones que son activados electromagnéticamente de acuerdo con los caracteres grabados en la memoria intermedia de la perforadora por el ordenador. Una vez perforada la tarjeta, atraviesa una estación de lectura que verifica las perforaciones realizadas.

Módem/Router

El módem es un dispositivo que permite conectar dos ordenadores remotos utilizando la línea telefónica de forma que puedan intercambiar información entre sí. Es uno de los métodos más extendidos para la interconexión de computadoras por su sencillez y bajo costo. Razones que lo hacen el método más popular de acceso a la Internet por parte de los usuarios privados y también de muchas empresas. La información que maneja el ordenador es digital, es decir, está compuesta por un conjunto discreto de dos valores el 1 y el 0. Sin embargo, por las limitaciones físicas de las líneas de transmisión no es posible enviar información digital a través de un circuito telefónico. Para poder

utilizar las líneas de teléfono (y en general cualquier línea de transmisión) para el envío de información entre computadoras digitales, es necesario un proceso de transformación de la información. Durante este proceso la información se adecua para ser transportada por el canal de comunicación. Este proceso se conoce como modulación-demodulación y es el que se realiza en el módem.

Un módem es un dispositivo que convierte las señales digitales del ordenador en señales analógicas que pueden transmitirse a través del canal telefónico.

Bits por segundo (b/s) es el número efectivo de bits por segundo que se transmiten en una línea por segundo. Como hemos visto, un módem de 600 baudios puede transmitir a 1200 b/s, 2400 b/s o, incluso, a 9600 b/s.

Tanto un módem como un router (también conocido como enrutador o ruteador) se encargan de la conexión a Internet de un hogar o negocio. Un módem se encarga de codificar y decodificar datos, para que el Internet pueda pasar entre la red doméstica y el proveedor de servicios de Internet (ISP, por sus siglas en inglés). En cambio, un ruteador dirige la información recolectada por el módem hacia los aparatos que estén dentro de la red. Por lo tanto, se podría decir que el módem trae la información y el enrutador la distribuye hacia los diferentes dispositivos, como computadoras y celulares.

Figura 9.17: Modem/*Router*

SISTEMAS OPERATIVOS

En primer lugar, trataremos de aproximarnos a una primera definición de lo que es un sistema operativo (SO); como todavía faltan conceptos básicos para comprender cabalmente el tema, haremos un recorrido por la evolución histórica de los sistemas operativos, para ir descubriendo y comprendiendo en cada paso de su evolución, los aportes que hicieron a los actuales SO. En ese punto ya seremos capaces de entender conceptos básicos inherentes al tema y estaremos en condiciones de dar una definición completa de lo que es un SO, sus misiones, funciones y la necesidad del mismo en un moderno sistema de cómputos. Por último, presentaremos un modelo de estudio de los SO, de tipo jerárquico, que nos permita comprender en forma global cómo están organizados.

Concepto de sistema operativo

El *hardware* de las computadoras modernas es muy poderoso y puede usarse para muchos propósitos. Para poder manejar los recursos básicos del *hardware*, se han desarrollado los SO. En una primera aproximación, un sistema operativo puede definirse como una colección de programas de administración, control y servicio, que provee una interface o ambiente amigable, para que los usuarios ejecuten sus programas.

Se puede ver como una colección organizada de *software* (extensión del *hardware*) que consta de rutinas de control para operar una computadora y proporcionar un entorno para la ejecución de los programas de usuario.

Al SO podemos verlo como el pegamento que mantiene juntos a todos los demás componentes *software* del sistema: editores, ensambladores, compiladores, etcétera.

El término SO denota aquellos módulos de programa dentro de una computadora que gobiernan el control de los recursos del equipo tales como procesador, memoria principal, memoria auxiliar, dispositivos de E/S y archivos.

Estos módulos tratan de optimizar la performance y simplificar el uso eficiente del sistema. A veces se los llama control, monitor, ejecutivo, supervisor, entre otros.

Se puede considerar que proporciona una infraestructura, es decir, un medio común en el cual los diferentes programas de sistema pueden operar, incluyendo los mecanismos para comunicarse con los operativos periféricos

El *sistema operativo* es un programa o conjunto de programas de un sistema informático que gestiona los recursos de *hardware* y provee servicios a los programas de aplicación, ejecutándose en modo privilegiado respecto de los restantes. Brindando una interface "amigable" al usuario.

Si dividimos un sistema de cómputos en cuatro componentes:

Figura 10.1: Esquema de Peterson. Sistema de Computo

Un SO puede clasificarse de diversas maneras y con distintos criterios, la siguiente figura representa esas clasificaciones.

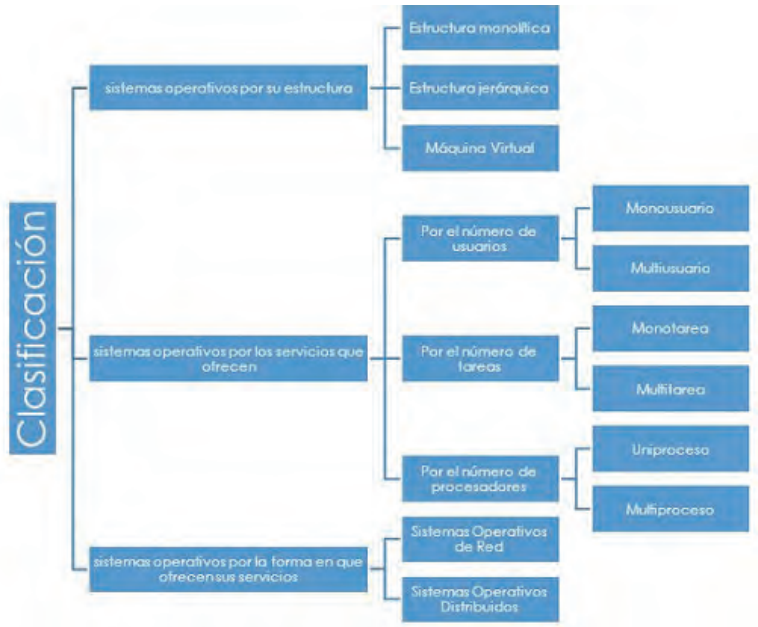


Figura 10.2: Clasificación de un sistema operativo

Para ver lo que es y lo que hace un SO consideraremos su desarrollo en los últimos 30 años. El SO y la arquitectura de las computadoras han tenido una gran influencia mutua. El primero se desarrolló para facilitar el uso del *hardware*. Mientras eran más usados y desarrollados, se hacía obvio que los cambios en el diseño del *hardware* podían simplificar los SO.

Veremos brevemente su evolución para dar sentido a la necesidad de los tipos de servicio que cada variedad de SO proporciona. El recorrido histórico presenta muchos de los conceptos que se unieron para formar la base de los SO actuales.

En particular, describiremos el procesamiento en serie, procesamiento por lotes y multiprogramación. No especificaremos fechas porque nos interesa la progresión de ideas y no el trazado histórico. Además, muchos conceptos e ideas se volvieron a repetir en diferentes momentos y se aplicaron en el desarrollo de las minicomputadoras y las microcomputadoras.

Evolución de los SO

Comienzos de la computación

La programación de la *máquina desnuda* fue habitual en los primeros sistemas de computación. Podemos decir que la máquina se preparaba *a mano*. ¿Qué significa esto? El operador (a menudo el programador) debía poner en marcha la computadora a través de un proceso complejo. Los programas se podían desarrollar traduciendo manualmente las secuencias de instrucciones en código binario o alguna otra base potencia entera de dos. Después se introducían en la computadora las instrucciones y datos mediante interruptores de consola y, en algunos casos, a través de un teclado hexadecimal. El programa se arrancaba cargando el PC con la dirección de la primera instrucción. Los resultados se obtenían de la ejecución examinando los contenidos de los registros relevantes y las posiciones de memoria.

Las máquinas, grandes y costosas, tenían largos períodos de inactividad mientras el operador decidía lo que había que hacer y lo realizaba a velocidad humana.

Este método de procesamiento dejó de ser aceptable, dado que con la evolución del *hardware*, la razón entre los tiempos de preparación y los tiempos de ejecución aumentó hasta llegar a proporciones inaceptables y así surgió la necesidad de automatizar la transición de un trabajo a otro.

Los esfuerzos por bajar esa diferencia llevaron a algunos desarrollos, como la introducción de una pieza de *hardware* llamada canal de E/S que controlaba esos dispositivos en forma autónoma.

Procesamiento en serie

El siguiente paso significativo de la evolución viene con la llegada de los dispositivos de E/S, tales como las tarjetas perforadas y las cintas de papel, así como de los traductores de lenguaje (este avance implica que los programas se comenzaron a escribir en un lenguaje de programación).

Otro programa, el *cargador*, automatiza el proceso de cargar en memoria los programas ejecutables y datos que el usuario coloca en los dispositivos de entrada.

Los mecanismos de desarrollo y preparación de programas en tales entornos son absolutamente lentos por la ejecución en serie de programas y las numerosas operaciones manuales implicadas en los procesos. En una secuencia típica:

1. Carga del programa editor para preparar código fuente.
2. Carga y ejecución del traductor. Alimentarlo con el código fuente del programa del usuario.
3. Repetición del proceso completo si se detectan errores.
4. Eventualmente, carga y ejecución del código objeto producido a partir del código fuente corregido.
5. Para los errores detectados en tiempo de ejecución, se tiene ayuda del *depurador*.
6. Impresión de resultados.

Evidentemente el refinamiento lógico que se produjo fue una colección de rutinas de E/S estándar para usar con todos los sistemas, dado la gran cantidad de programas que usaban estos dispositivos.

En los sistemas descriptos, las rutinas de E/S (drivers) y el cargador, constituían un entorno mínimo de ejecución, es decir, que ya representan una forma rudimentaria de SO.

Por su parte los traductores, editores y depuradores, conforman los programas de sistemas. Entran en los servicios del SO pero no se ven como parte del mismo.

Este modo de operación no era obviamente muy eficaz; la utilización de los recursos del sistema era muy baja, como la productividad de los usuarios, mientras esperaban su turno en la máquina.

Procesamiento por lotes

El paso lógico siguiente en la evolución de los SO fue automatizar la secuencia de operaciones involucradas en la ejecución de un programa y en los aspectos mecánicos del desarrollo de programas.

La E/S fuera de línea permitía que en lugar de utilizarse los lentos dispositivos periféricos, la entrada se transcribiese a cintas magnéticas.

Más específicamente, si se agrupan juntos varios programas en una cinta única de entrada para que las operaciones auxiliares se realicen una sola vez, el tiempo de ocupación por programa se reduce considerablemente. De esta

manera, se organizaban los trabajos de la siguiente forma, a modo de ejemplo, agrupando en lote varios trabajos de compilación FORTRAN y el compilador FORTRAN se cargaba una sola vez para procesarlos todos en fila.

El cuello de botella seguía siendo la lentitud de los dispositivos de E/S tales como lectoras e impresoras, por lo cual se desarrolló la técnica de E/S fuera de línea. En lugar de utilizar la computadora principal con lentos dispositivos periféricos, la entrada se transcribió de tarjetas a cintas magnéticas. Se utilizaba una pequeña computadora como "satélite".

La E/S era ejecutada por rutinas IOCS (I/O control system) a través de un canal autónomo.

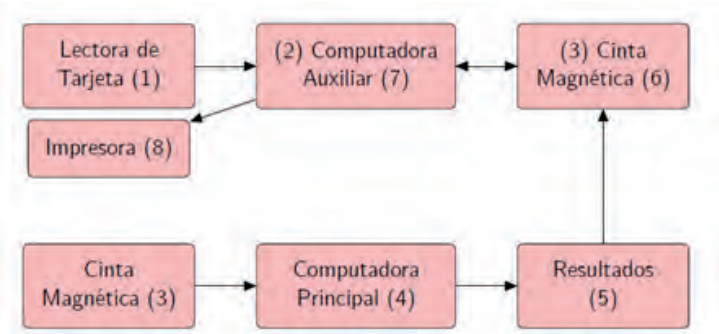


Figura 10.3: Operación fuera de línea de los dispositivos de E/S

Los perfeccionamientos en el manejo por lotes fueron más lejos en las líneas de incrementar el rendimiento total y la utilización de los recursos mediante las operaciones de E/S solapadas.

Estos desarrollos coinciden con avances en el *hardware* como la introducción de canales de acceso directo a memoria, controladores periféricos, etcétera.

Como resultado, se fueron sustituyendo frecuentemente las computadoras satélites para procesamiento fuera de línea por sofisticados programas de E/S ejecutados en la misma computadora con el monitor por lotes. Esto ha permitido el solapamiento de la ejecución de programas con las operaciones de E/S en beneficio de otros programas.

Por ejemplo, se puede almacenar por un tiempo en memoria la salida de un trabajo e imprimirlo después concurrentemente con la ejecución del siguiente trabajo.

El primero de estos sistemas por lotes fue el SOS (*share operating system*), precursor del clásico FMS (*fortran monitor system*).

Para evitar recargar compiladores, vinculadores o cargadores, muy pronto estos programas se hicieron *residentes*, es decir, se podían correr una y otra vez sin que fuera necesario volver a leer de cinta. Para aprovechar esta capacidad, el operador agrupaba en lotes (batches) los programas con requisitos semejantes, evitando así cambios de entorno, cada vez que se cargaba un programa, lo cual redujo el tiempo de preparación de las tareas.

Se proporcionó un medio para instruir al SO sobre cómo procesar cada trabajo individualmente. Estas instrucciones se proporcionan mediante órdenes del SO; las mismas representan las sentencias del Lenguaje de Control de Trabajo (JCL). Ordenes típicas del JCL: marcas de principio y fin del trabajo, órdenes para cargar y ejecutar programas, características como necesidades de memoria, etcétera.

Una porción residente en memoria del SO de lotes, llamada a veces monitor de lotes, lee, interpreta y ejecuta estas órdenes. En respuesta a ellas, se ejecutan los trabajos por lotes.

Ejemplo, cuando se encuentra la orden JOB-END, el monitor puede buscar otro trabajo que se identifica por una orden JOB-START.

Elementos del "Monitor de lotes" 1) Programa cargador, 2) Primitivas de E/S, 3) Intérprete de lenguaje de órdenes de control de lotes (LOAD, RUN,...).

En esta primera época en que las computadoras se especializaban en tareas de cálculo intensivo y los dispositivos que interactuaban con medios externos eran prácticamente desconocidos, el papel del sistema operativo monitor o de control era básicamente asistir al operador en la carga de los programas y las bibliotecas requeridas, la notificación de resultados y la contabilidad de recursos empleados para su cobro.

El siguiente desarrollo lo introdujo la llegada de una técnica de *hardware*: la interrupción. La dificultad que presentaba sincronizar un canal de E/S autónomo, se resuelve haciendo que el canal señale a la CPU cuando ha concluido su trabajo o cuando ocurre una condición de error. Entonces, el *hardware* conserva su estado actual e introduce una rutina de interrupción que se ocupa del canal. Luego, el programa interrumpido se reanuda.

Este concepto se utilizó para obtener un mejor uso del equipo por medio de la técnica de multiprogramación. La primera computadora que introdujo este concepto fue la Ferrari Atlas.

El concepto de multiprogramación tuvo una importante aplicación en el desarrollo de los sistemas operativos. Se deja de usar la computadora satélite y se comienza a usar una sola para efectuar los dos trabajos por medio de multiprogramación. Como no hay satélite, no existe transferencia física de un lote. Los primeros sistemas usaban cinta, pero con la aparición del disco magnético el sistema se volvió de flujo continuo.

A estos sistemas se los denomina "Sistemas manejados por SPOOL".

Sistemas de manejo por SPOOL

Formas más sofisticadas de almacenamiento temporal llamadas SPOOLing (Operaciones periféricas simultáneas en línea, *Simultaneous Peripheral Operations On Line* en inglés), usan discos para almacenar temporalmente las Entradas y Salidas de los trabajos.

El monitor de SPOOL realiza las operaciones de tarjeta a disco y de disco a impresora para los programas siguiente y anterior, respectivamente, y concurrentemente con la ejecución del programa actual.

La ventaja de SPOOLing sobre el uso de cintas es que permite la concurrencia de la E/S de un trabajo con el procesamiento de otros trabajos.

Toda la actividad de concurrencia se encapsula en un ejecutivo de programación.

Multiprogramación

Es un intento de incrementar la utilización de la CPU, teniendo siempre algo para que ejecute. Requiere que muchos programas residan en memoria al mismo tiempo.

El SO toma uno de los programas residentes en memoria y comienza a ejecutarlo. Eventualmente puede tener que esperar por algo, tal como el montaje de una cinta, un comando del teclado, o una operación de E/S. En un sistema multiprogramado, el SO simplemente toma otro trabajo y lo ejecuta. Cuando este necesita esperar, vuelve a interrumpirse y se sigue ejecutando otro, y así sucesivamente. Mientras haya algo por ejecutarse, la CPU nunca estará inactiva.

Los trabajos deben estar activos en memoria para poder ser corridos, esto implica algún tipo de manejador de memoria. Además están todos listos para correr al mismo tiempo, se deberá tomar una decisión entre ellos para elegir cuál entra. Este trabajo lo realiza el manejador del procesador. La multiprogramación también tiene en cuenta temas como control de concurrencias, protección, etcétera.

Sistemas de tiempo compartido y multiacceso

El concepto surgió de una idea de asombrosa sencillez, una variante de la idea de multiprogramación, que era la base de los sistemas de manejo por SPOOL que entonces se utilizaban: al permitirse que cierto número de programas utilizaran el procesador en estricta secuencia, cada uno en posesión de una ranura de tiempo (quantum), entonces, si la ranura de tiempo fuese lo suficientemente pequeña parecería que todos los programas avanzan en paralelo. Si cada programa está asociado con un usuario en una terminal, a cada uno le parecería que usaba su propio sistema de computación.

Resumiendo: a fin de servir simultáneamente a un gran número de usuarios, la computadora puede operar en tiempo compartido, lo cual significa que dedica a cada usuario una parte de su tiempo, con una periodicidad determinada, para que teniendo en cuenta los respectivos tiempos de respuesta de la computadora y del hombre, este tenga la impresión de poseer enteramente la máquina.

Sistemas conversacionales

Permiten a los usuarios seguir el desarrollo de las diferentes etapas de sus programas, así como intervenir sobre este desarrollo por medio de terminales adaptadas al diálogo (teclado, máquina de escribir conectada, etcétera).

La evolución de los métodos de explotación de las computadoras permite comprender mejor el papel y la complejidad de los sistemas operativos.

Reducidos al cargador y al traductor de lenguaje en la primera generación, tomaron a su cargo las funciones de encadenamiento de trabajos y de gestión de E/S en la segunda y, posteriormente, a partir de la tercera, la gestión de la multiprogramación, del tiempo compartido y de máquinas virtuales.

Funciones de un sistema operativo

Según Milenkovic un SO se puede ver como una colección organizada de *software* (que extiende al *hardware*) y que consta de rutinas de control para operar una computadora y proporcionar un entorno para la ejecución de programas.

Para Barron, el SO es como el pegamento que mantiene juntos a todos los otros componentes del *software* de sistema: editores, ensambladores, compiladores, etcétera.

Se puede considerar que el SO proporciona una infraestructura, un medio común en el cual los diferentes programas del sistema pueden operar, incluyendo los mecanismos para comunicarse con los periféricos, que se pueden puntualizar en los siguientes:

- Los programas invocan los servicios del SO mediante las "llamadas al SO".
- Los usuarios pueden interactuar con el SO directamente mediante las "órdenes del SO".
- Se desarrollaron inicialmente como respuesta a la necesidad de maximizar la utilización del procesador central y de los dispositivos periféricos.
- Con la reducción de los costos del *hardware*, conforme ha disminuido la necesidad de optimizar el uso, se ha vuelto cada vez más importante otro aspecto del SO: la interfaz del *software*.

Misión de un SO

1. Control de los recursos: los recursos que obligaron a la creciente automatización de la operación del equipo deben todavía ser controlados por el SO. Es decir, cada uno de los dispositivos que componen el *hardware* debe estar bajo el control del SO, deben iniciar su actividad cuando este lo dispone, y al concluirla, este debe ser quien reciba el mensaje y discrimine la acción a seguir.
2. Administración de los recursos: el SO no solo debe ocuparse de controlar los recursos sino también de administrarlos. Todos los dispositivos de un sistema de cómputo están a disposición de los usuarios a través de un SO, que debe intentar maximizar el aprovechamiento de estos recursos sin provocar conflictos.
3. Provisión de un ambiente amistoso o amigable: o interfaz con el

usuario. Es un ambiente que deja al usuario hacer lo que quiere de varias maneras, todas combinaciones de funciones elementales sencillas y poderosas. También debe informar los problemas que encuentra de una manera simple y clara.

Análisis funcional de un SO

A fin de poder cumplir con las misiones enunciadas, un SO realiza las siguientes funciones:

1. Ejecución de programas: este servicio es básico para la operación de la computadora. Desde aquellos días en que se codificó el primer cargador automático hasta hoy, la función principal del SO es la de brindar un ambiente de ejecución de programas. Debe permitir: almacenar, cargar, ejecutar e interrumpir programas de usuarios.
2. Operaciones de E/S: parte de las instrucciones reservadas de una arquitectura destinada a procesar programas de muchos usuarios simultáneamente deben ser reservadas, por las razones apuntadas. Entre ellas, las instrucciones de E/S que controlan los dispositivos periféricos. En consecuencia, es el SO quien debe realizar las operaciones en cuestión, pero eso le implica una administración de las solicitudes que realizan los usuarios. Para dispositivos específicos pueden desearse funciones especiales: *rewind* de una cinta, blanqueo de la CRT, etcétera.
3. Manipulación de archivos: seguramente durante nuestro trabajo queremos leer o escribir archivos. También querremos crear, modificar o borrar archivos (o carpetas contenedoras) por su nombre. A través del SO, se canalizan servicios esenciales de almacenamiento, recuperación, memoria secundaria y apoyo a la ejecución de programas.
4. Detección de errores: los programas, ni aún los más perfectos están libres totalmente de errores. Por lo tanto, los SO deben proveer funciones de atención de errores que puedan tener lugar durante la ejecución de programas de usuario, evitando su propagación, colaborando con su depuración y permitiendo el uso continuado del equipo. El error puede ocurrir en la CPU, en el *hardware* de memoria, en los dispositivos de E/S o en el

programa del usuario. Para cada tipo de error, el SO toma la acción apropiada para asegurar el correcto y consistente cómputo. También existe una serie de funciones que están no para el usuario sino para la eficiente operación del sistema en sí mismo.

5. Adjudicación de recursos: un sistema operativo debe distribuir sus recursos entre los usuarios según su necesidad y las posibilidades reales que haya; sin entorpecer la ejecución de los programas que no requieren recursos asediados por otros, ni permitir situaciones de "abrazo mortal" entre los programas que compiten por los mismos recursos.
6. Contabilidad: una computadora es un instrumento destinado a brindar servicios de procesamiento de datos. Como sea que se facturen esos servicios (ya sea para pago o para llevar estadísticas), es de interés común saber cómo y por quiénes se utilizan los recursos del equipo.
7. Protección: el SO es responsable de brindar todas las funciones necesarias para la operación de los programas de usuarios, a la vez que debe cuidar que esas funciones no perjudiquen a otros usuarios o al propio SO. En combinación con el *hardware*, el SO protege la operación del equipo, previniendo la invasión de sectores de memoria no accesibles, la ejecución de instrucciones privilegiadas o el uso de recursos para los que no se cuenta con la debida autorización.

Modelo de estudio para los SO

El SO puede dividirse en un núcleo y una superestructura. El núcleo tiene un doble papel:

- Simula una actividad concurrente para los procesos de la superestructura.
- Proporciona la interfaz con el *hardware* recibiendo las señales del mundo externo.

El núcleo mantiene una lista de procesos actuales, lleva un control de aquellos que están libres para continuar y comparte el procesador entre ellos de acuerdo a alguna estrategia elegida (monitores, semáforos, etcétera)

implantada por un controlador cronológico que se introduce siempre que cualquier proceso cambia su condición y controla cómo se comparten sus recursos.

Algunos autores como Tanenbaum (*Structured computer organization*) hacen caso omiso de la división entre *hardware* y *software*, independizando su análisis de la implementación de un determinado nivel y concentrándose en su funcionalidad.

Aunque esta visión, poco frecuente antes de Tanenbaum, es muy útil y está muy difundida, en la construcción de los SO es necesario establecer la diferencia entre el hard y el soft, ya que el SO es el *software* que corre sobre el *hardware*. Además, la modularidad que requiere toda buena solución de problemas de diseño de *software*, también debe preocupar a quien diseñe un SO.

Esta cuestión planteada fue resuelta por Dijkstra en la construcción del SO, quien presenta un diseño jerárquico en cinco niveles (capas de creciente nivel de abstracción). Cada nivel representa una distinta funcionalidad, permitiendo la depuración del sistema completo en sucesivas aproximaciones.

Siguiendo a Dijkstra y su "modelo de la cebolla", estudiaremos los SO con funciones jerárquicas que van desde los niveles muy cercanos a la máquina misma hasta niveles más "virtuales".

Modelo de estudio de SO según diseño jerárquico de Dijkstra

El modelo está formado por capas concéntricas alrededor de un núcleo en forma similar a una cebolla. La parte interna del conjunto jerárquico de programas que forman el SO recibe el nombre de núcleo (*kernel*, en inglés).

Nivel 1: Núcleo

- Manejador de interrupciones de bajo nivel.
- Despachador.
- Semáforos.

Nivel 2: Manejo de memoria

Alocación de memoria.
Liberación de memoria.

Nivel 3: Manejador de procesador de alto nivel

Crea / destruye procesos.

Envía / recibe mensajes entre proceso.
Detiene procesos.
Comienza procesos.

Nivel 4: Manejador de E/S
Guarda el status de todos los dispositivos I/O/.
Programa I/O.
Inicia el proceso I/O.

Nivel 5: Manejo de información.
Crea / destruye archivos.
Abre / cierra archivos.
Lee / escribe archivos.

Figura 10.4: Modelo de Dijkstra

Hardware

Provee las funciones básicas para la ejecución de instrucciones del repertorio común, más "extracódigos" que permiten ampliarlo.

Núcleo del SO

El núcleo del SO es la capa más interna que corre directamente sobre el *hardware*. Es una cantidad mínima de código esencial para su funcionamiento que es usado intensivamente por todos los programas situados en niveles superiores. Dos procesos son "concurrentes" cuando se ejecutan en el mismo intervalo de tiempo. Es claro que el problema de la concurrencia entre procesos tiene que resolverse en el nivel más cercano al núcleo del SO. Las funciones del núcleo consisten en tomar el control del procesador y determinar cuándo y cómo se lo va a repartir entre diferentes usuarios. El núcleo del SO está formado en términos generales por tres subprogramas: 1) Manejador de interrupciones del procesador central; 2) Congelador / descongelador: tiene como función escoger un nuevo proceso para ser ejecutado (descongelar) y la operación inversa (congelar el que fue interrumpido); 3) Semáforos: su función es la de coordinar los diversos procesos que interactúan en el núcleo del SO, para que no choquen entre sí.

Aclararemos algunos términos básicos:

Manejo de interrupciones: un proceso entra en espera cuando pide efectuar alguna operación que sea muy lenta en comparación con las velocidades normales de procesamiento. El conjunto de operaciones que puede efectuar un procesador se divide en normales y privilegiadas; una operación privilegiada es aquella que cuando es ejecutada causa que el procesador entre en un estado especial que se llama de interrupción. Durante la interrupción, el procesador se detiene momentáneamente y pregunta si "puede" o debe seguir ejecutándolo. Esta pregunta consiste en que el procesador ejecute automáticamente un pequeño programa de "atención a la interrupción" que averigua la causa de ella y determina los pasos a seguir. El despachador "congela" un proceso almacenando en registros especiales de la CPU, los datos volátiles que resultaron de su ejecución, hasta antes de ser congelado. Cuando proceda, el despachador "descongelará" ese proceso copiando los contenidos de esos registros especiales de regreso en el acumulador y demás registros de trabajo

de la CPU, de modo que el proceso recién despertado pueda seguirse ejecutando como si no pasara nada.

Gestión de la memoria

Un proceso cualquiera puede estar en uno de varios estados: activo, en ejecución o residente en disco magnético. La función del manejador de memoria consiste en mantener un espacio allí, para todos los procesos activos dentro del sistema. Recordemos que cuando el recurso formado por la memoria principal se usa en la ejecución de un programa, no puede estar disponible simultáneamente para otro programa. Si hay otros programas que deseen acceder a la memoria principal, se los mantiene normalmente en espera en un área de almacenamiento de reserva. Una parte del almacenamiento principal se usa permanentemente para contener la porción residente del SO, el núcleo. El núcleo debe funcionar correctamente para mantener un aislamiento de cada programa de usuario que se está ejecutando y evitar ser adulterado. Existen varias maneras de manejo de memoria en un sistema de cómputo, de las cuales nos ocuparemos más adelante.

Gestión del procesador o procesos

Este programa del sistema (también conocido como manejador de alto nivel, *scheduler*) tiene como función determinar en qué orden y con qué criterios se les va a dar atención a los diversos usuarios de un sistema de cómputo. Existen tres módulos principales del manejador del procesador: *job scheduler*, *processor scheduler* y controlador de tráfico. El *job scheduler* está relacionado con la elección de cuál será la próxima tarea en comenzar y ser convertida en proceso. Tal vez la función más importante del *scheduler* sea la de convertir los programas del usuario en procesos para el sistema (esto es, para que el SO pueda hacer una eficiente planificación de cómo repartir los recursos de cómputos entre los diversos usuarios). Debe existir un proceso del SO que se encargue de averiguar qué procesos ya terminaron de ejecutarse, cuáles van a imprimir resultados, etcétera. Este proceso recibe el nombre de "controlador de tráfico". Una vez estudiadas las funciones mínimas de los procesos, queda aún por resolver el problema de la comunicación entre los que están en estado de ejecución y el mundo exterior.

Gestión de entradas y salidas

El manejador de E/S tiene como función principal atender los pedidos que los procesos en ejecución hacen sobre las unidades periféricas.

Esta atención requiere, la mayoría de las veces, una traducción lógica y física entre las diversas unidades involucradas. La parte física logra que aparatos diferentes entre sí puedan comunicarse aunque manejen códigos internos distintos, y la traducción lógica (que nos interesa) tiene como función virtualizar los pedidos de E/S y postergar su ejecución física hasta el último momento posible. La aplicación típica de este concepto hace que los pedidos de E/S sean desviados del/al disco magnético y no dependen de las limitaciones de los aparatos físicos de lectura o escritura. Existirán tantas impresoras virtuales en disco magnético como procesos activos haya en el sistema. Lo mismo se señala para el caso de la lectura: primero se leen los datos y se guardan en una lectora "virtual" en disco magnético, para que cuando el proceso pida un dato, este le llegue del disco y no de la unidad física de lectura. Este concepto recibe el nombre de SPOOLing (operación simultánea de periféricos en línea). Ventajas: al permitir una virtualización de las unidades de E/S de la computadora, esta se comporta como si tuviera varias de cada una y los procesos no tienen que esperar a que la impresora esté libre para seguir ejecutando; permite la reimpresión de múltiples copias del mismo resultado de un programa grabado previamente en disco, etcétera.

Gestión de información o programa de usuarios

La siguiente capa del SO se encarga de las transferencias físicas de información entre las unidades de E/S y los procesos en ejecución. Sus funciones son las siguientes: permitir a los usuarios el manejo libre y simbólico de prácticamente cualquier cantidad de información que desee almacenar, leer, imprimir, alterar o desechar. Se trata de un manejo libre porque en lo posible, tendrá el menor número de restricciones físicas o lógicas y simbólico, ya que el usuario no tendrá que preocuparse de los modos de acceso al disco magnético ni de otros detalles, que ya le han sido virtualizados por el manejador de E/S y simplemente hará referencia a su información por el nombre simbólico que decidió asignarle libremente. El sistema también puede almacenar la información por plazos indefinidos y recuperarla en cualquier momento, a la vez que maneja criterios de seguridad de acceso y protección.

Definiciones importantes

Sistema operativo monolítico

Se denomina así a los sistemas operativos que se ejecutan como un solo programa en modo kernel. Un sistema operativo con núcleo monolítico concentra todas las funcionalidades posibles (planificación, sistema de archivos, redes, controladores de dispositivos, gestión de memoria, etcétera) dentro de un gran programa. Con frecuencia se produce un sistema poco manejable y difícil de comprender.

Sistema operativo jerárquico o capas

El sistema operativo jerárquico divide el sistema operativo en pequeñas partes, de tal forma que cada una de ellas estuviera perfectamente definida y con un claro interface con el resto de los elementos. El primero de ellos fue denominado THE (*technische hogeschool eindhoven*), de Dijkstra. Cada capa o jerarquía tiene una apertura, conocida como puerta o trampa (trap), por donde pueden entrar las llamadas de las capas inferiores. De esta forma, las zonas más internas del sistema operativo o núcleo del sistema estarán más protegidas de accesos indeseados desde las capas más externas.

Máquina virtual

Es el sistema operativo que ofrece una interface a cada proceso, mostrando una nueva máquina con un sistema operativo diferente. Este sistema operativo separa dos conceptos que suelen estar unidos en el resto de sistemas: la multiprogramación y la máquina extendida. El objetivo del sistema operativo de máquina virtual es el de integrar distintos sistemas operativos dando la sensación de ser varias máquinas diferentes.

Su núcleo se denomina monitor virtual y tiene como misión llevar a cabo la multiprogramación, presentando a los niveles superiores tantas máquinas virtuales como se soliciten.

Sistema operativo monousuario

El sistema operativo monousuario es aquel que soporta a un usuario a la vez, sin considerar el número de procesadores que la computadora posea o el número de procesos que el usuario pueda ejecutar en el mismo instante de tiempo.

Sistema operativo multiusuario

El sistema operativo multiusuario es capaz de dar servicio a más de un usuario a la vez, ya sea por medio de varias terminales conectadas a la computadora o por medio de sesiones remotas en una red. En este caso no importa el número de procesadores en la máquina ni el número de procesos que cada usuario puede ejecutar simultáneamente.

Sistema operativo monotareas

El sistema operativo monotarea solo permite una tarea a la vez por usuario. Puede darse el caso de un sistema multiusuario y monotarea, en el cual se admiten varios usuarios al mismo tiempo, pero cada uno de ellos puede estar haciendo solo una tarea a la vez.

Sistema operativo multitareas

Un sistema operativo multitarea permite al usuario realizar varias tareas al mismo tiempo. Por ejemplo, puede estar editando el código fuente mientras compila otro programa, recibiendo correo electrónico y descargando algo de Internet.

Sistema operativo monoproceso

Un sistema operativo monoproceso es aquel capaz de manejar solamente un procesador de la computadora, de manera que si esta tuviese más de uno le sería inútil.

Sistema operativo multiproceso

En un sistema operativo multiproceso se hace referencia al número de procesadores del sistema (más de uno) y a la capacidad de usarlos todos para distribuir su carga de trabajo.

Este sistema trabaja de dos formas: simétrica o asimétricamente. Cuando se trabaja de manera asimétrica, el SO selecciona a uno de los procesadores el cual hará el papel de procesador maestro y servirá como pivote para distribuir la carga a los demás procesadores. Cuando se trabaja de manera simétrica, los procesos o hilos (*threads*) son enviados indistintamente a cualquier procesador disponible. Teóricamente ofrece una mejor distribución y equilibrio en la carga de trabajo. Se dice que un *thread* es la parte activa en memoria y corriendo de un proceso, lo cual puede consistir de un

área de memoria, un conjunto de registros con valores específicos, la pila y otros valores de contexto.

Sistema operativo de red

El sistema operativo de red es aquel que tiene la capacidad de interactuar con otros sistemas operativos en otras computadoras por medio de una red de datos, con el objeto de intercambiar información, transferir archivos, ejecutar comandos remotos, etcétera. El punto crucial de este sistema es que el usuario debe saber la ubicación de los recursos a los que desee acceder.

Sistemas operativos distribuidos

El sistema operativo distribuido abarca los servicios de los de red, logrando integrar recursos (impresoras, unidades de respaldo, memoria, procesos, unidades centrales de proceso) en una sola máquina virtual para que el usuario acceda en forma transparente. Evita que el usuario tenga que saber la ubicación de los recursos, los conoce por nombre y simplemente los usa.

El sistema integrador de los microprocesadores que hace ver a varias memorias, procesadores, y demás recursos como una sola entidad en forma transparente, se llama sistema operativo distribuido. Las razones para crear o adoptar sistemas distribuidos se dan por dos razones principales: por necesidad o porque se desea tener más confiabilidad y disponibilidad de recursos. En el primer caso, tenemos, por ejemplo, el control de los cajeros automáticos.

Gestión de memoria

Una de las principales funciones, y más compleja, que tiene un SO es el manejo efectivo de la memoria. La gestión de memoria de un SO se encarga de asignar y reclamar porciones de memoria principal en respuesta a las peticiones de los usuarios y de otros módulos del SO de acuerdo con los objetivos de la gestión de los recursos en un sistema particular. Los procesos están creados y cargados en memoria principal, en respuesta a algún tipo de planificación. Se les asigna una cantidad de memoria en un instante dado y se libera cuando los objetos residentes terminan. El objetivo de la gestión de memoria es suministrar su uso eficiente, minimizando la cantidad de memoria desaprovechada. Además, el gestor de memoria debe suministrar protección aislando los diferentes espacios de direcciones y facilitando la cooperación entre los

procesos, permitiendo el acceso a código y datos compartidos. Las políticas de manejo de memoria elegidas pueden estar influenciadas por el deseo de obtener módulos pequeños y simples o por la necesidad de incrementar flexibilidad y eficiencia.

Técnicas de manejo de memoria

Asignación única contigua

Es un esquema muy simple de alocación de memoria, asociado con pequeñas computadoras. En dichos sistemas no es posible la multiprogramación. La memoria está dividida conceptualmente en tres regiones contiguas: una porción, permanentemente alocacionada para el SO. El espacio remanente está disponible para la tarea simple que está siendo procesada. El proceso en ejecución utiliza una porción de la memoria alocacionada y deja sin usar el remanente.

Figura 10.5: Asignación única contigua de la memoria

Toda la memoria es alocacionada a cada proceso. Su única ventaja es la simplicidad, no requiere mucha experiencia para entender su manejo. En contraposición, sus desventajas son las que se mencionan: pobre utilización de memoria y procesador y los trabajos del usuario están limitados al tamaño de memoria disponible.

Manejo de memoria por particiones

Es una de las formas más simples para soportar la multiprogramación, consiste en dividir la memoria en varias secciones fijas y asignar a cada una de ellas un usuario o proceso activo. En cada partición se asigna un espacio de direcciones para un proceso.

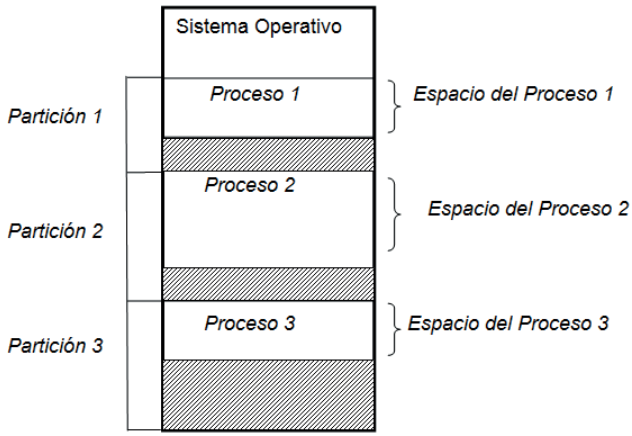


Figura 10.6: Asignación por particiones fijas

Un problema a resolver por el gestor del manejador de memoria es asegurar que ningún usuario intervenga en el área de memoria asignada a otro. Para eso se utilizan interrupciones de protección. El sistema puede ser de particiones estáticas o dinámicas. En estas últimas las particiones son creadas durante la ejecución del proceso, de modo de hacer coincidir el tamaño de las particiones con el de los procesos. En las primeras, toda la memoria es asignada antes de la ejecución de los procesos. El manejo de memoria consistiría entonces, en llevar un control de qué particiones están asignadas a cuáles procesos, para poder liberar particiones cuando los procesos residentes en ellas terminen o cambien y ofrecer particiones libres a procesos que soliciten atención.

Ventaja: permite la multiprogramación.

Desventaja: deja lugares libres en la memoria que, al ser de tamaño fijo, no pueden ser utilizadas más que por procesos de longitud igual o mayor.

Manejo por particiones relocalizables

Una solución al problema de fragmentación (expuesto previamente) es combinar periódicamente todas las áreas libres en un área contigua. Esto puede hacerse moviendo los contenidos de las particiones alocacionadas, de modo de que se vuelvan contiguas. Este proceso se llama "compactación". Las celdas en realidad no se mueven, sino que sus contenidos se copian de un lugar a otro. Este esquema es más flexible que el anterior, aunque más costoso, ya que mover una partición implica un proceso complejo de relocalización. Al relocalizar un proceso, las referencias a todas las direcciones tienen que ser alteradas (sumándoles el correspondiente desplazamiento), a fin de que el proceso pueda seguir ejecutando como antes.

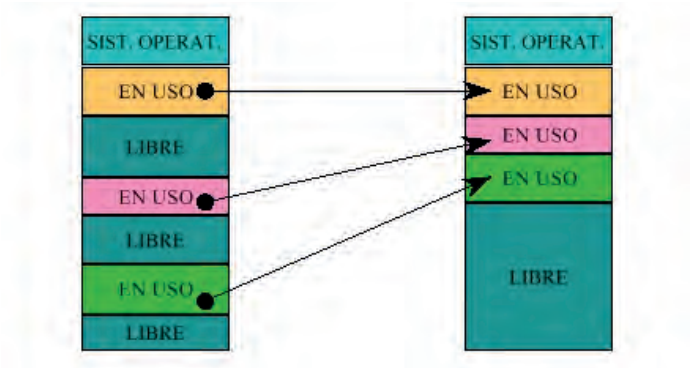


Figura 10.7: Asignación por particiones fijas

Ventajas:

- Elimina el problema de fragmentación y hace posible alocacionar más particiones.
- Permite un mayor grado de multiprogramación y, por lo tanto, un incremento de la utilización del procesador y la memoria.

Desventajas:

- El *hardware* de relocalización incrementa el costo de la computadora y puede reducir un poco la velocidad.
- El tiempo de compactación puede ser considerable. memoria.

- Alguna memoria puede seguir siendo inutilizada porque aunque esté compactada, el área libre puede ser menor que el tamaño de partición necesitado.
- El tamaño de particiones de los procesos está limitado al tamaño de la memoria física.

Manejo de memoria por paginación

Debido a los costos que representaba la relocalización, se inventó otro sistema más ágil y eficiente, llamado "paginación". Consiste en dividir los procesos en fragmentos de longitud fija llamados "páginas", que se almacenan en áreas de igual tamaño en memoria, llamados "bloques". Luego, proveyendo las adecuadas facilidades de *hardware* para mapeo, cualquier página puede ser colocada en cualquier bloque. Las páginas permanecen lógicamente contiguas, pero los correspondientes bloques no son necesariamente contiguos (físicamente). Para que el *hardware* ejecute el mapeo, desde el espacio virtual de direcciones a memoria física, debe haber un registro separado para cada página.

Al conjunto de estos registros se les llama *page map table*. Pueden ser tanto registros especiales del *hardware*, como una sección reservada de memoria física. Se usa un mecanismo de traducción de direcciones para hacer corresponder a cada página virtual su correspondiente página física. Obviamente, el tamaño de la página tiene un efecto sustancial en la utilización de este esquema. Si la página es demasiado grande, las desventajas pueden asimilarse al manejo por particiones relocalizables. Si es muy pequeña, se necesita tener una gran cantidad de registros de mapeo de páginas, lo cual encarece el costo del sistema.

Como resultado, los sistemas típicos de paginación han tenido páginas entre 1 KB y 4 KB.

Figura 10.8: Paginación

Ventajas: el esquema de memoria paginado elimina la fragmentación y hace posible acomodar el espacio de direcciones para más procesos simultáneamente. Permite mayor grado de multiprogramación, lo cual resulta en una optimización del uso de la memoria y el procesador.

Desventajas: el hard requerido (de mapeo de memoria) usualmente incrementa el costo de la computadora y al mismo tiempo es más lento.

Se debe usar memoria para tablas, tal como la PMT. Debe ocuparse tiempo del procesador para mantener y actualizar estas tablas. Como en los casos anteriores, el espacio de direcciones de procesos está limitado al tamaño de la memoria física.

Paginación por demanda

En los métodos anteriores, un proceso no podía correrse a menos que hubiera suficiente memoria disponible para cargar su espacio de direcciones completo. Esta restricción, a menudo provocaba la existencia de áreas libres en memoria, aun cuando había procesos esperando para ser cargados y ejecutados. Un método para trabajar con una memoria muy grande, es usar el SO para simular que se cuenta con ella. Como esta memoria es solo una ilusión se le denomina “memoria virtual”, es decir, se puede comenzar a

ejecutar un proceso cuando tan solo una parte del él está cargado en memoria e ir cargando a tiempo de ejecución las páginas que se requieran. Cuando un proceso pide una página que no está residente en memoria, el SO lo sabe por medio de una interrupción (atendida por el manejador de interrupciones del núcleo del SO). Este determina la causa de la interrupción (en este caso, interrupción por página) y copia la información solicitada (que está en disco) en un bloque libre de memoria. En este caso, se extiende la PMT, agregando un bit de status. (Y = la página está en memoria física, N = la página no se encuentra). Si el *hardware* de mapeo de direcciones encuentra una entrada en la PMT con status N, genera una interrupción por página. El SO debe procesar esta interrupción, cargando la página requerida y ajustando la información correspondiente en la PMT.

Existen algunas complicaciones. Cuando todos los bloques de memoria están en uso, ¿dónde se puede ubicar una página nueva? Esto requiere una técnica llamada *page swapping* o *page replacement*. La única forma de cargar una página más en estos casos es, primero, removiendo alguna página presente en memoria, la página se copia en un dispositivo de memoria secundaria, antes de cargar la nueva página. Los algoritmos de reemplazo han sido objeto de considerables estudios e investigaciones.

Ventajas: se elimina la fragmentación y no es necesaria la compactación, más memoria virtual, uso más eficiente de la memoria y un grado de multiprogramación no limitado por el tamaño de la memoria física.

Desventajas: costo de hard, complejidad, insume más tiempo del procesador atendiendo el problema de *page swapping*, se requieren más tablas para el manejo de las interrupciones por página, etcétera.

Segmentación

Un segmento puede definirse como un grupo lógico de información, tal como subrutinas, *arrays* o áreas de datos. El espacio de direcciones de cada proceso consiste en una colección de segmentos. La segmentación es una técnica de manejo de esos segmentos.

En un ambiente de segmentación, todas las referencias al espacio de direcciones requieren dos componentes, a saber:

- Un especificador de segmento.
- La locación dentro del segmento.

Un espacio de direcciones segmentado puede ser implementado usando *hardware* de mapeo de direcciones similar a la utilizada para la memoria por paginación. (*segment map table*, SMT). Esta tabla indica la dirección de cada segmento de memoria. La conversión de las dos partes de las direcciones segmentadas en memoria física lineal se hace automáticamente durante la ejecución. La mayor diferencia entre un segmento y una página es que un segmento es una unidad "lógica" de información, visible al programa del usuario y es de tamaño arbitrario. Una página es una unidad física de información invisible al programa del usuario y es de tamaño fijo, por ejemplo, 4 KB.

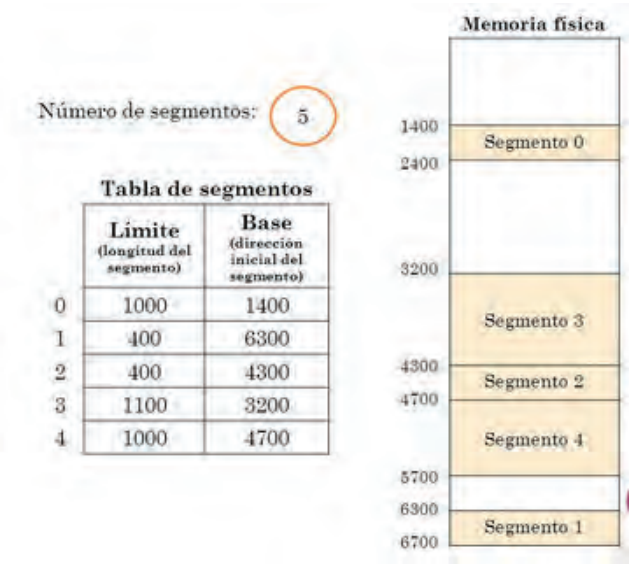


Figura 10.9: Paginación

Ventajas: elimina la fragmentación. Moviendo adecuadamente los segmentos, el espacio de memoria fragmentada puede combinarse en una única área libre; provee memoria virtual: guardando solo los segmentos activos usados; permite segmentos que crecen dinámicamente, además de compartir segmentos.

Desventajas: existe dificultad en el manejo de segmentos de tamaño variable en memoria secundaria; el tamaño máximo de un segmento está limitado

por el tamaño de la memoria principal; el costo de hard y la complejidad siguen aumentando.

Manejo de memoria por segmentación - Paginación por demanda

Una forma de obtener los beneficios de la segmentación y remover sus inconvenientes, consiste en combinar los mecanismos de la segmentación y la paginación. En lugar de tratar cada segmento como una entidad única contigua, cada uno puede ser dividido en páginas. Manipulando físicamente estas páginas en lugar de los segmentos, los problemas de manejo de almacenamiento secundario, limitación del tamaño de los segmentos y complejidad, pueden suprimirse.

El mapeo de las direcciones virtuales a las direcciones físicas está acompañado por una serie de tablas que se indican a continuación:

- SRT (*segment table register*). Indica la locación y el largo de la *segment map table* corriente.
- Cada entrada de la SMT indica la locación y el largo de la PMT (*page map table*).
- Cada entrada de la PMT indica el correspondiente número de bloque para cada página de ese segmento.

  Es el más general y flexible, pero de mayor costo de hard y complejidad.

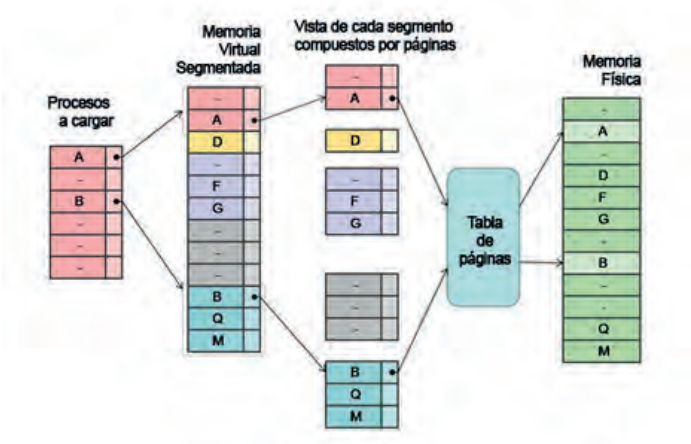


Figura 10.9: Segmentación

INGENIERÍA DE *SOFTWARE*

Muchos particulares y compañías todavía desarrollan el *software* de forma muy arriesgada. Algunos profesionales siguen sin conocer o utilizar los métodos modernos y como consecuencia, la calidad del *software* sigue en entredicho.

Actualmente, el *software* ha superado al *hardware* incluso en un alto porcentaje de casos, como la clave del éxito de muchos sistemas basados en computadoras; el *software* y la adecuada organización de las bases de datos, son el factor que marca la diferencia. El diseño de un producto de *software* "amigable a los humanos" lo diferencia de otros productos similares.

Concepto de *software*

Según Pressman (2011), el *software* se define de la siguiente manera:

- Instrucciones (programas de computadora) que cuando se ejecutan proporcionan la función y el rendimiento deseados.
- Estructuras de datos que permiten a los programas manipular adecuadamente la información.
- Documentos que describen la construcción y uso de programas.

Por su parte, Sommerville (2011) presenta la definición que sigue:

Se trata de programas de computadora y documentación asociada donde los productos de *software* pueden ser:

- Productos genéricos (desarrollados para clientes diversos).
- Productos hechos a medida (desarrollados a pedido de un cliente particular según un requerimiento específico).

Importancia del *software*

En los comienzos de la informática (durante las tres primeras décadas), el principal desafío era el desarrollo del *hardware*, de forma de poder reducir el costo de procesamiento y almacenamiento de datos. Durante la década de los 80, los avances en microelectrónica dieron como resultado mayor potencia de

cálculo y reducción de los costos. El problema actual es mejorar la calidad de las soluciones que se implementan con el *software*.

El *software* es el mecanismo que facilita la utilización, explotación y optimización del gran potencial disponible del *hardware* moderno. Es parte de un sistema automatizado hecho por el hombre y controlado por una o varias computadoras.

Un sistema se compone de los siguientes elementos:

- *Hardware*: CPU, discos, impresoras, etcétera.
- *Software*: sistema operativo, bases de datos, programas de aplicación, etcétera.
- Personas: proveen y/o consumen lo que produce el sistema.
- Datos: información que se mantiene por un período de tiempo.
- Procedimientos: políticas e instrucciones para operar el sistema.
- Documentación: manuales, formularios y otros modelos que describen en sistema.

Figura 11.1: Sistema

Las economías de los países desarrollados y en vías de desarrollo dependen, en gran parte, del *software*, cada vez más sistemas son controlados por este y se ha convertido en la *alma mater* de muchos sistemas productivos.

Está presente en sistemas de todo tipo: transportes, sanidad, telecomunicaciones, militares, procesos industriales, entretenimiento, entre muchos más, cubriendo todas las actividades humanas. El gasto en el desarrollo de *software* representa un alto porcentaje del PBI de todos los países.

Problemas en el desarrollo del *software*

Tanto las compañías como los profesionales particulares todavía desarrollan el *software* de forma muy arriesgada. Algunos profesionales siguen sin conocer o utilizar los métodos modernos, lo cual afecta la calidad del *software* y sus aplicaciones quedan en situaciones críticas presentando las siguientes características:

- Aplicaciones escritas hace más de 20 años han sufrido varias generaciones de cambios.
- Nadie tiene un conocimiento detallado sobre la estructura interna de las aplicaciones de ingeniería.
- Sistemas empotrados a veces se comportan de forma inexplicable.

Según Pressman (2011), la crisis del *software* se transformó en una "aflicción crónica" donde se deben afrontar los problemas, considerando los aspectos de fondo:

- La planificación y estimación de costos precisos.
- La "productividad" de la comunidad de *software* en correspondencia con la demanda.
- Calidad aceptable del *software*.

Entre las dificultades en el *software* se pueden citar las siguientes:

- Falta de metodología adecuada en el análisis de requisitos.
- Insatisfacción del cliente con el sistema terminado.
- Calidad del *software* cuestionable.
- Falta de adecuada importancia de la prueba sistemática y completa.
- La tarea de mantenimiento se lleva la mayor parte del dinero invertido en el soft.

LA CLAVE PARA RESOLVER LOS PROBLEMAS ES DAR UN ENFOQUE DE INGENIERÍA AL DESARROLLO DEL *SOFTWARE*

Importancia de la Ingeniería de *Software*

La necesidad de las empresas de producir productos de *software* más competitivos, a mejores costos y con un fuerte componente de servicios posventas.

La necesidad, cada vez más imperiosa, de desarrollar *software* sin fallas en tiempos reducidos y con presupuestos preestablecidos.

Por estas razones, consultores y académicos se deben esforzar en lo siguiente:

- Alinear objetivos del trabajo con tecnología.
- Poner énfasis en la formalidad del proceso de desarrollo.
- Trabajar con sistemas documentados.
- Realizar mediciones de cada actividad y análisis estadístico de los procesos.
- Madurez y mejora continua.
- Buscar soluciones integrales en paradigmas específicos.

La Ingeniería de *Software*

La ingeniería de *software* (IS) es una disciplina de la ingeniería que comprende todos los aspectos de la producción de *software*:

- Desde la especificación inicial al mantenimiento del sistema.
- Administración y gestión del proceso de producción.

Los ingenieros de *software* adoptan un enfoque sistemático para llevar a cabo su trabajo y utilizan los métodos, herramientas y técnicas necesarias para resolver el problema planteado.

1. IEEE – 610-12 (1990): *Aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software.*
2. Richard Fairley: *Disciplina tecnológica y de administración que se ocupa de la producción y evolución sistemática de productos de software que son desarrollados y modificados dentro de los tiempos y costos estimados.*
3. Barry Bohem: *Es la aplicación práctica de conocimiento científico al diseño y construcción de programas... y la documentación asociada requerida para su desarrollo, operación y mantenimiento.*

Ciencia de la computación e Ingeniería *software*

La ciencia de la computación se refiere a las teorías y los fundamentos subyacentes en los sistemas de computación.

La ingeniería del *software* trata los problemas prácticos del desarrollo de *software*.

Las teorías de la ciencia de la computación no son suficientes para desarrollar *software*, al menos cuando el sistema tiene suficiente envergadura.

Ingeniería *software* e ingeniería de sistemas

La ingeniería de sistemas concierne a todos los aspectos del desarrollo de sistemas basados en cómputo, que incluyen *hardware*, *software* y el proceso de ingeniería. La Ingeniería de *Software* es solo parte de este proceso.

La Ingeniería de *Software* define procesos para producir "sistemas de *software*".

Al ser el *software* muchas veces la parte más importante del sistema, las técnicas de ingeniería del *software* se aplican en el proceso de ingeniería de sistemas.

Elementos que distinguen la ingeniería ***software***

- Producción y evolución.
- Métodos sistemáticos.
- Disciplina tecnológica y de administración.
- Cuantificación.
- Estimación de tiempos y costos.
- Desarrollo, operación y mantenimiento.

Según Fritz Bauer, la Ingeniería de *Software* es el "establecimiento y uso de principios de ingeniería robustos, orientados a obtener *software* económico que sea fiable y funcione de manera eficiente sobre máquinas reales" (pág.).

Conocimientos del ingeniero de *software*

Un ingeniero de *software* debería tener los siguientes conocimientos y prácticas:

- Conocer los principios teóricos de representación y computación.
- Aplicar los métodos formales: lógica, matemática discreta, estadística, simulación.

- Usar notaciones de modelización, especificación, diseño, programación.
- Adoptar un enfoque sistemático y organizado en su trabajo y utilizar las herramientas y técnicas más apropiadas dependiendo del problema a resolver, las restricciones del desarrollo y los recursos disponibles.

La producción de *software* requiere de una combinación de recursos de metodología y tecnología, además del conocimiento de técnicas de administración de proyectos.

Metodología: de documentación, análisis, especificación, diseño, implementación y prueba.

Tecnología: sistemas operativos, lenguajes, herramientas CASE, bases de datos, notaciones gráficas, sistemas generadores de interfaces, bibliotecas de código.

Administración de proyectos: planificación, análisis de riesgos, control de calidad, seguimiento de proyectos, control de subcontratistas, entre otros.

Disciplinas de la Ingeniería de *Software*

- Requerimientos del *software*.
- Diseño del *software*.
- Construcción del *software*.
- Prueba del *software*.
- Mantenimiento del *software*.
- Gestión de la configuración del *software*.
- Gestión de la ingeniería del *software*.
- Proceso de ingeniería del *software*.
- Herramientas y métodos de la ingeniería del *software*.
- Calidad del *software*.

Evolución del *software*

La Ingeniería de *Software* es una familia de disciplinas técnicas y no técnicas que evoluciona continuamente. Deriva sus principios, notaciones y técnicas de distintas ramas de la lógica, matemática, lingüística, administración y psicología.

Características del proceso de evolución

- Expansión de su esfera de competencia.
- 2) Aumento en el nivel de formalización de procesos y productos.

- Mecanización de operaciones.
- Aumento en el nivel de conocimientos y profesionalismo que se exige de los ingenieros.

Durante los primeros años de desarrollo de las computadoras, el *hardware* sufrió continuos cambios, mientras que el *software* se contemplaba simplemente como un añadido. La programación se realizaba sin metodologías sistemáticas adecuadas, en forma "casera". El desarrollo de *software* se realizaba sin ninguna planificación. Lo normal era que el *hardware* fuera de propósito general. El *software* se diseñaba a medida para cada aplicación y tenía poca distribución; como producto, recién comenzaba. La misma persona lo escribía, ejecutaba y depuraba. Debido a este entorno personalizado del soft, el diseño estaba en la mente de alguien y la documentación no existía.

El mejor rendimiento del *hardware*, la miniaturización y la disminución del costo, han dado lugar a sistemas informáticos más sofisticados.

En la segunda generación, la multiprogramación, los sistemas multiusuario y las técnicas interactivas, permitieron el desarrollo de un gran número de aplicaciones y nuevos niveles de sofisticación en el hard y soft. Se estableció el *software* como producto y los "comercios de soft", abarcando un mercado multidisciplinar. Al mismo tiempo nacen las actividades correspondientes al mantenimiento del *software*, que comenzaron a absorber recursos en gran medida. La naturaleza personalizada de la mayoría de los programas los hacía imposibles de mantener.

Así comenzó la llamada "crisis del *software*" que motivó la aparición de metodologías sistemáticas de desarrollo de soft.

En la tercera generación, el procesamiento distribuido, las redes, la creciente demanda de acceso "instantáneo" a los datos, la llegada de las microcomputadoras, y demás, supuso una fuerte presión sobre los constructores de soft. Mientras que las compañías de *software* de la segunda generación vendían cientos o miles de copias de sus programas, las de la tercera venden millones de copias. Actualmente, las técnicas de desarrollo de *software* están cambiando la forma en que se construyen los programas.

Se puede decir que los problemas asociados con el *software* de computadoras, que aún continúan intensificándose son, entre otros:

- La creciente sofisticación del *hardware* ha dejado desfasada la capacidad de construir *software* que pueda explotar su potencial.

- La capacidad de construir nuevos programas no da abasto con la demanda.
- La capacidad de mantener los programas existentes se ve amenazada por el mal diseño y el uso inadecuado de recursos.

La respuesta a la crisis del *software* ha sido la adopción de prácticas de Ingeniería de *Software* por parte de muchas empresas.

En resumen:

Década del 60

- Dificultades para desarrollar los primeros sistemas de *software*.
- Los problemas desbordaban los recursos técnicos de una clase profesional formada por matemáticos, físicos, y otros.
- Conferencias de la OTAN de carácter fundacional para la Ingeniería de *Software* (Garmish, 1968 y Roma, 1969) para buscar la raíz de los problemas.

Década del 70: época de fundación y consolidación

- Se popularizaron los conceptos de diseño, modularidad, acoplamiento entre módulos, encapsulamiento de información.
- Se refinaron las técnicas de programación y diseño.
- Avanzaron los lenguajes de especificación y de programación.
- Se definieron "ciclos de vida" para los productos de *software*.
- Se reconoció la necesidad de definir técnicas de planificación y métodos de estimación cuantitativos.
- Se establecieron los métodos básicos de análisis de requerimientos y formalización de especificaciones de sistemas.
- Las técnicas de prueba de sistemas comenzaron a consolidarse sobre bases estadísticas.

Década de los 80

- Se caracterizó por la expansión cultural y técnica: notaciones de especificación, lenguajes de programación disponibles, variedad de modelos, técnicas de análisis y diseño.
- Se comenzó a utilizar en forma corriente el término "ingeniero de *software*".
- Se multiplicaron los congresos científicos.

- Se desarrolló una industria editorial específica y se incorporaron cursos en las carreras universitarias y posgrados.
- Surgieron los ambientes de programación y herramientas especializadas, los editores de notaciones gráficas, sistemas de bibliotecas, correo electrónico, y demás.
- Se consolidó la industria *Computer Aided Software* Engineering (CASE).
- Se popularizaron las métricas y técnicas de estimación y de Ingeniería de *Software*.

Década de los 90

- Transformaciones: de la programación artesanal a la producción masiva, del individuo (como constructor de sistemas) a la organización, de los métodos manuales a los métodos asistidos por computadora.
- El *software* pasa a ser parte integral de toda clase de productos y servicios.
- La industria de *software* se ve sometida a presiones: reducción en el ciclo de vida de productos, personalización de productos, costos, calidad y certificación de procesos.
- Necesidad de garantizar seguridad, confiabilidad o la responsabilidad legal por daños y perjuicios ocasionados por un producto.
- Se agrega la preocupación por formalizar, medir y optimizar el proceso de desarrollo y los controles de calidad.

Década del 2000

- Aplicación de buenas prácticas genéricas para su adaptación a problemas particulares (procesos de desarrollo y ciclos de vida genéricos y flexibles, lenguajes de modelado, técnicas de modelado basadas en el uso de patrones y frameworks,...).
- Se consolida la gestión del *software* a través de planificación de proyectos, estimaciones y gestión de recursos, aseguramiento de la calidad, gestión de la configuración y de documentación.
- Extenso desarrollo de la ingeniería de requerimientos.
- Desarrollo rápido de aplicaciones.
- Reutilización de *software*.
- Desarrollo basado en componentes.
- Desarrollo de aplicaciones web, frameworks de aplicaciones.

El producto *software*

La Ingeniería de *Software* produce productos de *software*. Un producto de *software* es un sistema de *software* distribuido a un cliente junto con su documentación (programas de computadora, procedimientos, documentación asociada, datos relativos a la operación de un sistema de cómputos.)

Los productos de *software* se clasifican de la siguiente manera:

Software a medida: desarrollado para un cliente particular bajo un contrato.

Software genérico: desarrollado para ser vendido a un mercado abierto.

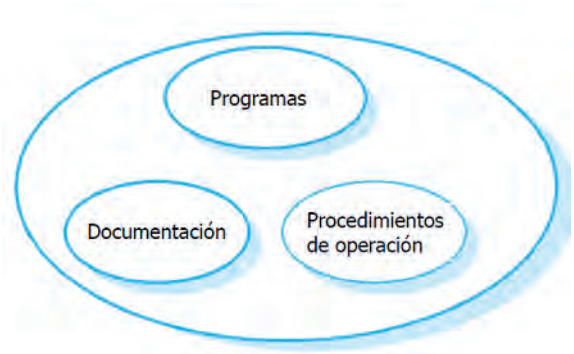


Figura 11.2: Componentes del *software*

Según Pressman (2011), el *software* se puede definir como las instrucciones (programas de computadora) que cuando se ejecutan proporcionan la función y el rendimiento deseados, estructuras de datos que permiten a los programas manipular adecuadamente la información y los documentos que describen la construcción y el uso de programas.

Un producto *software* es un producto particular, a diferencia de otros productos de ingeniería, que presenta las siguientes características:

- Es lógico y no físico.
- Es maleable.
- Se desarrolla, no se fabrica. Su construcción requiere principalmente de creatividad humana, más que de manufactura.
- No se estropea (aunque se vuelve obsoleto).

- Aunque la industria tiende a ensamblar componentes, la mayor parte del *software* que se construye es a medida.

Un *software* bien diseñado presenta los siguientes atributos:

• Mantenible: capaz de evolucionar según las necesidades de cambio de los clientes.

• Seguro: no produce daños incluso bajo un fallo del sistema.

• Eficiente: no desperdicia los recursos del sistema (memoria, procesador, disco).

• Amistoso: buena interfaz.

• Bien documentado.

Aplicaciones de *software*

- • *Software* de sistemas: colección de programas de servicio a otros programas (compiladores, sistemas operativos, etcétera).
- • *Software* de tiempo real: monitorea, analiza y controla eventos del mundo real a medida que ocurren (control de aviones, pronóstico del clima, etcétera).
- • *Software* empotrado (*embedded systems*): reside en la ROM de un producto y controla sus funciones. Maneja componentes de *hardware* (el producto puede ser un sistema de seguridad, señalización, etcétera).
- • *Software* de negocios: diseñado para procesar aplicaciones de negocios, herramientas de manejo de bases de datos (gestión de nóminas, control de almacén, etcétera).
- • *Software* de ingeniería y científico: basado en algoritmos numéricos con altas necesidades de cómputos (Programas CAD, MATLAB, predicción meteorológica, etcétera).
- • *Software* de PC: se venden en la gran distribución para computadoras personales (procesadores de texto, hojas de cálculo, etcétera).
- • *Software* de inteligencia artificial: algoritmos no numéricos para resolver problemas complejos en forma declarativa, para los que no son adecuados el cálculo o análisis directo (sistemas expertos, reconocimiento de patrones —voz, imágenes, y otros—, agentes *software*.
- • *Software* WEB: desarrollado para aplicaciones WEB (*e-commerce*, correo electrónico, etcétera).

Paradigmas de la Ingeniería de *Software*

Capas de la Ingeniería de *Software*

La Ingeniería de *Software* como tecnología multicapa se puede ver de la siguiente manera:

Figura 11.3: Ingeniería de SW como tecnología multicapa.

Capa de enfoque de calidad: base de procesos de ingeniería. La Ingeniería de *Software* se basa en calidad aplicando las mejores técnicas para construcción de *software*.

Capa de proceso: conjunto de actividades y resultados asociados que sirven para construir un producto SW.

Capa de métodos: forma en que se construye el *software*. Los métodos corresponden a análisis de requisitos, diseño, construcción de programas, prueba.

Capa de herramientas: soporte automático o semiautomático para el proceso y los métodos. Herramientas CASE.

En el proceso de desarrollo de *software*, en las tareas que se requieren para construir *software* de alta calidad, la Ingeniería de *Software* abarca un conjunto de tres elementos clave: métodos, herramientas y procedimientos.

Los *métodos* indican cómo construir técnicamente el *software*. Incluyen métodos para planificación y estimación del proyecto, análisis de los requerimientos del sistema y del *software*, diseño de estructuras de datos, arquitectura de programas y procedimientos algorítmicos, codificación, prueba y mantenimiento.

Las *herramientas* suministran un soporte automático para los métodos. Existen herramientas para soportar cada uno de los métodos como las herramientas CASE. Cuando se integran las herramientas, de modo que la información creada por una herramienta pueda ser utilizada por la otra, se establece un soporte para el desarrollo del *software* llamado "Ingeniería de *software* asistido por computadora" (CASE). CASE combina *software* y bases de datos para crear un entorno de ingeniería de *software*.

Los *procedimientos* son el pegamento que junta los métodos y herramientas y facilita un desarrollo racional; definen la secuencia en la que se aplican los métodos, las entregas que se requieren, los controles para asegurar la calidad, etcétera.

Visión genérica de la Ingeniería de *Software*

El trabajo que se asocia a la Ingeniería de *Software* se puede dividir en tres fases genéricas, con independencia del modelo de proceso, área de aplicación, tamaño o complejidad del proyecto:

La fase de definición se centra en el *qué*. Se intenta identificar qué información ha de ser procesada, qué función y rendimiento se desea, que interfaces se establecerán, restricciones de diseño y criterios de validación. Se deben identificar los requisitos clave del sistema.

Incluirá algunos pasos específicos, que se detallan:

- Análisis del sistema: se define el papel de cada elemento de un sistema informático, asignando al *software* el papel que va a desempeñar.
- Planificación del proyecto de *software*: establecido el ámbito del *software*, se analizan los riesgos, se asignan los recursos, se estiman los costos, se definen las tareas y se planifica el trabajo.
- Análisis de requisitos: el ámbito establecido para el soft proporciona la dirección a seguir, pero es necesaria una información más detallada del ámbito de información y de la función del soft.

La fase de desarrollo se centra en el *cómo*. Se intenta descubrir la manera óptima de diseñar las estructuras de datos y la arquitectura del *software*, implementación de los detalles procedimentales, traducción del diseño a un lenguaje de programación, y cómo se realizarán las pruebas.

Incluye tres pasos concretos, que se indican seguidamente:

- Diseño del *software*: traduce los requisitos del *software* a un conjunto de representaciones (gráficas, tabulares, incluso lenguajes) que describen la estructura de datos, la arquitectura, los procedimientos algorítmicos y las características de la interfaz.
- Codificación: durante este paso, las representaciones de diseño son traducidas a un lenguaje, ya sea procedimental o no procedimental, dando como resultado instrucciones ejecutables para la computadora.
- Prueba del *software*: una vez que el *software* ha sido implementado, debe ser probado, para descubrir los defectos que puedan existir en la función, en la lógica y en la implementación.

La fase de mantenimiento se centra en el *cambio* que va asociado a la corrección de errores, a las adaptaciones requerida por la evolución del entorno del *software* y a las modificaciones debidas a los cambios de requisitos del cliente dirigidos a readecuar el sistema.

Se suelen encontrar tres tipos de cambios:

- Corrección: es muy probable que el cliente descubra defectos en el soft; el mantenimiento correctivo lo cambia para corregir defectos.
- Adaptación: con el paso del tiempo suele cambiar el entorno original (CPU, sistema operativo, periféricos) para el que se desarrolló el soft. El mantenimiento adaptativo consiste en modificar el soft para adecuarlo a los cambios de entorno.
- Mejora: el cliente puede descubrir a través del tiempo, fuentes adicionales que le podría interesar incorporar al *software*. El mantenimiento perfectivo amplía el *software* más allá de sus requisitos funcionales originales.
- Prevención: el mantenimiento preventivo, llamado reingeniería del soft hace cambios en los programas a fin de que se puedan mejorar más fácilmente.

Actividades de soporte

Estas fases se complementan con las actividades de soporte que no crean *software*, mejoran su calidad y facilitan su desarrollo. Se aplican a lo largo de todo el proceso del *software*.

Se enumeran ejemplos de actividades de soporte:

- Documentación.
- Gestión de configuración.
- Seguimiento y control del proyecto de *software*.
- Revisiones técnicas formales.
- Garantía de la calidad del *software*.
- Gestión de reutilización.
- Mediciones.
- Gestión de riesgos.

Proceso de desarrollo de *software*

Para entender este apartado vamos a valernos de algunas definiciones que se detallan a continuación.

Proceso: serie de operaciones usadas en la creación de un producto.

Proceso de software: conjunto estructurado de actividades que tienen que ser realizadas para producir un producto de *software* de alta calidad.

Proceso de software (vinculado al concepto de ciclo de vida): proceso que se sigue para construir un producto de *software* desde la concepción de una idea, hasta la entrega y el retiro final del sistema.

Las actividades del proceso de desarrollo de *software* se describen en forma independiente, indicando rol y resultados, utilizan y producen documentos, se relacionan e interactúan de diferentes maneras conformando distintos modelos de procesos de desarrollo de *software* y, de acuerdo al modelo, una actividad puede jugar un rol preponderante o incluso pudiera no existir.

Existe una relación entre las actividades y los modelos de desarrollo.

En el proceso de desarrollo de *software* genérico (independientemente del modelo) se pueden distinguir las siguientes actividades:

Análisis de requerimientos:

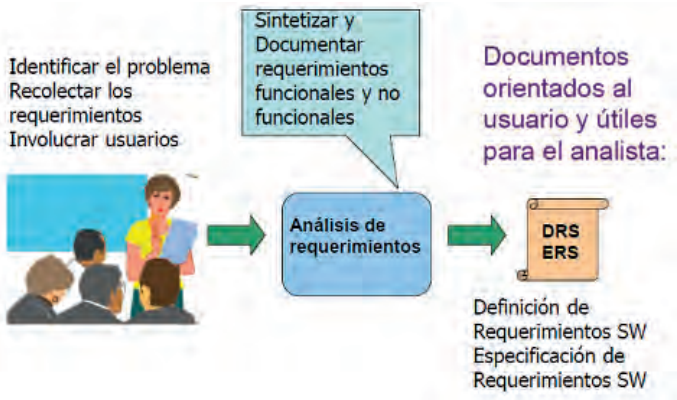


Figura 11.4: Análisis de requerimientos

Especificación:

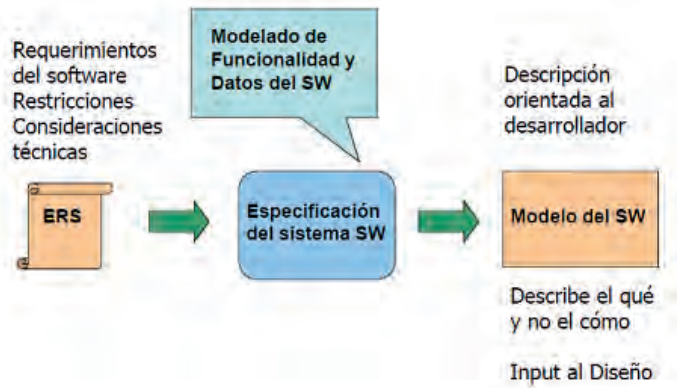


Figura 11.5: Especificación

Diseño arquitectura:

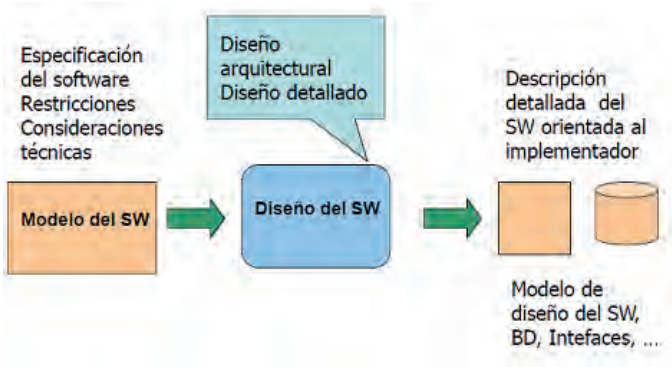


Figura 11.6: Diseño

Implementación de programas:

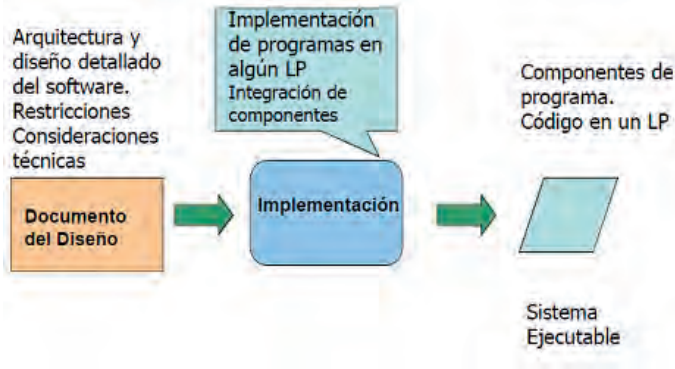


Figura 11.7: Implementación

Validación y verificación:

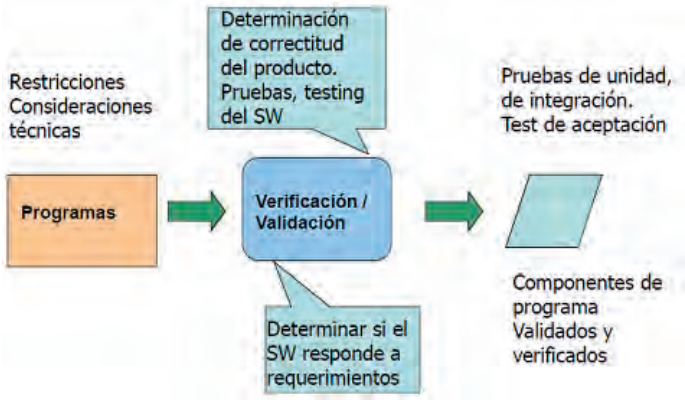


Figura 11.8: Validación y verificación.

Paradigmas de la Ingeniería de *Software*

La Ingeniería de *Software* está compuesta por una serie de pasos que abarcan los métodos, las herramientas y los procedimientos. Estos pasos se denominan "paradigmas de la Ingeniería de *Software*". La elección del paradigma se lleva a cabo con la naturaleza del proyecto y la aplicación, los métodos y herramientas a usar y los controles y entregas requeridos.

Concepto de ciclo de vida

Se define como ciclo de vida al periodo de tiempo que comienza al concebir la idea de un nuevo sistema de *software*, y termina cuando este se retira y deja de funcionar. El proceso consta de un conjunto de actividades y tareas relacionadas, que al ejecutarse de forma conjunta transforman una entrada en una salida.

Un modelo de proceso, o paradigma de Ingeniería de *Software*, es una plantilla, patrón o marco que define el proceso a través del cual se crea *software*. Se utilizan ambos términos (proceso y ciclo de vida) para definir los modelos disponibles. Estos sirven para definir las actividades a llevarse a cabo en un proyecto de desarrollo de *software* como para lograr una congruencia entre los distintos proyectos de desarrollo de *software* en una misma organización.

Proporcionar puntos de control y revisión de las decisiones sobre continuar con el proyecto o no.

El ciclo de vida puede organizar las actividades de administración del proyecto, aumentando las probabilidades de que se aborden los temas en el momento adecuado.

Modelos de desarrollo de *software*

Los modelos de desarrollo definen la estructura de un proceso de desarrollo racional y controlable. No existe un modelo universal, son una guía respecto al orden en que deben adelantarse las actividades, se basa en el reconocimiento que el *software* tiene un ciclo de vida.

Podemos encontrar los siguientes:

Modelos secuenciales: lineal, cascada, RAD.

Modelos evolutivos: incremental, espiral, basado en reuso, prototipos.

Modelos formales: transformacional.

Modelo lineal o secuencial (1970)

Propuesto por Royce en 1970, este modelo refleja un desarrollo marcado por la sucesión escalonada de las etapas que lo componen. Es necesario terminar por completo cada etapa para pasar a la siguiente, lo cual resulta muy rígido porque cada fase requiere como elemento de entrada el resultado completo de la anterior.

Resulta apropiado para sistemas pequeños, sin previsión de evolución a corto plazo. El modelo prácticamente idéntico, que evita esta rigidez es el de cascada, que se expone a continuación.

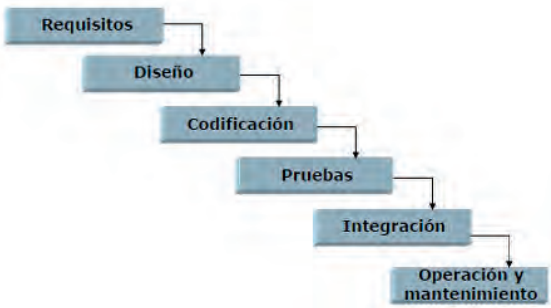


Figura 11.9: Modelo Lineal o Secuencial

Modelo en cascada

Ciclo de vida clásico llamado "modelo en cascada". Desarrollado por Boehm en 1981, sigue un enfoque ascendente y secuencial y una insistencia en la progresión lineal del desarrollo del *software* que progresa a través de las siguientes etapas:

Figura 11.10: Modelo en Cascada

Las actividades que abarca cada etapa son las que se detallan seguidamente.

Ingeniería y análisis del sistema: el *software* siempre es parte de un sistema mayor, por lo tanto, el trabajo comienza estableciendo los requisitos de todos los elementos del sistema y luego asignando algún subconjunto de estos requisitos al *software*. Este planteamiento del sistema es esencial cuando el *software* debe interrelacionarse con otros elementos, tales como *hardware*, personas y bases de datos.

Análisis de los requisitos del *software*: para construir los programas el analista debe conocer el ámbito de la información del *software*, así como la función, el rendimiento y la interfaz. Los requisitos, tanto del sistema como del *software*, deben ser documentados y revisados con el cliente.

Diseño: es un proceso que abarca distintos atributos de los programas: estructura de datos, arquitectura del *software*, detalle procedimental y caracterización de la interfaz.

Codificación: el diseño debe traducirse en una forma legible para la máquina. Si el diseño se realiza de manera detallada, la codificación es mecánica.

Este es el paradigma más antiguo y más ampliamente usado en la ingeniería del *software*. Presenta graves problemas, por lo cual, se cuestiona su aplicabilidad a todas las situaciones. Los proyectos reales raramente siguen el orden secuencial que propone el modelo, siempre existen iteraciones. Además, para el cliente es difícil plantear al principio todos los requisitos y, a veces, es difícil pretender que tenga paciencia para ver una versión operativa de sus programas al final del proyecto.

Dificultades del enfoque ascendente:

- No hay nada para mostrar al usuario hasta que todo esté terminado.
- Las fallas más triviales se encuentran al comienzo del período de prueba y las más graves al final.
- La eliminación de fallas suele ser extremadamente difícil durante las últimas etapas de prueba del sistema.
- La necesidad de prueba con la computadora aumenta exponencialmente durante las etapas finales de prueba.
- Dificultades de la progresión secuencial:
- El deseo de progreso ordenado de las etapas no es nada realista.
- Durante el tiempo que dura el proyecto, el usuario podría cambiar de parecer respecto a lo que debe hacer el sistema.
- Se apoya en técnicas anticuadas.
- Resulta apropiado para sistemas pequeños.

Modelo RAD

Este modelo sigue una adaptación a "alta velocidad" del modelo lineal secuencial, con equipos trabajando en paralelo, que aplican tecnología de componentes. RAD (*rapid application development*).

Es utilizado en proyectos grandes, es necesario contar con recursos suficientes para formar los equipos requeridos. Compromiso de colaboración entre desarrolladores y clientes para un tiempo de espera corto.

En todas las aplicaciones no es susceptible aplicar este modelo (no modularizables, bajo reuso de componentes), por lo cual, no es aplicable en sistemas con gran mantenimiento o cuando el grado de interoperatividad con programas ya existentes es alto.

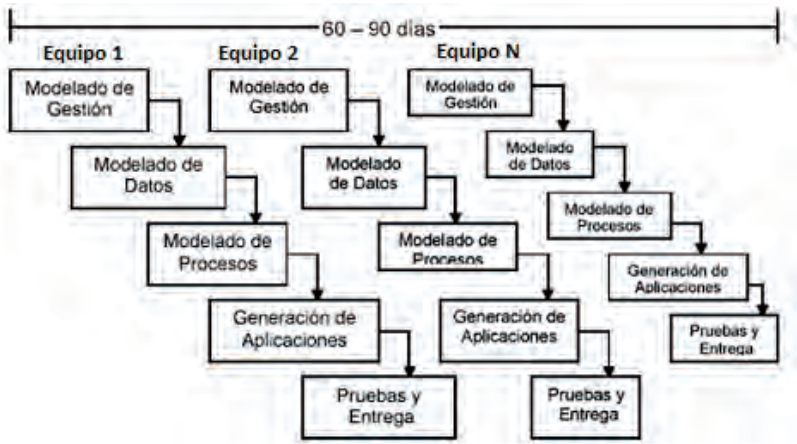


Figura 11.11: Modelo RAD

Modelos evolutivos

Este modelo presenta como principales características que se gestiona bien la naturaleza evolutiva del *software*, se adapta más fácilmente a los cambios introducidos a lo largo del desarrollo. Es iterativos: construye versiones de *software* cada vez más completas.

Se adapta bien a los cambios de requisitos del producto, fechas de entrega estrictas poco realistas y a especificaciones parciales del producto.

Modelos evolutivos: Modelo en espiral, Modelo incremental, Modelo de desarrollo basado en reuso de componentes.

Modelo en espiral (Bohem, 1988)

En este paradigma se añade el análisis de riesgos. El modelo define cuatro actividades principales:

1. 1) Planificación: determinación de objetivos, alternativas y restricciones. Incluye la recolección de requisitos y planificación del proyecto inicial, que se basa en los comentarios del cliente.
2. 2) Análisis de riesgos: análisis de alternativas e identificación/resolución de riesgos. Incluye el análisis de riesgos basado en

los requisitos iniciales, análisis de riesgos basados en la reacción del cliente. Se toma la decisión de seguir adelante o no.

3. 3) Ingeniería: desarrollo del producto de "siguiente nivel". Se construye un prototipo cero (inicial del *software*) y se va refinando en los niveles sucesivos.
4. 4) Evaluación del cliente: valoración de los resultados de la ingeniería en cada etapa.

Con cada iteración alrededor de una espiral (comenzando en el centro) se construyen sucesivas versiones del *software* cada vez más completas. Durante la primera vuelta, se definen los objetivos, las alternativas y las restricciones y se analizan e identifican los riesgos, el cliente evalúa el trabajo de ingeniería y sugiere modificaciones, basándose en los comentarios del cliente se produce la siguiente fase de planificación y análisis de riesgos. (En cada bucle de la espiral, este análisis se traduce en una decisión de "seguir adelante o no").

En la mayoría de los casos se sigue avanzando y ese camino lleva a los desarrolladores de soft a un modelo completo del sistema. Este paradigma es el enfoque más realista y completo para el desarrollo de *software* a gran escala.

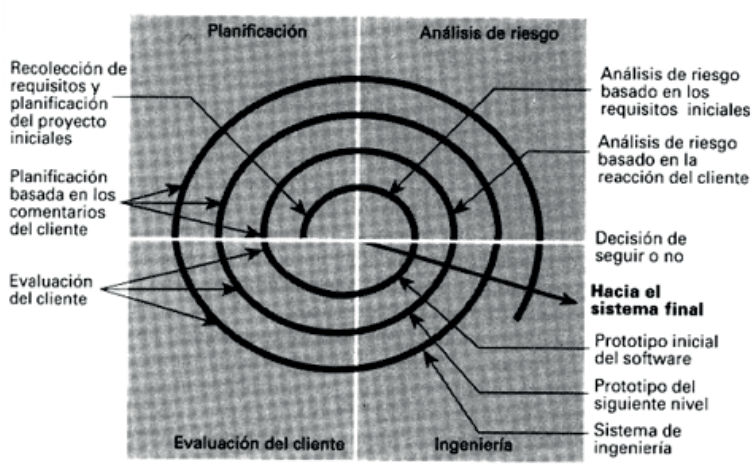


Figura 11.12: Modelo Espiral

Modelo incremental

El modelo incremental mitiga la rigidez del modelo en cascada, descomponiendo el desarrollo de un sistema en partes; para cada una de las cuales se aplica un ciclo de desarrollo.

Las ventajas que ofrece son las que se nombran seguidamente:

- El usuario dispone de pequeños subsistemas operativos que ayudan a perfilar mejor las necesidades reales del conjunto.
- El modelo produce entregas parciales en periodos cortos de tiempo y permite la incorporación de nuevos requisitos que pueden no estar disponibles o no ser conocidos al iniciar el desarrollo.

Es un modelo cuyas etapas consisten en incrementos expandidos de un producto de *software* operativo. Cada iteración devuelve un "incremento" del *software*. Un incremento es un producto operacional del *software*, el primer incremento suele ser un núcleo básico desarrollado con base en los requerimientos centrales. Muchas funciones suplementarias se dejan para después.

Se evalúa (el cliente, por ejemplo) el producto entregado y como resultado se desarrolla un plan para el incremento siguiente.

- Ventajas:
- Es interactivo.
- Con cada incremento se entrega al cliente un producto operacional que puede ser evaluado.
- Personal. Permite variar el personal asignado a cada iteración.
- Gestión riesgos técnicos. Por ejemplo, disponibilidad de *hardware* específico.

Desventajas: la primera iteración puede plantear los mismos problemas.

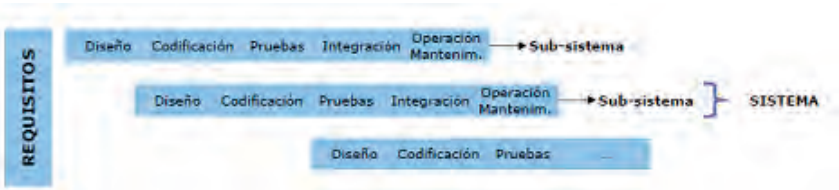


Figura 11.13: Modelo Incremental

Modelos evolutivos: ensamblaje de componentes

El modelo de ensamblaje de componentes es una espiral de Boston adaptado al uso de componentes *software* reutilizables.

Ventajas: reuso de componentes.

Desventajas: encontrar los componentes adecuados.

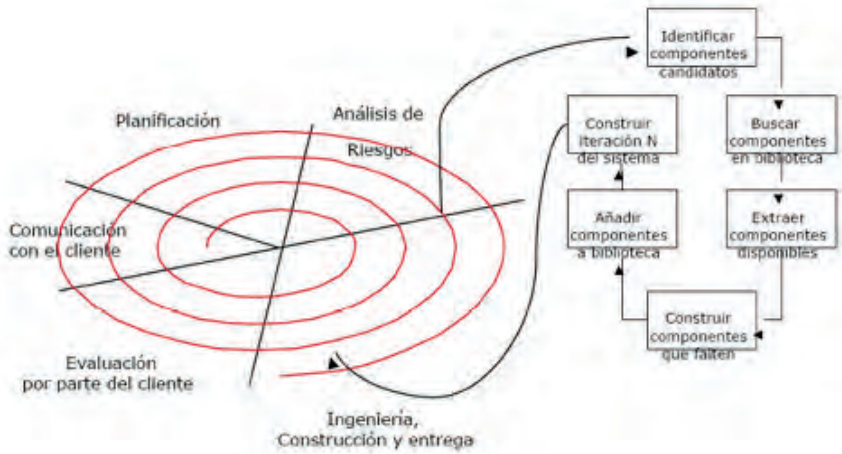


Figura 11.14: Modelo Ensamblaje de Componentes

Modelo de prototipos

Es un proceso que facilita al programador la creación de un modelo de *software*. Puede considerarse un modelo en sí mismo. El prototipado consiste en la construcción de modelos de prueba, que simulen el funcionamiento que se pretende conseguir en el sistema. Esta forma de trabajo previo suele tener como principal objetivo la experimentación con un entorno similar al pretendido, para obtener retroinformación del usuario o cliente que ayuda a los desarrolladores en la concreción de los requisitos.

Los prototipos pueden ser *ligeros* dibujos de pantallas de interfaz con simulación de funcionamiento por enlaces a otros dibujos, un prototipo en papel o un modelo basado en PC, que facilite al usuario la comprensión de

cómo se realizará la interacción hombre-máquina. *Operativos,* módulos de *software* con funcionamiento propio que se desarrollan sin cubrir las funcionalidades completas del sistema, normalmente en entornos RAD (*rapid application development*). Un programa existente que ejecute parte o toda la función deseada, con características que puedan ser mejoradas durante el desarrollo.

La construcción de prototipos comienza con la recolección de requisitos. El analista y el cliente se reúnen, definen los objetivos globales para el *software*, identifican los requisitos y perfilan las áreas donde se necesitará mayor definición. Luego se produce un "diseño rápido" que incluye, por ejemplo, métodos de entrada y formatos de salida. El prototipo es evaluado por el cliente/usuario y se utiliza para refinar los requisitos del *software* a desarrollar. Se produce un proceso interactivo en el cual el prototipo es refinado sucesivamente para que satisfaga las necesidades del cliente.

Generalmente este primer prototipo, llamado "prototipo cero", se descarta, porque es lento o grande o difícil de usar. Esto presenta algunos problemas, tales como que el cliente ve funcionando lo que parece una primera versión del *software*, pero en realidad está hecho "con alambres"; luego no comprende las demoras en el desarrollo final del *software*. Además el técnico, con el apuro de construir un prototipo rápido, muchas veces elige el sistema operativo o el lenguaje de programación inapropiado. Aunque presenta los problemas anteriormente enunciados, la construcción de prototipos es un paradigma muy efectivo de la Ingeniería de *Software*.

Desventajas: el cliente ve el prototipo y lo confunde con el sistema real. No entiende la necesidad de volver a hacerlo, el equipo de desarrollo toma decisiones rápidas para poder construir el prototipo, que más tarde son difíciles de revertir (por ejemplo, el lenguaje de programación).

Figura 11.15: Modelo de prototipos

Modelo transformacional

Se basa en especificaciones formales: el desarrollo de *software* como una secuencia de pasos que gradualmente transforma una especificación en implementación hasta un programa ejecutable. Se deben transformar requerimientos informales en especificaciones formales: métodos de especificación formal. Pretende reducir errores automatizando pasos del desarrollo. Las transformaciones preservan la corrección.

Problemas: hace falta una formación especializada para aplicar la técnica, muchos aspectos de los sistemas reales son difíciles de especificar formalmente: interfaz de usuario y requisitos no funcionales.

Aplicabilidad: sistemas críticos en los que la seguridad y fiabilidad deben poder asegurarse antes de poner el sistema en operación.

Figura 11.16: Modelo transformacional

Ingeniería de sistemas de computadora

La ingeniería de *hardware* y la Ingeniería de *Software* entran dentro de la amplia gama que llamaremos ingeniería de sistemas. Cada una de estas disciplinas intenta establecer un orden en el desarrollo de sistemas basados en computadoras.

Las técnicas de diseño de *hardware* que provienen del diseño electrónico, están bien establecidas y los métodos de fabricación mejoran continuamente. Desafortunadamente, el *software* no ha seguido todavía el mismo camino; la demanda es cada vez mayor y más compleja y no hay respuestas totalmente satisfactorias. Sin embargo, las técnicas de ingeniería para producción de *software* comienzan a tener amplia aceptación.

Sistemas basados en computadora

Se define un sistema basado en computadora como "un conjunto u ordenación de elementos organizados para llevar a cabo algún método, procedimiento o control mediante el procesamiento de la información".

Los elementos de un sistema se combinan de muchas formas para transformar la información. Cada elemento a su vez, puede representar un macroelemento de un sistema mayor. El papel del ingeniero de sistemas es el de definir

los elementos de un sistema basado en computadora específico dentro del contexto de toda la jerarquía de sistemas. La ingeniería de sistemas es una herramienta de solución de problemas. Las funciones que se desean para el sistema son descubiertas, analizadas y asignadas a elementos individuales del sistema.

El ingeniero de sistemas parte de los objetivos y de las restricciones definidas por el usuario y desarrolla una representación de la función, del rendimiento, de las interfaces, de las restricciones de diseño y de la estructura de la información, todo ello pudiendo ser asociado a cada uno de los elementos genéricos del sistema descrito en la sección anterior.

El ingeniero, primero, debe delimitar el sistema identificando el ámbito de funcionamiento y de rendimiento deseados. Una vez que la función, el rendimiento, las restricciones y las interfaces están delimitadas, procederá a realizar la tarea de asignación.

Lenguajes e Ingeniería de *Software*

Independientemente del paradigma, el lenguaje tendrá impacto en la planificación, el análisis, el diseño, la codificación, la prueba y el mantenimiento.

-Si se requieren complejas estructuras de datos, habrá que evaluar qué lenguaje las soporta.

-Si lo importante es un alto rendimiento y posibilidades de tiempo real, se especificará un lenguaje diseñado para aplicaciones de tiempo real (Ada) o eficiencia en memoria o velocidad (C).

-Si se especificaran muchos informes de salida y fuerte manipulación de archivos, se encontrarán lenguajes adecuados como COBOL y RPG.

-El diseño de datos también puede verse influenciado por las características del lenguaje. Ada, C++ y Smalltalk soportan el concepto de tipos abstractos de datos. Pascal y C, los tipos definidos por el usuario, etcétera.

LENGUAJES DE PROGRAMACIÓN

Un programa es la especificación de una tarea de computación. Un lenguaje de programación es una notación para escribir programas. Los lenguajes de programación proporcionan estructuras ("constructores") para organizar los cálculos.

El lenguaje ensamblador, según vimos, es una variante del lenguaje máquina en el cual se manejan identificadores en lugar de los códigos reales para las operaciones, los valores y las direcciones. Los lenguajes máquina y ensambladores se conocen como lenguajes de "bajo nivel". Por otra parte, el concepto "lenguajes de programación" se reserva para lenguajes de alto nivel.

Por lo tanto, los lenguajes máquina son una representación simbólica del conjunto de instrucciones de la UPC; los de alto nivel permiten al programador y al programa independizarse de la máquina. Hoy se utilizan cientos de lenguajes de programación, aunque algunos pocos tienen una gran aceptación en la industria.

Los códigos máquina, los ensambladores y los lenguajes de alto nivel son considerados como las tres primeras generaciones de lenguajes de computadora. Son lenguajes procedimentales. En la década pasada apareció un grupo de lenguajes no procedimentales o de cuarta generación, que hasta la fecha se han utilizado en aplicaciones de bases de datos y otras áreas de procesamiento de negocios.

Niveles de los lenguajes de programación:

- Lenguajes declarativos: lenguajes de órdenes: expresan qué hay que hacer, no cómo hacerlo.
- Lenguaje de alto nivel: lenguajes de órdenes: expresan qué hay que hacer, no cómo hacerlo.
- Lenguaje ensamblador: representación simbólica del lenguaje máquina.
- Lenguaje máquina: unos y ceros (instrucciones del procesador).

Elección de un lenguaje

La elección de un lenguaje de programación para un proyecto específico, debe tener en cuenta las características de ingeniería del mismo. Entre los criterios que se utilizan durante la evaluación de los lenguajes disponibles están las siguientes:

- Área de aplicación general.
- Complejidad algorítmica y computacional.
- Entorno de ejecución del soft.
- Consideraciones de rendimiento.
- Complejidad de las estructuras de datos.
- Disponibilidad de un buen compilador.

Existen lenguajes específicos para el área de la ingeniería y de la ciencia, como así también aquellos orientados a aplicaciones de negocios y los que han sido usados casi exclusivamente por usuarios de microcomputadoras.

También hay lenguajes orientados a la programación de sistemas, a aplicaciones de tiempo real.

Se desarrollaron lenguajes para aplicaciones específicas como la inteligencia artificial y la orientación a objetos.

Por otra parte, si se requieren complejas estructuras de datos, deberá evaluarse qué lenguaje las soporta. Si el requerimiento es un alto rendimiento, velocidad, etcétera, habrá que pensar en un lenguaje con alta eficiencia memoria-velocidad.

Si se deben especificar muchos informes de salida y se requiere una fuerte manipulación de archivos, se encontrarán estas prestaciones en los lenguajes orientados a los negocios.

Perspectiva histórica - Evolución

- Permite apreciar por qué presentan características diferentes.
- Permite ver la evolución de familias de lenguajes.
- Permite ver la influencia que ejercen las arquitecturas y aplicaciones de las computadoras sobre el diseño de lenguajes.

Primera generación de lenguajes (anterior a los 60), código máquina y su equivalente, el ensamblador. Muestran el menor nivel de abstracción con el cual representar un programa. Existen tantos como arquitecturas de procesadores.

Segunda generación (comienzos de 60 hasta mediados de 70), ha sido la base de los lenguajes modernos. Para el área de la ciencia y con origen lingüístico en el álgebra, se creó el lenguaje FORTRAN, que evolucionó a mediados de los 60 hasta el estándar FORTRAN IV y el ALGOL 58, ALGOL 60, entre otros, que fue el predecesor del PASCAL.

Para el procesamiento de datos se desarrolló el COBOL 58, COBOL 68 y sucesores.

Para aplicaciones específicas, como la inteligencia artificial, se desarrolló el LISP a principios de los 60.

Tercera generación (fines de los 60 a los 90), se caracterizaron por sus potentes posibilidades procedimentales y de estructuración de datos. Algunos permiten multitarea, asignación dinámica, etcétera. El ALGOL sirvió de base para otros lenguajes de esta generación.

A principios de los 70 se creó PASCAL para enseñar técnicas de programación; es descendiente de ALGOL y posee fuerte tipificación de datos, estructuración en bloques, soporte de recursividad, etcétera.

PL/1 se desarrolló como lenguaje de propósito general, permite multitarea, procesamiento de listas, compleja E/S, y demás.

A mediados de los 70 se creó el Lenguaje C para implementación de sistemas operativos (UNIX) y para programación de sistemas, pero evolucionó hasta ser de propósito general.

Para la programación de sistemas, también se desarrolló Modula-2, que es un descendiente evolucionado del PASCAL.

Ada fue desarrollado para aplicaciones de tiempo real por el Departamento de Defensa de EE.UU., como un nuevo estándar. Soporta multitarea, manejo de interrupciones, sincronización y comunicación, entre otras tareas.

Lenguajes orientados a objetos permiten implementar los modelos de análisis y diseño creados mediante Análisis orientado a objetos y Diseño orientado a objetos, tales como Smalltalk, Eiffel, C++, Pascal orientado a objetos, y otros.

Lenguajes especializados. Su formulación sintáctica ha sido diseñada para una aplicación particular: PROLOG (inteligencia artificial), APL, FORTH.

Cuarta generación (los 80). Se desarrollan lenguajes no procedurales. En ellos se especifica el resultado deseado en vez de la acción requerida para lograr el resultado. También se les llama lenguajes de consulta (se han

utilizado en aplicaciones de bases de datos y en otras áreas de procesamiento para negocios). Entre ellos podemos citar DBASE, FOX-PRO, CLIPPER, PARADOX.

Actualmente, se percibe un continuo avance de los lenguajes orientados a objetos, con versiones muy evolucionadas y lenguajes de gran orientación visual. (VISUAL BASIC, VISUAL C, entre otros).

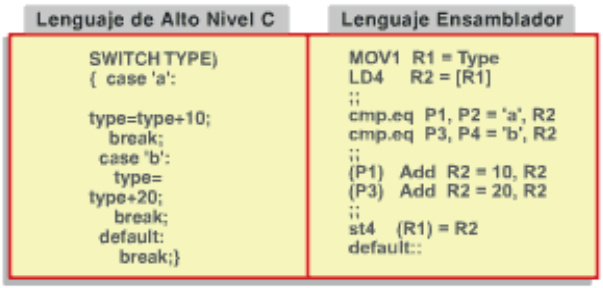


Figura 12.1

Tipos de aplicaciones

1- Aplicaciones científicas. Manipulan predominantemente números y arrays de números, usando principios matemáticos y estadísticos como base de los algoritmos: test estadísticos, programación lineal, análisis de regresión, aproximaciones numéricas para solución de ecuaciones diferenciales e integrales. Se caracterizan por poseer pocos datos, estructuras de datos simples; los problemas científicos requieren más trabajo del procesador central que de los dispositivos de E/S. Lenguajes adecuados a estas aplicaciones son el FORTRAN, PASCAL, APL, C...

2- Aplicaciones de procesamiento de datos. Se trata de problemas cuyo interés predominante es la creación, mantenimiento, extracción y compendio de datos en registros y archivos.

Incluyen funciones relativas a las nóminas, contabilidad, facturación, inventario, producción y ventas. Estas aplicaciones suponen un gran volumen de datos y cantidad de cálculos aritméticos bajo. Lenguajes adecuados a estas aplicaciones son el COBOL, RPG, PL/1...

3- Aplicaciones de inteligencia artificial. Son programas que han sido diseñados principalmente para emular el comportamiento inteligente. Incluyen algoritmos de juegos tales como ajedrez, comprensión del lenguaje natural, robótica y "sistemas expertos"; la computadora se programa para que haga el papel de un experto que realiza su trabajo. Lenguajes adecuados a estas aplicaciones son el LISP y PROLOG (diseñado sobre el principio de Programación lógica).
4- Aplicaciones de programación de sistemas. Incluyen el desarrollo de programas que hacen de interface entre el *hardware* y el programador. Incluyen compiladores, intérpretes, rutinas de E/S, gestión y planificación para la utilización de los distintos recursos de la computadora. Lenguajes adecuados a estas aplicaciones son el C, Ada, Modula-2...
5- Aplicaciones de sistemas de tiempo real. En este tipo de sistemas, la secuencia temporal de entradas es determinada total o parcialmente por el mundo real, lo cual obliga a sincronizar eventos, manejar tiempos absolutos, y demás. Lenguajes adecuados a estas aplicaciones son el Ada, Modula-2...
6- Aplicaciones de procesamiento de texto. Su principal actividad consiste en la manipulación de texto del lenguaje natural en vez de números. Lenguajes adecuados a estas aplicaciones son el SNOBOL, C, entre otros.

Criterios para evaluación y comparación de lenguajes

1- Expresividad: es la habilidad que el lenguaje posee para expresar el significado deseado.
2- Nivel de definición: se refiere a que la sintaxis y semántica del lenguaje no presenten ambigüedad.
3- Tipos y estructuras de datos: se evalúa la habilidad del lenguaje para soportar distintas estructuras de datos.
4- Modularidad: se considera el soporte de subprogramas y extensibilidad del lenguaje.
5- Facilidades de E/S: soporte de acceso a archivos y bases de datos.
6- Transportabilidad: diseño del lenguaje independiente de la máquina.
7- Eficiencia: se refiere a la rapidez que el lenguaje permite para la compilación y ejecución.
8- Pedagogía: se considera si el lenguaje es intrínsecamente fácil de enseñar y aprender.
9- Generalidad: se verifica su utilidad en un amplio rango de aplicaciones.

Estudio de un lenguaje

Agrupa tres intereses diferentes a saber:

1) El del programador profesional.

2) El del diseñador del lenguaje (hacen el lenguaje).

3) El del implementador del lenguaje (hacen los compiladores).

Organización de las descripciones de lenguajes

Programas tutelares: proporcionan una idea de los principales constructores del lenguaje y la forma en que deben usarse.

Manuales de referencia: describen la sintaxis y semántica de un lenguaje.

Definiciones formales: descripción precisa de la sintaxis y semántica de un lenguaje, dirigida a especialistas. Se han desarrollado notaciones estándar especiales.

PROGRAMACIÓN DE SISTEMAS

Se denomina *system programming* al conjunto de programas de apoyo, en *software* de base, necesarios para que la computadora pueda funcionar como un todo sólido y coherente ante los usuarios; estos programas brindan servicios cercanos a los de un sistema operativo (de bajo nivel), pero no forman parte de él.

El SO es como el pegamento que mantiene unidos a todos los otros componentes del *software* de sistemas: editores, ensambladores, compiladores, y otros.

Lenguaje máquina

Es el lenguaje propio de la computadora: unos y ceros; hace referencia a los registros de la UPC, al acumulador, a las celdas de memoria, etcétera. Como está muy lejos del lenguaje cotidiano, se lo trata de "acercar" a un lenguaje más comprensible. Por este motivo se desarrollaron los lenguajes simbólicos y, a un nivel más alto, los distintos lenguajes de programación, para que el ser humano no tuviera que expresarse en unos y ceros.

Es importante comprender dos conceptos que se mencionan a continuación:

Programa fuente: es el programa escrito en un lenguaje cercano al ser humano.

Programa objeto: es el programa escrito en código máquina: unos y ceros, listo para ser ejecutado.

Para poder comunicarnos con la computadora, entonces, escribimos los programas en lenguajes de más alto nivel que el código máquina y utilizamos traductores que se encargan de convertir el programa a lenguaje máquina.

Traductores

Según vimos, debemos resolver el problema de traducir el programa fuente (escrito en códigos cercanos al ser humano) a lenguaje máquina (único que "entiende" la UPC).

Un traductor, por lo tanto, debe:

1- Leer el programa fuente.

2- Traducirlo a lenguaje máquina.

Contando con un traductor, puede programarse en un lenguaje de más "alto nivel" que el lenguaje máquina.

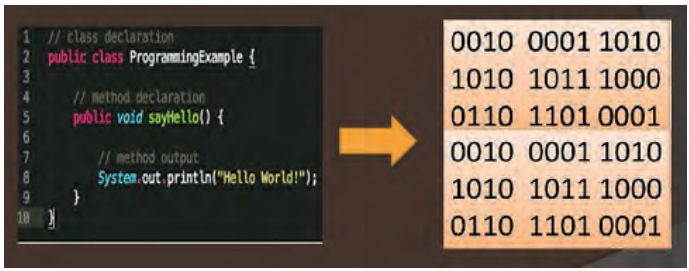


Figura 13.1: Función de un traductor

Existen distintos tipos de traductores, según veremos a continuación: ensambladores, editores, cargadores, intérpretes y compiladores.

Ensambladores

Si escribimos un programa fuente en un lenguaje de bajo nivel, tal como el código mnemónico o simbólico (por ejemplo, el utilizado en la programación del microprocesador IC2 del Anexo A), el traductor realizará las siguientes funciones:

1. Busca cada palabra mnemónica en el diccionario y la traduce (si no está, avisa que se produjo una condición de error).
2. Le asigna direcciones absolutas a las variables simbólicas que encuentre en el programa. Una variable será un nombre simbólico asociado a una celda cualquiera de memoria; la asociación se realiza de manera automática. La ventaja de esta situación es la de empezar a "despegarnos" de la computadora y dirigirnos a

ella en un lenguaje más simbólico. Por supuesto, deberá guardar las direcciones que asignó a las variables para "acordarse" de ellas.

3. Se ocupa del manejo de etiquetas: guarda las definiciones de etiquetas en una tabla especial, y luego, reemplaza las referencias a las etiquetas por las direcciones donde se encontraron. Las etiquetas son referencias simbólicas a pasos dentro del programa, que se usan para saltos "condicionales", cuando se necesita modificar la secuencia de ejecución de las instrucciones.

¿Cómo funcionaría este traductor?
Para cada renglón del programa fuente:
Si existe una etiqueta, guarda su dirección en la tabla.
Busca la palabra mnemónica en el diccionario.
Si está, la traduce a código máquina.
Si no está, avisa condición de error.
Localiza la variable simbólica del renglón.

Si es la primera vez que la encuentra, le asigna una dirección absoluta y la guarda. Si no, determina si se trata de una variable predefinida (y debe reemplazarse por su dirección absoluta) o de una referencia a una etiqueta (en ese caso, se reemplazará por la dirección guardada en la tabla).

A un traductor como este que presentamos, podemos llamarlo "ensamblador"; libera al programador de la tarea de asignar direcciones absolutas a las variables simbólicas.

A los programas producidos de esta manera se los conoce como "programas escritos en lenguaje ensamblador". La relación entre la cantidad de instrucciones escritas en código simbólico y las traducidas a código máquina es uno a uno; por este motivo los programas escritos en estos lenguajes suelen ser muy largos.

Macroprocesadores o macroensambladores

Es un traductor que tiene capacidad para compactar renglones repetitivos en uno solo y pedir al ensamblador que lo expanda en el momento de la traducción.

Ejemplo: Expansión de un macroensamblador:

$$\left.\begin{matrix} LDW, R0 \\ ADD(Rn), R0 \\ ST R0, W \end{matrix}\right\} \Longrightarrow SUM$$

Nota: LD carga un valor de R0 en Memoria (W), ADD suma dos valores y ST almacena el valor R0 en la dirección de memoria W

Cada vez que el ensamblador observara el mnemónico SUM, lo expandiría para producir los tres renglones anteriores.

```
MACRO SUM
    LD
    ADD
    ST
    FINMACRO
```

Cuando se decide utilizar la macro, se lo hace simplemente, nombrándola.

Compiladores

Si queremos tener la posibilidad de comunicarnos con la computadora en un lenguaje más cercano al nuestro (es decir, al humano), utilizamos un lenguaje de más alto nivel expresivo. Esto significa que nos permita con un solo comando poder decir muchas cosas.

Por lo tanto, necesitaremos un traductor que pase a la máquina, a su lenguaje, lo que le dijimos en un lenguaje de alto nivel.

Este es un proceso muy complejo, dado que la estructura de un lenguaje está definida por su gramática y una gran cantidad de significados, no reducibles al diccionario. La traducción de un párrafo implica mucho más que traducir palabra por palabra.

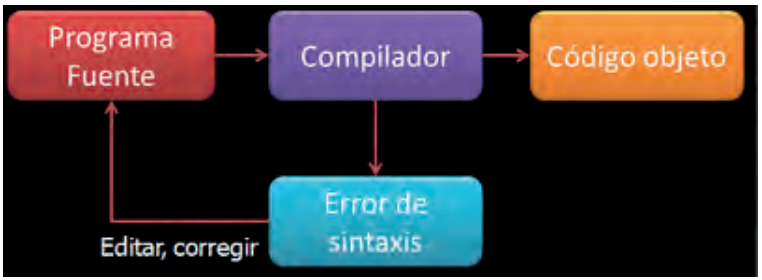


Figura 13.2: Compilador

Un compilador realiza tres etapas antes de poder generar el código objeto:

1. Análisis lexicográfico.
2. Análisis sintáctico.
3. Análisis semántico.

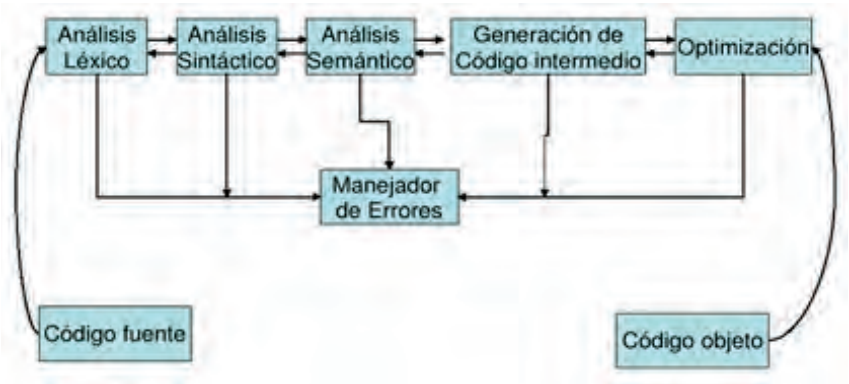


Figura 13.3: Etapas de un compilador

Se realiza una primera etapa de *análisis lexicográfico*, en la cual se reconocen todos los símbolos aislados que constituyen la frase: reconocer las letras (y signos de puntuación) y reconocer las palabras (no significa entender).

La tarea central del analizador lexicográfico consiste en separar los componentes sintácticos (tokens) de entre el conjunto de símbolos del programa fuente. Recordemos que en un renglón coexisten símbolos de diversas clases (letras, dígitos, signos de puntuación, y otros). El modelo matemático de un analizador como este se llama "autómata finito"; es una función matemática que logra reconocer grupos de caracteres.

La segunda etapa llamada *análisis sintáctico* consiste en encontrar la estructura gramatical de la frase cuyos elementos ya reconocimos. Este proceso utiliza métodos matemáticos para lograr desenmascarar la estructura inherente a la frase. Los analizadores sintácticos (parsers) se dividen entre los que funcionan de manera descendente (los más viejos) y ascendente (los más recientes). Todo esto está relacionado con la teoría de los lenguajes formales.

En la próxima etapa de *análisis semántico* se está en posición de entender lo que la frase significa. Lo importante es determinar la coherencia entre lo que decimos por medio de un lenguaje y los elementos de la computadora que referimos.

El compilador tiene que hacer estos tres tipos de análisis sobre el programa fuente para poder, así, obtener el programa en lenguaje máquina o "programa objeto".

La traducción o generación de código es posterior a los análisis realizados y busca representar la frase fuente original en términos de elementos de un lenguaje más sencillo no dotado de estructura (el lenguaje máquina).

Generalmente, la generación de código se realiza en más de una etapa: 1) generación de código intermedio y su optimización y, 2) generación de código objeto y su optimización.

El código objeto es de mayor extensión que el programa fuente. Se debe a la correspondencia de uno a varios que existe entre una expresión escrita en un lenguaje de alto nivel y otra en lenguaje máquina.

Ventajas

- Tienden a ser más rápidos que los traducidos en tiempo de ejecución, debido a la sobrecarga del proceso de traducción.
- Se compila una vez, se ejecuta n veces.

- El compilador tiene una visión global del programa, por lo que la información de mensajes de error es más detallada.
- Son unidades autónomas listas para ser ejecutadas.
- Si bien estar restringido a un paquete de *hardware* específico tiene sus desventajas, compilar un programa también puede incrementar el desempeño de este último.

Desventajas

- Necesita mucha más memoria.
- No se puede ejecutar hasta tanto esté libre de todo error.
- Dado que traduce el código fuente a un lenguaje máquina específico, los programas deben ser compilados específicamente para OS X, Windows o Linux, así como para arquitecturas de 32 o 64 bits.
- Tiempos de compilación, en grandes suites de aplicaciones pueden tardar cantidades significativas de tiempo en compilar todo el código.

Ejemplos de compiladores

ADA, ORACLE J DEVELOPER, C, C++, POWER BUILDER, FORTRAN, PASCAL, DELPHI, MODULA, Eiffel, entre otros.

Intérpretes

Cuando el traductor ejecuta inmediatamente el código obtenido del programa fuente (instrucción por instrucción) recibe el nombre de intérprete. No "genera" un programa objeto sino que ejecuta el código en cuanto lo obtiene.

Existe una gran diferencia de velocidad entre un programa (objeto) compilado y un programa fuente ejecutado.

El intérprete es más lento, ya que debe analizar, traducir cada instrucción, aun cuando esté dentro de un ciclo. Requiere de un programa auxiliar (el intérprete), el cual traduce y ejecuta el programa línea por línea.

Figura 13.4: Intérprete

Ventajas

- Necesita menos memoria.
- Permite una mayor interactividad con el código en tiempo de desarrollo.
- Elimina la necesidad de realizar una "verificación" después de cada modificación del programa.
- Independencia de plataforma.

Desventajas

- Un código traducido por un intérprete no puede funcionar sin este.
- Se ejecuta n veces, se interpreta n veces.
- No hay independencia entre la etapa de traducción y ejecución.

Ejemplos de intérpretes

BASIC 4GL, Java Script, LOGO, LISP, PERL, PHP, HTML, PROLOG, PYTHON, RUBY, entre otros.

INTÉRPRETE	COMPILADOR
El programa fuente se lee línea a línea.	El programa fuente se lee totalmente.
Traduce el programa cuando se ejecuta, convirtiendo el código fuente en lenguaje de máquina.	El proceso de compilación lo transforma en lenguaje de máquina.
Todo programa se puede interpretar en cualquier plataforma (sistemas operativo)	El archivo generado por el compilador solo funciona en la plataforma en donde se lo ha creado.
No genera un ejecutable.	Puede generar un ejecutable.
La ejecución es más lenta, ya que para cada línea del programa es necesario realizar la traducción (ej. un bucle)	La ejecución de un archivo compilado es de 10 a 20 veces mas rápida que un archivo interpretado.
El código fuente es necesario en cada ejecución. No funciona si no se tiene el intérprete.	El proceso de traducción se realiza una sola vez. La ejecución es independiente.
Los errores sintácticos se detectan durante la ejecución, ya que la traducción y ejecución se van haciendo simultáneamente.	Los errores sintácticos se detectan durante la compilación. Si el código fuente contiene errores sintácticos, no podrá ser ejecutado.

Figura 13.5: Diferencias entre un compilador y un intérprete

Lenguajes intermedios

Algunos lenguajes pertenecen a ambas categorías, es decir, un programa escrito en estos lenguajes pasan por una fase de compilación intermediaria (un archivo escrito en un lenguaje ininteligible y no ejecutable) que requeriría de un intérprete.

Los applets Java, pequeños programas que a menudo se cargan en páginas web, son archivos compilados que solo pueden ejecutarse dentro de un navegador web.

Cargadores

Diseño básico:

Localiza el dispositivo de almacenamiento secundario que contenga la información que se va a cargar, averigua a partir de qué dirección de memoria cargar el programa objeto.

Para cada renglón del programa objeto ejecuta lo siguiente:

• Determina la cantidad de celdas necesarias para almacenarlo.

• Deposita esos datos en memoria, a partir de la celda inicial.

El cargador se encuentra residente en memoria, pero la carga del cargador constituye un problema inicial que recibe el nombre de "bootstrap".

Se utiliza un minicargador que se carga por medios electrónicos en una memoria ROM, la cual automáticamente deposita su contenido en memoria central al encenderse la máquina. Su función es cargar el cargador.

A este proceso (carga del minicargador) se lo conoce como IPL (*inicial program load*). Se ejecuta cada vez que se enciende la computadora.

REDES DE COMPUTADORAS

Durante el transcurso del siglo xx ha habido una tecnología claramente dominante y clara. Esta tecnología está relacionada con la obtención, procesamiento y distribución de la información.

La fusión entre las computadoras y las comunicaciones ha tenido una profunda influencia en las organizaciones y las personas.

Una red de computadoras (también llamada red de ordenadores, red de comunicaciones de datos o red informática) es un conjunto de equipos nodos y *software* conectados entre sí por medio de dispositivos físicos o inalámbricos que envían y reciben impulsos eléctricos, ondas electromagnéticas o cualquier otro medio para el transporte de datos, con la finalidad de compartir información, recursos y ofrecer servicios.

Año	Evento
1830	Invención del telégrafo (Samuel Morse)
1876	Invención del teléfono (Alexander Graham Bell)
1880	Puesta en funcionamiento del sistema telefónico punto a punto manual
1890	Puesta en funcionamiento de los conmutadores electromagnéticos
1940	Aparición de los primeros computadoras
1960	Expansión de las computadoras y aparición del MODEM Desarrollo de la red precursora de Internet: ARPANET
1969	Aparición de las especificaciones RS-232: comunicaciones serie asíncronas
1970	Utilización de computadoras para funciones de conmutación Aparición de nuevos tipos de redes
1990	Explosión de Internet

Figura 14.1: Principales avances en comunicaciones

La estructura y el modo de funcionamiento de las redes informáticas actuales están definidos en varios estándares, siendo el más importante y extendido de todos ellos el modelo TCP/IP utilizado como base para el modelo de referencia OSI. Este último, concibe cada red como estructurada en siete capas con funciones concretas, pero relacionadas entre sí (en TCP/IP se habla de cuatro capas). Debe recordarse que el modelo de referencia OSI es una abstracción teórica, que facilita la comprensión del tema, si bien se permiten ciertos desvíos respecto a dicho modelo. Existen multitud de protocolos repartidos por cada capa, los cuales también están regidos por sus respectivos estándares

Comunicación

Como en todo proceso de comunicación, se requiere de un emisor, un mensaje, un medio y un receptor. La finalidad principal para la creación de una red de ordenadores es compartir los recursos y la información en la distancia, asegurar la confiabilidad y la disponibilidad de la información, aumentar la velocidad de transmisión de los datos y reducir el costo. Un ejemplo es Internet, el cual es una gran red de millones de ordenadores ubicados en distintos puntos del planeta interconectados básicamente para compartir información y recursos.

Figura 14.2: Comunicación

Tipos de servicios

- Servicios de archivos compartidos.
- Servicios de impresión.
- Servicios de mensajería.
- Servicios de backup.
- Internet.

Clasificación

Según su carácter

Redes públicas:

Una red se denomina pública cuando un usuario perteneciente a la red, no tiene otra restricción para conectarse que la disponibilidad de los medios técnicos de la red. Los servicios correspondientes a estas redes son prestados por compañías que se dedican a transportar señales (que habitualmente reciben el nombre de prestadores o carriers).

Redes privadas:

Una red es privada cuando su finalidad está bien determinada y los usuarios pertenecen a una o más compañías con intereses específicos en la utilización de la ella.

Es posible desarrollar una red privada sustentada por los medios de una red pública. En este caso, el cliente alquila los enlaces a un prestador público para transportar sus datos y, además, se encarga de su administración, teniendo el uso exclusivo de todo o una parte de la red pública.

Según las señales que transportan

Redes analógicas:

En estas redes se considera que el tipo de señal a transportar es analógica. Son las más difundidas, ya que las primeras de este tipo se desarrollaron para transportar la voz y lograron un amplio desarrollo.

Ejemplo: las centrales telefónicas, hoy en día también hay centrales digitales.

Redes digitales:

Las redes digitales están equipadas para transportar señales digitales y surgen de la necesidad de transportar señales en la misma forma en que son generadas.

Por ejemplo, las computadoras.

Hay una gran cantidad de ventajas que marcan una tendencia en el futuro a digitalizar las redes, entre las que se destacan la facilidad de desarrollo

de circuitos integrados, regeneración de señales sin amplificar, menor interferencia, entre otras.

Según su cobertura

Redes PAN:

Red de área personal (*personal area network*, PAN) es una red de nodos usada para la comunicación entre los dispositivos cercanos a una persona. Se utilizan para conectar dispositivos personales, como teléfonos celulares, auriculares y asistentes digitales personales entre sí, a otros dispositivos autónomos y redes más grandes, sin necesidad de cables. Las redes para espacios personales continúan desarrollándose hacia la tecnología del Bluetooth, hacia el concepto de redes dinámicas, el cual permite una fácil comunicación con los dispositivos que van adheridos al cuerpo o indumentaria de una persona, ya sea que esté en movimiento o no, dentro del área de cobertura de la red.

Figura 14.3: Redes PAN

Las redes PAN también son conocidas como redes de proximidad, debido a que los equipos permiten la transmisión de datos dentro de un espacio limitado y, generalmente muy próximo al usuario de los dispositivos. Puede

decirse que tienen una cobertura de una casa entera o una pequeña oficina, sin llegar a cubrir todo un edificio. La mayoría está basada en tecnologías inalámbricas.

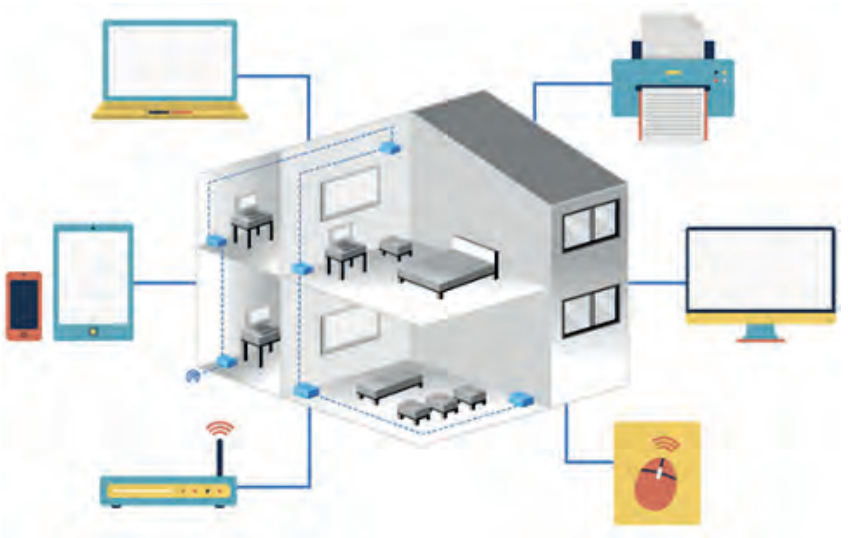


Figura 14.4: Representación de una red PAN

PAN prevé el acercamiento de un paradigma de redes, la cual atrae el interés de los investigadores y las industrias que quieren aprender más acerca de las soluciones avanzadas para redes, tecnologías de radio, altas transferencias de bits, nuevos patrones para celulares, y un soporte de *software* más sofisticado. El PAN debe proporcionar una conectividad usuario a usuario, comunicaciones seguras y garantizar calidad de servicio. El sistema tendrá que soportar diferentes aplicaciones y distintos escenarios de operación, y así poder abarcar una gran variedad de dispositivos.

Redes LAN

Una red de área local o LAN (*local area network* en inglés) es una red de computadoras que abarca un área reducida a una casa, un departamento, un edificio, un campus. Se trata de un conjunto de dispositivos que pertenecen

a una misma organización, están conectados dentro de un área geográfica pequeña y bajo una misma administración.

Gracias a la red, los usuarios de estas computadoras pueden compartir documentos e, incluso, hacer un uso común de ciertos periféricos, como una impresora. Las ventajas de la instalación de una red LAN en una empresa o en una casa son numerosas. Al compartir una impresora, por ejemplo, no es necesario que cada usuario tenga su propio dispositivo, lo que permite ahorrar una gran cantidad de dinero. Por otra parte, la facilidad para acceder a documentos alojados en cualquier nodo de la red LAN es muy útil a la hora de realizar un trabajo en conjunto.

Salvando excepciones, la velocidad de transmisión de datos dentro de una red LAN es mucho mayor que aquella que se consigue a través de Internet, ya que los dispositivos que vinculan los ordenadores o demás aparatos entre sí dentro de un edificio suelen superarla varias veces. Como si este beneficio no fuera suficiente, mientras se utiliza la conexión local no se gasta tráfico del servicio a Internet, lo cual puede resultar útil en el caso de planes limitados o cuando el volumen de datos a transferir es demasiado grande.

La historia de los videojuegos reserva un espacio muy especial para la red LAN, ya que fue gracias a esta posibilidad de conectar los equipos que se abrieron las puertas a una nueva forma de experimentar la diversión. Al principio, rompió con el límite de jugadores que podían competir o colaborar en un mismo juego, que en su momento estaba ligado a las capacidades de cada consola u ordenador y a la creatividad de los desarrolladores; pero con el tiempo sirvió de fase experimental para la red de redes, Internet.

Aunque la red LAN pueda parecer reservada al ámbito empresarial, muchos jugadores siguen disfrutando de sus posibilidades para divertirse con amigos o para competir en los torneos más importantes del mundo, donde la inestabilidad de Internet sería fatal. Como en otros aspectos de la informática y la electrónica, los expertos tienen muchos factores en cuenta a la hora de armar, optimizar y mantener sus redes locales, desde las marcas y modelos de los componentes hasta los detalles más remotos de la configuración del *software*, para asegurarse de no desperdiciar ni una sola fracción de segundo.

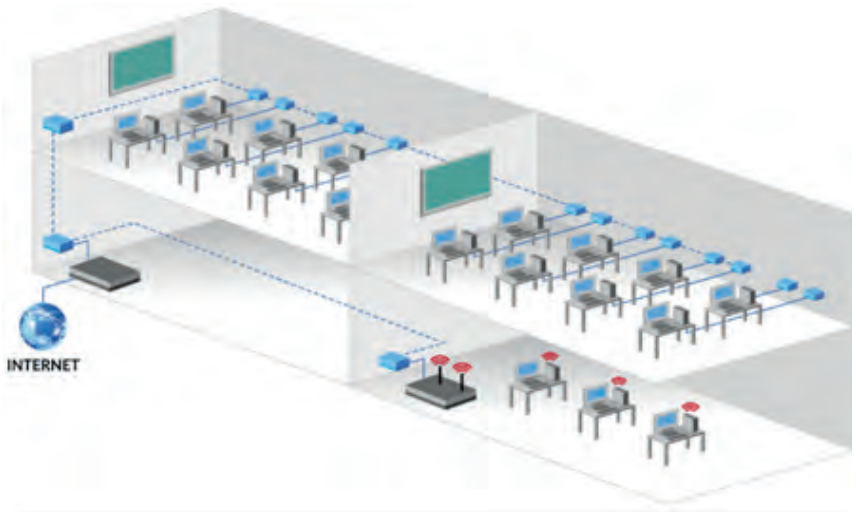


Figura 14.5: Representación de una Red LAN

Componentes:

- Servidor: es aquella computadora que va a compartir sus recursos *hardware* y *software* con los demás equipos de la red. Sus características son potencia de cálculo, importancia de la información que almacena y conexión con recursos que se desean compartir.
- Estación de trabajo: las computadoras que toman el papel de estaciones de trabajo aprovechan o tienen a su disposición los recursos que ofrece la red así como los servicios que proporcionan los servidores a los cuales pueden acceder.
- Gateways o pasarelas: es un *hardware* y *software* que permite las comunicaciones entre la red local y grandes computadoras (mainframes). El gateway adapta los protocolos de comunicación del mainframe (X25, SNA, etcétera) a los de la red, y viceversa.
- Bridges o puentes de red: es un *hardware* y *software* que permite que se conecten dos redes locales entre sí. Un puente interno es el que se instala en un servidor de la red, y un puente externo es el que se hace sobre una estación de trabajo de la misma red. Los puentes también pueden ser locales o remotos. Los primeros son los que conectan a redes de un

mismo edificio, usando tanto conexiones internas como externas. Los segundos conectan redes distintas entre sí, llevando a cabo la conexión a través de redes públicas, como la red telefónica, RDSI o red de conmutación de paquetes.

- Tarjeta de red o NIC (*network interface card*). Básicamente realiza la función de intermediario entre la computadora y la red de comunicación. En ella se encuentran grabados los protocolos de comunicación de la red. La comunicación con la computadora se realiza normalmente a través de las ranuras de expansión que este dispone, ya sea ISA, PCI o PCMCIA. Aunque algunos equipos disponen de este adaptador integrado directamente en la placa base.
- El medio: constituido por el cableado y los conectores que enlazan los componentes de la red. Los medios físicos más utilizados son el cable de par trenzado, cable coaxial y la fibra óptica, cada vez más usada.
- Concentradores de cableado (HUB/SWITCH): una LAN en bus usa solamente tarjetas de red en las estaciones y cableado coaxial para interconectarlas, además de los conectores; sin embargo, este método complica el mantenimiento de la red ya que si falla alguna conexión toda la red deja de funcionar. Para impedir estos problemas las redes de área local usan concentradores de cableado para realizar las conexiones de las estaciones, en vez de distribuir las conexiones, el concentrador las centraliza en un único dispositivo manteniendo indicadores luminosos de su estado e impidiendo que una de ellas pueda hacer fallar toda la red. Existen dos tipos de concentradores de cableado:
    - Concentradores pasivos: actúan como un simple concentrador cuya función principal consiste en interconectar toda la red.
    - Concentradores activos: además de su función básica de concentrador también amplifican y regeneran las señales recibidas antes de ser enviadas y ejecutadas.

Realmente no hay un límite máximo de computadoras, dependerá entre otras cosas de los switches que se utilicen. No obstante, considerando que se tuvieran muy buenos equipos y bien organizada la red, entre 400 y 500 CPU sería lo máximo que podría soportar la LAN sin que empezara a degradarse notablemente el rendimiento de la red a causa del propio tráfico de broadcast.

Redes MAN

Una red de área metropolitana, MAN, (*metropolitan area network*, en inglés) es una red de alta velocidad (banda ancha) que da cobertura en un área geográfica extensa, proporcionando capacidad de integración de múltiples servicios mediante la transmisión de datos, voz y video, sobre medios de transmisión tales como fibra óptica y par trenzado (BACKBONE), la tecnología de pares de cobre se posiciona como la red más grande del mundo, una excelente alternativa para la creación de redes metropolitanas, ofrecen velocidades de 10 Mb/s o 20 Mb/s, sobre pares de cobre, y 100 Mb/s, 1 Gb/s y 10 Gb/s mediante fibra óptica.

Otra definición de una MAN es que se trata de una colección de LAN dispersas en una ciudad (decenas de kilómetros). Una MAN utiliza tecnologías tales como ATM, *frame relay*, DSL (*digital subscriber line*), WDM (*wavelenght division multiplexing*), ISDN, E1/T1, PPP, etcétera, para conectividad a través de medios de comunicación tales como cobre, fibra óptica, y microondas.

El concepto de red de área metropolitana representa una evolución del concepto de red de área local a un ámbito más amplio, cubriendo áreas mayores que en algunos casos no se limitan a un entorno metropolitano sino que pueden llegar a una cobertura regional, e incluso, nacional mediante la interconexión de diferentes redes de área metropolitana.

Este tipo de redes es una versión más grande que la LAN, que normalmente se basa en una tecnología similar a esta. La principal razón para distinguir una MAN con una categoría especial es que se ha adoptado un estándar para que funcione, que equivale a la norma IEEE.

Las redes WAN también se aplican en las organizaciones, en grupos de oficinas corporativas cercanas a una ciudad, estas no contienen elementos de conmutación, los cuales desvían los paquetes por una de varias líneas de salida potenciales. Estas redes pueden ser públicas o privadas.

Las redes de área metropolitana comprenden una ubicación geográfica determinada "ciudad, municipio", y su distancia de cobertura es mayor de 4 km. Son redes con dos buses unidireccionales, cada uno de ellos es independiente del otro en cuanto a la transferencia de datos.

Una red de área metropolitana puede ser pública o privada. Un ejemplo de MAN privada sería un gran departamento o administración con edificios distribuidos por la ciudad, transportando todo el tráfico de voz y datos entre edificios por medio de su propia MAN y encaminando la información externa por medio de los operadores públicos. Los datos podrían ser transportados entre

los diferentes edificios en forma de paquetes o sobre canales de ancho de banda fijos. Los edificios pueden enlazar aplicaciones de video para reuniones, simulaciones o colaboración de proyectos.

Un ejemplo de MAN pública es la infraestructura que un operador de telecomunicaciones instala en una ciudad con el fin de ofrecer servicios de banda ancha a sus clientes localizados en esa área geográfica.

Figura 14.6: Representación de una Red MAN

Redes WAN

Una red de área amplia, WAN (*wide area network,* en inglés), es una red de computadoras que une varias redes locales, aunque sus miembros no estén todos en una misma ubicación física. Muchas WAN son construidas por organizaciones o empresas para su uso privado, otras son instaladas por los proveedores de Internet (ISP) para proveer conexión a sus clientes.

Una definición de las redes WAN, en el término de aplicación de protocolos y conceptos de redes de ordenadores, sería que se trata de **tecnologías de redes de ordenadores que se utilizan para transmitir datos a través de largas distancias, y entre las diferentes redes LAN, MAN y otras arquitecturas de redes de ordenadores localizadas**.

Esta distinción se debe al hecho de que las tecnologías LAN comunes que operan en la capa media (como Ethernet o wifi) a menudo están orientados a redes localizadas físicamente, y por lo tanto, no pueden transmitir datos a través de decenas, cientos o incluso miles de millas o kilómetros.

Las WAN se utilizan para conectar redes LAN y otros tipos de redes. Así, los usuarios se pueden comunicar con usuarios y equipos de otros lugares. Muchas WAN son construidas por una organización en particular y son privadas. Otras, construidas por los proveedores de servicios de Internet, que proporcionan conexiones LAN a una organización de Internet.

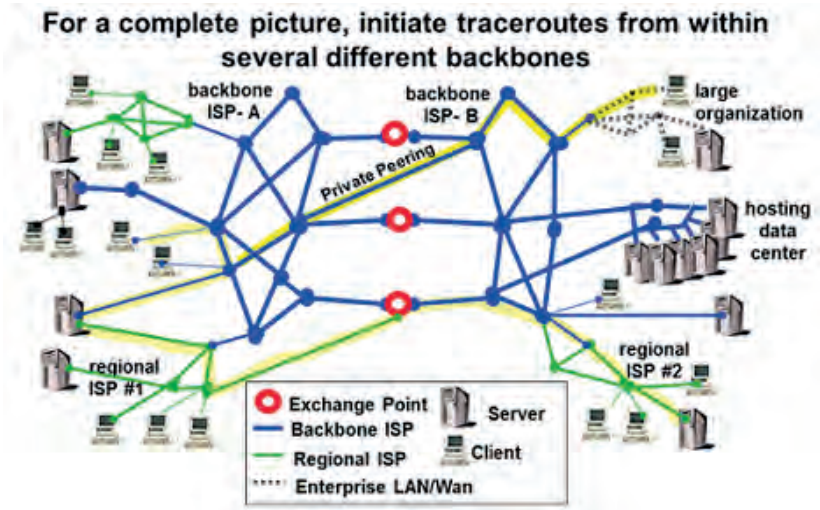


Figura 14.7: Representación de una Red WAN

Redes inalámbricas

Utilizan las ondas de radio para llevar la información de un punto a otro sin necesidad de un medio físico guiado. Al hablar de ondas de radio nos referimos normalmente a portadoras de radio, sobre las que va la información, ya que realizan la función de llevar la energía a un receptor remoto. Los datos a transmitir se superponen a la portadora de radio y, de este modo, pueden ser extraídos exactamente en el receptor final.

A este proceso se le llama modulación de la portadora por la información que está siendo transmitida. Si las ondas son transmitidas a distintas frecuencias de radio, varias portadoras pueden existir en igual tiempo y espacio sin interferir entre ellas. Para extraer los datos el receptor se sitúa en una determinada frecuencia, frecuencia portadora, ignorando el resto.

En una configuración típica de LAN (con cable) los puntos de acceso (*transceiver*) conectan la red cableada de un lugar fijo mediante cableado normalizado. El punto de acceso recibe la información, la almacena y la transmite entre la WLAN y la LAN cableada. Un único punto de acceso puede soportar un pequeño grupo de usuarios y puede funcionar en un rango de, al menos, treinta metros y hasta varios cientos. El punto de acceso (o la antena conectada al punto de acceso) es normalmente colocado en alto, pero podría colocarse en cualquier lugar en que se obtenga la cobertura de radio deseada. El usuario final accede a la red WLAN a través de adaptadores. Estos proporcionan una interfaz entre el sistema de operación de red del cliente y las ondas, mediante una antena. La naturaleza de la conexión sin cable es transparente a la capa del cliente.

WPAN: *wireless personal area network*

En este tipo de red de cobertura personal, existen tecnologías basadas en HomeRF, estándar para conectar todos los teléfonos móviles de la casa y los ordenadores mediante un aparato central; Bluetooth, protocolo que sigue la especificación IEEE 802.15.1; ZigBee, basado en la especificación IEEE 802.15.4 y utilizado en aplicaciones como la domótica, que requieren comunicaciones seguras con tasas bajas de transmisión de datos y maximización de la vida útil de sus baterías, bajo consumo; RFID, sistema remoto de almacenamiento y recuperación de datos con el propósito de transmitir la identidad de un objeto (similar a un número de serie único) mediante ondas de radio.

Una piconet es una red formada por dispositivos móviles utilizando tecnología Bluetooth, es una derivación de WPAN, está formada por dos a siete dispositivos. La piconet sigue una estructura de maestro-esclavo, donde el maestro es el que proporciona la conexión mediante un request que envía el esclavo, y define en qué frecuencia va a trabajar.

Tiene un alcance máximo de 10 metros y puede aumentar juntando varias piconets formando una scatternet: un nodo esclavo hace, a su vez, el rol de un maestro proporcionado conexión a los demás esclavos.

El alcance típico de este tipo de redes es de alrededor de los 10 metros, máximo. Su finalidad es comunicar cualquier dispositivo personal (ordenador, terminal móvil, PDA, y otros) con sus periféricos, así como permitir una comunicación directa a corta distancia entre estos dispositivos.

Red WLAN (*wireless local network*)

Una red de área local inalámbrica, más conocida como WLAN, es básicamente un sistema de transferencia y comunicaciones de datos, el cual no requiere que las computadoras que la componen tengan que estar cableadas entre sí, ya que todo el tráfico de datos entre ellas se realiza a través de ondas de radio.

A pesar de que son menos seguras que su contrapartida cableada, ofrecen una amplia variedad de ventajas, es por ello que su implementación crece, día a día, en todos los ámbitos.

Sin embargo, la característica más destacada de las redes inalámbricas es el ahorro en el tendido de los cables para la interconexión de las PC y dispositivos las que componen, ya que no requiere de ningún cable para su interconexión, una gran ventaja para el hogar, la oficina y las PYMES.

Figura 14.8: Representación de una Red WLAN

Red WMAN (*wireless metropolitan network*)

En términos muy básicos, la WMAN, o red metropolitana inalámbrica, es una versión inalámbrica de MAN, la cual puede llegar a tener un rango de alcance de decenas de kilómetros. Esta tecnología utiliza técnicas basadas en el estándar de comunicaciones WiMAX (*worldwide interoperability for microwave access*).

Red WWAN (*wireless local area network*)

Wireless WAN, o red inalámbrica de área amplia, es una red capaz de brindar cobertura inalámbrica en un área geográfica relativamente grande. Básicamente, una WWAN se diferencia de una WLAN en que utiliza tecnología de la red celular de comunicaciones móviles como WiMAX, UMTS, GPRS, EDGE, CDMA2000, GSM, CDPD, Mobitex, HSPA y 3G para realizar la transferencia de los datos entre los nodos que componen la red. También puede ser que nos encontremos con la posibilidad de utilizar LMDS y wifi autónoma para acceder a Internet.

Características

• *Configuraciones de red para radiofrecuencia*

Pueden ser de muy diversos tipos y tan simples o complejas como sea necesario. La más básica se da entre dos ordenadores equipados con tarjetas adaptadoras para WLAN, de modo que pueden poner en funcionamiento una red independiente siempre que esté dentro del área que cubre cada uno. Esto es llamado red de igual a igual (*peer to peer*). Cada cliente tendría únicamente acceso a los recursos del otro cliente, pero no a un servidor central. Este tipo de redes no requiere administración o preconfiguración.

Instalando un punto de acceso se puede doblar la distancia a la cual los dispositivos pueden comunicarse, ya que estos actúan como repetidores. Desde que el punto de acceso se conecta a la red cableada cualquier cliente tiene acceso a los recursos del servidor, y además, gestiona el tráfico de la red entre los terminales más próximos. Cada punto de acceso puede servir a varias máquinas, según el tipo y el número de transmisiones que tiene lugar. Tienen un alcance finito, del orden de 150 m en lugares o zonas abiertas. En zonas grandes, como por ejemplo, un campus universitario o un edificio, probablemente sea necesario más de un punto de acceso. La meta es cubrir el área con células que solapen sus áreas de modo que los clientes puedan moverse sin cortes

entre un grupo de puntos de acceso. Esto es llamado roaming. Para resolver problemas particulares de topologías, el diseñador de la red puede elegir usar un punto de extensión (EPs) para aumentar el número de puntos de acceso a la red, de modo que funcionen como tales, pero no están enganchados a la red cableada como los puntos de acceso. Los puntos de extensión funcionan, como su nombre indica, extendiendo el alcance de la red, retransmitiendo las señales de un cliente a un punto de acceso o a otro punto de extensión. También pueden encadenarse para pasar mensajes entre un punto de acceso y clientes lejanos de modo que se construye un puente entre ambos.

Uno de los últimos componentes a considerar en el equipo de una WLAN es la antena direccional. Por ejemplo, si se quiere una LAN sin cable a otro edificio que se encuentra a 1 km de distancia, una solución puede ser instalar una antena en cada edificio con línea de visión directa. La antena del primer edificio está conectada a la red cableada mediante un punto de acceso. Igualmente en el segundo edificio se conecta un punto de acceso, lo cual permite una conexión sin cable en esta aplicación.

• *Asignación de canales*

Los estándares 802.11a y 802.11g utilizan la banda de 2,4-2,5 GHz. En esta banda, se definieron once canales utilizables por equipos wifi, los cuales pueden configurarse de acuerdo a necesidades particulares. Sin embargo, los once canales no son completamente independientes (canales contiguos se superponen y se producen interferencias) y en la práctica solo se pueden utilizar tres canales en forma simultánea (1, 6 y 11). Esto es correcto para EE. UU. y muchos países de América Latina, pues en Europa, el ETSI ha definido trece canales. En este caso, por ejemplo en España, se pueden utilizar cuatro canales no adyacentes (1, 5, 9 y 13). Esta asignación de canales usualmente se hace solo en el punto de acceso, pues los "clientes" automáticamente detectan el canal, salvo en los casos en que se forma una red ad hoc, o punto a punto, cuando no existe punto de acceso.

• *Seguridad*

Uno de los problemas de este tipo de redes es, precisamente, la seguridad ya que cualquier persona con una terminal inalámbrica podría comunicarse con un punto de acceso privado si no se disponen de las medidas de

seguridad adecuadas. Dichas medidas van encaminadas en dos sentidos: por una parte, está el cifrado de los datos que se transmiten y, en otro plano, pero igualmente importante, se considera la autenticación entre los diversos usuarios de la red. En el caso del cifrado se están realizando diversas investigaciones ya que los sistemas considerados inicialmente se han conseguido descifrar. Para la autenticación se ha tomado como base el protocolo de verificación EAP (*extensible authentication protocol*), que es bastante flexible y permite el uso de diferentes algoritmos.

• *Velocidad*

Otro de los problemas que presenta este tipo de redes es que, actualmente (a nivel de red local) no alcanza la velocidad que obtienen las redes de datos cableadas. Además, en relación con el apartado de seguridad, el tener que cifrar toda la información supone que gran parte de la información que se transmite sea de control y no información útil para los usuarios, por lo que, incluso, se reduce la velocidad de transmisión de datos útiles y no se llega a tener un buen acceso.

Topologías de red

Habitualmente se entiende por topología de una red a la estructura de la red, es decir, la forma en que se lleva a cabo la conexión. La topología de red se define como una familia de comunicación usada por las computadoras que conforman una red para intercambiar datos. En otras palabras, la forma en que está diseñada la red, sea en el plano físico o lógico. El concepto de red puede definirse como "conjunto de nodos interconectados". Un nodo es el punto en el que una curva se intercepta a sí misma. Lo que un nodo es, concretamente, depende del tipo de redes a la que nos refiramos.

Las topologías más utilizadas son las que siguen:

• Bus.
• Estrella.
• Anillo y anillo doble.
• Árbol.
• Malla.
• Mixtas o híbridas.

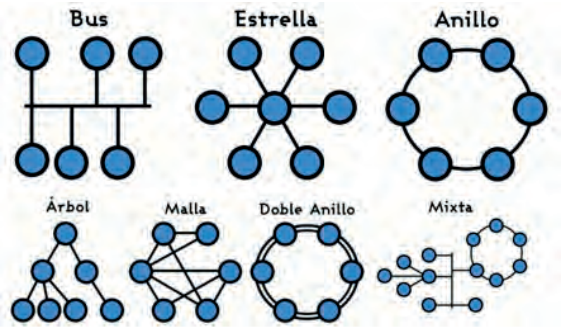


Figura 14.9: Topologías de red

Bus

Una red en bus es aquella topología que se caracteriza por tener un único canal de comunicaciones, denominado bus, troncal o backbone, al cual se conectan los diferentes dispositivos. De esta forma, todos los dispositivos comparten el mismo canal.

Sus extremos terminan con una resistencia de acople denominada terminador, que además de indicar que no existen más ordenadores en el extremo, permite cerrar el bus por medio de un acople de impedancias. Es la tercera de las topologías principales. Las estaciones están conectadas por un único segmento de cable. A diferencia de una red en anillo, el bus es pasivo, no se generan señales en cada nodo o router.

En la topología de bus todos los nodos están conectados a un circuito común (bus). La información que se envía de una computadora a otra, viaja, directa o indirectamente, si existe un controlador que enruta los datos al destino correcto. La información viaja por el cable en ambos sentidos a una velocidad aproximada de 10/100 Mb/s y tiene en sus dos extremos una resistencia (terminador). Se puede conectar una gran cantidad de computadoras al bus, si una falla, la comunicación se mantiene, no sucede lo mismo si el que falla es el bus. El tipo de cableado que se usa puede ser coaxial, par trenzado o fibra óptica. En una topología de bus, cada computadora está conectada a un segmento común de cable de red. El segmento de red se coloca como un bus lineal, es decir, un cable largo que va de un extremo a otro de la red, al cual se

conecta cada nodo. El cable puede ir por el piso, las paredes, el techo o por varios lugares, siempre y cuando sea un segmento continuo.

Ventajas
- Simplicidad en su estructura.
- Fácil de implementarla y de hacerla crecer.
- Fácil adaptación.
- No ocupa mucho espacio

Desventajas
- Hay límite de nodos y está determinado por la calidad de la señal.
- Complejidad para aislar las fallas de la red.
- Si el canal sufre un inconveniente, se afecta toda la red.
- Se pierden muchos paquetes por colisiones de mensaje.

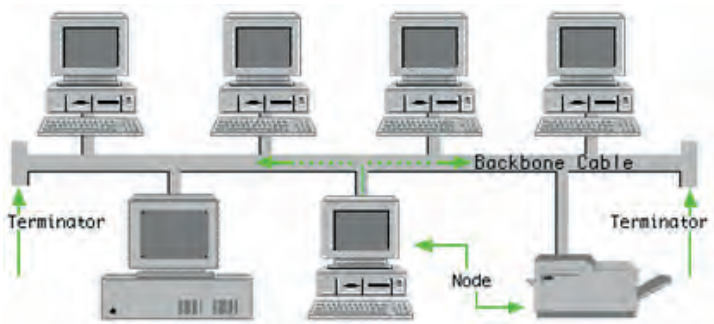


Figura 14.10: Topología de Bus

Estrella

Una red en estrella es una red de computadoras donde las estaciones están conectadas directamente a un punto central y todas las comunicaciones se hacen, necesariamente, a través de ese punto (conmutador, repetidor o concentrador). Los dispositivos no están directamente conectados entre sí, además de que no se permite tanto tráfico de información. Dada su transmisión, una red en estrella activa tiene un nodo central "activo" que, normalmente, tiene los medios para prevenir problemas relacionados con el eco.

Se utiliza sobre todo para redes locales (LAN). La mayoría de las redes de área local que tienen un conmutador (switch) o un concentrador (hub) siguen esta topología. El punto o nodo central en estas sería el switch o el hub, por el que pasan todos los paquetes de usuarios.

Ventajas

- Posee un sistema que permite agregar nuevos equipos fácilmente.
- Reconfiguración rápida.
- Fácil de prevenir daños o conflictos, ya que no afecta a los demás equipos si ocurre algún fallo.
- Centralización de la red.
- Fácil de encontrar fallas.

Desventajas

- Si el hub (repetidor) o switch central falla, toda la red deja de transmitir.
- Es costosa, ya que requiere más cantidad de cables que las topologías en bus o anillo.
- El cable viaja por separado del concentrador a cada computadora.
- Es más costosa que otro tipo de topologías.

Figura 14.11: Topología de estrella

Anillo y anillo doble

Una red en anillo es una topología de red en la que cada estación tiene una única conexión de entrada y otra de salida de anillo. Cada estación tiene un receptor y un transmisor que hace la función de traductor, pasando la señal a la siguiente estación.

En este tipo de red la comunicación se da por el paso de un token o testigo, que se puede conceptualizar como un cartero que pasa recogiendo y entregando paquetes de información, de esta manera, se evitan eventuales pérdidas de información debidas a colisiones.

En un anillo doble (token ring), dos anillos permiten que los datos se envíen en ambas direcciones (token passing). Esta configuración crea redundancia (tolerancia a fallos). Una topología en anillo doble, como su nombre lo indica, en vez de tener un solo anillo tiene dos anillos contrincantes para trasmitir la información, donde cada host de la red está conectado a ambos anillos, aunque estos dos no están conectados directamente entre sí. Es análoga la topología del anillo simple, con la diferencia de que, para incrementar la confiabilidad y flexibilidad de la red, hay un segundo anillo redundante que conecta los mismos dispositivos. La particularidad de anillo doble es que cada anillo trabaja por sí mismo; es decir, actúa de forma independiente para que, si uno de los dos sufre algún tipo de daño, el otro siga trabajando y cumpla su función de transmitir la información sin verse afectado por la falta del otro anillo.

Ventajas

- El sistema provee un acceso equitativo para todas las computadoras.
- El rendimiento no decae cuando muchos usuarios utilizan la red.
- Arquitectura muy sólida.
- Sistema operativo caracterizado con un único canal.

Desventajas

- Longitudes de canales (si una estación desea enviar a otra, los datos tendrán que pasar por todas las estaciones intermedias antes de alcanzar la estación de destino).
- El canal usualmente se degradara a medida que la red crece.
- Difícil de diagnosticar y reparar los problemas.
- Si se encuentra enviando un archivo, podrá ser visto por las estaciones intermedias antes de alcanzar la estación de destino.

- La transmisión de datos es más lenta que en las otras topologías (estrella, malla, etcétera), ya que la información debe pasar por todas las estaciones intermedias antes de llegar al destino.

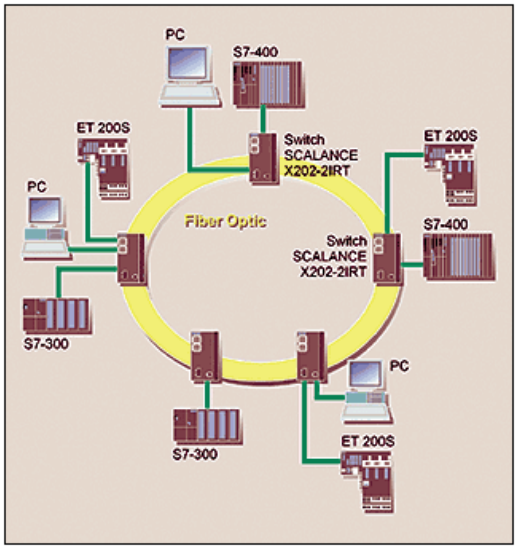


Figura 14.12: Topología de anillo

Árbol

La red en árbol es una topología de red en la que los nodos están colocados en forma de árbol. Se trata de un modelo muy similar a la topología de estrella, con la diferencia de que la primera no tiene un concentrador central. En cambio, tiene un nodo de enlace troncal, generalmente ocupado por un hub o switch, desde el que se ramifican los demás nodos. Es una variación de la red en bus, el fallo de un nodo no implica una interrupción en las comunicaciones. Se comparte el mismo canal de comunicaciones.

La topología en árbol puede verse como una combinación de varias topologías en estrella. Tanto la de árbol como la de estrella son similares a la de bus cuando el nodo de interconexión trabaja en modo difusión, pues la información se propaga hacia todas las estaciones, solo que en esta topología las ramificaciones se extienden a partir de un punto raíz (estrella), a tantas ramificaciones como sean posibles, según las características del árbol.

Los problemas asociados a las topologías anteriores radican en que los datos son recibidos por todas las estaciones sin importar para quién vaya dirigido. Entonces, es necesario dotar a la red de un mecanismo que permita identificar al destinatario de los mensajes, para que estos puedan recogerlos a su arribo. Además, debido a la presencia de un medio de transmisión compartido entre muchas estaciones, pueden producirse interferencias entre las señales cuando dos o más estaciones transmiten al mismo tiempo. Es la mejor topología de red que existe y con ella los datos fluyen de una manera más rápida que en los otros tipos de topologías de red.

Ventajas

- Cableado punto a punto para segmentos individuales.
- Soportado por multitud de vendedores de *software* y de *hardware*.
- Facilidad de resolución de problemas.
- Mucho más rápida que otra.

Desventajas

- Se requiere mucho cable.
- La medida de cada segmento viene determinada por el tipo de cable utilizado.
- Si se cae el segmento principal, todo el segmento también cae.
- Es más difícil su configuración.
- Si se llegara a desconectar un nodo, todos los que están conectados a él se desconectan también.

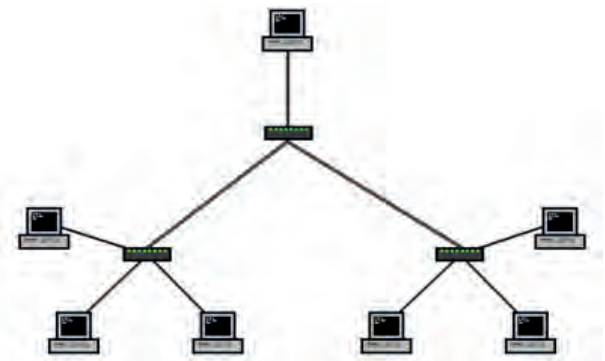


Figura 14.13: Topología de árbol

Malla

Una red en malla es una topología de red en la que cada nodo está conectado a todos los nodos. De esta manera es posible llevar los mensajes de un nodo a otro por distintos caminos. Si la red de malla está completamente conectada, no puede existir absolutamente ninguna interrupción en las comunicaciones. Cada servidor tiene sus propias conexiones con todos los demás servidores.

Esta topología, a diferencia de otras más usuales como la topología en árbol y la topología en estrella, no requiere de un nodo central, con lo que se reduce el riesgo de fallos, y por ende, el mantenimiento periódico (un error en un nodo, sea importante o no, no implica la caída de toda la red).

Las redes en malla pueden prescindir de enrutamiento manual, o apenas requerir atención para su mantenimiento. Si se implementan protocolos de enrutamiento dinámicos, podrían considerarse "autoenrutables", exceptuando escenarios en los que el tamaño o carga de la red son muy variables, o se requiere una tolerancia a fallos prácticamente nula (por ejemplo, debido a la labor crítica que desempeñan algunos de los nodos que la componen).

La comunicación entre dos nodos cualesquiera de una red en malla, puede llevarse a cabo, incluso, si uno o más nodos se desconectan de esta de forma imprevista, o si alguno de los enlaces entre dos nodos adyacentes falla, ya que el resto evitará el paso por ese punto. Los nodos adyacentes a un nodo o enlace fallido propagarán un cambio en la tabla de rutas, notificando a nodos contiguos del cambio en la red, y así sucesivamente. En consecuencia, una red en malla resulta muy confiable, ofrece total redundancia y, por tanto, una fiabilidad y tolerancia a fallos superiores. Aunque la facilidad de solución de problemas y el aumento de la confiabilidad son ventajas muy interesantes, estas redes resultan caras de instalar, pues requieren forzosamente la interconexión de cada nodo con los nodos vecinos (aumentando el número de interfaces de las que debe disponer cada nodo) y el coste de la infraestructura —cableado, switches/puentes, repetidoras de señal, puntos de acceso, etcétera— de toda la red. Por ello cobran mayor importancia en el caso de redes, parcial o totalmente, inalámbricas. La redundancia de rutas para un mismo destino compensa una mayor susceptibilidad a fallos, entre otros inconvenientes propios de las redes sin hilos.

Posee las mismas ventajas que el modelo visto anteriormente. Entre sus desventajas se puede citar que el costo de la red puede aumentar en los casos que se implemente en forma alámbrica, la topología de red y sus características implican el uso de una mayor cantidad de recursos.

Figura 14.14: Topología de malla

Mixta o híbrida

En la topología híbrida, o topología mixta, las redes pueden utilizar diversas topologías para conectarse. La topología mixta es una de las más frecuentes y se deriva de la unión de varios tipos de topologías de red, de aquí el nombre de "mixta" o "híbrida". Ejemplos de topologías mixtas: en árbol, estrella-estrella, bus-estrella, etcétera.

Su implementación se debe a la complejidad de la solución de red, o bien al aumento en el número de dispositivos, lo que hace necesario establecer una topología de este tipo. Las topologías mixtas tienen un costo muy elevado debido a su administración y mantenimiento, ya que cuentan con segmentos de diferentes tipos, lo que obliga a invertir en equipo adicional para lograr la conectividad deseada.

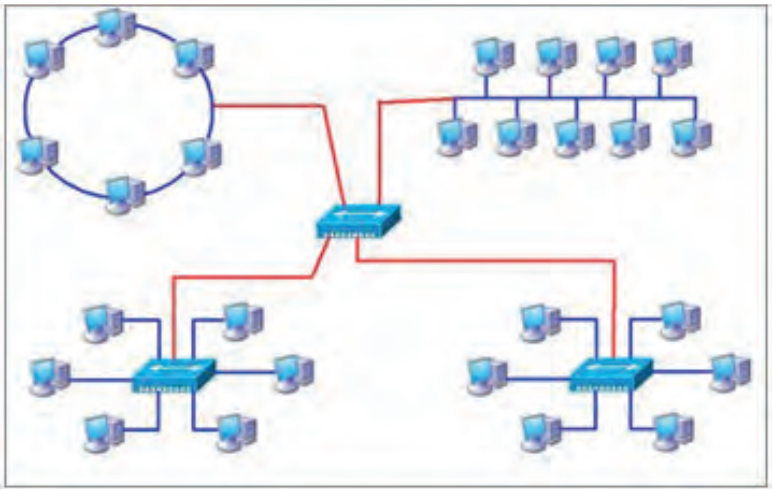


Figura 14.15: Topología de mixta o híbrida

Qué es Internet

Internet es, sin duda, el ejemplo de red de computadoras más conocido en la actualidad. Se compone de miles de millones de dispositivos que al conectarse entre sí forman un grafo conexo, que soportan la ejecución de aplicaciones de red. Para ser más exactos es una red de redes.

Internet es un conjunto de redes de comunicación descentralizadas, pero interconectadas entre sí, que utilizan la familia de protocolos **TCP/IP**, lo cual garantiza que las redes físicas **heterogéneas** que la componen, formen una red lógica única de alcance mundial. Sus orígenes se remontan a 1969, cuando se estableció la primera conexión de computadoras, conocida como ARPANET, entre tres universidades en California (Estados Unidos).

Uno de los **servicios** que más éxito ha tenido en internet ha sido la *world wide web* (WWW o la web), hasta tal punto que es habitual la confusión entre ambos términos. La WWW es un conjunto de **protocolos** que permite, de forma sencilla, la consulta remota de archivos de hipertexto. Esta fue un desarrollo posterior (1990) y utiliza internet como medio de transmisión.

Existen, por tanto, muchos otros **servicios y protocolos** en internet, aparte de la web: el envío de correos electrónicos (SMTP), la transmisión de archivos (FTP y P2P), las conversaciones en línea (IRC), boletines electrónicos

(NNTP), el acceso remoto a otros dispositivos (SSH y Telnet) o algún otro para los juegos en línea.

El uso de internet creció rápidamente en el hemisferio occidental desde la mitad de la década de 1990; y entre finales de la década de 1990 y principios de la década del 2000, en el resto del mundo. En los veinte años desde 1995, el uso de internet se ha multiplicado por 100, cubriendo en 2015 a la tercera parte de la población mundial.

La mayoría de las industrias de comunicación, incluyendo telefonía, radio, televisión, correo postal y periódicos tradicionales están siendo transformadas o redefinidas por Internet, y permitió el nacimiento de nuevos servicios como el streaming de contenido multimedia bajo demanda, servicios de mensajería, internet correo electrónico (email), telefonía por internet, televisión por internet, música digital, y video digital.

Las industrias de publicación de periódicos, libros y otros medios impresos se están adaptando a la tecnología de los sitios web, o están siendo reconvertidos en blogs, web feeds o agregadores de noticias en línea (por ejemplo, Google Noticias). Internet también ha permitido o acelerado nuevas formas de interacción personal por medio de mensajería instantánea, foros de internet y redes sociales. El comercio electrónico ha crecido exponencialmente tanto para grandes cadenas como para pequeñas y medianas empresas o nuevos emprendedores, ya que permite servir a mercados más grandes y vender productos y servicios completamente en línea. Relaciones business-to-business y de servicios financieros en línea han afectado las cadenas de suministro de industrias completas.

La estructura de Internet

La estructura actual de Internet está basada en la interconexión de redes de forma más o menos jerárquica con varios niveles, conocidos como tiers. En general, existen tres niveles conocidos como Tier 1, Tier 2 y Tier 3. Se detallan a continuación las principales características de cada nivel:

- Las redes Tier 1 son las de los grandes operadores globales (Global Carriers) que tienen tendidos de fibra óptica por, al menos, dos continentes. Desde una red Tier 1 se puede acceder a cualquier punto de Internet, gracias a que es una condición necesaria que todas las redes Tier 1 tienen que estar conectadas entre sí. Se puede decir que las redes Tier 1 forman el

actual backbone o troncal de Internet. Algunos ejemplos de compañías que poseen redes Tier 1 son las siguientes:

  - AOL a través de ATDN (AOL Transit Data Network).
  - AT&T.
  - Verizon.
  - Inteliquent.
  - NTT Communications.
  - Telefonica International Wholesale Services (TIWS).

- Las redes Tier 2 son operadores de ámbito más regional que no pueden alcanzar todos los puntos de Internet y que necesitan conectarse a una red Tier 1 para ello. Su principal función es ofrecer servicios de conectividad a los operadores Tier 3. Ejemplos de operadores Tier 2:

  - Cable&Wireless.
  - British Telecom.
  - SingTel (Singapore Telecommunications Limited).

- Las redes Tier 3 pertenecen a los operadores que dan servicio de conexión a Internet a los usuarios residenciales y a muchas empresas, los que conocemos como ISP (*Internet Service Provider*) o proveedores de acceso a Internet. Son ejemplos de ellas:

  - En España: Movistar, Vodafone, Orange, Ono...
  - En Latinoamérica: Movistar, TELMEX, AXTEL, Claro...

En la siguiente figura se muestra la estructura general de interconexión de los diferentes Tier l, en la cual aparecen algunos elementos que se explican en los próximos apartados.

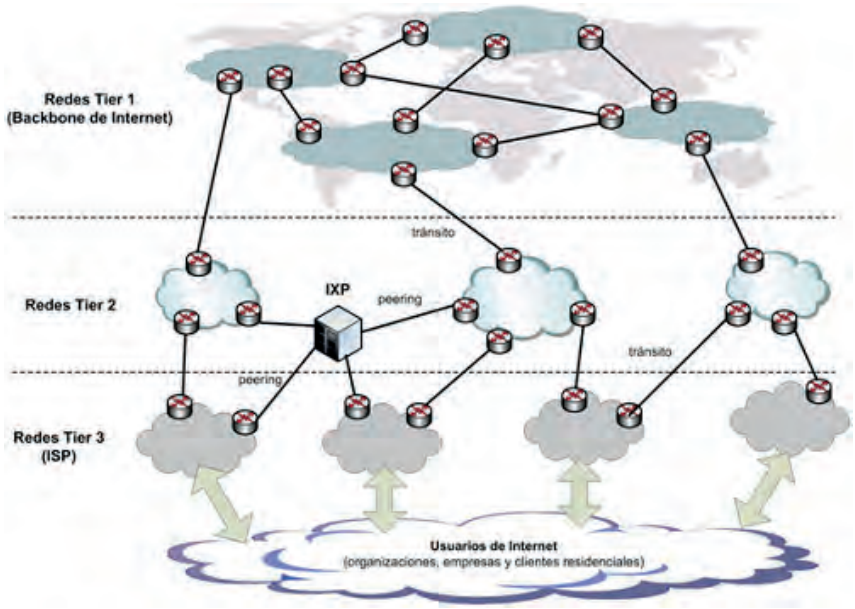


Figura 14.16: Estructura general de interconexión de los diferentes Tier

Tipos de conexiones entre operadores

La conexión entre las redes de diferentes operadores se puede hacer de dos formas:

- **Conexiones de tránsito**: conexión entre operadores de diferente jerarquía. El operador de mayor jerarquía (proveedor) vende una conexión de tránsito al operador de menor jerarquía (cliente). El proveedor le da acceso al cliente a todas sus rutas; es decir, el cliente recibirá tanto las rutas de la red del proveedor como las rutas con destino a otras redes. El cliente publica al proveedor solo sus rutas y no otras que pueda tener con otros proveedores. Por definición, las redes Tier 1 son las únicas que no utilizan conexiones de tránsito.
- **Conexión de peering**: conexión utilizada para el intercambio de tráfico sin costo entre dos operadores. Cada operador publica solo sus rutas y no otras rutas que tenga con otros proveedores u otras rutas de peering; es decir, el peering sirve para acceder desde un operador al rango de

direcciones IP del otro operador, pero no sirve para llegar a otros rangos de direcciones. Puede ser de dos tipos:

- Público: utilizando un IXP (ver el siguiente apartado).
- Privados: conexión directa entre los dos proveedores.

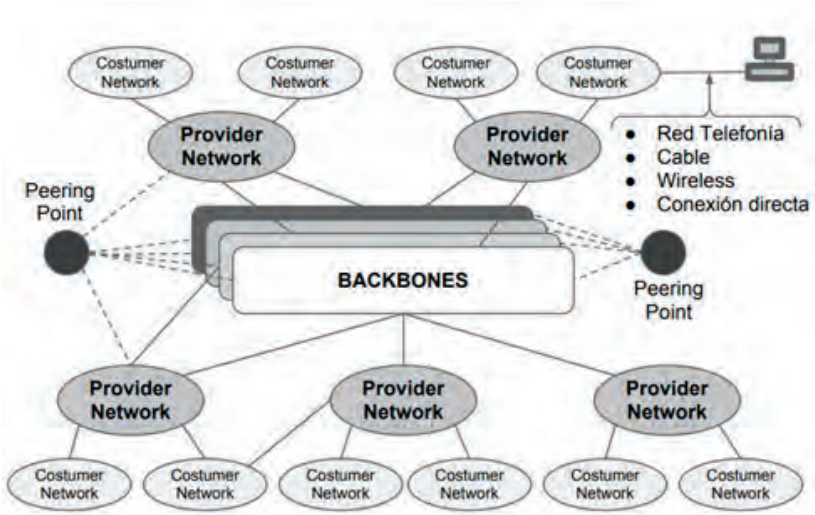


Figura 14.17: Conexión de peering

Puntos de intercambio de tráfico de Internet (IXP)

IXP (*internet exchange point* o punto de intercambio de tráfico de Internet) es una infraestructura física que permite a diferentes ISP intercambiar tráfico de Internet entre sus redes mediante conexiones peering. En realidad, cualquier empresa que quiera establecer una conexión pública de peering con un ISP puede utilizar un IXP.

Habitualmente, los acuerdos de peering entre empresas facilitan el intercambio más eficiente de datos entre sus redes, es por ello que los IXP han tenido un impacto muy beneficioso en el crecimiento de Internet. En Europa existe una asociación de IXP llamada Euro-IX (www.euro-ix.net) que agrupa a todos los IXP europeos y algunos IXP de Japón y Estados Unidos.

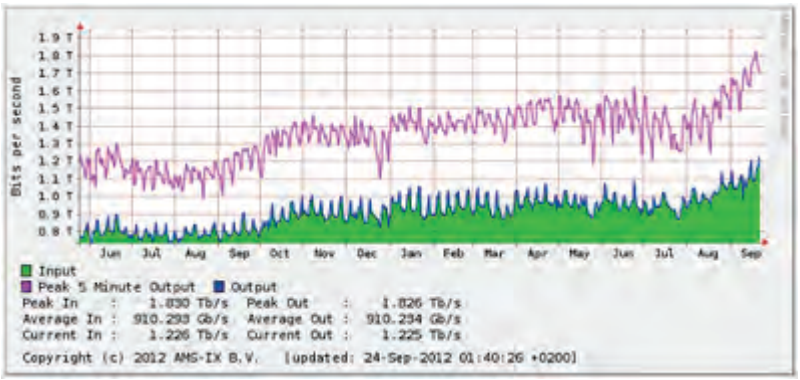


Figura 14.18: Tráfico gestionado por uno de los mayores IXP europeos AMS-IX situado en Amsterdam

Argentina

Para que todo esto sea posible, es decir, para poder conectarse entre redes y formar la red mundial, el mundo está interconectado mediante un sistema de cables submarinos de fibra óptica que cruzan los océanos y conectan los distintos continentes. Este es el sistema troncal, también está el cableado subterráneo que posee cada país. Esta es la forma más segura, eficiente y "económica" que se tiene para poder formar la red. Un corte de uno de estos cables puede dejar incomunicadas a millones de personas, aunque por ello siempre hay enlaces alternativos que se encargan de seguir proporcionando conexiones en cuyo caso, a lo sumo, se verá afectada la calidad de conexión. Hoy en día el 97% del tráfico de Internet se realiza por estos cables, el resto corresponde a enlaces inalámbricos y satelitales.

En Argentina la entrada de este cable se encuentra en una localidad llamada Las Toninas en la provincia de Buenos Aires. Se escogió esta zona ya que posee un lecho submarino libre de piedras lo cual es favorable para que el cable no resulte dañado. Puede ver más información actualizada en:

https://www.submarinecablemap.com/

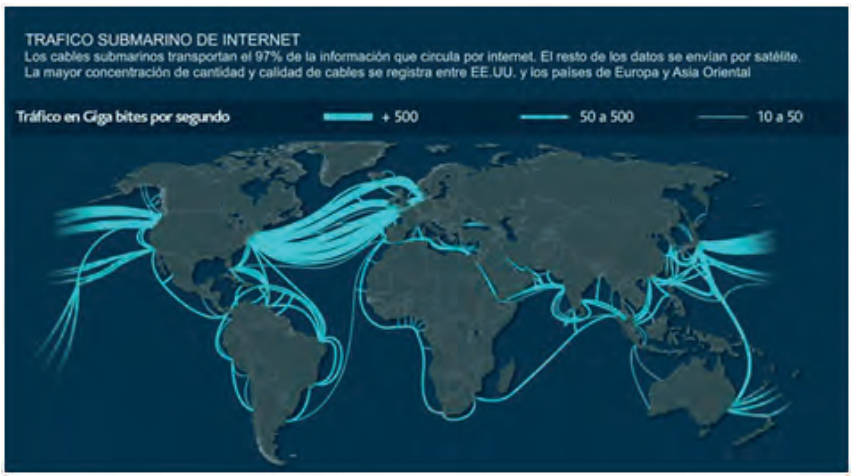


Figura 14.19: Tráfico submarino de Internet

Plan Federal de Internet

El Ministerio de Comunicaciones, por medio de la empresa de telecomunicaciones del Estado, Soluciones Satelitales (Arsat S. A.), trabajó para que 1300 localidades accedieran en 2018 a la red troncal de fibra óptica más extensa y de mayor capacidad del país: la Red Federal de Fibra Óptica. Gracias a ello, casi 10 millones de argentinos pueden acceder ahora a servicios de Internet de banda ancha de alta calidad y precio accesible. Arsat presta servicios de transporte de datos e Internet mayorista a los proveedores locales, que son los encargados de realizar el tendido y conexión domiciliaria.

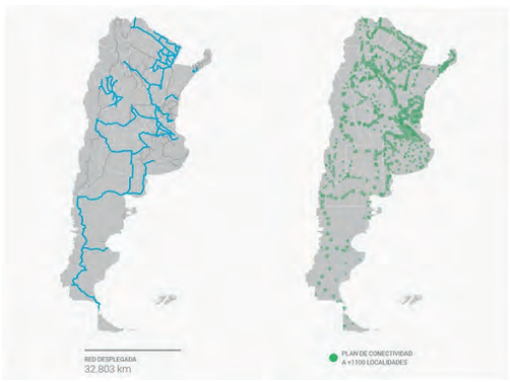


Figura 14.20: Plan Federal de Internet en Argentina

Medios de acceso

La parte de las redes que conecta los usuarios finales (residenciales o corporativos) a las redes de las operadoras de telecomunicaciones se conoce como "red de acceso", aunque también está muy extendida la denominación "última milla". El término de última milla se comenzó a utilizar en telefonía para referirse a la conexión entre el abonado y la central telefónica, también llamada bucle de abonado. Todas las conexiones entre los abonados y las centrales forman la red de acceso, mientras que las conexiones entre las distintas centrales de diferente jerarquía forman la red de transporte. Estos términos se pueden aplicar de igual manera a las redes telemáticas actuales.

Red telefónica computada (RTC)

Hoy en día es una red obsoleta, ya que utiliza la línea telefónica. La conexión se realiza con un modem por medio de una llamada telefónica (Dial-Up) convirtiendo la señal análoga (sonido) en digital para poder recibir los datos, y viceversa, para su envío.

Fue el primer sistema doméstico que se utilizó, alcanzando velocidades de hasta 56 Kb/s (0,056 Mb/s). Era una eternidad poder navegar o descargar algo de internet a pesar de que en su época las páginas eran muy sencillas y livianas. La ventaja de este tipo de conexión es que solo era necesario disponer de una línea telefónica (no requería infraestructura adicional) mientras estuviese disponible el servicio. La gran desventaja es que al estar conectado a internet no se podía utilizar el teléfono ya que no soporta la transmisión simultánea de voz y datos.

Existía una red digital RDSI que solventaba estos problemas de transmisión dual, pero para ello requería de infraestructura adicional alcanzando velocidades de hasta 128 Kb/s (0,128 Mb/s). Se desconoce si en Argentina llegó a desplegarse este servicio.

ADSL

Este tipo de conexión conjuga la conexión RTC y la RSD. ADSL (*asymmetric digital subscriber line*) es la primera "banda ancha" a la cual se accedió. El acceso sigue siendo por medio de la línea telefónica, pero permite utilizar internet y al mismo tiempo poder realizar comunicaciones telefónicas. Como la línea ADSL es "asimétrica", la capacidad de descarga es superior a la de subida. Esto se debe a que, la mayoría del tiempo utilizando internet, es superior la cantidad de datos descargados en comparación a los subidos.

Figura 14.21: Conexión ADSL

Internet inalámbrico

Esta es una buena opción cuando no se accede a un servicio cableado. Para ello el proveedor dispone en la zona de cobertura de antenas y nodos enlazados que mediante ondas de radio posibilita la transmisión de datos. Al usuario se le instala una antena exterior y por medio de un cable llega a un router wifi para acceder a internet.

La principal ventaja de este servicio es que no depende de cableados, su instalación es rápida y relativamente económica. Entre sus inconvenientes se encuentran las velocidades limitadas que rondan en un promedio máximo de 8 Mb/s, la visión entre la antena exterior y el nodo debe estar limpia sin nada que se interponga, las inclemencias climáticas suelen afectar el servicio al igual que los cortes de energía, ya que la antena establece conexión son un solo nodo sin posibilidad de utilizar alguno alternativo. El costo del servicio suele ser alto comparado al que ofrece un servicio cableado, no obstante, no deja de ser una muy buena opción.

Figura 14.22: Internet inalámbrico

Internet satelital

Este sistema es poco utilizado a un nivel doméstico ya que el costo es elevado y la velocidad es baja con mucha latencia. Para lugares remotos que no dispongan de cables o antenas esta termina siendo su única opción. Además de ello es necesario disponer de antenas parabólicas que tienen un costo elevado.

Figura 14.23: Internet satelital

Internet móvil

En los últimos años ganó mucho mercado por el uso de los Smartphone. La comunicación es inalámbrica y depende no solo de la cobertura de las antenas sino, también, de la disponibilidad del servicio.

El primer servicio de este tipo fue el WAP, luego fue evolucionando a GPRS (2G), UMTS (3G), HDSPA, y lo último, y vigente, LTE (4G) que alcanza velocidades promedio de hasta 17 Mb/s en Argentina. Singapur cuenta con la mayor velocidad que ronda en 44 Mb/s. Se está implementando el servicio LTE Advanced (4G+ o Carrier Aggregation), pero requiere de equipos homologados por los proveedores. Este tipo de equipos permite la conexión con dos antenas en simultáneo, utilizando frecuencias diferentes con la posibilidad de duplicar el ancho de banda.

Cabe aclarar que necesita, como requisito, que las antenas trabajen a frecuencias diferentes, esto limita las posibilidades ya que no en todas las ciudades hay antenas que trabajen en distintas frecuencias.

El desarrollo de internet móvil dio a lugar a que las páginas web también se programen con una interface más liviana y que permita una mejor experiencia de navegación desde un Smartphone.

Su mayor ventaja es disponer del acceso desde cualquier lugar. Su contra, que depende mucho de la infraestructura instalada y que en los planes es limitada la cantidad de datos, lo que lo convierte en un producto caro para un uso intensivo.

Figura 14.24: Internet móvil

Cable

Independientemente del tipo de acceso a Internet, cuando hablamos de redes, el cable es uno de los medios más comunes, y según el caso, más económico para las conexiones.

Cable Coaxial

El cable coaxial consiste de un núcleo de cable de cobre rodeado de un material aislante que, a su vez, está recubierto por una malla conductora de cobre tejida. Esta malla se cubre con una envoltura de plástico. La malla de cobre le confiere un muy buen aislamiento que permite un elevado ancho de banda, buena inmunidad al ruido y mayores distancias de tendido.

Un problema con este tipo de cable son los conectores. Como la malla de cobre que recubre el cable es parte del circuito, este debe estar correctamente colocado, garantizando su conexión a tierra.

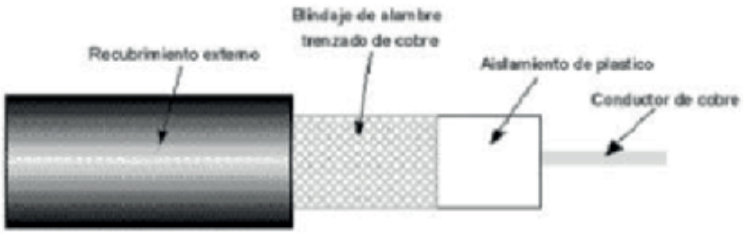


Figura 14.25: Cable coaxil

Figura 14.26: Conector coaxil

Par trenzado no blindado

El cable de par trenzado no blindado UTP está conformado por cuatro pares de hilos. Cada uno de los hilos de cobre está revestido por un material aislante, y cada par de hilos se encuentra trenzado para disminuir los problemas de interferencia.

Existen varias categorías de cables UTP, los más modernos son UTP5 y UTP6. El que se utiliza en las instalaciones de redes tiene una impedancia de 100 ohmios. Esto lo diferencia de los cables de par trenzado utilizados en tendidos de redes telefónicas.

Ventaja: su diámetro pequeño beneficia las tareas de instalación.

Desventaja: es más susceptible al ruido eléctrico y a interferencias que otros medios. Además, la distancia que puede abarcar la señal sin el uso de repetidores es más limitada que en los casos de cables coaxiales o fibra óptica.

En la actualidad, el cable UTP es el más popular para la instalación de redes de área local.

Par trenzado blindado

El cable de par trenzado blindado STP está conformado por cuatro pares de hilos. Cada par de hilos está trenzado y envuelto con un papel metálico, y a la vez, los cuatro pares se encuentran envueltos en una malla metálica.

Otra alternativa es el cable de par trenzado apantallado FTP, compuesto por los cuatro pares sin apantallar envueltos en una malla metálica o un papel metálico.

En ambos casos, los conectores que se utilizan son el estándar de cable UTP, es el conector denominado RJ-45. Se trata de un conector de plástico similar al conector del cable telefónico (RJ-11). La sigla RJ se refiere al estándar *registred jack*, creado por la industria telefónica. Este estándar define la colocación de los cables en su pin correspondiente.

Figura 14.27: Cable y conector UTP

Fibra óptica

La fibra óptica hoy en día es lo último que se está comenzando a ofrecer. A diferencia del sistema de cable mixto, la bajada se realiza con fibra.

Al utilizar este sistema, el ancho de banda ya no es un problema y la velocidad de subida y bajada pueden llegar a ser iguales. Teniendo en cuenta que la transmisión de datos sobre sistemas tradicionales, como son los cables de cobre, permiten llegar a velocidades máximas de 100 Mb/s, la fibra óptica permite llegar a 10 Gb/s y continúa creciendo. Por supuesto, estos datos son a nivel global y varían en cada país.

El cable está compuesto por uno o más filamentos de silicio o vidrio por el que se envían pulsos de luz que representan los datos a transmitir. Al ser

de vidrio, el filamento tiene la propiedad de no sufrir interferencias electromagnéticas lo que le da una gran ventaja sobre cables de cobre.

Pero su uso no solo se limita al internet, el cable también se utiliza, entre otras cosas, para señales telefónicas y de cablevisión, transmisión de luz, en aplicaciones medicinales y militares (al no ser metálico, es difícil detectarlo y no emite radiación electromagnética lo que aumenta la seguridad en las comunicaciones).

Ventajas: los datos se transmiten con excelente velocidad sin limitar el ancho de banda, es más liviano y de tamaño pequeño. El material para producirlo es abundante en la naturaleza y no requiere de mucha cantidad. La transmisión de audio y video se realiza, prácticamente, en tiempo real. Son difíciles de detectar y no los afectan las ondas electromagnéticas.

Desventajas: el elevado costo de instalación, la disponibilidad, son delicados de manipular y difíciles de reparar, para lo que se requiere de personal sumamente capacitado.

Se basa en un servicio de conexión mediante fibra óptica (FTTH) que, como se explicó anteriormente, mejora notablemente la experiencia de acceso a internet con paquetes con velocidades de 20, 50, 100 y 300 Mb/s. Dentro del mercado de servicios de internet, la fibra hoy es la mejor opción, se asegura que la velocidad contratada sea real y se eliminan los cuellos de botella generados por la falta de ancho de banda. Con esto se podrá bajar música, ver videos, realizar videollamadas, entrar en redes sociales, todo al mismo tiempo, sin pausas. El módem que se instala es de banda dual, con lo cual, aparte de la tradicional banda 2,4 GHz, también dispone de la banda de 5 GHz con transmisión en simultáneo, que no suelen estar congestionadas (punto a favor para las grandes ciudades) y alcanzan mayores velocidades mediante el uso simultáneo de varios canales. El servicio se complementa con llamadas locales ilimitadas a teléfonos fijos.

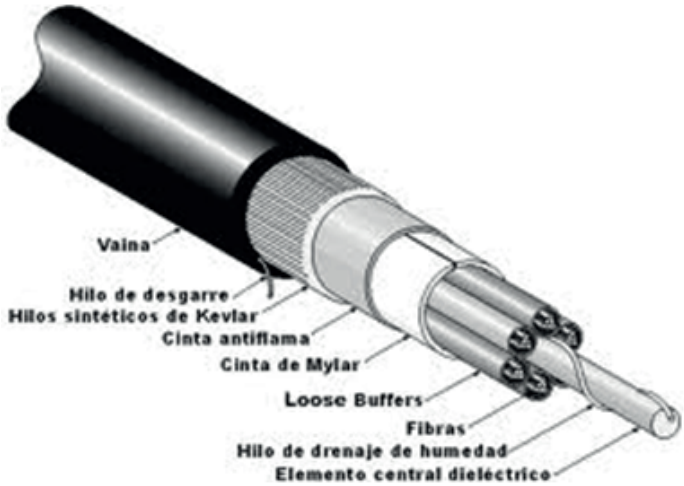


Figura 14.28: Cable de fibra óptica

Modelos de referencia ISO/OSI Y TCP/IP

Modelos divididos en capas

Un modelo dividido en capas permite dividir un problema complejo en partes más pequeñas y resolver, independientemente, cada uno de dichos problemas. La división en capas se aplica tanto en el origen como en el destino y en formas pares.

Con el objetivo principal de reducir la complejidad de su diseño, las redes de datos están organizadas en una serie de capas, cada una desarrollada sobre una capa inferior y con una función determinada.

Las reglas que rigen la comunicación entre estas capas se denomina **protocolo**, el cual no debe violarse si se desea el éxito de la comunicación.

Una red puede estar compuesta por múltiples computadoras, cada una de las cuales ejecuta múltiples procesos. Esto obliga a contar con un mecanismo de direccionamiento que identifique qué computadoras y qué procesos desean comunicarse.

Protocolo

Un protocolo es un conjunto de reglas y convenciones entre las partes que se comunican referido a cómo va a proceder la comunicación.

Interfaz

Entre cada par de capas adyacentes hay una interfaz que define cuáles operaciones y servicios ofrece la capa inferior a la superior. Una interfaz bien definida minimiza la cantidad de información que se debe pasar entre las capas.

Encapsulamiento

Los datos no se transfieren directamente de la capa n de una computadora a la capa n de la otra, sino que cada capa pasa datos e información de control (denominada encabezado) a la capa inferior adyacente, hasta llegar a la capa inferior del modelo, debajo de la cual se encuentra el medio físico a través del cual se transferirán los datos.

Arquitectura de red

Una arquitectura de red está compuesta por un conjunto de capas y protocolos. La información que provee una arquitectura de red debe permitirle a un implementador desarrollar el *hardware* y el *software* necesario para que cada capa obedezca en forma correcta el protocolo correspondiente.

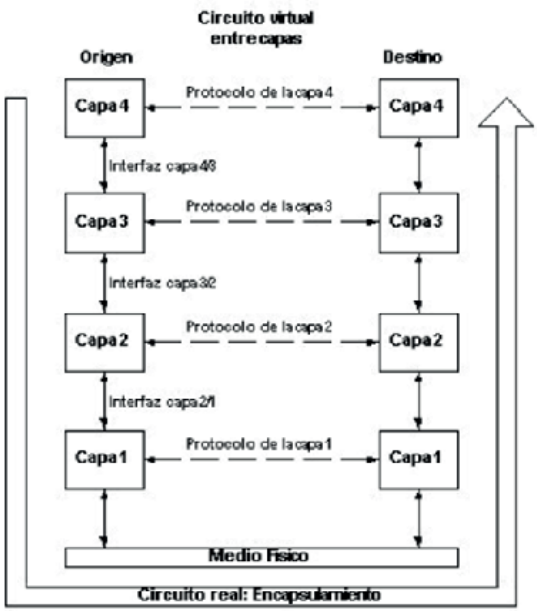


Figura 14.29: Modelo de referencia genérico

Modelo de referencia OSI

El modelo OSI (*open system interconnection*) describe cómo se mueve a través de una red la información de una aplicación de *software* de una computadora hasta otra aplicación de *software* de otra computadora.

Es un modelo conceptual compuesto de siete capas, donde cada una de ellas está encargada de ciertas funciones específicas. El modelo fue desarrollado por la Organización Internacional para la estandarización (ISO, en inglés) en 1984 y, actualmente, se considera como el principal modelo de arquitectura para comunicación entre computadoras.

El modelo divide las tareas involucradas en el movimiento de información entre dos equipos pertenecientes a una red, en siete grupos más pequeños de tareas. Se asigna, entonces, una tarea o grupo de tareas a cada una de las capas del modelo y la característica principal es que cada capa ofrece servicios a la capa inmediatamente superior, y a la vez utiliza los servicios que le ofrece la capa inmediatamente inferior.

Las siete capas que conforman el modelo son las siguientes: capa física, capa de enlace de datos, capa de red, capa de transporte, capa de sesión, capa de presentación y capa de aplicación.

Características de las capas OSI

Las siete capas del modelo OSI se pueden dividir en dos categorías: las capas superiores y las capas inferiores. Las primeras se ocupan de los asuntos relacionados con la aplicación y, en general, están implementadas solo en *software*. La capa de más arriba, la de aplicación, es la más cercana al usuario final. Tanto los procesos del usuario como los de la capa de aplicación interactúan con aplicaciones de *software* que contienen una componente de comunicaciones. Las capas inferiores se ocupan del transporte de datos. Las capas física y de enlace de datos se implementan en *hardware* y *software*. La capa de más abajo, la física, es la más cercana al medio físico de la red (a los cables, por ejemplo) y es la responsable de poner la información en el medio de transmisión.

Protocolos

El modelo OSI proporciona un marco conceptual para la comunicación entre computadoras, pero el modelo en sí no es un método de comunicaciones. El intercambio de información se hace posible gracias a la utilización de protocolos de comunicación.

Como ya se mencionó en el contexto de redes de datos, un protocolo es un conjunto de reglas y convenciones que definen la forma en que los equipos intercambian la información a través de la red. Un protocolo implementa las funciones de una o más capas del modelo OSI.

Existe una amplia gama de protocolos de comunicaciones. Algunos de ellos son protocolos LAN, protocolos WAN, protocolos de ruteo, y protocolos de red. Los primeros operan en la capa física y la de enlace de datos; los segundos, en las tres capas inferiores; los terceros pertenecen a la capa de red y son responsables del intercambio de información entre los *routers* para que estos puedan seleccionar el camino correcto para el tráfico de la red. Por último, los protocolos de red son varios protocolos que están en un determinado stack. Muchos de estos dependen de otros para su correcta operación. Por ejemplo, ciertos protocolos de ruteo utilizan protocolos de red para el intercambio de información entre routers.

Comunicación entre sistemas

La información que se transfiere de una aplicación de *software* en una computadora, a otra aplicación de *software* en otra, debe pasar por todas las capas del modelo OSI. Por ejemplo, si la aplicación en el sistema A quiere transmitir información a una aplicación del sistema B, el programa en el sistema A pasa la información a la capa de aplicación (capa 7) del sistema A.

La capa de aplicación, luego, le pasa la información a la capa de presentación (capa 6), la que a su vez, se la pasa a la capa de sesión (capa 5) y, así sucesivamente, hasta llegar a la capa física (capa 1). Luego, la información se pone en el medio físico, y se envía a través de esta al sistema B. La capa física del sistema B rescata la información del medio físico y se la pasa a la capa de enlace de datos (capa 2), la que a su vez se la pasa a la capa de red (capa 3) y, así sucesivamente, hasta que llega a la capa de aplicación (capa 7) del sistema B. Después de que esto sucede, la capa de aplicación le pasa la información al programa de aplicación, y entonces, se completa el proceso de comunicación.

Interacción entre capas del modelo OSI

Una determinada capa del modelo OSI, generalmente, se comunica con otras tres capas: la capa inmediatamente superior, la capa inmediatamente inferior y su capa par en el sistema con el que se está comunicando. Por ejemplo, la capa de enlace de datos en el sistema A se comunica con la capa física

del sistema A, con la capa de red del sistema A, y con la capa de enlace de datos del sistema B.

Servicios de las capas OSI

Una capa OSI se comunica con otra para hacer uso de los servicios que la segunda le puede proporcionar. Los servicios proporcionados por las capas adyacentes ayudan a una determinada capa a comunicarse con su par en otro sistema. Hay tres elementos básicos involucrados en los servicios: el usuario del servicio, el proveedor del servicio y el punto de acceso al servicio o *service access point* (SAP). En este contexto, el usuario del servicio es la capa que solicita un servicio de una capa inferior. El proveedor del servicio es la capa que proporciona el servicio al usuario. Las capas pueden proporcionar servicios a múltiples usuarios. El SAP es la ubicación conceptual en que una capa solicita los servicios de otra.

Capas OSI e intercambio de información

Las siete capas OSI utilizan varias formas de información de control para comunicarse con su capa par en otro computador. Esta información de control consiste en peticiones e instrucciones que son intercambiadas entre capas pares. La información de control típicamente toma una de dos formas posibles: *headers* o *trailers*. Los headers se agregan al comienzo a los datos a medida que pasan a las capas inferiores.

Los trailers son agregados al final a los datos que pasan de arriba hacia abajo. Los headers, trailers y los datos son conceptos relativos, dependiendo de la capa que analice la unidad de información. En la capa de red, por ejemplo, una unidad de información consta de un header de capa 3 más los datos. En la capa de enlace de datos, sin embargo, toda la información recibida de la capa de red (header de capa 3 y datos) es tratado como dato. En otras palabras, la porción de datos de una unidad de información en una determinada capa puede contener headers, trailers y datos de las capas superiores a ella. Esto se conoce con el nombre de encapsulación.

Proceso de intercambio de información

El proceso de intercambio de información ocurre entre capas pares del modelo OSI. Cada capa en el sistema fuente agrega información de control a los datos, y cada capa en el equipo de destino analiza y remueve la

información de control de los datos. Si el sistema A tiene datos de una aplicación para enviárselos a B, estos se pasan a la capa de aplicación.

Esta capa del sistema A comunica cualquier información de control requerida por la misma capa del sistema B, agregándole un encabezado a los datos. La unidad de información resultante (encabezado más datos) es pasada a la capa de presentación, la que le agrega su propio encabezado con información de control para la capa de presentación del sistema B. La unidad de información va creciendo a medida que cada capa va agregando su encabezado (y en algunos casos un trailer que va al final del paquete) con información de control que será utilizada por su capa par en el equipo B.

En la capa física, la unidad de información completa se transmite a través del medio. La capa física del sistema B recibe la unidad de información y la pasa a la capa de enlace de datos. Esta revisa la información de control contenida en el encabezado. Luego, se remueve el encabezado, y el resto se pasa a la capa de red. Cada capa realiza las mismas acciones: primero, lee la información del encabezado enviada por su capa par; después, remueve el encabezado; y finalmente, pasa la información que queda a la capa inmediatamente superior. Una vez que la capa de aplicación realiza estas acciones, los datos pasan a la aplicación de *software* en el sistema B tal cual fueron transmitidos por la aplicación en el sistema A.

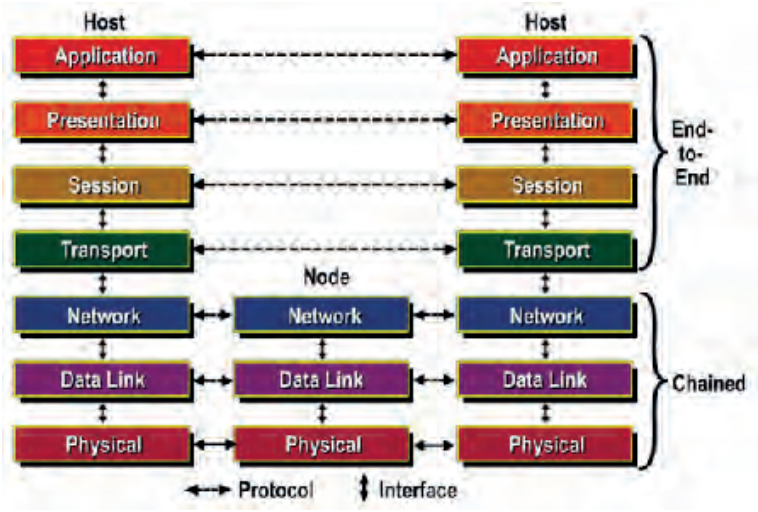


Figura 14.30: Modelo de referencia OSI

Descripción de las capas del modelo

En esta sección se describen las características y funciones más importantes de cada una de las siete capas que conforman el modelo OSI.

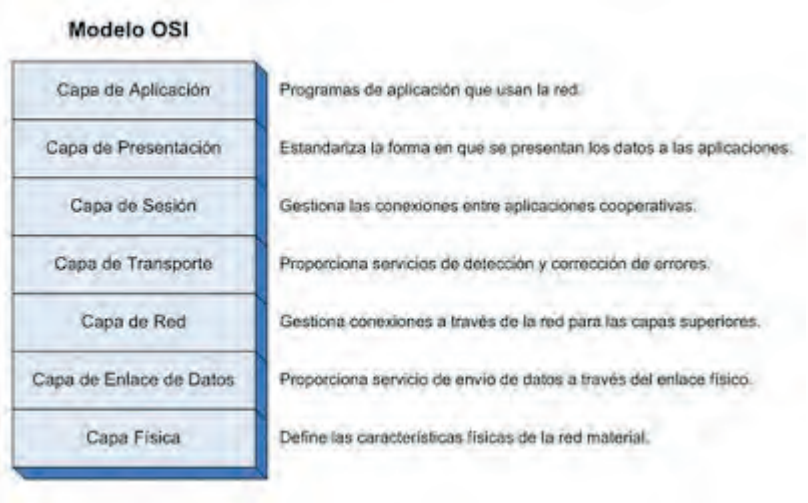


Figura 14.31: Capas del modelo de Referencia OSI

Capa física

Esta capa define las especificaciones eléctricas, mecánicas y funcionales para activar, mantener, y desactivar el enlace físico entre sistemas. Estas especificaciones definen características como el nivel de voltaje, tiempo entre cambios de voltaje, tasas de datos a nivel físico, distancias máximas de transmisión, y conectores. Esta capa le presta servicios a la capa de enlace de datos.

La capa física del modelo de referencia OSI es la que se encarga de las conexiones físicas de la computadora hacia la red, tanto en lo que se refiere al medio físico (medios guiados: cable coaxial, cable de par trenzado, fibra óptica y otros tipos de cables; medios no guiados: radio, infrarrojos, microondas, láser y otras redes inalámbricas); características del medio (por ejemplo, tipo de cable o calidad del mismo, tipo de conectores normalizados o, en su caso, tipo de antena, etcétera) y la forma en la que se transmite la información (codificación de señal, niveles de tensión/intensidad de corriente eléctrica, modulación, tasa binaria, y demás).

Es la encargada de transmitir los bits de información a través del medio utilizado para la transmisión. Se ocupa de las propiedades físicas y características eléctricas de los diversos componentes; de la velocidad de transmisión, si esta es unidireccional o bidireccional (símplex, dúplex o full-dúplex). También de aspectos mecánicos de las conexiones y terminales, incluyendo la interpretación de las señales eléctricas/electromagnéticas.

Se encarga de transformar una trama de datos proveniente del nivel de enlace en una señal adecuada al medio físico utilizado en la transmisión. Estos impulsos pueden ser eléctricos (transmisión por cable) o electromagnéticos (transmisión sin cables). Estos últimos, dependiendo de la frecuencia/longitud de onda de la señal pueden ser ópticos, de microondas o de radio. Cuando actúa en modo recepción el trabajo es inverso, se encarga de transformar la señal transmitida en tramas de datos binarios que serán entregados al nivel de enlace.

Capa de enlace de datos

La capa de enlace de datos se ocupa del direccionamiento físico, de la topología de la red, del acceso a la red, de la notificación de errores, de la distribución ordenada de tramas y del control del flujo.

Se hace un direccionamiento de los datos en la red, ya sea en la distribución adecuada desde un emisor a un receptor, la notificación de errores, de la topología de la red de cualquier tipo. La tarjeta NIC (*network interface card*, tarjeta de interfaz de red o tarjeta de red en español) que se encarga de que tengamos conexión, posee una dirección MAC (control de acceso al medio) y la LLC (control de enlace lógico). La IEEE ha subdividido la capa de enlace de datos en dos subcapas: LLC (*logical link control* o control de enlace lógico) y MAC (*media access control* o control de acceso al medio).

La subcapa LLC maneja las comunicaciones entre equipos sobre un enlace de red. Se define en la especificación IEEE 802.2 y soporta servicios orientados y no orientados a la conexión utilizados por protocolos de capas superiores. El IEEE 802.2 define un número de campos en los frames de la capa de enlace que permiten que múltiples protocolos de las capas superiores compartan un solo enlace físico.

La subcapa MAC maneja el acceso al medio físico. La especificación IEEE define las direcciones MAC, que permiten que múltiples equipos se identifiquen unívocamente unos a otros en la capa de enlace. Esta capa se encarga de

que los datos sean enviados libres de errores a su destino. Diferentes especificaciones para la capa de enlace de datos definen distintas características de red y de protocolos, incluyendo direccionamiento físico, topología de red, notificación de errores, secuenciado de frames y control de flujo.

El direccionamiento físico (en contraposición al direccionamiento de red) define cómo son las direcciones de los equipos en la capa de enlace. La topología de red consiste en las especificaciones de capa de enlace que a menudo definen como deben estar conectados físicamente los equipos, como por ejemplo, una topología de bus o de estrella. La notificación de errores alerta a los protocolos de las capas superiores que ha ocurrido un error en la transmisión, y el secuenciado de frames los reordena cuando son transmitidos desordenados. Por último, el control de flujo modera la transmisión de datos para que el equipo receptor no sea sobrecargado con más tráfico del que puede procesar. Los switches realizan su función en esta capa.

Capa de red

La capa de red define las direcciones de red, que son distintas a las direcciones MAC. Algunas implementaciones de la capa de red, como el protocolo IP (*internet protocol*), definen las direcciones de red de manera tal que la selección de ruteo puede ser determinada en forma sistemática comparando la dirección de origen con la de destino y haciendo uso de la máscara de subred. Los *routers* utilizan esta capa para determinar cómo rutear los paquetes. Debido a esto, gran parte del trabajo de configuración y diseño de las redes sucede en esta capa.

El cometido de la capa de red es hacer que los datos lleguen desde el origen al destino, aun cuando ambos no estén conectados directamente. Los dispositivos que facilitan tal tarea se denominan en castellano encaminadores, aunque es más frecuente encontrar el nombre inglés *router* y, en ocasiones enrutadores.

Adicionalmente, la capa de red lleva un control de la congestión de red, que es el fenómeno que se produce cuando la saturación de un nodo tira abajo toda la red (similar a un atasco en un cruce importante en una ciudad grande). La PDU de la capa 3 es el PAQUETE.

Los routers trabajan en esta capa, aunque pueden actuar como switch de nivel 2 en determinados casos, dependiendo de la función que se le asigne. Los firewalls actúan sobre esta capa principalmente, para descartar

direcciones de máquinas. En este nivel se determina la ruta de los datos (direccionamiento lógico) y su receptor final IP.

Los protocolos centrales de la capa de Internet son IP, protocolo de resolución de direcciones (ARP), protocolo de mensajes de control de Internet (ICMP) y protocolo de administración de grupos de Internet (IGMP). En esta capa, el IP agrega la cabecera a los paquetes, lo que se conoce como dirección IP. En la actualidad, existen tanto dirección IPv4 (32 bits) como dirección IP IPv6 (128 bits).

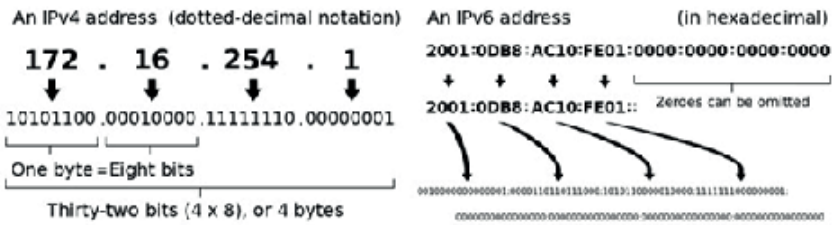


Figura 14.32: Capa de red del modelo OSI

Capa de transporte

Su función básica es aceptar los datos enviados por las capas superiores, dividirlos en pequeñas partes si es necesario, y pasarlos a la capa de red. La capa de transporte recibe los datos de la capa de sesión y los segmenta para que sean transportados a través de la red. Generalmente, esta capa es la responsable de asegurarse de que los datos sean recibidos libres de error y en la secuencia que corresponde."

El control de flujo ocurre generalmente en la capa de transporte. Este maneja la transmisión de datos entre equipos para que el que está transmitiendo no envíe más datos que los que el receptor puede procesar. La multiplexación permite que datos de varias aplicaciones sean transmitidos sobre un solo enlace físico. En esta capa se establecen circuitos virtuales. La revisión de errores involucra la creación de diversos mecanismos para detectar errores en la transmisión, mientras que la recuperación de errores involucra la acción, como por ejemplo, solicitar una retransmisión para recuperar los datos con error. Los

protocolos de transporte utilizados en Internet son TCP (*transmission control protocol*) y UDP (*user datagram protocol*). La PDU de la capa 4 se llama SEGMENTOS.

En resumen, podemos definir a la capa de transporte como la "capa encargada de efectuar el transporte de los datos (que se encuentran dentro del paquete) de la máquina origen a la de destino, independizándolo del tipo de red física que se esté utilizando".

Capa de sesión

Esta capa establece, maneja, y termina las sesiones de comunicación. Ellas consisten en solicitudes de servicios y sus respuestas que suceden entre equipos distintos en una red. Estas solicitudes y respuestas son coordinadas por protocolos pertenecientes a la capa de sesión.

Esta capa establece, gestiona y finaliza las conexiones entre usuarios (procesos o aplicaciones) finales. Ofrece varios servicios que son cruciales para la comunicación. En conclusión, esta capa es la que se encarga de mantener el enlace entre los dos computadores que estén trasmitiendo archivos. Los firewalls actúan sobre esta capa para bloquear los accesos a los puertos de un computador. Algunos ejemplos de implementaciones de la capa de sesión incluyen *zone information protocol* (ZIP), el protocolo *apple talk*, el *session control protocol* (SCP), etcétera. En esta capa no interviene el administrador de red.

Capa de presentación

Esta capa proporciona una variedad de funciones de codificación y conversión que se aplican a los datos de la capa de aplicación. Estas funciones garantizan que la información enviada por la capa de aplicación en un sistema sea legible para la capa de aplicación del sistema que la recibe.

Algunos ejemplos de funciones de codificación y conversión de la capa de presentación incluyen formatos comunes de representación de datos, formatos de representación de conversión de caracteres, y esquemas de encriptación de datos.

Los formatos comunes de representación de datos, o el uso de formatos estándar de imágenes, sonido y video, permiten el intercambio de información de aplicación entre distintos tipos de sistemas computacionales. Los esquemas de conversión son utilizados para intercambiar información con sistemas mediante la utilización de distintas representaciones de texto y datos, como EBCDIC y ASCII. Los esquemas estándar de compresión de datos

permiten que datos comprimidos en el origen sean correctamente descomprimidos en el destino. Los esquemas estándar de encriptación de datos permiten que datos encriptados en el origen sean correctamente descifrados en el destino.

Podemos resumir definiendo a esta capa como la encargada de manejar las estructuras de datos abstractas y realizar las conversiones de representación de datos necesarias para la correcta interpretación de los mismos. Esta capa también permite cifrar los datos y comprimirlos. En pocas palabras es un traductor.

Capa de aplicación

Esta capa es la que se encuentra más cerca del usuario final, lo que significa que tanto la capa de aplicación como el usuario final, interactúan con el *software* de aplicación. Interactúa con aplicaciones de *software* que implementan un componente de comunicaciones. Dichos programas de aplicación quedan fuera del alcance del modelo OSI. Las funciones de la capa de aplicación incluyen la identificación del compañero de comunicación, determinación de la disponibilidad de recursos, y sincronización de la comunicación. Algunos ejemplos de implementaciones de la capa de aplicación incluyen Telnet, *file transfer protocol* (FTP) y *simple mail transfer protocol* (SMTP).

Ventajas del modelo OSI

Al dividir el modelo en siete capas se obtienen las siguientes ventajas:

1. Divide la comunicación de red en partes más pequeñas y sencillas.
2. Normaliza los componentes de red para permitir el desarrollo y el soporte de los productos de diferentes fabricantes.
3. Permite a los distintos tipos de *hardware* y *software* de red comunicarse entre sí.
4. Impide que los cambios en una capa puedan afectar las demás capas.
5. Divide la comunicación de red en partes más pequeñas para simplificar el aprendizaje.

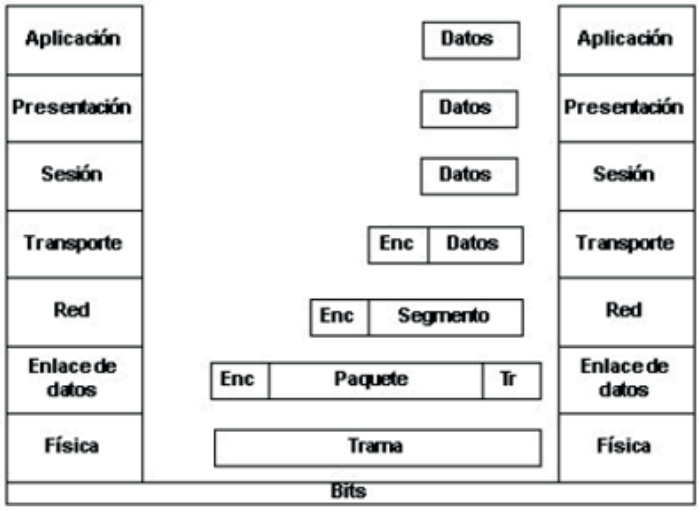


Figura 14.33: Tramas entre las capas del modelo OSI

Modelo de arquitectura de protocolo TCP/IP

Si bien el modelo OSI es ampliamente conocido, la mayoría de los fabricantes lo utilizan y es el estándar, TCP/IP es el protocolo o pila de protocolos que está más ampliamente distribuido para la interconexión de computadoras y es el utilizado por todas las computadoras conectadas a Internet, teniendo un alcance casi universal. TCP/IP es el nombre con que se conoce a un conjunto de protocolos que implementan las distintas funciones de las capas descriptas en el modelo OSI.

Este último modelo describe las comunicaciones de red ideales con una familia de protocolos. TCP/IP no se corresponde directamente con este modelo, combina varias capas OSI en una única capa, o no utiliza determinadas capas. La tabla siguiente muestra las capas de la implementación de Oracle Solaris de TCP/IP. La tabla enumera las capas desde la superior (aplicación) hasta la inferior (red física).

Normalmente, los tres niveles superiores del modelo OSI (aplicación, presentación y sesión) son considerados simplemente como el nivel de aplicación en el conjunto TCP/IP. Como este no tiene un nivel de sesión unificado sobre

el que los niveles superiores se sostengan, estas funciones son típicamente desempeñadas (o ignoradas) por las aplicaciones de usuario. La diferencia más notable entre los modelos de TCP/IP y OSI es el nivel de aplicación; en el primero se integran algunos niveles del modelo OSI en su nivel de aplicación. Una interpretación simplificada de la pila TCP/IP se muestra debajo:

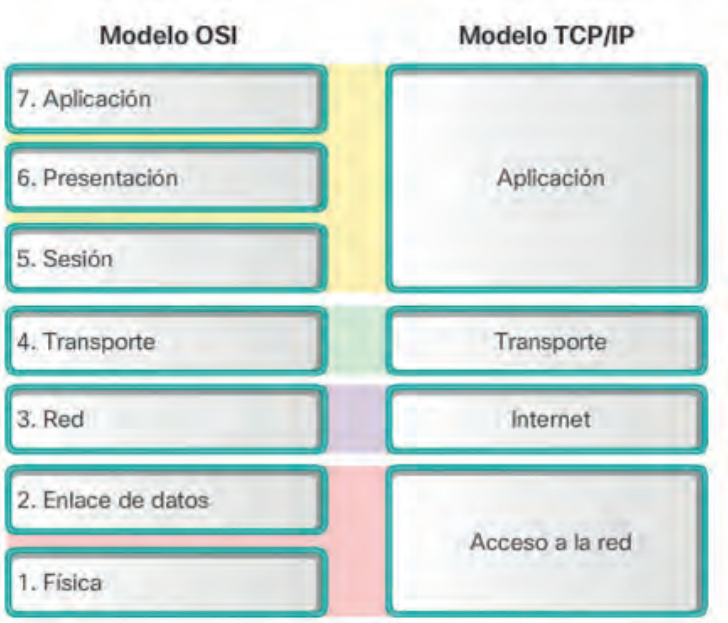


Figura 14.34: Comparación entre los modelos TCP/IP y OSI

La siguiente tabla muestra las capas de protocolo TCP/IP y los equivalentes del modelo OSI. También se muestran ejemplos de los protocolos disponibles en cada nivel de la pila del protocolo TCP/IP. Cada sistema que participa en una transacción de comunicación ejecuta una única implementación de la pila del protocolo.

REF. OSI – NRO DE CAPA	EQUIVALENTE DE CAPA OSI	CAPA TCP/IP	EJEMPLOS D PROTOCOLOS TCP/IP
5,6,7	Aplicación, sesión y presentación	Aplicación	NFS, NIS, DNS, DAP, telnet, ftp, rbgin, rsh, rcp, rip, rdisc, SNMP y otros.
4	Transporte	Transporte	TCP, UDP, SCTP
3	Red	Internet	IPv4, IPv5, ARP y ICMP
2	Vínculo de datos	Vínculo de datos	PPP, IEEE 802.2
1	Física	Física	Ethernet (IEEE 802.3), Token Ring, RS-232, FDDI y otros.

Figura 14.35: Capa modelo TCP/IP y su equivalente OSI

El modelo TCP/IP fue creado por el Departamento de Defensa de Estados Unidos con el objetivo de desarrollar una red redundante que pudiera sobrevivir a cualquier circunstancia, como por ejemplo, una guerra nuclear.

Algunos de los motivos principales que hicieron de TCP/IP el modelo de red más popular, y que su utilización sea de alcance global son los que se mencionan a continuación:

- Total independencia del fabricante. No importa el fabricante, si respeta las funciones de las capas y sus interfaces, la interconexión está garantizada.
- Soporta múltiples tecnologías. Puede utilizar cualquier tecnología física de transmisión.
- Funciona en computadoras de cualquier capacidad. No es necesario contar con una computadora potente para poder correr los protocolos TCP/IP.

Estos motivos son alcanzados debido a que TCP/IP se basa en un conjunto de metas: independencia de la tecnología utilizada en la conexión física y la arquitectura de *hardware* de la computadora y conectividad universal a través de la red.

Capa de red física

La capa de red física especifica las características del *hardware* que se utilizará para la red. Por ejemplo, las características físicas del medio de comunicaciones. La capa física de TCP/IP describe los estándares de *hardware* como IEEE 802.3, la especificación del medio de red Ethernet, y RS-232, la especificación para los conectores estándar.

El nivel físico describe las características físicas de la comunicación, como las convenciones sobre la naturaleza del medio usado para la comunicación (comunicaciones por cable, fibra óptica o radio), y todo lo relativo a los detalles como los conectores, código de canales y modulación, potencias de señal, longitudes de onda, sincronización y temporización y distancias máximas.

Capa de vínculo de datos

La capa de vínculo de datos identifica el tipo de protocolo de red del paquete, en este caso TCP/IP. La capa de vínculo de datos proporciona también control de errores y estructuras. El nivel de enlace de datos especifica cómo son transportados los paquetes sobre el nivel físico, incluyendo los delimitadores (patrones de bits concretos que marcan el comienzo y el fin de cada trama). Ethernet, por ejemplo, incluye campos en la cabecera de la trama que especifican qué máquina o máquinas de la red son las destinatarias de la trama. Ejemplos de protocolos de nivel de enlace de datos son Ethernet IEEE 802.2, Wireless Ethernet, SLIP, Token Ring, ATM, Protocolo punto a punto (PPP).

Capa de Internet

La capa de Internet, también conocida como capa de red o capa IP, acepta y transfiere paquetes para la red. Esta capa incluye el potente protocolo de internet (IP), el protocolo de resolución de direcciones (ARP) y el protocolo de mensajes de control de Internet (ICMP).

Protocolo IP

El protocolo IP y sus protocolos de enrutamiento asociados son posiblemente la parte más significativa del conjunto TCP/IP. El protocolo IP se encarga de lo siguiente:

- Direcciones IP: las convenciones de direcciones IP forman parte del protocolo IP. Como diseñar un esquema de direcciones IPv4 introduce las direcciones IPv4 y descripción general de las direcciones IPv6.

- Comunicaciones de host a host: el protocolo IP determina la ruta que debe utilizar un paquete, basándose en la dirección IP del sistema receptor.
- Formato de paquetes: el protocolo IP agrupa paquetes en unidades conocidas como datagramas. Puede ver una descripción completa de los datagramas en capa de Internet: preparación de los paquetes para la entrega.
- Fragmentación: si un paquete es demasiado grande para su transmisión a través del medio de red, el protocolo IP del sistema de envío lo divide en fragmentos de menor tamaño. A continuación, el protocolo IP del sistema receptor reconstruye los fragmentos y crea el paquete original.

Oracle Solaris admite los formatos de direcciones IPv4 e IPv6, que se describen en este manual. Para evitar confusiones con el uso del protocolo de Internet, se utiliza una de las convenciones siguientes:

- Cuando se utiliza el término IP en una descripción, esta se aplica tanto a IPv4 como a IPv6.
- Cuando se utiliza el término IPv4 en una descripción, esta solo se aplica a IPv4.
- Cuando se utiliza el término IPv6 en una descripción, esta solo se aplica a IPv6.

Protocolo ARP: el protocolo de resolución de direcciones (ARP) se encuentra conceptualmente entre el vínculo de datos y las capas de Internet. ARP ayuda al protocolo IP a dirigir los datagramas al sistema receptor adecuado asignando direcciones Ethernet (de 48 bits de longitud) a direcciones IP conocidas (de 32 bits de longitud).

Protocolo ICMP: el protocolo de mensajes de control de Internet (ICMP) detecta y registra las condiciones de error de la red como las que se detallan a continuación:

- Paquetes soltados: paquetes que llegan demasiado rápido para poder procesarse.
- Fallo de conectividad: no se puede alcanzar un sistema de destino.
- Redirección: redirige un sistema de envío para utilizar otro enrutador.

Capa de transporte

Los protocolos del nivel de transporte pueden solucionar problemas como la fiabilidad ("¿alcanzan los datos su destino?") y la seguridad de que los datos llegan en el orden correcto. En el conjunto de protocolos TCP/IP, los de transporte también determinan a qué aplicación van destinados los datos.

La capa de transporte TCP/IP garantiza que los paquetes lleguen en secuencia y sin errores, al intercambiar la confirmación de la recepción de los datos y retransmitir los paquetes perdidos. Este tipo de comunicación se conoce como transmisión de punto a punto. Los protocolos de capa de transporte de este nivel son el protocolo de control de transmisión (TCP), el protocolo de datagramas de usuario (UDP) y el protocolo de transmisión para el control de flujo (SCTP). Los protocolos TCP y SCTP proporcionan un servicio completo y fiable. UDP proporciona un servicio de datagrama poco fiable.

Protocolo TCP: permite a las aplicaciones comunicarse entre sí como si estuvieran conectadas físicamente. Envía los datos en un formato que se transmite carácter por carácter, en lugar de transmitirse por paquetes discretos. Esta transmisión consiste en lo siguiente:

- Punto de partida, que abre la conexión.
- Transmisión completa en orden de bytes.
- Punto de fin, que cierra la conexión.

TCP conecta un encabezado a los datos transmitidos, este contiene múltiples parámetros que ayudan a los procesos del sistema transmisor a conectarse a sus procesos correspondientes en el sistema receptor. TCP confirma que un paquete ha alcanzado su destino estableciendo una conexión de punto a punto entre los hosts de envío y recepción. Por tanto, el protocolo TCP se considera fiable orientado a la conexión.

Protocolo SCTP: es un protocolo de capa de transporte fiable orientado a la conexión que ofrece los mismos servicios a las aplicaciones que TCP. Además, admite conexiones entre sistemas que tienen más de una dirección, o de host múltiple. La conexión SCTP entre el sistema transmisor y receptor se denomina asociación. Los datos de la asociación se organizan en bloques. Dado que este protocolo admite varios hosts, determinadas aplicaciones, en especial las

que se utilizan en el sector de las telecomunicaciones, necesitan ejecutar SCTP en lugar de TCP.

Protocolo UDP: proporciona un servicio de entrega de datagramas. No verifica las conexiones entre los hosts transmisores y receptores. Dado que el protocolo UDP elimina los procesos de establecimiento y verificación de las conexiones, resulta ideal para las aplicaciones que envían pequeñas cantidades de datos.

Capa de aplicación

La capa de aplicación define las aplicaciones de red y los servicios de Internet estándar que puede utilizar un usuario. El nivel de aplicación es el que los programas más comunes utilizan para comunicarse a través de una red con otros programas. Los procesos que acontecen en este nivel son aplicaciones específicas que pasan los datos al nivel de aplicación en el formato que internamente use el programa y es codificado de acuerdo con un protocolo estándar.

Estos servicios utilizan la capa de transporte para enviar y recibir datos. Existen varios protocolos de capa de aplicación. En la lista siguiente se incluyen ejemplos de protocolos de capa de aplicación:

- Servicios TCP/IP estándar como los comandos ftp, tftp y telnet.
- Comandos UNIX "r", como rlogin o rsh.
- Servicios de nombres, como NIS o el sistema de nombre de dominio (DNS).
- Servicios de directorio (LDAP).
- Servicios de archivos, como el servicio NFS.
- Protocolo simple de administración de red (SNMP), que permite administrar la red.
- Protocolo RDISC (*router discovery server*) y protocolos RIP (*routing information protocol*).

Comparación de las capas de los modelos

Como se ha expresado, tanto en el modelo de referencia OSI como en el modelo TCP/IP, la funcionalidad de las capas es muy similar. Las capas por encima de transporte (incluyendo a esta) en ambos modelos tienen como objetivo prestar un servicio de transporte de extremo a extremo,

independientemente de las características de la red o las redes a través de las cuales se envíen los datos, a todos los procesos que deseen comunicarse.

Por último, TCP/IP implementa algunas de las funciones descriptas en las capas de enlace de datos y física del modelo OSI en una única capa denominada acceso a red. Hay que tener en cuenta también, que el modelo OSI es posterior al modelo TCP/IP. Mientras que el modelo OSI fue desarrollado definiendo capas y sus funciones para, posteriormente, implementar los protocolos, en TCP/IP se escribieron los protocolos primero y el modelo solo fue una descripción de los protocolos existentes.

Similitudes

- Los dos modelos se dividen en capas.
- Los dos modelos tienen capas de aplicación, aunque incluyen servicios muy distintos.
- Los dos modelos tienen capas de transporte y de red similares.
- Es necesario conocer ambos, ya que uno es de referencia para toda la documentación y el otro es el que está implementado.

Diferencias

- TCP/IP combina las funciones de la capa de aplicación, presentación y de sesión de OSI en una única capa de aplicación.
- TCP/IP combina las capas de enlace de datos y la capa física del modelo OSI en una sola capa de acceso a red.
- TCP/IP parece ser más simple porque tiene menos capas.
- Los protocolos TCP/IP son los estándares en torno a los cuales se desarrolló Internet, por lo que el modelo se encuentra muy popularizado.

ANEXOS

PROCESADOR IMAGINARIO IC2

El IC2 es un procesador imaginario utilizado en la cátedra para que el alumno interprete el funcionamiento de un procesador, identificando cada registro y cómo son utilizados cuando se ejecuta un programa.

¿Por qué IC2?

IC2 deriva del nombre Introducción a la Computación, tal como se denominaba a la cátedra en el anterior plan de estudios; el 2 se debe a su versión.

Organización

Una microcomputadora completa consta de varios dispositivos, además del microprocesador. Veremos rápidamente la estructura del microordenador, a continuación examinaremos con más detalle la organización interna del IC2. Este se diseñó para sustituir al IC1 con el que antiguamente se trabajaba en la cátedra.

Esquema

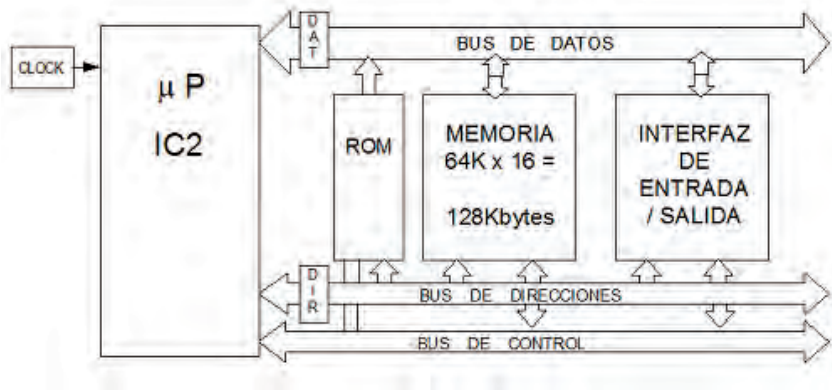


Figura A.1: Esquema del IC2

El IC2 consta de lo siguiente:

Unidad de control (UC)
Unidad aritmético-lógica (ALU)
Registros.
Tres buses:

1. Bus de datos bidireccional de 16 bits.
2. Bus de direcciones unidireccional de 16 bits.
3. Bus de control.

- A la salida de los buses se encuentran los registros DIR y DAT, los cuales cumplen la función de interfaz entre la CPU y la memoria.

- El µP necesita una referencia de tiempo exacta, la cual es proporcionada por un reloj o "clock".

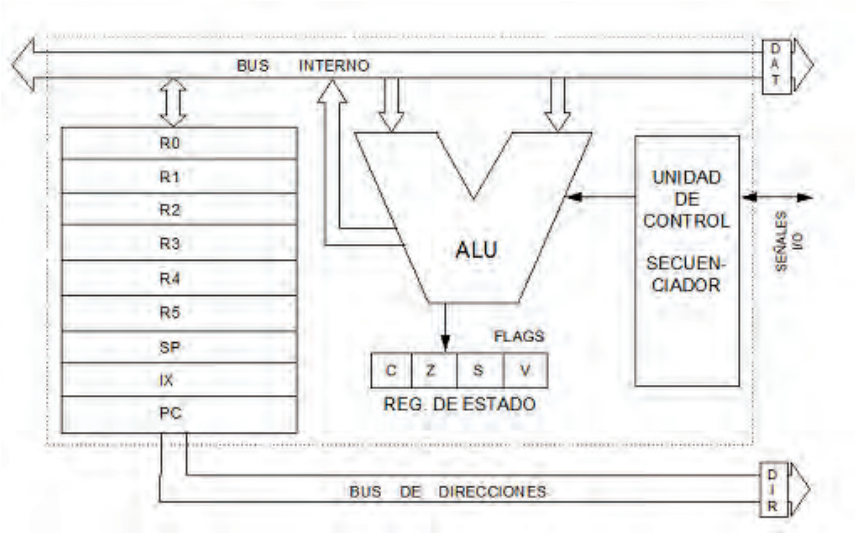
- A continuación, encontramos una memoria ROM (*read only memory*) que contiene el programa del sistema. La ventaja de la ROM es su contenido permanente. Por ello se utiliza para almacenar un "Programa de Control" encargado de garantizar la marcha del sistema.

- La memoria RAM de esta computadora tiene una capacidad de 128 KB, es la memoria de lectura/escritura del sistema.

- La computadora necesita de uno o más circuitos integrados de conexión con el mundo exterior. Se representa en el dibujo como I/O.

- No se describen aquí circuitos auxiliares como amplificadores, separadores, decodificadores, etcétera.

Arquitectura interna del IC2



Arquitectura interna del IC2

Empezaremos a analizar los módulos del microprocesador según se observan en la figura anterior.

- *Unidad de control*, que aparece a la derecha de la figura, es la encargada de sincronizar toda la actividad del sistema. En la sección de control se encuentra el registro de instrucción (IR) que contiene la instrucción que ha de ejecutarse. Bajo este se encuentra el decodificador de instrucciones y el secuenciador-controlador. Cuando desde la memoria se recibe una instrucción por el bus de datos, su código binario pasa al decodificador. Este último controla una memoria de microinstrucciones que proporciona los códigos necesarios al secuenciador, para que por sus líneas de E/S, transfiera las señales de gobierno necesarias para ejecutar la instrucción.

- *La unidad aritmético-lógica* realiza operaciones aritméticas y lógicas. Una de sus entradas está asociada a un registro acumulador (A), el cual en una misma instrucción puede referenciarse como E y S (fuente y destino). La ALU también se encarga de las operaciones de desplazamiento y rotación de bits.

Desplazamiento es la traslación del contenido de un byte una o más posiciones a la izquierda o a la derecha. En la figura, cada uno de los bits avanza una posición a la izquierda. Debajo de la ALU se observan las banderas o registro de estado (F). Su función es "llevar la cuenta" de las situaciones excepcionales que se producen en el interior del microprocesador.

- *Registros:* a la izquierda de la figura, se observan los registros del microprocesador, divididos conceptualmente en dos categorías: registros de tipo general y registros de direcciones.
  - *Registros de tipo general:* los registros internos están habitualmente etiquetados desde cero hasta n y su función no está definida de antemano (por eso se les llama de tipo general). Pueden contener cualquier tipo de dato manipulado durante el programa.
  - *Registros de direcciones:* son registros de 16 bits destinados al almacenamiento de direcciones y están conectados al bus de direcciones. Para diferenciar las mitades inferior y superior de cada registro, suele llamarse a la primera L (lower, comprende bits de 0 a 7) y H (higher, comprende los bits 8 a 15) a la segunda.
  - *Contador de programa (PC):* contiene la dirección de la instrucción que ha de ejecutarse a continuación. Su contenido se entrega al bus de direcciones que lo transmite a la memoria.
  - *Puntero de pila (SP):* es un registro de 16 bits que sirve para identificar el elemento superior de la pila en memoria. La pila es indispensable para programar interrupciones y acceder a las subrutinas.
  - *Registro de índice:* contiene un valor de desplazamiento que se suma automáticamente a una base (o viceversa, una base que se suma a un desplazamiento); de esta forma, el índice da acceso a cualquiera de las palabras de un bloque de datos.

Síntesis de registros de la CPU

- R_0: acumulador.
- R_1 a R_5: registros de uso general.
- XR: registro índice.
- PC: contador de programa.
- SP: puntero de pila.

Registros que se comunican con buses de direcciones y datos:

- DIR: registro de direcciones.
- DAT: registro de datos.

Registro asociado con la unidad de control:

- IR: registro de instrucciones.

Registro de estado:

- V = bit de desbordamiento:
  - V = 1 si el resultado produce desbordamiento.
  - V = 0 si no hay desbordamiento.
- S = bit de signo:
  - S = 1 si el signo del resultado es menos.
  - S = 0 si el signo del resultado es más.
- Z = bit de cero:
  - Z = 1 si el resultado de la operación es cero.
  - Z = 0 si el resultado de la operación es distinto de cero.
- C = bit de acarreo:
  - C = 1 si la operación produce acarreo del BMS del op.
  - C = 0 si la operación no produce acarreo del BMS.

Tipos de datos compatibles

1. Bit
2. Dígito BCD (4 bits)
3. Byte (8 bits)
4. Palabra (16 bits)

Formatos de las instrucciones

El formato de instrucción elegido para este microprocesador se ilustra a continuación. El código de instrucción consta de 16 bits divididos en cuatro campos generales:

- El primer campo, de dos bits, especifica el tipo de formato de la instrucción. Hay cuatro tipos de instrucciones.

- El segundo campo contiene 6 bits que especifican el código de operación de la instrucción. Con 6 bits pueden formularse 64 operaciones distintas independientes del tipo de instrucción.

- Los dos campos siguientes del formato de la instrucción dependen del tipo de la instrucción y especifican un registro del procesador o bien, el modo

de direccionamiento de una palabra de memoria, o bien como en los tipos 2 y 3, contienen 8 ceros.

Formato general

Tipo	Código de Oper.	Reg./Modo	Registro
2 bits	6 bits	4 bits	4 bits

Instrucción Tipo 0: registro - registro (de una palabra)

Este tipo de instrucciones tiene dos campos de registros además del código de operación. Se emplea en instrucciones de dos o tres operandos como el movimiento o la suma. El campo del registro fuente y el campo del registro destino constan cada uno de tres bits que especifican uno de los 8 registros procesadores, del R_0 alR_7. Cada registro puede referirse directa o indirectamente, según lo especifiquen los campos de bits indirecto SI y DI. Mientras que una referencia directa significa que el operando está en el registro, una indirecta significa que el registro contiene la dirección del operando en memoria.

0 0	Código de Oper.	SI	SRC	DI	DST
2 bits	6 bits	1 bit	3 bits	1 bit	3 bits

Donde:

- SRC: Registro fuente.
- SI: bit indirecto fuente: = 0 directo, =1 indirecto.
- DST: Registro destino.
- DI: bit indirecto destino: =0 directo, =1 indirecto.

Instrucción tipo 1: memoria - registro (de dos palabras)

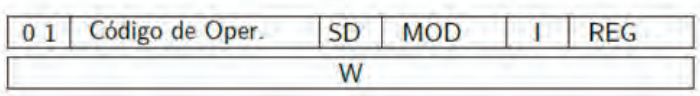
- El formato tipo 1 utiliza un registro para un operando y una palabra de memoria para el segundo operando; este se puede formular para instrucciones de uno o dos operandos. Las instrucciones de tipo 1 constan de dos

palabras, la segunda palabra está designada por W y se encuentra a continuación del código de instrucción. Los campos de registro indirecto son similares a los campos correspondientes en tipo 0. El campo MOD especifica cuatro modos de direccionamiento:

1. En el modo de dirección directa, la segunda palabra W contiene la dirección efectiva, es decir, la dirección del operando en memoria.
2. En el modo inmediato, el valor en W es el operando.
3. En el modo relativo, el valor en W se suma al contenido del registro PC para obtener la dirección efectiva.
4. En el modo indexado, el valor en W se suma al contenido del registro índice, XR para obtener la dirección efectiva.

El campo SD especifica la fuente, el destino y el número de operandos de la instrucción:

- Si el primer bit de SD = 0, especifica una instrucción de dos operandos; si es = 1, especifica una instrucción de un operando
- Si SD = 00, el operando fuente se evalúa a partir de la palabra de memoria W y el campo de modo MOD; si SD = 01, las asignaciones de operando fuente y destino se invierten.



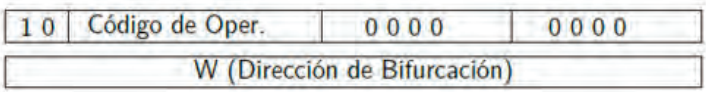
Donde:

- W palabra de memoria de dirección u operando inmediato.
- REG campo de registro.
- I Bit indirecto de registro: =0 directo, =1 indirecto.
- MOD modo de direccionamiento de la palabra de memoria W:
  - 00 -Modo de dirección directa.
  - 01 -Modo inmediato.
  - 10 -Modo relativo.
  - 11 -Modo indexado.
- SD Fuente/Destino y número de operandos:

- 00 -2 operandos. Fuente: palabra de memoria. Destino: registro.
- 01 -2 operandos. Fuente: registro. Destino: palabra de memoria.
- 10 -1 operando especificado por la palabra de memoria W y el campo MOD.
- 11 -1 operando especificado por los campos REG e I (W no se utiliza).

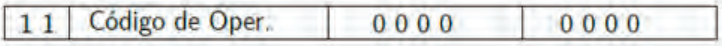
Instrucción tipo 2: Bifurcación (de dos palabras)

El formato tipo 2 corresponde exclusivamente a instrucciones de tipo bifurcación. Esta puede ser condicional o incondicional. La dirección de bifurcación W sigue al código de instrucción. Este formato se puede modificar para incluir una instrucción de bifurcación de tipo relativo. En este caso, los bits del 0 al 7 se emplean (intervalo +127 a -128) con el objeto de generar la dirección relativa que deberá sumarse a PC cuando se evalúe la dirección efectiva (en lugar de la palabra W)



Instrucción tipo 3: modo implícito (de una palabra)

El formato de tipo 3 especifica una instrucción de modo implícito como el regreso de interrupción u operación nula. Los bits 0 a 7 no se usan y son siempre igual a 0.



Clases de instrucciones

El conjunto de instrucciones de diversas computadoras difiere entre ellas, principalmente, en la forma en que se determinan los operandos a partir de los campos de dirección y de modo.

Las operaciones disponibles no varían mucho de una computadora a otra. La asignación de código binario en el campo de código de operación es

diferente en distintas máquinas. También varía el nombre simbólico de las instrucciones de una computadora a otra, aun siendo la misma instrucción.

Sin embargo, existe un conjunto de operaciones básicas incluidas en la mayoría de las computadoras que puede clasificarse de la siguiente forma:

1. Instrucciones de transferencia de datos.
2. Instrucciones de manipulación o tratamiento de datos.
3. Instrucciones de verificación y bifurcación.
4. Instrucciones de control.
5. Instrucciones de Entrada/Salida.

Instrucciones de transferencia de datos

Estas instrucciones llevan datos de un lugar a otro sin alterar su contenido en la computadora. Las transferencias más comunes se realizan entre la memoria y los registros del procesador, entre los registros del procesador y la Entrada y Salida y entre los registros internos.

A continuación, se presenta un listado de las instrucciones de transferencia de datos del IC2, con su correspondiente símbolo mnemónico, abreviatura de un lenguaje ensamblador estándar.

NOMBRE	MNEMÓNICO	FUNCIÓN
Carga	LD	Transferencia de la memoria a un registro.
Almacenamiento	ST	Transferencia de un registro a una palabra de memoria.
Movimiento	MOVE	Transferencia entre registros/ entre palabras de memoria.
Intercambio	EX	Canjea información e/registros o e/ reg. y memoria.
Introducción	PUSH	Transfiere datos entre la pila de memoria y un registro.
Descarga	POP	Transfiere datos entre la pila de memoria y un registro.
Entrada	IN	Transfiere datos entre dispositivos de entrada y registros.
Salida	OUT	Transfiere datos entre registros y dispositivos de salida.

Instrucciones de manipulación de datos

Estas instrucciones realizan operaciones con datos y son los medios de cómputo de la máquina. Suelen dividirse en los siguientes tipos:

1. Instrucciones aritméticas

NOMBRE	MNEMÓNICO	FUNCIÓN
Incremento	IN	Suma 1 al valor almacenado en un registro o memoria.
Decremento	DEC	Resta 1 al valor almacenado en un registro o memoria.
Suma	ADD	
Resta	SUB	
Multiplicación	MUL	
División	DIV	
Suma c/ acarreo	ADC	Adición de 2 operandos + el valor del acarreo de la operación anterior.
Resta c/Prest.	SBC	Resta 2 oper. y un préstamo que fue el result. de una operación anterior.
Negación	NEG	Realiza el complemento a 2 de un nro. c/signo, igual a multiplicar por -1.

2. Instrucciones lógicas y de manipulación de bits

Estas instrucciones realizan operaciones binarias con cadenas de bits almacenados en registros o en palabras de memoria y sirven para manipular bits individuales o bien un grupo de bits que representan información codificada en binario.

Las instrucciones lógicas consideran cada bit del operando por separado y lo tratan como una variable booleana.

Mediante la aplicación de estas operaciones, es posible cambiar valores de bits, anular un grupo de ellos o insertar nuevos valores de bits en operandos almacenados en registros o en palabras de memoria.

A continuación se exponen las instrucciones lógicas y de manipulación de bits disponibles en el IC2:

NOMBRE	MNEMÓNICO	FUNCIÓN
Poner a cero	CLEAR	El operando específico es sustituido por ceros
Iniciar	SET	El operando es reemplazado por unos
Complemento	COM	Invierte todos los bits del operando
AND	AND	Se utiliza para poner a 0 un bit seleccionado
OR	OR	Se utiliza para iniciación de un bit a 1
OR exclusivo	XOR	Se utiliza para complementar un bit
Poner a 0 acarr.	CLRC	
Iniciación acarr.	SETC	
Complemento de acarr.	COMC	

3. Instrucciones de corrimiento (desplazamiento y rotación)

Estas instrucciones se presentan en muchas variantes. Los corrimientos son operaciones en los cuales los bits del operando se mueven o desplazan a izquierda o derecha.

El desplazamiento es la traslación del contenido de un registro o de una posición de memoria a la izquierda o a la derecha en un bit; como resultado sale del registro un bit que pasa al bit de acarreo. El bit que entra por el lado opuesto será cero.

En la rotación, el bit que sale también pasa al acarreo; pero el que entra es el que estaba en dicho acarreo antes del inicio de la operación.

NOMBRE	MNEMÓNICO	FUNCIÓN
Lógico a der.	SHR	Los bits del operando se mueven hacia la der.
Lógico a izqu.	SHL	Los bits del operando se mueven a la izqu.
Aritmet. a der.	SHRA	
Aritmet. a izqu.	SHLA	
Rotación a der.	ROR	Corrimiento circular a derecha.
Rotación a izqu.	ROL	Corrimiento circular a izquierda.
Rota der. c/acarr.	RORC	
Rota izqu. c/acarr.	ROLC	

Instrucciones de control de programa (verificación y bifurcación)

Las instrucciones de verificación comprueban si los bits contenidos en el registro que se especifica son 0 o 1 o una combinación determinada de esos valores. Como mínimo es necesario que pueda verificarse el registro de estado.

Una instrucción de control del programa, cuando es ejecutada, puede cambiar el valor de la dirección del PC y hacer que se altere el flujo de control.

El cambio en el PC como resultado de la ejecución de una instrucción de control del programa ocasiona una interrupción en la secuencia de ejecución de las instrucciones.

Esta característica ofrece control sobre el flujo de ejecución del programa y un medio de bifurcación a diferentes segmentos dependiendo de operaciones de cálculo realizadas antes. La bifurcación suele ser una instrucción de una dirección. Cuando se ejecuta, la instrucción de bifurcación produce una transferencia de la dirección efectiva al PC.

Las instrucciones de bifurcación y salto pueden ser condicionales e incondicionales. La instrucción incondicional produce una bifurcación a la dirección efectiva especificada sin ninguna condición. La instrucción condicional

especifica una condición para la bifurcación como un resultado positivo o negativo.

La instrucción de salto no necesita un campo de dirección. Una instrucción de salto condicional saltará o pasará por alto la siguiente instrucción si se cumple la condición.

1. Instrucciones típicas de control de programa

NOMBRE	MNEMÓNICO	FUNCIÓN
Bifurcación	BR	Transferencia de la dirección efectiva al PC.
Salto	JMP	Pasa por alto la sig. instrucción si se cumple una condición.
Llamada subrutina	CALL	Se utiliza con prog. que utilizan subrutinas para llamarlas.
Regreso subrutina	RET	Se utiliza con prog. que utilizan subrutinas para retornar.
Comparación	CMP	Sustracción e/ dos operandos, sin almacenar resultado.
		Si actualizan los bits de estado.

2. Instrucciones de bifurcación condicional relativas a bits de estado

NOMBRE	MNEMÓNICO	FUNCIÓN
Bifurcar si es 0	BZ	Z=1
Bifurcar si no es 0	BNZ	Z=0
Bifurcar si hay acarreo	BC	C=1
Bifurcar si no hay acarreo	BNC	C=0
Bifurcar si es negativo	BM	S=1
Bifurcar si es positivo	BP	S=0
Bifurcar si hay desbordamiento	BV	V=1
Bifurcar si no hay desbordamiento	BNV	V=0

Modos de direccionamiento

El IC2 admite los siguientes modos de direccionamiento:

- Implícito.
- Inmediato.
- Registro e indirecto de registro.
- Directo.
- Indirecto.
- Relativo.
- Indexado.

Repertorio de instrucciones del IC2

El diseño completo del procesador incluye una lista de todas las instrucciones de la computadora, con los formatos a los cuales pertenecen, su código binario o código máquina correspondiente, modos de direccionamiento, etcétera.

A fin de demostrar la relación entre las instrucciones y sus formatos, se presentan algunas instrucciones típicas. A cada instrucción se le asigna un nombre simbólico, que usa el programador en el lenguaje ensamblador y un código binario de 6 bits que decodificará la Unidad de Control. La operación que realiza la instrucción determina el tipo de formato al cual pertenece. Cada instrucción tiene 16 bits que equivalen a 4 dígitos hexadecimales. El código de operación ocupa 6 bits, el cual junto con el código de tipo de 2 bits constituyen un código de 8 bits que se puede especificar con dos dígitos hexadecimales.

Algunas instrucciones y sus códigos

NOMBRE MNEMÓNICO	COD. OPER 6 BITS	TIPO 0	TIPO 1	TIPO 2	TIPO 3	OPERACIÓN
ADD	001001	09xx	49xx	-	-	adición
SUB	001010	0Axx	4Axx	-	-	Subtracción
MOVE	001011	0Bxx	4Bxx	-	-	Mov. de datos
INC	010011	-	53xx	-	-	Incremento
DEC	010100	-	54xx	-	-	Decremento
PUSH	011110	-	5Exx	-	-	Introd. en la pila.
BR	100001	-	-	A100	-	Bifurcación
BRC	100010	-	-	A200	-	Bifurcación en acarreo
CALL	100111	-	-	A700	-	Llamada de subrutina.
RET	110001	-	-	-	F100	Regreso de subrutina.

Por ejemplo, la instrucción ADD tiene un código de operación *001001* de 6 bits.

Si es de tipo 0, los primeros 2 bits son = 00, por lo tanto, su código de 8 bits será 00001001, es decir, un equivalente 09 hexadecimal.

La instrucción ADD de tipo 1 es el 01001001 binario, que es equivalente al 49 hexadecimal.

Las dos xx en los formatos de tipo 0 y 1 son valores de dígitos que se determinarán a partir de los últimos 8 bits de la instrucción que especifican el modo y los registros usados.

Veremos un ejemplo de la utilización de los formatos de tipo 0 y 1 analizando las combinaciones posibles que se pueden formular para la instrucción ADD. Se presenta, a continuación, una tabla con las cuatro instrucciones ADD con formato de tipo 0. Suponemos que R_5 = registro fuente y R_2 = registro destino.

Ejemplo de instrucción Tipo 0

CÓDIGO HEXA	SI	DI	CÓDIGO DE OPERACIÓN	OPERANDOS	OPERACIÓN
0952	0	0	ADD	R_5, R_2	$R_2 \quad R_2 + R_5$
095A	0	1	ADD	$R_5, (R_2)$	$M[R_2]M[R_2]+R_5$
09D2	1	0	ADD	$(R_5), R_2$	$R_2 \quad R_2 + M[R_5]$
09DA	1	1	ADD	$(R_5), (R_2)$	$M[R_2]M[R_2]+M[R_5]$

A continuación, se presenta una tabla de los modos de direccionamiento que se pueden formular para una instrucción de tipo 1. Suponemos que hay un campo REG 010 que corresponde al registro R_2. El tercer dígito hexadecimal del código de instrucción designa el valor en los campos SD y MOD. Si SD = 00, el operando destino se halla en R_2 y el campo MOD puede especificar hasta cuatro maneras de obtener el operando fuente de la segunda palabra W. Cuando SD = 01, el operando fuente está en el registroR_2 y el campo MOD especifica tres formas de obtener el operando destino.

Ejemplo de instrucción de tipo 1 con dos operandos

CÓD. HEXA	SI	MOD	MODO	CÓD. DE OPER.	OPERANDOS	OPERACIÓN
0902	00	00	Directo	ADD	W, R_2	$R_2 \leftarrow R_2 + M[W]$
0912	00	01	Inmediaro	ADD	$\#W , R_2$	$R_2 \leftarrow R_2 + W$
0922	00	10	Relativo	ADD	$\$W, R_2$	$R_2 \leftarrow R_2 + M[PC + W]$
0932	00	11	Índice	ADD	$W(XR) , R_2$	$R_2 \leftarrow R_2 + M[XR + W]$
0942	01	00	Directo	ADD	R_2 , W	$M[W] \leftarrow M[W]+R_2$
0962	01	10	Relativo	ADD	$R_2, \$W$	$M[PC + W] \leftarrow M[PC + W]+R_2$
0972	01	11	Índice	ADD	$R_2 , W(XR)$	$M[XR + W] \leftarrow M[XR + W]+R_2$

En la siguiente tabla se presentan los modos de direccionamiento que se pueden formular para la instrucción INC de tipo 1. Los primeros dos dígitos del código de operación son 53 de acuerdo a la tabla de códigos de instrucciones. El último dígito se hace igual a cero cuando el campo de registro no se utiliza, es igual a 2 cuando el registro (R_2) se utiliza directamente y es igual a A (hexa) cuando el registro(R_2) se utiliza de manera indirecta. Esta tabla presenta cinco maneras diferentes en que podemos formular una instrucción de tipo 1 de un operando.

Ejemplo de instrucción de tipo 1 con un operando.

CÓD. HEXA	SI	MOD	MODO	CÓD. DE OPER.	OPERANDOS	OPERACIÓN
5380	10	00	Directo	INC	W	$M[W] \leftarrow M[W]+1$
53A0	10	10	Relativo	INC	$\$W$	$M[PC + W] \leftarrow M[PC + W]+1$
53B0	10	11	Índice	INC	$W(XR)$	$M[XR + W] \leftarrow M[XR + W]+1$
53C2	11	00	Registro	INC	R_2	$R_2 \leftarrow R_2 + 1$
53CA	11	00	Indirect. reg.	INC	(R_2)	$M[R_2] \leftarrow M[R_2]+1$

Las instrucciones de tipo 2 y 3 tienen solo una combinación posible:
BR W - Bifurcación a W
CALL W - Llamada a subrutina en la dirección W
RET - Regreso de subrutina.

Las instrucciones de tipo 2 requieren un campo de dirección dado por la palabra de memoria W. Las de tipo 3 son de una palabra.

LISTADO DE INSTRUCCIONES DEL IC2

Para considerar: fff significa Registro Fuente, ddd significa Registro Destino y rrr Cualquier Registro.

Instrucciones de transferencia de datos

Transferencia de la memoria a un registro. Cod. Oper. 000001.

TIPO 0:

Transferencia en forma indirecta por registro, de la memoria al registro indicado.

LD (Rn), Rm

Operación: $Rm \leftarrow M[Rn]$

Codificación: |00|000001|1|fff|0|*ddd*|

TIPO 1:

Transferencia de la memoria a un registro. Fuente: W (con todos los modos de direccionamiento). Destino: cualquier registro.

LD W, Rn

Operación: $Rn \leftarrow M[W]$

Codificación: |01|000001|00|00|0|*ddd*|

|.....*W*.....|

Función: transfiere en forma directa el contenido de la palabra de memoria direccionada por W al registro Rn.

LD #W, Rn

Operación: $Rn \leftarrow W$

Codificación: |01|000001|00|01|0|*ddd*|

|.....*W*.....|

Función: transfiere en forma inmediata el operando en W al registro Rn.

LD $W, Rn

Operación: $Rn \leftarrow M[PC + W]$

Codificación: |01|000001|00|10|0|*ddd*|

|.....*W*.....|

Función: transfiere el contenido de la memoria en la dirección [PC+W] al registro Rn.

LD W(XR), Rn

Operación: $Rn \leftarrow M[XR + W]$

Codificación: |01|000001|00|11|0|*ddd*|

|.....*W*.....|

Función: transfiere en el contenido de la palabra de memoria indicada por el reg. XR más el desplazamiento W al registro Rn.

Transferencia de un registro a memoria. Cód. Oper. 000010.

TIPO 0:

Transferencia de un registro a memoria direccionada en forma indirecta a través de otro registro.

ST Rn, (Rm)

Operación: $M[Rm] \leftarrow Rn$

Codificación: |00|000010|0|fff|1|*ddd*|

TIPO 1:

Transferencia de un registro a la memoria. Fuente: cualquier registro (fff). Destino: W (con todos los modos de direccionamiento).

ST Rn,W

Operación: $M[W] \leftarrow Rn$

Codificación: |01|000010|01|00|0|fff|

|.....*W*.....|

Función: transfiere el contenido del reg. Rn a la dirección de memoria indicada por W.

ST Rn, $W

Operación: $M[PC + W] \leftarrow Rn$

Codificación: |01|000010|01|10|0|fff|

|.....*W*.....|

Función: transfiere el contenido del registro Rn a la palabra de memoria direccionada por [PC+W].

ST Rn, W(XR)

Operación: $M[XR + W] \leftarrow Rn$

Codificación: |01|000010|01|11|0|fff|

.....*W*.....|

Función: transfiere el contenido del registro Rn a la palabra de memoria direccionada por el reg. XR más el desplazamiento indicado en W.

Transferencia del contenido de un registro a otro o de una palabra de memoria a otra. Código de operación: 001011.

TIPO 0:

Transferencia del contenido de un registro a otro, o entre palabras de memoria direccionadas en forma indirecta a través de registros.

MOVE Rn, Rm

Operación: $Rm \leftarrow Rn$

Codificación: |00|001011|0|fff|0|*ddd*|

MOVE (Rn), (Rm)

Operación: $M[Rm] \leftarrow M[Rn]$

Codificación: |00|001011|1|fff|1|*ddd*|

TIPO 1:

Transferencia del contenido de una palabra de memoria a otra, usando como fuente y destino de la transferencia la dirección indicada por W (con todos los modos de direccionamiento) y una indirección por registro, o viceversa.

MOVE W, (Rn)

Operación: $M[Rn] \leftarrow M[W]$

Codificación: |01|001011|00|00|1|*ddd*|

|.....*W*.....|

Función: transfiere el contenido de la palabra de memoria direccionada por W a la palabra de memoria direccionada por el contenido del registro Rn.

MOVE (Rn), W

Operación: $M[W] \leftarrow M[Rn]$

Codificación: |01|001011|01|00|1|fff|

|.....*W*.....|

Función: transfiere el contenido de la palabra de memoria direccionada por el contenido del registro Rn a la palabra de memoria direccionada por W.

MOVE $W, (Rn)

Operación $M[Rn] \leftarrow M[PC + W]$

Codificación: |01|001011|00|10|1|*ddd*|

|.....*W*.....|

Función: transfiere el contenido de la palabra de memoria direccionada por [PC+W] a la palabra de memoria direccionada por el contenido del registro Rn.

MOVE (Rn), $W

Operación: $M[PC + W] \leftarrow M[Rn]$

Codificación: |01|001011|01|10|1|fff|

|.....*W*.....|

Función: transfiere el contenido de la palabra de memoria direccionada por el contenido el registro Rn a la palabra de memoria direccionada por [PC+W].

MOVE W(XR), (Rn)

Operación: $M[Rn] \leftarrow M[XR + W]$

Codificación: |01|001011|00|11|1|*ddd*|

|.....*W*.....|

Función: transfiere el contenido de la palabra de memoria direccionada por [XR+W], a la palabra de memoria direccionada por el contenido del registro Rn.

MOVE (Rn), W(XR)

Operación: $M[XR + W] \leftarrow M[Rn]$

Codificación: |01|001011|01|11|1|fff|

|.....*W*.....|

Función: transfiere el contenido de la palabra de memoria direccionada por el contenido del registro Rn a la palabra de memoria direccionada por [XR+W].

Intercambia los contenidos de registros con registros, registros con palabras de memoria o de palabras de memoria entre sí. Código de operación: 001100.

TIPO 0:
Permite el intercambio entre registros o de palabras de memoria direccionadas indirectamente por registro.

EX Rn, Rm
Operación: $Rm \Leftrightarrow Rn$
Codificación: |00|001100|0|fff|0|*ddd*|
Función: intercambia los contenidos de los registros Rm y Rn.

EX (Rn), Rm
Operación: $Rm \Leftrightarrow M[Rn]$
Codificación: |00|001100|1|fff|0|*ddd*|
Función: intercambia los contenidos del registro Rm y de la palabra de memoria direccionada indirectamente por Rn.

EX (Rn), (Rm)
Operación: $M[Rm] \Leftrightarrow M[Rn]$
Codificación: |00|001100|1|fff|1|*ddd*|
Función: intercambia los contenidos de las palabras de memoria direccionadas indirectamente por Rn y Rm.

TIPO 1:
Permite el intercambio de los contenidos de un registro y la palabra de memoria direccionada por W con sus distintos modos de direccionamiento.

EX W, Rn
Operación: $M[W] \Leftrightarrow Rn$
Codificación: |01|001100|00|00|0|*rrr*|
|.....*W*.....|
Función: intercambia los contenidos del registro Rn y de la palabra de memoria direccionada por W.

EX $W, Rn

Operación: $M[PC + W] \Leftrightarrow Rn$

Codificación: |01|001100|00|10|0|*rrr*|

|.....*W*.....|

Función: intercambia los contenidos del registro Rn y de la palabra de memoria direccionada por [PC+W].

EX W(XR), Rn

Operación: $M[XR + W] \Leftrightarrow Rn$

Codificación: |01|001100|00|11|0|*rrr*|

|.....*W*.....|

Función: intercambia los contenidos del registro Rn y de la palabra de memoria direccionada por [XR+W].

Introduce el contenido del registro Rn en la posición de memoria direccionada por el puntero de pila. Código de operación: 011110.

TIPO 1:

PUSH Rn

Operación: $M[SP - 1] \leftarrow Rn$; $SP \leftarrow SP - 1$

Codificación: |01|011110|11|00|0|fff|

Extrae el contenido de la posición de memoria direccionada por el puntero de pila y lo coloca en el registro Rn. Código de operación: 011111.

TIPO 1:

POP Rn

Operación: $Rn \leftarrow M[SP]$; $SP \leftarrow SP + 1$

Codificación: |01|011111|11|00|0|*ddd*|

Carga el registro indicado a partir del puerto de entrada direccionado por W. Código de operación: 001101.

TIPO 1:

IN W, Rn

Operación: $Rn \leftarrow M[W]$

Codificación: |01|001101|00|00|0|*ddd*|

|.....*W*.....|

Pone en la puerta de salida direccionada por W el contenido del registro Rn. Código de operación: 001110.

TIPO 1:
OUT Rn, W
Operación: $M[W] \leftarrow Rn$
Codificación: |01|001110|01|00|0|fff|
|.....W.....|
Suma 1 al valor del operando indicado. Cod. de Operación: 010011.

TIPO 1:
INC W
Operación: $M[W] \leftarrow M[W]+1$
Codificación: |01|010011|10|00|0|000|
|.....W.....|

INC $W
Operación: $M[PC + W] \leftarrow M[PC + W]+1$
Codificación: |01|010011|10|10|0|000|
|.....W.....|

INC W(XR)
Operación: $M[XR + W] \leftarrow M[XR + W]+1$
Codificación: |01|010011|10|11|0|000|
|.....W.....|

INC Rn
Operación: $Rn \leftarrow Rn + 1$
Codificación: |01|010011|11|00|0|*ddd*|

INC (Rn)
Operación: $M[Rn] \leftarrow M[Rn]+1$
Codificación: |01|010011|11|00|1|*ddd*|

Resta 1 al valor del operando indicado. Cod. de Operación: 010100.

TIPO 1:

DEC W

Operación: $M[W] \leftarrow M[W]-1$

Codificación: |01|010100|10|00|0|000|

|.....*W*.....|

DEC $W

Operación: $M[PC + W] \leftarrow M[PC + W]-1$

Codificación: |01|010100|10|10|0|000|

|.....*W*.....|

DEC W(XR)

Operación: $M[XR + W] \leftarrow M[XR + W]-1$

Codificación: |01|010100|10|11|0|000|

|.....*W*.....|

DEC Rn

Operación: $Rn \leftarrow Rn - 1$

Codificación: |01|010100|11|00|0|*ddd*|

DEC (Rn)

Operación: $M[Rn] \leftarrow M[Rn]-1$

Codificación: |01|010100|11|00|1|*ddd*|

Suma los dos operandos indicados, accediendo a ellos según los modos de direccionamiento posibles. El resultado queda almacenado en el segundo operando de la instrucción. Código de operación: 001001.

TIPO 0:

Se utilizan registros para almacenar ambos operandos, o almacenar la dirección de memoria para realizar un direccionamiento indirecto.

ADD Rn, Rm

Operación: $Rm \leftarrow Rn + Rm$

Codificación: |00|001001|0|fff|0|*ddd*|

ADD Rn, (Rm)

Operación: $M[Rm] \leftarrow Rn + M[Rm]$

Codificación: |00|001001|0|fff|1|*ddd*|

ADD (Rn), Rm

Operación: $Rm \leftarrow M[Rn] + Rm$

Codificación: |00|001001|1|fff|0|*ddd*|

ADD (Rn), (Rm)

Operación: $M[Rm] \leftarrow M[Rn] + M[Rm]$

Codificación: |00|001001|1|fff|1|*ddd*|

TIPO 1:

Se utiliza el operando W con los modos de direccionamiento disponibles, y el operando Rn para designar el segundo operando (no se realiza indirección con este último). Cualquiera puede ser destino de la operación según el valor del campo SD.

ADD W, Rn

Operación: $Rn \leftarrow M[W] + Rn$

Codificación: |01|001001|00|00|0|*ddd*|

|.....*W*.....|

ADD #W, Rn

Operación: $Rn \leftarrow W + Rn$

Codificación: |01|001001|00|01|0|*ddd*|

|.....*W*.....|

ADD $W, Rn

Operación: $Rn \leftarrow M[PC + W] + Rn$

Codificación:|01|001001|00|10|0|*ddd*|

|.....*W*.....|

ADD W(XR), Rn

Operación: $Rn \leftarrow M[XR + W] + Rn$

Codificación: |01|001001|00|11|0|*ddd*|

|.....*W*.....|

ADD Rn, W

Operación: $M[W] \leftarrow Rn + M[W]$

Codificación: |01|001001|01|00|0|fff|

|.....*W*.....|

ADD Rn, $W

Operación: $M[PC + W] \leftarrow Rn + M[PC + W]$

Codificación: |01|001001|01|10|0|fff|

|.....*W*.....|

ADD Rn, W(XR)

Operación: $M[XR + W] \leftarrow Rn + M[XR + W]$

Codificación: |01|001001|01|11|0|fff|

|.....*W*.....|

Resta los dos operandos indicados, accediendo a ellos según los modos de direccionamiento posibles. El resultado queda almacenado en el segundo operando de la instrucción. Código de operación: 001010.

TIPO 0:

Se utilizan registros para almacenar ambos operandos, o almacenar la dirección de memoria para realizar un direccionamiento indirecto.

SUB Rn, Rm

Operación: $Rm \leftarrow Rn - Rm$

Codificación: |00|001010|0|fff|0|*ddd*|

SUB Rn, (Rm)

Operación: $M[Rm] \leftarrow Rn - M[Rm]$

Codificación: |00|001010|0|fff|1|*ddd*|

SUB (Rn), Rm

Operación: $Rm \leftarrow M[Rn] - Rm$

Codificación: |00|001010|1|fff|0|*ddd*|

SUB (Rn), (Rm)

Operación: $M[Rm] \leftarrow M[Rn] - M[Rm]$

Codificación: |00|001010|1|fff|1|*ddd*|

TIPO 1:

Se utiliza el operando W con los modos de direccionamiento disponibles y el operando Rn para designar el segundo operando (no se realiza indirección con este último). Cualquiera puede ser destino de la operación según el valor del campo SD.

SUB W, Rn

Operación: $Rn \leftarrow M[W] - Rn$

Codificación: |01|001010|00|00|0|*ddd*|

|.....*W*.....|

SUB #W, Rn

Operación: $Rn \leftarrow W - Rn$

Codificación: |01|001010|00|01|0|*ddd*|

|.....*W*.....|

SUB $W, Rn

Operación: $Rn \leftarrow M[PC + W] - Rn$

Codificación: |01|001010|00|10|0|*ddd*|

|.....*W*.....|

SUB W(XR), Rn

Operación: $Rn \leftarrow M[XR + W] - Rn$

Codificación: |01|001010|00|11|0|*ddd*|

|.....*W*.....|

SUB Rn, W

Operación: $M[W] \leftarrow Rn - M[W]$

Codificación: |01|001010|01|00|0|fff|

|.....*W*.....|

SUB Rn, $W

Operación: $M[PC + W] \leftarrow Rn - M[PC + W]$

Codificación: |01|001010|01|10|0|fff|

|.....W.....|

SUB Rn, W(XR)

Operación: $M[XR + W] \leftarrow Rn - M[XR + W]$

Codificación: |01|001001|01|11|0|fff|

|.....W.....|

Multiplica el operando indicado con el contenido del registro R_0, el resultado queda almacenado en el mismo registro con extensión al registro R_1 si fuera necesario.

Código de operación: 001111.

TIPO 1:

Al utilizarse solo un operando explícito se utiliza este tipo de instrucción únicamente. Si se utiliza un registro como operando, este no puede ser R_0 ni R_1.

MUL Rn

Operación: $R_0 \leftarrow R_0 x Rn$

Codificación: |01|001111|11|00|0|fff|

MUL (Rn)

Operación: $R_0 \leftarrow R_0 x M[Rn]$

Codificación: |01|001111|11|00|1|fff|

MUL W

Operación: $R_0 \leftarrow R_0 x M[W]$

Codificación: |01|001111|10|00|0|000|

|.....W.....|

MUL #W

Operación: $R_0 \leftarrow R_0 x W$

Codificación: |01|001111|10|01|0|000|

|.....W.....|

MUL $W

Operación: $R_0 \leftarrow R_0 xM[PC + W]$

Codificación: |01|001111|10|10|0|000|

|.....W.....|

MUL W(XR)

Operación: $R_0 \leftarrow R_0 xM[XR + W]$

Codificación: |01|001111|10|11|0|000|

|.....W.....|

Divide el contenido del registro R_0 por el operando indicado, el resultado queda almacenado en el mismo registro R_0.
Código de operación: 010000.

TIPO 1:
Al utilizarse solo un operando explícito se utiliza este tipo de instrucción únicamente. Si se utiliza un registro como operando, este no puede ser R_0.

DIV Rn

Operación: $R_0 \leftarrow R_0 : Rn$

Codificación: |01|010000|11|00|0|fff|

DIV (Rn)

Operación: $R_0 \leftarrow R_0 : M[Rn]$

Codificación: |01|010000|11|00|1|fff|

DIV W

Operación: $R_0 \leftarrow R_0 : M[W]$

Codificación: |01|010000|10|00|0|000|

|.....W.....|

DIV #W

Operación: $R_0 \leftarrow R_0 : W$

Codificación: |01|010000|10|01|0|000|

|.....W.....|

DIV $W

Operación: $R_0 \leftarrow R_0 : M[PC + W]$

Codificación: |01|010000|10|10|0|000|

|.....W.....|

DIV W(XR)

Operación: $R_0 \leftarrow R_0 : M[XR + W]$

Codificación: |01|010000|10|11|0|000|

|.....W.....|

Suma el operando indicado y el valor del flag de acarreo al contenido del registro R_0, el resultado queda almacenado en el registro R_0.
Código de operación: 010001.

TIPO 1:
Al utilizarse solo un operando explícito se utiliza este tipo de instrucción únicamente. Si se utiliza un registro como operando, este no puede ser R_0.

ADC Rn

Operación: $R_0 \leftarrow R_0 + Rn + C$

Codificación: |01|010001|11|00|0|fff|

ADC (Rn)

Operación: $R_0 \leftarrow R_0 + M[Rn]+C$

Codificación: |01|010001|11|00|1|fff|

ADC W

Operación: $R_0 \leftarrow R_0 + M[W]+C$

Codificación: |01|010001|10|00|0|000|

|.....W.....|

ADC #W

Operación: $R_0 \leftarrow R_0 + W + C$

Codificación: |01|010001|10|01|0|000|

|.....W.....|

ADC $W

Operación: $R_0 \leftarrow R_0 + M[PC + W] + C$

Codificación: |01|010001|10|10|0|000|

|.....W.....|

ADC W(XR)

Operación: $R_0 \leftarrow R_0 + M[XR + W] + C$

Codificación: |01|010001|10|11|0|000|

|.....W.....|

Resta el operando indicado y el valor del flag de acarreo del contenido del registro R_0, el resultado queda almacenado en el registro R_0.
Código de operación: 010010.

TIPO 1:
Al utilizarse solo un operando explícito se utiliza este tipo de instrucción únicamente. Si se utiliza un registro como operando, este no puede ser R_0.

SBC Rn

Operación: $R_0 \leftarrow R_0 - Rn - C$

Codificación: |01|010010|11|00|0|fff|

SBC (Rn)

Operación: $R_0 \leftarrow R_0 - M[Rn] - C$

Codificación: |01|010010|11|00|1|fff|

SBC W

operación: $R_0 \leftarrow R_0 - M[W] - C$

Codificación: |01|010010|10|00|0|000|

|.....W.....|

SBC #W

Operación: $R_0 \leftarrow R_0 - W - C$

Codificación: |01|010010|10|01|0|000|

|.....W.....|

SBC $W

Operación: $R_0 \leftarrow R_0 - M[PC + W] - C$

Codificación: |01|010010|10|10|0|000|

|.....W.....|

SBC W(XR)

Operación: $R_0 \leftarrow R_0 - M[XR + W] - C$

Codificación: |01|010010|10|11|0|000|

|.....W.....|

Realiza el complemento a 2 del operando indicado. Esto es equivalente a multiplicarlo por -1. Código de operación: 010101.

TIPO 1:

Al utilizarse un único operando explícito se utiliza este tipo de operación, siendo aplicable únicamente a los registros de datos del procesador.

NEG Rn

Operación: $Rn \leftarrow 0 - Rn$

Codificación: |01|010101|11|00|0|*rrr*|

Carga con ceros el operando especificado.
Código de operación: 010110.

TIPO 1:

CLR Rn

Operación: $Rn \leftarrow H0000$

Codificación: |01|010110|11|00|0|*ddd*|

CLR (Rn)

Operación: $M[Rn] \leftarrow H0000$

Codificación: |01|010110|11|00|1|*ddd*|

CLR W

Operación: *M*[*W*]←*H*0000
Codificación: |01|010110|10|00|0000|
|.....*W*.....|

CLR $W

Operación: *M*[*PC* + *W*]←*H*0000
Codificación: |01|010110|10|10|0000|
|.....*W*.....|

CLR W(XR)

Operación: *M*[*XR* + *W*]←*H*0000
Codificación: |01|010110|10|11|0000|
|.....*W*.....|

Carga con (1) unos los bits del operando especificado.
Código de operación: 010111.

TIPO 1:

SET Rn

Operación: *Rn* ← *HFFFF*
Codificación: |01|010111|11|00|0|*ddd*|

SET (Rn)

Operación: *M*[*Rn*]←*HFFFF*
Codificación: |01|010111|11|00|1|*ddd*|

SET W

Operación: *M*[*W*]←*HFFFF*
Codificación: |01|010111|10|00|0000|
|.....*W*.....|

SET $W

Operación: $M[PC + W] \leftarrow HFFFF$

Codificación: |01|010111|10|10|0000|

|.....*W*.....|

SET W(XR)

Operación: $M[XR + W] \leftarrow HFFFF$

Codificación: |01|010111|10|11|0000|

|.....*W*.....|

Invierte todos los bits del operando especificado.

Código de operación: 011000.

TIPO 1:

COM Rn

Operación: $Rn \leftarrow \overline{Rn}$

Codificación: |01|011000|11|00|0|*ddd*|

COM (Rn)

Operación: $M[Rn] \leftarrow \overline{M[Rn]}$

Codificación: |01|011000|11|00|1|*ddd*|

COM W

Operación: $M[W] \leftarrow \overline{M[W]}$

Codificación: |01|011000|10|00|0000|

|.....*W*.....|

COM $W

Operación: $M[PC+W] \leftarrow \overline{M[PC+W]}$

Codificación: |01|011000|10|10|0000|

|.....*W*.....|

COM W(XR)

Operación: $M[XR+W] \leftarrow \overline{M[XR+W]}$

Codificación: |01|011000|10|11|0000|

|.....*W*.....|

Realiza la operación Y lógica entre el acumulador (R_0) y el operando especificado; el resultado se almacena en R_0. Código de operación: 011001.

TIPO 1:

AND Rn

Operación: $R_0 \leftarrow R_0 \wedge Rn$

Codificación: |01|011001|11|00|0|*ddd*|

AND (Rn)

Operación: $R_0 \leftarrow R_0 \wedge M[Rn]$

Codificación: |01|011001|11|00|1|*ddd*|

AND W

Operación: $R_0 \leftarrow R_0 \wedge M[W]$

Codificación: |01|011001|10|00|0000|
|.....*W*.....|

AND $W

Operación: $R_0 \leftarrow R_0 \wedge M[PC + W]$

Codificación: |01|011001|10|10|0000|
|.....*W*.....|

AND W(XR)

Operación: $R_0 \leftarrow R_0 \wedge M[XR + W]$

Codificación: |01|011001|10|11|0000|
|.....*W*.....|

Realiza la operación O lógica entre el acumulador (R_0) y el operando especificado, el resultado se almacena en R_0.
Código de operación: 011010.
TIPO 1:

OR Rn

Operación: $R_0 \leftarrow R_0 \vee Rn$

Codificación: |01|011010|11|00|0|*ddd*|

OR (Rn)

Operación: $R_0 \leftarrow R_0 \vee M[Rn]$

Codificación:|01|011010|11|00|1|*ddd*|

OR W

Operación: $R_0 \leftarrow R_0 \vee M[W]$

Codificación: |01|011010|10|00|0000|

|.....*W*.....|

OR $W

Operación: $R_0 \leftarrow R_0 \vee M[PC + W]$

Codificación: |01|011010|10|10|0000|

|.....*W*.....|

OR W(XR)

Operación: $R_0 \leftarrow R_0 \vee M[XR + W]$

Codificación: |01|011010|10|11|0000|

|.....*W*.....|

Realiza la operación O lógica exclusiva entre el acumulador (R_0) y el operando especificado, el resultado se almacena en R_0.
Código de operación: 011011.

TIPO 1:

XOR Rn

Operación: $R_0 \leftarrow R_0 \oplus Rn$

Codificación: |01|011011|11|00|0|*ddd*|

XOR (Rn)

Operación: $R_0 \leftarrow R_0 \oplus M[Rn]$

Codificación: |01|011011|11|00|1|*ddd*|

XOR W

Operación: $R_0 \leftarrow R_0 \oplus M[W]$

Codificación: |01|011011|10|00|0000|

|.....*W*.....|

XOR $W

Operación: $R_0 \leftarrow R_0 \oplus M[PC + W]$

Codificación: |01|011011|10|10|0000|

|.....W.....|

XOR W(XR)

Operación: $R_0 \leftarrow R_0 \oplus M[XR + W]$

Codificación: |01|011011|10|11|0000|

|.....W.....|

Pone a cero el bit de acarreo del Registro de Estado.
Código de operación: 000011.

TIPO 3:

CLRC

Operación: $C \leftarrow 0$

Codificación:|11|000011|00000000|

Pone en 1 el bit de acarreo del registro de Estado.
Código de operación: 000100.

TIPO 3:

SETC

Operación: $C \leftarrow 1$

Codificación:|11|000100|00000000|

Complementa el bit de acarreo del Registro de Estado.
Código de operación: 000101.

TIPO 3:

COMC

Operación: $C \leftarrow \overline{C}$

Codificación: |11|000101|00000000|

Desplazamiento lógico a derecha de los bits del operando ubicado en el registro o direccionado indirectamente por el registro.
Código de Operación: 101011.

TIPO 1:

SHR Rn

Operación: 0 → |15 → 0|→*C*

Codificación: |01|101011|11|00|0|*rrr*|

SHR (Rn)

Operación: 0 → |15 → 0|→*C*

Codificación: |01|101011|11|00|1|*rrr*|

Desplazamiento aritmético a derecha de los bits del operando ubicado en el registro o direccionado indirectamente por el registro.
Código de Operación: 101100.

TIPO 1:

SHRA Rn

Operación: |15 → 0|→*C*

Codificación: |01|101100|11|00|0|*rrr*|

SHRA (Rn)

Operación: |15 → 0|→*C*

Codificación: |01|101100|11|00|1|*rrr*|

Desplazamiento aritmético a izquierda de los bits del operando ubicado en el registro o direccionado indirectamente por el registro.
Código de Operación: 101101.

TIPO 1:

SHLA Rn

Operación: *C* ← |15 ← 0|←0

Codificación: |01|101101|11|00|0|*rrr*|

SHLA (Rn)

Operación: *C* ← |15 ← 0|←0

Codificación: |01|101101|11|00|1|*rrr*|

Corrimiento circular a derecha de los bits del operando ubicado en el registro o direccionado indirectamente por el registro, más el bit de acarreo.
Código de Operación: 000101.

TIPO 1:

RORC Rn

Operación: ↓ → |15 → 0|→*C* → ↑

Codificación: |01|000101|11|00|0|*rrr*|

RORC (Rn)

Operación: ↓ → |15 → 0|→*C* → ↑

Codificación: |01|000101|11|00|1|*rrr*|

Corrimiento circular a izquierda de los bits del operando ubicado en el registro o direccionado indirectamente por el registro, más el bit de acarreo.
Código de Operación: 101101.

TIPO 1:

ROLC Rn

Operación: ↑ ← *C* ← |15 ← 0|←↓

Codificación: |01|101101|11|00|0|*rrr*|

ROLC (Rn)

Operación: ↑ ← *C* ← |15 ← 0|←↓

Codificación: |01|101101|11|00|1|*rrr*|

Transferencia de la dirección efectiva indicada al registro PC.
Código de Operación: 100001.

TIPO 2:

BR #W

Operación: *PC* ← *W*

Codificación: |10|100001|00000000|

|.....*W*.....|

El contenido del PC se empuja en la pila, a continuación la dirección indicada (comienzo de la subrutina) se transfiere al registro PC.
Código de Operación: 100111.

TIPO 2:

CALL #W

Operación: $M[SP - 1] \leftarrow PC; SP \leftarrow SP - 1; PC \leftarrow W$
Codificación: |10|100111|00000000|
|.....*W*.....|

El contenido del tope de la pila se transfiere al registro PC.
Código de Operación: 110001.

TIPO 3:

RET

Operación: $PC \leftarrow M[SP]; SP \leftarrow SP + 1$
Codificación: |11|110001|00000000|

Compara los operandos indicados, realizando la diferencia entre ellos sin almacenar el resultado, pero actualizando los flags apropiadamente.
Código de operación: 111000.

TIPO 1:
Se utiliza el operando W con los modos de direccionamiento disponibles y el operando Rn para designar el segundo operando (no se realiza indirección con este último). Cualquiera puede ser destino de la operación según el valor del campo SD.

CMP W, Rn

Operación: $M[W] - Rn$
Codificación: |01|111000|00|00|0|*ddd*|
|.....*W*.....|

CMP #W, Rn

Operación: $W - Rn$
Codificación: |01|111000|00|01|0|*ddd*|
|.....*W*.....|

CMP $W, Rn

Operación: $M[PC + W] - Rn$

Codificación: |01|111000|00|10|0|*ddd*|

|.....*W*.....|

CMP W(XR), Rn

Operación: $M[XR + W] - Rn$

Codificación: |01|111000|00|11|0|*ddd*|

|.....*W*.....|

CMP Rn, W

Operación: $Rn - M[W]$

Codificación: |01|111000|01|00|0|fff|

|.....*W*.....|

CMP Rn, $W

Operación: $Rn - M[PC + W]$

Codificación: |01|111000|01|10|0|fff|

|.....*W*.....|

CMP Rn, W(XR)

Operación: $Rn - M[XR + W]$

Codificación: |01|111000|01|11|0|fff|

|.....*W*.....|

Transfiere la dirección indicada si el valor del flag de cero es uno. Código de Operación: 100000.

TIPO 2:

BZ #W

Operación: $siZ = 1 \Rightarrow PC \leftarrow W$

Codificación: |10|100000|00000000|

|.....*W*.....|

Transfiere la dirección indicada si el valor del flag de cero es cero. Código de Operación: 100011.

TIPO 2:

BNZ #W

Operación: $siZ = 0 \Rightarrow PC \leftarrow W$

Codificación: |10|100011|00000000|

|.....W.....|

Transfiere la dirección indicada si el valor del flag de acarreo es uno.
Código de Operación: 100010.

TIPO 2:

BC #W

Operación: $siC = 1 \Rightarrow PC \leftarrow W$

Codificación: |10|100010|00000000|

|.....W.....|

Transfiere la dirección indicada si el valor del flag de acarreo es cero.
Código de Operación: 100100.

TIPO 2:

BNC #W

Operación: $siC = 0 \Rightarrow PC \leftarrow W$

Codificación: |10|100100|00000000|

|.....W.....|

Transfiere la dirección indicada si el valor del flag de signo es uno.
Código de Operación: 100101.

TIPO 2:

BN #W

Operación: $siS = 1 \Rightarrow PC \leftarrow W$

Codificación: |10|100101|00000000|

|.....W.....|

Transfiere la dirección indicada si el valor del flag de signo es cero.
Código de Operación: 100110.

TIPO 2:

BP #W

Operación: $siS = 0 \Rightarrow PC \leftarrow W$

Codificación: |10|100110|00000000|

|.....*W*.....|

Transfiere la dirección indicada si el valor del flag de desbordamiento es uno. Código de Operación: 101000.

TIPO 2:

BV #W

Operación: $siV = 1 \Rightarrow PC \leftarrow W$

Codificación: |10|101000|00000000|

|.....*W*.....|

Transfiere la dirección indicada si el valor del flag de desbordamiento es cero. Código de Operación: 101001.

TIPO 2:

BNV #W

Operación: $siV = 0 \Rightarrow PC \leftarrow W$

Codificación: |10|101001|00000000|

|.....*W*.....|

SIMULADOR DE PROCESADOR SIMULPROC

Con el paso del tiempo la cátedra dejó de utilizar el IC2 y se "actualizó" con el uso de un simulador.

Como ya se explicó en la guía teórica, la programación de computadoras debe hacerse en un lenguaje que sea comprensible para ellas (lenguaje de máquina) o en lenguajes que puedan, finalmente, ser traducidos al lenguaje que las computadoras pueden interpretar.

Para visualizar e investigar cómo es el proceso de ejecución y programación en el primer nivel de lenguaje que le sigue al lenguaje de máquina, utilizaremos el lenguaje ensamblador. En este camino nos vamos a valer de un programa de simulación de un procesador virtual llamado SimulProc.

El objetivo será recorrer, desde el principio, el camino para desarrollar un programa de computadora. En este proceso se verá cómo se pasa por cada nivel de abstracción hasta llegar a la perspectiva del *hardware* presentada por el simulador que vamos a estudiar.

El autor de SimulProc es Vladimir Yepes, quien autoriza su libre distribución. El simulador puede ser descargado de http://simulproc.tk

Descripción del simulador SimulProc

SimulProc es un simulador de un procesador hipotético con el cual se podrán aprender las nociones básicas para empezar a programar en lenguaje ensamblador, en el que se observará todo el proceso interno de ejecución del programa a través de cada ciclo del procesador.

Este simulador hipotético muestra cómo funciona un procesador internamente, ver qué realiza en cada ciclo. El ciclo de ejecución de un procesador se divide en dos semiciclos simples, que se mencionan a continuación:

1. El ciclo de fetch:
   - Va al PC.
   - Va a la dirección que apunta el PC.
   - Hace IR = MEM[PC].
   - Incrementa PC.

2. El ciclo de ejecución:
   - Si tiene que ir a memoria.
   - Va a memoria.
   - Ejecuta instrucción.
   - Almacena resultados.

Características del procesador

Memoria:

La memoria es el dispositivo que almacena toda la información del programa que se ejecuta, tanto datos como instrucciones. Esta, en realidad, no es parte del procesador, sino que es un dispositivo aparte al que el procesador accede para ir leyendo las instrucciones y datos del programa.

La capacidad de la memoria simulada es de 4096 posiciones de 16 bits cada registro: Desde 000 hasta FFF. El simulador trabaja con constantes y variables en binario y direcciones (posiciones de memoria) en hexadecimal.

Registros generales:

Los registros generales del procesador se usan para almacenar información de uso rápido, ya que se accede a ellos a una velocidad mucho más alta que la memoria. Allí se pueden almacenar direcciones de memoria a las que se va a acceder bastante a lo largo de la ejecución del programa, o directamente variables que se desean usar.

Este procesador consta de tres registros de propósito general, AX, BX y CX cada uno con 16 bits de capacidad.

Registros apuntadores como:

PC IP (*program counter* o *instruction pointer*): contiene la dirección de memoria de la próxima instrucción a ejecutar y es incrementado en cada nueva instrucción.

MAR (*memory address register*, registro de dirección de memoria en español): es el registro en el que se almacena la dirección de memoria a la que se quiere acceder.

MDR (*memory data register* o *memory buffer register*): es un registro intermedio en el que se almacenan los datos que se escriben o leen de memoria. En el caso de una lectura, se pone en el MAR la dirección y se activa la señal de leer, obteniendo en el MDR el dato buscado. En el caso de una escritura, se pone en el MAR la dirección y en el MDR el dato a escribir en memoria, después de que la señal de escribir esté activa, de esta forma almacenamos en memoria el dato.

IR (*instruction register*): en este registro se introduce la instrucción a ejecutar, después de haberla leído de la memoria accediendo a ella mediante la dirección señalada en el PC. El contenido de este registro se puede dividir en código de operación (el código que señala la operación que se realizará) y el/los operandos. Puede haber dos operandos o uno solo. Acá es donde se decodifica e interpreta la instrucción así: se descompone la instrucción leída de forma que se pueda saber cuál es la operación que se desea realizar y cuáles son los operandos, en su caso, o el desplazamiento en caso de que se trate de una instrucción de bifurcación...

Registros de pila:

BP (*base pointer*, puntero de base de la pila en español): por defecto el valor es F80, este puede cambiarse desde un programa, asignándole otra dirección de memoria con la instrucción MOV. Digamos que quiero reservar más espacio para la pila haciendo que esta comience desde la posición CF1, entonces, copio esta dirección de memoria en cualquier posición de memoria; supongamos que lo copie en la dirección 3B, entonces, uso la instrucción MOV BP, 3B y así BP es igual a CF1. Mientras se ejecuta el programa se puede visualizar en una barra de porcentaje el uso de la pila.

SP (*stack pointer*, puntero de la pila en español): indica en qué próxima dirección de la pila está disponible; es decir, apunta a la cima de la pila. Este valor se cambia automáticamente cuando se usan las instrucciones PUSH POP.

Registros de control (flags) o registros de estado:

Estos registros se usan para poder controlar el comportamiento de un programa, los cuales se activan después de cada operación, según sea el resultado de la instrucción ejecutada.

Zero flag: se vuelve 1 si el resultado de la última operación = 0.

Negative sign flag: se vuelve 1 si el resultado de la última operación es igual a un número negativo.

Carry flag: se activa cuando la operación realizada ha producido un acarreo.

Overflow flag: se activa cuando la operación produjo desbordamiento (overflow), es decir, el resultado ocupaba más de los 16 bits que caben en un registro.

Estos flags se usan principalmente en instrucciones de bifurcación (por ejemplo, si queremos que, en caso de que el resultado de la última operación fuera cero, el programa se salte varias de las instrucciones siguientes, comprobamos el flag cero, y si está activo el programa, salta; esto se consigue con la instrucción JEQ).

ALU (*arithmetic logic unit,* unidad aritmética y lógica en español): es donde el procesador realiza las operaciones matemáticas, (suma, resta, multiplicación...) y las operaciones lógicas (AND, OR, desplazamiento de bits...).

Descripción de la interfaz gráfica del programa

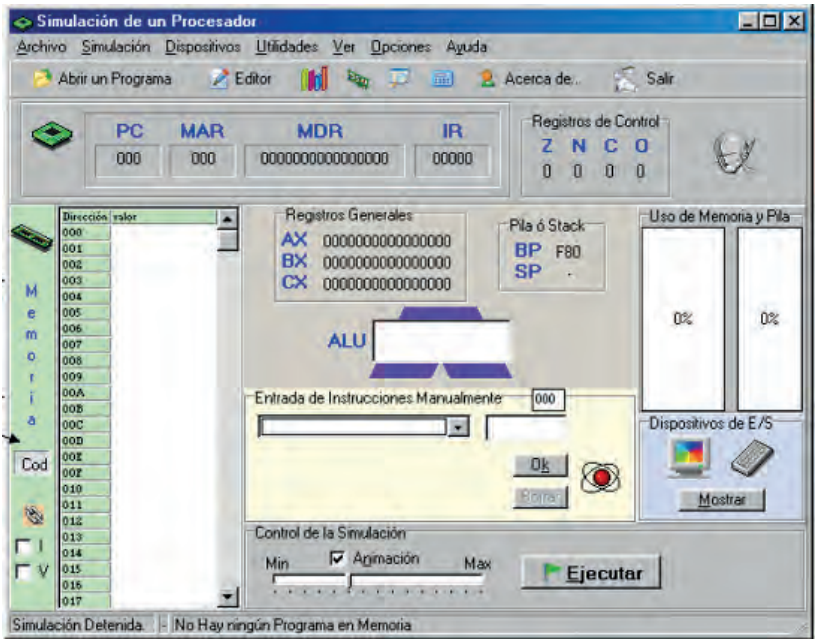


Figura B.1: SimuProc

En la figura podemos observar cómo se representan y distribuyen en la ventana de ejecución los componentes del procesador descripto anteriormente.

Control de simulación: en la figura también se observa un control de velocidad de simulación que permite graduar la rapidez con la que el procesador ejecutará el programa cargado en memoria.

Ventana de entrada/salida de datos: se acciona picando sobre el botón "Mostrar". Aquí se puede interactuar con el procesador observando los resultados que va arrojando el programa en ejecución o bien ingresando datos requeridos por el programa.

Definiciones generales: por medio de los controles que se mencionaron, esta interfaz gráfica permite observar e intervenir en la ejecución del programa, donde se verá cómo se carga una instrucción de memoria en los registros del procesador, cómo se decodifica y realizan cálculos en la ALU, y cómo se modifican los registros de estado según el resultado de las operaciones matemáticas realizadas en la ALU.

En la siguiente figura se observa la estructura del editor de programas.

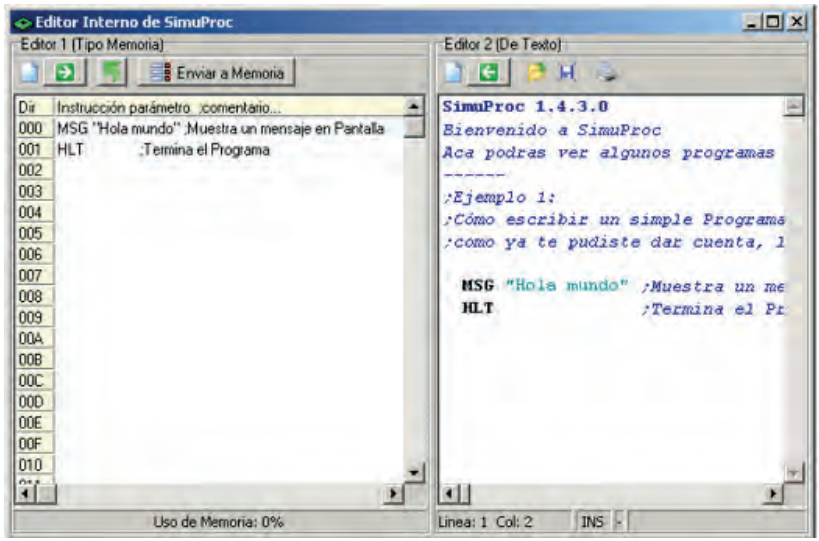


Figura B.2: Editor del SimulProc

Editor 2 (ventana derecha):

Aquí es donde se comienza con el proceso de implementación de los algoritmos obtenidos en la fase de resolución. Para este fin se introducen los mnemónicos de las instrucciones que se van a ejecutar. También se podrán agregar comentarios que hagan más sencilla la lectura e interpretación del algoritmo codificado en lenguaje ensamblador.

La documentación es muy importante en las fases de prueba y depuración, SimulProc permite agregar comentarios para este fin.

Editor 1 (ventana izquierda):

Aqué se verá cómo quedarán distribuidas en la memoria las instrucciones ingresadas en el editor 1. También tiene un asistente de corrección de errores donde acusará, a través de un mensaje, todas aquellas inconsistencias que pudieran tener las instrucciones cargadas.

Especificaciones de SimulProc

Memoria RAM:

- 4096 posiciones (o 512 bytes) de 16 bits cada palabra de memoria.
- Primer posición = 000x0.
- Última posición = FFFx0.
- Capacidad en bytes (8 bits) = 8KBytes (pos.c de mem. x 2 bytes por palabra).

Registros (AX, BC, CX, MAR, MDR, IR):

- Tres registros de propósito general de 16 bits cada uno (AX, BX, CX).
- Un registro apuntador PC o IP (*program counter* o *instruction pointer*) almac prox. dir de mem. a acceder 12 bits
- Una *memory addres register*, almac. la dir. de memoria a la que se va a acceder 12 bits.
- Una *memory data register* o *memory buffer*, reg. Almac. las palabras que se leen o escriben de memoria 16 bits.
- Máximo entero positivo aceptado por la entrada de datos: 65535.
- Registro de instrucción, este registro contiene el código de operación y los operandos. Aquí se decodifica e interpreta la instrucción

Registros de pila (BP, SP, STACK):

- Puntero base (BP), por default vale F80 x 0. Este puntero indica desde que dir. de memoria RAM está reservado para la estructura de almacenamiento llamada "pila". Ya que usa la memoria como medio físico, hereda la longitud de palabra de 16 bits de ella.
- *Stack Pointer* o puntero de pila (SP), indica cuál es la próxima dirección de la pila disponible, en otras palabras, apunta a la cima de la pila.
- Pila, utiliza parte de la memoria del procesador como medio de almacenamiento, tiene 128 posiciones (o 16 bytes). La estrategia de acceso es FILO (*first in last out*).

Registro de control (flags: Z, N, C, O):

- También llamado registro de estado, permite monitorear el comportamiento de un programa en tiempo de ejecución. Esto se lleva a cabo observando los flags que se actualizan luego de la ejecución de una instrucción en el procesador.

ALU:

- Unidad aritmética lógica, lugar donde se realizan las operaciones matemáticas y lógicas (+, -, /, * o AND, OR, bits shift)

Modos de direccionamiento del SimulProc

Direccionamiento implícito

$XAB \leftarrow AX \Leftrightarrow BX$

$CLA \leftarrow AX = 0$

Direccionamiento de registro

PUSH [registro]

POP [registro]

Direccionamiento directo

$LDA[mem] \leftarrow AX = [mem]$

$STA[mem] \leftarrow [mem] = [AX]$

Direccionamiento indexado

$STB[mem] \leftarrow [[mem] + [BX]] = AX$

$LDB[mem] \leftarrow AX = [[mem] + [BX]]$

En este modo, BX hace las veces de registro índice.

Funcionamiento del simulador paso a paso

El funcionamiento del UPC simulado por el *software* "SimulProc" consta de dos etapas. En la primera, el UPC se ocupa de recuperar la instrucción de memoria "FETCH" (y obtener los datos si así la instrucción lo indica) y, en la segunda, de ejecutar dicha instrucción "EXECUTE" que puede involucrar la operación de los datos previamente recogidos en la etapa de búsqueda.

A continuación, se desarrollan paso a paso las dos etapas señalando los registros usados y tareas realizadas:

1. Cargar el PC con la próxima dirección de memoria 000x0 a ejecutar.
2. Envía al MAR la dirección de memoria a leer.
3. Se carga en el MDR el contenido de la dirección apuntada por el MAR.
4. Se entrega al IR el contenido para que lo decodifique e incremente PC.
5. SI la instrucción indica un operando se carga en el MAR la dirección de dicho dato y se continúa desde el paso 2, DE OTRA FORMA, se ejecuta la instrucción decodificada.

Ejecución paso a paso del SimulProc

En las siguientes figuras se podrá ver, paso a paso, cómo evolucionan durante la ejecución de un programa que suma tres números almacenados en la memoria (posc. 010x0, 011x0, 012x0) los registros de Simulproc. La notación usada para expresar posc. de memoria será en hexadecimal con el formato: - - - x 0.

El programa se vería escrito y enviado a la memoria de la siguiente forma.

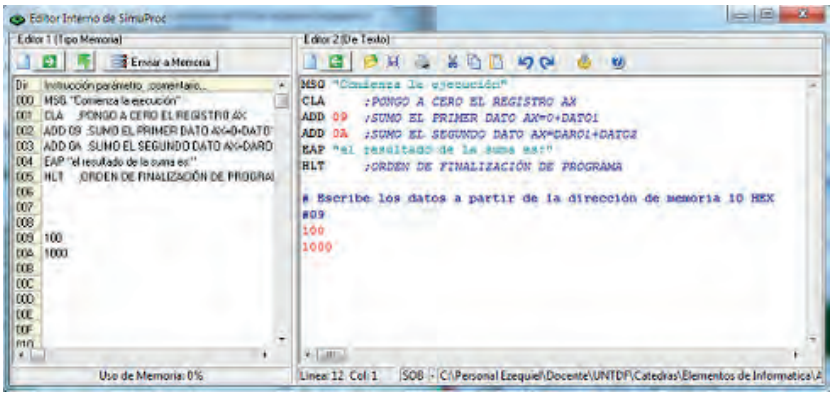


Figura B.3: Ejecución paso a paso del SimulProc

1- Al comenzar la ejecución del programa el simulador SIEMPRE carga PC con la dirección de memoria 0 0 0 x 0.

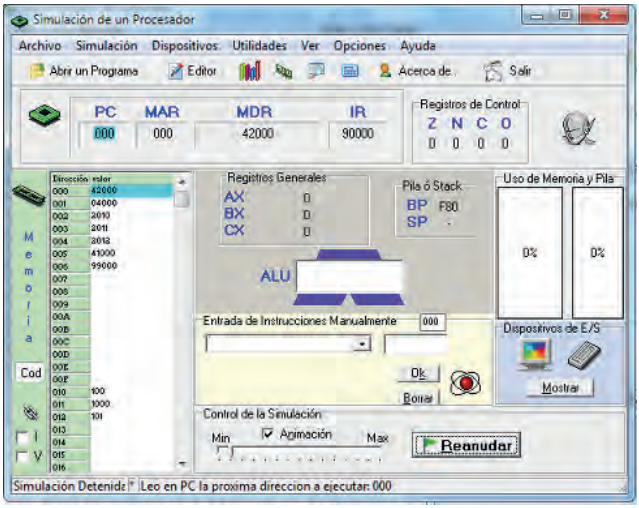
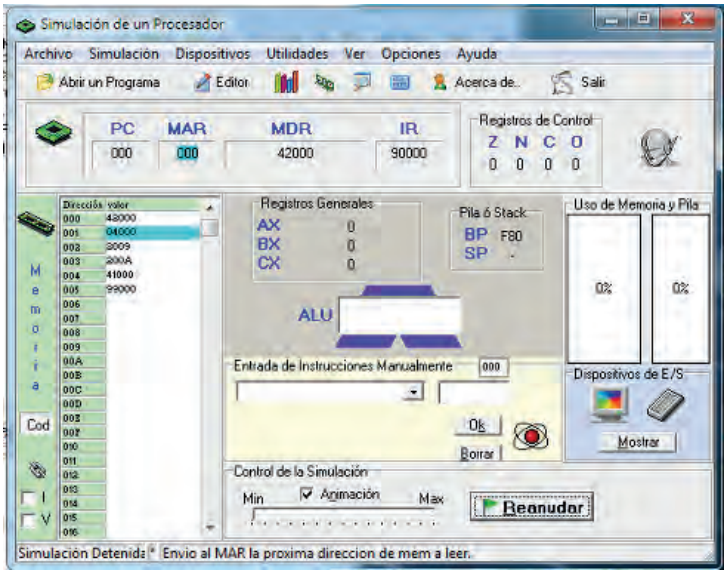
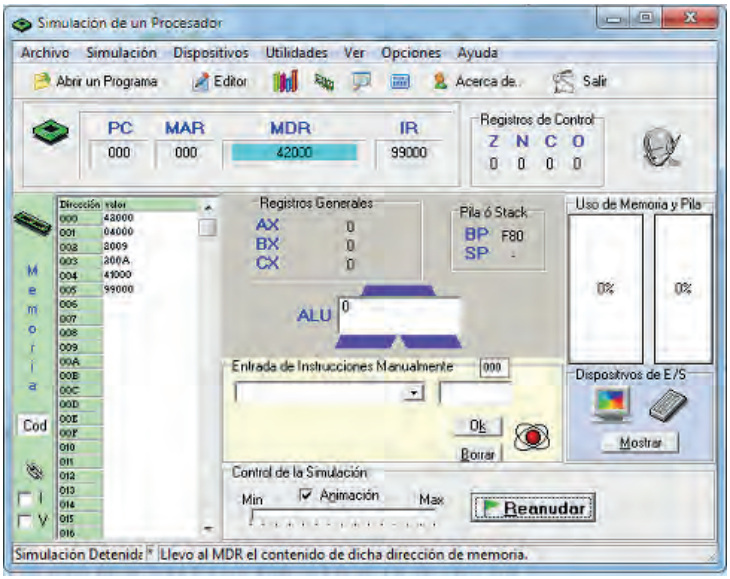


Figura B.4: Ejecución paso a paso del SimulProc

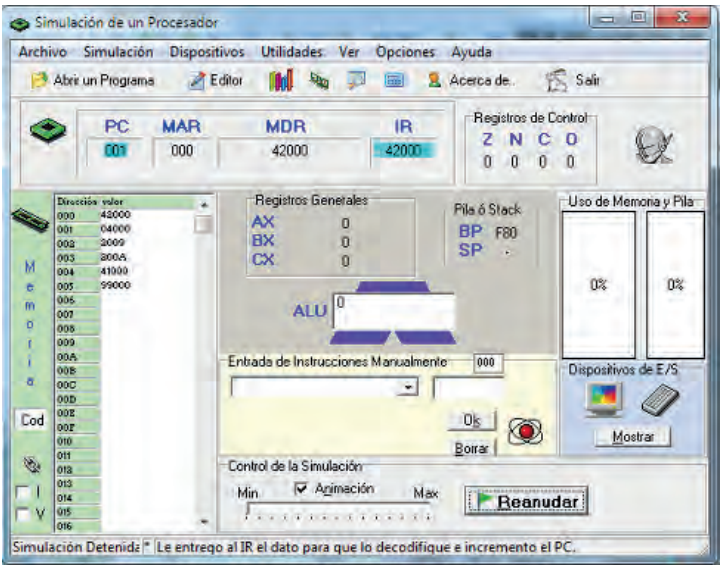
2- Se envía al MAR la dirección apuntada por el PC.



3- Se carga el MDR con el contenido de la dirección de memoria apuntada por el MAR.

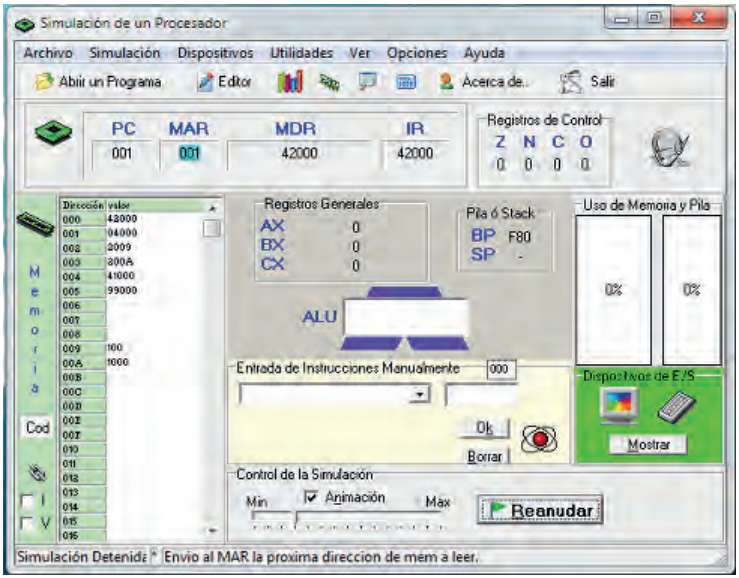


4- Se transfiere el contenido del MDR al IR para que sea posteriormente decodificado. Simultáneamente se incrementa el PC.

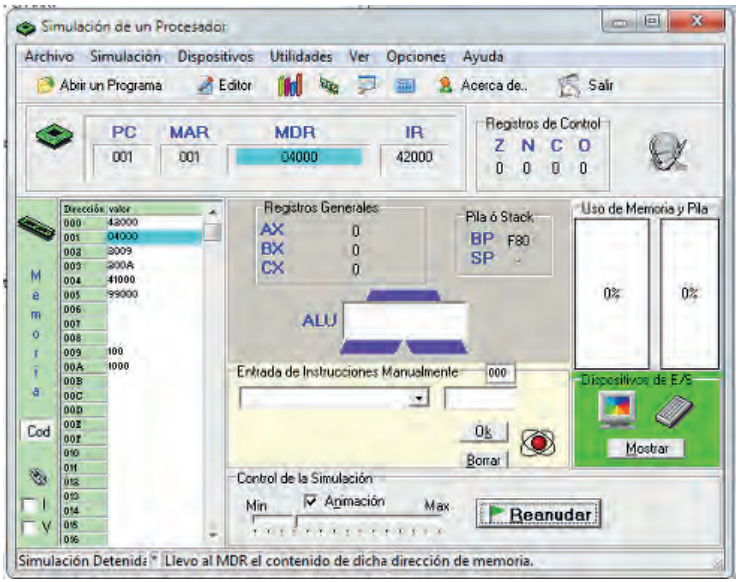


5- Se imprime en la salida estándar (la pantalla) el mensaje indicado en la instrucción.

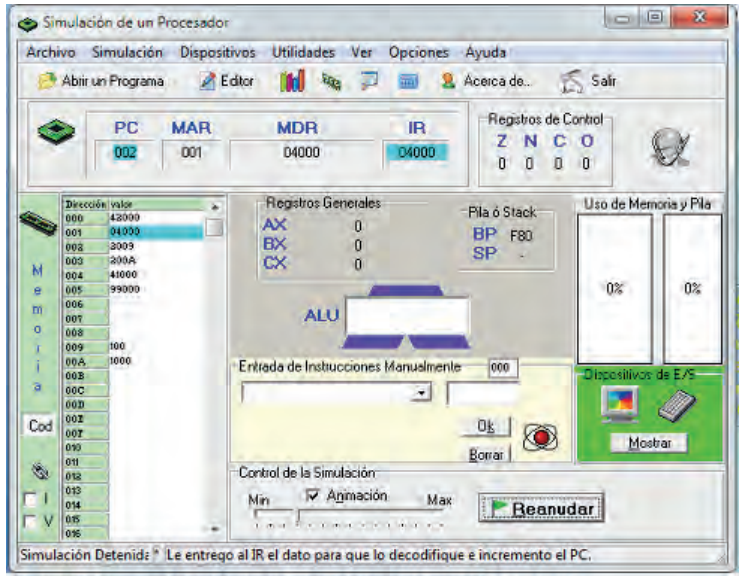
6- Se lee la próxima instrucción a ejecutar en el PC y se transfiere al MAR



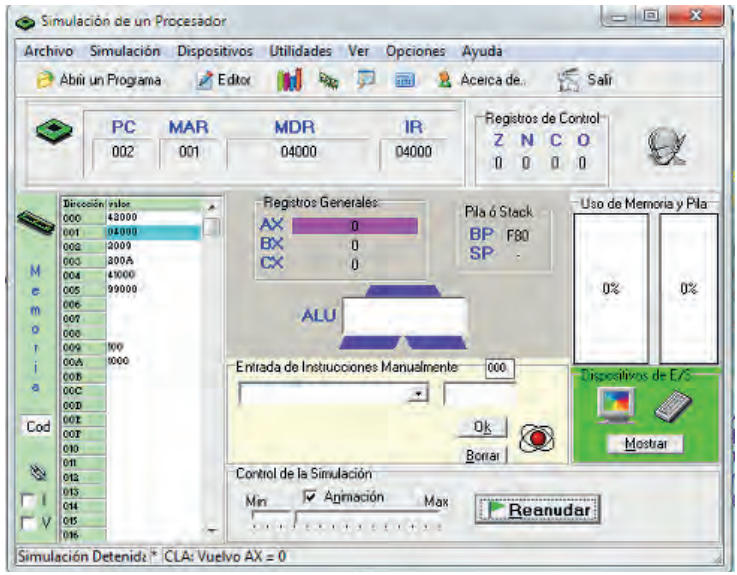
7- Se carga en el MDR el contenido de la dirección de memoria indicada en el MAR.



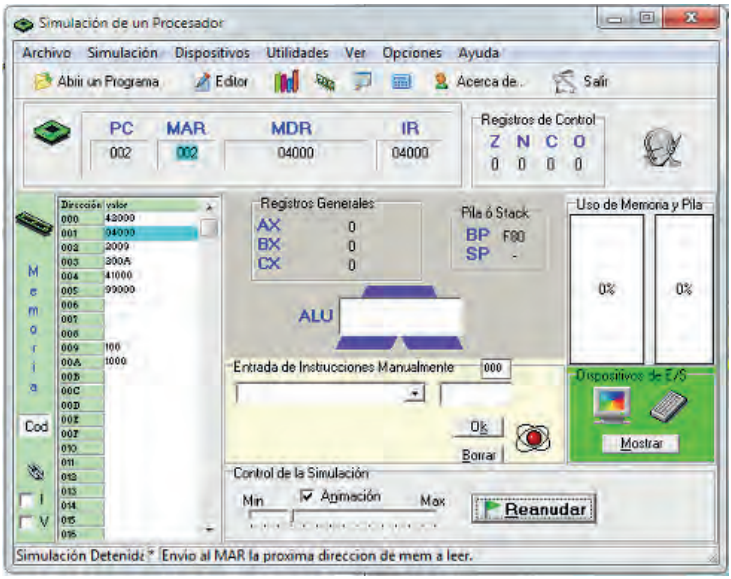
8- Se carga en el IR para que la instrucción sea decodificada e incremente el PC.



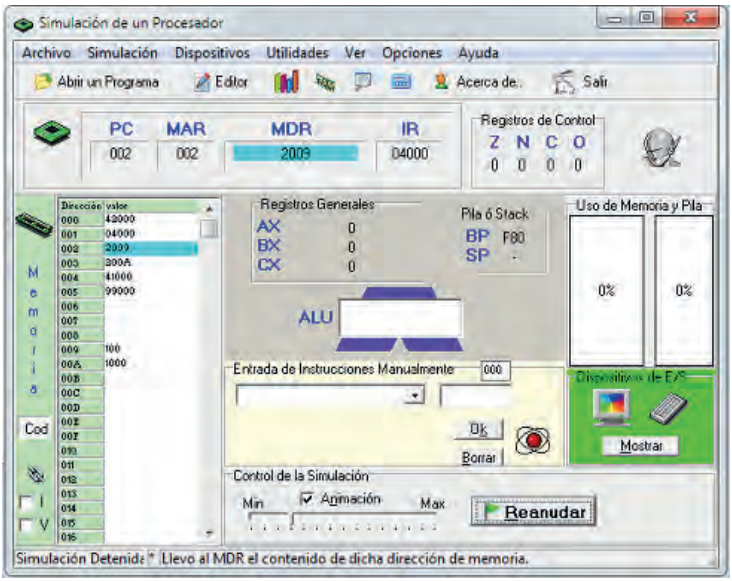
9- Se ejecuta la instrucción. En este caso se inicializa el registro AX seteándolo a cero (0). El PC ya está cargada con la siguiente instrucción a ejecutar.



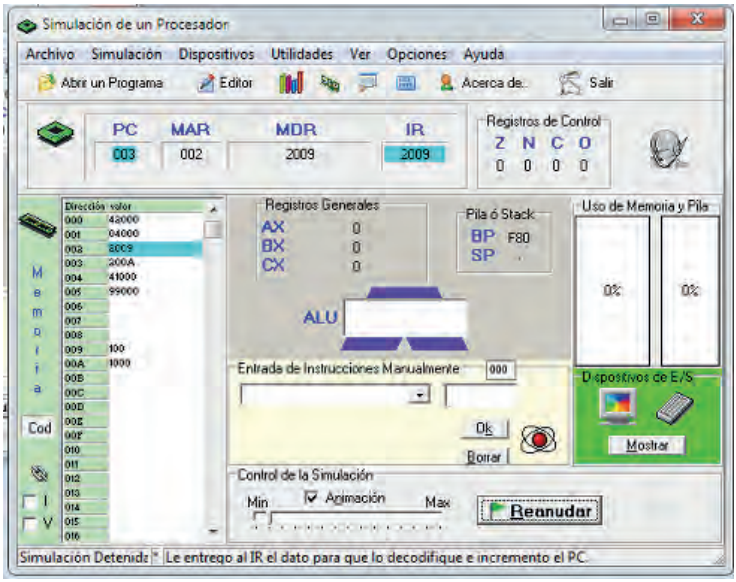
10- Envío al MAR de la dirección de memoria a ejecutarse (ahora la 002)



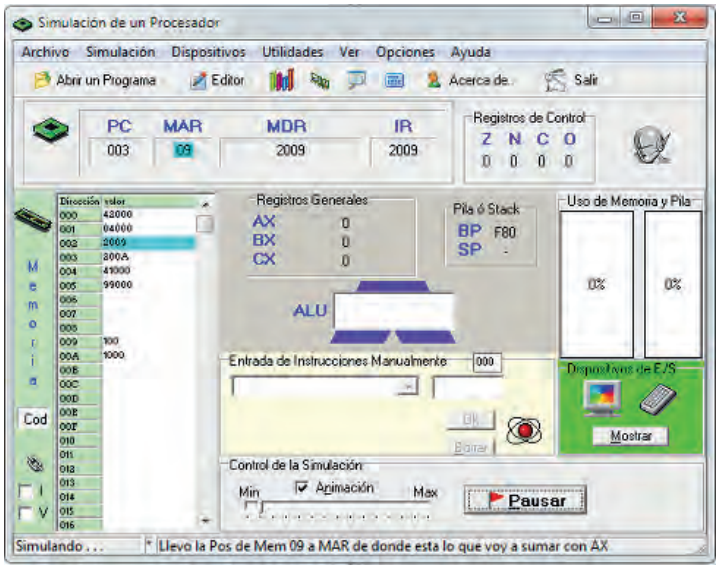
11-Se transfiere al MDR el contenido de la dirección de memoria 002x0



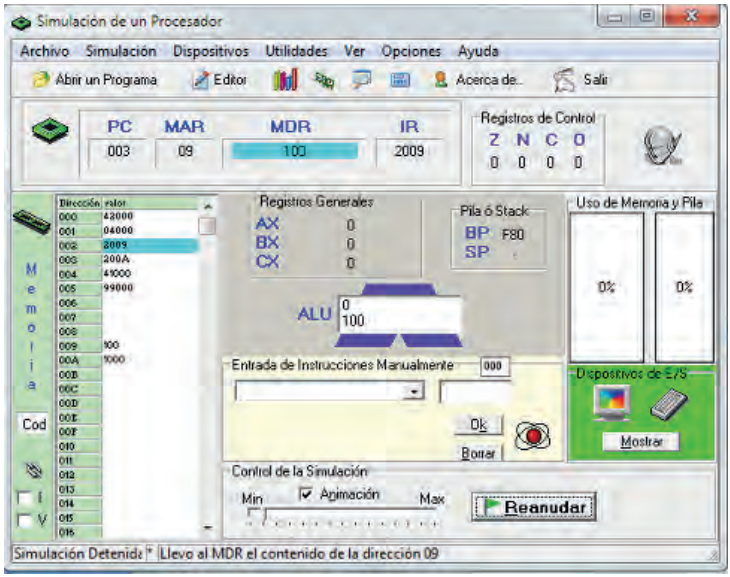
12-Le transfiero al IR la instrucción CO: 20 OP: 09, donde CO: código de operación y OP: operando.



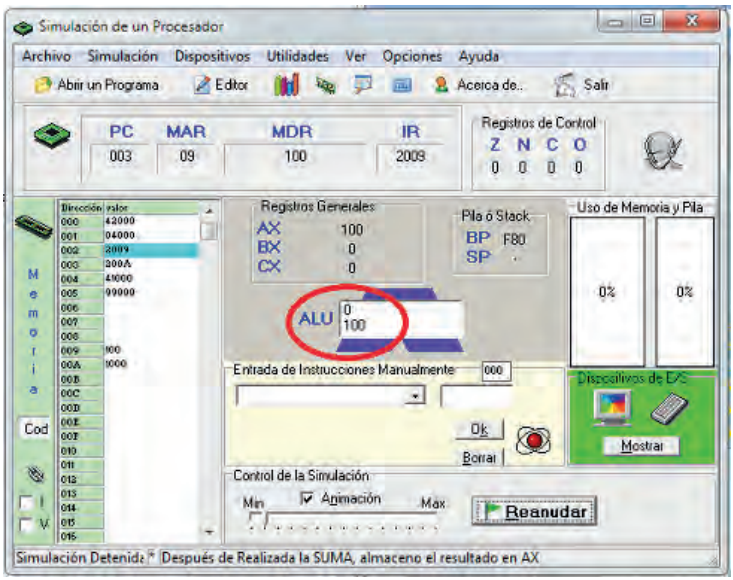
13- Cargo en el MAR la dirección de memoria donde está el dato en este caso 09x0.



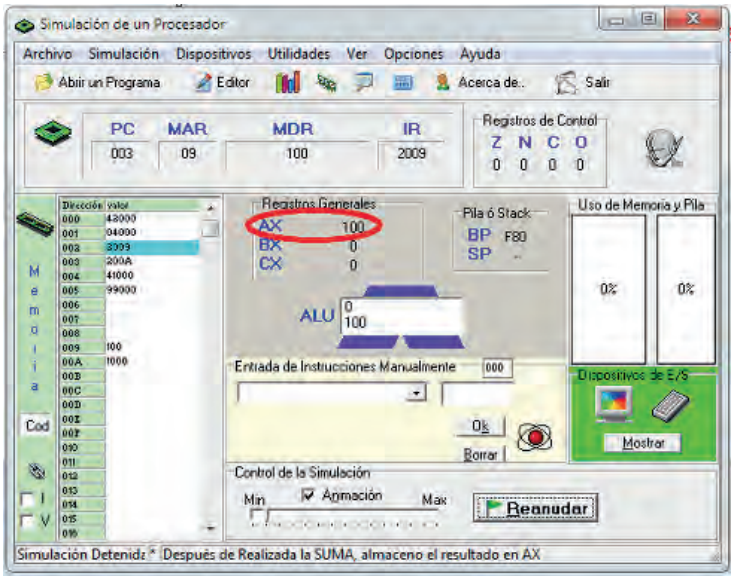
14- Recupero el dato cargado en 09x0 y lo cargo en MDR.



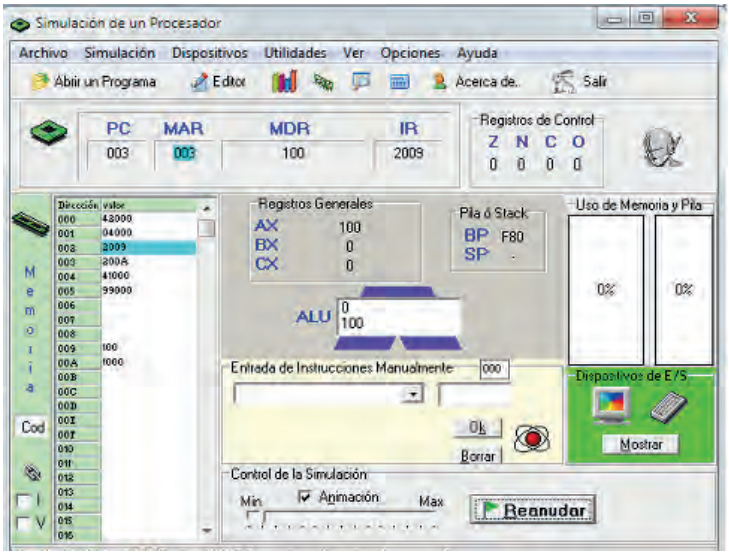
15- Los datos ya se encuentran cargados en la ALU y listos para ser operados.



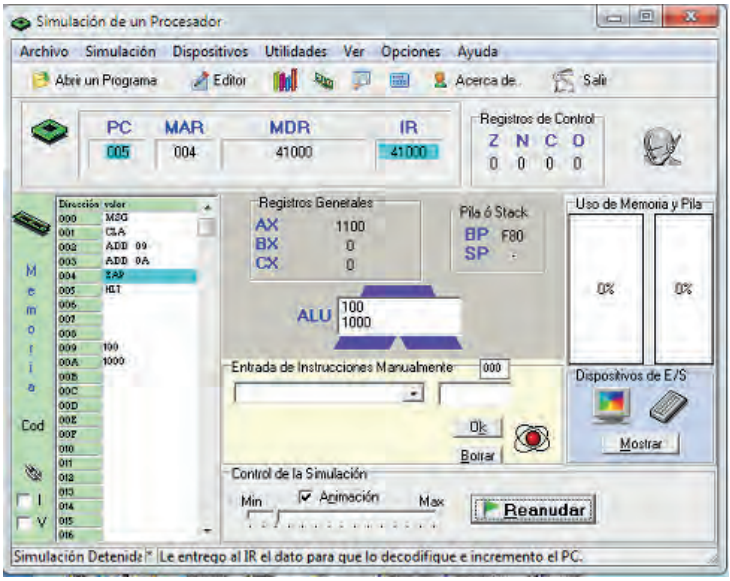
16- Una vez realizada la suma, inmediatamente se carga en AX el resultado. No perder de vista esto ya que el valor anterior de AX es reemplazado destruyendo el anterior (esta operación actualiza los flags, aunque visualmente permanezcan en 0).



17- Y comienza el ciclo nuevamente. Se lee en el PC la próxima dirección de memoria.



Las instrucciones que continúan tienen un desarrollo similar por lo que la traza del programa saltará directamente a la dirección de memoria 004x0



Se visualiza el resultado de la ejecución de la instrucción. Observándose el valor en decimal del contenido de AX y una descripción

Luego se cargará la última dirección y se ejecutará dando por finalizado en programa. Una vez finalizado el programa, SimulProc presenta una ventana de estadísticas de la traza del programa

Instrucciones soportadas por SimulProc

Descripción formato genérico de instrucción:

XX - INST [parámetro]

Donde:

XX es el código de la instrucción expresado en dos dígitos decimales.

INST es la instrucción, expresada como mnemónico.

[Parámetro] es el parámetro si esta tiene o [parametro1, parametro2] si el parámetro es doble, que puede indicar un registro o una dirección de memoria.

01 - LDA [mem]

Cargue en AX el contenido de la dirección de memoria especificada. Digamos que en la posición de memoria 1F está el valor 10111, después de ejecutada la instrucción LDA 1F se obtiene que AX = 10111.

Es equivalente a usar la instrucción MOV AX, 1F. Hay casos donde es mejor usar MOV si se desea pasar datos sin tener que pasarlos por AX.

02 - STA [mem]

Guarda el contenido de AX en la dirección de memoria especificada. Supongamos que se tiene el valor 1010110 en el registro AX y se quiere llevarlo a la posición de memoria 3C, la instrucción es STA 3C.

Es equivalente a usar la instrucción MOV 3C, AX. Es mejor usar MOV debido a que si se quiere pasar algún dato de una dirección de memoria a otra usando LDA y STA serían dos instrucciones: LDA mem1 y, luego, STA mem2, mientras que con MOV será así: MOV mem2,mem1.

03 - XAB

Intercambia los valores de los registros AX y BX. Esta instrucción no necesita parámetros.

04 - CLA

Hace AX = 0.

06 - PUSH [registro]

Envía el valor del registro especificado a la pila.

07 - POP [registro]

Trae de la pila el último valor llevado por PUSH (indicado por el registro SP) y lo almacena en el registro especificado.

08 - INC [dest]

Incrementa en 1 el destino especificado, el parámetro puede ser una dirección de memoria o un registro. Si en la posición de memoria EB está el valor 1001, al ejecutar INC EB se obtiene que ahora el valor de EB sea 1010.

09 - DEC [dest]

Decremento en 1 el destino especificado. Si el destino queda = 0, se vuelve Z = 1

10 - MOV [dest,orig]

Copia el valor almacenado en el origen al destino. El destino u origen pueden ser registros o direcciones de memoria o combinación de estos. Para copiar lo que está en la posición de memoria 12E a la posición D2 se usa la instrucción MOV D2,12E.

11 - AND [dest,orig]

Y lógico, hace un Y lógico entre todos los bits de los dos operandos, escribiendo el resultado en el destino. Los parámetros pueden ser direcciones de memoria o registros.

Si en AX está el número 1001101 y en la pos. 3F está el número 11011, al ejecutar la instrucción AND AX, 3F obtendré en AX el resultado 1001. El Y lógico deja los bits en común que tengan los dos números.

12 - NOT [destino]

NO lógico, invierte los bits del operando formando el complemento del primero. Si en AX tengo 10011 al ejecutar NOT AX obtengo AX=1111111111101100

13 - OR [dest,orig]

O inclusive lógico, todo bit activo en cualquiera de los operandos será activado en el destino. Si en 3A tengo el número 1001101 y en la pos 3B tengo el número 11011, al ejecutar la instrucción OR 3A,3B obtendré en 3A el resultado 1011111.

14 - XOR [dest,orig]

O exclusivo, realiza un O exclusivo entre los operandos y almacena el resultado en destino. Si en 3A tengo el número 1001101 y en la pos 3B tengo el número 11011, al ejecutar la instrucción XOR 3A,3B obtendré en 3A el resultado 1010110.

15 - ROL [dest,veces]

Rota los bits a la izquierda las veces especificadas (en decimal), los bits que salen por la izquierda re-entran por la derecha. En el carry flag queda el último bit rotado. Supongamos que en la posición 7E está el número 101110.

Al ejecutar obtengo en 7E C=

ROL 7E,2 10111000 0

ROL 7E,7 1011100000000 0

ROL 7E,13 1100000000000101 1

16 - ROR [dest,veces]

Rota los bits a la derecha las veces especificadas (en decimal), los bits que salen por la derecha re-entran por la izquierda. El carry flag guarda el último bit rotado.

17 - SHL [dest,veces]

Desplaza los bits a la izquierda el número de veces especificado (en decimal), agregando ceros a la derecha, el carry flag guarda último bit desplazado.

18 - SHR [dest,veces]

Desplaza los bits a la derecha el número de veces especificado (en decimal), agregando ceros a la izquierda, el carry flag guarda último bit desplazado.

Supongamos que en la posición 1A está el número 101110

Al ejecutar... obtengo en 1A C=

SHR 1A,2 1011 1

SHR 1A,6 0 1

SHR 1A,11 0 0

20 - ADD [mem]

Sumar: AX = AX + el contenido de la dirección de memoria. Si el resultado de la suma supera los 16 bits, el resultado queda así: en BX los bits más significativos y en BX los menos, también se activa el Overflow flag.

21 - SUB [mem]

Restar: AX = AX - el contenido de la dirección de memoria.

22 - MUL [mem]

Multiplicar: AX = AX * el contenido de la dirección de memoria. Si el número resultante supera su longitud en binario de 16 bits, este resultado se parte almacenando los bits más significativos en el registro BX. Si el resultado de una operación fuera 10110100011110001011 1 (21 bits)

Los registros quedarán así:

A = 1000111100010111 (los últimos 16 bits)

B = 10110 (los primeros bits (los más significativos))

También se activa el flag overflow, para indicar que en la última operación sucedió esto.

23 - DIV [mem]

Dividir: AX = AX / el contenido de la dirección de memoria, BX=AX, el contenido de la dirección de memoria (BX = módulo o residuo).

25 - CLC
Limpia el carry flag. C = 0.

26 - STC
Pone el carry flag. C = 1.

27 - CMC
Complementa (invierte) el carry flag. Si C = 1 vuelve C = 0, y viceversa.

29 - LOOP [mem]
Decrementa CX y salta a la pos de memoria si CX no es cero.

30 - JMP [mem]
Salto incondicional. PC = dirección de memoria donde está la siguiente instrucción a ejecutar.

Generalmente las instrucciones de salto condicional 31, 33, 34 y 39 se utilizan inmediatamente después de usar la instrucción 32 - CMP.

31 - JEQ [mem]
Saltar si son iguales. Si Z = 1, PC = contenido de la memoria. Función equivalente en C: if (AX == mem)

32 - CMP [mem]
Compara AX con [mem], si AX es mayor, Z = 0 N = 0, si es igual Z = 1 N = 0, si es menor Z = 0 N = 1. Supongamos que en AX está el número 10110 y en la posición de memoria 4G está el número 1100, al ejecutar la instrucción CMP 4G se obtiene que, como el número almacenado en AX es mayor, entonces, los flags de control quedan así: Z = 0 y N = 0.

33 - JME [mem]
Saltar si es menor. Si N = 1, PC = contenido de la memoria. Supongamos que ejecuto esta instrucción como JME. 3F inmediatamente después de ejecutar la instrucción del ejemplo que se coloque en la instrucción 32, al ejecutar JME 3F se verifica el flag N, y como en este caso se encuentra en 0 (porque el número no es menor) entonces no se realiza dicho salto a 3F, porque el valor de PC no se modifica, el programa sigue su ejecución.

34 - JMA [mem]

Saltar si es mayor. Si Z = 0 y N = 0, PC = contenido de memoria. Supongamos que ejecuto esta instrucción como JMA 2B inmediatamente después de ejecutar la instrucción del ejemplo que coloque en la instrucción 32, al ejecutar JMA 2B se verifican los flag N y Z, y como en este caso los dos son 0 (porque el número es menor), entonces si se realiza dicho salto a 2B ya que el valor del PC se modifica, el programa sigue su ejecución saltando a la dirección de memoria especificada.

35 - JC [mem]

Saltar si el carry flag está activado. Si C = 1, PC = contenido de memoria.

36 - JNC [mem]

Saltar si el carry flag no está activado. Si C = 0, PC = contenido de memoria.

37 - JO [mem]

Saltar si el overflow flag está activado .Si O = 1, PC = contenido de memoria.

38 - JNO [mem]

Saltar si el overflow flag no está activado. Si O = 0, PC = contenido de memoria.

39 - JNE [mem]

Saltar si no son iguales. Si Z = 0, PC = contenido de memoria.

40 - LDT

Lee un valor del teclado y lo lleva al registro AX. Esta instrucción es para comunicarse con el usuario, pidiéndole que entre un dato. Puede colocar una descripción del dato que pide, que se mostrará en tiempo de ejecución.

41 - EAP

Escribe en pantalla el contenido del registro AX. Esta instrucción también es para comunicarse con el usuario. Puede colocar una descripción del dato que se entrega, este se mostrará en tiempo de ejecución.

42 - MSG

Muestra un mensaje en pantalla, ejemplo, MSG “Hola Mundo!!”

50 - LDB [mem]

La instrucción carga en AX el contenido de memoria almacenado en [mem] + BX

Ejemplo, digamos que BX = 10; LDB 1A carga el contenido de 1C en AX.

51 - STB [mem]

Guarda el contenido de AX en la dirección [mem] + BX. Ejemplo, digamos que BX = 101; STB 3A guarda AX en 3F.

55 - LDF [mem]

Carga en BX y AX un número de 32 bits (IEEE) que está almacenado en la dir. [mem] y mem+1. En BX quedan los dígitos más significativos.

Ej: Supongamos que el número 01000010110010001000000000000000 está cargado en memoria como:

02A 0100001011001000 (los dígitos más significativos)

02B 1000000000000000 (los dígitos menos significativos)

Un LDF 2A dejará el siguiente resultado:

BX: 0100001011001000

AX: 1000000000000000’

56 - STF [mem]

Guarda en [mem] y mem+1 el contenido de BX y AX. Ejemplo: siendo AX y BX = al ejemplo anterior, un STF 2A deja la memoria como el ejemplo anterior.

60 - ADDF [mem]

Suma números de 32 bits: en BX y AX queda el resultado de la suma de estos más el contenido de [mem] y mem+1. Estos números IEEE 754 pueden ser de punto flotante, o enteros desde -2147483647 hasta 2147483647, si en cualquier operación de estas aritméticas, se sobrepasa este valor, se activa el overflow flag.

61 - SUBF [mem]

Resta el número de 32 bits: BX y AX = BX y AX - [mem] y mem+1. Se puede utilizar esta instrucción como un CMP para números de 32 bits.

62 - MULF [mem]

Multiplicación: BX y AX = BX y AX * [mem] y mem+1. Si el resultado es > 2147483647, Resultado = 2147483647 y overflow flag = 1.

63 - DIVF [mem]

División: BX y AX = BX y AX / [mem] y mem+1, en CX queda el residuo de la división en entero de 16 bits.

64 - ITOF

Conversión de entero a real: Convierte un número entero (16 bits) almacenado en AX al mismo número, pero representado en real IEEE754 (32 bits), el resultado de la conversión queda en BX (bits más significativos) y AX.

Los registros de control cambian de acuerdo al número convertido: "Z" si el número es cero, "N" si el número es negativo.

65 - FTOI

Conversión de real a entero: Convierte un número real (32 bits) a su equivalente en entero BX y AX en un entero (16 bits), el resultado queda en AX. Los registros de control cambian de acuerdo al número convertido: "Z" si el número es cero, "N" si el número es negativo, "O" si el número real es mayor de 65535.

80 - IN registro, puerto

Lleva al registro el valor retornado por el puerto especificado. Ejemplo: IN AX,8 ;lleva a AX el valor retornado por el puerto 8 (reloj: los segundos del sistema).

81 - OUT puerto, registro

Escribe en el puerto especificado, el valor del registro.

90 - NOP

Esta operación no hace nada. Útil para cuando se modifica la memoria para rellenar código y desactivar instrucciones.

99 - HLT

Terminar programa. Todo programa lleva esta instrucción para indicarle al simulador que el programa ha terminado su ejecución.

Ejercicios típicos

Ejemplo 1:

El factorial de un número entero es aquel resultado que se obtiene de multiplicar la sucesión de números empezando desde uno hasta llegar al número indicado.

Por ejemplo el factorial de 4 se resuelve como sigue:

4! = 1 x 2 x 3 x 4 = 24

5! = 1 x 2 x 3 x 4 x 5 = 120

Nota: el algoritmo debe considerar el caso particular del 0, donde 0! = 1.

Entonces, deberíamos:

1. Inicializar acumulador.
2. Solicitar el número a calcular el factorial.
3. Verificar si es cero el número ingresado.
    1. Si es cero el factorial, es igual a 1.
    2. Termina el algoritmo.
4. Copio el valor ingresado a una variable auxiliar.
5. Le resto uno a la variable auxiliar.
6. Verifico si es cero la variable auxiliar.
    1. Si es cero el factorial, es el valor que está en el acumulador.
    2. Termina el programa.
7. Multiplico el acumulador con la variable auxiliar guardando el resultado en acum.
8. Vuelvo a 5.

SOLUCIÓN: Codificación (solución particular) en lenguaje ensamblador.

```
00 MSG “Comienzo del programa”
01 CLA; inicializo AX
02 MOV 10, AX; inicializo la dir. de mem. 10
03 LDT “Ingrese el valor a calcular factorial el valor debe estar entre 0 y 8”
04 CMP 10; verifico si el valor ingresado es cero
05 JEQ 11; si la comparación me da que son iguales, voy a 11
06 STA 10; almaceno en pos dir. 10 el valor ingresado
07 DEC 10; decremento dir. 10 de mem.
08 JEQ 15; si el decremento dejo en cero a pos de mem. 10 voy a 15
09 MUL 10; multiplico por el valor menor
0A JMP 7; vuelvo a 7 de nuevo
#11
11 INC AX
12 EAP “El factorial del valor ingresado es:”
13 HLT
#15
15 EAP “El factorial del valor ingresado es:”
16 HLT
```

Ejemplo 2:

Dado el siguiente código, determinar qué hace el programa y obtener la traza del mismo.

Nota: al lado de cada instrucción, describir la explicación que realiza.

```
000 MSG ’TP5 Ej 1’; muestro el mensaje en pantalla
001 MSG ’XXX’; muestro el mensaje en pantalla
002 LDT; solicito número entero
003 STA 11; guardo el valor ingresado en la dir. memoria 11
004 DIV 10; lo divido por 2 para saber si es par
005 LDA 11
006 JEQ B; salto a la dirección de memoria A si el número ingresado es par
007 MSG’ El nro:
008 EAP
009 MSG’ es impar’
00A HLT
```

#B
00B MSG' El nro:
00C EAP
00D MSG' es PAR'
00E HLT
#10
010 00000010

Es decir, determina si un número es par o impar.

La traza sería:

Registros		Memoria		Reg Control				Salida de pantalla	Dato	Comentarios
PC	AX	10	11	Z	N	C	O			
000x0	—	$(10)_2$	—	0	0	0	0	—		Comienza la ejecución PC = 000
001x0	—	$(10)_2$	—	0	0	0	0	Tp5 Ej 1		Muestra mensaje en pantalla
002x0	—	$(10)_2$	—	0	0	0	0	Ejercicio de paridad		Muestra mensaje en pantalla
003X0	$(100)_2$	$(10)_2$	—	0	0	0	0		4	Pide dato e ingresa 4 el usuario
004x0	$(100)_2$	$(10)_2$	$(100)_2$	0	0	0	0			Se carga 4 en dir 011
005x0	$(100)_2$ / $(10)_2$	$(10)_2$	$(100)_2$	1	0	0	0			Se divide 4 / 2 lo que da resto 0 y pones a Z = 1
006x0	$(100)_2$	$(10)_2$	$(100)_2$	1	0	0	0			Se trae de mem 011 el valor 4 y se carga en AX
00Bx0	$(100)_2$	$(10)_2$	$(100)_2$	1	0	0	0			Salto condicional positivo PC = 00B
00Cx0	$(100)_2$	$(10)_2$	$(100)_2$	1	0	0	0	El nro:		Muestra mensaje en pantalla
00Dx0	$(100)_2$	$(10)_2$	$(100)_2$	1	0	0	0	4		Muestra en pantalla el valor de AX
00Ex0	$(100)_2$	$(10)_2$	$(100)_2$	1	0	0	0	es PAR		Muestra mensaje en pantalla
00Fx0	$(100)_2$	$(10)_2$	$(100)_2$	1	0	0	0			Ejecuta fin de programa

TABLA DE CARACTERES ASCII

ASCII son las siglas de *American Standar Code for Information Interchange*. Su uso primordial es facilitar el intercambio de información entre sistemas de procesamiento de datos y equipos asociados y dentro de sistemas de comunicación de datos. En un principio, cada carácter se codificaba mediante siete dígitos binarios y fue creado para el juego de caracteres ingleses más corriente, por lo que no contemplaba ni caracteres especiales ni caracteres específicos de otras lenguas. Esto hizo que, posteriormente, se extendiera a ocho dígitos binarios.

	000 0	001 1	010 2	011 3	100 4	101 5	110 6	111 7
0000 0	NUL 0	DLE 16	SP 32	0 48	@ 64	P 80	` 96	p 112
0001 1	SOH 1	DC1 17	! 33	1 49	A 65	Q 81	a 97	q 113
0010 2	STX 2	DC2 18	" 34	2 50	B 66	R 82	b 98	r 114
0011 3	ETX 3	DC3 19	# 35	3 51	C 67	S 83	c 99	s 115
0100 4	EOT 4	DC4 20	$ 36	4 52	D 68	T 84	d 100	t 116
0101 5	ENQ 5	NAK 21	% 37	5 53	E 69	U 85	e 101	u 117
0110 6	ACK 6	SYN 22	& 38	6 54	F 70	V 86	f 102	v 118
0111 7	BEL 7	ETB 23	' 39	7 55	G 71	W 87	g 103	w 119
1000 8	BS 8	CAN 24	(40	8 56	H 72	X 88	h 104	x 120
1001 9	HT 9	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121
1010 A	LF 10	SUB 26	* 42	: 58	J 74	Z 90	j 106	z 122
1011 B	VT 11	ESC 27	+ 43	; 59	K 75	[91	k 107	{ 123
1100 C	FF 12	FS 28	´ 44	< 60	L 76	\ 92	l 108	| 124
1101 D	CR 13	GS 29	- 45	= 61	M 77	] 93	m 109	} 125
1110 E	SO 14	RS 30	. 46	> 62	N 78	^ 94	n 110	~ 126
1111 F	SI 15	US 31	/ 47	? 63	O 79	_ 95	o 111	DEL 127

Tabla del código ASCII 7 bits

Figura C.1

Caracteres de control ASCII:

NUL	Nulo	DLE	Escape del enlace de datos.
SOH	Comienzo de cabeza		Carácter de control que cambia el significado del carácter que se da a continuación
STX	Comienzo de texto		
ETX	Final de texto		
EOT	Final de transmisión	DCi	Control del dispositivo *i*
ENQ	Petición, consulta	NAK	Acuse de recibo negativo
ACK	Acuse de recibo	SYN	Sincronización
BEL	Pitido	ETB	Final de bloque de transmisión
BS	Retroceso de un espacio	CAN	Anulación
HT	Tabulación horizontal	EM	Fin de soporte (de cinta, etc)
LF	Saltar a línea siguiente	SUB	Sustituir
VT	Tabulación vertical	ESC	Escape
FF	Alimentación de hoja	FS	Separador de archivo
CR	Retorno de carro	GS	Separador de grupo
SO	Fuera de código	RS	Separador de registro
SI	Dentro de código	US	Separador de sub-registro (campo)
		DEL	Borrar, suprimir

Figura C.2

TABLA DE CARACTERES EBCDIC

Este código surge como una ampliación del código BCD. En las transmisiones de datos es necesario utilizar un gran número de caracteres de control para la manipulación de los mensajes y la realización de otras funciones. De ahí que el código BCD se extendiera a una representación utilizando ocho bits dando origen al código EBCDIC (*Extended Binary Coded Decimal Interchange Code*).

	0000 0	0001 1	0010 2	0011 3	0100 4	0101 5	0110 6	0111 7	1000 8	1001 9	1010 A	1011 B	1100 C	1101 D	1110 E	1111 F
0000 0	NUL 0	DLE 16	DS 32		SP 64	& 80	- 96						{ 192	} 208	\ 224	0 240
0001 1	SOH 1	DC1 17	SOS 33			/ 81	 97	 113	a 129	j 145	 161	 177	A 193	J 209	 225	1 241
0010 2	STX 2	DC2 18	FS 34	SYN 50	 66	 82	 98	 114	b 130	k 146	s 162	 178	B 194	K 210	S 226	2 242
0011 3	ETX 3	IM 19	 35	 51	 67	 83	 99	 115	c 131	l 147	t 163	 179	C 195	L 211	T 227	3 243
0100 4	PF 4	RES 20	BYP 36	PN 52	 68	 84	 100	 116	d 132	m 148	u 164	 180	D 196	M 212	U 228	4 244
0101 5	HT 5	NL 21	LF 37	RS 53	 69	 85	 101	 117	e 133	n 149	v 165	 181	E 197	N 213	V 229	5 245
0110 6	LC 6	BS 22	ETB 38	UC 54	 70	 86	 102	 118	f 134	o 150	w 166	 182	F 198	O 214	W 230	6 246
0111 7	DEL 7	IL 23	ESC 39	EOT 55	 71	 87	 103	 119	g 135	p 151	x 167	 183	G 199	P 215	X 231	7 247
1000 8	 8	CAN 24	 40	 56	 72	 88	 104	 120	h 136	q 152	y 168	 184	H 200	Q 216	Y 232	8 248
1001 9	RLF 9	EM 25	 41	 57	 73	 89	 105	\ 121	i 137	r 153	z 169	 185	I 201	R 217	Z 233	9 249
1010 A	SMM 10	CC 26	SM 42	 58	cent 74	! 90	| 106	: 122	 138	 154	 170	 186	 202	 218	 234	 250
1011 B	VT 11	CU1 27	CU2 43	CU3 59	. 75	$ 91	, 107	# 123	 139	 155	 171	 187	 203	 219	 235	 251
1100 C	FF 12	IFS 28	 44	DC4 60	< 76	* 92	% 108	@ 124	 140	 156	 172	 188	 204	 220	 236	 252
1101 D	CR 13	IGS 29	ENQ 45	NAK 61	(77	) 93	_ 109	' 125	 141	 157	 173	 189	 205	 221	 237	 253
1110 E	SO 14	IRS 30	ACK 46	 62	+ 78	; 94	> 110	= 126	 142	 158	 174	 190	 206	 222	 238	 254
1111 F	SI 15	IUS 31	BEL 47	SUB 63	| 79	¬ 95	? 111	" 127	 143	 159	 175	 191	 207	 223	 239	 255

Tabla del código EBCDIC

Figura D.1

Caracteres de control EBCDIC:

NUL	Nulo	IGS	Separador para intercambio de grupos
SOH	Comienzo de cabeza	IRS	Separador para intercambio de registros
SOT	Comienzo de texto	IUS	Separador para intercambio de unidad
EOT	Final de texot	DS	Selección de dígito
PF	Perforadora desconectada	SOS	Comienzo de significado
HT	Tabulación horizontal	FS	Separador de campo
LC	Minúscula	BYP	Desviar
DEL	Eliminar, borrar	LF	Alimentación de línea
RLF	Alimentación de linea invertida	ETB	Final de bloque de transmisión
SMM	Comienzo mensaje manual	ESC	Escape
VT	Tabulación vertical	SM	Fijar modo
FF	Alimentación de hoja	ENQ	Solicitud, petición
CR	Retorno de carro	ACK	Acuse de recibo
SO	Fuera de código	BEL	Pitido
SI	Dentro de código	SYN	Sincronzación
DLE	Escape del enlace de datos	PN	Perforadora conectada
TM	Marca de cinta	RS	Detener lectora
RES	Restaurar	UC	Mayúsculas
NL	Pasar a línea siguiente	EOT	Fin de transmisión
BS	Retroceso de un espacio	NACK	Acuse de recibo negativo
IL	*sin función*	SUB	Sustituir
CAN	Cancelar	DC*i*	Control dispositivo *i*
EM	Final de soporte	CU*i*	Control usuario *i*
CC	Control del cursor		
SP	Espacio en blanco		
IFS	Separador para intercambio de archivos		

Figura D.2

TRABAJOS PRÁCTICOS

TRABAJO PRÁCTICO EVOLUCIÓN HISTÓRICA CAPÍTULO 1

1) ¿Qué es un dato?, indique sus cualidades. Ejemplos.

2) ¿Qué es el "procesamiento de datos"? Mencione características y formas.

3) Realice un esquema de la Máquina Analítica de Babagge, describa sus componentes.

4) Elabore una tabla en la que se describan los dispositivos o ideas (precursoras de las computadoras), sus autores y características previos a 1944. Dentro de las características, mencionar la mejora que se incorporaba.

Innovación	Autor/Inventor	Características
Contador de ruedas numéricas, Pascalina	Francia, 1642, Blas Pascal	Era una pequeña caja de madera que tenía en la tapa una hilera de discos numerados, con los agujeros para introducir los dedos y hacerlos girar. Cada disco tenía una ventanilla, y había toda una hilera de ventanillas bajo la hilera de discos: de derecha a izquierda se alineaban las unidades, decenas, centenas, milésimas, etc. Permitía realizar sumas y restas mucho más rápido.

Figura A.1

5) Confeccionar una tabla donde se muestren las características significativas de *hardware* y *software* en las distintas generaciones de las computadoras de acuerdo a los siguientes criterios:

Generación / Criterio	Primera	Segunda	Tercera	Cuarta	Quinta ¿? Investigar
Tecnología de cómputo					
Memoria Principal					
Memoria Auxiliar					
Dispositivos E/S					
Lenguaje de Programación					
Técnicas de explotación					
Otros Avances de interés					

Figura A.2

6) ¿Qué conceptos propuso el Dr. Von Newman para la construcción de una computadora (ordenador)?

7) En relación a la pregunta anterior, ¿qué problemas solucionaban los conceptos propuestos por Von Newman?

TRABAJO PRÁCTICO SISTEMAS DE NUMERACIÓN CAPÍTULO 3

Ejemplos del desarrollo de los ejercicios

I) Realizar la expansión de un número:

La expansión se realiza efectuando la suma de cada dígito por su valor posicional.

Ejemplo: 3457

$3457 = 3x10^3 + 4x10^2 + 5x10^1 + 7x10^0 = 3000 + 400 + 50 + 7$

II) Conversión de números a distintas bases.

a) Convertir números en el sistema decimal a números en el sistema binario.

Se divide el numeral decimal sucesivamente por 2 hasta llegar al cociente 0. El numeral binario se obtiene con el último resto 1 al que se agregan todos los restos de las divisiones (de atrás para delante).

Pasar $(47)_{10}$ a base 2:

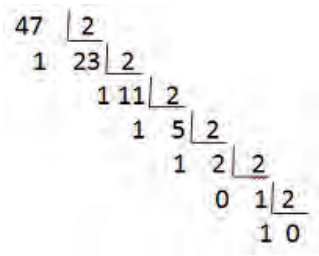


Figura B.1

El binario correspondiente a $(47)_{10}$ es $(101111)_2$

Si se tratara de un número decimal, se consideran en forma separada la parte entera de la fraccionaria. $(27,125)_{10}$ a base 2

Parte entera

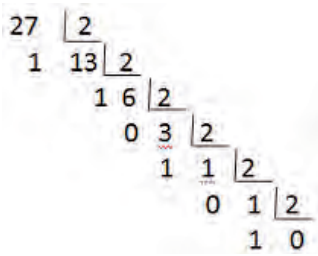


Figura B.2

Parte fraccionaria

$0.125 \times 2 \text{ —- } 0.25$
$0.25 \times 2 \text{ —- } 0.5$
$0.5 \times 2 \text{ —- } 1$
$(27,125)_{10} = (11011,001)_2$

b) Convertir números del sistema decimal a una base cualquiera.

Se divide el numeral decimal, sucesivamente por la base, hasta llegar al cociente 0.

Ejemplo: $(11,30)_{10}$ a base 3

Parte entera

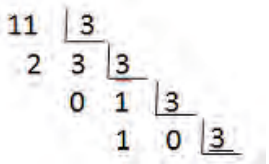


Figura B.3

Parte fraccionaria

$0.30 \times 3 \text{ —- } 0.90$
$0.90 \times 3 \text{ —- } 2.70$
$0.70 \times 3 \text{ —- } 2.10$
$0.10 \times 3 \text{ —- } 0.30$
$(11,30)_{10} = (102,022)_3$

Ejemplo: $(26,15)_{10}$ a base 16

Parte entera

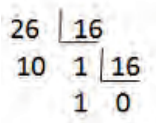


Figura B.4

Parte fraccionaria

0.15 x 16 —- 2.4
0.4 x 16 —- 6.4
0.4 x 16 —- 6.4
$(26,15)_{10} = (1A,266)_{16}$

c) Convertir a decimal numerales expresados en base 2.

Se realiza el desarrollo en potencias de 2. Como el número tiene siete dígitos, el primer dígito tendrá valor posicional 2 a la sexta.

Ejemplo: $(1010101)_2 = (?)_{10}$

$(1010101)_2 = 1x2^6 + 0x2^5 + 1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 1x2^0$

$(1010101)_2 = 64 + 0 + 16 + 0 + 4 + 0 + 1$

$(1010101)_2 = (85)_{10}$

d) Convertir numerales de distintas bases, al sistema decimal.

Se realiza la expansión, es decir, la sumatoria de los productos de los dígitos del número por las sucesivas potencias de la base r_1, y luego, las operaciones correspondientes en base 10.

Ejemplo: $(21403)_5$

Aquí la base r_1=5, por eso usamos potencias de 5 para escribir el desarrollo en potencias del número. Como el número tiene cinco dígitos, el primer dígito tendrá valor posicional 5 a la cuarta.

$(21403)_5 = 2x5^4 + 1x5^3 + 4x5^2 + 0x5^1 + 3x5^0$

$(21403)_5 = 1250 + 125 + 100 + 0 + 3$

$(21403)_5 = (1478)_{10}$

e) Convertir numerales de distintas bases, pasando por base 10.

Ejemplo: $(32,12)_{12} = (?)_4$

En primer lugar, se realiza el desarrollo en potencias del número, para pasar de base 12 a base 10.

$(32,12)_{12} = 3x12^1 + 2x12^0 + 1x12^{-1} + 2x12^{-2} = (38,097)_{10}$

Se realizan sucesivas divisiones del número en base 10, por la base 4 a la que se quiere pasar.

Parte entera

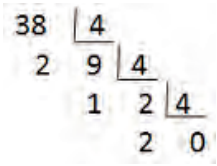


Figura B.4

Parte fraccionaria

0.097 x 4 —- 0.388
0.388 x 4 —- 1.552
0.552 x 4 —- 2.208
$(32,12)_{12} = (212,012)_4$

f) Convertir números de distintas bases, donde una de las bases es potencia entera de la otra.

Ejemplo 1: convertir el binario 1111001011,10011 en sus equivalentes octal y hexadecimal.

1) De base 2 a base 8; en este caso 8 = 23.

Se distribuye el número binario en grupos de tres bits procediendo hacia la izquierda y derecha del punto binario; luego, se reemplaza cada grupo de tres bits por su dígito octal equivalente.

001	111	001	011,	100	110
1	7	1	3,	4	6

$(1111001011,10011)_2 = (1713,46)_8$

2) De base 2 a base 16; en este caso $16 = 2^4$.

Se divide el número binario en paquetes de cuatro bits y se emplean los dígitos hexadecimales correspondientes.

0011	1100	1011,	1001	1000
3	C	B,	9	8

Ejemplo 2: $(2312003)_4 = (?)_{16}$; en este caso $16 = 4^2$.

Se divide el número en paquetes de dos dígitos y se reemplazan por sus equivalentes en la base mayor. (4 y 1 son los pesos de la base 4)

0 2	3 1	2 0	0 3
0x4+2x1	3x4+1x1	2x4+0x1	0x4+3x1
2	D	8	3

$(2312003)_4 = (2D83)_{16}$

Ejemplo 3: $(E1A)_{16} = (?)_2$; en este caso $16 = 2^4$.

Se reemplaza cada dígito de la base mayor (16) por su equivalente en la base menor (2).

E	1	A
8 4 2 1	8 4 2 1	8 4 2 1
1 1 1 0	0 0 0 1	1 0 1 0

Ejercicios:

I) Composición de numerales.

a) Componga los siguientes numerales:

1) $3x10^1 + 7x1^4 + 2x10^2 =$

2) $6x10^3 + 5x10^1 + 5x10^5 =$

3) $3x10^{-}2 + 8x10^4 + 3x10^0 + 6x10^{-}3 + 7x10^1 =$

4) $1x10 - 1 + 5x10^{-}3 + 2x10^{-}5 + 3x10^0 + 3x10^{-}2 + 7x10^{-}4 + 6x10^{-}6 =$

b) Descomponga los siguientes numerales:

1) 5314 =
2) 829,0029 =
3) 3012,18 =
4) 5200,3511 =

II) Sistemas de numeración

a) Construya por su propio medio un sistema de base 5 con sus equivalencias del 1 al 15 decimal.

b) Haga lo mismo con un sistema de base 8. Establezca sus equivalencias hasta el 30 decimal.

c) Construya las tablas de sumar y multiplicar de ambos sistemas.

III) Conversión de números a distintas bases.

a) Transformar c/u de los siguientes números decimales a números en el sistema binario.

1) 10	4) 32,12	7) 0,000123	10) 378,017
2) 112	5) 0,25	8) 32,501	11) 14,0123
3) 45	6) 31,127	9) 32102,1	12) 3281,0346

b) Convierta a decimal los siguientes numerales expresados en base binaria.

1) 10110	4) 0,11101	7) 100111,1111
2) 111101	5) 111,1010	8) 0,0001110
3) 101,10	6) 11001,0001	9) 1111,110011

c) Convierta a decimal los siguientes numerales expresados en las bases indicadas.

1) $(1022)_3$	4) $(A9A, 731)_{12}$	7) $(842, 4217)_9$
2) $(6612)_7$	5) $(A24E, BC)_{16}$	8) $(21, 212)_4$
3) $(1541)_8$	6) $(0, 4452)_6$	9) $(F6, 110)_{16}$

d) Convierta a las bases indicadas utilizando el método más adecuado. En caso de que una base sea potencia entera de la otra, aplicar la metodología estudiada.

1) $(1202)_3 = (?)_7$
2) $(384)_9 = (?)_2$
3) $(3143,011)_5 = (?)_3$
4) $(110010101011)_2 = (?)_4$
5) $(C92,F3)_{16} = (?)_4$
6) $(8466,143)_9 = (?)_3$
7) $(101100111,11100)_2 = (?)_8$
8) $(2131,1022)_4 = (?)_2$
9) $(10101011,1000111)_2 = (?)_{16}$
10) $(5301)_6 = (?)_5$
11) $(787,425)_{16} = (?)_2$
12) $(4231)_5 = (?)_{12}$
13) $(2131,012)_4 = (?)_{16}$
14) $(1110110,00011)_2 = (?)_8$

TRABAJO PRÁCTICO OPERACIONES ARITMÉTICAS TB CAPÍTULO 3

Ejemplos del desarrollo de los Ejercicios

I) Suma binaria

Sumar dos números binarios 1110,11 + 110,01.

$$\begin{array}{r} 1110{,}11 \\ +110{,}01 \\ \hline 10101{,}00 \end{array}$$

II) Suma en otra base

Sumar dos números binarios $A92,3_{12}$ + $73,B_{12}$.

$$\begin{array}{r} A92{,}3 \\ +73{,}B \\ \hline B46{,}2 \end{array}$$

III) Multiplicación

La multiplicación en binaria es muy sencilla puesto que se realiza por 0 y 1. Hay que tener cuidado con otras bases.

$(1100)_2$ x $(1010)_2$ y $(74,2)_8$ x $(51)_8$

$$\begin{array}{r} 1\,1\,0\,0 \\ \times\ 1\,0\,1\,0 \\ \hline 0\,0\,0\,0 \\ 1\,1\,0\,0\ - \\ 0\,0\,0\,0\ -\ - \\ 1\,1\,0\,0\ -\ -\ - \\ \hline 1\,1\,1\,1\,0\,0\,0 \end{array} \qquad \begin{array}{r} 7\,4,\,2 \\ \times\ \ \ 5\,1 \\ \hline 7\,4\,2 \\ 4\,5\,5\,2\ - \\ \hline 4\,6\,4\,6,\,2 \end{array}$$

Figura C.1

IV) Resta binaria

Restar dos números binarios 11101 - 1011

$$\begin{array}{r} 11101 \\ -1011 \\ \hline 10010 \end{array}$$

V) Resta en otra base

$(3021)_4 + (1231)_4$

$$\begin{array}{r} 3021 \\ -\ 1231 \\ \hline 1130 \end{array}$$

VI) Complementos auténtico y restringido

Complementar: $(7461)_9$

$$\begin{array}{rcl} A & = & 8888 \\ - & & \\ B & = & 7461 \\ \bar{B} & = & 1427 \\ + & & 1 \\ B' & = & 1428 \end{array}$$

VII) Resta usando complemento

Restar $(A46)_{16} - (123)_{16}$

$$\begin{array}{rcl} A & = & A46 \\ + & & \\ \bar{B} & = & EDC \\ A+\bar{B} & = & 1\,1922 \\ + & & 1 \\ A+B' & = & 1\,1923 \end{array}$$

Restar $(302)_5$ - $(432)_{16}$

$$\begin{array}{rcl} A & = & 302 \\ & + & \\ \bar{B} & = & 012 \\ A+\bar{B} & = & 0314 \end{array}$$

Como no hubo acarreo, complemento el resultado y es negativo.

$$\begin{array}{rcl} & = & 444 \\ & - & \\ d & = & 314 \\ \bar{d} & = & -130 \end{array}$$

Ejercicios

I) Efectúe las siguientes operaciones en las bases indicadas. Verifique convirtiendo a decimal.

a) $(110111, 01101)_2$ + $(11101, 111)_2$ =
b) $(9AF, 1E)_{16}$ + $(AE, 123)_{16}$ =
c) $(131, 16)_7$ - $(63, 01)_7$ =
d) $(4321, 12)_5$ - $(4432, 3)_5$ =
e) $(2210, 12)_3$ - $(1011, 01)_3$ =
f) $(10111, 1)_2$ + $(11011, 01)_2$ =
g) $(1010)_2$ - $(111, 11)_2$ =
h) $(4324, 131)_5$ + $(3222, 244)_5$ =
i) $(762, 123)_8$ + $(247, 763)_8$ =
j) $(111011, 11)_2$ x $(1101, 1)_2$ =
k) $(11011)_2$ x $(1001, 11)_2$ =
l) $(523, 22)_6$ x $(410, 02)_6$ =
m) $(1273, 43)_8$ x $(6121, 03)_8$ =

II) Hallar los complementos restringidos y auténticos de:

a) $(6478)_{10}$
b) $(101101110)_{2}$
c) $(AE213)_{16}$
d) $(12231)_{4}$
e) $(44003)_{5}$

III) Restar utilizando complemento:

a) $(1101101)_{2} - (11101)_{2} =$
b) $(5231)_{6} - (4201)_{6} =$
c) $(FB193)_{16} - (FA67)_{16} =$
d) $(11011)_{2} - (111001)_{2} =$
e) $(3271)_{8} - (22012)_{8} =$
f) $(F163)_{16} - (19583)_{16} =$
g) $(A157)_{12} - (3974)_{12} =$
h) $(200021)_{3} - (1120)_{3} =$

TRABAJO PRÁCTICO REPRESENTACIÓN DE DATOS CAPÍTULO 4

En el capítulo correspondiente se vieron ejemplos de cada ejercicio.

Ejercicios

I) Represente los siguientes caracteres en EBCDIC y ASCII-7, indique las cadenas binarias y en hexadecimal.

a) AazZ
b) 703E
c) Wvkp
d) P235

II) Dadas las siguientes representaciones hexadecimales:

a) indique qué representan en ASCII-7.
1) 6143
2) 44E9
3) 3F
b) indique qué representan en código EBCDIC:
1) B1C8
2) 19F8
3) C4

III) Suponiendo una longitud de palabra de 16 bits, codifique en los formatos de punto fijo estudiados, los siguientes números:

a) +268
b) -175
c) -54

IV) Para cada sistema de representación de enteros, utilizando longitud de palabra de 16 bits, ¿cuáles son el máximo y el mínimo número en decimal que se puede representar?

V) Hallar el número decimal que representa cada una de las siguientes cadenas hexadecimales, según el formato de punto flotante de IBM 360 y de PDP 11:

a) $(E56A41D2)_{16}$
b) $(223FB5C1)_{16}$
c) $(8630C684)_{16}$
d) $(DA13C879)_{16}$
e) $(43616ADF)_{16}$

VI) Expresar en ambas notaciones de punto flotante los siguientes números decimales:

a) $2,51x10^{-4}$
b) $-16,23x10^{18}$
c) 525
d) $723,41x10^{-28}$
e) $-124,88x10^{-1}$

VII) Suponga que la cifra 9025 ingresa a una computadora.

a) Escriba la representación hexadecimal en los códigos EBCDIC y ASCII.
b) Escriba la representación hexadecimal de la misma cifra en BCD empaquetado, según provenga de ambos códigos.
c) Escriba la representación hexadecimal de la cifra en un registro de 2 bytes.

TRABAJO PRÁCTICO COMPONENTES CAPÍTULO 5

1) Defina *hardware*.
2) ¿Cuál es la principal función del procesador?
3) Mencione las principales características de la Unidad de Control.
4) Identifique al menos cuatro registros de la UC e indique su función.
5) Explique cómo es el ciclo de ejecución de una instrucción.
6) Esquematice los semiciclos de búsqueda y ejecución de una instrucción.
7) ¿Qué diferencia hay entre la MBR y la MAR?
8) Enumere y explicite los tipos de buses.
9) ¿Qué es la ALU?
10) ¿Cuáles son los principales registros de la ALU?

TRABAJO PRÁCTICO - MEMORIA PRINCIPAL - CAPÍTULO 6

1) Caracterice los distintos tipos de RAM.
2) Diferencias entre SRAM y DRAM. Tipos de DRAM.
3) Concepto de memoria caché. Tecnología y utilización.
4) Capacidades de las memorias RAM.
5) Qué significa que el Zócalo de Memoria sea DIMM o SIMM
6) Tipos de memoria ROM.
7) ¿Qué es la BIOS?
8) ¿Qué es el ChipSet?
9) ¿Qué es la memoria Swap?

TRABAJO PRÁCTICO MODOS DE DIRECC. CAPÍTULO 7

Explicación de los ejercicios

El propósito de este práctico es aplicar, mediante ejercicios sencillos, los conceptos relativos a los formatos de instrucciones y modos de direccionamiento. Del mismo modo abarca el tema de ciclo de instrucción, el cual es necesario dominar para calcular el tiempo de ejecución.

Lo que realizaremos será expresar mediante instrucciones de un pseudolenguaje de máquina, distintas expresiones algebraicas, aplicando cuatro formatos de instrucciones posibles.

Luego de hallar el conjunto de instrucciones que resuelven la expresión en cuestión, calcularemos la cantidad de accesos a memoria necesarios para leer/escribir los datos durante la ejecución de dicho "segmento de programa".

Ejemplo:

Utilizaremos la siguiente expresión algebraica:

$$A = B \times C - X$$

Las letras u operandos de esta expresión, llamadas variables algebraicas, también son llamadas variables en programación, solo que en este caso, en lugar de representar un valor cualquiera, representan una dirección de memoria en la cual se almacena un valor. En una instrucción en lenguaje de máquina donde el operando se indica con una dirección de memoria, se puede simbolizar con una variable de este tipo y hablar, entonces, de dirección simbólica.

En estos ejercicios no se hablará del tipo de valores que se almacenan, sean estos enteros o reales.

Nota 1: Para poder descomponer adecuadamente cada expresión se deben tener en cuenta las reglas de precedencia y asociatividad de los operadores.

Los formatos de instrucción que utilizaremos son cuatro: de tres operandos explícitos, de dos operandos explícitos, de un operando explícito, y de cero operando explícito. Cuando hablamos de operando explícito nos referimos al operando que acompaña al código de operación dentro del formato de la instrucción.

Pasos para realizar el ejercicio:
1) Según las reglas de precedencia de operadores, debemos realizar la siguiente secuencia de operaciones elementales:

A = B x C - X

1- producto.
2- substracción.
3- asignación.

2) Debido a que trabajamos con un pseudolenguaje de máquina, se puede suponer la existencia de los códigos de operación necesarios, en general uno para cada operación aritmética y los necesarios para las transferencias y movimientos de datos.
3) Según cada formato de instrucción con el que se desarrolle la expresión, considerar la forma de almacenar los resultados intermedios, creando para ello variables auxiliares, y escribir las instrucciones según cada formato:

Formatos de Instrucción
A. Para tres operandos explícitos: Los dos primeros operandos son fuentes, y el tercero es destino.

MUL B C AUX1 $M[AUX1] \leftarrow M[B] \times M[C]$
RES AUX1 X A $[A] \leftarrow M[AUX1] - M[X]$

B. Para dos operandos explícitos: los dos operandos son fuentes y el segundo es, además, destino del resultado.

MOV C AUX1 $M[AUX1] \leftarrow M[C]$
MUL B AUX1 $M[AUX1] \leftarrow M[B] \times M[AUX1]$

MOV X AUX2 $M[AUX2] \leftarrow M[X]$
RES AUX1 AUX2 $M[AUX2] \leftarrow M[AUX1] - M[AUX2]$
MOV AUX2 A $M[A] \leftarrow M[AUX2]$

En este caso conviene utilizar operandos auxiliares para resguardar los valores originales de ciertas variables en caso de ser necesario, para ello utilizamos la operación mover (MOV).

C. Para un operando explícito: la operación individual, que involucra siempre dos operandos, se debe realizar utilizando un registro interno implícito llamado acumulador.

TRA X $Acc \leftarrow M[X]$
MUL B $Acc \leftarrow Acc \times M[B]$
RES X $Acc \leftarrow Acc - M[X]$
TRM A $M[A] \leftarrow Acc$

Se utilizan dos operaciones para cargar y descargar el acumulador: "TRA var" transfiere el contenido de var al acumulador y "TRM var" transfiere el valor del acumulador a la variable indicada por var.

D. Para cero operando explícito: los operandos involucrados en una expresión aritmética, deben ser almacenados previamente en una pila interna, propia a la UAL del procesador. La instrucción que ejecuta la operación no hace referencia a ningún operando explícitamente, sino que hace referencia implícita a los dos últimos valores "apilados" en la pila. Luego de ejecutarse la operación, dichos valores desaparecen y, en cambio, queda como tope de la pila, el resultado de la operación.

TRP X $Stack[SP - 1] M[X], SP \leftarrow SP + 1$
TRP C $Stack[SP - 1] \leftarrow M[C], SP \leftarrow SP + 1$
TRP B $Stack[SP - 1] \leftarrow M[B], SP \leftarrow SP + 1$
MUL $Stack[SP - 1] \leftarrow Stack[SP] \times Stack[SP - 1], SP \leftarrow SP - 1$
RES $Stack[SP - 1] \leftarrow Stack[SP] - Stack[SP - 1], SP \leftarrow SP - 1$
TRM A $M[A] \leftarrow Stack[SP], SP \leftarrow SP - 1$

Se utiliza una operación de transferencia de la memoria al tope de la pila, también llamada PUSH, que es "TRP var", donde el contenido de var se pone en el tope de la pila. En este caso la instrucción "TRM var", transfiere el tope de la pila a la dirección de memoria var, operación también llamada POP.

Ejercicios

I) Dadas las siguientes expresiones algebraicas, escribir las instrucciones de un pseudolenguaje de máquina que las resuelvan, para los formatos de instrucción pedidos:

1) A = (X + D - 5) * ((A - X) / H + 2) + C * 10
2) B = (A + B / 3) / 5 + (B + 4 - C) * (X + B) + C
3) A = X x A + B / (C - B + 1)
4) Y = C + (Z x A + Z / A) / B
5) RES = Z + 2 x A + B - 10 / X
6) A = (Y + 5) * (Z - 3) + ((X – 4 + 9) / (X – 3)) + (Y * 5 – 3)
7) B = (A + B – 1) / (A * C / 3) + ((9 * 8) – (C / 6))
8) F = 10 / ((X / 2) / 20 – (Q * 3) / (M – 15))

a) Máquina de tres operandos explícitos.
b) Máquina de dos operandos explícitos.
c) Máquina de un operando explícito.
d) Máquina de cero operando explícito.

II) Calcular la cantidad de accesos a memoria para recuperar datos, en cada uno de los ejercicios anteriores.

TRABAJO PRÁCTICO MEMORIA AUXILIAR CAPÍTULO 8

1) Explicar la organización física de un disco magnético y, de acuerdo a ello, cómo se calcula su capacidad.
2) Describa la tecnología de grabación de memorias ópticas.
3) Qué diferencia puede explicar entre las tecnologías ópticas y las magnéticas.
4) Realice un cuadro comparativo de los distintos tipos de discos ópticos, describiendo sus principales características.
5) Qué significan FAT y NTFS para el manejo de archivos.
6) Describa distintos tipos de memoria flash y sus características.
7) Qué son las flash cards.
8) Funcionamiento de una memoria USB. Componentes.

TRABAJO PRÁCTICO - PERIFÉRICOS CAPÍTULO 9

1) Complete el cuadro especificando, al menos, siete periféricos

PERIFÉRICO	ENTRADA - SALIDA	INTERNO - EXTERNO

2) ¿Qué es un periférico de entrada? ¿y de salida?

3) ¿Existen periféricos de entrada/salida? Ejemplifique.

4) Enumere, y especifique, cuatro tipos de periféricos de entrada.

5) Enumere, y especifique, cuatro tipos de periféricos de salida.

6) Enumere, y especifique, cuatro tipos de periféricos de entrada/salida o mixtos.

TRABAJO PRÁCTICO SISTEMAS OPERATIVOS CAPÍTULO 10

1) ¿Qué es un sistema operativo?
2) Indique las misiones y funciones de un sistema operativo.
3) ¿Qué es el procesamiento por lotes?
4) Defina qué es la multiprogramación.
5) ¿Qué ventajas supuso la multiprogramación?
6) Explique el modelo de Dijkstra.
7) Esquematice y explique cada capa del modelo de Dijkstra.
8) Mencione las diferencias entre un sistema operativo monousuario y multiusuario.
9) Mencione las diferencias entre un sistema operativo monotarea y multitarea.
10) Mencione las diferencias entre un sistema operativo monoproceso y multiproceso.
11) ¿Qué entiende por máquina virtual?
12) De los modelos de gestión de la memoria, ¿cuáles cree que son los que mejor la administran?

TRABAJO PRÁCTICO INGENIERÍA DE *SOFTWARE* CAPÍTULO 11

1) Intente definir, a través de sus conocimientos, qué es para usted la Ingeniería de *Software* y qué incumbencias debería tener un ingeniero en *software*.
2) Defina qué es la Ingeniería de *Software*. Compare su respuesta con la definición brindada en el primer punto.
3) ¿Qué conocimientos debería tener un ingeniero de *software*? Discuta su definición con la expresada en el primer punto.
4) Desde el punto de vista de la Ingeniería del *software*, defina Proceso y Producto.
5) Crisis del *software*: causas, consecuencias, conclusiones.
6) Dé ejemplos de desarrollos para cada área de aplicación. Mencione dos o más áreas de aplicación.
7) Mencione casos reales en que defectos de *software* hayan causado la pérdida de vidas humanas y/o económicas.
8) ¿Qué es un paradigma en Ingeniería de *Software*?
9) ¿Cuáles son las faces que debe tener cualquier paradigma de *software* y qué debe realizarse en cada una de ellas?
10) Ventajas y desventajas del modelo en cascada.
11) Explique el modelo en espiral.

TRABAJO PRÁCTICO LENGUAJES DE PROGRAMACIÓN CAPÍTULO 12

1. ¿Qué es un programa de computadora?
2. ¿Cuáles son las dos fases al escribir un programa?
3. ¿Un algoritmo es lo mismo que un programa? Justifique su respuesta.
4. ¿Cuáles son las ventajas de utilizar un lenguaje de alto nivel?
5. Dé dos ejemplos de tareas cotidianas que involucren secuencias lógicas.
6. ¿Quién es el responsable en el éxito de un programa?
7. Describa el esquema de algoritmo genérico.
8. Escriba un algoritmo sencillo que no ocupe más de 10 pasos. Describa el contexto y las consideraciones previas.

TRABAJO PRÁCTICO PROGRAMACIÓN DE SISTEMAS CAPÍTULO 13

1. ¿Por qué no se usa el lenguaje natural para programar computadoras?
2 ¿Describa qué es el lenguaje de máquina?
3. ¿Todas las computadoras tienen el mismo lenguaje de máquina? Dar ejemplos.
4. Describa el lenguaje ensamblador.
5. ¿Qué hace un programa traductor?
6. ¿Qué diferencia existe entre un compilador y un ensamblador?
7. Grafique los niveles de abstracción de los lenguajes.
8. Dé un ejemplo de algoritmo donde se aplique alguna de las estructuras básicas de la programación.
9. Investigar qué es un programa intérprete. Indique ventajas y desventajas
10. ¿Qué es un programa cargador? ¿Conoce alguno?

TRABAJO PRÁCTICO REDES CAPÍTULO 14

1) ¿Cómo se clasifican las redes de computadoras? Describa cada una de ellas.
2) ¿Qué es una red inalámbrica?
3) A qué se refiere la "Topología de una red".
4) ¿Qué tipo de topologías de red existen? Esquematice y describa.
5) ¿Qué es Internet?
6) ¿Qué diferencia existe entre Internet e Intranet?
7) Describa, al menos, cuatro medios de acceso.
8) Compare los modelos de referencia ISO y TCP-IP.
9. ¿Qué capas del modelo TCP-IP agrupan las del modelo ISO?
10) ¿Cuál de los dos modelo se impuso y por qué?

TRABAJO PRÁCTICO IC 2 - ANEXO A

1- Dados los siguientes códigos hexadecimales en memoria:

9000: A0A0 0000 0000 0000 4102 9000 56C0 56C1
9008: 09A0 53C1 53C2 7812 0000 A300 9008 4240
9010: 9001 4241 9002 50C1 4240 9003 F100
.....
.....
A09E: 01C2 B154 0050 0121 0279 0016 0000 01B1
A0A6: 0000 514C

Y considerando que:

- El programa/subrutina realiza el cálculo del promedio de los valores que se encuentran a partir del contenido de la dirección COMIENZA hasta encontrar un valor 0, almacenando el promedio en PROMEDIO.
- El contador de programa al comenzar la ejecución contiene el valor 9004
- Las etiquetas a utilizar son las siguientes:

  9000 COMIENZA
  9001 SUMATORIA
  9002 CONTADOR
  9003 PROMEDIO

a) Describir esta rutina en código mnemónico utilizando la planilla del ensamblador estudiada.
b) Realizar su seguimiento (trazado), confeccionando una planilla, indicando en cada paso el estado del PC, los registros y los flags del procesador y las direcciones de memoria utilizadas.

2- Dados los siguientes códigos hexadecimales en memoria:

9000:	4110	0000	4102	A000	4111	A001	0990	54C2
9008:	53C1	7812	0000	A300	9006	4240	A001	F100
.....								
.....								
A000:	0004	010A	0018	0024	0003			

Y considerando que:

- El programa/subrutina realiza la suma de una serie de números almacenados en la memoria, cuyo resultado se guarda en la dirección INICIO.
- El contador de programa al comenzar la ejecución contiene el valor 9000.
- Las etiquetas a utilizar son las siguientes:

  9006 **SUMA**
  A000 **INICIO**
  A001 **INDICE**

  a) Describir esta rutina en código mnemónico utilizando la planilla del ensamblador estudiada.
  b) Realizar su seguimiento (trazado), confeccionando una planilla, indicando en cada paso el estado de los registros que se utilicen, hasta terminar la ejecución del programa

3-Dados los siguientes códigos hexadecimales en memoria:

9000:	A000	A004	A008	0004	4101	9000	4102	9001
9008:	56C0	7840	9003	A000	9019	0193	09A3	4D04
9010:	9002	023C	53C1	53C2	5380	9002	53C0	A100
9018:	9009	F100						
.....								
A000:	0F1B	0048	CB1D	79A1	A002	FF20	21BE	63AA
A008:	0000	0000	0000	0000				

Y considerando que:

- El programa/subrutina realiza el cálculo de la suma de los elementos de dos vectores que se encuentran a partir de los contenidos de las direcciones V1, V2, almacenando los resultados a partir del contenido de la

dirección VR. El contenido de la dirección CANTIDAD establece la cantidad de elementos de los vectores.

- El contador de programa al comenzar la ejecución contiene el valor 9004.
- Las etiquetas a utilizar son las que se detallan a continuación:

9000: **V1**
9001: **V2**
9002: **VR**
9003: **CANTIDAD**

a) Describir esta rutina en código mnemónico utilizando la planilla del ensamblador estudiada.
b) Realizar su seguimiento (trazado), confeccionando una planilla, indicando en cada paso el estado de los registros que se utilicen, hasta terminar la ejecución del programa.

4- Dados los siguientes códigos hexadecimales en memoria:

9000:	0004	0000	4103	9000	4110	0001	7813	0000
9008:	A000	900E	4FC3	54C3	A100	9006	4240	9001
9010:	F100							

Y considerando que:

- El programa/subrutina realiza el cálculo del factorial del número almacenado en la dirección NUMERO y lo almacena en la dirección FACTORIAL.
- El contador de programa al comenzar la ejecución contiene el valor 9002.
- Las etiquetas a utilizar son las siguientes:

9000: **NUMERO**
9001: **FACTORIAL**

a) Describir esta rutina en código mnemónico utilizando la planilla del ensamblador estudiada.
b) Realizar su seguimiento (trazado), confeccionando una planilla, indicando en cada paso el estado de los registros que se utilicen, hasta terminar la ejecución del programa.

TRABAJO PRÁCTICO PROGRAMACIÓN EN SIMULPROC ANEXO B

I) Para las siguientes consignas:
Hacer una breve descripción del algoritmo que resuelve el problema y, luego, codificarlo en lenguaje ensamblador para el procesador virtual. Una vez probado el código, entregar una versión impresa en código mnemónico indicando en qué dirección de memoria se carga cada instrucción o dato (editor 2 de SimulProc).

II) Una vez resuelto, codificado y probado el problema de cada ejercicio, realizar la traza de cada programa con un juego de datos suficiente para verificar su funcionamiento.

a) Sumar tres números almacenados en memoria. Los valores se encuentran a partir de la posición 010x (convención usada para expresar un número en hexadecimal) de memoria.

b) Ingresardos2 números, por teclado, A y B, si el primero es mayor que el segundo, realizar una suma; en caso contrario, realizar una resta.

c) Calcular el promedio de cuatro números en memoria. Los valores se encuentran a partir de la posición 020x de memoria.

d) Calcular el factorial de N, ingresado por teclado.

e) Realizar la resta de dos números ingresados por teclado usando el método de "Resta por complemento".

f) Escribir el código correcto para que el procesador permita ingresar una serie de números (menos el 1) hasta que se ingrese el número 1. Al finalizar, informe la cantidad de número ingresados.

g) Interprete el fragmento de programa indicado más abajo. Explique qué hace y calcule la salida del programa, si por teclado se le cargan los números 2 y 4, en ese orden, en base 10.

```
000 MSG ‘que hago’
001 LDT ‘ing un nro’
002 MOV 010,AX
003 LDT ‘ing un nro’
004 MOV CX,AX
005 LDA 015
006 DEC AX
007 MUL 010
008 LOOP 007
009 EAP
010 HLT
#015
10
```

h) Interprete el fragmento de programa indicado más abajo. Explique qué hace y calcule la salida del programa, si por teclado se le carga el número $(31)_{10.}$ Recuerde que el procesador trata los valores ingresados en sistema binario.MSG “2 Parcial. Ejemplo 1”

```
CLA
MOV BX,AX
LDT “Ingrese dato entero positivo”
MOV CX,15
ROL AX,1
JNC 8
INC BX
LOOP 5
XAB
EAP “Resultado del proceso de programa:”
#015
10000; nro 16 representado en binario
```

BIBLIOGRAFÍA

Albarracín, M., E. Alcalde Lancharro. *Introducción a la Informática.* Madrid Mc. Graw Hill, 1994

Beekman, George. *Introducción a la Informática*. España Pearson, 2005

Brauder, E. *Ingeniería de Software: Una perspectiva orientada a objetos. (Spanish Edition)* España, Ra-Ma, 2003

Carretero Pérez, J. *Sistemas operativos una visión aplicada.* Mexico, Mc Graw Hill, 2001

Ginzburg, Mario Carlos. *Introducción a las técnicas digitales con circuitos integrados.* España: Ediciones del Autor, 2006

Joyanes Aguilar, Luis. *Fundamentos de programación.* España, McGraw-Hill, 2008

Kenneth Louden, *Lenguajes de programación*. Mexico McGraw-Hill, 2004

Levine Gutiérrez, Guillermo. *Introducción a la computación*. Mexico, McGraw Hill, 1990

Milenkovic, Milán. *Sistemas operativos: conceptos y diseño.* España, McGraw-Hill, 1995

Pfleeger, Shari. *Ingeniería de software.* Argentina, Prentice Hall, 2002

Pressman, Roger. *Ingeniería de Software un enfoque estructurado.* 7.ma edición, México, 2011.

Presser, Cárdenas y Marín. *Ciencias de la Computación*, España, Limusa Wiley, 1972

Sommerville, Ian *Ingeniería de software.* 10.ma edición, Addison-Wesley , 2017.

Stallings, William. *Redes y transmisión de datos*. Mexico, McGraw-Hill, 2004

Stallings, William. *Sistemas operativos*, Mexico Prentice Hall, 2005

Tanenbaum, Andrew S. *Organización de computadoras: un enfoque estructurado.* 6.ta edición: Pearson Educación Prentice Hall Addison Wesley, 2012.

Tanenbaum, Andrew. *Redes de computadoras*, España, Pearson Prentice Hall, 2004

A continuación se citan algunos enlacen para ampliar:

www.euro-ix.net

hhttps://foro.movistar.com.ar/

wwww.fundacionsadosky.org.ar/

hhttps://www.ieee.org/

hhttp://redestelematicas.com/arquitectura-de-internet/

El presente material surge, en primera instancia, para dar respuesta a la cátedra de Elementos de Informática de la UNTDF. Que tiene como objetivo fundamental posibilitar que lxs estudiantes dispongan de un conjunto de conocimientos globales de las Ciencias de la Computación, valerse de herramientas procedimentales y conceptuales, a fin de poder lograr su desarrollo como profesionales en el área de sistemas.

Ahora bien, su anclaje pedagógico y su forma de abordar las distintas temáticas permiten que cualquier lectorx pueda ampliar sus saberes en la materia incorporando conceptos tales como: fundamentos, misiones y funciones que debe tener un Sistema Operativo, como así también los principios básicos y el significado de los lenguajes de programación; entre otros.

UNTDF
Universidad Nacional de Tierra del Fuego, Antártida e Islas del Atlántico Sur

LUA Libro Universitario Argentino

ISBN 978-987-4680-47-7

9 789874 680747